Monograph:
Morles, E.C. and Mort, N. (1991) Identification and Control of Dynamic Systems via Adaptive Neural Networks. Research Report. Acse Report 433 . Dept of Automatic Control and System Engineering. University of Sheffield

IDENTIFICATION AND CONTROL OF DYNAMIC SYSTEMS VIA ADAPTIVE NEURAL NETWORKS

BY

E. COLINA MORLES

AND

N. MORT

DEPARTMENT OF AUTOMATIC CONTROL

AND SYSTEMS ENGINEERING

UNIVERSITY OF SHEFFIELD

P.O. Box 600.

Mappin Street

Sheffield S1 4DU

Research Report Number 433

July 1991

Abstract

In this work we study some applications of multilayer perceptron neural networks to identify and to control certain types of dynamic systems.

Two different methods to update the weights of the neural network are explored:a variable structure control formulation, and a gradient descent approach. Both methods are digitally simulated and their performances in parameter identification are compared.

Also, we present an adaptive model reference control scheme, and an indirect self-tuning control scheme based on a multilayer perceptron neural network. Examples of the simulated responses for both schemes are obtained using different linear plants.

1 INTRODUCTION

The term artificial neural network is related to a nonlinear circuit composed of many interconnected simpler circuits called neurons. Its origin can be traced back to the work of McCulloch and Pitts (1943) with the first mechanistic interpretation of the neuron doctrine. It was not until 1957, with the work published by F. Rosenblat when the term perceptron was put forward in terms of an automaton, rather than a class of models of the central nervous system. [1]

During the first years of the sixties there was an enthusiastic interest in studying the capabilities of single layer perceptron networks to generalize any pattern similar to those presented in their training sets. This enthusiasm vanished with a monograph published by Minsky and Pepert in 1969 demonstrating the incapability of simple two layer perceptron networks to represent the behavior of a large class of functions. [2]

After 1980, the neural network discipline has been emerging with attempts at applications ranging from pattern recognition to parametric identification and control of complex dynamic systems. (see [3, 4, 5, 6, 7])

A fundamental aspect in the design of multilayer perceptron neural networks is the adaptation algorithm. For single layer networks, a well known procedure to minimize the mean square error between the desired output and the actual output is called the Delta Rule [8]. The generalized delta rule [8], usually known as the Back Propagation algorithm, is frequently used for updating the weights in multilayer networks. A drawback of the back propagation algorithm is the requirement that the nonlinear activation functions be differentiable. [9]

In this paper we use two approaches to change the weights of our neural network. In the first approach the weighting elements of the network are adjusted in such a way that the error between the actual and desired outputs satisfy a stable difference equation. In this case , unlike back propagation, the algorithm does not require differentiability along the network signal paths. [9]

This work was supported by the Universidad De Los Andes, (Merida, Venezuela), and by Maraven S.A. Petroleos De Venezuela

The second approach is based upon making the neural network behave like a gradient estimator. That is, with a constant input to the network, the weights are updated in the converse direction of the gradient of the squared prediction error with respect to the desired parameters. Mathematically, this can be written as

$$a_s(k+1) - a_s(k) = -\gamma \frac{\partial [e^T e]}{\partial a_s}, \tag{1}$$

where a_s is the parameter to be identified, e is the prediction error, and γ is a convergence factor. In this case, the stability properties of the algorithm can be guaranteed using Lyapunov functions. [10]

This report is organized as follows. In the next section we present, after introducing a basic definition, the variable structure control approach for the adaptation algorithm of a single layer perceptron network. This adaptation algorithm is generalized for a three layer perceptron network in section 3. Then, in section 4, we briefly review the gradient estimator in order to propose an alternative way of adjusting the network's weights. Section 5 contains digitally simulated examples of the performances of the implemented algorithms for parameter identification where both constant and time-varying parameters are analysed. In section 6 we present a model reference-neural network-based control scheme, and the responses obtained from simulating the scheme using different linear model plants. An indirect self-tuning-neural network-based control scheme is shown in section 7. Two linear plants are used to illustrate its performance. Section 8 contains conclusions and recommendations for further research.

2 VARIABLE STRUCTURE CONTROL FOR THE ADAPTATION ALGORITHM

Before we present the algorithm, it is important to consider the following definition.

Consider a dynamic system modelled by the following controlled difference equation of the state vector W(k), with a single output signal e(k) defined by the mapping h,

$$W(k+1) = f(W(k), U(k)), \tag{2}$$

$$e(k) = h(W(k)), \tag{3}$$

where $W(k) \in \Re^n$, $U(k) \in \Re^m$, and $e(k) \in \Re$. Consider the following level curve of the output map

$$h^{-1}(0) = \{W \in \Re^n : e = h(W) = 0\} \tag{4}$$

DEFINITION. [11]

A quasi-sliding mode is said to exist on h^{-1} if there exists a control law U(k) such that the motion of equations 2 and 3 satisfies the relation

$$|e(k+1)e(k)| < e^2(k), \tag{5}$$

for $e(k) \neq 0$.

Notice that condition 5 is equivalent to

$$|e(k+1)| < |e(k)|; \qquad e(k) \neq 0. \tag{6}$$

A single layer perceptron with an arbitrary nonlinear odd operator Γ is shown in figure 1. Notice that $W(k) = [w_1(k)w_2(k)...w_n(k)]^T$ is the value at time k, of the weight vector, $X = [x_1x_2...x_n]^T$ is the input to the neural network, $e(k) = Y_d - Y(k)$ is the present error, Y_d is the desired output, and the operator Γ satisfies $\Gamma(-X) = -\Gamma(X)$.

THEOREM. [9]

If the weights w_i of the single perceptron network, shown in figure 1, are updated according to the rule

$$W(k+1) = W(k) + \alpha\frac{e(k)\Gamma(X)}{X^T\Gamma(X)}, \tag{7}$$

$X^T\Gamma(X) \neq 0$, $0 < \alpha < 2$, then the error e(k) tends asymptotically to zero with the rate of convergence $(1-\alpha)$.

PROOF.

Note that

$$\begin{aligned} e(k+1) - e(k) &= Y_d - Y(k+1) - [Y_d - Y(k)] \\ &= -\sum_{i=1}^{n}[w_i(k+1) - w_i(k)]x_i \\ &= -X^T[W(k+1) - W(k)], \end{aligned}$$

now, using the rule 7 yields:

$$e(k+1) - e(k) = -\frac{X^T\alpha e(k)\Gamma(X)}{X^T\Gamma(X)} = -\alpha e(k),$$

if $X^T\Gamma(X) \neq 0$. Then $e(k+1) = (1-\alpha)e(k)$. Thus, if $0 < \alpha < 2$ then $\lim_{k\to\infty} e(k) = 0$.

Observe that if Γ is the identity operator then equation 7 is the same as the Delta rule. Note that the error equation satisfies relation 5 for the existence of a quasi-sliding mode.

3 ADAPTATION ALGORITHM FOR THREE LAYER NETWORKS

In this section we shall extend the previous adaptation rule to three layer perceptron neural networks.This generalization of the algorithm is particular important because:

- It has been proved that a neural network with a hidden layer (i.e. a three layer network) can be used to approximate any mapping from $\Re^n$ to $\Re^m$. [12]
- The back-propagation algorithm has the disadvantages of lacking the convergence property, requiring a large number of iteration for good learning,[13], and requiring continuous differentiability of the nonlinearities along the network path. [9]

Consider the three layer network depicted in figure 2, and its schematic representation shown in figure 3. It is worth mentioning that in the proposed adaptation algorithm for this configuration we can include activation functions which are of the hard limiter type, saturation, or any other nonlinear odd operator.

The vector $Y_o(k)$ of the output components $yo_k(k)$ can be represented as

$$Y_o(k) = [W1(k)]^T Z1(k), \tag{8}$$

where $W1(k) \in \Re^{n1xno}$, and $Z1(k) = \Gamma(Y1(k))$. Similarly, the vector Y1(k) of the components $y1_n(k)$ is

$$Y1(k) = [W2(k)]^T Z2(k), \tag{9}$$

with $W2(k) \in \Re^{n2xn1}$, and $Z2(k) = \Gamma(Y2(k))$. Finally the vector Y2(k) of the components $y2_m(k)$ is

$$Y2(k) = [WI(k)]^T X, \tag{10}$$

where $WI(k) \in \Re^{nixn2}$, and $X = [x_1 x_2 ... x_{ni}]^T$.

Note that the nonlinear operator Γ does not have to be a diagonal one. The error vector E(k) at time k is given by

$$E(k) = [e_1(k)e_2(k)...e_{no}(k)]^T = Y_d - Y_o(k), \tag{11}$$

Y_d being the desired output vector.

The weight updates are represented by the following equations:

$$W1(k+1) = W1(k) + U1(k) \tag{12}$$

$$W2(k+1) = W2(k) + U2(k) \tag{13}$$

$$WI(k+1) = WI(k) + UI(k) \tag{14}$$

LEMMA.

The error vector E(k) satisfies the following difference equation as a function of the input layer, hidden layer, and output layer matrix update weights UI(k), U2(k), and U1(k):

$$\begin{aligned} E(k+1)-E(k) &= [W1(k)]^T\{\Gamma(Y1(k))-\Gamma\{[W2(k) \\ &+ U2(k)]^T\Gamma[Y2(k)+[UI(k)]^TX]\}\} \\ &- [U1(k)]^T\Gamma\{[W2(k)+U2(k)]^T\Gamma[Y2(k) \\ &+ [UI(k)]^TX\} \end{aligned} \tag{15}$$

PROOF.

The proof is straightforward by substituting equations 8, 9, 10, 12, 13, and 14 into $E(k+1)-E(k)=Y_o(k)-Y_o(k+1)$.

THEOREM.

If the weight update matrices UI(k), U2(k), and U1(k) are respectively chosen as

$$UI(k) = -\frac{2\Gamma(X)[Y2(k)]^T}{X^T\Gamma(X)}; \qquad X^T\Gamma(X)\neq 0, \tag{16}$$

$$U2(k) = -\frac{2\Gamma(Z2(k))[Y1(k)]^T}{[Z2(k)]^T\Gamma(Z2(k))}; \qquad [Z2(k)]^T\Gamma(Z2(k))\neq 0, \tag{17}$$

$$U1(k) = \frac{\Gamma(Z1(k))[AE(k)]^T}{[Z1(k)]^T\Gamma(Z1(k))}; \qquad [Z1(k)]^T\Gamma(Z1(k))\neq 0, \tag{18}$$

then the error vector E(k) satisfies the following asymptotically stable difference equation

$$E(k+1)=(I-A)E(k), \tag{19}$$

where I is the identity matrix, and A is an $n_o x n_o$ diagonal matrix given by

$$A=diag\{\alpha_1\alpha_2...\alpha_{n_o}\}, \tag{20}$$

such that $|1-\alpha_k|<1$, for k=1,2,....n_o.

PROOF.

Substituting the transposes of 16. 17, and 18 into the error difference equation 15 of the lemma yields equation 19.

4 GRADIENT ESTIMATOR

In this section we shall show how to use the three layer neural network depicted in figure 2 in order to emulate the behavior of a gradient estimator. we will use the same weight update matrices of the previous theorem, except that the input X for the neural network will be a unit step, and equation 18 will be modified to represent the gradient estimator.

Firstly we need an estimation model to relate the available data to the unknown parameters. A general linear model is the equation [10]

$$\hat{Y}(t) = W(t)\hat{a}(t), \tag{21}$$

where $\hat{Y}$ is the predicted output at time t, $\hat{a}(t)$ is the estimated parameter vector, and W(t) is an nxm signal matrix obtained from measurements of the system signals.

The prediction error e is the difference between the predicted output and the measured output Y,

$$e(t) = \hat{Y}(t) - Y(t). \tag{22}$$

The idea in gradient estimation is that the parameters should be updated so that the prediction error is reduced. This is

$$\dot{\hat{a}}(t) = -\alpha \frac{\partial[e^T(t)e(t)]}{\partial \hat{a}}, \tag{23}$$

where α is the estimator gain. In view of 21 and 22

$$\dot{\hat{a}}(t) = -\alpha W(t)e(t) \tag{24}$$

Equation 24 can be rewritten as

$$\begin{aligned} \hat{a}(k+1) - \hat{a}(k) &= -T\alpha W(k)e(k) \\ &= -\gamma W(k)e(k), \end{aligned} \tag{25}$$

with T the sampling time, and $\gamma = T\alpha$. This gradient estimator is always stable, as can be seen using the Lyapunov function candidate

$$V = \tilde{a}^T\tilde{a}, \tag{26}$$

with $\tilde{a} = \hat{a} - a$, and noting that $\dot{\tilde{a}} = -\alpha W^T W\tilde{a}$. The derivative $\dot{V}$ is easily found to be

$$\dot{V} = -2\alpha\tilde{a}^T W^T W\tilde{a} \leq 0 \tag{27}$$

Despite the stability property of the gradient estimator, the convergence of the estimate to the true value depends on the exciting signal. [10]

5 PARAMETER IDENTIFICATION EXAMPLES

Here we shall present some examples of the results obtained from implementing the two previous adaptation algorithms for parameter identification purposes. The objective is to identify a single parameter or function in dynamic systems of the type

$$\dot{x}(t) = f(t)u(t) \tag{28}$$

$$\dot{x}(t) = g(x)u(t) \tag{29}$$

$$\dot{x}(t) = h(x,t) + bu(t) \tag{30}$$

The corresponding identification models used to achieve our objective are respectively:

$$\dot{\hat{x}}(t) = \hat{f}(t)u(t) + k1(x(t) - \hat{x}(t)) \quad (31)$$

$$\dot{\hat{x}}(t) = \hat{g}(t)u(t) + k2(x(t) - \hat{x}(t)) \quad (32)$$

$$\dot{\hat{x}}(t) = \hat{h}(x,t) + bu(t) + k3(x(t) - \hat{x}(t)) \quad (33)$$

Figure 4 shows the identification scheme.Figure 8 illustrate the results obtained, with the variable structure control approach algorithm, considering a system of the type shown in equation 28. In particular, figure 8-A shows the estimate of the function and the state tracking error when the function f(t) was,

$$f(t) = \begin{cases} 2 & if \quad 0 \leq t < 1 \\ 0.4Sin(10t - 1) & if \quad 1 \leq t < \infty. \end{cases} \quad (34)$$

Figure 8-B shows the estimate of the parameter and the state tracking error when the the function f(t) was,

$$f(t) = 0.35 \qquad 0 \leq t < \infty. \quad (35)$$

Figure 9 shows the results obtained considering a system of type 29. The function identified is

$$g(x) = \sqrt{|x|}, \quad (36)$$

which was part of the system

$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = \sqrt{|x_1|}u(t)$$

Figures 10-A, 10-B, and 10-C illustrate the performance of the neural network when used in a system of the type 30 described by

$$J\ddot{x}(t) + B\dot{x}(t) + Fx(t) = u(t),$$

where J is the unknown moment of inertia, and B and F are the known viscous and compliance coefficients, respectively.

Figures 11-A and 11-B show the estimation of a constant and a time varying parameter when the the gradient estimator algorithm was implemented . In these two cases the system was persistently excited with a step function. In figure 12 can be seen different convergence rates of the estimate depending upon the selection of the estimation gain.

If the exciting signal for the system is not persistently exciting,[10], the convergence of the estimate to the true value is lost, as illustrated in figure 13. Note that, despite the lack of accuracy in the estimation of the real value of the parameter, the state tracking error remains small.

6 MODEL REFERENCE-NEURAL NETWORK-BASED CONTROL SCHEME

The aim of this section is to present a model reference control scheme where the adjustable parameters of the controller are supplied by a neural network. It will be assumed that the structure of the plant is known, although its parameters are unknown. Both the variable structure control approach algorithm and the gradient descent algorithm will be used to update the weights of the network.

The proposed scheme, for the variable structure control approach algorithm is shown in figure 5. Figure 6 shows the scheme used when the weights were updated implementing the gradient estimation algorithm.

Mathematically speaking the results can be illustrated with the following examples:

6.1 Variable Structure Control Approach Algorithm

Let the plant be described by

$$\dot{y}(t) = -a_p y(t) + b_p u(t), \tag{37}$$

a_p, b_p unknown.

Let the reference model be described by

$$\dot{y}_m(t) = -a_m y_m(t) + b_m r, \tag{38}$$

a_m, b_m known; r a step function.

Then

$$e(t) = y(t) - y_m(t) \tag{39}$$

$$\dot{e}(t) = \dot{y}(t) - \dot{y}_m(t) \tag{40}$$

It will be assumed that a_p and b_p are constant in $t \in [kT, (k+1)T]$. Notice that

$$\begin{aligned} \dot{e}[(k+1)T] - \dot{e}[(k)T] &= -a_m \{e[(k+1)T - e[(k)T]\} \\ &+ \tilde{a}\{y[(k+1)T] - y[(k)T]\} \\ &+ b_p \{u[(k+1)T] - u[(k)T]\}, \end{aligned} \tag{41}$$

with $\tilde{a} = a_m - a_p$. Now, if we select

$$u[(k+1)T] - u[(k)T] = -Sgn(b_p)\alpha e[(k)T], \tag{42}$$

where $\alpha = \frac{constant}{T}$, then

$$\ddot{e}|_{kT} + a_m \dot{e}|_{kT} + b_p \alpha e|_{kT} = \tilde{a}\dot{y}|_{kT} \tag{43}$$

Obviously, for all $kT \in [0, \infty)$ we have the second order differential equation

$$\ddot{e}(t) + a_m \dot{e}(t) + b_p \alpha e(t) = \tilde{a}\dot{y}(t) \tag{44}$$

Observe that the Laplace transform of the above equation yields

$$E(s) = \frac{\tilde{a}sY(s)}{s^2 + a_m s + b_p \alpha}, \tag{45}$$

and if Y(s) is bounded, then

$$\lim_{s \to 0} sE(s) = 0 \tag{46}$$

Figure 14 illustrates the results obtained when the unknown plant parameters were $a_p = 5$ and $b_p = 10$, and the input to the reference model was a changing step function. Observe the convergence of the plant output to the reference model output. Figure 15 shows the neural network adaptive control action.

6.2 Gradient Estimation Algorithm

Consider the plant and the reference model described by equations 39 and 40. Let the control law be defined by

$$u(t) = k_1 r + k_2 y(t), \tag{47}$$

then the closed-loop system will be

$$\dot{y}(t) = -(a_p - k_2 b_p) y(t) + k_1 b_p r \tag{48}$$

Notice that if k_1 and k_2 were selected as $k_1^* = \frac{b_m}{b_p}$, and $k_2^* = \frac{a_p - a_m}{b_p}$ then the closed-loop behavior would correspond to the desired reference model.

Since a_p and b_p are unknown, we have to estimate k_1 and k_2 on line. Let the parameter estimation errors be:

$$\tilde{k}_1 = k_1 - k_1^*, \tag{49}$$

$$\tilde{k}_2 = k_2 - k_2^*. \tag{50}$$

If we define

$$e(t) = y(t) - y_m(t), \tag{51}$$

then

$$\dot{e}(t) + a_m e(t) = b_p \tilde{k}_2 y(t) + b_p \tilde{k}_1 r$$

Finally, if we select

$$\dot{k}_1(t) = -Sgn(b_p)\gamma e(t) r, \tag{54}$$

$$\dot{k}_2(t) = -Sgn(b_p)\gamma e(t) y(t), \tag{55}$$

then

$$\lim_{t \to \infty} e(t) = 0 \tag{56}$$

Equations 56 and 57 can be rewritten in their discrete time versions as

$$k_1[(k+1)T] = k_1[(k)T] - Sgn(b_p)\alpha e(kT)r(kT), \tag{57}$$

$$k_2[(k+1)T] = k_2[(k)T] - Sgn(b_p)\alpha e(kT)y(kT), \tag{58}$$

where $\alpha = \frac{constant}{T}$ is the convergence rate. Figures 16, 17, and 18 show simulation results obtained using the above algorithm to update the weights of the neural network. The plant under consideration here was the same of the previous example, except for the amplitude of the reference signal.

7 SELF-TUNING-NEURAL NETWORK-BASED CONTROL SCHEME

In this section, we propose a self-tuning control scheme that uses a neural network for parameter estimation purposes. Figure 7 shows the suggested scheme. For dynamic systems of the classes described by 28 and 29, the estimation of the single parameter or function can be done using a neural network where the weight-update algorithm is of the variable structure control approach type. Figures 19 and 20 show the parameter estimate, state tracking performance, dynamic behavior, and self-tuning control action for the position control of a mass on frictionless surface. In this particular case, the plant is described by

$$\ddot{x}(t) = \frac{1}{m}u(t), \tag{59}$$

with u being the external force, and m the unknown mass.

The desired closed-loop behavior is represented by

$$\ddot{x}(t) + a_1\dot{x}(t) + a_2x(t) = a_2r, \tag{60}$$

where a_1 and a_2 are known, and r is a step function. In order to achieve the desired specification, the external force u should be selected as

$$u(t) = m[a_2(r - x(t)) - a_1\dot{x}(t)]. \tag{61}$$

Notice that in order to be able to implement the above control law, the parameter m must be estimated on-line.

Let us consider a final example of controlling the angular position of a robotic manipulator using a gradient estimation algorithm for updating the neural network weights. The linearized version of the plant's dynamics may be described by

$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = -\frac{F}{J}x_1(t) - \frac{B}{J}x_2(t) + \frac{1}{J}u(t), \tag{62}$$

where the parameters F, B, and J represent the known compliance and viscous coefficients, and the unknown moment of inertia respectively.

Let the desired closed-loop behavior be described by

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -w_0^2 x_1(t) - 2\xi w_0 x_2(t) + (w_0^2 - \frac{F}{J})u_{ref}, \end{aligned} \tag{63}$$

where w_0 and ξ are the desired natural frequency and damping coefficient respectively. In order to achieve the desired specifications, the control u must be selected as

$$u(t) = (F - Jw_0^2)[x_1(t) - u_{ref}] + (B - 2J\xi w_0)x_2(t). \tag{64}$$

Note the dependence of u with respect to the unknown moment of inertia J. The identification model used for parameter estimation purposes is represented by

$$\begin{aligned} \dot{z}_1(t) &= x_2(t) + k_1[x_1(t) - z_1(t)] \\ \dot{z}_2(t) &= -\frac{F}{J_s}x_1(t) - \frac{B}{J_s}x_2(t) + \frac{1}{J_s}u(t) + k_2[x_2(t) - z_2(t)], \end{aligned} \tag{65}$$

where J_s is the estimated moment of inertia, and k_1 and k_2 are observation gains. Let us define the predicted error

$$e(t) = x_2(t) - z_2(t). \tag{66}$$

Observe that:

$$\begin{aligned} \dot{e}(t) &= \dot{x}_2(t) - \dot{z}_2(t) \\ &= -(\frac{F}{J} - \frac{F}{J_s})x_1(t) - (\frac{B}{J} - \frac{B}{J_s})x_2(t) + (\frac{1}{J} - \frac{1}{J_s})u(t) - k_2 e(t) \end{aligned} \tag{67}$$

This is

$$\dot{e}(t) + k_2 e(t) = -(\frac{F}{J} - \frac{F}{J_s})x_1(t) - (\frac{B}{J} - \frac{B}{J_s})x_2(t) + (\frac{1}{J} - \frac{1}{J_s})u(t). \tag{68}$$

The gradient descent algorithm suggests that

$$(\frac{\dot{1}}{J_s})(t) = -\gamma e(t)u(t) \tag{69}$$

Note that if $(\frac{1}{J_s}) \rightarrow (\frac{1}{J})$ then $e \rightarrow 0$ as $t \rightarrow \infty$. Implementing the discrete-time version of equation 69 for updating the weights of the neural network produces the results shown in figures 21, 22, and 23. It is worth mentioning that the parameters of the plant were assumed to be constant, otherwise by virtue of the non-persistently exciting characteristic of the generated control signal, it would have been impossible to estimate a time-varying parameter accurately.

8 CONCLUSIONS

We have presented some applications of multilayer perceptron neural netwoks to identify and to control certain types of dynamic systems. Two approaches were followed to update the network's weights: a variable structure control formulation and a gradient descent algorithm. Both approaches were digitally implemented and the performance of the resulting neural networks were checked in terms of parameter estimation and control design for linear systems.

Two model reference-neural network-based control schemes were proposed and their performances were digitally tested using examples of linear systems. In the first scheme, the neural network acts as the controller for the given plant. In the second scheme the neural network is used to update the controller parameters in such a way that the error between the reference model output and the plant output goes to zero.

Finally, an indirect self-tuning-neural network-based control scheme was presented, and its performance to control two different linear systems was digitally tested.

It should be pointed out that the variable structure control approach algorithm presented here assumes a constant desired output vector Y_d -(equation 11)-. Also it should be noted that the linear systems considered in the control schemes are noise free. Our final coment is related to the selection of the sampling time for the neural network. Its selection must take into consideration the speed at which the system responds to an external input.

Research is now underway to apply the proposed adaptation algorithms to estimation and control of nonlinear dynamic systems.

References

[1] Nagy G. "Neural Networks-Then And Now".
IEEE Trans. On Neural Networks, Vol. 2, Number 2, pp 316-318, March 1991

[2] Minsky M.; Papert S. "Perceptrons: An Introduction To Computational Geometry.
The M.I.T. Press, Cambridge, Ma. 1969.

[3] Pao Y.H. "Adaptive Pattern Recognition And Neural Networks".
Addison-Wesley, Reading, Mass. 1989

[4] Haykin S. "Introduction To Adaptive Filters".
MacMillan publishing Co., New York 1984

[5] Widrow B.; Winter R. "Neural Nets For Adaptive Filtering And Adaptive Pattern Recognition".
IEEE Computer, Vol. 21, Number 3, pp 25-39, 1988

[6] Reynold Chu S.; Shoureshi R.; Tenorio M. "Neural Networks For System Identification".
IEEE Control Systems Magazine, pp 31-34, April 1990

[7] Narendra K.; Parthasarathy K. "Identification And Control Of Dynamical Systems Using Neural Networks".
IEEE Trans. On Neural Networks, Vol. 1, Number 1, pp 4-27, March 1990

[8] Rumelhart D.E.; McClelland J.L. "Parallel Distributed Processing: Explorations In The Macrostructure Of Cognition".
Chapter 8, A Bradford Book, The MIT Press, Cambridge 1986

[9] Żak S.H.; Sira Ramirez H. "On The Adaptation Algorithms For Generalized Perceptrons".
8^{th} International Congress Of Cybernetics And Systems, Hunter College, Cuny, New York, June 1990

[10] Slotine J.E.; Li W. "Applied Nonlinear Control".
Chapter 8, Prentice Hall International Editions 1991

[11] Sarpturk S.Z.; Istefanopulos Y.; Kaynak O. "On The Stability Of Discrete-Time Sliding Mode Control Systems".
IEEE Trans. On Automatic Control, Vol. AC-32, Number 10, pp 930-932, October 1987

[12] Hornik K.; Stinchcombe M.; White H. "Multilayer Feedforward Networks Are Universal Approximators".
Neural Networks, Vol. 2, pp 359-366, 1989.

[13] Levine E.; Gewirtzman R.; Inbar G. "Neural network architecture For Adaptive System Modelling And Control".
Neural Networks Vol. 4, pp 185-191, Pergamon Press 1991

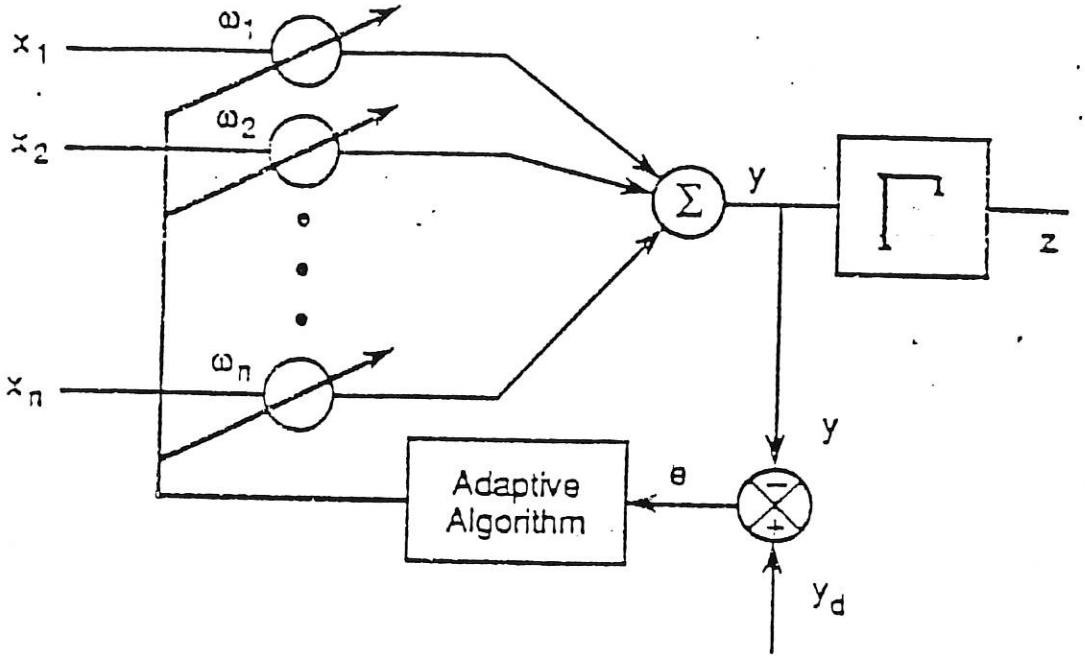

Figure 1: Single Layer Perceptron.

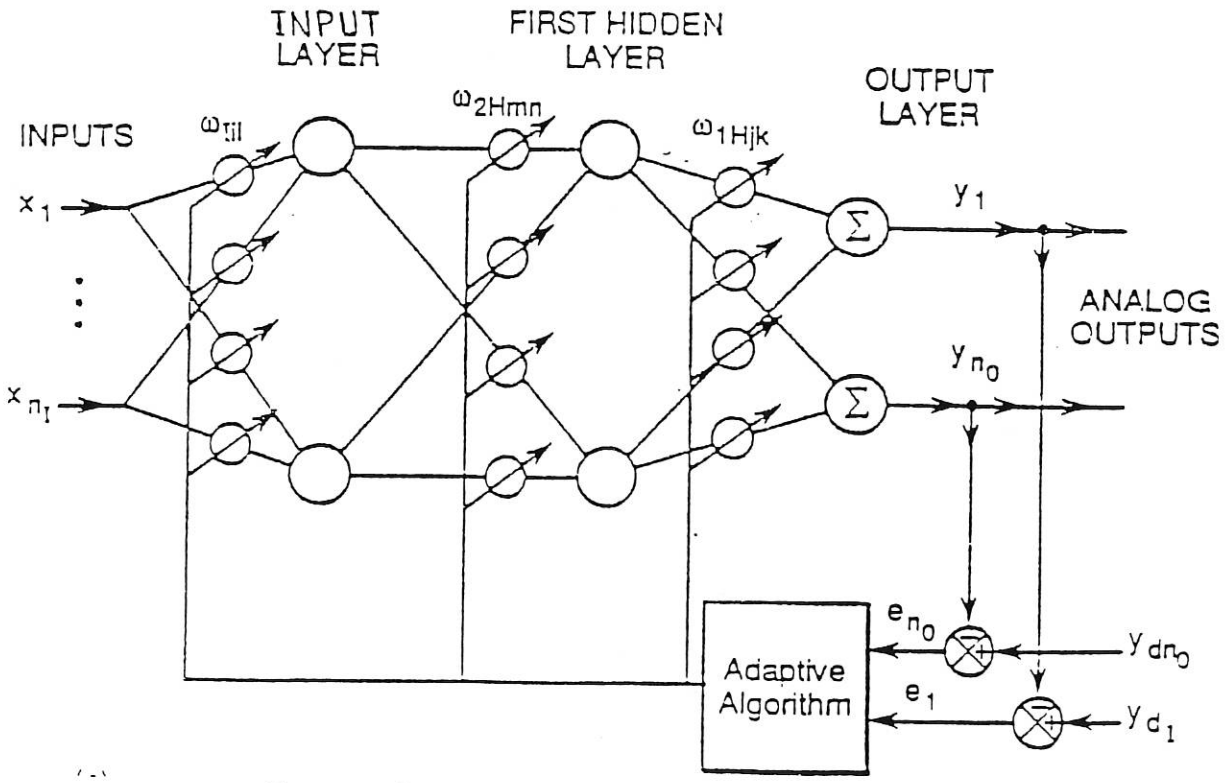

Figure 2: Three Layer neural Network.

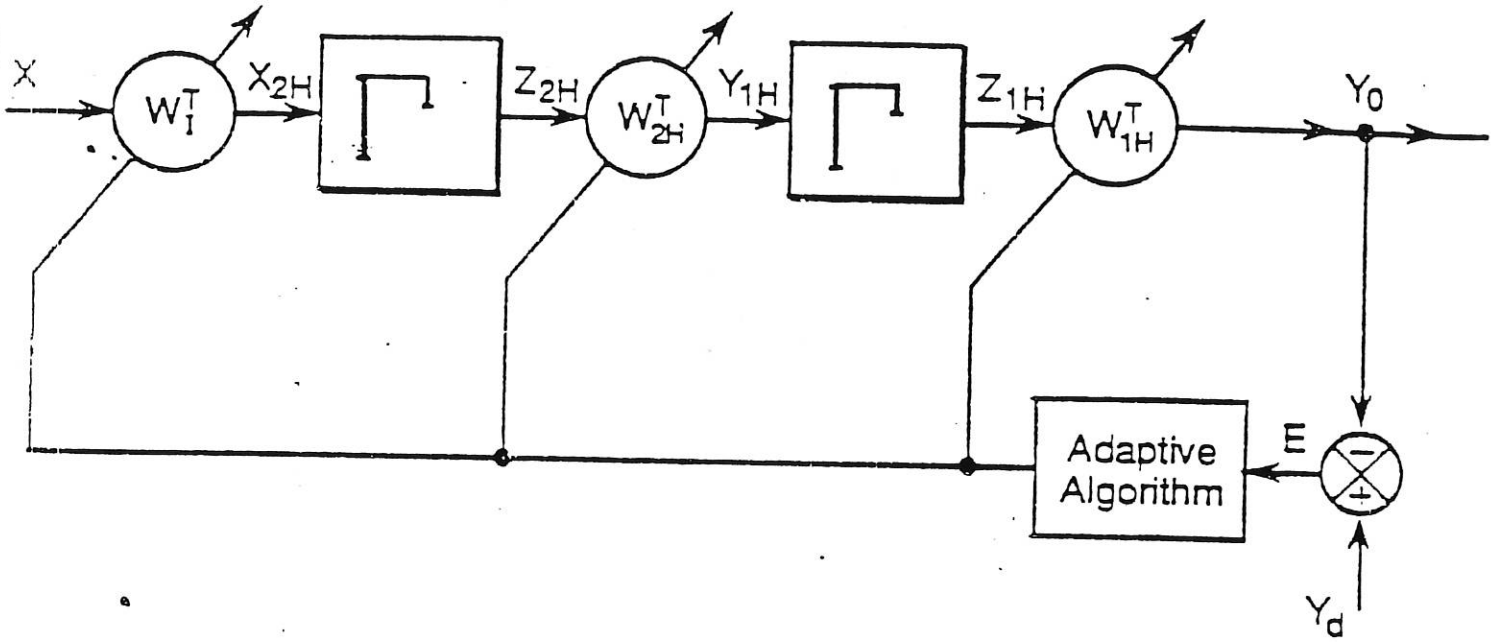

Figure 3: Schematic Representation Of The Three Layer Neural Network.

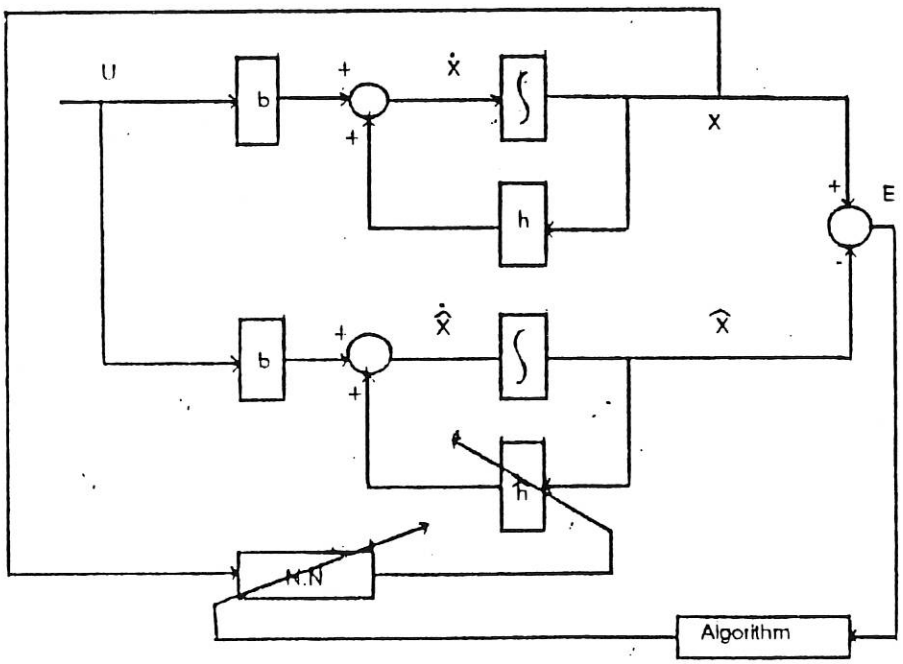

Figure 4: Neural Network-Based Identification Scheme.

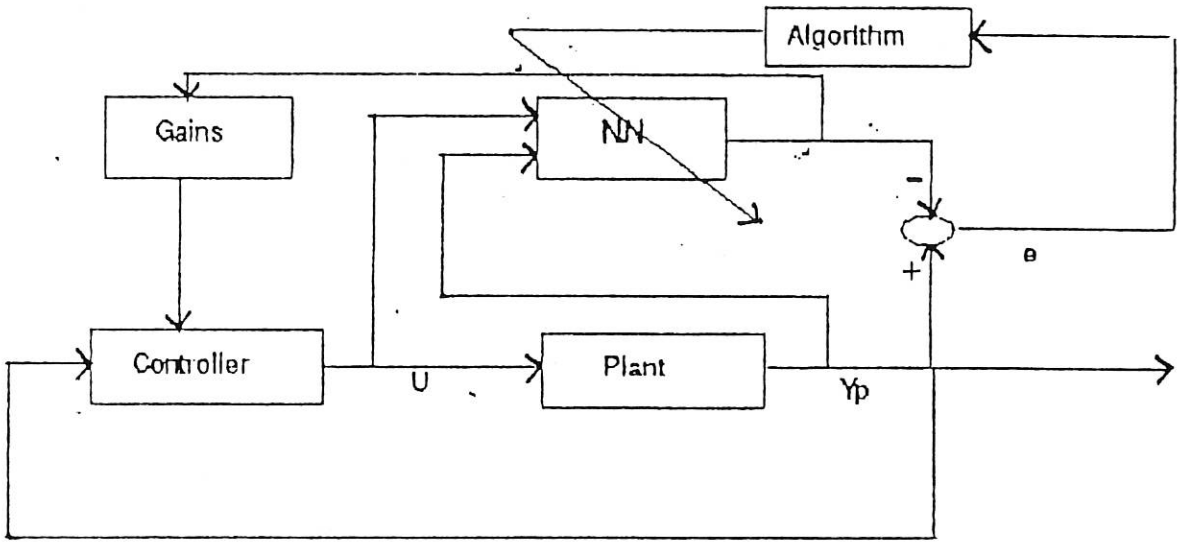

Figure 7: Self-Tuning-Neural Network-Based Control Scheme.

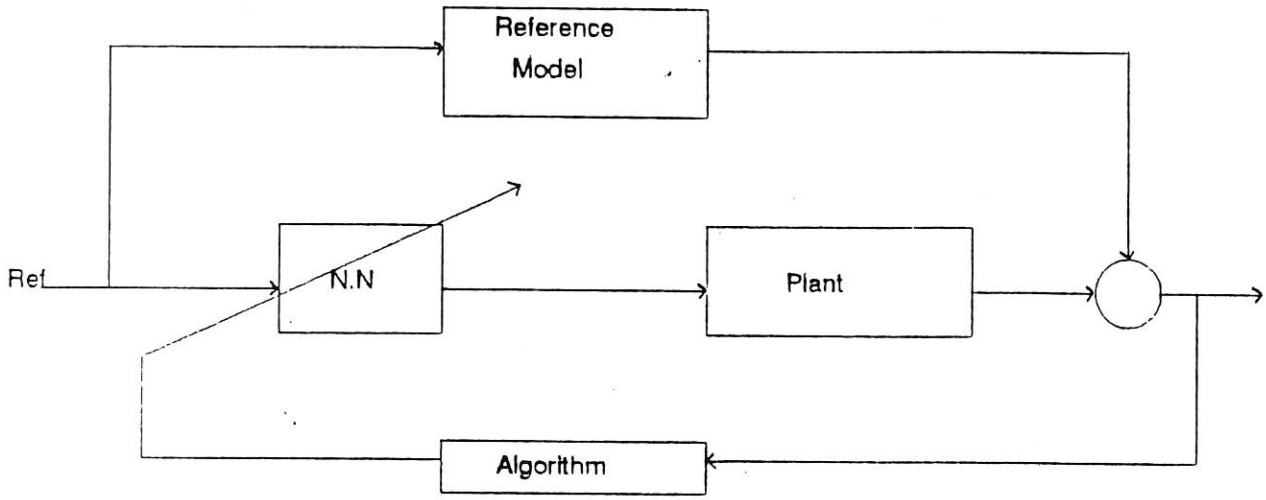

Figure 5: Model Reference-Neural Network-Based Control Scheme.
Variable Control Approach Algorithm.

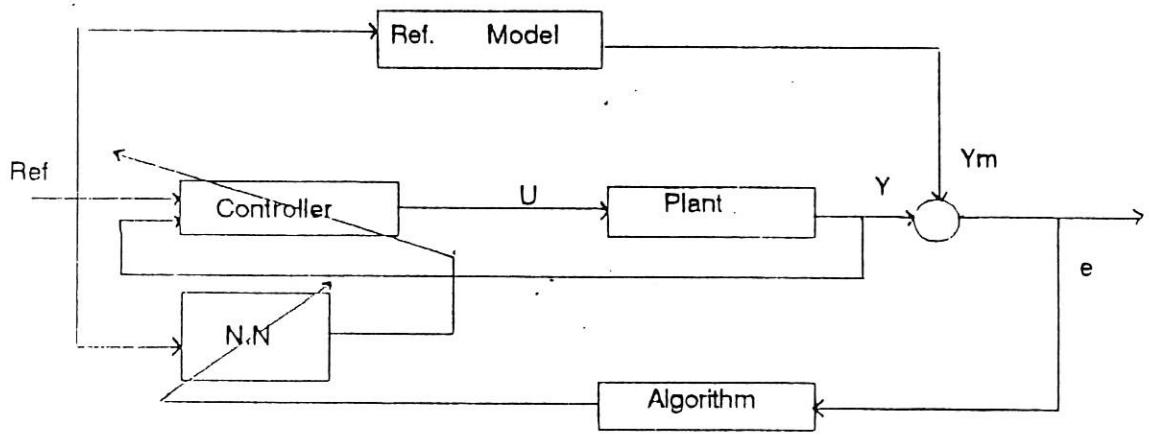

Figure 6: Model Reference-Neural Network-Based Control Scheme.
Gradient Estimation Algorithm.

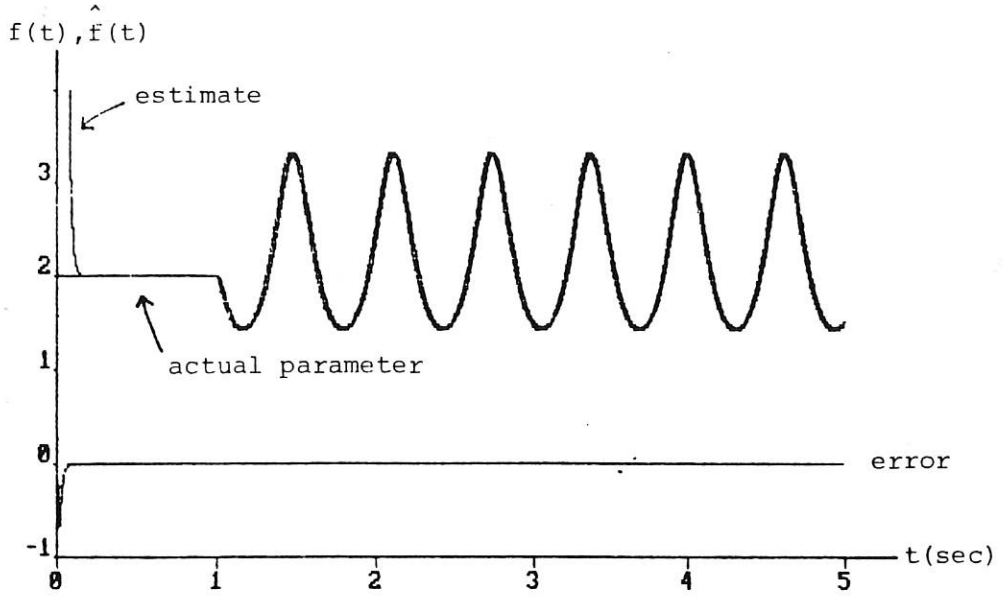

Figure 8-A.

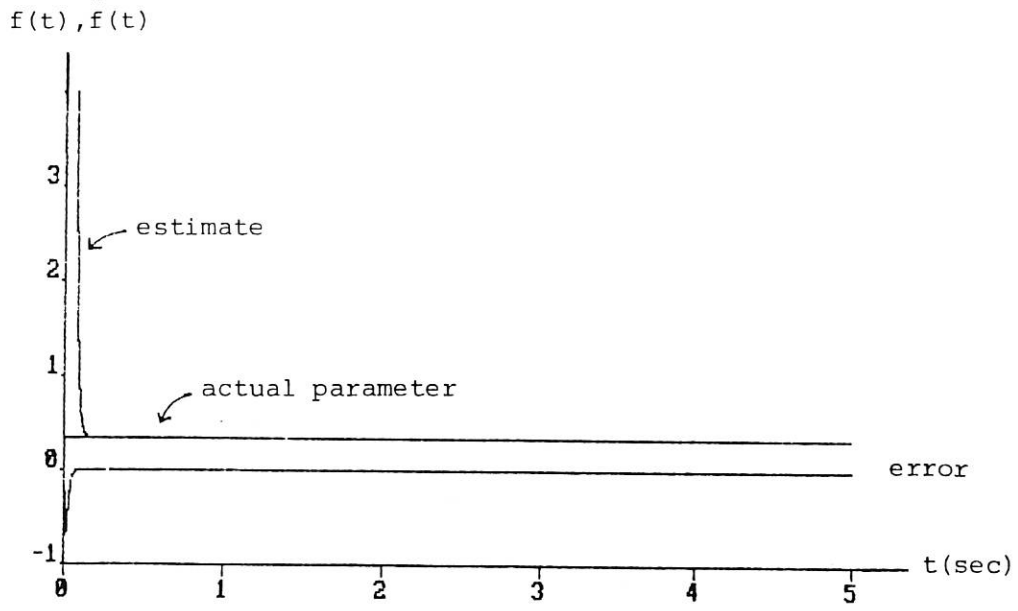

Figure 8-B.

Figure 8.
8-A Time-varying parameter estimate and state tracking error.
8-B Constant parameter estimate and state tracking error.

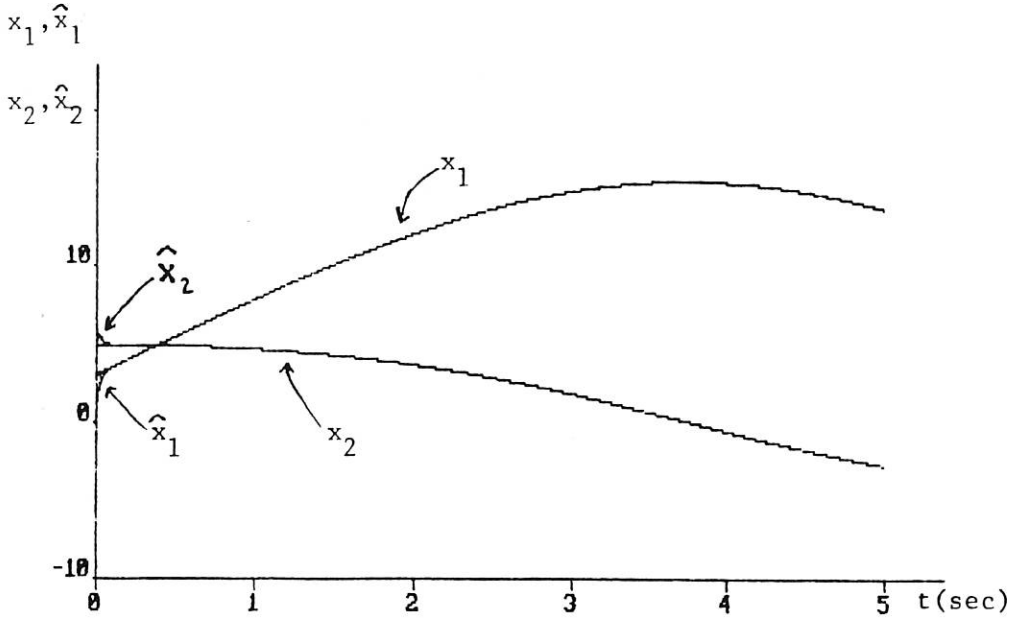

Figure 9-A.

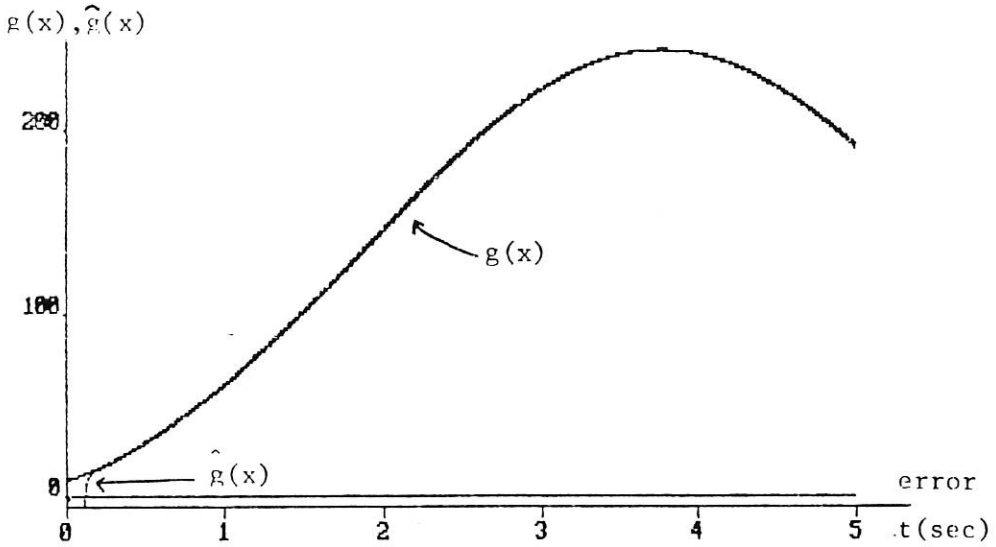

Figure 9-B.

Figure 9.
9-A State variables and their estimates.
9-B Estimate of sqr(abs(x1)) and state tracking error.

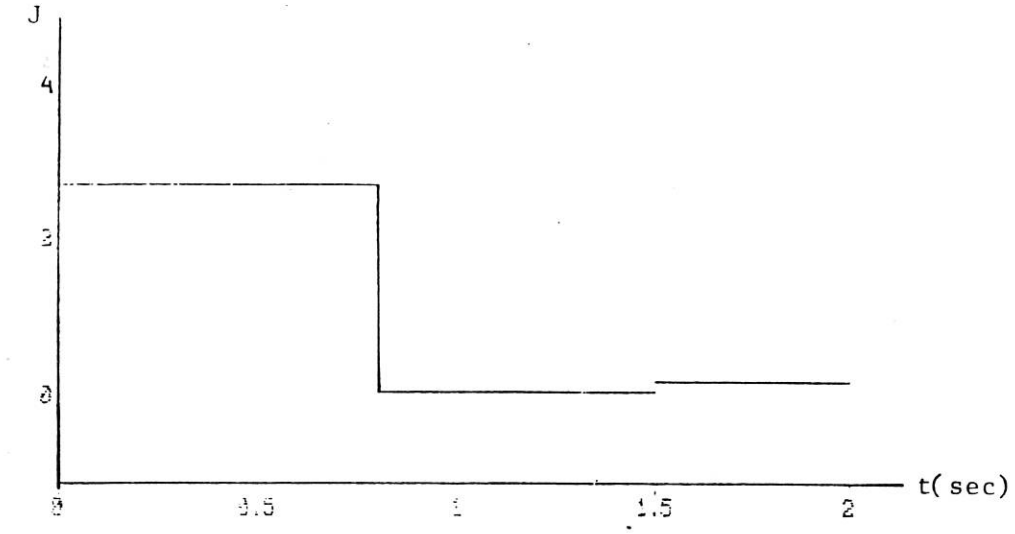

Figure 10-A.

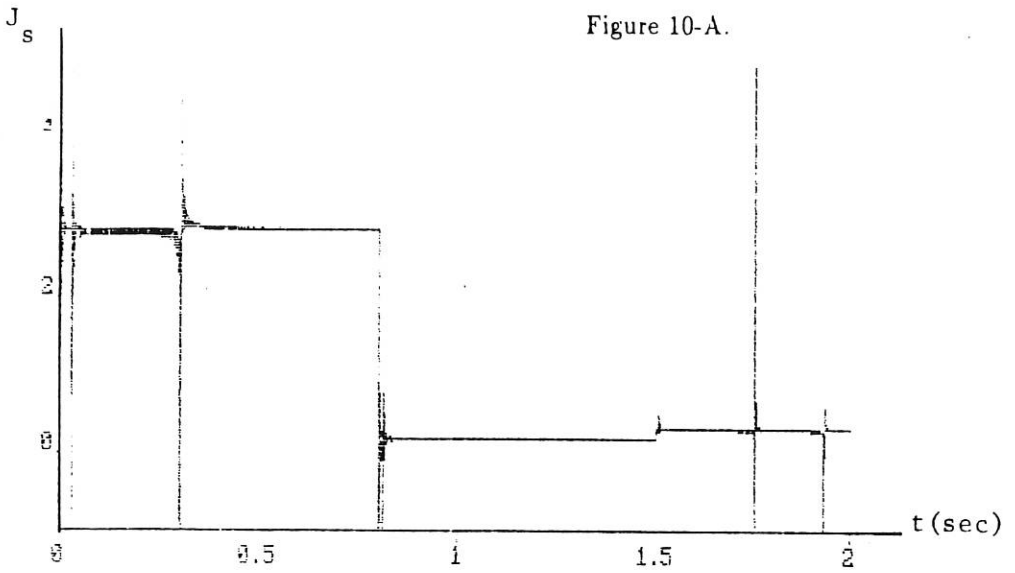

Figure 10-B.

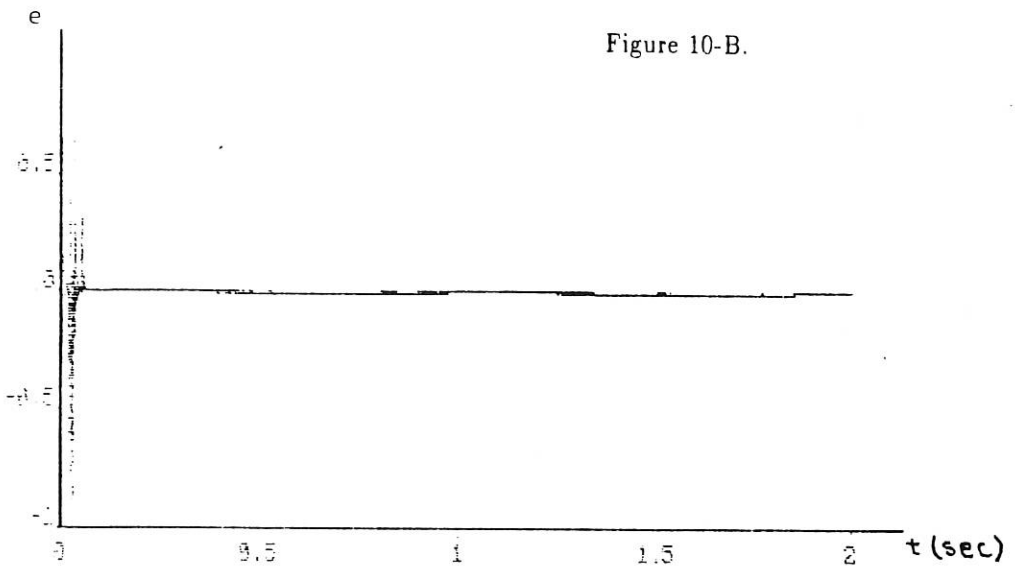

Figure 10-C.

Figure 10.
10-A Actual moment of inertia.
10-B Estimated moment of inertia.
10-C State tracking error.

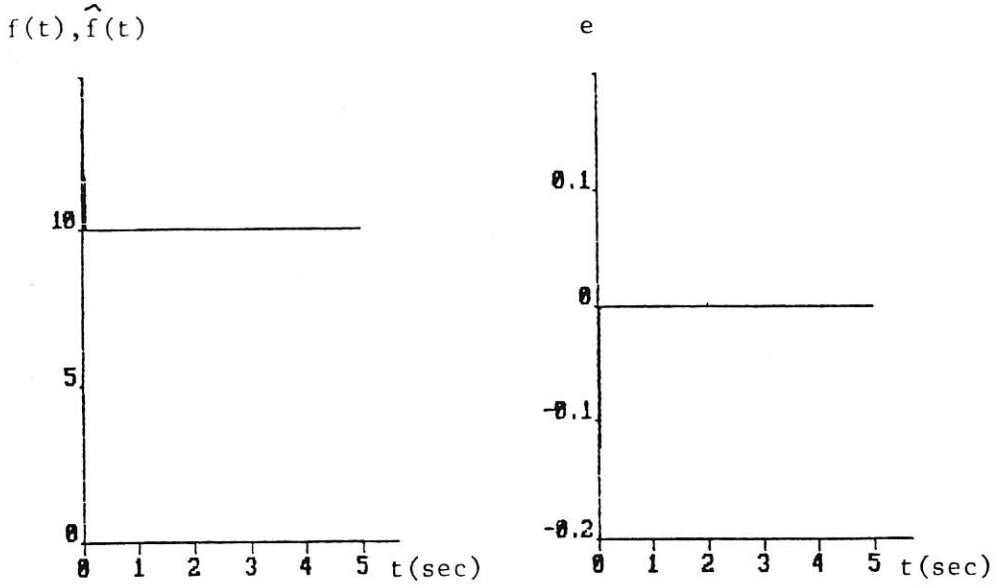

Figure 11-A.

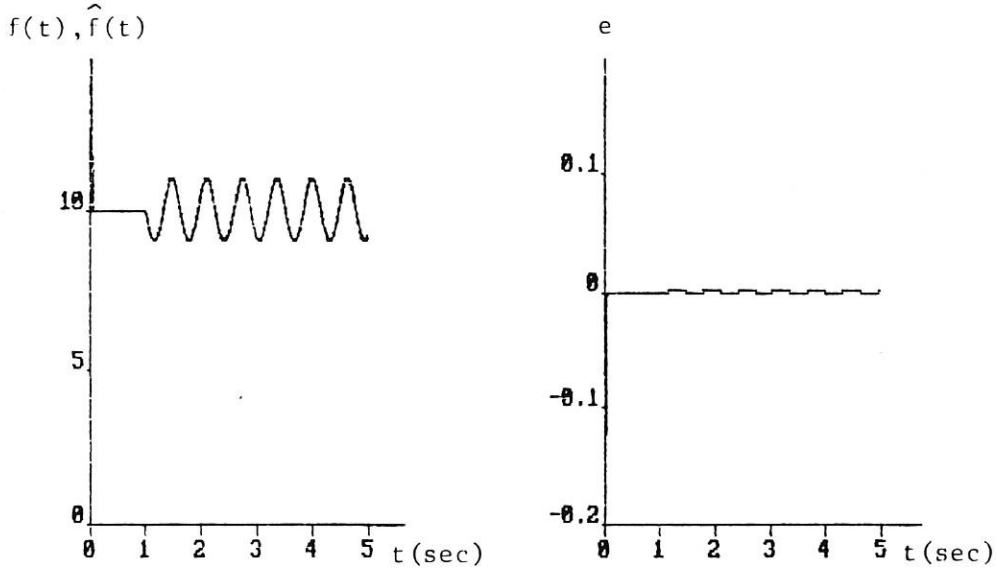

Figure 11-B.

Figure 11.
11-A Constant parameter estimation and state tracking error.
11-B Time-varying parameter estimation and state tracking error.

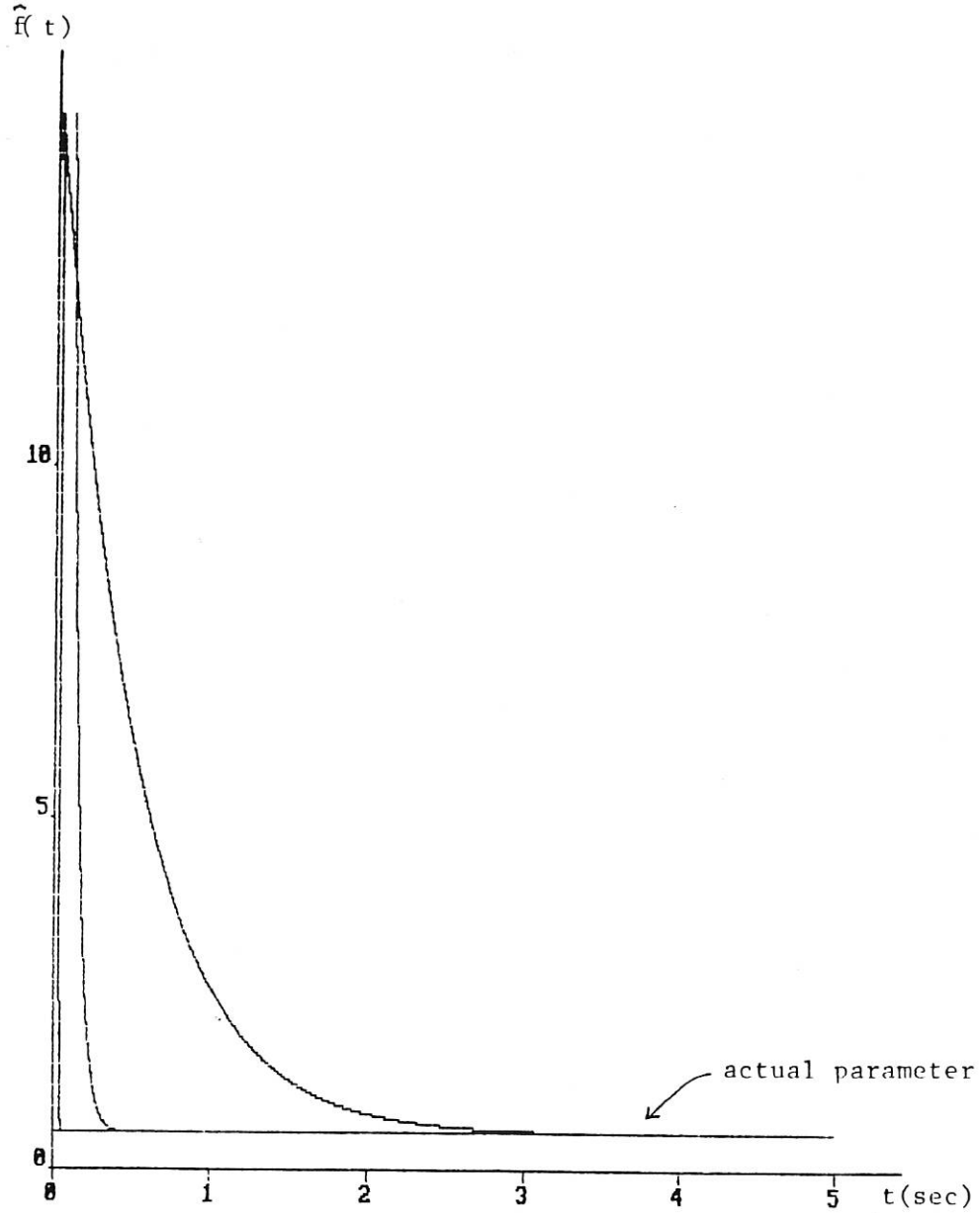

Figure 12: Parameter estimation for different estimation gains.

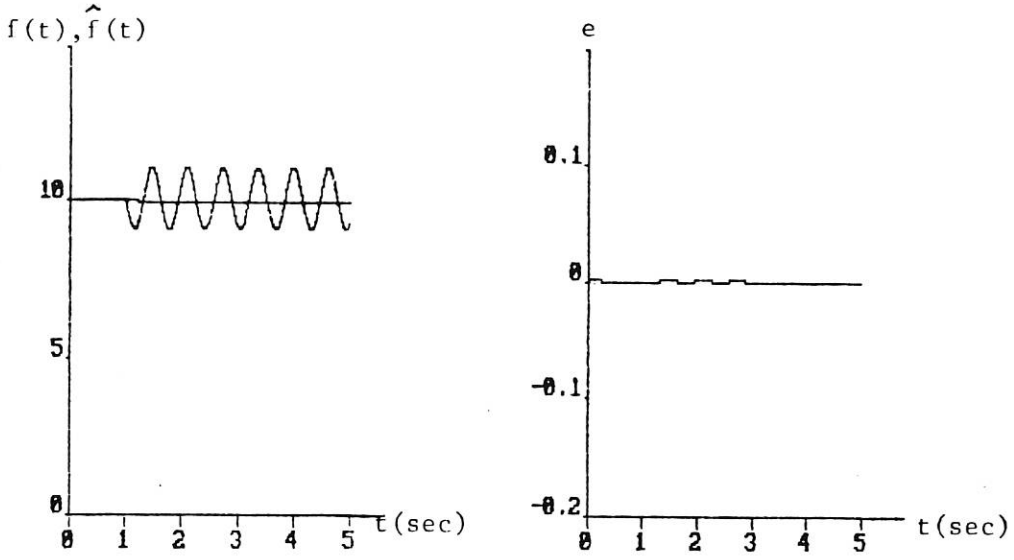

Figure 13-A.

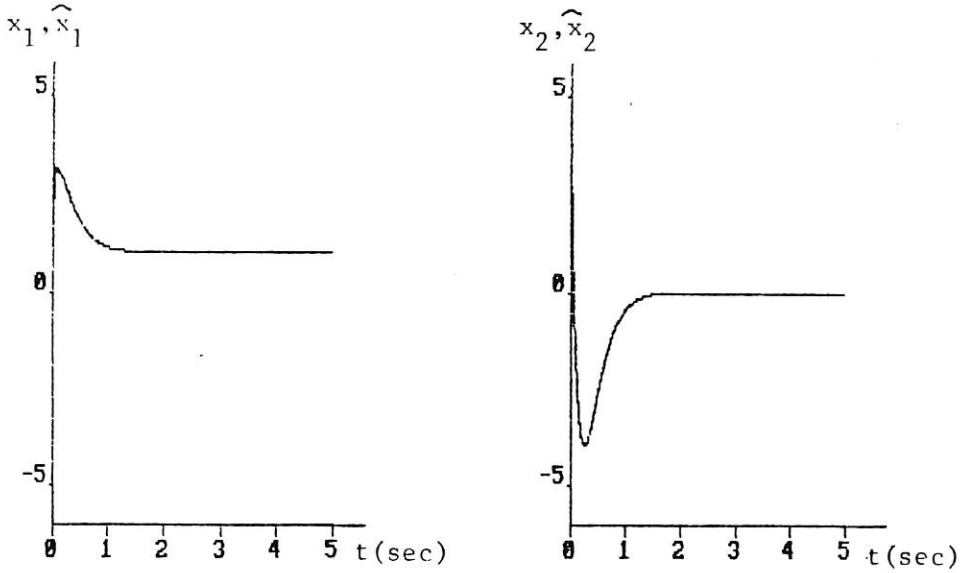

Figure 13-B.

Figure 13.
13-A Time-varying parameter estimation and state tracking error.
13-B State variables and their estimations.

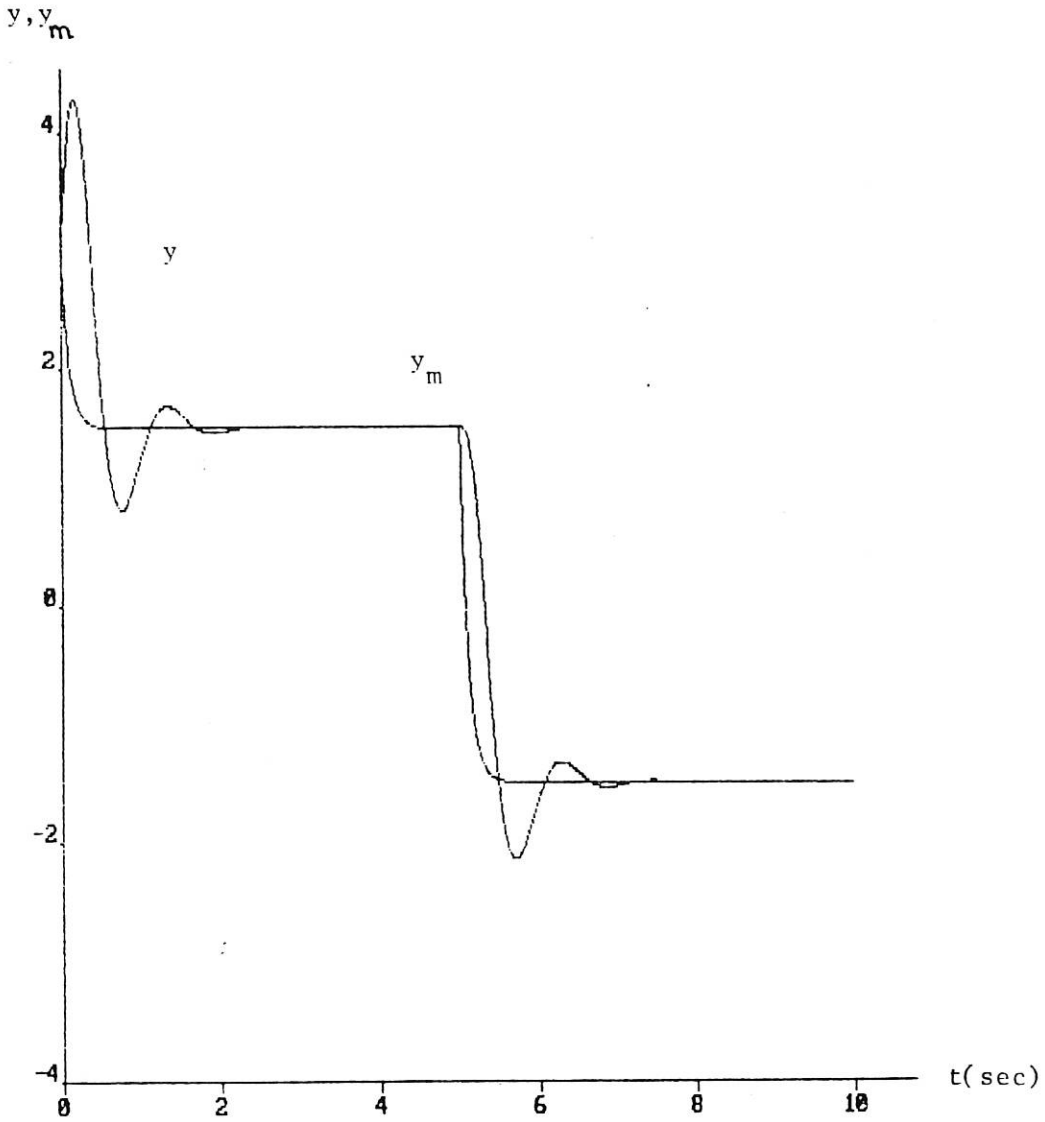

Figure 14: Plant and model reference outputs.
Variable structure control algorithm.

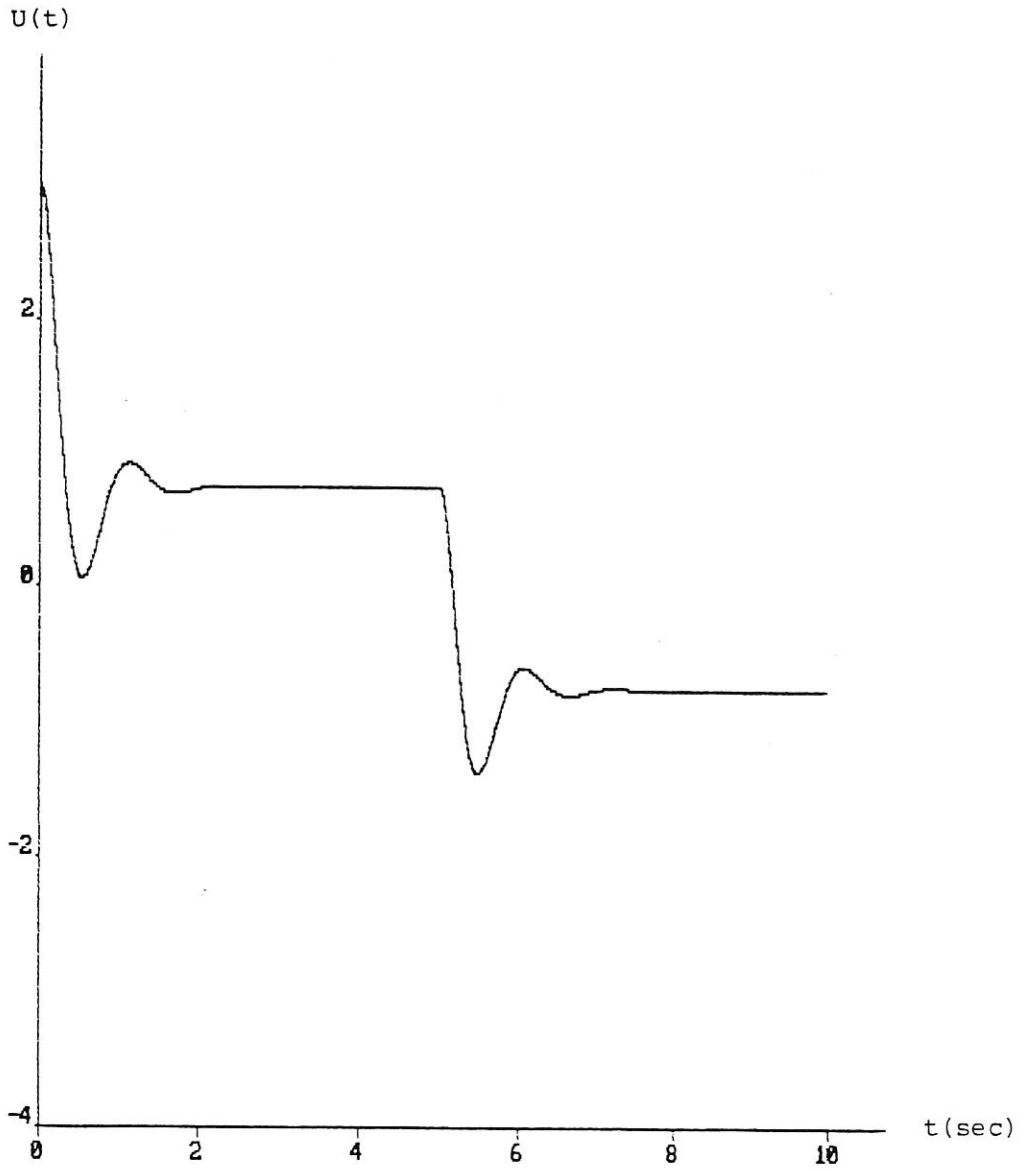

Figure 15: Neural network adaptive control action.

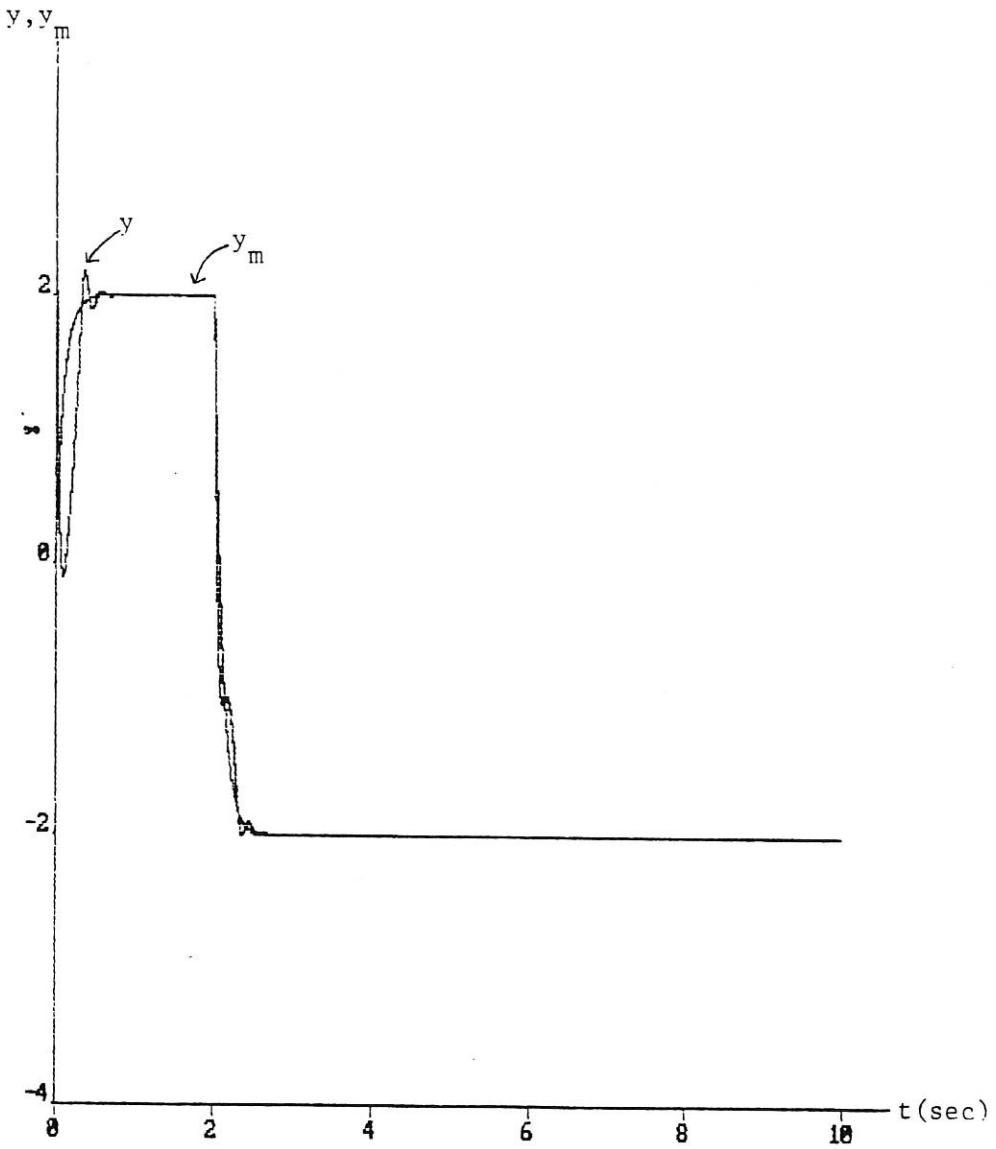

Figure 16: Plant and model reference outputs.
Gradient algorithm.

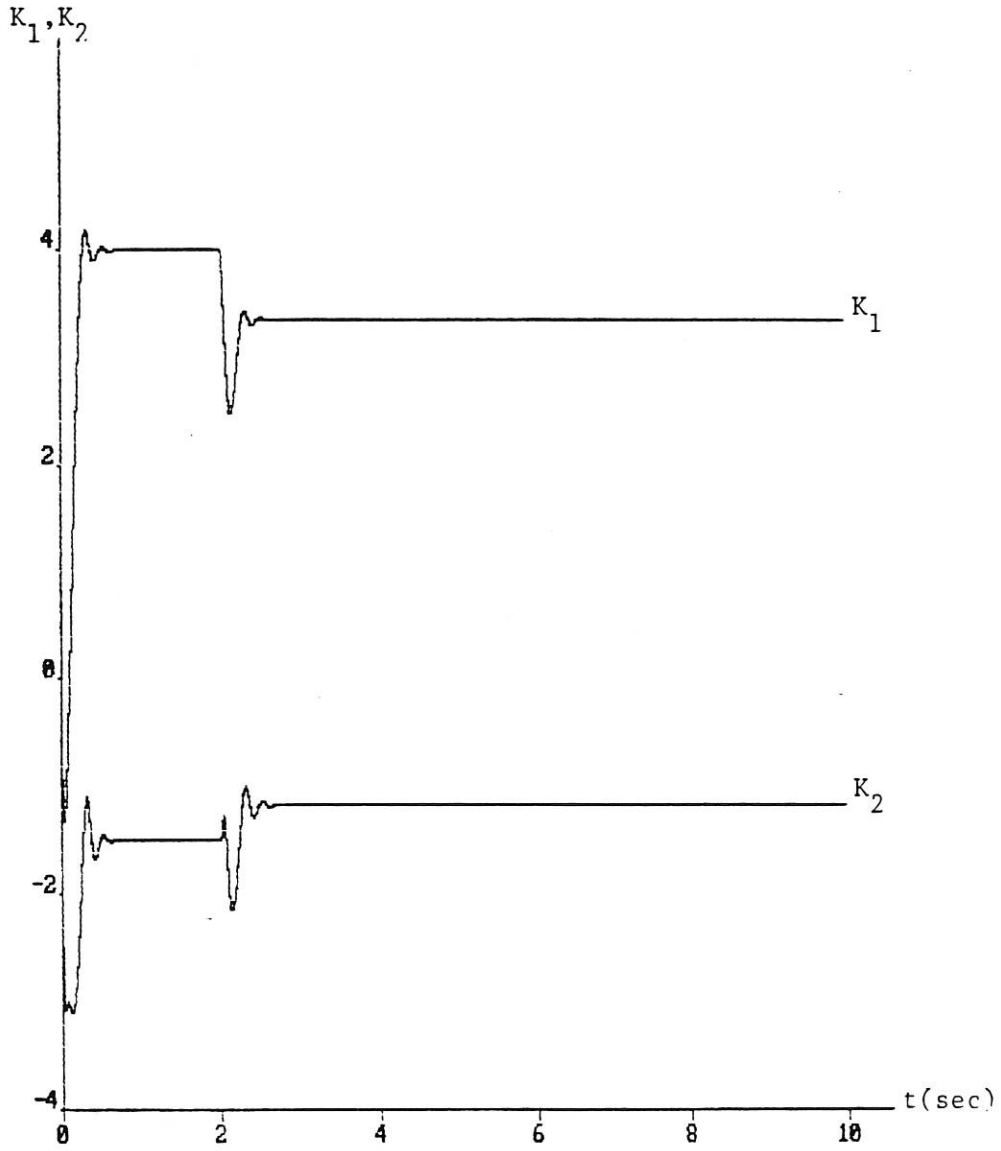

Figure 17: Adjustable controller parameters.

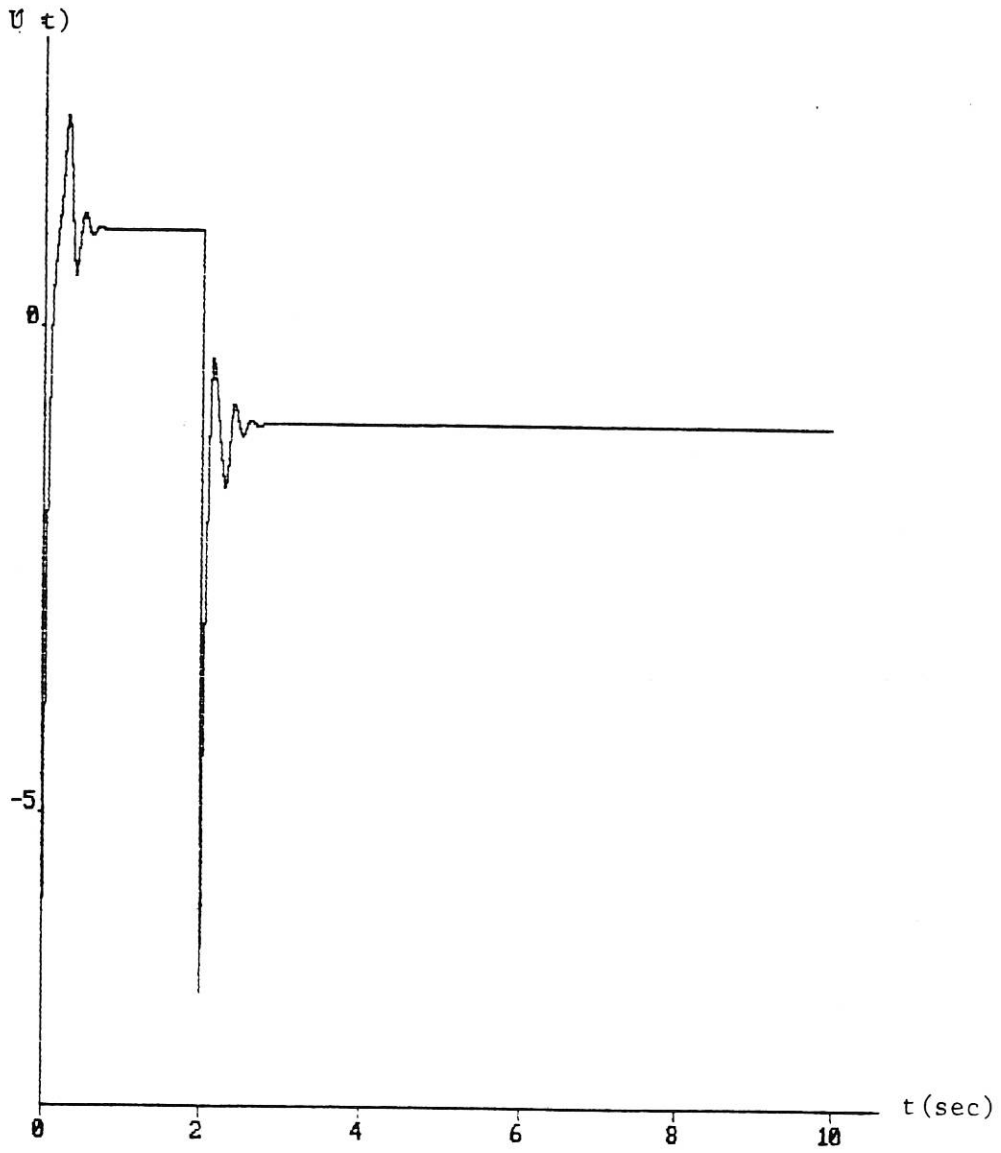

Figure 18: Model reference adaptive control law.

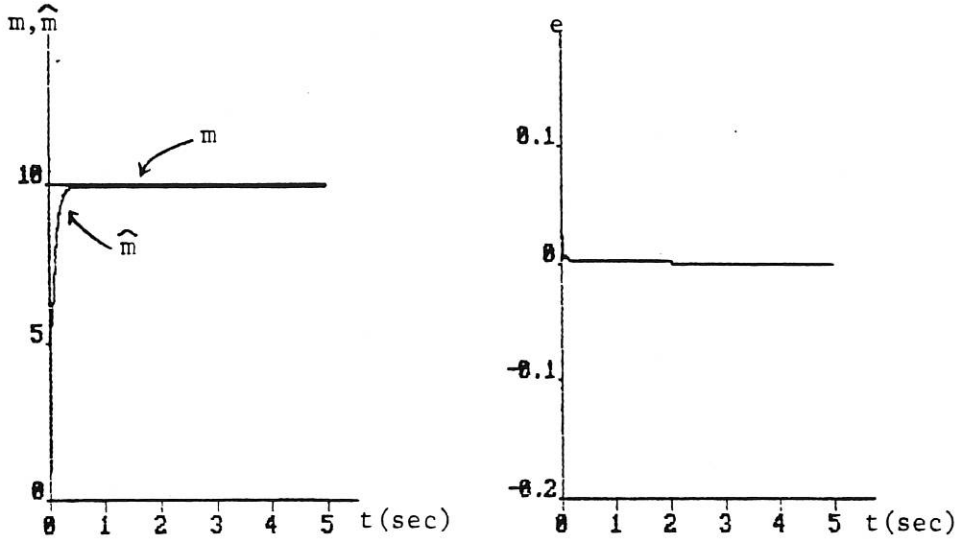

Figure 19-A.

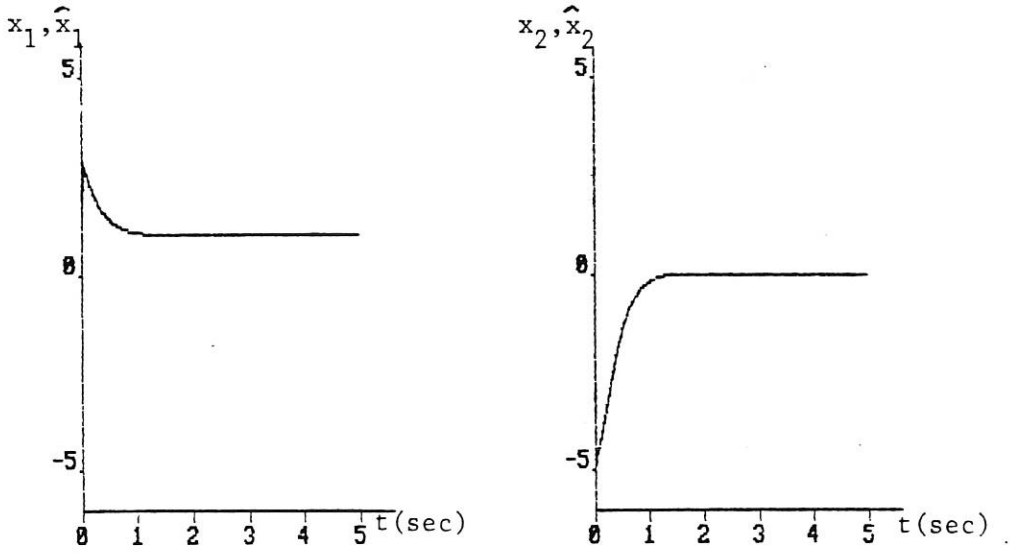

Figure 19-B.

Figure 19.
19-A Parameter estimate and state tracking error.
19-B State variables and their estimations.

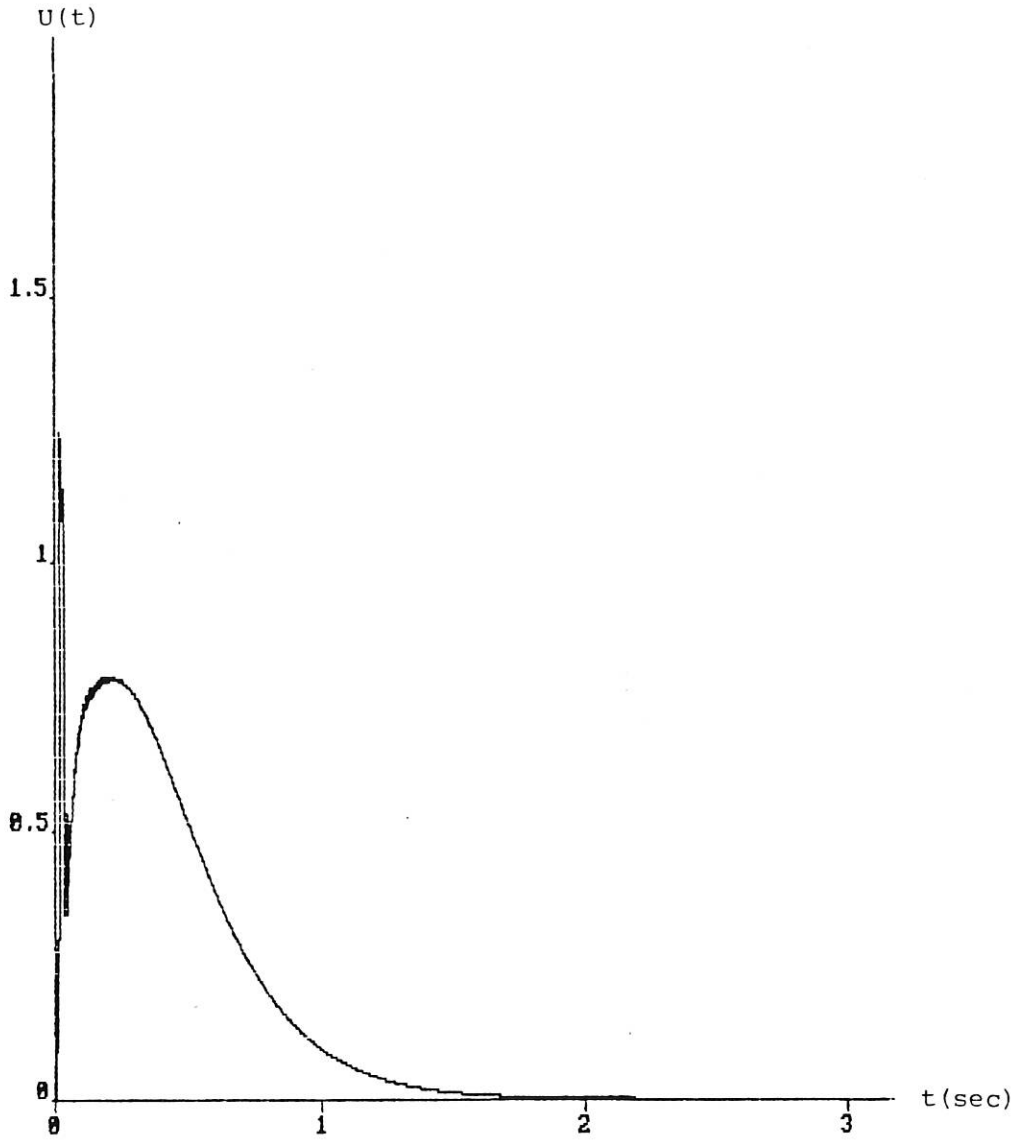

Figure 20: Self-tuning control action.

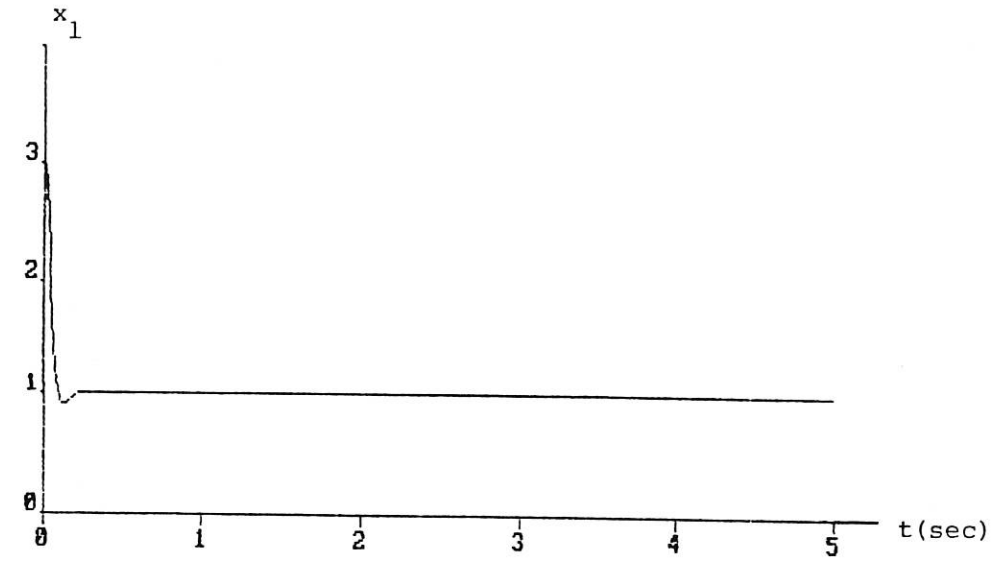

Figure 21-A.

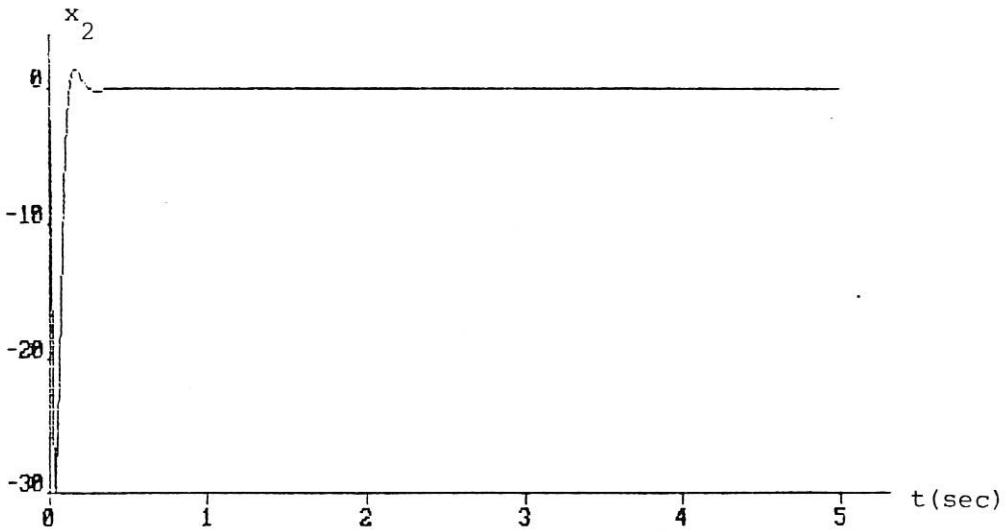

Figure 21-B.

Figure 21.
21-A Angular position of the arm.
21-B Angular velocity of the arm.

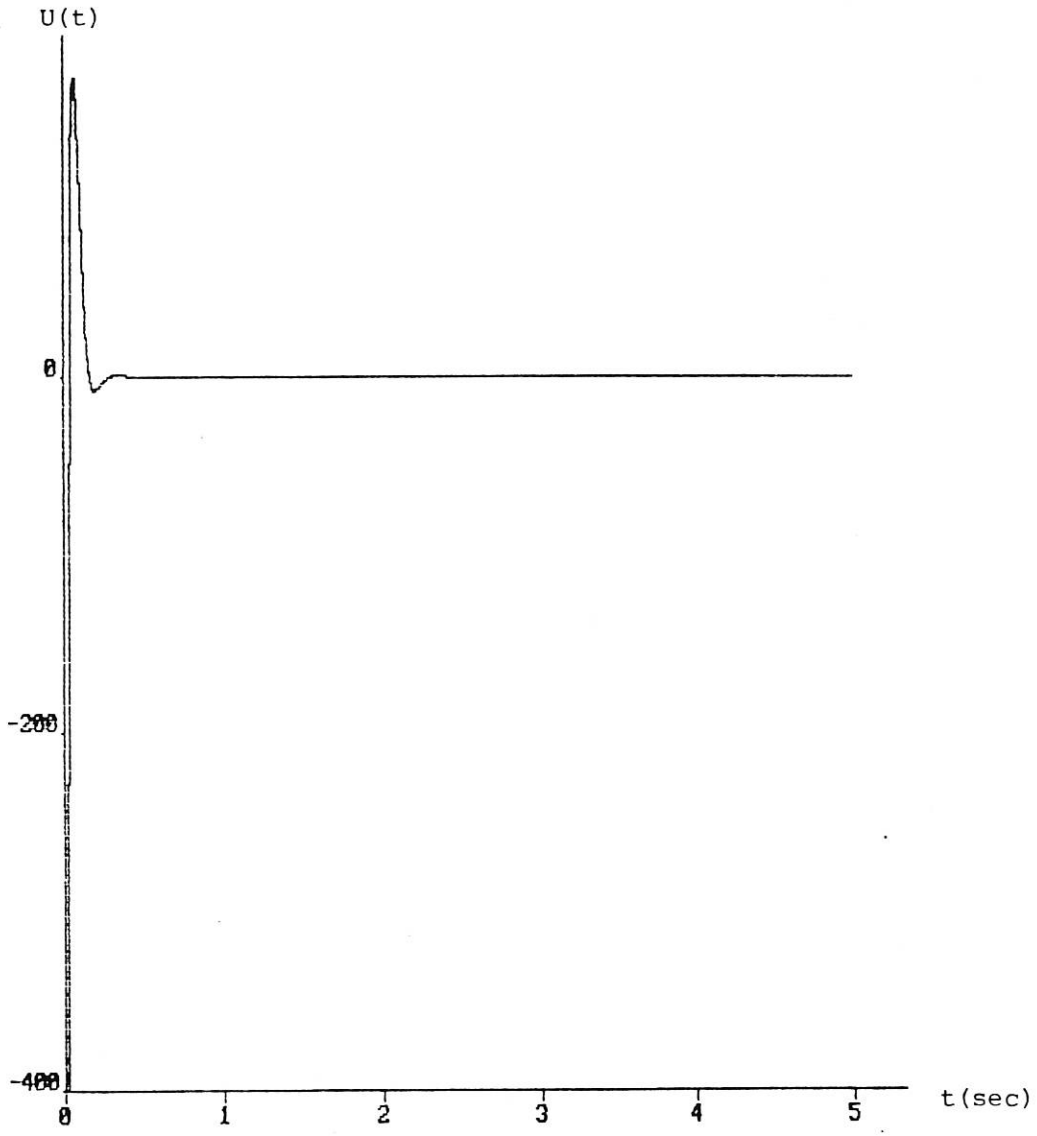

Figure 22: Self-tuning control action.