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

By
Long T. Ho
B.S.E.E., University of Colorado, 1992

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfilment
of the requirements for the degree of
Master of Science
Department of Electrical Engineering
1994
This thesis for the Master of Science degree by
Long Thanh Ho
has been approved for the
Department of
Electrical Engineering
by

Jan T. Bialasiewicz

Miloje Radenkovic

Marvin Anderson
Acknowledgments

The author wishes to express sincere appreciation to Dr. Jan T. Bialasiewicz for his guidance and joyful support during the course of this thesis. Also, enthusiasms of Dr. Miloje Radenkovic and Professor Marvin Anderson as the author's thesis committee are gratefully acknowledged. Special thanks to the author's family for their never ending support.
Ho, Long Thanh (M.S., Electrical Engineering)
Neural Self-Tuning Adaptive Control Strategies of Non-Minimum Phase System
Developed for Flexible Robotic Arm
Thesis directed by Professor Jan T. Bialasiewicz

The motivation of this research came about when a neural network direct adaptive control schemes were 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. 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 has been 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.

The form and content of this abstract are approved.

Signed Jan T. Bialasiewicz
# Table of Content

1. Introduction ................................................................. 9
   1.2. Neural Control Survey ................................................ 12
       1.2.1. Stochastic Neural Direct Semi-Adaptive Control ......... 13
       1.2.2. Stochastic Neural Direct Adaptive Control ........... 15
       1.2.3. Inverse Neural Adaptive Control........................ 15
       1.2.4. Feedback Error Learning and Control.................. 16
       1.2.5. Inverse Dynamic Model Reference Control of a
               Class of Nonlinear Plants.............................. 17
       1.2.6. Neural Linear State Space Control ...................... 18
       1.2.7. Neural Self-Tuning Control............................ 19
2. Stochastic Neural Self-Tuning Adaptive Control Scheme .......... 21
   2.1. Generalized Minimum Variance Control ........................ 22
   2.2. Neural System Identification .................................. 23
3. Flexible Arm Tip Position Dynamics .................................. 28
4. Empirical Studies ........................................................ 35
   4.1. Neural Direct Adaptive Control of Arm Hub and Tip ...... 35
   4.2. Neural Self-Tuning Adaptive Control of Arm Tip .......... 39
5. Conclusions .................................................................... 45
   5.1 Future Research ...................................................... 46
Figures and Tables

Figures:

1.1 Flexible Arm System...................................................... 10
1.2 Specialized Learning Control of HupVelocity........................ 11
1.3 Indirect Neural Adaptive Control Scheme............................. 11
1.4 Adaptive control general block diagram................................ 13
1.5 Direct semi-adaptive control scheme................................... 12
1.6 Neural Network structure................................................ 14
1.7 Direct neural adaptive controller....................................... 14
1.8 Inverse neural control.................................................... 15
1.9 Feedback error learning and control................................... 16
1.10 Inverse dynamic model reference neural control..................... 18
1.11 Neural linear state space control...................................... 19
1.12 Stochastic neural linear ARMA control................................. 20

2.1 Neural Network Structure................................................ 25

3.1 Pole-Zero Diagram of Flexible Arm Tip................................ 29
3.2 Servo Motor System Components....................................... 30
3.3 Frequency Magnitude Response of Arm Tip with Five Resonant Modes
................................................................................................. 32
3.4 Frequency Response of Open-Loop Components....................... 33
3.5 Frequency Response of the Aggregate filtered Open-Loop........ 33
4.1 Neural Direct Control Scheme of Hub Velocity.................... 36
4.2 Hub Velocity Response.............................................. 37
4.3 Control Tracking MSE Response.................................. 37
4.4 Unstable Response of Tip Control................................ 38
4.5 Diverging Tracking MSE of Tip Velocity.......................... 38
4.6 NSTC Scheme Block Diagram...................................... 40
4.7 Tip Position Response................................................ 42
4.8 Control Performance Index J(K) of the Adaptive STR and the NSTC
.......................................................... 43
4.9 Control Signal u(k) of the adaptive STR and the NSTC............. 43
4.10 Identification cost Index V(k) of the Adaptive STR and the NSTC. 44
4.11 True and Neural Network Estimated Tip Position................. 44
Tables:
3.1 Physical Properties of Arm and Motor............................ 31
3.2 Poles and Zeros...................................................... 32
CHAPTER 1

INTRODUCTION

Most existing neuro control schemes are in the form of the direct method, where the neural network is trained to approximate the inverse of the plant. In the case where the plant is non-minimum phase, the inverse approximation introduces instability in the closed-loop system. Therefore, an indirect neuro control scheme is proposed to deal with non-minimum phase systems. Specifically, we propose to use a neural network to identify the plant parameters, then combine this with a minimum variance control law. The plant in this study is a single degree of freedom flexible robotic arm.

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 thesis, the neural self tuning control (NSTC) scheme is used 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 Parthasarathy, 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 thesis, the performance of the NSTC is compared 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 at a different location than 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 in the right half of the s-plane. In other words, the transfer function of the tip angle is of the non-minimum phase type 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-minimum phase 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](image-url)
In this thesis, 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. Finally, section 5 gives the conclusion of the results found in this study and addresses the advantages and disadvantages of the neural control scheme used for treating linear system.

1.2 Neural Control Survey

In the past five years neural network based adaptive control has been a proliferated and challenging field for researchers in the area of adaptive and nonlinear control. As technology advances and more and more dynamical systems emerge with high degree of complexity in coupled and nonlinear characteristics, conventional modern and adaptive control techniques are showing to be less and less effective in achieving demanding control performance. This is partly due to the fact that many of these systems are linearized and decoupled beforehand in order to apply conventional control techniques, which consequently causes inaccuracies in representing system dynamics and therefore loses effectiveness in controlling the system. Neural network based adaptive control (NNBAC) has shown to have some unique and superior capabilities in controlling stochastic nonlinear time varying systems mainly because neural networks can model nonlinear complex processes more accurately. Furthermore, due to the inherent parallel structure of neural networks, NNBAC offers the major computational load advantage because of parallel computations. Hence, implementation is more possible in cases dealing with large scales and/or high bandwidth systems where sufficiently fast sampling rate is required. In this section, we briefly present a survey of existing neural control schemes.

Consider the general block diagram of an adaptive control scheme shown in Figure 1.4. Now, an adaptive control scheme may assume no a priori knowledge of the plant, but an effective and prudent adaptive control scheme should utilize and exercise all the a priori knowledge that is available. Some of the early neural control
schemes such as the inverse indirect learning and the specialized learning were impressive because these schemes required very little a priori information about the nonlinear plant and treated it like a "black box".

![Adaptive control general block diagram](image)

**Figure 1.4. Adaptive control general block diagram**

### 1.2.1 STOCHASTIC NEURAL DIRECT SEMI-ADAPTIVE CONTROL

Consider the first scheme called the stochastic neural direct semi-adaptive control shown in Figure 1.5 (Ho et al., 1991c). This scheme is the stochastic weighted version of the specialized learning (Psaltis et al., 1988) and is formulated with the well known weighted optimal control cost function

\[
J(k) = \frac{1}{2} \mathbb{E} \{[y(k)-y^*(k)]'Q[y(k)-y^*(k)] + u(k)'Ru(k)\}
\]

(1.1)
This control scheme is based on the nonlinear stochastic state space model

\[ x(k+1) = f(x(k)) + B(k)u(k) + w(k) \]

\[ y(k) = C(k)x(k) + v(k) \quad (1.2) \]

The a priori information required for this scheme is the input/output dynamic matrices B and C. This is so that the plant jacobian \( \partial y(k)/\partial u(k) \) can be computed and used in the back propagation algorithm. Figure 1.6 shows the typical structure of a multilayered neural network.
1.2.2. STOCHASTIC NEURAL DIRECT ADAPTIVE CONTROL

This scheme is almost identical to the previous scheme (Ho et al., 1991d) except that it has no a priori information about the plant input/output dynamics. Therefore it incorporates an additional neural network so that the plant jacobian can be estimated. The block diagram of this scheme is shown in Figure 1.7 where the plant was basically treated to be a "black box" nonlinear system with the general state space form

\[ x(k+1) = f[x(k), u(k)] + w(k) \]

\[ y(k) = g[x(k)] + v(k) \]

(1.3)

\[ y'(k) \text{ desired output} \]

\[ \hat{y}(k) \text{ estimation error} \]

\[ \frac{\partial y}{\partial u} \]

\[ y(k) \text{ output} \]

\[ u(k) \text{ input} \]

\[ w(k) \text{ noise} \]

\[ e_c(k) \text{ tracking error} \]

\[ \sum \]

Figure 1.7. Direct neural adaptive controller

1.2.3. INVERSE NEURAL ADAPTIVE CONTROL

This is one of the first neural adaptive control schemes known as the indirect learning proposed by (Psaltis et al., 1988) and is shown is Figure 1.8. The plant is assumed to be a "black box" nonlinear system

\[ y(k) = f[y(k-1), ..., y(k-n); u(k), ..., u(k-m)] \]

(1.4)
Here, the two neural networks at the input and output of the plant are identical. The network is to emulate the inverse of the plant based on optimization of the cost function

$$J(k) = \frac{1}{2} [u(k) - \hat{u}(k)]' [u(k) - \hat{u}(k)]$$

which indirectly minimizes the output tracking error $[y(k) - y^*(k)]$.

### 1.2.4. FEEDBACK ERROR LEARNING AND CONTROL (FELC)

This direct adaptive control method proposed by (Kawato et al. 1988) may be one of the most efficient "black box" neural control scheme as shown in Figure 1.9, the scheme utilizes a single neural network as an adaptive direct controller performing both learning and control simultaneously.
The neural network directly minimizes the output tracking error cost function

\[ J(k) = \frac{1}{2} [y^*(k) - y(k)]'Q[y^*(k) - y(k)] \]  \hspace{1cm} (1.6)

and does not require any a priori information such as the Jacobian.

1.2.5. INVERSE DYNAMIC MODEL REFERENCE CONTROL OF A CLASS OF NONLINEAR PLANTS

This approach was presented by (Narendra and Parathasarathy, 1990) addressing the issues of identification utilizing neural networks and control of nonlinear plant using inverse dynamic model reference techniques. The general diagram of this scheme is shown in Figure 1.10. The four input/output plant models addressed are

Model I:

\[ y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + g[u(k), u(k-1), \ldots, u(k-m+1)] \]  \hspace{1cm} (1.7a)

Model II:

\[ y(k+1) = f[y(k), y(k-1), \ldots, y(k-n+1)] + \sum_{i=0}^{m-1} \beta_i u(k-i) \]  \hspace{1cm} (1.7b)

Model III:

\[ y(k+1) = f[y(k), y(k-1), \ldots, y(k-n+1)] + g[u(k), u(k-1), \ldots, u(k-n+1)] \]  \hspace{1cm} (1.7c)

Model IV:
\[ y(k+1) = f[y(k), y(k-1), ..., y(k-n+1); u(k), u(k-1), ..., u(k-n+1)] \]

(1.7d)

Figure 1.10 Inverse dynamics model reference neural control

The a priori information required is which of these specific model fits the plant so that identification can be performed. However, the inverse dynamic control can be accomplished provided the representation of the inverse dynamics exists. In other words, it is recognize that \( u(k) \) can be expressed in terms of \( f(\cdot), g(\cdot), f^{-1}(\cdot) \) and \( g^{-1}(\cdot) \).

### 1.2.6. NEURAL LINEAR STATE SPACE CONTROL

This scheme shown in Figure 1.11. (Ho and Ho, 1991a) is used for controlling time varying linear stochastic state space plant

\[
x(k+1) = A(\theta,k)x(k) + B(\theta,k)u(k) + w(k)
\]

\[
y(k) = C(\theta,k) + v(k)
\]

(1.8)

where \( \theta \) is the parameter vector. The identification is performed by the neural network and the control can be selected by any modern state space control techniques, in
particular, the tracking per-interval control law. This neural parameter adaptive control approach is different from the conventional adaptive control approach by the identification process.

![Neural linear state space control diagram](image)

Figure 1.11. Neural linear state space control

### 1.2.7. NEURAL SELF-TUNING CONTROL

This scheme, shown in Figure 1.12. (Ho et al, 1991b) is similar to the state space control scheme only it is based on the ARMA plant model:

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

The identification is performed by the neural network and the control can be selected by any conventional control techniques, in particular, the minimum variance control. This neural self-tuning control scheme is different from the conventional self-tuning control by the identification algorithm.
Figure 1.12. Stochastic neural linear ARMA control
CHAPTER 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

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

(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

\[ A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na} \]

(2.2)

\[ B(q^{-1}) = b_0 + b_1 q^{-1} + b_2 q^{-2} + \ldots + b_{nb} q^{-nb} \]

(2.3)

\[ C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{nc} q^{-nc} \]

(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)

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

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

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

(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

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

where the order of \( F(q^{-1}) \) and \( G(q^{-1}) \) are \( n_f = d-1 \), \( n_G = n_A - 1 \), 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

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

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

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

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

\[ u(k) = \frac{R(q^{-1})y^*(k) - \phi_y(k+d)}{Q(q^{-1})} \]  

(2.1.7)

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

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

(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

\[ \phi_y(k+d) = C(q^{-1})^{-1}[G(q^{-1})y(k) + E(q^{-1})u(k)] + F(q^{-1})\xi(k+d) \]

\[ = C(q^{-1})^{-1}[G(q^{-1})y(k) + E(q^{-1})u(k)] + v(k) \]  

(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

\[ \phi_y(k+d) = \sum_{i=0}^{\text{ng}} g_i y(k-i) + \sum_{i=0}^{\text{ne}} e_i u(k-i) - \sum_{i=1}^{\text{nc}} e_i \phi_y(k+d-i) + v(k) \]  

(2.2.2)

\[ \phi_y(k+d) = \psi(k)\theta(k) + v(k) \]  

(2.2.3)

where
\[ \psi'(k) = \begin{bmatrix} y(k) & \cdots & y(k-ng) & u(k) & \cdots & u(k-ne) & \phi_y(k+d-1) & \cdots & \phi_y(k+d-nc) \end{bmatrix} \] (2.2.4)

\[ \theta'(k) = \begin{bmatrix} g_0 & g_1 & \cdots & g_{ng} & e_0 & e_1 & \cdots & e_{ne} & -c_1 & -c_2 & \cdots & -c_{nc} \end{bmatrix} \] (2.2.5)

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

\[ \hat{\phi}_y(k+d) = \hat{\psi}(k)\hat{\theta}(k) \] (2.2.6)

where

\[ \hat{\psi}(k) = \begin{bmatrix} y(k) & \cdots & y(k-ng) & u(k) & \cdots & u(k-ne) & \hat{\phi}_y(k+d-1) & \cdots & \hat{\phi}(k+d-nc) \end{bmatrix} \] (2.2.7)

\[ \hat{\theta}'(k) = \begin{bmatrix} \hat{g}_0 & \hat{g}_1 & \cdots & \hat{g}_{ng} & \hat{e}_0 & \hat{e}_1 & \cdots & \hat{e}_{ne} & \hat{-c}_1 & \hat{-c}_2 & \cdots & \hat{-c}_{nc} \end{bmatrix} \] (2.2.8)

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

\[ \hat{\theta}(k) = \begin{bmatrix} \hat{\theta}_1(k) & \hat{\theta}_2(k) & \cdots & \hat{\theta}_j(k) & \cdots & \hat{\theta}_n3(k) \end{bmatrix}' \]

\[ = \begin{bmatrix} \hat{O}_1(k) & \hat{O}_2(k) & \cdots & \hat{O}_j(k) & \cdots & \hat{O}_n3(k) \end{bmatrix}' \] (2.2.9)

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

\[ V(k) = \frac{1}{2} \mathbb{E} \{ \epsilon'(k)\Lambda^{-1}(k)\epsilon(k) \} \]

\[ = \frac{1}{2} \mathbb{E} \{ [\phi_y(k)\hat{\phi}_y(k)]'\Lambda^{-1}(k)[\phi_y(k)\hat{\phi}_y(k)] \} \] (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.
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

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

where

$$\Delta \omega_{ij}(k) = -\eta \frac{\partial}{\partial \omega_{ij}(k)} \left\{ \frac{1}{2} e'(k) \Lambda^{-1}(k)e(k) \right\}$$

$$= -\eta \frac{\partial}{\partial \omega_{ij}(k)} \left\{ \frac{1}{2} [\phi_y(k) - \hat{\phi}_y(k)]' \Lambda^{-1}(k) [\phi_y(k) - \hat{\phi}_y(k)] \right\}$$

$$= \eta \frac{\partial \hat{\phi}_y(k)}{\partial \omega_{ij}(k)} \frac{\partial \phi_y(k)}{\partial \hat{\phi}_y(k)} \Lambda^{-1}(k) [\phi_y(k) - \hat{\phi}_y(k)]$$  \hspace{1cm} (2.2.12)

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

$$\frac{\partial \phi_y'(k)}{\partial \hat{\theta}(k)} = \hat{\psi}(k-d)$$  \hspace{1cm} (2.2.13)

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

$$\frac{\partial \hat{\theta}'(k)}{\partial \omega_{ij}(k)} = O_i(k) e_j \frac{\partial f(Net_j(k))}{\partial Net_j(k)}$$  \hspace{1cm} (2.2.14)
where \( f(.) \) is the sigmoidal activation function, \( O_i(k) \) is the output of the second layer, and

\[
Net_j(k) = \begin{bmatrix} \text{net}_1 & \text{net}_2 & \ldots & \text{net}_j & \ldots & \text{net}_n \end{bmatrix}'
\]  

(2.2.15)

with

\[
\text{net}_j(k) = \sum_{i=1}^{n_2} \omega_{ij}(k) O_i(k)
\]

where \( n_2 \) is the number of neurons of the second hidden layer as shown in Figure 2.1. Also \( e_j \) in (2.2.14) is defined as

\[
e_j = [0 \ldots 0 1 0 \ldots 0]
\]

(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

\[
\Delta \omega_{ij}(k) = \eta e_j \delta_j(k) O_i(k)
\]

(2.2.17)

where

\[
\delta_j(k) = \frac{\partial [f(\text{Net}_j(k))]'}{\partial \text{Net}_j(k)} \frac{\partial \phi'(y(k))}{\partial \sigma(k)} (\Lambda^{-1}(k)[\phi_y(k) - \hat{\phi}_y(k)])
\]

(2.2.18)

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

\[
\omega_{ri}(k+1) = \omega_{ri}(k) + \Delta \omega_{ri}(k)
\]

(2.2.19)

where

\[
\Delta \omega_{ri}(k) = -\eta \frac{\partial}{\partial \omega_{ri}(k)} \left\{ \frac{1}{2} \varepsilon'(k) \Lambda^{-1}(k) \varepsilon(k) \right\}
\]

(2.2.20)

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

\[
\Delta \omega_{ri}(k) = \eta \delta_i(k) O_r(k)
\]

(2.2.21)

where \( O_r \) is the output of the first layer and
Lastly, the weights connecting the input to the first layer, in Figure 2.1, can be updated by the recursive equation

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

(2.2.23)

where

$$\Delta \omega_{sr}(k) = -\eta \frac{\partial}{\partial \omega_{sr}(k)} \left\{ \frac{1}{2} e'(k)(k) \right\}$$

(2.2.24)

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

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

(2.2.25)

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

$$\delta_r(k) = [\omega_{r1} \ldots \omega_{ri} \ldots \omega_{rn2}] \frac{\partial f(\text{net}_r(k))}{\partial \text{net}_r(k)} \frac{\partial f(\text{net}_i(k))}{\partial \text{net}_i(k)} \frac{\partial \text{net}_i(k)}{\partial f(\text{net}_i(k))} \delta_j(k)$$

(2.2.26)

with $\text{net}_i(k)$ being defined similarly as $\text{net}_j(k)$ 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 $\hat{\theta}$ 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

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

(2.2.27)
CHAPTER 3

FLEXIBLE ARM TIP POSITION DYNAMICS

This chapter 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

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

(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
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

$$\frac{K_A K_i}{(s+K_i)s}$$

(3.2)

where $K_T$ is the motor torque constant, $K_A$ represents the power amplifier and sensor gain, and $C_0$ 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.
Next, the flexible arm attached to the motor shaft is described by the term.

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

(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 $\zeta_i$ at a frequency $\omega_i$. Theoretically, 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 oscillations 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 $\zeta_i$'s and frequencies $\omega_i$'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 3.3. The peaks represent the resonant energy at the specific frequencies. Also notice that the energy of the modes lessens as the frequency increases.
<table>
<thead>
<tr>
<th>Physical properties of arm and motor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>effective beam length (m)</td>
<td>0.386</td>
</tr>
<tr>
<td>beam thickness (mm)</td>
<td>0.956</td>
</tr>
<tr>
<td>beam width</td>
<td>0.03</td>
</tr>
<tr>
<td>mass/unit length of arm m (kg/m)</td>
<td>0.222</td>
</tr>
<tr>
<td>flexural rigidity of beam(Nm^2)</td>
<td>0.426</td>
</tr>
<tr>
<td>hub moment of inertia (kg m^2)</td>
<td>0.00009</td>
</tr>
<tr>
<td>radius of hub (m)</td>
<td>0.034</td>
</tr>
<tr>
<td>Tip mass for loaded arm (kg)</td>
<td>0.065</td>
</tr>
<tr>
<td>tip inertia for loaded arm (kg m^2)</td>
<td>0.000005</td>
</tr>
<tr>
<td>continuous torque at rated speed (Nm)</td>
<td>0.177</td>
</tr>
<tr>
<td>pulse torque (Nm)</td>
<td>2.913</td>
</tr>
<tr>
<td>rated voltage (V)</td>
<td>24</td>
</tr>
<tr>
<td>torque constant (Nm/A)</td>
<td>0.048</td>
</tr>
<tr>
<td>total inertia (kg m^2)</td>
<td>0.000041</td>
</tr>
<tr>
<td>Ka*Kt</td>
<td>3.6</td>
</tr>
<tr>
<td>Co (rad/sec)</td>
<td>0.16</td>
</tr>
</tbody>
</table>
Table 3.2

<table>
<thead>
<tr>
<th>Mode</th>
<th>POLES (rad/sec)</th>
<th>ZEROS (rad/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expmt.</td>
<td>Theory</td>
</tr>
<tr>
<td>1</td>
<td>86.1</td>
<td>86.9</td>
</tr>
<tr>
<td>2</td>
<td>297.6</td>
<td>285.3</td>
</tr>
<tr>
<td>3</td>
<td>603.2</td>
<td>601.9</td>
</tr>
<tr>
<td>4</td>
<td>1011.6</td>
<td>1065.0</td>
</tr>
<tr>
<td>5</td>
<td>1445.1</td>
<td>1658.8</td>
</tr>
</tbody>
</table>

Figure 3.3 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 modes. Figure 3.4 shows a block diagram of the filtering process. The resulting frequency ideal response is shown in Figure 3.5.

Figure 3.4. Frequency response of open-loop components

Figure 3.5. 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

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

(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-collocattted 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).
CHAPTER 4

EMPIRICAL STUDIES

In this chapter 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. The simulation program used in this study is given in the appendix.

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

\[
\frac{\theta_h(s)}{U(s)} = \frac{10.2 \left(1+\frac{s^2}{32.7^2}\right)}{(s+0.57)(s+2000)}
\]  

\[\text{(4.1)}\]
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

\[ x(k+1) = Ax(k) + Bu(k) \]  
\[ \theta_h(k) = Cx(k) \]

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

\[ \frac{\partial \theta_h(k)}{\partial u(k)} = CB \]  

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.
This trackability is reflected in the mean square tracking error shown in Figure 4.3. Notice that the convergent time in control application is several 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.

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

\[
\theta_t(s) = \frac{3.6 (1+\frac{s^2}{48.4^2})}{s(s+0.16)}
\]  

(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: $\theta_1^*(k)$ and $\theta_1(k)$](image)

![Figure 4.5. Diverging tracking MSE of tip velocity](image)

Neural Network: The $N_{5,10,1}^2$ 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 were found to be most influential on the tracking error convergence.
rate. Predominantly the slope of the activation function was observed to be proportional to the convergence 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 result was achieved. Auto-tuning of the sigmoidal function parameters can also be applied to obtain statistically 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.
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 was done to performis a comparative study in order 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
\[ P(q^{-1}) = 1; \quad Q(q^{-1}) = 0.1 + 0.06q^{-1}; \quad R(q^{-1}) = 1 \]  

(4.6)

and the desired hub position \( \hat{\theta}_t(k) \) was a step command. Beginning with Figure 4.7, shows 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 \( \hat{\theta}_0 = [1 \ldots 1]^T \) for both control schemes), the further \( \hat{\theta}_0 \) is away from the optimum \( \theta^* \) 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 \( \theta \) in (2.2.5) so that the predictive output term \( \hat{y}(k+d) \) 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 \( N_{3,5,15,P_0} \) 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 \( P_0 \), respectively. \( P_0 \) 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 \( I_s = [1 \ 1]^T \) 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 needed to be set equal to or greater than the maximum component of the parameter vector \( \theta \). 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 statistically 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 \( N^3 \_2,5,15,30 \) was used throughout the simulation.

Figure 4.7. Tip position response: \( \theta_t(k) \) & \( \theta_r(k) \) 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: $\theta_t(k)$ & $\tilde{\theta}_t(k)$
CHAPTER 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 convergence 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 wide 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 is the lack of understanding how the dimension and activation characteristics of a network is related to its accuracy and stability. These issues of the recursive algorithms have been addressed and elaborately analysed in (Kumar, 1990).
5.1 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).
BIBLIOGRAPHY


Ho, T., Ho, H., and Bialasiewicz, J. "Stochastic neural adaptive control for nonlinear time varying systems", in proceedings of the 1991 International Conference on Artificial Neural Networks in Engineering, St. Louis, Missouri, Nov. 10-12, 1991d.


APPENDIX

Simulation Program

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

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==Deterministic

% 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];
% THETAp = [.7 .5 1 .2 -.1 .3]; % minimum phase 2nd order plant
% A = [.7 .5 -3]; B = [1 .2 -.1 .3];
% THETAp = [.7 .5 1 .2 -.1 .3]; % minimum phase 3rd order
% THETAp = [.7 .5 1 .2 -.1 1 3 ]; % non-minimum phase 3rd; Id=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 -.0068 .0065 -.0063 ]; %missile
% THETAp = [-2.979 2.96 -.979 -.0047 .0094 -.0047]; %Submarine
load plant
num=numd'/dend(1);
den=dend'/dend(1);
\[ B = \text{num}; \]
\[ A = \text{den}(2:length(\text{den})); \]
\[ \% \ B = \text{numd}, A = \text{dend}(2,:); \]
\[ \text{THETA}_p = [A' \ B']; \]
\[ \text{na} = \text{length}(A); \text{nb} = \text{length}(B) - 1; \text{d} = 1; \]
\[ \text{nf} = \text{d} - 1; \]
\[ \text{ne} = \text{nf} + \text{nb}; \]
\[ \text{ng} = \text{na} - 1; \]
\[ \text{nc} = 0; \% \text{This assumes the noise has no dynamics, i.e.} \ C = 1; \]
\[ \text{P} = (\text{ng} + 1) + (\text{ne} + 1) + \text{nc}; \% \text{Dimension of THETA} \]
\[ \text{P}_0 = \text{na} + \text{nb} + 1; \% \text{Dimension of plant's THETA} \]
\[ \text{THETA}_E = 1 * \text{ones(P0,1)}; \]
\[ \% \text{load thetaest} \]
\[ \text{THETA}_0 = \text{THETA}_E; \]
\[ \text{yest} = 0; \text{w} = 0; \]
\[ \text{P} = \text{P}00 * \text{eye(P0,P0)}; \]
\[ \text{PSI}_1 = \text{PSI}0 * \text{ones(P0p,1)}; \]
\[ \text{PSI}_d = \text{zeros}(\text{P0},1); \]
\[ \text{K} = 1.5 * \text{ones}(\text{P0},1); \]
\[ \text{Yl} = \text{zeros}(\text{ng} + 1,1); \text{Ul} = \text{zeros}(\text{ne} + 1,1); \]
\[ \text{Yc} = \text{zeros}(\text{ng} + 1,1); \text{Uc} = \text{zeros}(\text{ne},1); \]
\[ \text{Ylp} = \text{zeros}(\text{na},1); \text{Ulp} = \text{zeros}(\text{nb} + 1,1); \]
\[ \text{Ud} = \text{zeros}(\text{d},1); \text{Yd} = \text{zeros}(\text{d},1); \% \text{delayed values of u, y and w} \]
\[ \text{Wd} = \text{zeros}(\text{d},1); \]
\[ \text{y} = 0; \text{u} = 0; \% \text{output y(k), input u(k)} \]
\[ \text{VARV} = 0; \% \text{Output noise variance} \]
\[ \text{MEANV} = 0; \]

\[ \text{n0} = 2; \text{n1} = 5; \text{n2} = 15; \text{n3} = \text{P0}; \% \text{Dimensions of Neural Network} \]
\[ \text{NETr} = \text{zeros}(\text{n1},1); \]
\[ \text{NETi} = \text{zeros}(\text{n2},1); \]
\[ \text{ NETj} = \text{zeros}(\text{n3},1); \]
\[ \text{Is} = \text{zeros}(\text{n0},1); \]
\[ \text{Or} = \text{zeros}(\text{n1},1); \]
\[ \text{Oi} = \text{zeros}(\text{n2},1); \]
\[ \text{Oj} = \text{zeros}(\text{n3},1); \]
\[ \text{ALPHA}_j = 0.03 * \text{ones(n3,1)}; \% \text{ALPHA}_i = 0.03 * \text{ones(n2,1)}; \]
\[ \text{ALPHA}_r = 0.03 * \text{ones(n1,1)}; \]
\[ \text{Hj} = 0 * \text{ones(n3,1)}; \% \text{Hi} = 0 * \text{ones(n2,1)}; \% \text{Hr} = 0 * \text{ones(n1,1)}; \]
\[ \text{Kj} = 3 * \text{ones(n3,1)}; \% \text{Ki} = 2 * \text{ones(n2,1)}; \% \text{Kr} = 2 * \text{ones(n1,1)}; \]
\[ \text{Wsr} = \text{rand(n0,n1)}; \]
\[ \text{Wri} = \text{rand(n1,n2)}; \]
\[ \text{Wij} = \text{rand(n2,n3)}; \]

\[ \text{mu} = 0.8; \]
\[ \text{lambda0} = 0.99; \]
\[ \text{lambda} = 0.995; \]
\[ \text{LAMBDA} = 1; \]
\[ \text{Pij} = 10 * \text{ones(n2,n3)}; \]
\[ \text{Pri} = 5 * \text{ones(n1,n2)}; \]
\[ \text{Psr} = 5.3 * \text{ones(n0,n1)}; \]
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]
% yest(k) = PSI(k)*THETAEst
%
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-nb-d)]
% for i=na-1:-1:1 Ylp(i+1)=Ylp(i);, end; Ylp(1)=Yd(1);
% for i=nb-1:-1:1 Ulp(i+1)=Ulp(i);, end; Ulp(1)=Ud(1);
% PSIp = [Ylp' Ulp'];

% PSI(k) = [y(k-d)...y(k-d-ng) u(k-d)...u(k-d-ne)]
% for i=ng-1:-1:1 Yl(i+1)=Yl(i);, end; Yl(1)=Yd(1);
% for i=ne-1:-1:1 Ul(i+1)=Ul(i);, end; Ul(1)=Ud(1);
% PSI = [Yl' Ul']
% 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;
    temp = (ALPHAr/2).*Netr+Hr;
    Or = Kr.*tanh(temp);
    Or(1)=1;

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

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

THETAEST = Oj;
if k == 1 save thetaest THETAEST, end
PSi = PSIid;

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;
PSlij = Oi * dj;
di = (Wij * dj) * Fdoti;
PSiiri = Or * dr;
Q = Fdoti * (Wij * (Fdotj * PSI));

if ALGij == 1 % Gradient
    Lij = mu * PSiij / LAMBDA;
end
if ALGij == 2 % Newton
    Sij = (PSlij * PSlij * Pij) + (lambdak * LAMBDA * ones(n2, n3));
    Lij = (Pij * PSlij) / Sij;
    Pij = (Pij - (Lij * Sij * Lij)) / lambdak;
end
if ALGij == 3 % Minimum Variance
    Sij = (PSlij * PSlij * Pij) + Re * ones(n2, n3);
    Lij = (Pij * PSlij) / Sij;
    Pij = Pij - (Lij * PSlij * 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+l)=Uc(i);, end; Uc(1)=u;  %[u(k-l) .. u(k-ne)]

ld = .1;
lld2= .06;
lld3=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)+lld2;  %Eqest(3) = Eqest(2)+lld3;
%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

% % == == == END OF FOR LOOP(k)
% % :::: END OF ITERATION ::::
% % ----- SYSTEMS GRAPHICS ------
% SYGRAF
% % == == == END OF SIMULATION

% % == == == END OF SYGRAF

% % == == == END OF SIMULATION

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

% % == == == END OF SYGRAF

% % == == == END OF SIMULATION