AD-A233 030

:NTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 01, 1990	Final Report for Period 01 Nov. 88 - 31 Oct. 90

4. TITLE AND SUBTITLE
"Hierarchical Neural Network (HNN) For Closed Loop Decision Making."
Subtitle: "Designing the Architecture of a Hierarchical Neural Network to Model Attention, Learning and Goal Oriented Behavior."

5. FUNDING NUMBERS
Grant # 890010

AFOSR-89-0010

6. AUTHOR(S)
Allon Guez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ECE Department,
Drexel University,
Philadelphia, PA 19104

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 91 0136

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR / NE
Bldg 410
Bolling AFB, DC 20332-6448
Capt Suddarth

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
2305B3

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The objectives of the project were to design and evaluate a hierarchical neural network (HNN) capable of real time learning and decision making in closed loop.

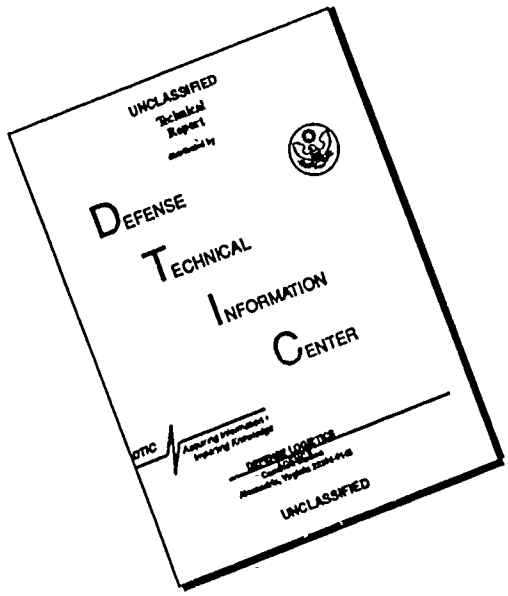
In the initial stages of the project the problem was defined and the relating state of the art methods were surveyed. Later control of a robotiic system was used as the prototypical task and a HNN was designed and compared with the state of the art adaptive control techniques.

During this project the concept of exploratory schedules (ES) was developed. ES is defined as system trajectories internally generated by the HNN for the purpose of efficient learning. This concept was implemented in an open-loop fashion for the control of robotic manipulators. A theorem was proved that gives constructive conditions for stable learing in closed loop. This technique yielded improved transients in tracking desired trajectories in comparison with adaptive control methods. HNN architecture was applied as a controller for a class of nonlinear systems linear in control. It was shown to have guaranteed asymptotic stability. HNN architecture was employed with partial success in areas of pattern recognition and control.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Learning, Neural Networks (NN), Control, Exploratory Schedules	10
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Report 2

HIERARCHICAL NEURAL NETWORK (HNN) FOR CLOSED LOOP DECISION MAKING

Designing the Architecture of a Hierarchical Neural Network to Model Attention, Learning and Goal Oriented Behavior

Allon Guez

ECE Department
Drexel University
Philadelphia, PA 19104

01 December 1990

Final Report for Period 01 November 88 - 31 October 90

Prepared for

AFOSR / NE

Hierarchical Neural Network (HNN) For Closed Loop Decision Making

Allon Guez

ECE Department
Drexel University
Philadelphia, PA 19104

Contents

This is the final report for the AFOSR grant #890010 which was granted for a period of two years starting on November 1988 and expiring on November 1990. It contributed to the publication of *26 !* articles of which *7* are archival and referred journals. For further details please refer to the annual report [AR1] and the attached publications.

Accomplishments

The main accomplishments of the project are summarized as follows:

- Presenting and demonstrating the new concept of exploratory schedules. ([1],[5],[7-12],[17],[22],[27-29]).

- Demostrating that even the preliminary version of HNN has benefits not available to presently employed conventional methods. ([1],[2],[5-7]).

- A new theorem states constructive conditions for stable learning in closed loop. ([5],[7]).

- Guaranteed stability in closed loop of HNN for a class of nonlinear systems linear in control [35-36].

- Application of the approach to several decision making problems: Robotics, Pattern Recognition, and Control. ([3-4],[7-9],[13-16],[18-21],[23-26],[30-34])

Summary

The objectives of the project were to design and evaluate a hierarchical neural network (HNN) capable of real time learning and decision making in closed loop.

In the initial stages of the project the problem was defined and the relating state of the art methods were surveyed. Later control of a robotiic system was used as the prototypical task and a HNN was designed and compared with the state of the art adaptive control techniques.

During this project the concept of exploratory schedules (ES) was developed [5],[7-12],[17],[22],[27-29]. ES is defined as system trajectories internally generated by the HNN for the purpose of efficient learning. This concept was implemented in an open-loop fashion for the control of robotic manipulators. A theorem was proved that gives constructive conditions for stable learing in closed loop [1-2],[5-7]. This technique yielded improved transients in tracking desired trajectories in comparison with adaptive control methods. HNN architecture was applied as a controller for a class of nonlinear systems linear in control. It was shown to have guaranteed asymptotic stability [35,36]. HNN architecture was employed with partial success in areas of pattern recognition and control [3,4,13,14,18-21,23-26],[30-34].

Publications

The following is a list of publications partially supported by AFOSR grant# 890010.

[1] Ahmad, Z, Selinsky, J., Guez, A. "Application of Neural Networks to Robotics ", Chapter in the book Current Perspectives on Neural Networks. (Submitted).

[2] Guez, A. " Neurocontrollers," Journal of Neural Networks. (Submitted).

[3] Kumar, S., Guez, A.," ART Based Adaptive Pole Placement For Nuerocontrol "Neural Network Journal. (Submitted.)

[4] Zaharakis, S., Guez, A., " Time Optimal Navigation via Slack Set Method,"IEEE Trans. on System Man and Cybernetics, vol. 20, No. 6, Nov./Dec. 1990, pp. 1396-1407.

[5] Guez, A., Selinsky J., "Design of Exploratory Schedules in Learning and Adaptive Robot Control," Journal of Adaptive Control and Signal Processing. (Accepted. Expected to Appear 9/90)

[6] Guez, A., Ahmad, Z. "Improving the Solution of the Inverse Kinematic Problem in Robotics Using Neural Networks" Journal of Neural Network Computing ,Vol. 1, No. 4, Spring 1990, pp. 21-32.

[7] Guez A., Selinsky J., " Neurocontroller Design Via Supervised and Unsupervised Learning" Journal of Intelligent and Robotic Systems,Vol. 2, 1989,pp. 307-335.

[8] Guez, A., Bar-kana, I., "Two Degrees of Freedom Neurocontroller" Neuronet 90, The Int'l Symposium on Neural Nets and Neurocomputing, Prague, Czechoslovakia, September 1990.

[9] Bar-Kana, I., Guez, A., "Two Degrees of Freedom Robot Adaptive Control," 29th IEEE Control and Decision Conference, Honolulu, Hawaii, December 1990.

[10] Guez, A. "Application of Neural Networks to Robotics ", Third International Symposium on Robotics and Manufacturing, B.C., Canada, July 1990. (Abstract).

[11] Guez, A., Bar-kana, I., "Two Degrees of Freedom Design of Learning / Adaptive Control" American Control Conference June 1990, San Diego, CA.

[12] Guez, A. "Application of Neural Networks to Robotics", 8th International Congress of Cybernetics and Systems, N.Y., N.Y., June 1990. (Abstract).

[13] Ahmad, Z., Guez, A., "On the Solution to the Inverse Kinematic Problem" 1990 IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, May 1990.

[14] Kumar, S., Guez, A., "Adaptive Pole Placement for Neurocontrol" International Joint Conference on Neural Networks 90, 1/90, Washington, D.C..

[15] Barkana, I., Guez, A. "Neuromorphic Computing Architecture for Adaptive Control" International Joint Conference on Neural Networks 90, 1/90, Washington, D.C..

[16] Guez, A., Selinsky, J. "A Neurocontroller with Guaranteed Performance for Rigid Robot" International Joint Conference on Neural Networks 90, 1/90, Washington, D.C..

[17] Guez, A., "Neurocontrollers," Scientific Computing and Automation Conference, Philadelphia, PA, October 1989. (Abstract, Oral Presentation)

[18] Kam, M., Naim, A., Labonski, P., Guez, A., "Adaptive Sensor Fusion with Nets of Binary Threshold Elements," IEEE/International Joint Conference on Neural Networks '89, Washington D.C., June 1989, Vol. II, pp. 57-64.

[19] Bar-Kana, I., Guez, A., "Neuromorphic Adaptive Control," 28th IEEE Control and Decision Conference, Tampa, FL, December 1989.

[20] Selinsky, J., Guez, A., "The Role of A Priori Knowledge of Plant Dynamics in Neuro-Controller Design," 28th IEEE Control and Decision Conference, Tampa, FL, Dec. 89.

[21] Kent, L., Ahmad, Z., Guez, A., Freedman, W., "Application of Neural Network Modelling for the Muscle EMG to Torque Relation in the Ankle Joint," Neuroscience Annual Meeting, Phoenix, AZ, November 1989.

[22] Guez, A., Selinsky, J., "Learning/Adaptive Robot Control," Proc. IEEE Int'l Symposium on Intelligent Controls 1989, Albany, NY, September 1989, pp. 576 - 581.

[23] Bar-Kana, I., Guez, A., "Unsupervised Parallel Distributed Computing Architecture for Adaptive Control," Proc. IEEE Int'l Symposium on Intelligent Controls 1989, Albany, NY, Sept. 1989, pp. 174-178.

[24] Selinsky, J., Guez, A, Eilbert, J., Kam, M., "A Neuro-Expert Architecture for Object Recognition," International Joint Conference on Neural Networks, Washington D.C., June 89, pp. II-574. (Abstract)

[25] Kumar, S., Guez, A., "A Neural Network Approach to Target Recognition," IEEE/ International Joint Conference on Neural Networks '89, Washington D.C., June 1989, pp.II-573

[26] Guez, A., Ahmed, Z.,"Accelerated Convergence in the Inverse Kinematics via Multilayer Feedforward Networks," IEEE/

International Joint Conference on Neural Networks '89, Washington D.C., June 1989. Vol. II, pp. 341-344.

[27] Guez, A., Ahmad, Z., "On Learning In Neurocontrollers", American Control Conf., Boston, MA, June 1991. (Abstract)

[28] Guez, A., Ahmad, Z., "On Learning In Neurocontrollers", IEEE Int'l Conf. on Robotic & Automation, April 1991. (Abstract)

[29] Guez, A., Ahmad, Z., "On Generation of Exploratory Schedules in Closed loop for Enhanced Machine Learning", SPIE Conf. on Applications of Neural Networks, Orlando,Florida, April 1991. (Abstract)

[30] Bar-Kana, I., Guez, A., Rusnak, I., "Hybrid Expert-Neurocontroller for Robots", American Control Conf., Boston, MA, June 1991. (Abstract)

[31] Nevo, I., Guez, A., Ahmed, F., Roth, J.,"Mathematical Model for Adaptive Expert System for Anesthesia", 65th Congress of the Int'l Anesthesia Research Society, San Antonio, Texas,March 1991.

[32] Nevo, I., Guez, A., Ahmed, F., Roth, J.,"Vital Function Status A Comprehensive Display to Enhance Decision Making in Anesthesia", 1st annual meeting of The Society for Technology in AnesthesiaOrlando, Fla, January 1991.

[33] Nevo, I., Guez, A., Ahmed, F., Roth, J.,"Vital Function Status A Comprehensive Display to Enhance Decision Making in Anesthesia", 11th Int'l Symposium on Information Technology in Anesthesia Critical Care and Cardio-Pulmonary Medicine, Netherland,October 1990.

[34] Ahmed, F., Nevo I., Guez, A., "Anesthesiologist's Adaptive Associate (ANAA)," EMBS, Philadelphia, November 90.

[35] Bar-kana, I., Guez, A. "Simplified Techniques for Adaptive Control of Robot Systems" Invited Chapter in Advances in Control and Dynamic Systems, Vol.XXXVIII. C.T. Leondes (Editor), Academic Press (Published). To appear in 1990. (abstract)

[36] Bar-kana, I., Guez, A. "Simple adaptive control for a class of non-linear systems with applicaition to robotics," Int. J. Control, VOL. 52, No. 1, 77-99, 1990.

Other References

[Guez et al.] Guez, A., Protopopescu, V., Barhen, J., "On the Stability, Storage Capacity, and Design of Nonlinear Continuous Neural Networks," IEEE Trans. on Systems Man and Cybernetics, Vol. 18, No. 1, January/February 1988, pp. 80-87.

[Guez-Selinsky 88] Guez, A., Selinsky J. "A Trainable Neuromorphic Controller," Journal of Robotic Systems, Vol 5, No. 4, pg. 363-388, 1988.

[Selinsky] Selinsky, J.W., "Closed loop learning of rigid robot dynamics via design of exploratory schedules," MS Thesis, Drexel University, Philadelphia, PA, 1989.

[Slotine-Li] Slotine, J.J.E., and Li, W., "Adaptive manipulator control: a case study," IEEE Trans. Automatic Control, Vol. 33, no.11, p. 995, 1988.

Annual Reports

[AR1] Guez, A., "Designing the Architecture of a Hierarchical Neural Network to Model Attention, Learning and Goal Oriented Behavior," annual report submitted to AFOSR, November 1989.

The Application of Neural Networks to Robotics

Ziauddin Ahmad John Selinsky Allen Guez

Drexel University, Department of Electrical and Computer Engineering
Commonwealth Building, Philadelphia, PA 19104

We report on the application of Neural Networks (NN) architecture to two important problems in robotics. A hybrid architecture is used to solve the inverse kinematics problem with performance superior to the that of the classical approach. We also utilize a hierarchical NN architecture for learning rigid robot dynamics for the purpose of real time control.

List of Terms

NN : *Neural Network(s)*, IKP : *Inverse Kinematic Problem*, LMS : *Least Mean Square*, FF : *FeedForward*, DOF : *Degree(s) Of Freedom*,

1. Introduction

In this chapter we summarize the results of [Guez 89b] and [Guez 90] in applying NN architecture to problems in robot control. Section 2 describes a hybrid architecture used to solve the inverse kinematics problem with performance superior to the that of the classical approach. In Section 3 we utilize a hierarchical NN architecture for learning rigid robot dynamics for the purpose of real time control.

2. The Inverse Kinematic Problem in Robotics

The purpose of a robot is to manipulate objects with its end-effector. The end-effector is at one end of a chain of links, connected with joints, the other end is fixed. The motion of the robot is linked with the transformations between the joints and the end-effector. The actuation of the joints result in a motion of the end-effector. In order for the end-effector, carrying an object, to follow a desired path in the world space it is required to actuate the joints in a manner that will make the end-effector to track the desired trajectory.

Solution to the inverse kinematics exist in closed form for some robots or may be obtained by iterative methods. However, the procedure to obtain a closed form soluton, if it exists, is tedious and the iterative methods have a tendency to diverge when the initial guess to the is far from the actual solution.

2.1 Problem statement

The inverse kinematic problem (IKP) deals with finding the 'n' joint angle values 'q' of the robot that will position the end-effector in a desired position and orientation 'X' in the 'm' dimensional world space. This may be expressed as:

$$q = f^{-1}(X) \tag{2.1}$$

However, in general this solution is not unique. In many cases (e.g. redundant manipulators) there may result infinite number of solutions. In these cases additional constraints [Yoshikawa] in terms of the allowed configurations or performance function minimization are used to reduce the number of legitimate joint configurations or to single out a unique preferable one.

2.2 The proposed method

The formation of the proposed method starts with the robotic manipulator for which the IKP is to be solved and a 'blank' (untrained) multilayered feedforward neural network of suitable size. Then, training data in the form of pairs of the end-effector position and orientation $X = [\phi\ \theta\ \psi\ x\ y\ z]^T$, and the corresponding joint values $q = [q_1\ q_2\ \ldots\ q_n]^T$, are generated. These data are used for training the NN via the back error propagation algorithm [Rumelhart]. The position and orientation vector is the input and the corresponding joint values are the desired outputs of the network. After the training is completed the trained NN is coupled with the iterative method, as shown in the Fig. 1, for the purpose of operation. During the operation phase the desired position and orientation X of the end-effector is provided to the NN. The NN gives the approximate solution q_0 based on the learned connection weights. This approximate solution is taken as the initial guess by the iterative method to give the final solution within the specified tolerance. We consider that the type of solution out of the finitely many solutions is pre-specified to us and therefore training of the NN is restricted to the set of examples that pertain to this specific solution only.

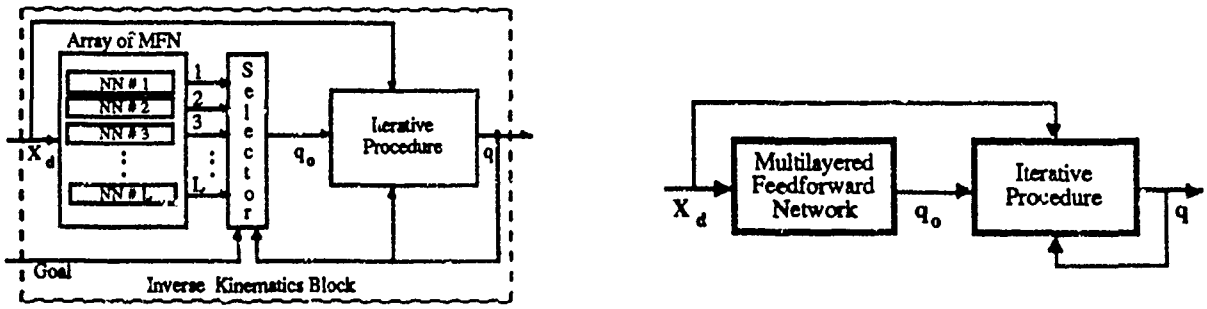


Fig. 1 Schematics of the proposed method employing the NN to provide the initial guess q_0 to the iterative procedure.

2.3 Examples

The back error propagation (BEP) algorithm simulating a three layer perceptron was employed to tackle the

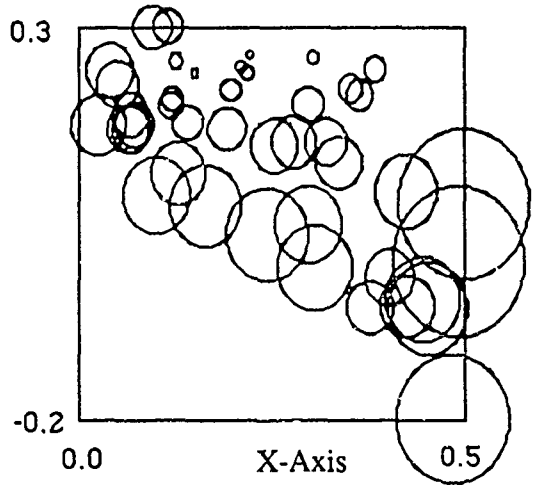
problems described below. Continuous inputs and outputs were assumed. The nodes assumed the symmetric sigmoidal nonlinearity [Scott]. Parameters of the NN are as given in reference [Guez-90]. Training was terminated when it was seen that the errors were not improving.

2.3-1 The Human Arm

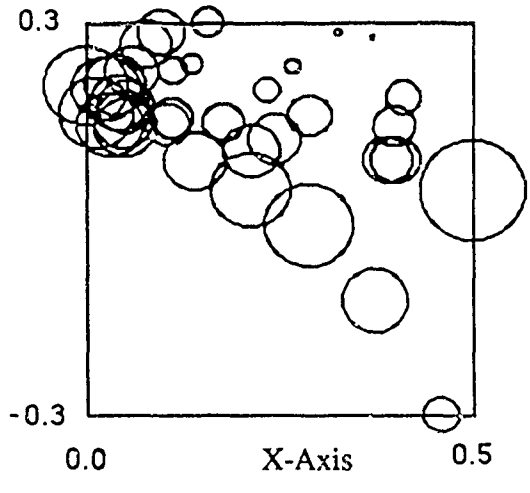
Here we show an attempt to capture the criteria that a human being allegedly optimizes in manipulating different objects by training the NN by a data set corresponding to some specified task. Planar motion, parallel to the ground was the considered task. The subject was asked to move an object in free space, in a plane parallel to the ground. Knowing the actual distances between the joints the data set was filtered to achieve a 10.0% tolerance about the respective actual values. The data set thus obtained contained only 43 words out of a total of 78 words. The network for this case constituted of 2 inputs and 3 output nodes and two hidden layers, each containing 10 nodes.

Fig. 4(a) shows the plot of the error in the positioning of the hand resulting for the trained data set, while Fig. 4(b) shows the same for a different data set obtained separately from the data set on which the network was trained. The two figures have similar errors indicating that the neural network has generalized on the trained data set. Large errors near X = 0.5 m are perhaps due to the singularity reasons or insufficient data near that region. Further, it was observed that the values for the elbow joint's were learned much better than those for the wrist joint and the shoulder joint.

As seen earlier, the filtered data set was only 55% of the total data gathered with 10% tolerance which indicates that the precision of the training data may not be adequate. However, it is observed that the NN is able to generalize upon the training data giving similar results for the untrained data to that of the trained data implying that implicit performance indices can be captured via NNs and perhaps identified via weight pruning and analysis.



(a) Hand positioning error for the training data.



(b) Hand positioning error for other than the training data.

Fig.4: Results of single MFN trained on human arm data in a plane perpendicular to the gravity. Shoulder is located at (0,0).

2.3-2 PUMA 560

The PUMA 560 parameters were taken from [Fu], page 37. This manipulator, which is a 6-DOF robot, was chosen for ease of generation of data and verification of results since it has a closed form solution. PUMA 560 has eight solutions for a given position and orientation signified by Right/Left - Shoulder, Above/Below - Elbow, and Up/Down - Wrist. In our simulations the training data corresponds to: LEFT Arm, BELOW Elbow and UP Wrist configuration. In the simulations the joint limits used for the 6th joint were -180^{o} to 180^{o} instead of -266^{o} to 266^{o}.

The network in this case consisted of 6 input nodes, one output node each for the 6 joints and two hidden layers for each joint consisting of 32 nodes in the first layer and 8 nodes in the second layer. The average error in the solution given by the NN for each joint taken over 100 samples ranged from 0.16^{o} to 6.02^{o} while the standard deviation of the error ranged from 11.5^{o} to 36.8^{o}. This solution was seen to be more scattered for Joints 4 to 6 as compared with the joints 1 to 3.

Next, the proposed method was compared by giving a fixed estimate to the iterative procedure. This Fixed Estimate was taken as: $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = \pi/4$, $\theta_4 = 0$, $\theta_5 = \pi/4$, and $\theta_6 = 0$, which is a configuration corresponding to which the NN was trained, as indicated in the beginning of this section. In the simulations the equations were solved by Gauss Elimination method and partial pivoting. The maximum number of iterations allowed for the iterative method were 100. The iterative method was successfully terminated when the norm of the difference between the desired and actual end-effector position and orientation was less than 1.0E-4. The average and standard deviation for the number of iterations for the proposed method and the Fixed Estimator, in a run of 100 data points is given in Table 1. From this table we can see that the proposed method achieves more than a two-fold efficiency in computing on the average with better consistency. Moreover, it was observed that the time taken by the NN equals two time units of the iterative procedure, which amounts to less than 10% of the time required to get the solution by the Fixed Estimator method.

Table. 1

Comparison of the proposed solution with the Fixed initial guess Newton Raphson Method, for PUMA-560 manipulator. The data corresponds to 100 samples.

Initial guess for Newton-Raphson Method	Number of Iterations	
	Average	Standard Deviation
Neural solution	9.96	12.95
Fixed	21.09	18.90

2.4 Conclusion

The proposed hybrid method which takes the solution given by the trained NN as an initial guess to an iterative procedure (Newton-Raphson in our case) combines the advantages of the NN and iterative methods, these being (1)

independent of the type of the manipulator, (2) simple to implement. Only forward kinematics is required for this method and as shown by our simulations this combination results in an increase in computational efficiency by 2-fold for the PUMA 560 (6-DOF) robot. This results in minimal processing within each control cycle and improves real time control performance.

3. Learning of Robot Dynamics Using Hierarchical Neural Network

Adaptation, in control, is the process of adjusting the controller to comply with the regulation and tracking requirements of the closed loop system. During operation, the controller is given a trajectory by a path planner in order to accomplish some useful task. The controller then adapts the parameters, on line, so as to satisfy the tracking requirements. If the parameters are not known exactly, there will be a transient period of tracking error while adaptation occurs. So that identification of the true parameters is desirable for increased tracking precision.

Inverse dynamics based control algorithms are computationally intensive, and may result in prohibitively slow control rates if implemented on serial computers. A parallel implementation of the above adaptive controller is proposed. The proposed implementation utilizes a hierarchical neural network architecture, which would ideally provide very fast control.

3.1 Dynamic Model

A rigid robot is defined as an open kinematic chain of rigid links, which are joined by linear or revolute joints. The dynamic model of a rigid robot manipulator can be written as

$$T(t) = I(q,t)\,\ddot{q}(t) + H(q,\dot{q},t)\dot{q} + B\dot{q}(t) + G(q,t) \qquad (3.1)$$

where

$q(t)$ is the n x 1 vector of joint linear or angular positions,

$I(q,t)$ is the n x n matrix of terms related to inertial forces,

$H(q,\dot{q},t)$ is the n x 1 vector of terms related to centripetal and coriolis forces,

B is the n x n diagonal matrix of viscous friction terms,

$G(q,t)$ is the n x 1 vector of terms related to gravitational forces,

$T(t)$ is the n x 1 vector of driving forces or torques

and n is the number of degrees of freedom (DOF) of the manipulator, (for further details see [Paul],[Fu]).

The representation (equation (2.2)) is not unique. Different choices of Ø result in different structures of the known function matrix $Y[q,\dot{q},\dot{q},\ddot{q}]$. In this work, it will be assumed that the parameter vector Ø will represent the p parameters to be identified.

3.2 Proposed Learning Method

The manipulator's dynamics can be represented by a weighted linear combination of suitable basis functions. The basis functions for a general class of manipulators are known apriori and may be trained into a series of three-layer feedforward network modules prior to use and then combined online in a fourth layer using a suitable weight

adjustment rule such as LMS (See [Widrow]).

We use a linear control law with added nonlinear compensation to control the manipulator. During an initial purely learning phase and whenever the manipulator is not needed for production, Exploratory Schedules (ES) are used to isolate and identify the nonlinear compensation terms. ES are trajectories specifically designed for efficient learning of the system dynamics. A block diagram of the closed loop system is shown in figure 2.

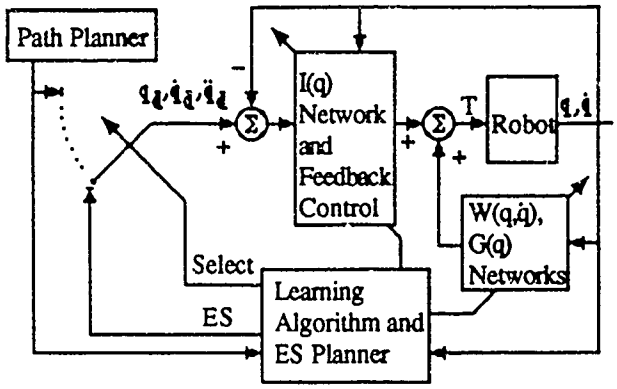


Fig. 2 Block diagram of the closed loop system

Learning of the nonlinear compensation terms is as follows. Let $G_O(q)$, $W_O(q,\dot{q})$ and $I_O(q)$ be the outputs of the G(q), W(q,q̇) and I(q) compensation networks respectively, where $W(q,\dot{q}) = H(q,\dot{q},t)\dot{q} + B\dot{q}$. After receiving the desired trajectory, the control that is to be applied to the robot is calculated using the feedback control (FB) from the linear controller and the feedforward control (FF) from the learned nonlinear compensation terms.

Learning of the G(q) terms is accomplished during positioning control phases of the trajectory. At the steady state of positioning control, $\ddot{q} = \dot{q} = 0$. Then from (3.1) it can be seen that $T = G(q)$. The applied torque at steady state can be used to learn the G(q) compensation network.

Learning of the W(q,q̇) compensation terms is performed during constant velocity portions of the trajectory. To isolate the W(q,q̇) terms, notice that at $\dot{q}$ = constant, $\ddot{q} = 0$ and $T = W(q,\dot{q}) + G(q)$. But G(q) has already been identified and is available as $G_O(q)$, so that we may isolate W(q,q̇) to be used in learning.

The inertia related terms I(q) may be continuously evaluated using the apriori known relationship (see [Slotine]) $dI(q,t)/dt = H(q,\dot{q}) + H(q,\dot{q})^T$.

After the dynamic equations of the manipulator are known, a feedback linearization or inverse dynamics based controller (see [Guez 82], [Fu], of the form

$$T = I_O(k_p e + k_v \dot{e} + \ddot{q}_d) + W_O(q,\dot{q}) + G_O(q) \tag{3.2}$$

where $e = q_d(t) - q(t)$, and $q_d(t)$ is the position reference signal is then employed. The closed loop system is then equivalent to

$$\ddot{q}(t) = k_p e + k_v \dot{e} + \ddot{q}_d(t) . \tag{3.3}$$

Which when k_p and k_v are appropriately chosen results in an asymptotically stable system. Note that this process does not presuppose the existence of an omnipotent controller that is capable of accurately tracking arbitrary trajectories. But rather it starts out with a simple controller capable of performing a limited task and uses learned knowledge of the manipulators dynamics to build a controller capable of controlling the manipulator at high speeds over the entire work space.

3.3 Simulation Results

The learning algorithm was tested on a simulated two planar DOF manipulator (see figure 3). Note that in these results exactly computed basis functions were used.

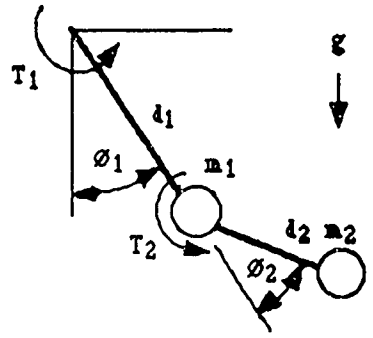


Fig. 3 Two degree of freedom planar manipulator.

The dynamic equations for this manipulator can be represented as linearly separable nonlinear subsystems

$$T_1 = (a_{11}+a_{12}C_2)\ddot{\emptyset}_1 + (a_{13}+a_{14}C_2)\ddot{\emptyset}_2 + a_{15}S_2)\dot{\emptyset}_1\dot{\emptyset}_2 + a_{16}S_2\dot{\emptyset}_2^2 + a_{17}\dot{\emptyset}_1 + a_{18}S_2 + a_{19}S_{12}$$

$$T_2 = (a_{21} + a_{22}C_2)\ddot{\emptyset}_1 + a_{23}\ddot{\emptyset}_2 + a_{24}S_2\dot{\emptyset}_1^2 + a_{25}\dot{\emptyset}_2 + a_{26}S_{12}$$

where $C_i = \cos(\emptyset_i)$, $S_i = \sin(\emptyset_i)$, $C_{ij} = \cos(\emptyset_i + \emptyset_j)$, $S_{ij} = \sin(\emptyset_i + \emptyset_j)$, and the a_{ij} terms are weighting constants to be identified.

Utilization of the ES for a 2 DOF manipulator result in small tracking errors may lead to small errors in the parameter identification. As can be seen in table 2, the learning algorithm was able to identify the a_{ij} terms to a very close approximation of their true values. The comparison of the 2 DOF manipulator controller before and after learning via ES showed that the controller performance is significantly improved. Further details of the method and results can be found in [Guez 89b].

Table. 2

Performance results of the learning algorithm

Weight Number	True Value	Estimated Value
a_{11}	30.0	29.51
a_{21}	10.0	10.06
a_{12}	20.0	19.28
a_{22}	10.0	10.20
a_{13}	10.0	9.79
a_{23}	10.0	10.12
a_{14}	10.0	9.47
a_{24}	10.0	10.22
a_{15}	-20.0	-19.28
a_{25}	5.0	5.01
a_{16}	-10.0	-9.97
a_{26}	98.1	98.10
a_{17}	5.0	4.61
a_{18}	196.2	196.19
a_{19}	98.1	98.10

3.4 Conclusion

The exploratory schedules have been specified as a desired trajectory that is to be followed to do learning while the manipulator is not doing other useful tasks. The simulation results of the ES for a 2 DOF manipulator showed how small tracking errors may lead to small errors in the parameter identification. The comparison of the 2 DOF manipulator controller before and after learning via ES showed that the controller performance is significantly improved.

References

[Benhabib]. B. Benhabib, A.A. Goldenberg, and R.G. Fenton, "A Solution to the Inverse Kinematics of Redundant Manipulators," Journal of Robotic Systems, 2(4), 373-385, 1985.

[Fu]. K.S. Fu, R.C. Gonzalez, and C.S.G. Lee, Robotics Control, Sensing, Vision, and intelligence. New York, NY , McGraw-Hill Book Company, 1987.

[Guez 82] Allon Guez, Optimal control of robotic manipulators, Ph.D Thesis, Univ. of Florida, Gainesville Florida, 1982.

[Guez-88]. Allon Guez, and Ziauddin Ahmad, "Solution to the Inverse Kinematics Problem in Robotics by Neural Networks," IEEE International Conference on Neural Networks, Vol-II, 617-624, July 1988.

[Guez-89a]. Allon Guez, and Ziauddin Ahmad, "Accelerated Convergence in the Inverse Kinematics via Multilayer Feedforward Networks," IEEE IJCNN, Vol-II, 341-344, June 18-22, 1989.

[Guez 89b] Allon Guez, and John Selinsky, "Neurocontroller Design Via Supervised and Unsupervised Learning,"

Journal of Intelligent and Robotic Systems, 2:307-335, 1989.

[Guez-90]. Allon Guez, and Ziauddin Ahmad, "Improving the Solution of the Inverse Kinematic Problem in Robotics by Neural Networks," Journal of Neural Network Computing, Spring 1990.

[Paul] R.P. Paul,. Robot Manipulators: Mathematics, Programming and Control. MA: The MIT Press, 1981.

[Rumelhart]. David E. Rumelhart, James L. McClelland, and the PDP Research Group, Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Vol. 1 and 2, MIT Press, Cambridge, MA , 1986.

[Scott]. W. Scott, and B.A. Huberman, "An Improved Three-Layer, Back Propagation Algorithm", IEEE First International Conference on Neural Networks, Vol-II, 637-643, June 1987.

[Slotine] J.J. Slotine and W. Li, "Adaptive Manipulator Control: A Case Study," Proceedings of the IEEE Robotics and Automation Conference, pp. 1392-1400, Rayleigh NC, 1987.

[Widrow] B. Widrow and S.D. Stearns, Adaptive Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[Yoshikawa]. Tsuneo Yoshikawa, "Manipulability of Robotic Mechanisms," International Journal of Robotics Research, vol. 4, No. 2, 3-9, Summer 1985.

[2]

NEUROCONTROLLERS

Allon Guez
ECE Dept. Drexel University
Philadelphia, P.A. 19104
Tel: (215) 895-1646

Work supported in part by grant AFOSR-89-0100

NEUROCONTROLLERS

Abstract

Neural network architectures possess computational features that may be useful in the construction of complex oynamical system controllers. In this article we review, define and classify such controllers, named neurocontrollers. The various neurocontroller classes, namely the supervised, unsupervised, fixed architecture and neurocontrollers with a critic are shown to differ mainly in the extent of apriori (teacher) knowledge which is mechanized in the neurocontroller architecture itself, rather than be made externally available. We discuss the neurocontroller's role as nonlinear, learning · ıd adaptive controllers.

Key Words: Neural Network, Supervised learning, Unsupervised Learning, Neurocontrol

I. Introduction

A neural network is a network of a large number of neuronlike subsystems which are dynamically coupled and exhibit via their collective behavior useful computational features. These networks were extensively studied by many researchers over the past 40 years.

Many different models have been suggested for neural networks. When all the neuron subsystems update their state simultaneously the network is called synchronous, otherwise, we have an asynchronous network. If the state of each neuron is represented with finite resolution we denote it a finite state neural network; else it is a continuous network.

Neural networks may also be classified by their principal operation phases. The "production" phase is the one in which the time evolution of the network's state manifests the useful computational properties sought for. For instance, when a neural network is used in an "associative memory" (i.e., retrieval of information by content), then the convergence of the network state to a stable attractor is the useful activity which, accordingly, is called the "production" phase. The learning/adaptation/design phase of a neural network, is the stage in which the network "learns", modifies, or designs its internal architecture as a result of its interaction with the environment (external input) and according to "metarules" which are inherent to the global context within which the network is to be useful. Early neural networks models separate between these operational phases. First the network is operated in a learning mode, where the network state is not allowed to change, but the neuron interconnections (architecture) are modified (designed); then learning ceases and

"production" is initiated by exposing the network to external excitation, which yields "useful" (converging) state trajectories.

In other models, the two phases are intermixed, allowing the network architecture and state to concurrently evolve, i.e., the network is simultaneously adaptive and productive.

Understanding of the structure and functionality of the Central Nervous System (CNS) of higher animals is important not only to neuroscientists, but also to the practitioners and theorists of the reemerging field of "BioMorphic Engineering" (BME). The latter includes Neuroengineering as a specific field.

These disciplines adopt the theory of evolution as their underlying axiom. It is believed that the solution to many complex engineering problems may be found by duplicating the hardware and functional structure of their (biological) analogous problem. The resulting solution is accepted as optimal one since evolution is regarded, albeit slow, as a recursive optimization procedure via the selection of new biological hardware and functional structure. The literature on BME, bionics, and cybernetics is very rich, we shall only mention ([Albus 75], [Arbib 85], [Rosenblat 61], [Tsypkin71] ,[Grossberg and Kuperstein 86], [Berkenblit et al 86]) as their context is closest to our focus: neurocontrol.

Even if evolution is not accepted as an optimization process, an engineer may "borrow" biological engineering principles by practicing the so called "learning analogy" paradigm, which is now widely accepted in the Artificial Intelligence (AI) community as complex engineering issues frequently posses an observable biological counterpart.

Thus, the principle of adopting algorithms, hardware architectures

and functional structures observed in nature, to the solution of engineering problems, without necessarily possessing an apriori rigorous understanding of their roles, is becoming an acceptable practice in the reemerging field of BME.

The design of neurocontrollers is a specific application of BME, and is the focus of this paper. Although neurocontrollers demonstrate adaptation and learning capabilities, they are distinct from both adaptive and learning controllers. In the next section, we define the class of neurocontrollers, compare them to adaptive and learning controllers and show how neurocontrollers are related to adaptive and learning controllers. Then in section III we review several neurocontroller architectures, and classify them according to the underlying learning algorithms and neural net architectures being employed. We propose and demonstrate a generalizing principle for all neurocontrollers architectures. This generalizing principle may allow for the natural incorporation of apriori knowledge in both the neural network architecture and learning algorithm selection for the neurocontroller design.

II. Adaptive, Learning, Nonlinear and Neurocontrollers

In this section, we define the class of neurocontrollers, compare it to the adaptive and learning controllers and demonstrate their relation to adaptive and learning controllers.

The problem of systems control can be defined as follows. Given the system S where

$$S: \dot{x} = f(x,u,t,p); \quad x(t_0) = x_0 \quad y = g(x,u,t,p) \quad (2.1)$$

where the time $t \in R$, the initial time is t_0, the state $X \in S_X \in R_n$, the control input $u \in S_U \in R^m$, the parameters vector $p \in S_p \in R^l$, the sets S_X, S_U and S_p are admissible state, control and parameter sets respectively, which account for general equality and

inequality constraints, and where n,m and l are appropriate integers. Find the control input $u(t)=c[v(t), z, \dot{z}]$ (open loop control) or $u(t)=c[v(t), x, z, \dot{z}]$ (closed loop control, such that certain terminal and optimality conditions are satisfied, where v(t) is a reference input vector and z(t) is a vector describing the controller state variables.

Under the above general definition, it is easy to classify the various controller classes according to the type of control function c which they are able to provide. If c is independent of z and $\dot{z}$, it defines the class of static controllers, in particular if c is also linear in x and v, we obtain the class of linear state feedback controllers. When c is a function of z and $\dot{z}$, we obtain a dynamic controller, where z(t) describe the controller's state variables. If $\dot{z}$ is a function of all or some of the system parameters, p, we obtain the class of adaptive controllers. (As there is no unique definition of adaptive control ([Astrom 87], [Landau 79]), the one given above represents the author's choice.)

An important feature of most adaptive controllers, including Model Reference Adaptive Controllers (MRAC), Self Tuning Regulators (STR) and Gain Schedulers (GS), is that they are model based, that is the control law c varies according to the best estimate of some system model parameters, p. Usually an estimation or identification scheme of some or all of the system parameters is directly or indirectly employed ([Astrom 87], [Landau 79]). The parameters estimates are then used in modifying the structure of the control law, c in an on-line fashion, automating in the process some control design rules. A general block diagram is depicted in Figure 1.

In contrast, the class of learning controllers ([Arimoto et al 85][Bondi et al 88], [Kawamura et al 84]) also called repetitive

control, or betterment control, (to be distinguished however from learning automata as described in [Narendra and Thatachar 74][Tsypkin 71][Fu 87] assume that the reference signal v(t) is periodic, the structure of the controller (i.e. the function c) is fixed, and learning is exhibited through the Iterative modification of the (open loop) time function u(t). Its general block diagram is described in Figure 2.

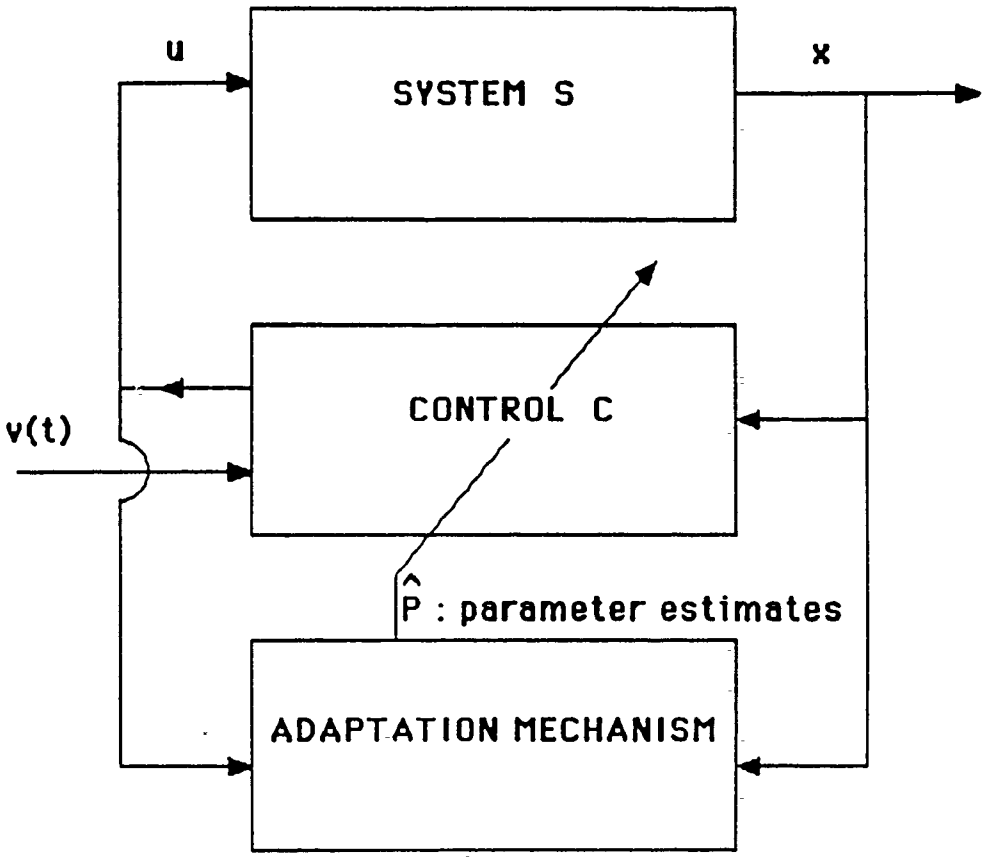


Figure 1: General block diagram for adaptive controller

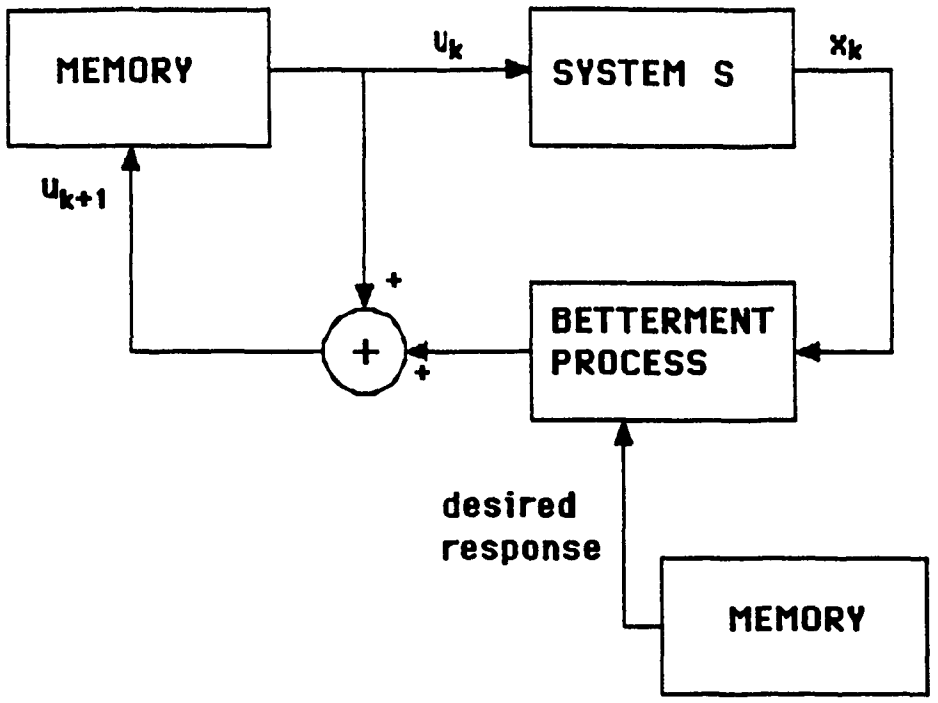


Figure 2: General block diagram for learning controller

Based upon the review of many works which are described below Table 1), the class of neurocontrollers is hereby defined as **the set of all controllers whose structure (control law) $c(x,v,z,\dot{z})$ is based on some neural network model architecture and learning paradigm.** It must be emphasized that the neural network model and the learning algorithm are defined independently from any specific process to be controlled (S), and is usually contributed by CNS anatomists and psychphsiologists. Neurocontrollers are, therefore, not model based, in the sense associated usually with adaptive controllers. Moreover, since the learning algorithm employed by neurocontrollers do not require periodic input, but rather can learn from arbitrary example sets, the class of learning controllers is a

proper subset of the neurocontroller set.

Although neurocontrollers are different from the model based adaptive controllers described above, they do perform real time adaptation through their online learning capabilities. The robustness of neurocontrollers stems from the fact that they are not model based. It is expected, however, that when compared with classical controllers with the same throughput (processing power) their accuracy will be inferior.

Finally when some or all of the nodes of the neural network employed in the neurocontroller are specifically allocated for the estimation and storage of all or a subset of the system parameters, p, we obtain a neuromorphic realization of model based adaptive controllers. In that sense one may regard the set of adaptive controllers as being contained in the class of neurocontrollers.

III. Neurocontrollers

Having defined the class of neurocontrollers in the previous section, we shall now focus our discussion on the various architectures and neurocontroller structures found in the literature. Neurocontrollers are classified according to: 1) the neural network model employed, 2) the learning algorithm employed, 3) their mode of operation, i.e., whether the learning/adaptation phase is done offline, prior to the engagement of the controller, or whether learning and adaptation are continuously present in the neurocontroller performance.

Vast amounts of literature, which deal with sensory motor control, understanding anthropomorphic motion, robot control, locomotion control, etc. is available today. Our focus has been on

reports that specifically aim at employing neural network architectures and learning algorithms in the design and construction of controllers for artificial dynamic systems. The criterion implied that many reports on the use of neural networks in understanding sensory motor control, (in the biological sense) and in learning static mapping, with or without feedback, as well as reports on learning automata which do not employ neural networks, were outside the scope of our study.

Table 1 below provides a classification of many neurocontroller design examples. For the readers convenience we shall briefly introduce the relevant terminology and algorithms.

Source	Application	Learning Algorithm	Critic Sup/ Unsup	Cont. Learning	Required Sensors	Expert Control Needed?	Comments
[Kawato, et al 87]	Robot Control	Widrow Hoff	Unsup.	Yes	Position	No	Requires linear Controller to direct the learning Based on CNS model
[Miller 87]	Robot Control	Modified "CMAC"	Unsup	Yes	Position velocity vision accel.	Yes, initially	Forms dynamic model in regions of state space. Uses fixed gain controller.
[Psaltis et al 87]	Control ex given for static plant	BEP	Unsup	No	Plant Output	No	Specialized learning propagates back through the plant
[Widrow 87]	Control of Cart-pole system	Widrow-Hoff	Sup	No	Position Velocity	Yes, Teacher	Uses single Adeline or Madeline network
[Barto et al 85]	Control of cart-pole system	Reinforcement Learning	Critic	Yes	Position, velocity	No, critic must be developed as well	

Source	Application	Learning Algorithm	Critic Sup/ Unsup	Cont. Learning	Required Sensors	Expert Control Needed?	Comments
[Elsey 88]	Kinematic robot control	Adaptive Inverse Control		Yes	Vision	No	
[Gawronski 88]	Nonlinear static plant	Neurophsiology based integral equation	Sup	No	State	Yes	
[Graf, Lalonde 88]	Kinematic robot camera coordination with obstacles		Unsup	Yes	Vision, state	No	Random exploration curse of dimensionality ignores difficult path planning
[Guez, Selinsky 88-a]	Cart-pole	BEP	Sup	No	State	Yes, could be human	Good robustness!
[Guez, et al 88]	Control of 2nd order linear sys.	Architecture Assignment	Off line	No	State	Yes	Compared with MRAC
[Guez, Selinsky 88-b]	Robot Control	Modified BEP	Unsup	Yes	State	No	
[Guez, Ahmad 87]	Robot Kinematic Control	Modified BEP	Unsup	Yes	State, vision	No	

Table 1b

Source	Application	Learning Algorithm	Critic Sup/ Unsup	Cont. Learning	Required Sensors	Expert Control Needed?	Comments
[Josin 88]	Improved inverse kinematic accuracy in robotics	BEP	Sup	No	State, vision	Yes	
[Kuperstein 88]	Kinematic robot camera coordination	Hebbian		Yes	State, vision	No	Functional approximation similar to CMAC
[Suddarth et al 88]	Planetary lander velocity profile tracking	BEP	Sup	No	State, vision	Yes	Combine a priori knowledge or "hint" to BEP
[Tolat, Widrow 88]	Cart-pole	LMS	Sup	No	Vision	Yes	
[Tsutsumi, Matsumoto 87]	Robot positioning	No learning Hopfield Net	Off line	No	State	Yes	
[Witten 77]	Discrete Time Markov Chain	"Bootstrap"			State	No	
[Widrow, Stearns 85]	Several linear examples	LMS	Sup	No	State	No	
[Widrow, et al 73]	Learns to play Black-Jack	Bootstrap	Critic	Yes	Encoder for card values	No	Uses single adeline network

TABLE 1C

Supervised Neurocontroller

Back Error Propagation - (BEP)

This is probably the most commonly used learning algorithm Its employment presumes a feedforward multilayer perception architecture as described in Figure 3.

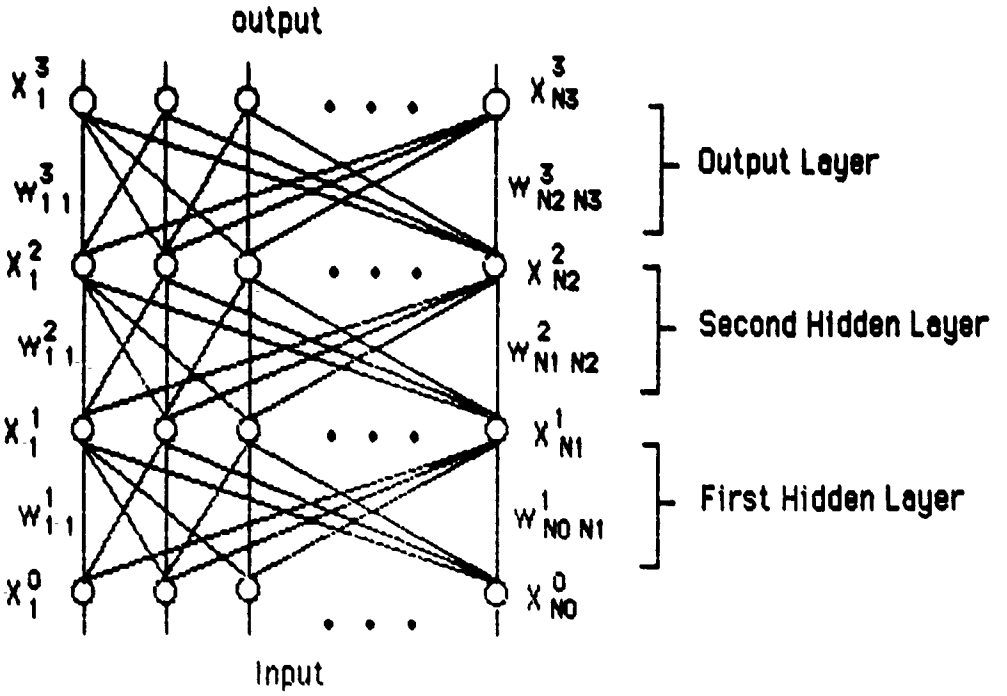


Figure 3: Three layer network

The network consists of several layers, with only feedforward interconnects. There are N_i nodes in the i-th layer. The output of

each node is usually a sigmoidal activation function of the form

$$x_j^k = \frac{1}{1 + e^{-\left(\sum_{j=1}^{N_{k-1}} w_{ij}^k x_j^{k-1} + w_{0i}^k \right)}} \qquad (3.1)$$

where Woi is a bias weight that is assumed to be connected to a unit that is permanently on, Wkji is the connection weight between the ith unit in the (k-1)th layer and the jth unit in the kth layer, x_j is the output of the jth unit in the kth layer.

The Back-Propagation learning algorithm is defined as follows. For the output layer the error is given by

$$\delta_i^k = \left(xd_j^k - x_j^k \right) \left(1 - x_j^k \right) x_i^k \qquad (3.2)$$

where Xd is the desired output of the network. For all other layers the error is

$$\delta_i^k = x_j^k \left(1 - x_j^k \right) \sum_{j=1}^{N_{k+1}} \delta_j^{k+1} w_{ij}^{k+1} \quad . \qquad (3.3)$$

The rule for changing the weights is

$$w_{ij}^k(n+1) = w_{ij}^k(n) + \eta \, \delta_i^k \, x_i^{k-1} + \gamma \left(w_{ij}^k(n) - w_{ij}^k(n-1) \right) \qquad (3.4)$$

where h and g are learning rate and momentum factors respectively.

CMAC (Cerebellar Model Arithmetic Computer) Developed by [Albus 75], CMAC is basically a table look up technique for reproducing vectorial functions, with a novel hashing method. Based on theories of brain structure and functions, Albus developed an architecture which consists of an ascending "sensory processing hierarchy" coupled to a descending "goal decomposition hierarchy" via "world models" at each level. Within

the goal decomposition hierarchy, sensory feedback variables and control goals are transformed into control outputs using a "CMAC" module. Albus implemented a controller for a seven-degree of freedom master-slave arm using the CMAC module. However, in general, the CMAC approach requires too much memory to produce control transformations with sufficient accuracy for practical application.

LMS (Least Mean Square) or Widrow Hoff and Adaline

The following LMS algorithm summary has been adopted from [Barto 83]. Figure 4 shows a functional diagram and schematic symbol for an adaptive linear threshold logic element (sometimes called "Adaline") [Widrow 85], [Widrow Smith 64]. The diagram indicates the terminology used and the input-output relationships. The zeroth input-signal component is always +1. Thus the zeroth weight Wo controls the threshold partitioning level. Before adaptation, an error e (defined as the difference between the output y and the desired response d) exists for each input pattern. The jth pattern would have an error of

$$e_j = d_j - y_j = d_j - X_j^T W_j \qquad (3.5)$$

where Xj is the jth input pattern vector and Wj is the jth weight vector. It is assumed here that the weight vector is adapted with each new input vector Xj.

If the inputs Xj and dj are statistically stationary, then the mean-square error (mse) is a quadratic function of the weights and there exists an optimal Weiner weight vector which minimizes it. Learning or updating of the weights can be done by a gradient-descent technique The least-mean-square (LMS) algorithm developed be Widrow and Hoff uses the error as an estimate of the gradient. This leads to the weight iteration rule

$$W_{j+1} = W_j + (a / (n+1)) e_j X_j \qquad (3.6)$$

where n + 1 is the total number of weights and a is a coefficient determining the fraction of the error ej corrected with each adaptation. The parameter a controls the stability of the adaptive process and the

rate of convergence. The adaptive process has been shown to be stable (convergent) if a is within the range $2 > a > 0$. Choosing a in this range ensures that e_j is reduced by the jth adaptation.

The "learning curve" plot of mse versus the number of adaptation cycles is a noisy exponential whose time constant can be shown to be

$$t_{mse} = (n+1)/2a \quad \text{adaptions.} \qquad (3.7)$$

Formula (3.7) is exact when all eigenvalues of the input correlation matrix $R = E[X_j X_j^T]$ are identical. Even when the eigenvalues differ substantially and the learning curve is not simply a single exponential plus noise, but is a sum of expo tials plus noise, experience has shown that in most cases the learning cu. .e can be well approximated by a single exponential having the time constant given by (3.7).

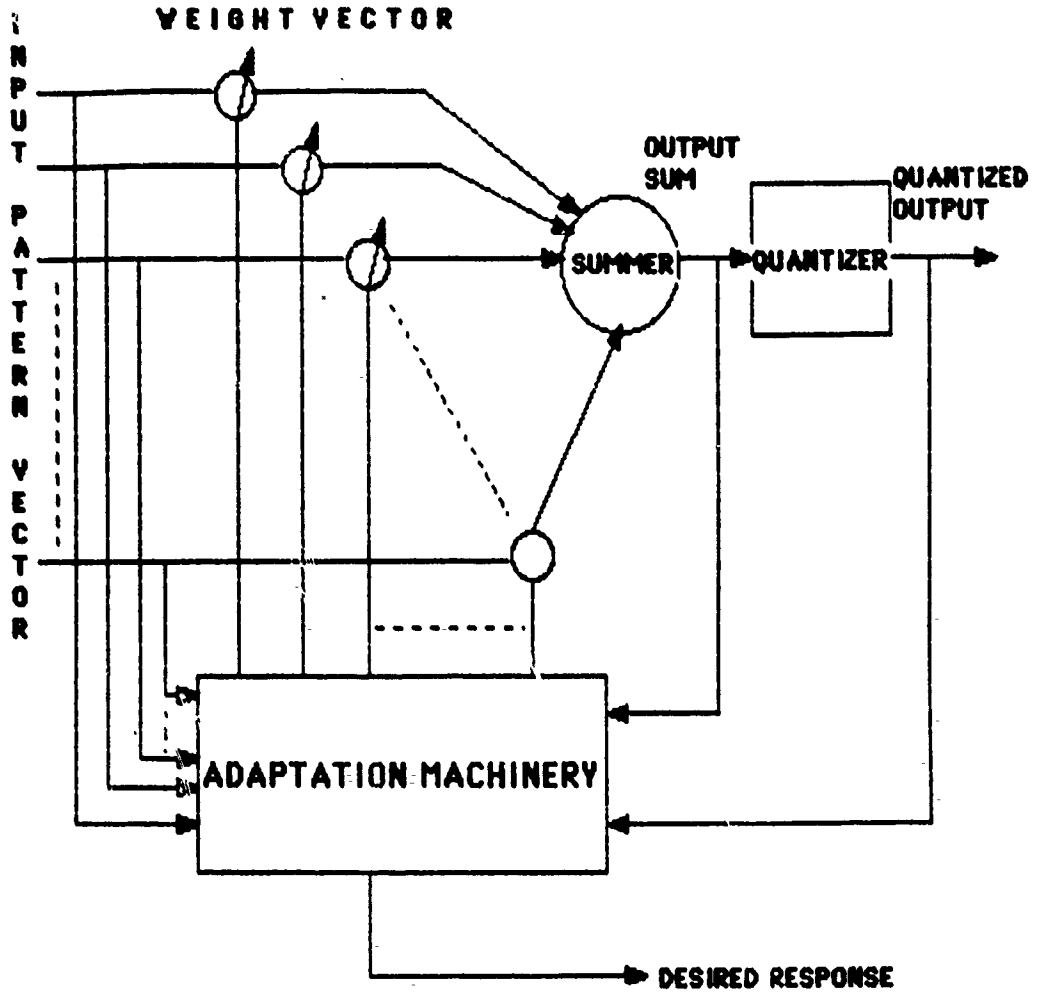


Figure 4: Automatically adapted threshold logic element (Adaline)

So far the BEP, CMAC, and LMS described neural network architectures which required explicit specifications of the desired response of each output unit at all times. Namely some kind of a supervisor or a teacher is required for their operation.

The employment of BEP, CMAC, LMS or any other supervised learning algorithm with its underlying neural network structure (multilayer perception for BEP, linear threshold units for LMS or simple computer memory for CMAC) may be lumped together under one category, hereby defined as the supervised neurocontrollers, as follows

Figure 5 describes the general architecture of a supervised neurocontroller architecture. It consists of a teacher, the trainable controller and the process to be controlled. The teacher may be automated as a linear or nonlinear control law, or it may be a human expert. In the case of training with an automated control law, the system engineer provides knowledge of the system dynamics through analysis of the controlled process. When the teacher is a human, we use knowledge of the system acquired through direct experience. The controller consists of one of the neural network architectures, e.g. LMS, BEP, or CMAC, that is suitable for supervised learning. Supervised learning is used in this architecture to provide control over the control law being learned. The state of the controlled process is provided to both the teacher and the network. The teacher defines the desired performance of the controller by providing examples of how the process is to be controlled. During training, the teacher controls the system by application of the continuously valued control signal u. After training the network controls the process and the teacher is removed.

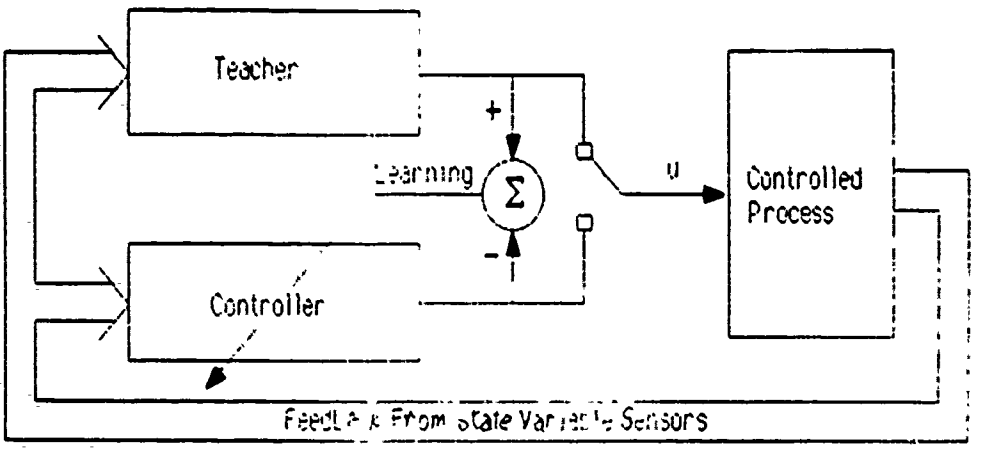


Figure 5: Supervised neurocontroller architecture

The class of supervised neurocontrollers is most popular. An overwhelming majority of the reported neurocontroller designs employed an architecture similar to the one given in Figure 5. CMAC is employed by

[Albus 81] and [Miller 88]. BEP is employed by many including ([Psaltis 87], [Pao 88], [Guez Selinsky 88a], [Guez Ahmad 87], [Josin 88]), LMS (Adaline arrays) is used by ([Widrow 87], [Kawato 87], [Guez Selinsky 88b], [Witten 77], [Widrow Stearns 85], [Tolat Widrow 88]).

Neurocontrollers With a Critic and Unsupervised Neurocontrollers

In many situations the assumptions made in the supervised neurocontroller design, regarding the availability of an expert knowledge to "guide" the neurocontroller during learning phases may be false. In particular, in neurocontroller design, the desired neurocontroller output (i.e. the plant's input) for a specific plant response is often unavailable, even theoretically, due to ignorance about the environment and the plant's dynamics. Under these circumstances, it is essential for the neurocontroller to possess some "self improving" or "self organizing" capabilities.

We distinguish between two situations. The first is named neurocontroller with a critic [Widrow 87], and is depicted in Figure 6. The teacher in Figure 5 is replaced with a critic. The key difference being that, while the teacher is able to provide a specific desired response for each and every output unit or effector in the neural network, the critic can only supply a scalar function which indicates how well the neurocontroller is performing. It is left to the learning algorithm employed in the neurocontroller to make use of this scalar critic signal together with the systems response in modifying the neurocontroller structure. This type of neurocontrol has several versions. It is called "bootstrap adaptation" or "punish/reward" in [Widrow et al 81], and "reinforced learning" in [Barto et al 81]. It has been employed in [Widrow 87] with Adalines, in [Barto et al 82] and in [Barto Sutton 81] with their associative search network.

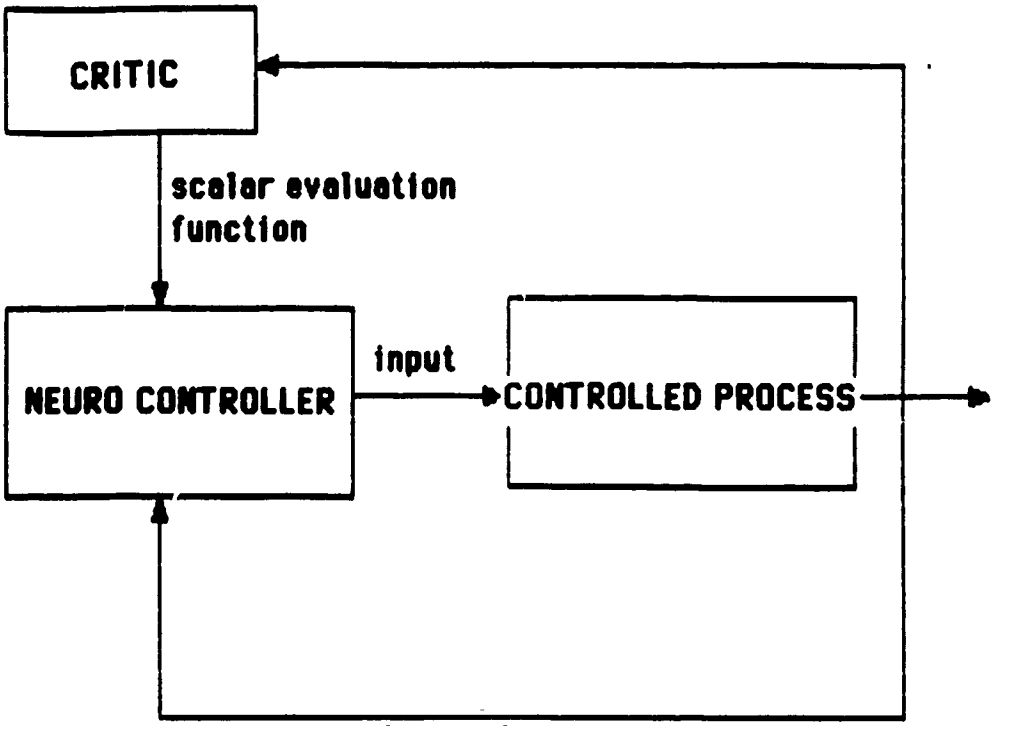


Figure 6: Neurocontroller with a critic

Another class of neurocontrollers named, unsupervised neurocontrollers is employed whenever no external information, other than the system's response, is available in real time to the neurocontroller, regarding its performance. Then in a manner similar to adaptive or learning controllers, the neurocontroller implements some internal performance index (specified through the selection of the neural network and learning model) minimization as its adaptation mechanism.

Several neural network models and learning algorithms belong to that class. The Adaptive Resonance Theory (ART) based sets of models of Grossberg, Carpenter, Cohen and their colleagues ([Carpenter Grossberg 87], [Cohen Grossberg 83] [Grossberg 82]), as well as the self organizing maps by Kohonen [Kohonen 84]. The latter model was applied to neurocontrol by [Graf La Londe 88].

Furthermore, several workers attempted to modify supervised learning algorithm and employ them in an unsupervised mode. Widrow [Widrow Stearns87] utilizes his LMS algorithm in what he names adaptive inverse control.

This technique is illustrated in Figure 7. The plant input is as before. The plant output is the input to the adaptive filter, which is implemented with an Adaline. The desired response for the adaptive filter is the plant input in this case. Minimizing mean square error causes the adaptive filter p^{-1} to be a best least squares inverse to the plant p for the given input spectrum and for the given set of weights of the adaptive filter. The adaptive algorithm attempts to make the cascade of plant and adaptive inverse behave like a unit gain.

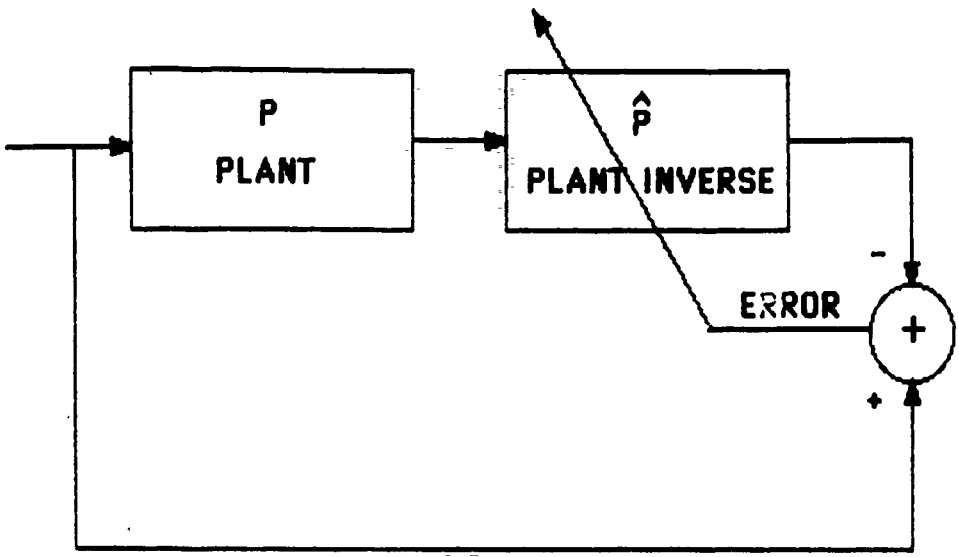


Figure 7: Adaptive inverse control

The Adaptive Inverse Control has been employed in various forms and modifications in ([Widrow and Stearns 85], [Eisley 88], [Psaltis et al 87]) where it is called "indirect learning" and [Kawato et al 87] when it is called "feedback error learning".

If the BEP algorithm is applied to configurations which lack access to the

desired neurocontroller response, we may use the output of the unknown system operator to propagate back the error as follows. Figure 8 describes the neural network (multilayer feedforward type with L layers) and the unknown map g to which it is connected.

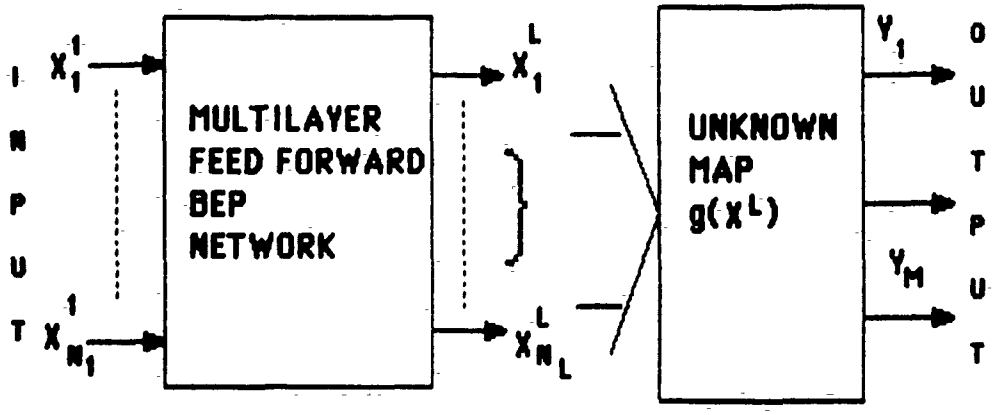


Figure 8: BEP through unknown system

Let $X^1 \in R^{N_1}$, $X^L \in R^{N_L}$, $g: R^{N_L} \to R^M$, $y = g(X^L)$ as in Figure 8. Also let d_i be the desired value of y_i when the vector x^1 is an input, let $e_i = y_i - d_i$ be the ith error. Note that no desired value for X^L is specified or available, which is usually the case in controller design. It can be shown that the learning equations (compare to equation (3.4)) are, starting at the last (output) layer connections:

$$W^L_{j\,k}(n+1) = W^L_{j\,k}(n) - \eta\, f'(X^{L-1}_j \left\{ \sum_{i=1}^{M} e_i J^i_k \right\} + \gamma \left(W^L_{j\,k}(n) - W^L_{j\,k}(n-1) \right) \quad (3.8)$$

Where $W^L_{j\,k}(n)$ is the connection weight at the time step n from the j^{th} unit in layer L-1 to the k-th unit in the last layer L, where f' is the derivative of f the node update (usually sigmoidal) function, $J^i_k = \partial y_i / \partial X^L_k$ estimated via $\Delta y_i / \Delta X^L_k$ and X^N_j the output of the j^{th} unit in the Nth layer. Then proceeding with the next (L-1)

$$w_{ij}^{L-1}(n+1) = w_{ij}^{L-1}(n) + \left\{ \sum_k \frac{\Delta w_{jk}^{L}}{x_j^{L-1}} w_{jk}^{L}(n) \right\} f' x_i^{L-2} - \gamma \left\{ w_{ij}^{L-1}(n) - w_{ij}^{L-1}(n-1) \right\}$$

(3.9).

This process is repeated for all layers. The modified BEP makes use of the usually available data about the desired systems output to direct learning.

Fixed Architecture Neurocontrollers

So far we only discussed neurocontrollers, which allow through some online or offline training or learning, for the modification of their architectures. Another class of neurocontroller which we denote as fixed architecture neurocontrollers are based upon neural network models with fixed architectures such as Hopfield binary and continuous models [Hopfield 82], [Hopfield 84] Examples of neurocontrollers in this class are given in [Tsutsumi Matsumoto 87] and [Guez et al 88] The architecture for these neurocontroller is selected according to a choice of parameters of an energy cost function to be minimized as in [Hopfield Tank 85], or use of the "outer product" rule (in the binary model case) [Hopfield 82]. Another method for the selection of the neurocontroller architecture to possess a prescribed steady state topology is described in [Guez Protopopescu Barhen 88]. It transforms the design problem to the solution of a set of piecewise linear inequalities

Based on the neurocontrollers discussion given above, it is easy to make the following observation Compare the block diagrams on Figures 5, 6, 7, and 8.) The difference between the supervised neurocontrollers, neurocontrollers with a critic, unsupervised neurocontrollers, and fixed architecture neurocontrollers mainly lies in the amount of apriori knowledge (teacher's or critic's knowledge) that is implemented in the neurocontroller model For example, it is clear from Figures 5 through 8 that if the teacher or critic functions of a supervised neurocontroller are implemented via neural network model, they could be included in the

"neurocontroller box" and result in an unsupervised neurocontroller.

IV. Summary

Neural network architectures possess computational features that may be useful in the construction of complex dynamical system controllers. In this article we reviewed, defined and classified such controllers, named neurocontrollers. The various neurocontroller classes, namely the supervised, unsupervised, fixed architecture and neurocontrollers with a critic have been shown to differ mainly in the extent of apriori (teacher) knowledge which is mechanized in the neurocontroller architecture itself, rather than made externally available. We also discussed the neurocontroller's role as a nonlinear, learning and adaptive controller.

REFERENCES

Albus, J.S (1975) "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, vol 97, pp. 220-227

Albus, J.S. (1981) *Brain Behavior and Robotics.* Peterborough, NH: BYTE Books, ch. 6, pp. 139-179.

Arbib, M.A. (1985) "Brain Theory and Cooperative Computation", *Human Neurobiology*, Springer Verlay, vol 4.

Arimoto, S., Kawamura, S., Miyazaki, F., Tamaki, S. (1985) "Learning Control Theory for Dynamical Systems," *Proc. IEEE 24th. CDC*, Ft. Lauderdale, FL

Astrom, Karl J. (1987) "Adaptive Feedback Control," *Proceedings of the IEEE*, vol 75, no 2, pp 185-217

Barto, A.G., Sutton, S.R., Brouwer, P. (1981) "Associative Search Network: A Reinforcement Learning Associative Memory," *Biol Cybern.*, vol 40, pp. 201-211

Barto, A.G., Sutton, S.R. (1981) "Landmark Learning: An Illustration of Associative Search," *Biol Cybern*, vol. 42, pp. 1-8.

Barto, A.G., Anderson, C.W., and Sutton, R.S. (1982) "Synthesis of nonlinear control surfaces by a layered associative search network," *Biol Cybern*, vol 43, pp 175-185

Barto, A.G., Anderson, C.W., and Sutton, R.S. (1983) "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans Syst, Man, Cybern* vol SMC-13, no 5, pp 834-846.

Berkinbilt, M., Feldman, A., Fukson, O. (1986) "Adaptability of Inate Motor Patterns and Motor Control Mechanisms," *Behavioral and Brain Sciences*, 9, 585-638, Cambridge University Press.

Bondi, P., Casalino, G., Gambardella, L. (1988) "On the Iterative Learning Control Thoery for Robotic Manipulators," *IEEE J. Rob and Auto.*, Vol 4,No.1.

Carpenter, G., Grossberg, S (1987) "A Massively Parellel Architecture for a Self-Organizing Neural Parttern Recognition Machine," *Computer Vision Graphics,and Image Processing*, Vol. 37, pp. 54-115.

Cohen, M.A. and Grossberg, S. (1983), "Absolute Stability of Global Patttern Formation and Parallel Memory Storage by Competetive Neural Networks," *IEEE Trans. Syst. Man Cyber.*, SMC-13(5), 815.

Elsley, R.K. (1988) "A Learning Architecture for Control Based on Back Propogation Neural Network," *Proc ICNN 88*, San Diego, CA.

Fu, K. (1987) "Learning control systems - Review and outlook," *IEEE Trans Automatic Control*, AC-15, 210-221

Gawronski, R. (1988) "A Learning Algorithm for Multilayer Neuronlike Network," *Proc of 1st Florida AI Research Symposium*

Graf D., Lalonde W (1988) "A Neural Controller for Collision Free Movement of General Robot Manipulator," *Proceedings of the IEEE First International Conference on Neural Network*

Grossberg, S (1982) *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition and Motor Control*, Reidel Press, Boston

Grossberg, S and Kuperstein, M (1986) *Neural Dynamics of Adaptive Sensory-Motor Control*, North-Holland Amsterdam.

Guez, A Protopopescue, V., Barhen, J (1988) "On the Stability, Storage Capacity, and Design of Nonlinear Continuous Neural Networks," *IEEE Trans Syst., Man, Cyber.*, Vol 18, No 1

Guez, A., Eilbert, J., Kam, M. (1988) "Neuromorphic architecture for control," *IEEE Control Systems Magazine*, Vol 8, No. 2.

Guez, A., Selinsky J. (1988-a) "A Trainable Neurmorphic Controller," *Journ. of Robotic Systems*, Vol 5, No. 6.

Guez, A., Selinsky J. (1988-b) "Supervised and Unsupervised Learning Neurocontrollers," submitted for publication.

Guez, A., Ahmad Z. (1987) "Solution to the Inverse Kinematic in Robotics by a Neural Network," *Proceedings of the IEEE First International Conference on Neural Networks*

Hopfield, J. J. (1982) "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat. Acad. Sci. USA* 79, 2554.

Hopfield, J. J. (1984) "Neurons with Graded Response Have Collective Computational Properties Like Those of Two State Neurons," *Proc. Nat. Acad., Sci, USA* 81, 3088

Hopfield, J. J., and Tank, D. W. (1985) "Neural Computations of Decisions in Optimization Problems," *Biol. Cyber.* 52, 141

Josin, G., Charney, D., White, D. (1988) "Robot Control Using Neural Networks," *Proc. ICNN 88*, San Diego

Kawamura, S., Miyazaki, F., Arimoto, S. (1984) "Iterative learning control for robotic systems," *Proc. of IECON 84*, Tokyo, Japan

Kawato, M., Furukawa, K., and Suzuki, R. (1987) "A hierarchical neural-network model for control and learning of voluntary movement," *Biological Cybernetics* No 56

Kohonen, T. (1984) *Self Organization and Associative Memory*, Springer-Verlag, New York

Kuperstein, M. (1988) "Neural Model of Adaptive Hand Eye Coordination for Single Postures," *Science*, Vol 239, pp 1308.

Landau, Y. D. (1979) *Adaptive control the model reference adaptive control approach*, Marcel Dekker, New York

Miller, T. (1988) "Sensor Based Control of Robotic Manipulators using a General Learning Algorithm," *IEEE J. Rob. and Auto*, Vol. RA-3, No. 2.

Narendra, K. S., and Thatachar, M. A. L. (1974) "Learning automata - A survey," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-4, pp. 323-334

Pao, Yoh-Han, and Sobajic, D. J. (1988) "Intelligent control of a robot manipulator using artificial neural networks," *Manufacturing International '88 Conference*, Atlanta, GA, April 17-20.

Psaltis, D., Sideris, A., Yamamura, A. (1987) "Neural controllers," *Proceedings of the IEEE First International conference on Neural Networks*, San Diego..

Rosenblatt, F. (1961) *Principles of Neurodynamics, Perceptrons and the Thoery of Brain Mechanics*, Washington: Spartan Books Co.

Suddarth, S., Sutton, S. , Holden, A. (1988) " A Symbolic Neural Method for Solving Control Problems," *Proceedings of the IEEE First International Conference on Neural Networks.*

Tolat, V., Widrow, B. (1988) "An Adaptive Broom Balancer with Visual Inputs," *Proceedings of the IEEE International Conference on Neural Network 88*, San Diego, Ca..

Tsutsumi, K., and Matsumoto, H. (1987) "Neural Computation and Learning Strategy for Manipulator Position Control," *Proceedings of the IEEE First International Conference on Neural Networks.*

Tsypkin, Z. (1971) *Adaptation and Learning in Automatic Systems* New York: Academic Press

Widrow, B., Gupta, N. K., Maitra, S.(1973) "Punish/reward: learning with a critic in adaptive threshold systems," *IEEE Trans. Syst. Man, Cybern.*, Vol SMC-3, pp. 455-465

Widrow, B., and Smith, F. (1964) "Pattern- recognizing control systems", in *Computer and Information Sciences*, J. T. Tow and H. R. Wilcox, Eds. Clever Hume Press, pp. 288-317

Widrow, B. (1987) "The original adaptive broom balancer," *IEEE Conference on Circuits and Systems*, Philadelphia PA.

Widrow, B., Stearns, S (1985) Adaptive Signal Processing, Prentice Hall.

Witten, I. (1977) "An Adaptive Optimal Controller for Discrete Time Markov Environments," *J. Information and Control*, Vol. 34, pp. 286-295.

[3]

ART BASED ADAPTIVE POLE PLACEMENT FOR NEUROCONTROLLERS

Sanjay S. Kumar and Allon Guez *
Dept. of Electrical & Computer Engineering,
Drexel University, Philadelphia, PA 19104

ABSTRACT

Indirect adaptive control of low order plants that are subjected to parametric variations arising from changes in operating environment requires real time dynamic system identification. In this paper we propose a control scheme that utilizes a nearest neighbor search type of classifier capable of *learning* to dynamically identify these variations in plant parameters. The neural network architecture employed is based on the *Adaptive Resonance Theory* (ART-II) proposed by Stephen Grossberg and Gail Carpenter, 1987. An adaptive pole placement controller for a slow time varying linear second order system is implemented based upon this architecture to assess the performance of the network and the overall control scheme with the neural network in the control loop. The control strategy is based upon identification of changes in the time response characteristics of the system to standard test signals which are assessed by the network. A pole placement algorithm is utilized to relocate the poles of the overall closed loop system by altering the gains of the process controller to obtain desired system response. Experimental studies on a simulated system, employing a Proportional Derivative controller are encouraging.

Key Words : Adaptive control; System identification; Feature extraction; Pole placement; Learning; Supervised learning mode; Parametric variation.

1 INTRODUCTION AND BACKGROUND

The most significant advantage in applying neural networks lies in the *parallel distributed nature of processing* (PDP) that can be realized through their implementation in hardware [Rumelhart & McClelland,1986; Lippmann,1987]. In addition to PDP, artificial

---

* This work was partially supported by a grant from AFOSR, grant # 890010.

neural networks demonstrate the ability to *learn* a task either through training by example, (*supervised learning*) or on their own, by means of heuristic mechanisms built into their architectures (*Unsupervised learning*). The necessity of specifying explicit instructions in the form of programs or algorithms is thereby eliminated. For instance, it is possible to model the dynamics of an unknown process by imposing on a neural network the process inputs and outputs to enable it to *learn* the transfer function in the form of a mapping from an input to output space. [Psaltis, 1987; Guez & Selinsky,1988; Guez 1988]. *Fast association and recall* is another attribute of massively interconnected networks wherein the time required for association and recall of information after training remains independent of the past learning history or the size of the network. The ability of a neural network to *generalize from insufficient or partial information* is found to be especially useful when the input is corrupted by noise and would enable it to be a succesful computing architecture for adaptive controllers.

Adaptive Control : The starting point in any adaptive control scheme is a feedback control loop within the process and a controller with adjustable parameters. The main issue is to find a method for changing the controller parameters in response to changes in the plant parameters resulting from changes in the environment. Of the many schemes for adaptive control, Model Reference Adaptive Control [Astrom,1983,84; Landau,1979], Self Tuning Regulation [Astrom,1983,84; Ortega & Kelly,1984; Cameroon & Seborg,1982] and Gain Scheduling are most popular. All of the above control schemes involve a system identification component. They differ only in the techniques they employ to accomplish the task of system identification and tuning of the controller. Adaptive Control schemes can be broadly classified into two categories, namely, Direct Adaptive Control, (DAC) and Indirect Adaptive Control, (IDAC). In the former, the controller parameters are directly adjusted on-line such that the error between the plant output and the output of an assumed model asymptotically tends to zero. In IDAC, the parameters of the unknown plant are first estimated using an estimation technique and these are in turn used to adjust the parameters of the process controller. See figure 1 for a block diagram a generic indirect adaptive control system.

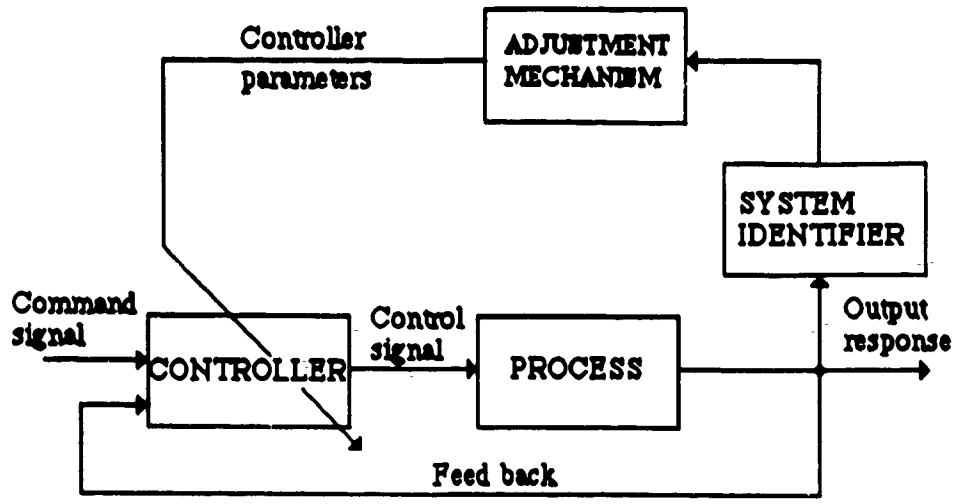


Figure 1 Block diagram of a generic indirect adaptive control scheme employing a system identifier, an adjustment mechanism and a process controller.

System Identification : System identification is the experimental approach to process modeling. It includes, experimental planning, selection of model structure, parameter estimation and validation. The underlying *purpose of system identification* in control systems is to design *control strategies* or to alter existing strategies to obtain *desired performance* in response to changes in the system's environment characterized by changes in the system's disturbance dynamics. System identification is also used to analyze the properties of a particular system. Literature survey on system identification [Astrom & Eykhoff,1971; Astrom & Wittenmark,1971,1984; Goodwin and Payne,1977] showed that in automatic control the knowledge about a system and its environment, which is required in the design of a control system, is seldom available *a priori*. Even if the equations governing the system are known in principle or it is possible to obtain them by performing experiments on the system, it often happens that knowledge of a particular parameter is missing or the system is too complex to model. Hence the need for approximating system behavior through assumed models and estimating their parameters through *system identification techniques*. Identification problems can be formulated using different frameworks. Most identification problems are modelled as *optimization problems* where the main objective is the formulation of different process models that minimize a functional of the process and model outputs called the *loss function*.

In this paper, we report on an attempt to formulate the system identification problem, for the purpose of adaptive control, in an identification framework that exploits

the computational features of an Adaptive Resonance Theory based neural network. We have chosen the ART-II model with the hope that the network would *learn on a single presentation of an input pattern* and that it would be *self organizing*, with the clustering of input patterns being done without an explicit specification of the classification desired. The additional advantages of using the ART-II that we expected were: a) Elimination of the need to retrain the network when there is a change in the prototype set of input patterns or when more patterns are added to it; b) Better control over the classification in terms of fine and coarse categorization via the attentional vigilance parameter of the network and c) The network's capability of unsupervised learning.

2 DESCRIPTION OF THE PROBLEM

We are given a slow time varying linear dynamical system (*Plant*), G_p, modeled, say as a second order system, with plant constant K_m and α , β as the *unknown constant* or *time varying* parameters and a *process controller* G_c, say a Proportional Derivative controller with gains K_p and K_d. Our goal is to implement an *adaptive pole placement control scheme* using a *neural network* architecture that would identify current plant parameters in order to estimate and tune arbitrarily assigned initial gains of the controller such that overall system poles can be relocated via a *pole placement module* to obtain a system response that matches the one given by an *assumed ideal model*. It should be noted that although the plant and the controller are linear the process is overall nonlinear.

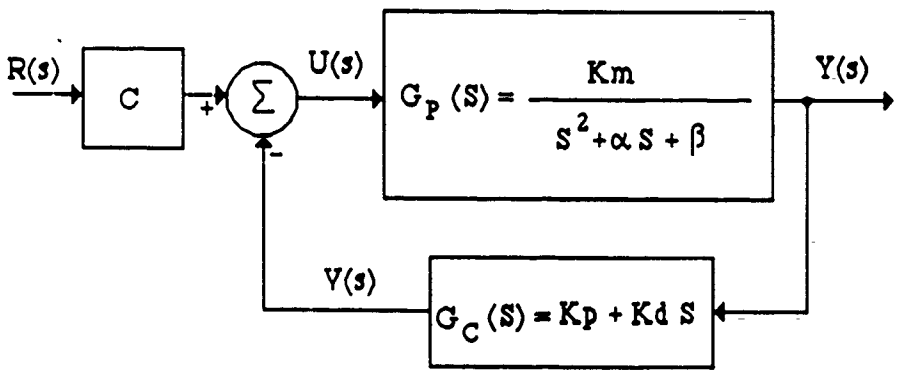


Figure 2 Block diagram of a process depicting a second order plant, $G_p(s)$ and a PD feed back controller, $G_c(s)$. C is a constant gain amplifier.

Let the *plant transfer function* be represented as

$$Gp(s) = \frac{Km}{s^2 + \alpha s + \beta} \qquad (1)$$

Let the *reference input* be a known periodic function of the type:

$$r(t) = r(t+T) \text{ , for all } t > 0 \qquad (2)$$

We seek to find online, the Proportional and Derivative (PD) gains, K_p, K_d and the D.C. bias C (see Figure 2), such that the control law given by:

$$u = C\,r - K_p y - K_d s\,y = C\,r - G_c y \qquad (3)$$

will result in the *ideal closed loop transfer function* of the form:

$$G_O^*(S) = \frac{Y(s)}{R(s)} = K^* \left[\frac{\omega_n^{*2}}{s^2 + 2\zeta^* \omega_n^* s + \omega_n^{*2}} \right] \qquad (4)$$

where, K^*, ζ^*, ω^*_n are respectively the *desired D.C. gain, damping coefficient* and the *natural frequency* of the system.

3 CONTROLLER DESIGN

We describe in this section, the design aspects of the proposed controller highlighting details of the various functioning modules involved in the process. We first provide an explaination of our approach followed by the underlying methodology.

Approach : The general block diagram of the overall control scheme is shown in figure 3. The common a priori assumption in the design of controllers for partially known processes is adopted with the design procedure being divided into two steps: *identification* and *control* (indirect adaptive control strategy) [Astrom & Eykhoff,1971].

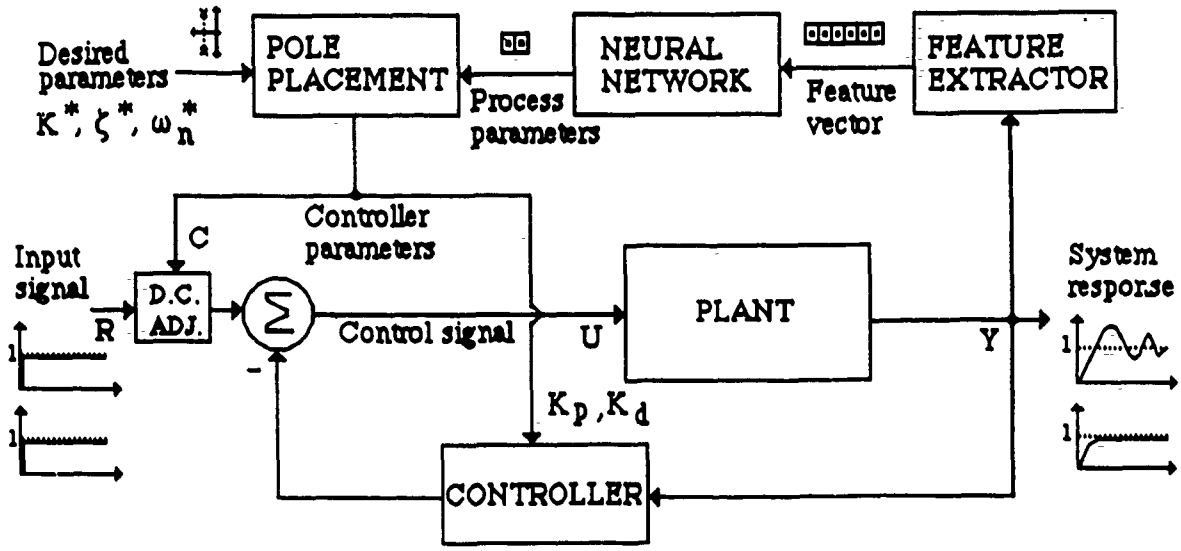


Figure 3 Block diagram of the neuromorphic adaptive control scheme. Parameters that correspond to the desired location of the overall system poles are input to the pole placement module along with the current estimates of the plant parameters provided by the neural network. The pole placement module computes the required controller gains and the D.C. adjust parameter to modify the original system response shown at the output of the plant.

Methodology : Identification of plant parameters is achieved by extracting the features of the system's *closed-loop* transient time response to a step input. The neural network in the control loop which is trained to map features of a system's time response to its parameters, gives the current parameter estimates of the plant in the closed loop process shown in figure 3. The pole placement algorithm incorporated utilizes the estimates provided by the network to compute new controller gains in order to modify the resulting system response to suit the one obtained by the assumed ideal model. A D.C. adjustment mechanism is incorporated to compensate for any D.C. bias that might be associated with the original system response..

The control scheme proposed therefore comprises of the following fundamental processing modules: (a) Plant; (b) System Identifier incorporating a Feature Extractor, a neural network and a pole placement module and (c) Controller with the DC gain adjust mechanism. The purpose of the system identifier is to identify the plant parameters in response to changes within its environment. The time response of the system is characterized by its *features* or *performance indices* which are nonlinear functions of the system parameters. The *feature extractor* incorporated in the control loop determines the performance indices associated with the response to enable system identification via the neural network. It must be noted that identification is dependent on the features of the time

response rather than the response per se. This is done in order to compress the information contained in the response such that the input vector to the neural network remains compact [Kumar & Guez,1989]. This procedure restricts the dimension of the neural network to a minimum thereby increasing its computational speed and overall efficiency of the process.

Pole Placement : The overall transfer function is obtained from equations (1),(2) and (3) as explained by the following input output relations. Refer to the appendix for a state space description of the proposed adaptive control scheme. The output of the process is given by:

$$Y(S) = G_p(S)\,U(S) = Gp(S)\,[\,C\,R(S) - G_c(S)\,Y(S)] \tag{5}$$

while the overall transfer function of the process is :

$$G_O(s) = \frac{Y(s)}{R(s)} = \frac{C\,G_p(s)}{1 + G_p(s)\,G_c(s)} = \frac{CK_m}{s^2 + (\alpha + K_mK_d)\,s + (\beta + K_mK_p)} \tag{6}$$

or

$$G_O(s) = \frac{CK_m}{(\beta + K_mK_p)} \left[\frac{(\beta + K_mK_p)}{s^2 + (\alpha + K_mK_d)\,s + (\beta + K_mK_p)}\right] \tag{7}$$

In order to achieve the *ideal transfer function*, we set $G_O(S) = G^*_O(S)$ to obtain

$$\frac{CK_m}{(\beta + K_mK_p)} \left[\frac{(\beta + K_mK_p)}{s^2 + (\alpha + K_mK_d)\,s + (\beta + K_mK_p)}\right] = K^* \left[\frac{\omega^{*2}_n}{s^2 + 2\,\zeta^*\omega^*_n s + \omega^{*2}_n}\right] \tag{8}$$

The desired parameters expressed in terms of the process parameters are then

$$(\beta + K_mK_p) = \omega^{*2}_n \tag{9}$$

$$(\alpha + K_mK_d) = 2\,\zeta^*\omega^*_n \tag{10}$$

$$K^* = \frac{CK_m}{\omega^{*2}_n} \tag{11}$$

from which the *new controller gains* are computed as:

$$K^*_p = \frac{\omega^{*2}_n - \beta}{K_m} \tag{12}$$

$$K^*_d = \frac{2\zeta^*\omega^*_n - \alpha}{K_m} \tag{13}$$

$$C^* = \frac{K^*(\beta + K_m K^*_p)}{K_m} = \frac{K^*\omega^{*2}_n}{K_m} \tag{14}$$

Equations (12) through (14) constitute the pole placement algorithm used in the pole placement module shown in figure 3.

The Feature Extractor :

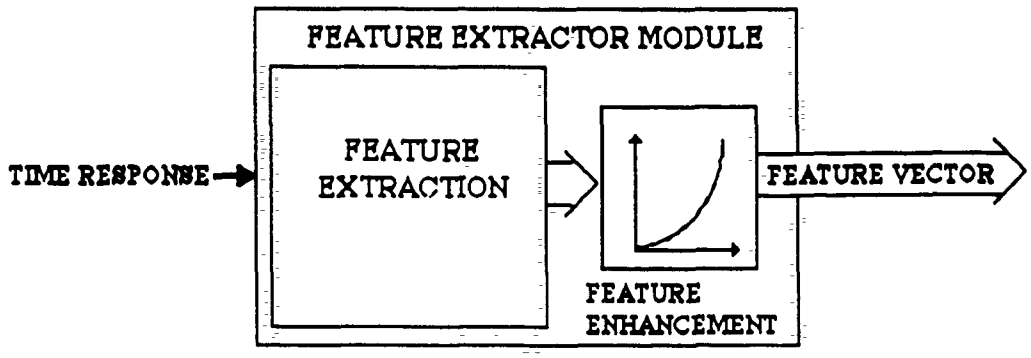


Figure 4 Block diagram of the feature extractor module.

The *feature extractor* in the adaptive control scheme, illustrated in the figure 4, is used to determine the *performance indices* or features of a particular time response during its transient state. The feature extractor is implemented in software and the performance indices given by it are passed to the identifier which is implemented in the form of a neural network. The performance indices used to characterize the time response are: a) *Delay time*, which is a measure of the time required by the response to reach 50% of the final value in the first attempt; b) *Rise time*, which measures the time required by the response to rise from 10% to 90% of the final value; c) *Settling time*, which is a measure of the time required by the response to reach and stay within a specified tolerance band (usually 2% to 5%) of its final value; d) *Peak*, which is the maximum value of the response (For the overdamped case and the critically damped case the peak value is assumed to be unit); e) *Peak time*, which is the time required for the response to reach its peak value and f) *Maximum Overshoot*, which is the largest deviation of the output over the step input during

the transient state (% maximum overshoot = (Maximum overshoot / final value) x 100). The feature extractor keeps track of the response values and the time at each sampling instant over an *adaptive time window* for the step input and a prespecified time window for the square function and the square wave. Since the majority of the performance indices depend on the transient part of the response, it seems valid to fix the size of the underlying time window. The sampling frequency is chosen to be well over the Nyquist frequency of the response. This enables the feature extractor to compute the various performance indices with a desirable accuracy.

Neural Network : The *system identifier* in the control loop is implemented in the form of the ART-II neural network,[Grossberg & Carpenter, 1987(a),87(c),87(d); Grossberg,1988]. The ART-II is a member of the class of Adaptive Resonance architectures that is designed to handle both binary and analog patterns and is a modification over the first proposed ART architecture called ART-I [Grossberg & Carpenter,1987(b)]. *See appendix for the mathematical model of the neural network.*

Training : The ART-II neural network is implemented on a digital computer. The network is used in the *supervised learning mode* and is trained off line before its inclusion into the control loop. The training data for the net is provided by a *training data module* also implemented digitally on a computer. It is also possible to train the network on line if the *system emulator* is included in the overall control scheme shown in figure 3. The training data module comprises of a data generator whose inputs form the approximate ranges of the system parameters which, in the present application, are the *natural frequency* and the *damping ratio* of the generalized second order system given by:

$$G(s) = \left[\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}\right]$$

A typical range of the natural frequency is estimated from the knowledge the plant time constant used in the process. The damping factor is assumed to vary between $0 < \zeta < 2.0$. The ranges selected thereof cover approximately the entire gamut of the systems time response to maximum deviation in the plant parameters.

The *system emulator* consisting of the process model (the generalised transfer function of the system, which in our case is second order) generates the simulated system

response to the various input signals using different settings of the parameters ζ and ω_n selected from respective parametric ranges specified.

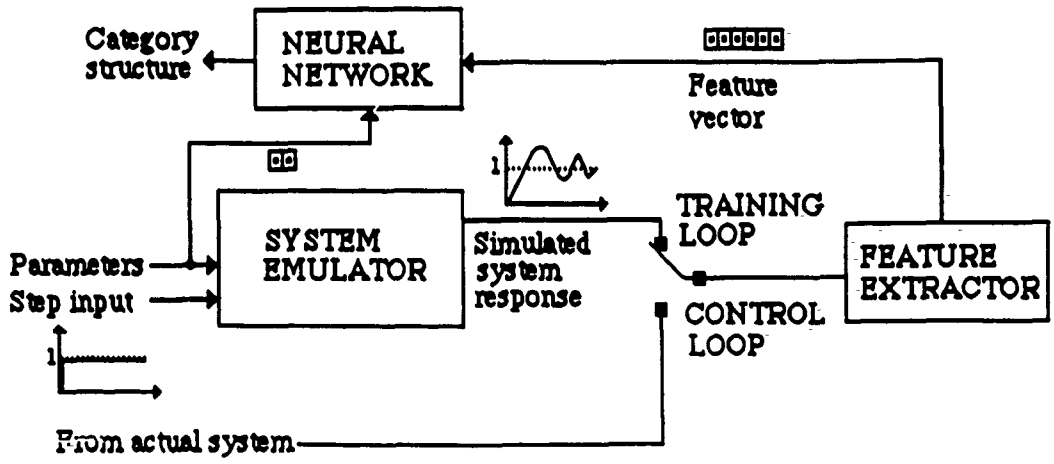


Figure 5 Block diagram of network training module comprising of the system emulator the feature extractor and the neural network.

To start data generation, parameters ζ and ω_n are set at their initial values and the standard input signal (the step input) is applied. The input to the *feature extractor* is the simulated system response and its output is a feature vector comprising of the response's performance indices. These along with the respective system parameters ζ and ω_n form the *training data set or the prototype set of patterns* for the neural network. See table 1 for a cross section of typical network training data, generated by the data generator.

CROSS SECTION OF NETWORK TRAINING DATA							
PARAMETERS		FEATURE VECTOR					
ζ	ω_n	peak	peak time	overshoot	rise time	settling time	delay time
0.30	1.50	1.37	2.20	0.37	1.31	8.80	0.80
0.40	1.50	1.25	2.28	0.25	1.44	6.67	0.84
0.50	1.50	1.16	2.40	0.16	1.61	5.41	0.89
0.60	1.50	1.09	2.60	0.09	1.85	4.59	0.94
0.70	1.50	1.05	2.92	0.05	2.19	4.05	1.00
0.80	1.50	1.02	3.48	0.02	2.78	3.69	1.05
0.90	1.50	1.00	4.80	0.00	4.12	3.51	1.11
1.00	1.50	1.00	0.00	0.00	1.88	19.96	0.52
1.10	1.50	1.00	0.00	0.00	2.92	19.96	0.80
1.20	1.50	1.00	0.00	0.00	3.52	19.96	0.96
0.70	2.00	1.05	2.20	0.05	1.64	3.03	0.75
0.80	2.00	1.02	2.60	0.02	2.08	2.76	0.79
0.90	2.00	1.00	3.60	0.00	3.09	2.63	0.83
1.00	2.00	1.00	0.00	0.00	1.40	19.96	0.36
1.10	2.00	1.00	0.00	0.00	2.20	19.96	0.60
1.20	2.00	1.00	0.00	0.00	2.64	19.96	0.72
1.30	2.00	1.00	0.00	0.00	3.00	19.96	0.84

Table 1 A cross section of typical training data generated by the network training module for $0.30 \leq \zeta \leq 1.30$, $\Delta\zeta = 0.10$ and $1.50 \leq \omega_n \leq 2.00$, $\Delta\omega_n = 0.50$. The feature vector corresponding to different settings of ζ and ω_n represents the performance indices of the response that are obtained from the feature extractor.

Once the training data is available, the network is configured (*see appendix for details on the network architecture*) such that the number of nodes in the *feature representation field* F_1, corresponds to the dimension of the input feature vector, which in the present application is fixed at six. The number of processing nodes in the *category representation field* F_2, is generally greater than total number of input patterns in the prototype set. Each pattern in the prototype set is sequentially presented to the network once. A second cyclic presentation of the prototype set may be made for a stable category confirmation. Subsequent presentations do not alter the resulting category structure. The *category structure* represents the state space partitioning of the neural network depending on the number of stable categories established during training and the different feature vectors that were classifed into these categories. During training, the *attentional vigilance parameter* is set at its highest value (0.99) to ensure a high resolution of the resulting category structure. The network associates with each new category established the system parameters ζ and ω_n that were responsible for the generation of the corresponding feature fector.

When the network is presented with a feature vector for the first time, it is encoded in the LTM through modification of the LTM connection weights. A node is allocated in the networks category representation field F_2 to represent the pattern. The parameters associated with the feature vector now get assigned to this allocated F_2 node. On presentation of subsequent feature vectors, the network's *orienting subsystem* first determines closeness of match between the pattern currently imposed on the network and any of the patterns that have previously been seen. Since the vigilance parameter is set high a new node is allocated for the pattern. However, if the current pattern happens to be closely matched to the one the network has already seen, it is clustered into the same category. It is therefore possible to partition the networks state space such that each partition serves as an attractor for a particular type of response characterized by its feature vector. The vigilance parameter helps to control the coarseness or fineness of classification desired. After completion of training the *top down* and the *bottom up connection weights* of the network along with the network parameters are saved into the computer memory. The above process is repeated for different sets of values for ζ and ω_n taken from their respective ranges. Increments in the parameteric values of the system during training depend upon the trade off between accuracy of identification desired and the size constraints on the network design.

Testing : When the network is introduced in the control loop, identification proceeds almost instantly. Search for the category associated with the right parameters is achieved by dynamically altering the attentional vigilance parameter until an "optimal vigilance" is found. The testing procedure is depicted through the flow diagram in figure 6.

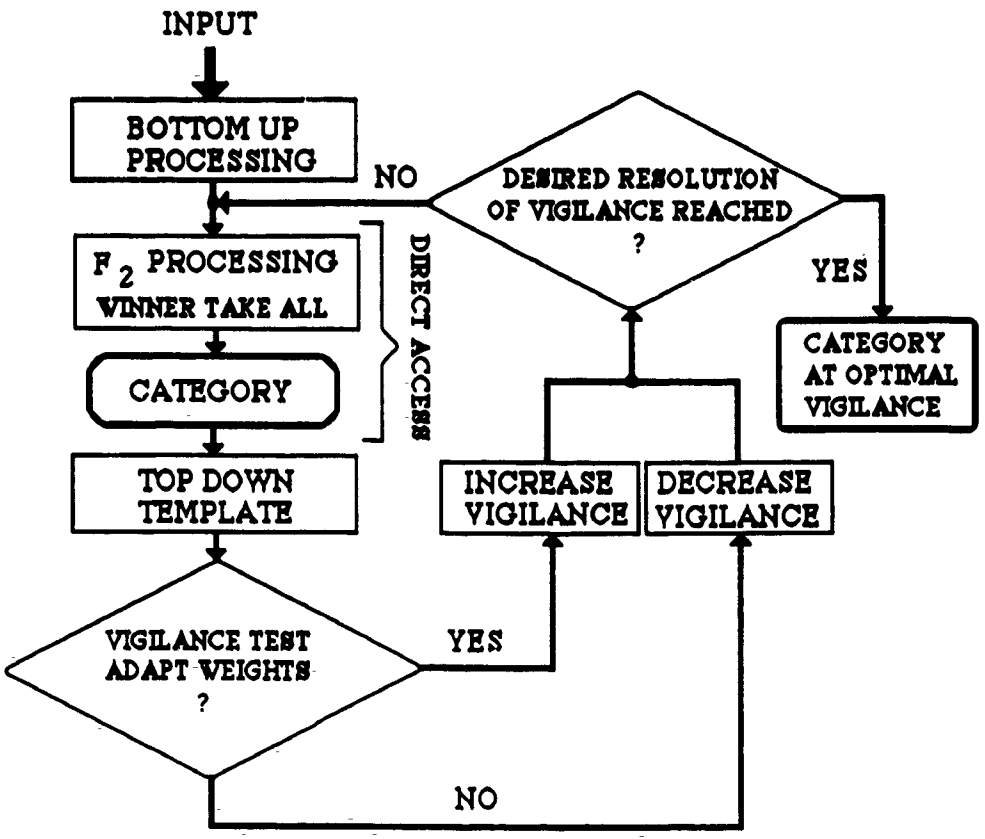


Figure 6 The optimal vigilance category search procedure

4 EXPERIMENTAL RESULTS

In this section we report the various experimental results obtained from computer simulations of the process described in sections 2 and 3 of the paper. We have described the simulation procedure and provided an explaination of the various results obtained.

The time response of the system was obtained using a Runge-Kutta fourth order differential equation solver (RK-4). Athough the simulations carried out on the sequential computer were slow, the results obtained suggest that a hardware implementation of the system identification module would lead to much faster identification owing to the parallel distributed nature of information processing within the neural network. Throughout the simulations sensor and plant noise were assumed negligible.

The neural network employed is trained off-line on the features of the response obtained from the generalized second order system with the following parameteric variations:

$$0.1 \le \zeta \le 1.50,\ \Delta\zeta = 0.10$$
$$0.5 \le \omega_n \le 2.50,\ \Delta\omega_n = 0.50 \qquad (15)$$

These parameters are not made available to the system identifier but are estimated through the neural network in the control loop, based upon the features of the system time response to the standard test signals. The only available information to the feature extractor module is the discrete time instant and the value of the system response at these time instants.

We have assumed our *ideal system* to have the following parameters :

Desired damping, $\zeta^* = 0.70$

Desired frequency, $\omega_n^* = 1.50$

Plant constant, $K_m = 1.00$

Plant parameter, $\alpha = 2.10$

Plant parameter, $\beta = 1.50$

Specification of plant parameters α and β determines the desired location of the system closed loop poles within the left half of the S- plane and enables in the computation of the desired damping , ζ^* and the desired natural frequency, ω_n^* of the system response.The responses of a system with the above set of parameters to a unit step input and the square input are depicted in figures 7 and 8 respectively. We call these responses " nominal responses of the assumed ideal model ".

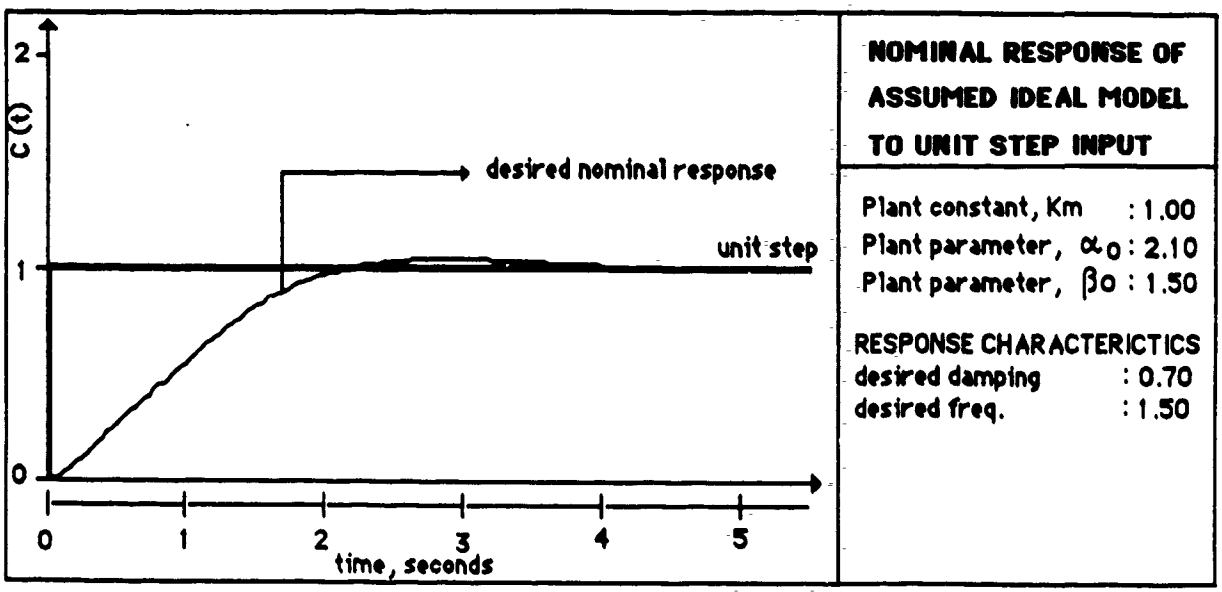


Figure 7 The desired system response of the assumed ideal model to a unit step input, shown in bold. The damping coefficient of the response, $\zeta = 0.70$ and the natural frequency, $\omega_n = 1.50$.

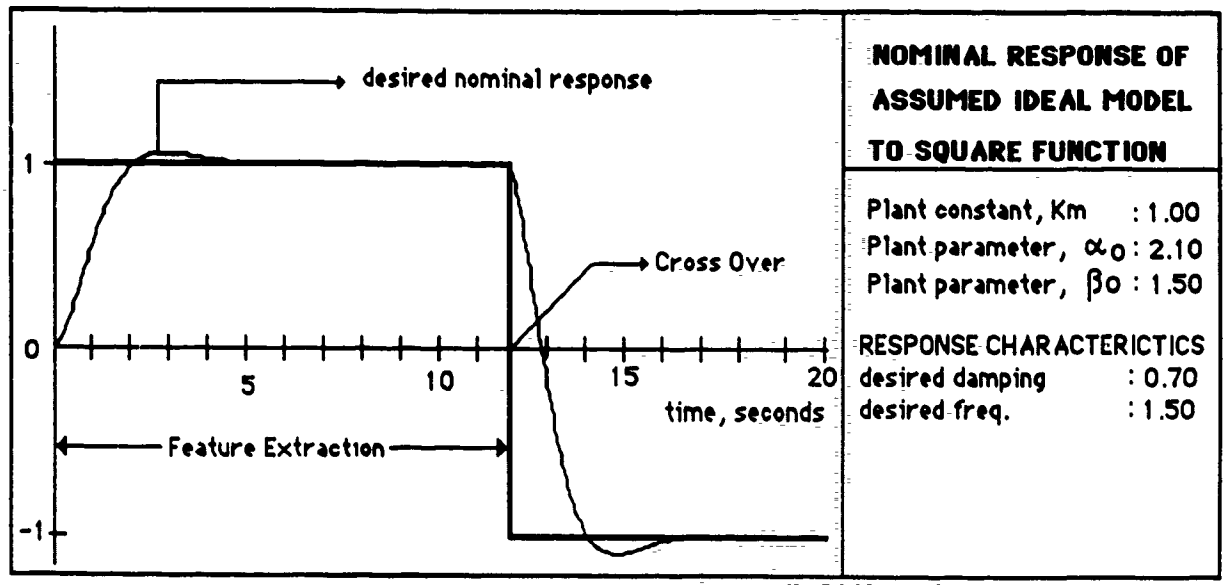


Figure 8 The desired system response of the assumed ideal model to a square function input, shown in bold. The damping coefficient of the response, $\zeta = 0.70$ and the natural frequency, $\omega_n = 1.50$. Feature extraction to this type of input is restricted to the first 12 seconds of the input profile before the cross over.

The transient response of a system that has been disturbed by an instantaneous change in the reference input, like the unit step, constitutes the most interesting and most significant part of the dynamic behavior of the system. In the present application, for the unit step input, the information contained in entire transient response was captured for .

feature extraction by making the time window variable depending upon the duration of the transient. In the case of the square function and the square wave, feature extraction was restricted to the step portion of the response (12 seconds).

Two time varying functions are considered to represent the variation in the plant parameters α and β. The plant parameter Km is kept constant. Figure 9 shows a linear variation while a slow periodic variation is depicted in figure 10. It should be noted that since parameter estimation takes place only during the step portions of the input signal and that no restrictions are placed on the control signal u(t), it is possible to add a 'probing' signal to the input at fixed time intervals to enable system identification when the input profile is arbitrary. The performance of the control scheme to inputs of this kind is currently under investigation.

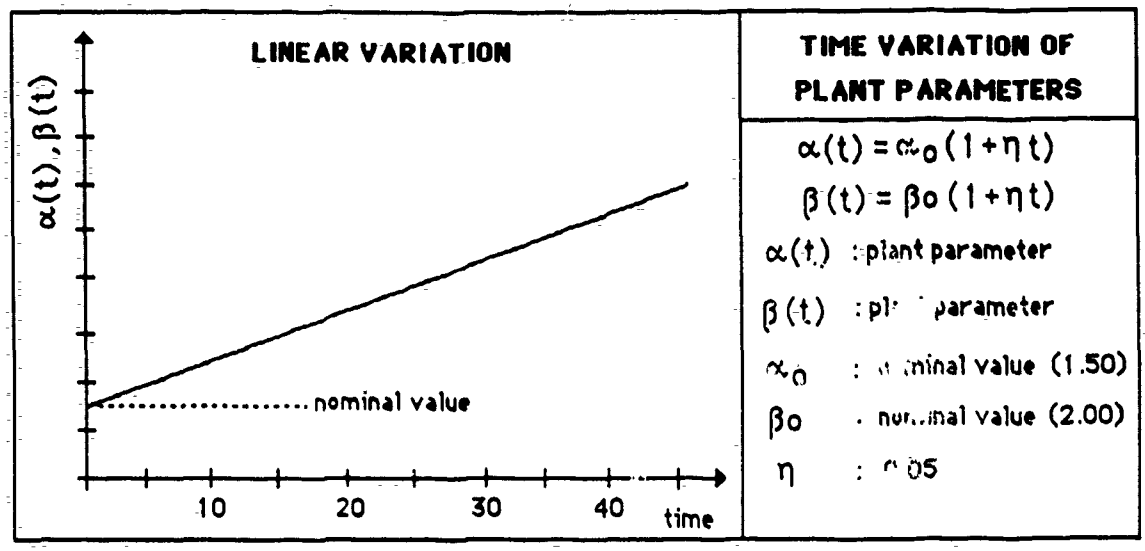


Figure 9 Linear time variation of plant parameters α and β.

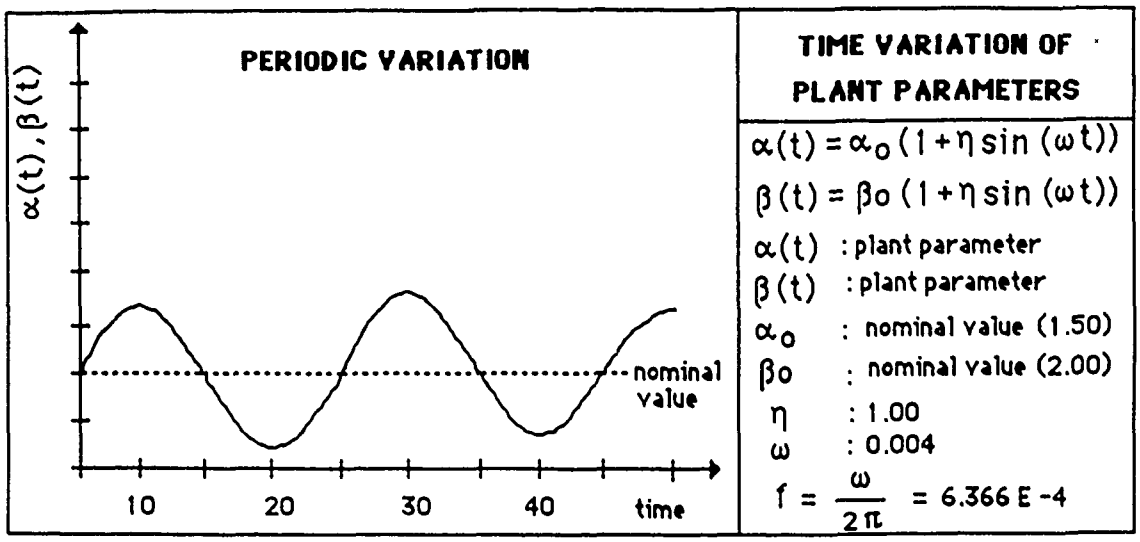


Figure 10 Periodic sinusoidal time variation of plant parameters α and β.

Since the components of the feature vector are not all independent and since no effort is made to determine whether a particular prototype training set forms a complete basis for all input vectors encountered by the controller in a particular situation, a network sensitivity study was carried out. The objective of this study was to determine which components of the feature vector influenced the network most in terms of the number of categories established. Figure 11 shows the sensitivity of an ART-II neural network to the different components of the feature vector. Seventeen prototype feature vectors were presented to the network at a vigilance of 0.995. The graphs depict variation of the different performance indices for the feature vectors versus the categories established. The sensitivity plots indicate that the network was not uniformly sensitive to all components of the feature vector, especially in the overdamped case ($\zeta > 1$) when most responses look very much alike. The network was found to be most sensitive to the delay time, rise time and the peak time of the response as shown in the figure. To enhance the feature vector uniformly it was passed through a nonlinear function given by:

$$f_i(x) = e^{\gamma x} \text{ where, } \gamma = 0.05 \qquad (16)$$

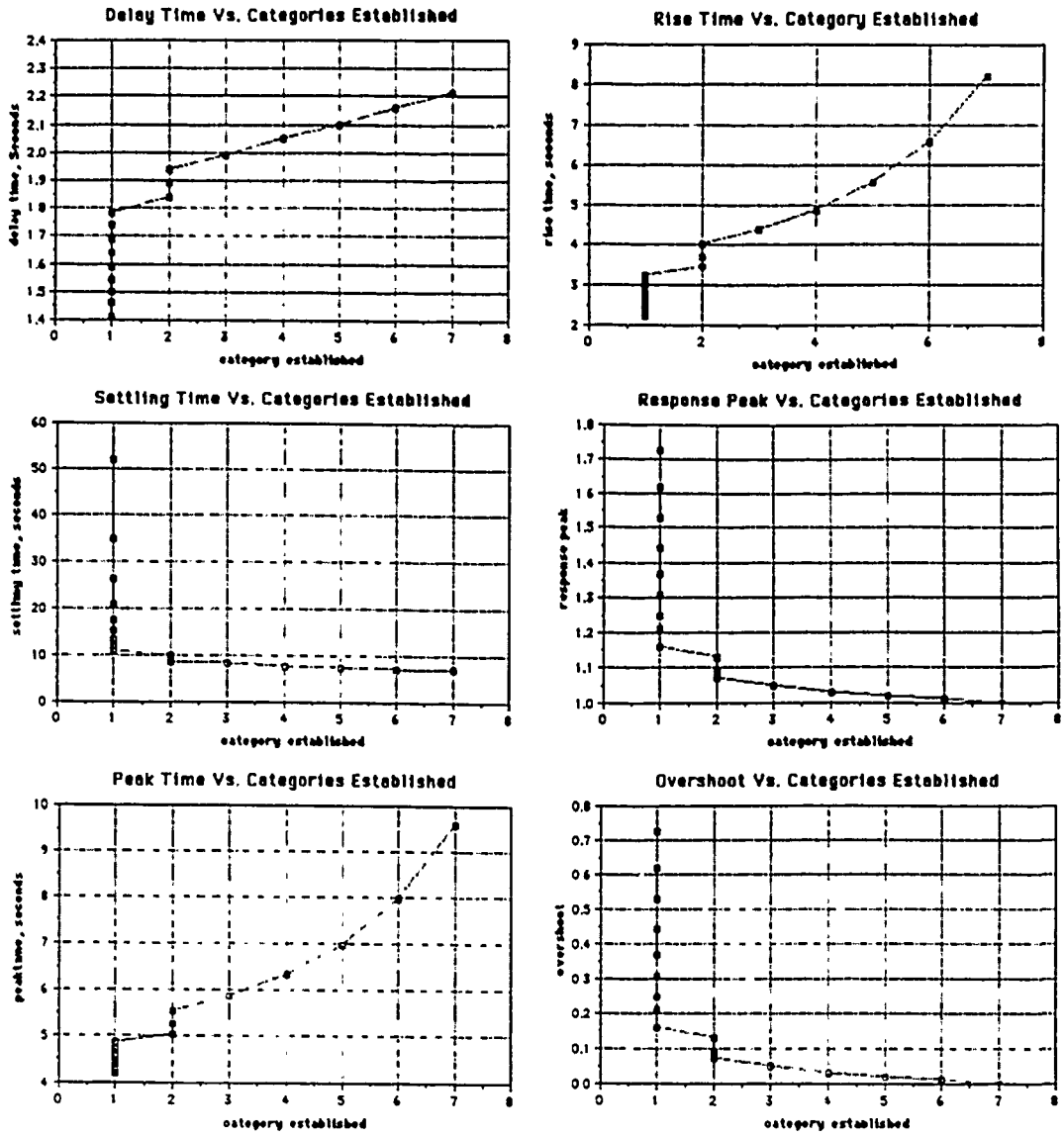


Figure 11 The above plots indicate the sensitivity of the neural network to changes in the various performance indices of the system time response at a constant attentional vigilance of 0.995. The input pattern set comprised if 17 different input patterns (number of points plotted on the graphs) which are presented to the network once cyclically. The abscissa represents the identification number of the category established while the ordinate shows the actual value of the performance index under consideration.

Figures 12 through 18, each depict two plots. (a) The time response of the actual system to the unit step input signal with arbitrary initial parameters α, β, and K_m, and (b) the final system response with the *D.C. bias* or the *steady-state error* compensated through the parameter C. The uncompensated final response with only the controller parameters K_p and K_d adjusted after the system identification and pole placement is completed is not shown in the plots.

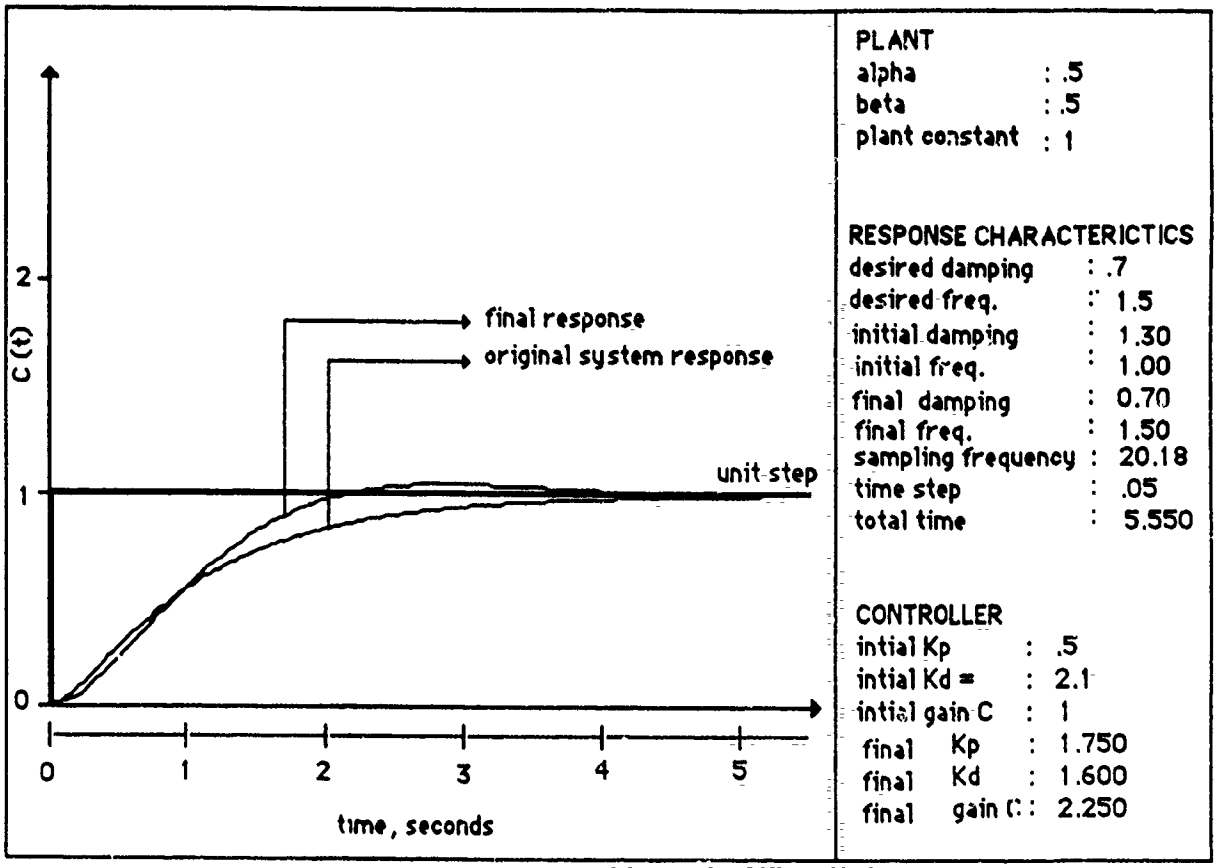


Figure 12 An overdamped system response with arbitrary initial plant parameters, $\alpha = 0.5$, $\beta = 0.5$ and $K_m = 1.00$ and initial controller gains $K_p = 0.50$, $K_d = 2.10$. The response has an initial damping coefficient $\zeta = 1.30$ and an initial natural frequency $\omega_n = 1.00$. The final controller parameters are those obtained after the plant parameters are estimated by the neural network system identifier and after the relocation of the overall closed loop system poles by the pole placement module. The final controller gains correspond to $K_p = 1.750$ and $K_d = 1.60$ and the D.C. bias, $C = 2.250$. The final response has the desired damping coefficient $\zeta = 0.7$ and the desired natural frequency $\omega_n = 1.50$ corresponding to the ideal system response. The time scale shown in the figure is that of the final system response. The time scale incorporated in the simulation is adaptive for the unit step input and depends upon the duration of the system transient response. The sampling frequency selected is common to both the response sampling rate and the RK-4 differential equation solver and is set well above the Nyquist frequency of the system.

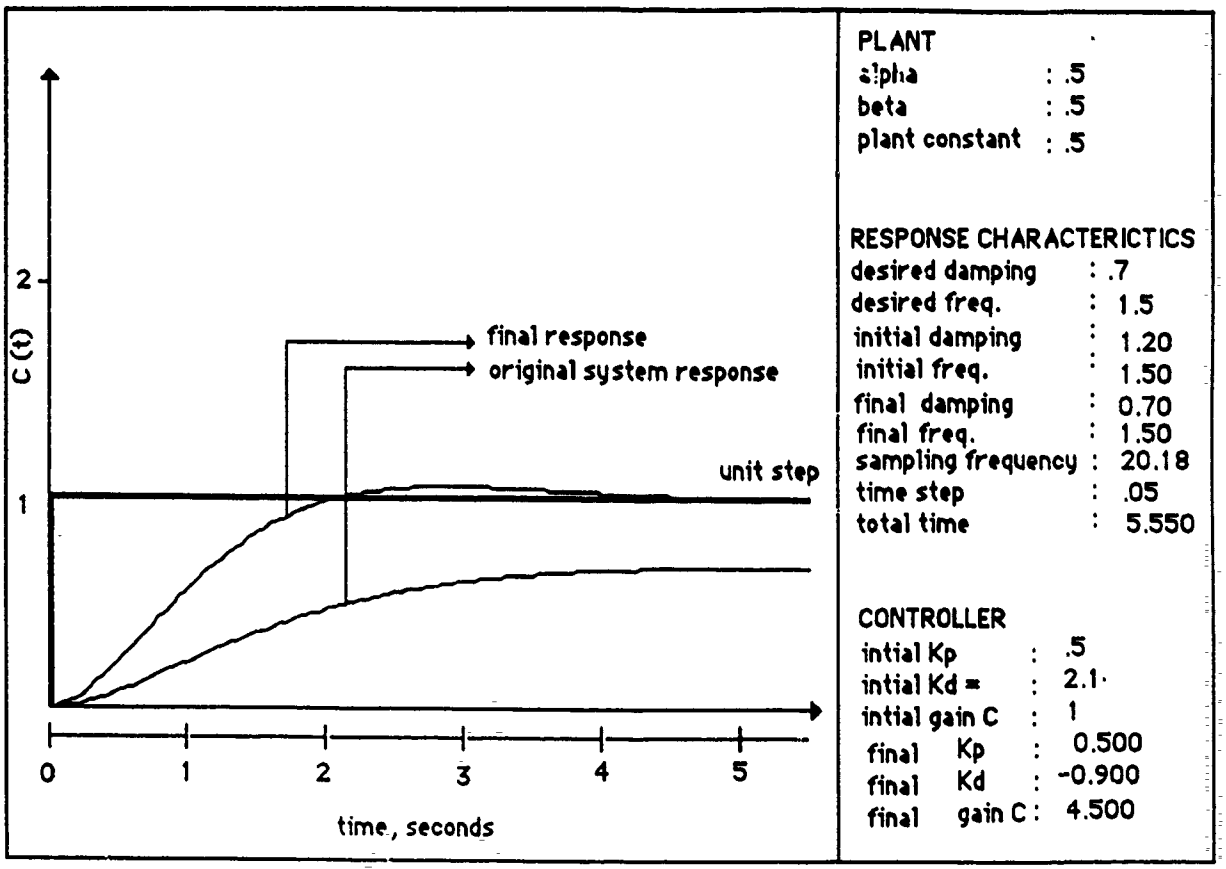


Figure 13 An example of an overdamped system response with arbitrary initial plant parameters, $\alpha = 0.5$, $\beta = 0.5$ and $K_m = 0.5$ and initial controller gains $K_p = 0.50$, $K_d = 2.10$. The response has an initial damping coefficient $\zeta = 1.20$ and an initial natural frequency $\omega_n = 1.50$.The final controller gains are $K_p = 0.50$ and $K_d = -0.900$ and the D.C. bias, $C = 4.50$. The final response has the desired damping coefficient $\zeta = 0.7$ and the desired natural frequency $\omega_n = 1.50$ corresponding to the ideal system response.

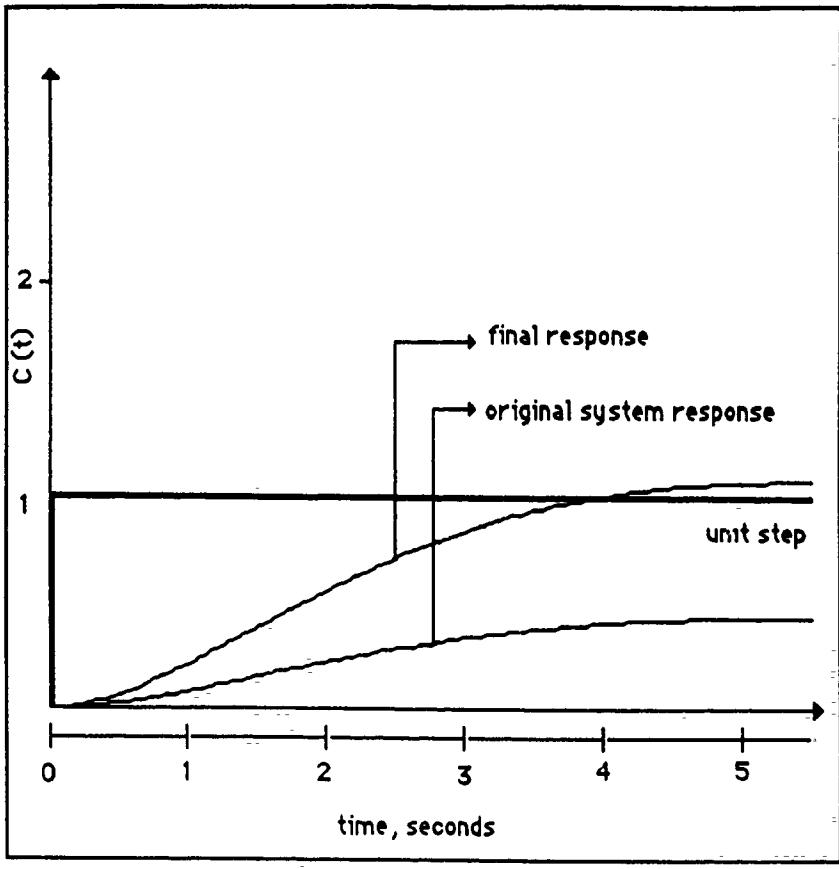
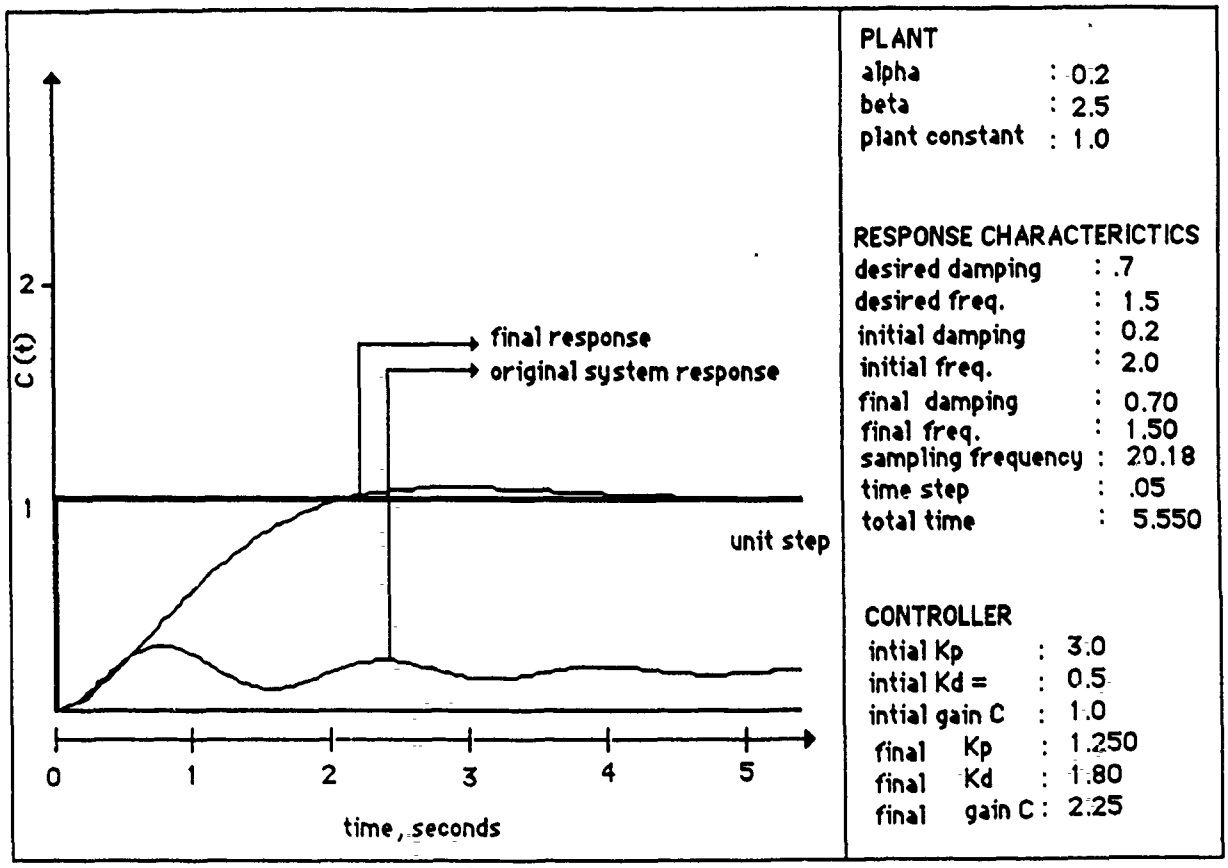
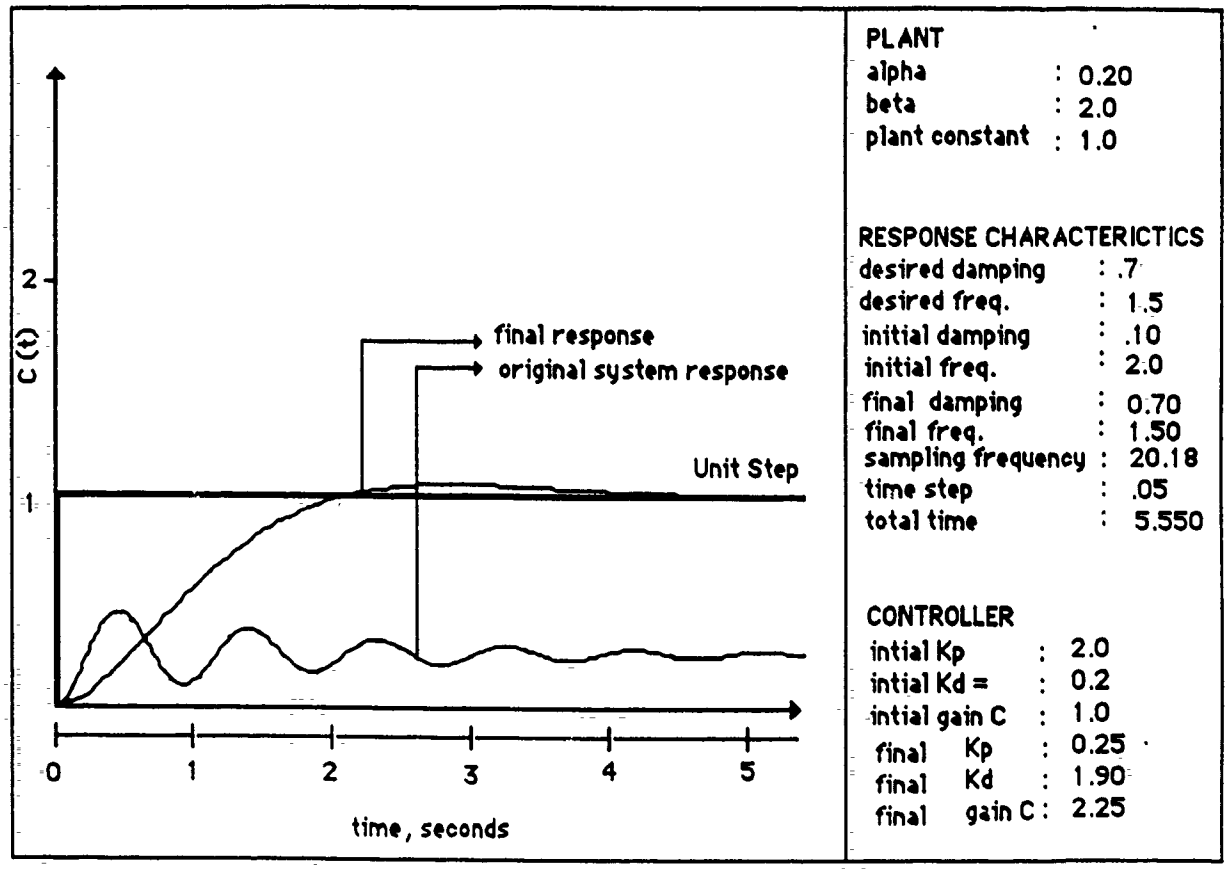


Figure 14 An example where the identification provided by the network is not accurate owing to the discrete nature of partitioning provided by the neural network identifier. A finer resolution can be obtained if the size of the neural network is increased by increasing the number of nodes at the category representation field F_2. This would eliminate the possibility of incorrect classification by the network as demonstrated in this example. The arbitrary initial plant parameters are $\alpha = 0.5$, $\beta = 0.5$ and $K_m = 0.25$ and initial controller gains $K_p = 0.50$, $K_d = 2.10$. The response has an initial damping coefficient $\zeta = 1.20$ and an initial natural frequency $\omega_n = 1.50$.The final controller gains are $K_p = 0.50$ and $K_d = -0.900$ and the D.C. bias, $C = 4.50$. The final response has a damping coefficient $\zeta = 0.8$ and a natural frequency $\omega_n = 1.00$. The desired damping and natural frequency corresponding to the ideal system response are $\zeta = 0.7$ and $\omega_n = 1.50$.



Figures 15 figure depicting the ability of the proposed adaptive controller to modify an underdamped system response. The arbitrary initial plant parameters are $\alpha = 0.2$, $\beta = 2.5$ and $K_m = 1.00$ and initial controller gains $K_p = 0.30$, $K_d = 0.5$. The response has an initial damping coefficient $\zeta = 0.20$ and an initial natural frequency $\omega_n = 2.00$. It should be noted that parameter identification of an underdamped response is easier when compared to the overdamped and critically damped cases in which the system responses look alike. In the underdamped case, the overshoot makes one of the feature vector components more significant. The final controller gains are $K_p = 1.25$ and $K_d = 1.80$ and the D.C. bias, $C = 2.25$. The final response has a damping coefficient $\zeta = 0.7$ and a natural frequency $\omega_n = 1.50$ corresponding to the ideal system response.



Figures 16 Example of a highly oscillatory underdamped system response with arbitrary initial plant parameters are $\alpha = 0.2$, $\beta = 2.0$ and $K_m = 1.00$ and initial controller gains $K_p = 2.00$, $K_d = 0.2$. The response has an initial damping coefficient $\zeta = 0.10$ and an initial natural frequency $\omega_n = 2.00$. The final controller gains are $K_p = 0.25$ and $K_d = 1.90$ and the D.C. bias, $C = 2.25$. The final response has a damping coefficient $\zeta = 0.7$ and a natural frequency $\omega_n = 1.50$ corresponding to the ideal system response.

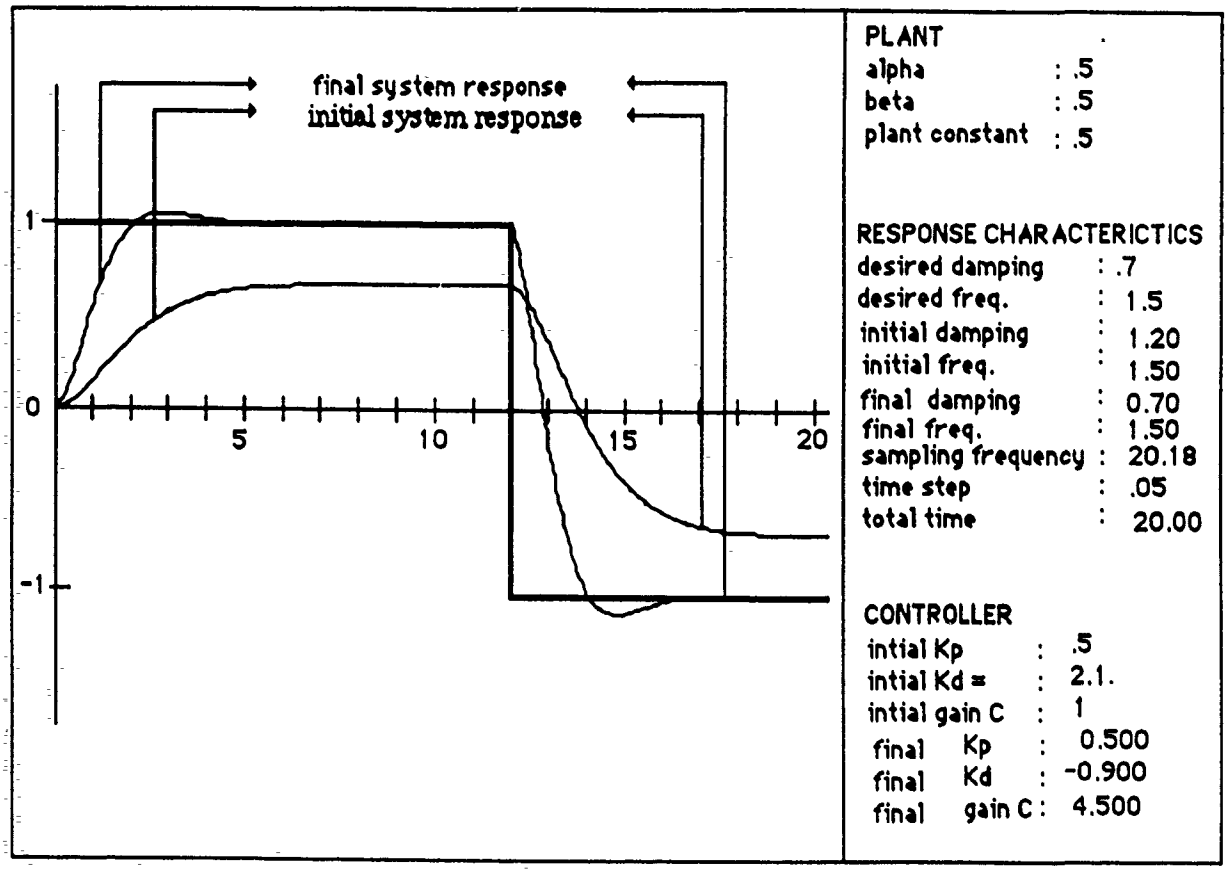


Figure 17 System response to a square input function with arbitrary initial plant parameters are $\alpha = 0.5$, $\beta = 0.5$ and $K_m = 0.50$ and initial controller gains K_p 0.50, $K_d = 2.10$. The initial response of the system is first obtained and the portion of the response corresponding to the first 12 seconds (upto the cross over) is selected for feature extraction and system identification.It has an initial damping coefficient $\zeta = 1.20$ and an initial natural frequency $\omega_n = 1.50$. The response after the controller gains have been adjusted by the pole placement module is depicted as the final system response. The final controller gains are $K_p = 0.50$ and $K_d = -0.90$ and the D.C. bias, $C = 4.50$. The final response matches the desired response given by the assumed ideal model which has a damping coefficient $\zeta = 0.7$ and a natural frequency $\omega_n = 1.50$. Note that the parameter K_d is negative implying negative derivative feedback.

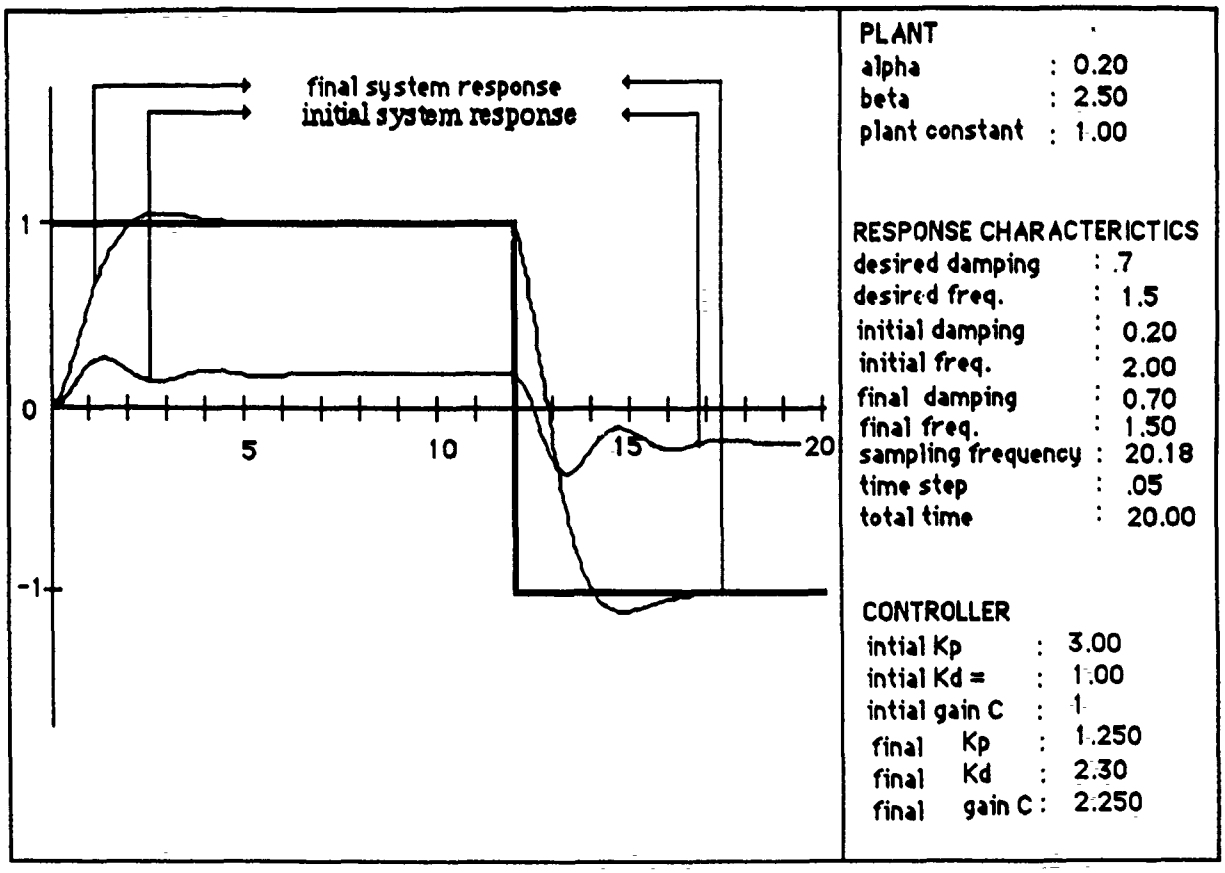


Figure 18 An underdamped system response to a square input function with arbitrary initial plant parameters are $\alpha = 0.2$, $\beta = 2.5$ and $K_m = 1.00$ and initial controller gains $K_p = 3.00$, $K_d = 1.00$. The initial damping coefficient $\zeta = 0.20$ and the initial natural frequency $\omega_n = 2.00$. The final controller gains are $K_p = 1.25$ and $K_d = 2.30$ and the D.C. bias, $C = 2.25$. The final response has a damping coefficient $\zeta = 0.7$ and a natural frequency $\omega_n = 1.50$ corresponding to the ideal model.

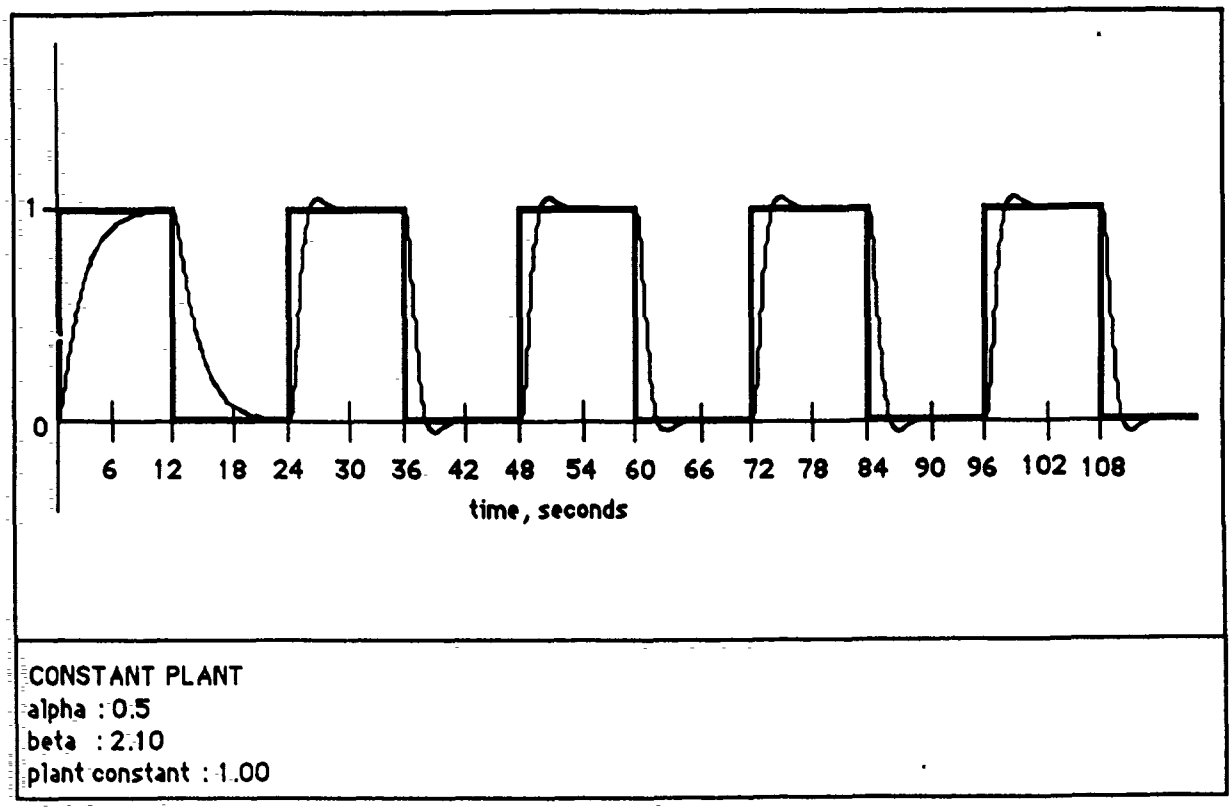


Figure 19 Response of the system to a square wave input. The plant parameters are kept constant throughout with $\alpha = 0.50$, $\beta = 2.10$ and $K_m = 1.00$. Parameter estimation takes place once after the first 24 seconds elapse. This constitutes the parameter estimation interval of the response. The response shown during the first 24 seconds is the unregulated response of the system during which no feature extraction or system identification takes place. The rest of the output profile corresponds to the desired response with the controller parameters tuned and the D.C. bias compensated.

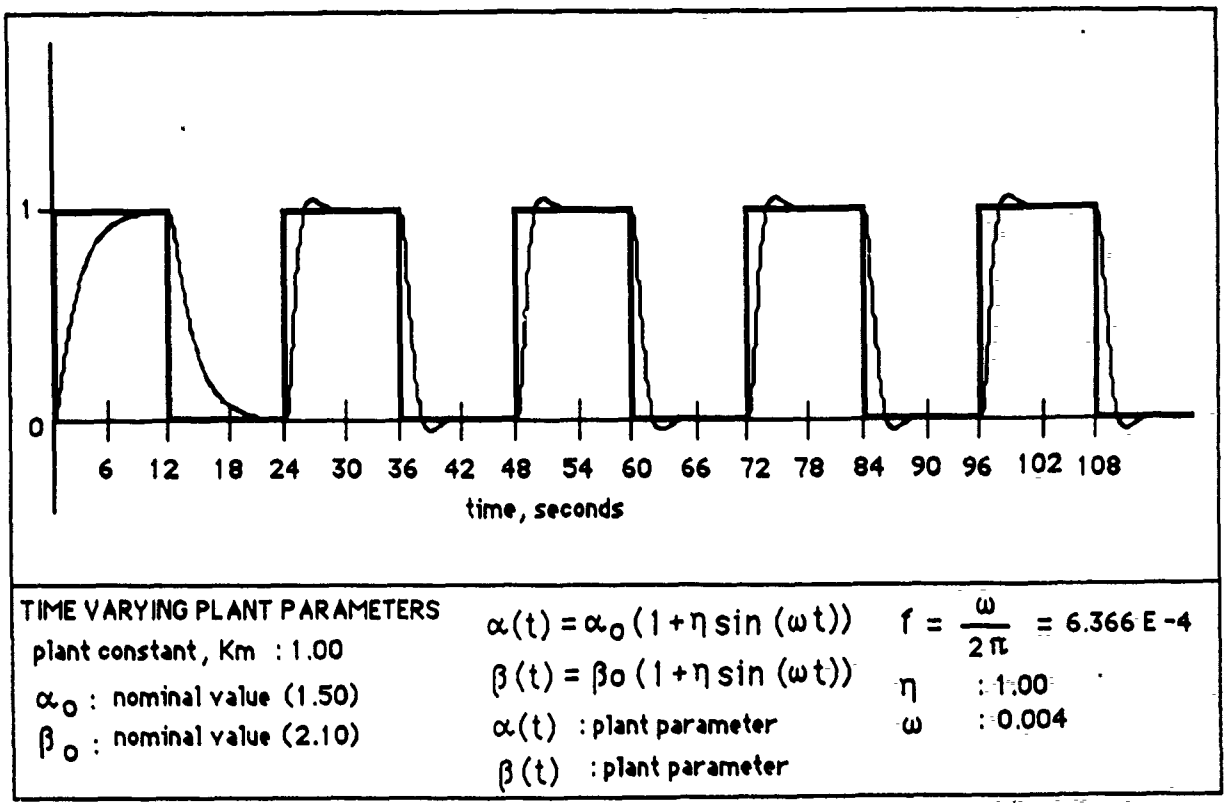


Figure 20 Response of the system to a square wave with a sinusoidal variation of plant parameters as depicted in Figure 11. Since there is no probing signal, parameter estimation takes place discretely at a time intervals of 24 seconds. Here again feature extraction is restricted to the step portion of every pulse. The response shown in the first 24 seconds is the unregulated response of the system during which no feature extraction or system identification takes place. The rest of the response profile corresponds to the desired response with the controller parameters tuned and the D.C. bias compensated every 24 seconds. The change in the system parameters are not quite apparent due to the fact that variation in parameter is small with respect to the resolution of the parametric space of the neural network.

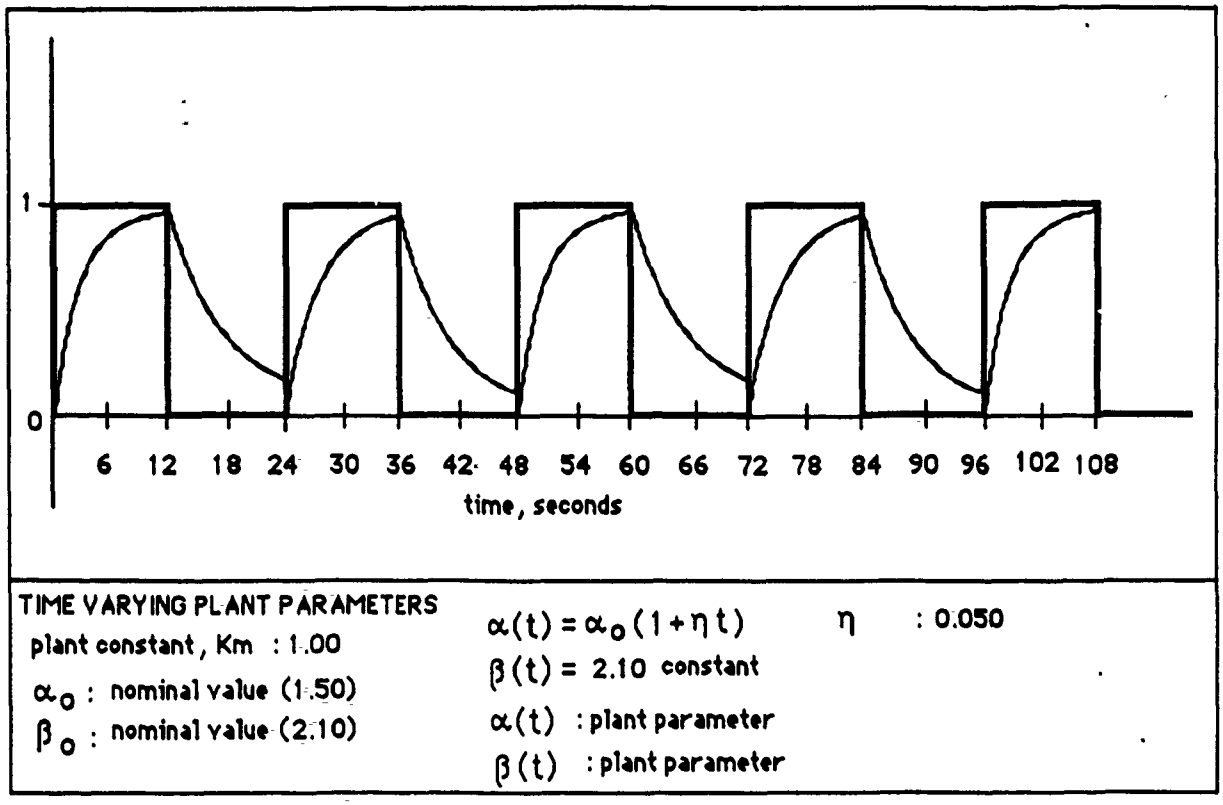


Figure 21 System response to linear time variation in one of the parameters, α while K_m and β are held constant. It should be noted that the response should settle to zero before the next pulse is applied to ensure proper identification.

5 CONCLUSION AND FUTURE WORK

A new approach for dynamic system identification has been attempted that involves the application of a neural network architecture based on the Adaptive Resonance Theory. It has been shown that it is possible to train the ART neural net offline (supervised learning mode) to identify the parameters of a simple second order system based upon attributes of its response to standard test signals. A simple indirect adaptive control scheme was formulated, implemented and tested to assess the performance of the network that was incorporated as the online system identifier in the control loop. Experimental results suggest that identification provided by the network is accurate within the resolution of the training data. The off line training of the network was found to be fast with the training experimental data set being presented to the network only once. During testing when the network is included in the control loop, the search procedure adopted helps to improve the identification provided by the network. The application of the proposed scheme based upon the ART architecture has to be studied further by applying it to other higher order systems. Future work also includes a critical comparison between the identification techniques adopted in this work and the other conventional methods such as the method of Least Squares on the same set of problems. Such an exercise would bring into sharper focus the relative merits and demerits of the proposed neural network based identifier. It would also be useful to study the possibility of implementing the network such that it is made capable of learning online. Future work also includes an evaluation of the proposed scheme to input signal profiles that are not regular via overriding probing signals.

6 REFERENCES

Astrom, K. J. (1983). Theory and Applications of Adaptive Control, *Automatica*, Vol. 19, No. 5, pp. 471-486.

Astrom, K. J. and Wittenmark, B. (1973). On Self Tuning Regulators, *Automatica*, Vol. 9, pp. 185-199.

Astrom, K. J. and Eykhoff, P. (1971). System Identification-A Survey, *Automatica*, Vol. 7, pp. 123-162.

Astrom, K. J. and Wittenmark, B. (1971). Problems of Identification and Control, *Journal of Mathematical Analysis and Applications*, Vol. 34, pp. 90-11.

Astrom, K. J. and Wittenmark, B. (1984). Computer Controlled Systems-Theory and Design, Prentice-Hall Information and System Science Series.

Goodwin, G.C. and Payne, R. L. (1977). Dynamic System Identification- Experiment Design and Data Analysis, *Mathematics in Science and Engineering*, Academic Press, Vol. 136.

Grossberg ,S. and Carpenter, G. A. (1987 (a)). ART:2 Self Organization of Stable Category Recognition Codes for Analog Input Patterns, *Applied Optics*, Vol. 26. No. 23, pp.4919-4930.

Grossberg ,S. and Carpenter, G. A. (1987(b)). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine,*Computer Vision Graphics & Image Process.*, 37, 54 .

Grossberg ,S. and Carpenter, G. A. (1987(c)). ART 2: Stable Self Organization of Pattern Recognition Codes for Analog Input Patterns, *Proc. of First International Conference on Neural Network*, San Deigo (IEEE, New York).

Grossberg ,S. and Carpenter, G. A. (1987(d)). Invariant Pattern Recognition and Recall by an Attentive Self Organizing ART Architecture in a Nonstationary World, *Proc. of First International Conference on Neural Network*, San Deigo (IEEE, New York).

Grossberg, S. (1988). *The Adaptive Brain*, North-Holland.

Guez, A. and Selinsky, J. (1988). A Trainable Neuromorphic Controller, *Journal of Robotics systems*, pp. 363-388.

Guez, A. (Oct. 1988). Neurocontroller, *NSF workshop on Neural Networks in Robotics and Control*, New Hampshire.

Kumar, S. S. and Guez,A. (1989). A Neural Network Approach to Target Recognition, *Proc. of IEEE International Joint Conference on Neural Networks*,Washington D.C., USA, pp. II-573.

Kumar, S. S. and Guez,A. (1990). Neural Network based Dynamic System Identification for Adaptive Control, *Proc. of IEEE International Joint Conference on Neural Networks*,Washington D.C., USA, pp. II-792.

Landau, Y.D. (1979). Adaptive Control- The Model Reference Approach, Marcel Dekker Inc., *Control and System Theory*, Vol. 8.

Lippmann, R. P. (1987). An introduction to computing with neural networks, *IEEE ASSP Magazine*, April.

Ortega, R. and Kelly, R. (1984). PID Self-Tuners: Some Theoretical and Practical Aspects, *IEEE Transactions on Industrial Electronics*, Vol.IE-31,No.4, pp.332 -337.

Ortega, R. and Spong, M. W. (1988). Adaptive Control of Rigid Robots, *Proc. 27th Conference on Decision and Control*, Austin, Texas.

Psaltis, D., Sideris, A. and Yamamura, A. (1987). Neural controllers, *Proceedings of the IEEE First International conference on Neural Networks*, San Diego, CA.

Rumelhart, D.E. and McClelland,J.L. (1986). *Parallel Distributed Processing: Exploration into the Microstructure of Cognition*, Vol. 1: Foundations, MIT press.

APPENDICES

The ART-II Mathematical model :

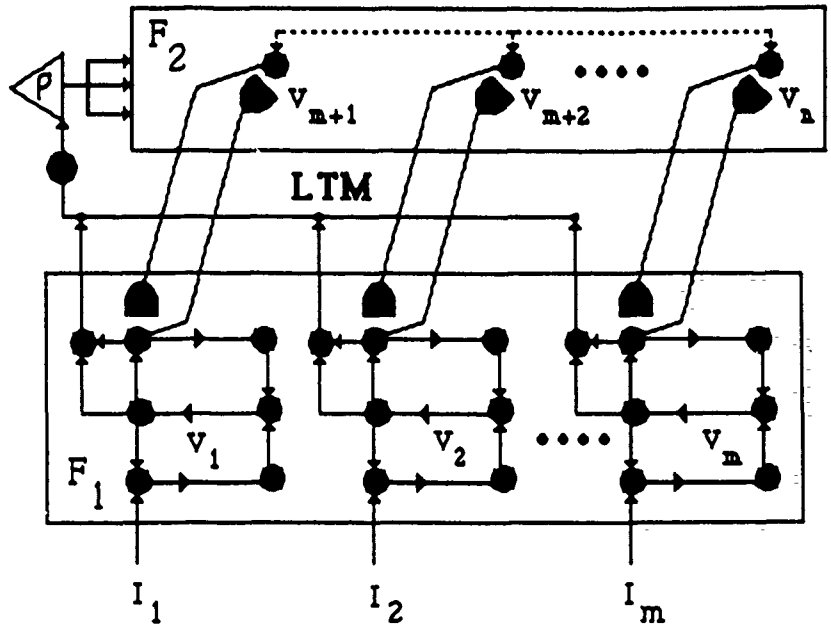


Figure 22 The ART-II Network structure

The equations presented in this section are taken from reference [Grossberg & Carpenter,1987(a)]. For the convinience of the reader we present the equations used in the our simulations in the form of the ART-II mathematical model. Figure 22 depicts the ART-II network architecture while figure 23 the network topology used in the simulations.

The global network parameters and constants are:

M : number of F_1 input channels

N : number of STM nodes at stage F_2

a : network parameter (a = 10.00)

b : network parameter (b = 10.00)

c : network parameter (c = 0.10)

d : network parameter (d = 0.9 (0 < d <1))

e : network parameter (e = 0.0)

ø : noise tolerance parameter (ø = 1 / √M)

ETP: error tolerance parameter (ETP = 0.05)

ρ : attentional vigilance parameter (0 ≤ ρ ≤1)

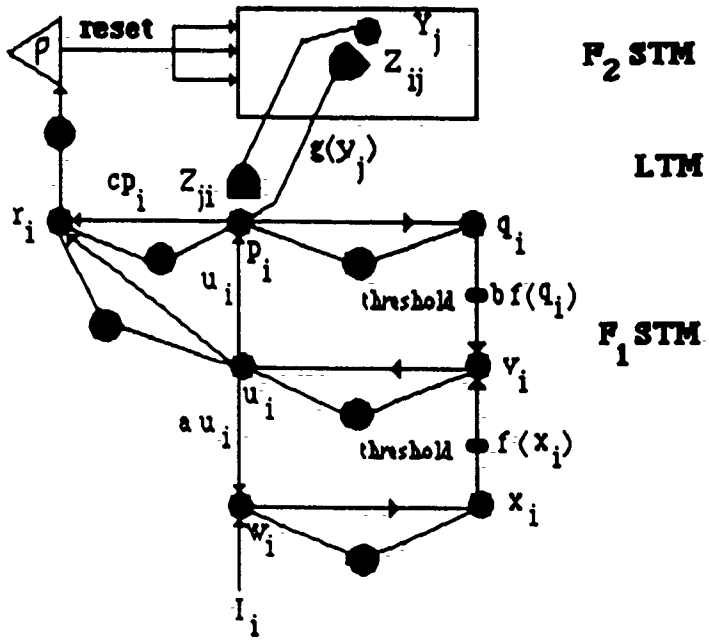


Figure 23 The ART-II Network topology

Network Equations:

Refer to ART-II network topology in figure 23. The local STM activities p_i, q_i, u_i, w_i and x_i computed at the STM stage F_1 (All equations expressed in dimensionless form) are :

For i = 1,.....,M and j = M+1,.....,N:

$$p_i = u_i + \sum_j g(y_j) z_{ji} \tag{17}$$

$$q_i = \frac{p_i}{e + |p|} \tag{18}$$

$$u_i = \frac{v_i}{e + |v|} \tag{19}$$

$$v_i = f(x_i) + bf(q_i) \tag{20}$$

$$w_i = I_i + au_i \tag{21}$$

$$x_i = \frac{w_i}{e + |w|} \tag{22}$$

$|V|$ represents the L_2 norm of the vector V.

$$f(x) = \frac{2\theta x^2}{(x^2+\theta^2)} \quad \text{if } 0 \le x \le \theta$$
$$= \quad x \quad \text{if } x \ge \theta \qquad (23)$$

where, f(x) is the thresholding function.
The local STM activity, T_j computed at the STM stage F_2 is given by:

$$T_j = \sum_i p_i Z_{ij} \qquad (24)$$

where Z_{ij} are the bottom up connection weights ($F_1 \rightarrow F_2$).

The LTM Equations :

Top Down (F2 → F1) : $\frac{d(Z_{ji})}{dt} = g(y_j)[p_i - Z_{ji}]$ (25)

Bottom Up ($F_1 \rightarrow F_2$) : $\frac{d(Z_{ij})}{dt} = g(y_j)[p_i - Z_{ij}]$ (26)

where,
Z_{ji} : Top down connection weights ($F_2 \rightarrow F_1$)
Z_{ij} : Bottom up connection weights ($F_1 \rightarrow F_2$)

$g(y_j) = d$ {if $T_J = \mathbf{Max}\ T_j$: the j^{th} F_2 node that has not been reset on current trial} (27)
0 otherwise

when F_2 makes a choice and the J^{th} F_2 node is selected during contrast enhancement process (winner take all or on center-off surround selection procedure), equations 25 and 26 can be modified as:

Top Down (F2 → F1) : $\frac{d(Z_{J\,i})}{dt} = g(y_J)[p_i - Z_{J\,i}] = d(1-d)\left\{\frac{u_i}{1-d} - Z_{J\,i}\right\}$ (28)

Bottom Up (F1 → F2) : $\frac{d(Z_{iJ})}{dt} = g(y_J)[p_i - Z_{i\,J}] = d(1-d)\left\{\frac{u_i}{1-d} - Z_{i\,J}\right\}$ (29)

for $j \neq J$: $\frac{d(Z_{i\,j})}{dt} = 0$ and $\frac{d(Z_{j\,i})}{dt} = 0$ (30)

$$\text{for } j \neq J: \quad \frac{d(Z_{ij})}{dt} = 0 \text{ and } \frac{d(Z_{ji})}{dt} = 0 \tag{30}$$

The *orienting subsystem* consisting of the *reset mechanism* resets F_2 whenever the input pattern is active and

$$\frac{\rho}{e + |\mathbf{r}|} > 1 \tag{31}$$

where,

$$r_i = \frac{u_i + c\,p_i}{e + |\mathbf{w}| + |c\,\mathbf{p}|} \tag{32}$$

r : the attentional vigilance parameter

The network initialization involves setting all the network parameters at typical values indicated in the section on network equations. The top-down connection weights are all initially set equal to 0. The bottom-up connection weights are initialized at:

$$z_{ij} = \frac{1}{2(1-d)\sqrt{M}} \tag{33}$$

ART-II Learning algorithm:

The ART-II learning algorithm used in all the simulations that were carried out is presented in the form of a simulation flow diagram shown in figure 24. The flow diagram is self explanatory and represents a complete cycle of events within the network during a single iteration. The major features of the algorithm are the *STM stabilization loop* and the *search loop*. The STM stabilization loop ensures that all STM activities at F_1 stabilize (no longer changes) before a bottom-up input pattern can be transferred from F_1 to F_2 through the LTM. The search loop comprises of the bottom-up and top-down processing mechanisms and the vigilance test that determines the appropriate recognition category corresponding to an input pattern.

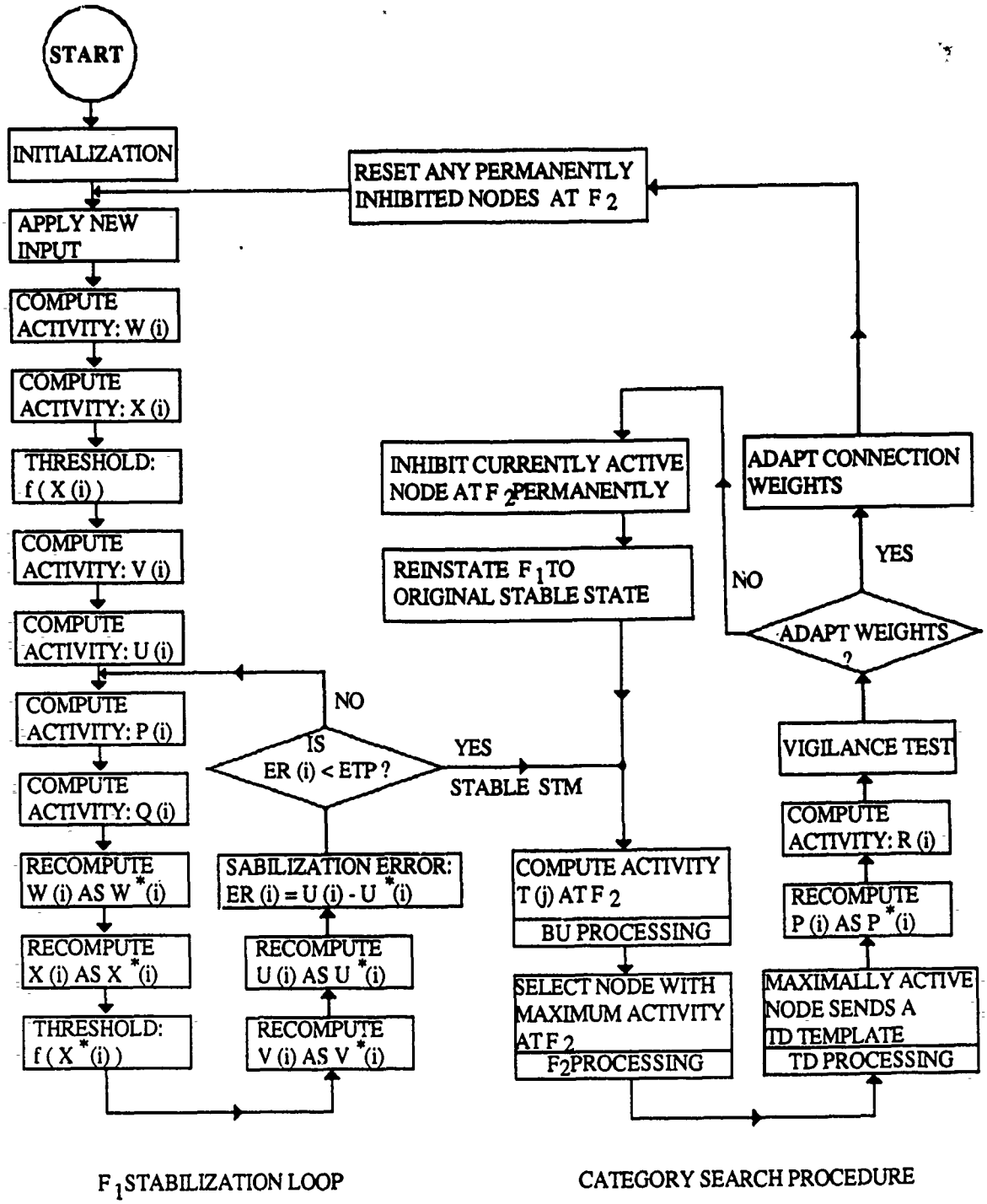


Figure 24 Flow diagram of the ART-II algorithm

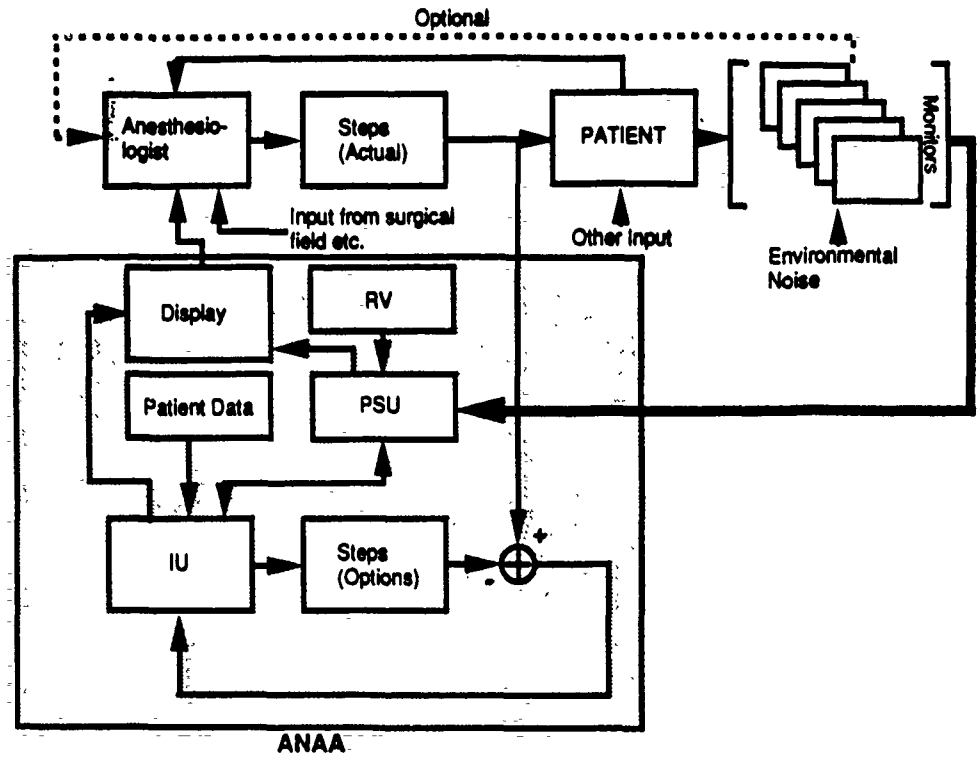


Figure: ANAA's Architecture

REFERENCES

[1] L.E. Widman, "Expert System Reasoning About Dynamic Systems by Semi-quantitative simulation", Comp. Meth. Prog. Biomed. Vol. 29(2), pp. 95-113: 1989.

[2] G.A. Miller, "The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information", Psych. Review Vol. 63(2), pp. 81-97.

[3] G. Sperling, B.A. Dosher, in: "Handbook of Perception and Human Performance", eds. K.R. Boff, L. Kaufman and J.P. Thomas, Vol., Chap. 2, pp. 4-17 (John Wiley & Sons,New York, 1986).

Time Optimal Robot Navigation via the Slack Set Method

STEVEN C. ZAHARAKIS AND ALLON GUEZ

Abstract—The minimum-time obstacle avoidance trajectory for a robot has been a long standing problem of considerable interest. The *slack set* technique, an algorithm for determining a minimum-time obstacle avoidance trajectory for a robot in a known environment, is presented. For time optimal trajectories with constrained acceleration and velocity, the shortest time of motion may be different for each joint or axis of the system. Thus some delay of a joint other than the slowest will not necessarily affect the time of motion for the entire system. This natural redundancy for obstacle avoidance in order to simplify the trajectory search algorithm by at least one order of magnitude (one DOF less) is exploited. By neglecting the presence of all obstacles and assigning to each actuator maximum control (bang-bang), a lower-bound estimate of the time needed to complete a task (Ttask) is calculated. The A^* heuristic search is used to search what we call the slack set; a subset of the state space that contains only those states that are members of a trajectory with a task time equal to T task. If no trajectory is found during the initial search, the subset of the state space being examined is sequentially increased until a valid trajectory is found. The slack set technique is guaranteed to find a feasible monotonic trajectory if such a trajectory exists in the slack set. Since, in general, the minimum-time obstacle avoidance trajectory is not unique, secondary constraints such as minimum distance, minimum distance in the state space, and others can also be satisfied. The method is demonstrated via a planar mobile robot.

I. Introduction

ROBOTS have been applied in manufacturing for many years. Today robotics has found applications in such diverse fields as medicine and space exploration. The most important capability of a robot is its ability to move in and modify its environment. After almost three decades of worldwide research in the area of robot motion control, the state of art offers rather simplistic control and motion planning methods which are employed in the most advanced systems of industrial robots. The current state of knowledge indicates that the problem of time optimal obstacle avoidance robot trajectory derivation is still unsolved. Dynamic constraints imposed by the robot's mechanical structure, and actuators have yet to be incorporated in the process of trajectory planning. The importance of inclusion of robot dynamics in the process of path planning has been recognized by many [1], [10], [28]. No practical technique has been developed due to the complexity of the problem, the severity of the computational requirements and the current approach to the problem. Motion planning and execution is heuristically decoupled mainly into two independently solved subproblems: path planning and path following. This artificial separation of the control and planning tasks degrades the overall performance of the system. It is expected with the continuous enhancement of speed and power per unit cost of processors, the advent of new computers, architectures and computational algorithms, that a global simultaneous solution of some of the mentioned subproblems may become feasible.

The main objective of this work is to develop a unified strategy of path planning and control. Our approach to time-optimal obstacle avoidance trajectory planning unifies several problems which are usually considered separately: path planning, minimum time control, real-time controller structure, and obstacle avoidance. We believe that such unification gives a number of theoretical benefits and helps in solving problems proven to be difficult when approached separately. The objective is to develop a real-time robot motion controller that will *simultaneously* generate, based on static and dynamic constraints (including parameters of the system, location of obstacles and task constraints), the time optimal obstacle avoidance control and trajectory. Our approach is based on the observation that optimal path planning cannot be achieved without accounting for robot dynamics, especially in an obstacle strewn environment [1], [7]. This dynamic path planning and execution is feasible by exploiting modern control methods such as global linearization and decoupling via state feedback and the Maximum Principle [5], [7]. This approach allows us to treat the position constraints as a part of the overall dynamic constraints on the state variables of the system. Other benefits of this approach are the ability to derive time optimal obstacle avoidance trajectories that also display secondary optimizations by exploiting the fact that the time optimal trajectory for robot manipulators satisfying inequality constraints on the jerks, accelerations and velocities of the joints is not unique [29]. This "redundancy" in the selection of a time optimal trajectory creates a "slack set" which will be used in selecting the trajectory that is optimal with respect to some prespecified criterion. The following paragraphs describe results related to our work.

The area of path planning among obstacles has been studied mainly during the last ten years. Lozano-Perez

Manuscript received April 1, 1989, revised October 25, 1989.

S. C. Zaharakis is with the Mathematics and Computer Science Department, Allentown College of Saint Francis de Sales, Allentown, PA 19104.

A. Guez is with Drexel University, Philadelphia, PA.

IEEE Log Number 9037600.

0018-9472/90/1100-1396$01.00 ©1990 IEEE

and Wesley [20] proposed a method for finding the minimum distance path through a polygonal obstacle infested workspace. A visibility graph (VGraph) consisting of vertices of the polygonal obstacles, the initial position, and the goal position are used [18]. Two vertices are connected with edges when the connecting edge does not intersect any obstacle. A cost is assigned to each edge according to its length. A searching algorithm is then employed to determine the set of these edges, which, when combined, lead us from the initial position to the goal position with the minimal total cost (minimum distance). This technique will be used as a comparison to the slack set technique introduced in this paper.

Pieper and Widdoes [27] used planes, cylinders, and spheres to represent obstacles. This approach has the advantage of eliminating the orientation problem, but introduced the problem of path elimination due to obstacle modification. The enlargement of the obstacle in some cases eliminates feasible paths. Additionally, the intersection functions are often nonlinear and involve square roots or transcendental functions. Udupa [27], Lozano-Perez and Wesley [19] and Brooks [4] adopted the polyhedra as the models that result in linear intersection functions, but the orientation problem must be handled with care. Udupa discretized the space into sectroids and passes that were labeled as free if not occupied by obstacles and objects. Lists of free passes are joined together to form a collision-free path. To allow for arbitrary orientation, obstacles are enlarged. This in turn, reduces the number and size of the free passes. Lozano-Perez described linked polyhedra using swept volumes. Free space is then represented as overlapping generalized cones. In these methods, some determine the free space inside which the point robot may move freely without collision, while others determine the forbidden region so that a collision-free path may be traced along the boundaries of the region. Luh [21] modifies the environment by inclusion of pseudo-obstacles that are generated by real obstacles' edges and faces. This process allows the robot itself to be represented by a point specifying the robot's tip location in space.

In order to derive the control required to traverse the derived path, a trajectory must be found for the path. This is generally accomplished by searching the state space for the velocity profiles that will allow for the traversability of the given path [2], [26], [27]. Using this technique, one can derive the velocity profile that allows for the minimum time traversal of a path. Note that traversing a minimum distance path in minimum time does not, in general, produce a minimum time trajectory. This subject will be discussed in detail later in this paper. None of these works, however, accounts for manipulator dynamics in the planning process. As a result, the traversability of such paths is questionable due to the dynamic and mechanical constraints of the manipulator.

Paul *et al.* [23] present a linear programming algorithm for finding the near minimum time path of the manipulator end effector. It deals with path planning by designing a set of time intervals for the transition among a sequence of preassigned points in the Cartesian space. The motion between each pair of points is assumed along a straight lien segment with constant velocity. The total traveling time is minimized while observing velocity and acceleration inequality constraints. The manipulator dynamics are not included since the path is assigned a priori. This technique cannot, in general, produce the time optimal traversal of a given path due to the fact that it deals with local and not global optimization. Later this method was extended by Lin *et al.* to fitting of cubic polynomials with velocity, acceleration and jerk constraints in planning a minimum time trajectory for a given sequence of points. Lynch [17] derives a sequential mode (one axis at a time) minimum time for two axis manipulator. In a sequential mode, coupling between the various axes is significantly reduced; the execution time, however, is much larger than in the one obtained in simultaneous motion of all axes.

The following techniques incorporate some dynamics of the robot during their planning. The idea of *artificial potentials* [13], [15], [22] is to allow to move about space while under the influence of the forces of attraction and repulsion. The attractive force would stem from the goal point, while all obstacles would exert repulsive forces on the robot. All forces would be directly proportional to the velocity by which the source is approached and inversely proportional to the distance from it, thus partial inclusion of robot dynamics is achieved. Loeff and Soni added to the end effector of a manipulator the velocity towards the destination and gave the joints repulsive velocities from the obstacles. Similarly, Khatib and Le Maitre [13] used an attractive force on the end effector attracting it to its desired position and orientation. Okutomi and Mori [22] took a more elaborate approach and used artificial potentials in the joint coordinate space. Repulsive forces were exerted from all forbidden regions in this space.

The minimum-time path for a robot has been a long-standing and unsolved problem of considerable interest. For problems of nontrivial dynamics, the derivation of a time-optimal solution cannot be guaranteed. This is true primarily due to the unavailability of an analytical solution and the limitations of today's computers which do not allow for the verification of a nontrivial "time-optimal obstacle avoidance trajectory." Though the optimal control theory of dynamic systems is well established, it is rarely used in practice due to the highly nonlinear and highly coupled form of the differential equations that govern the system. The need to incorporate heuristic knowledge about plausible solutions, along with the ability to make tradeoffs concerning the optimality of the solution, as well as the computational cost of the derivation, have been outlined in [7], [8].

One category of trajectory planning techniques uses a *bang-coast-bang* approach. One of the first attempts of time-optimal trajectory derivation was made by Kahn and Roth [12]. His technique assumed all trajectories to follow

a bang-bang trajectory with multiple switching times. Luh and Lin [20] chose to subdivide the problem into the derivation of time optimal trajectories connecting various points along a predefined path. Position, velocity and acceleration constraints of the weakest actuator are imposed on the whole system, thus severely degrading performance. Kornhauser and Brown [14] developed a technique based on state space tessellation and a graph search. In this technique they assume a bang-coast-bang solution with fixed switching points. Successors of a state are generated by using nine torque patterns. Through the use of a heuristic search technique the fastest trajectory is found.

The *time-scaling* technique is used in the derivation of a trajectory which will traverse a given path in minimum time [1]. The equations of motion are expressed in terms of the distance along the predetermined cartesian path. A finite distance-velocity state space is derived using the torques/forces specifications of the motors. A set of switching points is then found that moves the arm as close as possible to the limits of the state space. The optimality of this algorithm was proven by Bobrow in his Ph.D. paper. He found that one actuator is always saturated, and that the others adjust their torques. Other time-scaling implementations [11], [17] use polynomial splines for the initial path definition. Iterative modification of these paths is attempted until convergence of a local minima is achieved.

II. Problem Statement

Given the dynamic model of the N degree of freedom (NDOF) robot with its N DC Servo actuators driving joints:

$$\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= g(X,t) \\ \dot{X}_3 &= F_1 X_2 + F_2 X_3 + L^{-1} V \end{aligned} \tag{1}$$

where

$X_1 = (X_{11}, X_{12}, \cdots X_{1N})^T$
is the joint displacements vector

$X_2 = (X_{21}, X_{22}, \cdots X_{2N})^T$ is the joint velocities vector

$X_3 = (X_{31}, X_{32}, \cdots X_{3N})^T$
is the rotor currents of the actuators vector

$V = (V_1, V_2, \cdots V_N)^T$
is the vector of applied input voltages.

Given the initial configuration

$$X(t_0) = (X_1(t_0), X_2(t_0), X_3(t_0))^T \tag{2}$$

the desirable final configuration

$$X(t_f) = (X_1(t_f), X_2(t_f), X_3(t_f))^T \tag{3}$$

the inequality constraints

$$\begin{aligned} |X_{2i}| &\leq \omega_{m_i} \\ \left|\frac{dx_{2i}}{dt}\right| &\leq \alpha_{m_i} \\ \left|\frac{d^2x_{2i}}{dt}\right| &\leq \Delta_i \end{aligned} \tag{4}$$

where $\omega_{m_i}, \alpha_{m_i}, \Delta_i$ are the maximum allowed velocity, acceleration and jerk respectively for the ith joint.

The sets of forbidden regions are assumed to be given in the joint space (X_i) as time varying intervals which may or may not be connected. These are obtained from the kinematic translation of the obstacle's configuration in the workspace, thus Given also the sets of forbidden regions (obstacle) for each joint $S_i(t)$

where $S_i(t) \in R$, for all $t \in (t_o, t_f)$, $i = 1,2,3,\cdots,N$

and the joints hard limits

$$\Theta_{i\min} \leq X_{1i} \leq \Theta_{i\max}. \tag{5}$$

From among all the state trajectories that satisfy (1) through (5), (i.e., feasible trajectories), and minimize the travel time (6)

$$t_f - t_0 = \int_{t_0}^{t_f} 1\, dt \tag{6}$$

find the trajectory $X^*(t)$ that maximizes/minimizes the secondary functional

$$J(X(t), V(t)). \tag{7}$$

The problem stated previously is relevant for both industrial and mobile robots. In the case of point mass mobile robot models, however, coupling between the various DOFs occurs through the state inequality constraints and obstacles, rather than in the robot dynamics.

This paper will describe an algorithm for finding a time optimal or near time optimal obstacle avoidance trajectory for a robot of known dynamics, if such a monotonic trajectory exists. For reasons of simplicity, the algorithm will be illustrated by applying it to an autonomous vehicle with motion being constrained to the two-dimensional (2-D) XY plane, as follows. However, the algorithm can be generalized to a robot with an arbitrary number of degrees of freedom.

III. Slack Set Method

The slack set method is outlined in this section. The method introduces a new concept for severely reducing the search space, thus minimizing the computational cost of deriving such a trajectory. Details of the technique follow, as do explanations of relevant terms. The algorithm attempts to derive a time optimal obstacle avoidance trajectory for a robot of known dynamic capability. Through-out this paper it is assumed that the robot and the robot's environment are completely known. As a result, the maximal distance, velocity and acceleration of

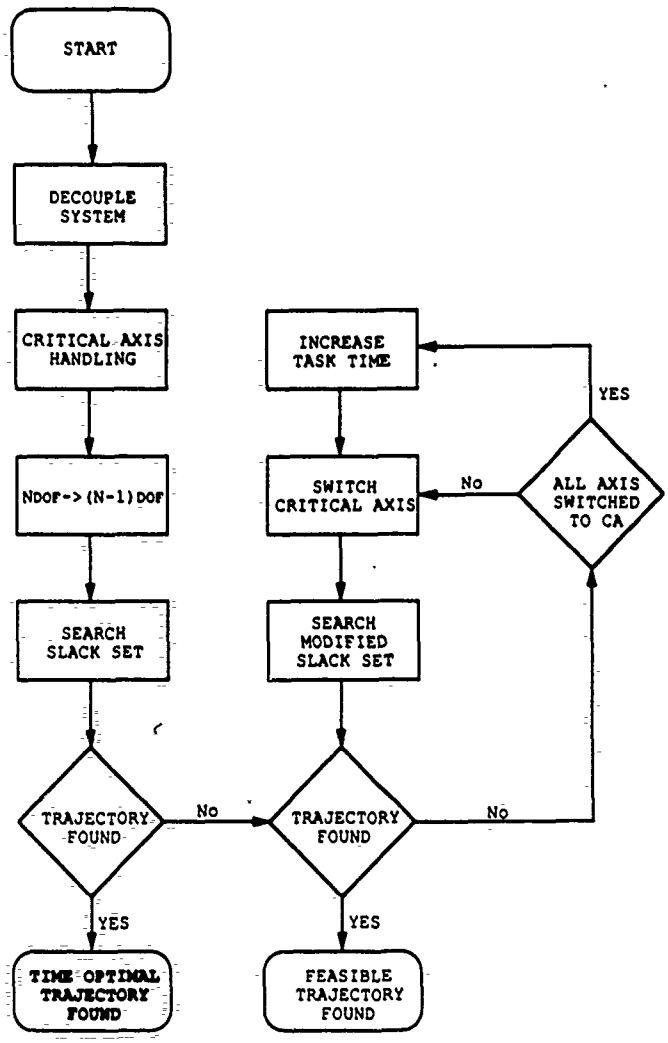


Fig. 1. Flowchart of the slack set technique.

the vehicle is known. Position, size and shape of all obstacles are also considered known.

For reasons of simplicity the slack set method will be applied to an autonomous vehicle with motion being constrained to the XY plane, with bounded distance, velocity and acceleration. However, the algorithm can be generalized to a robot with an arbitrary number of degrees of freedom. All obstacles are rectangular with random size and position in the workspace. When overlapping, they produce obstacles of random size and shape. All obstacles are considered static. The slack set technique is outlined by the flowchart of Fig. 1.

Decoupling the System

Let the dynamic model of a planar point mass robot in a two-dimensional plane be

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{V}_x \\ \dot{V}_y \end{pmatrix} = f(X, Y, V_x, V_y, T_x, T_y) \tag{8}$$

where X,Y denote the robot's position, V_x,V_y denote robot velocity, and T_x,T_y denote torque along the X and Y axis correspondingly.

In general, (8) is a coupled nonlinear four-dimensional (2-D) differential equation describing the coupling and nonlinearities of the robot. It is well known that when the robot dynamics are known, it is possible to employ the Feedback Linearizing and Decoupling Transformation to achieve global decoupling of an N-DOF system into N linear subsystems without disregarding robot dynamics. [6] As a result, we can assume that the motion along the axes has independent dynamics, however cross axis coupling still exist through the presence of obstacles as will be described below.

In the case of our 2DOF system, let $V_{x\max}, V_{y\max}$, $\alpha_{x\max}, \alpha_{y\max}$ be the maximum velocity and acceleration along the X and Y axis respectively. We assume total knowledge of the static environment, position and size of all obstacles. The robot's state is described by $S(X,Y,V_x,V_y)$. Given an initial position (x_o, y_o) and a goal position (x_g, y_g), we are to derive a trajectory which would take the robot from rest at (x_o, y_o) to rest at (x_g, y_g) while avoiding all obstacles and do so in the least amount of time.

Applying FLDT, the dynamics (8) can be expressed as the equations for two double integral plants:

$$\begin{aligned} \dot{X} &= V_x \\ \dot{V}_x &= \alpha_x \\ \dot{Y} &= V_y \\ \dot{V}_y &= \alpha_y \end{aligned} \tag{9}$$

where the admissible set (3) is:

$$|\alpha_x| \leq \alpha_{x\max} \qquad |\alpha_y| \leq \alpha_{y\max} \tag{10}$$

the inequality constraints (4) are:

$$\begin{aligned} |X| &\leq X_{\max} \qquad |Y| \leq Y_{\max} \\ |V_x| &\leq V_{x\max} \qquad |V_y| \leq V_{y\max} \end{aligned} \tag{11}$$

The cost functional which we choose to minimize is the time needed to complete the task (task time), of taking the robot from its rest at its initial state S_o, to rest at its goal state S_g while avoiding all obstacles. It is assumed that the task is started at time $t_o = 0$ thus (2) becomes

$$G = t_f - t_o = t_f. \tag{12}$$

Static obstacles can be expressed as functions of position only

$$O_i(X,Y) = \{\text{All } (x,y) \text{ points within the } i\text{th obstacle region}\} \tag{13}$$

where $i = 1,2,\cdots d$, d being the number of obstacles in the robots workspace.

The goal state expresses the position and velocity which the robot is to achieve by the end of the task time (t_g). If the goal velocities are both equal to zero, then the goal

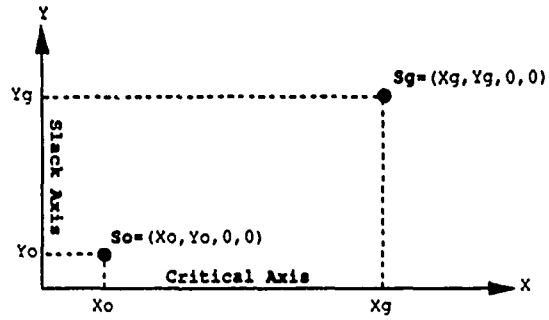


Fig. 2. Example task with no obstacle.

state is

$$S_g = (X_g, Y_g, 0, 0). \tag{14}$$

Using the previous definitions the problem can be stated as follows.

> Find the input $\alpha^*(t)$, $t \in [0, t_f]$, which globally minimizes the cost functional (12) while satisfying (10), and for which the resulting state trajectory $S^*(t)$ satisfies (9) and (11), avoiding the union of all obstacles (13) and reaching S_g at time t_g.

Before any computer algorithm can be implemented, we must first discretize time. For simplicity and without loss of generality, let the fixed sampling rate $\Delta T = 1$. After appropriate scaling, the equations of motion of the robot will be

$$\begin{aligned} X(k+1) &= X(k) + Vx(k) \\ Y(k+1) &= Y(k) + Vy(k) \\ V_x(k+1) &= V_x(k) + \alpha_x(k) \\ V_y(k+1) &= V_y(k) + \alpha_y(k) \end{aligned} \tag{15}$$

where $k = 0 \cdots k_g$ and k_g is such that $t_g = t_o + k_g \Delta T$.

The problem to be solved is therefore to find the sequence $(\alpha_x(k), \alpha_y(k))$, where $k = 0 \cdots k_f$, which takes the robot from rest at position (X_o, Y_o) at $k = 0$ and moves it to position (x_g, y_g) at $k = k_g$ while avoiding all obstacles for the minimum value of k_g, satisfying all state and input inequality constraints. Coupling still exists through the presence of obstacles.

Critical Axis Handling

Depicted in Fig. 2 is an example of a time optimal trajectory planning problem in the absence of obstacles. Given that $V_{x\max} = V_{y\max}$ and $\alpha_{x\max} = \alpha_{y\max} = \alpha_{\max}$, we wish to reach state *Sg* from state *So* in minimum time. By neglecting the presence of all obstacles and assigning to each actuator maximum control (bang-bang) a lower bound estimate of the time needed to complete a task (T task) can be calculated. When the maximum principle is applied to the problem described in Fig. 2 the time optimal trajectories of Fig. 3 are obtained [6], [9].

Trajectories displaying a monotonically decreasing distance to the goal, along at least one of the axis are denoted as monotonic. Let us assume here that, $T_{x\,\text{task}} > T_{y\,\text{task}}$ (Fig. 3). The task is complete when subtasks along both axis are complete, thus $T_{\text{task}} = \max(T_{x\,\text{task}}, T_{y\,\text{task}})$. The axis with the maximal task time is called the critical actuator (CA) while all others are called slack actuators (SA) (Fig. 3). Inputs of the slack actuator (Y axis here) can be modified as long as at time $T_{x\,\text{task}}$ we have $y = yg$ and $Vy = 0$; all these trajectories have 1) $T_{\text{task}} = \max(T_{x\,\text{task}}, T_{y\,\text{task}})$; and 2) $T_{y\,\text{task}} \leq T_{x\,\text{task}}$.

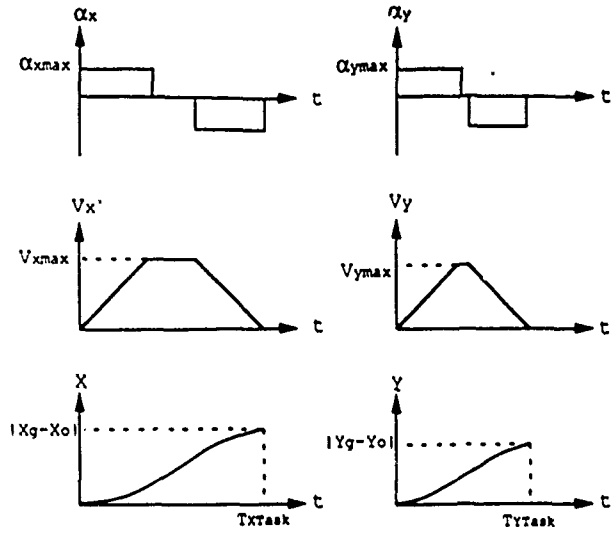


Fig. 3. Bang-bang profiles.

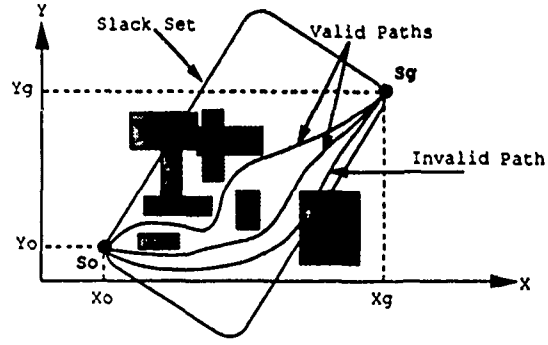


Fig. 4. Possible paths in the slack set.

We name the set containing all such trajectories the slack set. Note that the slack set contains a set of minimum-time trajectories of a specified task.

This multiplicity of trajectories motivates us for a closer examination of the slack set (Fig. 4). If valid trajectories exist in the presence of obstacles (traversable paths), then all these trajectories will be time optimal obstacle avoidance trajectories.

By neglecting the presence of all obstacles and assigning the maximum control torque to each actuator we can determine which actuator's task time is greatest (T_{task}). This actuator, called the critical actuator (CA), exerts a time constraint on the completion of the task. The CA retains its bang-bang profile while a search will determine the profiles of the other actuators. We assume here, with no loss of generality, that $T_{\text{task}} = T_x > T_y$, thus we will have a fixed bang-bang profile along the X axis.

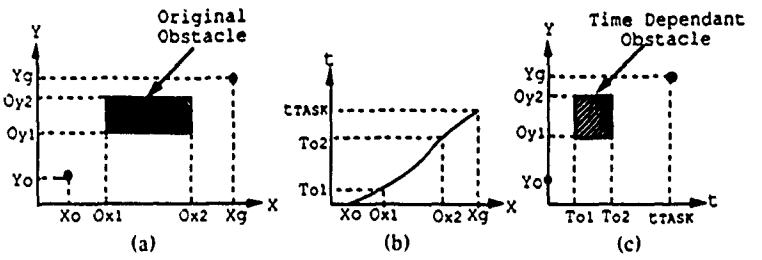


Fig. 5. Transforming the problem. (a) Original task. (b) Control actuator distance profile. (c) Transformed task.

Transformation from N-DOF to (N-1)-DOF

We can take advantage of the a priori knowledge of the CA profile by transforming the problem from the XYV_xV_y state space to the slack state space YV_y. This is done by using the Critical Actuator distance profile in conjunction with the knowledge of all obstacle positions.

Let $Ox1, Ox2$ be the projections of an obstacle's boundary along the X axis and $Oy1, Oy2$ be the projections of the obstacle's boundary along the Y axis (Fig. 5). Using the Critical Actuator's distance profile and the distances $Ox1, Ox2$ one can determine during what times t_{01} and t_{02} the robot will be in the projection of the obstacle along the X axis (Fig. 5). If (xn, yn) defines a position of the robot in the XY plane then:

$$\{\text{Robot in Obstacle}\} \Leftrightarrow \{x \in [Ox1 \cdots Ox2] \quad \text{and} \quad \in [Oy1 \cdots OY2]\}.$$

The obstacle is avoided if during this time period $[t_{01} \cdots t_{02}]$ the robot's y coordinate of position is not in the region $[Oy1 \cdots Oy2]$ (Fig. 5). By transforming all obstacles from the XYV_xV_y to YVy slack state space we severely reduce the state space size.

Note that our definition of an obstacle is independent of velocity. As a result, all states with a positional vector located in an obstacle are illegal and as such can not belong to the Slack Set. Thus, the size of the slack set is inversely proportional to the area occupied by obstacles. Secondly, we can model complex obstacles such as mud, by including velocity in our definition of an obstacle. For example we may allow the robot to move through a certain position if its velocity exceeds some defined "escape velocity." Work has already begun to include moving obstacles.

Searching the Slack Set

In order to derive the profile along the slack axis (Y) we search in a subset of the YV_y state space called the Slack Set. A state (y, v) belongs in the Slack Set if the following two conditions are satisfied: 1) $|y - Yg| \leq V^*_{y\max}(T_{\text{task}} - t)$—goal position is reachable, and 2) $|v| \leq \alpha^*_{y\max}(T_{\text{task}} - t)$—goal velocity is attainable. The slack set is searched using an A^* heuristic search [24]. As mentioned previously, any trajectory found in the slack set will be a time-optimal trajectory. For this reason we need not choose to use heuristics that focus on minimizing time. Dependent upon the heuristic $h(n)$ being used in the evaluation of each state, the minimum-time trajectory may further optimize criteria such as minimum distance, smoothness of the trajectory, safety, etc. For example:

1) $f(n) = |Dx(n)|$ min. distance

2) $f(n) = \sqrt{[k_1 Dx^2] + [k_2 Dv^2] + [k_3(D\alpha^2)]}$

. min. dist. in State Space.

We can guarantee that any trajectory found in the Slack Set will be a minimum-time trajectory because we have restricted the task time to Tx.

Unsuccessful search of the Slack Set — Modified Slack Set

Obviously, the task time of the time optimal trajectory in the presence of obstacles may exceed the task time of the time optimal trajectory in the absence of obstacles. If in the presence of obstacles all trajectories of the Slack Set are blocked, no trajectory will be found during the search of the Slack Set.

In the case where a time-optimal trajectory is not found by searching the slack set, the task time is increased and each actuator is sequentially defined to be the temporary CA. The task time may be arbitrarily increased or it may be increased through constraints imposed on the CA, e.g., through a restriction of the maximum allowable acceleration, or velocity. We define the modified slack set as the Slack Set which is created through an increase of the task time. The task time of any trajectory belonging to a modified slack set is larger than the task time of any trajectory of the slack set.

Consecutive modified slack sets are generated and searched until a valid trajectory is found. Solutions found using the modified slack set are not guaranteed to be minimum-time trajectories. The maximum error is bound though, and is equal to the increase of the original task time. A sufficient condition for the convergence of the slack set technique is that there exists a monotonic solution. If no solution is found after the first time increment and the sequential assignment of all $N - 1$ axis, in turn, as critical axis, the time is incremented again and the entire process is repeated.

IV. Example

This paper presents examples that highlight the performance of the slack set technique. The algorithm attempts to derive a time optimal obstacle avoidance trajectory for

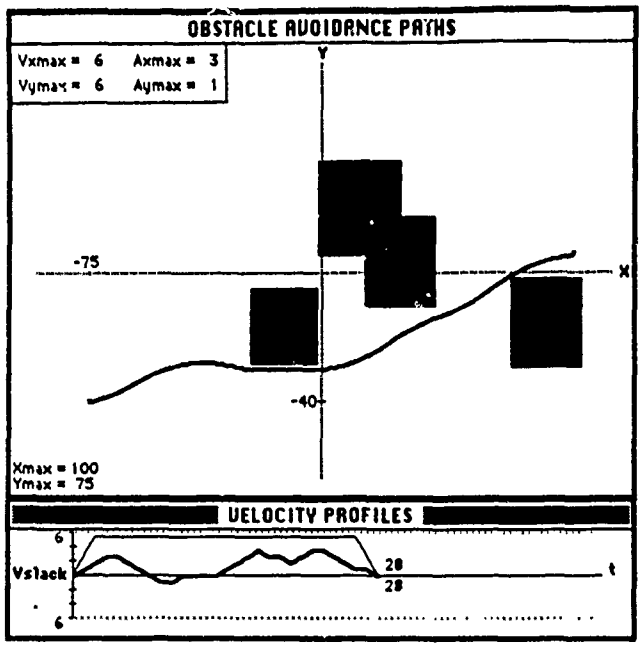


Fig. 6. Slack set Example 1.

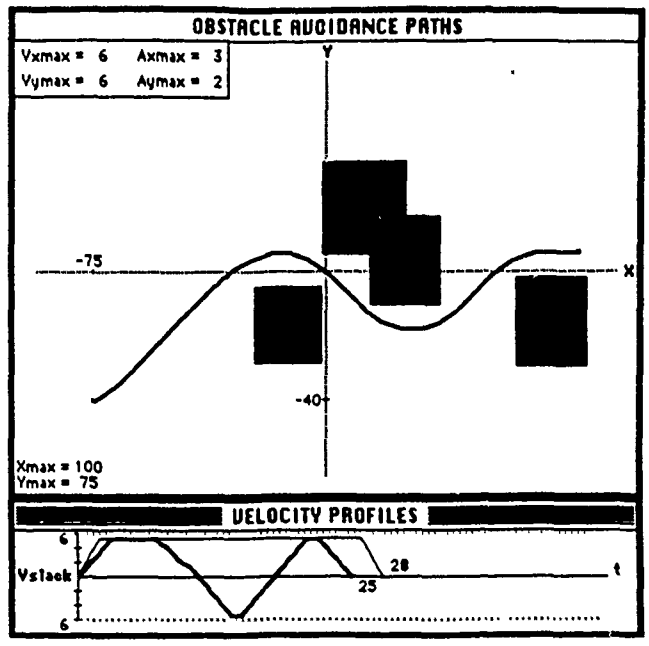


Fig. 7. Slack set Example 2.

a robot of known dynamic capability. Here it is applied to an autonomous vehicle with motion being constrained to the XY plane, with bounded distance, velocity and acceleration. However, the algorithm can be generalized to a robot with an arbitrary number of degrees of freedom. All obstacles are rectangular with random size and position in the workspace. When overlapping, they produce obstacles of random size and shape.

We are in search of the inputs needed to take the vehicle from rest at an initial state to rest at a goal state and do so in minimum time while avoiding all obstacles. The Slack Set method was implemented on a MAC II computer using Turbo Pascal.

A heuristic search is used in searching the slack set. Nodes expanded, are those nodes that are examined during the search. Nodes generated are those nodes which appear as successors of the nodes expanded. The size of the state space and the number of nodes examined in order to find a solution are indicative to the performance of the algorithm. We define the efficiency of the search as the ratio of the depth of the search (which in our case is equal to the task time), divided by the number of nodes expanded:

$$\text{Efficiency} = \left[\frac{\text{Depth of Search}}{\text{Number of Nodes Expanded}}\right] * 100\%. \tag{16}$$

In the first four examples, depicted in Figs. 6 through 9, we are looking for a time-optimal obstacle avoidance solution to the task of starting from rest at location $(-75, -40)$ and ending at rest at $(81, 6)$. Variations of the robots maximum velocity and/or acceleration result in drastic changes in the trajectory derived by the Slack Set technique. The ability of the technique to accommodate secondary constraints is illustrated in Figs. 10 and 11. A

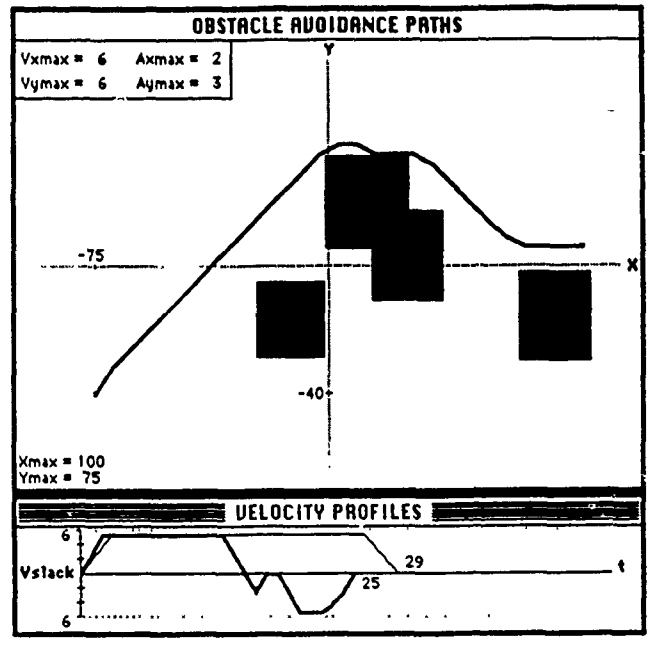


Fig. 8. Slack set Example 3.

trajectory obtained by searching the modified slack set, after unsuccessful searches of the Slack Set, is illustrated in Fig. 12. Trajectories derived through the Slack Set technique are compared with those derived using a path derived by the Vgraph technique in Figs. 13 and 14. Finally, Figs. 15 through 16 illustrate the performance of the slack set technique.

A. Effects of Robot Dynamics

Slack Set's dependence on robot dynamics is illustrated in Figs. 6–9. In the these examples we are looking for a time-optimal obstacle avoidance solution to the task of

TABLE I
SLACK SET TECHNIQUES DEPENDENCE ON ROBOT DYNAMICS

Example	$V_{x\max}$	$V_{y\max}$	$A_{x\max}$	$A_{y\max}$	#Nodes in Task	#Exp. Nodes	%Eff
1	6	6	3	1	3 277 908	194	14.1
2	6	6	3	2	6 555 816	225	12.2
3	6	6	2	3	6 555 816	314	9.1
4	6	4	3	3	6 555 816	164	17.1

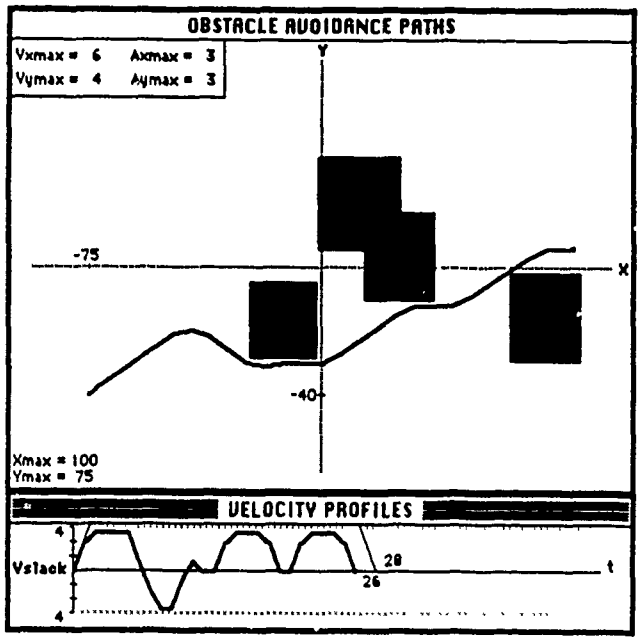


Fig. 9. Slack set Example 4.

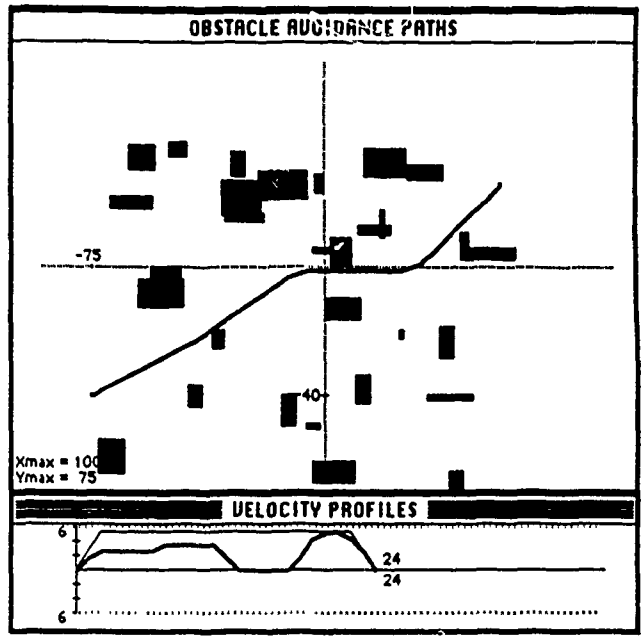


Fig. 11. Time optimal obstacle avoidance trajectory with secondary constraint on acceleration.

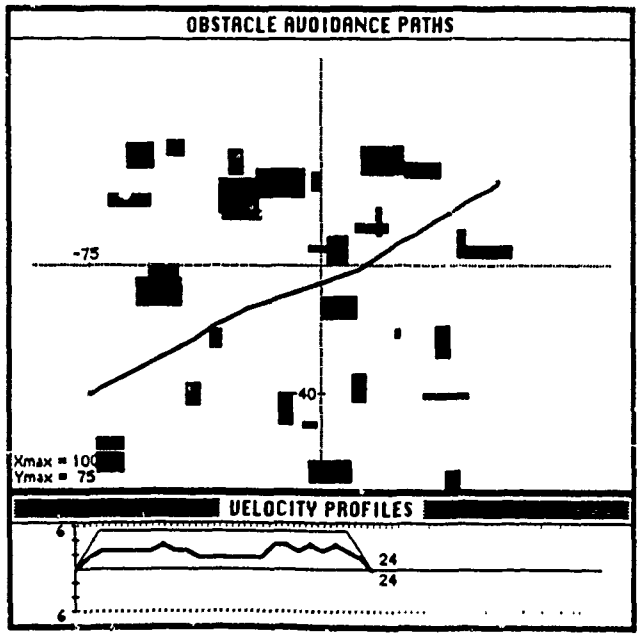
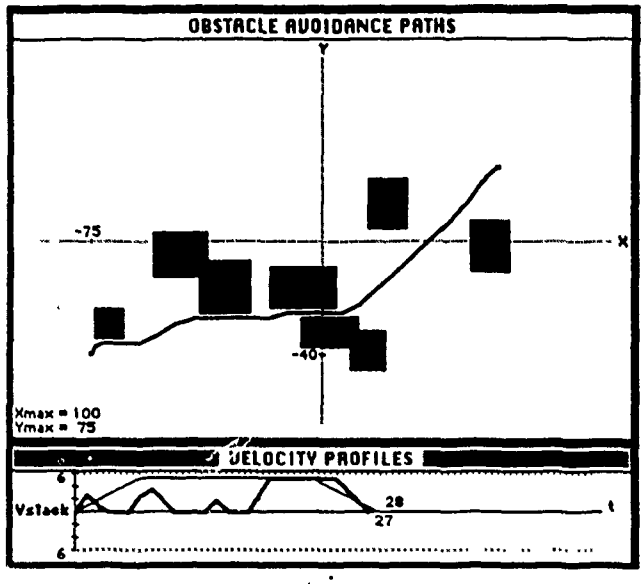


Fig 10 Time optimal obstacle avoidance trajectory with secondary constraint on distance.

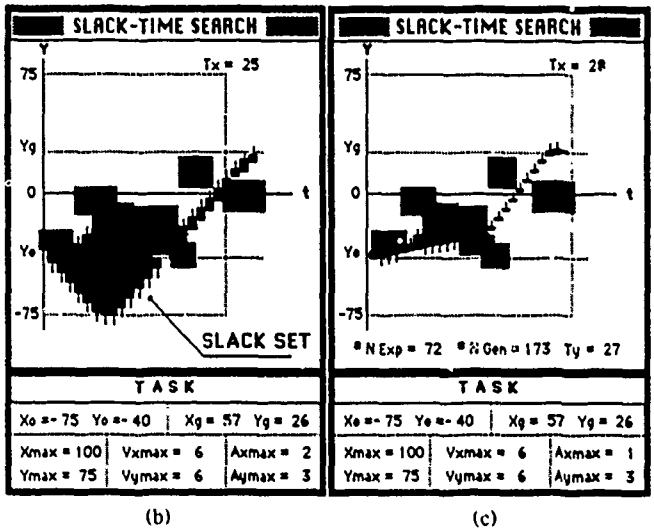
starting from rest at location $(-75, -40)$ and ending at rest at $(81, 6)$. Variations of the robots maximum velocity and/or acceleration result in drastic changes in the trajectory derived by the Slack Set technique. Results of variations in maximum acceleration are depicted in Examples 1, 2 and 3. while example 4 show results of variation in maximum velocity.

The obstacle avoidance paths illustrate the paths derived by the slack set technique. Velocity Profiles illustrate the profiles of the critical and slack axes. All obstacles are rectangular and are assumed fixed. In every example shown the critical axis in the X axis and the Slack Axis is the Y axis. The critical axis velocity profile is always trapezoidal as desired by the Slack Set technique. The slack axis velocity profile is drawn with the thicker line. No secondary constraints are imposed.

Dependence of the Slack Set technique on robot dynamics is illustrated in Table I where four variations of the robots dynamics resulted in four different trajectories. It is important to notice the difference which exists between trajectories which are derived with accommodation of robot dynamics and those which rely purely on the geometry of a particular task. One can see that although



(a)



(b) (c)

Fig. 12. Trajectory found by searching modified slack set. (a) Derived trajectory. (b) Unsuccessful search. (c) Successful search.

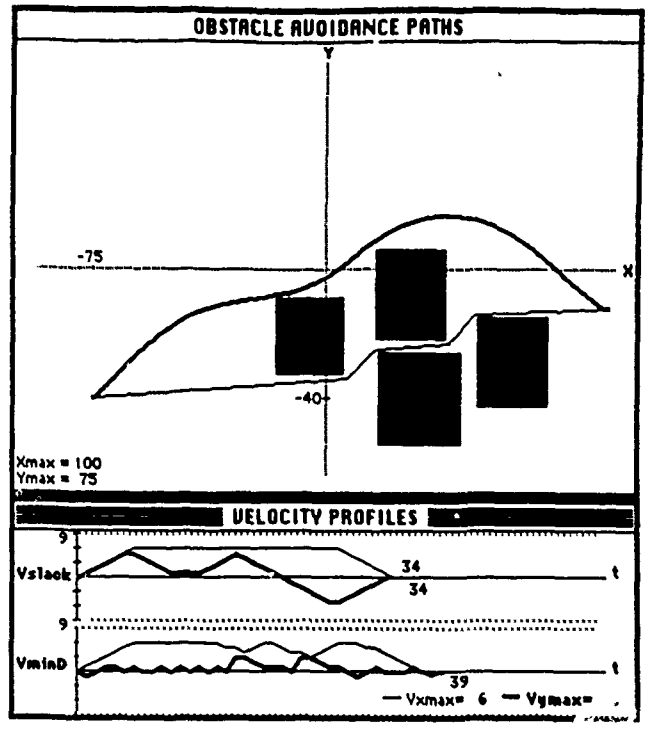


Fig. 13. Comparison between a slack set derived minimum time trajectory and a Vgraph derived minimum distance trajectory.

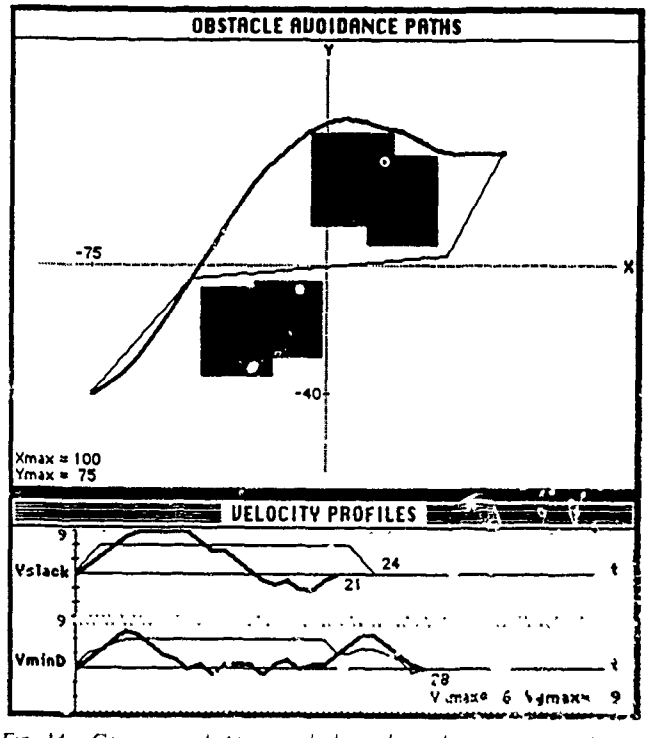


Fig. 14. Comparison between a slack set derived minimum time trajectory and a Vgraph derived minimum distance trajectory.

Examples 1 through 4 display the same geometric properties, each trajectory displays different velocity and distance profiles. On the other hand, all examples would be seen as the same problem by the Vgraph technique which relies purely on task geometry.

System dynamics affects the size of the state space. Using the Slack Set technique one severely reduces the number of nodes generated and examined. From Table I, we can see that the Slack Set technique examines less than 0.006% of the state space. The efficiency of the search is expressed by the ratio of the number of nodes expanded divided by the depth of the search. In these examples since we set the sampling rate to unity, the depth of the search is equal to the task time.

B. Multiobjective Optimal Navigation via Slack Set

In the following paragraphs we illustrate the capability of the slack set technique to derive time optimal trajectories which optimize secondary constraints. This is possible because the Slack Set usually contains more than one

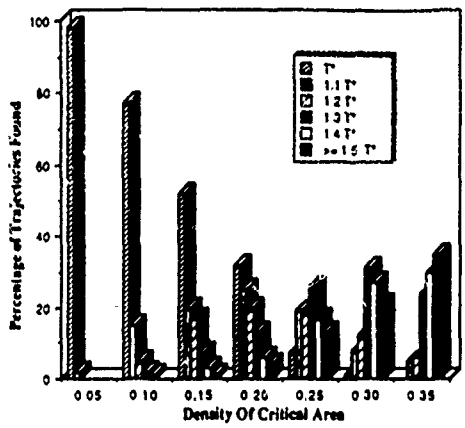


Fig. 15. Slack Set performance verses workspace density.

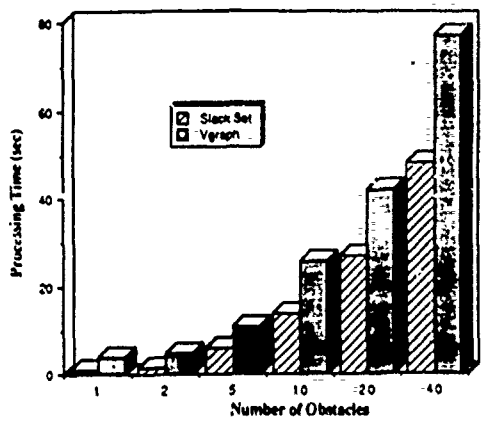


Fig. 16. Processing time comparison for slack set and *V*graph trajectories.

time optimal trajectory. As a result the slack set may be searched using heuristic functions that optimize constraints other than time constraints e.g., maximum safety by maximizing the minimum distance from the closest obstacle, smoothness via minimization of jerk, acceleration, velocity, etc.

Figs. 10 and 11 depict minimum-time trajectories of the same task with secondary constraints of minimum-distance and minimum-acceleration respectively. One can see how each trajectory differs significaintly from the other. The trajectory of Fig. 10 has a secondary constraint on distance. As a result the path derived in Fig. 10 is shorter than the path derived in Fig. 11. By maintaining a constant velocity as long as possible the acceleration is equal to zero as long as possible (Fig. 11).

C. Unsuccessful Slack Set Searches — Modified Slack Set Trajectories

Not every search of the slack set is successful. When such a situation presents itself, the slack set algorithm searches the modified slack set, which is generated by increasing the task time along the critical axis of motion by a fixed amount. This fixed increase of the task time along the critical axis provides us with an upper bound on the time difference between the derived task time and the actual minimum task time. Sequential switching of the critical axis among the axis of the robot guarantees the derivation of a feasible monotonic path, if such a path exists.

A task is depicted in Fig. 12. Initially $\Delta x_{max} = 3$. Examination of all states in the Slack Set does not produce a time optimal obstacle avoidance trajectory. The maximum acceleration along the Critical Axis is decremented ($\Delta x_{max} = 2$), thus producing a longer task time, and the Modified Slack Set is examined. Once again the slack set is searched without producing a trajectory (Fig. 12(c)). For reasons of simplicity, sequential switching of the critical axis will not be illustrated in this example.

Once again the maximum acceleration along the critical axis is reduced and the Modified Slack Set is searched. A trajectory is found (Fig. 12(c)) and we can guarantee that the maximum error is smaller than the difference between the initial task time and the derived task time. As a result, we can guarantee that the trajectory found in Fig. 12 has a maximum error of 12%. If a solution has not been found for a maximum acceleration of the critical axis equal to the minimum acceleration of the axis, we would restore the maximum acceleration to its initial value and decrement the maximum velocity of the critical axis.

D. Performance Comparison of the Slack Set and Vgraph Techniques

Fig. 13 depicts the results of a comparison between the trajectory found using the Slack Set algorithm and the *V*graph algorithm. In order to derive the minimum time trajectory which would follow the minimum distance path we did the following. First, we determined the minimum distance path using the *V*graph technique. Next, we define a *XYVxVy* state space with position on or close to the minimum distance path and valid velocity. The minimum time trajectories corresponding to the *V*graph paths were determined using A^*.

Fig. 13, emphasizes the distinction between the minimum-time trajectory and the minimum-distance trajectory, showing the minimum-time trajectory to have a smaller task time than the minimum-distance trajectory. The task times are shown to be 34 and 39 s for the slack set and the *V*graph techniques correspondingly. One can also see how the paths derived using the slack set are smooth curves compared to the paths obtained using *V*graph that are straight line segments with sharp turns, requiring complex velocity profiles. The velocity profiles required by *V*graph can be simplified by smoothing the derived path, but this would require additional computation. The velocity profiles show that slack set produces smoother results, always having one axis (the *X* axis in these examples) following a bang-bang profile. It is interesting to note here that it seems that *V*graph also tries to follow a bang-bang profile at least alone one axis.

Fig. 14 depicts another example were the trajectory derived using the slack set technique has a smaller task time than the trajectory derived using the *V*graph technique. This example also illustrates slack set's ability to utilize system dynamics to its fullest. For example the

slack set trajectory reaches and maintains the maximum velocity along the slack axis during the initial stage of the trajectory. This allows the velocity profile along the slack axis to terminate 21 units of time into the task. On the contrary, the Vgraph trajectory is in motion along both trajectory until the completion of the task. The X axis of the graph denotes the percentage of the critical area that is covered by obstacles. The various shades of each column denotes size of the determined task time with respect to the time optimal task time (T^*) in the absence of obstacles. As the area occluded by obstacles increases from 5% to 35% we see the percentage of trajectories derived with time optimal task time decreases. One very interesting result of this experiment was that the slack set technique never failed to find a feasible monotonic trajectory.

Fig. 16 was obtained by comparing the processing times needed in deriving a trajectory for a given task using the slack set technique and the Vgraph technique. Both techniques seem to require more computations as the number of obstacles is increased, though the search space of the slack set is inversely proportional to the size of the occluded area and the size of the Vgraph technique is proportional to the number of obstacles in the workspace.

Though implementations of each technique may not be optimal, it seems that the computational complexity of the Vgraph technique increases faster than the slack set technique.

VII. Conclusion

The slack set technique was introduced in this paper. This technique exploits the natural redundancy of a robot system, the fact that a time optimal trajectory for a given task is not in general unique, optimal control theory (Pontryagin's maximum principle) and linearization and decoupling techniques (FLDT), to reduce the size of the search space.

The slack set technique is able to derive a time-optimal obstacle avoidance trajectory for a robot when such a monotonic trajectory exists in the slack set. Otherwise a feasible trajectory will be found if such a monotonic trajectory exists.

Time optimal trajectories derived can also satisfy secondary constraints by searching the slack set using heuristic functions that optimize criterion other than time, e.g. maximum safety by maximizing the minimum distance from the closest obstacle, smoothness via minimization of jerk, acceleration and velocity, etc.

Trajectories derived using the slack set technique accommodate robot dynamics and as a result display smooth velocity and distance profiles. On the other hand, techniques that rely purely on the geometry of the task generate paths which in most cases have sharp turns (unless a smoothing algorithm is employed) and velocity profiles which are hard to follow.

For the slack set technique, the number of legal states in the state space decreases as the number of obstacles in the workspace increases. Thus, the size of the search space is inversely proportional to the number of obstacles. In Vgraph, the size of the state space is proportional to the number of obstacles in the workspace. Task times derived using the slack set technique also seem to have a linear relationship with respect to the density of the occluded area of the workspace. The percentage of trajectories derived with time optimal task times decreases as the area occluded by obstacles is increased.

Though implementations of each technique may not be optimal, it seems that the computational complexity of the Vgraph technique seems to increase faster than the slack set technique.

It was shown in the examples of Section IV, that trajectories comprised of the minimum distance path and velocity profiles required for the minimum-time traversal of the minimum distance path are not guaranteed to be time optimal trajectories. It is important to notice the difference which exists between trajectories which are derived with accommodation of robot dynamics and those which rely purely on the geometry of a particular task. One can see that although Examples 1 through 4 display the same geometric properties, each trajectory displays different velocity and distance profiles. On the other hand all four examples would be seen as the same problem by the Vgraph technique which relies purely on task geometry.

Not every search of the slack set is successful. When such a situation presents itself the slack set algorithm searches the modified slack set, which is generated by increasing the task time along the critical axis of motion by a fixed amount. This fixed increase of the task time provides us with an upper bound on the time difference between the derived task time and the actual minimum task time.

All trajectories derived via the slack set technique are monotonic. Degradation of the task time performance of the derived trajectory will occur when the critical axis velocity profile is forced to have a very small value.

Obstacles moving along a known trajectory can easily be included by the slack set method. This may be accomplished using swept volumes or through the inclusion of time in the transformation of the N-DOF problem to the (N-1) DOF problem. Accommodation of moving obstacles has begun and will be presented in future work.

The slack set technique was implemented using Turbo Pascal on a Mac II. time performance of the technique seldomly exceeded a few minutes, which includes some quite intense graphics. The size of the state space was in the range of 10^7 states, the tree structure being searched was in the range of 10^{11} nodes.

References

[1] J. E. Bobrow, S. Dubowsky, and J. S. Gibson "On the optimal control of robotic manipulators with actuator constraints," in *Proc. Amer. Contr. Conf.*, San Francisco, CA, June 1983, pp. 782–787.

[2] ____, "Time-optimal control of a robotic manipulator along specified paths," *The Int. J. Robotics Res.*, vol. 4, no. 3, pp. 3–17, Fall 1985.
[3] M. Brady *et al.*, *Robot Motion Planning and Control*. Cambridge, MA: MIT Press, 1982.
[4] R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst. Man Cybern*, vol-SMC-13, no. 3, Mar./Apr. 1983.
[5] E. Freund, "Fast nonlinear control with arbitrary pole placement for industrial robots and manipulators," *The Int. J. Robotics Res.*, vol. 1, no. 1, pp. 65–78, Spring, 1982.
[6] A. Guez, "Optimal control of robotic manipulators," Ph.D. Thesis, University of Florida, Gainesville, 1983.
[7] ____, "Heuristically enhanced optimal control," *Int. Proc. 25 IEEE Conf Decision Contr.*, Athens, Greece, Dec. 10–12, 1986, pp. 633–637.
[8] A. Guez and A. Meystel, "Time optimal, path planning and hierarchical control, via heuristically enhanced dynamic programming, a preliminary analysis," IEEE Workshop on Intelligent Control, Troy, NY, Nov. 1985.
[9] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," in *Proc. Amer. Contr. Conf.*, pp. 752–756, 1983.
[10] R. Horowitz and C. Tamizuka, "An adaptive control scheme for mechanical manipulators compensation of nonlinearity and decoupling control," ASME paper No. 80-Wa/DCS-6, 1980..
[11] K. Kant and S. X. Zucker, "Toward efficient trajectory planning: The path velocity decomposition," *The Int. J. Robotics Res.*, vol. 5, no. 3, pp. 72–89, Fall 1986.
[12] M. E. Kahn and B. Roth, "The near-minimum-time control of open-loop articulated kinematic chains," *J. Dyn. Sys. Meas. Contr.*, vol. 93, pp. 164–172, 1971.
[13] Khatib and F. LeMaitre, "Dynamic control manipulator operating in a complex environment," in *Proc. 3rd Symp. Theory and Practice Robots and Manipulators*, 1980.
[14] A. L. Kornhauser and M. L. Brown, "Approximate optimum paths for robot manipulators using state-space networks," in *Proc. IEEE Int. Conf. Robotics Automation*, pp. 750, Mar. 25–28, St. Louis, 1985.
[15] B. H. Krogh, "A generalized potential field approach to obstacle avoidance control," in *Robotics Research: The Next Five Years and Beyond*, Bethlehem, PA, Aug. 14–16, 1984.
[16] C. S. Lin and P. R. Chang, "Approximate optimum paths of robot manipulators under realistic physical constraints," in *Proc. IEEE Int. Conf. Robotics Automat.*, pp. 1066–1073, 1983.
[17] P. M. Lynch, "Minimum time, sequential axis operation of a cylindrical, two axis manipulator," in *Proc. 1981 Joint Automatic Control Conf.*, Charlottesville, VA, 1981, pp. WP-2A.
[18] T Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 10, pp. 681–689, Oct. 1981.
[19] T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979.
[20] J. Y. S. Luh and C. S. Lin, "Optimal path planning for mechanical manipulators," *ASME J. Dyn. Sys. Meas. Contr.*, vol. 12, pp. 142–151, 1981.
[21] J. Y. S. Luh and C. Campbell, Jr., "Minimum distance collision-free path planning for industrial robots with prismatic joints, *IELE Conf. Automatic Control*, vol. 29, no. 8, 1984, pp. 675–680.
[22] M. Okutomi and M. Mori, "Decision of robot's movement by means of potential field," *J. Robotics Soc. Jap.*, vol. 1, no. 3, pp. 226–232, 1983.
[23] R. Paul *et al.*, "Advanced industrial robot control systems," Purdue University, W. Lafayette, IN. Report No. NSF/RA-780167, 1978.
[24] J. Pearl, *Heuristics*. Reading, MA: Addison-Wesley, 1984.
[25] K. G. Shin and N. D. McKay, "An efficient robot arm control under geometric path constraints," in *Proc. 22nd IEEE Conf. Decision and Contr.*, pp. 1449–1457, 1983.
[26] ____, "Minimum-time control of robot arm control under geometric path constraints," in *Proc. 22nd IEEE Trans. Automatic Contr.*, vol. AC-30, 1985, pp. 531–541.
[27] S. M. Udupa, "Collision detection and avoidance in computer controlled manipulators," in *Proc. 5th Int. Joint Conf. Artificial Intell.*, 1977, pp. 737–748.
[28] S. C. Zaharakis and A. Guez, "Time optimal navigation via slack time sets," *IEEE Intern. Conf. Robotics Automat.*, pp. 650–651, Apr. 24–29, 1988.
[29] S. C. Zaharakis, "Time-optimal obstacle avoidance robot trajectory planning via the slack set method," M.S. thesis, Drexel Univ., Philadelphia, PA, 1988.

Steven Constantine Zaharakis was born on June 12, 1961, in Sydney, Australia. He received a B.S. in electrical engineering and a minor in international relations in 1984, and the B.S. degree in computer science from Lehigh University, Bethlehem, PA, in 1985, and the M.S. degree in electrical engineering from Drexel University, Philadelphia, PA, in 1988. He is currently working towards the Ph.D. degree in computer science at Lehigh University.

He is the Director for Robotics and Automation at Concept Engineering and an adjunct professor in the Mathematics and Computer Science Department at Allentown College of Saint Francis de Sales. His interests lie in the areas of automation, optimization and machine learning.

Allon Guez received the B.Sc.E.E. from the Technion—Israel Institute of Technology Haifa, in 1978, the M.S. and Ph.D. degrees in electrical engineering from University of Florida, Gainesville, FL, in 1980 and 1983, respectively.

He is on the faculty of Drexel University, Philadelphia, PA. His research interests include: nonlinear systems, robotics, machine intelligence and neuroengineering.

[5]

Design of Exploratory Schedules in Learning and Adaptive Robot Control

Allon Guez
John Selinsky

Drexel University, Department of Electrical and Computer Engineering
32nd and Chestnut St., Philadelphia, PA 19104

Abstract

In this article, we explore the relationship of learning and adaptation in robot control. Learning, in this context, is the process of identifying the robot dynamics and its interaction with the environment for the purpose of improved tracking over an infinite horizon. Whereas, adaptation is the process of adjusting the controller to comply with the stabilization (regulation and tracking) needs of the closed loop system for the present finite horizon problem. We thus demonstrate that learning conflicts with adaptation in its tendency to increase the present tracking error, due to the minimization of different criteria (Dual control principle).

Exploratory Schedules (ES) are reference trajectories which are specifically designed to provide efficient closed loop learning. We show how the design of ES is an essential aspect of learning which has been neglected. We relate ES design to the issue of input richness (or persistent excitation). Our ES represents a weaker criteria than persistent excitation

The robot model parameters are viewed as state variables. They are used to form the augmented state space in which asymptotic stability of the origin implies both asymptotic learning and adaptation A theorem regarding constructive sufficient conditions for asymptotically stable closed loop learning is proved, and examples of learning in 1 and 2 degree of freedom (DOF) manipulators are given.

Adaptive control of robot manipulators has been the subject of much research in recent years (see [Hsia]). Adaptation, in control, is the process of adjusting the controller to comply with the regulation and tracking requirements of the closed loop system. Direct adaptive controllers, e.g. [Li-Slotine 88], use tracking errors of the joint motion to direct the robot model parameter adjustment. The direct adaptive controllers are based on the full dynamic model of the robot. Learning, in this context is the identification of the true values of the manipulator parameters in closed loop operation.

In operation, the controller is given a trajectory by a path planner in order to accomplish some useful task. The controller then adapts the parameters, on line, so as to satisfy the tracking requirements. If the parameters are not known exactly, there will be a transient period of tracking error while adaptation occurs. So that identification of the true parameters is desirable for increased tracking precision.

Present robot adaptive controllers (see for example [Slotine-Li 87], [Craig], [Ortega-Spong]) use robot parameter update laws which rely on the assumption that the parameters are constants. Also, to guarantee learning, a trajectory which is "rich" enough to expose all the dynamics of the manipulator is required for convergence of the parameter estimates to their true values. Richness of input refers to trajectories such that the internal signals of the parameter adjustment mechanism is persistently exciting (see [Narendra-Annaswamy]). If the trajectory is not persistently exciting, stability is assured but not learning. Thus future tasks are performed with the same transient tracking errors.

We propose (see figure 1.1) that whenever it is possible to modify the input e.g. prior to putting the manipulator to work doing useful tasks or periodically when parameters change and the path planner is not demanding a new path, that there will be a learning period where the controller learns the true values of the parameters by tracking artificially designed Exploratory Schedules (ES). ES are trajectories specifically designed for asymptotic learning of the system dynamics. Any trajectory that is persistently exciting could be used as an exploratory schedule. However, the synthesis of persistently exciting trajectories is generally not straightforward. In this article, we show how to construct Exploratory Schedules which guarantee closed loop learning. Our ES are not necessarily persistently exciting.

The rest of the article is formatted as follows. Section 2 presents the general structure and properties of a rigid robot. Section 3 defines the adaptive control structure used, and presents sufficient conditions for global asymptotic learning of parameters which is not explicitly based on persistent excitation. Section 4 describes simulation results of learning with 1 and 2 degree of freedom (DOF) manipulators. Section 5 concludes the article.

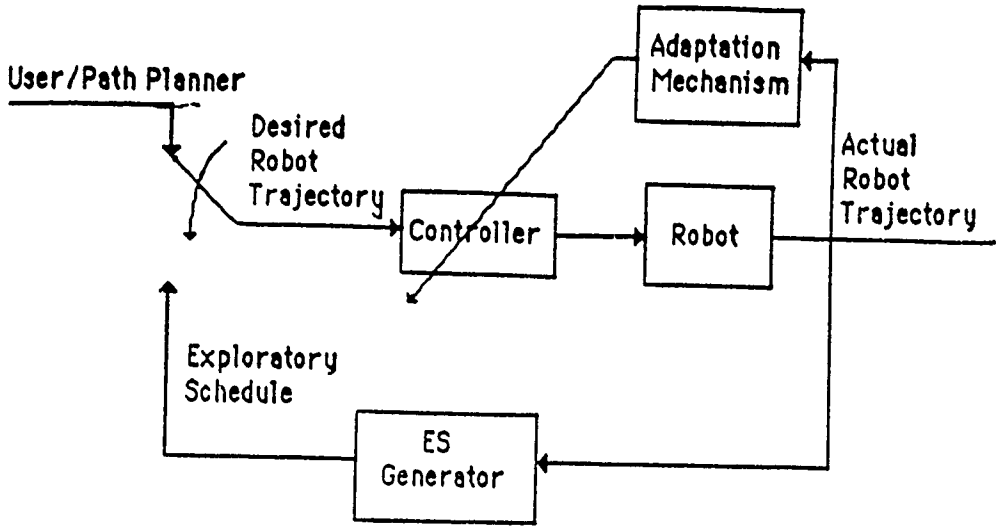


figure 1.1: Block diagram of the proposed system.

2. Rigid Robot Dynamics

2.1 Lagrange-Euler Representation of Rigid Robot Dynamics

A rigid robot is defined as an open kinematic chain of rigid links, which are joined by prismatic or revolute joints.

The Lagrange-Euler formulation of robot dynamics results in a closed form solution (the Lagrange-Euler formulation is well known in robotics literature, for a detailed derivation of the closed form solution see references [Fu et al.], [Paul 81], [Asada-Slotine]). The formulation requires an understanding of the spatial relationship between the links of the robot. This relationship is commonly represented by 4 x 4 homogeneous transformation matrices (see reference [Paul 81]), which relate the position and orientation of the ith link coordinate system to the (i-1)th coordinate system.

The closed form solution obtained by the Lagrange-Euler formulation has the general matrix form of

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad (2.1)$$

where

$D(q)$ is an n x n matrix of terms related to inertial forces acting on the robot,
$C(q,\dot{q})$ is an n x n matrix of terms related to coriollis and centrifugal forces ,
$G(q)$ is an n x 1 vector related to gravitational forces,
q is an n x 1 vector of generalized joint coordinates,

τ is an n x 1 vector of generalized forces/torques,

n is the number of degrees of freedom.

The formulation results in n second-order, coupled differential equations. If permanent-magnet-actuator dynamics are included in the formulation, another n first order differential equations are needed to describe the robot. Typically, a robot has 6 DOF, which results in an 18th order system which is coupled, time-varying, and highly nonlinear. Furthermore, the dynamics are dependent on the mass distribution of the robot, the exact values of which are rarely known at the time of controller design, and may change during use.

2.2 Properties of Robot Dynamics

It has been pointed out by a number of authors (see for example [Ortega-Spong]), that there are properties of the dynamics (2.1) that can be exploited for robot control. Three of the properties are repeated here, as they will be used in other sections.

Property 1: $D(q)$ is symmetric and positive definite $\forall\ q \Rightarrow D^{-1}(q)$ exists $\forall\ q$.

Property 2: $\dot{D}(q) - 2C(q,\dot{q})$ is skew symmetric $\Rightarrow \dot{q}^T(\dot{D}(q) - 2C(q,\dot{q}))\dot{q} = 0 \ \forall\ q$.

Property 3: $\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\dot{q},\ddot{q})\emptyset$

where $Y(q,\dot{q},\dot{q},\ddot{q})$ is an n x r matrix of known functions and $\emptyset$ is an r x 1 vector of constant parameters. Property 3 implies that the robot dynamics may be viewed as a linear operator from the parameters to the joint torques.

3. Closed Loop Learning Via Selection of Exploratory Schedules

3.1. Controller Structure

In [Slotine-Li], an adaptive controller which guarantees global asymptotic tracking was formulated. The controller is based on knowledge of the robot dynamics structure and the use of sliding surface control. A major advantage of this controller is that there is no need to measure the joint acceleration. The convergence of the parameter estimates to their true values, however is not guaranteed. In this work, the same control structure is used. However, we synthesize, whenever possible, the reference input to guarantee both tracking and learning.

For the system of equation (2.1), define the virtual reference trajectory

$$\begin{aligned} q_r &= q_d - A \int_0^t e\, dt \\ \dot{q}_r &= \dot{q}_d - A\, e \\ \ddot{q}_r &= \ddot{q}_d - A\, \dot{e} \end{aligned} \qquad (3.1)$$

where q_d is an n x1 vector of desired joint coordinates, $e = q - q_d$, and A is a positive definite matrix with constant coefficients. Define the sliding surface

$$s = \dot{e}_r = \dot{q} - \dot{q}_r = \dot{e} + Ae \qquad (3.2)$$
$$\dot{s} = \ddot{e}_r = \ddot{q} - \ddot{q}_r = \ddot{e} + A\dot{e} \ .$$

Let the control input be

$$\tau = \hat{D}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{G}(q) - K_d s = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\Phi} - K_d s \qquad (3.3)$$

where $\hat{(.)}$ is the estimate of (.), and K_d is an n x n positive definite matrix.
The parameter adaptation law is

$$\dot{\hat{\Phi}} = \dot{\tilde{\Phi}} = - K_a^{-1} Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)s \qquad (3.4)$$

where the estimation error is defined as $\tilde{(.)} = \hat{(.)} - (.)$, and K_a is an r x r positive definite matrix with constant elements.

3.2. Sufficient Conditions to Guarantee Learning of Rigid Robot Dynamics

3.2.1. Augmented State Space

Define the augmented state space

$$x = \begin{bmatrix} e \\ \dot{e} \\ \tilde{\Phi} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad (3.5)$$

Then the augmented system is

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ \ddot{e} \\ \dot{\hat{\Phi}} \end{bmatrix} \qquad (3.6)$$

Where $\dot{\hat{\Phi}}$ is as defined as in eq (3.4). To find $\ddot{e}$, combine equations (2.3) and (3.3),

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \hat{D}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{G}(q) - K_d s \ . \qquad (3.7)$$

Rewrite (3.7) in terms of

$$e = q - q_d \Rightarrow q = e + q_d \ ,$$
$$\dot{e} = \dot{q} - \dot{q}_d \Rightarrow \dot{q} = \dot{e} + \dot{q}_d \ ,$$
$$\ddot{e} = \ddot{q} - \ddot{q}_d \Rightarrow \ddot{q} = \ddot{e} + \ddot{q}_d \ ,$$
$$s = \dot{e} + Ae \ ,$$
$$\dot{q}_r = \dot{q}_d - A\,e \ ,$$
$$\ddot{q}_r = \ddot{q}_d - A\,\dot{e} \ ,$$

$$D(\ e+q_d\ )(\ \ddot{e}+\ddot{q}_d\ )+C(\ e+q_d\ ,\dot{e}+\dot{q}_d\ )(\ \dot{e}+\dot{q}_d\ )+G(\ e+q_d\ ) =$$

$$= \hat{D}(e+q_d)(\ddot{q}_d - A\dot{e}) + \hat{C}(e+q_d, \dot{e}+\dot{q}_d)(\dot{q}_d - Ae) + \hat{G}(e+q_d) - K_d(\dot{e} + Ae).$$

Then

$$D(e+q_d)\ddot{e} = \tilde{D}(e+q_d)\ddot{q}_d + \tilde{C}(e+q_d, \dot{e}+\dot{q}_d)\dot{q}_d + \tilde{G}(e+q_d) - \hat{D}(e+q_d)A\dot{e} - \hat{C}(e+q_d, \dot{e}+\dot{q}_d)Ae - C(e+q_d, \dot{e}+\dot{q}_d)\dot{e} - K_d(\dot{e} + Ae). \quad (3.8)$$

Note that
$$\begin{aligned} Y(q,\dot{q},\dot{q}_d,\ddot{q}_d)\tilde{\Phi} &= \tilde{D}(q)\ddot{q}_d + \tilde{C}(q,\dot{q})\dot{q}_d + \tilde{G}(q), \\ &= \tilde{D}(e+q_d)\ddot{q}_d + \tilde{C}(e+q_d, \dot{e}+\dot{q}_d)\dot{q}_d + \tilde{G}(e+q_d), \\ &= Y(e+q_d, \dot{e}+\dot{q}_d, \dot{q}_d, \ddot{q}_d)\tilde{\Phi}, \end{aligned}$$

$$\begin{aligned} Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{\Phi} &= \tilde{D}(q)\ddot{q}_r + \tilde{C}(q,\dot{q})\dot{q}_r + \tilde{G}(q), \\ &= \tilde{D}(e+q_d)(\ddot{q}_d - A\dot{e}) + \tilde{C}(e+q_d, \dot{e}+\dot{q}_d)(\dot{q}_d - Ae) + \tilde{G}(e+q_d), \\ &= Y(e+q_d, \dot{e}+\dot{q}_d, \dot{q}_d - Ae, \ddot{q}_d - A\dot{e})\tilde{\Phi}, \end{aligned}$$

and

$$D(q) > 0 \ \forall\, q \ \Rightarrow\ D(e+q_d) > 0 \ \forall\ e+q_d.$$

Then (3.8) can be written as

$$D(e+q_d)\ddot{e} = Y(e+q_d, \dot{e}+\dot{q}_d, \dot{q}_d, \ddot{q}_d)\tilde{\Phi} - \hat{D}(e+q_d)A\dot{e} - \hat{C}(e+q_d, \dot{e}+\dot{q}_d)Ae - C(e+q_d, \dot{e}+\dot{q}_d)\dot{e} - K_d(\dot{e} + Ae), \quad (3.9)$$

and $\ddot{e}$ is

$$\ddot{e} = D(e+q_d)^{-1}[Y(e+q_d, \dot{e}+\dot{q}_d, \dot{q}_d, \ddot{q}_d)\tilde{\Phi} - \hat{D}(e+q_d)A\dot{e} - \hat{C}(e+q_d, \dot{e}+\dot{q}_d)Ae - C(e+q_d, \dot{e}+\dot{q}_d)\dot{e} - K_d(\dot{e} + Ae)]. \quad (3.10)$$

Then the augmented system is

$$\dot{x} = \begin{bmatrix} x_2 \\ D(x_1+q_d)^{-1}\left\{ Y(x_1+q_d, x_2+\dot{q}_d, \dot{q}_d, \ddot{q}_d)x_3 - \hat{D}(x_1+q_d)Ax_2 - \hat{C}(x_1+q_d, x_2+\dot{q}_d)Ax_1 \right. \\ \left. - C(x_1+q_d, x_2+\dot{q}_d)x_2 - K_d(x_2+Ax_1) \right\} \\ - K_a^{-1} Y^T(x_1+q_d, x_2+\dot{q}_d, \dot{q}_d-Ax_1, \ddot{q}_d-Ax_2)(x_2+Ax_1) \end{bmatrix} \quad (3.11)$$

Equation (3.11) describes the closed loop dynamics of the augmented system. We emphasize that asymptotic stability of its origin implies both asymptotic tracking and asymptotic learning.

3.2.2 Global asymptotic stability of the augmented system.

Theorem

If $r \leq n$, and if

$$\text{Rank} \left\{ \lim_{t \to \infty} Y(q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d) \right\} = r$$

then the origin of system (3.11) is globally asymptotically stable

Proof

Write system dynamics (eq. (2.3)) in terms of $\dot{q} = s + \dot{q}_r$, $\ddot{q} = \dot{s} + \ddot{q}_r$ and substitute for τ using eq (3.1) to (3.4)

$$\begin{aligned} D(q)(\dot{s} + \ddot{q}_r) + C(q,\dot{q})(s + \dot{q}_r) + G(q) &= \hat{D}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{G}(q) - K_d s, \\ D(q)\dot{s} &= \hat{D}(q)\ddot{q}_r - D(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r - C(q,\dot{q})\dot{q}_r + \hat{G}(q) - G(q) - C(q,\dot{q})s - K_d s, \\ D(q)\dot{s} &= \tilde{D}(q)\ddot{q}_r + \tilde{C}(q,\dot{q})\dot{q}_r + \tilde{G}(q) - C(q,\dot{q})s - K_d s \ . \end{aligned} \quad (3.12)$$

Define the Lyapunov candidate function

$$V = 1/2\ s^T D(q)s + 1/2\ \tilde{\Theta}^T K_a \tilde{\Theta} \ . \quad (3.13)$$

which is positive definite. The derivative of V is

$$\dot{V} = 1/2\ \dot{s}^T D(q)s + 1/2\ s^T \dot{D}(q,\dot{q})s + 1/2\ s^T D(q)\dot{s} - 1/2\ \dot{\Theta}^T K_a \tilde{\Theta} + 1/2\ \tilde{\Theta}^T K_a \dot{\tilde{\Theta}} \ . \quad (3.14)$$

The terms in (3.14) are scalars and therefore symmetric, then $\dot{V}$ can be written as

$$\dot{V} = s^T D(q)\dot{s} + 1/2\, s^T \dot{D}(q)s + \tilde{\emptyset}^T K_a \tilde{\emptyset} \,. \tag{3.15}$$

Substituting for $D\dot{s}$ from (3.12)

$$\dot{V} = s^T[\, \tilde{D}(q)\ddot{q}_r + \tilde{C}(q,\dot{q})\dot{q}_r + \tilde{G}(q) - C(q,\dot{q})s - K_d s \,] + 1/2\, s^T \dot{D}(q)s + \tilde{\emptyset}^T K_a \tilde{\emptyset} \,.$$

Combining and rearranging terms and using the property $s^T[\dot{D}(q) - 2C(q,\dot{q})]s = 0$,

$$\dot{V} = s^T[\, \tilde{D}(q)\ddot{q}_r + \tilde{C}(q,\dot{q})\,\dot{q}_r + \tilde{G}(q)\,] - s^T K_d\, s + 1/2\, s^T[\dot{D}(q) - 2C(q,\dot{q})]\, s + \tilde{\emptyset}^T K_a \tilde{\emptyset} \,.$$

$$\dot{V} = -\, s^T K_d s + s^T[\, \tilde{D}(q)\ddot{q}_r + \tilde{C}(q,\dot{q})\,\dot{q}_r + \tilde{G}(q)\,] + \tilde{\emptyset}^T K_a \tilde{\emptyset} \,. \tag{3.16}$$

Which can be written as

$$\dot{V} = -\, s^T K_d s + \tilde{\emptyset}^T Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)s + \tilde{\emptyset}^T K_a \tilde{\emptyset} \,. \tag{3.17}$$

Substituting for $\tilde{\emptyset}$ from (3.4)

$$\dot{V} = -\, s^T K_d s + \tilde{\emptyset}^T Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)s - \tilde{\emptyset}^T K_a\, K_a^{-1}\, Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)s,$$

$$\dot{V} = -\, s^T K_d s \le 0 \,. \tag{3.18}$$

$\dot{V} < 0 \;\forall\; s \neq 0 \Rightarrow s \to 0.$

When $s = 0$, $\dot{x}_1 = -Ax_1 \Rightarrow$ x_1 and x_2 approach the origin exponentially on the sliding surface s. This in turn from eq. (3.11) implies that $\dot{x}_3 \to 0$.

Then

$$\lim_{t \to \infty} \dot{x} = \begin{bmatrix} 0 \\ D(q_d)^{-1} Y(q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d)x_3 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.19}$$

which when the condition

$$\text{Rank} \left\{ \lim_{t \to \infty} Y(q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d) \right\} = r$$

holds implies that $x_3 \to 0$.

QED

3.3. Selection of ES

An implication of the theorem is that if r > n, a sufficient condition for k ≤n parameters to be identified is that they are are not in the null space of

$$\lim_{t \to \infty} Y(q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d)$$

Which implies that a desired trajectory may be chosen such that columns corresponding to the k ≤n parameters are linearly independent, which will guarantee that the parameters are identified. So that different time sections of the Exploratory Schedule (which specifies $q_d, \dot{q}_d, \ddot{q}_d$) may be designed to learn different components of the vector x_3 it its dimension exceeds n.

4. SIMULATION RESULTS

4.1 Examples

4.1.1 Manipulator with 1 DOF

The first simulation was done for a single link manipulator as shown in figure 4.1

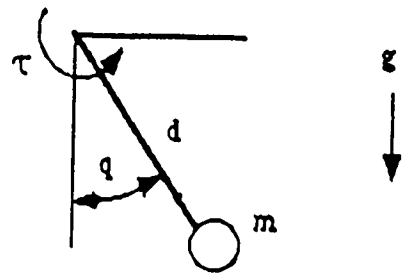


figure 4.1: Simulated 1 DOF manipulator.

The dynamics of the manipulator are

$$\tau = m\, d^2\, \ddot{q} + m\, g\, d \sin(q), \tag{4.1}$$

where, d = 1, g = 10, and m = 2 . The matrix of known functions is

$$Y(q, \ddot{q}) = [\, d^2\, \ddot{q} + g\, d \sin(q)], \tag{4.2}$$

with m as the single parameter to be estimated. The control law as specified by eq (3.3) is

$$\tau = [\, d^2\, \ddot{q}_r + g\, d \sin(q)]\hat{m} - k_d(\dot{q} - \dot{q}_r), \tag{4.3}$$

where $\dot{q}_r = \dot{q}_d - (q - q_d)$, kd = 1, and the parameter adaptation law as defined by (3.4) was used to estimate m, where the estimate is denoted as $\hat{m}$.

In experiment 1a,

$$\dot{q}_d = 0.0,\ q_d = 0.0 \Rightarrow \text{Rank}\begin{pmatrix} Y(q_d, \ddot{q}_d) \\ \text{Lim } t \to \infty \end{pmatrix} = 0$$

which is less than r = n = 1. Notice in figure 4.2, that even though the tracking error does converge to zero, the estimate of m does not converge to its true value of 2.0 .

In experiment 1b,

$$\dot{q}_d = 0.0,\ q_d = 1.0 \Rightarrow \text{Rank}\begin{pmatrix} Y(q_d, \ddot{q}_d) \\ \text{Lim } t \to \infty \end{pmatrix} = 1$$

which equals n. In the second experiment, shown in figure 4.3, the estimate of m does converge to its true value. Thus closed loop learning is obtained.

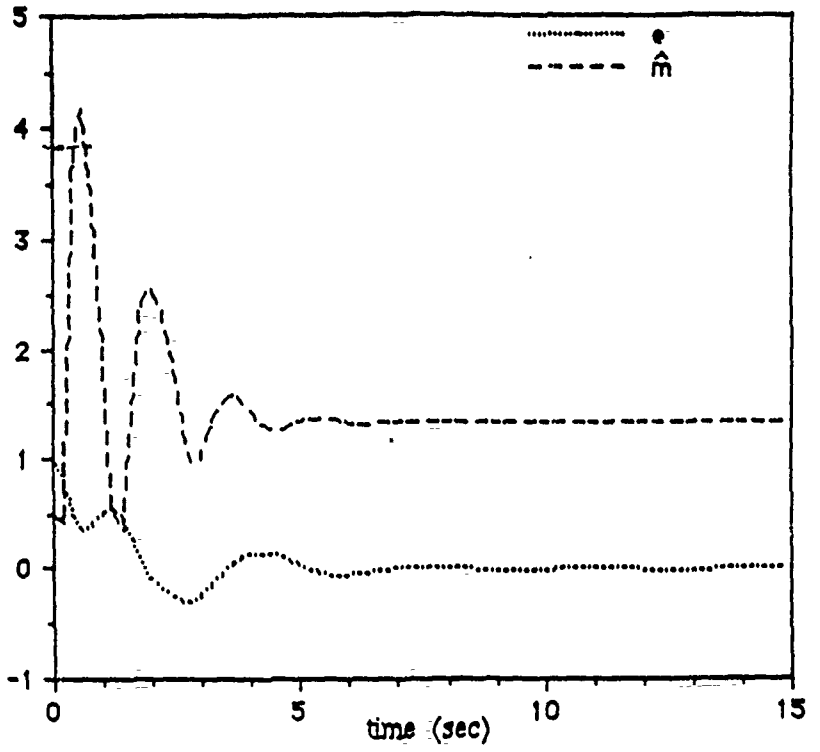


figure 4.2: Experiment 1a, trajectory chosen such that Y() is not full rank.

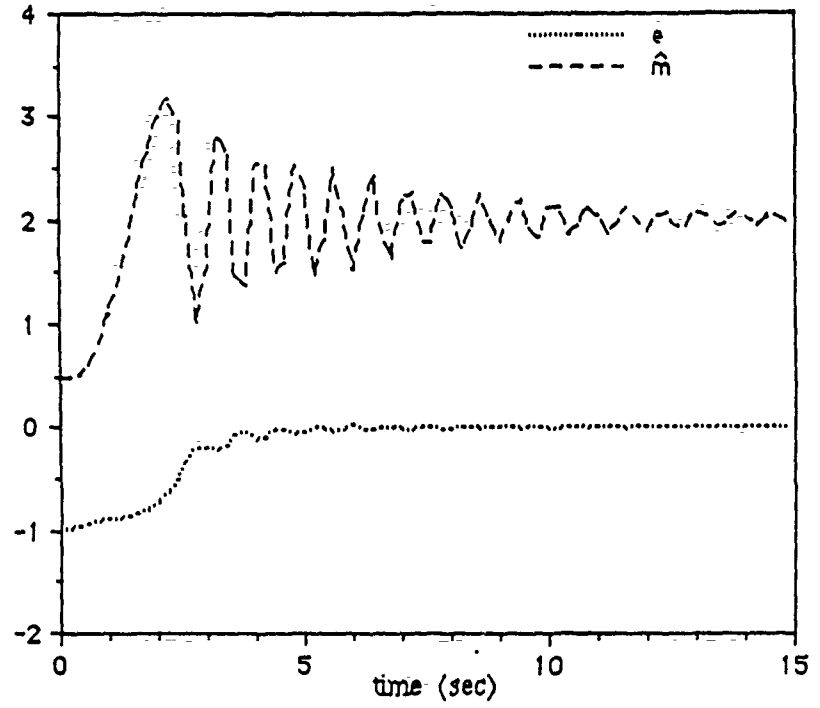


figure 4.3: Experiment 1b, trajectory chosen such that Y() is full rank.

4.1.2. Manipulator with 2 DOF

The second simulation was done for a 2 DOF manipulator as shown in figure 4.4

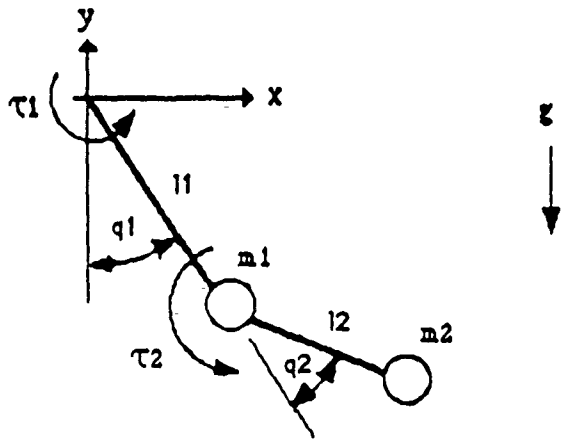


figure 4.4: Simulated 2 DOF manipulator.

The dynamics of the 2 DOF manipulator are

$$\tau_1 = D_{11}\ddot{q}_1 + D_{12}\ddot{q}_2 + 2C\dot{q}_1\dot{q}_2 + C\dot{q}_2^2 + G_1 \qquad (4.4)$$
$$\tau_2 = D_{12}\ddot{q}_1 + D_{22}\ddot{q}_2 - C\dot{q}_1^2 + G_2$$

where

$D_{11} = m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 \cos(q_2)$,
$D_{12} = m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)$,
$D_{22} = m_2 l_2^2$,
$C = m_2 l_1 l_2 \sin(q_2)$
$G_1 = g(m_1 + m_2) l_1 \sin(q_1) + g m_2 l_2 \sin(q_1 + q_2)$
$G_2 = g m_2 l_2 \sin(q_1 + q_2)$,

with $m_1 = m_2 = 10.0$, $l_1 = l_2 = 1.0$, $g = 9.81$.

The parameters are defined as

$$\emptyset_1 = (m_1+m_2) l_1^2 = 20.0 \qquad \emptyset_4 = g(m_1 + m_2) l_1 = 196.2$$
$$\emptyset_2 = m_2 l_2^2 = 10.0 \qquad \emptyset_5 = g m_2 l_2 = 98.1$$
$$\emptyset_3 = m_2 l_1 l_2 = 10.0. \qquad (4.5)$$

Which leads to the reparameterization

$$Y(q, \dot{q}, \dot{q}, \ddot{q}) = \begin{bmatrix} \ddot{q}_1 & \ddot{q}_1+\ddot{q}_2 & \cos(q_2)(2\ddot{q}_1+\ddot{q}_2)-\sin(q_2)(2\dot{q}_1\dot{q}_2+\dot{q}_2^2) & \sin(q_1) & \sin(q_1+q_2) \\ 0 & \ddot{q}_1+\ddot{q}_2 & \cos(q_2)\ddot{q}_1+\sin(q_2)\dot{q}_1^2 & 0 & \sin(q_1+q_2) \end{bmatrix} \qquad (4.6)$$

The control law as specified by eq (3.3) is

$$\tau = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{\emptyset} - K_d\, s, \qquad (4.7)$$

where $s = \text{diag}(\dot{e}_1+e_1, \dot{e}_2+e_2)$, $K_d = \text{diag}(1000, 500)$, and the parameter adaptation law as defined by (3.4) was used to update the parameter estimates $\hat{\emptyset}$, with $K_a^{-1} = \text{diag}(1000, 1000)$. The dimension of $Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$ in the 2 DOF case is 2 x 5 so that r >n, and at most 2 parameters can be guaranteed to be learned simultaneously. The initial conditions for both experiments (2a and 2b) are at

$$q_1(0) = q_2(0) = \dot{q}_1(0) = \dot{q}_2(0) = 0\,,$$

and the desired velocities for both experiments is

$$\dot{q}_{d1}(0) = \dot{q}_{d2}(0) = 0\,.$$

In experiment 2a,

$$q_{d1} = 0.5,\ q_{d2} = -0.5 \Rightarrow \text{Rank}\begin{pmatrix} Y(q_d, \ddot{q}_d) \\ \text{Lim } t \to \infty \end{pmatrix} = 1\,,$$

which is less than min(r, n) = 2. Notice in figure 4.4, that even though the tracking error (which is not shown) does converge to zero, the estimates of $\emptyset_4$ and $\emptyset_5$ do not converge to the true values.

In experiment 2b,

$$q_{d1} = 0.5,\ q_{d2} = 0.5 \Rightarrow \text{Rank}\begin{pmatrix} Y(q_d, \ddot{q}_d) \\ \text{Lim } t \to \infty \end{pmatrix} = 2\,,$$

which equals n. The linearly independent columns for experiment 2b correspond to parameters $\emptyset_4$ and $\emptyset_5$. As shown in figure 4.5, the estimates of $\emptyset_4$ and $\emptyset_5$ do converge to the true values.

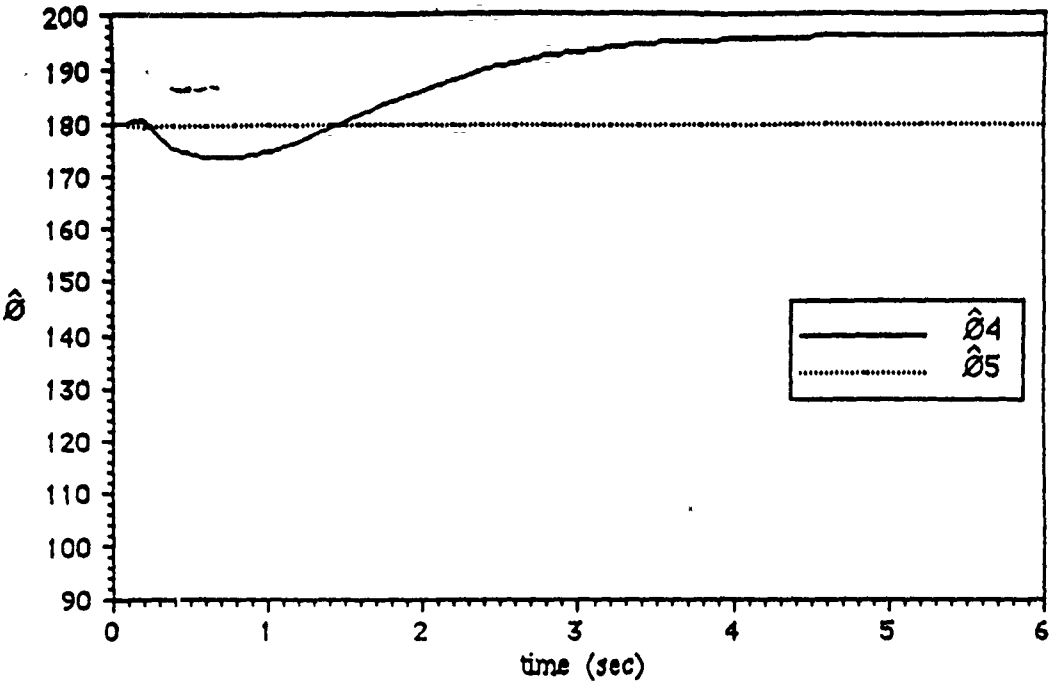


figure 4.4: Experiment 2a, trajectory chosen such that Y() is not full rank.

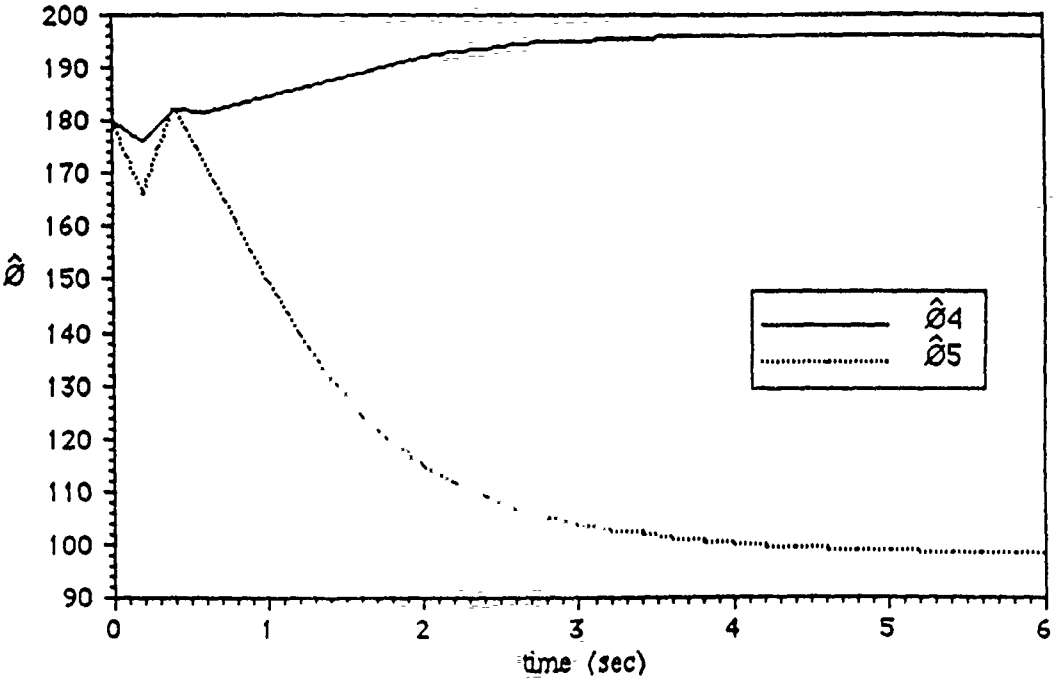


figure 4.5: Experiment 2b, trajectory chosen such that Y() is full rank.

4.2. Exploratory schedule for the 2 DOF manipulator

Selection of the exploratory schedule for the 2 DOF manipulator was accomplished by noting that columns 4 and 5 of (4.7) are functions only of position, therefore selecting $\ddot{q}_{d1}=\ddot{q}_{d2}=\dot{q}_{d1}=\dot{q}_{d2}=0.0$, and q_{d1}, q_{d2} such that columns 4 and 5 are nonzero as e_1, e_2, $\dot{e}_1$, $\dot{e}_2$ -> 0 will ensure that parameters $\emptyset_4$ and $\emptyset_5$ are identified. Next, selecting $\ddot{q}_{d1}=\ddot{q}_{d2}=0.0$, and q_{d1}, q_{d2}, $\dot{q}_{d1}$, $\dot{q}_{d2}$ such that column 3 is nonzero in the limit ensures identification of parameter $\emptyset_3$. Then selecting $\ddot{q}_{d1}$, $\ddot{q}_{d2}$ such that columns 1 and 2 are nonzero ensures the identification of parameters $\emptyset_4$ and $\emptyset_5$. The actual exploratory schedule is as shown in figures 4.6 to 4.8 . Theoretically, a parameter is guaranteed to be identified when $e_1=e_2=\dot{e}_1=\dot{e}_2= 0.0$. However, in practice, due to noise and disturbances, exact tracking may not occur and small tracking errors may lead to small errors in the parameter identification. In this simulation, a parameter is said to be identified when Maximum($|e_1|$, $|e_2|$,$|\dot{e}_1|$, $|\dot{e}_2|$) $\leq e_{tol}$, where e_{tol} = 0.01, and a further stipulation that attempted identification of each parameter is to last at least 3 seconds. The time period corresponding to identification of $\emptyset_4$ and $\emptyset_5$ is $0 \leq t < 4.2$. As can be seen in figures 4.9 and 4.11, the parameter estimation error for $\emptyset_4$ and $\emptyset_5$ approaches zero as all joint errors fall below 0.01 at t=4.2 . The time period corresponding to identification of $\emptyset_3$ is $4.2 \leq t < 7.3$. The parameter estimation error for $\emptyset_3$ approaches zero as all joint errors approach zero at t=7.3 . Parameters $\emptyset_4$ and $\emptyset_5$ are identified during the time period $7.3 < t \leq 10.3$.

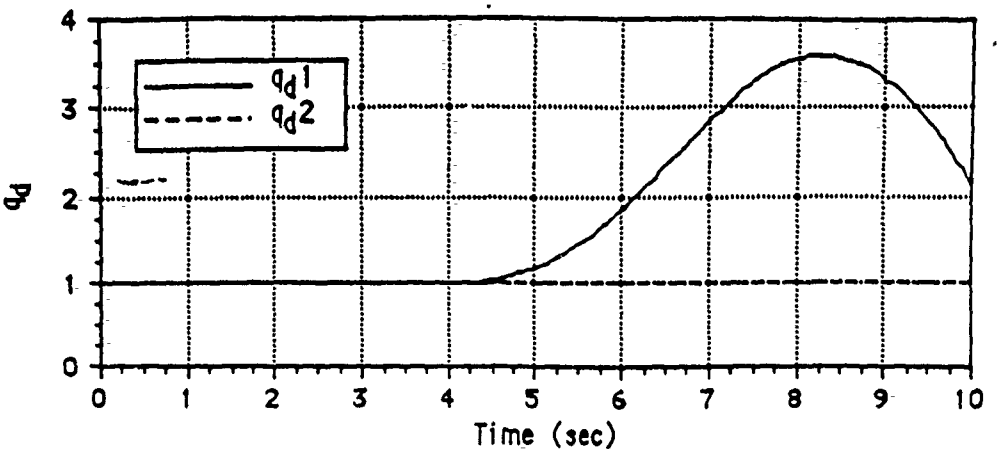


figure 4.6: Desired joint positions during exploratory schedule.

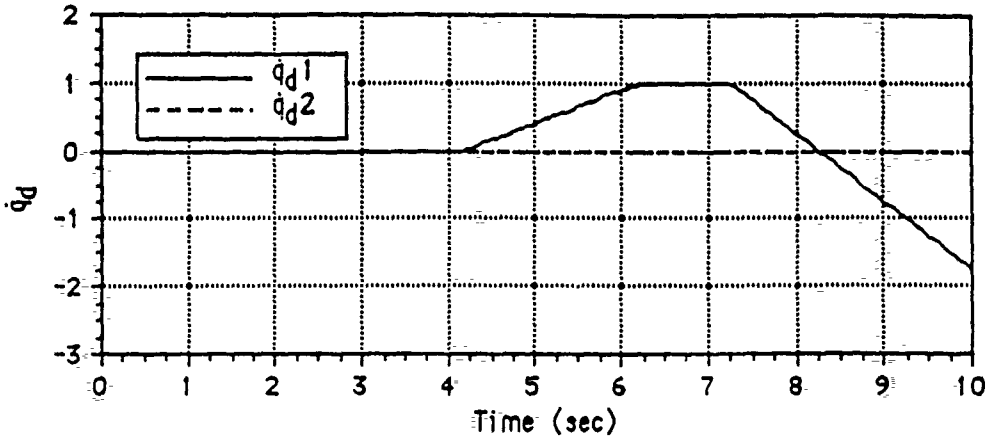


figure 4.7: Desired joint velocities during exploratory schedule.

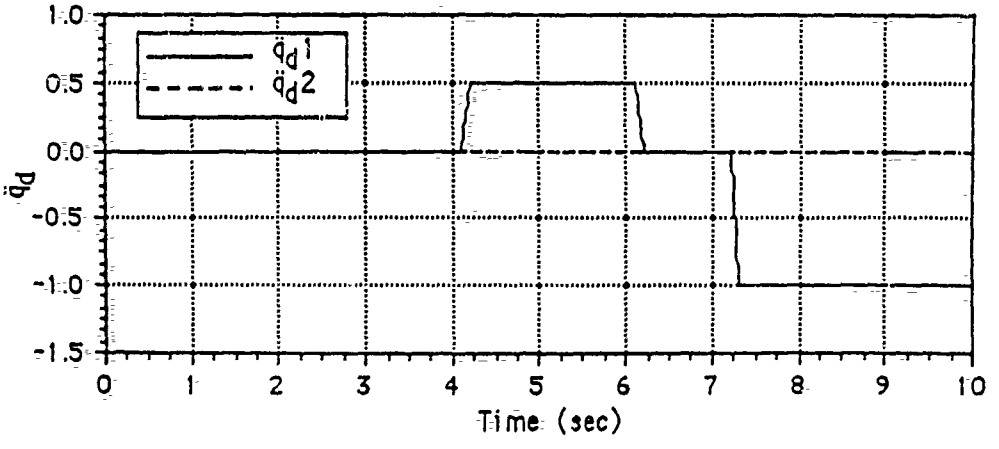


figure 4.8: Desired joint accelerations during exploratory schedule.

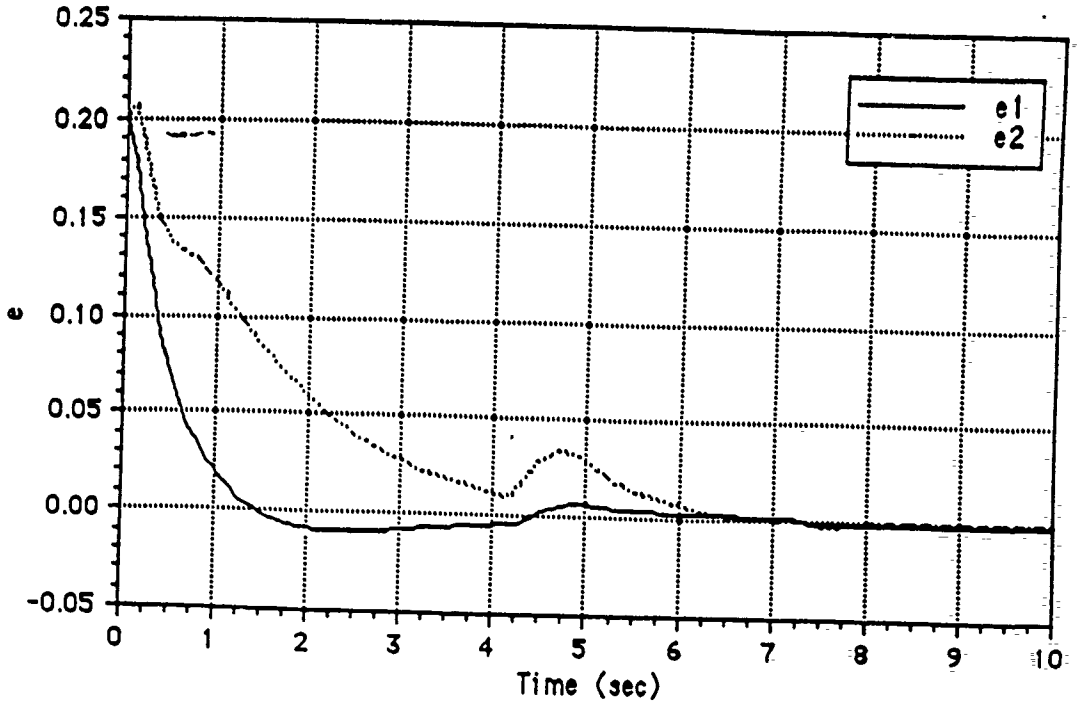


figure 4.9: Joint position error during exploratory schedule.

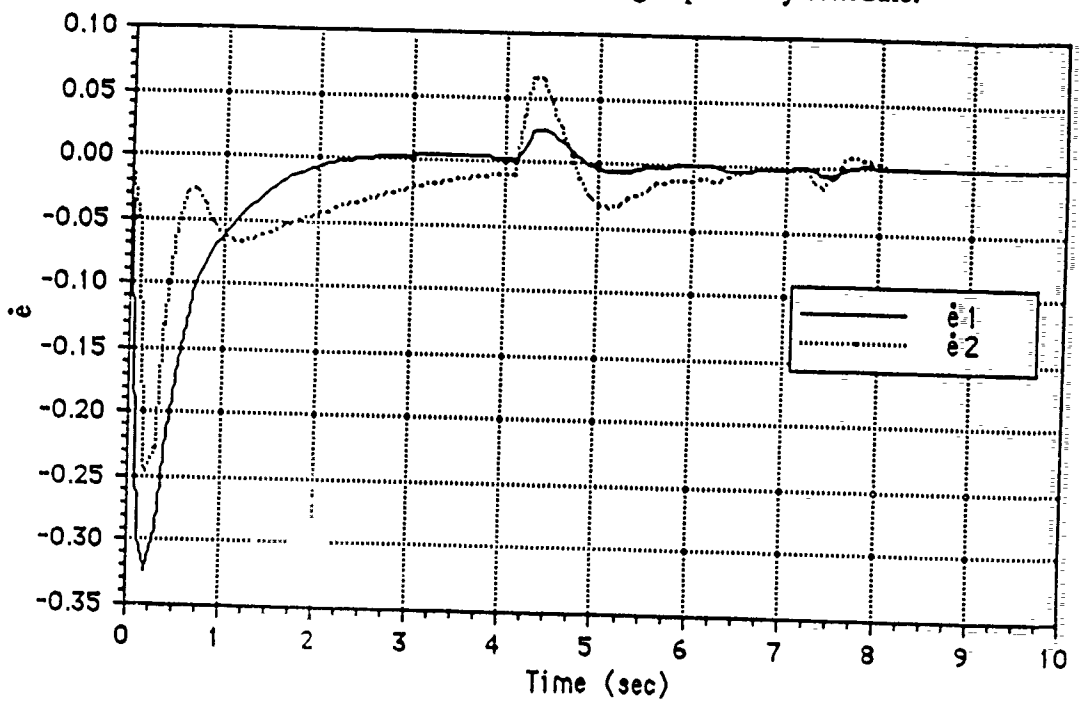


figure 4.10: Joint velocity error during exploratory schedule.

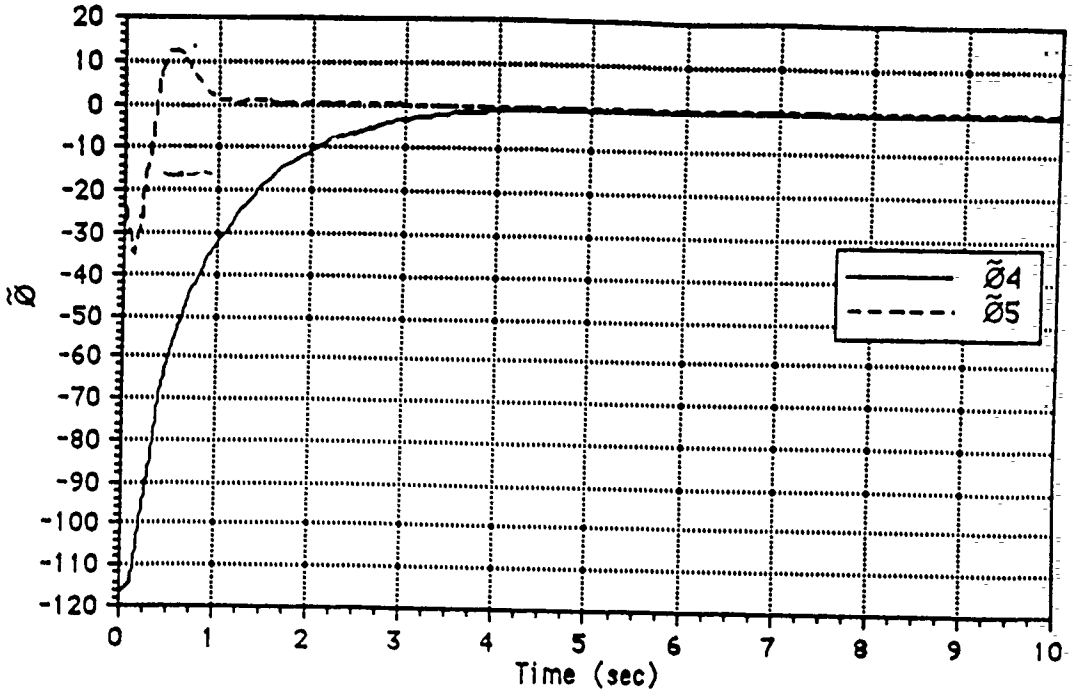


figure 4.11: Parameter estimation error for parameters $Ø_4$ and $Ø_5$ during exploratory schedule.

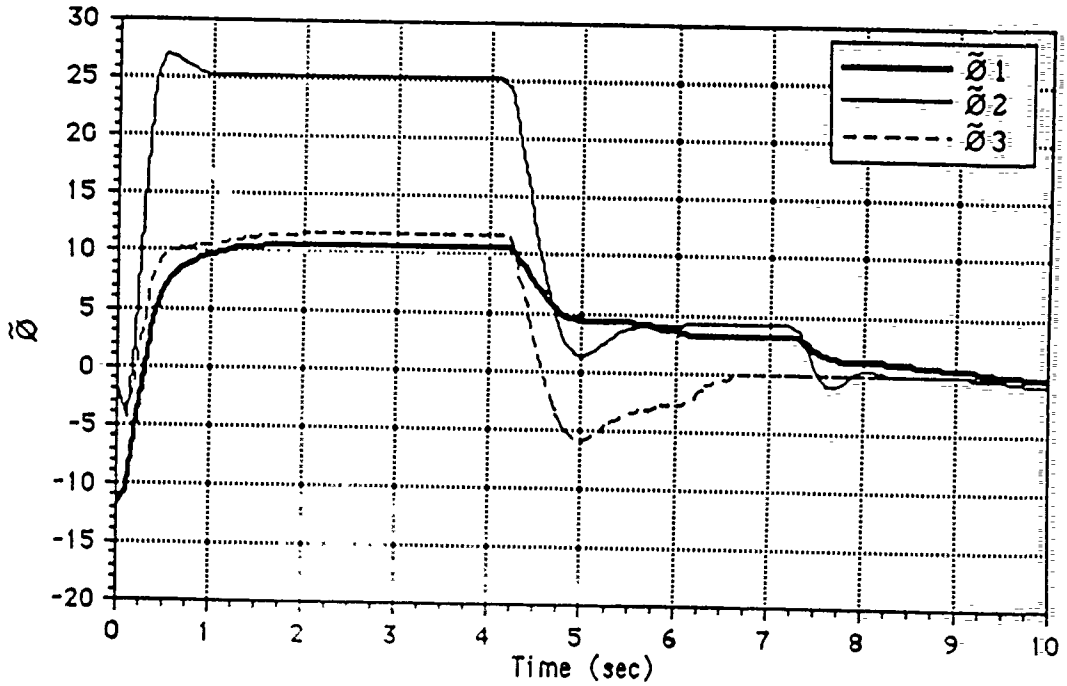


figure 4.12: Parameter estimation error for parameters $Ø_1$, $Ø_2$ and $Ø_3$ during exploratory schedule.

4.3. Comparison of controller performance before and after learning

A set of experiments was performed to compare the performance of the adaptive controller in tracking a test trajectory for two experiments: 3a) without prior learning of the parameters through tracking of the exploratory schedule, and 3b) with prior tracking of the exploratory schedule. The test trajectory is shown in figures 4.13 to 4.15. In experiment 3a, tracking of the test trajectory was attempted by the controller. Two attempts were made by the controller to track this trajectory. In experiment 3b, the controller first followed the exploratory schedule as detailed in section 4.2 , then the same test trajectory as used in experiment 3a was attempted. Figures 4.16 to 4.18 show the sampled Cartesian path of the end effector (i.e. the tip of link 2) as the controller attempts the trajectory. Although there is improvement in performance in the two attempts without learning via the exploratory schedule (figures 4.16 and 4.17), figure 4.18 shows that the tracking error is greatly reduced after the controller first learns the parameters by tracking the exploratory schedule. The increase in performance is more apparent when viewed in the joint space, as shown by the graphs of joint position and velocity errors in figures 4.19 to 4.22 .

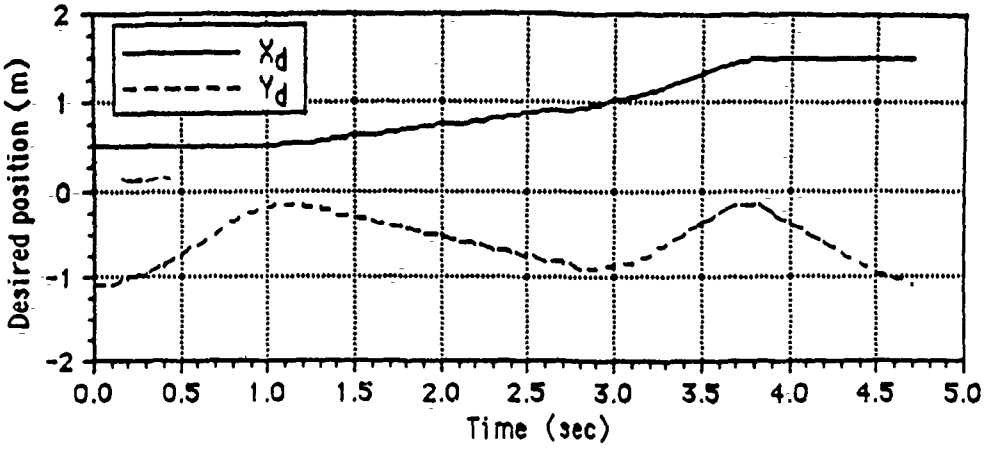


figure 4.13: Desired Cartesian position of the end effector during the test trajectory.

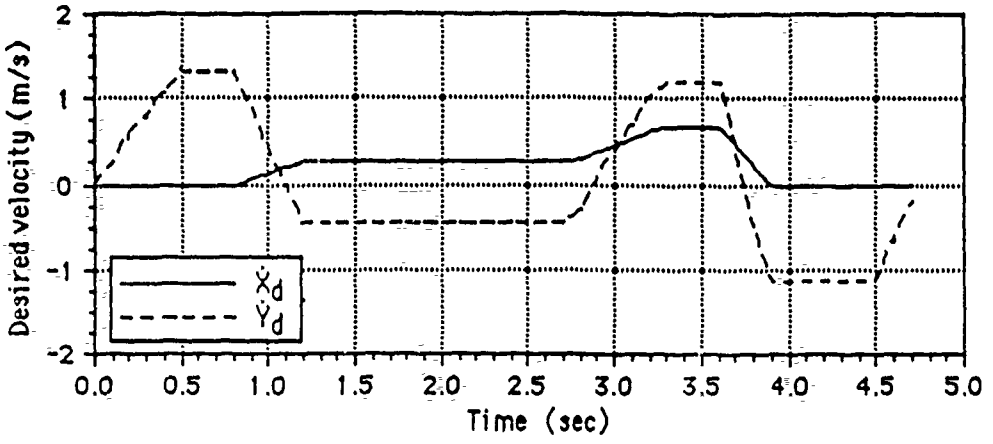


figure 4.14: Desired Cartesian velocity of the end effector during the test trajectory.

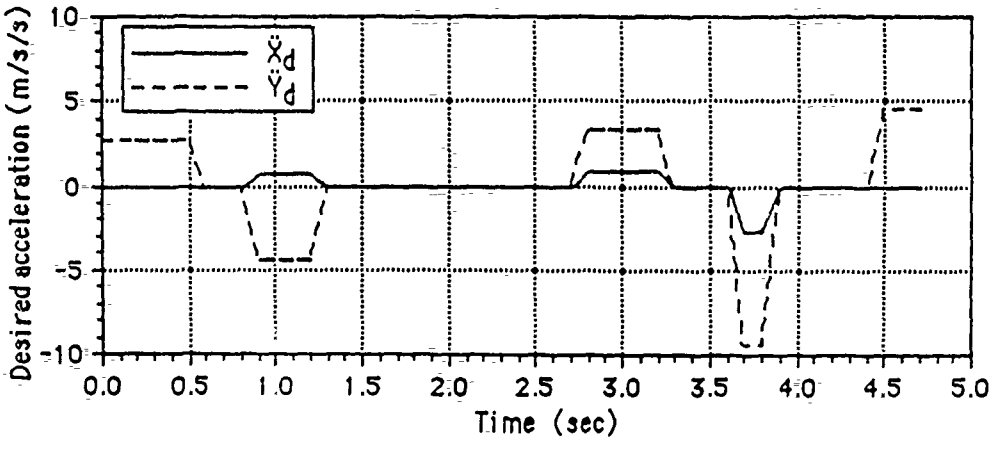


figure 4.13: Desired Cartesian acceleration of the end effector during the test trajectory.

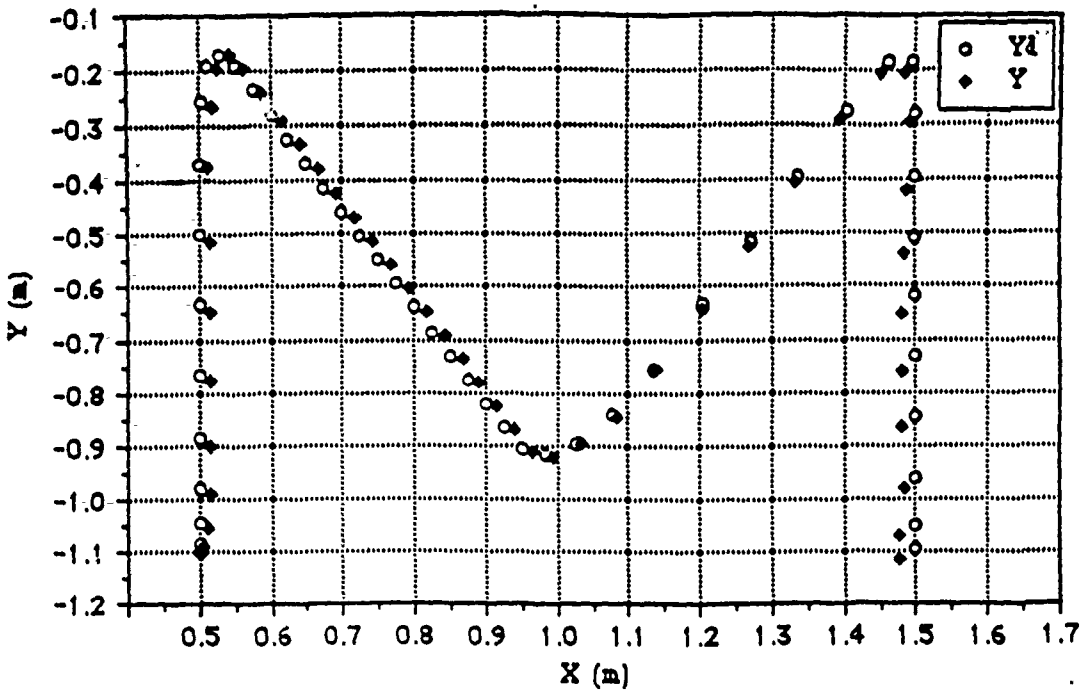


figure 4.16: Path of the end effector during first attempt by the controller, prior to learning via exploratory schedule.

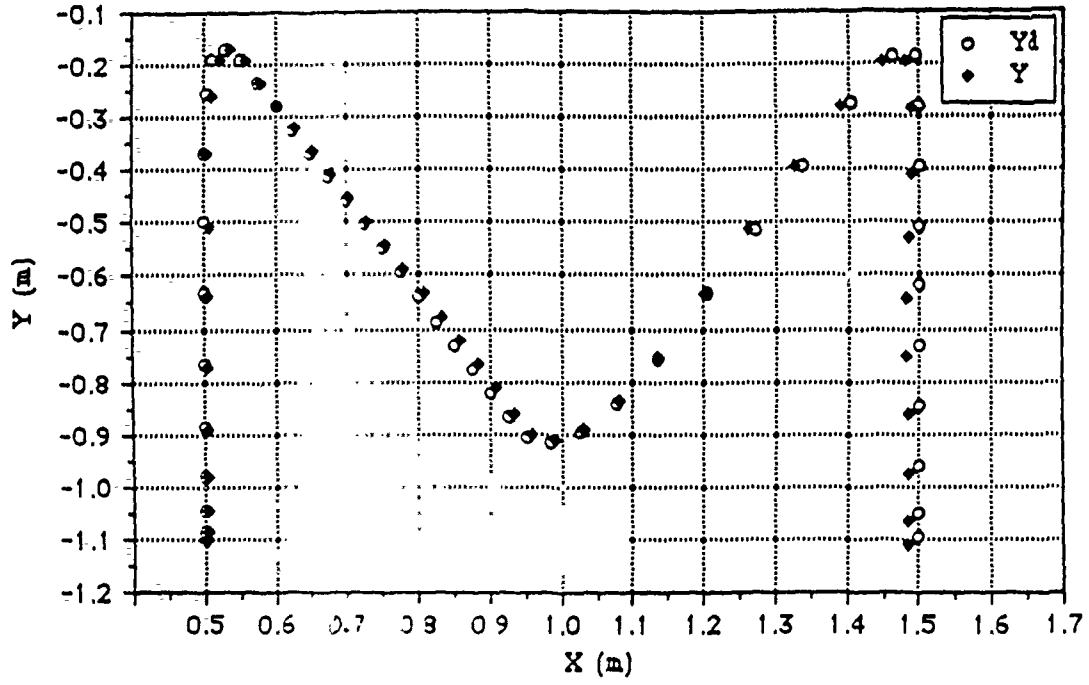


figure 4.17: Path of the end effector during second attempt by the controller, prior to learning via exploratory schedule.

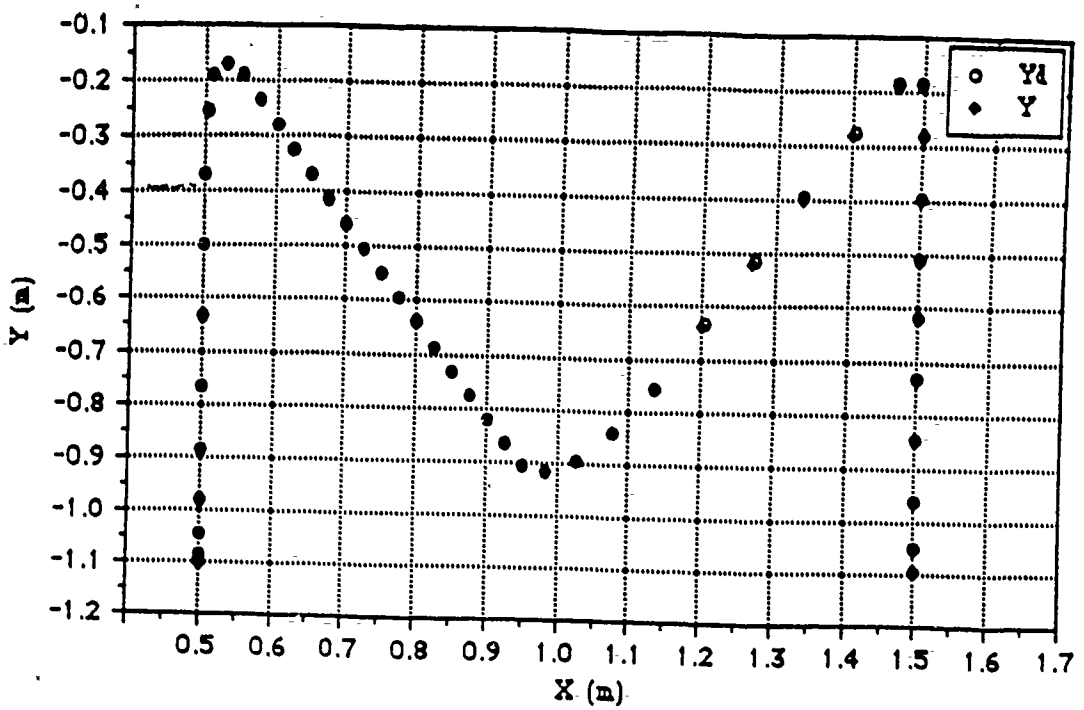


figure 4.18: Path of the end effector during first attempt by the controller, after learning via exploratory schedule.

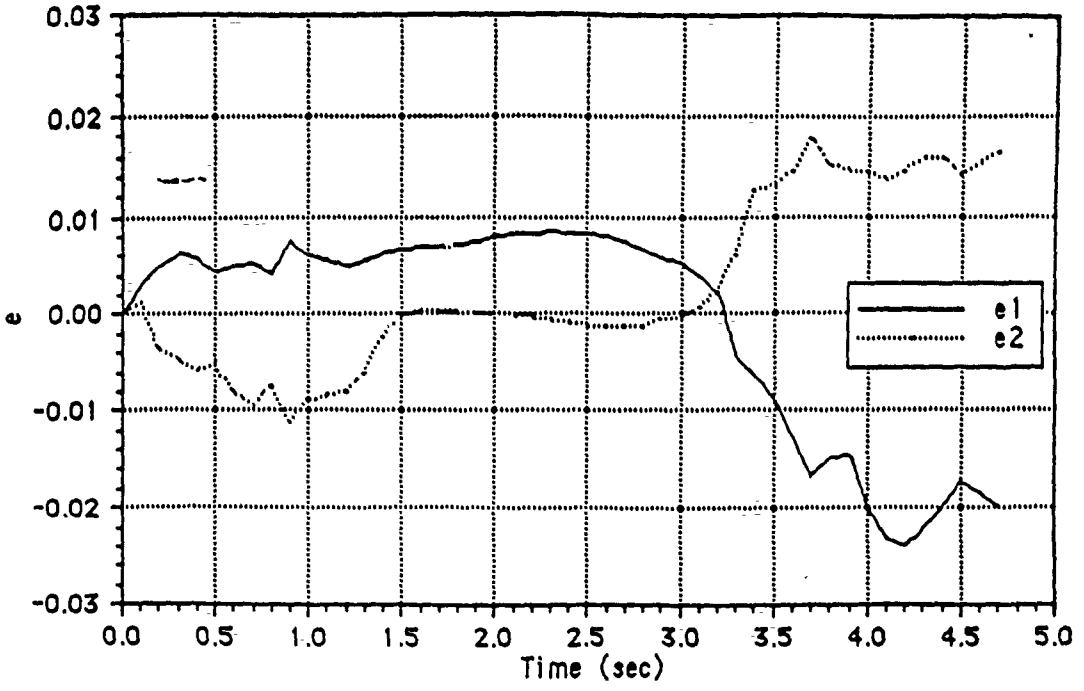


figure 4.19: Joint position errors during second attempt by the controller, prior to learning via exploratory schedule.

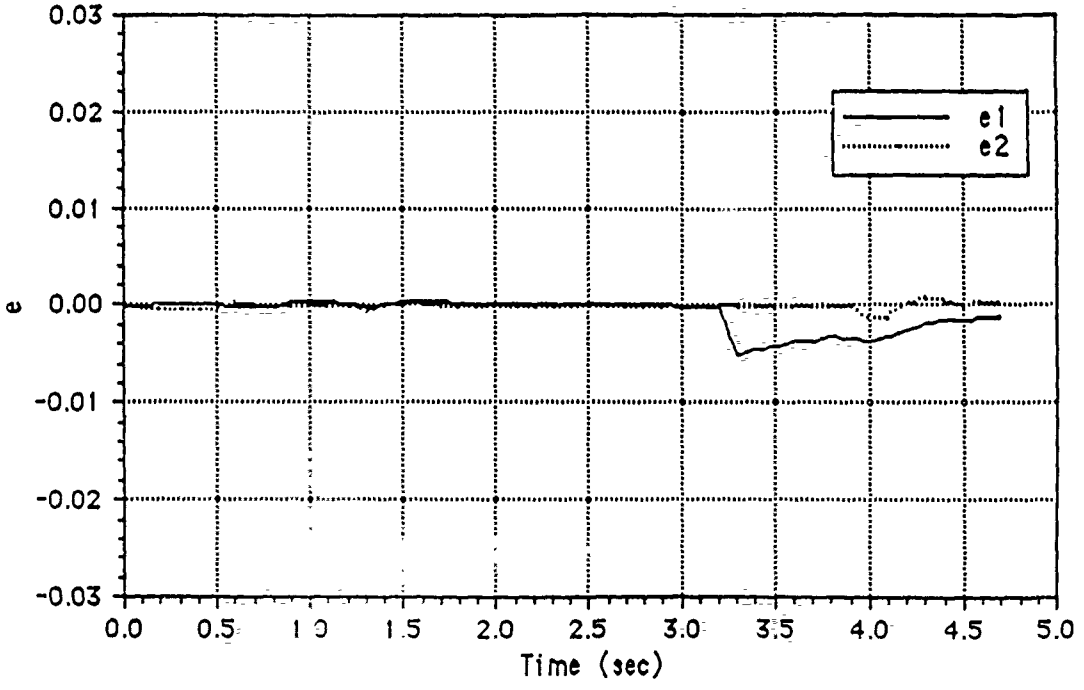


figure 4.20: Joint position errors during first attempt by the controller, after learning via exploratory schedule.

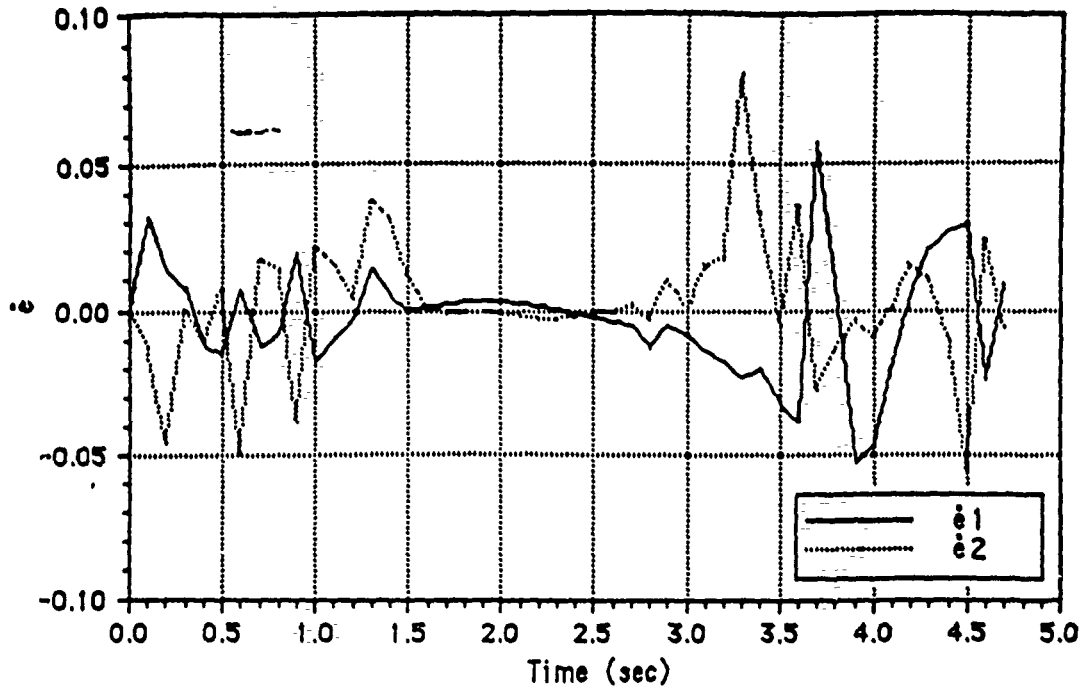


figure 4.21: Joint velocity errors during second attempt by the controller, prior to learning via exploratory schedule.

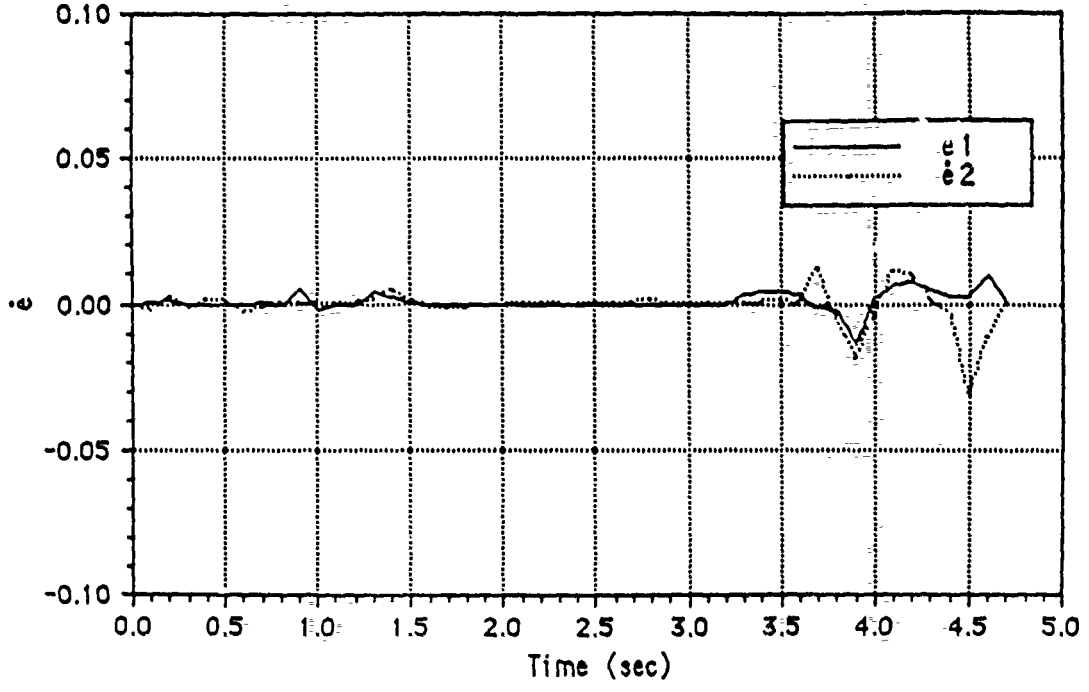


figure 4.22: Joint velocity errors during first attempt by the controller, after learning via exploratory schedule.

5. Conclusion

In this article, we explored the relationship of learning and adaptation in robot control. We have shown how the design of ES is an essential aspect of learning. We have related ES design to the issue of persistent excitation. The Exploratory Schedule represents a weaker criteria than persistent excitation

The robot model parameters were viewed as state variables. They were used to form the augmented state space in which the asymptotic stability of the origin implies both asymptotic learning and asymptotic tracking. A theorem regarding constructive sufficient conditions for asymptotically stable closed loop learning was proved, and examples of learning in 1 and 2 degree of freedom (DOF) manipulators were given.

The simulation results of the ES for a 2 DOF manipulator (section 4.2), showed how in practice, due to noise and disturbances, exact tracking may not occur and small tracking errors may lead to small errors in the parameter identification. The comparison of controller performance before and after learning (section 4.3) demonstrated how the tracking error can be greatly reduced after the controller learns the parameters via an Exploratory Schedule.

We have demonstrated a method for the selection of exploratory schedules for rigid robot manipulators employing inverse dynamics controllers with direct parameter adaptation. A natural extension to this work is to generalize the concept to a broader class of systems. Immediate extensions may be to other adaptive control laws e.g. that of [Sadegh-Horowitz].

We have specified the exploratory schedule as a desired trajectory that is to be followed to do learning while the manipulator is not doing other useful tasks. However, a more appealing idea would be to integrate the exploratory schedule into any trajectory that is specified by the path planner. The exploratory schedule would take the form of an exploratory (or probing) signal into the desired trajectory to cause asymptotic learning. The injection of exploratory schedules in this manner would be similar to the concept of dual control (see Ref. [Stengel]) in that the probing signal ES is combined with the so called cautious signal.

References

Asada, H., and Slotine, J.J.E . *Robot Analysis and Control.* New York: John Wiley & Sons, 1986.

Craig, J.J., *Adaptive Control of Mechanical Manipulators.* Addison-Wesley,1988.

Fu, K.S., Gonzales, R.C., and Lee, C.S.G., *Robotics: Control, Sensing, Vision, and Intelligence.* McGraw-Hill, 1987

Hsia, T.C., "Adaptive control of robot manipulators a review," *IEEE Int. Conf. Robotics and Automation*, San Francisco, 1986.

Li, W., and Slotine, J.J.E., "Indirect adaptive robot control," *IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA, 1988.

Narendra, K.S., and Annaswamy, A.M., *Stable Adaptive Systems*. New Jersey: Prentice-Hall, 1989.

Paul, R.P., *Robot Manipulators: Mathematics, Programming, and Control*. Massachusetts: The MIT Press, 1981.

Sadegh, N., Horowitz, R., "Stability analysis of an adaptive controller for robotic manipulators," *IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, 1987.

Slotine, J.J.E., and Li, W., "Adaptive manipulator control-a case study," *IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, 1987.

Stengel, R.F., Stochastic Optimal Control: Theory and Operation. New York: John Wiley & Sons, 1986.

[6]

Improving the Solution of the Inverse Kinematic Problem in Robotics Using Neural Networks

Allon Guez and Ziauddin Ahmad

A method for solving the inverse kinematic problem (IKP) for robotic manipulators is presented in this article. The IKP finds the joint angle values of the robot that will place the robot's end-effector in a desired position and orientation. There is more than one solution to this problem. The problem-solving method presented is a hybrid approach in which a neural network is trained to provide an approximate solution. The approximate solution is then used as an initial guess to an iterative procedure that provides a final solution within some specified tolerance in the robot's work space. The proposed hybrid method achieves about a twofold increase in computational efficiency for an industrial manipulator with more consistency in the time required to obtain the solution to the robotic manipulator. This article was accepted for publication August 1989.

A robot consists of a serial chain of rigid links connected to each other by actuated joints. In robot kinematics, the motion of the robot links is studied in terms of the robot's mechanical geometry and structure. The forward kinematic problem is that of finding the end-effector, or gripper, position-and-orientation m-dimensional vector in a Cartesian space, X, as a function of the n-dimensional joint vector, q. This can be expressed as

$$X = f(q) \quad (1)$$

where f is a nonlinear, continuous, and differential function determined by the robot's geometry.

The inverse kinematic problem is defined as: given the vector X, find the vector q. That is to say, solve equation 1 for q,

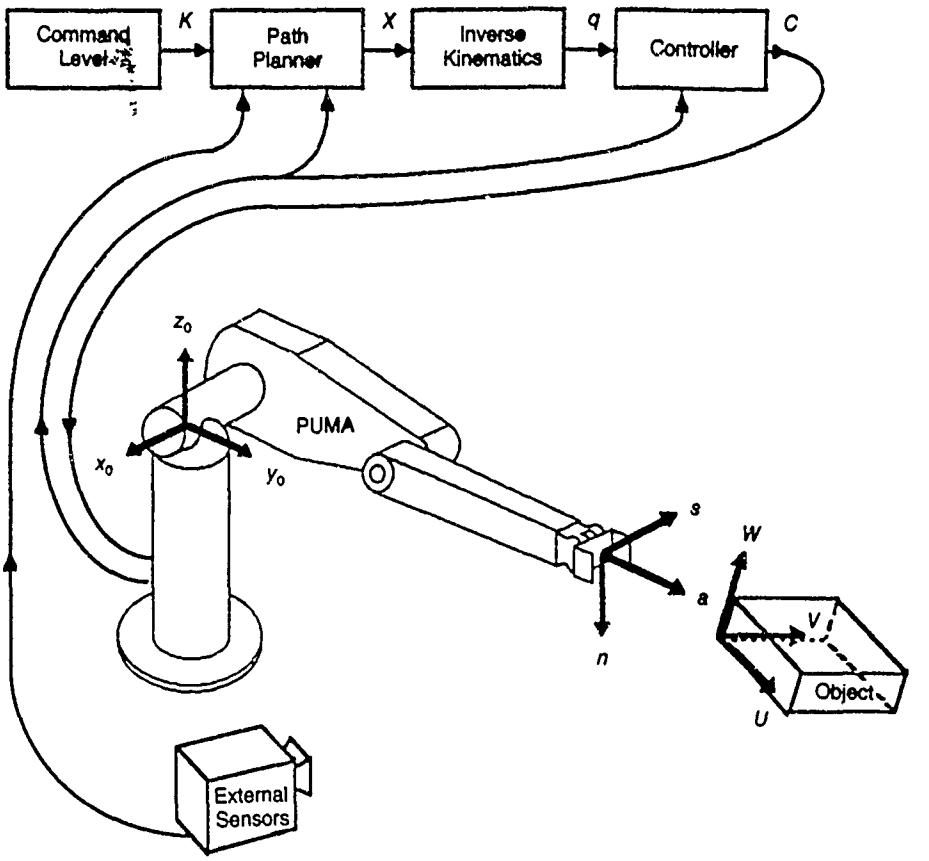
$$q = f^{-1}(X). \quad (2)$$

The IKP usually does not have a unique solution. (The IKP is also referred to as the arm solution.) The significance of the inverse kinematic problem is depicted in Exhibit 1. This exhibit shows the command level that generates a task K, which is passed to the path planner. The path planner generates a feasible path in the work space (i.e., the Cartesian space) to accomplish the task. In order to follow this path, the joints must be actuated, so the joint values that will make the end-effector follow the Cartesian path are needed. The inverse kinematics block in the diagram makes the desired transformation and provides the controller with the joint values for q to be followed. The controller then generates the control signal, C, that drives the joints to follow the desired path. Additional information concerning the current state of the robot comes from the sensors.

A robot's task is usually specified in a Cartesian space, so the inverse kinematics solutions are the ones most frequently used. The solutions can be split into two categories: closed form and iterative. Closed-form solutions include matrix algebraic or geometric methods.[1] In general, closed-form solutions are impossible because $f(q)$ in equation 1 is highly nonlinear. However, because of the importance of a fast solution to the IKP, robots are frequently designed to have simplified inverse kinematics (e.g., the last three joint axes intersect in a point). Thus, many existing industrial manipulators possess closed-form solutions to the IKP. Of course, whenever closed-form solu-

Allon Guez and Ziauddin Ahmad are both professors of electrical engineering and computer science at Drexel University, Philadelphia PA.

Exhibit 1. *Block Diagram of Inverse Kinematic Control*



tions are available, they can be rapidly and precisely implemented; however, such solutions are customized to the robot and require accurate knowledge of all geometric parameters, such as link lengths and joint axes geometry.

Iterative solutions are used for manipulators that do not have a closed-form solution, which include nonredundant and redundant manipulators. These methods start with an initial guess for the solution and use Newton-Raphson or gradient-descent algorithms to find the correction vector to be added to make the error small.[2] These methods are simple to implement and are independent of the type of manipulator. However, they require knowledge of forward kinematics, they are computationally intensive, they may have slow convergence, and their performance depends on the quality of the initial guess.

In many cases, there are an infinite number of configurations for q that place the end-effector in the prespecified position and orientation (i.e., the IKP has an infinite number of solutions). In these cases, additional constraints on the allowed configurations or performance functions are introduced to reduce the number of legitimate joint configurations or to single out a unique, preferable configuration.[3] In that way, the IKP is modified to have a finite number of solutions. Motion heuristics are then used during path planning to select a unique configuration.

Conventional methods for solving the IKP have the disadvantage of being complex and restricted to only a class of nonredundant robots or being slow in converging to a solution. These problems are addressed in this article and a combination of a neural network with a conventional method is suggested as

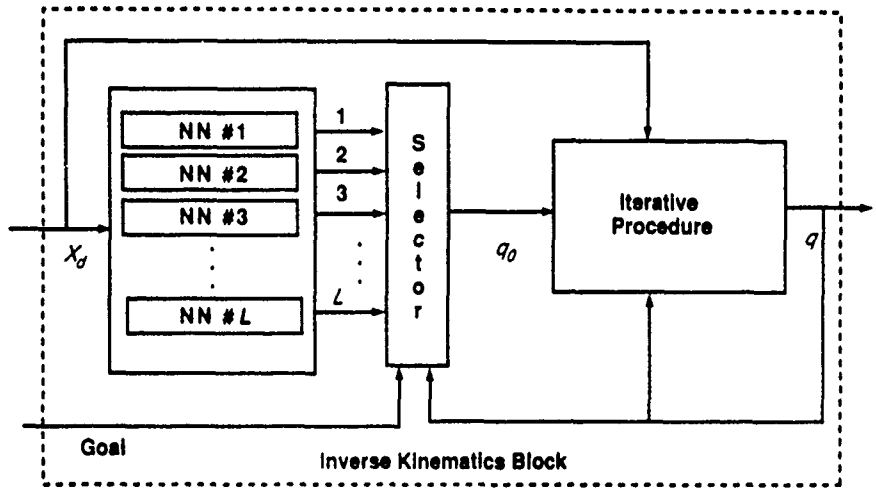
a workable alternative to conventional methods alone.

The hybrid method

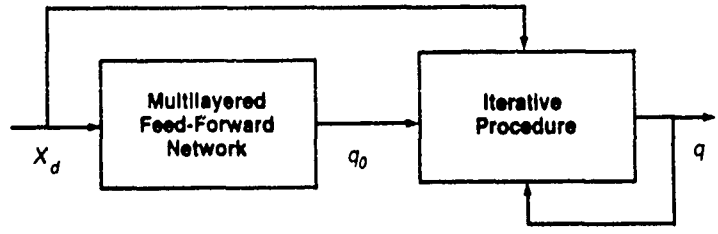
This approach to the solution of IKP of robots is an iterative procedure for which the initial guess is given by trained neural networks. Exhibit 2 shows the schematics of the proposed method. Exhibit 2a shows an array of neural networks. Each of the networks, when provided with the desired position and orientation, X_d, gives the type of solution (in terms of configuration) that nearly matches its training data. The selector block, shown in the exhibit, chooses one of the solutions and passes it to the iterative procedure as the initial guess, q_0. The iterative procedure picks up the initial guess and iterates until an acceptable solution is obtained.

Implementing the proposed method starts with the robotic manipulator for which the IKP is to be solved and an untrained neural network of suitable size. Data pairs of the end-effector position and orientation and the corresponding joint values are generated. This data is used for training the neural network, with the position and orientation vectors as the input and the corresponding joint values as the desired output. After the training is completed, the trained neural network is coupled with the iterative method, as shown in Exhibit 2b, for system operation. During system operation, the desired position and orientation of the end-effector are provided to the neural network. The neural network gives the approximate solution based on the learned connection weights. This approximate solution is taken as the initial guess by the iterative method to yield the final solution.

Exhibit 2. *Diagram of the Proposed Method*



a. The selector block chooses an initial guess from solutions provided by the multilayered feed-forward neural network array



b. Trained network is coupled with the iterative procedure

The multilayered feed-forward network used consists of the supervised back-propagation algorithm used to train the output nodes that correspond to the joint values, given by

$$q = [q_1\ q_2 \cdots q_n]^T \quad (3)$$

related to the desired position and orientation of the end-effector, X_d, which is provided to the input-layer nodes. The position specification consists of the displacement of the origin of the coordinate system (x, y, z) attached to the end-effector with respect to the base frame. The orientation specification consists of the Eulerian angles (i.e., ϕ,θ,ψ).[4]

$$X_d = [\phi\ \theta\ \psi\ x\ y\ z]^T \quad (4)$$

where ϕ is the rotation around the OZ axis, θ is the rotation around the OU axis, and ψ is the rotation around the OW axis in the given sequence and in the end-effector's frame of reference. The data pair can be generated using only the forward kinematics of the robot, as discussed in the following paragraph.

A problem with this arbitrary set of data points is that there is more than one solution to the inverse kinematic problem. By nature, feed-forward neural networks can learn only a single transformation, which requires the selection of one type of solution as the training data set. Each type of solution is bounded by singular regions (defined in the inset), which correspond to the rank deficiency of the Jacobian (also defined in the inset) of the manipulator.[5] Thus the data set corresponding to one type of solution can be generated by starting with any configuration and joint values, storing the relevant data while incrementing in joint space, and at each step ensuring that the Jacobian is not near rank deficiency. The rank deficiency of the Jacobian can be easily verified by taking the determinant of the Jacobian (J), or that of (JJ^T) in case of an under- or over-determined manipulator and comparing the result with zero. However, it is desirable to obtain data that is sufficiently rich in the configuration space. Two methods that may be used to achieve this are:

- Following the isocline of a function of the joint variables
- Taking all the combinations by varying one joint at a time.[6]

Another problem encountered is in the range of the joint angles. For a particular type of solution, the entire joint range for each joint may not be covered; thus only disjoint portions of the maximum joint range can be used. This problem of disjoint regions can be handled in two ways:

- Modifying the network architecture.
- Making the regions continuous.[7]

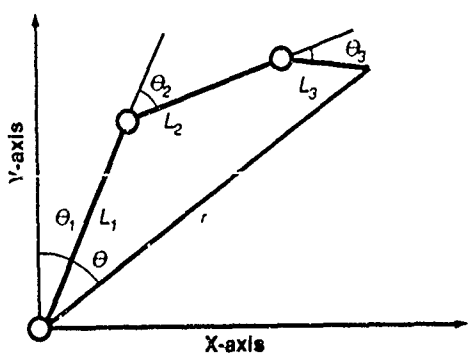
Case studies

The back-propagation (BP) algorithm simulating a three-layer perceptron was employed to tackle the problems described in the following case studies. Continuous input and output were assumed. The nodes assumed symmetric sigmoidal nonlinearity.[8] The learning rate and the momentum term assumed the values of 0.1 and 0.4, respectively. Also, the desired outputs were normalized between -0.9 and $+0.9$. Training was terminated when the error rates were not improving.

Example 1: A three-degrees-of-freedom planar manipulator

A three-degrees-of-freedom (DOF) planar manipulator is shown in Exhibit 3. Before describing the simulations and results, redundance as specific to a 3-DOF planar manipulator must be defined. In a plane, two DOFs are sufficient for the purpose of positioning, therefore a 3-DOF planar manipulator has one redundant DOF. (A detailed definition is given in the Definitions section.) This extra DOF introduces a problem of an infinite number of joint configurations that result in the same end-effector position. The problem can be resolved by adding an additional independent constraint to be fulfilled in addition to finding the best position for the end-effector.

Exhibit 3. *A Three-Degrees-of-Freedom Planar Manipulator*



Definitions

Denavit-Hartenberg transformation The Denavit-Hartenberg matrix, shown in Exhibit A, is a representation of the general displacement of one frame with respect to the other. It consists of the rotation matrix and the translation vector. In addition to these, the matrix also contains the information concerning perspective transformation and a scaling factor.

Jacobian The Jacobian (of a manipulator) specifies a mapping from velocities in joint space to velocities in Cartesian space.[11] Therefore, the Jacobian $J(x,q)$ of a manipulator is a matrix as defined below:

$$J(x,q) = [\partial x_i/\partial q_j] \quad ; \; i = 1...m; \; j = 1...n \qquad (A)$$

where m equals the number of independent Cartesian variables, and n equals the number of joints. The columns of the Jacobian matrix show the change in the position and orientation of the end-effector due to the change of the individual joints.

Singularities A singularity corresponds to a configuration of the manipulator for which the end-effector motion is constrained in some directions. Also, at a singularity, the Jacobian is rank deficient.

$$|J| = 0 \qquad (B)$$

There are two types of singularities. One corresponds to the boundary of the work space where the singularity means that the manipulator cannot cross this boundary. The other singularity corresponds to the situation where the axes of two or more similar joints are aligned, and a motion in one of these joints can be canceled by a counter-motion in any one of the other aligned joints.

Degrees of Freedom (DOF) The number of degrees of freedom that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism.[12] For a robot, which is an open kinematic chain, the degrees of freedom equal the number of joints: each joint corresponds to one variable.

Redundant DOF The number of DOF in excess of the minimum number of DOF required to arbitrarily position and orient an object in the Cartesian space is called the redundant DOF.

An Iterative Procedure The schematics of the iterative method is shown in Exhibit B. This procedure, commonly called the Newton-Raphson method, uses the linear approximation

$$q_{k+1} = q_k + dq_k \qquad (C)$$

where q_k is the joint variable vector at the

Exhibit A. *A Denavit-Hartenberg Matrix*

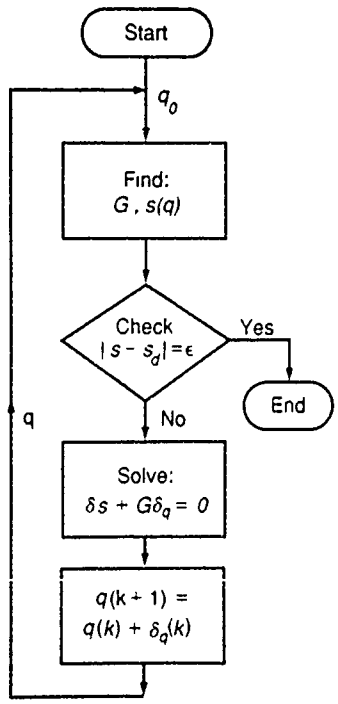
$$\left[\begin{array}{ccc|c} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

a. Matrix representation of the displacement of one frame

Rotation Matrix	Translation Vector
Perspective Transformation	Scaling Factor

b. Submatrices and Factors

Exhibit B. *Flow Diagram for the Iterative Procedure to Find the Joint Values*



(*cont*)

Definitions (Cont)

kth iteration and dq_k is the solution to the linear system

$$s(q_k) + G_k\, dq_k = 0, \qquad \text{(D)}$$

that is,

$$dq_k = -G_k^{-1}\, s(q_k) \qquad \text{(E)}$$

Therefore Equation C becomes

$$q_{k+1} = q_k - G_k^{-1}\, s(q_k), \qquad \text{(F)}$$

where G_k is found numerically by the central difference formula

$$G_{ij} = (s_i^+ - s_i^-)/2\, dq_j \;\; ; \;\; i,j = 1..n \qquad \text{(G)}$$

Here m residuals result because of the forward kinematics in the hand Cartesian space ($m = 6$ for a general manipulator) (i.e., $(f(q) - X_d)$, where $f(q)$ is the forward kinematics and X_d is the desired end-effector position and orientation vector). The n-m residuals correspond to the constraints to be optimized and correspond to

$$Zh = 0 \qquad \text{(H)}$$

where

$$Z = [J_{n-m}\, J_m^{-1} : -I_{n-m}] \qquad \text{(I)}$$

where

- J_m = a square matrix corresponding to any m columns of the Jacobian matrix.
- J_{n-m} = transpose of the matrix consisting of columns not included in J_m matrix.
- I_{n-m} = an identity matrix of order $n - m$

and

$$h_i = \partial H/\partial q_i \;\; ; \;\; i = 1...n \qquad \text{(J)}$$

where H defines the criteria function to be optimized.

Here it may be noted that

$$s(q) = [(f(q) - X_d)^T : (Zh)^T]^T$$

and

$$G(q) = [J^T(q) : [d(Zh)/d(q)]^T]^T,$$

where $J(q)$ is the Jacobian of the manipulator and d is the differential operator.

The performance criterion optimized for the 3-DOF planar manipulator was the manipulability index, defined as

$$H = \sqrt{|JJ^T|} \qquad (5)$$

where J is the Jacobian of the manipulator.

The forward kinematics of this manipulator is given by

$$f_1(\theta_1,\theta_2) = X = L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2) - L_3 \sin(\theta_1 + \theta_2 + \theta_3) = 0 \qquad (6)$$

$$f_2(\theta_1,\theta_2) = Y = L_1 \cos(\theta_1) - L_2 \cos(\theta_1 + \theta_2) - L_3 \cos(\theta_1 + \theta_2 + \theta_3) = 0 \qquad (7)$$

where L_1, L_2, and L_3 are the link lengths. By the iterative procedure (described in the inset) the m (= 2) residuals correspond to the equations

$$f_1(\theta_1,\theta_2) - X_d = 0 \qquad (8)$$

$$f_2(\theta_1,\theta_2) - Y_d = 0 \qquad (9)$$

and the n − m (= 1) residual corresponds to

$$\begin{aligned} Zh &= 0 \qquad (10)\\ &= 2u^2v^3(s_{23}s_{33} - c_{23}s_3^2)\\ &\quad + uv^3(2s_{23}s_{233} - 3s_{2233}s_3)\\ &\quad + 2u^3v^2s_2s_{33} + u^2v^2(c_{33} - c_{22})\\ &\quad + 2uv^2(s_2s_{2233} - c_2s_{23}^2) - u^3vs_{22}s_3\\ &\quad - 2u^2v^2s_2s_3 \end{aligned}$$

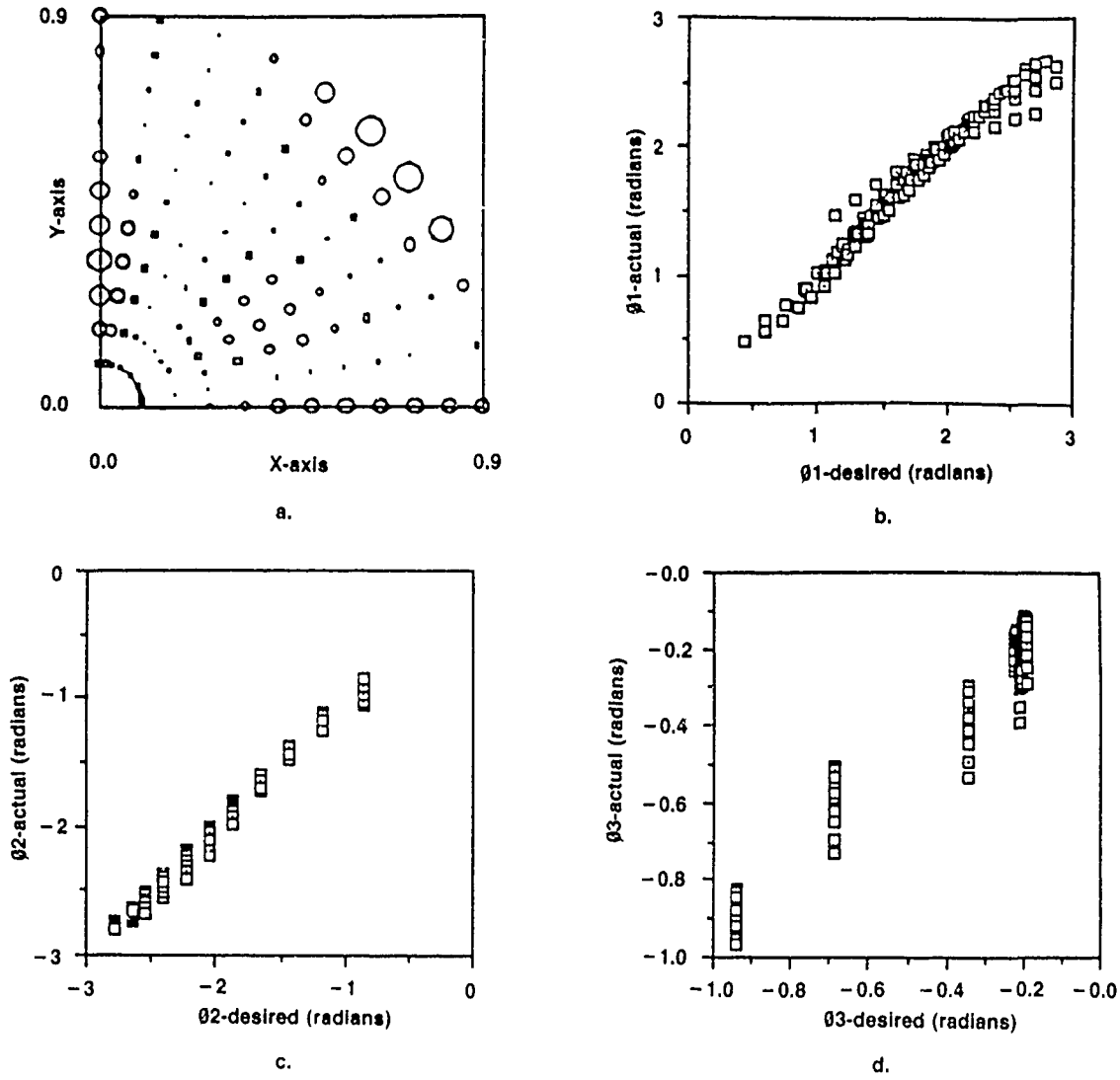
where

$$\begin{aligned} s_{ij} &= \sin(\theta_i + \theta_j)\\ s_{ijk} &= \sin(\theta_i + \theta_j + \theta_k)\\ s_{ijkl} &= \sin(\theta_i + \theta_j + \theta_k + \theta_l)\\ c_{ij} &= \cos(\theta_i + \theta_j) \end{aligned}$$

The network for this case consists of two input nodes and three output nodes and two hidden layers, each containing 10 nodes. The network was trained on a data set giving the X,Y coordinates for the end-effector and the three joint angles that optimized the manipulability criterion. This data set contained 121 input/output pairs that were obtained by dividing the work space into 10 angular and 10 radial regions and taking the data points at the corners of the intersection of these regions. The program that generated this data set employed the Newton-Raphson method for solving the equations, and the Jacobian of the manipulator was found by the central difference formula.[9] In addition, the single inverse was ascertained by taking the initial condition as the current configuration for the arm solution to a subsequent nearby end-effector position. The link lengths were 0.5m, 0.4m, and 0.1m for links 1, 2, and 3 respectively, so that the maximum reach was 1.0m.

The result of this training is shown in Exhibit 4. Exhibit 4a shows that the error begins to increase as the manipulator moves nearer to the singularity. In this case, singularity occurs

Exhibit 4. *Results of a Neural Network Trained to Give a Manipulability Index Optimized Solution for a 3-DOF Planar Manipulator*



Note:
a shows the error in end-effector positioning at the center of the circle and the magnitude is depicted by the diameter of the circles. b through d show the relationship between the desired joint angles and those given by the neural network.

at $\theta_2 = \theta_3 = 0$ radians and corresponds to a circle of radius of 1m around the point of origin in the Cartesian space. Exhibits 4b through 4d show the desired joint values and those given by the neural network. The neural network learned the solution fairly well.

When the neural network solution was coupled with Newton-Raphson iterative procedure, it resulted in nearly a five-fold increase in computational efficiency. Exhibit 5 gives a detailed description of the number of operations required for the iterative method and the solution given by the neural network. It was found that, on the average, the iterative method alone with random initial guess required 12 iterations to give the solution, whereas the iterative method that used the solution given by the neural network as the initial guess required, on the average, only two iterations. This information is taken into account in the last two

Exhibit 5. *Number of Operations Required for the 3-DOF Manipulator Solution*

Methods / Operations	Iterative Method		Neural Net Solution	Random Initial Guess (A + 12B)	MFN Solution as Initial Guess (A + 2B + C)
	Constant (a)	One Iteration (b)			
	A	B	C	D	E
Assignment Statements	27	419	434	5055	1299
Structured Statements	—	61	193	732	315
Multiplications	15	885	219	10275	1944
Additions	16	787	242	9460	1832
Function Calls	13	34	30	421	111
Total	71	2156	1118	25943	5491

Note:
MFN Multilayered feed-forward network

columns of Exhibit 5 so that the corresponding numbers in the last row give an indication of the difference in time required for the final solution using both the conventional method and the neural network augmented iterative method.

Example 2: PUMA 560 robotic arm positioning

The PUMA 560 consists of 6 revolving joints (This is a 6-DOF robot.) The first three joints, which correspond to the waist, shoulder, and elbow motions, contribute mainly to positioning the end-effector. The last three joints correspond to the wrist motion and contribute mainly to orienting the end-effector. The PUMA 560 parameters were taken from *Robotics, Control, Sensing, Vision, and Intelligence.*[10] These parameters are given in Exhibit 6. This manipulator was chosen for its ease-of-use. Generation of data and verification of results are simplified because the manipulator has a closed-form solution. PUMA has eight solutions for a given position and orientation, which are signified by right shoulder or left shoulder, above elbow or below elbow, and up wrist or down wrist.

The training data used in this case corresponds to a left arm, below elbow and up wrist configuration. In the simulations, the joint limits used for the sixth joint were -180 degrees and $+180$ degrees, instead of -260 degrees and $+260$ degrees.

As mentioned in the previous section, there may be more than one disjoint region in which the solution for a particular configuration lies. In the case of PUMA 560 with the considered configuration, the values for joint number 6 lie in two disjoint regions. One corresponds to the region between $-\pi/2$ and $-\pi$, and the other corresponds to the region between $\pi/2$ and π. These need to be trained separately or

Exhibit 6. *PUMA Robot Arm Link Coordinate Parameters*

Joint i	Variable	θ_i (degrees)	α_i (degrees)	a_i (mm)	d_i (mm)	Lower Limit (degrees)	Upper Limit (degrees)
1	θ_1	90	−90	0	0	−160	160
2	θ_2	0	0	431 8	149 09	−225	45
3	θ_3	90	90	−20 32	0	−45	225
4	θ_4	0	−90	0	433.07	−110	170
5	θ_5	0	90	0	0	−100	100
6	θ_6	0	0	0	56.25	−266	266

else combined by transforming these two regions by π and $-\pi$, respectively. In this case, they were combined so that the two regions lie in the first and the fourth quadrant instead of the third and second quadrants and then trained with a single network.

The network in this case consisted of six input nodes, one output node each for the six joints and two hidden layers for each joint consisting of 32 nodes in the first layer and eight nodes in the second layer. The standard back-propagation algorithm was used with the learning rate of 0.1 and the momentum term of 0.2. The training was done for 138 presentations of the 800-sample data set.

The average error and the standard deviation of the error in the solution given by the neural network for each joint taken over 100 samples is charted in Exhibit 7. This exhibit shows that the solution given by neural networks is more scattered for joints 4 to 6 as compared with joints 1 to 3.

To compare the results of a random initial guess to the initial guess provided by the neural network, the initial guess was presented as the actual solution uniformly perturbed randomly between a fraction of joint limits to the iterative method alone for 100 samples that corresponded to each fraction. The average number of iterations for the 100 samples presented for each fraction of perturbation of this informed Newton-Raphson method (INRM) is plotted in Exhibit 8. For these same samples, requiring the positioning and orientating of the end-effector, the guess given by the neural network was also presented to the iterative method as the initial guess. The corresponding average of the 100 samples for the solution to the initial guess provided by the neural network is also included in this exhibit. The average number of iterations for the neural network method corresponds to less than 20% perturbation with respect to the joint limits around the actual solution for the INRM. In

Exhibit 7. *Statistics of the Neural Network's Solution*

Joint No	Error in Degrees: Average	Error in Degrees: Standard Deviation
1	4 06	13 428
2	−0 16	30 492
3	5 64	11 52
4	3.66	61 236
5	−3 70	24 912
6	−6 02	36 828

Exhibit 8. *Comparison of the Neural Network Hybrid Method with the Informed Newton-Raphson Method*

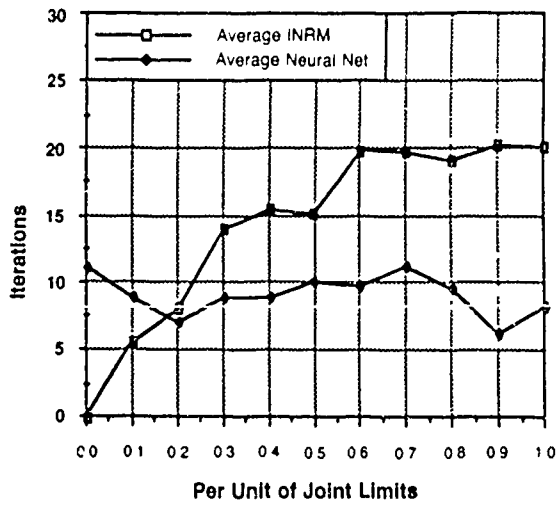


Exhibit 9. *Comparison of the Neural Hybrid Method with the Fixed Estimate Newton-Raphson Method, for 100 Samples*

Method	No of Iterations	
	Average	Standard Deviation
Proposed	9.96	12.95
Newton-Raphson	21.09	18.90

simulations, equation D in the Definitions section was solved by the Gaussian elimination method and partial pivoting. The maximum number of iterations allowed for the iterative method was 100. The iterative method was successfully terminated when the norm of the difference between the desired and actual end-effector position and orientation was less than 1.0E-4.

Next, the neural hybrid method was evaluated by giving a fixed estimate to the iterative procedure. This fixed estimate was taken as: $\theta_1 = 0, \theta_2 = 0, \theta_3 = \pi/4, \theta_4 = 0, \theta_5 = \pi/4$ and $\theta_6 = 0$, which is a configuration corresponding to the one on which the neural network was trained. The average and standard deviation for the number of iterations for the proposed method and the fixed estimator, in a run of 100 data points, are given in Exhibit 9.

The neural hybrid method achieves more than a twofold efficiency in computing with more consistency. Moreover, the time taken by the neural network equals two time units of the iterative procedure, which is less than 10% of the time required to get the solution by the fixed estimator method.

Exhibit 10 shows the comparison plots for two cases indicating the sequence of Newton-Raphson iterations for the fixed initial guess, and the guess given by the neural network. Here, the error along the ordinate axis corresponds to the norm of the error, defined as:

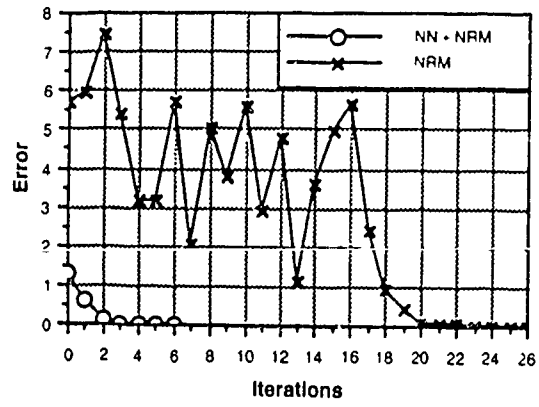
$$\text{Error} = \sqrt{\epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2 + \epsilon_\phi^2 + \epsilon_\theta^2 + \epsilon_\psi^2}$$

where ϵ is the difference between the desired and the actual values, and the subscripts indicate the variable for which this difference is taken. Exhibit 11 shows the initial guesses by the fixed estimator and the neural network corresponding to Exhibits 10a and 10b respectively. Also, these tables show the actual desired solution and the number of iterations taken for each of the two types of initial guesses. The allowed end-effector position and orientation error in this case was 1.0E-7.

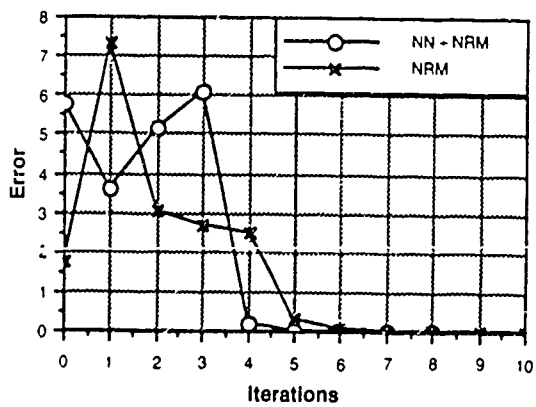
Conclusion

Neural networks were trained to learn the inverse kinematic problem of a manipulability optimized solution of a 3-DOF robot and a 6-DOF robot. It was found that the neural networks were able to learn the transformations fairly well and results using untrained data indicated that the neural networks are able to generalize these transformations.

Exhibit 10. *Sequence of Newton-Raphson Iterations for the Neural Network's Guess and the Fixed Initial Guess*



a Case 1. For 26 Iterations



b Case 2. For 10 Iterations

Exhibit 11. *Initial Guess and Solution Corresponding to Examples Displayed in Exhibit 10*

	Fixed Initial Estimate (26 Iterations)	Initial Guess by NN (6 Iterations)	Solution
Joint-1	0 00	105.50	89 02
Joint-2	0 00	24 74	6 82
Joint-3	45 00	63 46	54 19
Joint-4	0 00	- 7 17	37 72
Joint-5	45 00	56 48	45 21
Joint-6	0 00	170 32	173 46

Case 1

	Fixed Initial Estimate (10 Iterations)	Initial Guess by NN (8 Iterations)	Solution
Joint-1	0 00	- 43.25	- 12 76
Joint-2	0 00	- 40 57	- 51 26
Joint-3	45 00	69 54	82.56
Joint-4	0.00	- 12 37	- 85.25
Joint-5	45 00	- 58 34	- 82 76
Joint-6	0 00	160 80	113 35

Case 2

With the increase in the complexity of the transformation, which occurs with the increase in the number of DOFs, the size of the network needs to be increased. This means that the number of hidden nodes must be increased. But an increase in the number of nodes by one increases the number of weights by the sum of the nodes in the two adjacent layers. Therefore, the training time per iteration cycle of training increases, and so does the overall training time required to train the network. Therefore, for larger manipulators the training time may increase exponentially.

The proposed hybrid method, which takes the solution given by the trained neural network as an initial guess to an iterative procedure (Newton-Raphson in this case), combines the advantages of the neural network and iterative methods; the neural network is independent of the type of manipulator and simple to implement. Only forward kinematics is required for this method and this combination results in a fivefold increase in computational efficiency for a 3-DOF planar manipulator and a twofold increase for the PUMA 560 (6-DOF) robot. The decrease in computational efficiency for the 6-DOF case is due to the trade-off between the size of the network and the training time, and the accuracy of the solution given by the neural network. The training time is decreased with a decrease in the size of the network at the expense of the accuracy of the solution.

Where otherwise the number of iterations may be unacceptable for real-time operation, an initial good guess given by the neural network cuts the number of iterations to only a very few, which allows a better operation of the manipulator. In such cases, minimal processing is required within each control cycle, and real-time control performance improves. ▲

Notes

1 K S Fu, R.C Gonzalez, and C S G. Lee, *Robotics Control, Sensing, Vision, and Intelligence* (New York McGraw-Hill, 1987) and R Paul, *Robot Manipulators Mathematics, Programming, and Control* (Cambridge MA MIT Press, 1981)

2 B Benhabib, A A Goldenberg, and R G Fenton, "A Solution to the Inverse Kinematics of Redundant Manipulators, *Journal of Robotic Systems* 2, no 4 (1985), pp 373–385

3 D E Whitney, The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators, *Transactions of the ASME Journal of Dynamic System, Measurement and Control* 94 no 4 (1972) pp 303–309, and Tsuneo Yoshikawa, "Manipulability of Robotic Mechanisms," *International Journal of Robotics Research* (Summer 1 pp 3–9

4 D E Rumelhart J L McClelland, and the PDP Research Group *Parallel Distributed Processing Explorations in the Microstructure of Cognition* 1 and 2 (Cambridge MA MIT Press 1986) and Fu, Gonzalez, and Lee

5 E Shamir and Y Yomdin, "Repeatability of Redundant Manipulators Mathematical Solution of the Problem," *IEEE Transactions on Automatic Control* 33 (November 1988) pp 1004–1009

6 Z Ahmad, *Fast Solution to the inverse Kinematic Problem in Robotics Via Multilayered Feedforward Networks* (MS thesis Drexel University June 1989)

7 Ahmad

8 W Scott and B A Huberman, "An Improved Three Layer Back Propagation Algorithm," *IEEE First International Conference on Neural Networks* II June 1987) pp 637–643

9 P H Chang, "A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundance," *IEEE Journal of Robotics and Automation* RA 3 (October 1987) pp 393–403 and Yoshikawa

10 Fu, Gonzalez and Lee

11 J J Craig *Introduction to Robotics Mechanics & Control* Reading MA Addison-Wesley 1985

12 Craig

References

Goldenberg, A A Benhabib B and Fulton R G "A Complete Generalized Solution to the Inverse Kinematics of Robotics," *IEEE Journal of Robotics and Automation* RA-1 (March 1985) pp 14–20

Guez A, and Ahmad Z Solution to the Inverse Kinematics Problem in Robotics by Neural Networks *IEEE International Conference on Neural Networks* II (July 1988) pp 617–624

Jordan, M.I. Supervised Learning and Systems with Excess Degrees of Freedom. *COINS Technical Report 88-27* Cambridge MA. MIT Press, 1988.

Lapedes, A., and Farber, R. "Nonlinear Signal Processing Using Neural Networks Prediction and System Modelling. Internal Report, Los Alamos National Laboratory, July 1987

Manseur, R., and Doty, K.L. "A Fast Algorithm for Inverse Kinematic Analysis of Robot Manipulators *The International Journal of Robotics Research* 7 (June 1988), pp 157–165

Miller, W T . III. 'Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm. *IEEE Journal of Robotics and Automation* RA-3 (April 1987), pp 157–165

Paul, R P . and Zang, H. 'Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on the Homogeneous Transformation Representation " *The International Journal of Robotics Research* 5 (Summer 1986)

Widrow, B . and Stearns, S D *Adaptive Signal Processing* Englewood Cliffs NJ Prentice-Hall, 1985

Journal Intelligent and Robotic Systems 2 307–335, 1989
(1989 *Kluwer Academic Publishers Printed in the Netherlands*

[7]

Neurocontroller Design Via Supervised and Unsupervised Learning

ALLON GUEZ and JOHN SELINSKY
ECE Department Drexel University, Philadelphia, PA 19104, U S A

(Received: 3 May 1989)

Abstract. In this paper we study the role of supervised and unsupervised neural learning schemes in the adaptive control of nonlinear dynamic systems. We suggest and demonstrate that the teacher s knowledge in the supervised learning mode includes a-priori plant sturctural knowledge which may be employed in the design of exploratory schedules during learning that results in an unsupervised learning scheme. We further demonstrate that neurocontrollers may realize both linear and nonlinear control laws that are given explicitly in an automated teacher or implicitly through a human operator and that their robustness may be superior to that of a model based controller Examples of both learning schemes are provided in the adaptive control of robot manipulators and a cart-pole system

Key words Nonlinear control, neurocomputi ig, global linearization, supervised learning, unsupervised learning, robot control, manual tracking.

1. Introduction

The recent revival in neuroengineering research, which started in 1982, has focused mainly in the field of pattern recognition and signal processing. Only a few efforts have dealt with applications to control engineering.

The massive parallelism, natural fault tolerance and implicit programming of neural network computing architectures suggest that they may be good candidates for implementing real-time adaptive controllers for large-scale nonlinear dynamic systems.

Several neural network models and neural learning schemes were applied to system controller design during the last three decades. Widrow and Smith (1964) utilized the ADALINE (Adaptive LInear Neuron) and MADALINE (Many ADALINES) architectures and the Widrow-Hoff (LMS) algorithm to provide a 'bang-bang' (control where the applied force is constant in the + or − direction) type of control through duplication of a known switching surface, for the linearized dynamics of a cart-pole system Barto, Anderson and Sutton (1983) developed the ASE (Adaptive Search Element) to perform unsupervised learning and 'bang-bang' control of a cart-pole system The dynamics were assumed to be totally unknown. Psaltis, Sideris and Yamamura (1987) proposed using the Back Error Propagation algorithm (BEP) (see Rumelhart *et al* (1986)) and 'specialized learning' to reduce the total error of the controlled process in specific regions of interest Kawato, Furukawa and Suzuki

(1987) used the LMS algorithm and a linear combination of specific nonlinear subsystems to perform 'feedback learning' of robot control. Guez, Eilbert and Kam (1988) proposed a 'neural estimator' with a steady-state topology that corresponds to optimal control laws. Guez and Selinsky (1988) applied BEP in the design of trainable adaptive controllers (TACs).

In all of the works cited above, the learning schemes were totally independent from the control task. We believe that some, however small, a-priori knowledge of the plant dynamics is always available. We expect that utilizing the a-priori knowledge in the learning phase will improve the neurocontroller performance.

In this paper, we report preliminary results in neurocontroller design which we found encouraging. In Section 2, we describe how a-priori structural knowledge of robot dynamics may be employed in an unsupervised learning routine for the automated design of a nonlinear robot controller. In Section 3, we employ supervised learning with various teaching modes and teacher models, and compare the neurocontroller robustness to that of a model based controller. Section 4 provides a discussion and conclusion.

2. Unsupervised Learning of Control

In supervised learning, we assume the existence of some teacher that is capable of demonstrating the required control performance. That is, in order to train a neurocontroller to control some dynamic system, we must have a controller already capable of controlling the system (see Psaltis *et al*; 1987; Widrow, 1987; Guez and Selinsky, 1988). This is not a detriment in the case of a human trained controller as it can be used to automate a previously human controlled process. In the case of automated linear and nonlinear teachers, the dynamics of the controlled system must be known a-priori for the design of the teacher. We now present the preliminary results of a neurocontroller learning to control a dynamic system where only a general knowledge of the dynamic's structure is known and no external teacher is available. The system under consideration is a robot manipulator. This problem is of particular interest as the full dynamics of a robot are rarely known. Furthermore, the advent of new direct drive robots requires a controller capable of compensating for all of the nonlinearities of the robot.

2.1 ROBOT DYNAMICS

For a robotic manipulator consisting of an open kinematic chain of rigid links the dynamics can be described by

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

where $q(t)$ = the $n \times 1$ vector of joint linear or angular positions,
$\dot{q}(t)$ = the $n \times 1$ vector of joint linear or angular velocities,
$\ddot{q}(t)$ = the $n \times 1$ vector of joint linear or angular accelerations,
$I(q, t)$ = the $n \times n$ matrix of terms related to inertial forces,
$H(q, \dot{q}, t)$ = the $n \times 1$ vector of terms related to centripetal and coriolis forces,
B = the $n \times n$ diagonal matrix of viscous friction terms,
$G(q, t)$ = the $n \times 1$ vector of terms related to gravitational forces,
$T(t)$ = the $n \times 1$ vector of driving forces or torques, and
n = the number of degress of freedom (DOF) of the manipulator.

More details on the derivation of the dynamics can be found in Paul (1981). The dynamics of the robot are highly nonlinear, coupled, time varying and configuration dependent. Conventional control of robots has relied on PID control of each joint individually. These controllers have been relatively effective because the slow speed and high gear reduction of the robots reduce the nonlinearities introduced by the inertial, centripetal and Coriolis forces. However, for high speed operation and accurate trajectory following, controllers, which compensate for the nonlinearities of the robot's dynamics are required.

2.2. FEEDBACK LINEARIZED AND DECOUPLED CONTROL FOR A MANIPULATOR

In feedback linearization or the computed torque method (see Guez, 1982; Fu, Gonzalez and Li, 1987), nonlinear feedback described by a Feedback Linearizing and Decoupling Transform (FLDT) is used to 'cancel' the nonlinearities of the system and transform it to a linear controllable one as shown in Figure 1. The dynamics of the resulting system appear linear and may be controlled using linear control theory. A major advantage of this type of control is that the poles of the transformed system may be arbitrarily assigned using state feedback.

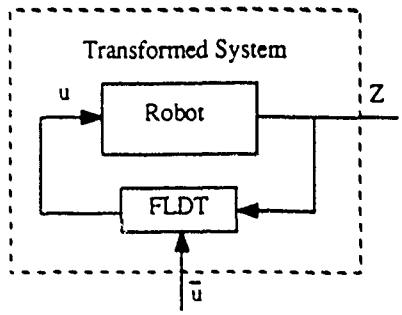


Fig 1 Block diagram of transformed system

The FLDT for an n degree of freedom (DOF) manipulator can be formed as follows. The dynamic equations (1) may be rewritten in the form

$$\ddot{q}(t) = I(q, t)^{-1}[T(t) - W(q, \dot{q}, t) - G(q, t)], \tag{2}$$

where

$$W(q, \dot{q}, t) = H(q, \dot{q}, t) + B\dot{q}(t). \tag{3}$$

The control law is

$$T(t) = W(q, \dot{q}, t) + G(q, t) + I(q, t)(k_p e + k_v \dot{e} + \ddot{q}_d(t)). \tag{4}$$

where $e = q_d(t) - q(t)$.
Here, $q_d(t)$ is the position reference signal and k_p, k_v are appropriately chosen for pole placement of the system, the transformed system is then equivalent to one described by

$$\ddot{q}(t) = k_p e + k_v \dot{e} + \ddot{q}_d(t) \tag{5}$$

which, when k_p and k_v are appropriately chosen, results in an asymptotically stable system.

2.3 UNSUPERVISED LEARNING OF ROBOT MANIPULATOR DYNAMICS USING HIERARCHICAL NN

We present a method for the control of robot manipulators *based only on a-priori knowledge of the number of degrees of freedom and the assumption that the manipulator is an open serial linkage of rigid links.* We employ a hierarchical neural network computing architecture for the automated design of an adaptive controller. A closer examination of the manipulators' dynamics reveals that they can be represented by a weighted linear combination of suitable functions. Note that the functions for a general class of manipulators are a-priori known. We propose that the nonlinear functions may be trained into a series of three-layer feedforward network modules prior to use and then combined online in a fourth layer. Thus, information available prior to use is utilized in one level of the hierarchy, while information that can only be obtained online is learned in another level. The general architecture is shown in Fig. 2(a), (b)

We will use a linear control law with added nonlinear compensation to control the manipulator. During initial purely learning phase and whenever the manipulator is not needed for production, we will generate appropriate exploratory schedules to isolate and identify the nonlinear compensation terms. During the purely learning phase, the desired trajectory for the manipulator is provided by the exploratory schedule planner (ESP). In the learning-production phase, the path planner (PP) produces the trajectory for the production task and the nonlinear compensation terms are learned during suitable parts of the trajectory.

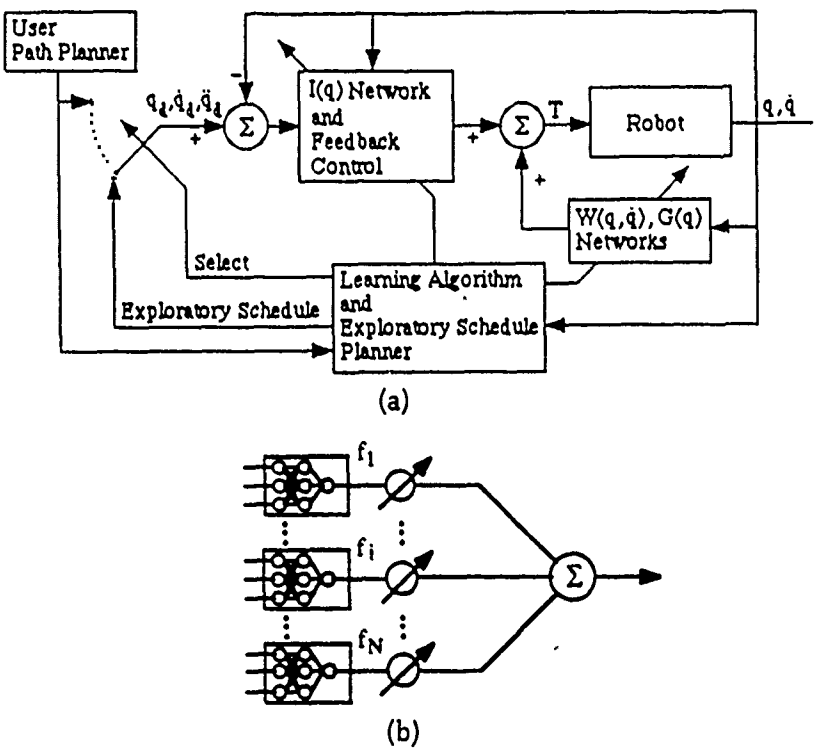


Fig. 2. (a) Unsupervised learning architecture, functional block diagram and (b) neurocontroller architecture.

In either the purely learning or learning-production phase, a simple expert system is used to identify the appropriate conditions for learning the nonlinear compensation terms as follows. Let $G_0(q)$, $W_0(q, \dot{q})$ and $I_0(q)$ be the outputs of the $G(q)$, $W(q, \dot{q})$ and $I(q)$ compensation networks, respectively. After receiving the desired trajectory, the control that is to be applied to the robot is calculated using the feedback c[illegible] from the linear controller and the control from the learned nonlinear compens[illegible] terms. The network first learns the $G(q)$ compensation terms. Then the [illegible] compensation can be learned.

Learning of the $G(q)$ terms is accomplished during positioning control phases of the trajectory. At the steady state of positioning control, $\ddot{q} = \dot{q} = 0$. Then from (1) it can be seen that

$$T = G(q). \tag{6}$$

The applied torque at steady state can be used to learn the $G(q)$ compensation network. Learning of the $W(q, \dot{q})$ compensation terms is performed during the velocity control phases of the trajectory. To isolate the $W(q, \dot{q})$ terms, notice that at $\dot{q} =$ constant, $\ddot{q} = 0$ and

$$T = W(q, \dot{q}) + G(q). \tag{7}$$

But $G(q)$ has already been identified and is available as $G_0(q)$, so that we may isolate $W(q, \dot{q})$ to be used in learning. The inertia related terms $I(q)$ may be continuously evaluated using the a-priori known relationship (see Slotine and Li, 1987)

$$\mathrm{d}I(q, t)/\mathrm{d}t = H_m(q, \dot{q}) + H_m(q, \dot{q})^T, \tag{8}$$

where $H_m(q, \dot{q})$ is an $n \times n$ matrix such that in (1) $H(q, \dot{q}) = H_m(q, \dot{q})\dot{q}$.

After sufficient learning time, the dynamic equations of the manipulator are known. A controller of the form

$$T = I_0(k_p e + k_v \dot{e} + \ddot{q}_d) + W_0(q, \dot{q}) + G_0(q) \tag{9}$$

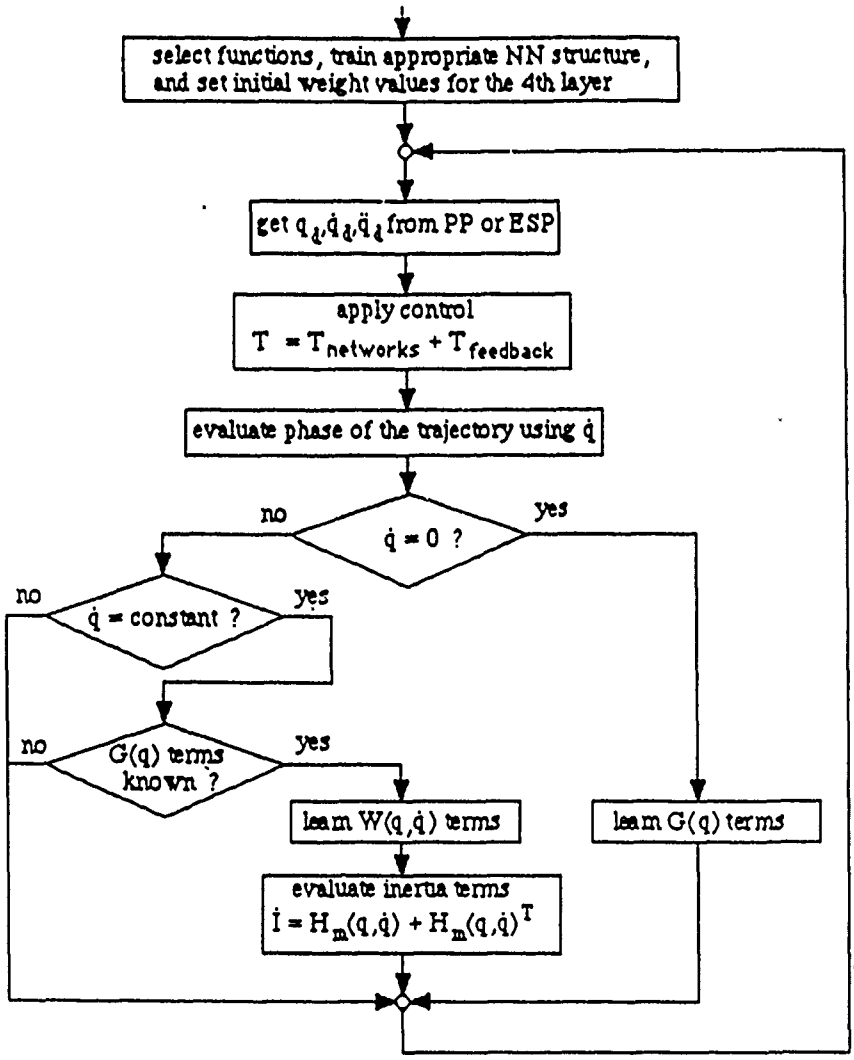


Fig. 3. Flowchart of unsupervised learning algorithm.

may then be employed and the resulting system is equivalent to

$$\ddot{q} = k_p e + k_v \dot{e} + \dot{q}_d. \tag{10}$$

Note that this process does not resuppose the existence of an omnipotent controller that is capable of accurately tracking a trajectory, but rather it starts out with a simple controller capable of performing a limited task and uses learned knowledge of the manipulators dynamics to build a controller capable of controlling the manipulator at high speeds over the entire work space. The unsupervised learning algorithm can be summarized as shown in Figure 3.

2.4. SIMULATION RESULTS

The unsupervised learning algorithm is tested via a simulated two DOF manipulator. The algorithm used for these results differs from the one presented in the discussion in that the external path planner for the learning-production phase was not implemented. Only the exploratory schedule planner for the purely learning phase was used. Also, in this example we assume the presence of a suitable joint acceleration sensor so that we may identify the inertial compensation terms. The two DOF planar manipulator shown in Fig. 4, consists of two links of length d_1, d_2 and mass m_1, m_2. The mass of each link is assumed to be concentrated at the end of the link.

The dynamic equations for this manipulator are

$$T_1 = I_{11}\ddot{\varnothing}_1 + I_{12}\ddot{\varnothing}_2 + H_1(2\dot{\varnothing}_1\dot{\varnothing}_2 + \dot{\varnothing}_2^2) + b_1 + G_1, \tag{11}$$

$$T_2 = I_{12}\ddot{\varnothing}_1 + I_{22}\ddot{\varnothing}_2 - H_1\dot{\varnothing}_1^2 + b_2 + G_2 \tag{12}$$

where

$$I_{11} = (m_1 + m_2)d_1^2 + m_2 d_2^2 + 2m_2 d_1 d_2 \cos(\varnothing_2), \tag{13}$$

$$I_{12} = m_2[d_2^2 + d_1 d_2 \cos(\varnothing_2)], \tag{14}$$

$$I_{22} = m_2 d_2^2, \tag{15}$$

$$H_1 = -m_2 d_1 d_2 \sin(\varnothing_2), \tag{16}$$

$$G_1 = (m_1 + m_2)g d_1 \sin(\varnothing_1) + m_2 g d_2 \sin(\varnothing_1 + \varnothing_2), \tag{17}$$

$$G_2 = m_2 g d_2 \sin(\varnothing_1 + \varnothing_2), \tag{18}$$

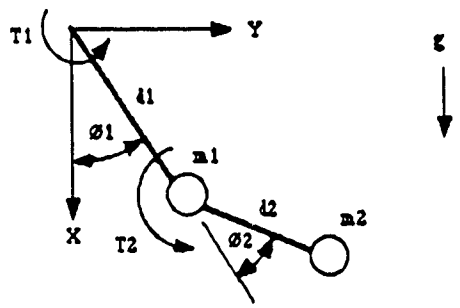


Fig. 4. Two degrees of freedom planar manipulator.

which may be rewritten to provide general forms of linearly separable nonlinear subsystems as

$$T_1 = W_{11}\ddot{\varnothing}_1 + W_{12}\cos(\varnothing_2)\ddot{\varnothing}_1 + W_{13}\ddot{\varnothing}_2 + W_{14}\cos(\varnothing_2)\ddot{\varnothing}_2 + + W_{15}\sin(\varnothing_2)\dot{\varnothing}_1\dot{\varnothing}_2 + W_{16}\sin(\varnothing_2)\dot{\varnothing}_2^2 + W_{17}\ddot{\varnothing}_1 + + W_{18}\sin(\varnothing_2) + W_{19}\sin(\varnothing_1 + \varnothing_2). \quad (19)$$

$$T_2 = W_{21}\ddot{\varnothing}_1 + W_{22}\cos(\varnothing_2)\ddot{\varnothing}_1 + W_{23}\ddot{\varnothing}_2 + W_{24}\sin(\varnothing_2)\dot{\varnothing}_1^2 + + W_{25}\dot{\varnothing}_2 + W_{26}\sin(\varnothing_1 + \varnothing_2). \quad (20)$$

Note that for convenience we are employing the exactly computed values of the functions rather than network modules which have learned the functions. In a similar way to that used by Kawato *et al.* (1987), we employ a network consisting of a linear combiner of the subsystems. However, in contrast to Kawato *et al.* (1987), we use a-priori knowledge of the manipulator's dynamics to excite only specific subsystems of the dynamics. Isolating a subset of the manipulator's dynamics decreases the dimension of the weight space to be searched, which reduces learning time and increases the likelihood of a convergence of the weight vector to the optimum. We have found this to be supported by our simulations.

The optimal values for the weights as calculated from the dynamic equations using the values

$$m_1 = m_2 = 10.0\,\text{kg}, \quad d_1 = d_2 = 1.0\,\text{m}, \quad b_1 = b_2 = 5.0\,\text{kg/s}.$$

are

$$W_{11} = 30.0, \quad W_{12} = 20.0, \quad W_{13} = 10.0, \quad W_{14} = 10.0,$$
$$W_{15} = -20.0, \quad W_{16} = -10.0, \quad W_{17} = 5.0, \quad W_{18} = 196.2,$$
$$W_{19} = 98.1, \quad W_{21} = 10.0, \quad W_{22} = 10.0, \quad W_{23} = 10.0,$$
$$W_{24} = 10.0, \quad W_{25} = 5.0, \quad W_{26} = 98.1.$$

The final weights arrived at by application of the learning algorithm are

$$W_{11} = 29.51, \quad W_{12} = 19.28, \quad W_{13} = 9.79, \quad W_{14} = 9.47,$$
$$W_{15} = -19.28, \quad W_{16} = -9.97, \quad W_{17} = 4.61, \quad W_{18} = 196.19,$$
$$W_{19} = 98.10, \quad W_{21} = 10.06, \quad W_{22} = 10.20, \quad W_{23} = 10.12,$$
$$W_{24} = 10.22, \quad W_{25} = 5.01 \quad W_{26} = 98.0.$$

In total, 240 390 iterations of the LMS algorithm were used for taining. In determining $G(q)$, 300 presentations of a 115 example training set was needed. Identification of $W(q)$ required 171 presentations of a 590 example training set.

Figures 5 to 8 shows the performance of the controller as the identified compensation terms $C(\varnothing, \dot{\varnothing})$ are added to the linear controller

$$T = K_p(\varnothing - \varnothing_d) + K_v(\dot{\varnothing} - \dot{\varnothing}_d) + C(\varnothing, \dot{\varnothing}).$$

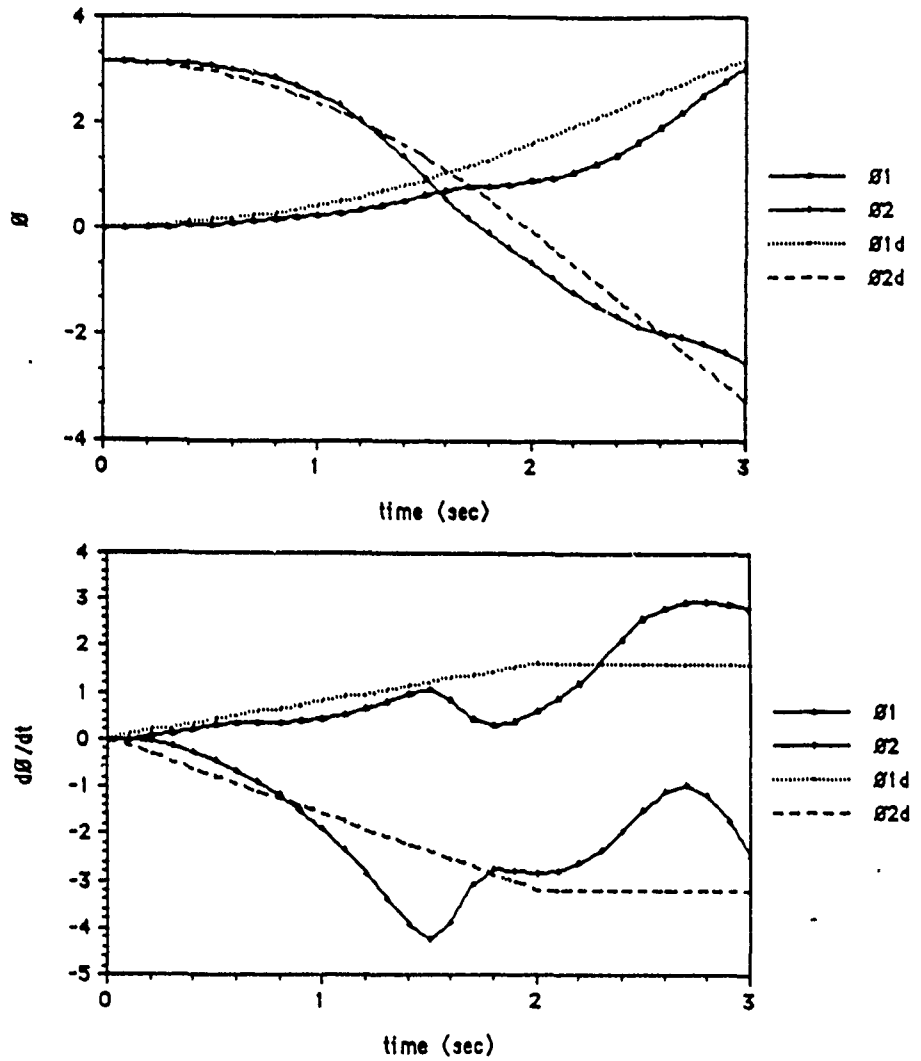


Fig. 5. Controller with $C(\varnothing, \ddot{\varnothing}) = 0$.

where K_p, K_v are 2×2 diagonal matrices with elements $k_{p1} = 400.0$, $k_{p2} = 100.0$ and $k_{v1} = 40.0$, $k_{v2} = 20.0$, respectively. The figures show the desired angular position ($\varnothing_d$), desired angular velocity ($\dot{\varnothing}_d$), actual angular position ($\varnothing$) and actual angular velocity ($\dot{\varnothing}$) of the manipulator's joints as the controller attempts to track the given trajectory. Figure 5 depicts the performance before any learning has occurred. The compensation term $C(\varnothing, \dot{\varnothing})$ is zero. Figure 6 shows the performance of the controller when the gravitational compensation terms have been found and applied. The compensation term is $C(\varnothing, \dot{\varnothing}) = G_0(\varnothing)$, where $G_0(\varnothing)$ is the output of the $G(\varnothing)$ compensation network. Figure 7 shows the performance of the controller when the

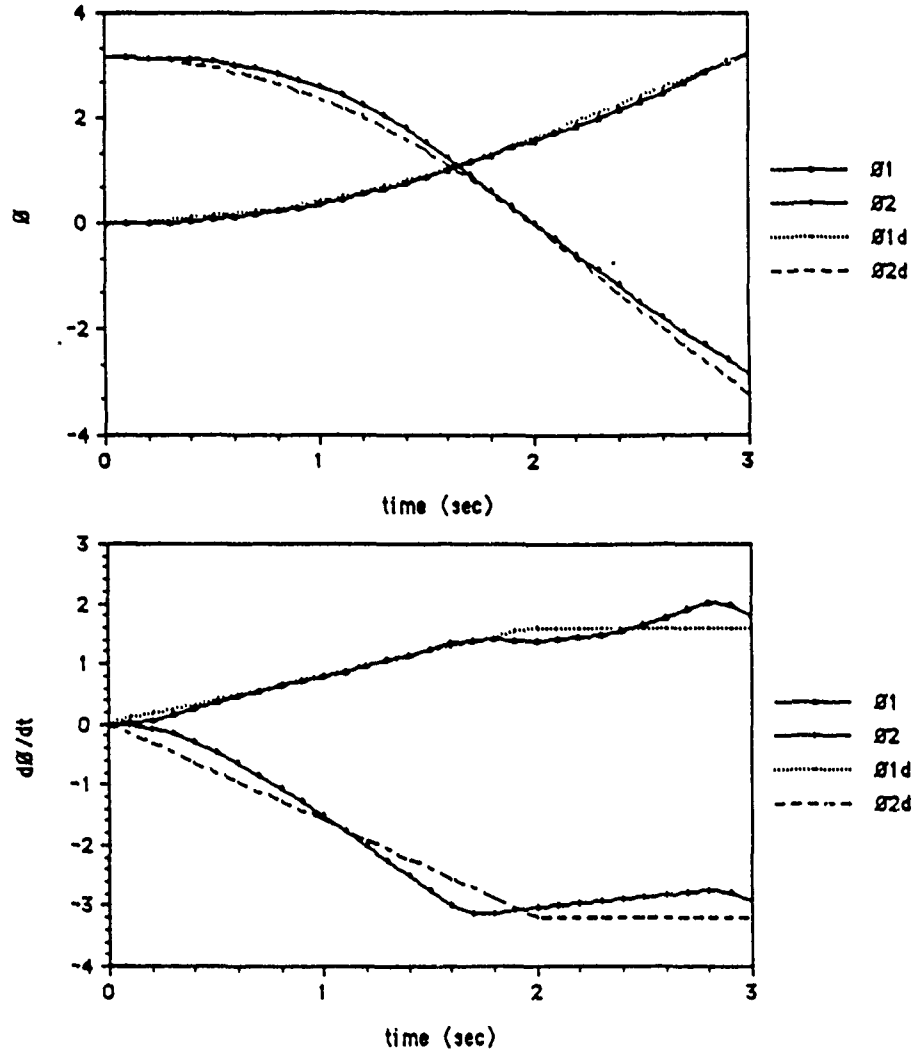


Fig. 6. Controller with $C(\varnothing, \dot{\varnothing}) = G_0(\varnothing)$.

gravitational, centripetal, Coriollis and viscous friction compensation term is $C(\varnothing, \dot{\varnothing}) = W_0(\varnothing, \dot{\varnothing}) + G_0(\varnothing)$, where $W_0(\varnothing, \dot{\varnothing})$ is the output of the $W(\varnothing, \dot{\varnothing})$ compensation network. Figure 8 depicts the performance of the controller when all compensation terms have been found and applied. The controller is now

$$T = I_0(k_p(\varnothing - \varnothing_d) + k_v(\dot{\varnothing} - \dot{\varnothing}_d)) + W_0(\varnothing, \dot{\varnothing}) + G(\varnothing),$$

where $I_0(.)$ is the output of the $I(\ddot{\varnothing})$ compensation network and k_{p1}, k_{v1}, k_{p2} and k_{v2} are as previously defined.

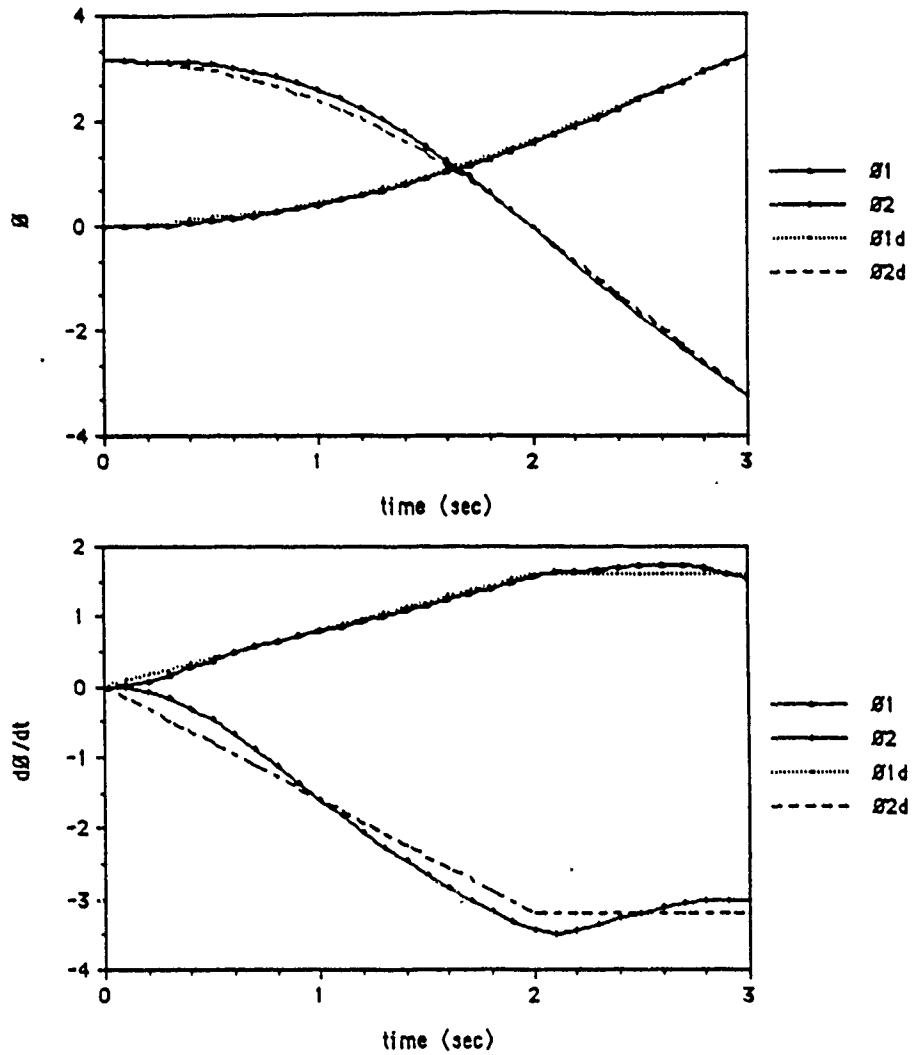


Fig. 7. Controller with $C(\varnothing, \dot{\varnothing}) = W_0(\varnothing, \dot{\varnothing}) + G_0(\varnothing)$.

3. Supervised Neurocontroller (SNC)

We now describe the architecture of a supervised neurocontroller. The SNC architecture, as shown in Figure 9, consists of a teacher, the trainable controller and the controlled process.

The controlled process is, in general, a nonlinear dynamic system with unknown or partially known dynamics. The variables describing the state of the system (i.e. position, velocity, etc) are obtained through suitable sensors and are provided to both the teacher and trainable controller. The teacher describes the desired performance of

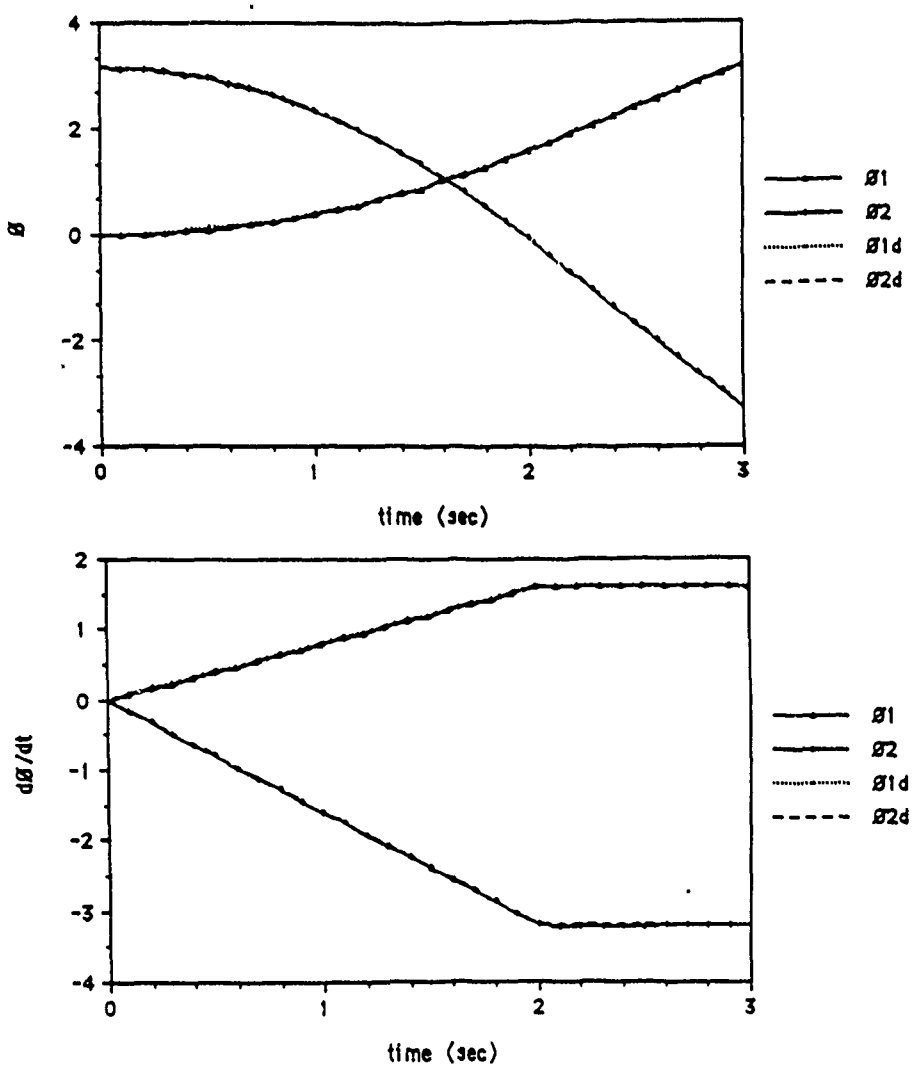


Fig. 8. Controller with full compensation

the SNC by providing examples of the control commands (u) required to successfully control the process. Note that the teacher may be another automated process or a human that is capable of generating the correct control commands. The trainable controller is a neural network architecture suitable for supervised learning. prior to training. as the teacher controls the process, the control signals as well as the state of the process are sampled and stored for use in training the network. During training, the stored process state samples are used as the input to the network and the corresponding stored control signal is used as the desired output of the network. After successful completion of training, the network has discovered the basis for the

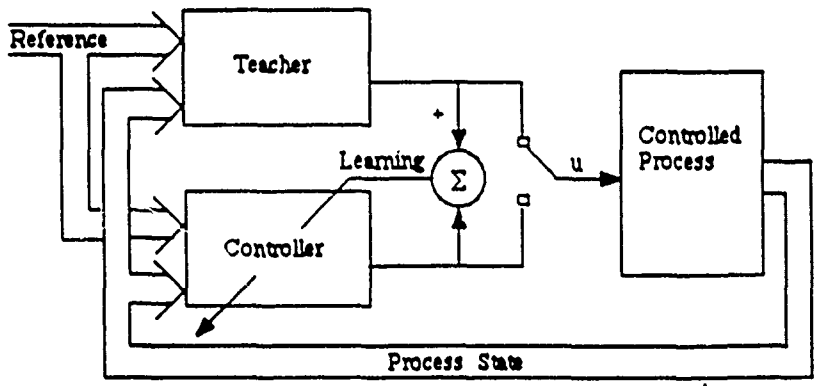


Fig. 9. SNC's architecture.

mapping from the input to the output and will respond with the correct control command. The teacher is then removed and the network controls the process.

3.1. EXAMPLE: CONTROL OF A CART-POLE SYSTEM

We have tested the SNC architecture on a simulated four-dimensional nonlinear dynamic system. The system chosen is the cart-pole system (or broom balancer) which has been the subject of previous research by Widrow and Smith (1964), Widrow (1987) and Barto, Anderson and Sutton (1983). In contrast to the previous work, we provide continuous rather than binary outputs (see Guez and Selinsky (1988) for additional details and experiments with the cart-pole system). The system, as shown in Figure 10, consists of an inverted pendulum of length $2L$ and mass m mounted on a cart of mass M.

The system operates in the vertical plane and is controlled by the force u. Assuming the pole is a narrow rod of uniform composition, no actuator dynamics or friction at the pivot, the dynamics of the system are described by

$$\ddot{\varnothing} = \frac{3}{4L}(g \sin \varnothing - \ddot{X} \cos \varnothing), \tag{21}$$

$$\ddot{X} = \frac{m(L \sin \varnothing \dot{\varnothing}^2 - \frac{3}{8} g \sin 2\varnothing) - f\dot{X} + u}{M + m(1 - \frac{3}{4}\cos^2 \varnothing)}. \tag{22}$$

The state variables of the system are the cart's position, the cart's velocity, the pole's angular position and the pole's angular velocity, respectively

$$Z = [X, \dot{X}, \varnothing, \dot{\varnothing}]^T. \tag{23}$$

We choose a feedforward network with two hidden layers for the controller. The network contains 4 neurons in the input layer, 16 in the first hidden layer, 4 in the second hidden layer and 1 output. The activation of the input layer neurons are linear, all others use a sigmoidal activation function.

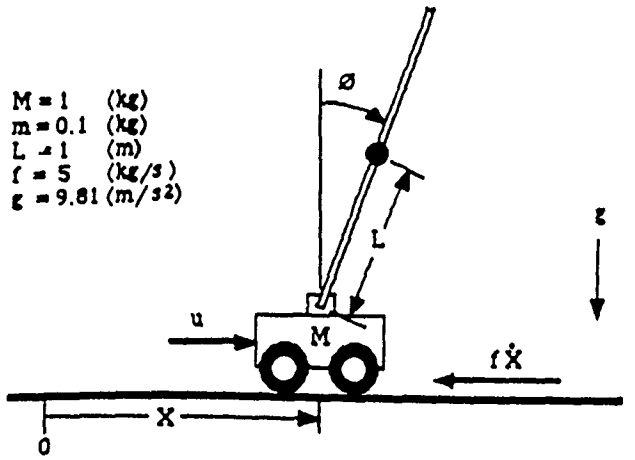


Fig. 10. Cart-pole system.

The state variables of the cart-pole system were used as inputs to the network. The output of the network was scaled to reflect a continuous value of from $+1$ to -1 which is designated to be the normalized control signal u_n. The normalized control signal represents the signed fraction of the total force F_s (where F_s is constant) that is to be applied to the carts as the control signal u, in other words $u = u_n F_s$. We note that the inputs and outputs are continuously valued, not binary. Only the dynamic range of the output has been limited, any real value was allowed for the inputs.

The training data consists of periodic samples of the cart-pole system's state and the normalized control signal from the teacher. The training samples are recorded as the teacher returns the cart-pole system to the origin of the state space from some nonzero initial state. Training data is typically recorded over a 1 to 2 minute period of operation. The network is trained offline using the Back Error Propagation (BEP) algorithm (see Rumelhart, Hinton and Williams (1986)) and the recorded training data.

3.2. SUPERVISED LEARNING WITH A LINEAR TEACHER

An automated teacher consisting of a linear control law is first used to train the SNC. Although a linear combiner could have been used to duplicate the control law, we have included this example to show that a multilayer network can be trained using a wide range of control laws. The cart-pole system (Equations (21) and (22)) were linearized about $\varnothing = 0$ (see Kwakernaak and Sivan, 1972). The linearized dynamics used to formulate the control law are

$$\ddot{X} = \frac{1}{M}(u - f\dot{X}), \tag{24}$$

$$\ddot{\varnothing} = \frac{3}{4L}(g\varnothing - \ddot{X}). \tag{25}$$

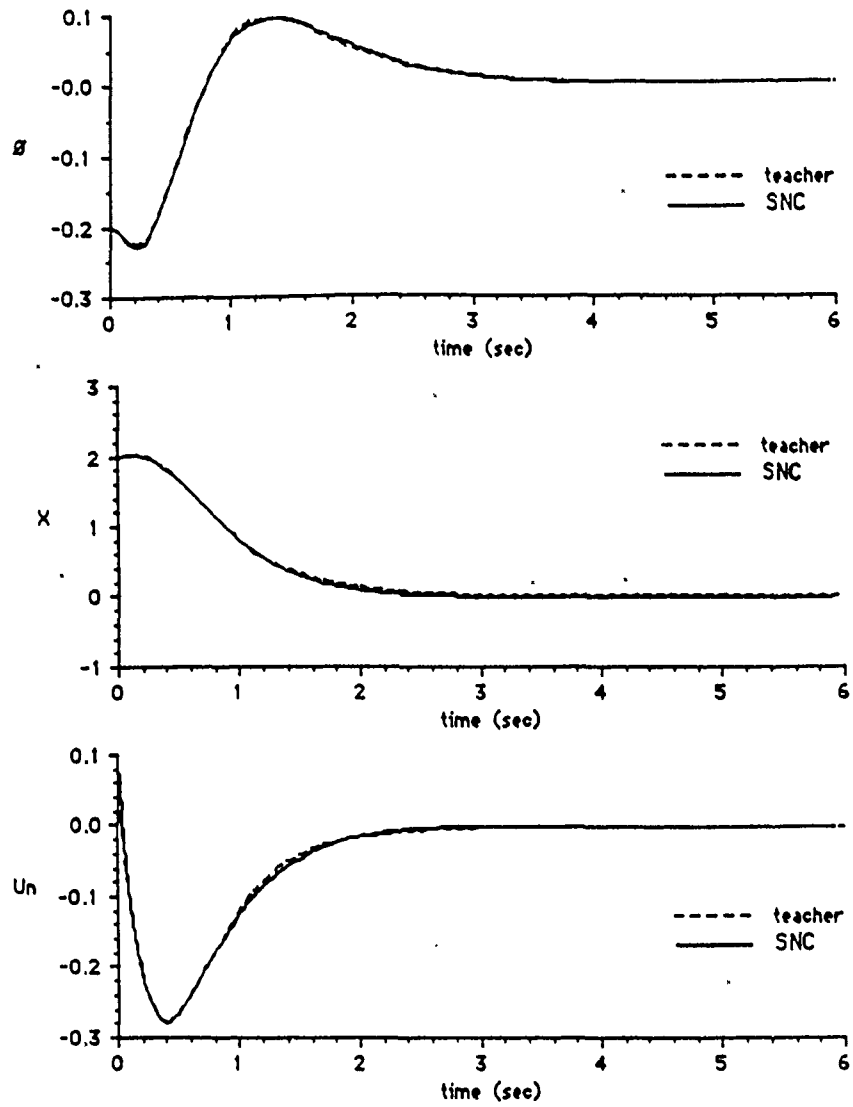


Fig. 11. Comparison of linear teacher and corresponding SNC.

The linear control law that was used as the teacher is

$$u = k_1 X + k_2 \dot{X} + k_3 \varnothing + k_4 \dot{\varnothing}, \tag{26}$$

where $k_1 = 11.01$, $k_2 = 19.68$, $k_3 = 96.49$ and $k_4 = 35.57$. The network was able to learn the linear control law after 20 000 iterations of the BEP algorithm. Figure 11 compares the performance of the linear teacher with the corresponding linear trained SNC (LSNC).

From the initial state $Z = (2, 0, -0.2, 0)$, both the linear teacher and the LSNC follow nearly identical trajectories as they generate stable control to the origin of the

state-control space. Thus we conclude that a multilayer feedforward network can be used to generate linear control in a local neighborhood of the state space origin.

3.3. SUPERVISED LEARNING WITH A NONLINEAR TEACHER

The linear teacher used in the previous section was derived by linearizing the cart-pole system's dynamics about $\varnothing = 0$, thus it is only applicable around the state space origin. An automated teacher that is stabilizing in the entire state space for the nonlinear system Equations (21) and (22), can be obtained by the use of a Feedback Linearizing and Decoupling Transform (FLDT) as used previously in Section 2. Rewrite the system Equations (21) and (22) in the form

$$\ddot{\varnothing} = h_1 - h_2 \ddot{X}, \tag{27}$$

$$\ddot{X} = \frac{f_1 + u}{f_2}, \tag{28}$$

where

$$h_1 = \frac{3}{4L} g \sin \varnothing, \tag{29}$$

$$h_2 = \frac{3}{4L} \cos \varnothing, \tag{30}$$

$$f_1 = m\left(L \sin \varnothing \dot{\varnothing}^2 - \tfrac{3}{8} g \sin 2\varnothing\right) - f\dot{X}, \tag{31}$$

$$f_2 = M + m\left(1 - \tfrac{3}{4} \cos^2 \varnothing\right). \tag{32}$$

Then the FLDT control law that is to be used as the automated teacher is

$$u = \frac{f_2}{h_2}\left[h_1 + k_1(\varnothing - \varnothing_d) + k_2\dot{\varnothing} + c_1(X - X_d) + c_2\dot{X}\right] - f_1, \tag{33}$$

with $k_1 = 25$, $k_2 = 10$, $c_1 = 1$ and $c_2 = 26$.

Examples generated by the above FLDT control law (17) were used to train a SNC. Figure 12 compares the performance of the FLDT teacher and its corresponding SNC. The SNC used was trained for 80 000 iterations of the BEP algorithm.

Although the nonlinear teacher and corresponding trained SNC do not follow trajectories that match as closely as in the case of the linear teacher, they both exhibit the same characteristic transient and steady state behavior. Subsequent tests also show similar performance. We conclude that the SNC was able to learn and generate stable control from the nonlinear teacher.

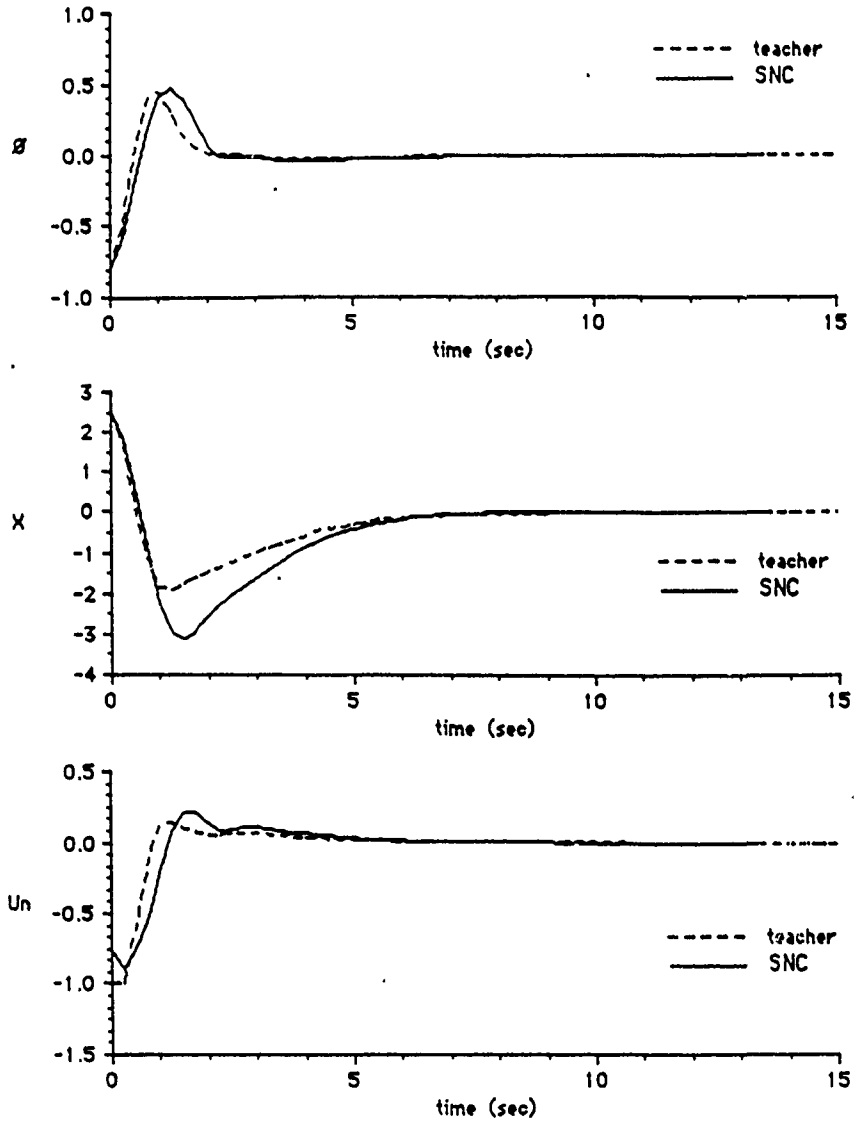


Fig. 12. Comparison of nonlinear teacher and corresponding SNC.

3.4. SUPERVISED LEARNING WITH A HUMAN AS THE TEACHER

In the previous sections, the exact form of the control law that was used as the teacher was a-priori known. In this section, we used examples generated by the responses of a human. In training with a human teacher, the exact form of the control law is not known as it has been acquired by the teacher though direct interaction with the system rather than through formal analysis of the system's dynamics. To generate the training data the teacher observes the cart-pole system on the computer screen and

returns the system to the state space origin by application of the control signal via an input device. The results reported are for Human Trained Supervised Neurocontrollers (HSNC) that have been trained for 40 000 iterations of the BEP algorithm. Figure 13 depicts the angle of the pendulum ($\varnothing$ in radians), the position of the cart (X in meters) and the normalized control signal (u_n) as a trained HSNC stabilizes the system from the initial state $Z = (2, 0, -0.4, 0)^T$.

Figure 13 demonstrates that the HSNC was able to learn to generate stable control from the examples of the humans response.

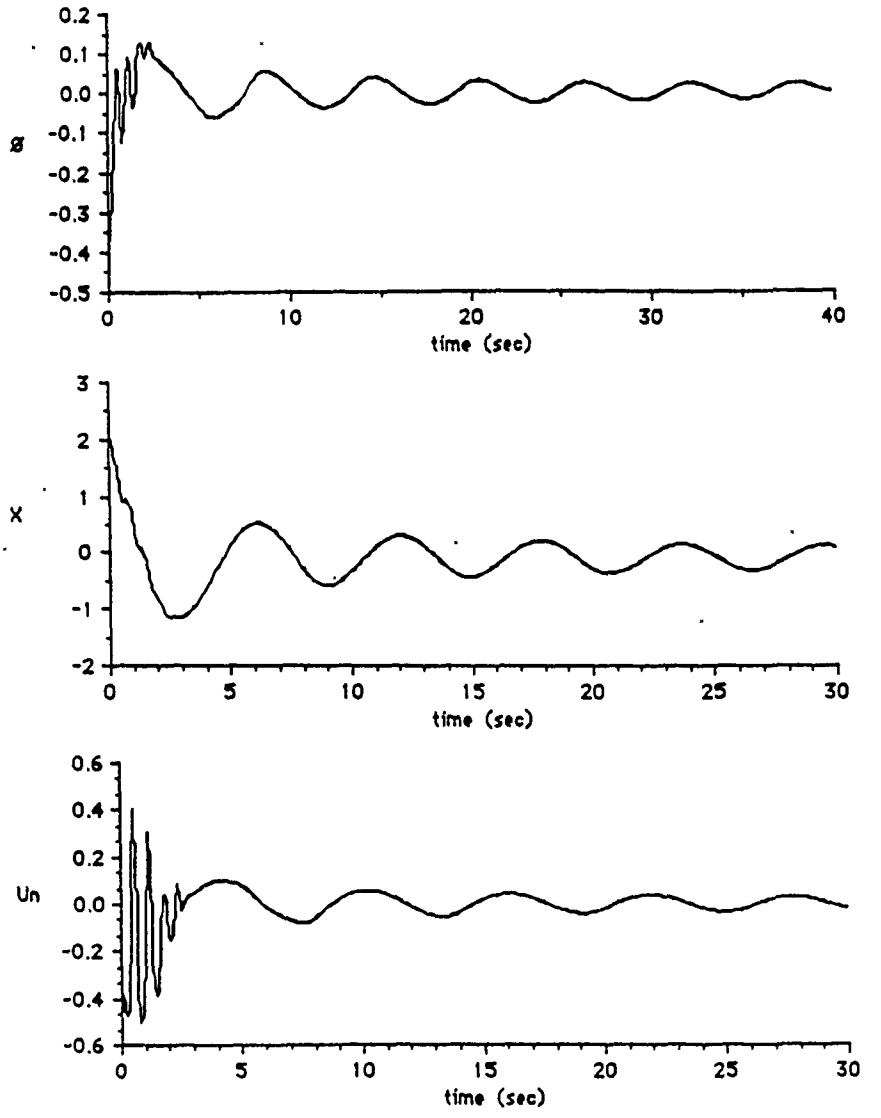


Fig. 13. Results of training HSNC

3.5. SUPERVISED LEARNING WITH A HUMAN AS THE TEACHER USING FILTERED DATA

When training a network using examples that have been generated by a human teacher, we have to take into account that the teacher performs some processing of the information before applying the control signal. State information is obtained by the teacher through the image on the computer screen, then the control signal that is applied is shaped not only by the teachers intention, but also by physiological characteristics. Characteristics such as response delay (reaction time) and hand tremor can be a limiting factor in a human's ability to perform manual tracking tasks (see Guez and Boykin, 1982). In our studies, the response delay has been shown to play a significant role in the ability of a network to learn from a human. To correct this problem we have filtered the training data to account for the simplest model of the human transfer function

$$G(jw)_{\text{human}} = e^{-jw\,T_{\text{human}}}, \tag{34}$$

which is a time delay. The filtering consists of shifting the sampled normalized control signal by the estimated reaction time of the teacher. In our experiments, the samples are recorded every 50 ms. The ith sample of the state variables was paired with the $(i + 2)$th sample of the normalized control signal, to produce a total shift of 100 ms. Figure 14 shows the performance of an HSNC that was trained using the same data as in the previous section, only the data was filtered as described above. The initial state is the same as in Figure 13.

By comparing Figures 13 and 14, we see that the performance of the network that was trained using the filtered data is much improved.

Furthermore, as can be seen in Figure 15, the performance of the network surpasses that of the teacher. This is reasonable in that a teacher with an inherent time delay would have to have perfect knowledge of the cart's dynamics to compensate for the delay. By shifting the data, we allow the network to learn the intent of the teacher rather than what physiological limitations allow. Also, note that the network was able to learn the control from very noisy training data as would be expected in a realistic situation.

In these experiments, the object is to return the cart-pole system to the origin of the state space. The controller trained with the human teacher and the filtered data has learned to stabilize the system, but it has not learned to return the cart to the origin ($X = 0$). We attribute this to the examples not providing a representative of all aspects of the control problem. This problem was manifested only in the controller trained with the filtered human examples. To correct this problem, a -0.33 meter bias has been added to the cart position input of the HSNC's reported in the rest of this article.

Comparison of HSNC and FLDT Controllers

We now compare the performance of the FLDT teacher and the human trained controller. Note that in this comparison the FLDT control law was derived using the

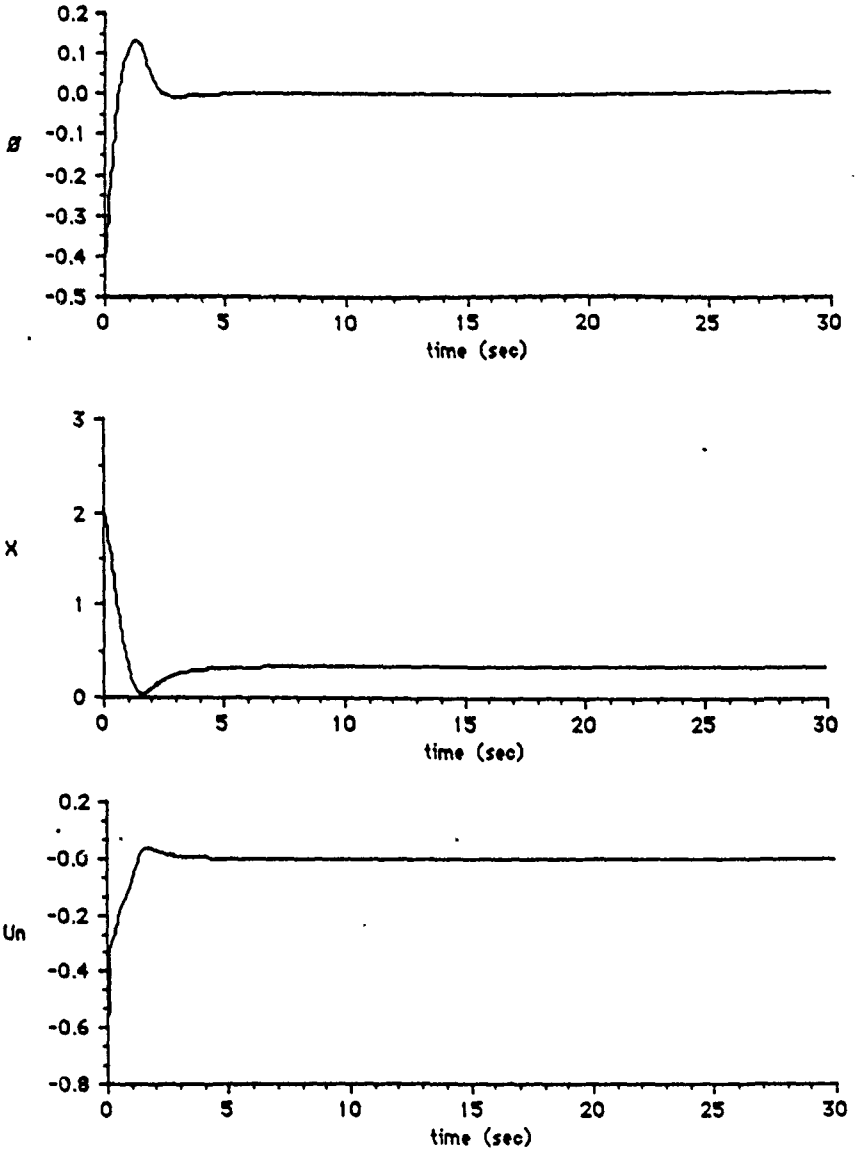


Fig. 14. Results of training HSNC using filtered data.

full knowledge of the cart-pole system's dynamics. The HSNC, on the other hand was trained using only the knowledge that the human teacher acquired through interaction with the cart-pole system. Figures 16 to 19 compare the performance of the HSNC and FLDT controllers for various initial states.

The performance of the HSNC is comparable to FLDT when the region of operation is within the region that the training examples represented. Note that

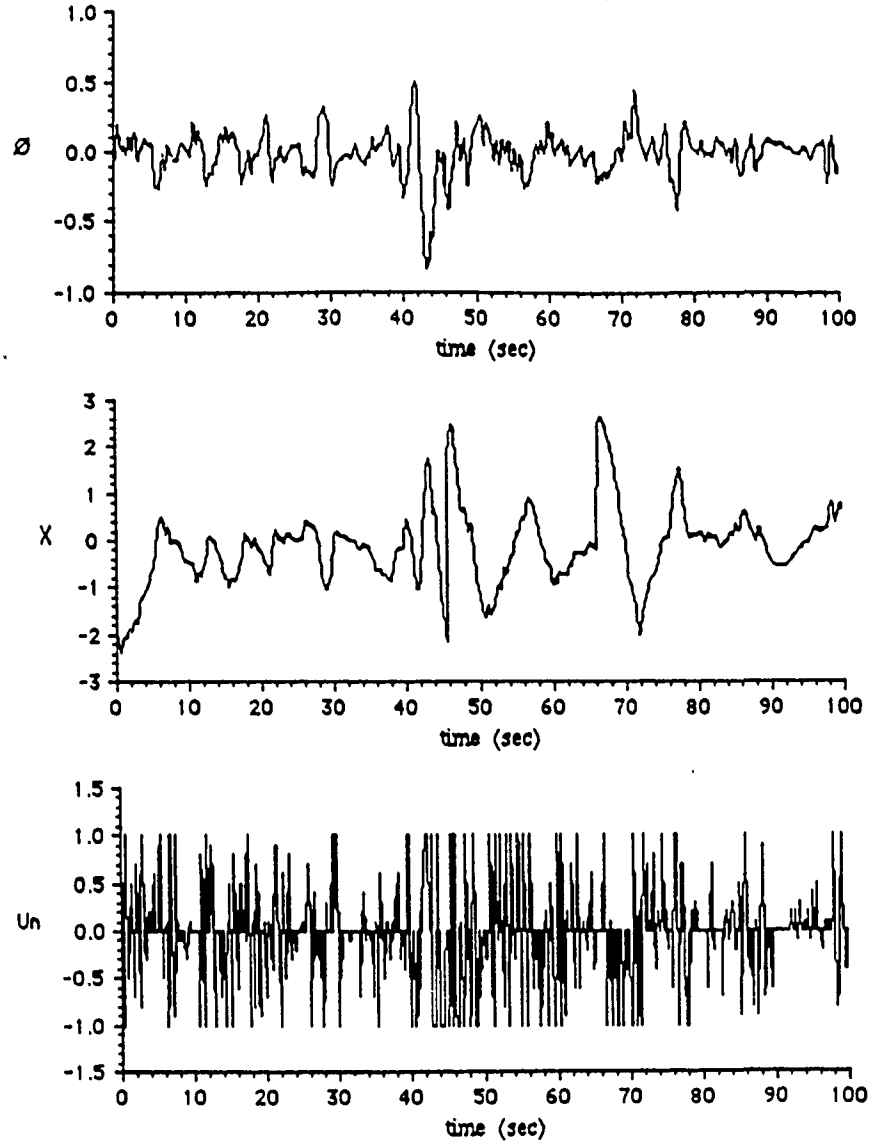


Fig. 15. Training data provided by the human teacher.

even though the performance is degraded in Figure 19 the network shows some generalization of its learning and is able to stabilize the system.

Comparison of Robustness of HSNC and FLDT

We compare the sensitivity of the HSNC and FLDT controllers to parameter variations. Figures 20 to 24 shows the angle of the pendulum as each controller returns

the cart-pole system to the state space origin from the initial state $Z = (1.5, 0, -0.5, 0)^T$. Each figure corresponds to one of the parameters being varied.

From Figures 20 to 24, we see that for variations in cart mass, pole mass, force gain and friction the HSNC is more robust than the model based FLDT controller.

4. Conclusions

We have outlined a method whereby the use of a-priori knowledge has resulted in a previously supervised learning algorithm to be made into an unsupervised one. The use of a hierarchical neural network, one where there are three layer network modules

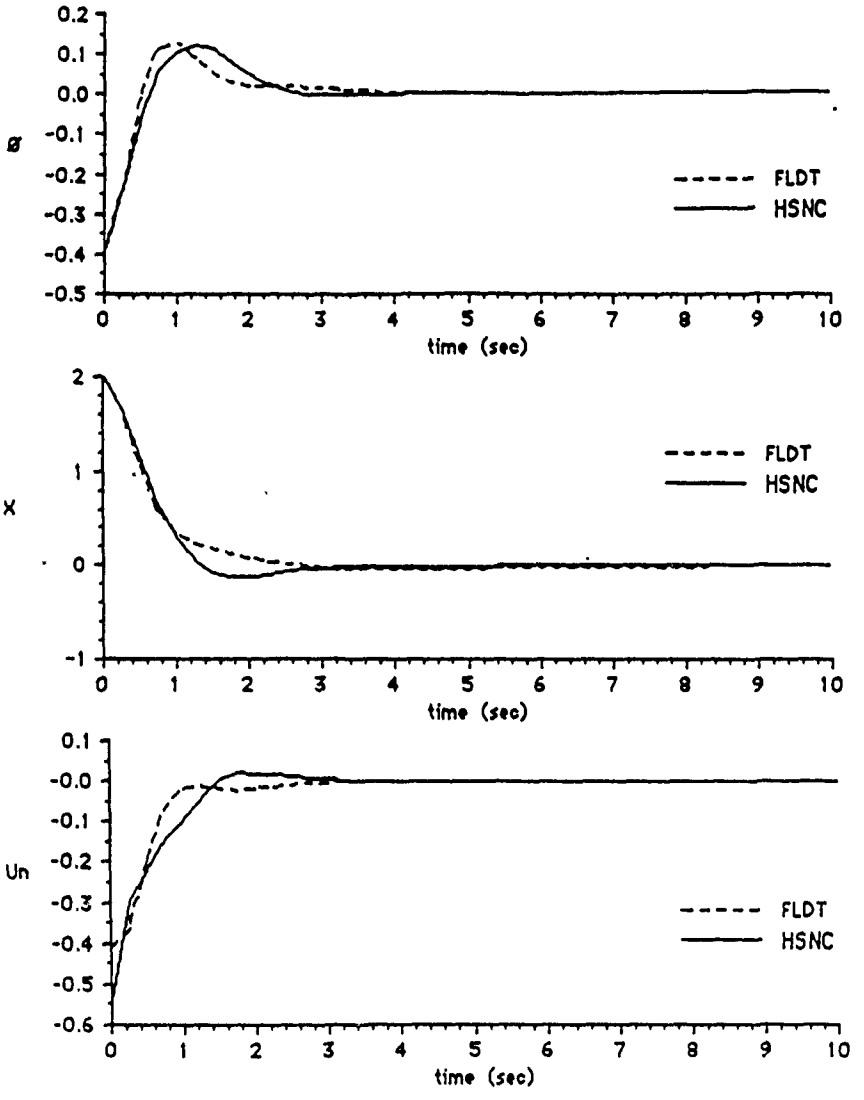


Fig. 16. Comparison of performance of HSNC and FLDT. $Z = (2, 0, -0.4, 0)^T$.

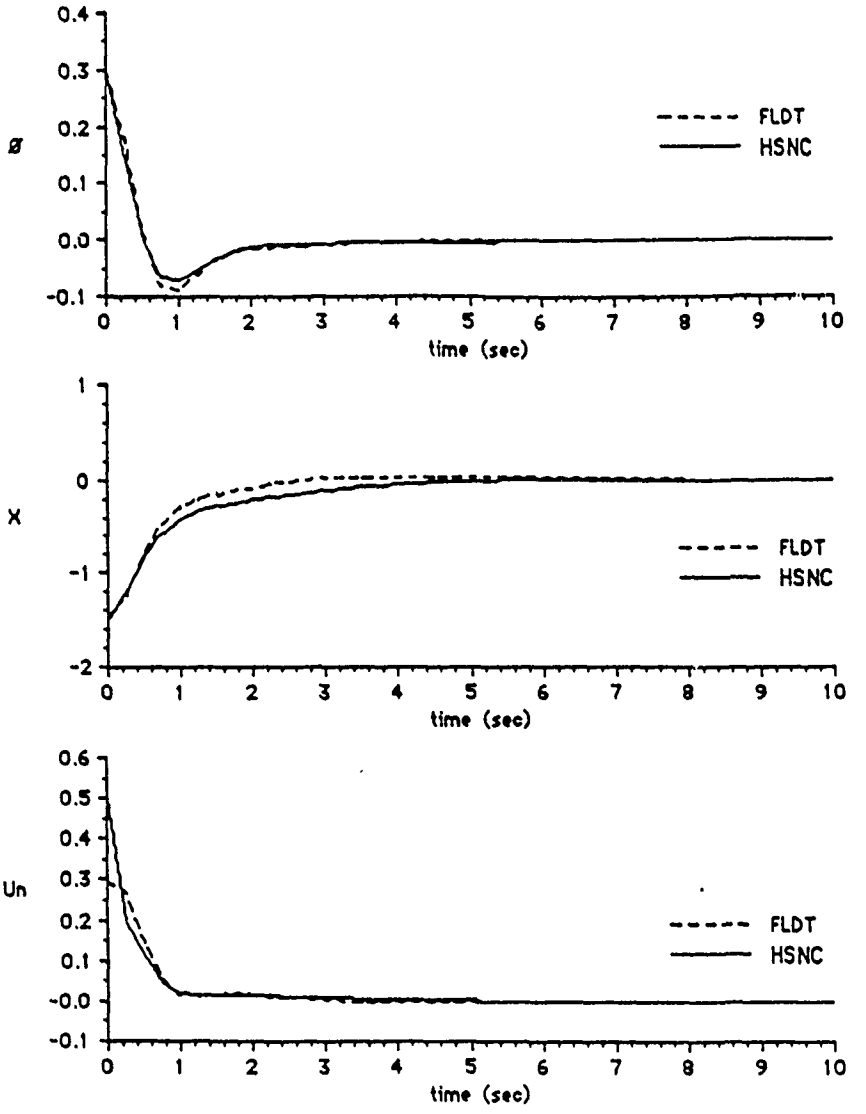


Fig. 17. Comparison of performance of HSNC and FLDT. $Z = (-1.5, 0, 0.3, 0)^T$

which are combined in a fourth layer and which allows selective attention to learning, has the potential to provide natural fault tolerance, robustness, flexibility of programming and high speed operation when implemented in hardware. We note that there will be a tradeoff between robustness and accuracy when using the network modules as functions rather than the exactly calculated functions. The networks can learn to approximate the functions but will not reproduce them exactly.

We have also shown that a neural network can be taught, using a supervised learning paradigm, to generate stable control for a nonlinear dynamic system. The

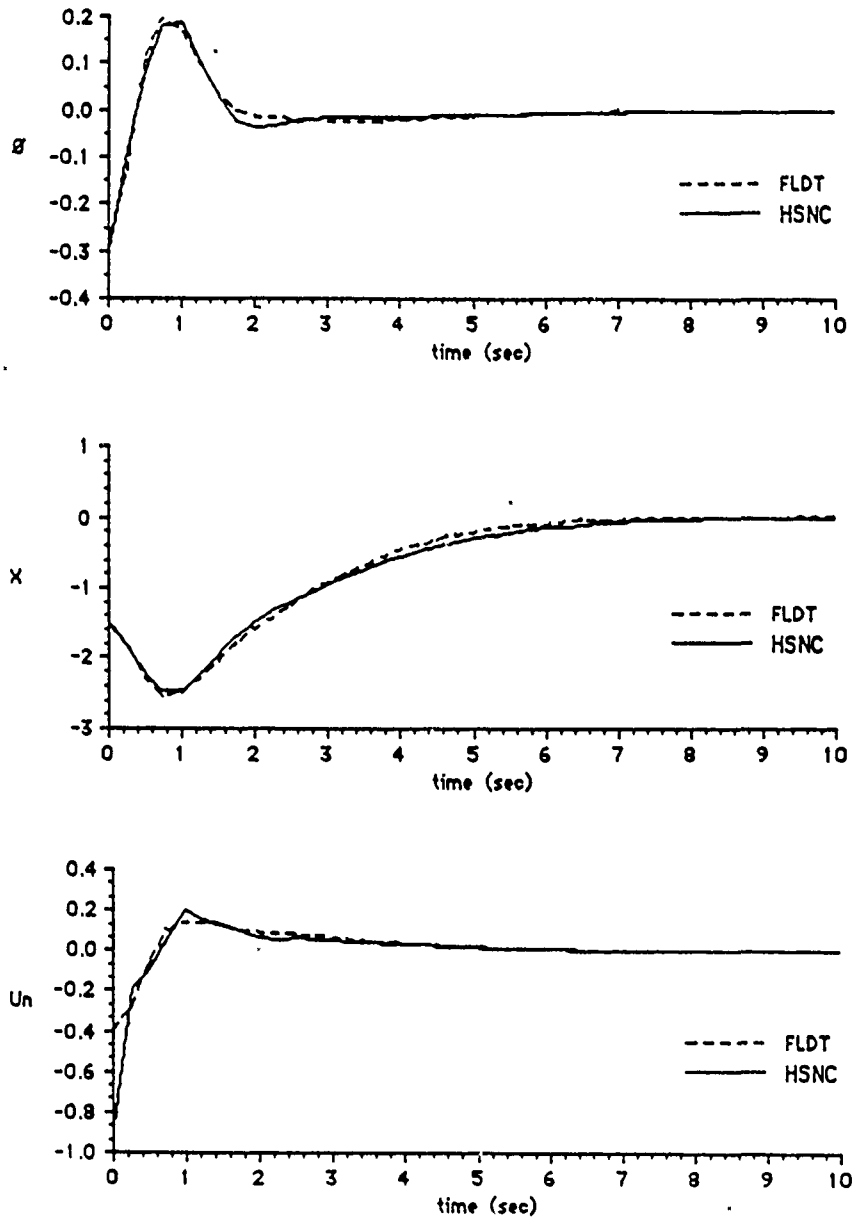


Fig. 18. Comparison of performance of HSNC and FLDT. $Z = (-1.5, 0, -0.3, 0)'$.

results of training with a human teacher demonstrate that stable control can be learned even when the exact form of the control law is not known. Furthermore, the human trained adaptive controller was shown to be more robust with respect to parameter variations than a model based controller.

We conclude from these results that neural networks using both supervised and

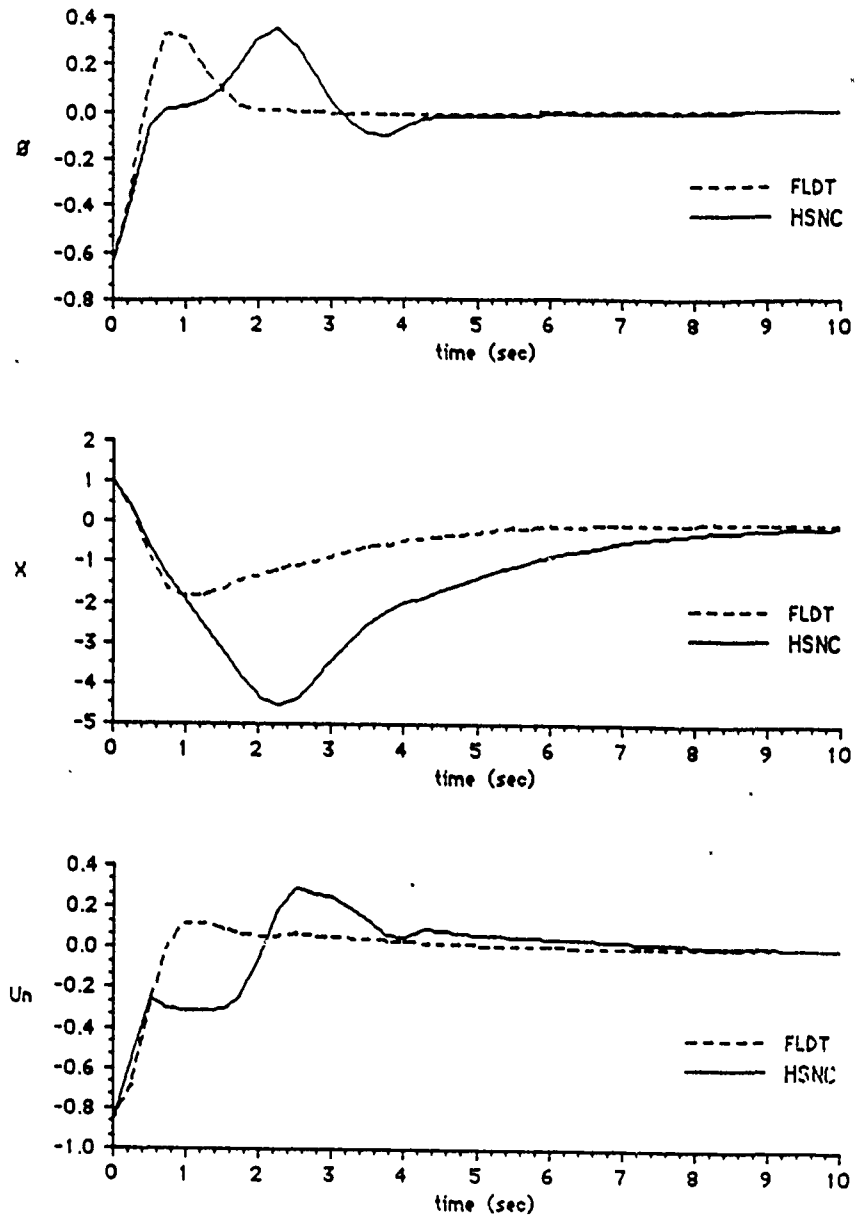


Fig. 19. Comparison of performance of HSNC and FLDT. $Z = (1, 0, -0.65, 0)^t$.

unsupervised learning paradigms can be a basis for robust, realtime, adaptive controllers. Moreover, we conclude that a-priori knowlege of the controlled process and control environment, no matter how little, can contribute to the structural design of neurocontrollers. Continuing research is directed towards issues of neurocontroller stability, ES design, and suitable network architectures (see Selinsky, 1989).

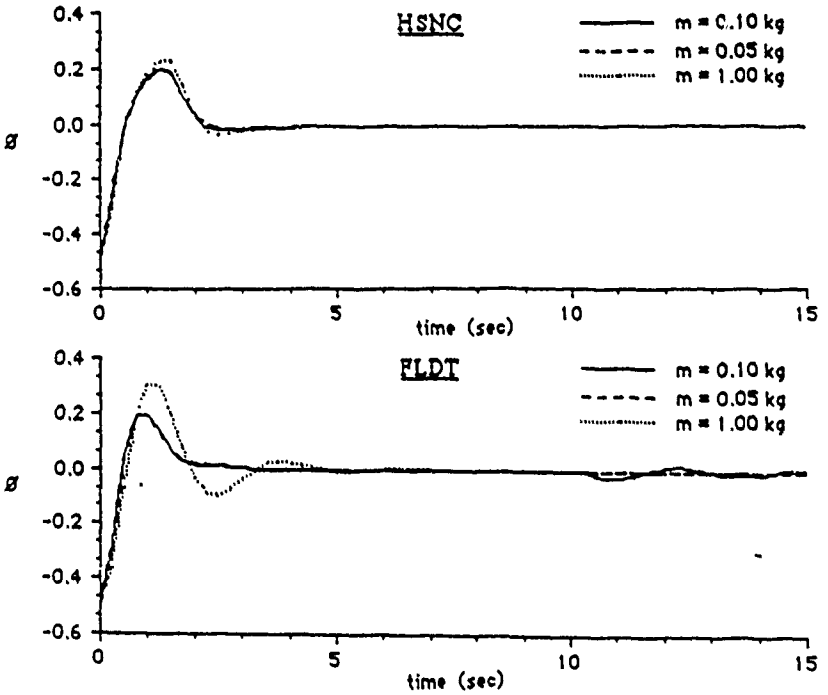


Fig. 20. Comparison of sensitivities of HSNC and FLDT to mass of the pole.

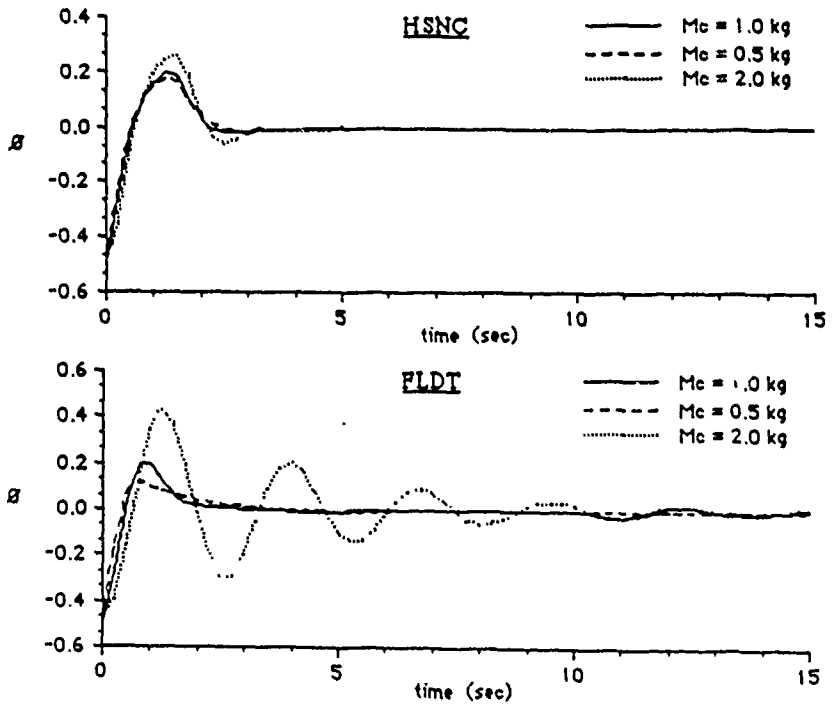


Fig. 21. Comparison of sensitivites of HSNC and FLDT to mass of the cart.

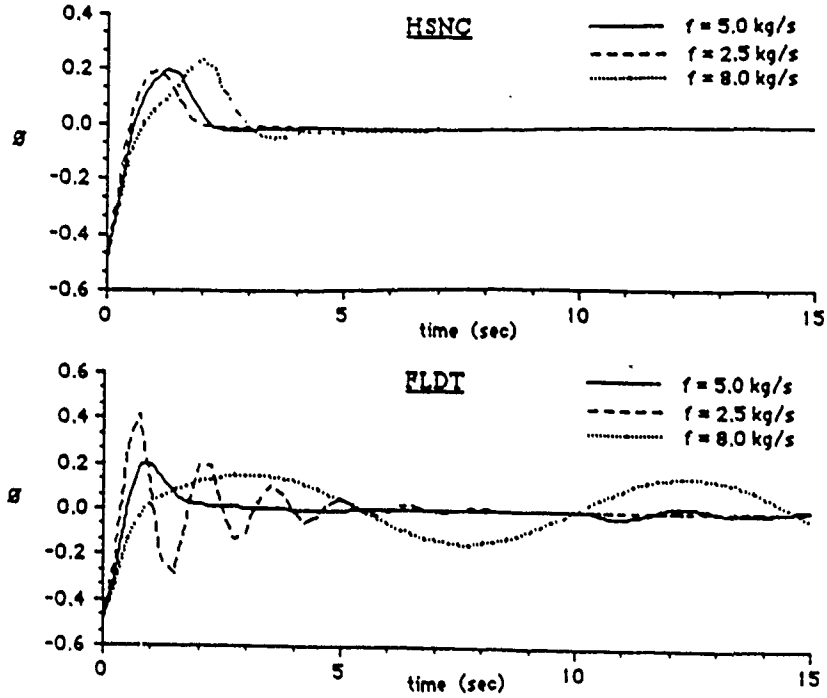


Fig. 22. Comparison of sensitivities of HSNC and FLDT to friction.

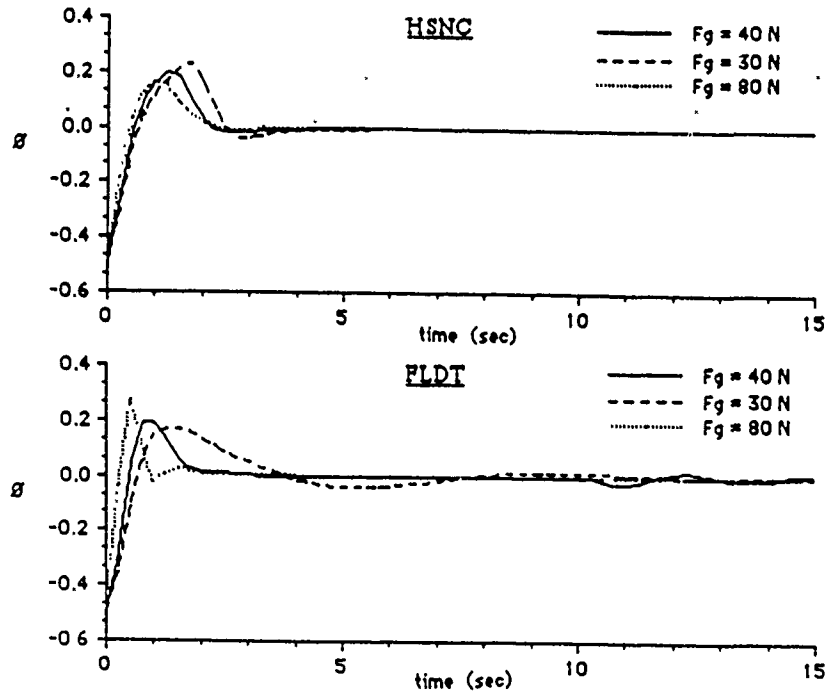


Fig. 23. Comparison of sensitivities of HSNC and FLDT to force gain.

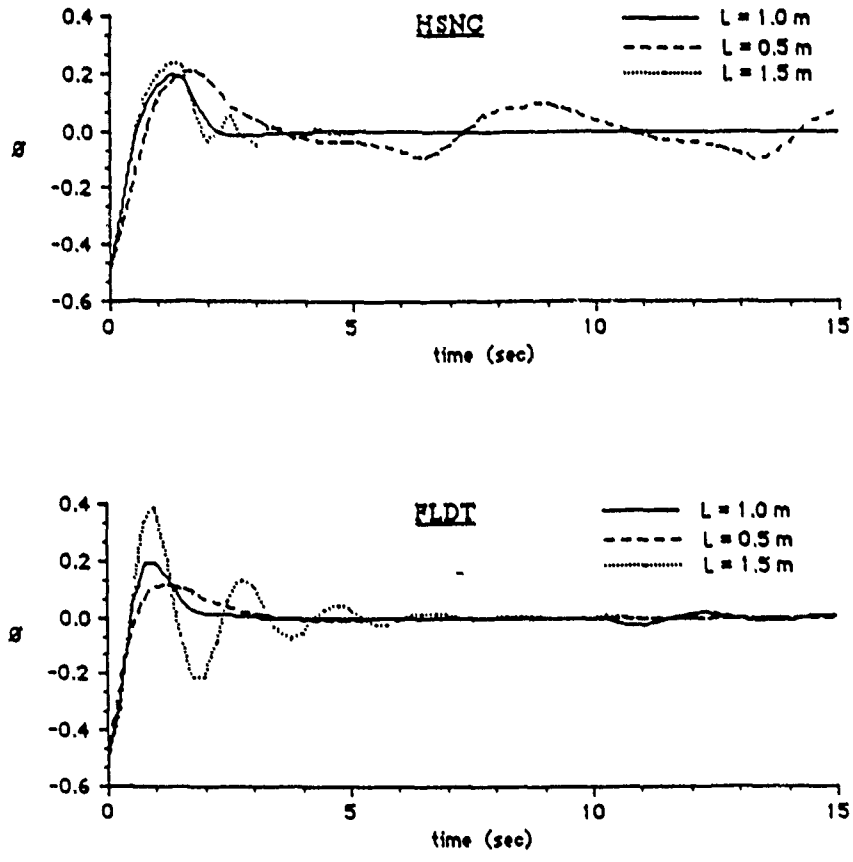


Fig. 24. Comparison of sensitivities of HSNC and FLDT to length of the pole.

References

Barto, A.G., Sutton, R.S., and C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Systems Man Cybernet.* SMC 13, 834-846 (1983).

Fu, K.S., Gonzales, R.C., and Li, C.S.G., *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, 1987.

Guez, A., Eilbert, J., and Kam, M., Neuromorphic architecture for control, *IEEE Control Systems Magazine*, April 1988.

Guez, A., and Selinsky J., A trainable neuromorphic controller. *J. Robotics Systems*, August 1988.

Guez, A., Optimal control of robotic manipulators, PhD Thesis, Univ. of Florida, Gainesville, Florida, 1982.

Guez, A., and Boykin, W.H., Manual tracking modeling and analysis of a helicopter borne precision laser pointing system, System Dynamics Incorporated, Technical Report 82-5, 1982.

Kawato, M., Furukawa, K., and Suzuki, R., A hierarchical neural-network model for control and learning of voluntary movement, *Biological Cybernetics* 56.

Kwakernaak, H. and Sivan, R., *Linear Optimal Control Systems*, Wiley, New York, 1972.

Paul, R.P., Robot Manipulators: Mathematics, Programming and Control, MIT Press, Mass., 1981.

Psaltis, D., Sideris, A., and Yamamura, A., Neural controllers, *Proc. IEEE First International Conference on Neural Networks, San Diego*, 1987.

Rumelhart, D., Hinton, G.E., and Williams, R.L., Learning internal representations by error propagation, in D.E. Rumelhart and J.L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, MIT Press, Mass, 1986.

Selinsky, J., Closed loop learning of rigid robot dynamics via design of exploratory schedules. MS thesis, Drexel Univ., Philadelphia, PA, 1989.

Slotine, J.J. and Li, W., Adaptive manipulator control: a case study, *Proc. IEEE Robotics and Automation Conference, Rayleigh N.C.*, 1987.

Widrow, B. and Smith, F.W., Pattern recognizing control systems, *1963 Computer and Information Sciences Symposium Proceedings, Washington, DC*, 1964.

Widrow, B. The original adaptive broom balancer, *IEEE Conference on Circuits and Systems, Philadelphia P.A*, 1987.

[8]

TWO DEGREE OF FREEDOM NEUROCONTROLLER

Allon Guez and Izhak Bar-Kana
ECE Dept., Drexel University, Philadelphia, PA 19104

Summary

In this paper we combine the simple technique for adaptive robot control proposed by [Bar-Kana and Guez 1989] and the exploratory learning schedule proposed by [Selinsky and Guez 1989] into a unified two degrees of freedom (DOF) robot controller, thereby improving the performance of both approaches. Both approaches are first summmarized then the 2 DOF controller is presented and the motivation for its design is explained.

Keywords:Neurocontrol; Learning; Tracking.

In [Bar kana and Guez 1989] an unsupervised distributed parallel computing architecture is proposed for the adaptive control of nonlinear dynamic systems of the class

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \qquad (1)$$

$$y(t) = C(x)x(t) + D(x)u(t) \qquad (2)$$

with some degree or other of uncertainty, where $x(t)\ R^n$ is the plant state vector, $y(t)\ R^m$ is the output vector, and $u(t)\ R^m$ is the input command vector, and where A(x), B(x), C(x), and D(x) are uniformly bounded matrices of corresponding dimensions.

The proposed controller consists of a teacher model which incorporates the knowledge regarding the desired input/output plant response as well as the repertoire of reference commands that the system may be subjected to.

The teacher dynamic model is assumed to have the following representation:

$$\dot{x}_t(t) = A_t(x_t)x_t(t) + B_t(x_t)u_t(t) \qquad (3)$$

$$y_t(t) = C_t(x_t)x_t(t) + D_t(x_t)u_t(t) \qquad (4)$$

where $x_t(t)\ R^{n_t}$ is the state vector, $y_t(t)\ R^m$ is the output vector, and $u_t(t)\ R^m$ is the input command vector, and where $A_t(x_t)$, $B_t(x_t)$, $C_t(x_t)$, and $D_t(x_t)$ are uniformly bounded matrices of corresponding dimensions. It is emphasized that the dimension of the model is unrestricted, except that $\dim(y_t) = \dim(y) = m$.

The parallel distributed adaptive controller receives the input 'features' vector $f(u_t, x_t, y)$ and generates as an output the process control vector u, where

$$u(t) = K(t)f(u_t, x_t, y) \qquad (5)$$

where K(t) is the adaptive gain matrix of appropriate dimension. Each K_{ij} gain monitors the sensitivity of the i-th control loop, namely u_i, to the j-th feature of the system, namely $f_j(u_t, x_t, y)$ and may be viewed as the state of the *ij*-th neuron.

The K_{ij} gain adjusts its value independently of, and simultaneously with all other gains, according to:

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \quad (6)$$

$$M_{ij}(t) = a_{ij}(y_{t_i} - y_i)f_j \quad (7)$$

$$\frac{d}{dt} N_{ij}(t) = - b_{ij}N_{ij}(t) + g_{ij}(y_{t_i} - y_i)f_j \quad (8)$$

where a_{ij}, b_{ij} and g_{ij} are positive constants.

Bar-Kana and Guez show that perfect regulation of the plant may be obtained in the purely deterministic case. Notice however that, while reducing the amount of prior knowledge **required** to guarantee stability of the adaptive systems, this simple adaptive controller does not take into account any valid knowledge on the specific plant to be controlled. No learning is used to improve performance in future trials or to make the control assignment easier. The initial adaptive gains are always zero, as no better initial value is available, and also the range of the adaptive gains may me unnecessarily large. Indeed the main purpose of this simple algorithm is first of all to reduce the necessary condition to mere stabilizability.

The system proposed by [Selinsky and Guez 1989] for robot manipulator control consists of the Exploratory Schedule Generator, Neurocontroller and associated Learning Algorithm, the Robot, and a mechanism which allows selecting between the User / Path Planner and ES Generator as the originator of the desired trajectory (q_d).

The neurocontroller implements the control law

$$t_i = \sum_{j=1}^{p} Y_{ij}[q, \dot{q}, \dot{q}_r, \ddot{q}_r]\hat{\emptyset}_j + K_{d_{ii}}\dot{e}_{r_i}, \qquad i = 1,2, \ldots, n, \qquad (9)$$

where $\hat{\emptyset}$ denotes the estimate of the robot parameters $\emptyset$, **and the $K_{d_{ii}}$ are constant weights** for the servo portion of the controller . Selinsky and Guez indeed use learning in the adaptation process. Their exploratory schedules also guarantee ...at along with tracking, the unknown plant parameters are finally identified. Yet, in order to get satisfactory results for both tracking and learning, the supplementary gains $K_{d_{ii}}$ are needed in (9). The results of both tracking and learning are strongly affected by the right selection of these parameters , and in practice they are fixed by trial and error procedures. Furthermore, if one selects high fixed gains to get small tracking errors, one then also gets high noise amplification and possibly high cost of control even when not necessarily needed. An adaptive procedure for $K_{d_{ii}}$ like in Bar-Kana and Guez, will fix the gains as a function of the tracking errors. It thus results in large gains only if the tracking error (in our case represented by $\dot{e}_r$) tends to increase, and decrease afterwards, fitting thus the right control gains to the right situation.

Instead (see [Bar Kana and Guez 1989-b] for details) we propose the 2 DOF controller shown in Figure 1.

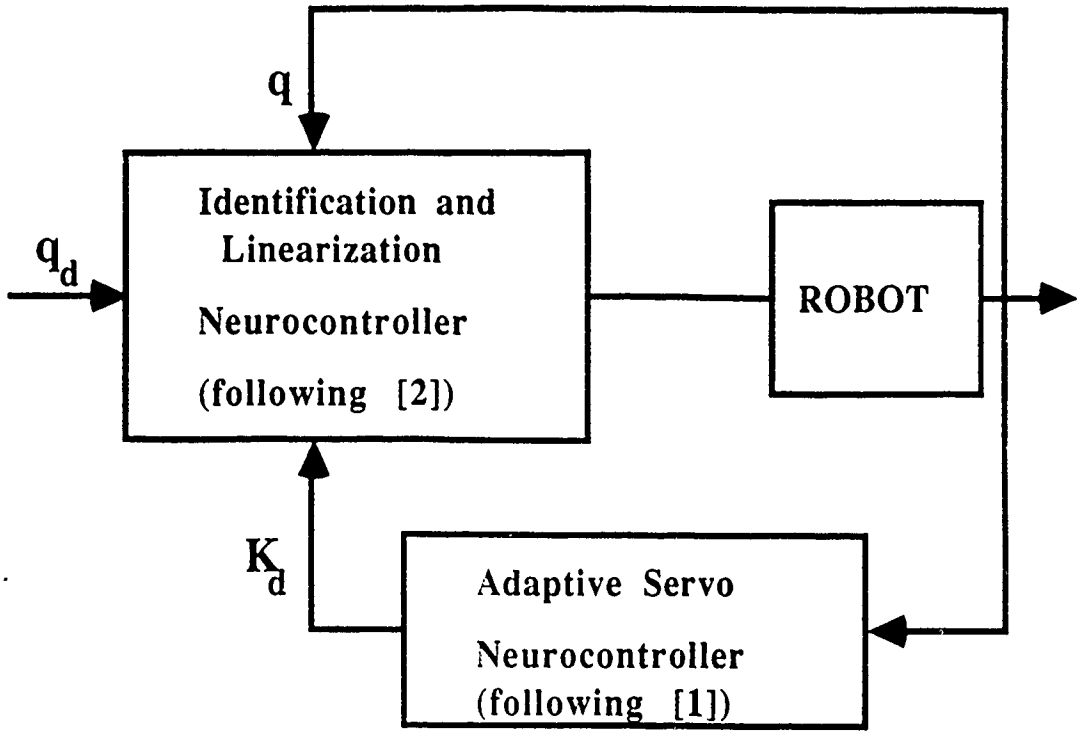


Fig. 1: Two Degrees of Freedom NeuroController.

The identification and linearization neurocontroller algorithm is given by equations (9), and (10) and the adaptive servo neurocontroller algorithm has the form of equations (5), (6), (7) and (8) Main Result. It can be shown [Bar-Kana and Guez 1989-b] that global asymptotic tracking and identification as in [Selinsky and Guez 1989] still holds even for the 2 DOF controller, i. e., when the servo gain is time varying. That is, global asymptotic tracking as well as global asymptotic identification of up to n robot parameters can be guaranteed with our 2 DOF robot controller if

$$\lim_{t \to \infty} [\text{Rank } (Y(q, \dot{q}, \dot{q}_d, \ddot{q}_d))] = p.$$

REFERENCES

I. Bar-Kana and A. Guez, 1989 "Neuromorphic Adaptive Control," *Proc. 28th Conference on Decision and Control,* Tampa, Florida, 1989, pp. 1739-1743.
I. Bar-Kana and A. Guez, 1989 b *A Two Degree Of Freedom Adaptive Controller,* Technical Report, Drexel University.

J. W. Selinsky and A. Guez, 1989 "The Role of Apriori Knowledge of Plant Dynamics in Neurocontroller Design," *Proc 28th Conference on Decision and Control,* Tampa, Florida, 1989, pp. 1754-1758.
J. J. E. Slotine and W. Li., 1988 "Adaptive Manipulator Control: A Case Study," *IEEE Transactions on Automatic Control,* Vol. 33, No. 11, pp. 995-1003, 1988.

IEEE CSS
ON DECISION AND CONTROL
Hilton Hawaiian Village
Honolulu, Hawaii
December 5-7, 1990
[9]

TWO-DEGREE-OF-FREEDOM ROBOT NEUROCONTROLLER

Allon Guez and Izhak Bar-Kana
ECE Dept., Drexel University, Philadelphia, PA 19104

ABSTRACT

This paper describes the performance of a two-degree-of-freedom (2 DOF) robot neurocontroller. The main theorem suggests that global asymptotic stability is guaranteed despite the highly nonlinear nature of the closed-loop dynamics. Examples demonstrating the applicability of the neurocontroller are provided.

1. INTRODUCTION

The neurocontroller proposed here is based on a combination of the architectures proposed in [1],[2], [3], and [4] and demonstrates some useful features. In section 2 we describe the controller architecture. Section 3 presents the main results. In section 4 we provide a two-links rigid manipulator example.

2. THE ADAPTIVE CONTROLLERS

The proposed 2 DOF neurocontroller is described in Figure 1.

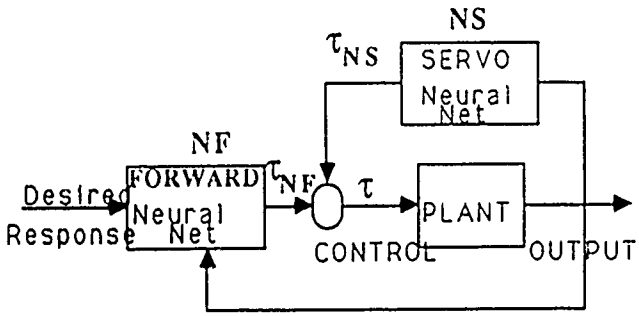


Figure 1. 2 DOF Neurocontroller

We shall develop the neurocontroller and describe its performance for a rigid n DOF robot manipulator. The marhematical notations and the underlying component algorithms follow closely the references [1] and [3].
The robot dynamic model is assumed as follows:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad (1)$$

where

$D(q)$	is an n x n matrix of inertial terms,
$C(q,\dot{q})$	is an n x n matrix of coriollis and centrifugal terms,
$G(q)$	is an n x 1 vector of gravitational terms,
q	is an n x 1 vector of joint coordinates,
τ	is an n x 1 vector of forces/torques,
n	is the number of degrees of freedom.

Equation (1) can also be expressed as

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\dot{q},\ddot{q})\Theta \qquad (2)$$

where $Y(q,\dot{q},\dot{q},\ddot{q})$ is an *nxp* matrix of known functions and Θ is an *px1* vector of weighting constants (Slotine and Li [3]). When (2) is written in terms of the individual torques at each joint, it can be viewed as a single layer linear network, where the inputs to the network are the $Y_{ij}(.)$ and the weights are the Θ_j (Fig. 2). Notice that the $Y_{ij}(\cdot)$ are transcendental algebraic functions of the manipulators states that are a priori known and may be realized via feedforward neural networks that are trained offline with a suitable learning algorithm.

Let q_d, $\dot{q}_d$, $\ddot{q}_d$ designate the desired trajectory. We define $e(t) = q(t)-q_d(t)$ the joint coordinate error, and following [3], we define the virtual reference trajectory q_r, $\dot{q}_r$, $\ddot{q}_r$, and the virtual trajectory velocity error $s = \dot{e}_r$

$$\dot{q}_r = \dot{q}_d - A\,e, \quad \ddot{q}_r = \ddot{q}_d - A\,\dot{e}, \quad s = \dot{q} - \dot{q}_r = \dot{e} + Ae \qquad (3)$$

where A is a positive definite matrix with constant coefficients.

The neurocontroller provides the control command $\tau = \tau_{NF} + \tau_{NS}$, where τ_{NS} is the output of the servo neural net (NS) and τ_{NF} is the output of the forward neural net (NF).

We choose the architecture of NF to implement the control law

$$\tau_{NF} = \sum_{j=1}^{p} Y_{ij}(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\Theta}_j, \quad i = 1,2,.....,n \qquad (4)$$

where $\hat{\Theta}$ denotes the estimate of Θ, and is implemented as a synaptic weight in NF with the Slotine and Li [3] learning (adaptation) rule:

$$\dot{\hat{\Theta}}_j = -\sum_{i=1}^{n} \frac{1}{K_{a_{ii}}} Y_{ij}(q,\dot{q},\dot{q}_r,\ddot{q}_r)\, s_i, \qquad i = 1,2,.....,p. \qquad (5)$$

Notice in equation (5), the similarity to the LMS learning rule (see Widrow and Stearn [5]) where the weight change is proportional to the error ε and the input features X. In equation (5) the input features are the $Y_{ij}(.)$ functions and the error is s_i.

Figure 2 shows the internal structure of the neurocontroller for joint i. Each feedforward network module is trained to provide one of the $Y_{ij}(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ functions.

The $Y_{ij}(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ are implemented by neural networks providing the opportunity to capture nonrigid dynamics effects when their number overestimates n.

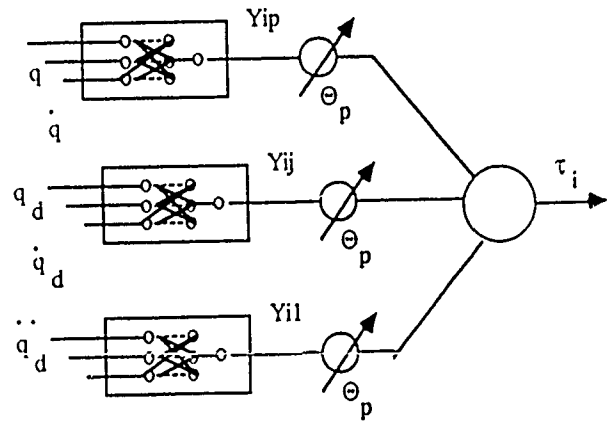


Figure 2: Neural implementation of NF.

The servo neurocontroller NS implements the Bar-Kana adaptative rule [1], [2]

$$K_{d_{ij}}(t) = K_{dI_{ij}}(t) + K_{dp_{ij}}(t) \qquad (6)$$

$$\dot{K}_{dI_{ij}}(t) = \alpha_{ij} s s^T - \beta_{ij} K_{dI_{ij}}(t) \qquad (7)$$

$$K_{dp_{ij}}(t) = \gamma_{ii} s s^T \quad (8)$$

$$\tau_{NS} = K_d s \quad (9)$$

Notice that without the dynamic NS and without the neural implementation of the NF, the neurocontroller reduces to the adaptive controller of Slotine and Li [3]. For constant servo feedback (constant K_d) they proved global stability. We shall extend their result and prove that global stability is maintained with our adaptive servo (6)-(9), and with added benefit.

In [2] an unsupervised distributed parallel architecture is proposed for the adaptive control of nonlinear dynamic systems of the class

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \quad (10)$$

$$y(t) = C(x)x(t) + D(x)u(t) \quad (11)$$

with some degree or other of uncertainty, where $x(t) \in \mathbb{R}^n$ is the plant state vector, $y(t) \in \mathbb{R}^m$ is the output vector, and $u(t) \in \mathbb{R}^m$ is the input command vector, and where A(x), B(x), C(x), and D(x) are uniformly bounded matrices of corresponding dimensions.

The controller proposed in [2] is shown in Figure 3 below: It consists of a teacher model which incorporates the knowledge regarding the desired input/output plant response.

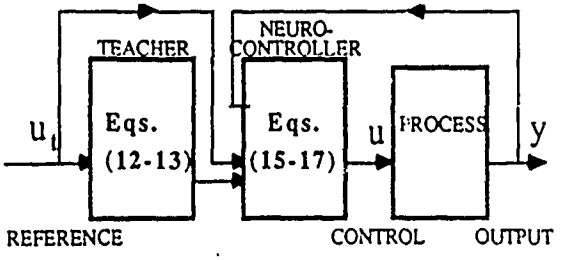


Figure 3: Neurocontroller Proposed in [2].

The teacher dynamic model is assumed to have the following representation:

$$\dot{x}_t(t) = A_t(x_t)x_t(t) + B_t(x_t)u_t(t) \quad (12)$$

$$y_t(t) = C_t(x_t)x_t(t) + D_t(x_t)u_t(t) \quad (13)$$

It is emphasized that the dimension of the model is unrestricted, except that $\dim(y_t) = \dim(y) = m$.

The parallel distributed adaptive controller (Fig. 4) receives the input 'features' vector $f(u_t, x_t, y)$ and generates as an output the process control vector u(t), where

$$u(t) = K(t)f(u_t, x_t, y) \quad (14)$$

where K(t) is the adaptive gain matrix of appropriate dimension. Each K_{ij} gain adjusts its value independently of, and simultaneously with all other gains, according to:

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \quad (15)$$

$$M_{ij}(t) = \alpha_{ij}(y_{t_i} - y_i)f_j \quad (16)$$

$$\frac{d}{dt} N_{ij}(t) = -\beta_{ij}N_{ij}(t) + \gamma_{ij}(y_{t_i} - y_i)f_j \quad (17)$$

where α_{ij}, β_{ij} and γ_{ij} are positive constants. Notice in equations (17) the similarity to the LMS learning rule (see Widrow and Stearn [5]) where the weight change is proportional to the error ε and the input features X. In equation (17) the input features are the f_j functions and the error is $y_{t_i} - y_i$. The β-term in (17) is added to avoid divergence of the integral adaptive gains in the presence of disturbances and uncertainties [1].

3. MAIN RESULT

Theorem: The dynamic system described by equations (1)-(9) and represented in figure 1 provides global asymptotic tracking for all positive α, β, γ, and for all twice differentiable trajectories $q_d(t)$.

Proof: Let us select the positive definite quadratic Lyapunov function of the parameters s, Θ̃, K_{dI}

$$V = \frac{1}{2} s^T D(q)s + \frac{1}{2} \tilde{\Theta}^T k_a \tilde{\Theta} + \frac{1}{2} \text{trace}\,[K_{dI}(t)\alpha^{-1}K_{dI}^T(t)] \quad (18)$$

Using the fact that for rigid robots $x^T[\dot{D} - 2C]x = 0 \; \forall x$, we obtain after some manipulations [7]

$$\dot{V} = -\alpha\|s\|^2 - \beta \text{Trace}\,[K_{dI}(t)\alpha^{-1}K_{dI}^T(t)] < 0 \; \forall \; s \neq 0, \; K_{dI} \neq 0.$$

Therefore $\lim_{t\to\infty} \|s\| = 0$ implying $q(t) \to q_d(t)$.

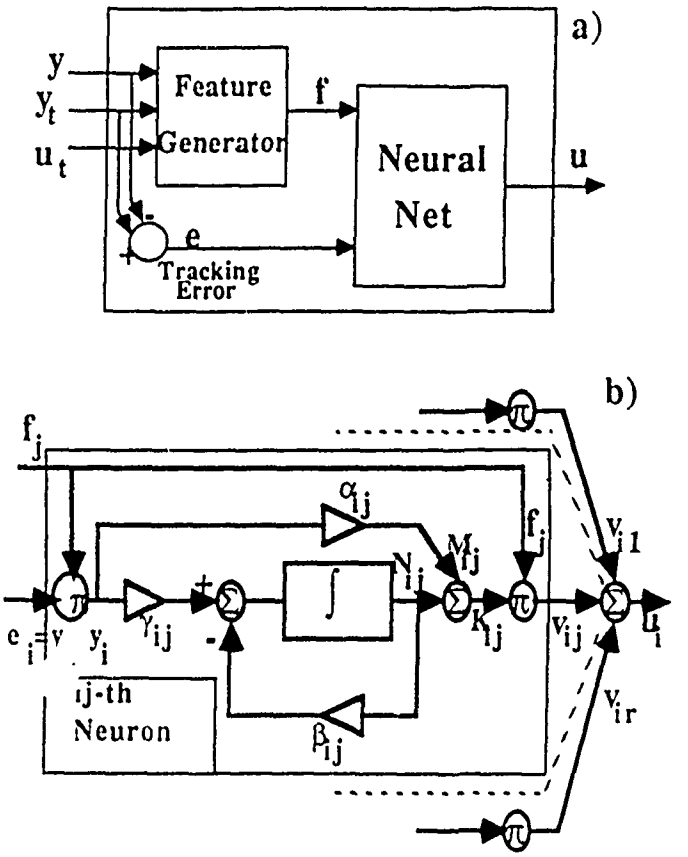


Figure 4 - Neurocontroller Architecture
a) General Block Diagram b)ij-th Neuron Architecture.

4. ROBOTIC EXAMPLE

We simulated the exact system as in [4] with our new controller. The equations of motion are:

$$\tau_1 = D_{11}\ddot{q}_1 + D_{12}\ddot{q}_2 + D(\dot{q}_2^2 + 2\dot{q}_1\dot{q}_2) + D_1$$

$$\tau_2 = D_{12}\ddot{q}_1 + D_{22}\ddot{q}_2 - D\dot{q}_1^2 + D_2$$

where

$L_1 = L_2 = 1.0$ m, $m_1 = m_2 = 10$ kg, $g = 9.81$ m/s^2.

$D_{11} = (m_1 + m_2)L_1^2 + m_2L_2^2 + 2m_2L_1L_2\cos q_2$,

$D_{12} = m_2L_2^2 + m_2L_1L_2\cos q_2$,

$D_{22} = m_2L_2^2$,

$D = -m_2L_1L_2\sin q_2$,

$D_1 = (m_1 + m_2)gL_1\sin q_1 + m_2gL_2\sin(q_1 + q_2)$,

$D_2 = m_2gL_2\sin(q_1 + q_2)$.

With the above numerical values, the plant parameters have the values $\Theta_1 = 20$, $\Theta_2 = 10$, $\Theta_3 = 10$, $\Theta_4 = 196.2$, $\Theta_5 = 98$ i The performance of the two-degree-of-freedom robotic control system is

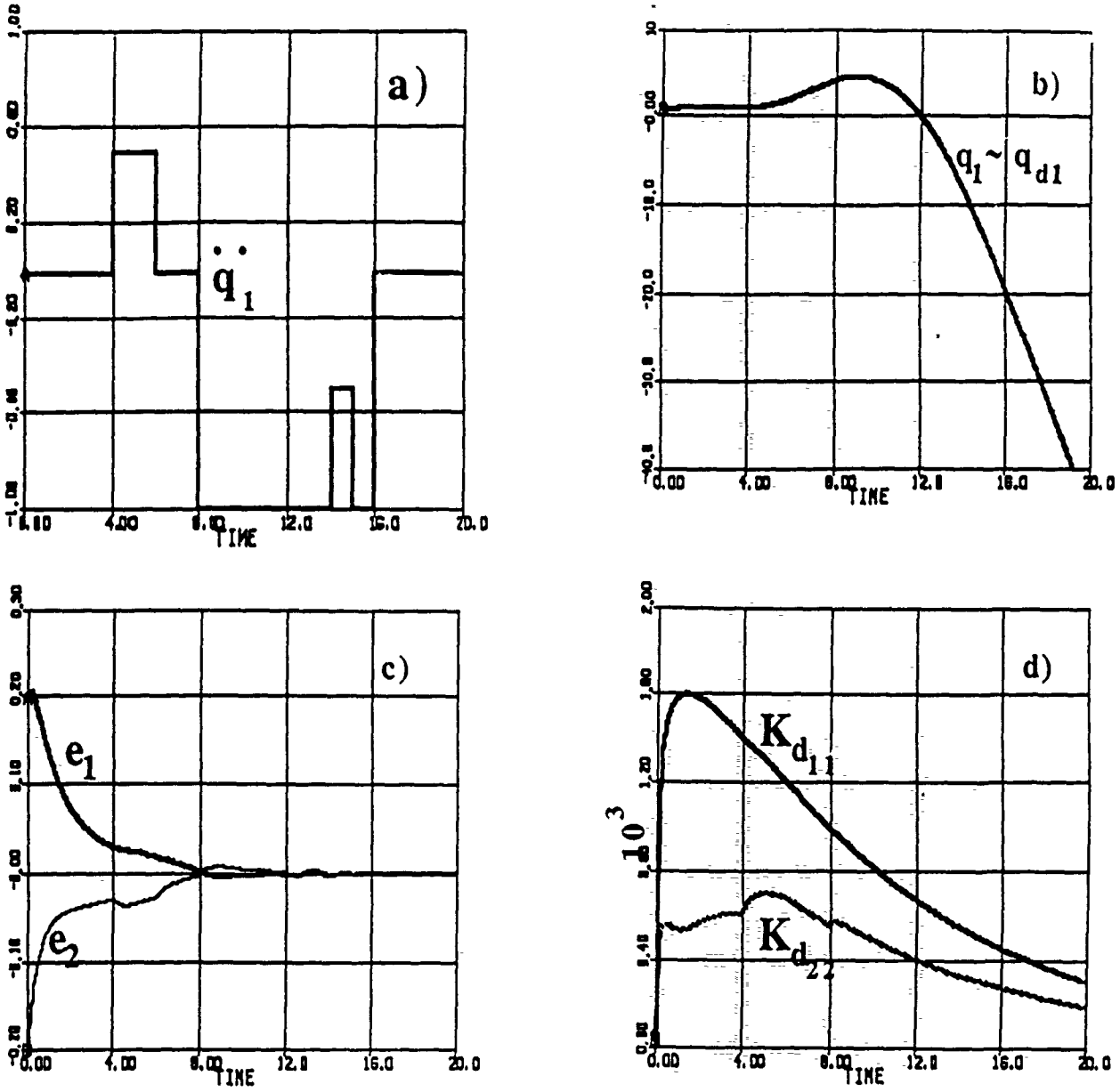


Figure 5. Adaptive Servo Gains: (a) Acceleration schedule; (b) Trajectories $q_{d_1}(t)$, $q_1(t)$; (c) Tracking Errors $e_1(t)$, $e_2(t)$; (d) Controller Gains $K_{d11}(t)$, $K_{d22}(t)$.

represented in Figure 5 with identification coefficients $K_{a_{ii}}$=0.001 and with adaptation coefficients α=100000., β=0.1, γ=1000. The parameters adjustment is only as good as the tracking system is affected. We do not introduce any persistant excitation, that is usually used to guarantee convergence of the parameters. However, at idle times between production times, the robot can execute exploratory schedules [4] in order to bring the identified parameters as close as possible to the actual values. Afterwards, the robot performs any desired assignment, and then NF and NS work in tandem to execute it. Figure 5a shows the acceleration process $\ddot{q}_{d1}$. Figure 5b represents the desired q_{d1} and the actual q_1, which at this scaling appear to be identical. During this experiment we kept q_{d2} = 0. Figure 5c represents the small tracking errors with a more appropriate scaling. Figure 5d shows the behavior of the adaptive gains $K_{11}(t)$ and $K_{22}(t)$. It can be seen how the adaptive gains move up-and-down and maintain small tracking errors. They may even vanish, if they are not needed any more.

The performance of the robotic control system with constant servo gains is strongly affected by the selection of the constant control gains $K_{d_{ii}}$. With identification coefficients $K_{a_{ii}}$=0.001 and with low gains $K_{d_{11}}$ = 100., $K_{d_{22}}$ = 50., both the identification and the tracking errors are bad [6]. With high constant gains $K_{d_{11}}$ = 1000., $K_{d_{22}}$ = 500, the results improve and are very similar to the results presented in Fig. 5. However, the servo gains remain always high, even if they are not needed.

We wanted the servo gains to drop when they are not needed any more, because we expected the controller with high gains to be strongly effected by disturbances, even after the plant parameters are actually identified. Figure 6 shows indeed that the identification is terribly disturbed by noise, and tracking is bad when high servo gains $K_{d_{11}}$ = 1000., $K_{d_{22}}$ = 500 are used with measurement disturbances of 0.1 sin 20t and 0.2sin30t, correspondingly. The surprize came when the adaptive control gave also similar (bad) results. This result can be explained by the deterioration of the identification due to the disturbance, which introduces a noisy control signal τ_{NF} to the robot.

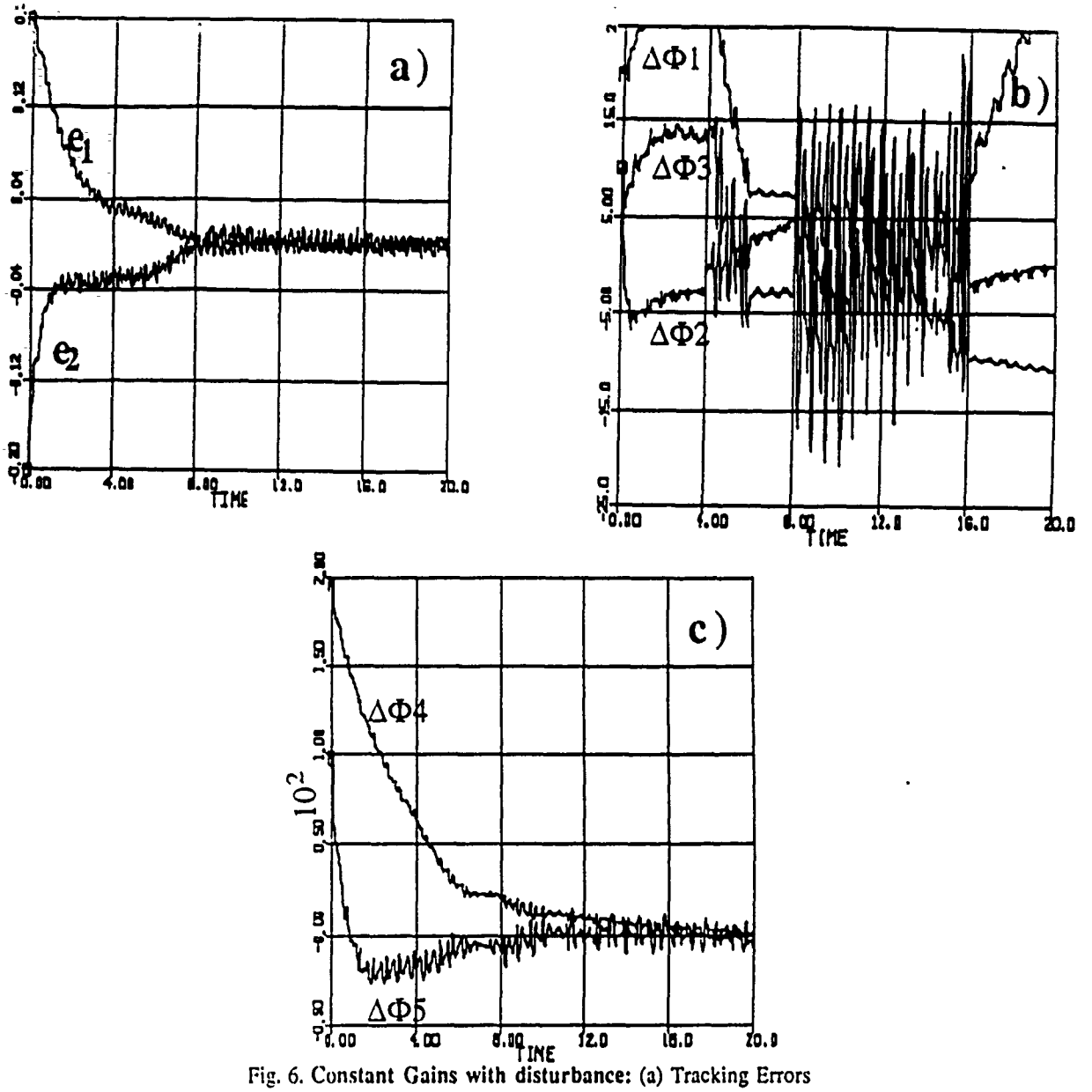


Fig. 6. Constant Gains with disturbance: (a) Tracking Errors $e_1(t)$, $e_2(t)$; (b) $\Delta\Phi_1(t)$, $\Delta\Phi_2(t)$, $\Delta\Phi_3(t)$; (c) $\Delta\Phi_4(t)$, $\Delta\Phi_5(t)$;

Therefore, we slowed the rate of identification to $K_{a_{ii}}=0.1$ and the results shown in figure 7 with adaptive servo gains show good identification and reasonable tracking in spite of the unknown disturbance. The adaptive gains $K_{11}(t)$ and $K_{22}(t)$ move up-and-down in order to maintain small tracking errors. The system thus combines a slow long-term memory which is not affected by temporary disturbances, and a fast short term adaptive controller to overcome transients and uncertainties. If the disturbance vanishes, the adaptive gains also eventually decrease or vanish, and the control signal will be given almost entirely by the correctly identified parameters.

5. CONCLUSION

A two-degrees-of-freedom robot neurocontroller provides opportunities for noise filtering and disturbance rejection not available with constant gain servo neurocontroller.

REFERENCES

[1] I. Bar-Kana: "Robust Simplified Adaptive Stabilization of Not Necessarily Minimum-Phase Systems," *Trans. ASME, J. Dyn. Syst, Meas., and Control*, Vol. 111, pp. 364-370, 1989.

[2] I. Bar-Kana and A. Guez, "Neuromorphic Adaptive Control," *Proc. 28th Conference on Decision and Control*, Tampa, Florida, 1989, pp. 1739-1743.

[3] J. J. E. Slotine and W. Li., "Adaptive Manipulator Control: A Case Study," *IEEE Transactions on Automatic Control*, Vol. 33, No. 11, pp. 995-1003, 1988.

[4] J. W. Selinsky and A. Guez, "The Role of Apriori Knowledge of Plant Dynamics in Neurocontroller Design," *Proc. 28th Conference on Decision and Control*, Tampa, Florida, 1989, pp. 1754-1758.

[5] B. Widrow and S. Stearn , *Adaptive Signal Processing*, Prentice Hall, 1985.

[6] I. Bar-Kana and A. Guez, "A Two Degree of Freedom Robot Adaptive Controller," *Proc. 1990 ACC*, San Diego, California.

[7] I. Bar-Kana and A. Guez, *Two Degree of Freedom Robot Neorocontroller*, Technical Report, Drexel University.

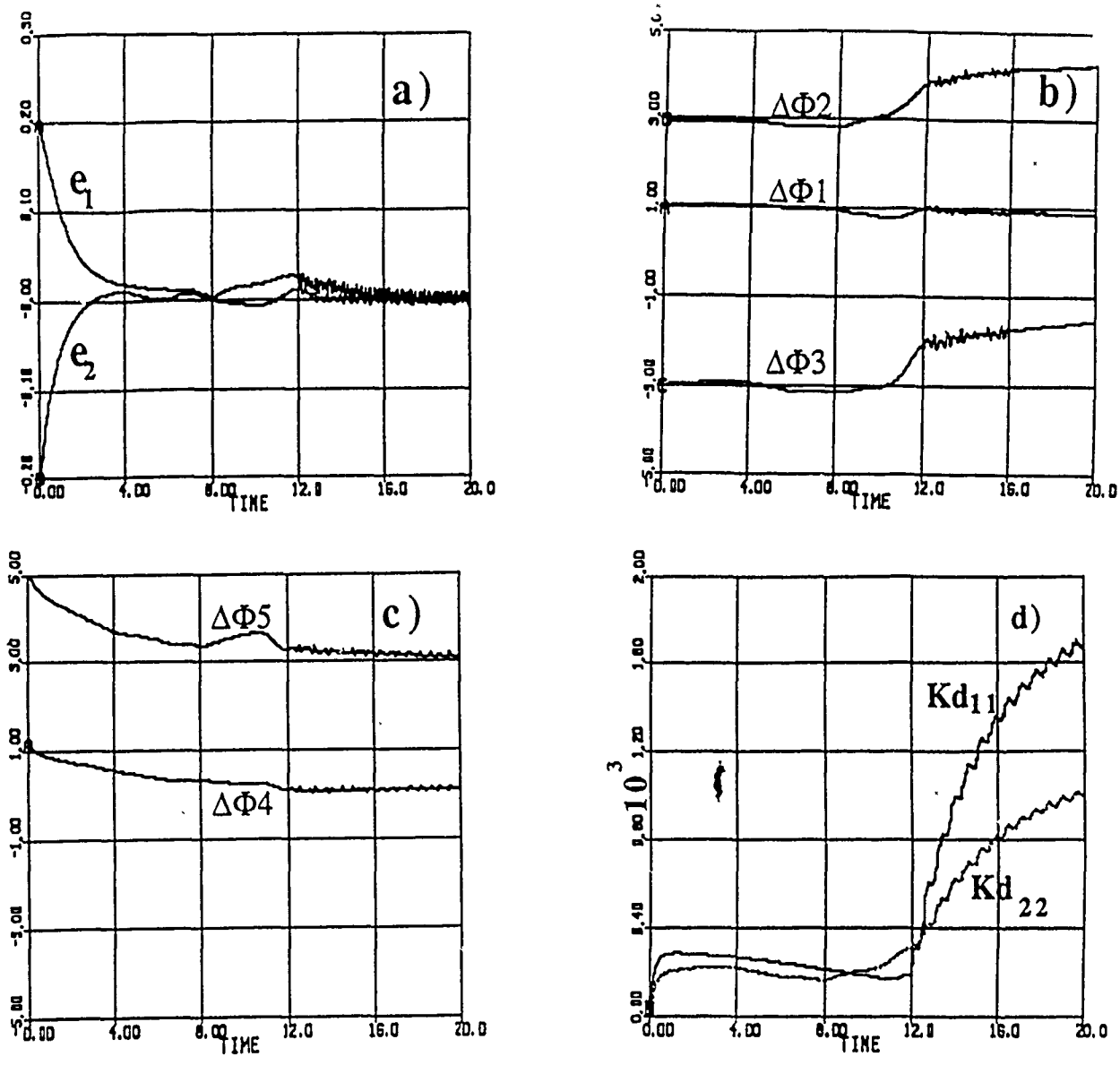


Figure 8. Adaptive servo gains and slow identification:
(a) Tracking Errors $e_1(t)$, $e_2(t)$; (b) $\Delta\Phi_1(t)$, $\Delta\Phi_2(t)$, $\Delta\Phi_3(t)$;
(c) $\Delta\Phi_4(t)$, $\Delta\Phi_5(t)$; (d) Adaptive Controller Gains $K_{d11}(t)$, $K_{d22}(t)$.

Robotics and Automation, B.C. CANADA

(17)

(C|I)

"APPLICATION OF NEURAL NETWORKS TO ROBOTICS" 7/90

[10]

A. GUEZ
ECE DEPT., DREXEL UNIVERSITY
PHILADELPHIA, PA. 19004

ABSTRACT

We apply neural networks computing architecture to the solution of the Inverse Kinematic Problem (IKP), and to the adaptive and learning on line control of robotic manipulators. Our results are encouraging. We obtained a twofold saving in the computaional load of the IKP for a PUMA 560 arm. We also found that global asymptotic tracking and parameters closed loop learning are attainable with neurocontrollers.

1990 AMERICAN CONTROL CONFERENCE
May 23-25, 1990
The Sheraton Harbor Island Hotel
San Diego, California

[11]

TWO DEGREE OF FREEDOM ROBOT ADAPTIVE CONTROLLER

Allon Guez and Izhak Bar-Kana
ECE Dept., Drexel University, Philadelphia, PA 19104

ABSTRACT

This paper combines the simple technique for adaptive robot control proposed by Bar-Kana and Guez and the exploratory leming schedule proposed by Selinsky and Guez into a unified two degrees of freedom (DOF) robot controller, thereby improving the performance of both approaches. We first briefly summarize both approaches. We then propose our new 2 DOF controller and explain our motivation for its design. Next we present the main theorem and conclude with a two-joints rigid manipulator example.

1. INTRODUCTION

In this paper we combine the simple technique for adaptive robot control proposed by Bar-Kana and Guez and the exploratory learning schedule proposed by Selinsky and Guez into a unified two degrees of freedom (DOF) robot controller, thereby improving the performance of both approaches. Both approaches are first summmarized in Chapter 2. The new 2 DOF controller is presented and the motivation for its design is explained in Chapter 3. Chapter 4 presents the main theorem and Chapter 6 conclude with a two-joints rigid manipulator example.

2. THE ADAPTIVE CONTROLLERS

In [1] an unsupervised distributed parallel computing architecture is proposed for the adaptive control of nonlinear dynamic systems of the class

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \qquad (1)$$

$$y(t) = C(x)x(t) + D(x)u(t) \qquad (2)$$

with some degree or other of uncertainty, where $x(t)\in R^n$ is the plant state vector, $y(t)\in R^m$ is the output vector, and $u(t)\in R^m$ is the input command vector, and where A(x), B(x), C(x), and D(x) are uniformly bounded matrices of corresponding dimensions.

The proposed controller is depicted in Figure 1 below: It consists of a teacher model which incorporates the knowledge regarding the desired input/output plant response as well as the repertoire of reference commands that the system may be subjected to.

The teacher dynamic model is assumed to have the following representation:

$$\dot{x}_t(t) = A_t(x_t)x_t(t) + B_t(x_t)u_t(t) \qquad (3)$$

$$y_t(t) = C_t(x_t)x_t(t) + D_t(x_t)u_t(t) \qquad (4)$$

where $x_t(t)\in R^{n_t}$ is the state vector, $y_t(t)\in R^m$ the output vector, and $u_t(t)\in R^m$ is the input command vector, and where $A_t(x_t)$, $B_t(x_t)$, $C_t(x_t)$, and $D_t(x_t)$ are uniformly bounded matrices of corresponding dimensions. It is emphasized that the dimension of the model is unrestricted, except that $\dim(y_t) = \dim(y) = m$.

This paper is based on research supported in part by AFOSR Grant No. 890010 and by Drexel University's Stein Fellowship Foundation.

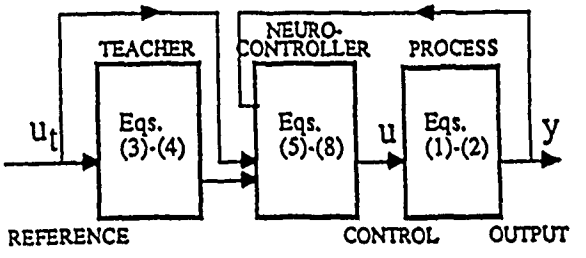


Fig. 1: Neurocontroller Proposed by Bar-Kana and Guez [1]

The parallel distributed adaptive controller receives the input 'features' vector $f(u_t, x_t, y)$ and generates as an output the process control vector u, where

$$u(t) = K(t)f(u_t, x_t, y) \qquad (5)$$

where K(t) is the adaptive gain matrix of appropriate dimension. Each K_{ij} gain monitors the sensitivity of the i-th control loop, namely u_i, to the j-th feature of the system, namely $f_j(u_t, x_t, y)$ and may be viewed as the state of the *ij*-th neuron.

The K_{ij} gain adjusts its value independently of, and simultaneously with all other gains, according to:

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \qquad (6)$$

$$M_{ij}(t) = \alpha_{ij}(y_{t_i} - y_i)f_j \qquad (7)$$

$$\frac{d}{dt}N_{ij}(t) = -\beta_{ij}N_{ij}(t) + \gamma_{ij}(y_{t_i} - y_i)f_j \qquad (8)$$

where α_{ij}, β_{ij} and γ_{ij} are positive constants.

Bar-Kana and Guez show that perfect regulation of the plant may be obtained in the purely deterministic case. Notice however that, while reducing the amount of prior knowledge required to guarantee stability of the adaptive systems, this simple adaptive controller does not take into account any valid knowledge on the specific plant to be controlled. No learning is used to improve performance in future trials or to make the control assignment easier. The initial adaptive gains are always zero, as no better initial value is available, and also the range of the adaptive gains may me unnecessarily large. Indeed the main purpose of this simple algorithm is first of all to reduce the necessary condition to mere stabilizability.

A general block diagram of the system proposed by Selinsky and Guez [2] for a robot manipulator is shown in figure 2. It consists of the Exploratory Schedule Generator, Neurocontroller and associated Learning Algorithm, the Robot, and a mechanism which allows selecting between the User / Path Planner and ES Generator as the originator of the desired trajectory (q_d).

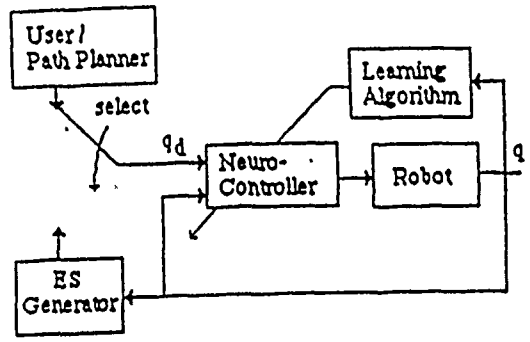


Figure 2: Block diagram of the system proposed by Selinsky and Guez [2]

The robot dynamic model is assumed as follows:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad (9)$$

where

$D(q)$	is an n x n matrix of inertial terms,
$C(q,\dot{q})$	is an n x n matrix of coriollis and centrifugal terms,
$G(q)$	is an n x 1 vector of gravitational terms,
q	is an n x 1 vector of joint coordinates,
τ	is an n x 1 vector of forces/torques,
n	is the number of degrees of freedom.

Equation (9) can also be expressed as

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\dot{q},\ddot{q})\emptyset \qquad (10)$$

where $Y(q,\dot{q},\dot{q},\ddot{q})$ is an n x p matrix of known functions and $\emptyset$ is an p x 1 vector of weighting constants (see Slotine and Li [3]).

When (10) is written in terms of the individual torques at each joint, it can be viewed as a single layer linear network, where the inputs to the network are the $Y_{ij}(.)$ and the weights are the $\emptyset_j$ (See Figure 3).

Note that the $Y_{ij}(.)$ are transcendental algebraic functions of the manipulators states that are a priori known and may be realized via feedforward neural networks that are trained offline with a suitable learning algorithm .

Let q_d, $\dot{q}_d$, $\ddot{q}_d$ designate the desired trajectory. Following Slotine and Li [3] define the virtual reference trajectory $\dot{q}_r$, $\ddot{q}_r$, and the virtual trajectory velocity error $\dot{e}_r$

$$\dot{q}_r = \dot{q}_d - A\,e, \quad \ddot{q}_r = \ddot{q}_d - A\,\dot{e}, \quad \dot{e}_r = \dot{q} - \dot{q}_r = \dot{e} + Ae \qquad (11)$$

where $e[t] = q[t]-q_d[t]$ is the joint coordinate error, and A is a positive definite matrix with constant coefficients.

The output of the neurocontroller implements the control law

$$\tau_i = \sum_{j}^{p} Y_{ij}[q, \dot{q}, \dot{q}_r, \ddot{q}_r]\,\hat{\emptyset}_j + K_{d_{ii}}\,\dot{e}_{r_i}, \quad i = 1,2,...,n \qquad (12)$$

where $\hat{\emptyset}$ denotes the estimate of $\emptyset$, and the $K_{d_{ii}}$ are constant weights for the servo portion of the controller . Equation (12) has been shown to be asymptotically stable when the learning rule

$$\dot{\hat{\emptyset}}_j = -\sum_{i=1}^{n} \frac{1}{K_{a_{ij}}} Y_{ij}[q, \dot{q}, \dot{q}_r, \ddot{q}_r]\,\dot{e}_{r_i}, \quad j = 1,2,..., p \qquad (13)$$

is used (Slotine and Li [3]).

Notice in equation (13) the similarity to the LMS learning rule (see Widrow and Steam [4]) where the weight change is proportional to the error ε and the input features X. In equation (13) the input features are the $Y_{ij}(.)$ functions and the error is $\dot{e}_{r_i}$.

Figure 3 shows the internal structure of the neurocontroller for joint i. Each feedforward network module is trained to provide one of the $Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]$ functions. This training can be done offline since the $Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]$ functions are known a priori and are the same for all rigid robots of the same kinematics and number of degrees of freedom. The inputs to the neurocontroller are the components of the trajectories as required. The outputs of the neurocontroller are the control torques to be applied at each joint of the manipulator.

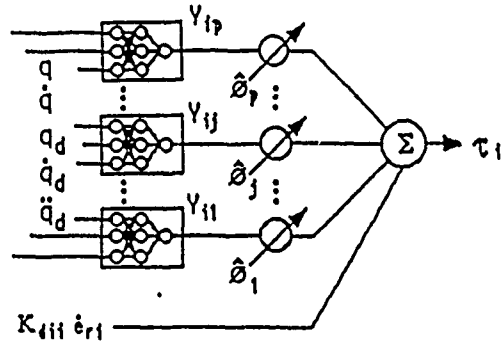


Figure 3: Neurocontroller structure for joint i.

The main result in [2] is that If $p \le n$, and if

$$\text{Rank} \left\{ \begin{matrix} Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d) \\ \lim\ t \to \infty \end{matrix} \right\} = p$$

then the controller (eq. (12), (13)) guarantees global asymptotic tracking of the desired trajectory and identification of the weights.

3. THE TWO DEGREE OF FREEDOM ADAPTIVE CONTROLLER

Selinsky and Guez indeed use learning in the adaptation process. Their exploratory schedules also guarantee that along with tracking, the unknown plant parameters are finally identified. Yet, in order to get satisfactory results for both tracking and learning, the supplementary gains $K_{d_{ii}}$ are needed in (12). The results of both tracking and learning are strongly affected by the right selection of these parameters [3], and in practice they are fixed by trial and error procedures. Furthermore, if one selects high fixed gains to get small tracking errors, one then also gets high noise amplification and possibly high cost of control even when not necessarily needed. An adaptive procedure for $K_{d_{ii}}$ like in Bar-Kana and Guez, will fix the gains as a function of the tracking error (in our case represented by $\dot{e}_r$) tends to increase, and decrease afterwards, fitting thus the right control gains to the right situation.

In this article we propose the 2 DOF controller shown in Fig. 4.

The robot dynamic model is given by equation (9), the identification and linearization neurocontroller algorithm is given by equations (11), (12) and (13) and the adaptive servo neurocontroller algorithm has the form of equations (5), (6), (7) and (8) but for notation compatibility it is

redefined as

$$K_{d_{ij}}(t) = K_{dI_{ij}}(t) + K_{dP_{ij}}(t) \quad (14)$$

$$K_{dP_{ij}}(t) = \alpha_{ij}\dot{e}_i\dot{e}_r \quad (15)$$

$$\dot{K}_{dI_{ij}}(t) = -\beta_{ij}N_{ij}(t) + \gamma_{ij}\dot{e}_i\dot{e}_r \quad (16)$$

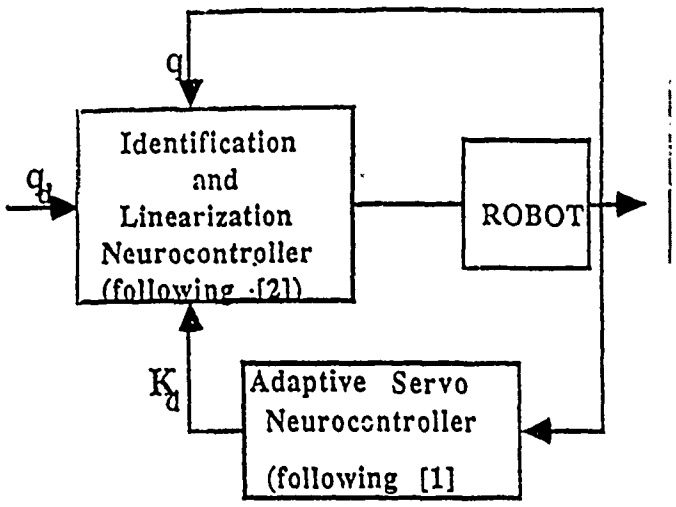


Fig. 4: Two Degrees of Freedom Robot Controller.

Main Result. It can be shown [5] that global asymptotic tracking and identification as in [2] still holds even for the 2 DOF controller, i. e., when the servo gain is time varying according to equation (14). That is, global asymptotic tracking as well as global asymptotic identification of up to n robot parameters can be guaranteed with our 2 DOF robot controller if

$$\lim_{t\to\infty} [\text{Rank}\,(Y(q, \dot{q}, \dot{q}_d, \ddot{q}_d))] = p. \quad (17)$$

4. ROBOTIC EXAMPLE

We simulated the exact system as in [2] with the new 2 DOF controller as follows:

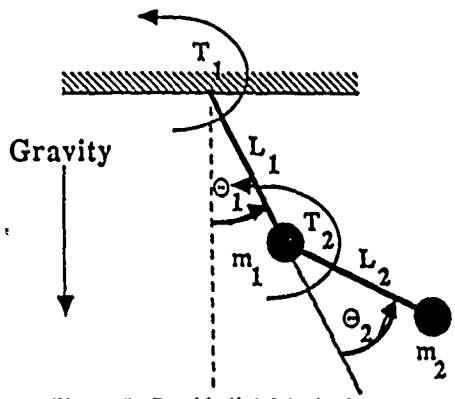


Figure 5 - Double link Manipulator

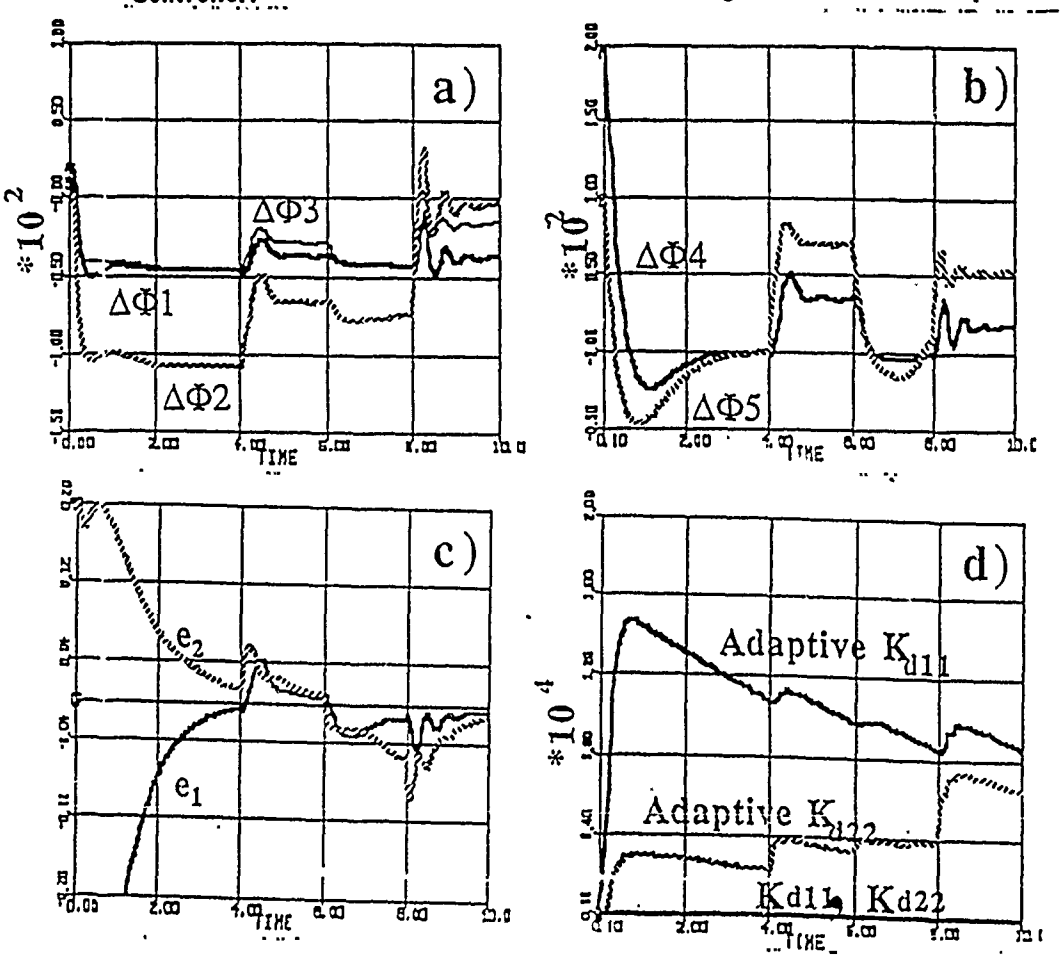


Figure 6. Fixed Gains: (a) $\Delta\Phi_1(t)$, $\Delta\Phi_2(t)$, $\Delta\Phi_3(t)$; (b) $\Delta\Phi_4(t)$, $\Delta\Phi_5(t)$; (c) Tracking Errors $e_1(t)$, $e_2(t)$; (d) Controller Gains $K_{d_{11}}$, $K_{d_{22}}$, $K_{d_{11}}(t)$, $K_{d_{22}}(t)$.

Figure 5 describes the double link system configuration. The equations of motion are:

$$\tau_1 = D_{11}\ddot{q}_1 + D_{12}\ddot{q}_2 + D(\dot{q}_2^2 + 2\dot{q}_1\dot{q}_2) + D_1$$

$$\tau_2 = D_{12}\ddot{q}_1 + D_{22}\ddot{q}_2 - D\dot{q}_1^2 + D_2$$

where

$L_1 = L_2 = 1.0$ m, $m_1 = m_2 = 10$ kg, $g = 9.81$ m/s^2,

$D_{11} = (m_1 + m_2)L_1^2 + m_2L_2^2 + 2m_2L_1L_2\cos q_2$,

$D_{12} = m_2L_2^2 + m_2L_1L_2\cos q_2$,

$D_{22} = m_2L_2^2$,

$D = -m_2L_1L_2\sin q_2$,

$D_1 = (m_1 + m_2)gL_1\sin q_1 + m_2gL_2\sin(q_1 + q_2)$,

$D_2 = m_2gL_2\sin(q_1 + q_2)$.

and where g is gravity, τ_1 and τ_2 are torques at the first and second joints, m_1 and m_2 are masses, and L_1 and L_2 are lengths of the first and second link correspondingly.

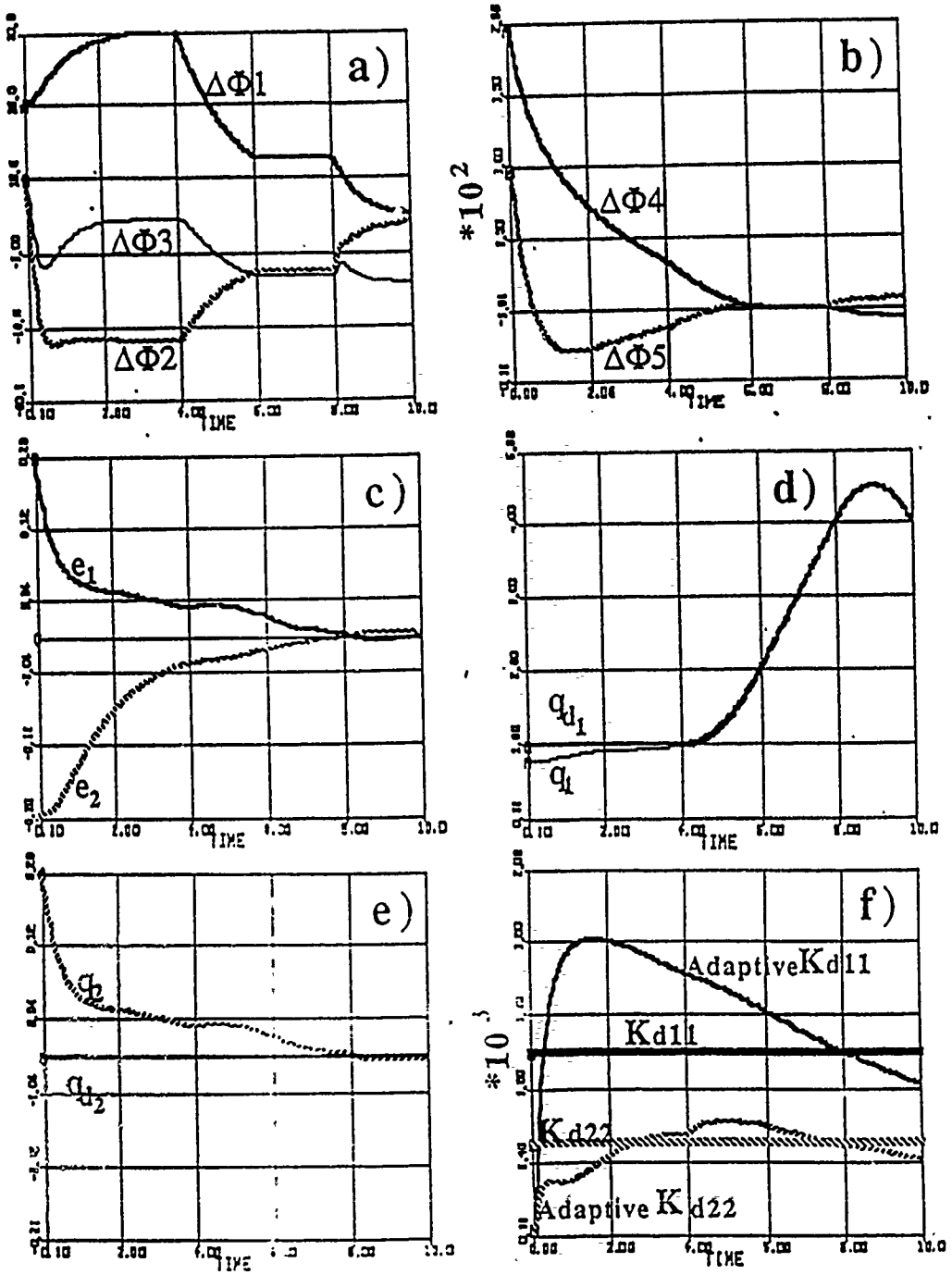


Figure 7. Fixed gains: (a) $\Delta\Phi_1(t)$, $\Delta\Phi_2(t)$, $\Delta\Phi_3(t)$; (b) $\Delta\Phi_4(t)$, $\Delta\Phi_5(t)$;
(c) Tracking Errors $e_1(t)$, $e_2(t)$;(d) Trajectories $q_{d_1}(t)$, $q_1(t)$;
(e) Trajectories $q_{d_2}(t)$, $q_2(t)$; (f) Controller Gains K_{d11}, K_{d22}, $K_{d11}(t)$, $K_{d22}(t)$.

As already mentioned, the performance of the adaptive ro ɔtic control system is strongly affected by the selection of th constant control gains $K_{d_{ii}}$. Results of simulations w ı constant gains $K_{d_{11}} = 100.$, $K_{d_{22}} = 50.$, are shown in Fi ure 6. Figure 6a shows the identification error for the fi t three parameters, figure 6b shows the error for the re aining parameters, and figure 6c represents the tracking er rs. It can be seen that neither the parameters are very w l identified, nor the tracking is very good. In parallel with th closed-loop system we compute (in open-loop) and show in igure 6d, for illustration, the constant gains $K_{d_{11}} = 100$, K :2 = 50, and the corresponding adaptive gains $K_{d_{11}}(t)$ and K :2(t) that would result from these tracking errors. It can b seen that adaptive gains corresponding to such errors w ıld be very large. We can expect that in closed-loop, the la ;er adaptive gains would actually reduce the tracking er rs.

Results of simulations with a better selection of constant g: ıs $K_{d_{11}} = 1000.$, $K_{d_{22}} = 500.$, are shown in Figure 7. Figure 7a shows the identification error for the first three parameters, and figure 7b shows the error for the remaining parameters. It can be seen that all parameters are eventually identified within ± 5m. Figure 7c represents the tracking errors, and in 7d and 7e we show the desired and the actual trajectories, correspondingly. It can be seen that results of both identification and tracking are satisfactory. Figure 7f shows the constant gains $K_{d_{11}} = 1000$, $K_{d_{22}} = 500$, and the corresponding adaptive gains $K_{d_{11}}(t)$ and $K_{d_{22}}(t)$. It is interesting to see that the adaptive gains reach similar values with the "good" constant gains, in order to maintain small tracking errors.

Results of simulations with adaptive control gains in closed-loop are shown in Figure 8. Figure 8a shows the identification error for the first three parameters, and figure 8b shows the error for the remaining parameters. It can be seen again that all parameters are eventually identified within ± 5m. Figure 8c represents the small tracking errors, and figure 8d shows the behavior of the adaptive gains $K_{11}(t)$ and $K_{22}(t)$. It can be seen how the adaptive gain moves up-and-down in order to maintain small tracking errors.

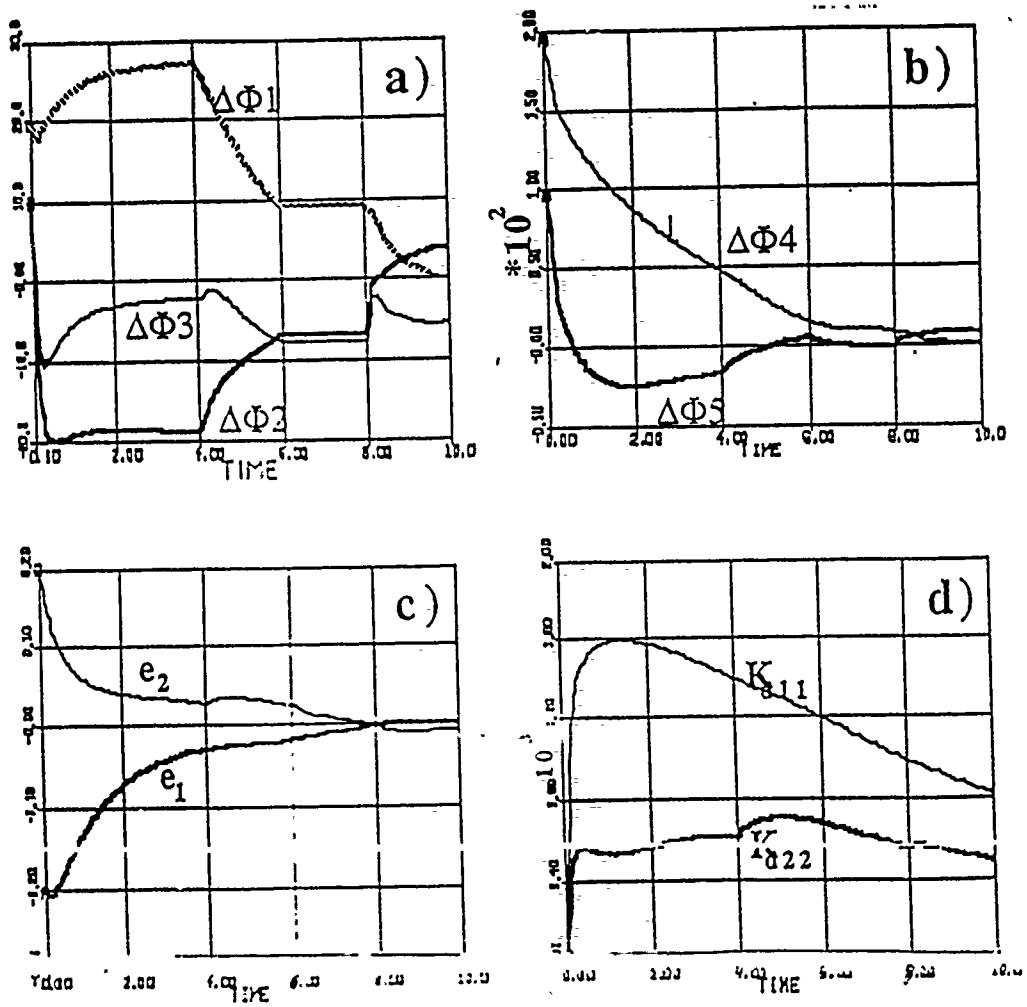


Figure 8. Example: (a) $\Delta\Phi_1(t)$, $\Delta\Phi_2(t)$, $\Delta\Phi_3(t)$; (b) $\Delta\Phi_4(t)$, $\Delta\Phi_5(t)$; (c) Tracking Errors $e_1(t)$, $e_2(t)$; (d) Adaptive Controller Gains $K_{d11}(t)$, $K_{d22}(t)$.

5. CONCLUSION

This paper combines a simple technique for adaptive robot control proposed and an exploratory learning schedule into a unified two degrees of freedom (DOF) robot controller, thereby improving the performance of both approaches. The power of the new controller is tested with a two-joints rigid manipulator example.

REFERENCES

[1] I. Bar-Kana and A. Guez, "Neuromorphic Adaptive Control," *Proc. 28th Conference on Decision and Control*, Tampa, Florida, 1989, pp. 1739-1743.

[2] J. W. Selinsky and A. Guez, "The Role of Apriori Knowledge of Plant Dynamics in Neurocontroller Design," *Proc. 28th Conference on Decision and Control*, Tampa, Florida, 1989, pp. 1754-1758.

[3] J. J. E. Slotine and W. Li., "Adaptive Manipulator Control: A Case Study," *IEEE Transactions on Automatic Control*, Vol. 33, No. 11, pp. 995-1003, 1988.

[4] B. Widrow and S. Stearn , *Adaptive Signal Processing*, Prentice Hall, 1985.

[5] I. Bar-Kana and A. Guez, *A Two Degree Of Freedom Adaptive Controller*, Technical Report, Drexel University.

[12]

"APPLICATION OF NEURAL NETWORKS TO ROBOTICS"

A. GUEZ
ECE DEPT., DREXEL UNIVERSITY
PHILADELPHIA, PA. 19004

ABSTRACT

We apply neural networks computing architecture to the solution of the Inverse Kinematic Problem (IKP), and to the adaptive and learning on line control of robotic manipulators. Our results are encouraging. We obtained a twofold saving in the computaional load of the IKP for a PUMA 560 arm. We also found that global asymptotic tracking and parameters closed loop learning are attainable with neurocontrollers.

[13]

On the Solution to the Inverse Kinematic Problem

Ziauddin Ahmad Allon Guez

Department of Electrical and Computer Engineering
Drexel University, Philadelphia PA 19104

Abstract

A nonalgorithmic method is presented for the solution to the inverse kinematic problem of a robot. The method is robot independent and involves a hybrid approach, whereby a neural solution is augumented with an iterative procedure which provides the final solution within some specified tolerance. Essentially the neural solution is similar to a lookup table in providing a good initial guess to a classical iterative search. It has been found that for the industrial manipulator PUMA 560 the proposed hybrid method achieves about 2-fold increase in computational efficiency with better uniformity of the time required to obtain the solution to the robotic manipulator.

1. Introduction and Background

The inverse kinematic problem (IKP) deals with finding the 'n' joint angle values 'q' of the robot that will position the end-effector in a desired position and orientation 'X' in the 'm' dimensional workspace. This may be expressed as:

$$q = f^{-1}(X) \qquad (1)$$

However, in general this solution is not unique. In many cases (e.g. redundant manipulators) there may result infinite number of solutions. In these cases, additional constraints [15] in terms of the allowed configurations or performance-function minimization are used to reduce the number of legitimate joint configurations or to single out a unique preferable one (see example 1 below). Therefore, where redundancy provides us with more flexibility it also requires elaborate solution techniques which are time consuming and generally impractical. The conventional methods consist of closed form methods and iterative methods. These are either limited to only a class of simple non-redundant robots or are time consuming and the solution may diverge due to bad initial guess [3].

In a class of neural networks (NN) called Feedforward Networks the processing elements, termed as nodes indicated by circles in Fig. 1, are connected in layers through links, termed as weights indicated by arrows in Fig. 1.

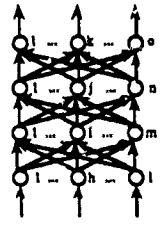


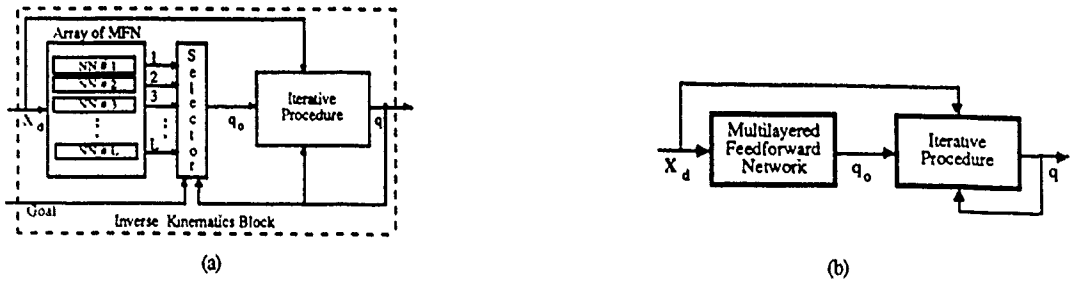
Fig. 1 : Feedfoward Neural Network.

The output of the node is a function of the inputs, which are weighted outputs of the nodes of the previous layer, and the threshold of the node. The learning takes place through the modification of the weights and the thresholds as specified by the training algorithm that acts on the supplied input and output data pairs as the training set. The training algorithm used in our simulations is the Back Error Propagation (BEP) Algorithm [12]. The nodes to which the input is applied are called as the input nodes and the nodes from which the output is taken are called as the output nodes. The remaining nodes are termed as hidden nodes.

The solution to the IKP has been attempted by different approaches by a number of researchers including [2],[7],[8],[9],[10]. Here we address the problems in conventional methods and suggest a combination of the neural network with a conventional method to aid in eradicating them. Previous work by us in this direction employing neural networks only [7] yielded good results but were not accurate enough to be practically utilized. However, these results encouraged us to extend this approach, as presented here, which is nonalgorithmic and, therefore, not specific to any single manipulator.

Work Partially supported by AFOSR grant 890100.

CH2876-1/90/0000/1692$01.00 © 1990 IEEE



(a) (b)

Fig. 2 Schematics of the proposed method employing the NN to provide the initial guess q_0 to the iterative procedure.

2. The Hybrid Method

The formation of proposed method starts with the robotic manipulator for which the IKP is to be solved and a 'blank' (untrained) NN of suitable size. The selection of the size is empirical in nature. Then training data in the form of pairs of the end-effector position and orientation and the corresponding joint values is generated. This data is used for training the NN with the position and orientation vector as the input and the corresponding joint values as the desired output. After the training is completed the trained NN is coupled with the iterative method, as shown in the Fig. 2(b), for the purpose of operation. During the operation phase the desired position and orientation of the end-effector is provided to the NN. The NN gives the approximate solution based on the learned connection weights. This approximate solution is taken as the initial guess by the iterative method to give the final solution within the specified tolerance. We consider that the type of solution out of the finitely many solutions is pre-specified to us and therefore training of the NN is restricted to the set of examples that pertain to this specific solution only.

3. Examples

The back error propagation (BEP) algorithm simulating a three layer perceptron was employed to tackle the problems described below. Continuous inputs and outputs were assumed. The nodes assumed the symmetric sigmoidal nonlinearity [13]. Parameters of the NN are as given in reference [1]. Training was terminated when it was seen that the errors were not improving.

A 3-DOF Planar Robot

In a plane it is clear that, for the purpose of positioning only, two DOF are sufficient. Therefore, a 3-DOF planar manipulator, for this purpose, has one redundant DOF, see Fig. (3). This extra DOF introduces a problem of infinitely many joint configurations resulting in the same end-effector position. This problem can be resolved by adding an additional independent constraint to be fulfilled in addition to the positioning of the end-effector.

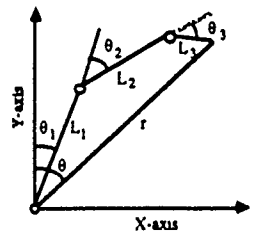


Fig 3: A three DOF planar Manipulator

The performance criterion optimized for the 3-DOF planar manipulator was the "Manipulability Index" [15], defined as:

$$H = \sqrt{|JJ^T|} \qquad (2)$$

where J is the Jacobian of the manipulator. Solution of this manipulator was obtained by iteratively solving, by Newton-Raphson method, the equations obtained by the procedure outilined in reference [4].

The network for this case constituted of 2 inputs and 3 output nodes and two hidden layers, each containing 10 nodes. The network was trained on a data set giving the X-Y coordinates for the end-effector and the three joint angles that optimized the manipulability criterion. Single inverse in the training set was ascertained by taking the initial condition as the current configuration for the arm solution to a subsequent nearby end-effector position.

The result of this training is shown in Fig. 4. This figure shows the positioning error in the cylindrical coordinate workspace. It is seen that the error begins to increase as the manipulator moves nearer to the singularity. In this case singularity occurs at $\theta_2 = \theta_3 = 0$ radians.

When the NN solution was coupled with the Newton-Raphson iterative procedure, with a tolerance of 10^{-6} meter in the end-effector positioning, it resulted in nearly a 2-fold increase in computational efficiency. Table 1 gives a comparison

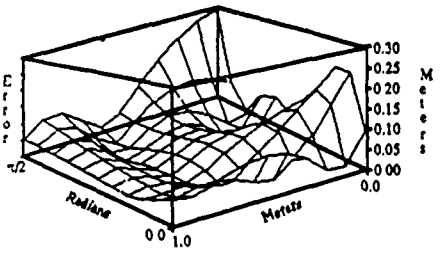
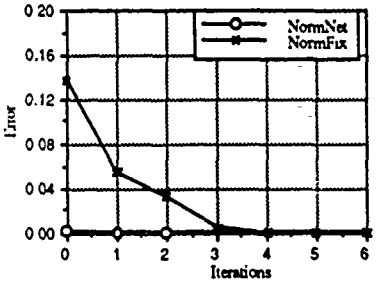


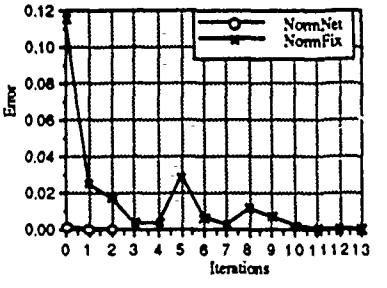
Fig. 4: Result of a NN trained to give a manipulability index optimized solution for a 3-DOF planar manipulator. The verticle axis corresponds to error in positioning of the end-effector.

Table 1: Comparison of the proposed solution with the Fixed initial guess Newton Raphson Method, for a 3-DOF manipulator. The data corresponds to 100 Samples.

Initial guess for Newton-Raphson Method	Number of Iterations	
	Average	Standard Deviation
Neural solution	2.93	0.381
Fixed	5.95	4.01



(a)



(b)

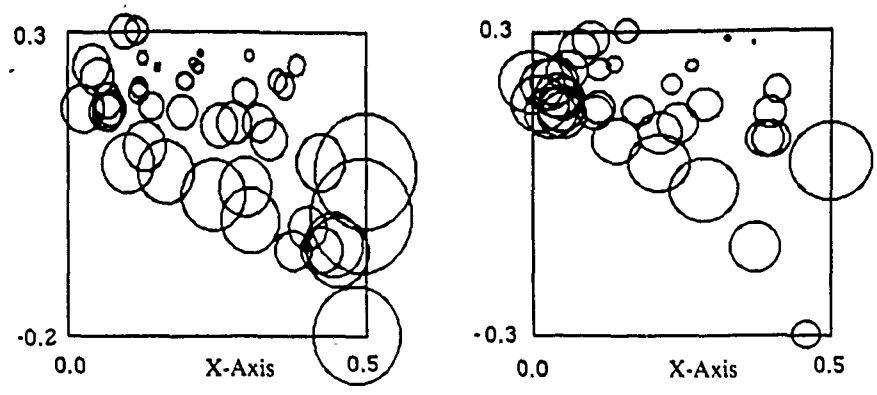
Fig. 5: Iteration sequences showing a comparison for the convergence of the Newton-Raphson method for the initial guess given by the neural network and the fixed initial guess.

between the number of iterations required for the iterative method with fixed initial guess and the initial guess given by the NN. The fixed initial guess was taken as $\theta_1=0^o$, $\theta_2=90^o$, $\theta_3=5^o$. It was found that, on the average, the iterative method alone with fixed initial guess required 6.0 iterations to give the arm solution, whereas the iterative method that utilized the solution given by the NN as the initial guess required on the average 2.9 iterations. Moreover, the standard deviation of the number of iterations for the fixed initial guess is 10 times that for the NN initial guess. Fig. 5 shows the iteration sequences for the fixed initial guess and the initial guess given by the NN for two situations. Fig. 5(a) is one of the most typical situations. From Fig 5(b) we see that sometimes the number of iterations may become large for the fixed initial guess

The Human Arm

Here we show an attempt to capture the criteria that a human being allegedly optimizes in manipulating different objects by training the NN by a data set corresponding to some specified task. Planar motion, parallel to the ground was the considered task. The subject was asked to move an object in free space, in a plane parallel to the ground. Knowing the actual distances between the joints the data set was filtered to achieve a 10.0% tolerance about the respective actual values. The data set thus obtained contained only 43 words out of a total of 78 words. A NN similar to that for the 3-DOF case was trained on this data set.

Fig. 6(a) shows the plot of the error in the positioning of the hand resulting for the trained data set, while Fig. 6(b) shows the same for a different data set obtained separately from the data set on which the network was trained. The diameter of the circles indicate the magnitude of error occured at their center. The two figures have similar errors indicating that the neural network has generalized on the trained data set. Large errors near X = 0.5 m are perhaps due to the singularity reasons or insufficient data near that region. Further, it was observed that the values for the elbow joint's were learned much better than those for the wrist joint and the shoulder joint.



(a) Hand positioning error for the training data.

(b) Hand positioning error for other than the training data.

Fig.6: Results of single MFN trained on human arm data in a plane perpendicular to the gravity. Shoulder is located at (0,0).

As seen earlier, the filtered data set was only 55% of the total data gathered with 10% tolerance which indicates that the precision of the training data may not be adequate. However, it is observed that the NN is able to generalize upon the training data giving similar results for the untrained data to that of the trained data implying that implicit performance indices can be captured via NNs and perhaps identified via weight pruning and analysis.

The PUMA 560

The PUMA 560 parameters were taken from [6], page 37. This manipulator, which is a 6-DOF robot, was chosen for ease of generation of data and verification of results since it has a closed form solution. PUMA 560 has eight solutions for a given position and orientation signified by Right/Left - Shoulder, Above/Below - Elbow, and Up/Down - Wrist. In our simulations the training data corresponds to: LEFT Arm, BELOW Elbow and UP Wrist configuration. In the simulations the joint limits used for the 6th joint were -180° to 180° instead of -266° to 266°.

The network in this case consisted of 6 input nodes, one output node each for the 6 joints and two hidden layers for each joint consisting of 32 nodes in the first layer and 8 nodes in the second layer. The average error and the standard deviation of the error in the solution given by the NN for each joint taken over 100 samples is given in table 2. From table 2 it is seen that the solution given by NNs is more scattered for Joints 4 to 6 as compared with the joints 1 to 3.

Next, the proposed method was compared by giving a fixed estimate to the iterative procedure. This Fixed Estimate was taken as: $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = \pi/4$, $\theta_4 = 0$, $\theta_5 = \pi/4$, and $\theta_6 = 0$, which is a configuration corresponding to which the NN was trained, as indicated in the beginning of this section. In the simulations the equations were solved by Gauss Elimination method and partial pivoting. The maximum number of iterations allowed for the iterative method were 100. The iterative method was successfully terminated when the norm of the difference between the desired and actual end-effector position and orientation was less than 1.0E-4. The average and standard deviation for the number of iterations for the proposed method and the Fixed Estimator, in a run of 100 data points is given in Table 3. From this table we can see that the proposed method achieves more than a two-fold efficiency in computing with more consistency. Moreover, it was observed that the time taken by the NN equals two time units of the iterative procedure, which amounts to less than 10% of the time required to get the solution by the Fixed Estimator method.

Fig. 7 shows the comparison plots for two cases indicating the sequence of Newton Raphson iterations for the fixed initial guess and the guess given by the NN. Here the error along the ordinate axis corresponds to the norm of the error, defined as:

$$\text{Error} = \sqrt{\varepsilon_x^2+\varepsilon_y^2+\varepsilon_z^2+\varepsilon_\phi^2+\varepsilon_\theta^2+\varepsilon_\psi^2} \qquad (3)$$

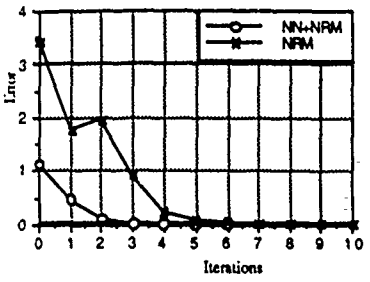
where ε is the difference between the desired and the actual values and the subscripts indicate the variable for which this difference is taken. The allowed end-effector position and orientation error in this case was 10^{-7}. These are typical cases and we can see form these that the number of iterations required for the proposed method has been decreased by two-fold.

Table 2: Statistics of the solution given by the NN.

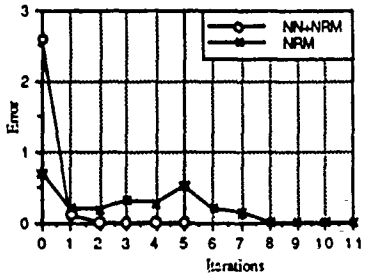
Joint #	Error in Degrees	
	Average	Standard Deviation
1	4.06	13.428
2	-0.16	30.492
3	5.64	11.52
4	3.66	61.236
5	-3.70	24.912
6	-6.02	36.828

Table 3: Comparison of the proposed solution with the Fixed initial guess Newton Raphson Method, for PUMA-560 manipulator. The data corresponds to 100 Samples.

Initial guess for Newton-Raphson Method	Number of Iterations	
	Average	Standard Deviation
Neural solution	9.96	12.95
Fixed	21.09	18.90



(a)



(b)

Fig. 7 : Sequence of Newton Raphson iterations for the guess given by the NN and the fixed initial guess.

4. Discussion

The proposed hybrid method, which takes the solution given by the trained NN as an initial guess to an iterative procedure (Newton-Raphson in our case) combines the advantages of the NN and iterative methods, these being (1) independent of the type of the manipulator, (2) simple to implement. Only forward kinematics is required for this method and, as shown by our simulations this combination results in an increase in computational efficiency by 2-fold for a 3-DOF planar manipulator and the PUMA 560 (6-DOF) robot. It is observed that the efficiency increases with the increase in the size (the number of nodes in the hidden layers) of the network for that manipulator. Therefore, a trade-off between the size of the network / the training time and the accuracy of the solution given by the NN exists and can be employed in order to increase the efficiency of the method or decrease the training time. Where otherwise the number of iterations may be unacceptable for real time operation, an initial good guess given by the neural network consistently cuts the number of iterations to only a very few allowing a better operation of the manipulator. This results in minimal processing within each control cycle and improves real time control performance. Basically the NN operates as a lookup table in providing a good initial guess to an iterative procedure. However, when the NN is implemented in hardware, after the initial off-line training, the adaption of the weights and therefore the learning of the inverse kinematics can be continued on-line. This will continue to decrease the error of the solution given by the NN. Therefore, the number of iterations required by the iterative method would approach zero.

References

[1] Ziauddin Ahmad, Fast Solution to the Inverse Kinematic Problem in Robotics via Multilayered Feedforward Networks, MS Thesis, ECE department, Drexel University, June 1989.

[2] Jacob Barhen, Sandeep Gulati, Michail Zak, "Neural Learning of Constrained Nonlinear Transformations," *IEEE Computer*, 67-76, June 1989.

[3] B. Benhabib, A.A. Goldenberg, and R.G. Fenton, "A Solution to the Inverse Kinematics of Redundant Manipulators," *Journal of Robotic Systems*, 2(4), 373-385, 1985.

[4] Pyung H. Chang, "A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy," *IEEE Journal of Robotics and Automation*,Vol. RA-3, No. 5, 393-403, October 1987.

[5] John J. Craig, Introduction to Robotics Mechanics & Control, Reading MA , Adison-Wesley Publishing Company.

[6] K.S. Fu, R.C. Gonzalez, and C.S.G. Lee, Robotics Control, Sensing, Vision, and intelligence. New York, NY , McGraw-Hill Book Company, 1987.

[7] Allon Guez, and Ziauddin Ahmad, "Solution to the Inverse Kinematics Problem in Robotics by Neural Networks," *IEEE International Conference on Neural Networks*, Vol-II, 617-624, July 1988.

[8] Allon Guez, and Ziauddin Ahmad, "Accelerated Convergence in the Inverse Kinematics via Multilayer Feedforward Networks," *IEEE IJCNN*, Vol-II, 341-344, June 18-22, 1989.

[9] Michael I. Jordan, "Supervised learning and systems with excess degrees of freedom," COINS Technical Report 88-27, MIT. May 1988.

[10] Bartlett W. Mel, "MURPHY: A Robot that Learns by Doing," Report No. UIUCDCS-R-88-1397, Dept. of CS, Univ. of Illinois at Urbana-Champaign, Jan. 1988

[11] Richard P. Paul, Robot Manipulators: Mathematics, Programming, and Control.Cambridge, MA , MIT Press, 1981.

[12] David E. Rumelhart, James L. McClelland, and the PDP Research Group, Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Vol. 1 and 2, Cambridge, MA , MIT Press, 1986.

[13] W. Scott, and B.A. Huberman, "An Improved Three-Layer, Back Propagation Algorithm", *IEEE First International Conference on Neural Networks*, Vol-II, 637-643, June 1987.

[14] Bernard Widrow, Samuel D. Stearns, "Adaptive Signal Processing," Englewood Cliffs, N.J. 07632, Prentice-Hall, Inc., 1985.

[15] Tsuneo Yoshikawa, "Manipulability of Robotic Mechanisms," *International J of Robotics Research*, vol. 4, No. 2, 3-9, Summer 1985.

ADAPTIVE POLE PLACEMENT FOR NEUROCONTROL

Sanjay S. Kumar and Allon Guez
Dept. of Electrical & Computer Engineering,
Drexel University, Philadelphia, PA 19104

Abstract : In this paper we deal with the study of a self organizing neural network architecture called the *Adaptive Resonance Theory* (ART-II) [Grossberg and Carpenter 1987(a)], in its application to a simple problem of adaptive control, through real time dynamic system identification. An adaptive pole placement controller for a linear second order system is implemented based upon this architecture to assess the performance of the network and the overall control scheme with the neural network in the control loop. The network employed demonstrates the capabilities of *fast classification* and *learning*. These attributes of the architecture are exploited to train a network to dynamically identify parametric variations of a plant in response to changes in its environment. The control strategy is based upon identification of changes in the time response characteristics of the system to standard test signals which are assessed by the neural network. A pole placement algorithm is utilized to relocate the poles of the overall closed loop system by altering the gains of the process controller to obtain desired system response. Experimental studies on a simulated second order system, employing a Proportional Derivative controller show that the neural network considered is indeed capable of system identification and simple indirect adaptive control of low order plants that are subjected to parametric variations reflected by changes in their operating environments.

Problem Statement : Our goal is to implement an adaptive pole placement algorithm using a neural network architecture. Let the possibly slow time varying linear dynamical system (Plant) G_p, be given as in equation (1).

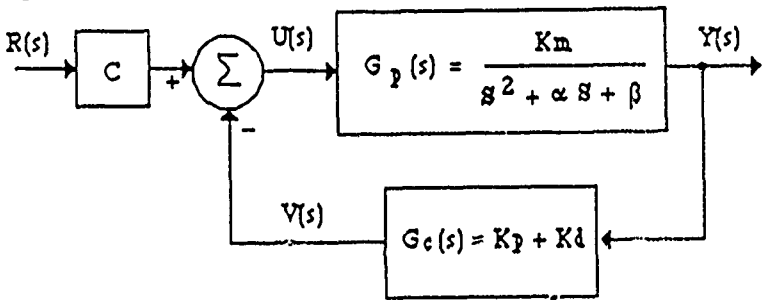


Figure 1 Process block diagram

Plant transfer function : $$Gp(s) = \frac{Km}{s^2 + \alpha s + \beta} \quad (1)$$

Let the reference input be a known periodic function

Reference input: $$r(t) = r(t + T), \text{ for all } t > 0 \quad (2)$$

Find online, the Proportional and Derivative (PD) gains, (Kp, Kd) and the D.C. bias (C) (see Figure 1), such that the control law give by equation (3),

Control law : $$u = C\,r - K_p y - K_d \dot{y} = C\,r - G_c y \quad (3)$$

will result in the ideal closed loop transfer function

$$G_o^*(S) = \frac{Y(s)}{R(s)} = K^* \left[\frac{\omega_n^{*2}}{s^2 + 2\zeta^* \omega_n^* s + \omega_n^{*2}} \right] = G_o \quad (4)$$

where, K^*, ζ, ω^*_n are respectively the desired D.C. gain, damping coefficient and the natural frequency of the system.

Neuromorphic approach to the controller design : The general block diagram of the overall control scheme is depicted in figure 3. The common a priori assumption in the design of controllers for partially known processes is adopted with the design procedure being divided into two steps: *identification* and *control* (indirect adaptive control strategy) [Astrom & Eykhoff,1971].

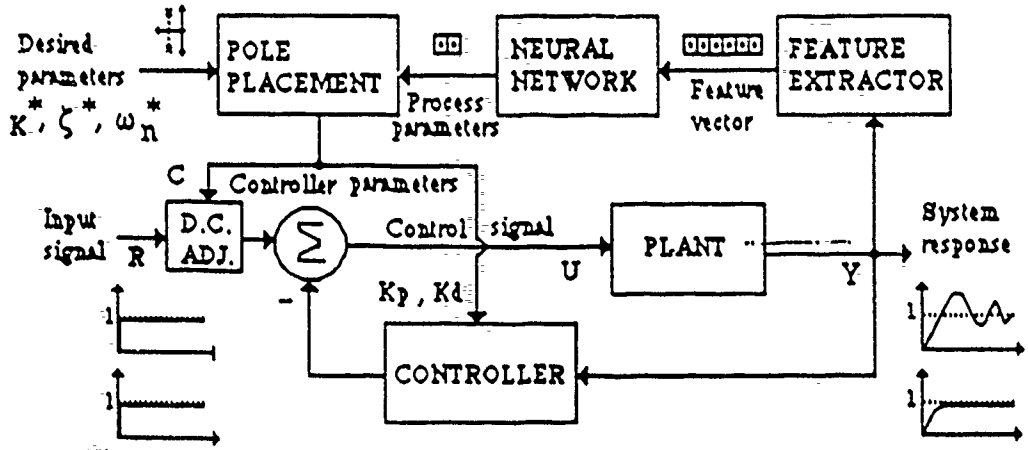


Figure 2 Block diagram of the neuromorphic adaptive control scheme

Objective : Given a second order plant with unknown constant or time varying parameters, the objective of the control scheme is to tune the controller by modifying its arbitrarily assigned initial gains such that the overall closed-loop system response matches the one given by the assumed ideal model, based upon parameter identification provided by the neural network in the control loop. It should be noted that although the plant and the controller are linear the process is overall nonlinear.

Methodology : When the input is a unit step function, identification of the plant parameters is achieved by extracting the features of the system's closed-loop transient time response via a feature extractor. In the case of the square function and the square wave function, identification is restricted to the step portion of the input signal. The neural network in the control loop which is trained to map the features of a system's time response to its parameters, gives the current parameter estimates of the closed loop process. The pole placement algorithm in the loop utilizes the estimates provided by the network to compute new controller gains to suit the desired response specified by the ideal model. A D.C. adjustment mechanism is incorporated to compensate for any D.C. bias that might be associated with the response. The system identifier comprises of the feature extractor, the neural network and the pole placement algorithm. The purpose of the identifier is to determine the plant parameters in response to changes within the plant environment. The time response of the system is characterized by its *features* or *performance indices* which are nonlinear functions of the system parameters.These include the rise time, the settling time, overshoot and the delay time of the response. A *feature extractor* incorporated in the control loop determines the performance indices associated with the response to enable system identification via the neural network. It must be noted that identification is dependent on the features of the time response rather than the response per se. This is done in order to compress the information contained in the response such that the input vector to the neural network remains compact [Kumar & Guez,1989]. This procedure restricts the dimension of the neural network to a minimum thereby increasing its computational speed and overall efficiency of the process. The overall transfer function is obtained by using equations (1),(2) and (3) as follows:

$$Y(S) = G_p(S)\,U(S) = G_p(S)\,[C\,R(S) - G_c(S)\,Y(S)] \qquad (5)$$

$$G_o(s) = \frac{Y(s)}{R(s)} = \frac{C\,G_p(s)}{1 + G_p(s)\,G_c(s)} = \frac{CK_m}{s^2 + (\alpha + K_m K_d)\,s + (\beta + K_m K_p)} \qquad (6)$$

multiplying and dividing the numerator by $(\beta + K_m K_p)$,

$$G_O(s) = \frac{CK_m}{(\beta + K_m K_p)} \left[\frac{(\beta + K_m K_p)}{s^2 + (\alpha + K_m K_d)s + (\beta + K_m K_p)} \right] \quad (7)$$

In order to obtain the ideal transfer function, we set $G_O(S) = G_O^*(S)$ which implies:

$$\frac{CK_m}{(\beta + K_m K_p)} \left[\frac{(\beta + K_m K_p)}{s^2 + (\alpha + K_m K_d)s + (\beta + K_m K_p)} \right] = K^* \left[\frac{\omega^{*2}_n}{s^2 + 2\zeta^* \omega^*_n s + \omega^{*2}_n} \right] \quad (8)$$

the new controller gains obtained from the above equations are then given by:

$$K^*_p = \frac{\omega^{*2}_n - \beta}{K_m} \qquad K^*_d = \frac{2\zeta^* \omega^*_n - \alpha}{K_m} \qquad C^* = \frac{K^*(\beta + K_m K^*_p)}{K_m} = \frac{K^* \omega^{*2}_n}{K_m}$$

Training: The network employed is used in the *supervised learning mode* and is trained off line before its inclusion into the control loop.The *training data module* comprises of a data generator whose inputs form the approximate ranges of the system parameters which, in the present application, are the *natural frequency* and the *damping ratio* of the generalized second order system. A typical range of the natural frequency is estimated from the knowledge the plant time constant used in the process.The ranges selected thereof cover approximately the entire gamut of the systems time response to maximum deviation in the plant parameters. A *system emulator* that consists of the process model generates the actual system time response to the various input signals using different settings of the parameters within the parameter ranges specified. The parameters are incriminated in discrete steps according to a prescribed resolution that is dependent upon the trade off between accuracy of identification desired and the size of the network. The time responses of the assumed system model are then passed on to the *feature extractor* that determines the various features or performance indices of the response. The performance indices along with the respective system parameters form the training data set for the neural network. Once the training data is available, the network is configured such that the number of nodes in the feature representation field corresponds to the dimension of the input feature vector,while the number of processing nodes in the category representation field are generally set greater than total number of input patterns in the prototype set. Each pattern in the prototype set is sequentially presented to the network once. A second cyclic presentation of the prototype set may be made for a stable category confirmation. During training, the attentional vigilance parameter is set at its highest value (0.999) to ensure a high resolution of the resulting category structure. The category structure represents the state space partitioning of the neural network depending on the number of stable categories established during training.When the network is presented with a feature vector for the first time, it is encoded in the LTM through modification of the LTM connection weights. The parameters associated with the feature vector now get assigned to this allocated node in the category representation field. On presentation of the subsequent feature vectors, the network's orienting subsystem first determines closeness of match between the pattern currently imposed on the network and any of the patterns that have previously been seen. Since the vigilance parameter is set so high a new node is allocated for the pattern. However, if the current pattern happens to be exactly similar to the one the network has seen before it is clustered into the same category. It is therefore possible to partition the networks state space such that each partition serves as an attractor for a particular type of response characterized by its feature vector. After completion of training the top down and the bottom up connection weights of the network along with the network parameters are saved. To make the network uniformly sensitive to all components of the feature vector, the feature vector components were enhanced by passing the feature vector through a nonlinear function given by:

$f_i(x) = e^{\gamma x}$ where $\gamma = 0.05$

Testing: When the network is introduced in the control loop, identification proceeds almost instantly. Search for the category associated with the right parameters is achieved by dynamically altering the attentional vigilance parameter until an "optimal vigilance" is found, [Kumar & Guez, 1989].

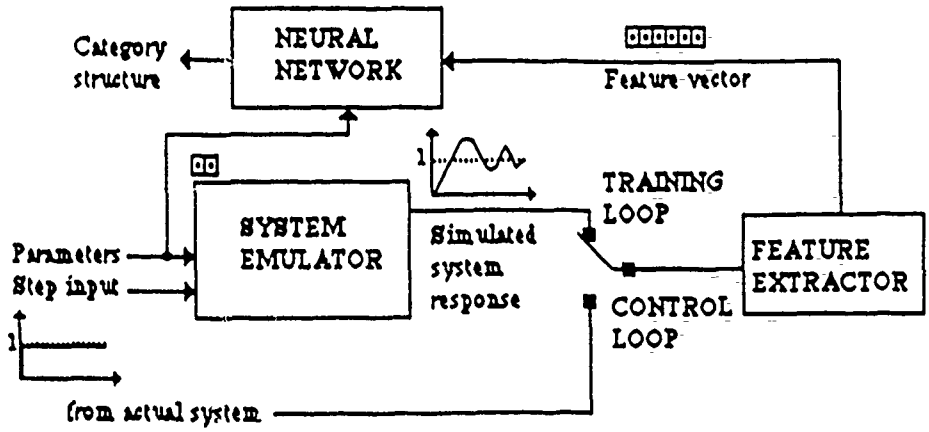


Figure 3 Block diagram of network training module

Experimental Results : The time response of the system is obtained using a Runge-Kutta fourth order differential equation solver (RK-4).Each of the following figures depict two plots. (a) The response of the actual system to the unit step input signal with arbitrary initial parameters α, β, and K_m, and (b) the final system response with the *D.C. bias* or the *steady-state error* compensated through the parameter C. The neural network employed is trained off-line on the features of the response obtained using the generalized second order equation with the following parameter variations:

$$0.1 \le \zeta \le 1.50, \quad \Delta\zeta = 0.10$$
$$0.5 \le \omega_n \le 2.50, \quad \Delta\omega_n = 0.50 \qquad (9)$$

These parameters are not made available to the system identifier but are estimated through the neural network in the control loop, based upon the features of the system time response to the standard test signals. The only available information to the feature extractor module which precedes the neural network in the overall process block diagram is the time and the value of the system response at that time instant. Specification of the parameters α_o and β_o enable in the computation of the desired damping and the desired natural frequency of the system response based upon the desired location of the system closed loop poles within the left half of the S- plane. In the case of the square function and the square wave, it should be noted that feature extraction was restricted to the step portion of the response (12 seconds). Figure (a) and (b) represent the nominal responses of the assumed ideal model to the unit step and the square function. Figures (c) and (d) depict the time variation of the plant parameters during the identification process. Figures (e) and (f) show the ability of the proposed neurocontroller to modify an overdamped and an underdamped system response respectively, with the input being a unit step function. It should be noted that the original system response indicated refers to arbitrarily assigned initial plant parameters. The final controller parameters are those obtained after the plant parameters are estimated by the neural network system identifier and the relocation of the overall closed loop system poles by the pole placement module. Figures (g) and (h) refer to the square input function. In Figures (i) and (j) are shown the output responses of the plant when the square wave is imposed at the input. The time scale incorporated in the simulation is adaptive for the unit step input and depends upon the duration of the system transient response.The sampling frequency selected is common to both the response sampling rate and the RK-4 differential equation solver. It is well above the Nyquist frequency of the system.

Conclusion : A new approach for dynamic system identification has been attempted that involves the application of a neural network architecture based on the Adaptive Resonance Theory. It has been shown that it is possible to train the ART neural net offline (supervised learning mode) to identify the parameters of a system based upon attributes of its response to standard test signals. A simple indirect adaptive control scheme was formulated, implemented and tested to assess the performance of the network that was incorporated as the online system identifier in the control loop. Experimental results suggest that identification provided by the network is accurate within the resolution of the training data, and its is possible to control a low order plant whose parameters are either unknown or are time varying. The off line training of the network was found to be fast with the training experimental data set being presented to the network only once.

References :

Astrom, K. J and Wittenmark, B. (1971). Problems of Identification and Control, *Journal of Mathematical Analysis and Applications*, Vol. 34, pp. 90-11.

Goodwin, G.C. and Payne, R. L. (1977). Dynamic System Identification- Experiment Design and Data Analysis, *Mathematics in Science and Engineering*, Academic Press, Vol. 136.

Grossberg ,S. and Carpenter, G. A. (1987 (a)). ART:2 Self Organization of Stable Category Recognition Codes for Analog Input Patterns, *Applied Optics*, Vol. 26. No. 23, pp.4919-4930.

Grossberg ,S. and Carpenter, G. A. (1987(b)). Invariant Pattern Recognition and Recall by an Attentive Self Organizing ART Architecture in a Nonstationary World, *Proc. of First International Conference on Neural Network*, San Deigo (IEEE, New York).

Kumar, S. S. and Guez,A. (1989). A Neural Network Approach to Target Recognition, *Proc. of IEEE International Joint Conference on Neural Networks*, Washington D.C., USA, pp. II-573.

Kumar, S. S. and Guez,A. (1989). ART based adaptive pole placement for neurocontrol, *Journal of Neural Networks*, (submitted for publication)

Figures :

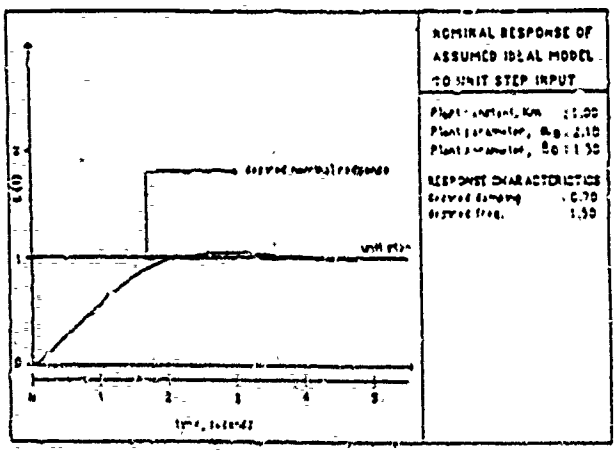


Figure (a)

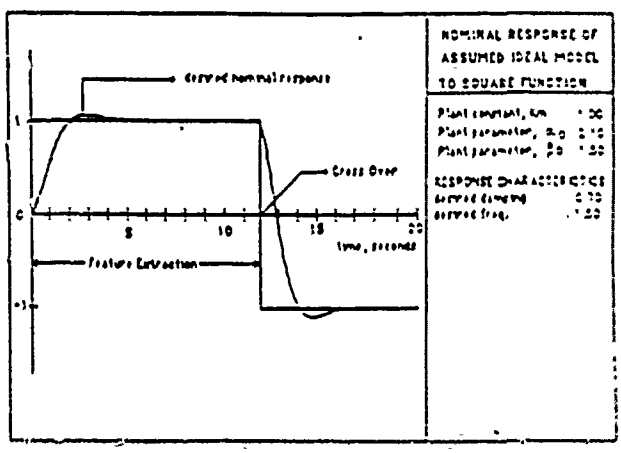


Figure (b)

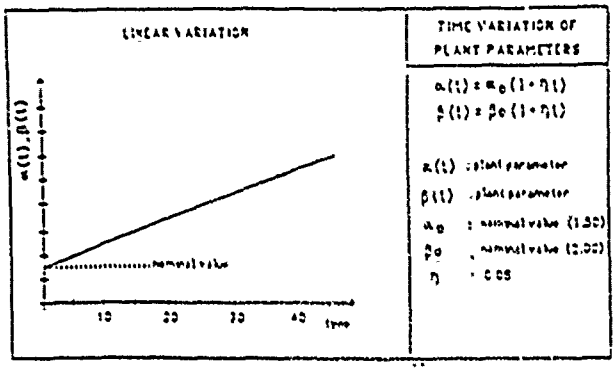


Figure (c)

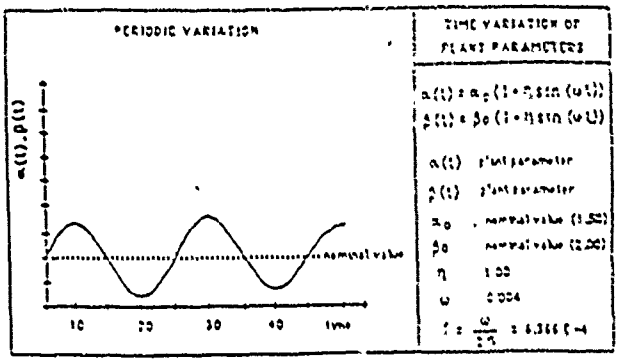


Figure (d)

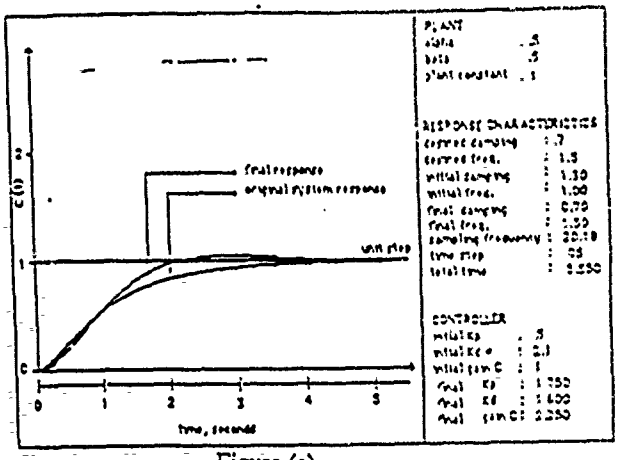


Figure (e)

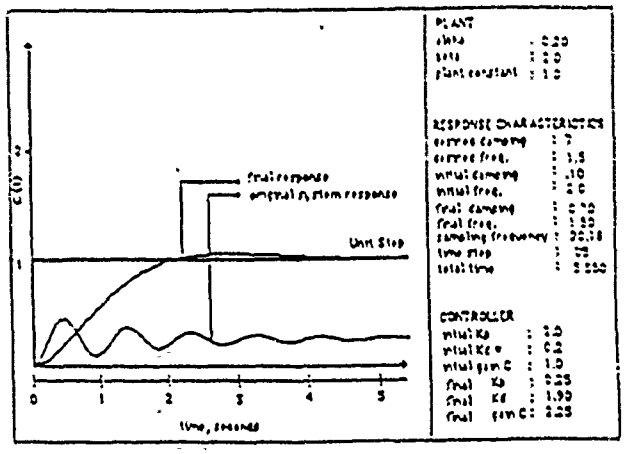


Figure (f)

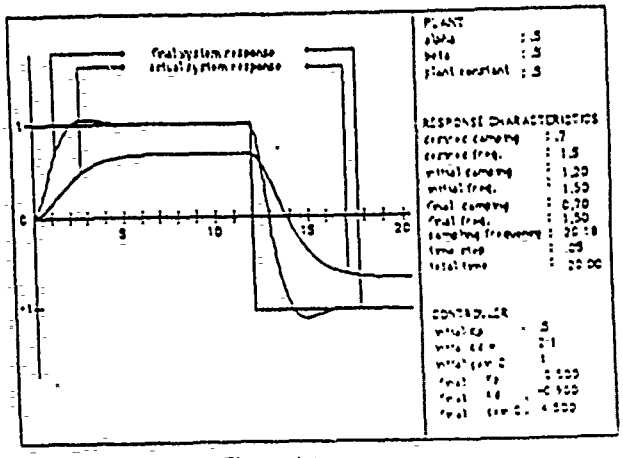


Figure (g)

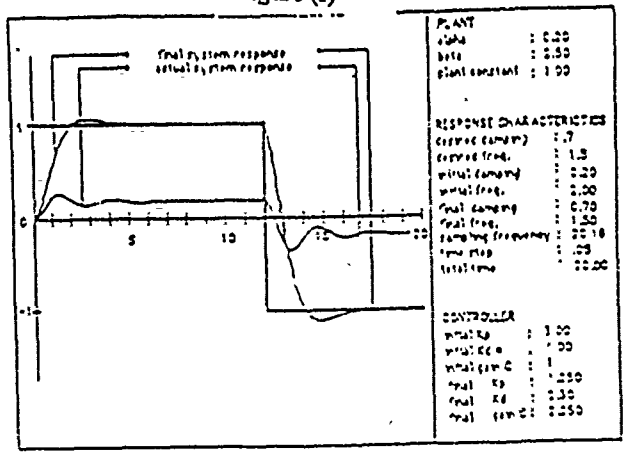


Figure (h)

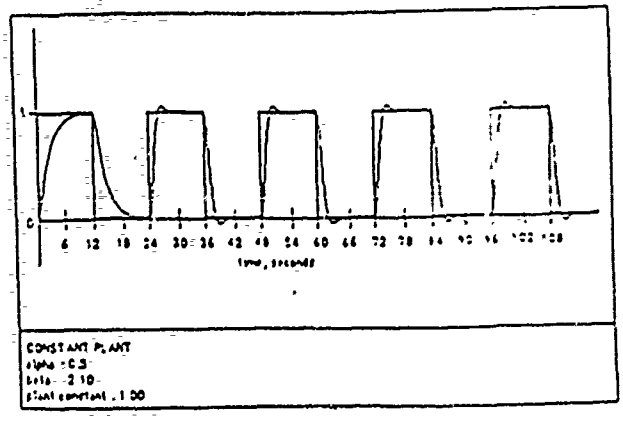


Figure (i)

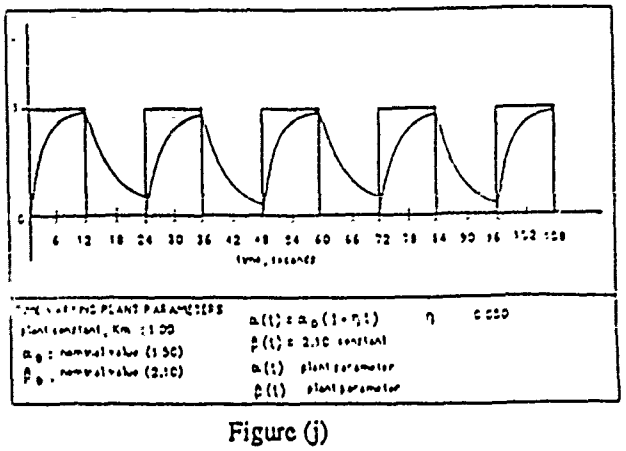


Figure (j)

[16]

A Neurocontroller with Guaranteed Performance for Rigid Robots

Allon Guez J.W. Selinsky

Drexel University, ECE Department
32nd and Chestnut St., Philadelphia, PA 19104

Summary

In this article, we propose and evaluate a neural network based adaptive controller for rigid robots. The proposed neurocontroller incorporates a priori knowledge of the robot's dynamic structure so as to provide proven trajectory tracking and parameter identification. A theorem regarding constructive sufficient conditions for asymptotically stable closed loop learning is stated and used in the design of Exploratory Schedules (ES). ES are reference trajectories which are specifically designed to provide efficient learning. In neurocomputing, ES are the training samples/examples that are used to modify the network architecture during learning. We specify the ES as a desired trajectory that is to be followed to do learning while the manipulator is not doing other useful tasks.

Simulation results of a 2 degrees of freedom (DOF) manipulator are given.

Rigid Robot Dynamics

The Lagrange-Euler formulation of rigid robot dynamics [1], has the form

$$\tau = Y[q,\dot{q},\dot{q},\ddot{q}]\Ø \tag{1}$$

where $Y[q,\dot{q},\dot{q},\ddot{q}]$ is an n x p time varying matrix of known and generally nonlinear functions, $\Ø$ is a p x 1 vector of constant parameters. τ is an n x 1 vector of joint torques, q is an n x 1 vector of joint coordinates, and n is the number of degrees of freedom.

Neurocontroller Design

When (1) is written in terms of the individual torques at each joint, it can be viewed as a single layer linear network, where the inputs to the network are the $Y_{ij}[.]$ and the weights are the $\Ø_j$ [2],[3]. The $Y_{ij}[.]$ are transcendental algebraic functions of the manipulators states and may be realized via feedforward neural networks that are trained with a suitable learning algorithm.

Let q_d, $\dot{q}_d$, $\ddot{q}_d$ designate the desired trajectory. Following [4] define the virtual reference trajectory $\dot{q}_r$, $\ddot{q}_r$, and the virtual trajectory velocity error $\dot{e}_r$

$$\dot{q}_r = \dot{q}_d - A\,e, \quad \ddot{q}_r = \ddot{q}_d - A\,\dot{e}, \quad \dot{e}_r = \dot{q} - \dot{q}_r = \dot{e} + Ae \tag{2}$$

where $e[t] = q[t]-q_d[t]$ is the joint coordinate error, and A is a positive definite matrix with constant coefficients.

The output of the neurocontroller implements the control law

$$\tau_i = \sum_{j=1}^{p} Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]\,\hat{\Ø}_j + K_{d_{ii}}\,\dot{e}_{r_i}, \quad i = 1,2,\ldots,n, \tag{3}$$

where $\hat{\Ø}$ denotes the estimate of $\Ø$. Equation (3) has been shown to be asymptotically stable when the learning rule

$$\dot{\hat{\Phi}}_j = -\sum_{i=1}^{n} \frac{1}{K_{a_{ii}}} Y_{ij}[q, \dot{q}, \dot{q}_r, \ddot{q}_r]\, \dot{e}_{r_i}, \qquad j = 1,2, \ldots, p. \tag{4}$$

is used [4].

Notice in equation (4) the similarity to the LMS learning rule (see [5]) where the weight change is proportional to the error ε and the input features X. In equation (4) the input features are the $Y_{ij}[.]$ functions and the error is $\dot{e}_{r_i}$.

Figure 1 shows the internal structure of the proposed neurocontroller for joint i. Each feedforward network module is trained to provide one of the $Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]$ functions. **This training can be done offline since the $Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]$ functions are known a priori and are the same for all rigid robots of the same kinematics and number of degrees of freedom.** The inputs to the neurocontroller are the components of the trajectories as required. The outputs of the neurocontroller are the control torques to be applied at each joint of the manipulator.

Closed Loop Learning and Tracking Via Exploratory Schedules

Theorem 1 [6]: If $p \le n$, and if

$$\text{Rank} \left\{ \begin{matrix} Y[q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d] \\ \lim\ t \to \infty \end{matrix} \right\} = p, \tag{5}$$

then the controller specified by equations (3) and (4) guarantees global asymptotic tracking and parameter identification i.e. $q(t) \to q_d(t)$, and $\hat{\Phi}(t) \to \Phi(t)$.

Lemma 1 [7]: $\tilde{\Phi} \to 0 \ \forall\ \Phi \in R^k$, $k \le n$, such that Φ is not contained in

$$N\left(\begin{matrix} Y[q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d] \\ \lim\ t \to \infty \end{matrix} \right), \tag{6}$$

where N(.) denotes the null space of the matrix (.), and denotes the parameter estimation error vector. The implication of lemma 1 is that if, $k \le n$ parameters are not in the null space of equation (6) then the components of the parameter estimation error vector corresponding to the k parameters are zero.

Design of Exploratory Schedules

Lemma 1 implies desired trajectories (which specify $q_d, \dot{q}_d, \ddot{q}_d$) can be designed such that $k \le n$ specific columns of $Y[q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d]$ will be linearly independent in the limit. The exploratory schedule then consists of a sequence of desired trajectories which are designed to learn different components of the parameter estimation vector $\hat{\Phi}$, where the number of desired trajectories is such that all p components of $\hat{\Phi}$ are identified.

Simulation Results: 2 DOF Manipulator

The simulation results reported were obtained using exactly computed values of the $Y_{ij}[.]$ functions, rather than the three-layer neural network form. This simulation provides experimental verification of lemma 1. The dimension of $Y[q, \dot{q}, \dot{q}_r, \ddot{q}_r]$ in the 2 DOF case is 2 x 5 so that $p > n$,

and at most 2 parameters can be guaranteed to be learned simultaneously.

The Exploratory Schedule for the 2 DOF manipulator (see figure 2) consists of a sequence of three desired trajectories so chosen as to guarantee learning of different parameters. In trajectory 1, $\emptyset_4$ and $\emptyset_5$ are learned. The time period corresponding to trajectory 1 is $0 \le t < 4.2$. In trajectory 2, $\emptyset_3$ is learned. The time period corresponding to trajectory 2 is $4.2 \le t < 7.3$. In trajectory 3, $\emptyset_1$ and $\emptyset_2$ are learned. Trajectory 3 corresponds to the time period $7.3 < t \le 10.3$.

The time period corresponding to identification of $\emptyset_4$ and $\emptyset_5$ is $0 \le t < 4.2$. As can be seen in figures 3 to 6, the parameter estimation error for $\emptyset_4$ and $\emptyset_5$ approaches zero as all joint errors approach zero at t= 4.2. The time period corresponding to identification of $\emptyset_3$ is $4.2 \le t < 7.3$. The parameter estimation error for $\emptyset_3$ approaches zero as all joint errors approach zero at t= 7.3. Parameters $\emptyset_1$ and $\emptyset_2$ are identified during the time period $7.3 < t \le 10.3$.

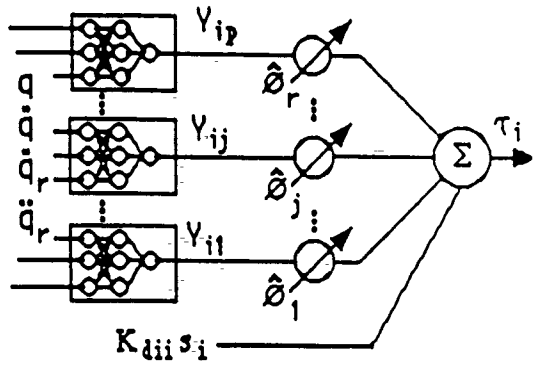


figure 1: Neurocontroller structure for link i.

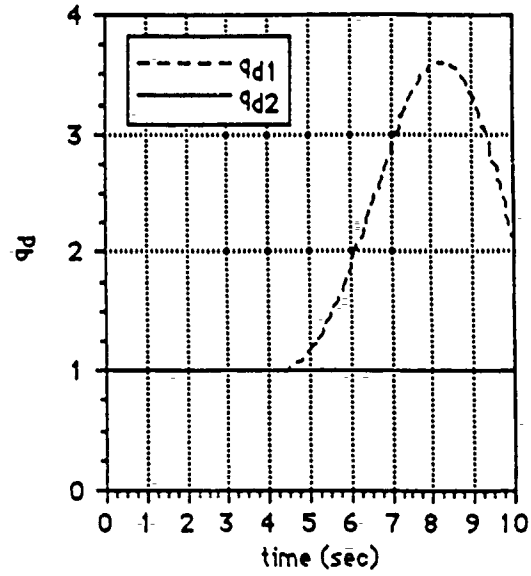


figure 2: Desired joint positions during ES.

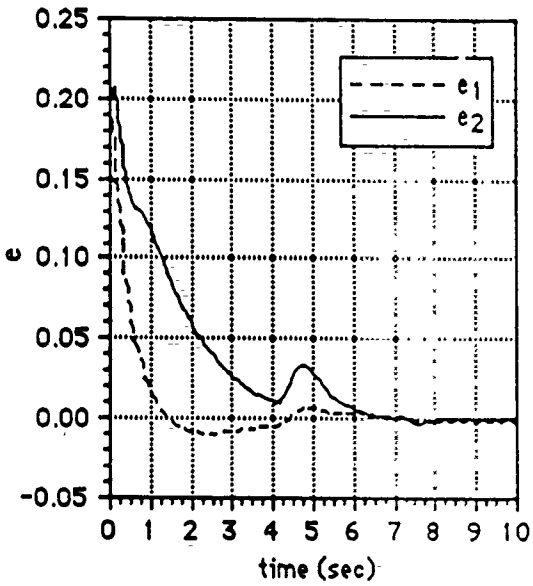


figure 3: Joint position error during ES.

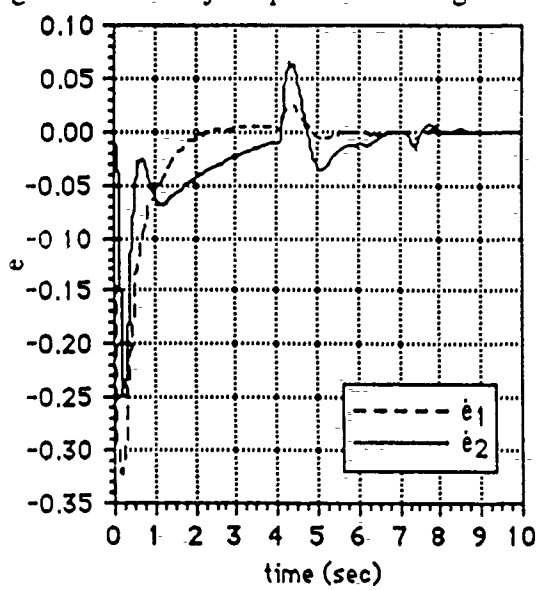


figure 4: Joint velocity error during ES.

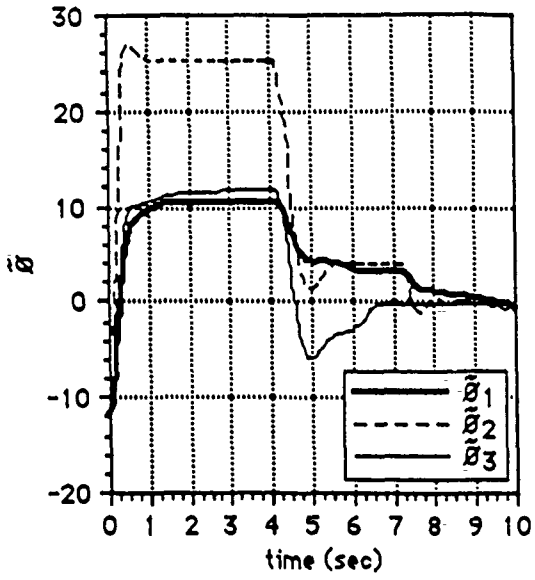


figure 5: Estimation error for $\emptyset_1$, $\emptyset_2$ and $\emptyset_3$.

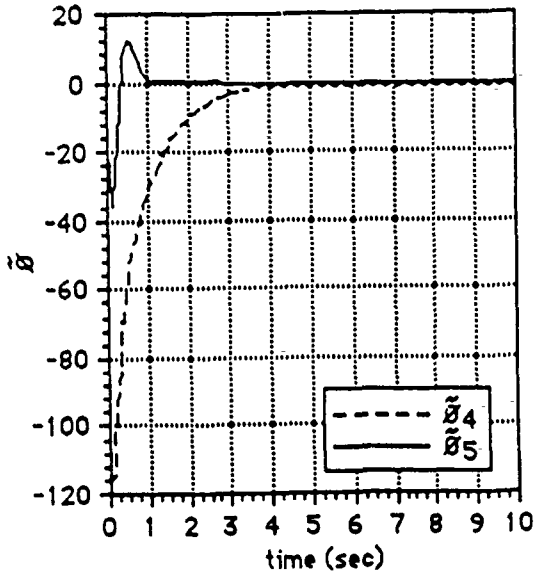


figure 6: Estimation error for $\emptyset_4$ and $\emptyset_5$.

References

[1] Paul, R.P., *Robot Manipulators: Mathematics, Programming, and Control.* Massachusetts: The MIT Press, 1981.

[2] Kawato, M., Furukawa, K. and Suzuki, R., "A hierarchical neural-network model for control and learning of voluntary movement," *Biological Cybernetics*,56, 1987.

[3] Guez, A., and Selinsky J., "Neurocontroller design via supervised and unsupervised learning," to appear in the *Journal of Intelligent and Robotics Systems*, 1989.

[4] Slotine, J.J.E., and Li, W., "Adaptive manipulator control: a case study," *IEEE Trans. Automatic Control*, Vol. 33, no.11, p. 995, 1988.

[5] Widrow, B., and Stearns, S. D., *Adaptive Signal Processing*. New Jersey: Prentice-Hall, 1985.

[6] Guez, A., and Selinsky J., "Design of Exploratory Schedules in Learning and Adaptive Robot Control," accepted for publication in the *International Journal of Adaptive Control and Signal Processing*, 1989.

[7] Guez, A., and Selinsky J., "A Neurocontroller with Guaranteed Tracking and Model Identification for Rigid Robots," submitted for publication, 1989.

[18]

ADAPTIVE SENSOR FUSION WITH NETS OF BINARY THRESHOLD ELEMENTS

Moshe Kam, Ari Naim, Paul Labonski, and Allon Guez

Department of Electrical and Computer Engineering, Drexel University, Philadelphia PA 19104

Abstract

Distributed detection and estimation, and sensor fusion, are at present topics of intensive research due to the variety of systems which employ different sensing devices at geographically separated sites for tasks such as target detection and tracking, diversity in communications and threat classification. In this study, we demonstrate a simple distributed-detection scheme whose probability of error can be calculated analytically, and show that it corresponds to a two-layer network of binary threshold elements. We proceed to assume that the sensors and the fusion center are subject to sudden unpredictable changes in the environment that they survey, and show how learning algorithms can be used in order to maintain good performance, in spite of these changes. We conclude with an example, involving five unequal sensors which distinguish between two time-varying Gaussian populations of different means.

I. Introduction and statement of the problem

The term *sensor fusion* is used to describe the mechanism for combining data from distributed sensors. The fusion is usually performed for purposes of detection or estimation, with applications such as: target detection and estimation in radar systems, diversity in digital communications systems and crime countermeasures. For relevant references see Bar-Shalom and Fortmann (1988), Thomopoulos et al. (1987) and Riebman and Nolte (1987). In the present paper we concentrate on the *distributed parallel detection problem with binary local decisions*, depicted in figure 1b. We consider a system which observes a phenomenon H, through a set of sensors which may have different operating principles, and may be located at different sites. An example of the former case is a combination of radar active sensors, operating at different frequencies (e.g. infrared and RF sensors); an example of the latter is an array of receiving antennae, located at different physical sites - for *spatial diversity*. Each one of the front-end sensors, also called *local detectors*, observes the phenomenon H, and makes a *local decision* as to the existence (u_i=+1) or non-existence (u_i= -1) of a target within its volume of coverage. This decision can be arrived at on the basis of a single, multiple or sequential observations at the local detector's site. Typically, a likelihood ratio test will be performed, with a decision threshold determined by the objective function (usually a Bayesian risk or a Neyman-Pearson criterion.) The calculation of the threshold typically involves the solution of a set of coupled nonlinear algebraic equations, a task which can become computationally untractable even for a moderate number of sensors (e.g. Tsitsiklis and Athans (1985), Reibman and Nolte (1988).)

The decisions of the local detectors are transmitted to a global detector (the Data Fusion Center, DFC) over communication channels that are practically noiseless. The DFC is to make a *global decision* about the existence of the target (u=1: decide H_1, or u=-1: decide H_0.)

Figure 1 depicts two sensor fusion systems which differ in terms of the capability to transmit local decisions to the DFC. The first case (figure 1a) is applicable when all sensors are located at the same site, or when the sensors are distributed, and the capacity of the channels between the sensors and the DFC allows practically error-free transmittal of the sensor observations (z_i) to the central processing site. The DFC obtains a vector of real observations, and its decision is based on classical multiple-observation hypothesis-testing theory (e.g. Sage and Melsa (1971).)

When the sensors are distributed, and capacity constraints do not allow the transmittal of the observation (or when the sensors are at the same site - but central processing capabilities are limited,) there is a need to make partial decisions at the sensors' sites, and transmit those partial decisions to the DFC. The extreme case (*fully decentralized decision-making*) is depicted in Figure 1b. Here each sensor makes a local decision: u_i=1 if a target is detected, and u_i= -1 otherwise. Only this binary decision is available at the DFC as the contribution of the i^{th} sensor to the decision process. The DFC has to map the binary vector of local decisions into the global decision.

Research supported in part by NSF grant IRI 8810186

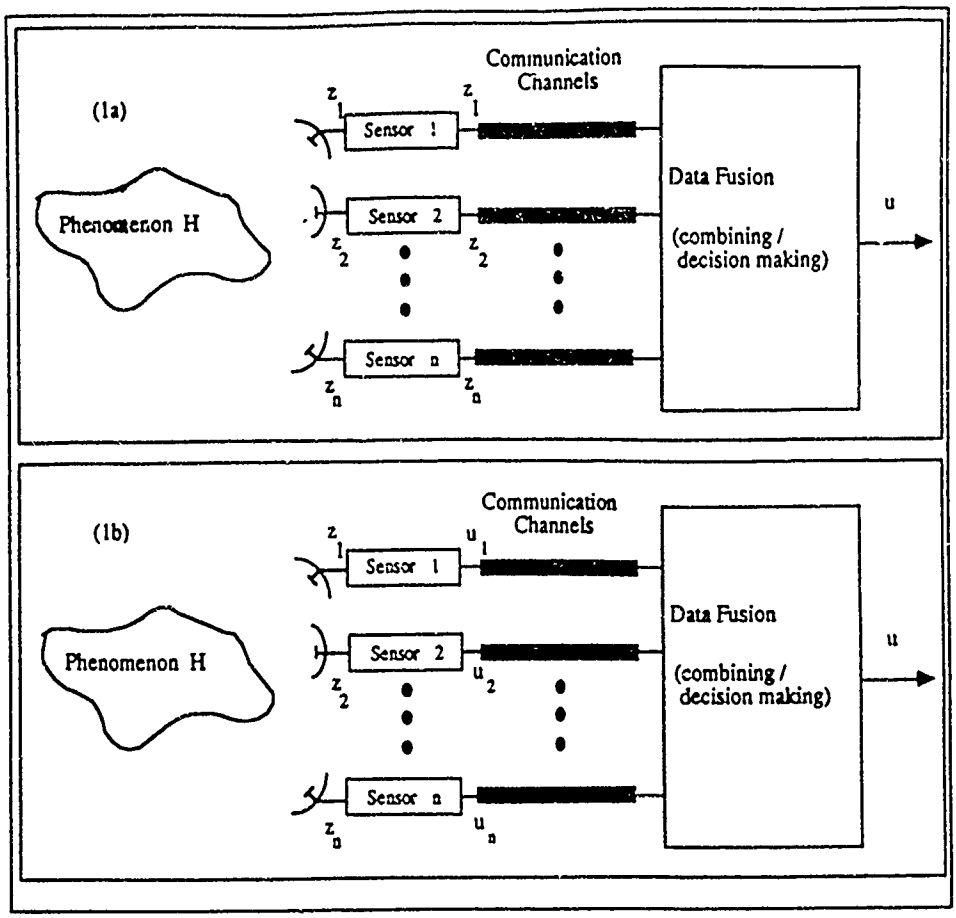


Figure 1: Multiple-observation hypothesis-testing
a) centralized system
b) fully-decentralized system

The basic questions that arise from the described situation are:

1) What are the *(optimal and sub-optimal) configurations* that could be employed for the local sensors and for the DFC when the system is *fully informed*: By *fully informed* we mean that the statistics of the environment are known to the local detectors and the DFC; and the reliability of each local decision is known to the DFC in terms of local probability of missed detection and the probability of false alarm.

2) What is the *performance* that the fully-informed decision maker achieves?

3) How can *learning techniques* be incorporated in the decision-making schemes, such that unknown or time-varying properties of the environment could be extracted from available data, and used for adaptive decision making?

The present paper contributes to the study of the second and third problems. We are interested in exploiting the ability of neural networks to learn, in order to maintain performance of the 'ignorant' distributed configuration, which is close to that of the fully-informed system: such performance is to be maintained, in spite of unpredictable changes in the a-priori probabilities of the hypotheses and in the signal-to-noise ratios at the individual sensor sites. We develop, analyze and simulate a parallel architecture with local and global binary neurons as the decision-making elements. These neurons adapt their weights and thresholds to track the (unknown) parameters of the environment which supplies them with data for decision making.

Section II presents the distributed sensor configuration that we study - which considers two hypotheses in the observed environment, and non-communicating sensors. The configuration is relatively easy to implement, but is suboptimal with respect to a *minimum probability of error* performance criterion. In section III we assume that the individual sensors do not have access to the exact statistics of the hypotheses that they examine, and that the data fusion center does not have access to the exact performance of the individual sensors (their probabilities of false alarms and missed-detections.) We present a stochastic-approximation rule which is used for learning,

in order to allow the system to achieve asymptotically the performance of a system which is fully informed of the necessary probabilities. Finally, we present in section IV the results of numerical studies of the performance of a five-sensor target detection radar system, with unequal and time-varying signal-to-noise ratios and a priori probabilities.

II. The distributed-sensor configuration

The design of a distributed-sensor configuration for minimizing the probability of error, a Bayesian cost performance index, or a Neyman-Pearson criterion has been the subject of many studies (see Thomopoulos et al. 1987) and Reibman and Nolte (1987) for a list of references.) The *minimum probability of error criterion* which we are using here is
on the *local* level

$$P_{el} = P_r(H_0)\, P_r\,(u_i = 1 \mid H_0) + P_r(H_1)\, P_r\,(u_i = -1 \mid H_1) \tag{1a}$$

and on the global level

$$P_{eG} = P_r(H_0)\, P_r\,(u = 1 \mid H_0) + P_r(H_1)\, P_r\,(u = -1 \mid H_1) \tag{1b}$$

The optimal local decision that is necessary in order to minimize P_{eG} does not minimize P_{el}, in general. It has been shown that the optimal local decision involves a maximum-likelihood test with a threshold whose calculation is coupled to that of the calculation of thresholds for all other local detectors. We are using here a simpler, and hence sub-optimal, configuration, where each detector makes its own local *minimum probability of error* detection, rather than the decision which corresponds to the global *minimum probability of error* decision.

We assume that the decision problem corresponds to choosing one of two Gaussian populations with different means. In other words the detection system has to decide whether its i^{th} observation z_i is

$$\left.\begin{aligned} z_k &= \upsilon_k && (2a)\\ \text{or} & \\ z_k &= m + \upsilon_k && (2b) \end{aligned}\right\} \quad k = 1,2,\ldots,K$$

where the υ_k's are independent with the probabilty density

$$p_V(\alpha) = \frac{1}{(2\pi\sigma^2)^{\frac{K}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{K}\alpha_k^2\right)$$

and K observations are used before a decision is made. The local *minimum probability of error* decision is

$$\frac{1}{\sigma\sqrt{K}}\sum_{i=1}^{K} z_i \underset{\text{accept } H_0}{\overset{\text{accept } H_1}{\gtrless}} \frac{\sigma}{m}\ln\frac{P_r(H_0)}{P_r(H_1)} + \frac{Km}{2} = \tau_s. \tag{3}$$

In this case one can express analytically the local probability of error, as well as the local probability of a missed detection and the local probability of false alarm (Sage and Melsa (1971), p. 130):

$$P_e = \frac{1}{2} - \frac{1}{2}\int_{\tau_s}^{\frac{\sqrt{K}m}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}}\, ds \tag{4a}$$

$$P_M = \int_{-\infty}^{\frac{\sqrt{K}m}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}}\, ds \tag{4b}$$

$$P_F = \int_{\tau_s}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}}\, ds \tag{4c}$$

Assuming now that each local detector (with its own σ_i, P_{Mi} and P_{Fi}) makes a decision according to equation (3), the *global minimum-probability-of-error decision, given the local decision rules* is (Chair and Varshney (1986)):

$$\text{Decide } u_i = 1 \text{ if } a_0 + \sum_{i=1}^{n} a_i u_i > 0$$

$$\text{Decide } u_i = -1 \text{ otherwise} \tag{5}$$

where

$$a_0 = \log\frac{P(H_1)}{P(H_0)} \tag{6a}$$

$$a_i = \frac{1}{2}\log\frac{1-P_{Mi}}{P_{Fi}}(u_i + 1) + \frac{1}{2}\log\frac{1-P_{Fi}}{P_{Mi}}(1 - u_i) \tag{6b}$$

The overall probabilities of missed detection and false alarm can be calculated for this system as well - appendix A gives the details. When the noise variances at the local detectors are equal, the answer is

$$P_{MG} = 1 - \sum_{j=0}^{n}\binom{n}{j}(1-P_M)^j P_M^{n-j}\, U[A_0 + a(2j-n)] \tag{7a}$$

$$P_{FG} = \sum_{j=0}^{n}\binom{n}{j}P_F^j (1-P_F)^{n-j}\, U[A_0 + a(2j-n)] \tag{7b}$$

where

$$A_0 = \ln\frac{P(H_1)}{P(H_0)} + \frac{n}{2}\ln\frac{P_M(1-P_M)}{P_F(1-P_F)} \tag{7c}$$

$$a = \frac{1}{2}\ln\frac{P_M(1-P_M)}{P_F(1-P_F)} \tag{7d}$$

U(.) is the unit step function,

and P_M and P_F have been defined in equation (4).

From examination of equations (4) and (6), we note that

the distributed decision system is a two-layer binary neural network. The local detectors have K real inputs and one binary output each (K = number of observations per decision,) and the data fusion center: n binary inputs and one binary global decision output (n= number of sensors.) Figure 2 depicts the overall system.

III. On-line Learning

So far we have assumed that the local detectors and the DFC know the a priori probability ($P(H_0)$), and that the DFC knows the probabilities of missed detection and false alarm (P_{Mi} and P_{Fi}) of each of the sensors which feed it with information. When these probabilities are not available, we use simple stochastic approximation rules in order to estimate them. These rules have been successfully used in other neural network learning applications (e.g. Kohonen (1988), Kam et al. (1988)).

Approximation of the a priori probabilities

At the local detector level

$$\hat{P}_i(H_0)^{(k+1)} = \hat{P}_i(H_0)^{(k)} + \alpha_{li}\left(\frac{1-u_i}{2} - \hat{P}_i(H_0)^{(k)}\right) \qquad (8a)$$

where $\hat{P}_i(H_0)^{(k)}$ is the estimate of $P(H_0)$ at the i^{th} local detector at the k^{th} time step; and α_{li} is the learning constant at that detector, $0 < \alpha_{li} < 1$.

At the DFC level

$$\hat{P}_{DFC}(H_0)^{(k+1)} = \hat{P}_{DFC}(H_0)^{(k)} + \alpha_{FDC}\left(\frac{1-u}{2} - \hat{P}_{DFC}(H_0)^{(k)}\right) \qquad (8b)$$

where $\hat{P}_{DFC}(H_0)^{(k)}$ is the estimate of $P(H_0)$ at the DFC at the k^{th} time step; and α_{DFC} is the learning constant $0 < \alpha_{DFC} < 1$.

Approximation of the error probabilities

At the DFC level

$$\hat{P}_{Mi}^{(k+1)} = \hat{P}_{Mi}^{(k)} + \beta_{DFC}\left(\frac{-u_i+1}{2} - \hat{P}_{Mi}^{(k)}\right)\left(\frac{u+1}{2}\right) \qquad (9a)$$

$$\hat{P}_{Fi}^{(k+1)} = \hat{P}_{Fi}^{(k)} + \beta_{DFC}\left(\frac{u_i+1}{2} - \hat{P}_{Fi}^{(k)}\right)\left(\frac{-u+1}{2}\right) \qquad (9b)$$

where $\hat{P}_{Mi}^{(k)}$ and $\hat{P}_{Fi}^{(k)}$ are the estimates of P_{Mi} and P_{Fi} at the i^{th} local detector at the k^{th} time step; and β_{DFC} is the corresponding learning constant at the DFC, $0 < \beta_{DFC} < 1$.

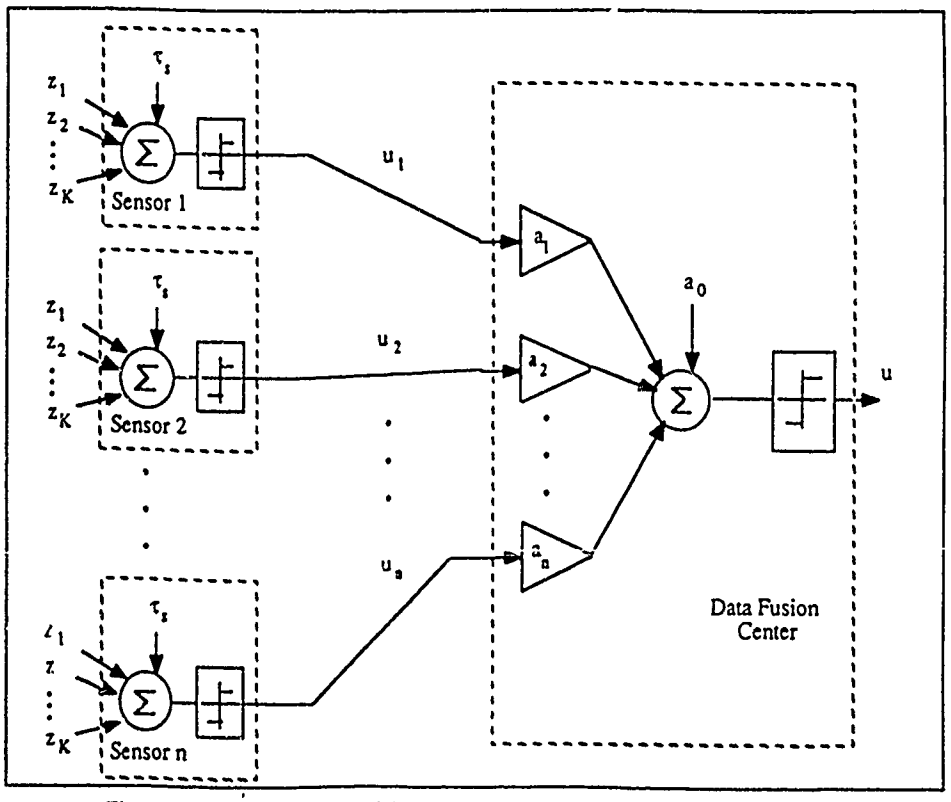


Figure 2: The distributed decision system as a two-layer binary neural network

While these approximation rules do exhibit a.s. convergence, they nevertheless are very useful in many applications that call for on-line parameter estimation.The estimated values in equations (8) and (9) are used in the decision rules (3) and (5) - to yield the global decision, u.

IV Example: a five-sensor target detection fusion system

We consider a fusion system for target detection using five unequal detectors. The system has been simulated on an IBM-AT using Turbo PASCAL 5.0. The presented results were obtained via Monte-Carlo simulations.

CASE I: Static environment

We have studied a system of five sensors in Gaussian white noise with signal to noise ratios:

Sensor 1:	5.0 dB
Sensor 2:	3.0 dB
Sensor 3:	0.0 dB
Sensor 4:	-3.0 dB
Sensor 3:	-5.0 dB

Figures 3a,b show the probability of correct decision (P_c = 1-P_e) versus time, for sensors 2,3, and 5, along with the results for the fully-informed DFC and the learning DFC.

We observe that both DFC's perform on average better than each sensor alone. Moreover, the learning DFC gradually approaches the performance of the fully-informed DFC.

Figures 4 a,b show the DFC weights (a_i in equation (6)) for sensors 2,3, and 5. Weights are shown for the case of a present target (figure 4a) and the case of no target in sight (figure 4b.) The horizontal lines are the weights for a fully-informed DFC, the other curves correspond to the learning DFC.

We observe that, as expected, unreliable sensors (like 5) have lower weights than more reliable sensors (like 2). Moreover, the learned weights converge to the values of the fully-informed DFC.

CASE II: Time-varying environment - varying SNR

We now introduce an abrupt change (from -5.0 dB to 5.0 dB) in the signal to noise ratio at sensor five, halfway into the simulation (after 50 time steps.) In figure 5a we show one 100-step run for the probability of correct decision for sensor 5, the learning and fully-informed DFC's. We observe the expected improvement in the probability of correct decision for both the sensor and the DFC's. Figure 5b shows the weights for sensors 2,3 and 5. The weights of (the

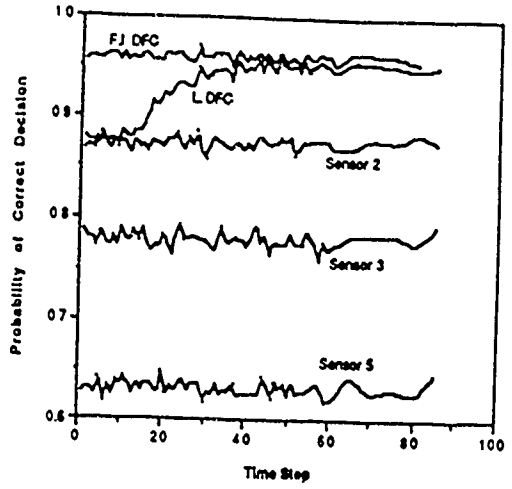


Figure 3a: The probability of correct decision for: sensors 2,3,and 5; the learning data fusion center (L.DFC); and the fully-informed data fusion center (F.I.DFC).

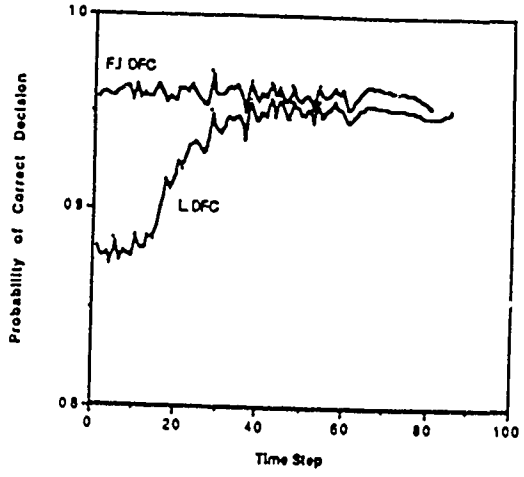


Figure 3b: The probability of correct decision for the learning data fusion center (L.DFC) and the fully informed data fusion center (F.I.DFC).

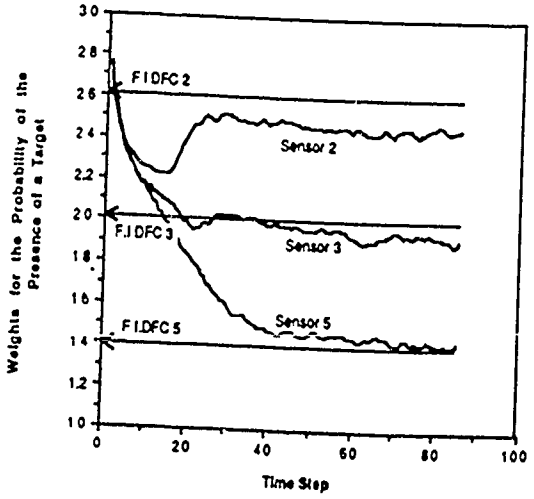


Figure 4a: The weights, associated with the decisions of sensors 2,3 and 5 when the target is present - learned by the data fusion center (L. DFC) and fixed for the fully informed data fusion center (F.I.DFC).

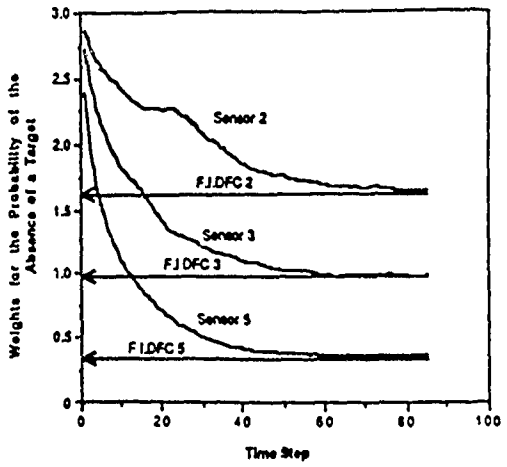


Figure 4b: The weights, associated with the decisions of sensors 2,3 and 5 when the target is not present - learned weights by the data fusion center (L. DFC) and fixed weights for the fully-informed data fusion center (F.I.DFC).

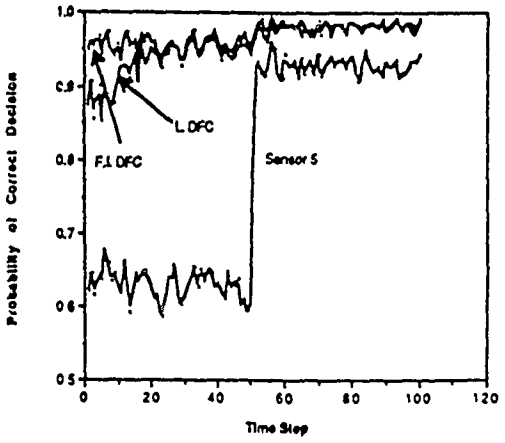


Figure 5a: The probability of correct decision for sensor 5, the learning data fusion center (L.DFC) and the fully informed data fusion center (F.I.DFC). A step change in the SNR of sensor 5, from -5dB to 5dB, is introduced after 50 time steps, resulting in an improvement of the performance of sensor 5 and the DFC's.

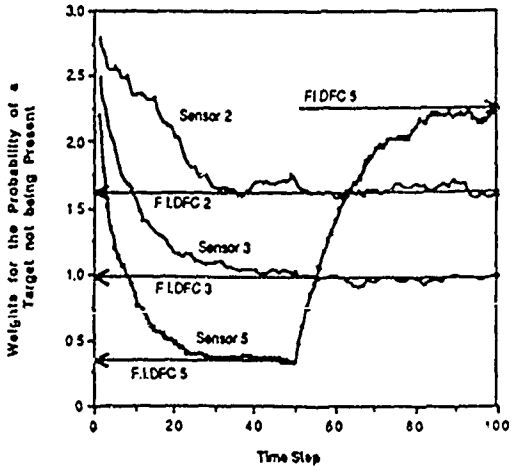


Figure 5b: The weights, associated with the decisions of sensors 2,3 and 5 when a target is not present, learned by the data fusion center (L. DFC) and fixed for the fully informed data fusion center (F.I.DFC). A step change in the SNR of sensor 5, from -5dB to 5dB, is introduced at 50 time steps.

learning) sensor 5 are shown to converge asymptotically to the optimal a_i weights of equation (6) (the horizontal lines.)

CASE III: Time-varying environment - varying a priori probabilities

We now change the a priori probability of H_0 from 0.3 to 0.7 halfway into the simulation (after 50 time steps.) In figure 6a we show one 100-step run for the probability of correct decision using the learning DFC and the fully-informed DFC. We observe that the learning system experiences a performance degradation after the probability change, but is able to recover, through learning, in a short period. The recovery is due to the restoration of correct estimates for the a-priori probabilities, as shown for the sensors in figure 6b. In figures 6c,d we show convergence to the optimal weights both before and after the a priori probability change.

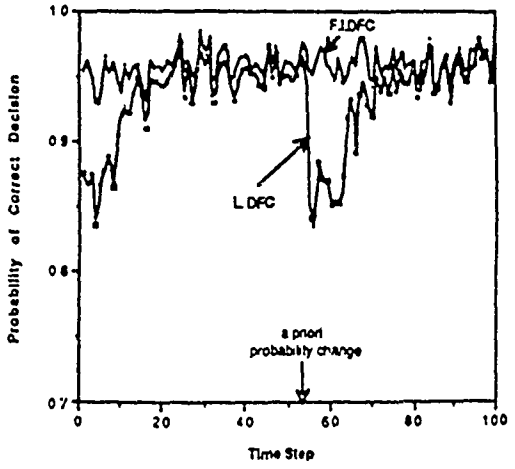


Figure 6a: The probability of correct decision for the learning data fusion center (L.DFC) and the fully-informed data fusion center (F.I.DFC). A step change in the probability of a target being present from 0.7 to 0.3 is introduced after 50 time steps, resulting in a temporary degradation in performance of the learning DFC (L. DFC.)

Conclusion

We have considered a distributed detection system whose performance, in terms of probability of error, can be calculated analytically. We have shown that the system can be realized as a two-layer network of binary threshold elements, and that the weights of these elements can be estimated, using on-line stochastic-approximation rules to form an adaptive detection system. The results are applicable to systems which use a variety of sensors located in separated geographical locations for decision making. Further research is necessary in order to determine an adaptation scheme for the learning constants: high learning constants encourage

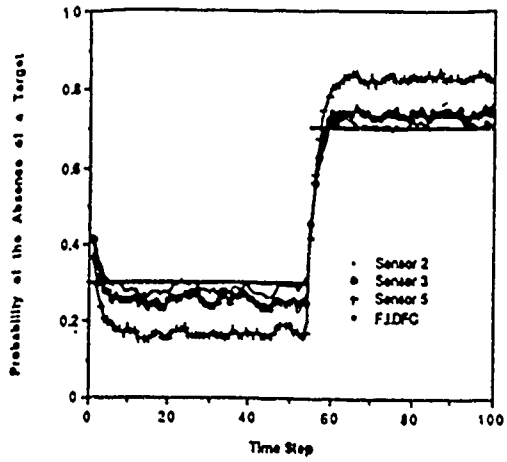


Figure 6b: The probability of a target not being present, learned locally by sensors 2,3 and 5, compared with the actual value known to the fully-informed data fusion center (F.I.DFC). A step change in the the probability of a target being present, from 0.7 to 0.3, is introduced after 50 time steps.

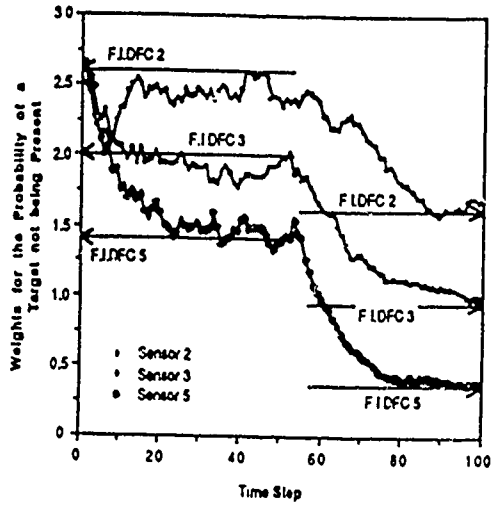


Figure 6c: The weights, associated with the decisions of sensors 2,3 and 5 when a target is not present - learned by the data fusion center (L. DFC) and fixed for the fully-informed data fusion center (F.I.DFC). A step change in the probability of a target being present, from 0.7 to 0.3, is introduced after 50 time steps.

the quick learning of changes, but suffer from high variance in the estimates. Small learning constants require long settling times. The solution - adaptive learning constants - should imitate adaptation mechanisms in similar systems, e.g. delta modulation in digital communications.

Appendix A: Performance of the distributed sensor configuration

The overall expression for the sufficient statistic of the Chair and Varshney's (1986) DFC is

$$\lambda = a_0 + \sum_{i=1}^{N} a_i u_i \qquad \text{(A1)}$$

where a_0 and a_i have been defined in equation (6). This expression can be re-written as

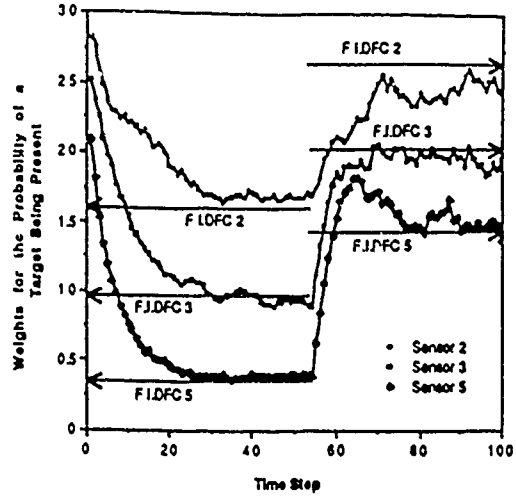


Figure 6d: The weights, associated with the decisions of sensors 2,3 and 5 when a target is present - learned by the data fusion center (L. DFC) and fixed for the fully-informed data fusion center (F.I.DFC). A step change in the probability of a target being present, from 0.7 to 0.3, is introduced after 50 time steps.

$$\lambda = \left(A_0 + \frac{1}{2}\sum_{i=1}^{N}(A_i - B_i) \right) + \sum_{i=1}^{N}\left(\frac{1}{2}(A_i + B_i)\right)u_i \qquad \text{(A2)}$$

where

$$A_0 = \ln \frac{P(H_1)}{P(H_0)} + \frac{1}{2}\sum_{i=1}^{n} \ln \frac{P_{Mi}(1-P_{Mi})}{P_{Fi}(1-P_{Fi})} \qquad \text{(A3-1)}$$

$$A_i = \ln \frac{1-P_{Mi}}{P_{Fi}} \qquad \text{(A3-2)}$$

$$B_i = \ln \frac{1-P_{Fi}}{P_{Mi}} \qquad \text{(A3-3)}$$

We consider first the case of equal noise variances at the local detector. All detectors use the same detection scheme, have the same threshold and the same probabilities of missed detections and false alarms.

$$P_{M1} = P_{M2} = \ldots = P_{Mn} = P_M$$

$$P_{F1} = P_{F2} = \ldots = P_{Fn} = P_F$$

Let:

$$P(u_i = +1 \mid H_k) = p_k$$

$$P(u_i = -1 \mid H_k) = 1 - p_k = q_k$$

where $k = 0,1$

Let x be the random variable

$$x = \frac{1}{2}\left(\sum_{i=1}^{n} u_i + n\right)$$

(x is the number of 1's in the observation vector at the DFC.)

Clearly,

$$P(x=j \mid H_k) = \binom{n}{j} p_k^j q_k^{n-j} \qquad \text{(A4)}$$

$j = 0, 1, \ldots, n.$

Equal variances We assume now that the noise variance is the same for all sensors.

Under this condition, all coefficients of u_i are the same and equal to

$$a = \frac{1}{2} \ln \frac{P_M (1-P_M)}{P_F (1-P_F)} \qquad (A5)$$

λ now has the probability density

$p_{\Lambda|H_k}(\lambda \mid H_k) =$

$$\sum_{j=0}^{n} \binom{n}{j} p_k^j q_k^{n-j} \delta(\lambda - (A_0 + a[2j-n])) \qquad (A6)$$

where $\delta(\cdot)$ is the Dirac delta function.

The global probabilities of missed detection and false alarm can now be found by integrating:

$$P_{MG} = \int_{-\infty}^{0} p_{\Lambda|H_1}(\lambda|H_1)\, d\lambda$$

$$= \sum_{j=0}^{n} \binom{n}{j} (1-P_M)^j P_M^{n-j}\, U[\, a(n-2j) - A_0] \qquad (A7\text{-}1)$$

$$P_{FG} = \int_{0}^{\infty} p_{\Lambda|H_0}(\lambda|H_0) d\lambda$$

$$= \sum_{j=0}^{n} \binom{n}{j} P_F^j (1-P_F)^{n-j}\, U[\, a(2j-n) + A_0] \qquad (A7\text{-}2)$$

Equation A7 can also be written as

$$P_{FG} = \sum_{j=0}^{n} \binom{n}{j} P_F^j (1-P_F)^{n-j}$$

$$U\left(j - \frac{\ln\left(\frac{P(H_0)}{P(H_1)}\right) + n \ln\left(\frac{1-P_F}{P_M}\right)}{\ln\left(\frac{1-P_M}{P_F}\right) + \ln\left(\frac{1-P_F}{P_M}\right)} \right) \qquad (A8\text{-}1)$$

$$P_M = 1 - \sum_{j=0}^{n} \binom{n}{j} (1-P_M)^j (P_M)^{n-j}$$

$$U\left(j - \frac{\ln\left(\frac{P(H_0)}{P(H_1)}\right) + n \ln\left(\frac{1-P_F}{P_M}\right)}{\ln\left(\frac{1-P_M}{P_F}\right) + \ln\left(\frac{1-P_F}{P_M}\right)} \right) \qquad (A8\text{-}2)$$

Non-equal variances We present an algorithm to find the error probabilities for the case of non-equal noise variances of the detectors. We consider first the second term in equation A-1:

$$\lambda_1 = \sum_{i=1}^{n} \left(\frac{1}{2}(A_i + B_i) \right) u_i = \sum_{i=1}^{n} \gamma_i$$

with

$$p_{\Gamma_i|H_0}(\gamma_i \mid H_0) = P_{Fi}\, \delta\left(\gamma_i - \frac{A_i+B_i}{2}\right) +$$

$$(1 - P_{Fi})\, \delta\left(\gamma_i + \frac{A_i+B_i}{2}\right) \qquad (A9)$$

due to statistical independence of the terms,

$$p_{\Lambda_1|H_0}(\lambda_1 \mid H_0) = p_{\Gamma_1}(\gamma) * p_{\Gamma_2}(\gamma) * \cdots * p_{\Gamma_n}(\gamma) \qquad (A10)$$

We consider the set of 2^n binary tuples that can appear as an input vector to the DFC. We use the index $k = 0, 1, \ldots, 2^n-1$ to distinguish between the different tuples. The probability distribution of λ_1 under assumption H_0 is

$$p_{\Lambda_1|H_0}(\lambda_1 \mid H_0) = \sum_{k=0}^{2^n-1} W_{k|H_0}\, \delta(z - \omega_k) \qquad (A11)$$

where

$$W_{k|H_0} = \prod_{i=1}^{n} (P_{Fi})^{\frac{1+u_{ki}}{2}} (1-P_{Fi})^{\frac{1-u_{ki}}{2}}$$

$$= \left[\prod_{i=1}^{n} (1-P_{Fi}) \right] \prod_{i=1}^{n} \left(\frac{P_{Fi}}{1-P_{Fi}} \right)^{\frac{1+u_{ki}}{2}} \qquad (A12)$$

and u_{ki} is the i^{th} bit of the k^{th} tuple,

and

$$\omega_k = \sum_{i=1}^{n} \frac{A_i+B_i}{2} u_{ki} \qquad (A13)$$

To find the probability of false alarm, we shall sum all weights W_k such that their corresponding $\omega_k > -A_0$. The calculation of the probability of missed detections follows a similar path.

References

Bar-Shalom, Y., Fortmann, T.E (1988): *Tracking and Data Association*, Boston, MA: Academic Press.

Chair, Z., Varshney, P.K. (1986): "Optimal Data Fusion in Multiple Sensor Detection Systems," *IEEE Trans. Aer. Elec. Sys.*, Vol. AES-22, no. 1, pp. 98-101.

Kam, M., Naim, A., Atteson, K. (1988): "SMART - SymMetric Adaptive Resonance Theoretic Model for Binary Pattern Classification," *Proc. 1988 Conf. Inf. Sci. Sys.*, Princeton, NJ.

Kohonen, T. et al. (1988): "Phonetic Typewriter for Finnish and and Japanese," *Proc. 1988 IEEE ICASSP*, NY, pp. 607-610.

Reibman, A.R., Nolte, L.W. (1987): "Optimal Detection Performance of Distributed Sensor Systems," *IEEE Trans. Aer. Elec. Sys.*, Vol. AES-23, No. 1, pp. 24-30.

Reibman, A.R., Nolte, L.W. (1988): " On Determining the Design of Fusion Detection Networks," *Proc. 27th Conference on Decision and Control*, Austin, Texas, Vol. 3, pp. 2474-2478.

Sage, A.P., Melsa, J.L. (1971) *Estimation Theory*, New York, NY: McGraw Hill

Tenney, R.R., Sandell, N.R. (1981): "Detection with Distributed Sensors," *IEEE Trans. Aer. Elec. Sys.*, Vol. AES-17, no. 4, pp. 501-509.

Thomopoulos, S.C.A., Viswanathan, R., Bougoulias, D.C. (1987): "Optimal Decision Fusion in Multiple Sensor Systems," *IEEE Trans. Aer. Elec. Sys.*, Vol. AES-23, No. 5, pp. 644-653.

Tsitsiklis, J.N., Athans, M. (1985): "On the Complexity of Decentralized Decision Making and Detection Problems," *IEEE Trans. Automatic Control*, Vol. AC-30, number 5, pp. 440-446.

THE 28th IEEE CONFERENCE ON DECISION AND CONTROL

Hyatt Regency, Tampa
Tampa, Florida
December 13-15, 1989

NEUROMORPHIC ADAPTIVE CONTROL

[19]

Izhak Bar-Kana and Allon Guez
ECE Dept., Drexel University, Philadelphia, PA 19104

ABSTRACT

A neuromorphic unsupervised parallel distributed adaptive controller for a class of nonlinear systems is proposed. It is shown to provide bounded tracking and asymptotic regulation following a class of 'teacher models.' A double link planar manipulator example is provided for a simulative illustration of the properties of the

1. PROBLEM FORMULATION

An unsupervised distributed parallel computing architecture is proposed for the adaptive control of nonlinear dynamic systems of the class

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \qquad (1)$$

$$y(t) = C(x)x(t) + D(x)u(t) \qquad (2)$$

with some degree or other of uncertainty, where $x(t) \in R^n$ is the plant state vector, $y(t) \in R^m$ is the output vector, and $u(t) \in R^m$ is the input command vector, and where $A(x)$, $B(x)$, $C(x)$, and $D(x)$ are uniformly bounded matrices of corresponding dimensions. This representation includes the linear time invariant systems and allows for extentions of the following presentation to certain classes of nonlinear systems like manipulators.

Figure 1 presents a schematic illustration of a common problem in modern control. If perfect knowledge on the parameters Θ of the linear plant is available, one has many ways to design his "best" controller represented here by the fixed gains K. Since perfect knowledge is not available, in general, one uses some estimate $\hat{\Theta}$ of the parameters of the plant to design the "best" controller $\hat{K}$, based on this estimate. The controller $\hat{K}$ must control the real-world plant Θ, and it is not necessarily a good controller for this real plant. Still, there is some domain where any *fixed* gains $\hat{K}$ would guarantee, at least, stability of the control system. However, in order to achieve better performance, modern control usually employs time-varying nonlinear, adaptive, or intelligent algorithms to compute dynamic estimates of the parameters $\hat{\Theta}(t)$ of the plant, or directly the parameters of the controller, which are thus nonstationary or nonlinear functions $\hat{K}(t)$. But before mentioning performance, how can we guarantee stability of these nonstationary systems. Intuitively, one may assume that if the nonstationary gains leave the stability region that was established for the fixed gains, one might get into troubles. Unfortunately, stability with nonstationary gains is not guaranteed, in general, even if one is cautious and does impose upon the variable gains the bounds of stability that were established for the fixed gains K [1].

Therefore, nonstationary or nonlinear controllers must usually assume slow variation of the estimated parameters and of the control gains (which then makes them "almost" fixed). Most adaptive methods, in particular, require external persistent excitation (to guarantee convergence of the parameters), and assume the prior knowledge or an upper bound on the order of the unknown controlled plant. Prior knowledge on the pole-excess in the plant is also needed. This knowledge is needed for design and then used to implement observer-based controllers of the same order as the plant [2].

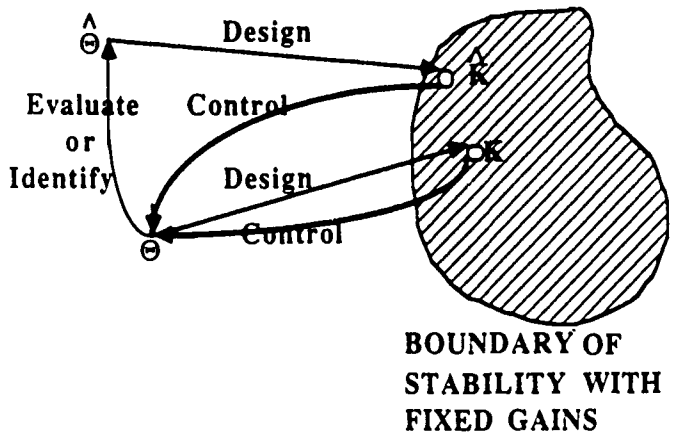


Figure 1: The modern control problem.

We propose a nonlinear neuromorphic adaptive *controller* that maintains stability under the same bounds that would guarantee stability with *fixed* linear controllers. The proposed controller is depicted in Figure 2 below: It consists of a teacher model which incorporates the knowledge regarding the desired input/output plant response as well as the repertoir of reference commands that the system may be subjected to.

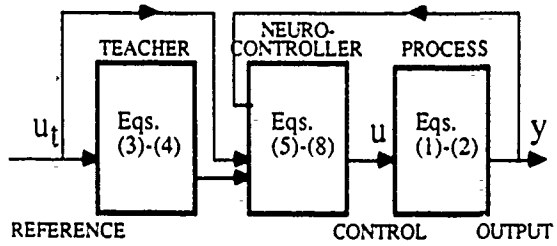


Figure 2: Proposed Neurocontroller

The teacher dynamic model is assumed to have the following representation:

$$\dot{x}_t(t) = A_t(x_t)x_t(t) + B_t(x_t)u_t(t) \qquad (3)$$

$$y_t(t) = C_t(x_t)x_t(t) + D_t(x_t)u_t(t) \qquad (4)$$

where $x_t(t) \in R^{n_t}$ is the state vector, $y_t(t) \in R^m$ is the output vector, and $u_t(t) \in R^m$ is the input command vector, and where $A_t(x_t)$, $B_t(x_t)$, $C_t(x_t)$, and $D_t(x_t)$ are uniformly bounded matrices of corresponding dimensions. It is emphasized that the dimension of the model is unrestricted, except that $\dim(y_t) = \dim(y) = m$.

This paper is based upon research supported in part by Drexel University's Stein Fellowship Foundation and by AFOSR grant No. 890010

The parallel distributed adaptive controller has structure similar to the LMS adaptive layer [3], and to many other neurocontroller architectures [4]. It receives the input 'features' vector $f(u_t, x_t, y)$ and generates as an output the process control vector

$$u(t) = K(t)f(u_t, x_t, y) \qquad (5)$$

where K(t) is the adaptive gain matrix of appropriate dimension. Each K_{ij} gain monitors the sensitivity of the i-th control, namely u_i, to the j-th feature of the system, namely $f_j(u_t, x_t, y)$ and may be viewed as the state of the *ij*-th neuron (Fig. 3).

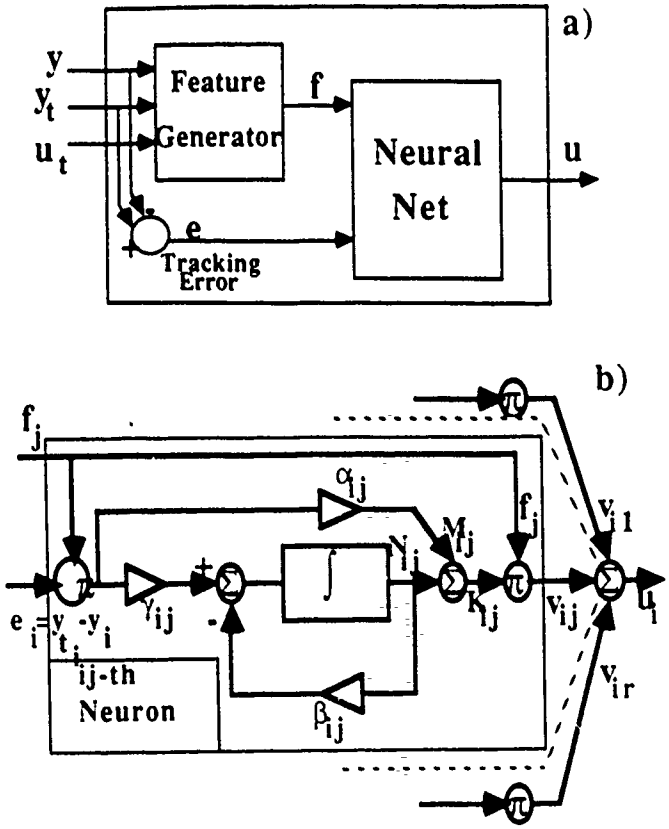


Figure 3 - Neurocontroller Architecture
a) General Block Diagram b)ij-th Neuron Architecture.

The K_{ij} gain adjusts its value independently of, and simultaneosly with all other gains, according to the adaptive rule:

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \qquad (6)$$

$$M_{ij}(t) = \alpha_{ij}(y_{t_i} - y_i)f_j \qquad (7)$$

$$\frac{d}{dt}N_{ij}(t) = -\beta_{ij}N_{ij}(t) + \gamma_{ij}(y_{t_i} - y_i)f_j \qquad (8)$$

where α_{ij}, β_{ij} and γ_{ij} are positive constants. We emphasize that the adaptation law (6)-(8) is similar in structure to the Widrow-Hoff rule [3] modified with a momentum term. We will show later that the modified structure also includes a supplementary term which takes care of the stabilization of unstable systems. The adaptation is performed in **parallel** and in a **distributed** fashion, that is, the K_{ij} **gain only needs data from the j-th feature and i-th output components.** Thus, very large scale dynamic systems may be considered for this controller. The adaptive gains consist of two terms: a "proportional" term, $M_{ij}(t)$, and an "integral" term, $N_{ij}(t)$. Notice that the role of the teacher (3)-(4) is to demonstrate to the adaptive controller what should be the appropriate and desired response y_t for any specified reference input u_t. Since the teacher's model is incorporated in the controller structure (Figure 2), it yields the so-called 'unsupervised' learning or adaptation.

For the above described control architecture we use recent results by Bar-Kana [5, 6] to show that, under some realistic assumptions, large classes of nonlinear plants of the form (1)-(2) with adaptive controllers of the form (5)-(8) can perform good trajectory tracking and also guarantee robust adaptive stabilization in the presence of *any* bounded input commands and input or output disturbances. Furthermore, this adaptive controller has ben shown to have good graceful degradation (or fault resistant) properties with control failure or fast changes of plant parameters [7].

MAIN RESULTS

We describe below a summary of our main results regarding the performance of the controller in (6), (7), and (8). We will assume that if a plant is stabilizable, the resulting stable configuration is "**exponentially stable**" as defined below:

DEFINITION 1 (Hahn) [8]: Let the general nonlinear system be represented by the *n* th-order vectorial equation $\dot{x}(t) = f(x,t)$ and let x=0 be an equilibrium point. The equilibrium point is called "**exponentially stable**" if all solutions x(t) satisfy the relation $|x(t)| \le \alpha |x(0)| e^{-\beta t}$ for some scalars $\alpha > 0$ and $\beta > 0$.

THEOREM 1 [8]: Let the right hand of the equation $\dot{x}(t) = f(x,t)$ have bounded continuous first order partial derivatives. Let the equilibrium be **exponentially stable.** Then there exists a Lyapunov function V(x,t) which satisfies estimates of the form

$$1.1 \qquad \alpha_1 |x(t)|^2 \le V(x,t) \le \alpha_2 |x(t)|^2$$

$$1.2 \qquad \dot{V}(x,t) \le -\alpha_3 |x(t)|^2$$

$$1.3 \qquad \left|\frac{\partial V(x,t)}{\partial x_i}\right| \le \alpha_4 |x(t)|^2 \ , \ i=1,2,\ldots,n$$

for α_1, α_2, α_3, and α_4 positive constants.

Because we deal with nonlinear systems we cannot expect to show, like in linear time-invariant systems, existence of positive definite quadratic Lyapunov functions of the form $V(x) = x^T(t)Px(t)$ where P is constant and positive definite. However, after some experience with specific nonlinear systems like robots, and because we restrict our discussion to nonlinear systems linear in control of the form (1)-(2), we assume that exponential stability of the *autonomous* system (1), with $u(t) \equiv 0$, implies existence of Lyapunov functions V(x) which are not explicit functions of time. Also, since then we can always write $V(x) = x^T(t)P(x)x(t)$ and thus $\dot{V}(x) = x^T(t)[\dot{P}x) + P(x)A(x)+A^T(x)P(x)]x(t)$ or $\dot{V}(x) = -x^T(t)Q(x)x(t)$, we will use in the subsequent discution the following assumption:

ASSUMPTION 1: Exponential stability of the *autonomous* system (1) (with $u(t) \equiv 0$) implies existence of Lyapunov functions of the form $V(x) = x^T(t)P(x)x(t)$ and derivative of the form $\dot{V}(x) = -x^T(t)Q(x)x(t)$, where P(x) and Q(x) are positive definite for all $x \in$

After establishing the basic definitions and facts, we can start presenting the properties of the neuromorphic adaptive controller. A first result will illustrate the stabilizing properties of the main term of the nonlinear adaptive controller that we propose. To this end, let us assume that the plant

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \qquad (9)$$

$$y(t) = C(x)x(t) \qquad (10)$$

can be stabilized by some constant output feedback, K_y. In other words, the resulting (fictitious) closed-loop system is **exponentially stable** according to assumption 1. Let us define the

$$y_a(t) \triangleq y(t) + Du(t) \qquad (11)$$

where $D = K_y^{-1}$, and use the adaptive algorithm

$$u(t) = -K(t)y_a(t) \qquad (12)$$

with the integral adaptive gain K(t) = N(t) given by

$$\dot{N}(t) = y_a(t)y_a^T(t)\Gamma \qquad (13)$$

(where Γ is a selected positive definite scaling matrix) or (for each scalar gain)

$$\frac{d}{dt}N_{ij} = \gamma_{ij}y_{a_i}y_{a_j} \tag{14}$$

THEOREM 2: The simple algorithm (12)-(14) guarantees boundedness of all values involved in the adaptation process, namely, states, outputs, and adaptive gains, and asymptotically perfect regulation for the augmented system

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \tag{15}$$

$$y(t) = C(x)x(t) \tag{16}$$

$$y_a(t) = y(t) + Du(t) = C(x)x(t) + Du(t) \tag{17}$$

such that $y(t) \to 0$ *and* $y_a(t) \to 0$ as $t \to \infty$. The gains are bounded and ultimately reach constant values (Appendix A). It is worth mentioning that K_y can be any stabilizing constant gain. Actually, some estimate of the maximal admissible gain is sufficient to guarantee stability of the adaptive controller, with no dependence on the rate of variation of the gains.

We want to use the stabilizing nonlinear algorithm (12), in combination with other adaptive terms in order to implement a stable trajectory-following adaptive algorithm. Let the teacher generate the desired trajectories that the plant must follow. Let the feature vector

$$f(u_t, x_t, y) = [\,(y_t - y)^T,\ x_t^T,\ u_t^T\,]^T \tag{18}$$

where $f(u_t, x_t, y)$ uses all values that can be measured, like the input commands, the teacher's states, and the tracking errors. We could try to use the adaptive algorithm

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \tag{19}$$

$$M_{ij}(t) = \alpha_{ij}(y_{t_i} - y_i)f_j \tag{20}$$

$$\frac{d}{dt}N_{ij}(t) = \gamma_{ij}(y_{t_i} - y_i)f_j \tag{21}$$

for perfect tracking in idealistic environments. However, we are aware that (19)-(21) must further be adjusted to be applicable in realistic environments. It is clear that the perfect integrator in (21) increases without bound whenever perfect tracking is not possible. Since the nonlinear system includes uncertainties that we do not assume to know or identify, and since moreover, input and output disturbances may usually be present, we do not try to proof perfect tracking, which could be obtained in some ideal situation, but rather concentrate onrobust stability under nonideal conditions. By *"robust stability"* we mean boundness of all values involved in the adaptation process like states, adaptive gains and errors, and tracking with arbitrarily small tracking errors. Perfect tracking in idealistic conditions is a particular case of the general robust tracking under nonideal conditions. The only modification of (19)-(21) needed to guarantee robustness of the adaptive system is the addition of the by-pass term $-\beta_{ij}N_{ij}(t)$ to the right term of (21), which gives the complete adaptive algorithm (5)-(8). It can be shown that under the assumptions of theorem 2, the adaptive algorithm (5)-(8) guarantees *robust stability* of the system (15)-(17) [9].

It is also worth mentioning that we assumed output stabilizability via constant feedback only for a simple introduction of parallel feedforward . Let us assume now that the plant needs some general dynamic configuration to reach stability. Specifically, if $G(\cdot)$ is some nonlinear system of the form (9)-(10) and if $H: \{A_f, B_f, C_f, D_f\}$ is a stabilizing controller, the adaptive algorithm (5)-(8) guarantees *robust stability* of the augmented system $G_a(\cdot) = G(\cdot) + H^{-1}$ [9]. More important, when nonphysical *improper* linear controllers H are desireded to control the nonlinear plant, we can use their *proper inverse* in parallel with the plant. This way, we only use the knowledge on the *existence* of an improper controller and actually use a proper configuration in parallel with our plant. Specifically, as in the case of nonlinear robotic manipulators, PD controllers of the form $H(s) = K(1+qs)$ can stabilize the manipulators, and if K is very large, we can get very good tracking in ideal situation. However, in practice we do not want to employ either differentiators or high gains in control loops and indeed, the adaptive controller only uses the knowledge on their mere *existence*, to implement simple first order poles of the form $H^{-1}(s) = \frac{D}{1+qs}$ in parallel with the plant, where $D = K^{-1}$ can be a very small gain.

Notice that we do not guarantee any more that the plant is perfectly tracking, because the best we can obtain is $y_a(t) = y(t) + Du(t) \to y_t(t)$, although we actually want $y(t) \to y_t(t)$. Still, if the maximal admissible gain K_{max} is large compared with the gain of the plant, then we can use $D = K_{max}^{-1}$ and get $y_a(t) = y(t) + Du(t) \approx y(t)$. For a quite common example, assume that the gain of the plant is 10 and the maximal stabilizing gain $K_{max} = 100$. Then we use $D = 1/100 = 0.01$ in parallel with 10 and, as the illustrations bellow and the references show, $y_a(t) \approx y(t)$ for all practical purposes.

ROBOTIC EXAMPLE

In this section we apply the proposed neurocontroller (5)-(8) to position control of a double link planar manipulator [10]. This example contains all major dynamical components such as gravity, Coriolis and centripetal terms.

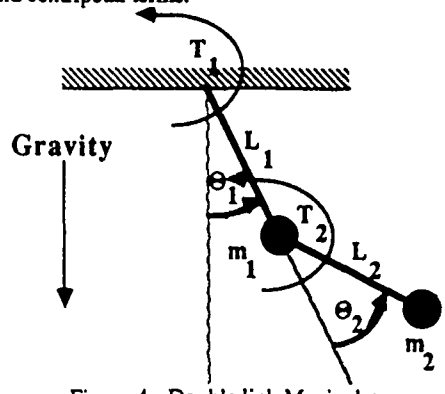


Figure 4 - Double link Manipulator

Figure 4 describes the double link system configuration. The equations of motion are:

$$u_1 = D_{11}\ddot{\Theta}_1 + D_{12}\ddot{\Theta}_2 + D(\dot{\Theta}_2^2 + 2\dot{\Theta}_1\dot{\Theta}_2) + D_1 \tag{22}$$

$$u_2 = D_{12}\ddot{\Theta}_1 + D_{22}\ddot{\Theta}_2 - D\dot{\Theta}_1^2 + D_2 \tag{23}$$

where

$L_1 = 0.1$ m, $L_2 = 0.1$ m, $m_1 = 0.6$ kg, $m_2 = 0.6$ kg, $g = 10$ m/s^2,

$D_{11} = (m_1 + m_2)L_1^2 + m_2L_2^2 + 2m_2L_1L_2\cos\Theta_2$,

$D_{12} = m_2L_2^2 + m_2L_1L_2\cos\Theta_2$,

$D_{22} = m_2L_2^2$,

$D = -m_2L_1L_2\sin\Theta_2$,

$D_1 = (m_1 + m_2)gL_1\sin\Theta_1 + m_2gL_2\sin(\Theta_1 + \Theta_2)$,

$D_2 = m_2gL_2\sin(\Theta_1 + \Theta_2)$.

and where g is gravity, u_1 and u_2 are torques at the first and second joints, m_1 and m_2 are masses, and L_1 and L_2 are lengths of the first and second link correspondingly.

We only employ position measurements and the outputs of the plant are

$$y_1 = \Theta_1 \tag{24}$$

$$y_2 = \Theta_2 \tag{25}$$

For the teacher we used the simple decoupled model:

$$\dot{x}_t(t) = \begin{bmatrix} -25 & 0 \\ 0 & -25 \end{bmatrix} x_t(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_t(t) \tag{26}$$

$$y_t(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_t(t) \qquad (27)$$

and the adaptive control system was tested with demanding square-wave input commands. In paralel with the plant (22)-(25) we employ the supplementary feedforward [5, 11]

$$y_s(s) = \frac{\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}}{1+s/10} u(s) \qquad (28)$$

and the adaptation coefficients were

$$\alpha_{ij} = \gamma_{ij} = 100000.; \quad \beta_{ij} = 0.01 \qquad (29)$$

Results of simulations are shown in Figure 5. Figure 5a compares the first plant and model output, and figure 5b shows the second output. Figure 5c shows, for ilustration, the behavior of the adaptive gains $K_{11}(t)$ and $K_{22}(t)$. It can be seen how the adaptive gain moves up-and-down in accord with the specific operational needs. Figure 5d represents the norm of all plant states, to show that no hidden state diverges. All values remained bounded while the plant tracked with small errors, although we used *only position* (no velocity and no acceleration) output measurements.

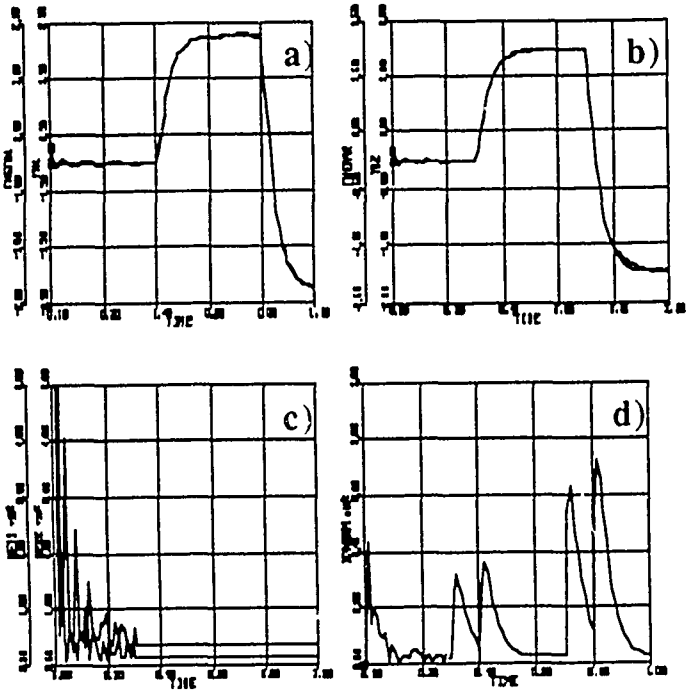


Figure 5. Example: (a) Output $y_{t_1}(t)$, $y_1(t)$; (b) Output $y_{t_2}(t)$, $y_2(t)$; (c) Gains $K_{11}(t)$, $K_{22}(t)$; (d) Norm of all plant states.

CONCLUSIONS

This paper presents a neuromorphic computing architecture for adaptive control. Starting with some prior assumptions about stabilizability of the plants it results in a stable unsupervized architecture. The feasibility of the method is demonstrated on an example of robotic manipulator.

APPENDIX A - PROOF OF THEOREM 2

It is easy to see that by using the controller

$$u_p(t) = -K_e y_a(t) + u_{pc}(t) \qquad (A.1)$$

with (15)-(17), we get the closed-loop system (Figs. 6 and 7)

$$\dot{x}(t) = A_c(x)x(t) + B_c(x)u_c(t) \qquad (A.2)$$

$$y_a(t) = C_c(x)x(t) + D_c u_c(t) \qquad (A.3)$$

where

$$A_c(x) = A(x) - B(x)K_{ec}(x)C(x) \qquad (A.4)$$

$$K_{ec}(x) = K_e[\, I + DK_e \,]^{-1} = [\, I + K_eD \,]^{-1}K_e \qquad (A.5)$$

$$B_c(x) = B(x)[\, I + K_eD \,]^{-1} \qquad (A.6)$$

$$C_c(x) = [\, I + DK_e \,]^{-1}C(x) \qquad (A.7)$$

$$D_c = [\, I + DK_e \,]^{-1}D = D[\, I + K_eD \,]^{-1} \qquad (A.8)$$

where $D > 0$ and $D_c > 0$, which means that D and D_c are positive definite.

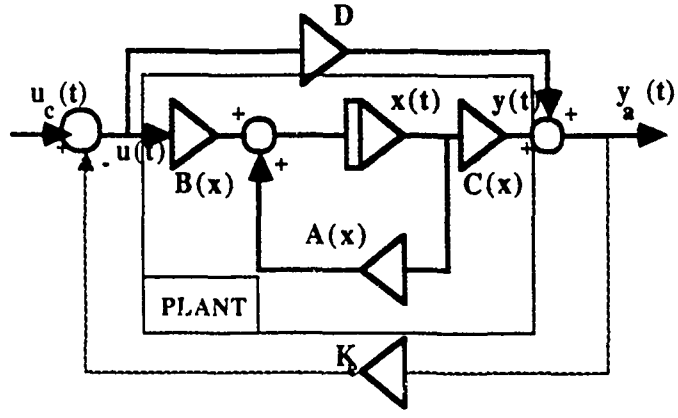


Figure 6: The closed-loop configuration.

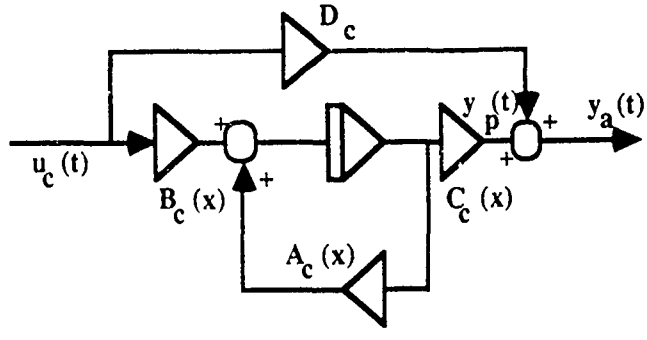


Figure 7: The equivalent closed-loop system.

DEFINITION 2. The closed-loop system (A.2) - (A.3) is called "strictly passive" (SP) if it satisfies the following relations

$$\dot{P}(x) + P(x)A_c(x) + A_c^T(x)P(x) = -Q(x) - L^T(x)L(x) \qquad (A.9)$$

$$P(x)B_c(x) = C_c^T(x) - L^T(x)W \qquad (A.10)$$

$$D_c + D_c^T = W^TW \qquad (A.11)$$

for some uniformly bounded positive definite matrices, P(x) and Q(x), and some matrices $L(x) \in R^{mxn}$ and $W \in R^{mxm}$, and where

$$\dot{P}(x)=\frac{dP(x)}{dt}=\frac{\partial P(x)}{\partial x}\frac{dx}{dt}.$$

DEFINITION 3: The augmented open-loop system (15) - (17) is called "almost strictly passive" (**ASP**) if there exists a positive definite static feedback matrix K_e (unknown and not needed for any implementation) such that the resulting closed-loop system (A.2)-(A.3) is strictly passive (**SP**).

LEMMA 1: If K_y stabilizes (9)-(10), then the augmented open-loop (15)-(17) is "almost strictly passive" (**ASP**) [6].

The lemma and the various definitions will be used for the proof of Theorem 2. The adaptive control system of (15)-(17) with the adaptive controler (12)-(13) is

$$y_a(t) = C(x)x(t) + Du(t) = C(x)x(t) - DN(t)y_a(t) \quad (A.12)$$
$$y_a(t) = C(x)x(t) - D[N(t) - K_e]y_a(t) - DK_e y_a(t) \quad (A.13)$$
$$y_a(t) = [\,I + DK_e\,]^{-1}C(x)x(t) - [\,I + DK_e\,]^{-1}D[N(t) - K_e]y_a(t) \quad (A.14)$$

$$y_a(t) = C_c(x)x(t) - D_c[N(t) - K_e]y_a(t) \quad (A.15)$$

$$\begin{aligned}\dot{x}(t) &= A(x)x(t) - B(x)N(t)y_a(t)\\ &= A(x)x(t) - B(x)K_e y_a(t) + B(x)K_e y_a(t) - B(x)N(t)y_a(t)\\ &= A(x)x(t) - B(x)K_e C_c(x)x(t) + B(x)K_e D_c[N(t) - K_e]y_a(t)\\ &\quad - B(x)[N(t) - K_e]y_a(t)\\ &= [A(x) - B(x)K_{ec}C(x)]x(t) - B_c(x)[N(t) - K_e]y_a(t)\\ &= A_c(x)x(t) - B_c(x)[N(t) - K_e]y_a(t)\end{aligned} \quad (A.16)$$

We must prove stability of all the values involved in the adaptation process, like states, outputs, gains. Let us select the quadratic Lyapunov function

$$V(t) = x^T(t)P(x)x(t) + tr[(N(t) - K_e)\Gamma^{-1}(N(t) - K_e)^T] \quad (A.17)$$

where *tr* denotes trace and where K_e denotes the unknown ideal gain that makes the plant strictly passive.. Notice that the second term in (A.17) is only a short notation for the sum of all terms of the form $[N_{ij}(t) - K_{ij}]^2/\gamma_{ij}$. Therefore, (A.17) is a positive definite quadratic function of all states and adaptive gains. The derivative of the Lyapunov function along the trajectories of (13) and (A 16) is

$$\begin{aligned}\dot{V}(t) &= x^T(t)\dot{P}(t)x(t) + x^T(t)P(x)\dot{x}(t) + \dot{x}^T(t)P(x)x(t)\\ &\quad + tr[(N(t) - K_e)\Gamma^{-1}\dot{N}^T(t)] + tr[\dot{N}(t)\Gamma^{-1}(N(t) - K_e)^T]\\ &= x^T(t)[\dot{P}(t) + P(x)A_c(x) + A_c^T(x)P(x)]x(t)\\ &\quad - x^T(t)P(x)B_c(x)[N(t) - K_e]y_a(t)\\ &\quad - y_a^T(t)[N(t) - K_e]^T B_c^T(x)P(x)x(t)\\ &\quad + tr[(N(t) - K_e)\Gamma^{-1}\Gamma y_a(t)y_a^T(t)]\\ &\quad + tr[y_a(t)y_a^T(t)\Gamma\Gamma^{-1}(N(t) - K_e)^T]\end{aligned} \quad (A.18)$$

By using (A.9)-(A.11) we get

$$\begin{aligned}\dot{V}(t) &= -x^T(t)Q(x)x(t) - x^T(t)L^T(x)L(x)x(t)\\ &\quad - x^T(t)C_c^T(x)[N(t) - K_e]y_a(t) - y_a^T(t)[N(t) - K_e]^T C_c(x)x(t)\\ &\quad - x^T(t)L^T(x)W[N(t) - K_e]y_a(t) - y_a^T(t)[N(t) - K_e]^T W^T L(x)x(t)\\ &\quad + x^T(t)C^T(x)[N(t) - K_e]y_a(t) + y_a^T(t)[N(t) - K_e]^T C_c(x)x(t)\\ &\quad - y_a^T(t)[N(t) - K_e]^T D_c^T[N(t) - K_e]y_a(t)\\ &\quad - y_a^T(t)[N(t) - K_e]^T D_c[N(t) - K_e]y_a(t)\\ &= -x^T(t)Q(x)x(t) - x^T(t)L^T(x)L(x)x(t)\\ &\quad - x^T(t)L^T(x)W[N(t) - K_e]y_a(t) - y_a^T(t)[N(t) - K_e]^T W^T L(x)x(t)\\ &\quad - y_a^T(t)[N(t) - K_e]^T W^T W[N(t) - K_e]y_a(t)\\ &= -x^T(t)Q(x)x(t)\\ &\quad - \{L(x)x(t) - W[N(t) - K_e]y_a(t)\}^T\{L(x)x(t) - W[N(t) - K_e]y_a(t)\}\\ &\le 0\end{aligned} \quad (A.19)$$

Notice that we only claim that the derivative of the Lyapunov function in (A.19) is positive *semidefinite*, namely $\dot{V}(t) \le 0$. The selected positive definite quadratic form of V(t) then guarantees that all states *and* adaptive gains are bounded. The trajectories of the adaptive control system finally reach the region defined by $\dot{V}(t) \equiv 0$ [13]. It is easy to see that $\dot{V}(t) \equiv 0$ implies $x(t) \equiv 0$. Therefore, if we ignore the adaptive gains, the adaptive control system is asymptotically perfect regulator, but what about the gains? First, we proved already that the gains are bounded. Second, $x(t) \equiv 0$ implies $y(t) \equiv 0$ and $y_a(t) \equiv 0$, which implies $\dot{N}(t) \equiv 0$ and then $N(t)$ = constant. Therefore, N(t) ultimately reaches some constant value such that the plant is perfect regulator, as the theorem claims.

REFERENCES

[1] M. A. Aizerman and F. R. Gantmacher, *Absolute Stability of Regulator Systems*, Holden Day, San Francisco, 1964.

[2] K. J. Åström, "Theory and Applications of Adaptive Control - A Survey," *Automatica*, Vol. 19, pp. 471-481, 1983.

[3] B. Widrow and S. Stearn, *Adaptive Signal Processing*, Prentice Hall, 1985.

[4] A. Guez, "Neurocontrollers," *NSF Workshop on Neurorobotics*, New Hampshire, 1988.

[5] I. Bar-Kana, "Parallel Feedforward and Simplified Adaptive Control," *International Journal of Adaptive Control and Signal Processing*, Vol. 1, No. 2, pp 95-109, 1987.

[6] I. Bar-Kana, I., *On Passivity of a Class of Nonlinear Systems*, Technical Report, Drexel University, 1989.

[7] W. Morse and K. Ossman, "Flight Control Reconfiguration Using Model Reference Adaptive Control," *Proceedings of 1989 American Control Conference*, Pittsburgh, Pennsylvania, pp. 159-164.

[8] W. Hahn, W., *Stability of Motion*, Springer Verlag, New York, 1967.

[9] I. Bar-Kana and A. Guez "Simple Adaptive Control for a Class of Nonlinear Systems with Application to Robotics," *Int. J. Control* (forthcoming).

[10] A. Guez, *Optimal Control of Robotic Manipulators*, PhD Disertation, University of Florida, Gainsville, Florida, 1983.

[11] I Bar-Kana, "On Positive Realness in Multivariable Stationary Linear Systems," *Proceedings of 1989 Conference on Informational Sciences and Systems - CISS '89*, Baltimore, Maryland, pp. 383-388; also (full report) Technical Report, Drexel University, 1989.

[12] J. C. Willems, "Dissipative Dynamical Systems," in *Archive for Rational Mechanics and Analysis*, Vol. 45, pp.321-393,

[13] J. LaSalle, "Stability of Noautonomous Systems," *Nonlinear Analysis Theory, Methods and Applications*, Vol. 1, No. 1, pp. 83-91, 1981.

[20]

THE ROLE OF A PRIORI KNOWLEDGE OF PLANT DYNAMICS IN NEUROCONTROLLER DESIGN

J.W. Selinsky Allon Guez

Drexel University
Department of Electrical and Computer Engineering
32nd and Chestnut St., Philadelphia, PA 19104

ABSTRACT

In classical model-based techniques, construction of a control architecture capable of accurately tracking a desired trajectory throughout the state-space requires a detailed knowledge of the controlled system's dynamics. However, the system dynamics are rarely fully known at the time of controller design. In this paper, we modify our earlier neurocontroller architecture [1] so as to guarantee the performance of the neurocontroller. An innovative aspect of this architecture is in the use of a priori knowledge of the general structure of the system's dynamics. The knowledge is utilized for the selection of Exploratory Schedules (ES) to excite selected subsets of the dynamics. The controller does not require a priori knowledge of the exact system dynamics, as they are learned online. Nor does it assume the existence of an explicit external teacher. The developed control architecture is also not limited to tracking of a prespecified trajectory. The architecture is developed through application to the control of a robot manipulator.

1. INTRODUCTION

A number of different neural network models and neural learning schemes have been applied to system controller design with varying degrees of success [2],[3],[4],[5],[6],[7],[8],[9]). In general, the work surveyed assumed very little a priori knowledge of the structure of the open loop system and in none of the cases was there proof of stability of the closed loop system. In system controller design, proof of stability of the closed loop system is essential. Furthermore, if neurocontrollers are to successfully compete with currently available controllers, they must be shown to have performance that is at least comparable if not superior. In this work we incorporate a priori knowledge of the plant's dynamics in the neurocontroller design. It is shown that assuming a dynamic model for the plant allows the design of a neurocontroller with guaranteed performance. The dynamic model used for the development of the neurocontroller is that of a rigid robot manipulator.

Selection of training examples to provide efficient learning is also an issue which is rarely addressed. In this article, training example selection for neurocontrollers is provided with the introduction of Exploratory Schedules (ES). ES are trajectories which are designed so as to produce examples which allow the neurocontroller architecture to learn efficiently by exciting selected subsets of the dynamics.

Research supported in part by AFOSR grant # 890010

A general block diagram of the proposed system for a robot manipulator is shown in figure 1.1 . It consists of the Exploratory Schedule Generator, Neurocontroller and associated Learning Algorithm, the Robot, and a mechanism which allows selecting between the User / Path Planner and ES Generator as the originator of the desired trajectory (qd).

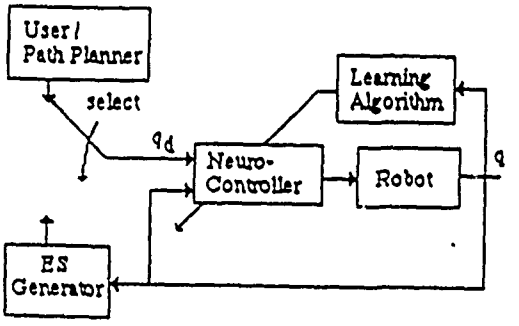


figure 1.1: Block diagram of the proposed system.

In section 2, a priori knowledge of the incompletely known plant dynamics are utilized such that the closed loop system consisting of the Neurocontroller, Learning Algorithm, and Robot can be shown to asymptotically track a desired trajectory. Furthermore, a priori knowledge is also used to select exploratory schedules which can be shown to provide asymptotic identification of the dynamics that are not a priori known. Section 3 gives results of simulations for a 2 DOF manipulator. Section 4 concludes the article.

2. Neurocontroller Design for a Rigid Robot Manipulator

Robot Dynamic Model

The closed form dynamics obtained by the Lagrange-Euler formulation has the general matrix form of [10]

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad (2.1)$$

where

- $D(q)$ is an n x n matrix of inertial terms,
- $C(q,\dot{q})$ is an n x n matrix of coriollis and centrifugal terms ,
- $G(q)$ is an n x 1 vector of gravitational terms,
- q is an n x 1 vector of joint coordinates,
- τ is an n x 1 vector of forces/torques,
- n is the number of degrees of freedom.

The formulation results in n second-order, coupled differential equations. (2.1) can also be expressed as

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\dot{q},\ddot{q})\emptyset \qquad (2.2)$$

where $Y(q,\dot{q},\dot{q},\ddot{q})$ is an n x p matrix of known functions and $\emptyset$ is an p x 1 vector of weighting constants.

Neurocontroller Architecture

When (2.2) is written in terms of the individual torques at each joint, it can be viewed as a single layer linear network, where the inputs to the network are the $Y_{ij}(.)$ and the weights are the $\emptyset_j$ [1],[11].

Note that the $Y_{ij}(.)$ are transcendental algebraic functions of the manipulators states that are a priori known and may be realized via feedforward neural networks that are trained offline with a suitable learning algorithm (i.e. BEP [12]).

Let $q_d, \dot{q}_d, \ddot{q}_d$ designate the desired trajectory. Following [13] define the virtual reference trajectory $\dot{q}_r$, $\ddot{q}_r$, and the virtual trajectory velocity error $\dot{e}_r$

$$\dot{q}_r = \dot{q}_d - A e, \quad \ddot{q}_r = \ddot{q}_d - A \dot{e}, \quad \dot{e}_r = \dot{q} - \dot{q}_r = \dot{e} + Ae \qquad (2.3)$$

where $e[t] = q[t]-q_d[t]$ is the joint coordinate error, and A is a positive definite matrix with constant coefficients.

The output of the neurocontroller implements the control law

$$\tau_i = \sum_{j=1}^{p} Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]\hat{\emptyset}_j + K_{d_{ii}}\dot{e}_{r_i}, \qquad i = 1,2,\ldots,n, \qquad (2.4)$$

where $\hat{\emptyset}$ denotes the estimate of $\emptyset$, and the $K_{d_{ii}}$ are constant weights for the servo portion of the controller. Equation (2.4) has been shown to be asymptotically stable when the learning rule

$$\dot{\hat{\emptyset}}_j = -\sum_{i=1}^{n} \frac{1}{K_{a_{ii}}} Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]\dot{e}_{r_i}, \qquad j = 1,2,\ldots,p. \qquad (2.5)$$

is used [13].

Notice in equation (2.5) the similarity to the LMS learning rule (see [14]) where the weight change is proportional to the error e and the input features X. In equation (2.5) the input features are the $Y_{ij}(.)$ functions and the error is $\dot{e}_{r_i}$.

Figure 2.1 shows the internal structure of the neurocontroller for joint i. Each feedforward network module is trained to provide one of the $Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]$ functions. This training can be done offline since the $Y_{ij}[q,\dot{q},\dot{q}_r,\ddot{q}_r]$ functions are known a priori and are the same for all rigid robots of the same kinematics and number of degrees of freedom. The inputs to the neurocontroller are the components of the trajectories as required. The outputs of the neurocontroller are the control torques to be applied at each joint of the manipulator.

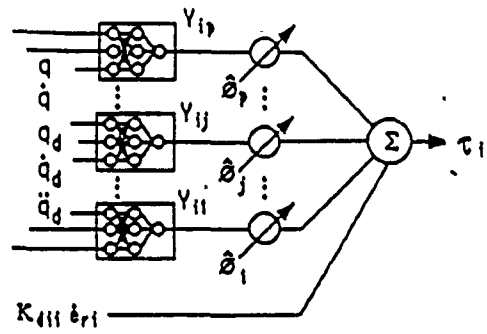


figure 2.1: Neurocontroller structure for joint 1.

Selection of Exploratory Schedules

Theorem [15]
If $p \leq n$, and if

$$\text{Rank} \left\{ \lim_{t \to \infty} Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d) \right\} = p$$

then the controller (eq. (2.4), (2.5)) guarantees global asymptotic tracking of the desired trajectory and identification of the weights.

An implication of the theorem is that if $p > n$, a sufficient condition for $k \leq n$ weights to be identified is that they are are not in the null space of

$$\lim_{t \to \infty} Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d)$$

Which implies that a desired trajectory may be chosen such that columns corresponding to the $k \leq n$ weights are linearly independent, which will guarantee that the weights are identified.

The exploratory schedule then consists of a sequence of desired trajectories which are designed to learn different components of the parameter estimation vector $\hat{\emptyset}$, where the number of desired trajectories is such that all p components of $\hat{\emptyset}$ are identified.

3. SIMULATION RESULTS

Dynamic Model for Manipulator with 2 DOF

The 2 DOF manipulator is shown in figure 3.1

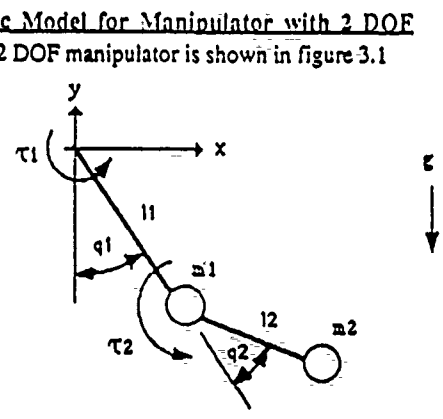


figure 3.1 Simulated 2 DOF manipulator.

Assuming the link mass to be concentrated at the tip of each link, the dynamics of the 2 DOF manipulator are

$$\tau_1 = D_{11}\ddot{q}_1 + D_{12}\ddot{q}_2 + 2C\dot{q}_1\dot{q}_2 + C\dot{q}_2^2 + G_1 \qquad (3.1)$$
$$\tau_2 = D_{12}\ddot{q}_1 + D_{22}\ddot{q}_2 - C\dot{q}_1^2 + G_2$$

where

$D_{11} = m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 \cos(q_2)$,
$D_{12} = m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)$,
$D_{22} = m_2 l_2^2$,
$C = m_2 l_1 l_2 \sin(q_2)$
$G_1 = g(m_1 + m_2) l_1 \sin(q_1) + g m_2 l_2 \sin(q_1 + q_2)$
$G_2 = g m_2 l_2 \sin(q_1 + q_2)$,

with $m_1 = m_2 = 10.0$, $l_1 = l_2 = 1.0$, $g = 9.81$.

Which can be written as the

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} & Y_{15} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} & Y_{25}\end{bmatrix}[\varnothing_1\ \varnothing_2\ \varnothing_3\ \varnothing_4\ \varnothing_5]^T, \qquad (3.2)$$

where

$Y_{11} = \ddot{q}_1$,
$Y_{21} = 0$,
$Y_{12} = Y_{22} = \ddot{q}_1 + \ddot{q}_2$,
$Y_{13} = \cos(q_2)(2\ddot{q}_1 + \ddot{q}_2) - \sin(q_2)(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)$,
$Y_{23} = \cos(q_2)\ddot{q}_1 + \sin(q_2)\dot{q}_1^2$,
$Y_{14} = \sin(q_1)$,
$Y_{14} = 0$,
$Y_{15} = Y_{25} = \sin(q_1 + q_2)$.

The true values of the weights are

$\varnothing_1 = (m_1 + m_2) l_1^2 = 20.0$ $\quad \varnothing_4 = g(m_1 + m_2) l_1 = 196.2$
$\varnothing_2 = m_2 l_2^2 = 10.0$ $\quad \varnothing_5 = g m_2 l_2 = 98.1$
$\varnothing_3 = m_2 l_1 l_2 = 10.0$.

The control law as specified by eq. (2.4) is

$$\tau_i = \sum_{j=1}^{p} Y_{ij}[q, \dot{q}, \dot{q}_r, \ddot{q}_r]\hat{\varnothing}_j + K_{d_{ii}}\dot{e}_{r_i}, \qquad (3.3)$$

where $\dot{e}_{r_i} = \dot{e}_i + e_i$, $K_{d_{11}} = 1000$, $K_{d_{22}} = 500$, and the learning algorithm as defined by (2.5) was used to update the weight estimates $\hat{\varnothing}_i$ with the adaptation constants $K_{a_{ii}} = 0.001$.

Nonlinear feedforward network modules were trained offline using the BEP learning algorithm [12] to approximate sin(x) and x*y, where x and y denote the inputs to the network. These modules were then combined to implement the $Y_{ij}[q, \dot{q}, \dot{q}_r, \ddot{q}_r]$ functions in (3.3).

Exploratory schedule for the 2 DOF Manipulator

In the 2 DOF case, p>n, and at most 2 weights can be guaranteed to be learned simultaneously. An exploratory schedule was selected such that different portions of the schedule would provide for learning different weights.

The selection of the exploratory schedule manipulator was accomplished by noting that columns 4 and 5 of the Y matrix in (3.2) are functions only of position, therefore selecting $\ddot{q}_{d1} = \ddot{q}_{d2} = \dot{q}_{d1} = \dot{q}_{d2} = 0.0$, and q_{d1}, q_{d2} such that columns 4 and 5 are nonzero as e_1, e_2, $\dot{e}_1$, $\dot{e}_2$ -> 0 will ensure that weights $\varnothing_4$ and $\varnothing_5$ are identified. Next, selecting $\ddot{q}_{d1} = \ddot{q}_{d2} = 0.0$, and q_{d1}, q_{d2}, $\dot{q}_{d1}$, $\dot{q}_{d2}$ such that column 3 is nonzero in the limit ensures identification of weight $\varnothing_3$. Then selecting $\ddot{q}_{d1}$, $\ddot{q}_{d2}$ such that columns 1 and 2 are nonzero ensures the identification of weights $\varnothing_4$ and $\varnothing_5$. The actual exploratory schedule is as shown in figures 3.2 to 3.4.

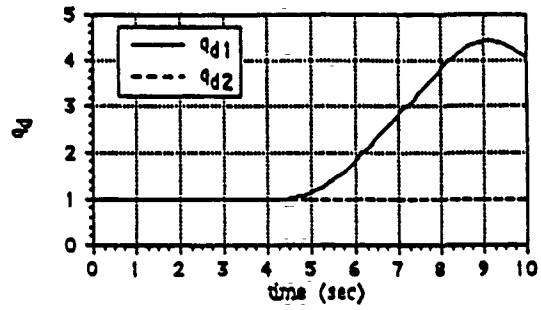


figure 3.2: Desired joint coordinates.

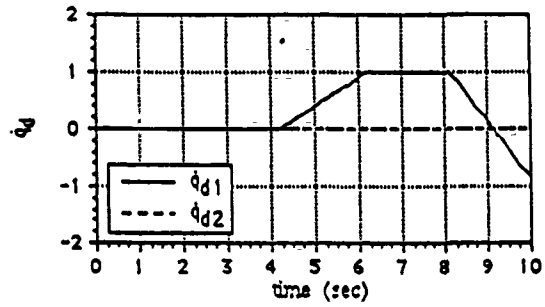


figure 3.3: Desired joint velocities.

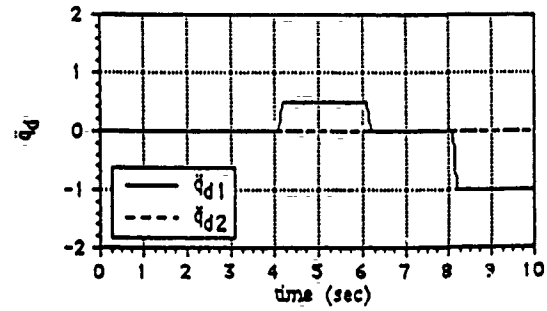


figure 3.4: Desired joint accelerations.

Theoretically, a weight is guaranteed to be identified when $e_1 = e_2 = \dot{e}_1 = \dot{e}_2 = 0.0$. However, in practice, due to noise, disturbances or the inability to wait until all transients have died out, exact tracking may not occur and small tracking errors may

lead to small errors in the identification of the weights. In this simulation, a weight is said to be identified when Maximum($|e_1|$, $|e_2|$, $|\dot{e}_1|$, $|\dot{e}_2|$) $\leq e_{tol}$, where $e_{tol} = 0.01$, and a further stipulation that attempted identification of each weight is to last at least 3 seconds. Furthermore, errors in weight identification may occur due to the feedforward network modules only approximating the Yij(.) functions.

The time period corresponding to identification of $\emptyset_4$ and $\emptyset_5$ is $0 \leq t < 4.2$. As can be seen in figures 3.5 to 3.8, the weight estimation error for $\emptyset_4$ and $\emptyset_5$ tends to zero as all joint errors fall below 0.01 at t=4.2, but does not reach zero. The time period corresponding to identification of $\emptyset_3$ is $4.2 \leq t < 8.3$. Again, the weight estimation error for $\emptyset_3$ approaches zero as all joint errors approach zero at t=8.3. weights $\emptyset_1$ and $\emptyset_2$ are identified during the time period $8.3 < t \leq 10$.

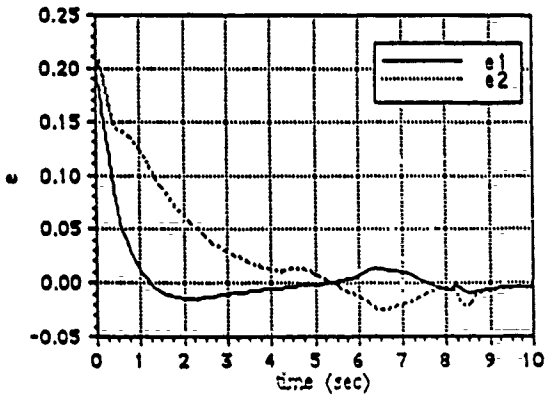


figure 3.5: Joint coordinate error.

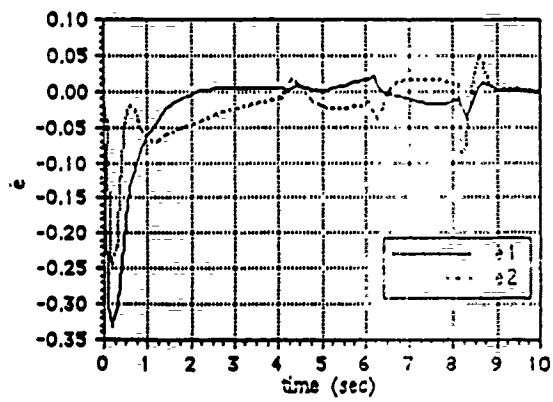


figure 3.6: Joint velocity error during.

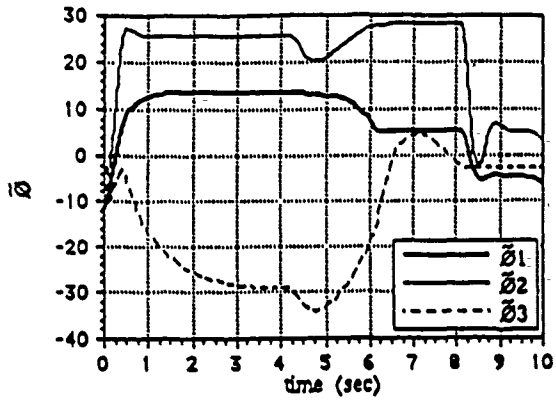


figure 3.7: Estimation error for weights 1,2, and 3.

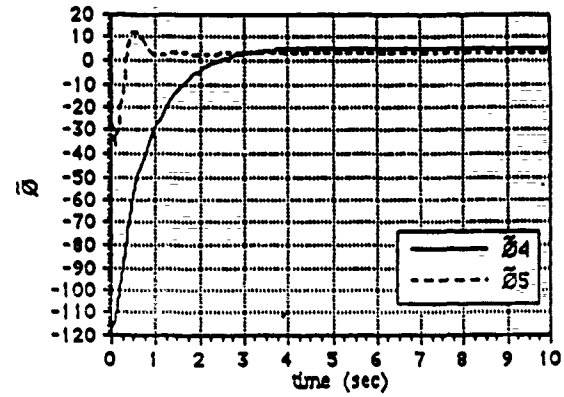


figure 3.8: Estimation error for weights 4 and 5.

4. Conclusion

We have shown that assuming a dynamic model for the plant allows the design of a neurocontroller with guaranteed performance. A neurocontroller has been proposed which utilizes available a priori knowledge of the plant dynamics to train a number of feedforward network modules. The output of these modules were combined online in an output layer to provide the control torques for a robot manipulator.

A procedure for the design of exploratory schedules for a 2 DOF manipulator was presented. Theoretically, the exploratory schedule, when used in conjunction with the proposed neurocontroller would guarantee identification of the dynamics that are not known a priori. However, as shown by the results of the simulations, some error in the identification does occur. This error is mainly attributed to the approximations provided by the feedforward networks. In previous work [13], the control architecture was employed using exactly computed values of the $Y_{ij}(.)$ functions rather than the approximations provided by the network modules. In that case, identification of the unknown dynamics was accomplished using the same exploratory schedule as presented here.

REFERENCES

[1] Guez, A. and Selinsky, J., "Neurocontroller design via supervised and unsupervised learning," to appear in the Journal of Intelligent and Robotics Systems, 1989.

[2] Widrow, B., and Smith, F.W., "Pattern recognizing control systems," *1963 Computer and Information Sciences Symposium Proceedings*, Washington, DC, 1964.

[3] Widrow, B., "The original adaptive broom balancer," *IEEE Conference on Circuits and Systems*, Philadelphia, PA, 1987.

[4] Barto, A.G., Sutton, R.S., and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Tranactions on Systems, Man and Cybernetics*, SMC 13, no. 5, 834-846, 1983.

[5] Psaltis, D., Sideris, A. and Yamamura, A., "Neural controllers," *Proceedings of the IEEE First International conference on Neural Networks*, San Diego, CA, 1987.

[6] Kawato, M., Furukawa, K. and Suzuki, R., "A hierarchical neural-network model for control and learning of voluntary movement," *Biological Cybernetics*,56, 1987.

[7] Guez, A., Eilbert, J., and Kam, M., "Neuromorphic architecture for control," *IEEE Control Systems Magazine*,April, 1988.

[8] Tolat, V.V., and Widrow, B., "An adaptive 'broom balancer' with visual inputs, " *Proceedings of the IEEE International conference on Neural Networks*, San Diego, CA, 1988.

[9] Guez, A., and Selinsky J., "A trainable neuromorphic controller," *Journal of Robotics Systems*, August, 1988.

[10] Paul, R.P., Robot Manipulators: Mathematics, Programming, and Control. Massachusetts: The MIT Press, 1981.

[11] Kawato, M., Furukawa, K. and Suzuki, R., "A hierarchical neural-network model for control and learning of voluntary movement," Biological Cybernetics,56, 1987.

[12] Rumelhart, D., Hinton, G.E. and Williams, R.L., "Learning internal representations by error propagation," In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*. MIT Press, 1986 .

[13] Slotine, J.J.E., and Li, W., "Adaptive manipulator control: a case study," IEEE Trans. Automatic Control, Vol. 33, no.11, p. 995, 1988.

[14] Widrow, B., and Stearns, S. D., Adaptive Signal Processing. New Jersey: Prentice-Hall, 1985.

[15] Guez, A., and Selinsky J., "Design of Exploratory Schedules in Learning and Adaptive Robot Control," accepted for publication in the *International Journal of Adaptive Control and Signal Processing*, 1989.

APPLICATION OF NEURAL NETWORK MODELLING FOR THE MUSCLE EMG TO TORQUE RELATION IN THE ANKLE JOINT

Neurosci Annual Meet, 11/ 1989 Phoenix, AZ

L.M.Kent [2,3], Z. Ahmad[1], A. Guez[1], W. Freedman[1,2,3]

[1]Dept of Electrical and Computer Engineering, Drexel University

[2]Biomedical Engineering Institute, Drexel University

[3]Moss Rehabilitation Hospital, Philadelphia, Pennsylvania

ABSTRACT

The application of a neural network approach to solution of the EMG-torque relation in the ankle joint under isometric, supine conditions is addressed in this study. The backpropagation neural network model was used in this analysis to simulate a multilayer perceptron for solution to the muscle EMG - joint torque mapping. Measurements from 6 muscle sites are entered into the model as the input, while the torque is entered into the model as the ideal output, to which the model output is compared. The incoming muscle signals may be considered the "intent" of the system, while joint torque is the "controlled" variable. It is expected that the results of this study will enable the application of neural network models as the adapter (processor, controller) for intent recognition systems in modelling intact human motor control about one joint and potentially, for applications, such as in multi-joint control of robotic manipulators, myoelectric control of prosthetic devices, and control of functional electrical stimulators.

A. INTRODUCTION

In biological organisms, movement is controlled through an amalagam of voluntary, stereotypic, and reflex actions. The particular mixture for any specific movement appears to be based on the functional task (goal) to be performed or the intent desired. Basic movement patterns (strategies) appear to be the result of a complex control system that interacts with sensory feedback mechanisms to produce movement which adapts to perform the functional task at hand. Biological organisms are real time, highly parallel and distributed systems which have adaptive ability, learning capability, and are governed by organizing principles.

Acknowledgement

Research partially supported by Grant AFOSR #890010 and Calhoun Research Endowment. We gratefully acknowledge R. Triolo, Research Director and the staff of the FNS Laboratory at Shriners Hospital, Philadelphia Unit,for use of laboratory facilities and technical expertise.

The development of devices in both robotics and in prosthetics for amputees have been similarly constrained by a lack of understanding of movements performed by biological organisms. The human amputee and paraplegic offer an important test bed from which an understanding of the human control of limbs can be studied. It is clear in the case of prosthetic and orthotic devices that movements must be anthropomorphic; however, for robotic systems, anthropomorphic operation is not necessary, although it is usually assumed since the function of robots is seen to be a substitution for the human operator. If a device (either biological or robotic), is to function in other than a pre-programmed mode, the communication between the user and the device must be accurate and reliable.

The problems that exist in achieving well designed, functional limbs are therefore related to our primitive knowledge of how human limbs operate. In both cases, the question arises concerning the basis upon which to build the controller of the limb motion. The need exists to develop a control scheme for a multi-joint limb based upon anthropomorphic mechanisms.

B. BACKGROUND

During the past 20 years, several prosthetic limbs have been developed which utilize spatial patterns of signals from muscles which remain after a limb amputation. In addition, the employment of electromyographic (EMG) signatures for controlling functional electrical stimulation (FES) in paraplegics in a manner that recognizes and executes the patient's intended limb functions and compensates for muscle fatigue is currently being developed. The control concept is that by sensing the activity of the muscles which would have supported the movement of the missing or paralyzed limb, the character of the intended movement of the prosthesis or orthosis can be determined. Sensing of muscle activity is achieved by recording the appropriate muscle (EMG) signals. This scheme has resulted in some partial success in devices such as the "Temple Arm"(32), the "Swedish Hand" (2), the"Case Western Reserve Arm" (23) and the "Utah Arm". In each case, the pattern of EMG activity was used to classify an intended limb movement. Later developments have included the "Drexel/Moss Knee" (27,33), Graupe's model of EMG as a scalar time series (8,9), and Hogan and Mann's model of muscle activity using maximum likelihood methods (15,16). Also, Saridis has developed an upper limb prosthesis based on pattern recognition (29).

In each of these developments, assumptions have been made concerning the relationship between EMG activity and force developed about a particular joint. Either implicitly (15,16) or explicitly (25,19), it is assumed that a surface EMG signal can be modelled as band limited white noise modulated by the level of muscle contraction. According to this assumption, using the EMG signal as an indirect measure of muscle force requires processing the raw EMG signal to extract the level of contraction from the band limited white noise.

Under the limited condition of isometric contraction (i.e., muscle contraction without movement), there are disputes among investigators concerning the dependency of muscle force on processed EMG. Some investigators have suggested a linear relationship (19,25,31), while others have reported relations of higher order (24, 34, 36,37). Moreover, in some previous work, we have shown that most of the variability in reported results comes from the variability in isometric muscle contraction under identical conditions rather than from the type of signal processing used in analysis (30). The question of the proper relationship between EMG and force becomes even more complicated when non-isometric contractions are allowed.

C. NEURAL NETWORK MODELLING BACKGROUND

Neural network models derive their principles from neural systems and seek to attain similar capabilities in artificial devices such as VLSI collective decision circuits. These models have parallel inputs, outputs, and internal computations. The models are composed of computational elements or nodes that are connected by weights and are adapted or trained during use to improve performance. An underlying theme of the neural network concept is that the functional units that govern behavioral adaptation are distributed patterns across a network of cells. In distributed models, the strength of patterns of activity over many units determines the degree of participation of these entities in functional events.

Neural net models are commonly used as pattern classifiers for extracting and classifying features. These models can generate spontaneously useful global computational functions such as classification, optimization, and control. The ability of these models to capture non-linear mappings has been documented (28). The computational properties of NN have been successfully applied to applications such as sensory/ motor control (7,20,21),robot control (4,10,12,26), associative retrieval of information (3,17,22,); process control (12,35),feature detection (6), combinatorial optimization (18); and adaptive pattern recognition (5,12,14).

The promising features of a neural based approach to adaptive control of myoelectric prosthesis, orthotic devices or robotic manipulators include 1. control laws that are not explicitly stated since learning occurs by showing examples; 2. fault tolerance provided by massive parallelism; 3. robustness to unmodelled parameters due to the network's generalization properties; and 4. abundance of local minima in the networks state space used for content addressability and retrieval of information under noise and uncertainty (11,12,13).

In this study, we address the feasibility of designing a "neural network" which will obviate the need to specify the EMG-force relationship apriori. If the neural network succeeds in classifying the force output from a general set of EMG input signals, then we will have a solid basis for our expectation that a complete multi-joint limb can be controlled using a neural network approach. The outcome will be a more anthropomorphic limb which will benefit the fields of prosthetics, electrical stimulation, and robotics.

D. THE BACKPROPAGATION NEURAL NETWORK MODEL

The neural network backpropagation model (NN model) was used in this analysis. The backpropagation model simulated a multilayer perceptron for solution of the muscle EMG-joint torque mapping under isometric supine conditions. Measurements from 6 muscle sites are entered into the model as the input, while the torque is entered into the model as the ideal output, to which the model output is compared (Figure 1). Data from an architecture composed of 6 inputs, two hidden layers, with six nodes per hidden layer , and one output was implemented. Symmetric sigmoid nonlinearities were used in this model. Continuous analog inputs and outputs were assumed. The learning rate and momentum, two parameters of the model, were initially 0.4 and 0.8, respectively. The desired output torque ranged from -1200 (sub-maximal plantarflexion) through +1000 (maximal dorsiflexion) N-m. The calculated and desired outputs were normalized between -0.9 and +0.9. The inputs were not scaled; however, these values all ranged from 0.0 through +8.0 volts (rectified, filtered EMG signals).

During training, each presentation to the NN model was composed of 6 input values (muscle signals) and 1 output value (joint torque) at 0 degree. One sweep of the data of the network represented 4000 presentations (training signal). The 4000 presentations were repeatedly presented to the NN model until the weights stabilized and the error (i.e., measured joint torque output-calculated (model) joint torque output) reached a minimum. Within the training signal there are represented all levels of torque. The data presented during training is composed of a data file that is pre-sorted by joint torque levels. The NN is trained by being shown all input combinations under isometric, supine conditions. The relation between the EMG signals and torque is embedded within the network; i.e., the hidden units, and connections (or weights). If the architecture is large enough to store the EMG -joint torque relations and if training has occurred for an adequate amount of time , then it can be expected that test signals composed of muscle EMG that are presented to the network will predict the corresponding joint torque. Successful prediction of joint torque during operation from any set of EMG inputs under isometric supine conditions is expected. Therefore, it is predicted that a successful training method can be implemented in training the EMG-joint torque mapping with the use of this NN approach. During the prediction (or retrieval) stage, signals are presented to the trained network. They are presented to the NN only once for prediction, since no type of training occurs during this stage of the analysis.

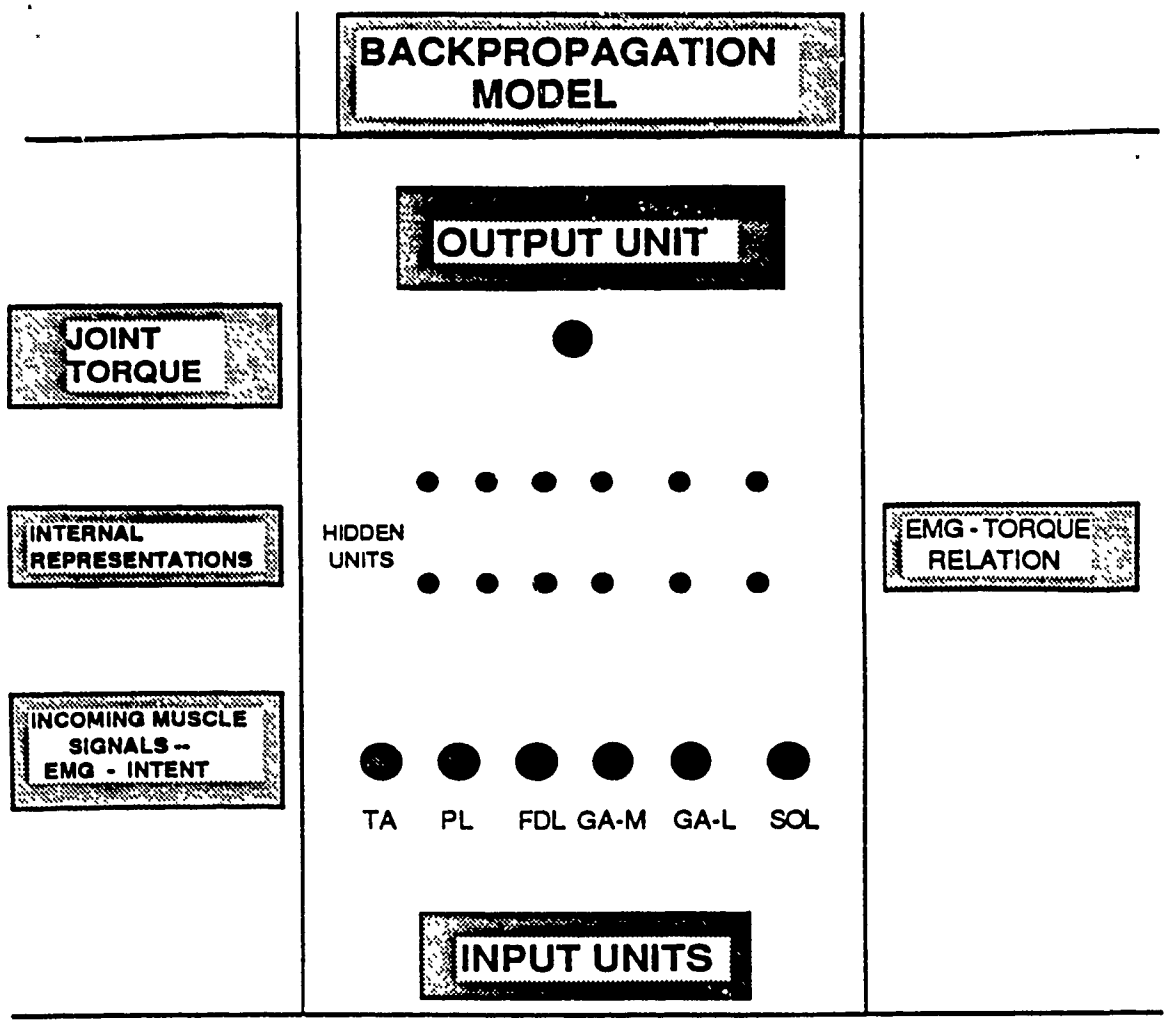


FIGURE 1. NEURAL NETWORK ARCHITECTURE 6-6-6-1

E. EXPERIMENTAL METHOD

The experiment was performed in the FNS Laboratory of the Philadelphia Unit of Shriners Hospitals. A KinCom Robotic Dynamometer, 6 surface Electromyography (EMG) electrodes and EMG amplifier system, MicroVax II minicomputer and MicroVax laboratory system for data acquisition were utilized. Data analysis was performed with the use of a PDP 11-73 minicomputer.

The KinCom II Robotic Dynamometer (Chattecx Corp) is a microcomputer-based system which records the joint angular position and torque developed about the joint upon muscular contraction. With the use of the KinCom computer monitor, subjects visually monitored to the requested torque level. The torque output and muscle signals were collected on the MicroVax II computer for each torque level. The data presented is with the ankle joint set to 0 degree.

Six sites of muscle activity, monitoring the agonist/antagonist muscle activity about the ankle joint, were recorded on the subject's right leg using surface EMG electrodes. The muscle sites were tibialis anterior, peroneous longus, flexor digitorum longus, gastrocnemius-medial, gastrocnemius-lateral, and soleus. The EMG electrodes (Motion Control) contain a preamplifier (gain-200). The signals were full wave rectified and filtered appropriately (Band-pass 100-1000 Hz). Sampling rate was 100 Hz. The signals were amplified to allow minimum and maximum range (across all conditions) to fall within the computer limits of ± 10 V.

A MicroVax II minicomputer was used for data acquisition. Sampling rate for all channels (joint angle, joint torque, . muscle sites) was 100 Hz. Data were transferred to a PDP - 11/73 minicomputer for analysis.

E. EXPERIMENTAL PROCEDURE

This experiment was designed to measure the isometric ankle EMG-torque relations at the measured range of torque output (-1200 to 1000 N-m) during isometric, supine conditions. The subject was instructed to lie supine on the clinical bed. The right knee was braced and the foot was securely fit into a boot which was attached to the KinCom II device. The thigh was securely strapped down so that body motion was prevented during the tasks. The rotational axis of the system was aligned to the lateral malleolus of the subject's foot.

Under the control of the KinCom II system, torque levels in both plantarflexion and dorsiflexion at 0,50,100,200,400,600, 800, 1000,1200 N-m were measured (note that the maximum dorsiflexion torque was 1000 N-m and -1200 N-m was sub-maximal plantarflexion torque). The subject was instructed to generate the designated plantarflexion or dorsiflexion torque. Subjects started from a relaxed position before every muscle contraction. The subjects maintained the muscle contraction for 5 seconds and then relaxed again. Adequate rest was provided between each contraction as well as between each data set. The subject was provided visual information of the designated torque output by the KinCom II system monitor.

G. RESULTS

The NN was trained by being shown all possible examples of the EMG-torque relation of the ankle joint under isometric, supine conditions with respect to torque levels (for both dorsiflexion and plantarflexion torques). Trials of dorsiflexion and plantarflexion signals were windowed and 100 samples (representing 1 sec of data) of each EMG signal from each torque level were extracted for the training set. The total number of samples (presentations) in the training set was 4000. The training set is sufficiently sorted with respect to torque level that good training did result.

The results that follow describe a NN implementation with an architecture of 6 inputs (EMG signals), 2 hidden layers (each 6 nodes), and 1 output (joint torque). The number of iterations during training was 120,000. This represents 30 sweeps of the full training set to the network. The learning rate and momentum were 0.4 and 0.8, respectively. Once learning has begun to occur at approximately 16000 presentations (see Figure 2), these parameters were decremented to achieve a more finer learning. Figure 2 depicts the error, i.e., actual joint torque (o) - calculated torque (+), as a function of the number of presentations. Note that by 16000 iterations, learning has clearly begun, as evidenced by the reduction in the error to less than 6 percent.

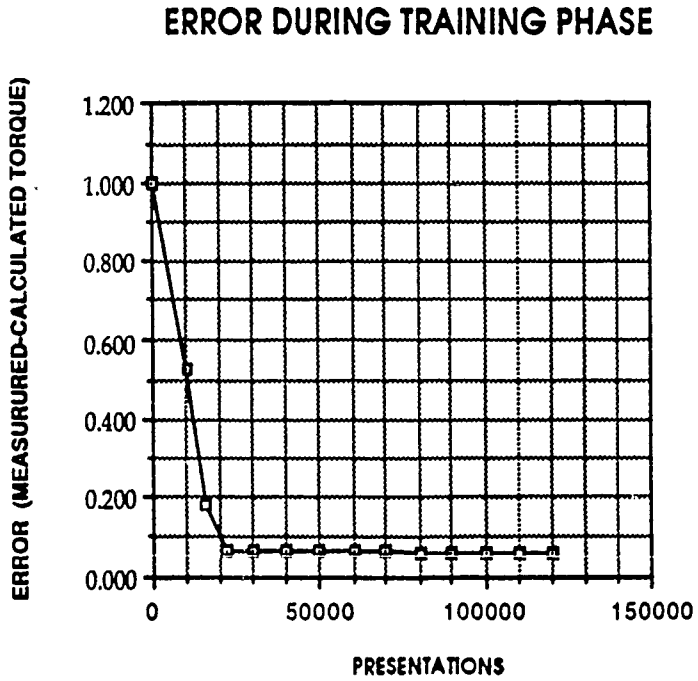


Figure 2 represents the error during the training phase of the backpropagation model.

Figures 3a and 3b, represent the same 100 presentations (of 4000) of the sorted training set at 1000 (during the 1st sweep of the training set) and 117,000 (during the 30th sweep of the training set) iterations. The activity of the 6 muscle sites represent the input, while the torque represents the output. For the output, the actual torque levels are symbolized by o, while the calculated torque levels (during training) are symbolized by +. When the NN has adequately learned the EMG-torque relation (at 117,000), the actual and calculated (via the NN) torque levels are similar. Comparing Figures 3a and 3b, note that at 1000 presentations, the actual and calculated torque levels are quite dissimilar - learning had not yet begun to occur.

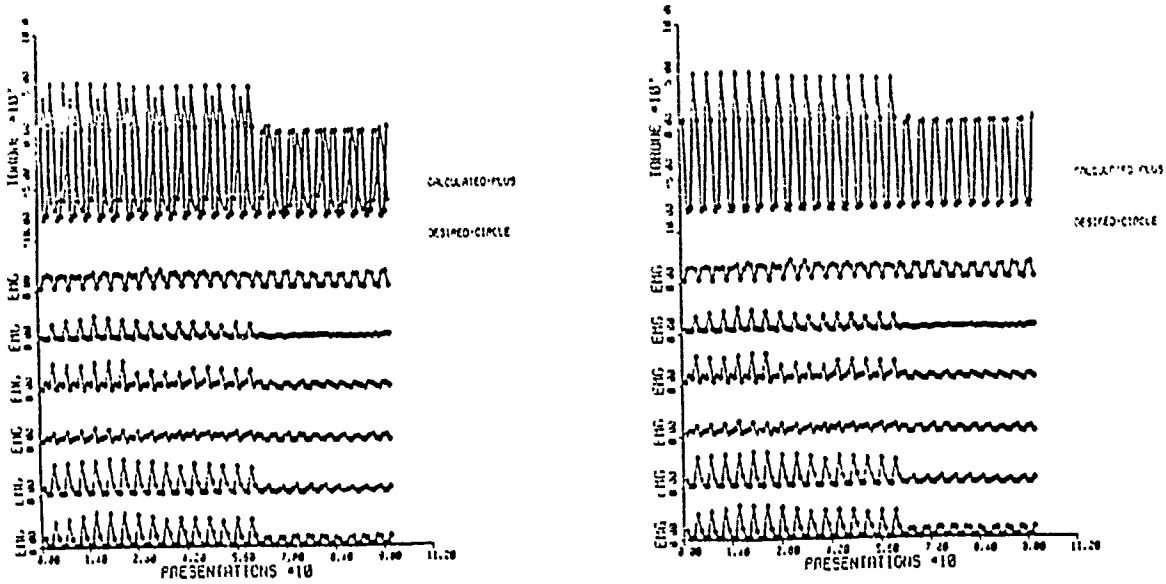


FIGURE 3. Figures 3a and 3b represent 100 presentations of the sorted training data at 1000 (during the 1st sweep of the training set) and at 117,000 (during the 30th sweep of the training set) presentations.

Figures 4a and 4b, represent some typical presentations during operation to the previously trained network. Essentially, the muscle signals are input feedforward to the network. Accuracy is determined by comparing the calculated torque (+) (via the NN) which may be compared with the actual torque (o). Figure 4a represents random input during operation, while Figure 4b represents a signal input , composed of three torque levels.

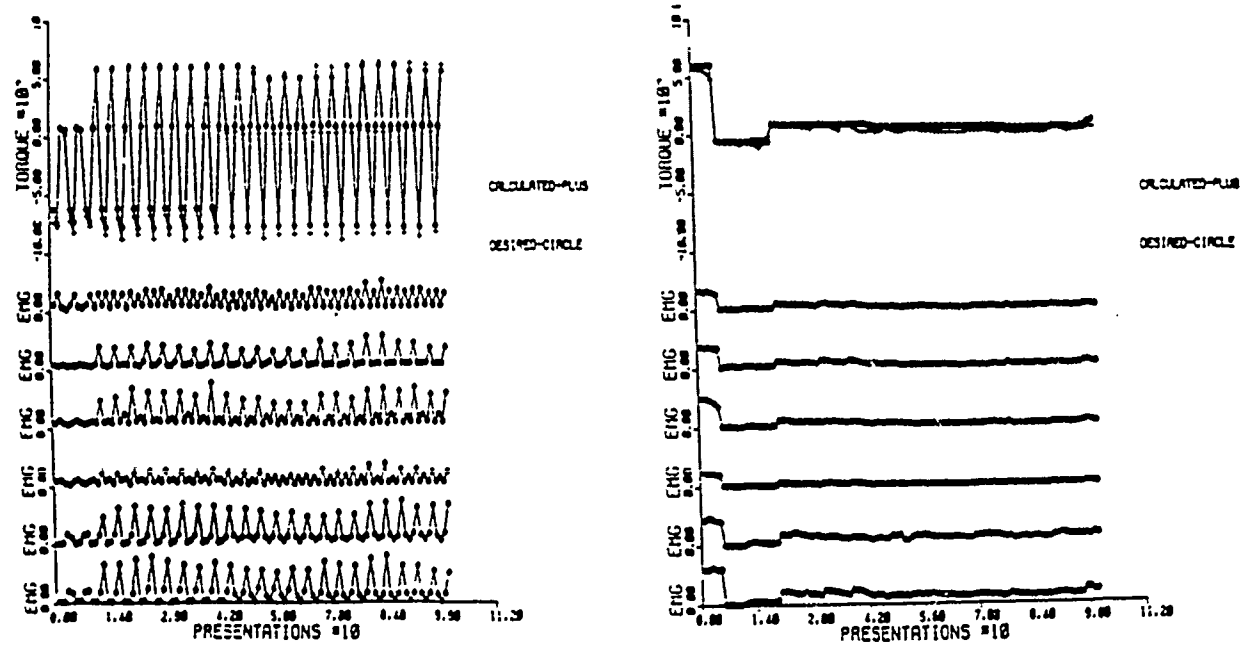


FIGURE 4. During operation, data are presented feedforward to the trained network. Figure 4a represents a random input of various torque levels. Figure 4b represents a signal input, composed of three torque levels.

H. CONCLUSION

The model requirements based on the current approaches to multi-joint (as well as single joint) motion are complex, computationally overtaxing, require apriori constraints and simplifications and are not predictive in all cases. It appears that through evolution and in development that the CNS has learned solutions (whether via inverse kinematics, inverse dynamics, trajectory planning , or task dependent strategies) for the control of multi-joint limb and trunk motion to achieve goal-directed movements in an organized, efficient and adaptive manner. Economy of control would suggest that stereotypic movement patterns should be available from a library of goal-directed solutions to motor problems to accomplish a specific task. These learned solutions may decrease the computations required by the CNS for optimal achievement of the intended movement goal.

The application of the perceptron as an adaptive system for robotic manipulator control was first suggested by Albus (1) in the CMAC (Cerebellar Model Articulation Controller), a model based on neurophysiological theory of cerebellum function. This theory suggests that an input to the cerebellum will compute an address, in which the contents of the address are the appropriate muscle control signals required to carry out the intended movement. In the CMAC model, control functions for many degrees of freedom of a manipulator operating simultaneously are computed by referring to a table rather than by mathematical solutions of simultaneous equations. In perceptron based controller models (such as CMAC and backpropagation), the memory management techniques take advantage of the continuous nature of a control function by allowing similar inputs only to generalize to produce similar outputs and dissimilar inputs to produce dissimilar (independent) outputs. For this application, the neural network backpropagation model may be described as a multilayer perceptron feedforward type of controller. The incoming muscle signals may be considered the "intent" of the system, while joint torque is the "controlled" variable. Embedded within the trained NN are the mapping of the EMG-torque relationship over the full range of torque levels. Essentially this mapping may be considered the library of learned solutions of the EMG-torque relations of the ankle joint under isometric, supine conditions (at 0 deg) (Figure 5). As the incoming muscle signals are applied to the trained network, the library of solutions (i.e., via the look-up table of weights) is addressed for the corresponding output variable (torque).

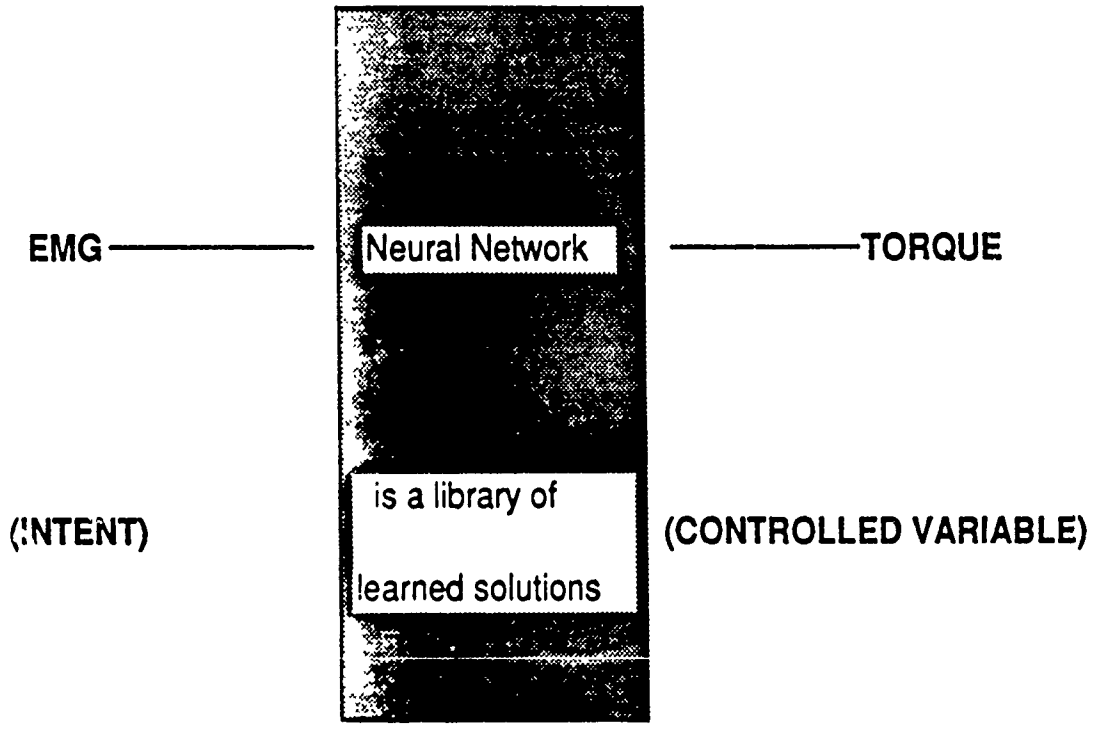


Figure 5. Neural network modelling for EMG to Torque relation.

Effective understanding and modelling of biological motor control will aid in achieving similar characteristics for real-time control of complex systems. The application of neural learning algorithms in the controller for intent recognition systems such as robotic manipulators, myoelectric prostheses and functional electrical stimulators may be appropriate. In fact, for applications such as these, it is desirable to have an adaptive controller that learns to control the system while in operation. This is achieved through heirarchy of control and through variations of the learning procedure of the neural net backpropagation model which desires state information rather than error gradients (26). Based on the results obtained above, specific recommendations will be made concerning the design of adaptive controllers for multijoint prostheses, orthoses, and anthropomorphic automata.

I. REFERENCES

1. J.S.Albus. "A new approach to manipulator control: The cerebellar model articulation controller - CMAC". Trans. of the ASME., September 1975, pp 220-227.

2. C. Almstroem. "An electronic control system for a prosthetic hand with six degrees of freedom." Tech. Rep. 1:77, Research Lab of Med Electronics, Chalmers University of Tech., Goteborg, Sweden, 1977.

3. J.A.Anderson. "Cognitive and psychological computation with neural models." IEEE, SMC. Vol. 13, No 5, pp 799-814,1983.

4. R.K. Elsley. " A learning architecture for control based on backpropagation neural networks." Proc. 2nd Int'l Conf on Neural Networks, Vol II, pp 587-594, 1988.

5. K. Fukushima. "Neocognitron: A hierarchical neural network capable of visual pattern recognition." Neural Networks, Vol. 1, pp 119-130,1988.

6. S. Grossberg. "Adaptive pattern classification and universal recoding I: Parallel development and coding of neural feature detectors." Biol. Cyber. 23, pp 121-134, 1976.

7. S. Grossberg and M. Kuperstein. "Neural dynamics and adaptive sensory-motor control", North-Holland, Amsterdam, 1986.

8. D. Graupe, W. Cline. "Functional separation of EMG signals via ARMA identification methods of prosthetic control purposes." IEEE Trans. systems. Man Cyber., SMC-5:252-259,March,1975.

9. D. Graupe, K.Kohn, and S. Basseas. "Control of electrically-stimulated walking of paraplegics via above- and below- lesion EMG signature identification." IEEE Trans on Automatic Control, Vol 34, No. 2, February 1989, pp 130-138.

10. A. Guez, M.Kam, J. Eilbert. "Neuromorphic architecture for adaptive robot control." Proc 1st Int'l Conf on Neural Networks, Vol 1, San Diego, June, 1987.

11. A. Guez, J. Protopoescv, J. Barhen. "On the stability, storage capactity and design of nonlinear continuous neural networks." IEEE Trans on Sys Man Cyber, 1988.

12. A. Guez and J. Selinsky. "A neuromorphic controller with a human teacher." Proc. 2nd Int'l Conf on Neural Networks, San Diego, June, Vol. II, pp 595-602, 1988.

13. A. Guez, J. Eilbert, and M. Kam. "Neuromorphic architectures for fast adaptive robot control." IEEE pp 145-149, 1988.

14. O.E. Hinton and T. Sejnowski. "Optimal perceptual inference." Proc. IEEE Conf of Computer Vision and Pattern recognition, pp 443-453,1986.

15.N.Hogan and R. Mann. "Myoelectric signal processing: optimal control applied to electromyography. Part I: Derivation of the optimal myoprocessor." IEEE Trans of Biomed Eng, BME-27, No. 7, pp 390-395, 1980.

16.N.Hogan and R. Mann. "Myoelectric signal processing: optimal estimation applied to electromyogram - Part II: Experimental demonstration of optimal myoprocessor performance." IEEE Trans of Biomed Eng, BME-27, No 7, pp 396-410, May 1980.

17.J. J. Hopfield. "Neural networks and physical systems with emergent collective comutational abilities." Proceed. Nat. Acad. of Science (USA), Vol 79, pp 2554-2558, 1982.

18.J.J. Hopfield and D.W. Tank. "Neural computation of decision optimization problems." Biol. Cybernetics, Vol. 52, pp 1-12, 1985.

19.V.F. Inman, H.J. Ralston, C.M. Saunders, B. Freinstein, E. Wright. "Relation of human electromyogram to muscle tension." Electro. Clin. Neurophys. 4:107-294, 1952.

20.M. Kawato, K. Furukawa, and R. Suzuki. "A hiedrarchical neural network model for control and learning of voluntary movement." Biol. Cybern. 57, pp 169-185, 1987.

21 M. Kawato, M.Isobe, Y.Maeda, and R. Suzuki."Coordinate transformation and learning control for visually-guided voluntary movement with iteration: A Newton-like method in a function space." Biol. Cybern. 59, pp 251-265, 1988.

22.T. Kohonen. "Self-organized formation of topologically correct feature maps." Biol. Cybern. Vol 43, pp 59-69, 1982.

23.P.D. Lawrence, W.C.Lin. "Statistical decision making in real-time control of an arm aid for the disabled." IEEE Trans. Systems, Man Cyber. SMC - 2:35-42, 1972.

24.L. Lindstrom, R. Magnuss, I. Petersen. "Muscle load influence on myoelectric signal characteristics." Scand. J. Rehab. Med. Suppl 3: 127-148, 1974.

25.O.C.J.Lippold. "The relation between integrated potentials in human muscle and its isometric tension." J. Physiol. 117:492-499,1952.

26.W.T.Miller,III. "Sensor-based control of robotic manipulators using a general learning algorithm." IEEE J. of Robotics and Automation Vol RA-3, No. 2, pp 157-165, April 1987.

27.D.R.Myers and G.D.Moskowitz. "Myoelectric pattern recognition for use in volitional control of above-knee prosthesis." IEEE Trans Systems Man Cyber SMC -11(4):296-302,1981.

28.D.E.Rumelhart, G.E. Hinton, and R.J.Williams. "Learning internal representations by error propagation." Parallel distributed processing", MIT Press, Chapter 8, pp 318-362, 1986.

29. G.N.Saridis, T.P.Gootee. "EMG pattern analysis and classification for a prosthetic arm." IEEE Trans on Biomed Eng. Vol. BME-29, No. 6, June 1982.

30. S.Siegler, H.J.Hillstrom, W.Freedman,G.Moskowitz. Effect of myoelectric signal processing on the relationship between muscle force and processed EMG." Amer. J. Phys. Med.64(3),pp 130-149,1985.

31. J.A.Stephens,A.Taylor: "The relationship between integrated electrical activity and force in normal and fatiguing human voluntary muscle contraction.in New Developments in Electromyography and Clinical Neurophysiology, J.E.Desmedt, Ed., Vol 1:623-627, Basel, Karger, 1973.

32. D.R.Taylor and R.W. Wirta. "Developments of a myoelectrically controlled prosthetic arm." Proc. of Their Intern. Symp on External Control of Human Extremities, Dubrovnik, Yugoslavia,1969.

33. R.J.Triolo, D.H.Nash, and G.D.Moskowitz. "The identification of time series models of lower extremity EMG for the control of prostheses using Box-Jenkins Criteria." IEEE Trans on Biomed.Eng Vol 35, No 8, pp 584-594, August 1988.

34. J. Vredenbregt and G. Rao. "Surface electromyography in relation to force, muscle length and endurance." in New Developments in electromyography and clinical neurophysiology, J.E.Desmedt, E., Vol I:607-622,Basel,Karger,1973.

35. B.Widrow. "The original adaptive broom balancer." IEEE Conf on Circuits and Systems, Philadelphia, PA 1987.

36. D.A.Winter. Biomechanics of human movement, pp 140-141, John Wiley and Sons, NY, 1979.

37. E.N.Zumiga and D.G.Sim. "Non-linear relationship between averaged electromyogram potential and muscle tension in normal subjects." Arch. Phys. Med.Rehab. 50:613-620,1968.

[22]

A Learning/Adaptive Robot Controller

Allon Guez John Selinsky

Drexel University
Department of Electrical and Computer Engineering
32nd and Chestnut St., Philadelphia, PA 19104

ABSTRACT

In this article, we explore the relationship of learning and adaptation in robot control. Learning, in this context, is the process of identifying the robot dynamics and its interaction with the environment for the purpose of improved tracking over an infinite horizon. Whereas, adaptation is the process of adjusting the controller to comply with the regulation and tracking needs of the closed loop system. We thus demonstrate that learning conflicts with adaptation in its tendency to increase the present tracking error, due to the minimization of different criteria (Dual control principle).

Exploratory Schedules (ES) are reference trajectories which are specifically designed to provide efficient closed loop learning. We relate ES design to the issue of input richness (or persistent excitation). Our ES represents a weaker criteria than persistent excitation

A theorem regarding constructive sufficient conditions for asymptotically stable closed loop learning is stated, and examples of learning in 1 and 2 degree of freedom manipulators are given.

I. INTRODUCTION

Adaptive control of robot manipulators has been the subject of much research in recent years (see Ref. [1]). Adaptation, in control, is the process of adjusting the controller to comply with the regulation and tracking requirements of the closed loop system. Direct adaptive controllers, e.g. [2] (see also Ref.[3]), use tracking errors of the joint motion to direct the robot model parameter adjustment. The direct adaptive controllers are based on the full dynamic model of the robot. Learning, in this context is the identification of the true values of the manipulator parameters in closed loop operation.

In operation, the controller is given a trajectory by a path planner in order to accomplish some useful task. The controller then adapts the parameters, on line, so as to satisfy the tracking requirements. If the parameters are not known exactly, there will be a transient period of tracking error while adaptation occurs. So that identification of the true parameters is desirable for increased tracking precision.

Research supported in part by AFOSR grant # 890010

Present robot adaptive controllers (see for example [2], [4], [5]) use robot parameter update laws which rely on the assumption that the parameters are constants. Also, to guarantee learning, a trajectory which is "rich" enough to expose all the dynamics of the manipulator is required for convergence of the parameter estimates to their true values. Richness of input refers to trajectories such that the internal signals of the parameter adjustment mechanism is persistently exciting (see [6]). If the trajectory is not persistently exciting, stability is assured but not learning. Thus future tasks are performed with the same transient tracking errors.

We propose (see figure 1.1) that whenever it is possible to modify the input, e.g. prior to putting the manipulator to work doing useful tasks or periodically when parameters change and the path planner is not demanding a new path, that there will be a learning period where the controller learns the true values of the parameters by tracking artificially designed Exploratory Schedules (ES).

ES are trajectories specifically designed for asymptotic learning of the system dynamics. Any trajectory that is persistently exciting could be used as an exploratory schedule. However, the synthesis of persistently exciting trajectories is generally not straightforward. In this article, we show how to construct Exploratory Schedules which guarantee closed loop learning. Our ES are not necessarily persistently exciting.

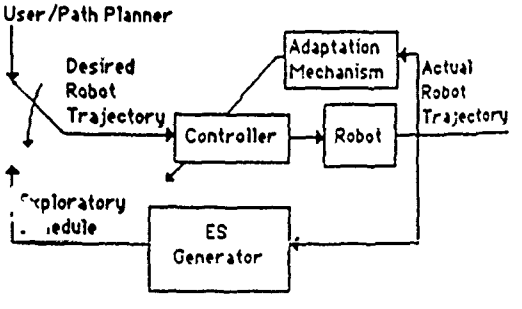


figure 1.1: Block diagram of the proposed system.

The rest of the article is formatted as follows. Section 2 presents the general structure and properties of a rigid robot. Section 3 defines the adaptive control structure used, and presents sufficient conditions for global asymptotic learning of parameters which is not explicitly based on persistent excitation. Section 4 describes simulation results of learning with 1 and 2 degree of freedom (DOF) manipulators. Section 5 concludes the article.

2. Rigid Robot Dynamics

The closed form dynamics obtained by the Lagrange-Euler formulation has the general matrix form of [7]

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \qquad (2.1)$$

where

$D(q)$ is an n x n matrix of inertial terms,
$C(q,\dot{q})$ is an n x n matrix of coriollis and centrifugal terms,
$G(q)$ is an n x 1 vector of gravitational terms,
q is an n x 1 vector of joint coordinates,
τ is an n x 1 vector of forces/torques,
n is the number of degrees of freedom.

The formulation results in n second-order, coupled differential equations.

Reparameterization Property of Robot Dynamics

It has been pointed out by a number of authors (e.g. [5]), that there are properties of the dynamics that can be exploited for robot control. The reparameterization property is repeated here, as it will be used in other sections. The property states that

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\dot{q},\ddot{q})\Phi$$

where $Y(q,\dot{q},\dot{q},\ddot{q})$ is an n x r matrix of known functions and Φ is an r x 1 vector of constant parameters. This property implies that the robot dynamics may be viewed as a linear operator from a suitably chosen set of parameters to the joint torques.

3. Closed Loop Learning Via Exploratory Schedules

In [2], an adaptive controller which guarantees global asymptotic tracking was formulated. The controller is based on knowledge of the robot dynamics structure and the use of sliding surface control. A major advantage of this controller is that there is no need to measure the joint acceleration. The convergence of the parameter estimates to their true values, however is not guaranteed. In this work, the same control structure is used. However, we synthesize, whenever possible, the reference input to guarantee both tracking and learning.

Controller Structure

For the system of equation (2.1), define the virtual reference trajectory

$$\begin{aligned} q_r &= q_d - A\int_0^t e\,dt \\ \dot{q}_r &= \dot{q}_d - A\,e \\ \ddot{q}_r &= \ddot{q}_d - A\,\dot{e} \end{aligned} \qquad (3.1)$$

where q_d is an n x 1 vector of desired joint coordinates, $e = q - q_d$, and A is a positive definite matrix with constant coefficients. Define the sliding surface

$$s = \dot{e}_r = \dot{q} - \dot{q}_r = \dot{e} + Ae \qquad (3.2)$$

Let the control input be

$$\begin{aligned} \tau &= \hat{D}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{G}(q) - K_d s \\ &= Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\Phi} - K_d s \end{aligned} \qquad (3.3)$$

where $\hat{D}(q)$, $\hat{C}(q,\dot{q})$, $\hat{G}(q)$ and $\hat{\Phi}$ are estimates of $D(q)$, $D(q,\dot{q})$, $D(q)$ and Φ respectively, and K_d is an n x n positive definite matrix. The parameter adaptation law is

$$\dot{\hat{\Phi}} = -\dot{\tilde{\Phi}} = -K_a^{-1}\,Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)s \qquad (3.4)$$

where the estimation error is defined as $\tilde{(.)} = \hat{(.)} - (.)$, and K_a is an r x r positive definite matrix with constant elements.

Theorem [8]
If $r \le n$, and if

$$\text{Rank}\left\{ \lim_{t\to\infty} Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d) \right\} = r$$

then the controller (eq. (3.3), (3.4)) guarantees global asymptotic tracking and identification.

Selection of Exploratory Schedules

An implication of the theorem is that if $r > n$, a sufficient condition for $k \le n$ parameters to be identified is that they are are not in the null space of

$$\lim_{t\to\infty} Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d)$$

Which implies that a desired trajectory may be chosen such that columns corresponding to the $k \le n$ parameters are linearly independent, which will guarantee that the parameters are identified. So that different time sections of the Exploratory Schedule (which specifies $q_d,\dot{q}_d,\ddot{q}_d$) may be designed to learn different components of the vector $\hat{\Phi}$ it its dimension exceeds n.

4. SIMULATION RESULTS

Manipulator with 1 DOF

The first simulation was done for a single link manipulator as shown in figure 4.1

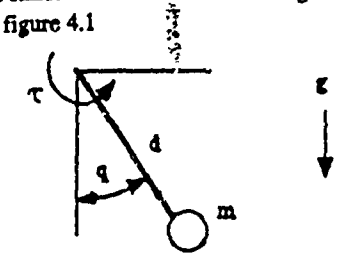


figure 4.1: Simulated 1 DOF manipulator.

The dynamics of the manipulator are

$$\tau = m\, d^2\, \ddot{q} + m\, g\, d \sin(q), \qquad (4.1)$$

where, d = 1, g = 10, and m = 2 . The matrix of known functions is

$$Y(q, \ddot{q}) = [\, d^2\, \ddot{q} + g\, d \sin(q)], \qquad (4.2)$$

with m as the single parameter to be estimated. The control law as specified by eq (3.3) is

$$\tau = [\, d^2\, \ddot{q}_r + g\, d \sin(q)]\hat{m} - k_d(\dot{q} - \dot{q}_r), \qquad (4.3)$$

where $\dot{q}_r = \dot{q}_d - (q - q_d)$, kd = 1, and the parameter adaptation law as defined by (3.4) was used to estimate m, where the estimate is denoted as $\hat{m}$.

In experiment 1a,

$$\dot{q}_d = 0.0,\ q_d = 0.0 \Rightarrow \text{Rank}\left(\begin{matrix} Y(q_d, \ddot{q}_d) \\ \text{Lim } t \to \infty \end{matrix}\right) = 0$$

which is less than r = n = 1. Notice in figure 4.2, that even though the tracking error does converge to zero, the estimate of m does not converge to its true value of 2.0 .

In experiment 1b,

$$\dot{q}_d = 0.0,\ q_d = 1.0 \Rightarrow \text{Rank}\left(\begin{matrix} Y(q_d, \ddot{q}_d) \\ \text{Lim } t \to \infty \end{matrix}\right) = 1$$

which equals n. In the second experiment, shown in figure 4.3, the estimate of m does converge to its true value. Thus closed loop learning is obtained.

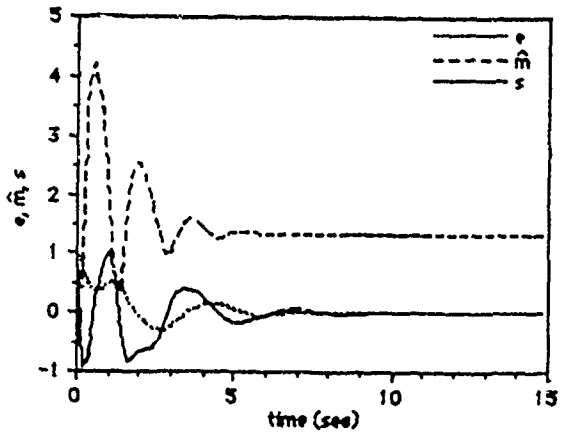


figure 4.2: Experiment 1a.

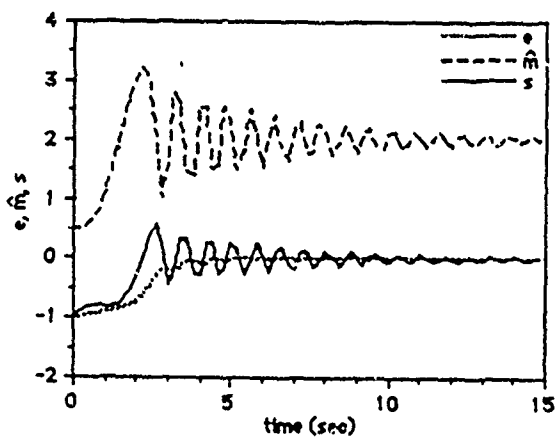


figure 4.3: Experiment 1b.

Manipulator with 2 DOF

The second simulation was done for a 2 DOF manipulator as shown in figure 4.4

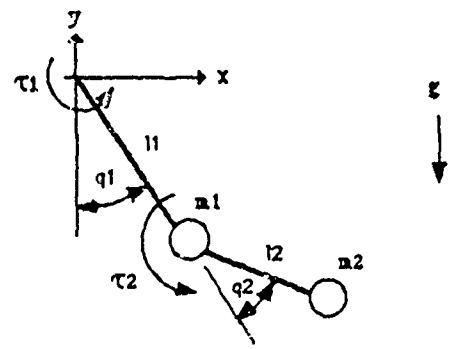


figure 4.4: Simulated 2 DOF manipulator.

The dynamics of the 2 DOF manipulator are

$$\tau_1 = D_{11}\ddot{q}_1 + D_{12}\ddot{q}_2 + 2\,C\,\dot{q}_1\dot{q}_2 + C\,\dot{q}_2^2 + G_1 \qquad (4.4)$$
$$\tau_2 = D_{12}\ddot{q}_1 + D_{22}\ddot{q}_2 - C\,\dot{q}_1^2 + G_2$$

where

$D_{11} = m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 \cos(q_2)$,
$D_{12} = m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)$,
$D_{22} = m_2 l_2^2$,
$C = m_2 l_1 l_2 \sin(q_2)$
$G_1 = g(m_1 + m_2) l_1 \sin(q_1) + g m_2 l_2 \sin(q_1 + q_2)$
$G_2 = g m_2 l_2 \sin(q_1 + q_2)$,

with $m_1 = m_2 = 10.0$, $l_1 = l_2 = 1.0$, $g = 9.81$.

The parameters and their true values are

$$\begin{aligned} \emptyset_1 &= (m_1+m_2) l_1^2 = 20.0 & \emptyset_4 &= g(m_1 + m_2) l_1 = 196.2 \\ \emptyset_2 &= m_2 l_2^2 = 10.0 & \emptyset_5 &= g m_2 l_2 = 98.1 \\ \emptyset_3 &= m_2 l_1 l_2 = 10.0. \end{aligned} \qquad (4.5)$$

Which leads to the reparameterization matrix

$$Y(q, \dot{q}, \dot{q}, \ddot{q}) = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} & Y_{15} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} & Y_{25} \end{bmatrix}, \qquad (4.6)$$

where

$Y_{11} = \ddot{q}_1$,
$Y_{21} = 0$,
$Y_{12} = Y_{22} = \ddot{q}_1 + \ddot{q}_2$,
$Y_{13} = \cos(q_2)(2\ddot{q}_1+\ddot{q}_2) - \sin(q_2)(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)$,
$Y_{23} = \cos(q_2)\ddot{q}_1 + \sin(q_2)\dot{q}_1^2$,
$Y_{14} = \sin(q_1)$,
$Y_{14} = 0$,
$Y_{15} = Y_{25} = \sin(q_1+q_2)$.

The control law as specified by eq. (3.3) is

$$\tau = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{\emptyset} - K_d s, \qquad (4.7)$$

where

$s = (\dot{e}_1+e_1, \dot{e}_2+e_2)^T$
$K_d = \mathrm{diag}(1000, 500)$.

The parameter adaptation law as defined by (3.4) was used to update the parameter estimates $\hat{\emptyset}$, with

$K_a^{-1} = \mathrm{diag}(1000, 1000)$.

The dimension of $Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$ in the 2 DOF case is 2 x 5 so that r >n, and at most 2 parameters can be guaranteed to be learned simultaneously.

The initial conditions for both experiments (2a and 2b) are at $q_1(0) = q_2(0) = \dot{q}_1(0) = \dot{q}_2(0) = 0$. The desired velocities for both experiments is $\dot{q}_{d1}(0) = \dot{q}_{d2}(0) = 0$.

In experiment 2a,

$$q_{d1} = 0.5, q_{d2} = -0.5 \Rightarrow \mathrm{Rank}\begin{pmatrix} Y(q_d \ddot{q}_d) \\ \mathrm{Lim}\ t \to \infty \end{pmatrix} = 1,$$

which is less than minimum(r, n) = 2. Notice in figure 4.4, that even though the tracking error (which is not shown) does converge to zero, the estimates of $\emptyset_4$ and $\emptyset_5$ do not converge to the true values.

In experiment 2b,

$$q_{d1} = 0.5, q_{d2} = 0.5 \Rightarrow \mathrm{Rank}\begin{pmatrix} Y(q_d \ddot{q}_d) \\ \mathrm{Lim}\ t \to \infty \end{pmatrix} = 2,$$

which equals n. The linearly independent columns for experiment 2b correspond to parameters $\emptyset_4$ and $\emptyset_5$. As shown in figure 4.5, the estimates of $\emptyset_4$ and $\emptyset_5$ do converge to the true values.

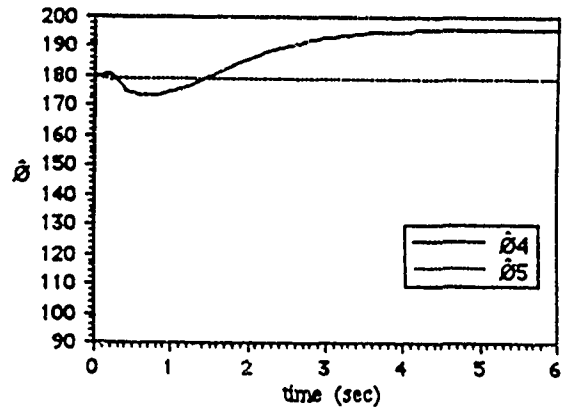


figure 4.4: Experiment 2a.

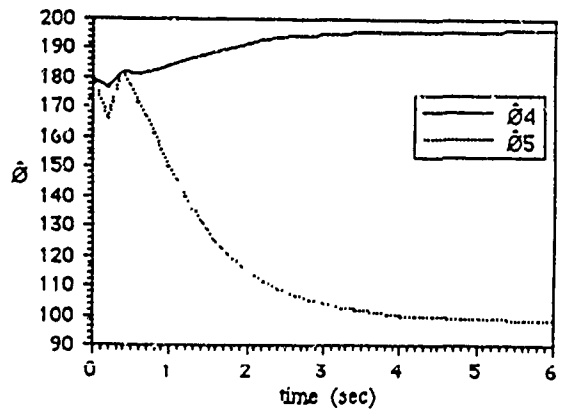


figure 4.5: Experiment 2b.

Exploratory schedule for the 2 DOF Manipulator

Selection of the exploratory schedule for the 2 DOF manipulator was accomplished by noting that columns 4 and 5 of (4.7) are functions only of position, therefore selecting $\ddot{q}_{d1}=\ddot{q}_{d2}=\dot{q}_{d1}=\dot{q}_{d2}=0.0$, and q_{d1}, q_{d2} such that columns 4 and 5 are nonzero as e_1, e_2, $\dot{e}_1$, $\dot{e}_2$ -> 0 will ensure that parameters $\emptyset_4$ and $\emptyset_5$ are identified. Next, selecting $\ddot{q}_{d1}=\ddot{q}_{d2}=0.0$, and q_{d1}, q_{d2}, $\dot{q}_{d1}$, $\dot{q}_{d2}$ such that column 3 is nonzero in the limit ensures identification of parameter $\emptyset_3$. Then selecting $\ddot{q}_{d1}$, $\ddot{q}_{d2}$ such that columns 1 and 2 are nonzero ensures the identification of parameters $\emptyset_4$ and $\emptyset_5$. The actual exploratory schedule is as shown in figures 4.6 to 4.8 .

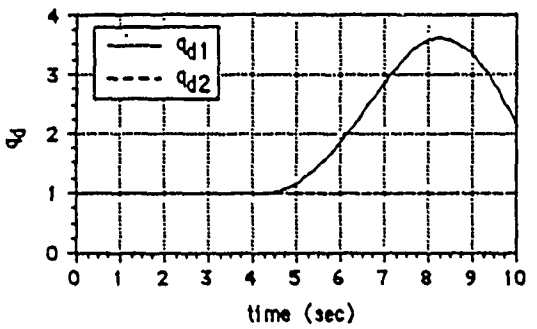


figure 4.6: Desired joint positions during ES.

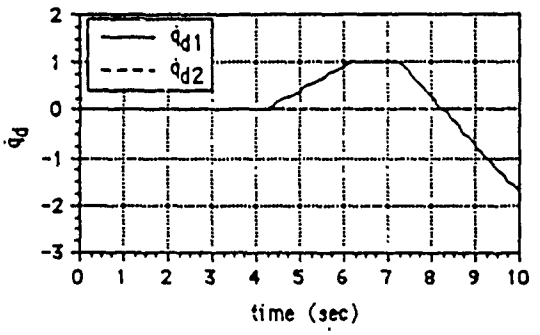


figure 4.7: Desired joint velocities during ES.

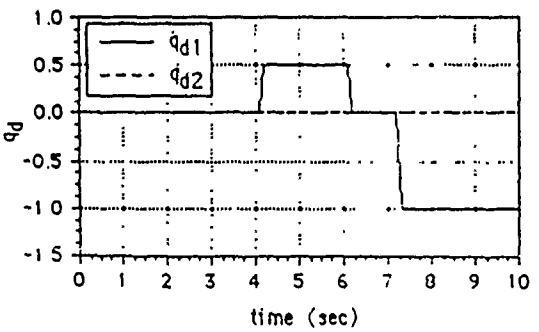


figure 4.8: Desired joint accelerations during ES.

Theoretically, a parameter is guaranteed to be identified when

$e_1=e_2=\dot{e}_1=\dot{e}_2= 0.0.$

However, in practice, due to noise and disturbances, exact tracking may not occur and small tracking errors may lead to small errors in the parameter identification. In this simulation, a parameter is said to be identified when

$\text{Maximum}(|e_1|, |e_2|, |\dot{e}_1|, |\dot{e}_2|) \le e_{tol}$,

where e_{tol} = 0.01, and a further stipulation that attempted identification of each parameter is to last at least 3 seconds.

The time period corresponding to identification of $\emptyset_4$ and $\emptyset_5$ is $0 \le t < 4.2$. As can be seen in figures 4.9 and 4.11, the parameter estimation error for $\emptyset_4$ and $\emptyset_5$ approaches zero as all joint errors fall below 0.01 at t=4.2.

The time period corresponding to identification of $\emptyset_3$ is $4.2 \le t < 7.3$. The parameter estimation error for $\emptyset_3$ approaches zero as all joint errors approach zero at t=7.3 .

Parameters $\emptyset_1$ and $\emptyset_2$ are identified during the time period $7.3 < t \le 10.3$.

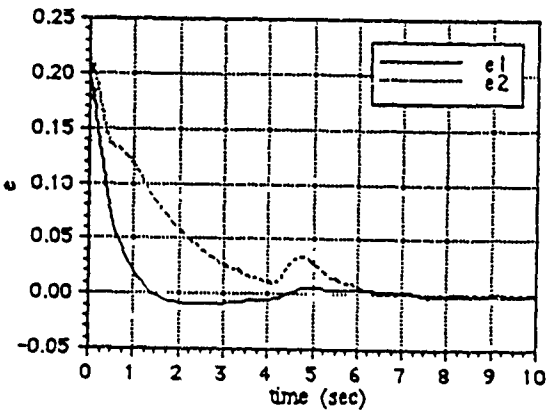


figure 4.9: Joint position error during ES.

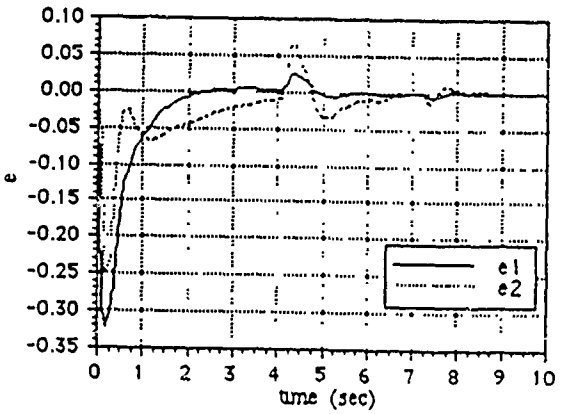


figure 4.10: Joint velocity error during ES.

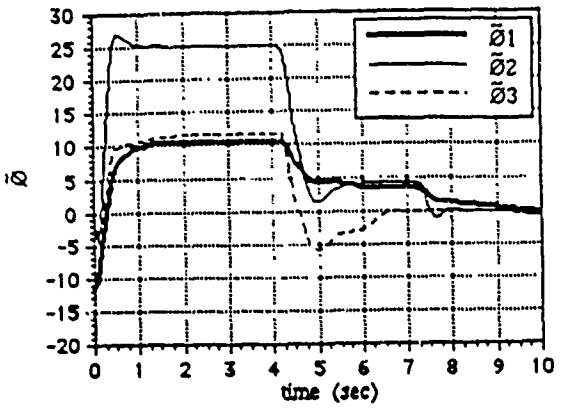


figure 4.11: Estimation error for parameters 1,2 and 3.

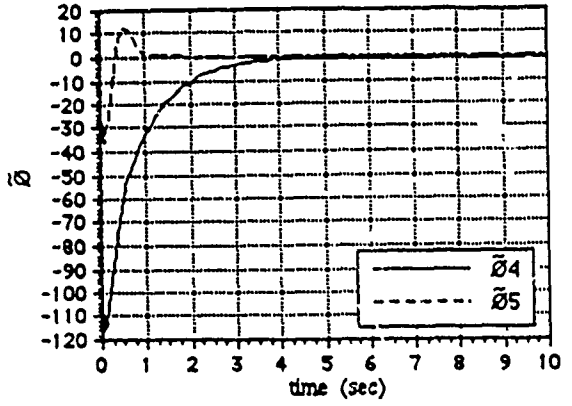


figure 4.12: Estimation error for parameters 4 and 5.

5. Conclusion

In this article, we explored the relationship of learning and adaptation in robot control. We have shown how the design of ES is an essential aspect of learning. We have related ES design to the issue of persistent excitation. The ES represents a weaker criteria than persistent excitation

A theorem regarding constructive sufficient conditions for asymptotically stable closed loop learning was stated, and examples of learning in 1 and 2 DOF manipulators were given.

The simulation results of the ES for a 2 DOF manipulator (section 4.2), showed how in practice, due to noise and disturbances, exact tracking may not occur and small tracking errors may lead to small errors in the parameter identification.

We have demonstrated a method for the selection of exploratory schedules for rigid robot manipulators employing inverse dynamics controllers with direct parameter adaptation. A natural extension to this work is to generalize the concept to a broader class of systems.

We have specified the ES as a desired trajectory that is to be followed to do learning while the manipulator is not doing other useful tasks. However, a more appealing idea would be to integrate the exploratory schedule into any trajectory that is specified by the path planner. The ES would take the form of an exploratory (or probing) signal into the desired trajectory to cause asymptotic learning. The injection of exploratory schedules in this manner would be similar to the concept of dual control (see Ref.[9]) in that the probing signal ES is combined with the so called cautious signal.

REFERENCES

[1] Hsia, T.C., "Adaptive control of robot manipulators-a review," *IEEE Int. Conf. Robotics and Automation*, San Francisco, 1986.

[2] Slotine, J.J.E., and Li, W., "Adaptive manipulator control: a case study," *IEEE Trans. Automatic Control*, Vo. 33, no.11, p. 995, 1988.

[3] Li, W., and Slotine, J.J.E., "Indirect adaptive robot control," *IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA, 1988.

[4] Craig, J.J., *Adaptive Control of Mechanical Manipulators*. Addison-Wesley,1988.

[5] Ortega, R., and Spong, M. W., "Adaptive motion control of rigid robots: a tutorial," *Proceedings of the 27th Conference on Decision and Control, Austin, Texas*, 1988

[6] Narendra, K.S., and Annaswamy, A.M., *Stable Adaptive Systems*. New Jersey: Prentice-Hall, 1989.

[7] Paul, R.P., *Robot Manipulators: Mathematics, Programming, and Control*. Massachusetts: The MIT Press, 1981.

[8] Guez, A. and Selinsky, J., "Design of Exploratory Schedules in Learning and Adaptive Robot Control," submitted for publication.

[9] Stengel, R.F., Stochastic Optimal Control: Theory and Operation. New York: John Wiley & Sons, 1986.

[23]

UNSUPERVISED PARALLEL DISTRIBUTED COMPUTING ARCHITECTURE FOR ADAPTIVE CONTROL

Izhak Bar-Kana and Allon Guez

ECE Dept., Drexel University, Philadelphia, PA 19104

ABSTRACT

An unsupervised parallel distributed adaptive controller for nonlinear systems is proposed. It is shown to provide bounded tracking and asymptotic regulation following an arbitrary 'teacher'. A two degrees of freedom robotic simulation example is provided.

1. PROBLEM FORMULATION

An unsupervised distributed parallel computing architecture is proposed for the adaptive control of nonlinear dynamic systems of the class

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \quad (1)$$
$$y(t) = C(x)x(t) + D(x)u(t) \quad (2)$$

where $x(t) \in R^n$ is the plant state vector, $y(t) \in R^m$ is the output vector, and $u(t) \in R^m$ is the input command vector, and where $A(x)$, $B(x)$, $C(x)$, and $D(x)$ are uniformly bounded matrices of corresponding dimensions.

The proposed controller is depicted in Figure 1 below: It consists of a teacher model which contains apriori knowledge regarding the desired input/output plant response as well as the repertoir of reference commands that the system may be subjected to.

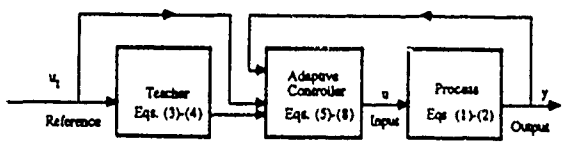


Fig. 1: Proposed Adaptive Controller

The teacher dynamic model is assumed to have the following representation:

$$\dot{x}_t(t) = A_t(x_t)x_t(t) + B_t(x_t)u_t(t) \quad (3)$$
$$y_t(t) = C_t(x_t)x_t(t) + D_t(x_t)u_t(t) \quad (4)$$

where $x_t(t) \in R^{n_t}$ is the state vector, $y_t(t) \in R^m$ is the output vector, and $u_t(t) \in R^m$ is the input command vector, and where $A_t(x_t)$, $B_t(x_t)$, $C_t(x_t)$, and $D_t(x_t)$ are uniformly bounded matrices of corresponding dimensions. It is emphasized that the dimension of the model is unrestricted, except that $\dim(y_t) = \dim(y) = m$.

The parallel distributed adaptive controller has structure similar to the LMS adaptive layer [Widrow, 85], and to many other neurocontroller architectures [Guez, 88]. It receives the input 'features' vector $f(u_t, x_t, y)$ and generates as an output the process contorl vector u, where

$$u(t) = K(t)f(u_t, x_t, y) \quad (5)$$

where K(t) is the adaptive gain matrix of appropriate dimension. Each K_{ij} gain monitors the sensitivity of the i-th control loop, namely u_i, to the j-th feature of the system, namely $f_j(u_t, x_t, y)$. The K_{ij} gain adjusts its value independently of, and simultaneosly with all other gains, according to:

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \quad (6)$$
$$M_{ij}(t) = \alpha_{ij}(y_{t_i} - y_i)f_j \quad (7)$$
$$\frac{d}{dt}N_{ij}(t) = -\beta_{ij}N_{ij}(t) + \gamma_{ij}(y_{t_i} - y_i)f_j \quad (8)$$

where α_{ij}, β_{ij} and γ_{ij} are positive constants. We emphasize that the adaptation law (6)-(8) is performed in parallel and in a **distributed** fashion, that is, the K_{ij} gain only needs data **from the j-th feature and i-th output components.** Thus, very large scale dynamic systems may be considered for this controller. The adaptive gains consist of two terms: a "proportional" term, $M_{ij}(t)$, and an "integral" term, $N_{ij}(t)$. Notice that the role of the teacher (3)-(4) is to demonstrate to the adaptive controller what should be the appropriate and desired response y_t for any specified reference input u_t. Since the teacher's model is incorporated in the controller structure (Figure 1), it yields the so-called 'unsupervised' learning or adaptation.

For the above described control architecture we use recent results by [Bar-Kana, 87, 89b] to show that, under some realistic assumptions, large classes of nonlinear plants of the form (1)-(2) with adaptive controllers of the form (5)-(8) can perform good trajectory tracking and also guarantee robust adaptive stabilization in the presence of *any* bounded input

This paper is based upon research supported in part by Drexel University's Stein Fellowship Foundation and by AFOSR Grant No. 890010.

commands and input or output disturbances. Furthermore, this adaptive controller has ben shown to have good graceful degradation (or fault resistant) properties with control failure or fast changes of plant parameters [Morse and Ossman, 1989].

MAIN RESULTS

We describe below a summary of our main results regarding the performance of the controller in (6), (7), and (8). We will assume that if the plant is stabilizable the resulting stable configuration is "**exponentially stable**" as defined below:

DEFINITION 1 [Hahn, 1967]: Let the general nonlinear system be represented by the *n* th-order vectorial equation $\dot{x}(t) = f(x,t)$ and let x=0 be an equilibrium point. The equilibrium point is called "**exponentially stable**" if all solutions x(t) satisfy the relation $|x(t)| \leq \alpha |x(0)| e^{-\beta t}$ for some scalars $\alpha > 0$ and $\beta > 0$.

THEOREM 1 [Hahn, 1967]: Let the right hand of the equation $\dot{x}(t) = f(x,t)$ have bounded continuous first order partial derivatives. Let the equilibrium be **exponentially stable**. Then there exists a Lyapunov function V(x,t) which satisfies estimates of the form

$$1.1 \qquad \alpha_1 |x(t)|^2 \leq V(x,t) \leq \alpha_2 |x(t)|^2$$

$$1.2 \qquad \dot{V}(x,t) \leq -\alpha_3 |x(t)|^2$$

$$1.3 \qquad \left|\frac{\partial V(x,t)}{\partial x_i}\right| \leq \alpha_4 |x(t)|^2 \ , \ i=1,2,\ldots,n$$

for α_1, α_2, α_3, and α_4 positive constants.

Because we deal with nonlinear systems we cannot expect to be able to find or even show, like in linear time-invariant systems, existence of positive definite quadratic Lyapunov function of the form $V(x) = x^T(t)Px(t)$ where P is constant and positive definite. However, after some experience with specific nonlinear systems like robots, and because we restrict our discussion to nonlinear systems linear in control of the form (1)-(2), we assume that exponential stability of the *autonomous* system (1), with $u(t) \equiv 0$, implies existence of Lyapunov functions V(x) which are not explicit functions of time. Also, since we can always write $V(x) = x^T(t)P(x)x(t)$ and then $\dot{V}(x) = x^T(t)[\dot{P}(x) + P(x)A(x)+A^T(x)P(x)]x(t)$ or $\dot{V}(x) = -x^T(t)Q(x)x(t)$, we will use in the subsequent discution the following assumption:

ASSUMPTION 1: Exponential stability of the *autonomous* system (1) ,with $u(t) \equiv 0$, implies existence of Lyapunov functions of the form $V(x) = x^T(t)P(x)x(t)$ and derivative of the form $\dot{V}(x) = -x^T(t)Q(x)x(t)$, where P(x) and Q(x) are positive definite for all $x \in R^n$.

After establishing the basic definitions and facts, we can start presenting the properties of the adaptive controller. The following result shows the stabilizing properties of the main part of the adaptive controller that we propose.

RESULT 1: Let us assume that the plant

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \qquad (9)$$

$$y(t) = C(x)x(t) \qquad (10)$$

can be stabilized by some constant output feedback, K_y. In other words the fictiticus closed-loop system is **exponentially stable** according to assumption 1. Let us define $y_a(t) \triangleq y(t) + K_y^{-1}u(t)$, and use the adaptive algorithm

$$u(t) = -K(t)y_a(t) \qquad (11)$$

with the integral adaptive gain K(t) = N(t) given by

$$\dot{N}(t) = y_a(t)y_a^T(t)\Gamma \qquad (12)$$

(where Γ is a selected positive definite scaling matrix) or (for each scalar gain)

$$\frac{d}{dt}N_{ij} = \gamma_{ij}y_{a_i}y_{a_j} \qquad (13)$$

The simple algorithm (11)-(13) guarantees boundedness of all values involved in the adaptation process, namely, states, outputs, and adaptive gains, and asymptotically perfect regulation for the augmented system (Fig. 2)

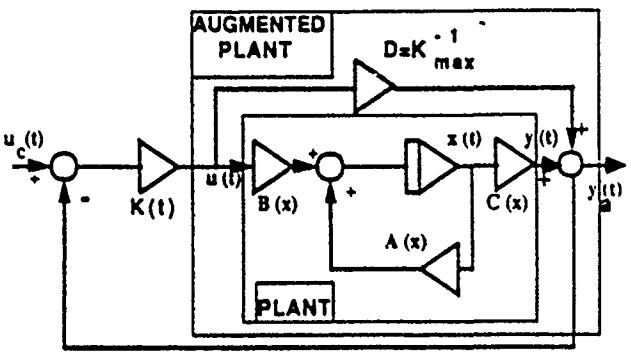


Fig. 2. The closed-loop control system.

$$\dot{x}(t) = A(x)x(t) + B(x)u(t) \qquad (14)$$

$$y(t) = C(x)x(t) \qquad (15)$$

$$y_a(t) = y(t) + K_y^{-1}u(t) = C(x)x(t) + K_y^{-1}u(t) \qquad (16)$$

such that $y(t) \to 0$ *and* $y_a(t) \to 0$ as $t \to \infty$. The gains ultimately reach some constant values which allow perfect regulation.(Appendix A).

We want to use the stabilizing nonlinear algorithm (12), in combination with other adaptive terms in order to implement a stable trajectory-following adaptive algorithm. Let the teacher generate the desired trajectories that the plant must follow. Let

$$f(u_t, x_t, y) = [\,(y_t - y)^T,\ x_t^T,\ u_t^T\,]^T \qquad (17)$$

where the feature vector $f(u_t, x_t, y)$ uses all values that can be measured, like the input commands, the teacher's states, and the tracking errors. We could try to use the adaptive algorithm

$$K_{ij}(t) = M_{ij}(t) + N_{ij}(t) \qquad (18)$$

$$M_{ij}(t) = \alpha_{ij}(y_{t_i} - y_i)f_j \qquad (19)$$

$$\frac{d}{dt}N_{ij}(t) = \gamma_{ij}(y_{t_i} - y_i)f_j \qquad (20)$$

for perfect tracking in idealistic environments. However, we are aware that (18)-(20) must further be adjusted to be applicable in realistic environments. It is clear that the perfect integrator in (20) increases without bound whenever perfect tracking is not possible. Since the nonlinear system includes uncertainties that we do not assume to know or identify, and since moreover, input and output disturbances may usually be present, we do not start with the proof of perfect tracking, which could be obtained in some ideal situation, but concentrate on the proof of robust stability under nonideal conditions. By *"robust stability"* we mean boundness of all values involved in the adaptation process like states, adaptive gains and errors, and tracking with arbitrarily small tracking errors. Perfect tracking in idealistic conditions is a particular case of the general robust tracking under nonideal conditions. The only change needed to guarantee robustness of the adaptive system is the addition of the by-pass term $-\beta_{ij}N_{ij}(t)$ to the right term of (20), and get the complete algorithm (5)-(8).

RESULT 2 : Under the assumptions of result 1, the adaptive algorithm (5)-(8) guarantees *robust stability* of the system (14)-(16) [Guez and Bar-Kana, 1989].

Let us assume that the plant needs some general dynamic configuration to reach stability. Specifically, let H: $\{A_f, B_f, C_f, D_f\}$ be some LTI dynamic feedback controller that guarantees that the closed loop system is exponentially stable according to assumption 1. Then we can state the following result:

RESULT 3 : Let $G(\cdot)$ be some nonlinear system of the form (9)-(10) and let H: $\{A_f, B_f, C_f, D_f\}$ be a stabilizing controller under assumption 1. Then, the adaptive algorithm (5)-(8) guarantees *robust stability* of the augmented system $G_a(\cdot) = G(\cdot) + H^{-1}$ [Guez and Bar-Kana, 1989].

More important, when *improper* linear controllers H are needed to stabilize the nonlinear plant, we can use their *proper inverse* in parallel with the plant. This way, we only use the knowledge on the *existence* of an improper controller and actually use a proper configuration in parallel with our plant. Specifically, as in the case of nonlinear robotic manipulators, PD controllers of the form $H(s) = K(1+qs)$ can stabilize the manipulators, and if K is very large, we can get very good tracking in ideal situation. However, in practice we do not want to use differentiators or high gains in control loops. In our approach we only use the knowledge on their mere *existence* to implement simple first order poles of the form $H^{-1}(s)=\frac{D}{1+qs}$ in parallel with the plant, where $D= K^{-1}$ can be a very small gain.

Notice that we do not guarantee any more that the plant is perfectly tracking, because the best we can obtain is $y_a(t) = y(t) + Du(t) \rightarrow y_t(t)$, although we actually want $y(t) \rightarrow y_t(t)$. Still, if the maximal admissible gain K_{max} is large compared with the gain of the plant, then $y_a(t) = y(t) + Du(t) \approx y(t)$. For a quite common example, assume that the gain of the plant is 10 and the maximal stabilizing gain $K_{max} = 100$. Then we use $D = 1/100 = 0.01$ in parallel with 10 and, as the references show, $y_a(t) \approx y(t)$ for all practical purposes.

ROBOTIC EXAMPLE

The proposed controller (5)-(8) was applied to the nonlinear robotic system [Desa and Roth, 1985]:

$$\dot{x}_1 = x_2 \tag{21}$$

$$\dot{x}_2 = D^{-1}[u_1 + x_2x_4(B_1 - B_3)\sin(2x_3)] + d_{i_1}(t) \tag{22}$$

$$\dot{x}_3 = x_4 \tag{23}$$

$$\dot{x}_4 = [u_2 - (x_2^2/2)(B_1 - B_3)\sin(2x_3)]/B_2 + d_{i_2}(t) \tag{24}$$

where

$$D = A_1 + m_B L_1^2 + B_1\cos^2 x_3 + B_3\sin^2 x_3,$$
$$L_1 = 0.1\ m^3,\ A_1 = 0.01\ kgm^2,\ m_B = 0.6\ kg,$$
$$B_1 = 0.06\ kgm^2,\ B_2 = 0.01\ kgm^2,\ B_3 = 0.05\ kgm^2.$$

The output measurements are

$$y_1 = x_1 + d_{O_1}(t) \tag{25}$$

$$y_2 = x_2 + d_{O_2}(t) \tag{26}$$

where d_i and d_O are the input and output disturbances. The teacher was given by the simple decoupled model:

$$\dot{x}_t(t) = \begin{bmatrix} -25 & 0 \\ 0 & -25 \end{bmatrix} x_t(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_t(t) \tag{27}$$

$$y_t(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_t(t) \tag{28}$$

and it was tested with demanding square-wave input commands.

In paralel with the plant (21)-(24) we employ the supplementary feedforward [Bar-Kana, 87, 89a]

$$y_s(s) = \frac{\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}}{1+s/10} u_s(s) \tag{29}$$

Supplementary disturbances were added to check robustness

$$d_i(t) = \begin{bmatrix} 5 \\ 10\sin(100t) \end{bmatrix};\quad d_O(t) = \begin{bmatrix} 0.1 \\ 0.2\sin(120t) \end{bmatrix} \tag{30}$$

and the adaptation coefficients were

$$\alpha_{ij} = \gamma_{ij} = 100.;\ \beta_{ij} = 0.1 \tag{31}$$

Results of simulations are shown in Figure 3. Figure 3a compares the first plant and model output, and figure 3b shows the second output. Figure 3c shows, for illustration, the behavior of the adaptive gain $K_{22}(t)$. It can be seen how the adaptive gain moves up-and-down in order to maintain small tracking errors. Figure 3d represents the norm of all plant states, to show that no hidden state diverges. The input disturbances were introduced from the start and their effect is hardly felt at the output. The output disturbances were then introduced at $t_1 = 0.16$sec and $t_2 = 0.23$ s. All values remained bounded while the plant tracked with small errors, although we

used only position (no velocity and no acceleration) output measurements.

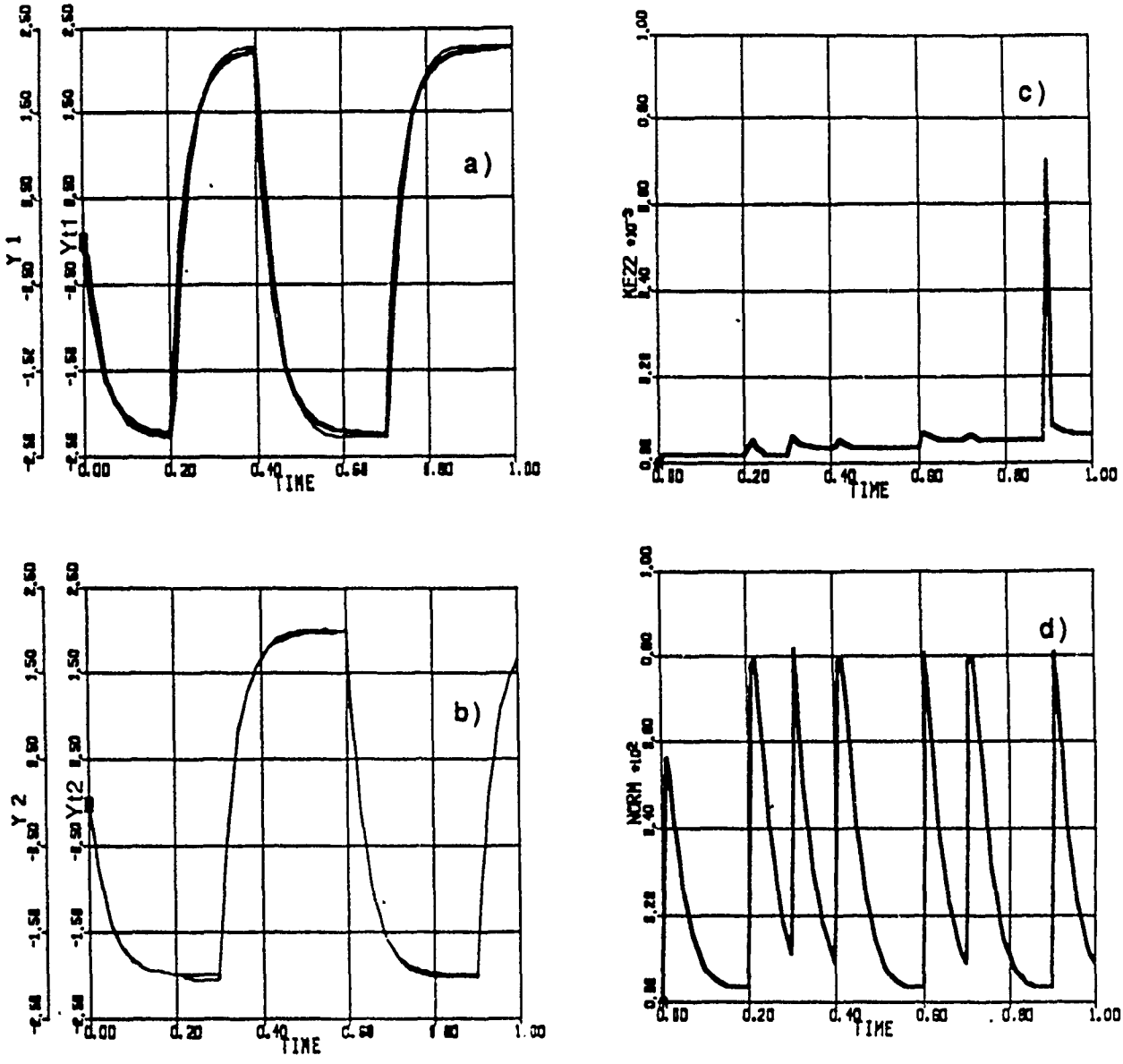


Figure 3. **Example:** (a) Output $y_{t_1}(t)$, $y_1(t)$; (b) Output $y_{t_2}(t)$, $y_2(t)$; (c) Gain $K_{22}(t)$; (d) Norm of all plant states.

CONCLUSIONS

This paper presents a parallel distributed computing architecture for adaptive control. Starting with some prior assumptions about stabilizability of the plants it results in a stable unsupervized architecture. The feasibility of the method is demonstrated on an example of robotic manipulator.

APPENDIX A - PROOF OF RESULT 1

The control system with the adaptive controler (18)-(20) is

$$\begin{aligned}\dot{x}(t) &= A(x)x(t) - B(x)N(t)y(t)\\ &= A(x)x(t) - B(x)K_e C(x)x(t) + B(x)K_e C(x)x(t) - B(x)N(t)C(x)x(t)\\ &= [A(x) - B(x)K_e C(x)]x(t) - B(x)[N(t)-K_e]C(x)x(t)\\ &= A_s(x)x(t) - B(x)[N(t)-K_e]C(x)x(t) \qquad (A.1)\end{aligned}$$

$$y(t) = C(x)x(t) \qquad (A.2)$$

We must prove stability of all the values involved in the adaptation process, like states, outputs, gains. Let us select the quadratic Lyapunov function

$$V(t) = x^T(t)P(x)x(t) + tr[(N(t) - K_e)\Gamma^{-1}(N(t) - K_e)^T] \qquad (A.3)$$

where *tr* denotes trace. Notice that the second term in (A.3) is only a short notation for the sum of all terms of the form $[N_{ij}(t) - K_{ij}]^2/\gamma_{ij}$. Therefore, (A.3) is a positive definite quadratic function of all states and adaptive gains. The derivative of the Lyapunov function is

$$\begin{aligned}\dot{V}(t) &= x^T(t)\dot{P}(t)x(t) + x^T(t)P(x)\dot{x}(t) + \dot{x}^T(t)P(x)x(t)\\ &\quad + tr[(N(t) - K_e)\Gamma^{-1}\dot{N}^T(t)] + tr[\dot{N}(t)\Gamma^{-1}(N(t) - K_e)^T]\\ &= x^T(t)[\dot{P}(t) + P(x)A_s(x) + A_s^T(x)P(x)]x(t)\\ &\quad - x^T(t)P(x)B(x)[N(t)-K_e]C(x)x(t)\\ &\quad - x^T(t)C^T(x)[N(t)-K_e]^T B^T(x)P(x)x(t)\\ &\quad + tr[(N(t) - K_e)\Gamma^{-1}\Gamma y(t)y^T(t)]\\ &\quad + tr[y(t)y^T(t)\Gamma\Gamma^{-1}(N(t) - K_e)^T] \qquad (A.4)\end{aligned}$$

By using (14)-(15) we get

$$\begin{aligned}\dot{V}(t) &= -x^T(t)Q(x)x(t) - 2x^T(t)C^T(x)[N(t)-K_e]C(x)x(t)\\ &\quad + 2x^T(t)C^T(x)[N(t)-K_e]C(x)x(t) = -x^T(t)Q(x)x(t) \le 0 \qquad (A.5)\end{aligned}$$

Notice that because $\dot{V}(t)$ in (A.5) is not a function of the adaptive gains, we only claim in (A.5) that the derivative of the Lyapunov function is positive semidefinite, namely $\dot{V}(t) \le 0$. The selected positive definite quadratic form of V(t) then guarantees that all states *and* adaptive gains are bounded. The trajectories of the adaptive control system finally reach the region defined by $\dot{V}(t) \equiv 0$ (LaSalle, 1981).

It is easy to see that $\dot{V}(t) \equiv 0$ implies $x(t) \equiv 0$. Therefore, if we ignore the adaptive gains, the plant is asymptotically perfect regulator, but what about the gains? First of all, we proved already that the gains are bounded. Second, $x(t) \equiv 0$ implies $y(t) \equiv 0$, which implies $\dot{N}(t) \equiv 0$ and then $N(t)$ = constant. Therefore, N(t) ultimately reaches some constant value such that the plant is perfect regulator, as the theorem claims.

REFERENCES

Bar-Kana, I. (1987) "Parallel Feedforward and Simplified Adaptive Control," *International Journal of Adaptive Control and Signal Processing*, Vol. 1, No. 2, pp. 95-109.

Bar-Kana, I.(1989a) "On Positive Realness in Multivariable Stationary Linear Systems," *Proceedings of 1989 Conference on Informational Sciences and Systems - CISS '89*, Baltimore, Maryland, pp. 383-388; also (full report) Technical Report, Drexel University.

Bar-Kana, I. (1989b) *On Passivity of a Class of Nonlinear Systems*, Technical Report, Drexel University.

Guez, A., and Bar-Kana, I.(1989) *Unsupervised Parallel Distributed Computing Architecture For Adaptive Control*, (full report) Technical Report, Drexel University.

Desa, S., and Roth, B., (1985) "Synthesis of Control Systems for Manipulators Using Multivariable Robust Servomechanism Theory," *International Journal of Robotics*, Vol. 4, pp. 18-34.

Guez, A. (1988) "Neurocontrollers," *NSF Workshop on Neurorobotics*, New Hampshire, 1988.

Hahn, W. (1967) *Stability of Motion*, Springer Verlag, New York.

LaSalle, J. (1981) "Stability of Noautonomous Systems," *Nonlinear Analysis Theory, Methods and Applications*, Vol. 1, No. 1, pp. 83-91.

Morse, W., and Ossman, K. (1989) "Flight Control Reconfiguration Using Model Reference Adaptive Control," *Proceedings of 1989 American Control Conference*, Pittsburgh, Pennsylvania, pp. 159-164.

Widrow B. and Stearn S. (1985) *Adaptive Signal Processing*, Prentice Hall.

[24]

A NEURO-EXPERT ARCHITECTURE FOR OBJECT RECOGNITION

John Selinsky*, Allon Guez*, James Eilbert** and Moshe Kam*

* Department of Electrical and Computer Engineering
Drexel University, Philadelphia, PA 19104

** Grumman Corp. Research Center
m/s A01-26 Stewart Avenue
Bethpage, New York 11714

Abstract

This article reports on results of experiments in object recognition with a combined neural network / expert system architecture (Neuro-Expert). The Neuro-Expert architecture is outlined with a description of the experimental object recognition system. Results are reported for the recognition of a 20 pattern prototype set of synthesized binary images placed at arbitrary rotations. A 100% recognition rate was obtained under noiseless conditions. Addition of 1% and 2% random pixel noise resulted in recognition rates of 95.2% and 89.5% respectively.

A NEURAL NETWORK APPROACH TO TARGET RECOGNITION

Sanjay S. Kumar and Allon Guez
Department of Electrical and Computer Engineering
Drexel University, Philadelphia PA 19104

ABSTRACT

In this paper, an approach for distortion invariant target identification and target classification based on the Adaptive Resonance Theory-II (ART-II) is presented. The neural network employed demonstrates fast unsupervised learning coupled with stable retention and retrieval of information. To keep the dimensions of the network to a bare minimum and to speed up the computational process as well as to achieve invariance, six distortion invariant features are extracted from each target image and are used as network inputs. These continuous valued features are derived from the geometrical moments of the image. The ART based target recognition system, (ATRS) can be used in two different modes a) as a target classifier and b) as a target recognition system. Several parameters associated with the network itself, allow for greater flexibility in feature manipulation. The performance of the ATRS is compared with a similar system employing the Multi Layer Perceptron, (MLP) with the Back Error Propagation learning algorithm. The ATRS is found to perform better on three major counts: Speed of processing, flexibility in feature manipulation and noise tolerance. Determination of critical settings of the various parameters associated with the network is crucial in tuning the ATRS to ensure stable and consistent performance.

1. INTRODUCTION

The problem of recognition and image interpretation is an old one. It involves two major components: a) preliminary processing, involving extrapolation of information from given sensory data leading to an intrinsic image and b) knowledge based interpretation of the intrinsic image yielding information about its contents[11][12]. What is new is the deployment of varied associative neural network models that attempt to solve the problems associated with area b) mentioned above.

This research was partially supported by a grant from the Air Force, Grant No. AFOSR 890010.

Associative neural networks operate by correlating input data with internally stored templates to determine best match. This correlation is performed almost instantly with essentially no searching or sequential testing and is in the form of an association between the vast array of information that describes the current situation and its approximate context. Such networks process more information and exhibit a higher degree of fault tolerance.

Most associative neural network models that are employed in feature extraction and object recognition involve an explicit learning or training session. During training they have to be specifically familiarized with the objects in the problem domain they are expected to identify. This training procedure is usually lengthy and restricts the process of recognition to only those features that have been learnt by the network. To circumvent these drawbacks, we have considered a neural network architecture based on the Adaptive Resonance Theory-II to solve the problem of on line target recognition. We have shown in this paper that this network gives an improved performance that reflects significantly upon the processing power and speed of computation.

2. PROBLEM STATEMENT

Fast image identification and interpretation is crucial for any intelligent, real time, target recognition system. Accuracy of target identification, noise tolerance and the capacity to handle image distortions are aspects that play an equally important role in their design. We have proposed an implementable, yet simplistic scheme for target classification and target recognition in the form of the ATRS, that has associated with it, some of the attributes mentioned above. The ATRS treats a target as a collection of sensory data describing the object represented in the form of an image or simply a scanner photograph of the same, detached from its background. Given a set of target images the recognition system, as a target classifier, is expected to classify the target images into appropriate target categories without any external supervision or prior learning. No restrictions are placed on scaling, translation and orientation of the target image and the amount of admissible noise.

In the recognition mode, the set of all possible prototype target images encountered in the problem domain, called the *prototype set*, is presented to the ATRS prior to any testing. This supervised mode of operation enables us to associate with each prototype category established, a target name or identification number. Learning of the prototype features is expected to take place on a single presentation of the prototype set. Once the prototype categories are established, the ATRS is called upon to correctly identify any given target image that may be a distorted or noise prone version of the prototype.

3. IMPLEMENTATION OF THE ATRS

In this section we present our approach to the solution of the problem stated in section 2. Figure 1. is a general schematic diagram for the Art based Target Recognition System :

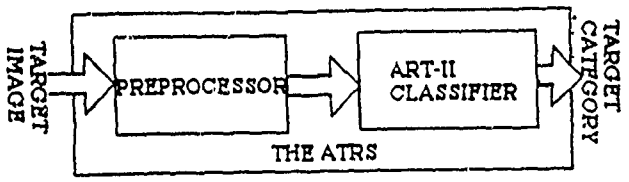


Figure 1. Schematic of the ATRS.

Functioning of the ATRS :

When used in the target classification mode, the ATRS is presented with a stream of differently oriented, scaled and translated binary images of targets. These images are quickly encoded by the systems moment generating preprocessor and are passed down to the ART-II classifier that instantly classifies the images into appropriate target categories. The output of the ATRS is a category structure that describes which of the input target images belong to which category and the number of different target categories established thereof. In the recognition mode, the ATRS is first presented with the prototypes before it is called upon to identify an arbitrary target image. It should be noted that the ATRS is sensitive to novelty. If a particular target image has not been seen before, during the prototype presentations, a new target category is established.The output of the ATRS in this mode is the identification name or identification number of the input target image. The various components of the ATRS are described in the following sections:

The moment generating preprocessor :

One of the primary issues involved with the problem of image interpretation is that of coding information contained in the intrinsic image. Coding reflects directly upon the size of the network, its storage capacity and the speed of computation. We have restricted ourselves to the use of features of the intrinsic target image instead of the image per se. The moment generating preprocessor takes a two dimensional M X M image of the target and encodes it into a compact feature vector comprising of six components. This feature vector is derived using a set of nonlinear moment invariant functions defined on the geometrical moments of the target image[1][4][13]. These features are translation, orientation and scale invariant. They were introduced by Hu and were later applied in aircraft identification by Dudani et.al.[3][5]. The moment generating preprocessor used in the ARTS is different from the one used by Stephen Grossberg and Gail Carpenter [9], that employs the Fourier Mellon invariant filter to achieve 2-D spatial invariant image coding. However, in either case, the target image has to be isolated from the image background before it can be input to the moment generating preprocessor. On the other hand, this problem can be circumvented by determining critical settings of the vigilance and noise tolerance parameters associated with the network itself.

Given a two dimensional M X M image g(x,y), {g(x,y),x,y = 0,1,.....M-1} the (p+q)th geometrical moment is defined by:

$$m_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{M-1} x^p y^q g(x,y) \qquad p,q = 0,1,2...$$

Scale invariance is achieved if the M X M target image is mapped on to a square defined by X and Y belonging to [-1,+1]. The grid locations will no longer be integers but will have real values in the range [-1,+1].

$$m_{pq} = \sum_{x=-1}^{+1}\sum_{y=-1}^{+1} x^p y^q g(x,y) \qquad p,q = 0,1,2...$$

To make the moments translation invariant, the central moments are defined as:

$$\mu_{pq} = \sum_{x=-1}^{+1}\sum_{y=-1}^{+1} (x-\bar{x})^p (y-\bar{y})^q g(x,y) \qquad p,q = 0,1,2...$$

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}$$

Scale invariance can also be achieved by taking into account the central moments with:

$$\Omega_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\beta}} \text{ and } \beta = \frac{p+q}{2} + 1$$

A set of nonlinear functions defined on Ω_{pq} that are invariant to rotation translation and scale changes have been derived in [1].The first six functions are:

$$f_1 = \Omega_{20} + \Omega_{02}$$

$$f_2 = (\Omega_{20} - \Omega_{02})^2 + 4\Omega_{11}^2$$

$$f_3 = (\Omega_{30} - 3\Omega_{12})^2 + (3\Omega_{21} - \Omega_{03})^2$$

$$f_4 = (\Omega_{30} + \Omega_{12})^2 + (\Omega_{21} + \Omega_{03})^2$$

$$f_5 = (\Omega_{30} - 3\Omega_{12})(\Omega_{30} + \Omega_{12})[(\Omega_{30} + \Omega_{12})^2 - 3(\Omega_{21} + \Omega_{03})^2] + (3\Omega_{21} - \Omega_{03})(\Omega_{21} + \Omega_{03})[3(\Omega_{30} + \Omega_{12})^2 - (\Omega_{21} + \Omega_{03})^2]$$

$$f_6 = (\Omega_{20} - \Omega_{02})[(\Omega_{30} + \Omega_{12})^2 - (\Omega_{21} + \Omega_{03})^2] + 4\Omega_{11}(\Omega_{30} + \Omega_{12})(\Omega_{03} + \Omega_{21})$$

The feature vector representing the target image is given by:

$\text{Log}\,|\Omega_i|, \quad i = 1,\ldots,6$

The ART-II classifier :

A neural network based on the Adaptive Resonance Theory-II is adopted as the primary classifier in the ATRS. The Adaptive Resonance Theory-II is a neural network architecture that is capable of learning recognition categories. Recognition categories are described as subsets of environmental input patterns, continuous valued in the case of ART-II, that share invariant properties. The network extracts and learns to recognize these invariants to enable *self organization* and *self stabilization* of its recognition codes in response to an arbitrary sequence patterns. Recognition codes are represented as stable states and their spontaneous emergence through the systems interaction with the environment is described as self organization[6][7][8].

The network used comprises of six input channels at the STM stage designated as F_1. These input channels are directly linked to the ATRS moment generating preprocessor. The number of nodes at the STM stage F_2 is kept as a variable depending upon the the number of prototype target categories that are to be established. An important aspect of the network identification process is the determination of critical settings of the various parameters associated with the network. One such parameter is the associative vigilance parameter that determines how fine or coarse the learned categories will be. Another parameter associated with the network is the noise tolerance parameter that determines the level of noise tolerated by the system. The ART-II network topology used in [7][6], is shown in figure 2:

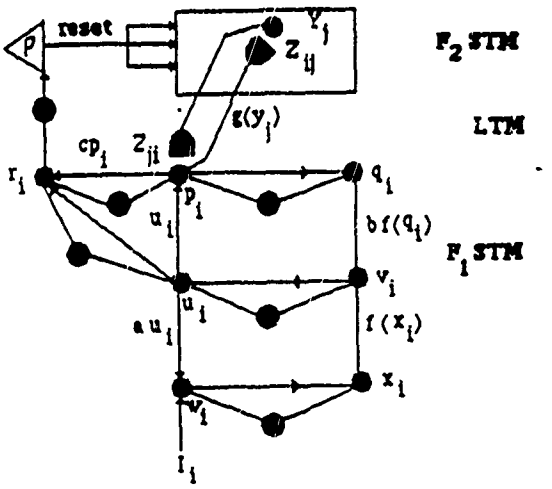


Figure 2. ART-II network topology.

For the convenience of the reader, we present the network equations as in [6], below:

The local STM activities p_i, q_i, u_i, w_i and x_i computed at stage F_1 are as follows: For i = 1,.....,M and j = M+1,.....,N where, M: number of F_1 input channels and N: number of STM nodes at stage F_2

$$p_i = u_i + \sum_j g(y_j) z_{ji} \qquad q_i = \frac{p_i}{e + \|p\|}$$

$$u_i = \frac{v_i}{e + \|v\|} \qquad v_i = f(x_i) + bf(q_i)$$

$$w_i = I_i + au_i \qquad x_i = \frac{w_i}{e + \|w\|}$$

(All equations are expressed in a dimensionless form)

$\|v\|$ represents the L_2norm of the vector v. Vector norms are shown as large filled circles in figure 2, while the transfer of information within the network is represented by the corresponding arrows.

$f(x) = 0$ if $0 < x < \varnothing$
$\quad\quad 1$ if $x > \varnothing$

STM parameters: a = 10.00 b = 10.00 c = 0.1 d = 0.9 ($0 < d < 1$)

noise tolerance parameter, $\varnothing = 1 / \sqrt{M}$.

I_i : Input vector component

Z_{ji} : Top down connection weights

Z_{ij} : Bottom up connection weights

The STM Equations at stage F_2:

$$T_j = \sum_i p_i Z_{ij} \qquad T_J = \max\,[\,T_j : j = M+1,.....N]$$

$g(y_j) = d$ if $T_J = \max\,[\,T_j$: the jth F_2 node has not been reset on the current trial]

$\quad\quad 0$ otherwise

The order of presentation is unimportant. Target images also include noise upto 3%. See figure 5.

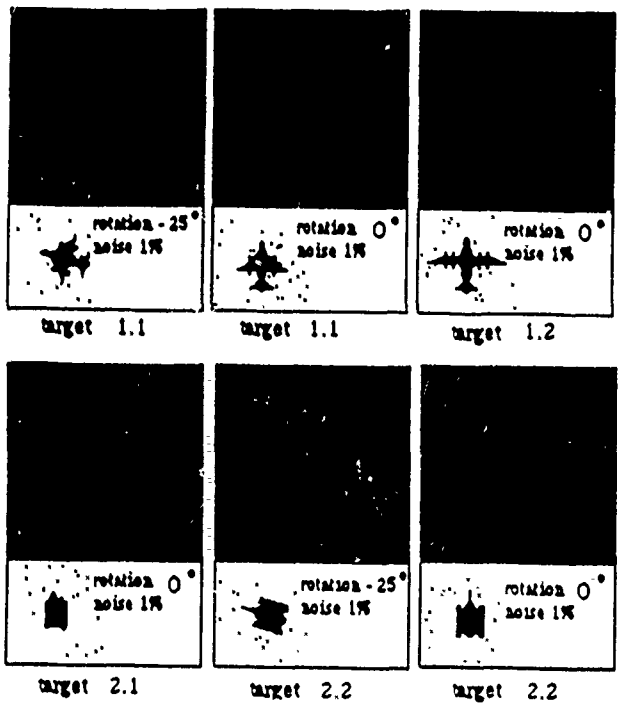


Figure 5. Examples of target images (rotation and noise)

The ATRS is capable of identifying all these target images with a near 100% accuracy. When a new target image, one that has not been seen by the network before, is input, the ATRS establishes a new target category to represent it. In the on line case, when the prototype pattern set is not deterministic and is not presented before testing, It should be noted that the category structure established by the ATRS for a given set of target images is different from the one established when the order of presentation is altered. In this situation, the ATRS establishes a separate target category for the first target image presented. If the following target image is the same as the first or is a distorted variation of it, the ATRS groups it with the already established target category associated with the image. In the default case, when the following target image is different from the one seen before, a new target category is established.

A disadvantage in employing the ART-II as the major classifier in the recognition system, as it is used in the proposed model, is the its incapacity to handle noise beyond a fixed percentage. This problem is related to fixing the attentional vigilance and the noise parameters during the recognition process.

The vigilance parameter associated with the network allows for feature manipulation. It defines how coarse or fine the recognition categories will be. If the vigilance parameter is kept too low there is a possibility of false grouping. However, this feature allows for dealing with noisy feature vectors apart from the noise tolerance parameter of the network. Determination of on line critical vigilance tuning to obtaining a desired categorization of target images is the subject of ongoing research. In the simulation carried out all parameters are kept constant.

A crossection of the sample data relating to Figures 4,5 and 6, showing the different moments, the noise associated with the images, the rotations, the actual target category and the category established by the ATRS are shown in Table 1.1.

A comparison of the ATRS with the Multilayer Perceptron using the Back Error Propagation Learning Algorithm is summarized in Table 1.2[14]. The data set used in the comparison is kept the same. Seventy two target images with variable percentages of noise and distortions are presented. The various parameters of the network are kept fixed during the presentations.

C O M P A R I S O N T A B L E					
NETWORK	# of iterations per pattern	% noise	# of patterns	%of patterns classified	% of false classification
BEP	3000	0	72	100	0.00
		1	72	97.8	4.8
		2	72	95 5	10.5
		3	72	91 2	19.7
ART-II	1	0	72	100	0.00
		1	72	100	0.00
		2	72	100	2.64
		3	72	100	8.87

Table 1.2: Performance comparison of the ART-II with the BEP

5. DISCUSSION

We have shown in this paper a new technique for the implementation of a neural network based approach to on line target recognition and target classificanon employing a neural network model that is capable of self organization and fast unsupervised learning. A comparison with another popular model, the BEP, has brought into sharper focus some of its attributes. The approach taken is straight forward and simple. Future work in this direction includes integrating the ATRS with

MOMENTS						TARGETS			
FEATURE VECTOR						CONDITIONS		IDENTIFICATION CATEGORY NO.	
f1	f2	f3	f4	f5	f6	rotation	noise	ACTUAL	ATRS
3.035	2.513	-2.685	-1.146	-3.061	0.110	0	0	1.1	1.1
3.828	4.815	3.746	0.125	2.061	2.532	0	0	1.2	1.2
3.051	4.622	2.345	-2.027	-2.138	0.104	0	0	1.3	1.3
2.256	2.122	-1.510	-7.077	-12.39	-7.773	0	0	2.1	2.1
2.784	1.770	0.119	-2.316	-3.414	-1.431	0	0	2.2	2.2
2.626	1.954	0.092	-3.476	-5.167	-2.499	0	0	2.3	2.3
3.013	2.502	-2.605	-0.960	2.984	0.291	25	0	1.1	1.1
3.805	4.850	3.727	0.046	1.930	2.470	25	0	1.2	1.2
3.013	4.526	2.344	-2.135	-2.038	0.122	25	0	1.3	1.3
2.266	2.179	-1.705	-7.206	-12.33	-8.986	25	0	2.1	2.1
2.781	1.679	0.149	-2.386	-3.505	-1.547	25	0	2.2	2.2
2.622	2.003	0.182	-2.893	-4.250	-1.895	25	0	2.3	2.3
3.826	2.535	3.302	2.721	2.009	3.854	0	1.0	1.1	1.1
3.425	3.279	0.140	2.992	4.558	4.623	0	1.0	1.1	1.1
3.956	4.422	4.541	0.181	1.388	2.365	0	1.0	1.2	1.2
3.296	3.952	3.841	3.299	6.206	5.262	0	1.0	2.1	2.1
3.450	2.061	3.409	3.459	3.264	2.715	0	1.0	2.2	2.2
3.460	2.496	2.715	0.294	1.439	1.435	25	1.0	2.2	2.2

Table 1.1: A crossection of simulation data

a segmentation module that does a dynamic scene analysis, isolates the images from the scene and presents it to the recognition system. We also propose an on line adaptive tuning of the attentional vigilance parameter based on some measure of confidence, to further improve recognition in the ATRS.

Acknowledgements:

We wish to thank John Selinsky, Drexel University, for the ATRS test data and the comparison figures for the Multilayer Perceptron performance using the Back Error Propagation learning algorithm.

References

[1] M. Hu, "Visual Pattern Recognition by Moment Invariants", *IRE Trans. Information Theory*," Vol. IT-8, pp. 179-187, Feb. 1962.

[2] A. Khotanzad and Y.H. Hong, "Rotation and Scale Invariant Features of Texture Classification," *Proc. of IASTED Int. Synposium Robotics and Automation*, Santa Barbara, CA, USA, pp. 16-17, May 1987.

[3] A. Khotanzad and J.H. Lu, "Distortion Invariant Character Recognition by a Multi-LayerPerceptron and Back Propagation Learning," *Proc. of* IEEE *International Conference on Neural Networks*,SanDiego, CA, USA, pp. I-625-630, July 1988.

[4] M. Hu, "Visual Pattern Recognition by Moment Invariants,"*IRE Trans. Information Theory*," Vol. IT-8, pp. 179-187, Feb. 1962.

[5] S.A. Dudani, K.J. Breeding and R.B. McGhee, "Aircraft Identification by Moment Invariants," IEEE*Trans. on Computers*, Vol. C-26, No. 1, pp. 39-45, Jan. 1977.

[6] Gail A. Carpenter and Stephen Grossberg, "ART:2 Self Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, Vol. 26. No. 23, December 01,1987 pp.4919-4930.

[7] Gail A. Carpenter and Stephen Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision Graphics Image Process.*., 37. 54 (1987).

[8] Gail A. Carpenter and Stephen Grossberg, "ART 2: Stable Self Organization of Pattern Recognition Codes for Analog Input Patterns," *Proc. of First International Conference on Neural Network*, San Deigo (IEEE, New York, 1987).

[9] Gail A. Carpenter and Stephen Grossberg, "Invariant Pattern Recognition and Recall by an Attentive Self Organizing ART Architecture in a Nonstationary World," *Proc. of First International Conference on Neural Network*, San Deigo (IEEE,New York, 1987).

[10] D.E.Rumelhart and J.L Mc Clelland, Parallel Distributed Processing: Exploration into the Microstructure of Cognition, Vol. 1: Foundations, MIT press, 1986.

[11] Eugene Charniak and Drew McDermott, Introduction to Artificial Intelligence, Addison-Wesley Publishing Co., 1986.

[12] A. Rosenfeld, "Computer Vision: Basic Principles," *Proc. of the* IEEE ,Vol. 76, No. 8,pp. 863-868, Aug. 1988

[13] P.J. Besl, "Geometric Modeling of Computer Vision," *Proc. of the* IEEE ,Vol. 76, No. 8,pp. 936-955, Aug. 1988.

[14] Selinsky et.al., "Neuro-Expert Architecture for Object Recognition," Drexel University, *submitted for publication*, Jan. 1989.

[26]

ACCELERATED CONVERGENCE IN THE INVERSE KINEMATICS VIA MULTILAYER FEEDFORWARD NETWORKS

Allon Guez and Ziauddin Ahmad
Department of Electrical and Computer Engineering
Drexel University, Philadelphia, PA 19104

Abstract

A hybrid approach to the iterative solution of robotic manipulators has been presented. In this method a Multilayer Feedforward Network (MFN) is trained to provide an approximate solution to the inverse kinematic problem of a robot. This approximate solution is then utilized as the initial guess to the iterative procedure which provides the solution within some specified tolerance. Examples involving the PUMA 560 robot are given. Since the time taken by the MFN to provide the initial estimate is only a fraction of the time taken by the iterative procedure to reach a solution, this combination can be usefuly employed in order to decrease the time for obtaining the solution to the inverse kinemetic problem in robotics.

1. Introduction

Among the methods of controlling a robot the inverse kinematic method of control when fully utilized requires a total knowledge of the kinematics of the robot. A robot is composed of 'links' connected in the form of a chain with the so called 'joints' also known as the degrees of freedom (DOF). The determination of the position of these links in space, given the joint displacements is called the 'forward kinematics'. The problem of finding the joint displacements from a specification of the desired endeffector's position and orientation is called the inverse kinematic problem (IKP).

The task of a robot requires it to manipulate different objects. These objects are represented in a single coordinate system or multiple coordinate systems related to each other with a single (global) coordinate system. The handling of the objects is done by the hand or the 'endeffector' of the manipulator. The necessity of the inverse kinematics is thus a natural consequence. Unlike the forward kinematics problem the IKP usually does not have a unique solution. Moreover if the system is underdetermined it will have infinite number of solutions. This later type constitutes the redundant class of robots [4],[13].

Research Partially supported by grant No. AFOSR890010.

A variety of procedures for solving the inverse kinematics of these robots have been developed [2],[10]. Robots having finite number of solutions have been usefully employed. The limitations to successfully operate only such robots is largely due to real time controllability. The class of robots that do not have a closed form solution, also called unsolvable manipulators, require iterative procedures in order to obtain the joint values that will position the endeffector in the desired position and orientation.

A robot with 6-DOF is required in order to arbitrarily place an object in a 3D space. The class of 6-DOF robots for which closed form solution to their inverse kinematics exists is small. Moreover, the inverse kinematic solution of these robots is usually very involved and requires some intuition into the kinematic structure. The 6-DOF robots that do not have a closed form solution are usually solved by iterative procedures such as Newton Raphson and gradient descent methods [4]. Robots having greater than minimum number of DOF necessary for their task are called redundant robots. These robots are solved either by fixing the redundant DOF and solving the remaining joints in a closed form manner, or by using iterative methods so as to obtain a solution.

This paper discusses problems in using Multilayers Feedforward Networks (MFNs) to learn the inverse kinematic solution of robots. Also shown is the computational efficiency achieved when MFNs are used as an initial guess for the inverse kinematic solution by iterative methods. An example involving PUMA 560 robot illustrates this procedure.

2. Problem Statement

The forward kinematic problem may be defined as: given 'q' the n dimentional joint variables vector, find 'X' the m dimentional endeffector position and orientation specificaton vector in the Work Space. This can be expressed as

$$X = f(q) \qquad (1)$$

where f is a nonlinear, continuous and differentiable function relating the joint variables to the work space coordinates. In these terms the inverse kinematic problem is defined as: given 'X' the m dimentional endeffector position and orientation vector, find 'q' the n dimentional joint variable vector. That is to say to solve Eq. (1) for

q given the vector X.

$$q = f^{-1}(X) \quad (2)$$

Contrary to the forward kinematic problem the IKP in general does not have a unique solution. In the method proposed in this paper only the forward kinematics is utilized to solve the inverse kinematics of a robot.

3. Proposed Method

Our approach to the solution of IKP of robots involves an iterative procedure for which the initial guess is given by trained MFNs. Fig. 1 shows that the iterative method using the initial guess

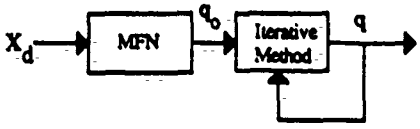


Fig. 1: Schematics of the proposed method employing the MFN to provide the initial guess q_0 to the iterative procedure.

q_0 given by the MFN. The MFN is provided with the desired position and orientation 'X_d' of the endeffector of the robot and it provides the estimate q_0 of the joint angles. This estimate is utilized by the iterative procedure to provide with a solution q that satisfies a specified tolerance for the position and orientation of the endeffector.

The MFN is trained in a supervised manner using multilayered BEP Algorithm. The training data constitutes the $f_6(q)$ vector as the input and the corresponding 'q_n' vector as the desired output of the network, where the subscripts signify the dimension of the vector. This pair of data can be generated using only the forward kinematics of the robot. The problem related to arbitrary data points is that in general there are more than one solution to the inverse kinematics of a robot, which requires us to select only one 'type' of solution out of all the possibilities as the training data set. Each 'type' of solution is bounded by the so termed singular regions which correspond to the rank deficiency of the Jacobian of the manipulator [13]. Thus the data set corresponding to one type of solution can be generated by starting at any configuration/joint values, and storing the relevant data while incrementing in joint space and at each step ensuring that the jacobian is not near rank deficiency. The rank deficiency of the jacobian can be easily verified by taking the determinant of the jacobian (J) or that of (JJ^T) in case of under / over-determined manipulator and comparing the result with zero. Where the superscript indicates the transpose operation.

Another problem encountered is in the range of the joint angles. For a particular type of solution the entire joint range for each joint may not be covered. Thus only portions of the maximum joint range may be utilized. These portions if disjoint need to be trained by separate networks. This problem will be further clarified in the next section.

The Network Architecture

The proposed MFN architecture, shown in Fig. 2, consists of n (= number of joints) cluster of networks each cluster consisting of K_i networks. Where K_i is the number of disjoint regions in the solution of the *i* th joint, where *i* varies from 1 to n. There is a pair of output nodes for each K_ith network, one of which is trained to give the solution to the joint variable 'i' and the other is trained to indicate the validity of the solution given by the first node. This validity may be taught by making the 2nd node to give a 'high' if the solution lies in the domain of the corresponding 1st node and a 'Low' otherwise. The input nodes are equal to the dimension of the f(q) vector. Also two layers of hidden units are employed. All nodes have a symmetric sigmoidal activation function [12]. In order to have the solution to all the possible types we would require L such networks, where L is the number of types of solutions.

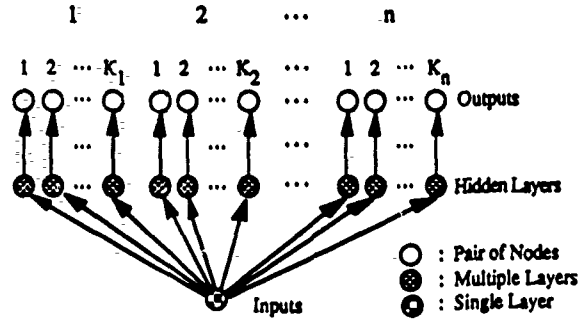


Fig. 2: The Architecture of the Network.

The training of the network consists of the supervised Back Error Propagation (BEP) algorithm used to train the output nodes corresponding to the joint values, given by:

$$q = [q_1\ q_2\ \dots\ q_n]^T \quad (3)$$

for the disjoint region of solution related to the position and orientation specification of the endeffector X_d which is provided to the input layer nodes. The position specification consists of the (x,y,z) displacement of the origin of the coordinate system attached to the endeffector with respect to the work space frame. The orientation specification consists of (ϕ,θ,ψ), the Eulerian angles [2].

$$X_d = [\phi\ \theta\ \psi\ x\ y\ z]^T \quad (4)$$

Where ϕ is the rotation about the OZ axis, θ is the rotation about the OU axis, and ψ is the rotation about the OW axis, in the given sequence, in endeffector's frame of reference.

The Iterative Procedure

The schematics of the iterative method is shown in Fig. 3. This procedure, commonly called the Newton Raphson method utilizes the linearized approximation as given below.

$$d(f(q_{(t)})) + J(q_{(t)},x_{(t)})\ dq_{(t)} = 0 \qquad (5)$$

Eq. (5) is solved for $dq_{(t)}$, which is the change in the $q_{(t)}$ required to give a better estimate for the joint values. The joint variables are updated as follows

$$q_{(t+1)} = q_{(t)} + dq_{(t)} \qquad (6)$$

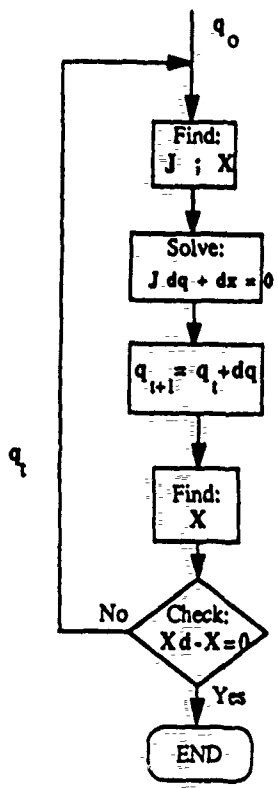


Fig. 3: Flow diagram for the iterative procedure to find the joint values.

Here, $d(f(q_{(t)}))$ is the 6 dimensional differential position & orientation vector. $J(q_{(t)},x_{(t)})$ is the (6 x n) jacobian matrix of the manipulator signifying the rates of each element of $f(q_{(t)})$ with respect to $q_{(t)}$. Where 'n' is the number of joints. $dq_{(t)}$ corresponds to the incremental change required in the joint variables to satisfy equation (5) for a 'better' solution. The subscript t corresponds to time steps. Here it may be mentioned that only the knowledge of the forward kinematics of the robot is sufficient for the computation of f(q) and J(q,x).

4. Examples

The solution to the inverse kinematics has been previously given for 2 and 3 DOF manipulators [5]. The Kinematic Structure of PUMA 560 robot by Unimation was used for the proposed method. Also the results thus obtained were compared with the conventional method of Iterative method alone. The PUMA 560 robot consists of '6' revolute joints. The first three joints, that correspond to the Waist, Sholder and Elbow motion, contribute mainly in positioning of the endeffctor. While the last three joints correspond to the Wrist motion and contribute mainly in orienting the endeffector. This manipulator was chosen for ease of generation of data and verification of results since it has a closed form solution. It has eight solution for a given positon and orientation signified by Right/Left - Shoulder, Above/Below - Elbow, and Up/Down - Wrist. The training data corresponded to: LEFT Arm, BELOW Elbow and UP Wrist configuration. The PUMA 560 parameters were taken from reference [2], page 37. However, in the simulations the joint limits used for the 6th joint were -180° to 180° instead of -266° to 266°.

As mentioned in the previous section that there may be more than one disjoint region in which the solution for a particular configuration lies. In the case of PUMA 560 with the above considered configuration the values for joint number 6 lie in two disjoint regions. One corresponds to ($-\pi/2$ to $-\pi$) and the other corresponds to ($\pi/2$ to π). These need to be trained separately. But in our case we were able to combine the two by transforming these two regions by π and $-\pi$ respectively so that now they lie in the 1st and the 4th quadrant instead of 3rd and 2nd quadrants and then trained them with a single network.

The network in this case consisted of 6 input nodes one output node each for the 6 joints and two hidden layers for each joint consisting of 32 nodes in the first layer and 8 nodes in the second layer. The standard Back Error Propagation algorithm was used with learning rate of 0.1 and the momentum term as 0.2. The training was done for 138 persentation of the data set which consisted of 800 samples. The average error and the standard deviation of the error in the solution given by the MFN for each joint taken over 100 samples is given in table 1. From table 1 it is seen that the solution given by MFNs is more scattered for Joints 4 to 6 as compared with the joints 1 to 3.

Table 1: Statistics of the solution given by The MFN.

Joint #	Error in Degrees	
	Average	Standard Deviation
1	4.06	13.428
2	-0.16	30.492
3	5.64	11.52
4	3.66	61.236
5	-3.70	24.912
6	-6.02	36.828

For the purpose of comparison as to how good is the initial guess provided by the MFN, we presented the initial guess as the actual solution uniformly perturbed randomly between a fraction of joint limits to the iterative method alone for 100 samples corresponding to each fraction. The average number of iterations for the 100 samples presented for each fraction of perturbation of this Informed Newton Raphson Method (INRM), indicated by INRM, is plotted in Fig. 4. For these same samples, requiring the positioning and orientating of the endeffector, the guess given by the MFN was also presented to the iterative method as the initial guess. The corresponding average of the 100 samples for the solution to the initial guess provided by the MFN is also included in this Figure which is indicated by MFNNRM. From Fig. 4 it is seen that the average number of iterations for MFNNRM correspond to less that 20% perturbation with respect to the joint limits about the actual solution for INRM. In the simulations Eq. (5) was solved by Gauss Elimination method and partial pivoting. The maximum number of iterations allowed for the iterative method were 100. The iterative method was successfully teminated when the norm of the difference between the desired and actual endeffector position and orientation was less than 1.0E-4.

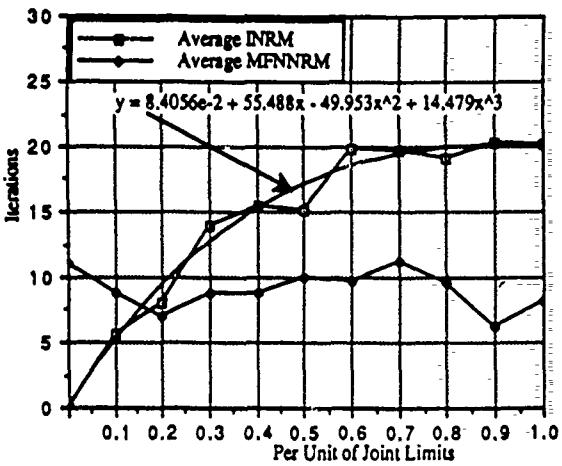


Fig. 4: Comparison of the proposed method with informed Newton Raphson Method.

Lastly, the proposed method was compared by giving a fixed estimate to the iterative procedure. This Fixed Estimate was taken as: $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = \pi/4$, $\theta_4 = 0$, $\theta_5 = \pi/4$, and $\theta_6 = 0$, which is a configuration corresponding to which the MFN was trained, as indicated in the begining of this section. The average and standard deviation for the number of iterations for the proposed method and the Fixed Estimator, in a run of 100 data points is given in Table 2. From this table we can see that the proposed method achieves more than a two fold efficiency in computing with more consistency. Moreover, it was observed that the time taken by the MFN equals two time units of the iterative procedure, which amounts to less than 10% of the time required to get the solution by the Fixed Estimator method.

Table 2

Method	# of Iterations	
	Average	Standard Deviation
Proposed	9.96	12.95
Newton Raphson	21.09	18.90

5. Discussion

A new approach to the iterative solution of robotic manipulators has been presented. This approach makes use of the trained MFN to provide the initial guess to the iterative procedure. It has been shown through examples with the PUMA 560 robot that the proposed method achieves at least two fold computational efficiency, where the time taken by the MFN to provide the initial estimate is only a fraction of the total time to reach a solution. Future efforts will be directed toward achieving better estimates by the MFN and to the learning of the solution to the IKP of redundant robots.

References

[1] W. Mel Bartlett, "MURPHY: A Robot that Learns by Doing," Report No. UIUCDCS-R-88-1397, Dept. of CS, Univ. of Illinois at Urbana-Champaign, Jan. 1988

[2] K.S. Fu, R.C. Gonzalez, and C.S.G. Lee, Robotics Control, Sensing, Vision, and intelligence, New York, NY : McGraw-Hill Book Company, 1987.

[3] C. George, "Calculating Robot-Joint Coordinates From Image Coordinates," NASA Technical Briefs, MFS-27194, Marshall Space Flight Center.

[4] Andrew A. Goldenberg, B. Benhabib, Robert G.Fenton, "A Complete Generalized Solution to the Inverse Kinematics of Robots," IEEE Journal of Robotics and Automation, Vol. RA-1, No. 1, pp 14-20, March 1985.

[5] Allon Guez, and Ziauddin Ahmad, " Solution to the Inverse Kinematics Problem in Robotics by Neural Networks," IEEE International Conference on Neural Networks, Vol-II, 617-624, July 1988.

[6] Michael I. Jordan, "Supervised learning and systems with excess degrees of freedom," COINS Technical Report 88-27, MIT. May 1988.

[7] Alan Lapedes, and Robert Farber, "Nonlinear Signal Processing Using Neural Networks Prediction and System Modelling," Los Almos National Laboratory, July 1987.

[8] Rachid Manseur, Keith L. Doty, "A Fast Algorithm for Inverse Kinematic Analysis of Robot Manipulators," The International Journal of Robotics Research, Vol. 7 No. 3, pp:52-63, June 1988.

[9] Richard P. Paul, Bruce Shimano, Gordon E. Mayer, "Kinematic Control Equations for Simple Manipulators," IEEE Transactions on System, Man, and Cybernetics, Vol. SMC-11, No.6, pp:449-460, June 1988.

[10] Richard P. Paul, Robot Manipulators: Mathematics, Programming, and Control.Cambridge, MA : MIT Press, 1981.

[11] David E. Rumelhart, James L. McClelland, and the PDP Research Group, Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Vol. 1 and 2, Cambridge, MA : MIT Press, 1986.

[12] W. Scott, and B.A. Huberman, "An Improved Three-Layer, Back Propagation Algorithm", IEEE First International Conference on Neural Networks, Vol-II, 637-643, June 1987.

[13] Tzila Shamir and Yosef Yomdin, "Repeatability of Redundant Manipulators: Mathematical Solution of the Problem," IEEE Transactions on Automatic Control, Vol. 33, No. 11, pp:1004-1009, Nov. 1988.

On Learning in Neurocontrollers

A. Guez, Z. Ahmad
ECE Dept. Derexel University

Exploratory schedules (ES) are reference input trajectories designed to enhance the learning of system parameters. Such trajectories in general may not be the desired trajectories, resulting in larger tracking errors. However, ES offer faster convergence to the system parameters and therefore yield smaller long term tracking errors. The automation for the design of ES requires online modification of the desired trajectory to enhance learning at the expense of poorer initial tracking. We discuss this closed loop mode of generation of ES, and give an example of the benefits achieved by the utilization of ES in the context of controller design.

[28]

On Learning in Neurocontrollers

A. Guez, Z. Ahmad
ECE Dept. Derexel University

Exploratory schedules (ES) are reference input trajectories designed to enhance the learning of system parameters. Such trajectories in general may not be the desired trajectories, resulting in larger tracking errors. However, ES offer faster convergence to the system parameters and therefore yield smaller long term tracking errors. The automation for the design of ES requires online modification of the desired trajectory to enhance learning at the expense of poorer initial tracking. We discuss this closed loop mode of generation of ES, and give an example of the benefits achieved by the utilization of ES in the context of controller design.

[29]

On Generation of Exploratory Schedules in Closed Loop for Enhanced Machine Learning

A. Guez, Z. Ahmad
ECE Dept. Drexel University

Exploratory schedures (ES) are reference input trajectories designed to enhance the learning of system parameters. Such trajectories in general may not be the desired trajectories, resulting in larger tracking errors. However, ES offer faster convergence to the system parameters and therefore yield smaller long term tracking errors. The automation for the design of ES requires online modification of the desired trajectory to enhance learning at the expense of poorer initial tracking. We discuss this closed loop mode of generation of ES, and give an example of the benefits achieved by the utilization of ES in the context of controller design.

[30]

HYBRID EXPERT-NEURO CONTROLLER FOR ROBOTS

I. Bar-Kana, A. Guez, and I.Rusnak
ECE Dept., Drexel University, Philadelphia, PA 19104.

ABSTRACT

A hybrid neural network and expert system based controller is designed for the control of rigid robot manipulators. The composition of neural net and expert system technologies is used to combine the strength of both. The neural network based module deals well with continuous adaptation to small and *continuous* parameter perturbations in the robot dynamics [1], and with improved parameter identification. The expert system is capable of handling a low-order, *discontinuous* and complex set of model and environment parameter changes via a set of prestored rules realizing informed sequential decision making expertise.

The combination of these modules has the potential of improving the performance of adaptive robot controllers under complex task requirements in changing environments, when various objectives, such as performance, stability, identification, error reduction, noise robustness, task variation, etc., become dominant. The final product is an Expert System that coordinates the different objectives of the tasks of a robot during its life cycle.

[1] A. Guez and I. Bar-Kana, "Two degree-of-freedom Robot Neurocontroller," Conf. on Decision and Control, Honolulu, Hawaii, 1990.

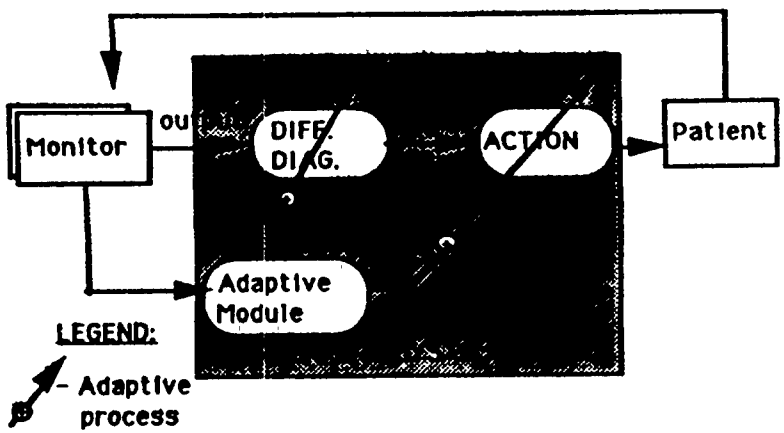
[2] J. W. Selinsky and A. Guez, "The Role of Apriori Knowledge of Plant Dynamics in Neurocontroller Design," Conf. on Decision and Control, Tampa, Florida, 1989, pp. 1754-1758.

Title: Mathematical Model for An Adaptive Expert System in Anesthesia

Authors: I. Nevo MD, A. Guez Ph.D*, F. Ahmed B.S.E.E*, J.V. Roth MD

Affiliation: Dept. of Anesthesiology, Albert Einstein Medical Center, Philadelphia, PA-19141; *Dept. of Electrical and Computer Engineering, Drexel University, Philadelphia, PA-19104.

A mathematical model for an adaptive expert system in anesthesia has been developed and is presented in the following block diagram:



The new architecture encompasses the following distinctive features:

A. Adaptation: The expert system is capable of learning and adjusting its diagnostic capabilities based upon its cumulative experience.

B. Low resolution high abstraction knowledge base representation: Mathematically, the patient's condition can ideally be defined in time and space only by having a very large number of measurements and attributes. In clinical situations, only a few determinants are available[1,2]. The new concept of 'clusters' which have attributes (e.g. hypoxia, tachycardia etc.) are subsets of 'diagnostic categories'[3]. This allows a high level description of the patient's state and minimizes the computational burden of state-space search by the the expert system algorithm. The use of clusters permit more subtle definitions of clinical categories.

C. Flexible data structure allowing bi-directional detection flow: In the clinical setting the patient's state is first estimated, then a treatment and/or action is selected. In contrast, our system enables the matching of appropriate treatment to various clinical categories, and the mapping of the patients' states to a finite repertoire of treatments. The implication is that future clinical situations can be anticipated and better treated. The finite set of categories currently used in medical decision making may not be optimal. One of our goals is not only to utilize the existing categories, but also to find out whether new and more efficient categories can be defined.

Bibliography:

1. Chen, C.T. Linear System Theory and Design. New York:HRW.
2. Sijlak, D.D. Non Linear Systems New York: Wiley
3. Duda, R.O. & Hart, P.E. Pattern Classification and Scene Analysis. New York: Wiley

VITAL FUNCTION STATUS - A Parameter To Facilitate Decision Making In Anesthesia

I. Nevo, MD, J. V. Roth, MD, F. Ahmed*, A. Guez, Ph.D*
Dept. of Anesthesiology, Albert Einstein Medical Center, Philadelphia, PA;
* Dept. of E.C.E, Drexel University, Philadelphia, PA.

The anesthesia environment is a dynamic environment where the anesthesiologist has to make, with a degree of uncertainty, prompt decision in order to prevent disastrous outcomes to the patient. The anesthesia environment often consists of 7 to 8 monitors displaying multiple numbers and waveforms in different formats at different locations. Anesthesia accidents are typically caused by more than one factor including human errors, equipment failure, surgical events, or alteration in physiology caused by patient's disease. Whereas equipment failure is still a component of accidents, the greatest number of anesthetic mishaps can be attributed to human error (1,2). There is a need for simplification and improvement of the information made available to the anesthesiologist.

We propose a new parameter, the Vital Function Status (VFS), as part of our overall project to develop an expert system in anesthesia. The VFS is a semi-quantitative value which summarizes the patient's physiologic condition and depicts at what level of danger the patient is in.

Exploiting recent developments in the fields of anesthesia, perception and cognition, expert systems, artificial intelligence, distributed sensing, adaptive signal processing, machine learning, and man machine interface technologies, a prototype has been partially developed. An IBM AT-286 with an 80287 coprocessor samples, from physiologic monitors, analog signals at rates up to 500 Hz and digital signals every second. Turbo C is the development environment. The system contains two major subunits, the Vital Function Unit (VFU) and the Inference Unit (IU). The physiologic monitors are interfaced into the VFU. The values are validated, the signals are filtered, and derivative parameters are computed. Any derivative parameter (e.g. lung compliance, systemic and pulmonary vascular resistance, left and right-ventricular stroke work index, oxygen delivery, arterial-venous oxygen difference) for which there is sufficient information is calculated every second. Each value is compared to reference values residing in the Reference Value Unit (RVU) and assigned one of six levels of danger ranging from 0 (no danger) to 5 (immediate critical danger). All of the assigned values are grouped by threshold into one parameter VFS having a value from 0 to 5.

On the screen two graphs are displayed. The VFS evolved within the last three minutes (actual VFS) and the trend during the last 30 minutes (trend VFS) with scroll back option. Once a change is detected, the system displays the parameter that first deviated. The the causal event can than be more easily identified.

We believe that more frequent analysis of monitoring data and earlier detection will facilitate decision making in anesthesia. It may reduce the anesthesiologist's response time which may prevent further deterioration in the patient's condition.

Bibliography:

1. Cooper, J. B., Gaba, D. M. (1989). A strategy for preventing anesthesia accidents. Int Anesthesiol Clin, 27, 148-52.
2. Gaba, D. M., Maxwell, M., DeAnda, B. S. (1987). Anesthetic mishaps: Breathing the chain of accident evolution. Anesthesiology, 66, 670-6.

VITAL FUNCTION STATUS - A Comprehensive Display To Enhance Decision Making in Anesthesia and ICU

[33]

I. Nevo, MD, J. V. Roth, MD, F. Ahmed*, A. Guez, Ph.D.*
Dept. of Anesthesiology, Albert Einstein Medical Center, Philadelphia, PA; *Dept. of E.C.E, Drexel University, Philadelphia, PA.

It is estimated that between 2,000 and 10,000 patients die in the USA each year from causes at least partially related to anesthesia. The morbidity rates related to anesthesia are higher. Whereas equipment failure is still a component of accidents, the greatest number of anesthetic mishaps can be attributed to human error. The anesthesia environment is a dynamic environment where the anesthesiologist has to make, with a degree of uncertainty, prompt decisions in order to prevent disastrous outcomes to the patient. The anesthesia environment often consists of 7 to 8 monitors displaying multiple numbers and waveforms in different formats at different locations. The human brain is capable of perceiving concomitantly seven informational inputs and of processing efficiently only 2 or 3 at the same time. Fatigue, inexperience and stress further decrease efficiency, may cause one to fixate on a specific idea, and increase the probability of human error. There is a need for the simplification and improvement of the information made available to the anesthesiologist.

We propose a new parameter, the Vital Function Status (VFS), as part of our overall project to develop an expert system in anesthesia. The VFS is a semi-quantitative value which summarizes the patient's physiologic condition and depicts at what level of danger the patient is in.

Exploiting recent developments in the fields of anesthesia, perception and cognition, expert systems, artificial intelligence, distributed sensing, data acquisition, adaptive signal processing, machine learning, and man-machine interface technologies, a prototype has been partially developed. An IBM AT-286 with an 80287 co-processor samples from the physiologic monitors analog signals at 500 Hz and digital signals every second. Turbo C is the development environment. The system contains two major subunits, the Vital Function Unit (VFU) and the Inference Unit (IU). The physiologic monitors are interfaced into the VFU. The values are validated, the signals are filtered, and derivative parameters for which there is sufficient information are calculated every second. Each value is compared to reference values residing in the Reference Value Unit (RVU) and assigned one of six levels of danger ranging from 0 (no danger) to 5 (immediate critical danger). All of the assigned values are grouped by threshold into one parameter, VFS, having a value from 0 to 5.

On a screen, two graphs are displayed: the VFS evolved within the last three minutes (actual VFS) and the trend over the last 30 minutes (trend VFS). Once a change is detected, the system displays the parameter that first deviated. This allows for the causal event to be more easily identified. This will assist the anesthesiologist in responding to the causal effector and not just to an event in a cascade.

We believe that more frequent analysis of monitoring data and earlier detection of deviations will facilitate decision making in anesthesia. It may reduce the anesthesiologist's response time which may prevent further deterioration in the patient's condition.

11/90 Phila

[34]

ANESTHESIOLOGIST'S ADAPTIVE ASSOCIATE (ANAA)

F. Ahmed*, I. Nevo+, A. Guez*

*Electrical and Computer Engineering Department
Drexel University, Philadelphia, PA 19104

+Anesthesiology Department
Albert Einstein Medical Center, Philadelphia, PA 19141

ABSTRACT

The anesthesia environment is a dynamic and stochastic system where an overwhelming amount of data is constantly displayed. Nevertheless, prompt decisions ought to be made and subsequent treament should be applied. We propose the ANAA (ANesthesiologist's Adaptive Associate) which is an adaptive real time expert system.

INTRODUCTION

The anesthesia environment is a dynamic and stochastic system in which the anesthesiologist has to make decisions promptly to prevent disastrous outcomes [1]. Throughout the surgical procedure, the anesthesiologist has to maintain the patients vital functions stable and within normal ranges, and to prevent a change in any vital function which usually propels a deterioration of other vital functions.

The human brain is capable of perceiving on the order of seven informational inputs concomitantly and processing effectively only 2 or 3 at one time [2]. Moreover, sharing the attention reduces the efficacy of such performance [3]. The anesthesia workstation often consists of seven to eight monitors each one displaying multiple numbers and waveforms in different formats. The physicians task is further complicated by the scattered information and the responsibility of watching the patient as well as the surgical field. As a result, too often times the physician performing his duties reacts, remotely in time, not to a cause of an initial problem but rather to its consequence.

To enhance the anesthesiologist performance and to improve the patient's outcome we are developing ANesthesiologist's Adaptive Associate (ANAA), an adaptive knowledge based, real time expert system (See figure below), that will work in association with the anesthesiologist.

ANAA DESCRIPTION

The system is described in the figure below. It consists of monitors, data acquisition hardware, signal interfaces, PC, expert system software, feedback loops for knowledge base refinement and an elaborate display. The hardware consists of IBM-PC AT (w/ Intel 80286) interfaced with seven monitors through analog I/O board (Analog Devices) and three RS232 serial communication ports. The expert system consists of two units: Patient Status Unit (PSU) and Inference Unit (IU).

ANAA OPERATION

Patient Status Unit (PSU):
The system contains a set of Reference Values (RV). These consists of normal ranges (matched for the patient's age group) and five different levels of danger for each vital sign (from none to severe danger) [1]. Prior to the begining of the anesthesia the PSU incorporates into the RV the patient's basal vital signs values. During the anesthesia new data, as well as computed values (e.g. compliance, resistance etc.), are compared to the RV and to previous values. Subsequently, individual danger level for each vital sign is computed; all the danger levels are condensed to give a **single display parameter**. The displayed new parameter is a real time patient's Vital Function Status (VFS).

Inference Unit (IU):
This unit contains the anesthesiologist's expertise.The danger levels of each vital sign will be sent to this unit. Once a deviation from normal is detected IU will display the following: a) the first parameter that changed, b) a list of differential diagnosis (ranked according to their life threat potential, and c) a list of proposed actions to correct the deviation.

The IU will be adaptive in nature, i.e., it will suggest course of actions drawn from a knowledge base, which can be updated and refined through a learning loop based on real time performance criteria. The IU will compare its suggested actions to the action taken by the anesthesiologist. If they are different and the anesthesiologist's action brings the patient to a more desirable state, then the knowledge base gets updated.

This system will not use any probabilistic or statistical methodology, rather it will be based on correlation analyses. It will reduce the informational overload from the anesthesiologist, by providing a composite and concise display of information on one screen. It will detect the first change that took place and the VFS will provide an holistic information regarding the patient situation. By suggesting differential diagnosis and actions, it will give the physician a wider choice of options, which he, otherwise, may overlook. The adaptive learning loop will make it more flexible. Overall, it will enhance the quality assurance and will improve patient outcome.

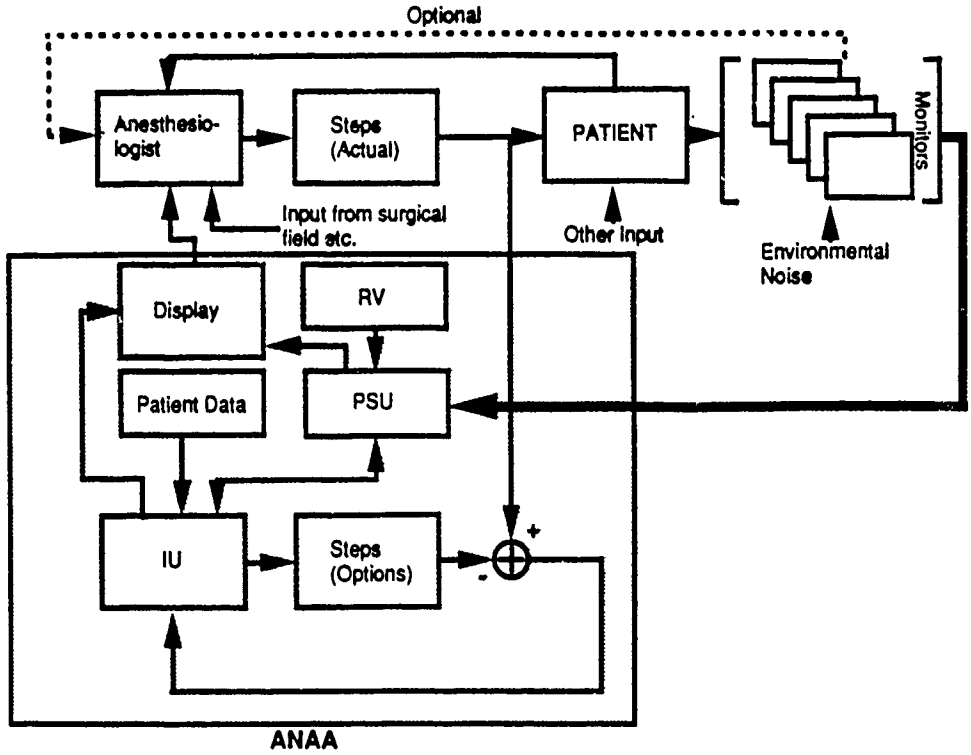


Figure: ANAA's Architecture

REFERENCES

[1] L.E. Widman, "Expert System Reasoning About Dynamic Systems by Semi-quantitative simulation", Comp. Meth. Prog. Biomed. Vol. 29(2), pp. 95-113: 1989.

[2] G.A. Miller, "The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information", Psych. Review Vol. 63(2), pp. 81-97.

[3] G. Sperling, B.A. Dosher, in: "Handbook of Perception and Human Performance", eds. K.R. Boff, L. Kaufman and J.P. Thomas, Vol., Chap. 2, pp. 4-17 (John Wiley & Sons,New York, 1986).

INT. J. CONTROL, 1990, VOL. 52, NO. 1, 77–99

[36]

Simple adaptive control for a class of non-linear systems with application to robotics

IZHAK BAR-KANA† and ALLON GUEZ†

Simple adaptive control techniques have recently been developed for continuous stationary and non-stationary linear systems, based on the fact that the output stabilizability property of systems can be used to realize 'almost passive' configurations that may facilitate the implementation of simple robust adaptive controllers. This paper extends these concepts to a class of non-linear systems that are linear in the control; this includes manipulators and similar systems. Since it has recently been shown that non-linear systems that can be stabilized via static or dynamic output feedback become 'almost passive' via some corresponding parallel feedforward, a rigorous proof of the robust stability of the simple adaptive control of non-stationary almost passive systems is presented.

1. Introduction

Most adaptive control techniques (Landau 1979, Åström 1983) assume prior knowledge of the order (or an upper bound on the order) of the unknown controlled plant. Prior knowledge of the pole-excess in the plant is also needed. This prior knowledge is used to implement observer-based controllers of the same order as the plant. Since these assumptions may not be satisfied in realistic large systems, a simplified adaptive control approach was developed by Sobel *et al.* (1979, 1982) and Bar-Kana and Kaufman (1982 a, b). The developers of this approach were influenced by the progress of adaptive control methods in the late 70s, especially the model reference and stability analysis approach of Landau (1979), Monopoli (1974), and Narendra and Valavani (1979). However, they were also interested in the simple adaptive control methods of the 60s (Whitaker 1959, Osburn *et al.* 1961, Donaldson and Leondes 1963, Parks 1966). Since from the very beginning they had to deal with real-world large scale systems, Kaufman *et al.* (1981)—the developers of this approach—were forced to consider real world constraints, which had recently been considered realistically in adaptive and general control design under the topic 'robustness', as follows.

(*a*) The order of real world plants is large and generally unknown, and the models available for the control design are only low-order approximations of the real plant. Even if the order is known, it is normally too large to allow using controllers of the same (or a larger) order as the plant

(*b*) The same applies for the pole-excess, or, in the general multivariable case, for the MacMillan degree of the plant.

(*c*) Although the algebraic conditions for the minimum order of stabilizing controllers are an outstanding unsolved problem, most real-world systems can easily be stabilized with simple controllers. Many plants are even stable at the

Received 21 February 1989.

† Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104, U.S.A.

0020-7179/90 $3.00

start. However, stability is the first and necessary condition but is not sufficient; the control designer is interested in performance.

The question that arises from the constraints and assumptions above is: can the basic stabilizability of the controlled plants, combined with specific non-linear adaptive control techniques, fit the right gains to the right operational situation so that performance is obtained while stability is guaranteed? The answer is not necessarily positive when non-stationary or non-linear controllers are used. Yet, starting with modest results (Sobel *et al.* 1982), which only presented the first conditions needed for the stable behaviour of a basic simplified adaptive control algorithm, this approach was developed and extended to result in a simple, robust, and well-performing algorithm (Bar-Kana 1989 d) whose implementation is feasible for most real-world applications.

The first motivation and representative examples of this approach are large flexible structures. Whether we treat them as finite systems or as infinite dimensional sytems, their order is very large and unknown. Still, they may be stabilizable via low-order controllers. Fixed low-order controllers may not be able, however, to guarantee the desired performance, and this job we leave for non-linear adaptive controllers. Although one can find examples of plants that cannot be stabilized by controllers of lesser order than the plant, these are not common where processes, planes, missiles, manipulators, etc. are concerned, rather than the 'general model'. Root-locus or Bode analysis shows when the information on the plant permits such an analysis. High order of a plant and 'unmodelled dynamics' do not necessarily justify problems that some adaptive control algorithms are expected to solve in the well-known examples due to Rohrs *et al.* (1982, 1985) which are both stable and minimum-phase, and could have been controlled even by a controller of zero order (constant).

Recently, it has been shown for stationary and non-stationary multivariable linear systems (Bar-Kana and Kaufman 1985 a, Bar-Kana 1986 a, 1987 a, 1988 e, 1989 b) that stable, simple adaptive controllers (SACs) can be implemented using some vague prior knowledge or assumptions about the basic stabilizability properties of the plant to be controlled. In fact, the simple result states that if a plant can be stabilized by some simple configuration, the adaptive controller can adaptively compute the desired controller gains without using any real initial knowledge. The stability of a non-linear adaptive control system is guaranteed if the inverse of some stabilizing configuration is used as parallel feedforward to augment the plant to be controlled (Bar-Kana 1987 a, b, 1989 c). The (multivariable) augmented controlled system is now called 'almost strictly passive' (ASP), because it has been shown that there exists some positive definite static (unknown) output feedback, such that the fictitious resulting closed-loop system is strictly passive (also strictly positive real, in linear time-invariant systems), and strict passivity is characterized by some relations which guarantee the robust stability of non-linear controllers.

In other words, the passivity (or positive realness in stationary linear systems) conditions required to guarantee the stability of a non-linear adaptive control system are in fact a different and equivalent formulation of the basic stabilizability properties of the plant to be controlled. The result is a simple and robust direct model reference adaptive control (MRAC) algorithm, and its applications now include non-minimum-phase and unstable, non-stationary or non-linear examples of missiles, flexible structures, motors, helicopters, manipulators, and other servo systems. In fact, the applications people are now trying to collect the most difficult examples that can be found. The simplicity of implementation may make it attractive for engineers with practical

applications on their mind, although theoretically one can ask: 'How do you know that the system is stabilizable?' This question is not so serious when one tries to control a process, plane or manipulator, rather than the general control model. The adaptive control designer must take into consideration that most control designs and even robust control designs are not adaptive. The problem of the control designer is the performance, rather than the stability, of the control system. However, without guaranteeing stability, performance cannot even be discussed. When called to select the fixed gains of the controller, the designer must trade-off his various choices. High gains imply high manoeuvrability, but they also imply high cost of control, high noise amplification and a 'nervous' response even if these are not necessarily needed. Low-gains imply smooth response and low control cost, but they do not provide the needed manoeuvrability when it is required. 'Optimal' gains may sometimes provide a suitable compromise, but again may not be satisfactory for any of these cases. Changing the gains 'adaptively', or in accordance with the specific situation in oder to maintain performance in various operational environments is an attractive idea. However, when we use non-stationary control, stability is not necessarily guaranteed in general, even if the corresponding linear system was guaranteed stable. In the past it has been shown for linear systems, that SAC guarantees stability with non-linear adaptive controllers over the entire region that could have been used by fixed gains one at a time. This paper extends this result for a class of non-linear systems that are linear in the control. While it does not attempt to present a general solution to the general problem, this approach may still offer a simple and robust adaptive solution for many difficult realistic problems, and the completion of preliminary linear designs based on only vague knowledge about the plant to be controlled.

If a controlled system is ASP, then simple adaptive control procedures find the necessary gains that can stabilize and control the system without having to know or use this fictitious static feedback or any prior knowledge about the plant. This property can be very useful if one attempts to present a solution when the order of the plant is very large and unknown, because in this case very low-order adaptive controllers may control even large flexible structures that are passive under idealistic assumptions, or can be made passive in realistic environments (Bar-Kana *et al.* 1983, Bar-Kana and Kaufman 1984, Balas *et al.* 1984, Ih *et al.* 1987).

While a reader with the 'general problem' in mind may still wonder about the basic assumption of the prior availability of some stabilizing configuration, this result is useful in particular control designs, when some prior knowledge on the plant to be controlled is sufficient to stabilize the plant, even though not necessarily sufficient to obtain the desired performance. (It is usually enough to know that the plant is a motor, a manipulator or a flexible structure.) In many cases, proper controllers may not necessarily be desired, like the PD controllers with the general (matrix) transfer function $H(s) = K(1 + s/s_0)$ (Kidd 1985, Slotine and Li 1988, Bar-Kana *et al.* 1989). However, while Kidd (1985) does not use this desired controller because of the need for derivatives, and while Slotine and Li (1988) try to cope with a noisy tachometer, we only exploit the assumed existence of the *fictitious* improper controller and actually use its proper inverse in parallel with the plant. This way, although we only use position measurement sensors, we still get the effect of the desired differentiators without really differentiating (Bar-Kana 1987 a, 1988 a, d). Since it was proven that global stability and robustness of the non-linear adaptive controller operating on the augmented plant is guaranteed, the adaptive controller is now called to fit the right gains to the right situation in order to minimize the tracking errors (Bar-Kana 1987 b).

An objective evaluation of this approach as compared with others, although still based on its first results, can be found in the paper of Mattern and Shoureshi (1987). Meanwhile, new papers and reports may better explain some aspects that were not clear enough in the past. Along with new developments in the theory and applications, attempts are made to write some explanation that would intuitively give the motivation behind our approach. Section 4 of Bar-Kana (1987 a) may be the best introduction to this approach. It is mainly concerned with SISO systems. One may also note the non-linear manipulator application of that paper. A new tutorial (Bar-Kana 1989 d) which may be the most complete presentation of the theory and applications so far, gives the necessary explanation and motivation for the various steps and parts of SAC in multivariable systems, using the minimal amount of mathematics possible.

A good challenge to this simple adaptive control algorithm in difficult practical applications is offered by the adaptive control of robots, missiles, or planes. Missiles (Guez 1980), manipulators (Guez and Dritsas 1987), and planes (Morse and Ossman 1989), like flexible structures, are in fact stable, or easily stabilizable, configurations. However, they are difficult control objects because stability does not necessarily imply performance. In various operational environments the control parameters (angles, loads, inputs, disturbances) change quite a bit and fixed controllers may have a difficult task trying to maintain good performance. Here, SAC may provide a solution: the necessary gains are computed automatically such that the performance (small tracking errors) is maintained because the adaptive controller may use the entire domain of admissible gains, the right gain at the right time. We mention, in particular, flight control reconfiguration after multiple control-surface failure (Morse and Ossman). The adaptive controllers move rapidly from some value of the adaptive gains to a new value more fitting to the new situation after failure, thus maintaining a stable flight instead of what could have been total catastrophe. This approach is therefore quite different from other mainstream adaptive control methodologies that try to use adaptation in order to reach some fixed 'optimal' controller. SAC may be considered a first, modest, or even primitive example of intelligent control, because it only uses a single control configuration, yet attempts to fit the right controller gains to the right environmental and operational condition in order to maintain the required performance (represented by small tracking errors).

However, although SAC has been successfully applied in non-stationary and non-linear systems, proofs of stability only cover linear systems. The state-space representation of the systems and the Lyapunov stability analysis of the adaptive algorithms allowed in the past for a unified presentation of SISO and multivariable systems (Bar-Kana 1987 b, 1989 c), the the first attempts at non-linear systems show that the techniques can be rigorously extended to the classes of non-linear systems that include, for example, robots and missiles.

This paper extends these concepts to non-linear multivariable systems. The robust model reference adaptive control algorithm for almost passive systems is presented in § 2, and is shown to be based on the basic sabilizability of the plants. A robot manipulator application is then presented in § 3, and some conclusions in § 4.

2. Preliminaries and basic definitions

Let us consider a dynamic non-linear plant given in the following state-space representation

$$\dot{x}_p(t) = A_p(x_p)x_p(t) + B_p(x_p)u_p(t) \tag{1}$$

$$y_a(t) = C_p(x_p)x_p(t) + D_p(x_p)u_p(t) \tag{2}$$

where $x_p(t) \in \mathbb{R}^n$, $y_a(t) \in \mathbb{R}^m$, $u_p(t) \in \mathbb{R}^m$, and where the matrices $A_p(x_p)$, $B_p(x_p)$, $C_p(x_p)$, $D_p(x_p)$ are uniformly bounded. We use the not necessarily strictly causal form $\{A_p(x_p), B_p(x_p), C_p(x_p), D_p(x_p)\}$ only for the sake of completion of the treatment, and the following results remain perfectly valid if the systems are strictly causal ($D_p(x_p) = 0$).

Definition 1

The system (1), (2) is said to be *almost strictly passive* (ASP) if there exists a positive definite static feedback matrix K_e such that the resulting closed-loop system is *strictly passive* (SP) (Fig. 1).

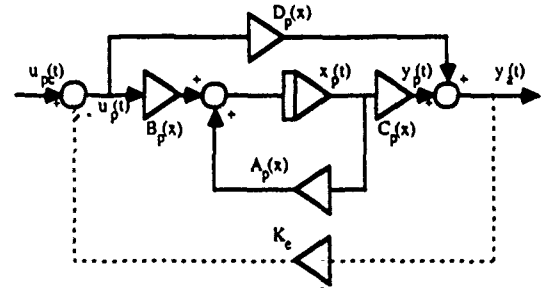


Figure 1. First strictly passive configuration.

It is easy to see that by using the controller

$$u_p(t) = -K_e y_a(t) + u_{pc}(t) \tag{3}$$

we get the following closed-loop system (Fig. 2)

$$\dot{x}_p(t) = A_{pc}(x_p)x_p(t) + B_{pc}(x_p)u_{pc}(t) \tag{4}$$

$$y_a(t) = C_{pc}(x_p)x_p(t) + D_{pc}(x_p)u_{pc}(t) \tag{5}$$

where

$$A_{pc}(x_p) = A_p(x_p) - B_p(x_p)K_{ec}(x_p)C_p(x_p) \tag{6}$$

$$K_{ec}(x_p) = K_e[I + D_p(x_p)K_e]^{-1} = [I + K_e D_p(x_p)]^{-1}K_e \tag{7}$$

$$B_{pc}(x_p) = B_p(x_p)[I + K_e D_p(x_p)]^{-1} \tag{8}$$

$$C_{pc}(x_p) = [I + D_p(x_p)K_e]^{-1}C_p(x_p) \tag{9}$$

$$D_{pc}(x_p) = [I + D_p(x_p)K_e]^{-1}D_p(x_p) = D_p(x_p)[I + K_e D_p(x_p)]^{-1} \tag{10}$$

where $D_p(x_p) > 0$ and $D_{pc}(x_p) \geqslant 0$, which means that $D_p(x_p)$ and $D_{pc}(x_p)$ are non-singular and positive definite for all $x_p \in \mathbb{R}^n$.

The strictly passive system (4), (5) satisfies the following relations (Willems 1972):

$$\dot{P}(x_p) + P(x_p)A_{pc}(x_p) + A_{pc}^{\mathrm{T}}(x_p)P(x_p) = -Q(x_p) - L^{\mathrm{T}}(x_p)L(x_p) \tag{11}$$

$$P(x_p)B_{pc}(x_p) = C_{pc}^{\mathrm{T}}(x_p) - L^{\mathrm{T}}(x_p)W(x_p) \tag{12}$$

$$D_{pc}(x_p) + D_{pc}^{\mathrm{T}}(x_p) = W^{\mathrm{T}}(x_p)W(x_p) \tag{13}$$

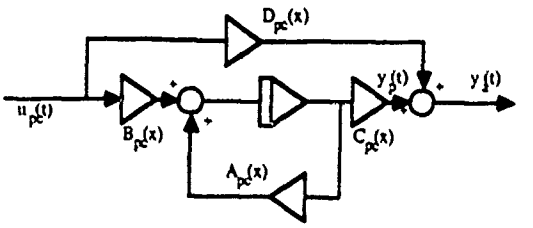


Figure 2. Second equivalent strictly passive system.

for some uniformly bounded positive definite matrices, $P(x_p)$ and $Q(x_p)$, and some matrices $L(x_p) \in \mathbb{R}^{m \times n}$ and $W(x_p) \in \mathbb{R}^{m \times m}$, and where

$$\dot{P}(x_p) = \frac{dP(x_p)}{dt} = \frac{\partial P(x_p)}{\partial x_p} \frac{dx_p}{dt}$$

It is easy to see (Willems 1972) that relations (11)–(13) are equivalent to the following Riccati equation:

$$\dot{P}(x_p) + P(x_p) A_{pc}(x_p) + A_{pc}^{\mathrm{T}}(x_p) P(x_p) + [P(x_p) B_{pc}(x_p) - C_{pc}^{\mathrm{T}}(x_p)] \times [D_{pc}(x_p) + D_{pc}^{\mathrm{T}}(x_p)]^{-1} [B_{pc}^{\mathrm{T}}(x_p) P(x_p) - C_{pc}(x_p)] + Q(x_p) = 0 \quad (14)$$

In strictly causal systems, when $D_p(x_p) = 0$, the strict passivity relations are:

$$\dot{P}(x_p) + P(x_p)[A_p(x_p) - B_p(x_p) K_e C_p(x_p)] + [A_p(x_p) - B_p(x_p) K_e C_p(x_p)]^{\mathrm{T}} P(x_p) = -Q(x_p) < 0 \quad (15)$$

$$P(x_p) B_p(x_p) = C_p^{\mathrm{T}}(x_p) \quad (16)$$

Relations (15), (16) can be obtained directly from (14) which, when $D_p(x_p)$ is singular, cannot hold unless (16) holds, and then (15) follows immediately. Let us also assume that $B_p(x_p)$ is of maximal rank. In this case, (16) requires that $C_p(x_p)$ also be of maximal rank and that

$$B_p^{\mathrm{T}}(x_p) C_p^{\mathrm{T}}(x_p) = B_p^{\mathrm{T}}(x_p) P(x_p) B_p(x_p) > 0 \quad (17)$$

which is the equivalent of the relative conditions of linear time-invariant systems, which requires that strictly causal ASP systems have $n - m$ minimum-phase finite zeros and n arbitrary poles (Shaked 1977, Bar-Kana 1987 a).

The reader may find different definitions of passivity in the literature, but the forms above were selected because they are direct results of the basic stabilizability properties of the plant (Bar-Kana 1989 a), and also because they are very convenient in the subsequent proofs of stability. ASP systems are also high gain stabilizable (Bar-Kana 1989 a), a property that is essential for the robustness of the non-linear adaptive controllers. We base our development on exponential stability as defined by Hahn (1967) with application to our specific structures. We assume that if the plant is stabilizable, the resulting stable configurations is *exponentially stable* as defined below.

Definition 2 (Hahn 1967)

Let the general non-linear system be represented by the n-order vectorial equation $\dot{x}(t) = f(x, t)$ and let $x = 0$ be an equilibrium point. The equilibrium point is called

exponentially stable if all solutions $x(t)$ satisfy the relation $|x(t)| \leqslant \alpha|x(0)| \exp(-\beta t)$ for some scalars $\alpha > 0$ and $\beta > 0$.

Theorem 1 (Hahn 1967)

Let the right-hand side of $\dot{x}(t) = f(x, t)$ have bounded continuous first-order partial derivatives. Let the equilibrium be exponentially stable. Then there exists a Lyapunov function $V(x, t)$ that satisfies estimates of the form

$$(a) \qquad \alpha_1|x(t)|^2 \leqslant V(x, t) \leqslant \alpha_2|x(t)|^2$$

$$(b) \qquad \dot{V}(x, t) \leqslant \alpha_3|x(t)|^2$$

$$(c) \qquad \frac{\partial V(x, t)}{\partial x_i} \leqslant \alpha_4|x(t)|^2, \quad i = 1, 2, \ldots, n$$

Because this paper deals with non-linear systems of the form (1), (2) rather than linear time-invariant systems, we cannot expect to find, or even show the existence of a positive definite quadratic Lyapunov function of the form $V(x_p) = x_p^T(t)Px_p(t)$ where P is constant and positive definite. However, after some experience with specific non-linear systems, such as robots, and because we restrict our discussion to non-linear systems that are linear in the control of the form (1), (2), we *assume* that exponential stability of the *autonomous* system (1), with $u_p(t) \equiv 0$, implies the existence of non-linear Lyapunov functions $V(x)$ which are not explicit functions of time. Furthermore, since we can always write $V(x_p) = x_p^T(t)P(x_p)x_p(t)$, and because then $\dot{V}(x_p) = x_p^T(t)[\dot{P}(x_p) + P(x_p)A(x_p) + A^T(x_p)P(x_p)]x_p(t) = -x_p^T(t)Q(x_p)x_p(t)$, we restrict the subsequent discussion to system that satisfy the following assumption.

Assumption

Exponential stability of the *autonomous* system (1), with $u_p(t) \equiv 0$, implies the existence of Lyapunov functions of the form $V(x_p) = x_p^T(t)P(x_p)x_p(t)$ and a derivative of the form $\dot{V}(x_p) = x_p^T(t)Q(x_p)x_p(t)$, where $P(x_p)$ and $Q(x_p)$ are positive definite for all $x_p \in \mathbb{R}^n$. By using the same Lyapunov function with the non-autonomous equation (1), it is easy to see that exponential stability implies BIBO stability.

Lemma: Almost passivity

Let G_p be any *non-linear* system of the form (15), (16). Let H be some stabilizing linear controller for G_p, and assume that we use the inverse of H in parallel with G_p, so that the augmented system $G_a = G_p + H^{-1}$ has the form $G_a = \{A_p(x_p), B_p(x_p), C_p(x_p), D_p(x_p)\}$. Then G_a is 'almost strictly passive' (ASP). In other words, the augmented system G_a satisfies almost strict passivity relations of the form (11)–(13) (Bar-Kana 1989 a).

The most attractive and direct applications of the almost passivity lemma can be implemented if some raw prior information about the controlled plant is given and some estimate of the maximal stabilizing output feedback, K_{max}, is known. If it can be evaluated, then by using the parallel feedforward $D_p = K_{max}^{-1}$, we obtain an augmented system that can easily and reliably be controlled by almost any reasonable control method. Since instead of controlling the plant output $y_p(t) = C_p(x_p)x_p(t)$ we now control the augmented output $y_a(t) = y_p(t) + D_p u_p(t)$, the lemma is especially useful in systems that maintain stability with high feedback gains. In this case, the

supplementary gain may be very small and may not essentially affect the controlled plant output. For example, in very common cases, if the gain of the plant is ten and the largest admissible gain is evaluated to be about 100, then we only add $1/100 = 0{\cdot}01$ in parallel with the plant. It is amazing how such a small parallel term changes the stability properties of the plant (Bar-Kana and Kaufman 1987).

To end these remarks, we note that K_{max} (or H in general), is indeed required to guarantee the stability of the fictitious closed-loop system, and thus the almost passivity of the augmented open-loop system (15), (16), but only in a very weak sense. The fictitious stable system is by no means required to be good. Thus, stability is only a sufficient condition that enables the subsequent non-linear control to impose the desired behaviour of, for example, a well designed low-order reference model upon the (large) plant.

The proof of stability of the subsequent multivariable adaptive control systems extends the applicability of simple adaptive controllers to non-linear systems, and some references at the end of this paper may give a good illustration about what can be done in stationary, non-stationary, or non-linear systems when appropriate feedforward can be selected *a priori* (Bar-Kana and Kaufman 1983, 1984, 1985 a, b, 1988, Bar-Kana *et al.* 1983, Bar-Kana 1987 a, b) or if it must also be computed adaptively (Bar-Kana 1986 b, 1987 b).

3. Model following problem for non-linear ASP systems

In § 4 we show how to select the necessary parallel configurations in practical design. This section assumes that the adaptive control is applied to the augmented ASP system. In realistic environments we must take into consideration the input and output disturbances, and therefore our augmented ASP controlled plant has the following representation

$$\dot{x}_p(t) = A_p(x_p)x_p(t) + B_p(x_p)u_p(t) + d_i(x_p, t) \tag{18}$$

$$y_a(t) = C_p(x_p)x_p(t) + D_p(x_p)u_p(t) + d_0(x_p, t) \tag{19}$$

where the matrices $A_p(x_p)$, $B_p(x_p)$, $C_p(x_p)$, and $D_p(x_p)$ are uniformly bounded. We assume that $d_i(x_p, t)$ and $d_0(x_p, t)$ are bounded input and output disturbances that can also represent some inaccuracies of the representation of the real plant. The plant is assumed to be ASP; however, it may also be very large and basically unknown. The output of the plant is required to follow the output of the (possibly) very low-order model

$$\dot{x}_m(t) = A_m(x_m)x_m(t) + B_m(x_m)u_m(t) \tag{20}$$

$$y_m(t) = C_m(x_m)x_m(t) + D_m(x_m)u_m(t) \tag{21}$$

The model represents the desired behaviour of the plant, but is otherwise free and does not have to be the result of some prior modelling of the plant. It is also allowed to be of any arbitrarily high or low order. It will usually be a linear time-invariant model, and we represent it in the general, non-linear form only for the sake of the generality of the solution.

We now define the *output tracking error*

$$e_y(t) = y_m(t) - y_p(t) \tag{22}$$

and use the following simple multivariable adaptive control algorithm (Fig. 3)

(Bar-Kana and Kaufman 1983, 1985 b, Bar-Kana 1987 a)

$$u_p(t) = K_{e_y}(t)e_y(t) + K_{x_m}(t)x_m(t) + K_{u_m}(t)u_m(t) = K(t)r(t) \tag{23}$$

$$K(t) = [K_{e_y}(t) \quad K_{x_m}(t) \quad K_{u_m}(t)] \tag{24}$$

$$r^T(t) = [e_y^T(t) \quad x_m^T(t) \quad u_m^T(t)] \tag{25}$$

where the adaptive gains are computed as a combination of 'proportional' and 'integral' gains:

$$K_p(t) = [e_y(t)e_y^T(t)\Gamma_{e_y} \quad e_y(t)x_m^T(t)\Gamma_{x_m} \quad e_y(t)u_m^T(t)\Gamma_{u_m}] = e_y(t)r^T(t)\Gamma \tag{26}$$

$$\dot{K}_I(t) = e_y(t)r^T(t)\Gamma - \sigma K_I(t) \tag{27}$$

$$K(t) = K_p(t) + K_I(t) \tag{28}$$

where Γ and $\bar{\Gamma}$ are selected positive definite scaling matrices.

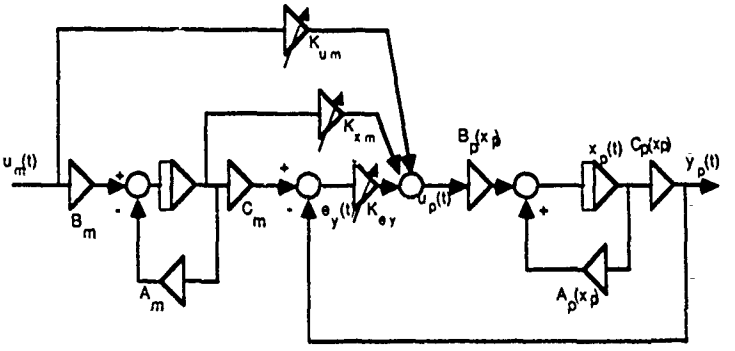


Figure 3. Complete adaptive control system.

The basic algorithm (23)–(28), without the σ-term in (27), was introduced by Sobel *et al.* (1979, 1982) and extended by Bar-Kana and Kaufman (1983, 1984, 1985 a). The gain $K_{e_y}(t)$ takes care of the stability of the controlled plant, while $K_{x_m}(t)$ and $K_{u_m}(t)$ help improve the performance of the adaptive model-following system and even achieve perfect following in ideal clean environments (Bar-Kana and Kaufman 1985 a).

The σ-term is a 'forgetting' factor (Ioannou and Kokotovic 1983) that avoids divergence of the integral gain in the presence of disturbances. Since it is only a low-pass filtering of $e_y(t)r^T(t)\Gamma$, this gain will not diverge unless the output tracking error diverges. Furthermore, the adaptive gain now moves up and down according to the specific situation (errors), as adaptive gains should do. The proportional gain $K_p(t)$ is added for its immediate action. Owing to this action, it heavily punishes high tracking errors and quickly brings the system towards tracking with small errors (Landau 1979).

The efficiency of the integral term $K_{e_y}(t)$ was also recently demonstrated (Nussbaum 1983, Willems and Byrnes 1984, Morse 1984, Bar-Kana 1989 d) with respect to the problem of the unknown sign of the high frequency gain.

The subsequent theorem of robust adaptive stability extends the applicability of this simple adaptive algorithm to 'almost passive' multivariable non-linear systems, and gives the necessary theoretical background for the successful applications of the

algorithm in non-linear or non-stationary systems (Bar-Kana 1987 a, b, 1988 c, Bar-Kana and Kaufman 1988).

Theorem 2

The model reference adaptive controller (23)–(28) guarantees robust adaptive stabilization of the ASP plant (18)–(19) in the presence of any bounded input commands and input or output disturbances.

Proof

For the proof, see Appendix A for causal and Appendix B for strictly causal ASP systems.

'Robust adaptive stabilization' means that all values involved in the adaptation process, namely states, gains and errors, are bounded in the presence of any bounded input commands and input or output disturbances. It should be mentioned that this is a very general result because we do not assume any prior knowledge on the plant, and because a perfect tracking solution may not exist since the model reference represents only some desired behaviour of an ideal plant. However, because stability of the non-linear adaptive controller with ASP plants is guaranteed, the non-linear controller gains are called to make the tracking errors arbitrarily small, as a trade-off between large gains and large errors. Asymptotic perfect following or asymptotic perfect regulation are obtained in idealistic conditions. Note that we do not need the usual assumptions that the dynamics (Seraji 1989) or the adaptation (Ortega and Spong 1988) are slow in order to guarantee stability of this adaptive control system.

Although Theorem 2 guarantees the boundness of all dynamic values involved in the adaptation process, some undesired phenomena were observed in the adaptive algorithms with forgetting factors (Åström 1983, Anderson 1985, Bar-Kana 1988 b, 1989 c, Fortesque *et al.* 1981, Hsu and Costa 1987) when no external excitation is present. In our configuration, these effects are apparent in particular if the controlled plant is unstable (Hsu and Costa 1987, Bar-Kana 1988 b). In this case, the adaptive gain may initially increase due to the initial errors, and reach some stabilizing value. As a result, the states and outputs move towards their zero value, and a decrease of the output leads to a decrease of the gains towards their zero value. However, since the equilibrium point ($y_a(t) = 0$, $K(t) = 0$) is unstable, and since on the other hand all values are bounded, we can expect that the system will reach some other limiting trajectories or equilibrium points (Willems 1971).

Since the gain initially has a stabilizing effect, the states and outputs come very close to their zero value. Therefore, the gains must go well into the unstable region before their destabilizing effect is felt. We therefore have a sudden 'burst' of error that brings the gains into the stable region again, and so on (Mareels and Bitmead 1986, Bar-Kana 1988 b, 1989 c). If we use fast adaptation, which in our case means using high adaptation coefficients Γ and $\bar{\Gamma}$, the adaptive gains increase immediately to such values that the tracking errors can be reduced to less than noticeable dimensions (Bar-Kana 1988 b). However, when the phenomenon appears, it finally settles at, or about, the minimum stabilizing gain $K_{\min}$. As shown below, we can use this value to totally eliminate the bursts.

If we replace $K_{e_y}(t)$ in (24) by

$$K_{e_y}(t) + K_0 \tag{29}$$

where K_0 is any stabilizing gain

$$K_0 \geqslant K_{min} \tag{30}$$

then the bursting phenomena are eliminated, because if either $\|y_a(t)\|$, $\|K(t)\|$ or $\|K(t)y_a(t)\|$ become small, the adaptive system enters the domain of attraction of the equilibrium point ($y_a(t) = 0$, $K(t) = K_0$) (Appendix C). Since it can be shown that the equilibrium point ($x_p(t) = 0$, $K(t) = K_0$) is both asymptotically stable (attractive) and unique, the eventual existence of other equilibrium points is excluded (Appendix D).

In conclusion, although in general prior knowledge of the bounds of stability, K_{min}–K_{max}, which is admissible with fixed controllers, does not guarantee stability with non-linear controllers (Aizerman and Gantmacher 1964), stability *is* guaranteed in the specific case of this particular non-linear adaptive algorithm.

4. Adaptive control of manipulators

A state space representation of the manipulator is (Slotine and Li 1988):

$$\dot{x}_p(t) = A_p(x_p)x_p(t) + B_p(x_p)u_p(t) + d(x_p) \tag{31}$$

$$y_p(t) = x_p(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{32}$$

where $x_p \in \mathbb{R}^{2n}$, $y_p \in \mathbb{R}^{2n}$, $u_p \in \mathbb{R}^n$, and where $x_1(t)$ is the n position-state vector and $x_2(t)$ is the n velocity-state vector. The various matrices of corresponding dimensions in (31), (32) are:

$$A_p(x_p) = \begin{bmatrix} 0 & I \\ 0 & -H^{-1}(x_1)C(x_1, x_2) \end{bmatrix} \tag{33}$$

$$B_p(x_p) = \begin{bmatrix} 0 \\ H^{-1}(x_1) \end{bmatrix} \tag{34}$$

$$d(x_p) = \begin{bmatrix} 0 \\ -H^{-1}(x_1)g(x_1) \end{bmatrix} \tag{35}$$

where $d(x_p)$ is considered to be a state-dependent disturbance generated by the gravity $g(x_1)$.

We first neglect the gravity term in order to establish some basic relations and show that the ideal system can be stabilized by a feedback controller of the form

$$u_p(t) = -K(x_1 + \alpha x_2) \tag{36}$$

for any positive definite matrix K and positive coefficient α. In other words, the closed-loop system

$$\dot{x}_p(t) = A_{CL}(x_p)x_p(t) \tag{37}$$

where

$$A_{CL}(x_p) = \begin{bmatrix} 0 & I \\ -H^{-1}(x_1)K & -H^{-1}(x_1)[C(x_1, x_2) + \alpha K] \end{bmatrix} \tag{38}$$

is uniformly asymptotically stable. To this end we select the positive definite Lyapunov function

$$V(x_p) = x_p^T(t) P(x_p) x_p(t) \tag{39}$$

where

$$P(x_p) = \begin{bmatrix} K & 0 \\ 0 & H(x_1) \end{bmatrix} \tag{40}$$

and therefore, the derivative of the Lyapunov function along the trajectories of (37) is

$$\begin{aligned} \dot{V}(x_p) &= x_p^T(t)[\dot{P}(x_p) + P(x_p)A_{CL}(x_p) + A_{CL}^T(x_p)P(x_p)]x_p(t) \\ &= x_p^T(t)\begin{bmatrix} 0 & K - K^T \\ K^T - K & \dot{H}(x_1) - C(x_1, x_2) - C^T(x_1, x_2) - \alpha(K + K^T) \end{bmatrix} x_p(t) \end{aligned} \tag{41}$$

and where we use (Slotine and Li 1988) the manipulator property that $H(x_1)$ is positive definite and that $\dot{H}(x_1) = C(x_1, x_2) + C^T(x_1, x_2)$, to obtain finally

$$\dot{V}(x_p) = -2\alpha x_2^T(t) K x_2(t) \tag{42}$$

Although $\dot{V}(x_p)$ is only negative semidefinite rather than negative definite, uniform asymptotic stability of (37) is guaranteed by the following theorems, which we formulate in a form that fits our specific problem.

Theorem 3 (Lyapunov: see Vidyasagar 1978)

Let $V(x_p)$ be a positive definite Lyapunov function of $x_p(t)$. Then, the system (37) is globally asymptotically stable if and only if the derivative of the Lyapunov function along the trajectories of (37) is a negative definite function of $x_p(t)$ (Vidyasagar 1978).

Theorem 4 (LaSalle 1981)

Under the assumptions of Theorem 3, the system (37) is stable if the derivative of the Lyapunov function along its trajectories is negative semidefinite. The trajectories of the system finally reach the region defined by $\dot{V}(x_p) \equiv 0$ (LaSalle 1981). The system (37) is still asymptotically stable if the negative semidefinite derivative $\dot{V}(x_p)$ is *not* identically zero along any non-trivial solution of (37).

In our case, since $\dot{V}(x_p)$ is quadratic, the limiting trajectories condition $\dot{V}(x_p) \equiv 0$ is equivalent to $x_2(t) = \dot{x}_1(t) \equiv 0$, which implies $x_1(t) \equiv 0$, and therefore $x_p(t) \equiv 0$.

Although stabilizability of the robotic manipulator through some PD controller was established using a negative semidefinite derivative of the Lyapunov function, the passivity relations needed for the convergence of the non-linear adaptive controller require that the underlying $\dot{V}(x_p)$ be negative definite, or have a negative definite upper bound with respect to $x_p(t)$. Now, since for any $x_2(t) \neq 0$ we have $V(x_p) > 0$ (strictly positive) and $\dot{V}(x_p) < 0$ (strictly negative), which implies that $V(x_p)$ is strictly decreasing, then $\dot{V}(x_p)/V(x_p) \leqslant -\beta$, for some scalar $\beta > 0$, sufficiently small. Then $\dot{V}(x_p) \leqslant -\beta V(x_p)$ and

$$\dot{V}(x_p) = -2\alpha x_2^T(t) K x_2(t) \leqslant -\beta x_p^T(t) P(x_p) x_p(t) = -\beta V(x_p) \tag{43}$$

and therefore all passivity lemmas and the proofs of the robust stability of the adaptive systems are valid in the case of robot manipulators.

The stabilizability of a robot via PD controllers is only needed as underlying knowledge. The implementation of the simplified adaptive controller does not actually make use of the derivative of the output and we can only use position sensors. Instead of the fictitious stabilizing PD controller $H(s)$ given by (36) we use the configuration $H^{-1}(s)$ given by

$$y_s(s) = \frac{K_{\max}^1}{1+\alpha s} u_p(s) \tag{44}$$

in parallel with the controlled plant (Bar-Kana 1987 a), where $K_{\max}$ is some reasonable evaluation of the maximal admissible gain K in (36), which theoretically may be arbitrarily large. Since we treat the gravity term as some unknown and unmeasured bounded disturbance, and since we do not use any identification algorithm, the adaptive controller can only guarantee stability with respect to the boundness of all states, gains, and tracking errors, in general. However, since the adaptive algorithm moves the gains up and down in order to reduce the errors, this residual error may be negligible (Bar-Kana 1987 a, Laniado *et al.* 1989). The ideal perfect tracking is thus given up to obviate the need for more complex algorithms using identifiers in the closed-loop.

5. Conclusions

In this paper the basic stabilizability properties of systems have been used to achieve 'almost passive' configurations for a class of non-linear continuous-time systems. It has also been shown that the 'almost passivity' property of systems can be used to extend the applicability of simple non-linear adaptive algorithms to non-linear systems, since it guarantees the global stability and robustness of the adaptive control systems in realistic environments. Other prior knowledge and conditions, usually needed, such as the order of the plant, the relative degree, inverse stability, stationarity, external excitation, etc. are immaterial in this context. The paper suggests a simple and robust adaptive controller for non-linear systems with unknown parameters, with particular application for robot manipulators.

ACKNOWLEDGMENTS

The authors are grateful to Howard Kaufman for valuable comments and suggestions regarding this paper.

This paper is based on research supported in part by Drexel University's Stein Fellowship Foundation and by AFOSR Grant No. 890010.

Appendix A

Proof of stability of the adaptive control algorithm (23)–(28)

The adaptive controller is desired to bring the control system to that ideal situation where perfect output tracking is performed. Therefore, we shall first question

the possible existence of some 'ideal' bounded trajectories $x_p^*(t)$ and 'ideal' control $x_p^*(t)$ that satisfy some unknown differential equation

$$\dot{x}_p^*(t) = A_p^*(x_p^*)x_p^*(t) + B_p^*(x_p^*)u_p^*(t) \tag{A 1}$$

$$y_p^*(t) = C_p(x_p)x_p^*(t) + D_p(x_p)u_p^*(t) \tag{A 2}$$

where the matrices $A_p^*(x_p^*)$, $B_p^*(x_p^*)$, $C_p(x_p)$, and $D_p(x_p)$ are uniformly bounded. Note that the ideal system (A 1) may be entirely different from the plant to be controlled, and that only their output equations (A 2) are assumed to be equal.

We now try to represent $x_p^*(t)$ and $u_p^*(t)$ as linear combinations (Bar-Kana and Kaufman 1985 a) of $x_m(t)$ and $u_m(t)$

$$x_p^*(t) = Xx_m(t) + Uu_m(t) \tag{A 3}$$

$$u_p^*(t) = \tilde{K}_{x_m}x_m(t) + \tilde{K}_{u_m}u_m(t) \tag{A 4}$$

If the plant states reach the ideal trajectories, $x_p(t) = x_p^*(t)$ we require that

$$y_p^*(t) = C_p(x_p^*)x_p^*(t) + D_p(x_p^*)u_p^*(t) = y_m(t) \tag{A 5}$$

Substituting (A 2)–(A 4) and (22) into (A 5) gives

$$C_p(x_p^*)Xx_m(t) + C_p(x_p^*)Uu_m(t) + D_p(x_p^*)\tilde{K}_{x_m}x_m(t) + D_p(x_p^*)\tilde{K}_{u_m}u_m(t) = C_m(x_m)x_m(t) + D_m(x_m)u_m(t) \tag{A 6}$$

or

$$[C_p(x_p^*)X + D_p(x_p^*)\tilde{K}_{x_m} - C_m(x_m)]x_m(t) + [C_p(x_p^*)U + D_p(x_p^*)\tilde{K}_{u_m} - D_m(x_m)]u_m(t) = 0 \tag{A 7}$$

It is clear, of course, that if the meauring matrices, C_p, C_m, D_p, and D_m are all constant, (A 7) usually has a solution (at least) for the unknown matrices X, U, $\tilde{K}_{x_m}$, and $\tilde{K}_{u_m}$, because it has many more variables than equations. In the general non-linear case, we may think that these unknown matrices are functions of x_p^*. In reality, x_p^* is any bounded trajectory that may satisfy (A 7) and is free otherwise. Therefore, we can assume that (A 7) is satisfied, in general.

Notice that even if (A 7) assumes that the ideal trajectories exist, perfect tracking by the real plant may not be possible, in the most general case, when the model and the plant are totally different. We leave this result in its most general form, to show that even in this case, when perfect tracking is not possible, the tracking errors can be reduced arbitrarily as a trade-off between large adaptive gains and large errors. Notice also that none of the solutions that were discussed above are needed for implementation of the adaptive algorithm (23)–(28). The conditions of existence that were established for the idealistic perfect following solution will subsequently be used for the proof of stability.

Since we want plant states to reach the 'ideal' states that allow perfect tracking, we define the state error as

$$e_x(t) = x_p^*(t) - x_p(t) \tag{A 8}$$

and the output error as

$$e_y(t) = y_m(t) - y_p(t) = y_p^*(t) - [C_p(x_p^*) - C_p(x_p)]x_p^*(t) - [D_p(x_p^*) - D_p(x_p)]u_p^*(t) - y_p(t) \quad \text{(A 9)}$$

$$e_y(t) = C_p(x_p)x_p^*(t) + D_p(x_p)u_p^*(t) - C_p(x_p)x_p(t) - D_p(x_p)u_p(t) - [C_p(x_p^*) - C_p(x_p)]x_p^*(t) - d_0(t) \quad \text{(A 10)}$$

$$e_y(t) = C_p(x_p)e_x(t) + D_p(x_p)[\tilde{K}_{x_m}x_m(t) + \tilde{K}_{u_m}u_m(t)] + D_p(x_p)\tilde{K}_{e_y}e_y(t) - D_p(x_p)\tilde{K}_{e_y}e_y(t) - D_p(x_p)K(t)r(t) - [C_p(x_p^*) - C_p(x_p)]x_p^*(t) - [D_p(x_p^*) - D_p(x_p)]u_p^*(t) - d_0(t) \quad \text{(A 11)}$$

$$e_y(t) = C_p(x_p)e_x(t) - D_p(x_p)[K(t) - \tilde{K}]r(t) - D_p(x_p)\tilde{K}_{e_y}e_y(t) - [C_p(x_p^*) - C_p(x_p)]x_p^*(t) - [D_p(x_p^*) - D_p(x_p)]u_p^*(t) - d_0(t) \quad \text{(A 12)}$$

Denote

$$d_1(t) = [C_p(x_p^*) - C_p(x_p)]x_p^*(t) + [D_p(x_p^*) - D_p(x_p)]u_p^*(t) + d_0(t) \quad \text{(A 13)}$$

Then

$$[I + D_p(x_p)\tilde{K}_{e_y}]e_y(t) = C_p(x_p)e_x(t) - D_p(x_p)[K(t) - \tilde{K}]r(t) - d_1(t) \quad \text{(A 14)}$$

and finally

$$e_y(t) = C_{pc}(x_p)e_x(t) - D_{pc}(x_p)[K(t) - \tilde{K}]r(t) - [I + D_p(x_p)\tilde{K}_{e_y}]^{-1}d_1(t) \quad \text{(A 15)}$$

where

$$\tilde{K} = [\tilde{K}_{e_y} \quad \tilde{K}_{x_m} \quad \tilde{K}_{u_m}] \quad \text{(A 16)}$$

and where $C_{pc}(x_p)$ and $D_{pc}(x_p)$ were defined in (9)–(10).

Differentiating $e_x(t)$ in (A 8) gives

$$\dot{e}_x(t) = \dot{x}_p^*(t) - \dot{x}_p(t) = \dot{x}_p^*(t) - A_p(x_p)x_p^*(t) + A_p(x_p)x_p^*(t) - \dot{x}_p(t) \quad \text{(A 17)}$$

$$\dot{e}_x(t) = A_p^*(x_p^*)x_p^*(t) + B_p^*(x_p^*)u_p^*(t) - A_p(x_p)x_p^*(t) + A_p(x_p)x_p^*(t) - A_p(x_p)x_p(t) - B_p(x_p)K(t)r(t) - d_i(t) \quad \text{(A 18)}$$

$$\dot{e}_x(t) = A_p(x_p)e_x(t) - B_p(x_p)K(t)r(t) + B_p(x_p)\tilde{K}_{x_m}x_m(t) + B_p(x_p)\tilde{K}_{u_m}u_m(t) - B_p(x_p)\tilde{K}_{e_y}e_v(t) + B_p(x_p)\tilde{K}_{e_y}e_y(t) + [A_p^*(x_p^*) - A_p(x_p)]Xx_m(t) + [A_p^*(x_p^*) - A_p(x_p)]Uu_m(t) - d_i(t) \quad \text{(A 19)}$$

$$\dot{e}_x(t) = A_p(x_p)e_x(t) - B_p(x_p)\tilde{K}_{e_y}e_y(t) - B_p(x_p)[K(t) - \tilde{K}]r(t) + \{[A_p^*(x_p^*) - A_p(x_p)]X + [B_p^*(x_p^*) - B_p(x_p)]\tilde{K}_{x_m}\}x_m(t) + \{[A_p^*(x_p^*) - A_p(x_p)]U + [B_p^*(x_p^*) - B_p(x_p)]\tilde{K}_{u_m}\}u_m(t) - d_i(t) \quad \text{(A 20)}$$

Substituting $e_y(t)$ from (A 15) gives

$$\begin{aligned}\dot{e}_x(t) = {} & A_p(x_p)e_x(t) - B_p(x_p)\tilde{K}_{e_y}[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}C_p(x_p) \\ & - B_p(x_p)\tilde{K}_{e_y}[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}D_p(x_p)[K(t) - \tilde{K}]r(t) \\ & - B_p(x_p)\tilde{K}_{e_y}[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}d_0(t) - B_p(x_p)[K(t) - \tilde{K}]r(t) \\ & + \{[A_p^*(x_p^*) - A_p(x_p)]X + [B_p^*(x_p^*) - B_p(x_p)]\tilde{K}_{x_m}\}x_m(t) \\ & + \{[A_p^*(x_p^*) - A_p(x_p)]U + [B_p^*(x_p^*) - B_p(x_p)]\tilde{K}_{u_m}\}u_m(t) - d_i(t) \quad \text{(A 21)}\end{aligned}$$

Since

$$\begin{aligned}& - B_p(x_p)\tilde{K}_{e_y}[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}D_p(x_p) + B_p(x_p) \\ & \qquad = B_p(x_p)\{I + \tilde{K}_{e_y}[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}D_p(x_p)\} \\ & \qquad\qquad \text{(using the matrix inversion lemma)} \\ & \qquad = B_p(x_p)[I + D_p(x_p)\tilde{K}_{e_y}]^{-1} = B_{pc}(x_p) \quad \text{(A 22)}\end{aligned}$$

we get

$$\dot{e}_x(t) = A_{pc}(x_p)e_x(t) - B_{pc}(x_p)[K(t) - \tilde{K}]r(t) + F(t) \quad \text{(A 23)}$$

where $A_{pc}(x_p)$, $B_{pc}(x_p)$, $K_{ec}(x_p)$ were defined in (4)–(10) and where $F(t)$ is the residual uniformly bounded term

$$\begin{aligned}F(t) = {} & \{[A_p^*(x_p^*) - A_p(x_p)]X + [B_p^*(x_p^*) - B_p(x_p)]\tilde{K}_{x_m}\}x_m(t) \\ & + \{[A_p^*(x_p^*) - A_p(x_p)]U + [B_p^*(x_p^*) - B_p(x_p)]\tilde{K}_{u_m}\}u_m(t) \\ & - B_p(x_p)\tilde{K}_{ec}d_1(t) - d_i(t) \quad \text{(A 24)}\end{aligned}$$

The following quadratic Lyapunov function will be used for the proof of stability of the adaptive system (A 22) and (27):

$$V(t) = e_x^T(t)P(x_p)e_x(t) + \operatorname{tr}\{[K_I(t) - \tilde{K}]\Gamma^{-1}[K_I(t) - \tilde{K}]^T\} \quad \text{(A 25)}$$

where $P(x_p)$ is the positive definite matrix defined in (11)–(13) and Γ is the positive definite scaling factor defined in (27). The derivative of $V(t)$ 'along the trajectories' of (A 22) and (27) gives:

$$\begin{aligned}\dot{V}(t) = {} & e_x^T(t)\dot{P}(x_p)e_x(t) + \dot{e}_x^T(t)P(x_p)e_x(t) + e_x^T(t)P(x_p)\dot{e}_x(t) \\ & - 2\operatorname{tr}\{[K_I(t) - \tilde{K}]\Gamma^{-1}\dot{K}_I^T(t)\} \quad \text{(A 26)}\end{aligned}$$

Substituting $\dot{e}_x(t)$ from (A 23) and $\dot{K}_I(t)$ from (27) gives

$$\begin{aligned}\dot{V}(t) = {} & e_x^T(t)\dot{P}(x_p)e_x(t) + \dot{e}_x^T(t)A_{pc}^T(x_p)P(x_p)e_x(t) \\ & - r^T(t)[K(t) - \tilde{K}]^T B_{pc}^T(x_p)P(t)e_x(t) + F^T(t)P(t)e_x(t) \\ & + e_x^T(t)P(x_p)A_{pc}(x_p)e_x(t) - e_x^T(t)P(x_p)B_{pc}(x_p)[K(t) - \tilde{K}]r(t) \\ & - e_x^T(t)P(x_p)F(t) + 2\operatorname{tr}\{[K_I(t) - \tilde{K}]\Gamma^{-1}[e_y(t)r^T(t)\Gamma - \sigma K_I(t)]\}\} \quad \text{(A 27)} \\ \dot{V}(t) = {} & e_x^T(t)[\dot{P}(x_p) + P(x_p)A_{pc}(x_p) + A_{pc}^T(x_p)P(x_p)]e_x(t) \\ & - e_x^T(t)P(x_p)B_{pc}(x_p)[K(t) - \tilde{K}]r(t) - r^T(t)[K(t) - \tilde{K}]^T B_{pc}^T(x_p)P(t)e_x(t) \\ & + 2e_x^T(t)P(x_p)F(t) \\ & - e_y^T(t)[K_I(t) - \tilde{K}]r(t) + r^T(t)[K_I(t) - \tilde{K}]^T e_y(t) \\ & - 2\sigma\operatorname{tr}\{[K_I(t) - \tilde{K}]\Gamma^{-1}K_I^T(t)\} \quad \text{(A 28)}\end{aligned}$$

Substituting

$$K_I(t) = K(t) - K_p(t) = K(t) - e_y(t)r^T(t)\Gamma \tag{A 29}$$

gives

$$\begin{aligned}\dot{V}(t) = {} & e_x^T(t)[\dot{P}(x_p) + P(x_p)A_{pc}(x_p) + A_{pc}^T(x_p)P(x_p)]e_x(t) \\ & - e_x^T(t)P(x_p)B_{pc}(x_p)[K(t) - \tilde{K}]r(t) - r^T(t)[K(t) - \tilde{K}]^T B_{pc}^T(x_p)P(t)e_x(t) \\ & + 2e_x^T(t)P(x_p)F(t) \\ & - e_y^T(t)[K(t) - \tilde{K}]r(t) + r^T(t)[K(t) - \tilde{K}]^T e_y(t) \\ & - 2e_y^T(t)e_y(t)r^T(t)\Gamma r(t) \\ & - 2\sigma \text{ tr } \{[K_I(t) - \tilde{K}]\Gamma^{-1}[K_I(t) - \tilde{K}]^T\} \\ & - 2\sigma \text{ tr } \{[K_I(t) - \tilde{K}]\Gamma^{-1}\tilde{K}^T\}\end{aligned} \tag{A 30}$$

Using (11)–(13) and (A 15) gives

$$\begin{aligned}\dot{V}(t) = {} & -e_x^T(t)Q(x_p)e_x(t) - e_x^T(t)L^T(x_p)L(x_p)e_x(t) \\ & - e_x^T(t)C_{pc}^T(x_p)[K(t) - \tilde{K}]r(t) - r^T(t)[K(t) - \tilde{K}]^T C_{pc}^T(x_p)e_x(t) \\ & - e_x^T(t)L^T(x_p)W(x_p)[K(t) - \tilde{K}]r(t) \\ & + r^T(t)[K(t) - \tilde{K}]^T W^T(x_p)L(x_p)e_y(t) \\ & + e_x^T(t)C_{pc}^T(x_p)[K(t) - \tilde{K}]r(t) \\ & - r^T(t)[K(t) - \tilde{K}]^T D_{pc}^T(x_p)[K(t) - \tilde{K}]r(t) \\ & - d_1^T(t)[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}[K(t) - \tilde{K}]r(t) \\ & + r^T(t)[K(t) - \tilde{K}]^T C_{pc}(x_p)e_x(t) \\ & - r^T(t)[K(t) - \tilde{K}]^T D_{pc}(x_p)[K(t) - \tilde{K}]r(t) \\ & - r^T(t)[K(t) - K]^T[I + D_p(x_p)K_{e_y}]^{-1}d_1(t) \\ & - 2e_y^T(t)e_y(t)r^T(t)\Gamma r(t) \\ & - 2\sigma \text{ tr } \{[K_I(t) - \tilde{K}]\Gamma^{-1}[K_I(t) - \tilde{K}]^T\} \\ & - 2\sigma \text{ tr } \{[K_I(t) - \tilde{K}]\Gamma^{-1}\tilde{K}^T\} \\ & + 2e_x^T(t)P(x_p)F(t)\end{aligned} \tag{A 31}$$

Rearranging and using (13) gives

$$\begin{aligned}\dot{V}(t) = {} & -e_x^T(t)Q(x_p)e_x(t) \\ & - \{L(x_p)e_x(t) - W(x_p)[K(t) - \tilde{K}]r(t)\}^T \\ & \times \{L(x_p)e_x(t) - W(x_p)[K(t) - \tilde{K}]r(t)\} \\ & - 2e_y^T(t)e_y(t)r^T(t)\Gamma r(t) \\ & - 2\sigma \text{ tr } \{[K_I(t) - \tilde{K}]\Gamma^{-1}[K_I(t) - \tilde{K}]^T\} \\ & - 2\sigma \text{ tr } \{[K_I(t) - \tilde{K}]\Gamma^{-1}\tilde{K}^T\} \\ & - 2r^T(t)[K(t) - \tilde{K}]^T[I + D_p(x_p)\tilde{K}_{e_y}]^{-1}d_1(t) \\ & + 2e_x^T(t)P(x_p)F(t)\end{aligned} \tag{A 32}$$

It is easy to see that some positive coefficients α_1–α_{10} exist such that

$$\dot{V}(t) \leqslant -\alpha_1 \|e_x(t)\|^2 - \alpha_2 \|[\tilde{K}(t) - \tilde{K}]r(t)\|^2 - \alpha_3 \|e_y(t)\|^4 - \alpha_4 \|e_y(t)\|^2 \|x_m(t)\|^2$$
$$- \alpha_5 \|e_y(t)\|^2 \|u_m(t)\|^2 - \alpha_6 \sigma \|K_I(t) - \tilde{K}\|^2 + \alpha_7 \sigma \|K_I(t) - \tilde{K}\|$$
$$+ \alpha_8 \|[K(t) - \tilde{K}]r(t)\| + \alpha_9 \|e_x(t)\| + \alpha_{10} \|e_x(t)\| \cdot \|e_y(t)\| \qquad \text{(A 33)}$$

If either $\|e_x(t)\|$, $\|[K(t) - \tilde{K}]r(t)\|$, or $\|K_I(t) - \tilde{K}\|$ increase beyond some bound, the negative definite quadratic terms in (A 33) become dominant, and thus $\dot{V}(t)$ becomes negative. The quadratic form of the Lyapunov function $V(t)$ then guarantees that all the dynamic values, namely $e_x(t)$, $K_I(t)$, and $e_y(t)$ are bounded. In idealistic situations with $d_I(t) = 0$ and $d_0(t) = 0$, $F(t)$ may vanish and thus allow perfect following (Bar-Kana and Kaufman 1985 a). If, however, the σ-term were missing in (27), then the adaptive controller could not have avoided those situations when $\|K_I(t) - \tilde{K}\|$ might increase without bound in spite of $\|[K(t) - \tilde{K}]r(t)\|$ being small. As shown in § 2, the properties of ASP systems guarantee that theoretically, the adaptive system remains stable, even if the gains increase without bound. Therefore, on one hand one can use very high adaptation scaling factors such that the weighting of the negative terms is dominant and the tracking errors become very (arbitrarily) small. However, the gains may also reach unnecessarily or unrealistically large values. The negative quadratic σ-term in (A 32) shows that such a situation is not possible and all values are therefore bounded. Moreover, the adaptive gains now move up and down according to the required performance (small tracking errors), as a practical implementation of the adaptive control aim to fitting the right gains to the right operational situation. This way, on one hand one can have smooth tracking with small adaptive gains (and low cost of control and low noise amplification) when the tracking problem is easy. On the other hand, the adaptive gains increase to any large desired value when required by the transient mode or the desired manoeuvrability, so that the tracking errors remain small. This property of the adaptive algorithm may give it an advantage over fixed linear controllers, even if the plants (not the operational environment) were perfectly known.

As already mentioned, only the $K_{e_y}(t)$ terms are needed for the stability of the adaptive control system. The effect of the other terms, like the proportional adaptive gains or the gains multiplying $x_m(t)$ and $u_m(t)$ is expressed by the corresponding supplementary negative terms in (A 32) and (A 33), which increase the rate of convergence and also reduce the bounds of the final bounded error (where $\dot{V}(t)$ is not necessarily negative) thus improving the performance of the adaptive controller. Notice that without external disturbances, the smaller the tracking error, the closer $x_p(t)$ is to the ideal trajectory and, in turn, the smaller the residual term $F(t)$. Although in general, the adaptive controller does not promise perfect tracking, in most situations the tracking errors are very small and sometimes negligible. At the price of bounded rather than vanishing errors, the adaptive controller maintains the stability of the adaptive control system in most practical non-idealistic and ever changing situations (Bar-Kana 1988 c, 1989 d, Bar-Kana and Kaufman 1988, Morse and Ossman 1989).

Appendix B

Adaptive control of strictly causal almost passive systems

The adaptive controller (37)–(42) can also be applied to *strictly causal* ASP systems, that satisfy relations (15), (16).

The proof of stability in such a case follows immediately from the proof of Appendix A, if we substitute $D_p(x_p)=0$, $L(x_p)=0$, $W(x_p)=0$.

Then

$$\begin{aligned}\dot{V}(t) = &-e_x^T(t)Q(x_p)e_x(t)\\ &-2e_y^T(t)e_y(t)r^T(t)\Gamma r(t)\\ &-2\sigma\,\mathrm{tr}\,\{[K_I(t)-\tilde{K}]\Gamma^{-1}[K_I(t)-\tilde{K}]^T\}\\ &-2\sigma\,\mathrm{tr}\,\{[K_I(t)-\tilde{K}]\Gamma^{-1}\tilde{K}^T\}\\ &-2r^T(t)[K(t)-\tilde{K}]^T d_1(t)\\ &+2e_x^T(t)P(x_p)F(t)\end{aligned} \tag{B 1}$$

and the stability results following (A 32)–(A 33) are also perfectly valid here.

Appendix C

Role of the stabilizing gain K_0

Let us assume that we used the fixed gain K_0 and control the plant

$$\dot{x}_p(t) = A_0(x_p)x_p(t) + B_p(x_p)u_p(t) \tag{C 1}$$

$$y_a(t) = C_p(x_p)x_p(t) + D_p(x_p)x_p(t) \tag{C 2}$$

where

$$A_0(x_p) = A_p(x_p) - B_p(x_p)K_0C_p(x_p) \tag{C 3}$$

Since (C 1) is, by assumption, uniformly asymptotically stable, then it can be seen (Bar-Kana 1989 a) that there exist some positive definite matrices $P_0(x_p)$ and $Q_0(x_p)$, and some matrices $L_0(x_p)$, W_0, and D_0 such that the following relations are satisfied:

$$\dot{P}_0(x_p) + P_0(x_p)A_0(x_p) + A_0^T(x_p)P_0(x_p) = -Q_0(x_p) - L_0^T(x_p)L_0(x_p) \tag{C 4}$$

$$P_0(x_p)B_p(x_p) = C_p^T(x_p) - L_0^T(x_p)W_0 \tag{C 5}$$

$$D_0 + D_0^T = W_0^TW_0 \tag{C 6}$$

Let us assume that no external input command or disturbance is present, and that therefore $u_m(t)=0$, $x_m(t)=0$, $y_m(t)=0$, $K_x(t)=0$, $K_y(t)=0$, and for convenience, $K_p(t)=0$, $K_I(t)=K_{e_y}(t)=K_{e_yI}(t)$.

We use the following Lyapunov function:

$$V_1(t) = x_p^T(t)P_0(x_p)x_p(t) \tag{C 7}$$

and the derivative of $V_1(t)$ along the trajectories of (C 1)–(C 2) is

$$\begin{aligned}\dot{V}_1(t) = &\,x_p^T(t)[\dot{P}_0(x_p) + P_0(x_p)A_0(x_p) + A_0^T(x_p)P_0(x_p)]x_p(t)\\ &+ x_p^T(t)P_0(x_p)B_p(x_p)u_p(t) + u_p^T(t)B_p^T(x_p)P_0(x_p)x_p(t)\end{aligned} \tag{C 8}$$

Substituting $u_p(t)$ from (23) and using (C 4)–(C 6) gives

$$\begin{aligned}\dot{V}_1(t) = &-x_p^T(t)Q_0(x_p)x_p(t) - x_p^T(t)L_0^T(x_p)L_0(x_p)x_p(t)\\ &- x_p^T(t)C_p^T(x_p)K_I(t)y_a(t) - y_a^T(t)K_I^T(t)C_p(x_p)x_p(t)\\ &+ x_p^T(t)L_0^T(x_p)W_0K_I(t)y_a(t) + y_a^T(t)K_I^T(t)W_0^TL_0(x_p)x_p(t) \qquad \text{(C 9)}\\ \dot{V}_1(t) = &-x_p^T(t)Q_0(x_p)x_p(t) - x_p^T(t)L_0^T(x_p)L_0(x_p)x_p(t)\\ &+ z_p^T(t)L_0^T(x_p)W_0K_I(t)y_a(t) + y_a^T(t)K_I^T(t)W_0^TL_0(x_p)x_p(t)\\ &- y_a^T(t)K^T(t)(D_0 + D_0^T)K_I(t)y_a(t)\\ &- y_a^T(t)K_I^T(t)(D_0 + D_0^T)K_I(t)y_a(t)\\ &- t_a^T(t)K_I^T(t)[D_p(x_p) + D_p^T(x_p)]K_I(t)y_a(t)\\ &- y_a^T(t)[K_I(t) + K_I^T(t)]y_a(t) \qquad \text{(C 10)}\\ \dot{V}_1(t) = &-x_p^T(t)Q_0(x_p)x_p(t)\\ &- [L_0(x_p)x_p(t) - W_0K_I(t)y_a(t)]^T[L_0(x_p)x_p(t) - W_0K_I(t)y_a(t)]\\ &- y_a^T(t)[K_I(t) + K_I^T(t)]y_a(t)\\ &+ y_a^T(t)K_I^T(t)[D_0 + D_0^T - D_p(x_p) - D_p^T(x_p)]K_I(t)y_a(t) \qquad \text{(C 11)}\end{aligned}$$

Because only the last term in (C 11) is not negative, then if either $\|y_a(t)\|$, $\|K(t)\|$ or $\|K(t)y_a(t)\|$ becomes small, the adaptive system enters the domain of attraction of the equilibrium point ($y_a(t) = 0$, $K(t) = K_0$). Because boundness of all values is guaranteed (Appendixes A and B), the conclusions that follow (30) are immediate.

We now show that for any $K_0 \geqslant K_{\min}$, the point ($x_p = 0$, $K_I(t) = 0$) or ($x_p = 0$, $K(t) = K_0$) is also the only equilibrium point of the system, if we restrict the discussion to those non-linear systems for which uniform asymptotic stability guarantees that the system matrix is non-singular.

We then assume that before applying the adaptive controller, we used some preliminary constant matrix K_0 such that the system represented by

$$A_0(x_p) = A_p(x_p) - B_p(x_p)K_0C_p(x_p) \qquad \text{(C 12)}$$

is uniformly asymptotically stable.

The maximal admissible (supplementary) feedback gain is then

$$K_{\max} - K_0 \qquad \text{(C 13)}$$

and therefore

$$D_p = [K_{\max} - K_0]^{-1} \qquad \text{(C 14)}$$

From (C 1) we get

$$\dot{x}_p(t) = A_K(x_p)x_p(t) \qquad \text{(C 15)}$$

where

$$A_K(x_p) = A_0(x_p) - B_p(x_p)K_c(t)C_p(x_p) \qquad \text{(C 16)}$$

$$u_p(t) = -K_I(t)y_a(t) = -K_I(t)C_p(x_p)x_p(t) - K_I(t)D_px_p(t) \qquad \text{(C 17)}$$

$$u_p(t) = -K_c(t)C_p(x_p)x_p(t) \qquad \text{(C 18)}$$

$$K_c(t) = [I + K_I(t)D_p]^{-1}K_I(t) \qquad \text{(C 19)}$$

Form (C 19) it can be seen again that

$$0 < K_I(t) < \infty \quad \text{implies} \quad 0 < K_c(t) < D_p^{-1} = K_{max} - K_0 \qquad \text{(C 20)}$$

The equilibrium points of (C 15) and (27) are obtained from

$$A_K(x_p)x_p(t) = 0 \qquad \text{(C 21)}$$

$$\sigma K_I(t) - y_a(t)y_a^T(t)\Gamma = 0 \qquad \text{(C 22)}$$

If $K_0 > K_{min}$, then $A_K(x_p)$ is stable for any value of $K_c(t)$ and it is, therefore, non-singular. In this case (C 21) has only the unique solution $x_p(t) = 0$, and then, from (C 22) we get $K_I(t) = 0$.

In conclusion, if $K_0 > K_{min}$, then the equilibrium point ($x_p(t) = 0$, $K_I(t) = 0$) or ($x_p(t) = 0$, $K(t) = K_0$) is both stable (attractive) and unique.

References

Aizerman, M. A., and Gantmacher, F. R., 1964, *Absolute Stability of Regulator Systems* (San Francisco: Holden Day).

Anderson, B. D. O., 1985, Adaptive systems, lack of persistency of excitation and bursting phenomena. *Automatica*, **21**, 247–258.

Åström, K. J., 1983, Theory and applications of adaptive control—a survey. *Automatica*, **19**, 471–481.

Balas, M., Kaufman, H., and Wen, J., 1984, Stable direct adaptive control of linear infinite dimensional systems using command generator approach. *Workshop on Identification and Control of Flexible Space Structures*, San Diego, California.

Bar-Kana, I., 1986 a, Positive realness in discrete-time adaptive control systems. *International Journal of Systems Sciences*, **17**, 1001–1006; also in *Proceedings of the American Control Conference*, Seattle, Washington, pp. 1440–1443; 1986 b, Extension of a continuous-time multivariable adaptive control algorithm. *Ibid.*, 1081–1086; 1987 a, Parallel feedforward and simplified adaptive control. *International Journal of Adaptive Control and Signal Processing*, **1**, 95–109; 1987 b, Adaptive control—a simplified approach. *Control and Dynamic Systems—Advances in Theory and Applications*, Vol. 25, edited by C. Leondes, pp. 187–235; 1988 a, *Rohrs examples and robustness of simplified adaptive control*. Technical Report, Rafael; 1988 b, *Reduction of bursting without external excitation*. Technical Report, Rafael; 1988 c, Comments on a paper by Kidd. *International Journal of Control*, **48**, 1011–1023; 1988 d, On the Lurė problem and stability of non-linear controllers. *Journal of The Franklin Institute*, **325**, 687–693; 1988 e, Almost passivity and simple adaptive control of nonstationary continuous linear systems. Technical Report, Rafael; 1989 a, On passivity of a class of non-linear systems. Technical Report, Drexel University; 1989 b, Positive realness in multivariable stationary linear systems. *Proceedings of the 1989 Conference on Informational Sciences and Systems* (Baltimore, Md: Johns Hopkins University Press), pp. 383–388; 1989 c, Absolute stability and robust discrete adaptive control of multivariable systems. *Control and Dynamic Systems—Advances in Theory and Applications*, edited by C. Leondes, Vol. 31, pp. 157–183; 1989 d, Robust simplified adaptive stabilization of not necessarily minimum-phase systems. *Transactions of the American Society of Mechanical Engineers*, Ser G, *Journal of Dynamic Systems, Measurements, and Control*, **111**, 364–370; also (in part) *Proceedings of the 1988 American Control Conference*, Atlanta, Georgia, pp. 760–765.

Bar-Kana, I., Fischl, R., and Kalata, P., 1989, Direct position-plus-velocity feedback control of large flexible space structures. *Proceedings of the 1989 Conference on Informational Sciences and Systems* (Baltimore, Md: Johns Hopkins University Press), pp. 574–577.

Bar-Kana, I., and Kaufman, H., 1982 a, Model reference adaptive control for time-variable input commands. *Proceedings of the 1982 Conference on Informational Sciences and Systems*, Princeton, New Jersey, pp. 208–211; 1982 b, Multivariable direct adaptive control for a general class of time-variable commands. *Proceedings of the 21st I.E.E.E*

Conference on Decision and Control, Orlando, Florida, pp. 750–751; 1983, Direct adaptive control with bounded tracking errors. *Proceedings of the 22nd Conference on Decision and Control*, San Antonio, Texas, pp. 181, 182; 1984, Some applications of direct adaptive control to large structural systems. *A.I.A.A. Journal of Guidance, Control and Dynamics*, **7**, 717–724; also *Aerokosmicheskaia Technika*, **3**, 88–96 (in Russian); 1985 a, Global stability and performance of a simplified adaptive algorithm. *International Journal of Control*, **42**, 1491–1505; 1985 b, Robust simplified adaptive control for a class of multivariable continuous-time systems. *Proceedings of the 24th Conference on Decision and Control*, pp. 141–146; 1987, Robust simplified adaptive control of large flexible space structures. *Control of Distributed Parameter Systems*, IFAC Proceedings Series, edited by H. E. Rauch, Vol. 3, pp. 121–126; also in (1988), Technical Report, Rafael, full report; 1988, Simple adaptive control of uncertain systems. *International Journal of Adaptive Control and Signal Processing*, **2**, 133–143

BAR-KANA, I., KAUFMAN, H., and BALAS, M., 1983, Model reference adaptive control of large structural systems. *A.I.A.A. Journal of Guidance, Control and Dynamics*, **6**, 112–118; also in *Aerokosmicheskaia Technika*, **1**, 163–171 (in Russian).

DONALDSON, D. D., and LEONDES, C. T., 1963, A model referenced parameter tracking technique for adaptive control systems, Part I and II. *Journal of the American Society of Mechanical Engineers*.

FORTESQUE, R. R., KERSHENBAUM, L. S., and YDSTIE, B. D. E., 1981, Implementation of self-tuning regulators with variable forgetting factors. *Automatica*, **17**, 831–835.

GUEZ, A., 1980, Application of singular perturbation methods to air-to-air missile control. *Technical Report, System Dynamics*.

GUEZ, A., and DRITSAS, L., 1987, Ultimate boundedness of robot tracking. *Proceedings of the 2nd International I.E.E.E. Intelligent Control Conference*, Philadelphia, Pennsylvania.

HAHN, W., 1967, *Stability of Motion* (New York: Springer-Verlag).

HSU, L., and COSTA, R. R., 1987, Bursting phenomena in continuous-time adaptive systems with σ-modification. *I.E.E.E. Transations on Automatic Control*, **32**, 84–85.

IH, C. H. C., WANG, S. J., and LEONDES, C. T., 1987, Adaptive control for the space station. *Control Systems Magazine*, **7**, 29–34.

IOANNOU, P. A., 1986, Adaptive stabilization of not necessarily minimum phase plants. *Systems and Control Letters*, **7**, 291–287.

IOANNOU, P. A., and KOKOTOVIC, P. V., 1983, *Adaptive Systems with Reduced Models* (New York: Springer-Verlag).

KAUFMAN, H., BALAS, M., BAR-KANA, I., and RAO, L., 1981, Model reference adaptive control of large scale systems. *Proceedings of the 20th I.E.E.E. Conference of Decision and Control*, San Diego, California, pp. 984–989.

KIDD, P. T., 1985, Comparison of the performance of a model reference adaptive system and a classical linear control system under non-deal conditions. *International Journal of Control*, **42**, 671–694.

LANDAU, I., 1979, *Adaptive Control—the Model Reference Approach* (New York: Marcel Dekker).

LANIADO, I., KREINDLER, E., and BAR-KANA, I., 1989, Simple adaptive control of a robot arm. *Proceedings of the 1989 I.E.E.E. Conference on Control and Applications—ICCON*, Jerusalem.

LASALLE, J. P., 1981, Stability of Nonautonomous Systems. *Non-linear Analysis Theory, Methods and Applications*, **1**, 83–91.

MAREELS, I. M. Y., and BITMEAD, R. R., 1986, Nonlinear dynamics in adaptive control—chaotic and periodic stabilization. *Automatica*, **22**, 641–655.

MATTERN, D. L., and SHOURESHI, R., 1987, Model reference adaptive control—a perspective. *Proceedings of the 1987 American Control Conference* pp. 201–206.

MONOPOLI, R. V., 1974, Model reference adaptive control with an augmented error signal. *I.E.E.E. Transactions on Automatic Control*, **25**, 433–439.

MORSE, A. S., 1984, New directions in parameter adaptive control systems. *Proceedings of the 23rd Conference of Decision and Control*, Las Vegas, Nevada, 1556–1568.

MORSE, W., and OSSMAN, K., 1989, Flight control reconfiguration using model reference adaptive control. *Proceedings of the 1989 American Control Conference*, Pittsburgh, Pennsylvania, pp. 159–16

NARENDRA, K. S., and VALAVANI, L., 1979, Direct and indirect adaptive control. *Automatica*, **15**, 653–661.

NUSSBAUM, R. O., 1983, Some remarks on a conjecture in parameter adaptive control. *Systems and Control Letters*, **3**, 243–246.

ORTEGA, R., and SPONG, M., 1988, Adaptive motion control of rigid robots: a tutorial. *Proceedings of the 27th Conference on Decision and Control*, Austin, Texas, pp. 1575–1584.

OSBURN, P. V., WHITAKER, H. P., and KEZER, A., 1961, New developments in the design of model reference adaptive control systems. *Institute of Aeronautical Sciences*, Paper 61–39.

PARKS, P. C., 1966, Lyapunov redesign of model reference adaptive control systems. *I.E.E.E. Transactions on Automatic Control*, **11**, 362–367.

ROHRS, C., VALAVANI, L., ATHANS, M., and STEIN, G., 1982, Robustness of adaptive control algorithms in the presence of unmodelled dynamics. *Proceedings of the 21st I.E.E.E. Conference on Decision and Control*, Orlando, Florida, pp. 3–11; 1985, Robustness of continuous-time adaptive control algorithms in the presence of unmodelled dynamics. *I.E.E.E. Transactions on Automatic Control*, **30**, 881–889.

SERAJI, H., 1989, Decentralized adaptive control of manipulators: theory, simulation, and experimentation. *I.E.E.E. Transactions on Robotics and Automation*, **5**, 183–201.

SHAKED, U., 1977, The zero properties of linear passive systems. *I.E.E.E. Transactions on Automatic Control*, **22**, 973–976.

SLOTINE, J.-J. E., and LI, W., 1988, Adaptive manipulator control: a case study. *I.E.E.E. Transactions on Automatic Control*, **33**, 995–1003.

SOBEL, K., KAUFMAN, H., and MABIUS, L., 1979, Model reference output control systems without parameter identification. *Proceedings of the 18th Conference on Decision and Control*, Fort Lauderdale, Florida; 1982, Implicit adaptive control for a class of MIMO systems. *I.E.E.E. Transactions on Aerospace and Electronic Systems*, **18**, 576–590.

VIDYASAGAR, M., 1978, *Nonlinear Systems Analysis* (Engelwood Cliffs, NJ: Prentice Hall).

WHITAKER, H. P., 1959, An adaptive system for control of the dynamics performance of aircraft and spacecraft. *Institute of Aeronautical Sciences*, Paper 59–100.

WILLEMS, J. C., 1971, *The Analysis of Feedback Systems* (Cambridge, Mass: MIT Press); 1972, Dissipative dynamical systems. *Archive for Rational Mechanics and Analysis*, Vol. 45 (New York: Springer-Verlag), pp. 321–393.

WILLEMS, J. C., and BYRNES, C. I., 1984, Global adaptive stabilization in the absence of information on the high frequency gain. *Proceedings of the Sixth International Conference on Analysis and Optimization of Systems* (New York: Springer-Verlag), pp. 49–57.