NASA-CR-193111

ADAPTIVE CONTROL STRATEGIES FOR FLEXIBLE ROBOTIC ARM

University of Colorado at Denver
1200 Larimer Street, Campus Box 110
Denver, Colorado 80217-3364

GRANT
IN-37-CR
167268
p- 37

CURRENT STATUS REPORT
July 15, 1992 to May 14, 1993

Principle Investigator:
Jan T. Bialasiewicz

Research Grant Number:
NAG-1-1444

ADAPTIVE CONTROL STRATEGIES FOR FLEXIBLE ROBOTIC ARM

Name and Address of Institution:

University of Colorado at Denver
1200 Larimer Street, Campus Box 110
Denver, Colorado 80217-3364

Type of Report:

Current Status Report
July 15, 1992 to May 14, 1993

Priciple Investigator:

Jan T. Bialasiewicz

Research Grant Number:

NAG-1-1444

NEURAL SELF-TUNING ADAPTIVE CONTROL STRATEGIES OF NON-MINIMUM PHASE SYSTEM DEVELOPED FOR FLEXIBLE ROBOTIC ARM

By

Long T. Ho

Table of Content

Abstract

The motivation of this research came about when a neural network direct adaptive control schemes was applied to control the tip position of a flexible robotic arm. Satisfactory control performance was not attainable due to the inherent non-minimum phase characteristics of the flexible robotic arm tip. Most of the existing neural network control algorithms are based on the direct method and exhibit very high sensitivity if not unstable closed-loop behavior. Therefore a neural self-tuning control (NSTC) algorithm is developed and applied to this problem and showed promising results. Simulation results of the NSTC scheme and the conventional self-tuning (STR) control scheme are used to examine performance factors such as control tracking mean square error, estimation mean square error, transient response, and steady state response.

1. Introduction

Self-tuning adaptive control used for controlling unknown ARMA plants has traditionally been based on the minimum variance control law and a recursive identification algorithm (Astrom and Wittenmark, 1973; Clark and Gawthrop, 1979). Although the advancement in VLSI has made it more possible to implement real-time recursive algorithms but it is still computationally intensive and expensive due to the recursive nature of the algorithm. On the other hand, neural networks VLSI has been made available commercially with extreme processing capability due to its parallel architecture. With this in mind the possibility of formulating neural networks to perform functions of conventional recursive algorithms becomes important. Hence, in this paper we propose the neural self tuning control (NSTC) scheme where the implicit identification is performed by a multilayer neural network (MNN) and the control is based on the generalized minimum variance (GMV) control law.

Neural networks have undoubtedly demonstrated its effectiveness in controlling nonlinear systems with known/unknown dynamics and uncertainties (Narendra and Parthasrathy, 1990; Levin and Narendra, 1993; Werbos et al. 1990; Hunt et al., 1992). In addition, neural network adaptive control algorithms have also been developed for specific linear system model such as the state space model (Ho et al., 91a) and the ARMA model (Ho et al., 1991b). It was shown in the simulation results that neural network controllers produced comparable results to conventional adaptive controllers. In this paper, we investigate the performance of the NSTC and compare it to the conventional adaptive STR.

The flexible arm to be controlled is shown in Figure 1.1. There are two system outputs that are of interest, one is the hub angle $\theta_h(t)$ and the other is the tip angle $\theta_t(t)$ of the arm. The goal is to apply a neural network control scheme to control these outputs to track the command signals. The neural controller will generate a control voltage signal u(t) that will feed the power amplifier in which will force current through the motor and cause the arm position to react. The dynamical transfer function of the hub angle is a linear minimum phase system in which will be shown readily controllable by a neural network. In fact, the direct adaptive neural control scheme in Figure 1.2. can be used to control the hub. This control scheme belongs to the type called specialized learning control (Psaltis et al., 1988; Ho et al., 1991c). However, the tip of the arm, being in a different location from the actuator point, therefore making the system to be of the type non-collocated system. The effect of this dynamically is that there is a zero on the right half side of the s-plane. In other words, the transfer function of the tip angle is of the type non-minimum phase which presents itself to be very difficult to control when direct adaptive control methodology is applied. This difficulty may be due to the controller trying to emulate the inverse dynamics of the non-minimun plant and results in an unstable behavior. According to simulation studies the specialized learning control algorithm diverges when applied to control the tip angle. Most other neural control schemes are also based on the inverse dynamics including the indirect learning method by (Psaltis et al., 1988), the feedback error learning by (Kawato et al., 1988), and the methods presented by (Narendra and Parthasarathy, 1990).

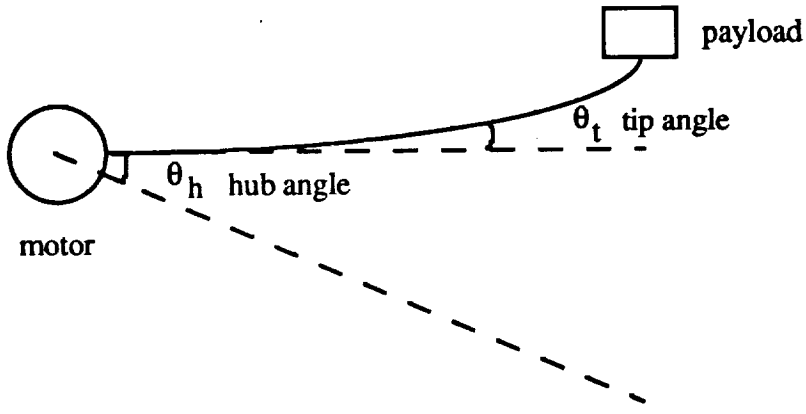

Figure 1.1. Flexible arm system

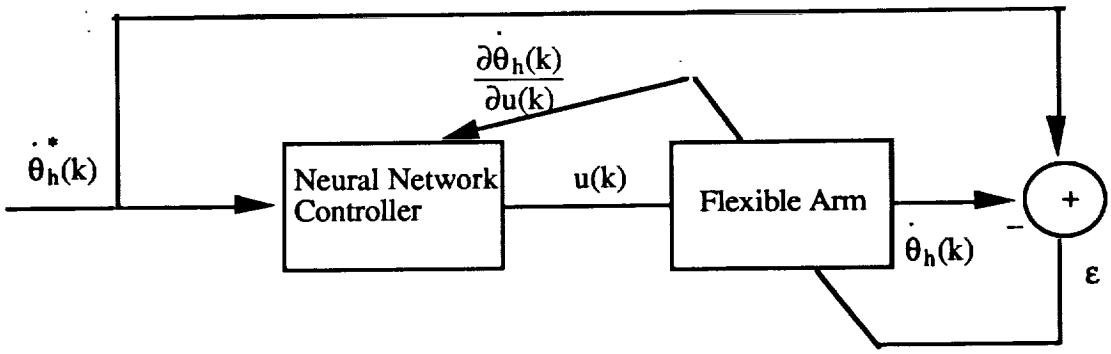

Figure 1.2. specialized learning control of hub velocity

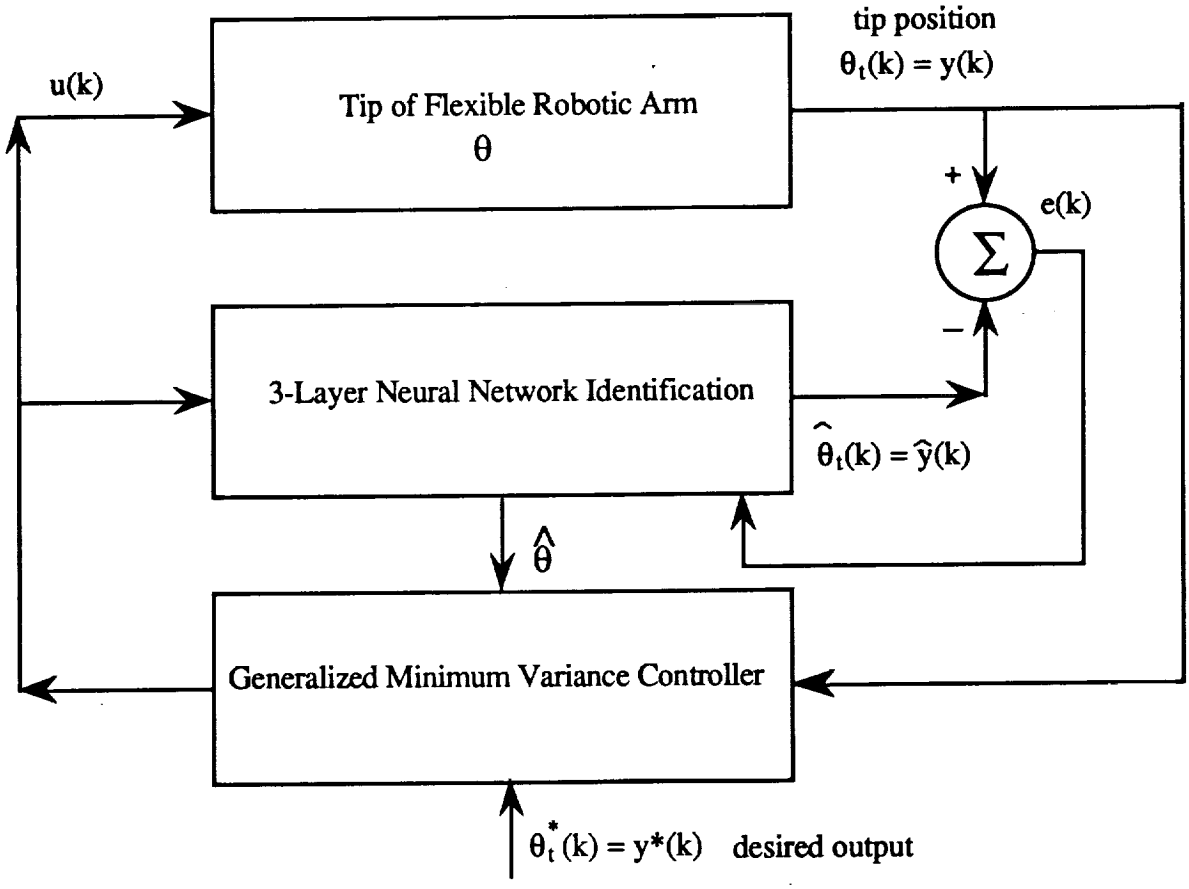

Figure 1.3 Indirect neural adaptive control scheme

In this report, we propose to use the neural self tuning control scheme which is based on an indirect control method (Ho et al., 1991c) to control the tip angle. This scheme is shown in Figure 1.3 where the identification is performed by the MNN and the control is performed by the generalized minimum variance (GMV) controller. The GMV control algorithm has a dynamic

weighting function $Q(q^{-1})$ applied to the plant control signal u(k) in the cost function to limit and condition the control energy. Thus, upon selecting the proper weighting function the controller can be input/output stable and effective in controlling the non-minimum phase plant. In section 2 the neural self-tuning control (NSTC) which consists of the minimum variance control algorithm and the neural identification is presented. Section 3 covers the basic dynamics of the flexible arm tip position. Section 4 presents a comparative simulation study of the adaptive STR scheme and the NSTC scheme. And section 5 gives the conclusion of the results found in this study and address the advantages and disadvantages of the neural control scheme used for treating linear system.

2. Stochastic neural self-tuning adaptive control (NSTC)

The NSTC consists of the minimum variance control law and the neural identification algorithm. The model assumed for the plant is of ARMA input/output type having the form

$$y(k) = q^{-d}\frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})}\xi(k) \tag{2.1}$$

where u(k), y(k), $\xi(k)$, and d are system input, output, uncertainty, and delay, respectively. A, B, and C are unknown system dynamics defined as

$$A(q^{-1}) = 1 + a_1q^{-1} + a_2q^{-2} + \ldots + a_{na}q^{-na} \tag{2.2}$$

$$B(q^{-1}) = b_0 + b_1q^{-1} + b_2q^{-2} + \ldots + b_{nb}q^{-nb} \tag{2.3}$$

$$C(q^{-1}) = 1 + c_1q^{-1} + c_2q^{-2} + \ldots + c_{nc}q^{-nc} \tag{2.4}$$

where q is the shift operator. For the above unknown plant, in Figure 1.3, the objective is to control its output to track a command signal y*(k) based on the generalized minimum variance control index (Clark and Gawthrop, 1979)

$$\begin{aligned} J(k+d) &= E\{\phi^2(k+d)\} \\ &= E\{[P(q^{-1})y(k+d)+Q(q^{-1})u(k)-R(q^{-1})y^*(k)]^2\} \\ &= E\{[\phi_y(k+d)+Q(q^{-1})u(k)-R(q^{-1})y^*(k)]^2\} \end{aligned} \tag{2.5}$$

where E is the expectation operator, $\phi_y(k+d)$ is the auxiliary output, and P,Q, and R are the weighting dynamics which can be chosen depending on the required response characteristics.

2.1. Generalized minimum variance control

In this section, the generalized minimum variance self-tuning control algorithm for the above problem statement is summarized (Clark and Gawthrop, 1979). To obtain the optimal control u(k) which minimizes the performance index (2.5), the predictive auxiliary output $\phi_y(k+d)$ in terms of the system dynamics must be determined. Consider the following identity

$$\frac{P(q^{-1})C(q^{-1})}{A(q^{-1})} = F(q^{-1}) + q^{-d}\frac{G(q^{-1})}{A(q^{-1})} \tag{2.1.1}$$

where the order of $F(q^{-1})$ and $G(q^{-1})$ are nf=d-1, ng=na-1, respectively. The output prediction can be shown to have the form

$$\phi_y(k+d) = \hat{\phi}_y(k+d) + \tilde{\phi}_y(k+d) \tag{2.1.2}$$

where

$$\begin{aligned}\hat{\phi}_y(k+d) &= C(q^{-1})^{-1}[G(q^{-1})y(k) + F(q^{-1})B(q^{-1})u(k)] \\ &= C(q^{-1})^{-1}[G(q^{-1})y(k) + E(q^{-1})u(k)]\end{aligned} \tag{2.1.3}$$

and

$$\tilde{\phi}_y(k+d) = F(q^{-1})\xi(k+d) \tag{2.1.4}$$

$\hat{\phi}_y(k+d)$ and $\tilde{\phi}_y(k+d)$ are the deterministic and uncorrelated random components of $\phi_y(k+d)$. Next, substituting (2.1.2) into (2.5), there results

$$J(k+d) = E\{[\hat{\phi}_y(k+d)+Q(q^{-1})u(k)-R(q^{-1})y^*(k)]^2\}+E[\tilde{\phi}_y(k+d)]^2\} \tag{2.1.5}$$

Since the second term in (2.1.5) is unpredictable random noise which is uncompensatable by the control input u(k), and the first term is a linear function of u(k), J(k+d) can be minimized by setting

$$[\hat{\phi}_y(k+d)+Q(q^{-1})u(k)-R(q^{-1})y^*(k)] = 0 \tag{2.1.6}$$

Solving for the generalized minimum variance (GMVC) control in (2.1.6) gives

$$u(k) = \frac{R(q^{-1})y^*(k)-\hat{\phi}_y(k+d)}{Q(q^{-1})} \tag{2.1.7}$$

using (2.1.3), (2.1.7) can also be written as

$$u(k) = \frac{C(q^{-1})R(q^{-1})y^*(k)-G(q^{-1})y(k)}{E(q^{-1})+C(q^{-1})Q(q^{-1})} \tag{2.1.8}$$

Remarks : Recall that $E(q^{-1})$ is equal to $F(q^{-1})B(q^{-1})$ where $B(q^{-1})$ contains the zeros of the plant. Notice that having the weighting function $Q(q^{-1})$ additive to $E(q^{-1})$ in (2.1.8) gives the designer the ability to alter the poles of the controller. Thus with a non-minimum phase plant $B(q^{-1})$ shall have unstable roots and proper selection of $Q(q^{-1})$ in (2.1.8) can assure the control signal u(k) to be bounded.

2.2. Neural system identification

In this section, a stochastic neural identification algorithm is developed for the self-tuning control scheme in Figure 1.3. Recall the predicted auxiliary output in (2.1.3) which can also be written as

$$\begin{aligned}\phi_y(k+d) &= C(q^{-1})^{-1}[G(q^{-1})y(k) + E(q^{-1})u(k)] + F(q^{-1})\xi(k+d) \\ &= C(q^{-1})^{-1}[G(q^{-1})y(k) + E(q^{-1})u(k)] + v(k)\end{aligned} \tag{2.2.1}$$

where the uncorrelated noise sequence $F(q^{-1})\xi(k+d)$ is replaced by v(k). Also (2.2.1) can be written as

$$\phi_y(k+d) = \sum_{i=0}^{ng} g_i y(k-i) + \sum_{i=0}^{ne} e_i u(k-i) - \sum_{i=1}^{nc} c_i \phi_y(k+d-i) + v(k) \tag{2.2.2}$$

$$\phi_y(k+d) = \psi'(k)\theta(k)+ v(k) \tag{2.2.3}$$

where

$$\psi'(k) = [y(k)...y(k-ng);\ u(k)...u(k-ne);\ \phi_y(k+d-1)...\phi_y(k+d-nc)] \tag{2.2.4}$$

$$\theta'(k) = [g_0\ g_1 \cdots g_{ng};\ e_0\ e_1 \cdots e_{ne};\ -c_1\ -c_2 \cdots -c_{nc}] \tag{2.2.5}$$

since the parameter vector θ is unknown, the estimated form of $\phi_y(k+d)$ is given as

$$\hat{\phi}_y(k+d) = \hat{\psi}'(k)\hat{\theta}(k) \tag{2.2.6}$$

where

$$\hat{\psi}'(k) = [y(k)...y(k-ng);\ u(k)...u(k-ne);\ \hat{\phi}_y(k+d-1)...\hat{\phi}(k+d-nc)] \tag{2.2.7}$$

$$\hat{\theta}'(k) = [\hat{g}_0\ \hat{g}_1 \cdots \hat{g}_{ng};\ \hat{e}_0\ \hat{e}_1 \cdots \hat{e}_{ne};\ -\hat{c}_1\ -\hat{c}_2 \cdots -\hat{c}_{nc}] \tag{2.2.8}$$

The unknown parameter vector in (2.2.8) (Figure 2.1), is taken from the output of the neural network

$$\begin{aligned}\hat{\theta}(k) &= [\hat{\theta}_1(k)\ \hat{\theta}_2(k) \cdots \hat{\theta}_j(k) \cdots \hat{\theta}_{n3}(k)]' \\ &= [O_1(k)\ O_2(k) \cdots O_j(k) \cdots O_{n3}(k)]'\end{aligned} \tag{2.2.9}$$

Where n3 is the number of neurons at the output layer. Consider the system identification cost function

$$\begin{aligned}V(k) &= \frac{1}{2} E\{\varepsilon'(k)\Lambda^{-1}(k)\varepsilon(k)\} \\ &= \frac{1}{2} E\{[\phi_y(k)-\hat{\phi}_y(k)]'\Lambda^{-1}(k)[\phi_y(k)-\hat{\phi}_y(k)]\}\end{aligned} \tag{2.2.10}$$

where $\Lambda(k)$ is a symmetric positive definite weighting matrix, and $V(k)$ is minimized by adjusting the weights of the neural identifier.

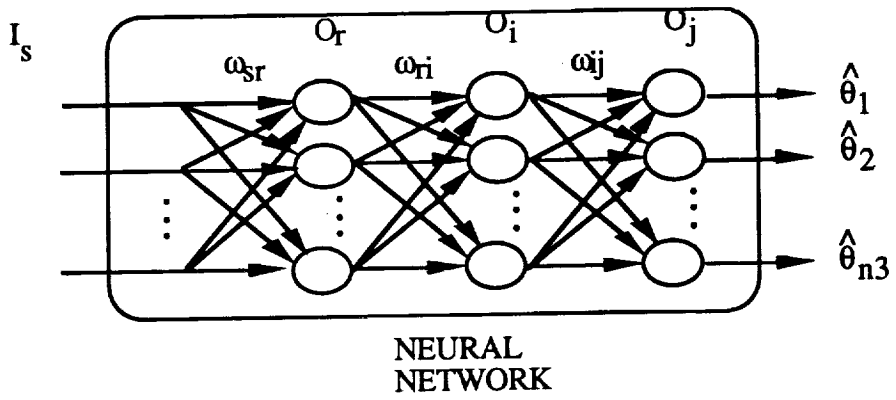

Figure 2.1. Neural network structure

In Figure 2.1, the weights connecting the second layer to the output layer, using the gradient search (Rumelhart and McClelland, 1987), can be updated as

$$\omega_{ij}(k+1) = \omega_{ij}(k) + \Delta\omega_{ij}(k) \tag{2.2.11}$$

where

$$\begin{aligned}\Delta\omega_{ij}(k) &= -\eta\frac{\partial}{\partial\omega_{ij}(k)}\{\frac{1}{2}\varepsilon'(k)\Lambda^{-1}(k)\varepsilon(k)\} \\ &= -\eta\frac{\partial}{\partial\omega_{ij}(k)}\{\frac{1}{2}[\phi_y(k)-\hat{\phi}_y(k)]'\Lambda^{-1}(k)[\phi_y(k)-\hat{\phi}_y(k)] \\ &= \eta\frac{\partial\hat{\theta}'(k)}{\partial\omega_{ij}(k)}\frac{\partial\hat{\phi}_y'(k)}{\partial\hat{\theta}(k)}\Lambda^{-1}(k)[\phi_y(k)-\hat{\phi}_y(k)]\end{aligned} \tag{2.2.12}$$

with η being the search step size. Consider the derivative of $\hat{y}(k)$ with respect to $\hat{\theta}(k)$ in (2.2.12)

$$\frac{\partial\hat{\phi}_y'(k)}{\partial\hat{\theta}(k)} = \hat{\psi}(k-d) \tag{2.2.13}$$

In (2.2.13) we have assumed that $\theta(k) \sim \theta(k-d)$, that is, θ is slowly time varying with respect to the delay time d. The other partial derivative in (2.2.12) can be determined as

$$\frac{\partial\hat{\theta}'(k)}{\partial\omega_{ij}(k)} = O_i(k)e_j\frac{\partial[f(Net_j(k))]'}{\partial Net_j(k)} \tag{2.2.14}$$

where f(.) is the sigmoidal activation function, $O_i(k)$ is the output of the second layer, and

$$Net_j(k) = [net_1\ net_2\ ...\ net_j\ ...\ net_{n3}]' \tag{2.2.15}$$

with

$$net_j(k) = \sum_{i=1}^{n2} \omega_{ij}(k) O_i(k)$$

where n2 is the number of neurons of the second hidden layer as shown in Figure 2.1. Also ej in (2.2.14) is defined as

$$e_j = [0...0\ 1\ 0...0] \tag{2.2.16}$$

with the j-th element in e_j being 1, and other elements are 0. Thus, substituting (2.2.14) back into (2.2.12) gives

$$\Delta\omega_{ij}(k) = \eta e_j \delta_j(k) O_i(k) \tag{2.2.17}$$

where

$$\delta_j(k) = \frac{\partial[f(Net_j(k))]'}{\partial Net_j(k)} \frac{\partial \hat{\phi}_y'(k)}{\partial \hat{\theta}(k)} \Lambda^{-1}(k)[\phi_y(k) - \hat{\phi}_y(k)] \tag{2.2.18}$$

Next, the weights connecting the first to the second layer, in Figure 2.1, can be updated by the recursive equation

$$\omega_{ri}(k+1) = \omega_{ri}(k) + \Delta\omega_{ri}(k) \tag{2.2.19}$$

where

$$\Delta\omega_{ri}(k) = -\eta \frac{\partial}{\partial \omega_{ri}(k)} \{\frac{1}{2} \varepsilon'(k) \Lambda^{-1}(k) \varepsilon(k)\} \tag{2.2.20}$$

using the similar back propagation approach, (2.2.20) can be shown to result in the following form

$$\Delta\omega_{ri}(k) = \eta \delta_i(k) O_r(k) \tag{2.2.21}$$

where O_r is the output of the first layer and

$$\delta_i(k) = [\omega_{i1}...\omega_{ij}...\omega_{in3}] \frac{\partial f[net_i(k)]}{\partial net_i(k)} \delta_j(k) \tag{2.2.22}$$

Lastly, the weights connecting the input to the first layer, in Figure 2.1, can be updated by the recursive equation

$$\omega_{sr}(k+1) = \omega_{sr}(k) + \Delta\omega_{sr}(k) \tag{2.2.23}$$

where

$$\Delta\omega_{sr}(k) = -\eta\frac{\partial}{\partial\omega_{sr}(k)}\{\frac{1}{2}\varepsilon'(k)\Lambda^{-1}(k)\varepsilon(k)\} \tag{2.2.24}$$

Again, using the back propagation approach, (2.2.24) can be determined as

$$\Delta\omega_{sr}(k) = \eta\delta_r(k)I_s(k) \tag{2.2.25}$$

where $I_s(k)$ is the input from the delay network and

$$\delta_r(k) = [\omega_{r1}...\omega_{ri}...\omega_{rn2}]\frac{\partial f(net_r(k))}{\partial net_r(k)}\frac{\partial[f(Net_i(k))]'}{\partial Net_i(k)}\frac{\partial Net_j(k)'}{\partial[f(Net_i(k))]'}\delta_j(k) \tag{2.2.26}$$

with $Net_i(k)$ being defined similarly as $Net_j(k)$ in (2.2.14). By adjusting the weights $\omega_{ij}(k)$, $\omega_{ri}(k)$, and $\omega_{sr}(k)$ with the above algorithm, the unknown implicit plant's parameters can be identified and obtained at the output of the neural identifier, as shown in Figure 2.1. Once the estimate of θ is available, $\hat{\phi}_y(k+d)$ in (2.2.6) can be computed, and then the control signal can be generated using (2.1.7) as

$$u(k) = \frac{R(q^{-1})y^*(k)-\hat{\phi}_y(k+d)}{Q(q^{-1})} \tag{2.2.27}$$

3. Flexible arm tip position dynamics

This section describes the components and the control model of the flexible arm tip. A detailed discussion of the dynamics of flexible arm tip and hub can be found in (Fraser and Daniel, 1991). In order to control the flexible robotic arm shown in Figure 1.1, it is required that the control action produced by the control program running on a processor board is converted to a voltage by the D/A board and forms the input to the power amplifier of the motor. The output of the power amplifier is a motor current directly proportional to the input voltage. The motor then converts this current to a torque to drive the arm. The resulting motion of the arm is detected by the various sensors and fed back to the controller.

The adaptive control algorithm design does not require the complete knowledge of the plant dynamics. However, for the purpose of simulation study, the transfer function model of the plant needs to be known. This model must incorporate not only the behavior of the flexible arm itself but also the power amplifier, the motor and the output sensors. In a servo system, the power amplifier and the sensors usually have a much higher bandwidth than that of the motor and load therefore they can be approximated as a constant. The general transfer function of the flexible arm tip is

$$\frac{\theta_t(s)}{u(s)} = \frac{K_A K_T}{s(s+c_0)} \prod_{i=1}^{\infty} \frac{(1-s^2/\alpha_{ti}^2)}{(1+2\zeta_i s/\omega_i + s^2/\omega_i^2)} \qquad (3.1)$$

where the physical interpretation of the above equation is as follows:

First, poles and zeros of the system is depicted in Figure 3.1 below

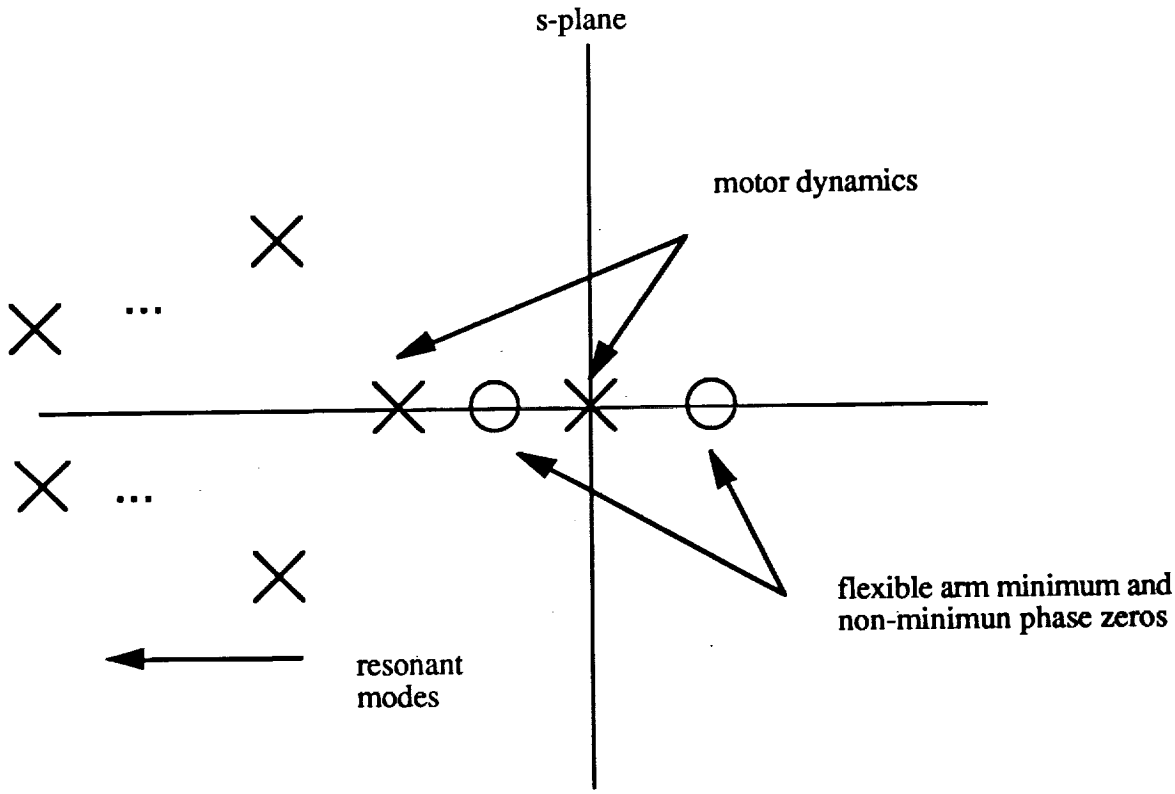

Figure 3.1 Pole-zero diagram of flexible arm tip

The above diagram shows the three constituting dynamic components of the plant which are the motor, the resonant modes of the flexible arm, and the arm non-minimum phase characteristics. The dynamics of the servo motor system is represented by the term

$$\frac{K_A K_t}{(S+C_o)s} \tag{3.2}$$

where K_T is the motor torque constant, K_A represents the power amplifier and sensor gain, and C_o represents the back emf and viscous damping effects know as the mechanical time constant. The motor can be seen as a series of subcomponent connected in series as shown in Figure 3.2.

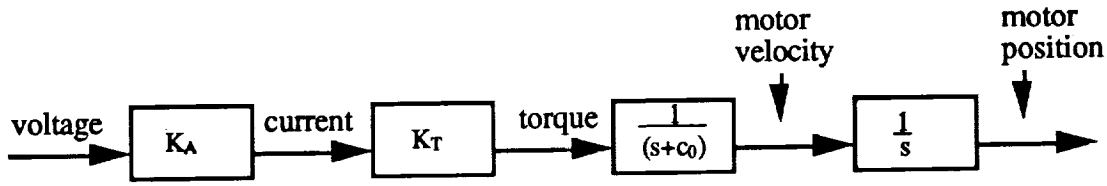

Figure 3.2 Servo motor system components

Next, the the flexible arm attached to the motor shaft is describe by the term.

$$\prod_{i=1}^{\infty} \frac{(1-s^2/\alpha_{ti}^2)}{(1+2\zeta_i s/\omega_i+s^2/\omega_i^2)} \tag{3.3}$$

Here, the denominator of (3.3) represents the set of flexible resonant modes of the arm. Each flexible mode is associated with the corresponding damping ζ_i at a frequency ω_i. In theory, there is an infinite number of flexible modes, but in practice only the sufficiently low frequency modes will be noticeable by the control system. This is because a real system is always band-limited therefore most of the modes are attenuated by the low-pass frequency behavior. Also, the frequency range of operation can be limited to be below the major dominant resonant mode so that oscillation will not be present in the system response. If higher frequency range of operation is desired, the dominate resonant modes can be notch filtered out provided their damping ζ_i's and frequencies ω_i's are determinable.

Consider the physical properties of the flexible arm and the servo system given in Table 3.1. Based on these parameters the transfer function was derived and measured by experiment (Fraser and Daniel, 1991). Both results agreed as shown in Table 3.2. The five resonant modes occupy the frequency range from 86 rad/sec to 1445 rad/sec. The frequency response of this system was simulated and is shown in Figure 3.2. The peaks represent the resonant energy at the specific frequencies. Also notice that the energy of the modes lessens are the frequency increases.

Table 3.1.

Physical properties of arm and motor

effective beam length (m)	0.386
beam thickness (mm)	0.956
beam width	0.03
mass/unit length of arm m (kg/m)	0.222
flexural rigidity of beam(Nm^2)	0.426
hub moment of inertia (kg m^2)	0.00009
radius of hub (m)	0.034
Tip mass for loaded arm (kg)	0.065
tip inertia for load arm (kg m^2)	0.000005
continous torque at rated speed (Nm)	0.177
pulse torque(Nm)	2.913
rated voltage (V)	24
torque constant (Nm/A)	0.048
total inertia (kg m^2)	0.000041
Ka*Kt	3.6
Co (rad/sec)	0.16

Table 3.2.

	POLES (rad/sec)		ZEROS (rad/sec)	
Mode	Expmt.	Theory	Expmt.	Theory
1	86.1	86.9	48.4	47
2	297.6	285.3	-48.4	-47
3	603.2	601.9	---	---
4	1011.6	1065.0	---	---
5	1445.1	1658.8	---	---

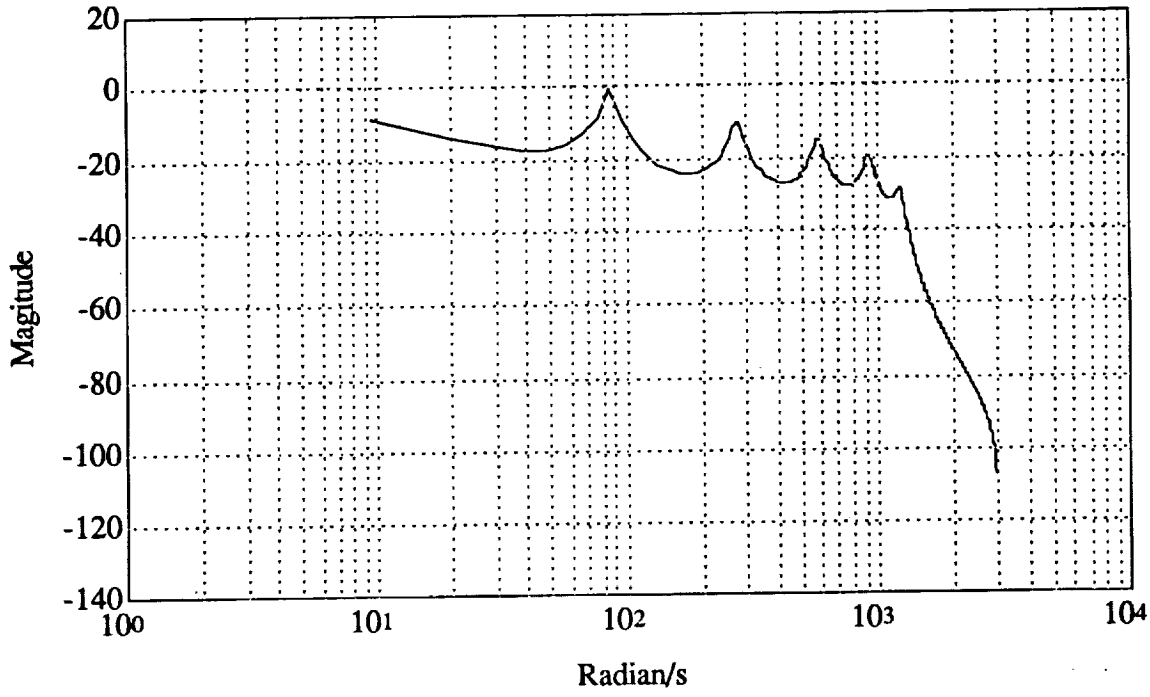

Figure 3.2 Frequency magnitude response of arm tip with five resonant modes

For easy controllability it is desirable to filter out these resonance modes. Therefore, a notch filter is designed to notch out the first resonance mode and a low pass filter is used to filter out the rest of the resonance mode. Figure 3.3 shows a block diagram of the filtering process. The resulting frequency ideal response is shown in Figure 3.4.

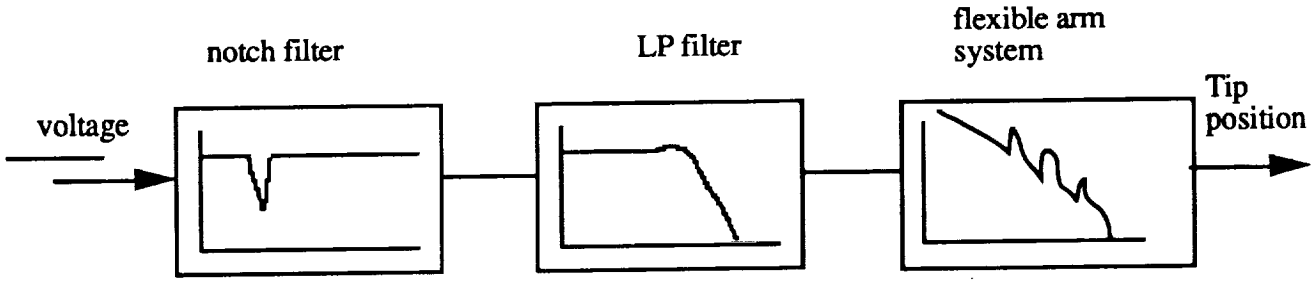

Figure 3.3 . Frequency response of open-loop components

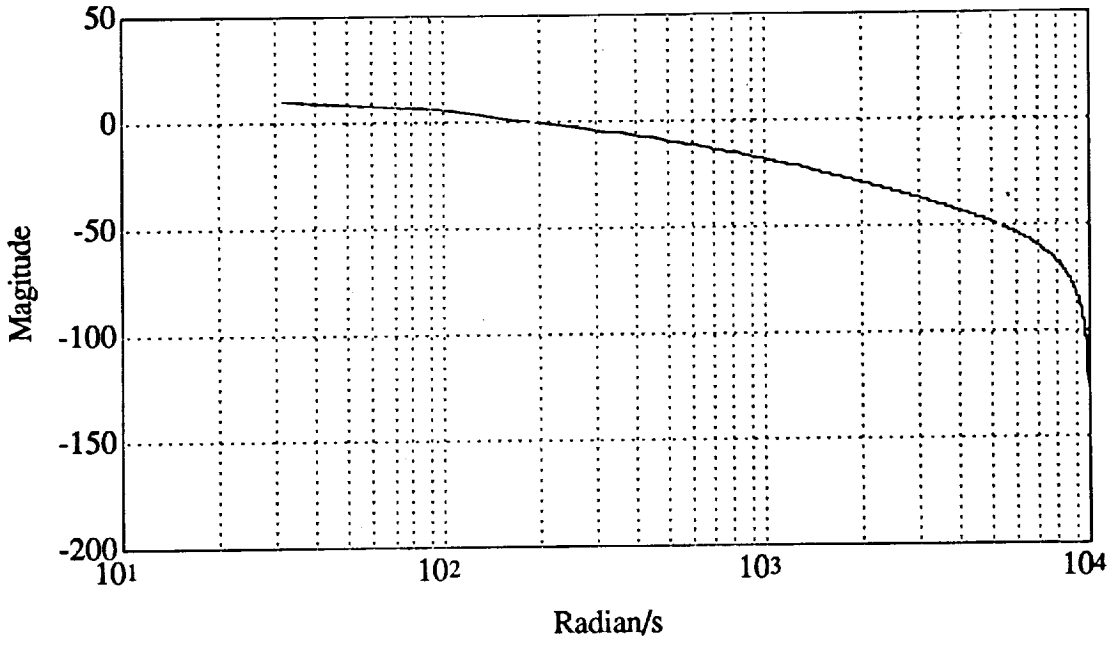

Figure 3.4 Frequency response of the aggregate filtered open-loop

Since we are primarily interested in learning the controllability and behavior of the non-minimum phase characteristics of the plant, we can simplify the arm tip transfer function to have the form

$$\frac{\theta_t(s)}{u(s)} = \frac{K_A K_T(1-s^2/\alpha^2)}{s(s+c_0)} \qquad (3.4)$$

Lastly, the non-minimum characteristics of the arm tip is describe in (3.1) and (3.4) by the numerator term.

$$(1-s^2/\alpha^2)$$

This is due to the fact that the control system sensing and actuation do not take place at the same location and therefore being a non-collocatted system. It should be mentioned that the non-minimum phase characteristics is very difficult for the neural network to control (since most neural network adaptive control schemes are based on the direct method).

4. Empirical studies

In this section we examine some simulation results of the direct and indirect neural control schemes for controlling the flexible arm hub and tip. We will show that the hub having a well behaved linear transfer function produced very satisfactory controlled response. We also attempted to use the direct adaptive control scheme to control the tip velocity and found unstable response even after numerous controller parameter changes. Next, the NSTC scheme in section 2 was applied to control the tip position and produced encouraging results. Lastly, the neural identifier in the NSTC algorithm is compared with the recursive least square identifier and show faster convergent rate.

4.1. Neural direct adaptive control of arm hub and tip

The neural direct adaptive control scheme was first introduced by (Psaltis et al., 1988) and was later reformulated for nonlinear/linear state space system by (Ho et al., 1991c). We will apply this scheme, shown in Figure 4.1. to control the hub velocity of the arm.

The dynamic transfer function of the hub is a linear minimum phase system. The numerical transfer function found in (Fraser and Daniel, 1991) is

$$\frac{\theta_h(s)}{U(s)} = \frac{10.2\,(1+\frac{s^2}{32.7^2})}{(s+0.57)(s+2000)} \qquad (4.1)$$

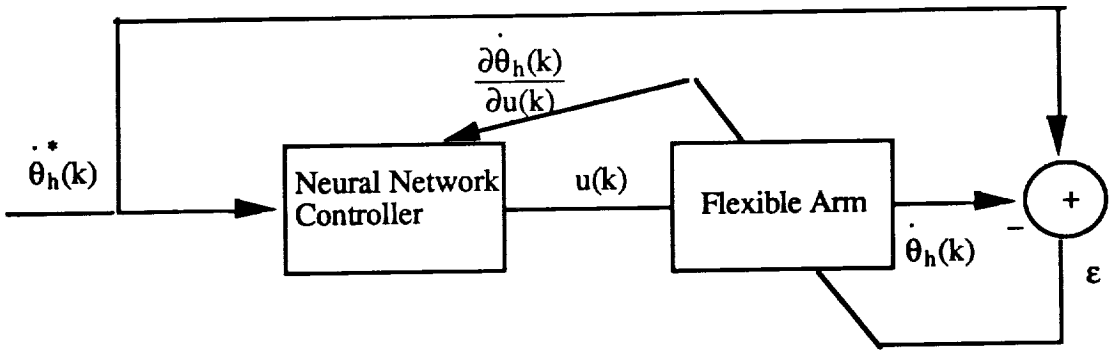

Figure 4.1. Neural direct control scheme of hub velocity

where the resonant modes are assumed to be filtered out. In the simulation process , the model in (4.1) was first discretized and then converted to state space form

$$x(k+1) = Ax(k) + Bu(k) \tag{4.2}$$
$$\dot{\theta}_h(s) = Cx(k)$$

When using this scheme (Figure 4.1.) there is a priori information that is needed and that is the jacobian of the plant $\frac{\partial \dot{\theta}_h(k)}{\partial u(k)}$. This term was computed based on the discretized model and resulted as

$$\frac{\partial \dot{\theta}_h(k)}{\partial u(k)} = CB \tag{4.3}$$

Information on the neural network algorithm is refered to (Ho et al., 1991c).

Remarks: : The hub position was not suitable for this specialized learning control scheme because the jacobian turns out to be near zero. Therefore the velocity is the selected controlled variable and an additional outer control loop may be incorporated to achieve position control. This outer loop will have a velocity profile generator which resembles to a proportional controller with saturation (Franklin and Powell, 1981).

Simulation: A smoothed square wave command was presented to the control system, after 50 iterations (about .3 seconds, sampling period was 6 ms) the hub had tracked the command signal as shown in Figure 4.2 where the solid line is the desired response and the dashed line is the actual response.

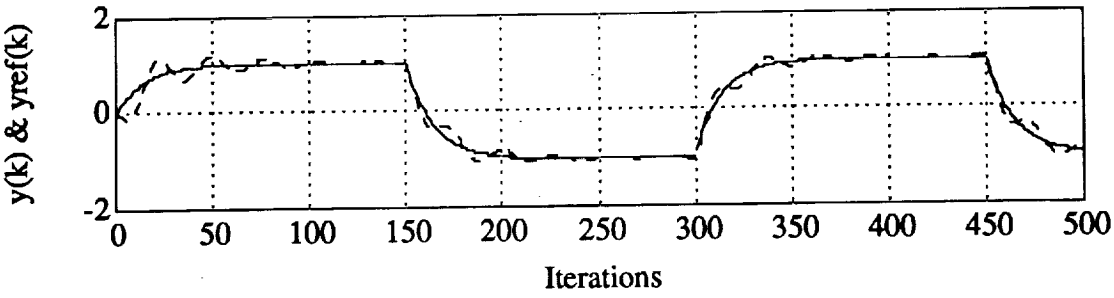

Figure 4.2. Hub velocity response: $\dot{\theta}_t^*(k)$ & $\dot{\theta}_t(k)$

This trackability is reflected in the mean square tracking error shown in Figure 4.3. Notice that the convergent time in control application is serveral orders of magnitude faster than other applications. In this case it took only 50 iterations for the 2-layer neural network to be maturely trained with initial random weights. This fast convergent time makes it very practical for real-time control implementation.

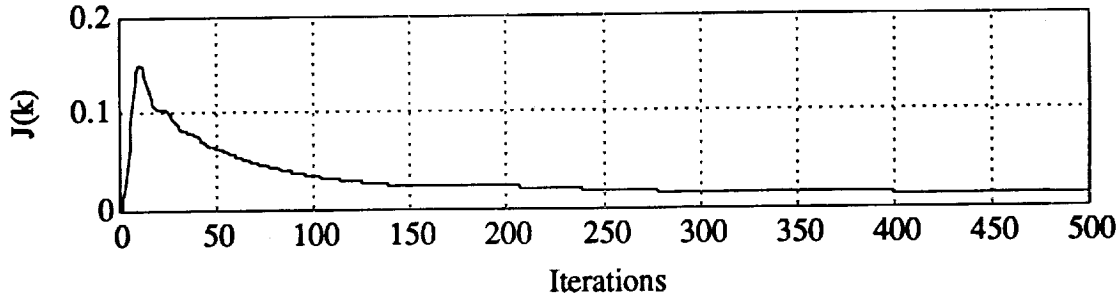

Figure 4.3. Control tracking MSE response

Next, the same scheme is applied to control the tip velocity. The numerical tip transfer function (based on the flexible arm and motor properties in Tables 3.1 and 3.2) is given in (Fraser and Daniel, 1991) as

$$\frac{\theta_t(s)}{U(s)} = \frac{3.6\ (1+\frac{s^2}{48.4^2})}{s(s+0.16)} \qquad (4.4)$$

Here again, we are primarily interested in the non-minimum phase characteristics and therefore assumed that the resonant modes are filtered out.

Simulation: After numerous attempts to vary the neural network parameters, an unstable closed-loop response was prevalent as shown in Figures 4.4 and 4.5. This is due to the fact that the neural network in Figure 4.1. trying to emulate the inverse dynamics of the plant (4.4.) and in effect produced an unstable pole behavior. Note in Figure 4.4. that the command signal is small compared to the plant diverging output response therefore it looks like a straight line.

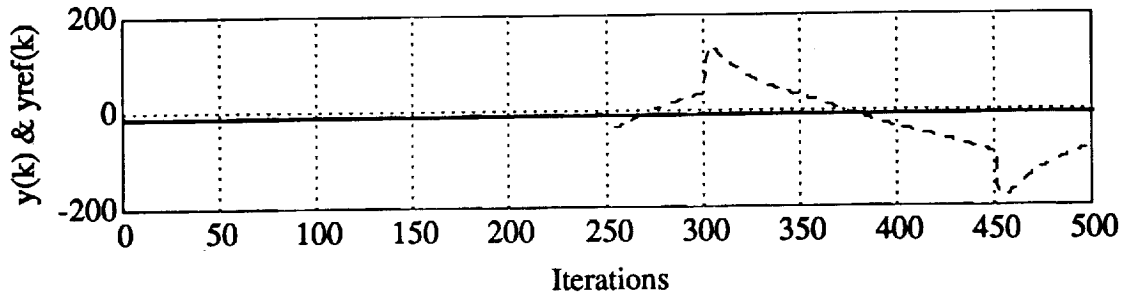

Figure 4.4. Unstable response of tip control: $\theta_t^*(k)$ and $\theta_t(k)$

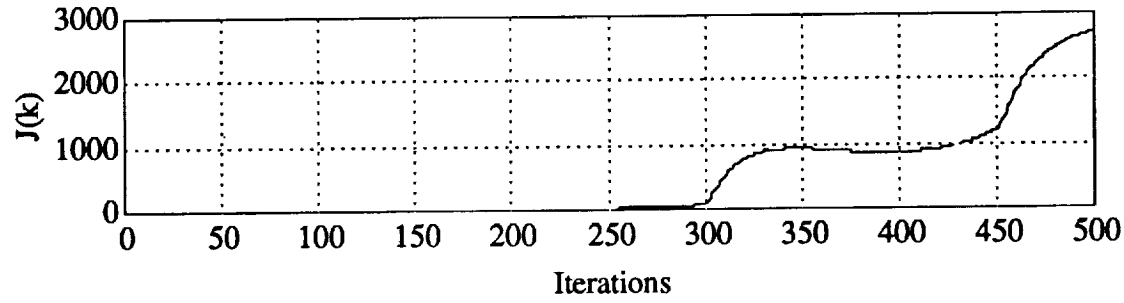

Figure 4.5. Diverging tracking MSE of tip velocity

Neural Network: The $N^2_{5,10,1}$ neural network used in this scheme consists of one input layer, one hidden layer, and one output layer with the number of neurons as 5, 10, and 1, respectively. Also at the input of the neural was the desired response vector [y*(k) y*(k-1) y*(k-2) y*(k-3) y*(k-4)]^T. The parameters of the sigmoidal activation function at the output node was found to be most influential on the tracking error convergent rate. Predominantly the slope of the activation function was observed to be proportional to the convergent rate. Also the bipolar sigmoidal saturation levels of the output neuron needed to be set equal to or greater than the maximum allowable plant input. The tuning of the sigmoidal functions was done manually by trial and error, typically for linear system like that of the hub, it takes very few tweaks (around 1 or 2) before the tracking results was achieved. Auto-tuning of the sigmoidal function parameters can also be applied to obtain statiscally better results (Yamada and Yabuta, 1992; Proano, 1989).

4.2. Neural self-tuning adaptive control (NSTC) of tip position

In section 4.1. we showed by simulation that the direct neural adaptive control scheme was unable to control the tip position (Figures 4.4 and 4.5). In fact, this was why the NSTC algorithm was developed. Recall that this scheme has two distinct functions, identification and control, which are done by the neural network and the (GMV) control, respectively. The NSTC scheme is shown again in Figure 4.6.

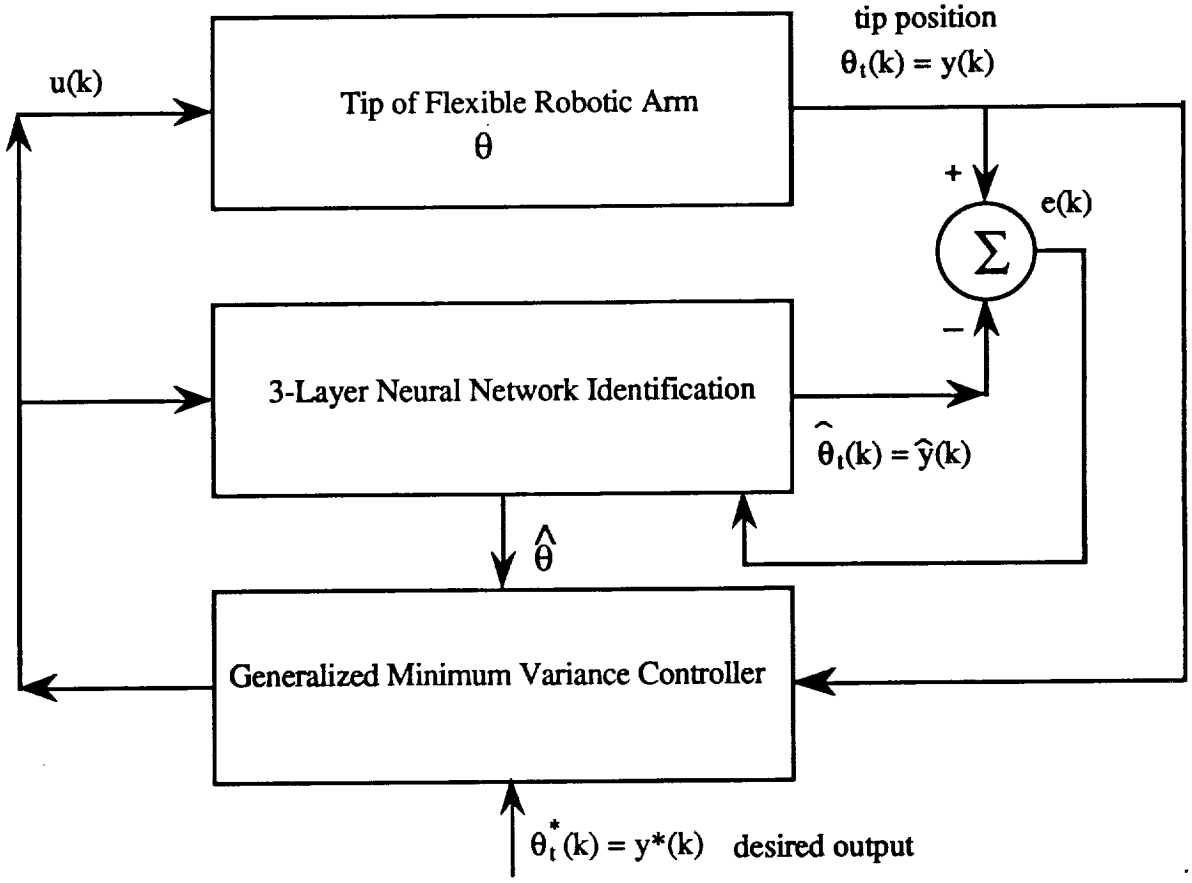

Figure 4.6. NSTC scheme block diagram

In this section we perform the simulations of two schemes which are: The adaptive STR using recursive least square identification, and the NSTC using the neural identification. This is so that a comparative study can be done to assess the performance of the developed NSTC.

Simulation: The model of the tip position is the discretized model of (4.4). Recall the control index defined in section 2

$$J(k+d) = E\{\phi^2(k+d)\}$$

$$= E\{[P(q^{-1})y(k+d)+Q(q^{-1})u(k)-R(q^{-1})y^*(k)]^2\} \quad (4.5)$$

where the weighting functions were chosen as

$$P(q^{-1})=1;\ Q(q^{-1})=.1+.06q^{-1};\ R(q^{-1})=1 \quad (4.6)$$

and the desired hub position $\theta_t^*(k)$ was a step command. Beginning with Figure 4.7. showing the desired step tip response, the controlled tip response based on the adaptive STR and the tip response from the NSTC. Obviously both controllers manage to track the command signal. However, the NSTC seems to have a slower settling time. Figure 4.8. shows the converging tracking control index (2.1.5) where both schemes seem very comparable to each other. Figure 4.9. displays the comparable control energy produced by these controllers. Note that the transient control energy was affected by two factors: one is the initial condition of the estimated parameter vector $\hat{\theta}_0$ (which was set as $\hat{\theta}_0$ = [1 1 ... 1]' for both control schemes), the further $\hat{\theta}_0$ is away from the optimum $\hat{\theta}^*$ in the parameter state space, the longer the convergence of the tracking control index (2.1.5). The other factor is the selection of the input weighting function $Q(q^{-1})$ which has the effect of limiting the control energy with the tradeoff of slower tracking convergence. Lastly, we compare the recursive least square identification with the neural network identification. The two identifiers estimate the parameter vector θ in (2.2.5) so that the predictive output term $\hat{\phi}_y(k+d)$ in (2.2.2) can be computed. Figure 4.10. shows the estimation cost function V(k) in (2.2.10) response of the RLS and the neural network. V(k) of the RLS has a slightly faster convergence than the neural network but not by a significant degree. Again, this indicates that the identification performance of the two algorithms are comparable to each other. For completeness, the time response of the true output $\theta_t(k)$ and the estimated output $\hat{\theta}_t(k)$ produced by the neural network is shown in Figure 4.11.

Neural Network: The three layer neural network $N^3_{2,5,15,P0}$ used in this scheme consists of one input layer, two hidden layers, and one output layer with the number of neurons as 2, 5, 15, and P0, respectively. P0 is the length of the vector defined in (2.2.8) which is (ng+1)+(ne+1)+nc, and is 11 for the case of the arm tip plant. The input of the neural network was a selected as constant vector Is = [1 1]' because it was desired that the output of the neural network to be correlated to the

its input. The parameters of the sigmoidal activation function at the output node was found to be most influential on the tracking error convergent rate. Predominantly the slope of the activation function was observed to be proportional to the estimation convergent rate V(k). Also the bipolar sigmoidal saturation levels of the output neuron needed to be set equal to or greater than the maximum component of the parameter vector θ. The tuning of the sigmoidal functions was done manually by trial and error. Autotuning of the sigmoidal function parameters can also be applied to obtain statiscally better results (Yamada and Yabuta, 1992; Proano, 1989). However, the optimal dimension of the neural network in terms of number of layers and nodes was not known and therefore an initial pick of $N^3_{2,5,15,P0}$ was used throughout the simulation.

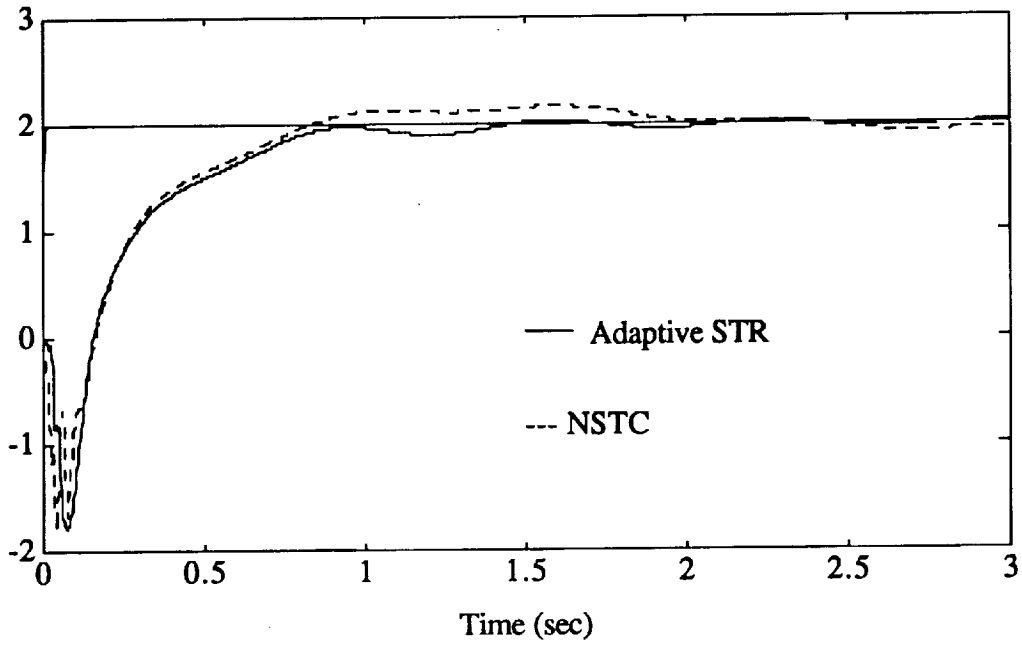

Figure 4.7. Tip position response: $\theta_t^*(k)$ & $\theta_t(k)$ of the adaptive STR and the NSTC

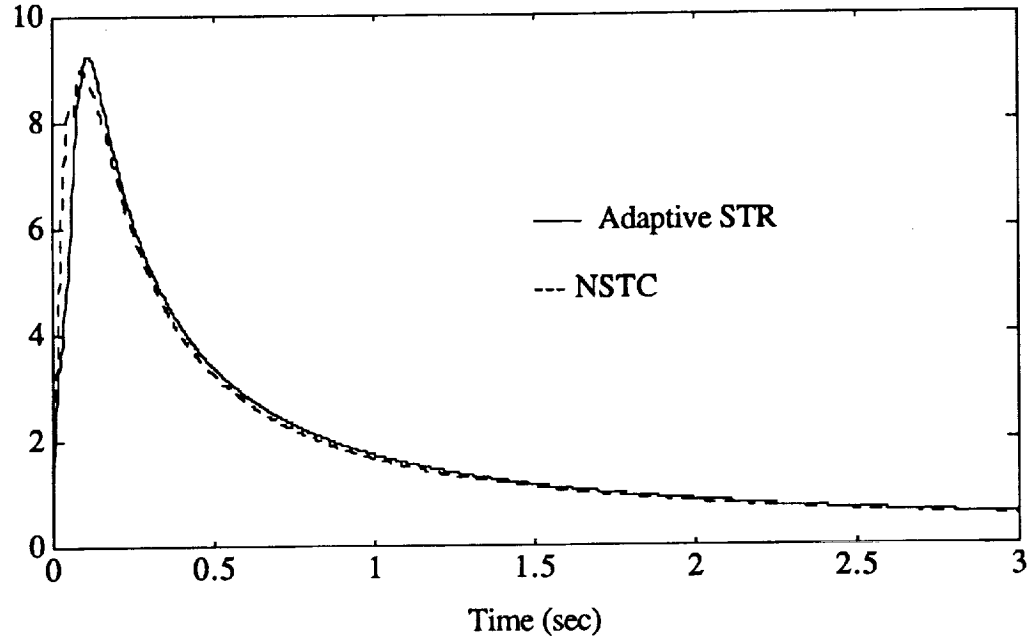

Figure 4.8. Control performance index J(k) of the adaptive STR and the NSTC

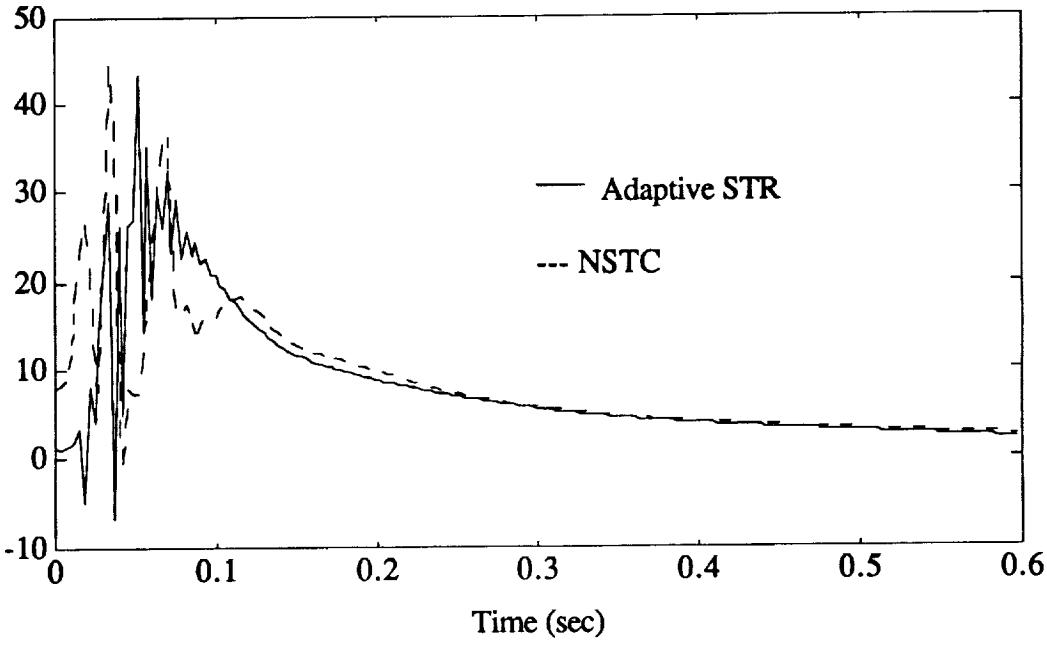

Figure 4.9. Control signal u(k) of the adaptive STR and the NSTC

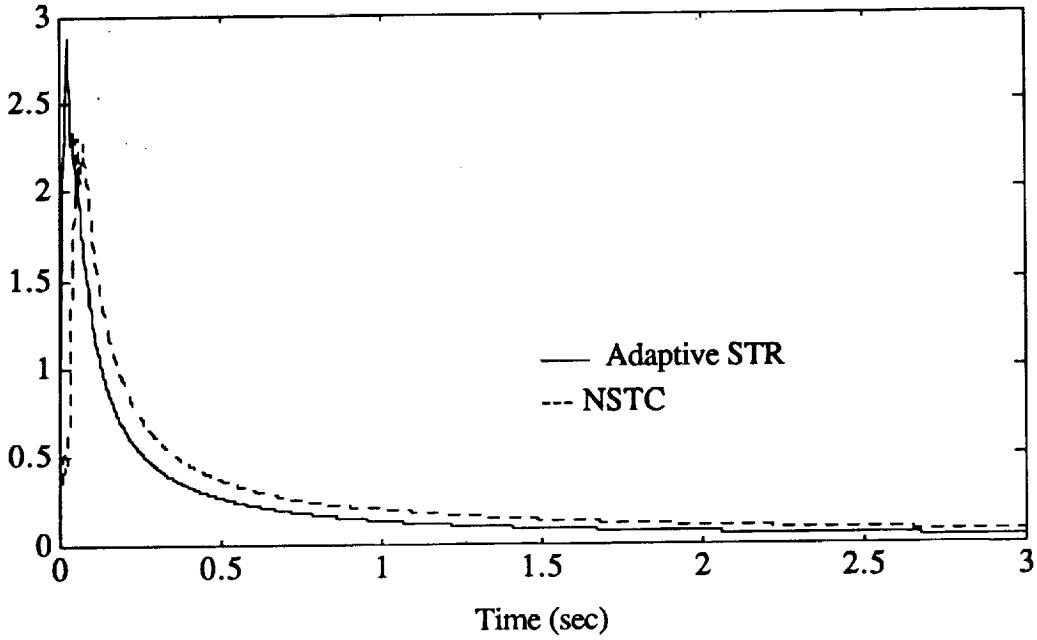

Figure 4.10. Identification cost index V(k) of the adaptive STR and the NSTC

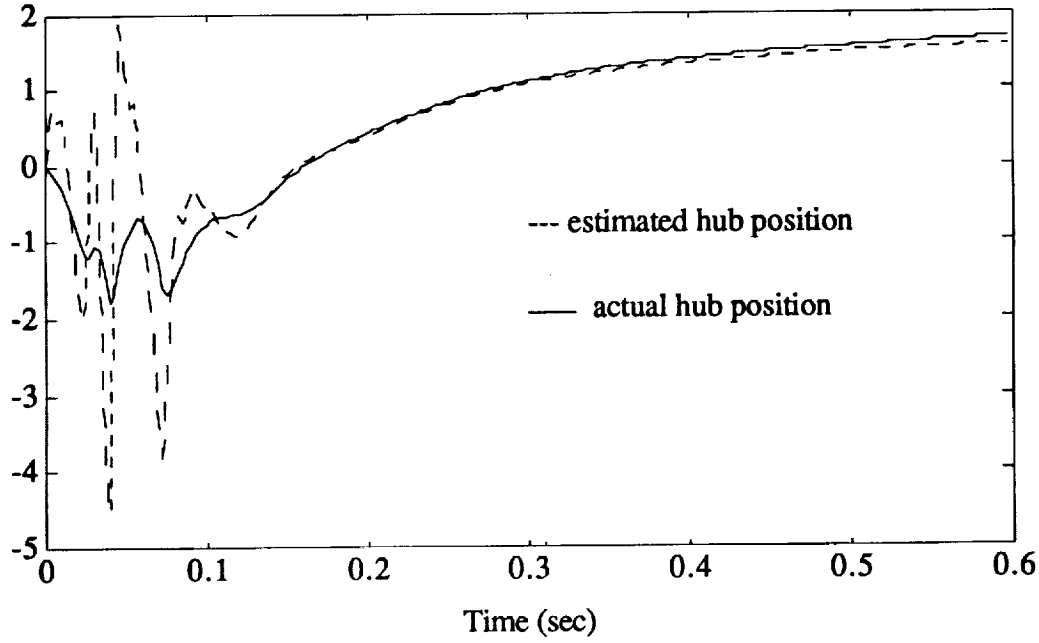

Figure 4.11. True and neural network estimated tip position: $\theta_t(k)$ & $\hat{\theta}_t(k)$

5. Conclusion

The neural self-tuning control (NSTC) algorithm was developed and applied to control the tip of a flexible arm system. The dynamics of the flexible arm tip involves an unstable zero and therefore making the system non-minimum phase. most of the existing neural adaptive control are based on the inverse dynamics and therefore would not be able to control this type of plant. The NSTC was based on an indirect control method where the identification is performed by the neural network and the control was based on the generalized minimum variance (GMV) control law. The performance of the NSTC was investigated and was compared to the adaptive STR by means of simulation.

In summary, the NSTC has a very comparable performance to the adaptive STR shown by simulation results in section 4.2. Unlike other applications of neural networks where thousands of iterations were required before the network can be maturely trained, in this application the neural network identification had a convergent rate comparable to that of the RLS. Another advantage of the NSTC is due to the availability of neural network VLSI and the massive parallel architecture of the neural network there will be a computation advantage over conventional recursive algorithms. This will enable real-time implementation with faster sampling rate for system with high bandwidth. Also another advantage of the NSTC is that because the identification is done by the neural network it inherits the decentralize property, meaning if there is a failure in a node or connection the impact on the performance will be minimal. Whereas with the conventional digital filter a failure in one of the coefficient will have a major impact on the output. With all the above encouraging characteristics there is one disadvantage of using the neural network and that the the lack of understanding how the dimension and activation characteristics of a network is related to its accuracy and stability. Whereas these issues of the recursive algorithms have been addressed and elaborately analysed (Kumar, 1990).

6. Future research

The NSTC can be modified and extended to control systems that are not only non-minimum phase but also nonlinear. This is so that the properties of neural networks can be fully exploited. A system that have the above characteristics is a two degree of freedom robotic manipulator with the second link being flexible. Most conventional adaptive control schemes rely heavily on the inverse dynamics and therefore showed great limitations with this type of system (Centinkunt and Yu, 1990).

Acknowledgement

This research was funded by the NASA grant No. NAG-1-1444. I would like to thank Mr. Hai Ho for his technical advices throughout the course of this research.

REFERENCES

Astrom, K. , and Wittenmark, B., "On self-tuning regulators", Automatica, 9, pp.185-199, 1973.

Centinkunt, S., and Book, W., "Performance Limitations of Joint Variable Feedback Controllers due to Structural Flexibility", IEEE Transaction on Robotics and Automation, Vol. 6, No. 13, 1990.

Clark, D., and Gawthrop, P., "Self-tuning controller", in proceedings IEE, 126, pp.633-640, 1979.

Franklin, F.G., and Powell, J.D., Digital Control of Dynamic Systems, Addison Wesley, Reading, Mass. 1981.

Fraser, A., and Daniel, R., Perturbation Techniques for Flexible Manipulators, Kluwer Academic Publishers, Norwell, Massachusetts, 1991.

Ho, T., Ho, H. "Stochastic state space neural adaptive control", in proceedings the Third International Conference on Advances in Communication and Control Systems, Victoria, B.C., Canada, Oct. 16-18, 1991a.

Ho, H., Ho, T., Wall, E., and Bialasiewicz, J., "Stochastic neural self-tuning adaptive control", in proceedings of the Third International Conference on Advances in Communication and Control Systems, Victoria, B.C., Canada, Oct. 16-18, 1991b.

Ho, T., Ho, H., Wall, E., and Bialasiewicz, J., "Stochastic Neural Direct Adaptive Control", in proceedings of the 1991 IEEE International Symposium on Intelligent Control, Arlington, Virginia, Aug. 13-15, 1991c.

Hunt K., Sbarbaro, D., Zbikkowski, R., and Gawthrop, P., "Neural Networks for Control System - A survey", Automatica, Vo. 28, No. 6, 1992.

Kawato, M., Setoyama, T. and Suzuki, R. "Feedback-error-learning of movement by multi-layer neural network", in proceedings of the International Neural Networks Society First Annual Meeting, 1988.

Kumar, P., "Convergence of Adaptive Control SchemesUsing Least-Squares Parameter Estimates", IEEE Transactions on Automatic Control, Vol. 35, NO 4, pp. 416-424, April 1990.

Levin. A., and Narendra, K., "Control of Nonlinear SystemsUsing Neural Networks: Controllability and Stablilization", IEEE Transactions on Neural Networks, Vol 4. NO. 2, March 1993.

Miller, T., Sutton, R., and Werbos, P., Neural Networks for Control, The MIT press, Cambridge, Massachusetts, 1990.

Narendra, K., and Parthasarathy, K., "Identifciation and control of dynamical systems using neural networks" IEEE Transactions on Neural Networks, Vol 1. NO. 1, March 1990.

Proano, J., Neurodynamic Adaptive Control Systems, Ph.D Dissertation, University of Colorado, Boulder, 1989.

Psaltis, D., Sideris, A., and Yamamura, A., "A multilayered neural network controller" IEEE Control Systems Magazine, April 1988.

Rumelhart, D., and McClelland, J., Parallel Distributed Processing: Vol 1, Foundations, The MIT Press, 1987.

Yamada, T., and Yabuta, T., "Neural Network Controller Using Autotuning Method for Nonlinear Functions", IEEE Transactions on Neural Networks, Vo 3, No. 4, July 1992.

APPENDIX

Simulation Program

The simulation was perforemed using the software MATLAB. The program shown below is the NSTC scheme.

```
clc
clear
%
%========== BEGIN SIMULATION ==========
 N3=1500;    % NUMBER OF ITERATIONS
 ndisp=30;
 ALGsr = 1;          % 1==> Gradient  2==> Newton  3==>MV
 ALGri = 1;          % 1==> Gradient  2==> Newton  3==>MV
 ALGij = 1;          % 1==> Gradient  2==> Newton  3==>MV
 ID=1;              % 1==> RLS  2==> Neural I.D. 0==>Determistic

%=============================================================
%
%========== INITIALIZATION ==========
% PLEASE SELECT THE DIMENSION OF THE STATE VECTOR X0,
% INPUT VECTOR U0, OUTPUT VECTOR Y0, AND PARAMETER VECTOR
% P0 BY MODIFYING THE FOLLOWING STATEMENTS:
%
%  P0=4;
   P00=1; PSI0=1;
%=============================================================
%
  % Plant [a1 a2 a3...ana b0 b1 b2...bnb];
%  A = [.7 .5 -.3]'; B=[1 .2 -.1 .3]';

%  A = [.7 .5]'; B=[1 .2 -.1]';
%  THETAp = [.7 .5 1 .2 -.1]';   % minimum phase 2nd order plant
%    A = [.7 .5 -3]'; B =  [1 .2 -.1 3]';
%  THETAp = [.7 .5 -.3 1 .2 -.1 .3]';  % minimum phase 3rd order
% THETAp = [.7 .5 -.3 1 .2 -.1 3]'; % non-minimum phase 3rd ; ld=4-10
% A=[.7 .5 -.3]'; B=[1 .2 -.1 3]';
%  THETAp = [-2.58 2.18 -.5965 -429.7 884.8 -430.8]';  %missile nmp
%  THETAp = [-3.987 5.96 -3.96 .987 -6.94e-5 6.92e-5 6.9e-5 -6.88e-5]';
%  Mxl
%THETAp = [-3.87 5.63 -3.64 .882 -.0068 .0066 .0065 -.0063]'; %missile
%THETAp = [-2.979 2.96 -.979 -.0047 .0094 -.0047]';  %Submarine
load plant
num=numd'/dend(1);
den=dend'/dend(1);
B=num;
A=den(2:length(den));
% B=numd A=dend(2,:)

  THETAp = [A' B']';
```

```
na=length(A);nb=length(B)-1;d=1;
nf=d-1;
ne=nf+nb;
ng=na-1;
nc=0;      %This assumes the noise has no dynamics, i.e. C=1;
P0 = (ng+1)+(ne+1)+nc;      %Dimension of THETA
P0p=na+nb+1;                %Dimension of plant's THETA
THETAEST=1*ones(P0,1);
%load thetaest
THETA0=THETAEST;
yest=0; w=0;
P=P00*eye(P0,P0);
PSIp=PSI0*ones(P0p,1);
PSId = zeros(P0,1);
K=1.5*ones(P0,1);
Y1=zeros(ng+1,1); U1=zeros(ne+1,1);
Yc=zeros(ng+1,1); Uc=zeros(ne,1);
Y1p=zeros(na,1); U1p=zeros(nb+1,1);
Ud=zeros(d,1); Yd=zeros(d,1);  % delayed values of u, y and w
Wd=zeros(d,1);
y=0; u=0;                    % output y(k), input u(k)
VARV = 0;                   % Output noise variance
MEANV=0;


n0=2; n1 = 5; n2=15; n3=P0;      % Dimensions of Neural Network
NETr = zeros(n1,1);
NETi = zeros(n2,1);
NETj = zeros(n3,1);
Is = zeros(n0,1);
Or = zeros(n1,1);
Oi = zeros(n2,1);
Oj = zeros(n3,1);
ALPHAj = .03*ones(n3,1);, ALPHAi = .03*ones(n2,1);
ALPHAr = .03*ones(n1,1);

Hj = 0*ones(n3,1);, Hi = 0*ones(n2,1);, Hr = 0*ones(n1,1);
Kj = 3*ones(n3,1);, Ki = 2*ones(n2,1);, Kr = 2*ones(n1,1);
Wsr = rand(n0,n1);
Wri = rand(n1,n2);
Wij = rand(n2,n3);

mu=.8;
lambda0=.99;
lambdak = .995;
LAMBDA = 1;
Pij = 10*ones(n2,n3);
Pri = 5*ones(n1,n2);
Psr = 5.3*ones(n0,n1);
Rn = .001;
Re=.9;
%
% ===== END OF INITIALIZATION ==========================
rand('seed',10);

% ------------------
% ::::: BEGIN ITERATION :::::
```

```
for k=1:N3

% ====== STOCHASTIC ARMA REPRESENTATION OF A LINEAR PLANT ========
%
%      y(k)+a1y(k-1)+...+anay(k-na)=b0u(k-d)+b1u(k-1-d)+...+bnbu(k-nb-d)+v(k)
%      y(k)=PSIp'(k)*THETAp(k)+v(k)
%      PSIp'=[-y(k-1)...-y(k-na) u(k-d)...u(k-nb-d)]
%       THETAp(k)'=[a1 a2...ana b0 b1 b2...bnb]
%       THETA'(k) = [g0 g1 ... gng e0 e1...ene]
%      PSId'(k) = [y(k-d).. y(k-d-ng) u(k-d)..u(k-d-ne)]
%       yest(k) = PSId(k)'*THETAEST

%==========Computing THETA ==========

if d==1
  E=B;
  G=-A;     %g(i)=-a(i+1), i=0..ng G=q-1(1-A)
end
THETA = [G'E']';


% PARAMETRIZATION FOR PSI(k).
%
%=================================================
%

                                        % u=u(k-1) y=y(k-1)
 for i=d-1:-1:1 Ud(i+1)=Ud(i);, end, Ud(1)=u;   %[u(k-1)...u(k-d)]
 for i=d-1:-1:1 Yd(i+1)=Yd(i);, end, Yd(1)=y;   %[y(k-1)...y(k-d)]
 for i=d-1:-1:1 Wd(i+1)=Wd(i);, end, Wd(1)=w;   %[w(k-1)...w(k-d)]

%===== PSIp(k) = [-y(k-1).. -y(k-na) u(k-d)..u(k-d-nb)]'
 for i=na-1:-1:1 Y1p(i+1)=Y1p(i);, end; Y1p(1)=-Yd(1);
 for i=nb:-1:1 U1p(i+1)=U1p(i);, end; U1p(1)=Ud(d);
 PSIp = [Y1p' U1p']';

%=====PSId(k) = [y(k-d).. y(k-d-ng) u(k-d)..u(k-d-ne)]
for i=ng:-1:1 Y1(i+1)=Y1(i);, end; Y1(1)=Yd(d);  %[y(k-d).. y(k-d-ng)]
for i=ne:-1:1 U1(i+1)=U1(i);, end; U1(1)=Ud(d);  %[u(k-d)..u(k-d-ne)]
PSId = [Y1' U1']';    % PSI(k-d)


%=================================================
% ----- GENERATING NOISE v(k) -----
 rand('normal')
 v=sqrt(VARV)*rand(1,1)+MEANV;

%
% ----- COMPUTING y(k) & w(k) -----
%

    y=PSIp'*THETAp+v;        %y(k)

 tau=.5;
 w=tau*w + (1-tau)*2; %*sign(sin(0.004*(k)));     %command signal w(k)
 w=(w/2)+1;
%
% ----- END OF PLANT ------------
```

```
%═══════════════════════════════════════════════

% ----- ADAPTIVE ESTIMATION -----
% ═══ THE STOCHASTIC LEAST SQUARES ALGORITHM (SLA) ═══
%
%═══════════════════════════════════════════════
% ---- BEGIN ESTIMATION -----
% THETAEST = [g0 g1 ... gng e0 e1 ... ene]'

 yest=PSId'*THETAEST;            % PREDICTED OUTPUT yest(k)
 e=y-yest;                  % PREDICTION ERROR e(k)

if ID==1
 K=P*PSId*inv(1+(PSId'*P*PSId));      % OPTIMAL GAIN
 THETAEST=THETAEST+K*e;               % PARAMETER ESTIMATION
 P=(P-K*PSId'*P);
end
%═══════════Neural identification ═══════

if ID==2
 Is(1)=1;

 Netr = Wsr'*Is;
 tempr = (ALPHAr/2).*(Netr+Hr);
 Or = Kr.*tanh(tempr);
 Or(1)=1;

 Neti = Wri'*Or;
 tempi = (ALPHAi/2).*(Neti+Hi);
 Oi = Ki.*tanh(tempi);
 Oi(1)=1;

 Netj = Wij'*Oi;
 tempj = (ALPHAj/2).*(Netj+Hj);
 Oj = Kj.*tanh(tempj);

 THETAEST = Oj;
 if k==1  save thetaest THETAEST, end
 PSI=PSId;

 tempj2 = cosh(tempj).*cosh(tempj);
 tempi2 = cosh(tempi).*cosh(tempi);
 tempr2 = cosh(tempr).*cosh(tempr);
 Fdotj = (Kj.*ALPHAj/2)./(tempj2);
 Fdoti = (Ki.*ALPHAi/2)./(tempi2);
 Fdotr = (Kr.*ALPHAr/2)./(tempr2);

 dj = Fdotj.*PSI;
 PSIij = Oi*dj';
 di = (Wij*dj).*Fdoti;
 PSIri = Or*di';
 Q = Fdoti .* (Wij*(Fdotj.*PSI));
 dr = Fdotr .* (Wri*Q);
 PSIsr = Is*dr';

 if ALGij == 1     % Gradient
     Lij = mu*PSIij/LAMBDA;
 end
```

```
 if ALGij == 2     % Newton
      Sij = (PSIij.*PSIij.*Pij) + (lambdak*LAMBDA*ones(n2,n3));
       Lij = (Pij.*PSIij)./Sij;
      Pij = (Pij - (Lij.*Sij.*Lij))/lambdak;
 end
 if ALGij == 3     % Minimum Variance
      Sij = (PSIij.*PSIij.*Pij) + Re*ones(n2,n3);
      Lij = (Pij.*PSIij)./Sij;
      Pij = Pij - (Lij.*PSIij.*Pij) + Rn*ones(n2,n3);
 end

 if ALGri == 1     % Gradient
      Lri = mu*PSIri/LAMBDA;
 end
 if ALGri == 2     % Newton
      Sri = (PSIri.*PSIri.*Pri) + (lambdak*LAMBDA*ones(n1,n2));
      Lri = (Pri.*PSIri)./Sri;
      Pri = (Pri - (Lri.*Sri.*Lri))/lambdak;
 end
 if ALGri == 3     % Minimum Variance
      Sri = (PSIri.*PSIri.*Pri) + Re*ones(n1,n2);
      Lri = (Pri.*PSIri)./Sri;
      Pri = Pri - (Lri.*PSIri.*Pri) + Rn*ones(n1,n2);
 end

 if ALGsr == 1     % Gradient
      Lsr = mu*PSIsr/LAMBDA;
 end
 if ALGsr == 2     % Newton
      Ssr = (PSIsr.*PSIsr.*Psr) + (lambdak*LAMBDA*ones(n0,n1));
      Lsr = (Psr.*PSIsr)./Ssr;
      Psr = (Psr - (Lsr.*Ssr.*Lsr))/lambdak;
 end
 if ALGsr == 3     % Minimum Vasrance
      Ssr = (PSIsr.*PSIsr.*Psr) + Re*ones(n0,n1);
      Lsr = (Psr.*PSIsr)./Ssr;
      Psr = Psr - (Lsr.*PSIsr.*Psr) + Rn*ones(n0,n1);
 end

  Wij = Wij + Lij*e;
  Wri = Wri + Lri*e;
  Wsr = Wsr + Lsr*e;

 %LAMBDA = LAMBDA + (e*e'-LAMBDA)/k;
 lambdak = lambda0*lambdak+(1-lambda0);
 Re = Re + (e*e'-Re)/k;


 for i=n0:-1:2 Is(i)=Is(i-1);, end

end

%
% ----- END OF ESTIMATION ----

%
% ==== MINIMUM VARIANCE ADAPTIVE CONTROL ==========================
```

```
%
%
%=============================================================
% ----- BEGIN ADAPTIVE CONTROL -----

for i=ng:-1:1 Yc(i+1)=Yc(i);, end; Yc(1)=y;  %[y(k).. y(k-ng)]
for i=ne-1:-1:1 Uc(i+1)=Uc(i);, end; Uc(1)=u;  %[u(k-1)..u(k-ne)]

ld = .1;
ld2= .06;
ld3=0;
if ID==0 THETAEST=THETA;, end                  % Q = ld + q-1ld2
if k<1 THETAc=THETA;, else THETAc=THETAEST;, end
Gest(1:ng+1,1) =  THETAc(1:ng+1);            % G
Eqest(1:ne+1,1) = THETAc(ng+2:ng+2+ne);      % E
Eqest(1) = Eqest(1)+ld;                      % E+Q
Eqest(2) = Eqest(2)+ld2;
%Eqest(3) = Eqest(2)+ld3;
% u(k) = {w(k)-[g0y(k)+...+gncy(k-ng)] -[e1u(k-1)+...+eneu(k-ne)]}/e0

SUM1 = Eqest(2:ne+1)'*Uc;
SUM2 = Gest'*Yc;
%roots(Eqest)
%break
u=(w-SUM2-SUM1)/Eqest(1);      %u(k)
%u=w;

% ----- END OF ADAPTIVE CONTROL -----
%=============================================================

%
% ----- SIMULATION ERRORS ----------
% ----- SAVE THETA(k) & THETAEST(k) -----
 for j=1:P0
  THETA1(k,j)=THETA(j);
  THETA1EST(k,j)=THETAEST(j);
 end

% ----- SAVE y(k) & yest(k) -----
 Y(k,1)=y;
 YEST(k,1)=yest;
% ----- SAVE K(k) -----
 for j=1:P0
  K1(k,j)=K(j);
 end

% -----SAVE U(k) -----
 U(k)=u;
 W(k)=Wd(d);
%=============================================================
%
% ----- THE PARAMETER IDENTIFICATION MSE(k) -----
 THETAER=THETA1-THETA1EST;
 for j=1:P0
  if k==1, TMSE(k,j)=THETAER(k,j)^2; else
   TMSE(k,j)=TMSE(k-1,j)+(THETAER(k,j)^2-TMSE(k-1,j))/k;
  end
 end
```

```
% ---- THE OUTPUT PREDICTION MSE(k) -----
 YER(k)=y-yest;
 if k==1, YMSE(k)=YER(k)^2; else
   YMSE(k)=YMSE(k-1)+(YER(k)^2-YMSE(k-1))/k;
 end

% ----- THE COST FUNCTION(k) J(k)-----
 YERc(k)=y-Wd(d);
 if k==1, J(k)=YERc(k)^2; else
   J(k)=J(k-1)+(YERc(k)^2-J(k-1))/k;
 end

% ======= DISPLAY MATRIX ==========================================
% THIS M-FILE IS USED TO MONITOR SYSTEM PERFORMANCES DURING
% SIMULATION.
%=================================================================
% ---- TRANSFER DATA TO MATRIX DISMAT1 ----
 DISMAT1(1,1)=k;
 DISMAT1(1,2)=TMSE(k,1);
 DISMAT1(1,3)=TMSE(k,2);
 DISMAT1(1,4)=TMSE(k,3);
%  DISMAT1(1,5)=TMSE(k,4);
 DISMAT1(1,6)=YMSE(k);
 DISMAT1(1,7)=U(k);
% ----- TRANSFER DATA TO MATRIX DISMAT2 -----
 DISMAT2(1,1)=k;
 DISMAT2(1,2)=J(k);
 DISMAT2(1,3)=Wd(d);
 DISMAT2(1,4)=y;
 DISMAT2(1,5)=yest;
% ----- DISPLAY DISMAT1 & DISMAT2 -----
if rem(k,ndisp)==0
 home
 disp('     k    TMSE1     TMSE2     TMSE3     TMSE4     YMSE     U(k)')
 disp(DISMAT1)
 disp('     k    J(k)    w(k-d)    y(k)     yest')
 disp(DISMAT2)
 %[THETA THETAEST]
end

%
% ======= END OF DISMAT ==========================================

 %keyboard
 end       % END OF FOR LOOP(k)
% ::::: END OF ITERATION :::::
%
% ---- SYSTEMS GRAPHICS ----------
 SYGRAF
% ----------------------------

% ======= END OF SIMULATION ======================================
```