DESIGN OF AN ADAPTIVE PID CONTROLLER FOR IMPLEMENTATION ON THE MOTOROLA DSP56000

Thesis

Submitted to

Graduate Engineering & Research
School of Engineering

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree

Master of Science in Electrical Engineering

by

Mark R. DePoyster

UNIVERSITY OF DAYTON

Dayton, Ohio

August, 1993

DESIGN OF AN ADAPTIVE PID CONTROLLER FOR IMPLEMENTATION ON THE MOTOROLA DSP56000

APPROVED BY:

Malcolm W. Daniels, Ph.D.
Assistant Professor, Electrical Engineering
Committee Chairperson

Dana B. Rogers, Ph.D.
Professor, Electrical Engineering
Committee Member

John J. Westercamp, Ph.D.
Associate Professor, Electrical Engineering
Committee Member

Donald L. Moon, Ph.D.
Interim Associate Dean / Director
Graduate Engineering & Research
School of Engineering

Joseph Lestingi, D.Eng., P.E.
Dean
School of Engineering

ABSTRACT

DESIGN OF AN ADAPTIVE PID CONTROLLER FOR IMPLEMENTATION ON THE MOTOROLA DSP56000

Name: DePoyster, Mark, R.
University of Dayton, 1993

Advisor: Dr. Malcolm W. Daniels

An adaptive PID control algorithm is developed for implementation on a DSP chip. Basic PID theory is reviewed and a practical discrete-time PID algorithm is developed that includes a number of enhancements over the classical PID control algorithm. The advantages of using a DSP chip for control are also discussed. The PID control algorithm is then implemented on a Motorola DSP56000 processor. The architecture and instruction set of the DSP56000 are examined and the DSP56000-based controller is tested in the laboratory. Test results are presented and the DSP56000-based PID controller is shown to function as designed.

An adaptive PID algorithm is then developed based on the Simplified Self-Tuning Control (SSTC) model. The-SSTC PID controller is based on a self-tuning regulator structure that employs a recursive least-squares parameter estimation algorithm and a pole-cancellation control law design strategy. The process zeros are not canceled in the SSTC approach, allowing the algorithm to be used with non-minimum phase systems. The plant model is constrained to be second-order to force the general SSTC control algorithm

into a PID-like structure. The recursive least-squares estimation algorithm employs U-D cofactorization to guarantee numerical robustness. The estimator data is prefiltered to attenuate high frequency noise and low frequency disturbances to ensure that the parameter estimates are not biased. A dead-zone is also included in the estimation algorithm to prevent bursting due to parameter drifting. The adaptive PID algorithm is simulated and is shown to perform well for both setpoint tracking and disturbance rejection applications.

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to my advisor, Dr. Malcolm Daniels, for his direction and leadership in bringing this project to a conclusion. Dr. Daniels has been a tremendous source of encouragement and inspiration and sets a standard of excellence to which I will always strive to attain.

I also wish to express my thanks to my family for their patience and understanding during the past two and a half years. First of all, I want to thank my parents for their unending help and encouragement; to my mother, for her willingness to take the children to provide me with much needed solitude, and to my father, who instilled within me a love of learning. I would also like to thank my mother and father in-law for their support, not only to me, but to my wife during this period. Most of all, I wish to express my deepest appreciation to my wife for the countless sacrifices she has made in my many absences and for her proofreading of the documents contained herein. And to my three children, Jeremy, Jonathan and Jenna, I will do my best to someday repay the time that they gave to their father during the period of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$A(q)$	denominator of discrete plant model
$B(q)$	numerator of discrete plant model
$\mathbf{D}(t)$	covariance matrix in U-D cofactorization
$D(k)$	discrete derivative term
d	dimension of parameter vector $\theta(t)$
d_0	discrete derivative term coefficient
d_1	discrete derivative term coefficient
$e(t)$	error signal
$f(t)$	$\mathbf{U}^T(t-1)\phi(t)$
$g(t)$	$\mathbf{D}(t-1)f(t)$
$I(k)$	discrete integral term
K	continuous-time proportional gain
K'	discrete-time proportional gain
$\mathbf{L}(t)$	estimator gain matrix
N	maximum derivative gain
$\mathbf{P}(t)$	$\overline{R}^{-1}(t)$
q	quantization step size
q^{-1}	backward shift operator
$\overline{R}(t)$	covariance matrix
$\mathbf{R}(t)$	$\frac{1}{t}\overline{R}(t)$
s	scalar feedforward term in SSTC algorithm
T	sampling interval
T_d	derivative time
T_i	integral time
T_r	rise time
$\mathbf{U}(t)$	information matrix in U-D cofactorization
$u(t)$	controller output
$V_N(\theta)$	least-squares cost function
$w(t)$	reference input

$y(t)$	plant output
α_t	least-squares weighting coefficient
β	$e^{-T/\tau}$
$\beta(t)$	$\lambda + f^T(t)g(t)$
$\varepsilon(t)$	parameter or modeling error
$\phi(t)$	regression vector formed from observed data
$\phi_{xx}(\tau)$	autocorrelation function
γ	T_d/T
$\eta(t)$	measurement noise
λ	exponential forgetting factor
$\nu(t)$	load disturbance
θ	parameter vector of unknown coefficients
$\hat{\theta}(t)$	recursive estimate of θ based on data up to time t
ω_N	Nyquist frequency (rad/sec)
ζ	relative damping

LIST OF ABBREVIATIONS

A/D	analog-to-digital
ADM	Application Development Module
AGU	address generation unit
ALU	arithmetic-logic unit
ARMAX	autoregressive moving average with exogeneous input
ARX	autoregressive with exogeneous input
D/A	digital-to-analog
DSP	digital signal processing
EVB	Evaluation Board
ISR	interrupt service routine
LTI	linear time-invariant
LTV	linear time-varying
MAC	multiply-accumulator logic unit
MRAC	model reference adaptive control
PID	proportional-integral-derivative
PRBS	pseudo-random binary sequence
RLS	recursive least-squares
SCI	serial communications interface
SISO	single-input, single-output
SSI	synchronous serial interface
SSTC	simplified self-tuning control
STR	self-tuning regulator
VLSI	very large scale integration

CHAPTER I

INTRODUCTION

1.1 Introduction to PID Control

In recent years, significant advances have been made in the field of automatic control theory. Sophisticated computer-based design tools now allow even large scale multivariable control designs to be developed in a relatively short amount of time. A large number of processes, however, are still controlled by single loop, single-input, single-output (SISO) control systems. A block diagram of a basic SISO control loop is shown in Fig. 1,

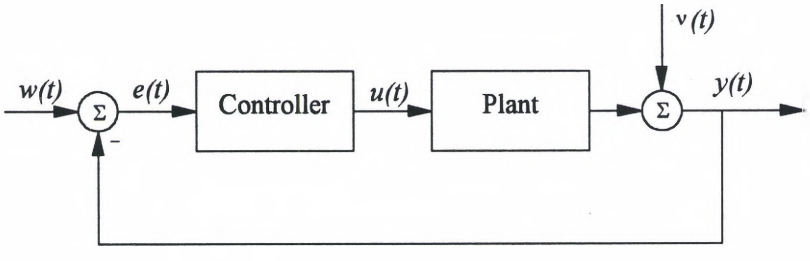

Figure 1. SISO error driven controller block diagram

where $w(t)$ is the reference input, $e(t)$ is the error signal, $u(t)$ is the control output, $y(t)$ is the plant output, and $v(t)$ is a load disturbance on the output. For many years, the

most widely used SISO controller has been the *proportional-integral-derivative* controller, commonly referred to as a PID controller. Its name is derived from the fact that the PID control algorithm is composed of three terms (the *proportional* term, the *integral* term and the *derivative* term) that are added together to form the control output signal, $u(t)$. The equation for the controller output is given as:

$$u(t) = K\left\{e(t) + \frac{1}{T_i}\int^{t} e(s)ds + T_d\frac{de(t)}{dt}\right\} \tag{1.1}$$

where:

K is the controller gain

T_i is the integral time

T_d is the derivative time.

PID controller implementation has gone through many stages of development over the years. The earliest controllers developed in the mid-1930s were pneumatic devices that used pressure capsules or vapor temperature bulbs for sensors and a combination of mechanical linkages and needle valves for adjusting the proportional, integral and derivative times. These pneumatic instruments were later replaced with electronic designs that eventually relied on operational amplifier circuits. With the advent of digital computers, and more recently, microprocessors, virtually all new PID controllers are implemented digitally.

PID control has proven to be robust and is used in a wide variety of commercial, industrial and military applications. PID control can be used in large-scale industrial applications where a single large computer controls hundreds of individual control loops. More commonly, however, PID controllers are implemented as stand-alone devices using microprocessors. General purpose PID controllers are commercially available *off-the-shelf*, equipped with options that allow them to be interfaced to a wide variety of sensors and actuators. PID algorithms are also frequently employed in embedded control applications, where the controller is designed into a larger system to perform a specific

function. Regardless of the application, the PID controller must be *tuned* to produce the desired closed-loop response in the plant output. As seen in equation (1.1), three parameters must be adjusted in order to tune the PID controller. They are the controller gain K, the integral time T_i and the derivative time T_d. Equation (1.1) may be expressed in Laplace transform form as:

$$u(s) = K\left(e(s) + \frac{1}{sT_i} e(s) + sT_d e(s) \right). \tag{1.2}$$

The controller transfer function may then be expressed as:

$$G_c(s) = \frac{u(s)}{e(s)} = K\left(\frac{1 + sT_i + s^2 T_i T_d}{sT_i} \right). \tag{1.3}$$

Equation (1.3) shows what takes place mathematically in the *s*-plane when the PID controller is *tuned*. The controller provides one pole at the origin (for removal of d.c. offsets) and two zeros that the designer can position by adjusting the parameters T_i and T_d. The two controller zeros are frequently used to cancel the dominant poles of a second-order plant.

PID controllers are normally used to control plants that are assumed to be linear and time-invariant. If a model of the plant is available, standard design methods such as pole-placement may be used to determine suitable PID controller parameters. If an accurate plant model is not available, empirical methods such as the Ziegler-Nichols (1942) techniques have been developed for *optimally* tuning PID controllers. Ziegler-Nichols procedures are often not employed in practice, however, as they can be time-consuming and can require operation of the plant near its stability limits. Many plants are therefore tuned by trial-and-error methods that can result in poorly controlled processes. Even if the controller is properly tuned, many plants that are assumed to be time-invariant are actually not. Plant parameters can change due to aging, component failure or environmental changes. Changes in the plant parameters may also be inherent to the process, as in some chemical reactions or an aircraft changing altitude in flight. Even

though changes in the plant parameters may occur at a relatively slow rate, the performance of the system can eventually degrade to the point where it is no longer acceptable. In some instances, the controller can be manually retuned to compensate for changes in the plant parameters. In other applications, however, the controller may not be accessible after it has been initially tuned, as in embedded control applications. In order to ensure robustness in cases where the system is not available after the initial tuning, the PID controller is *detuned* to compensate for changes in the plant parameters. This can result in suboptimal system performance. In situations where the plant is time-varying and constant-gain feedback control does not provide an acceptable solution, it would be desirable for the controller to *adapt* to changes in the plant parameters by continually updating its own parameters without any operator intervention. The solution to this problem is known as *adaptive control.*

1.2 Introduction to Adaptive Control

Research into the area of adaptive control began in the 1950s with the design of autopilots for high performance aircraft. Ordinary constant-gain feedback had difficulty dealing with the wide range of speeds and altitudes that such aircraft may have to operate in. Interest in adaptive control was somewhat diminished, however, after a disaster occurred in a flight test where adaptive control was employed. In the 1960s, the development of state space and stability theories broke down many of the barriers that impeded adaptive control research. Bellman's (1957) work on dynamic programming and Feldbaum's (1960) introduction of dual control theory also aided in the advancement of adaptive control theory. The idea that learning and adaptive control could be described in a common framework of recursive equations was put forth by Tsypkin (1971) during this period as well. Another extremely important area of research that was key to the

development of adaptive control theory in the 1960s was the subject of system identification and parameter estimation. A survey paper by Astrom and Eykhoff (1971) serves as an excellent reference to the research on system identification conducted during that period. In the 1970s, interest in adaptive control continued to increase as many different estimation schemes were combined with a variety of control law design methods to form adaptive controllers. In the late 1970s and early 1980s, correct proofs for stability of certain adaptive control models began to appear, although under very restrictive assumptions. The 1980s saw many applications of adaptive control systems even while the theory continued under development. According to Astrom (1987), by the spring of 1986, several thousand adaptive regulators were already in industrial use. With the development of stability proofs for the *ideal case* in 1980, the main thrust of the research shifted in the mid-1980s to robust adaptive control. Research in the area of stochastic adaptive control also intensified during that period. Today, adaptive control continues to be the subject of much research, as evidenced by the number of international conferences and journals dedicated to the subject.

Several different approaches to adaptive control have been proposed in the literature. Many of the concepts of the early adaptive schemes, such as the General Electric autopilot proposed by Marx (1959) and Marsik's (1970) adaptive regulator, are used in many of the later approaches. There are four heuristic schemes that encompass most of the current work in the field of adaptive control:

- Self-Oscillating Adaptive Systems
- Gain Scheduling
- Model Reference Adaptive Systems
- Self-Tuning Regulators

Each of these approaches will be explained briefly in the following paragraphs.

Self-oscillating adaptive systems represent some of the earliest work in adaptive control. A block diagram of the self-oscillating system proposed by Minneapolis-Honeywell (see Schuck, 1959) for an autopilot is shown in Figure 2.

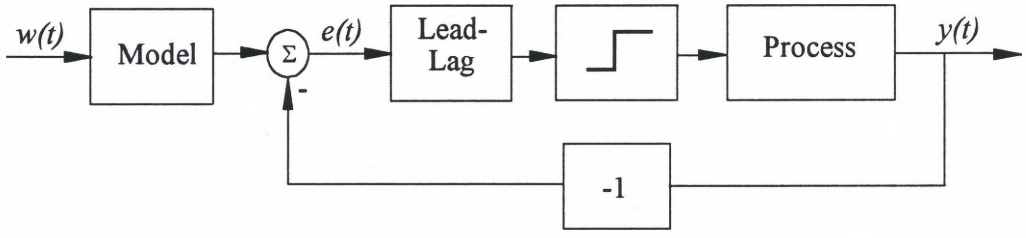

Figure 2. Block diagram of Self-Oscillating Adaptive System

The idea behind the system of Figure 2 is to have a feedback loop whose gain is as high as possible combined with feedforward compensation to produce the desired response to command signals. The high loop gain is maintained by the relay in the feedback loop. It can be shown that for signals whose frequencies are much lower than the limit cycle oscillation, the equivalent amplitude margin is approximately equal to 2. The system therefore continuously adjusts itself to yield an acceptable amplitude margin. The self-oscillating adaptive system has been used successfully in flight control systems for many different missiles. It has the drawback, however, in that experience has shown that pilots will usually notice the limit cycle, thus limiting its application to unmanned flight. Attempts have been made to reduce the amplitude of the limit cycle, but if the relay amplitude is too small, the response to command signals may be too slow. Other attempts have been made to quench the relay oscillations by the use of a dither signal with limited success.

A second method commonly used for adaptive control is *gain scheduling*. Like self-oscillating adaptive systems, gain scheduling was originally applied to the

development of flight control systems. It has also been successfully applied, however, in numerous industrial applications. A block diagram of a typical gain scheduling system is shown in Figure 3.

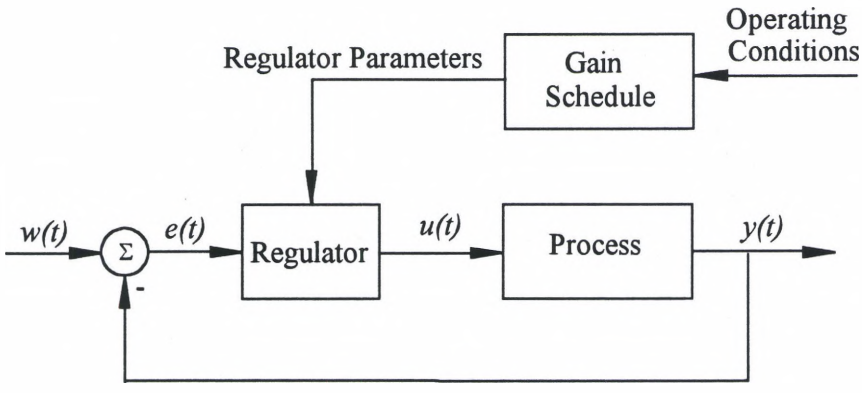

Figure 3. Block diagram of control system with Gain Scheduling

The system of Figure 3 monitors certain characteristics of the process denoted as *operating conditions* that relate to changes in the process dynamics. Regulator parameters are determined for a number of different operating conditions. Different sets of regulator parameters can then be activated as the operating conditions change. One of the advantages to gain scheduling is that the controller parameters can be changed very quickly in response to process parameters. The major drawback of gain scheduling, however, is that the control design process must be repeated for the number of parameter sets in the schedule. When extensive simulations are involved, this can be a time-consuming process. There has also been some controversy as to whether or not gain scheduling should be considered as a truly *adaptive* method, as the parameter changes are made in open-loop. Gain scheduling remains a viable solution, however, to many control problems, and it is easily implemented in computer-based control systems. It is still commonly used in flight control systems and has been used for controlling industrial robots and in various process control applications.

The third adaptive control scheme that will be considered is the *Model Reference Adaptive Controller*. The Model Reference Adaptive Controller (MRAC) was originally proposed by Whitaker (1958) at the Massachusetts Institute of Technology. A block diagram of the system is shown in Figure 4.

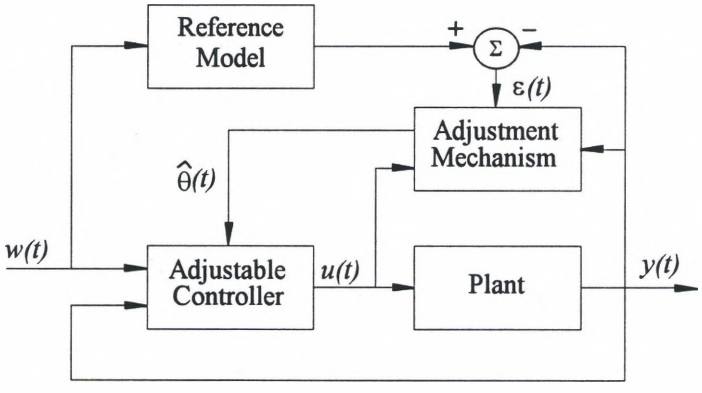

Figure 4. Block diagram of Model Reference Adaptive Controller

The MRAC model is essentially composed of two loops. The inner loop is an ordinary feedback loop consisting of an adjustable controller and the plant. An additional outer loop has been added to the system, which includes a reference model and a controller parameter adjustment mechanism. A reference model is chosen that produces the specified system response characteristics. As the actual controlled plant output differs from the output of the reference model, a model error signal $\varepsilon(t)$ is generated. The model error $\varepsilon(t)$ drives an on-line adjustment mechanism that updates the parameters $\hat{\theta}(t)$ to the adjustable controller in attempting to drive $\varepsilon(t)$ to zero. The adjustment mechanism is the key to the entire system and determining an appropriate one is not a trivial task. The parameter adjustment mechanism in Whitaker's original proposal has come to be known as the "MIT-rule".

Whitaker's original "MIT-rule" is given as:

$$\frac{d\hat{\theta}(t)}{dt} = -k\varepsilon(t)\text{grad}_{\hat{\theta}}\varepsilon(t) \tag{1.4}$$

where: $\hat{\theta}(t)$ is the controller parameter vector

$\varepsilon(t)$ is the model error

and k determines the parameter adaptation rate.

The "MIT-rule" given in equation (1.4) assumes that the controller parameters $\hat{\theta}(t)$ change at a much slower rate than the other system variables. (This assumption is almost always made in the analysis of adaptive control systems.) In order to make the model error $\varepsilon(t)$ small, the parameters are changed in the direction of the negative gradient of $\varepsilon^2(t)$. The "MIT-rule" has been shown to perform well if the parameter k is small. Difficulties arise, however, if k is too large relative to the size of the reference input. The stability of the system using the "MIT-rule" cannot therefore be guaranteed. Parks (1966) proposed an alternative adjustment mechanism based on Lyapunov's second method to deal with the stability problem of the "MIT-rule". In another important work, Monopoli (1973) eliminated the need to determine the derivative of the plant output that was required in Parks' work by using an *augmented error* signal instead of using the model error directly. A good bibliography on the subject of model reference adaptive control is given in Astrom and Wittenmark (1989).

MRAC was originally conceived for continuous-time systems. In fact, the first design of a MRAC for discrete-time SISO systems was not proposed until 1977 (Ionescu and Monopoli, 1977). In 1980, several important proofs of global stability for both continuous-time and discrete-time MRAC systems were presented by Egardt (1979, 1980), Goodwin, *et. al.* (1980), Morse (1980) and Narendra, *et. al.* (1980). These works

proved that if the plant is linear and time-invariant with unknown parameters, it can be stabilized based on the following assumptions:

- the plant zeros are stable
- the plant relative degree is known exactly and matches that of the reference model
- the sign of the high frequency gain is known
- an upper bound for the order of the plant is known.

Once proof of global stability for the ideal case had been established, much of the research in the 1980s concentrated on relaxing the above assumptions, making MRAC design more robust. Model reference adaptive control is generally considered to be one of the two most important branches of adaptive control and remains the subject of considerable research.

The last adaptive control scheme to be considered is known as the s*elf-tuning regulator.* The self-tuning regulator (STR) ranks with model reference adaptive control in importance in the adaptive control community. The STR was first conceived by R. E. Kalman (1958). Kalman divided the control design procedure into three basic steps:

I. Measure the dynamic characteristics of the process.

II. Specify the desired characteristics of the controller.

III. Put together a controller using standard elements which has the required dynamic characteristics.

Kalman's goal was "to design a machine which, when inserted in the place of the controller ... will automatically perform steps (I-III), and set itself up as a controller which is optimum in some sense." Armed with the dream of developing a machine that would eliminate the need for a control designer, Kalman designed a special-purpose computer to

implement the controller, but the project was plagued by hardware problems. His concept went on to serve as a model, however, for what is now known as the self-tuning regulator. A block diagram of the self-tuning regulator may be seen in Fig. 5.

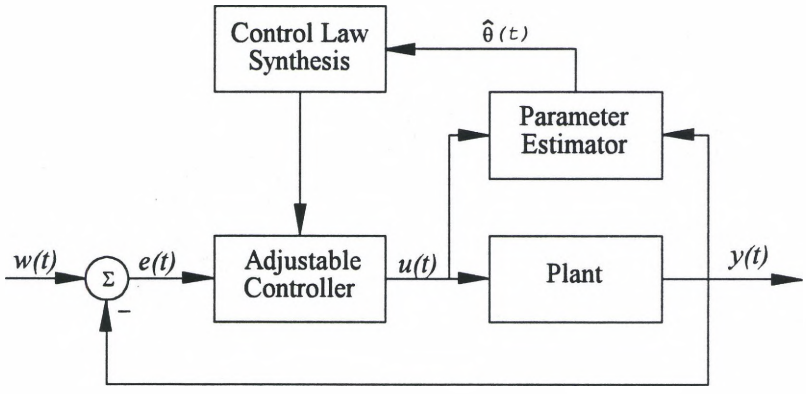

Figure 5. Block diagram of Self-Tuning Regulator

Like the model reference adaptive controller, the STR model is comprised of two loops with the inner loop consisting of the regulator and the process. The outer loop, however, is significantly different in the two approaches. In the model of Figure 5, the outer loop consists of a parameter estimator and an on-line controller design mechanism labeled "Control Law Synthesis" in the figure. The function of the estimator is to select parameters that best fit a preconceived *prejudice model* of the plant. The parameter updates are based on the dynamic characteristics of the plant as determined from the plant input and output signals. The *certainty equivalence principle* is then applied in which the uncertainties of the estimated parameters are ignored and the estimated parameters are assumed to be the true parameters of the plant. The estimated plant parameters are then used in a design calculation to determine the updated parameters for the controller. It is sometimes possible to reconfigure the controller so that the estimator parameters become

the controller parameters themselves, thus eliminating any intermediate calculations. This is referred to as a *direct* implementation. If intermediate calculations are required to obtain the controller parameters from the parameter estimates, it is referred to as an *indirect* implementation.

One of the advantages of the STR approach is that it offers considerable flexibility in implementation. Kalman's (1958) discrete-time design used a stochastic least-squares parameter estimation scheme with a deadbeat control law. Astrom and Wittenmark (1973) proposed a deterministic least-squares estimator used in conjunction with a minimum variance controller. Wellstead (1978) proposed using pole-zero assignment for STRs, an idea that was also expanded upon by Astrom and Wittenmark (1980). By the late 1970s, the STR had caught the interest of many researchers. Most of the estimators proposed for self-tuning controllers have included some sort of least-squares based algorithm. Sternby (1977) provided the first general proof for the convergence of the least-squares algorithm based upon martingale theory. Several years later, his work was extended to include adaptive control systems (see Sternby and Rootzen, 1982). To prove convergence of the estimated parameters, the later work employed a probabilistic approach known as "Bayesean embedding" which assumed the plant parameters to be random variables. The proof assumed, however, that the system is excited by white, Gaussian noise. Almost all of the other stability and convergence analyses have been based on finding a "stochastic Lyapunov function" (Kumar, 1990); however, the method has only been successful in a few isolated cases when the parameter estimator is either a stochastic gradient algorithm or a modified least-squares algorithm, and the control law is of the minimum variance type (see Goodwin, *et. al.* (1981), Becker, *et. al.* (1985), Kumar and Praly (1987) and Sin and Goodwin (1982)) .

Kumar (1990) points out that even today, very little is known about the behavior of recursive least-squares parameter estimate based adaptive control schemes from an

analytical perspective. While stability and convergence theories have been developed for idealized conditions, these conditions are often unrealistic in practice. Difficulties such as non-linearities, unmodeled dynamics and actuator saturation can arise that violate the assumptions made in the theoretical stability proofs making it necessary to circumvent the theoretical limitations when implementing self-tuning control. Much of the research on self-tuning control in the last decade has therefore been focused on *ad hoc* methods for making self-tuning algorithms more robust.

1.3 Adaptive PID Control

Due to the complexity of the algorithms involved, adaptive control research was severely hampered by a lack of adequate hardware in the 1950s and 1960s. As digital computers became less expensive and more powerful in the 1970s, adaptive control research began to flourish. Research again intensified as microprocessors appeared on the scene in the 1980s. The advent of the microprocessor offered the potential for widespread use of adaptive control in many of the applications where PID controllers have performed poorly due to non-linearities, time-varying plant parameters or inadequate tuning by process operators. Although the general adaptive methods described previously, such as MRAC and self-tuning control, could be readily implemented in microprocessors, many plant engineers and technicians have found the adaptive algorithms difficult to understand and have, hence, tended to reject them. For this reason, adaptive control algorithms have been developed that conform to a PID-like structure.

The development of adaptive PID algorithms has been approached in two fundamentally different ways. The first approach is to develop a controller that automatically *tunes itself* to the plant , either on a power-up condition or upon operator initiation. After the initial *tuning-in* period, the controller tuning parameters are fixed until

the automatic tuning process is manually reinitiated. Controllers designed for this type of application are known as *auto-tuning* controllers. The advantage of this method is that considerable time is saved in the tuning process and the system is often better tuned than when it is tuned by an operator. Astrom and Hagglund (1984b) proposed an automatic tuning method that uses relay feedback to determine the critical point on the Nyquist curve of the open-loop transfer function of the controlled plant. Once the critical point has been determined, any of several methods may be used to choose appropriate PID tuning parameters (see Astrom and Hagglund (1984a), Astrom and Hagglund (1988) and Hagglund and Astrom (1985)). This method has been used in many of the so-called *auto-tuning* PID controllers that are on the market today. Krause and Myron (1984) proposed a method involving pattern recognition of the process reaction curve of the open-loop system. The Foxboro EXACT controller is based on this technique. These and several other industrial implementations of auto-tuning PID control have been compared by Radke (1987). Although auto-tuning PID controllers have been extremely effective and are widely used in industry today, they cannot be considered truly *adaptive* controllers, since once the controller has been tuned, albeit automatically, the controller parameters are thereafter fixed. Human intervention is required to retune the controller if the process changes over time. The application of these devices is, in fact, limited to the same applications where conventional PID control is appropriate. The manual tuning process is merely eliminated.

The alternative approach to *auto-tuning* PID controllers is to employ one of the adaptive models described earlier, i.e., the MRAC or the STR model. The control law is modified, however, to conform to a PID-like structure. Established PID design techniques, such as pole-placement, can then be used to modify the controller design based on the current values of the estimated parameters. A number of variations of this approach have been proposed in the literature, most of which fall into the *self-tuning*

regulator category. Normally, some form of recursive least-squares algorithm is used to estimate the parameters of the plant on-line, and any one of a number of available control law design methods can be employed. One method used in the implementation of adaptive PID control employs a deadbeat control strategy. Such a design was proposed by Kurz, *et. al.* (1980). This method is suitable for low-pass processes with small dead times; however, it suffers the same limitation as conventional deadbeat control, i.e., the control output is directly dependent on the sample time. If the size of the control output must be limited, the sample time must be made proportionally large to compensate. Pole-placement techniques, such as those proposed by Wittenmark (1979), Astrom and Wittenmark (1980) and Wittenmark and Astrom (1980), are also used in adaptive PID applications. They provide the designer the advantage of being able to control the system response by selecting the locations of the closed-loop poles. Wittenmark and Astrom (1980) proposed still another adaptive PID method utilizing pole-zero cancellation. Similar approaches have been proposed by Lammers (1982) and Banyasz and Keviczky (1982). The pole-zero cancellation approach is a direct implementation and is thus, computationally efficient; however, it is limited to plants that can be modeled well by second-order, that are without dead time and that have stable poles and zeros.

Warwick, *et. al.* (1987) proposed a parameter adaptive control methodology which is essentially a pole cancellation method whose primary objective is servo tracking. Since the technique does not result in the cancellation of the process zeros, it can be applied to non-minimum phase systems as well. The technique, known as *simplified self-tuning control* (SSTC), is flexible enough to allow the basic algorithm to be modified into several interesting variations, one of which is an adaptive PID algorithm. A standard recursive least-squares algorithm is used as the parameter estimator. (An extended least-squares algorithm could be used if colored noise were present, but the authors contend

that the ordinary least-squares approach usually works sufficiently well, even if the noise is colored.)

The STR model has been shown to allow considerable flexibility in the implementation of parameter estimation algorithms and control law synthesis methodologies. In each case, however, it is necessary to impose certain restrictions on the plant model to accommodate the PID-like controller structure. If the plant can be adequately modeled under the restrictive assumptions, adaptive PID algorithms offer a potentially simpler solution to the adaptive control problem than some of the more general approaches.

1.4 DSP-based Control Implementation

Most adaptive control applications require parameter-based plant models, and most parametric identification schemes involve some form of a least-squares algorithm, although specific computation methods vary widely. Least-squares based techniques require iterative solutions that must be implemented on a digital computer. The algorithms can be math intensive and require considerable processing power if an iteration is to be completed during each sample interval. Some algorithms may require hundreds of multiplications and divisions in a single iteration, depending on the model order assumed. Until recently, microprocessor-based adaptive control was limited to applications requiring relatively slow sample rates, such as temperature control or control of chemical processes. Recent advances in very large scale integration (VLSI) methods have allowed integrated circuit manufacturers to develop a special class of microprocessors for processing digital signals. Features such as fast clock speeds (40 Megahertz), multiple large accumulators (up to 96 bits), hardware multipliers and Harvard architectures provide these processors with the ability to perform high precision operations at very high speeds. Consequently,

these special function microprocessors, known as DSP chips, are ideal candidates for adaptive control applications that require significantly faster sampling rates than are possible using general-purpose microprocessors.

1.5 Statement of the Problem:

A large number of SISO processes are currently controlled by PID controllers. PID controllers have proven to be robust in many applications and they are easily understood by control engineers and technicians. It may therefore be desirable in many cases for adaptive controllers to conform to a PID-like structure. Although conformance to a PID model puts constraints on control system performance, PID control remains a viable alternative for many adaptive applications. In addition, DSP chips offer many advantages over general-purpose microprocessors for the implementation of adaptive PID algorithm. The problem to be investigated may therefore be stated as follows:

Is it possible to implement an adaptive PID controller on a digital signal processing chip?

The results of the investigation of this problem are presented in this thesis.

1.6 Thesis Organization

The remainder of the paper is organized as follows: First, a practical discrete-time PID algorithm is developed in Chapter 2. A number of enhancements to the classical PID control algorithm given in equation (1.1) are incorporated to make the algorithm more robust. Several methods for designing with PID controllers are also explained. In Chapter 3, the discrete PID algorithm is implemented on the Motorola DSP56000 digital signal processing chip. The basic architecture and instruction set of the DSP56000 are explained, and the DSP56000 implementation of the PID algorithm is also analyzed. The PID control program is then tested in real-time using a Motorola ADS56000 development system tied to an Intel 80386-compatible computer modeled as the plant. The Zenith 80386-based computer is equipped with a National Instruments AT-MIO-16 analog I/O board that serves as the input/output interface for the plant. In Chapter 4, an adaptive PID control algorithm based on a self-tuning regulator model is developed. A recursive least-squares algorithm is selected for the parameter estimator. The parameter estimator is then incorporated into a PID version of the SSTC controller presented by Warwick, *et. al.* (1987). The estimation algorithm includes several *ad hoc* improvements to make it more robust. The adaptive PID algorithm is then simulated, the results of which are presented and analyzed in Chapter 5. The recursive least-squares parameter estimator is tested, followed by simulations of the general SSTC algorithm. The SSTC algorithm is then forced into a PID-like structure, which is also simulated and analyzed. Finally, conclusions are drawn in Chapter 6 and recommendations are made for future work.

CHAPTER II

DEVELOPMENT OF THE DISCRETE-TIME PID ALGORITHM

2.1 Introduction

Equation (1.1) described the *classical* form of the PID algorithm as:

$$u(t) = K\left\{e(t) + \frac{1}{T_i}\int^{t} e(s)ds + T_d\frac{de(t)}{dt}\right\}$$

Although the classical PID control law yields a controller that is suitable for a wide variety of applications, implementation in that form can result in some difficulties. Several enhancements to the basic algorithm have been proposed over the years to deal with some of the difficulties.

One problem arises from the fact that each of the terms of equation (1.1) acts on the error signal, $e(t)$, which is the difference between the reference input, $w(t)$, and the plant output, $y(t)$. The control law treats a change in the reference input or a disturbance on the output in an identical fashion. A large, sudden change in the setpoint will generate a large error signal from the derivative term known as *derivative kick* that could possibly drive the plant into a non-linear region. Derivative kick can be addressed by modifying the structure of the controller so that the derivative term is only acted upon by the plant

output and not the setpoint. The proportional term can also contribute to excessive overshoot in response to large setpoint changes. The problem of *proportional kick* can also be dealt with by modifying the controller structure so that the proportional term is also only acted upon by the output and not the error. The disadvantage of this method is that responses to setpoint changes can be somewhat sluggish. Another common difficulty, known as *integral windup*, occurs when a large setpoint change causes the control output to remain saturated for an extended time. The integral term continues to grow larger even after the output has saturated. When the output finally reaches the setpoint value and the error changes signs, it takes some time for the integral term to *unwind* and allow the control output to change signs as well. This results in a large overshoot in the plant output. Modification of the classical PID structure, combined with the addition of a non-linear limiting function, is one of the simpler methods for dealing with integral windup. Using the derivative term in its classical form can also lead to difficulties. In the form of equation (1.1), the nature of the derivative term, $T_d \frac{d}{dt} e(t)$, (or in Laplace transform form, $sT_d e(s)$), is that its gain increases with frequency. The derivative term tends to amplify higher frequency measurement noise, possibly leading to erratic behavior in the control output. The derivative term is therefore often filtered in order to limit its gain at high frequencies.

Once a suitable model has been developed for the PID controller, it must be converted to discrete-time form before it can be digitally implemented. Many different methods have been developed for approximating a differential equation by a difference equation. Each method possesses certain advantages and disadvantages over the others in how the frequency response of the discrete model compares to the frequency response of the continuous model. When considering control applications, a key criterion for evaluating the different methods is the amount of phase error generated by the discretization, as the stability of a control loop is directly related to the phase margin. A

discretization method commonly employed in control applications is a numerical integration technique known as the bilinear transformation, or Tustin's approximation. A major benefit of using the bilinear transformation is that it produces zero phase error in the discrete-time model. There are side effects, however, that become apparent when applying it to the derivative term, i.e., it can cause ringing in the output if T_d is too small. Frequently, therefore, the derivative term is converted using the backward difference integration method, while the bilinear transformation is employed to convert the proportional and integral terms to discrete-time form.

After developing a discrete-time PID model, the PID parameters must be selected to produce the desired response in the plant output. A number of different design methods have been developed for determining appropriate PID parameter values. Most of the techniques require working in the $s-$plane and then converting the continuous-time controller model to discrete form. If the process model is of a low enough order, the controller design can also be done directly in the $z-$plane using discrete-time pole-placement techniques. Discrete pole-placement requires a discrete-time model of the plant, however. Direct digital design methods are commonly used in implementing adaptive controllers since the parameter estimation algorithms are usually based on discrete-time models. Direct digital design methods do have a drawback, however, in that it can be difficult to translate the discrete-time control law to a PID structure.

There are a number of other issues that must be addressed when implementing a digital PID controller. As with any sampled data system, it is crucial to select an appropriate sampling rate for the controller. Too slow a sampling rate can result in poor control of the plant or even aliasing. Too fast a sampling rate can lead to numerical difficulties. A number of guidelines have been proposed for selecting appropriate sampling rates for digital controllers. Another important issue to consider when designing a digital control system is quantization. Analog signals must be quantized before they can

be processed by a computer or microprocessor. Too few bits in the A/D converter, for instance, can lead to problems such as limit cycling. Not only are signals quantized in digital control systems, but coefficients and parameters must be quantized as well. Quantization can be modeled as a noise source in the control system and may need to be considered in the design of the controller. Also, the length of the storage words in the computer memory must be considered. Too few bits in a memory word can result in an offset between the setpoint and the plant output. The word length required to achieve a given steady-state error can be calculated if the PID parameters and the sampling frequency are known.

In this chapter, a continuous-time PID algorithm is developed that incorporates many of the modifications discussed. The continuous-time algorithm is converted to discrete-time using the bilinear transformation and the ramifications of the conversion are discussed. Several PID controller design methods are also explained and some advantages and disadvantages of each method are pointed out. Finally, some key PID controller implementation issues are explored, including selection of an appropriate sampling rate and analysis of quantization error.

2.2 Development of the PID Terms

Proportional action can be described by the control law:

$$u(t) = Ke(t).$$

The control output, $u(t)$, is *proportional* to the error $e(t)$. To analyze the closed-loop behavior of the proportional term, consider a model of a linear time-invariant plant under simple proportional feedback. A block diagram of the system with a load disturbance $v(t)$ and measurement noise $\eta(t)$ is shown in Figure 6.

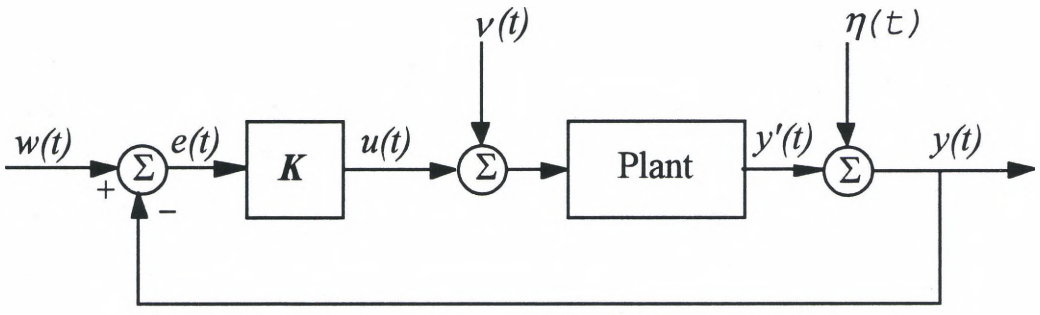

Figure 6. Block diagram of system with proportional feedback

If K_p is the D.C. gain of the plant, then the steady-state output of the plant is given as:

$$y' = \frac{KK_p}{1+KK_p}(w-\eta) + \frac{K_p}{1+KK_p}v \quad . \tag{2.1}$$

Astrom and Hagglund (1988b) make the following observations from equation (2.1):

1. A high controller gain, K, is desirable to make the plant output y' as close as possible to the setpoint, w.
2. A high controller gain, K, makes the system less sensitive to the load disturbance, v.
3. A high controller gain, K, makes the process more sensitive to the measurement noise η.
4. Measurement noise, η, responds to the system in the same way as the setpoint, w.

Although equation (2.1) does not address the dynamics of the system, it does point out that unless K or $K_p = \infty$, there will always be an offset between the output and the setpoint. In order for K or K_p to equal infinity, either the controller or the plant must contain an integrator. Since it is rare for a plant to contain an integrator, it is usually necessary for the controller to contain the integrator in order to remove the offset.

Referring again to equation (1.1), the control output resulting from the integral term being added to the proportional term is expressed as:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int^{t} e(s)ds \right). \tag{2.2}$$

Assuming the system is in steady-state with $u(t) = u_0$ and $e(t) = e_0$, equation (2.2) becomes:

$$u_0 = K\left(e_0 + \frac{e_0}{T_i}t \right).$$

As long as e_0 does not equal zero, u_0 will never remain constant. A positive error, no matter how small, always yields an increasing control output, and a negative error always causes a decreasing control output. Integral action, therefore, guarantees zero steady-state error for a step disturbance with the offset being removed in a time proportional to T_i (The smaller T_i is, the faster the integral removes the offset).

The use of integral control alone frequently leads to an unstable closed-loop system, as it adds 90 degrees of phase lag to the forward path. Integral control, therefore, is almost always used in combination with proportional control. Proportional plus integral control (commonly known as PI control) generally leads to a stable closed-loop system, providing T_i is appropriately chosen. However, even when used in combination with the proportional term, the integral term adds a degree of instability to the system. It is often necessary to add a stabilizing influence to counteract the effects of the integrator. This is accomplished with the addition of a derivative term.

Very often, process dynamics are such that there is a time lag from when a change is made in $u(t)$ to when a change is noticeable in the process output, $y(t)$. The response could be improved if the controller *predicted* changes in the process output. This is the function of the derivative term of equation (1.1). As explained by Astrom and Hagglund (1988b), the prediction is made by extrapolating the error along the tangent to the error curve, as seen in Figure 7.

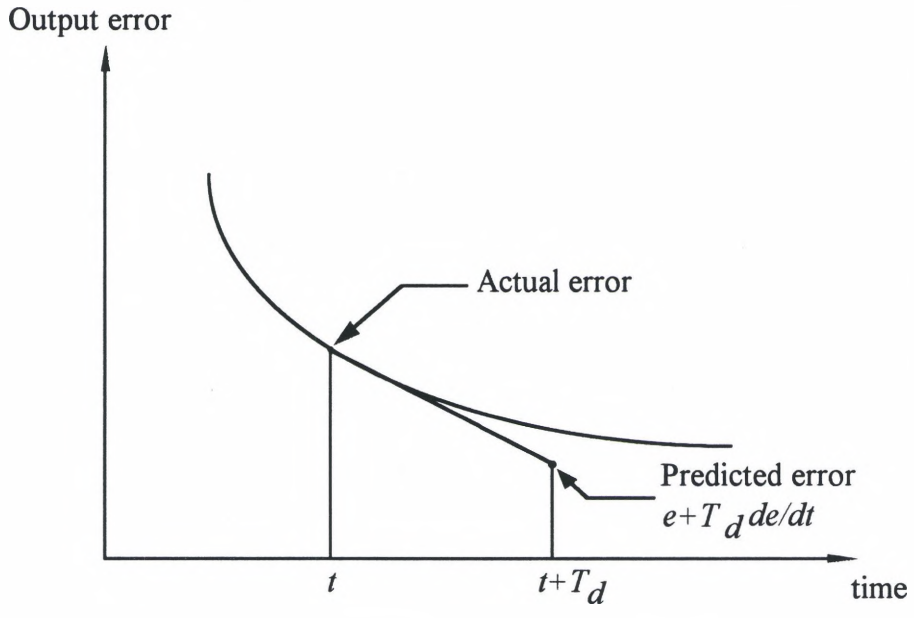

Figure 7. Graph depicting predictive action of derivative term

Derivative action is often required to control plants with excessive phase lag, such as processes of order greater than three or processes with large dead times (Clarke, 1984). The derivative term adds phase lead and therefore stability into the system. It is used less than the other PID terms in practice, however, as it can be difficult to tune and tends to amplify noise at high frequencies. This is one of several practical problems that must be addressed when implementing PID controllers. As stated earlier, the solution to many of these difficulties lies in the modification of the basic PID structure given in equation (1.1).

2.3 Development of the PID Controller Structure

A simplified block diagram of the classical PID structure is shown in Figure 8. Some of the difficulties encountered when implementing a PID controller in the form of Figure 8 have already been discussed. In this section, a practical PID controller structure is developed that addresses many of those difficulties.

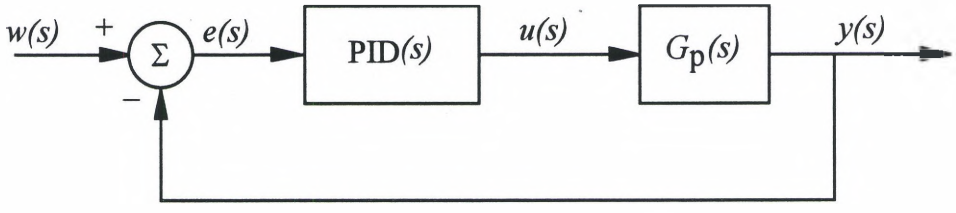

Figure 8. Simplified block diagram of *classical* PID structure

The proportional and integral terms from the classical PID algorithm given in equation (1.1) can be used to form a PI controller described by:

$$G_c(s) = K\left(1+\frac{1}{sT_i}\right) = K\left(\frac{1+sT_i}{sT_i}\right). \tag{2.3}$$

Equation (2.3) can be drawn as a simple lag in positive feedback as shown in Figure 9.

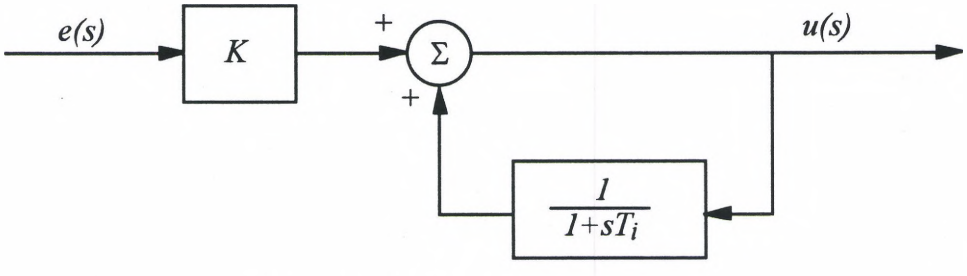

Figure 9. Block diagram of PI controller

In order to avoid the problem of derivative kick, the derivative term should only be acted on by the output and not the setpoint. One way of accomplishing this, referred to by

Astrom and Wittenmark (1990) as a *derivative-of output* controller, is shown in the model of Figure 10.

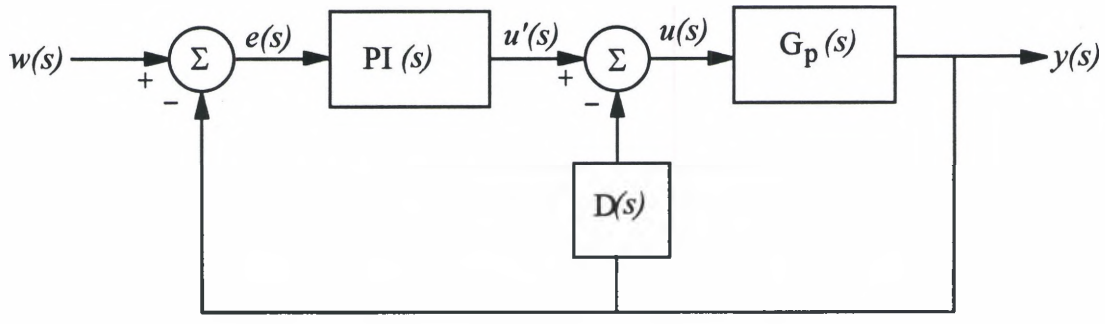

Figure 10. Block diagram of *Derivative-of-Output* PID model

Assuming $D(s) = K(1+sT_d)$, the control output from Figure 10 is determined to be:

$$u(s) = K\frac{(1+sT_i)}{sT_i}w(s) - \left\{K\left(\frac{1+2T_is+T_iT_ds^2}{sT_i}\right)\right\}y(s) \qquad (2.4)$$

From equation (2.4), it can be seen that the reference input drives a PI response from the controller. The derivative term, however, is acted upon only by the output and derivative kick is thus avoided. The parameters of equation (2.4) relate to the parameters of the classical PID model, K', T_i', T_d', in the following manner:

$$\begin{aligned} K' &= K \\ T_i' &= 2T_i \\ T_d' &= T_d/2 \end{aligned} \qquad (2.5)$$

The parameters of the derivative-of-output controller are similar to those of the *classical* controller, as seen in equation (2.5). The parameters T_i and T_d do not interact as they do in the some *interacting* forms (see Clark, 1984).

The derivative-of-output controller is an improvement over the classical PID structure in that it eliminates derivative kick without introducing interaction between the T_i and T_d terms. It does not, however, address the problem of excessive overshoot resulting from proportional kick. An alternative model, referred to by Astrom and

Wittenmark (1990) as the *setpoint-on-I-only* controller, addresses proportional kick by changing the PID model so that, like the derivative term, the proportional term is acted upon only by the output and not the setpoint. The setpoint-on-I-only controller can be tuned to react quickly to load disturbances; however, the controller's reaction to setpoint changes can be somewhat sluggish, relying only on the integration of the error signal to drive the plant.

The derivative term from the derivative-of-output PID model can be expressed as:

$$u_d(s) = -KsT_d y(s) \tag{2.6}$$

The gain of the derivative term given in equation (2.6) is expressed as:

$$\left|-Kj\omega T_d\right| = K\omega T_d \tag{2.7}$$

From equation (2.7), it can be seen that the gain of the derivative term grows without bound as ω increases. This implies that for large ω (i.e., outside the bandwidth of the plant), process and measurement noise are the dominating factors in driving the derivative term. This problem is somewhat alleviated by filtering the derivative term to limit its gain at high frequencies. A common method of filtering the derivative term is given in equation (2.8):

$$D(s) = -K\left(\frac{sT_d}{1+s\frac{T_d}{N}}\right) \tag{2.8}$$

where N is the maximum allowable gain of the derivative term. (Astrom and Wittenmark (1990) report that N is typically set in the range of 3-20.) The magnitude response of the filtered derivative term with K=1, $N = 10$ and $T_d = 0.1$ is shown compared to the unfiltered derivative term response in Figure 11.

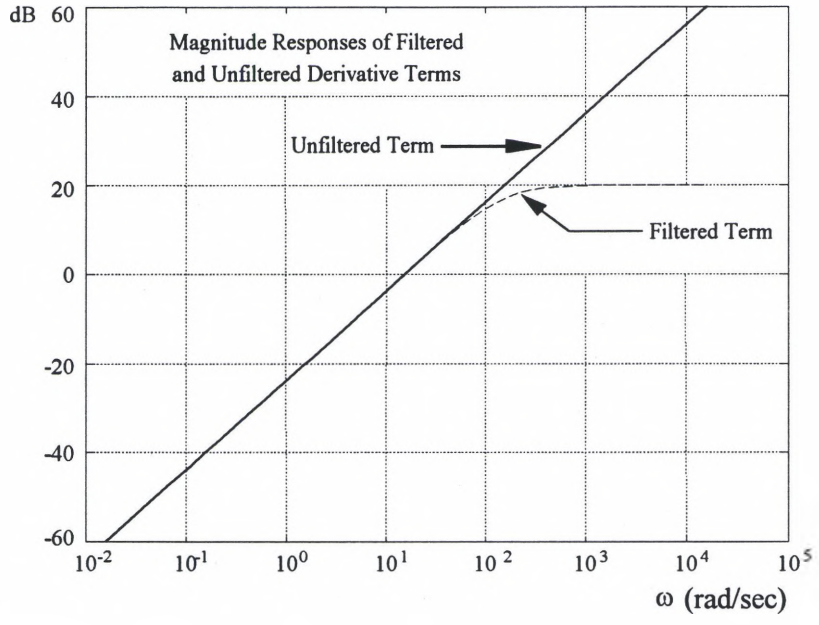

Figure 11. Magnitude responses of filtered vs. unfiltered derivative terms

The cutoff frequency of the filtered term occurs at N/T_d radians with a maximum gain of 20 dB (N=10.0). Figure 11 demonstrates how limiting the gain of the derivative term at high frequencies minimizes the effect of measurement noise on the control output. A block diagram of the modified continuous-time PID model is shown in Figure 12.

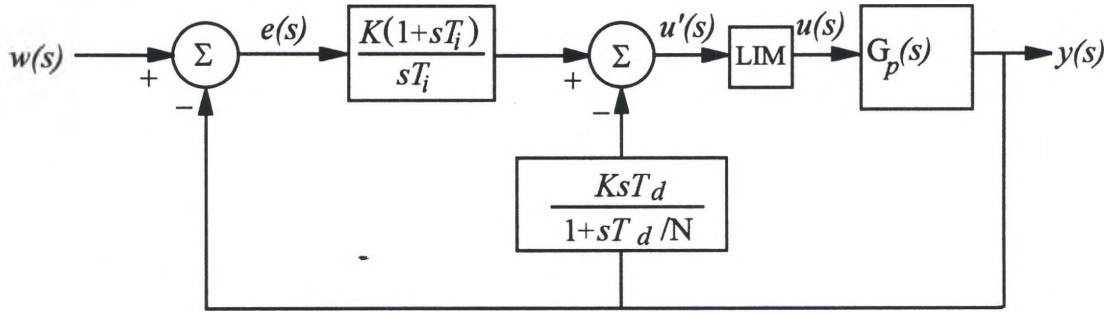

Figure 12. Block diagram of modified PID controller

The PID model of Figure 12 is a derivative-of-output controller in which the derivative term is acted upon only by the output, thus eliminating the problem of

derivative kick. The derivative term filter also minimizes the effect of measurement noise on the control output.

2.4 Discretization of the PID Algorithm

In order to digitally implement a continuous-time system, it must first be converted to discrete-time form. Many approaches have been developed for converting continuous-time transfer functions to discrete equivalents. Franklin, Powell and Workman (1990) have divided these methods into three categories:

1. hold equivalence
2. zero-pole mapping
3. numerical integration.

In essence, the goal of each of the approaches is identical; i.e., to convert a differential equation to a difference equation that approximates the differential equation. Franklin, Powell and Workman (1990) analyze the merits of each procedure in terms of its application to digital control. The most common methods for discretizing continuous-time controller models come under the heading of numerical integration. Three approaches are commonly used to approximate an integral numerically. They are the *forward rectangular rule*, the *backward rectangular rule* and the *trapezoidal rule*, referring to how the incremental value of the area under the curve is calculated in each case. With forward integration, the left half of the s-plane maps onto the entire z-plane, including the area outside the unit circle defined by $|z| = 1$. A stable continuous-time system can therefore be made unstable using the forward rectangular rule. The backward rectangular rule, however, maps the left half of the *s*-plane into a region entirely within the unit circle in the

z-plane, thus guaranteeing stability in the transformation. The trapezoidal rule (also known as the *bilinear transformation*) is unique in that the left half of the *s*-plane maps into the entire stable region of the *z*-plane, i.e., inside the unit circle. The $j\omega$ axis in the *s*-plane maps directly onto the unit circle in the *z*-plane. This leads to a significant amount of distortion, since the $j\omega$ axis in the *s*-plane ranges from $-\infty$ to $+\infty$ while the unit circle in the *z*-plane ranges from 0 to 2π. The frequency distortion resulting from this transformation can be significantly reduced by employing a technique known as *pre-warping*. (Franklin, Powell and Workman (1990) discuss the methodology at some length.) This method can only be applied, however, if the critical frequency is known in advance.

Although any of these approaches may be used to convert a continuous-time controller model to discrete-time form, the methods most generally used for control applications are the backward rectangular rule (sometimes referred to as the backward difference) and the bilinear transformation, since both methods guarantee stable poles in the *z*-plane if the poles in the *s*-plane are stable. Clarke (1984) analyzes the frequency response characteristics of the two methods in terms of a normalized frequency. If only frequencies up to the Nyquist frequency are considered, i.e.,

$$\omega_N = \frac{\pi}{\mathrm{T}},$$

the frequency with respect to the Nyquist frequency can be normalized by letting $x = \dfrac{\omega}{\omega_N}$ where x is the normalized frequency ranging from $0 \rightarrow 1$. The gain and phase error for $0 < x < 1$ are summarized in Table 1.

Table 1. Phase and gain error from backward rectangular rule

Freq x	Gain error	Phase error
0.1	.9959	-9
0.25	.9745	-22.5
0.50	.9003	-45
0.75	.7842	-67.5
1.00	.6366	-90

Table 1 demonstrates the criticality of the sampling rate relative to gain and phase error. The phase error becomes significant for $\omega > 0.1\omega_N$, indicating that sampling rate should be at least ten times the Nyquist frequency. The gain and phase error resulting from the bilinear transform are summarized in Table 2.

Table 2. Phase and gain error from bilinear transform

Freq x	Gain error	Phase error
0.1	1.0083	0
0.25	1.0548	0
0.50	1.2732	0
0.75	2.0492	0
1.00	∞	0

In Table 2, the phase error is zero up to the Nyquist frequency, and the gain error does not become significant until $\omega > 0.5\omega_N$. Since phase error is generally more important than gain error in control applications, the bilinear transformation appears to be the better

choice for conversion of the continuous-time controller to discrete-time. Having developed a practical PID model and selected the bilinear transformation as the method of discretization, the continuous-time PID model must now be transformed to a discrete-time model.

The following development follows Clarke (1984). From the PI regulator given in equation (2.3), the integral term from Figure 9 is given as:

$$\mathrm{H}(s) = \frac{1}{1 + s\mathrm{T}_i} = \frac{1/\mathrm{T}_i}{1/\mathrm{T}_i + s}$$

Solving for the exact z-transform of $\mathrm{H}(s)$ yields:

$$\mathrm{H}(z) = (1 - z^{-1})\mathrm{Z}\left\{\frac{\mathrm{H}(s)}{s}\right\}$$

$$L^{-1}\left\{\frac{\mathrm{H}(s)}{s}\right\} = 1 - e^{-\mathrm{T}/\eta}$$

$$\therefore \mathrm{H}(z^{-1}) = \frac{(1 - e^{-\mathrm{T}/\eta})z^{-1}}{1 - e^{-\mathrm{T}/\eta}z^{-1}}.$$

Substituting β for $e^{-\mathrm{T}/\eta}$ gives:

$$\mathrm{H}(z^{-1}) = \frac{(1-\beta)z^{-1}}{1-\beta z^{-1}}. \tag{2.9}$$

Equation (2.9) is the exact z-transform of the integral term of Figure 9. Placing equation (2.9) in feedback with gain K' yields:

$$U(z^{-1}) = K'E(z^{-1}) + \frac{(1-\beta)z^{-1}}{1-\beta z^{-1}}U(z^{-1}). \tag{2.10}$$

Solving for $U(z^{-1})$ gives:

$$U(z^{-1}) = K'\left\{1 + \frac{(1-\beta)z^{-1}}{(1-z^{-1})}\right\}E(z^{-1}). \tag{2.11}$$

Equation (2.11) is an expression for the exact z-transform of the PI controller where $\beta = e^{-\mathrm{T}/\eta}$.

The PI controller from Figure 9 can be expressed in Laplace transform form as:

$$G_c(s) = K\left\{\frac{1 + sT_i}{sT_i}\right\}.$$

Applying the bilinear transformation to G_c yields:

$$G_c(z^{-1}) = K\left\{\frac{1+\frac{2(1-z^{-1})}{\mathrm{T}(1+z^{-1})}T_i}{\frac{2(1-z^{-1})}{\mathrm{T}(1+z^{-1})}T_i}\right\} = K\frac{(2T_i+\mathrm{T})}{2T_i}\left\{1+\frac{2\mathrm{T}z^{-1}}{(2T_i+\mathrm{T})(1-z^{-1})}\right\}. \tag{2.12}$$

Comparing equation (2.11) to (2.12):

$$(1-\beta) = \frac{2\mathrm{T}}{(2T_i+\mathrm{T})} \therefore \beta = \frac{2T_i-\mathrm{T}}{2T_i+\mathrm{T}} \text{ and } K' = K\left(\frac{2T_i+\mathrm{T}}{2T_i}\right) = K\left(1+\mathrm{T}/2T_i\right). \tag{2.13}$$

Equation (2.12) is an approximation of the exact z-transform of the PI controller expressed in equation (2.11) with equation (2.13) giving the discrete approximation for β.

From equation (2.10):

$$\begin{aligned} U(z^{-1}) &= K'E(z^{-1}) + \frac{(1-\beta)z^{-1}}{1-\beta z^{-1}}U(z^{-1}) \\ U(z^{-1}) &= K'E(z^{-1}) + z^{-1}U(z^{-1}) - K'\beta z^{-1}E(z^{-1}) \end{aligned} \tag{2.14}$$

Taking z^{-1} as the backwards shift operator, equation (2.14) becomes:

$$u(k) = K'e(k) + u(k-1) - K'\beta e(k-1). \tag{2.15}$$

The control output in equation (2.15), $u(k)$, can be split into the proportional part:

$$P(k) = K'e(k), \tag{2.16}$$

and the integral part:

$$I(k) = u(k-1) - K'\beta e(k-1).$$

Simplifying the integral term gives:

$$\begin{aligned} I(k) &= K'e(k-1) + I(k-1) - \beta K'e(k-1) \\ &= \beta I(k-1) + (1-\beta)u(k-1) \end{aligned} \tag{2.17}$$

or,

$$I(k+1) = \beta I(k) + (1-\beta)u(k). \tag{2.18}$$

Equation (2.18) then becomes an expression for the integral term in terms of β, which can then be approximated to any degree one desires.

Next, the derivative term is transformed to discrete-time form. The filtered derivative term was given previously as:

$$D(s) = -K\left(\frac{sT_d}{1+s\frac{T_d}{N}}\right). \tag{2.19}$$

Applying the bilinear transform to equation (2.19) yields:

$$D(z^{-1}) = -K'\left(\frac{\frac{2T_d(1-z^{-1})}{\mathrm{T}(1+z^{-1})}}{1+\frac{T_d}{N}\left(\frac{2(1-z^{-1})}{\mathrm{T}(1+z^{-1})}\right)}\right)Y(z^{-1}),$$

Letting $\gamma = \frac{T_d}{\mathrm{T}}$,

$$D(z^{-1}) = -\frac{K'2\gamma(1-z^{-1})}{1+\frac{2\gamma}{N}+\left(1-\frac{2\gamma}{N}\right)z^{-1}}Y(z^{-1})$$

$$\left(1+{}^{2\gamma}\!/_{N}\right)D(z^{-1})+\left(1-{}^{2\gamma}\!/_{N}\right)z^{-1}D(z^{-1}) = -K'2\gamma Y(z^{-1})+K'2\gamma z^{-1}Y(z^{-1}).$$

Taking z^{-1} as the backwards shift operator yields:

$$D(k) = \frac{\left({}^{2\gamma}\!/_{N}-1\right)}{\left({}^{2\gamma}\!/_{N}+1\right)}D(k-1)+\frac{2K'\gamma}{\left({}^{2\gamma}\!/_{N}+1\right)}[y(k-1)-y(k)].$$

Setting:

$$d_0 = \frac{\left({}^{2\gamma}\!/_{N}-1\right)}{\left({}^{2\gamma}\!/_{N}+1\right)} \text{ and } d_1 = \frac{2K'\gamma}{\left({}^{2\gamma}\!/_{N}+1\right)}$$

gives:

$$D(k) = d_0D(k-1)+d_1[y(k-1)-y(k)]. \tag{2.20}$$

Equation (2.20) is the discrete-time expression for the filtered derivative term in terms of $\gamma = \frac{T_d}{\mathrm{T}}$.

The complete PID algorithm can now be formed by combining equations (2.16), (2.18) and (2.20) to give:

$$\begin{aligned} u(k) &= P(k)+I(k)+D(k) \\ u(k) &= K'e(k)+I(k)+d_0D(k-1)+d_1[y(k-1)-y(k)] \end{aligned} \tag{2.21}$$

with:

$$I(k+1) = \beta I(k) + (1-\beta)u(k) \tag{2.22}$$

where:

$$K' = K\left(1 + \mathrm{T}/2T_i\right),\ \beta = \frac{2T_i - \mathrm{T}}{2T_i + \mathrm{T}},\ d_0 = \frac{\left(\frac{2T_d}{N\mathrm{T}} - 1\right)}{\left(\frac{2T_d}{N\mathrm{T}} + 1\right)},\ \text{and}\ d_1 = \frac{2K'T_d/\mathrm{T}}{\left(\frac{2T_d}{N\mathrm{T}} + 1\right)}. \tag{2.23}$$

It only remains to determine the appropriate values for the parameters K, T_i, T_d, T and N.

Many different methods have been developed for selecting PID controller parameters to yield a specified plant performance. Some of the more commonly used approaches to PID controller design are discussed briefly in the next section.

2.5 Designing with PID Controllers

PID control is used for a wide variety of applications. A PID algorithm may be used in a custom design to control a specific plant. In such cases, a mathematical model of the plant is developed and the controller parameters are obtained using a model-based design method, such as pole-placement. PID controllers may also be purchased *off-the-shelf* for use in an industrial application. In such cases, the engineer determines the controller parameters empirically. Undoubtedly, the most significant contribution to the area of empirically-based PID design has been made by Ziegler and Nichols (1942). Ziegler and Nichols proposed two different techniques for developing *optimum* controller parameters that remain widely used. Both methods share a common criteria in that they are designed to achieve a 1/4 decay ratio in the response of the output of the controlled process to a step change, but the methods differ in their implementation.

The first method proposed by Ziegler and Nichols defines the process dynamics from the open-loop step response of the system. The open-loop step response is

characterized by two parameters: the maximum slope of the response curve and the process time delay. These two parameters are obtained graphically by drawing a tangent at the point where the slope of the curve is at a maximum as shown in Figure 13. The distance from the intersection of the tangent line to the x and y axes are labeled L and a, respectively. The values L and a are used to calculate the controller parameters as given in Table 3.

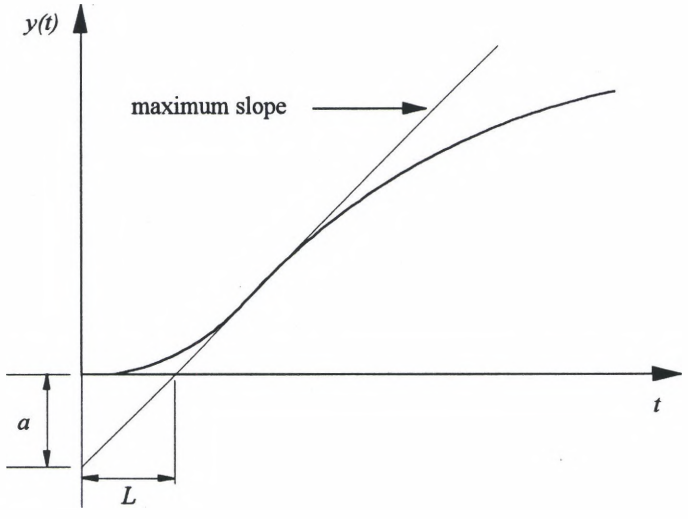

Figure 13. Parameters obtained from Ziegler-Nichols Step Response Method

Table 3. Ziegler-Nichols step response method parameters

Controller	K	T_i	T_d
P	$1/a$		
PI	$.9/a$	$3L$	
PID	$1.2/a$	$2L$	$L/2$

The Ziegler-Nichols frequency response method is based on the knowledge of a single point on the Nyquist frequency response curve, i.e., the critical point. (The critical point is where the Nyquist curve intersects the negative real axis.) This point can be

obtained experimentally by controlling the process with purely proportional feedback and increasing the controller gain until the process output begins to oscillate. At this point, the controller output and the plant output are 180^o out of phase. The gain required to bring the system to the point of oscillation is referred to as the ultimate gain, k_c, and the period of oscillation of the resultant output is called the ultimate period, t_c. Table 4 gives the recommended PID parameters for the Ziegler-Nichols frequency response method.

Table 4. Ziegler-Nichols frequency response method parameters

Controller	K	T_i	T_d
P	$0.5k_c$		
PI	$0.4k_c$	$0.8t_c$	
PID	$0.6k_c$	$0.5t_c$	$0.85t_c$

The Ziegler-Nichols criteria are designed for cases where the primary control objective is disturbance rejection (the regulator case) as opposed to set-point tracking (the servo case). The gain obtained from the two Ziegler-Nichols methods is relatively high in order to meet this objective. It can be shown that a decay ratio of 1/4 equates to a relative damping $\zeta = 0.22$ which causes a rather large overshoot for setpoint changes. Other empirically-based tuning criteria have also been proposed that offer improved performance over Ziegler-Nichols techniques. Miller, Lopez, Smith and Murrill (1967) compare the most significant of the empirically based tuning criteria (including Ziegler-Nichols) and conclude that controller tuning methods that use an integral error criteria are superior to the other techniques.

It is not always possible, or even desirable, to use empirical methods to obtain PID controller parameters. If a model of the plant is available, the parameters can be derived using *pole-placement*. Consider a plant characterized by the second-order model:

$$G_p(s) = \frac{k_p}{(1+sT_1)(1+sT_2)}.$$

If the transfer function of the PID controller is given as:

$$G_c(s) = \frac{K(1+sT_i+s^2T_iT_d)}{sT_i},$$

the characteristic equation of the closed-loop system can be determined to be:

$$s^3 + s^2\left[\frac{1}{T_1}+\frac{1}{T_2}+\frac{k_pKT_d}{T_1T_2}\right]+s\left[\frac{1}{T_1T_2}+\frac{k_pK}{T_1T_2}\right]+\frac{k_pK}{T_iT_1T_2}=0. \qquad (2.24)$$

If the desired closed loop characteristic equation can be described as:

$$(s+\alpha\omega)(s^2+2\zeta\omega s+\omega^2)=0, \qquad (2.25)$$

the controller parameters can be determined by substituting equation (2.25) into equation (2.24) and comparing like powers of s. A more detailed description of the continuous-time pole-placement procedure is given in Astrom and Hagglund (1988b).

The response of many SISO systems can be characterized by a pair of complex poles, commonly referred to as the *dominant poles*. Whereas the Ziegler-Nichols (1942) techniques are based on the knowledge of one point on the Nyquist curve, Astrom and Hagglund (1988b) have developed a procedure for designing a controller based on the knowledge of two points on the Nyquist curve. The *dominant pole design* method estimates the locations of the dominant poles of the closed-loop system from the Nyquist curve of the open-loop system. A complete development of the method is given by Astrom and Hagglund (1988b).

A number of PID design methods are based on the concept of selecting controller parameters so that the dominant poles of the plant are canceled. These methods are simple to implement and yield a system that responds well to setpoint changes; however, Astrom and Hagglund (1988b) contend that the response of these systems to load disturbances is poor, as it includes the dynamics of the canceled poles. The closed-loop system will therefore respond to load disturbances similarly to the response of the open-

loop system. The same effect occurs if the cancellation of the poles is not exact. These methods should therefore be avoided in practice if the controller is going to be used primarily as a regulator.

All of the design methods mentioned so far have been based on continuous-time models of the controller and the plant. The controller parameters obtained must be converted to discrete-time before applying the method. It is also possible, however, to position the closed-loop poles of lower-order discrete-time models directly. A thorough treatise on the subject of discrete-time pole-placement is given by Astrom and Wittenmark (1990).

Some of the more common techniques for designing with PID controllers have been presented, but many other methods have also been developed. As long as the sample rate is relatively fast, the controller design can be done in continuous-time and the parameters obtained can be converted for use in the discrete-time algorithm. The controller design may also be done directly in discrete-time, although it may be more difficult to force the resulting controller into a PID structure.

2.6 PID Implementation Issues

Once the PID control algorithm has been developed and the method for selecting the controller parameters has been determined, several important issues remain to be addressed before the PID algorithm is implemented. A key consideration in any digital control design is the selection of the appropriate sampling rate. Sample rates that are too fast can lead to numerical difficulties, and sample rates that are too slow can result in a poorly controlled process, or even aliasing. In this section, some guidelines are given for selection of a sampling frequency that is appropriate for the application. Quantization error and word length also play an important role in the implementation of digital

controllers, particularly in the selection of the controller hardware. Both hardware quantization and computational quantization are discussed briefly in this section. Word lengths that are too short can result in problems such as integration offset or limit cycling. A method for determining the minimum requirements for memory elements is also presented.

Sample rate selection

Selection of an appropriate sampling interval is critical for the controller to be able to meet the design specifications. The effect of sample rate on frequency response has already been discussed briefly in Section 2.4. The constraining factor in the selection of the sampling rate is found in Shannon's sampling theorem, which states that "a continuous-time signal with a Fourier transform that is zero outside the interval $(-\omega_0, \omega_0)$ is given uniquely by its values in equidistant points if the sampling frequency is higher than $2\omega_0$" (Astrom and Wittenmark, 1990). The frequency $2\omega_0$ is referred to as the Nyquist frequency. Simply stated, a continuous signal can be completely recovered after sampling if the signal is sampled at a rate at least twice the highest frequency component of the signal. If this rule is not adhered to, the original continuous-time signal cannot be recovered. This effect is known as aliasing. Anti-aliasing filters are often used to filter the input signal before sampling to remove the high frequency components. Anti-aliasing filters must be used carefully in control applications, however, as they introduce additional phase lag into the system which, in turn, causes instability. The additional phase lag may have to be factored into the control design if it is significant. This can be done by approximating it as a simple time delay.

It is often assumed that the sampling rate should be as high as possible, particularly if the control design is performed in continuous-time and then converted to discrete-time. Faster sampling leads to a discrete equivalent that more closely approximates the analog model. Clarke (1984) points out, however, that there are cases where high sampling frequencies can lead to difficulties. Problems can arise, for instance, in cases where derivative action is employed and relatively long integral and derivative times are required for a given plant. Middleton and Goodwin (1990) demonstrate how too rapid sampling can also lead to numerical difficulties. Because word lengths are fixed by hardware limitations in digital systems, as the sampling rate is increased, the maximum computational error also increases. Middleton and Goodwin (1990) recommend that the sampling rate be selected to be approximately ten times the closed-loop bandwidth of the system; however, they do state that sampling rates up to fifty times the closed-loop bandwidth are often acceptable when implemented in high-precision hardware. Clarke (1984) states that in most cases, there is little point in selecting the sampling interval T such that there are more that ten to twenty samples during the ninety-five percent rise time of the step response. Astrom and Wittenmark (1990) recommend a sampling rate of ten to thirty times the closed-loop bandwidth of the closed-loop system. If N_r is the number of sampling periods per rise-time:

$$N_r = \frac{T_r}{\mathrm{T}}$$

where T_r is the rise time. Astrom and Wittenmark (1990) recommend choosing N_r to be between four and ten for a first or second-order system.

Quantization and word length

Quantization errors are the result of having to store digital numbers in memory elements with a finite accuracy. Quantization errors can be introduced in several ways. Quantization occurs in hardware devices such as analog-to-digital (A/D) converters. Quantization also occurs when performing numerical computations on parameters or coefficients that result in overflow, underflow and roundoff. The consequences of quantization depend on the structure of the feedback control system that is used. Detailed analysis of quantization leads to a very complicated non-linear model that is difficult to analyze. Some insight can be gained, however, by examining some simple cases using linear analysis.

Quantization of parameters or signals can cause three different effects; bias, noise and limit cycles. Bias is caused primarily through truncation. In two's complement arithmetic, truncating a positive number or a negative number results in a bias in the same direction, i.e., trunc(x) < x for both positive and negative x. If the quantization step is defined to be q, the maximum error resulting from truncation is q, which results in a bias of $q/2$. It can be similarly shown that the maximum error resulting from roundoff is $q/2$, which yields a bias of 0. Rounding is therefore preferred over truncation.

Quantization error can also appear in the form of noise. Hanselmann (1987) presents a model for handling quantization error as noise. Assuming two's-complement arithmetic and a quantization step of q, the mean and variance of the quantization noise can be determined to be:

$$\text{variance:} \quad \sigma^2 = q^2/12$$

$$\text{mean:} \quad \mu = -q/2 \quad \text{for truncation}$$

$$\mu = 0 \quad \text{for rounding}$$

These expressions assume a uniform quantization error distribution in the interval q, which has been shown to be a valid assumption under some conditions. It also assumes a

continuous amplitude input into the quantizer. This is true for A/D converters, but is not valid in considering internal computations. If the quantization is done as truncation, the error is equally distributed over the interval $(0, q)$. If quantization is performed as rounding, the error is equally distributed over the interval $(-q/2, q/2)$. Astrom and Wittenmark (1990) also show how quantization error can be modeled using linear analysis, treating the quantization error as a stochastic input. The linear models serve only as an approximation, however, and do not completely describe all aspects of quantization error. Another technique for analyzing quantization error using *describing function analysis* is given by Astrom and Wittenmark (1990). Describing function analysis can be used to predict limit cycles due to quantization and roundoff.

The necessity of roundoff when dealing with finite precision machines raises the question of what computer word length is required for the application. A problem known as *integration offset* can arise when the length of the storage element for the integral term is too short. The expression for updating the integral term was given as:

$$I(k+1) = \beta I(k) + (1-\beta)u(k)$$

where β was determined for the bilinear transformation to be:

$$\beta = \frac{2T_i - \mathrm{T}}{2T_i + \mathrm{T}}.$$

The correction term, $(1-\beta)u(k)$, is normally much smaller than the integral term $\beta I(k)$. If the word length is too short, the correction term will be rounded off causing an offset in the output. In a processor that utilizes fixed-point fractional arithmetic, the maximum value of $u(k)$ is limited to 1. For example, let $\mathrm{T} = .01$ sec and $T_i = 10$ sec. This results in $\beta = 0.9990005$ and $(1-\beta) = 0.0009995$. In order to obtain less than 5 percent error in the integral term, the minimum number of bits required can be calculated to be:

$$\text{number of bits} = -\frac{\log[(0.0009995)(0.05)]}{\log(2)}$$

$$= 14.28 \approx 15 \text{ bits.}$$

2.7 Chapter Summary

In this chapter, basic PID control theory has been presented. The action of the PID terms was individually explained. It was noted that the classical version of the PID control law presents some difficulties in practical implementation. For instance, the problem of derivative kick stems from the derivative term being driven by the error signal which changes suddenly in response to a setpoint change or a load disturbance. A solution was proposed whereby the controller structure is modified so that the derivative term is only acted upon by the output and not the setpoint. This allows for quick response to load disturbances without generating a large overshoot from a change in setpoint. The problem of integral windup was also addressed by the addition of a limiting function at the output of the integral term. Another difficulty with the classical controller was found in the derivative term. The pure derivative term was shown to have a gain that increased without bound with the frequency, resulting in controller sensitivity to measurement noise. The problem was lessened by incorporating a filter into the derivative term that limited its gain at high frequencies.

After the continuous-time PID controller was developed, the discretization of the algorithm for digital implementation was discussed. The backward rectangular rule and the trapezoidal rule were selected as the primary methods for discretizing a continuous-time control system, and some frequency response considerations of the two methods were analyzed. Since the bilinear transformation caused no phase error in the discretization, it was selected as the means of discretizing the continuous-time controller model. The continuous-time PID model was then discretized and developed for implementation.

Having developed a discrete-time PID algorithm, some of the more popular methods for designing with PID controllers were briefly discussed. The Ziegler-Nichols

step response and frequency response methods were explained. Also, the dominant pole design method of Astrom and Hagglund (1988b) was explained, as well as a continuous-time pole-placement technique. In addition, a discrete-time pole placement method was briefly discussed for performing direct digital design.

Two important implementation issues were then presented. The ramifications of the sampling frequency on the control system were discussed and some guidelines for selecting an appropriate controller sampling frequency were presented. Also, the issues of quantization and word length were explored. Some of the difficulties of signal and computational quantization were explained and several methods were referenced for analyzing quantization errors. In addition, the issue of memory word lengths was discussed.

CHAPTER III

PID CONTROLLER IMPLEMENTATION

3.1 Introduction

A discrete-time version of the PID algorithm was developed in Chapter 2. The derivative-of-output model was selected for implementation and the bilinear transformation was chosen as the method for conversion of the continuous-time model to discrete-time. In this chapter, the PID algorithm is coded in Motorola DSP56000 assembly language and tested on the Motorola ADS56000 development system. An Intel 80386-based Zenith computer running a Microsoft QuickBASIC program is used as a plant model. The Motorola ADS56000 development system acting as the PID controller is physically linked to the Zenith computer via a National Instruments AT-MIO-16 A/D board installed in the computer backplane. A Motorola DSP56ADC16 Evaluation Board serves as the interface for the control output and the feedback input to the DSP56000. The purpose of this chapter is to discuss the implementation of the PID control algorithm on the DSP56000 and to demonstrate the performance of the PID algorithm operating in a real-time environment.

DSP chips offer a number of hardware and software advantages over general-purpose microprocessors. In this chapter, the features of DSP chips that are of particular

importance for control applications are described first. Next, a brief overview of the Motorola DSP56000 is presented, with both architectural issues and significant hardware and software features being discussed (The knowledgeable reader may wish to omit this section). The development and organization of the PID algorithm in DSP56000 assembly code is then explained, including both the supervisory code (such as DSP chip initialization, peripheral port programming and sample rate control) and the actual PID algorithm. In addition, some of the more important features of the National Instruments AT-MIO-16 A/D board and DOS LabDriver software that serve as the plant input and output interface are discussed. The discussion includes an explanation of the real-time plant model and the program written to implement it.

Following the explanation of the hardware and software used to implement the PID controller and the plant, the results of testing the PID controller are presented and analyzed. First, the actions of the proportional, integral and derivative terms are tested and compared to analytical results for the same conditions. The PID controller is then tested operating in a closed-loop with the 80386-based plant. Two sets of controller parameters are selected to test the performance of the controller. The first set of controller parameters is obtained empirically by performing the Ziegler-Nichols (1942) frequency response test on the actual plant. The second parameter set is taken from a simulation by Hagglund and Astrom (1985) of their auto-tuning PID controller based on the dominant pole design method. Finally, the performance of the PID controller from the perspective of processing speed is evaluated.

3.2 Using a DSP Chip for Control

A key issue to consider when designing a microprocessor-based control system is which microprocessor to use in the implementation. Several factors must be considered before the microprocessor can be selected. Ahmed (1991) categorizes these factors as:

- Architecture
- Performance
- Peripheral Integration.

Probably the most important of the three categories is *architecture*. The processor architecture not only has a direct effect on the resolution and bandwidth of the control system, but it also plays a vital role in system performance. Architecture affects signal and coefficient quantization levels, as well as numerical factors such as truncation, roundoff and overflow. For instance, insufficient register and memory element word lengths can cause excessive quantization noise, limit cycling and integration offset. Truncation can cause bias in the system output, and register overflow can cause positive numbers to become negative and vice-versa with potentially catastrophic results. Architecture also plays a vital role in minimization of computational overhead, which greatly affects performance. The traditional Von-Neuman architecture used in general-purpose microprocessors creates a bottleneck where instructions and data share the same data bus. Also, in most general-purpose microprocessors, multiplication is accomplished through repeated addition. Only recently have hardware multipliers become available on the central processor chip, and they are not usually an integral part of the arithmetic-logic unit.

The second criteria to consider when evaluating microprocessors for control applications is *performance*. It has been stated that the sampling rate of the controller should normally be between 10 and 20 times the bandwidth of the system. The maximum

sampling rate of the control system is dependent on the speed at which the processor can execute instructions. This could be a non-issue in cases where sampling rate is on the order of seconds, or even minutes. In many control applications, however, very high sampling frequencies are required. Performance can become even more critical when a more sophisticated control strategy is necessary, such as adaptive control, where orders of magnitude more instructions must be executed in each sampling interval. Performance is also important in the consideration of computational delay. If the control system is designed so that there is no direct feedthrough of the controller input to the controller output, there will be a delay from the time the plant output is measured to when the new control output is calculated. This time period is referred to as *computational delay*. Computational delay will introduce additional phase lag into the system and can degrade system performance if it is significant. Processor throughput, therefore, will directly impact computational delay.

The third factor to consider when selecting a microprocessor is *peripheral integration*. If external hardware can be minimized by on-chip peripherals, both cost and space requirements can be reduced. If the control system is to be used in a mobile application, such as in an aerospace or automotive application, board real estate can become a critical issue. In a control application, the microprocessor will need to interface with A/D converters to monitor the feedback signals from the plant outputs and possibly monitor one or more reference inputs. The interfacing to the controller output signals is normally handled with D/A converters. The microprocessor may also be required to interface with a host computer for monitoring purposes. The number and type of on-chip peripherals available on the processor will determine how much external hardware will be required to perform the above functions, and thus have a direct bearing on implementation cost and space requirements.

The field of digital signal processing has stimulated the development of dedicated VLSI chips specifically designed to handle the demands of processing digital signals. These dedicated microprocessors, known as digital signal processing, or DSP chips, offer architectural and performance advantages over general-purpose microprocessors. They also address many of the difficulties encountered when using general-purpose microprocessors. In addition, they frequently incorporate on-chip peripherals that can reduce implementation cost and minimize board space requirements. The Motorola DSP56000 offers many features that make it suitable for control applications, and it has therefore been selected for implementation of the PID controller in this project. Although Texas Instruments offers competitive hardware, the Motorola chip has been selected because a DSP56000 development system is available for use at the University of Dayton. Some of the key features of the DSP56000 that are significant for control applications are examined in the following section.

3.3 An Overview of the Motorola DSP56000

The DSP56000 architecture

The Motorola DSP56000 has been specifically designed to maximize processor throughput for signal processing applications. The DSP56000 is referred to by Motorola as "dual-natured" (Motorola, 1990). This refers to the fact that the DSP56000 has two independent memory spaces, two address generation units (AGUs), and a data arithmetic logic unit (ALU) having two accumulators and two shifter-limiter circuits. The dual-natured architecture makes the DSP56000 ideally suited for digital signal processing that requires many successive multiply and add operations. A block diagram showing the dual bus structure of the DSP56000 is given in Figure 14. The two independent memory

spaces are denoted as *x-memory* and *y-memory*. Each memory space has its own address generation unit and its own data bus. The program memory functions independently of the data memory and also has its own address generation unit and data bus. This structure allows the next instruction to be fetched while the current instruction is executing, thus minimizing the number of clock cycles per instruction cycle. It also permits parallel data moves in a single instruction cycle.

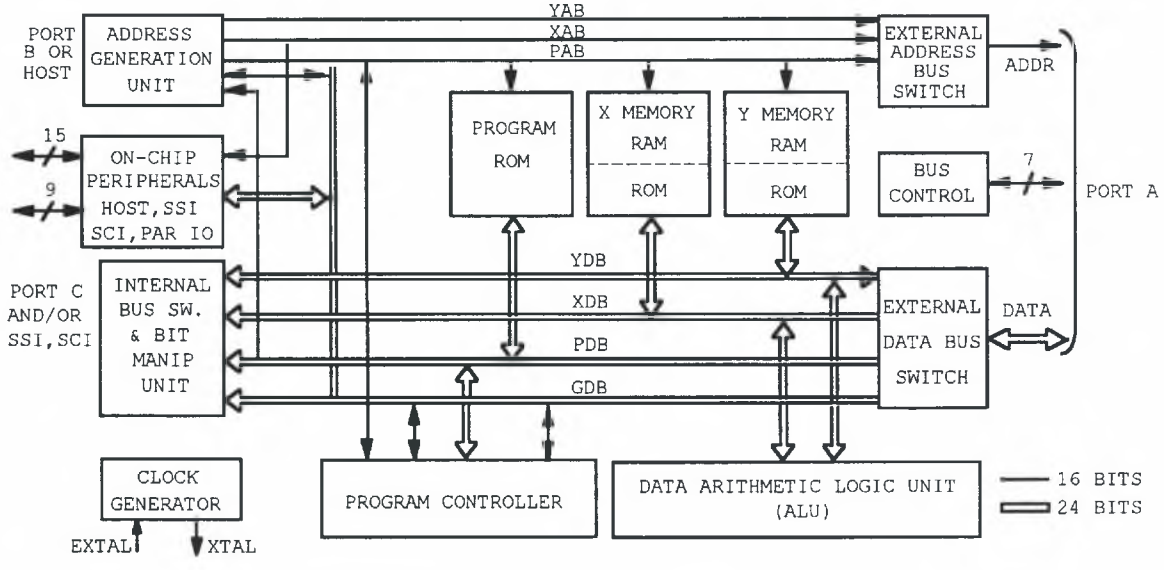

Figure 14. Block diagram of Motorola DSP56000

All arithmetic and logical operations in the DSP56000 are performed in the ALU. A block diagram of the ALU is shown in Figure 15. The ALU can perform any of the following operations in a single instruction cycle:

- 24-bit by 24-bit multiplication
- Multiply-accumulate with positive or negative accumulation
- Convergent rounding
- Multiply-accumulate with positive or negative accumulation and convergent rounding

- 56-bit addition
- 56-bit subtraction
- A divide iteration
- A normalization iteration
- Shifting
- Logical operations

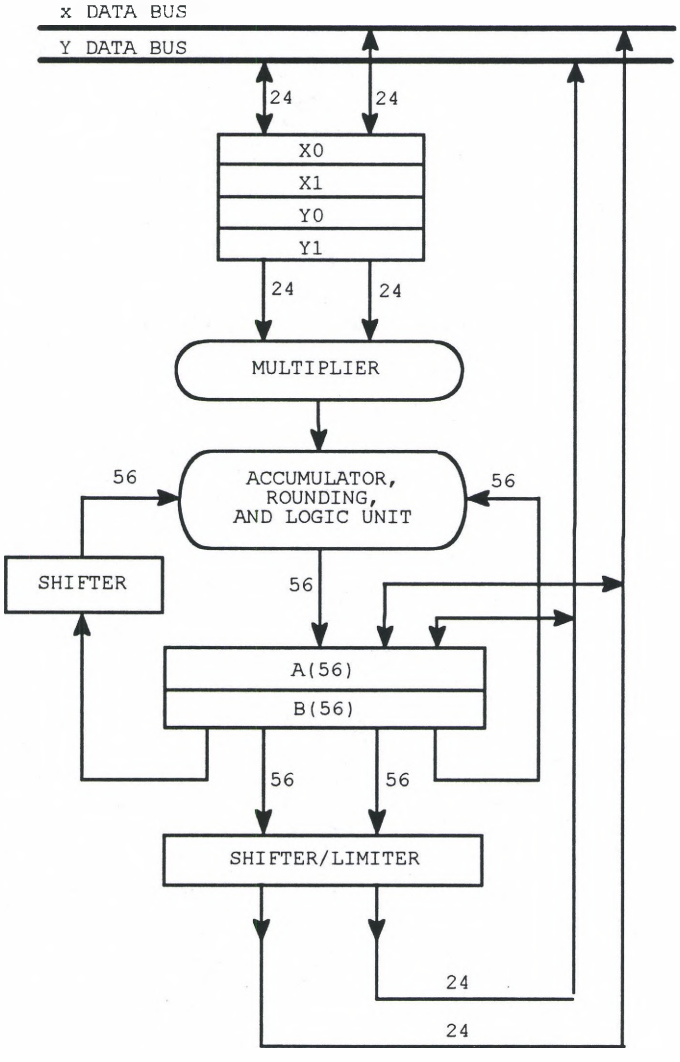

Figure 15. Block diagram of DSP56000 Data ALU

The structure of the ALU shown in Figure 15 is referred to as a *Harvard architecture*. The dual accumulators combined with independent data buses allow for parallel data moves. Data for the next operation can be loaded into the input registers in the same instruction cycle that the current operation is executing. The multiply-accumulator/logic (MAC) unit within the ALU performs rounding of the accumulators if requested in the instruction. The rounding method used is called *convergent rounding*, which rounds down if the number is odd and rounds up if the number is even, eliminating any possibility of introducing a bias.

Many of the difficulties normally encountered when using general-purpose microprocessors for control applications have been amply addressed in the architecture of the DSP56000. In addition to the features already mentioned, the 24-bit wide registers in the ALU result in quantization levels that allow for fast sampling rates without introducing limit cycling or integration offset. The 48-bit accumulators combined with the 8-bit extension registers allow for 24 by 24 bit multiplication without loss of precision, reducing the possibility of cumulative errors. Although the structure of the DSP56000 is specifically designed for digital signal processing, the architecture is well suited for control applications as well.

Arithmetic considerations

An important issue that must be addressed when specifying a microprocessor is the formatting of data. The two most common data formats are fixed-point and floating-point. Floating-point representation offers a large dynamic range compared to the rather limited dynamic range of fixed point representations. Floating-point numbers also reduce the risk of overflow, underflow and truncation errors. There is a significant hardware cost

disadvantage to process floating-point numbers, however, and for a given number of bits, floating-point numbers are less accurate than fixed-point representations. Floating-point processors are also typically slower, require more memory and consume more power than their fixed-point counterparts.

The DSP56000 uses a fixed-point fractional two's complement representation of data. As in the case of most fixed-point formats, two's complement numbers are used primarily because they require very simple hardware for addition and subtraction. Unlike most general-purpose microprocessors, however, the DSP56000 uses fractional numbers rather than integers. Fractional numbers are preferred in digital signal processing because multiplication of two fractions always yields a fraction, thus eliminating the possibility of an overflow condition when performing a large number of successive multiplications. The least significant bits of the product are simply truncated (or rounded) and the resulting number is an approximation of the actual product to within the accuracy of the number of storage bits available. Fractional representation thus trades precision for control of number growth. Integer representations, on the other hand, are always accurate, but at the increased risk of a multiplicative overflow.

In order to eliminate the potential for an overflow caused by storing a 56-bit accumulator in a 24-bit memory location, the DSP56000 has the ability to perform *saturation arithmetic.* The accumulators in the DSP56000 are equipped with 8-bit extension registers that allow numbers up to 255.9999998 to be represented. In a general-purpose microprocessor, if a number greater than $7FFFFF is transferred to a 24-bit register or memory location, an overflow occurs and the number in the 24-bit register is interpreted as being negative. Using saturation arithmetic, the DSP56000 automatically substitutes the maximum positive (or negative) number (e.g., $7FFFFF) for the number in the accumulator when data is stored to a 24-bit register or memory location, thus minimizing the error.

The DSP56000 instruction set

The Motorola DSP56000 has an instruction set containing 62 instructions. In this section, a few of the more significant features of the instruction set will be highlighted. The programming model for the DSP56000 is shown in Figure 16. The DSP56000 can be viewed as consisting of three functional units: the data arithmetic logic unit (ALU), the address generation unit (AGU), and the program controller. These three units essentially operate in parallel. The instruction set is designed to keep each of the units busy each instruction cycle in order to maximize processor performance. Because of the parallelism designed into the DSP architecture, up to three data transfers can be specified in a single instruction; one on the X data bus (XDB), one on the Y data bus (YDB) and one within the data ALU itself. Of the 62 instructions in the set, 30 allow for parallel data moves.

Another powerful feature of the DSP56000 is the number of addressing modes available to the programmer. The addressing modes are divided into three main categories: *register direct*, *register indirect*, and *special*. Within these three categories are 18 specific addressing modes. The indirect addressing modes that utilize the address index registers are of particular importance As the name implies, the *offset registers* allow the address registers to be offset. The modifier registers specify whether the offset is to be applied as a straight linear addition, or whether a special arithmetic offset is to be applied, such as modulo arithmetic. In addition, the address registers can be automatically incremented or decremented before or after the instruction is executed.

The DSP56000 also offers several other instructions that minimize the amount of code required and maximize the speed of operation. A hardware "DO LOOP" is available. Multiply and multiply-and-add instructions can be selected with or without rounding. X-memory and y-memory can be treated as *long words* if double precision (48-bits) is

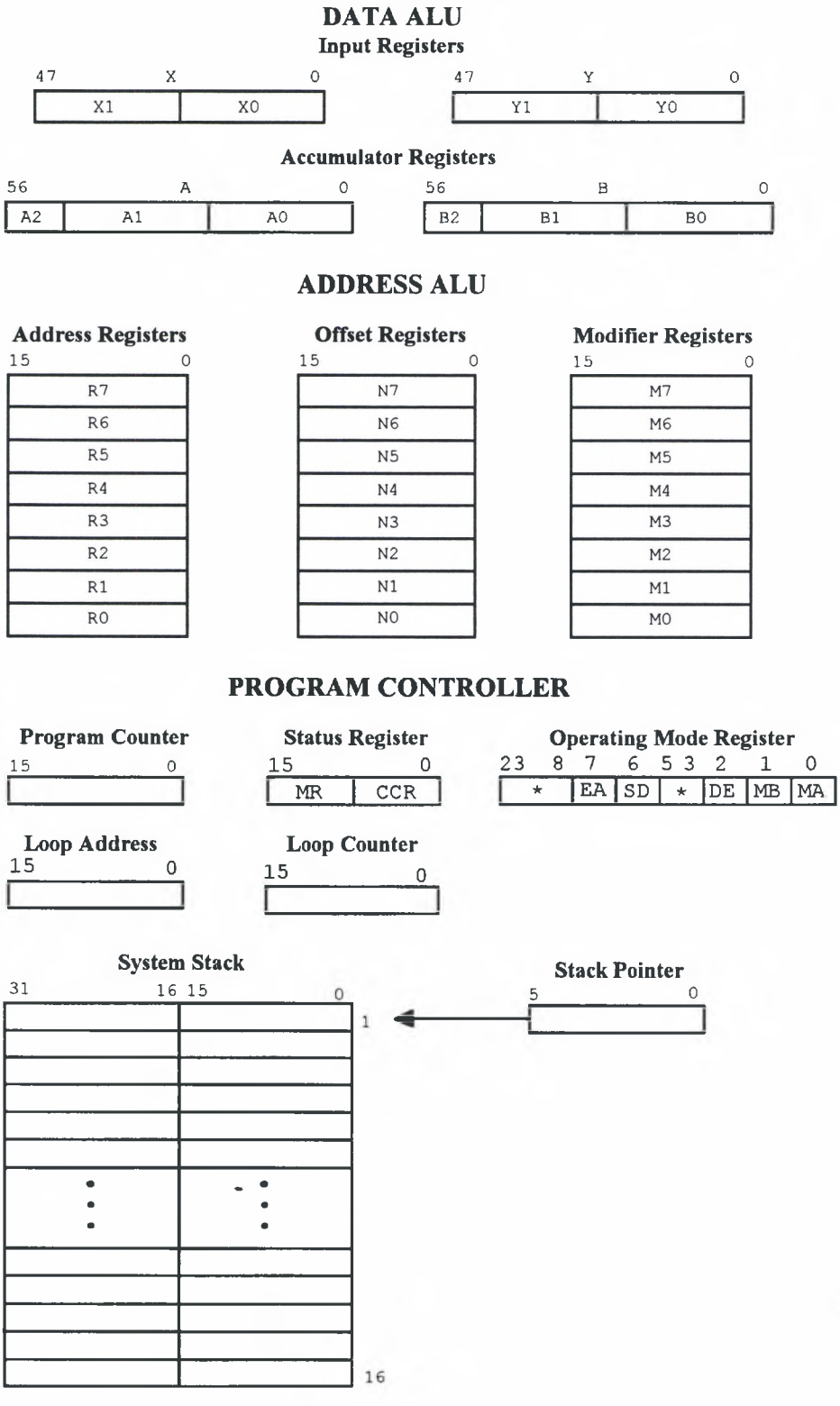

Figure 16. Programming model for Motorola DSP56000

required. A one-bit hardware divide instruction can also be used to develop a full division subroutine. In summary, the DSP56000 instruction set provides the programmer with a powerful set of tools for developing fast, efficient code for control applications.

On-chip peripherals

The DSP56000 contains three *ports* that provide the hardware link to off-chip devices. Port A is the memory expansion port that can be used for memory expansion or for memory-mapped I/O. Port B is a dual-purpose I/O port that can serve one of two different functions. It can be used as a general-purpose parallel I/O port, with 15 pins that can be individually configured as inputs or outputs, or it can serve as an 8-bit bi-directional host interface (HI). Port C is a 9 pin I/O port that can be configured in one of three ways. With the first option, the 9 pins can be set up as parallel I/O that can be configured as inputs or outputs. The second option is to configure three pins as the *serial communications interface* or SCI. When set up in this manner, the other six pins can be configured as general-purpose parallel I/O. If the third option is selected, Port C can be configured as the *synchronous serial interface* or SSI. A block diagram of the configuration and pin-outs of the three ports is shown in Figure 17.

The SCI: The SCI provides full-duplex serial communication to other DSPs, microprocessors, or peripheral devices. Communication can be selected to be synchronous or asynchronous. The SCI uses three pins denoted as *transmit data* (TXD), *receive data* (RXD) and the SCI serial clock (SCLK). The SCI uses industry standard baud rates and protocols. The SCI consists of separate transmit and receive sections which operate independently. It also contains an internal programmable baud-rate

generator which can double as a general-purpose timer when not being used by the SCI. A block diagram of the SCI internal baud rate generator is shown in Figure 18.

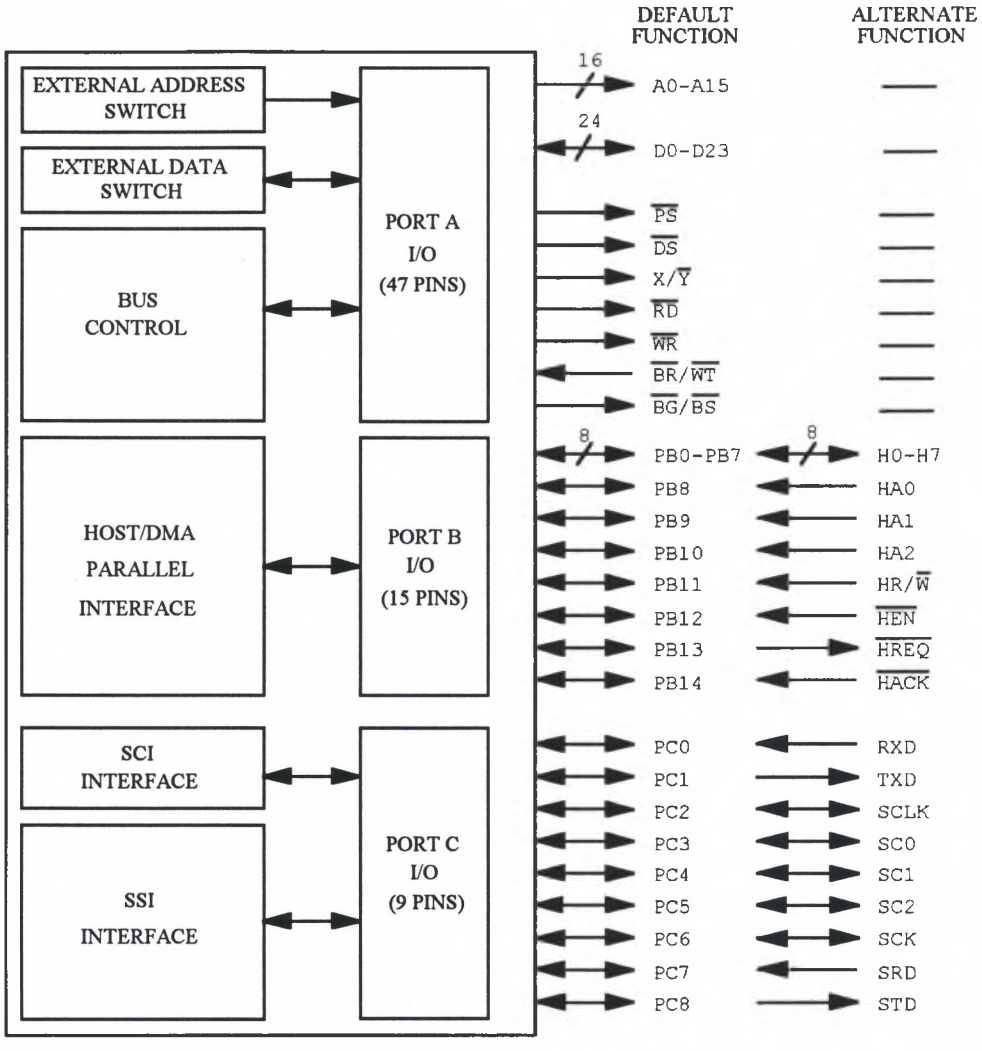

Figure 17. Motorola DSP56000 Peripheral Ports

By setting the clock divider bits (CD0-CD12), the clock prescaler bit (SCP), and the clock out divider (COD), the desired baud rate can be selected knowing the oscillator clock frequency, f_o , which is normally 20.5 megahertz. In addition to controlling baud rates for transmission and reception of data, the SCI clock can also be used as a timer to

generate periodic interrupts of the DSP. (This will prove to be useful for controlling the sampling rate of the PID controller.)

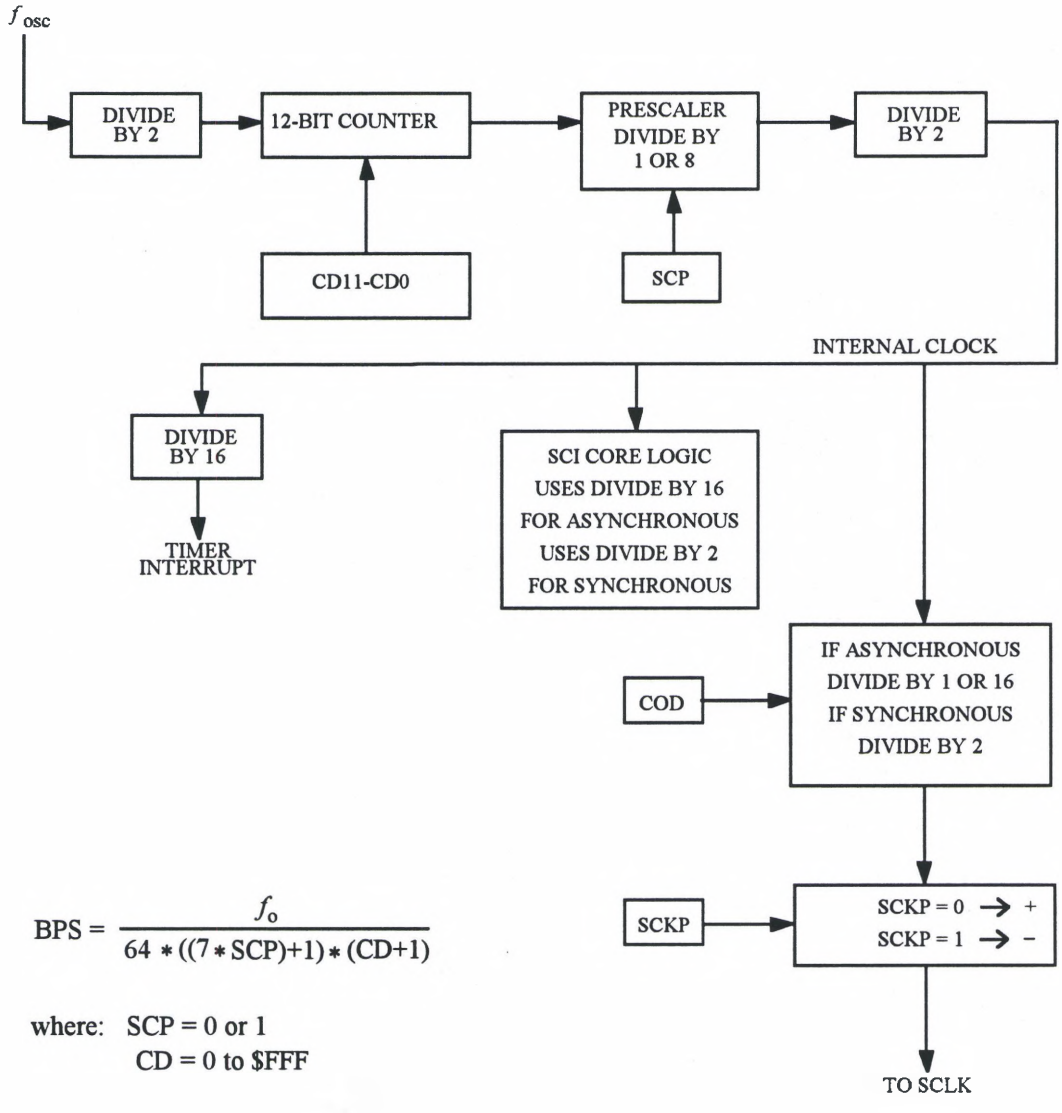

Figure 18. Block Diagram of SCI baud rate generator

The SSI: The SSI provides a full-duplex serial port that can communicate with a number of different devices, including codecs, other DSPs, microprocessors and peripheral devices. In the SSI, all serial transfers of data are synchronized to a clock, with one word

being transferred per period in *normal mode* or up to 32 words per period in *network mode*. It can also function in an *on-demand mode*.

The on-chip peripherals provide the DSP56000 with the flexibility to interface with external peripheral devices, other DSPs, or a large number of other intelligent devices. The brief explanation given provides only an overview of the operation of the peripherals. Detailed explanations are available in the DSP56000/DSP56001 Digital Signal Processor User's Manual (Motorola, 1991). In this project, the SCI timer is used to control the sampling frequency of the PID controller. The SSI provides the interface from the DSP to the A/D converter that serves as the PID controller's feedback input. The SSI also serves as the interface to the D/A converter that is used as the control output of the PID controller.

The DSP56000ADS Application Development System

In order to develop, debug and evaluate microprocessor-based applications, it is essential to have a development system that provides the designer with a window to the inner workings of the microprocessor. A development system should also provide the designer with a means of interfacing to the microprocessor so that the design can be tested in a real-time environment. Motorola has provided the DSP56000ADS for developing DSP56000 applications. The DSP56000ADS consists of three major components. The first component of the system is the Application Development Module (ADM). The ADM is a stand-alone circuit board containing a DSP56001 processor and related control circuitry. The second component of the DSP56000ADS is a HOST-BUS interface board. The HOST-BUS interface board physically resides in the host computer and provides the interface between the host computer and the ADM. The third element of the DSP56000ADS is the software program that serves as the operator interface to the ADM.

The DSP56000ADS has been used to develop the PID controller for this project. For a more detailed description of the function of the DSP56000ADS, refer to the DSP56000ADS Application Development System User's Manual (Motorola, 1989).

The DSPADC16 evaluation board

The DSP56000ADS allows the DSP-based PID controller to be operated in a real-time environment. The ADM board provides the DSP56001 processor and the necessary support circuitry, but separate hardware is required for the A/D and D/A converters required for the controller's input-output interface. The Motorola DSPADC16 Evaluation Board (EVB) is used for this purpose.

The DSPADC16 EVB is an A/D and D/A conversion system that can be used either in a stand-alone mode or in conjunction with the DSP56000ADS. The EVB utilizes the Motorola DSP56ADC16 16-bit 100 KHz sigma-delta A/D converter and the Motorola PCM-56 D/A converter. The DSP56ADC16, which is manufactured with an on-chip serial interface, can be directly linked to the SSI receiver port of the DSP56001 located on the ADM board. The PCM-56 is also equipped with a serial interface and ties directly to the SSI transmitter port of the DSP56001 on the ADM. A block diagram of the EVB is shown in Figure 19.

The DSP56ADC16 is a sigma-delta A/D converter that utilizes oversampling of the analog input signal. In principal, a series of coarsely quantified (1-bit) data are obtained by oversampling of the input. A digital-domain decimation process is then used to compute a more precise estimate of the analog signal at a lower sampling rate. In the case of the Motorola EVB, the *input* sampling frequency (i.e., the 1-bit sampling done at the input of the converter) is 2.8244 MHz, which is one-half the frequency of the on-board

oscillator. The *output* sampling rate, i.e., the rate at which reconstructed digital data is available from the A/D converter, is 48 KHz. The Nyquist frequency for the A/D converter is therefore 24 KHz. (In reality, the practical bandwidth of the EVB is about 22 KHz due to other hardware constraints.) A more detailed explanation of sigma-delta A/D conversion and a list of references is given by Park (1990).

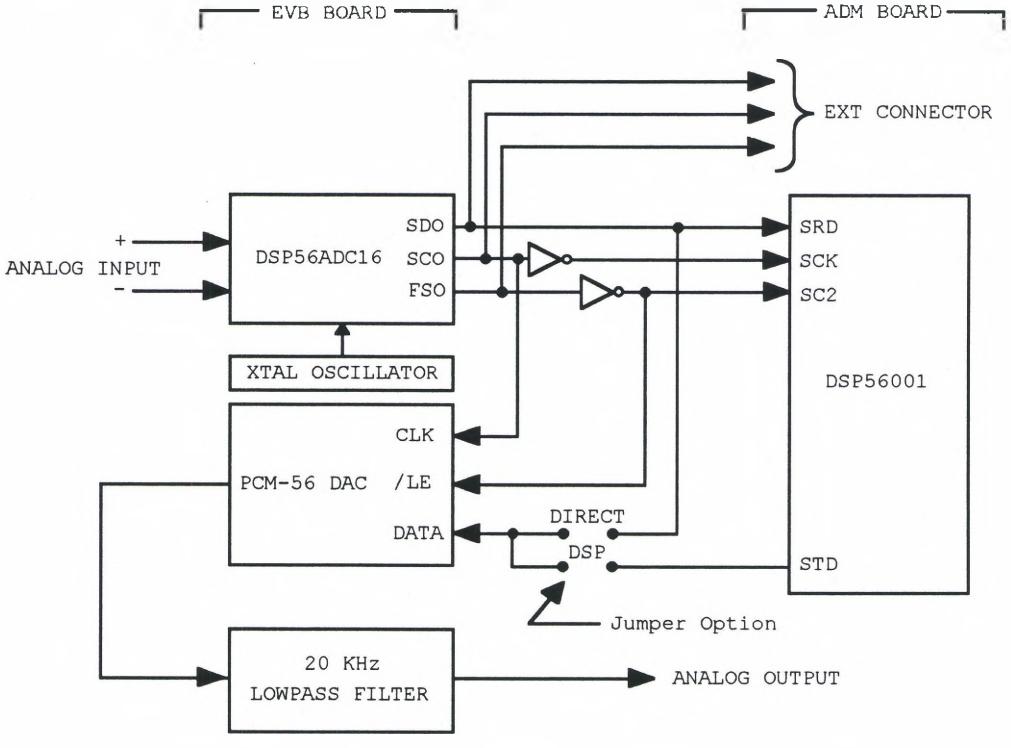

Figure 19. Block diagram of DSP56ADC16 Evaluation Board

The EVB can be configured for either a fully differential input or a single-ended input. The EVB is configured for differential operation for this application because an AC coupling circuit that blocks d.c. levels is activated when the board is configured for single-ended operation. The maximum peak-to-peak differential input signal to the board is approximately four volts.

3.4 The PID Controller Software

The PID algorithm developed in Chapter 2 has been implemented in DSP56000 assembly language. The DSP assembly language program is named "PID64B.asm" and is listed in the appendix. A simplified flowchart of the program is given in Figure 20. A brief explanation of the program operation follows.

DSP initialization

In the first section of the program, the DSP registers mapped to x-memory are assigned variable names and the program variables are assigned to addresses in either x-memory or y-memory. The Interrupt Priority Register (IPR), the Bus Control Register (BCR), the SCI and the SSI are initialized and the historical variables are cleared.

Next, the SSI is set up for operation with the EVB. The Port C control register (PCC) is initialized to set up Port C to function as the SSI. The SSI is set to operate in *normal mode* with an external continuous synchronous clock. It is also set for a data word length of 16-bits and the frame synch is set to be one word long.

The program uses the SCI timer to control the sample rate of the controller by interrupting the DSP at periodic intervals. When an interrupt is generated by the SCI timer, the program flow is redirected via an interrupt vector to an interrupt service routine (ISR). The ISR then becomes the main code for the PID algorithm. Once the algorithm has been completed for one sampling interval, the ISR is exited and the program loops until the next timer interrupt occurs. As shown in Figure 18, bits CD11-CD0 and bit SCP in the SCCR are used to determine the time base according to the formula:

$$\text{Interrupts/sec} = \frac{f_{osc}}{64\left(7(\text{SCP})+1\right)(\text{CD}+1)}$$

where f_{osc} is 20,500,000 in this case.

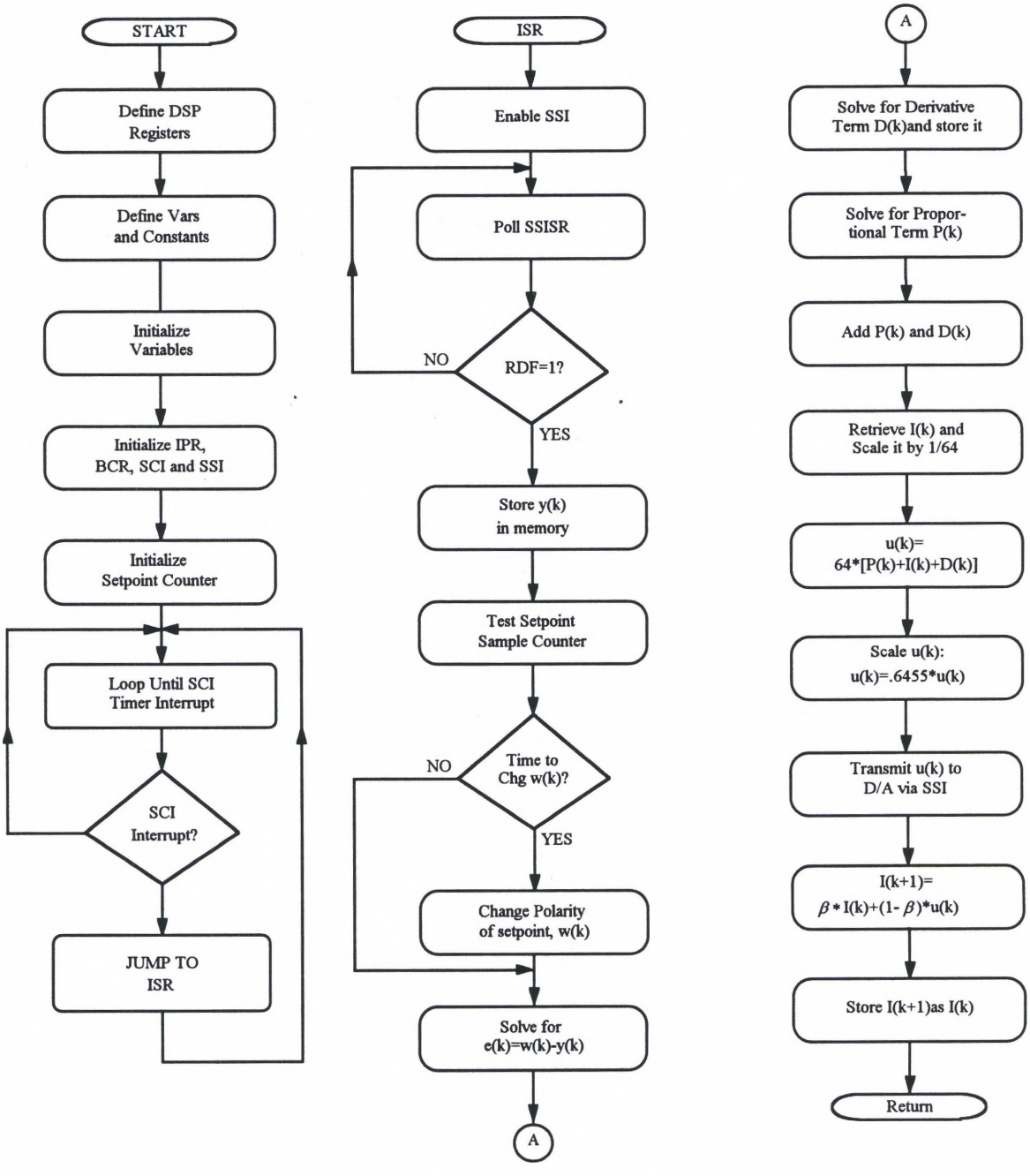

Figure 20. Flowchart of DSP56000 PID controller program

Although the operating conditions of the SSI have been initiated in the DSP, the SSI is not enabled to transmit and receive data until the SCI timer interrupt routine has been initiated. Once powered up, the EVB continuously samples the input of the A/D

converter and transmits data across the SSI. If the DSP does not read the data in the *receive data register* (RX), the next time the receive data shift register is filled, it will not be able to transfer the new data to the RX register. The SSI will then generate a *receiver overrun error* (ROE). In order to keep this from happening, the DSP would have to be reading the SSI *receive data register* at a frequency of 48 KHz. The SSI is therefore not enabled until the SCI timer interrupt service routine has been initiated. It is disabled again before control is returned to the main program.

The reference input

For this project, the reference input $w(k)$ is assumed to be a fixed setpoint as opposed to a dynamic reference signal to be tracked. The setpoint value is stored in an x-memory location. In order to test the system's response to setpoint changes, the reference input $w(k)$ is switched between a positive and a negative value at selected intervals. An address register R0 serves as a sample counter to control the switching of the setpoint.

Coefficient scaling

As discussed previously, the DSP56000 utilizes two's complement, fixed-point fractional arithmetic. It has been noted that the 24-bit words in the DSP have a numeric range of $800000 to $7FFFFF hexadecimal, or -1 to .99999988 decimal. It is therefore necessary that all PID controller coefficients be scaled to fit within that range. The discrete-time control law was given previously in equation (2.31) and is repeated here as:

$$\begin{aligned} u(k) &= P(k) + I(k) + D(k) \\ u(k) &= K' e(k) + I(k) + d_0 D(k-1) + d_1 [y(k-1) - y(k)] \end{aligned} \tag{3.1}$$

The integral term was given previously in equation (2.32) and is repeated here as:

$$I(k+1) = \beta I(k) + (1-\beta) u(k). \tag{3.2}$$

The discrete-time coefficients of equations (3.1) and (3.2) are determined from the coefficients of the classical continuous-time PID algorithm to be:

$$K' = K\left(1 + \mathrm{T}/2T_i\right),\ \beta = \frac{2T_i - \mathrm{T}}{2T_i + \mathrm{T}},\ d_0 = \frac{\left(\frac{2T_d}{N\mathrm{T}} - 1\right)}{\left(\frac{2T_d}{N\mathrm{T}} + 1\right)},\ \text{and}\ d_1 = \frac{2K'T_d/\mathrm{T}}{\left(\frac{2T_d}{N\mathrm{T}} + 1\right)}. \tag{3.3}$$

The coefficients that must be considered for scaling are therefore:

$$K', \beta, d_0 \text{ and } d_1.$$

In order to determine an appropriate scaling factor for K' , a limit must be set on K'. An arbitrary limit of 64 is selected for K' in order to be able to achieve a good response in relatively slow plants. The parameter K' had to therefore be prescaled by $1/64$ to insure that its scaled value would not exceed one. From equation (3.3), since T_i and T are always assumed to be positive, the variable β will always be less than one and thus does not have to be scaled. Likewise, the quantity (1-β) will also always be less than one.

In solving for the derivative term, d_1 is multiplied by K'. From equation (3.3), d_0 is given as:

$$d_0 = \frac{\left(\frac{2T_d}{N\mathrm{T}} - 1\right)}{\left(\frac{2T_d}{N\mathrm{T}} + 1\right)}.$$

The term d_0 will always be less than one and does not require scaling. From (3.3), the term d_1 is given as:

$$d_1 = \frac{2K'T_d/\mathrm{T}}{\left(\frac{2\mathrm{T_d}}{N\mathrm{T}} + 1\right)} = \frac{2K'T_dN}{\left(N\mathrm{T} + 2T_d\right)}.$$

Assuming T<<N, the term

$$\frac{2K'T_dN}{\left(N\mathrm{T} + 2T_d\right)}$$

is limited by $K'N$. K' has already been limited to a value of 64, so if an arbitrary limit of 16 is imposed on N, the maximum value of d_1 becomes (64)(16)=1024. The term d_1 must therefore be prescaled by $1/1024$.

In summary, the coefficients of the PID controller that must be prescaled and their respective scaling factors are given in equation (3.4).

$$K'_{\text{scaled}} = 1/64\, K' \qquad d_{1\,\text{scaled}} = 1/1024\, d_1 \tag{3.4}$$

By limiting K' to a maximum of 64 and N to a maximum of 16 and applying the scaling factors shown in equation (3.4), the coefficients of the PID algorithm will be assured of staying within the limits of the two's complement fractional representation of the DSP56000.

Calculation of the control output

The calculation of the control output *u(k)* is detailed in the following paragraphs (refer to the flowchart in Figure 20). First, the error term is calculated as $e(k)=w(k)-y(k)$ and $y(k)$ is stored as $y(k-1)$ to be used in the next sampling interval. Next, the solution for the derivative term is calculated. (It proved to be prudent to solve for the derivative term first in order to minimize program overhead.) From equation (3.1), the derivative term is determined to be:

$$D(k) = d_0 D(k-1) + d_1[y(k-1) - y(k)].$$

From the previous section, it was determined that d_1 is prescaled by $1/1024$. Therefore, before adding the two terms comprising $D(k)$, the term $d_0 D(k-1)$ must also be divided by 1024. However, since the derivative term will be added to the proportional term which is already scaled by $1/64$, it is convenient to divide $d_0 D(k-1)$ by $1/64$ and to multiply the term $d_1[y(k-1) - y(k)]$ by 16 before adding them together. This results in $D(k)$ being

scaled by $1/64$, making it the same scale as the proportional term, $P(k)$. The term $D(k)$ is stored using limiting, or saturation arithmetic. This is to eliminate the possibility of overflow when storing the 56-bit accumulator in a 24-bit memory location.

Next, the proportional term $P(k)$ is calculated as:

$$P(k) = K'e(k)$$

where K' has been prescaled by $1/64$. $P(k)$ and $D(k)$ are then added together. The integral term that was computed in the previous sampling interval is then scaled by $1/64$ and added to the sum of $P(k)$ and $D(k)$. The sum of $P(k)$, $I(k)$ and $D(k)$ is then multiplied by 64, completing the calculation of $u(k)$.

Once the completed control output, $u(k)$, is calculated, it is multiplied by a hardware scaling factor. The control output is then moved to the SSI *transmit/receive data register.* The SSI *status register* is polled until the *transmit data enabled* bit (TDE) is set to one, indicating that the data has been transferred to the transmit data shift register. The data remains in the transmit data shift register awaiting the next SSI frame synch to be transmitted to the D/A converter. After the control output has been sent to the D/A converter, the next value of the integral term is precalculated for the next sampling interval in order to minimize the computational delay. The expression for the integral term is given in equation (3.2) as:

$$I(k+1) = \beta I(k) + (1-\beta)u(k).$$

$I(k+1)$ is then stored (with limiting) at full scale as $I(k)$ to be used in the calculation of $u(k)$ during the next sampling interval.

3.5 The Real-time Plant Model

The plant hardware

A QuickBASIC program running on an Intel 80386-based 33 MHz Zenith computer is used to emulate a plant in order to test the PID controller in a real-time environment, A National Instruments AT-MIO-16 analog I/O board installed in the computer backplane serves as the input-output interface for the plant. The AT-MIO-16 is equipped with a 12-bit, 25 μsec A/D converter that can be multiplexed as 16 single-ended A/D channels or 8 differential channels that can sample at selected frequencies up to 91 KHz. The A/D channels can be configured for several input ranges with programmable gains. The AT-MIO-16 also has two 12-bit D/A converters, three 16-bit counter/timers and eight digital I/O lines available. For this project, the AT-MIO-16 has been configured as follows:

- Differential, bipolar analog input
- $\pm$10 volt analog input range
- Internal $\pm$10 analog output voltage reference
- Two's-complement mode for analog output.

The AT-MIO-16 is capable of generating internal clock frequencies up to 1 MHz (1 μsec resolution). Experiments indicated that sampling frequencies up to 10 KHz are achievable before the software (i.e., the *computational delay*) becomes the constraining factor. Numerical difficulties are encountered, however, for sample intervals less than 1 msec as the coefficients for some relatively slow plant models become extremely small. An internal clock frequency of 1 KHz has therefore been selected to produce a timebase of

1 millisecond for the sample counter, thus allowing sampling intervals to be selected in 1 millisecond increments.

The plant model

For this project, an adaptive control scheme has been selected that is based on a parametric system identification algorithm. The algorithm attempts to identify parameters in a polynomial representation of the plant by minimizing a least-squares error cost function. A polynomial representation is therefore selected for the plant model to facilitate later testing of the controller parameter estimator.

Many different parametric models have been developed to represent dynamic systems. A commonly used polynomial model is the *autoregressive with exogenous input* (or ARX) model , given as:

$$A(q)y(k) = B(q)u(k-nk) + e(k) \tag{3.5}$$

where $e(k)$ is assumed to be white noise sequence. Equation (3.5) can also be expressed as:

$$y(k) = G(q)u(k) + H(q)e(k)$$

where: $$G(q) = q^{-nk}\frac{B(q)}{A(q)} \text{ and } H(q) = \frac{1}{A(q)}$$

$$A(q) = 1 + a_1 q^{-1} + \cdots a_{na} q^{-na}$$

and: $$B(q) = 1 + b_1 q^{-1} + \cdots \mathrm{b}_{nb} q^{-nb}.$$

The variables *na* and *nb* are the orders of polynomials $A(q)$ and $B(q)$, respectively, and *nk* is the number of unit delays from the input $u(k)$ to the output $y(k)$. Also, q^{-1} represents the backward shift operator.

From the perspective of the plant, no noise is to be injected into the model; therefore, a deterministic version of the ARX model is selected to be used as the plant. The plant model can thus be expressed as:

$$A(q)y(k) = q^{-nk}B(q)u(k). \tag{3.6}$$

It was also assumed that $nk \geq 1$, resulting in the model:

$$G(q) = \frac{B(q)}{A(q)} = \frac{b_1q^{-1} + b_2q^{-2} + \cdots + b_{nb}q^{-nb}}{1 + a_1q^{-1} + a_2q^{-2} + \cdots + a_{na}q^{-na}}. \tag{3.7}$$

A practical limitation must be imposed on the orders of $A(q)$ and $B(q)$. According to Astrom and Haaglund (1988b), "PID control is sufficient for processes where the dominant dynamics are of second order". Isermann (1982) demonstrates that for a plant to be identifiable in a closed-loop, if the controller is second-order (e.g., PID), then the plant cannot exceed fourth-order (This subject will be examined more thoroughly in Chapter 4). Therefore, *na* and *nb* are both limited to four in the program emulating the plant model given in equation (3.7).

The plant software

A set of software drivers from National Instruments called DOS LabDriver provides the link between the plant program and the AT-MIO-16 board. The DOS LabDriver function library is linked to the compiled QuickBASIC program, allowing function calls to be made to the AT-MIO-16. A simplified flowchart of the plant program is shown in Figure 21. Several features are incorporated into the plant program to facilitate testing of the PID controller. Coefficients for five different plants can be stored within the program. A constant load disturbance can be added to the plant output at a selected sample number to test the disturbance rejection capability of the controller. Also, plant input and output data can be stored to a computer file for later analysis. Data

storage can start and terminate at any sample number. In order to minimize the computational delay, the value of the plant output is calculated in advance for the next sample interval. An on-board timer/counter is used to control the sampling frequency of the plant. After the next $y(k)$ is calculated, the counter is polled until the counter value

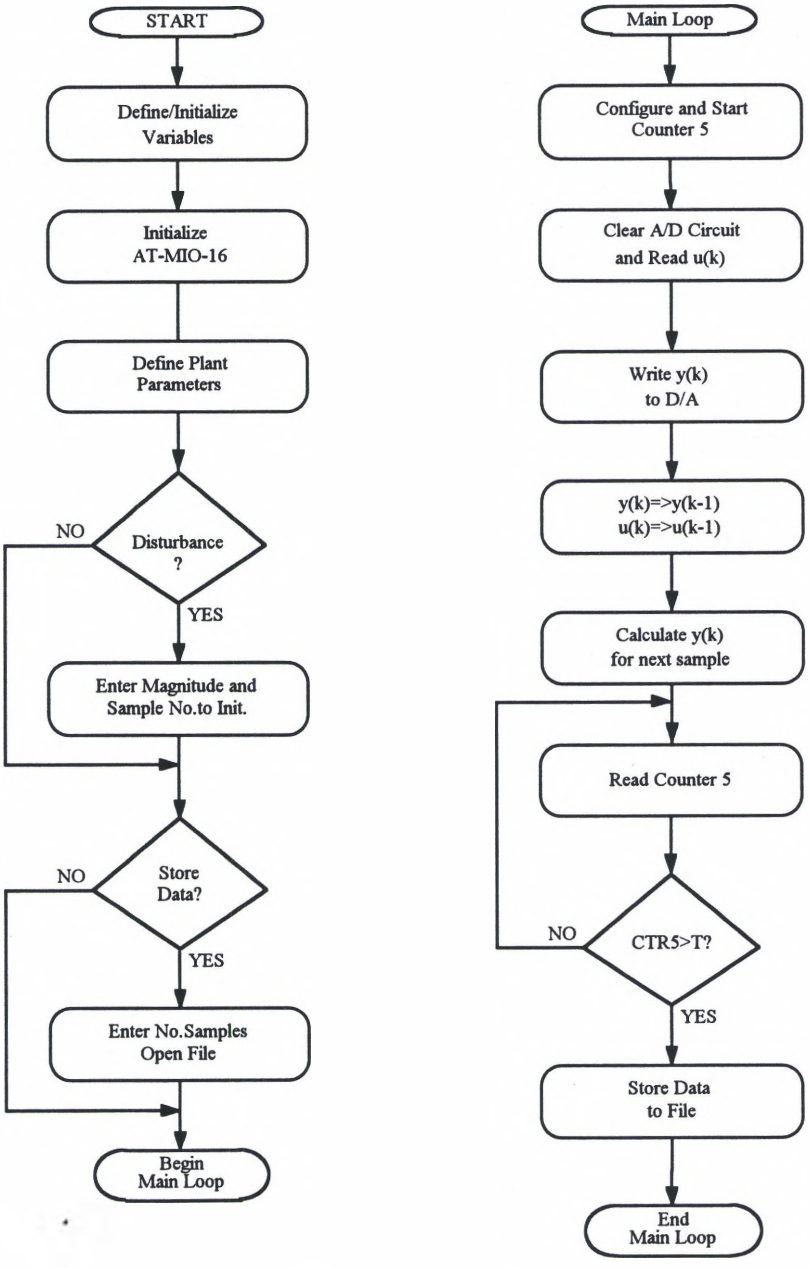

Figure 21. Simplified flowchart of plant model program

reaches the sampling interval preset value. The counter is reset and reinitiated and the main loop repeats until a key on the computer keyboard is pressed.

The DOS LabDriver software provides an extensive library of functions that act as the interface between the QuickBASIC program and the AT-MIO-16 board. The initialization of the AT-MIO-16 is performed via function calls, as well as writing and reading data to and from the A/D and D/A converters. Details for all of the functions are available in the DOS LabDriver User Manual (National Instruments, 1991). The operation of the plant software has been tested by generating step responses from a number of discrete-time models and comparing them to simulation results obtained from Program CC for the same models. The test results demonstrate that the plant model software accurately reproduces the plant models as specified.

3.6 Testing the PID Algorithm

Testing the proportional, integral and derivative terms

Before testing the PID controller in a real-time environment, the operation of the various parts of the DSP PID code are checked under controlled conditions for numerical accuracy. A technique suggested by Astrom and Steingrimsson (1991) is used to test the computation of the proportional and integral terms. The test consists of applying a symmetrical square wave with an amplitude of ± 0.1 and a period of 400 samples as the feedback input to the controller and recording the calculated control output at each sample interval. The control output computed by the DSP can then be compared to theoretical values. The following continuous-time PID controller parameters are chosen for the test:

$$K = 0.6$$
$$T_i = 2.2$$
$$T_d = 0$$
$$\text{T} = 0.1 \text{ sec}$$

From equation (3.3), the discrete-time parameters are calculated as:

$$K' = 0.6136364$$
$$\beta = .9555555.$$

The ADS56000 development system allows the DSP56001 on the ADM board to read and write data to files on the computer hard drive. An input file is created containing 200 samples of +0.1 followed by 200 samples of -0.1. An output file is then opened from the ADS56000 to store the control output samples. The proportional and integral action of the controller can be seen in Figure 22.

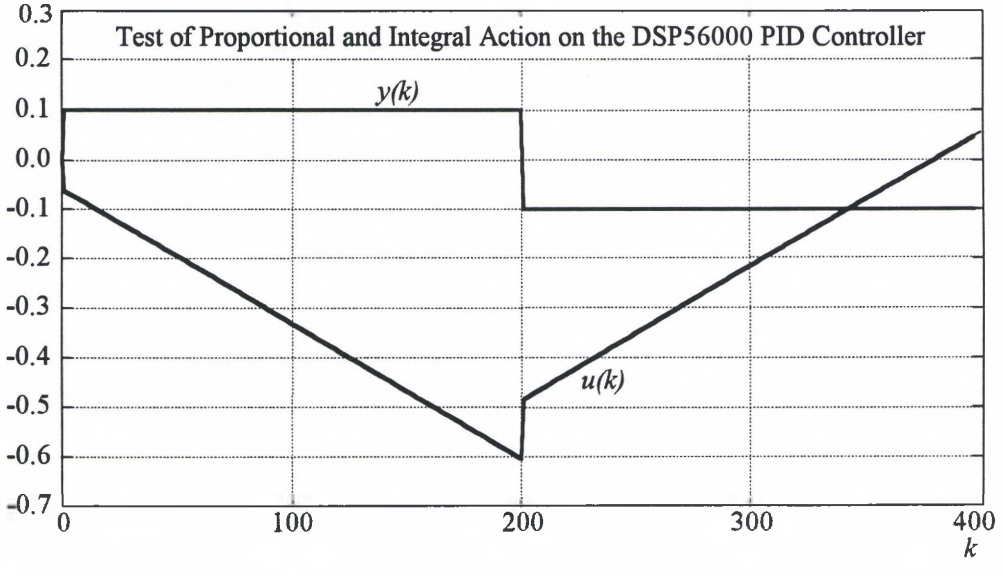

Figure 22. Test of PI algorithm on Motorola DSP56001

With the reference input $w(k)$ held constant at zero, when $y(k)$ is changed from 0 to $+0.1$ at $k = 1$, a constant error of $e(k) = -0.1$ is generated. The proportional term responds to the error by producing an effect on $u(k)$ equal to:

$$K'e(k) = -0.6136...$$

Since $e(k)$ remains constant at $e_0 = -0.1$, the integral term causes $u(k)$ to ramp in the negative direction at a rate of :

$$\frac{e_0}{T_i}k = -0.04545k.$$

At $k = 200$, a step change in $y(k)$ of -0.2 results in an error $e(k) = +0.2$. The proportional term again reacts by increasing $u(k)$ by approximately 1.2. With the error $e(k)$ becoming positive, the integral term causes $u(k)$ to ramp in the positive direction. Theoretical values have been calculated for several values of k and compared with the actual data obtained from the test of Figure 22. The results of the comparison are shown in Table 5.

Table 5. Calculated vs. actual results of PI algorithm test

k	$e(k)$	$u(k)_{theoretical}$	$u(k)_{actual}$	% error
1	-0.1	-0.613636	-0.613637	0.00016
2	-0.1	-0.640909	-0.640911	0.00031
3	-0.1	-0.668182	-0.668185	0.00045
⋮	⋮	⋮	⋮	⋮
100	-0.1	-0.331364	-0.331375	0.00332
⋮	⋮	⋮	⋮	⋮
200	-0.1	-0.604091	-0.604114	0.00381

The actual values closely approximate the values obtained from hand calculations, demonstrating that the algorithm is functioning correctly, although there appears to be a very small cumulative error as k increases.

The derivative term is also tested using a procedure followed by Astrom and Steingrimsson (1991). Two impulses of magnitude +0.1 and -0.1 and lasting one sampling period are applied to the controller input at times $t = 1$ sec and $t = 3$ sec. The continuous-time PID controller parameters are set as follows:

$$K = 0.6$$
$$T_i = 2.2$$
$$T_d = 0.5$$
$$N = 8$$
$$\text{T} = 0.1 \text{ sec}$$

where N is the maximum derivative gain. The discrete-time parameters are calculated from equation (3.3) to be:

$$K' = .6136364$$
$$\beta = .9555555$$
$$d_0 = .0681818$$
$$d_1 = 2.727273.$$

The resulting controller output is shown in Figure 23. The derivative term reacts to the rate of change in the controller input, $e(k)$. When the input to the controller is forced to +0.1 for one sample at $k = 10$, the derivative term reacts as predicted by causing a large negative pulse in $u(k)$. When $e(k)$ changes direction on the very next sample ($k = 11$), the derivative term drives the control output $u(k)$ in the opposite direction. Again, the actual values calculated by the DSP56001 are compared to theoretical values. The results are given in Table 6.

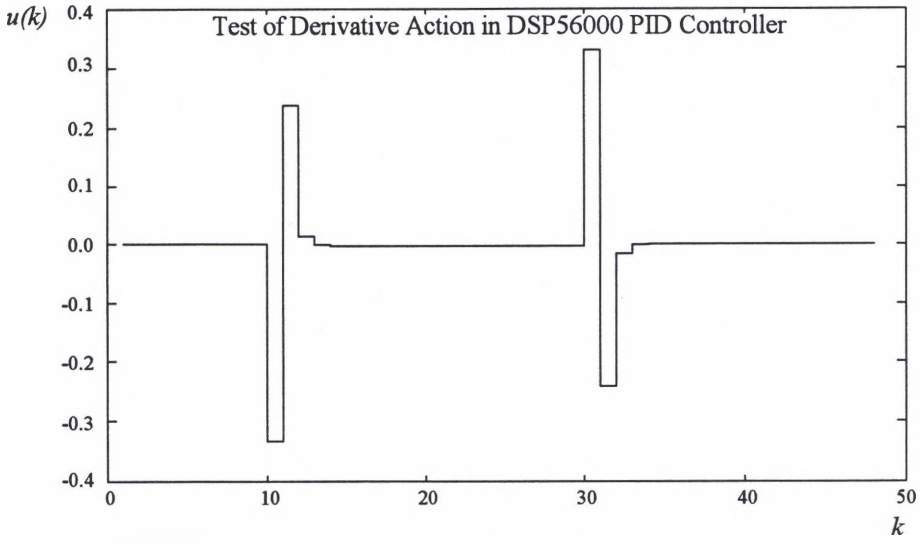

Figure 23. Test of derivative action on Motorola DSP56001

Table 6. Calculated vs. actual results of derivative term test

k	$e(k)$	$u(k)_{theoretical}$	$u(k)_{actual}$	% error
10	+0.1	-.3340909	-.3340902	-0.00021
11	+0.1	.2392837	.2392830	-0.00029
12	+0.1	.0137735	.0137733	-0.00145
13	+0.1	-.0016022	-.0016024	+0.01248
14	+0.1	-.0026506	-.0026509	+0.01132
15	+0.1	-.0027220	-.0027224	+0.01469
⋮	⋮	⋮	⋮	⋮
30	-0.1	.3313636	.3313620	-0.00048
31	-0.1	-.2420110	-.2420105	-0.00021
32	-0.1	-.0165007	-.0165012	+0.00303
33	-0.1	-.0011250	-.0011257	+0.06222
34	-0.1	-.0000767	-.0000774	+0.91265

The results from Table 6 show that the actual values of the control output are very close to the theoretical values. The percentage of error remains less than 0.1% until $k = 34$. Only when the values approach zero does the percentage of error become significant.

Testing the anti-integral windup feature

If the error term remains large for a long period of time (as may occur after a large setpoint change), the integral term can continue to grow long after the output has saturated. One method of eliminating the problem of *integral windup* is the use of a limiting function as shown in Figure 12. The limiting function is given as:

$$\begin{aligned} &\text{If } u_{\min} \le u' \le u_{\max} \Rightarrow u = u' \\ &\text{if} \quad u' < u_{\min} \Rightarrow u = u_{\min} \\ &\text{if} \quad u' > u_{\max} \Rightarrow u = u_{\max}. \end{aligned} \tag{3.8}$$

If $u_{\min}$ and $u_{\max}$ are set to the minimum and maximum limits respectively of the DSP56000 fractional number range, equation (3.8) can be implemented by storing u' using the limiting function of the DSP56000. The function given in equation (3.8) is tested in the DSP56000 program by applying an input of magnitude +0.1 to the controller for 600 samples, allowing the integral term to grow in the negative direction until the controller output saturates at -1.0. After a delay, the input to the controller is changed to -0.1, causing the error term to change signs and the controller output to *integrate* in the opposite direction. The input sequence to the controller is read from an input file from the ADS56000 system and the control output is stored in a file on the computer hard drive. The PID controller parameters are set the same as in the derivative test of the previous section with the sample rate set at T = 0.1 seconds. The results of the test are shown in Figure 24.

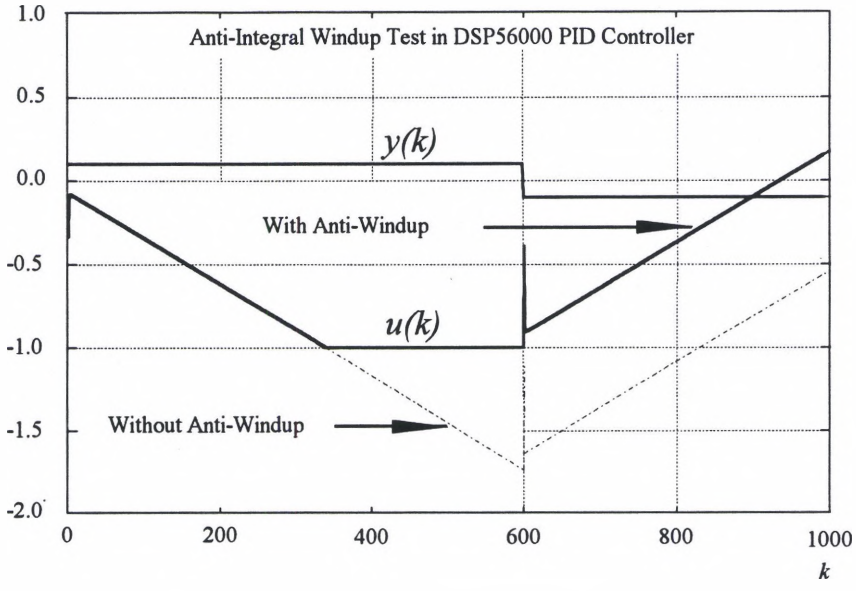

Figure 24. Anti-integral windup test on Motorola DSP56001

The control output $u(k)$ *integrates* in a negative direction until both the control output and the integral term are saturated at a value of -1.0. When the controller input $y(k)$ changes signs at $k = 601$, ($t = 60.1$ sec), $u(k)$ immediately responds positively. The dashed line in the figure shows the effect of *integral windup*. If the integral term had not been limited, the control output would have remain saturated for more than 200 additional samples (20 seconds) causing excessive overshoot in the plant output $y(k)$. (The values of the control output were calculated for several samples and compared to the actual values and the actual values compared closely to the theoretical values.)

The tests of the proportional and integral terms, the derivative term and the anti-integral windup function have demonstrated that the DSP56000 PID control algorithm functions properly and that the PID calculations are numerically accurate. In each case, data was read from a computer input file to ensure predictable results, and the control output was written to an output file so that the actual values could be compared to theoretical results. The DSP56000-based PID controller is now ready for testing in a real-time environment.

3.7 Real-Time Test Results

System Configuration

The purpose of this section is to demonstrate the capability of the DSP-based PID controller operating under real-time conditions. The data for the graphs displayed in this section are taken in real-time from the plant input and output of the AT-MIO-16 board via the plant simulation program. The PID controller setpoint is automatically cycled between a positive and negative value to check the step response of the system. The setpoint is limited to ±0.1 in the experiments so as not to clip the controller output signal. A simulated d.c. load disturbance of -0.1 (equal to the setpoint) is placed on the plant output in the plant simulation program in order to check the disturbance rejection capability of the controller. A block diagram of the test configuration hardware is shown in Figure 25.

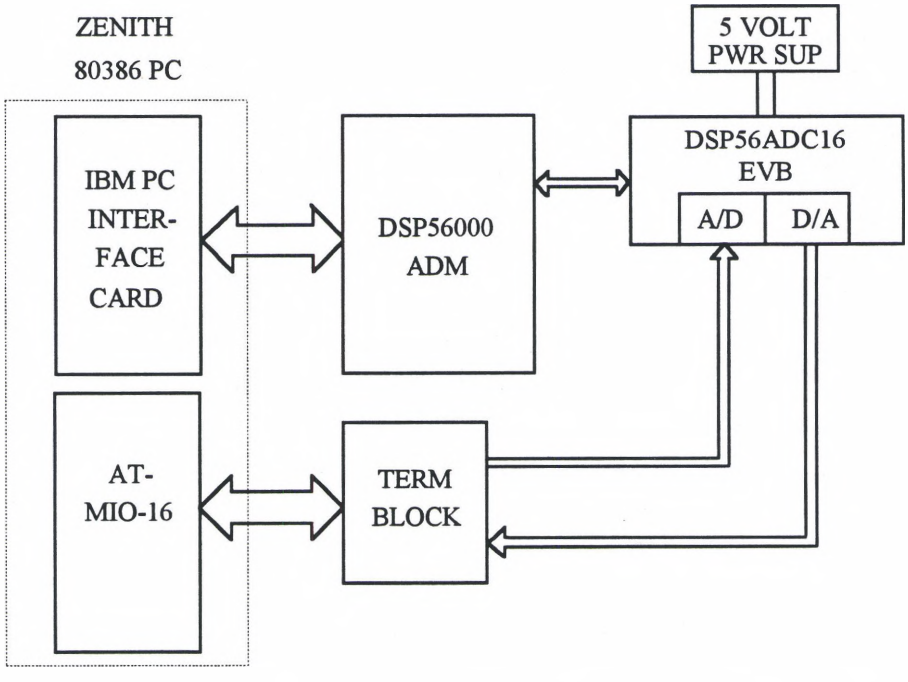

Figure 25. Block diagram of controller-plant test configuration hardware

A fourth-order model was selected as the plant. The continuous-time plant model is expressed in Laplace transform form as:

$$G_p(s) = \frac{1}{(1+s)(1+.2s)(1+.05s)(1+.01s)}. \tag{3.9}$$

Astrom and Hagglund (1988b) used the transfer function of equation (3.9) to analyze several different methods for designing PID controllers. Data from their work proved helpful in analyzing the DSP-based controller in this project. The Bode plot of the continuous-time model is shown in Figure 26.

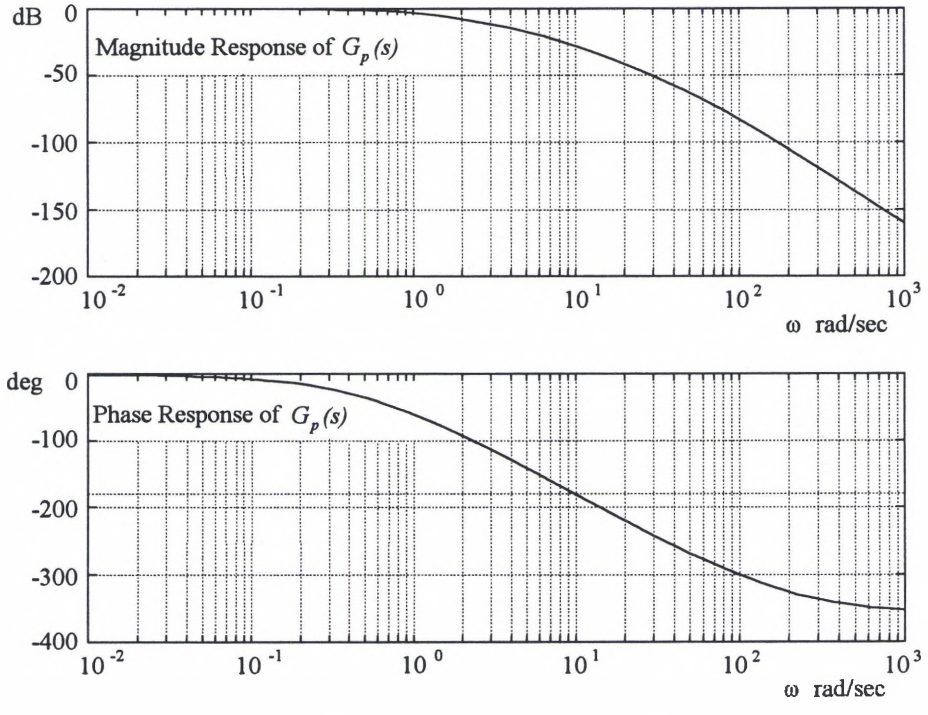

Figure 26. Bode plot of continuous-time plant model

The continuous-time model is converted to discrete-time using the zero-order hold equivalent. The zero-order hold yields a discrete-time transfer function that fits the prescribed model of the plant as described by equation (3.5). A sample time of 0.004 seconds was selected for the plant. (Experimentation revealed that sample times less than 0.004 seconds produce discrete models with extremely small coefficients that lead to

numerical difficulties.) The sampling frequency of 250 Hz (1570.8 rad/sec) yields a Nyquist frequency of 785.4 rad/sec, well beyond the significant bandwidth of the plant. The discrete-time model of the plant is determined to be:

Equation (3.10):

$$G_p(z^{-1}) = \frac{(9.662176\ \mathrm{E}-08)z^{-1} + (9.639456\ \mathrm{E}-07)z^{-2} + (8.716264\ \mathrm{E}-07)z^{-3} + (7.141308E-08)z^{-4}}{1-(3.569643)z^{-1} + (4.7440286)z^{-2} - (2.7784929)z^{-3} + (0.604109335)z^{-4}}$$

The magnitude and phase plots of the discrete model given in equation (3.10) are shown in Figure 27.

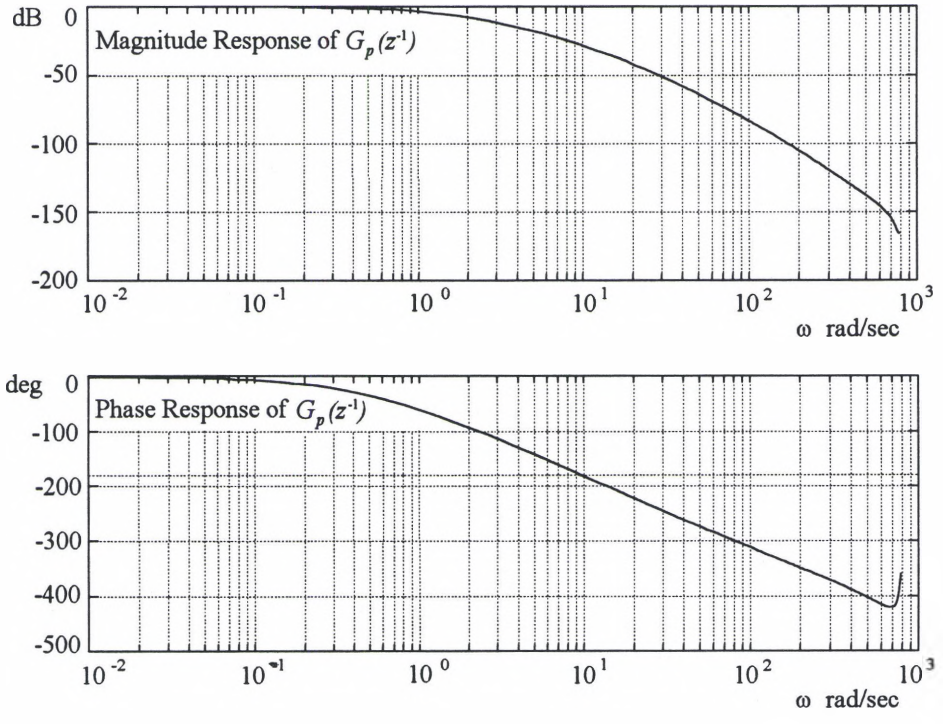

Figure 27. Bode plot of discrete-time plant model used in real-time test

The magnitude response of Figure 27 is identical to the response of the continuous-time model up to the Nyquist frequency. There is a small phase lag in the discrete-time model,

however, that becomes apparent as the frequency exceeds about 10 rad/sec, which is typical of the zero-order hold equivalent.

A sampling frequency of 100 Hz is selected for the controller. (When a sampling frequency of 10 Hz was attempted, the controller proved to be unable to control the dynamics of the plant.) The controller is able to adequately control the plant with a sampling rate of 0.01 seconds. The sampling rate of the plant is therefore 2.5 times faster than the controller's sampling rate. The open-loop step response of the plant operating with a sampling frequency of 250 Hz is shown in Figure 28. The plant exhibits a monotonic step response, making it a suitable candidate for PID control.

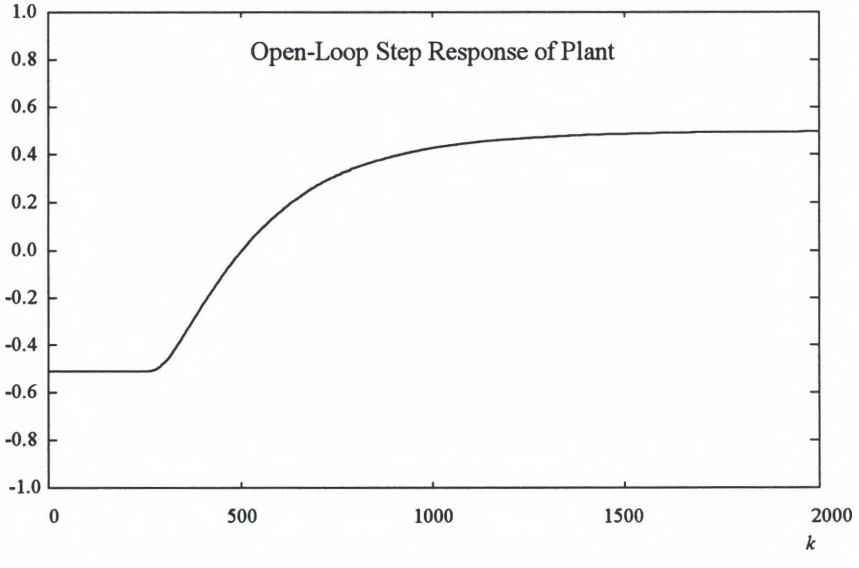

Figure 28. Open-loop step response of plant used in real-time test

Test overview

Four experiments are conducted to test the operation of the PID controller under real-time conditions. In the first two trials, the response of the controller to setpoint changes and load disturbances is tested using two different sets of controller parameters. The first set of controller parameters is obtained by performing the Ziegler-Nichols (1942) frequency response test on the actual plant operating in closed-loop with the DSP-based controller. The second set of controller parameters is obtained from a simulation of Hagglund and Astrom's (1985) auto-tuning PID controller based on the dominant pole design method. In both cases, continuous-time parameters are discretized using the relationships given in equation (3.3). The parameters are also scaled for use in the DSP56000 as described in Section 3.4.

In the third test, the effect of the derivative filter on the controlled plant is observed. The proportional gain K, the integral time T_i and the derivative time T_d are held constant while the maximum derivative gain N is adjusted to four different values. In the fourth test, the maximum sampling rate of the DSP-based controller is determined.

PID controller test using Ziegler-Nichols Frequency Response Method

In this test, the controller parameters are determined using the Ziegler-Nichols (1942) frequency response method performed on the actual plant. With the integral and derivative terms set to zero in the controller, the controller gain is gradually increased until the plant output begins to oscillate. From the experiment, the critical gain, k_c, is determined to be 21.145 and the critical period, t_c, is 0.75848 seconds. (The critical gain must be determined by working backwards from the discrete-time controller gain, which is what is actually being varied.) The controller sample rate is set at T = 0.01 seconds and

the maximum derivative gain is set at $N = 16$. The continuous and discrete-time parameters determined from the critical gain and critical period are listed in Table 7.

Table 7. PID parameters from Ziegler-Nichols frequency response method

Continuous-Time Parameters		Discrete-Time Parameters	
K	12.687	K'	0.2008479
T_i	.37924	β	0.9739746
T_d	.0910176	d_1	0.0644238
		d_2	0.1068936

The reference input is switched between -0.1 and +0.1, and a d.c. load disturbance of -0.1 is introduced at $k = 3000$ ($t = 12$ sec). The resultant plant and controller outputs are shown in Figure 29.

The plant output exhibits a rather large overshoot (approximately 40%) in response to the change in the reference input. Ziegler-Nichols (1942) tuning rules are designed to give superior disturbance rejection and quarter-amplitude damping with $\zeta = 0.22$, but the step response of Figure 29 is somewhat more underdamped than the classic Ziegler-Nichols response. One explanation for this behavior is the discretization of the PID controller using the bilinear transformation. Although the bilinear transform does not produce any phase lag, Astrom and Steingrimsson (1991) report that if the derivative term is discretized using the bilinear transformation, it produces a *ringing* response in the output for small values of T_d. (The value of T_d from Table 7 is .0910176.) Although the plant output is somewhat underdamped, it does respond quite well to a relatively large disturbance ($v = -0.1$) at $k = 3000$.

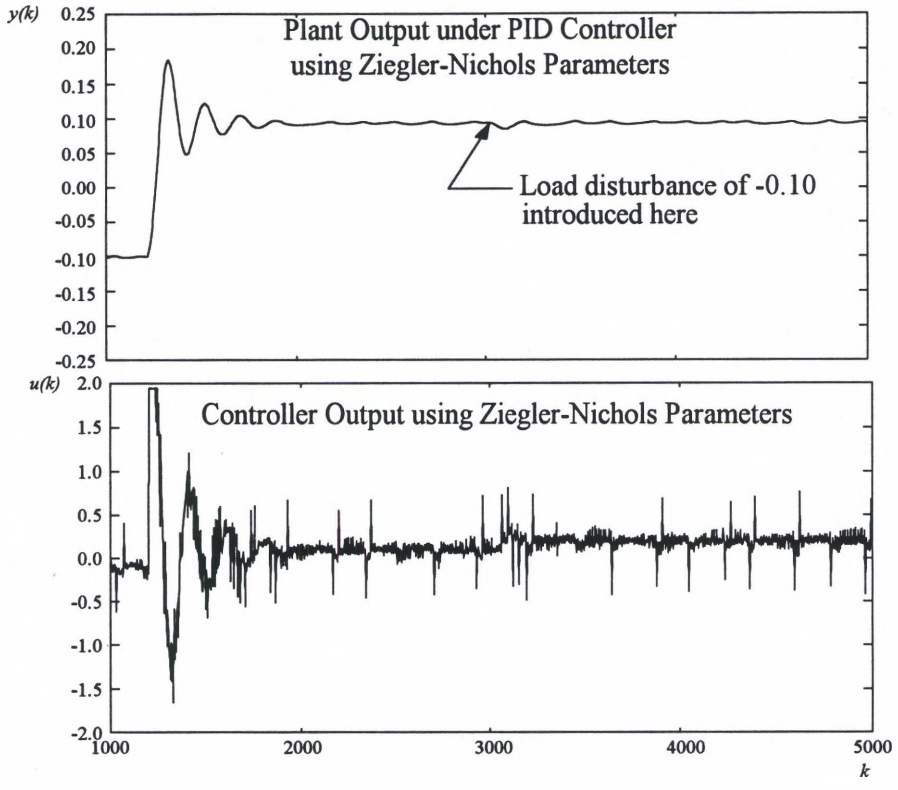

Figure 29. Plant and controller outputs with Ziegler-Nichols parameters

Also notice that the control output is slightly clipped in the first peak of the response. (Recall that the voltage on the EVB card ranges from -2.0 to +2.0 volts.) Some large impulses can be seen in the control output as well. The impulses are due to the response of the derivative term to noise on the output. (An investigation into the source of the noise revealed that it is primarily due to quantization of the 16-bit A/D converter.) For this test, the gain of the derivative filter is set to the maximum value of 16. Further testing showed that the size of the impulses can be significantly reduced by decreasing the value of N. The effect of N is analyzed further later in this chapter.

PID controller test using Hagglund-Astrom (1985) auto-tuner parameters

The plant model expressed in equation (3.9) is used by Hagglund and Astrom (1985) in a continuous-time simulation that tests the auto-tuning PID controller algorithm based on the dominant pole design method. The PID parameters obtained from the continuous-time simulation of Hagglund and Astrom (1985) are used to test the DSP-based PID controller further. The continuous-time parameters of Hagglund and Astrom and their scaled, discrete-time equivalents are given in Table 8.

Table 8. PID parameters from Hagglund-Astrom auto-tuner

Continuous-Time Parameters		Discrete-Time Parameters	
K	9.62	K'	0.15184
T_i	.492	β	0.97988
T_d	.123	d_1	0.2118
		d_2	0.092

The proportional gain K' in Table 8 is smaller than the value of K' obtained using Ziegler-Nichols (1942), and the derivative terms d_1 and d_2 in Table 8 are larger than the previous values Again, with $N = 16$, the resultant outputs are shown in Figure 30. The plant response is considerably improved over the previous test. The overshoot is smaller (down from 40% to 25%), there is less ripple in the steady-state output and the disturbance rejection response is better damped. The smaller proportional gain and the larger derivative terms produced a response that is more damped than in the previous case.

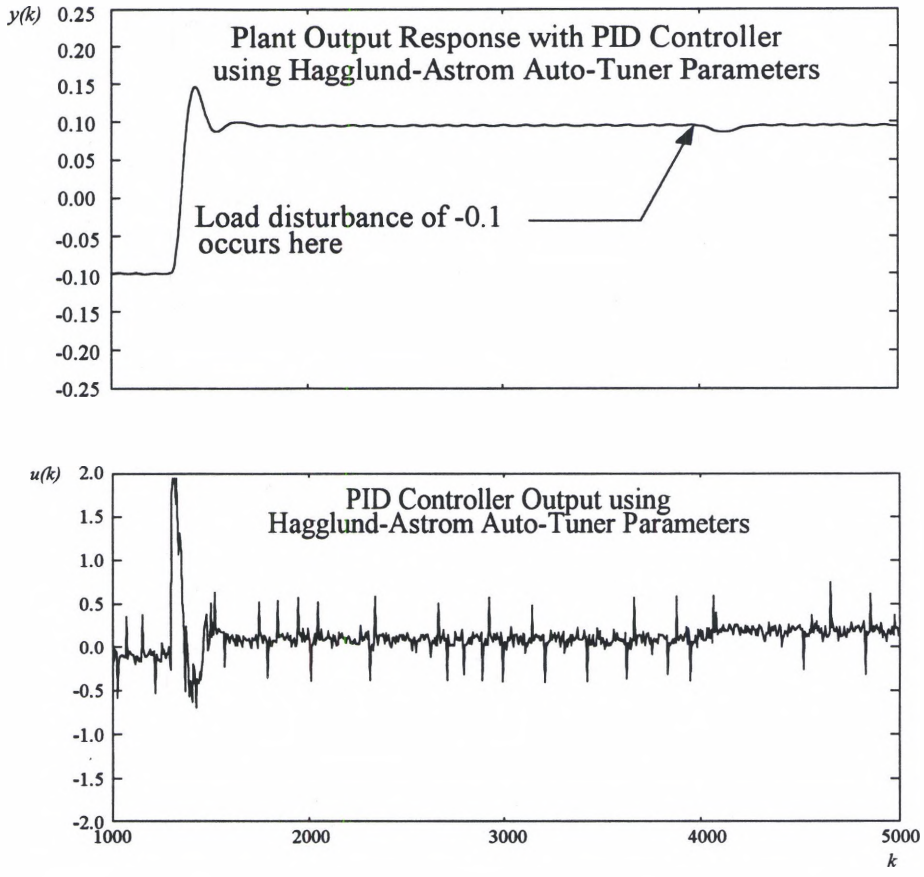

Figure 30. Plant and controller outputs with Hagglund-Astrom parameters

Testing the derivative term filter

The previous two trials have demonstrated the capability of the DSP-based PID controller to respond to changes in the reference input and to reject disturbances on the output. In both tests, rather large impulses were observed on the control output as a result of the derivative term reacting to measurement noise. Testing indicated that the size of the impulses could be reduced by decreasing N, the maximum derivative gain. In this section, the derivative term filter is evaluated. The controller parameters will remain the same as in the previous case. Again, a disturbance is introduced at $k = 4000$. The effect of N on d_1 and d_2 can be seen in Table 9.

Table 9. Effect of N on derivative term coefficients

N	d_1	d_2
16	0.211823	0.092002
8	0.509202	0.057289
4	0.720279	0.032651
2	0.849264	0.017553
1	0.921875	0.009119

Recalling that the derivative term, $D(k)$, is expressed as:

$$D(k) = d_1 D(k-1) + d_2[y(k-1) - y(k)],$$

as N is increased, the previous value of $D(k)$ becomes more heavily weighted, effectively filtering $D(k)$ from higher frequencies. Figures 31 through 34 show the plant output for $N = 16$, 4, 2 and 1, respectively. Although there is some difference in the response between $N = 16$ and $N = 4$, the response of the plant begins to seriously degrade at $N = 2$. Obviously, the selection of N must be made carefully as it represents a tradeoff between filtering unwanted noise and impeding the action of the derivative term. The results do indicate, however, that the derivative filter is functioning as predicted.

Establishing the maximum controller sampling rate

One of the primary reasons for using a DSP chip for control is increased processor throughput to achieve the maximum possible sampling frequency. In the final test of the DSP56000-based PID controller, the maximum sampling rate is determined. A function generator is attached to the input of the controller. An oscilloscope is attached to the

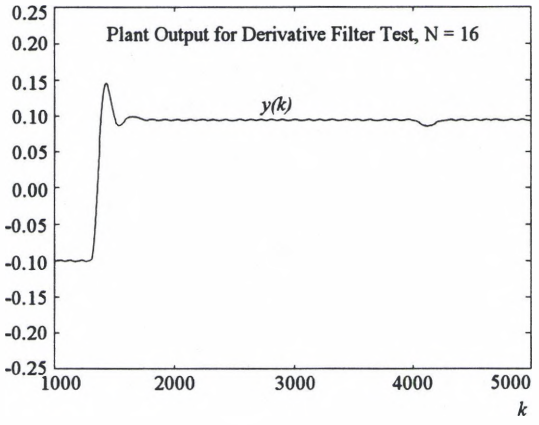

Figure 31. Plant output for $N = 16$

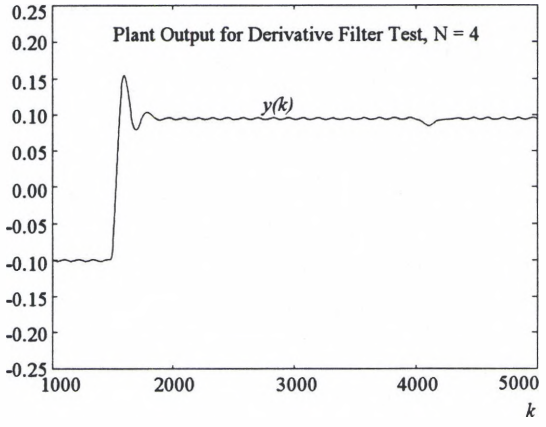

Figure 32. Plant output for $N = 4$

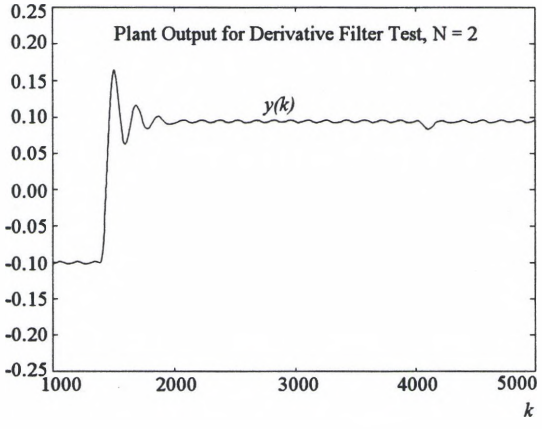

Figure 33. Plant output for $N = 2$

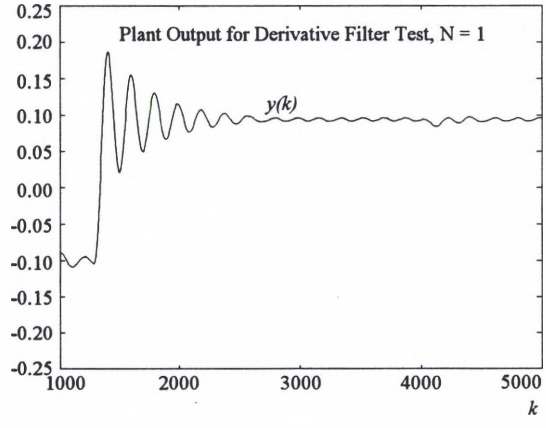

Figure 34. Plant output for $N = 1$

controller output so that the sampling frequency of the controller can actually be measured. The sampling frequency of the controller is controlled by the SCI timer that generates interrupts to the processor. The maximum frequency that the SCI timer can be set to operate at is :

$$f_{osc}/64 \approx 320\ kHz.$$

This corresponds to a sample interval of 3.12 μsec. With the SCI timer set at the maximum frequency, the controller's sampling frequency is measured to be approximately 48 KHz. This is the *output* sampling frequency of the DSP56ADC16 A/D converter, indicating that the A/D converter is the limiting factor. As a check, the number of

oscillator cycles required to execute the main loop of the PID program is calculated to be 200. At 20.5 MHz, this estimates the execution time of the PID code to be roughly 10 μsec, which corresponds to a sampling frequency of 100 KHz, approximately double the sampling frequency measured on the controller output. The limiting factor on controller sampling frequency is therefore the DSP56ADC16. Higher sampling frequencies are theoretically achievable (up to 100 KHz) if a faster responding A/D converter is used.

3.8 Chapter Summary

In this chapter, the PID algorithm developed in Chapter 2 was implemented on the Motorola DSP56000. The question of why to use a DSP chip for control was addressed. The Harvard architecture of the DSP was shown to provide a number of advantages over the Von Neuman architecture of most general-purpose microprocessors. The hardware and software features of the Motorola DSP56000 that make it attractive for control applications were also discussed. The function of the on-chip peripherals was explained as well. Included in the discussion were the use of the SCI timer interrupt to control sampling rate, and the operation of the synchronous serial interface (SSI) used to interface with the DSP56ADC16 A/D converter.

The PID algorithm was implemented in Motorola DSP56000 assembly language. A simplified flowchart of the program was presented and a brief explanation of the program was given. The need for scaling of the PID controller coefficients for DSP56000 implementation was pointed out and appropriate scaling factors were determined for the controller coefficients. The program was organized to minimize the effect of computational delay and was written to take advantage of the numerical stability of a

parallel computational structure, forming each of the three PID terms independently and then summing them to form the control output.

The development of the plant model used for real-time testing of the PID controller was presented as well. A deterministic version of the ARX model was selected for the plant. The National Instruments AT-MIO-16 A/D, D/A converter board was examined, and the DOS LabDriver software to drive it was briefly covered. The program written to implement the plant model was also explained.

The DSP56000-based PID controller was then tested. The function of the three PID controller terms was first tested in a controlled mode where each of the three controller terms could be evaluated independently and compared to theoretical data. The numerical accuracy of each of the three controller terms was established, and the function of the anti-integral windup feature was verified. The PID controller was then tested in real-time connected to a functioning plant. Two sets of controller parameters were selected to test the effect of the controller parameters on the response of the plant. The action of the derivative filter was also tested in a real-time mode. Finally, the maximum sampling rate of the DSP56000-based controller was established.

In conclusion, the PID controller implemented on the DSP56000 functioned as designed in all cases.

CHAPTER IV

DEVELOPMENT OF THE

ADAPTIVE PID ALGORITHM

4.1 Introduction

Having developed a working, practical PID controller on the Motorola DSP56000, the question remains as to how to make the PID controller adaptive. The concept of adaptive control was introduced in Chapter 1 and an overview of the most commonly employed adaptive control schemes was presented. Gain scheduling is currently being successfully implemented and could easily be implemented using a PID controller; however, it does have several drawbacks. Gain scheduling requires that a suitable model of the plant be available. Even if a good model of the plant is available, several sets of controller parameters must be obtained for the different ranges of variation of the plant parameters. The procedure can be time-consuming and is also prone to transition problems between gain sets. Self-oscillating adaptive systems have proven to be robust and are commonly used in flight control systems for missiles. The application of self-oscillating adaptive systems is somewhat limited, however, in that a limit cycle oscillation is usually discernible in the output. Also inherent in the concept is a trade-off between the

amplitude of the induced limit cycle and the system's ability to respond to command signals. It also does not lend itself well to adaptive PID control. Model reference adaptive control (MRAC) remains a subject of great interest to researchers in the field of adaptive control. The concept has been extended to include non-minimum phase systems, multivariable systems and nonlinear systems. MRAC is rarely implemented in a PID-like structure, however, although Clarke and Gawthrop (1975) proposed an adaptive PID controller that applied a model reference controller using least-squares estimation.

A survey of the literature indicates that a considerable amount of work has been done using a self-tuning regulator structure for adaptive PID control. A key to the successful design of a self-tuning regulator is the selection of an appropriate parameter estimation method. Since the parameter estimator must function *on-line* in an adaptive controller, the estimation algorithm must perform recursively. Most recursive estimation methods are derived from an off-line counterpart. The off-line method is extended to perform the calculation one iteration at a time using a newly acquired data element at each iteration. Several different recursive estimation schemes have been developed, including recursive least-squares (RLS), recursive extended least-squares (RELS), recursive maximum likelihood (RML) and recursive instrumental variables (RIV). By far, the most widely used recursive estimation method is recursive least squares. Actually, all of the above methods are based on some variation of the RLS algorithm. Ljung and Soderstrom (1983) go so far as to state that "There is only one recursive identification method. It contains design variables to be chosen by the user".

For a system to be identifiable, some restrictions must be placed on the input signal to the parameter estimator. The input signal must have sufficient frequency content to adequately excite all modes of the system. An input that meets this requirement is said to be *persistently exciting*. It can be shown that if the input is persistently exciting and the parameter errors are uncorrelated with each other and with the input signal, the least

squares-estimates will converge to the true values of the parameters. Severe numerical problems can occur, however, when the input is not persistently exciting. Some of the matrices used in the calculation of the estimated parameters can become singular or near singular causing the estimation algorithm to fail. Numerical difficulties can also occur when numerical inaccuracies develop in the computations due to roundoff. Variations of the basic least-squares method, called *square root algorithms* have been successfully used to overcome some of these numerical difficulties. By using the square root of the covariance matrix in the calculations, the effective dynamic range of the calculations is limited. One such method proposed by Bierman (1977) is known as *U-D cofactorization*. Other modifications to the least-squares algorithm have been proposed as well. Astrom and Wittenmark's (1973) original self-tuning regulator assumes that the plant parameters remain constant, which is why they refer to it as *self-tuning* as opposed to *adaptive*. The idea has been extended, however, to include plants with time-varying parameters. A *forgetting factor* can be introduced into the least-squares algorithm that discounts the effect of past data on the calculations, thus allowing the algorithm to track time-varying parameters.

Other difficulties must also be considered when implementing recursive parameter estimation. It can be shown that disturbances in the input-output data to the estimator result in biased estimates. Filtering of the data may therefore be required, as disturbances will disguise the true dynamics of the plant to the estimator. The effects of measurement noise in the data can be minimized by low-pass filtering. Several different techniques, including the use of high-pass filters, may be used to remove offsets and low frequency disturbances from the input and output signals. Very often, a band-pass filter is used to filter the estimator data. The allowable bandwidth of the filter depends on the bandwidth of interest of the plant being identified. The filter must be designed specifically for the

plant in question; therefore, some knowledge of the plant dynamics must be known in advance if data filtering is to be utilized.

Special considerations must also be made when parameter estimation is performed on a plant under closed-loop control. A criterion for being able to identify a plant is that the input signal to the estimator must be independent of the residuals (estimation errors). Although this is typically true in an open-loop case, it is not necessarily true in a closed-loop situation. In fact, it can be demonstrated that when simple proportional feedback is used, the controller output and the residuals are correlated. Assuming the reference input is constant, once the controller has brought the plant output to equilibrium, no new information about the plant dynamics is provided to the estimator. If a forgetting factor has been introduced into the algorithm, the estimator will then tend to *forget* what it has learned about the plant dynamics and the covariances of the parameters will begin to increase exponentially, a condition known as *estimator wind-up*. When new information about the plant becomes available to the estimator, as is the case when the plant parameters change, the covariance matrix is saturated and the estimator is unable to identify the changes in the parameters. Several methods have been proposed to deal with estimator windup, such as covariance resetting, time variable forgetting factors, constant trace algorithms and directional forgetting. Probably the simplest method is to reset the covariances to some initial values if a limit is reached, although this may introduce a time-lag from when the parameters change to when the estimator responds, depending on when the estimator covariances were last reset. A related problem, known as *bursting*, can also occur when attempting identification of a plant operating in a closed-loop. When the plant reaches equilibrium due to feedback and the input to the estimator is no longer sufficiently exciting, the estimated parameters may drift, causing instability in the output. As the output becomes more unstable, the input to the estimator is excited to the point where the estimator is again able to identify the plant parameters. The controller can then bring the

plant back under control and the process repeats itself indefinitely. One way to deal with bursting is to turn off the estimator when the parameter error becomes sufficiently small. A *dead-zone* is thus introduced into the algorithm that allows the estimator to work only when the parameter error becomes significant. The size of the dead-zone is another parameter that must be adjusted and depends on the noise floor of the given application.

As evidenced in the previous discussion, recursive parameter estimation is not a trivial task, particularly when attempted in a closed-loop, and many ad hoc methods have been developed to alleviate some of the inherent difficulties encountered. The second main element of the self-tuning controller is the on-line control design mechanism. The minimum variance controller originally proposed by Astrom and Wittenmark (1973) was extended by Clarke and Gawthrop (1975) who proposed a technique known as *generalized minimum variance control*. The work of Astrom and Wittenmark and Clarke and Gawthrop spawned development into many other control law design variations, including pole-placement and LQG design. When the controller is to conform to a PID-like structure, the number of options for choosing an underlying design principle becomes somewhat limited. To accommodate a PID controller structure, generally some type of pole-placement algorithm is used, and the model order of the parameter estimator is limited to second-order. Imposing a limit on the model order of the estimator limits its application to plants that can be adequately modeled assuming lower order dynamics only. Another factor to consider is the computational complexity of the design mechanism. If the controller will be required to operate at high sampling rates or if the algorithm is to be implemented on a microprocessor or a DSP chip, some of the pole-placement techniques that require on-line solution of simultaneous linear equations may not be suitable. The algorithm should also be robust, in the sense that it should be able to perform in presence of random disturbances.

Warwick, Karam and Tham (1987) proposed a self-tuning controller algorithm that meets most of the controller objectives just mentioned. The proposed control law design mechanism uses the estimated plant model parameters directly in the control law and is thus considered an *implicit* or *direct* realization. The algorithm is computationally efficient, making it suitable for implementation on a microprocessor or DSP chip. The primary objective of the algorithm is setpoint tracking, and the controller design is based on a *deadbeat* control strategy that attempts to cancel the dominant process poles, but not the process zeros. This allows the controller to be used with non-minimum phase plants. The algorithm is also versatile, in that it can be easily modified to work with a number of different control strategies. For instance, although the proposed self-tuning controller is not specifically designed for PID control, a PID-like controller can be realized by imposing specific limitations on the general controller structure.

The disadvantage of the algorithm proposed by Warwick, *et. al.* is that the closed-loop dynamics of the system are fixed by the pole cancellation of the deadbeat control strategy. Cancellation of the plant poles may allow for good reference input tracking, but the response to load disturbances may be less than optimal. Pole-placement offers an advantage over deadbeat control in that the locations of the closed-loop system poles can be arbitrarily selected. Some of the early work on self-tuning PID controllers based on pole-placement was done by Wittenmark (1979) and Astrom and Wittenmark (1980). McInnis, *et.al.* (1985) modified the pole-placement algorithm of Astrom and Wittenmark (1980) so that the two extra closed-loop zeros are placed at the origin, simplifying the calculations required in the implementation.

In this chapter, an adaptive PID algorithm is developed. A recursive least-squares estimator using U-D cofactorization as proposed by Bierman (1977) is used for the parameter estimator. The estimation scheme incorporates exponential forgetting and resetting for tracking time-varying parameters. In order to deal with estimator windup

and bursting, the estimator also includes a dead-zone that causes it to shut-off when the parameter error drops below a given threshold. Two different control law design mechanisms are also presented in the chapter. First, the *simple self-tuning control* (SSTC) algorithm proposed by Warwick, Karam and Tham is developed. The basic SSTC algorithm is modified to accommodate a PID-like controller structure by imposing certain constraints on the algorithm. The second control law design mechanism presented is a pole-placement algorithm proposed by McInnis, *et. al.* The pole-placement method allows for a *setpoint-on-I-only* PID structure which minimizes proportional and derivative kick. The remainder of the chapter deals with the development of the two outer loop sections of the adaptive controller: the parameter estimation algorithm and the control law design mechanism.

4.2 Open-Loop Parameter Estimation

Development of the recursive least-squares algorithm

A critical step in the system identification process is the definition of a suitable model for the plant. One of the most commonly used parametric models in system identification is the autoregressive moving average with exogenous input (ARMAX) model. The ARMAX model is defined as:

$$A(q)y(t) = B(q)u(t) + C(q)\varepsilon(t)$$

where:

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$$
$$B(q) = b_1 q^{-1} + \cdots + b_n q^{-n}$$
$$C(q) = 1 + c_1 q^{-1} + \cdots + c_n q^{-n}$$

$\varepsilon(t)$ is a sequence of uniformly distributed, independent random disturbances, q^{-1} is the backward shift operator and t are samples taken in discrete time. The ARMAX model allows for modeling of non-white noise by the inclusion of a coloring filter, $C(q)$. However, if the noise, $\varepsilon(t)$, is assumed to be white, a simplified version of the ARMAX model, called the ARX model, can then be used. The ARX model is defined as:

$$A(q)y(t) = B(q)u(t) + \varepsilon(t) \tag{4.1}$$

where :

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$$
$$B(q) = b_1 q^{-1} + \cdots + b_m q^{-m}.$$

Equation (4.1) can also be written in a familiar form as:

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_m u(t-m) + \varepsilon(t). \tag{4.2}$$

By assuming $\varepsilon(t)$ to be white noise, the ordinary least-squares estimator can then be used to estimate the parameters of the model. The ensuing development of the recursive least-squares algorithm follows Ljung and Soderstrom (1983).

A parameter vector θ is defined as:

$$\theta^T = [a_1 \ldots a_n \; b_1 \ldots b_m]$$

and a *regression* vector is defined as:

$$\varphi^T(t) = [-y(t-1) \ldots -y(t-n) \; u(t-1) \ldots u(t-m)].$$

Equation 4.2 can then be written as:

$$y(t) = \theta^T \varphi(t) + \varepsilon(t).$$

Having defined the model structure, the problem is to estimate the parameter vector, θ. One way to do this is to minimize what is left unexplained by the model, i.e., $\varepsilon(t)$. A cost function can thus be constructed as:

$$V_N(\theta) = \frac{1}{N} \alpha_t \sum_1^N \left[y(t) - \theta^T \varphi(t) \right]^2 \tag{4.3}$$

where $V_N(\theta)$ is minimized with respect to θ. (α_t is a series of positive numbers that allow different weights to be given to different observations. Setting α_t equal to one gives all the data equal weight.) Since $V_N(\theta)$ is a quadratic in θ, it can be minimized analytically. The least squares estimate of θ is defined as $\hat{\theta}(N)$ and can be determined to be:

$$\hat{\theta}(N) = \left[\sum_{t=1}^{N} \alpha_t \varphi(t)\varphi^T(t)\right]^{-1} \sum_{t=1}^{N} \alpha_t \varphi(t) y(t). \tag{4.4}$$

In order to solve equation (4.4) recursively, a new vector, $\overline{R}(t)$, is defined as:

$$\overline{R}(t) = \sum_{k=1}^{t} \alpha_k \varphi(k)\varphi^T(k).$$

Then from equation (4.4):

$$\sum_{k=1}^{t-1} \alpha_k \varphi(k) y(k) = \overline{R}(t-1)\hat{\theta}(t-1).$$

Therefore:

$$\overline{R}(t-1) = \overline{R}(t) - \alpha_k \varphi(t)\varphi^T(t)$$

which yields an expression for $\hat{\theta}(t)$:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \overline{R}^{-1}(t)\varphi(t)\alpha_t\left[y(t) - \hat{\theta}^T(t-1)\varphi(t)\right]$$

with:

$$\overline{R}(t) = \overline{R}(t-1) + \alpha_t \varphi(t)\varphi^T(t).$$

If $\mathbf{R}(t)$ is defined as:

$$\mathbf{R}(t) = \frac{1}{t}\overline{R}(t),$$

then it can be shown that:

$$\mathbf{R}(t) = \mathbf{R}(t-1) + \frac{1}{t}\left[\alpha_t \varphi(t)\varphi^T(t) - \mathbf{R}(t-1)\right],$$

thus giving:

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + \frac{1}{t}\mathbf{R}^{-1}(t)\varphi(t)\left[y(t) - \hat{\theta}^T(t-1)\varphi(t)\right] \\ \mathbf{R}(t) &= \mathbf{R}(t-1) + \frac{1}{t}\left[\alpha_t \varphi(t)\varphi^T(t) - \mathbf{R}(t-1)\right]. \end{aligned} \tag{4.5}$$

Unfortunately, equations (4.5) are not well suited for on-line computations as they require the inversion of $\mathbf{R}(t)$ at each sampling interval. A new variable is therefore introduced:

$$\mathbf{P}(t) = \overline{R}^{-1}(t) = \frac{1}{t}\mathbf{R}^{-1}(t).$$

The *matrix inversion lemma* is given as:

$$\left[\mathbf{A}+\mathbf{BCD}\right]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left[\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right]^{-1}\mathbf{DA}^{-1} \quad . \tag{4.6}$$

Applying the lemma of (4.6) to equation (4.5) with:

$$\mathbf{A} = \mathbf{P}(t-1),\ \mathbf{B} = \varphi(t),\ \mathbf{C} = \alpha_t,\ \mathbf{D} = \varphi^T(t)$$

gives:

$$\begin{aligned}\mathbf{P}(t) &= \left[\mathbf{P}^{-1}(t-1) + \varphi(t)\alpha_t\varphi^T(t)\right]^{-1} \\ &= \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\varphi(t)\varphi^T(t)\mathbf{P}(t-1)}{1/\alpha_t + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)}.\end{aligned} \tag{4.7}$$

The result of equation (4.7) is that the inversion of a square matrix is replaced by the inversion of a scalar. The recursive least-squares algorithm for determining $\hat{\theta}(t)$ is thus determined to be:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathbf{L}(t)\left[y(t) - \hat{\theta}^T(t-1)\varphi(t)\right] \tag{4.8a}$$

$$\mathbf{L}(t) = \frac{\mathbf{P}(t-1)\varphi(t)}{1/\alpha_t + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)} \tag{4.8b}$$

$$\mathbf{P}(t) = \left[1/\alpha_t - \mathbf{L}(t)\varphi^T(t)\right]\mathbf{P}(t-1) \tag{4.8c}$$

It is seen in algorithm (4.8) that initial conditions $\hat{\theta}(0)$ and $\mathbf{P}(0)$ must be established for the vectors $\hat{\theta}(t)$ and $\mathbf{P}(t)$. The *correct* initial values would be obtained if the recursion of algorithm (4.8) did not begin until time $t_o = \dim\varphi(t) = \dim\theta$ (which is when $\overline{\mathbf{R}}(t)$ of equation (4.5) becomes invertable). However, since the estimation normally begins at time $t = 0$, some arbitrary values for $\hat{\theta}(0)$ and $\mathbf{P}(0)$ must be selected. According to Ljung and Soderstrom (1983), it can be shown that as $\mathbf{P}^{-1}(0) \to 0$, the value of the recursive estimate approaches the value of the off-line estimate. The initial conditions for $\hat{\theta}(t)$ and $\mathbf{P}(t)$ are therefore typically chosen to be:

$$\mathbf{P}(0) = a\mathbf{I} \quad \text{where } a \text{ is a large constant}$$

and $$\hat{\theta}(0) = 0.$$

The question now arises as to how the algorithm behaves as the number of samples grows large. It can be shown that the recursive least-squares estimate of the parameters $\hat{\theta}(t)$ will converge to the true parameters $\theta(t)$ under the following conditions:

1. The input $\{u(t)\}$ is *persistently exciting*
2. The residuals $\{\varepsilon(t)\}$ are independent
3. The input sequence $\{u(t)\}$ is independent of the disturbance sequence $\{\varepsilon(t)\}$.

Ljung and Soderstrom (1983) define *persistently exciting* as follows:

> Let $\{u(t)\}$ be such that the limits
>
> $$\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} u(t)u^T(t-j) \stackrel{\Delta}{=} r(j) \quad \text{exist for all } 0 \le j \le n.$$
>
> Form the $n \times n$ block matrix R_n whose i,k block entry is $r(i-k)$. The sequence $\{u(t)\}$ is then said to be *persistently exciting of order n*, if R_n is nonsingular.

Astrom and Wittenmark (1989) investigated the persistence of excitation of several special signals of interest. Their conclusions are presented in Table 10.

Table 10. Order of persistence of excitation of various signal types

Signal Type	Order n of Persistence of Excitation
Pulse	Not PE for any n
Step	PE of order 1
Sinusoid	PE of order 2
Periodic Signal with period n	At most, PE of order n
Random signals	PE of any order

It can be shown that a signal that is persistently exciting of order n has a spectral density that is non zero at least n points. Persistence of excitation, therefore, implies that the signal contains sufficient frequency content to excite all of the modes of the system in question.

Tracking time-varying parameters

The recursive least-squares algorithm expressed in algorithm (4.8) assumes that the system being identified is time-invariant. One of the primary reasons for using adaptive control, however, is to compensate for system parameters that vary slowly over time. The least-squares cost function was given in equation (4.3) as:

$$V_N(\theta) = \frac{1}{N}\alpha_t \sum_1^N \left[y(t) - \theta^T \varphi(t)\right]^2 .$$

If the parameters are time-varying, the criterion of equation (4.3) gives an estimate of the *average* behavior over the interval from 1 to N. The criterion of equation (4.3) can be modified to discount older data and thus provide an estimate of the *current* values of the parameters. The modified criterion is given as:

$$\tilde{V}_N(\theta) = \sum_1^N \lambda^{t-k} \alpha_k \left[y(t) - \theta^T \varphi(t)\right]^2 \qquad (4.9)$$

where λ is referred to as the *forgetting factor*. If $\lambda = 1$, all of the data is weighted equally; however, if $\lambda < 1$, recent data are weighted more heavily and an exponential forgetting profile is imposed upon the least-squares cost function of equation (4.3). If $\overline{R}(t)$ is redefined to be:

$$\overline{R}(t) = \sum_{k=1}^{t} \lambda^{t-k} \alpha_k \varphi(k) \varphi^T(k),$$

the recursive least squares algorithm can be redeveloped as:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathbf{L}(t)\left[y(t) - \hat{\theta}^T(t-1)\varphi(t)\right] \quad (4.10a)$$

$$\mathbf{L}(t) = \frac{\mathbf{P}(t-1)\varphi(t)}{\lambda/\alpha_t + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)} \quad (4.10b)$$

$$\mathbf{P}(t) = \frac{1}{\lambda}\left[1/\alpha_t - \mathbf{L}(t)\varphi^T(t)\right]\mathbf{P}(t-1) \quad (4.10c)$$

Observation of algorithm (4.10) reveals that the normal RLS algorithm (4.8) is merely a special case of algorithm (4.10) with $\lambda = 1$. In the standard RLS algorithm, as the estimates converge, $\mathbf{P}(t)$ tends to go to zero. Making $0 < \lambda \leq 1$ has the effect of keeping $\mathbf{P}(t)$, and thus $\mathbf{L}(t)$, relatively large. The algorithm will then remain active, able to track the time-varying parameters. The final value at which $\mathbf{L}(t)$ converges will ultimately be a compromise between tracking ability and noise sensitivity. Ljung and Soderstrom (1983) state that it is impossible to track rapidly varying parameters; however, slowly varying parameters can often be tracked reasonably well. If some prior knowledge of the variation of the parameters is available, λ can be made to be dynamic as $\lambda(t)$; however, if prior knowledge of the variability of the parameters is not available, the forgetting factor is generally be chosen to be a constant. Assuming α_k is equal to one, equation (4.9) can be expressed as:

$$V_N(\theta) = \sum_{t=1}^{N} \lambda^{N-t}\varepsilon^2(t,\theta).$$

When λ is close to one,

$$\lambda^t = e^{t\ln\lambda} = e^{t\ln(\lambda-1+1)} \approx e^{t(\lambda-1)}$$

which results in an exponentially decaying time constant of:

$$T_0 = \frac{1}{1-\lambda}$$

where T_o is referred to as the *memory time constant*. T_o should be chosen to coincide with the expected rate of variation of the parameters. According to Isermann (1982), λ must

be maintained between $0.95 \leq \lambda \leq 0.995$ in most cases. Table 11 shows the effect of the relative weighting of the forgetting factor over a 50 sample interval for $\lambda = 0.99$ and $\lambda = 0.95$. The current sample $k = 50$ is given a relative weight of one in both cases. For the case where $\lambda = 0.99$, the earlier sample of $k = 40$ is given a relative weight of 0.90. The sample taken fifty samples earlier at $k = 1$ still carries a weight of 0.60. On the other hand, for the case where $\lambda = 0.95$, the data at $k = 1$ will be weighted by a factor of 0.08, making its effect on the estimate almost negligible.

Table 11. Effect of relative weighting of the forgetting factor λ

k	1	10	20	30	40	47	48	49	50
$\lambda = 0.99$	0.61	0.67	0.73	0.82	0.90	0.97	0.98	0.99	1
$\lambda = 0.95$	0.08	0.13	0.21	0.35	0.60	0.85	0.90	0.95	1

A recursive least squares algorithm has now been developed that is suitable for estimating parameters of linear time-varying (LTV) systems. The algorithm in its present form, however, may be subject to numerical difficulties due to model mismatching, noise and computer roundoff. In the next section, a more robust estimation approach will be developed that exhibits better stability than the standard RLS algorithm without imposing a penalty on the number of computations performed.

U-D Cofactorization

It has been demonstrated (see Ljung and Soderstrom, 1983) that if the input to the estimator $\{u(t)\}$ is not persistently exciting, the matrix $\mathbf{P}(t)$, which is referred to as the *covariance* matrix, may become singular causing a failure of the RLS algorithm. Additionally, it is well established that computer inaccuracies, primarily due to roundoff,

can also lead to numerical difficulties in the computation of the Kalman filter and likewise, the RLS algorithm. One possible solution to the problem is to ensure that the measurements and the dynamic model are sufficiently noisy to prevent near singularities from occurring. This approach, however, could lead to inaccuracies in the estimation process as the true dynamic characteristics of the system will be masked by the noise. Another possible solution is to perform some of the computations using extra-precision arithmetic. This option, however, requires additional memory elements and may suffer from performance degradation due to the additional computational effort required. A third, and undoubtedly more attractive alternative is to replace the RLS algorithm with one that is numerically better conditioned. The *square root algorithm* presents such an alternative. Bierman (1977) contends that methods involving square root algorithms have numerical properties that are superior to the alternative methods. The use of square root matrices preserves symmetry and assures non-negative eigenvalues for the covariance matrix. Square root algorithms also effectively reduce the dynamic range of the numbers used in the computations of the covariance matrix. Roughly speaking, computations in the range of 10^{-N} to 10^{N} are reduced to the range of $10^{-N/2}$ to $10^{N/2}$. This essentially halves the word length requirements for computing the covariance matrix. Bierman (1977) presents a method whereby the covariance matrix $\mathbf{P}(t)$ is factored in the form:

$$\mathbf{P}(t) = \mathbf{U}(t)\mathbf{D}(t)\mathbf{U}^{T}(t),$$

where $\mathbf{U}(t)$ is an upper triangular matrix with ones on the diagonal and $\mathbf{D}(t)$ is a diagonal matrix. The recursions are therefore performed in $\mathbf{U}(t)$and $\mathbf{D}(t)$ as opposed to $\mathbf{P}(t)$. The use of triangular matrices involves fewer computations and the factorization process avoids the necessity of computing time-consuming square roots. According to Bierman (1977), using factorization guarantees that $\mathbf{P}(t)$ will remain positive definite. The following development follows Bierman (1977):

Assume α_t in equation (4.10c) is equal to one. Then:

$$\mathbf{P}(t) = \frac{\mathbf{P}(t-1)}{\lambda} - \frac{\mathbf{P}(t-1)\varphi(t)\varphi^T(t)\mathbf{P}(t-1)}{\lambda + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)} \cdot \frac{1}{\lambda} . \tag{4.11}$$

Also, assume $\mathbf{P}(t-1)$ is factored as $\mathbf{U}(t-1)\mathbf{D}(t-1)\mathbf{U}^T(t-1)$ where $\mathbf{U}(t)$ is upper triangular with all diagonal elements equal to one, and $\mathbf{D}(t)$ is a diagonal matrix. Let:

$$f(t) = \mathbf{U}^T(t-1)\varphi(t)$$

$$g(t) = \mathbf{D}(t-1)f(t)$$

$$\beta(t) = \lambda + \varphi^T(t)\mathbf{P}(t-1)\varphi(t) = \lambda + f^T(t)g(t).$$

Substituting into equation (4.11) gives:

$$\mathbf{U}(t)\mathbf{D}(t)\mathbf{U}^T(t) = \left[\mathbf{U}(t-1)\mathbf{D}(t-1)\mathbf{U}^T(t-1) - \frac{\mathbf{U}(t-1)g(t)g^T(t)\mathbf{U}(t-1)}{\beta(t)}\right] \cdot \frac{1}{\lambda} \tag{4.12}$$

$$\mathbf{U}(t)\mathbf{D}(t)\mathbf{U}^T(t) = \mathbf{U}(t-1)\left[\mathbf{D}(t-1) - \frac{g(t)g^T(t)}{\beta(t)}\right]\mathbf{U}^T(t-1)\Big/\lambda$$

Now let:

$$\mathbf{D}(t-1) - \frac{g(t)g^T(t)}{\beta(t)} = \overline{\mathbf{U}}(t)\overline{\mathbf{D}}(t)\overline{\mathbf{U}}^T(t) \tag{4.13}$$

where,

$$\mathbf{U}(t) = \mathbf{U}(t-1)\overline{\mathbf{U}}(t)$$

$$\mathbf{D}(t) = \overline{\mathbf{D}}(t)\Big/\lambda$$

Now it remains to find the factorization of equation (4.13). Let:

$$\overline{\mathbf{U}}(t) = \begin{pmatrix} 1 & \overline{U}_{1,2} & \cdots & \overline{U}_{1,d} \\ & 1 & & \vdots \\ & & \ddots & \overline{U}_{d-1,d} \\ 0 & & & 1 \end{pmatrix}$$

where $d = \dim \theta$. And let:

$$\overline{\mathbf{D}}(t) = \begin{pmatrix} \overline{D}_1 & & 0 \\ & \ddots & \\ 0 & & \overline{D}_d \end{pmatrix}$$

$$\mathbf{D}(t-1) = \begin{pmatrix} D_1 & & 0 \\ & \ddots & \\ 0 & & D_d \end{pmatrix}$$

and let e_i be the ith unit vector. Then equation (4.13) can be expressed as:

$$\sum_{i=1}^{d} \overline{U}_i \overline{U}_i^T \overline{D}_i = \sum_{i=1}^{d} D_i e_i e_i^T - \frac{1}{\beta} g g^T .$$

Now let f_i be equal to the ith component of f. Then:

$$\beta_d = \beta = \lambda + \sum_{i=1}^{d} f_i g_i, \quad c_d = \frac{1}{\beta}, \quad V_d = g .$$

Then (4.13) can be rewritten as:

$$\sum_{i=1}^{d} \overline{D}_i \overline{U}_i \overline{U}_i^T = \sum_{i=1}^{d} D_i e_i e_i^T - \frac{1}{\beta_d} V_d V_d^T \quad . \tag{4.14}$$

Now $\overline{D}_i$ and $\overline{U}_i$ can be determined from equation (4.14) given β, D_i and V_d. Consider the matrix:

$$M_d = \overline{D}_d \overline{U}_d \overline{U}_d^T - D_d e_d e_d^T + \frac{1}{\beta_d} V_d V_d^T .$$

Denoting $V_{d,i}$ as the ith component of the column vector V_d, by choosing:

$$\overline{D}_d = D_d - \frac{V_{d,d}^2}{\beta_d}$$

$$\overline{U}_{d,d} = 1$$

$$\overline{U}_{i,d} = -\frac{V_{d,d}}{\overline{D}_d \beta_d} V_d; \quad i = 1, \cdots, d-1,$$

the last column of M_d will be made equal to zero. With:

$$V_{d-1} = \begin{pmatrix} V_{d,1} \\ \vdots \\ V_{d,d-1} \\ 0 \end{pmatrix},$$

M_d can then be written as:

$$M_d = \left(\frac{V_{d.d}^2}{\overline{D}_d \beta_d^2} + \frac{1}{\beta_d} \right) V_{d-1} V_{d-1}^T.$$

Now if the kth value of β is determined to be:

$$\beta_k = \lambda + \sum_{j=1}^{k} f_i g_i$$

then equation (4.12) can be used to determine:

$$\frac{V_{d,d}^2}{\overline{D}_d \beta_d^2} + \frac{1}{\beta_d} = \frac{1}{\beta_{d-1}}$$

and

$$\overline{D}_d = D_d \beta_{d-1} / \beta_d, \quad \overline{U}_{i,d} = -\left(f_d / \beta_{d-1} \right) g_i \quad i = 1, \ldots, d-1. \tag{4.15}$$

Now returning to equation (4.14):

$$\sum_{i=1}^{d-1} \overline{D}_i \overline{U}_i \overline{U}_i^T = \sum_{i=1}^{d-1} D_i e_i e_i^T - M_d = \sum_{i=1}^{d-1} D_i e_i e_i^T - \frac{1}{\beta_{d-1}} V_{d-1} V_{d-1}^T \tag{4.16}$$

provided that $\overline{U}_d$ and $\overline{D}_d$ are chosen according to equation (4.15). Notice that equation (4.16) is exactly the same form as equation (4.14) except that d been decreased to $d-1$. The same procedure can therefore be used to find $\overline{D}_{d-1}, \overline{U}_{d-1}$, etc. The algorithm to find $\mathbf{U}(t)$ and $\mathbf{D}(t)$ can then be used to determine:

$$\mathbf{L}(t) = \frac{\mathbf{U}(t-1)\mathbf{D}(t-1)\mathbf{U}^T(t-1)\varphi(t)}{\beta(t)}$$

$$= \frac{\mathbf{U}(t-1)g(t)}{\beta(t)}.$$

Ljung and Soderstrom (1983) summarize Bierman's complete U-D Cofactorization Algorithm as follows:

- Initialize $\mathbf{U}(0)$ and $\mathbf{D}(0)$ at time $t = 0$, $\mathbf{U}(0)\mathbf{D}(0)\mathbf{U}^T(0) = \mathbf{P}(0)$.
- At time t, compute $\mathbf{L}(t)$ and update $\mathbf{U}(t-1)$ and $\mathbf{D}(t-1)$ by performing steps 1 - 6.

1. Compute $f := \mathbf{U}^T(t-1)\varphi(t)$, $g := \mathbf{D}(t-1)f$, $\beta_0 = \lambda$.
2. For $j = 1,\ldots,d$ do steps 3 - 5.
3. Compute:

$$\beta_j := \beta_{j-1} + f_j g_j,$$

$$\mathbf{D}(t)_{jj} := {\beta_{j-1}\mathbf{D}(t-1)_{jj}}\Big/{\beta_j \lambda},$$

$$v_j := g_j,$$

$$u_j := {-f_j}\Big/{\beta_{j-1}}.$$

4. For $i = 1,\ldots,j-1$, do step 5 (If $j = 1$, skip step 5).
5. Compute:

$$\mathbf{U}(t)_{ij} := \mathbf{U}(t-1)_{ij} + v_i u_j,$$

$$v_i := v_i + \mathbf{U}(t-1)_{ij} v_j.$$

6. $$\bar{\mathbf{L}}(t) := \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix}; \quad \mathbf{L}(t) := {\bar{\mathbf{L}}(t)}\Big/{\beta_d}.$$

Bierman (1977) points out that the number of computations required for the U-D cofactorization algorithm is roughly the same as for the conventional Kalman filter. Yet, factorization of the covariance matrix $\mathbf{P}(t)$ greatly enhances the numerical stability of the estimation process. In addition, Bierman's U-D algorithm does not require square root extractions that are required in some other square root algorithms, making the U-D cofactorization more suitable for use in microprocessors or DSP chips. The use of a square root algorithm alone, however, does not make for robust parameter estimation. Several other issues need to be considered before the estimator can be practically implemented.

Data filtering

Generally speaking, real-world processes are always more complicated than the models used to describe them. Often, however, information about the process is available *a priori* for incorporation into a *prejudice model.* The purpose of the parameter estimator is to determine the parameters that best fit the prejudice model. Normally, disturbances are included as part of the prejudice model. The disturbances can either be described as stochastic, as in a random noise sequence, or as deterministic disturbances, such as levels, ramps, sinusoids, etc. Ljung (1987) gives some insight into the frequency domain interpretation of the least squares estimation problem as follows: Assuming an ARX model described as:

$$A(q)y(t) = B(q)u(t) + \varepsilon(t),$$

the least squares estimate can be expressed as:

$$\hat{\theta}_N = \arg\min_{\theta} \frac{1}{N}\sum_{t=1}^{N}\left(A(q)y(t) - B(q)u(t)\right)^2$$

which, as $N \to \infty$, becomes,

$$\theta^* = \arg\min_{\theta} \int_{-\pi}^{\pi} \left| G_p - \frac{B(q)}{A(q)} \right|^2 \cdot |A(q)|^2 \, d\omega \qquad (4.17)$$

where G_p represents the true plant. In equation (4.17), the magnitude of A is small at the poles, which are the frequencies of interest in the estimation process, but A becomes large at higher frequencies. In the presence of measurement noise, the least-squares estimator produces biased estimates that tend to be weighted toward the higher frequencies. The bandwidth of interest in control applications, however, is around the crossover frequency, not at higher frequencies. Low pass filtering of the input and output data is therefore typically required to improve the accuracy of the estimates around the crossover frequency.

Assume the process can be modeled as:

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{C(q)}{D(q)}\varepsilon(t) + \nu(t) \tag{4.18}$$

where $\varepsilon(t)$ is a sequence of independent, zero mean random variables (Gaussian white noise) and $\nu(t)$ is a deterministic signal of known form but unknown amplitude. If $\frac{C(q)}{D(q)} \neq 1$, the noise is no longer white and the elements of $\{\varepsilon(t)\}$ are no longer independent; therefore, the least-squares estimate will again be biased. If the prior estimates of $C(q)$ and $D(q)$ are known, then the regression vector $\varphi(t)$ can be filtered by $H_f = \frac{D(q)}{C(q)}$ and the regular RLS algorithm can still be used. If $C(q)$ and $D(q)$ are not known, a model such as in equation (4.18) can be assumed and an alternative estimation procedure, such as extended least-squares, can be implemented.

The presence of deterministic disturbances (represented by $\nu(t)$ in equation (4.18)) in the input-output data will also lead to biased estimates. Wittenmark (1988) discusses the need for an appropriate disturbance annihilation filter to eliminate deterministic disturbances. Offsets, drift and other low frequency disturbances can be dealt with by filtering the data through a high pass filter, whereas, sinusoidal disturbances can be filtered with an appropriate notch filter. In order to deal with high frequency noise, offsets and low frequency drift, Wittenmark (1988) recommends a filter of the form show in Figure 35.

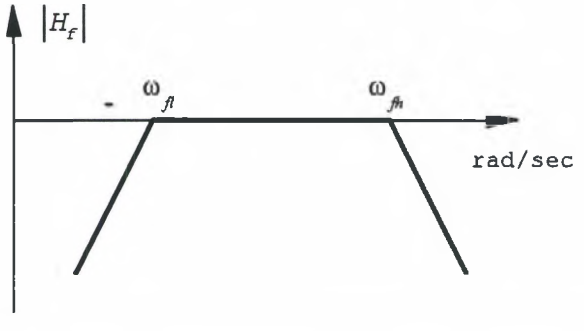

Figure 35. Magnitude response of disturbance annihilation filter $H_f(q)$

Astrom and Wittenmark (1989) recommend that ω_{fl} be at least one decade below the desired crossover frequency and that ω_{fh} be set at 2-10 times the crossover frequency. Of course, the higher the desired bandwidth of the closed-loop system, the larger the bandwidth of the filter needs to be. Astrom and Wittenmark (1989) recommend that $H_f(q)$ be of the form:

$$H_f(q) = \frac{(1-\alpha)(q-1)}{(q-\alpha)}$$

where $|\alpha| < 1$. (It is assumed that the controller will be designed to compensate for the disturbances filtered out of the data to the estimator.)

Unmodeled dynamics

As discussed previously, the prejudice model is an attempt to model the complex dynamics of the true process by a simple linear model. Not only will disturbances adversely influence the accuracy of the parameter estimates, unmodeled dynamics can lead to problems as well. Unmodeled dynamics can drive the parameter estimates to inaccurate or even unreasonable values that can cause poor or possibly catastrophic controller performance. Astrom and Wittenmark (1989) use averaging analysis to show that data filtering as described above can make the estimator less sensitive to unmodeled dynamics. The effect of unmodeled dynamics can also be reduced by increasing the complexity of the plant model. A general family of model sets can be defined as:

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}\varepsilon(t). \qquad (4.19)$$

The family of model sets of equation (4.19) allows for a rather complicated description of the plant; however, Ljung and Soderstrom (1983) point out that overparameterization of the model can lead to singularity problems. Singularity occurs in the model of equation (4.19) when:

- There is a factor common to all of A, B and C,
- B and F have a common factor,
- C and D have a common factor.

Ljung and Soderstrom (1983) show results of extensive simulations in the presence of colored and uncolored noise using 11 different model sets derived from the family of models described by equation (4.19). They have drawn several important conclusions from their work, three of which are summarized as follows:

1. Most of the model sets show very similar results, with a few exceptions.
2. There is no advantage to using the complex model set of equation (4.19). In fact, most of the other less complex models yield superior results.
3. If the signal-to-noise ratio is relatively high, the model set assuming $C = D = 1$ and either $F = 1$ or $A = 1$ (as in the case of the ARX model) is usually the best choice.

The third conclusion noted above is extremely important, as that assumption allows for the use of the ordinary RLS algorithm when the noise level is relatively low, as opposed to one of the more complicated algorithms, such as extended least-squares.

After the model set has been selected, the appropriate model order must also be selected. In an off-line identification problem, the model order can simply be increased until it proves to be of no advantage to do so further; however, the on-line problem presents a more difficult situation. Unless several estimation algorithms using different model orders are run in parallel, the model order must be selected ahead of time. When using a RLS algorithm in the presence of noise, a relatively high model order might be required to obtain satisfactory results. Ljung and Soderstrom (1983) point out that when the model polynomials are of a higher order than the minimal description of the plant at

the convergence point of θ, overparameterization may result. High model orders also tend to emphasize the influence of measurement noise in the estimation process, causing the model to be inaccurate at higher frequencies. Too high a model order can also lead to false local minima in the estimation process, resulting in biased estimates. If the controller design is constrained to a pole-placement method using a PID-like structure, the order of the estimator model will also necessarily be constrained. Assuming the input is persistently exciting, the performance of the estimator in the case of a reduced order estimator will depend on the level of the noise present and on how well the data to the estimator is prefiltered.

Estimator windup

The equation for updating the covariance matrix $\mathbf{P}(t)$ in the RLS algorithm was given in equation (4.11) as:

$$\mathbf{P}(t) = \frac{\mathbf{P}(t-1)}{\lambda} - \frac{\mathbf{P}(t-1)\varphi(t)\varphi^T(t)\mathbf{P}(t-1)}{\lambda + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)} \cdot \frac{1}{\lambda}.$$

The potential of $\mathbf{P}(t)$ becoming singular or near singular has already been discussed. $\mathbf{P}(t)$ can become singular due to:

- Lack of persistence of excitation on the input signal,
- Overparameterization of the plant model,
- Computer roundoff errors.

The risk of singularity due to computer round off can be virtually eliminated by using a square root factorization method, as per Bierman (1977). Overparameterization can be controlled by ensuring that a model set is selected that is not overly complex (e.g., an

ARX model) and that a model order is chosen that adequately approximates the dynamics of the plant. Unless an external perturbation signal is applied, however, it is difficult to guarantee that the input to the plant (and the estimator) will remain general enough so that the regression vector is persistently spanning (i.e., spans the entire vector space). The problem becomes especially difficult when the estimator is operating on a plant in a closed-loop configuration and the controller drives the output of the plant to equilibrium. The update equation for $\mathbf{P}(t)$ was given in equation (4.10c) as:

$$\mathbf{P}(t) = \frac{1}{\lambda}\left[{}^{1}\!/_{\alpha_t} - \mathbf{L}(t)\varphi^T(t) \right]\mathbf{P}(t-1) = \frac{\mathbf{P}(t-1)}{\lambda\alpha_t} - \frac{\mathbf{P}(t-1)\mathbf{L}(t)\varphi^T(t)}{\lambda} \tag{4.20}$$

Since $\mathbf{P}(t)$ contains the covariances of the regression vector $\varphi(t)$, as $\varphi(t)$ tends to contain no new information, the product $\mathbf{P}(t-1)\varphi^T(t) \to 0$. Assuming $\alpha_t = 1$, if no new information is contained in the measurements, equation (4.20) reduces to

$$\frac{\mathbf{P}(t-1)}{\lambda}. \tag{4.21}$$

Equation (4.21), and thus $\mathbf{P}(t)$, will tend toward infinity at a rate of ${}^{1}\!/_{\lambda}$. In an adaptive control application, in order to ensure stability of the controller, $\mathbf{P}(t)$ must be guaranteed to remain bounded and positive definite at all times.

One way of ensuring that $\mathbf{P}(t)$ remains bounded and positive definite is to constantly add a positive definite matrix to $\mathbf{P}(t)$. This is known as the *Levenberg-Marquardt regularization method.* Ljung and Soderstrom (1983) propose a similar concept for the U-D factorization algorithm by limiting the elements of the matrices $\mathbf{U}(t)$ and $\mathbf{D}(t)$. Unlike Levenberg-Marquardt, however, the modification is only applied when an eigenvalue is tending to zero. Ljung and Soderstrom (1983) show that since $\det \mathbf{U}(t) = 1$, the elements of $\mathbf{U}(t)$ will always remain bounded; therefore, limits need only be imposed on $\mathbf{D}(t)$. The second calculation in step 3 of the U-D cofactorization algorithm given previously is thus changed to:

$$D(t)_{jj} := \min\left(C, \frac{\beta_{j-1} D(t-1)_{jj}}{\beta_j \lambda} \right) \tag{4.22}$$

where C is a positive number that bounds the elements of $\mathbf{D}(t)$. Providing the input sequence is well behaved, the upper bound of C in equation (4.22) is never reached. If an element of $\mathbf{D}(t)$ becomes too large, it is bounded by C. Limiting the elements of $\mathbf{D}(t)$ ensures that estimator does not fail due to singularity problems if the input is not persistently exciting, as will be the case in an adaptive control problem when the reference input and the plant output have reached steady-state and the noise level is insufficient to adequately excite the system.

Although the modification given in equation (4.22) prevents the estimation algorithm from failing by limiting the size of the elements of $\mathbf{D}(t)$, difficulties may be encountered if the covariances are allowed to remain large over time. The covariances grow large when the system is not persistently excited. When a disturbance does finally arrive after the estimator gains are large, the estimates can move very rapidly, causing the controller to behave poorly during that period. Also, if the covariances remain large, the algorithm becomes excessively sensitive to noise. It is therefore advantageous to automatically *reset* the covariances to some smaller values (e.g., $\mathbf{U}(0)$ and $\mathbf{D}(0)$) once they have reached the bound of C given in the modification of (4.22).

Several other approaches for dealing with estimator windup have also been successfully implemented. Some of the most common are:

- time-variable forgetting factors,
- constant trace algorithms,
- directional forgetting,
- conditioning techniques.

Estimator windup is only one of several difficulties that can be encountered when attempting parameter estimation of a plant operating in a closed-loop system. The criteria

for being able to identify a system in closed-loop and some of the other difficulties that can be encountered when attempting closed-loop identification are discussed in the next section.

4.3 Identification in Closed-Loop

Although the basic RLS algorithm has been made more robust by the modifications discussed above, the success of the algorithm depends to a large degree on the generality of the input signal. If the input ceases to be persistently exciting, an ill-conditioned covariance matrix results and the algorithm will fail to produce an accurate model of the plant. Problems can also be encountered when the identification is performed on a plant under closed-loop control. Feedback of sufficiently low order will introduce linear dependencies among the elements of the regression vector $\varphi(t)$ which means that a unique solution to the least-squares algorithm will not exist, making the system unidentifiable. This problem disappears if the feedback is of sufficiently high order, or if a time-varying controller gain is used. Isermann (1982) develops the conditions for identifiability in a closed-loop as follows:

Assume a plant is described by the model:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})} z^{-d} u(t) + \frac{D(q^{-1})}{A(q^{-1})} \varepsilon(t)$$

where:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_m q^{-m}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_m q^{-m}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + \cdots + d_m q^{-m}.$$

And let:

$$G_p(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} z^{-d} \quad \text{and} \quad G_\varepsilon(q^{-1}) = \frac{D(q^{-1})}{A(q^{-1})}.$$

Assume that the plant is placed in a feedback loop with a controller described by the transfer function:

$$G_c(q^{-1}) = \frac{u(t)}{e(t)} = \frac{S(q^{-1})}{R(q^{-1})} = \frac{s_0 + s_1 q^{-1} + \cdots + s_\nu q^{-\nu}}{r_0 + r_1 q^{-1} + \cdots + r_\mu q^{-\mu}}$$

where the error is defined as $e(t) = w(t) - y(t)$ and $w(t)$ is the reference input into the controller. Also assume that the only excitation to the plant is unmeasurable stochastic noise, $\varepsilon(t)$. The non-recursive least-squares algorithm is based on the equation:

$$y(t) = \theta^T \varphi(t) + \varepsilon(t) \tag{4.23}$$

where:

$$\varphi^T(t) = \left[-y(t-1) \ldots -y(t-m)\ u(t-d-1) \ldots u(t-d-m)\right]. \tag{4.24}$$

Because of the feedback, if $w(t) = 0$, the input and output signals are related by:

$$\begin{aligned} u(t-d-1) = &-r_1 u(t-d-2) - \cdots - r_\mu u(t-\mu-d-1) \\ &- s_0 y(t-d-1) - \cdots - s_\nu y(t-\nu-d-1) \end{aligned} \tag{4.25}$$

The element $u(t-d-1)$ from the regression vector $\varphi^T(t)$ in equation (4.24) is shown in equation (4.25) to be linearly dependent on the other elements of $\varphi^T(t)$ if $\mu \le m-1$ and $\nu \le m-d-1$. There is therefore no unique solution for the parameter vector, θ; however, if $\mu \ge m$ or $\nu \ge m-d$, the linear dependency is removed and the system will be identifiable. The relationship between the output signal and the noise input can be determined to be:

$$\frac{y(t)}{\varepsilon(t)} = \frac{D(q^{-1})R(q^{-1})}{A(q^{-1})P(q^{-1}) + B(q^{-1})q^{-d}S(q^{-1})} = G_{id}(q^{-1})R(q^{-1}).$$

The polynomials $R(q^{-1})$ and $S(q^{-1})$ are assumed to be known and relatively prime. If the transfer function $G_{id}(q^{-1})$ contains p common poles and zeros, then in order for the system to be identifiable in closed-loop, the order of the controller must be:

$$\max\{\mu\,;\, \nu+d\}, \qquad -p \ge m. \tag{4.26}$$

According to Isermann (1982), if the condition given in equation (4.26) is not met, identifiability can still be obtained by either applying a controller with a time-varying gain, or by applying an extra perturbation signal that is persistently exciting of order m to the closed-loop, but not between the measured signals $u(t)$ and $y(t)$. In addition to the above criteria, Isermann (1982) also shows that in order for the plant to be identifiable in closed-loop, the order m and the dead time d of the process must be known a priori.

Even if the above criteria are met, other difficulties may arise when attempting to identify a plant under closed-loop control. The RLS algorithm given in equation (4.10b) can be expressed as:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{\mathbf{P}(t-1)\varphi(t)}{\lambda/\alpha_t + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)}\left[y(t) - \hat{\theta}^T(t-1)\varphi(t)\right] \tag{4.27}$$

Substituting equation (4.23) into equation (4.27) yields:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{\mathbf{P}(t-1)\varphi(t)\varepsilon(t)}{\lambda/\alpha_t + \varphi^T(t)\mathbf{P}(t-1)\varphi(t)} \tag{4.28}$$

The term on the right hand side of equation (4.28) represents the change in the parameter $\hat{\theta}(t)$ from the last sampling interval. If $\varphi(t)$ and $\varepsilon(t)$ are correlated, as will normally be the case when the plant is under closed-loop control, the term $\varphi(t)\varepsilon(t)$ in (4.28) will cause $\hat{\theta}(t)$ to drift. The problem becomes even more significant when the input, and consequently $\varphi(t)$, is not persistently exciting. Assuming the reference input is constant, when the estimated parameters converge to some reasonable values, the control law will force the system into a steady-state condition. The input to the plant, and thus the regression vector $\varphi(t)$, will cease to be persistently exciting and the parameters will drift. Eventually, the parameter estimates become poor enough that the control law no longer behaves satisfactorily, causing instability in the output. The system is excited by the variation in the output causing the parameter estimates once again to converge to

reasonable values and the process repeats itself. This phenomenon is known as *bursting*. (The same problem may also be experienced in the case of an undermodeled plant.) According to Middleton and Goodwin (1990), the best way to deal with bursting that is currently known is to incorporate a *dead-zone* into the algorithm that stops the parameter vector and covariance matrix update when the parameter error reaches a threshold value. The switching threshold of the dead-zone is established based on the expected level of the measurement noise that manifests itself in the parameter error. Not only will the dead-zone prevent parameter drift when the input ceases to be persistently exciting, but it also helps to minimize estimator windup. Of course, the addition of the dead-zone into the algorithm introduces another application specific parameter into the estimation process. Rey, *et. al.* (1990) point out that in order for a dead-zone to be effective in the avoidance of bursting, it must be "properly tuned"; i.e., the parameter error threshold must be properly adjusted. They further state that the current level of understanding about the behavior of dead-zones is not even sufficient to assertion in an actual implementation whether the dead-zone is properly tuned or not. There is also no way to predict from the data when a system is going to burst. Although it is an *ad hoc* approach to the problem, at this time, the use of the dead-zone appears to be one of the better alternatives to control bursting.

The first major component of the self-tuning controller, the parameter estimator, has now been developed. Several important modifications have been proposed to make the basic RLS algorithm more robust. The other major component in the outer loop of the adaptive controller is the control law design mechanism. The underlying design principal and the development of a control law design mechanism suitable for an adaptive PID application are presented in the following section.

4.4 The Adaptive Control Law Design Mechanism

One of the attractive features of the self-tuning regulator structure is the flexibility gained in the selection of a control law design mechanism. A number of different approaches to the development of an adaptive control law design mechanism for self-tuning regulators have been proposed in the literature. Kalman's (1958) indirect self-tuning controller used a least-squares estimator in conjunction with a deadbeat controller. Astrom and Wittenmark's (1973) original self-tuning regulator was designed for a minimum variance control problem, and a self-tuning regulator using a generalized minimum variance controller was proposed by Clarke and Gawthrop (1975). A linear quadratic Gaussian version of the self-tuning regulator was presented by Peterka and Astrom (1973). Some of the earliest work in self-tuning regulators based on pole-placement algorithms was reported by Wellstead (1978). Wellstead's work primarily dealt with the regulator case, whereas, Astrom and Wittenmark's (1980) work on pole-placement algorithms focused strictly on the servo-mechanism case. If the controller design is to be constrained to fit a PID-like structure, the number of options for the choice of a control law design mechanism is significantly more limited.

The vast majority of self-tuning regulators in the literature that adhere to a PID controller model are based on some type of pole-placement technique, although there are a few exceptions (see Radke and Isermann, 1984). Warwick, Karam and Tham (1987) proposed a self-tuning control strategy with the goals of:

1. Computational efficiency
2. Robustness
3. Versatility
4. Good steady-state servo tracking.

They refer to their controller algorithm as a *simple self-tuning controller* (SSTC) due to the reduced complexity of the direct adaptive control strategy employed compared to some other pole-placement techniques requiring many more computations. The SSTC is versatile in that it is not restricted to a plant model of any particular order. In addition, the basic controller can be modified to accommodate control laws of different structures, including pole-placement and deadbeat control strategies. The controller is also shown to be robust, in that it can be applied to processes with variable time delays and to non-minimum phase systems. They also demonstrate how the controller can perform well in the presence of random disturbances. The basic control strategy is that of a deadbeat controller that attempts to cancel the process poles with the controller zeros. An option that does not cancel the process zeros is also presented, which allows the controller to be applied to non-minimum phase systems. The SSTC is developed following Warwick, Karam and Tham (1987).

Assume the plant is defined by the model:

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + C(q^{-1})\varepsilon(t) \tag{4.29}$$

where:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_n q^{-n}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \cdots + c_n q^{-n}$$

and $\varepsilon(t)$ is a sequence of zero mean random inputs that are uncorrelated with the input and output signals, $u(t)$ and $\bar{y}(t)$. The integer d is the time delay of the plant expressed as an integer multiple of the sampling time. Also, assume that $d \geq 1$ such that $b_0 \neq 0$, the roots of $A(q^{-1})$ lie in the unit circle of the z-plane and $B(1) \neq 0$.

Defining $w(t)$ as the reference input and s as a scalar feed forward term, the error $e(t)$ is defined as:

$$e(t) = sw(t) - y(t).$$

An error controller is defined as the transfer function:

$$\frac{u(t)}{e(t)} = \frac{G(q^{-1})}{F(q^{-1})} \tag{4.30}$$

where $F(q^{-1})$ is monic, in the same form as $A(q^{-1})$ of equation (4.29).

The characteristic equation of the closed-loop system is determined to be:

$$A(q^{-1})F(q^{-1}) + z^{-d}B(q^{-1})G(q^{-1}) = 0.$$

The desired closed-loop performance is achieved by the proper selection of the polynomials, $F(q^{-1})$ and $G(q^{-1})$. A simple form of the controller that follows a deadbeat control strategy is obtained by selecting:

$$G(q^{-1}) = A(q^{-1})$$

and

$$F(q^{-1}) = 1 - z^{-d}B(q^{-1}).$$

The control signal is obtained from equation 4.30 as:

$$u(t) = A(q^{-1})e(t) + B(q^{-1})u(t-d). \tag{4.31}$$

In this case, the coefficients of the plant polynomials $A(q^{-1})$ and $B(q^{-1})$ make up the parameters of the controller. Substituting (4.31) into (4.29) yields:

$$y(t) = z^{-d}B(q^{-1})sw(t) + \frac{F(q^{-1})C(q^{-1})\varepsilon(t)}{G(q^{-1})}. \tag{4.32}$$

To achieve zero steady-state error, s must be selected as:

$$s = {}^{1}\!/_{B(1)}.$$

As stated in the objectives, one of the primary goals of this controller is good steady-state servo tracking. Assuming the error term, $\varepsilon(t)$, is made up of two components, $\varepsilon'(t)$, a zero mean white noise sequence, and v, which is a d.c. bias, the disturbance can be redefined as:

$$\varepsilon(t) = \varepsilon'(t) + v.$$

The tracking error of the controller is therefore expressed as:

$$\text{tracking error} = \frac{F(1)C(1)\varepsilon(t)}{G(1)}.$$

The expected value of the tracking error is then determined to be:

$$E\{\text{tracking error}\} = \frac{F(1)C(1)\nu}{G(1)} . \tag{4.33}$$

Equation (4.33) shows the inability of the controller to achieve zero steady-state error in the presence of a d.c. bias. However, by changing the definitions of the controller polynomials to:

$$G(q^{-1}) = A(q^{-1})$$

$$F(q^{-1}) = B(1) - q^{-d}B(q^{-1}),$$

the control output can be recalculated to be:

$$u(t) = \frac{A(q^{-1})e(t)}{B(1)} + \frac{B(q^{-1})u(t-d)}{B(1)} \tag{4.34}$$

and the closed-loop output is determined to be:

$$y(t) = \frac{q^{-d}B(q^{-1})sw(t)}{B(1)} + \frac{\left[B(1) - q^{-d}B(q^{-1})\right]C(q^{-1})\varepsilon(t)}{G(q^{-1})}. \tag{4.35}$$

Now, steady-state tracking can be obtained by setting $s = 1$. Since:

$$\left[B(1) - q^{-d}B(q^{-1})\right] = 0$$

in the steady-state, the effect of the d.c. bias ν is eliminated. Notice that in equation (4.35), the process zeros are not canceled. Therefore, when controlling non-minimum phase systems, unstable poles will not be introduced into the closed-loop equation.

The control law just developed can be extended to models of any order. The method can also be modified to conform to a PID-like structure by imposing certain limitations on the algorithm. Assume a classical PID controller is converted to discrete-time using a backward difference giving:

$$u(t) = \left[K_p + \frac{K_i}{(1-q^{-1})} + K_d(1-q^{-1})\right]e(t). \tag{4.36}$$

Equation (4.36) can be rewritten as:

$$(1-q^{-1})u(t) = K(q^{-1})\left[w(t) - y(t)\right]$$

where

$$K(q^{-1}) = k_1 + k_2 q^{-1} + k_3 q^{-2},$$
$$k_1 = K_p + K_i + K_d,$$
$$k_2 = -\left(K_p + 2K_d\right),$$
$$k_3 = K_d.$$

Equation (4.34) can be rearranged as:

$$\left[B(1) - q^{-d}B(q^{-1})\right]u(t) = A(q^{-1})\left[sw(t) - y(t)\right]. \tag{4.37}$$

In order for equation (4.37) to take on the form of equation (4.36), the following conditions must be true:

$$s = 1$$
$$K(q^{-1}) = A(z^{-1})$$
$$d = 1$$
$$B(1) = B(q^{-1}) = b_0.$$

By restricting the assumed plant model to second-order with no zeros and by forcing the time delay of the plant to be equal to one (a single sampling interval), the SSTC can be forced to exhibit a PID-like structure. The resultant control output is determined to be:

$$u(t) = u(t-1) + \frac{\left(1 + a_1 q^{-1} + a_2 q^{-2}\right)\left[w(t) - y(t)\right]}{b_0}. \tag{4.38}$$

The controller described by equation (4.38) has the advantage of being a *direct* implementation, as the controller parameters are obtained directly from the plant model parameters without requiring any intermediate calculations. The method is somewhat limited, however, by the deadbeat-like control design strategy that cancels the poles of the plant. This restricts the ability of the designer to control the dynamics of the closed-loop system by arbitrary placement of the closed-loop system poles. Pole-placement methods offer an alternative control law design method that allows for control of the dynamics of the closed-loop system.

McInnis, *et.al.* (1985) proposed a pole-placement PID algorithm for controlling left ventricular bypass lift devices. The controller design is based on a *setpoint-on-I-only* PID structure that incorporates filtering of the derivative term. By also adding the same filter to the integral term, the two extra closed-loop zeros are forced to be located at the origin in the z-plane, simplifying the calculations. The resulting controller structure is given in equation (4.39).

$$u(t) = \left[-K_p - \frac{K_d(1-q^{-1})}{1+\kappa q^{-1}} \right] y(t) + \frac{K_i}{(1-q^{-1})(1+\kappa q^{-1})} [w(t) - y(t)] \qquad (4.39)$$

The control law give in equation (4.39) can be formulated as:

$$P(q^{-1})u(t) = Lw(t) - S(q^{-1})y(t) \qquad (4.40)$$

where

$$P(q^{-1}) = (1-q^{-1})(1+\kappa q^{-1}),$$

$$S(q^{-1}) = s_0 + s_1 q^{-1} + s_2 q^{-2},$$

$$L = S(1) = K_i,$$

and

$$s_0 = K_p + K_i + K_d,$$

$$s_1 = K_p(\kappa - 1) - 2K_d,$$

$$s_2 = K_d - \kappa K_p.$$

If the plant model is assumed to be:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \varepsilon(t)$$

then the closed-loop relationship is determined from equation (4.40) to be:

$$\left[A(q^{-1})P(q^{-1}) + B(q^{-1})S(q^{-1}) \right] y(t) = L \cdot B(q^{-1})w(t) + P(q^{-1})\varepsilon(t). \qquad (4.41)$$

If the plant model is forced to be second-order, the relationship of equation (4.41) shows that four closed-loop poles of the desired model, $A_m(q^{-1})$, can be arbitrarily positioned, assuming that:

- the plant model is second-order,
- $A(q^{-1})$ and $B(q^{-1})$ are coprime,
- $B(q^{-1})$ does not have the zero $q = +1$.

By equating coefficients of $A_m(q^{-1})$ and $A(q^{-1})P(q^{-1}) + B(q^{-1})S(q^{-1})$, the parameters s_0, s_1, and s_2 can be determined assuming the parameters of the plant model are known. By applying the certainty equivalence principle, the estimated plant parameters are then substituted into the relationship of equation (4.41) in place of the actual plant parameters. Let the second-order plant model be defined as:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2}$$

$$B(q^{-1}) = b_0 q^{-1} + b_1 q^{-2}$$

and let the desired closed-loop polynomial be defined as:

$$A_m(q^{-1}) = a_{m0} + a_{m1} q^{-1} + a_{m2} q^{-2} + a_{m3} q^{-3} + a_{m4} q^{-4}.$$

Then,

$$A_m(q^{-1}) = A(q^{-1})P(q^{-1}) + B(q^{-1})S(q^{-1})$$

which leads to:

$$s_0 = \frac{a_{m1} - (\kappa - 1) - a_1}{b_0} \tag{4.42a}$$

$$s_1 = \frac{b_0(a_{m2} + \kappa - a_1(\kappa - 1) - a_2) - b_1(a_{m1} - (\kappa - 1) - a_1)}{b_0 b_1} \tag{4.42b}$$

$$s_2 = \frac{a_{m4} + a_2 \kappa}{b_1}. \tag{4.42c}$$

Now the parameters of equation (4.42) can be substituted into equation (4.40) to give the control law:

$$u(t) = (1 - \kappa)u(t-1) + \kappa u(t-2) + K_i w(t) - s_0 y(t) - s_1 y(t-1) - s_2 y(t-2). \tag{4.43}$$

Thus, by specifying the desired closed-loop dynamics as the four pole locations of $A_m(q^{-1})$, and by substituting the estimated plant parameters into equation (4.42), the control output given in equation (4.43) can be determined. The on-line control law design

mechanism in this case is explicit, or indirect, as the control law design requires an intermediate calculation between the estimation of the parameters and the control law design.

4.5 Stability and Convergence of Self-Tuning Regulators

The fact that adaptive control systems are non-linear makes stability analysis a nontrivial task. Nonlinear systems can be analyzed for Lyapunov stability, i.e., stability of a particular solution, but establishing global stability is difficult. According to Astrom and Wittenmark (1984b), some stability and convergence proofs for simple algorithms under ideal conditions were proposed as early as 1979. But they also state that there have not been any proofs based on more realistic assumptions. Astrom and Wittenmark (1989) did prove that for a direct adaptive pole assignment algorithm, given a number of critical assumptions, the estimates are bounded and the normalized prediction error converges to zero. They did not prove, however, that the parameter estimates converge. Astrom and Wittenmark (1989) point out that parameter convergence is not necessary for error convergence in direct algorithms. This is not the case, however, in indirect schemes where parameter convergence is essential to acceptable controller performance.

According to Kumar (1990), "...very little is known regarding the behavior of recursive least-squares parameter estimate based adaptive control schemes. For example, whether the original minimum variance self-tuning regulators of Peterka and Astrom and Wittenmark actually self-tune has been an open question for more than 15 years. Also, no conclusive results are available for certainty equivalent control laws which are of pole-zero placement type, or based on LQG design, etc." In what are considered two very important papers, Sternby (1977) and Rootzen and Sternby (1984) introduced a procedure known as "Bayesian embedding," which proves that for systems excited by white,

Gaussian noise of a sufficient magnitude, the parameter estimates generally converge. Kumar (1990) expands on the work of Rootzen and Sternby to prove the stability and convergence of least-squares based adaptive control schemes under the condition that the additive noise to the system is white and Gaussian, and that the true system is strictly minimum phase. Although the proofs are too involved to be included here, the results of Kumar's work are summarized as follows:

Theorem 1: *The parameter estimates, and thus the adaptive control law, both converge asymptotically stable.*

Theorem 1 is based on two important conditions. First, the additive noise entering the system is required to be white, Gaussian noise. Second, the convergence result may not be valid on an exceptional set of true parameter vectors of Lebesgue measure zero.

Theorem 2: *The overall adaptively controlled system is stable in an averaged squared sense whenever the estimated parameters are used in a certainty-equivalent fashion to design a control law which is stable for the estimated parameters, and the true system is of minimum phase.*

Kumar goes on to give convergence and stability proofs for specific cases based on Theorem 1 and Theorem 2. But given the above constraints, the convergence of the estimated parameters and the stability of the overall adaptive control system can be guaranteed.

4.6 Chapter Summary

In this chapter, the self-tuning regulator has been presented as a solution to an adaptive control problem. The basic structure of the self-tuning regulator consists of two loops: an inner loop consisting of a regular feedback controller and an outer loop made up of a parameter estimator and a control law design mechanism. An on-line parameter estimation algorithm has been developed based on recursive least-squares. The basic RLS algorithm has been modified to be able to track time-varying parameters by the addition of an exponential forgetting factor. Also, U-D cofactorization of the covariance matrix has been incorporated to improve the numerical stability of the RLS algorithm. Some of the practical implementation issues of parameter estimation, such as data filtering and estimator windup, have also been presented. The covariance matrix $\mathbf{P}(t)$ is bounded in the event the input ceases to be persistently exciting, and covariance resetting of $\mathbf{P}(t)$ is also employed to keep the algorithm from becoming overly sensitive to noise and to keep the estimator gains from becoming excessively large.

Two different options for the control law design mechanism have also been presented. The first option, proposed by Warwick, Karam and Tham (1985), is referred to as a *simple self-tuning controller*. The SSTC is based on a control strategy that seeks to cancel the process poles without canceling the process zeros. The SSTC is a direct structure that is computationally efficient, but is limited in that the closed-loop system poles cannot be arbitrarily positioned. The basic idea, however, can be expanded to include a pole-placement scheme and it can be used in a PID-like controller structure by limiting the model order to two. The second option is a pole-placement method as proposed by McInnis, *et.al.* (1985). Although it is an indirect structure and requires more computations than the SSTC to implement, it allows for the closed-loop system poles to be chosen arbitrarily. The pole-placement method also very easily conforms to a PID

structure. It can, in fact, be adapted to PID structures of different types. The convergence and stability of the adaptive controller was also established for certain limited conditions based on the work of Kumar (1990).

CHAPTER V

SIMULATION OF THE ADAPTIVE PID CONTROLLER

5.1 Introduction

An adaptive PID controller has been developed based on the self-tuning regulator (STR) model. A recursive least-squares parameter estimator identifies the parameters of a second-order ARX model of the plant. The estimation algorithm includes a forgetting factor that allows the controller to work with linear time-varying (LTV) systems, and U-D cofactorization is incorporated into the algorithm to improve the numerical conditioning of the calculation of the covariance matrix. Several ad hoc improvements, such as covariance resetting and a dead-zone on the parameter error, have also been included in the algorithm to mitigate some of the difficulties encountered when attempting to identify a plant under closed-loop control. Two different control law design techniques have also been presented. The first method, referred to as *simplified self-tuning control* (SSTC), is a direct approach that employs a deadbeat control strategy. The SSTC attempts to cancel the poles of the plant model, but not the plant zeros. Although SSTC is not specifically designed for PID control, it can be readily modified to conform to a PID-like structure by imposing certain constraints on the algorithm. The method is somewhat limited, however,

in that the dynamics of the closed-loop system are inherent in the deadbeat-like controller structure and cannot be modified by the designer. The second control law design method presented is a pole-placement technique that is derived from a discrete-time PID controller model. The method allows for four closed-loop poles to be arbitrarily positioned while fixing two closed-loop zeros at the origin of the z-plane. Although the method is an indirect controller implementation, the additional intermediate calculations required are minimal.

The PID version of the SSTC controller has been selected for testing the concepts presented in Chapter 4. Although the SSTC solution to the adaptive control problem possesses certain inherent limitations, it contains a control law design mechanism that is easily implemented and will simplify the testing of the U-D RLS algorithm under closed-loop conditions. The adaptive control law may then be modified to incorporate a number of different control law design approaches. Several computer programs have been written to test the SSTC algorithm developed in the previous chapter. In this chapter, those programs will be briefly explained and simulation results will be presented in order to evaluate the performance of the algorithms. First, the RLS parameter estimation algorithm employing U-D cofactorization is tested in open-loop using a *pseudo- random binary sequence* (PRBS) as the input to the plant. The PRBS will be shown to approximate white noise in its frequency spectrum and thus meets the requirement that the input to the plant be persistently exciting. The RLS algorithm is then incorporated into the SSTC algorithm in a second program, and the SSTC adaptive control algorithm is simulated. Finally, the SSTC algorithm is constrained to a PID-like structure to produce an adaptive PID controller. The PID version of the SSTC controller is also simulated and analyzed in this chapter.

5.2 The Pseudo-Random Binary Sequence

In order to evaluate the U-D RLS algorithm, an input signal must be provided to the plant that is persistently exciting of a suitable order. In Chapter 4, it was shown that random signals are known to be persistently exciting of any order and would therefore be an ideal source of excitation to the plant. White noise is a realization of a random signal that contains constant power per unit bandwidth for all frequencies. (A signal can be considered to be white if its power density spectrum is flat over a frequency range that is much greater than the bandwidth of the system being considered.) The present value of a white noise signal is completely independent of all past values of the signal. It can be shown that the autocorrelation function[1] of white noise is an impulse of height σ^2 (the mean square value) at $\tau = 0$ (i.e., $\phi_{xx}(0)$), and zero for all other values of τ. White noise can thus be considered persistently exciting of any order.

A pseudo-random noise signal has the same type of autocorrelation function as white noise (i.e., an impulse function) but it is repeated with a period T. The autocorrelation function of a pseudo-random signal is given as:

$$\phi_{xx}(\tau) = \frac{1}{T}\int_0^T x(t)x(t+\tau)dt.$$

Davies (1970) presents a method for generating pseudo-random noise using a binary signal as shown in Figure 36.

---

[1]The Autocorrelation Function of a signal is a statistical measure of the degree to which future and past values of a signal are dependent on the current value of the signal. It is defined as:

$$\phi_{xx}(\tau) = \underset{\tau\to\infty}{\text{Lim}}\frac{1}{2T}\int_{-\tau}^{\tau} x(t)x(t+\tau)d\tau.$$

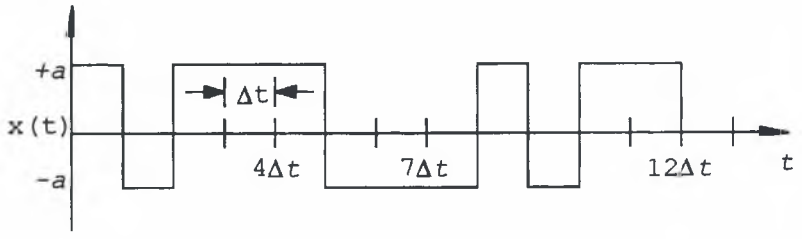

Figure 36. Example of pseudo-random binary signal

If the signal of Figure 36 repeats itself every $11\Delta t$ units of time, the autocorrelation function of the signal will be a periodic function as depicted in Figure 37.

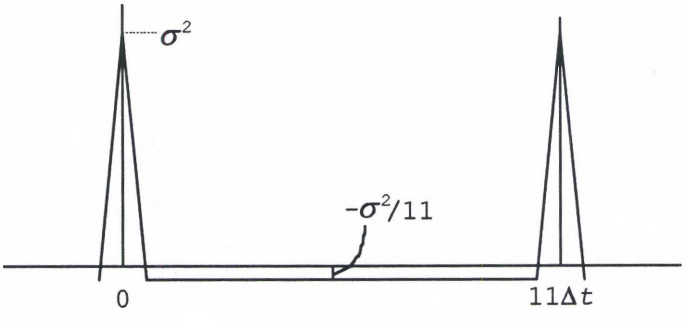

Figure 37. Autocorrelation function of pseudo-random binary signal of Figure 36

Davies (1970) presents several rules that must be adhered to in order to ensure that a pseudo-random binary signal approximates a truly random signal:

1. The signal must be periodic with period T, can only take on two constant values, $\pm a$, and can only change from one state to another at discrete times, $k\Delta t$, where Δt is a constant and k is an integer.
2. The number of $+a$ states should be approximately equal to the number of $-a$ states (the difference should not exceed one).

3. In every period, runs of consecutive $+a$ states or $-a$ states should frequently occur, with short runs being more frequent than long runs, e.g., 1/2 of the runs of length one, 1/4 of the runs of length two, 1/8 of the runs of length three, etc. Also, for each run length, the $+a$ states should equal the $-a$ states.
4. The autocorrelation function of the signal should be two-valued, peaking in the middle and flat toward the ends.

Davies states that any binary sequence with all of the above properties can be defined as a pseudo-random binary signal (PRBS).

A PRBS can be generated using a shift register with a modulo-2 gate in a feedback loop. For a given number of stages in the shift register, there is a maximum number of digits that occur before the sequence repeats itself. This is referred to as the *maximum length sequence.* Not every feedback connection will result in a maximum length sequence. The largest possible period for an *n*-stage shift register is $(2^n - 1)$. Therefore, if a given output sequence has a period:

$$N = 2^n - 1,$$

the sequence is a maximum length sequence. Since the maximum length sequence is a square wave, its autocorrelation function can be written as:

$$\phi_{xx}(P) = \frac{1}{N}\sum_{j=1}^{N} x(j)x(j+k).$$

The autocorrelation function of the binary maximum length sequence is shown graphically in Figure 38. A slight d.c. component can be seen in the figure, but for large *N,* Davies (1970) asserts that this can be safely neglected.

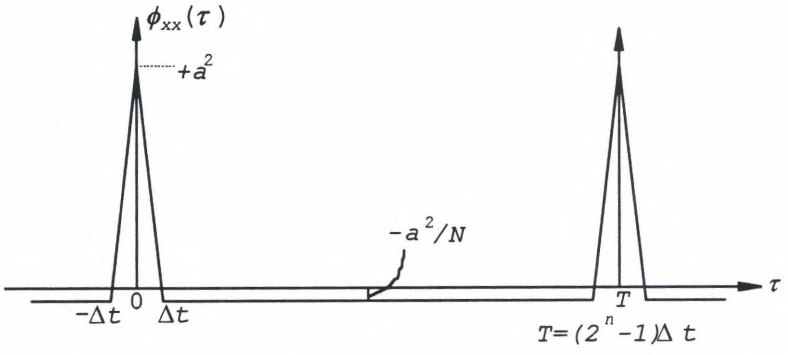

Figure 38. Autocorrelation function of a binary maximum length sequence

Provided the feedback operation in the shift register is restricted to addition, the shift register generator is a linear device. The possible feedback combinations that produce maximum length sequences can thus be determined algebraically and can be implemented in a digital computer program. Davies (1970) has produced a table that includes all the possible feedback connections that produce maximum length sequences up to $n = 10$. Based on Davies' (1970) results, a routine was written to generate a pseudo-random binary sequence with $N = 1023$ to be used as a persistently exciting input signal with which to test the U-D RLS algorithm. The open-loop test results are presented in the next section.

5.3 Testing the U-D RLS Algorithm in Open Loop

A QuickBASIC program has been written to test the parameter estimation algorithm to be used as part of the adaptive PID controller. A simplified flowchart of the program is shown in Figure 39. The PRBS that serves as the input to the simulated plant is generated from within the program. The PRBS generated by the program may be seen in Figure 40. A frequency divider is built into the PRBS routine the allows the PRBS

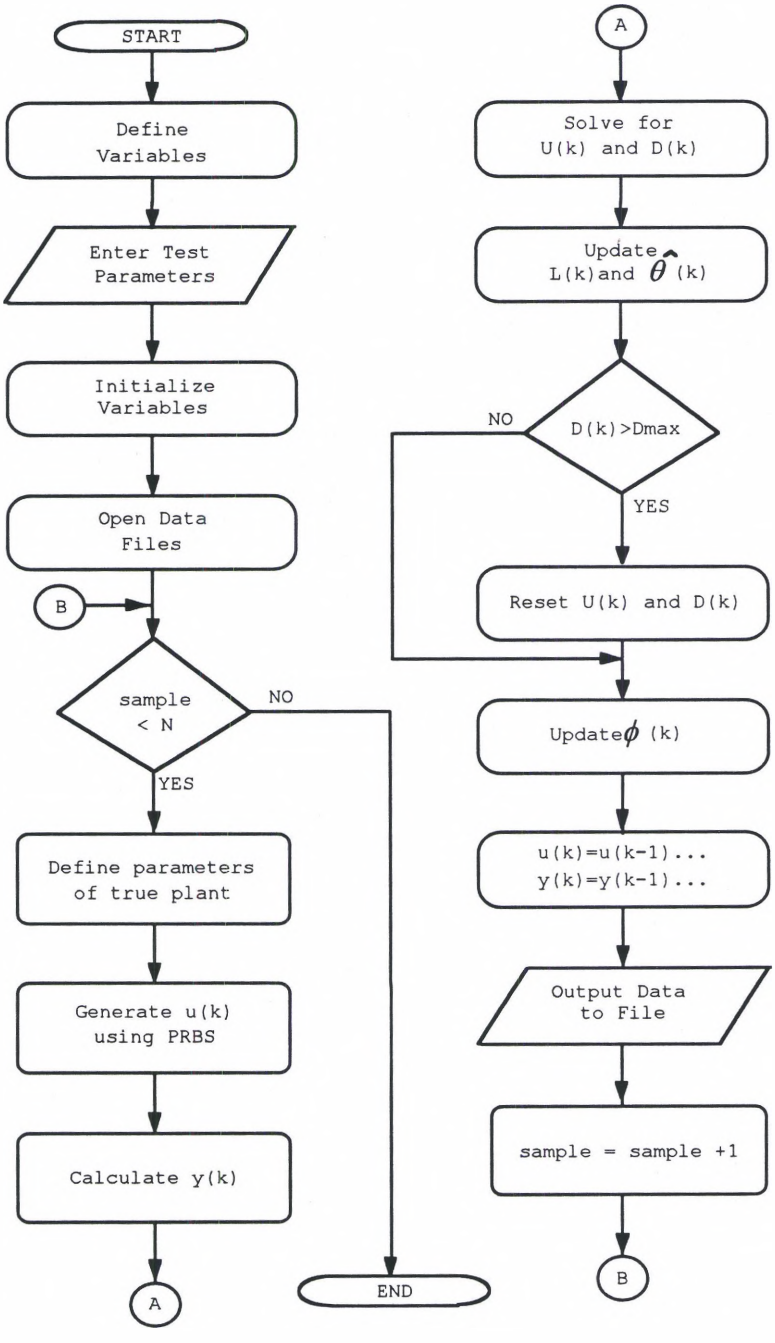

Figure 39 Simplified flowchart of U-D RLS test program

sampling interval to be made a multiple of the plant sampling interval. The PRBS will have different frequency characteristics depending on the sampling interval selected. The program also simulates the response of the *true* plant using an ARX model. The input and

output of the simulated true plant are used to form the regression vector used in the U-D RLS algorithm. The covariance matrix $\mathbf{D}(k)$ is bounded in the program and will automatically reset to $\mathbf{D}(0)$ if the estimator *winds up* to the upper bound $\mathbf{D}_{max}$. The RLS algorithm also includes a forgetting factor and the program allows the plant parameters to change in the middle of the simulation to test the ability of the algorithm to track time-varying parameters. The plant input and output, $u(t)$ and $y(t)$, the parameter error $\varepsilon(t)$, the covariance and gain matrices, $U(t), D(t)$ and $L(t)$, and the estimated parameters, $\hat{\theta}(t)$ are all saved to files on the computer hard drive for later analysis.

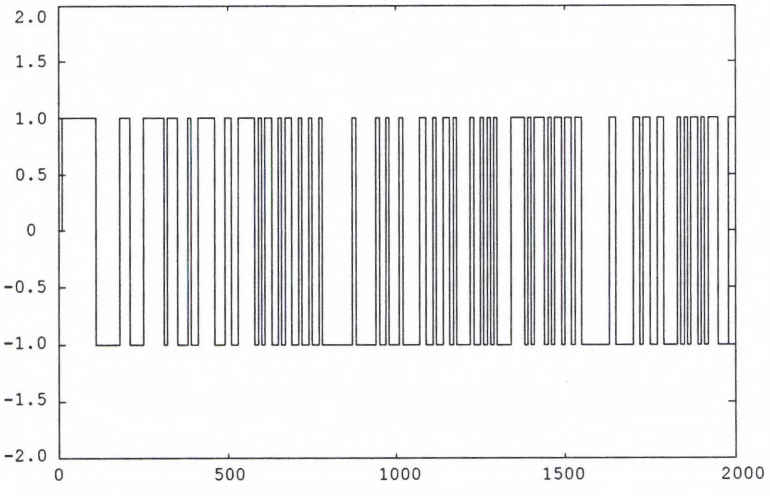

Figure 40. Pseudo-random binary sequence used as input to the plant

The first test of the U-D RLS algorithm determines how the forgetting factor λ affects the convergence rate of the parameter vector $\hat{\theta}(t)$. A second-order approximation of the plant used in the trials of Chapter 3 is selected for the test. The plant is described by the transfer function:

$$G_p(s) = \frac{1}{(1+s)(1+.26s)}. \tag{5.1}$$

The continuous-time model of equation (5.1) is discretized using the zero-order hold equivalent with a sampling interval of $T = 1$ sec. The resulting discrete-time model is given as:

$$G_p(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} = \frac{0.5103711q^{-1} + 0.1082462q^{-2}}{1 - 0.3892412q^{-1} + 0.0078585q^{-2}}. \tag{5.2}$$

The *true* plant parameters to be estimated are thus:

$$\begin{aligned} a_1 &= -0.3892412 \\ a_2 &= 0.0078585 \\ b_1 &= 0.5103711 \\ b_2 &= 0.1082462 \end{aligned} \tag{5.3}$$

The forgetting factor λ is set to 1.0, 0.999, 0.995 and 0.95 for four trials. Due to the relatively slow dynamics of the plant, the sampling rate of the PRBS is multiplied by 10 in order to adequately excite the slower modes of the system. The covariance matrix $\mathbf{D}(k)$ is limited to $\mathbf{D}_{max} = 10^6$ and resets to its initial value of $\mathbf{D}(0) = 10$ if the upper bound $\mathbf{D}_{max}$ is reached. The results of the four trials are shown in Figures 41, 42, 43 and 44.

The parameter convergence appears exponential in each case, which is characteristic of the RLS algorithm. The differences in the parameter convergence rates are most evident in the parameter a_1. In Figures 41 and 42, a_1 converges slower than in Figure 43. In fact, a_1 does not reach steady-state in any of the first three trials. In Figure 44, however, a_1 appears to have completely converged by $t = 600$. In the four trials, as the forgetting factor λ is decreased, the parameters converge faster.

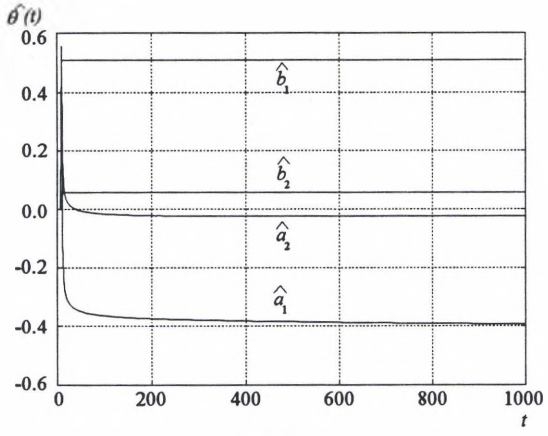

Figure 41. $\hat{\theta}(t)$ for $\lambda = 1.0$

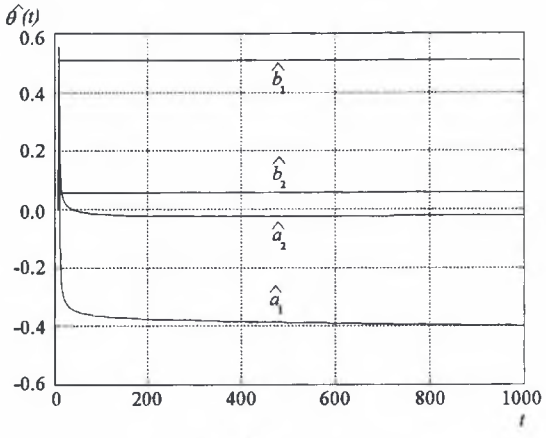

Figure 42. $\hat{\theta}(t)$ for $\lambda = 0.999$

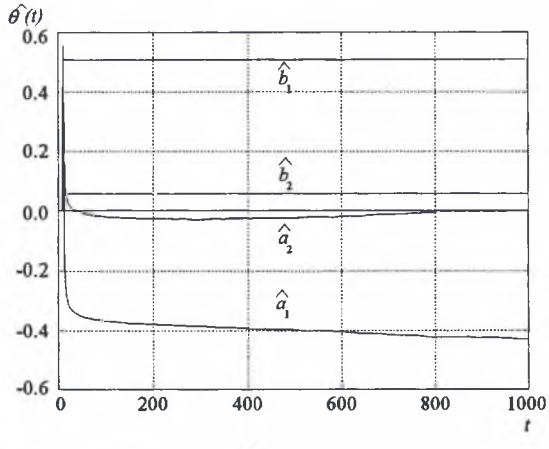

Figure 43. $\hat{\theta}(t)$ for $\lambda = 0.995$

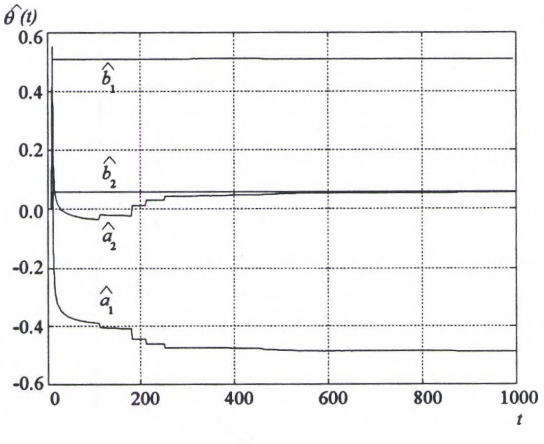

Figure 44. $\hat{\theta}(t)$ for $\lambda = 0.95$

The parameter error $\varepsilon(t)$ is shown for each of the above cases in Figures 45, 46, 47 and 48. In the deterministic case, the parameter error $\varepsilon(t)$ is a measure of the accuracy of the estimated model relative to the true plant as given by:

$$\varepsilon(t) = y(t) - \hat{\theta}^T(t)\phi(t).$$

For $\lambda = 1.0$ and $\lambda = 0.999$, $\varepsilon(t)$ is still significant even after 2000 samples. For $\lambda = 0.995$, $\varepsilon(t)$ converges to nearly zero after 1500 samples; however, for $\lambda = 0.95$, $\varepsilon(t)$ becomes insignificant relatively quickly (in less than 250 samples). Analysis of frequency response plots reveals that for $\hat{\theta}(2000)$, the case where $\lambda = 0.95$ produces a slightly more

accurate model than do the other cases. As λ decreases in value, however, the estimates become more susceptible to noise.

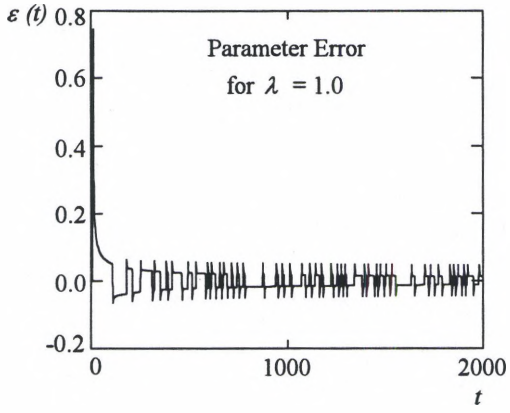

Figure 45. $\varepsilon(t)$ for $\lambda = 1.0$

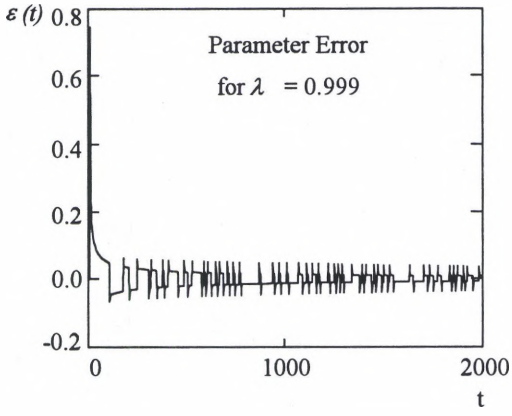

Figure 46. $\varepsilon(t)$ for $\lambda = 0.999$

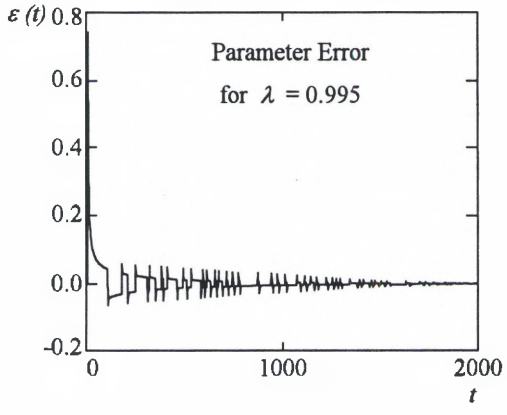

Figure 47. $\varepsilon(t)$ for $\lambda = 0.995$

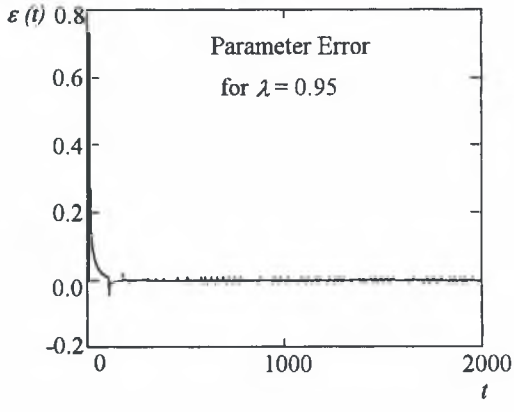

Figure 48. $\varepsilon(t)$ for $\lambda = 0.95$

The next trial assesses the ability of the U-D RLS algorithm to track time-varying parameters. A 3000 sample test is performed with the parameters of the true plant changing at $t = 1000$ and at $t = 2000$. The parameters for the original plant model of equation (5.2) (Plant 1) and for the two modified plants (Plant 2 and Plant 3) are shown in Table 12:

Table 12. Parameters of *true* Plants 1, 2 and 3

parameter	Plant 1	Plant 2	Plant 3
a_1	-0.3892412	-0.50	-0.60
a_2	0.0078585	-0.01	-0.05
b_1	0.5103711	0.40	0.30
b_2	0.1082462	0.08	0.005

The magnitude and phase plots of the original plant (Plant 1) and the two modified plants (Plant 2 and Plant 3) are shown in Figure 49.

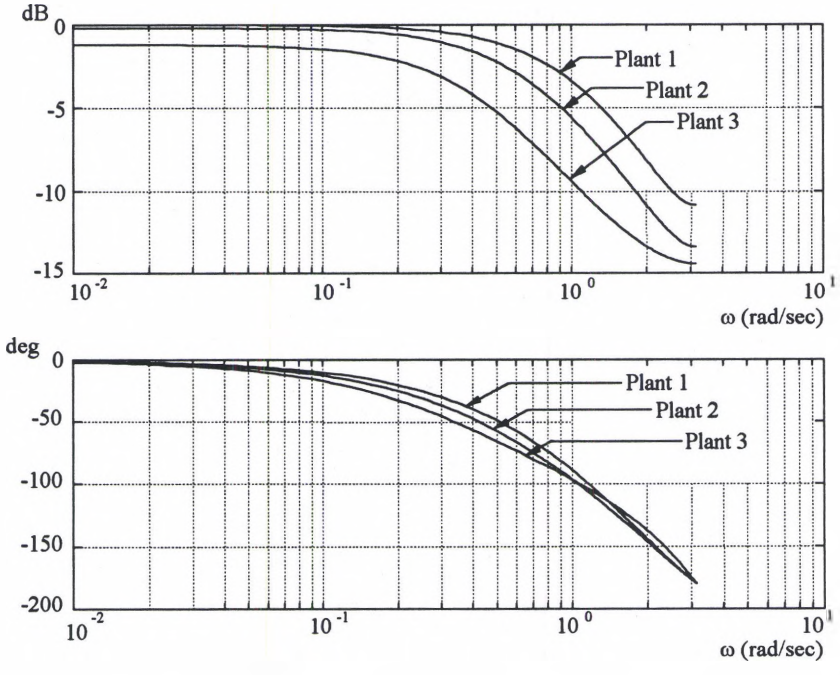

Figure 49. Magnitude and phase responses of three test plants

The forgetting factor λ is set at 0.95 and the upper bound on $\mathbf{D}(t)$ is set at 10^6. A graph of the estimated parameters $\hat{\theta}(t)$ vs. time for the simulation is shown in Figure 50.

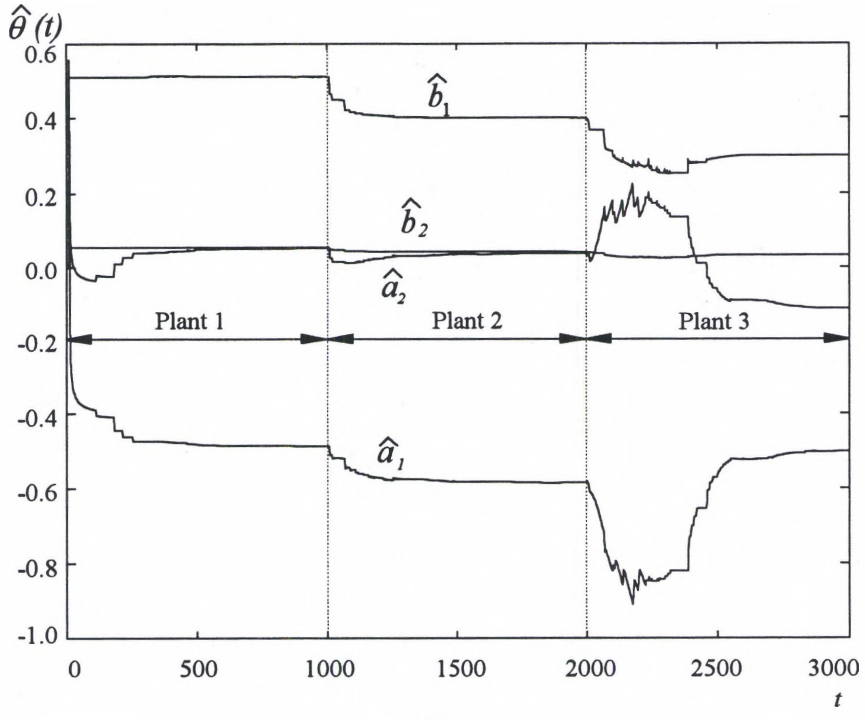

Figure 50. Graph of parameter vector $\hat{\theta}(t)$ tracking time-varying plant parameters

The values of the estimated parameters at $t = 1000$ and at $t = 2000$ and are listed in Table 13.

Table 13. Estimated and actual parameters for test of time-varying system

θ_o	Plant 1 (actual)	Plant 1 (estimated)	Plant 2 (actual)	Plant 2 (estimated)	Plant 3 (actual)	Plant 3 (estimated)
a_1	-0.3892412	0.4876259	-0.50	-0.5850750	-0.60	-0.5022256
a_2	0.0078585	0.0536163	-0.01	0.0379177	-0.05	-0.1144118
b_1	0.5103711	0.5105271	0.40	0.3992493	0.30	0.2997126
b_2	0.1082462	0.5546293	0.08	0.0431978	0.005	0.0342870

Table 13 indicates that the estimated parameters do not converge to the actual parameters of Plants 1, 2 or 3. However, the magnitude and phase plots of Figures 51, 52 and 53, respectively, show that the frequency responses of the estimated models are nearly identical to those of the actual plants.

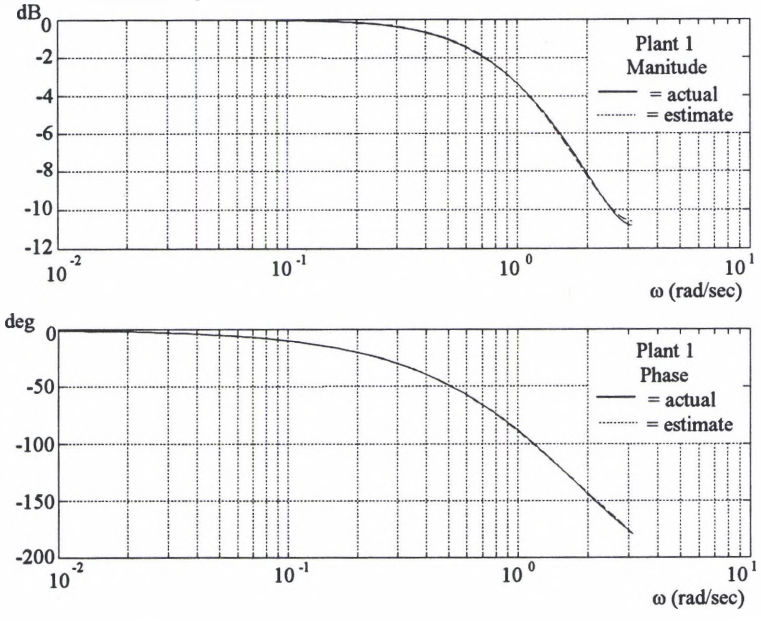

Figure 51. Magnitude and phase plots of actual Plant 1 and estimated Plant 1

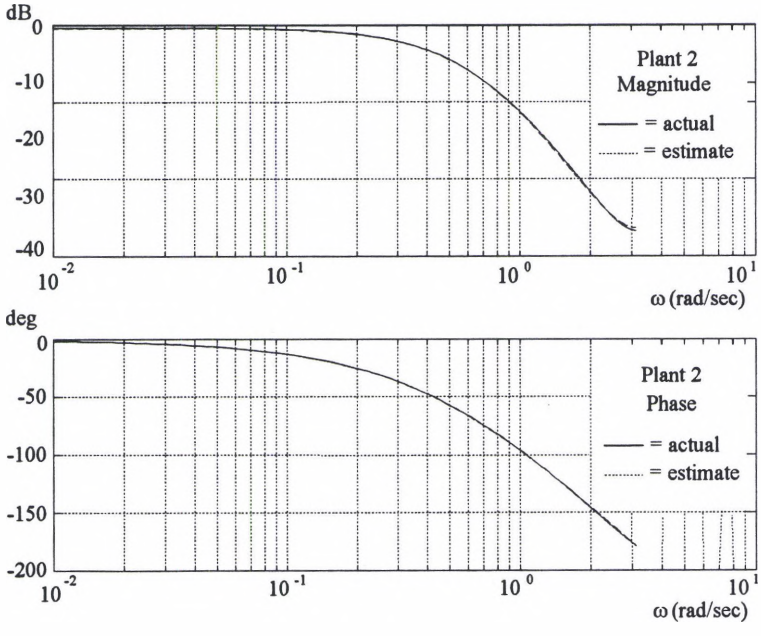

Figure 52. Magnitude and phase plots of actual Plant 2 and estimated Plant 2

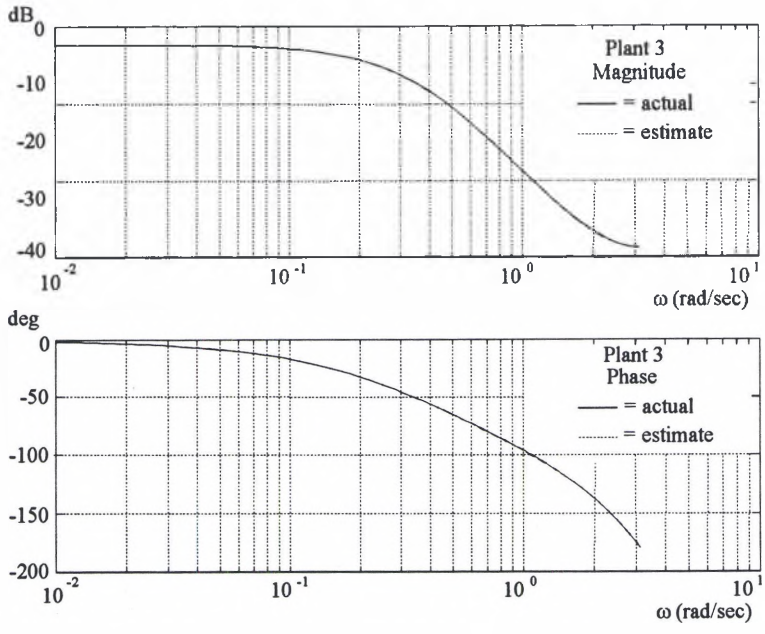

Figure 53. Magnitude and phase plots of actual Plant 3 and estimated Plant 3

The Bode diagrams demonstrate that the estimator is capable of accurately tracking time-varying parameters for the noise-free case. A graph of the elements of the information matrix $\mathbf{U}(t)$ for the trial is shown in Figures 54. (Recall that $\mathbf{U}(t)$ is upper triangular with ones on the diagonal, and for the second-order model, it can be described by six elements.) Although the elements of $\mathbf{U}(t)$ vary widely, they do remain bounded as predicted. A plot of the elements of the covariance matrix $\mathbf{D}(t)$ is shown in Figure 55. The regularization of $\mathbf{D}(t)$ is evident in the figure. The effect of automatic covariance resetting is also apparent as the elements of $\mathbf{D}(t)$ are reset to $\mathbf{D}(0)$ when the upper bound of $\mathbf{D}_{max} = 10^6$ is reached.

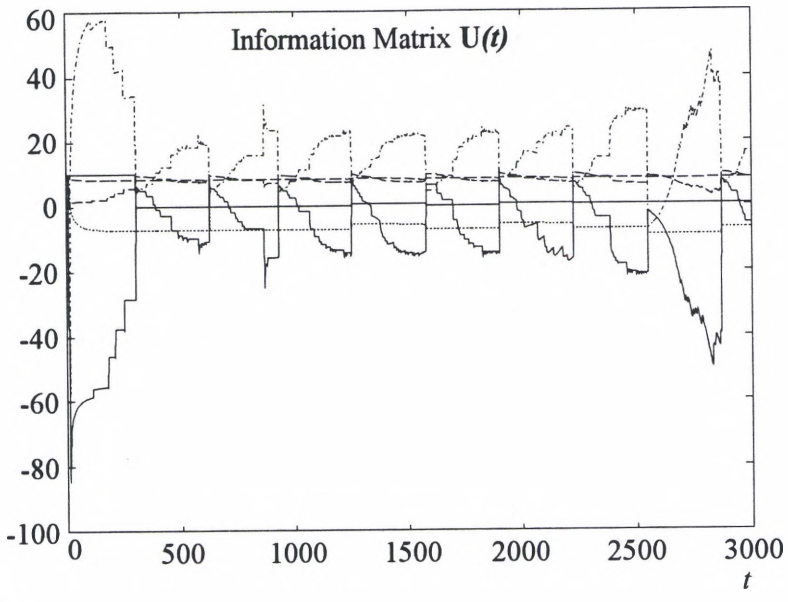

Figure 54. Elements of information matrix $\mathbf{U}(t)$

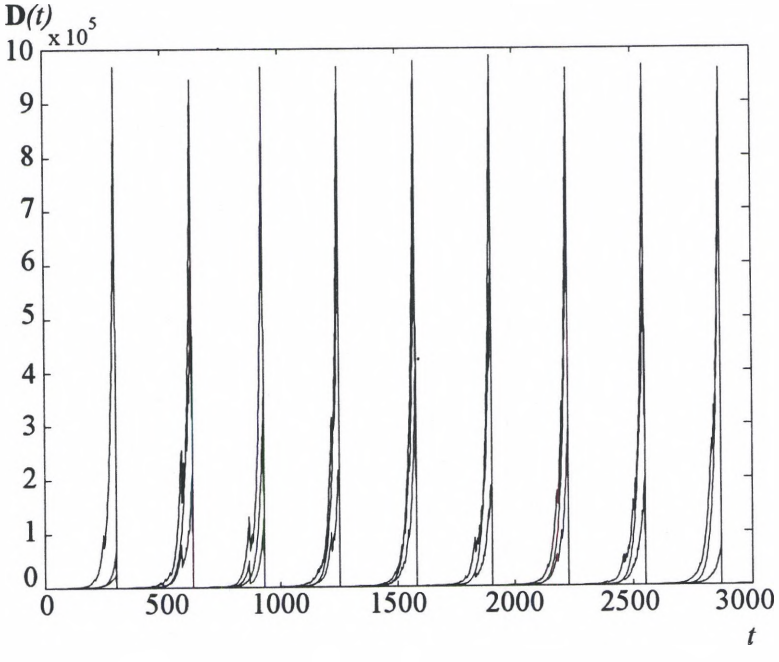

Figure 55. Elements of covariance matrix $\mathbf{D}(t)$

The above simulations demonstrate that the U-D RLS estimation algorithm performs well for the deterministic case. The algorithm is now tested with measurement noise present in the system. For the simulation, the noise, denoted as $\eta(t)$, is generated using a PRBS and is introduced to the *true* plant by the relation:

$$A(q)y(t) = B(q)u(t) + \eta(t). \tag{5.4}$$

The 3000 sample simulation presented above is repeated with the true plant changing from Plant 1 to Plant 2 at $t = 1000$ and from Plant 2 to Plant 3 at $t = 2000$. For this trial, however, noise with a mean of zero and a variance of 0.05 is added to the system by the relation given in equation (5.4). A sample output of the plant response with noise added is shown in Figure 56.

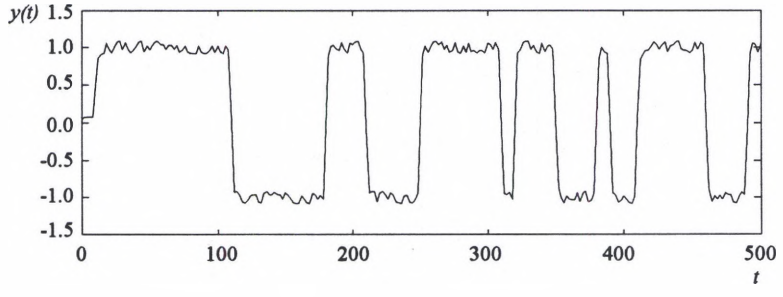

Figure 56. Plant output $y(t)$ with added noise

With the forgetting factor $\lambda(t)$ set to 0.95, the resulting estimated parameters are shown in Figure 57.

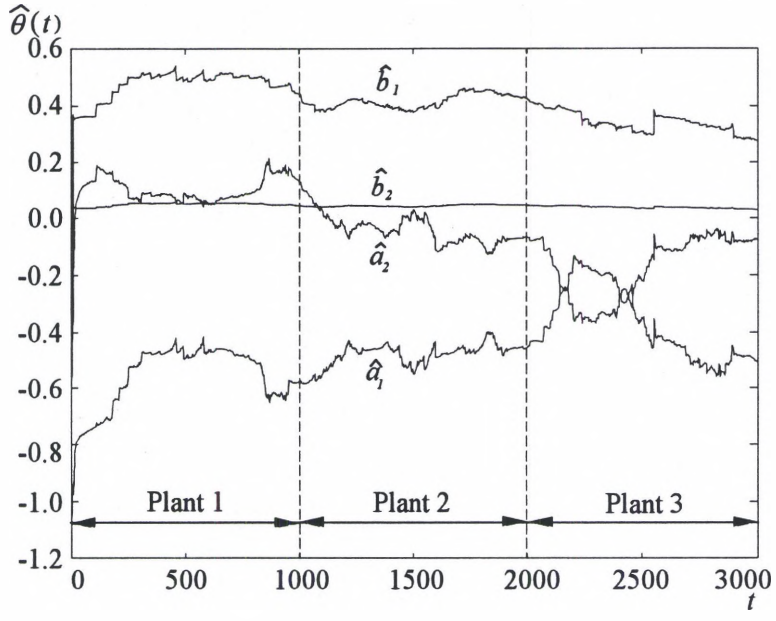

Figure 57. Estimated parameters with noise present

The behavior of the parameters is significantly more erratic with noise present. The added noise prevents the parameters from converging with $\lambda = 0.95$. In order to evaluate the accuracy of the estimated plant models, the frequency responses of the estimates at $t = 1000,\ 2000$ and 3000 are shown in Figures 58, 59 and 60 respectively. It is evident that the presence of added noise in the system has caused the estimates to be biased.

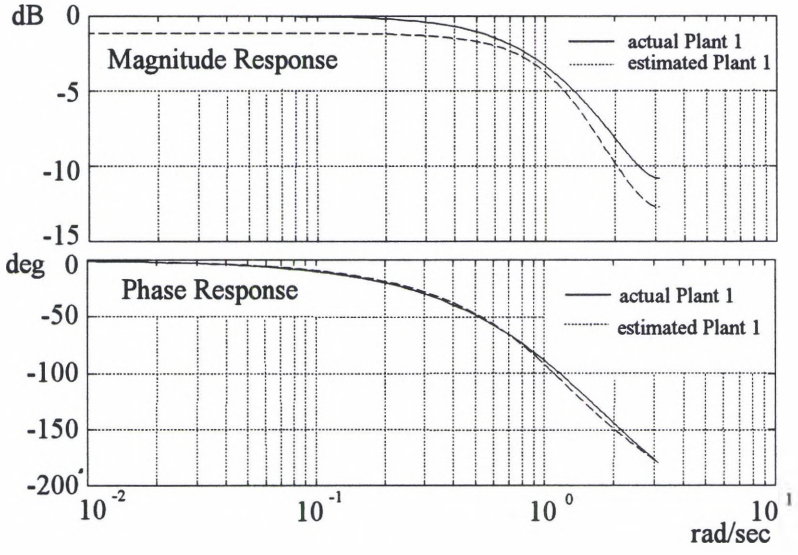

Figure 58. Frequency response of estimated and actual Plant 1 with noise

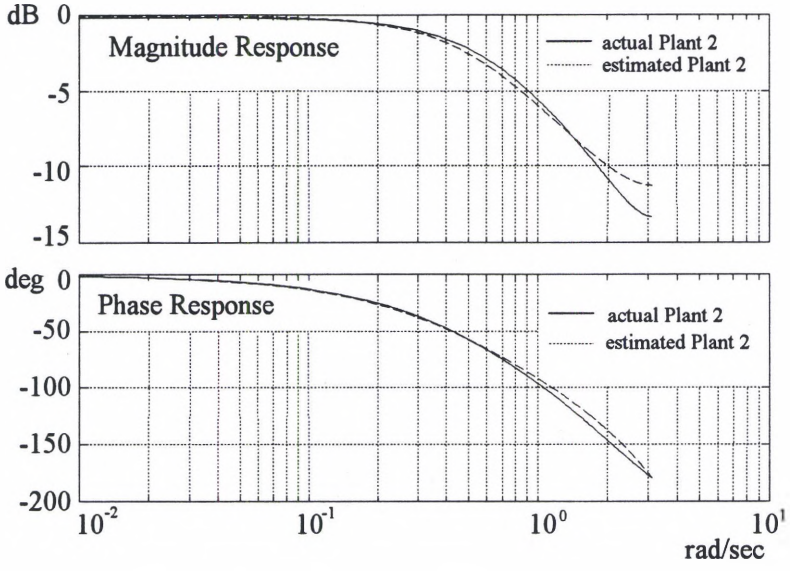

Figure 59. Frequency response of estimated and actual Plant 2 with noise

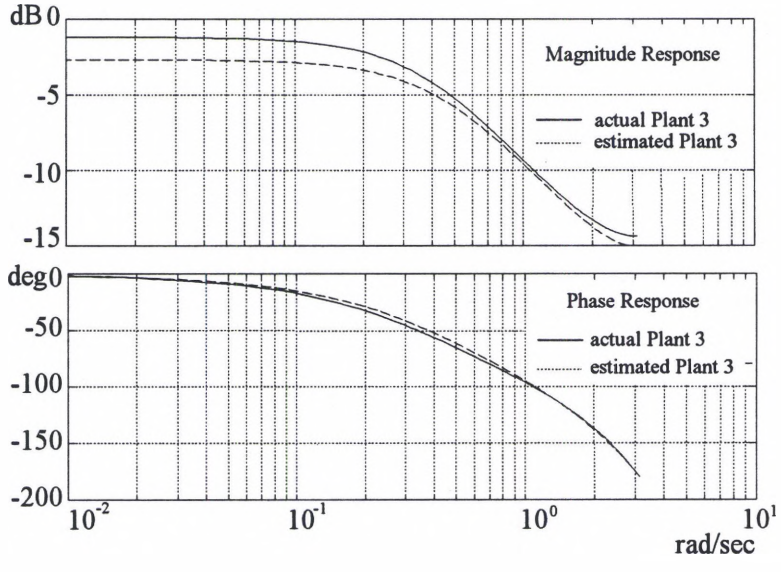

Figure 60. Frequency response of estimated and actual Plant 3 with noise

The effect of the noise on the estimates can be reduced if the input and output data to the parameter estimator are prefiltered with a low-pass filter. Consider a second-order

Butterworth filter with a cutoff frequency $\omega_c = 3$ rad / sec, discretized using the bilinear transformation with a sampling interval of one second. The discrete-time filter transfer function is determined to be:

$$H_f(z^{-1}) = \frac{0.4188914 + 0.8277828z^{-1} + 0.4188914z^{-2}}{1 + 0.4654349z^{-1} + 0.2101308z^{-2}}. \tag{5.5}$$

The magnitude response of the filter is shown in Figure 61.

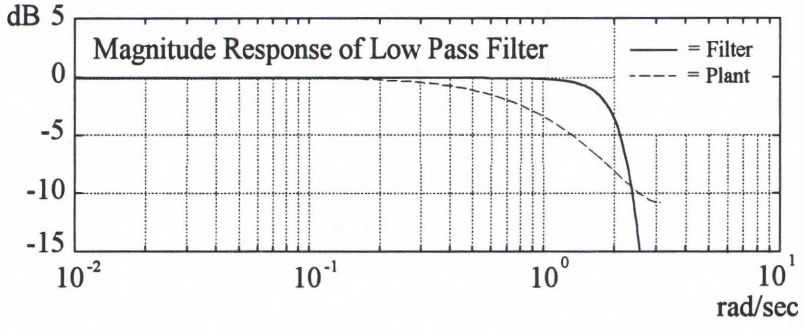

Figure 61. Magnitude response of second-order Butterworth filter H_f

Two 3000 sample simulations are performed with the true plant remaining fixed as Plant 1 and the forgetting factor λ set at 0.95. In the first simulation, noise is added to the system and the estimator data is not filtered. In the second simulation, the estimator data is pre-filtered through H_f. The results of the two simulations are presented in Figures 62 through 65.

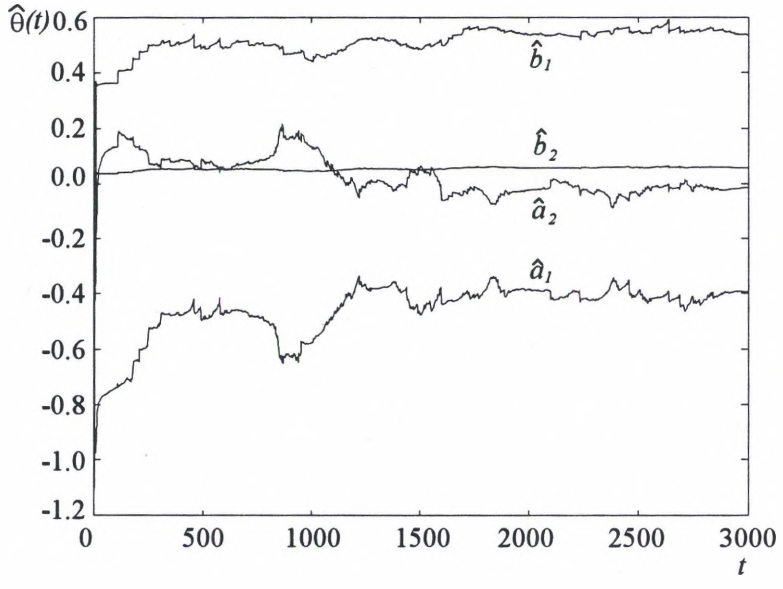

Figure 62. Estimated parameters obtained with unfiltered estimator data

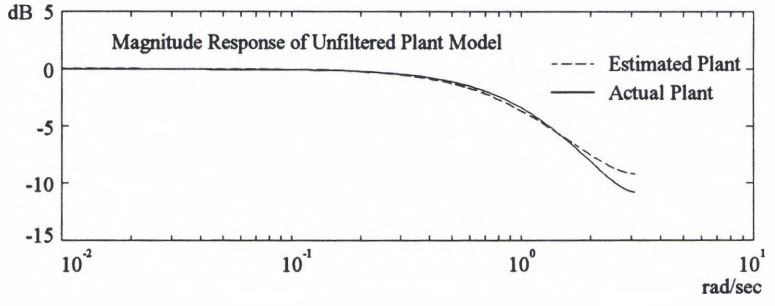

Figure 63. Magnitude response of plant model obtained with unfiltered data

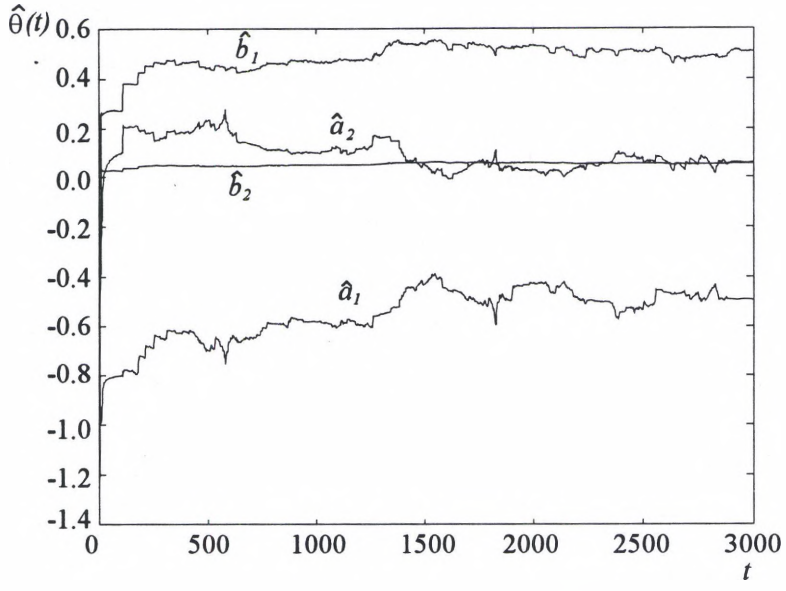

Figure 64. Estimated parameters obtained with filtered estimator data

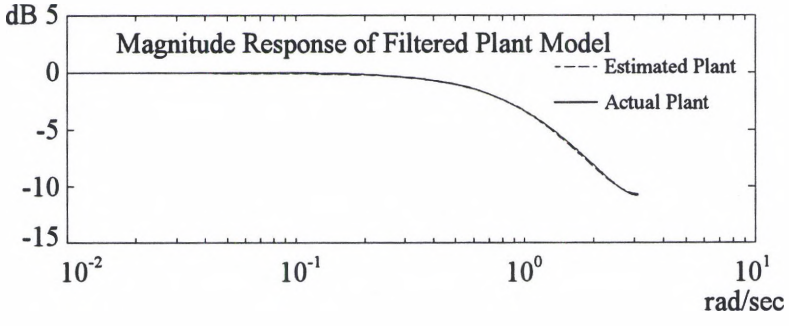

Figure 65. Magnitude response of plant model obtained with filtered data

Filtering of the data has caused the behavior of the estimated parameters in $\hat{\theta}(t)$ to be less erratic. In each case, the presence of noise slows the convergence rate of the parameter estimates compared to the noise-free case presented earlier. Filtering the estimator data has also improved the accuracy of the magnitude response of the estimated plant, particularly in the high frequency region. The two preceding simulations have shown the importance of data filtering when attempting parameter estimation in the presence of noise.

The tests performed in this section have demonstrated the capability of the U-D RLS estimation algorithm to generate accurate models when the plant is properly excited in the open-loop case. The estimator is able to accurately track time-varying parameters, and with proper filtering, can provide an accurate model of the plant even in the presence of low level noise.

5.4 Simulation of the SSTC Controller

A deterministic assessment

The U-D RLS algorithm is now incorporated into a program that simulates the Simplified Self-Tuning Controller. To test the operation of the SSTC control law design mechanism, a simulation is run with the estimated plant parameters held constant. A second-order ARX model is assumed for the plant, given as:

$$G_p(q^{-1}) = \frac{b_1 q^{-1} + b_2 q^{-2}}{1 + a_1 q^{-1} + a_2 q^{-2}}.$$

The parameters of the true plant being controlled are assumed to be:

$$a_1 = -0.3892412$$
$$a_2 = 0.0078585$$
$$b_1 = 0.5103711$$
$$b_2 = 0.1082462$$

which correspond to Plant 1 of the previous section. For the purposes of this test, the true plant parameters are used in the control law calculation as the controller parameters. The control law for the SSTC controller was given in Chapter 4 as:

$$u(t) = \frac{A(q^{-1})e(t)}{B(1)} + \frac{B(q^{-1})u(t-d)}{B(1)},$$

which for the second-order plant model is determined to be:

$$u(t) = \frac{e(t) + a_1 e(t-1) + a_2 e(t-2) + b_1 u(t-1) + b_2 u(t-2)}{b_1 + b_2}. \tag{5.6}$$

The resulting plant and controller outputs in response to the reference input switching between plus and minus one may be seen in Figure 66.

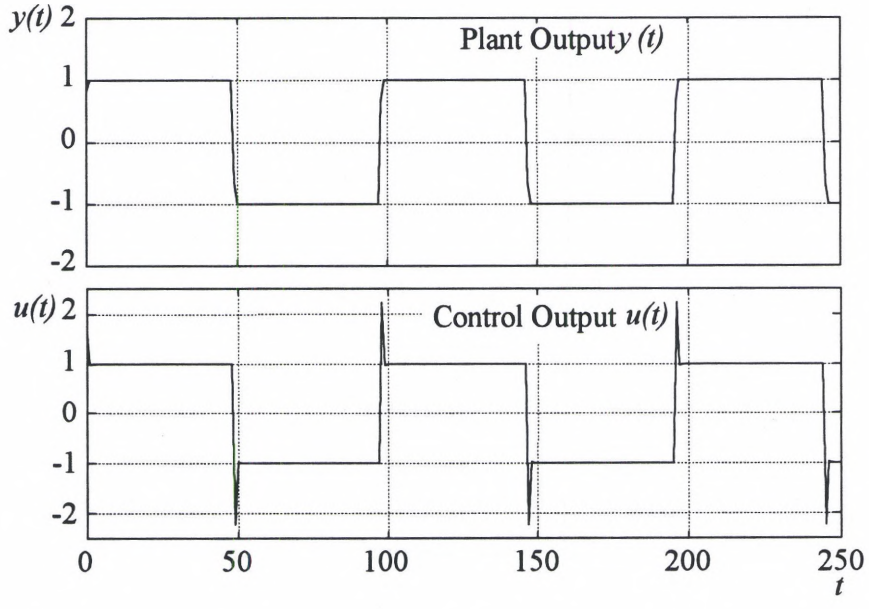

Figure 66. Plant and control outputs using SSTC control algorithm with fixed parameters

The plant output is typical of the deadbeat-like control strategy used in the SSTC algorithm, indicating that the control law design algorithm is functioning properly.

The U-D RLS algorithm is now enabled in the SSTC simulation program, making the controller truly adaptive. The program allows the plant model to conform to an ARX model of any order. In this simulation, the plant model is assumed to be second-order, requiring the estimation of four plant parameters. The true plant is left the same as in the previous simulation, given as:

$$G_p(q^{-1}) = \frac{0.5103711q^{-1} + 0.1082462q^{-2}}{1 - 0.3892412q^{-1} + 0.0078585q^{-2}}.$$

Since the SSTC conforms to a direct adaptive control structure, this implies that there are four controller parameters to be established as well. Several other controller parameters, such as $\mathbf{U}(0)$, $\mathbf{D}(0)$, $\mathbf{D}_{max}$, $\hat{\theta}(0)$ and λ, must also be specified before running the simulation. Pre-testing of the algorithm indicated that the selection of $\mathbf{U}(0)$ and $\mathbf{D}(0)$ is somewhat arbitrary, as they only affect the estimates for the first few samples. Ljung and Soderstrom (1983), however, suggest that they be set to a relatively large value. $\mathbf{U}(0)$ and $\mathbf{D}(0)$ are set to 10 for this simulation. The selection of $\mathbf{D}_{max}$, the upper bound on the covariance matrix $\mathbf{D}(t)$, becomes a trade-off between the sensitivity and accuracy of the estimated plant parameters. If $\mathbf{D}_{max}$ is fixed at too large a value, the estimates tend to be unstable and overly sensitive to noise in closed-loop estimation. Too small a value of $\mathbf{D}_{max}$, on the other hand, can produce less accurate estimates. Pre-testing indicated that $\mathbf{D}_{max} = 10^5$ achieves an acceptable balance in this case. In testing the open-loop estimator in the previous section, $\hat{\theta}(0)$ was set to zero; however, examination of equation (5.6) reveals that the sum of parameters b_1 and b_2 cannot be permitted to be equal to zero. An initial value of 0.5 is arbitrarily selected for the elements of the parameter vector $\hat{\theta}(t)$. If the parameters b_1 and b_2 should ever go to zero during the simulation, the program is designed to change the control law to be:

$$u(t) = u(t-1)$$

to ensure controller stability until the parameters return to acceptable values. Since the open-loop trials yielded the best results with the forgetting factor $\lambda = 0.95$, λ was also set at 0.95 for this simulation. The results of a 500 sample simulation may be seen in Figure 67.

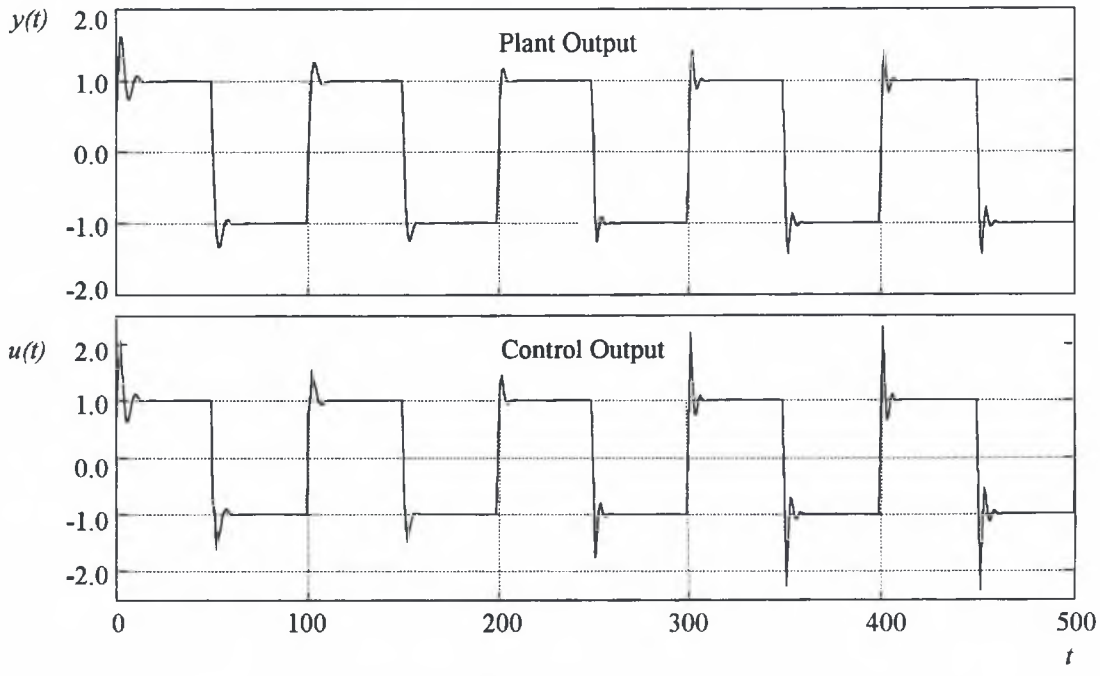

Figure 67. Plant and control outputs of second-order SSTC controller with $\lambda = 0.95$

After only two or three cycles of the reference input, the controller has tuned itself to the point where the overshoot of the plant output is less than 10% and the settling time is less than 10 seconds. The optimal solution of Figure 66 yielded an output with no overshoot; however, the overshoot observed in Figure 67 is attributed to modeling error due to a lack of a persistently exciting input to the plant, since the only excitation in this case is the change in the reference input. At $t = 250$, the overshoot significantly increases. The reason for this can be seen in the behavior of the elements of the resultant parameter vector shown in Figure 68. The parameters appear to have converged at around $t = 100$. Subsequently, the estimates make relatively large changes each time the reference input is cycled. Again, this is attributed to the lack of excitation to the plant and the forgetting factor λ being set to heavily weight the most recent data to the estimator. The simulation indicates that the forgetting factor λ needs to be set higher to make the estimator less sensitive to sudden changes in the reference input.

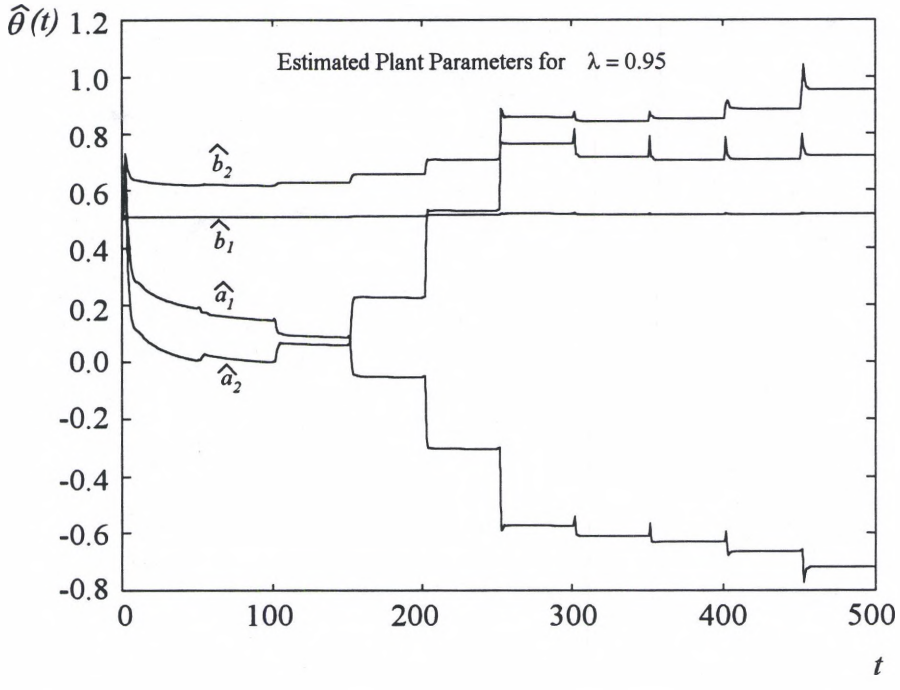

Figure 68. Estimated plant parameters for second-order SSTC with $\lambda = 0.95$

In the deterministic case, the parameter error $\varepsilon(t)$ represents the modeling error of the estimated plant versus the true plant. The parameter error $\varepsilon(t)$ for the simulation above is shown in Figure 69. The parameter error eventually converges to zero when the plant output is in steady-state following each change in the reference input. $\varepsilon(t)$ jumps to higher levels, however, during changes in the reference input. This indicates that the estimated model is accurate at low frequencies, but is less accurate at higher frequencies. Since the plant is only excited every 50 samples when the reference input changes signs, the model tends to be biased toward the lower frequencies.

The behavior of the information and covariance matrices $\mathbf{U}(t)$ and $\mathbf{D}(t)$ for the simulation may be seen in Figures 70 and 71, respectively.

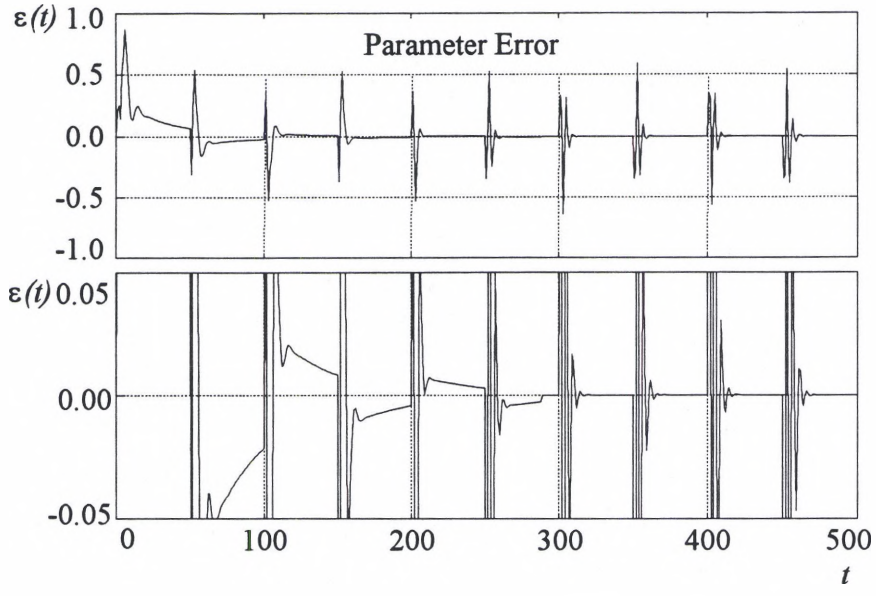

Figure 69. Parameter error for second-order SSTC with $\lambda = 0.95$

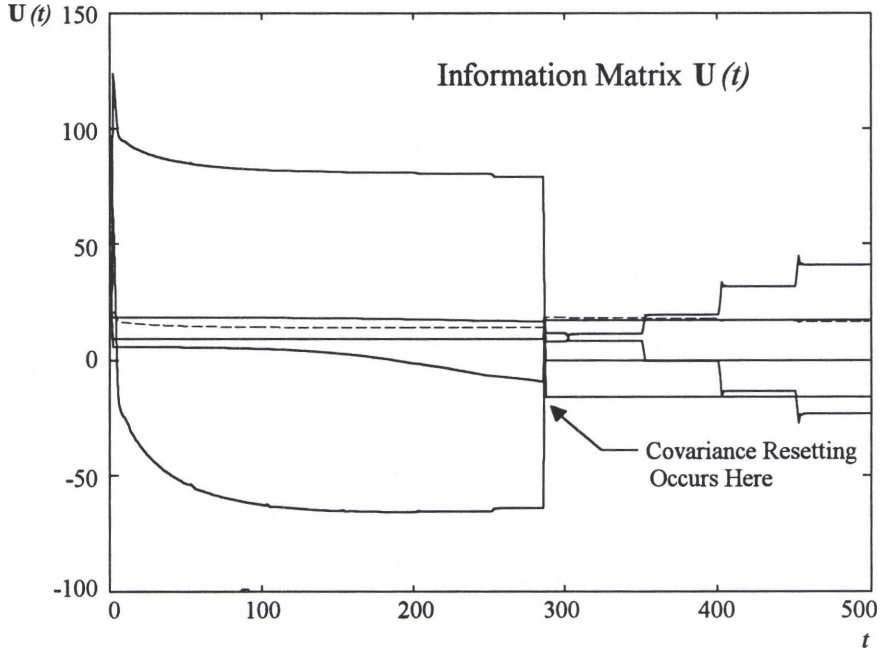

Figure 70. $\mathbf{U}(t)$ for second-order SSTC with $\lambda = 0.95$

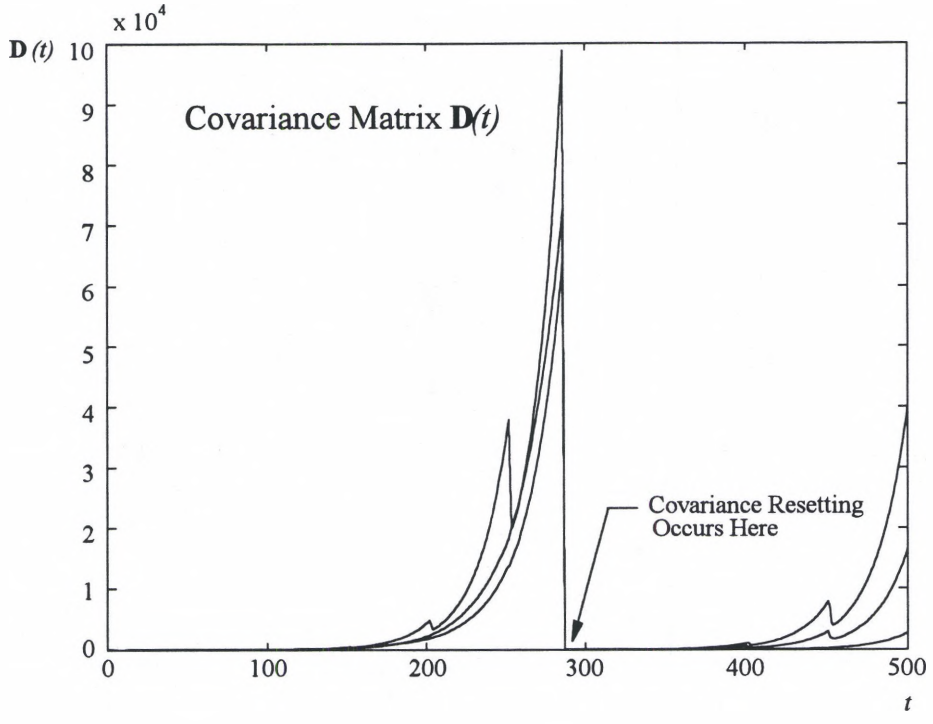

Figure 71. $\mathbf{D}(t)$ for second-order SSTC with $\lambda = 0.95$

The elements of $\mathbf{U}(t)$ remain bounded as predicted. The tendency of the covariance matrix $\mathbf{D}(t)$ to become singular when the plant is not sufficiently excited was discussed in Chapter 4. One of the difficulties encountered in closed-loop estimation when employing a forgetting factor is that the covariance matrix $\mathbf{P}(t)$ (or in this case, $\mathbf{U}(t)\mathbf{D}(t)\mathbf{U}^T(t)$) is governed by the relationship:

$$\mathbf{P}(t) = \frac{\mathbf{P}(t-1)}{\lambda} \tag{5.7}$$

when no new information is contained in the measurements. The exponential effect of equation (5.7) is seen in the elements of $\mathbf{D}(t)$. *Covariance resetting* has been incorporated into the SSTC algorithm to reset $\mathbf{U}(t)$ and $\mathbf{D}(t)$ to $\mathbf{U}(0)$ and $\mathbf{D}(0)$ when $\mathbf{D}(t)$ reaches the limit $D_{max} = 10^5$. The elements of the estimator gain matrix $\mathbf{L}(t)$ are plotted in Figure 72.

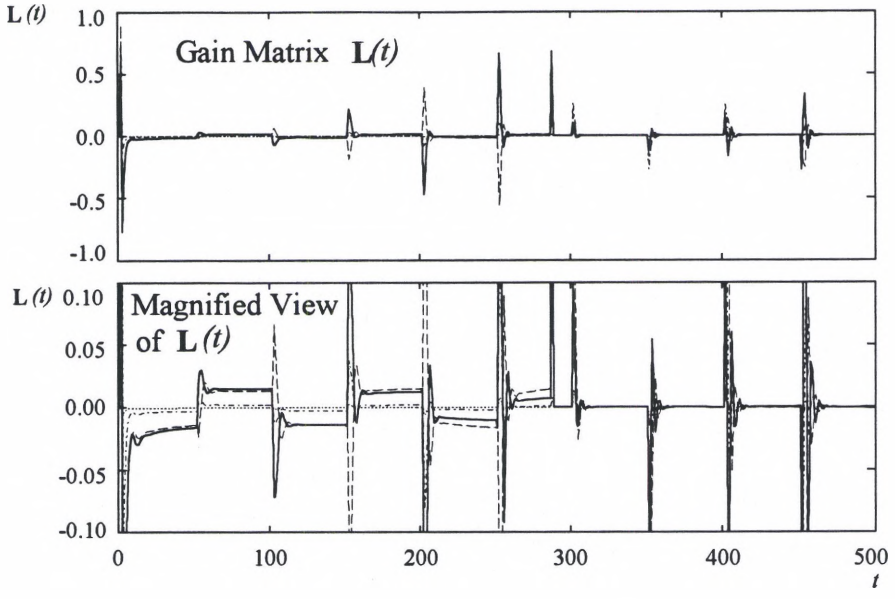

Figure 72. $\mathbf{L}(t)$ for second-order SSTC with $\lambda = 0.95$

The estimator gains achieve very small values (less than 0.05) in a few samples, demonstrating the rapid convergence rate of the U-D RLS algorithm. When the parameters diverge at around 150 samples, the gains become large during the transients as the estimator attempts to minimize $\varepsilon(t)$.

The first simulations indicate that the second-order SSTC algorithm functions as predicted. The U-D RLS estimator yields suboptimal, yet reasonable parameters when operating on a plant in closed-loop, even though the input is not persistently exciting, thus allowing the SSTC controller to generate a control law that performs acceptably. Covariance resetting also prevents the covariance matrix from generating an overflow condition in the computations. Although a forgetting factor λ of 0.95 produced excellent results in the open-loop trials run previously, it seems to cause the estimated parameters to be overly sensitive when the only source of excitation to the plant is an occasional change in the reference input.

To test this hypothesis, the forgetting factor is set to 0.99 and the simulation is run again with all of the other parameters remaining the same as before. The resulting plant and controller outputs are shown in Figure 73.

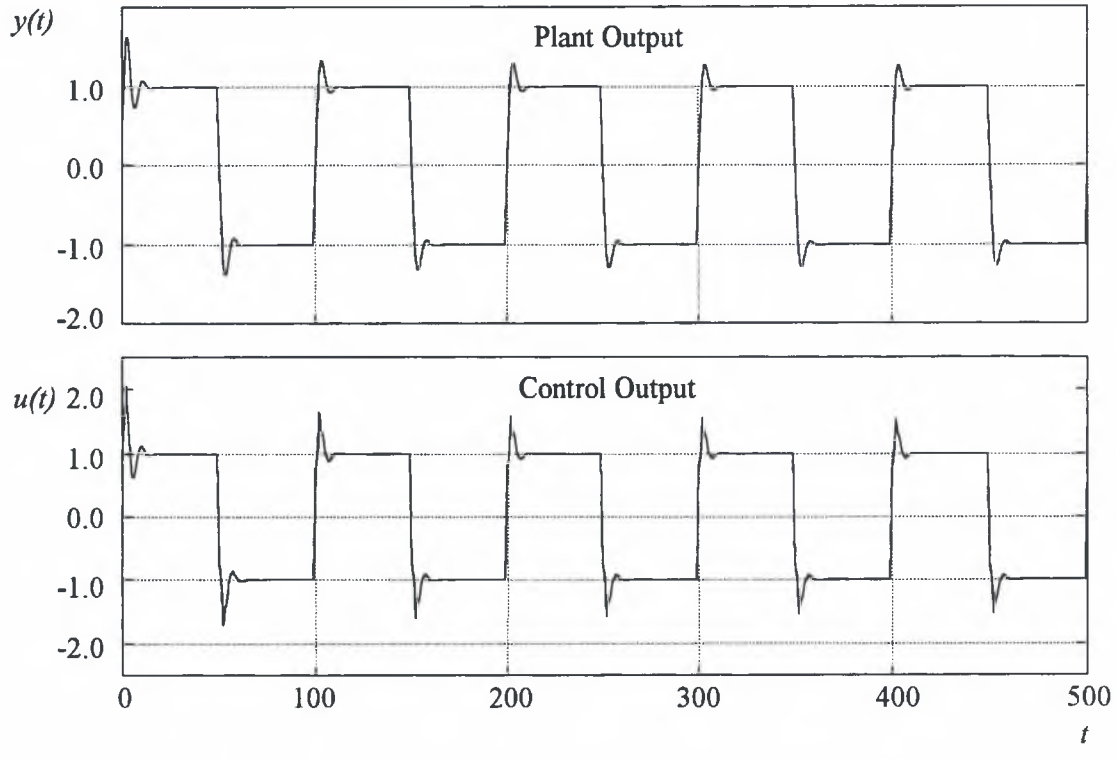

Figure 73. Plant and control outputs of second-order SSTC controller with $\lambda = 0.99$

The plant output with $\lambda = 0.99$ is more stable than the output with $\lambda = 0.95$; however, the simulation with $\lambda = 0.95$ produces a better step response, at least until the parameters diverge. Step responses obtained with $\lambda = 0.99$ and $\lambda = 0.95$ are shown in Figures 74 and 75, respectively.

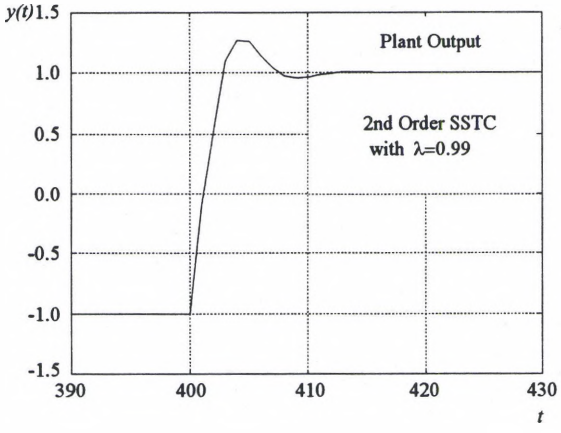

Figure 74. 2nd-order SSTC step with $\lambda = 0.99$

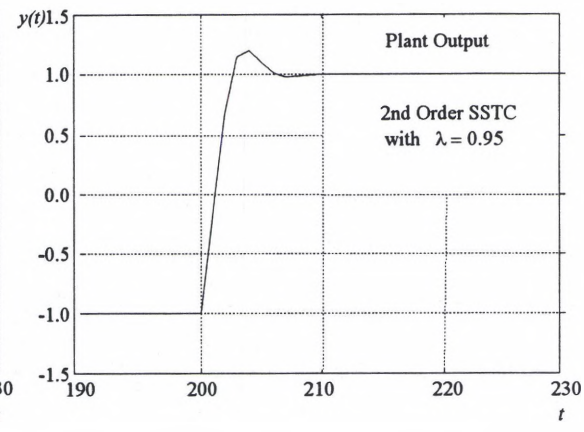

Figure 75. 2nd-order SSTC step with $\lambda = 0.95$

The overshoot and the settling time in Figure 75 is less than that of Figure 74. On the other hand, the response with $\lambda = 0.95$ begins to degrade in the next cycle after the response depicted in Figure 75, whereas, after the tuning-in period, the response with $\lambda = 0.99$ remains virtually unchanged for the full 500 samples.

A graph of the estimated parameters for the simulation with $\lambda = 0.99$ is shown in Figure 76. The averaging effect of the larger forgetting factor is evident in the figure. Although the parameter jumps at the changes in the reference input are still evident, they are much smaller and the parameters are significantly more stable than with $\lambda = 0.95$. The averaging effect of the larger forgetting factor, however, somewhat limits the accuracy of the estimates.

In the absence of a persistently exciting input to the plant, the estimated parameters have been shown to be prone to drift. The value of the forgetting factor that produces the best results in the open-loop trials is less suitable when the estimation is performed on a closed-loop plant assuming a second-order plant model. The parameters do converge rapidly, but then drift as the level of excitation to the plant input decreases as the control output stabilizes.

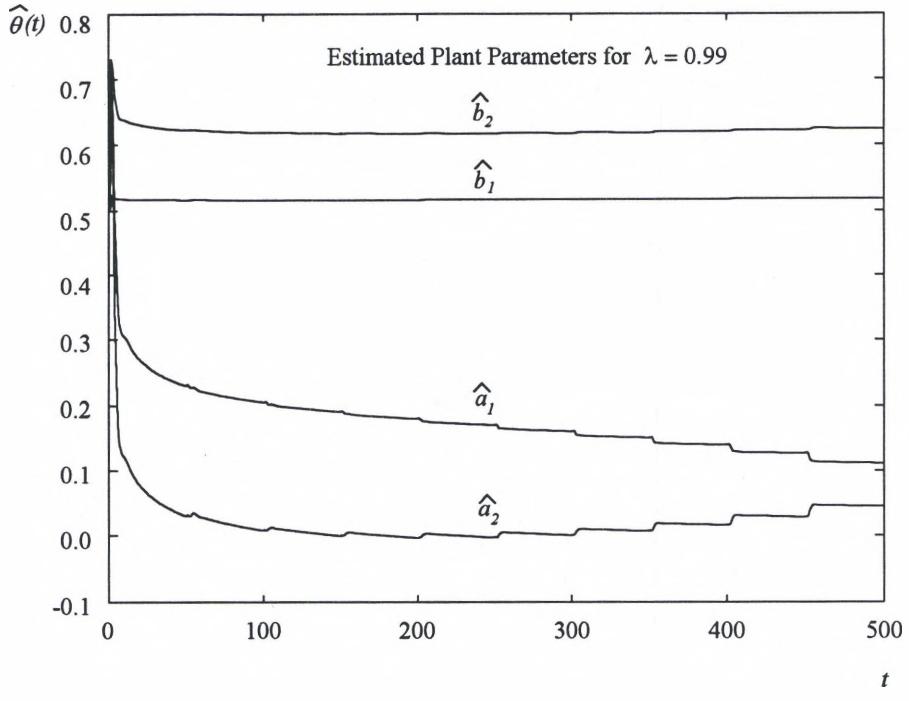

Figure 76. Estimated parameters for 2nd-order SSTC with $\lambda = 0.99$

To investigate the effect of the model order of the estimated plant on the performance of the controller, several trials are run assuming higher order plant models. All other parameters are kept the same as the previous tests. The results of the simulations are shown in Figures 77 through 80. In each of the figures, the output remains stable for the duration of the simulation. As the assumed model order is increased, the overshoot decreases and the settling time increases as the output becomes more damped. This is seen more clearly in Figure 81, which shows an enlarged view of the outputs of the simulation taken at $t = 200$.

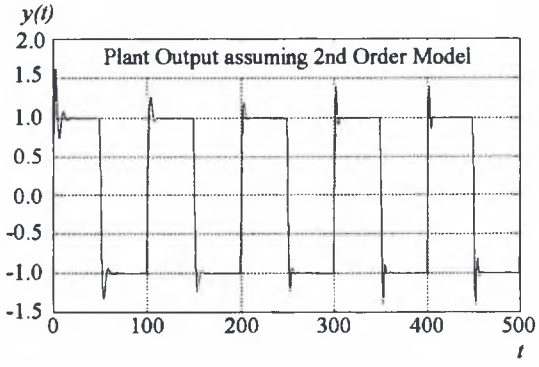

Figure 77. Plant output for 2nd-order model

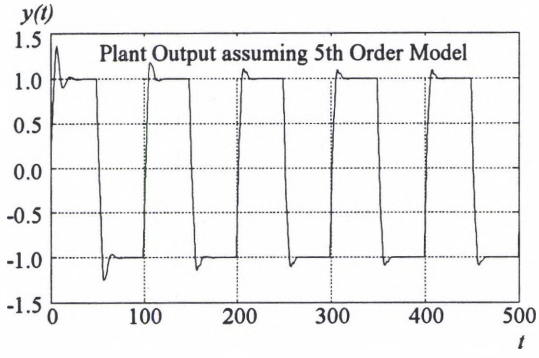

Figure 78. Plant output for 5th-order model

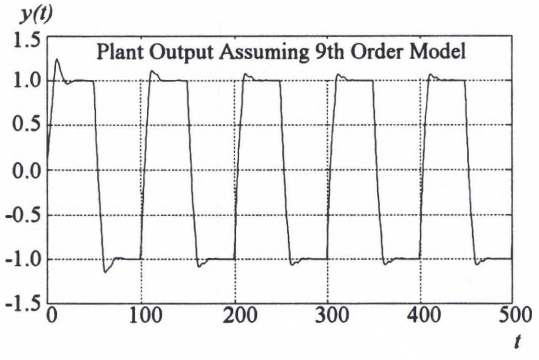

Figure 79. Plant output for 9th-order model

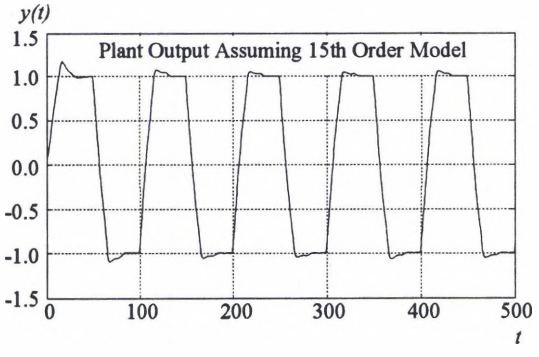

Figure 80. Plant output for 15th-order model

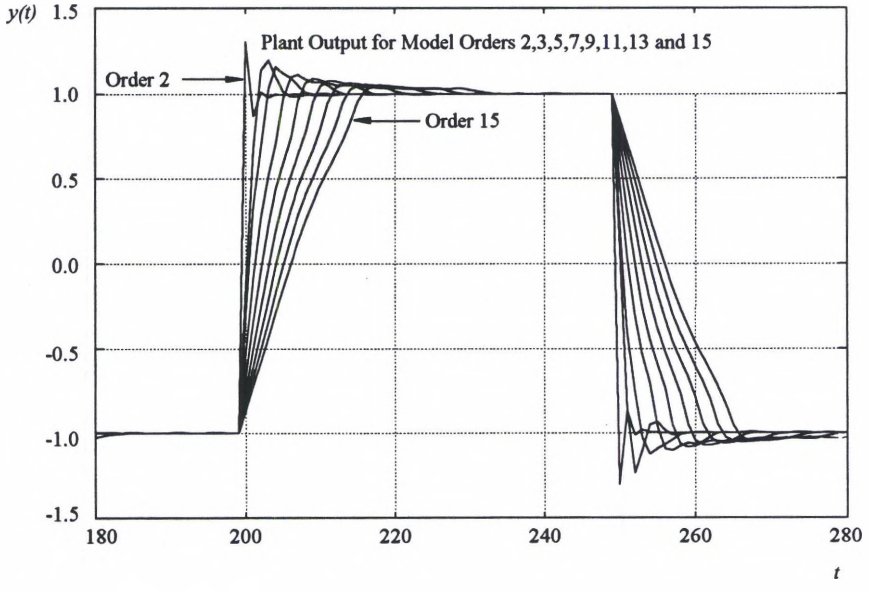

Figure 81. Effect of higher model orders on output response

Figure 81 reveals how increasing the assumed model order affects, and can even improve, the performance of the adaptive system. It also indicates that a higher order model is not a substitute for a persistently exciting input signal. Increasing the model order can improve system performance to a degree. However, too high a model order requires extra computational effort for little or no improvement in performance, and can actually adversely affect system performance in closed-loop estimation.

It is also revealing to examine the locations of the poles and zeros of the estimates of the models of different orders. The values of the parameters at $t = 200$ are selected and the locations of the poles and zeros of the true plant, the second order model and the fifteenth-order model are plotted in Figures 82, 83 and 84, respectively.

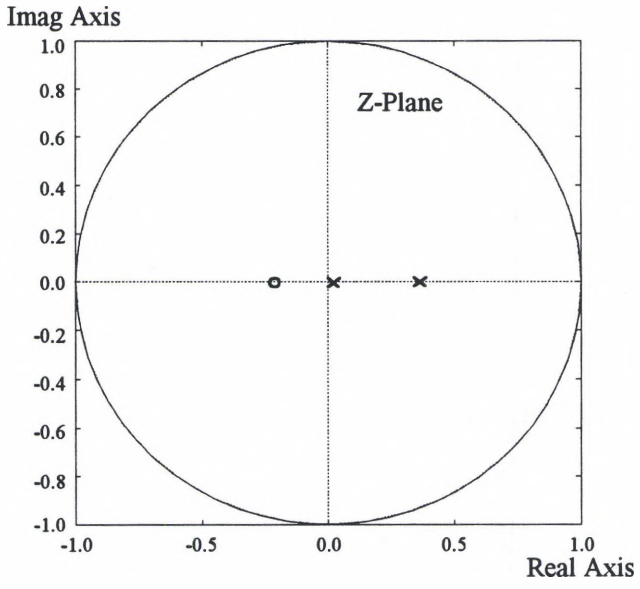

Figure 82. Pole-zero locations of true plant

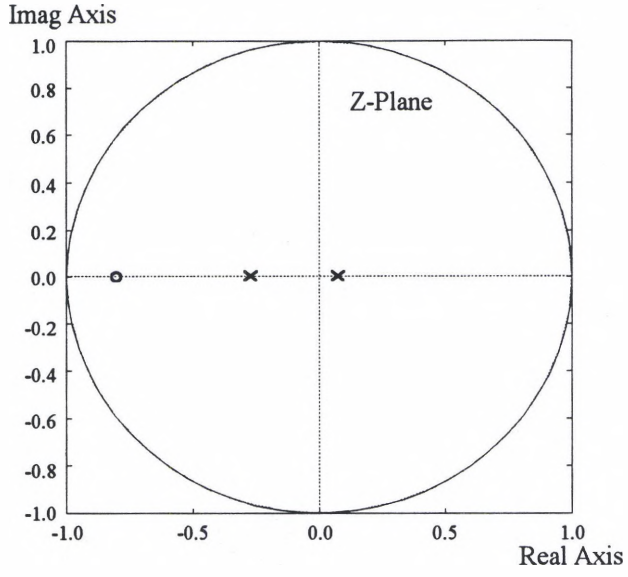

Figure 83. Pole-zero locations of second-order model at $t = 200$

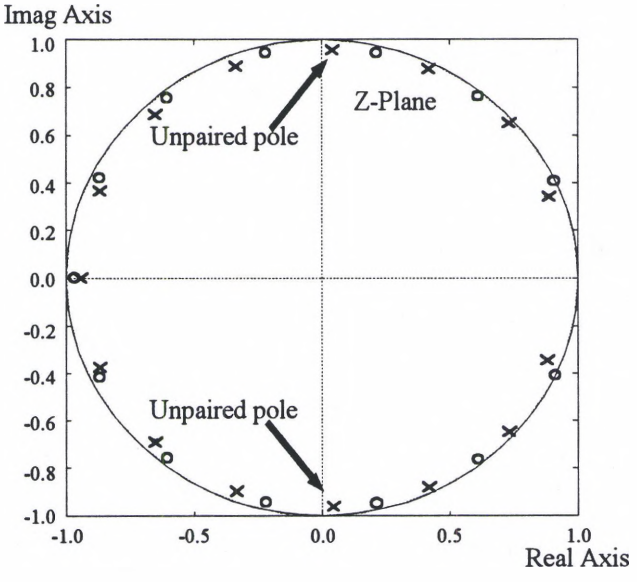

Figure 84. Pole-zero locations of fifteenth-order model at $t = 200$

The second-order estimate yields two poles and one zero on the real axis, although the estimated poles and zeros are not near the true pole and zero locations. The fifteenth-order model, on the other hand, yields poles and zeros positioned symmetrically around the unit circle, with all but two of the poles (denoted in Figure 84 with the black arrows)

being effectively canceled out by a zero. This illustrates how the U-D RLS estimator responds to overmodeling. The additional poles and zeros from the higher order model that have little or no bearing on the system response are positioned to effectively cancel out the effect of one another.

A stochastic assessment

The effect of random disturbances on the SSTC algorithm is examined in this section. The same plant model is used as in the previous simulations; however, a PRBS is added to the measurement of the plant output $y(t)$ to approximate zero-mean white noise. The simulation with $\lambda = 0.95$ is repeated here assuming plant models of orders 2, 3, 5 and 7, and noise with a variance of 0.1 has been added to the measurement signal. The output responses are plotted in Figures 85, 86, 87 and 88. The ability of the SSTC algorithm to function well in the presence of a significant amount of noise is evident in the figures. As with the deterministic case, as the model order is increased, the overshoot decreases and the settling time increases. Also, as the model order is increased, the SSTC controller compensates for the noise, as evidenced by the decreasing variability of the plant output with increasing controller model order. Figure 85 exhibits similar behavior to the noise-free simulation of the second-order model shown in Figure 77, in that, after the first two or three cycles of the reference input, the output response somewhat degrades. Examination of the parameters in Figure 89 reveals why.

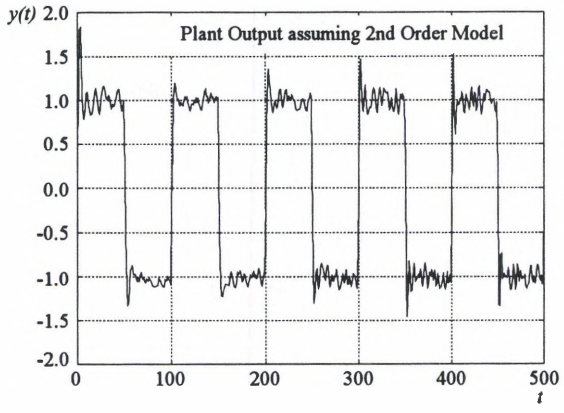

Figure 85. 2nd-Order model with noise added

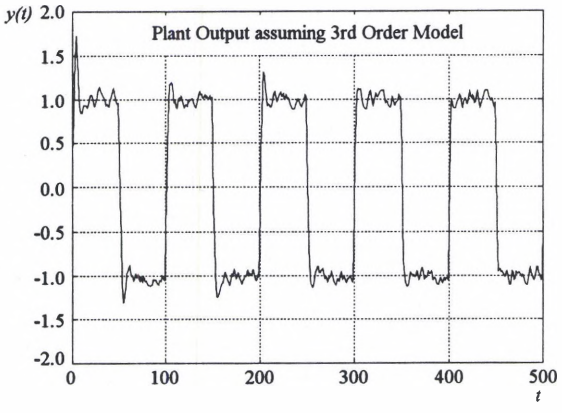

Figure 86. 3rd-Order model with noise added

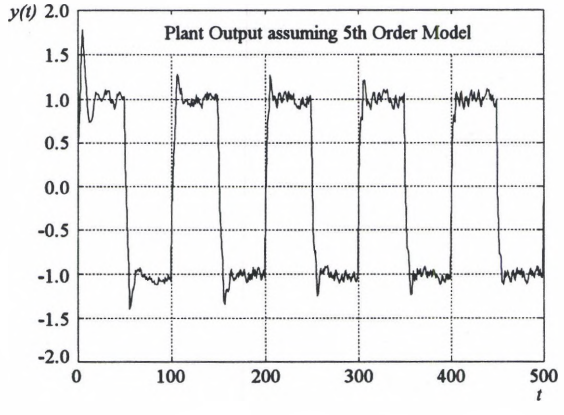

Figure 87. 5th-order model with noise added

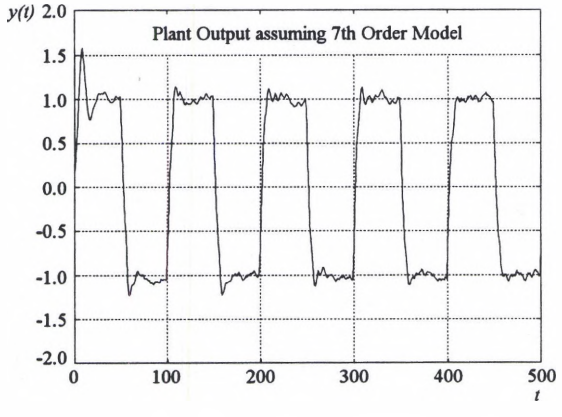

Figure 88. 7th-order model with noise added

Figure 89 is almost identical to the plot of the parameter estimates for the noise-free case in Figure 68, indicating that low level noise does not exert a major influence on the estimates. (Recall that the plant input and output data are pre-filtered through a low pass filter.) For comparison purposes, the parameters of the third-order model are presented in Figure 90. The estimated parameters still make sudden jumps in response to the changes in the reference input, but they are not nearly as severe as with the second-order model.

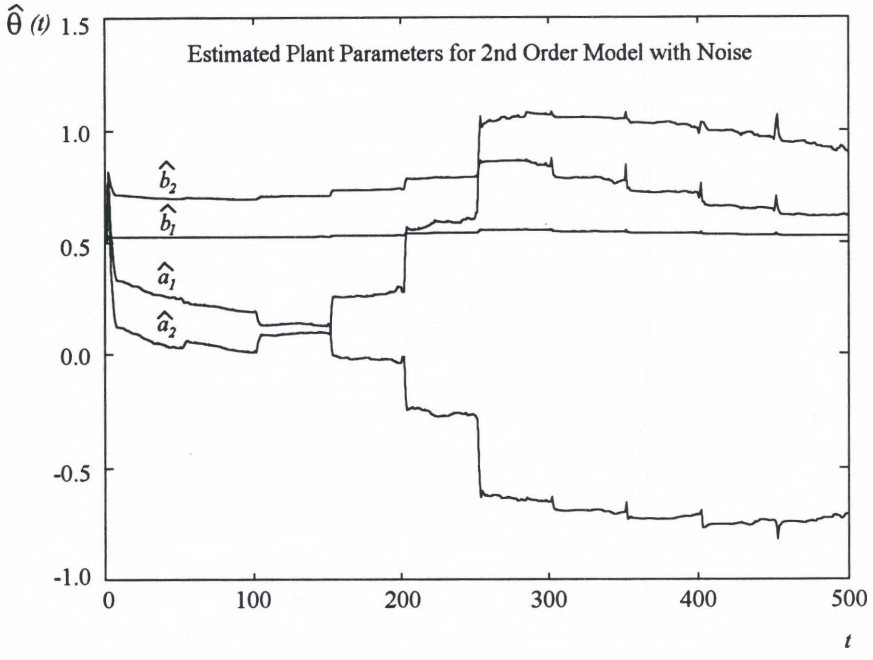

Figure 89. Estimated parameters for 2nd-order model with noise added

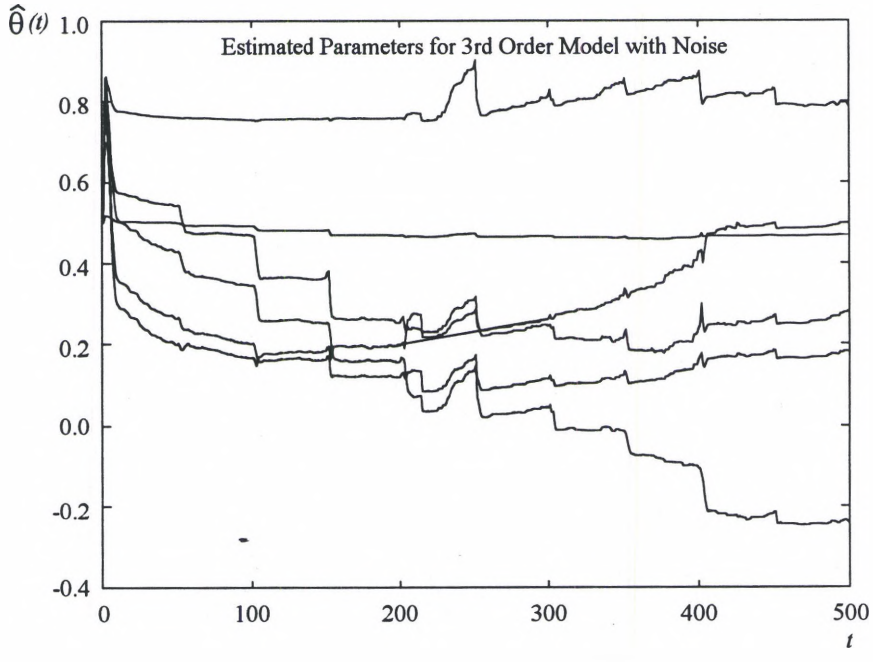

Figure 90. Estimated parameters for 3rd-order model with noise added

The SSTC algorithm has been shown to perform well in the presence of low level noise. In fact, with the estimator data being pre-filtered, the noise seems to make little

impact on the controller performance. The simulations have been of fairly short duration, however. The algorithm is now tested for an extended time period. A second-order model is assumed for the simulation and zero-mean noise with a variance of 0.1 is added to the output measurement. λ is set at 0.95 and the other parameters are set to be the same as in the previous simulations. (Due to limitations of the plotting package, only every tenth sample is stored. The transient responses of the output are therefore not accurately portrayed in the graph.) The results of the simulation are shown in Figure 91.

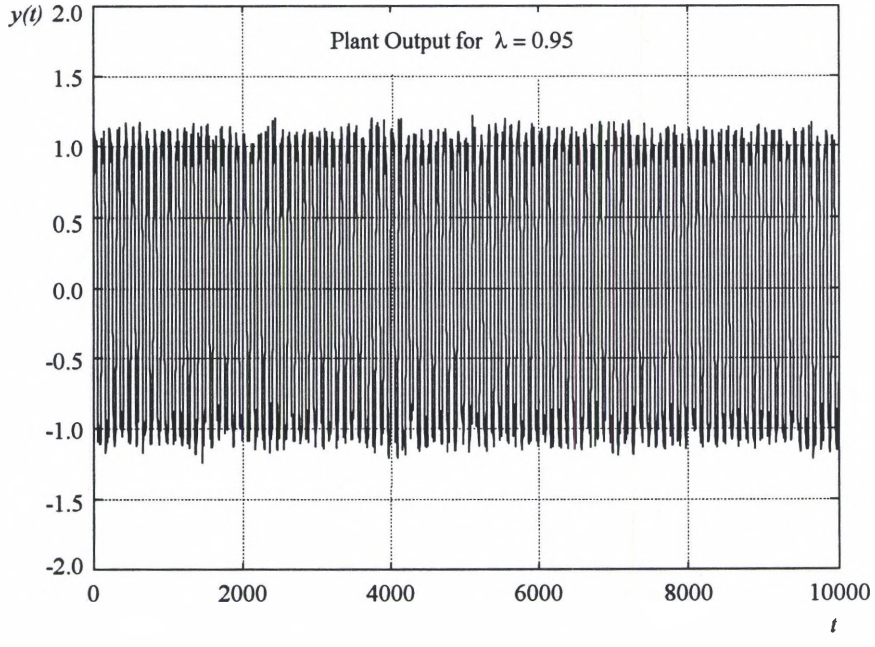

Figure 91. Extended simulation of plant output for $\lambda = 0.95$

The output continues to remain stable even after 10,000 samples with the reference input changing signs every 50 samples. A graph of the estimated parameters for the test is given in Figure 92.

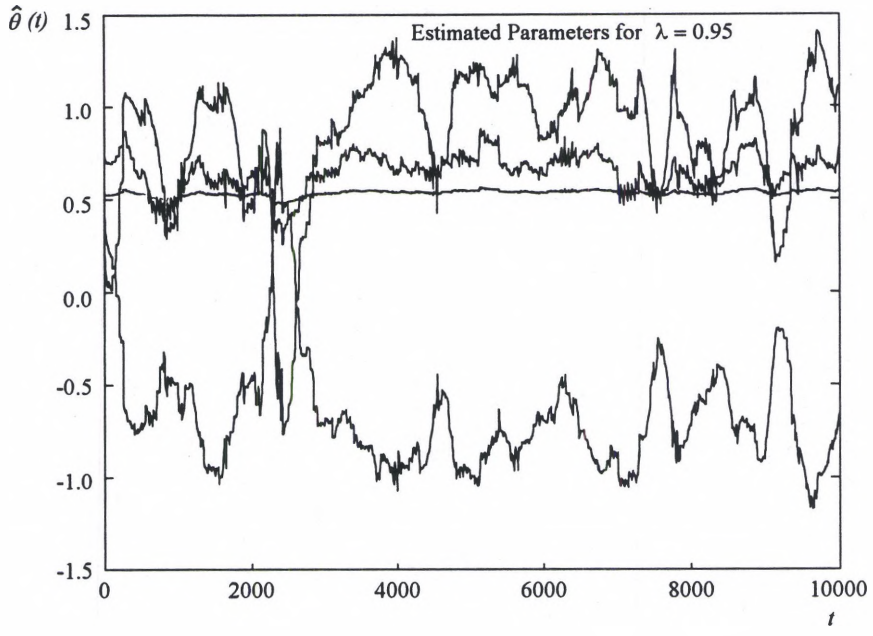

Figure 92. Extended simulation of parameter estimates for $\lambda = 0.95$

With the forgetting factor set at 0.95, the parameter estimates behave erratically for the entire simulation period; however, the output remains stable and under reasonable control despite the parameter variations.

The same simulation is rerun with $\lambda = 0.99$. The results of the simulation are shown in Figure 93. As in the previous case, the output remains stable. The estimated parameters are plotted in Figure 94. The parameter estimates are much less erratic than those with $\lambda = 0.95$. Although they fluctuate throughout the simulation, the estimates remain bounded and appear to have converged around some mean values. Comparing Figure 94 with the noise-free case of Figure 76, the presence of noise destabilizes the parameter estimates significantly.

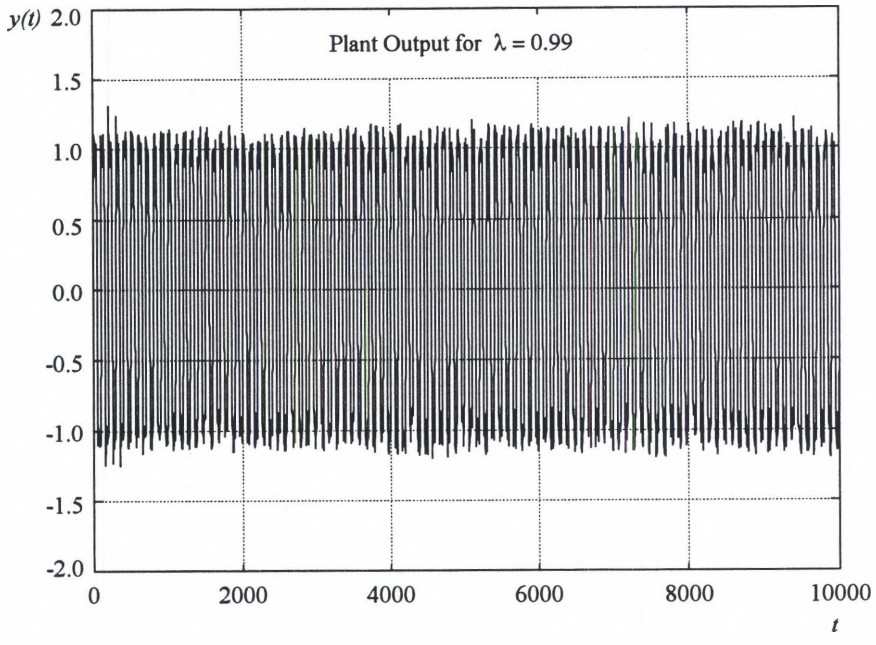

Figure 93. Extended simulation of plant output for $\lambda = 0.99$

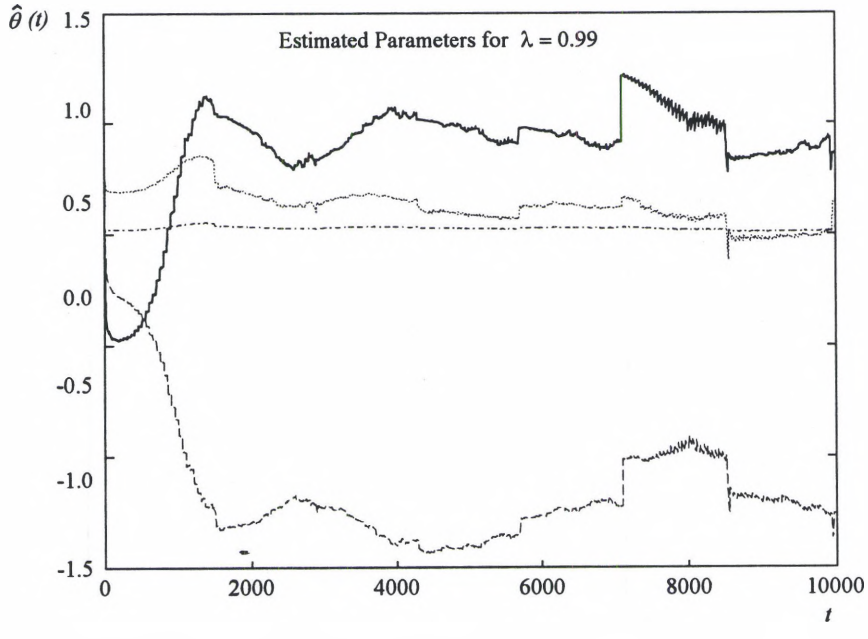

Figure 94. Extended simulation of parameter estimates for $\lambda = 0.99$

Controlling bursting with a dead-zone

For the cases investigated so far, the SSTC algorithm has performed well when the assumed model of the plant is at least as high of an order as the true plant. One of the conditions for identifiability in a closed-loop system proposed by Isermann (1982) is that the order and the deadtime of the process to be controlled be known a priori. This is necessary to ensure that the estimator is of at least as high of an order as the process to be controlled. The case where the model is of a lower order than the actual plant is examined in this section. The estimator attempts to identify a second-order plant assuming the first-order model:

$$G_m(q^{-1}) = \frac{b_1 q^{-1}}{1 + a_1 q^{-1}}.$$

For the simulation, the forgetting factor λ is set to 0.95 and all of the other parameters are left the same as in the previous simulations. The resultant output of the plant is shown in Figure 95.

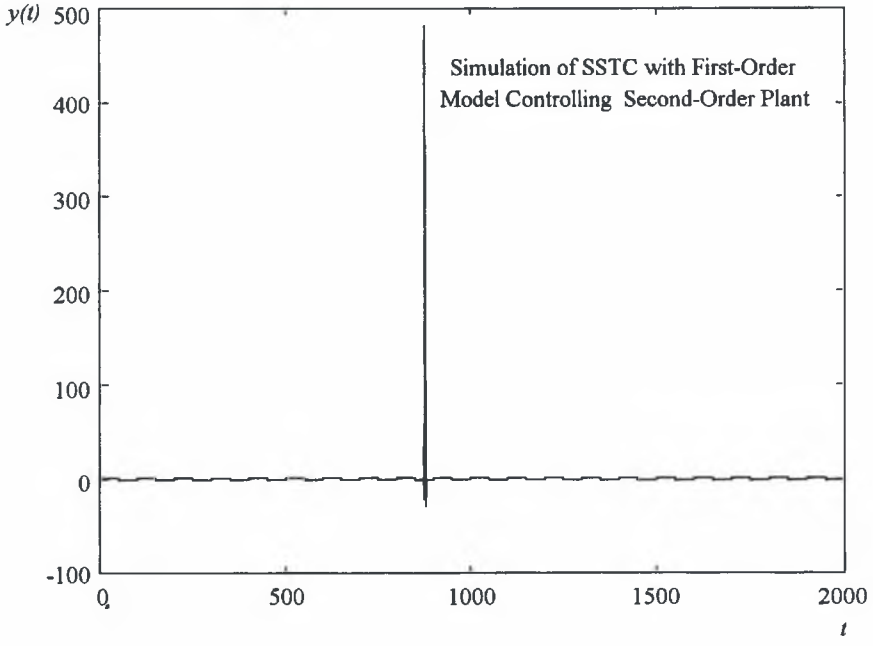

Figure 95. Simulation of first-order SSTC controlling second-order plant

The output of Figure 95 is an example of *bursting*. The same graph is shown at a smaller scale in Figure 96.

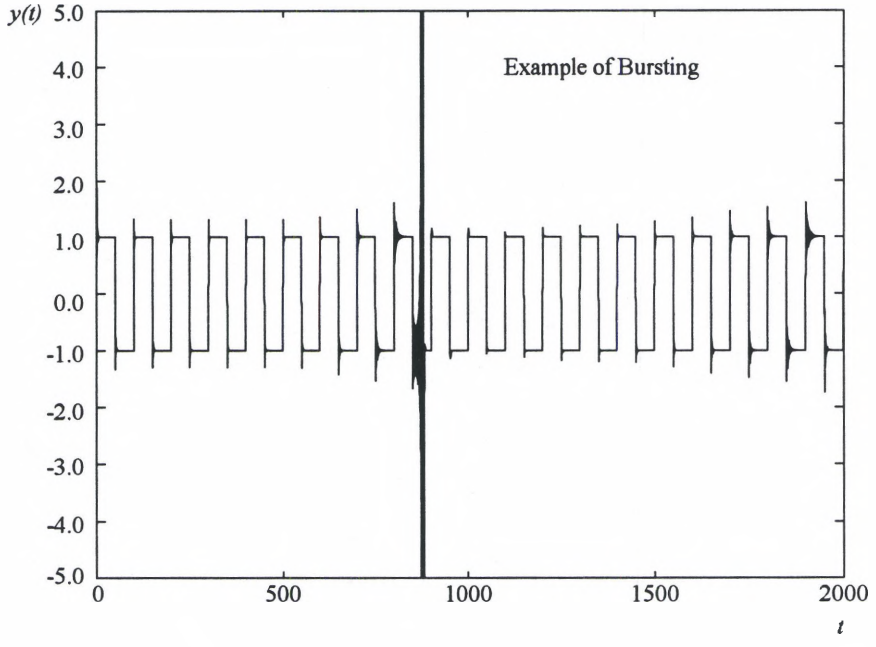

Figure 96. Example of bursting with first-order SSTC controller

Although the plant output rapidly achieves a reasonable response, after a few cycles of the reference input, the response degrades as the parameters drift, as shown in Figure 97. Eventually, the estimated parameters drift past a stability threshold and the controller can no longer control the plant. The erratic output during the burst excites the plant input sufficiently to bring the parameter estimates back to reasonable levels. The plant output shown in Figure 96, however, appears to be diverging again toward the end of the simulation. Although bursting can also be caused by other factors, such as allowing $\mathbf{D}(t)$ to grow too large, the simulation demonstrates the danger of undermodeling, particularly when attempting closed-loop estimation. This illustrates the necessity for Isermann's (1982) criterion for closed-loop identification that the model order of the plant to be identified must be known a priori.

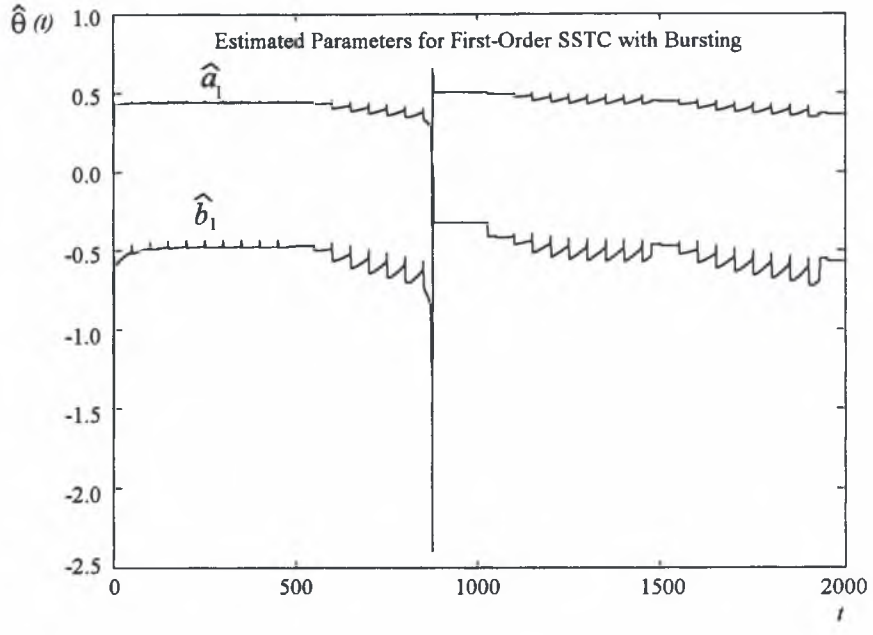

Figure 97. Estimated parameters during *bursting* with first-order SSTC controller

In Chapter 4, the incorporation of a dead-zone in the parameter estimator was proposed to deal with the problem of bursting. The dead-zone is designed to shut off the parameter estimator when the absolute value of the parameter error $\varepsilon(t)$ remains below a threshold value for a given number of samples. The estimator reactivates if $\varepsilon(t)$ increases above the threshold for another preset number of samples. The previous simulation is repeated with a dead-zone incorporated into the algorithm. For this trial, the parameter threshold is set at ± 0.001 with the estimator *shut-off time* set at 30 samples. The resultant plant output is shown in Figure 98. With the addition of the dead-zone, the estimator shuts off at $t = 540$ when $|\varepsilon(t)| < 0.001$ for 30 samples. The parameter vector $\hat{\theta}(t)$ remains constant thereafter. A magnified view of the parameter error $\varepsilon(t)$ for the simulation is shown in Figure 99.

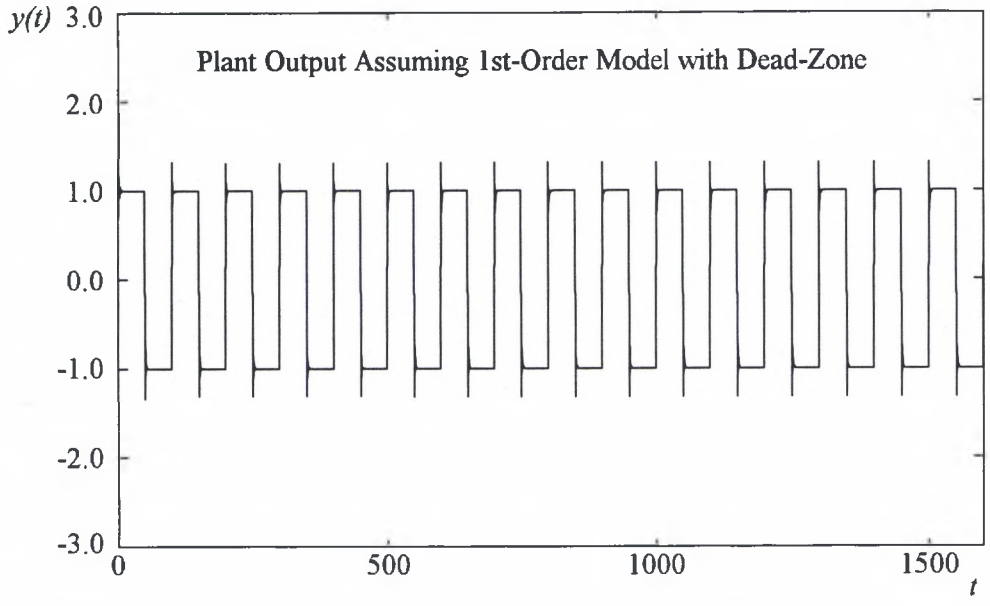

Figure 98. Plant output under control of first-order SSTC controller with dead-zone

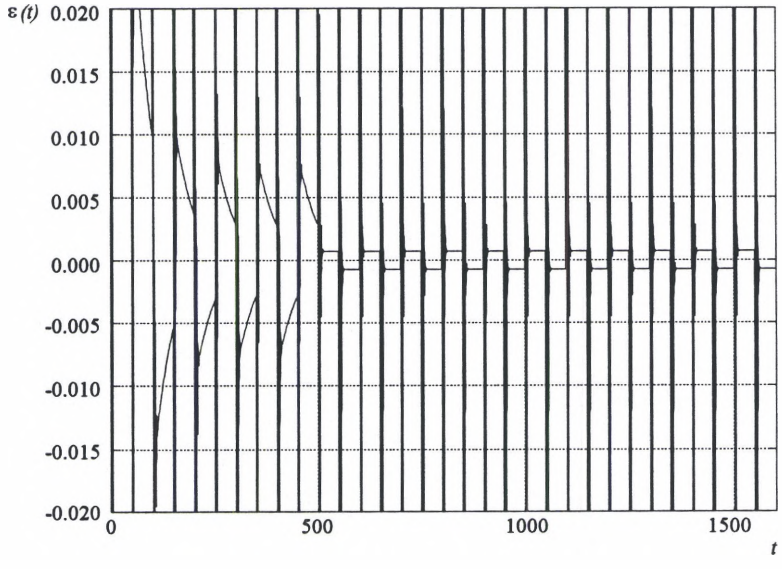

Figure 99. Magnified view of parameter error with dead-zone enabled

The parameter error $\varepsilon(t)$ fell below the threshold of 0.001 after the transients decayed from the change in the reference input at $t = 500$. After the estimator shuts off at $t = 540$, the steady-state parameter error remains below the threshold of ± 0.001 for as long as the dead zone inhibits the operation of the parameter estimator.

5.5 Simulation of the SSTC PID Algorithm

A time-invariant plant assessment

It was noted in Chapter 4 that the SSTC control algorithm could be modified to conform to a PID-like structure. The control law for the PID version of the SSTC algorithm was given as:

$$u(t) = u(t-1) + \frac{\left(1 + a_1 q^{-1} + a_2 q^{-2}\right)\left[w(t) - y(t)\right]}{b_0}$$

which can be expressed as:

$$u(t) = u(t-1) + \frac{e(t) + a_1 e(t-1) + a_2 e(t-2)}{b_0}. \tag{5.8}$$

From equation (5.8), it can be seen that three parameters must be estimated for the SSTC PID algorithm. The SSTC simulation program is modified to accommodate the change in the estimation algorithm and the revised control law of equation (5.8). A simulation is run to test the performance of the adaptive PID algorithm. To meet the model order criterion for identification in a closed-loop, the true plant is modified to be the same order as the PID estimator. The model of the true plant is described as:

$$G_p(q^{-1}) = \frac{0.6186173 q^{-1}}{1 - 0.3892412 q^{-1} + 0.0078585 q^{-2}}.$$

In the simulation, the forgetting factor is set to $\lambda = 0.95$ and the remaining estimator parameters are set the same as in the previous SSTC simulations. The resultant plant output is shown in Figure 100.

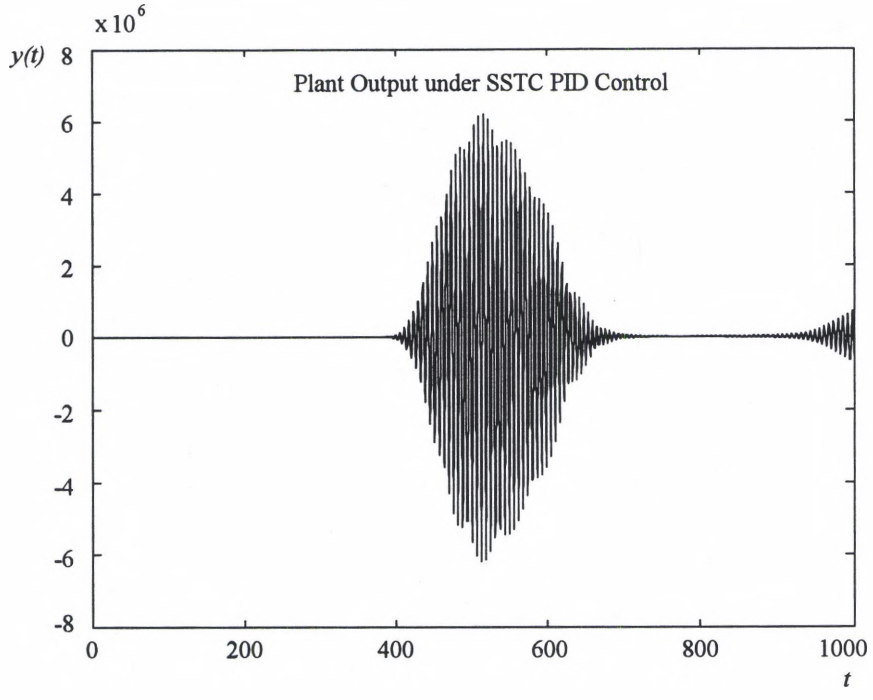

Figure 100. Plant output under SSTC PID control

The results of the first SSTC PID controller simulation are not encouraging. If the simulation is continued past 1000 samples, the output becomes unstable. The estimated parameters for the simulation are shown in Figure 101. The parameters exhibit extremely erratic behavior throughout the simulation. An investigation into the problem revealed that the instability was caused by the initial values of the elements of the parameter vector, $\hat{\theta}(t)$, which for all of the previous simulations were arbitrarily fixed at 0.5. The control law expressed in equation (5.8) contains a single term b_0 in the denominator of the term on the right hand side of the equation. Setting $b_0(0)$ to a value less than one causes the control law to drive the plant into instability, at least in this particular case. (This may not be true for other plants.)

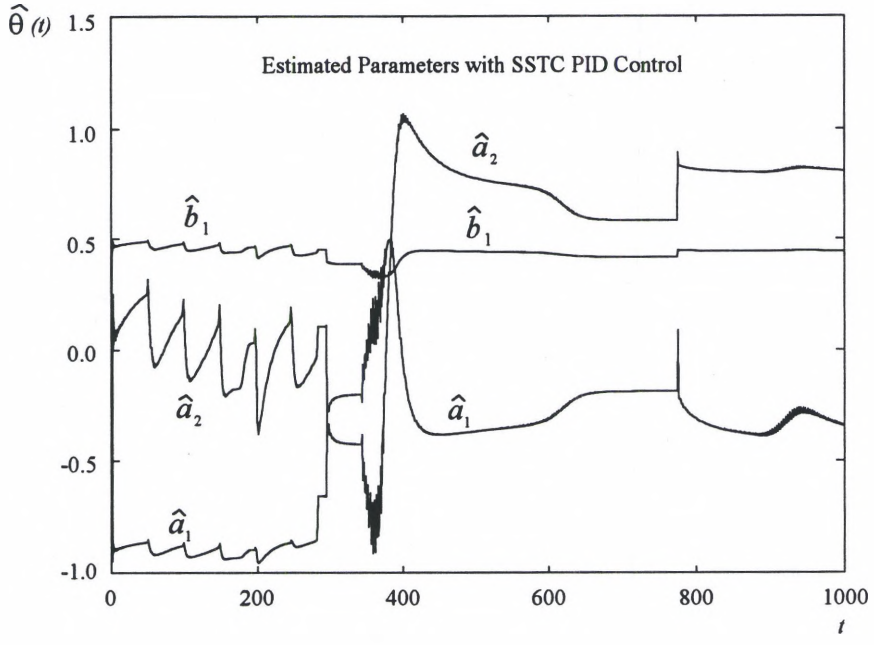

Figure 101. Estimated parameters under SSTC PID control

The initial parameter estimates are changed to 2.0 and the simulation is repeated. The output of the plant is displayed in Figure 102. The output response is considerably improved. The controller is able to bring the plant from a highly unstable initial state to a reasonable level of control in a relatively short time. Although the output is stable after the parameters have converged, it remains significantly underdamped. A third trial is thus performed with $\hat{\theta}(0) = 1.0$. The resultant plant and control outputs are shown in Figure 103.

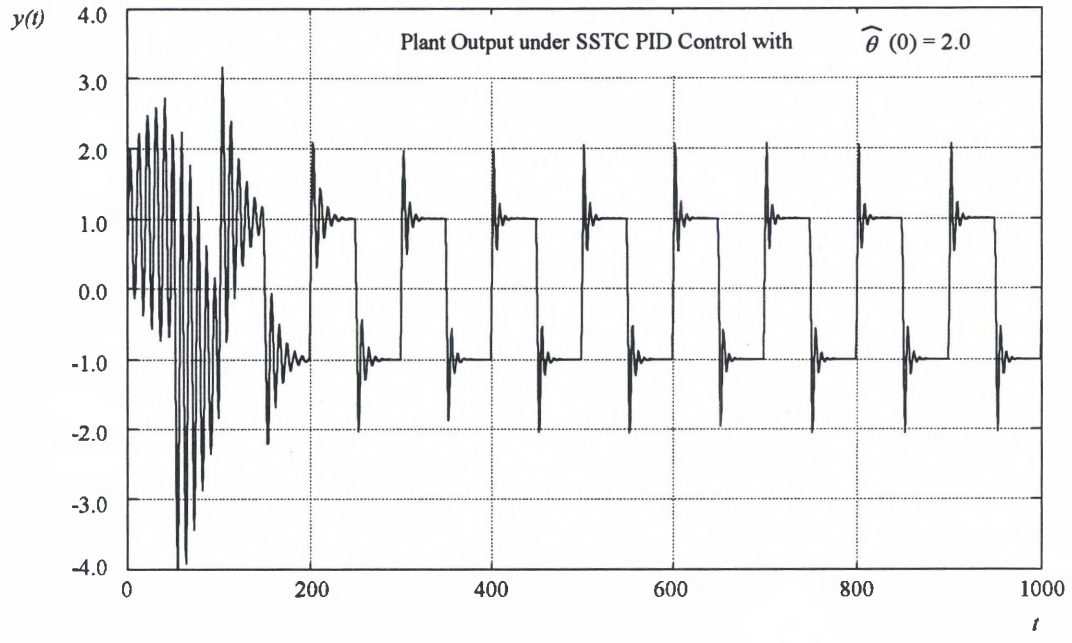

Figure 102. Plant output under SSTC PID control with $\hat{\theta}(0) = 2.0$

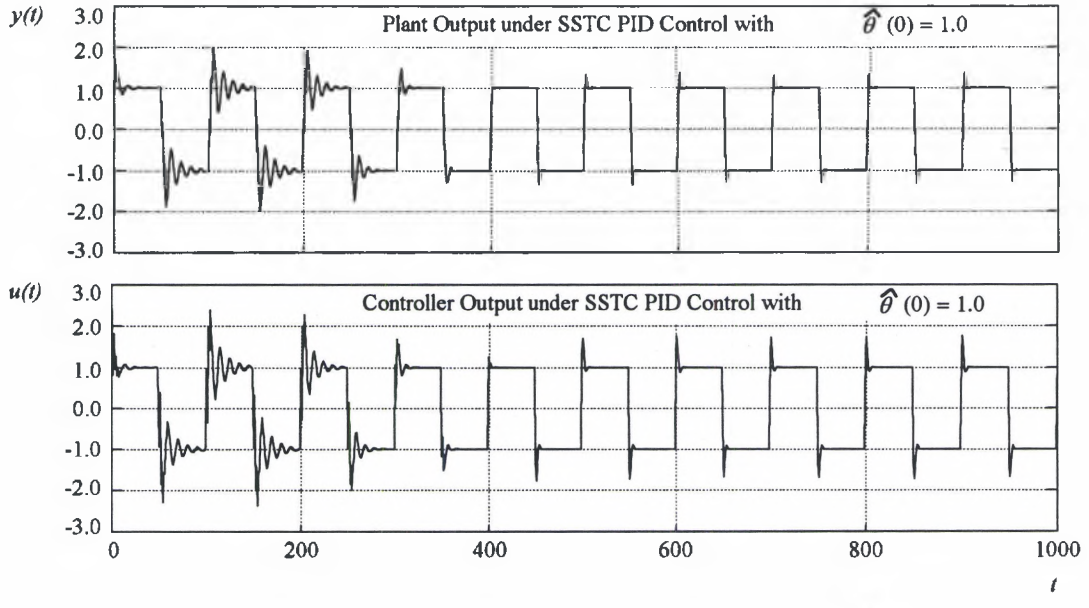

Figure 103. Plant and control outputs under SSTC PID control with $\hat{\theta}(0) = 1.0$

With $\hat{\theta}(0) = 1.0$, the output adapts within four or five cycles of the reference input. The tuned response has an overshoot of less than three percent and a settling time between five and eight seconds, as seen in the magnified view of Figure 104.

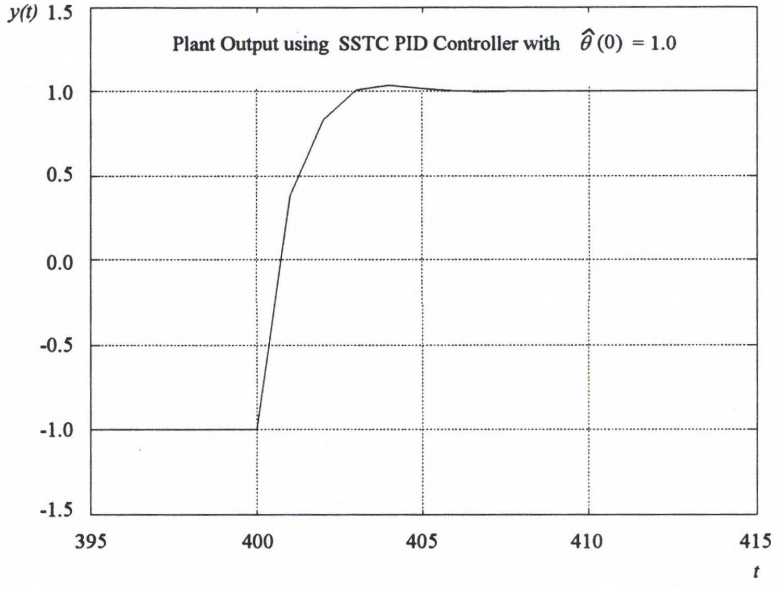

Figure 104. *Tuned* step response of SSTC PID control with $\hat{\theta}(0) = 1.0$

The output response degrades somewhat on the next cycle of the reference input, as seen in Figure 105.

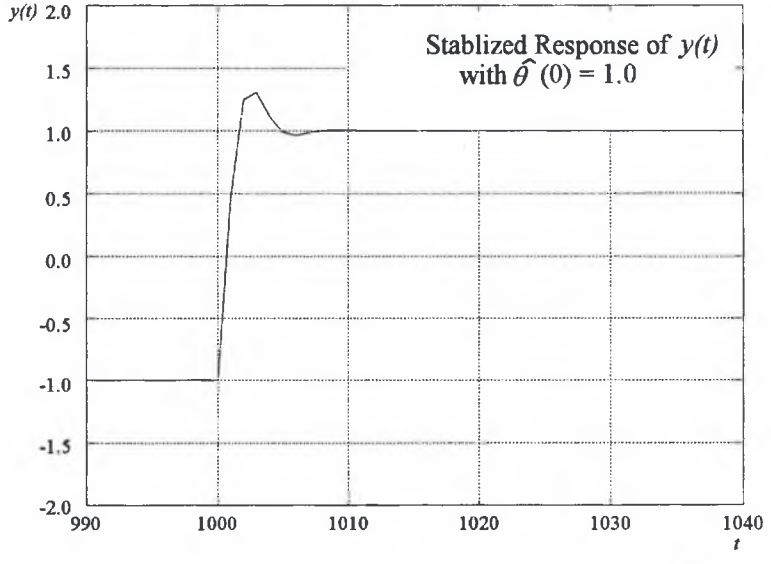

Figure 105. Stabilized step response of SSTC PID control with $\hat{\theta}(0) = 1.0$

A graph of the estimated parameters for the simulation is shown in Figure 106.

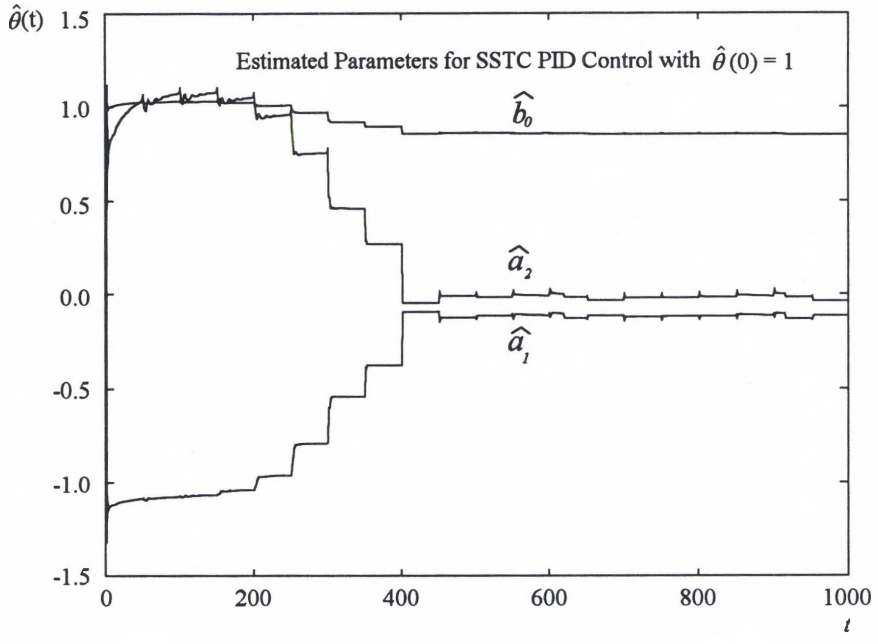

Figure 106. Parameter estimates of SSTC PID control with $\hat{\theta}(0) = 1.0$

At first glance, it would appear that the degradation of the response from Figure 104 to Figure 105 is caused by the parameters changing to less optimal values around $t = 400$. This seems to violate the nature of the RLS algorithm to cause the parameters to seek the most optimal values (i.e., those values that produce the minimal parameter error, $\varepsilon(t)$) for a given level of input excitation. Further investigation revealed, however, that the response of Figure 105 cannot be reproduced with the parameter values of either $t = 399$ or $t = 401$. Rather, the response of Figure 105 is a transient due to the rapidly changing parameter vector $\hat{\theta}(t)$ at $t = 400$ (with $\lambda = 0.95$) that occurs when the reference input cycles.

The parameter estimates shown in Figure 106 are also quite different from those of Figure 101. Their behavior is characteristic of earlier simulations using the regular SSTC algorithm. They are not erratic as in the previous simulation with $\hat{\theta}(0) = 0.5$ and they converge to stable steady-state levels. The simulations of the SSTC PID algorithm

illustrate the sensitivity of the reduced-order algorithm to the initial value of the parameter vector, $\hat{\theta}(t)$.

A time-varying plant assessment

In the previous section, the plant model was assumed to be time-invariant. In a practical sense, the SSTC PID algorithm must be tested when the true plant parameters are time-varying. To test the ability of the SSTC PID algorithm to track a linear time-varying plant, an 1800 sample simulation, where the parameters of the true plant are changed at $t = 900$, is evaluated. The following two plant models were selected to represent the true plant:

$$G_1(z^{-1}) = \frac{.5z^{-1}}{1-.4z^{-1}-.1z^{-2}} \tag{5.9a}$$

$$G_2(z^{-1}) = \frac{.7z^{-1}}{1-.5z^{-1}+.2z^{-2}}. \tag{5.9b}$$

The magnitude plots of the two models are given in Figure 107.

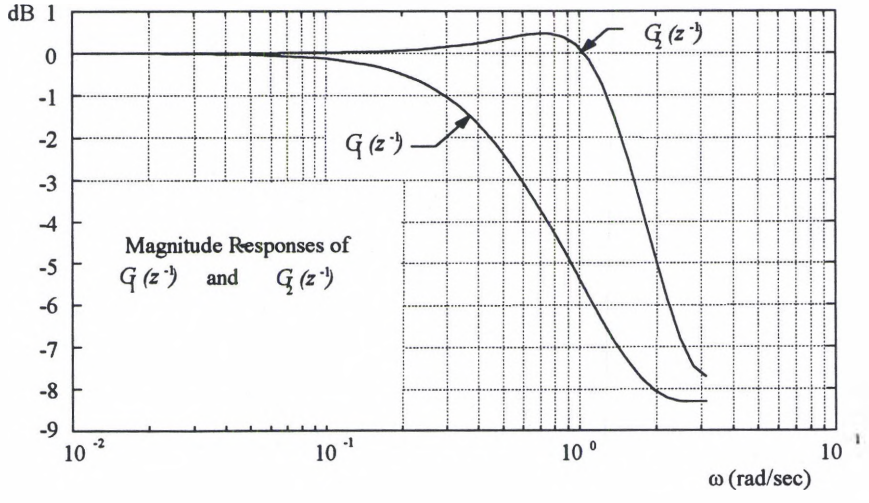

Figure 107. Magnitude responses of *true* plants $G_1(z^{-1})$ and $G_2(z^{-1})$

In the simulation, the true plant is $G_1(z^{-1})$ for $0 \le t < 900$ and $G_2(z^{-1})$ for $900 \le t \le 1800$. The forgetting factor λ is maintained at 0.95 and the other controller parameters are kept the same as in the previous simulations. The resulting plant output $y(t)$ and the controller output $u(t)$ are shown in Figures 108 and 109, respectively.

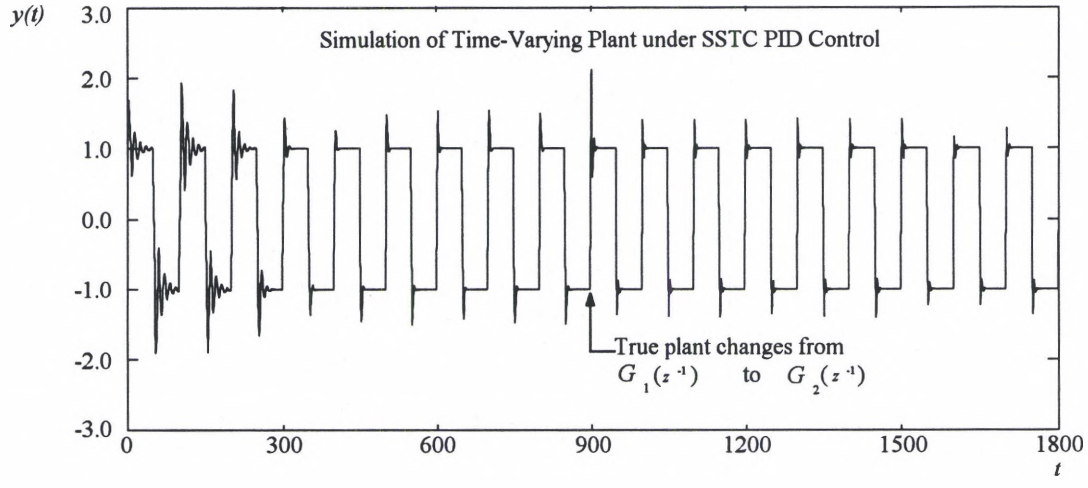

Figure 108. Output of time-varying plant under SSTC PID control

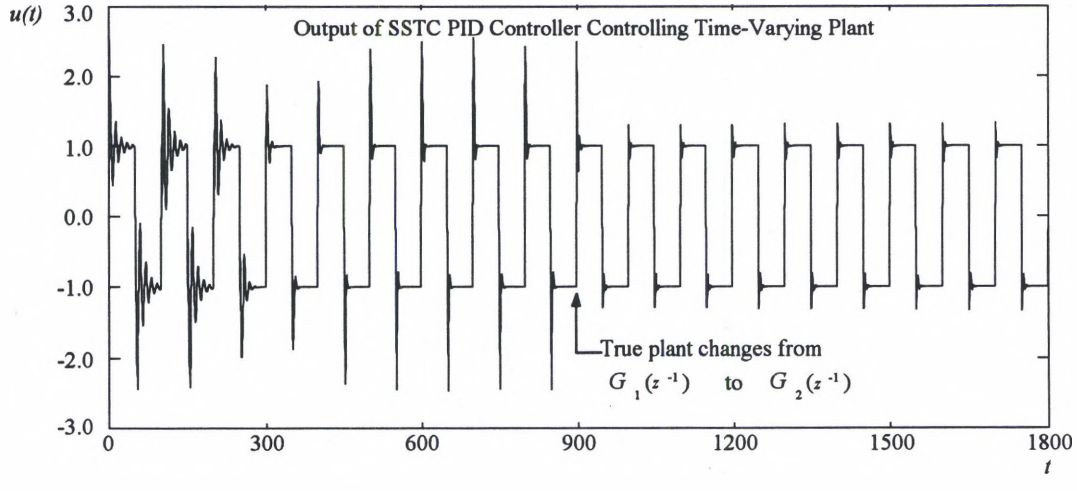

Figure 109. Output of SSTC PID controller with time-varying plant

In simulation, the controller has converged and the plant output has stabilized at about $t = 500$. When the true plant changes from $G_1(z^{-1})$ to $G_2(z^{-1})$ at $t = 900$, the plant

output reacts with a large overshoot for one cycle. By the next cycle of the reference input, the plant output is back under control. A close-up view at the point of transition between plant parameters is shown in Figure 110. The controller rapidly adapts to the change in plant parameters at $t = 900$.

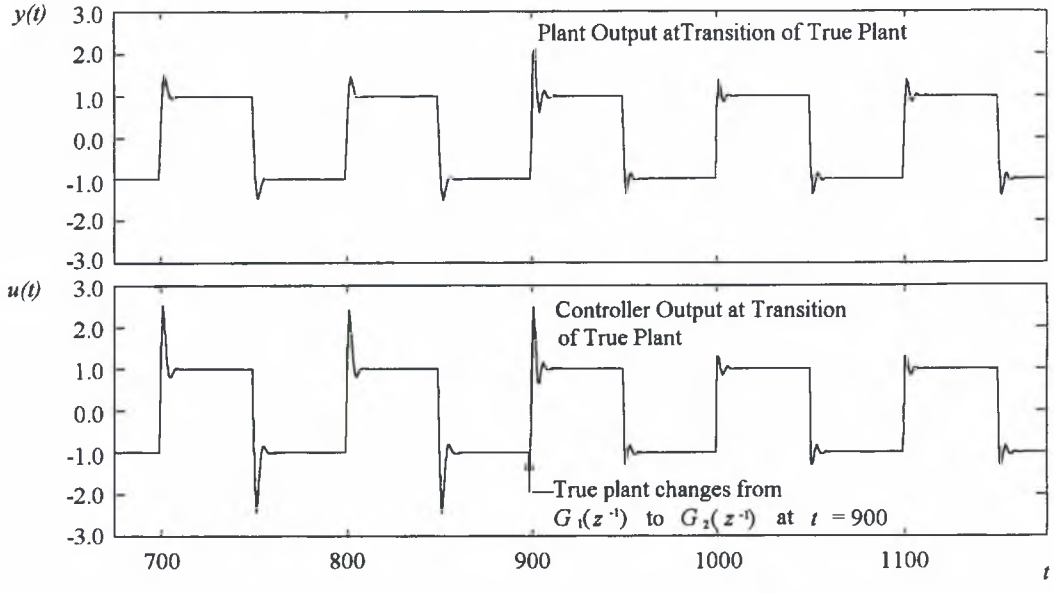

Figure 110. Plant and controller outputs at time of transition of plant parameters

The estimated parameters for the simulation are plotted in Figure 111. With the forgetting factor λ set to 0.95, the parameters converge to new values in a few samples after the plant changes. It is generally assumed that the plant parameters vary slowly in relation to the plant dynamics. In that case, the forgetting factor could be set to a larger value and still be able to track the parameter variations in the plant.

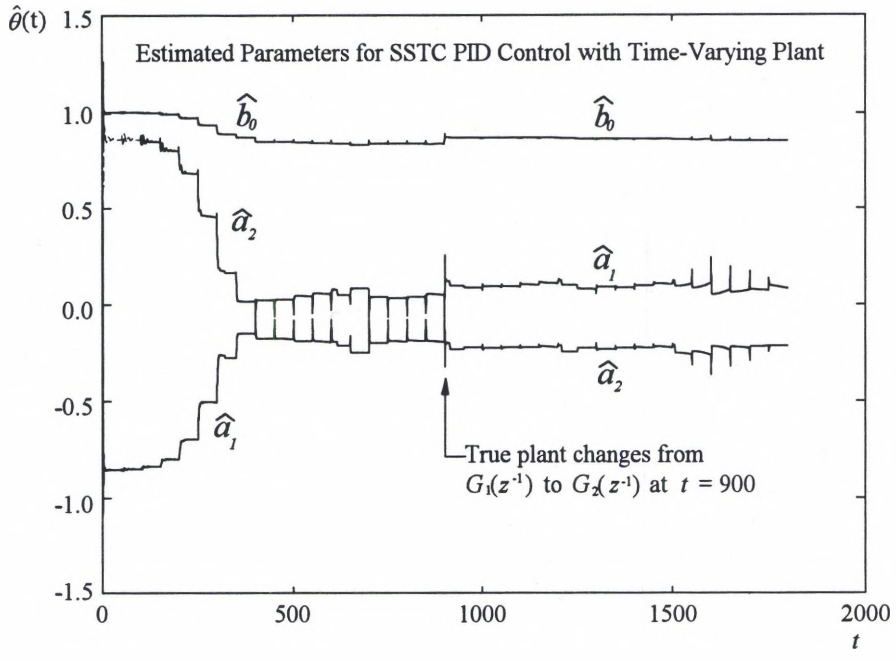

Figure 111. Estimated parameters for SSTC PID control of time-varying plant

Next, the SSTC PID algorithm is tested in the presence of noise. The previous simulation is repeated with zero-mean noise with a variance of 0.075 added to the system. The plant output resulting from the simulation is shown in Figure 112.

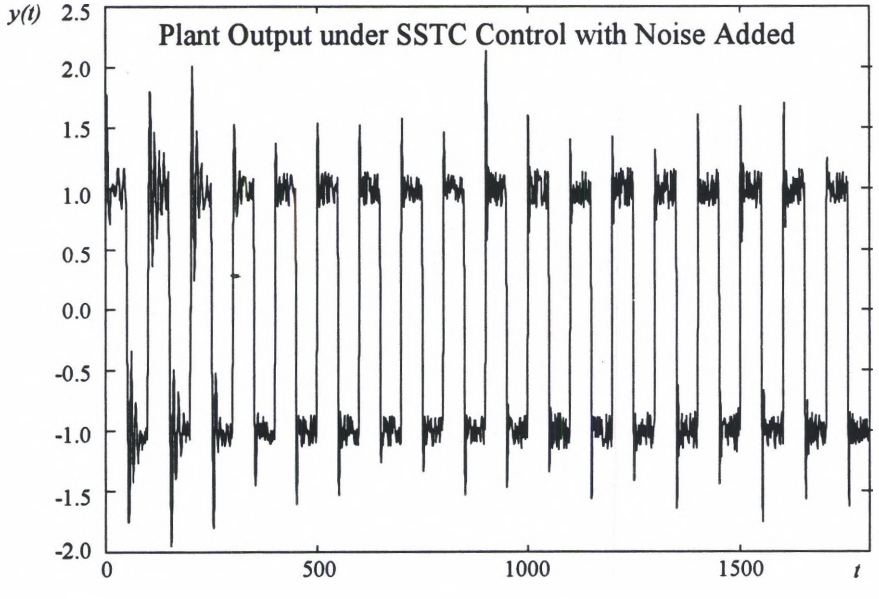

Figure 112. Simulation of SSTC PID controller with added noise

The system behaves satisfactorily at relatively low noise levels. In testing performed at higher noise levels, however, the output becomes unstable after the plant is changed from $G_1(z^{-1})$ to $G_2(z^{-1})$.

An assessment of the SSTC PID disturbance rejection capability

The simulations performed so far have focused on the servo tracking ability of the SSTC PID controller and have assumed that the reference input is changing at regular intervals. In the next (and final) set of simulations, the regulation capability of the SSTC PID controller is tested. The SSTC PID program is modified to allow the reference input to cycle a predetermined number of times in order to allow the controller to *tune* itself to the plant. The reference input is then held constant and three separate d.c. disturbances are added to the plant. The forgetting factor λ is set to 0.95 and all of the other controller parameters are set the same as the previous simulations. The plant is assumed to be time-invariant in this case and is the same model used previously, given as:

$$G_p(q^{-1}) = \frac{0.6186173q^{-1}}{1 - 0.3892412q^{-1} + 0.0078585q^{-2}}. \tag{5.10}$$

The results of the first simulation are shown in Figure 113.

The controller adapts to the plant by approximately $t = 400$ when the reference input is held constant. The controller removes the first offset of -0.25 at $t = 500$ without difficulty; however, shortly after that, the output destabilizes. When the second disturbance of +0.25 is encountered at $t = 700$, the output becomes oscillatory up through the third disturbance of -0.25 at $t = 900$, after which it stabilizes again. The parameter estimates for the simulation are shown in Figure 114.

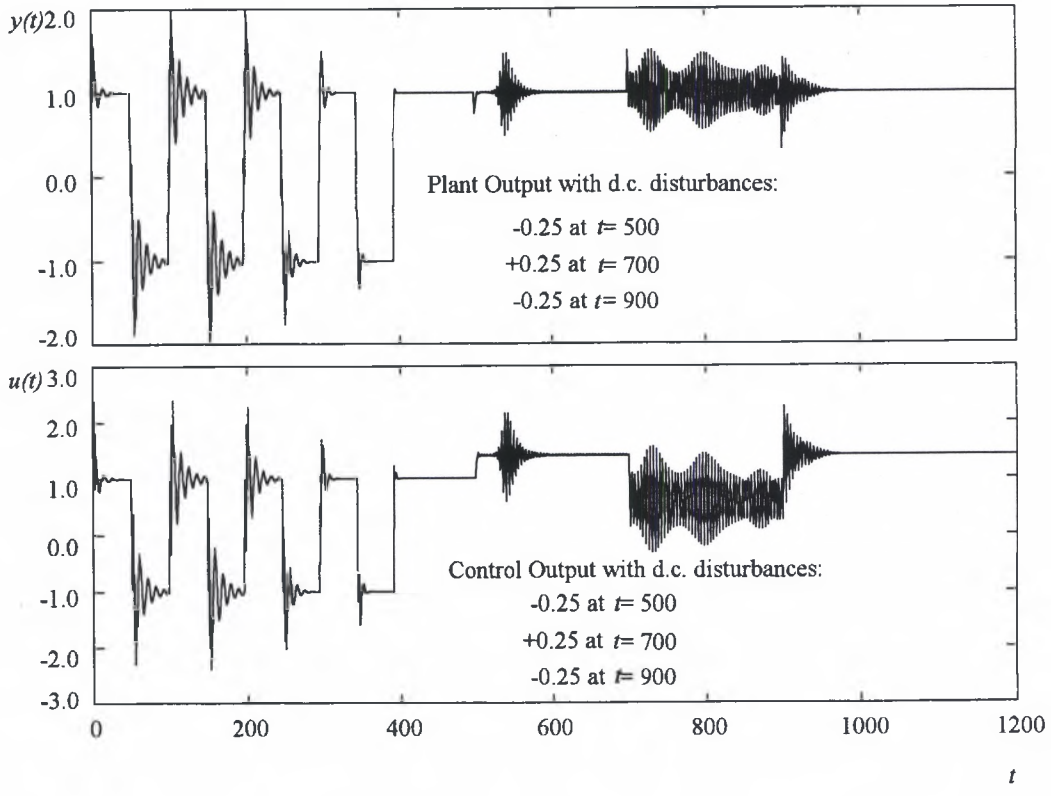

Figure 113. Disturbance rejection capability of the SSTC PID algorithm

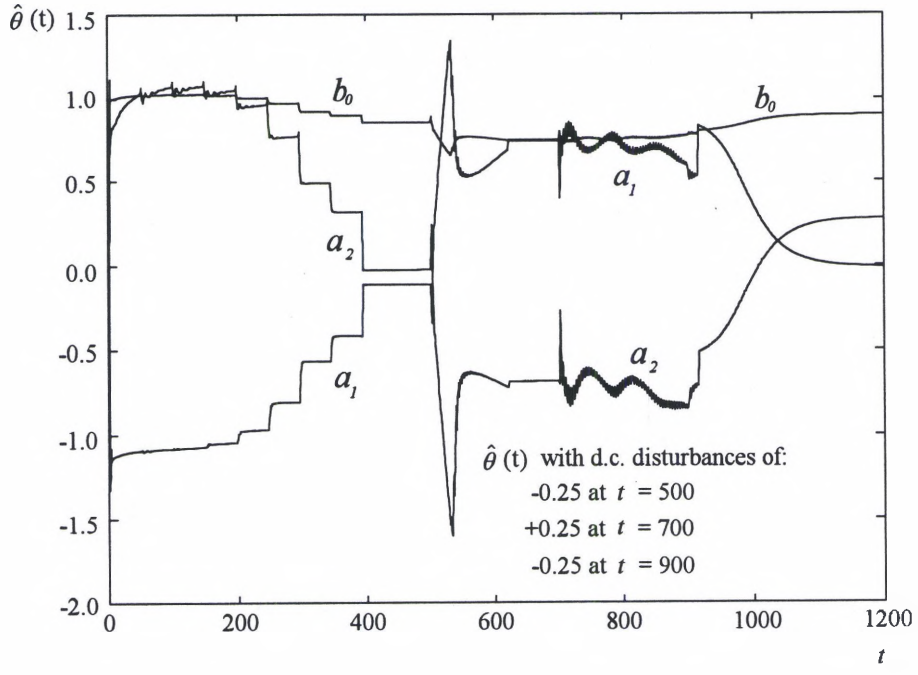

Figure 114. Response of parameter estimates to D.C. load disturbances

Figure 114 reveals why the behavior of the plant output is so erratic. When the d.c. disturbance of -0.25 is introduced at $t = 500$, the estimator is no longer receiving accurate information about the dynamics of the plant and, consequently, produces biased estimates. The biased estimates lead to errant controller parameters resulting in poor system performance. One possible solution to the problem is to incorporate a dead-zone into the estimation algorithm. The simulation just run is repeated with the dead-zone enabled. The dead-zone is adjusted so the estimator shuts off when:

$$|\varepsilon(t)| \leq 0.002 \text{ for 25 samples}$$

and the estimator reactivates if:

$$|\varepsilon(t)| > 0.002 \text{ for 250 samples}.$$

The plant and controller outputs with the dead-zone incorporated into the estimation algorithm are shown in Figure 115.

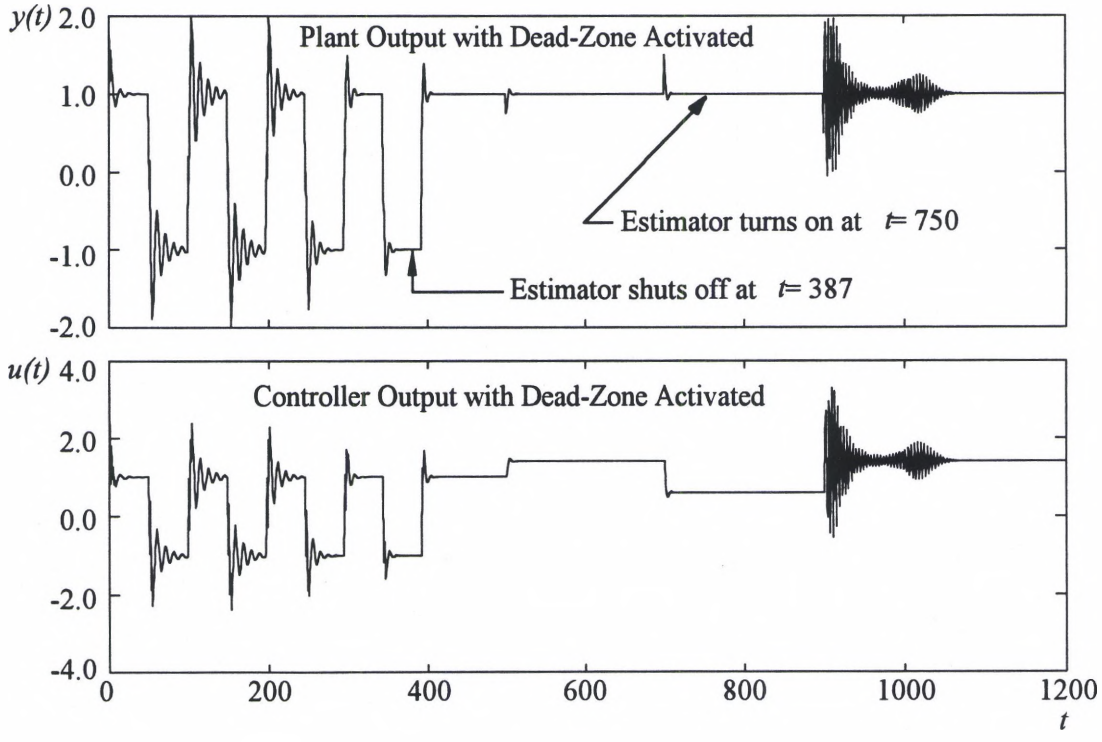

Figure 115. Plant and controller outputs with dead-zone activated

The dead zone disables the estimator at $t = 387$ when the parameter error $\varepsilon(t)$ drops below the threshold value of 0.002 for 25 samples. The controller compensates for the first disturbance of -0.25 at $t = 500$ and the second disturbance of +0.25 at $t = 700$. In order to allow the estimator to track time-varying parameters, however, the estimator must be reactivated if $\varepsilon(t)$ grows too large. As seen in Figure 115, the estimator is reactivated at $t = 750$. By the time the third offset of -0.25 is imposed on the output, the estimator has been reactivated and is producing biased estimates due to the load disturbance, resulting in the unstable output at $t = 900$.

Although the dead-zone provides a temporary solution to the disturbance rejection problem by disabling the estimator when the error is small, a disturbance on the output may produce a large enough error to reactivate the estimator. The presence of the disturbance will then lead to biased parameter estimates. In Chapter 4, it was suggested to prefilter the estimator data to remove low frequency disturbances. A first-order high pass filter is therefore combined with the low pass filter given in equation (5.5) to form a band pass filter with the transfer function:

$$H_{BP}(z^{-1}) = \frac{0.6018139 + 0.601814z^{-1} - 0.601814z^{-2} - 0.6018139z^{-3}}{1 + 0.2186251z^{-1} - 0.6121637z^{-2} - 0.3657355z^{-3}} \qquad (5.11)$$

The magnitude plot of the transfer function given in equation (5.11), along with the magnitude plot of the true plant described in equation (5.10), is shown in Figure 116. The band-pass filter rejects frequencies outside the band ranging from $\omega = 0.1$ to π rad / sec. The previous simulation is repeated with the dead-zone enabled, but this time the data to the estimator is prefiltered through $H_{BP}(z^{-1})$. The resultant plant and controller outputs are shown in Figure 117.

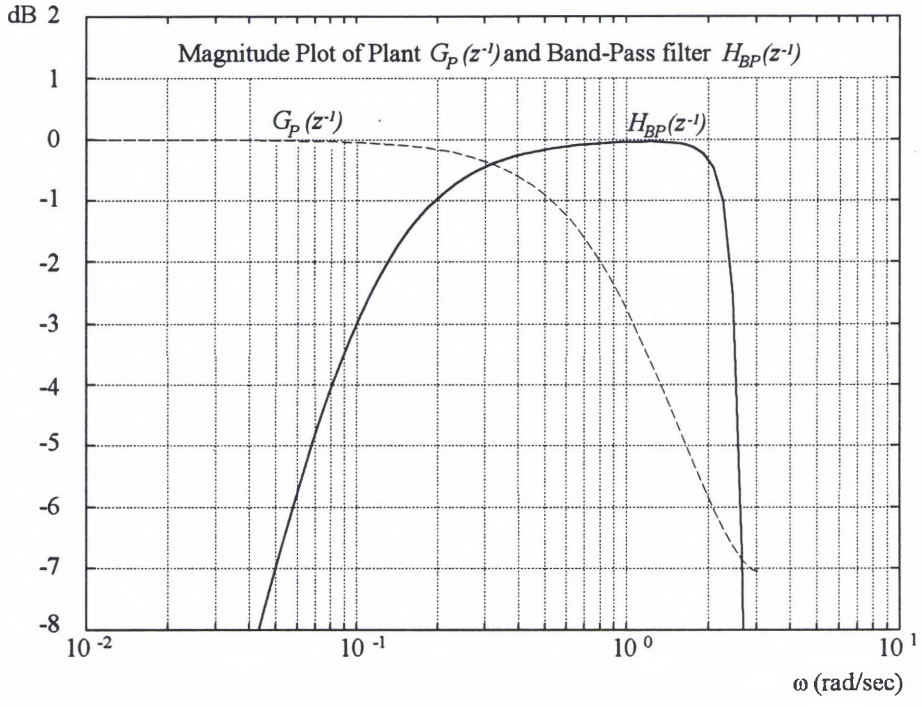

Figure 116. Magnitude responses of plant and band pass filter

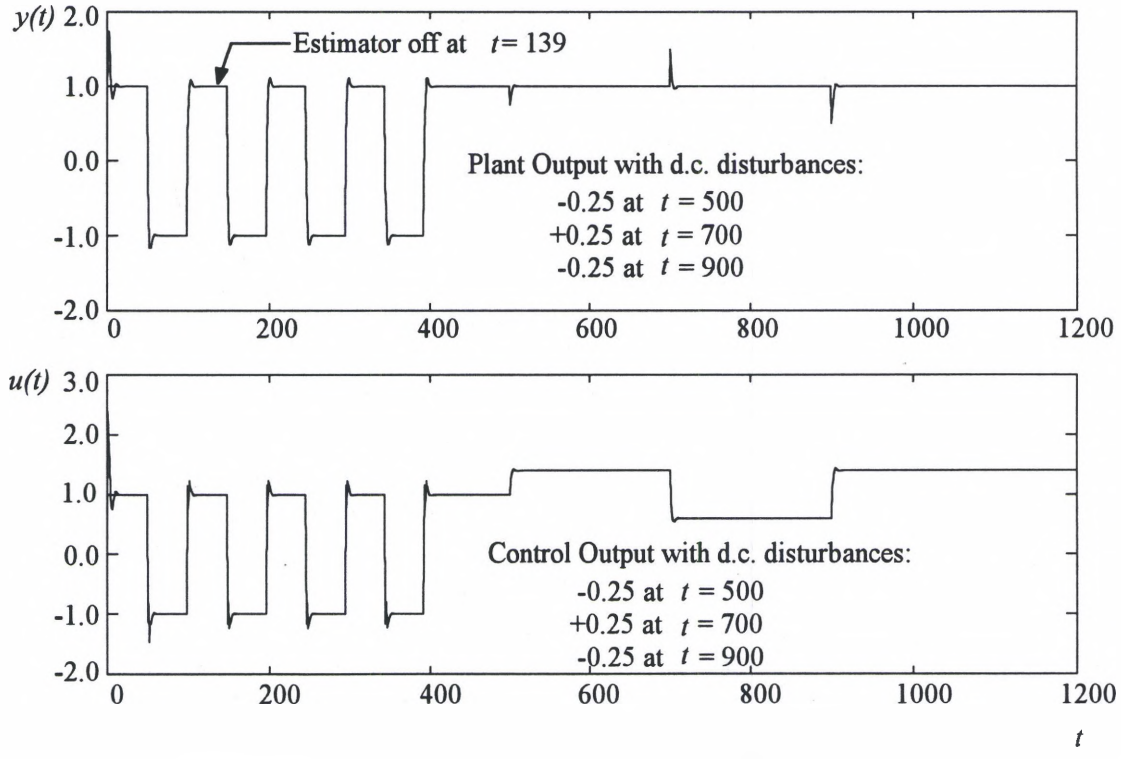

Figure 117. SSTC PID control with dead-zone and band-pass filter

Prefiltering the estimator data through $H_{BP}(z^{-1})$ solves the disturbance problem by preventing the estimator from reacting to the disturbances. It has also improved the response of the controller in general, which can be seen in more detail in Figure 118.

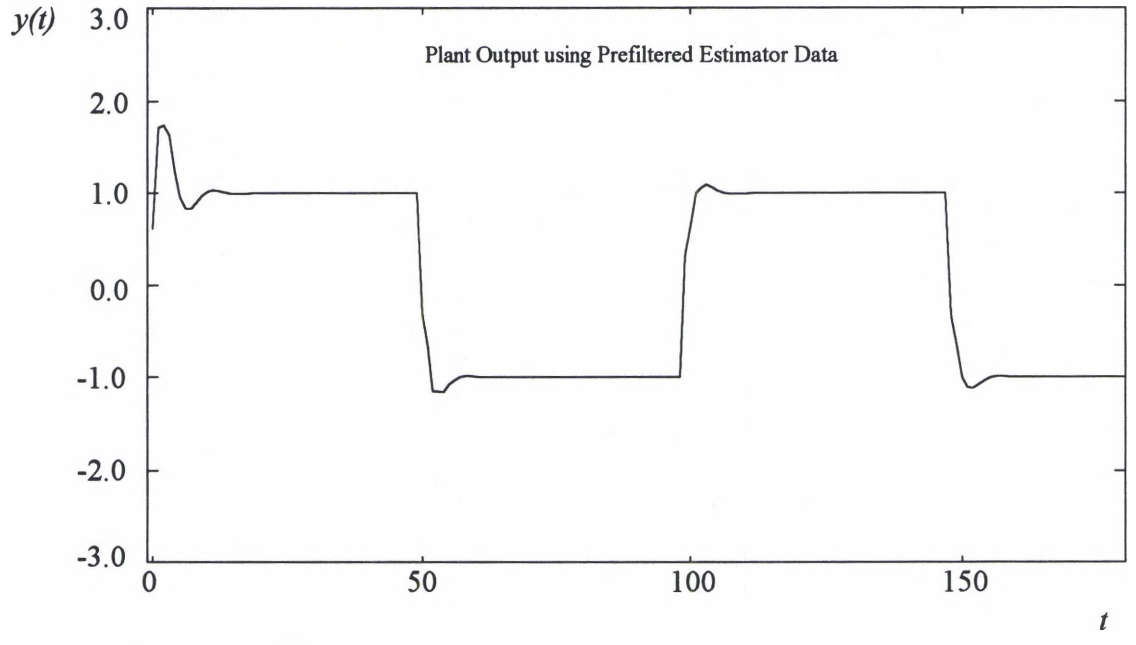

Figure 118. Magnified SSTC PID controlled plant with dead-zone and band-pass filter

The response of the plant output is significantly improved over the previous simulations. The controller adapts much more quickly than before and the step response exhibits less overshoot. A plot of the parameter vector is shown in Figure 119. Unlike the parameter estimates from the previous simulations, the estimates remain nearly constant after the initial adaptation period. The dead-zone disables the estimator when $|\varepsilon(t)| \leq 0.002$ for 25 samples, and the band-pass filter removes the d.c. disturbances before the estimator can reactivate. The effect of the filter on the input and output data is seen in Figure 120.

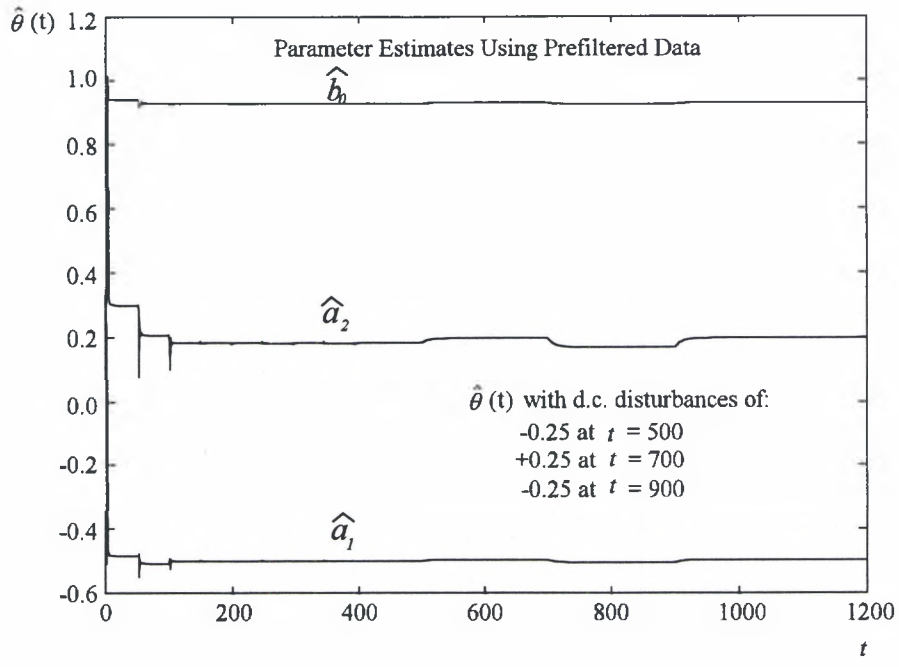

Figure 119. Estimated parameters for controller with dead-zone and prefiltering

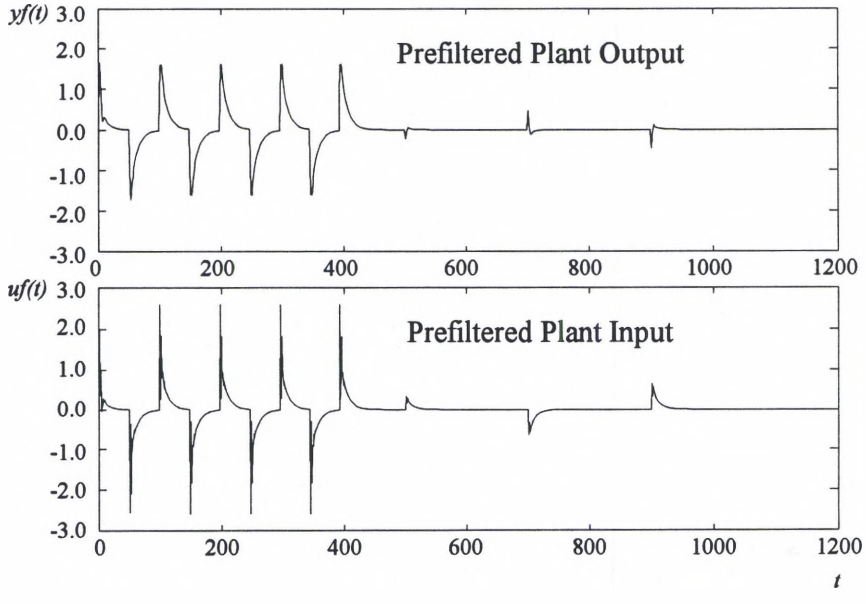

Figure 120. Prefiltered input and output signals to estimator

The filtered input and output signals shown in Figure 120 hardly resemble the unfiltered signals shown in Figure 117 since the low frequency content of the signals has been removed. The filtered signals do, however, contain the frequency information that the U-D RLS algorithm requires to develop a reasonable model of the plant. The band-pass filter removes the d.c. disturbances from the data before the parameter error is able to reactivate the estimator. The *turn on* time of the dead-zone must be set to be longer than the settling time of the filter in order to give the filter time to remove the disturbances.

Combining the band-pass filter with the dead-zone provides an effective mechanism for dealing with low frequency disturbances, at least in the deterministic case. In the final simulation of this chapter, the adaptive PID controller including the dead-zone and the estimator data filter is tested in the presence of noise. The results of the simulation are shown in Figure 121.

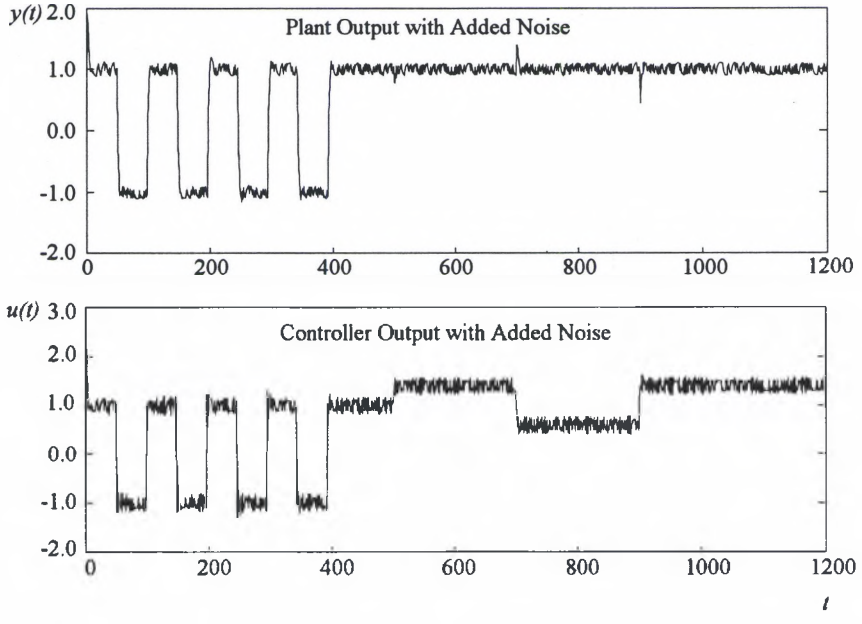

Figure 121. Plant input and output under SSTC PID Control with added noise

The noise injected into the system has negligible effects on the performance of the SSTC PID controller. The controller is able to adapt to the plant very rapidly, after only one or two cycles of the reference input. The controller also exhibits good setpoint tracking and disturbance rejection capabilities.

5.6 Chapter Summary

In this chapter, the SSTC PID controller has demonstrated both the ability to track a reference input and to regulate a steady-state output. The key to the operation of the SSTC PID controller is the U-D RLS estimation algorithm, which was tested first. The algorithm was tested in open-loop with the input to the plant being excited by a pseudo-random binary sequence. The PRBS was shown to meet the criteria for persistent excitation and proved to be easily implemented in software. The U-D RLS algorithm demonstrated the ability to produce accurate plant models when the plant input was sufficiently rich in frequency content. To track time-varying parameters, the U-D RLS algorithm employed a forgetting factor to weigh the more recent estimator data more heavily than the older data. The effect of the forgetting factor on parameter convergence was tested and analyzed. The estimation algorithm was also tested in the presence of higher frequency noise, which produced biased parameter estimates. A digital implementation of a second-order Butterworth low pass filter was added to the estimation algorithm to filter out the noise before the data was passed to the estimation algorithm, improving the accuracy of the estimated plant model.

The U-D-RLS algorithm was combined with the SSTC control law design mechanism to produce the Simplified Self-Tuning Controller proposed by Warwick, Karam and Tham (1987). The control law design mechanism was tested first by disabling the estimator and fixing the values of the estimator parameter vector $\hat{\theta}(t)$ to equal the

known actual values of the true plant. Simulations showed that the control law functioned as predicted when the parameter estimates were equal to the true plant parameters. The estimator was then activated and the completed SSTC controller was tested in closed-loop with a second-order plant. Assuming a second-order model in the controller, the controller functioned well; however, the optimum response obtained in the test with fixed estimates could not be duplicated. With the excitation to the plant being a square wave on the reference input, the plant was not sufficiently excited to allow the estimator to produce an unbiased model of the plant. With the forgetting factor set to a relatively low value (0.95), the parameters converged rapidly, but tended to drift and were susceptible to noise. Larger values of λ produced more stable estimates at the expense of somewhat reduced accuracy.

Next, the effect of using higher order models in the controller algorithm was tested. The same second-order plant was tested with SSTC model orders ranging from two to fifteen. The higher order models generated less overshoot in the output of the plant, but tended to increase the settling time. Also, the locations of the poles and zeros of the higher order models were examined. Higher order models tended to create poles and zeros that canceled out the effect of one another. The SSTC controller performance was evaluated for an extended time period. In a 10,000 sample simulation, the controller remained stable, with or without noise in the system. It was demonstrated, however, that a larger forgetting factor produced more stable parameter estimates in extended simulations. The effect of undermodeling was also tested. A second-order plant was controlled using a first-order model in the SSTC algorithm. The simulation results demonstrated how undermodeling can lead to bursting in the plant output. A dead-zone was incorporated into the algorithm to shut off the estimator when the parameter error converged to a relatively small value. The dead-zone eliminated bursting in the output.

The control law of the SSTC algorithm was constrained to conform to a PID-like structure. The PID version of the SSTC controller required that only three parameters be estimated by the U-D RLS algorithm. When the SSTC PID algorithm was tested using the same values of the controller parameters used in the regular SSTC controller, the PID version of the algorithm became unstable. It was discovered that the SSTC PID algorithm exhibited significant sensitivity to the initial values of the parameter vector $\hat{\theta}(t)$ that was not observed in the regular SSTC algorithm. Acceptable results were finally obtained by setting $\hat{\theta}(0)$ to be equal to 1.0. The ability of the SSTC PID algorithm to track time-varying parameters was also tested. The simulation was performed by changing the values of the parameters of the true plant in the middle of the test. The SSTC PID controller was able to quickly respond to the sudden change in the plant parameters with minimal disruption in the plant output.

The ability of the SSTC PID controller to provide disturbance rejection was also tested. In the simulations performed, the reference input to the controller was cycled a few times to allow the controller to tune itself to the plant. The reference input was then held constant while three d.c. disturbances were imposed on the plant output. The addition of the first disturbance caused the estimator to produce biased estimates which caused a significant amount of instability in the plant output. A dead-zone was used to shut off the estimator when the parameter error fell below a threshold value for a predetermined number of samples. However, the d.c. disturbances caused the parameter error to increase to the point where the estimator was reactivated, again generating biased estimates. A first-order high pass filter was added to eliminate the low frequency disturbances from the estimator data. The filter eliminated the level disturbances before the estimator was reactivated, allowing a constant steady-state output to be maintained. The disturbance rejection capability of the controller was also tested in the presence of

low level, high frequency noise, and the adaptive PID controller was able to function without difficulty.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

6.1 Project Overview

The objective of the work presented in this paper was to design an adaptive PID controller for implementation on the Motorola DSP56000. In Chapter 2, the concept of PID control was first examined from a continuous-time perspective. The classical PID algorithm was developed and some of the difficulties encountered with the classical form of the algorithm were explained. The derivative-of-output PID model was presented as an alternative to classical algorithm. A practical discrete-time PID algorithm was then developed from the continuous-time derivative-of-output model. Several important PID control design methods were also examined, including both empirical and model-based design techniques.

The discrete-time algorithm was implemented in Motorola DSP56000 assembly language and tested. The test results were reported in Chapter 3. The idea of using a DSP chip for control was examined and an overview of the DSP56000 architecture and instruction set was presented. The DSP56000-based PID control algorithm was then tested in the laboratory. A Motorola ADS56000 development system was connected to an Intel 80386-based computer via a National Instruments AT-MIO-16 analog I/O board.

The personal computer served as the plant for real-time testing of the PID controller. The DSP56000-based controller functioned as expected in all of the tests performed.

The development of an adaptive PID control algorithm was presented in Chapter 4. A recursive least-squares algorithm based on Bierman's (1977) U-D Cofactorization method was selected for the parameter estimator. The parametric estimation algorithm incorporated a forgetting factor for tracking linear time-varying plants and covariance resetting was used to mitigate the problem of estimator wind-up. The algorithm also employed a dead-zone to prevent the parameter estimates from drifting if the plant input ceased to be persistently exciting. Achieving a persistently exciting input proved to be a problem when the estimation was performed on a plant operating in a closed-loop, which was the case when the U-D RLS algorithm was used as part of an adaptive controller. The U-D RLS algorithm was combined with a pole-cancellation control law design scheme to form the Simplified Self-Tuning Controller proposed by Warwick, *et. al.* (1987). The control law of the SSTC algorithm was modified to conform to a PID controller structure by constraining the plant model to a specific second-order model. An alternative control law design based on a pole-placement technique was also presented in Chapter 4.

In Chapter 5, the SSTC adaptive control algorithm was simulated. The U-D RLS algorithm was first tested in open-loop. A discrete-time version of a second-order Butterworth filter was incorporated into the algorithm to deal with biased parameter estimates caused by high frequency noise. The U-D RLS algorithm produced accurate models when the input to the plant was persistently exciting, even in the presence of low level noise, provided the signals to the estimator were prefiltered. The U-D RLS algorithm was then combined with the SSTC control law design mechanism to form an adaptive controller. Several important aspects of the SSTC algorithm were tested, including the effect of the forgetting factor on the convergence rate of the parameter

estimates and the performance of the algorithm assuming high model orders. Bursting was also observed in an investigation of undermodeling. The SSTC algorithm performed well in simulation, even when the plant input was not persistently excited.

The SSTC algorithm was then modified to form an adaptive PID controller. The SSTC PID controller was able to track time-varying parameters and performed well in both servo tracking and regulation simulations. Some difficulties were manifested in the PID version of the SSTC algorithm, however, that were not evident in the regular SSTC algorithm. The SSTC PID algorithm exhibited sensitivity to the initial values of the parameter vector $\hat{\theta}(t)$, whereas $\hat{\theta}(0)$ seemed to have a negligible effect on the regular SSTC algorithm. The SSTC PID controller also reacted poorly to a series of constant load disturbances on the output until a high-pass filter was incorporated into the algorithm. The high-pass filter eliminated the load disturbance effects from the estimator data allowing the adaptive controller to maintain control of the plant. Although a number of difficulties were encountered, in general, the simulations in Chapter 5 proved the SSTC PID algorithm to be functional when applied within the prescribed limitations of the algorithm.

In this chapter, some conclusions are drawn based on the results of the preceding chapters. Some issues regarding the performance of the real-time testing of the normal PID algorithm on the DSP56000 are discussed. After that, performance issues related to the U-D RLS estimation algorithm are considered. The practical viability of the SSTC PID algorithm is then examined. Issues regarding the implementation of an adaptive PID control algorithm on the DSP56000 are also reviewed, and finally, recommendations for future work are presented.

6.2 Real-Time Testing of the Regular PID Algorithm

A practical discrete-time PID algorithm was developed and implemented on the Motorola DSP56000. The performance of the DSP56000-based PID controller was tested in real-time using two different sets of controller parameters. When the controller was tested with parameters obtained using the Ziegler-Nichols (1942) frequency response method, the response of the plant output was generally underdamped. The Ziegler-Nichols (1942) experiment yielded continuous-time parameters were discretized for use in the DSP56000. In Chapter 2, the discretized derivative term was determined to be:

$$D(k) = d_0 D(k-1) + d_1[y(k-1) - y(k)] \tag{6.1}$$

with the coefficients d_0 and d_1 given as:

$$d_0 = \frac{\left[\frac{2T_d}{NT} - 1\right]}{\left[\frac{2T_d}{NT} + 1\right]} \quad \text{and} \quad d_1 = \frac{2K'T_d N}{NT + 2T_d}.$$

A disadvantage of using the bilinear transform on the derivative term is that as the continuous-time coefficient $T_d \to 0$, the discrete-time coefficients $d_0 \to -1$ and $d_1 \to 0$. For small T_d., this produces a ringing effect on the output as equation (6.1) reduces to:

$$D(k) = -D(k-1).$$

From the Ziegler-Nichols (1942) experiment, the continuous-time derivative term was determined to be $T_d = 0.0910176$. The response of the plant output was significantly improved when the controller was tested using parameters from Hagglund and Astrom's (1985) auto-tuner simulation, which provided a larger derivative term of $T_d = 0.123$. Even though the plant response was improved in the second trial, Hagglund and Astrom's (1985) results could not be duplicated. This is due to the fact that Hagglund and Astrom's continuous-time plant model was discretized using a zero-order hold equivalent for the real-time tests of Chapter 3. A significant phase lag was therefore introduced in the

discrete-time model of the plant which led to a plant response with slightly more overshoot than experienced by Hagglund and Astrom (1985). The tests in Chapter 3 did demonstrate, however, that the DSP56000-based PID algorithm functioned as designed.

6.3 Performance of the U-D RLS Algorithm

In the simulations of Chapter 5, the U-D RLS algorithm produced accurate models when the identification was executed on an open-loop plant with a persistently exciting input. The estimated models were almost indistinguishable from the true plant in terms of frequency response. When the identification was attempted on the plant operating in a closed-loop configuration with a feedback controller, however, the resulting parameter estimates were biased, causing the controller to perform at suboptimal levels. One of Isermann's (1982) criteria for identifiability in the closed-loop case is that the order of the denominator of the controller transfer function must be greater than or equal to the order of the numerator and denominator of the plant model in order to eliminate the linear dependence between the input and output signals. Although this criterion was met in simulations of both the SSTC algorithm and the SSTC PID algorithm, the U-D RLS algorithm still produced biased estimates when operating in a closed-loop scenario. The estimated models were sufficiently close to the actual plant, however, to allow the controller to bring the plant to a reasonable level of control in most of the cases studied.

Stability in a self-tuning controller can only be guaranteed if the estimated parameters eventually converge to reasonable values. According to Kumar (1990), for a self-tuning controller utilizing a recursive least-squares algorithm, this condition can only be guaranteed if the system is strictly minimum phase and if the system is externally perturbed with Gaussian white noise of sufficient magnitude. The plants studied in Chapter 5 met the criterion of being strictly minimum phase, but no noise was added to the

plant input. Of the simulations run in Chapter 5, the output remained stable in all but three of the cases. In the first case, the output became unstable when the initial value of the parameter vector $\hat{\theta}(t)$ was set to 0.5. In the second case, the output became unstable when the SSTC PID algorithm attempted to control a plant of a higher order than the assumed model. In the third case, the output was driven into instability when certain high pass filter designs were attempted. In each case, the instability was not caused by a failure of the control law, but rather by the failure of the estimator to produce estimates that converged to reasonable values. In the disturbance rejection simulations where the reference input was held constant after an initial tuning-in period, the only excitation to the plant was the presence of disturbances on the output fed back through the controller. If Kumar's (1990) assertions are correct, if the random disturbances on the output do not sufficiently excite the plant, global stability cannot be assumed without the addition of an external perturbation on the plant input. The potential use of additive noise is a subject to be considered for future work . Some questions that remain to be answered are:

1. What should be the spectral qualities and the magnitude of the additive noise?
2. What effect will additive noise have on estimate biasing?
3. What are the limitations for which global convergence can be assumed, even when additive noise is used?

Another estimation issue to be considered is the use of the dead-zone to solve the problem of drifting parameters, particularly once the output has stabilized in the regulation case. The parameter error represents the effect of modeling error plus the effect of random disturbances. Since the dead-zone is configured to disable the estimator when the parameter error drops below a threshold value, when noise is present in the system, the threshold must be set to a level higher than the noise floor. The determination of an

appropriate threshold value is therefore significantly more difficult with noise present on the output. If the threshold value is set too high, the estimator is disabled before the parameter estimates have converged, resulting in suboptimal controller performance. If, on the other hand, the threshold value is set too low, the estimator never shuts off and the parameters may drift. The noise level must therefore be determined *a priori* to properly fix the level of the shut-off threshold. Even if the dead-zone threshold is properly adjusted initially, it would have to be readjusted if the noise level changes significantly. Hysteresis may also be required in the dead-zone to prevent the estimator from prematurely reactivating if an impulse occurs in the parameter error (as takes place with a sudden change in the reference input or with a load disturbance). In Chapter 5, a time window was utilized to filter out such impulses. The time window allowed the parameter error transients to decay before the estimator was reactivated.

Selection of the dead-zone thresholds in Chapter 5 was accomplished through extensive trial-and-error attempts in numerous simulations. Even in a simulation environment where the behavior of the noise was carefully controlled, establishment of appropriate threshold values proved to be a time-consuming task. Reiterating the assertions of Rey and Johnson (1990), "... in order for leakage and dead-zones to be effective in the avoidance of bursting and preservation of stability, they must be properly tuned. Further, our current understanding about them is not sufficient even to assert in actual implementation whether they are appropriately tuned." Middleton and Goodwin (1990), however, present the dead-zone as a viable solution to avoid potentially "catastrophic behavior in the presence of undermodeling and poor excitation." The use of the dead-zone as a solution to the problem of bursting remains controversial. At this time, however, it appears to be one of the better solutions for addressing the problems of undermodeling and lack of persistent excitation.

Another important issue is the necessity of prefiltering the estimator data to remove low frequency disturbances from the parameter vector to prevent biased estimates. Several methods have been proposed in the literature for removing level disturbances from the estimator data (see Isermann, 1982), one of which is the use of a high-pass filter. In the simulation in Chapter 5, the estimator data was prefiltered using a first-order high-pass filter with a cutoff frequency of 0.1 rad/sec. A number of other filters, with cut-off frequencies as low as 0.0001 rad/sec and of order as high as 8, were also attempted, the results of which were not included in the discussion. Two problems were encountered that precluded the use of the other filters. The first problem was that when very low cut-off frequencies were attempted (e.g., $\omega_c = .0001$ or $\omega_c = .001$), small filter coefficients led to numerical difficulties in the calculations. The other difficulty encountered was the inability of estimator to identify the plant when the alternative filters were attempted. The exact explanation for this is not known at this time. In any case, for the filters to be properly designed, the frequency response of the plant must be known *a priori*. Further study of the effect of prefiltering of the estimator data is another topic for future work.

One positive observation from the simulations of Chapter 5 is that the U-D RLS algorithm proved to be numerically robust. In all of the simulations performed, no computational problems occurred (i.e., singularity of the information and covariance matrices) that could be attributed to numerical difficulties with the U-D RLS algorithm.

6.4 Performance of the SSTC PID Controller

Another issue brought to light in the simulations of Chapter 5 is the sensitivity of the PID version of the SSTC algorithm to the initial value of the parameter vector $\hat{\theta}(t)$. The selection of $\hat{\theta}(0)$ has little effect on the general SSTC algorithm. When the control law is changed from:

$$u(t) = \frac{A(q^{-1})e(t)}{B(1)} + \frac{B(q^{-1})u(t-d)}{B(1)}$$

to:

$$u(t) = \frac{u(t-1) + (1 + a_1 q^{-1} + a_2 q^{-2})e(t)}{b_0}, \tag{6.2}$$

the selection of $\hat{\theta}(0)$ becomes critical. Improper selection of $b_0(0)$ drives the plant into instability before the estimator has an opportunity to generate a reasonable estimate of the parameter. Although the algorithm functioned when $b_0(0)$ was set to 1.0, it is not known whether this value for $b_0(0)$ is appropriate for other plants. The problem was never observed in simulations where the order of the numerator and the order of the denominator of the control law were the same, even when a first-order plant model was assumed. Further investigation into the selection of $\hat{\theta}(0)$ is therefore recommended.

The PID version of the SSTC algorithm is derived by placing certain constraints on the general SSTC algorithm. While the regular SSTC control law allows for models of any order to be assumed, the PID version of the algorithm assumes a specific second-order plant model, given as:

$$G_p(q^{-1}) = \frac{b_0}{1 + a_1 q^{-1} + a_2 q^{-2}} q^{-1}. \tag{6.3}$$

The model of equation (6.3) can be derived from a continuous-time plant given as:

$$G_p(s) = \frac{1}{c_2 s^2 + c_1 s + c_0}$$

using the backward difference transformation and assuming a time delay of 1 sample. Although it is not uncommon to assume a reduced-order plant model when employing PID control, constraining the plant to fit the model of equation (6.3) severely limits the application of the SSTC PID algorithm. In addition to the model order, the PID version of the SSTC algorithm assumes a delay time of one sample. Several unsuccessful simulations were attempted assuming the controller model of equation (6.3) with a plant having a transfer function with a second-order numerator. (The algorithm was not tested

using plants with longer timer delays.) It appears, therefore, that the plant must be able to be accurately modeled by equation (6.3) for the SSTC PID algorithm to function properly.

Another factor potentially limiting the application of both the regular SSTC algorithm and the PID version of the SSTC algorithm is that the designer cannot adjust the controller to modify the dynamic response of the plant. As mentioned previously, the deadbeat-like control strategy used by the SSTC family of algorithms inherently fixes the response of the plant output for a given set of parameters. In simulation, the output typically exhibited some degree of overshoot as a result of modeling error in the closed-loop estimation. Although the basic SSTC strategy provides no opportunity to design the response to eliminate the overshoot, Warwick, *et. al.* (1987) present a pole-placement version of the algorithm that allows the response of the reference input to be defined by selection of target locations of the closed-loop system poles. The SSTC pole-placement algorithm, however, does not conform to a PID-like structure. The pole-placement algorithm presented in Chapter 4 offers the ability to select the locations of the closed-loop system poles while conforming to a standard PID controller model.

6.5 Implementation of Adaptive PID on the DSP56000

The implementation of the PID controller in Chapter 3 and the simulation of the adaptive PID controller in Chapter 5 indicate that implementation of an adaptive PID controller on the Motorola DSP56000 is quite possible. Several issues remain to be addressed, however, before the adaptive PID control algorithm can be implemented. In the proposed adaptive PID control algorithms, the uncertainties of the estimated parameters were ignored by applying the *certainty equivalence principle.* In order to prevent computational overflow and underflow in the DSP56000, the maximum values of the estimated plant parameters must be estimated a priori to ensure that they remain

bounded. Since the DSP56000 uses fractional arithmetic, the bounds on the estimated parameters must be normalized. Experiences from Chapter 5 indicate that the parameter estimates generally remain less than one. There were occasions, however, when the parameters diverged to much larger values. When the parameter estimates are scaled, the question of word length becomes an issue, as available bits in the word are used up in the scaling process. Parameter estimates can become quite small if higher order models are assumed. Parameter scaling not only introduces the increased potential for modeling error, but it also increases the risk of singularity of the covariance matrix, as less bits become available for the computations. Tan and Kyriakopoulos (1988) recommend simulation of the algorithm over a wide range of operating conditions on a large word-length general-purpose computer using floating-point arithmetic to determine the proper scaling factors for the variables. Ideally, however, the range of the parameters should be determined statistically using Monte Carlo simulations.

The simulations of Chapter 5 demonstrated that adaptive control can work well if all of the theoretical pre-conditions are met and if all of the design parameters are properly selected. In the real world, however, pre-conditions may be violated and the proper design parameters may not necessarily be chosen. Placing bounds on the estimates, for instance, could compromise system stability if improperly handled. Noise levels may suddenly change, rendering dead-zone thresholds ineffective. Even the selection of initial values of some controller parameters has been shown to be critical in some cases. These factors all point to the need for some type of *supervisory control* in the final DSP implementation of the algorithm. Knapp and Isermann (1990) have proposed the addition of two levels of control, called the *supervision level* and the *coordination level*. The incorporation of these levels into the self-tuning regulator adaptive controller model is shown in the block diagram of Figure 122.

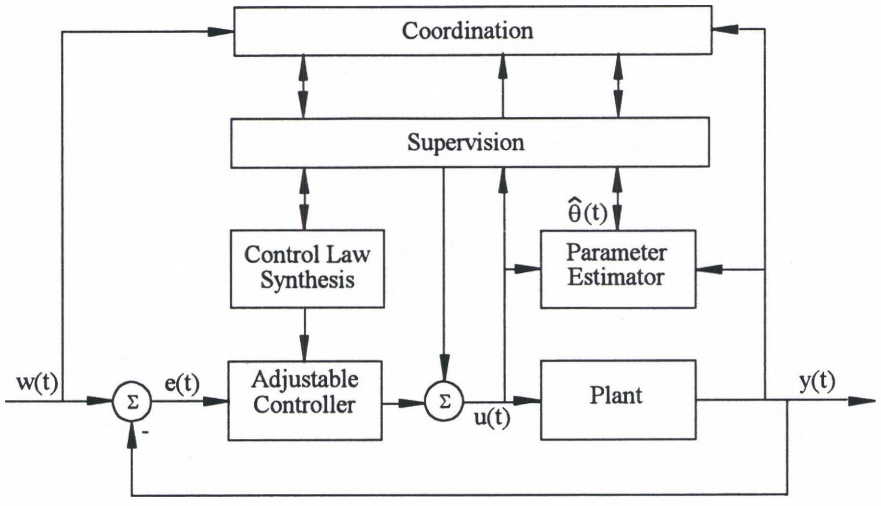

Figure 122. Adaptive controller model with supervision and coordination levels

Knapp and Isermann (1990) have given the following functions for the two additional levels:

SUPERVISION LEVEL

- monitoring the parameter estimates
- detecting a process model mismatch
- decision making, what has changed?
- monitoring the controller design
- monitoring the closed-loop behavior

COORDINATION LEVEL

- performing a start-up procedure
- switching on/off parameter estimation
- choosing the most suitable control algorithm
- decision making, what sort of controller parameters will be used

The details of the above functions are beyond the scope of this paper, but the lists provide a general idea of what sort of supervisory functions might be required in a real-world implementation of adaptive PID control.

6.6 Viability of the SSTC PID Algorithm

This project has focused on the development of an adaptive PID controller. The results of Chapter 5 raise some serious questions, however, as to whether the benefits of using a simplified controller model outweigh the disadvantages of constraining the algorithm to a PID-like structure. It has been noted that assuming the SSTC PID model of equation (6.3) significantly limits the application of the SSTC PID controller. The difficulties encountered with the PID version of the SSTC algorithm not encountered in the general SSTC algorithm have also been discussed. The benefits of using the PID version of the SSTC controller can be summarized as follows:

- Reduced computational burden over the general algorithm
- A well established structure that is easily understood by operators.

The computational advantage using the SSTC PID algorithm over the general SSTC algorithm when constrained to a second-order model is not significant. But the addition of the second parameter (b_1) in the general SSTC algorithm eliminates the sensitivity of the algorithm to $\hat{\theta}(0)$. Also, the advantage of using a well established controller structure is only gained if the controller must be periodically readjusted by the operator. The very purpose for using adaptive control, however, is to eliminate the need for readjustment of the controller parameters. Therefore, the limitations imposed by forcing the SSTC controller into a PID-like structure, and the difficulties encountered in simulation of the SSTC PID algorithm in Chapter 5, appear to outweigh any advantage gained by using the PID version of the SSTC algorithm over the general SSTC algorithm.

The advantages of using a pole-placement algorithm over the SSTC algorithms have also been discussed. Pole-placement gives the designer the ability to compensate for modeling errors introduced by biased parameter estimates by re-selecting target locations of the closed-loop system poles. The pole-placement algorithm presented in Chapter 4 assumed a second-order ARX model for the plant, rather than the more restrictive model of equation (6.3). Also, the adaptive pole-placement controller of Chapter 4 allows for the selection of four closed-loop poles, giving the designer much more flexibility than with the SSTC algorithms. Adaptive PID using pole-placement appears to have greater practical potential than the SSTC PID algorithm proposed by Warwick, *et. al.* (1987).

6.7 Recommendations for Future Work and Concluding Remarks

Several recommendations for future work have already been suggested to. In this section, those recommendations, along with some others, are summarized and some concluding remarks are made.

1. **Code and simulate the adaptive pole-placement algorithm:**

   The limitations of the SSTC algorithms have been stated earlier. Although the SSTC approach to adaptive control is computationally efficient and easy to implement, its potential for application is limited. The pole-placement algorithm derived in Chapter 4 should therefore be simulated.

2. **Develop guidelines for development of estimator data filters:**

   The simulations of Chapter 5 made it clear that prefiltering of estimator data is essential. However, selection of filters that did not impede the operation of the estimator proved to be difficult. The area of estimator data filtering needs to be investigated more thoroughly.

3. **Investigate the use of external perturbation signals:**

   The requirements for convergence of the parameters to ensure stability have been discussed at length. In many applications, it may be difficult to guarantee that the plant input is persistently excited. In those cases, external perturbation of the plant may be required. This entire area warrants further study.

4. **Determine Supervisory-Coordination level needs for DSP implementation:**

   The level of supervision and coordination for implementation of the adaptive PID algorithm on the DSP56000 needs to be determined. At a minimum, some method of monitoring the estimated parameters must be developed to ensure that they remain bounded and that the controller remains stable when the parameter boundaries are reached.

5. **Develop the adaptive PID algorithm for implementation on the DSP56000:** Once the above steps have been completed, the simulation program must be converted for DSP56000 implementation. The program can be tested in the ADS56000 development system using the real-time plant model running on the 80386-based computer.

In conclusion, implementation an adaptive PID controller on the Motorola DSP56000 appears to be feasible. The use of DSP chips for control applications have been shown to offer a number of advantages over conventional microprocessors. Although DSP chips were develop primarily for signal processing applications, the Harvard architecture of the DSP56000 allows for increased throughput and consequently, decreased computation time which results in increased sampling rates. This additional processing power becomes extremely important when faced with the extra computations required in an adaptive control application.

Adaptive control offers a number of advantages over other control methods, particularly when dealing with time-varying plants. It is apparent from the work presented in this paper, however, that adaptive control is not a universal solution to all control problems. Since adaptive control is inherently non-linear, stability and robustness analyses are extremely difficult. Although stability proofs have been given for a few specific cases, they are generally subject to unrealistic constraints and assumptions. In addition, the whole area of robustness theory as applied to adaptive control is still under development. The success and failure of the adaptive control algorithm, especially the self-tuning regulator variety, hinges on the performance of the parameter estimator. Although the recursive least-squares algorithm is considered to be one of the more robust approaches to parametric estimation, its ability to accurately estimate the plant parameters depends entirely on the level of excitation provided to the plant. This can become a problem when

the plant is operating in a closed-loop, particularly when the primary function of the controller is regulation. Although several ad hoc approaches have been proposed for dealing with problems such as lack of excitation and the presence of disturbances on the output, solutions to these problems remain the subject of much debate.

Very often, the concept of a *self-tuning* controller is thought to eliminate the work of the control designer, since the controller tuning parameters are derived automatically *on-line*. It is evident from the work presented here that this is by no means the case. It is true that the parameters used in the control law are derived on-line; however, the implementation of adaptive control is a complex process requiring the selection of a host of other parameters, most of which are application specific. With the need for prefiltering of estimator data, it may be difficult to implement an adaptive controller without having a priori knowledge of the plant. In microprocessor or DSP-based applications, extensive simulations must be run to determine the bounds of the parameter estimates to allow for scaling of the estimated parameters. Accurate noise models must be available for the selection of dead-zone threshold values. Depending on the control algorithm employed, even the selection of the initial value of the parameter vector can be critical. As stated by Astrom (1987), "An adaptive regulator , being inherently nonlinear, is more complicated than a fixed gain regulator. Before attempting to use adaptive control it is, therefore, important to first examine if the control problem cannot be solved by constant gain feedback." Adaptive control does, however, offer a viable solution to control problems where fixed-gain feedback is not a viable option.

APPENDIX

```
;*********************************************************
; file PID64B.ASM                                        *
;                                                        *
; This program implements a PID control algorithm in  DSP56001  *
; assembly language using the SCI clock interrupt to control the  *
; sample rate.  The controller algorithm is given as follows:  *
;                                                        *
;       1. KP = [K*(1+h/2/Ti)]/64                        *
;       2. Beta = (2*Ti - h)/(2*Ti + h)                  *
;       3. Gamma = Td/h                                  *
;       4. d1 = (2*Gamma/N - 1.0)/(2*Gamma/N + 1.0)      *
;       5. d2 = Kp/16 * (2*Gamma)/(2*Gamma/N + 1.0)      *
;       6. input y(k)                                    *
;       7. e(k) = w(k) - y(k)                            *
;       8. D(k) = d1*D(k-1) + 1024*d2*[y(k-1)-y(k)]      *
;       9. y(k) => y(k-1)                                *
;      10. if D(k) > HILIMIT, D(k) = HILIMIT             *
;      11. if D(k) < LOWLIMIT, D(k) = LOWLIMIT           *
;      12. P(k) = Kp * e(k)                              *
;      13. I(k) = I(k)/64                                *
;      14. u(k) = P(k) + I(k) + D(k)                     *
;      15. if u(k) > HILIMIT, u(k) = HILIMIT             *
;      16. if u(k) < LOWLIMIT, u(k) = LOWLIMIT           *
;      17. u(k) = u(k) * 64                              *
;      18. output u(k)                                   *
;      19. I(k+1) = Beta*I(k) + (1-Beta)*u(k)            *
;      20. if I(k+1) > HILIMIT, I(k+1) = HILIMIT         *
;      21. if I(k+1) < LOWLIMIT, I(k+1) = LOWLIMIT       *
;      22. I(k+1) => I(k)                                *
;      23. go to step 6                                  *
;                                                        *
;*********************************************************
;
;
```

```
; Written by M. DePoyster    7/18/92
; Revised                    8/25/92
; Revised                    9/12/92
; Revised                    9/19/92
; Revised                    1/01/93

page 80

;****************************************
;              Define DSP Registers      *
;****************************************

IPR         equ $FFFF       ;(x:mem) Interrupt Priority Register
BCR         equ $FFFE       ;(x:mem) Bus Control Register
SCCR        equ $FFF2       ;(x:mem) SCI Clock Control Register
SCR         equ $FFF0       ;(x:mem) SCI Control Register
RX_TX       equ $FFEF       ;(x:mem) SSI Transmit/Receive Data Register
SSISR       equ $FFEE       ;(x:mem) SSI Status Register
CRB         equ $FFED       ;(x:mem) SSI Control Register B
CRA         equ $FFEC       ;(x:mem) SSI Control Register A
PCC         equ $FFE1       ;(x:mem) Port C Control Register


;****************************************
;          Define Variables & Constants  *
;****************************************

w           equ $0000       ;reference input (setpoint)
e           equ $0001       ;error term storage
I           equ $0002       ;integral term storage word
temp        equ $0003       ;temporary storage
ykm1        equ $0004       ;previous value of y(k), y(k-1)
yk          equ $0005       ;current sample of y(k)
D           equ $0006       ;derivative term storage word
Kp          equ $0007       ;K*(1+h/2/Ti)/64 ... Kp is prescaled by 1/64
Beta        equ $0008       ;(2*Ti-h)/(2*Ti+h)
Betam       equ $0009       ;1-Beta
d1          equ $000A       ;(2Td/Nh - 1)/(2Td/Nh + 1)
d2          equ $000B       ;Kp*(2Td/h)/(2Td/hN + 1)/16 .. D2 scaled 1/1024
timfact     equ $000C       ;SCI interrupts/sec =
                            ;fosc/(64*(7(SCP)+1)*(CD+1))
                            ;(fosc=20,500,000 in our case)
                            ;($002fa3 yields 10 Hz
preset      equ $000D       ;samples before switching setpoint
```

```
;*****************************************
;                START PROGRAM                *
;*****************************************

org    p:$40

;*******************************************************
;                Initialize Variables, I(k), D(k) and y(k-1)          *
;*******************************************************

        clr a
        move a,x:I
        move a,x:D
        move a,x:ykm1
        move a,x:temp

;*************************************************************
;                Initialize IPR, BCR, SCI and SSI                                *
;*************************************************************

; Initialize IPR to allow interrupts to occur
; Set SSI to Level 2 and SCI to Level 1

        movep #$C000,x:IPR ;allow SCI interrupts only

; Set up ADS board in case of force break instead of force reset

        movep #0,x:BCR      ;set bcr to zero
        movec #0,sp         ;init stack pointer
        movec #0,sr         ;clear loop flag/interrupt mask bits

; Set up the SSI for operation with the DSP56ADC16EVB
; The following code sets port C to function as SCI/SSI

        move #$0,a0         ;zero PCC to cycle it
        movep a0,x:PCC

        move #$0001ff,a0
        movep a0,x:PCC      ;write PCC

; The following code sets the SSI CRA and CRB control registers for external
; cont. synchronous clock, normal mode.
```

```
        move #$004000,a0    ;CRA pattern for word length=16 bits
        movep a0,x:CRA
        move #$000200,a0    ;CRB pattern for cont. ck,synch,normal mode
        movep a0,x:CRB      ;word long frame synch, external clock and frame
                            ;synch

; Set Up SCI Timer

      movep #$2000,x:SCR          ;Enable SCI Timer Interrupt
      movep x:timfact,x:SCCR;     ;Set clock to 20kHz (.05ms Ts)
      andi  #$FC,MR               ;Enable interrupts

;*****************************************
;              Initialize set-point counter          *
;*****************************************

      move x:preset,a0
      move a0,r0

;*************************************************************
;                    Loop until interrupt                                  *
;*************************************************************

self  jmp self                    ;looping waiting for interrupt

;*************************************************************
;              SCI TIMER INTERRUPT SERVICE ROUTINE          *
;                                                           *
;                          Main Control Loop                *
;*************************************************************

;*************************************************
;       Enable SSI to Transmit and Receive Data          *
;*************************************************

timer   move #$003000,x1          ;Set up X1 for OR instruction
        move x:CRB,a              ;Move SSI CRB to Acc A
        or x1,a                   ;Turn on RE and TE
        move a1,x:CRB             ;Move Acc A back to SSI CRB

;*************************************************
;              Wait for A/D word to be clocked in          *
;              then move it to Acc A.                      *
;*************************************************
```

```
poll1   jclr #7,x:SSISR,poll1       ;Loop until RDF bit=1
        move x:RX_TX,a              ;Read A/D data
        move a,x:yk                 ;store y(k)

;*************************************************
;       Check counter to see if time to switch                  *
;       setpoint (reference) polarity                   *
;*************************************************

        clr b                       ;Clear Acc B
        move r0,b0                  ;Move current count to B
        move (r0)-
        tst b
        jne go                      ;jump to "go" if count not = 0
        move x:preset,b0            ;else:  1. reset counter = preset
        move b0,r0
        move x:w,b                  ;       2. change polarity of w
        neg b
        move b,x:w

;*************************************************
;               Store input y(k) and solve for          *
;               error term, e(k)=w(k)-y(k)              *
;*************************************************

go      clr b
        move x:yk,a
        move a,y1                   ;y(k) => Y1
        neg a           x:w,x1      ;Negate y(k)... w(k) => X1
        add x1,a        x:ykm1,b    ;e(k)=w(k)-y(k)... y(k-1) => Acc B

;*************************************************
;               Store e(k), solve for y(k-1)-y(k) and   *
;               store y(k) as y(k-1)                    *
;*************************************************

        move y1,x:ykm1              ;y(k) => y(k-1)
        sub y1,b        a,x:e       ;y(k-1)-y(k) => Acc B..store e(k)
        move x:d1,x1                ;d1 => X1

;*************************************************
;               Solve for the Derivative Term, D(k)     *
;*************************************************
```

```
        move x:D,y1                 ;D(k) => Y1
        mpy x1,y1,a    b,x1         ;d1*D(k)=>Acc A ... Acc B => X1
        rep #6                      ;
        asr a                       ;divide d1*D(k) by 64
        move x:d2,y1                ;d2 => Y1
        mpy x1,y1,b                 ;d2*[y(k-1)-y(k)]
        rep #15                     ;repeat next instr. 15X
        mac x1,y1,b                 ;16*[d2*(y(k-1)-y(k))] (scaled)
        add b,a         x:e,x1      ;D(k) ... e(k) => X1
        tfr a,b                     ;Move D(k) to Acc B for scaling
        rep #6                      ;Multiply D(k) by 64
        asl b
        move b,x:D                  ;Store D(k) (with limiting)

;************************************************
;               Solve for Proportional Term, P(k)          *
;************************************************

        move x:Kp,y1                ;Kp => Y1
        mpy x1,y1,b                 ;P(k)=Kp*e(k)

;************************************************
;               Add P(k) and D(k) together for storage     *
;************************************************


        add b,a                     ;P(k)+D(k)=>Acc A
        clr b
        move x:I,b

;************************************************
;               Scale Integral Term by 1/64                *
;************************************************

        rep #6                      ;Repeat next instr 6X
        asr b                       ;I(k)/64


;************************************************
;               Form Complete PID Term                     *
;************************************************

        add b,a                     ;P(k)+I(k)+D(k) scaled by 1/64
```

```
        rep #6                  ;Repeat next instr 6X
        asl a                   ;PID(k)*64
        move a,x:temp           ;store u(k) temporarily


;***********************************************
;           Send Control Output to D/A converter          *
;***********************************************

        move x:temp,x1
        move #$529fbe,y1
        mpy x1,y1,a             ;Multiply u(k) by scaling factor
                                ;of .6455 before outputting
        move a,x:RX_TX          ;Move PID(k) to RX_TX w/ limiting
poll2   jclr #6,x:SSISR,poll2   ;Loop until TDE bit = 1

;***********************************************
;              Solve for next value of I(k), I(k+1)          *
;***********************************************

        move x:Betam,y1
        mpy x1,y1,a    x:I,y1   ;(1-Beta)*u(k) .. I(k)=>Y1
        move x:Beta,x1          ;Beta => X1
        mac x1,y1,a             ;I(k+1) = Beta*I(k) + (1-Beta)*u(k)
        move a,x:I              ;I(k+1) => I(k) (with limiting)

        rti

        org p:$001c             ;SCI Timer interrupt vector
        jsr timer

        end
```

REFERENCES

Ahmed, Irfan (1991): *Digital Control Applications with the TMS320 Family.* Digital Signal Processing - Semiconductor Group, Texas Instruments, Inc.

Astrom, K. J. (1987): "Adaptive Feedback Control," *Proc.of the IEEE*, vol. 75, no. 2.

Astrom, K. J. (1983): "Theory and Applications of Adaptive Control - A Survey," *Automatica*, vol. 19, pp. 471-486.

Astrom, K. J. and P. Eykhoff (1971): "System Identification - A Survey," *Automatica*, vol. 7, pp. 123-162.

Astrom, K. J. and T. Hagglund (1984a): "Automatic tuning of simple regulators with specifications on phase and amplitude margins," *Automatica*, vol. 20, no. 5.

Astrom, K. J. and T. Hagglund (1984b): "A frequency domain method for automatic tuning of simple feedback loops," *Proc. 23rd IEEE Conference on Decision and Control*, Las Vegas, NV, USA.

Astrom, K. J. and T. Hagglund (1988a): "A new auto-tuning design," *Proc. IFAC Adaptive Control of Chemical Processes*, Copenhagen, Denmark.

Astrom, K. J. and T. Hagglund (1988b): *Automatic Tuning of PID Controllers.* Research Triangle Park: Instrument Society of America.

Astrom , K.J. and H. Steingrimsson (1991): "Implementation of a PID Controller on a DSP". *Digital Control Applications with the TMS320 Family*, Texas Instruments Digital Signal Processing - Semiconductor Group.

Astrom, K. J. and B. Wittenmark (1990): *Computer Controlled Systems, Theory and Design.* Prentice Hall, Englewood Cliffs, New Jersey 07632.

Astrom, K. J. and B. Wittenmark (1989): *Adaptive Control.* Addison Wesley Publishing Company, Reading, Massachusetts.

Astrom, K. J. and B. Wittenmark (1984): "Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins," *Automatica*, vol. 20, no. 5.

Astrom, K. J. and B. Wittenmark (1984b): "Practical Issues in the Implementation of Self-tuning Control," *Automatica*, vol. 20, pp. 595-605.

Astrom, K. J. and B. Wittenmark (1980): "Self-tuning controllers based on pole-zero placement," *Proc. Inst. Elec. Eng.*, Part D, vol. 127, pp. 120-130.

Astrom, K. J. and B. Wittenmark (1973): "On self-tuning regulators," *Automatica*, vol. 9, pp. 185-199.

Banyasz, Cs. and L. Keviczky (1982): "Direct methods for self-tuning PID regulators," *Proc. 6th IFAC Symp. Ident.* Washington, DC. Pergamon Press, Oxford.

Becker, A., P. R. Kumar and C. Z. Wei (1985): "Adaptive control with the stochastic approximation algorithm: Geometry and convergence," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 4, pp. 330-338.

Bellman, R. (1957): *Dynamic Programming.* Princeton University Press, Princeton, New Jersey.

Bierman, G. J. (1977): *Factorization Methods for Discrete Sequential Estimation.* Academic Press, 111 Fifth Avenue, New York, New York 10003.

Clarke, D. W. (1984): "PID algorithms and their computer implementation," *Trans Inst M C*, vol. 6, no. 6.

Clarke, D. W. and P. J. Gawthrop (1979): "Self-tuning control", *Proc. Inst. Elec. Eng.*, vol. 126, pp 633-640.

Clarke, D. W. and P. J. Gawthrop (1975): "A Self-tuning controller", *Proc. Inst. Elec. Eng.*, vol. 122, pp 929-934.

Davies, W. D. T. (1970): *System Identification for Self-Adaptive Control.* Wiley-Interscience, a division of John Wiley and Sons, Ltd. New York.

Egardt, B. (1979): "Stability of Adaptive Controllers," *Lecture Notes in Control and Information Sciences*, p. 20. Springer-Verlag, Berlin.

Egardt, B. (1980): "Stability analysis of continuous-time adaptive control systems," *SIAM J. Control Optimiz.*, vol. 18, pp. 540-557.

Feldbaum, A. A. (1960): "Theory of dual control theory I-IV," *Automat. Remote Contr.*, vol. 21, pp. 874-880; vol.21, pp. 1033-1039; vol. 22, pp. 1-12; vol. 22, 109-121.

Goodwin, G. C., P. J. Ramadge and P. E. Caines (1980): "Discrete multivariable adaptive control," *IEEE Trans. Aut. Control*, vol. AC-25, pp. 449-456.

Goodwin, G. C., P. J. Ramadge and P. E. Caines (1981): "Discrete time stochastic adaptive control," *SIAM J. Contr. Optimiz.*, vol. 19, pp. 829-853.

Hagglund, T. and K. J. Astrom (1985): "Automatic Tuning of PID Controllers Based on Dominant Pole Design," *Proc. IFAC Workshop on Adaptive Control of Chemical Processes*, Frankfurt, FRG.

Hannselmann, H. (1987): "Implementation of Digital Controllers - A Survey", *Automatica*, vol. 23 no. 1, pp 7-32.

Ionescu, I. and R. V. Monopoli (1977): Discrete model reference adaptive control with an augmented error signal," Automatica, vol. 13.

Isermann, R. (1982): "Parameter Adaptive Control Algorithms - A Tutorial", *Automatica*, vol. 18, Sept. 1982, pp. 513-528.

Kalman, R. E. (1958): "Design of a Self-Optimizing Control System," *Trans. ASME*, vol. 80, no. 2, pp. 468-478.

Knapp, T. and R. Isermann (1990): "Supervision and Coordination of Parameter-Adaptive Controllers," *Proc. 1990 Automat. Contr. Conf.*, pp. 1632-1637.

Kosut, R. L. (1992): "System Identification for the User: Modeling, Filtering, Detection, Adaptive and Robust Control," *Notes from IEEE tutorial workshop at 1st IEEE Conf. on Control Applications*, 1992.

Kraus, T. W. and T. J. Myron (1984): "Self-Tuning PID Controller Uses Pattern Recognition Approach," *Control Engineering*, June, 1984.

Kumar, P. R. (1990): "Convergence of Adaptive Control Schemes Using Least-Squares Parameter Estimates," *IEEE Trans. Automat. Contr.*, vol. 35, no. 4, pp. 416-424.

Kumar, P. R. and L. Praly (1987): "Self-tuning trackers," *SIAM J. Contr. Optimiz.*, vol. 25, no. 4, pp. 1053-1071.

Kurz, H., R. Isermann and R. Schumann (1980): "Experimental comparison and application of various parameter adaptive control algorithms," *Automatica*, vol. 16, pp. 117-133.

Lammers, H. C. (1982): "A simple self-tuning controller," *IEEE Conf. Applic. Adaptive and Multivariable Control*, Hull. Cotswold Press, Oxford.

Lopez, A. M., J. A. Miller, C. L. Smith, and P. W. Murrill (1967): "Controller Tuning Based on Integral Performance Criteria," *Instrumentation Technology*, November, 1967.

Ljung, Lennart (1987): *System Identification, theory for the user.* Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632.

Ljung, L. and T. Soderstrom (1983): *Theory and Practice of Recursive Identification.* The MIT Press, Cambridge, Massachusetts, London, England

Marsik, J. (1970) : "A simple adaptive controller," in *Proc. 2nd IFAC Symp. on Identification and Process Parameter Estimation*, (Prague, Czechoslovakia).

Marx, M. F. (1959): "Recent adaptive control. Work at the General Electric Co.," in P.C.Gregory, Ed., *Proc. Self-Adaptive Flight Control Systems Symp.*, WADC Tech. Rep. 59-49 (Wright Air Development Center, Wright Patterson Air Force Base, Dayton, Ohio.

McInnis, B. C., Z. Guo, P.C. Lu and J. Wang (1985): "Adaptive Control of Left Ventricular Bypass Assist Devices," *IEEE Trans. on Automat. Contr.*, vol. AC-30, pp. 322-329.

Middleton, R. H. and G. C. Goodwin (1990): *Digital Control and Estimation; A Unified Approach.* Prentice-Hall, Inc. A Division of Simon and Schuster, Englewood Cliffs, New Jersey 07632.

Monopoli, R. V. (1974): "Model reference adaptive control with an augmented error signal," *IEEE Trans. Automat. Contr.*, vol. AC-19, pp. 474-484.

Morse, A. S. (1980): "Global stability of parameter adaptive control systems," *IEEE Trans. Aut. Control*, vol. AC-25, pp. 433-439.

Motorola (1989): *DSP56000ADS Application Development System User's Manual, version 2.00*, January 23, 1989. Semiconductor Products Sector, Phoenix, Arizona.

Motorola (1989): *DSP56ADC16 Evaluation Board User's Manual. revision 1.0.* Semiconductor Products Sector, Phoenix, Arizona.

Motorola (1990): *DSP56000/DSP56001 Digital Signal Processor User's Manual.* Semiconductor Products Sector, Phoenix, Arizona.

Narendra, K. S., Y. H. Lin and L. S. Valavani (1980): "Stable adaptive control design," *IEEE Trans. Aut. Control*, vol. AC-25, pp. 440-448.

National Instruments (1990): *AT-MIO-16 User Manual.* August, 1990 Edition, Part Number 320146-01. National Instruments Corp. Austin, Texas.

National Instruments (1991): *DOS LabDriver Software Reference Manual, Version 4.0.* April 1991 Edition, Part Number 320273-01. National Instruments Corp. Austin, Texas.

Park, S. (1990): *Principles of Sigma-Delta Modulation for Analog-to-Digital Converters.* Publication No. APR8/D, Motorola Semiconductor Products Sector, Phoenix, Arizona.

Parks, P. C. (1966): "Liapunov Redesign of Model Reference Adaptive Control Systems," *IEEE Trans. Automat. Contr.*, vol. AC-11, pp. 362-367.

Peterka, V. and K. J. Astrom (1973): "Control of multivariable systems with unknown but constant parameters," *3rd IFAC Symposium on Identification and System Parameter Estimation*, pp. 535-544, The Hague, Netherlands.

Radke, F. (1987): "Microprocessor Based Adaptive PID Controllers," *Proc. of the ISA International Conference and Exhibit*, vol. 42, no. 3.

Radke, F. and R. Isermann (1987): "A Parameter-adaptive PID-controller with Stepwise Parameter Optimization," *Automatica*, vol. 23, pp. 449-457.

Rey, G. J., R. Johnson, Jr., and R. Bitmead (1990): "Nonlinear Interactions between Signals and Parameters in Robust Adaptive Control," *Proc. 1990 Automat. Contr. Conf.*, pp. 1064-1069.

Rootzen, H., and J. Sternby (1984): "Consistency in least squares: A Bayesian approach," *Automatica*, vol. 20, pp. 474-475.

Schuck, O. H. (1959): "Honeywell's history and philosophy in the adaptive control field," in P. C. Gregory, Ed., *Proc. Self-Adaptive Flight Control Systems Symp.*, WADC Tech. Rep. 59-49 (Wright Air Development Center, Wright Patterson Air Force Base, Dayton, Ohio.

Sin, K. and G. Goodwin (1982): "Stochastic adaptive control using a modified least squares algorithm," *Automatica*, vol. 18, pp. 315-321.

Smith, C. L. and P. W. Murril (1966): "A More Precise Method for Tuning Controllers," *ISA Journal*, May, 1966.

Sternby, J. (1977): "On consistency for the method of least squares using martingale theory," *IEEE Trans. Automat. Contr.*, vol. AC-22, pp. 346-352.

Sternby, J. and H. Rootzen (1982): "Martingale theory in Bayesean least squares estimation," *Proc. 6th IFAC Symposium on Identification and System Parameter Estimation*, Arlington.

Tan, J. and N. Kyriakopoulos (1988): "Implementation of a Tracking Kalman Filter on a Digital Signal Processor," *IEEE Transactions on Industrial Electronics*, vol. 35, no. 1.

Tsypkin, Y. Z. (1971): *Adaptation and Learning in Automatic Systems*. Academic Press, New York, New York.

Warwick, K., K. Z. Karam and M. T. Tham (1987): "Simplified Parameter Adaptive Control," *Optimal Control Applications & Methods*, vol. 8, pp 37-48.

Wellstead, P. E. (1978): "On the self-tuning properties of pole-zero assignment regulators," Report 402, Control Systems Centre, The University of Manchester Institute of Science and Technology, Manchester, England.

Whitaker, H. P., J. Yamron, and A. Kezer (1958): "Design of Model Reference Adaptive Control Systems for Aircraft," Rep. R-164, Instrumentation Laboratory, MIT, Cambridge, MA.

Wittenmark, B. W. (1988): "Implementation and Application of Adaptive Control," *IFAC Adaptive Control of Chemical Processes*, Copenhagen, Denmark.

Wittenmark, B. W. (1979): "Self-tuning PID-controllers based on pole placement," Dept. Automat. Contr., Lund Inst. Technol., Lund, Sweden, Tech. Rep. CODEN: LUTFD/(TERT-7179)/1-037, 1979.

Wittenmark, B. W. and K. J. Astrom (1980): "Simple self-tuning controllers," *Proc. Symp. Methods and Applications in Adaptive Control*, Bochum, F.R.G.

Ziegler, J. G. and N. B. Nichols (1942): "Optimum Settings for Automatic Controllers," *Trans. of the ASME*, November, 1942.