

24417

AN EXPERIMENTAL INVESTIGATION OF ADAPTIVE SERVO CONTROL
OF A SINGLE AXIS TABLE

by
F. İlkin Özel
B.S. in M.E., Boğaziçi university, 1990

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Mechanical Engineering

Boğaziçi University
1992

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ


AN EXPERIMENTAL INVESTIGATION OF ADAPTIVE SERVO CONTROL
OF A SINGLE AXIS TABLE

APPROVED BY

Doç. Dr. Osman TÜRKAY
(Thesis Supervisor)



Prof. Dr. Okyay KAYNAK



Prof. Dr. Akın TEZEL



DATE OF APPROVAL : June 23, 1992

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to Assoc. Prof. Dr. Osman Türkay, who has supervised my thesis and given me extensive support throughout my study.

I would like to thank to Prof. Dr. Okyay Kaynak and his assistant Mr. İlker Tunay for their help in the implementation of the electronic hardware.

I would also like to thank Prof. Dr. Akın Tezel for his review.

And I am grateful to Research Assistant U. Hakan Eres for his invaluable help whenever I needed.

ABSTRACT

The object of this thesis is to model a dc motor driven single axis table system and to perform adaptive velocity and position control of the table.

After modeling the system, in the first part of the thesis, fixed parameter and adaptive PI control of the velocity of the table has been achieved and the effects of several effective parameters in adaptive control such as initial estimates, initial covariance matrix and forgetting factor have been investigated. Recursive least squares estimation method has been used to estimate the system parameters at each step. The experimental results have shown that for speed control of a dc motor driven single axis table system, implementing adaptive control algorithm is better solution.

In the second part of the thesis, the ordinary PD and adaptive pole assignment position control of the table were performed. The effect of estimator parameters on the estimator and controller response has been examined. The results revealed that the adaptive regulator is advantageous over the ordinary PD pole assignment algorithm.

KISA ÖZET

Bu tez çalışmasının amacı dc motor tahrikli tek eksenli bir tabla sisteminin modellenmesi ve tablanın hız ve konum denetiminin kendi-ayarlamalı uyarlanabilir denetim yöntemi ile sağlanmasıdır.

Sistem modellendikten sonra, tezin ilk bölümünde, sabit parametrelili ve kendi-ayarlamalı uyarlanabilir PI hız denetimleri gerçekleştirilmiş ve uyarlanabilir denetimdeki ilk kestirmeler, unutma faktörü ve başlangıç kovaryans matrisi gibi bazı etkin parametrelerin etkisi araştırılmıştır. Sistem parametrelerinin her adımda kestirilmesinde ardışık en küçük kareler parametre kestirme yöntemi kullanılmıştır. Deneysel sonuçlar, bir dc motor tahrikli tek eksenli tabla sisteminin hız denetiminin sağlanmasında, kendi-ayarlamalı uyarlanabilir denetim yönteminin daha iyi bir çözüm olduğunu göstermiştir.

Tezin ikinci bölümünde, tablanın sıradan PD ve uyarlanabilir kutup atamalı konum denetimleri gerçekleştirilmiştir. Kestirici parametrelerinin, kestirici ve denetleyici tepkileri üzerindeki etkileri incelenmiştir. Sonuçlar, uyarlanabilir denetimin sıradan PD denetleyiciden daha iyi olduğunu göstermiştir.

TABLE OF CONTENTS

	<u>PAGE</u>
TITLE PAGE	i
PAGE OF APPROVAL	ii
ACKNOWLEDGMENTS	iii
ABSTRACT	iv
KISA ÖZET	v
LIST OF FIGURES	viii
LIST OF SYMBOLS	xiii
LIST OF ABBREVIATIONS	xv
I. INTRODUCTION	1
II. ADAPTIVE CONTROL THEORY	5
2.1. Introduction	5
2.2. Adaptive Control Schemes	10
2.2.1. Model Reference Adaptive Systems	10
2.2.2. Self-Tuning Regulators(STR)	11
2.3. Pole Placement Design	15
2.4. Relationship Between MRAS and STR	20
III. IDENTIFICATION AND PARAMETER ESTIMATION	22
3.1. Schroeder-Phased Harmonic Sequence Input ...	24
3.2. Recursive Least Squares Parameter Estimation in Time Domain	25
IV. MATHEMATICAL MODELING AND ADAPTIVE VELOCITY CONTROL OF A DC MOTOR DRIVEN TABLE SYSTEM	31
4.1. Mathematical Modeling of the System	31
4.2. Recursive Least Squares Estimation Formulation.....	36
4.3. Control Algorithm	37
4.4. Pole Assignment for PI Velocity Control	39

V.	ADAPTIVE POSITION CONTROL OF THE DC MOTOR DRIVEN	
	TABLE SYSTEM	44
	5.1. Mathematical Modeling of the Position	
	Controlled System	44
	5.2. Pole Assignment for Adaptive Position	
	Control	46
VI.	EXPERIMENTAL SET-UP ANALYSIS AND RESULTS.....	51
	6.1. Experimental Set-up	51
	6.1.1. Linear Encoder	51
	6.1.2. Ball Screw	53
	6.1.3. Interface Card and Power Amplifier ..	53
	6.2. Computer Programming	54
	6.3. Experimental Analysis	55
	6.4. Experimental Results	56
	6.4.1. Adaptive Velocity Control	57
	6.4.2. Adaptive Position Control	59
VII.	CONCLUSIONS AND RECOMMENDATIONS.....	62
APPENDIX A	65
APPENDIX B	76
FIGURES	88
BIBLIOGRAPHY	103
REFERENCES NOT CITED	108

LIST OF FIGURES

	<u>PAGE</u>
Figure 1.1 A DC motor driven single axis table model.	89
Figure 2.1 Basic configuration of an adaptive system.	6
Figure 2.2 Block diagram of model reference adaptive system.	11
Figure 2.3 Block diagram of a self-tuning regulator.	12
Figure 4.1 Schematic representation of a DC motor driven single axis table system.	31
Figure 4.2 Structural diagram for DDC of the table speed of a DC motor-table system.	33
Figure 4.3 Block diagram of speed control of a DC motor driven table system.	34
Figure 5.1 Block diagram of position control of a dc motor driven table system.	44
Figure 6.1. (a) Effect of initial estimate on speed response in self-tuning adaptive control with $\lambda = 0.96$ and $\rho = 10000$ where $J_e(0) = 0.00001 \text{ kgm}^2$.	90

Figure 6.1.(b) Effect of initial estimate on speed response in self-tuning adaptive control with $\lambda = 0.96$ and $\rho = 10000$ where $J_e(0) = 0.01 \text{ kgm}^2$. 90

Figure 6.2.(a) Effect of forgetting factor on estimated parameters in self-tuning adaptive control with $\rho=10000$ and $J_e=0.0001 \text{ kgm}^2$ where $\lambda=0.93$. 91

Figure 6.2.(b) Effect of forgetting factor on estimated parameters in self-tuning adaptive control with $\rho=10000$ and $J_e=0.0001 \text{ kgm}^2$ where $\lambda=0.99$. 91

Figure 6.3 Effect of forgetting factor on speed response in self-tuning adaptive control with 10% overshoot and 1s settling time. 92

Figure 6.4.(a) Effect of initial covariance matrix on speed response with $\lambda=0.96$, $J_e=0.0001 \text{ kgm}^2$ and $\rho=100$ where $P(0)=\rho I$. 93

Figure 6.4.(b) Effect of initial covariance matrix on parameter estimates with $\lambda=0.96$, $J_e=0.0001 \text{ kgm}^2$ and $\rho=100$ where $P(0)=\rho I$. 93

Figure 6.5.(a) Self-tuning adaptive speed response with optimal $\lambda=0.96$, $\rho=10000$, and estimated J_e of 0.0001 kgm^2 . 94

Figure 6.5.(b) PI speed response with controller parameters tuned using Ziegler-Nichols method.	94
Figure 6.6.(a) Estimated parameter 1 of the self-tuning adaptive control algorithm with $\lambda=0.96$, $\rho=10000$, and estimated J_e of 0.0001 kgm^2 .	95
Figure 6.6.(b) Estimated parameter 2 of the self-tuning adaptive control algorithm with $\lambda=0.96$, $\rho=10000$, and estimated J_e of 0.0001 kgm^2 .	95
Figure 6.7.(a) One period of SPHS signal with $T_p=2.5\text{s}$ and $NH=30$.	96
Figure 6.7.(b) Parameter Estimate θ_1 of the SPHS injected open-loop plant.	96
Figure 6.8.(a) Effect of initial estimate on estimated parameter 1 in adaptive position control with $\lambda = 0.96$ and $\rho = 1000$.	97
Figure 6.8.(b) Effect of initial estimate on estimated parameter 3 in adaptive position control with $\lambda = 0.96$ and $\rho = 1000$.	97
Figure 6.9.(a) Effect of forgetting factor on estimated parameter 1 in adaptive position control with $J_e(0) = 0.0001 \text{ kgm}^2$ and $\rho = 1000$.	98

- Figure 6.9.(b) Effect of forgetting factor on estimated parameter 3 in adaptive position control with $J_e(0) = 0.0001 \text{ kgm}^2$ and $\rho = 1000$. 98
- Figure 6.10.(a) Effect of initial covariance matrix on estimated parameter 1 in adaptive position control with $\lambda=0.96$ and $J_e(0)=0.0001 \text{ kgm}^2$. 99
- Figure 6.10.(b) Effect of initial covariance matrix on estimated parameter 3 in adaptive position control with $\lambda=0.96$ and $J_e(0)=0.0001 \text{ kgm}^2$. 99
- Figure 6.11.(a) Effect of forgetting factor on the system response in adaptive position control with $J_e(0)=0.0001 \text{ kgm}^2$, $\rho=1000$ and one percent overshoot. 100
- Figure 6.11.(b) Effect of forgetting factor on the system response in adaptive position control with $J_e(0)=0.0001 \text{ kgm}^2$, $\rho=1000$ and 10 percent overshoot. 100
- Figure 6.12. Effect of initial estimate on the system response in adaptive position control with $\lambda = 0.96$ and $\rho = 1000$. 101
- Figure 6.13. Effect of initial covariance matrix on the system response in adaptive position control with $\lambda=0.96$ and $J_e(0)=0.0001 \text{ kgm}^2$. 101

Figure 6.14.(a) Comparison of ordinary PD pole assignment(1) and adaptive(2) controllers in position control of the table with $J_e(0)=0.0001 \text{ kgm}^2$. 102

Figure 6.14.(b) Comparison of ordinary PD pole assignment(1) and adaptive(2) controllers in position control of the table with $J_e(0)=0.01 \text{ kgm}^2$. 102



LIST OF SYMBOLS

b_m	Viscous frictional coeff. of the dc motor
B_e	Equivalent viscous frictional coefficient
$D(s), D(z)$	Controller transfer function
$e(i)$	Estimation error ($y(i) - \hat{y}(i)$)
$E(s), E(z)$	Function of error
I	Current
i	Timing belt conversion rate
J	The loss function
J_m	Moment of inertia of the dc motor
J_e	Equivalent inertia of the motor and load
K_a	Power amplifier gain
K_e	Encoder gain
K_m	DC motor gain
K_p	Proportional control coefficient
K_i	Integral control coefficient
K_d	Derivative control coefficient
L_a	Armature inductance of the dc motor
N	Conversion rate between the rotational motion of the dc motor shaft and the translational motion of the table
NH	Number of harmonics of the SPHS signal
P	Covariance matrix
q	Forward shift operator
R_a	Armature resistance
s	Laplace domain operator
T	Sampling interval, torque
t	Time

u	Controller output
$U(s), U(z)$	Function of control signal
v	Velocity of the table
V	Voltage
$V(s), V(z)$	Function of speed of the table
x	Position of the table
$X(s), X(z)$	Function of position of the table
z	z domain operator
ϕ	Magnetic field
Φ	Measurement vector
λ	Forgetting factor
ω	Velocity of the dc motor shaft
ω_{bs}	Velocity of the ball screw shaft
ω_n	Natural frequency
ω_0	Fundamental frequency in frequency domain
	least squares estimation
$\Omega_m(s)$	Function of the dc motor shaft speed
$\Omega_{bs}(s)$	Function of the ball screw speed
Θ	Parameter vector
$\hat{\Theta}$	Parameter estimate vector
τ	Time constant
ζ	Damping factor

LIST OF ABBREVIATIONS

AC	Adaptive Control
IP	Index of Performance
MRAS	Model Reference Adaptive Systems
PRBS	Pseudo-Random Binary Sequence
PWM	Pulse Width Modulated
RLS	Recursive Least Squares
STAC	Self-Tuning Adaptive Control
STR	Self-Tuning Regulator

I. INTRODUCTION

In most feedback control systems, small deviations in parameter values from their design values will not cause any problem in the normal operation of the system, provided these parameters are inside the loop. If plant parameters vary widely according to the environmental changes, however, then the control system may exhibit satisfactory response for one environmental condition but may fail to provide satisfactory performance under other conditions. In certain cases, large variations of plant parameters may even cause instability.

If the plant transfer function can be identified continuously, then we can compensate variations in the transfer function of the plant simply by varying adjustable parameters of the controller and thereby obtain satisfactory system performance continuously under various environmental conditions. Such an adaptive approach is quite useful to cope with a problem where the plant is normally exposed to varying environments so that plant parameters change from time to time [1].

In the early 1950s there was extensive research on adaptive control, in connection with the design of autopilots for high performance aircraft. Such an aircraft operates over a wide range of speeds and altitudes. It was found that ordinary constant-gain, linear feedback control could work well in one operating condition, but that changed operating conditions led to difficulties. A more sophisticated regulator, which could work well over a wide range of

operating conditions, was therefore needed. Interest in the subject diminished due to the lack of insight and a disaster in a flight test.

In the 1960s many contributions to control theory were important for the development of adaptive control. State space and stability theory were introduced. There were also important results in stochastic control theory. Dynamic programming, introduced by Bellman, increased the understanding of the adaptive processes. Fundamental contributions were also made by Tsytkin, who showed that many schemes for learning and adaptive control could be described in a common frame work as recursive equations of a special type. There were also major developments in system identification and in parameter estimation. There was a renaissance of adaptive control in the 1970s, when different estimation schemes were combined with various design methods. Many applications were reported, but theoretical results were very limited.

In the late 1970s and early 1980s correct proofs for stability of adaptive systems appeared, albeit under very restrictive assumptions. Investigation of the necessity of those assumptions has sparked new and interesting research into the robustness of adaptive control, as well as into controllers that are universally stabilizing. Rapid and revolutionary progress in microelectronics has made it possible to implement adaptive regulators simply and cheaply. Vigorous development of the field is now taking place, both at universities and in industry. Several commercial adaptive regulators based on different ideas are appearing on the market, and the industrial use of adaptive control is growing

slowly but surely [2].

Likewise the situation described in the first paragraph, the control of DC motors and DC motor driven systems reveals the need for more sophisticated control algorithms. This is mainly due to the fact that the dynamic behavior of a DC motor varies essentially with the moment of inertia and with the friction of the load [3]. This is a typical situation taking place in a variety of applications such as machine tools, rolling mills, wiring machines, packaging machines, robots, etc. Also, in number of cases encountered in DC motor control systems, it is difficult to determine the values of the dynamic parameters with known structures [4]. Similarly, the cutting force dependent upon the depth of cut, feedrate, speed of cutter in machining operations are of typical "time-variant" parameters which may affect the quality and productivity of the production process [5].

Thus, to eliminate the necessity of initial tuning and detuning of the DC motor controller, and to overcome the problems that are expected to be arised when there are parameter changes because of the slightly time-variant friction throughout the guide rods, an adaptive controller has been designed to control the servo system. As a design technique, a pole-placement explicit STR has been implemented. In the estimation of the system parameters, the well-known recursive least squares estimation algorithm with UD factorization method has been used.

In Chapter II and III, adaptive control theory and the most important part of it, identification and parameter

estimation, have been introduced. The mathematical modelling of the system and the adaptive velocity controller design have been presented in Chapter IV. The design of the adaptive position controller with no cancellation of process zero has been given in Chapter V. In Chapter VI, the experimental set-up has been described first, and then the experimental procedure and the experimental results have been given. Finally, the major conclusions were drawn in the last chapter of the thesis.



II. ADAPTIVE CONTROL THEORY

2.1. Introduction

In everyday language to "adapt" means to change a behavior to conform to new circumstances. Intuitively, an adaptive regulator, is a regulator that can modify its behavior in response to changes in the dynamics of the process and the disturbances. Since ordinary feedback has been introduced for the same purpose, the question of the difference between feedback control and adaptive control immediately arises. In a sense, adaptive control is a special type of nonlinear feedback control in which the states of the process can be separated in two categories, which change at different rates. The slowly changing states are viewed as parameters. This introduces the idea of two time scales: a fast time scale for the ordinary feedback and a slower one for updating the regulator parameters. This also implies that linear constant parameter regulators are not adaptive. In an adaptive controller it is also assumed that there is some kind of feedback from the performance of the closed-loop system. This implies that gain scheduling should not be regarded as an adaptive controller, since the parameters are determined by a schedule, without any feedback from the performance [2,4].

Several definitions of adaptive system have been made and the agreement over definitions for adaptive systems remains a very controversial area [1,6,7]. Some of the other

definitions of adaptive system can be found in [8].

Definition of an adaptive system [3]: An adaptive system measures a certain index of performance (IP) using the inputs, the states, and the outputs of the adjustable system. From the comparison of the measured index of performance and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable system or generates an auxiliary input in order to maintain the index of performance close to the set of given ones. This definition is illustrated in Fig.2.1.

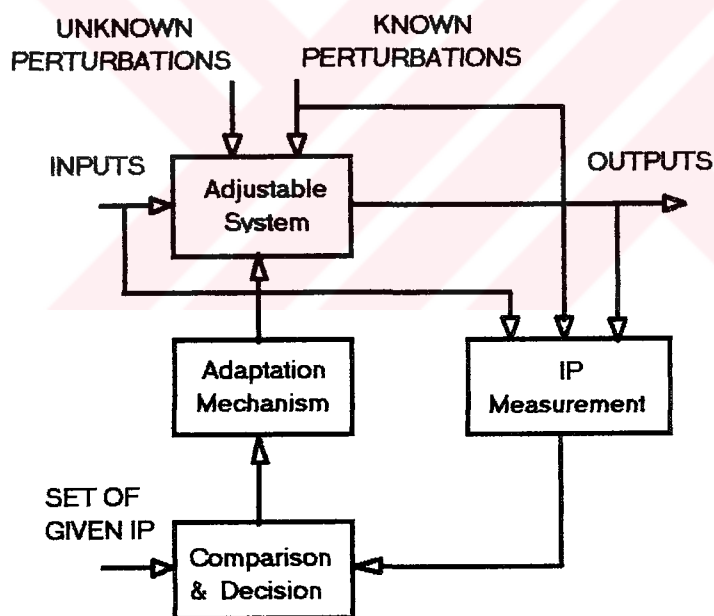


Fig.2.1. Basic configuration of an adaptive system.

In the definition, the *adjustable* system must be understood to be a system capable of adjusting its performance either by modifying its parameters (internal structure) or by modifying its input signals.

The measurement of the *index of performance* can be done in several ways, sometimes directly, sometimes indirectly, as, for example, through the *identification* of the dynamic parameters of the system.

The *comparison-decision block* makes the comparison between the given set of IP's and the measured IP and decides if the measured IP is within the *acceptable* IP set. If not, the adaptation mechanism will act accordingly so as to modify the system performance either by modifying the parameters of the adjustable system or by changing the system control inputs.

In general, the index of performance measurement is achieved through identification of the process. Especially, when the model is unknown, on-line parameter estimation (identification) could be combined with on-line control. Most of the adaptive controllers are designed with regard to this principle.

From the discussion above it can be seen that underlying each of the problems of adaptive control, there is some form of parameter estimation. Virtually, parameter estimation forms an integral part of almost any adaptive scheme. It is useful to distinguish between situations in which the system parameters are constant and situations in which the parameters vary with time. Obviously, the former situation is easier to deal with than the latter. A feature of many of the parameter estimation algorithms used in the time-invariant case is that the gain of the algorithms ultimately decreases to zero. This means that when all of the parameters have been estimated, the algorithm "turns off". On the other hand, if the parameters are time varying, it is

necessary for the estimation algorithm to have the capability of tracking the time variations. This capability directly affects the performance of the adaptive controller.

Summing up, an adaptive control system must provide continuous information about the present state of the plant, that is it must *identify* the process; it must compare the present system performance and make a *decision* to adapt the system so as to tend toward optimum or desired performance; and finally it must initiate a proper *modification* to drive the system toward optimum [10]. These three functions are inherent in an adaptive system.

Adaptation and Tuning: It is customary to separate the tuning and adaptation problems. In the tuning problem it is assumed that the process to be controlled has unknown but constant parameters; in the adaptation problem it is assumed that the parameters are changing or simply a function of time. Many issues are much easier to handle in the tuning problem. The convergence problem is to investigate whether the parameters converge to their true values [2,11].

Parameter Adaptive and Structurally Adaptive Systems: In a dynamical system uncertainties may arise due to a lack of knowledge of some of the system or noise parameters that are elements of a parameter vector or some of the functions. In both cases, other elements of the parameter vector or the functions, may be varied appropriately to achieve the desired control. The former are said to be parameter adaptive, while the latter are said to be structurally adaptive. Switching systems, where structural changes can take place due to modifications in the interconnections between subsystems

belong to the latter class [8].

Direct and Indirect Control: Two philosophically different approaches exist for the solution of the adaptive control problem. In the first approach, referred to as *indirect control*, the plant parameters are estimated on-line and the control parameters are adjusted based on these estimates. Such a procedure has also been referred to as *explicit identification* in the literature [2,8]. In contrast to this, in what is referred to as *direct control*, no effort is made to identify the plant parameters but the control parameters are directly adjusted to improve a performance index. This is also referred to as *implicit identification*.

Relations to Other Areas of Automatic Control: Adaptive control is by no means a mature field. Many of the algorithms and approaches used are of an ad hoc nature; the tools are gathered from a wide range of fields; and good systematic approaches are still lacking. Yet the algorithms and systems have found good uses, and there are adaptive systems that clearly outperform conventional feedback systems. To pursue adaptive control one must have a background in conventional feedback control and also sampled data systems. There are strong ties to nonlinear system theory, because adaptive systems are inherently nonlinear. Stability theory is a key element. Adaptive control also has connections to singular perturbations and averaging theory, because of the separation of time scales in adaptive systems. There are also links to stochastic control and parameter estimation, because one way to look at adaptive systems is to view them as a combination of parameter estimation and control [2].

2.2. Adaptive Control Schemes

A robust high-gain regulator which is a constant gain regulator designed to cope with parameter variations is a point of departure in the discussion of the adaptive control schemes. The different heuristic adaptive systems such as self-oscillating adaptive controller and gain scheduling are the other techniques for reducing the effects of parameter variations [2]. However, in this section, only the model reference adaptive systems (MRAS) and the self-tuning regulators (STRs) will be outlined. The latter scheme, the so-called self-tuning regulator which is the scheme used in this study will be discussed in detail.

2.2.1. Model Reference Adaptive Systems

The model reference adaptive system is one of the main approaches to adaptive control. The basic principle is illustrated in Fig.2.2. The desired performance is expressed in terms of a reference model, which gives the desired response to a command signal. The system also has an ordinary feedback loop composed of the process and the regulator. The error e is the difference between the outputs of the system and the reference model. The regulator has parameters that are changed based on the error. There are thus two loops in Fig.2.2: an inner loop, which provides the ordinary control feedback, and an outer loop, which adjust the parameters in the inner loop. The inner loop is assumed to be faster than the outer loop.

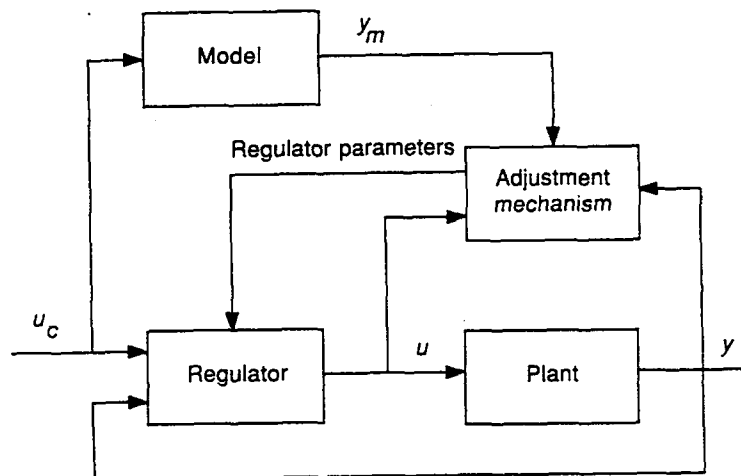


Fig.2.2. Block diagram of a model-reference adaptive system.

Fig.2.2 is the original MRAS, proposed by Whitaker in 1958, in which two new ideas were introduced. First, the performance of a system is specified by a model; secondly, the parameters of the regulator are adjusted based on the error between the reference model and the system. Model reference adaptive systems were originally derived for the servo problem in deterministic continuous-time systems. The idea and the theory have been extended to cover discrete-time systems and systems with stochastic disturbances [2].

2.2.2. Self-Tuning Regulators

In an adaptive system it is assumed that the regulator parameters are adjusted all the time. This implies that the regulator parameters follow the changes in the process. However, it is difficult to analyze the convergence and stability properties of such systems. To simplify the problem it can be assumed that the process has constant but

unknown parameters. When the process is known, the design procedure specifies a set of desired controller parameters. The adaptive controller should converge to these parameter values even when the process is unknown. A regulator with this property is called *self-tuning*, since it automatically tunes the controller to the desired performance. Block diagram of a STR is shown in Fig.2.3. Self-tuning controllers represents an important class of adaptive controllers and they are easy to implement and applicable to complex processes with wide variety of characteristics involving unknown parameters, the presence of time delay, time-varying process dynamics, and stochastic disturbances. It should be noted that, in the literature, traditionally, control of systems with unknown constant or slowly varying parameters is called self-tuning control, and the control of systems with time-varying parameters is called adaptive control. Of course, in fact, an STR is capable of controlling both systems with constant or time-varying parameters beyond the traditional classifications in the literature.

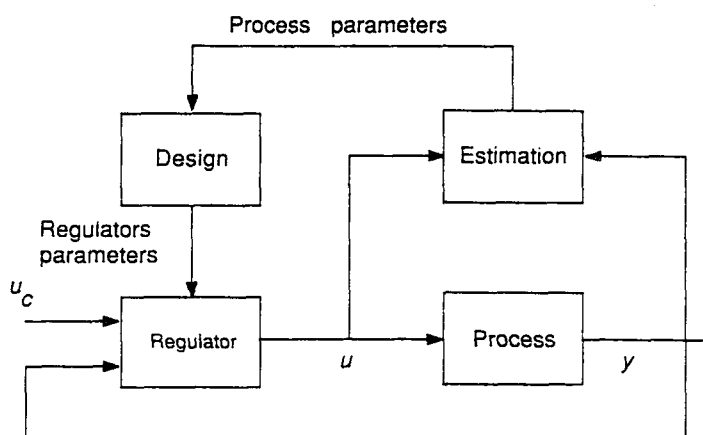


Fig.2.3. Block diagram of a self-tuning regulator.

The original idea of self-tuning was first given by Kalman in 1958. The major breakthrough came with the self tuning regulator of Aström and Wittenmark [12] in 1973.

In self-tuning control, process parameters are updated and the regulator parameters are obtained from the solution of a design problem as shown in the figure. The adaptive regulator can be thought of as being composed of two loops. The inner loop consists of the process and an ordinary linear-feedback regulator. The parameters of the regulator are adjusted by the outer loop, which is composed of a recursive parameter estimator and a design calculation. As it has been treated in chapter III, to obtain good estimates it may also be necessary to introduce perturbation signals [9,11]. Notice that the system may be viewed as an automation of process modeling and design where the process model and the control design are updated at each sampling period.

The block labeled "design calculations" in Fig.2.3 represents an on-line solution to a design problem for a system with known parameters. This is called the *underlying design problem* [2,6,7]. Such a problem can be associated with most adaptive control schemes. However, the problem is often given indirectly. To evaluate adaptive control schemes it is often useful to find the underlying design problem because it will give the characteristics of the system under the ideal conditions when the parameters are known exactly.

The self-tuner also contains a recursive parameter estimator. Many different estimation schemes have been used [2,10,13,14,15,16], for example least squares, stochastic approximation, extended and generalized least squares,

instrumental variables, extended Kalman filtering, and the minimum and the maximum likelihood methods. The most common scheme least squares method has been formulated and discussed in detail in Chapter III.

The self-tuning regulator is very flexible with respect to the design method. Virtually any design technique can be accommodated. Self-tuners based on pole placement, phase and amplitude margins, minimum-variance control and LQG (Linear Quadratic Gaussian) control are some of the common design techniques [2,6,7,11]. However, basically, the self-tuning algorithms can be divided into two major classes: *implicit* and *explicit* algorithms. In an explicit algorithm, there is an estimation of an explicit process model. Thus, the self-tuner shown in Fig.2.3 is an explicit STR. It is sometimes possible to reparameterize the process so that it can be expressed in terms of the regulator parameters. This gives a significant simplification of the algorithm because the design calculations are eliminated. In terms of Fig.2.3, the block labeled design calculations disappears and the regulator parameters are updated directly. Such a self-tuner is called an implicit self-tuning regulator and it is therefore based on the estimation of an implicit process model.

The most common implicit self-tuning regulator is the minimum-variance STR [17]. On the other hand, the most commonly used explicit STR is the pole-placement STR. The pole-placement STR is of use even in regulating non-minimum phase systems (systems having a pole or zero outside the unit circle) or systems involving unknown time delays and such

systems cannot be handled by direct minimum variance methods [13].

Another pole-placement STR approach is to use PID (or PI,PD,P) self-tuning regulator scheme [14]. This scheme is principally of type explicit self-tuning regulators. In this approach, regulator parameters are parameterized in terms of the parameters of the process and the parameters of the desired closed-loop transfer function. At each step, the recursive estimates of the process parameters are used to find the regulator parameters K_p , K_i , and K_d that are expected to give the desired performance.

2.3. Pole Placement Design [2]:

Pole placement is one of the simpler direct design procedures. The key idea is to find a feedback law such that the closed-loop poles have the desired locations. Let the process to be described by

$$Ay = Bu + v \quad (2.1)$$

where u is the control variable, y the measured output, and v a disturbance. The symbols A and B denote polynomials in the differential operator $p = d/dt$ for continuous-time systems or the forward shift operator q for discrete-time systems. It is assumed that A and B are relatively prime, i.e., that they do not have any common factors. Further, it is assumed that A is *monic*, i.e., that the coefficient of the highest power in A

is unity. The *pole-excess* $d = \deg A - \deg B$ is for discrete-time systems the time delay of the process. Let the desired response from the reference signal u_c to the output be described by the dynamics

$$A_m y_m = B_m u_c \quad (2.2)$$

Furthermore, let A_o be the specified observe polynomial. The dynamics of the observer are not controllable from the reference input u_c . To get a differentiation-free (continuous-time) or causal (discrete-time) controller, the model Eq.(2.2) must have same or higher pole-excess as the process of Eq.(2.1). This gives the condition

$$\deg A_m - \deg B_m \geq \deg A - \deg B \quad (2.3)$$

A general linear regulator can be described by

$$Ru = Tu_c - Sy \quad (2.4)$$

Elimination of u between Eqs.(2.1) and (2.4) gives

$$y = \frac{BT}{AR+BS} u_c + \frac{R}{AR+BS} v$$

$$u = \frac{AT}{AR+BS} u_c - \frac{S}{AR+BS} v$$

To achieve the desired input-output response, the following condition must hold

$$\frac{BT}{AR+BS} = \frac{B_m}{A_m} \quad (2.5)$$

The denominator $AR + BS$ is the closed-loop characteristic polynomial. To carry out the design, the polynomial B is factored as

$$B = B^+ B^-$$

where B^+ is a monic polynomial whose zeros are stable and so well damped that they can be canceled by the regulator. When $B^+ = 1$, there is no cancellation of any zeros. Since B^+ is canceled, it also factors the closed-loop characteristics polynomial. The other factors of this are A_m and A_o . This gives the Diophantine equation

$$AR + BS = A_o A_m B^+$$

It follows from this equation that B^+ divides R . Hence

$$R = R_1 B^+$$

$$AR_1 + B^- S = A_o A_m \quad (2.6)$$

The solution of the Diophantine equation (Eq.2.6) is essentially the same as solving a set of linear equations. Eq.(2.6) has a unique solution if A and B^- are relative prime. Furthermore, it follows from Eq.(2.5) that B^- must divide B_m and that

$$T = A_0 B_m / B^- \quad (2.7)$$

The pole placement procedure can now be summarized in the following algorithms.

Algorithm 1-Pole placement design

Data: Polynomials A, B.

Specifications: Polynomials A_m , B_m , and A_0 .

Compatibility Conditions:

$$B^- \text{ divides } B_m$$

$$\deg A_m - \deg B_m \geq \deg A - \deg B \quad (2.8)$$

$$\deg A_0 \geq 2\deg A - \deg A_m - \deg B^+ - 1 \quad (2.9)$$

Step 1: Factor B as $B = B^+ B^-$

Step 2: Solve R_1 and S from the equation $AR_1 + B^- S = A_0 A_m$.

Step 3: Form $R = R_1 B^+$ and $T = A_0 B_m / B^-$.

The control law is then given by

$$Ru = Tu_c - Sy$$

There are many variations of the pole placement procedure. A particularly simple case is when the whole B polynomial is canceled. We then obtain the following algorithm.

Algorithm 2-Pole placement design with all zeros cancelled

Data: Polynomials A, B.

Specifications: Polynomials A_m , B_m , and A_0 .

Compability Conditions:

$$\deg A_m - \deg B_m \geq \deg A - \deg B$$

$$\deg A_o \geq 2\deg A - \deg A_m - \deg B - 1$$

Step 1: Factor B as $B = b_o B^+$

Step 2: Solve $AR_1 + b_o S = A_o A_m$.

Step 3: Form $R = R_1 B^+$ and $T = A_o B_m / b_o$.

The control law is

$$Ru = Tu_c - Sy$$

Notice that Step 2 is very simple in this case. R_1 is simply the quotient, and $b_o S$ the remainder when dividing $A_o A_m$ by A . This implies that the coefficients in the R and S polynomials can be obtained from a triangular linear system of equations. Another useful special case is when there are no cancellations at all. The design procedure then becomes that of the following algorithm.

Algorithm 3-Pole placement design with no cancellations

Data: Polynomials A , B .

Specifications: Polynomials A_m , B_m , and A_o .

Compability Conditions:

B divides B_m

$$\deg A_m - \deg B_m \geq \deg A - \deg B \quad (2.8)$$

$$\deg A_o \geq 2\deg A - \deg A_m - 1 \quad (2.9)$$

Step 1: Solve $AR + BS = A_o A_m$.

Step 2: Form $T = A_o B_m / B$.

The control law is

$$Ru = Tu_c - Sy$$

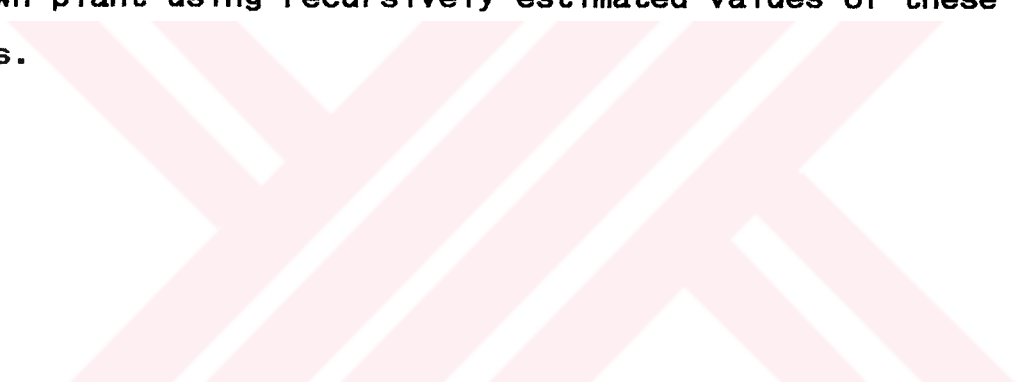
2.4. Relationship between MRAS and STR

The MRAS originated from a continuous time, deterministic servo-problem and the STR from a discrete time, stochastic regulation problem. The two approaches are quite similar in spite of their different origins. This can be seen superficially from the comparison of Fig.2.2 and Fig.2.3. Both systems have two feedback loops. The inner loop is an ordinary feedback loop with a process and a regulator. The regulator has adjustable parameters which are set by the outer loop. The adjustments are based on feedback from the process inputs and outputs. The method for designing of the inner loop and the techniques used to adjust the parameters in the outer loop are, however different.

The regulator parameters are updated directly in the MRAS in Fig.2.2. In the STR in Fig.2.3, they are updated indirectly via parameter estimation and design calculations. This difference is, however, not fundamental because the STR may be modified so that the regulator parameters are updated directly, i.e., if the implicit scheme is adopted. Virtually, the regulator shown in Fig.2.3 can also be derived from the MRAS approach if the parameter estimation is done by updating a reference model. This scheme is called the *indirect* MRAS

because the regulator parameters are updated indirectly via the design calculation. So, we can say that the *direct* MRAS, where the regulator parameters are updated directly, is closely related to the *implicit* STR; the *indirect* MRAS is related to the *explicit* STR.

However, besides this relationship, it should be noticed that whereas in MRAS the basic idea is to asymptotically drive the output of an unknown plant to that of a reference model, in self-tuning the basic procedure is to select a design for known plant parameters and apply it to the unknown plant using recursively estimated values of these parameters.



III. IDENTIFICATION AND REAL-TIME PARAMETER ESTIMATION

On-line determination of process parameters is a key element in adaptive control and it is an important part of a self-tuning regulator. Parameter estimation also occurs implicitly in a model-reference adaptive regulator. In adaptive systems, parameter estimation is used in the larger context of system identification. The key elements of system identification are selection of model structure, experiment design, parameter estimation, and validation. The model structures are derived from prior knowledge of the process and the disturbances. In some cases the only a priori knowledge is that the process can be described as a linear system in a particular operating range. It is then natural to use general representations of linear systems. It is often difficult and costly to experiment with industrial processes. Therefore, it is desirable to have identification methods that do not require special input signals. Many "classic" methods depend strongly on having the input be of a precise form, e.g., sinusoid or impulses. One requirement of the input signal is that it should excite all the modes of the process sufficiently. A good identification method should thus be insensitive to the characteristics of the input signal. Solving the parameter estimation problem requires, input-output data from the process, a class of models and a criterion. The criterion is introduced to give a measure how well a model fits the experimental data. Parameter estimation can then be formulated as an optimization problem [2].

The performance of the controller strictly depends on

how close the estimated parameters are to the real ones. Consequently, several recursive estimation algorithms have been developed. Among them, recursive least squares, extended and generalized least squares, minimum and maximum likelihood are the most popular schemes. However, recursive least squares method have dominated the other schemes. Furthermore, there are several recursive least squares algorithms such as square-root algorithm and the so-called UD method. All the mentioned schemes are time domain parameter estimation methods. On the other hand, there exist least-squares frequency-domain parameter estimation methods. Frequency-domain parameter estimation scheme is used in general in off-line form and is not recursive. A further work is needed to make it an attractive on-line estimation method [18]. Therefore, one can use least squares frequency-domain estimation method to estimate the parameters of a time-invariant process. Another use of frequency-domain estimation scheme is that the estimation of the *initial state* of the parameters of a time-varying process.

For convergence of the parameter estimates to the true parameter values, it is well known that the input signal must be "sufficiently rich" or "persistently exciting" [2,9,19]. This perturbation signal should be a white noise, i.e., a signal having a flat modulus spectrum. In general, the pseudo-random binary sequence (PRBS) [20] or the Schroeder-phased harmonic sequence (SPHS) [5,18,21] is used. In this study, the latter is used to excite the system, and to validate the obtained experimental results, the SPHS is injected as input to the experimental set-up. This will be discussed extensively in experimental results part of the thesis. In this chapter, only the formulation of the SPHS

will be presented and then time domain least squares parameter estimation scheme will be outlined.

3.1. Schroeder-Phased Harmonic Sequence (SPHS) Input

SPHS is a periodical signal which can be constructed using the formula;

$$f(t) = \sum_{k=1}^{NH} (2p_k)^{1/2} \cos(k\omega_0 t + \theta_k) \quad (3.1)$$

where p_k , with $\sum p_k = 1$, is the relative power and θ_k is the phase angle of the k th harmonic. NH is the number of harmonics, and ω_0 is the fundamental frequency of the SPHS and equal to $2\pi/T$, where T is the period of the signal.

The SPHS can be synthesized to give any arbitrarily defined spectrum, including a flat modulus spectrum with a sharp cut-off. This is not possible with PRBS which always has the same shape of modulus spectrum and possesses parasitic frequencies beyond its flat bandwidth. The SPHS is a low peak factor signal and persistently excites all of the system modes without violating the linear operating condition [18].

In order to reach a low-peak factor signal with flat spectrum, one should choose $p_k = 1/NH = \text{const.}$, so that each harmonic will contain a constant proportion $1/NH$ of the total average power. Then Eq.(3.1) becomes;

$$f(t) = \sum_{k=1}^{NH} (2/NH)^{1/2} \cos\left(\frac{2\pi k}{T}t + \theta_k\right), \quad k=1,2,\dots,\delta,\dots,NH \quad (3.2)$$

with

$$\theta_\delta = \frac{2\pi}{NH} \sum_{k=1}^{\delta} (k), \quad \delta=1,2,\dots,NH$$

The characteristics of the SPHS are then

$$\text{Resolution (k=1)} = 2\pi/T \text{ rad/sec} = 1/T \text{ Hz}$$

$$\text{Bandwidth (k=NH)} = 2\pi NH/T \text{ rad/sec} = NH/T \text{ Hz}$$

3.2. Recursive Least Squares Parameter Estimation in Time

Any single-input single-output dynamic system can be represented by a single high order difference equation;

$$y(k) + \rho_1 y(k-1) + \dots + \rho_n y(k-n) = \eta_0 u(k-d) + \eta_1 u(k-d-1) + \dots + \eta_m u(k-d-m) \quad (3.3)$$

where $d=(n-m) \geq 0$ and represents the system delay. For a system described by Eq.(3.3), the pulse transfer function is;

$$\frac{Y(z)}{U(z)} = \frac{z^{-d} (\eta_0 + \eta_1 z^{-1} + \dots + \eta_m z^{-m})}{1 + \rho_1 z^{-1} + \dots + \rho_n z^{-n}} = \frac{\eta_0 z^m + \eta_1 z^{m-1} + \dots + \eta_m}{z^n + \rho_1 z^{n-1} + \dots + \rho_n} \quad (3.4)$$

The problem is that determining ρ_i and η_i from input-output measurements. Assume that the loss function to be minimized is chosen to be;

$$J = \sum_{i=1}^k e(i)^2 \quad (3.5)$$

where the estimation error is

$$e(i) = y(i) - \hat{y}(i) \quad (3.6)$$

and

$$\hat{y}(i) = -\hat{\rho}_1(i)y(i-1) - \dots - \hat{\rho}_n(i)y(i-n) + \hat{\eta}_0(i)u(i-1) + \dots + \hat{\eta}_{n-1}(i)u[i-(n-1)] \quad (3.7)$$

is an estimate of $y(i)$ calculated using the parameter estimates $\hat{\rho}_i$ and $\hat{\eta}_i$. Thus, one could stop when J is decreased below a certain absolute level.

Express Eq.(3.3) using the following notation;

$$y(k) = \Theta^T \Phi(k-1) \quad (3.8)$$

where

$$\Theta^T = [\rho_1 \ \rho_2 \ \dots \ \rho_n \ \eta_0 \ \eta_1 \ \dots \ \eta_m] \quad (3.9)$$

is termed the *parameter vector*, and

$$\Phi^T(k-1) = [-y(k) \ -y(k-1) \ \dots \ -y(k-n) \ u(k-d) \ u(k-d-1) \ \dots \ \dots u(k-d-m)] \quad (3.10)$$

is the *measurement vector*. Then write Eq.(3.7) as

$$\hat{y}(k) = \hat{\Theta}(k)^T \Phi(k-1) \quad (3.11)$$

where

$$\hat{\Theta}(k)^T = [\hat{\rho}_1(k) \hat{\rho}_2(k) \dots \hat{\rho}_n(k) \hat{\eta}_0(k) \hat{\eta}_1(k) \dots \hat{\eta}_m(k)] \quad (3.12)$$

is the *parameter estimate vector* containing the $(n+m)$ parameters to be estimated. The minimization for $k \geq (n+m)$ of

$$J = \sum_{i=1}^k [y(i) - \hat{\Theta}(k)^T \Phi(i-1)]^2 \quad (3.13)$$

gives the solution;

$$\hat{\Theta}(k) = P(k) \left[\sum_{i=1}^k y(i) \Phi(i-1) \right] \quad (3.14)$$

where

$$P(k) = \left[\sum_{i=1}^k \Phi(i-1) \Phi(i-1)^T \right]^{-1} \quad (3.15)$$

Notice that the least-squares parameter estimation as given by equations (3.14) and (3.15) is suitable only for off-line computation. That is, one must first collect the input $u(i)$ and output $y(i)$ measurements of all $i=1..k$, then calculate the terms in Eq.(3.15) and invert the matrix to get $P(k)$, finally, use Eq.(3.14) to get $\Theta(k)$.

In adaptive control, it is needed to obtain an estimate of $\Theta(k)$ while collecting data and continuing to update it as new data becomes available. Such an on-line version of least-squares parameter estimation algorithm can be obtained by writing Eq.(3.14) at k and $k+1$, then writing

an expression for $\hat{\Theta}(k+1)$ in terms of $\hat{\Theta}(k)$. The matrix inversion implied by Eq.(3.15) can also be avoided by using the *matrix inversion lemma* [1,3]. This on-line version of the algorithm, usually termed recursive least squares (RLS), is given by;

$$\hat{\Theta}(k+1) = \hat{\Theta}(k) + \frac{P(k)\Phi(k)[y(k+1) - \hat{\Theta}(k)^T\Phi(k)]}{\lambda + \Phi(k)^T P(k)\Phi(k)} \quad (3.16)$$

and

$$P(k+1) = \frac{1}{\lambda} \left[P(k) - \frac{P(k)\Phi(k)\Phi(k)^T P(k)}{\lambda + \Phi(k)^T P(k)\Phi(k)} \right] \quad (3.17)$$

where initial estimates $\hat{\Theta}(0)$ and $P(0)$ are required to start the the algorithm. As stated before, in adaptive velocity control part of this study, to obtain a good estimate of $\hat{\Theta}(0)$ frequency domain least squares estimation scheme is used. For the matrix $P(0)$, one often selects $P(0)=\rho I$ where I is the identity matrix and ρ is a design parameter that a number greater than zero (e.g. 10, 1000, 1000000) and then varied until good parameter convergence is obtained. λ is called the "forgetting factor" and is a measure of how fast the old data are forgotten. If $\lambda=1$, there is no forgetting. In order to control plants having constant and unknown parameters, there is no need to forget the old data. Therefore, since the algorithm Eq.s (3.16)-(3.17) is general, to control an unknown and time- invariant process λ is chosen to be unity [22,23,24]. In order to control plants having time-varying parameters, the convenient and much-used way to make the estimator able to follow these changes is to introduce a forgetting factor less than unity. This was originally proposed in [13] and λ is chosen in general $0.95 \leq \lambda < 1$. An

another way is to introduce a variable forgetting factor [2,24,25,26].

The recursive least squares parameter identification equations, equations (3.16) and (3.17), have good physical interpretation. To interpret them, one could write these equations in the following form assuming $\lambda=1$;

$$e(k+1) = y(k+1) - \Phi(k)^T \Theta(k) \quad (3.18)$$

$$P(k+1) = P(k) - \frac{P(k)\Phi(k)\Phi(k)^T P(k)}{1 + \Phi(k)^T P(k)\Phi(k)} \quad (3.19)$$

$$\Theta(k+1) = \Theta(k) + P(k+1)\Phi(k)e(k+1) \quad (3.20)$$

Here, the new estimate $\Theta(k+1)$ is obtained by adding a correction term ($P\Phi e$) to the old estimate $\Theta(k)$. The correction term is a product of three quantities: P , Φ , and e . The error e is the difference between the last measurement $y(k+1)$ and the prediction ($\Phi^T \Theta$) of this measurement based on old estimates. Regression or the measurement vector Φ can be interpreted as the gradient of the error with respect to the parameters. Eq.(3.19) may be interpreted as follows. Matrix $P(k+1)$ is proportional to the covariance matrix of the estimates; the last term in Eq.(3.19) is the reduction in uncertainty due to the last measurement [4].

Because RLS parameter estimation algorithm is quite time consuming from a numerical point of view, some better ways of doing the calculations are developed. Two schemes are much popular. One way is to update the square root of P

instead of updating P . The other way to do the calculations is to use the U-D algorithm by Bierman and Thornton [27,28]. This method is based on a factorization of the covariance matrix P as $P=UDU^T$, where D is diagonal and U is an upper-triangular matrix. This algorithm reduces the calculations and is well suited for microcomputers and real-time applications. In this thesis, UD covariance factorization algorithm is used to mechanize the recursive least squares parameter estimation.



IV. MATHEMATICAL MODELING AND ADAPTIVE VELOCITY CONTROL OF THE DC MOTOR DRIVEN TABLE SYSTEM

4.1. Mathematical Modeling of the System

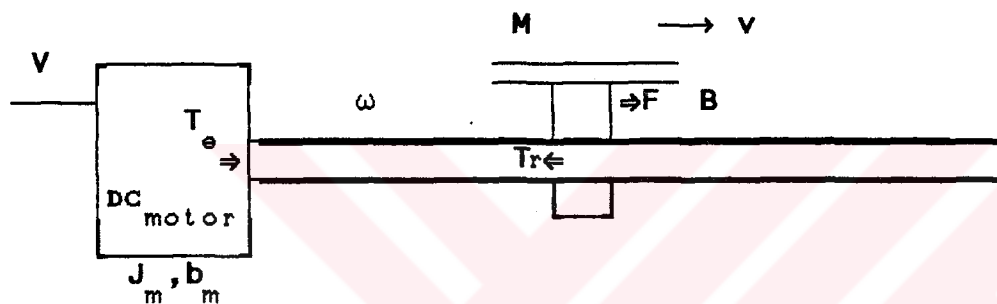


Fig.4.1.Schematic representation of a dc motor driven single axis table system [4].

The modeling of the system is done with respect to Fig.4.1. According to Newton's second law, the equation of motion of the system at the motor level is;

$$J_m \dot{\omega} + b_m \omega + T_r = T_e \quad (4.1)$$

where J_m is the moment of inertia of the motor and the actuator shaft, b_m is the viscous frictional coefficient of the motor, T_r is the resistance torque by the table, T_e is the torque applied by the dc motor and ω is the shaft velocity of the motor.

On the other hand, the equation of motion can also be

written at the table level;

$$M \dot{v} + B v = F \quad (4.2)$$

where M is the mass of the table and the load, B is the viscous frictional coefficient between the shaft and the table, F is the resistance force and v is the velocity of the table.

The conversion rate between the rotational motion of the shaft of the dc motor and the translational motion of the table is defined to be;

$$N = \frac{\omega}{v} = \frac{F}{T_r} \quad (4.3)$$

Thus, Eq.(4.1) becomes;

$$J_m \dot{\omega} + b_m \omega + \frac{F}{N} = T_e$$

or inserting Eq.(4.2);

$$J_m \dot{\omega} + b_m \omega + \frac{1}{N} (M\dot{v} + Bv) = T_e$$

and substituting Eq.(4.3), the equation of the motion is obtained;

$$\underbrace{\left(J_m + \frac{M}{N^2} \right)}_{J_e} \dot{\omega} + \underbrace{\left(b_m + \frac{B}{N^2} \right)}_{B_e} \omega = T_e \quad (4.4)$$

where J is the equivalent moment of inertia and B is the equivalent viscous frictional coefficient of the dc motor-

table system.

If Eq.(4.2) was chosen as the basis, one would obtain similarly;

$$\underbrace{(M + N^2 J_m)}_{J'_e} \dot{v} + \underbrace{(B + N^2 b_m)}_{B'_e} v = N T_e \quad (4.5)$$

Such an experimental system is shown in Fig.1.1. To control speed of the table of such a system, one could choose a linear encoder as the feedback element [29]. Therefore for the direct digital control (DDC) of such a system, the structural diagram is shown in Fig.4.2.

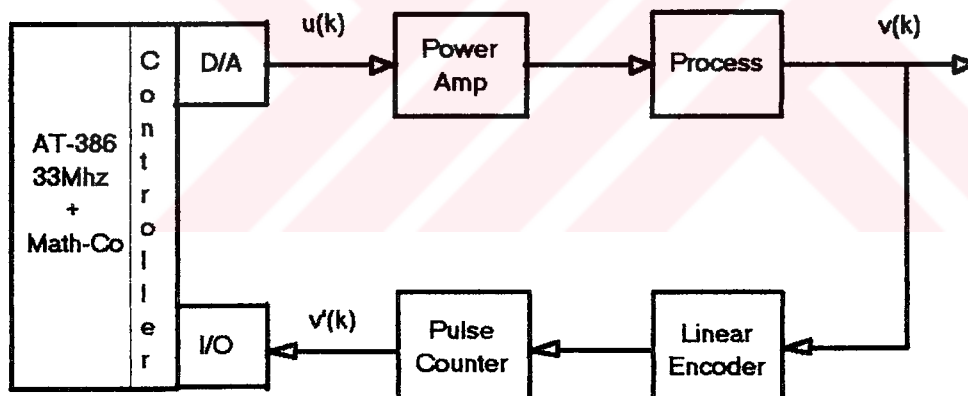


Fig.4.2. Structural diagram for DDC of the table speed of a dc motor-table system [4].

The block diagram representation of the control system is given in Fig.4.3.

In the block diagram, K_a is the power amplifier gain, K_m is the dc motor gain, K_e is encoder gain, N is the conversion rate given by Eq.(4.3), and i is the timing belt

conversion rate between the dc motor shaft and the ball screw and given by $\omega_{\text{motor}}/\omega_{\text{ballscrew}}$. In the presence of a timing belt transmission between the dc motor and the ball screw which is always the case in practice, Eq.(4.3) needs some modification and takes the form;

$$N = \frac{\omega_{\text{bs}}}{v} \quad (4.3)^*$$

where ω_{bs} is the rotational speed of the ball screw.

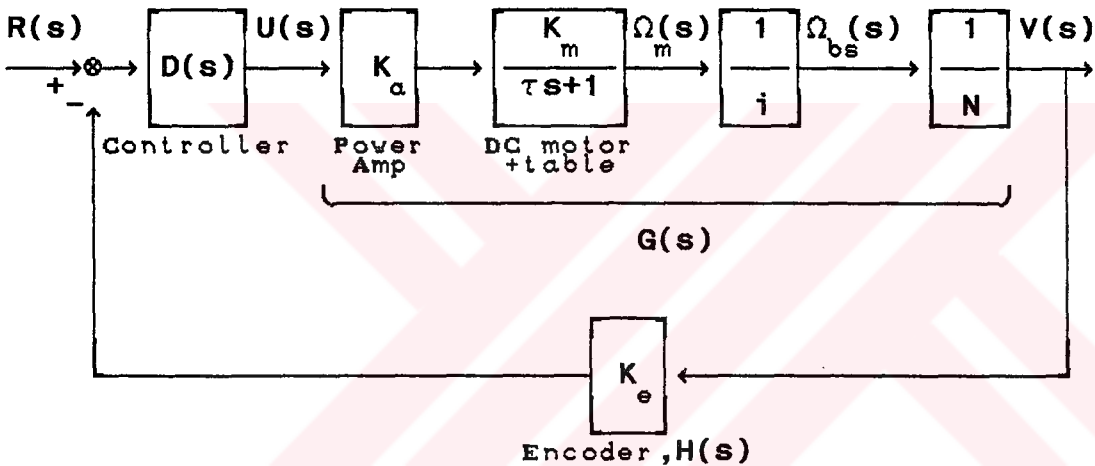


Fig.4.3. Block diagram of speed control of a dc motor driven table system [4].

Also, in the block diagram;

$$K_m = \frac{K_t}{R_a B_e + K_t K_b} \quad (4.6)$$

which is the loaded dc motor gain, and;

$$\tau = \frac{R_a J_e}{R_a B_e + K_t K_b} \quad (4.7)$$

which is the time constant of the dc motor-ball screw-table system. Here, $J_e = J_m + M/N^2$ represents the equivalent moment of inertia of the dc motor-ball screw-table system and the moment of inertia of the ball screw is assumed to be included in J_m . Similarly, $B_e = b_m + B/N^2$ represents the equivalent viscous frictional coefficient of the dc motor-ball screw-table system.

From the block diagram, the process transfer function $G(s)$ is;

$$G(s) = \frac{V(s)}{U(s)} = \frac{K_a K_m}{iN} \frac{1}{\tau s + 1} = \frac{K_a K_m}{iN} \frac{b}{s + b} \quad (4.8)$$

where $b = 1/\tau$.

Assuming that a zero-order-hold type of digital-to-analog converter (DAC) is used, the pulse transfer function of $G(s)$ is;

$$G(z) = \frac{z-1}{z} Z \left[\frac{b}{s(s+b)} \right] \frac{K_a K_m}{iN} = \frac{K_a K_m}{iN} \frac{1 - e^{-bT}}{z - e^{-bT}} \quad (4.9)$$

From the Fig.4.2, we should be aware of the fact that the computer sends the signal $u(k)$ and gets it back as $v(k) \cdot K_e = v'(k)$, therefore we must use in our design and identification problems $v'(k)$ not $v(k)$. Thus;

$$G'(z) = \frac{V'(z)}{U(z)} = K' \frac{1 - e^{-bT}}{z - e^{-bT}} \quad (4.10)$$

where $K' = \frac{K_a K_m K_e}{iN}$ is the open loop gain.

From the pulse transfer function $G'(z)$, the difference equation of the system is obtained to be;

$$v'(k) = e^{-bT} v'(k-1) + K'(1 - e^{-bT}) u(k-1) \quad (4.11)$$

Defining the so-called parameter vector Θ and the measurement vector Φ described in chapter 3.2, Eq.(4.11) can be written as [4,5];

$$v'(k) = \theta_1 \phi_1 + \theta_2 \phi_2 = \Theta^T \Phi(k-1) \quad (4.12)$$

where $\Theta^T = [\theta_1 \theta_2]$ with $\theta_1 = \exp(-bT)$ and $\theta_2 = K'(1 - \theta_1)$, and $\Phi^T(k-1) = [v'(k-1) \quad u(k-1)]$.

4.2. Recursive Least Squares Estimation Formulation

Recalling the system difference equation;

$$v'(k) = e^{-bT} v'(k-1) + K'(1 - e^{-bT}) u(k-1) \quad (4.11)$$

where $b=1/\tau$ and $K' = \frac{K_a K_m K_e}{iN}$. One can write this equation in vector form in terms of the parameter estimates as;

$$v'(k) = \hat{\Theta}^T \Phi(k-1) \quad (4.13)$$

where

$$\hat{\Theta} = [\hat{\theta}_1 \quad \hat{\theta}_2]^T$$

$$\Phi(k-1) = [v'(k-1) \quad u(k-1)]^T$$

and real Θ is given by

$$\Theta = [\exp(-bT) \quad K'(1-\exp(-bT))]$$

RLS algorithm given by equations (3.16) and (3.17) uses $P(0)$, $\hat{\Theta}(0)$ and $\Phi(0)$. Initial covariance matrix $P(0)$ is a positive definite matrix where the diagonal terms are 10, 100, or a larger value and the off-diagonal terms are zero. As introduced in Section 3.2, UD covariance factorization algorithm [27,28] is used to mechanize the recursive least squares parameter estimation.

4.3. Control Algorithm

For adaptive velocity control, PI self-tuning regulator scheme described in Section 2.2.2 is used. Because, it is well known that PI control is adequate for all processes where the dynamics are essentially of the first order like the system governed by Eq.(4.8). In other words, the integral (I) action provides zero steady-state offset and the proportional action (P) gives a good transient response for such a system.

The transfer function of the PI controller is;

$$D(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} \quad (4.14)$$

where K_p is the proportional control coefficient, K_i is the integral control coefficient, $U(s)$ is the function of the control signal, and $E(s)$ is the function of error. To obtain the discrete-time representation of the PI controller, one can use Tustin's approximation;

$$s = \frac{2}{T} \frac{z-1}{z+1} \quad (4.15)$$

where T is the sampling period. Therefore;

$$D(z) = K_p + K_i \frac{T}{2} \frac{z+1}{z-1}$$

By organizing this equation;

$$D(z) = \frac{(K_p + \frac{TK_i}{2})z + (\frac{TK_i}{2} - K_p)}{z-1}$$

We obtain;

$$D(z) = \frac{U(z)}{E(z)} = \frac{a_0 z + a_1}{z-1} \quad (4.16)$$

where

$$a_0 = K_p + \frac{TK_i}{2}$$

$$a_1 = \frac{TK_i}{2} - K_p$$

The difference equation of digital PI controller is then;

$$u(k) = a_0 e(k) + a_1 e(k-1) + u(k-1) \quad (4.17)$$

Eq.(4.17) is called "position algorithm" in PI control since it calculates the total output value $u(k)$. By a simple manipulation on Eq.(4.17) one can write;

$$\Delta u = u(k) - u(k-1) = a_0 e(k) + a_1 e(k-1) \quad (4.18)$$

$$u(k) = u(k-1) + \Delta u$$

which is called to be "velocity algorithm" in PI control. Velocity algorithm computes a signal difference and adds this value to the previous signal. By this algorithm a smooth approach (bumpless transfer) is obtained and integral wind-up problem is prevented [11,30]. In this study, this algorithm is used to compute the control signal in adaptive velocity control.

4.4. Pole Assignment for PI Velocity Control

For a linear single input single output (SISO) system the closed loop poles not only determine the stability of the system, but their locations in the z plane determines the closed-loop system transient characteristics as well. Thus, one method for selecting controller parameters is to specify closed-loop pole locations to achieve certain desired performance characteristics. This is referred to as pole or eigenvalue assignment as introduced and formulated in Section 2.3.

From the block diagram, Fig.4.3, the characteristic equation in z domain is; $1+D(z)HG(z) = 0$ where $H(z)$ denotes the feedback element and $G(z)$ denotes the plant transfer functions. Recalling the plant pulse transfer function given by Eq.(4.9);

$$G(z) = \frac{V(z)}{U(z)} = \frac{K_{\alpha} K_m}{iN} \frac{1-a}{z-a} \quad (4.9)$$

where $a = \exp(-T/\tau)$ and $H(z) = K_e$ yields;

$$1 + D(z)HG(z) = 1 + \frac{a_0 z + a_1}{z-1} K_e \frac{K_{\alpha} K_m}{iN} \frac{1-a}{z-a} = 0$$

and for simplicity;

$$1 + D(z)HG(z) = 1 + \frac{a_0 z + a_1}{z-1} K' \frac{1-a}{z-a} = 0 \quad (4.19)$$

where

$$K' = \frac{K_{\alpha} K_m K_e}{iN}$$

After a few straightforward calculations, we obtain the characteristic equation in the polynomial form;

$$z^2 + [-a-1+K'(1-a)a_0]z + [a+K'(1-a)a_1] = 0 \quad (4.20)$$

where $b=1/\tau$ and $a=\exp(-bT)$.

The desired characteristic equation will then be in the form of [11];

$$z^2 + c_1 z + c_2 = 0 \quad (4.21)$$

where

$$c_1 = -2\exp(-\zeta\omega_n T)\cos(\sqrt{1-\zeta^2}\omega_n T) \quad (4.22)$$

$$c_2 = \exp(-2\zeta\omega_n T)$$

in which ζ denotes the damping factor, ω_n denotes the natural frequency, and T is the sampling interval. This equation may readily be obtained by sampling (at intervals T) a continuous-time system with the characteristic equation $s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$.

Therefore, pole-assignment method will be used so as to find the regulator parameters K_p and K_i in terms of the plant parameters and the desired performance characteristics. The poles are simply assigned to the desired locations in the z plane by simply equating Eq.(4.20) and Eq.(4.21). Thus, we obtain;

$$a_0 = \frac{c_1 + a + 1}{K'(1-a)}, \quad a_1 = \frac{c_2 - a}{K'(1-a)} \quad (4.23)$$

Further recalling $a_0 = K_p + (TK_i/2)$ and $a_1 = (TK_i/2) - K_p$ the regulator parameters are found to be;

$$K_p = \frac{c_1 - c_2 + 1 + 2a}{2K'(1-a)} \quad (4.24)$$

$$K_i = \frac{c_1 + c_2 + 1}{TK'(1-a)}$$

where $a_0 = \exp(-T/\tau)$, $K' = (K_a K_m K_e / iN)$ and c_1 and c_2 are given by Eq.(4.22). Eq.(4.24) gives the pole assignment results for PI

control of the plant given by Eq.(4.11).

According to the self-tuning adaptive control procedure shown in Fig.2.3, Eq.(4.22) forms the block labeled "design calculations" with the modification in which the recursive estimates of the process parameters are used to obtain the regulator parameters K_p and K_i at each sampling interval. Therefore, in PI self-tuning adaptive control of the system, the regulator parameters are;

$$K_p = \frac{c_1 - c_2 + 1 + 2\hat{\theta}_1}{2\hat{\theta}_2} \quad K_i = \frac{c_1 + c_2 + 1}{T\hat{\theta}_2} \quad (4.25)$$

where $\hat{\theta}_1 = \exp(-T/\tau)$ and $\hat{\theta}_2 = K'(1-\hat{\theta}_1)$, and $\hat{\theta}_1$ and $\hat{\theta}_2$ denotes the recursive estimates of the process parameters θ_1 and θ_2 .

In this study, at the beginning, and therefore in the simulations, the time-varying parameter (in adaptive control) was chosen to be the equivalent moment of inertia J_e [4]. The variations in the equivalent moment of inertia would then affect the time constant of the system according to Eq.(4.7), and the variations in the time constant affect the process parameters θ_1 and θ_2 . However, due to the limitations arising from the mechanical constructions it was impossible to change J_e in considerable amounts. Namely, the reduction ratio between the table and the DC motor makes the system insensitive to load inertia changes. On the other hand, the nonlinear and random friction resulting from the guide rods makes the system inherently time-variant [5]. Hence, these variations are followed by the RLS estimation algorithm which is described in Chapter III. At each step, the recursive

estimates for θ_1 and θ_2 , namely $\hat{\theta}_1$ and $\hat{\theta}_2$ are used to obtain the optimal K_p and K_i values.

The values c_1 and c_2 appearing in Eq.(4.26) show the nature of the desired dynamics. According to Eq.(4.22), the desired dynamics are given in terms of the natural frequency ω_n and the damping ratio ζ . Also, the desired dynamics can be given in terms of the maximum overshoot and the settling time. In such a case, natural frequency and damping ratio values are determined according to the following set of equations [31,32];

$$\zeta = \frac{\ln^2(t_p/100)}{\pi^2 + \ln^2(t_p/100)} \quad (4.26)$$

$$\omega_n = \frac{4}{\zeta t_s} \quad (\text{for 2\% settling}) \quad (4.27)$$

where t_p is the maximum overshoot and t_s is the settling time.

In the thesis, principally $\zeta=0.826$ and $\omega_n=6.456$ rad/sec, in other words 0.75 sec. settling time and an overshoot of 1% have been used [5]. This choice implies that the closed-loop poles in z-plane are chosen to be $z_{1,2} = 0.872 \mp i(0.08)$.

V. ADAPTIVE POSITION CONTROL OF THE DC MOTOR DRIVEN TABLE SYSTEM

5.1. Mathematical Modeling of the Position Controlled System

The block diagram representation of the system is similar to the block diagram of the velocity controlled system, Fig.4.3. The exception is only the integrator $1/s$. Thus, the block diagram of the position controlled system is given in Fig.5.1.

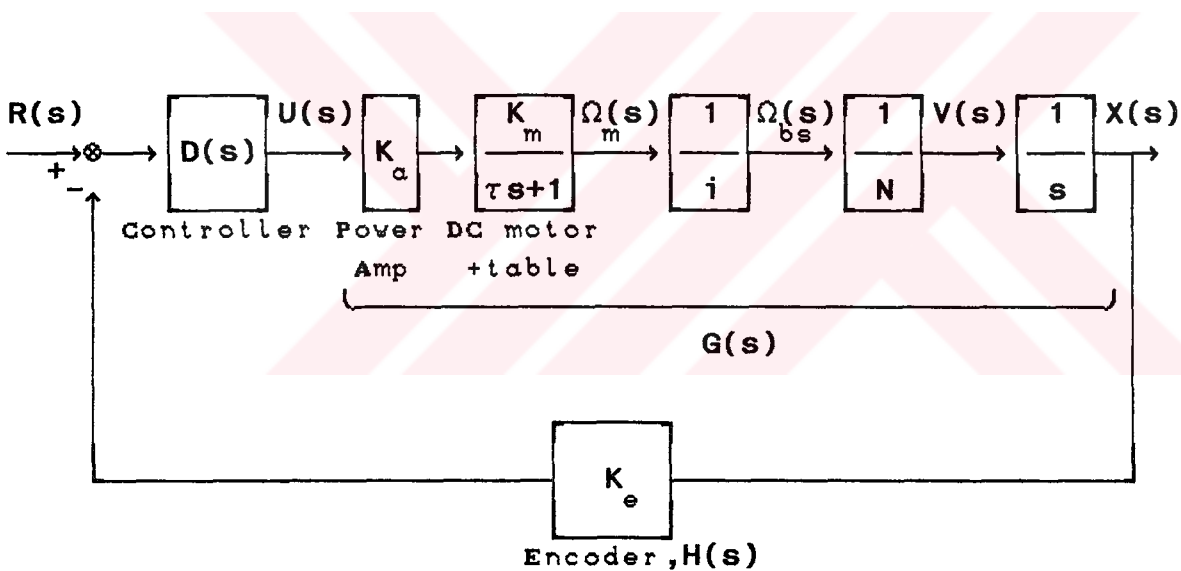


Fig.5.1. Block diagram of position control of a dc motor driven table system.

According to Fig.5.1, the process transfer function is;

$$G(s) = \frac{X(s)}{U(s)} = \frac{K_\alpha K_m b}{i N s (s + b)} \quad (5.1)$$

where $b = 1/\tau$. Assuming that a zero-order-hold type of DAC is

used, the pulse transfer function of $G(s)$ is [33];

$$G(z) = \frac{K_a K_m}{iN} \frac{(bT-1+a)z+(1-a-baT)}{(z-1)(z-a)} \quad (5.2)$$

where $b=1/\tau$, $a=\exp(-bT)$, and T is the sampling period.

As stated in Section 4.1, the computer sends the signal $u(k)$ and gets it back as $x(k) \cdot K_e = x'(k)$, therefore we must use $x'(k)$ being the output. Thus, the pulse transfer function becomes;

$$G'(z) = \frac{X'(z)}{U(z)} = K' \frac{\alpha z + \beta}{(z-1)(z-a)} \quad (5.3)$$

where

$$K' = \frac{K_a K_m K_e}{iNb}$$

$$\alpha = bT - 1 + a$$

$$\beta = 1 - a - baT$$

$$a = \exp(-bT)$$

$$b = 1/\tau$$

From the pulse transfer function $G'(z)$, the difference equation of the position controlled system is obtained;

$$x'(k) = (a+1)x'(k-1) - ax'(k-2) + K'\alpha u(k-1) + K'\beta u(k-2) \quad (5.4)$$

Defining the parameter vector Θ and the measurement vector Φ described in Section 3.2, Eq.(5.4) can be written in the compact form;

$$x'(k) = \theta_1 \phi_1 + \theta_2 \phi_2 + \theta_3 \phi_3 + \theta_4 \phi_4 = \Theta^T \Phi(k-1) \quad (5.5)$$

where $\Theta^T = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4]$ with

$$\begin{aligned} \theta_1 &= a+1 \\ \theta_2 &= -a \\ \theta_3 &= K'\alpha \\ \theta_4 &= K'\beta \end{aligned}$$

and $\Phi^T(k-1) = [x'(k-1) \quad x'(k-2) \quad u(k-1) \quad u(k-2)]$.

5.2. Pole-Assignment for Adaptive Position Control

According to the pole-placement design algorithm described in Section 2.3, let the process, Eq.(5.3), be;

$$H(z) = \frac{B(z)}{A(z)} = K' \frac{\alpha(z-d)}{(z-1)(z-a)} \quad (5.6)$$

where $d = -\beta/\alpha$

Let the desired closed-loop transfer function in the form of;

$$H_m(z) = \frac{B_m(z)}{A_m(z)} = \frac{(z-d)(1+c_1+c_2)}{(z^2+c_1z+c_2)(1-d)} \quad (5.7)$$

where

$$c_1 = -2\exp(-\zeta\omega_n T)\cos(\sqrt{1-\zeta^2}\omega_n T)$$

$$c_2 = \exp(-2\zeta\omega_n T)$$

Notice that the process zero on the negative real axis is now also a zero of the desired closed-loop transfer function. This means that the zero does not have to be canceled by the regulator.

According to the pole-placement algorithm, $B(z)$ is factorized in the following way;

$$B = B^+ \cdot B^-$$

where

$$B^+ = 1 \quad (5.8)$$

$$B^- = \alpha K'(z-d) \quad (5.9)$$

Here, B^+ has the process zero inside the unit circle, and B^- has the zero outside the unit circle.

Further B'_m is defined by $B'_m = B_m/B^-$, therefore;

$$B'_m = \frac{(1+c_1+c_2)}{\alpha K'(1-d)} \quad (5.10)$$

According to the causality condition the degree of the observer polynomial A_0 is given by [2,11];

$$\deg A_0 \geq 2\deg A - \deg A_m - \deg B^+ - 1.$$

This gives $\deg A_0 \geq 1$. We choose $A_0(z)=z$ for simplicity.

The degree of the polynomials R' and S are given according to the causality conditions by [2,11];

$$\deg R' = \deg A_o + \deg A_m - \deg A$$

$$\deg S = \deg A - 1$$

These relations imply $\deg R'=1$ and $\deg S=1$. Thus, we choose;

$$R' = z + r_1 \quad (5.11)$$

$$S = s_o z + s_1 \quad (5.12)$$

According to the so-called Diophantine equation

$$A R' + B S = A_o A_m \quad (2.4)$$

the following polynomial identity is obtained;

$$(z-1)(z-a)(z+r_1) + \alpha K'(z-d)(s_o z + s_1) = z(z^2 + c_1 z + c_2) \quad (5.13)$$

Equating the coefficients, and using the relation given by Eq.(2.5), the regulator parameters are found to be;

$$r_1 = -d + \frac{d^3 + c_1 d^2 + c_2}{(d-1)(d-a)}$$

$$t_o = (1 + c_1 + c_2) / (\alpha K'(1-d))$$

$$s_o = \frac{(1 + c_1 + c_2)(a-d) - (1-d)(a^3 + a^2 c_1 + a c_2)}{\alpha K'(1-a)(1-d)(a-d)}$$

$$s_1 = t_o - s_o \quad (5.14)$$

According to the pole assignment algorithm with no cancellation of process zero given in Section 2.3, the regulator is given by;

$$R \cdot u = T \cdot u_c - S \cdot y \quad (2.3)$$

where $R=B^+R'$, $T=A_0 B'_m$, u_c is the reference input y is the output. Therefore, the regulator is given by;

$$(z+r_1)U(z) = t_0 z \cdot X_R(z) - (s_0 z + s_1)X'(z) \quad (5.15)$$

The difference equation form of the pole placement explicit self tuning regulator for adaptive position control is then;

$$u(k) = t_0 X_R(k) - s_0 X'(k) - s_1 X'(k-1) - r_1 u(k-1) \quad (5.16)$$

where t_0, r_1, s_0 and s_1 are given by Eq.(5.14), $u(k)$ denotes the regulator control signal, $X_R(k)$ denotes the reference position, and $X'(k)$ denotes the current position.

According to the self-tuning adaptive control procedure shown in Fig.2.3, the recursive estimates of the process parameters a_1, a_2, b_1 , and b_2 should be used at each sampling period to control the regulator parameters. Therefore, in terms of the recursive parameter estimates, the regulator is given by;

$$u(k) = \hat{t}_0 X_R(k) - \hat{s}_0 X'(k) - \hat{s}_1 X'(k-1) - \hat{r}_1 u(k-1) \quad (5.17)$$

where

$$d = -\hat{b}_2 / \hat{b}_1$$

$$\hat{r}_1 = -d + \frac{d^3 + c_1 d^2 + c_2}{(d-1)(d-a)}$$

$$\hat{t}_0 = (1 + c_1 + c_2) / (\hat{b}_1 (1-d))$$

$$\hat{s}_0 = \frac{(1 + c_1 + c_2)(a-d) - (1-d)(a^3 + a^2 c_1 + a c_2)}{b_1 (1-a)(1-d)(a-d)}$$

$$\hat{s}_1 = \hat{t}_0 - \hat{s}_0 \quad (5.18)$$

in which the estimate vector $\hat{\Theta}^T = [\hat{a}_1 \quad \hat{a}_2 \quad \hat{b}_1 \quad \hat{b}_2]$ is calculated by the recursive least squares estimator at each step.

VI. EXPERIMENTAL SET-UP, ANALYSIS AND RESULTS

6.1. Experimental Set-Up

The purpose of this research was to realize real-time adaptive control of a dc motor driven single axis table. The experimental set-up, which is constructed to accomplish this, consist of a dc motor that drives the table, an incremental linear encoder as feedback unit, an interface card, a power amplifier, and finally, an AT type 33 MHz microcomputer with built in mathematical coprocessor. The layout of the experimental set-up is presented in Fig.1.1.

In order to convert the rotational motion of the dc motor into linear motion a ball screw was employed. The use of the ball screw ceased friction, prevented vibration, and improved efficiency by providing smooth action. Additionally, in the experimental set-up, to prevent vibration and misalignment, carbon steel plates and hardened steel guides of safe dimensions were used. Another possible source of friction between the moving table and guides was eliminated by using one linear ball bearing on each guide [34].

6.1.1. Linear Encoder

An incremental linear encoder has three basic parts, a light source, a scaling glass and a sensor (photovoltaic cell, phototransistor, photodiode). In this research, a "Dr.

Johannes Heidenhain GmbH-LS 707" incremental linear encoder was used. Its specifications are [35];

- i. Scanning principle: Photoelectric transmitted light.
- ii. Scale: Glass scale with DIADUR grating.
- iii. Grating pitch: 20 μm
- iv. Light source: Long life miniature lamp prefocused, 5V/0.6W
- v. Scanning elements: Silicon solar cells in push pull arrangement.
- vi. Maximum permissible speed: 24 m/min
- vii. Permissible acceleration: 30 m/s^2
- viii. Required feeding power: $\leq 10\text{N}$
- ix. Output signals of transducer: 2 sine-wave signals, I_{e1} and I_{e2} phase shifted by 90 degrees.
- x. Supply voltage: 5 Volt (lamp)
- xi. Weight of transducer: 0.8 kg + 2 kg/m measuring length.

For most purposes the low signal levels obtained directly from the encoder optoelectronic sensors are not adequate for control or signal processing. Therefore, amplifiers and wave form shapers are incorporated into the encoder packages. However, we did not have this package. Thus, to convert the $10\mu\text{A}$ sinusoidal signal to a 0-5V square wave signal an electronic circuit has been designed and constructed.

6.1.2. Ball Screw

Ball screws are one of the high-tech products used in modern CNC machines. The use of steel balls between nut and screw minimizes friction and provides smooth action. These screws have efficiencies up to 95%. The choice of lead angle of this screw is such that no self-locking exists.

In this research a Steinmeyer Ref. Nr. 1530/4.12.700.700 ball screw was used and all the mechanical assembly was constructed as a ME 492 Project by C. Akçadağ and S. Ünsal in 1990. The specifications of the ball screw are [34];

i.Nut: $D_o = 25.4$ mm., 24.1 mm. long

ii.Balls: $D = 2.03$ mm.

iii.Pitch = 4 mm.

iv. D_o of the screw = 12 mm

v.Total length = 700 mm

6.1.3. Interface Card and Power Amplifier

The dc motor interface card was designed to do various tasks for computer controlled dc systems [36]. It is plugged into one of the computer slots and has the following functions: address decoding, clock adjustment, PWM generation, speed reading, interrupt generation and position counting. In realizing these experiments, except interrupt generation, all the parts of the interface card have been used. Both the pulses from the encoder and separate high frequency clock were counted in a predetermined time interval, and the speed found by division.

The power amplifier employed is a PWM (pulse width modulated) device. The period of the output signal is constant, but the pulse length changes according to the control signal. The PWM signal and the direction of rotation is fed to the dc motor drive circuit which is also fed by 15 volt regulated power supply. According to the PWM signal, the driver voltage is supplied and fed to the dc motor.

6.2. Computer Programming

In this study, four different real-time control programs have been written in Pascal.

The program PIVEL performs classical PI control of the table velocity. In this program, PI velocity algorithm was implemented and the controller parameters were tuned by using Ziegler-Nichols method.

The program ADAPVEL performs self-tuning adaptive PI velocity control of the DC motor driven single axis table according to the formulation presented in Chapter IV. In this program, PI velocity algorithm using Tustin's approximation has been used. The controller parameters were updated by the procedure LS which performs RLS parameter estimation at each sampling period.

The program PAPDPOS performs pole assignment PD position control of the table. In the program, the procedure GAUSSEELIMINATION solves the matrix equation which is needed

to assign the poles to the desired locations.

Finally, the program ADAPPOS performs the self-tuning adaptive position control of the DC motor driven single axis table according to the formulation given in Chapter V. In this program, algorithm 3 of Section 2.3 has been used. Again, parameters were updated by the procedure LS.

The procedures InitPPI_Cnt, Power_Up, and Power_Down are similar for all those above mentioned programs. The procedure InitPPI_Cnt initializes 8255 and 8253. The Power_Up procedure resets and enables the interface card. Similarly, the Power_Down procedure disables the interface. The detailed information can be found in [36].

6.3. Experimental Analysis

An AT type microcomputer of 33 MHz clock frequency with a built in mathematical coprocessor has been used to control the servo system shown in Fig.4.3 for velocity control and Fig.5.1 for position control of the table.

In the preliminary experiments, first, armature resistance, the torque and voltage constants of the DC motor, and equivalent viscous frictional coefficient of the experimental system were found. Then, the step response tests were conducted to determine the open-loop gain and time constant of the system. These were roughly found to be as

$$K_e = K_t = 0.07037 \text{ V/rad/sec.}$$

$$R_a = 1.3 \Omega$$

$$B_e = 1.442 \cdot 10^{-4} \text{ Nm/rad/sec.}$$

$$K' = 3.8 \cdot 10^{-4} \text{ m/s/V}$$

$$\tau = 40\text{ms.}$$

For the subsequent experiments, the sampling time of $T=25\text{ms}$ was chosen.

6.4. Experimental Results

6.4.1. Adaptive Velocity Control

For the adaptive velocity control experiments, the sampling time of $T=25\text{ms}$ and the reference speed of the table of $v=0.03\text{m/s}$ were chosen. During the experiments it was observed that the speed measurement was quite noisy, mainly due to the poor mechanical construction of the set-up.

The performance of the estimator and controller depends seriously upon the estimator parameters. Hence a number of experiments were implemented to determine the effects of initial estimates, forgetting factor and covariance matrix on the whole adaptive procedure and to choose their "best" values.

i) In Fig.6.1.(a) and (b), the self-tuning controller is started for two different sets of initial estimates, namely for $J_e(0)=10^{-5} \text{kgm}^2$ and $J_e(0)=10^{-2} \text{kgm}^2$. The low value of $J_e(0)$ causes a slow response and a high value leads to a faster but oscillatory response at the start-up. It was observed that in both cases the estimates θ_1 and θ_2 converged to the same values nearly at the same time.

ii) The forgetting factor λ of RLS-UD algorithm is a parameter which determines how fast the previous data is forgotten. It is usually chosen around $\lambda=0.95$. It is seen in Fig.6.2.(a) and (b) that due to noisy measurement of speed, the lower value of $\lambda=0.93$ caused higher fluctuation of the estimate θ_1 . On the other hand, the higher value of $\lambda=0.99$ caused extremely small convergence to the average value of $\theta_1=0.55$. However, the implementation of a variable or directional forgetting factor are possible methods to overcome the difficulties arising from the choice of the forgetting factor [24,25,26,37]. In Fig.6.3, the desired overshoot and settling time are set to 10% and 1s, respectively. With $\lambda=0.99$ the desired overshoot and settling time are achieved. However, $\lambda=0.99$ an overshoot of 6% is reached which is less than the desired value. This discrepancy is expected to be due to the inadequate modeling of damping between the table and the guide rods.

iii) The AC is also affected by the choice of initial covariance matrix. For matrix $P(0)$, one often selects $P(0)=\rho I$ where I is the identity matrix and ρ is a design parameter which is greater than zero. Setting ρ to a low value of $\rho=100$ eliminates the sudden jump of the estimate θ_1 at the start-up, but it causes the speed response to have a maximum

overshoot far from the desired value of 1% (Fig.6.4.(a) and (b)).

Based upon the experimental observations, the forgetting factor and the initial covariance matrix diagonal elements are set to $\lambda=0.96$ and $\rho=10000$, respectively, to achieve a desired speed response with 1% overshoot and 0.75s settling time.

The theoretical and experimental speed response is shown in Fig.6.5.(a). It is seen that the experimental response takes about 1s to reach the desired reference value with approximately 0.20s delay. This delay is mainly due to fair mechanical construction and partially due to a sticktion of the motor at the start-up. Moreover, it is observed that the theoretical and experimental overshoots do not agree well. This can be explained by the effect of nonlinear frictional damping which exists in the experimental system, but not included in the simulation model. It is seen that the experimental self-tuning adaptive controller (STAC) follows quite well the desired performance criteria. Additionally the comparison of STAC response with that of the fixed PI controller (Fig.6.5.(b)), tuned by using Ziegler-Nichols method, reveals that the adaptation algorithm provides a faster and smoother response in this experimental set-up.

The corresponding parameter estimates θ_1 and θ_2 illustrated in Fig.6.6 converges at approximately 6 seconds. The fluctuation of the estimates is explained by the mechanical noise arising from the misalignment of the guide rods. Using the average values of $\theta_1=0.55$ and $\theta_2=1.6*10^{-4}$ of Fig.6.6.(a) and (b), the gain and the open-loop time constant

of the system are obtained as 3.6×10^{-4} m/s/V and 42ms, respectively. These values are approximately equal to the ones obtained from the open-loop step response of the system, i.e., 3.8×10^{-4} m/s/V and 40ms. These results validate the estimation procedure.

It is known that the excitation signal is a factor which affects the quality of the estimates in most identification procedures [9]. The test signal should preferably be rich with flat spectrum, and should persistently excite all the system modes. The so called Schroeder Phased Harmonic Sequence (SPHS) which is shown in Fig.6.7.(a), providing such features, was injected as input to the open-loop plant while the table was at a given point on the rod. The resulting parameter estimate θ_1 is plotted in Fig.6.7.(b). It is observed that the estimate θ_1 converges in 2s to the approximate value of $\theta_1=0.57$ which is also nearly the mean value of the fluctuating estimate θ_1 of the STAC [5].

6.4.2. Adaptive Position Control

In the position control of the DC motor driven single axis table two algorithms were used: the ordinary PD pole assignment control algorithm and the pole assignment self-tuning adaptive control algorithm.

For the experiments, sampling time of 25ms and reference position of 0.05m were chosen.

The effect of the estimator parameters on the

estimator and the controller performance was examined first. It was observed that the estimation procedure did not work well, since the settling time for the position was too short and after the table settles no more useful information could be transferred to the estimation algorithm. Thus, to obtain better estimates and consistent convergence, a persistently exciting signal was needed. However, the trial to implement a SPHS to the algorithm has failed. To sum up, some effects of the initial estimate, forgetting factor and initial covariance matrix on the parameter estimates were observed, but due to the inconsistent data, no precise results could be obtained. The inconsistency and unreliability of the parameter estimates can be seen in Figures 6.8, 6.9, and 6.10.

Secondly, the effect of estimator parameters on the system response was determined. It was observed that the effects of forgetting factor and initial covariance matrix on the system response were not remarkable. The effect of forgetting factor and initial covariance matrix are illustrated in figure 6.11.(a)-(b) and 6.13, respectively. On the other hand, the effect of initial estimate was found to be drastic. High value of equivalent load of inertia causes the system settle in 2s and low value of it results in overshoot percentage of higher than the desired value of one percent (Figure 6.12).

Finally, in position control experiments, ordinary PD pole assignment and adaptive algorithms were compared. In Figure 6.14.(a), with good initial estimate, it is seen that adaptive algorithm is superior, since it eliminates the steady state error which is encountered in ordinary pole

assignment position control. Moreover, when the initial estimate is bad, that is to say, the system characteristics are totally unknown, adaptive algorithm still provides the table to reach the reference value. However, the ordinary PD pole assignment algorithm causes oscillation under the same conditions (Figure 6.14.(b)).



VII. CONCLUSIONS AND RECOMMENDATIONS

In this thesis, a microcomputer based adaptive controller using self-tuning pole assignment technique together with recursive least squares estimation algorithm for system identification was investigated experimentally on a laboratory model of a DC motor driven single axis table.

i) Velocity Control: The misalignment of guide rods and unmodelled friction damping affected adversely the estimator and controller performances. The steady-state mean value of the recorded parameter estimates of the adaptive velocity control experiments were found approximately to be equal to the ones obtained from the preliminary experiments. The effects of the initial estimates, the forgetting factor and the initial covariance matrix on the estimates and the speed response dynamics have been examined. It was noted that;

*The effect of forgetting factor on the convergence time of the estimates is more significant than its effect on the performance of the response.

*On the other hand, the effect of initial estimates on the performance of the speed response is important, whereas initial estimates have no effect on the parameter estimates.

*However, the initial covariance matrix affected both the performance of the response and the parameter estimates

considerably.

It was observed that the AC algorithm performed better than the fixed parameter PI controller. The slight time-variance of the system plant parameters originating from the mechanical construction were followed by the on-line recursive estimation procedure.

ii) Position Control: In the position control experiments, the performances of the ordinary PD pole assignment and adaptive algorithms were examined and compared. It was observed that;

*No consistent and reliable results were obtained when the parameter estimates and the effects of initial estimate, forgetting factor and initial covariance matrix on the parameter estimates are considered. This is because of the lack of persistent excitation in the system.

*The effects of forgetting factor and initial estimates on the system response were not considerable whereas the effect of initial estimate was quite remarkable.

*The comparison of classical and adaptive pole assignment algorithms revealed that the latter was superior when compared to the former.

Summing up, in this thesis an experimental investigation of adaptive servo control of single axis table has been done. The overall results showed that in this experimental set-up, since the system has inherently time-variant characteristics, the use of adaptive control is

advantageous. On the other hand, the need for adaptation algorithm in both the position and the velocity control is controversial. Before attempting to use adaptive control, it is important to investigate whether the control problem might be solved by constant-gain feedback, since in the literature on adaptive control there are many cases where constant-gain feedback can do as well as an adaptive regulator [2].

For further research on this subject, several recommendations can be made. First, it was noted that the effects of forgetting factor on parameter estimates and initial estimate on the speed response in adaptive PI velocity control were quite remarkable. Thus, to automatically choose the forgetting factor and the initial estimate is important. To automate the forgetting factor, a directional or variable forgetting factor algorithm can be added to the estimation program. Additionally, to overcome the difficulties arising from the bad initial estimate, an off-line frequency domain estimation procedure can be implemented which uses the experimental block data. By doing this, reasonable initial estimates can be obtained.

On the other hand, to improve the experimental results, some parts of the mechanical hardware should be reconstructed. The placement of the encoder can be changed and the guide rod friction can be reduced or at least homogenized. For example, to replace the encoder in the middle of the table is a better solution. Moreover, using direct drive rather than the power transmission with the help of a timing belt, and a more sensitive DC servo motor can be other recommendations to improve the experimental setup.



APPENDIX A

**Listings of the Classical and Adaptive
PI Velocity Control Programs**


```

{$R+}
Program PIVEL;
(* Classical PI Velocity Control *)
uses graph,dos,crt;
Const
  PortA_Addr = $340;    { 8255 Ports }
  PortB_Addr = $341;
  PortC_Addr = $342;
  Cont_8255   = $343;

  Cnt0_Addr  = $348;    { 8253 Ports }
  Cnt1_Addr  = $349;
  Cnt2_Addr  = $34A;
  Cont_Reg   = $34B;

  PosL_Addr  = $350;
  PosH_Addr  = $358;

Var
  NextReady : word;    {if the controller has calculated the next output}
  CalcLate  : word;
  LastDir   : word;    {the direction of rotation, 1=clockwise }
  Pulses    : word;    {# of encoder pulses in the measurement interval }
  Secs      : word;    {# of MMLOCK pulses in the measurement interval }
  Position  : word;

{$I graf.inc}
{$L endcheck.obj}

Procedure EndCheck ; external;

var
  pass,lob,hib:byte;
  c:char;
  posmax,spmax,posmin,spmin:real;
  nper:word;
  old,
  oldspeed,tsam,speed,absspeed,u,err,Kp,Ki:real;
  sphist,poshist:array[0..2000] of real;
  incr,i:integer;
  refspeed,ur,du,vel,vel1:real;
  st:string[10];
  file1,file2:text;

Procedure InitPPI_Cnt;
{ Configures 8255 and 8253 }
begin
  port[Cont_8255]:=$8A; {A inmode0,output.B input.C Hiinput,C Lo output}
  port[cont_8255]:=3;   { reset card }

  port[Cont_Reg]:=$36;   { Counter 0 in square wave mode   }
  port[Cnt0_Addr]:=$8B;  { set sampling period }
  port[Cnt0_Addr]:=$02;  { first LSB then MSB }
  tsam:=10;
  { clk period=15.36 micro secs }
  { measurement clk frequency 1.0416 MHz }

```

```
port[Cont_Reg]:= $70;      { Counter 1 in Mode 0 }
port[Cnt1_Addr]:= $FF;
port[Cnt1_Addr]:= $FF;
```

```
port[Cont_Reg]:= $B0;      { Counter 2 in Mode 0 }
port[Cnt2_Addr]:= $FF;
port[Cnt2_Addr]:= $FF;
```

```
end;
```

```
procedure power_up;
```

```
begin
  port[cont_8255]:= $03;    { set port C1 to reset card }
  port[cont_8255]:= $00;    { clear position cnt , toggle C0 }
  port[cont_8255]:= $01;

  { clear counters }
  port[cont_reg]:= $B0;     { load cnt2 }
  port[cnt2_addr]:= $FF;    { LSB }
  port[cnt2_addr]:= $FF;    { MSB }
  port[cont_reg]:= $70;     { load cnt1 }
  port[cnt1_addr]:= $FF;    { LSB }
  port[cnt1_addr]:= $FF;    { MSB }

  port[cont_8255]:= 7;      { set port C3 for position latch signal }
  port[cont_8255]:= $02;    { reset port C1 to start card }
end;
```

```
Procedure Power_Down;
```

```
begin
  port[portA_addr]:= 0;
  port[cont_8255]:= 7;
  delay(round(tsam));
  port[cont_8255]:= $03;    { set port C1 to reset card }
end;
```

```
Procedure NewSpeed(rsp:real);
```

```
{if sp>0 then positive direction ,else negative }
```

```
var
```

```
sp:integer;
```

```
begin
  sp:=round(rsp);
  port[portA_addr]:= 0;
  if sp>255 then sp:=255
  else if sp<-255 then sp:=-255;
  if sp>=0 then port[cont_8255]:=5
  else port[cont_8255]:=4;
  port[portA_addr]:=abs(sp);
end;
```

```
begin      { main }
```

```
  initppi_cnt;
  clrscr;
  writeln('refspeed?');
  readln(refspeed);
  nper:=0;
  spmax:=-1e30;
```

```

posmax:=-1e30;
spmin:=1e30;
posmin:=1e30;
for i:=0 to 2000 do
begin
  sphist[i]:=0;
  poshist[i]:=0;
end;
err:=0;
u:=0;
vel:=0;
vel1:=0;
assign(file1,'pisp.dat');
assign(file2,'iconsig.dat');
rewrite(file1);
rewrite(file2);
if (refspeed>0) then port[cont_8255]:=5
                    else port[cont_8255]:=4;

power_up;
repeat

  EndCheck; { wait until the end of sampling period }
  speed:=0;
  if (LastDir=0) and (secs>0) then
    speed:=20.832*Pulses/Secs
  else if secs>0 then
    speed:=-20.832*Pulses/Secs; { in cm/s }
  abs speed:=abs(speed);

  if speed>spmax then spmax:=speed;
  if position>posmax then posmax:=position;
  if speed<spmin then spmin:=speed;
  if position<posmin then posmin:=position;
  sphist[nper]:=speed;
  poshist[nper]:=Position;

  if nper>4 then sphist[nper]:=0.25*(sphist[nper]+sphist[nper-1]+
                                sphist[nper-2]+sphist[nper-3]);
  writeln(file1,sphist[nper]);

{ CONTROL SIGNAL CALCULATION }

  if (nper mod 20)=0 then
  begin
    gotoxy(3,3);
    writeln(nper,' ',pulses,' ',secs,' ',speed:7:2,' ');
  end;
  vel:=sphist[nper];
  err:=refspeed-vel;
  Kp:=3.5;
  Ki:=0.15;
  du:=Kp*(vel1-vel)+Ki*err;
  vel1:=vel;
  u:=u+du;
  ur:=2550*u;
  newspeed(ur);
  writeln(file2,ur);
  if nper<2000 then inc(nper);
until keypressed or (nper>=2000);

```

```

{ CONTROL SIGNAL CALCULATION END}

power_down;
nper:=0;
incr:=1;
while nper<=2000 do
begin
  old:=trunc(1.0*poshist[nper+1]-poshist[nper]);
  writeln(nper:7,'      ',sphist[nper]:7,'      ',poshist[nper]:7,'      ',
          old:5:2,'      ');
  if (nper mod 19)=0 then
  begin
    writeln('E to end');
    c:=readkey;
    if upcase(c)='E' then nper:=2001;
    if upcase(c)='I' then incr:=20;
    if upcase(c)='D' then incr:=1;
  end;
  inc(nper,incr);
end;

writeln;
writeln('Graph?');
readln(c);
pass:=0;
if (c='y') or (c='Y') then
while (c='y') or (c='Y') do
if (pass+1)*720<2000 then
begin
  for i:=0 to 720 do
  begin
    fn[1][i]:=round(sphist[i+pass*720]);
    fn[2][i]:=round(poshist[i+pass*720]);
  end;
  ymin[2]:=posmin;
  ymax[2]:=posmax;
  ymin[1]:=spmin-2;
  ymax[1]:=spmax+2;
  xmin:=pass*720;
  xmax:=(pass+1)*720;
  axinf[0].name:='Time';
  str(tsam,st);
  axinf[0].axunit:='('+st+'ms)';
  axinf[1].name:='Speed';
  axinf[1].axunit:='(';
  axinf[2].name:='Position';
  axinf[2].axunit:='(';
  grapher(fn, ymin,ymax,xmin,xmax,2,720,
          'Speed vs Time',axinf,true);
  pass:=pass+1;
  writeln;
  writeln('Graph?');
  readln(c);
end;

close(file1);
close(file2);
end.

```

```

{$R+}
PROGRAM ADAPVEL;
(* Self-Tuning Adaptive PI Velocity Control *)
USES
    graph,dos,crt;
CONST
    npar=2;
    noff=1;
TYPE
    vec1=ARRAY[1..npar] OF real;
    vec2=ARRAY[1..noff] OF real;
VAR
    km,tauest,denom,kprime,natlog :real;
    speed,ainitial,c1,c2,ddenom,kp,ki,settlingtime :real;
    ka,ke,iii,N,tsamp,tsam,rho,vref,zeta,wn :real;
    jeo,ra,lambda,overshoot :real;
    kt,kb,be,initialestje,binitial :real;
    index,M :integer;
    v,u,sphs :ARRAY[0..2000] OF real;
    aaa,ek,bbb,ccc,ek_1 :real;
    PAR1,PHI,diag :vec1;
    offdiag :vec2;
    deltau,jeest :real;
    Kad :real;
    file1,file2,file3 :text;

Const
    PortA_Addr = $340;    { 8255 Ports }
    PortB_Addr = $341;
    PortC_Addr = $342;
    Cont_8255   = $343;

    Cnt0_Addr  = $348;    { 8253 Ports }
    Cnt1_Addr  = $349;
    Cnt2_Addr  = $34A;
    Cont_Reg   = $34B;

    PosL_Addr  = $350;
    PosH_Addr  = $358;

Var
    NextReady : word;    { if the controller has calculated the next output}
    CalcLate  : word;
    LastDir   : word;    { the direction of rotation, 1=clockwise      }
    Pulses    : word;    { # of encoder pulses in the measurement interval }
    Secs      : word;    { # of MMCLOCK pulses in the measurement interval }
    Position  : word;

{$I graf.inc}
{$L endcheck.obj}

Procedure EndCheck ; external;

Procedure InitPPI_Cnt;
{ Configures 8255 and 8253 }
begin

```

```

port[Cont_8255]:=$8A; {A inmode0,output.B input.C Hiinput,C Lo output}
port[cont_8255]:=3;   { reset card }

port[Cont_Reg]:=$36;   { Counter 0 in square wave mode }
port[Cnt0_Addr]:=$5C;  { set sampling period }
port[Cnt0_Addr]:=$06;  {first LSB then MSB }
tsamp:=25;
{ clk period=15.36 micro secs }
{ measurement clk frequency 1.0416 MHz }

port[Cont_Reg]:=$70;   { Counter 1 in Mode 0 }
port[Cnt1_Addr]:=$FF;
port[Cnt1_Addr]:=$FF;

port[Cont_Reg]:=$B0;   { Counter 2 in Mode 0 }
port[Cnt2_Addr]:=$FF;
port[Cnt2_Addr]:=$FF;

end;

procedure power_up;
begin
port[cont_8255]:=$03;   { set port C1 to reset card }
port[cont_8255]:=$00;   { clear position cnt , toggle C0 }
port[cont_8255]:=$01;

{ clear counters }
port[cont_reg]:=$B0;    { load cnt2 }
port[cnt2_addr]:=$FF;   { LSB }
port[cnt2_addr]:=$FF;   { MSB }
port[cont_reg]:=$70;    { load cnt1 }
port[cnt1_addr]:=$FF;   { LSB }
port[cnt1_addr]:=$FF;   { MSB }

port[cont_8255]:=7;     { set port C3 for position latch signal }
port[cont_8255]:=$02;   { reset port C1 to start card }
end;

Procedure Power_Down;
begin
port[portA_addr]:=0;
port[cont_8255]:=7;
(* delay(round(tsamp));*)
port[cont_8255]:=$03;   { set port C1 to reset card }
end;

Procedure NewSpeed(rsp:real);
{if sp>0 then positive direction ,else negative }
var
sp:integer;
begin
sp:=round(rsp);
port[portA_addr]:=0;
if sp>255 then sp:=255
else if sp<-255 then sp:=-255;
if sp>=0 then port[cont_8255]:=5
else port[cont_8255]:=4;
port[portA_addr]:=abs(sp);

```

end;

```

PROCEDURE LS(VAR bbb,aaa,lambda:real;
             VAR PAR1,PHI,diag:vec1;
             VAR offdiag:vec2);
(*computes the least-squares estimate using the U-D method
by Bierman & Thornton*)
VAR
  kf,ku,i,j :integer;
  perr,fj,vj,alphaj,ajlast,pj,w :real;
  k :ARRAY[1..2] OF real;
  n,na :integer;
(*par1 vector : theta vector*)
(*diag vector : initially, rho times identity vector*)
(*offdiag vector : initially, null vector*)
BEGIN
n:=2; (*for n, refer to chapter 3 of the thesis*)
na:=1; (*for na, refer to chapter 3 of the thesis*)
perr:=aaa; (*perr:=y*)
for i:=1 to n do perr:=perr-PAR1[i]*PHI[i];
(*calculate gain and covariance using U-D method*)
fj:=PHI[i];
vj:=diag[i]*fj;
k[i]:=vj;
alphaj:=1.+vj*fj;
diag[i]:=(diag[i]/alphaj)/lambda;
if n>1 then
begin
kf:=0;
ku:=0;
for j:=2 to n do
begin
fj:=PHI[j];
for i:=1 to j-1 do
begin (*f=PHI*U*)
kf:=kf+1;
fj:=fj+PHI[i]*offdiag[kf];
end;
vj:=fj*diag[j]; (*v=D*f*)
k[j]:=vj;
ajlast:=alphaj;
alphaj:=ajlast+vj*fj;
diag[j]:=diag[j]*ajlast/alphaj/lambda;
pj:=-fj/ajlast;
for i:=1 to j-1 do
begin
ku:=ku+1;
w:=offdiag[ku]+k[i]*pj;
k[i]:=k[i]+offdiag[ku]*vj;
offdiag[ku]:=w
end;
end;
end;
end;
(*update parameter estimates*)
for i:=1 to n do PAR1[i]:=PAR1[i]+perr*k[i]/alphaj;
(*updating of phi*)

```

```

for i:=1 to n-1 do PHI[n+1-i]:=PHI[n-i];
PHI[1]:=aaa;
PHI[na+1]:=bbb;
END; (*procedure LS*)

```

```

BEGIN (*main program*)
initppi_cnt;
clrscr;
writeln('number of steps?');
readln(M);
writeln('Maximum overshoot?');
readln(overshoot);
writeln('Settling time?');
readln(settlingtime);
writeln('forgetting factor[0.95<=lambda<=1]?');
readln(lambda);
writeln('reference velocity?');
readln(vref);
writeln('initially estimated Je (Jeo^)?');
readln(initiallestje);
natlog:=sqr(ln(overshoot/100));
zeta:=sqrt(natlog/(sqr(pi)+natlog));
wn:=4/(zeta*settlingtime);
tsam:=0.025;
rho:=10000;
ra:=1.3;
be:=0.0001442;
kt:=0.07037;
kb:=kt;
ka:=4.656;
ke:=50000;
KAD:=1/50000;
N:=1570.8;
iii:=2.1274;
diag[1]:=rho;
diag[2]:=rho;
offdiag[1]:=0;
denom:=(ra*be+kt*kb);
km:=kt/denom;
tauest:=(ra*initiallestje)/denom; (*initially estimated time const.*)
kprime:=(ka*km*ke*KAD)/(iii*N);
kprime:=0.0003782;
ainitial:=exp(-tsam/tauest);
binitial:=kprime*(1-ainitial);
PAR1[1]:=ainitial;
PAR1[2]:=binitial; (* initial thetaest's *)
ek:=0;
ek_1:=0;

(*beginning of STR box*)
(*beginning of Pole Ass. box*)
c1:=-2*exp(-zeta*wn*tsam)*cos(sqrt(1-zeta*zeta)*wn*tsam);
c2:=exp(-2*zeta*wn*tsam);
ddenom:=((kprime)*(1-ainitial));
kp:=(ainitial-c2)/ddenom; (*initial Kp*)
ki:=(c1+c2+1)/(tsam*ddenom); (*initial Ki*)
(*end of Pole Ass. box*)

```



```

u[0]:=0;
(*end of STR box*)
v[0]:=0;
aaa:=v[0];    (*aaa=v[0]*)
bbb:=u[0];    (*bbb=u[0]*)
PHI[1]:=aaa;
PHI[2]:=bbb;  (*initial PHI vector*)

Assign(file1,'VELOCITY.DAT');
Assign(file2,'PAR1.DAT');
Assign(file3,'PAR2.DAT');
Rewrite(file1);
Rewrite(file2);
Rewrite(file3);
clrscr;
gotoxy(1,20);
writeln('As the loop proceeds, press any key to exit...');
delay(1200);
power_up;
index:=1;
repeat
  EndCheck;          { wait until the end of sampling period }
  ccc:=aaa;  (*i.e., ccc=v[index-1]*)
  (*process*)
  speed:=0;
  if (LastDir=0) and (secs>0) then
    speed:=20.832*Pulses/Secs
  else if secs>0 then
    speed:=-20.832*Pulses/Secs;    { in m/s }
  v[index]:=speed;
  if index>4 then v[index]:=0.25*(v[index]+v[index-1]+v[index-2]
    +v[index-3]);
  (*end process*)

  aaa:=v[index];
  WRITE('CURRENT VELOCITY(m/s)____');
  WRITELN(v[index]:9:7);
  writeln(file1,v[index]);
  if index>5 then LS (bbb,aaa,lambd,PAR1,PHI,diag,offdiag);
  writeln(file2,PAR1[1]);
  writeln(file3,v[index]);
  (*pole-assignment*)
  tauest:=-tsam/ln(PAR1[1]);
  jeest:=denom*tauest/ra;
  kp:=(c1-c2+1+2*PAR1[1])/(2*(kprime*(1-PAR1[1])));
  ki:=(c1+c2+1)/(tsam*kprime*(1-PAR1[1]));
  (*pole-assignment end*)

  (*controller*)
  ek:=vref-aaa;    (*aaa=v[index]*)
  ek_1:=vref-ccc;  (*ccc=v[index-1]*)
  deltau:=((kp+(ki*tsam/2))*ek + ((ki*tsam/2)-kp)*ek_1);
  u[index]:=u[index-1]+deltau;
  (*controller end*)
  if (u[index]>255) then u[index]:=255
  else if (u[index]<-255) then u[index]:=-255;
  newspeed(u[index]);
  bbb:=u[index];
  inc(index);

```

```
until keypressed or (index>M);  
power_down;  
Close(file1);  
Close(file2);  
Close(file3);  
END. (*main program*)
```





APPENDIX B

**Listings of the Classical PD and Adaptive
Pole Assignment Position Control Programs**

```

{$R+}
PROGRAM PAPDPOS;
(*Pole Assignment PD Position Control*)
USES Crt,graph;
CONST
M=300;
TYPE
matr=ARRAY[1..3,1..3] of real;
solution=ARRAY[1..3] of real;

VAR
km,tauest,je,c1,c2,ka,ke,iii,N,tsamp,rho,posref,initialpos :real;
zeta,wn,fff,coeff1,coeff2,coeff3,jeo,ra,lambda,kt,kb,be :real;
denom,aaa,bbb,kp,kd,r,a :real;
alpha,beta,kprime,pr1,pr2,pr3,pr4,initialu :real;
ccc,a1,a2,ek,ek_1 :real;
index :integer;
initiallestje,b :real;
Amat :matr;
Bmat,X :solution;
pos :ARRAY[-1..M] of real;
u :ARRAY[-1..M] of real;
file1,file2 :text;
Const
PortA_Addr = $340;      { 8255 Ports }
PortB_Addr = $341;
PortC_Addr = $342;
Cont_8255  = $343;

Cnt0_Addr  = $348;      { 8253 Ports }
Cnt1_Addr  = $349;
Cnt2_Addr  = $34A;
Cont_Reg   = $34B;

PosL_Addr  = $350;
PosH_Addr  = $358;

Var
NextReady : word;      {if the controller has calculated the next output }
CalcLate   : word;
LastDir    : word;     { the direction of rotation, 1=clockwise      }
Pulses     : word;     { # of encoder pulses in the measurement interval }
Secs       : word;     { # of MMLOCK pulses in the measurement interval }
Position   : word;

{$I graf.inc}
{$L endcheck.obj}

Procedure EndCheck ; external;

Procedure InitPPI_Cnt;
{ Configures 8255 and 8253 }
begin
port[Cont_8255]:=$8A; {A inmode0,output.B input.C Hiinput,C Lo output}
port[cont_8255]:=3;   { reset card }

port[Cont_Reg]:=$36;   { Counter 0 in square wave mode   }

```

```

port[Cnt0_Addr]:=$5C;      { set sampling period }
port[Cnt0_Addr]:=$06;      { first LSB then MSB }
tsamp:=25;
{ clk period=15.36 micro secs }
{ measurement clk frequency 1.0416 MHz }

```

```

port[Cont_Reg]:=$70;      { Counter 1 in Mode 0 }
port[Cnt1_Addr]:=$FF;
port[Cnt1_Addr]:=$FF;

```

```

port[Cont_Reg]:=$B0;      { Counter 2 in Mode 0 }
port[Cnt2_Addr]:=$FF;
port[Cnt2_Addr]:=$FF;

```

```
end;
```

```
procedure power_up;
```

```

begin
  port[cont_8255]:=$03;      { set port C1 to reset card }
  port[cont_8255]:=$00;      { clear position cnt , toggle C0 }
  port[cont_8255]:=$01;

  { clear counters }
  port[cont_reg]:=$B0;      { load cnt2 }
  port[cnt2_addr]:=$FF;      { LSB }
  port[cnt2_addr]:=$FF;      { MSB }
  port[cont_reg]:=$70;      { load cnt1 }
  port[cnt1_addr]:=$FF;      { LSB }
  port[cnt1_addr]:=$FF;      { MSB }

  port[cont_8255]:=7;        { set port C3 for position latch signal }
  port[cont_8255]:=$02;      { reset port C1 to start card }
end;
```

```
Procedure Power_Down;
```

```

begin
  port[portA_addr]:=0;
  port[cont_8255]:=7;
  (* delay(round(tsamp));*)
  port[cont_8255]:=$03;      { set port C1 to reset card }
end;
```

```
Procedure NewSpeed(rsp:real);
```

```

{if sp>0 then positive direction ,else negative }
var
sp:integer;
begin
  sp:=round(rsp);
  port[portA_addr]:=0;
  if sp>255 then sp:=255
  else if sp<-255 then sp:=-255;
  if sp>=0 then port[cont_8255]:=5
  else port[cont_8255]:=4;
  port[portA_addr]:=abs(sp);
end;
```

```
Procedure Setting(VAR wn,zeta,ra,kt,kb,be,iii,N,ka,ke,initialpos:real;
```

```

                VAR rho,lambda,fff,initiallestje,posref:real;
                VAR jeo :real;
                VAR tsamp:real);
VAR
f,overshoot,settlingtime,natlog :real;
choice :char;

BEGIN
clrscr;
writeln('To input the desired dynamics in terms of maximum overshoot');
writeln('and settling time press ( S ), or in terms of natural freq. ');
writeln('and damping ratio press( D )');
REPEAT
readln(choice);
choice:=UpCase(choice);
IF choice='S' THEN BEGIN
                write('Maximum overshoot(%)____');
                readln(overshoot);
                write('Settling time(s)____');
                readln(settlingtime);
                natlog:=sqr(ln(overshoot/100));
                zeta:=sqr(natlog/(sqr(pi)+natlog));
                wn:=4/(zeta*settlingtime)
                END
ELSE IF choice='D' THEN BEGIN
                write('Natural freq.(Hz)[1 Hz]____');
                readln(f);
                write('Damping ratio[0.95]____');
                readln(zeta);
                wn:=2*pi*f
                END
ELSE BEGIN
                sound(400);
                delay(200);
                nosound
                END;
UNTIL ((choice='S') OR (choice='D'));
write('Sampling time(sec)[0.01 sec]____');
readln(tsamp);
write('Reference position(m)____');
readln(posref);
write('Initially estimated Je(kgm2)____');
readln(initiallestje);
END; (*procedure setting*)

```

```

Procedure GaussElimination(VAR Amat :matr;
                            VAR Bmat :solution;
                            VAR X :solution;
                            VAR zeta,wn :real);

```

```

VAR
i,ia1,j,k,nej,v :integer;
piv,tot :real;
BEGIN
(*elimination*)
for i:=1 to 2 do begin
ia1:=i+1;
for j:=ia1 to 3 do begin
piv:=Amat[j,i]/Amat[i,i];

```

```

Amat[j,i]:=0.;
for k:=ia1 to 3 do Amat[j,k]:=Amat[j,k]-Amat[i,k]*piv;
Bmat[j]:=Bmat[j]-Bmat[i]*piv;
end;
end;
(*backward scanning*)
X[3]:=Bmat[3]/Amat[3,3];
write('3rd root_____');writeln(X[3]);
for j:=1 to 2 do begin
nej:=3-j;
v:=nej+1;
tot:=0.;
for i:=v to 3 do tot:=tot+Amat[nej,i]*X[i];
X[nej]:=(Bmat[nej]-tot)/Amat[nej,nej];
end;
END; (*procedure gausselimination*)

```

```

BEGIN (*main program*)
initppi_cnt;
clrscr;
Setting(wn,zeta,ra,kt,kb,be,iii,N,ka,ke,posref,initialpos,
rho,lambda,fff,initiallestje,jeo,tsamp);
ra:=1.3;
kt:=0.07037;
kb:=kt;
be:=0.0001442;
Assign(file1,'POSITION.DAT');Rewrite(file1);
Assign(file2,'CSIG.DAT');Rewrite(file2);
denom:=(ra*be+kt*kb);
km:=kt/denom;
tauest:=(ra*INITIALESTJE)/denom; (*initially estimated time constant*)
b:=1/tauest;
kprime:=0.0003782*tauest;
a:=exp(-b*tsamp);
alpha:=(b*tsamp-1+a);
beta:=(1-a-b*a*tsamp);
index:=1;

(*pole assignment*)
c1:=-2*exp(-zeta*wn*tsamp)*cos(sqrt(1-zeta*zeta)*wn*tsamp);
c2:=exp(-2*zeta*wn*tsamp);
Amat[1,1]:=kprime*alpha;
Amat[1,2]:=Amat[1,1]/tsamp;
Amat[1,3]:=1;
Amat[2,1]:=kprime*beta;
Amat[2,2]:=kprime*(beta-alpha)/tsamp;
Amat[2,3]:=c1;
Amat[3,1]:=0;
Amat[3,2]:=-kprime*beta/tsamp;
Amat[3,3]:=c2;
Bmat[1]:=c1+a+1;
Bmat[2]:=c2-a;
Bmat[3]:=0;
GAUSSELMINATION(Amat,Bmat,X,zeta,wn);
kp:=X[1];
kd:=X[2];
r:=X[3];
(*end of Pole Ass. box*)

```

```

aaa:=0; (*aaa=pos[0]*)
bbb:=0; (*bbb=u[0]*)

pos[-1]:=0;
pos[0]:=0;
u[-1]:=0;
u[0]:=0;
writeln('As the loop proceeds, press any key to exit...');
power_up;
repeat;
Endcheck;
ccc:=aaa; (*i.e., ccc=pos[index-1]*)

(*process*)
pos[index]:=position/50000.0;
aaa:=pos[index];
WRITE('CURRENT POSITION(m)___');
WRITELN(pos[index]);
writeln(file1,pos[index]);
(*process end*)

(*PD regulator*)
a2:=-kd/tsamp;
a1:=kp-a2;
ek:=posref-pos[index]; (*aaa=pos[index]*)
ek_1:=posref-pos[index-1]; (*ccc=pos[index-1]*)
u[index]:= a1*ek + a2*ek_1;
if (u[index]>255) then u[index]:=255;
else if (u[index]<-255) then u[index]:=-255;
newspeed(u[index]);
bbb:=u[index];
writeln(file2,u[index]);
(*PD end*)

inc(index);
until keypressed or (index>M);
power_down;
Close(file1);
Close(file2);
END. (*main program*)

```



```

{$N+}
{$R+}
PROGRAM ADAPPOS;
(* Self-Tuning Adaptive Position Control *)
USES
  Crt,graph;
TYPE
  vec1=ARRAY[1..4] OF extended;
  vec2=ARRAY[1..6] OF extended;
VAR
  km,tauest,denom,kprime,tau,je,areal :extended;
  c1,c2,ddenom,sa,sb,sc,s0,s1,r1,t0,d :extended;
  ka,ke,iii,N,tsamp,rho,xref,zeta,wn,fff :extended;
  coeff1,coeff2,coeff3,jeo,ra,lambda,temporary: extended;
  kt,kb,be,initialstje,a1init,a2init,b1init,b2init :extended;
  ii,jj,index,i,M :integer;
  x,u :ARRAY[-1..2000] OF extended;
  aaa,a,f, a1,a2,b1,b2,bbb,ccc,ddd,eee :extended;
  err_a1,err_a2,err_b1,err_b2 :extended;
  PAR1,PHI,diag :vec1;
  offdiag :vec2;
  jeest :extended;
  c,Kad :extended;
  file1,file2,file3 :text;
  noisedist :boolean;
  momint,ansquestion :char;

Const
  PortA_Addr = $340;    { 8255 Ports }
  PortB_Addr = $341;
  PortC_Addr = $342;
  Cont_8255  = $343;

  Cnt0_Addr  = $348;    { 8253 Ports }
  Cnt1_Addr  = $349;
  Cnt2_Addr  = $34A;
  Cont_Reg   = $34B;

  PosL_Addr  = $350;
  PosH_Addr  = $358;

Var
  NextReady : word;    {if the controller has calculated the next output}
  CalcLate  : word;
  LastDir   : word;    {the direction of rotation, 1=clockwise }
  Pulses    : word;    {# of encoder pulses in the measurement interval }
  Secs      : word;    {# of MMCKLOCK pulses in the measurement interval }
  Position  : word;

{$I graf.inc}
{$L endcheck.obj}

Procedure EndCheck ; external;

Procedure InitPPI_Cnt;
{ Configures 8255 and 8253 }

```

```

begin
port[Cont_8255]:=$8A; {A inmode0,output.Binput.C Hiinput, C Lo output}
port[cont_8255]:=3; { reset card }

port[Cont_Reg]:=$36; { Counter 0 in square wave mode }
port[Cnt0_Addr]:=$5C; { set sampling period }
port[Cnt0_Addr]:=$06; {first LSB then MSB }
tsamp:=25;
{ clk period=15.36 micro secs }
{ measurement clk frequency 1.0416 MHz }

port[Cont_Reg]:=$70; { Counter 1 in Mode 0 }
port[Cnt1_Addr]:=$FF;
port[Cnt1_Addr]:=$FF;

port[Cont_Reg]:=$B0; { Counter 2 in Mode 0 }
port[Cnt2_Addr]:=$FF;
port[Cnt2_Addr]:=$FF;

end;

procedure power_up;
begin
port[cont_8255]:=$03; { set port C1 to reset card }
port[cont_8255]:=$00; { clear position cnt , toggle C0 }
port[cont_8255]:=$01;

{ clear counters }
port[cont_reg]:=$B0; { load cnt2 }
port[cnt2_addr]:=$FF; { LSB }
port[cnt2_addr]:=$FF; { MSB }
port[cont_reg]:=$70; { load cnt1 }
port[cnt1_addr]:=$FF; { LSB }
port[cnt1_addr]:=$FF; { MSB }

port[cont_8255]:=7; { set port C3 for position latch signal }
port[cont_8255]:=$02; { reset port C1 to start card }
end;

Procedure Power_Down;
begin
port[portA_addr]:=0;
port[cont_8255]:=7;
(* delay(round(tsamp));*)
port[cont_8255]:=$03; { set port C1 to reset card }
end;

Procedure NewSpeed(rsp:real);
{if sp>0 then positive direction ,else negative }
var
sp:integer;
begin
sp:=round(rsp);
port[portA_addr]:=0;
if sp>255 then sp:=255
else if sp<-255 then sp:=-255;
if sp>=0 then port[cont_8255]:=5
else port[cont_8255]:=4;

```

```
port[portA_addr]:=abs(sp);
end;
```

```
PROCEDURE SETTING(VAR wn,zeta,ra,kt,kb,be,iii,N,ka,ke,xref:extended
                  VAR rho,lambda,fff,initialstje :extended;
                  VAR jeo :extended;
                  VAR tsamp,KAD:extended;
                  VAR noisedist:boolean;
                  VAR momint:char;
                  VAR ansquestion:char;
                  VAR M :integer);

LABEL 10;
VAR
  f,overshoot,settlingtime,natlog :real;
  choice,ans,answer :char;
BEGIN
  clrscr;
  gotoxy(23,10);
  writeln('P A R A M E T E R   S E T T I N G');
  gotoxy(23,11);
  writeln('-----');
  gotoxy(20,13);
  writeln('Default values are in square brackets...');
  delay(1800);
  10: clrscr;
  write('Number of steps____');
  readln(M);
  writeln('■To input the desired dynamics in terms of max. overshoot'
  writeln('and settling time press ( S ), or in terms of nat. freq.')
  write ('and damping ratio press ( D )____');
  REPEAT
  readln(choice);
  choice:=UpCase(choice);
  IF choice='S' THEN BEGIN
    write('Maximum overshoot[1%]____');
    readln(overshoot);
    write('Settling time[.75 sec]____');
    readln(settlingtime);
    natlog:=sqr(ln(overshoot/100));
    zeta:=sqrt(natlog/(sqr(pi)+natlog));
    wn:=4/(zeta*settlingtime)
  END
  ELSE IF choice='D' THEN BEGIN
    write('Natural freq.[1 Hz]____');
    readln(f);
    write('Damping ratio[0.95]____');
    readln(zeta);
    wn:=2*pi*f
  END
  ELSE BEGIN
    sound(300);
    nosound
  END;
UNTIL ((choice='S') OR (choice='D'));
write('Sampling time[Ts=0.025 sec]____');
readln(tsamp);
write('Reference position(m)____');
```

```

readln(xref);
write('rho[1<rho<10000000]____');
readln(rho);
write('Forgetting factor[0.95<=lambda<=1]____');
readln(lambda);
write('Initially estimated equivalent mom. of inertia____');
readln(initiallestje);
clrscr;
gotoxy(20,20);
write('■Do you want to change any data? (Y/N)...');
readln(ans);
ans:=Uppcase(ans);
IF ans='Y' THEN goto 10;
clrscr;
END;  (*procedure setting*)

```

```

PROCEDURE LS(VAR bbb,aaa,lambda:extended;
             VAR PAR1,PHI,diag:vec1;
             VAR offdiag:vec2);
(*computes the least-squares estimate using the U-D method
by Bierman & Thornton*)
VAR
  kf,ku,i,j :integer;
  perr,fj,vj,alphaj,ajlast,pj,w :real;
  k :ARRAY[1..4] OF real;
  n,na :integer;
(*par1 vector : theta vector*)
(*diag vector : initially, rho times identity vector*)
(*offdiag vector : initially, null vector*)
BEGIN
n:=4;  (*for n, refer to chapter 3 of the thesis*)
na:=2; (*for na, refer to chapter 3 of the thesis*)
perr:=aaa;  (*perr:=y*)
for i:=1 to n do perr:=perr-PAR1[i]*PHI[i];
(*calculate gain and covariance using U-D method*)
fj:=PHI[1];
vj:=diag[1]*fj;
k[1]:=vj;
alphaj:=1+vj*fj;
diag[1]:=(diag[1]/alphaj)/lambda;
if n>1 then
begin
kf:=0;
ku:=0;
for j:=2 to n do
begin
fj:=PHI[j];
for i:=1 to j-1 do
begin  (*f=PHI*U*)
kf:=kf+1;
fj:=fj+PHI[i]*offdiag[kf];
end;
vj:=fj*diag[j];  (*v=D*f*)
k[j]:=vj;
ajlast:=alphaj;
alphaj:=ajlast+vj*fj;
diag[j]:=diag[j]*ajlast/alphaj/lambda;
pj:=-fj/ajlast;

```

```

for i:=1 to j-1 do
begin
ku:=ku+1;
w:=offdiag[ku]+k[i]*pj;
k[i]:=k[i]+offdiag[ku]*vj;
offdiag[ku]:=w
end;
end;
end;
(*update parameter estimates*)
for i:=1 to n do PAR1[i]:=PAR1[i]+perr*k[i]/alphaj;
(*updating of phi*)
for i:=1 to n-1 do PHI[n+1-i]:=PHI[n-i];
PHI[1]:=aaa;
PHI[na+1]:=bbb;
END; (*procedure LS*)

BEGIN (*main program*)
initppi_cnt;
clrscr;
Setting(wn,zeta,ra,kt,kb,be,iii,N,ka,ke,xref,
rho,lambda,fff,initialestje,
jeo,tsamp,KAD,noisedist,momint,ansquestion,M);
diag[1]:=rho; diag[2]:=rho; diag[3]:=rho; diag[4]:=rho;
offdiag[1]:=0; offdiag[2]:=0; offdiag[3]:=0;
offdiag[4]:=0; offdiag[5]:=0; offdiag[6]:=0;
ra:=1.3;
kt:=0.07037;
kb:=kt;
be:=0.0001442;
denom:=(ra*be+kt*kb);
km:=kt/denom;
c1:=-2*exp(-zeta*wn*tsamp)*cos(sqrt(1-zeta*zeta)*wn*tsamp);
c2:=exp(-2*zeta*wn*tsamp);
tauest:=(ra*initialestje)/denom; (*initially estimated time const*)
kprime:=0.0003782*tauest; (*kprime=(ka*km*ke*KAD)/(iii*N) *)
a1init:=1+exp(-tsamp/tauest);
a2init:=1-a1init;
b1init:=kprime*(tsamp/tauest-(1+a2init));
b2init:=kprime*(1+a2init+a2init*(tsamp/tauest));
PAR1[1]:=a1init;
PAR1[2]:=a2init;
PAR1[3]:=b1init;
PAR1[4]:=b2init; (*initial theta's *)
index:=1;
aaa:=0;
bbb:=0;
ddd:=0;
u[0]:=0.0; u[-1]:=0.0;
x[0]:=0.0; x[-1]:=0.0;
PHI[1]:=aaa; PHI[2]:=0;
PHI[3]:=bbb; PHI[4]:=0; (*initial PHI vector*)
ccc:=0;
eee:=0;
Assign(file1,'POS.DAT');Assign(file2,'A1EST.DAT');
Assign(file3,'B1EST.DAT');
Rewrite(file1);Rewrite(file2);Rewrite(file3);
clrscr;

```

```

gotoxy(1,20);
writeln('As the loop proceeds, press any key to exit...');
delay(1200);
power_up;
repeat
  Endcheck;
ccc:=aaa;    (*i.e., ccc=x[index-1]*)
(*process*)
x[index]:= (position)/50000.0;
(*end process*)

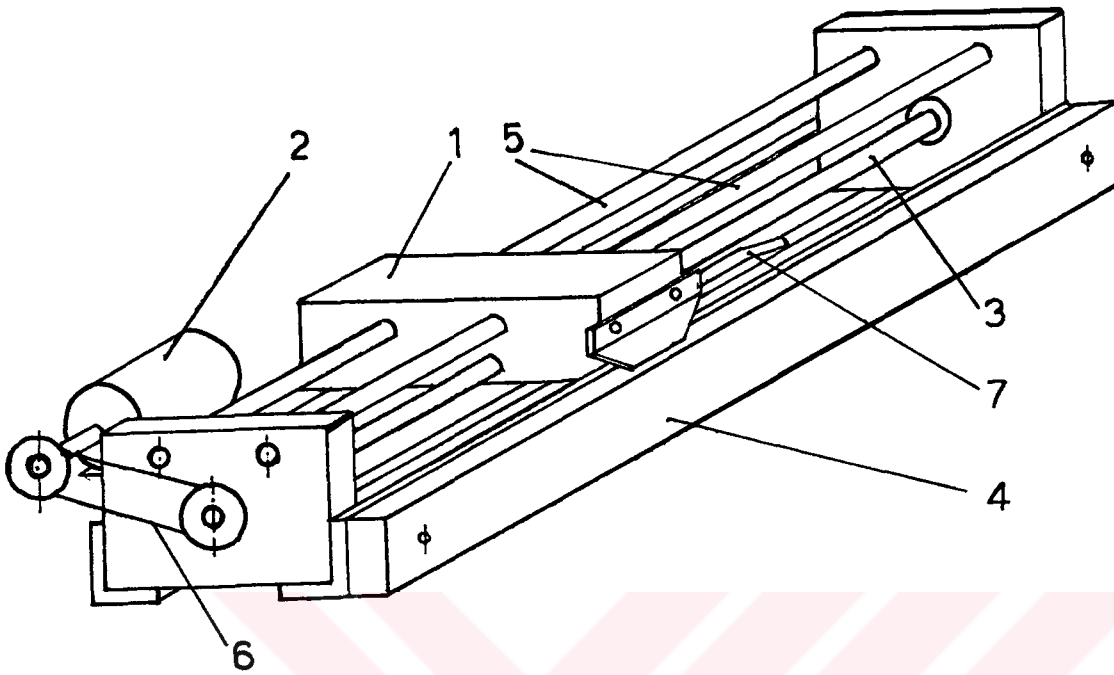
  aaa:=x[index];
  ddd:=ccc;
WRITE('CURRENT POSITION(m)____');
WRITELN(x[index]:9:7);
      writeln(file1,x[index]);
if index>5 then
LS (bbb,aaa,lambda,PAR1,PHI,diag,offdiag);

(*pole-assignment*)
d:=-PAR1[4]/PAR1[3];
t0:=(1+c1+c2)/(PAR1[3]*(1-d));
a:=-PAR1[2];
r1:= -d + (d*(d*d+c1*d+c2))/((d-1)*(d-a));
sa:=(1+c1+c2)*(a-d);
sb:=(a*a*a+c1*a*a+c2*a)*(1-d);
sc:=PAR1[3]*(1-d)*(a-d)*(1-a);
s0:=(sa-sb)/sc;
s1:=t0 - s0;
eee:=bbb;

(* control signal*)
u[index]:= t0*xref - s0*aaa - s1*ccc - r1*bbb;
  if (u[index]>255) then u[index]:=255
  else if (u[index]<-255) then u[index]:=-255;
newspeed(u[index]);
bbb:=u[index];
writeln(file2,PAR1[1]);
writeln(file3,PAR1[3]);
(*controller end*)
inc(index);
until keypressed or (index>M);
power_down;
Close(file1);Close(file2);Close(file3);
END. (*main program*)

```





1. Table
2. DC Motor
3. Ball Screw
4. Linear Encoder
5. Cylindrical Guides
6. Timing Belt
7. Encoder Cable

Figure 1.1. Schematic diagram of the experimental set-up.

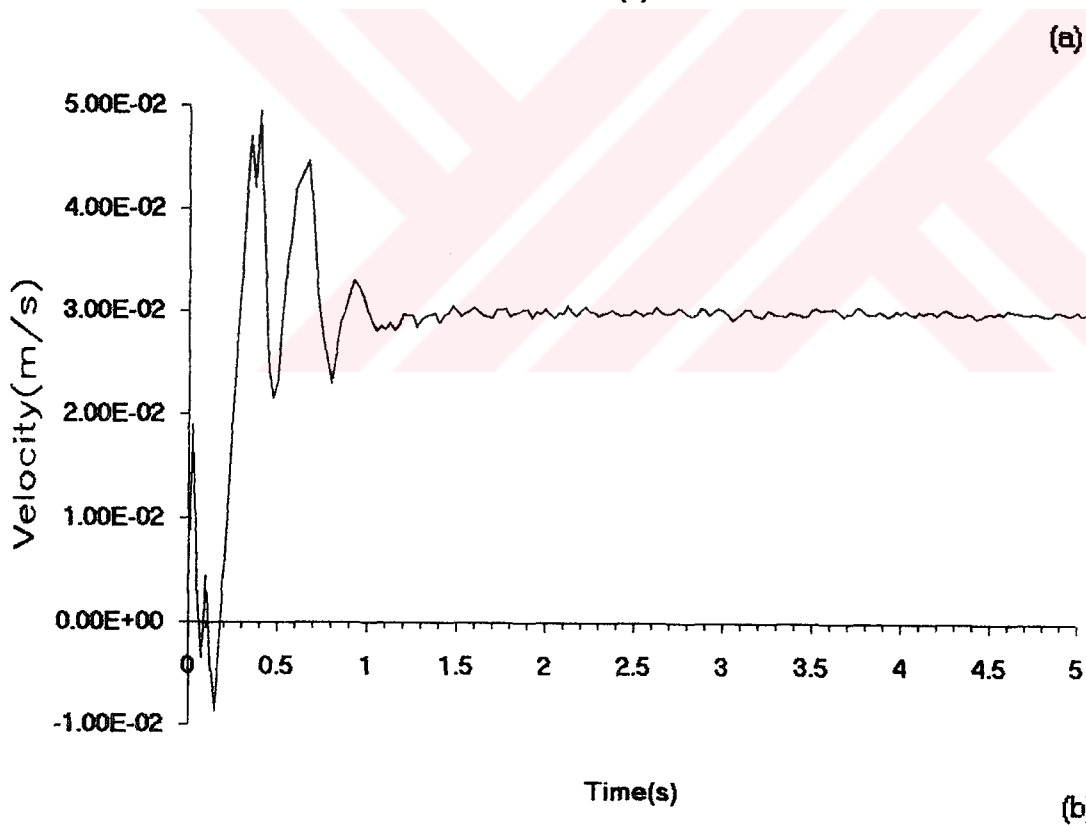
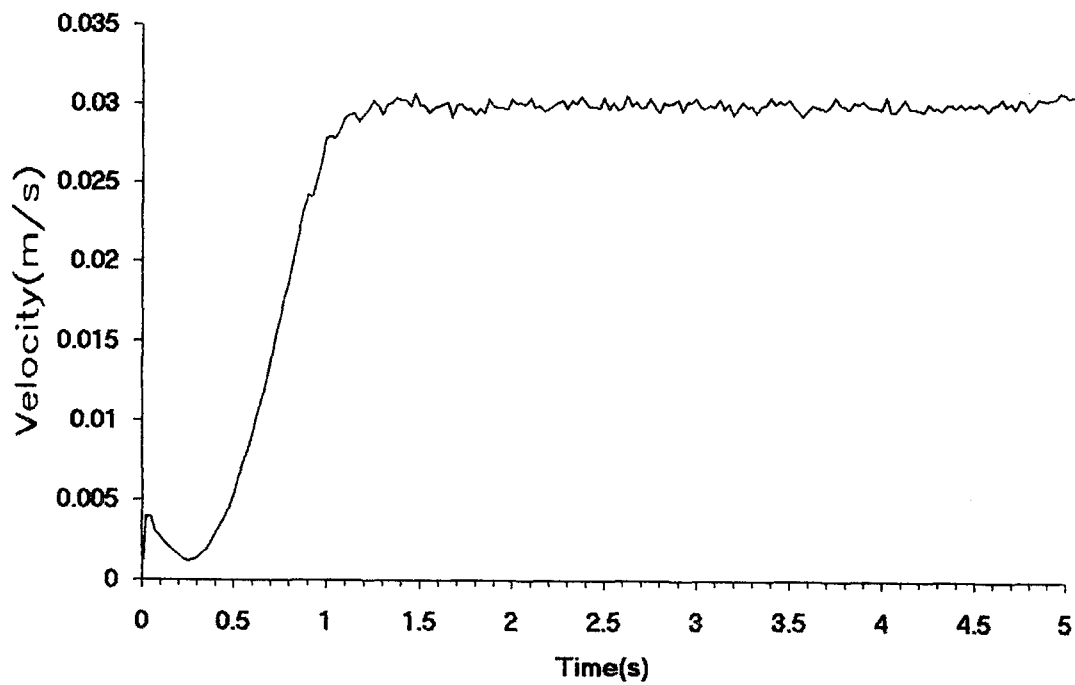


Figure 6.1. Effect of initial estimate on speed response in self-tuning adaptive control with $\lambda=0.96$ and $\rho=10000$.
 (a) $J_e=0.00001 \text{ kgm}^2$ and (b) $J_e=0.01 \text{ kgm}^2$.

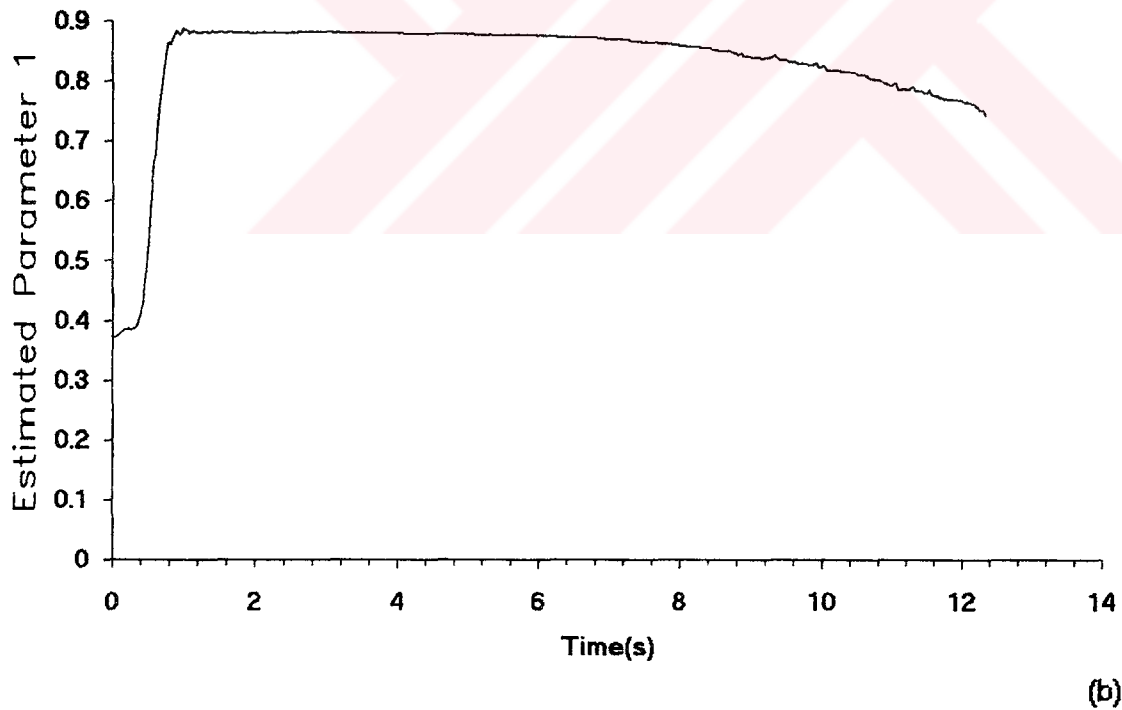
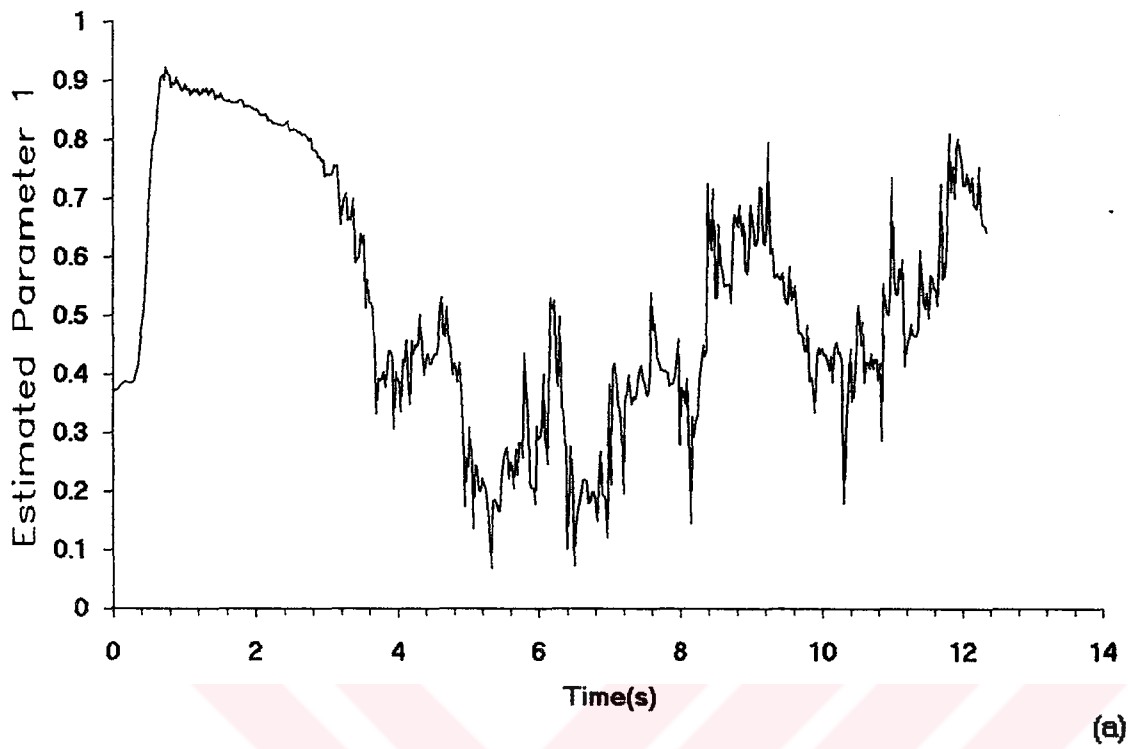


Figure 6.2. Effect of forgetting factor on estimated parameters in self-tuning adaptive control with $\rho=10000$ and $J_e=0.0001$ kgm².
 (a) $\lambda=0.93$, (b) $\lambda=0.99$.

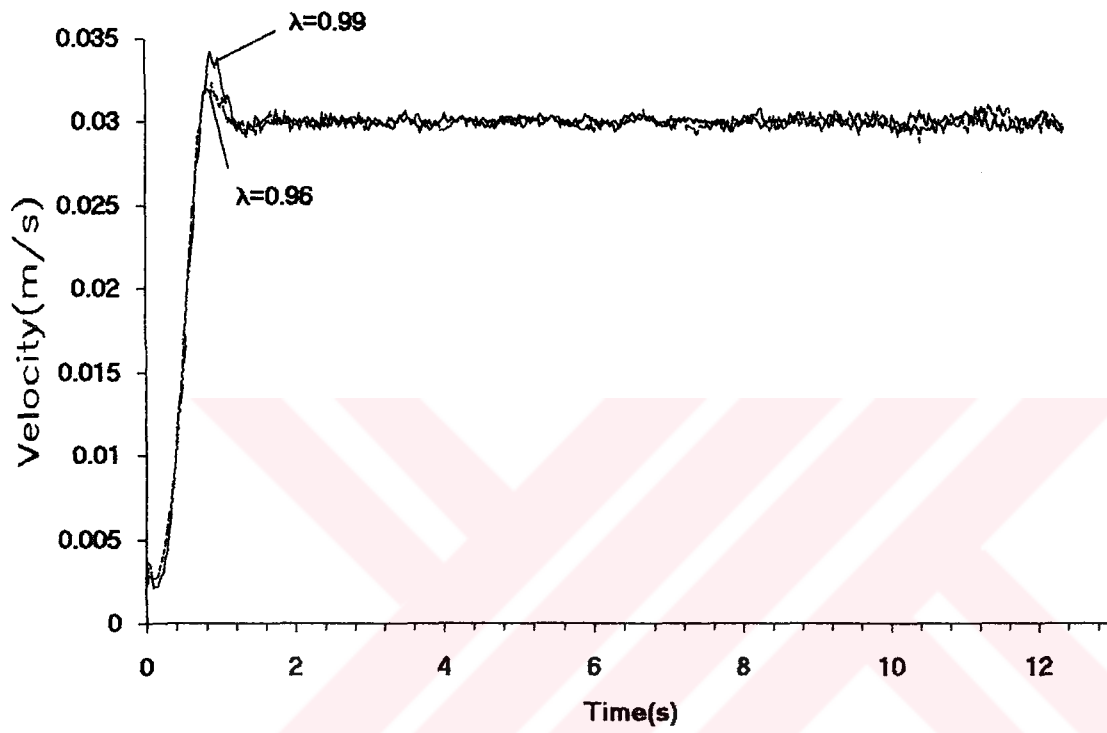
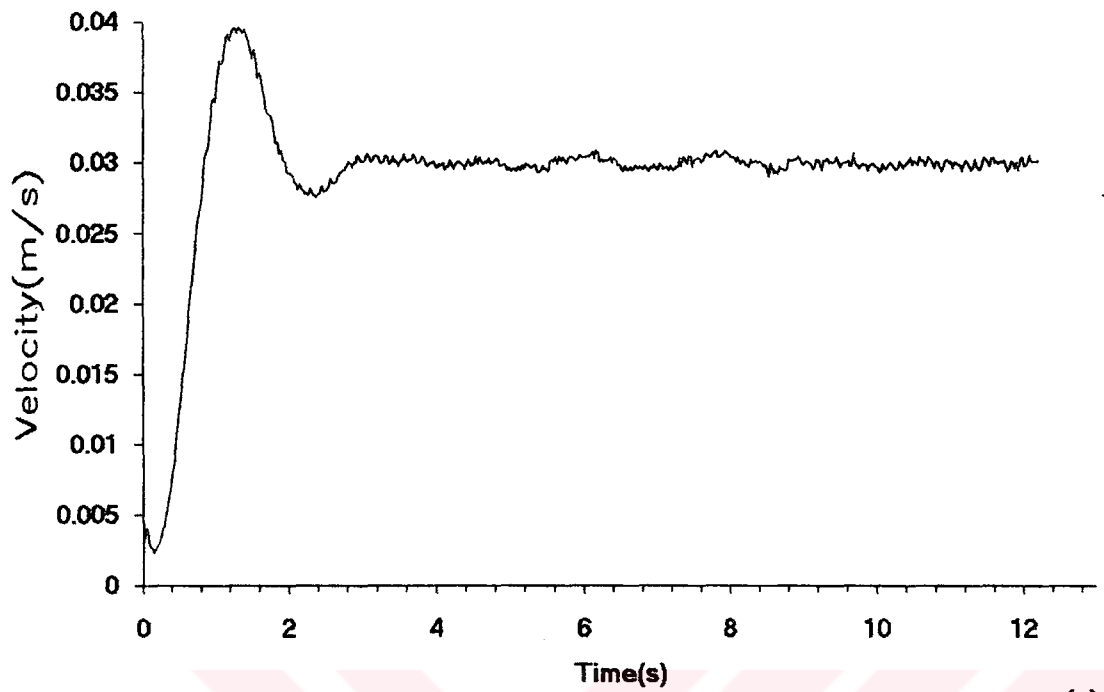
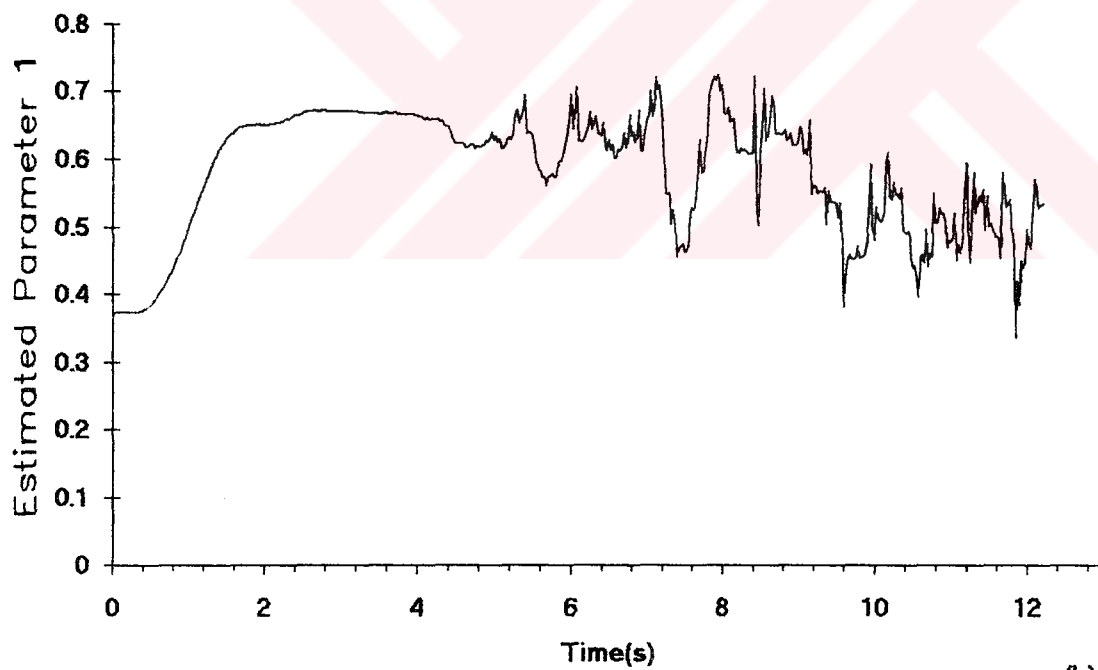


Figure 6.3. Effect of forgetting factor on speed response in self-tuning adaptive control with 10% overshoot and 1 s settling time.

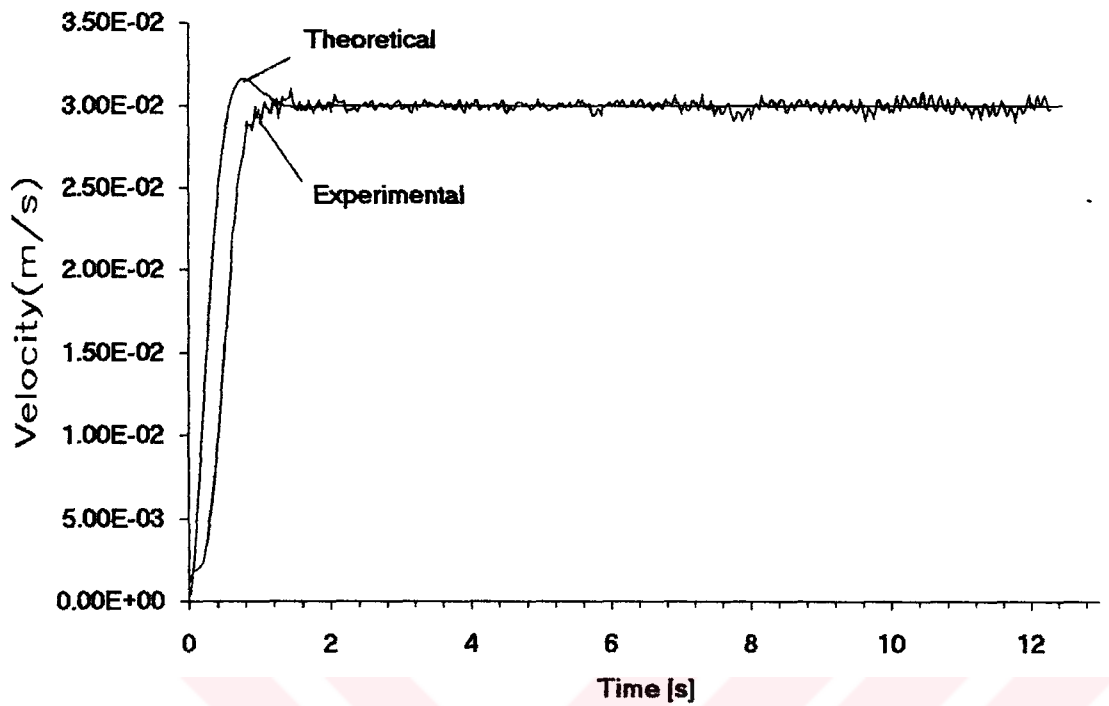


(a)

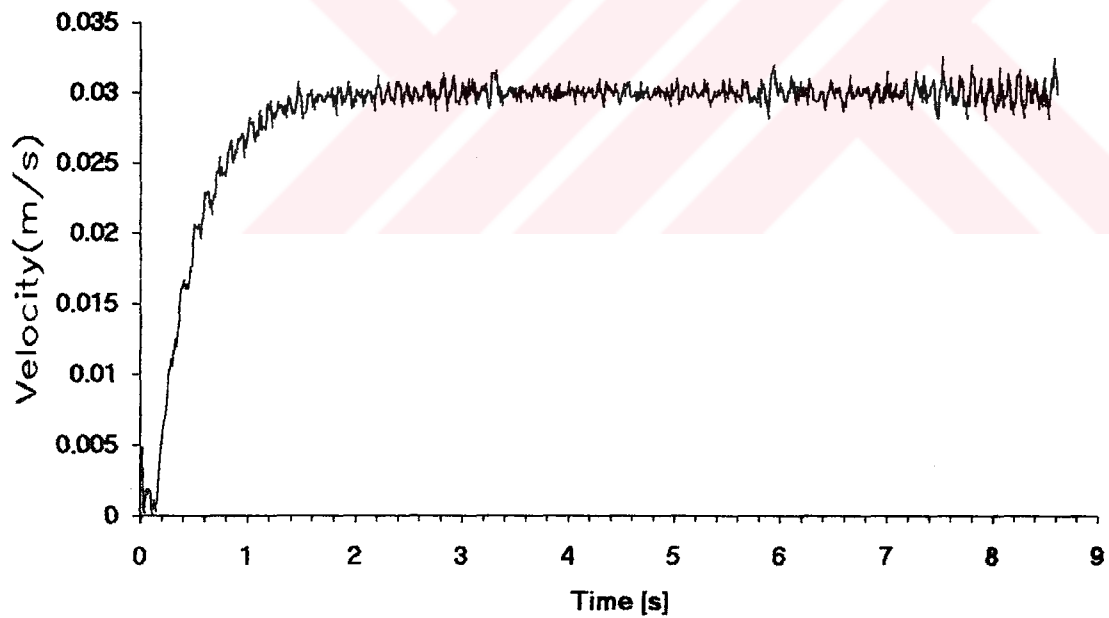


(b)

Figure 6.4. Effect of initial covariance matrix on (a) speed response, (b) parameter estimates with $\lambda=0.96$, $J_e=0.0001 \text{ kgm}^2$ and $\rho=100$ where $P(0)=\rho I$.

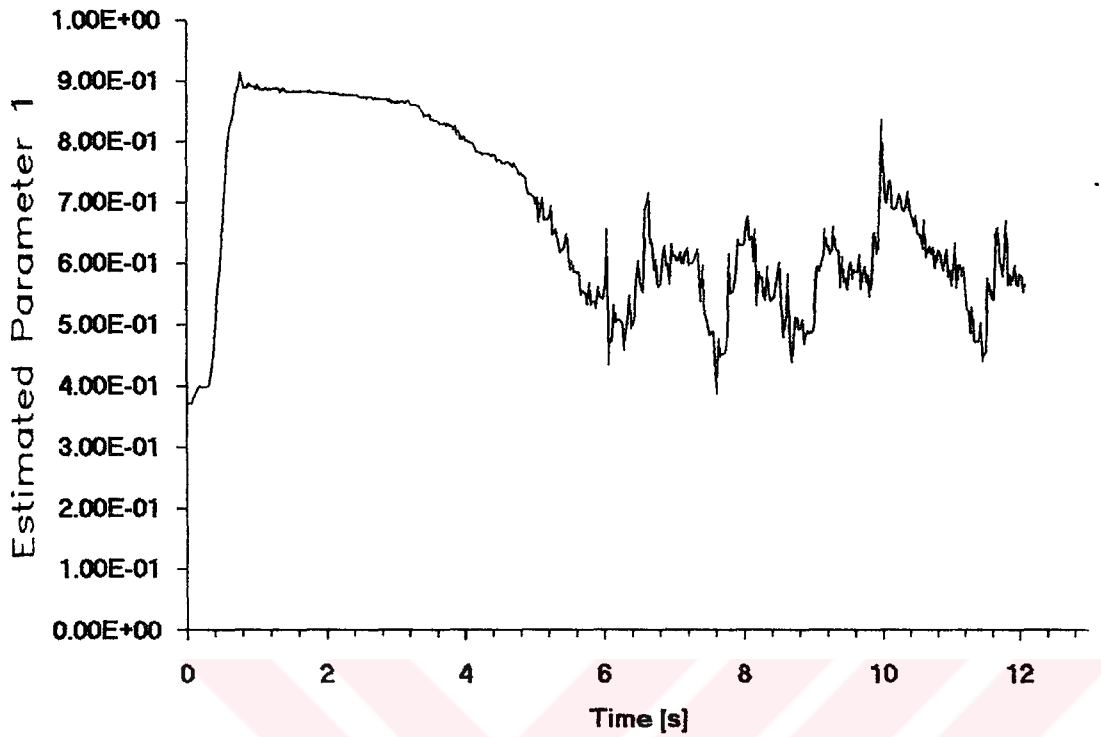


(a)

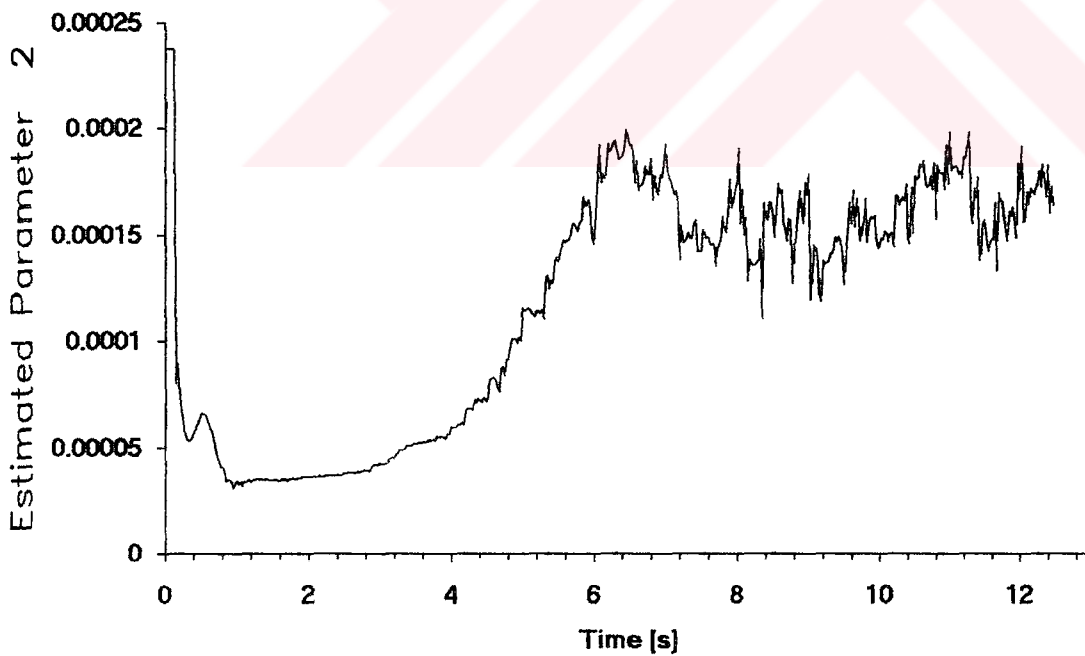


(b)

Figure 6.5. (a) Self-tuning adaptive speed response with optimal $\lambda=0.96$, $\rho=10000$, and estimated J_e of 0.0001 kgm^2 .
 (b) PI speed response with controller parameters tuned using Ziegler-Nichols method.

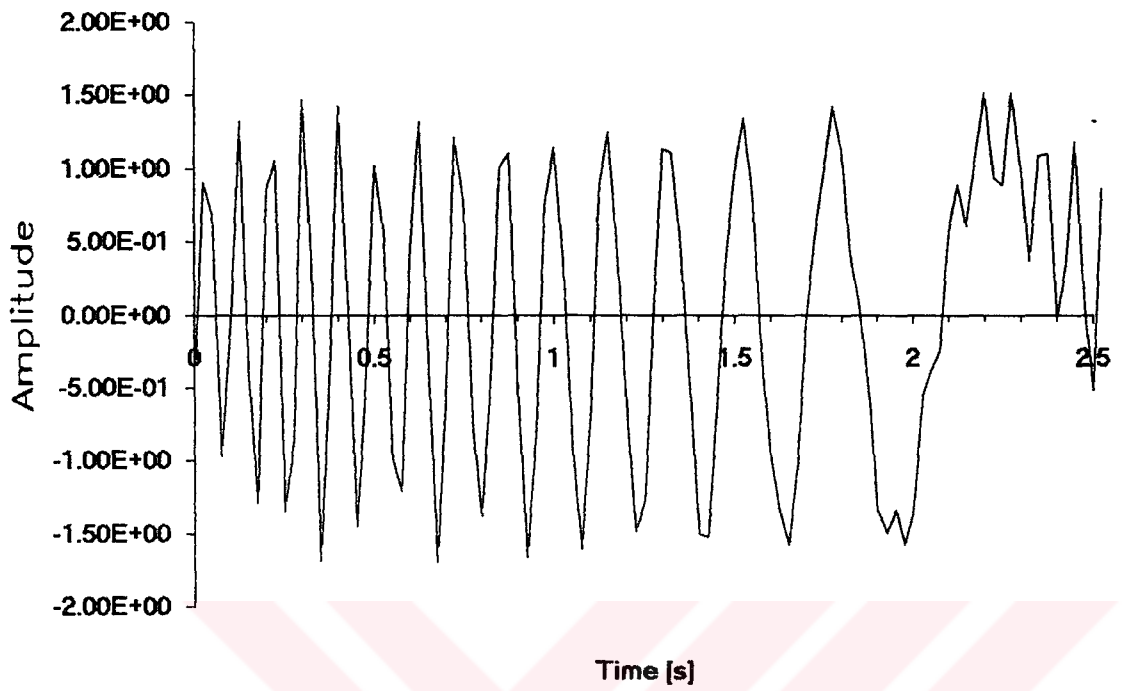


(a)

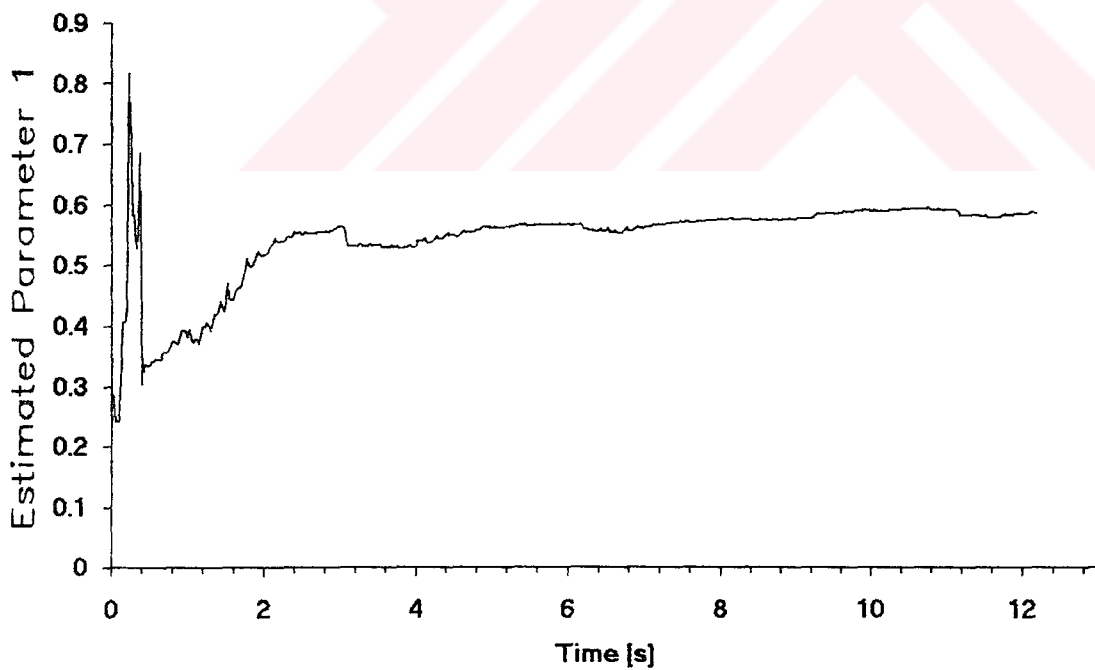


(b)

Figure 6.6. (a)-(b) Estimated parameters of the self-tuning adaptive control algorithm with $\lambda=0.96$, $\rho=10000$, and estimated J_e of 0.0001 kgm^2 .



(a)



(b)

Figure 6.7. (a) One period of SPHS signal with $T_p=2.5$ s and $N_H=30$.
(b) Parameter estimate θ_1 of the SPHS injected open-loop plant.

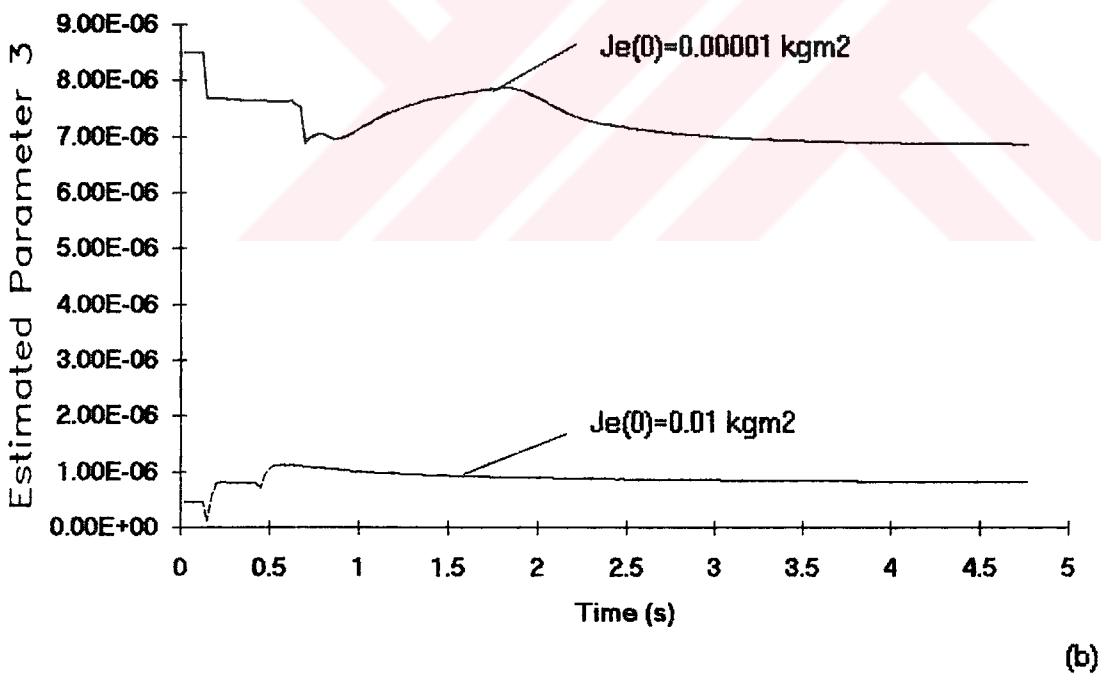
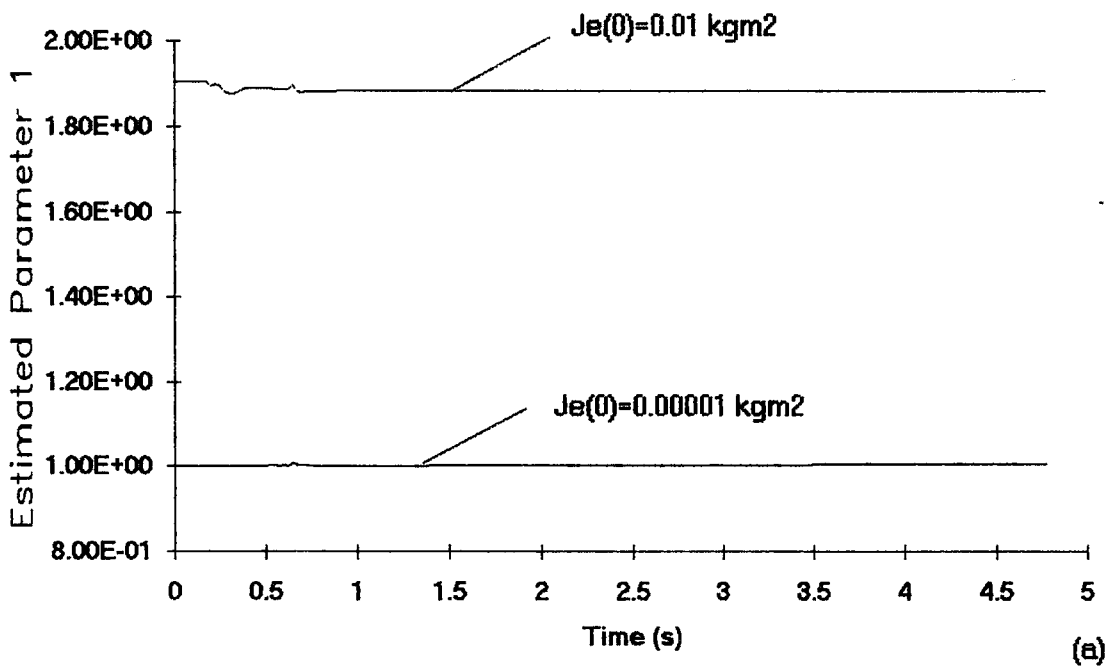


Figure 6.8. Effect of initial estimate on parameter estimates in adaptive position control with $\lambda=0.96$ and $\rho=1000$;
 (a) Parameter Estimate 1,
 (b) Parameter Estimate 2.

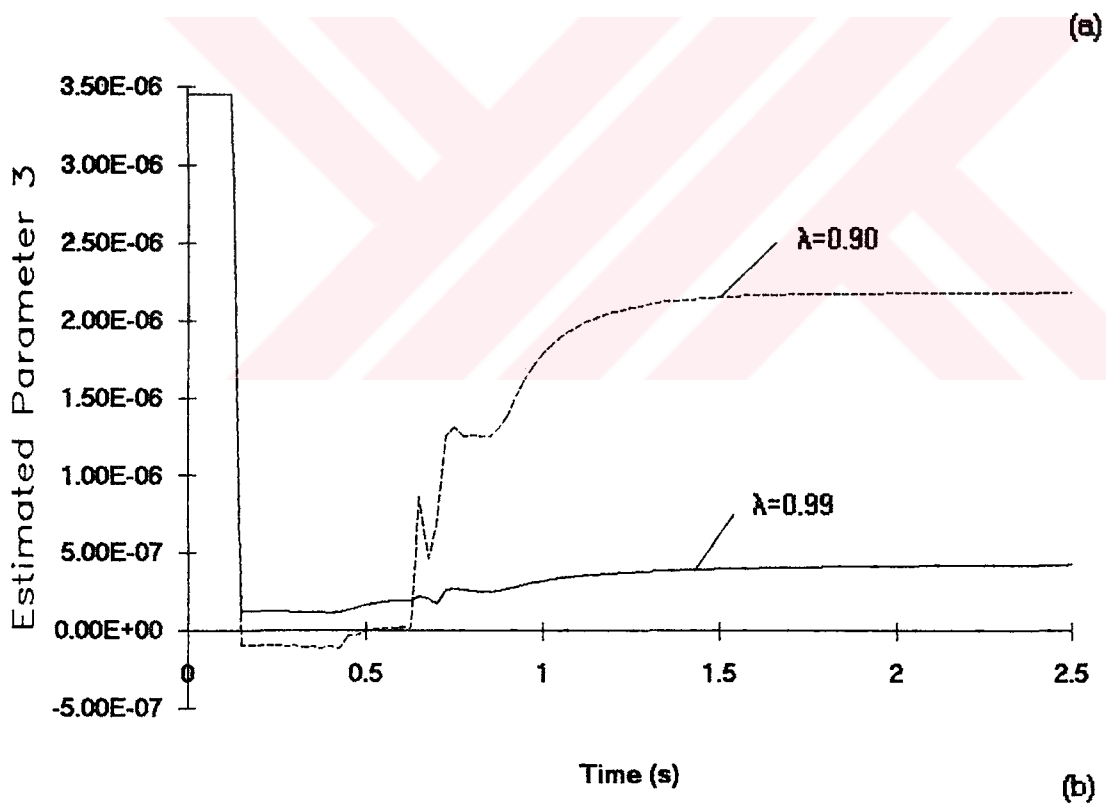
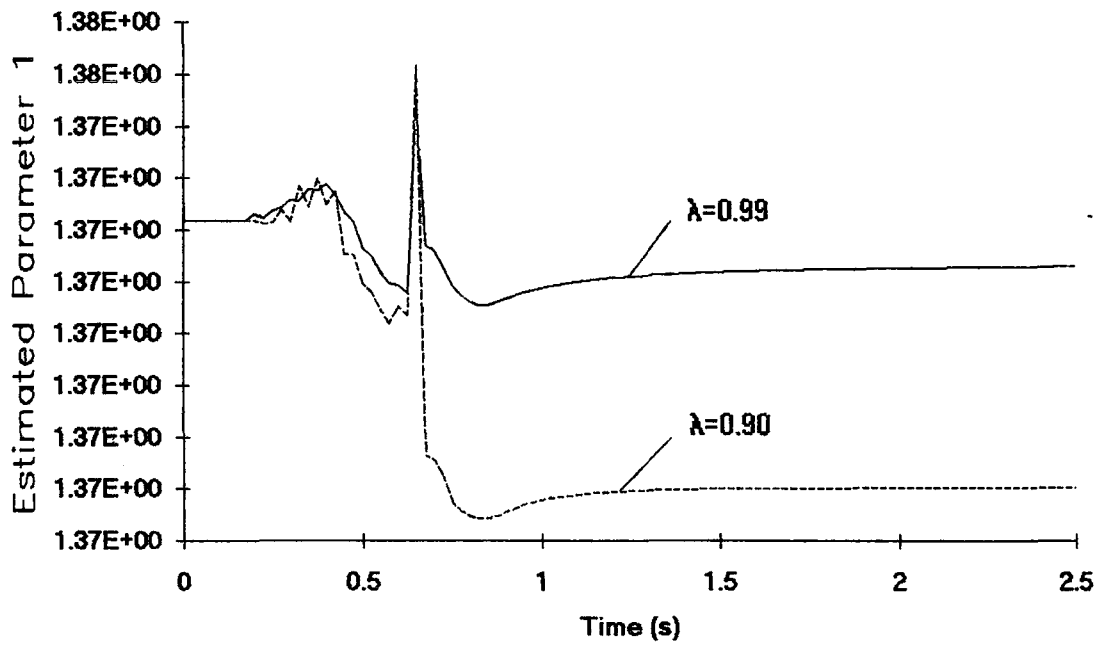


Figure 6.9. Effect of forgetting factor on parameter estimates in adaptive position control with $\rho=1000$ and $J_e(0)=0.0001 \text{ kgm}^2$;
 (a) Estimated Parameter 1,
 (b) Estimated Parameter 3.

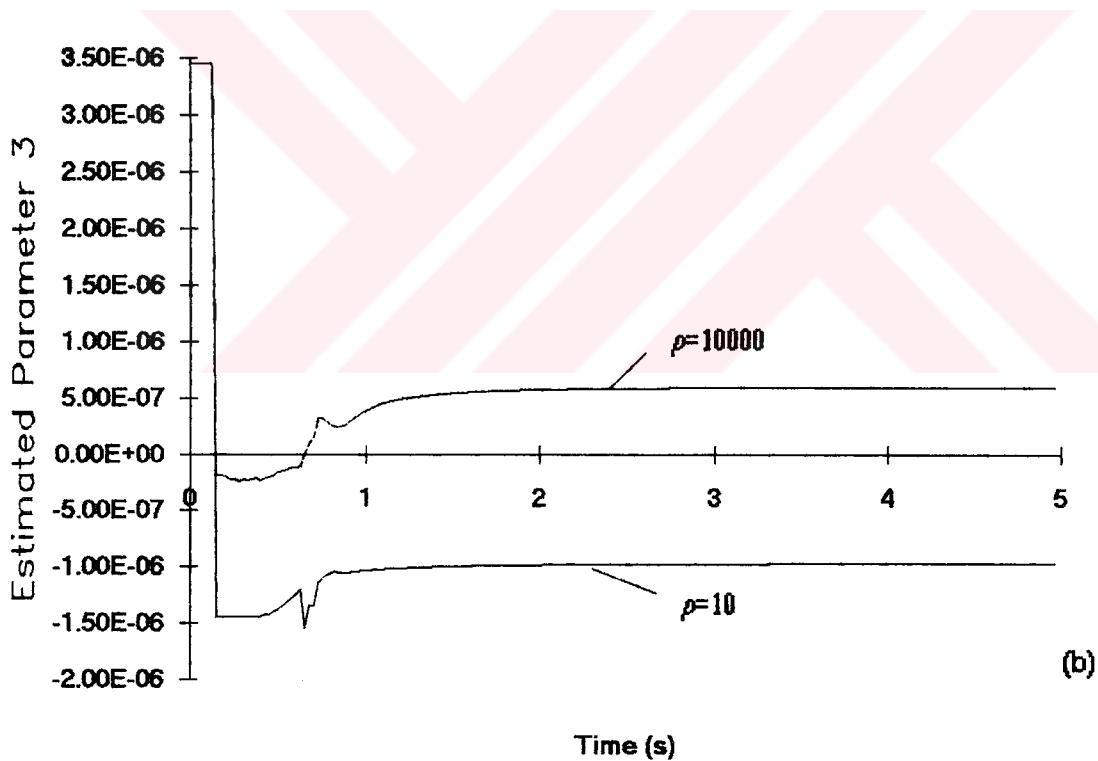
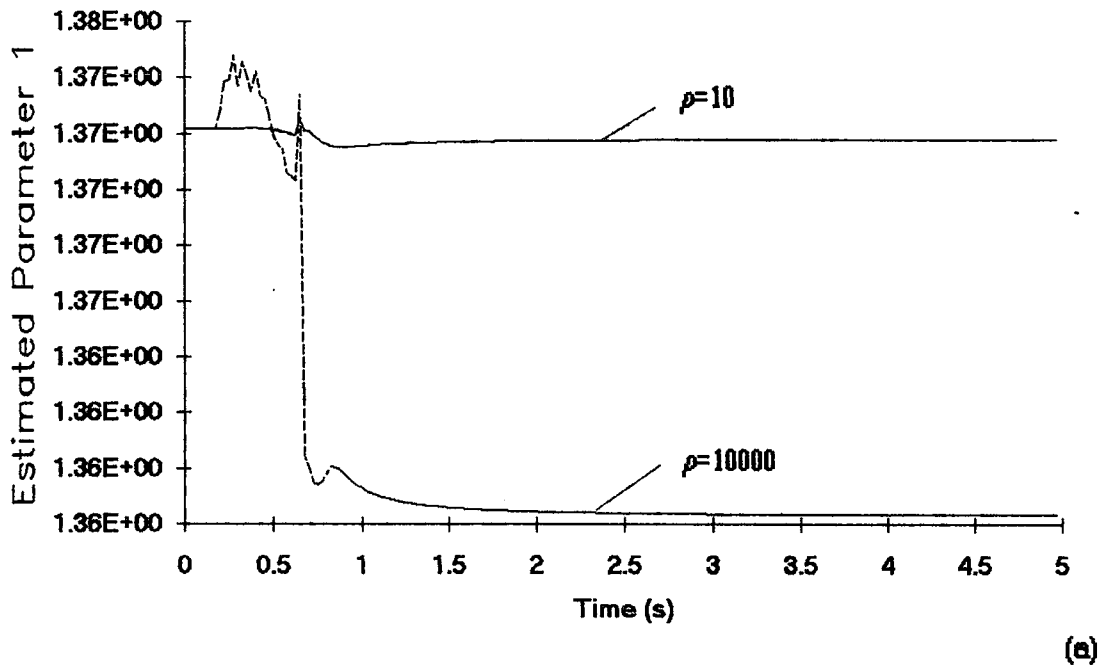
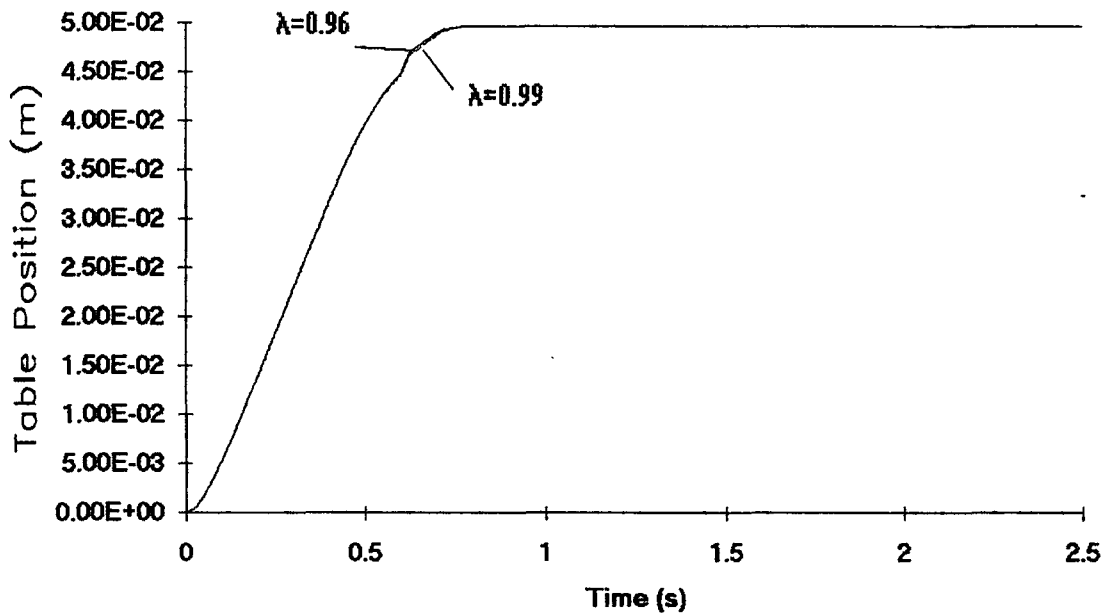
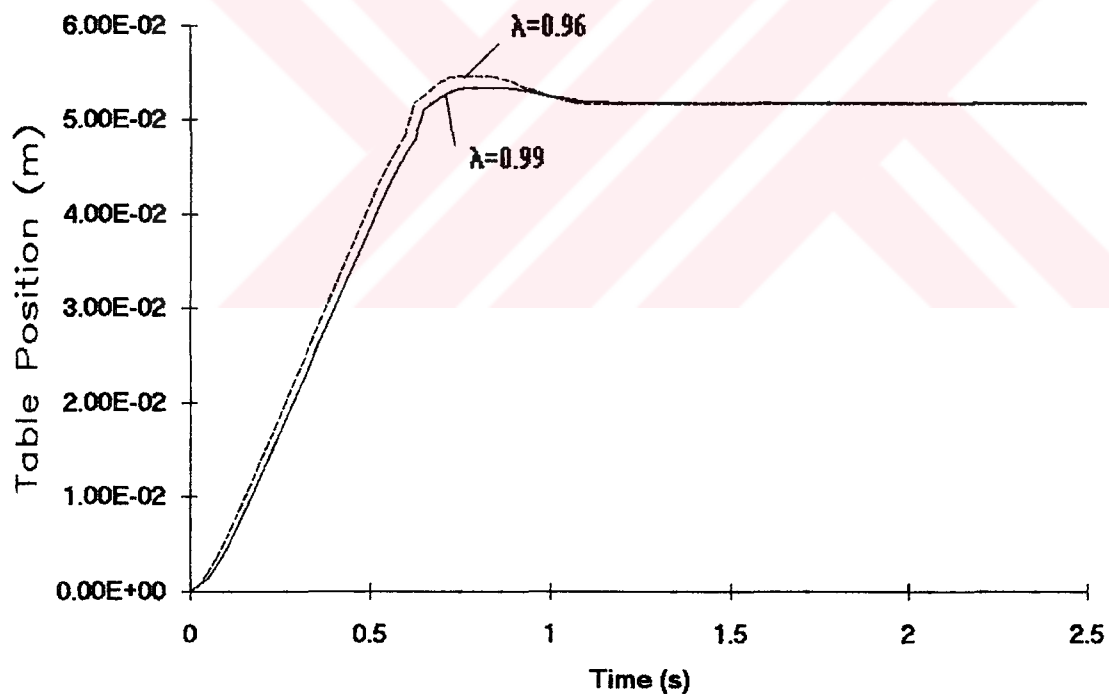


Figure 6.10 . Effect of initial covariance matrix on parameter estimates in adaptive position control with $\lambda=0.96$ and $J_e(0)=0.0001 \text{ kgm}^2$:
 (a) Estimated Parameter 1,
 (b) Estimated Parameter 3.



(a)



(b)

Figure 6.11. Effect of forgetting factor on the system response in adaptive position control with $\rho=1000$ and $J_e(0)=0.0001 \text{ kgm}^2$;
 (a) 1% overshoot
 (b) 10% overshoot.

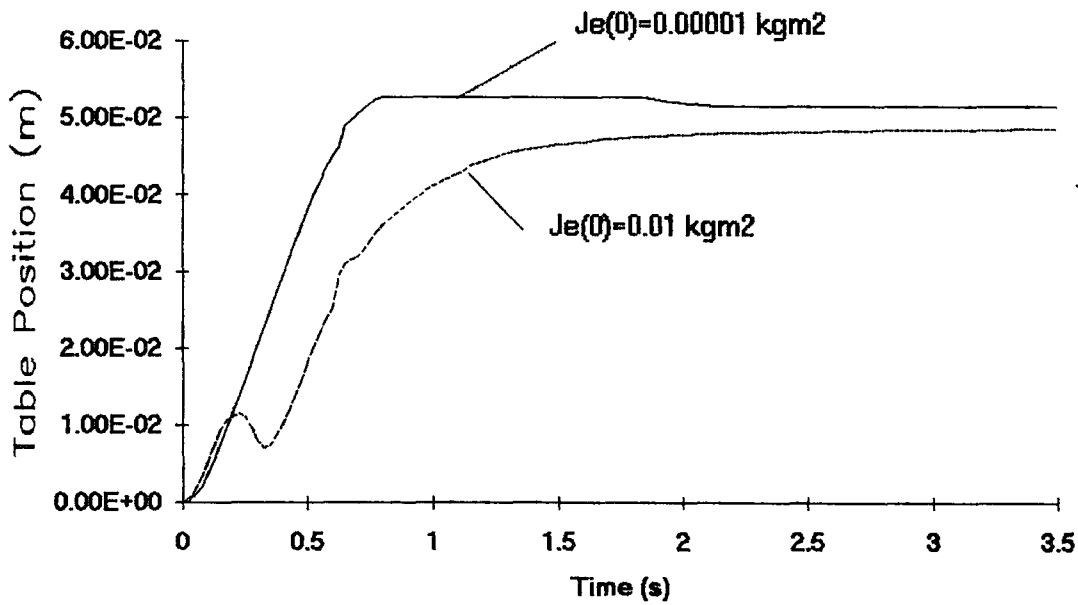


Figure 6.12. Effect of initial estimate on the system response in adaptive position control with $\lambda=0.96$ and $\rho=1000$.

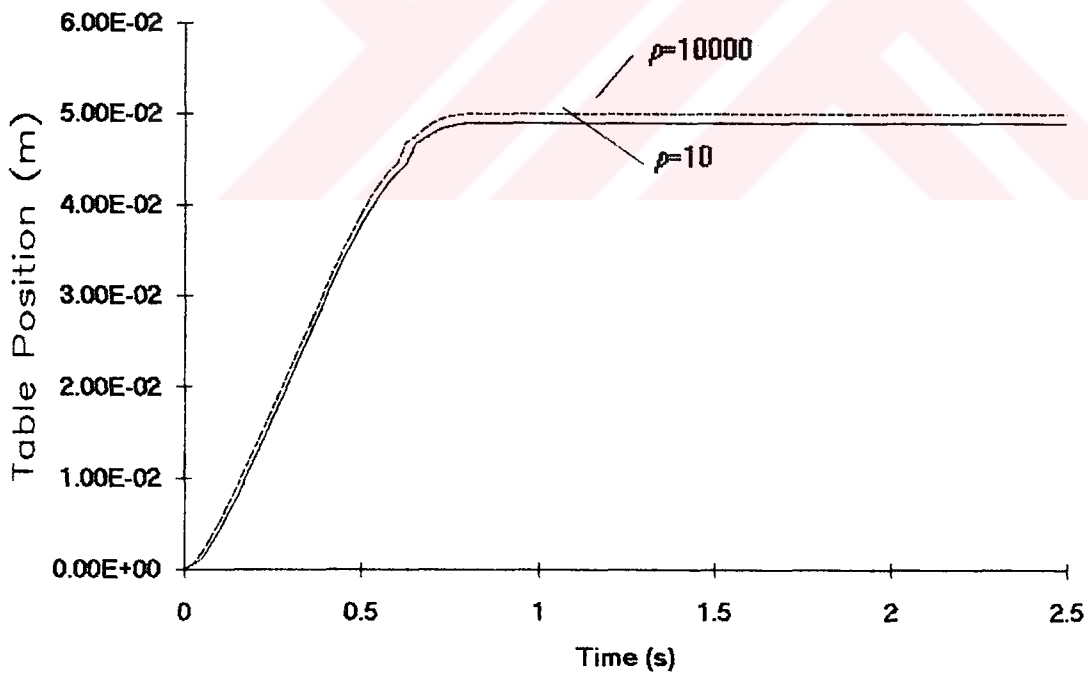


Figure 6.13. Effect of initial covariance matrix on the system response in adaptive position control with $\lambda=0.96$ and $J_e(0)=0.0001 \text{ kgm}^2$.

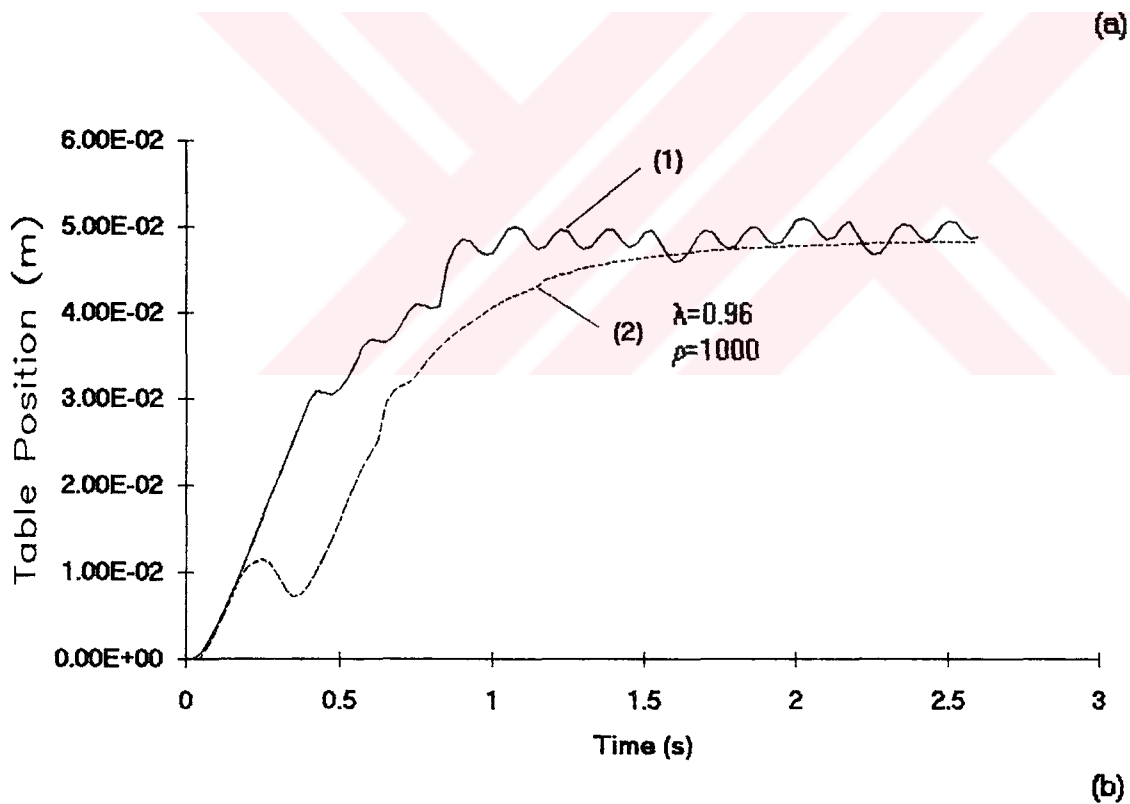
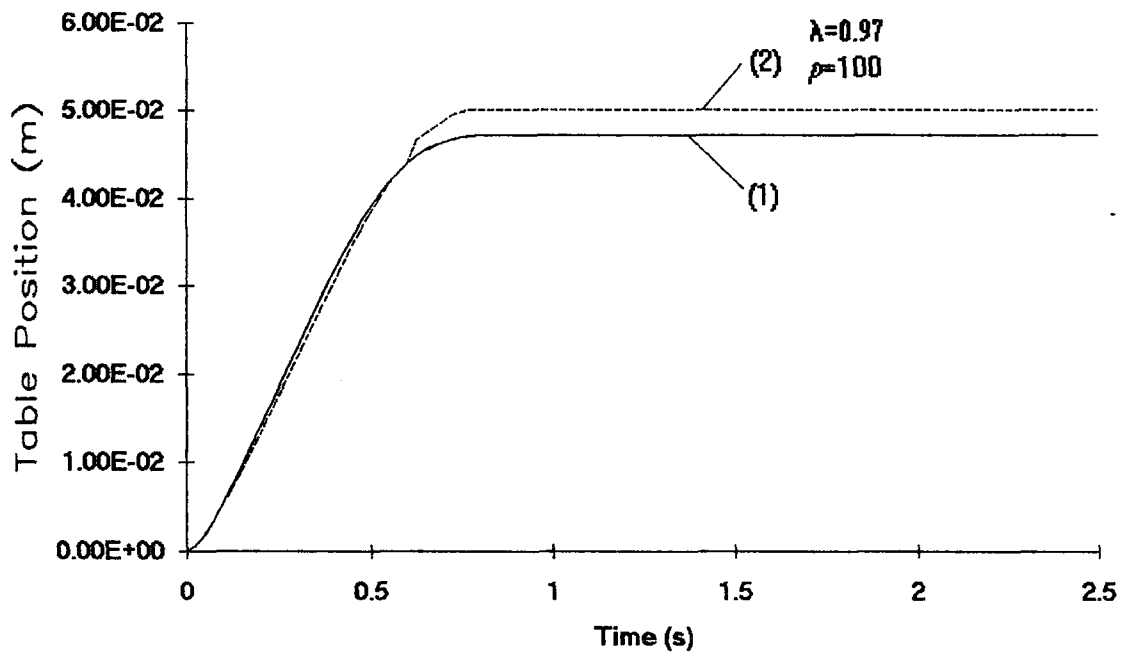


Figure 6.14. Comparison of ordinary PD pole assignment(1) and adaptive(2) controllers in position control of the table;
 (a) $J_e(0)=0.0001 \text{ kgm}^2$,
 (b) $J_e(0)=0.01 \text{ kgm}^2$.

BIBLIOGRAPHY

1. Ogata, K., "Modern Control Engineering", *Prentice-Hall*, 1970.
2. Aström, K.J., Wittenmark, B., "Adaptive Control", *Addison-Wesley*, 1989.
3. Landau, I.,D., "Adaptive Control: The Model Reference Approach", *M. Dekker Inc.*, N.Y., 1979.
4. Küçükoglu, M., "Parameter Adaptive Control of a single Axis Machine Table", M.Sc. Thesis, Boğaziçi University, 1991.
5. Özel, F.İ., Türkay, O.S., Küçükoglu, M., "Self-tuning Adaptive Control of a DC Motor Driven Single Axis Table", IFAC Proceedings on Automatic Control for Quality and Productivity, Volume 2, pp.517-525, 1992.
6. Aström, K.J., "Adaptive Feedback Control", Proceedings of the IEEE, Vol.75, pp.185-217, 1987.
7. Aström, K.J., "Theory and Applications of Adaptive Control-A Survey", Automatica, Vol.19,pp.471-486, 1983.
8. Narendra, K. S., Annaswamy, A.M., "Stable Adaptive Systems", *Prentice-Hall*, 1989.

9. Goodwin, G. C., Sin, K. S., "Adaptive Filtering, Prediction, and Control", *Prentice-Hall*, N.J., 1984.
10. Chalam, V.V., "Adaptive Control Systems, Techniques, and Applications", *M. Dekker Inc.*, N.Y., 1987.
11. Aström, K.J., Wittenmark, B., "Computer Controlled Systems", *Prentice-Hall*, N.J., 1984.
12. Aström, K.J., Wittenmark, B., "On Self-Tuning Regulators", *Automatica*, Vol.9, pp.185-199, 1973.
13. Aström, K.J., Borrisson, U., Ljung, L., "Theory and Applications of Self-Tuning Regulators", *Automatica*, Vol.13, pp.457-476, 1977.
14. Harris, C., Billings, S. (ed.s), "Self Tuning and Adaptive Control: Theory and Applications", *P. Peregrinus Ltd.*, London, 1985.
15. Clarke, D.W., Gawthrop, B.A., "Self-Tuning Control", *Proc. IEE*, Vol.126, pp.633-640, 1979.
16. Fargeon, C. (ed.), "The Digital Control of Systems", *V. Nostrand Reinhold*, N.Y., 1986.
17. Wellstead, P.E., Edmunds, J., Prager, D., Zanker, P., "Self-Tuning Pole/Zero Assignment Regulators", *Int. J. Control*, Vol.30, pp.1-26, 1979.

18. Türkay, O., Ulsoy, A.G., "Frequency versus Time Domain Parameter Estimation: Application to a Slot Milling Operation", Mechanical Systems and Signal Processing, Vol.2, pp.265-277, 1988.
19. Wittenmark, B., Aström, K.J., "Practical Issues in the Implementation of Self-Tuning Control", Automatica, Vol.20, pp.595-605, 1984.
20. Schwarzenbach, J., Gill, K., "System Modeling and Control", *E.Arnold*, London, 1984.
21. Kilitçiöglu, H., "An Algorithm for Adaptive Vibration Control of a Rigid Rotor Supported on Squeeze-Film Bearings", M.Sc. Thesis, Boğaziçi University, 1989.
22. Ulsoy, A.G., Koren, Y., "Principal Developments in the Adaptive Control of Machine Tools", ASME J. of Dyn. Systems, Meas., and Control, Vol.105, pp.107-112, 1989.
23. Ulsoy, A.G., Lauderbaugh, L., "Model Reference Adaptive Force Control in Milling", ASME J. of Engineering for Industry, Vol.111, pp.13-21, 1989.
24. Fortescue, T., "Implementation of Self Tuning Regulators with Variable Forgetting Factors", Automatica, Vol.17, pp.831-835, 1981.
25. Bittanti, S., Bolzern, P., Campi, M., "Convergence and Exponential Convergence of Identification Algorithms with Directional Forgetting factor", Automatica, Vol.26, No.5, pp.929-932, 1990.

26. Bittanti, S., Bolzern, P., Campi, M., "Adaptive Identification via Prediction-error Directional Forgetting Factor: Convergence Analysis", Int. J. of Control, Vol.50, No.6, pp.2407-2421, 1989.
27. Bierman, G.J., "Factorization Methods for Discrete Sequential Estimation", *Academic Press Inc.*, 1977.
28. Thornton, C., Bierman, G., "Filtering and Error Analysis via the UDU^T Covariance Factorization", IEEE Trans. on Automatic Control, Vol.AC-23, pp.901-907, 1978.
29. Gross, H. (ed.), "Electrical Feed Drives for Machine Tools", *Wiley & Siemens AG*, Berlin, 1983.
30. Benneth, S., "Real-Time Computer Control: An Introduction", *Prentice-Hall*, 1988.
31. Bulca, F., "Microcomputer Control of Speed and Position of a DC Motor", M.Sc. Thesis, Boğaziçi University, 1990.
32. Franklin, G.F., Powell, J.D., "Digital Control of Dynamic Systems", *Addison-Wesley*, Mass., 1980.
33. Neuman, C.P., Baradello, C.S., "Digital Transfer Functions for Microcomputer Control", IEEE Trans. on Systems, Man, and Cybern., Vol.SMC-9, pp.856-860, 1979.
34. Akçadağ, C., Unsal, S., "Design and Control of a CNC Model", Internal Report, Boğaziçi University, 1990.

35. Heidenhein, Dr. Johannes Heidenhein GmbH, LS 707 Incremental Linear Encoder Mounting and Operating Instructions.
36. Tunay, İ, "Computer Control for DC Motors", Internal Report, Boğaziçi University, 1990.



REFERENCES NOT CITED

1. Smith, R.J., "Circuits, Devices and Systems", *John Wiley and Sons*, 1984.
2. Tal, J., "Motion Control Applications", 1989.
3. High-Reliability Products, GE Solid State Data Book, Volume 1, 1988.