

AN IMPLEMENTATION OF AN ADAPTIVE STATE-VARIABLE
FEEDBACK MOTOR CONTROL SYSTEM

by

Mark Rogenski

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science in Engineering

in the

Electrical Engineering

Program

YOUNGSTOWN STATE UNIVERSITY

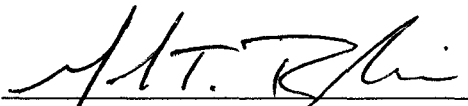
August, 2002

**An Implementation of an Adaptive State
Motor Control System**


Mark Rogenski

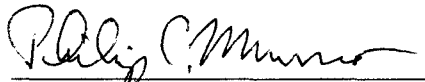
I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library for public access. I also authorize the University or other individuals to use this thesis as needed for scholarly research.

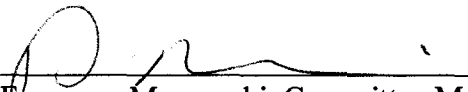
Signature:


Mark T. Rogenski, Student

Approvals:


Robert H. Foulkes, Jr., Thesis Advisor


Philip C. Munro, Committee Member


Faramarz Mossayebi, Committee Member



ACKNOWLEDGEMENTS

I express my sincerest appreciation to the following people who gave so generously of their time and energy so that I could complete this paper: Mary Lou Rogenski, Jerome Rudzik, Dr. Monica Becker, and Dr. Robert Gardner. I thank Dr. Philip C. Munro and Dr. Faramarz Mossayebi for serving on the thesis committee with very short notice. I especially thank my advisor, Dr. Robert H. Foulkes, Jr., for his extensive assistance in this endeavor. Most of all, I thank my beautiful wife, Becky. Her sacrifices, support, and patience have been beyond measure.

*To my wife, Becky
and my children, Libby, Ellie, and Marik*

TABLE OF CONTENTS

	PAGE
ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	iv
DEDICATION.....	v
TABLE OF CONTENTS.....	vi
LIST OF SYMBOLS.....	viii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xiii
CHAPTER	
I. INTRODUCTION.....	1
II. MOTOR SYSTEM.....	3
2.1 Motor System Components.....	3
2.2 Motor System Model.....	4
2.3 Motor System Discrete State-Variable Model.....	8
2.4 Open-Loop Response.....	10
III. STATE-VARIABLE FEEDBACK CONTROLLER DESIGN.....	12
3.1 Overview of Digital State-Variable Feedback Controllers.....	12
3.2 Including a State Estimator (Observer).....	14
3.3 Adaptive Control.....	15
IV. CONTROL SYSTEM IMPLEMENTATION.....	18
4.1 Hardware.....	18
4.2 Software Overview.....	18
4.2.1 NI-DAQ and Measurement & Automation Explorer (MAX).....	18
4.2.2 LabVIEW 6i.....	19
4.2.3 MATLAB/Simulink.....	19
4.3 Simulink Model.....	19
4.4 LabVIEW / MATLAB Implementation.....	23
4.5 Analog Test System.....	26
V. RESULTS.....	29
5.1 Simulink Model.....	29

5.2 Controlled GP-6 System	39
5.3 Controlled Motor System	51
VI. SUMMARY AND CONCLUSION	61
APPENDIX: PROGRAM LISTINGS	63
A.1 Simulink Initialization m-file: rls2init.m	63
A.2 Simulink S-Function (Modified Template Sections)	64
B.1 LabVIEW: MATLAB Initialization Script	67
B.2 LabVIEW: MATLAB Loop Script	68
REFERENCES	70

LIST OF SYMBOLS

A	system parameter: a_{11} element in discrete motor model \mathbf{A}_d matrix
a	$1/t_m$
\mathbf{A}_d	discrete motor model state matrix
$\hat{\mathbf{A}}_d$	estimated discrete motor model state matrix
B	system parameter: a_{21} element in discrete motor model \mathbf{A}_d matrix
b	K_m/t_m
\mathbf{B}_d	discrete motor model input vector
$\hat{\mathbf{B}}_d$	estimated discrete motor model input vector
$\boldsymbol{\beta}$	system parameter estimate vector
$\boldsymbol{\beta}_N$	nominal (or 'true') system parameter vector
$\boldsymbol{\beta}_0$	initial parameter estimate vector
$\boldsymbol{\beta}_1$	A and B system parameter estimate vector
$\boldsymbol{\beta}_2$	C_1 and C_2 system parameter estimate vector
$\boldsymbol{\beta}_{\text{FINAL}}$	final parameter estimate vector
\mathbf{C}_d	discrete motor model output matrix
$\hat{\mathbf{C}}_d$	estimated motor model output matrix
C_1	system parameter
C_2	system parameter
$G_1(z)$	discrete transfer function from $U(z)$ to $Y_1(z)$
$G_2(z)$	discrete transfer function from $U(z)$ to $Y_2(z)$
\mathbf{K}	state-feedback gain vector
K_m	tachogenerator gain
K_p	proportionality constant
\mathbf{K}_w	RLS algorithm weighting matrix
λ	RLS algorithm forgetting factor
\mathbf{L}	observer gain matrix
\mathbf{N}_x	state command matrix
\mathbf{N}_u	proportionality constant

\mathbf{P}	RLS algorithm proportionality matrix
$\phi(z)$	characteristic polynomial
$r(k)$	discrete reference signal
$\boldsymbol{\psi}(k)$	RLS algorithm known function vector
T	sample period
τ_m	motor time constant
$u(k)$	discrete input signal
$U(s)$	frequency domain input signal
V_{pot}	output potentiometer voltage
V_{tach}	tachogenerator voltage
x_1	motor system state variable
x_2	motor system state variable
$\mathbf{x}(k)$	motor system discrete state vector
\hat{x}_1	observer state variable estimate
\hat{x}_2	observer state variable estimate
$\mathbf{y}(k)$	motor system discrete output vector
$\mathbf{y} - \hat{\mathbf{y}}$	observer error signal input
$Y_1(s)$	frequency domain tachogenerator voltage
$Y_2(s)$	frequency domain potentiometer voltage
$Y_1(z)$	discrete-time tachogenerator voltage
$Y_2(z)$	discrete-time potentiometer voltage
z_1	desired closed-loop pole
z_2	desired closed-loop pole

LIST OF FIGURES

FIGURE	PAGE
2.1 Connectivity of the motor system.....	4
2.2 (a) Motor system s-domain block diagram.....	5
(b) Zero-order-hold equivalent.....	5
2.3 (a) y_1/u transfer function block diagram.....	9
(b) y_2/y_1 transfer function block diagram.....	9
2.4 Motor system open-loop response to a $\pm 1V$ square wave.....	10
2.5 Motor system open-loop response to a $\pm 0.1V$ square wave.....	11
3.1 Discrete closed loop system with state variable feedback.....	13
4.1 Simulink S-Function implementation of RLS algorithm.....	20
4.2 Simulink implementation of the full-order observer.....	21
4.3 Simulink state-variable feedback control implementation.....	22
4.4 LabVIEW VI block diagram.....	24
4.5 LabVIEW VI front panel.....	25
4.6 (a) Motor system s-domain block diagram.....	26
(b) Block diagram with values.....	26
4.7 GP-6 Analog Computer test circuit.....	27
4.8 GP-6 test system open-loop response to a 1V step input.....	28
5.1 Simulink closed-loop response to a 5V square wave. $\beta_0 = \beta_N$	30
5.2 Simulink calculated system parameters. $\beta_0 = \beta_N$	30
5.3 Simulink closed-loop states and observer states. $\beta_0 = \beta_N$	31
5.4 Simulink input $u(k)$ and tachogenerator output V_{tach} . $\beta_0 = \beta_N$	31
5.5 Simulink closed-loop response to a 5V square wave. $\beta_0 = 1/2\beta_N$	32
5.6 Simulink calculated system parameters. $\beta_0 = 1/2\beta_N$	32
5.7 Simulink calculated system parameters (first ten samples). $\beta_0 = 1/2\beta_N$	33
5.8 Simulink closed-loop states and observer states. $\beta_0 = 1/2\beta_N$	33
5.9 Simulink closed-loop and observer states (first 10 samples). $\beta_0 = 1/2\beta_N$	34
5.10 Simulink input $u(k)$ and tachogenerator output V_{tach} . $\beta_0 = 1/2\beta_N$	34
5.11 Simulink closed-loop response to a 5V square wave. $\beta_0 = 2\beta_N$	36

5.12	Simulink calculated system parameters. $\beta_0=2\beta_N$	36
5.13	Simulink closed-loop response to a 5V square wave. $\beta_0=4\beta_N$	37
5.14	Simulink calculated system parameters. $\beta_0=4\beta_N$	37
5.15	Simulink closed-loop response to a 5V square wave. $\beta_0=-\beta_N$	38
5.16	Simulink calculated system parameters. $\beta_0=-\beta_N$	38
5.17	GP-6 closed-loop response to a 5V square wave. $\beta_0=\beta_N$	39
5.18	GP-6 calculated system parameters. $\beta_0=\beta_N$	40
5.19	GP-6 closed-loop output and estimated output. $\beta_0=\beta_N$	40
5.20	GP-6 input $u(k)$ and tachogenerator output V_{tach} . $\beta_0=\beta_N$	41
5.21	GP-6 system parameters after 10 minutes. $\beta_0=\beta_N$	41
5.22	GP-6 closed-loop response to a 5V square wave. $\beta_0=1/2\beta_N$	43
5.23	GP-6 calculated system parameters. $\beta_0=1/2\beta_N$	43
5.24	GP-6 closed-loop response to a 5V square wave. $\beta_0=2\beta_N$	44
5.25	GP-6 calculated system parameters. $\beta_0=2\beta_N$	44
5.26	GP-6 closed-loop response to a 5V square wave. $\beta_0=4\beta_N$	45
5.27	GP-6 calculated system parameters. $\beta_0=4\beta_N$	45
5.28	GP-6 closed-loop response to a 5V square wave. $\beta_0=-\beta_N$	46
5.29	GP-6 calculated system parameters. $\beta_0=-\beta_N$	46
5.30	GP-6 closed-loop response: Pot 1 varied from 0.262 to 0.165.....	48
5.31	GP-6 calculated system parameters: Pot 1 varied from 0.262 to 0.165.....	48
5.32	GP-6 closed-loop response: Pot 2 varied from 0.400 to 0.263.....	49
5.33	GP-6 calculated system parameters: Pot 2 varied from 0.400 to 0.263.....	49
5.34	GP-6 closed-loop response: Pot 3 varied from 0.600 to 0.485.....	50
5.35	GP-6 calculated system parameters: Pot 3 varied from 0.600 to 0.485.....	50
5.36	Motor closed-loop response to a 5V square wave. $\beta_0=\beta_N$	51
5.37	Motor calculated system parameters. $\beta_0=\beta_N$	52
5.38	Motor closed-loop output and estimated output. $\beta_0=\beta_N$	52
5.39	Motor input $u(k)$ and tachogenerator output V_{tach} . $\beta_0=\beta_N$	53
5.40	Motor calculated system parameters after 10 minutes. $\beta_0=\beta_N$	53
5.41	Motor closed-loop response to a 5V square wave. $\beta_0=1/2\beta_N$	55
5.42	Motor calculated system parameters. $\beta_0=1/2\beta_N$	55

5.43	Motor closed-loop response to a 5V square wave. $\beta_0=2\beta_N$	56
5.44	Motor calculated system parameters. $\beta_0=2\beta_N$	56
5.45	Motor closed-loop response to a 5V square wave. $\beta_0=4\beta_N$	57
5.46	Motor calculated system parameters. $\beta_0=4\beta_N$	57
5.47	Motor closed-loop response (last 600 samples of 6000). $\beta_0=4\beta_N$	58
5.48	Motor calculated system parameters after 10 minutes. $\beta_0=4\beta_N$	58
5.49	Motor closed-loop response to a 5V square wave. $\beta_0=-\beta_N$	59
5.50	Motor calculated system parameters. $\beta_0=-\beta_N$	59

LIST OF TABLES

TABLE	PAGE
4.1 MAX Channel Configuration.....	19
4.2 Simulink model pole locations.....	22
5.1 Parameter Estimate Data.....	60
5.2 Average Final Parameter Estimates.....	60

CHAPTER I

INTRODUCTION

Adaptive control has emerged as a fundamental way of controlling complicated modern systems. This type of control system modifies itself to fit the requirements of the plant it is controlling or the environment in which the plant is operating. The adaptive control system continuously senses the system's behavior and adjusts itself to maintain the system performance at an optimal level. Adaptive systems have been developed for many complicated control applications. Some of these include aircraft flight control, adaptive ship steering, raw material processing control, and robot locomotion control [1].

There are two major techniques for implementing adaptive control [1]:

- i) signal synthesis adaptation
- ii) parameter adaptation

The signal synthesis approach involves generating an adaptive feedback signal that modifies the plant input control signal. This input signal is tailored such that the plant outputs are forced to optimize system performance. The second method uses mathematical algorithms to calculate system parameters while the system is running. These parameters are then dynamically incorporated into the control calculations to optimize the control structure. This second approach of parameter adaptation is employed in this investigation.

As adaptive control theories have progressed, sophisticated software programs have also evolved to model, test, and implement advanced control systems. The question we investigate here is, are these software packages capable of implementing complicated control algorithms like adaptive control and how effective is the implementation?

Modern software packages have emerged that simplify control system development. Dragging and dropping system blocks on a workspace have essentially replaced countless lines of C code and weeks of development time. Using graphical programming or simple batch type files, programs like MATLAB and Simulink allow engineers to quickly model, test, and optimize many complicated control systems. By fabricating I/O interface hardware, an engineer can use the graphical program to implement a complete control system.

Today, manufacturers have simplified even this last step by creating multifunction interface hardware that can be used in conjunction with graphical software to realize a complete system [7]. An example of such a package is from National Instruments. This includes LabVIEW software and an E-series data acquisition card. A control system can be completely "wired" in LabVIEW using only the mouse and typing in a few parameters [8]. The data acquisition card is then plugged into the PC and interfaced to a breadboard. Plant inputs and outputs only need to be hardwired to the breadboard and the system is operational.

An additional benefit to the software packages is the ability to plot acquired data. MATLAB and Simulink have extensive customizable plotting capabilities and LabVIEW has built-in real-time waveform charting functions [9][8].

As we consider the advancements in control methods, software, and hardware, it is natural to integrate the components into a modern adaptive control implementation. It is worth noting here that adaptive control methods are generally applied to systems that are not suited to either deterministic (well-defined plant parameters) or stochastic (partially known plant parameters) control methods. However, for comparative purposes, here the adaptive system is applied first to a deterministic analog test system and then to a more stochastic motor system. In this way, the adaptive parameter estimates can be compared to known (or generally known) parameters to determine the system's effectiveness in adapting to the plant [1].

This thesis investigates the ability of pre-packaged control hardware and software to implement an adaptive state-feedback control system and attempts to determine the effectiveness of such an implementation. The investigation begins with the plant, a laboratory servomotor system. Chapter II describes the motor and develops the necessary system equations that describe it in continuous and discrete time. Next the controller design is explained in Chapter III. State-variable feedback is introduced, an observer is defined, and the Recursive Least Squares algorithm for adaptive control is presented. In Chapter IV the implementation of the control system is described. The hardware, pre-packaged software, and custom written scripts are integrated into an operational control system. The results of the implementation are given and interpreted in Chapter V; and final conclusions are reached in Chapter VI.

CHAPTER II

MOTOR SYSTEM

The plant utilized for the adaptive control system is a Feedback Type MS 150 Modular Servo System [2]. This laboratory motor system consists of modules that can be added as required for a specific type of operation. In this chapter the motor system components are described and the necessary system equations are developed. Also the motor's open-loop response to a step input is plotted and explained.

2.1 Motor System Components

In this investigation the following modules are used [2].

Motor Unit 150F: Integral servomotor/tachogenerator with a 30:1 reduction gearbox.

Power Supply 150E: Input 120V at 60Hz, output ± 15 VDC 50mA.

Pre-amp Unit 150C: Two input channels and a push-pull output to directly drive the servo amplifier.

Servo Amplifier 150D: Operates the motor via an 8-way connection socket. Can be wired so the motor performance characteristics demonstrate field or armature control.

Output Potentiometer 150K: Servo-type potentiometer with a calibrated position dial. Carries a rear extension shaft that couples directly with the motor unit.

Figure 2.1 shows the connectivity of the motor system as tested. In this system, the tachogenerator provides a voltage signal proportional to the shaft velocity. The output potentiometer produces a voltage proportional to its angular position.

The amplifier circuit is field connected for these tests. This means the armature of the motor is connected to the collector end of the power transistor amplification circuit. This arrangement is advantageous because it provides the motor a high gain. A small increase in input signal will cause a significant increase in motor speed. One disadvantage of this configuration is that it makes the motor more difficult to control than an armature connected circuit. Although not a factor in these tests, another disadvantage is that shaft torque drops quickly when a load is applied to the motor.

The pre-amplifier module is switched to *defined* τ , which introduces a compensating circuit into the pre-amplifier. This circuit in conjunction with tachogenerator feedback provides a motor characteristic that approximates a linear

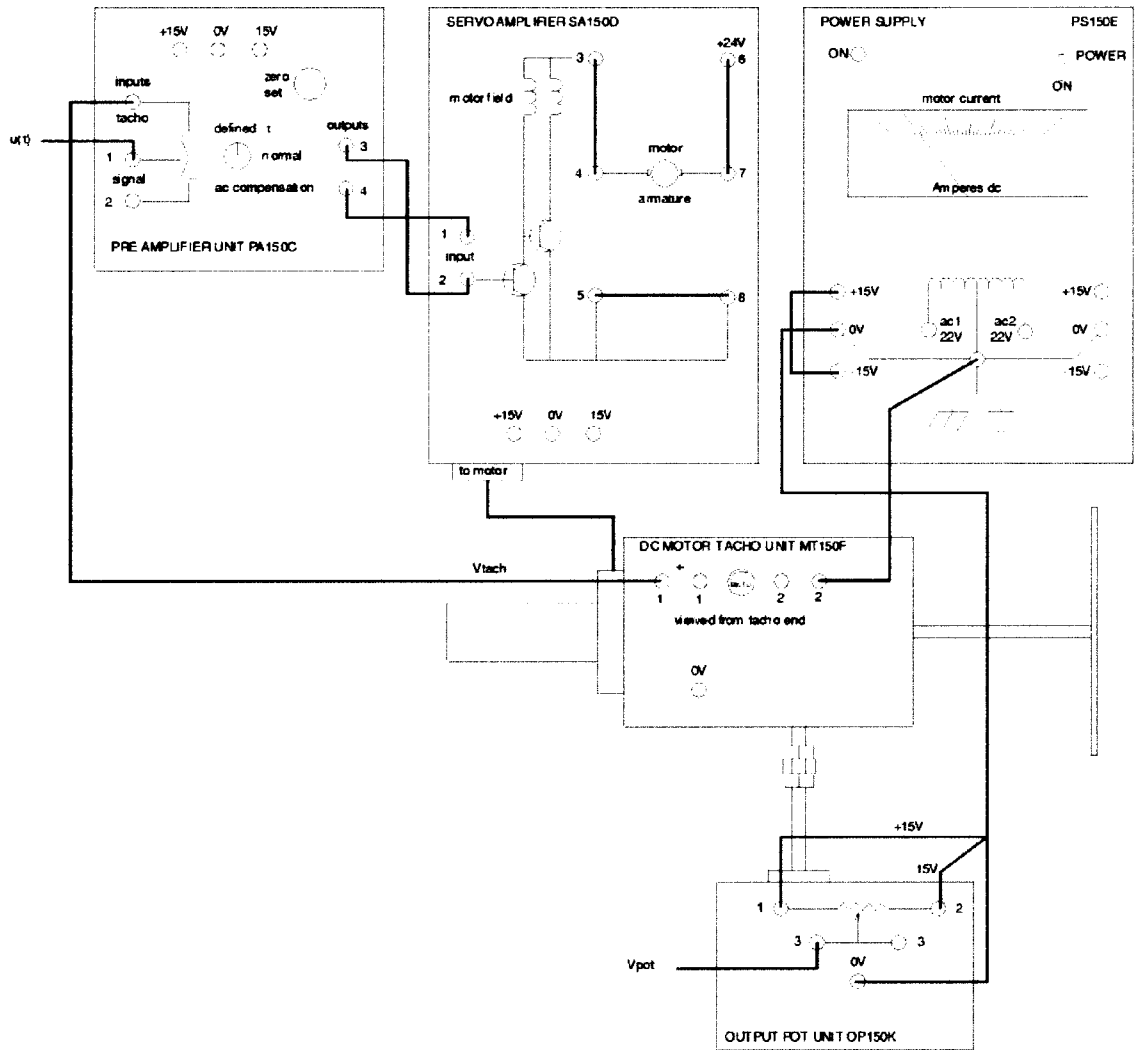


Figure 2.1 Connectivity of the motor system.

system with a single time constant between the pre-amplifier input and the motor speed or tachogenerator voltage. The motor is a Type 1 system and thus will track a constant reference without error if used in a unity-feedback control loop [3].

2.2 Motor System Model

The general transfer function between the input voltage $U(s)$ and the tachogenerator voltage $V_{tach}(s)$ is given by

$$\frac{V_{tach}(s)}{U(s)} = \frac{K_m}{1 + \tau_m s} \quad (2-1)$$

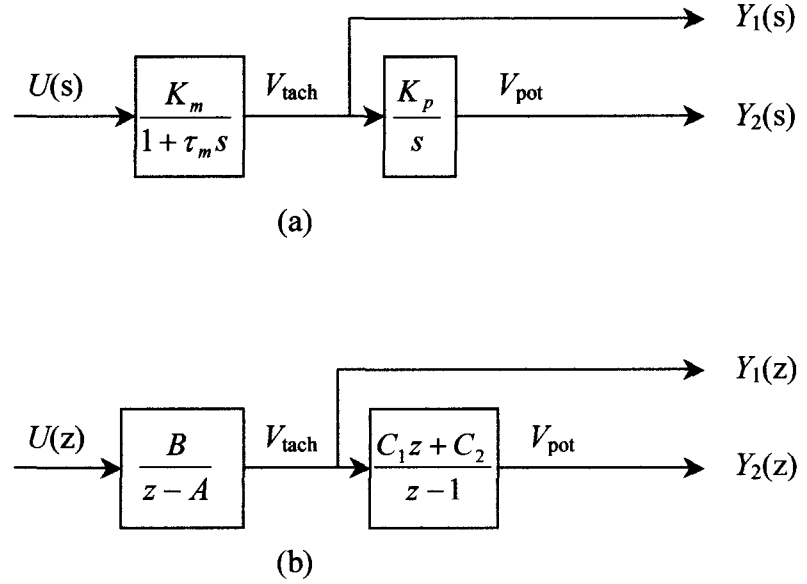


Figure 2.2 (a) Motor system s-domain block diagram. (b) Zero-order-hold equivalent.

where τ_m is the motor time constant and K_m is the tachogenerator gain. Nominal values of motor parameters are initially used in calculations for the control system. These values are $\tau_m=0.25$ and $K_m=6.5$ [2].

Recall that V_{tach} is proportional to the speed of the motor and the pot voltage, V_{pot} , is proportional to angular shaft position. Using the relationship that angular position is the integral of the speed, it follows that

$$\frac{V_{pot}(s)}{V_{tach}(s)} = \frac{K_p}{s} \quad (2-2)$$

where K_p is a constant whose value is approximately 6.

Equations (2-1) and (2-2) can be combined into one system transfer function, but it works to our advantage to keep them separate in the model. Using the block diagram in Figure 2.2(a), we see that two outputs can be defined, $Y_1=V_{tach}$ and $Y_2=V_{pot}$.

The zero-order hold equivalent to the system in Figure 2.2(a) is shown in Figure 2.2(b). From the block diagram in 2.2(a) the transfer functions in 2.2(b) are obtained as follows. In equation (2-1) let $a=1/\tau_m$ and $b=K_m/\tau_m$ and substitute $Y_1(s)$ for $V_{tach}(s)$ to yield

$$\frac{Y_1(s)}{U(s)} = \frac{b}{s + a} \quad (2-3)$$

Taking the step-response of equation (2-3) gives

$$Y_1(s) = \frac{1}{s} \left(\frac{b}{s+a} \right) = \frac{\left(\frac{b}{a} \right)}{s} + \frac{\left(-\frac{b}{a} \right)}{s+a}. \quad (2-4)$$

Converting (2-4) to the time-domain finds

$$y_1(t) = \frac{b}{a} - \frac{b}{a} e^{-at}. \quad (2-5)$$

Sampling equation (2-5) with a sample period T gives

$$y_1(kT) = \frac{b}{a} - \frac{b}{a} e^{-akT}. \quad (2-6)$$

Now take the z-transform of (2-6) to get

$$\begin{aligned} Y_1(z) &= \frac{b}{a} \left(\frac{z}{z-1} \right) - \frac{b}{a} \left(\frac{z}{z-e^{-aT}} \right) \\ &= \frac{bz}{a} \left(\frac{(z-e^{-aT}) - (z-1)}{(z-1)(z-e^{-aT})} \right) \\ &= \frac{\frac{bz}{a} (1-e^{-aT})}{(z-1)(z-e^{-aT})} \end{aligned} \quad (2-7)$$

Finally from (2-7) we can find the discrete transfer function

$$\begin{aligned} G_1(z) &= \frac{Y_1(z)}{U(z)} \\ &= \left(\frac{z-1}{z} \right) Y_1(z) \\ &= \frac{\frac{b}{a} (1-e^{-aT})}{z-e^{-aT}} \\ &= \frac{B}{z-A} \end{aligned} \quad (2-8)$$

Letting $K_p=K$, the same process can be followed to find $G_2(z)$. Starting with the step-response

$$\begin{aligned}
 Y_2(s) &= \frac{1}{s} \left(\frac{Kb}{s(s+a)} \right) \\
 &= \frac{\left(\frac{Kb}{a} \right)}{s^2} + \frac{\left(-\frac{Kb}{a^2} \right)}{s} + \frac{\left(\frac{Kb}{a^2} \right)}{s+a}
 \end{aligned} \tag{2-9}$$

Convert (2-9) to the time-domain:

$$y_2(t) = \left(\frac{Kb}{a} \right) t - \frac{Kb}{a^2} + \left(\frac{Kb}{a^2} \right) e^{-at}. \tag{2-10}$$

Sample equation (2-10) at period T :

$$y_2(kT) = \left(\frac{KbT}{a} \right) k - \frac{Kb}{a^2} + \left(\frac{Kb}{a^2} \right) e^{-akT}. \tag{2-11}$$

Take the z -transform of (2-11):

$$Y_2(z) = \frac{KbT}{a} \left(\frac{z}{(z-1)^2} \right) - \frac{Kb}{a^2} \left(\frac{z}{z-1} \right) + \frac{Kb}{a^2} \left(\frac{z}{z-e^{-aT}} \right). \tag{2-12}$$

Finally, find the discrete transfer function:

$$\begin{aligned}
 G_2(z) &= \frac{Y_2(z)}{U(z)} = \left(\frac{z-1}{z} \right) Y_2(z) = \frac{KbT}{a} \left(\frac{1}{z-1} \right) - \frac{Kb}{a^2} + \frac{Kb}{a^2} \left(\frac{z-1}{z-e^{-aT}} \right) \\
 &= \frac{Kb}{a^2} \left[\frac{aT(z-e^{-aT}) - (z-1)(z-e^{-aT}) + (z-1)^2}{(z-1)(z-e^{-aT})} \right] \\
 &= \frac{Kb}{a^2} \left[\frac{z(aT-1+e^{-aT}) + (-aTe^{-aT} - e^{-aT} + 1)}{(z-1)(z-e^{-aT})} \right] \\
 &= \frac{\frac{b}{a}(1-e^{-aT})}{z-e^{-aT}} \cdot \left[\frac{K}{a} \cdot \frac{aT+e^{-aT}-1}{1-e^{-aT}} \right] z + \left[\frac{K}{a} \cdot \frac{1-e^{-aT}-aTe^{-aT}}{1-e^{-aT}} \right] \\
 &= \frac{B}{z-A} \cdot \frac{C_1 z + C_2}{z-1}
 \end{aligned} \tag{2-13}$$

Summarizing, from (2-8) and (2-13) the discrete transfer functions are

$$\begin{aligned} G_1(z) &= \frac{Y_1(z)}{U(z)} = \frac{B}{z-A} \\ G_2(z) &= \frac{Y_2(z)}{U(z)} = \frac{C_1 B z + C_2 B}{(z-A)(z-1)} \end{aligned} \quad (2-14)$$

where the system parameters A , B , C_1 , and C_2 are

$$\begin{aligned} A &= e^{-aT} \\ B &= \frac{b}{a}(1 - e^{-aT}) \\ C_1 &= \frac{K}{a} \cdot \frac{(aT + e^{-aT} - 1)}{1 - e^{-aT}} \\ C_2 &= \frac{K}{a} \cdot \frac{(1 - e^{-aT} - aTe^{-aT})}{1 - e^{-aT}} \end{aligned} \quad (2-15)$$

Replacing a , b , and K in (2-15) leads to system parameters in terms of motor parameters:

$$\begin{aligned} A &= e^{-\frac{T}{\tau_m}} \\ B &= K_m \left(1 - e^{-\frac{T}{\tau_m}} \right) \\ C_1 &= \frac{K_p (T - \tau_m + \tau_m e^{-\frac{T}{\tau_m}})}{1 - e^{-\frac{T}{\tau_m}}} \\ C_2 &= \frac{K_p (\tau_m - \tau_m e^{-\frac{T}{\tau_m}} - Te^{-\frac{T}{\tau_m}})}{1 - e^{-\frac{T}{\tau_m}}} \end{aligned} \quad (2-16)$$

2.3 Motor System Discrete State-Variable Model

The state variables for the motor system are determined using the transfer functions in equation (2-14). By considering each equation in the block-diagram form shown in Figures 2.3(a) and (b), we establish the state variables, x_1 and x_2 .

From these diagrams we can write the system equations

$$\begin{aligned}
 x_1(k+1) &= Ax_1(k) + u(k) \\
 x_2(k+1) &= x_2(k) + y_1(k) \\
 y_1(k) &= Bx_1(k) \\
 y_2(k) &= (C_1 + C_2)x_2(k) + C_1y_1(k)
 \end{aligned}
 \tag{2-17}$$

In matrix form the system becomes:

$$\begin{aligned}
 \mathbf{x}(k+1) &= \mathbf{A}_d \mathbf{x}(k) + \mathbf{B}_d u(k) \\
 \mathbf{y}(k) &= \mathbf{C}_d \mathbf{x}(k)
 \end{aligned}
 \tag{2-18}$$

where

$$\mathbf{A}_d = \begin{bmatrix} A & 0 \\ B & 0 \end{bmatrix} \quad \mathbf{B}_d = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{C}_d = \begin{bmatrix} B & 0 \\ C_1 B & C_1 + C_2 \end{bmatrix}.
 \tag{2-19}$$

Equation (2-18) is the discrete state-variable model of the motor system.

A consequence of this selection of system states is the inaccessibility of the states x_1 and x_2 . In order to feed these states back to the input, a construct called an *observer* is needed. The observer is discussed in Chapter III.

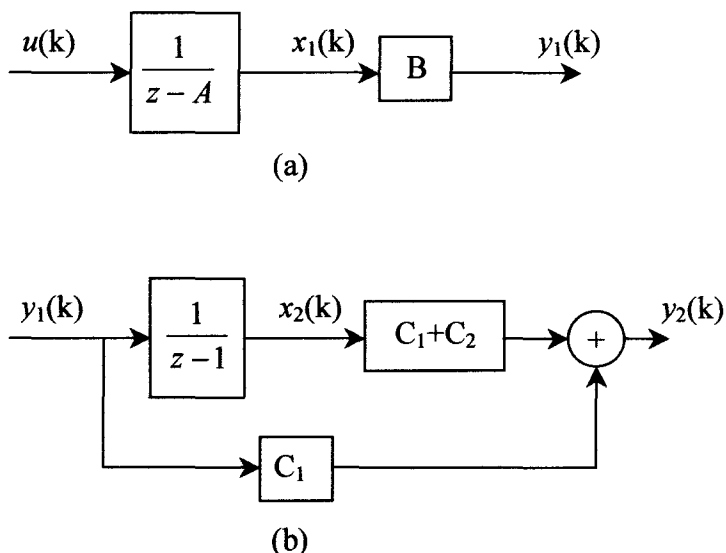


Figure 2.3 (a) y_1/u transfer function block diagram. (b) y_2/y_1 transfer function block diagram.

2.4 Open-Loop Response

The open-loop response of the motor to a $\pm 1\text{V}$ square wave is shown in Figure 2.4. As the plot shows, the potentiometer voltage (V_{pot}) is a sawtooth function between $+15\text{V}$ and -15V . When the potentiometer dial spins counter-clockwise (as in the 0 to 3s time range in Figure 2.4), its output voltage drops linearly from $+15\text{V}$ to -15V . Spinning clockwise, the output voltage rises linearly between $+15\text{V}$ and -15V . Once per revolution, the potentiometer contacts pass over a discontinuous transition region between $+15\text{V}$ and -15V . During this same time, the tachogenerator voltage V_{tach} is a continuous signal switching as the motor shaft moves in opposite directions.

When a small square wave is applied to the motor input, a drift effect of friction is seen in the V_{pot} signal. Instead of making complete revolutions, the potentiometer dial rotates back and forth. The V_{pot} signal oscillates between a minimum and maximum voltage. Due to dissimilar friction effects as the motor turns in opposite directions, the values of the minimum and maximum voltage change by some drift factor. This is illustrated in the response to a 0.1V square wave shown in Figure 2.5. Since the motor essentially spins more easily in one direction than the other, the median value of V_{pot} drifts in the direction of less friction.

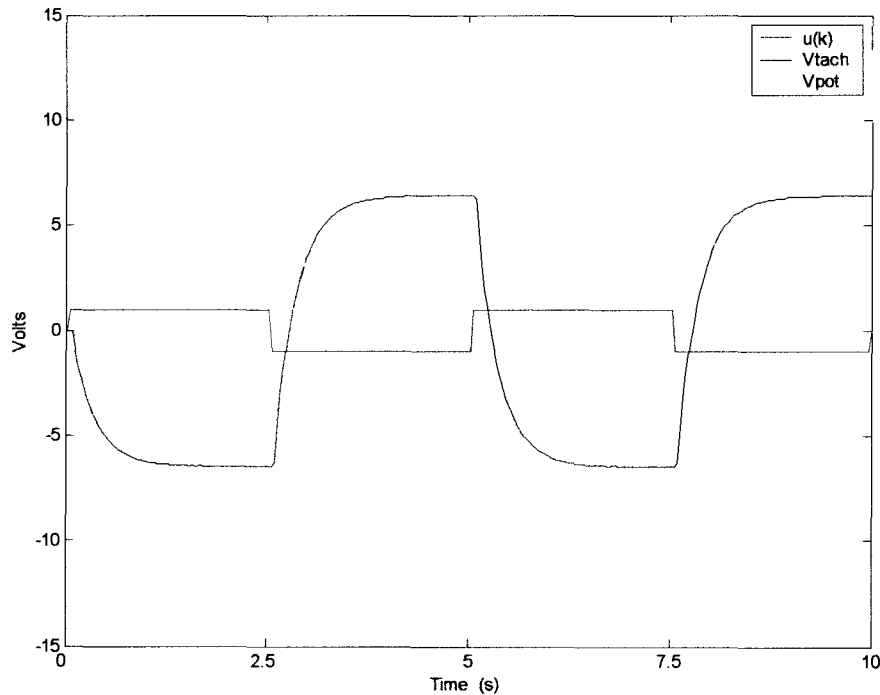


Figure 2.4 Motor system open-loop response to a $\pm 1\text{V}$ square wave.

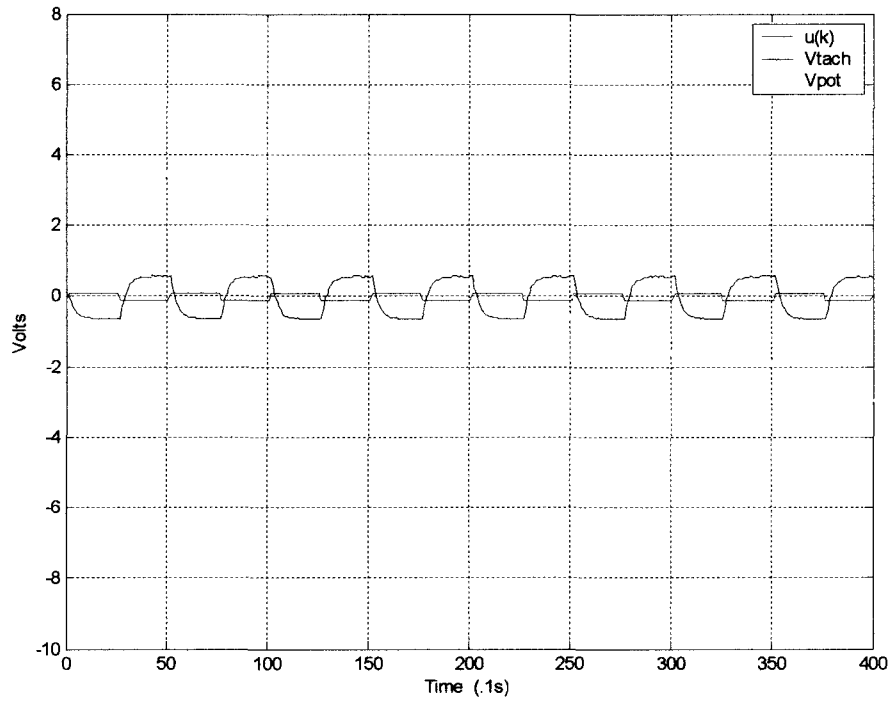


Figure 2.5 Motor system open-loop response to a ± 0.1 V square wave.

CHAPTER III

STATE-VARIABLE FEEDBACK CONTROLLER DESIGN

Now that the motor system model is developed, the controller can be designed to obtain the desired system behavior. The control objective is position control with zero steady-state error and good transient performance. The closed-loop V_{pot} signal must track a constant reference input with a fast rise time and minimal overshoot. A constraint on the reference input is that it must not force the potentiometer voltage beyond the discontinuous region at $\pm 15\text{V}$ since the motor system model does not account for this region.

The fundamental control method chosen for the servomotor system is based on state-variable feedback. A state estimator, or observer, is included in the controller to provide access to the system states. Finally, using an adaptive algorithm to estimate values of system parameters completes the controller structure.

This chapter provides an overview of state-variable feedback control for discrete systems. The observer is then introduced, and the adaptive control algorithm is discussed.

3.1 Overview of Digital State-Variable Feedback Controllers

For simplicity, the dynamic equations of physical systems are often written as state equations. State equations reduce n^{th} -order differential equations into a set of n first-order equations. The variables used to write these equations are the *state variables*.

In state-variable feedback, the system state variables are fed back through a gain \mathbf{K} to the input. By correctly defining \mathbf{K} , the system's \mathbf{A}_d matrix is modified in such a way as to improve output performance. A discrete state-feedback system is illustrated with an example [4]:

Consider the discrete state-space system:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{A}_d \mathbf{x}(k) + \mathbf{B}_d u(k) \\ \mathbf{y}(k) &= \mathbf{C}_d \mathbf{x}(k)\end{aligned}\tag{3-1}$$

This system can be controlled by output feedback, i.e. connecting the output $\mathbf{y}(k)$ back to input $u(k)$. However, assuming there is access to the state variable vector $\mathbf{x}(k)$, the system can also be controlled by feeding $\mathbf{x}(k)$ back to $u(k)$ through a gain vector \mathbf{K} . Figure 3.1 shows the block diagram of this closed-loop system.

Using $u(k) = \mathbf{K} \mathbf{x}(k)$, the state equations can be written as

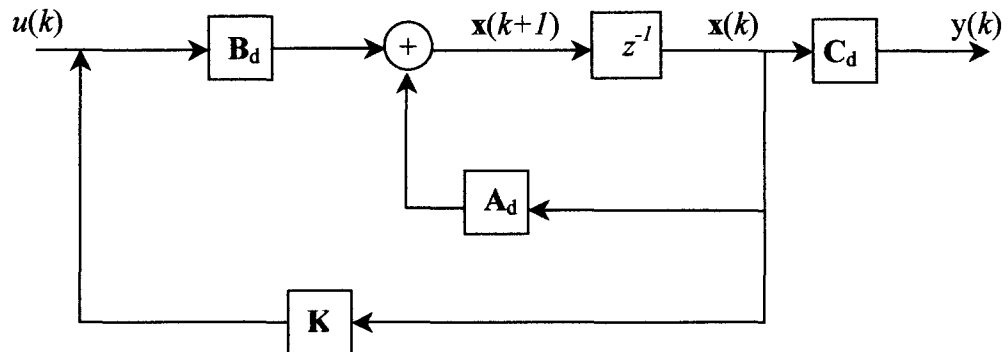


Figure 3.1 Discrete closed loop system with state variable feedback.

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{A}_d \mathbf{x}(k) + \mathbf{B}_d \mathbf{K} \mathbf{x}(k) = (\mathbf{A}_d + \mathbf{B}_d \mathbf{K}) \mathbf{x}(k) \\ \mathbf{y}(k) &= \mathbf{C}_d \mathbf{x}(k)\end{aligned}\quad (3-2)$$

We need to force the system to respond in the desired manner by moving the closed-loop system poles with the feedback gain matrix \mathbf{K} . For a second-order system we need to find $\mathbf{K}=[k_0 \ k_1]$ to specify the terms in the desired characteristic polynomial

$$\varphi(z) = z^2 + (a_1 - k_1)z + (a_0 - k_0). \quad (3-3)$$

This process is illustrated in the following second-order case using a general \mathbf{A}_d matrix and our specific \mathbf{B}_d matrix (for simplification of notation). The system is defined as

$$\mathbf{x}(k+1) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k). \quad (3-4)$$

Substituting \mathbf{K} into (3-2),

$$\begin{aligned}\mathbf{A}_d + \mathbf{B}_d \mathbf{K} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} [k_0 \ k_1] \\ &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} k_0 & k_1 \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} a_{11} + k_0 & a_{12} + k_1 \\ a_{21} & a_{22} \end{bmatrix}\end{aligned}\quad (3-5)$$

Now computing the closed-loop poles of (3-5),

$$\begin{aligned}
|z\mathbf{I} - (\mathbf{A}_d + \mathbf{B}_d\mathbf{K})| &= \begin{vmatrix} z - a_{11} - k_0 & -a_{12} - k_1 \\ -a_{21} & z - a_{22} \end{vmatrix} \\
&= z^2 + z(-k_0 - a_{11} - a_{22}) + (a_{22}k_0 - a_{21}k_1 + a_{11}a_{22} - a_{12}a_{21})
\end{aligned} \tag{3-6}$$

If the desired poles are at z_1 and z_2 then the desired characteristic polynomial is

$$\begin{aligned}
\phi^{\text{des}}(z) &= (z + z_1)(z + z_2) \\
&= z^2 + (z_1 + z_2)z + z_1z_2
\end{aligned} \tag{3-7}$$

The final solution for \mathbf{K} is obtained by setting (3-6) equal to (3-7) and matching coefficients. In MATLAB, the function *place* can solve for \mathbf{K} .

In order to drive the output to a desired level, we must introduce a constant reference input, $r(k)$, into the control law [3]:

$$u(k) = \mathbf{K}(\mathbf{x}(k) - \mathbf{N}_x\mathbf{r}) + \mathbf{N}_u\mathbf{r}. \tag{3-8}$$

In this structure, \mathbf{N}_x is a state command matrix whose value forces a system output to a desired reference level. \mathbf{N}_u is a proportionality vector. Using this method, the steady-state DC gain of the closed-loop system is forced to unity. \mathbf{N}_x and \mathbf{N}_u are found by the relationship [3]

$$\begin{bmatrix} \mathbf{N}_x \\ \mathbf{N}_u \end{bmatrix} = \begin{bmatrix} \mathbf{A}_d - \mathbf{I} & \mathbf{B}_d \\ \mathbf{C}_d & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix}. \tag{3-9}$$

Since the motor system is Type 1, the proportionality constant $\mathbf{N}_u=0$.

This discussion of state-variable feedback assumes the accessibility of the state variable vector $\mathbf{x}(k)$. The state variables defined in the discrete motor system model are not physically measurable. Fortunately a construct exists that will provide estimates of the state variable vector. The next section addresses this system.

3.2 Including a State Estimator (Observer)

A state estimator, or observer, is a linear system whose inputs are the plant's inputs and outputs and whose output is an estimate of the plant's state vector. This estimated vector is fed back into the plant in place of the true state vector. An observer also has noise reducing properties that minimize input and system noise effects on the output response. The observer designed here is a full-order observer. This means that the observer estimates all system states.

A closed-loop observer has the same basic structure as the plant, but with an additional input, the error signal $y - \hat{y}$. The error signal is used to force the output of the observer to converge with the true plant states. In equation form the observer is [4]

$$\begin{aligned}\hat{\mathbf{x}}(k+1) &= \mathbf{A}_d \hat{\mathbf{x}}(k) + \mathbf{B}_d u(k) + \mathbf{L}(y(k) - \hat{y}(k)) \\ \hat{y}(k) &= \mathbf{C}_d \hat{\mathbf{x}}(k)\end{aligned}\quad (3-10)$$

The matrix \mathbf{L} is a gain matrix. Substituting for $\hat{y}(k)$ gives

$$\begin{aligned}\hat{\mathbf{x}}(k+1) &= \mathbf{A}_d \hat{\mathbf{x}}(k) + \mathbf{B}_d u(k) + \mathbf{L}[y(k) - \mathbf{C}_d \hat{\mathbf{x}}(k)] \\ &= (\mathbf{A}_d - \mathbf{L}\mathbf{C}_d) \hat{\mathbf{x}}(k) + \mathbf{B}_d u(k) + \mathbf{L}y(k)\end{aligned}\quad (3-11)$$

The observer is made stable by properly placing the eigenvalues of the matrix $(\mathbf{A}_d - \mathbf{L}\mathbf{C}_d)$. The MATLAB function *place* can be used to find the gain \mathbf{L} .

Although not noted above, the preceding observer and controller design is performed with the understanding that the separation principle holds. This principle allows that the observer and controller can be designed independently. One important design consideration that must be kept in mind is that pole placement needs to be done considering the observer and controller together. For proper state estimate convergence, the observer poles need to be faster than the closed-loop plant poles. Therefore, the discrete observer poles should be closer to the origin on the complex plane than the closed-loop poles [4].

3.3 Adaptive Control

State-variable feedback provides a method to place closed-loop poles in a desired location; and the observer provides estimates of the true system states to feed back into the plant. Now to finish the control design we need a technique to estimate plant parameters. The parameters are necessary to tune the observer and controller for performance at an optimal level. In order to ensure stable system performance under varying conditions an adaptive control scheme is implemented. Adaptive control utilizes sampled system data to estimate system parameters. The estimated parameters are then used in the controller and observer equations. In this way, the control scheme can detect the system it is controlling and adapt to any changes that may occur in that system.

The adaptive control scheme chosen for this experiment is the Recursive Least Squares algorithm or RLS. Using RLS, the parameters of the discrete system model (A , B , C_1 , C_2) are directly estimated. First, the transfer functions for the outputs $Y_1(z)$ and $Y_2(z)$ are found. Recall from equation (2-14) that

$$\frac{Y_1(z)}{U(z)} = \frac{B}{z - A} \quad (3-12)$$

Using the result of equation (2-13) with equation (3-12), the transfer function from $Y_1(z)$ to $Y_2(z)$ is determined to be

$$\frac{Y_2(z)}{Y_1(z)} = \frac{C_1 z + C_2}{z - 1}. \quad (3-13)$$

Re-written in terms of z^{-1} , these become

$$\frac{Y_1(z)}{U(z)} = \frac{Bz^{-1}}{1 - Az^{-1}} \quad (3-14)$$

$$\frac{Y_2(z)}{Y_1(z)} = \frac{C_1 + C_2 z^{-1}}{1 - z^{-1}}. \quad (3-15)$$

These equations correspond to the following difference equations.

$$y_1(k) = Ay_1(k-1) + Bu(k-1) \quad (3-16)$$

$$y_2(k) - y_2(k-1) = C_1 y_1(k) + C_2 y_1. \quad (3-17)$$

Equations (3-16) and (3-17) are used in the RLS algorithm to solve for A , B , C_1 , and C_2 .

As the name implies, the RLS algorithm is based on the Gaussian principle of least squares. This principle dictates constraints under which the unknown parameters of a model should be chosen [5]. The least-squares method can be used to find system parameters in the following way. Consider the equation (3-16). Assume a sequence of p inputs has been applied and a corresponding sequence of p outputs has been observed. Define the unknown parameter vector, $\beta = [A, B]^T$. The least-squares estimate can be found from the representation

$$\begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-p) \end{bmatrix} = \begin{bmatrix} y(k-2) & u(k-2) \\ y(k-3) & u(k-3) \\ \vdots & \vdots \\ y(k-p-1) & u(k-p-1) \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \quad (3-18)$$

If we denote the left-hand vector θ and the right-hand matrix Ψ such that (3-18) is written

$$\theta = \Psi \beta, \quad (3-19)$$

then the least squares solution to this system is

$$\boldsymbol{\beta} = (\boldsymbol{\Psi}^T \boldsymbol{\Psi})^{-1} \boldsymbol{\Psi}^T \boldsymbol{\theta}. \quad (3-20)$$

New sequential data can be incorporated in (3-20) recursively [5]. See [5] for a discussion of Recursive Least-Squares estimation (RLS).

To illustrate the RLS adaptive algorithm, consider equation (3-16). Again, let $\boldsymbol{\beta} = [A, B]^T$ and define

$$\boldsymbol{\psi}(k) \equiv [y_1(k-1), u(k-1)]^T. \quad (3-21)$$

The algorithm can then estimate the parameter vector $\boldsymbol{\beta}$ by the following equations [6].

$$\mathbf{K}_w(k) = \mathbf{P}(k-1) \boldsymbol{\psi}(k) [\lambda \mathbf{I} + \boldsymbol{\psi}(k)^T \mathbf{P}(k-1) \boldsymbol{\psi}(k)]^{-1} \quad (3-22)$$

$$\mathbf{P}(k) = \frac{1}{\lambda} [\mathbf{I} - \mathbf{K}_w(k) \boldsymbol{\psi}(k)^T] \mathbf{P}(k-1) \quad (3-23)$$

$$\boldsymbol{\beta}(k) = \boldsymbol{\beta}(k-1) + \mathbf{K}_w(k) [y(k) - \boldsymbol{\psi}(k)^T \boldsymbol{\beta}(k-1)]. \quad (3-24)$$

Here, \mathbf{K}_w is a weighting matrix and \mathbf{P} is a proportionality matrix [5]. Also, $0 < \lambda \leq 1$ is a forgetting factor. New data are given full weight, but data that are m units old are weighted down by λ^m [6].

Once a solution for $\boldsymbol{\beta}$ is found, the estimated values of A and B (and C_1 and C_2) are used in controller calculations. In the observer equation (3-11), the parameter estimates are used in the system matrices A_d , B_d , and C_d as defined in equation (2-19). These estimated system matrices, denoted \hat{A}_d , \hat{B}_d , and \hat{C}_d , are then used to calculate the observer gain matrix \mathbf{L} and the state-variable feedback gain vector \mathbf{K} . For this implementation two $\boldsymbol{\beta}$'s are defined:

$$\boldsymbol{\beta}_1 = \begin{bmatrix} A \\ B \end{bmatrix} \quad \boldsymbol{\beta}_2 = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \quad (3-25)$$

This allows for two simultaneous RLS implementations with more compact matrix dimensions than if only one $\boldsymbol{\beta}$ is defined.

CHAPTER IV

CONTROL SYSTEM IMPLEMENTATION

The individual parts of the adaptive state-feedback control system are unified in the computer control system. The control system consists of four elements. These include the servomotor system, a Pentium 4 PC, a data acquisition card, and the software to drive the actual control. The motor system is described in Chapter II. The other parts of the control system are detailed here.

4.1 Hardware

The PC used in this experiment is a Dell 1.8GHz Pentium 4 with 512MB RAM. The operating system is Windows 98 Second Edition.

The data acquisition (DAQ) card used is a National Instruments 6024E multifunction I/O board for a PCI bus computer [7]. The 6024E uses built-in A-to-D and D-to-A converters and on-board timers and counters for data acquisition. The card has 16 channels of analog input, two channels of analog output, and eight lines of digital I/O. The card plugs into a spare PCI slot in the Dell PC and is automatically detected as a plug-and-play device by Windows.

For this thesis, only three channels of analog input and both channels of analog output are used. The input channels are configured for differential inputs. Each input channel's bipolar input range is set to $\pm 10V$. This sets a corresponding gain that makes the 12-bit analog-to-digital resolution precise to within 4.88mV. The output channels' ranges are fixed at $\pm 10V$.

In addition to the 6024E, a SCB-68 68-Pin shielded connector block is used to connect the motor system and instrumentation to the DAQ card. The SCB-68 connects to the 6024E with a SH6868 shielded 68-pin cable.

4.2 Software Overview

The software employed to implement the adaptive control scheme consists of three main bundles. First the NI-DAQ and Measurement & Automation Explorer programs are used to configure the 6024E card. Next LabVIEW is used to program the data acquisition interface. Finally MATLAB and Simulink are used to model the system and implement the RLS algorithm.

4.2.1 NI-DAQ and Measurement & Automation Explorer (MAX)

Just prior to installing the 6024E DAQ card in the PC, the NI-DAQ configuration software is installed. NI-DAQ is a utility that ships with the 6024E card that allows for easy configuration through a standard Windows GUI. Some of the functions that NI-DAQ can perform include buffered data acquisition, waveform generation, and counter/timer operations [7].

Also included with the NI-DAQ software is the Measurement & Automation Explorer (MAX) software. MAX enables quick setup of data acquisition devices and channels. The 6024E is the only device configured for this investigation. Table 4.1 lists the channels that are configured.

Table 4.1 MAX Channel Configuration

Name	Channel	I/O Connector Pins	Connected Signal
AnalogIn1	Input 0	ACH0-ACH8	V_{pot} from output potentiometer
AnalogIn2	Input 1	ACH1-ACH9	V_{tach} from motor tachogenerator
AnalogIn3	Input 2	ACH2-ACH10	$u(k)$ from AnalogOut1 for charting
AnalogOut1	Output 0	DAC0OUT-AOGND	$u(k)$ to pre-amplifier input and AnalogIn3
AnalogOut2	Output 1	DAC1OUT-AOGND	GP-6 OP to toggle OP/IC mode

4.2.2 LabVIEW 6i

The NI-DAQ and MAX software integrates seamlessly with National Instruments graphical automation program LabVIEW [8]. LabVIEW version 6i is used for this experiment. LabVIEW allows a user to design and test a control program by wiring graphic icons together on a workspace. The LabVIEW interface is separated into two main regions. The user interface region is called the *front panel*. The front panel displays any necessary controls (i.e. switches, dials, pushbuttons) and indicators (like graphs, LEDs, textboxes) that would be manipulated by the end user to interface with the external system. The second region is the *block diagram*. This area contains the code to drive the front panel devices. The code is ‘written’ graphically using icons connected in a flowchart format.

A LabVIEW program is called a *virtual instrument* or VI. This name stems from the resemblance of the front panel to actual laboratory instruments like multimeters or oscilloscopes. A VI can contain other VIs that are called subVIs.

4.2.3 MATLAB/Simulink

MATLAB [9] is a powerful matrix-oriented programming and data visualization software package. It offers a wide array of problem-solving functions and tools for solving engineering and mathematical problems. Simulink [10] is an extension of MATLAB that is used to model dynamic systems. Simulink is programmed by creating block diagrams in the Simulink workspace. Once the model is simulated, MATLAB functions can easily be used to manipulate or plot the resulting data.

4.3 Simulink Model

Using MATLAB and Simulink, the complete adaptive motor-control system is simulated and the results plotted to establish an expectation of real-world system performance.

The model begins as a Simulink block diagram. The motor system (plant) is programmed as a discrete state-space block. The nominal values of motor parameters ($\tau_m=0.25$, $K_m=6.5$, and $K_p=6$) are used to calculate the state-space system matrices using equations (2-16) and (2-19).

The vector output of the plant block is fed into the RLS algorithm block. The RLS equations of (3-22) to (3-24) are programmed in a customizable Simulink block called an S-Function. An S-Function block allows you to program custom functions using either C-language code or MATLAB code. In this simulation, MATLAB code is employed.

The basic program structure in MATLAB is the m-file. Similar to a DOS batch file, an m-file consists of MATLAB functions that are executed in sequence when the m-file name is called. An S-Function block allows you to incorporate the m-file code into a standard S-Function program template. When the S-Function block is called, the program executes and the resulting data can be output and used elsewhere in the simulation. The sections of the S-Function template modified for this model are listed in Appendix Section A.2. Figure 4.1 shows the graphical input and output vectors of the S-Function block.

The outputs of the S-Function define the controller for the simulation. First, the full-order observer equations of (3-11) are drawn out using Simulink math function blocks. (Figure 4.2 shows the observer structure.) Then the parameters calculated by the S-Function code are input into the observer blocks to find the state estimates, \hat{x}_1 and \hat{x}_2 . These parameters include the observer A_d-LC_d matrix and the gain matrix L . In Figures 4.1 and 4.2, the A_{mn} values are the elements of the A_d-LC_d calculation. Similarly, the L_{mn} values are the elements of the L matrix. The observer state estimates, denoted $x1hat$ and $x2hat$ in Figure 4.2, are fed back to the plant input through the state-feedback control structure.

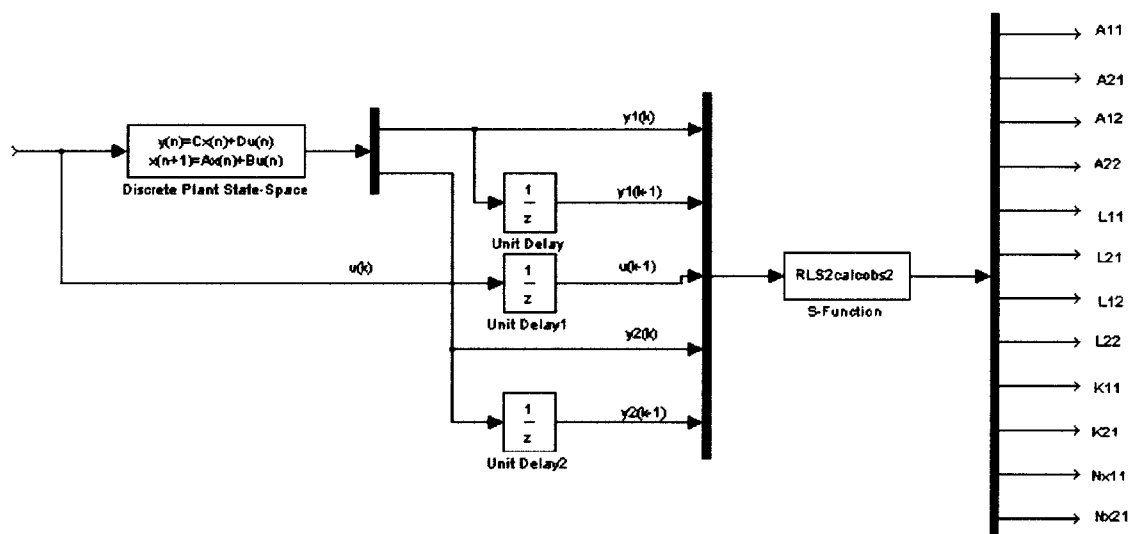


Figure 4.1 Simulink S-Function implementation of RLS algorithm.

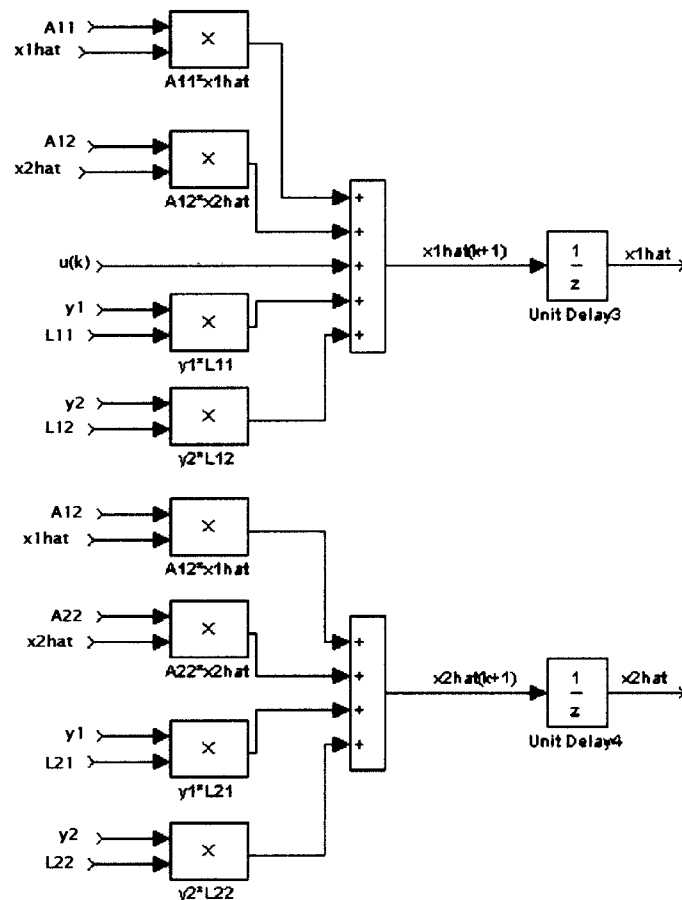


Figure 4.2 Simulink implementation of the full-order observer.

The state-variable feedback control detail is presented in Figure 4.3. Like the observer, the equations are drawn out using mathematical function blocks. The gain vector \mathbf{K} and the reference input parameter vector \mathbf{N}_x are calculated in the RLS S-Function program. The reference input signal is a 5V 0.1Hz square wave generated by the Signal Generator block. This reference signal is discretized by the Zero Order Hold block with a sample rate of $T=0.1s$.

Before the model is simulated, an initialization routine is necessary to set up global variables and initial values for the RLS S-Function. Global variables can be passed between the MATLAB workspace and the S-Function workspace. These variables include parameter vectors β_1 and β_2 from equation (3-25) and two proportionality matrices \mathbf{P}_1 and \mathbf{P}_2 from (3-23). Other global variables are created for data plotting purposes. The initialization routine sets initial 'guesses' of the β and \mathbf{P} parameters. The closed-loop and observer poles are set and initial values are calculated for the closed-loop gain \mathbf{K} and observer gain \mathbf{L} . The initialization routine is an m-file and is listed in Appendix Section A.1.

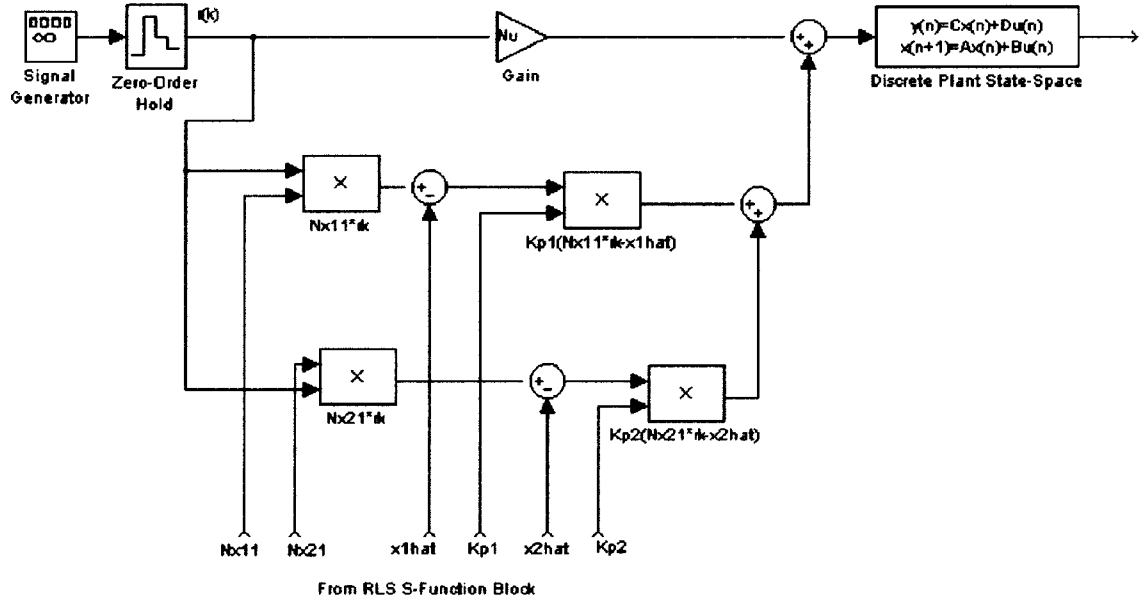


Figure 4.3 Simulink state-variable feedback control implementation.

After the initialization file is run, the Simulink model can be executed. Initially, the Simulink model is tested using the exact values of the system parameters found via (2-16). These values are $A=0.67$, $B=-2.14$, $C_1=0.32$, and $C_2=0.28$. Initializing the model with these values means that no adaptation need occur. In this way, the effects of closed-loop and observer pole selection are more readily apparent. Several simulations are run to find poles that produce good transient performance. The final pole locations are listed in table 4.2.

With the Simulink model complete, the next step in this investigation is to test the controller design in a real-world system. This is accomplished by integrating LabVIEW and MATLAB with the data acquisition hardware into one complete adaptive control system.

Table 4.2 Simulink model pole locations.

	Discrete	Analog Equivalent
Closed-Loop Poles	$0.667+0.067j$	$-4+j$
	$0.667-0.067j$	$-4-j$
Observer Poles	0.407	-9
	0.368	-10

4.4 LabVIEW / MATLAB Implementation

The software program that is actually used to control the external system is written in LabVIEW using embedded MATLAB scripts. This main VI is based on an example VI included with the LabVIEW software. This example is “Analog I/O Control Loop (Hardware-Timed)” [11]. The example VI is modified from a simple hardware-timed analog I/O loop into the desired adaptive motor control program.

The complete adaptive control VI block diagram is shown in Figure 4.4. (The two MATLAB script boxes do not show the complete script files. These are listed in Appendix Sections B.1 and B.2.) The blocks along the left side of the diagram are for initialization. The AI Config subVI configures a set of channels for analog input. Controls on the front panel (shown in Figure 4.5) set the device number and specific channels for this block. For this setup, the device is ‘1’ and the input channels are ‘AnalogIn1’, ‘AnalogIn2’, and ‘AnalogIn3’. Also in this application the buffer size is set to zero for an unbuffered acquisition. Thereby no data will be stored. The LabVIEW program must read each scan before it is overwritten by the next scan, or an error will result.

The next subVI is the AI Start VI. This block starts the analog data acquisition. The number of scans to acquire is set to zero for continuous acquisition and the scan rate is set to 10 scans/sec to get the sample period, $T=0.1s$. The AI Start VI starts the scan clock on the 6024E card and data is acquired into the on-board FIFO stack at the set scan rate.

Other initialization functions that are performed include clearing the waveform chart at the start of each run. This is accomplished using the History block at the top of the diagram. Overwriting the chart’s data history array with zeros clears the waveform chart. The AO One Pt block executes the AO Update Channel subVI. This subVI sends the specified value (here $-5V$) to the channel indicated. This value is sent to a GP-6 analog computer’s SW input to switch the GP-6 into OP mode when the VI is run. This ensures that the analog test system (described in Section 4.5) is synchronized with the data acquisition.

The last part of the initialization segment of the VI is the MATLAB script node in the bottom-left corner of Figure 4.4. It is one of two MATLAB scripts in the VI. These MATLAB scripts are essentially m-files that execute when LabVIEW processes the MATLAB script node block. The MATLAB script nodes invoke the MATLAB script server to execute the code. LabVIEW uses ActiveX to implement MATLAB script nodes so they are only available on Windows platforms. This particular script, like the Simulink initialization m-file, defines variables and sets up initial values that are necessary for the second MATLAB script in the main section of the VI.

This main section is the *while* loop. The while loop is the large, central grey box in the VI. This loop is programmed to cycle until the number of requested samples have been taken, or the stop button is pressed. The AI S-Scan block reads the sampled data from the FIFO on the 6024E card. In the first loop iteration, the newest (most recent)

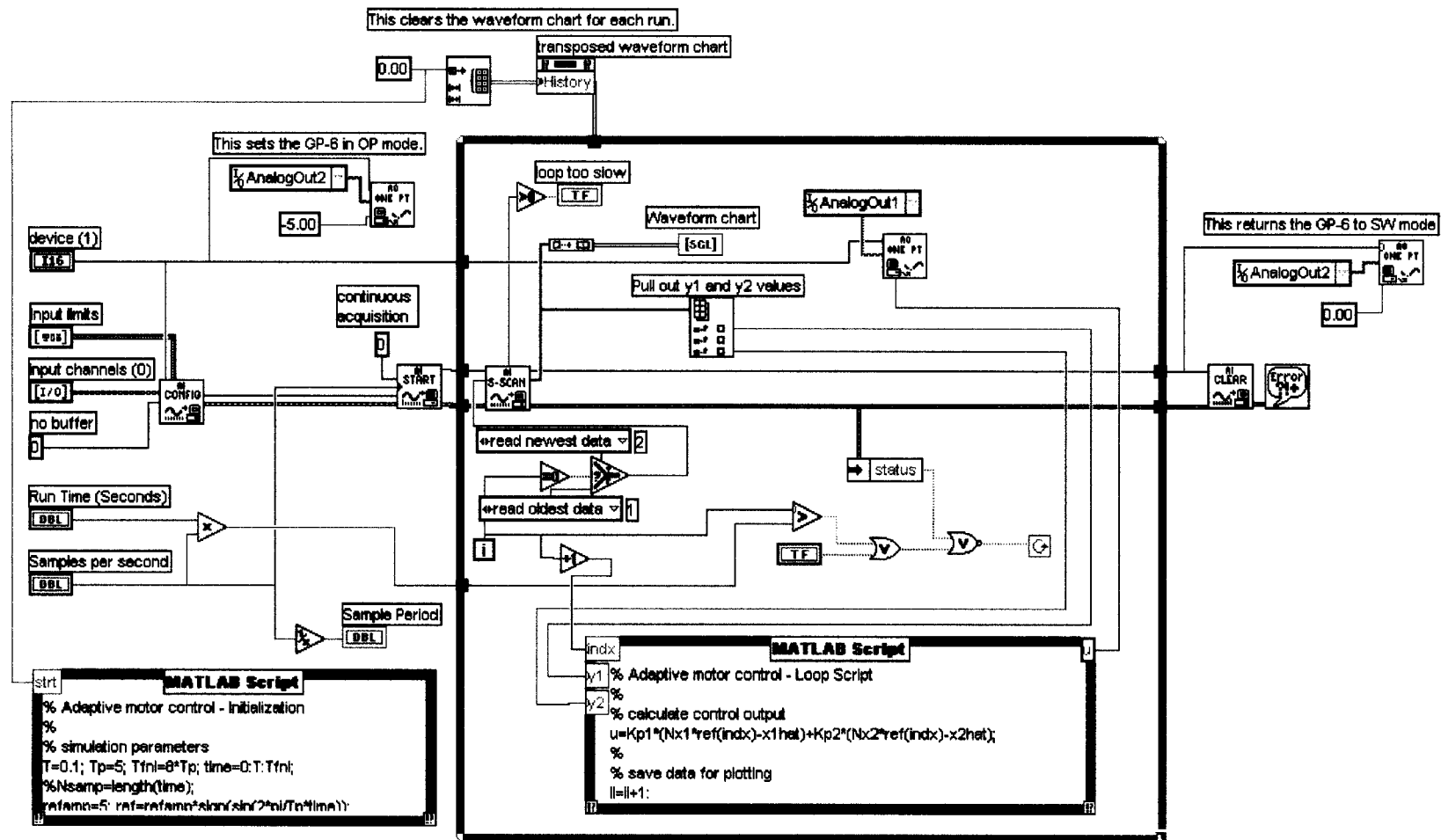


Figure 4.4 LabVIEW VI block diagram.

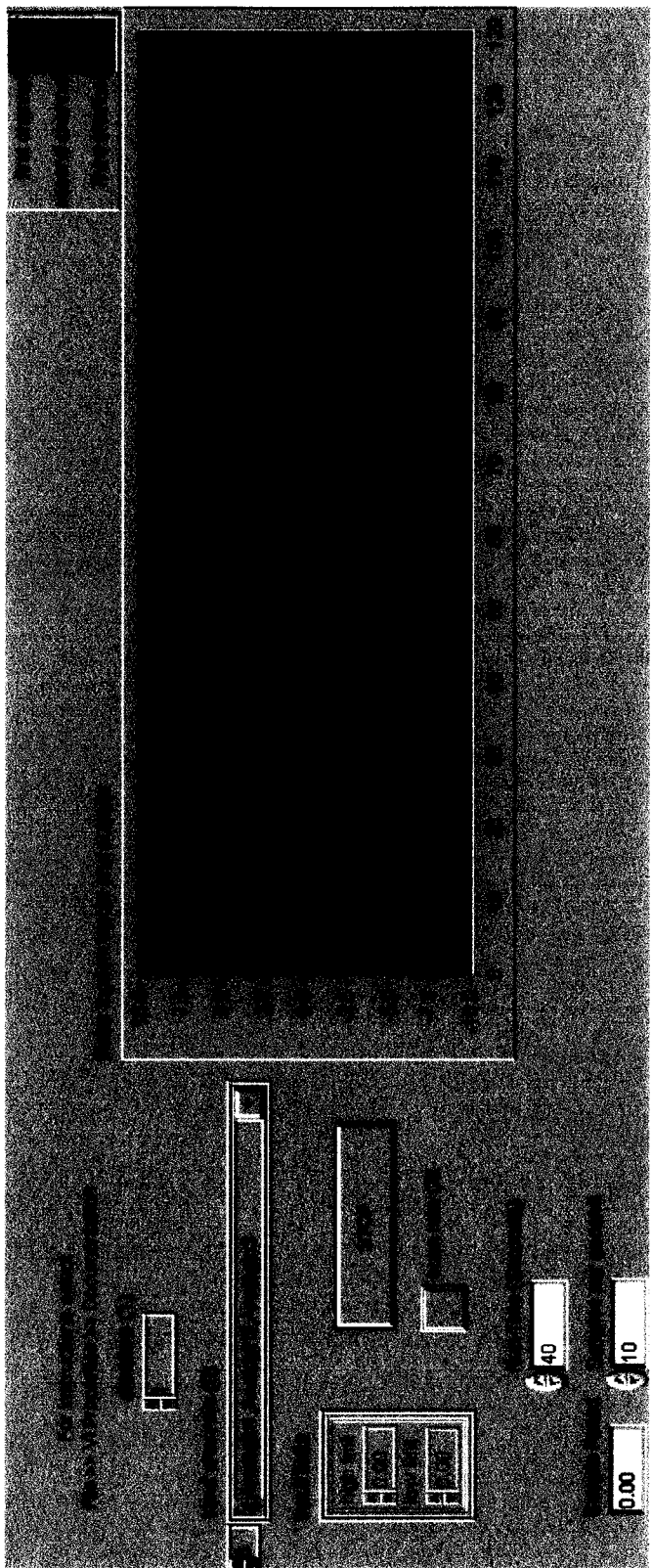


Figure 4.5 LabVIEW VI front panel.

scan is read from the FIFO. In subsequent loop iterations AI S-Scan is called to read the oldest scan from the FIFO. Each scan is plotted on the waveform chart as it is read. If a scan is missed because the acquisition does not keep up, an error message is displayed. In the event of an error, the acquisition will continue, however, some data will have been lost.

During each cycle of the while loop, the MATLAB script in the loop is executed once. On the first scan, initialization data from the first MATLAB script is used to calculate the control signal u . This script also uses the scanned data to calculate the system, observer, and state-feedback parameters that will be used in the next iteration to calculate the new u value. The u value is output to the motor system by the AO One Pt block. Also included in this script are storage arrays to hold data for plotting in MATLAB after the test run is complete.

When the requested number of scans have been acquired and processed, the while loop terminates and one final step occurs. The AO One Pt. block outside the loop sends a +5V signal to the GP-6 system to put it back into IC mode. After this, the program stops and the results can be plotted in MATLAB.

The LabVIEW program was first tried on an analog test system.

4.5 Analog Test System

A test system was built on a Comdyna GP-6 analog computer to simulate the motor system and to test the LabVIEW control program. This also allowed the motor system results to be compared to a known quantity since the plant on the analog computer was exactly defined.

The motor transfer functions of equations (2-1) and (2-2) were set up in the block diagram of Figure 4.6(a). Using nominal motor parameters: $K_m=6.5$, $\tau_m=0.25$, $K_p=6$, the new block diagram of Figure 4.6(b) can be defined. This yields the continuous time system

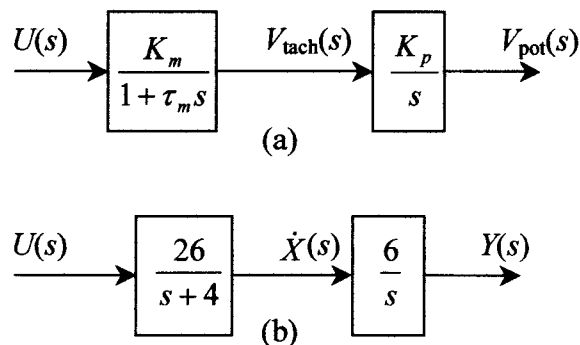


Figure 4.6 (a) Motor system s-domain block diagram. (b) Block diagram with values.

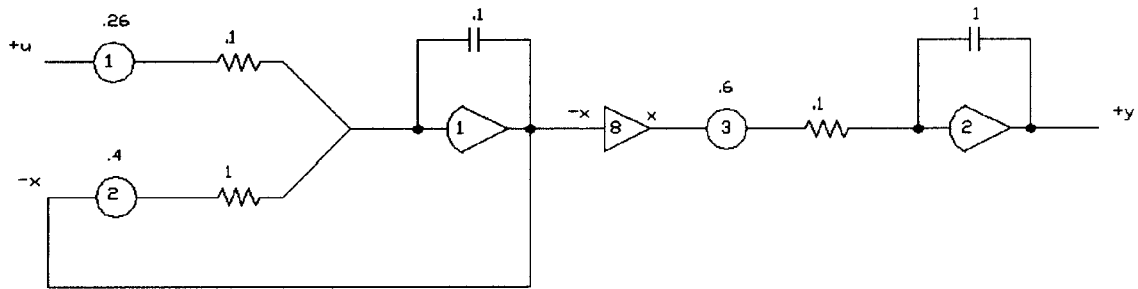


Figure 4.7 GP-6 Analog Computer test circuit.

$$\begin{aligned}\dot{x}(t) &= -4x(t) + 26u(t) \\ y(t) &= 6x(t)\end{aligned}\tag{4-1}$$

This system was then implemented on the GP-6 in the circuit of Figure 4.7.

The GP-6 system was initially excited open loop for comparison to the open-loop motor response. The GP-6 open-loop response to a 1V square wave is shown in Figure 4.8. Comparison with the motor response in Figure 2.4 verifies that the analog test system responds similarly to the actual system with two major exceptions. First, the GP-6 is a $\pm 10\text{V}$ system. Therefore the V_{pot} signal saturates at 10V instead of reaching 15V as the motor system does. The second major difference in the system responses is the lack of the sawtooth pattern in the V_{pot} signal. The system equations do not model the discontinuity of the output potentiometer. To ensure accurate results, a restraint is placed on the closed-loop tests that the V_{pot} signal cannot exceed $\pm 7\text{V}$. This will prevent saturation in the GP-6 system and avoid the unmodeled, discontinuous region.

Once the GP-6 system was tested, the final step in this investigation was to wire the actual motor system to the LabVIEW control program. Initial tests on both the GP-6 and motor systems show that the closed-loop and observer poles chosen for the Simulink model provide good results. Therefore the poles of Table 4.1 apply also to the GP-6 and motor tests. All of the results of the Simulink, GP-6, and servomotor simulations are presented in Chapter V.

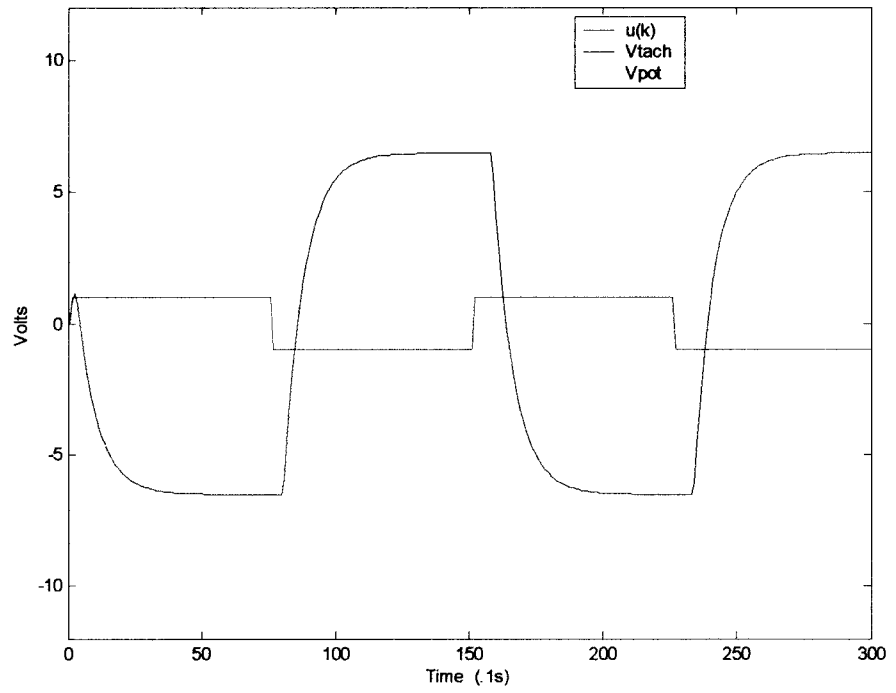


Figure 4.8 GP-6 test system open-loop response to a 1V step input.

CHAPTER V

RESULTS

The results of applying a 5V 0.1Hz square wave reference signal to each of the systems are presented in this chapter. Each system was tested under similar sets of initial estimates of the system parameters, A, B, C₁, and C₂. For compactness of notation, define the term $\beta_N = [A \ B \ C_1 \ C_2]^T$ where A, B, C₁, and C₂ are the nominal (or 'true') parameter values for each system. Also let β_0 represent the initial parameter values set at the start of each trial. Generally two to four plots are given for each trial of a system. The plots fall into four categories:

- a) potentiometer output V_{pot} and the reference input signal $r(k)$
- b) parameter estimates calculated by the RLS algorithm
- c) true system states and observer state estimates
- d) control input $u(k)$ and tachogenerator output V_{tach} .

Where beneficial, time-expanded or time-compressed plots are presented for clarity. The Simulink model results are given first.

5.1 Simulink Model

For the first simulation, the Simulink model was initialized with $\beta_0 = \beta_N = [0.67 \ -2.14 \ 0.32 \ 0.28]^T$. The Simulink system was simulated for 600 sample periods (60 seconds). The resulting signals are plotted in Figures 5.1 to 5.4. Inspection of Figure 5.1, the reference/output plot, shows a reasonable rise time and no overshoot in the output signal. Figure 5.2 shows the parameter estimates as they remain constant at their true values. Figure 5.3 compares the true system states to the observer states. The observer states track with zero error. Finally, in Figure 5.4, the input signal and tachometer voltage are plotted. The control signal is very reasonable staying within $\pm 1.2V$.

For the next simulation, the parameter values were initialized at one-half their nominal values i.e. $\beta_0 = 1/2(\beta_N)$. The resulting signals are plotted in Figures 5.5 to 5.10. Inspection of Figure 5.5 shows initial overshoot in the response to the rising edge of the first reference pulse. The subsequent pulses, both positive and negative show no overshoot. This initial overshoot can be explained with the next plot in Figure 5.6. Figure 5.6 shows the parameter estimates. During the initial samples (expanded in Figure 5.7) the parameter estimates deviate from the actual values. During this transient period, the control system is not tuned to the plant and therefore some initial under-control is apparent. Figure 5.8 compares the true system states to the observer states. Figure 5.9 shows only the first ten samples of the system states. The initial transient region is seen in this plot, but after approximately eight samples the observer states track with zero error. Finally, in Figure 5.10, the input signal and tachometer voltage are plotted.

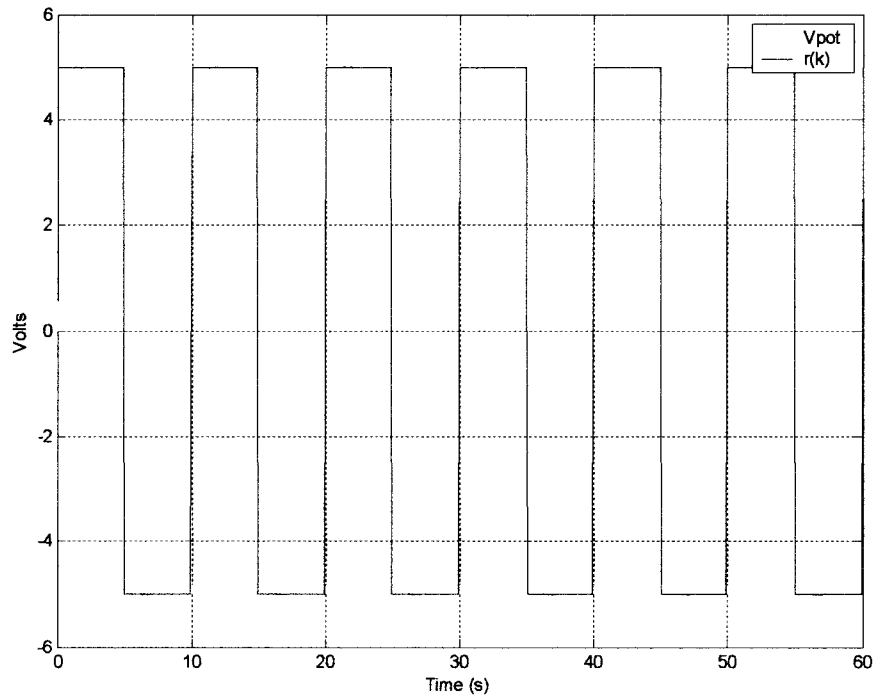


Figure 5.1 Simulink closed-loop response to a 5V square wave. $\beta_0 = \beta_N$.

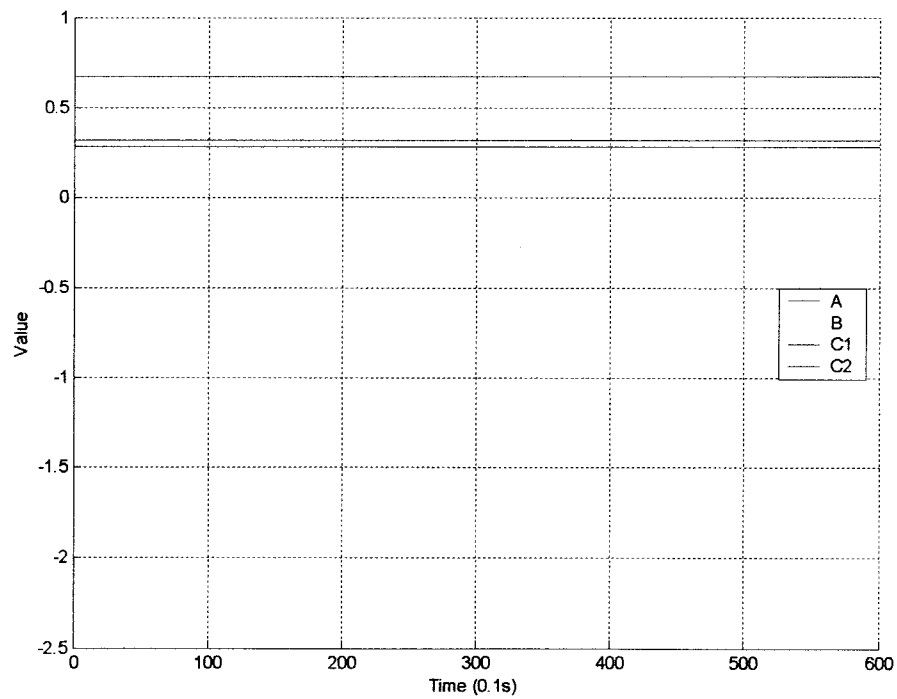


Figure 5.2 Simulink calculated system parameters. $\beta_0 = \beta_N$.

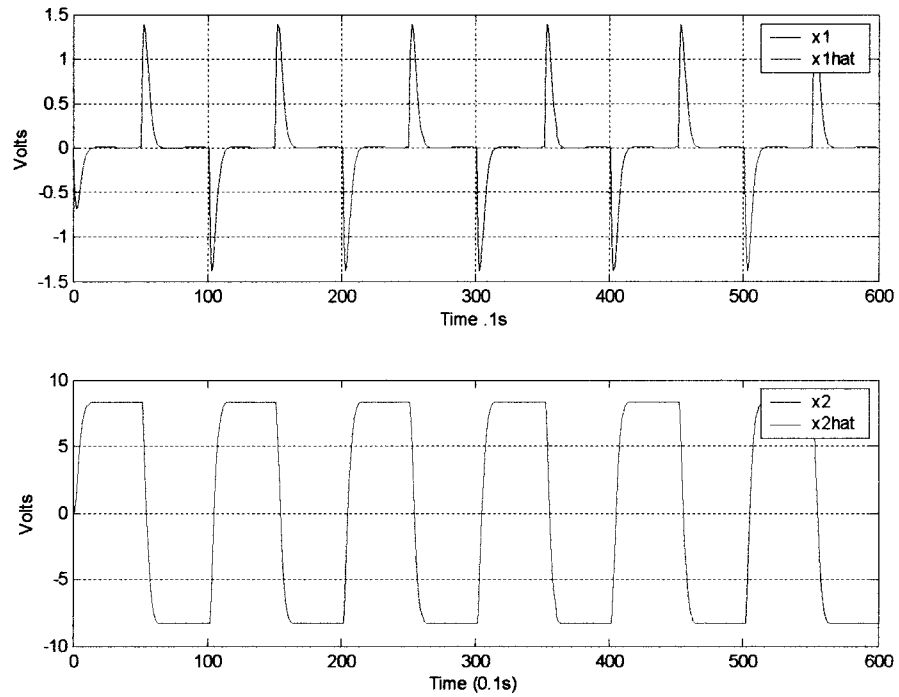


Figure 5.3 Simulink closed-loop states and observer states. $\beta_0 = \beta_N$.

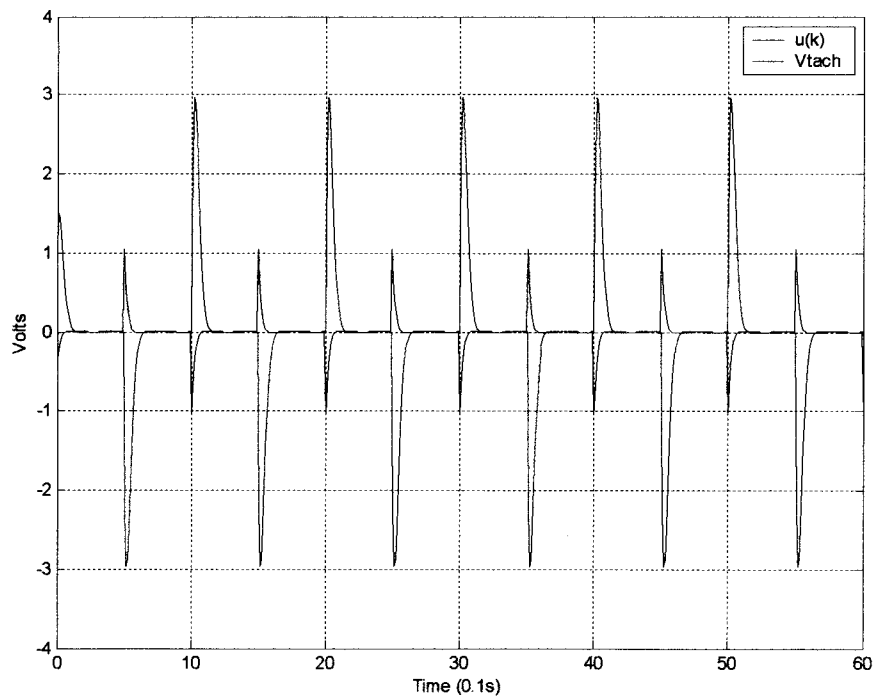


Figure 5.4 Simulink input $u(k)$ and tachogenerator output V_{tach} . $\beta_0 = \beta_N$.

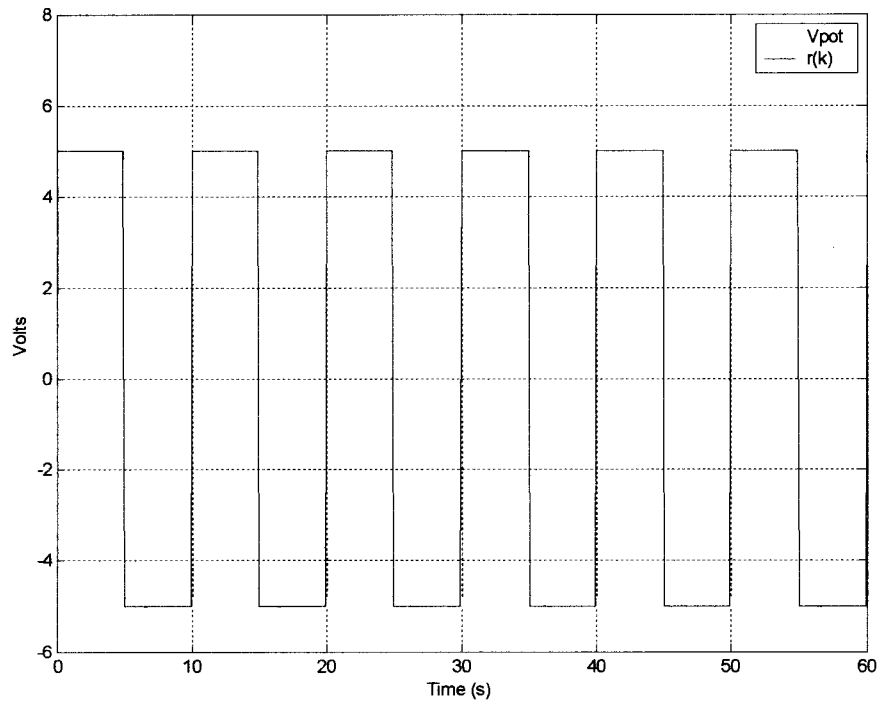


Figure 5.5 Simulink closed-loop response to a 5V square wave. $\beta_0=1/2\beta_N$.

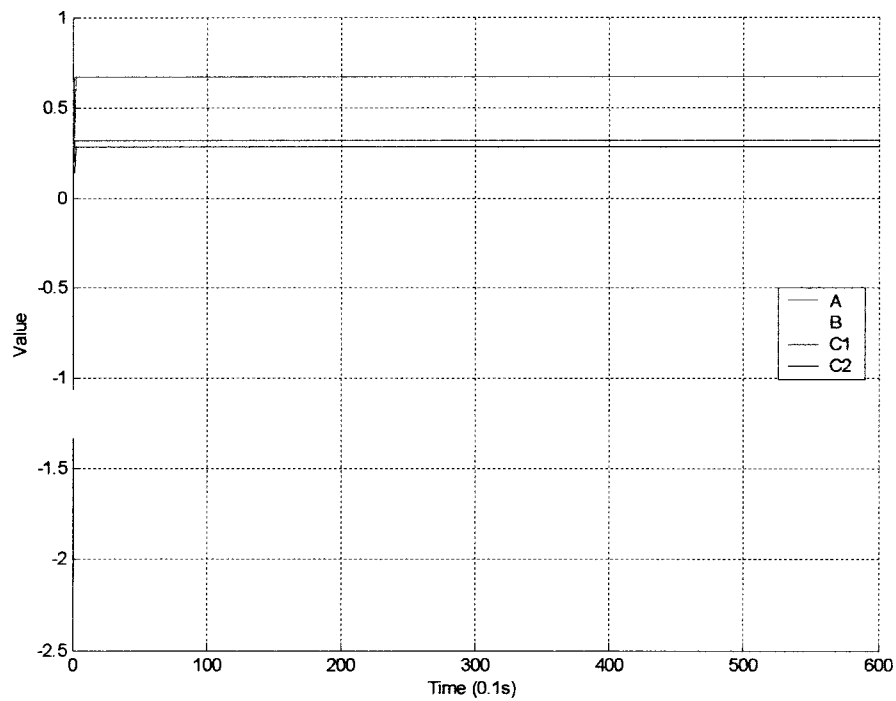


Figure 5.6 Simulink calculated system parameters. $\beta_0=1/2\beta_N$.

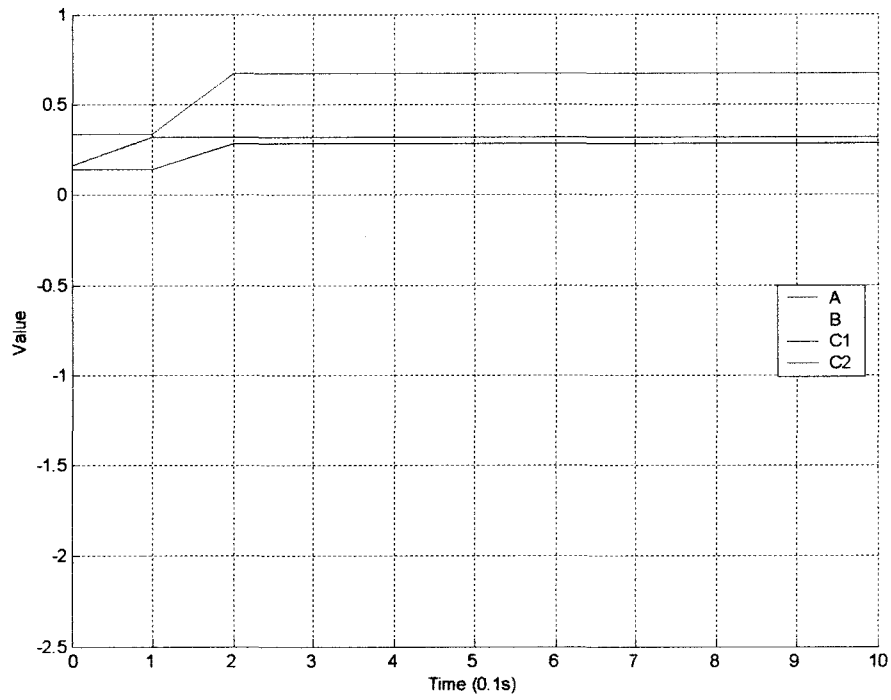


Figure 5.7 Simulink calculated system parameters (first ten samples). $\beta_0=1/2\beta_N$.

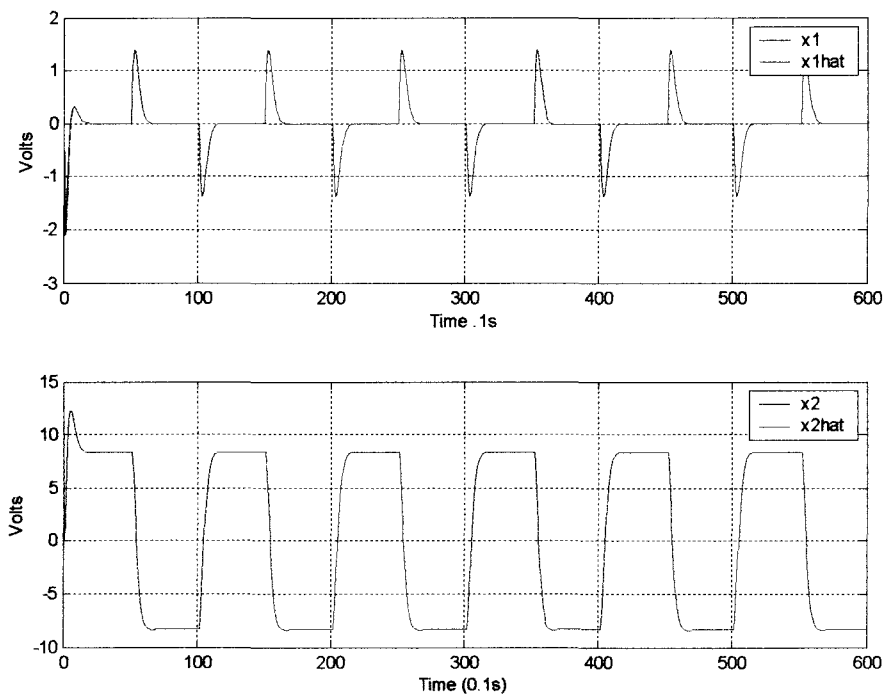


Figure 5.8 Simulink closed-loop states and observer states. $\beta_0=1/2\beta_N$.

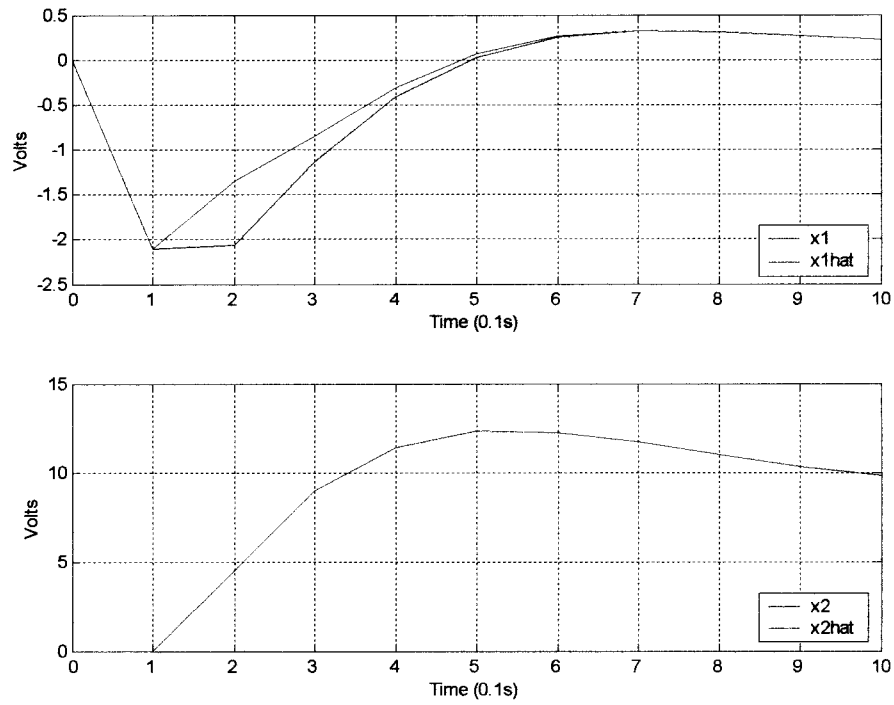


Figure 5.9 Simulink closed-loop states and observer states (first 10 samples). $\beta_0=1/2\beta_N$.

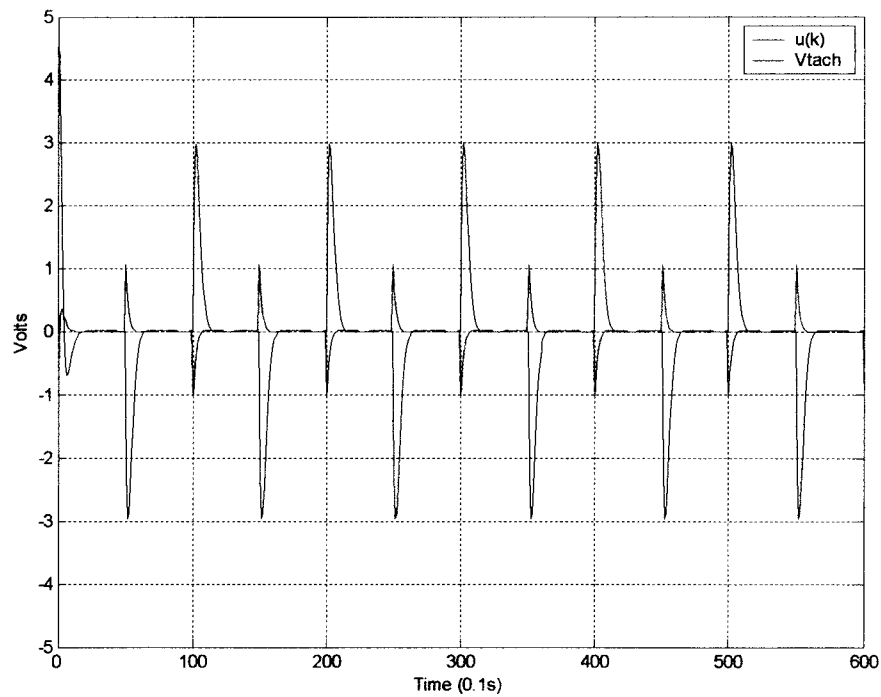


Figure 5.10 Simulink input $u(k)$ and tachogenerator output V_{tach} . $\beta_0=1/2\beta_N$.

Additional simulations were run with other initial parameter values. Figures 5.11 and 5.12 show the output and parameters respectively for initial values set at $\beta_0=2\beta_N$. In these plots, the initial overshoot is smaller than when $\beta_0=1/2\beta_N$. However, the initial rise time increases as compared to Figure 5.5. Figures 5.13 and 5.14 show the results for $\beta_0=4\beta_N$. As Figure 5.13 shows, there is no initial overshoot, but the rise time is significantly longer than any of the previous simulations. Some initial instability seems to be the cause of the slow response to the first pulse.

One more simulation is presented with interesting results. Figures 5.15 and 5.16 show the response when $\beta_0=-\beta_N$. Although the parameter estimates quickly converge to the nominal values, a large spike appears in the output response to the first input pulse.

In all the Simulink trials, the system parameter estimates always converged to the exact values set in the model plant.

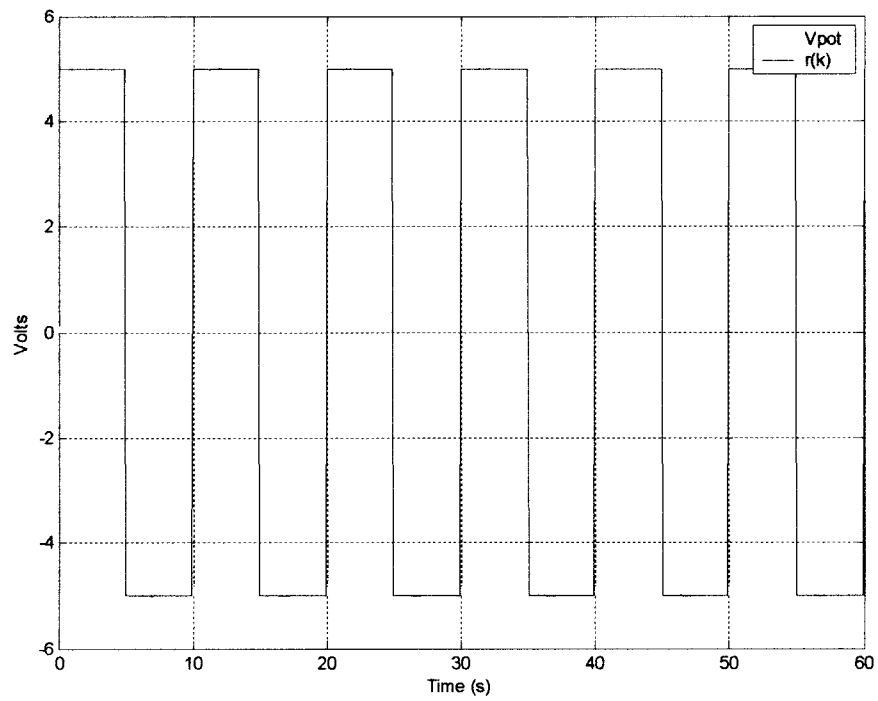


Figure 5.11 Simulink closed-loop response to a 5V square wave. $\beta_0=2\beta_N$.

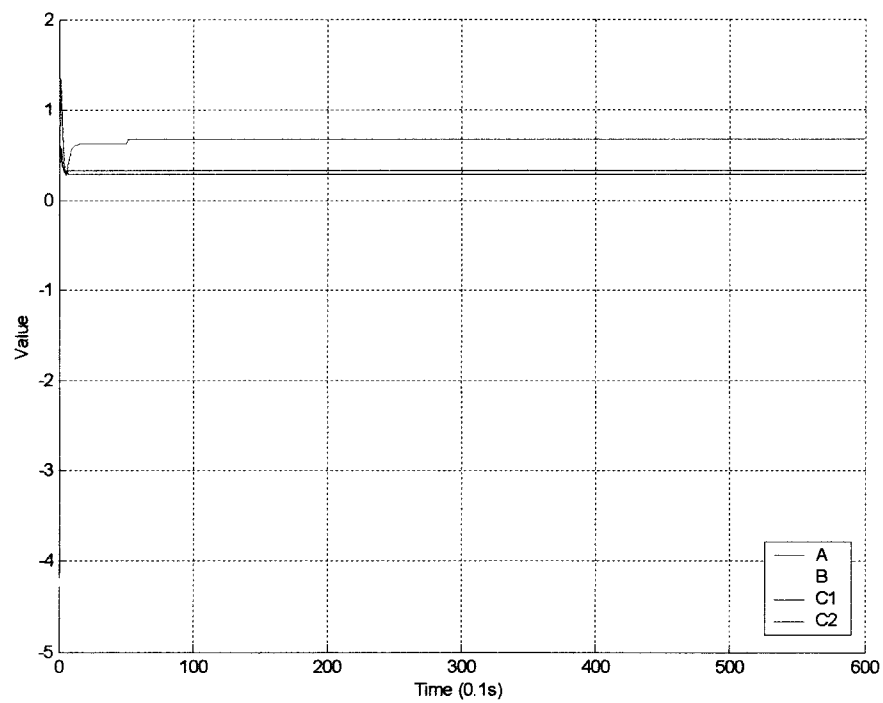


Figure 5.12 Simulink calculated system parameters. $\beta_0=2\beta_N$.

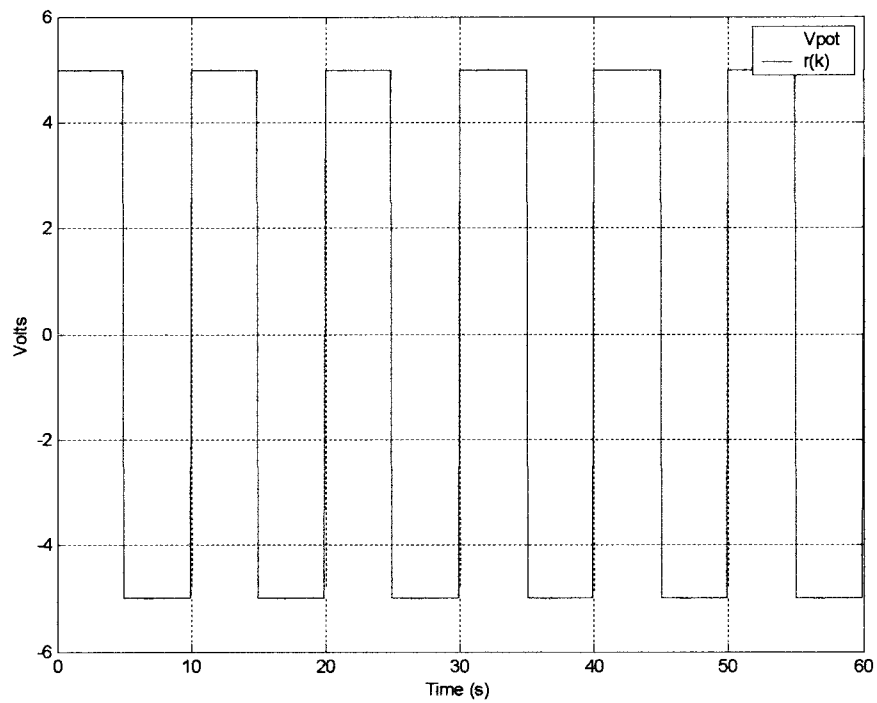


Figure 5.13 Simulink closed-loop response to a 5V square wave. $\beta_0=4\beta_N$.

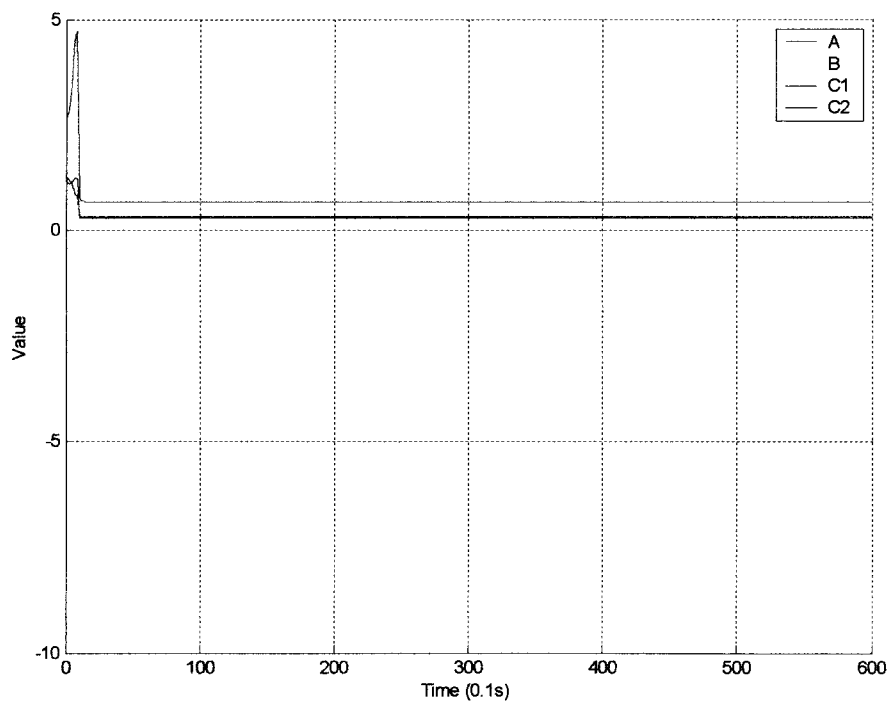


Figure 5.14 Simulink calculated system parameters. $\beta_0=4\beta_N$.

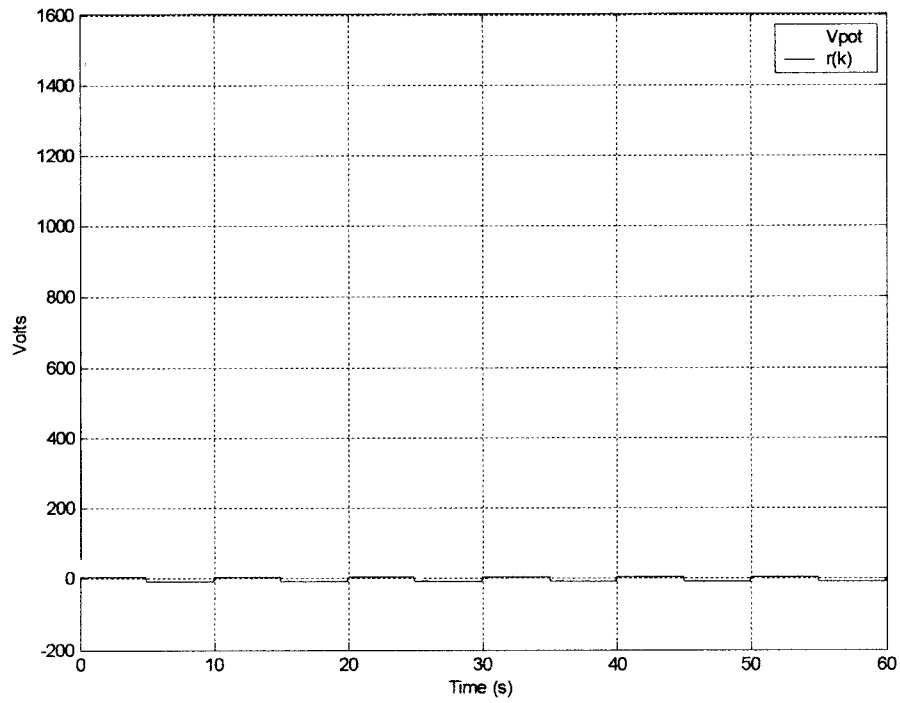


Figure 5.15 Simulink closed-loop response to a 5V square wave. $\beta_0 = -\beta_N$.

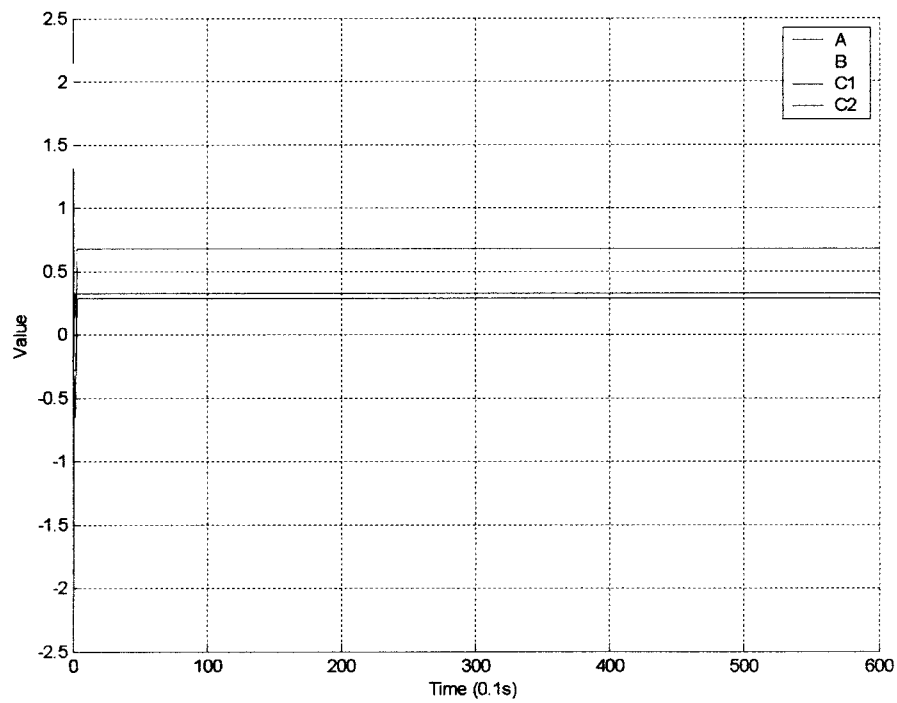


Figure 5.16 Simulink calculated system parameters. $\beta_0 = -\beta_N$.

5.2 Controlled GP-6 System

When the GP-6 system was wired to the LabVIEW control system, the resulting system response varied somewhat from the Simulink expected data. The initial trial with $\beta_0 = \beta_N$ responded close to the expected results. The plots in figures 5.17 through 5.20 show the closed-loop signals for the first GP-6 simulation. Inspection of Figure 5.17 shows a well-controlled output with no overshoot. Figure 5.18 presents the parameter estimates as they converge to the nominal values. It is noteworthy that the GP-6 system parameters converge to different values than the Simulink model. In Figure 5.18, these values are $A=0.77$, $B=-1.61$, $C_1=0.27$, and $C_2=0.33$. For the rest of the GP-6 trials $\beta_N = [0.77 \ -1.61 \ 0.27 \ 0.33]^T$.

In the Simulink model, all the signals are calculated mathematically; therefore we have access to all the signals in the system. In the GP-6 tests, the true system states cannot be measured directly. For this reason another method of verifying the state estimates is used. This involves comparing the measured system outputs, $y_1(k)$ and $y_2(k)$, with estimates of these same outputs. The output estimates, $\hat{y}_1(k)$ and $\hat{y}_2(k)$, are calculated using the RLS \hat{C} matrix and the estimated state vector \hat{x} in the following relationship:

$$\hat{y}(k) = \hat{C}\hat{x}(k) \quad (5-1)$$

Figure 5.19 compares the measured system outputs to calculated outputs and Figure 5.20 shows the control signal and the V_{tach} output.

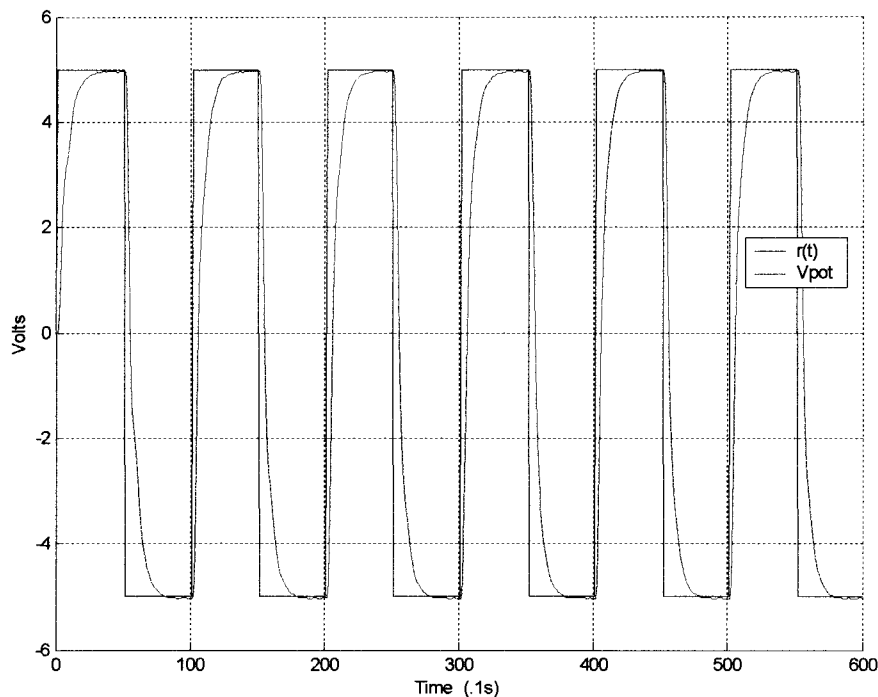


Figure 5.17 GP-6 closed-loop response to a 5V square wave. $\beta_0 = \beta_N$.

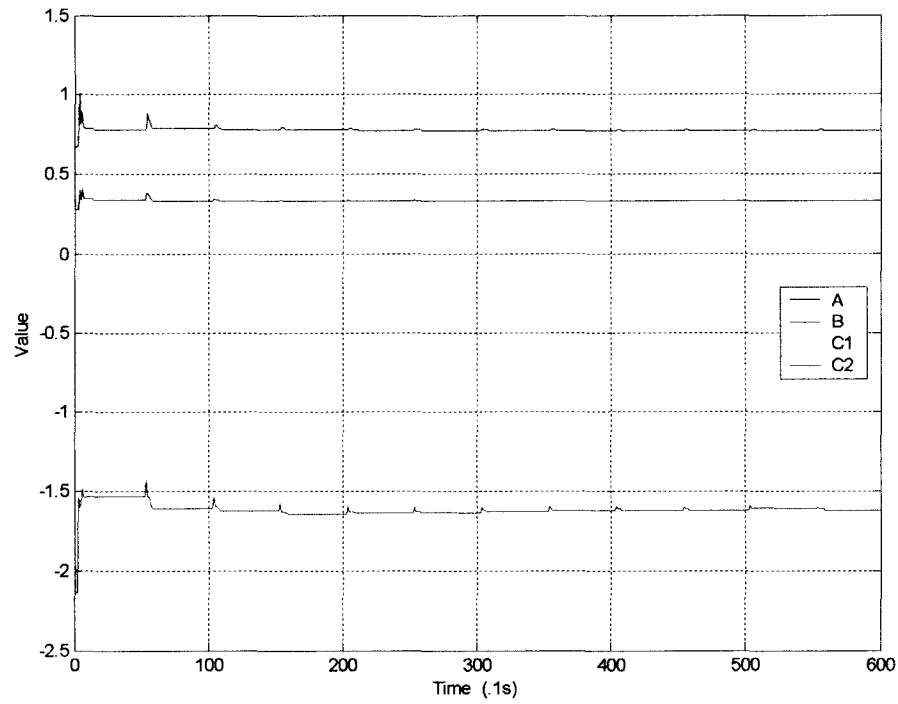


Figure 5.18 GP-6 calculated system parameters. $\beta_0 = \beta_N$.

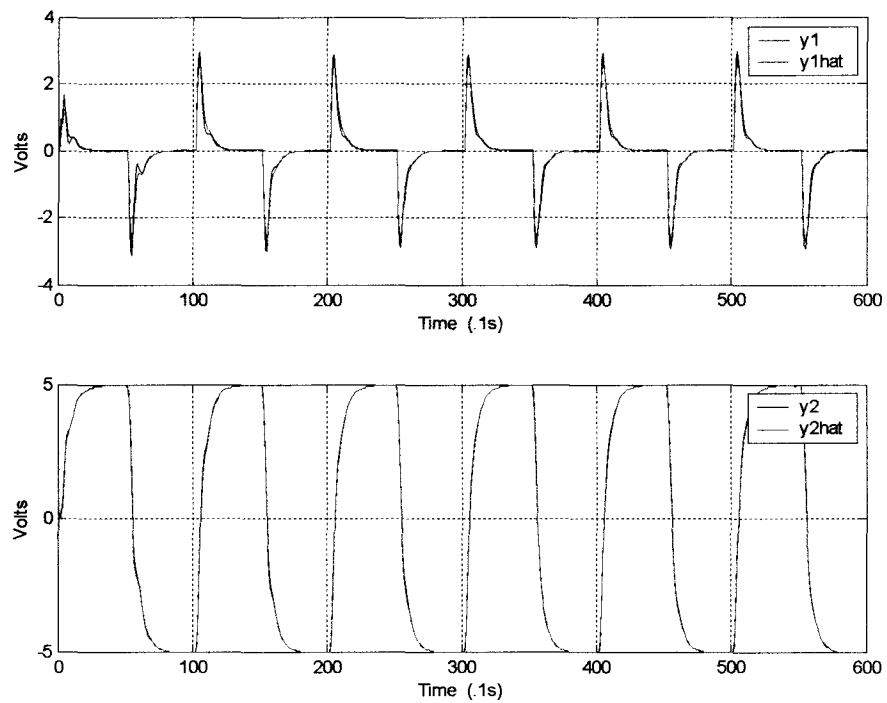


Figure 5.19 GP-6 closed-loop output and estimated output. $\beta_0 = \beta_N$.

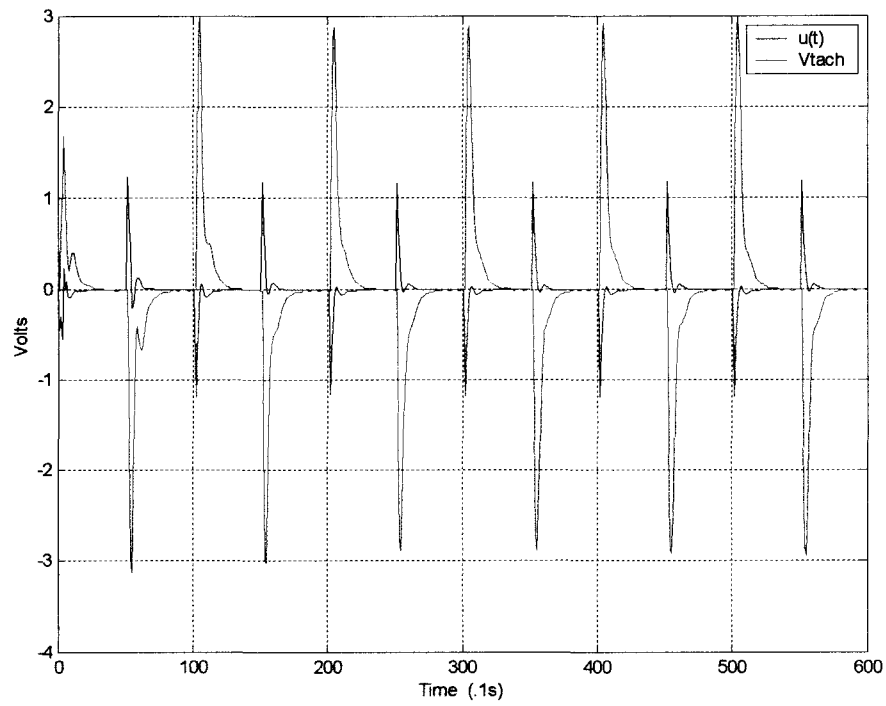


Figure 5.20 GP-6 input $u(k)$ and tachogenerator output V_{tach} . $\beta_0 = \beta_N$.

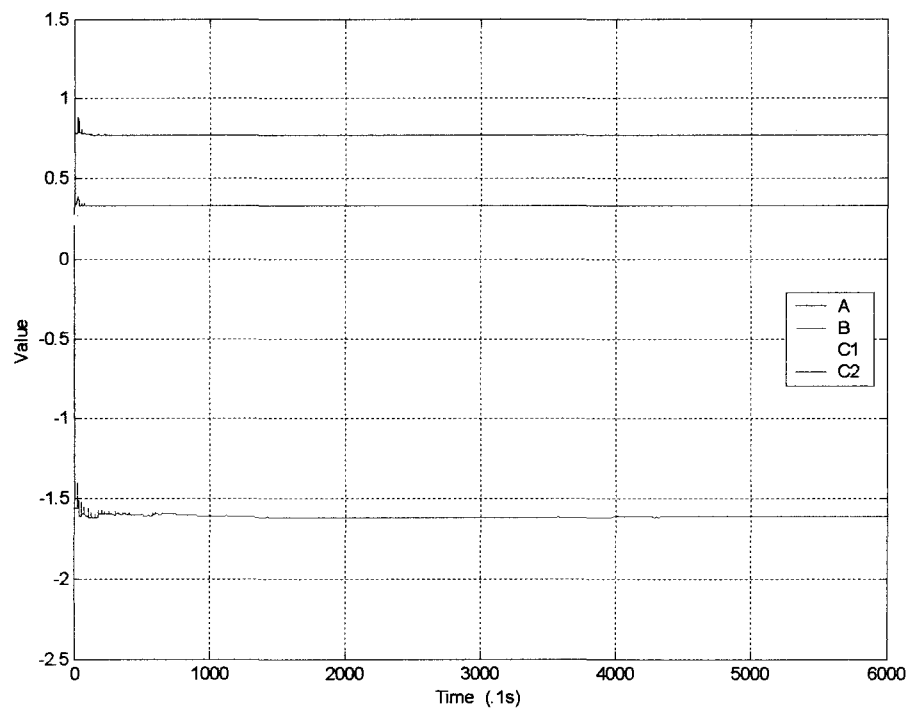


Figure 5.21 GP-6 system parameters after 10 minutes. $\beta_0 = \beta_N$.

As an additional experiment, the GP-6 system was allowed to cycle for 6000 samples, or 10 minutes. The plot in Figure 5.21 shows that almost no parameter variation is seen after 1000 samples. This confirms that the RLS algorithm has locked on to the true parameter values.

For the next simulation, the parameter values were set to $\beta_0=1/2(\beta_N)$. The resulting data are plotted in Figures 5.22 and 5.23. Inspection of Figure 5.22 shows a large initial overshoot in the response. The subsequent pulses, both positive and negative show no overshoot. Figure 5.23 shows the parameter estimates as they converge. This plot seems to indicate that the parameters may have not completely reached steady-state even after 600 samples. This is much slower convergence than is seen in the Simulink results.

As before, additional simulations were run with other initial parameter values. Figures 5.24 and 5.25 show the output and parameters respectively for initial values set at $\beta_0=2\beta_N$. In these plots, unlike with Simulink, there is no initial overshoot. However, the parameter estimates seem to converge relatively quickly (<150 samples). Figures 5.26 and 5.27 show the results for $\beta_0=4\beta_N$. As Figure 5.26 shows, the same initial instability that was seen in the Simulink model is present, albeit much more pronounced than in Figure 5.13. Figures 5.28 and 5.29 show the response when $\beta_0=-\beta_N$. While there is a spike in the initial response, it does not approach the aberration seen in the Simulink plot of Figure 5.15.

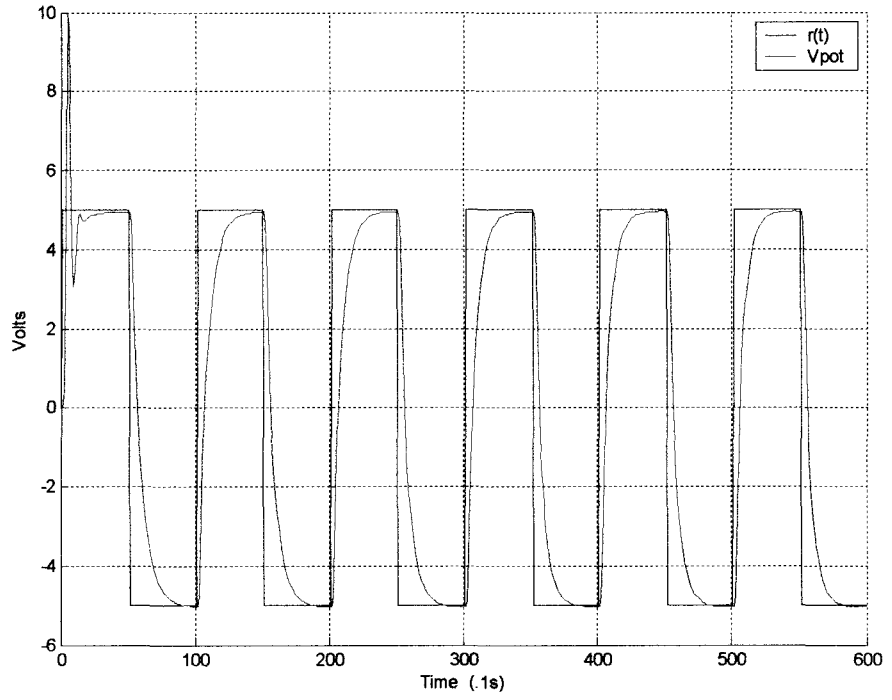


Figure 5.22 GP-6 closed-loop response to a 5V square wave. $\beta_0=1/2\beta_N$.

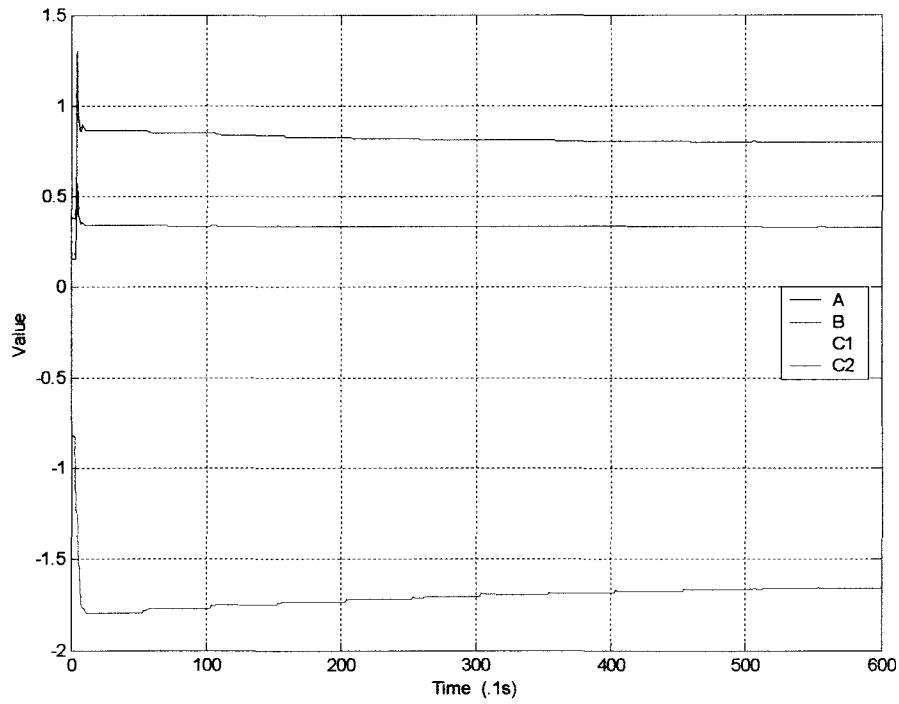


Figure 5.23 GP-6 calculated system parameters. $\beta_0=1/2\beta_N$.

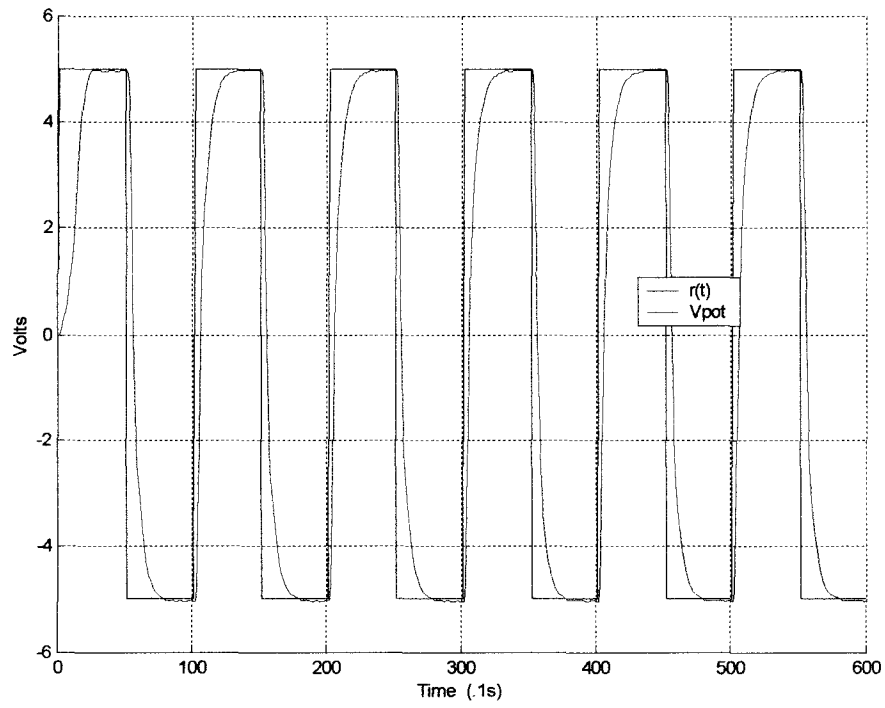


Figure 5.24 GP-6 closed-loop response to a 5V square wave. $\beta_0=2\beta_N$.

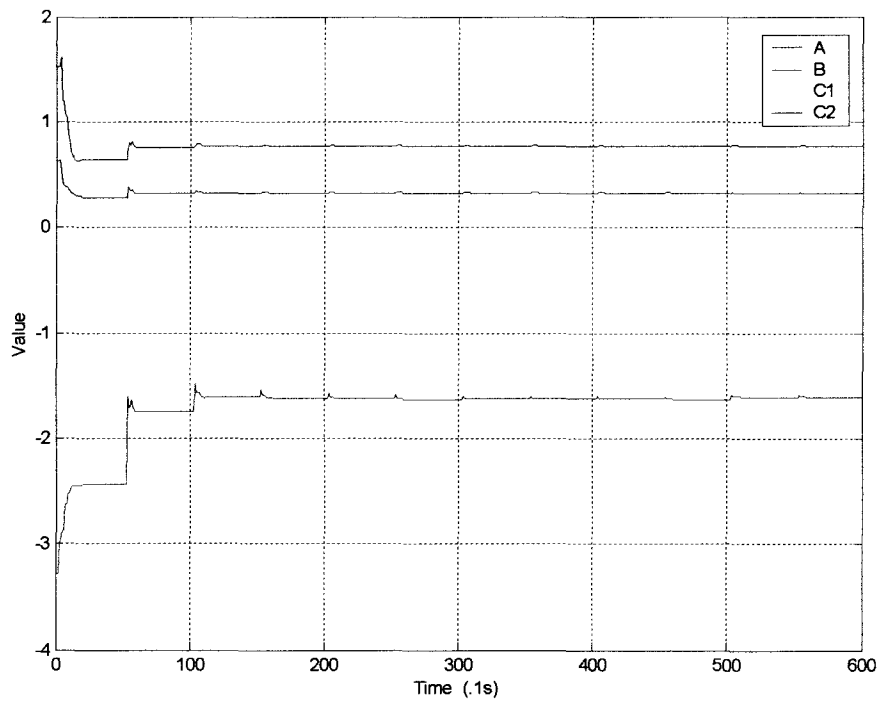


Figure 5.25 GP-6 calculated system parameters. $\beta_0=2\beta_N$.

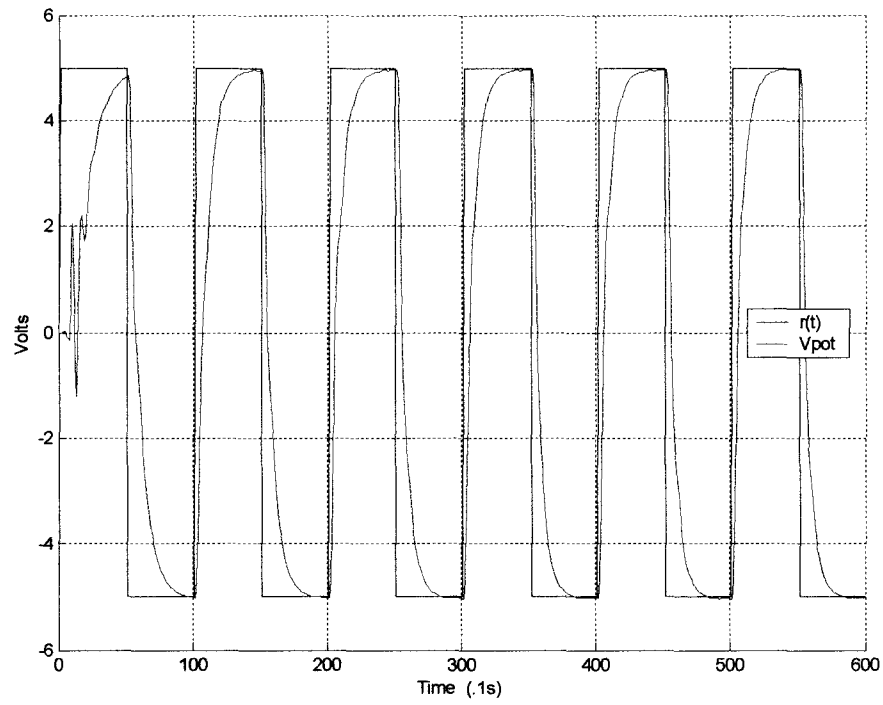


Figure 5.26 GP-6 closed-loop response to a 5V square wave. $\beta_0=4\beta_N$.

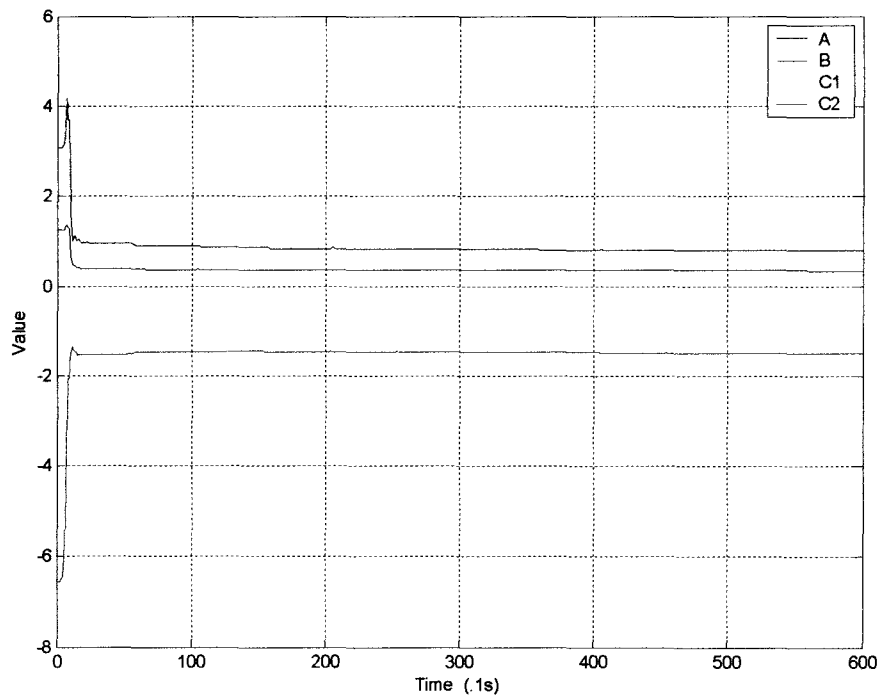


Figure 5.27 GP-6 calculated system parameters. $\beta_0=4\beta_N$.

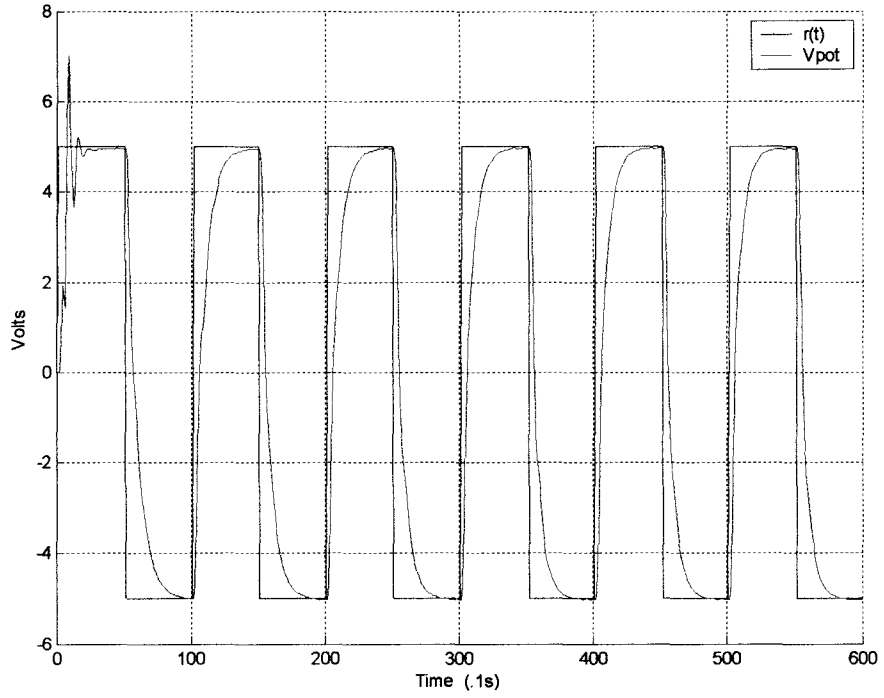


Figure 5.28 GP-6 closed-loop response to a 5V square wave. $\beta_0 = -\beta_N$.

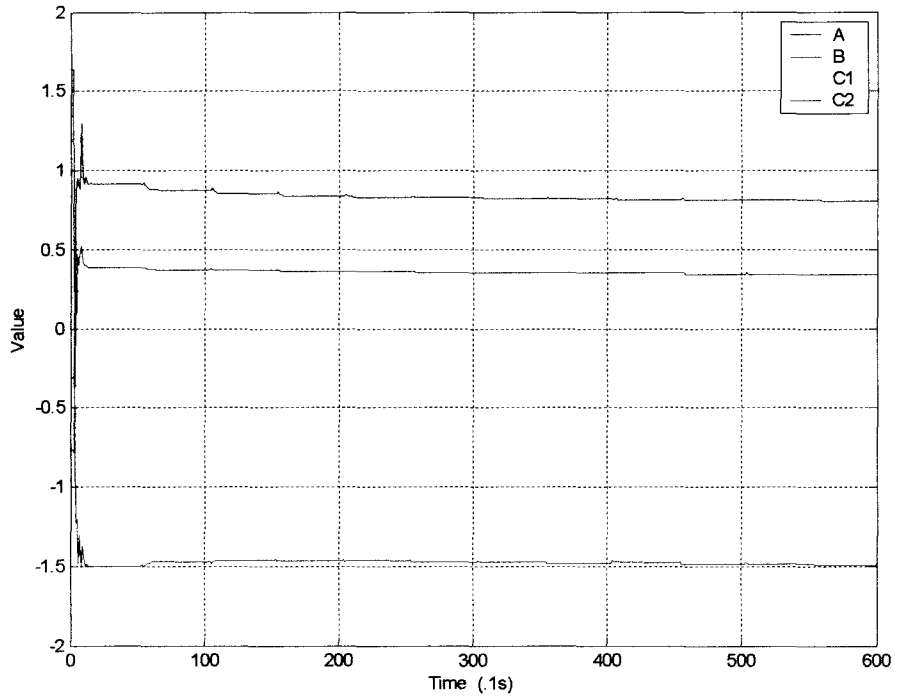


Figure 5.29 GP-6 calculated system parameters. $\beta_0 = -\beta_N$.

An extra experiment was performed on the GP-6 that could not easily be performed on the other two systems. One minute into a two-minute run, one of the GP-6 potentiometers (shown in the circuit of Figure 4.7) was turned 360° . This change in pot setting is equivalent to changing the motor parameters τ_m , K_m , and K_p . The effects are seen in the plots of Figures 5.30 through 5.35. In Figures 5.30 and 5.31 pot 1's value was changed from 0.262 to 0.165. Figures 5.32 and 5.33 correspond to changing pot 2's value from 0.400 to 0.263. Finally, changing pot 3 from 0.600 to 0.485 is shown in the response of Figures 5.34 and 5.35.

In each case, a slight change can be seen between the V_{pot} signal before and after the pot adjustment. This change seems to be an increase in rise time. The parameter estimates also changed when the pot was turned. In all three cases the parameter values started to converge to new values after the one-minute point on the plots.

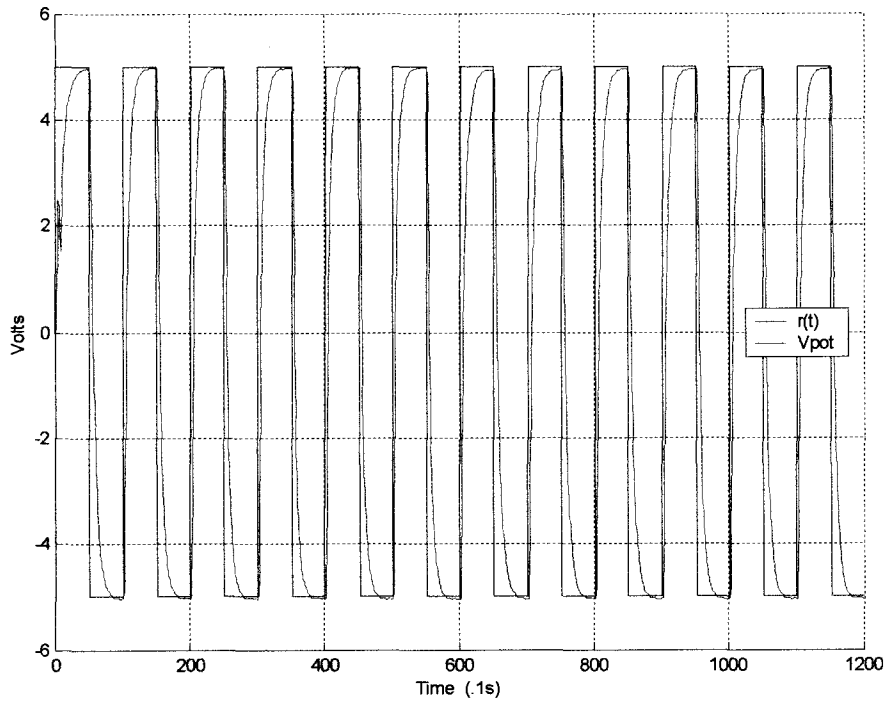


Figure 5.30 GP-6 closed-loop response: Pot 1 varied from 0.262 to 0.165.

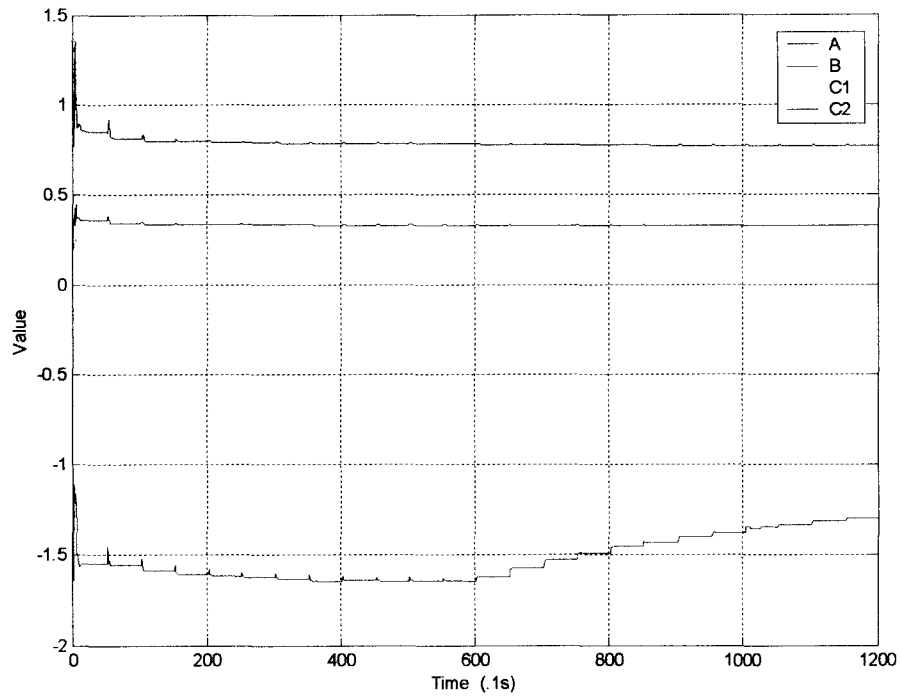


Figure 5.31 GP-6 calculated system parameters: Pot 1 varied from 0.262 to 0.165.

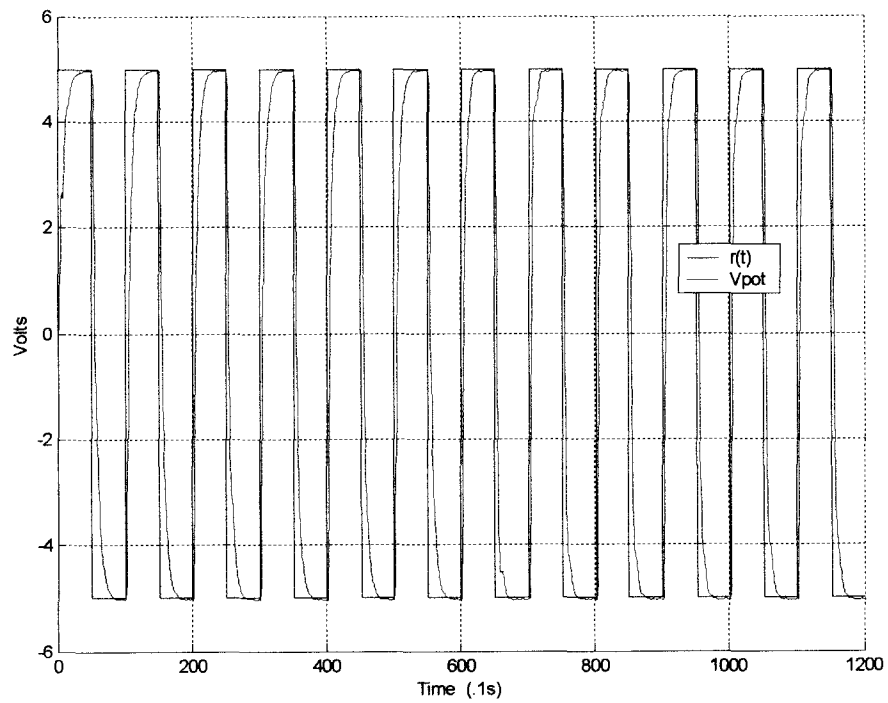


Figure 5.32 GP-6 closed-loop response: Pot 2 varied from 0.400 to 0.263.

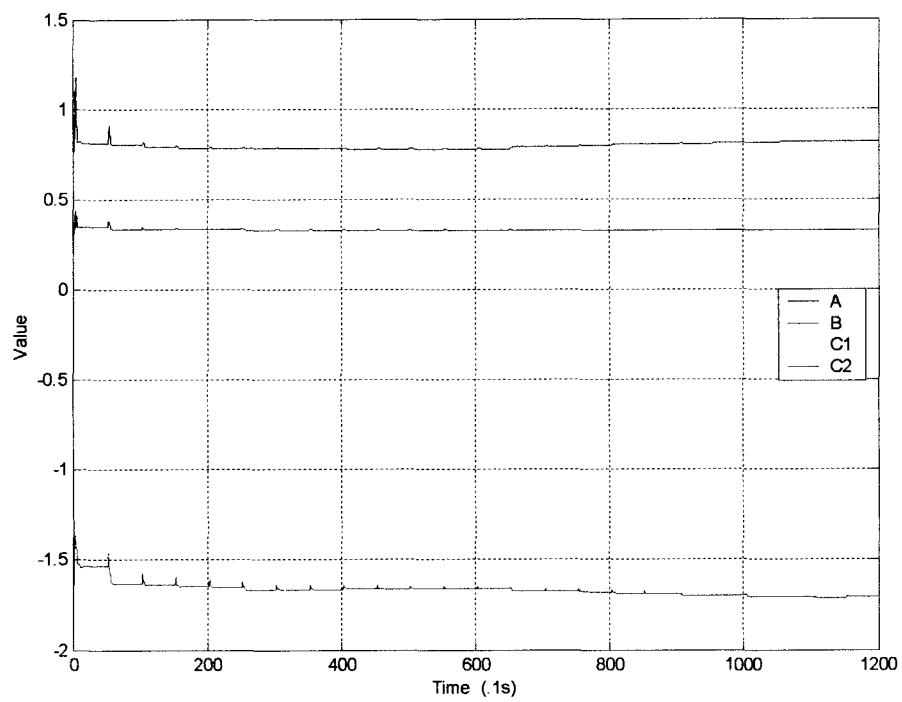


Figure 5.33 GP-6 calculated system parameters: Pot 2 varied from 0.400 to 0.263.

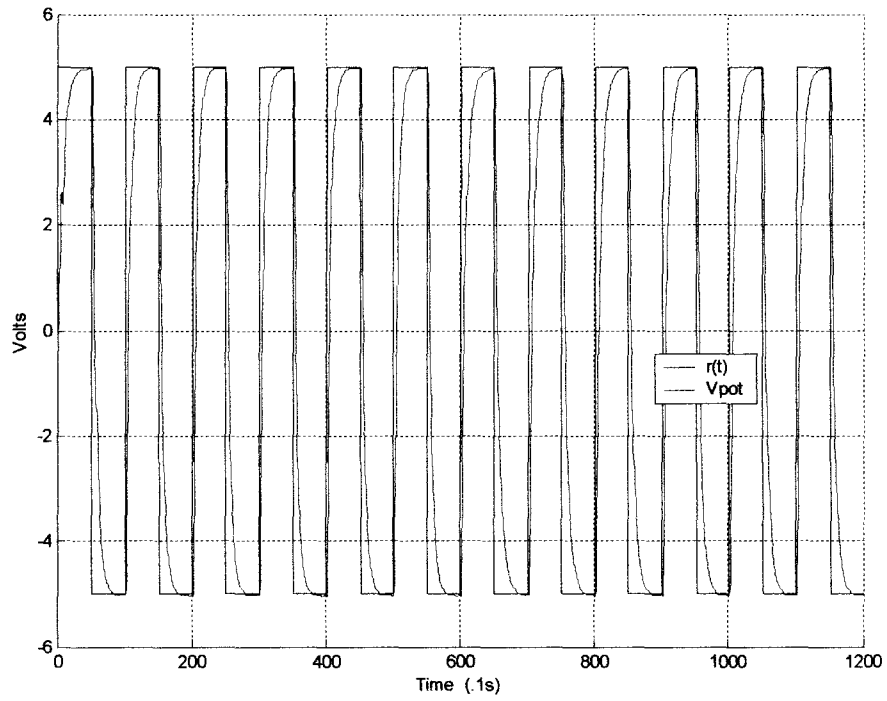


Figure 5.34 GP-6 closed-loop response: Pot 3 varied from 0.600 to 0.485.

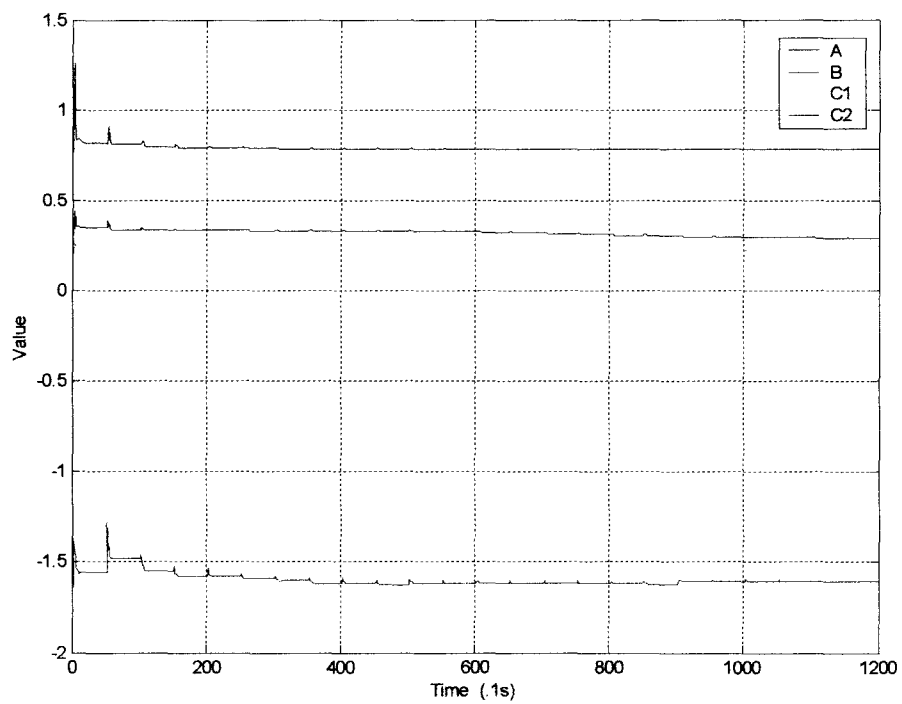


Figure 5.35 GP-6 calculated system parameters: Pot 3 varied from 0.600 to 0.485.

5.3 Controlled Motor System

After the GP-6 simulations, the motor system was wired to the LabVIEW control program. Initially several trial runs were performed to find a nominal set of system parameters. The average parameter values found were $\beta_N = [0.76 \ -1.6 \ 0.36 \ 0.3]^T$. Figures 5.36 through 5.39 show the results for $\beta_0 = \beta_N$. As Figure 5.36 shows, the output is similar to the GP-6 response in Figure 5.17. The actual V_{pot} signal was somewhat more irregular than seen on the GP-6. This was likely due to unmodeled conditions such as inertia, motor brush friction, bearing wear, or torque on the connecting shaft between the motor and potentiometer. Noise in the V_{pot} and V_{tach} signals was another probable cause of signal jitter. There was also approximately 0.2V of steady-state error in response to the +5V step voltage.

The parameters found in Figure 5.37 converged to slightly different values than those found in the GP-6. The nominal system parameters used in the GP-6 equations likely caused this discrepancy. The actual motor's parameters would vary from these assumed values. Like with the GP-6, the observer states were tested by plotting the true system outputs with the estimated system outputs. Figure 5.38 plots these signals. The input u and V_{tach} signals are plotted in Figure 5.39. Both of these figures closely match the results of the GP-6. Again as an additional test, the motor system was allowed to cycle for 6000 samples. The plot in Figure 5.40 shows that some minor parameter variation is seen throughout the run time. This demonstrates that the RLS algorithm has some difficulty exactly locking on to true parameter values, or more likely, the true parameters change slightly while the motor runs.

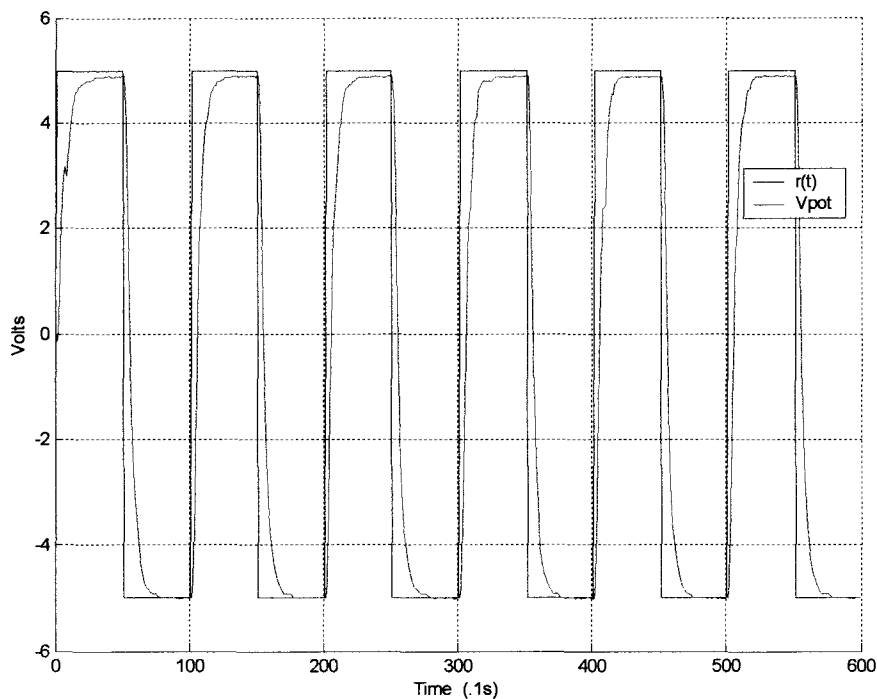


Figure 5.36 Motor closed-loop response to a 5V square wave. $\beta_0 = \beta_N$.

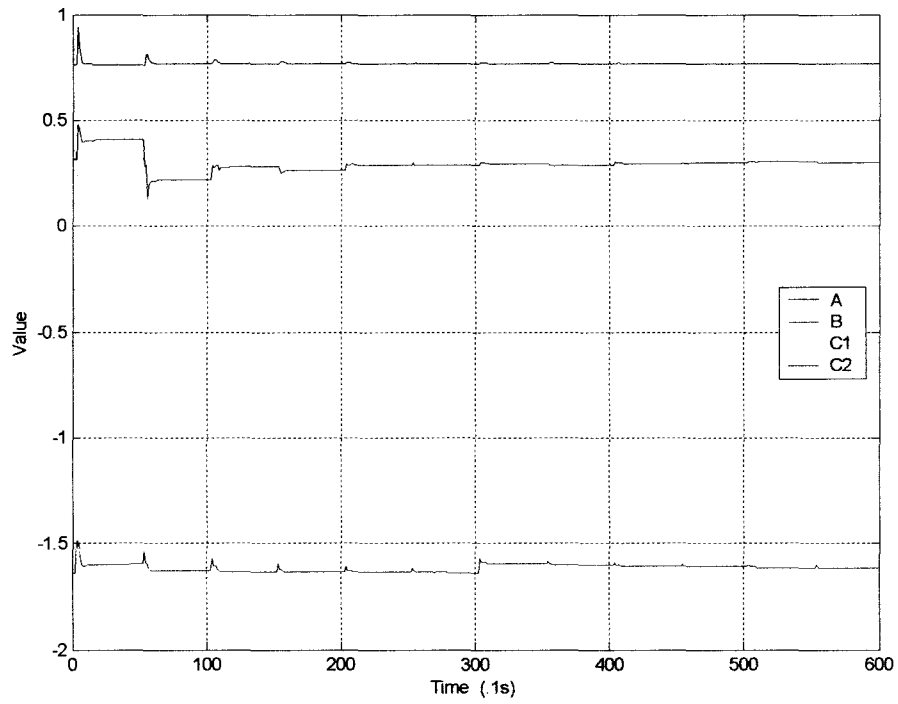


Figure 5.37 Motor calculated system parameters. $\beta_0 = \beta_N$.

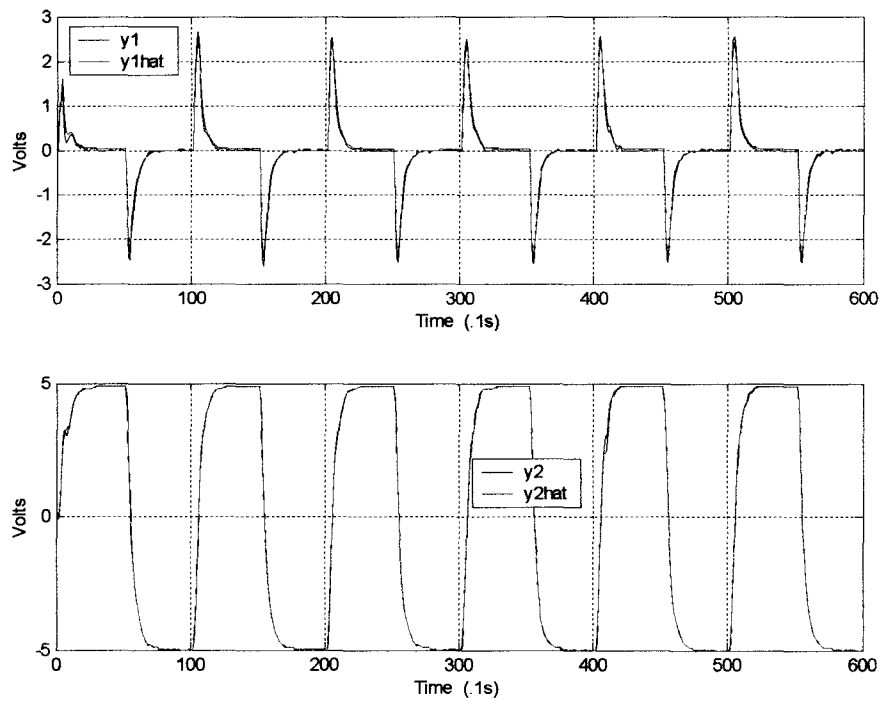


Figure 5.38 Motor closed-loop output and estimated output. $\beta_0 = \beta_N$.

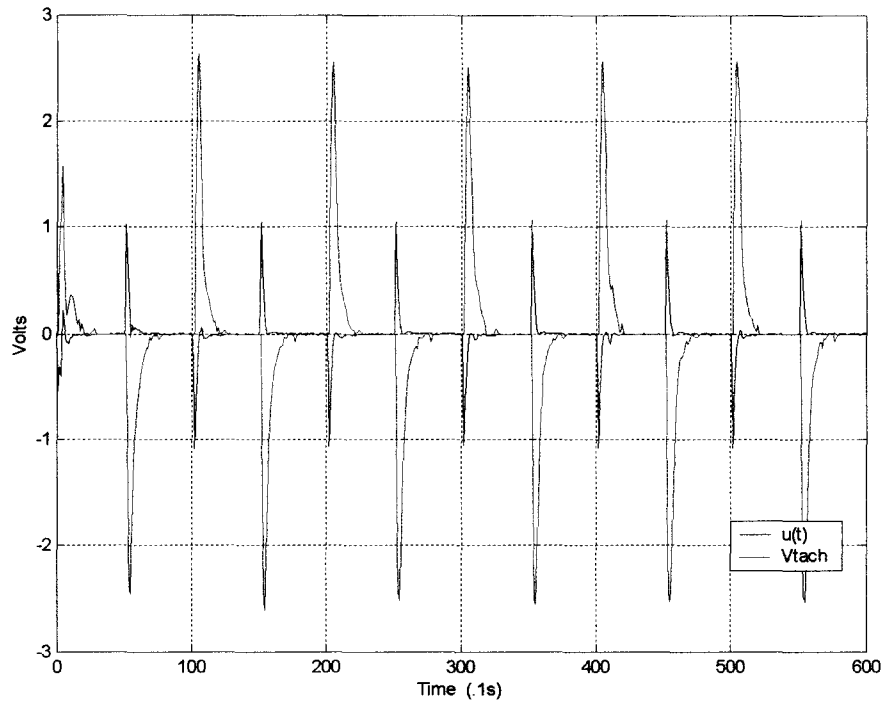


Figure 5.39 Motor input $u(k)$ and tachogenerator output V_{tach} . $\beta_0 = \beta_N$.

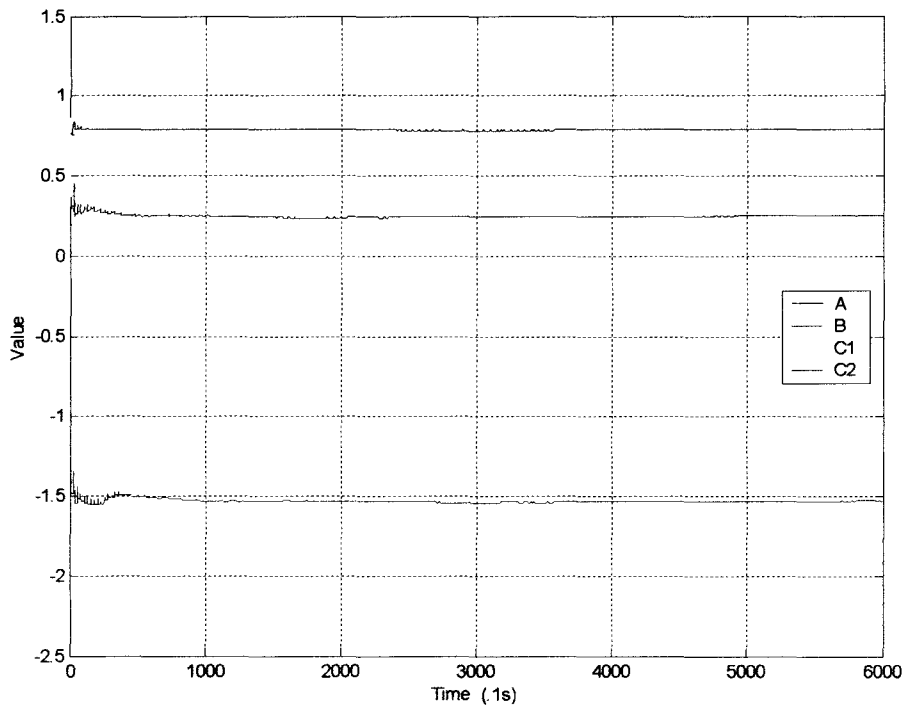


Figure 5.40 Motor calculated system parameters after 10 minutes. $\beta_0 = \beta_N$.

Figures 5.41 and 5.42 show the response when the parameter values were set to $\beta_0=1/2(\beta_N)$. Inspection of Figure 5.41 shows a large initial overshoot in the response. The subsequent pulses, both positive and negative show no overshoot, but demonstrate that the system is having difficulty recovering to a smooth response. Figure 5.42 shows the parameter estimates as they slowly converge toward steady-state. Like the GP-6, the control system converged to the parameter values much more slowly than the Simulink model.

Figures 5.43 and 5.44 show the response for $\beta_0=2\beta_N$. Figures 5.45 to 5.48 show the results for $\beta_0=4\beta_N$. As Figure 5.45 shows, the system was very unstable in response to the first pulse. For the next several pulses the response looks overdamped. It is interesting to see in Figure 5.46 that the parameters settled near the β_N values, but the output response remained bad. The system was allowed to run for a full 10 minutes and the results are in Figures 5.47 and 5.47. Figure 5.47 shows the last minute of the test and demonstrates that the system had recovered to a reasonable response. Figure 5.48 shows that the parameter estimates seemed to drift slightly for the entire run time.

Figures 5.28 and 5.29 show the response when $\beta_0=-\beta_N$.

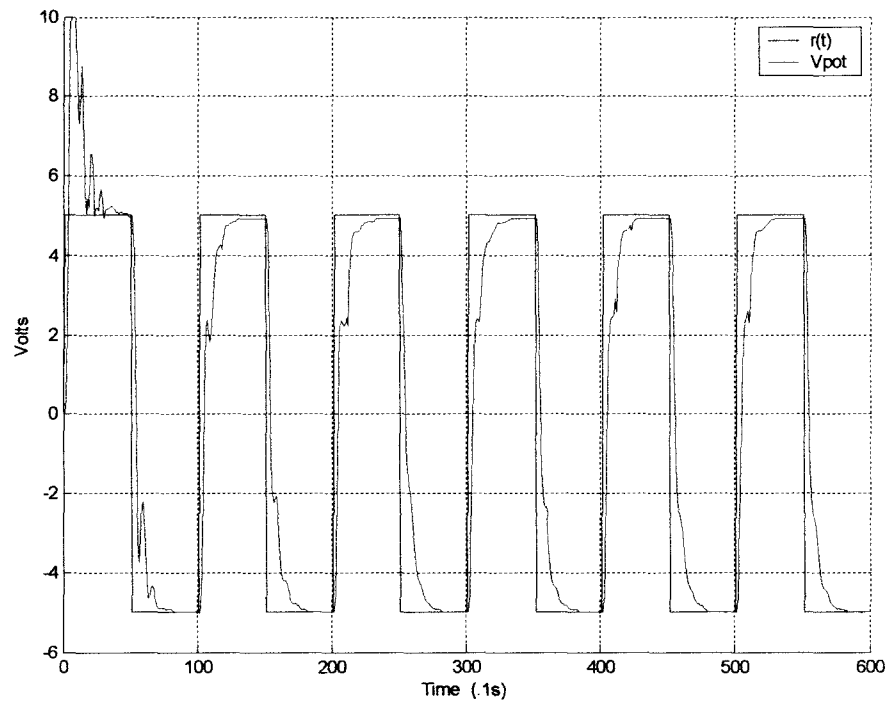


Figure 5.41 Motor closed-loop response to a 5V square wave. $\beta_0 = 1/2\beta_N$.

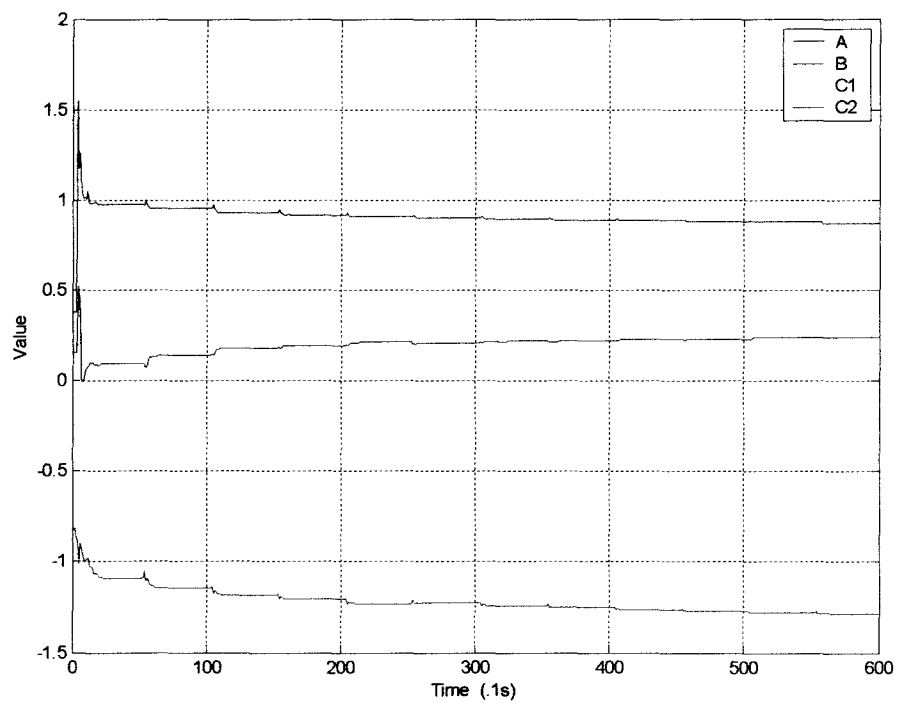


Figure 5.42 Motor calculated system parameters. $\beta_0 = 1/2\beta_N$.

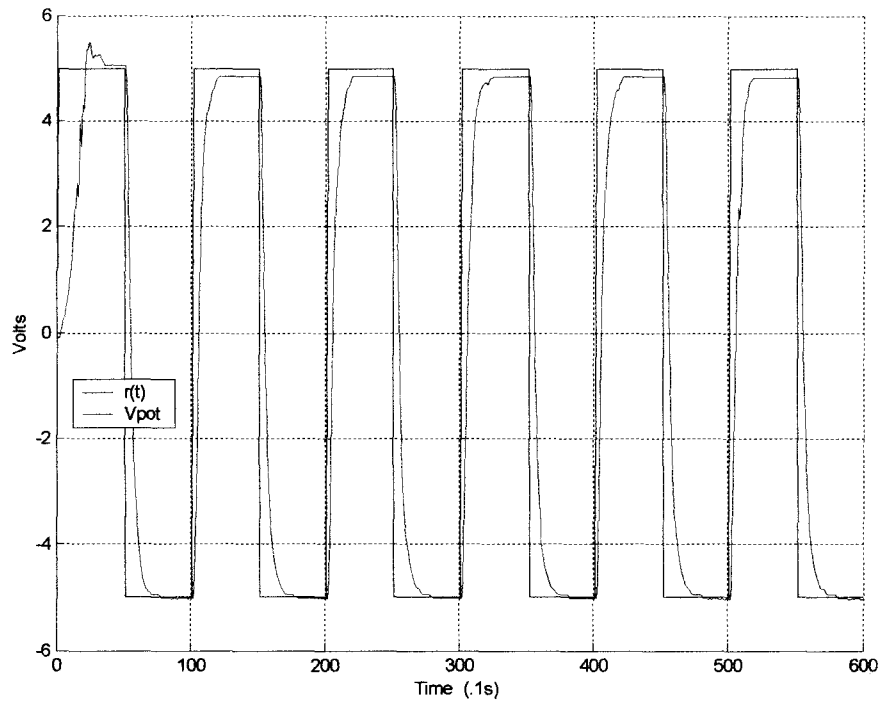


Figure 5.43 Motor closed-loop response to a 5V square wave. $\beta_0=2\beta_N$.

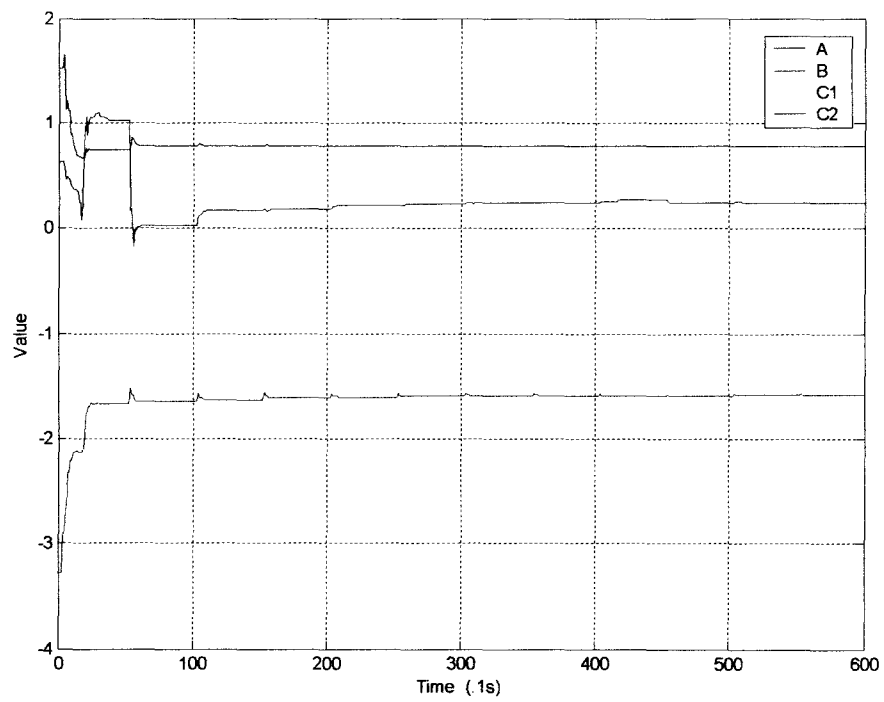


Figure 5.44 Motor calculated system parameters. $\beta_0=2\beta_N$.

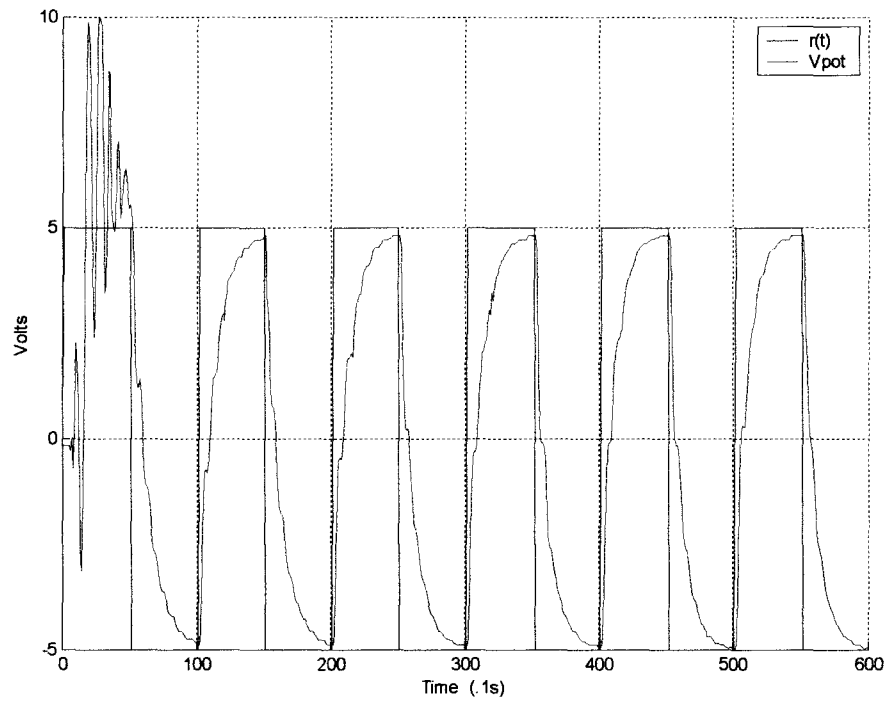


Figure 5.45 Motor closed-loop response to a 5V square wave. $\beta_0=4\beta_N$.

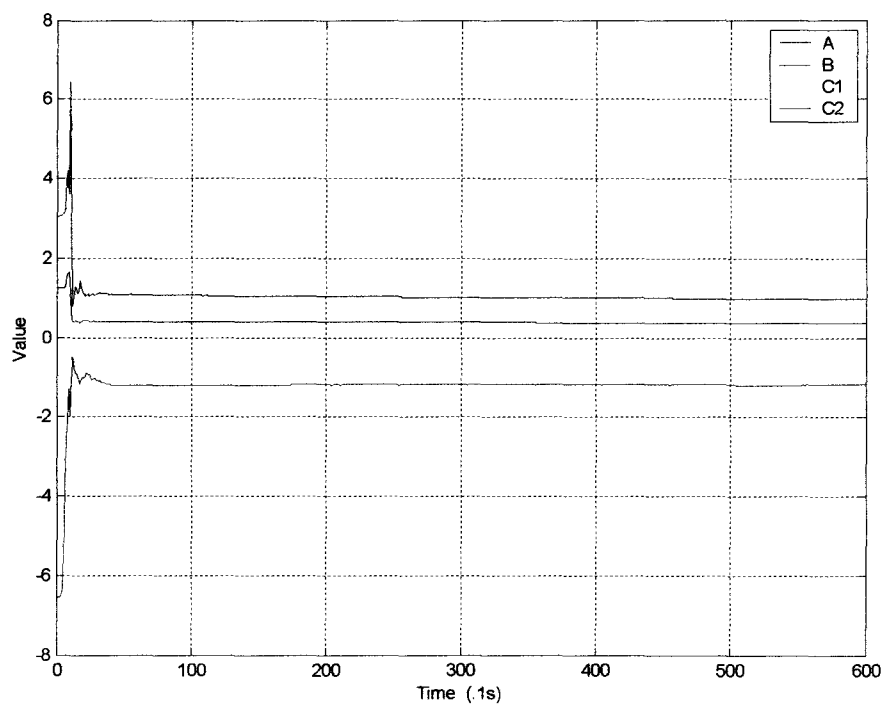


Figure 5.46 Motor calculated system parameters. $\beta_0=4\beta_N$.

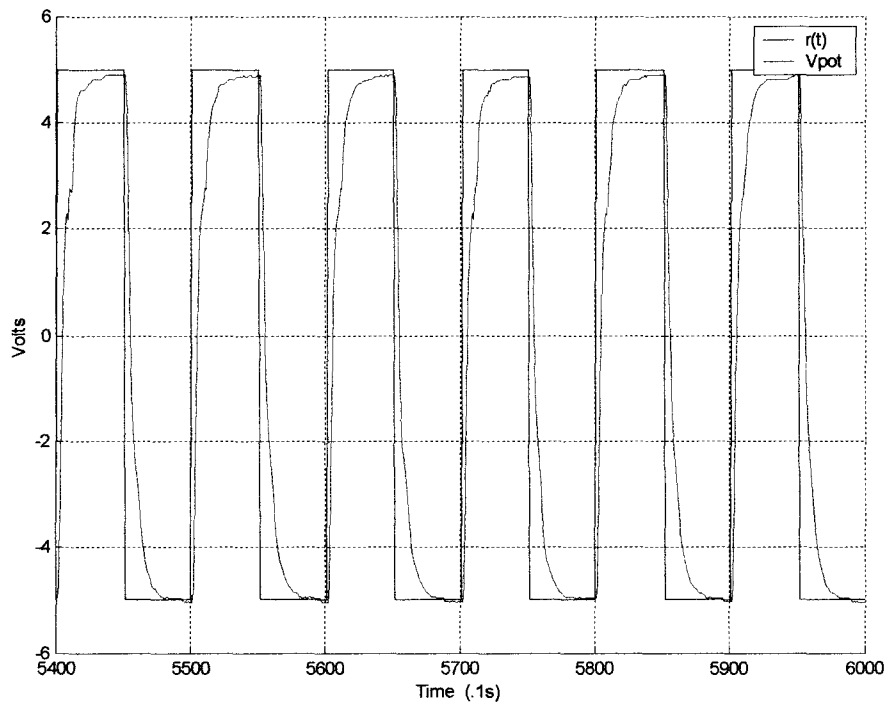


Figure 5.47 Motor closed-loop response (last 600 samples of 6000). $\beta_0=4\beta_N$.

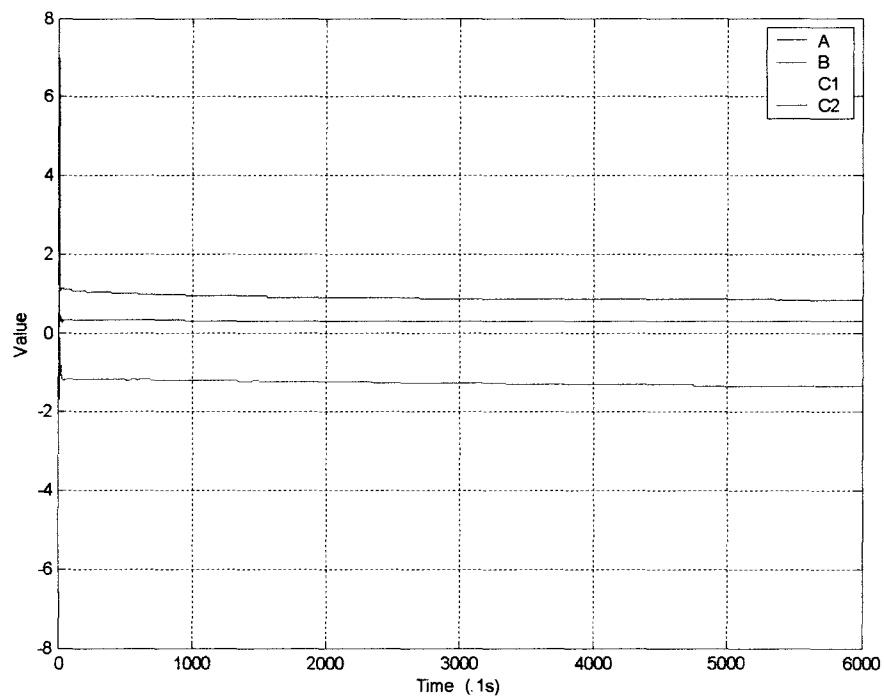


Figure 5.48 Motor calculated system parameters after 10 minutes. $\beta_0=4\beta_N$.

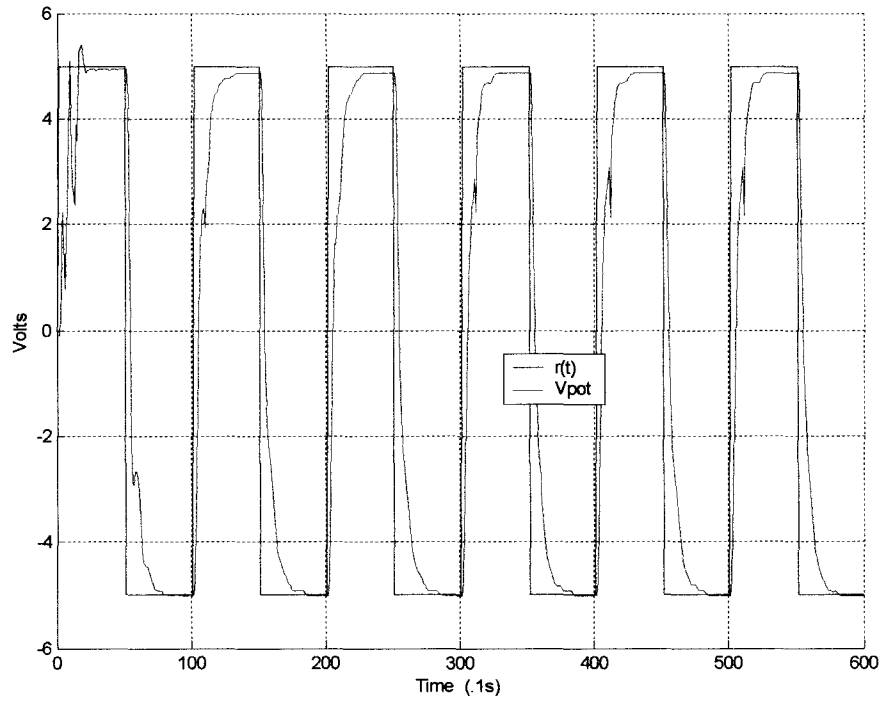


Figure 5.49 Motor closed-loop response to a 5V square wave. $\beta_0 = -\beta_N$.

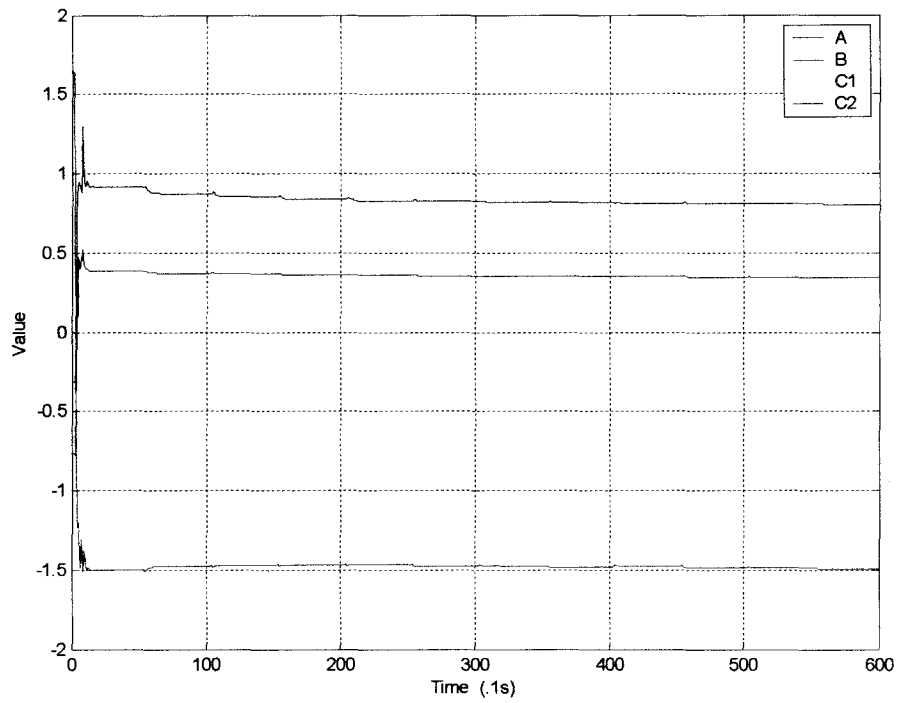


Figure 5.50 Motor calculated system parameters. $\beta_0 = -\beta_N$.

Table 5.1 lists the final parameter estimates, β_{FINAL} , for each trial of the three systems. The initial values, β_0 , are also given for reference. Table 5.2 lists the average β_{FINAL} for each system.

Table 5.1 Parameter Estimate Data

		A	B	C ₁	C ₂	A	B	C ₁	C ₂
Simulink	$\beta_0 = \beta_N$	0.67	-2.14	0.32	0.28	0.67	-2.14	0.32	0.28
	$\beta_0 = 1/2\beta_N$	0.335	-1.07	0.16	0.14	0.67	-2.14	0.32	0.28
	$\beta_0 = 2\beta_N$	1.34	-4.28	0.64	0.56	0.67	-2.14	0.32	0.28
	$\beta_0 = 4\beta_N$	2.68	-8.56	1.28	1.12	0.67	-2.14	0.32	0.28
	$\beta_0 = -\beta_N$	-0.67	2.14	-0.32	-0.28	0.67	-2.14	0.32	0.28
GP-6	$\beta_0 = \beta_N$	0.67	-2.14	0.32	0.28	0.77	-1.61	0.27	0.33
	$\beta_0 = 1/2\beta_N$	0.385	-0.805	0.135	0.165	0.78	-1.66	0.28	0.33
	$\beta_0 = 2\beta_N$	1.54	-3.22	0.54	0.66	0.77	-1.6	0.28	0.33
	$\beta_0 = 4\beta_N$	3.08	-6.44	1.08	1.32	0.81	-1.5	0.27	0.34
	$\beta_0 = -\beta_N$	-0.77	1.61	-0.27	-0.33	0.81	-1.49	0.27	0.35
Motor	$\beta_0 = \beta_N$	0.76	-1.6	0.36	0.3	0.76	-1.62	0.39	0.27
	$\beta_0 = 1/2\beta_N$	0.38	-0.8	0.18	0.15	0.87	-1.29	0.37	0.24
	$\beta_0 = 2\beta_N$	1.52	-3.2	0.72	0.6	0.78	-1.58	0.41	0.25
	$\beta_0 = 4\beta_N$	3.04	-6.4	1.44	1.2	0.85	-1.35	0.38	0.3
	$\beta_0 = -\beta_N$	-0.76	1.6	-0.36	-0.3	0.81	-1.39	0.36	0.32

Table 5.2 Average Final Parameter Estimates

	A	B	C ₁	C ₂
Simulink	0.67	-2.14	0.32	0.28
GP-6	0.79	-1.57	0.27	0.34
Motor	0.81	-1.45	0.38	0.28

CHAPTER VI

SUMMARY AND CONCLUSION

This thesis investigated an adaptive state-variable feedback control system. As introduced in Chapter I, adaptive control uses signal synthesis or parameter estimation to adapt to the system being controlled or to changes in system characteristics. Modern control software packages like LabVIEW and MATLAB in conjunction with data acquisition hardware can implement an adaptive system. The effectiveness of such a system was studied here.

In Chapter II, the real-world plant to be controlled was discussed. The modular servomotor system was configured with high gain and a defined time constant to behave like a linear system. The continuous-time model was presented and this was used to derive the discrete transfer functions between the input and tachogenerator and between the input and position potentiometer output. The discrete state-variable model was then developed and the open-loop response to a step function was shown.

A discrete state-variable feedback controller was developed in Chapter III. Since the system states were not physically measurable, an observer was introduced to estimate the states. Next, the Recursive Least-Squares algorithm for adaptive control was outlined.

Chapter IV introduced the hardware and software used in the control implementation. The NI 6024E multifunction I/O card was employed for data acquisition. A Simulink model was developed using a custom S-Function block to perform the RLS algorithm. LabVIEW and MATLAB were then integrated in the real-world control system. The LabVIEW virtual instrument was developed using MATLAB scripts for control calculations. A GP-6 test system was then constructed to define an expectation of motor-system performance.

Finally, in Chapter V the systems were simulated with varying initial parameter estimates and the results were plotted. The Simulink model adapted in all cases within the first 5V input pulse. After the first pulse, the transient response was smooth with no overshoot in all the trials. The GP-6 and motor systems showed somewhat different behavior. The adaptation (seen in the system parameter convergence plots) took longer for these two systems than for Simulink. This generally led to a less smooth transient response. The motor system seemed to have difficulty recovering from initial instability. Even after the system parameters had converged, the response remained visibly jittery.

This investigation leads to several conclusions. The motor system has characteristics that are not accounted for in the system model. Brush friction, coupling shaft torque, and noise are likely factors in the response that are not modeled in the system equations. These unmodeled properties seem to cause a continuation of any

initial instability through many subsequent reference pulses. For instance, an underdamped response to the initial pulse seems to be followed by what appear to be noisy responses for many subsequent pulses. In both the Simulink and GP-6 systems the initial underdamped response is gone by the second pulse of the input signal.

Conclusions about the control implementation can also be made. The LabVIEW program is not well suited for mathematical computations, especially matrix algebra. Building a block diagram to implement the RLS algorithm would have been time-consuming and would have been extremely complicated to troubleshoot. It was much easier to insert a MATLAB script to do the control calculations. The main disadvantage of this is that the MATLAB software must be purchased in addition to LabVIEW to use this feature. Another problem with LabVIEW is that the real-time plotting functions take significant computing power. This steals processor time from the control program and slows down the maximum sample rate that can be achieved. The data plotting capabilities of LabVIEW are also limited. To plot data for presentation, LabVIEW must write the data to a spreadsheet file and the spreadsheet software (also an additional cost) must create the plots.

Overall, the adaptive control implementation worked reasonably well on the GP-6 system and on the motor system. The control objective of position control with a fast rise time and no overshoot was met. Also, no steady-state error to the reference input was seen in the GP-6 system and only slight steady-state error (-0.2V) was seen for the motor system. Correcting this steady-state error and reducing noise in the motor response would be a starting point for further research. Other ideas for further study include implementing the controller in a Linux environment [6]. Since Windows is a multitasking operating system, the control program must share processor time with other background applications. Linux provides the ability to prioritize tasks so that the control program can monopolize the processor time to achieve real time control. Much higher sample rates could be achieved using Linux.

The LabVIEW program itself is also a good place for more study. Attempting the adaptive control scheme using only LabVIEW functions and subVIs could demonstrate that cost-savings could be achieved in this system. Another area for investigation would be to explore the other features of the 6024E DAQ card. This card has thermocouple inputs that could be employed for an adaptive temperature-control system. A buffered data acquisition scheme could be used, and the discrete I/O card functions could be investigated.

APPENDIX

PROGRAM LISTINGS

Appendix Sections A.1 and A.2 list the program files used in the Simulink system model. The initialization routine used to set up variables in the MATLAB workspace is listed in Section A.1. This routine had to be executed before the Simulink model was simulated. The sections of the Simulink S-Function template that were modified to implement the RLS algorithm are listed in Section A.2. The other sections of the S-Function template were left unedited.

A.1 Simulink Initialization m-file: rls2init.m

```
% Adaptive motor control - Initialization
%
% Create global parameters and plot variables
global P1;
global beta1;
global P2;
global beta2;
global beta1A;
global beta1B;
global beta2C1;
global beta2C2;
global ii;
global tplot;
ii=0;
tplot=[];
beta1A=[];
beta1B=[];
beta2C1=[];
beta2C2=[];
%
% simulation parameters
T=0.1;
%
% system parameters
Km=6.5;
tm=.25;
Kpt=6;
A=exp(-T/tm);
B=-Km*(1-exp(-T/tm));
C1=Kpt*(T-tm+tm*exp(-T/tm))/(1-exp(-T/tm));
C2=Kpt*(tm-tm*exp(-T/tm)-T*exp(-T/tm))/(1-exp(-T/tm));
%
% initial values
P1=[10 0;0 10];
%beta1=[A;B];
beta1=[-.763;1.64];
P2=[10 0;0 10];
%beta2=[C1;C2];
```

```

beta2=[-.364;-.311];
ii=0;
Ltrack(1)=0;
%
% control & observer poles
% calculate closed-loop poles
a1=-4+j;
a2=-4-j;
p1=exp(a1*T);
p2=exp(a2*T);
p=[p1 p2];
%
% calculate observer poles
a3=-9;
a4=-10;
p3=exp(a3*T);
p4=exp(a4*T);
pp=[p3 p4];
%
% controller parameters
AA=[A 0;B 1];
BB=[1;0];
CC=[B 0;C1*B C1+C2];
DD=[0;0];
Kp=place(AA,BB,p);
%
% Observer parameters
L=place(AA',CC',pp)';
Aobs=[AA-L*CC];
Bobs=[BB L];
C11=[CC;1 0;0 1];
D11=[0 0 0;0 0 0;0 0 0;0 0 0];
D2=[0;0;0;0];
%
% solve for initial Nx and Nu
M=[[AA-eye(2)] BB; CC [0;0]];
R=[0;0;0;1];
N=MR;
Nx=N(1:2);
Nu=N(3);

```

A.2 Simulink S-Function (Modified Template Sections)

```

function [sys,x0,str,ts] = RLScalc(t,x,u,flag)
%RLScalc M-file S-function
% This is an s-file subsystem with no states. It uses the Recursive
% Least Squares algorithm to calculate the system parameters of the
% servo-motor system.
%
%*****
%=====
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=====
%
function [sys,x0,str,ts]=mdlInitializeSizes

```

```

%
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
%
% Note that in this example, the values are hard coded. This is not a
% recommended practice as the characteristics of the block are typically
% defined by the S-function parameters.
%
sizes = simsizes;

sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs    = 12;
sizes.NumInputs     = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0 = [];

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts = [0 0];

% end mdlInitializeSizes
%
%=====
% mdlOutputs
% Return the block outputs.
%=====
%
function sys=mdlOutputs(t,x,u)

% Set up global variables and constants. Run the file rls2init.m before running
% the Simulink simulation of RLS_trial.mdl
global P1;
global beta1;
global P2;
global beta2;
global beta1A;
global beta1B;
global beta2C1;
global beta2C2;
global ii;
global tplot;
ii=ii+1;

```

```

tplot(ii)=ii;
%
% system parameters
lambda=.9;
%
% Give the inputs recognizable names.
y1=u(1);
y1old=u(2);
uold=u(3);
y2=u(4);
y2old=u(5);
%
% Perform the RLS algorithm. beta1=[A; B]. beta2=[C1; C2].
psi1=[y1old;uold];
K1=P1*psi1/(lambda+psi1'*P1*psi1);
P1=(eye(2)-K1*psi1')*P1/lambda;
beta1=beta1+K1*(y1-psi1'*beta1);
beta1A(ii)=beta1(1);
beta1B(ii)=beta1(2);

psi2=[y1;y1old];
K2=P2*psi2/(lambda+psi2'*P2*psi2);
P2=(eye(2)-K2*psi2')*P2/lambda;
beta2=beta2+K2*((y2-y2old)-psi2'*beta2);
beta2C1(ii)=beta2(1);
beta2C2(ii)=beta2(2);
%
% update controller parameters
Arls=[beta1(1) 0;beta1(2) 1];
Brls=[1;0];
Crls=[beta1(2) 0; beta2(1)*beta1(2) beta2(1)+beta2(2)];
a1=-4+2*j;
a2=-4-2*j;
p1=exp(a1*.1);
p2=exp(a2*.1);
p=[p1 p2];
Kp=place(Arls,Brls,p);

a3=-9;
a4=-10;
p3=exp(a3*.1);
p4=exp(a4*.1);
pp=[p3 p4];
L=place(Arls',Crls',pp)';
Aobs=Arls-L*Crls;
%
%Solve for Nx and Nu
M1=[[Arls-eye(2)] Brls; Crls [0;0]];
R1=[0;0;0;1];
N1=M1\R1;
Nx=N1(1:2);
%
% send output to Simulink workspace
sys=[Aobs(1) Aobs(2) Aobs(3) Aobs(4) L(1) L(2) L(3) L(4) Kp(1) Kp(2) Nx(1) Nx(2)];
%
% end mdlOutputs

```

Appendix Sections B.1 and B.2 list the program files embedded in the LabVIEW control program. The initialization routine used to set up variables in the MATLAB workspace is listed in Section B.1. The loop routine that performed the control and RLS calculations is listed in Section B.2.

B.1 LabVIEW: MATLAB Initialization Script

```
% Adaptive motor control - Initialization
%
% simulation parameters
T=0.1; Tp=5; Tfnl=8*Tp; time=0:T:Tfnl;
refamp=5; ref=refamp*sign(sin(2*pi/Tp*time));
%
% system parameters
lambda=1;
Km=6.5;
tm=.25;
Kpt=6;
A=exp(-T/tm);
B=-Km*(1-exp(-T/tm));
C1=Kpt*(T-tm+tm*exp(-T/tm))/(1-exp(-T/tm));
C2=Kpt*(tm-tm*exp(-T/tm)-T*exp(-T/tm))/(1-exp(-T/tm));
%
% initial values
x1hat=0; x2hat=0; uold=0; y1old=0; y2old=0;
y1=0; y2=0; indx=1;
P1=[10 0;0 10];
beta1=[A;B];
P2=[10 0;0 10];
beta2=[C1;C2];
ii=0;
Ltrack(1)=0;
%
% control & observer poles
% calculate closed-loop poles
a1=-4+1*j;
a2=-4-1*j;
p1=exp(a1*T);
p2=exp(a2*T);
p=[p1 p2];
%calculate observer poles
a3=-9;
a4=-10;
p3=exp(a3*T);
p4=exp(a4*T);
pp=[p3 p4];
%
% controller parameters
AA=[A 0;B 1];
BB=[1;0];
CC=[B 0;C1*B C1+C2];
DD=[0;0];
Kp=place(AA,BB,p);
Kp1=Kp(1);
Kp2=Kp(2);
```

```

Arls=[beta1(1) 0;beta1(2) 1];
Brls=[1;0];
Crls=[beta1(2) 0;beta2(1)*beta1(2) beta2(1)+beta2(2)];
L=place(Arls',Crls',pp)';
L1=L;
Aobs=[Arls-L1*Crls];
%solve for initial Nx and Nu
M=[[AA-eye(2)] BB; CC [0;0]];
R=[0;0;0;1];
N=MR;
Nx1=N(1);
Nx2=N(2);

```

B.2 LabVIEW: MATLAB Loop Script

```

% Adaptive motor control - Loop Script
%
% calculate control output
u=Kp1*(Nx1*ref(indx)-x1hat)+Kp2*(Nx2*ref(indx)-x2hat);
%
% save data for plotting
ii=ii+1;
P11plot(ii)=P1(1);
P12plot(ii)=P1(2);
P13plot(ii)=P1(3);
P14plot(ii)=P1(4);
tplot(ii)=ii;
beta1A(ii)=beta1(1);
beta1B(ii)=beta1(2);
beta2C1(ii)=beta2(1);
beta2C2(ii)=beta2(2);
y1old1(ii)=y1old;
uold1(ii)=uold;
uplot(ii)=u;
vtachplot(ii)=y1;
vpotplot(ii)=y2;
%
% calculate next states
nx1hat=Aobs(1)*x1hat+Aobs(3)*x2hat+u+L1(1)*y1+L1(3)*y2;
nx2hat=Aobs(2)*x1hat+Aobs(4)*x2hat+L1(2)*y1+L1(4)*y2;
%
% update parameter estimates
psi1=[y1old;uold];
K1=P1*psi1/(lambda+psi1'*P1*psi1);
P1=(eye(2)-K1*psi1')*P1/lambda;
beta1=beta1+K1*(y1-psi1'*beta1);

psi2=[y1;y1old];
K2=P2*psi2/(lambda+psi2'*P2*psi2);
P2=(eye(2)-K2*psi2')*P2/lambda;
beta2=beta2+K2*((y2-y2old)-psi2'*beta2);

uold=u;
y1old=y1;
y2old=y2;
%

```



```
% update controller parameters
Arls=[beta1(1) 0;beta1(2) 1];
Brls=[1;0];
Crls=[beta1(2) 0;beta2(1)*beta1(2) beta2(1)+beta2(2)];
Kp=place(Arls,Brls,p);
Kp1=Kp(1);
Kp2=Kp(2);
L=place(Arls',Crls',pp)';
L1=L;
if abs(L(1))>1
    L1(1)=-.4;
end;
Aobs=[Arls-L1*Crls];

%%%%Solve for Nx and Nu
M1=[[Arls-eye(2)] Brls; Crls [0;0]];
R1=[0;0;0;1];
N1=M1\R1;
Nx1=N1(1);
Nx2=N1(2);
%
% update states
x1hat=nx1hat;
x2hat=nx2hat;
```

REFERENCES

- [1] M. Gupta, editor, *Adaptive Methods for Control System Design*. New York:IEEE Press, 1986.
- [2] *Modular Servo Type MS 150 Technical Information*. Crowborough, England: Feedback Ltd.
- [3] G.F. Franklin, J.D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 2nd ed. Reading, MA: Addison Wesley, 1990.
- [4] J. S. Bay, *Fundamentals of Linear State Space Systems*. Boston: WCB/McGraw-Hill, 1999.
- [5] K.J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 1997.
- [6] Y.-C. Chen and J. Naughton, "An Undergraduate Laboratory Platform for Control System Design, Simulation, and Implementation", *IEEE Control Systems Magazine*, pp. 12-20, June 2000.
- [7] *DAQ: PCI-6023E/6024E/6025E User Manual*. Austin, TX: National Instruments, October, 1998.
- [8] *National Instruments LabVIEW User Manual*. Austin, TX: National Instruments, November, 2001.
- [9] D. Hanselman and B. Littlefield, *Mastering MATLAB 6: A Comprehensive Tutorial and Reference*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [10] J.B. Dabney and T.L. Harman, *Mastering Simulink 4*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [11] LabVIEW 6i Start-up Screen> *Search Examples> I/O Interfaces> Data Acquisition (DAQ)> Simultaneous Analog I/O> Analog I/O Control Loop (Hardware-Timed)*. Austin, TX: National Instruments, July, 2000.