ADAPTIVE CONTROL OF AN INDUSTRIAL ROBOT WITH APPLICATION TO ARC WELDING

By

© AJIT M. KARNIK, B.Tech., M.Tech.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

August 1985

ADAPTIVE CONTROL OF AN INDUSTRIAL ROBOT

TO MY PARENTS

WITH GRATITUDE AND LOVE

DOCTOR OF PHILOSOPHY (1985)
(Electrical and Computer Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE: Adaptive Control of an Industrial Robot with Application To Arc Welding

AUTHOR: Ajit M. Karnik, B.Tech. (Indian Institute of Technology, New Delhi)

M.Tech. (Indian Institute of Technology, New Delhi)

SUPERVISOR: Professor N.K. Sinha

NUMBER OF PAGES: xiii, 194

ABSTRACT

This thesis addresses the problem of position control of an industrial robot. The robot used for this study is a modified UNIMATE-2000 with five degrees of freedom, which is dedicated to research in arc welding applications.

The problem of position control of an industrial robot is first analysed. It is evident that due to several factors such as the elasticity of the arm, gravitational effects due to link orientation and load variations, a conventional fixed feedback controller is not adequate. It is therefore necessary to use an adaptive control scheme for position control.

It is well known that the dynamics of a robot are non-linear and coupled, because of which it is not convenient to derive control laws which can be implemented in real time. Further, because the dynamics change as a function of link orientation and load variation, the model has to be evaluated on-line.

In the past, robots were modelled from laws of classical mechanics. Although this scheme can result in accurate models, they are not suitable for adaptive control applications because of their complexity. In addition, this scheme of modeling requires a-priori knowledge of mass of links, specification of servo amplifiers and actuators. Often, to simplify models, dynamics of actuators and elasticity of links is neglected which results in unsatisfactory control.

In this thesis we describe a new method for modeling the robot. The model is derived from experimental observations and does not require knowledge of the structure and internal subsystems of the robot. Further, it includes actuator dynamics and the elasticity of the arm. The method is demonstrated by modeling one axis of the UNIMATE-2000.

Since the model is of non-minimum phase, the explicit pole-placement type of a self-tuning regulator was designed. Parameters of the model are updated at every sampling interval using a recursive least squares estimator. Since, as mentioned before, dynamics of the robot change as a function of link orientation and load variation, a 'weighting factor' is used for parameter estimation. Studies presented include the effect of truncation of controller parameters, adaptivity of the regulator to changes in system dynamics, and the effect of a variable weighting factor.

Finally, several aspects of implementation of the self-tuning regulator, including selection of the controller structure, selection of the microprocessor, A/D, D/A converter and programming language are discussed.

ACKNOWLEDGEMENTS

The author is deeply indebted to his supervisor, Professor N.K. Sinha, for his guidance and encouragement throughout the course of this work. Gratitude is also extended to Prof. R. Kitai, Prof. W.R. Newcombe and Prof. J. Tlusty as members of the Supervisory Committee.

The financial assistance from the Commonwealth Scholarship plan and the opportunity provided by the Ministry of Education and Culture and the Department of Electronics, Government of India is gratefully acknowledged. Thanks are due to my colleagues, Dr. W. Zaton, Mr. J.R. Raol, Mr. S. Puthenpura and Mr. F. Omani for many helpful discussions.

The typing skills of Mrs. Debbie Harris in the preparation of this thesis are greatly appreciated.

The author especially wishes to express his appreciation to his wife, Nandini, for her patience, understanding and the innumerable sacrifices she made during this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Since the beginning of industrialization, scientists and engineers have striven to increase and improve production in a cost effective manner. In the past two decades, the strategy for increasing productivity has been the introduction of automation. More recently, automation has been in the form of industrial robots. The robot manipulator system is defined as a programmable, multifunction manipulator designed to move material, parts, tools or specialized devices through programmed motions for the performance of a variety of tasks. In the literature, the terms mechanical arm, artificial hand, robotic arm, open articulated chain and manipulator are used interchangeably.

In general, the robot manipulator system is composed of several rigid links, connected in series by kinematic joints, into an open loop kinematic chain. The joints are rotary or linear (sliding), with the first joint fastened to a fixed support (base), whereas the last link functions as the 'end effector', which holds the device performing the task. Each joint is assumed to have one degree of freedom, so the number of joints defines the degrees of freedom of the robot. A functional sketch of a five degree of freedom robot is shown in figure 1.1.

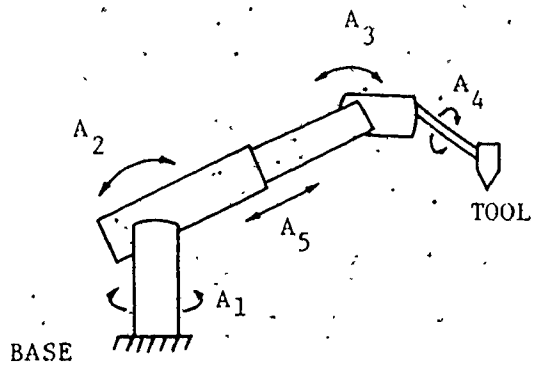


Fig. 1.1: A five degree of freedom robot.

The first commercial robot was manufactured in 1961, but robots were not introduced on a large scale until late 1970's. Since then, however, mainly due to technological advance in micro electronics, industrial robots are being used in a variety of applications in wide ranging areas [1].

Applications of robots in the industry can be classified in two main catagories [15].

1. Tasks involving high personal risk such as loading or unloading hot parts from furnaces, satellite repair in space; underwater exploration, handling of radioactive and toxic waste, etc.

2. Routine functions that have to be done over and over again, such as in a typical production environment. These jobs vary from simple 'pick and place' operations such as stacking boxes of paper or testing electric lamps to highly skilled jobs like spray painting and arc welding.

In general, the use of computer controlled robots offers a significant number of advantages. These include cost savings, reliability, tolerance to working environment and immunity from socio-economic and political problems.

For example, in the arc welding process, the heat required to fuse the metal surfaces together is derived from an electric arc. When the arc is struck, the temperature in the vicinity rises rapidly to about 6500^{o}F. This heat causes a small pool of molten metal in the workpiece and the end of the electrode also melts to supply additional filler metal. The area surrounding the arc is flooded with argon or carbon dioxide to protect the weld from the atmosphere to inhibit oxidation. When performed manually, human judgement is required throughout to position the electrode at an optimum distance from the workpiece, and to move it along the line of weld at an even rate to produce good flow of molten metal between surfaces. It is evident that the whole procedure is subject to human variations and error. In addition, the welding process creates a very unpleasant and unhealthy working environment. Considerable ultraviolet radiation is present in the glare from the arc, requiring the operator to wear dark glasses for preventing damage to eyes [2]. Smoke, gas and sparks have also to be contended with.

Since the robot is undaunted by the environmental problem, and under normal operation is incapable of any deviation from a well defined task once programmed, it is ideally suited to the arc welding process.

In the analysis of robotic manipulators, there are three main problems that need to be solved. The first is that of coordinate transformation. The desired position and orientation of the end effector in the cartesian (world) coordinate space are given or calculated, and the problem is to find the positions of individual joints which will result in the robot reaching the desired position. If the number of degrees of freedom of the robot is more than the number of coordinates (and orientation) specified, there may be an infinite number of solutions. However, if the number of degrees of freedom is the same as the coordinates specified, the solution is unique and the joint position may be determined from the inversion of the geometric transformation which relates the joint variables to the end effector position and orientation. Several references for such transformations are available [2,4-10]. Throughout this work, however, it is assumed that an appropriate solution for this problem exists [9].

The second problem is that of dynamic control of the manipulator. This involves determining the structure of the controller and the generation of a control signal which directs the movement of the manipulator system from its present configuration to the desired one. This thesis deals with this control problem.

In general, from a position control point of view, tasks performed by an industrial robot can be divided into two categories:

(1) point-to-point type; such operations include stacking boxes, drilling printed circuit boards, spot welding, etc., and

(2) continuous path type; typical applications in this catagory are spray painting and arc welding.

The third aspect refers to the control of the process itself. In case of arc welding, the most important consideration is the quality of weld. This is judged in terms of the fusion of workpieces, and penetration of the molten metal. Often, as described in later chapters, it is possible to combine the two control problems, that of position control and the process control. In the arc welding process, it is possible to use a single sensor which gives information about the accuracy of positioning the end effector, the temperature and size of the weld pool. However, in general, these issues can be dealt with independently; and for this work it is assumed that process related parameters are appropriately controlled.

A schematic drawing of the block diagram of the operating environment for an industrial robot is shown in figure 1.2.

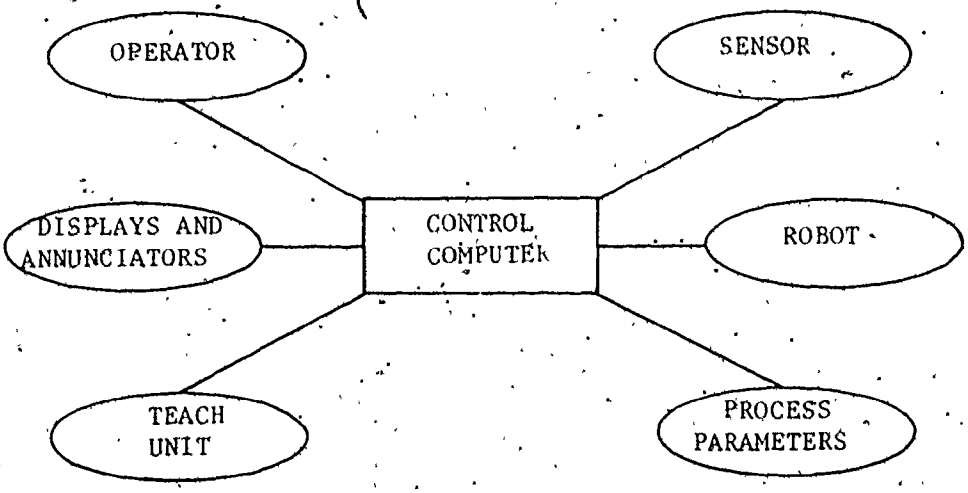


Fig. 1.2: Operating environment for an industrial robot.

A common method for obtaining a desired motion is to first lead the end effector through the desired path with the help of a 'teach pendant'. The control computer 'records' the control signals required for several points on the path. Recorded values are used subsequently to carry out the process. This method is slow and unsuitable for most modern applications.

In the other method for obtaining the desired motion, the operator programs several points (usually corner points), through which the tool attached to the end effector must pass. The control computer, in real time, calculates and generates control signals required to move the joints of the robot to produce the desired motion.

The control problem consists of (a) obtaining the dynamic model of the manipulator, and (b) using this model to determine control laws and generate signals to achieve the desired motion.

The current approach, as evident from commercially available systems [11,91], for robot arm control is to treat each joint as a simple servo mechanism. The servo is controlled with velocity feedback from a tachogenerator and position feedback from a position encoder, mounted on the motor shaft. Fixed gains are used in the control loop. However, because the industrial robot is inherently nonlinear, the control system described above is inadequate. The nonlinearity in the robot is due to inertial loading, coupling between joints, backlash in gear trains, and gravity loading of the mass of arms. Furthermore, the structure of the robot varies significantly depending on the position of various links and the mass of the

load carried by the end effector. It is therefore easy to see that conventional feedback, when used with robots will result in undesirable oscillations of the end point due to factors such as the elasticity of links, backlash and inertia. The result is reduced operating speed of the manipulator. Any significant performance gains in the control of the robot arm require the consideration of elaborate dynamic models and sophisticated control techniques.

In the past decade, many schemes for deriving a dynamic model of the robot have been proposed. Most of the approaches are based on representing the robot as a multi-input multi-output, coupled, non-linear set of differential equations. These equations, usually written in a matrix form, are derived from laws of classical (Newtonian) mechanics. [4,5,7,8,10,13]. The complexity of these equations makes it impossible to derive analytical control laws. Furthermore, since the model of the robot varies due to link orientation and load variation, models used for the control strategy have to be evaluated for each control cycle. Even with on-line numerical techniques, this is a considerable computational load requiring a large amount of memory and execution time.

For fast and efficient control of robots, it is therefore necessary to simplify models using techniques such as linearization and decoupling.

Several methods have been proposed, but in all cases the computational time required is still large and they are therefore not suitable for implementation with high speed manipulators. Moreover, they do not account for payload variation and link orientation. In addition, these models require the knowledge of several parameters of

the robot, such as, the mass of links, friction and backlash in gears and models for electric motors.

It is clear that if any significant improvement in tracking the desired trajectory, as closely as possible over a wide range of robot arm and load variation, is to be achieved, one would have to take recourse to adaptive controllers.

An adaptive control scheme using a model reference is proposed in [12]. A linear, second order time invariant, underdamped model was used as references for the output of each joint. The manipulator is controlled in a signal synthesis type model reference adaptive controller [MRAC], so that the difference between the output of the actual system and that of the reference is minimized. Coupling terms between joints are neglected; thus, simplifying the models of the manipulator. Simulation studies [12] demonstrate the feasibility of the approach. In the MRAC, the complicated model of the manipulator based on Newtonian mechanics is not directly used. However, since the MRAC scheme assumes perfect cancellation of the robot model, stability cannot be guaranteed in practice. This problem has also been discussed in Chapter 5.

Another approach was proposed [10] in which a second order AR model was used to represent the manipulator. The model was not obtained directly from input-output data of the robot. The dynamics of the manipulator were first modelled as nonlinear, coupled differential equations of high order, using classical mechanics. This model was then used to simulate the robot. Further, it is not clear if the model accounts for the elasticity of the arms, motor drives, gear drives, and belt couplings.

1.2 Organization of the Thesis and Contributions

In this section, we discuss the objective; organization and major contributions made in this thesis.

The thesis focuses on deriving accurate dynamic models of the robot from samples of input and output data and the use of these models to design adaptive controllers to control the positioning of the robot. Guidelines for the implementation of the adaptive controller are also discussed.

The thesis is organized as follows:

In Chapter 2, we describe various components in the robot system and describe their operation. Specifically, the structure of the robot, the welding equipment and the existing controller are described.

In Chapter 3, we first discuss problems encountered when using conventional position control methods. Several sources of error are identified, both from the point of view of the position control of the end effector and from the point of view of the weld quality. A brief review of various types of position sensors is included.

Also included in this chapter is a review of various methods of position control. A new method, in which an auxiliary robot is used for 'vernier' movements is described. A conventional feedback scheme using a proximity probe to sense the deviation from the seam to be traced was implemented. A brief description of the implementation is given. Since for adaptive control, a model of the process is necessary, the chapter concludes with a review of system identification and parameter estimation techniques.

In Chapter 4, we discuss the modeling of the robot arm from experimental observations. Details of the four steps involved in modeling a system using a 'black-box' approach, viz. data acquisition, structural identification, parameter estimation and model verification, are discussed in detail. A direct method of obtaining a continuous-time model from samples of input-output data is developed.

In Chapter 5, after a review of adaptive control methods, we describe the design of a pole-placement type of an adaptive controller. Two commonly used structures are evaluated for robustness. Several simulations are presented showing the adaptivity of the controller to parameter changes in the system. A study of the effects of including a variable weighting factor and the effects of numerical truncation of controller parameters on the performance of the regulator are presented.

In Chapter 6, we present a survey of microprocessors, analog to digital converters, digital to analog converters, shaft encoders and programming languages from the point of view of implementing the adaptive controller. Guidelines are provided for the selection of these components and suggestions regarding the structure of the control system are made.

In Chapter 7, we present conclusion and discuss topics for further research.

Some of the contributions made in this thesis are:

1. Successful application of a pole-placement type of adaptive self tuning regulator for the position control of the robot

arm. It is shown by simulation studies, that this regulator enhances the performance of the robot system significantly in terms of positioning accuracy and sensitivity to changes in system parameters. Details are given in Chapter 5.

2. Development and testing of a seam tracking scheme for the robot, using a proximity probe sensor and an LSI-11/23 microcomputer.

3. Development of a general method for identifying a continuous time system directly from samples of input-output data. An advantage of this scheme is that it does not require the input to be persistently exciting; also it can be used with step, ramp and other deterministic inputs [Chapter 4].

4. Conducted experiments for data acquisition for the welding robot and used this data for modeling. In this way, the robot is treated as a black box and no a-priori knowledge of the hardware is required. In addition it also accounts for the elasticity of limbs, actuator and gear drive characteristics.

5. Design, simulation and comparison of two commonly used structures of the pole-placement type self-tuning regulator. It is shown that although the two structures are almost identical from the design point of view, their performance is significantly different with respect to the numerical precision used for computation [Chapter 5].

6. Simulated cases were used to study the adaptivity of the regulator to changes in the system dynamics, and the performance of a binary switching of the 'weighting factor'. Results indicate a significant improvement in the rate of adaptation and accuracy in position control.

In addition, a comprehensive survey of the existing methods of position control of a manipulator was conducted. Their merits and demerits were analysed and a new method involving an auxiliary robot is suggested.

CHAPTER 2

COMPONENTS AND OPERATION OF THE ROBOT SYSTEM

2.1 Introduction

In Chapter 1, we describe why the arc welding process is suitable for automation. Justification was based on the fact that welding produces toxic and carcinogenic fumes, splatters of molten metal and intense radiation that may cause eye injury and skin cancer. The welder has to take up awkward position in hot, confined spaces while being weighted down with equipment and protective clothing.

Flexible devices, such as industrial robots are idealy suited for automating the arc welding process. Problems related to automated arc welding can be divided into two major catagories - torch position related and arc process related. Both of these issues are discussed in detail in the next chapter. In this chapter we describe various components of the robot system used for arc welding and their operation.

Basically, the system consists of a continuous round wire fed from a spool through the torch. This wire serves as a consumable electrode. The arc generated between the tip of the wire and the workpiece melts the electrode (wire) and some of the workpiece. Molten wire serves as filler metal to fuse the joint. The torch connected to the end of the robot arm is made to move along the seam at a uniform velocity to get an acceptable weld.

Details of various subsystems, operator interface and safety features are given below:

2.2 The Arc Welding Robot

The robot used for arc welding is a redesigned UNIMATE-2000. It has five degrees of freedom, as shown in figure 2.1. As indicated in table 2.1, four of the five axes are 'rotational' type and one is linear.

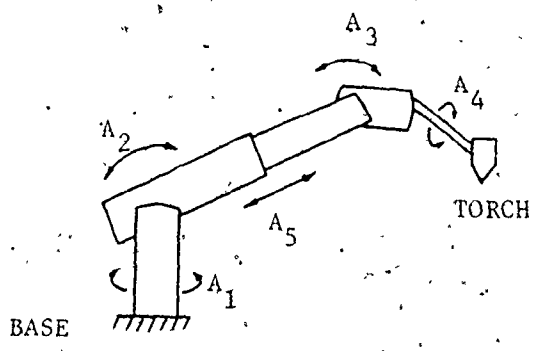


Fig. 2.1: The arc welding robot.

The robot was originally hydraulically actuated and was equipped with absolute (gray scale) encoders in the positional feedback loop. This configuration had serious problems with repeatability of the robot motion. Hydraulic actuators were replaced with electrical (DC) servomotors and the absolute encoders were replaced by relative encoders. Electrical drives required a mechanical drive train with a large (200:1) speed reduction. An initial design using worm gears proved unsuitable because of backlash in the worm gears which affected repeatability. They were therefore replaced by

"harmonic" drives which provide large speed reduction with negligible backlash.

Table 2.1: Description of Robot Axes

Axis	Type	Motion
A_1	Sweep	Rotational
A_2	Lift	Rotational
A_3	Bend	Rotational
A_4	Swivel	Rotational
A_5	Extend	Linear

In general, each axis of the robot is equipped with a d.c. motor coupled to a tachogenerator, an incremental position encoder and a harmonic drive.

A welding torch is mounted at the free end on the robot.

2.3 Welding Apparatus [15,18]

The process selected for robot welding is the 'gas-metal arc welding' (GMAW), or metal-inert-gas (MIG) technique; it is commonly used in most automatic and semi-automatic welders. The equipment consists of a Hobart MEGA-MIG 450-RVS, constant-voltage-rectifier welding machine with a MEGA-CON model 111 welding control unit. This control unit drives a Hobart model H4S wire drive feed head. A

continuous round solid wire feeds through a model GAL-400 welding gun fixed to the end of the robot. The MEGA-CON welding control unit is interfaced to the main robot controller through optically isolated switches. Welding parameters - the wire feed rate and the voltage, are the two main parameters directly controlled through software. The welding current is indirectly regulated by the wire feedrate. If the wire is fed quickly, the current increases and if fed slowly, the current decreases. The shielding gas used is carbon dioxide.

In manual welding, coordination of various parameters is done by a skilled operator, but in the case of robot welding it is done by the controller software. The wire feed rate and welding voltage is set by software, and is coordinated with the travel speed and electrode to workpiece distance of the welding gun.

2.4 Control System [3,16,17]

The Master Computer:

As described earlier, each axis of the robot is equipped with a d.c. motor, a tachogenerator and a position encoder. The system controller is built on a two level hierarchy. The lower level is called the 'slave' controller and the higher one is called the 'master' controller. The master controller is built around the Intel iSBC 86/12A single board computer combined with the 80 bit, data processor "iSBC 337". Key I.C.s in the master controller are the 8086 and 8087 microprocessors. The 8087 is used for high speed fixed and floating point functions. Responsibility of the master controller includes communicating with the operator and issuing commands to the

welding controller described earlier and the slave controllers.

The overall interconnection diagram of the master and slave controllers is shown in figure 2.2.

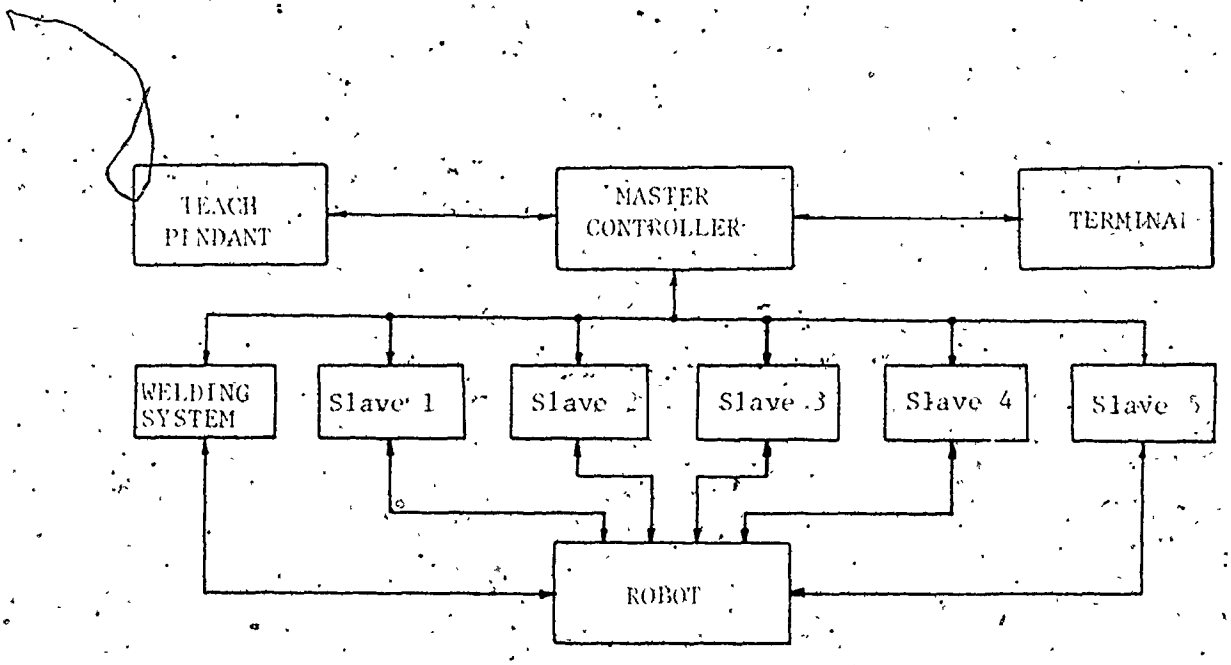


Fig. 2.2: Robot Controller.

The Slave Controller:

There is one slave controller for each axis. Except for their addresses on the master bus, they are identical in operation. Slave controllers are built around the 8 bit Intel 8748 microprocessor. It is the responsibility of this controller to generate the velocity error following a command from the master controller, generate an analog voltage to drive the motor, and to stop the motor from rotating once that axis has reached the commanded position. Feedback from the position encoder is used to determine the positional information.

Various components on the slave controller are shown in block diagram form in figure 2.3.

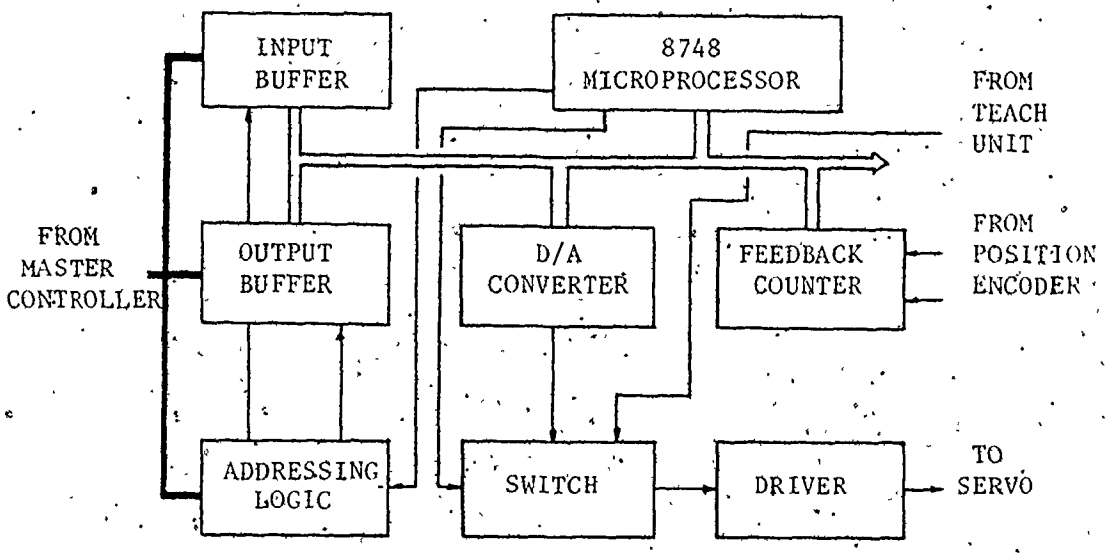


Fig. 2.3: Simplified block diagram of slave controller.

Since the 8748 is an 8 bit microprocessor, it has to communicate with the 16 bit master computer through 16 bit buffers. Data from the 'master' has to be read in 2 bytes by the slave. Similarly, the slave writes two bytes, which are read by the master computer.

Once the path coordinates are programmed as described in section 2.5, the master computer is set in the 'run' mode. The master computer computes intermediate points in 'world' coordinates and transforms them to 'joint' coordinates. The amount of motion required for the joint (in joint increments of motion), is passed to slave controllers every 20 to 30 milliseconds. The slave controller

compares the value of these increments to the actual (incremental) position recorded by the position counter to generate the error. This error is converted to a velocity signal to drive the motor to null error. A block diagram showing the motor drive and feedback details for a typical axis is shown in figure 2.4. Components enclosed in the box are a part of the slave controller card.

The switch shown in figure 2.3 is used to allow manual movement of the robot from the teach pendant.

2.5 Operator Interface

The operator communicates with the robot system through two devices. The operators console or terminal is used to input all commands and setting parameters. The terminal is also used for display and editing of program sequences and path coordinates.

After power is turned ON, the operator initializes the robot to its 'home' position. At this point, he may select one of several modes of operation. The robot may be commanded to weld along a preprogrammed path or a new path may be entered. Corner points of the 'path' are input using the teach pendant. The robot is first moved to the desired position and coordinates of the point are saved by pressing the 'record' switch. After all points have been recorded, the operator edits/saves them and writes a program (sequence) specifying whether a particular path is to be welded or not. When 'WELD ON' is specified, the speed of wire feed and the voltage have to be input. The program is executed after ascertaining its correctness.

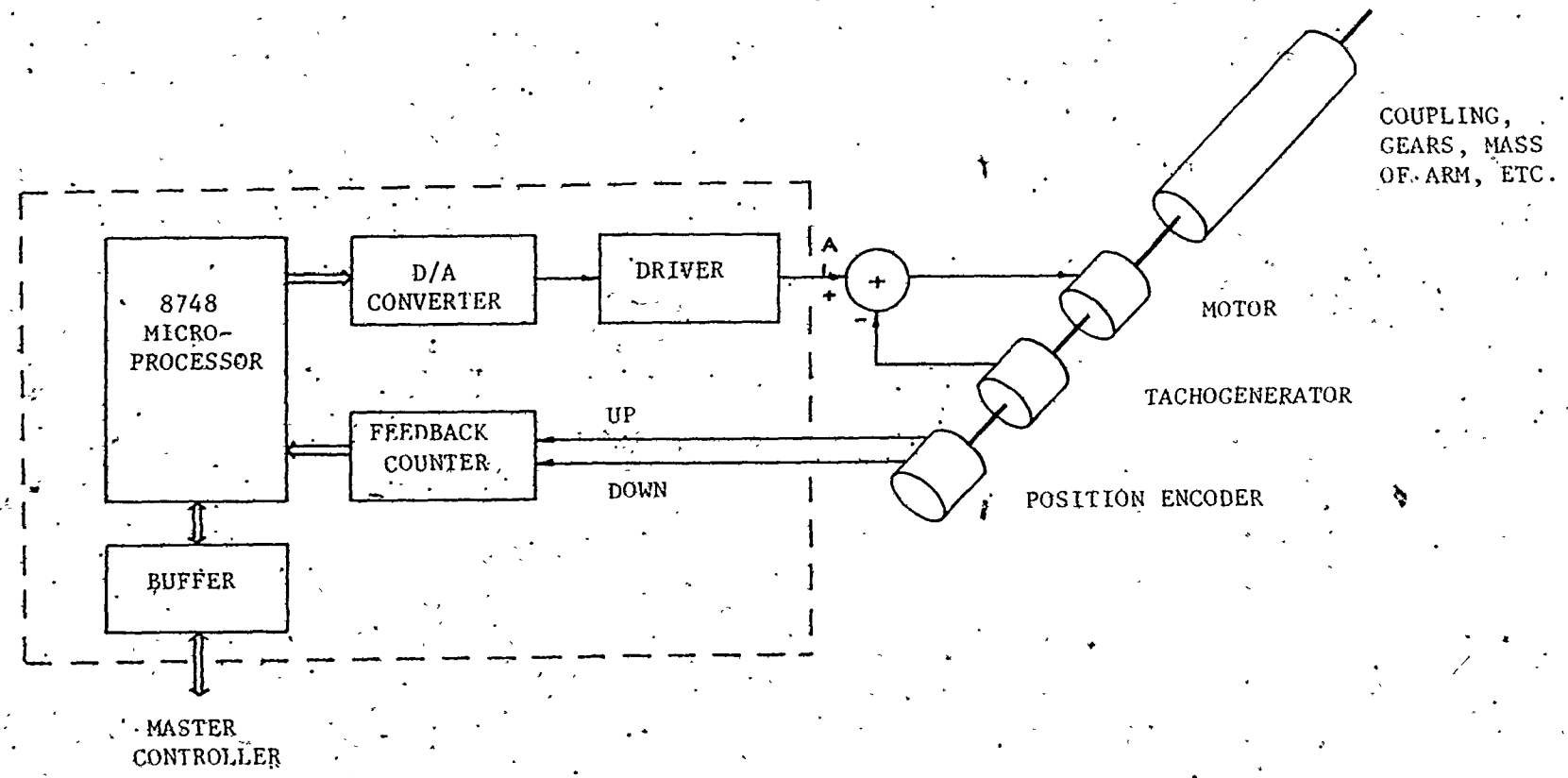


Fig. 2.4: Slave controller-robot interface.

A new, high level, robot control language has been developed, which allows the operator to use the robot in a 'user friendly', interactive way. Commands are menu driven and extensive ON-LINE help is provided. Details of this package are given in [19].

2.6 Safety Requirement

Safety of personnel operating the robot is an important consideration for any robot installation. The swing of the robot arm is presently limited by software limit switches. Provision for hardware limit switches has been made and are to be installed shortly. A "dead man's" switch, which disconnects power to the servo amplifiers when released has been installed. This switch is always in use when the robot is operating, since it is spring loaded with normally open contacts. A brake has been installed and plans are underway for the installation of a steel fence around the robot work area.

2.7 Concluding Remarks

In this chapter we describe various components of the robot system. The robot used for arc welding has five axes, of which four are of the revolute type and one is linear. The gas metal arc welding (GMAW) process has been selected because of its suitability for automatic welding. A two level controller is used to control the movement of the robot. A 16 bit master controller provides the interface between the operator and each of the five axes. Each axis is controlled by a 'slave' controller based on an 8 bit microprocessor. Since the safety of personnel is an important consideration, various safety features incorporated are discussed.

CHAPTER 3

SYSTEM IDENTIFICATION AND CONTROL -- A REVIEW

3.1 Introduction

In Chapter 2 we described various components of the robot system. The operation of the control system was given in section 2.4. End coordinates of the path to be followed were input to the controller, which interpolated intermediate points, converted them to the joint coordinates and also generated voltages required to drive the actuators. It was implicitly assumed that the robot would then result in motion along the desired path.

When the welding unit is turned ON, this motion should result in the desired weld. Unfortunately in practice, this is not always the case and non-ideal welding is achieved.

In this chapter we discuss reasons for this degradation in quality and review various methods to improve the performance. The chapter is organized as follows. In section 3.2, we outline causes of error. Basically, there are two types of error, (1) error in positioning the robot on the seam, and (2) errors in setting the weld parameters. Both of these aspects are discussed. In section 3.3, a brief survey of various methods used for sensing the position of the seam and quality of weld are discussed. Section 3.4 describes several methods for controlling the motion of the robot. A scheme using a

proximity sensor to detect the position and an LSI-11 in a feedback loop was implemented. Details of this scheme are also included in this section.

Adaptive control schemes are considered for the position control of the robot. Since they are based on obtaining a dynamic model of the robot, the problem of system identification is reviewed in section 3.5.

Specific details on modeling the robot arm are given in the next chapter. In this section on system identification, a general review of various forms of modeling along with their merits and demerits are discussed.

3.2 The Control Problem

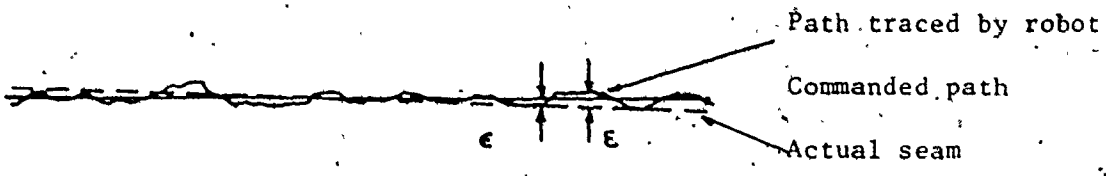
As described in section 2.4, the end point coordinates of the path to be traced are input to the master controller. The controller evaluates intermediate points by path interpolation, and converts them to equivalent joint coordinates. Command increments are then generated and transmitted from the master controller to the slave controller at periodic intervals. The slave controller accepts these increments and generates analog voltages to drive the motor. Feedback from the position encoder is used to ensure that the motor rotates as required by the command. Welding is performed when the welding unit is turned ON as the robot moves along the desired path.

The control of arc welding process is directed towards two general objectives. One objective is to maintain the point of arc at the proper location relative to the seam. This is commonly referred

to as joint or seam tracking or 'position control'. The other aim is the control of the weld quality. This encompasses control of many variables which affect the quality of the finished weld. Both these control problems are discussed in this section.

The issue of position control is of interest not only in case of arc welding, but for many other robot applications such as gas cutting, painting, etc. Even in applications that require point-to-point movement, such as in spot welding, machine part assembly and drilling of printed circuit boards, etc., good position control is desirable for accuracy and speed.

Although voltages required for the motors are evaluated by accurate coordinate transformation, and the motion checked by feedback from the position encoder mounted on the motor shaft, the path generated by the robot is usually not in exact agreement with the planned path. This error shown in figure 3.1 is caused by several factors, some of which are listed in table 3.1, [15].



ϵ - error due to workpiece variation

E - gross error

Fig. 3.1: Positional error.

The above figure shows errors which are transverse to the seam. It is imperative that for acceptable welding, such positional

error be minimized and the welding be carried out on the seam.

Table 3.1: Types and Causes of Error

SNO	TYPE	CAUSE OF ERROR
1.	Structural	(a) Inertia of the arm (b) Backlash in gears (c) Elasticity of limbs
2.	Workpiece	(a) Piece to piece variation (b) Expansion and warping of the structure
3.	Algorithmic	(a) Accuracy of the model
4.	Computational	(a) Finite word length effect

The second aspect, that of weld quality control, refers to the control of many possible variables which measure the acceptability of the finished weld. A poor quality of weld and improper positioning of the point of application of the arc, both lead to a 'weak' joint. On close observation of the arc welding process, a comprehensive list of welding parameters and variables can be developed. Since many variables have significant effect on others, a tabular format is used to show the interaction. Table 3.2 lists the primary and secondary weld related parameters against the controller and unmanipulatable arc/process variables [61].

Table 3.2: Variables/Parameters Relevant to Weld Quality

ARC or PROCESS VARIABLES (Input) / WELD PARAMETERS (Output)			PRIMARY									SECONDARY							
			PENETRATION	BEADWIDTH	CROSS SECTION	SURFACE FINISH	HARDNESS	STRENGTH	SOUNDNESS(crack etc.)	RESIDUAL STRESS	POROSITY	* PEAK TEMPERATURE & DISTR.	COOLING RATE	* ACOUSTIC EMMISION	ARC GEOMETRY	ARC MOTION	POOL MOTION	RADIATED SPECTRAL COMP.	* ARC POSITION wrt POOL/WIRE
MANIPULATABLE INPUTS	PRIMARY	TRAVEL SPEED *	•	•	•	•	•			•		•	•						•
		WIRE SPEED *	•		•		•					•	•		•				
		ARC CURRENT *	•		•	•	•					•	•		•	•	•	•	
		ARC VOLTAGE *		•	•	•	•					•	•		•		•	•	
	SECONDARY	GAS FLOW	•		•	•		•			•	•	•		•	•		•	
		TORCH ANGLE *	•		•	•											•		•
		TORCH POSITION *	•	•	•	•											•		•
		WEAVE (motion)	•	•	•	•											•		•
		METAL TRANSFER	•	•				•	•		•	•			•		•		•
UNMANIPULATABLE INPUTS		PLATE GEOMETRY *	•				•			•		•	•	•			•		
		JOINT:																	
		Root face	•						•										
		Gap *	•						•										
		Inc. angle	•	•	•			•		•									
		Offset *						•		•									
		TACK WELDS	•					•		•						•	•		
		PHYSICAL PROP.																	
		Base					•	•	•	•	•	•	•	•		•			
		Filler						•	•		•			•		•		•	

* - INDICATES A VALID RELATION
• - INDICATES COMMONLY MEASURED VARIABLES

3.3 Position Sensing

As described above, one of the primary objectives in the control of the robot is to ensure that the path traced by the robot is exactly on the seam of the two pieces being welded together. Many different types of position sensing devices are available, however, most of them can be classified in the three general catagories described below [20-24].

1. Arc sensing: In this method, the robot is programmed to oscillate across the seam, and the location of the seam is determined by measuring the variation of the arc voltage and current. Such systems are commercially available, but are limited to welding certain types of joints. Also, the weaving effect caused by the oscillation, may not be desirable in all cases.

2. Tactile or proximity sensors: In this method a group of sensors connected to a pointer are made to trace the groove in the workpiece to be welded. Signals emitted by these sensors give a measure of the location of the seam. Most of the devices available commercially use tactile sensors, but some using 'non-contact' type sensors are also available. The problem with such position sensing devices is that because they 'feel' close to the point of welding, they are often affected by the weld splatter from the molten pool and arc.

3. Vision sensors: Vision sensors are popular since they have the advantage of being 'non-contact' type and also they do not require the welding torch to oscillate about the seam.

There are several variations in terms of the type of light sources used for illuminating the workpiece and the kind of sensors used for image pickup. But essentially, a T.V. camera and interface is used to quantify the light reflected by the workpiece. Some of the vision systems require special lamps for illuminating the object, while in some cases, the light emitted from the arc is used. The optical image picked up by a C.C.D. T.V. camera is digitized and analyzed to locate the position of the seam.

In some sensors, arrays of infra red sensing diodes are used instead of the T.V. camera. These do not require special illumination since they sense the position of the seam from the thermal distribution of the workpiece. When a line sensor is used, it is common to record the temperature profile, as shown in figure 3.2.

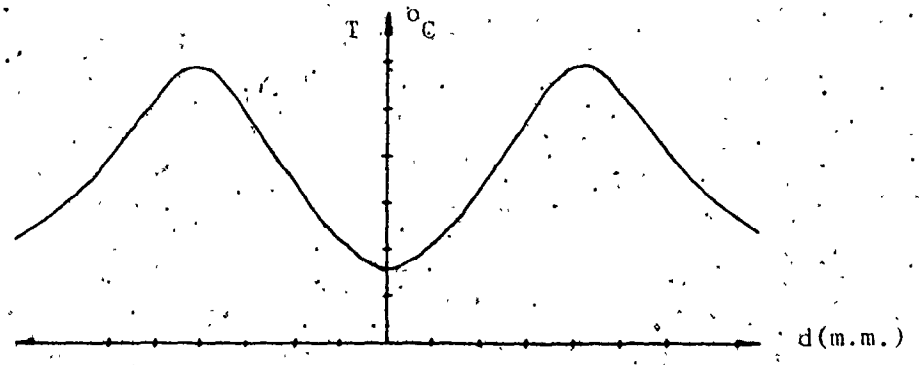


Fig. 3.2: Temperature profile.

For a rectangular grid type infra red sensor the image is digitized and analyzed by a vision processor for detecting weld pool features such as the pool width and isothermal contours [24]. Position information is obtained from the shapes of the isotherms as shown in figure 3.3.

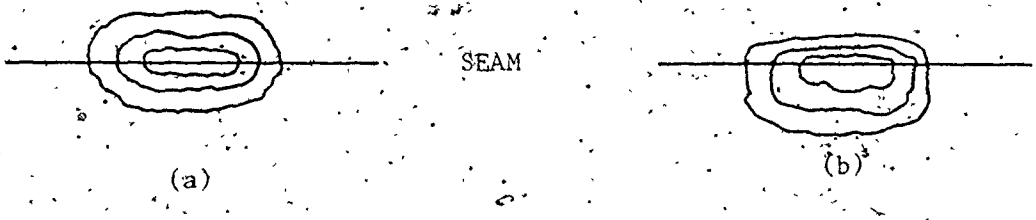


Fig. 3.3: Isotherms

(a) arc on seam, (b) arc offset from seam.

Isothermal contours are symmetric around the seam if the arc is being applied on the seam. A deviation from the seam results in asymmetric contours. An advantage of this method is that, it also provides a measure of the pool width, temperature, travel velocity, etc., which may be used to control the quality of the weld. One of the drawbacks of this type of sensor is that it is expensive and requires a high speed processor for real time applications.

Some manufacturers make infra red sensors that are mounted on the backside of the workpiece. These sensors have the disadvantage that they can be used only when the backside is accessible. Instead of measuring pool characteristics, 'backside sensors' can measure the penetration of the filler metal directly.

It is evident from this survey of sensors that a variety of devices for position sensing are available. Because of their relative merits and demerits it is not possible to pick out the most suitable sensor. The choice will be governed by the speed of the application and from economic considerations.

3.4 Methods of Control [15]

As mentioned earlier, the problem of position control is of prime importance for many applications of the industrial robot, including arc welding. Positional errors occur due to several reasons. Some of them are listed in table 3.1.

In this section, a survey of various methods for reducing these errors is given. One scheme which is based on the feedback of the end position for tracking the seam was implemented. Details of this scheme are also included.

1. Open loop method: In this scheme, an accurate dynamic model is constructed for the joints and links in the robot, which also includes the effect of the elasticity of the arm, backlash in gears, etc. Controls required are then evaluated using this model. Such a scheme suffers from several limitations. A very elaborate model may result if all factors are taken into account while modeling the robot. With such a complex model, the number of computations may be too many, and hence time consuming. The resulting system may not be useful for real time applications. On the other hand, if the model is simplified by making assumptions, such as assuming linearity, and neglecting the

coupling between joints etc., inaccuracies will exist, resulting in deviation from the required performance. Also, since this method is 'open loop', there is no check for system failure.

2. Off-line technique: In this method, a detailed dynamic model of the manipulator is not made, instead errors between the programmed path and the seam are 'learned' and compensated for. This requires two passes over the seam. The first pass is dedicated to 'learning' the controls required for proper positioning, and the second pass is used for the carrying the actual welding operation. The off-line method is also called a 'multipass' method because it requires more than one pass over the joint to be welded.

Since at least two passes are required for each job, the method is not as efficient as the one-pass methods. Also, it cannot account for errors due to the expansion of the structure.

3. Feedback control: Unlike the previous technique, this method performs on-line control. The difference between the actual seam and the path traced by the robot controlled welding gun is continuously measured and this error is fed back to modify the control signal. The welding torch is thus constrained to follow the seam. A block diagram representation of this method is given in figure 3.4.

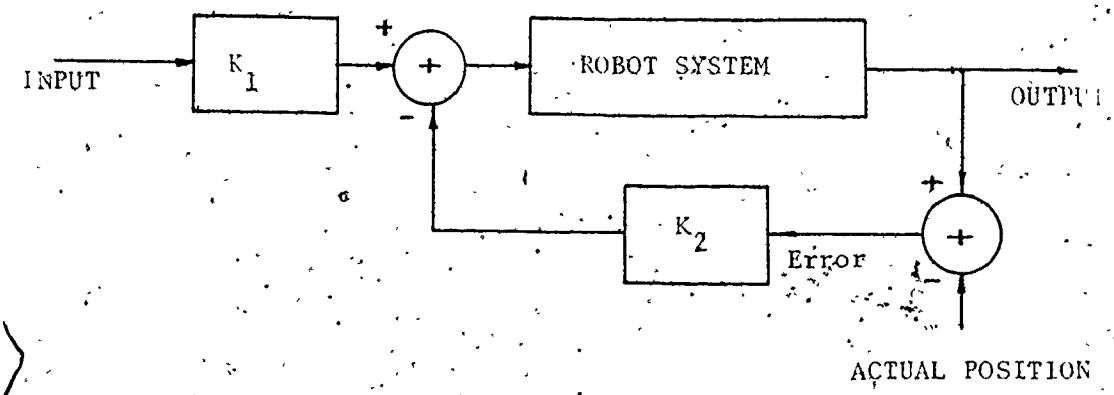


Fig. 3.4: Conventional feedback control system.

A controller based on the above was designed and implemented. The controller was used to force the robot to track a seam intentionally placed at an angle to the programmed path. Details of the scheme are given below.

The existing control system of the robot was described in section 2.4. The block diagram of the controller is reproduced in figure 3.5.

The position encoder is installed on the rotor shaft of the motor. Feedback from the position encoder is compared with the command signal to ensure that the motor rotates through the required angle so that the resulting motion is as desired. However, because this feedback is not from the position of the 'end point' of the robot arm, it does not reflect errors due to the inertia of the arm, its elasticity, errors in gear drives, etc. Also feedback from the position encoder does not account for workpiece positioning.

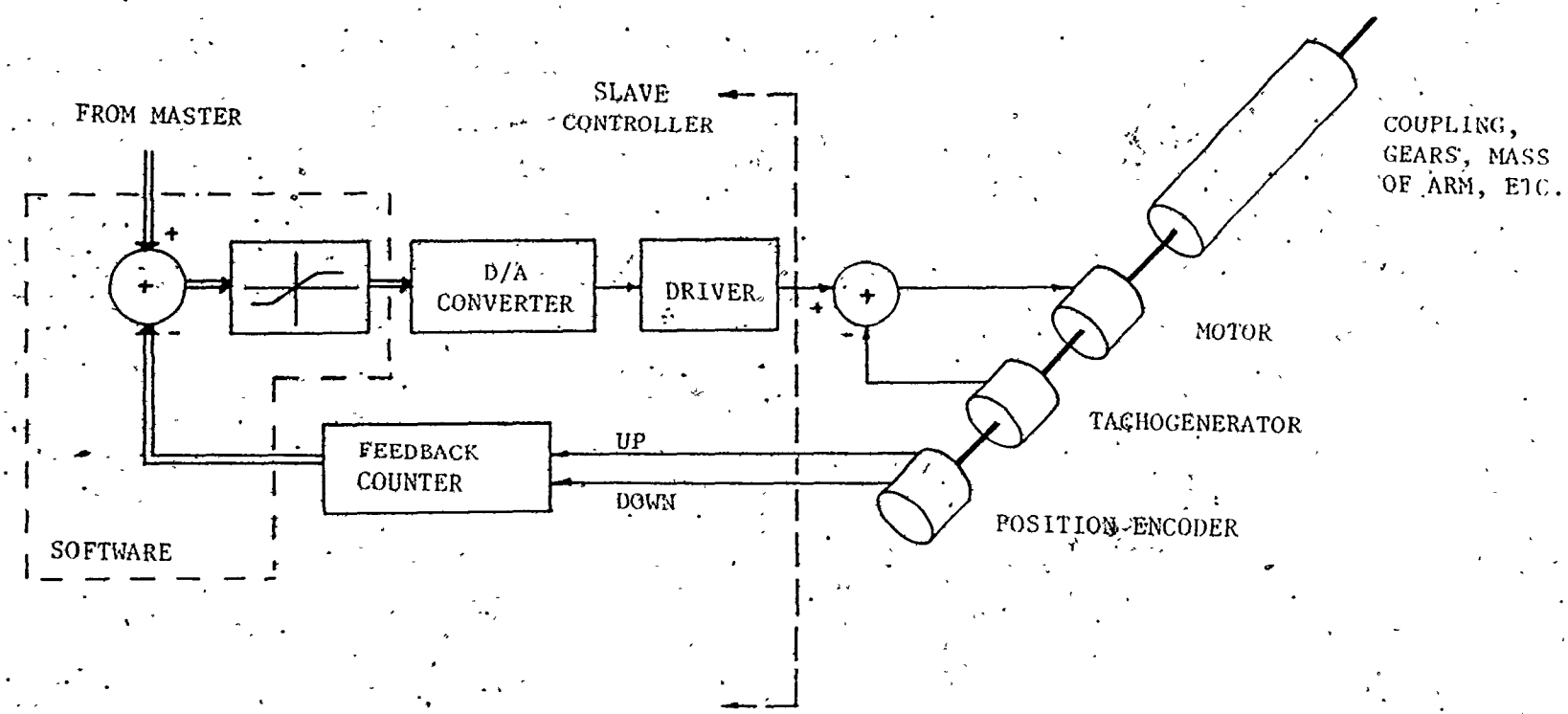


Fig. 3.5: Existing robot controller.

Using a proximity probe to detect the distance between the programmed path and the actual location of the seam, a feedback loop was added to the controller so that the robot could be forced to track the seam. A steel strip was placed at an angle across the programmed path of the robot, as shown in figure 3.6.

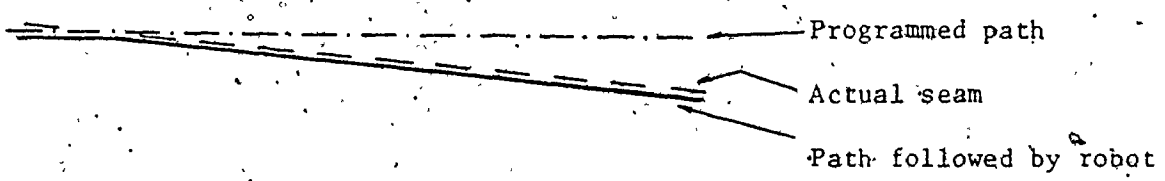


Fig. 3.6: Robot motion with additional loop.

This additional loop was controlled through software written in the LSI-11 assembly language. A block diagram of the overall control system with the addition is shown in figure 3.7.

Signals from the proximity probe and the output of the original controller were input to the LSI-11/23 microcomputer, through an AXV-11, analog-digital interface. A digital to analog (D/A) converter was used to generate a voltage to drive the d.c. motor. This additional loop operates as an ON-OFF controller, so that the distance measured by the proximity probe was kept within a pre-specified band. With the addition of this extra loop, the manipulator was forced to track the seam of the workpiece.

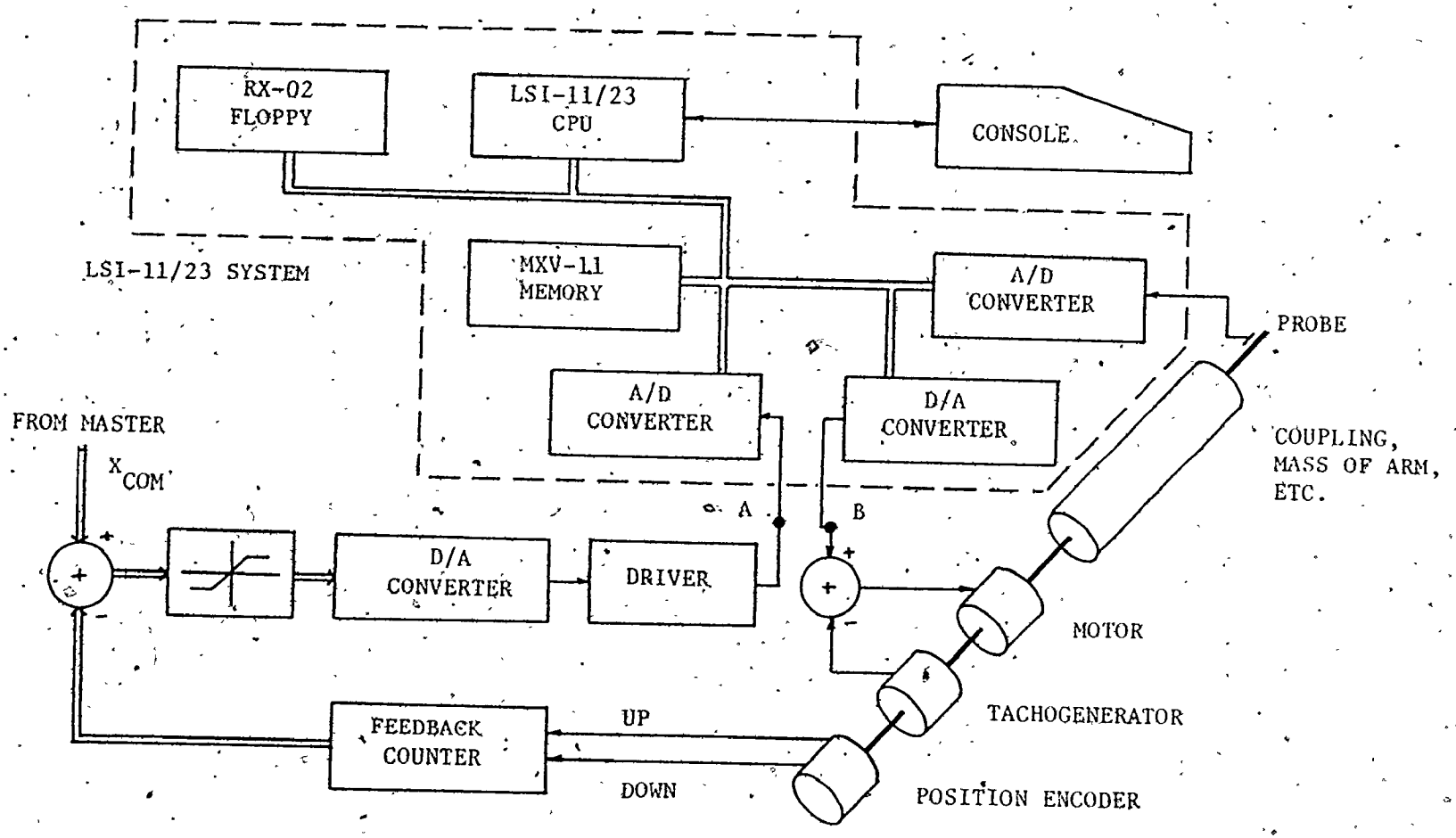


Fig. 3.7: Modified controller.

Since the controller described is not adaptive, it may prove inadequate in the presence of a large unpredictable disturbance and in case of structural changes or component failures. Unlike fixed control systems, adaptive control systems adapt to changes in the properties of the controlled processes and their signals. An adaptive controller may be more robust and has the advantage of 'graceful degradation' in case of component changes.

4. Adaptive controllers: An adaptive system measures a certain index of performance using the inputs and outputs of the adjustable system. From the comparison of the measured index of performance values and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable system or generates an auxiliary input in order to maintain the index of performance value close to the set of given ones. Adaptive systems have been classified into two distinct classes, although recent papers have shown that some equivalent properties exist between the two types of adaptive controllers. These two approaches can be summarized as follows:

(a) Explicit identification approach

In this approach, a model of the process to be controlled is identified in real-time, from the measurements of the input-output signals. Controller parameters are calculated from the process parameters obtained from the identification algorithm. Controllers based on this approach are also referred to as 'explicit self tuning regulators' and 'indirect controllers'. A block diagram of a self tuning regulator is shown in figure 3.8.

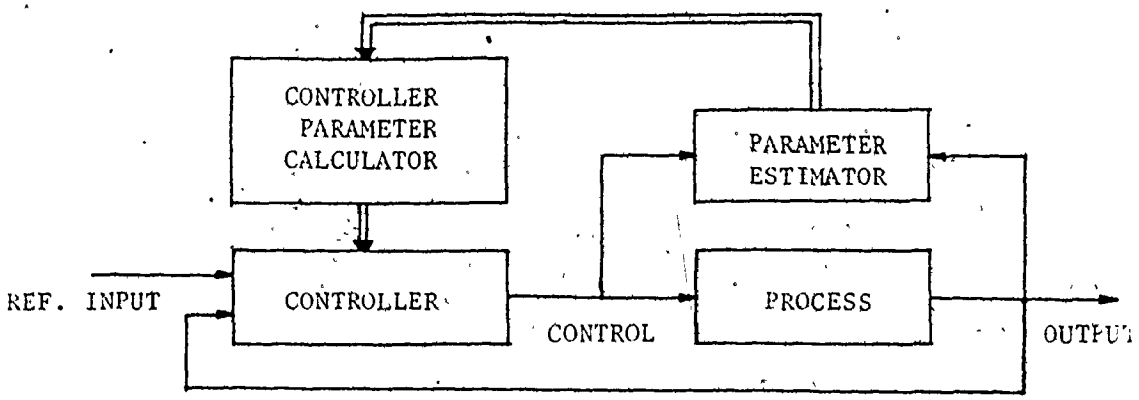


Fig. 3.8: Self tuning regulator.

Discrete time models and recursive parameter estimators are particularly suitable for implementing this type of regulators on digital computers. Detailed design and the performance of the self tuning regulator for the robot is given in Chapter 5.

(b) Implicit identification approach

Instead of an explicit identification of the plant, one may simply use an implicit method, i.e., one may adjust the parameters of the controller directly so that a performance index is satisfied. This is done more conveniently by using a reference model. Such systems are called model reference adaptive systems (MRAS). MRAS can be implemented in two ways: (1) Parameter adaptation method, and (2) the signal synthesis method. A block diagram of an MRAS is shown in figure 3.9. Dashed lines are applicable when the signal synthesis method is used.

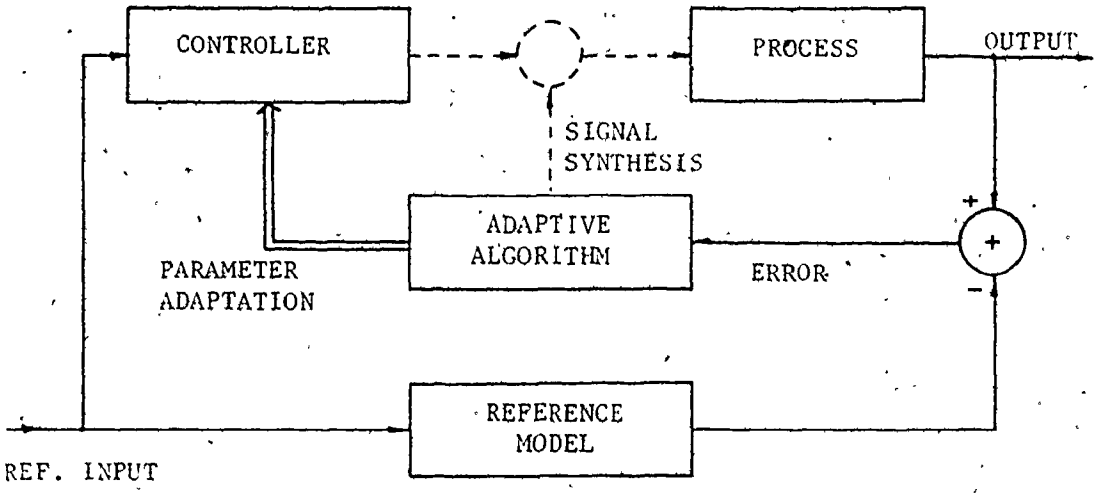


Fig. 3.9: Model reference adaptive system.

The two approaches are explained in greater detail in Chapter 5, on the design and testing of an adaptive controller for the industrial robot. Due to some limitations of the MRAS, this scheme could not be used for the adaptive controller for the robot.

5. Vernier robot method: In the schemes for robot control mentioned so far, errors in positioning the torch were minimized by suitably modifying the signals or parameters of the controller.

It is evident from the list of factors contributing towards the deviation of the weld point from its desired position, (e.g. Table 3.1), that inertia of the robot arm, elasticity of the arm and the backlash in the joints are major factors. Using precision servos for all axis motions will reduce errors, but this may not be economically viable.

Size and weight of the arm are governed by structural design to achieve the required span area and the job function. So, while error deviations to be corrected for, are small, the total span of the robot arm may be large. These requirements impose a conflicting limitation on the size of the manipulator arm.

One approach to solving this problem is by the addition of a precision robot with 1 or 2 degrees of freedom, immediately behind the welding torch. A sketch of such a robot is shown in figure 3.10.

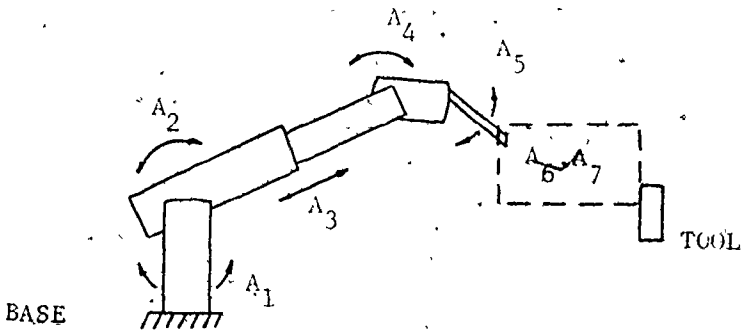


Fig. 3.10: Vernier robot.

Finer movement required for error correction is made with this vernier (or auxiliary) robot, which being of smaller dimensions may be equipped with a precision drive mechanism. Gross movement is controlled by the main robot. This scheme has been discussed in greater detail in [15].

From the above survey it appears that to achieve higher performance from the robot in a cost effective manner, an adaptive controller or a vernier type of robot may be required. A vernier robot increases the effective degrees of freedom (DOF) and should give

good performance from simple controllers, however, they may not be easy to implement. Adaptive controllers seem to be a viable alternative.

A self tuning type of an adaptive controller based on figure 3.8 has been designed for the modified UNIMATE-2000. Details of the design are given in Chapter 5. One of the requirements of the self tuning regulator is on-line modeling of the robot. Further, the structure of the robot has to be known in advance. A review of system identification is given in the next section.

3.5 System Identification

It is evident from the above, that a description of the process involved is necessary for the analysis and design of an adaptive controller. Such a description is often called the model of the process (or system) since it provides an insight into its characteristics. A model can take several forms, such as graphs, plots, expressions, etc., however, the type of model required most frequently is the mathematical model, which is a description in terms of mathematical relations.

Since an industrial robot usually has five to six joints, each driven by an independent motor, the robot has to be modelled as a multivariable system with several inputs and outputs. It has been shown [4-7,10], that the relations for the dynamics of a robot are non-linear, complex and consist of coupled terms. Closed form analytic solutions are not available, and numerical solutions obtained on the digital computer are time consuming.

The dynamic model of the robot, has to be simplified by assuming linearity, and neglecting the interaction between joints. Simulation studies in [12,13] demonstrate the feasibility of using simplified models for adaptive control.

A multivariable system can be represented in several ways [27]. Some of the commonly used schemes are:

1. Transfer function matrix,
2. Impulse response matrix,
3. Polynomial matrix, and
4. State space formulation.

Although these are different representations, they are equivalent in the sense that it is possible to transform one to the other. A comparison of these methods and algorithms for their estimations are given in [27].

For the industrial robot, because the interaction between joints is assumed to be small, each joint is treated independently as a single input single output system (SISO).

As mentioned earlier, a model for a system can be represented in a variety of ways. A brief account of methods and types of models is given below. Methods of modeling a robot arm can be classified into two catagories. The first method involves modeling the robot arm from the knowledge of its physical subsystems and their interaction. Information such as the time constants of motors, load-torque characteristics, etc., is usually available from the manufacturer. This information and the knowledge of physical laws may be used to formulate a model of the system.

The main drawback of this method is that often, when the system has complex dynamics, complete knowledge of interacting subsystems is not available. Also, in some case, resulting models are too complex for real time implementation on microcomputers.

The second method is called the 'black-box' approach, because the model is derived from the measurements of the input (excitation) and the output (response) of the system. The main advantage of this approach is that the model is obtained by practical testing, so no a-priori information about the constituent subsystems is necessary. This approach, popularly known as 'system identification' has been the topic of research for a long time and is extensively documented in books [26-30,40] and papers [31,32,34-39].

A model for a system may be expressed in various forms, broadly classified in two catagories - the non-parametric model, which is usually in the form of a impulse, step or frequency response plots; and the parametric model, usually in the form of the system function in the z-domain, difference equation, state space equations or in the s-domain transfer function. The type of model desired is dictated by the application for which it is used. Various types of models are represented in a diagramatic form in figure 3.11.

Among non-parametric models, one can isolate frequency and impulse response estimation as the basic techniques [39]. Both methods yield models in graphical form. Since, in most cases, it is not possible to use an 'impulse' test signal, the impulse response is obtained in an implicit way from the response to more feasible inputs. Several methods for 'implicit' impulse response modeling are available [27,32,39].

In the frequency domain, the most common representation of system dynamics, is in the form of Bode plots. While these representations are convenient for controller design and stability analysis, they are generally off-line methods and are not suitable for the design of adaptive controllers. Compact models, in the form of mathematical expressions are required. Non-parametric models can be converted to parametric ones by curve fitting and straight line approximations. References for some methods of conversion which assume that the systems order is known are given in [27,39]. An iterative method for obtaining a state space formulation from samples of the impulse response without a-priori knowledge of the system order is given in [35].

As shown in figure 3.11, parametric models can be obtained directly from the input-output data. The figure also shows the different types of parametric models in both the discrete time and the continuous type.

The process of obtaining a mathematical representation of the system dynamics (often the transfer function) from 'readings' of the input and output is termed as 'identification'. The input and output are normally analog signals, and often in the past, analog computers were used for identification. Due to recent advances in micro electronics, digital computers are currently quite inexpensive, and have the advantages of small weight and size. Hence, recent trend has been to use digital computers for identification. This requires that the system input and output signals be sampled, so that they can be processed by a digital computer.

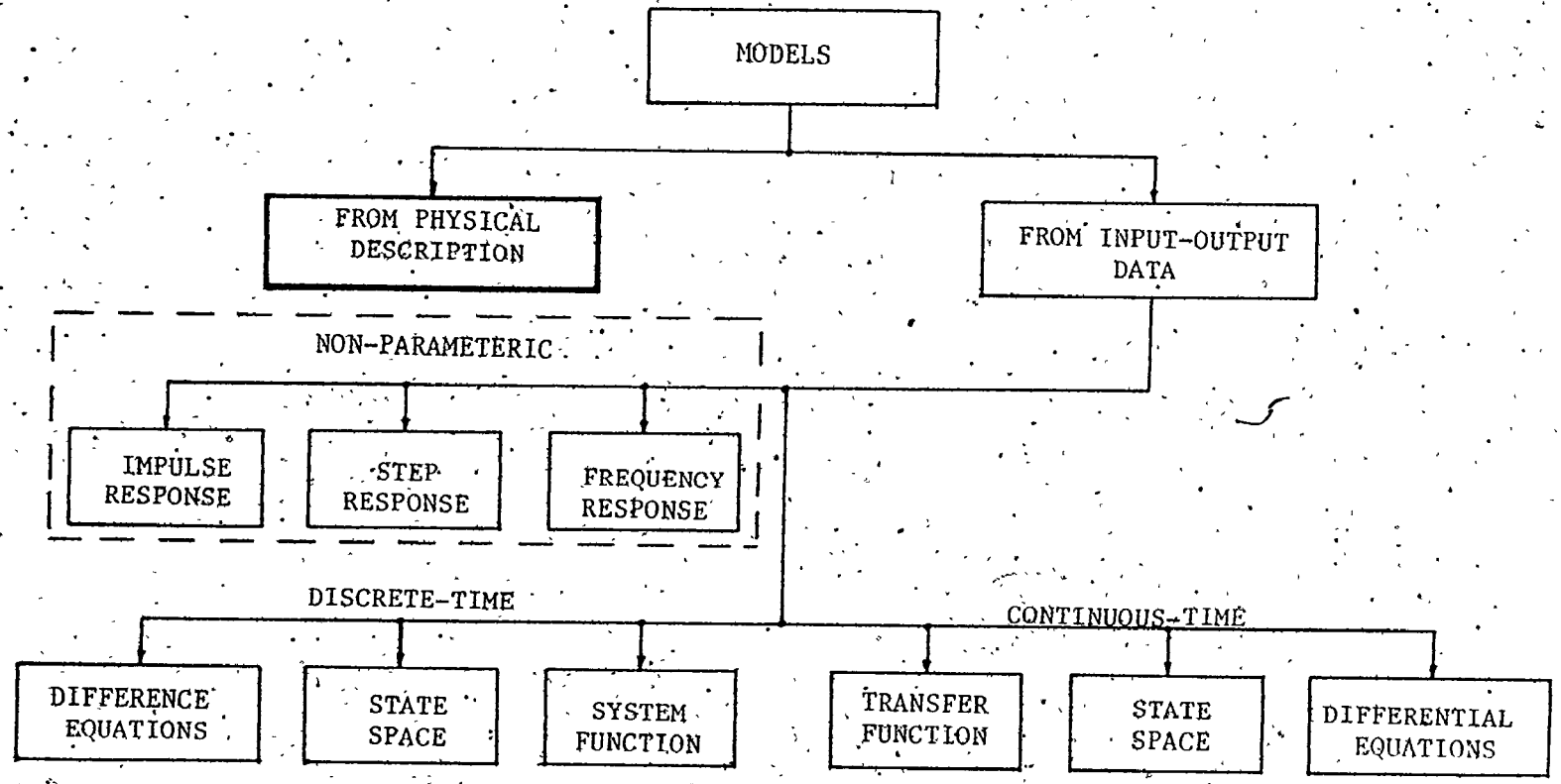


Fig. 3.11: Types of model representation.

Since the adaptive controller is implemented on a digital computer, a model in the form of a z-domain transfer function or a discrete time difference equation is most suitable. Sometimes, it is not possible to obtain the z-domain model directly from samples of I/O data. In such cases one has to derive them from non-parametric models as described above or from other types of parametric models. In the case of the industrial robot, it was necessary to obtain the discrete time model in two steps. First, a continuous time parametric model was evaluated from the samples of input-output data, which was then converted to an equivalent discrete-time model. Complete details of the measurement of data and modeling the robot is given in the next chapter. The process of parameter estimation, which is common to most models described above is reviewed below.

Methods of parameter estimation:

The purpose of identification is to develop a mathematical model which describes the behaviour of the unknown system in a sufficiently accurate manner.

A critical examination of the quality of the model is obtained by a comparison of the system output and the model output, when they both are excited by the same input signal sequence.

Consider a dynamic system with an input $\{u(t)\}$ and output $\{y(t)\}$. Let these signals be sampled such that $t = 1,2,3,\ldots$, and the sampled values be related by the difference equation.

$$y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = b_1 u(t-1) + \ldots + b_m u(t-m) + v(t) \qquad (3.1)$$

where v(t) is a disturbance.

Using the backward shift operator for convenience, we may rewrite 3.1 as

$$A(q^{-1})y(t) = B(q^{-1})u(t) + v(t) \tag{3.2}$$

where

$$q^{-1}y(t) = y(t-1) \tag{3.3}$$

and

$$A(q^{-1}) = 1 + a_1q^{-1} + a_2q^{-2} + \ldots + a_nq^{-n} \tag{3.4a}$$

$$B(q^{-1}) = b_1q^{-1} + b_2q^{-2} + \ldots + b_mq^{-m} \tag{3.4b}$$

The model describing the system by the equation 3.1 or 3.2 is expressed completely in terms of the parameter vector.

$$\theta^T = [a_1 \ \ldots \ a_n b_1 \ \ldots \ b_m] \tag{3.5}$$

Defining a vector of lagged input output data

$$\phi^T(t) = [-y(t-1) \ \ldots \ -y(t-n)u(t-1) \ \ldots \ u(t-m)] \tag{3.6}$$

Equation 3.1 may be rewritten as

$$y(t) = \theta^T\phi(t) + v(t) \tag{3.7}$$

The object of parameter estimation is to estimate the best value of $\theta = \hat{\theta}$ from measurements y(t) and $\phi(t)$ for $t = 1,2,\ldots,N$ where N is the measurement period.

There are several methods available in the literature [27-33] for evaluating θ.

Since we are interested in adaptive control, only recursive (on-line) approaches will be considered. Some methods are: (1) Recursive least squares, (2) Recursive instrumental variables, (3) Recursive extended least squares and (4) Recursive maximum likelihood methods. Also, since all these methods are well documented [30], details have not been included here. However, some comment on their merits and a unified method for estimation has been included.

Parameter estimation algorithms have been compared with respect to the performance of the estimates, reliability of convergence and the computational effort. Results of comparison of the recursive parameter estimation algorithms can be summarized as follows:

Recursive least squares (RLS): Applicable for small noise/signal ratios and uncorrelated noise, otherwise gives biased estimates. Reliable convergence. Relatively small computational effort.

Recursive extended least squares (RELS): Applicable for larger noise/signal ratios. Convergence slow in the starting phase and not always reliable. Noise parameters are estimated. Somewhat larger computational effort than RLS.

Recursive instrumental variables (RIV): Good performance of parameter estimates. Initial convergence can be accelerated by starting with the RLS. Computational effort greater than RLS.

Recursive maximum likelihood (RML): High performance of parameter estimates. Convergence in the beginning is slow but is more reliable than the RELS. Computational effort is greater than that for RLS, RELS and RIV.

In general, for small noise-to-signal ratios, the RLS method is preferred because of its simplicity and reliable convergence. The least squares estimate $\hat{\theta}(t)$ for $\theta(t)$ can be recursively calculated by the following expressions:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)\,[y(t) - \hat{\theta}^T(t-1)\phi(t)] \tag{3.8a}$$

$$L(t) = \frac{P(t-1)\phi(t)}{1/\alpha_t + \phi^T(t)P(t-1)\phi(t)} \tag{3.8b}$$

and

$$P(t) = P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{1/\alpha_t + \phi^T(t)P(t-1)\phi(t)} \tag{3.8c}$$

Initial conditions for starting the algorithm are $\hat{\theta}(0) = 0$ and $P(0) = C.I$, where C is some large constant. α_t is used to assign weights to data and is therefore called the 'weighting factor' [40].

A unified algorithm for all methods listed earlier is given in Table 3.3. Basic expressions are

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)e(t) \tag{3.9a}$$

$$e(t) = y(t) - \theta^T(t-1)\phi(t) \tag{3.9b}$$

and

$$L(t) = \mu(t)P(t-1)\psi(t) \tag{3.9c}$$

3.6 Concluding Remarks

Robotic welding is done by programming the robot to move along the joint while the welding equipment is turned ON. In practice, it is seen that due to several errors and limitations, perfect welding is rarely achieved. In this chapter, we first identify the causes of error. It is seen that they can basically, be listed in two catagories - those causing positioning errors and the others as errors in setting weld parameters. The problem of containing positioning errors is addressed in greater detail. First, a review of various position sensing devices is included, followed by a survey of possible control schemes. Details of the implementation of a feedback method for seam tracking are included and a new method using an auxiliary robot is suggested. A self tuning type of an adaptive controller was seen to be most suitable for the controller design. Since adaptive controllers are based on modeling the system, a review of system identification methods is included.

Table 3.3: Unified Method for Parameter Estimation

Method	$\hat{\theta}$	$\phi^T(t)$	$\mu(t)$	$P(t)$	$\psi(t)$
RLS	$\begin{bmatrix} a_1 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_m \end{bmatrix}$	$[-y(t-1) \quad -y(t-n)\, u(t-1) \ldots u(t-m)]$	$\dfrac{1}{1/a_t + \phi^T(t) P(t-1) \phi(t)}$	$[I - L(t)\phi^T(t)] P(t-1)$	$\phi(t)$
RIV		$[-y(t-1) \quad -y(t-n)\, u(t-1) \ldots u(t-m)]$	$\dfrac{1}{1/a_t + \phi^T(t) P(t-1) \psi(t)}$	$[I - L(t)\psi^T(t)] P(t-1)$	$[-h(t-1) \quad h(t-n) \quad u(t-1) \quad u(t-m)]$ $h(\cdot)$ – instrumental variables
RELS	$\begin{bmatrix} a_1 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_m \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$	$[-y(t-1) \ldots -y(t-n) \quad u(t-1) \quad u(t-m) \quad e(t-1) \ldots e(t-n)]$	$\dfrac{1}{1/a_t + \phi^T(t) P(t-1) \phi(t)}$	$[I - L(t)\phi^T(t)] P(t-1)$	$\phi(t)$
RML		$[-y(t-1) \quad -y(t-n) \quad u(t-1) \ldots u(t-m) \quad e(t-1) \ldots e(t-n)]$	$\dfrac{1}{1/a_t + \psi^T(t) P(t-1) \psi(t)}$	$[I - L(t)\psi^T(t)] P(t-1)$	$[-y'(t-1) \ldots -y'(t-n) \quad u'(t-1) \quad u'(t-m) \quad e'(t-1) \quad e'(t-n)]$

CHAPTER 4

MODELING THE ROBOT ARM

4.1 Introduction

It was shown in Chapter 3 that an adaptive controller was necessary to get satisfactory performance from the industrial robot. Several forms of adaptive controllers were discussed and it was observed that all of them are based on a model of the system.

In this chapter, we describe details of modeling the industrial robot. The chapter is organized in four main sections as described below.

Since the model was derived from the input-output behaviour of the robot, the first section describes the experimental set-up for measuring the input and output signals for the robot arm. Samples of the input and output signals were then used for obtaining the model. Because the procedure for modeling and the design of an adaptive controller is the same for all axes, details of only one axis are included. In a practical implementation of the controller, this procedure has to be repeated for the other axes.

An ideal input from a system identification point of view is the pseudo random binary sequence (PRBS). However, for some systems, it may not be convenient to apply the PRBS. So with a view of keeping the algorithm as general as possible, the step signal was used to excite the system.

The second section is on specifying the structure of the model. Since the transfer function type of model was selected, this step refers to specifying the order of system polynomials.

Section 4.4 describes the theory for evaluating the parameters of the model, from samples of input-output data. The controller described in Chapter 5 is based on a discrete time model, but because it was not possible to get a discrete time model directly from the input-output samples, a two step approach was used. The two steps involved, are evaluating a continuous time model and then converting it to a discrete time model.

This section describes the theory for evaluating the continuous time model. The second part, that of converting it to an equivalent discrete time model is explained in the next chapter. It is important to note that this process of modeling the robot is off-line and has to be done only once. On-line parameter tracking and modeling required for adaptive control is performed directly in the discrete domain.

Section 4.5 deals with the aspect of model verification. This is to check how well a model fits the input-output data and to select between several models for the system. Finally, in section 4.6, we present results of modeling the robot from samples of its step response. Theory described in earlier sections is used and several models for the robot are obtained. The theory is first verified by simulation studies. In conclusion, a third order model is chosen for the adaptive controller. These steps are diagramatically represented in figure 4.1.

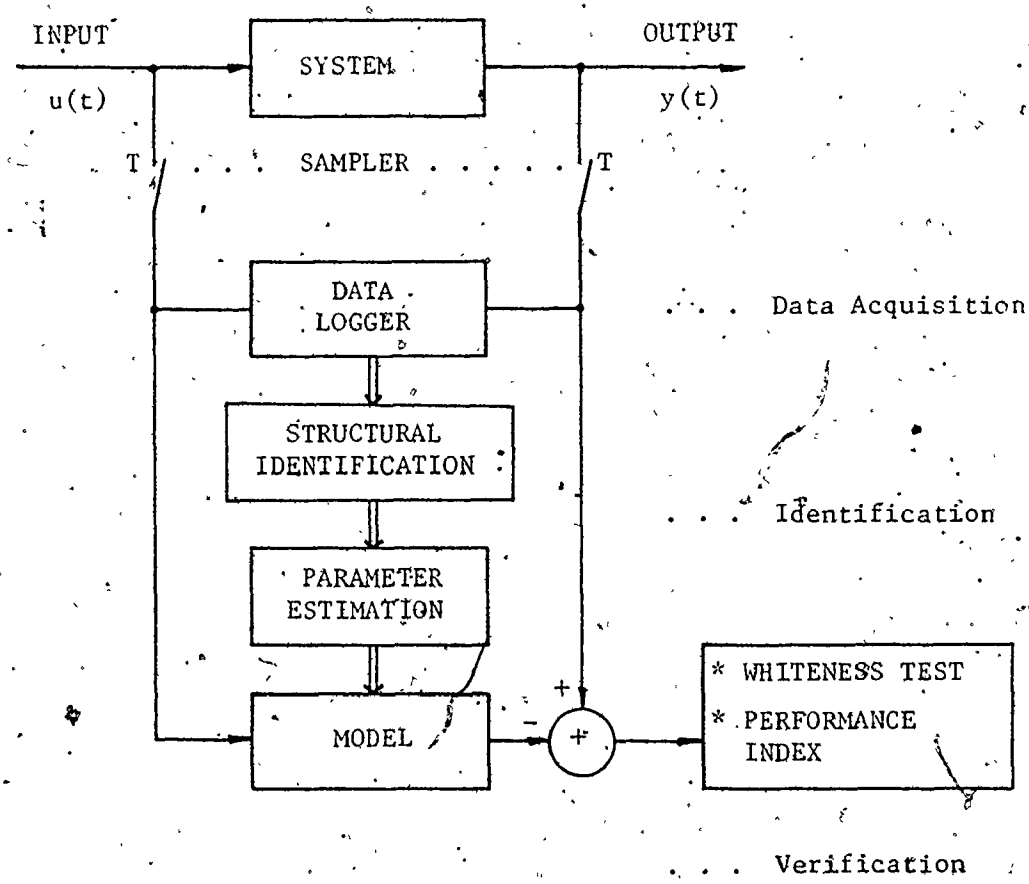


Fig. 4.1: System identification.

4.2 Data Acquisition

(a) Type of input:

This step involves the generation of the input or excitation, and measurement of the system response. The two main issues involved are the choice of the input signal and the choice of the sampling interval. As mentioned earlier, the PRBS is an ideal input from the system identification point of view. In practice, a square wave input has also been found quite suitable. However, for many biomedical and mechanical systems, it is not possible (or convenient) to apply such

'persistantly exciting' inputs, and the system has to be excited by inputs such as the step, ramp, pulse, etc.

Since the step input is the easiest to generate, it was used for modeling the robot arm. One of the serious consequences of restricting the input to a non-persistantly exciting one such as the step input, is that the commonly used methods to obtain the discrete time model H(z) can not be used. As described in later sections, H(z) is now obtained indirectly, by first evaluating the continuous time model H(s).

(b) Choice of sampling interval:

The other important decision is the choice of the sampling interval T. The value of T is governed by [34], (a) the sampling time required for the final application (e.g. control), (b) accuracy of the model, and (c) numerical problems.

In general, if too small a value is chosen, in addition to more computations, it results in ill-conditioned matrices and equations, which cause numerical problems. Large values of T clearly result in a loss of information (accuracy). A rule-of-thumb suggested in [45] is that T should be chosen such that $\lambda_m T \leq 0.5$; where λ_m is the largest eigenvalue of the continuous time model. In practice however, a-priori knowledge of the location of eigenvalues is not available and one has to estimate T based on the information of system time constants. It is suggested in [34] that the sampling time may be chosen such that $T_{95}/T = 5 \ldots 15$; where T_{95} is the 95% settling time of the transient function.

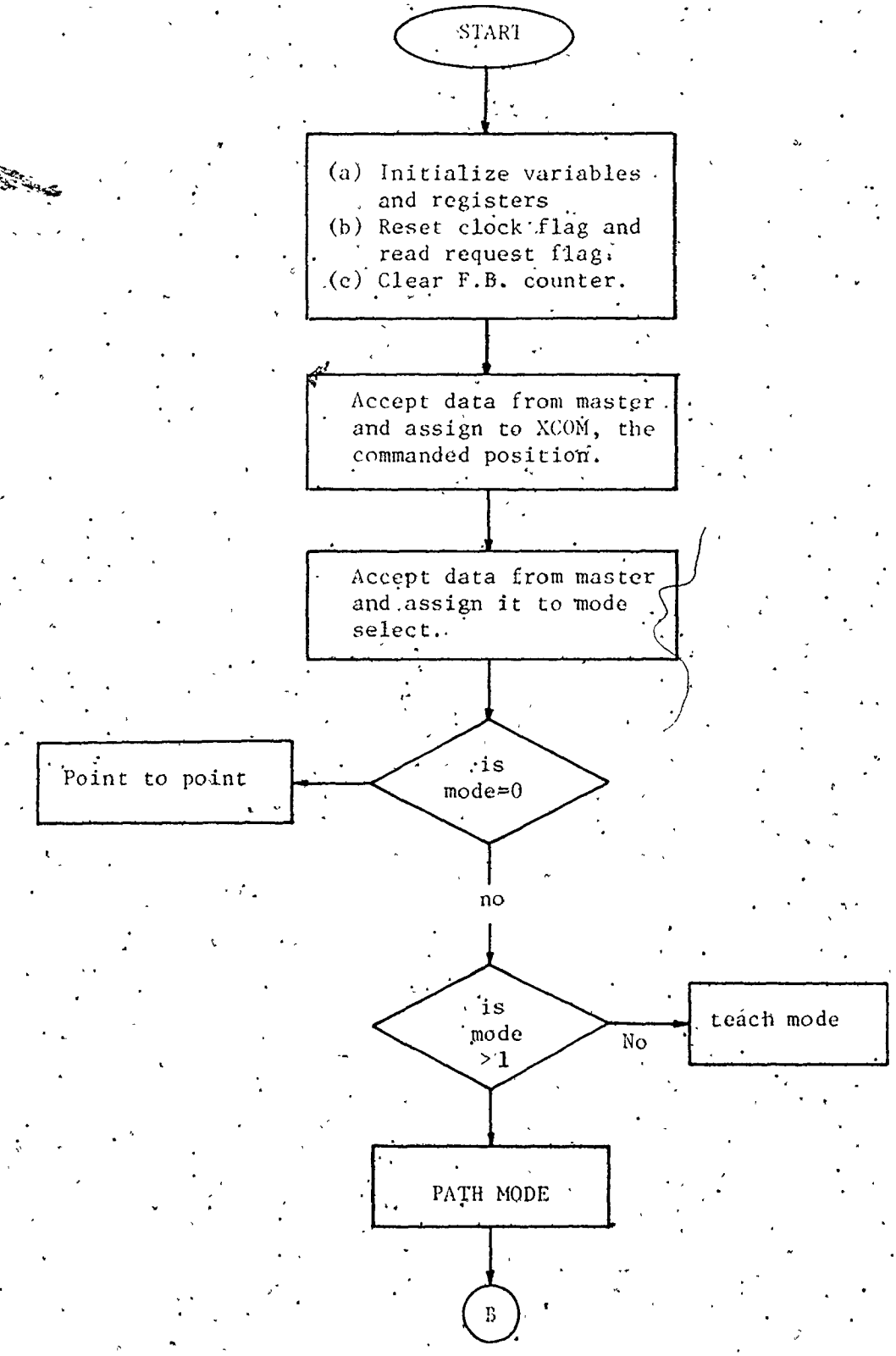
A sampling time of 1 ms was selected for sampling the step input and response of the robot.

(c) Experimental details:

One of the most important tasks in data acquisition for modeling the industrial robot from samples of its step response, is locating a convenient input point. The present control and drive system of the robot were described in Chapter 2. A convenient point for the application of the input is the terminal marked 'A' in figure 2.4. This terminal can be isolated easily by removing all 'slave' cards. The problem with this approach is that with the slave card removed, the 'position loop' is left open. Since the position loop is closed through the slave card, it was necessary to excite the system with the slave controllers 'in-circuit'. Further, because the loop is closed by the firmware of the microcomputer, it was necessary to analyze the firmware so that an appropriate 'software' entry point could be determined. Also since the slave controller communicates with the master controller, software on the master controller has to be modified to include signal generating software.

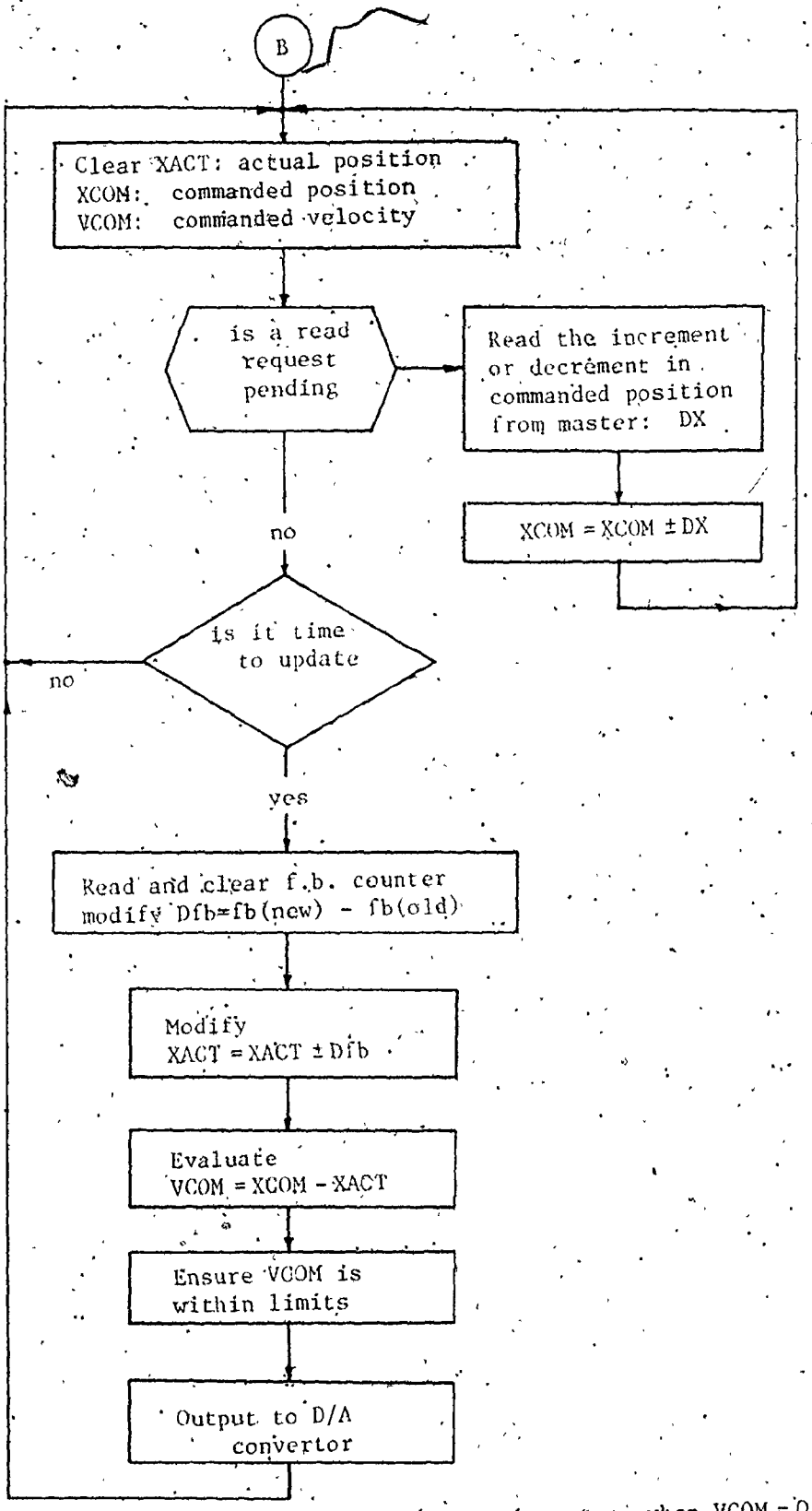
Slave controller firmware

As mentioned earlier, the slave controller is built around the 8748 micro-computer. The firmware was written in its assembly language viz. ASM-48. Since an assembly language listing may not be easy to comprehend without familiarity with the circuit details, a block level flow chart has been presented in figure 4.2. The path



continued on next page

Fig. 4.2: Slave controller firmware.



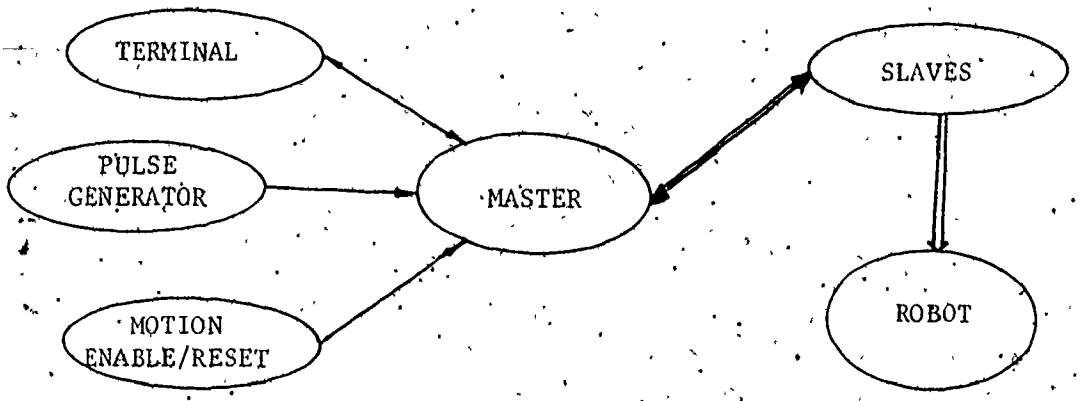
* - motion stops when VCOM = 0

Fig ued

mode was selected since it was the only mode which closed the position loop and did not force a trapezoidal velocity profile. It is evident from the flow chart that once the path mode has been selected, it is possible to control the direction and magnitude of motion for the selected axis of the robot under the control of the master computer. This mode has been used to excite the robot to procure its step response.

Scheme for applying the input

A step input signal can be generated by turning a switch ON or OFF manually. However, it was decided to use a pulse generator so that by selecting an appropriate repetition rate (>> settling time), the input can be made to recur periodically. In addition, it would also facilitate the application of 'pulse train' type of excitation. The amplitude of the motion is entered through the main keyboard and a 'motion enable' switch is used to enable or disable motion (fig. 4.3). This switch performs another function - every time the motion is disabled, it re-runs the program so that the operator may change the magnitude of motion and also the axis to be exercised.



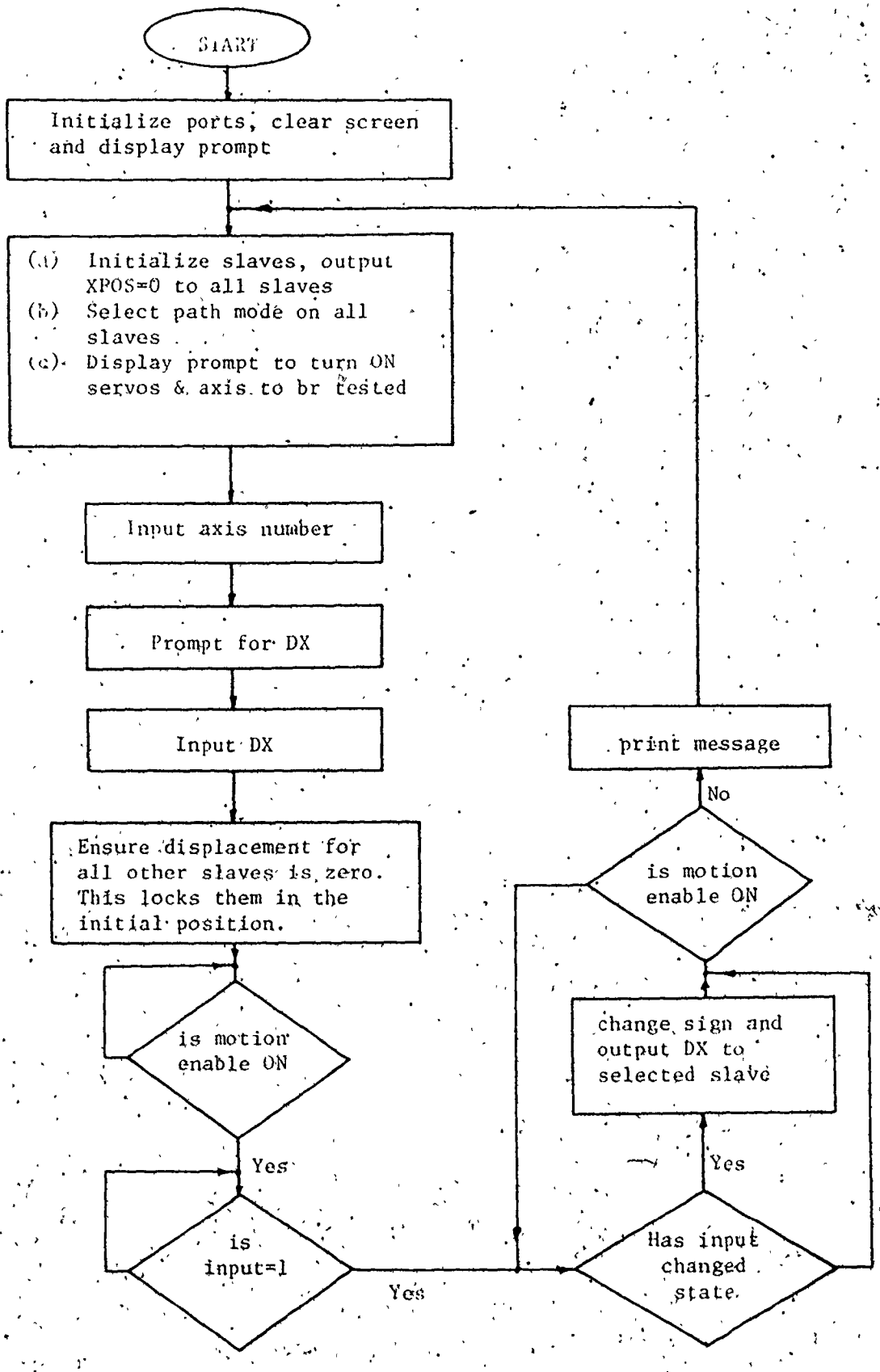
Software for the master controller

The software used to generate commands for the slave controller is shown in the form of a simplified flow chart in figure 4.4. The software is interactive in the sense that it prompts the operator with messages such as 'Turn on-servos', 'motion disabled', 'Input magnitude of motion', etc. The pulse generator and the motion enable switch are connected on a parallel port. If the motion is enabled, the master controller sends commands to the selected slave everytime the signal from pulse generator changes state from '0' to '1' or '1' to '0'.

Data measurement

The input signal and a signal corresponding to the end position of the manipulator, as picked up by a proximity probe, was digitized and stored on a floppy disk. This task was simplified because of the availability of a digital scope which had a built-in floppy system. A block diagram of the system used for data collection and storage is shown in figure 4.5. The scope had a capacity of storing approximately 8000 samples. The measurement time for one record with a sampling rate of 1 ms is 8 seconds which is long enough for the manipulator end to attain a steady state value.

Typical measurements recorded are plotted in figures 4.6 and 4.7. The set up shown in figure 4.5 was also used to record the signals corresponding to the commanded velocity.


START
Initialize ports, clear screen and display prompt
(a) Initialize slaves, output XPOS=0 to all slaves
(b) Select path mode on all slaves
(c) Display prompt to turn ON servos & axis to br tested
Input axis number
Prompt for DX
Input DX
Ensure displacement for all other slaves is zero. This locks them in the initial position.
is motion enable ON
Yes
is input=1
Yes
Has input changed state
Yes
change sign and output DX to selected slave
is motion enable ON
No
print message

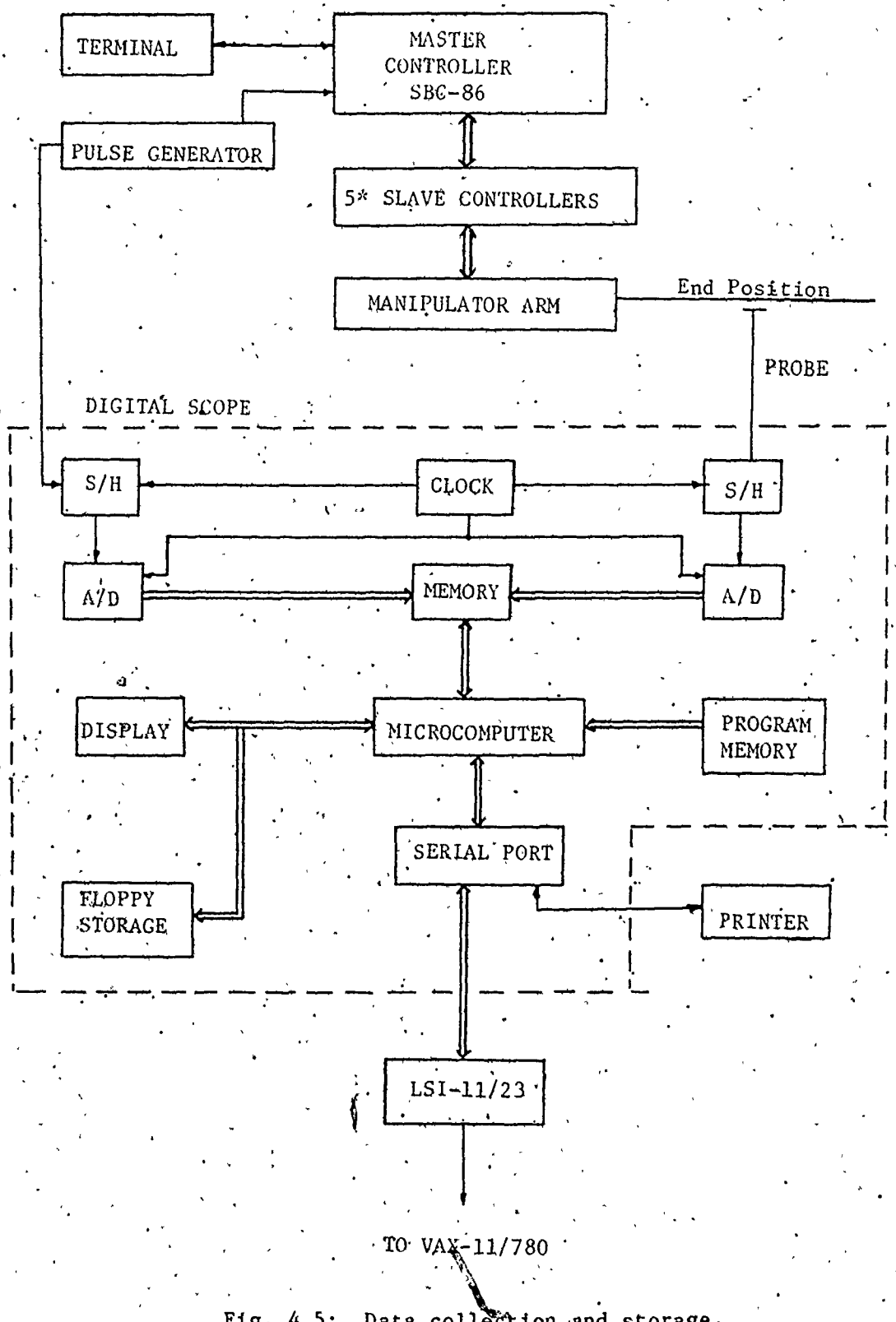


Fig. 4.5: Data collection and storage.

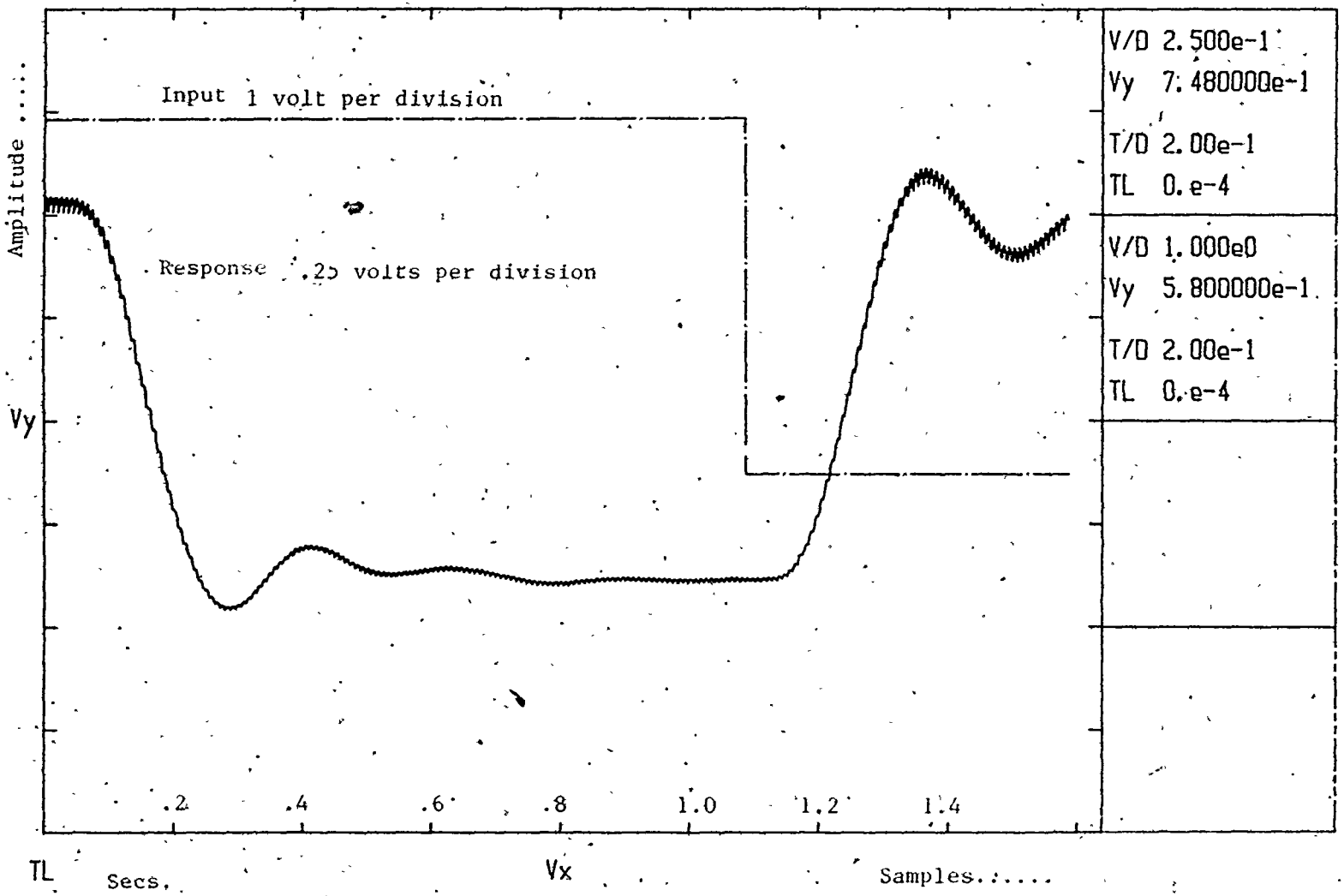


Fig. 4.6: Step response of robot arm.

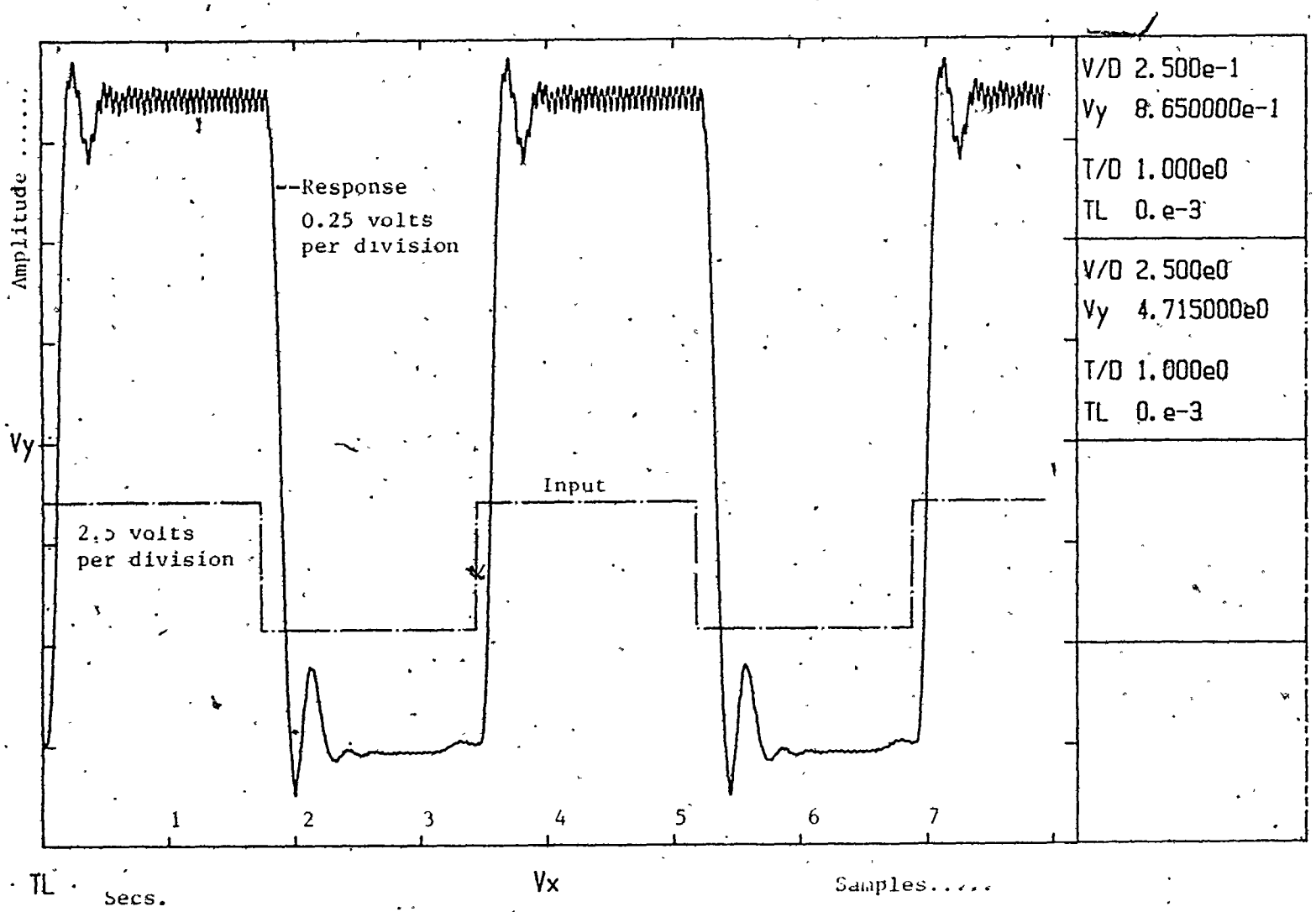


Fig. 4.7: Plot of input-output data.

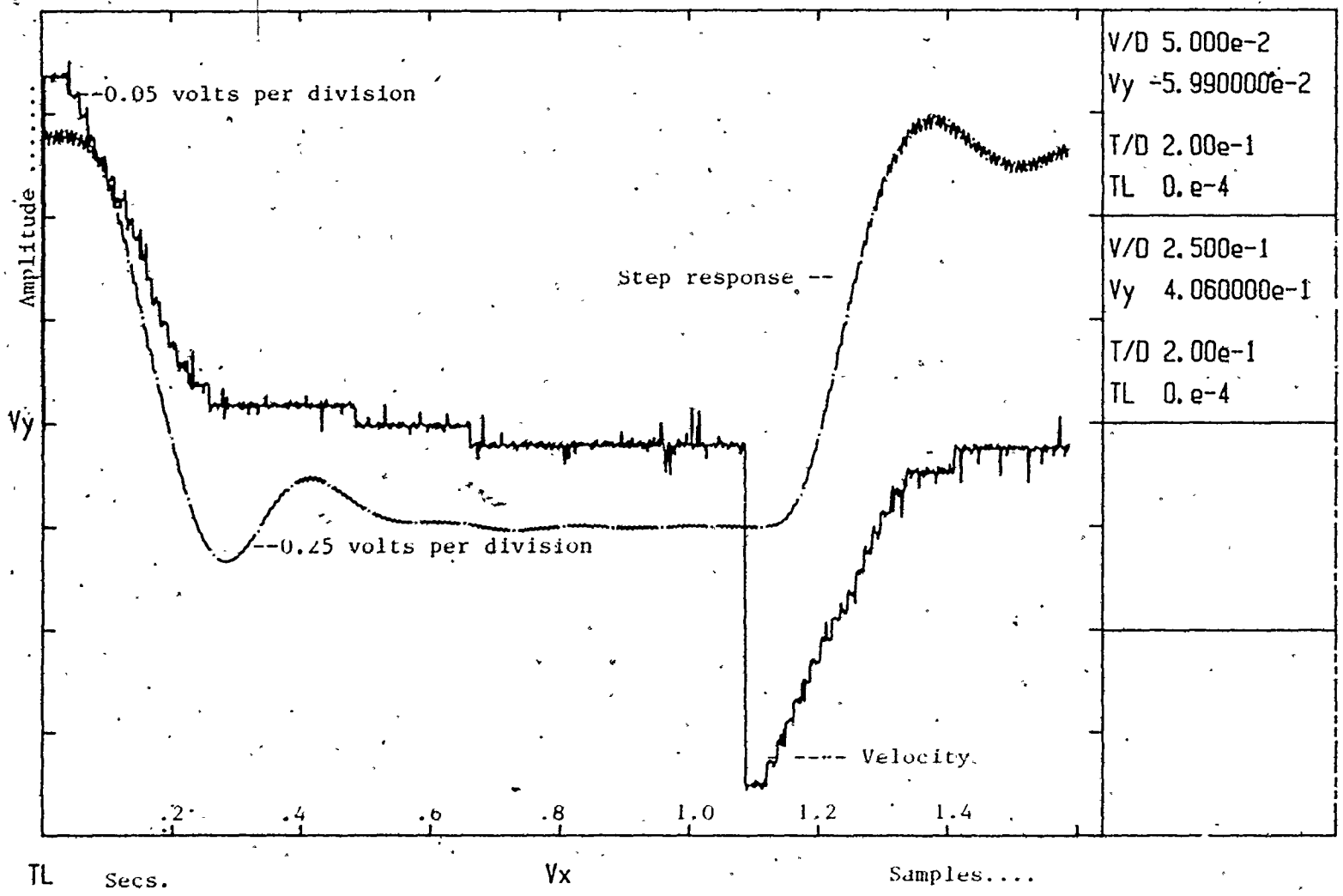


Fig. 4.8: Plot of commanded velocity and output position.

It may be noted that the amplitude of the step as shown on the plots is not representative of the actual commanded displacement. The displacement was input to the master computer directly through the operator terminal.

To facilitate the analysis of these signals, the scope was interfaced to an LSI-11 microcomputer and recorded waveforms were transferred to the VAX-11/785 mainframe computer in an RT-11 format.

4.3 Structural Identification

This step involves the estimation of the structure of the model to be used. As mentioned in Chapter 3, there are several ways of representing a system. In this case, the transfer function approach was selected because of its suitability in the design and analysis of the controller. For the transfer function model, structural parameters required to characterize the model are the degrees of the numerator and denominator polynomials.

In general,

$$H(s) = \frac{B(s)}{A(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_0} \qquad (4.1)$$

The problem is to specify m and n.

In some cases, it is possible to estimate the order of the system from transient response plots. However, in most practical cases, particularly for higher order systems and when the noise level is high, selecting the order is more difficult.

Frequency domain plots such as Bode plots, provide some insight to the order and natural frequencies of the system. But in practice, frequency response curves are not always available and it may not be convenient to obtain them experimentally.

There are some methods that indicate the model order from matrices formed from samples of input-output data or from samples of the impulse response. Since most of them rely on evaluating determinants of these matrices, such methods are not always reliable because of the ill-conditioning caused by the presence of noise. Other methods of obtaining the model order are given in several references, e.g. in [27,32,33,46]. In practice, for unknown systems, a few different structures have to be tried and the model corresponding to the 'best fit' selected. Many criteria for determining the 'best fit' have been suggested; an index defined as 'ratio test' (RT) was used for selecting the most appropriate model for the robot. The index RT is based on comparing the step response of the robot to that of the model, and is defined in section 4.5.

4.4 Parameter Estimation

Once the structure of the model is decided and the input-output data acquired, the next step involves estimating the parameters of the model. The design of the controller described in Chapter 5, is based on the discrete time model of the robot.

Several methods of estimating the discrete time model from samples of input-output data were described in Chapter 3. It was shown that graphical methods were not suitable when the system order

is high and also when data is corrupted by noise. Some other common approaches to 'system identification' were not applicable in this case, because of its requirement that the input (test) signal has to be 'persistantly' exciting.

In the approach used, the discrete time model is evaluated in two steps. The first step is to obtain the continuous time model in the form as shown in Eq. 4.1, by using the 'direct' method of identification. This model is then converted to an equivalent discrete time model by assuming a zero-order hold circuit at the input and a sampler at the output.

In this section, we give details of the direct method of evaluating parameters of the continuous time model. The scheme is based on obtaining piecewise constant solutions of a linear differential equation describing the system behaviour over a short interval. It has the advantage that it is not restricted to low order systems, is fairly insensitive to noise in the data, and can be used with step and ramp inputs also.

The direct method:

The direct method for evaluating coefficients of the transfer function from input-output data has been discussed by several authors [36,38,41-44]. A systematic classification of various methods for estimating parameters of continuous time systems is given in [47].

In the time domain, the system H(s) in Eq. 4.1, relating input and output may be written as

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1} y}{dt^{n-1}} + \ldots + a_1\frac{dy}{dt} + a_0 y$$

$$= b_m\frac{d^m u}{dt^m} + b_{m-1}\frac{d^{m-1} u}{dt^{m-1}} + \ldots + b_1\frac{du}{dt} + b_0 u \qquad (4.2)$$

The problem is to obtain an estimate of the model parameters $a_0, a_1, \ldots, a_{n-1}$, $b_0, b_1, \ldots, b_m$, from samples of u(t) and y(t), that is from sequences {u(kT)} and {y(kT)} for k = 0,1,...,N-1. Where T is the sampling period and N the measurement interval.

Integrating Eq. (4.2), n-times with respect to 't', in the duration $kT \leq t \leq (k+1)T$, and rearranging, we get

$$[I_{n,k+1}(y) - I_{n,k}(y)]\, a_0 + [I_{n-1,k+1}(y) - I_{n-1,k}(y)]\, a_1 + \ldots +$$

$$[I_{1,k+1}(y) - I_{1,k}(y)]\, a_{n-1} - [I_{n,k+1}(u) - I_{n,k}(u)]\, b_0 \ldots -$$

$$[I_{n-m,k+1}(u) - I_{n-m,k}(u)]\, b_m = y_k - y_{k+1} \qquad (4.3)$$

where $I_{n,k+1}(u)$ is the n^{th} integral of u(t) at t = (k+1)T

$I_{n,k+1}(y)$ is the n^{th} integral of y(t) at t = (k+1)T

and $y_k = y(kT)$

Eq. (4.3) may be represented in vector form as

$$\underline{A}_{k+1}^T \underline{\theta} = y_k - y_{k+1} \qquad (4.4)$$

where $\underline{\theta}^T = [a_0, a_1 \cdots a_{n-1}\ b_0\ b_1 \cdots b_m]$

The vector $\underline{A}_{k+1}$ is made of integrals of u(t) and y(t).

If $\alpha_i = I_{n-i,k+1}(y) - I_{n-i,k}(y)$

and $\beta_i = I_{n-i,k+1}(u) - I_{n-i,k}(u)$

then $\underline{A}_{k+1}^T = [\alpha_0\ \alpha_1 \cdots \alpha_{n-1}\ -\beta_0 - \beta_1 \cdots -\beta_m]$

Setting k = 0,1,...,N-1 in equation (4.4), we get N equations for n+m+1 unknowns. This set of equations may be represented by

$$\underline{\psi}^T \underline{\theta} = \underline{B} \tag{4.5}$$

where the vector $\underline{B}$ is given by

$$\underline{B} = \begin{bmatrix} y_0 - y_1 \\ y_1 - y_2 \\ \vdots \\ y_{N-1} - y_N \end{bmatrix} \quad \text{and} \quad \psi = \begin{bmatrix} A_0^T \\ A_1^T \\ \vdots \\ A_{N-1}^T \end{bmatrix}$$

Equation 4.5 may be solved for an estimate $\hat{\underline{\theta}}$ by any of the well known parameter estimation techniques described in Chapter 3.

It may be observed that the vector $\underline{\psi}$ is made up of elements which are a function of successive integrals of u(t) and y(t). It is evident that even when the input u(t) = c; - a constant step function, equations used for identification do not contain linear combinations

of parameters. For the discrete case, as mentioned earlier and shown in [26], the vector corresponding to $\underline{\psi}$ would contain lagged samples of the input and output. This causes some terms in the identification equations to appear as linear combinations thereby making identification impossible.

Complete identification of the system using Eq. 4.4 may be represented as shown in figure 4.9.

It may be noted that this scheme of using multiple integrators is a special case of the state variable filter (SVF) approach described in [47].

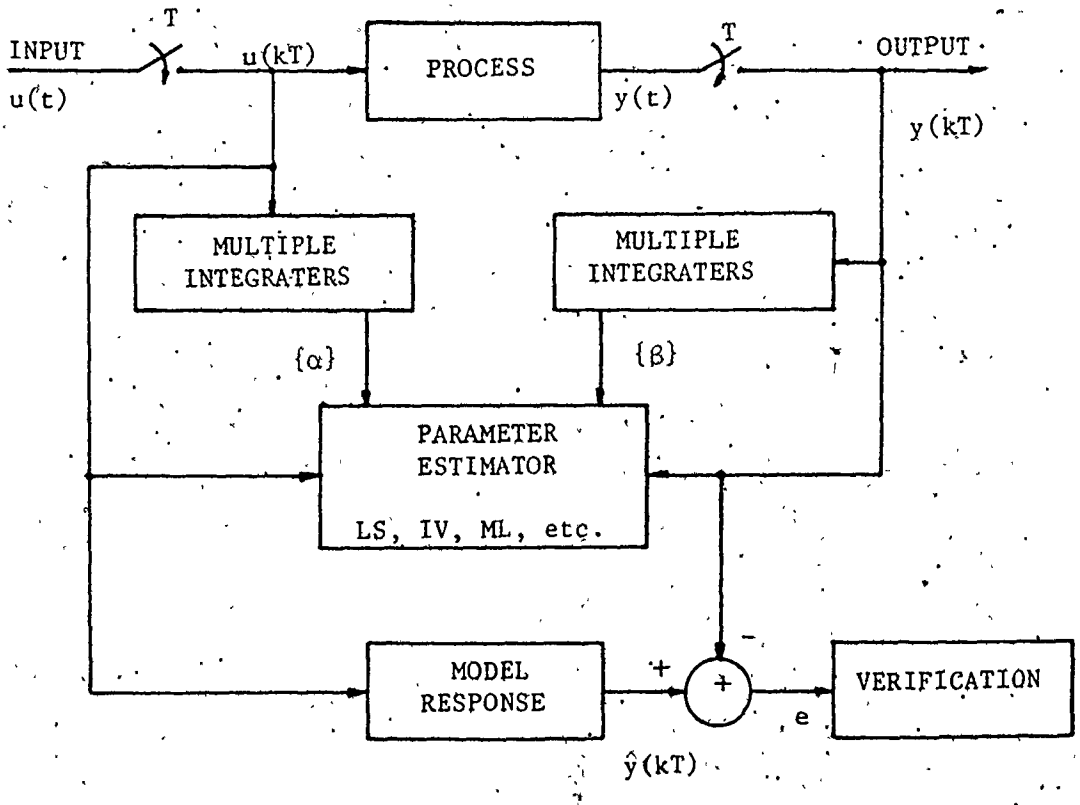


Fig. 4,9: Direct method of modeling a continuous-time system.

Integrals given by {α} and {β} are evaluated from samples of the input-output signals by first approximating piecewise constant functions over the sampling intervals. Several methods have been used in the past [42,43] to approximate a signal from its samples. Some of them are (i) Walsh function method, (ii), Fourier-Walsh method, (iii) Block pulse function approximation, (iv) Trapezoidal pulse functions and, (v) Cubic spline approximations.

The cubic spline approximation of sampled signals is more accurate, but requires a large number of computations. The trapezoidal pulse function approximation (TPF), is a good compromise between the desired accuracy and the computational complexity. In the TPF approach, the continuous time function y(t) is approximated from its samples as shown in figure 4.10.

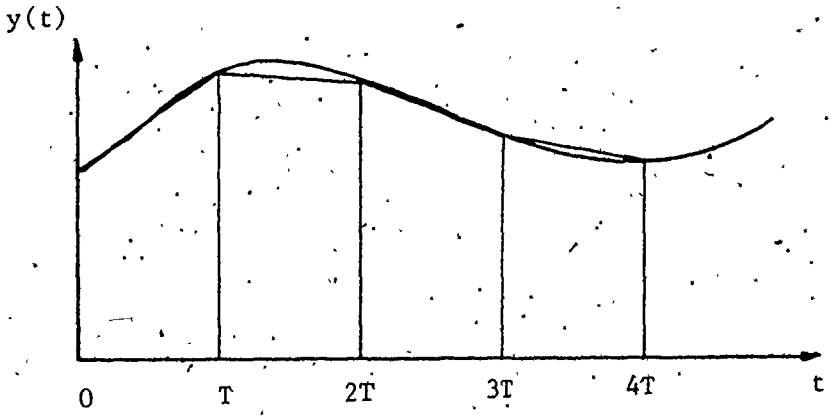


Fig. 4.10: Trapezoidal pulse function approximation.

Mathematically, TPF approximation may be represented by,

$$y(t) = \frac{1}{T}\left[\{(k+1)T-t\}\, y(kT) + (t-kT)y((k+1)T)\right] \qquad (4.6)$$

Using the above approximation, it can been shown [43], that the general expression for the n^{th} integral of y(t) with respect to 't' at t = (k+1)T is given by

$$I_{n,k+1}(y) = I_{n,k}(y) + T\, I_{n-1,k}(y) + \frac{T^2}{2!} I_{n-2,k}(y) + \ldots$$

$$+ \ldots \frac{T^{n-1}}{(n-1)!} I_{1,k}(y) + \frac{T^n}{(n+1)!} [ny(kT) + y(k+1)T] \qquad (4.7)$$

with $I_{n,0}(y) = 0$.

Eq. (4.7) may be rewritten as

$$I_{n,k+1}(y) = \sum_{m=0}^{n-1} \frac{T^m}{m!} I_{n-m,k}(y) + \frac{T^n}{(n+1)!} [ny(kT) + y(k+1)T] \qquad (4.8)$$

From Eq. 4.8 and Eq. 4.4, we get a set of N equations for k = 0,1,..,N-1. This set of equations, of the form $\underline{A\theta} = \underline{E}$ may be solved for the vector $\underline{\hat{\theta}}$ by any of the methods described in Chapter 3 and [27,40]. The least squares estimate requires the least amount of computations and storage. In all simulations and also for modeling the robot, the least squares method was used to estimate $\underline{\hat{\theta}}$.

It may be noted that the smoothing effect due to the integration performed, results in fairly good estimates, even in the presence of noise. For very noisy data one may use the instrumental variable (IV) or the maximum likelihood (ML) methods.

4.5 Model Verification

After a model has been obtained, it is necessary to test it for suitability. Diagnostic tests are also required for selecting the best among several models for the same process. These tests are based on comparing the response of the process with that of the model. A model is considered suitable if the two responses are 'sufficiently close'. Several methods for model verification are available in the literature, both in books [26-28,49] and papers [41,48]. Commonly used methods of model verification (figure 4.11) are: (i) testing residuals for whiteness, (ii) using an index to define closeness and, (iii) by a visual comparison of the plots of the actual output and the model output.

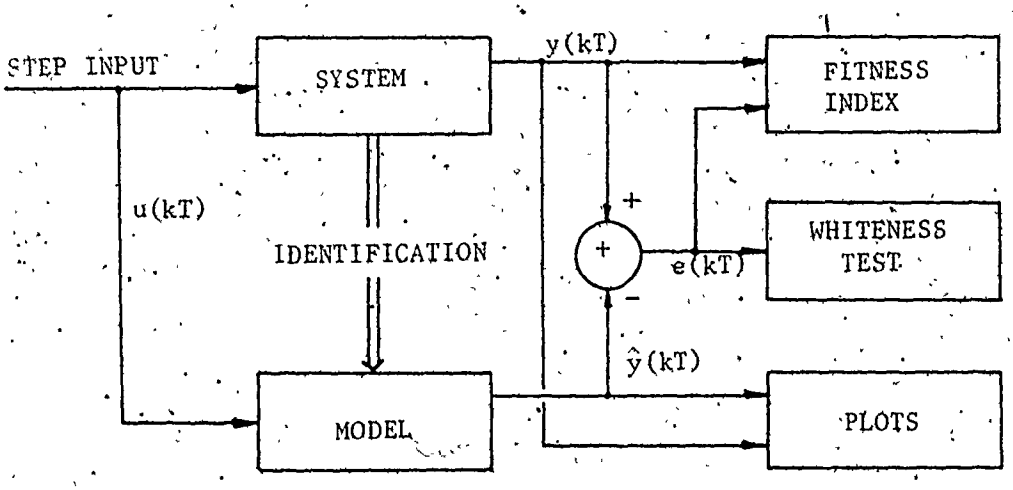


Fig. 4.11: Model verification.

The residual error e(kT) is defined as $y(kT) - \hat{y}(kT)$ where $\hat{y}(kT)$ is the output of the model and y(kT) the actual system output, corresponding to the input u(kT).

For results presented here, the fitness index RT[50] as defined in equation 4.9 was used

$$RT = 1 - \frac{\sum_{k=0}^{N-1} e^2(kT)}{\sum_{k=0}^{N-1} y^2(kT)} \qquad (4.9)$$

An idea of how well the model fits the data can be obtained by testing the value of RT. A good fit is indicated if RT is close to +1.

The model output $\hat{y}(kT)$ is required for all verification tests. It may be evaluated by spliting the model transfer function into its partial fractions, and then determining the corresponding time domain function. A computationally more efficient method can be derived from Eq. 4.3.

We have from Eq. 4.3

$$\hat{y}_{k+1} = \hat{y}_k - \sum_{j=0}^{n-1} [I_{n-j,k+1}^{(\hat{y})} - I_{n-j,k}^{(\hat{y})}]a_j + \sum_{r=0}^{m} [I_{n-r,k+1}^{(u)} - I_{n-r,k}^{(u)}]b_r \qquad (4.10)$$

We have from Eq. 4.4, by changing the index 'n' to x

$$I_{x,k+1}^{(\hat{y})} - I_{x,k}^{(\hat{y})} = \sum_{t=1}^{x-1} \frac{T^t}{t!} I_{x-t,k}^{(\hat{y})} + \frac{T^x}{(x+1)!} [x\hat{y}_k + \hat{y}_{k+1}] \qquad (4.11)$$

Substituting (4.11) in (4.10)

$$\hat{y}_{k+1} = \hat{y}_k = \sum_{j=0}^{n-1} \sum_{t=1}^{n-j-1} \frac{T^t}{t!} I_{n-j-t,k}(\hat{y}) + \frac{T^{n-j}}{(n-j+1)!} [(n-j)\hat{y}_k + \hat{y}_{k+1}] \; a_j$$

$$+ \sum_{r=0}^{m} \sum_{t=1}^{n-r-1} \frac{T^t}{t!} I_{n-r-t,k}(u) + \frac{T^{n-r}}{(n-r+1)!} [(n-r)U_k + U_{k+1}] \; b_r \qquad (4.12)$$

Transferring terms containing $\hat{y}_{k+1}$ to the LHS of the expression we have, the LHS

$$\hat{y}_{k+1} \left[1 + \sum_{j=0}^{n-1} \frac{T^{n-j}}{(n-j+1)!} a_j\right] \qquad (4.13a)$$

The RHS is

$$\hat{y}_k + \sum_{j=0}^{n-1} \sum_{t=1}^{n-j-1} \frac{T^t}{t!} I_{n-j-t,k}(\hat{y}) \; a_j - \sum_{j=0}^{n-1} \frac{T^{n-j}}{(n-j+1)!} (n-j).\hat{y}_k.a_j$$

$$+ \sum_{r=0}^{m} \sum_{t=1}^{n-r-1} \frac{T^t}{t!} I_{n-r-t,k}(u) \; b_r + \sum_{r=0}^{m} \frac{T^{n-r}}{(n-r+1)!} ((n-r)U_k + U_{k+1}) \; b_r$$

(4.13b)

Starting with $\hat{y}(0) = 0$, subsequent values of $\hat{y}(kT)$ may be evaluated from equations 4.13 and 4.8. The complete procedure for identification is represented in the flow chart in figure 4.12.

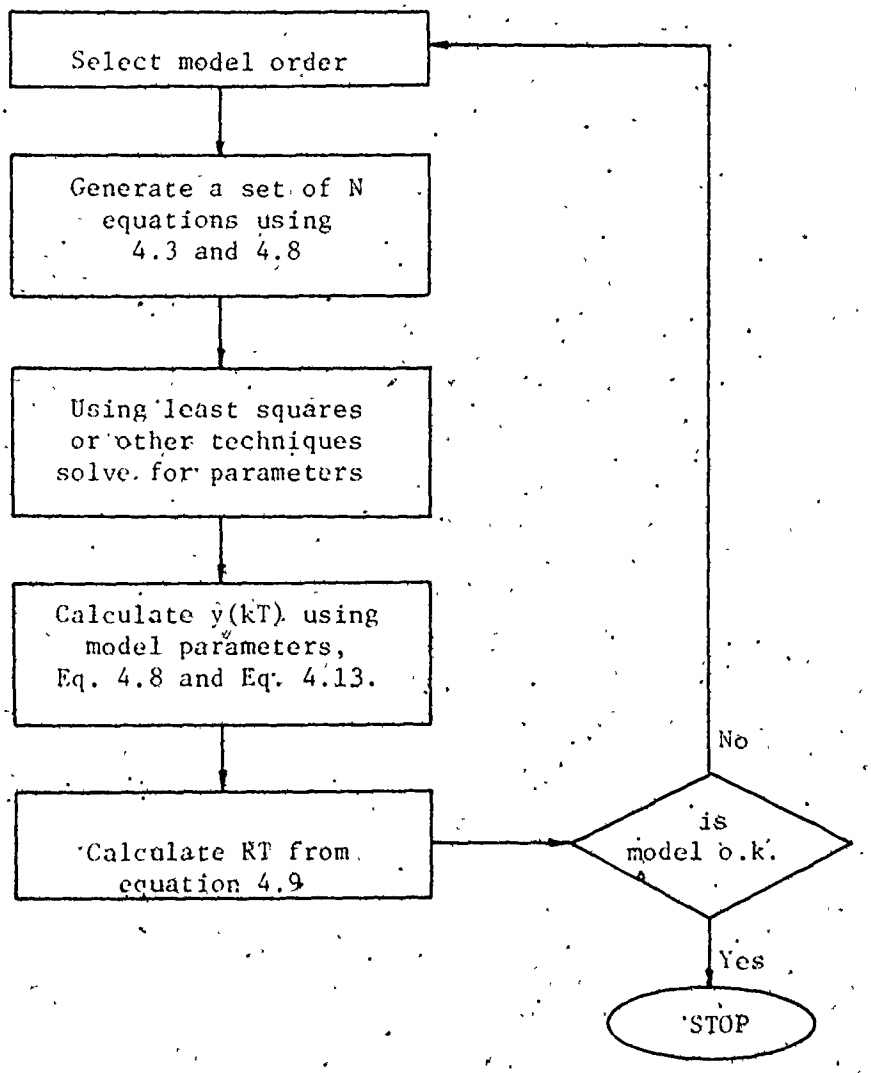


Fig. 4.12: Sequence of steps system identification.

4.6 Discussion of Results

The algorithm described above was implemented in FORTRAN-77 on the VAX-11/785 computer and used to obtain the transfer function of the robot arm.

Before applying it to the robot data, several simulations were tried to test the algorithm and its implementation. Input-output data corresponding to second, third and fourth order systems was generated and used as test input for the identification algorithm and the model obtained checked for accuracy. To test the robustness and performance of the algorithm, several parameters, such as the type of input, sampling time, signal to noise ratio, and the system order, etc., were varied.

We present here, results of modeling two simulations and the industrial robot.

(a) Simulated systems:

Case 1

Samples of the step response of the system described by

$$G(s) = \frac{1}{(s+0.6)^2+(3.5)^2} = \frac{1}{s^2+1.2s+12.61} \qquad (4.14)$$

were generated and stored. Results of the identification for different sampling intervals are presented in Table 4.1. The order of the system was assumed to be known. In each case 500 samples were used and no noise was introduced in the measurements.

Table 4.1: Effect of sampling interval

SNO	Structure [n,m]	T	RT	Parameters	Roots
1	2,0	0.5	0.999	1.265, 1.545, 15.955	-0.772 ± j 3.919
2	2,0	0.2	0.999	1.043, 1.270, 13.156	-0.635 ± j 3.571
3	2,0	0.1	1.0	1.010, 1.214, 12.744	-0.607 ± j 3.518
4	2,0	0.05	1.0	1.003, 1.204, 12.644	-0.602 ± j 3.504
5	2,0	0.01	1.0	1.000, 1.200, 12.611	-0.600 ± j 3.500
6	2,0	0.005	1.0	1.000, 1.200, 12.61	-0.600 ± j 3.500
7	2,0	0.001	Numerical ill conditioning caused abnormal termination of the program.		
8	Actual values			1.000, 1.200, 12.610	-0.600 ± j 3.500

It is evident that better accuracy is obtained for lower sampling intervals. This is expected, since more accurate approximation is made to the continuous time signal, when T is small. However, for very low sampling intervals, no results could be obtained due to numerical ill conditioning.

Table 4.2 lists the results of identifying the same system, but without prior knowledge of the structure. This time, the sampling interval was fixed at T = 0.1 secs. and 100 samples were used. As before there was no noise introduced.

Table 4.2: Effect of model order variation

SNO	Structure (n,m)	RT	Poles	Zeros
1	1,0	0.942	-1.088 ± j 0	-
2	3,1	0.618	-.605 ± j 3.577	1.308 ± j 0
			1.308 ± j 0	
3	4,1	-0.302E38	-0.579 ± j 3.489	
			-3.391 ± j 0	-3.544 ± j 0
			116.257 ± j 0	
4	4,2	-0.953	-0.606 ± j 3.516	
			1.125 ± j 0	1.125 ± j 0
			0.035 ± j 0	0.035 ± j 0
5	5,3	-0.136E29	-0.544 ± j 3.476	
			4.304 ± j 0	4.342 ± j 0
			0.319 ± j 0	0.321 ± j 0
			-0.225 ± j 0	-0.243 ± j 0
6	actual values		-0.600 ± j 3.500	-

Model orders ranging from 1 to 5 were tried. It is evident that for over-parameterized models, the excess roots in the numerator and denominator can be picked out since they are numerically close. Obviously, this is possible only when the number of excess poles and excess zeros are equal. A poor fit is indicated by the value of RT.

The effect of noise was considered by adding random Gausian noise to the step response data. Results corresponding to signal-to-noise ratios (SNR) of 30, 20 and 10 dB are shown in Table 4.3. In this case, it is assumed that the structure of the system is known in advance. As before, 100 samples at T = 0.1 secs were used. As indicated by the factor RT and the values obtained, less accurate models result as the noise level is increased.

Table 4.3: Performance in presence of noise.

SNO	SNR dB	RT	Roots
1	∞	0.999	$-0.607 \pm j\ 3.518$
2	30	0.998	$-0.664 \pm j\ 3.503$
3	20	0.988	$-0.768 \pm j\ 3.450$
4	10	0.906	$-0.909 \pm j\ 3.200$
5	actual values		$-0.600 \pm j\ 3.500$

Case 2

In this simulation, an attempt was made to identify the system from samples of the step response of

$$G(s) = \frac{s^2 + s + 2}{(s+1)[(s+0.6)^2+(3.5)^2]} \qquad (4.15)$$

A plot showing the step response is shown in figure 4.13.

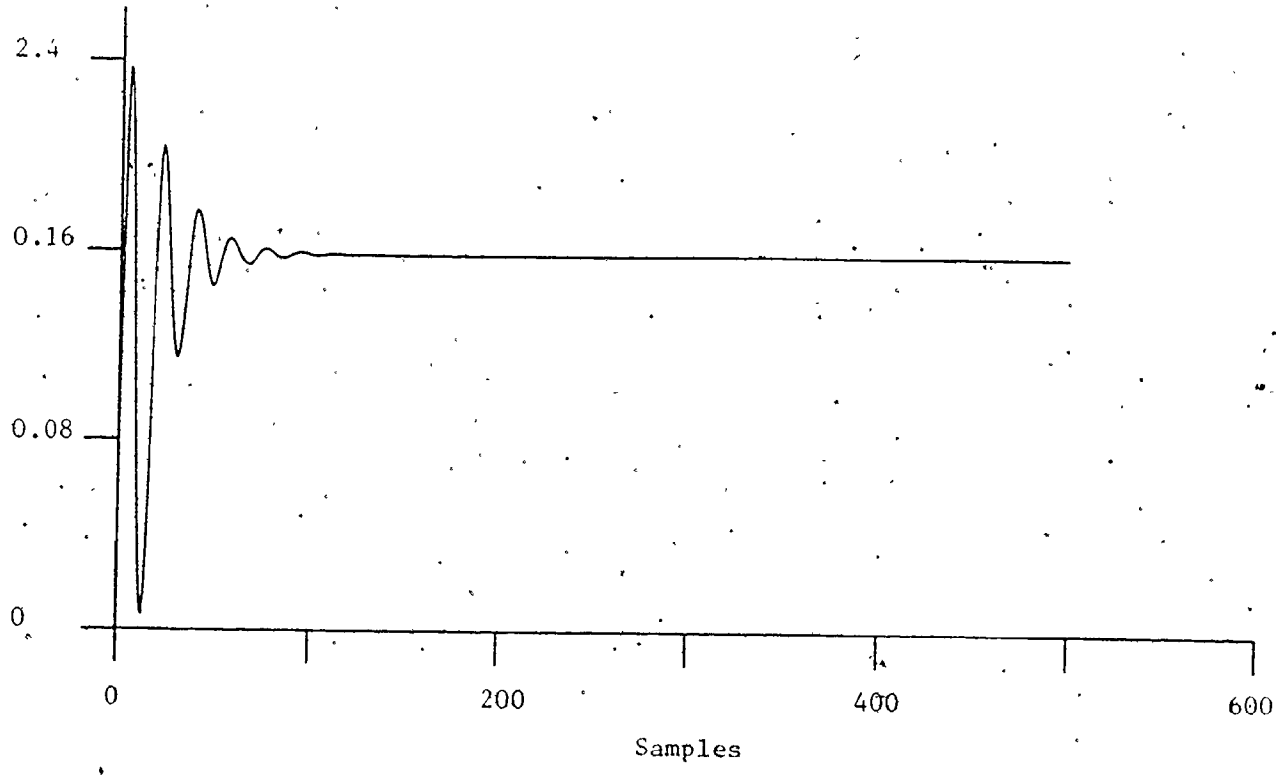


Fig. 4.13: Step response of simulated system.

As shown in Table 4.4, the method could be used successfully to determine the coefficients of the transfer function. The structure was assumed to be known, and no noise was added.

Table 4.4: Results for Case 2

SNO	Structure [n,m]	T	RT	Parameters	Poles	Zeros
1	3,2	0.1	0.9999	0.997,0.997,2.007 2.253,13.880,12.656	-0.625±j3.498 -1.002±j0	-0.500±j1.327
2	Actual values			1.0,1.0,2.0 2.2,13.81,12.61	-0.600+j3.500 -1.00±j0	-0.500±j1.323

A figure showing the plot of the model response y(t) and the actual step response y(t) is shown in figure 4.14. Both plots have been shown on the same set of axes for easy comparison. It is evident that the model output is close to the actual output.

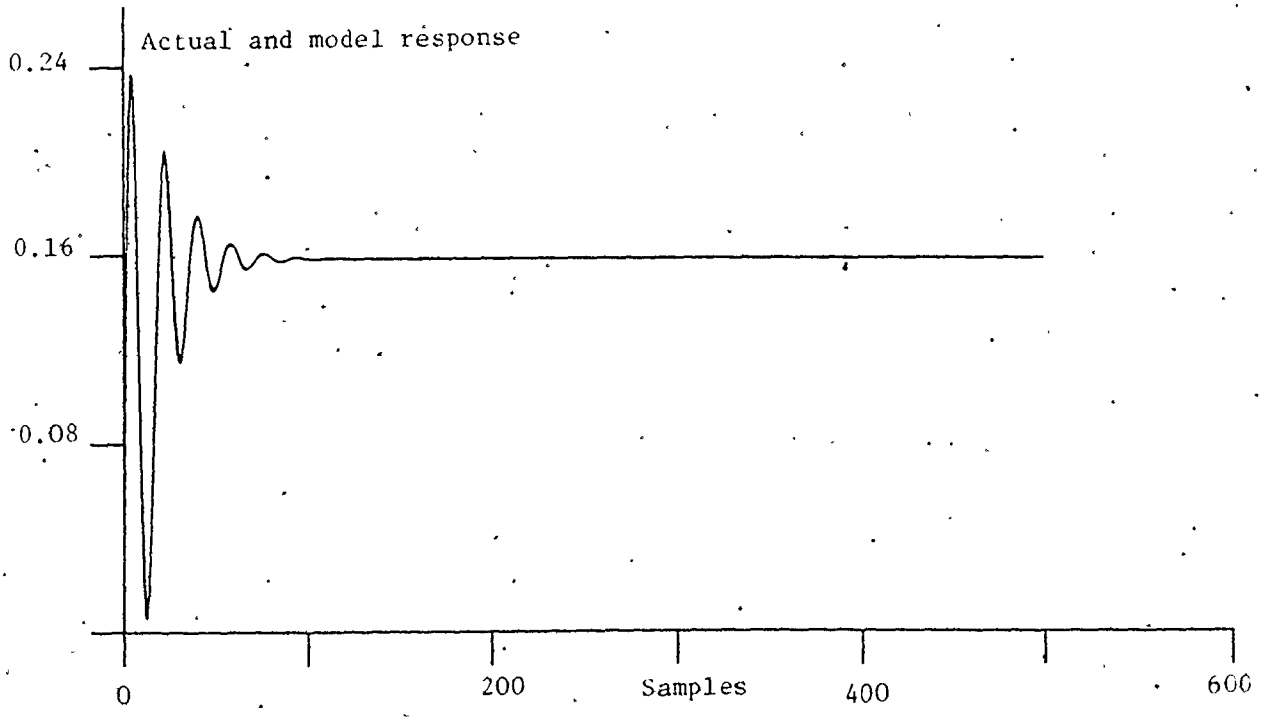


Fig. 4.14: Actual and model response of simulated system.

(b) Industrial robot

As mentioned before, design details are given only for one of the five axes. Data for modeling the robot arm corresponds to the 'rotate' axis which also has the lowest damping thus causing the greatest 'ringing'. Results of the modeling of the arm using different model structures were obtained. The step response of the model superimposed on the actual robot response are shown in figures 4.15 to 4.19.

Denominator order N: : 2
Numerator order M: : 0
Sampling interval T: : 0.008 seconds
Performance index: : 0:994903E+00

P O L E S

NO.	Real	Imaginary
1	-6.765588	7.797898
2	-6.765588	-7.797898

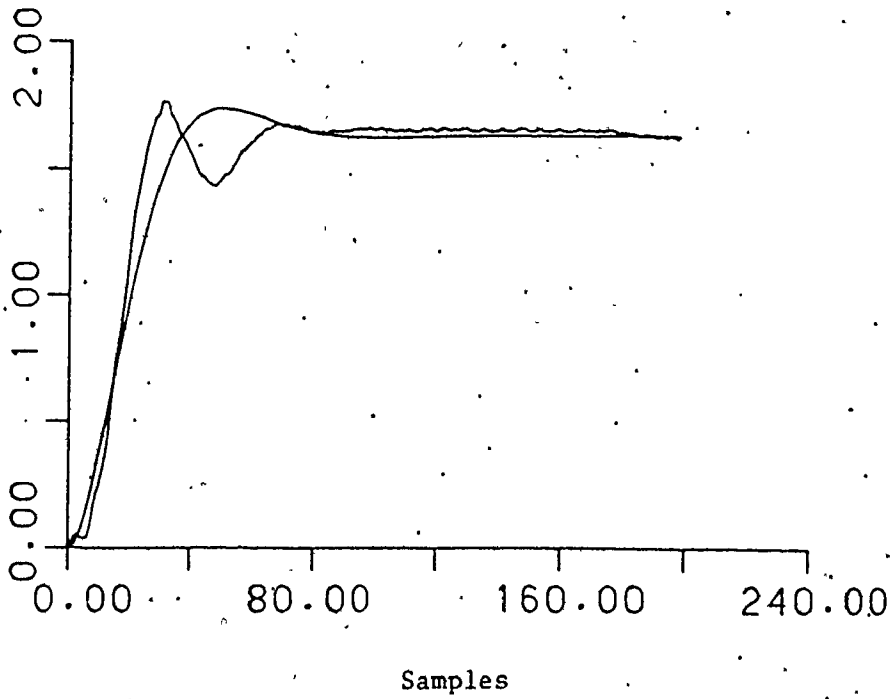


Fig. 4.15: Industrial robot: Second order model with no zeros.

Denominator order N: : 2
Numerator order M: : 1
Sampling Interval T: : 0.008 seconds
Performance index: : 0.995605E+00

P O L E S

NO.	Real	Imaginary
1	-7.482889	8.454309
2	-7.482889	-8.454309

Z E R O S

NO.	Real	Imaginary
1	90.745102	0.000000

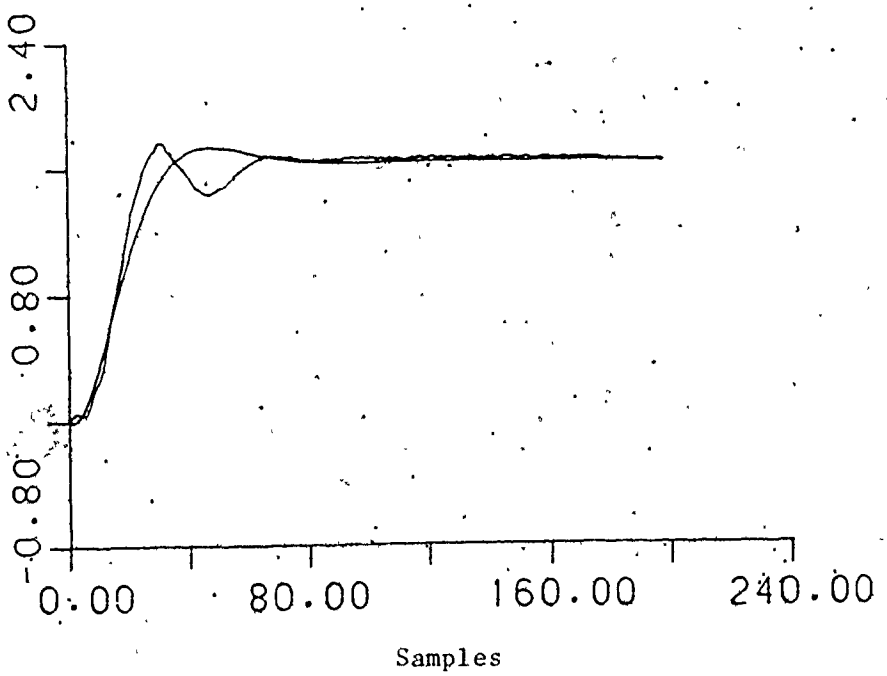


Fig. 4.16: Industrial robot: Second order model with one zero.

Denominator order N: : 3
Numerator order M: : 0
Sampling interval T: : 0.008 seconds
Performance index: : 0.999747E+00

P O L E S

NO.	Real	Imaginary
1	-3.894698	18.840031
2	-3.894698	-18.840031
3	-8.316282	0.000000

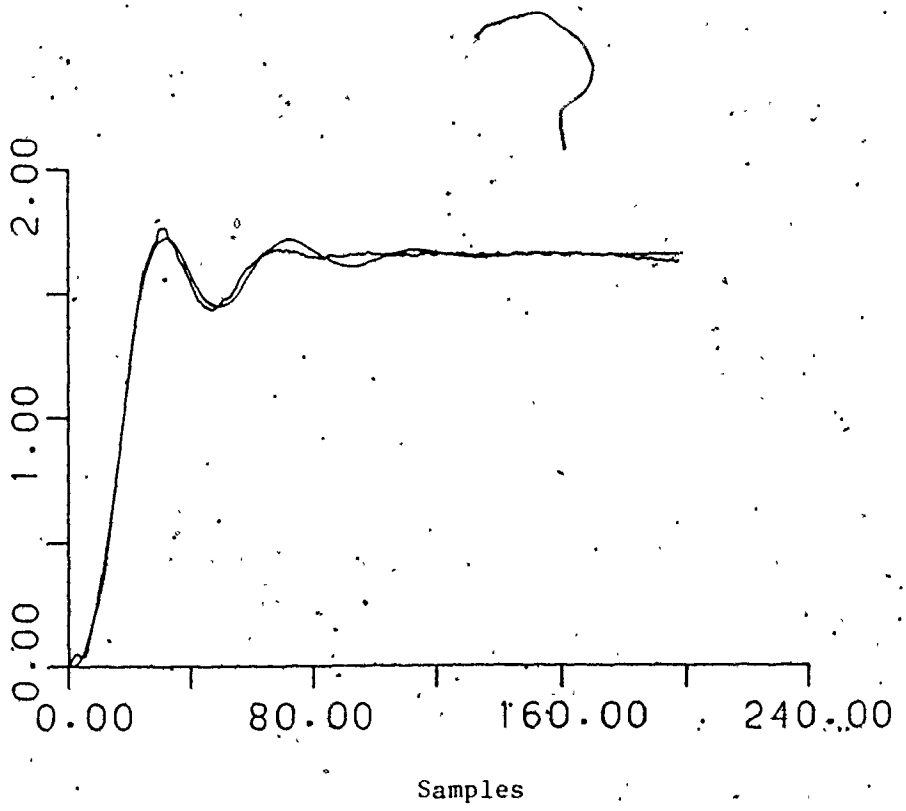


Fig. 4.17: Industrial robot: Third order model with no zeros.

Denominator order N: : 3
Numerator order M: : 1
Sampling interval T: : 0.008 seconds
Performance index: : 0.999777E+08

P O L E S

NO.	Real	Imaginary
1	-3.743144	19.354004
2	-3.743144	-19.354004
3	-8.416816	0.000000

Z E R O S

NO.	Real	Imaginary
1	330.972839	0.000000

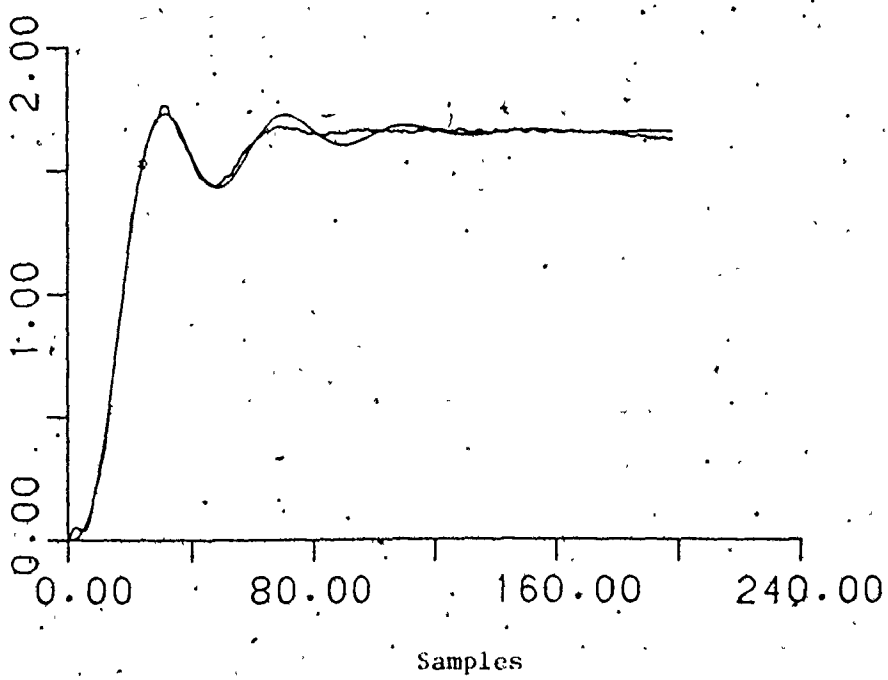


Fig. 4.18: Industrial robot: Third order model with one zero.

Denominator order N: : 4
Numerator order M: : 1
Sampling interval T: : 0.008 seconds
Performance index: : 0.999668E+00

P O L E S

NO.	Real	Imaginary
1	-3.392128	19.816912
2	-3.392128	-19.016912
3	-7.911962	0.000000
4	-0.310197	0.000000

Z E R O S

NO.	Real	Imaginary
1	-0.299213	0.000000

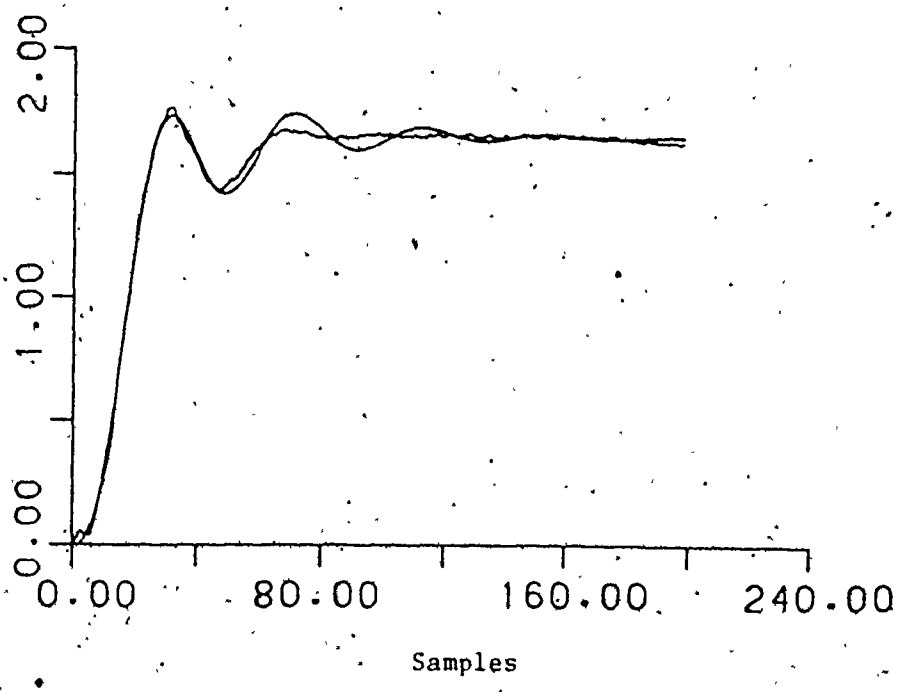


Fig. 4.19: Industrial robot: Fourth order model with one zero.

Location of the poles and zeros for the model and its fitness index are also included in the plot.

Since it is desirable to select the model with least complexity, a trade off has to be made between the accuracy and complexity. In this case the third order model with no zeros shown in figure 4.17 was considered most suitable.

4.7 Concluding Remarks

A model of the industrial robot is required for the design of an adaptive controller. There are two basic approaches to modeling the robot. The first, in which the robot is modelled from physical laws relating various subsystems, is not suitable because of its complexity and the lack of information on interaction of subsystems. The other approach known as the black box method involves modeling the robot from its input-output behaviour.

This chapter gives details of the measurement of input and output data, the theory of the algorithm for identification, and discusses the result of applying the theory to the data obtained for the industrial robot. Since the procedure is identical for all the five axes, details for only one of them is provided. In particular, the 'rotate' axis which is also the one with least damping was chosen.

The chapter is divided in five sections that deal with (1) data acquisition, (2) structural identification, (3) parameter estimation, (4) model verification and (5) discussion of results.

Although the PRBS is known to be an ideal input from the system identification point of view, it is often not possible to apply

it to real systems. In order to keep the identification algorithm as general as possible, the step signal was used as an input. Application of an input (test) signal to the robot required modification to the 8086 software on the master controller and an understanding of the 8748 software on the slave controller. Details of these modifications and data acquisition and storage are described in section 4.2.

Since conventional methods of system identification are not applicable with step response data, a two step method was used. Details of identifying the continuous time model by the 'direct' method are included.

Results are presented for two simulated systems for testing the performance of the algorithm and its implementation, followed by results of modeling the robot. Several second, third and fourth order models were obtained, and finally the third order model with no zeros was selected to represent the 'rotate' axis of the industrial robot for the development of the adaptive controller. Details of the design of the adaptive controller are given in the next chapter.

CHAPTER 5

ADAPTIVE CONTROL

5.1 Introduction

The concept of adaptive control was introduced in the early 1950's. It was motivated by the design of auto pilots for high performance aircraft designed after the second world war. The idea originated from the need for sophisticated controllers for the aircraft which operated over a wide range of speeds and altitudes.

However, since these controllers were based on the use of analog techniques, they were not very successful and economical. Research activity in this field was rejuvenated in the early 70's with the advent of digital computers, microprocessors, analog to digital (A/D) and digital to analog (D/A) convertors.

The fundamental difference between fixed feedback control systems and adaptive control systems is that the latter adapt their behaviour to the (changing) properties of controlled processes and their signals [52]. There are several ways of implementing an adaptive controller - a basic distinction can be made depending on whether it incorporates feed forward adaptation or feedback adaptation. The feed forward method is often called the 'gain scheduling' method. Adaptive controllers based on the feedback approach are further classified into two catagories: The model reference adaptive controller (MRAC) and the self tuning regulator (STR).

Due to several advantages of the STR over the MRAC, a self tuning type regulator was designed for the robot.

Contents of this chapter are in the following order. A brief description of the gain scheduling, MRAC and the STR type of adaptive controllers is first given. This is followed by a description of the discrete time model for the robot. Details of the design of the STR are then included. Two structures for the regulator were simulated. Results of the simulation tests are then given. The chapter concludes with a discussion on the performance of the regulator.

5.2 Gain Scheduling

Of the three schemes for adaptive control, the gain scheduling method is the only one based on feed forward adaptation. It is based on the idea that often in practice it is possible to find auxiliary variables which correlate well with changes in the system dynamics. If it is known in advance, how the controller is to be adapted in dependance on these variables, the system can be controlled by changing the parameters of the regulator [51,65]. In the context of the auto pilot, the Mach number and dynamic pressure are measured by air data sensors, and used as scheduling variables. In the case of the robot arm used for arc welding, since the dynamics change with the length and position of various joints; joint coordinates may be used as the auxiliary variables.

In a typical design procedure, auxiliary variables are first identified. Regulator parameters are then determined at a number of

operating positions, using some suitable design method. Regulator parameters are then stored for use on-line. A block diagram of this type of an adaptive controller is shown in figure 5.1.

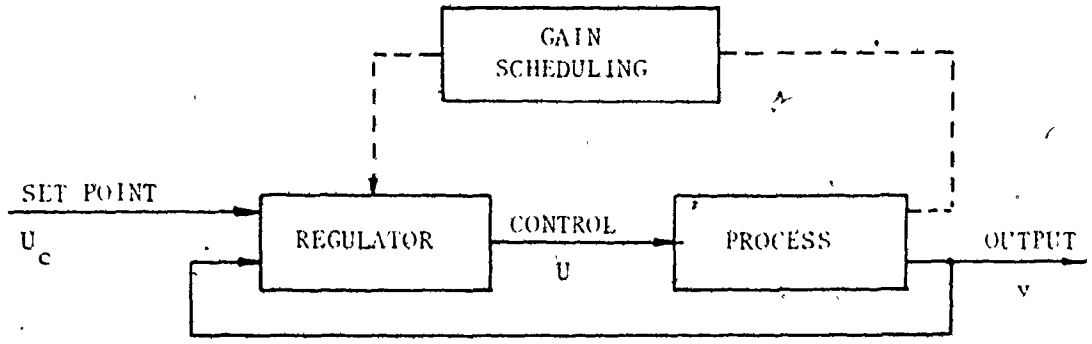


Fig. 5.1: Block diagram of gain scheduling method.

Particular attention has to be given to the transition between different operating points. The number of intermediate points is increased if necessary.

Gain scheduling has the advantage that the regulator parameters can be changed quickly - in response to system changes - limited by how fast auxiliary variables relate to system changes.

The main disadvantage is that it is an open loop method and as such there is no feedback which compensates for an inaccurate schedule. Also the design is very time consuming. In effect, one has to determine regulator parameters for many systems and store the values in a look-up table. The size of the look up table depends on the number of intermediate points that are used to design the regulator. For very accurate control, the number of points may be large, thus resulting in a large table.

A combination of the gain scheduling method and a feedback method may be used to advantage, the gain scheduling may be used to control gross movement and the other for finer control.

5.3 Model Reference Adaptive Control

As mentioned before, the model reference adaptive controller (MRAC) is based on feedback adaptation. In the MRAC, the characteristics of the desired closed loop system are specified in terms of a model [56]. The difference between the desired response and the response of the process is used in an adaptive way, to force the process response to track that of the model.

Basically, the MRAC can be implemented in two ways: (i) the parameter adaptation method and, (ii) the signal synthesis method. A block diagram of the parameter adaptation scheme is shown in figure 5.2.

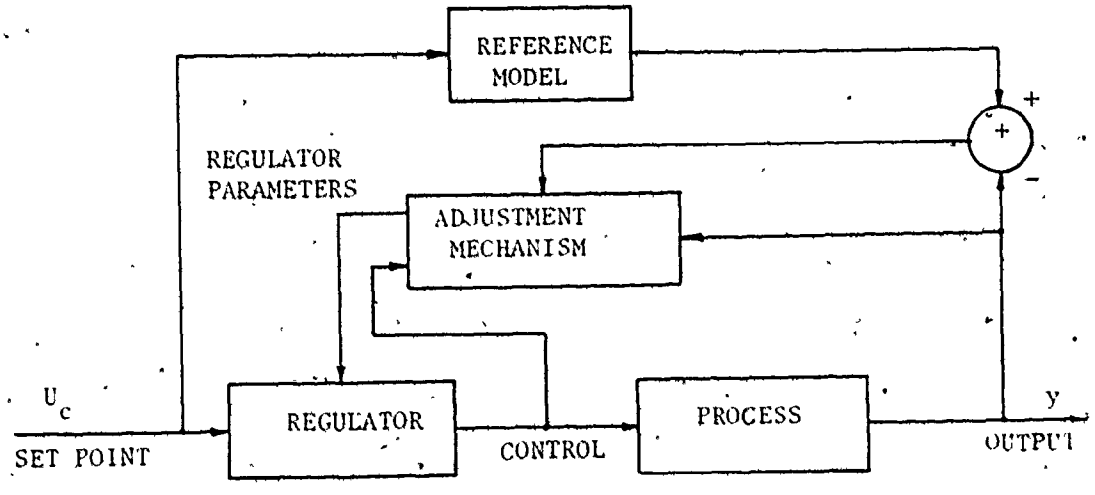


Fig. 5.2: Block diagram of parameter adaptive MRAC.

Here, as the name suggests, feedback from the adjustment mechanism is used to control the parameters of the regulator. In the signal synthesis method, shown in figure 5.3, the plant response is forced to track the model output by manipulating input signals.

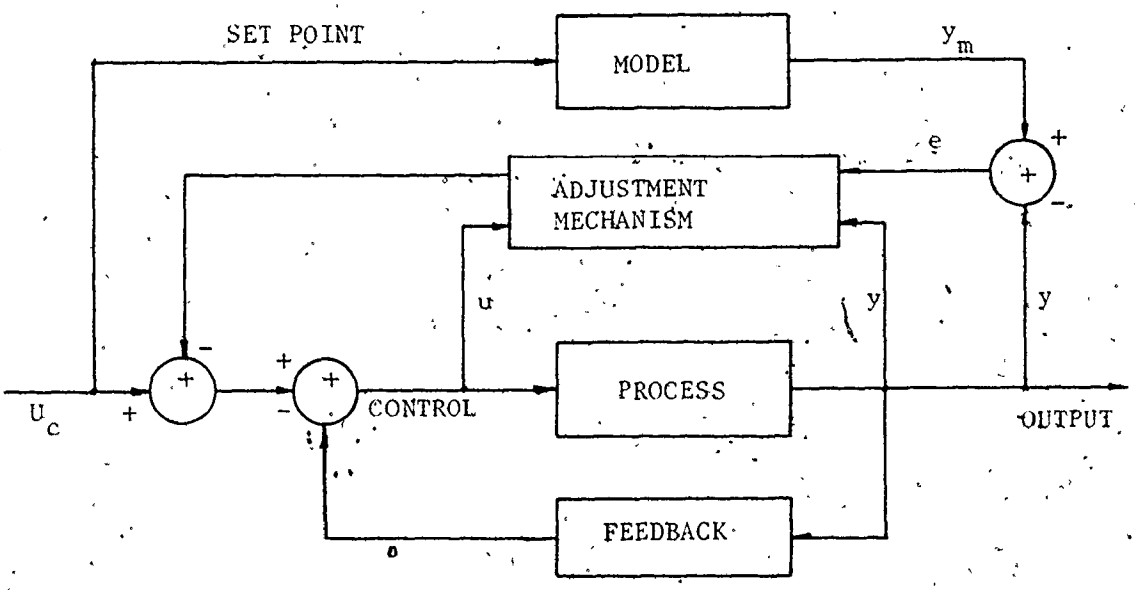


Fig. 5.3: Signal synthesis type MRAC.

Since in general, the number of regulator parameters are more than the number of signals, the signal synthesis approach leads to a relatively simpler realization and implementation of the adaptive controller. Details of analysis and design of the MRAC system is given in several books and papers [53,54,57,59,60-62].

The MRAC was not used for the control of the industrial robot due to some of its limitations listed below.

The following is a list of some observations on the model reference adaptive control method.

1. MRAC is a feedback method of adaptation which does not require on-line identification of the process. The error between the process and model output is used to adjust the controller parameters or the control signal.

2. The error signal is a measure of the performance quality.

3. The stability of the closed loop system can be assured only when the relative order (i.e. the difference between the number of poles and zeros) of the process is known. Also, the higher the relative order the greater the complexity of the MRAC.

4. Reference models with a smaller relative order than the system, leads to a large (practically unbounded) control signal.

5. Since MRAC schemes, conceptually, involve cancellation of the process poles and zeros, they can not be used for non-minimum phase systems. This is important, since often in practice, the conversion of continuous time models (including those which are minimum phase) to discrete time models, result in non-minimum phase systems.

5.4 Self Tuning Regulator

The second type of feedback adaptive system is the self tuning regulator (STR). STR was first proposed under the title 'self optimizing system', by Kalman in 1958 [52]. This proposal had little

immediate consequence because both theory and technology were not developed sufficiently enough.

In recent years, both problems have been significantly reduced. The theory was revived and extended by Peterka [52] and later by Astrom and Whittenmark in 1973 [64]. Strides made in digital technology, in particular, the availability of computers and microprocessors, have made it possible to apply these theories.

The STR basically consists of three elements as shown by the block diagram in figure 5.4.

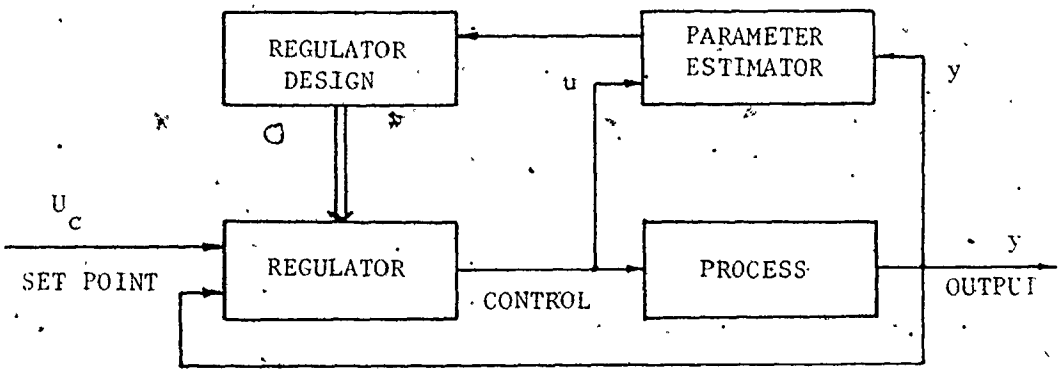


Fig. 5.4: Block diagram of self tuning regulator.

A recursive parameter estimator continuously monitors the plant input and output signals and computes the estimates of the plant dynamics in terms of a parameteric model. The structure of the model is pre-assigned. Current parameter estimates are used in a control design algorithm which generates the regulator parameters. The unit shown as 'regulator design', represents on-line solution to the controller design problem, for a system with known parameters.

The control law is usually in the form of a difference equation which acts on the output, reference input (or set point) and past control signals to generate the new signal. Since this type of a STR involves identification of the system model in an explicit way, it is called an 'Explicit STR'. It is sometimes possible to re-arrange the design of the regulator such that it can be expressed directly in terms of regulator parameters. A regulator based on this approach is called an 'Implicit STR'.

The use of an implicit algorithm leads to a significant simplification of the STR, because steps involving design calculations are eliminated. Such an approach is however, possible for minimum phase systems only.

Many different parameter estimation schemes have been used; for example, stochastic approximation, least squares, instrumental variables, Kalman filtering, and the maximum likelihood methods.

The choice of identification method depends on the process, disturbances and the time available to perform calculations. In the case of the industrial robot, the recursive least squares method was used because of its advantages of simplicity in terms of the number of computations.

Several regulator designs have been used in the past. The most common are: (i) minimum variance and the linear quadratic control, and (ii) pole/zero assignment techniques. The problem with the optimal control strategies is again related to the non minimum phase behaviour of most discrete time systems.

Minimum variance strategies may try to cancel the non minimum phase zeros with unstable poles. Suboptimal solutions may be used but they are often algorithmically more involved, requiring on-line polynomial factorization or equivalently, iteration of a Ricati equation.

A possible practical difficulty which may occur in minimum variance regulators due to these cancellations is that the amplitude of the control signals may be large and may exceed the dynamic range of the D/A converter used to drive the plant input actuator.

The other commonly used method is intutively more appealing from an engineering point of view. In the 'pole/zero assignment' method, the desired response of the system is specified in terms of the closed loop transfer function. The motivation for such an approach is that it is easy to relate pole-zero locations to transient performance.

The problem of non minimum phase plants is avoided by cancelling only those zeros of the system, that are outside the unit circle. This however requires on-line polynomial factorization. A variation of this approach, which avoids factorization is the 'pole placement' algorithm, in which none of the zeros are cancelled. Pole assignment self tuners are therefore the most reliable controllers for non minimum phase systems.

Complete design procedure for the self tuning regulator for the industrial robot is given in the following sections. The critical computation is the solution of a Diophantine equation which is known to have numerical problems if there are common factors in the model

polynomials. Another problem arises from the parameter estimator, when a constant forgetting factor (<1) is used for discounting old data. When the self tuner operates for some time with little or no disturbances the value of the co-variance matrix increases rapidly, so that when a disturbance occurs, the estimation gain is high and the control exercised is poor. This problem is resolved by adjusting the forgetting factor according to an 'information measure'. The value of the factor is made close to 1 when there is no new information available from the data.

Details of the design and performance of the STR for the industrial robot are given in the remaining sections of this chapter.

As a closing comment on this brief description of various methods of adaptive control, it may be pointed out that recently several papers have been published on combining the MRAC and STR approaches. It has been established that for some cases, notably for minimum phase systems, the two approaches are identical. However, in general for non-minimum phase systems, the pole placement STR is the most reliable and convenient to implement. As will be seen in section 5.5.1, the model for the robot is non minimum phase.

5.5 Design of Self Tuning Regulator

As explained in the previous section, a self tuning regulator is based on recursive identification of a model of the process, and on-line use of these model parameters for the design of a controller. The desired performance is specified in terms of a closed loop model. The control action is to move the poles (and sometimes the zeros), to

those specified. A block diagram of such a regulator is given in figure 5.4.

The design procedure is influenced by the choice of (i) the model for the process, (ii) structure of the compensator and, (iii) choice of the parameter estimation scheme. These steps are described in greater detail in the following sections.

5.5.1 Discrete-Time Model for the Robot Arm

In the self tuning regulator, the process to be controlled has to be modelled on-line. Several methods such as the least squares, instrumental variables, maximum likelihood method, etc., are applicable. However, the structure of the model is assumed fixed and known a-priori. In the previous chapter, on modeling the robot arm, a third order model was seen to adequately describe the robot arm. The model chosen was in the s-domain and is reproduced below.

$$H(s) = \frac{1}{(s+8.316282)[(s+3.894698)^2 + (18.840031)^2]} \qquad (5.1)$$

Since the self tuner is to be implemented on a digital computer, the process model has to be expressed in the discrete-time form or in the z-domain.

A z-domain function was derived from the transfer function in Eq. 5.1, by assuming a zero-order hold and a sampler as shown in figure 5.5.

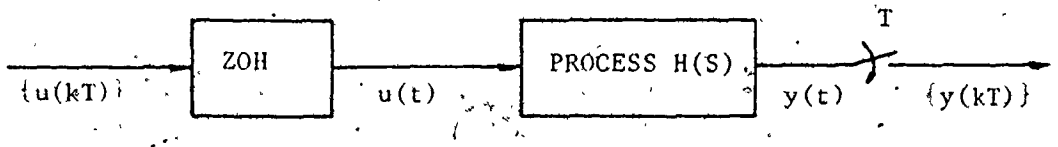


Fig. 5.5: Diagram showing zero-order hold approximation.

The z-domain transfer function from the input of the ZOH to the output of the sampler is obtained as shown in equation 5.2.

$$H(z) = \frac{Y(z)}{U(z)} = (1-z^{-1})\ Z\ \{L^{-1}[\frac{H(s)}{s}]\} \qquad (5.2)$$

where U(z) and Y(z) are the z-transforms of the input and output sequences {u(kT)} and {y(kT)} respectively. T corresponds to the sampling interval, k the time index, L^{-1} and Z the inverse Laplace and the z-transform operators respectively.

Choice of sampling interval:

The sampling interval T is chosen next. We see from Eq. 5.1 that the time constants associated with the system are τ_1 = 0.12 secs and τ_2 = 0.25 secs. The sampling interval is chosen to be somewhat smaller than the smaller of the system time constants. Also, it may be noted that choosing a very small interval may result in numerical ill conditioning. In addition, since in a STR, the identification, calculation of parameters and the generation of the control signal have to be done within the sampling interval, choosing a very small value may not be feasible. These points are illustrated here by

discretizing the model (Eq. 5.1) for T = 0.1, 0.05, 0.025 and 0.01 secs.

Following the procedure in Eq. 5.2, we have

For T = 0.1 secs

$$H(z) = \frac{0.9405208314\ z^{-1} + 2.080534434\ z^{-2} + 0.42109749\ z^{-3}}{1 - 0.0179013064\ z^{-1} + 0.277164697\ z^{-2} - 0.1997761501\ z^{-3}} \tag{5.3}$$

For T = 0.05 secs

$$H(z) = \frac{0.16328826\ z^{-1} + 0.5114003\ z^{-2} + 0.10918917\ z^{-3}}{1 - 1.6279936\ z^{-1} + 1.3162310\ z^{-2} - 0.446961\ z^{-3}} \tag{5.4}$$

For T = 0.025 secs

$$H(z) = \frac{0.02330\ z^{-1} + 0.083387157\ z^{-2} + 0.01905212\ z^{-3}}{1 - 2.4291611\ z^{-1} + 2.136415062\ z^{-2} - 0.6685514314\ z^{-3}} \tag{5.5}$$

and for T = 0.01 secs

$$H(z) = \frac{0.001598191\ z^{-1} + 0.006130115\ z^{-2} + 0.001474539\ z^{-3}}{1 - 2.8097667\ z^{-1} + 2.66384309\ z^{-2} - 0.85124373\ z^{-3}} \tag{5.6}$$

The location of poles and zeros for the models described above are shown in Table 5.1.

Table 5.1: Location of Roots for Robot Models

Root \ T	0.1 secs	0.05 secs	0.025 secs	0.01 secs
Poles	0.4353±j0	0.6598±j0	0.8123±j0	0.9202±j0
	-0.2087±j0.6445	0.4841±j0.6656	0.8084±j0.4117	0.9448±j0.1801
Zeros	-1.9868±j0	-2.9014±j0	-3.3328±j0	-3.5778±j0
	-0.2254±j0	-0.2305±j0	-0.2453±j0	-0.2579±j0

The following observations may be made from table 5.1.

(i) None of the models are minimum phase.

(ii) The unstable zero moves towards the unit circle as the sampling period T is increased.

(iii) Poles of the system move towards the unit circle as the sampling period T is decreased.

Also it may be observed from Eqs. 5.3 to 5.5 that as the value of T is decreased, the coefficients of the numerator get smaller and those of the denominator increase. Since the numerator values become very small, low values of T may cause numerical problems.

A sampling interval of 0.1 secs was chosen for the self tuning regulator for the industrial robot. Since the model is non-minimum phase, the pole placement type of STR would be suitable.

5.5.2 Structure of the Regulator

A compensator can be configured in several ways. The two commonly used approaches are shown in figures 5.6 and 5.7.

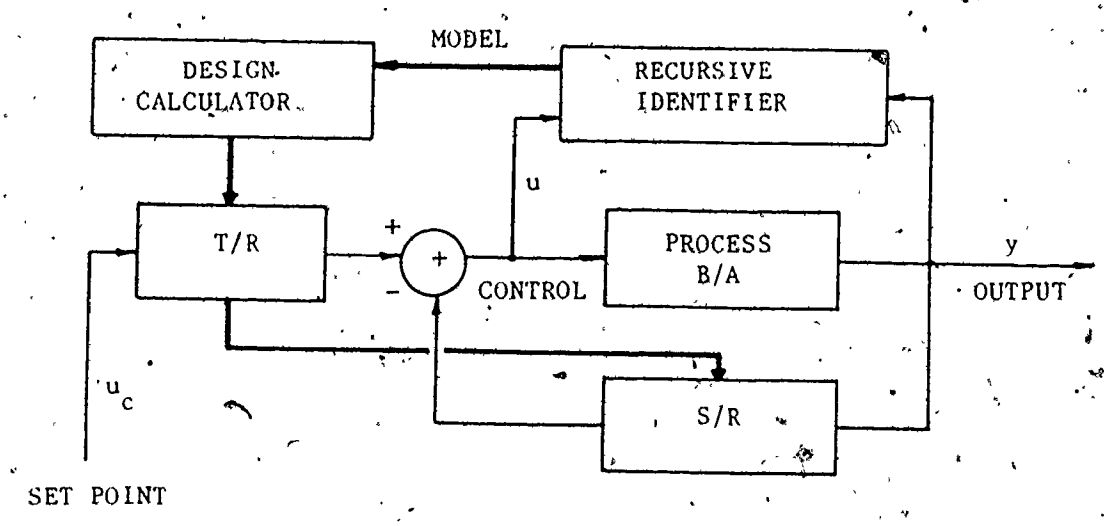


Fig. 5.6: Self Tuning Regulator ... I.

This configuration has been used by several authors [51,52, 64-66,68], and has the advantage that it allows some flexibility for cancellation of stable zeros. This is described in the next section.

The configuration shown in figure 5.7 originates from the observer-controller scheme popularly used in the state-space theory.

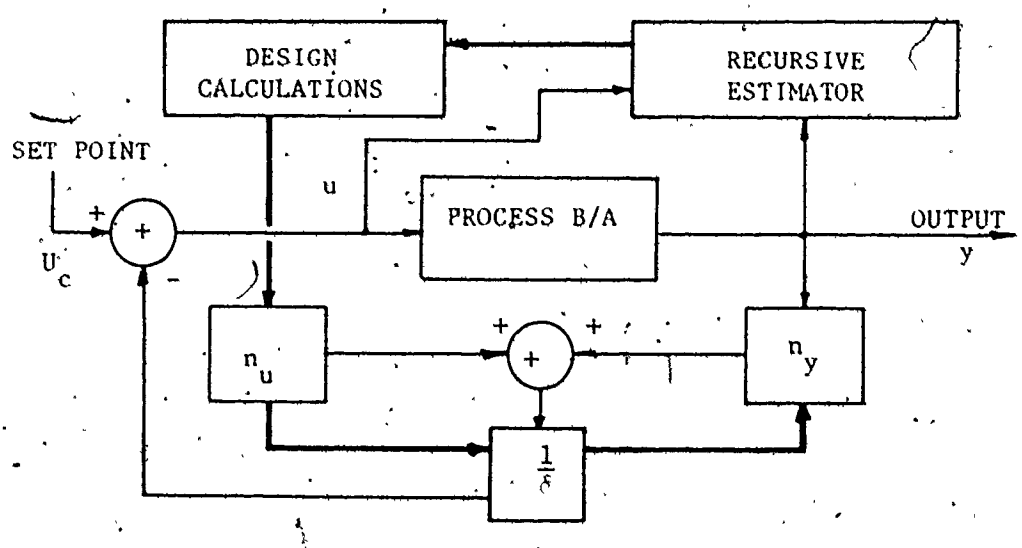


Fig. 5.7: Self tuning regulator ... II.

This was first proposed by Wolvovich [72]. Its interpretation from a transfer function point of view is given by Kailath [63]. In this case, there is a feedback from the control signal as well as the output signal. It is seen that with full computer precision, in a purely pole placement design, the two approaches yield identical results.

5.5.2.1 Details of Design (Scheme I)

In this section, we describe the design of a regulator for the industrial robot, based on figure 5.6. The recursive parameter estimator is described in section 5.5.4. The regulator part of the adaptive controller is drawn below.

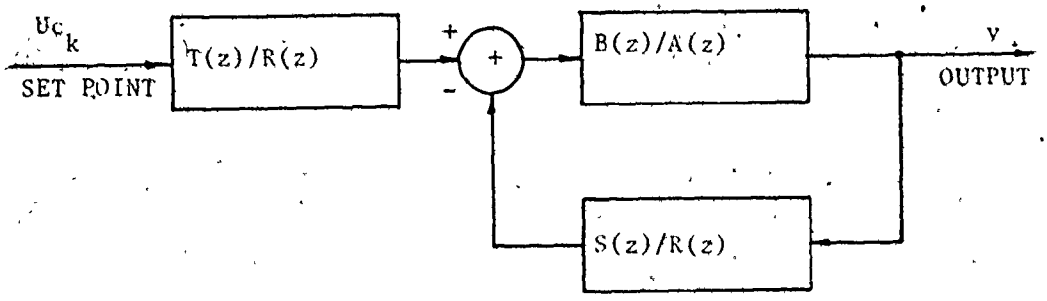


Fig. 5.8: Block diagram of regulator (I).

The relation between the control signal u and the output signal y is given by the transfer function of the system or,

$$H(z) = B(z)/A(z) \tag{5.6}$$

where B(z) and A(z) are polynomials with no common factors. The

function H(z), represents the dynamics of the system including the hold circuit, actuator, drive, sensors, etc.

The desired closed loop pulse transfer function is specified by

$$H_m(z) = \frac{B_m(z)}{A_m(z)} \tag{5.7}$$

where $B_m(z)$ and $A_m(z)$ are polynomials with no common factors.

In general, with output feedback, there will be additional dynamics, which are not excited by the command signal. These are specified in terms of the 'observer' polynomial $A_0(z)$.

We have from figure 5.8

$$U(z) = \frac{T(z)}{R(z)} U_c(z) - \frac{S(z)}{R(z)} Y(z) \tag{5.8}$$

The above may be rewritten, after dropping 'z' for convenience

$$RU = TU_c - SY \tag{5.9}$$

We have from Eq. 5.6

$$H = \frac{Y}{U} = \frac{B}{A}$$

or

$$YA = BU \tag{5.10}$$

Eliminating U from Eq. 5.10

$$YA = B \frac{T}{R} U_c - \frac{BS}{R} Y$$

or

$$\frac{Y}{U_c} = \frac{BT}{AR + BS} \tag{5.11}$$

This equation represents the overall transfer function of the closed loop system from the set point to the output. Since the desired closed loop performance is specified by H_m, we have

$$M_{\text{closed loop}} = \frac{Y}{U_c} = \frac{BT}{AR + BS} = \frac{B_m A_0}{A_m A_0} \tag{5.12}$$

where A_0 represents the cancelled dynamics.

The design problem is to determine the polynomials R, S and T that satisfy Eq. 5.12.

Pole-zero cancellations:

The poles of the closed loop system are the solution to the characteristic Eq.

$$AR + BS = 0 \tag{5.13}$$

The zeros of the closed loop system are the zeros of B and T. Since, these closed loop poles and zeros are also specified by H_m, in general, there have to be cancellations. Consider first the open loop zeros, i.e. the zeros of the polynomial B. If a factor of B is not a

factor of B_m then it has to be cancelled by a factor of AR + BS. Since the closed loop system must be stable, it is essential that only stable zeros be cancelled.

Let

$$B = B^+B^- \tag{5.14}$$

where B^- has all the zeros outside the unit circle and B^+ has all zeros inside. Since B^- contains 'unstable' zeros they can not be cancelled and have to be retained in the closed loop transfer function. All zeros in B^+ may be cancelled. A unique factorization may be obtained by constraining B^+ to be a monic polynomial. It is clear that this procedure permits the design of regulators for non-minimum phase systems also.

Since B^- is not a factor of AR+BS, it has to be a factor in B_m, or

$$B_m = B^-B'_m \tag{5.15}$$

Also since B^+ is cancelled by a factor of AR+BS, and A and B have no common factors

$$R = R'B^+ \tag{5.16}$$

Using the relations derived above, Eq. 5.12 may be rewritten as

$$\frac{Y}{U_c} = \frac{B^+B^-T}{B^+(AR'+B^-S)} = \frac{B^-B'_mA_0}{A_mA_0} \tag{5.17}$$

$$\frac{Y}{U_c} = \frac{T}{AR' + B^- S} = \frac{B'_m A_0}{A_m A_0} \tag{5.18}$$

Therefore

$$T = B'_m A_0 \tag{5.19a}$$

and

$$AR' + B^- S = A_m A_0 \tag{5.19b}$$

Steps involved in the design of the regulator may be summarized:

1. Factorize B as $B^+ B^-$.
2. Determine R' and S satisfying Eq. 5.19b.
3. Evaluate R and T using Eqs. (5.16), (5.19a) and R' and S obtained in step 2.
4. Calculate the control signal using Eq. 5.9.

The polynomials A, B, are either available or specified. The degrees of A_m, B_m, A_0, R, S and T are derived from constraints of causality. These conditions are listed below [51].

1. $$[\text{Deg } A_m - \text{Deg } B_m] \geq [\text{Deg } A - \text{Deg } B] \tag{5.20a}$$

2. $$\text{Deg } A_0 \geq 2 \text{ Deg } A - \text{Deg } A_m - \text{Deg } B^+ - 1 \tag{5.20b}$$

3. $$\text{Deg } R \geq \text{Deg } T \tag{5.20c}$$

4. $$\text{Deg } R \geq \text{Deg } S \tag{5.20d}$$

Also, we have

$$\text{Deg}(AR) = \text{Deg}(AR+BS) = \text{Deg}(B^{+}A_0A_m)$$

or

$$\text{Deg } R = \text{Deg } A_0 + \text{Deg } A_m + \text{Deg } B^{+} - \text{Deg } A \tag{5.21}$$

Design details of regulators for industrial robot

We have from Eq. 5.3

$$H(z^{-1}) = \frac{z^{-1}[0.9405208314 + 2.080534434z^{-1} + 0.42109749z^{-2}]}{1 - 0.0179013064\ z^{-1} + 0.277164697\ z^{-2} - 0.1997741501\ z^{-3}}$$

or

$$H(z) = \frac{0.9405208314\ z^2 + 2.080534434\ z + 0.42109749}{z^3 - 0.0189013064\ z^2 + 0.277164697\ z - 0.1997741501} = \frac{B}{A} \tag{5.22}$$

In general,

$$H(z) = \frac{b_2z^2 + b_1z + b_0}{z^3 + a_2z^2 + a_1z + a_0} \tag{5.23}$$

We know from table 5.1 that the model for the robot is non-minimum phase and as such has zeros outside the unit circle. Since, unstable zeros cannot be cancelled, we would have to factorize the polynomial B. Keeping in mind that a new model for the robot is obtained [from recursive identifier] at every sampling interval, this factorization has to be performed on-line, within each sampling interval.

One way to avoid this is to retain all open loop zeros in the closed loop system. The problem is strictly a pole-assignment problem.

$$\therefore \quad B^{+}=1 \text{ and } B^{-}=B = b_2 z^2 + b_1 z + b_0 \tag{5.24}$$

The closed loop performance is specified by $H_m = B_m/A_m$. Since all zeros are retained, let $B_m=B$. Also, for a unity gain

$$H_m = \frac{A_m(1)\, B}{B(1)\, A_m} \tag{5.26}$$

From conditions listed in Eq. (5.20), orders of various polynomials are evaluated

(a) $\text{Deg } A_m \geq \text{Deg } A - \text{Deg } B + \text{Deg } B_m$

$\therefore \text{Deg } A_m \geq 3 - 2 + 2$

≥ 3

Let $\text{Deg } A_m = 3$ and the general form of A_m be

$$A_m(z) = P_3 z^3 + P_2 z^2 + P_1 z + P_0. \tag{5.27}$$

$$A_m(1) = \sum_{i=0}^{3} P_i \quad \text{and} \quad B(1) = \sum_{i=0}^{2} b_i$$

$$H_m(z) = \frac{B_m(z)}{A_m(z)} = \frac{\sum_{i=0}^{3} P_i}{\sum_{i=0}^{2} b_i} \cdot \frac{[b_2 z^2 + b_1 z + b_0]}{[P_3 z^3 + P_2 z^2 + P_1 z + P_0]} \qquad (5.28)$$

$\therefore \quad B_m = B \bar{B}'_m = BB'_m$

From Eq. 5.28

$$B'_m = \sum_{i=0}^{3} P_i \Big/ \sum_{i=0}^{2} b_i = K_m \qquad (5.29)$$

(b) Deg $A_0 \geq 2$ Deg A - Deg A_m - Deg $B^+ - 1$

or Deg $A_0 \geq 2$

Let

Deg $A_0 = 2$ and A_0 be of the form $x_2 z^2 + x_1 z + x_0$ (5.30)

This polynomial is chosen arbitrarily.

One of the choices is to make the dynamics fast, i.e. a deadbeat observer.

or $A_0 = z^2$ (5.31)

We have

$T = B'_m A_0$

$T(z) = K_m z^2$ (5.32)

From Eq. 5.21

$$\text{Deg } R = \text{Deg } A_0 + \text{Deg } A_m + \text{Deg } B^+ - \text{Deg } A$$

or

$$\text{Deg } R = 2$$

$$R = R'B^+$$

$$\text{Deg } R' = 2$$

Let

$$R'(z) = r_2 z^2 + r_1 z + r_0 \tag{5.33}$$

(c) The condition of Eq. 5.20c is satisifed.

(d) Since we have Deg $R \geq$ Deg S

Let

$$\text{Deg } S = \text{Deg } R$$

and

$$S(z) = S_2 z^2 + S_1 z + S_0 \tag{5.34}$$

The next step involves solving for the coefficients of the polynomials R and S. This is done by solving the equation $AR' + B^- S = A_m A_0$.

In this case, because $B^+=1$, $R=R'$ and $B^-=B$,

∴ Eq. (5.19b) may be rewritten as

$$AR + BS = A_m A_0 \tag{5.35}$$

This class of equations is called the Diophantine equation. We have from Eq. 5.35

$$[z^3 + a_2z^2 + a_1z + a_0][r_2z^2 + r_1z + r_0] + [b_2z^2 + b_1z + b_0][s_2z^2 + s_1z + s_0] = z^2[P_3z^3 + P_2z^2 + P_1z + P_0] \quad (5.36)$$

In matrix form

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ a_2 & 1 & 0 & b_2 & 0 & 0 \\ a_1 & a_2 & 1 & b_1 & b_2 & 0 \\ a_0 & a_1 & a_2 & b_0 & b_1 & b_2 \\ 0 & a_0 & a_1 & 0 & b_0 & b_1 \\ 0 & 0 & a_0 & 0 & 0 & b_0 \end{bmatrix} \cdot \begin{bmatrix} r_2 \\ r_1 \\ r_0 \\ s_2 \\ s_1 \\ s_0 \end{bmatrix} = \begin{bmatrix} P_3 \\ P_2 \\ P_1 \\ P_0 \\ 0 \\ 0 \end{bmatrix} \quad (5.37)$$

This system of equations may be solved for s_2, s_1, s_0, r_2, r_1 and r_0 using any numerical technique. The three popular methods are:

1. LU factorization (Crout's method).
2. Gauss-Seidel relaxation algorithm, and
3. Successive over-relaxation method.

The first method is a direct one, while the other two are iterative, using successive approximation.

Iterative methods are useful for solving systems of linear equations, when dealing with large sparse matrices. They are also effective when dealing with ill-conditioned systems. In this case,

the Crout's method was used because it has the advantage that it is particularly well adapted for programmed computers.

The final step involved the evaluation of the control signal u. This is achieved by using the control law in Eq. 5.9.

We have

$$RU = TU_c - SY$$

or

$$[r_2 z^2 + r_1 z + r_0]U(z) = K_m z^2 U_c(z) - (S_2 z^2 + S_1 z + S_0) Y(z)$$

Converting to time domain and dropping the sampling interval T for convenience, we get

$$U(K+1) = \frac{1}{r_2} [K_m U_c(k+1) - \sum_{i=0}^{2} S_i y(k+i-2) - \sum_{i=0}^{1} r_i u(k+i-1)] \qquad (5.38)$$

This completes the design of the regulator.

5.5.2.2 Details of design (Scheme II)

In this section we describe the design of the regulator based on figure 5.7. The recursive identifier is described in section 5.5.4. A block diagram of the regulator is shown in figure 5.9.

This configuration is adapted from the combined observer-controller form commonly used in the state space method of pole-placement regulators. A procedure for the design of such regulators using the transfer function approach is given by Kailath [63].

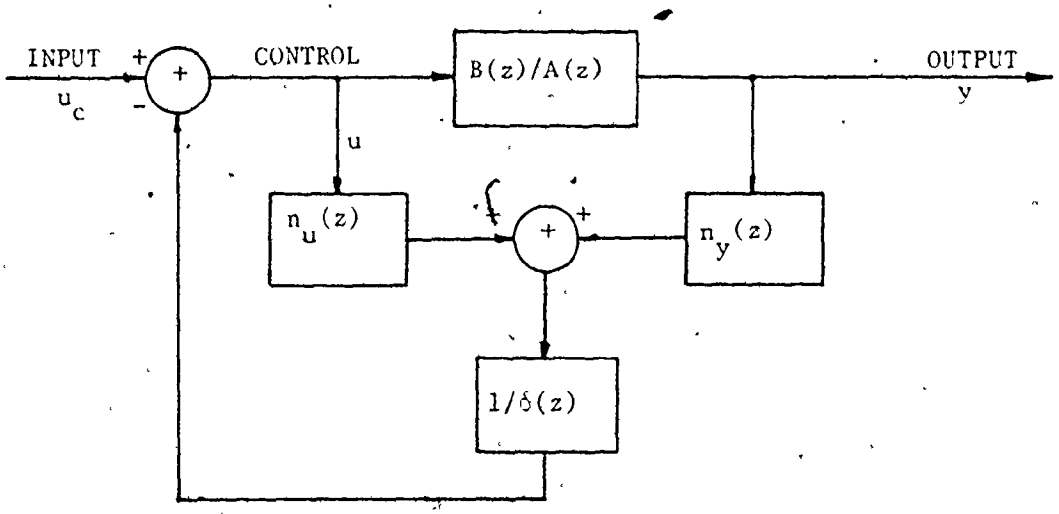


Fig. 5.9: Block diagram of regulator (II).

We have from figure 5.9

$$U_c(z) - [\frac{1}{\delta(z)} (n_u(z)U(z) + n_y(z)Y(z))] = U(z) \tag{5.39}$$

or

$$[\delta(z) + n_u(z)]\, U(z) = \delta(z)U_c(z) - n_y(z)Y(z) \tag{5.40}$$

or

$$U(z) = \frac{\delta(z)}{[\delta(z) + n_u(z)]} U_c(z) - \frac{n_y(z)}{[\delta(z) + n_u(z)]} Y(z) \tag{5.41}$$

The process transfer function relates to the control signal u and the output y as

$$U(z) \frac{B(z)}{A(z)} = Y(z)$$

The transfer function from the set point U_c to the output y is given by

$$[\delta(z) + n_u(z)] \frac{A(z)}{B(z)} Y(z) = \delta(z)U_c(z) - n_y(z)Y(z)$$

or

$$\delta(z)U_c(z) = \{[\delta(z) + n_u(z)] \frac{A(z)}{B(z)} + n_y(z)\} Y(z)$$

or

$$H_c(z) = \frac{Y(z)}{U_c(z)} = \frac{B(z)\delta(z)}{[\delta(z) + n_u(z)]A(z) + n_y(z)B(z)} \tag{5.42}$$

The desired closed loop performance is specified in terms of a model. As before

$$H_m(z) = \frac{B_m(z)}{A_m(z)} = \frac{K_m B(z)}{A_m(z)} \tag{5.43}$$

We have from equations 5.42 and 5.43

$$H_c(z) = \frac{Y(z)}{U_c(z)} = \frac{\delta(z)B(z)}{[\delta(z) + n_u(z)]A(z) + n_y(z)B(z)} = \frac{K_m B(z)}{A_m(z)} \frac{\delta(z)}{\delta(z)} \tag{5.44}$$

Where $\delta(z)$ corresponds to the observer used in the state-space method and represents the hidden modes cancelled between the input and output.

The polynomials A(z) and B(z) are assumed to be known and the polynomials $\delta(z)$ and $A_m(z)$ are specified. The remaining polynomials $n_u(z)$ and $n_y(z)$ are evaluated by solving the Diophantine equation

$$[\delta(z) + n_u(z)]A(z) + n_y(z)B(z) = \frac{1}{K_m}\delta(z)A_m(z) \qquad (5.45)$$

Relation to previous method

The control law of Eq. 5.41 may be represented by the block diagram

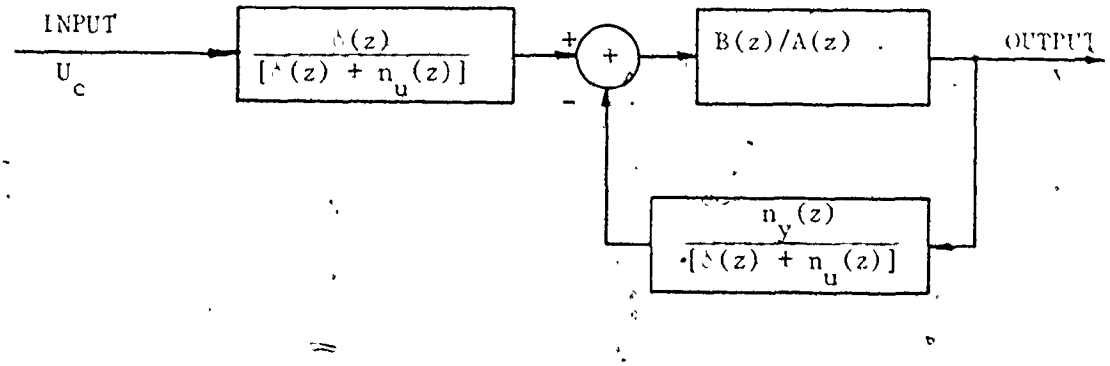


Fig. 5.10: An interpretation of the control laws.

Comparing figure 5.10 with figure 5.8, we may deduce the following:

$$A_0(z) = \delta(z) \qquad (5.46a)$$

$$T(z) = \delta(z) \qquad (5.46b)$$

$$R(z) = [\delta(z) + n_u(z)] \qquad (5.46c)$$

and $$S(z) = n_y(z) \qquad (5.46d)$$

Rewriting equations of the closed loop from section 5.5.2.1 we have

$$H_c(z) = \frac{B(z)T(z)}{A(z)R(z) + B(z)S(z)} = \frac{A_0(z)K_m B(z)}{A_0(z)A_m(z)} \qquad (5.47)$$

Substituting values of $T(z)$, $A_0(z)$, $R(z)$ and $S(z)$ from Eq. 5.46 we have,

$$\frac{\delta(z)B(z)}{[\delta(z) + n_u(z)A(z) + n_y(z)B(z)} = \frac{K_m\delta(z)B(z)}{\delta(z)A_m(z)} \qquad (5.48)$$

This equation is identical to Eq. 5.44. The main difference in the two procedures is the way gain K_m is handled. In the first case, it is taken into account in the feedforward block. The polynomial $T(z) = B_m'A_0(z)$ or $K_mA(z)$. In the second case the numerator is simply equivalent to $A_0(z)$. K_m has to be taken into account in the evaluation of $n_u(z)$ and $n_y(z)$. This is achieved by Eq. 5.45.

As shown in the following section, although the numerical values of the coefficients are different, the overall performance of the two schemes is identical.

Summary of design procedures for the regulator

1. Obtain the process model in terms of $A(z)$ and $B(z)$.
2. Specify the poles of the closed loop system in terms of $A_m(z)$ and calculate K_m.
3. From conditions listed in Eq. 5.20 and 5.21 determine the order of polynomials R,S,T and A_0 for the first scheme or δ, n_u and n_y for the second method.
4. Specify the observer polynomial $A_0(z)$ or $\delta(z)$.
5. Solve the Diophantine equation 5.35 for R and S or equation 5.45 for n_u and n_y.
6. Evaluate the control signal from Eq. 5.38 or 5.40.

When this regulator is used in a self tuning mode, the model of the process $B(z)/A(z)$ has to be identified on-line. The sequence of steps listed above have to be repeated for each sampling interval.

Recursive identification and other aspects of the self tuning regulator for the industrial robot are addressed in sections 5.5.4. Regulators described above are first tested using fixed models. Details of various cases studied are described next.

5.5.3 Testing the Regulator

Both the algorithms described above were implemented on the VAX 11/785. When used with an actual system (real life), a signal generator will be required for the input signal U_c. The output signal would be available through a suitable sensor. For the purpose of these tests, a step input was simulated. The output signal 'y' is evaluated from the control signal 'u' as follows.

We have for a general system

$$Y(z) = \frac{B(z)}{A(z)} U(z) = \frac{b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0} U(z)$$

or, by converting to the time domain and dropping the sampling interval T for convenience,

$$y(k) = \sum_{i=0}^{m} b_i U(k+i-n) - \sum_{i=0}^{n-1} a_i \, y(k+i-n) \qquad (5.49)$$

Similarly, the expression for the output of the open loop system for the same input U_c is

$$y_0(k) = \sum_{i=0}^{m} b_i U_c(k+i-n) - \sum_{i=0}^{n-1} a_i\, y_0(k+i-n) \qquad (5.50)$$

The desired performance of the closed loop system is specified by H_m. The output corresponding to the set point is

$$y_d(k) = K_m\{ \sum_{i=0}^{m} b_i U_c(k+i-n) - \sum_{i=0}^{n-1} a_i y_d(k+i-n)\} \qquad (5.51)$$

where

$$K_m = \sum_{i=0}^{n} a_i \;/\; \sum_{i=0}^{m} b_i \qquad (5.52)$$

The following cases were studied. The input to the system was chosen to be a step signal, and the desired poles were arbitrarily placed at $z = 0, 0$ and 0.7.

(1) Model of the robot at T = 0.1 secs.

(2) Model of the robot at T = 0.05 secs.

(3) Model of the robot at T = 0.025 secs.

(4) Model of the robot at T = 0.01 secs.

(5) Model of robot at T = 0.1 secs, except that its real pole is shifted to $z = 1.0$.

$\therefore\ A(z) = z^3 - 0.5826\, z^2 + 0.0415359\, z - 0.45893594$

(6) As in (1) except that the real pole is moved outside the unit circle [z = 1.05].

$$\therefore \quad A(z) = z^3 - 0.6326\, z^2 + 0.0206659\, z - 0.4818827$$

Discussion of results:

Since the performance of both structures was seen to be identical, only one set of plots is presented.

Figure 5.11 represents the uncompensated, open-loop step response of the system for cases 1-4. The 'desired' or specified performance of the closed loop is shown in figure 5.12 (a). Performance of the closed loop system in all the four cases was identical and is plotted in figure 5.12 (b). It is evident that the regulator output is close to the desired output. The control signal required to generate this is shown in figure 5.13, for all the four cases. The control signal for T = 0.05, 0.025 and 0.01 secs has an overshoot. Also it is evident that the swing of the control signal increases as the sampling time is decreased. This poses a requirement that the amplifiers used to drive the actuator must have a high dynamic range. In addition, such a swing may not be within safe operating limits.

Cases 5 and 6:

The other two cases are variations of case 1. For case 5, the real pole is shifted to the unit circle. This has the effect of including an integrator in the system. The performance of the regulator is shown by the plots in figure 5.14. As expected, the

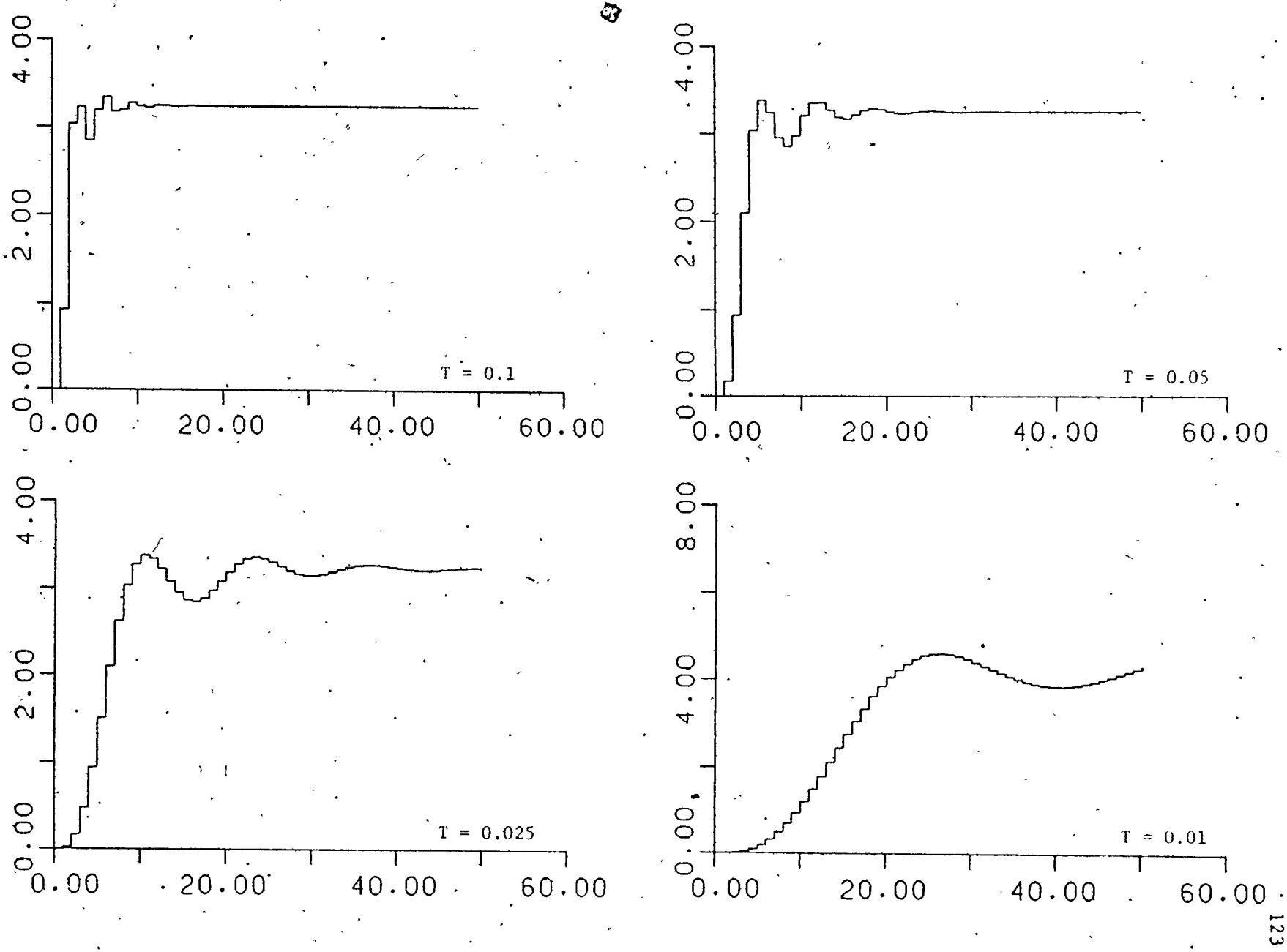
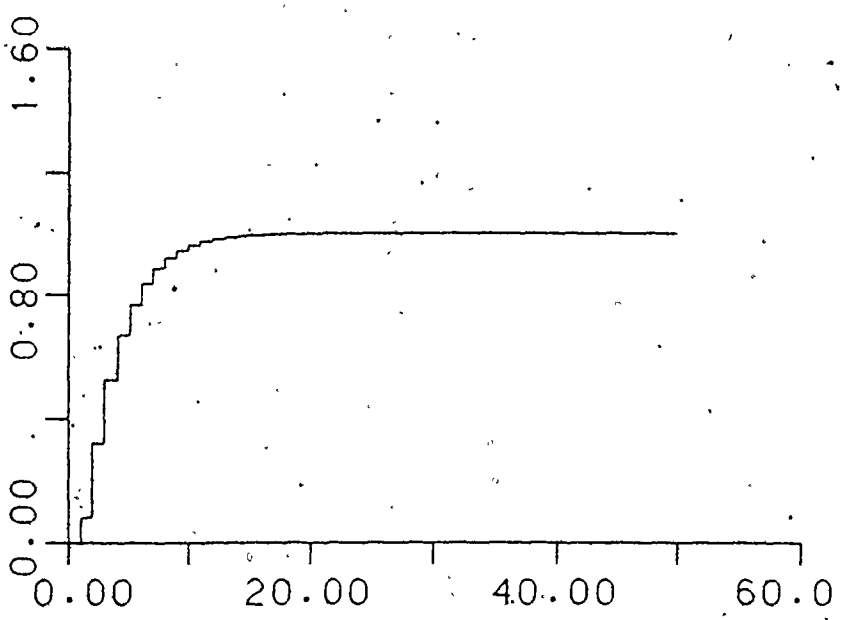
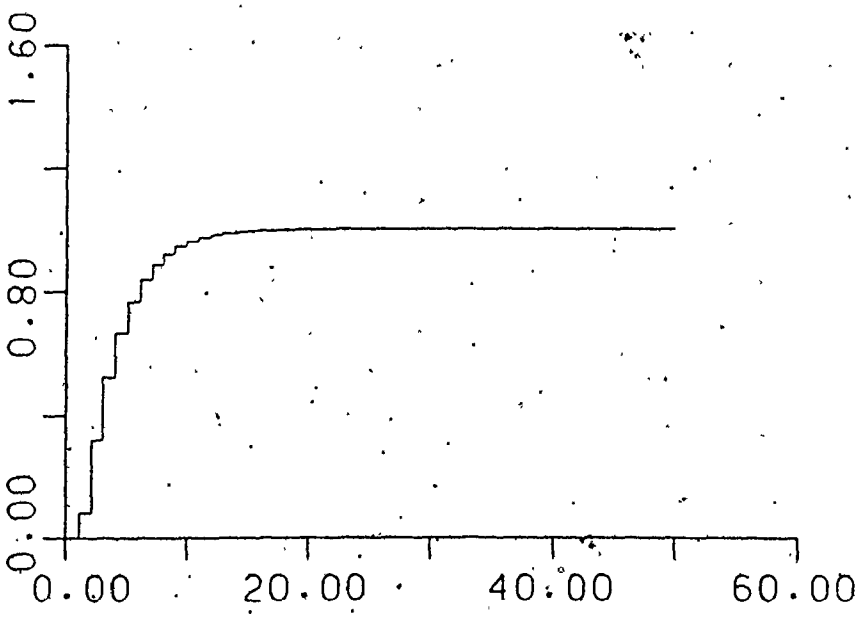


Fig. 5.11: Uncompensated outputs.



(a) Desired output



(b) Compensated output

Fig. 5.12: (a) Desired output, (b) Compensated output.

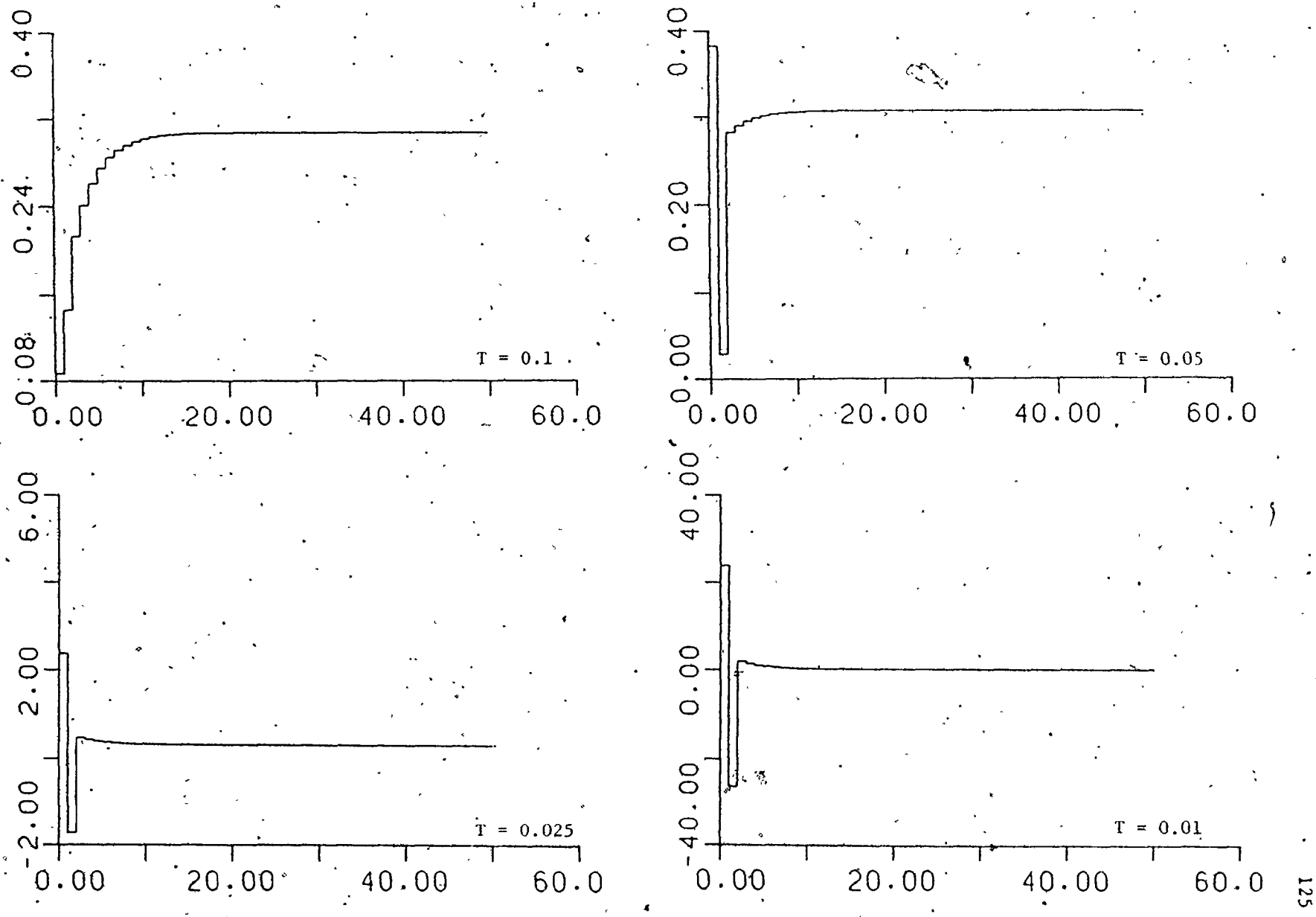
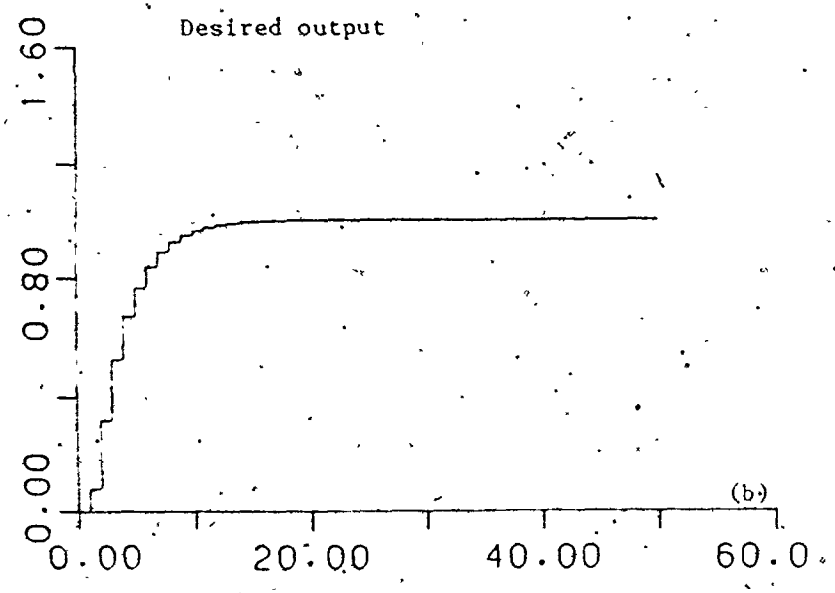
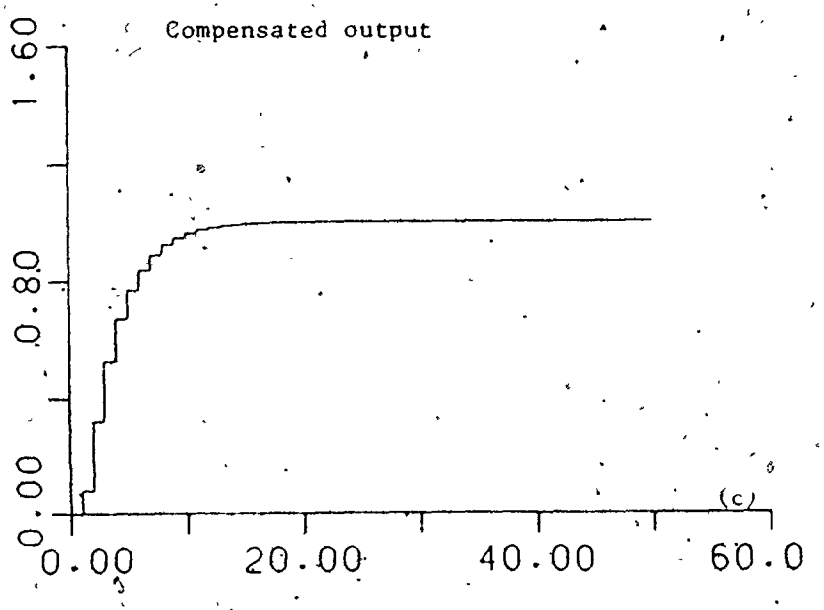
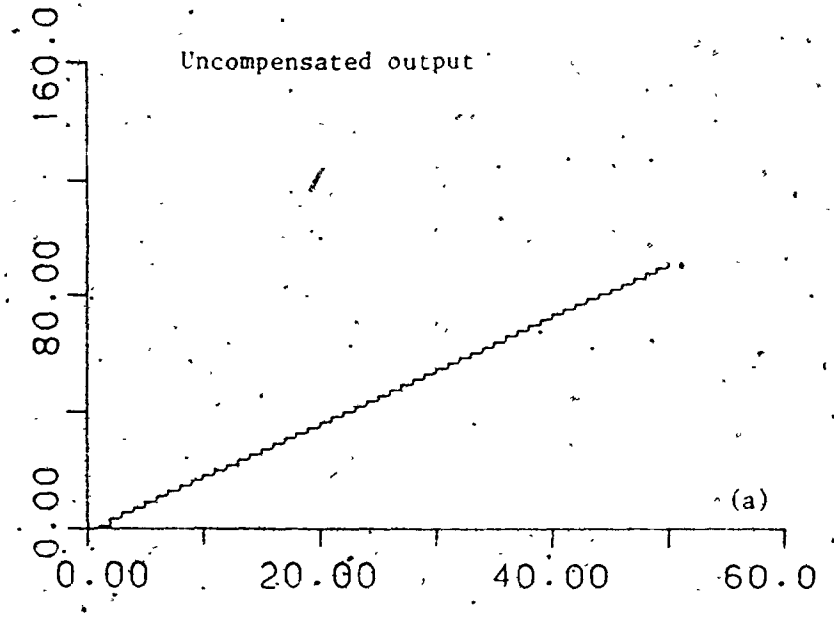


Fig. 5.13: Control signals.







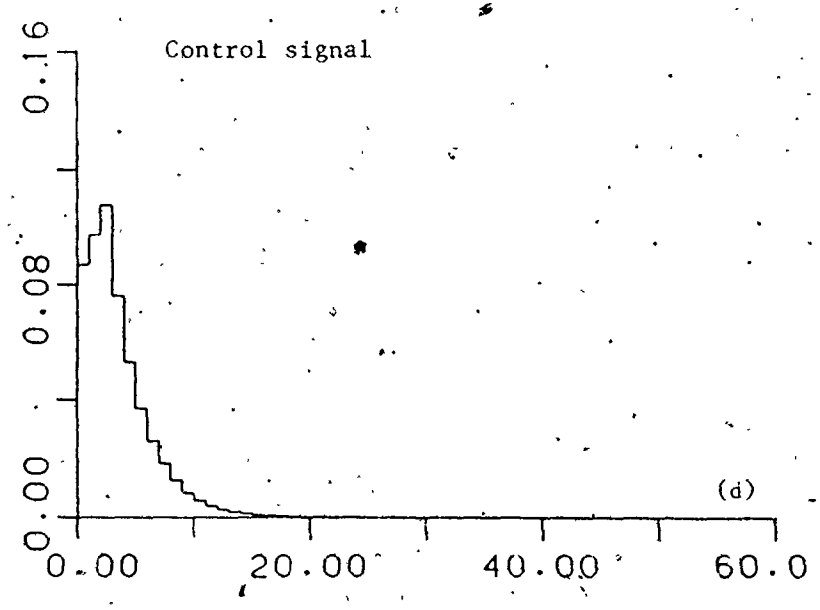


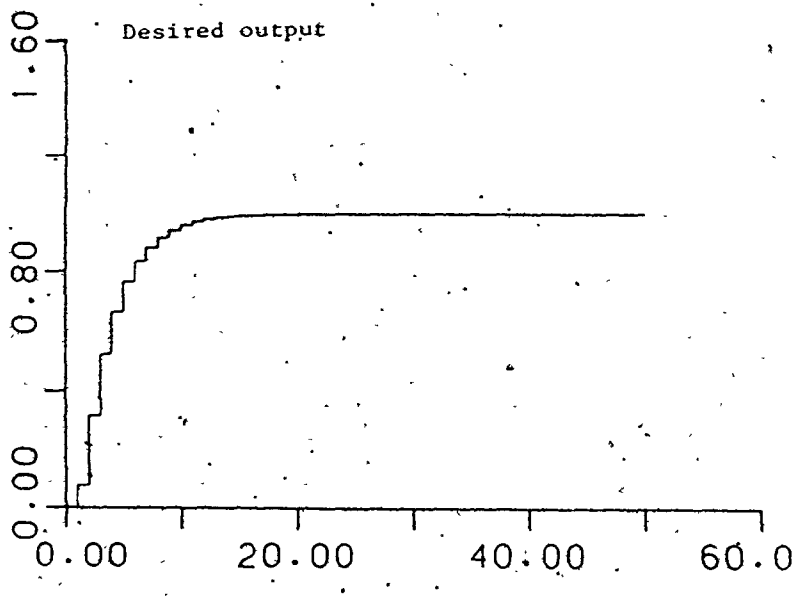
Fig. 5.14: Performance of the regulator for a system with an integrator.

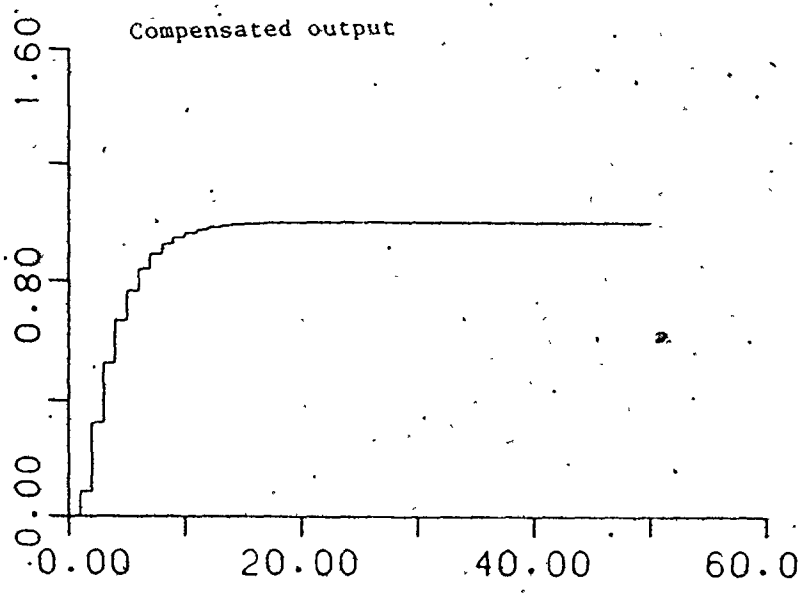
uncompensated (open-loop) output, figure 5.14 (a), increases without bound. However, the closed loop output follows the desired signal [figure 5.14(b)] closely. The control signal is plotted in figure 5.14 (d).

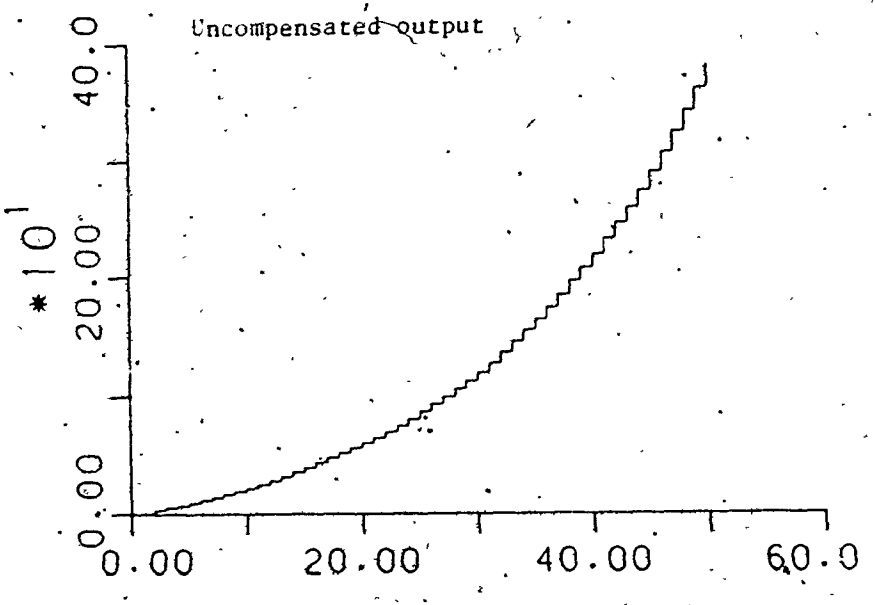
Results of case 6 are shown in figure 5.15. In this case, since one pole is outside the unit circle, the uncompensated output of the system increases monotonically as shown in figure 5.15 (a). The closed loop performance [figure 5.15 (c)] is quite close to the desired output [figure 5.15 (b)]. The control signal is plotted in figure 5.15 (d).

It may be concluded from these results that the regulator performs satisfactorily for a variety of situations. In particular, the situations considered were (i) different sampling periods, (ii) change in robot model such that a pole is on the unit circle, and (iii) change in the robot model such that a pole is outside the unit circle. These situations, although severe, may occur in practice for some orientation of the arm or in case of a component (or sub system) failure.

Table 5.2 lists the values of the coefficients of the controller for the 6 cases. Results for both methods 1 and 2 are given. While the performance of both methods was the same, it can be seen from Table 5.2 that for cases 1 to 4, as the sampling interval is reduced, the coefficients of the polynomial $S(z)$ for method 1 increase substantially. A significant variation is also observed in the coefficients of the $[\delta(z)+n_u(z)]$ polynomial of method 2. However, for this case it appears that the change in $S(z)$ polynomial is greater.







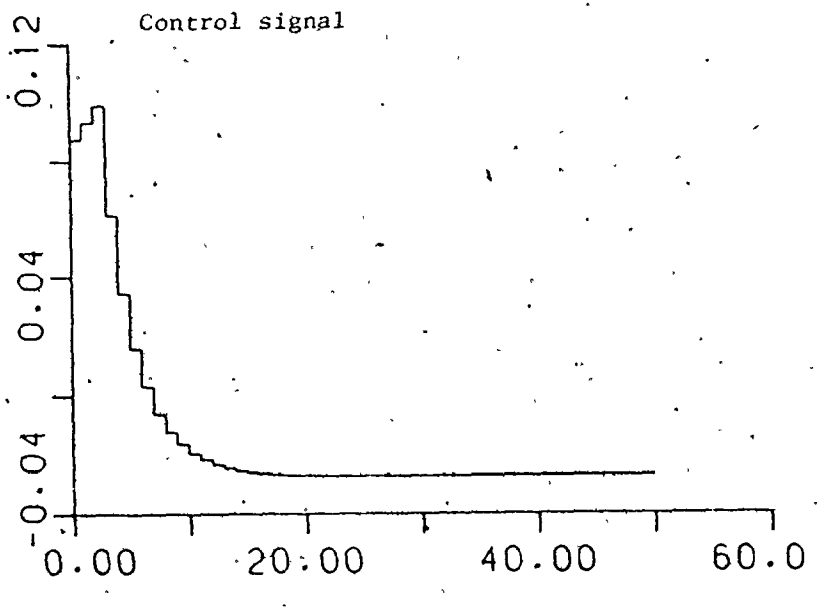


Fig. 5.15: Performance of the regulators for an unstable system.

Table 5.2: Controller parameters for different models under study.

Case #	Method	Controller Coefficients					
		Polynomial R [or $(\delta+n_u)$]			Polynomial S' [or n_y]		
1 T=0.1	1	1.0	-0.526	-0.161	-0.166	0.233	-0.0762
	2	11.474	-6.038	-1.844	-1.902	2.670	-0.875
2 T=0.05	1	1.0	0.879	0.153	0.299	-1.166	0.625
	2	2.613	2.297	0.399	0.781	-3.047	1.632
3 T=0.025	1	1.0	1.334	0.267	16.950	-24.828	9.461
	2	0.419	0.559	0.113	7.104	-10.406	3.966
4 T=0.01	1	1.0	1.545	0.334	353.290	**** Too large	193.040
	2	0.031	0.047	0.010	10.838	-15.787	5.922
5 T=0.1	1	1.0	-0.010	-0.048	-0.114	0.254	-0.053
	2	11.474	-0.112	-0.554	-1.313	2.917	-0.604
6 T=0.1	1	1.0	0.027	-0.040	-0.1	0.260	-0.046
	2	11.474	0.307	-0.462	-1.148	2.985	-0.529

Also, for cases 1, 5 and 6, where the sampling period is the same [T=0.1 secs.], the variation in the coefficients evaluated by method 1 is larger, though not by a significant amount. Another observation is that for this case, the magnitude of the coefficients evaluated by method 2 are larger than the corresponding ones by method 1. This has the advantage that any rounding (or truncation) due to the finite word length of microprocessors will result in smaller deviations from the theoretical values.

5.5.4 On-Line Parameter Estimation and Adaptive Control

Results presented in the previous section assumed that the parameters of the process were known in advance. Using these parameters for the model, algorithms for a pole placement regulator were developed which resulted in the desired closed loop performance. In this section, the regulator is modified so that it can operate in an adaptive fashion. This is achieved by identifying the model of the system, on-line, so that the regulator performance is as specified, even when there are changes in the process. The regulator is then called self tuning and can be represented as in figure 5.6 or figure 5.7.

The control signal 'u' and the process output 'y' are used in a recursive identification algorithm. In Chapter 3, we described several methods for on-line parameter estimation including the recursive least squres (RLS), recursive instrumental variables (RIV), and the recursive maximum likelihood (RML) methods. Any of these methods could be used, but due to advantages in terms of computational

complexity, the RLS was chosen.

It is assumed for the design of the self tuning regulator (STR) that though model parameters are not known (and possibly changing), the structure of the model is fixed and known a-priori.

In general, the model is represented as

$$A(z)Y(z) = B(z)U(z)$$

where

$$\frac{B(z)}{A(z)} = \frac{b_1 z^{-1} + b_2 z^{-2} + \ldots b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots a_n z^{-n}} \tag{5.53}$$

The model in 5.53 describes the dynamic relation between the control input 'u' and the output 'y'.

Define a parameter vector

$$\underline{\theta}^T = [a_1 \; a_2 \; \ldots \; a_n \; b_1 \; \ldots \; b_m] \tag{5.54}$$

and a vector of lagged input and output samples

$$\underline{\phi}^T = [-y(t-1) \; \ldots \; -y(t-n)u(t-1) \; \ldots \; u(t-m)] \tag{5.55}$$

In the time domain, the output y(t) may be expressed as

$$y(t) = \underline{\theta}^T \cdot \phi(t) + v(t) \tag{5.56}$$

where v(t) is a noise term.

As described in the previous chapter, the least squares estimate ($\underline{\hat{\theta}}$) of the parameter vector $\underline{\theta}$ can be calculated recursively

from the following:

$$\underline{\hat{\theta}}(t) = \underline{\hat{\theta}}(t-1) + \underline{L}(t)\,[y(t) - \underline{\hat{\theta}}^{T}(t-1)\,\underline{\phi}(t)] \tag{5.57a}$$

$$\underline{L}(t) = \frac{\underline{P}(t-1)\,\underline{\phi}(t)}{1/\alpha_t + \underline{\phi}^{T}(t)\,\underline{P}(t-1)\,\underline{\phi}(t)} \tag{5.57b}$$

and

$$\underline{P}(t) = \underline{P}(t-1) - \frac{\underline{P}(t-1)\,\underline{\phi}(t)\,\underline{\phi}^{T}(t)\underline{P}(t-1)}{1/\alpha_t + \underline{\phi}^{T}(t)\,\underline{P}(t-1)\,\underline{\phi}(t)} \tag{5.57c}$$

The matrix $\underline{P}$ is often called the covariance matrix and α_t is the 'weighting factor' because it is used to assign weights to data. When $\alpha_t = 1$, all samples have the same weight.

The complete algorithm for the self tuning regulator is as follows:

1. From the current values and few past samples of u_k and y_k, identify the parameter vector $\underline{\theta}$, using the RLS algorithm [Eq. 5.57].
2. Solve the Diophantine equation for the polynomials $R[\delta+n_u]$ and $S[n_y]$, as described in preceeding sections. Parameters obtained in step 1 are used instead of A and B.
3. Evaluate the next sample value of the control signal u.
4. Repeat steps 1, 2 and 3 for each sampling interval.

One of the problems, often encountered in practice with such self tuning regulators is due to the identification algorithm. For good parameter estimation, the input signal to the identifier has to be of the 'persistently exciting' type. Since the control signal is a result of feedback, there is no direct control over it. It is possible that this signal may not satisfy the 'persistently exciting' condition. This is particularly true if the reference input (set point) is held constant for a long interval and the system model remains unchanged.

Another problem is that if the forgetting factor is less than one, under some circumstances, the covariance matrix may become large, so that when a change occurs, the algorithm may result in undesirable control transients. This however, has the fortuituous consequence of enhancing the identifiability of process parameters [55].

One approach toward solving this problem is to make α_t a variable. In this case, the weighting factor was made a function of the residual error. So that, when the error is large, the value of α_t is made small, and the value of α_t is returned close to 1 if the error is small. In addition, when the residual error is large, the diagonal elements of the covariance matrix $\underline{P}$ are increased. This signifies that the confidence in the current estimates is low, and aids convergence.

5.5.5 Study of Adaptivity and Robustness

Simulation:

A computer software package has been developed in FORTRAN-77

for the VAX-11/785 computer system. It contains different modules for parameter estimation, controller design, print and plot routines, etc., in an interactive way. Some other facilities include, inputing any desired model, specifying the observer polynomial, facilities for changing plant parameters and for testing adaptivity and controlling the numerical precision of controller parameters.

Testing:

(a) Constant forgetting factor

The performance of both schemes with full computer precision (no rounding or truncation of coefficients), was found to be identical. Therefore, only one set of plants corresponding to scheme II are presented.

Two cases were considered for the evaluation of the performance of the self tuning regulator.

Case 1: The 'motion amplitude' of the robot-arm is abruptly doubled after 5 secs. of normal operation.

and

Case 2: The system abruptly becomes unstable after 10 secs. of normal operation.

The set point in both cases was varied as a pseudo random binary sequence [PRBS] of length 2^4. Results are presented for two conditions of the forgetting factor. In the first set discussed, the value of the factor was held constant at 0.99.

Case I:

The input signal is shown in figure 5.16 (a), and the desired response is plotted in figure 5.16 (b). The uncompensated (open loop) response of the robot to this input is shown in figure 5.16 (c). As expected, the output amplitude doubles at the 50th sample. The closed loop output, control signal and residuals are plotted in figure 5.17. The closed loop output seems to have recovered from the sudden change, but the deviation from the required output, even after the 500th sample, is large. Also the rate of convergence is slow. Dashed line on some plots indicate the saturation level.

Case II:

As in the previous case, a PRBS was used to change the set point. The input signal, desired response and the uncompensated (open-loop) output are shown in figure 5.18. Since the system is forced to become unstable at the 100th sample, the uncompensated output increases monotonically from that point onwards until it saturates.

The output of the self tuning regulator, as shown in figure 5.19 (a), is bounded but is not close to the desired value. It appears from the plot that the compensated output is tending towards the desired vlaue, but the rate of convergence is slow. The corresponding values of the control signal and residuals are plotted in figure 5.19 (b) and (c), respectively.

It is evident from the above simulations that although the controller responds favourably, its response is very slow and therefore unsatisfactory.

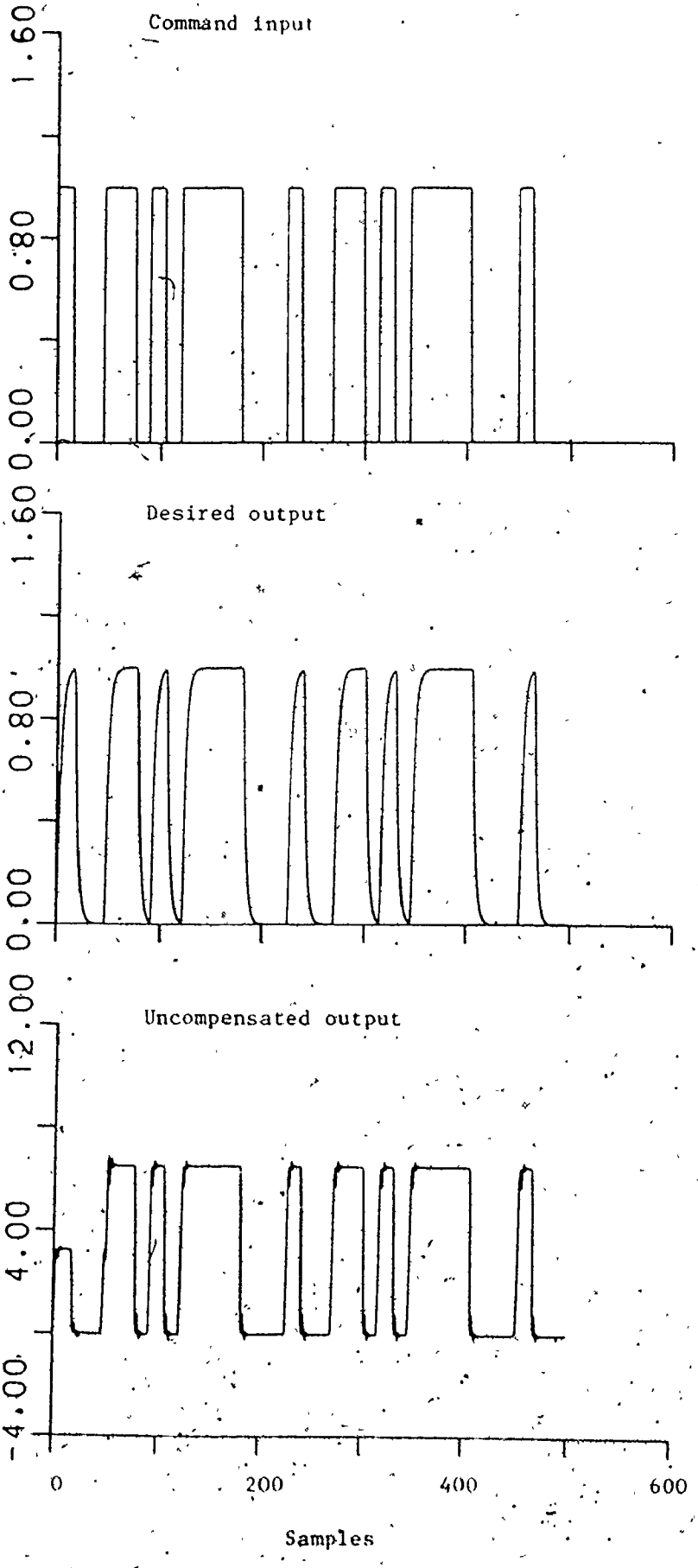


Fig. 5.16: Input-output signals for case I with constant α.

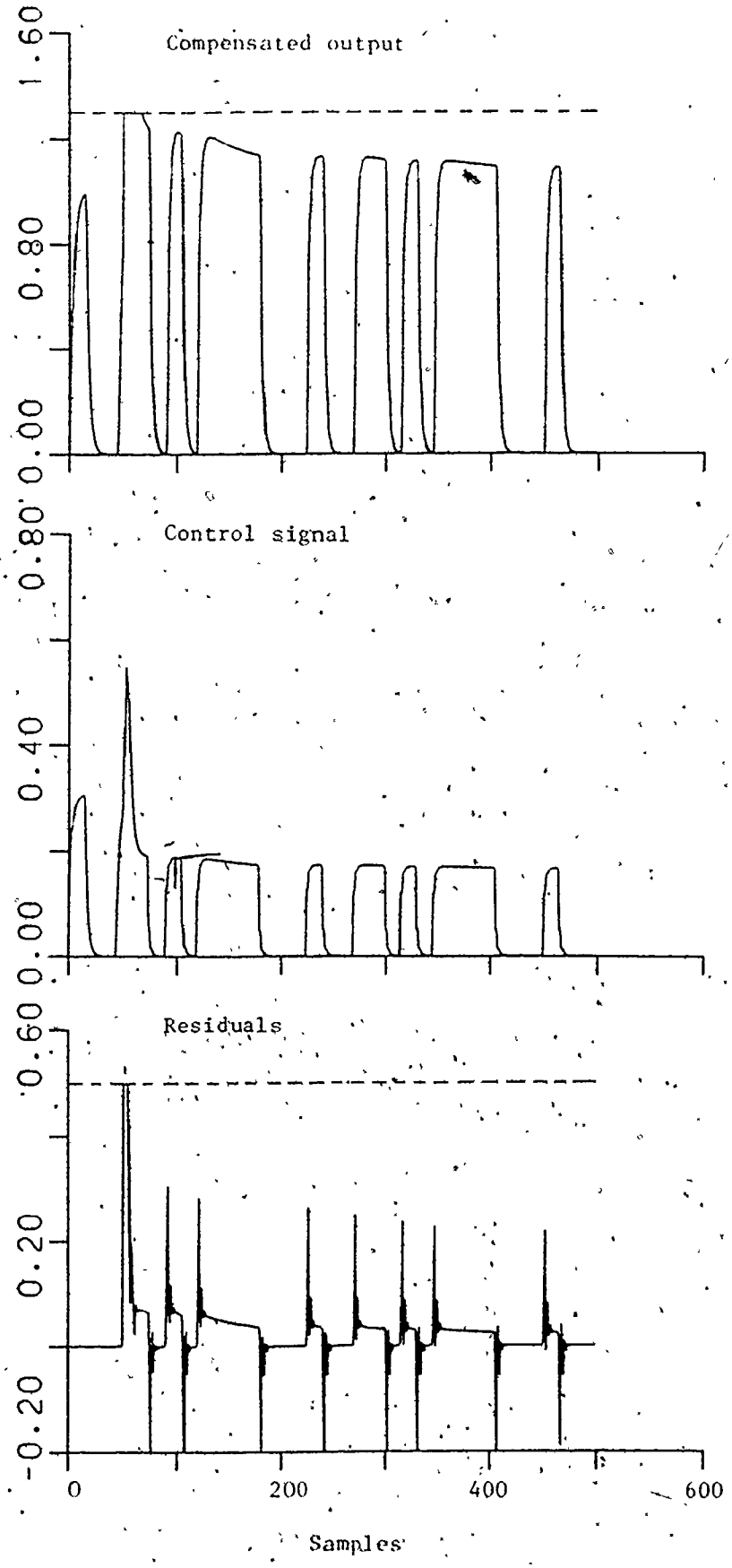


Fig. 5.17: Performance of regulator for case I with constant α.

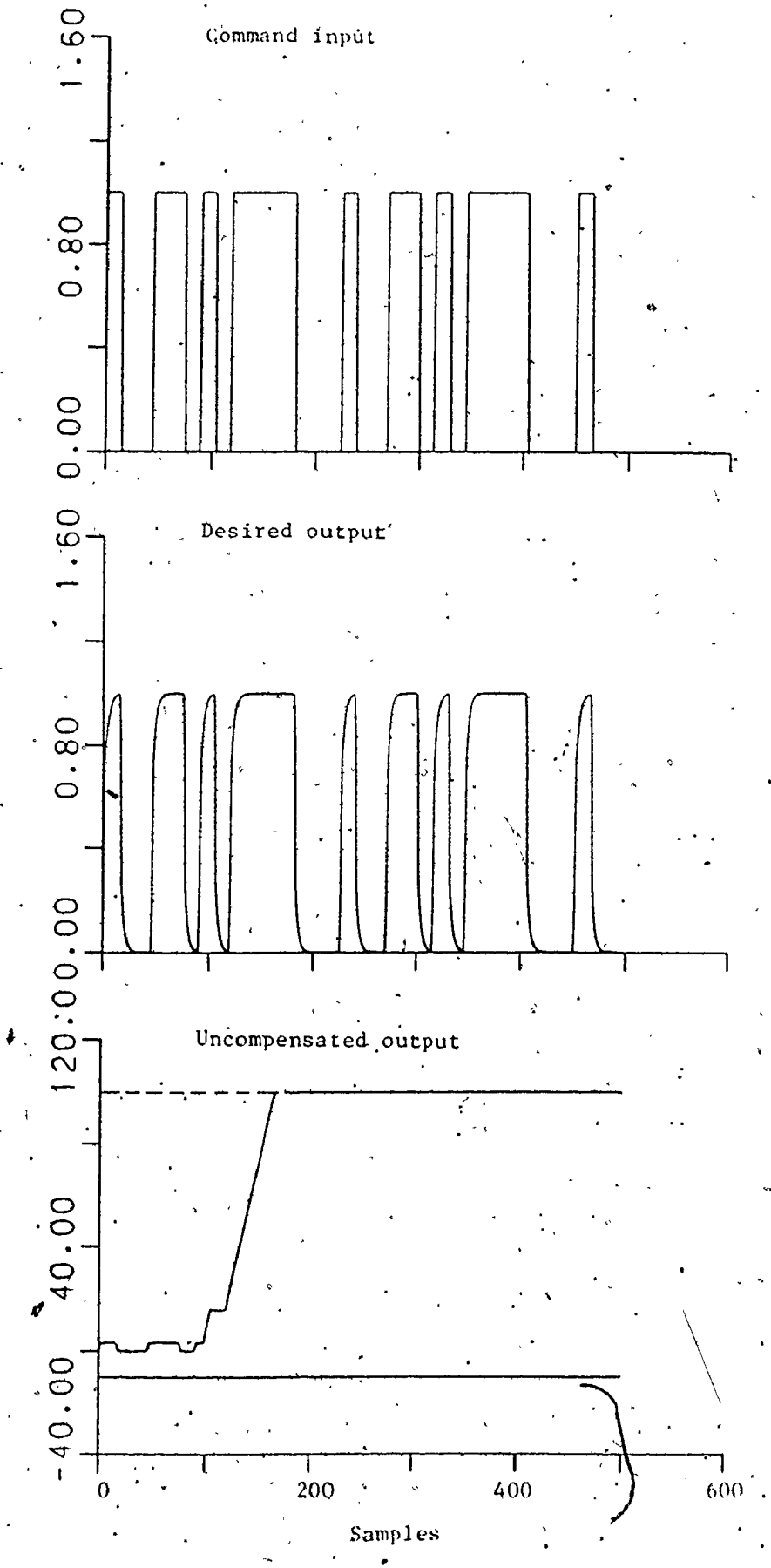


Fig. 5.18: Input-output signals for case 2 with constant α.

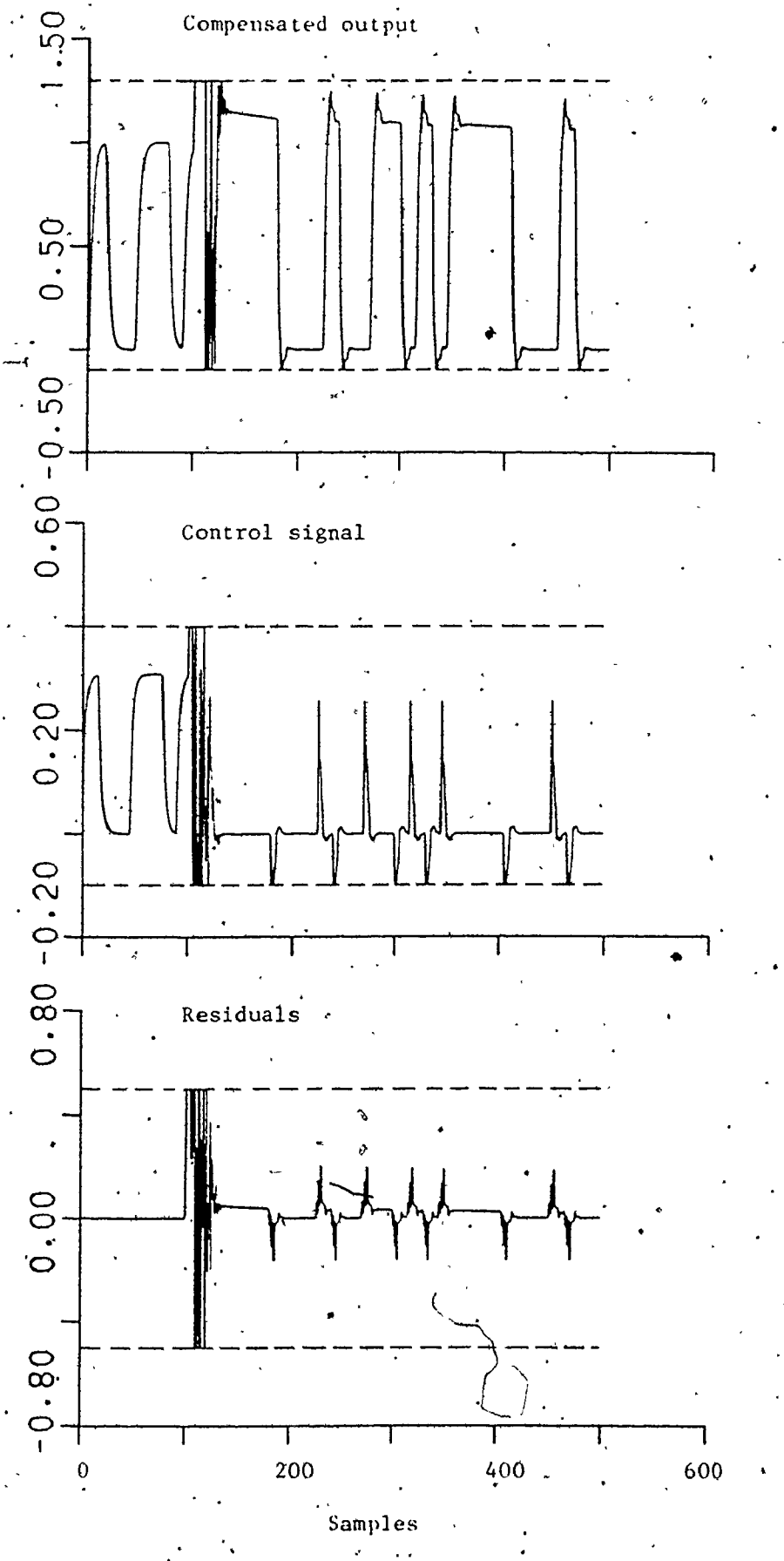


Fig. 5.19: Performance of regulator for case 2 with constant α.

Variable weighting factor:

The STR in the previous section was based on an identification scheme with a constant weighting factor. In this section we discuss the effect of making the value of the weighting factor a function of the residual error. Some strategies for varying the value of the factor have been discussed in [79,80]. However, for ease of implementation, the complexity of the function was kept to a minimum and simple binary switching was used.

The weighting factor was switched between two values 0.99 and 0.90 depending on the absolute value of the residual error. An arbitrary level of 0.1 was selected. In addition to the weighting factor, the diagonal elements of the P-matrix were reset to a large value (100.0) whenever the error exceeded this bound. Results for the same cases are discussed again.

Case 1:

The compensated (closed loop) output is shown in figure 5.20 (a). It is evident that the system recovers quickly from the abrupt change and is close to the desired output [figure 5.16 (b)]. The control signal is shown in figure 5.20 (b) and the residuals in figure 5.20 (c). The plot shows that except where the system was abruptly changed, residual errors are small.

Case 2:

The compensated output is shown in figure 5.21 (a). The control signal and the residuals are plotted in figure 5.21 (b) and

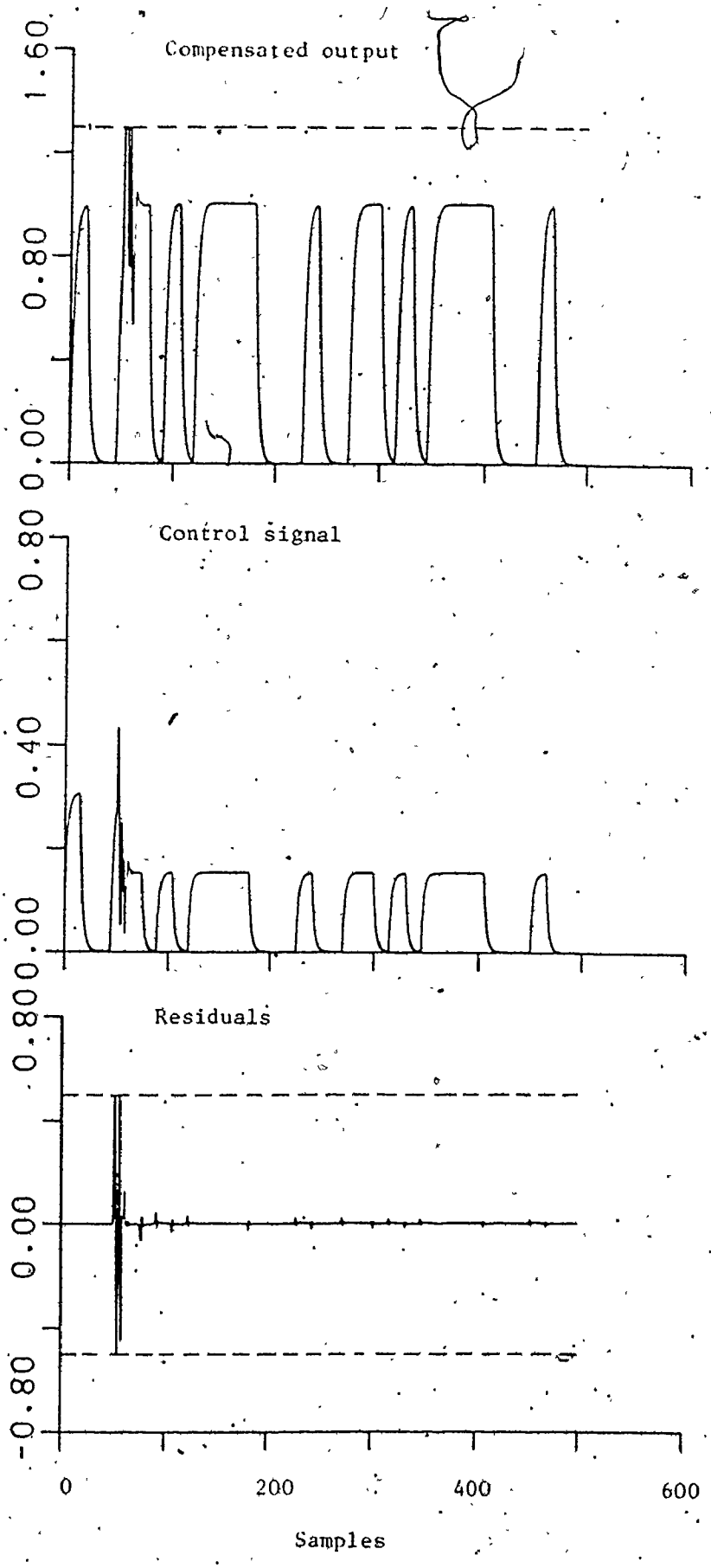


Fig. 5.20: Performance of regulator for case 1 with variable α.

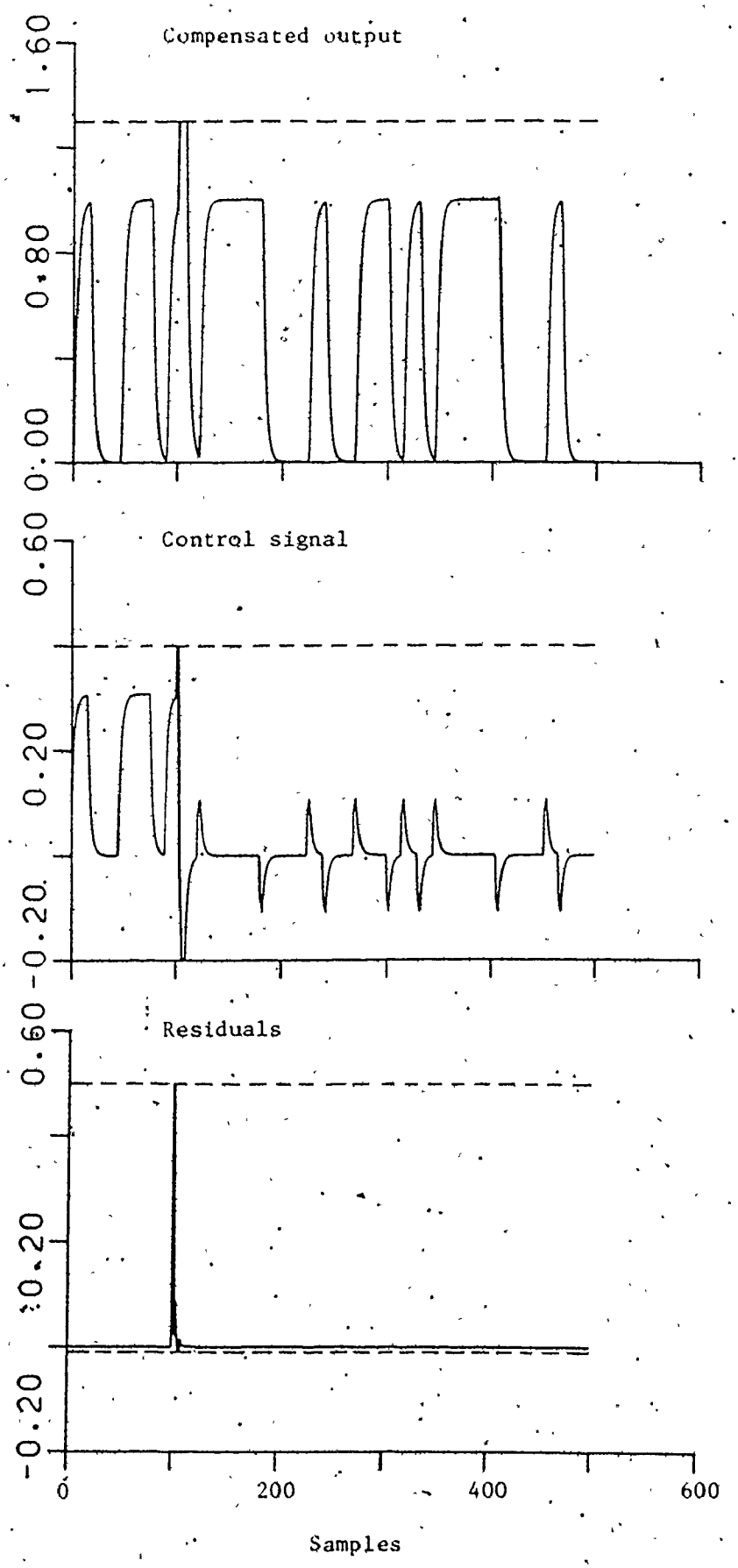


Fig. 5.21: Performance of regulator for case 2 with variable α.

5.21 (c), respectively. It is clear from the plot of the compensated output that the controller action is fast and the output follows the desired signal closely. As in the previous case, the residual error is low except when the system changes abruptly.

Effect of truncation of regulator coefficients

This section presents results of a study conducted to verify the robustness of the self tuning regulator against numerical errors. Case 2 discussed above was simulated for both schemes. A routine was included to truncate the value of regulator parameters to the number of decimal places specified by the user. Figure 5.22 (a) shows the compensated output of the plant with the regulator designed according to scheme I, and figure 5.22 (b) shows the output corresponding to the design according to the second method. Regulator parameters have been truncated to an accuracy of 3 decimal places. Very little difference is noticable. However, as shown in figure 5.23 (a) and figure 5.23 (b), when the accuracy was cut down to 2 decimal places, the difference is appreciable.

The output corresponding to scheme 1 [figure 5.23(a)] is not as close to the desired output as that shown in figure 5.23 (b). This may be attributed to the low values of the coefficients for the first method. Therefore, one may conclude that for the self tuning regulator for the industrial robot with a sampling time of T = 0.1 secs., the second method is more robust. However, as shown earlier, since the computational complexity of the two methods is identical, and due to the similarity in the design procedure, it is easy to modify from one to the other.

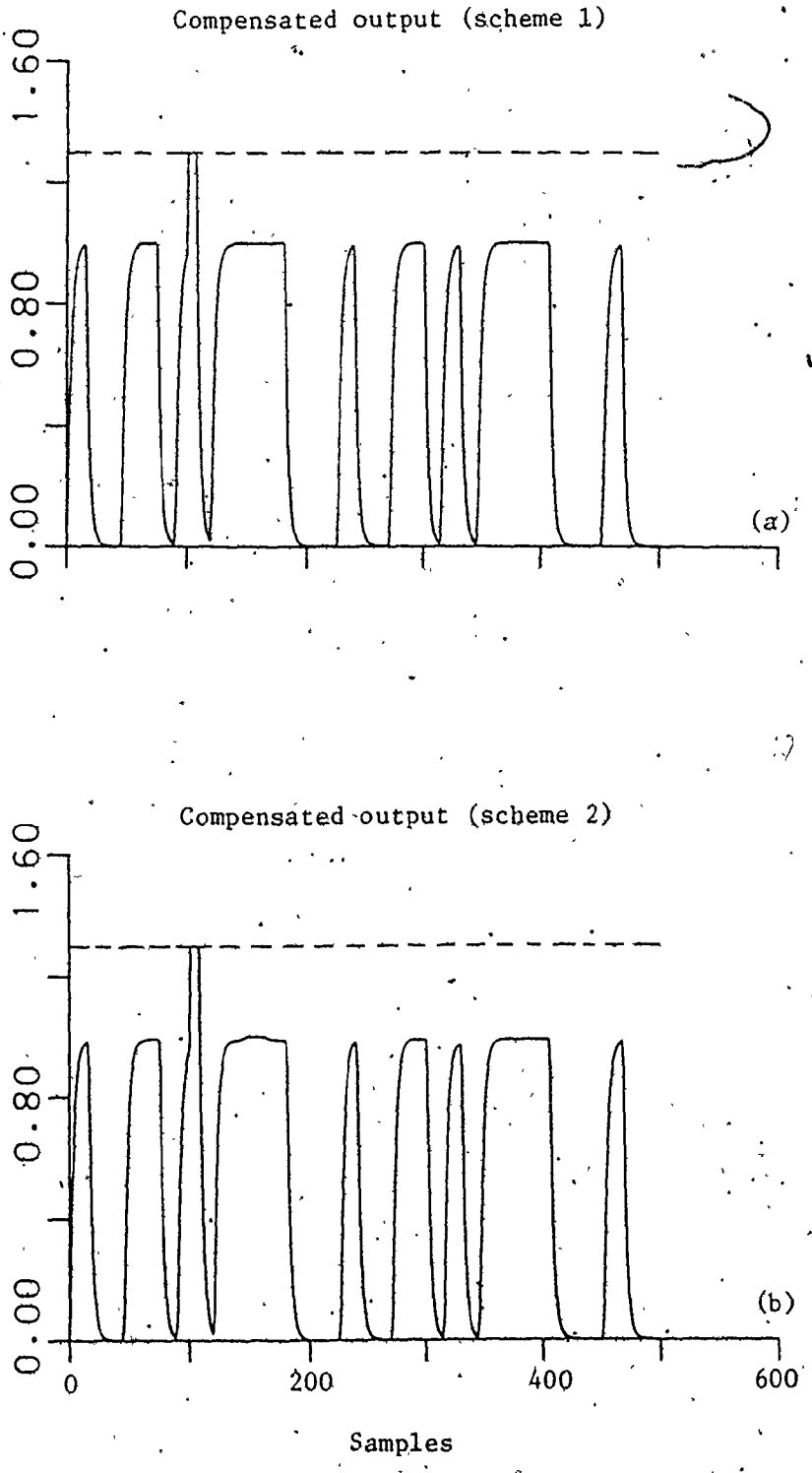


Fig. 5.22: Effect of truncating controller parameters to 3 decimal places.

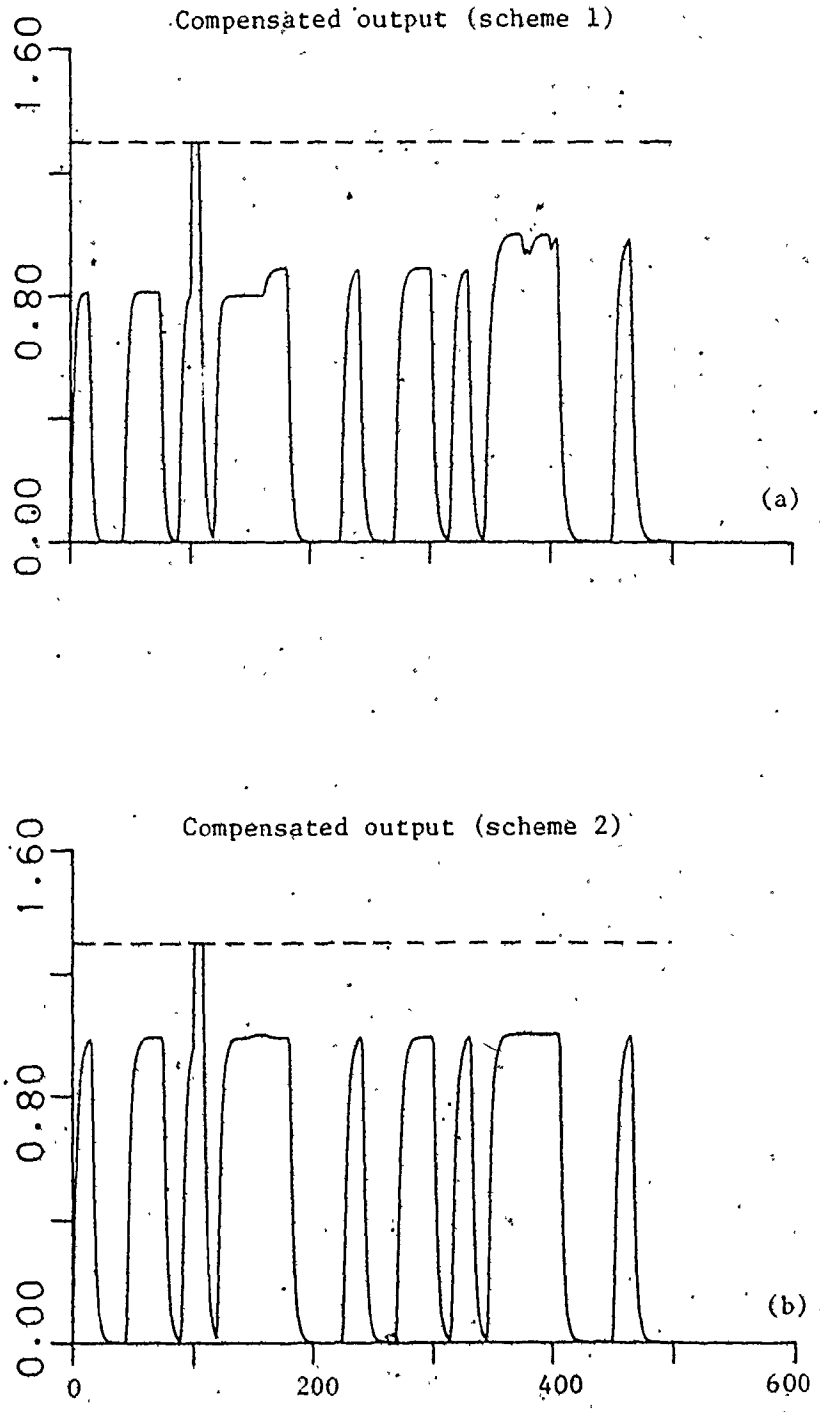


Fig. 5.23: Effect of truncating controller parameters to 2 decimal places.

5.6 Concluding Remarks

In this chapter, after a brief review of the methods of adaptive control, we discuss the details of design, testing and performance of the self tuning regulator for the industrial robot. Two configurations are discussed. The first is general in the sense that it can be used for both pole and zero cancellation. The second method, originally designed using state-space methods, is suitable only for pole placement regulators.

The model for the industrial robot is seen to be non-minimum phase. To avoid on-line factorization, only the pole-placement type of a regulator, in which all system zeros are retained, is used. In both design procedures, controller parameters are evaluated by solving a Diophantine equation. The two design procedures are identical except for a few minor variations. On-line parameter estimation is done using the recursive least squares technique because it is computationally simple.

Results from several tests are included to show the performance in terms of adaptivity and robustness of the regulator. As expected, when full computer precision is used, both methods result in identical performance. It is shown through simulations that the performance of the regulator is significantly improved by using a variable weighting factor instead of a constant one. A binary switching of the 'weighting factor' and the covariance matrix, depending on the residual error, is very effective in improving the speed and performance of the self tuning regulator.

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

In Chapter 3 we described details of modeling one axis of the robot. This model was of the continuous time type. A discrete-time equivalent was used for the design of a self tuning type of adaptive controller. Since the model of the robot varies significantly with respect to load and orientation, a 'weighting factor' was introduced in the identification part of the STR. To eliminate problems of matrix ill conditioning caused by a constant value (<1) of the weighting factor, it was made variable. Simulations described in Chapter 5 show that satisfactory results are obtained, even when the model of the robot axis was changed appreciably.

In this chapter, we give guidelines for the implementation of STR developed in the previous chapter. The most important part of implementation is the choice of components of the system. This is particularly true in the case of microprocessor based real time applications, because of the great variety of processors and support chips available. In the case of microprocessors the choice is not only governed by its technical suitability, but also from economic considerations of the availability of development aids and software.

This chapter is organized as follows. In section 6.2 we give a general overview of the controller and identify the type of components required for the implementation. A 'layered' structure has been proposed for the controller. The next section deals with the selection of the microprocessor for the controller. A comprehensive study of a variety of microprocessors was made which included both technical and other considerations. From the numerical precision point of view, it was decided that a 16-bit processor would be the most appropriate. A comparative study of various 16-bit processors available is presented.

Selection of the memory elements (RAM,ROM) and peripheral chips such as tri-state buffers, latches, etc. is less critical, because they do not affect either the development time or the cost of the system significantly. Often, the selection is based on recommendations of the manufacturer of the microprocessor.

An approach, similar to that for the selection of the microprocessor, was used for choosing the Analog-to-digital (A/D) and digital-to-analog (D/A) converters. As in the case of microprocessors, D/A and A/D converters are available in different bit lengths (resolution), conversion times, and are based on different methods of fabrication. Suggestions and guidelines for the selection of A/D and D/A converters are included in section 6.4.

Position encoders are required to measure the motion of the actuator. This information is used to verify that the motor has rotated as commanded. Shaft position encoders can be classified into two catagories - relative and absolute. Though most commercially

available robots use relative (often called incremental) encoders, absolute encoders are also used in practice. A review of both types of encoders is given in section 6.5.

For all programmed systems, the operation of the system is controlled by a sequence of instructions called the program. In the past, programs for real time operation were often written in machine or assembly language code. This approach has the advantage that the resulting software is efficient, in terms of both the storage required and execution time. The major disadvantage is that developing software in this way is difficult and time consuming. In the last few years, several 'high level' languages have been developed for writing efficient microprocessor programs. Several 'high level' languages are reviewed in section 6.6 and recommendation on the language to be used for developing the self tuning regulator has been made.

6.2 General Overview

In general, a robot control system is required to perform the following tasks.

1. Accept commands from the operator through video terminals or other input devices.

2. Execute the command by transmitting appropriate signals to actuators.

and

3. Ensure that the command has been properly executed by measurements of data from transducers monitoring the process.

In addition to the above, it usually incorporates the analysis of data, display of messages, outputting performance data to printers, monitoring alarm conditions, etc. A schematic drawing of a typical computer controlled system is shown in figure 6.1.

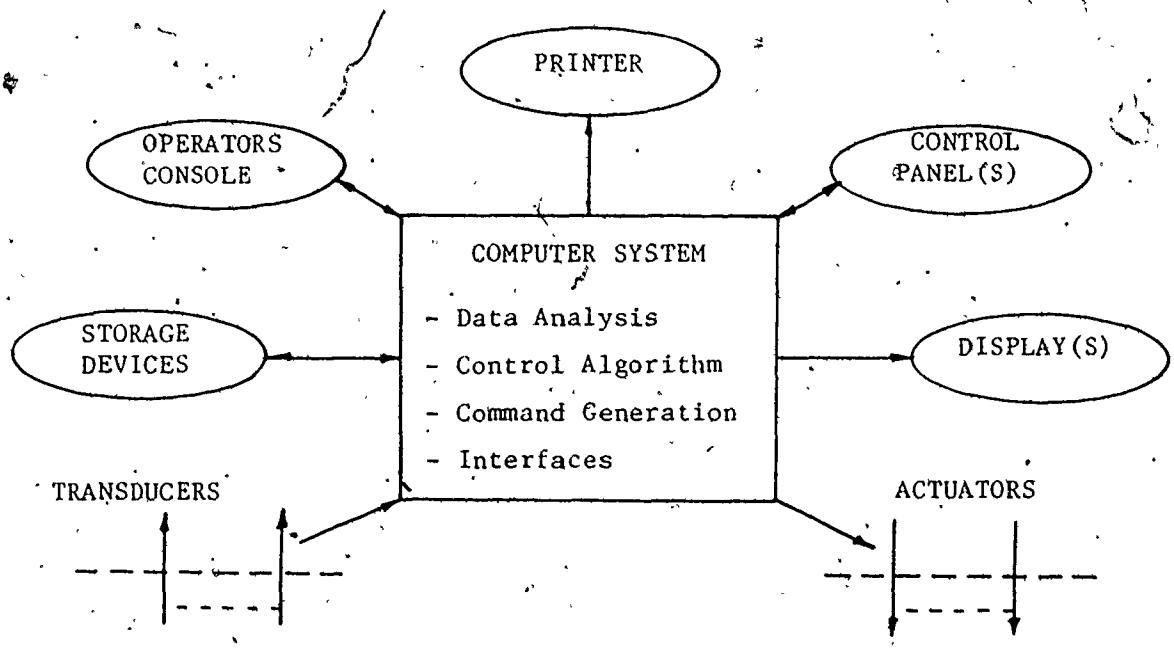


Fig. 6.1: Block diagram of computer controlled system.

The system shown in figure 6.1 is general and represents the concept of the controller. It implicitly assumes that one computer is used to perform all tasks listed. Rapid developments in solid state technology and more specifically, the availability of cheap, reliable

microprocessors has changed the concept of one processor-per-system to that of distributed processing.

In a distributed processing system, the task is shared by several processors. A block diagram of a system based on distributed processing is shown in figure 6.2.

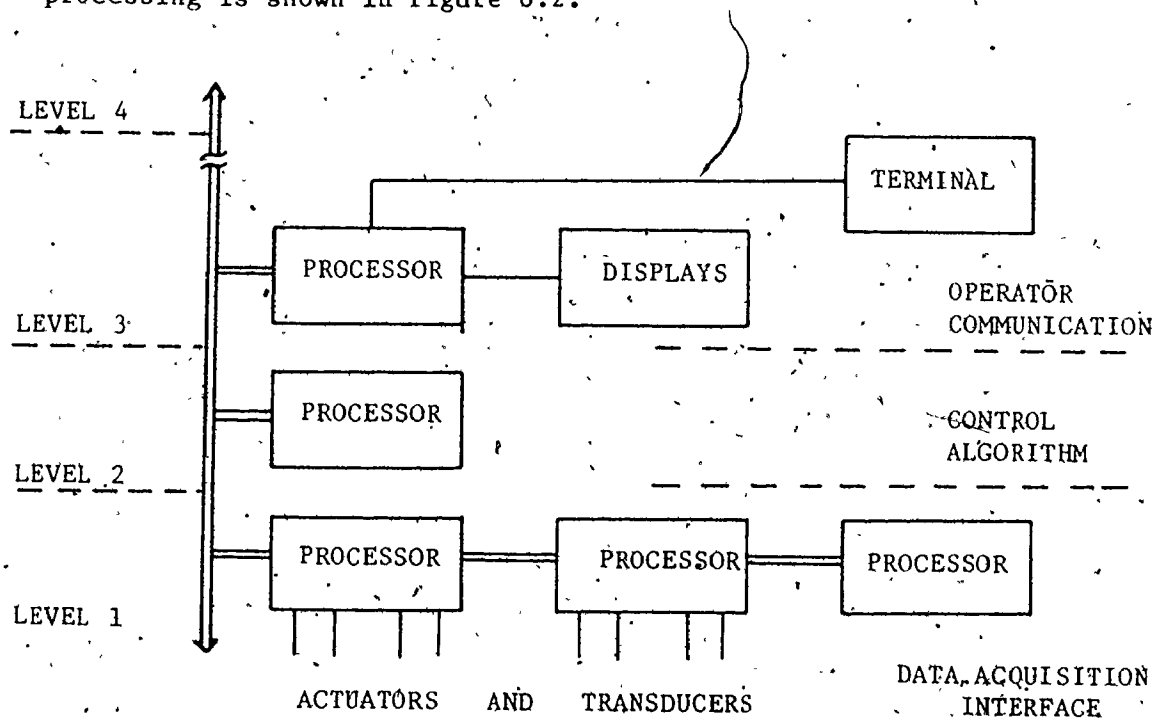


Fig. 6.2: Block diagram of distributed processing.

Boundaries between levels are not rigid and depending on system requirement, often two or more levels are merged. Some of the advantages of distributed processing are:

1. The system capability is greatly enhanced because the task is shared between processors.

2. The system is much more flexible than a single processor based system. It is also more modular and has greater expandibility. Additional functions can be added in terms of both the hardware and software.

3. Because of modularity, it is easier to identify a malfunctioning subsystem. Also the mean time to repair (MTTR) is significantly reduced.

4. In systems such as the industrial robot, in which there are several similar actuators to be controlled, it is possible to standardize modules. This leads to lower production cost due to advantages of volume production.

For computer based numerical control systems and industrial robots, it is common to use a 2-level hierarchy as described in Chapter 2. A block diagram of a 2-level control system for an industrial robot is shown in figure 6.3.

As mentioned before, this configuration is typical for an industrial robot and has been used for the constant feedback controller described in Chapter 2. The master controller (or level 1) was based on the Intel 8086 and 8087 microprocessors and the slaves (level 2) were built around the Intel 8748 microprocessor. This configuration is also frequently used in commercially available systems. For example, Unimation Inc., use the LSI-11/02 microcomputer for level 1 and six 6503 microprocessors for level 2 for the PUMA

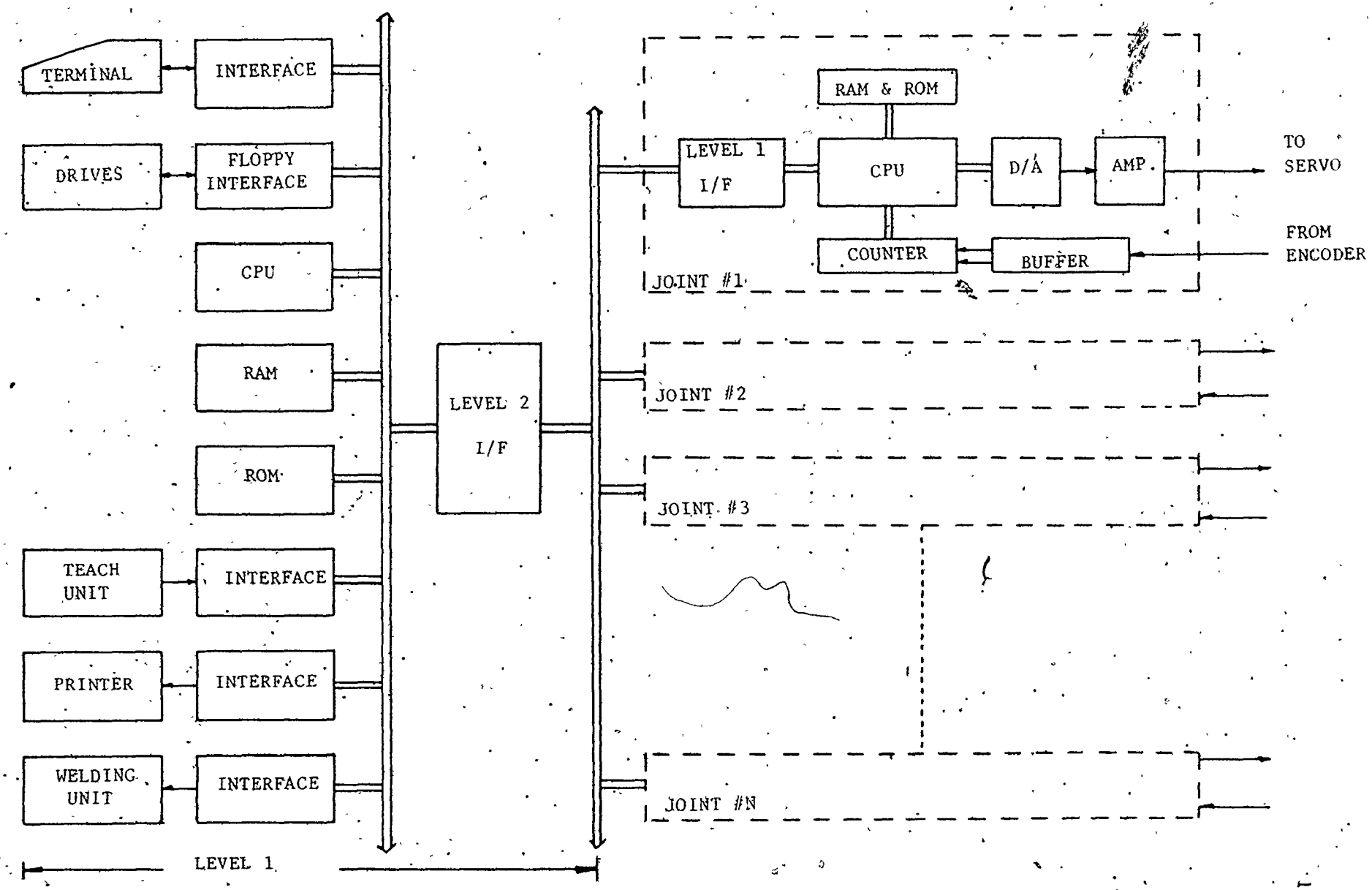


Fig. 6.3: Block diagram of control system for industrial robot.

manipulator. Cincinnati-Milacron have based their level 2 controller on the Intel 16-bit 8086 microprocessor.

The task of robot control is divided between the two levels (figure 6.3). Level 1 performs the following major tasks [16,17].

1. On-line user interface.

2. Transformation of the end point coordinates from the world coordinate system to the joint coordinate system [2,4-10].

3. Joint interpolated trajectory planning - this involves sending incremental location updates corresponding to set point values to level 2.

4. Confirming from level 2 that the axis has completed the required incremental motion.

At the lower level, main functions of the microprocessor include:

1. Receive and acknowledge set points from level 1.

2. Evaluate the difference between the current joint value and the desired joint value. The current joint value is obtained from the register which stores incremental values from the encoder mounted on each axis of rotation.

3. Convert the error to a voltage signal using a D/A converter and send the voltage to the analog servo which moves the joint.

As pointed out earlier, the main disadvantage of this control scheme is that feedback gains are constant and prespecified. It does not account for the flexibility of the arm of the robot and error in the coupling because the position encoder is mounted on the motor shaft. This error was described in detail in Chapter 3. One solution to the problem, suggested in Chapter 3 was to use an end point feedback in an adaptive loop. Details of the design of the adaptive controller are given in Chapter 5. Essentially the same set-up is required for implementing the adaptive controller. An additional A/D converter is required to monitor the position of the free end. Sensors for end point monitoring were reviewed in section 3.3.

A block diagram of one level 2 controller is shown in figure 6.4.

Comparing figure 6.4 with that of a level 2 controller shown in figure 6.3, it is evident that the only difference is in the addition of an extra feedback signal. It may be noted that the length of the program on the level 2 processor will be much larger.

One advantage of this configuration is that the adaptive loop may be switched off if a suitable sensor for end point feedback is not available or is inoperative. The resulting system will then become a constant feedback controller of the type available commercially. Since the controller is no longer adaptive, and does not have feedback to include variations due to load or elasticity of the arm, the system may have to operate at a lower speed.

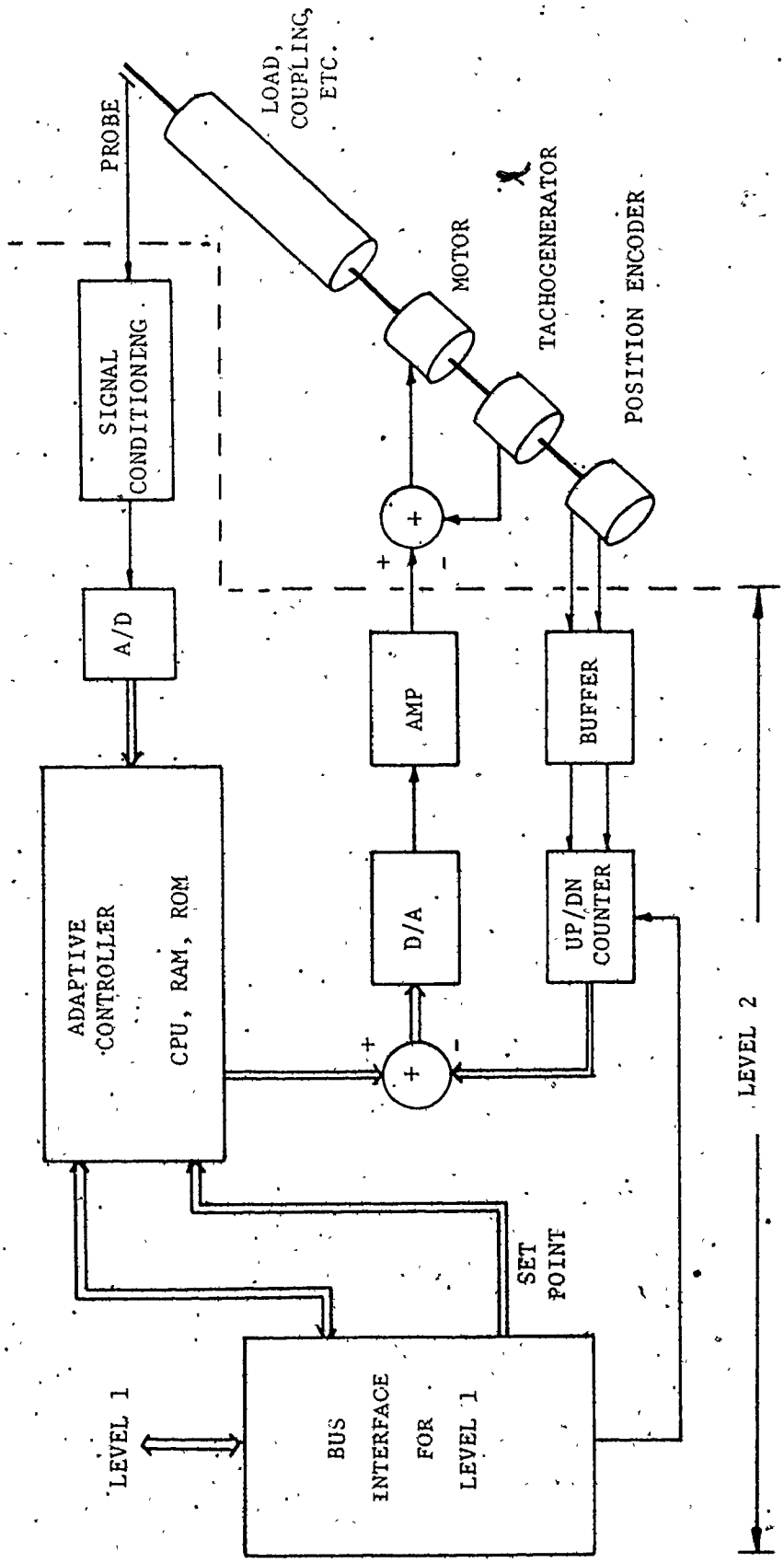


Fig. 6.4: Block diagram for adaptive controller implementation.

Switching between the adaptive and non-adaptive modes may be done manually by the operator or by a 'watch dog timer' operating under a supervisory control loop.

The algorithm for the self tuning regulator was discussed in Chapter 5. As shown in section 6.6, implementing it on the level 2 processor is straightforward because of the availability of efficient high level languages.

The next three sections deal with the choice of the microprocessor, D/A and A/D converters and shaft position feedback encoders.

6.3 Choice of Microprocessor [88-90]

In this section we discuss the criteria for selecting processors for the controller.

Selection of level 1 processor:

Popular choices for the level 1 processor or the master controller have been the LSI-11 microcomputer from Digital Equipment Corporation or a single board system based on the Intel 8086 and 8087 processors. The choice of a system for the master controller is governed by two main factors: (1) technical feasibility, and (2) economic considerations. Technically, any system capable of high speed calculations with 16 or more bits of precision is adequate. However, an important factor to be considered is the availability of software and hardware development aids.

The LSI-11 series has an extensive support and compilers for all popular computer languages are available. Also the LSI-11 system is self contained in terms of storage media, terminals, etc. In the case of the single board systems, this facility is not available, and one has to invest in the purchase of a development system. Software is developed on this system, which is usually a full fledged computer system, and then transported to the single board computer (SBC). One of the advantages of choosing microprocessors for both level 1 and level 2 from the same manufacturer, is that a common development system can be used.

Until recently, Intel development systems were based on the 8-bit 8085 microprocessor. Although they could be used to develop software for other microprocessors including 16-bit 8086 and the 64-bit 8087 numeric processors, they were too slow to be used directly as the level 1 system. Newer models such as the Series III and Series IV are available with 8086/8087 options [92]. It may therefore be possible to use these development systems more effectively by using the system directly as the level 1 processor, and also for developing software for the level 2 processor, if an Intel microprocessor is selected.

Bench mark test runs will have to be carried out to test for the speed of operation of the system to be used.

Selection of level 2 processor:

Major functions of the level 2 controller when operating in the adaptive mode are:

1. Communicating with the level 1 processor.

2. Acquiring the position of the free end from the A/D converter.

3. Maintaining a closed inner loop using feedback from the position encoder.

4. Performing ON-LINE identification of the process.

5. Solving the Diophantine equation for the controller parameters.

6. Evaluating the value of control signal from parameter obtained in step 5 above.

7. Conversion of the above to an analog voltage for driving the actuator.

A general block diagram of the level 2 controller, at a conceptual level was shown in figure 6.4. A diagram identifying various components of the controller is shown in figure 6.5. The serial interface is optional and may be used only during the development phase.

Selection of the microprocessor and its peripheral chips is based on several factors, some of which are discussed later in this section. Perhaps the most logical way to classify microprocessors is on the basis of the number of bits and speed but because of the very large number of variations in their capabilities, it is not possible to classify them in a simple way. Some manufacturers classify their products in the following four catagories, although in practice, they

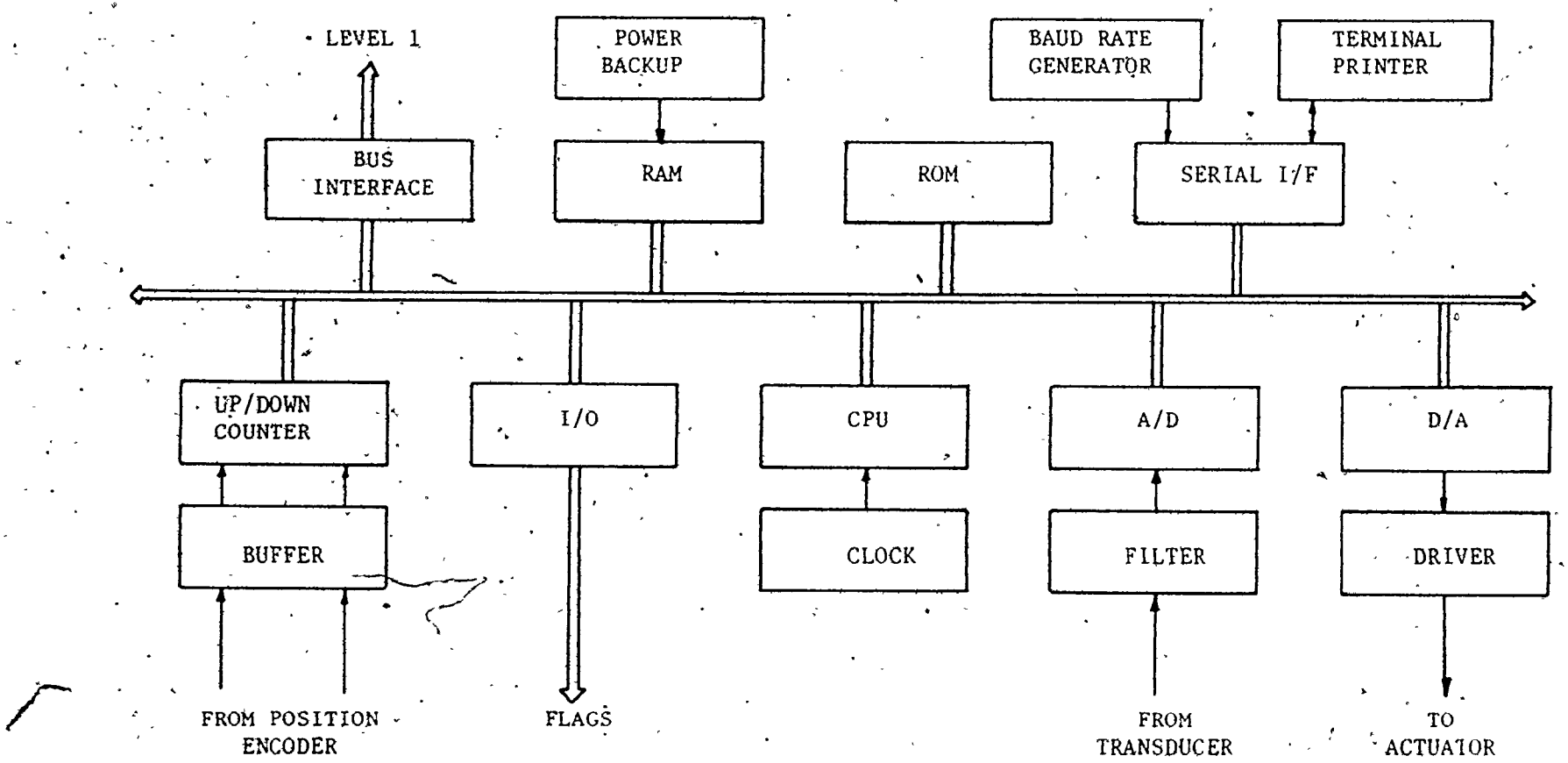


Fig. 6.5: Components of level 2 controller.

are all commonly referred to as 'microprocessors'. The catagories are:

1. microprocessors
2. microcomputers
3. microcontrollers

and 4. signal processors.

'Microprocessors' were the first to be developed and basically consist of only the control processing unit (CPU), capable of performing arithmetic and logical data manipulation. Signal processors are special CPU's, designed to operate at very high speeds. Common applications for 'signal processors' are speech, image, radar and other audio-video applications. Several CPU's have been announced in the past two to three years. Some of them are the NEC-μPD 7720, TI-TMS320, Fijitsu-MB8764 and the AMI-528211. The TMS320 and the μPD 7720 are more popular because of the availability of extensive development support. In spite of being high speed CPU's, signal processors are not commonly used for motion control because their instruction set and memory size is limited. The TMS320, for example, can address a maximum program memory of 4K, which may often be inadequate for adaptive control.

The term 'microcomputer' is used for chips which include some program and data memory in addition to the CPU. The main advantage of these integrated circuits (I.C.) is that because they are self contained, they require few additional components.

'Microcontroller' usually refers to chips with a higher level of integration. In general, they include input, output lines, timers,

etc., in addition to the CPU, RAM and ROM. Since most of the circuits required to implement (simple) controllers are included on the chip they are more convenient for several applications.

The main advantage of using highly integrated circuits is small size, fewer design problems, and modularity. However, if these factors are not important, the controller may be designed around a simple microprocessor and support chips added externally. In the selection of the processor for the level 2 controller, the option was left open and all three types of chips were considered. The main factors influencing the choice of the processor are:

1. Number of bits.
2. Instruction set and addressing modes.
3. Speed of operation.
4. Availability of support memory chips.
5. Availability of peripheral and interface chips.
6. Architecture (availability of general purpose registers).
7. Second sourcing.
8. Cost
9. Hardware and software development support.

Factors listed are discussed below.

Number of bits

Most of the commonly used microprocessors have a word length of 4, 8, 16 or 32 bits. The bit size is important because it affects

the speed of data manipulation as well as the accuracy of arithmetic operations. Both these factors are crucial to the implementation of an adaptive controller. The problem of word length/quantization in sampled data systems has been considered in a number of references [82-84]. Three common quantization sources are input signal quantization, coefficient quantization and quantization in arithmetic operations. Problems due to finite word length were a serious problem with some of the early 4-bit microprocessors [82]. An example of a successful implementation of an adaptive controller on an 8-bit microprocessor is given in [81]. In this implementation an Intel 8085 microprocessor with a 12-bit A/D converter was used. A floating point software package was used to improve the accuracy of the calculations. Sixteen-bit manipulation can be carried out in 8-bit microprocessors by stringing two 8-bit words. However, for higher speed, it is necessary to use a 16-bit microprocessor to implement the self tuning regulator.

Instruction set and addressing modes

The basic instruction set and addressing modes available to the programmer varies from one microprocessor to another. A larger instruction set and more addressing modes result in a smaller program length and higher execution speeds.

Speed of operation

Since all instructions are executed by sequential data manipulation within the microprocessor, a clock signal is required to

provide the timing. The period of the clock is referred to as 'basic cycle time', and instructions are executed in 'integral' multiples of the cycle time. Typical clock frequencies vary from 1 MHz to 12 MHz. However, because the time taken to execute an instruction also depends on the interval working of the microprocessor, the clock frequency does not provide an accurate indication of the speed of operations. In general, bench mark tests have to be carried out to compare the speed of execution of two microprocessors.

Availability of memory chips

All microprocessor applications require memories for storing programs and data. Basically two types of memories are required. Read only memory (ROM) storing programs and read-write memories (RWM) also commonly called random access memories (RAM) for storing variables and intermediate data. In some cases (microcomputers and microcontrollers), RAM and ROM are often included on the chip. The size of memory required, varies from one application to another.

Availability of peripheral and interface chips

It is evident from figure 6.5 that in addition to CPU, RAM and ROM, several other integrated circuits are required in the controller. General purpose input output (I/O), timers, analog input-output, etc., are required. Most I.C. manufacturers make a variety of interface chips some of which can be 'hooked up' easily, while others require additional circuitry.

Microcontrollers have some of these features built-in on the chip along with the CPU, RAM and ROM.

Architecture

As mentioned earlier, the performance of the system depends on the instruction set, clock frequency and the number of bits in the microprocessor. Performance is also affected by the architecture of the processor. One of the most important factors is the number of addressing modes and the availability of general purpose registers. A large number of registers implies that less time is required for data manipulation and fetching. This is due to the fact that the time required to fetch or store data in external memory (RAM) is much more than that for the register.

Another important factor is the interrupt structure of the microprocessor. Interrupts are required for the controller to take action rapidly, whenever a fault or alarm condition occurs. Typical alarm conditions are power blackout and brownouts, system failures, etc.

Second sourcing

It is important to choose a component which will be available throughout the life time of the controller. Due to rapid development of the microprocessor technology, it is possible that the microprocessor used in the design will be superseded by a better one, and the manufacturer may abandon the production of the older version. With a chip that is second and third sourced, there is a better chance of a continuous supply.

Development support

This factor was also considered when deciding the processor to be used for level 1. Microprocessor and associated peripheral circuits represent only the hardware of the controller. The time required to develop the software depends on the availability of suitable development aids. A development system, which has assemblers and high level language compilers is extremly important for developing software in a reasonable time. Although the cost of the development system is high ($10,000-$75,000), it may be pointed out that it is a one time investment.

Sixteen 16-bit microprocessors were evaluated for their suitability for the implementation of the adaptive controller. Some of the important features are listed in table 6.1. This compilation does not include 16-bit bipolar microprocessors such as the Fairchild 9445, Signetics 8X305 and the AMD Am2900, because of their high price, reduced instruction set and expensive (or inadequate) development system. Since it is not possible to tabulate all factors used to compare microprocessors, only some of the major factors have been listed. Further, due to space limitations, only the availability or non-availability of functions is given. In some cases, functions are available, but not to an acceptable degree. Such cases have been marked 'not available'.

It is evident from table 6.1 that Intel has the largest variety in 16-bit microprocessors, including some with integrated memory and I/O elements. Manufacturers offering good development support, in terms of systems, assemblers, math packages, and high

Table 6.1 Comparison of Microprocessors

Device	Manufacturer	# of Bits	Clock Speed	Address Space	Cost	General Registers	RAM Bytes	ROM	I/O			Timer	Interupts	H/W Multiply	Development Support
									Digital	A/D	D/A				
80286	Intel	16	8	16M	$240	√	×	×	×	×	×	×	√	√	√
Z8000	Zilog	16	12	8M	$50	√	×	×	×	×	×	×	√	√	√
68000	Motorola	16/32	12	16M	$450	√	×	×	×	×	×	×	√	√	√
7811	NEC	8/16	4	64K	$50	√	128-256	2K-6K	√	√	×	√	√	8 bit only	×
6809	Motorola	8/16	2	64K	$25	×	×	×	×	×	×	×	√	8 bit only	×
65816	Western Design	8/16	4	16M	$75	×	×	×	×	×	×	×	√	×	×
9995	TI	8/16	4	32K	$50	√	256	×	√	×	×	√	√	√	×
99000	TI	16	24	16M	$100	√	×	×	×	×	×	×	√	√	×
8086	Intel	16	10	1M	$100	√	×	×	×	×	×	×	√	√	√
8088	Intel	8/16	8	1M	$50	√	×	×	×	×	×	×	√	√	√
8089	Intel	8/16	8	1M	$50	√	×	×	√	×	×	×	×	×	√
8096	Intel	16	12	64K	$50-$100	√	232	8K	√	√	√	√	√	√	√
80188	Intel	16	8	1M	$75	√	×	×	×	×	×	√	√	√	√
Micro J 11	DEC	16	15	64K	$600	√	×	×	×	×	×	×	√	√	√
Micro T-11	DEC	16	7.5	64K	$80	√	×	×	×	×	×	×	√	√	√
68200	Mostek	16	6	64K	$100	√	128	×	√	×	×	√	√	√	×

level languages are Intel (80xx-80xxx), Zilog (Z8000), Motorola (68000) and the DEC (Micro T-11,J-11). DEC Micro J-11, Intel 80286 and Motorola 68000, are expensive. The Zilog Z8000, DEC Micro T-11 and Intel 8096, 8086 and 80186 are suitable from the cost/performance point of view.

One advantage that Z8000, 8086 and 80186 have over the 8096 and T-11 is that they have a larger memory addressing space. However, for the application under consideration, this is not important. The 8096 has several additional features such as high clock frequency, built in RAM, ROM, timers, digital and analog I/O, etc. Since most of these features are required for the adaptive controller, the Intel 8096 seems to be the most cost effective choice, without having to compromise on the performance. Outline specifications, along with some benefits of the 8096 are given in Table 6.2. One important advantage of the 8096 structure is the 232 byte RAM. This RAM can be accessed as bytes, words or double words. In effect, therefore, all these bytes can be used as accumulators, thus enhancing the throughput. The program length and execution time is reduced because fewer external memory 'moves' are required. In an accumulator based CPU, every byte to be manipulated has to be 'moved' from/to the memory through the accumulator. The microprocessor is offered in several options as shown in Table 6.3.

Since the on-chip ROM may not be sufficient to hold the complete adaptive control program, it may be necessary to add external memory. Also, since the EPROM version has not been announced as yet, from a development point of view it is better to choose the ROMless version.

Table 6.2: Features of the Intel 8096 microprocessor

SNO	Features	Benefits
1	16 Bit CPU	Efficient machine with high throughput and precision.
2	8K Bytes ROM	Program space.
3	232 Bytes RAM	Large on-board register file.
4	Hardware MUL/DIV	Provides good maths capability. 16 by 16 multiply or 32 by 16 divide in 6.5 microseconds at 12 MHz clock.
5	6 Addressing modes	Provides greater flexibility in programming.
6	Digital I/O	40 I/O lines [input only, output only and bidirectional].
7	Approx. 100 instructions	Provides greater flexibility in programming.
8	64K addressing space	Room for large programs.
9	8 source priority interrupt system	Capability for responding to asynchronous events.
10	Power down mode	Provides RAM backup for recovery from blackouts.
11	Timers	4 - 16-bit timers for time keeping operations.
12	A/D converter	10-bit A/D converter with 8 channel mux and conversion rate of 42 microseconds.
13	D/A converter	On chip pulse width modulated output.
14	Serial I/O	Full duplex with 4 modes of operation.
15	Baud rate generator	Built in baud rate generator of rates upto 187.5 Kbauds.

Table 6.3: Options available with the 8096

Options		68 pin	48 pin
Digital I/O only	Romless	8096	8094
	with ROM	8396	8394
Digital and Analog I/O	Romless	8097	8095
	with ROM	8397	8395

Similarly, due to the slow conversion rate of the pulse width modulator type of D/A converter, and a relatively low precision (10-bit) A/D converter, it may be better to opt for the version without the A/D and D/A converter. The selection would then be the 8096 processor with external EPROM, A/D and D/A converter.

6.4 Choice of D/A and A/D Converter [87,88]

From figure 6.4 it is evident that a D/A converter is required to convert the error between the commanded position and the current position of the robot, to an analog signal (voltage). This signal is then amplified and transmitted to the actuator.

Basically, a D/A converter involves a network of precision resistors, set of switches and some form of level shifting to adapt the switch drives to the specified logic levels. In addition, the device may contain output conditioning circuitry. Since most A/D converters contain a D/A converter as a reference element, much of the circuit is common to both devices.

As in the case of microprocessors, A/D converters are available in widely varying capabilities and price. The resolution, for example, varies from 8-bits to 18-bits. In the past, 10 and 12-bit D/A converters were used for position control. In recent years, however, due to improvement in technology, the price of units has dropped, and it is now common to use 14-16 bit converters.

For the selection of suitable D/A converters, several specifications, such as resolution, monotonicity, operating temperature range and conversion time, etc., have to be considered. However, for closed loop servo applications, monotonicity of the converter, over the complete operating temperature range is very important. If the converter is not monotonic, the servo will hunt/chatter around the code that produces the undesirable voltage drop. Further, if the converter is not monotonic over the entire range a stable system will become unstable, when a particular temperature is reached.

The corresponding specification for A/D converters is that of 'no missing codes'. As before, 'no missing codes' for the selected A/D converter must be guaranteed over the entire temperature range.

A survey of some of the available D/A converters is given in table 6.4, and that of A/D converters in table 6.5. Only D/A converters with voltage output have been considered.

Among the models listed in table 6.4, serial numbers 3 to 8 and 10 have good monotonicity over the desired temperature range. If the specification on the settling time is made more stringent, such as $\leq$ 10 microseconds, serial numbers 3 to 5 and 10 will qualify. Model

TABLE 6.4: Comparison of D/A Converters.

SNO	MODEL NO.	MANUFACTURER	RESOLUTION Bits	INPUT CONFIGURATION Bits	MONOTONICITY Bits	SETTLING TIME μs	PACKAGE Pins
1	HDD-1409	Analog Devices	14	Latched 8+6 parallel	14 at 25 ± 3°C	5	32
2	PCM53-V	Burr Brown	16	16 parallel	15 at 25°C 14 - range unspecified	3	24
3	DAC 701/703	Burr Brown	16	16 parallel	13 or 14 at 25°C and over range	8	24
4	DAC 705/707	Burr Brown	16	Double buffered 16 parallel	14 minimum at 25°C and over range	8	28
5	DAC 711	Burr Brown	16	16 parallel	15 minimum at 25°C and over range	8	24
6	DAC 377-18	Hybrid System	18	Latched 8+8+2 parallel	16 at 25°C 15 at 10-40°C	20	28
7	DAC 9377	Hybrid System	16	Latched 16 parallel	16 at 25°C 15 at 10-40°C	20	24
8	DAC-02900	ILC Data Device	16	16 parallel	16 at 10-40°C	15	24
9	DAC-02310	ILC Data Device	14	14 parallel	13 at 25°C	1.1	32
10	MN 3290	Micro networks	16	16 parallel	15 minimum at 25°C and over range	10	24

Table 6.5: Comparison of A/D converters.

SNO	MODEL NO.	MANUFACTURER	RESOLUTION	OUTPUT CONFIGURATION	NO MISSING CODES Bits	CONVERSION TIME μs	PACKAGE Pins
1	ADC 71/72	Analog Devices	16	from SAR	14 10-40°C	50 to 14 bits	32
2	AD 376		16	from SAR	14 0-70°C	17 to 16 bits	32
3	HAS-1409		14	3-state latch	14 0-85°C	9 max	40
4		Burr-Brown	14	from SAR	14 10-40oC	50 to 14 bits	32
5	ADC 76		16	from SAR	14 10-40°C	15 to 14 bits	32
6	PLM 75		16	from SAR	16 at 25°C	17 to 16 bits	32
7	HS9516-6	Hybrid Systems	16	from SAR	16 0-70°C	100 to 16 bits	32
8	ICL 7115	Intersit	14	3-state latch	14 at 25°C Not specified over range	40 to 14 bits	40
9	MN 5284	Micro Networks	16	from SAR	15 at 25°C and over range	50 max	32
10	MN 5290		16	from SAR	15 at 25°C and over range	40 max	32
11	CX 20018	Sony	16	Serial Output	No Spec.	22	28

number DAC 705/707 has the advantage of built-in buffers, which are convenient from microprocessor interfacing point of view. According to manufacturers' specifications, the model DAC 711 has been designed for servo applications and have specified enhanced linearity around the zero output point.

High speed, high resolution A/D converters have been listed in table 6.5. Since for this application, the conversion speed is not very critical, the relatively low speed, low cost, industrial standard ADC 71/72 may be used. The unit price of ADC 71/72 is approximately $150.00. If the resolution of the converter can be sacrificed, a 12-bit A/D converter with conversion rates of < 50 microseconds are available for less than $50.00. Examples of suitable 12-bit A/D converters are ADC 674A (Burr-Brown) and ADC 574A (Datel).

6.5 Position Feedback Sensor [91]

Key elements in position control systems are the motion transducers. For angular motion, these include rotating components such as synchros, resolvers and rotary shaft encoders, which are mounted directly on the motor shaft. As described in Chapter 3, motor shaft feedback is not adequate for accurate position control because it does not reflect errors due to backlash in gears, elasticity of the robot arm, etc. A review of position sensing elements for 'end point' feedback was given in section 3.3. In this section, we review only shaft encoding systems listed above.

For industrial applications, resolvers and synchros (both commonly referred to as synchros) are favoured because of their

accuracy, ruggedness and reliability. Synchros produce a 'Y' connected output, while resolvers produce independent sine and cosine a.c. voltages, that unambiguously define shaft angles. Using this sensor requires a corresponding synchro-to-digital (S/D) converter on the level 2 controller. S/D converters are expensive, with prices ranging in the $300 to $500 range.

To obtain higher accuracy, a two speed resolver system has to be used. In such systems 'coarse' resolver shaft turns with the motor in a 1:1 ratio, while a second 'fine' resolver geared to the first in a 32:1 ratio. This requires an even more sophisticated two-speed resolver-to-digital converter. In spite of being an expensive approach, it is used in some Cincinnati-Milacron robot systems.

Another approach is to use optical or magnetic shaft encoders instead of synchros. Optical devices are by far the most popular, but magnetic units have the advantage that they do not have light sources to decay or fail. They also provide better resistance to vibration and shock. These devices are called 'incremental' position encoders because they give only relative position information. Incremental devices produce two serial pulse train outputs in quadrature facilitating direction sensing. Decoding on the controller is simpler than for synchro systems and usually involve only an up/down counter.

An advantage of using absolute encoders is that rezeroing is not necessary when power is restored after an interruption or when excessive noise obliterates the digital shaft output. In the absolute encoder, an exact correspondence exists between the shaft position and the digital word, which is unique to the position. When an

incremental encoder is used, software in the level 2 processor has to keep track of the actual position after a reset.

Regardless of whether an incremental or an absolute encoder is used, an important factor from the safety point of view is the use of limit switches. They should be wired such that the robot is not allowed to swing beyond its operating envelope.

6.6 Software [88]

All digital computers operate under the context of 'software' or 'programs'. The most important factor for developing a program is the selection of an appropriate language. For real time applications such as the adaptive control of a robot, there are two general objectives. Firstly, the code generated should be efficient and compact so that the required set of instructions can be executed within the specified time interval. Secondly, programming should be easy, which implies that the language should permit writing of programs in easily understandable terms.

Programming languages are classified in terms of 'levels'. The lowest level refers to writing programs in the 'machine code', i.e. the basic code in the form of binary numbers for the processor. This results in perhaps the most efficient program, from the point of view of compactness and execution times. However, programming complex tasks in machine code is very difficult and time consuming.

At the next level, the program may be written in HEX code. Hex code simply refers to representing the machine code in groups of 4-bits rather than in binary. It suffers from essentially the same

problems as writing in machine code. 'Assembly language' is considered as the next level. In this language various instructions for the microprocessor are represented in the form of mnemonics instead of numbers. Writing in this language requires the availability of an 'assembler' software for the processor used, which translates the user program from mnemonics to the machine code. Although it is much more convenient, assembly language programming also requires an intimate knowledge of the architecture of the processor. In the past assembly language was the most popular language for programming a microprocessor - particularly for real time applications. However, in the past 10 years, several efficient 'high level' languages have been developed for writing programs for the microprocessor. A 'compiler' or an 'interpreter', is required to translate the 'high level' language program to the machine executable code. Using high level language has the advantage that programming is easy and the programmer does not require as much knowledge of the architecture of the microprocessor.

A large number of such languages have been developed. Popular languages are ADA, ALGOL, APL, BASIC, C, COBOL, CORAL, FORTH, FORTRAN, MODULA, PASCAL, PL/M and RTL/2.

Some of the languages listed above are compared and results summarized in table 6.6.

It seems that from a futuristic point of view, ADA may be an appropriate choice. However, at present, few compilers are available for software development in ADA. Further, it seems that to make full use of special features of ADA, a 32-bit processor is required. It is

Table 6.6: Comparison of programming languages.

SNO	LANGUAGE	AVAILABILITY ON MOST μP	BLOCK STRUCTURED?	REAL TIME I/O	EASY INTERFACE WITH ASSEMBLY	CODE EFFICIENCY	REMARKS
1	ADA	✗	✓	✓	✓	✓	Language standardized by U.S. Department of Defence. Expected to become standard language for real time applications.
2	BASIC	✓	✗	✗	✗	✗	Useful for commercial data processing.
3	C	✗	✓	✗	—	✓	Basically designed for mainframe systems.
4	FORTH	✗	✗	✓	—	✓	Lacks floating point packages.
5	FORTRAN	✗	✗	✗	✗	✗	Suitable for scientific or engineering calculations.
6	MODULA	✗	✓	✗	✓	✓	As above. Has good structure and i/f with assembly.
7	PASCAL	✗	✓	✗	✓	✓	As above. Gaining popularity with microprocessors.
8	PL/M	✓	✓	✓	✓	✓	Very popular for real time uP programming. Particularly for Intel uPs.

significant to note that Intel is developing a 32-bit microprocessor, designed specifically with ADA in mind.

At present, of all the 'high level' languages available for the development of microprocessor software, the PL/M offers maximum benefit. This is especially true if Intel microprocessors like the 8096 are selected.

6.7 Concluding Remarks

In this chapter we discussed some of the basic issues involved in implementing the adaptive controller described in Chapter 5. The first aspect is to decide the structure of the controller. Due to several advantages, such as modularity and easy expandability, a two level, hirearchial control structure is chosen.

The second aspect discussed is the identification and selection of various components required for both levels of the controller. There are basically four critical components in the system - microprocessors, A/D and D/A converters, position feedback sensor and the programming language for the microprocessors.

Selection of the microprocessors is complicated because of the availability of about 60 processors with widely varying specifications and capabilities. However, because the numerical precision required is of at least 16-bits, the number of microprocessors to choose from was reduced to sixteen. From a comparative study of these (sixteen) microprocessors, the Intel 8096 was seen to be a suitable one.

A similar approach was used for the selection of the A/D and D/A converter. Salient features of 10 D/A converters are listed in

table 6.4 and those for 11 A/D converters in table 6.5. A review of four types of shaft encoders is also included. It is shown that while absolute position encoders such as the 2-speed resolver are more accurate, incremental position encoders are most cost effective.

Until a few years ago, programming a microprocessor required a through knowledge of its architecture. Programs were usually written in the assembly language which is difficult to learn and is also time consuming. Recently, several high level languages are available which offer advantages of easy programming and also result in efficient code for real time applications. A review of some of the popular languages show that the PL/M is most suitable.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

As a result of technological advances, many industrial processes have become automated in the past two decades. More recently, automation has been in the form of robots. Initially, the distinguishing features of robotic systems, which set them apart from existing automated machines were their versatility and flexibility. They were developed more in the form of programmed automation, and were unable to react to changes in environment. In the past few years, emphasis has been on the development of 'interactive' robots, which can apprehend and react to changes in its environment. As a consequence, more complex and sophisticated control systems are required to control the robot.

In the analysis of robotic control systems there are three main problems.

The first is that of coordinate transformation. The position of the tool held by the robot gripper is usually specified in terms of its position and orientation in the cartesian coordinate system. The control system is required to transform this vector of coordinates to a vector in the joint coordinate space. This problem is well studied and is solved by 'inverse' transformation.

The second problem is that of the dynamic control of the robot. This involves generation of appropriate signals to drive each joint such that the tool can be moved from the present position to the desired one.

The third issue is that of process control. This involves controlling all process related parameters. For example, in the case of arc welding, this would involve controlling the voltage, feed rate of the electrode, and control of the shielding gas, etc.

The second problem, that of position control of the robot is addressed in this thesis.

It is shown in Chapter 3 that due to the influence of several factors such as the inertia of the arm, elasticity of limbs, etc., a conventional position control system utilizing feedback from the motor shaft is inadequate. It is essential to use feedback from the actual position of the tool. A scheme utilizing such a feedback in an additional loop was implemented on the arc welding robot. Since the primary intention was to track a seam deliberately placed at an angle to the programmed path, a proximity probe was used to detect the deviation. This signal was transmitted to the control computer in the outer loop which generated an error signal for the actuator.

However, for a general position control problem, several other methods were reviewed, from which it is evident that the dynamic control of the robot arm could be improved by using advanced control methods such as adaptive techniques. Since adaptive control techniques require an accurate parametric model of the robot arm, various methods for modeling a robot were reviewed. It is evident

from the literature surveyed, that the dynamics of the robot are commonly modelled using laws of classical (Newtonian) mechanics.

An accurate model, based on this approach requires the knowledge of the internal structure of the robot, masses of various links, frictional forces, characteristics of the actuator, etc. This results in a complex set of equations which are difficult to implement on microprocessors for use in real time control. It may also be observed that due to gravitational loading, the model of the robot changes significantly as a function of the orientation of various links and the load carried by the robot. These models are often simplified by assuming linearity and neglecting coupled terms. However, models are still quite complex for real time implementation and often all actuator and mechanical characteristics have to be experimentally measured since they are not available from the manufacturer of the robot.

In this thesis, we describe a different approach to modelling the robot, which does not require the knowledge of the system and its constituents. In this method, the robot is treated as a black box and a model is derived from experimental observations. This form of modeling systems is well established and is referred to in the literature as 'system identification'.

In system identification approaches, a 'persistently exciting' signal, such as the pseudo random binary sequence (PRBS) is applied to the system. A model is derived from the samples of this input and the corresponding output signal. Since in practice, it may not be possible to apply 'persistantly exciting' inputs like the PRBS to the

robot, one has to use simple inputs like the step signal for excitation. Due to this restriction, it is not possible to use conventional system identification techniques to determine the model.

In this thesis, we describe a method for identifying the robot from samples of its step response. The method is general, and has been tested for a number of simulated systems of different complexities and orders. A discrete-time model is obtained in two steps. First, a continuous-time model is derived from the samples of input-output data. This model is then converted to a discrete-time model, which is used for adaptive control. Various aspects of modeling the robot, such as data acquisition, structural identification, parameter estimation and model verification are described in detail in Chapter 4. Results of modeling the axis with the lowest damping are given. It is seen that a third order model with a complex pair of poles represents the measured data quite accurately. As mentioned earlier, this scheme does not require the knowledge of the internal subsystem of the robot and it also accounts for the actuator and arm dynamics.

An equivalent discrete time model is given in Chapter 5. A review of various adaptive control methods show that the gain scheduling method has the fastest response. This is due to the fact that controller parameters for a variety of situations are precalculated and stored. However, the disadvantage of its being an open loop method makes it unsuitable for robotic applications.

Since the model of the robot arm is non-minimal phase, the pole-placement self tuning type of adaptive controller is most

suitable. Chapter 5 gives details of the design, testing and performance of the regulator for the robot arm. There are two configurations commonly used for the regulator. It is shown that for pole-placement self tuning operation, both representations are quite similar in performance, though due to differences in the design procedure, numerical values for the controller parameters are different. Extensive simulation tests were done to verify the performance of the self tuning regulator, for variations due to sampling interval and changes in the robot dynamics. Further tests were done to verify the robustness and adaptivity of the regulator. Adaptivity was checked by monitoring the regulated output while the robot dynamics were changed abruptly. It is demonstrated that the rate of adaptation is significantly improved by simple binary switching of the 'weighting factor' and elements of the covariance matrix used for on-line identification. Numerical robustness is verified by studying the effect of truncating controller parameters, on the performance of the self tuning regulator.

Finally, guidelines are given for the choice of the controller structure and the selection of components for implementing the regulator.

A two level heirarchy is recommended for the implementation of the designed self tuning regulator. It is suggested that the higher level controller be used for operator communication, coordinate transformation and controlling process related parameters. The lower level controller (one for each axis) may be used for dynamic position control. A comprehensive survey of several microprocessors, analog to

digital (A/D), digital to analog (D/A) converters and computer programming languages is included, and suggestions regarding their choice is given.

In summary, this thesis describes a new method for modeling an industrial robot. The main features of this method are that no a-priori information about the structure and its constituent subsystems is required, and it is possible to use simple signals like the step input for system excitation. This method was applied to model a robot and the model was used for the design of an explicit pole-placement type of self tuning regulator. Extensive simulations show satisfactory performance of the regulator for abrupt changes in the system dynamics. The adaptation rate is shown to improve significantly by simple manipulation of the forgetting factor and covariance matrix during identification. Several implementation aspects such as the structure of the controller, selection of microprocessors, A/D, D/A converters and programming languages are discussed.

7.2 Suggestions for Further Research

1. The design of the self tuning regulator presented assumes that the structure (order) of the model of the robot arm is fixed, and only its parameters change due to orientation of links and loading effects. Further investigation is needed to verify this. If investigations indicate different model structures to be more appropriate, the design of the regulator has to be modified to account for a varying model structure.

2. Further research is required to develop accurate models which include the effect of coupling between joints and yet be suitable for real time implementation.

3. Although accurate real data was used for modeling the robot, the performance of the self tuning regulator was verified by simulation. Design details and recommendations on the control structure and component selection given in this thesis should be used for experimental verification of the performance of the regulator.

4. Since it is apparent that in future several robots will be operating simultaneously, further work is necessary for the development of a hierarchal controller for multi-robot systems. Such controllers will be required for coordinated operation and also for intelligent robot-power management, particularly in case of failure of some robots.

5. Investigation and development of higher level control systems for intelligent robots - using artificial intelligence [93-95] concepts.

REFERENCES

[1] Joseph, F., Robotics in practice, Engelberger, N.Y., 1980, pp. 171-179.

[2] Tlusty, J., Szvoboda, G. and Rafauli, R., "A continuous path algorithm for robot control", Proceedings Canadian Conference on Robotics, Canada, Sept., 20-21, 1982, pp. 91-101.

[3] Rafauli, R., Sinha, N.K. and Tlustv, J., "A distributed microprocessor control system for an industrial robot", Proceedings, IEEE Conference on Mini- and Microcomputers, San Francisco, 1981, pp. 319-323.

[4] Thomas, M. and Tesar, D., "Dynamic modeling of serial manipulator arms", Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 104, Sept. 1982, pp. 218-228.

[5] Luh, J.Y.S., Walker, M.W. and Paul, R.P.C., "On-line computational scheme for mechanical manipulators", Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 102, June 1980, pp. 69-76.

[6] Lumelsky, V.J., "Iterative co-ordinate transformation procedure for one class of robots", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-14, No. 3, May/June 1984, pp. 500-505.

[7] Stone, H.W. and Neuman, C.P., "Dynamic modeling of a three Degree-of-Freedom Robotic Manipulator", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-14, No. 4, July/August 1984.

[8] Ardayfio, D.D., "Dynamic Formulation Techniques for Robot Manipulators", The Fourth International Conference on Mathematical Modelling in Science and Technology, Zurich, Switzerland, August 1983, pp. 897-907.

[9] Szvoboda, G., A continuous path algorithm and its implementation to a micro computer controlled industrial robot, M.Eng. thesis, McMaster University, 1982.

[10] Lee, G.S.G., "On the control of robot manipulator", Proceedings of SPIE on Robotics and Robot Sensing Systems, San Diego, California, August 1983, pp. 58-83.

[11] Coiflet, P. and Chironze, M., An Introduction to Robot Technology, McGraw-Hill Book Company, U.S.A., 1983.

[12] Dubowsky, S. and DesForges, D.T., "The application of Model-reference Adaptive Control to robotic manipulators", Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 101, Sept. 1979, pp. 193-200.

[13] Koivo, A.J. and Guo, T.H., "Adaptive linear Controller for Robotic Manipulators", IEEE Transactions on automatic control, Vol. AC-28, No. 2, February 1983, pp. 162-171.

[14] Cannon, R.H. and Schmitz, E., "Initial Experiments on the End point control of a flexible one-line robot", International Journal of Robotics Research, Vol. 3, No. 3, pp. 62-75.

[15] Karnik, A.M. and Sinha, N.K., "The adaptive control of a robot arm for arc welding process", Proceedings of the International Conference on Systems, Man and Cybernetics, Bombay, India, 1983, pp. 614-618.

[16] Rafauli, R., A distributed microprocessor control system for an industrial robot, M.Eng. thesis, McMaster University, June 1981.

[17] Rafauli, R., Sinha, N.K. and Tlusty, J., "A microprocessor control system for an industrial robot", Proceedings, IEEE International Electrical, Electronics Conference & Exposition, Toronto, 1981, pp. 118-119.

[18] Anon., Hobart Welding Guide, Hobart School of Welding Technology, Ohio, 1980.

[19] Ching, J., "A simulated robot control language and programming environment, M.Eng. thesis, McMaster University, under preparation.

[20] Richardson, R.W., Gutow, D.A. and Rao, S.H., "A vision based sensor for arc weld pool size control", Measurement and control for batch manufacturing, Hardt, D.E. (Ed.), N.Y., 1982.

[21] Estochen, E.L., Neuman, C.P. and Prinz, F.B., "Application of acoustic sensors to Robotic seam tracking", IEEE Transactions on Industrial Electronics, Vol. IE-31, No. 3, August 1984, pp. 219-224.

[22] Stauffer, R.N., "Update on non-contact seam tracking systems", Robotics Today, August 1983, pp. 29-34.

[23] Kawahara, M., "Tracking control system using image sensor for arc welding", Automatica, Vol. 19, No. 4, pp. 359-363.

[24] Boilot, J.P. and Begin, G., "IR vision applied to adaptive welding", reported at Robotics International, ASEA, Malton, January 1983.

[25] Freund, E., "Direct design methods for the control of Industrial robots", Computers in mechanical engineering, April 1983, pp. 71-79.

[26] Franklin, G.F. and Powell, J.D., Digital control of dynamic system, Addison-Wesley Publishing Company, 1980.

[27] Sinha, N.K. and Kuszta, B., Modeling and identification of dynamic systems, Van-Nostrand-Reinhold Publishing Company, N.Y. 1983.

[28] Eykhoff, P., System identification - parameter and state estimation, John Wiley and Sons, Great Britain, 1976.

[29] Schwarzenback, J. and Gill, K.F., System modelling and control, Edward Arnold, U.K., 1978.

[30] Isermann, R., Digital Control Systems, Springer Verlag, N.Y., 1981.

[31] Saridis, G.N. and Lobbia, R.N., "Parameter identification and control of linear discrete-time systems", IEEE Transactions on automatic control, Vol. AC-17, No. 1, February 1972, pp. 52-60.

[32] Karnik, A.M. and Sinha, N.K., "Modeling a robot arm from sampled input-output data", Proc. 7th IFAC Symposium on Identification and System Parameter Estimation, York, U.K., July 1985.

[33] Astrom, K.J. and Eykhoff, P., "System identification - a survey", Automatica, vol. 7, 1971, pp. 123-162.

[34] Isermann, R., "Practical aspects of process identification", Automatica, Vol. 16, 1980, pp. 575-587.

[35] Rissanen, J., "Recursive identification of linear systems", SIAM Journal Control, Vol. 9, No. 3, 1971.

[36] Rake, M., "Step response and frequency response methods", Automatica, Vol. 10, 1980, pp. 519-526.

[37] Young, P.C., "An instrumental variable method for real time identification of a noisy process", Automatica, Vol. 6, 1970, pp. 271-287.

[38] Ljung, L. and Glover, K., "Frequency domain versus time domain methods in system identification", Automatica, Vol. 17, No. 1, 1981, pp. 71-86.

[39] Wellstead, P.E., "Non-parameteric methods of system identification" Automatica, Vol. 17, No. 1, 1981, pp. 55-69.

[40] Ljung, L. and Sodestrom, T., Theory and practice of recursive identification, The MIT Press, Massachusetts, 1983.

[41] Karnik, A.M. and Sinha, N.K., "A direct approach to modeling an industrial robot from samples of input-output data", Robotica, Vol. 2, 1984, pp. 161-167.

[42] Zhou, Qi-Jie and Sinha, N.K., "Identification of continuous-time systems from sampled data: a comparison of 3 direct methods", Proc. 7th IFAC/IFORS Symposium on Identification and System Parameter Estimation, York, U.K., 1985.

[43] Prasad, T. and Sinha, N.K., "Modelling of continuous-time systems from sampled data using trapezoidal pulse functions", IEEE International Conferrence on Systems, Man and Cybernetics, India, 1983, pp. 427-430.

[44] Bohn, E.V., "Estimation of continuous-time linear system parameters from periodic data", Automatica, Vol. 18, No. 2, 1982, pp. 27-36.

[45] Puthenpura, S. and Sinha, N.K., "A procedure for determining the optimal sampling interval for system identification using a digital computer", Canadian Electrical Engineering Journal, to appear.

[46] Young, P., Jakeman, A. and McMurtrie, R., "An instrumental variable method for model order identification", Automatica, Vol. 16, 1980, pp. 281-294.

[47] Young, P., "Parameter estimation for continuous-time models - a survey", Automatica, Vol. 17, No. 1, 1981, pp. 23-39.

[48] Stoica, P., "A test for whiteness", IEEE Transactions on Automatic Control, Vol. AC-22, 1977, pp. 992-993.

[49] Box G.E.P. and Jenkins, G.M., Time series analysis - forecasting and control, Holden Day, San Francisco, 1970.

[50] Davies, P. and Hammond, J.K., "A comparison of Fourier and parametric methods of structural system identification", Transactions of ASME Journal of Dynamic Systems, Measurement and Control, Vol. 106, 1984, pp. 40-48.

[51] Astrom, K.J. and Whittenmark, B., Computer controlled systems - theory and design, Prentice-Hall, Inc., U.S.A., 1984.

[52] Harris, C.J. and Billings, S.A. [Ed], Self-tuning and adaptive control: Theory and Applications, Peter Peregrinus Ltd., U.K., 1981.

[53] Landau, Y.D., Adaptive control - the model reference approach, Marcel Dekker, Inc., 1979.

[54] Saridis, G.N., Self-organizing control of stochastic systems, Marcel Dekker, Inc., New York, 1977.

[55] Johnson, T.L., Harvey, C.A. and Stein, G., "Self tuning regulator design for adaptive control of aircraft wing/store flutter", IEEE Transactions on Automatic Control, Vol. AC-27, No. 5, October 1982, pp. 1014-1023.

[56] Sinha, N.K., Gupta, M.M. and Nikiforuk, P.N., "Recent advances in adaptive control", Journal of Cybernetics, Vol. 6, 1976, pp. 79-100.

[57] Isermann, R., "Parameter adaptive control algorithms - a tutorial", Automatica, Vol. 18, No. 5, 1982, pp. 513-528.

[58] Asher, R.B., Andrisanill, D. and Dorato, P., "Bibliography on adaptive control systems", Proceedings of the IEEE, Vol. 64, No. 8, August 1976, pp. 1226-1240.

[59] Dubowsky, S. and Des Forges, D.T., "The application of model-referenced adaptive control to Robotic manipulators", Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 101, September 1979, pp. 193-200.

[60] Balestrino, A., DeMaria, G. and Sciaricco, L., "An adaptive model following control for robotic manipulators", Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 105, September 1983, pp. 143-151.

[61] Dornfeld, D.A., Tomizuka, M. and Langari, G., "Modeling and adaptive control of arc welding processes", Measurement and control for batch manufacturing, Hardt, D.E. [Ed.], 1982.

[62] Yoshimura, T. and Tomizuka, M., "Application of model reference adaptive techniques to a class of non-linear systems", Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 102, June 1981, pp. 158-163.

[63] Kailath, T., Linear Systems, Prentice-Hall, Inc., U.S.A., 1980.

[64] Astrom, K.J. and Whittenmark, B., "On self-tuning regulators", Automatica, Vol. 9, 1973, pp. 185-199.

[65] Astrom, K.J., "Theory and applications of adaptive control - a survey", Automatica, Vol. 19, No. 5, 1983, pp. 471-486.

[66] Astrom, K.J., Borrison, U., Ljung, L. and Whittenmark, B., "Theory and applications of self tuning regulators", Automatica, Vol. 13, 1977, pp. 457-476.

[67] Clarke, D.W. and Gawthrop, P.J., "Self tuning control", Proceedings of IEE, Vol. 126, No. 6, June 1979, pp. 633-640.

[68] Astrom, K.J. and Whittenmark, B., "Self-tuning controllers based on pole-zero placement", IEE Proceedings, Vol. 127, Pt. D., No. 3, May 1980, pp. 120-130.

[69] Wellstead, P.E., Prager, D. and Zanker, P., "Pole assignment self tuning regulator", Proceedings IEE, Vol. 126, No. 8, August 1979, pp. 781-787.

[70] Wellstead, P.E., Edmunds, J.M., Prager, D. and Zanker, P., "Self-tuning pole/zero assignment regulator", International Journal of Control, Vol. 30, No. 1, 1979, pp. 1-26.

[71] Prager, D.L. and Wellstead, P.E., "Multivariable pole-assignment self-tuning regulators", IEEE Proceedings, Vol. 128, Pt. D, No. 4, January 1980, pp. 9-18.

[72] Elliott, M. and Wolovich, W.A., "Parameter adaptive identification and control", IEEE Transactions on Automatic Control, Vol. AC-24, No. 4, August 1979, pp. 592-599.

[73] Clarke, D.W., "Self-tuning control of non-minimum-phase systems", Automatica, Vol. 20, No. 5, 1984, pp. 501-517.

[74] Allidina, A.Y. and Hughes, F.M., "Self-tuning controllers for deterministic systems", International Journal of Control, Vol. 37, No. 4, 1983, pp. 831-841.

[75] Clarke, D.W. and Gawthrop, P.J., "Self-tuning controller", Proceedings IEE, Vol. 124, No. 10, 1977, pp. 889-894.

[76] Gawthrop, P.J., "Some interpretations of the self-tuning controller", Proceedings IEE, Vol. 124, No. 10, 1977, pp. 889-894.

[77] Borison, U., "Self-tuning regulators for a class of multivariable systems" Automatica, Vol. 15, 1974, pp. 209-215.

[78] Wellstead, P.E. and Zanker, P., "Servo self-tuners", International Journal of Control, Vol. 30, No. 1, 1979, pp. 27-36.

[79] Fortescue, T.R., Kershenbaum, L.S. and Ydstie, B.E., "Implementation of self-tuning regulators with variable forgetting factors", Automatica, Vol. 17, No. 6, 1981, pp. 831-835.

[80] Goodwin, G.C. and Sin, K., Adaptive filtering, prediction and control, Prentice-Hall, Inc., N.Y., 1984.

[81] Bonivento, C. and Spisni, F., "Realization of a microprocessor based adaptive controller", Control and Computers, Vol. 11, No. 1, 1983, pp. 1-5.

[82] Dorato, P., "Theoretical developments in discrete-time control", Automatica, Vol. 19, No. 4, 1983, pp. 395-400.

[83] Farrar, F.A. and Eidens, R.S., "Microprocessor requirements for implementing modern control logic", IEEE Transactions on Automatic Control, Vol. AC-25, No. 3, 1980, pp. 461-468.

[84] Moroney, P., Willsky, A.S. and Houpt, P.K., "The digital implementation of control compensators: the coefficient word length issue", IEEE Transactions on Automatic Control, Vol. AC-25, No. 4, 1980, pp. 621-630.

[85] Wittenmark, B. and Astrom, K.J., "Practical issues in the implementation of self-tuning control", Automatica, Vol. 20, No. 5, 1984, pp. 595-605.

[86] Cushman, R.H., "µp support chips present a wide array of choices", Electronic Design News, March 7, 1985, pp. 137-186.

[87] Travis, B., "High resolution D/A and A/D converters advance in crucial specifications", Electronic Design News, February 1985, pp. 65-90.

[88] Kocher, A.K. and Burns, N.D., Integrating microprocessors into product design, Marcel Dekker, Inc., N.Y., 1983.

[89] Cushman, R.H., "EDN's eleventh Annual µp/µc chip directory", Electronic Design News, November 15, 1984, pp. 165-290.

[90] Anon, Microcontroller Handbook, Intel Corporation, 1984.

[91] McDermott, J., "Digital motion control", Electronic Design News, January 12, 1984, pp. 129-152.

[92] Anon., Intellec series IV manuals, Intel Corporation, 1984.

[93] Simons, J., Van Brussel, H., Deschutter, J., and Verhaert, J., "A self-learning automation with variable resolution for high precision assembly by industrial robots", IEEE Transactions on Automatic Control, Vol. AC-27, No. 5, October 1982, pp. 1109-1113.

[94] Carter, W.C., "Modular multiprocessor design meets complex demands of robot control", Control Engineering, March 1983, pp. 73-77.

[95] Saridis, G., "Toward the realization of intelligent controls", Proceedings of the IEEE, Vol. 67, No. 8, August 1979, pp. 1115-1133.