'Self-Tuning Digital Controllers for Servo Systems'

by

Ghassan Mohammad Al-Sadigi, B.Sc, M.Sc.

A doctoral thesis submitted in partial
fulfilment of the requirements of
Loughborough University of Technology

April 1987

Supervisors

J. E. Cooling B.Sc., C.Eng., MIEE.

Electronic and Electrical Engineering Dept.

Dr. A. H. Whitfield

Engineering Mathematics Dept.

Acknowledgement

I wish to express my sincere thanks to Mr. Jim Cooling, of Electronic and Electrical Engineering Dept. and Dr. Allen Whitfield, of Engineering Mathematics Dept., for their invaluable help and guidance during my period of research. Their supervision has been excellent throughout and I am indebted to them both for the inspiration and encouragement they have given me.

I would also like to thank Mr. Mark Morgan Lloyd for his technical advice concerning the software work involved in the thesis. I am thankful to the staff and student members and technicians for their assistance.

SYNOPSIS

Self-tuning Digital Controllers for Servo Systems.

Adaptive self-tuning systems have been the subject of a great deal of research effort in recent years. Practical applications have lagged behind such work, in the main being applied in the process industries. Few servo applications have been reported, the wide bandwidths and demanding performance specifications raising problems not found in the process world. The research project described here is concerned with the use of self-tuning digital controllers applied to servo-systems, specifically an electro-mechanical actuation unit. Practical limitations, such as stiction, friction and velocity saturation effects are taken into account.

The following factors are considered within this thesis;

* Plant identification and modelling using off-line identification techniques.

* Plant identification and modelling using on-line methods.

* Control strategies based on statistical and deterministic principles.

* Mathematical methods to achieve optimal plant models through the use of model reduction techniques.

* Self-tuning strategy evaluation via simulation procedures.

* Self-tuning strategy evaluation via plant testing.

* Development and use of a mechanical simulator for the evaluation of control strategies.

* Hardware design and development of a 16 bit microprocessor based digital controller.

* Software design of the identification, control and self-tuning algorithms.

* The use of a PC based integrated work-station as a tool for the development of embedded control systems.

Significant improvements in mathematical methods for model order reduction are presented, their validity being demonstrated in practical testing.

The digital controller described here uses an Intel 8088 processor augmented by an 8087 Numeric Data Processor, plant interfacing being carried out using analogue to digital and digital to analogue converters. Controller software is written in ASM86 and Pascal/MT+, that for the simulation being produced in Fortran 77.

CONTENTS

CHAPTER 10: CONCLUSION

APPENDIX A: THE CONTROLLED PLANT

APPENDIX B: SYSTEM IDENTIFICATION

APPENDIX D: SELF-TUNING CONTROLLER - HARDWARE

APPENDIX E: MODEL DETERMINATION

CHAPTER-1

1 INTRODUCTION

1.1 CONTROL SYSTEMS

1.1.1 General

Control systems vary considerably. At one extreme are fast high performance systems such as weapon stabilisers. In such applications many aspects of the design are new and unique, the related control design effort being considerable. At the other end are process systems where the control engineer is asked merely to supply a three-term controller. Here the control design is minimal (though the system design may be considerable).

The system described here, together with relevant research and design techniques, falls into the first category. Fig.1.1 shows the main objectives of the programme, these being the development of the target system controller and the development of appropriate control techniques. The target system ("plant") is a real as opposed to a simulated one, being described in section 1.2

1.1.2 Control system design methods

Numerous control techniques are available for use in the design of control systems [1.1]. These are divided into two major categories, optimal control techniques [1.2] and classical control techniques [1.3].

In optimal control, widely used in the aerospace industries, the objective is to minimise a specified cost function. These methods are highly mathematical. As a result most practical engineers do not understand optimal methods; neither can they specify their system performance requirements in a relatively straightforward

manner. Highly skilled and specialised engineers are needed in such situations. It is therefore not surprising that these techniques are not widely accepted in control system work.

In contrast classical control techniques are widely used because;

* they are relatively simple,
* they are well understood by practical engineers,
* performance criteria can be specified simply and unambiguously in both the time and frequency domains.

Here the tuning of the controller is adjusted as required by the plant engineer, controller settings being based on both experience and trial and error techniques [1.4].

Self-tuning designs using classically specified performance criteria are serious contenders for use in modern control systems. In such systems the controller is adjusted automatically until (hopefully) the specified performance is achieved. Therefore tuning does not require operator experience or involvement, an added feature of self-tuning controllers.

1.1.3 Self-tuning systems

Adaptive self-tuning systems have been the subject of a great deal of research effort in recent years, a major review being given by Warwick [1.5]. Practical applications have lagged behind such work, in the main being applied in the process industries (1.6,1.7,1.8,1.9). Few servo applications have been reported, though the advantages of self-tuners in these areas have been shown by Daley [1.10] and Hope [1.11]. The wide bandwidths and demanding performance specifications raise problems not found in the process world. In fact such problems have lead researchers to develop alternative adaptive control techniques [1.12].

1.2 THE CONTROLLED PLANT

A block diagram of the "plant" used for control system development is shown in Fig.1.2, Fig.1.3 being a photo of the actual test rig. This is an electromechanical actuator (suitable for the use with process control valves) coupled to a mechnical load simulator. The actuator motor and associated control/power electronics are considered to be part of the plant itself, as is the load shaft position sensor.

The drive unit of the actuator is a 1/6 horse power induction motor, originally designed for 115v. 60 Hz. 3 phase operation. It is powered from a static inverter which, when used with its controller, provides full linear speed control from zero to maximum speed in both directions.

A gearbox is used to translate motor shaft rotary motion to linear motion of the load shaft. Both the load resilience and viscous forces can be varied by the rig operator. Further, the effect of valve loading is simulated using a coulomb damper (actually a disc brake) on the motor shaft; this, too, is adjustable. Position sensing is carried out using a continuous track rectilinear potentiometer.

Motor speed control is carried out by a pulse width modulated (PWM) controller in conjunction with a 3 phase static inverter [1.13]. Motor speed is determined by the analogue input signal to the controller. A detailed description of the plant is shown in Appendix-A.

1.3 SYSTEM DEVELOPMENT IN A PC ENVIRONMENT

Computerised tools for the analysis, synthesis and development of control systems have generally been based on the use of Mainframe and Mini computers. Typical of such packages are CAD on PRIME [1.14] and MATRIXx on the VAX [1.15]. Recently, however, much attention has been focussed on low cost alternatives using

personal computers (PCs). In many cases the emphasis has been on training, analysis and theoretical synthesis [1.16]. Little has been reported concerning their use in the development of practical embedded systems, typified by servo applications. Yet the PC can provide many facilities needed for all phases of the research and design activity. Central to this research work is a general purpose PC based work-station (Fig.1.4) which has been developed specifically for the design of microprocessor based control systems. The use of this forms a topic in its own right as it had a major impact on the design methods adapted during the research programme.

1.4 THESIS OBJECTIVE AND ORGANISATION

This thesis describes the development of a microprocessor-based self-tuning controller. The performance of "conventional" self-tuning techniques applied to electro-mechanical systems in presence of stiction, friction and velocity saturation are evaluated with respect to fixed digital controllers.

Off-line analysis is carried out to model the plant transfer function and determine the minimum model order.

A digital computer simulation for the plant is carried out using the derived model to evaluate its performance under various control algorithms.

In this thesis it is also shown how a PC based development environment satisfies the requirements of the control engineer, from hardware design to system testing.

The thesis organisation is as follows:

In Chapter 2 a brief general history of control is given, the development of adaptive control being described in more detail. An overview of adaptive control is given here.

Chapter 3 describes an off-line identification technique used as a means to determine the tansfer function of the controlled plant (via the PC). A number of mathematical techniques are presented which seek to improve the model order reduction technique proposed by Soderstrom. Further, a new mathematical method designed to improve reduced order model structures is also given in this chapter.

In Chapter 4 implementation considerations in digital control systems such as computational delay, choice of sampling rate, practical constraints, and numerical accuracy are discussed. This chapter also describes the design of the digital control algorithm using a number of different control criteria.

Chapter 5 deals specifically with the implementation of self-tuning algorithms. It describes the basic structural blocks of self-tuning systems and evaluates the practical implementation aspects of self-tuners. The design of a Pole/Zero cancellation self-tuner and a PID self-tuner is shown in this chapter.

Chapter 6 describes the electronic control system used as the target digital controller. The target board functions and facilities are explained. A detailed description of the electronic hardware is included in this chapter.

Chapter 7 explains the software structure and the software design technique. The role of the PC in developing both the off-line (computing) and on-line (plant controller) code is illustrated. In this chapter the implementation of the self-tuning controller (STC) is shown, the functions of the software used on both the PC and the microprocessor being explained in detail.

In Chapter 8 the advantages of using the improved transfer function reduction technique are shown using test results obtained for models of different order. The plant transfer functions used are for both real plants and simulated ones.

In Chapter 9 both simulated and actual test results for the plant control system are presented, different control techniques being used in these tests. The

actual response of system under self-tuning control (STC) is compared with that under the fixed digital controller. The evaluation and implication of these results are discussed.

In Chapter 10 the conclusions and comments arising from this work are outlined.

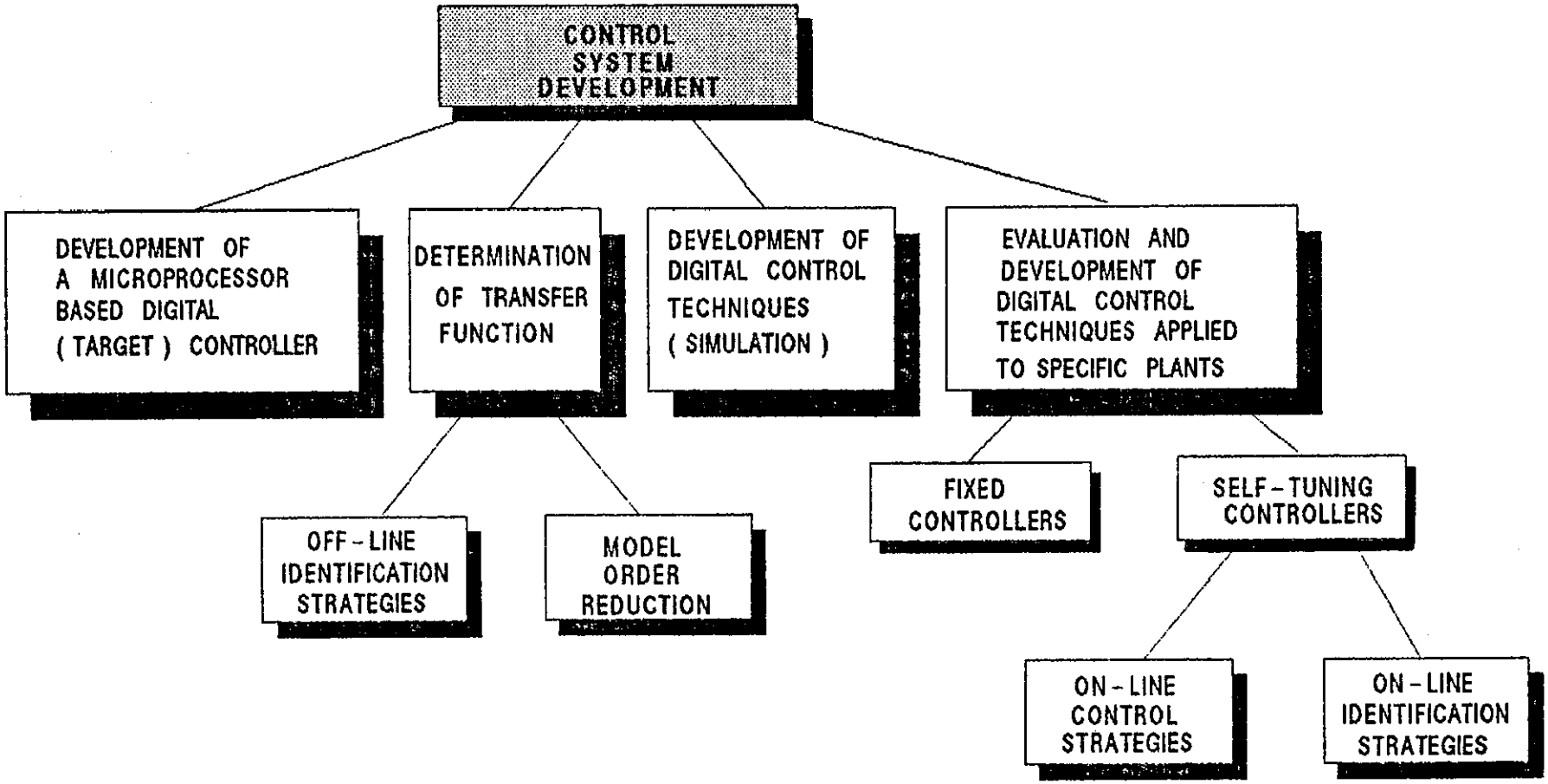


FIG.1.1

SYSTEM DEVELOPMENT PROGRAMME

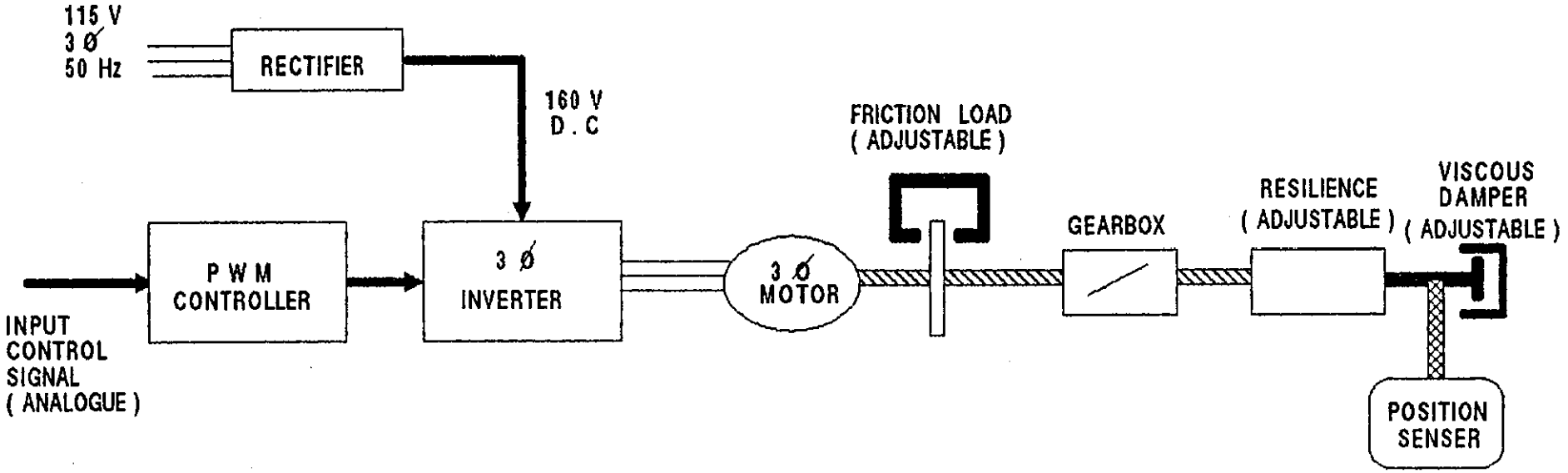


Fig.1.2

THE CONTROLLED PLANT

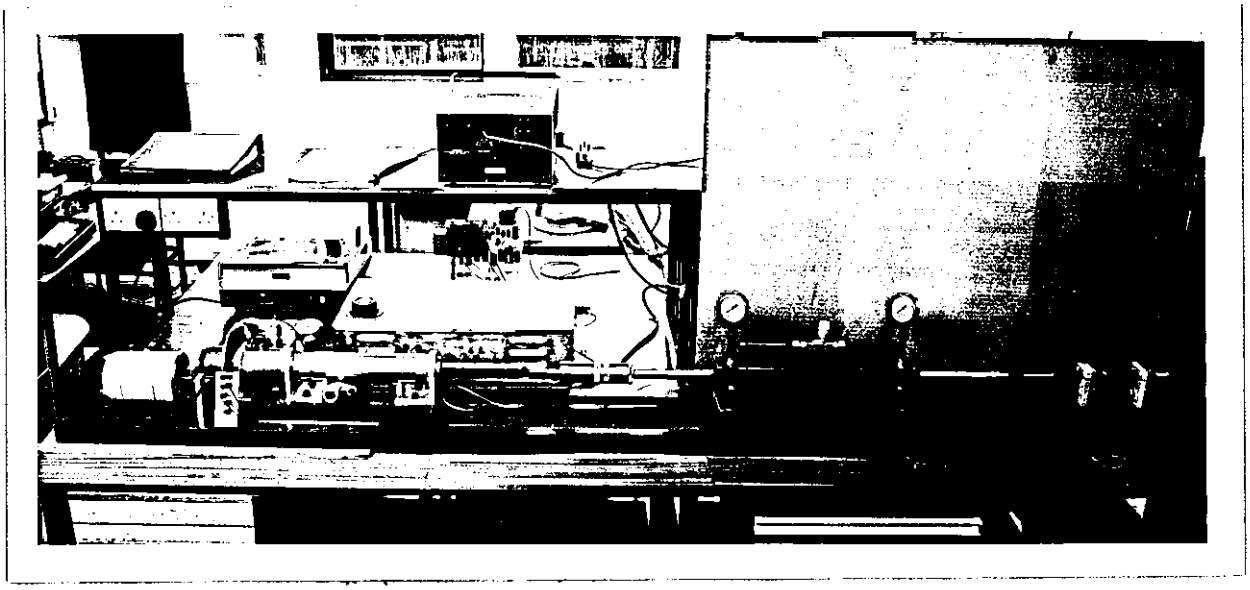


Fig . 1 . 3

The Test Rig

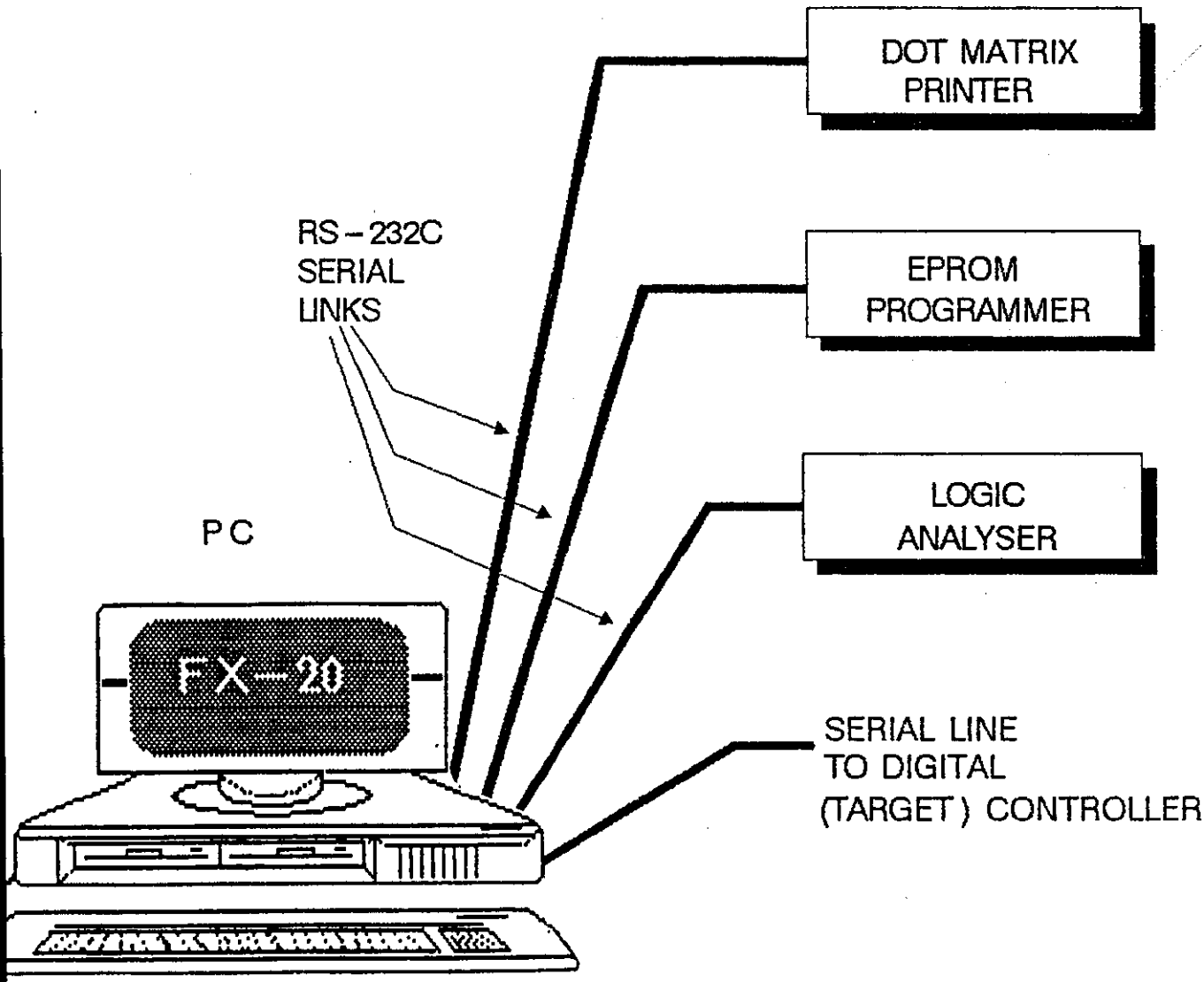


Fig.1.4

PC BASED INTEGRATED WORK – STATION

CHAPTER-2

2 THE DEVELOPMENT OF THE THEORY OF CONTROL

2.1 HISTORY OF CONTROL

Fig.2.1 shows some events in the history of control. The basic concepts of automatic control and their analysis in terms of ordinary differential equations were well established by the beginning of the twentieth century [2.1]. These techniques were consolidated in review papers by Hort [2.2] and Von Mises [2.3], and in early textbooks on automatic control by Tolle [2.4] and Trinks [2.5]. Important studies carried out by Minorsky [2.6], and by Hazen [2.7] produced further development in the field. Minorsky proposed the use of a proportional-plus-derivative-plus-integral control action in the automatic steering of ships. His work was of particular significance in being practically tested in a famous series of trials on the automatic steering of the USS New Mexico in 1922-23 [2.8]. Hazen's studies were on shaft-positioning servomechanisms. Both Minorsky's and Hazen's work was explained in terms of ordinary differential equations, and their success with practical devices led to the widespread use of this approach for the analysis of automatic control systems [2.1].

In 1936 Callender, Hartree, and Porter published the first paper [2.9] describing the application of the PID controller on an analogue computer [2.10]. Zieglar and Nichols [2.11] made an important study which led to simple rules for tuning the PID controller [2.22].

By the late 1930's there were thus two separate but well-developed methods of analysing feedback system behaviour [2.12]:

(a) The "time-response approach" which involved ordinary differential equations and their associated characteristic algebraic equations. This

approach was much used in mechanical, naval, aeronautical, and chemical engineering studies of automatic control systems; and

(b) the "frequency-response approach" which involved Nyquist and Bode plots, transfer functions, etc.. This approach was used in the studies of feedback amplifiers [2.12].

The frequency-response approach describes systems in terms of their input/output relationships in the frequency domain. In practice this proved to be a very flexible and general way of representing systems [2.1]. The use of transfer functions in analysing feedback systems was introduced by Harris [2.13]. The idea of Harris enabled a mechanical servomechanism or a chemical process control system to be represented in terms of a block diagram [2.12].

The demands of World War II greatly accelerated work in the field of automatic control. In particular the need for precise and rapid control of ships, aircraft, and radar antenna systems led to significant advances in theory and practice. With the lifting of wartime security restrictions in 1945 [2.10], rapid progress took place in the control field, many books and innumerable articles and papers being written in the post-war period. This has resulted in the widespread dissemination and adoption of frequency-response ideas [2.1] and the application of control systems in industrial and military fields almost without limit [2.10].

The development of frequency response techniques was soon followed by another approach to control system design [2.15]. This was introduced in 1948 by W. R. Evans, who was working in the field of guidance and control of aircraft. Many of his problems had unstable or neutrally stable dynamics, and he suggested a return to the study of the characteristic equation that had been the basis of work of Maxwell and Routh nearly 70 years earlier. However, Evans developed techniques and rules allowing one to follow graphically the paths of the roots of the

characteristic equation as a parameter was changed. His method, the root locus, is suitable for design as well as for stability analysis and remains an important technique today.

Until the beginning of the second world war work concentrated mainly on either linear systems or simple "bang-bang" controllers. However, the need arose at this time to control the performance of systems which provided "plant" information at, and only at, discrete time intervals. For instance, the rotating aerial of a radar system illuminates its target intermittently; hence data is available only in pulsed or sampled form. Therefore, many of the fire-control systems developed during the Second World War had to be designed to deal with data available in this form [2.1]. Hurewicz laid the basis for an effective treatment of sampled-data automatic control systems [2.14], extending the Nyquist stability criterion to sampled-data systems. Digital controllers operating on continuous-time plants required analysis techniques which relate both discrete-time and continuous-time systems. Linvill discussed this problem from the transfer point of view, including a consideration of the Nyquist approach to closed-loop stability [2.15]. Tsypkin used frequency-response methods to analyse sampled-data systems [2.16]. A "Z-transform" theory for systems described by difference equations emerged to match the "S-transform" theory for systems described by differential equations [2.17]. This theory was treated in textbooks by Ragazzini and Franklin [2.17], Jury [2.18], [2.19], Freeman [2.20], and others.

The effect of random disturbances on automatic control systems was also studied during the Second World War [2.14]. Wiener studied the relationship between the time-response and frequency-response descriptions of a stochastic process [2.21]. His books had the important effect of propagating feedback control ideas in general, and frequency-response methods in particular, into the field of stochastic system theory [2.1].

The emergence of the digital computer by the late 1950's as a widely availabe engineering device was a necessary prerequisite for the next developments in automatic control systems. The computing power and versatility of the big scientific machines made the lengthy and intricate calculations involved a practicable proposition [2.22]. During the decade of the 1950's several authors including Bellman, Kalman, and Pontryagin began again to consider the ordinary differential equations (ODE) as a model for control systems [2.10]. The new methods considered the simultaneous control of a number of interacting variables; these also assessed the use of different types of controller objectives, such as the minimisation of fuel consumption. As with the previous major developments in automatic control theory, the next advances arose out of an important technical problem, in this case the launching, manoeuvering, guidance, and tracking of space vehicles [2.1]. This development was supported by digital computers, which could be used to carry out calculations that would have been unthinkable 10 years earlier. The study of optimal controls begun and much of this work was presented at the first conference of the newly formed International Federation of Automatic Control (IFAC) held in Moscow in 1960. This work did not use the frequency response or the characteristic equation but instead worked directly with the ordinary differential equation. Generally such methods required the extensive use of computers [2.10]. This approach is now often called "modern control", as opposed to the methods of Bode et. al, which are termed "classical".

In the late 1950's, the control world became deeply involved in so called adaptive systems [2.23]. As Truxal noted [2.24] these are nonlinear feedback systems derived from an identification viewpoint. In the 1950's and early 1960's, such advanced techniques were seen as a means to substantially improve control performance in parameter varying systems. Unfortunately on-line digital computers with sufficient speed and computation capability were not available at

that time [2.25]. Analogue computers of that time period did not have either adequate accuracy or reliability to perform in the adaptive control mode [2.25]. Adaptive control has been a challenge to control engineers for a long time, many schemes having been proposed [2.26]. In spite of this, progress in the field has been comparatively slow. One reason is that it is difficult to understand how adaptive systems work because they are inherently nonlinear. Another reason is that it has been costly and fairly complicated to implement adaptive controllers. Computer control was still out of reach for many control problems until the development of the microcomputer in 1970 [2.22]. The situation changed dramatically with the advent of powerful low-cost microprocessors, making the implementation of adaptive controllers feasible and economical. Recently there has also been significant progress in the theory of adaptive control [2.27].

2.2 APPROACHES TO ADAPTIVE CONTROL

During the past few years the field of adaptive control has become increasingly active. Major advances in microprocessor technology have also made sophisticated algorithms more feasible for practical applications.

Self-Tuning Controllers (STC) and Model Reference Adaptive Control (MRAC) are the two principal solutions to the adaptive control problem [2.28,2.29]. STC is derived for discrete time systems while MRAC is derived for deterministic continuous control systems [2.30].

In MRAC the objective is to force the response of the closed loop control system to follow that generated by some defined model. In STC a design procedure for known plant parameters is first chosen and this is applied to the unknown plant using recursively estimated values of these parameters.

Figs.2.2 and 2.3 represent the adaptive control problem using MRAC and

STC approaches respectively. In MRAC (Fig.2.2) the input and the output of a linear plant are u and y_p respectively. A linear model and reference input r are specified which result in a model output y_m. From all the available data (u, y_p, r and y_m) it is desired to determine a control input (u) such that the error (e) between y_p and y_m tends to zero asymtotically [2.31].

In the STC approach (Fig.2.3) the first step is the selection of an appropriate known procedure for the design of a controller when the plant parameters are known. The second step is to estimate the unknown plant parameters; from these the control algorithm coefficients are modified (up-dated) in accordance with the defined control criterion [2.27]. Identification carried out continuously by the controller as the plant runs is known as "on-line" identification. Off-line identification is done on recorded data by transfering this data to a host computer where it is processed.

According to Clarke [2.27] a "good" adaptive controller should be characterised by:

* Closed loop performance criteria easily understood by control engineers.
* Simplicity of coding.
* Robustness when applied to as broad a class of plants as possible.

In MRAC the zeros of the plant must lie in the interior unit circle (i.e. a non-minimum phase plant is not allowed) [2.29]. The general approach of MRAC yields a relatively complicated adaptive control law [2.44] while STC is, in a sense, the simplest possible adaptive control algorithm [2.30]. STC can use "conventional" control techniques that are robust and well known. The selection of an appropriate control scheme removes the restriction on the type of plant in STC.

Although STC and MRAC techniques are based on different design

principles, it has recently been shown [2.32,2.33,2.34,2.35] that both control schemes are very similar and in some special cases even identical [2.35].

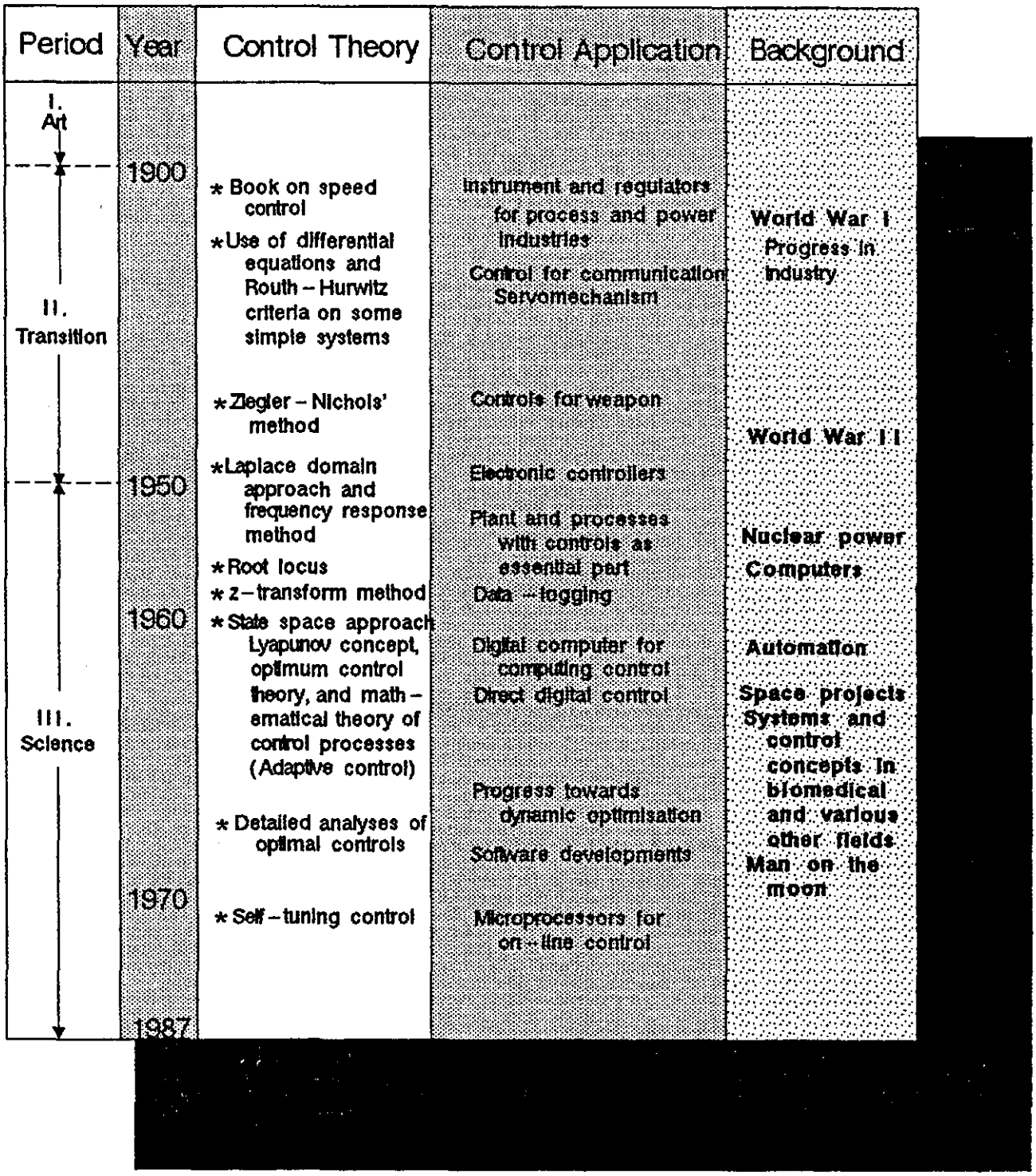


Fig.2.1

Some events in the history of control

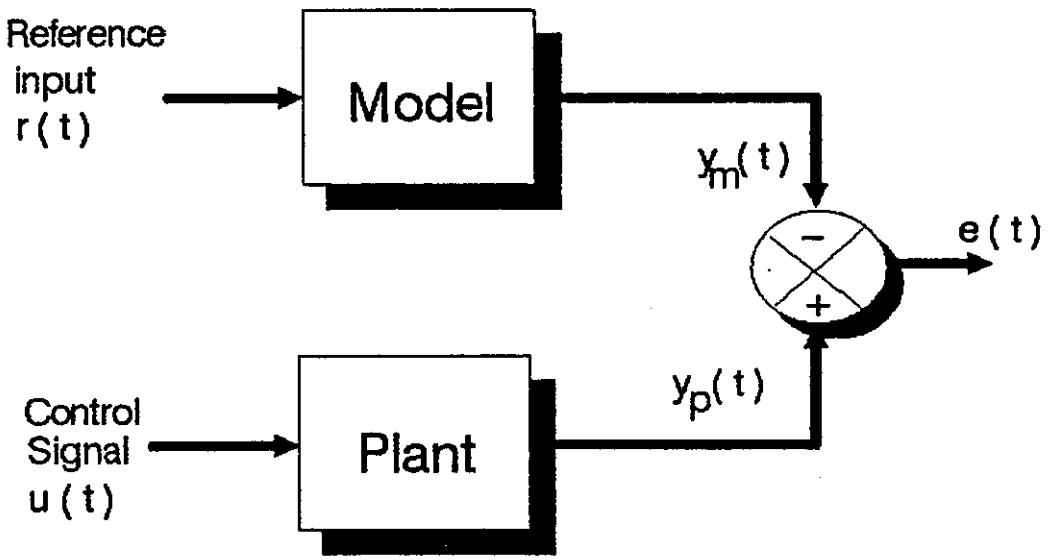


Fig.2.2

Model Reference Adaptive Controller

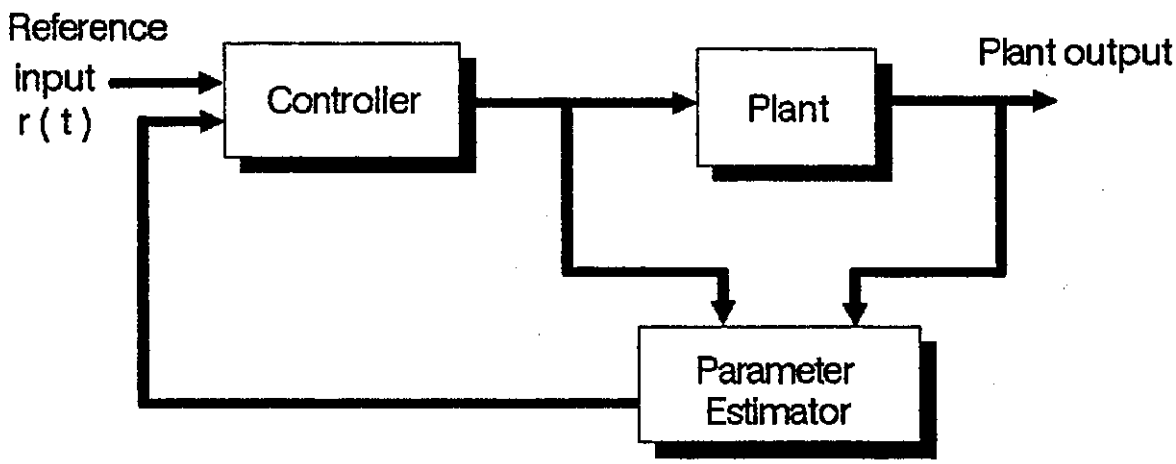


Fig.2.3

Self – Tuning Adaptive Controller

CHAPTER-3

3 MODEL DETERMINATION

3.1 INTRODUCTION

A model may be defined as "a representation of the essential aspects of a system which presents knowledge of that system in a usable manner" [3.8]. In order to design a controller for a dynamic system it is necessary in many cases to develop a model that adequately describes the system's behaviour. The mathematical model forms an important part of the design cycle, engineers thereby gaining an understanding of the nature of the dynamic behaviour of the system [3.1].

For a model to be useful it must not be so complicated that it cannot be understood and thereby be unsuitable for analysis; at the same time it must not be oversimplified to the extent that predictions of the behaviour of the system based on this model are inadequate [3.8].

The system model is usually represented by a transfer function, shown in the form of a block diagram, Fig.3.1. Note that the block is "unidirectional". The input may be regarded as the "cause" and the output as the "effect". The block diagram is unidirectional since the "effect" cannot produce the "cause" [3.3]. The transfer function G(s) relates the Laplace transform Y(s) of the output y(t) to the Laplace transform U(s) of the input u(t) through the relationship

$$Y(s) = G(s)U(s) \qquad (3.1)$$

This chapter describes the methods used to obtain a mathematical model of the controlled plant. These are based on established statistical test methods, using an assumed high order model in the identification process. Once the "best" estimate of the plant model is obtained its order is reduced to a minimum acceptable

level.

The model order reduction technique developed here, although based on that proposed by Soderstrom [3.9], offers significant improvements on earlier methods; furthermore several concepts involved are new and original. Its effectiveness is demonstrated in a series of practical tests, leading to fast identification combined with satisfactory closed-loop control.

3.2 APPROACHES TO MODEL DETERMINATION

There are two approaches to the problem of determining a mathematical model of a given system [3.1].

(a) Physics: Many systems can be analysed in terms of laws of physics, thermodynamics etc. and expressed in mathematical terms by looking directly at the mechanism that generates signals within the system. Equations are then assembled to form a mathematical model, based on the physical laws and relationships that govern the system's behaviour [3.3]. An advantage of model-building from physics is that the variables have physical interpretations [3.1].

(b) Identification: System identification is the experimental approach to plant modelling in which data obtained from the system is used to model that system [3.5]. Many methods are suitable for analysing data obtained from such experiments [3.2,3.8] [Appendix-B], one basic approach being the principle of Least Squares (LS) [Appendix-B].

In many situations direct modelling using physical knowledge may not be possible [3.6]. One reason may be that knowledge of the system's mechanism is incomplete, since this method needs insight. A further reason is that the system may be subject to on-line changes in an unpredictable manner, as occurs when the environment of the system changes (e.g. an aircraft changes altitude, a paper machine is given a different composition of fibre, etc.). The first approach can also be quite time-consuming, furthermore it may lead to models that are complex and of high order which in turn require order reduction [3.1]. In these circumstances the designer may turn to an identification method. These methods are also especially

useful when plant dynamics change with time or with operating conditions. On these occasions the control parameters need to be changed to "tune" the controller [3.4] for best performance. In order to achieve this it is necessary to obtain a plant model under the new conditions. Deriving this from experimental data is often the most effective (if not the only) way to do this.

3.3 FUNCTIONAL TESTING AND SYSTEM IDENTIFICATION

3.3.1 Overview

The ultimate objective of this test is to produce a mathematical model of the plant based on measurements of the plant input (control) and output (measured value) signals. Therefore it is necessary for the experimenter to control the test procedures, setting conditions such as signal type and duration, sampling rates and number of measurement points. The set-up used for these tests is shown in Fig.3.2, the use of the PC being self-evident from the following text. Note there are two distinct aspects of this operation. In the first place the plant has to be perturbed to obtain data; subsequently the information so obtained is used as part of the model identification process. This is carried out within the PC.

3.3.2 Plant testing

The control program which actually runs the plant and collects data measurements sits within the target controller in Eprom. For this, the source code is written in MT+ Pascal, details being given the software development chapter. Program development, i.e. code writing, compilation, linkage and PROM blowing,

takes place within the PC environment. For the duration of the plant test the PC functions as a terminal having data storage facilities. It communicates with the target controller using its RS232 serial data line.

To start the test procedure the controller and plant are powered up. The PC must be connected to the controller and placed in terminal mode. Instructions from the controller to the operator are displayed on the PC screen, responses being entered at the keyboard. These include plant test data and designation of the PC data file (on floppy disk) which is to be used to hold the plant measurements. A typical test procedure is shown in Fig.3.3 where, once the test parameters are entered by the operator, the controller runs the plant through its test sequence. During this, measurements are made of control signal and measured value, these being stored within the controller memory. At the end of the test this data is transferred to the named disk file in the PC. If required, the information can also be printed out for evaluation and review.

3.3.3 Off-line identification

The purpose of the off-line identification process is twofold. First, using the data recorded during the test run, it enables a mathematical model of the plant to be generated. Second, using this same set of data, a number of models can be obtained by applying various identification schemes. Details of several identification schemes are given in Appendix-B.

Evaluation of several identification schemes was carried out, including Recursive Least Squares (RLS), Recursive Extended Least Squares (RELS), Recursive Maximum Likelihood (RML) and Recursive Instrumental Variable (RIV) methods [Appendix-B]. The process of plant identification can be better understood by considering one of these in more details, RLS being a suitable

candidate.

Fig.3.4 illustrates the concept involved in identification and model generation using RLS techniques. Here a program running under the PC operating system performs the following actions;

(a) Set up a preliminary (guessed) model of the plant using information supplied by the operator.

(b) Read the recorded plant control signal at a specific sample instant, apply this to the model, and calculate the resulting output.

(c) Read the recorded plant measured value at the same sample instant and calculate the error between this and the model output.

(d) Compute the average error power.

(e) Adjust the model parameters to reduce the absolute value of the power gradient.

(f) Repeat the above steps (b) to (e) for all recorded values, working iteratively towards a condition of a minimum error power.

As this program is executed sample by sample the model parameters (hopefully) converge toward those of the plant itself. What we are left with is a best estimate of the plant transfer function, this information being used in the implementation of an appropriate control scheme. Also, the results may be stored at each calculation interval, thus allowing the experimenter to review the convergence rate and accuracy of the identification process. It is, of course, necessary to design and write the identification program in the first case. This process is covered under software development (chapter.9).

The identification process is repeated to obtain models of different orders. These are then adjusted until a minimum order model is obtained which adequately

represents the plant. Details of the model order reduction process are given in the following section.

3.4 MODEL ORDER REDUCTION

3.4.1 Overview

In a self-tuning controller the time required to execute the identification scheme is very significant and is usually the limiting factor on system sample rates. As the order of the assumed model increases the execution time rapidly increases (as shown in Table.3.1). A balance needs to be struck between the order of a transfer function and its effect on control system performance. Increasing the order of a model may produce only a marginal change in closed-loop performance; it may, though, produce a major increase in the time taken to run the identification process. What is needed is a minimum order model that adequately represents the plant for control purposes.

The objective here is to investigate the performance of lower order plant transfer functions derived from the transfer function produced by the identification process. By comparing this with the model derived from frequency response testing of the plant the relative performance of the models can be assessed. This test can be repeated for models of various orders.

3.4.2 The effect of choosing different model orders

If the order of the assumed model in the identification scheme is higher than the order of the plant, the following will occur:

(a) Zero coefficients will occur in the numerator polynomial of the transfer function when it is in the rational expansion form in the s-domain. This will take place until the ratio of the numerator to the denominator of the assumed model is equal to the ratio of the numerator to the denominator of the plant. Then,

(b) Pole-Zero cancellation will take place until the assumed model matches the plant. This is very clear when the model is in factored form.

Therefore from the theoretical point of view the choice of a higher model order does not affect the estimated model since it will match the real plant as simulated in Appendix-E.

In practice this does not occur, but instead negligible, rather than zero, coefficients appear in the numerator and poles and zeros that should cancel are not exactly equal. This is due to the fact that there are different noise sources in the system (e.g. quantisation effect).

Numerous model order reduction techniques have been proposed in the literature, these being reviewed in [3.10]. In general these techniques have different objectives. Model reduction using pole-zero cancellation is the aim of our tests; hence the pole-zero cancellation method proposed by Soderstrom [3.9] is used.

3.4.3 The Pole-Zero cancellation technique

(a) The problem formulation

The objective of this method is to test possible pole-zero cancellations in order to reduce a model order. The identification technique (e.g. Recursive Least Squares (RLS)) used in section 3.3.3 forms the basis for this method.

The problem that is considered can be formulated as follows: The two polynomials that form the plant transfer function (A/B) are

$$A(z^{-1}) = a_0 + a_1 z^{-1} + \ldots a_{na} z^{-na}$$

$$B(z^{-1}) = 1 + b_1 z^{-1} + \ldots b_{nb} z^{-nb}$$

where the values of parameters are those estimated in the previous test (section 3.3.3). To account for the uncertainty of the parameters estimated the covariance matrix (P) of these estimates produced from the identification technique is used. The problem now is to test whether the polynomials A and B have common factors or not. The test is carried out for N common factors by starting with N=1, repeat the test for N=2,3,..., etc. as long as common factors are found.

(b) The criterion

Let the n-dimensional vector $\underline{\hat{x}}$ consist of the estimated values of the coefficients of the polynomials $\hat{A}$ and $\hat{B}$. Introduce a vector $\underline{x}$ that has the same dimension as $\underline{\hat{x}}$ corresponding to two polynomials A and B. The problem now is to look for a vector $\underline{x}$ in the same domain of $\underline{\hat{x}}$ such that the corresponding polynomials (A and B) have at least N common factors.

The technique achieves this by minimising a cost function $J(\underline{x})$ of the form

$$J(\underline{x}) = (\underline{x} - \underline{\hat{x}})^T P^{-1} (\underline{x} - \underline{\hat{x}}) \qquad (3.1)$$

Notice that the error criterion (Eq.3.1) uses P^{-1} as a weighting matrix. Each element in the covariance (P) matrix reflects the uncertainty of each estimate; a large uncertainty means large variance which in turn implies that the corresponding element in the P matrix is large. Therefore P^{-1} gives approximate relative weighting in the error criterion (Eq.3.1) because in P^{-1} large variance generates small weighting on the corresponding error.

(c) The algorithm

Introduce the polynomials $\tilde{A}(z^{-1})$, $\tilde{B}(z^{-1})$ and $\tilde{C}(z^{-1})$ where

$$\tilde{A}(z^{-1}) = \tilde{a}_0 + \tilde{a}_1 z^{-1} + \ldots + \tilde{a}_{na-N} z^{-(na-N)}$$

$$\tilde{B}(z^{-1}) = b_0 + \tilde{b}_1 z^{-1} + \ldots + \tilde{b}_{nb-N} z^{-(nb-N)}$$

$$\tilde{C}(z^{-1}) = 1 + \tilde{c}_1 + \ldots + \tilde{c}_N z^{-N}$$

where N is the number of common factors. Consider the polynomials $A(z^{-1})$ and $B(z^{-1})$ have the following form

$$A(z^{-1}) = \tilde{A}(z^{-1}) \tilde{C}(z^{-1}) ,$$

$$B(z^{-1}) = \tilde{B}(z^{-1}) \tilde{C}(z^{-1}).$$

Thus the polynomial $\tilde{C}(z^{-1})$ represents the pole/zero cancellation factors.

The coefficients of these three polynomials ($\tilde{A}$, $\tilde{B}$, and $\tilde{C}$) are collected in a vector $\underline{y}$. Thus $\underline{x}$ can be written as a function of $\underline{y}$, $\underline{x} = f(\underline{y})$. Then the optimisation problem is to find the global minimum without constraints of

$$V[f(\underline{y})] = [f(\underline{y}) - \hat{\underline{x}}]^T P^{-1} [f(\underline{y}) - \hat{\underline{x}}]$$

The resulting coefficients of $\tilde{A}$ and $\tilde{B}$ represent the coefficients of the reduced model while the coefficients of $\tilde{C}$ represent the common factors in the numerator and the denominator of the original model.

This is a non-linear optimisation problem which may yield several minima,

therefore the selection of the initial values of the $\underline{y}$ vector is important. A reasonable set of initial values can be found by using the values of the original model. This is done by looking at the poles and zeros of the original transfer function and using the ones that are close to each other as the initial value for the C polynomial while the rest of the coefficients are used as initial values for A and B polynomials.

Several methods are available to solve a non-linear optimisation problem [3.12]. The quasi-Newton method is one of the powerful techniques used in such optimisation problems and is available in the NAG library on the Honeywell MULTICS main frame computer. For these reasons it is used to handle the optimisation process involved in model order reduction.

3.4.4 Static gain correction

The static gain of a discrete transfer function is simply its value when z is set equal to 1. Looking at a typical transfer function produced from the optimisation for one common factor

$$\frac{(\tilde{a}_0 + \tilde{a}_1 z^{-1} + \ldots + \tilde{a}_{na-1} z^{-(na-1)})(1+ \tilde{c}_1 z^{-1})}{(1 + \tilde{b}_1 z^{-1} + \ldots + \tilde{b}_{nb-1} z^{-(nb-1)})(1+ \tilde{c}_1 z^{-1})}$$

There is one too many degree of freedom in the $\tilde{A}$ polynomial. Therefore the optimisation may yield numerator estimates which are incorrect by a scaling factor and hence give rise to a static gain difference between the original model and the reduced model. This was found in practice as will be shown in Chapter 8.

Several solutions to this problem are introduced:

(a) Using the above method as proposed by Soderstrom, then multiplying the

reduced order model by a scaling factor. The resultant model has the following form

$$G_{new}(z^{-1}) = \frac{G_o}{G_r} G_r(z^{-1})$$

where $G_{new}(z^{-1})$ is the new model, G_o is the static gain of the original model, and G_r is the static gain of the reduced order model using the Soderstrom method.

(b) Modifying the algorithm to retain the static gain during the optimisation procedure by forcing $\tilde{b}_1$ to have the value

$$\tilde{b}_1 = \frac{1}{G_o} \sum_{i=0}^{m} \tilde{a}_i - \sum_{j=2}^{n} \tilde{b}_j - 1$$

where G_o is the static gain of the original model.

(c) Retaining the static gain during the optimisation procedure by forcing $\tilde{a}_0$ to have the following value

$$\tilde{a}_0 = G_o \sum_{i=0}^{n} \tilde{b}_i - \sum_{j=1}^{m} \tilde{a}_j$$

instead of $\tilde{b}_1$.

(d) Produce a reduced order model using the Soderstrom method and then amend the value of either $\tilde{a}_0$ or $\tilde{b}_1$ so that the static gain is equal to the original model.

3.4.5 Model trimming

The method proposed by Soderstrom tests for pole-zero cancellation but does not test for negligible numerator coefficients that could appear if the order of the model produced from system identification is over estimated as mentioned in section 3.4.2. Therefore the structure of the continuous reduced order model may not match the actual model of the continuous plant. It is shown in Appendix-E that it is not possible to eliminate these coefficients if the model is in the Z-domain since they contribute to each coefficient, but if the model is in the S-domain these small coefficients appear as separate coefficients in the numerator.

A technique is introduced to test for negligible coefficients after performing model order reduction. This trimming method uses the same error criterion (Eq.3.1) used above and the quasi-Newton method is used as the optimisation method. Details are given in Appendix-E.

3.4.6 Model order reduction technique

Fig.3.5 shows the procedure to produce a reduced order model, these being

* Reduce the model order using the improved method a, b, c or d (Section 3.4.5)
* Transform the model to the S-domain.
* Trim the model using model trimming technique (Section 3.4.6)
* Transform the model back to the Z-domain.

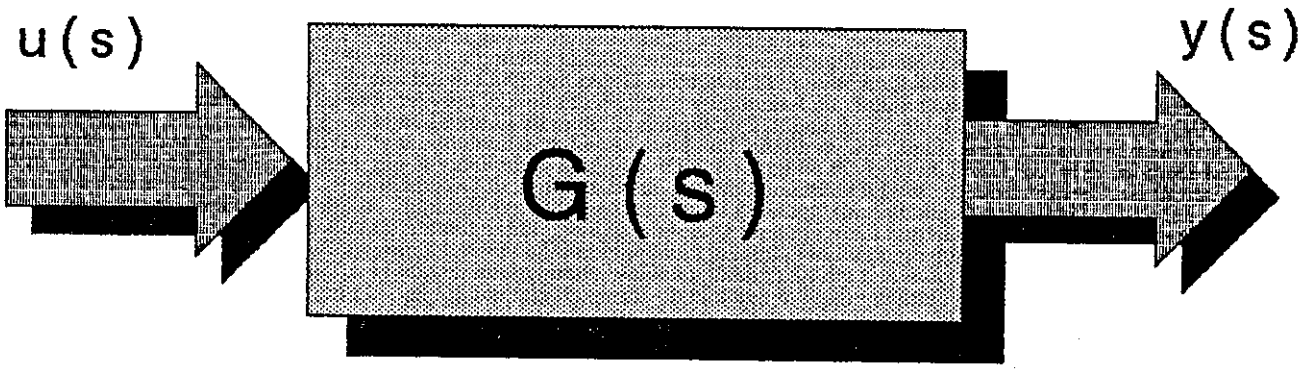


Fig . 3 . 1

THE TRANSFER FUNCTION BLOCK DIAGRAM

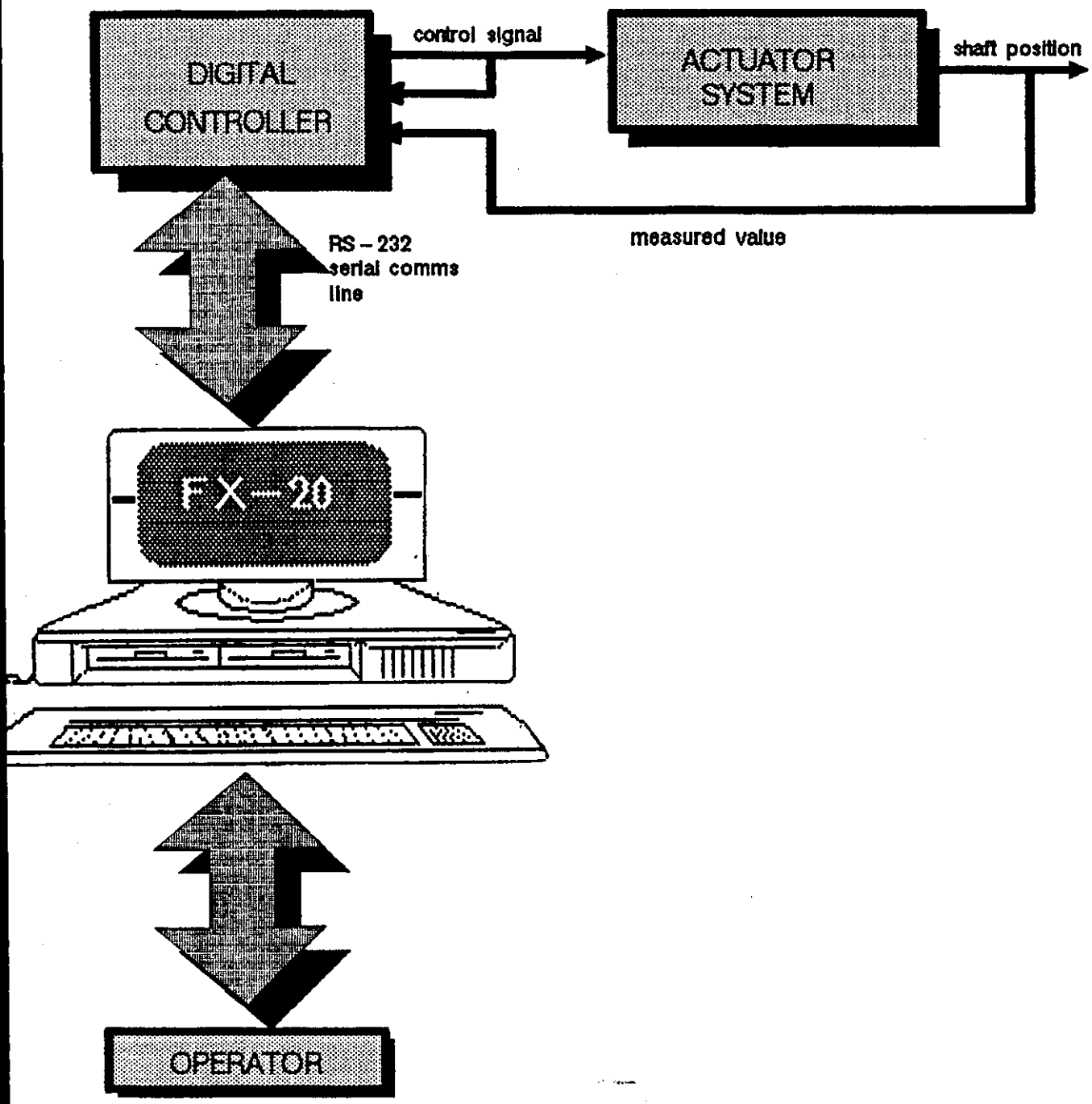


Fig . 3 . 2

SYSTEM ORGANISATION – FUNCTIONAL TESTING

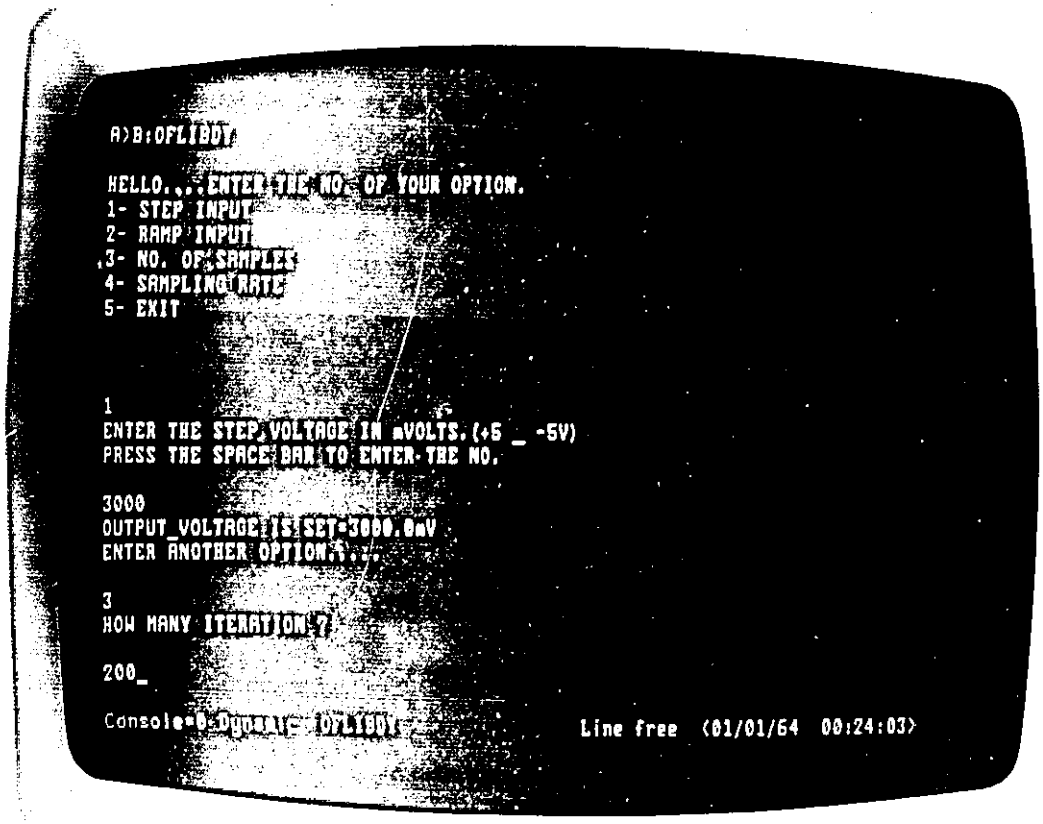


Fig . 3 .3

A TYPICAL TEST SEQUENCE

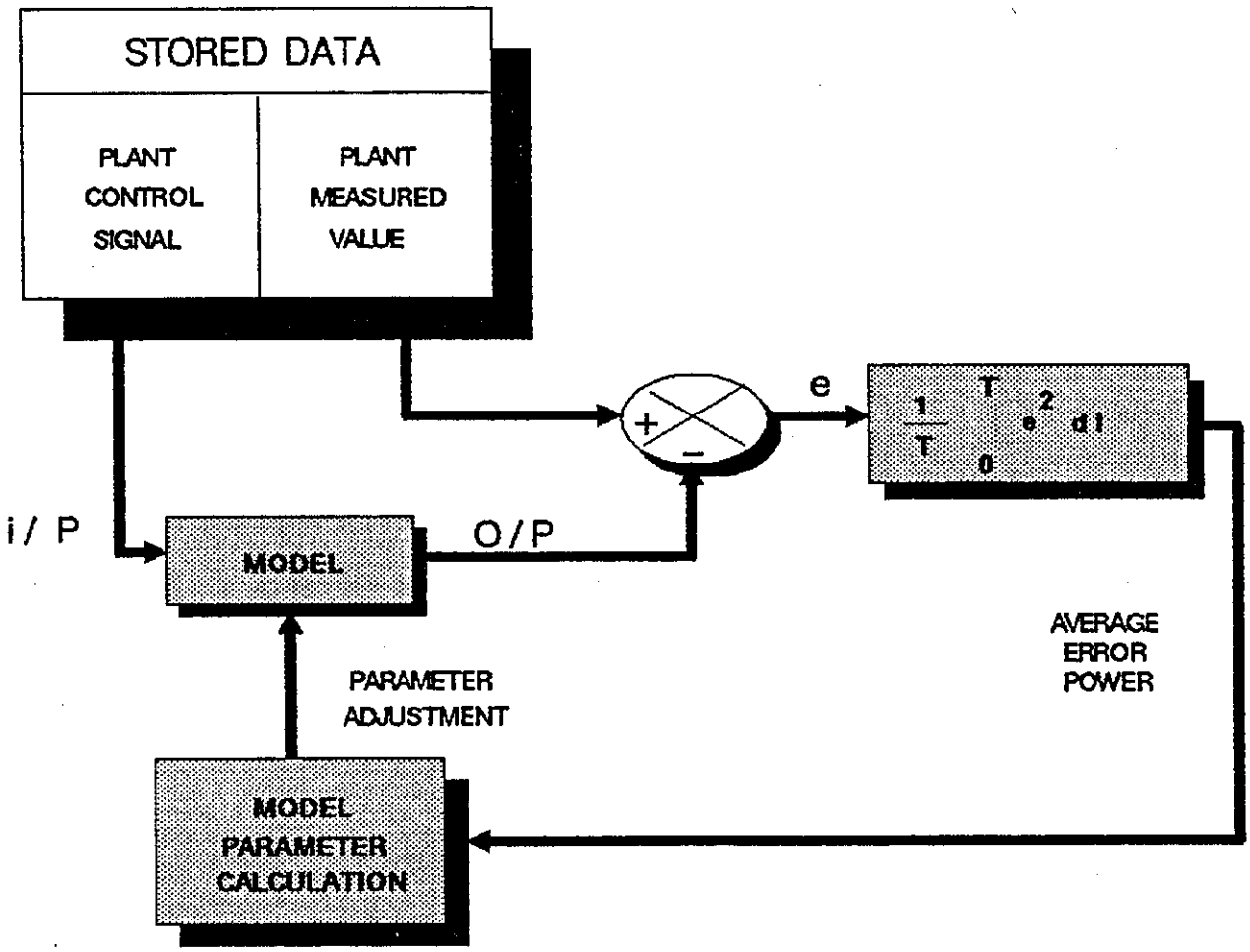


Fig . 3 . 4

PLANT OFF – LINE IDENTIFICATION PROCESS

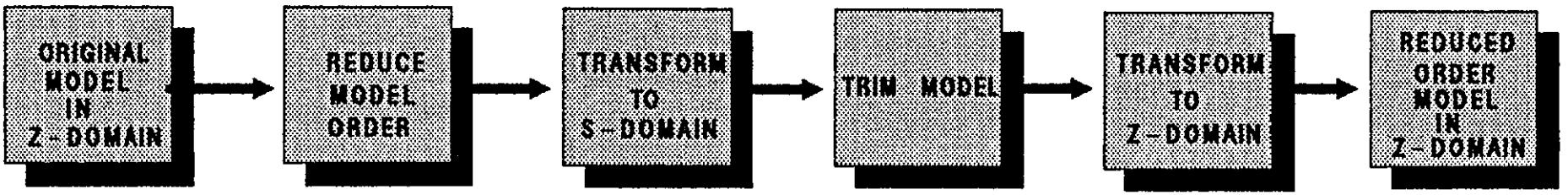
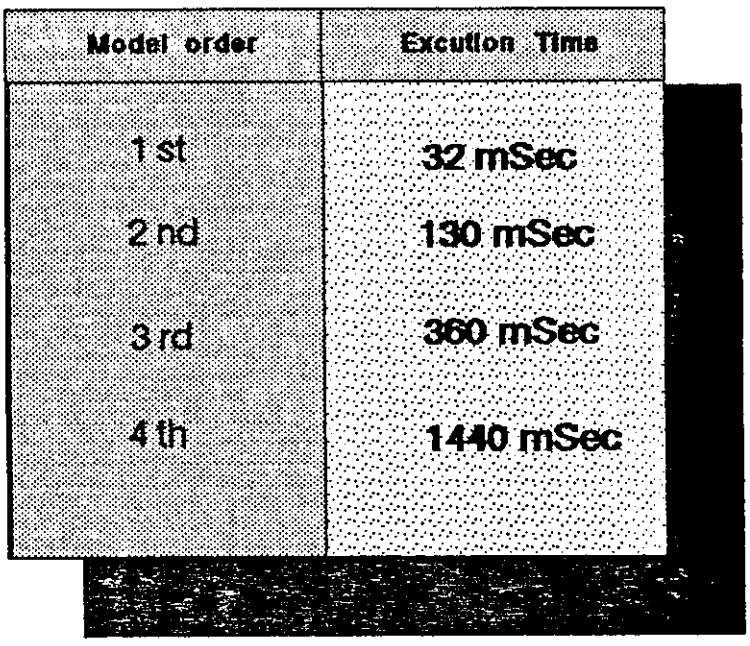


Fig . 3 . 5

MODEL ORDER REDUCTION SEQUENCE



Model order	Excution Time
1 st	32 mSec
2 nd	130 mSec
3 rd	360 mSec
4 th	1440 mSec

Table . 3 . 1

Time Required To Excute System Identification Procedure

CHAPTER-4

4 IMPLEMENTATION OF THE DIGITAL CONTROL ALGORITHM

4.1 INTRODUCTION

Numerous control algorithms have been proposed for use in digital controllers [4.1,4.2,4.3]. In this chapter a number of algorithms are implemented in order to evaluate their effectiveness in controlling the plant. The ones evaluated here are those which have (or appear to have) given satisfactory results in practical situations. These algorithms are related to the performance specification methods, being expressed either in statistical or deterministic form. However performance effectiveness is not the only criterion for evaluation. In an on-line adaptive controller for servo applications the control action, and especially adaptation of the algorithms, must be carried out quickly. Thus simple algorithms which require only a small computing effort are desirable.

Once the performance specification (criterion) and control algorithm are defined then the algorithm coefficients are adjusted until the closed loop performance specification is met. However the loop includes the plant; hence its dynamics must be taken into account when computing the coefficient values. This requires us to evaluate the plant transfer function in discrete form (i.e. to derive its pulse transfer function) before calculating the controller settings. Finally practical considerations involving system non-linearities and quantisation effects have to be considered before actually running the control loop.

The performance actually achieved depends to some extent on the controller hardware and software. In this case a 16-bit microprocessor with 12-bit Analogue to Digital and Digital to Analogue converters is used; the software is written in PASCAL, calculations being carried out in floating-point form.

4.2 PULSE TRANSFER FUNCTION

Fig.4.1 shows the system diagram of the microprocessor controller together with the plant. The analogue and the digital parts of the system are connected via D/A and A/D converters respectively.

To develop a form of a sampled system as shown in Fig.4.2 the system is depicted as follows. Each A/D converter is represented by an ideal sampler [4.4]. Each D/A converter is represented by a sampler followed by Zero Order Hold (ZOH) circuit having the following transfer function [4.5]:

$$h_{zoh} = \frac{1}{s}\left(1 - e^{-sh}\right) \qquad (4.1)$$

where h is the sample time.

The plant is modelled by its linear transfer function, which, in this case is

$$G(s) = \frac{k}{\tau s} \qquad (4.2)$$

and the calculations in the microprocessor are expressed by their pulse transfer function H(z) as shown in Fig.4.2 [4.2].

Fig.4.3 is obtained by first combining the analogue parts (the hold, sampler and plant [4.3,4.4,4.6]) to give a combined transfer function of;

$$G(s) = \frac{1}{s}\left(1 - e^{-sh}\right) G(s) \qquad (4.3)$$

The discrete equivalent of this is obtained using invariant impulse transform tables [4.5]:

$$G(z) = \left(1 - z^{-1}\right) Z[G(s)/s]$$

$$G(z) = \frac{kh\ z^{-1}}{\tau(1 - z^{-1})}$$

where h is the sampling time.

In more general terms this transfer function can be written as:

$$G(z) = \frac{a_1\ z^{-1}}{(1 + b_1\ z^{-1})} \qquad (4.4)$$

where $a_1 = kh/\tau$, and $b_1 = -1$

This method of representing the continuous plant as a discrete model is used for the following reason: If at sampling instants,as shown in Fig.4.4, E is zero then the continuous and discrete models are identical (note this does restrict us to the sample instants). Using the impulse invariant transform technique the resulting discrete model has a pulse response which is identical to that of the impulse response of the continuous model.

4.3 THE CONTROL LAW

Fig.4.5 and Fig.4.6 show the most general form of a system controller, F, S and 1/R representing individual processing functions. Any closed loop controller can be represented by this structure. For instance with F=1, S=1, and R=0.01 the unit is a simple proportional (P) controller having a gain of 100.

Note that the symbology (F,S,1/R) is used for mathematical convenience only and has no other significance. From this structure the control law can be written in a general form [4.3] as:

$$\text{Loop error} = F(z)w(t) - S(z)y(t)$$

i.e. $$R(z)u(t) = F(z)w(t) - S(z)y(t) \qquad (4.5a)$$

or

$$u(t) = \frac{F}{R}\,w(t) - \frac{S}{R}\,y(t) \qquad (4.5b)$$

where R, F and S are polynomials in the Z domain.

The control law (4.5) represents a combination of a feedforward from the command signal w with the pulse-transfer function [4.2]:

$$G_{ff} = \frac{F}{R}$$

and a feedback from the measured output y with the pulse transfer function:

$$G_{fb} = \frac{S}{R}$$

For a digital controller G_c to be physically realisable, the power series expansion of the controller transfer function G_c must not contain any positive power in z [4.10]. Any positive power in z in the series expansion of G_c indicates that the

output precedes the input to G_c. This cannot be implemented since it requires a knowledge of future values of the input to the controller [4.11]. Therefore, the feedforward and the feedback transfer functions should satisfy the causality conditions [4.2]:

$$\deg R \geq \deg F$$

$$\deg R \geq \deg S$$

4.4 IMPLEMENTATION CONSIDERATIONS

4.4.1 Introduction

When implementing a digital controller the following are considered:

* Choice of sampling rate
* Computational delay
* Practical constraints
* Numerical accuracy

4.4.2 Choice of sampling rate

The selection of the best sample rate for a digital control system is the compromise of many factors. The basic motivation to lower the sample rate ω_s is cost. A decrease in rate means more time is available for the control calculations; hence slower microprocessors can be used for a given control function and the cost per function is lowered. Alternatively the original processor can perform extra tasks above that of controlling the plant. For systems with A/D converters, less demand on conversion speed will also lower cost. These economical arguments indicate that the best engineering choice is the slowest possible sample rate that still meets all performance specifications [4.7].

Factors that provide a lower limit to the acceptable sample rate are:

* Tracking effectiveness
* Disturbance rejection
* Prefilter design

(a) Tracking effectiveness

This is defined in terms of closed-loop bandwidth or transient time response performance, such as rise time and settling time. The lowest bound to the sample rate is set by a specification to track a command input with certain frequency (the system bandwidth) [4.7]. The sampling theorem states that in order to reconstruct a continuous signal from samples of that signal, one must sample at least twice as fast as the highest frequency contained in the signal [4.8].

This is likely to be significantly higher than the closed loop bandwidth. In practice test results show that this theoretical lower bound of sampling is not sufficient in terms of the quality of the desired time responses, and a sample rate of 4-20 times ω_{BW} is reasonable [4.4]. This is done mainly to reduce the delay between the set-point and the system response to that set-point change [4.7]. Furthermore it smooths the system output response to the control steps coming out of the DAC since, as sampling rates are increased, the step change in DAC output amplitude reduces. Consequently control action becomes less abrupt and control roughness is reduced at high sampling rates [4.4].

It is important to distinguish between closed-loop bandwidth and the highest frequencies present in the open-loop system. In the majority of cases closed-loop bandwidths are significantly higher than their corresponding open-loop values. i.e.

$$O.L.T.F = \frac{1}{1 + \tau s} = G(s)$$

For G_c=10 as shown in Fig.4.7, then

$$C.L.T.F = \frac{G_cG(s)}{1 + G_cG(s)} = \frac{10/1+\tau s}{1 + 10/1+\tau s}$$

$$= \frac{10}{1 + \tau s + 10} = \frac{10}{11 + \tau s}$$

$$= \frac{10}{11} \quad \frac{1}{1 + \tau/11\ s}$$

This may, however, be complicated by resonance effects within the loop which have high natural frequencies.

The situation is further complicated if the controller is likely to experience high frequency electrical interference. Here sampling rates are determined by the input filter design (see paragraph (c)); the resulting performance aspects may lead to the use of multi-rate sampling techniques [4.3] in conjunction with digital low pass filtering of the input signal.

(b) Disturbance rejection

Disturbance rejection is an important aspect in control systems. If the sample rate is very fast compared with the frequencies contained in the noise disturbance, the noise rejection will be high [4.5]. However, if the sample time is very long compared with the characteristic frequencies of the noise, the response of the system to the noise is essentially the same as if there were no control at all [4.7].

Although the best choice of sample rate is dependent on the frequency characteristics of the noise and the degree to which random disturbance rejection is important to the quality of the controller, sample rate requirements of 10-20 times ω_{BW} are common [4.4,4.7].

(c) Prefilter design

Digital control systems with analogue sensors typically include an analogue prefilter between the sensor and the A/D converter as an antialiasing device. The prefilters are low-pass, typically 3rd to 6th order filters being used in fast systems [4.9]. The simplest transfer function is [4.7]:

$$H_P(s) = \frac{a}{s + a}$$

so that the noise above the prefilter breakpoint a is attenuated [4.10]. The design goal is to provide enough attenuation at half the sample rate ($\omega_s/2$) so that the noise above ω_s, when aliased into lower frequencies by the sampler, will not be detrimental to the control system performance [4.4]. A conservative design procedure is to select the breakpoint and ω_s sufficiently higher than the system bandwidth so that the phase lag from the prefilter does not significantly alter the system stability. Thus the prefilter can be ignored in the basic control system design [4.7]. Furthermore, for a good reduction in the high frequency noise at $\omega_s/2$, the sample rate is selected about 5 or 10 times higher than the prefilter breakpoint [4.7]. This selection is related to the order of the filter [4.9]. The implication of this prefilter design procedure is that the sample rates need to be of the order of 20 to 100 times faster than the system bandwidth. This means that the prefilter specification determines the lower bound of the sample rate.

△ The rise time of the actuator system is approximately 4.3 Sec. A sampling time of 0.1 Sec, roughly 40 times faster than the rise time, is used in the controller.

4.4.3 Computational delay

There is always time delay when implementing a control algorithm using

microprocessors [4.10]. This delay is due to A/D and D/A conversion times and the time required to carry out the control calculations. The A/D and D/A conversion time is usually negligible (e.g. 25 uSec.), the significant time being the computational time. This is called the computational delay which is determined mainly by how the control algorithm is implemented. There are two ways to implement the control algorithm as shown in Fig.4.8 and Fig.4.9 [4.11].

In Fig.4.8 the measured value is read at time t and used to compute the control signal which is then applied at time t+1. This means that there is a time delay in the controller of one sample interval. In Fig.4.9 the measured value is read at time t, the control signal computed immediately, and the result output via the D/A converter as soon as possible. This means that the control signal is delayed only a fraction of the sample time [4.11].

The advantage of the second case is that the computation time can be ignored if it is very small. This delay is variable, depending upon the programming and mathematical floating point operations [4.5]. In many situations this variability can be ignored. However if it is significant when compared with the sampling time it may cause problems in the control loop [4.3]. The disadvantage of the first case is that the control signal is delayed one sample, but the advantage is that, as the delay time is constant, it is a straightforward to include one time-delay in the controller.

The effect of having one time-delay in the controller is the same as having one extra time-delay in the plant model [4.11]. Thus system performance can be evaluated by including the sample time delay in the plant model [4.11].

In servo systems the response should be virtually instantanous. Therefore it is unacceptable for the controller to have any time delay at all; hence the second implementation only is considered in this work.

The execution times obtained for different algorithms are shown in Appendix-C. In this system the slight delay between the y(t) sample and the u(t)

output has negligible effect on the actual response of the system. A rule of thumb would be to keep the delay in the order of 1/20th of the system rise time [4.7].

4.4.4 Practical constraints

The non-linearity produced by system saturation (Fig.4.10) represents the practical behaviour of many actuators and final control elements [4.12]. For example, a motor-amplifier combination can produce a torque proportional to the input voltage over limited range. However, no amplifier can supply an infinite current; there is a maximum current and thus a maximum torque that any practical system can produce [4.8]. The final control element is said to be overdriven when commanded by the controller to do something beyond its capabilities. The performance of all control elements is ultimately limited by the rate at which they can supply energy. Therefore it is important that all system performance specifications and controller designs be consistent with the energy-delivery capabilities of real devices [4.8,4.13].

In this work system identification is part of the self-tuning controller. Therefore, the system identification may fail to identify the system if it saturates. This is due to the fact that the identification algorithm will identify a saturating plant as having a very low gain. Consequently a controller with an unsuitable high gain will be implemented [4.14]. It is therefore essential to make sure that the controller never allows the plant to saturate. There are two ways to achieve this [4.8]:

(a) Limiting the output of the controller so that it does not exceed the value that causes saturation. This technique is useful in applications where the magnitudes of the inputs are difficult to predict in advance [4.8].

(b) Selecting the gains so that saturation will never occur. This requires knowledge of the maximum input magnitude that the system will encounter. Denote the maximum expected value of w(t) (set-point) by W_{max} and the value of the controller output at saturation by U_{max}. Then at t=0 the maximum error is W_{max}. The maximum gain for linear operation is [4.13]:

$$K_{max} = \frac{U_{max}}{W_{max}}$$

The plant will always operate within the linear region if one of the above methods is used.

Although low gain is a desirable feature to reduce both noise effects and wear and tear on control components it is a problem for system loops that exhibit steady state errors (e.g. positional errors in a type-0 system). In theory type-1 systems do not have steady state positional error but in practice this is not the case due to the presence of nonlinear effects such as stiction and friction. Furthermore low gains cause a deterioration of the system dynamic responses [4.13]. Hence output limiting techniques are preferred. It is therefore essential that, when identifying the plant, an amplitude limited control signal is used. However if control only is required (i.e. no identification) then the plant may be overdriven if higher output levels are demanded. This is done by incorporating the non-linearities into the controller software.

This ensures that the actuator system operates within its linear region. Test results [Chapter-9] show that system performance is not degraded when amplitude limits are set into the controller.

It is suggested [4.8] that control system performance can be improved if the limited energy-delivering capacity of the final control element is taken into account.

4.4.5 Numerical accuracy

There are a number of different sources of numerical errors when implementing a controller on a microprocessor. The major soures ones are:

* Arithmetic operations
* Memory word length
* A/D converter
* D/A converter

Internal number values are represented using integer and floating-point formats. Using the Intel 8087 Numeric Data Processor as a co-processor real numbers are represented in the system as an 8-byte word. This gives a range of $\pm$ $4.19*10^{-307}$ to $\pm$ $1.6*10^{308}$, the corresponding precision being approximately 18 (decimal) digits; integers are in the range of $\pm 2^{64}$ [4.17]. Applications that need to deal with data and final results outside this range are rare.

The only limitations are then the A/D and the D/A converters. Typically the converters used are 8, 10, 12, 14 and 16 bits ones, which correspond to resolutions of 0.4 %, 0.1 % , 0.025 % , 0.006 % and 0.00153 % . For ADCs, going above 12-bit operation is very expensive, therefore a 12-bit converter is a good compromise between cost and resolution.

4.5 CONTROLLER DESIGN TECHNIQUES

4.5.1 Overview

A number of different control algorithms are used here in the design of the digital controller. These are implemented in the microprocessor in software, the resulting system performance being evaluated for each implementation. The most important factors to be considered for each algorithm are the execution time and the ability to specify the dynamic (transient) response of the system. In this section the following control criteria are implemented and evaluated:

* One-Step-Ahead Prediction
* Weighted One-step-Ahead Prediction
* Pole/Zero Cancellation
* PID Controller

Design details for each criterion are described in depth in Appendix-C, the general results being summarized in the following sections.

4.5.2 One-Step-Ahead Prediction

The control objective here is to compute the control signal at the present instant of time in order to bring the future plant output to some desired value. Thus the design criterion is to minimise the prediction error (i.e. the difference between the actual value at some future instant and that actually desired).

The one-step-ahead controller minimises a cost function (I) which is defined as:

$$I = [\, y(T+1) - w(T)]^2 \qquad (4.6)$$

where y(T+1) is the output at time T+1 and w(T) is the desired value.

The transfer function (4.4) is rewritten here:

$$G(z) = \frac{a_1 z^{-1}}{1 + b_1 z^{-1}} \qquad (4.7)$$

where $a_1 = kh/\tau$ and $b_1 = -1$

The discrete time domain of Eq.(4.7) is:

$$y(T) = -b_1 y(T-1) + a_1 u(T-1) \qquad (4.8)$$

To minimise I in Eq.(4.6) $\frac{\partial I}{\partial u(T)}$ is set equal to zero.

The design details are shown in Appendix-C. The control signal that minimises Eq.(4.6) is:

$$u(T) = \frac{1}{a_1} w(T) + \frac{b_1}{a_1} y(T) \qquad (4.9)$$

Eq.(4.9) can be written in the general form (Fig.4.5) of Eq.(4.5) as:

$$u(T) = f_0 w(T) - s_0 y(T) \qquad (4.10)$$

where R = 1 , $f_0 = 1/a_1$ and $s_0 = -b_1/a_1$

Since $b_1 = -1$ then $f_0 = s_0$ and the controller is just a pure gain:

$$u(T) = K\,[\,w(T) - y(T)\,] \qquad (4.11)$$

where $K = 1/a_1 = \tau/kh$.

4.5.3 Weighted One-Step-Ahead Prediction

The purpose of the One-Step-Ahead control method is to bring the output to the desired value as quickly as possible. In a digitally controlled system this means that correspondence must be achieved within one sample period. Even if this is practicable it is likely that an excessive control effort will be called for. This can result in considerable wear and tear of the control elements. To avoid such effects some cost (weighting function) is put on the control effort, this being called the Weighted-One-Step-Ahead control method.

The cost function becomes:

$$I = [\ y(T+1) - w(T)]^2 + \lambda\, u^2(T) \qquad (4.12)$$

where the value of λ determines the compromise between the time taken to eliminate the plant error (w-y) and the amount of effort expended in doing this.

To minimise I in (4.12), set $\dfrac{\partial I}{\partial u(T)} = 0$

The controller that minimises Eq.(4.12) is:

$$u(T) = \frac{1}{a_1 + \lambda/a_1}\, w(T) + \frac{b_1}{a_1 + \lambda/a_1}\, y(T) \qquad (4.13)$$

Using the design information shown in Appendix-C. Eq.(4.13) can be written in the general form (Fig.4.5):

$$u(T) = f_0\, w(T) - s_0\, y(T) \qquad (4.14)$$

where $R = 1$, $f_0 = 1/[\, a_1 + \lambda/a_1]$ and $s_0 = -b_1/[\, a_1 + \lambda/a_1]$

4.5.4 Pole/Zero Cancellation

This design approach considers a servo problem expressed in terms of a model that gives the desired dynamic response to command signals [4.1]. Fig.4.11 and Fig.4.12 show the block diagrams of the system and a general model form.

Many closed loop systems can be satisfactorily modelled using a 2nd order (quadratic) lag, as follows.

$$G_m(s) = \frac{\omega_n^2}{s^2 + 2\,\zeta\,\omega_n s + \omega_n^2} \qquad (4.15)$$

For such a model the important dynamic parameters to the control engineer are

* Rise time (T_r)
* Overshoot (M_p)
* Settling time (T_s)

These are illustrated in Fig.4.13. Note that the transfer function itself defines the closed loop to have zero steady state error. Using this model, and working with specified values of T_r, M_p and T_s, ζ and ω_n are calculated [see Appendix-C]. Since the controller is implemented on a microprocessor, the discrete form of this continuous transfer function ($G_m(s)$) is determined using the Bilinear Z-transform [4.9]. The equivalent discrete form is

$$G_m(z) = \frac{d_0\,z^2 + d_1\,z + d_2}{z^2 + p_1\,z + p_2} \qquad (4.17)$$

The poles of the discrete time system are given by the characteristic equation:

$$z^2 + p_1 z + p_2 = 0 \qquad (4.18)$$

where

$$p_1 = \frac{2T^2\omega_n^2 - 8}{4 + 4T\zeta\omega_n + T^2\omega_n^2} \qquad (4.19)$$

$$p_2 = \frac{4 - 4T\zeta\omega_n + T^2\omega_n^2}{4 + 4T\zeta\omega_n + T^2\omega_n^2} \qquad (4.20)$$

The values of these poles (p_1 and p_2) are determined using ζ and ω_n.

Pole/zero cancellation techniques are used, the objective being to make the closed-loop transfer function of the system equal to the desired transfer function Eq.(4.17). The design details are discussed in Appendix-C, where the parameters (R, F and S) of the digital controller are determined. Substituting for these values in the general control Eq.(4.5) gives:

$$u(T) = f_0\, w(T-1) - s_0\, y(T-1) - r_1\, u(T-1) \qquad (4.21)$$

where

$$f_0 = \frac{1 + p_1 + p_2}{a_1}$$

$$s_0 = \frac{b_1^2 - b_1 p_1 + p_2}{a_1}$$

$$r_1 = p_1 - b_1$$

4.5.5 PID Controller

PID controllers are employed extensively in process systems and, to a lesser extent, in servo control applications. As a result they are perhaps the type most commonly encountered in practice. Fig.4.14 shows the conventional PID controller, its idealised equation being [4.8]:

$$u(t) = K \left[e(t) + \frac{1}{T_I} \int_0^t e(t)\, dt + T_D \frac{de(t)}{dt} \right] \qquad (4.22)$$

where the parameters

K = proportional gain

T_I = integration time

T_D = derivative time

For any sample time T Eq.(4.22) can be turned into a difference equation by discretisation. The derivative is simply replaced by a first order difference expression and the integral by a sum [4.15]. Applying backward rectangular integration gives [4.8]:

$$u(n) = K \left[e(n) + \frac{T}{T_I} \sum_{i=0}^{n-1} e(i) + \frac{T_D}{T} (e(n) - e(n-1)) \right] \qquad (4.23)$$

where u(n) is the current output, e(n) is the current error, etc.

This is a non-recursive control algorithm, which means that all past error values e(n) have to be stored. However a recursive algorithm is more suitable for programming on computers, since past error values do not have to be stored. This algorithm is characterized by the calculation of the current control signal u(n)

based on the previous value u(n-1) and correction terms [4.8]. The recursive algorithm is derived by subtracting from Eq.(4.23)

$$u(n-1)= K[\ e(n-1) + \frac{T}{T_I}\sum_{i=0}^{n-2} e(i) + \frac{T_D}{T}(e(n-1)- e(n-2))] \qquad (4.24)$$

where one obtains

$$u(n)- u(n-1)= q_0\ e(n) + q_1\ e(n-1) + q_2\ e(n-2) \qquad (4.25)$$

and

$$u(n) = q_0\ e(n) + q_1\ e(n-1) + q_2\ e(n-2) + u(n-1)$$

with parameters

$$q_0 = K(1 + \frac{T_D}{T}) \qquad (4.26)$$

$$q_1 = -K(\ 1 + 2\ \frac{T_D}{T}\ - \frac{T}{T_I}\ ) \qquad (4.27)$$

$$q_2 = K\ \frac{T_D}{T} \qquad (4.28)$$

Fig.4.15 shows the recursive algorithm of the digital PID controller, Fig.4.16 illustrating the complete control loop. The Z-transfer function of the controller is (from 4.25)

$$u\ (\ 1 - z^{-1}) = q_0 e + (q_1 e)z^{-1} + (q_2 e)z^{-2}$$

i.e.

$$G_c(z) = \frac{u(k)}{e(k)} = \frac{Q}{J} = \frac{q_0 + q_1\ z^{-1} + q_2\ z^{-2}}{1 - z^{-1}} \qquad (4.29)$$

and as $e(n) = w(n) - y(n)$

$$(1 - z^{-1})u(k) = (q_0 + q_1\ z^{-1} + q_2\ z^{-2})[w(k) - y(k)] \qquad (4.30)$$

From Fig.4.16 the closed loop equation is found to be:

$$y(T) = \frac{Q\ A}{B\ J + Q\ A}\ w(T) \qquad (4.31)$$

The general control input form (Fig.4.17) can be compared with the PID control input function (Fig.4.18, Eq.4.30). The control input of Eq.(2.31) can be associated with the general controller form Eq.(4.5) if

$$F/R = S/R$$

then

$$R = J = (1 - z^{-1}) \text{ and } F = S = Q = q_0 + q_1\ z^{-1} + q_2\ z^{-2}$$

The coefficients of the Q polynomial q_0, q_1 and q_2 must be selected to meet the desired performance objective, hence the K, T_I and T_D coefficients will be determined using Eqs.(4.26) to (4.28). The pole assignment criterion is used, in which the denominator of the closed loop Eq.(4.31) is equated with P, the apriori

selected polynomial. The equation required to meet the pole assignment objective is therefore:

$$B\,J + Q\,A = P \tag{4.32}$$

from which the coefficients of the Q polynomial are found.

The apriori defined polynomial P corresponds to the continuous time polynomial

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \tag{4.33}$$

as shown in Eq.(4.18) using the backward shift operator is

$$1 + p_1 z^{-1} + p_2 z^{-2} \tag{4.34}$$

where

$$p_1 = \frac{2T^2\omega^2 - 8}{4 + 4T\zeta\omega_n + T^2\omega_n^2}$$

$$p_2 = \frac{4 - 4T\zeta\omega_n + T^2\omega_n^2}{4 + 4T\zeta\omega_n + T^2\omega_n^2}$$

The values of ζ and ω_n are calculated according to the specified rise time and overshoot as shown in Appendix-C.

Substitute for A, B, J, Q and P from Eq.(4.7),Eq.(4.29) and Eq.(4.34) respectively in the design Eq.(4.32) to calculate the parameters of the controller.

The details are shown in Appendix-C. The control signal function is then:

$$u(T) = q_0 [w(T)-y(T)] + q_1 [w(T-1)-y(T-1)] + u(T-1) \qquad (4.35)$$

and this is equivalent to a PI controller as shown in Appendix-C.

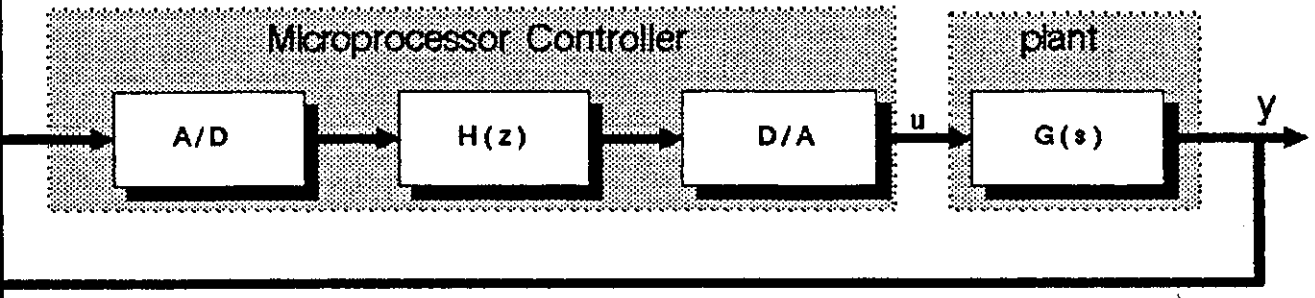


Fig . 4 . 1

system diagram

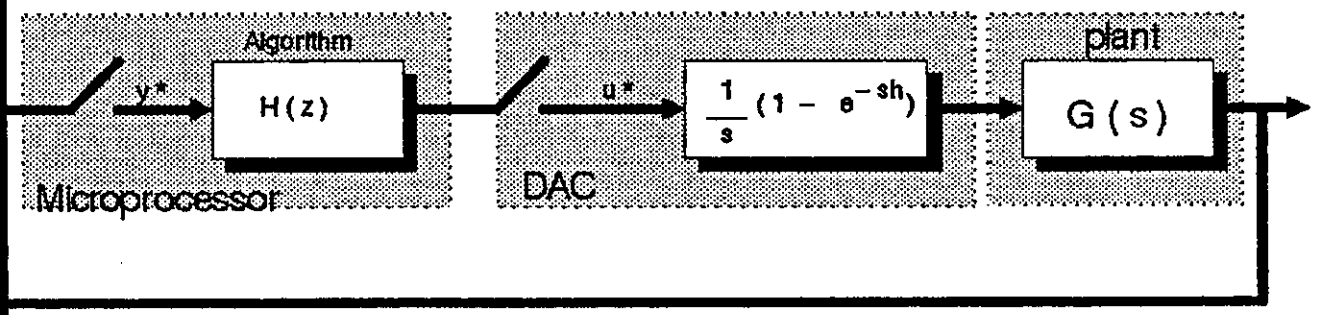


Fig . 4 . 2

System diagram – sampled data form

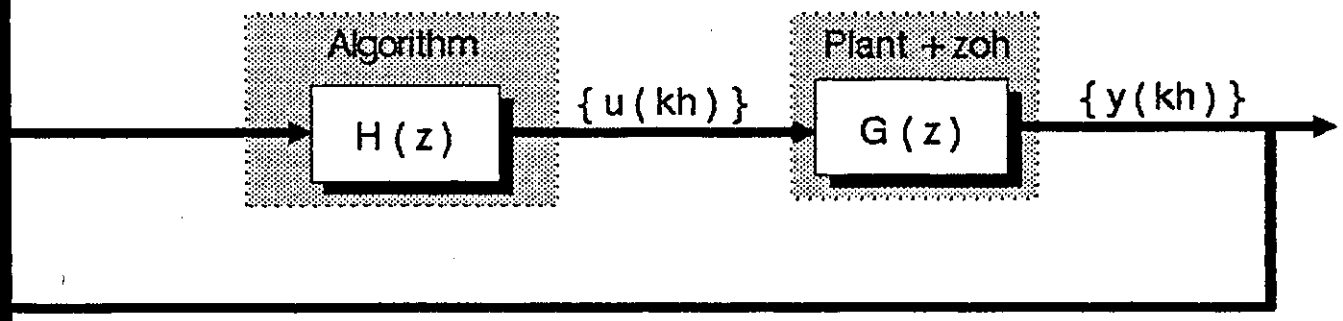


Fig . 4 . 3

Discrete equivalent form of controller and plant

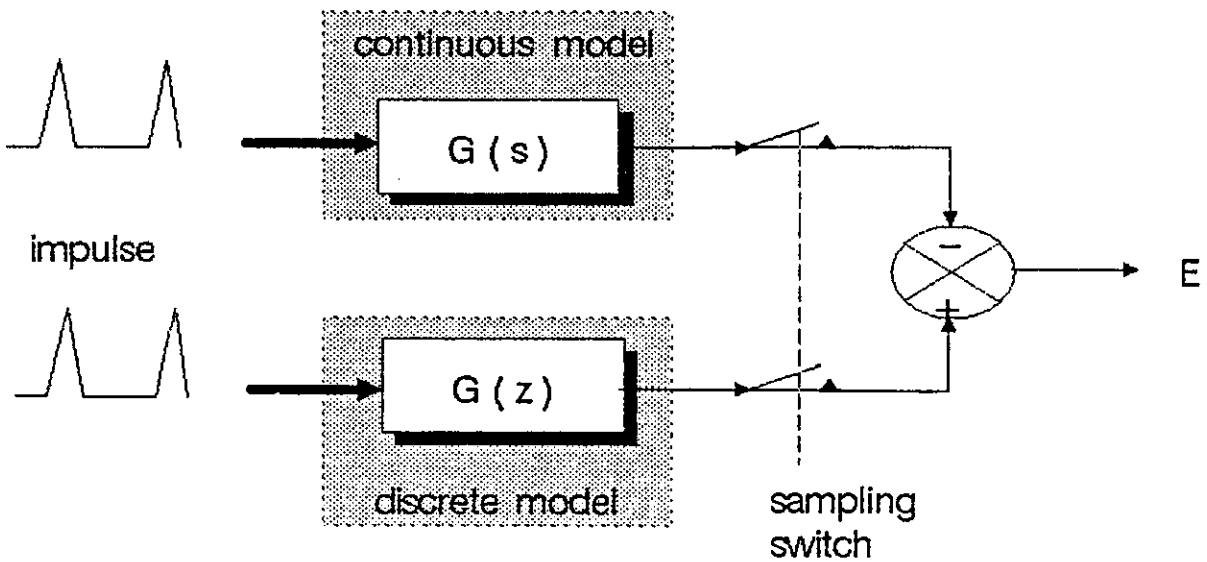


Fig . 4 . 4

The Impulse Invariant Transform (I I T) technique

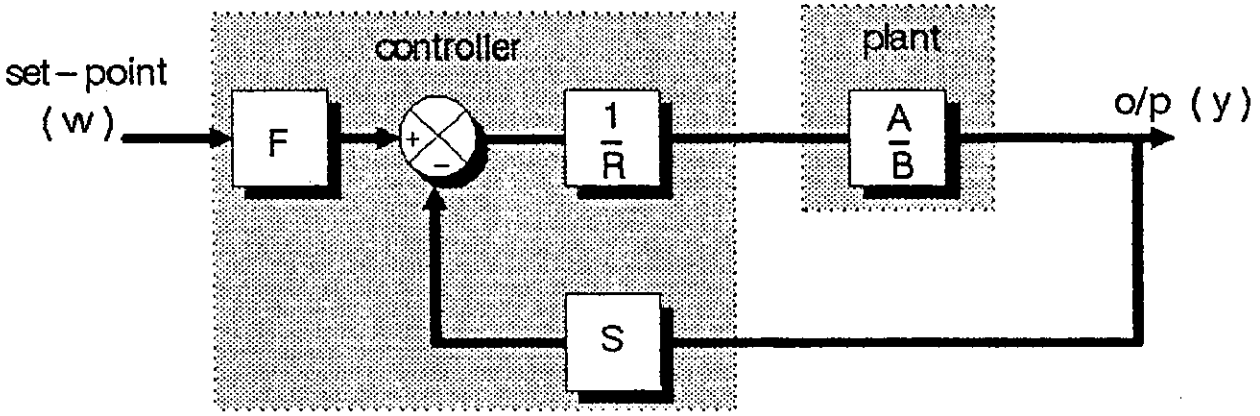


Fig . 4 . 5

GENERALISED CONTROLLER SRTUCTURE

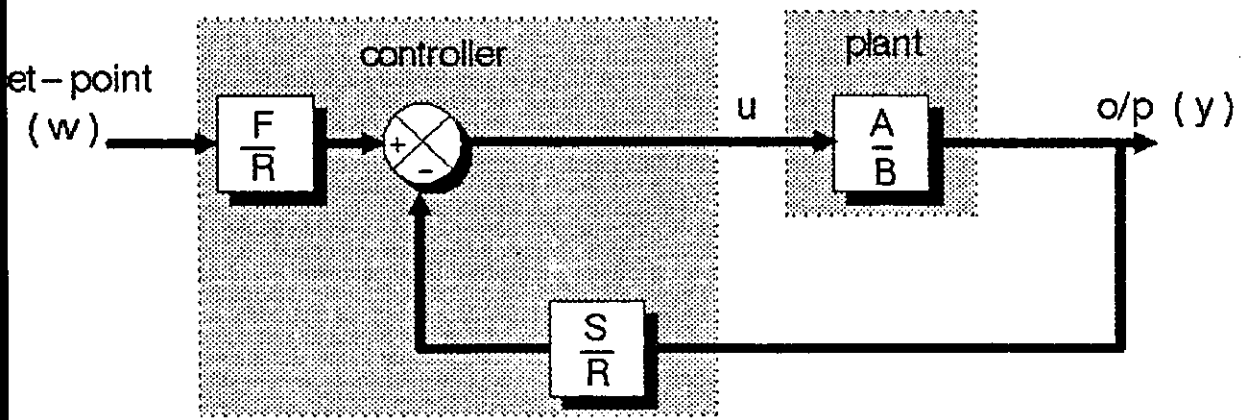


Fig 4 . 6

EQUIVALENT STRUCTURE TO FIG. 4.5

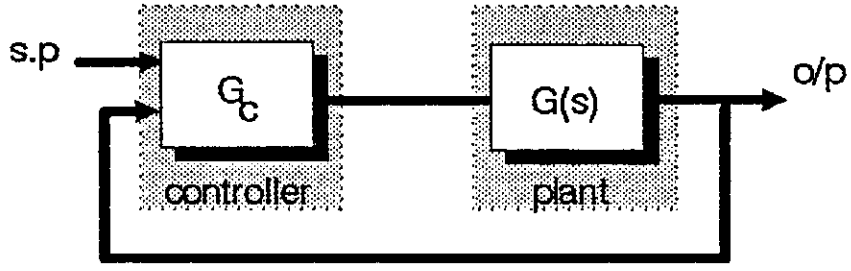


Fig . 4 . 7

Complete control loop

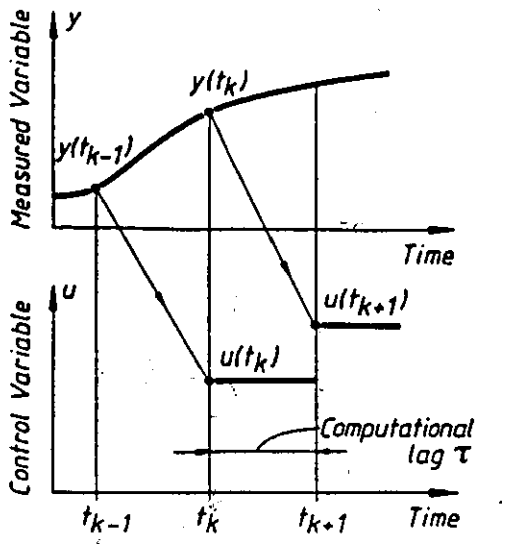


Fig . 4 . 8

Input – output delay – sample rate limited

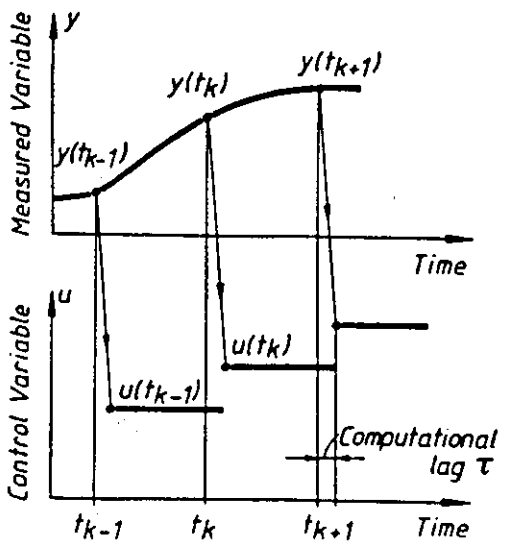


Fig . 4 . 9

Input – ouput delay – computation time limited

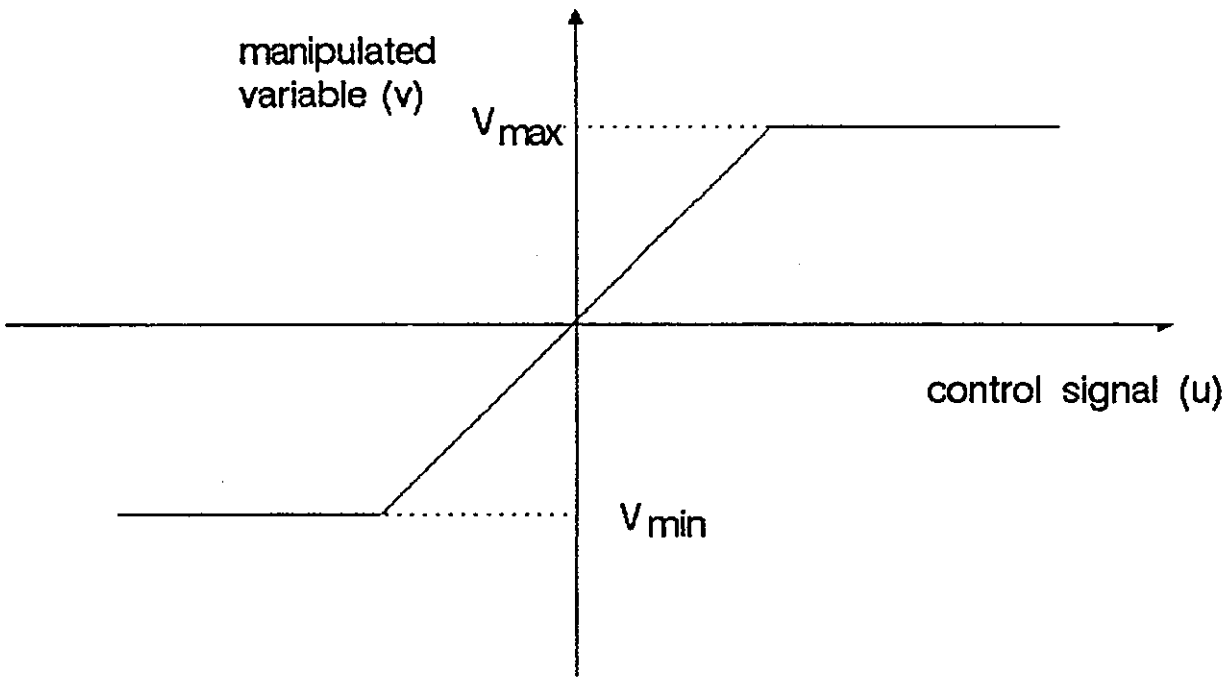


Fig . 4 . 10

non – linearity due to system saturation

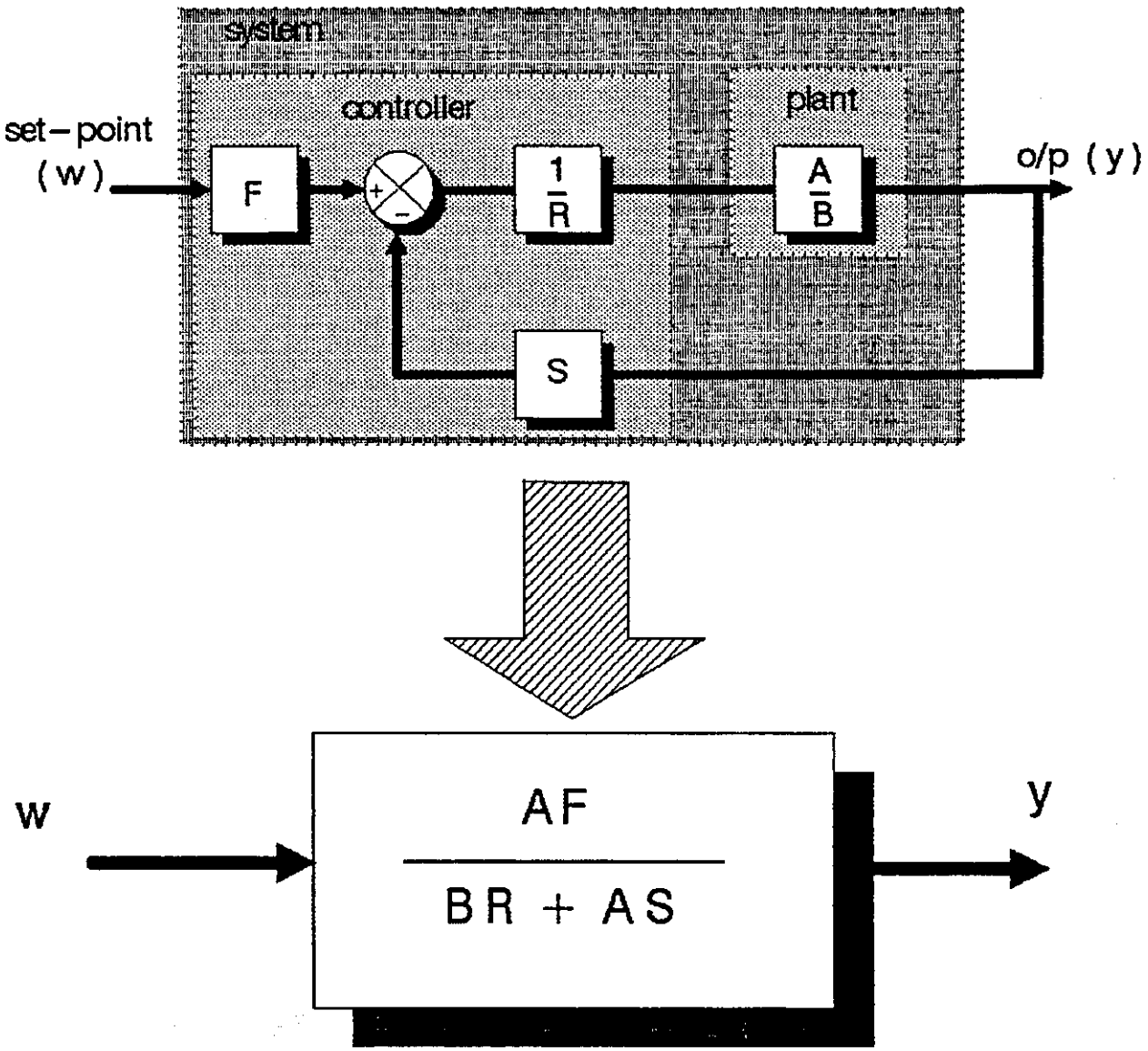


Fig . 4 . 11
System closed loop transfer function

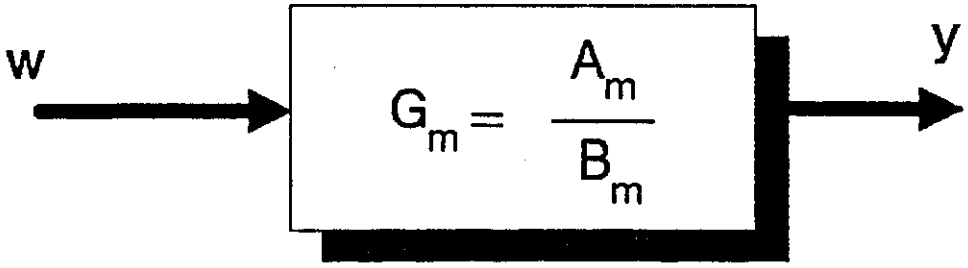


Fig . 4 . 12
Generalised closed loop transfer function

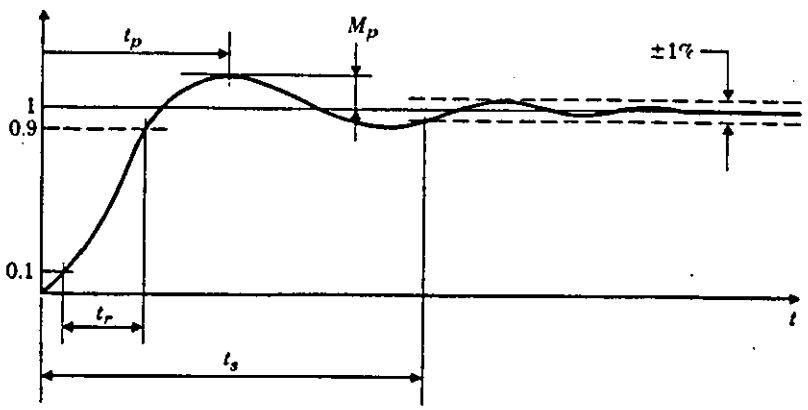


Fig . 4 . 13

Definition of rise time t_r , settling time t_s , and overshoot M_p

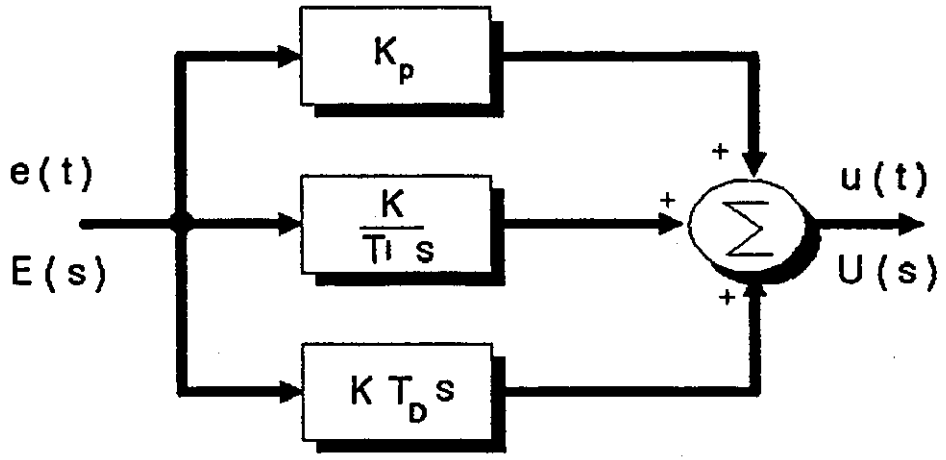


Fig . 4 . 14

Continuous PID Controller

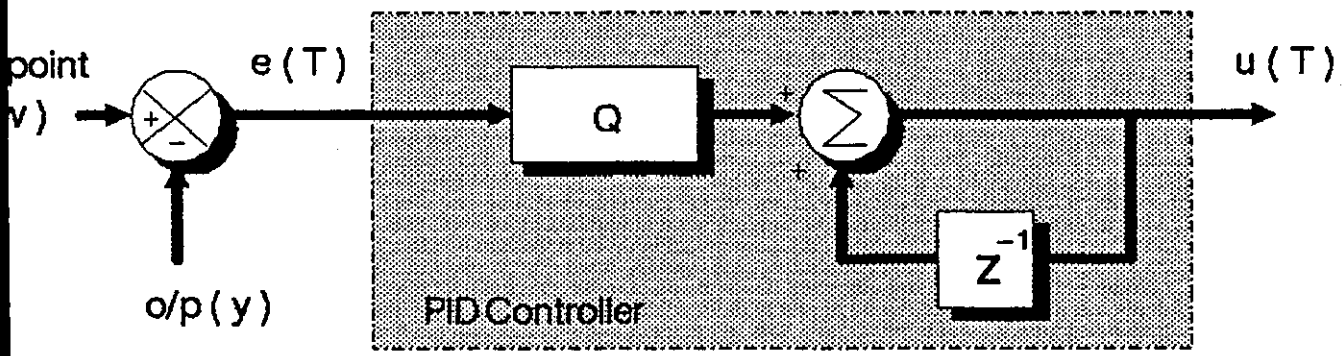


Fig . 4 . 15

Digital PID Controller – Recursive Algorithm Implementation

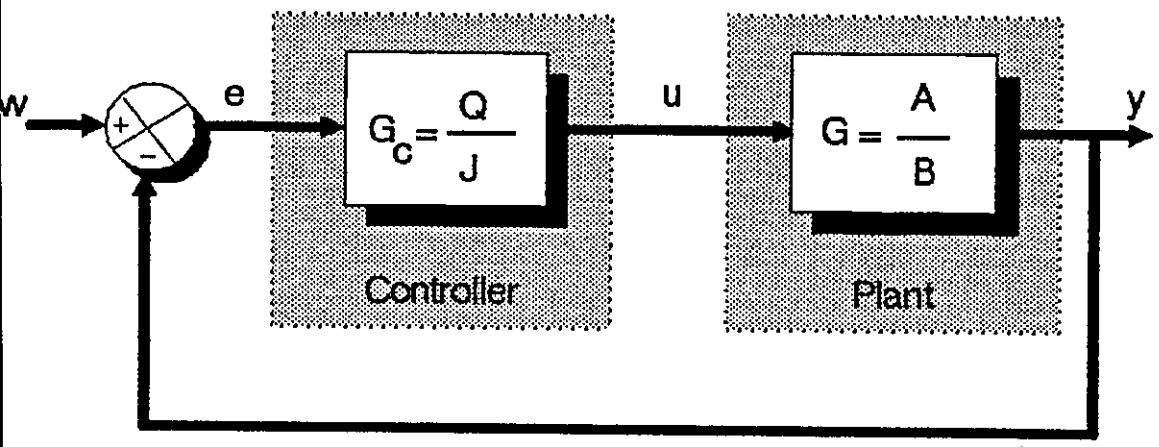


Fig . 4 . 16

Complete control loop

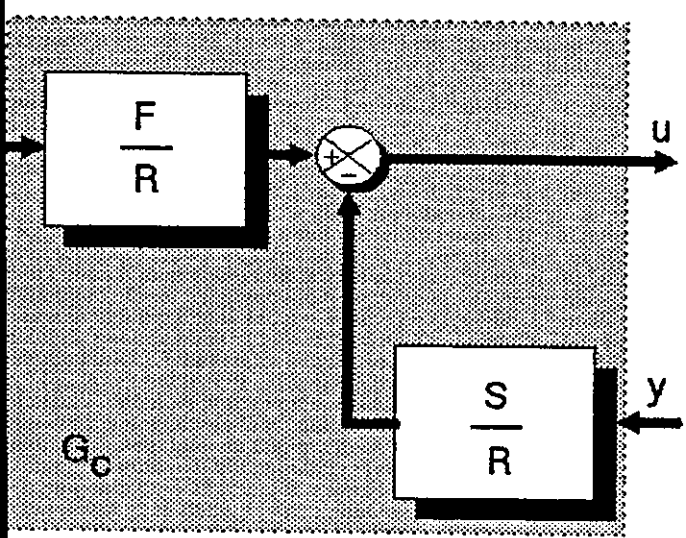


Fig . 4 . 17

The general controller

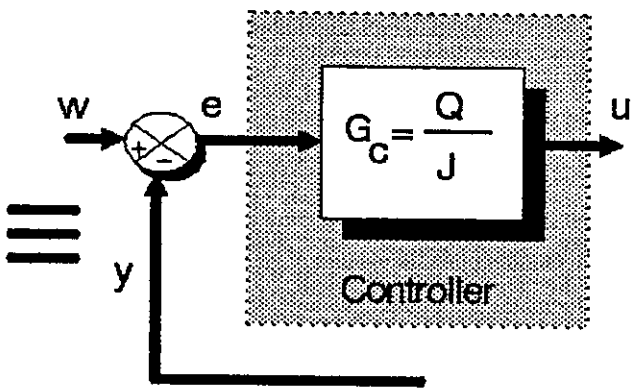


Fig . 4 . 18

The PID controller

CHAPTER - 5

5 SELF-TUNING CONTROLLER

5.1 INTRODUCTION

In a conventional closed loop control system (Fig.5.1) the controller is designed for fixed and known plant parameters. If these parameters vary due to environmental changes, then the control system may exhibit satisfactory responses for one environmental condition but may fail to do this under others [5.8]. Therefore a fixed controller will not meet the specified closed loop performance criteria over the full operating range of the plant. Furthermore, large variations of plant parameters may cause instability [5.8]. In such situations, to meet the desired performance criteria, it is usually necessary to adjust the controller parameters every time the environmental conditions change. This can be done manually as in the case of the conventional PID controller. Here the tuning of the controller is adjusted as required by the plant engineer, controller settings being based on both experience and trial and error techniques [5.4]. In contrast, for self-tuning systems, we compensate for variations in the transfer function of the plant simply by retuning the controller until satisfactory system performance is achieved under all operating conditions [5.7]. Hence it is necessary for the plant transfer function to be identified continuously, a process which is carried out automatically as the plant is in operation.

Another feature of the self-tuning controller is that tuning does not require operator experience or involvement.

5.2 BASICS OF THE ADAPTIVE (SELF-TUNING) CONTROL SYSTEM

The control system is called adaptive if it has the ability to modify its behaviour, on-line, to meet specified closed loop criteria [5.12].

Fig.5.2 shows the basics of the adaptive controller. A recursive parameter estimator monitors the plant's input (u) and output (y) signals. From these it computes an estimate of the plant dynamics in terms of a set of parameters in a prescribed structural model. The parameter estimates are fed into a control design algorithm which then provides a new set of coefficients for the feedback law [5.3].

The important point about self-tuning is that, even if detailed knowledge of the plant dynamics is unavailable, the adaptation mechanism achieves this closed-loop performance by adjusting the controller coefficients automatically. So we can view the self-tuner as consisting of two parts:

* A Controller: designed to meet specified closed loop performance criteria.
* A Parameter Estimator: to monitor the plant and compute its transfer function.

5.3 IMPLEMENTATION CONSIDERATIONS

5.3.1 The control function

The general form of the control algorithm is designed and implemented in software, running on a microprocessor. It still remains necessary to calculate the controller coefficients as a function of the plant parameters [5.6], these being adjusted dynamically.

Many techniques are available which are suitable for the design of digital controllers [5.1,5.2,5.5,5.7,5.11]. For servo applications Pole/Zero cancellation or PID controllers, as mentioned in Section.(4.5), are especially useful. These allow the operator to specify the desired plant performance in terms of the system dynamics.

5.3.2 The identification process

The self-tuner has to have the facility to estimate the parameters of the plant from its input and output signals. If identification is done while the plant is running, the technique is called on-line estimation. Recursive techniques are suitable for on-line identification and include the following methods (Fig.5.3) [5.7,5.10]:

* Recursive Least Squares (RLS).
* Recursive Extended Least Squares (RELS).
* Recursive Maximum Likelihood (RML).
* Recursive Instrumental Variable (RIV).
* Stochastic Approximation Method (STA).

These methods are discussed in details in Appendix-B. It has been found that RLS, RELS and RML are most suitable for self-tuning due to the simplicity of the

implementation and the good results achieved [5.7,5.9].

When identification is done on-line the estimated parameters are updated at each sample instant. Servo systems are usually fast, having wide bandwidths; as such the sampling rates are high compared with process control. Therefore the identification calculation must be carried out quickly. Further, the estimated values of the parameters need to converge to the real values in as few samples as possible. Table.B.1 [Appendix-B] shows that the Recursive Least Squares (RLS) identification algorithm is most suited for such requirements.

If the plant parameters change rapidly, the estimated parameters must converge rapidly to their new actual values. Convergence is speeded up by modifying the identification algorithm through the use of a forgetting factor, as shown in Appendix-B [5.10,5.9].

5.4 GENERAL DESCRIPTION OF SELF-TUNING SYSTEMS

Fig.5.4 shows the structure of a self-tuning controller. This is divided into three main components;

(a) an identification component which identifies the parameters of the system using the input and the output signals to the system

(b) a control-design algorithm which uses the estimated parameters of the system to provide the controller coefficients

(c) a controller part which uses these coefficients to compute the output control signal [5.3]

The control-design algorithm uses the estimated parameters for the purpose of controller parameter calculation as if they are the true parameters. This is called the certainty equivalence principle [5.12], the resulting design being called a "certainty equivalent controller". This controller does not take into

account any uncertainty of the estimates [5.12].

We can view the self-tuner (Fig.5.4) as an intelligent feedback control technique which, at each control interval, executes the following sequence:

* estimates the unknown plant parameters via system identification techniques (using input and output measured signals)
* calculates the controller parameters using these estimated values
* implements the resultant control law

By applying this sequence continually the controller is automatically tuned up for a particular system until it meets the required performance. The self-tuner constantly monitors the system but, as the estimated parameters converge to their true values, the self-tuning algorithm will perform as a conventional fixed controller. If the system characteristics change the control law is once more retuned to meet the new system configuration.

Either RLS, RELS, or RML (Appendix-B) methods can be combined with one of the control algorithms discussed in Section (4.5) to form the parameter adaptive controller [5.7].

5.5 IMPLEMENTATION OF THE SELF-TUNING CONTROLLER ALGORITHM

5.5.1 Pole/Zero cancellation Self-Tuning

(a) Overview

The Pole/Zero cancellation method discussed in Section (4.5.4) is used here to implement the self-tuning controller. In Section (4.5) the plant model used is:

$$B(z)y(T) = A(z)u(T) \qquad (5.1)$$

where u is the control signal, y is the measured signal, and A and B polynomials are the true parameters of the plant. In a self-tuning controller the plant parameters are replaced by estimated ones, and the assumed plant model is defined as;

$$\hat{B}(z)y(T) = \hat{A}(z)u(T) \qquad (5.2)$$

where $\hat{B}$, and $\hat{A}$ represent the estimated parameter polynomials.

We shall consider the desired closed-loop behaviour to be given by the classical continuous time system

$$G_m(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \qquad (5.3)$$

since the dynamics of such systems (e.g rise time, overshoot, etc) are explicit functions of ζ and ω. Hence a system specification expressed in these familior control terms also implicitly specifies ζ and ω.

The equivalent discrete form, $G_m(z)$, of the continuous transfer function,

$G_m(s)$, is determined using the Bilinear z-transform as shown in Section (4.5.4). The desired closed loop transfer function from set-point to output signal is then:

$$G_m(z) = \frac{A_m}{B_m} \qquad (5.4)$$

and the design equation is:

$$\hat{B}\ R + \hat{A}\ S = B_m \qquad (5.5)$$

The general form of the controller Eq.(4.5) is rewritten:

$$R(z)\ u(T) = F(z)\ w(T) - S(z)\ y(T) \qquad (5.6)$$

where w is the set-point signal. The design details are discussed in Appendix-C, where the parameters (R, F and S) of the digital controller (5.6) are determined using the estimated polynomials A and B in Eq.(5.5) instead of the true polynomials A and B. Substituting for these values in the general control Eq.(5.6) gives:

$$u(T) = f_0\ w(T-1) - s_0\ y(T-1) - r_1\ u(T-1) \qquad (5.7)$$

The parameters of the model, A and B, of Eq.(5.2) are estimated using the Recursive Least Squares (RLS) method.

(b) Tuning technique

* Estimate the coefficients of the polynomials A and B in Eq.(5.2) using the Recursive Least Squares (RLS) identification method.

* Substitute for these values in Eq.(5.5) and solve for R, S and F.

* Calculate the control signal from Eq.(5.7).

* Repeat these steps at each sampling period.

5.5.2 PID Self-Tuning

(a) Overview

The PID controller form Eq.(4.31) derived in Section (4.5.5) is rewritten

$$(1 - z^{-1})u(T)=(q_0+ q_1z^{-1}+ q_2z^{-2})[w(T)-y(T)] \qquad (5.8)$$

The control signal of Eq.(5.8) can be associated with the general cotroller form Eq.(4.5) as shown in Section (4.5.5) if

$$R = 1 - z^{-1}, \quad F = S = q_0+ q_1z^{-1}+ q_2z^{-2} \qquad (5.9)$$

In Section (4.5.5) it is shown that the values of q_0, q_1, and q_2 are determined by solving Eq.(4.33)

$$B\ J + Q\ A = P \qquad (5.10)$$

In self-tuning the estimated values $\hat{A}$, and $\hat{B}$ of the assumed model are used instead of the true parameters A and B. Hence Eq.(5.10) becomes

$$\hat{B}\ J + Q\ \hat{A} = P \qquad (5.11)$$

The apriori selected polynomial P is designed to satisfy a specific system performance criteria such as rise time and overshoot; the corresponding controller parameters q_0, q_1, and q_2 are calculated by solving Eq.(5.11) [see Appendix-C].

$$q_0 = (2 + p_1)/ \hat{a}_1 \qquad (5.12)$$

$$q_1 = (p_2 - 1)/ \hat{a}_1 \qquad (5.13)$$

$$q_2 = 0$$

The control signal function is then:

$$u(T) = q_0 [w(T) - y(T)] + q_1 [w(T) - y(T)] + u(T-1) \qquad (5.14)$$

Note that this is equivalent to a PI controller [Appendix-C].

The Recursive Least Squares method is used to estimate the parameters of the model, A and B, of Eq.(5.2).

(b) Tuning technique

* Estimate the coefficients of the polynomials A and B in Eq.(5.2) using the Recursive Least Squares (RLS) identification method.
* Calculate the values of q_0 and q_1 from Eq.(5.12) and Eq.(5.13).
* Substitute for these values in the controller Eq.(5.14) to produce the control signal.
* Repeat these steps each sampling period.

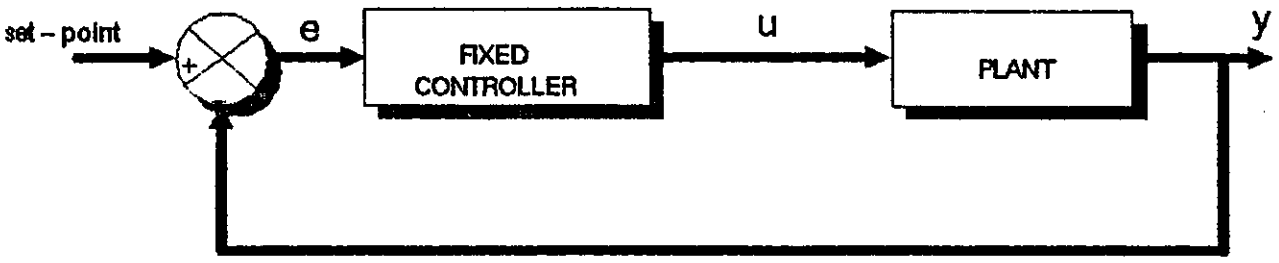


Fig . 5 . 1

CONVENTIONAL CONTROL SYSTEM

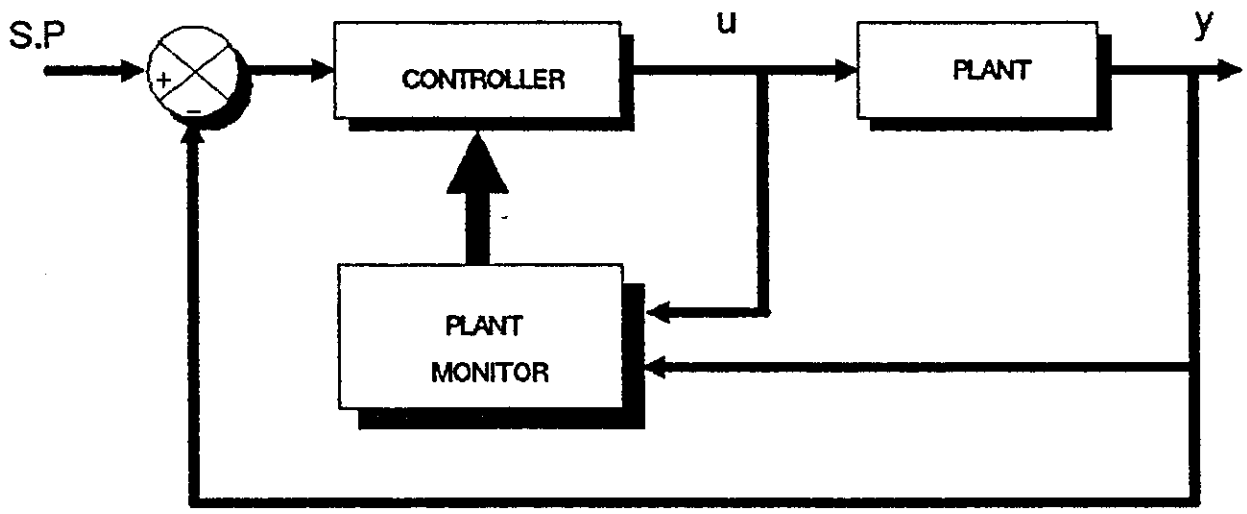


Fig . 5 . 2

ADAPTIVE CONTROL SYSTEM

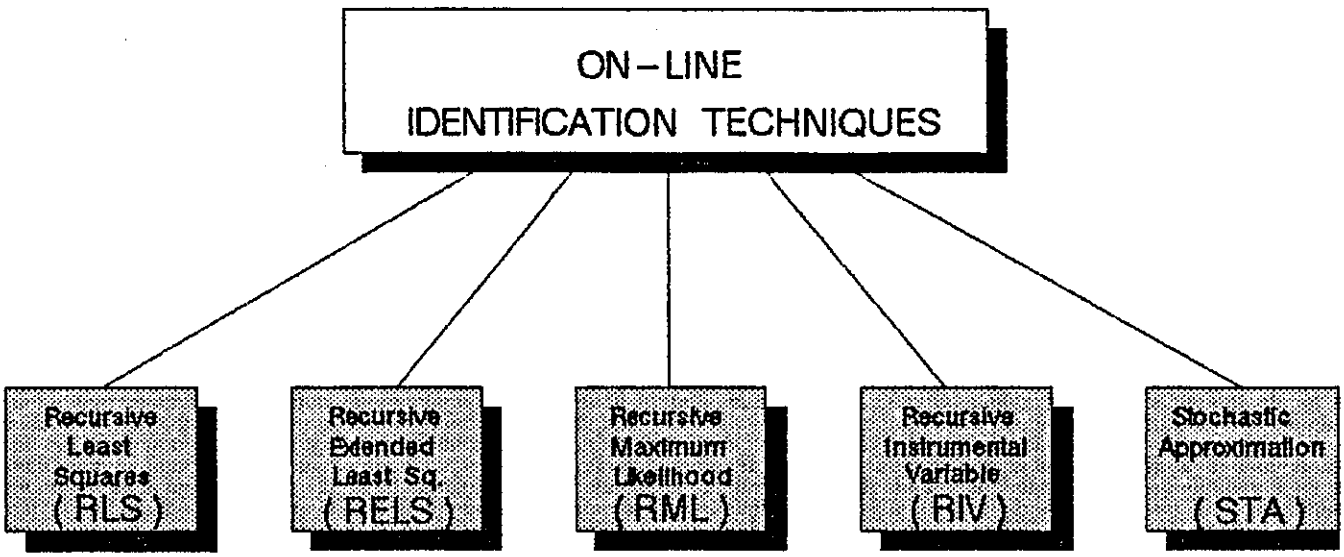


Fig . 5 . 3

IDENTIFICATION TECHNIQUES

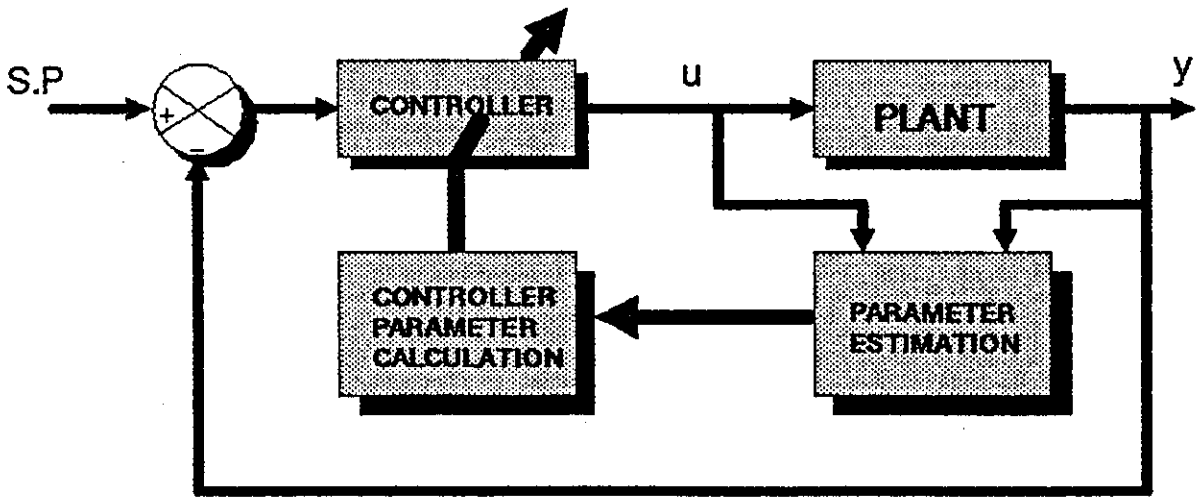


FIG . 5 . 4

BASIC STRUCTURE OF SELF – TUNING CONTROLLER

R&D

621.380413/TRE

Communication system design using DSP algorithms

001.6425/TEX - Assembly Language Programmers Guide

US AL-Sadigi, G.M.

Oliver, Michael Andrew

Lopes, L.E

Digital to analogue interfacing with micro (Alexander.)

621.3819583/ALE

Sensors and Signal Conditioning

by Ramon Pallas-Areny

John G. Webster (620.0044 PAR)

Telecommunication Networks: Protocols, Modeling by Mischa Schwartz

(621.380413 SCH).

621.38043 DEF Digital Signal Processing by David J. Defatta

CHAPTER - 6

6 DIGITAL CONTROLLER - ELECTRONIC HARDWARE

6.1 General information

The unit described here (Fig.6.1) is designed to be used as a general purpose digital controller in closed loop systems. Although meant initially for use in laboratory conditions its design reflects the requirements of real systems. It can be seen that the complete system consists of three main building blocks;

(a) Microcomputer (CPU) section.

(b) Analogue input/output (I/O) section.

(c) Serial communications I/O section.

All of the digital electronics are housed on a single printed circuit board, the analogue sub-system being located on a small adaptor board. This technique enables a standard computer section to be tailored to meet specific I/O requirements. The complete assembly is defined as the "target system", the micro being the target processor.

6.2 CPU section

This is based on the use of the Intel 8088 microprocessor, augmented by an 8087 numeric data co-processor. It is designed to handle a maximum memory address space of 64 kByte, all devices being memory (not I/O) mapped. The address space is programmable through the use of PROM decoding techniques. Currently the system is equipped with 32 kByte of Eprom for program and fixed data storage together with 16 kByte of RAM for variable data storage and stack operations. The design also requires the use of timing and interrupt functions. These are

implemented using standard microprocessor compatible components, being programmable for flexibility. As a safety feature a watchdog timer is included in the CPU design. Included also is a wait-state generator to enable the processor to be single-stepped through its program sequence.

In normal circumstances restricting the address space to 64 kByte allows the processor to operate in minimum mode [6.1]. Unfortunately, when using an 8087 the system has to be in maximum mode; as a result an 8288 bus controller must be added to the design.

6.3 Analogue I/O section

Two analogue input channels are provided. One is for the measured value variable (shaft position), the other being an optional extra for shaft velocity. Each signal is input via a differential amplifier, bandlimited by a 3rd order low-pass anti-alias filter and digitised by a 12 bit successive approximation ADC. Conversion time is insignificant when compared with computing activities.

Additional internal analogue signals are digitised, including calibration reference values and DAC outputs. Signal selection is carried out using an analogue multiplexer, the output from this being applied to a sample-hold module prior to digitisation.

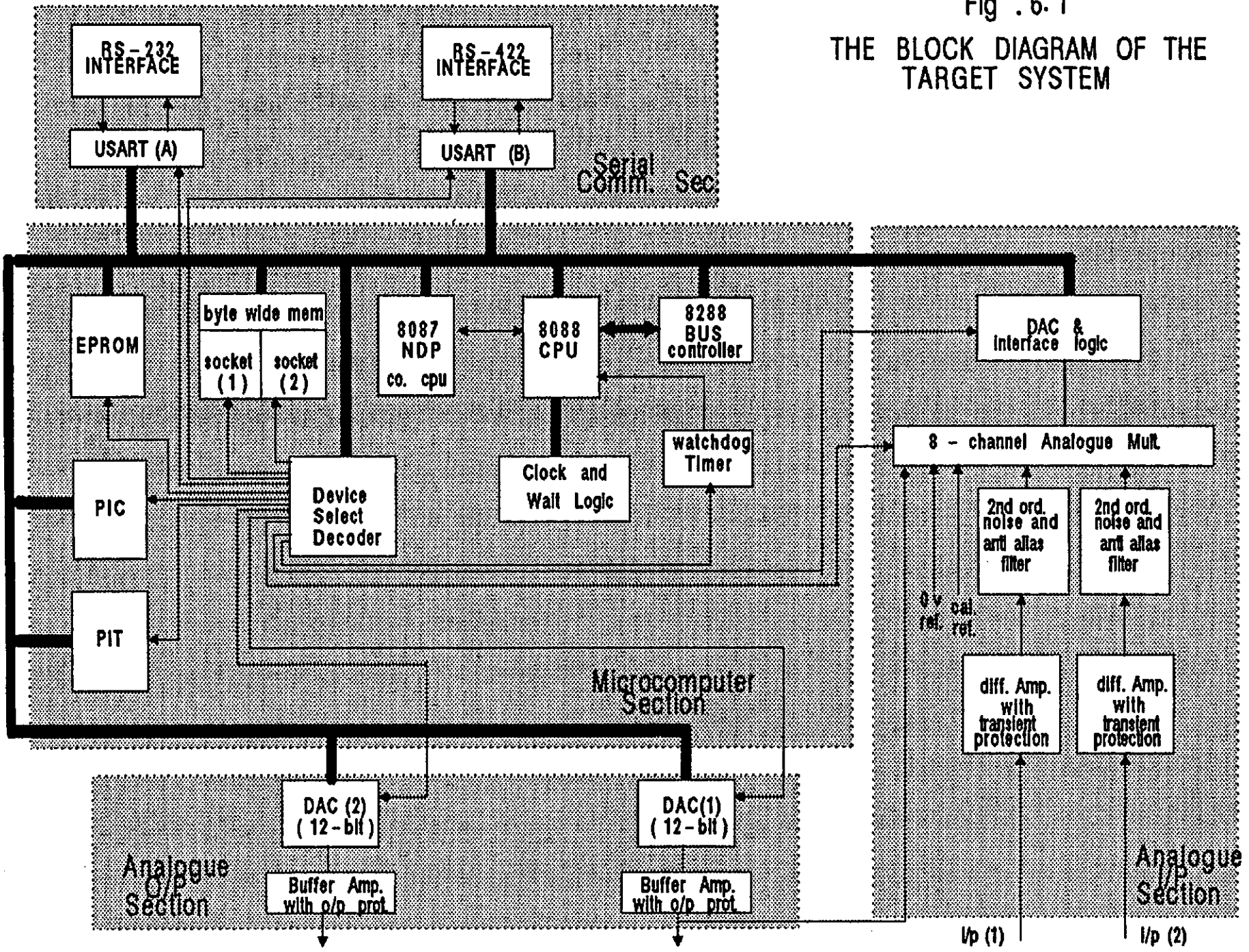
Two output stages are provided, each one consisting of a 12 bit digital to analogue (DAC) converter followed by a buffer amplifier. Simple low-pass filtering is used to minimise the effects of DAC glitch spikes and to act in part as a reconstitution filter. Both short circuit and transient overvoltage protection are included for the output amplifiers.

6.4 Serial communications

Two full-duplex serial communication channels are incorporated into the design; a short distance one corresponding to RS232C standard [6.2] and a longer distance RS422 one [6.3]. The RS232 one is designed mainly for interactive working with the PC. When used in this mode the maximum data rate is 9.6 kBaud, speed being limited by the PC. The RS422 channel is included to support controller operations within a distributed control system.

Fig . 6.1

THE BLOCK DIAGRAM OF THE TARGET SYSTEM



CHAPTER - 7

7 SOFTWARE DESIGN AND DEVELOPMENT

7.1 THE HOST DEVELOPMENT SYSTEM

A Future FX20 Personal Computer (described in Appendix-A) is used as a host for the development of software and down-loading of ROMable code to the target system. Fig.7.1 shows its connection in the system where it also provides an interface between the operator and the target system.

7.2 DESIGN TECHNIQUES

It is very important in the development of reliable software to have a complete and correct understanding of what the system is expected to carry out. The design method used to convert the specified requirements into software code also affects the reliability of the software [7.1].

The basic software design method used is that of Structured Programming [7.2], a technique which contains the merits of both Top-Down design and Modular programming [7.3]. A set of regulations which defines structured programming uses is given in Section 7.3. Diagramming techniques are used throughout, specifically that based on the Jackson chart system [7.4]. This diagram (e.g. Fig.7.5) is read from top to bottom to obtain more detail on program activities and from left to right to get the time sequences. Such techniques are excellent at describing what needs to be done rather than how it should be carried out. The eye can often appreciate the information carried in shape of a diagram much more quickly than it can the written word. Furthermore sections can be added or removed as the work proceeds without disturbing the rest of the diagram [7.3]. This greatly assists programme development and modification activities.

The Jackson chart is constructed from the test requirements and describes the software to a specific level of detail. The lowest levels represent simple functions that can be translated into program format. Generally the recommended control structures of structured programming [7.1] have been used in the writing of the program source code.

The programs consist of procedures which are grouped according to their functional components (modules). Each module is independent of others from the compilation point of view. However any procedure can call any other procedure, irrespective of module location. The advantage of such modularisation is that new functions can be easily added to the program without disturbing other modules.

7.3 THE RULES OF STRUCTURED PROGRAMMING

Structured programming uses a set of rigid constructs which can be defined collectively by three main rules [7.3].

(i) The first rule defines the syntax structures allowed. These include:

a- Sequencing

b- Selection of the next statement by the testing of a condition (The IF-THEN-ELSE structure)

c- Iteration

GOTO statements should be used only in exceptional circumstances.

(ii) The second rule relates to the program design. A Top-Down design technique incorporating step-wise refinement should be used. The program is divided into levels, with the highest level showing the flow of control among other major modules of the program. Each module should have only one entry and one exit point.

(iii) The third rule is to control the size of the program modules. The size of each module is limited so it may be easily read and understood. This improves the readibility of the code, an essential feature of reliable software [7.1].

7.4 PROGRAMMING LANGUAGE

It would have been possible to develop the software for the target system entirely in 8086 assembly language. This was rejected, the decision being to use a high level language wherever possible, turning to assembler only as a last resort. Three factors influenced this;

* Speed of development.
* Problem (and not processor) orientation of high level languages.
* Inherent support by block structured languages for structured design techniques.

In the context of the PC development environment it would make little sense to use a different language for the programs that run on the PC. Software commonality is highly advisable. Thus the basic language selection criteria are;

* The compiler must be capable of generating ROMable code.
* Code produced must also run under the PC operating system.
* Object ("in-line") code inserts must be supported.
* Access of specific memory addresses and hardware devices from the source code (i.e. the high level language statements) must be a standard feature.
* Access to the processor interrupt structure is essential.

Digital research Pascal/MT+ was selected as the software source code language for both the PC and the target system.

7.5 PROGRAM STRUCTURE AND DEVELOPMENT

In Pascal/MT+ the program comprises of a main body and modules. Fig.7.2 shows the software organisation for this project, the details being listed below.

* The main body and each module (extension .PAS) is compiled separately to produce a relocatable file of extension R86.
* These files and other files provided by Pascal/MT+ (library modules) are linked together to produce a single output, the CMD file. This file can be run under CCP/M-86 on the FX20 PC.
* As the CMD file is not in Intel hex format it cannot be loaded directly to the target system. A program called CMD2H86 is used to convert it to Intel hex format, the resulting output having the extension H86.
* The H86 file is then down-loaded to an EPROM Programmer to be used in the target system.

7.6 THE SOFTWARE FUNCTIONS

The software described here has three functions (Fig.7.3);

* MODEL DETERMINATION
* FIXED DIGITAL CONTROLLER
* SELF-TUNING CONTROLLER

The function of MODEL DETERMINATION is to determine the transfer function that models the plant.

The FIXED DIGITAL CONTROLLER implements a digital controller in the target system using different control criteria.

The function of SELF-TUNING CONTROLLER is to implement an adaptive controller based on various control criteria to control the plant via the target system.

7.7 MODEL DETERMINATION

7.7.1 Overview

Fig.7.4 shows the basic blocks of this function which consists of four programs, namely "DATA-COLLECTION", "OFF-LINE- SYSTEM-IDENTIFICATION", "MODEL-ORDER-REDUCTION", and "MODEL TRIMMING". The function of each program is as follows. The first one implements collection of plant data by the target system, the second performs system identification using this data (done off-line on the FX20), the third executes transfer function order reduction of the model obtained from system identification whilst the fourth trims the reduced order model to adjust its structure.

7.7.2 Data collection - general description

Fig.7.5, the structure design chart, is produced from the system requirements; as such it should reflect what needs to be done to run the plant and collect appropriate data. From this the source code (Pascal) is generated. At the higher levels of the chart operations are fairly self explanatory; at lower ones a detailed knowledge of the system and hardware is required to fully understand what is happening.

The Pascal program must correspond to the design diagram (otherwise there is not much point in having the diagram in the first place). As an aid to program visibility and clarity, modular programming techniques are used extensively. Fortunately Pascal/MT+ allows modules to be compiled separately. The top level or "program" module is essentially made up of a set of procedures which are located in lower level modules, as follows;

```
(* This is the body of the Data Collection program module *)
BEGIN

  STOP_PLANT;
  INITIALISE_CNT_DATA_INT;
  SET_UP_SERIAL_COMMS;

  GET_TEST_PARAMETERS;
  SET_UP_ANALOGUE_SYSTEM;
  COLLECT_DATA_VIA_INT;
  TRANSFER_DATA;

END.
```

```
(* End of main program module *)
```

Each program statement above consists of a procedure without parameters; as such the sequence is clear, readable and unambiguous. In turn these procedures call lower level procedures, which, depending on the complexity of operations, may in turn call lower level operations still. For instance, the code for the procedure "SET_UP_SERIAL_COMMS" is;

```
(* This procedure is called by the main program module *)
PROCEDURE SET_UP_SERIAL_COMMS;
BEGIN
  INITIALISE_DEVICES;
  SEND_SIGN_ON_MESSAGE;
END;
(* End of the procedure *)
```

Procedure INITIALISE_DEVICES is implemented as follows;

```
(* This procedure initialises all CPU hardware *)
PROCEDURE INITIALISE_DEVICES;
BEGIN

  (* Initialise the Programmable Interval Timer (PIT) *)
  PIT_CONT  := CON_WORD_0;
  PIT_CTER_0:= DATA_LOW;
  PIT_CTER_0:= DATA_HIGH;
  PIT_CONT  := CON_WORD_2;
```

```
  PIT_CTER_2:= DATA_2_LOW;
  PIT_CTER_2:= DATA_2_HIGH;
  PIT_CONT  := CON_WORD_1;

  (* Initialise the UART *)
  UART_COMD:= $00;
  UART_COMD:= $00;
  UART_COMD:= $00;
  UART_COMD:= $40;
  UART_COMD:= MODE_UART;

  (* Initialise the Programmable Interrupt Controller (PIC) *)
  PIC_COM_0:= ICW_1;
  PIC_COM_1:= ICW_2;
  PIC_COM_1:= ICW_4;
  PIC_COM_1:= ICW_1;

END;
(* End of hardware initialisation procedure *)
```

The major software design objectives for reliability and clarity are modularisation, information hiding, loose coupling and high cohesion [7.2]. It can be seen that these are well supported by the structure and organisation of Pascal. However, for embedded applications, the language must

* allow the programmer to access the various hardware devices, preferrably from high level language statements.

* Use meaningful and recognisable names and identifiers.

Procedure INITIALISE_DEVICES shows how Pascal/MT+ facilitates such requirements. It also shows that, to develop software for target systems, an intimate knowledge of the processor hardware is needed.

7.7.3 Data collection - detailed information

It is shown from the Jackson chart of the Data Collection program, Fig.7.5, that the software is divided into two subsystems:

* INITIALISE SYSTEM.
* RUN TEST.

The function of INITIALISE-SYSTEMS is to stop the plant, initialise the "number-of-samples" counter, data space array and the vector interrupt, to set up the system devices so that the operator communicates with the target system, and finally to send a message to the VDU indicating completion of initialisation.

The function of RUN-TEST is to get the test parameters, select the analogue input measurement channel, send the plant input test signal, collect plant data, stop the plant and transfer the collected data the FX20 (PC).

(a) INITIALISE SYSTEM

The lower level of this subsystem consists of four procedures, "STOP-PLANT", "INITIALISE-CNT-DATA-INT", "SET-UP-SERIAL-COMM-SYS-DATCL", and "SEND-MESSAGE". These procedures may in turn call other (lower level)

procedures. Their functions are:

* STOP-PLANT: This sends a command signal to stop the plant as soon as the power is switched on.

* INITIALISE-COUNTER-DATA-INT: This initialises the number-of-samples counter, the data space array, and the vector interrupt.

* SET-UP-SERIAL-COMMS: This initialises the PIT, USART, and PIC. A message is then sent to the screen to indicate completion of initialisation.

* SEND-SIGN-ON-MESSAGE: This sends a message to the screen explaining the function of the main program.

(b) RUN TEST

The lower level of this subsystem consists of four procedures "GET-TEST-PARAMETERS", "SET-UP-ANALOGUE-SYS", "COLLECT-DATA-VIA-INT", and "TRANSFER-DATA". The funtion of these procedure are:

* GET-TEST-PARAMETERS: This gets the test parameters specified by the operator, the sampling rate, the test signal, and the number of samples.

* SET-UP-ANALOGUE-SYS: This sets the analogue system by selecting the input measurement channel and sending the specified command signal to run the plant. This is used as the plant input test signal, being either a step or a ramp.

* COLLECT-DATA-VIA-INT: This procedure enables the processor hardware interrupt, sets the sampling counter, and then waits for a hardware interrupt to occur. A separate design chart has been produced to show the function of the interrupt driven procedure (Fig.7.6). This procedure, when activated, collects samples of the plant output, increments the sampling counter, and returns to the main program. The COLLECT-DATA-VIA-INT procedure checks this counter until the specified number of samples is reached; at this point it disables the interrupt and sends a command signal to stop the plant.

* TRANSFER-DATA: This procedure transfers the collected data from the target board to the FX20 via the serial channel.

7.7.4 Off-line system identification

The off-line identification program is derived from the software design diagram of Fig.7.7. Only the higher levels are shown to maintain diagram clarity; however the essentials of the identification process can be deduced from this.

This program runs on the PC, having been developed on it in the first place. Thus, to produce the identification software, the programmer does not require any special knowledge of the target system. In fact test data is not even needed to evaluate the identification processes; simulations can be run on the PC using programmer provided information.

The Jackson chart (Fig.7.7) of the system identification software consists of three main subsystems:

* INITIALISE SYSTEM.

* SET RUN TEST PARAMETERS.

* IDENTIFY SYSTEM.

INITIALISE-SYSTEM initialises the data arrays and sets initial condition values needed for system identification.

SET-RUN-TEST-PARAMETERS gets the test parameters, selects the identification scheme and sets the structure of the (assumed) model for system identification.

IDENTIFY-SYSTEM sets the number of iterations, sets the input and output vectors, identifies the system parameters and displays values of the estimated system parameters.

(a) INITIALISE SYSTEM

The lower level of this part comprises two procedures "SET-INITIAL-CONDITIONS", and "SEND-MESSAGE". The function of these procedure are:

* SET-INITIAL-CONDITIONS: This procedure initialises all the data arrays and sets initial conditions needed for system identification.

* SEND-MESSAGE: This procedure sends a message to the screen indicating completion of initialisation.

(b) SET RUN TEST PARAMETERS

The lower level of this subsystem contains three procedures, namely "GET-TEST-PARAMETERS", "SET-IDEN-SCHEME", and "SET-MODEL-STRUCTURE", their details being as follows:

* GET-TEST-PARAMETERS: This procedure sends a list of options to the operator to set the test signal, the number of iterations needed, and the sampling rate used in data acquisition.

* SET-IDEN-SCHEME: This procedure allows the operator to select the identification scheme to be used in the test.

* SET-MODEL-STRUCTURE: This procedure enables the operator to set the structure of the (assumed) model for system identification.

(c) IDENTIFY SYSTEM

The lower level of this subsystem is comprised of "SET-NO-ITERATION-COUNTER", and "CARRY-OUT-RECURSIVE-IDENTIFICATION". The function of this is:

* SET-NO-OF-ITERATION: Sets the number of iterations specified by the operator during the acquisition of test parameters.

* CARRY-OUT-RECURSIVE-IDENTIFICATION: The lower level of this part comprises procedures that sets the input and output vectors, identifies

system parameters, and displays the values of these parameters. Note that recursive identification is repeated until the iteration counter reaches the preset value.

7.7.5 Model order reduction

Fig.7.8 shows the Jackson chart of this program, the design being coded in FORTRAN to run on MULTICs (main frame computer). It can be seen that the software is divided into three subsystems:

* INITIALISE-SYSTEM
* SET-RUN-TEST-PARAMETERS
* REDUCE-ORDER

INITIALISE-SYSTEM initialises the data arrays, specifies the names of data files, sets initial condition values needed for model order reduction and sends a message to the screen to indicate completion of initialisation.

SET-RUN-TEST-PARAMETERS gets system test parameters and sets the required model order.

REDUCE-ORDER inverts the covariance matrix, calculates the static gain of the original transfer function, finds the reduced order model coefficients by optimising a cost function, and finally stores these coefficients in a predefined file.

(a) INITIALISE SYSTEM

The lower level of this subsystem consists of: "INITIALISE-DATA-ARRAYS", "SPECIFY-FILES", "SET-INITIAL-CONDITIONS-VALUES", and "SEND-MESSAGE".

The function of this subsystem is:

* INITIALISE-DATA-ARRAYS: This initialises the data arrays needed for the test.

* SPECIFY-FILES: This lets the operator specify the files that contain data and those needed to store test results.

* SET-INITIAL-CONDITION-VALUES: This sets the initial condition values needed to start the test.

* SEND-MESSAGE: This sends a message to the screen to indicate completion of initialisation.

(b) SET RUN TEST PARAMETERS

The lower level of this subsystem consists of: "GET-TEST-PARAMETERS", and "SET-MODEL-ORDER", its function being:

* GET-TEST-PARAMETERS: This gets the coefficients of the original model and the covariance matrix of the original model.

* SET-MODEL-ORDER: This sets the order of the required model by specifying the original model order and the number of terms to be cancelled.

(c) REDUCE ORDER

This subsystem consists of "INVERT-MATRIX", "CALCULATE-STATIC-GAIN", "OPTIMISE", and "STORE-RESULTS". Their functions are:

* INVERT-MATRIX inverts the covariance matrix used in the calculation of the cost function.

* CALCULATE-STATIC-GAIN calculates the static gain of the original transfer funtion.

* OPTIMISE finds the coefficients of the reduced order model that minimise a pre-specified cost function.

* STORE-RESULTS stores the results in a predefined file.

7.7.6 Model trimming

Fig.7.9 shows the Jackson chart of this program, the design being coded in FORTRAN to run on MULTICs (main frame computer). It can be seen that the software is divided into three subsystems:

* INITIALISE-SYSTEM
* SET-RUN-TEST-PARAMETERS
* TRIM-MODEL

INITILISE-SYSTEM initialises the data arrays, specifies the names of data files, sets initial condition values needed for model order reduction and sends a message to the screen indicating completion of initialisation.

SET-RUN-TEST-PARAMETERS gets system test parameters and sets the required model structure.

TRIM-MODEL inverts the covariance matrix and tests for negligible coefficients by optimising a cost function, and finally stores new coefficients in a predefined file.

(a) INITIALISE SYSTEM

The lower level of this subsystem consists of: "INITIALISE-DATA-ARRAYS", "SPECIFY-FILES", "SET-INITIAL-CONDITIONS-VALUES", and "SEND-MESSAGE". The function of this subsystem is:

* INITIALISE-DATA-ARRAYS: This initialises the data arrays needed for the test.

* SPECIFY-FILES: This lets the operator specify the files that contain data and those needed to store test results.

* SET-INITIAL-CONDITION-VALUES: This sets the initial condition values needed to start the test.

* SEND-MESSAGE: This sends a message to the screen to indicate completion of initialisation.

(b) SET RUN TEST PARAMETERS

The lower level of this subsystem consists of: "GET-TEST-PARAMETERS", and "SET-MODEL-STRUCTURE", its function being:

* GET-TEST-PARAMETERS: This gets the coefficients of the reduced order model and the covariance matrix.

* SET-MODEL-STRUCTURE: This sets the structure of the required model by specifying the reduced model order and the number of terms to be neglected.

(c) TRIM MODEL

This subsystem consists of "INVERT-MATRIX", "OPTIMISE", and "STORE-RESULTS". Their function are:

* INVERT-MATRIX inverts the covariance matrix used in the calculation of the cost function.

* OPTIMISE finds the new coefficients of the reduced order model after trimming that minimise a pre-specified cost function.

* STORE-RESULTS stores the results in a predefined file.

7.8 FIXED DIGITAL CONTROLLER

7.8.1 Overview

Fig.7.10 shows the Jackson chart of the fixed digital controller software system, the design being coded in PASCAL. The software is divided into three subsystems:

* INITIALISE-SYSTEM
* SET-RUN-TEST-PARAMETERS
* CONTROL-SYSTEM-VIA-INT

(a) INITIALISE-SYSTEM: The function of this subsystem is to set the output of the DAC to zero, initialise the "number of samples" counter, initialise the vector interrupt, set up the system devices to enable the operator to communicate with the target system and finally to send a message to the VDU indicating completion of initialisation.

(b) SET-RUN-TEST-PARAMETERS: The function of this subsystem is to get the test parameters, select the analogue input channel, set up the plant model and to set up the selected control criterion.

(c) CONTROL-SYSTEM-VIA-INT: The function of this subsystem is to control the system according to a specified criterion.

7.8.2 Initialise system procedure

The lower level of this subsystem consists of four procedures, namely "STOP-PLANT", "INITIALIZE-CNT-DATA-INT", "SET-UP-SERIAL-COMM-SYS-CONT", and "SEND-MESSAGE-CONT". Note that these procedures may call other (lower level) procedures. Their function are:

(a) STOP-PLANT: This sends a command signal to stop the plant as soon as power is switched on.

(b) INITIALISE-CNT-DATA-INT: This initialises the sample counter, the data space arrays, and the vector interrupt.

(c) SET-UP-SERIAL-COMM-SYS-CONT: This initialises the PIT, the USART, and the PIC. A message is then sent to the screen to indicate completion of initialisation and show that the system is ready for communication with the operator.

(d) SEND-MESSAGE-CONT: This sends a message to the screen explaining the function of the main program.

7.8.3 Set run test parameters procedure

The lower level of this subsystem consists of four procedures, namely "GET-TEST-PARAMETERS-CONT", "SET-UP-ANALOGUE-SYS", "SET-CONTROL-CRITERION", and "SET-PLANT-MODEL". These deal with the following:

(a) GET-TEST-PARAMETERS: This gets the test parameters specified by the operator, the test signal, and the sampling rate.

(b) SET-UP-ANALOGUE-SYS: This sets the analogue system by selecting the input measurement channel.

(c) SET-PLANT-MODEL: This enables the operator to specify the type of plant model and to enter the coefficients of the plant transfer function.

(d) SET-CONTROL-CRITERION: This sends a list of options of the available criteria and then sets the controller according to the criterion specified by the operator.

7.8.4 - Control system via int procedure

This procedure enables the processor hardware interrupt and then waits for a hardware interrupt to occur. The control algorithm is executed every time the interrupt signal is activated. Fig.7.11 shows a representation of the interrupt procedure, and Fig.7.12 shows the corresponding Jackson diagram. The interrupt procedure, when activated, measures the plant output through the A/D converter, computes the control signal, issues the control signal through the D/A converter, increments the sampling counter, and returns to the main program.

7.9 SELF-TUNING CONTROLLER

7.9.1 Overview

Fig.7.13 shows the Jackson diagram of the self-tuning controller software system, the design being coded in PASCAL. The software is divided into three subsystems:

* INITIALISE-SYSTEM
* SET-RUN-TEST-PARAMETERS
* RUN-SELF-TUNING

(a) INITIALISE-SYSTEM: The function of this subsystem is to initialise the number of samples counter, initialise the vector interrupt, set up the system devices to enable the operator to communicate with the target system and finally to send a message to the VDU to indicate completion of initialisation.

(b) SET-RUN-TEST-PARAMETERS: The function of this subsystem is to get the test parameters, select the analogue input channel, set up the selected control criterion, and set the identification scheme selected.

(c) RUN-SELF-TUNING: The function of this subsystem is to set the initial conditions required for self-tuning and then control the plant according to a specified criterion using a self-tuning technique.

7.9.2 - Initialise system procedure

The lower level of this subsystem consists of four procedures, namely "STOP-PLANT", "INITIALISE-CNT-DATA-INT", "SET-UP-SERIAL-COMM-SYS-CONT", and "SEND-MESSAGE-CONT". Note that these procedures may call other (lower level) procedures. The function of these procedures are:

(a) STOP-PLANT: This sends a command signal to stop the plant as soon as the power is switched on.

(b) INITIALISE-CNT-DATA-INT: This initialises the sample counter, the data space arrays, and the vector interrupt.

(c) SET-UP-SERIAL-COMM-SYS-ONLINE: This initialises the Programable Interval Timer (PIT), the Universal Synchronous/Asynchronous Receiver Transmitter (USART), and the Programable Interrupt Controller (PIC). A message is then sent to the screen to indicate completion of initialisation and show that the system is ready for communication with the operator.

(d) SEND-MESSAGE: This sends a message to the screen explaining the function of the main program.

7.9.3 Set run test parameters procedure

The lower level of this subsystem consists of four procedures, namely "GET-TEST-PARAMETERS-ONLINE", "SET-UP-ANALOGUE-SYS", "SET-CONTROL-CRITERION", and "SET-IDENTIFICATION-SCHEME". Their functions are detailed below.

(a) GET-TEST-PARAMETERS-ONLINE: This gets operator specified test parameters, the set-point, and the sampling rate.

(b) SET-UP-ANALOGUE-SYS: This sets up the analogue system by selecting the input measurement channel.

(c) SET-CONTROL-CRITERION: This sends a list of options of the available criteria and then sets the controller according to the criterion specified by the operator.

(d) SET-IDENTIFICATION-SCHEME: This puts up a list of identification schemes available to the operator on the VDU display. It then sets the plant model according to the identification scheme selected by the operator.

7.9.4 Run self tuning procedure

The lower level of this subsystem contains two procedures, "SET-INITIAL-CONDITIONS", and "PERFORM-SELF-TUNING-VIA-INT", their details being as follows:

(a) SET-INITIAL-CONDITIONS: Here the initial conditions required to start the identification algorithm are set.

(b) PERFORM-SELF-TUNING-VIA-INT: This procedure enables the processor hardware interrupt, and then waits for a hardware interrupt to occur. The self-tuning control algorithm is executed every time the interrupt signal is activated. Fig.7.14 shows the Jackson chart of the interrupt procedure. This procedure, when

activated, measures the plant output through the A/D converter, computes the control signal, issues the control signal through the D/A converter, identifies the system parameters, increments the sampling counter, and returns to the main program.

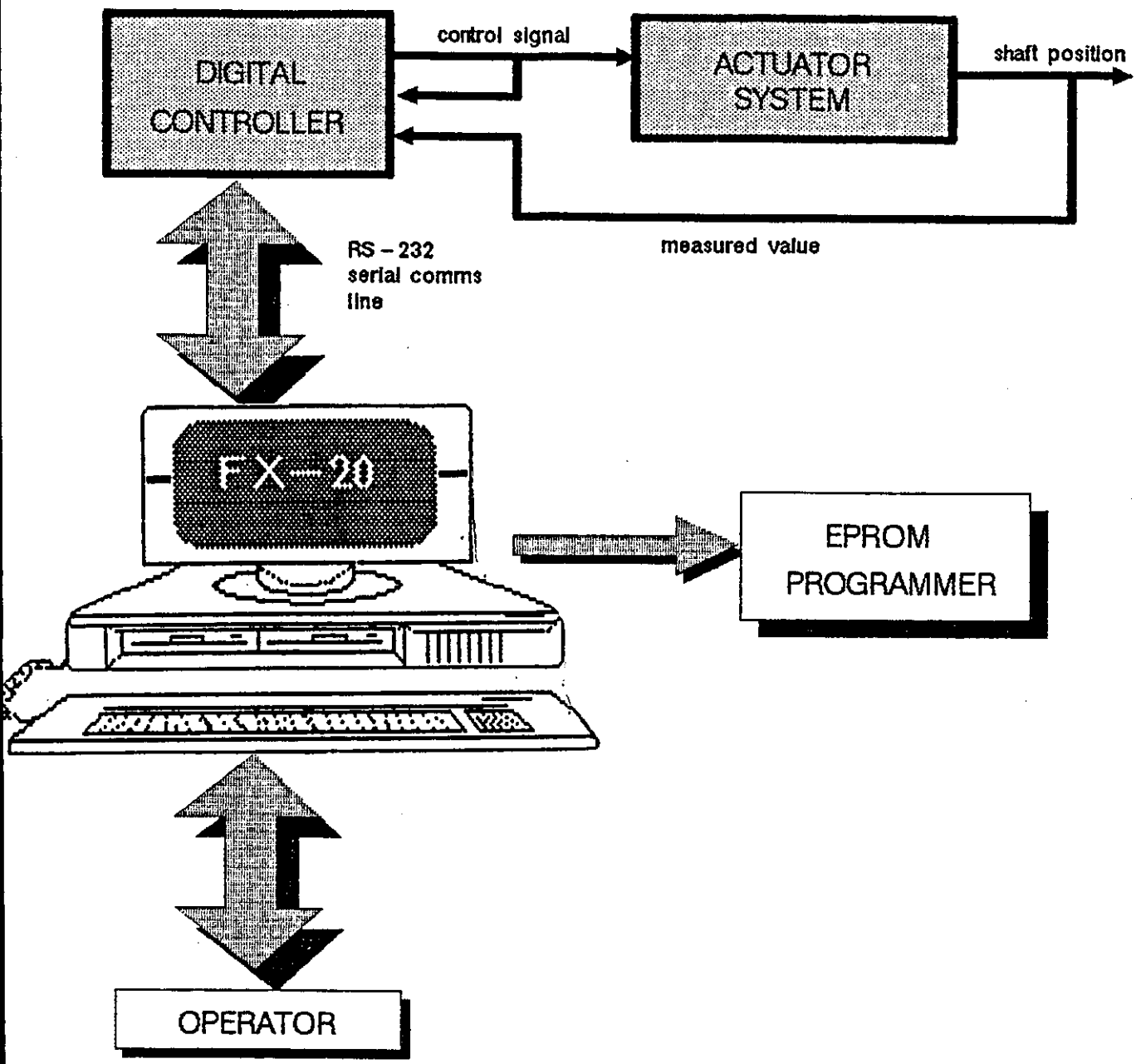


FIG . 7 . 1 THE HOST DEVELOPMENT SYSTEM

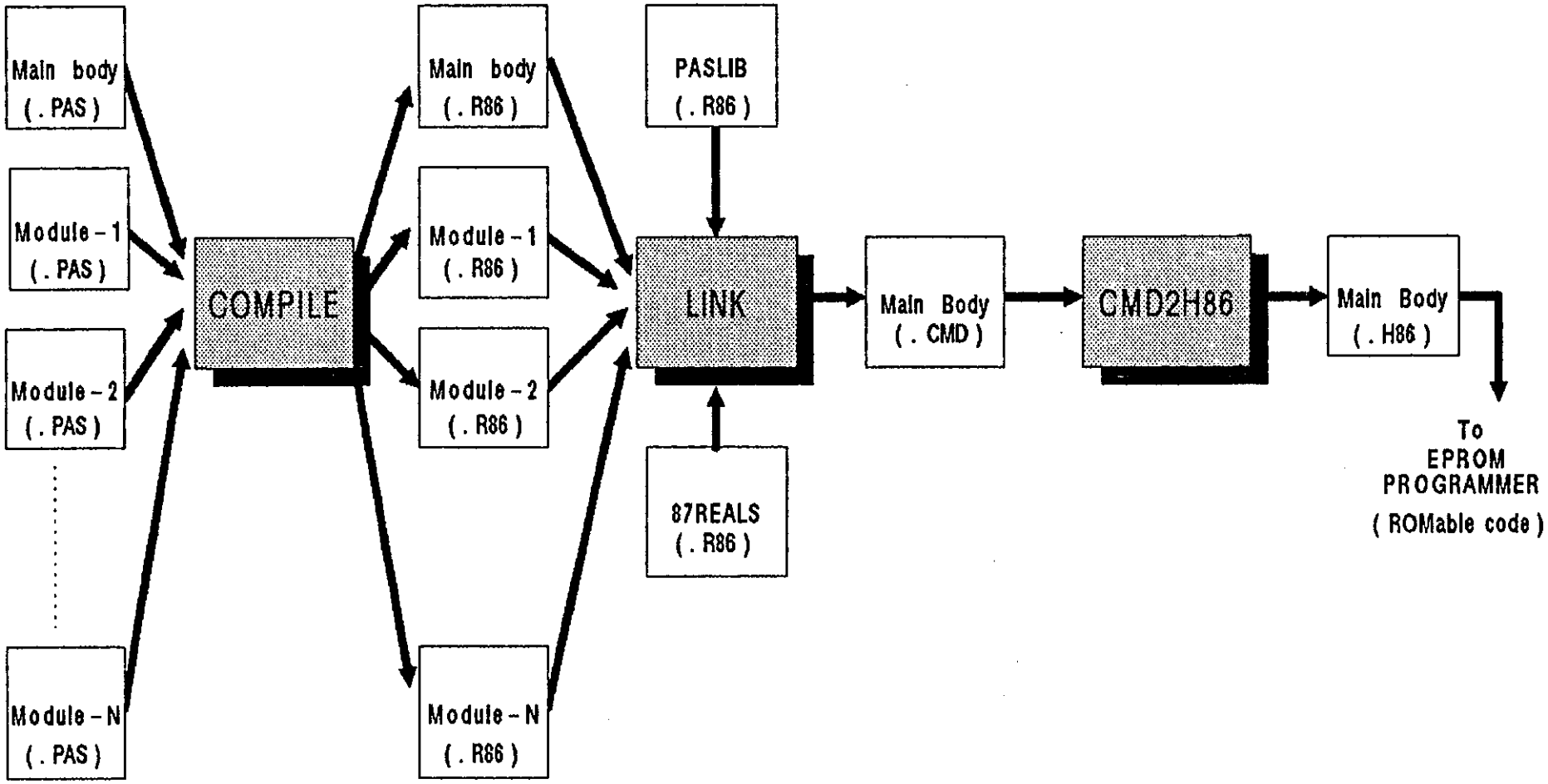


Fig . 7 . 2

SOFTWARE ORGANISATION

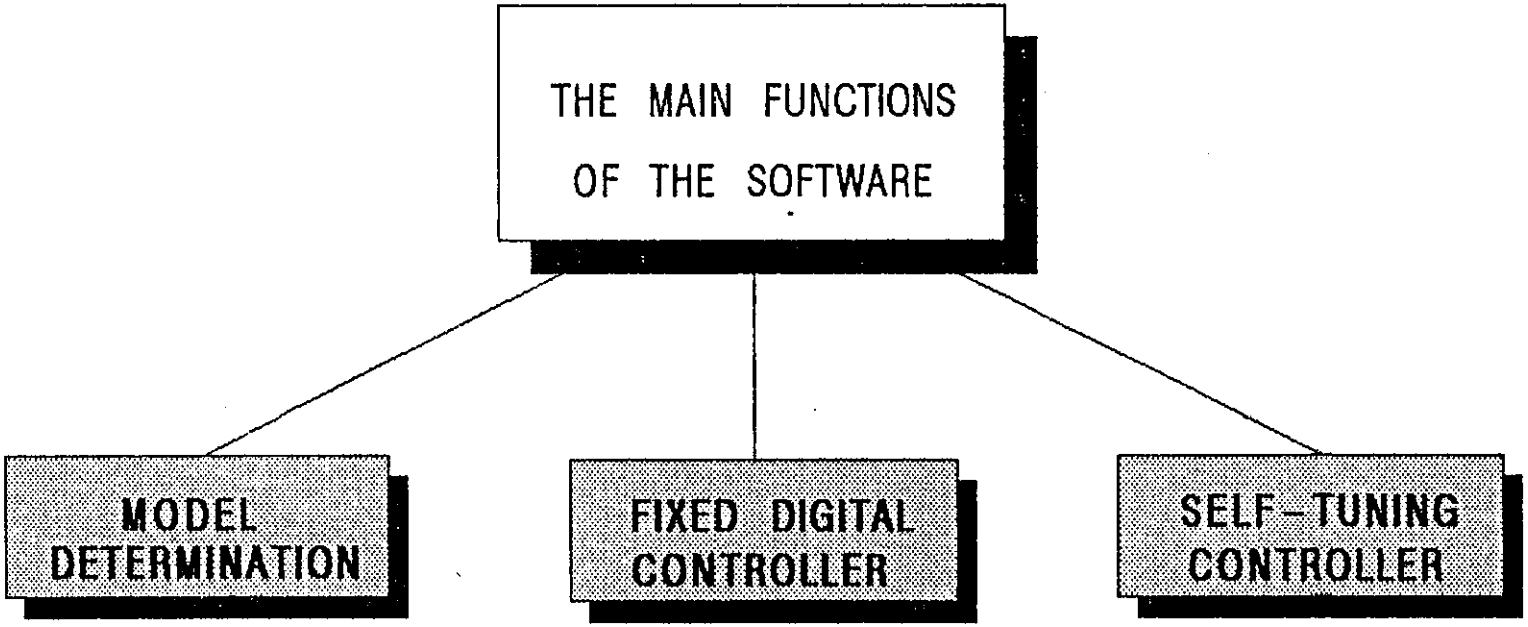


Fig . 7 . 3 THE SOFTWARE OBJECTIVES

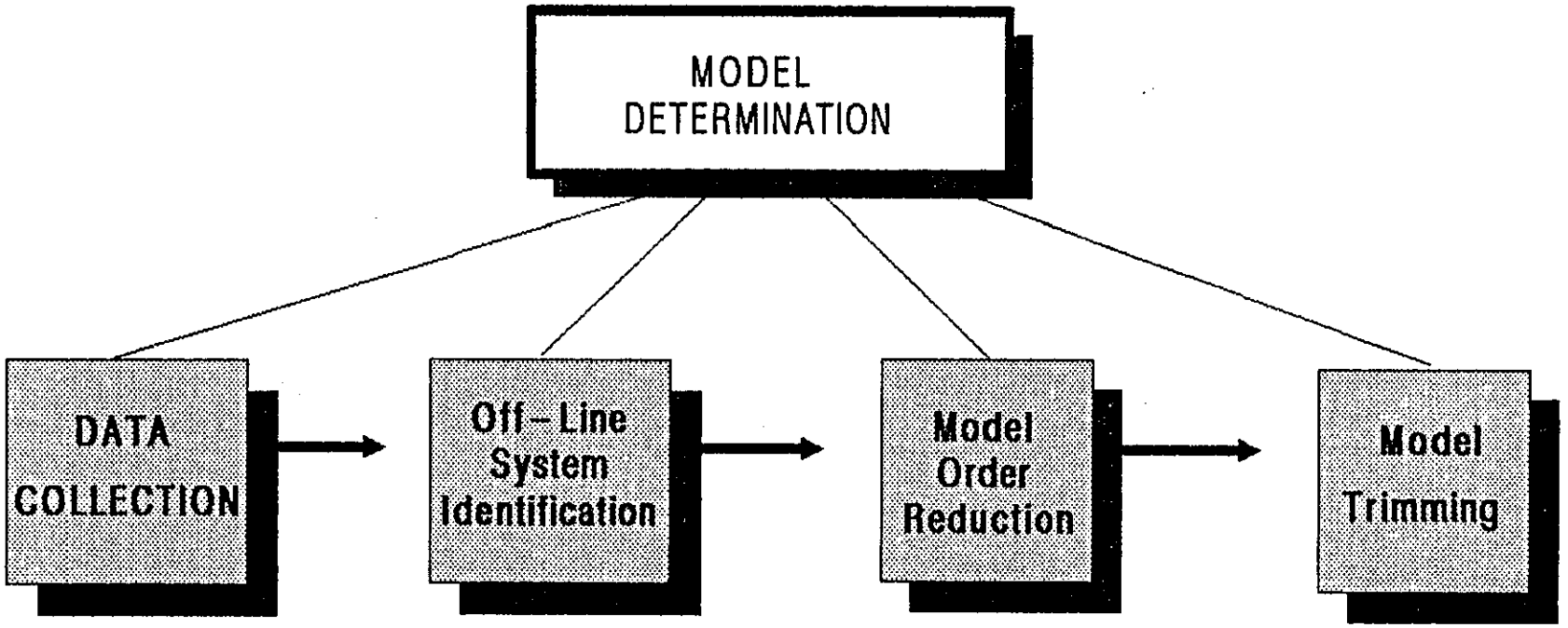


Fig . 7 . 4 MODEL DETERMINATION FUNCTION

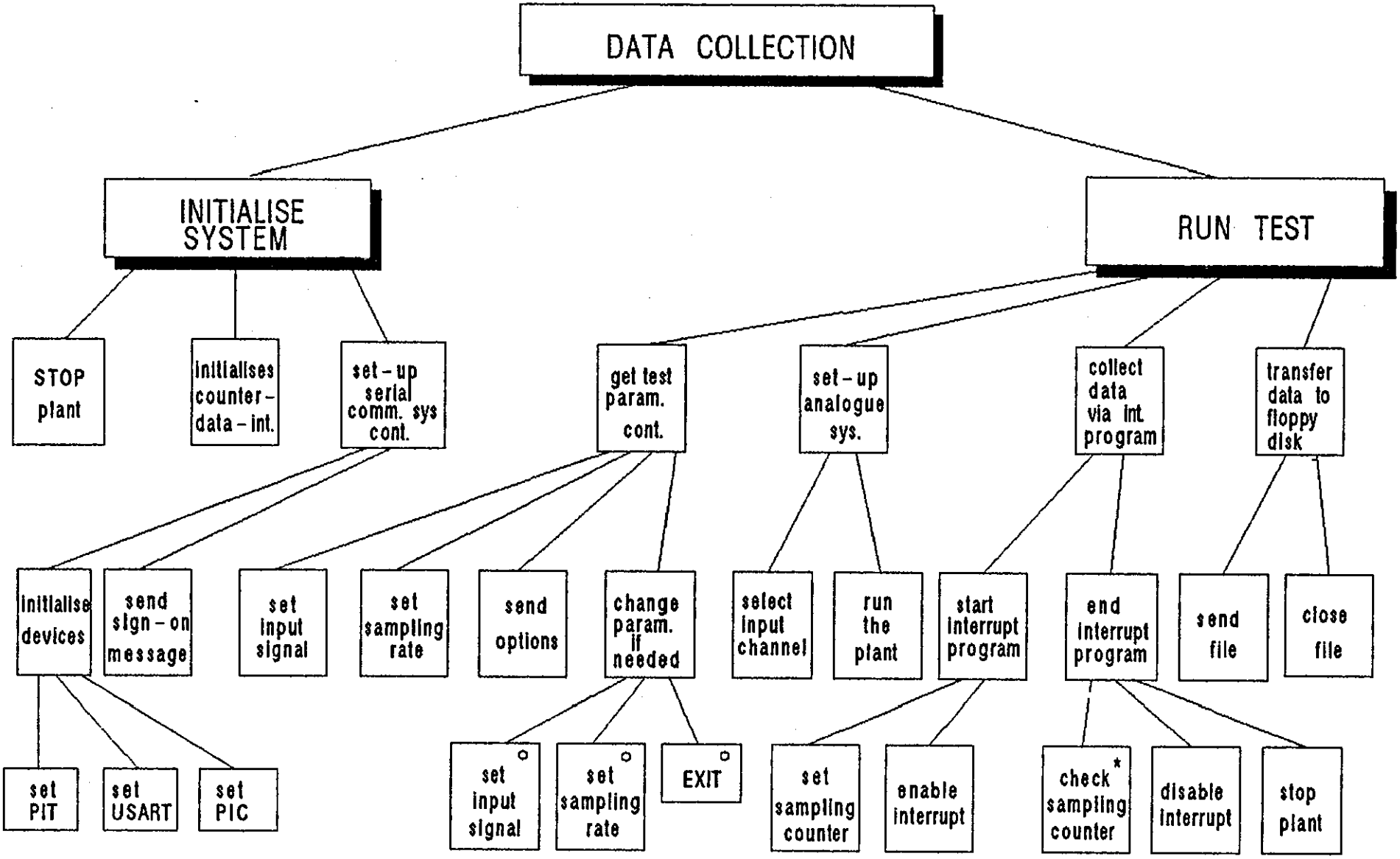


Fig . 7 . 5 JACKSON CHART FOR THE MAIN PROGRAM " DATA-COLLECTION "

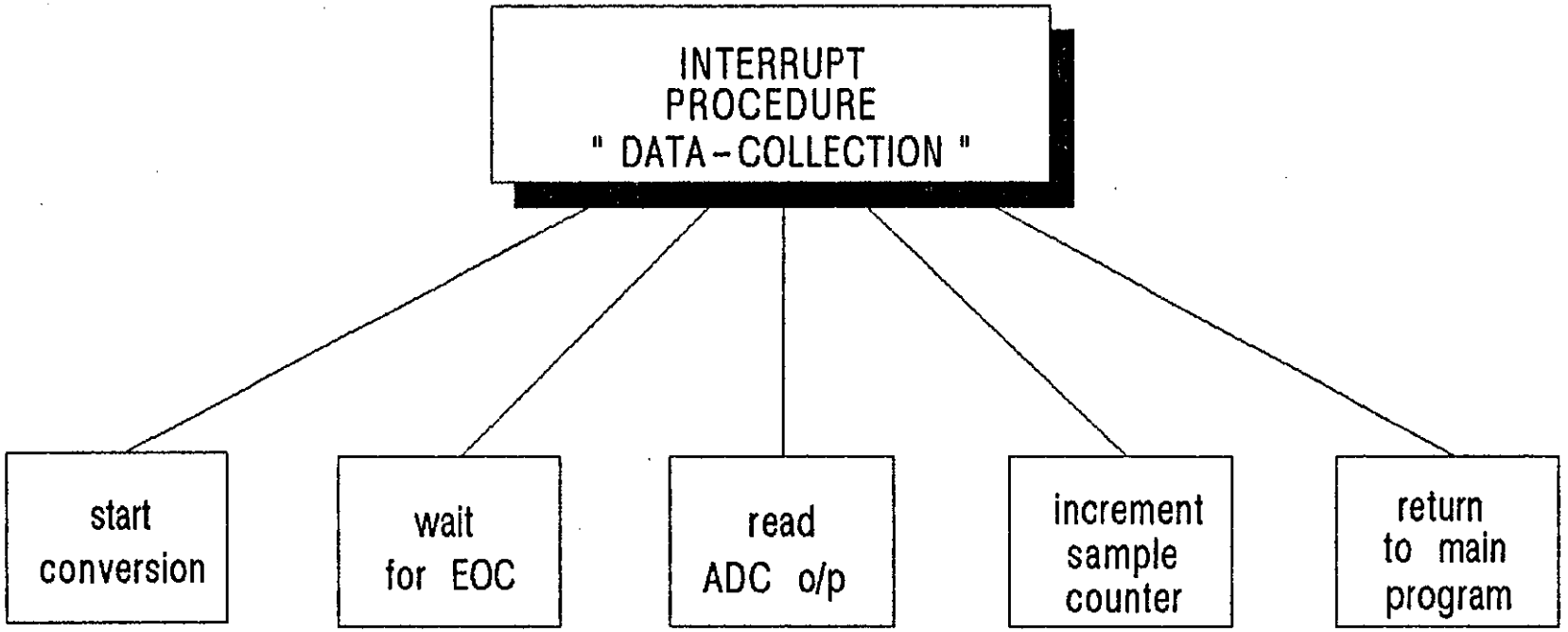


Fig . 7 . 6 JACKSON CHART FOR THE INTERRUPT PROCEDURE

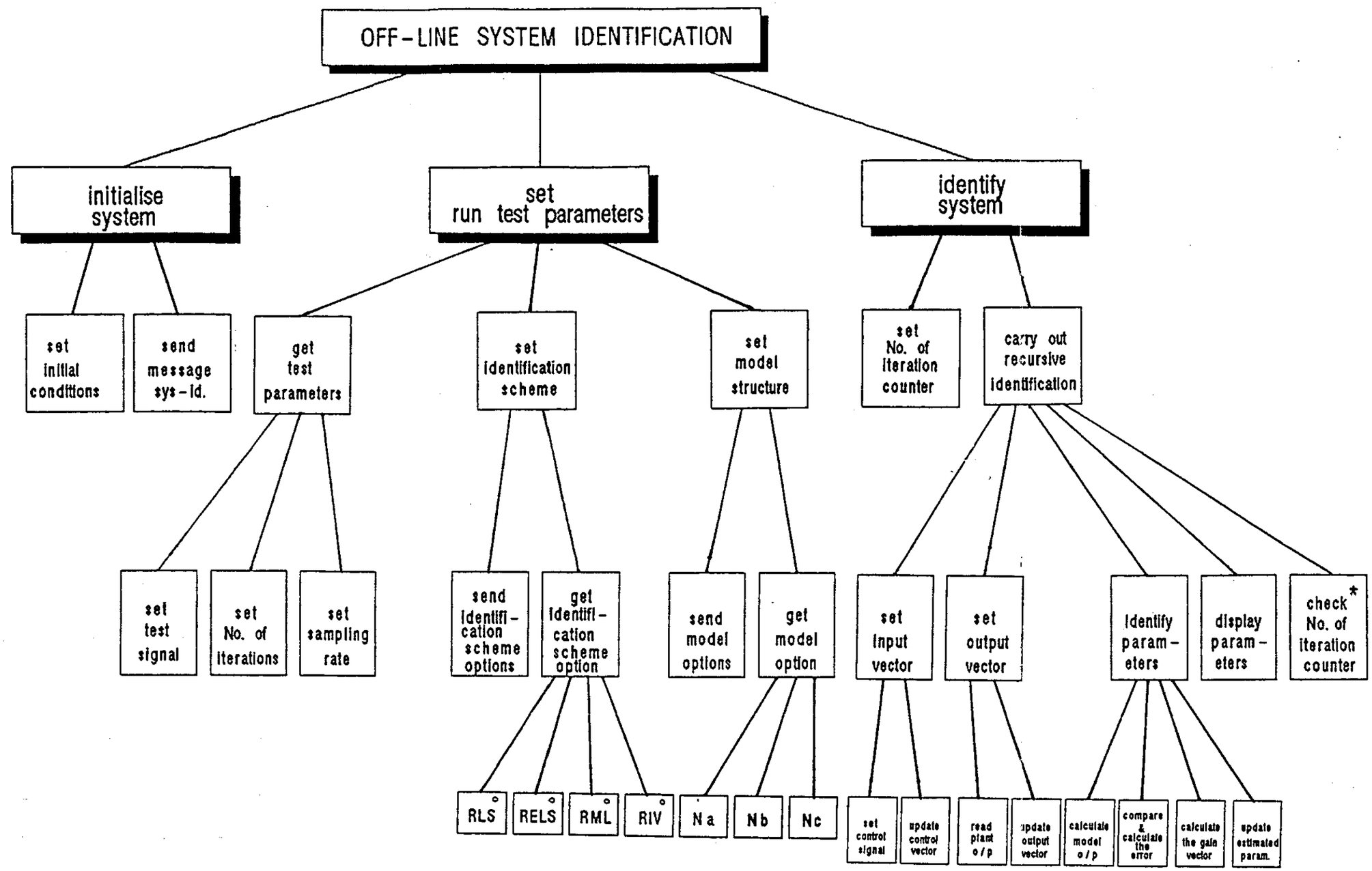
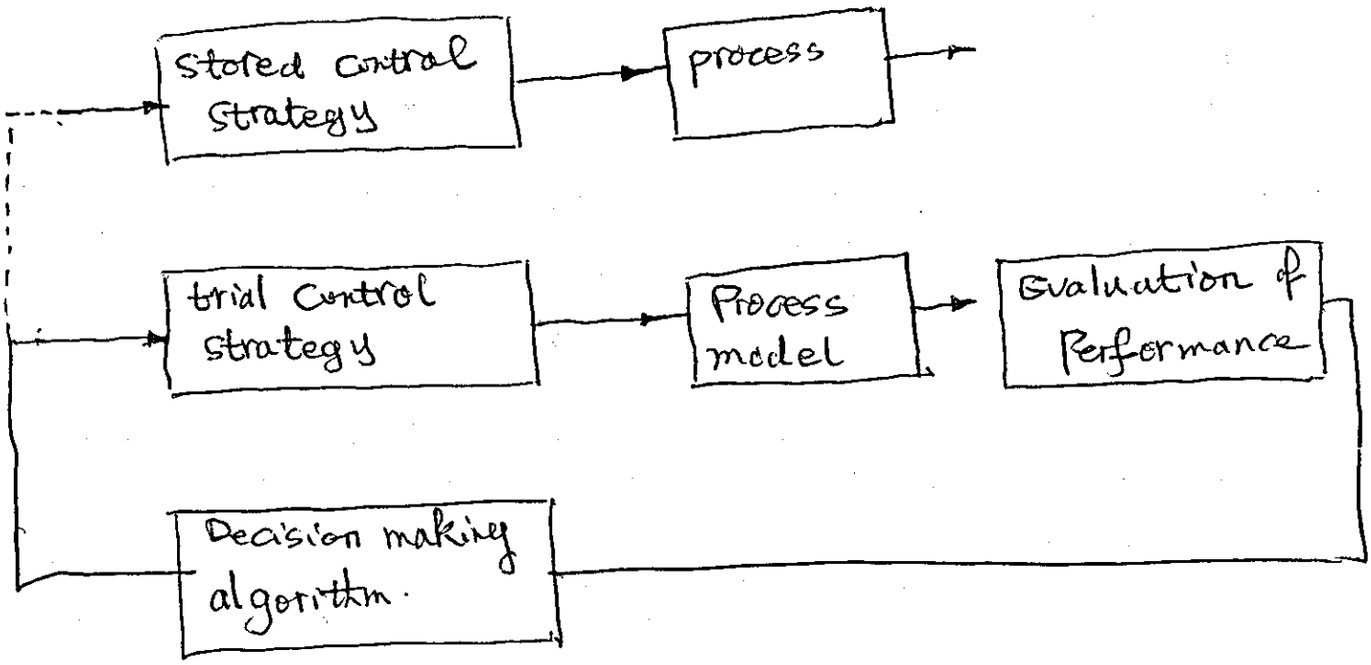


Fig . 7 . 7 THE JACKSON CHART OF THE MAIN PROGRAM " OFF-LINE SYSTEM IDENTIFICATION "

Predictive - iterative Control using a fast process model

Any process for which a fast on-line model exists can be ~~controlled~~ by the technique shown in Figure 12.13



Predictive - iterative Control strategy

Advantages:

(a) No restrictions are imposed on the type of model provided that it satisfies the requirement of being suitable for on-line computation.

(b) A quite inaccurate model will still give good control. This is because at the beginning of each interval of process time T, the model starts off with the current initial conditions. The time T is chosen so that the prediction ability of the model is not overstretched

(c) The difficult problem of control synthesis is avoided and no concessions need to be made to allow the synthesis problem to become tractble. Any criterion whatever can be used in the block marked 'evaluation of performance'. There is no need to be restricted to quadratic criteria.

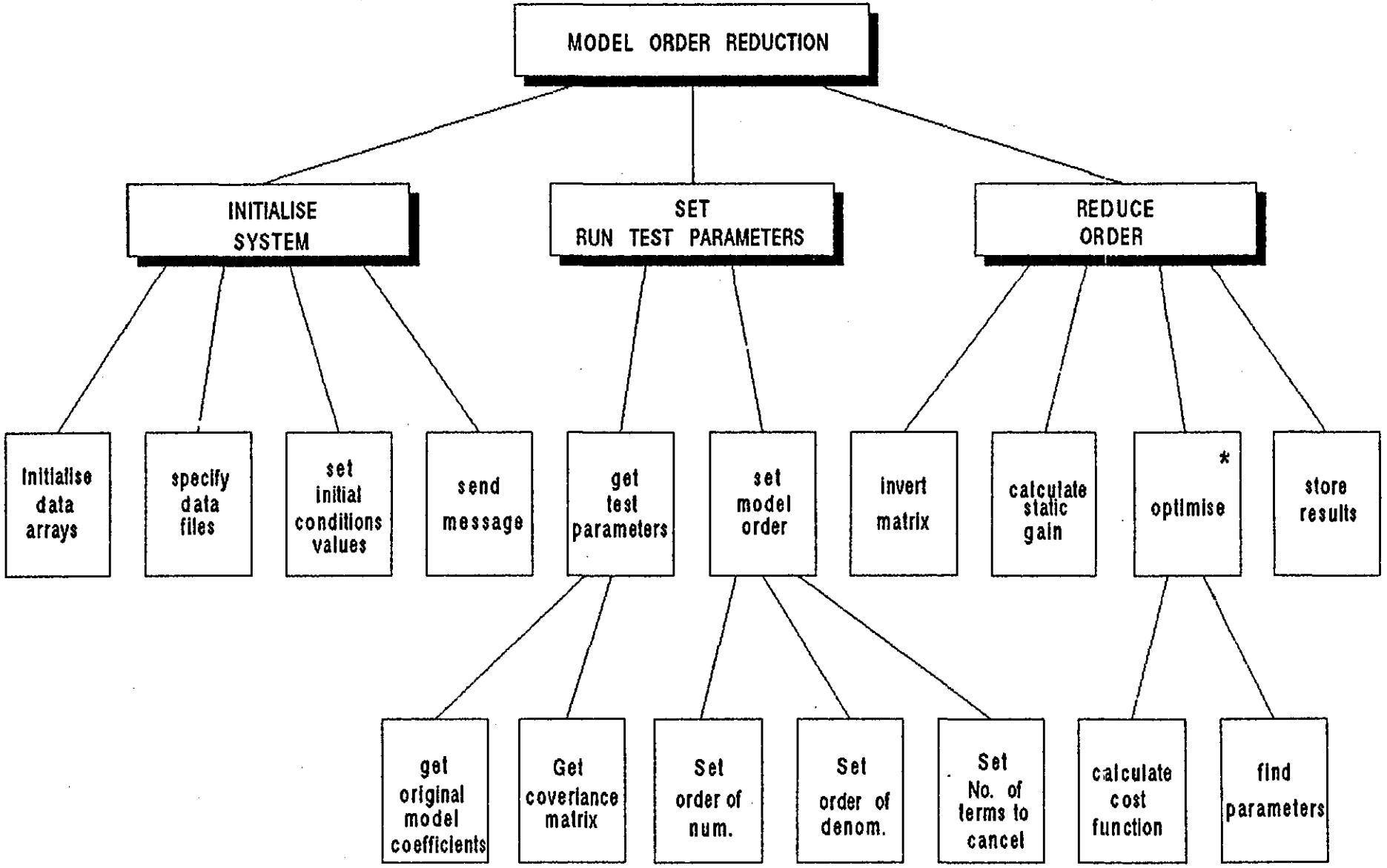


Fig . 7 . 8 JACKSON CHART FOR THE MAIN PROGRAM " MODEL ORDER REDUCTION "

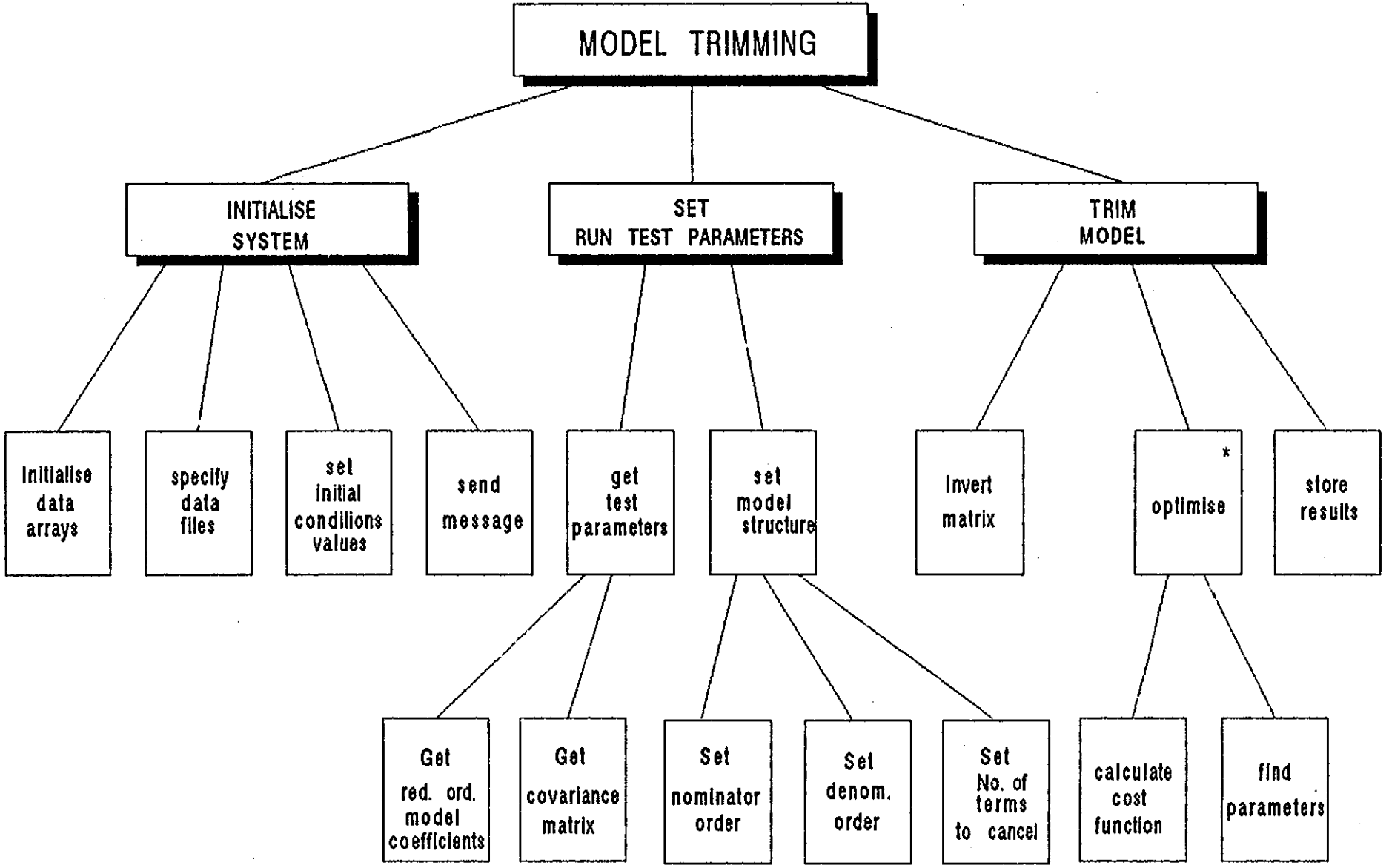


Fig . 7.9 JACKSON CHART FOR THE MAIN PROGRAM " MODEL TRIMMING "

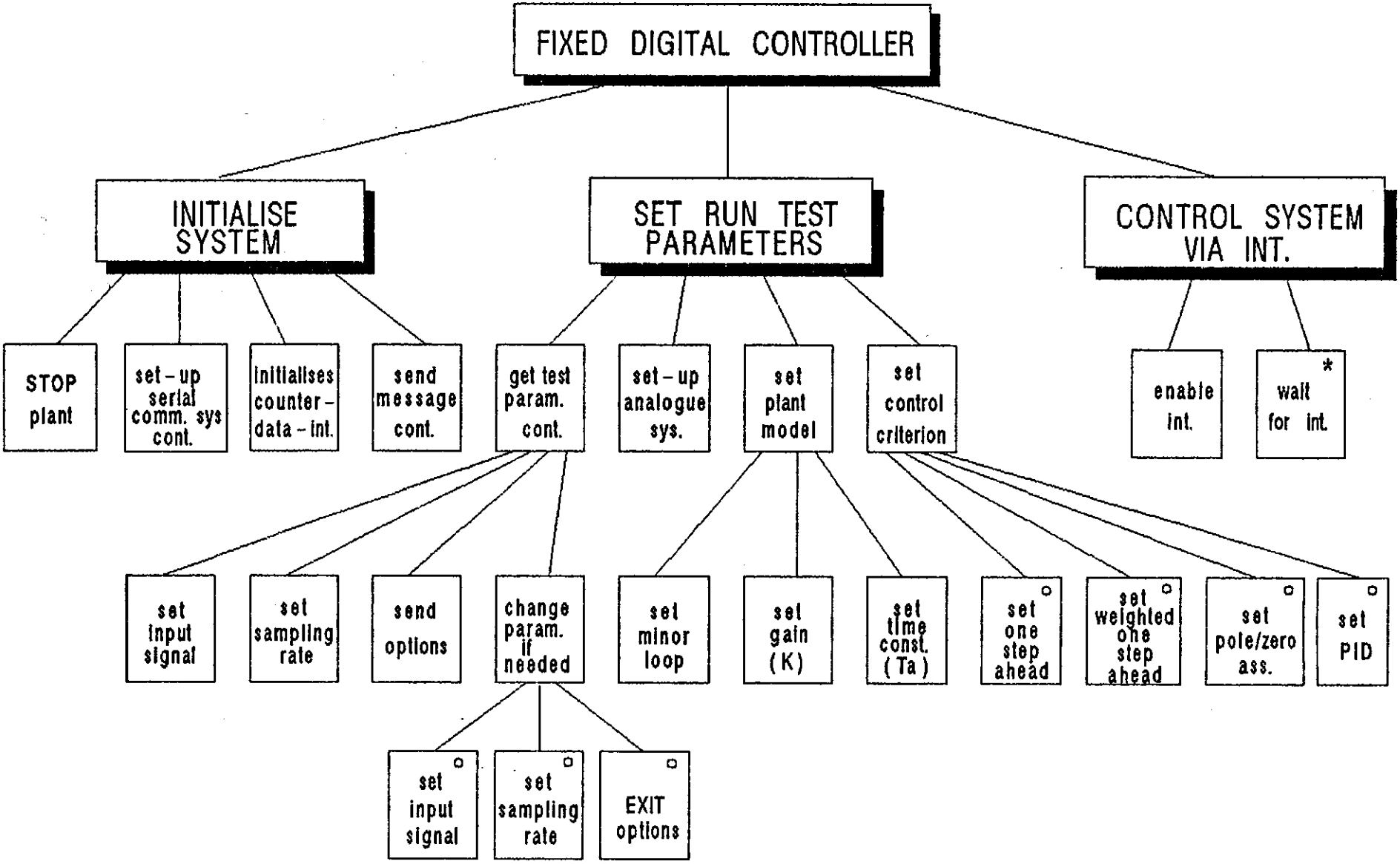


Fig . 7 . 10 THE JACKSON CHART OF THE MAIN PROGRAM " FIXED DIGITAL CONTROLLER "

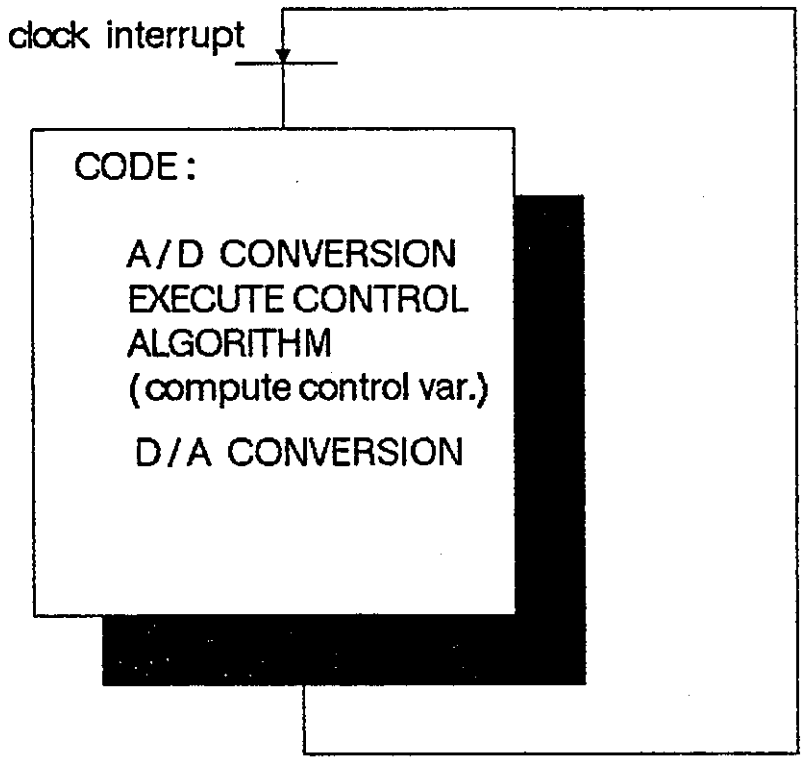


Fig . 7 . 11

REPRESENTATION OF THE INTERRUPT PROCEDURE
THAT IMPLEMENTS THE DISCRETE CONTROLLER .

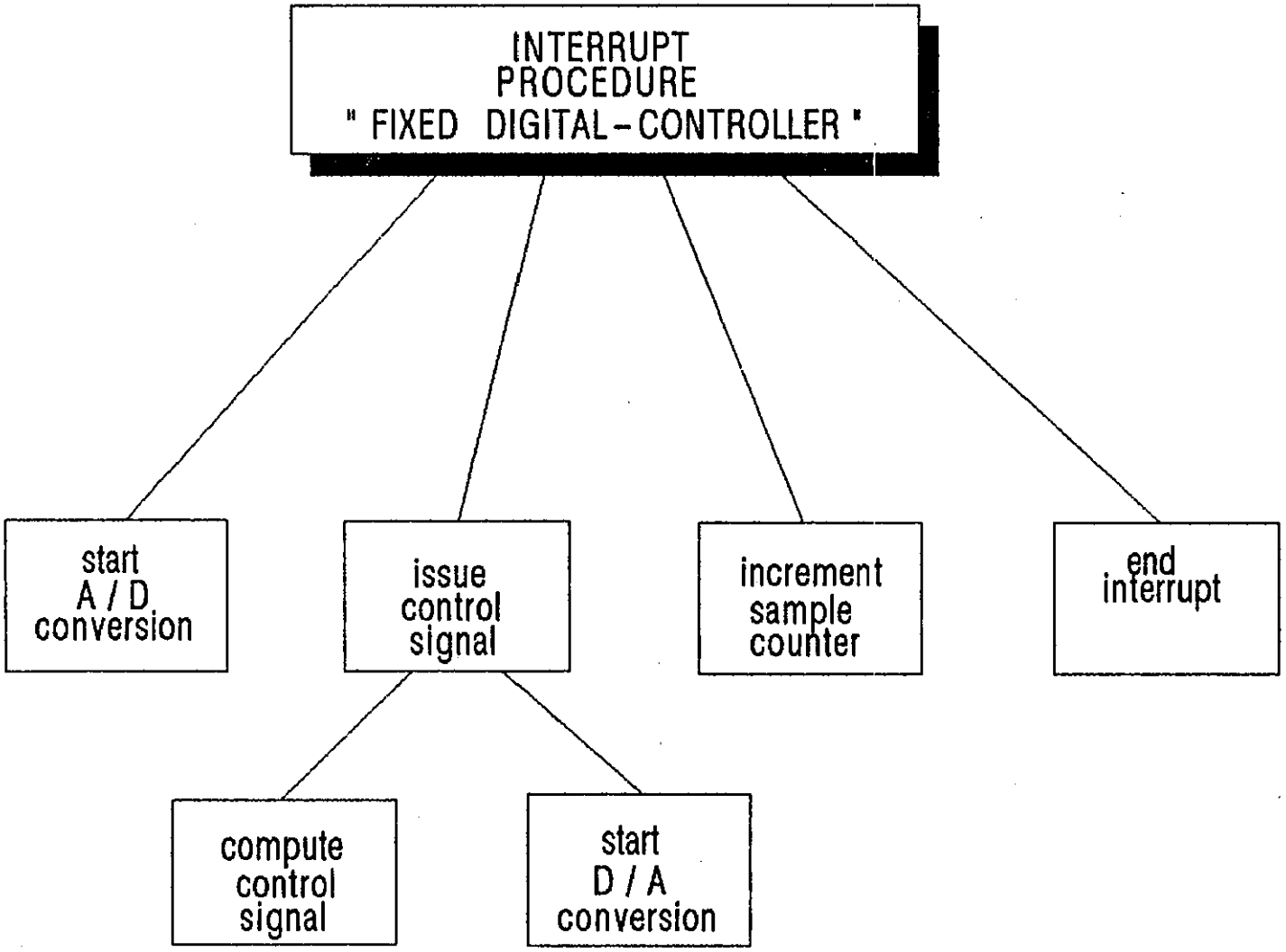


Fig . 7 . 12 THE JACKSON CHART OF THE INTERRUPT PROCEDURE

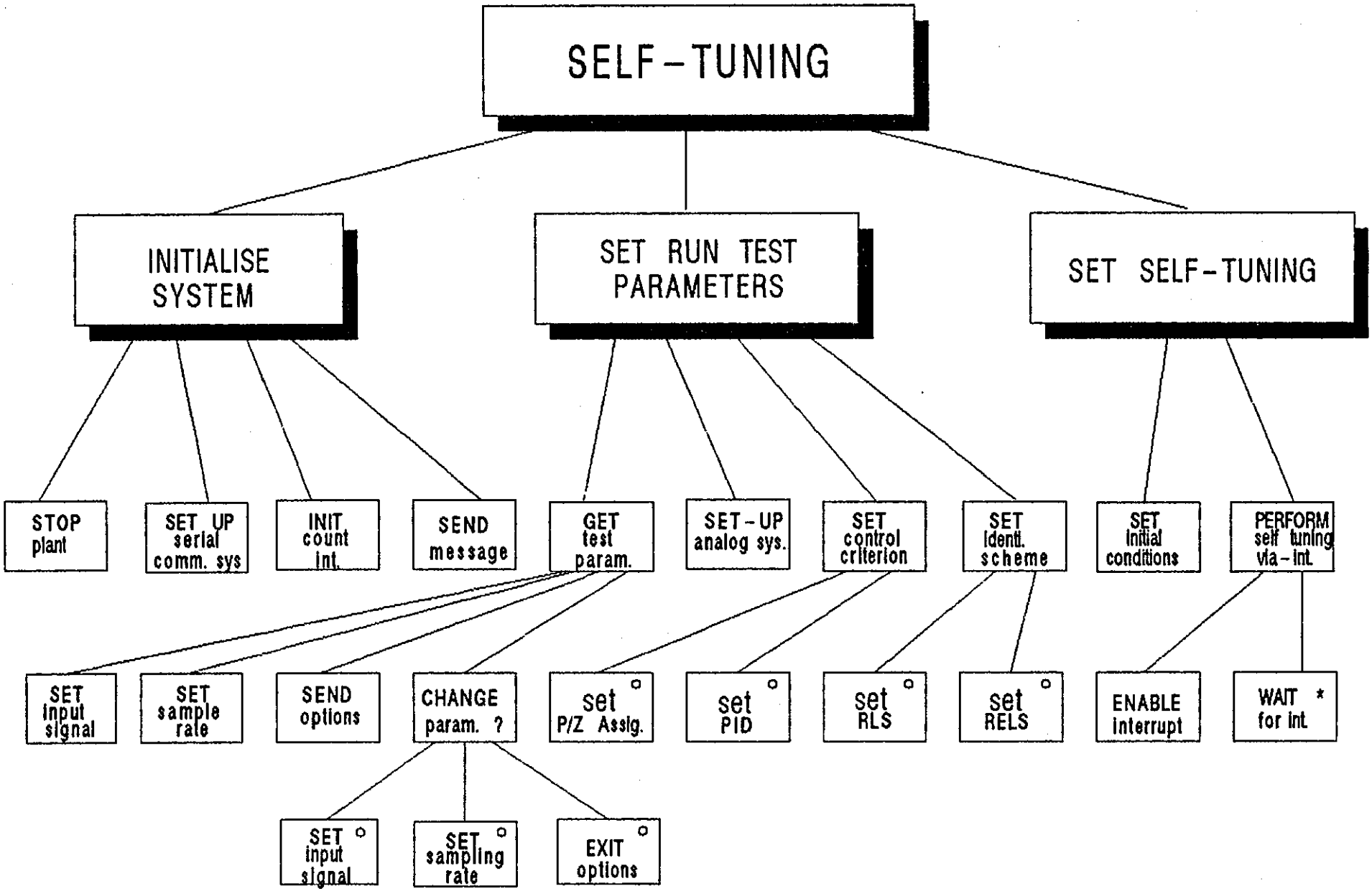


Fig . 7 . 13 Jackson Chart For The Main Program " Self-tuning controller "

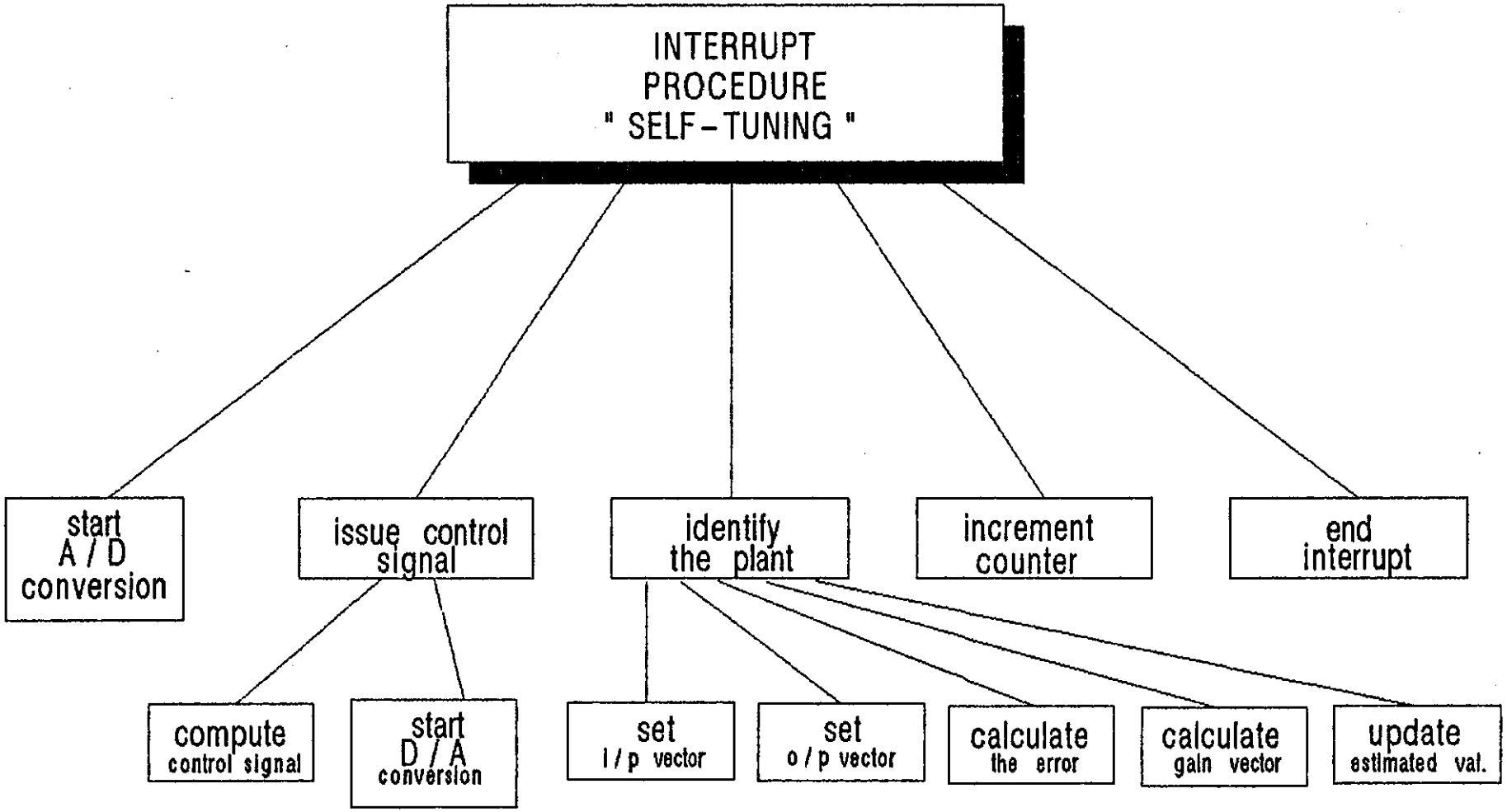


Fig . 7 . 14 THE JACKSON CHART OF THE INTERRUPT PROCEDURE

CHAPTER - 8

8 MODEL DETERMINATION PERFORMANCE - ACTUAL AND SIMULATED

8.1 OVERVIEW

This chapter describes the derivation of a mathematical model of the controlled plant using system identification techniques applied to practical data. From this a reduced order model is determined using an established pole/zero cancellation method. A modified algorithm technique is subsequently used to perform the same task. Both practical and simulated test results are given to demonstrate improvements obtained through the use of this algorithm modification technique. A new model trimming algorithm is introduced, its function being to improve the structure of the transfer function obtained from the reduction process. The theoretical effects produced by this trimming technique on actual data is also shown.

8.2 THE MATHEMATICAL MODEL

Data collected from the plant are used by the system identification process to determine the plant transfer function. These are used to develop a 3rd order model, the estimated transfer function being:

$$G(z^{-1}) = \frac{0.0334 + 0.0081\ z^{-1} + 0.0199\ z^{-2} + 0.0202\ z^{-3}}{1 - 0.588\ z^{-1} - 0.257\ z^{-2} - 0.0918\ z^{-3}} \qquad (8.1)$$

The performance of this model is compared with those obtained previously using time domain (step) and frequency response testing of the actual plant. A comparison of the behaviour of the various models is given in Fig.8.1, these being for a step input test.

8.3 MODEL ORDER REDUCTION

8.3.1 Overview

In this section both practical and simulated data are used to demonstrate the improvement in the pole/zero cancellation model order reduction technique.

8.3.2 Practical results

(a) Pole/Zero cancellation

The 3rd order model (8.1) is reduced to a first order model using the algorithm proposed by Soderstrom, Fig.8.2 shows the step response of the original model and the reduced order model. It is shown that there is a static gain difference between the two responses.

(b) Improved Pole/Zero cancellation

Fig.8.3 shows the step response of both the original and the reduced order models when the improved technique is used. It is clear how the reduced order model retained all the characteristics of the original model. Fig.8.4 shows the step response of the original model compared with the reduced order models when both pole/zero cancellation and the improved techniques are used.

8.3.3 Simulated results

A 2nd order model was simulated to produce data that was used by system identification process to estimate this model. A 4th order model was assumed in the

identification scheme to produce a higher order model than the original model. This model is reduced back to a 2nd order model using the different techniques.

(a) Pole/Zero cancellation

Fig.8.5 simulates the step response of the original and the reduced order models when the pole/zero cancellation technique is used. It is shown that the reduced order model does not retain the static gain of the original model.

(b) Improved Pole/Zero cancellation

Figs.8.6 and 8.7 show a simulation of the original and the reduced order models response to a step input. It is shown that the reduced order model retains the static gain of the original model. Fig.8.8 shows a comparison of all the techniques.

8.4 MODEL TRIMMING

The transfer function produced from model order reduction using the improved technique is

$$G(z^{-1}) = \frac{0.06427 - 0.0208\ z^{-1}}{1 - 0.95652\ z^{-1}} \qquad (8.2)$$

and

$$G(s) = \frac{1 + 0.19565\ s}{1 + 4.4459\ s} \qquad (8.3)$$

The model determined previously using step and frequency response tests is

$$G(s) = \frac{1}{1 + 4.3\ s} \qquad (8.4)$$

and

$$G(z^{-1}) = \frac{0.02273 + 0.02273\ z^{-1}}{1 - 0.95455\ z^{-1}} \qquad (8.5)$$

Notice the structure differences between the models in the z-domain introduced by the small term in the numerator in the s-domain.

The transfer function produced after trimming the model (8.3) is

$$G(z^{-1}) = \frac{0.02174 + 0.02174\ z^{-1}}{1 - 0.95652\ z^{-1}} \qquad (8.6)$$

and

$$G(s) = \frac{1}{1 + 4.4459\ s} \qquad (8.7)$$

It is shown that the structures of (8.6) and (8.7) are very close to that of (8.4) and (8.5).

8.5 DISCUSSION

Test results showed that the improved technique overcame the drawbacks in the method proposed by Soderstrom [3.9]. The model trimming is a further improvement in model order reduction techniques as shown from test results.

The first order model (8.7) adequately represents the controlled plant.

Fig.8.1

Step input response – Comparison

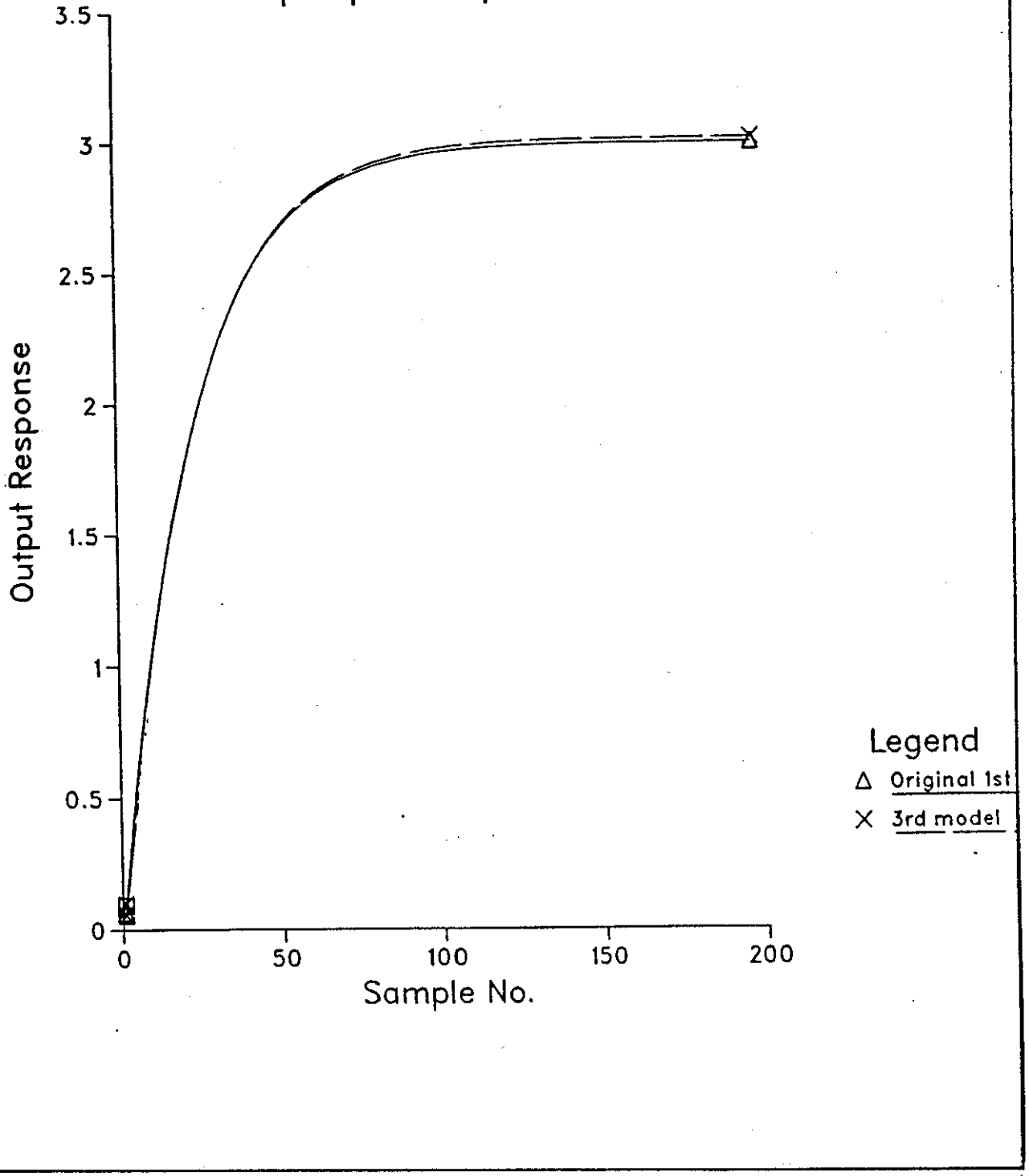


Fig.8.2

Step input Response – Reduced Model Order using P/Z

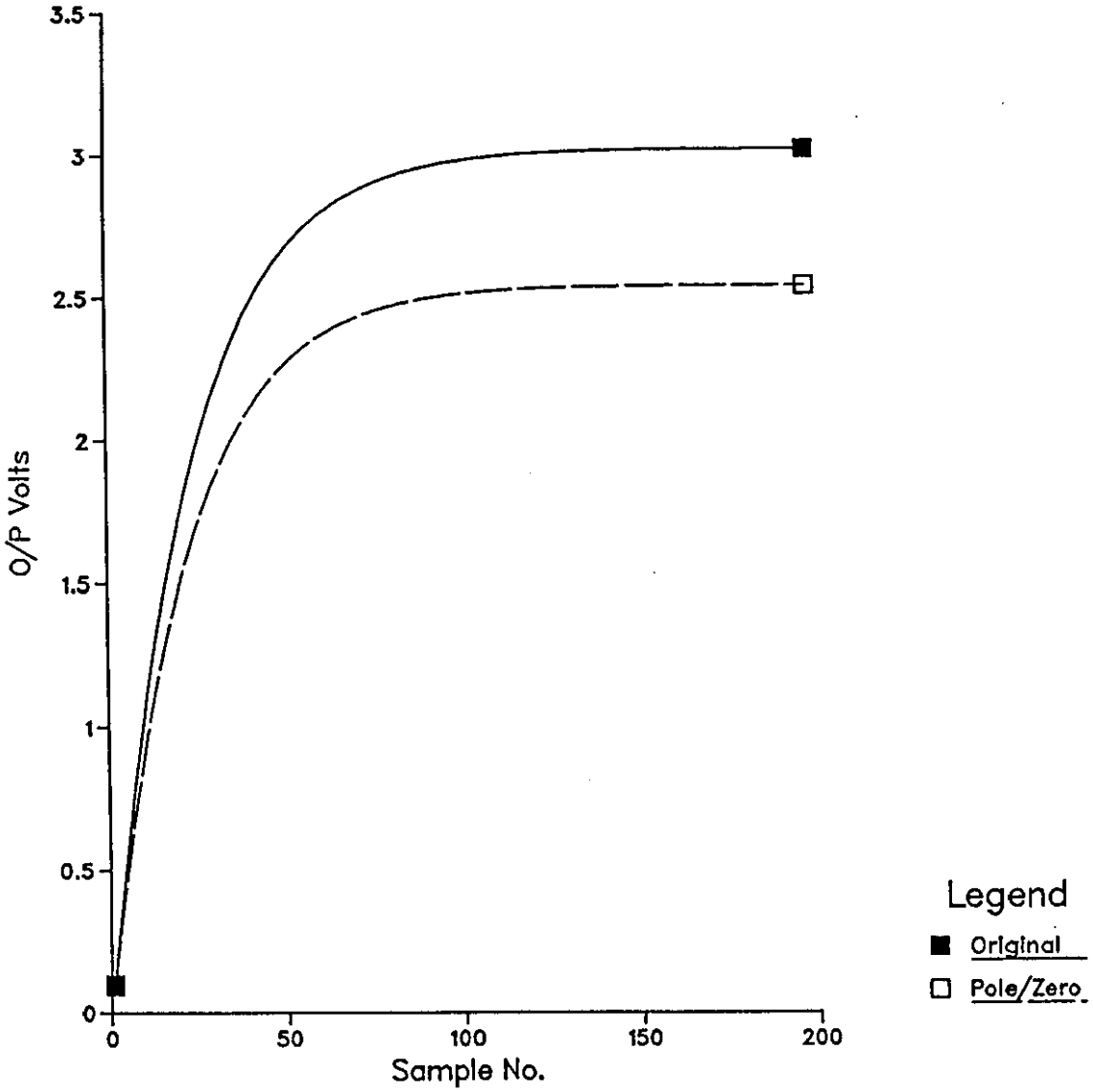


Fig.8.3

Step input response – Reduced Order Model

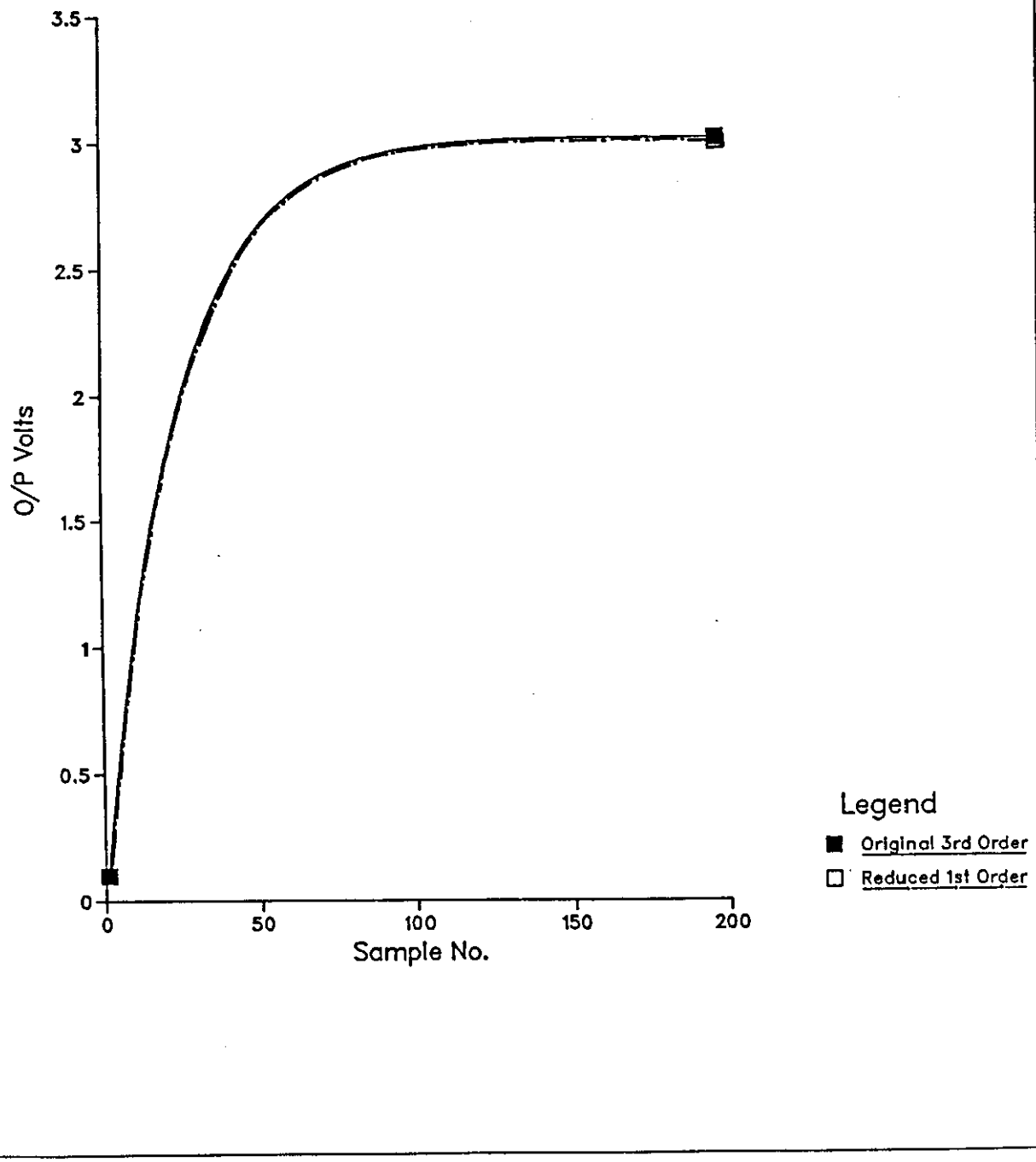


Fig.8.4

Step input response— Comparison

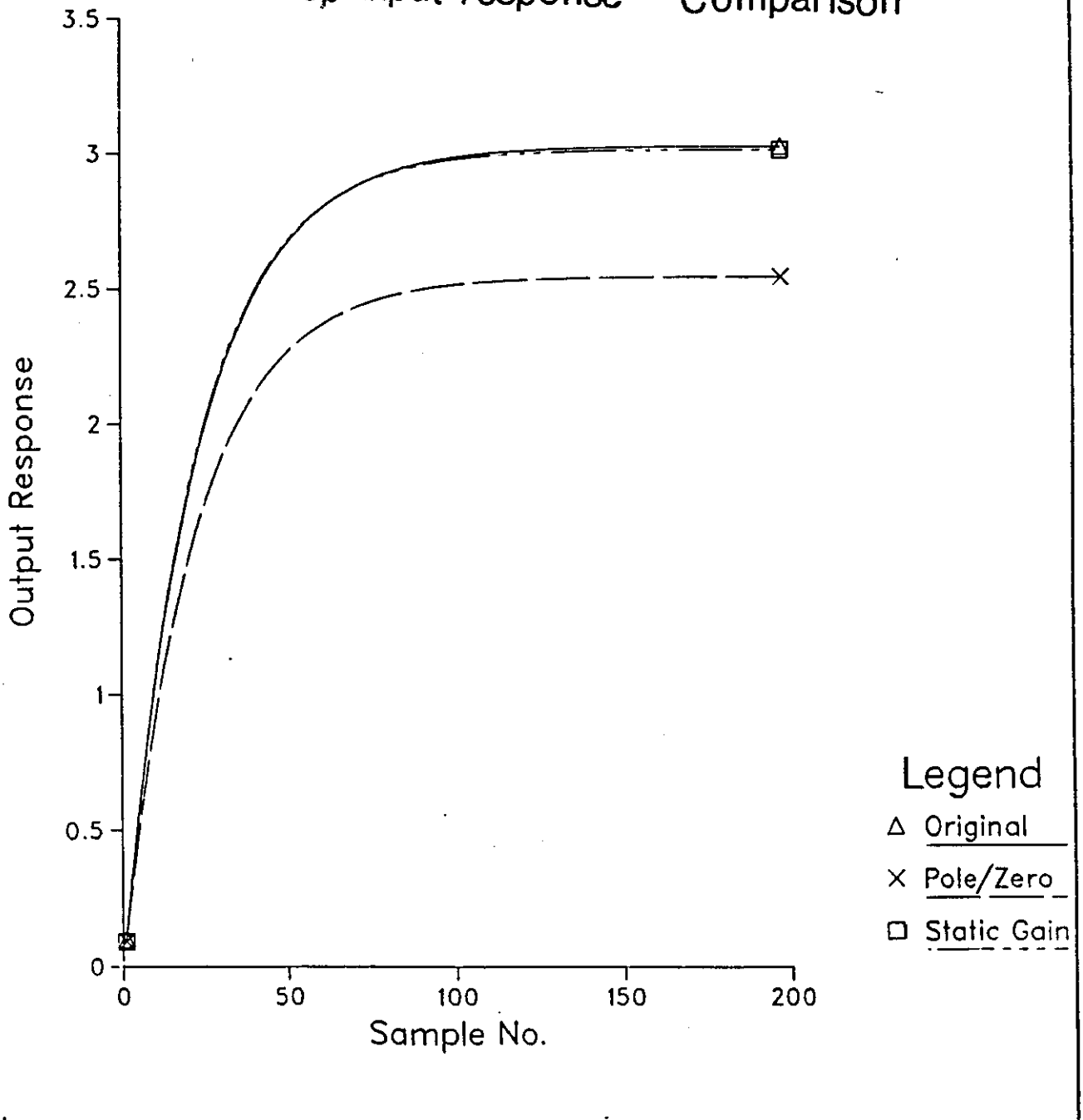


Fig.8.5

Step input response – Reduced model order using Soderstrom tech

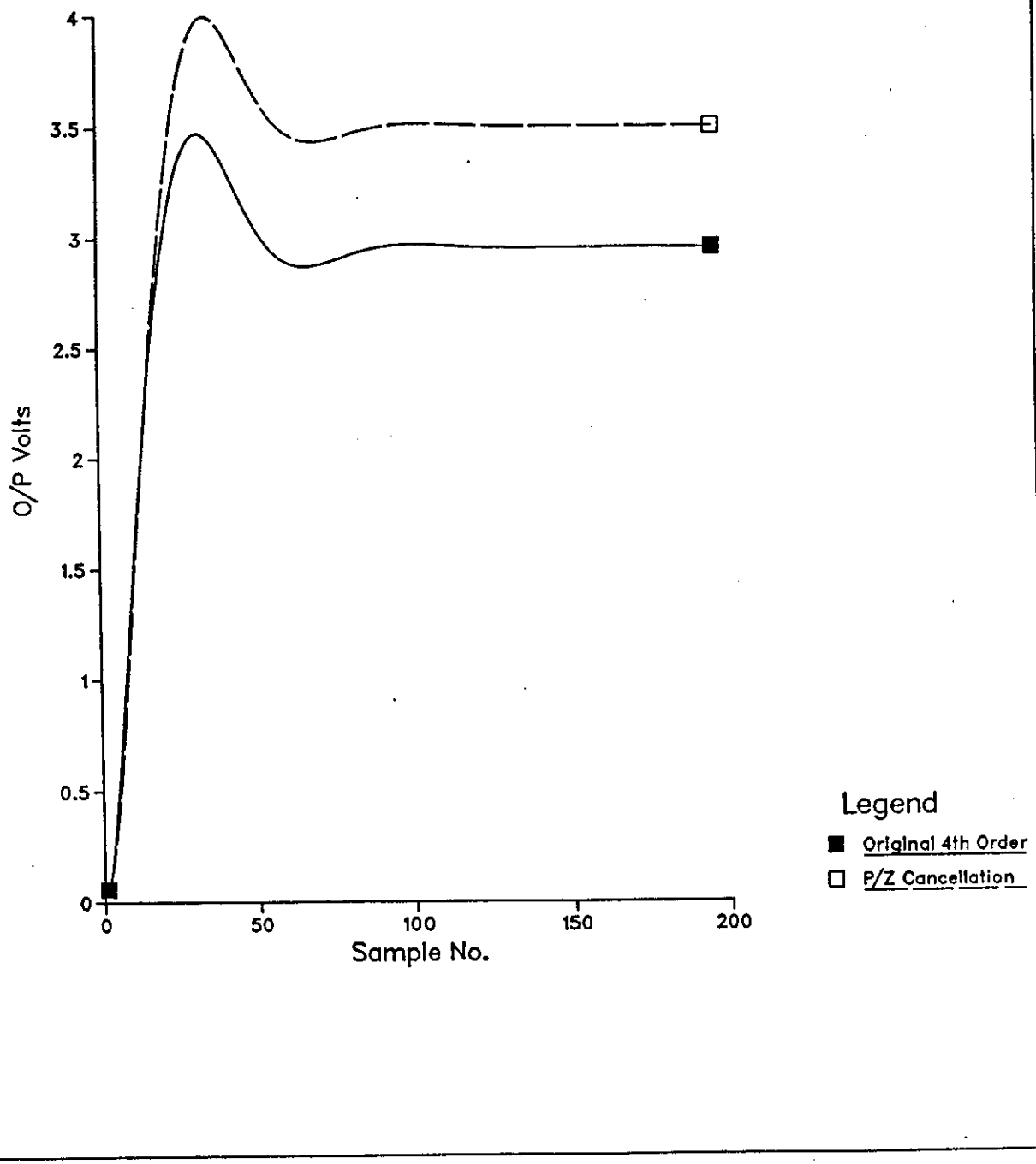


Fig.8.6

Step input response – Reduced model using improved algorithm

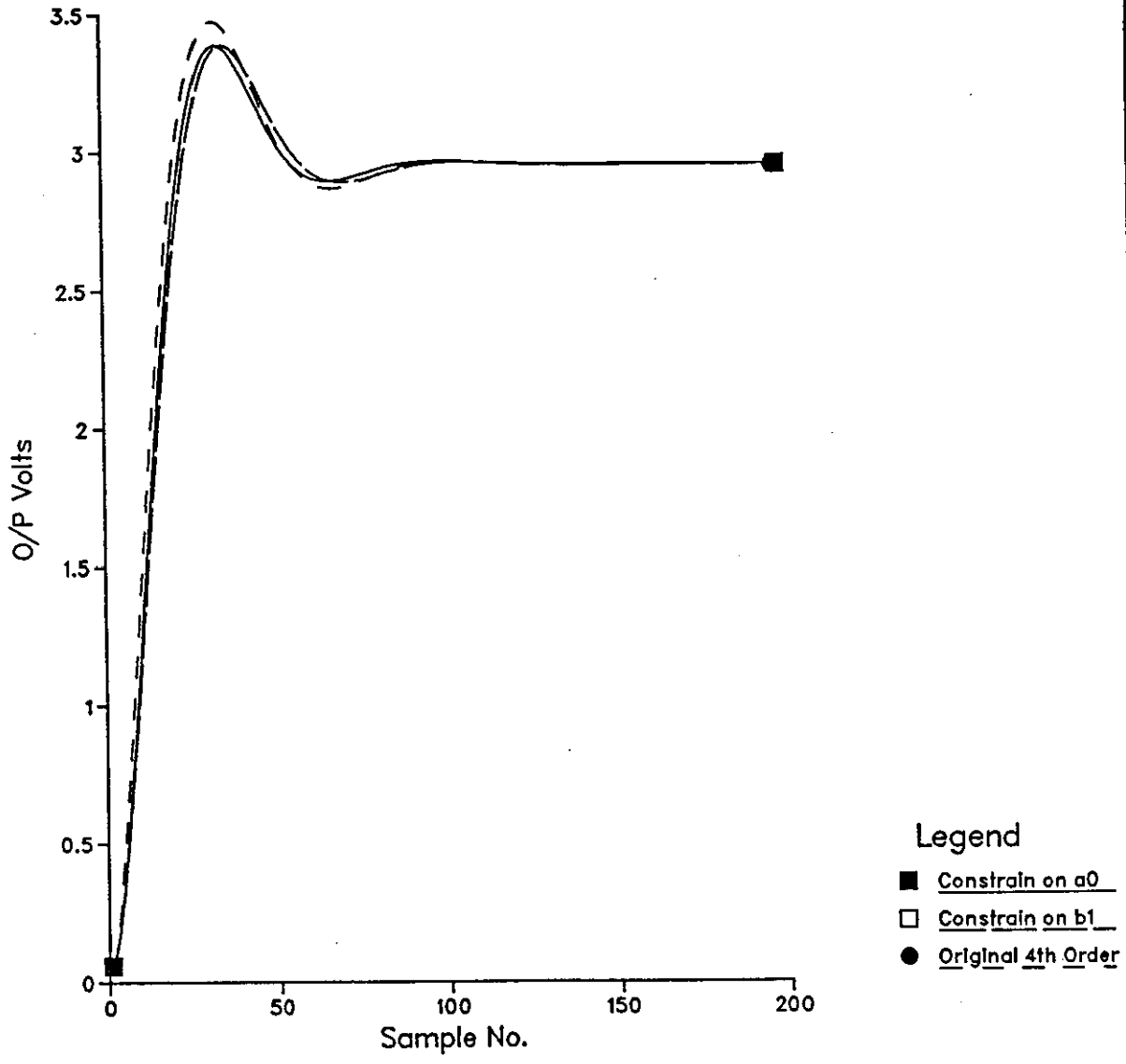


Fig.8.7

Step input response – Reduced model order using Improved Tech.

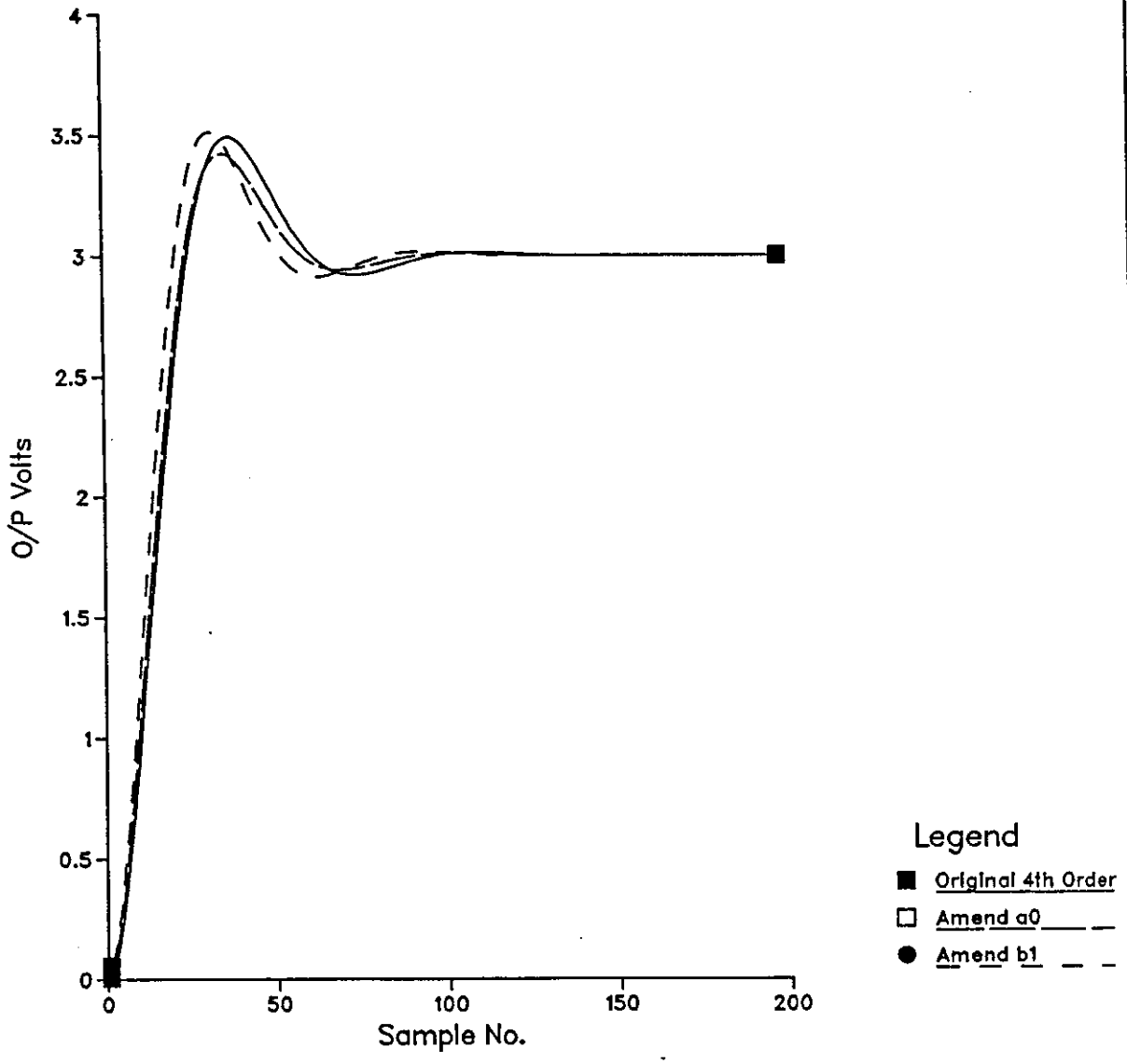
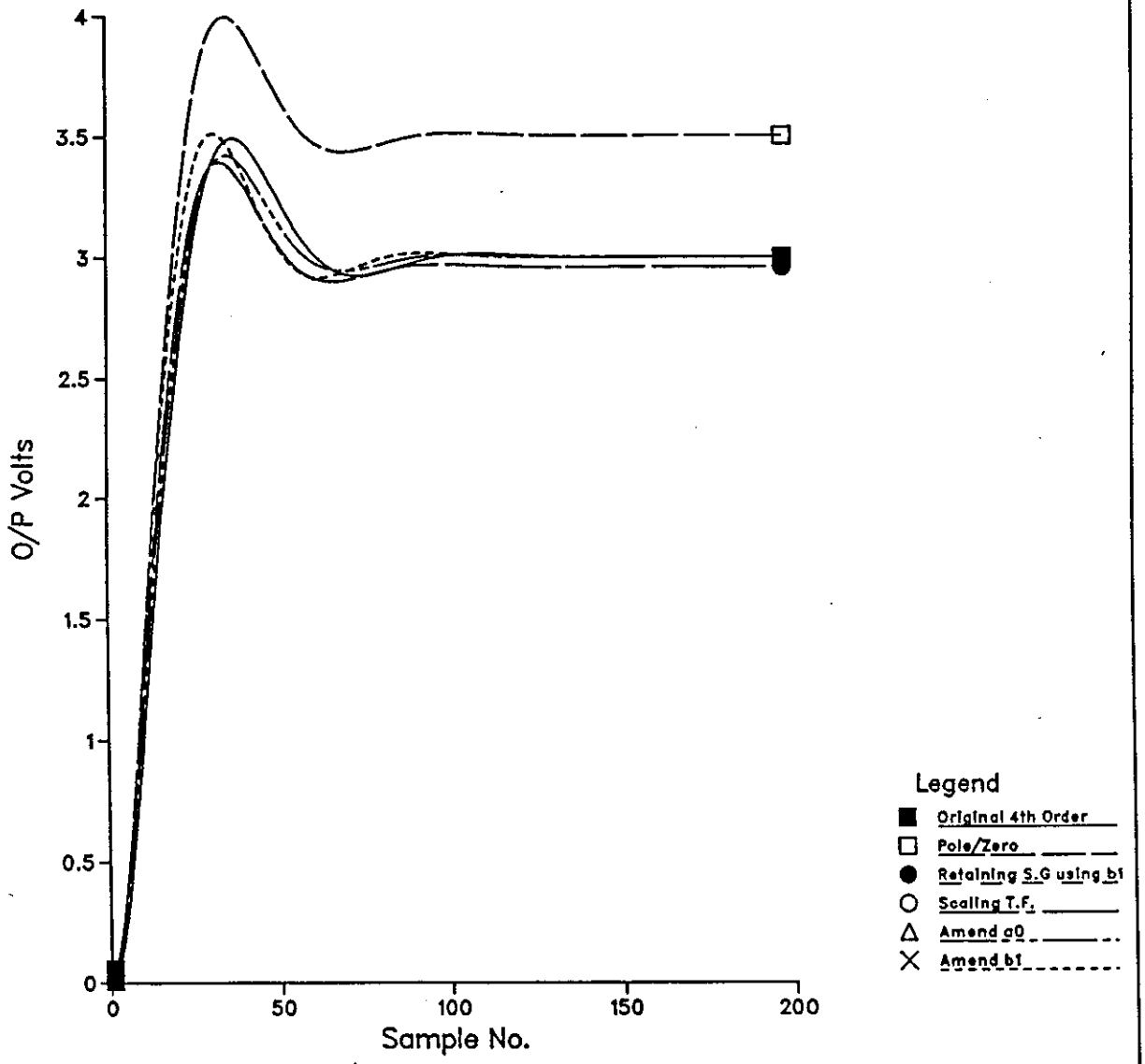


Fig.8.8

Step input response – Comparison of techniques



CHAPTER-9

9 CLOSED LOOP PERFORMANCE TESTS

9.1 OVERVIEW

In this chapter test results for the plant control system using both self-tuning and fixed digital controllers are given. The performance of the plant using these techniques is also compared for different system loading conditions.

9.2 TEST RESULTS - FIXED DIGITAL CONTROLLER

9.2.1 Overview

In this section both simulated and actual test results are given for the plant control system. The following control techniques are used;

* One-Step-Ahead criterion
* Weighted-One-Step-Ahead criterion
* Pole/Zero Cancellation criterion
* PID controller criterion

Simulation is carried out on a Multics mainframe computer whilst the target controller uses an Intel 8088 microprocessor.

9.2.2 Simulation results

(a) One-Step-Ahead Controller

Fig.9.1 shows a simulation of the plant response to a square wave set-point demand, Fig.9.2 being the corresponding control signal. The response is very fast, but this is obtained only by the use of a substantial control effort. Notice the ringing, or ripple, in the control signal. In Fig.9.1 no ripple is observable in the output signal. This is totally misleading as the results are calculated and plotted out on a sample by sample basis. Nothing is said about the intervening period. In fact the output does ripple in the real system, as shown in Fig.9.12.

(b) Weighted-One-Step-Ahead Controller

Figs.9.3 to 9.6 display the simulated plant response to a square wave set-point demand. It can be seen that the response becomes slower as λ gets bigger, but that the corresponding control effort is less. Fig.9.7 shows the relationship between the output response and λ. Increasing the value of λ means more weight is put in minimising u(T) in the performance criterion of equation 4.12 (repeated below).

$$I = [\, y(T+1) - w(T)]^2 + \lambda \cdot u^2(T)$$

(c) Pole/Zero Cancellation Controller

Fig.9.8 is a simulation of the plant response to the same square wave test signal using the pole-zero cancellation control algorithm. The system achieves the rise time, T_r, overshoot, M_P, and settling time, T_s, specified as part of the closed loop

performance criteria. Fig.9.9 shows the corresponding control signal.

(d) PID Controller

Fig.9.10 displays a simulation of the plant response, under PID control, to a square wave set-point signal. The transient response of the system satisfies the required closed loop criteria of rise time, T_r, overshoot, M_p, and settling time, T_s. Fig.9.11 shows the corresponding control signal.

9.2.3 Practical results

(a) One-Step-Ahead Controller

Figs.9.12 and 9.13 show the O/P response of the plant to the square wave reference signal. The response is oscillatory, appearing to limit cycle. The choice of the sample time affects the stability of the system, as can be seen from Fig.9.13. Here limit cycling is still present but its amplitude is significantly reduced.

(b) Weighted-One-Step-Ahead Controller

Figs.9.14 and 9.15 show the output response of the plant to a square wave as a reference signal for different values of λ. It is clear that the choice of λ affects the nature of the response to the test signal. Fig.9.7 shows the theoretical relationship between the response of the system and the value of λ. When λ gets smaller the output follows the reference signal faster. This is achieved only at the expense of more control effort as less weighting has been put on minimising u(T) in the performance criterion.

(c) Pole/Zero Cancellation Controller

Figs.9.16 to 9.19 show the output response of the plant when the controller uses the Pole/Zero cancellation algorithm; these are recorded for different transient performance objectives. It is shown that the dynamics of the system satisfy the system performance requirements, defined in terms of desired rise time, T_r, overshoot, M_p, and settling time, T_s. The system response is always robust and stable.

(d) PID Controller

Figs.9.20 to 9.22 show the response of the plant when used with a PID controller. These illustrate that, in all circumstances, the closed loop transient performance specified by the operator has been attained.

9.2.4 Discussion

The practical features of these criteria observed from the test results are:

(a) Stability

An obvious feature of the Pole/Zero cancellation and PID criteria is that stability can be guaranteed for arbitrary linear systems (provided that B(z) and A(z) have no common unstable roots). This is in contrast to the One-Step-Ahead design method which requires stability of the zeros, A(z), of the plant. Another problem with the One-Step-Ahead controller is that it is a pure gain controller which is very sensitive to sample time. The reason is that the required controller gain is proportional to the sample time, as shown in Eq.(4.11) (repeated below)

$$u(T) = K [w(T) - y(T)]$$

where

$$K = \frac{\gamma}{kh}$$

Hence, as the sample time (h) gets smaller the gain increases and the system becomes unstable. In practice it will be driven into a non-linear region of operation, producing limit-cycle effects.

(b) Transient response

The transient response is often described in terms of the damping ratio, ζ, and natural frequency ω. However it depends on the locations assigned to the closed loop poles. Given a specification for the desired transient performance, Pole/Zero cancellation and PID control techniques translate this into a feasible set of closed loop pole locations. In practice, we notice that there is constraint on the allowable size of the control signal. Such constraint is taken into account in computing the coefficients of the control algorithm. The selection of these values is explained in Appendix-C.

9.2.5 Conclusion

In servo problems the dynamics of the system have to be specified, such as stability, rise time, overshoot and settling time. Pole/Zero cancellation and the PID criteria allow the operator to specify these requirements explicitly and easily. The execution time for each criterion is short compared to the sample time as shown in Appendix-C. From practical results obtained it is clear that simulation does not reflect the true behaviour of real systems, but it does provide a starting point for the design and implementation of control techniques.

9.3 TEST RESULTS - SELF-TUNING CONTROLLER

9.3.1 Overview

In this section actual test results are given for the plant control system. The performance of the system is illustrated for both fixed and variable loads. The following self-tuning control techniques are used:

* Pole/Zero Cancellation Self-Tuning
* PID Self-Tuning

These tests are carried out on the target controller which uses an Intel 8088 microprocessor.

9.3.2 Fixed load

The main feature of the self-tuning controller is its ability to identify unknown system parameters which are then used by the control algorithm to satisfy a specified performance objective. This feature is illustrated in Figs.9.23 to 9.31 where different transient performance objectives are selected for each test. It is clear how self-tuning controller fulfills these requirements. Fig.9.23 for example shows the output response to a square wave input signal using Pole/Zero cancellation self-tuning. It can be seen that the initial response of the system does not meet the preset specifications; performance errors are caused by using an incorrect plant model, i.e. the estimated system parameters have not yet reached their true values. In subsequent pulses the performance objective is fulfilled since, by this time, the estimated parameters correspond to the true values.

Fig.9.24 shows how the estimated values of the system parameters converge to their actual values as time goes on. Fig.9.25 shows the response of the system when the estimated parameters converge to their actual values. Fig.9.26 shows the rise time and the overshoot of the system as specified by the operator. Figs.9.27 to 9.29 show the response of the system under PID self-tuning.

9.3.3 Variable load

If the load on the shaft changes the system parameters will change. Therefore, the new values of the system parameters should be used in the controller to meet the same performance objective. In the fixed controller case these changes are not sensed by the algorithm; as a result the parameter values used in the control algorithm are not correct. Hence the system response will fail to meet the predefined performance objective. This point is confirmed in Figs.9.32 to 9.35.

Figs.9.32 and 9.33 illustrate the response of the system under the fixed digital controller when the system loading is increased and decreased respectively. Figs.9.34 and 9.35 show how the rise time decreases and the overshoot increases when the load is reduced.

Figs.9.36 and 9.37 display the response of the system when using a self-tuning controller. This controller continuously identifies the values of the plant parameters, tuning the controller to meet the specified performance objective as necessary. Fig.9.38 shows the estimator performance as system loading is changed.

9.3.4 Discussion

The test results show that the transient response of the system to the first pulse (step-input) (in terms of overshoot and rise time) fails to meet its

specification. This is caused by the use of an incorrect model. It takes 30-60 samples before the estimated values of the transfer function coefficients converge to their true values. During this time the estimated values of the pole and zero of the system influence the transient performance as follows:

(a) Fig.9.39 illustrates the effect of the estimated value of a_1. If the value of a_1 is less than the true value then this increases the overshoot M_p.

(b) Fig.9.39 demonstrates the effect of the pole on the response. The rise time decreases if the estimated value of the pole is less than the true value.

(c) Fig.9.40 shows the effects of the sign of the estimated value on system response. If the initial estimate of the sign of a_1 is negative this may cause the step response to start out in the wrong direction. To prevent this a constraint is built into the identification process to prevent negative sign being generated.

Since a constraint is put on the estimated value sign, the poor estimates in the first few samples affect the overshoot and the rise time of the system only.

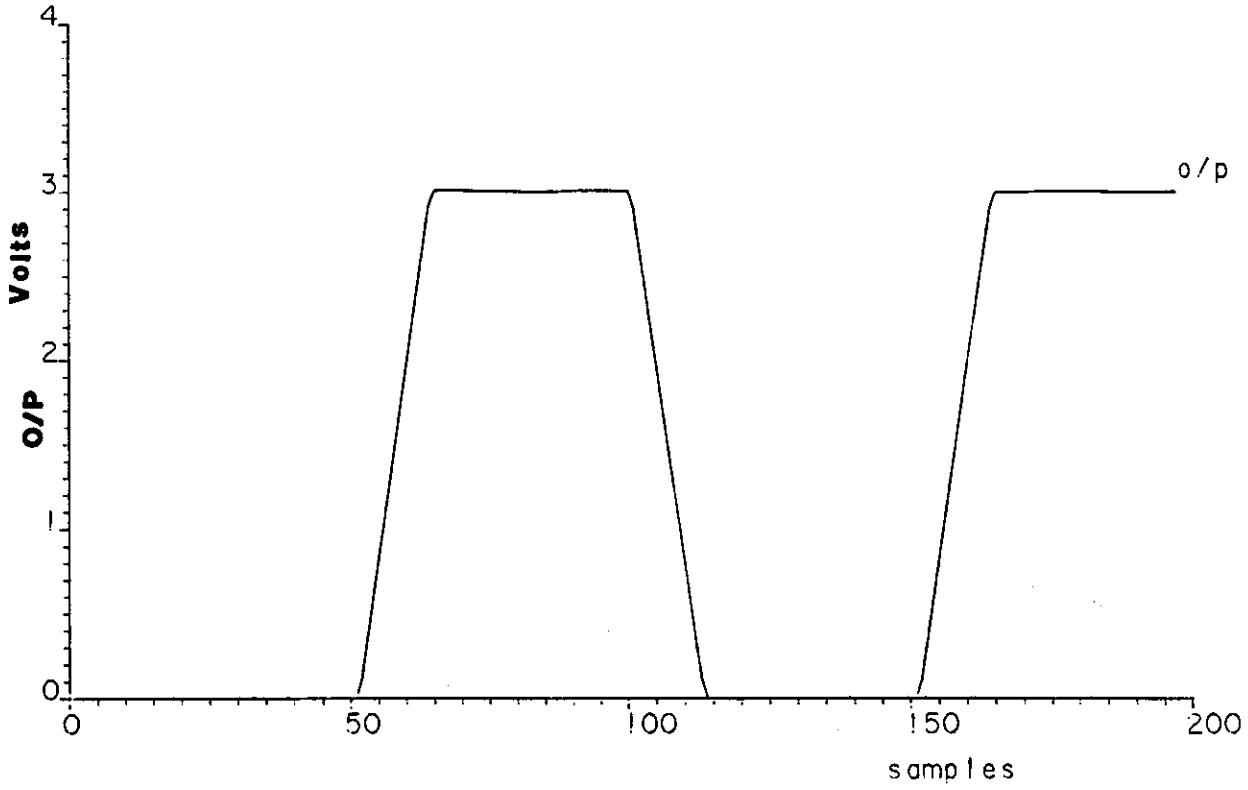


Fig.9.1

Simulation of the plant response to a step input – One Step Ahead (OSA)

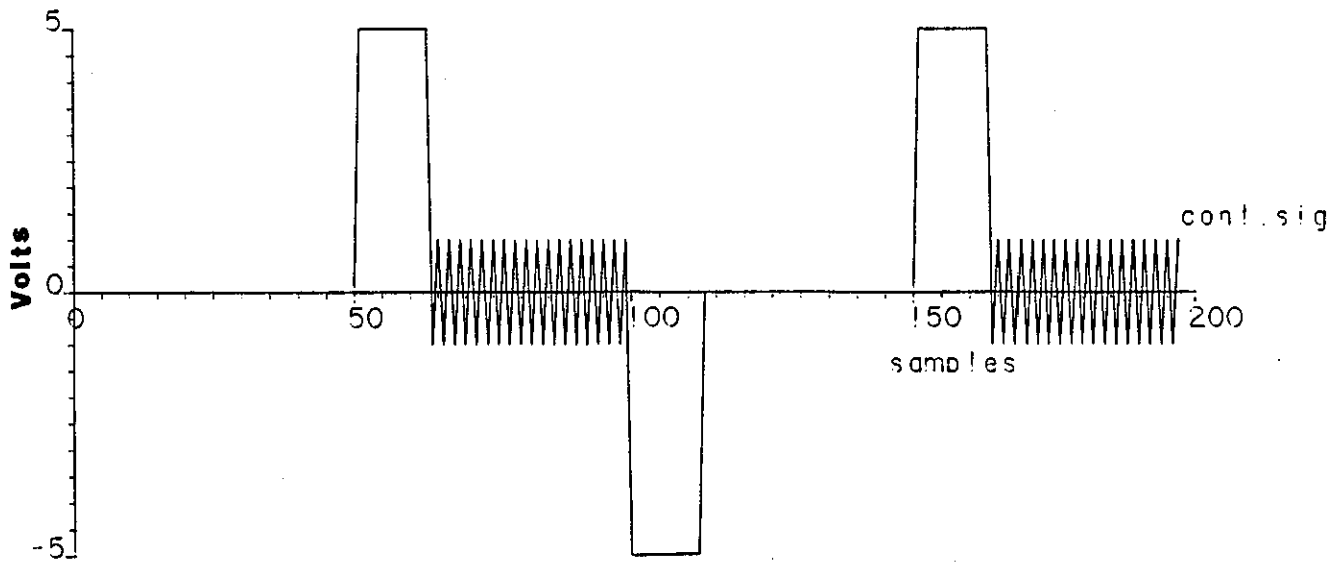


Fig.9.2

The corresponding control signal – One Step Ahead (OSA)

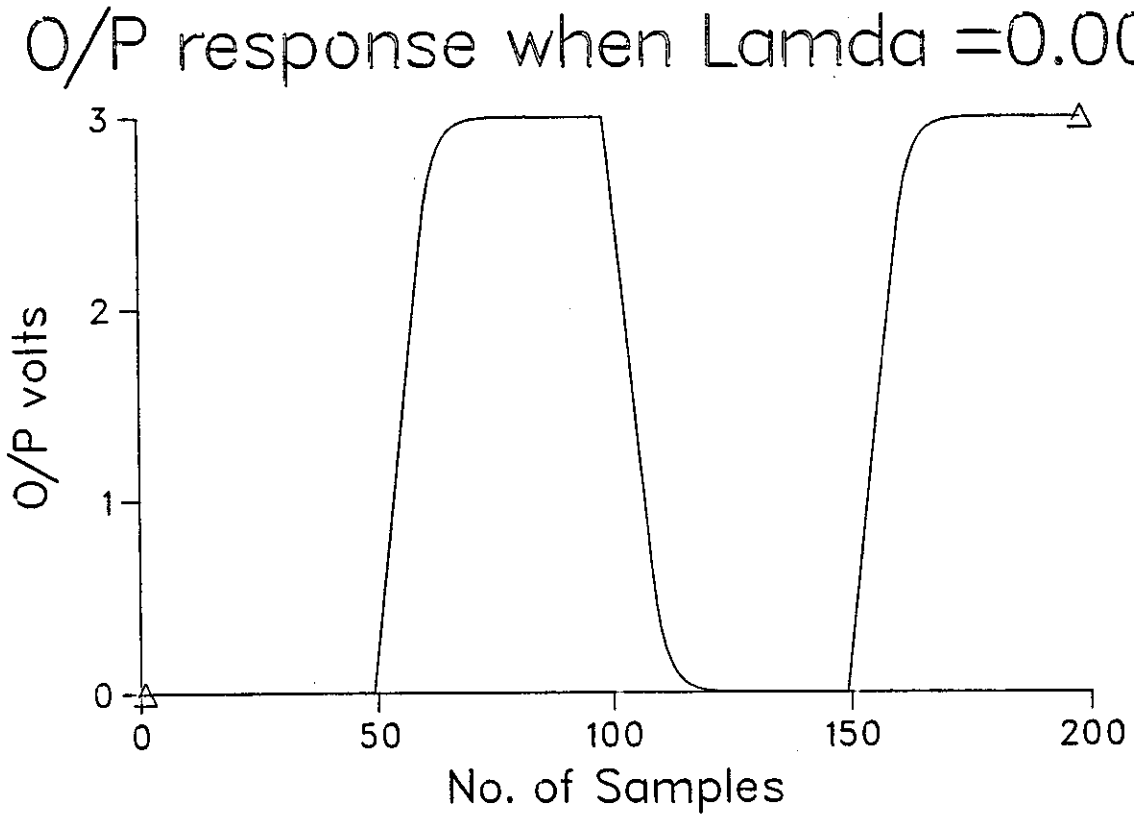


Fig.9.3
Simulation of the plant response to a step input – Weighted One Step Ahead (WOSA)

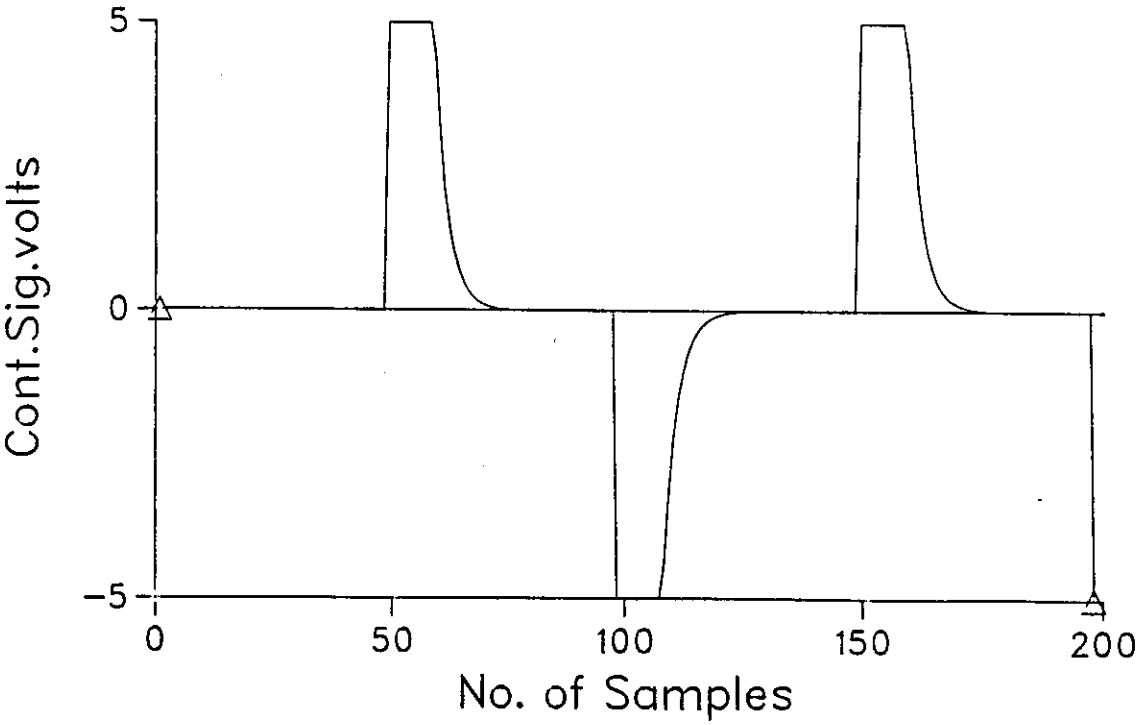


Fig.9.4
The corresponding control signal – WOSA

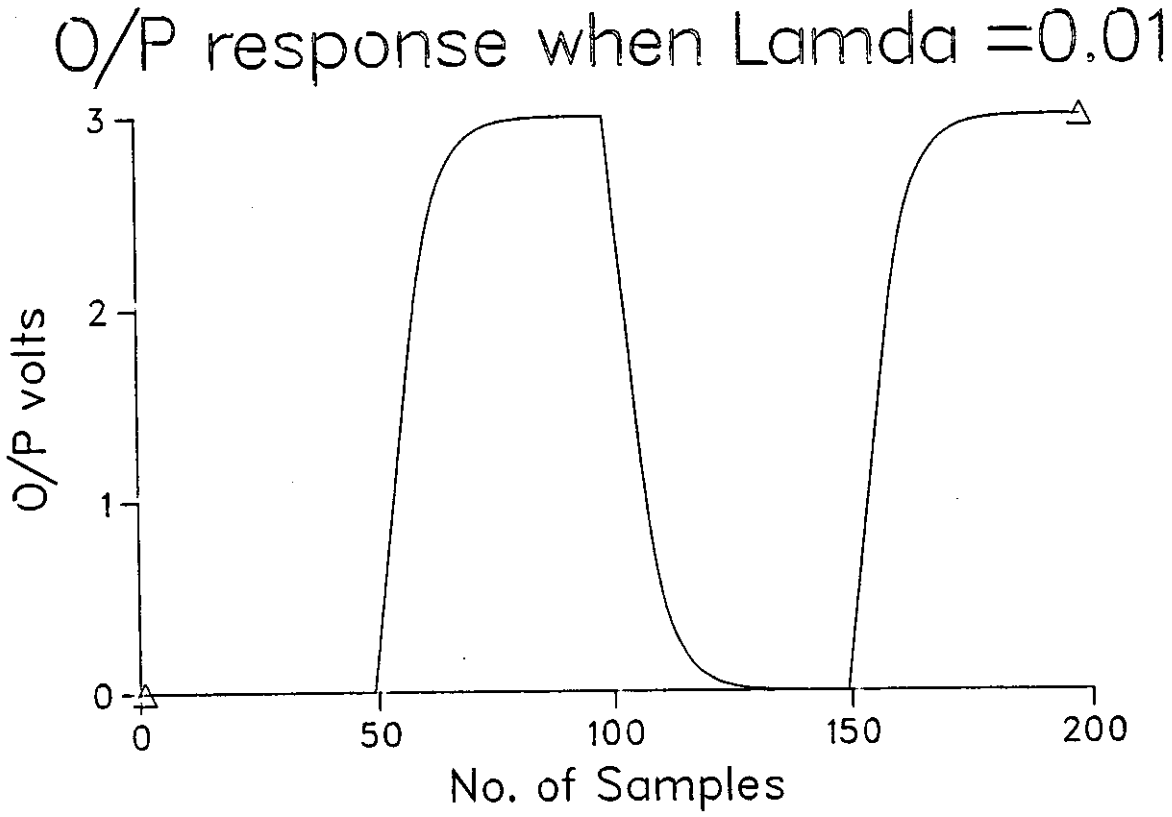


Fig.9.5

Simulation of the plant response to a step input – WOSA

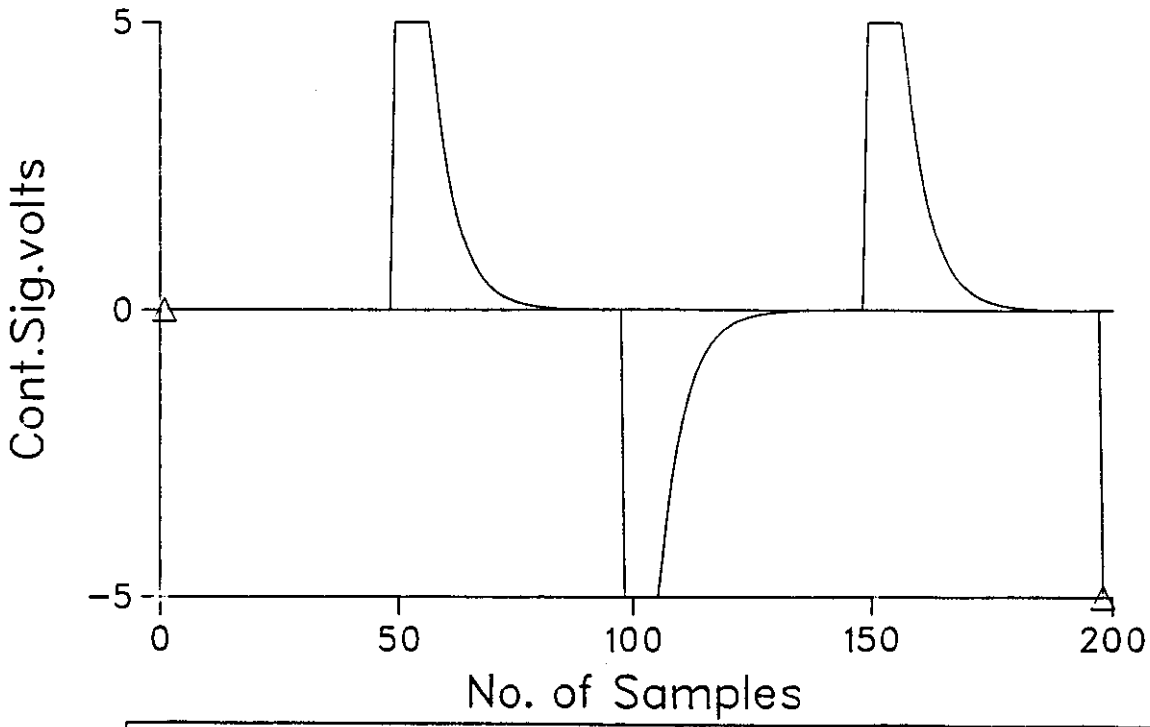


Fig.9.6

The corresponding control signal – WOSA

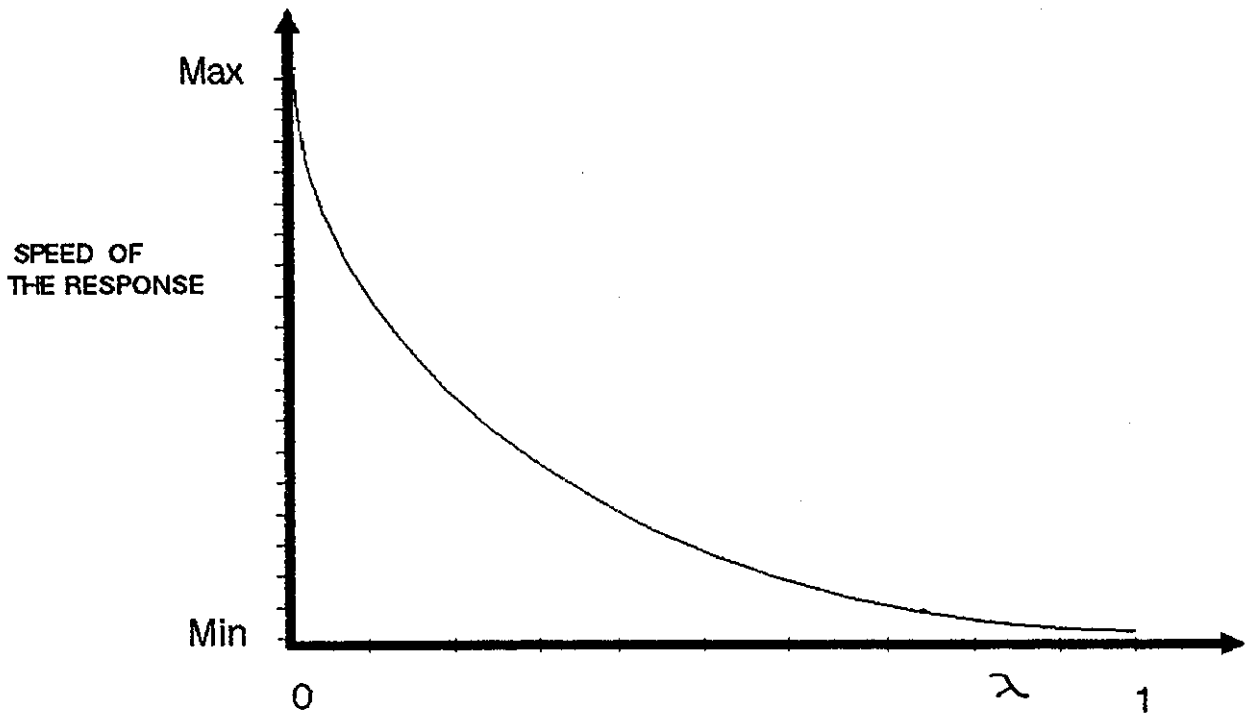


Fig . 9 . 7

System response VS λ

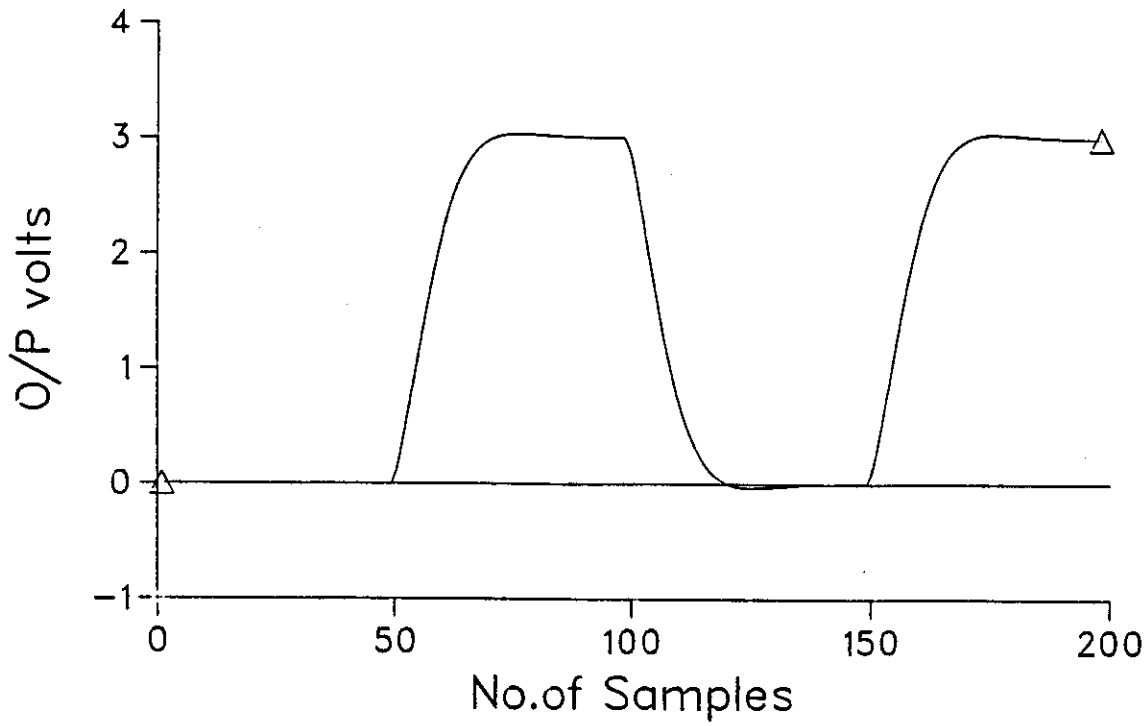


Fig.9.8
Simulation of the plant response to a step input – Pole/Zero Cancellation (P/Z)

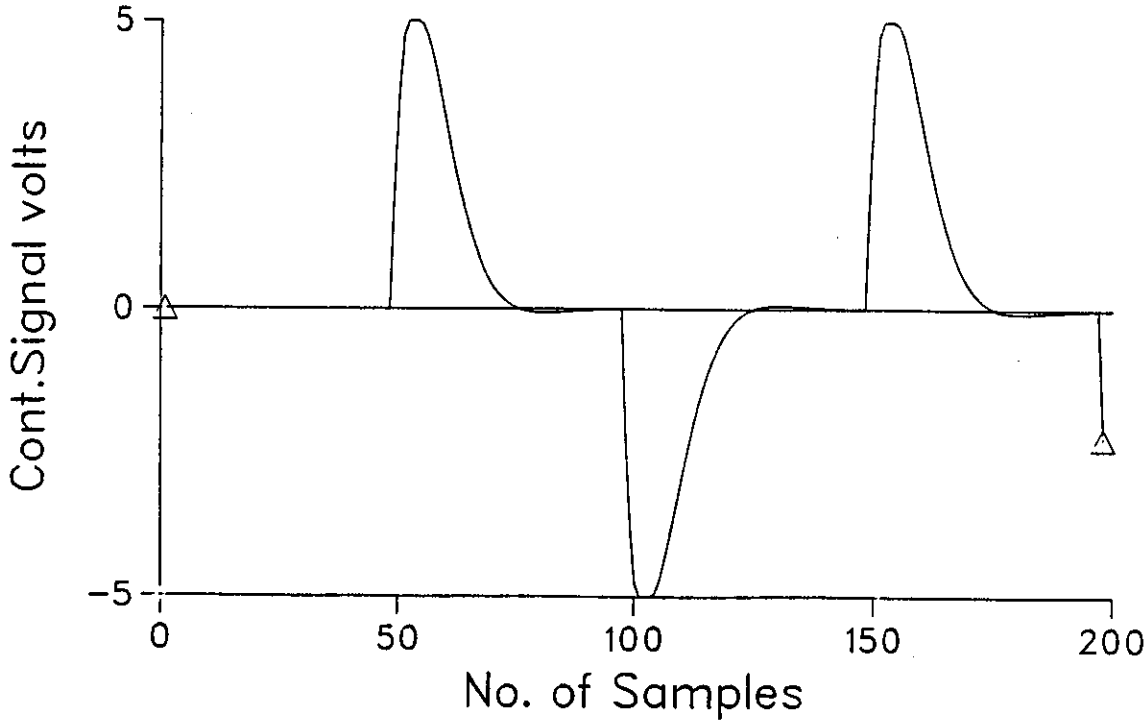


Fig.9.9
The corresponding control signal – P/Z

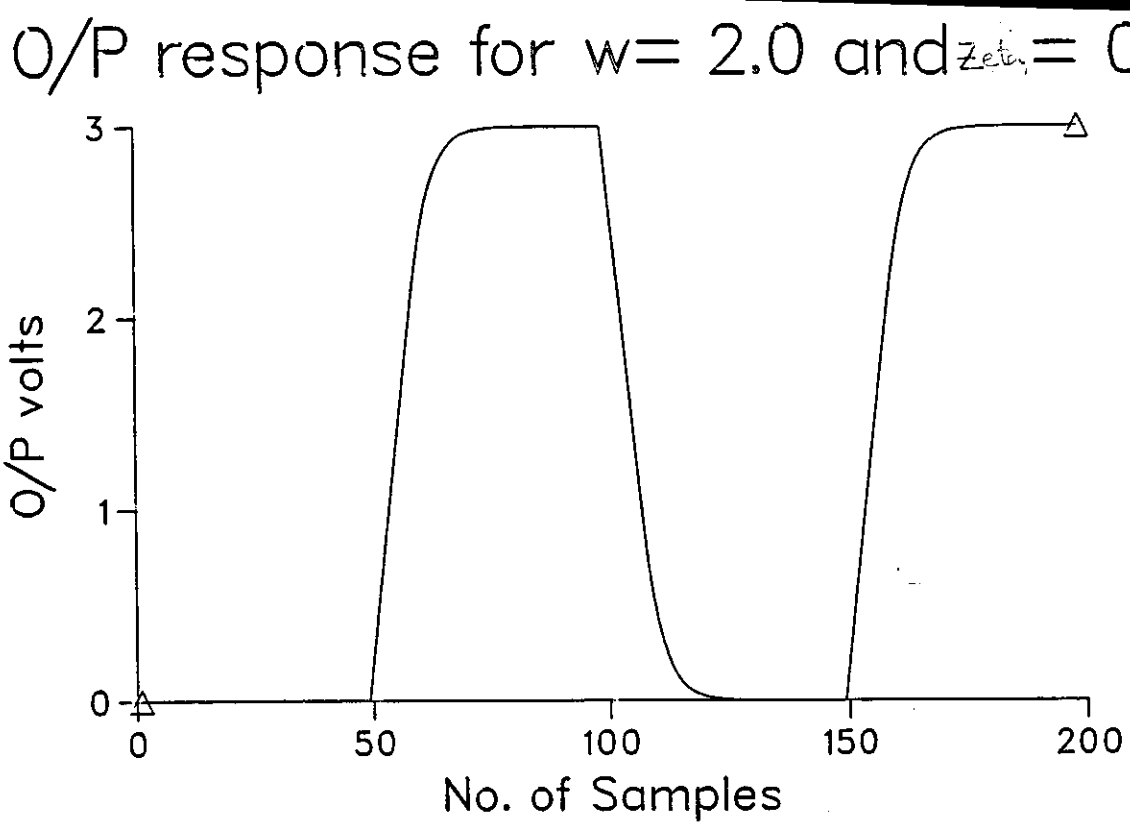


Fig.9.10

Simulation of the plant response to a step input – PID

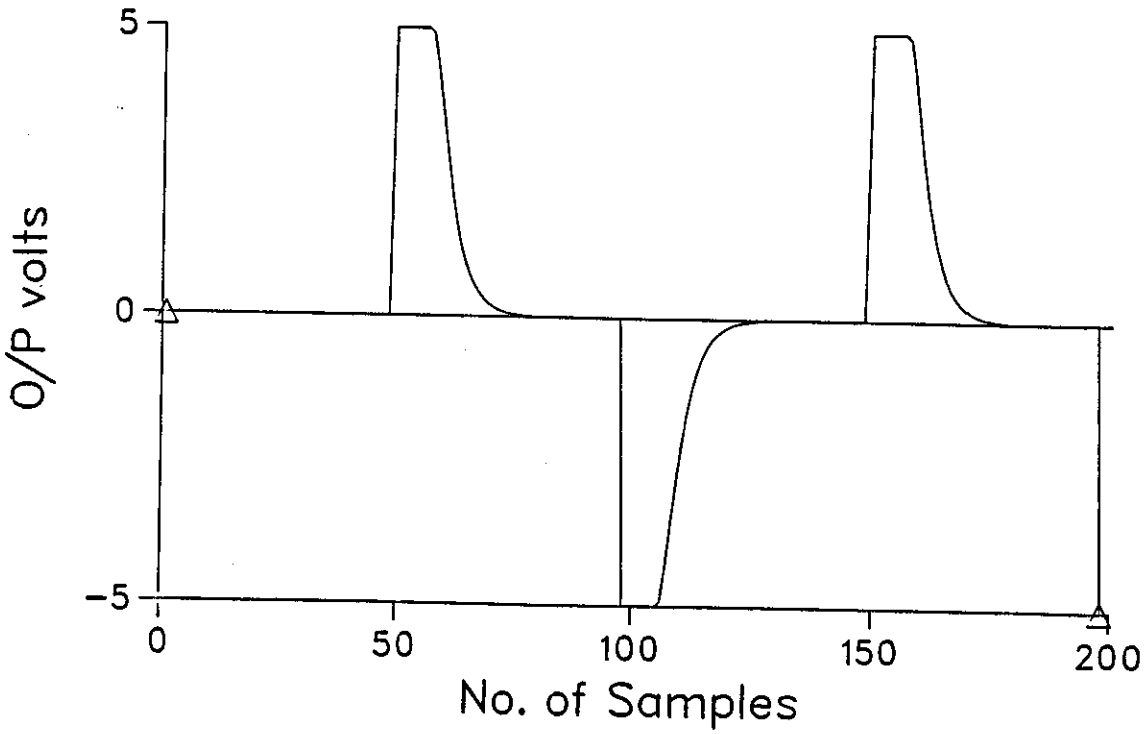


Fig.9.11

The corresponding control signal – PID

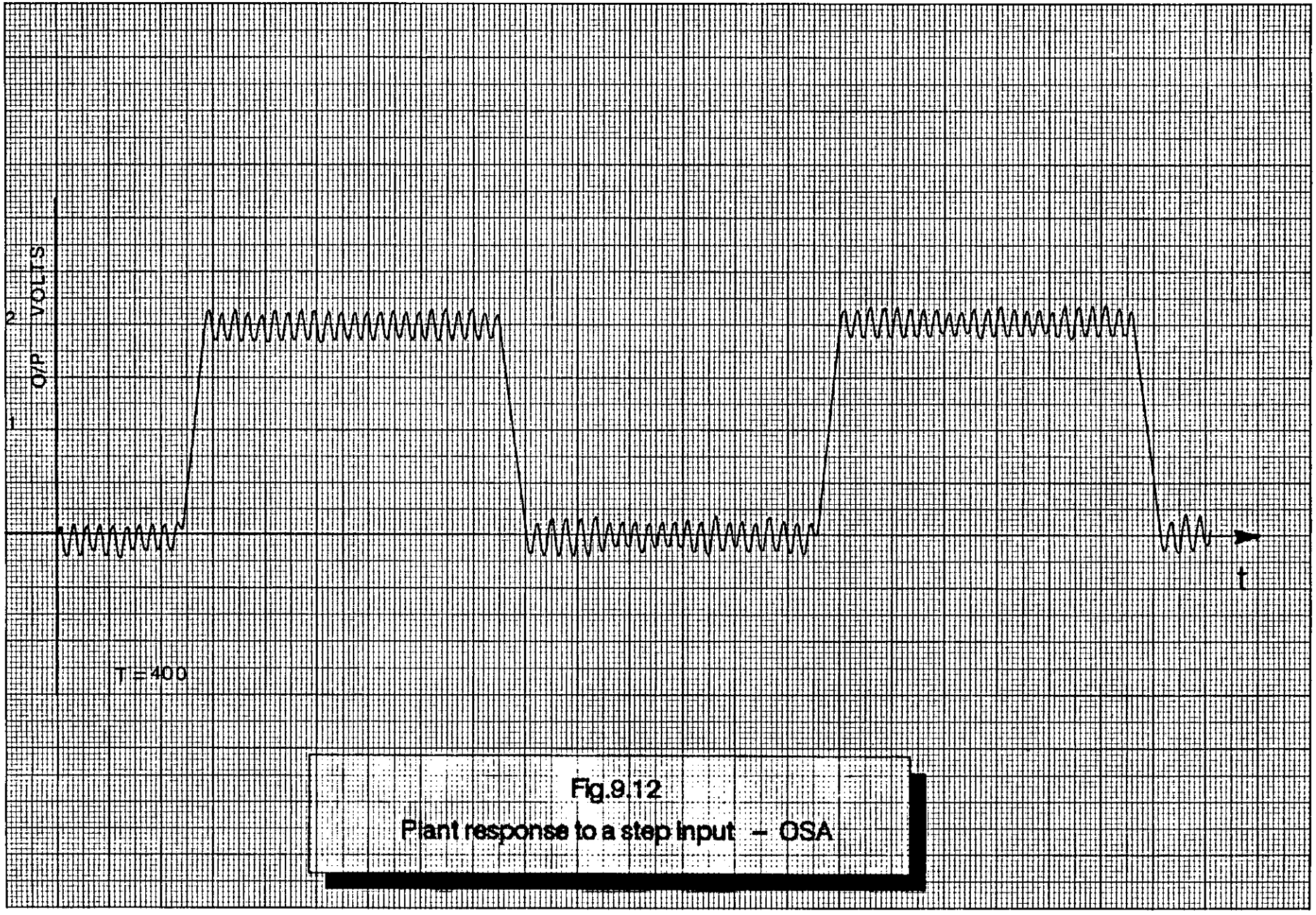


Fig.9.12

Plant response to a step input – OSA

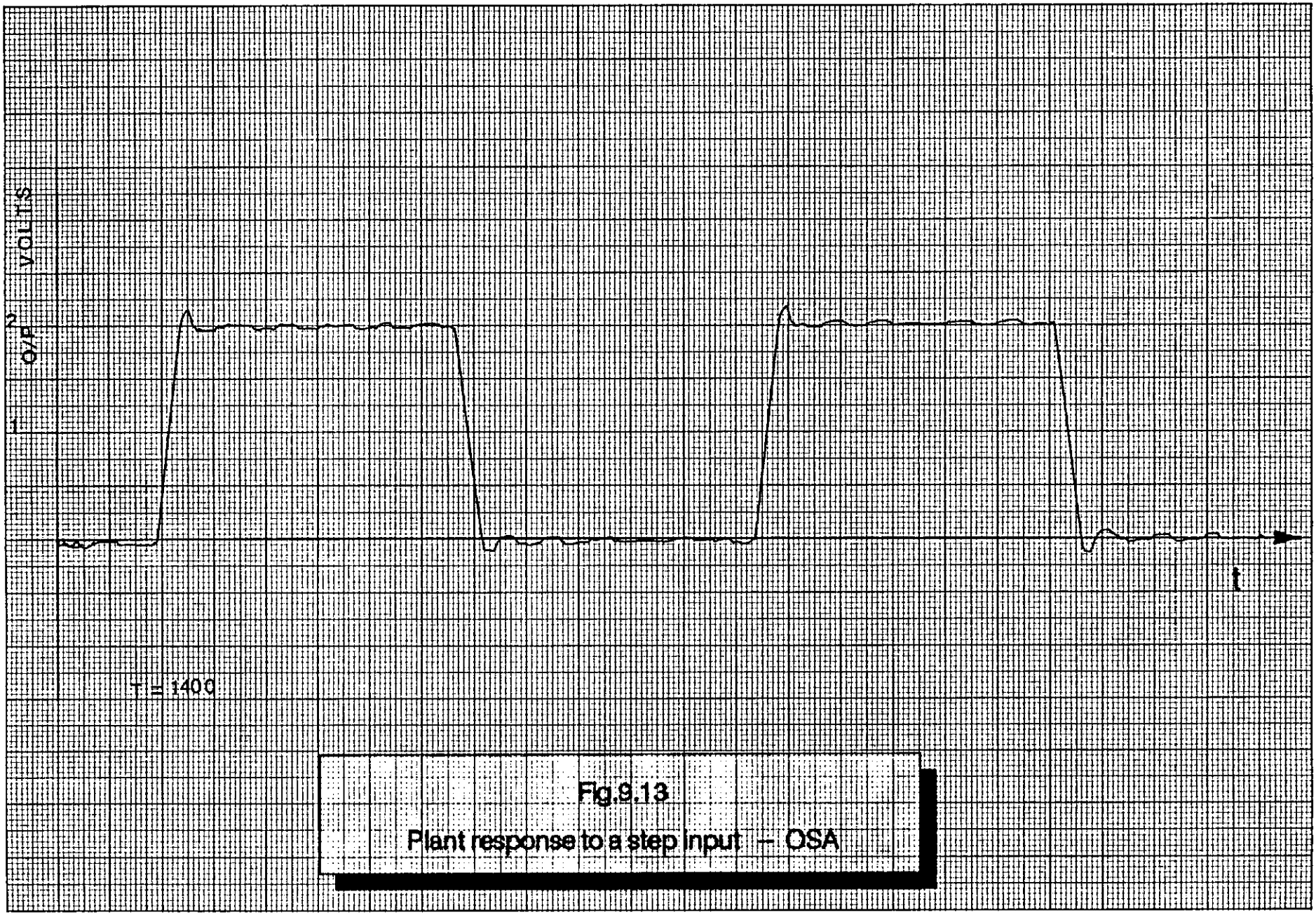


Fig.9.13

Plant response to a step input – OSA

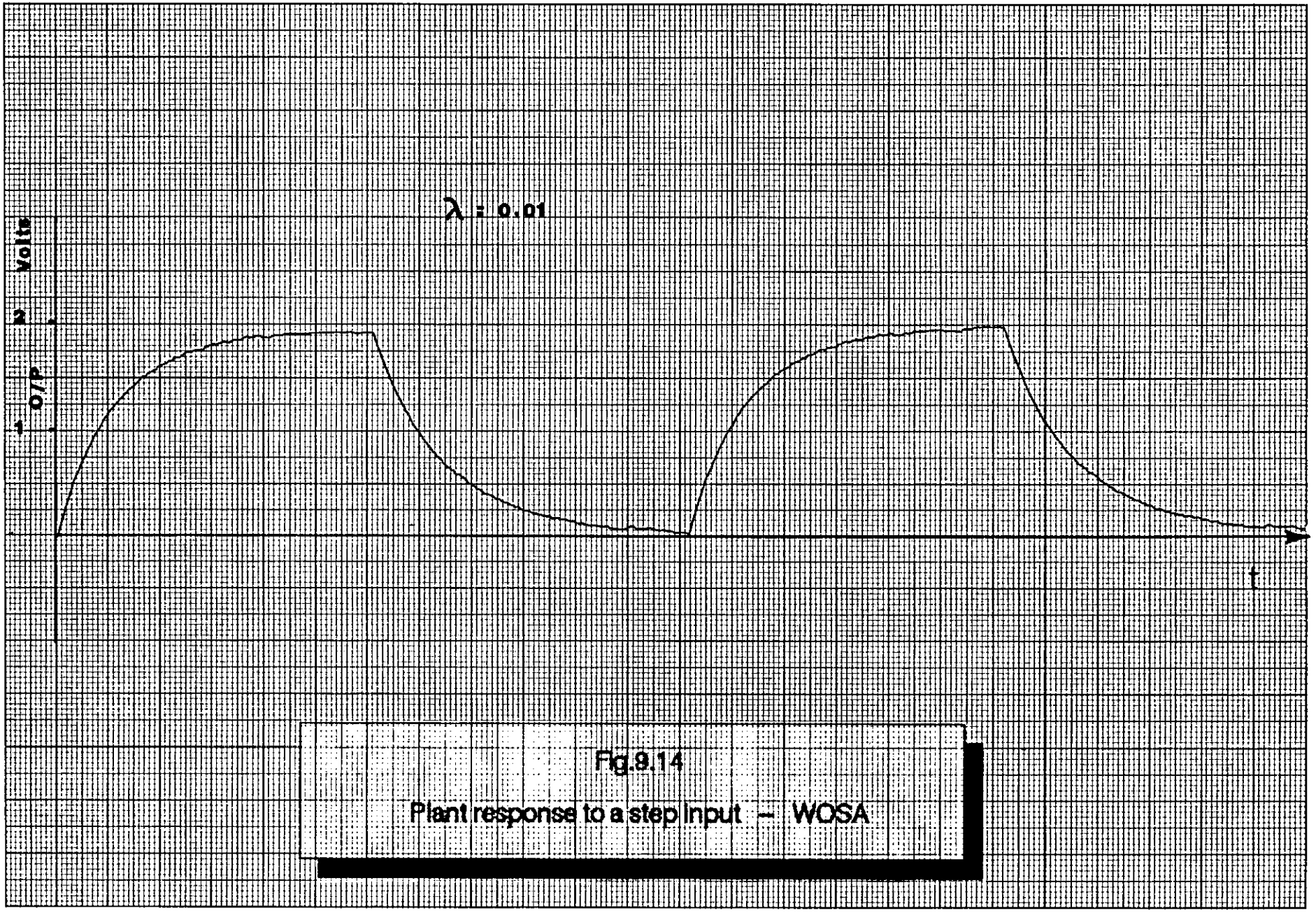


Fig.9.14

Plant response to a step input – WOSA

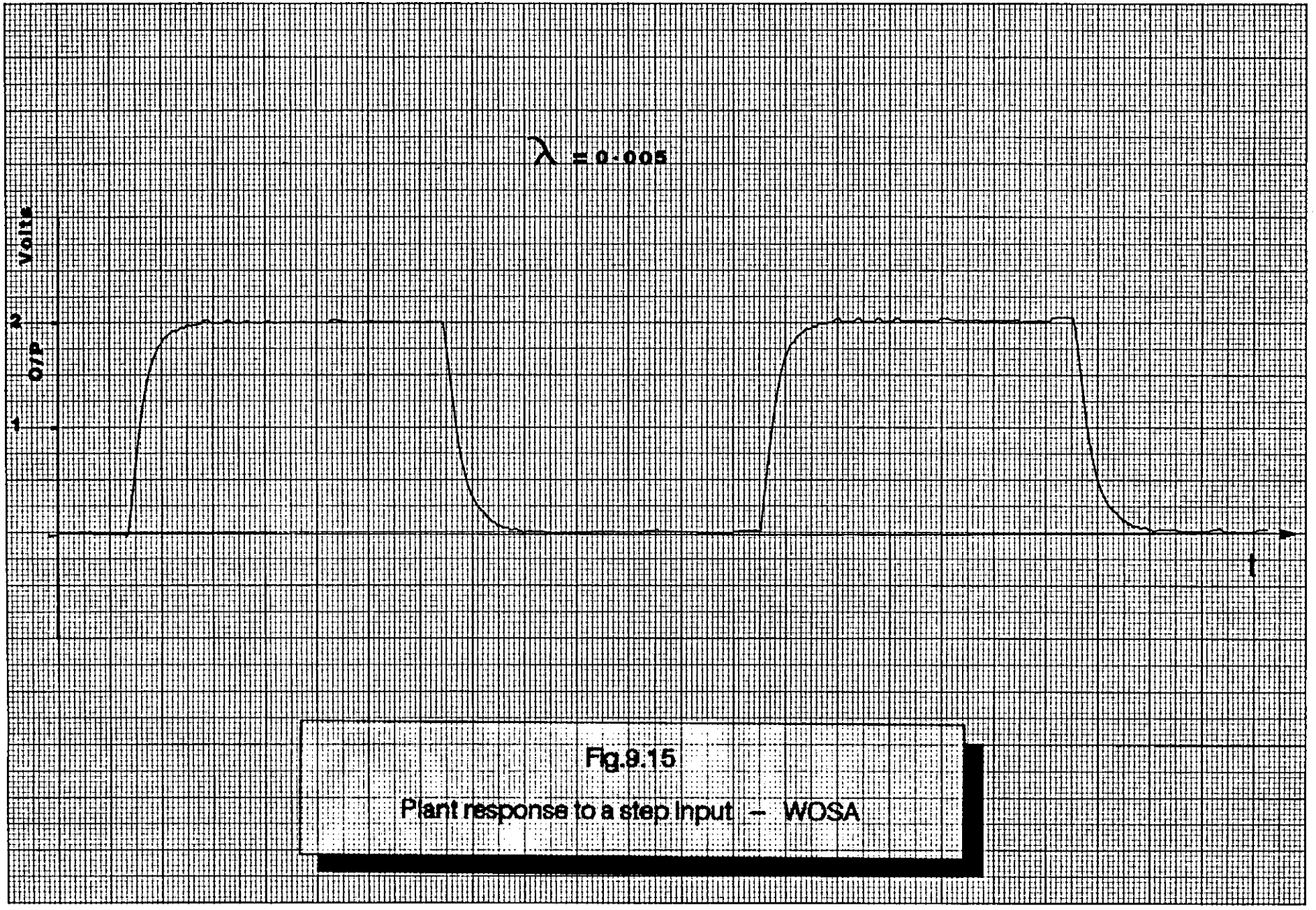


Fig.9.15

Plant response to a step input – WOSA

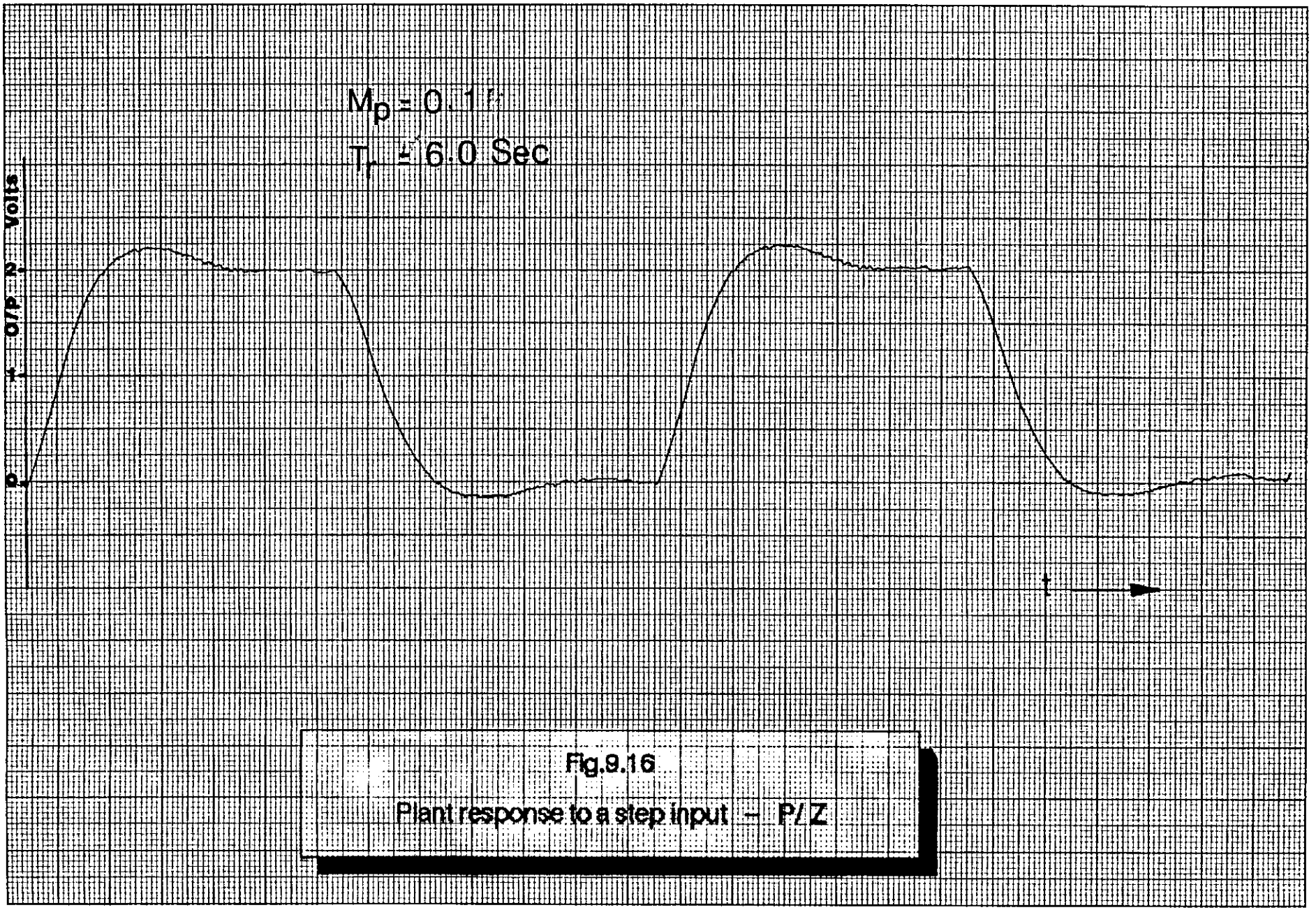


Fig.9.16

Plant response to a step input – P/Z

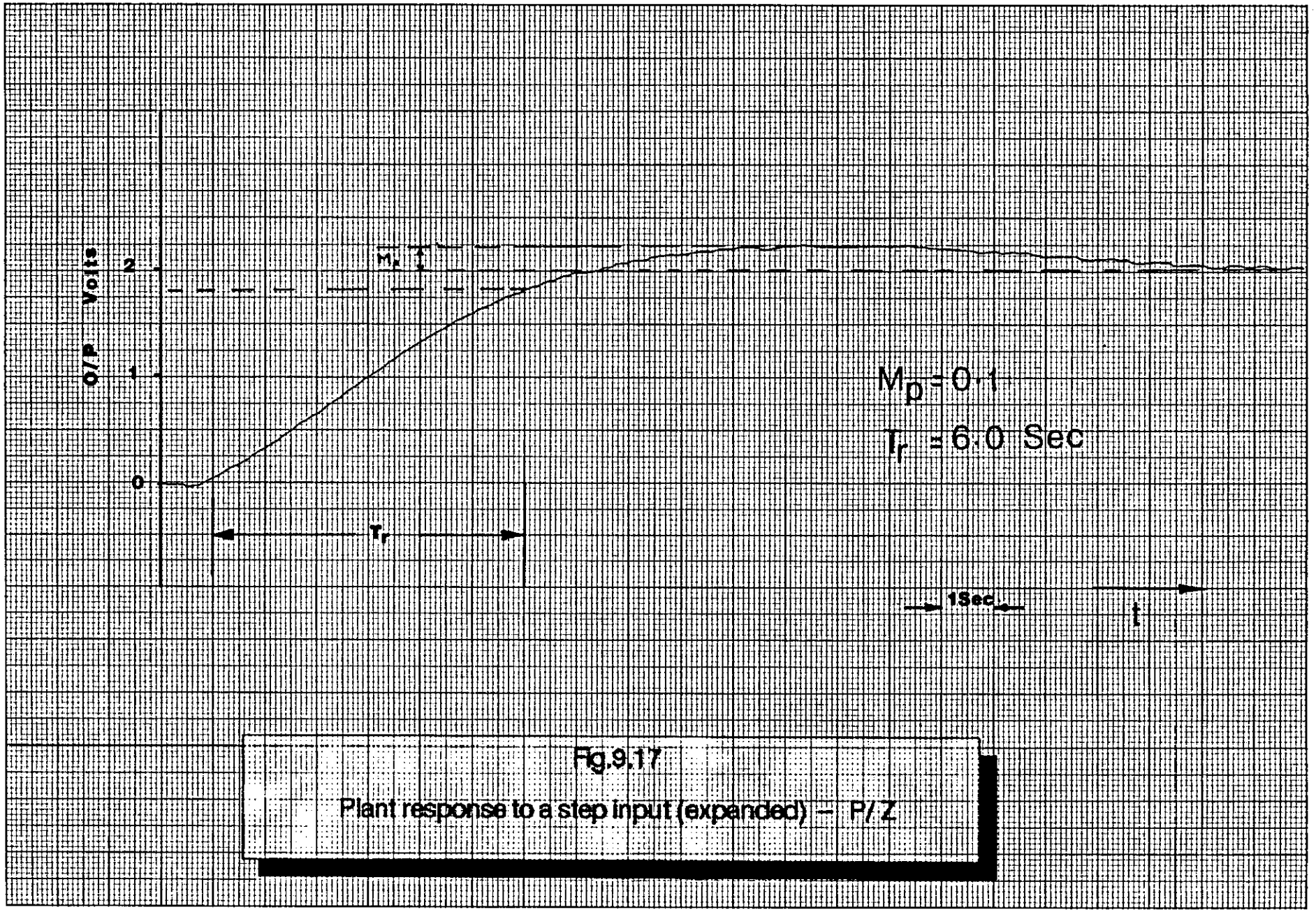


Fig.9.17

Plant response to a step input (expanded) – P/Z

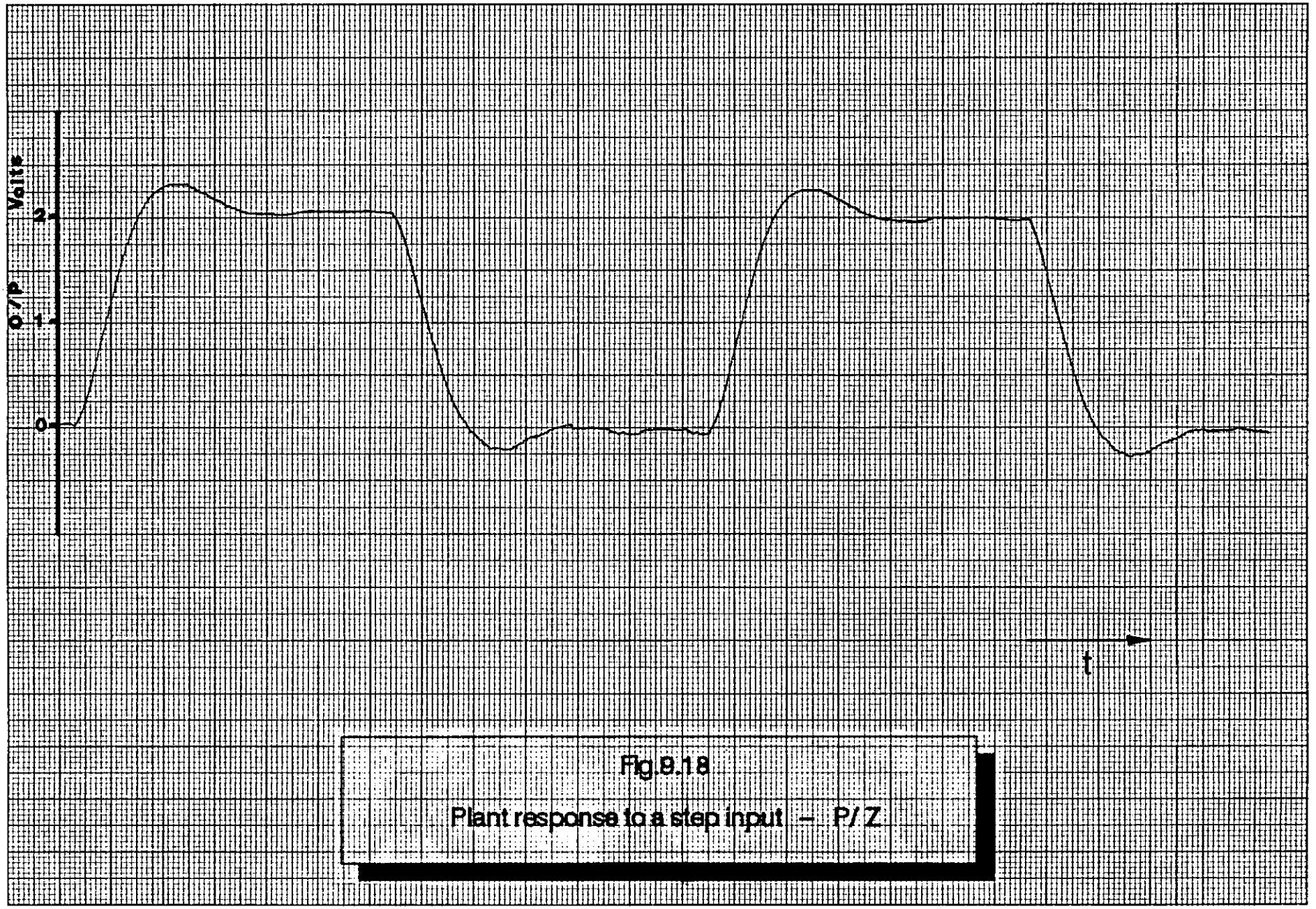


Fig.8.18

Plant response to a step input – P/Z

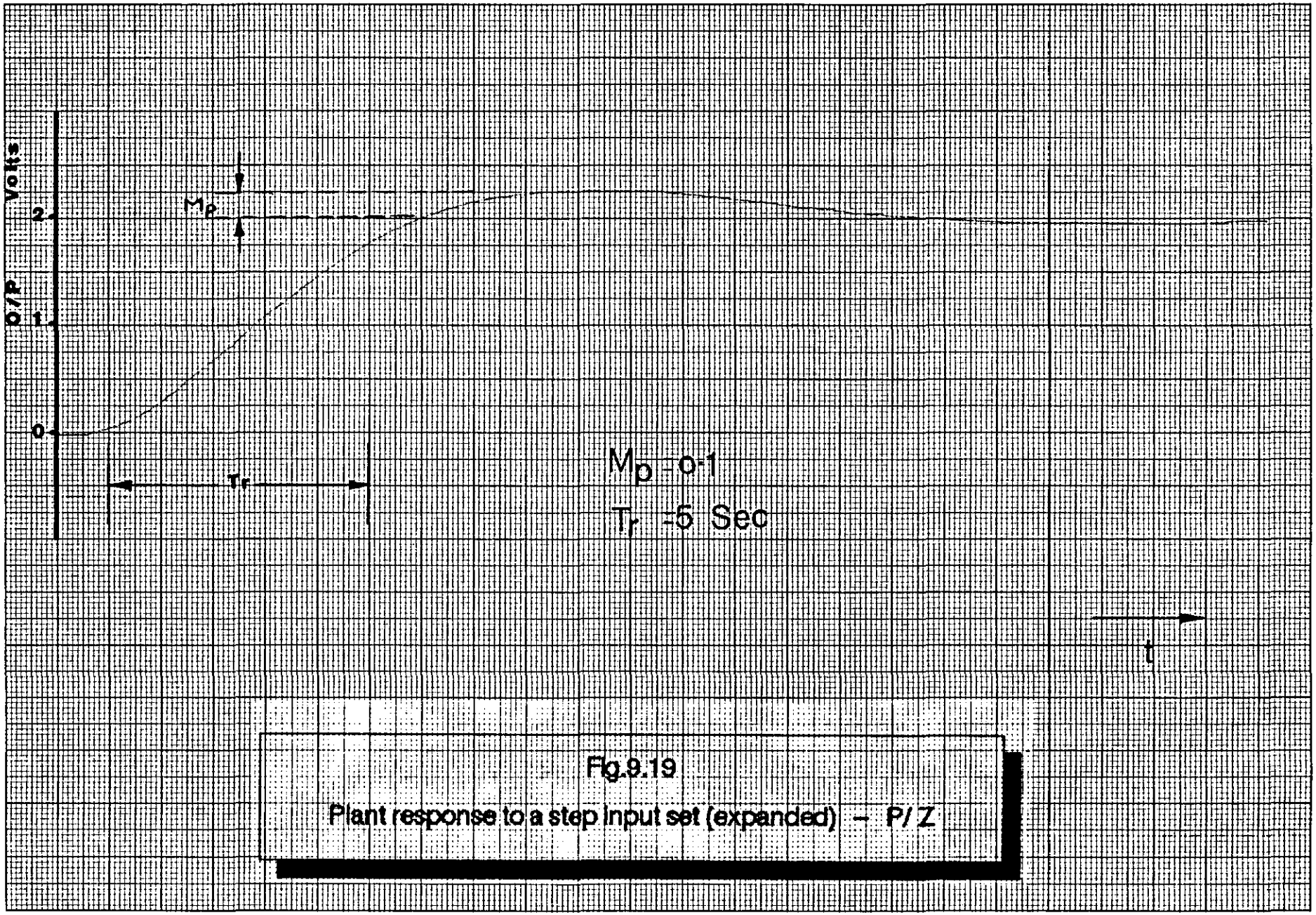


Fig.9.19

Plant response to a step input set (expanded) – P/Z

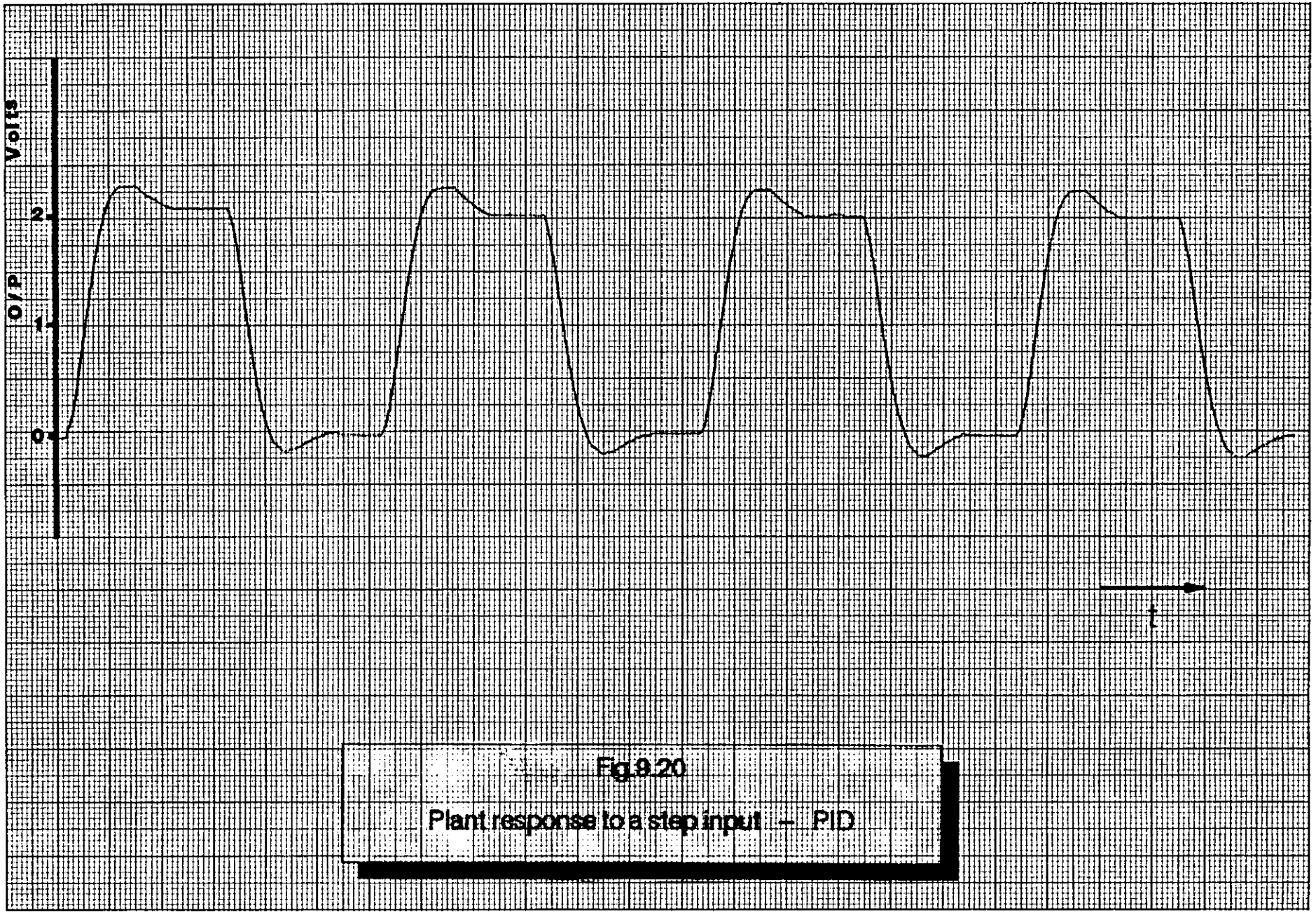


Fig.9.20

Plant response to a step input – PID

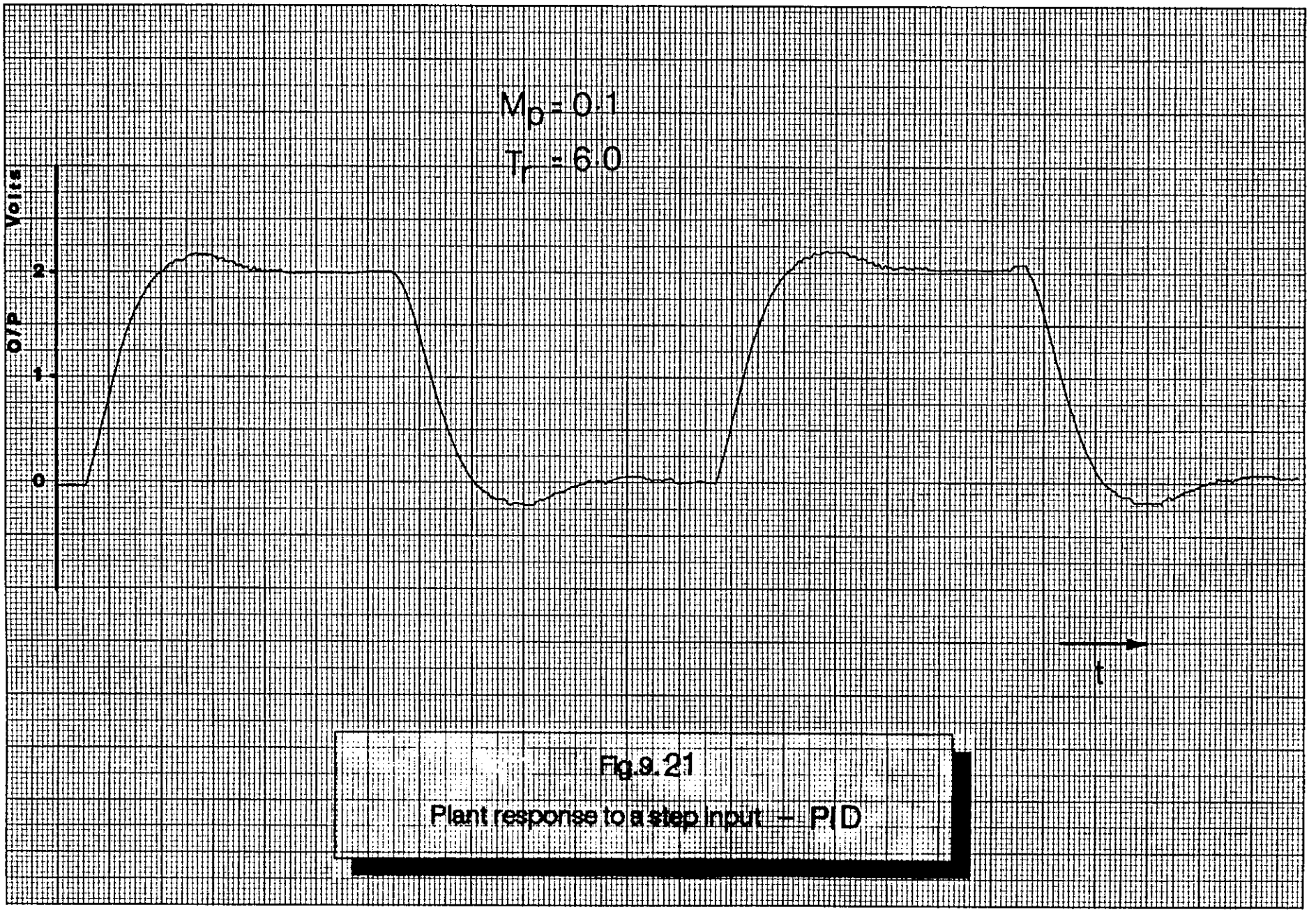


Fig.9.21

Plant response to a step input – PID

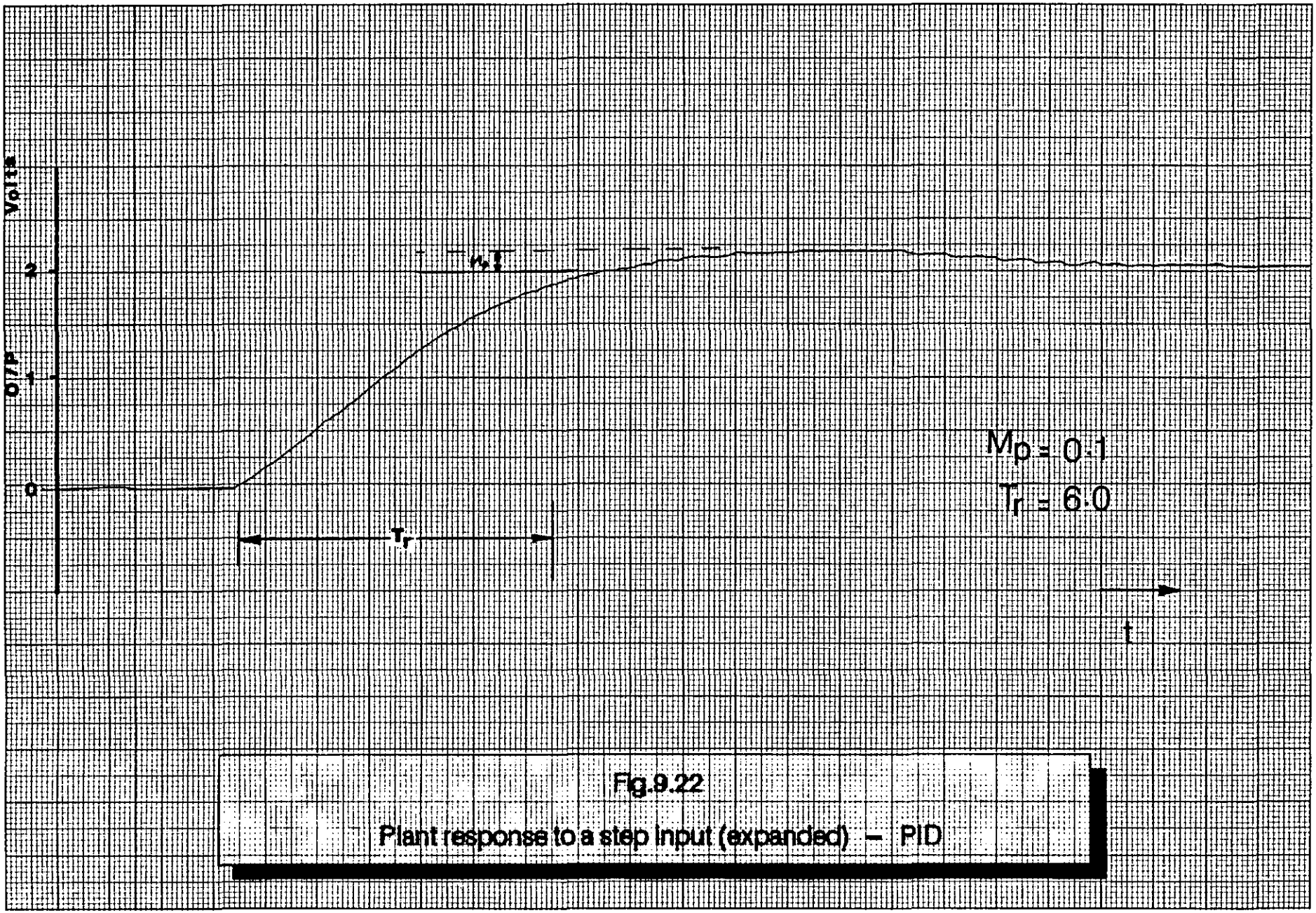


Fig.9.22

Plant response to a step input (expanded) – PID

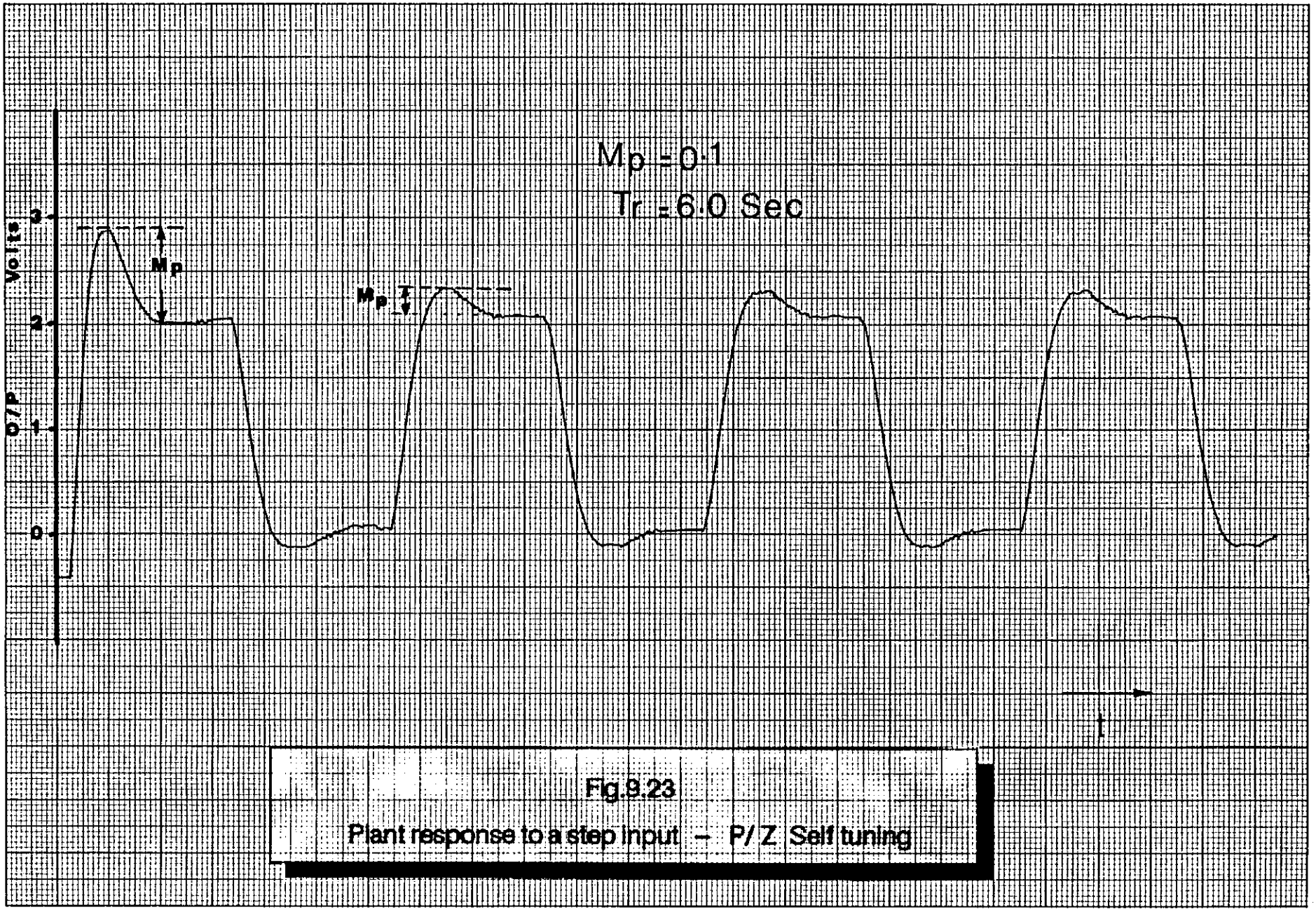
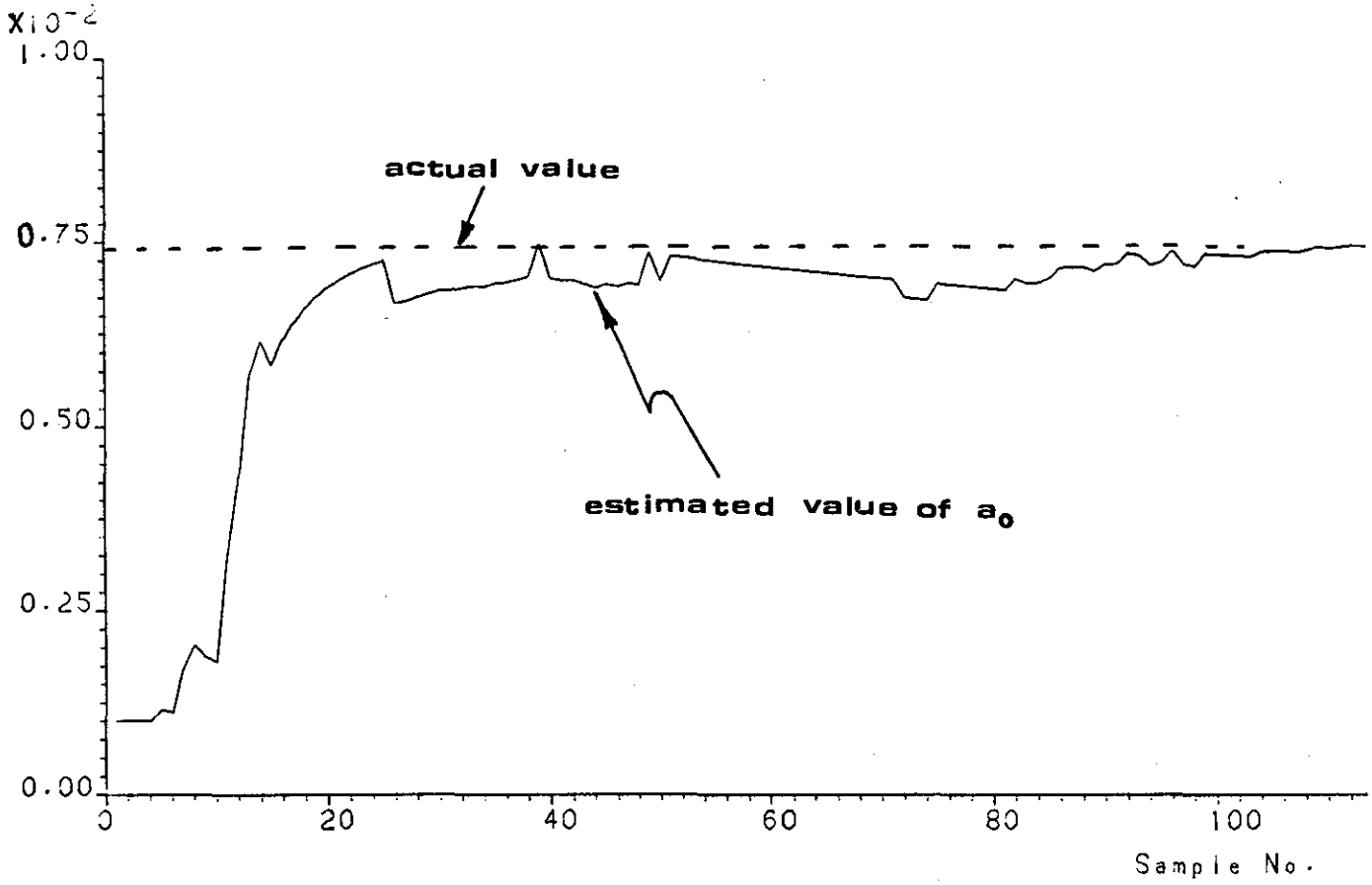


Fig.9.23

Plant response to a step input – P/Z Self tuning



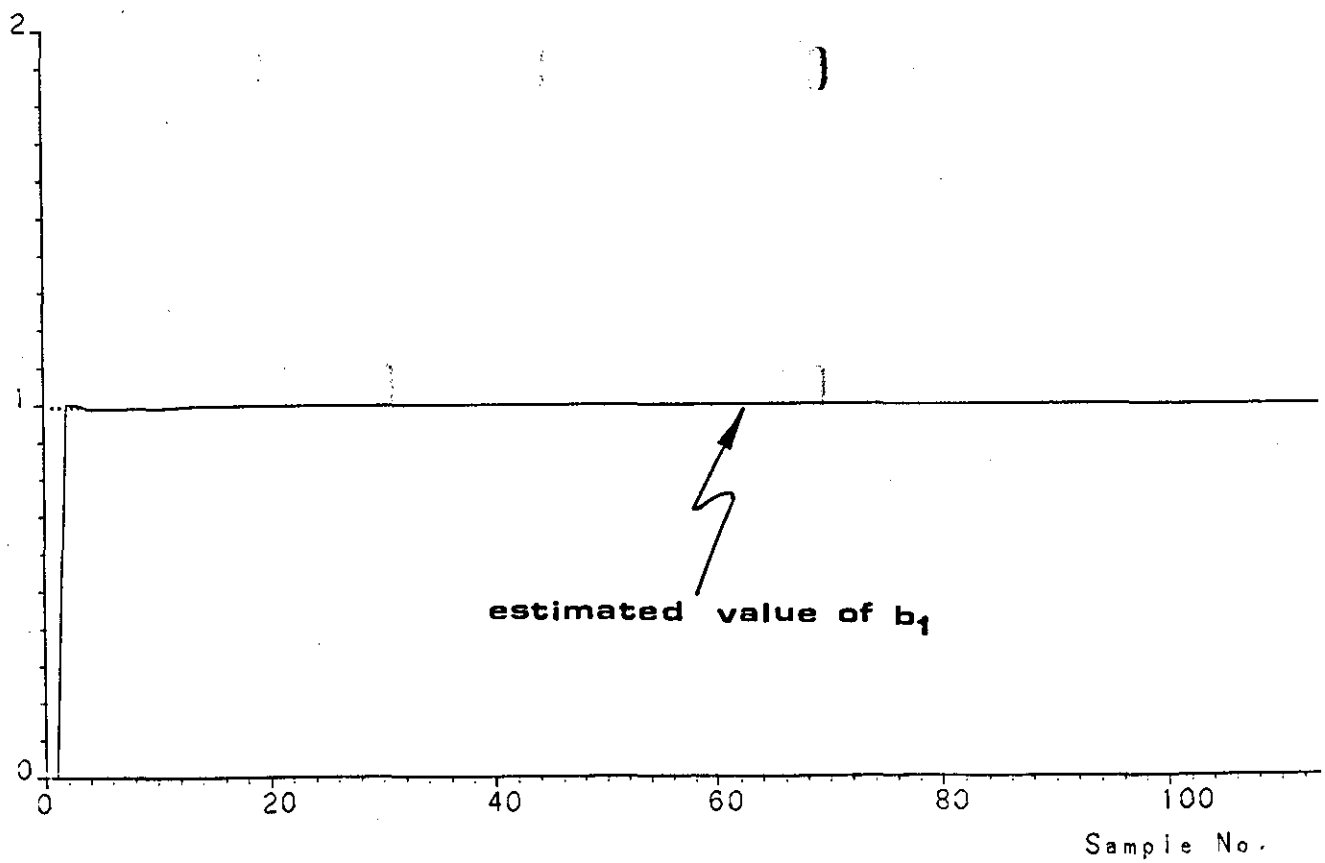


Fig.9.24

Convergance of estimated plant parameters

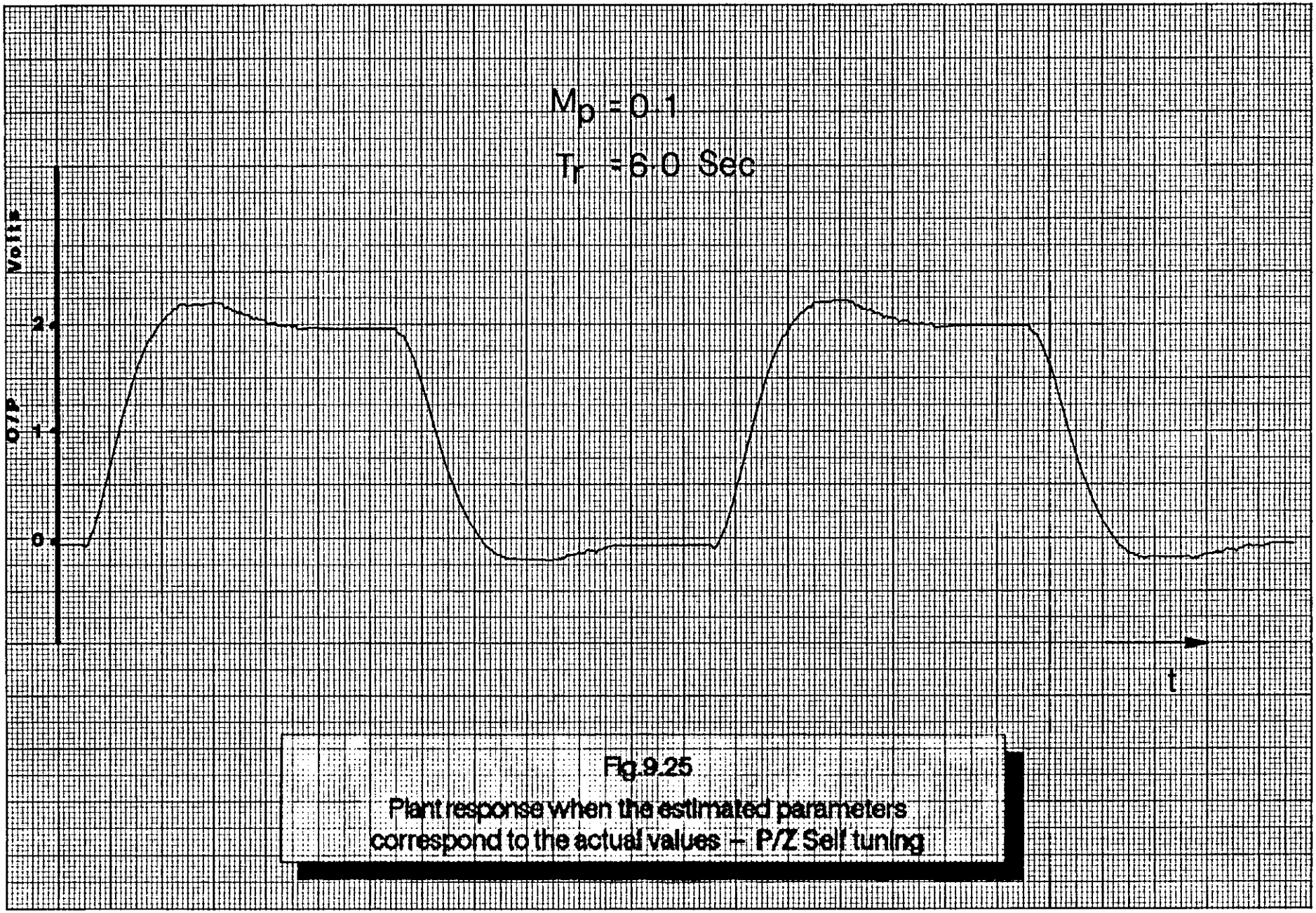


Fig.9.25
Plant response when the estimated parameters correspond to the actual values – P/Z Self tuning

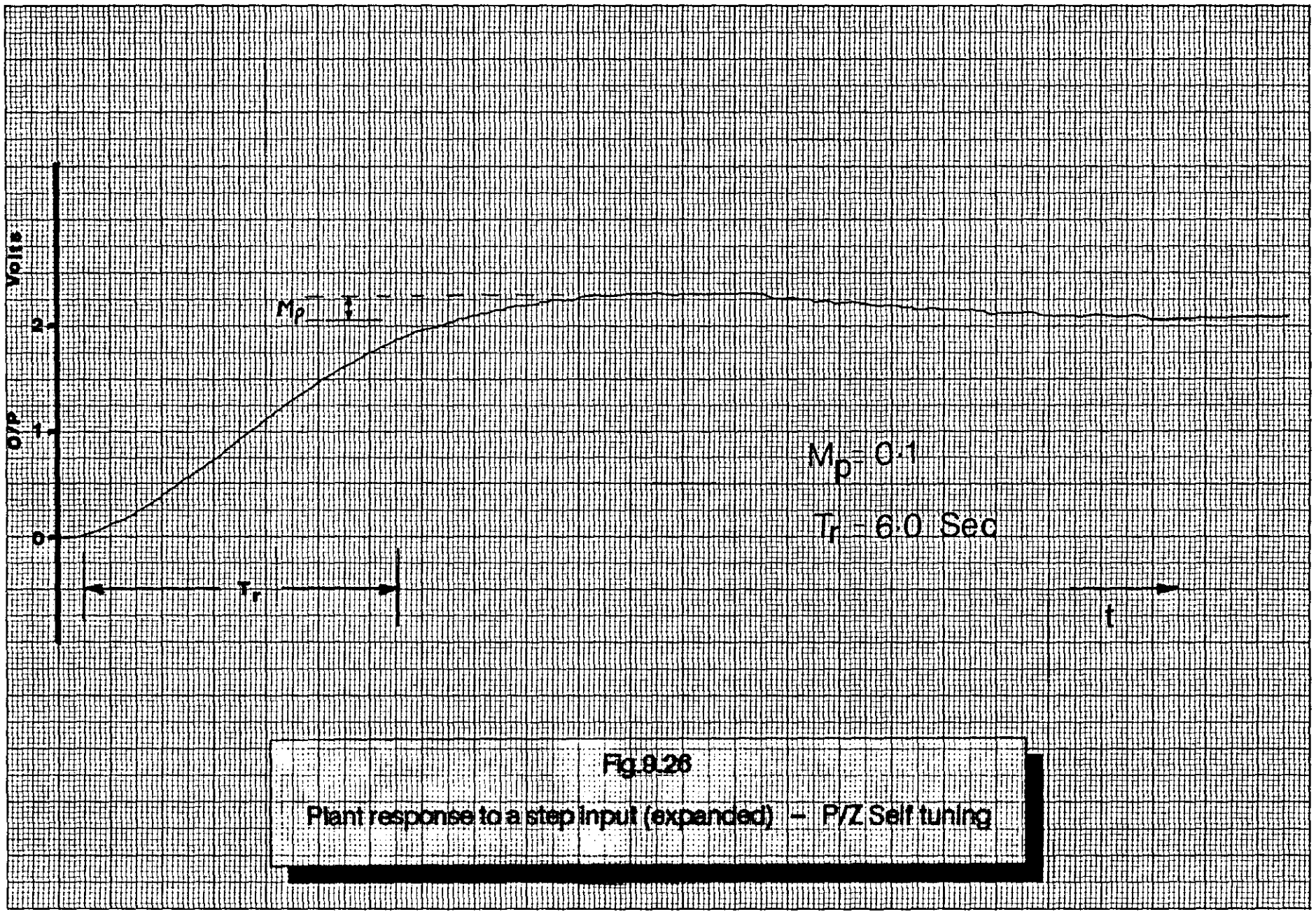


Fig.9.26

Plant response to a step input (expanded) – P/Z Self tuning

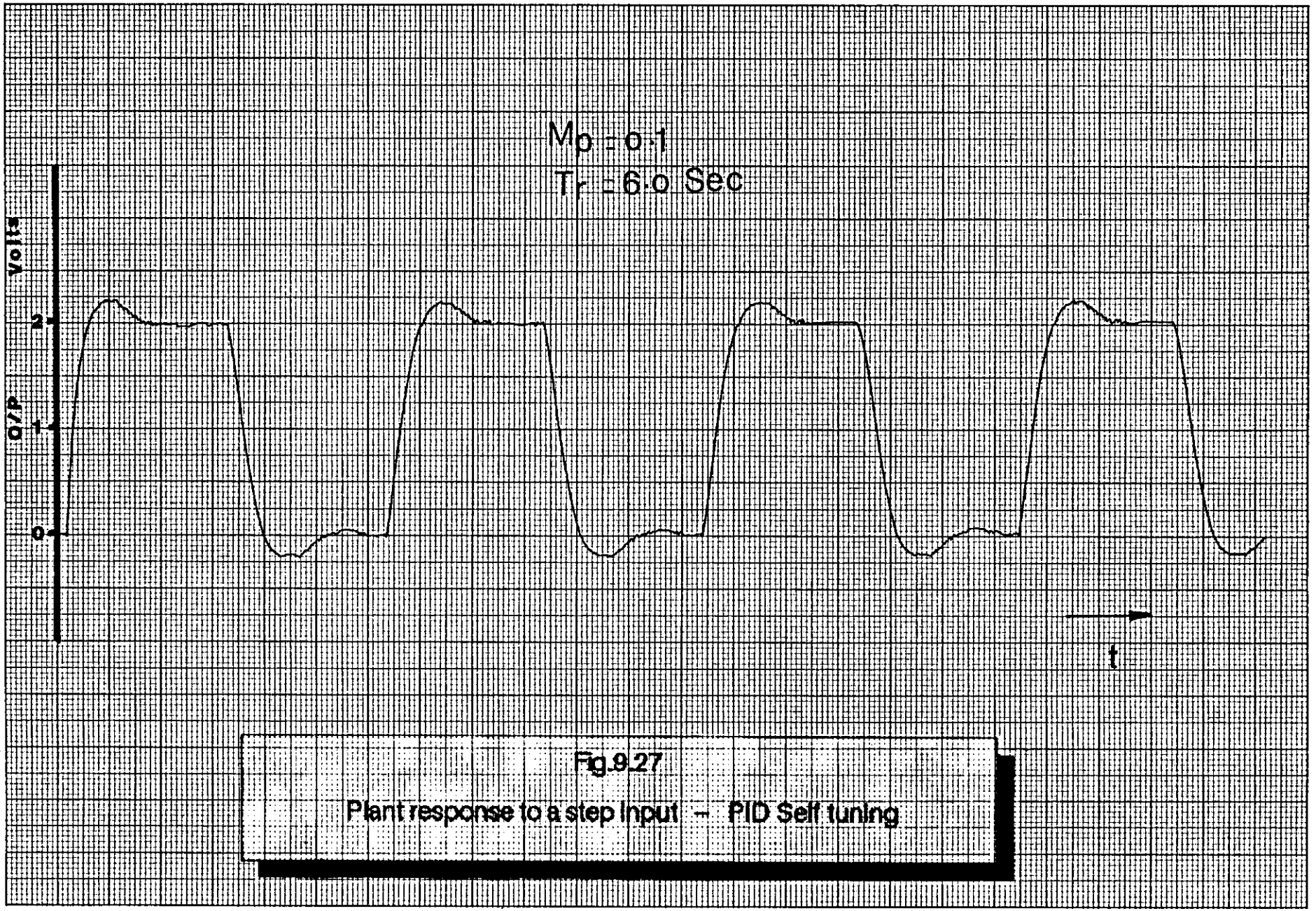


Fig.9.27

Plant response to a step input – PID Self tuning

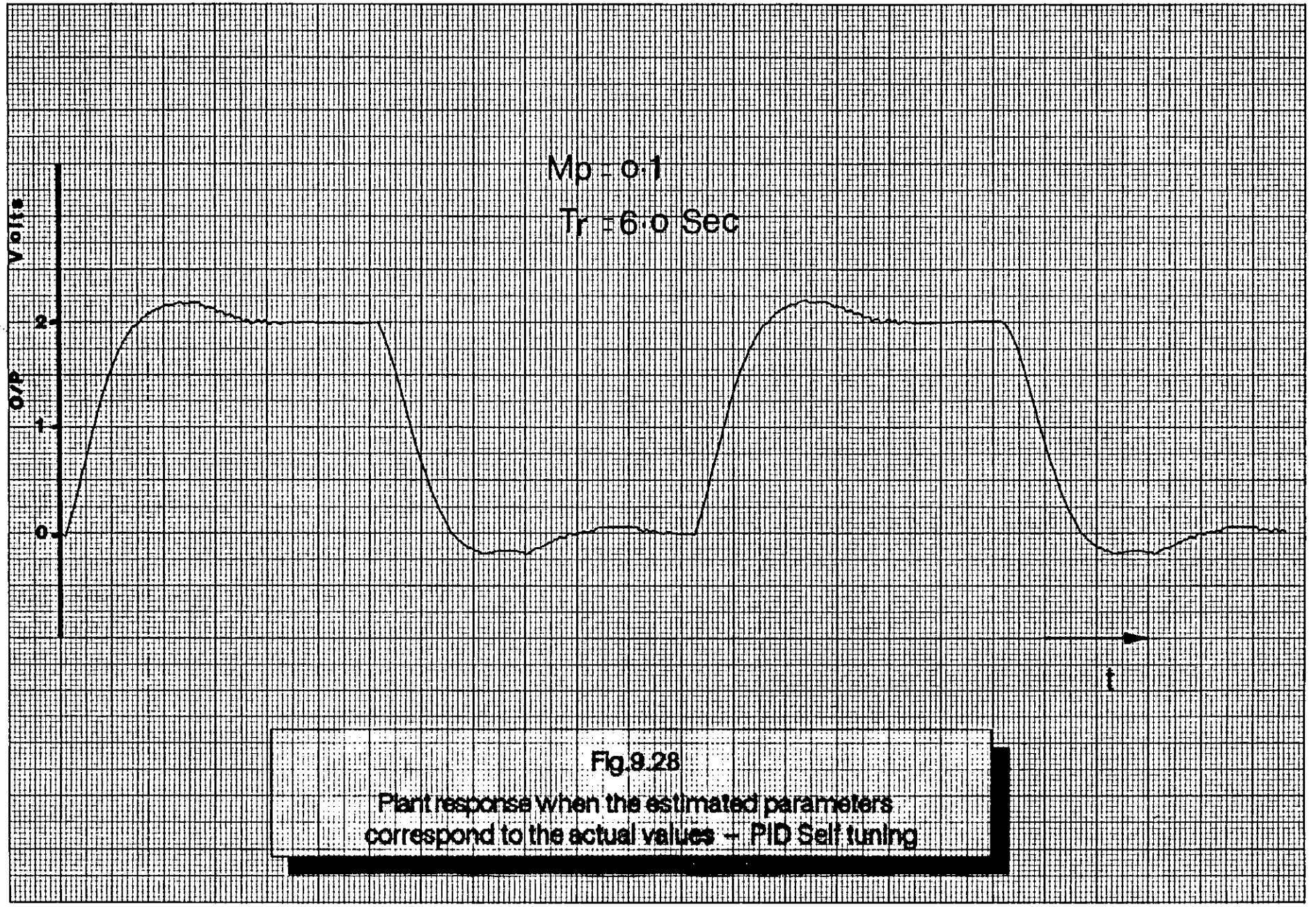


Fig.9.28
Plant response when the estimated parameters correspond to the actual values – PID Self tuning

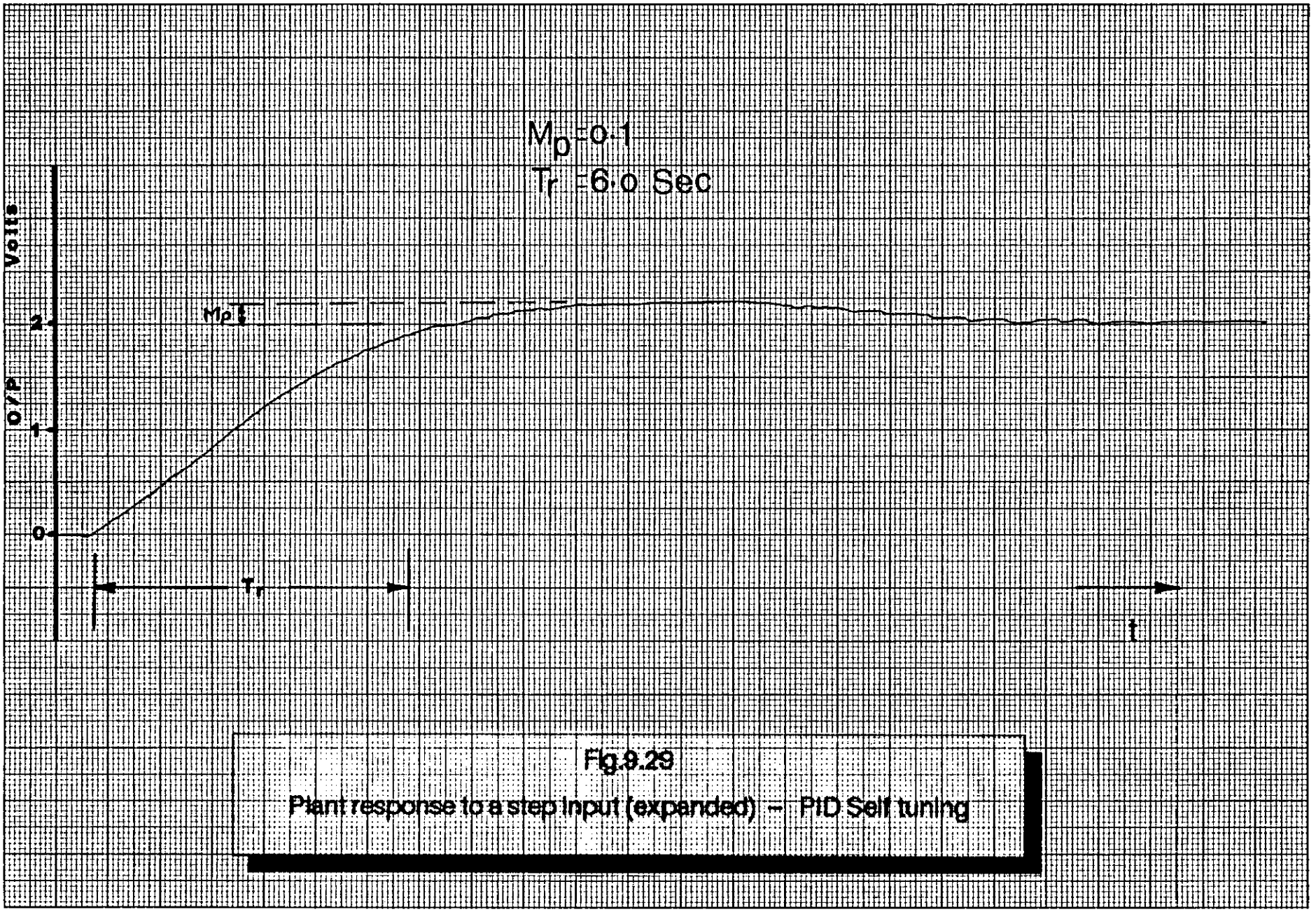


Fig.9.29

Plant response to a step input (expanded) – PID Self tuning

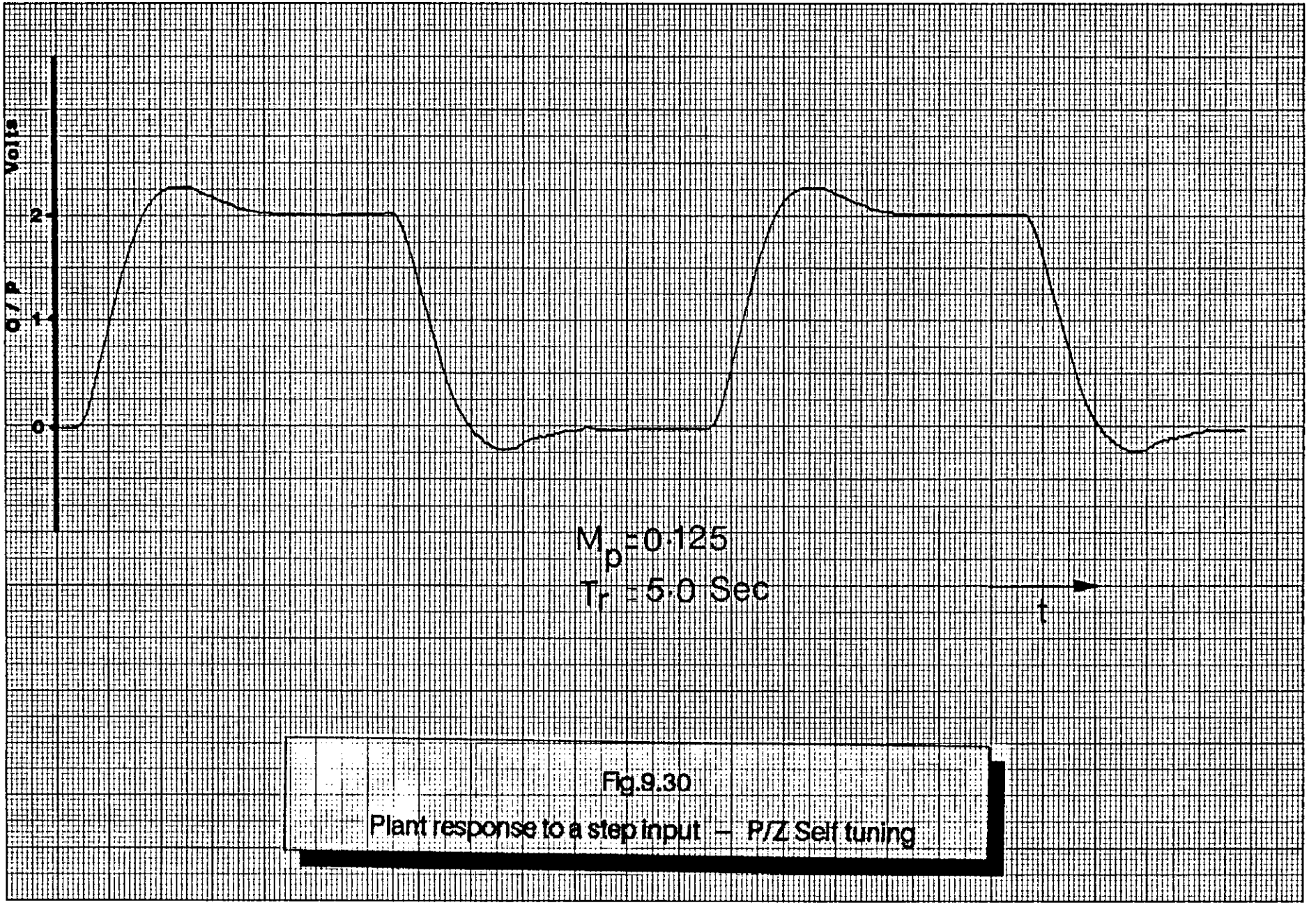


Fig.9.30

Plant response to a step input – P/Z Self tuning

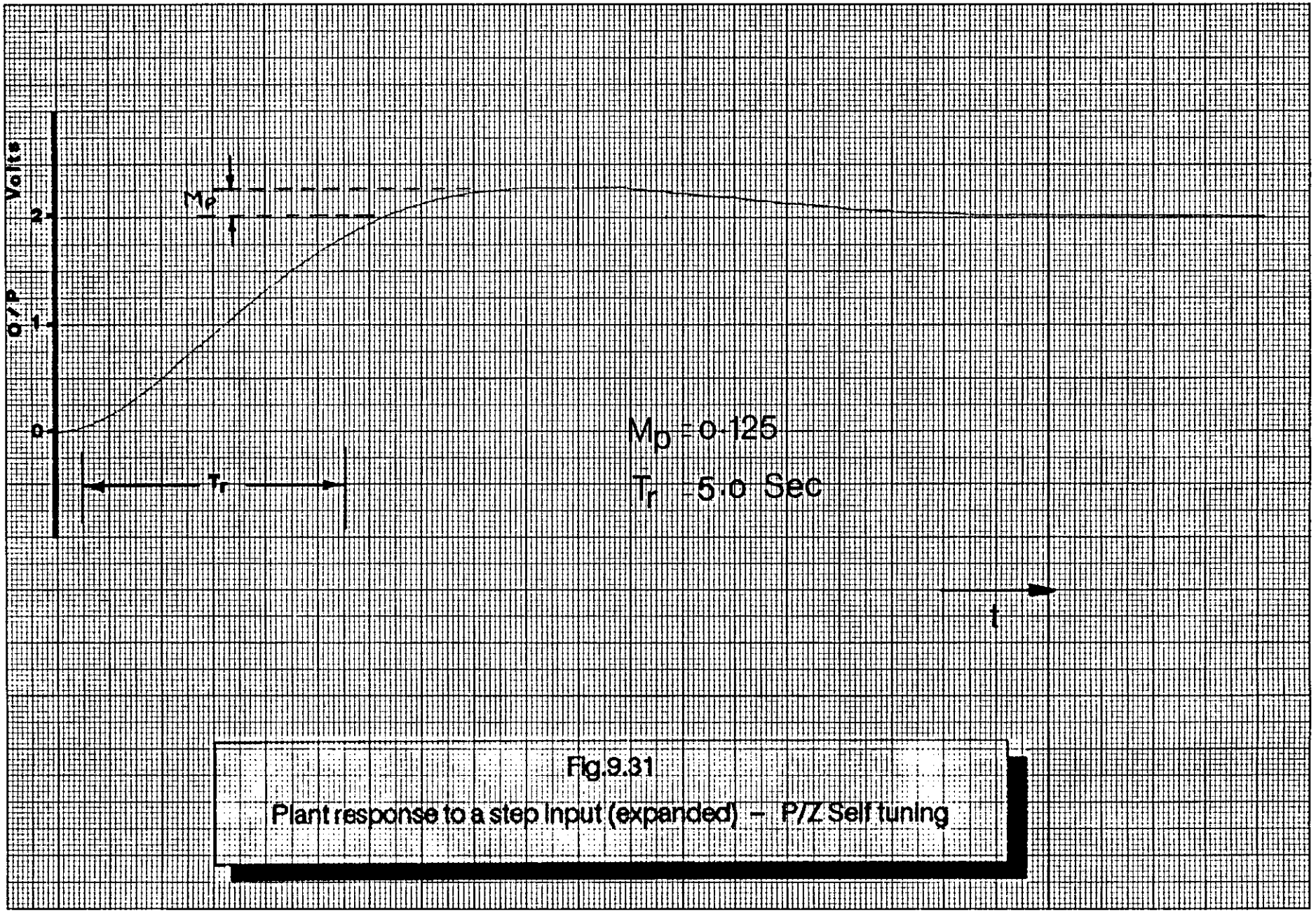


Fig.9.31

Plant response to a step input (expanded) – P/Z Self tuning

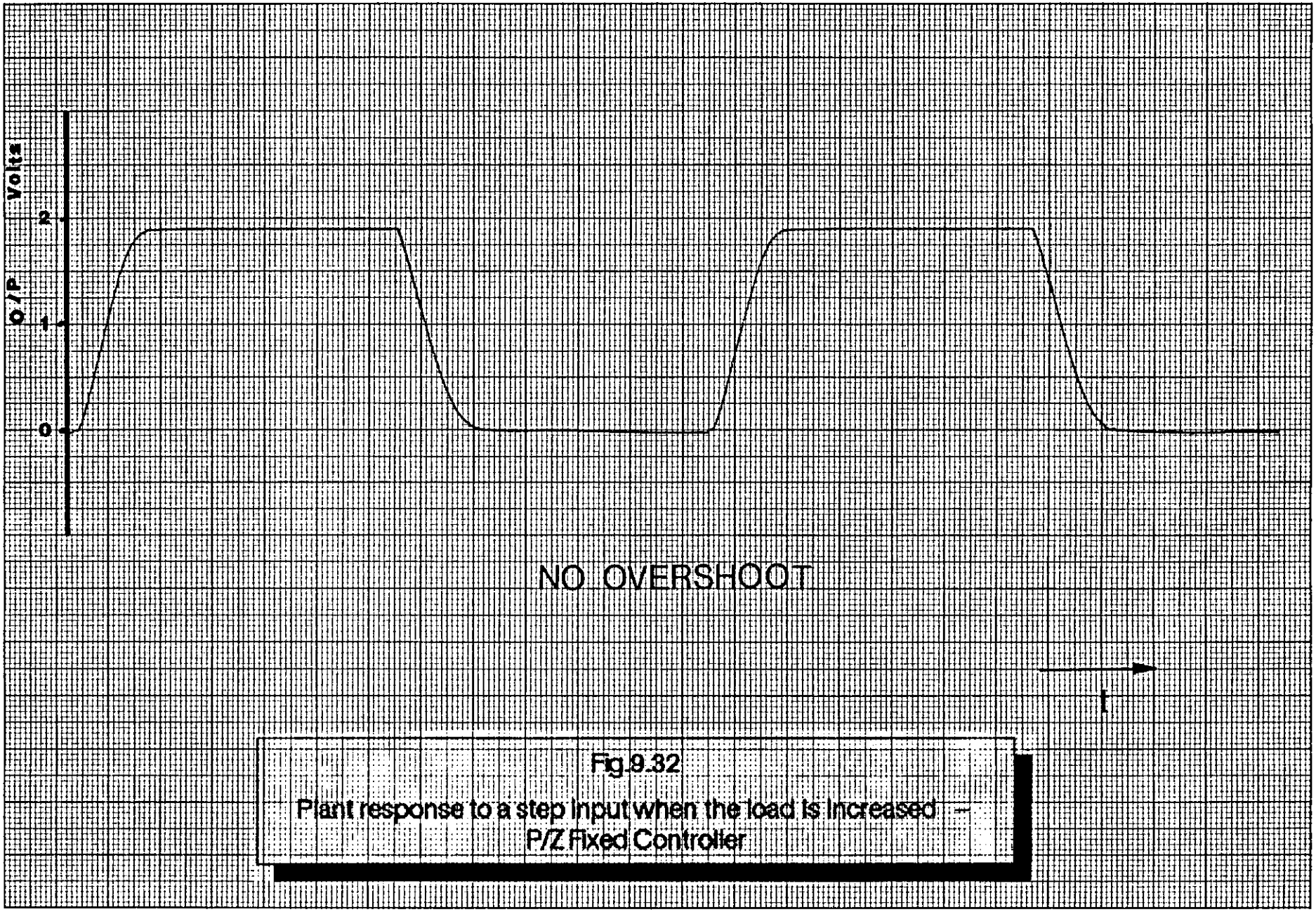


Fig.9.32

Plant response to a step input when the load is increased – P/Z Fixed Controller

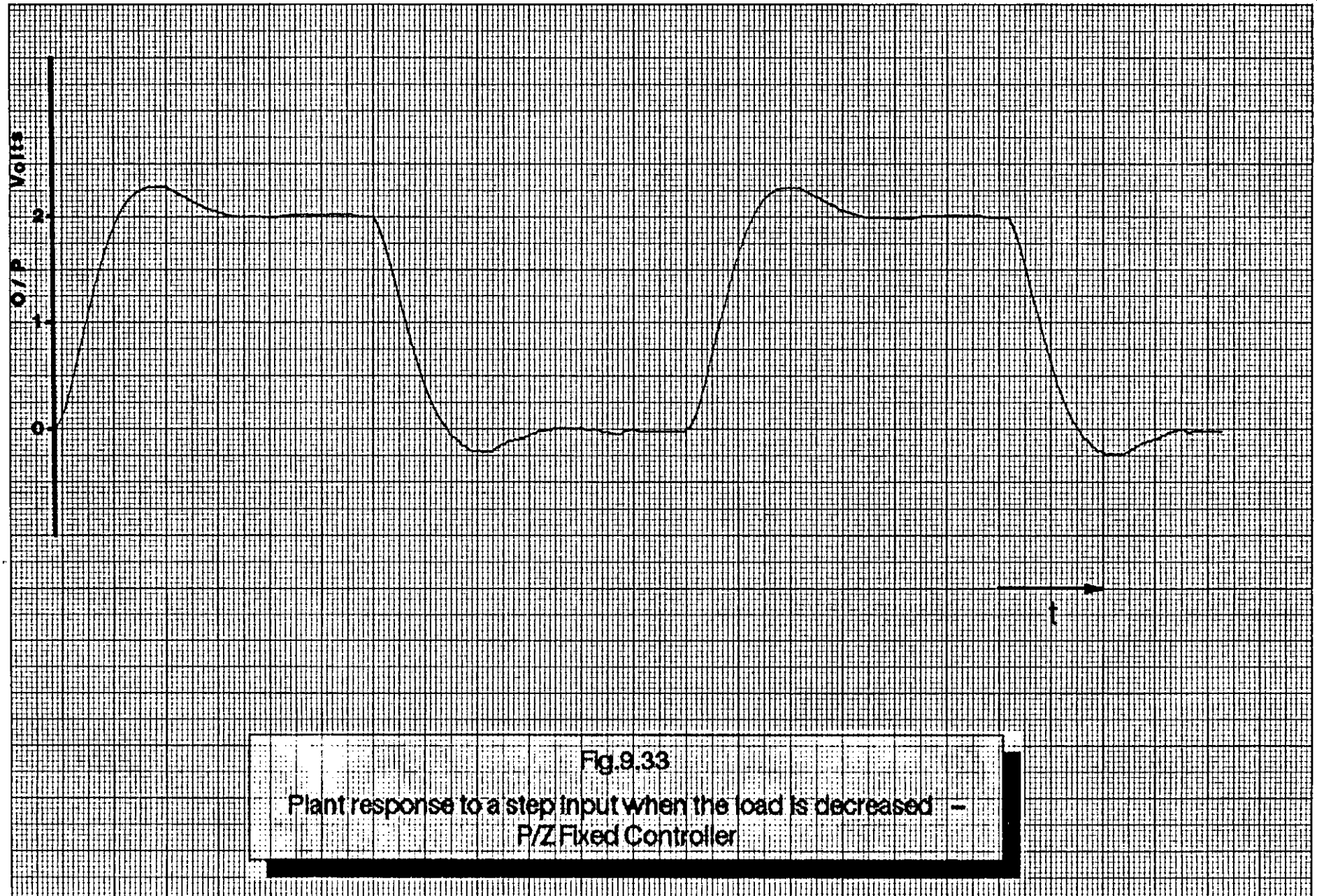


Fig.9.33

Plant response to a step input when the load is decreased – P/Z Fixed Controller

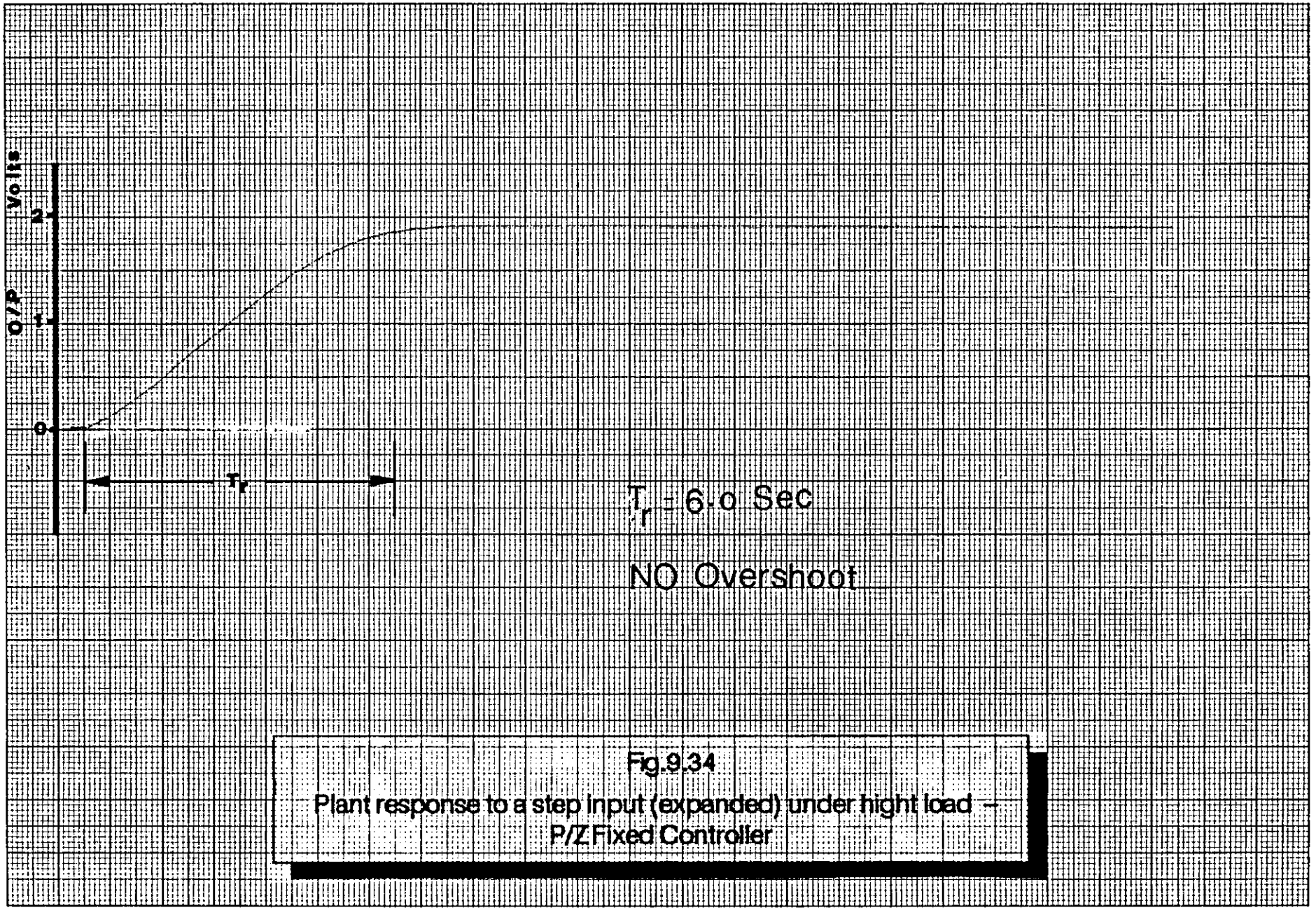


Fig.9.34

Plant response to a step input (expanded) under hight load – P/Z Fixed Controller

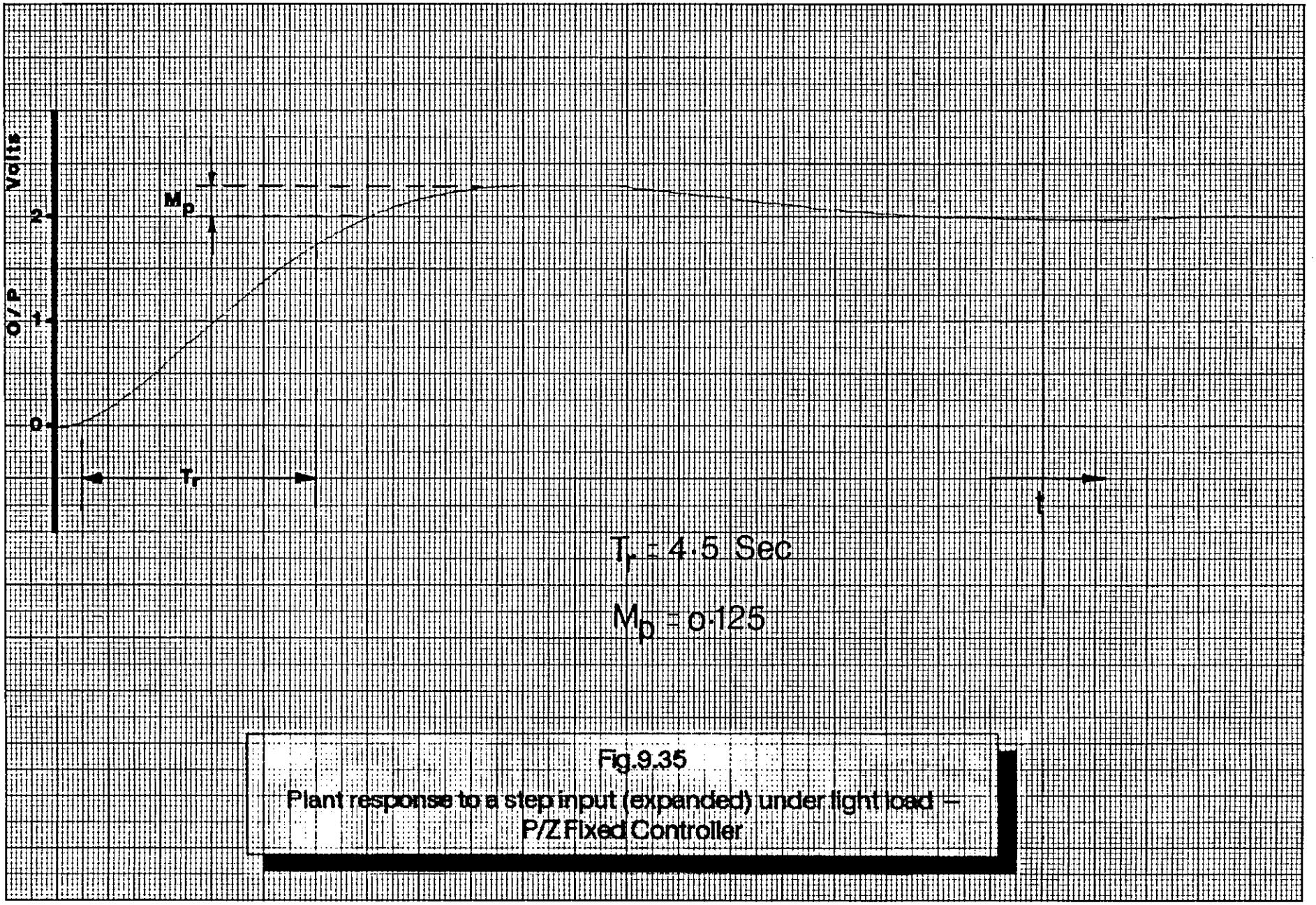


Fig.9.35
Plant response to a step input (expanded) under light load – P/Z Fixed Controller

Fig.9.36

Plant response to a step input as load changes – P/Z Self tuning

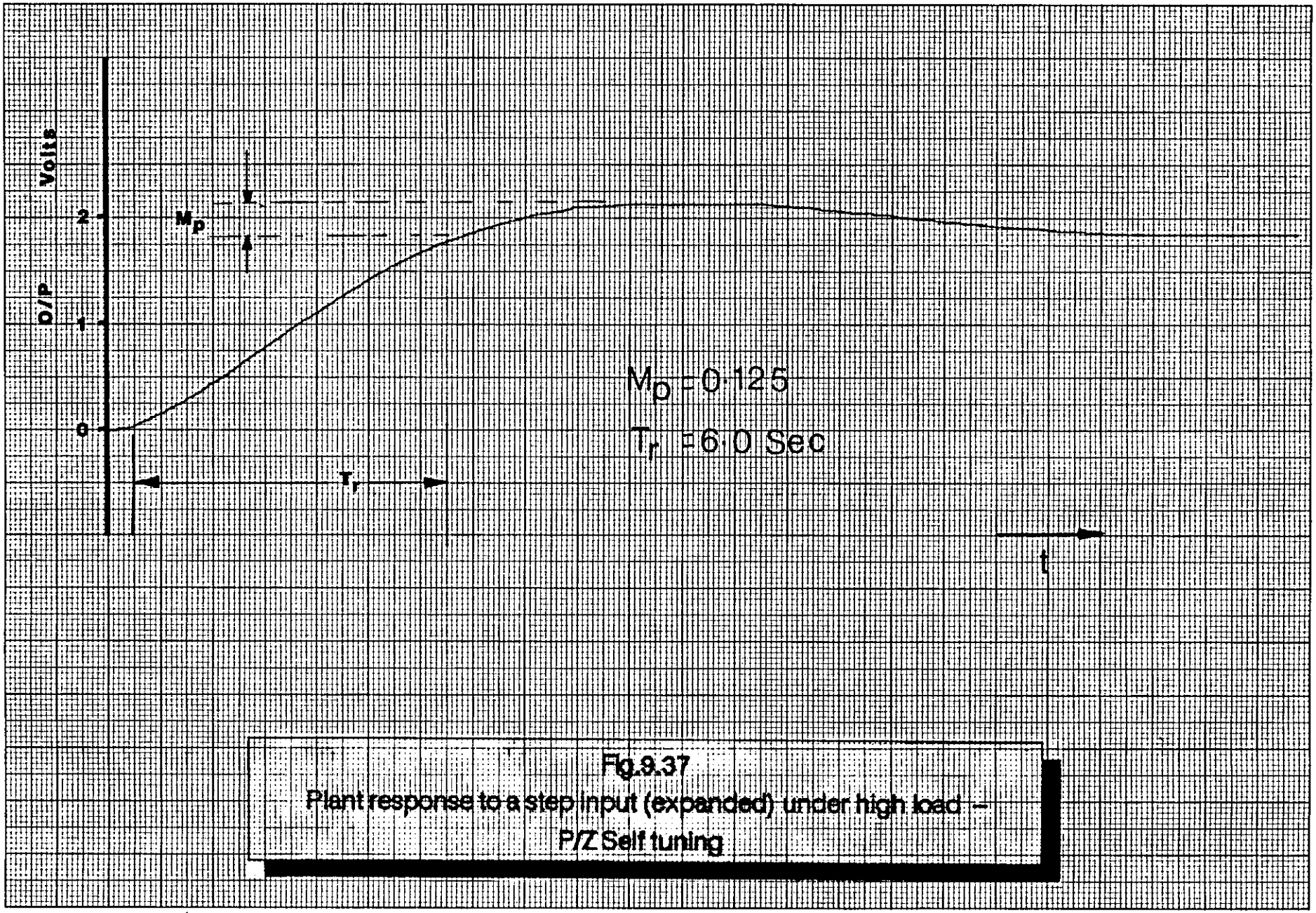
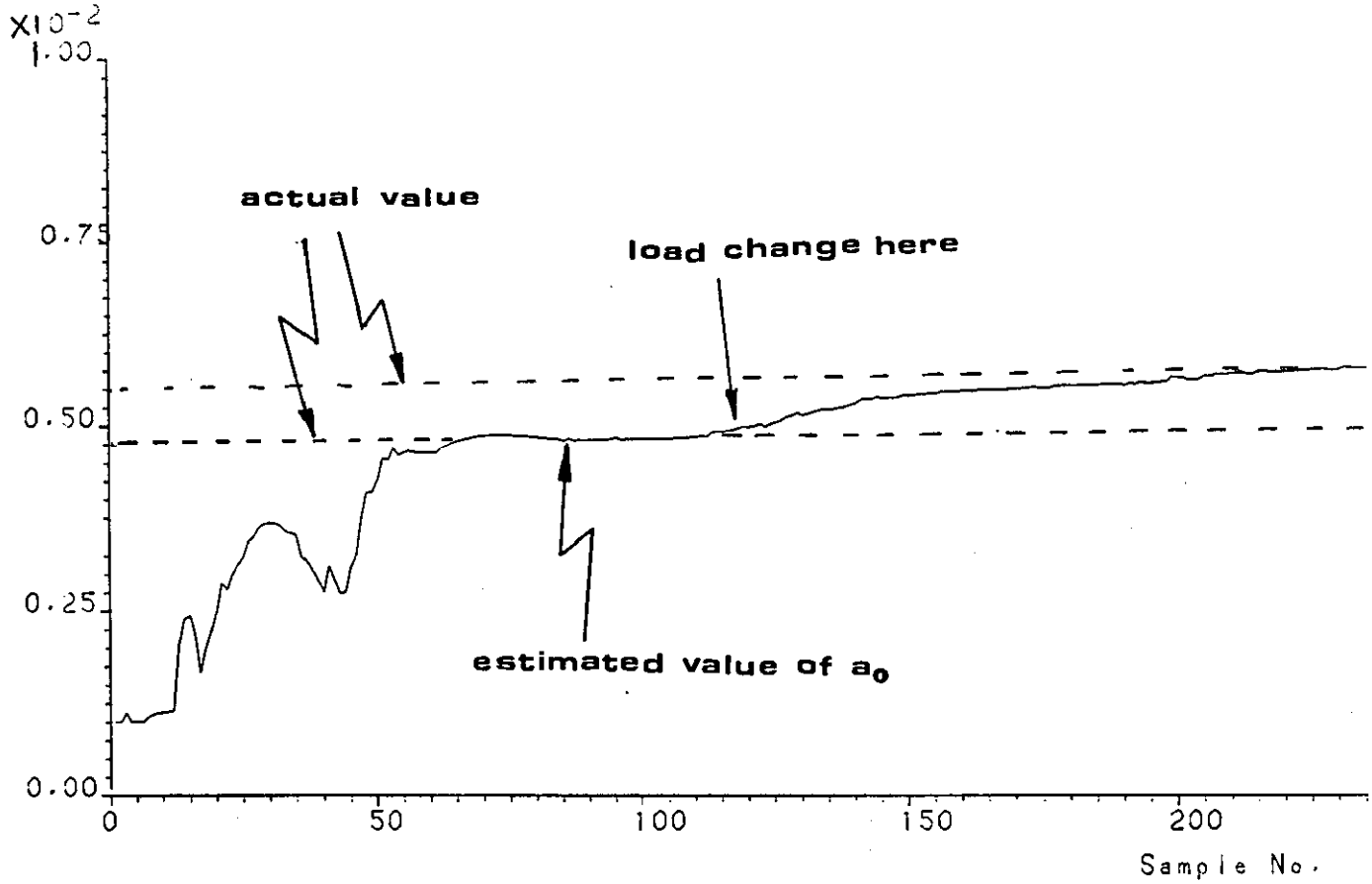


Fig.9.37
Plant response to a step input (expanded) under high load – P/Z Self tuning



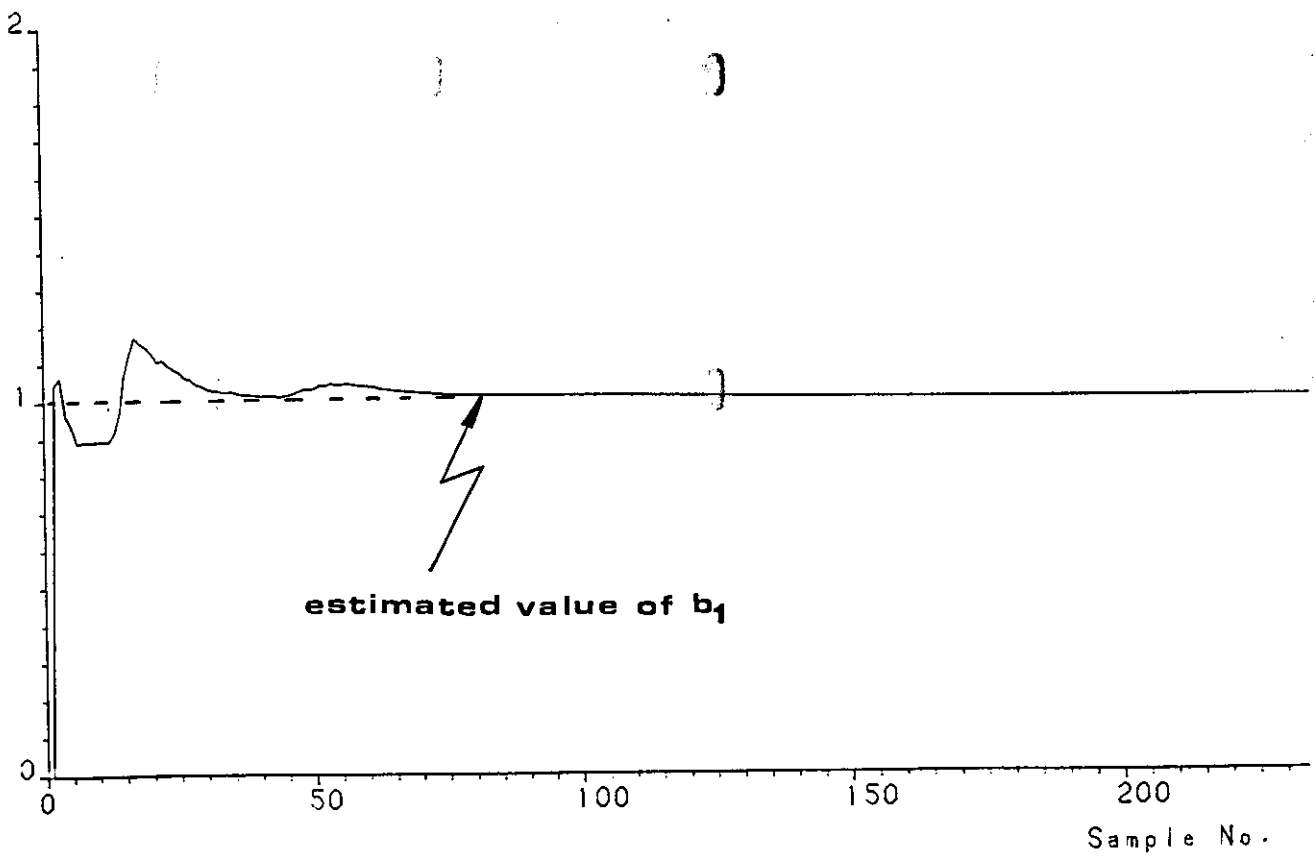


Fig.9.38

Estimator performance as system loading is changed

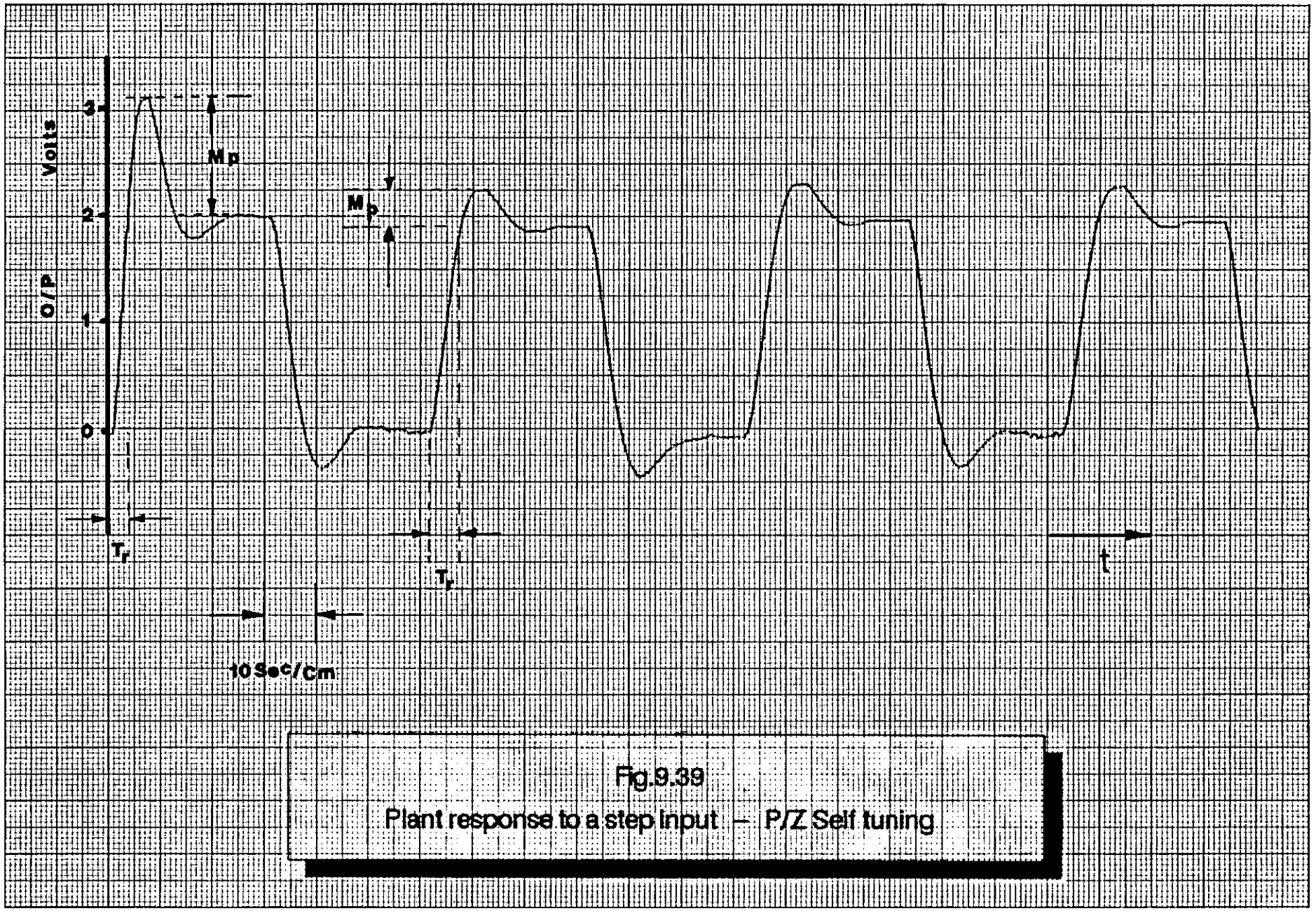


Fig.9.39

Plant response to a step input – P/Z Self tuning

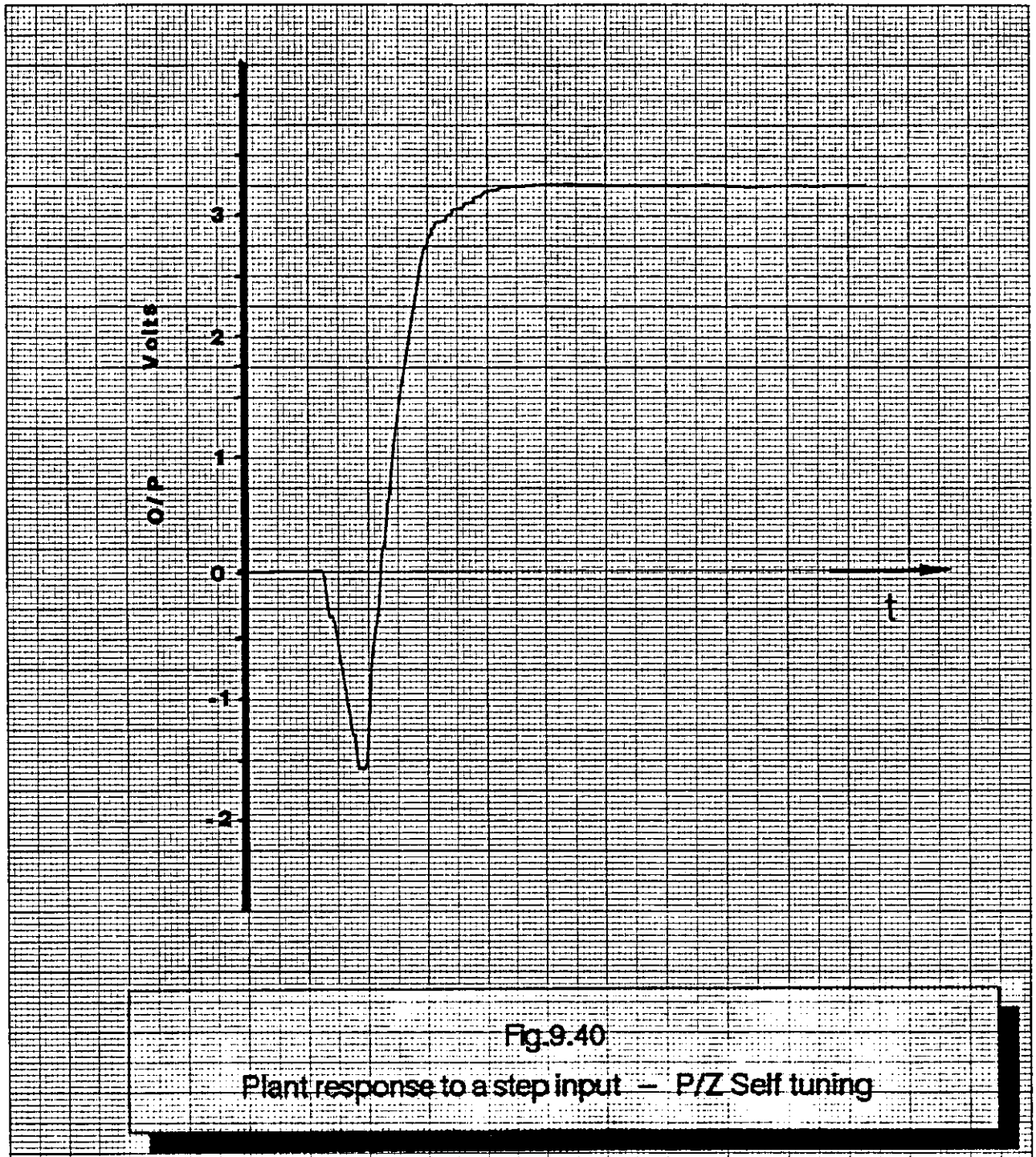


Fig.9.40

Plant response to a step input – P/Z Self tuning

CHAPTER-10

10 CONCLUSION

10.1 A GENERAL COMMENT

In order to review and comment on the work described in this thesis it is essential to understand clearly what is meant by an adaptive control system. Some confusion exists concerning this point. This has arisen because the concept of adaptivity was, in the first place, mixed with the concept of optimality. The original form of adaptive control system was misleadingly referred to as an "optimal control system" by Draper and Li [10.1]. However, an optimal control system is a control system in which where the objective is to minimise a specified cost function.

In contrast, the adaptive (self-tuning) control system is basically a feedback control method that automatically adjusts its control algorithm to achieve a desired system response. This algorithm may be based either on classical OR optimal control techniques. Allied to the controller is an identification process. Its purpose is to supply the controller with a valid model of the plant under all operating conditions. This allows the control algorithm to adapt to variations in the plant parameters.

Adaptive control should not be viewed as a technique which sets out to replace existing control methods. Rather it seeks to improve system performance where time varying plants are encountered. What it does is to eliminate the need for manual tuning of the controller. Instead the controller settings are automatically adjusted as the plant characteristics change, so maintaining the desired closed loop performance.

10.2 CONCLUSIONS

The work described in this thesis has demonstrated the superiority of classical techniques in servo systems (when compared with optimal methods, chapter 4). It has shown that self-tuning designs using classically specified performance criteria are serious contenders for use in modern digital control systems.

It has been shown that pole/zero cancellation and PID self-tuners can be successfully used to regulate fast single-input single-output servo systems, provided that practical constraints are carefully considered.

The self-tuners developed can quickly compensate for large changes in the controlled plant characteristics and are robust in the sense that they are based on robust classical control techniques.

The model order reduction technique used here is an important tool for the simplification of complex mathematical plant models. It has been demonstrated that, as a consequence of model reduction, controller design is considerably simplified.

One final point concerns the use of the 8087 maths co-processor. Tests carried out showed that the speed-up factor attained by using this device lies in the region 20-100. From this it can be concluded that self-tuning controllers must be equipped with powerful maths computing facilities if they are required to work with fast systems.

10.3 SUGGESTION FOR FURTHER WORK

The model order reduction process was done off-line in this work. It is suggested that this should be done on-line in future systems. This will remove many of the limitations of current self-tuning systems, thus making the technique more widely usable and acceptable in practical systems. By adopting this approach the controller behaves like a "black box" that can adapt automatically to meet the

needs of any controllable system.

For slow plants, where the sample time is long, one processor similar to the one designed here is sufficient. Dual sampling rates can be used where system identification and model order reduction can be carried out as a background task while control is performed as a timed, interrupt driven, function.

For fast systems where the sample time is very short, there are two solutions to the problem. These are

* One processor (DSP) or
* Multi-processors

Two main factors have to be considered when deciding which approach should be adopted; these are

* Cost
* High level language support

(a) One processor: A high speed processor that can do the task by itself is the Digital Signal Processor (DSP). This type of processor is designed for arithmetically intensive operations; hence their instruction sets are chosen with this in mind. These complex multipurpose instructions make for very high processing rates. Unfortunately they demand a high level of skill on the part of programmers, who must understand all the ramifications of each instruction if they are to write a compact and efficient code. Code size can be of crucial importance in DSP work, especially if there is a need to keep the chip count down, i.e. avoid the use of off-chip program memory [10.2].

The complications of fixed-point arithmatic are an additional potential headache for the programmer. It has been said that in programming a DSP chip, 90% of the effort goes into worrying about where the decimal point is [10.2].

Existing DSP chips, then, require the services of skilled assembly-code

programmer; unfortunately, such individuals are short in supply. This shortage is partly a reflection of recent developments in general-purpose microprocessors, which are increasingly designed to be programmed in high-level languages. As a consequence, this is seen as a major factor in limiting the use of DSP devices.

(b) Multi-processor: In this approach the task may be divided among the processors. For instance, one processor can be dedicated to control work whilst a second performs system identification and model order reduction.

From the cost point of view there is little to separate this from the first approach. It is to be expected that such design would be based on the use of conventional general purpose microprocessors. Hence the major advantage of the implementation is the ability to program in high level languages and to use floating point maths functions.

APPENDIX - A

A THE CONTROLLED PLANT

A.1 GENERAL DESCRIPTION

A block diagram of the "plant" used for control system development is shown in Fig.A.1, Fig.A.2 being a photo of the actual test rig. This is an electromechanical actuator (suitable for use with process control valves) coupled to a mechanical load simulator. The actuator motor and associated control/power electronics are considered to be part of the plant itself, as is the load shaft position sensor.

The drive unit of the actuator is a 1/6 horse power induction motor, originally designed for 115v. 60 Hz. 3 phase operation. It is powered from a static inverter which, when used with its controller, provides full linear speed control from zero to maximum speed in both directions. Maximum motor shaft speed is 2000 r.p.m., though the maximum shaft speed is 1.2 cm/sec.

A gearbox is used to translate motor shaft rotary motion to linear motion of the load shaft. Both the load resilience and viscous force can be varied by the rig operator. Further, the effect of valve loading is simulated using a coulomb damper (actually a disc brake) on the motor shaft; this, too, is adjustable. Position sensing is carried out using a continuous track rectilinear potentiometer.

Motor speed control is carried out by a pulse width modulated (PWM) controller in conjunction with a 3 phase static inverter [A.1]. The inverter uses power field effect transistors, connected in a full bridge configuration, to switch power to the motor. Motor speed is determined by the switching frequency of the transistors, this being set by the PWM controller. In turn this is determined by the analogue input signal to the controller, the switching frequency range being 0 to 1 KHz. (approximately). Constant flux conditions within the motor are maintained by

modulating the drive pulse width, though voltage boosting is used at low frequencies to compensate for shaft stiction effects. Thus the motor drive frequency is directly proportional to the input signal, setting the synchronous speed of the motor.

A.2 MECHANICAL DESCRIPTION

Fig.A.3 shows a block diagram of the mechanical part of the plant. This consists of

* An induction motor
* A gearbox
* A disc brake
* A shaft
* Disc spring compressor
* Hydraulic cylinder
* A potentiometer

(a) The induction motor: This is a 1/6 horse power induction motor, originally designed for 115V. 60 Hz. 3 phase operation. It is powered from a static inverter. Maximum motor shaft speed is 2000 r.p.m..

(b) The gearbox: This is used to translate motor shaft rotary motion to linear motion of the load shaft. The gear ration is 1/10.

(c) The disc brake: This simulates a coulomb damper. The amount of brake force is adjusted manually using the hydraulic valve mounted on the brake unit.

(d) The shaft: The shaft length is 50 cm. and its maximum speed is 1.2 cm/sec.

(e) Disc spring compressor: Fig.A.4 shows a diagram of the compressor. It provides different forces on the shaft, by rearranging the ten disc springs supplied, from 500 lb to 1000 lb. The compressor can be pushed or pulled to compress the springs.

(f) The hydraulic cylinder: Fig.A.5 shows a diagram of this device. Its purpose is to act as a viscous load on the shaft; it too, is adjustable, and can be varied manually by the operator. This device consists of a cylinder with a shaft that is connected to the rig shaft. The two ends of the cylinder are connected together via a pipe. Oil flows through this pipe as the shaft travels through the cylinder. Within this pipe is a flow control valve which acts as a resistance to the flow of oil, i.e. a viscous damper. The viscous force experienced by the shaft is determined by the valve setting.

(g) The potentiometer: Position sensing is accomplished through the use of a rectilinear continuous track potentiometer mounted on the end of the shaft.

A.3 ELECTRONICS

The electronics involved is devided into two parts, these being

* The analogue interface board
* The inverter/actuator board

A.3.1 Analogue interface board

(a) Functions and Facilities

(i) Functions:

* Provides protection features for the inverter under all operational conditions
* provides constant-flux operation within the motor
* provides all signals required by the inverter LSI PWM controller chip.
* Interfaces between the microcontroller and the inverter-actuator unit.

(ii) Facilities

* Adjustment of motor speed in both directions from zero to 1.5 times its nominal speed.
* Limitation of regenerated power during speed deceleration to protect the inverter against overvoltage.
* Adjustment of starting torque values via "IR" compensation [A.1].

(b) Analogue interface operation

The block diagram for the open loop system is shown in Fig.A.6, and consists of the following sub-sections:

(a) ON-OFF circuit: This controls the inverter on/off function in response to external push button controls of inverter overcurrent fault conditions.

(b) Potentiometer: This controls the motor speed in both directions.

(c) Soft start/stop circuit: Ramps the voltage output from the potentiometer.

(d) Acceleration deceleration circuit: Adjusts the time constant of the ramped voltage.

(e) Direction detector circuit: Alters the direction of rotation of the motor.

(f) Absolute value unit: Provides positive voltage for both positive and negative input ramped voltages.

(g) Voltage-controlled-oscillator and Control Logic Unit: Provides clocks needed by the PWM controller chip.

A.3.2 Inverter/actuator structure

The typical system block diagram is shown in Fig.A.7. The incoming 3-phase a.c. supply is stepped down by a 3-phase auto-transformer to an r.m.s. voltage of 115V. This voltage is rectified and smoothed to produce about 160V and this is fed to the three-phase inverter via a current-sensing circuit. The inverter chops the d.c. to give an output of 160V peak-to-peak pulse width modulated at a maximum frequency 1 kHz. This output is fed to the a.c. motor which responds mainly to the envelope of the PWM switching frequency.

The six PFET switches in the inverter are under the command of a waveform-generation circuit which determines the conduction time of each switch. Since the control electrodes of the six switches are not at the same potential, the outputs of the waveform-generation circuit must be isolated and buffered. A low-voltage power supply feeds the low-power signal processing circuit, and a further low-voltage power supply drives a transistorised switch-mode isolating stage to provide floating power supplies to the gate drive circuits [A.1].

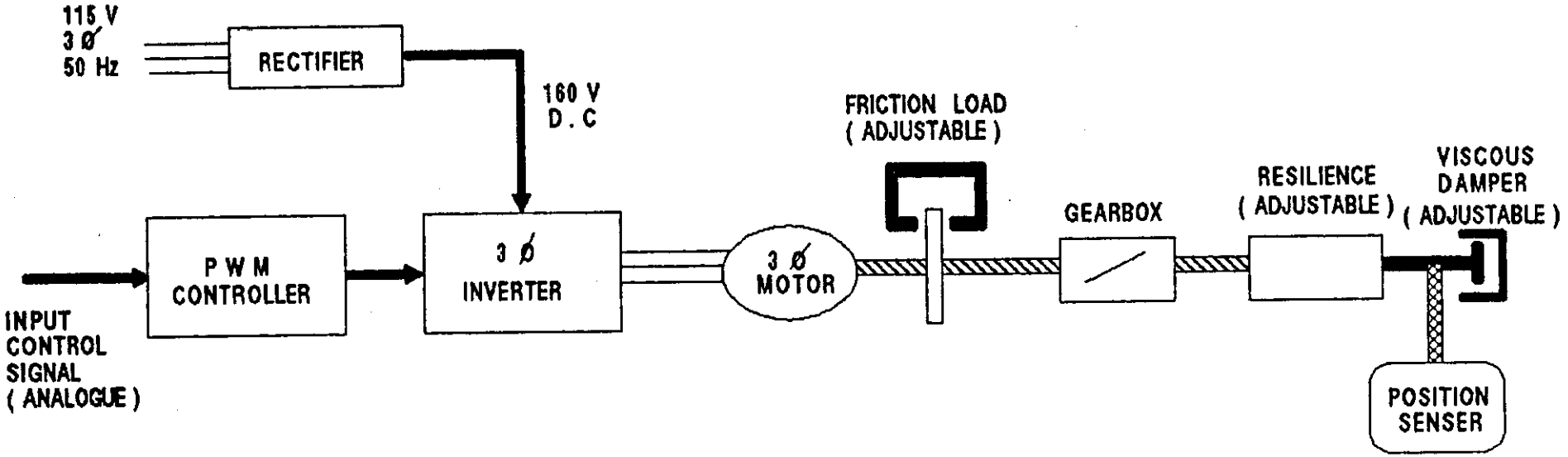


FIG . A . 1 THE CONTROLLED PLANT

Fig . A . 2

THE TEST RIG

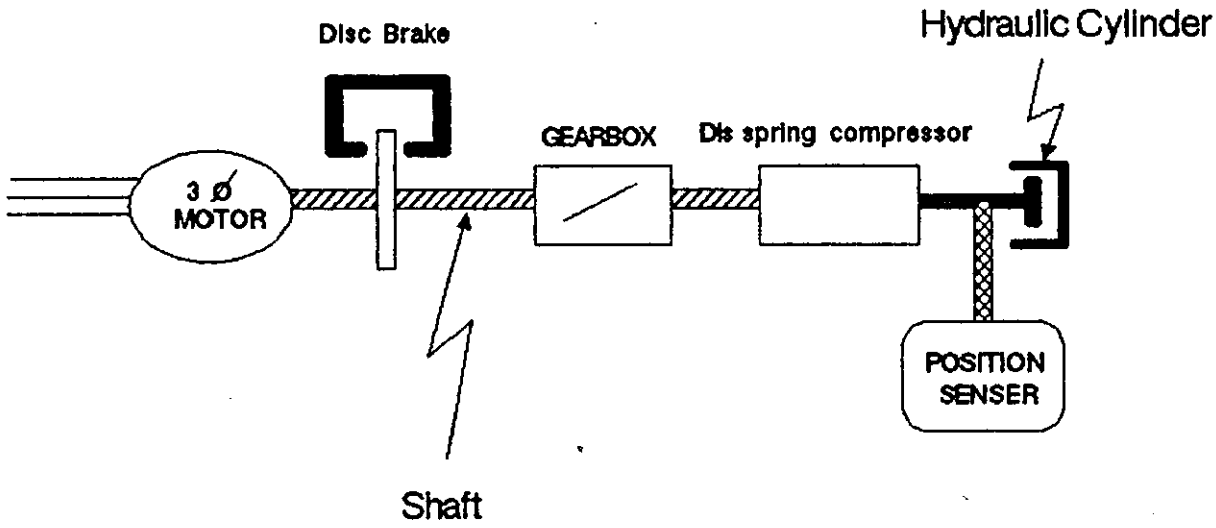


Fig . A . 3

Mechanical Part of the Plant

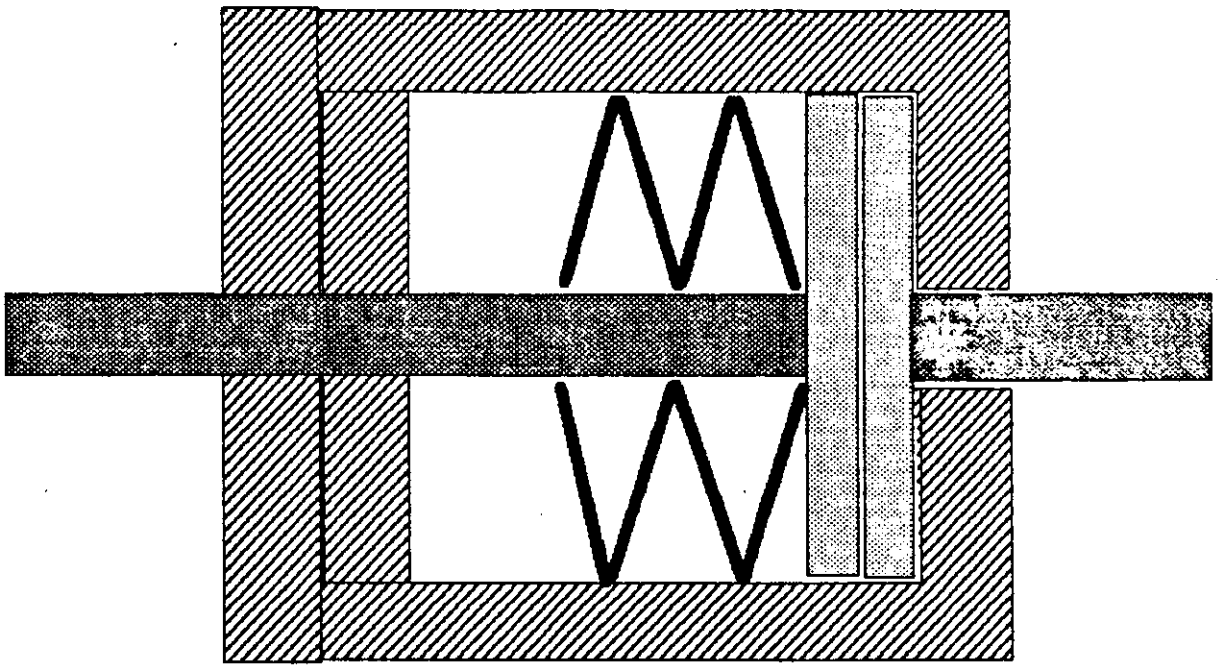


Fig . A . 4 Disc Spring Compressor

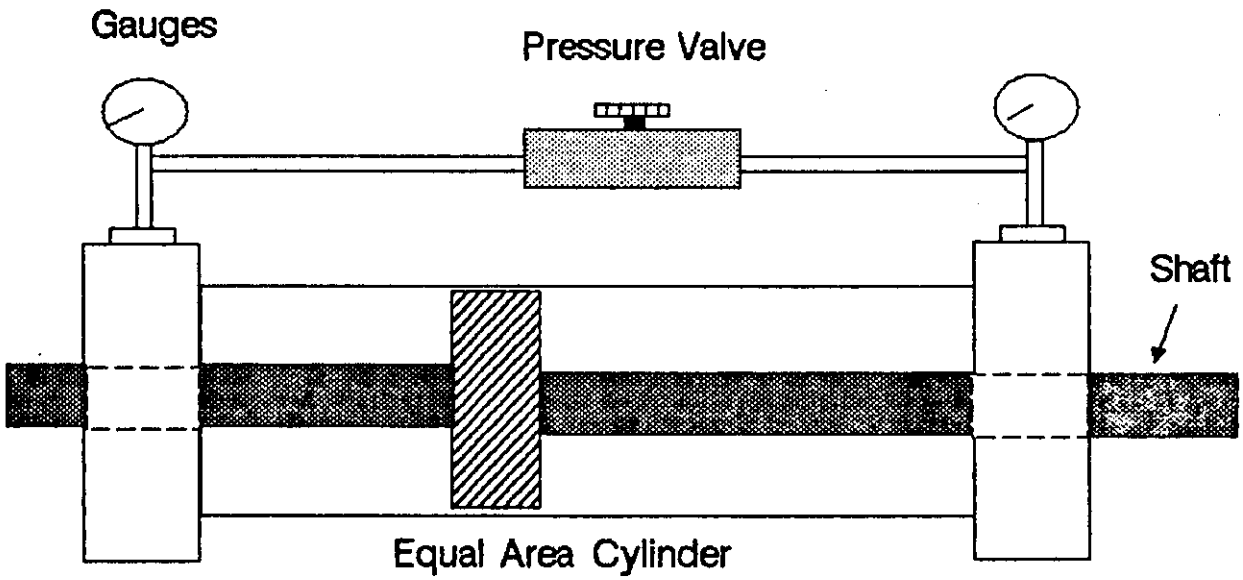


Fig . A . 5 Hydraulic Cylinder

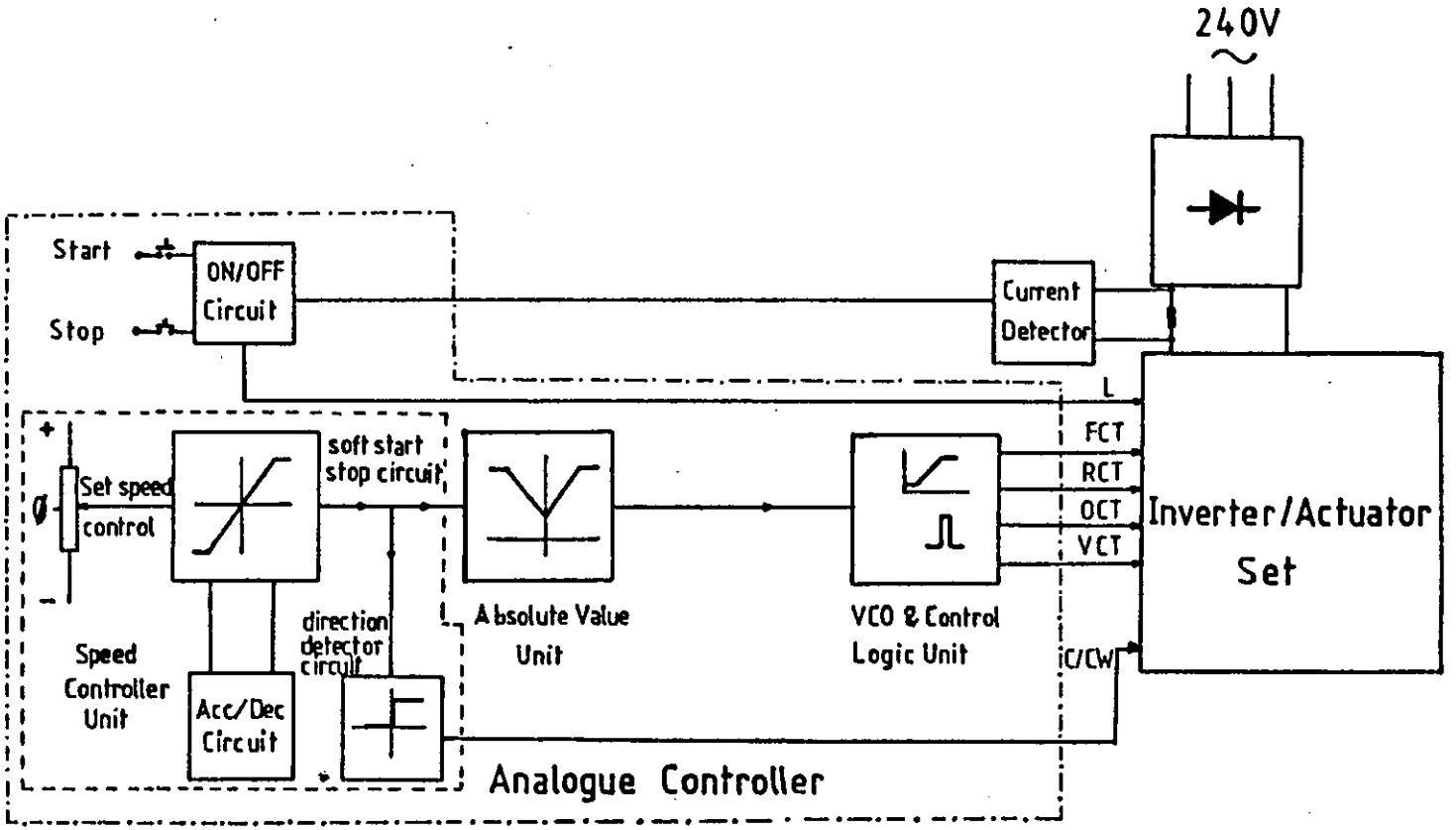


Fig. A.6 Open Loop System (Analogue Controller)

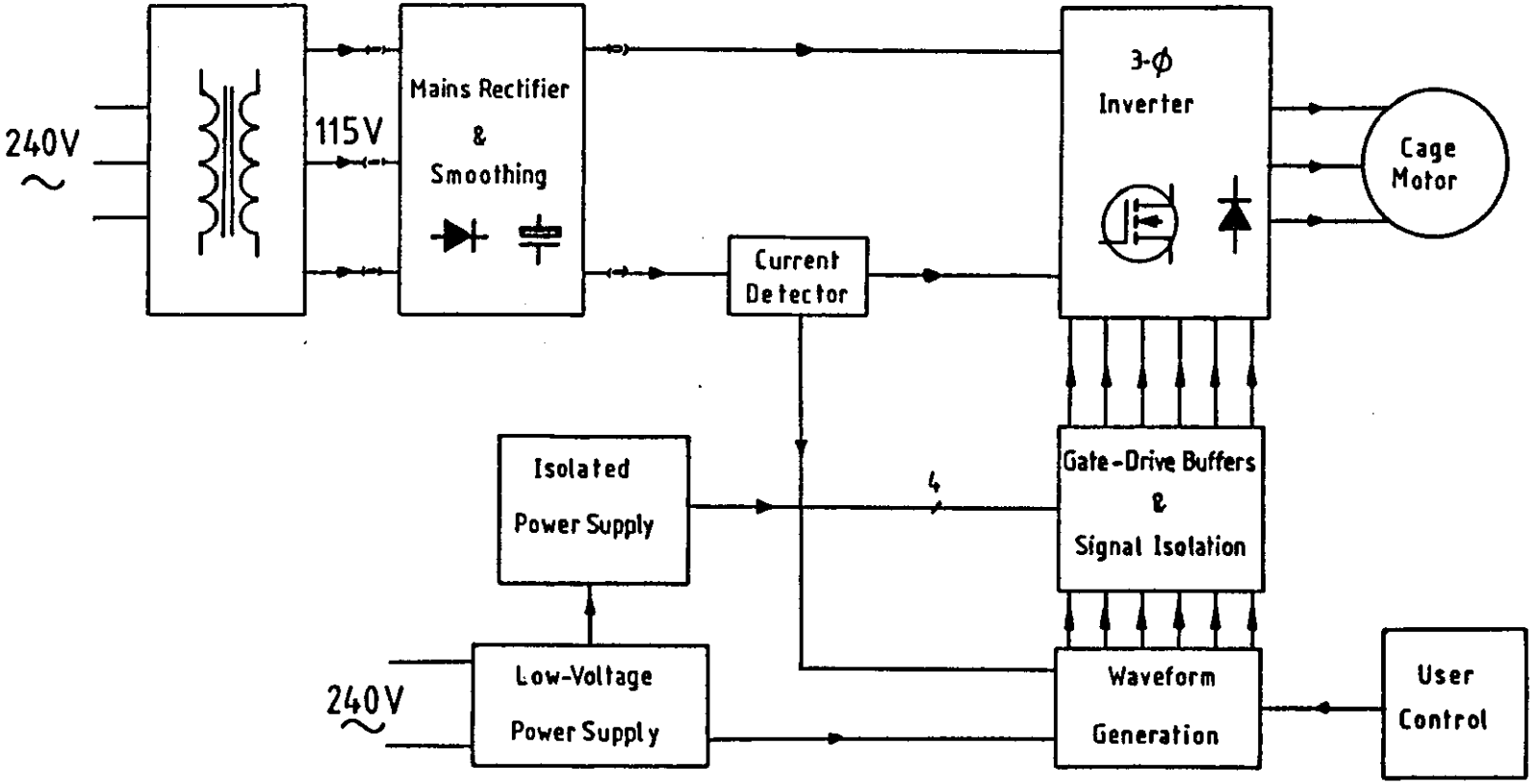


Fig.A.7 System Block Diagram

APPENDIX - B

B SYSTEM IDENTIFICATION

B.1 PARAMETER ESTIMATION METHODS

In general, solving the parameter estimation problem requires:

* Input-output data from the process.
* A defined model structure.
* A criterion for parameter estimation.

Parameter estimation can be formulated as an optimization problem in which the best model is the one that best fits the data according to the given criterion [B.2]. There are a large number of methods for analyzing data obtained from experiments and one major distinction is between on-line and off-line methods. On-line methods give estimates recursively as the measurements are obtained while off-line methods first store the input/output data records and then estimate a model by using the entire data set. If identification is to be used in an adaptive controller then on-line recursive identification methods are appropriate. Fig.B.1 shows a block diagram of an on-line identification scheme.

When considering the selection of a parameter estimation method applicable in adaptive control, the following items have to be considered [B.5]:

* The accuracy of the identified model and the convergence properties of the estimated parameters to the actual ones.
* Computational effort (computer storage, computation time).

Appropriate parameter estimation schemes for adaptive controllers include [B.6]:

* Recursive Least Square (RLS)
* Recursive Extended Least Square (RELS)
* Recursive Maximum Likelihood (RML)
* Recursive Instrumental Variable (RIV)

* Stochastic Approximation (STA).

B.2 MODELS

We shall now define the notation and conventions to be used for describing discrete time models. Fig.B.2 shows the model to be used in all the latter identification methods. Such a model has the following general structure [B.8]:

$$B(z^{-1})y(T)= A(z^{-1})u(T) + H(z^{-1})e(T) \qquad (B.1)$$

where e(T) denotes the noise, y(T) denotes the output and u(T) the input. $B(z^{-1})$ and $A(z^{-1})$ are polynomials in the backward shift operator z^{-1},

$$B(z^{-1})= 1 + b_1z^{-1}+.....+ b_{nb}z^{-nb}$$

$$A(z^{-1})= a_1z^{-1}+.....+ a_{na}z^{-na}$$

where z^{-n} y(T) notationally denotes the value of y(T-n). The term $H(z^{-1})$ has a different structure assumed by the various methods [B.8]. For RLS and RIV

$$H(z^{-1})= 1$$

while for RELS and RML

$$H(z^{-1})= C(z^{-1})= 1 + c_1z^{-1}+.....+ c_{nc}z^{-nc}$$

B.3 A GENERAL CRITERION

All the methods mentioned above fall within the class of prediction error identification methods [B.7]. In the following example a simple model is considered to demonstrate the idea of prediction error minimisation. Consider the model

$$y(T) + b\, y(T-1) = a\, u(T-1) + e(T) \qquad \text{(B.2)}$$

in which e(T) is zero mean white noise. The parameters to be estimated are collected into the vector:

$$\underline{\theta}^T = [\, b \quad a \,]$$

First let us discuss how to determine the prediction of y(T) based on observations of y(i),u(i), $0 < i < T-1$, and based on the assumption that the data is produced by the model(B.2). We denote such a prediction by $y(T/\underline{\theta})$, which means the predicted value of y at time t based on the known data y(T-1), y(T-2)..., u(T-1), u(T-2) ... . We shall make an elementary derivation of $y(T/\underline{\theta})$ for this simple example; the general case will be shown later. From Eq.(B.2)

$$y(T) = -b\, y(T-1) + a\, u(T-1) + e(T) \qquad \text{(B.3)}$$

Here y(T-1) and u(T-1) are known at time T. The value of e(T) cannot be predicted from previous data since it is independent of everything that happened up to time T-1 and by definition of e(T) the expected value of e(T) is zero. Hence, the natural prediction of y(T) is [B.7]:

$$y(T/\underline{\theta}) = -b\, y(T-1) + a\, u(T-1) \qquad \text{(B.4)}$$

Introducing $\hat{y}(T)$ as the predicted value $y(T/\underline{\theta})$, we have from Eq.(B.3) and Eq.(B.4) that the prediction error can be evaluated

$$e(T) = y(T) - \hat{y}(T) \qquad \text{(B.5)}$$

Defining $\underline{q}(T)$ as the vector containing the measured values at time T-1:

$$\underline{q}(T)^T = [\, -y(T-1) \;\; u(T-1)]$$

then

$$\hat{y}(T) = \underline{\theta}^T \, \underline{q}(T) \qquad \text{(B.6)}$$

From Eq.(B.5) and Eq.(B.6) the model Eq.(B.3) can be written as:

$$y(T) = \underline{\theta}^T \, \underline{q}(T) + e(T) \qquad \text{(B.7)}$$

Consider now a general model:

$$B(z^{-1})y(T) = A(z^{-1})u(T) + C(z^{-1})e(T) \qquad \text{(B.8)}$$

where, as before, e(T) is a sequence of independent random variables and $B(z^{-1})$, $A(z^{-1})$ and $C(z^{-1})$ are polynomials in the delay operator z^{-1}.

$$B(z^{-1}) = 1 + b_1 z^{-1} + \ldots\ldots + b_{nb} z^{-nb}$$

$$A(z^{-1}) = a_1 z^{-1} + \ldots\ldots\ldots + a_{na} z^{-na}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + \ldots\ldots + c_{nc} z^{-nc}$$

We shall now derive the prediction error in this more general case [B.1]. Let us introduce the parameter vector

$$\underline{e}^T = [\, b_1 \ldots b_{nb}\ a_1 \ldots a_{na}\ c_1 \ldots c_{nc}\,]$$

Dividing Eq.(B.7) by $C(z^{-1})$ we have

$$B(z^{-1})/C(z^{-1})\ y(T) = A(z^{-1})/C(z^{-1})\ u(T) + e(T) \qquad (B.9)$$

or

$$y(T) = [1 - B(z^{-1})/C(z^{-1})]y(T) + A(z^{-1})/C(z^{-1})\ u(T) + e(T) \qquad (B.10)$$

Since the polynomials $B(z^{-1})$ and $C(z^{-1})$ have unity as the coefficient of their constant terms then the coefficient of the constant term in

$$[\ 1 - B(z^{-1})/C(z^{-1})]\ y(T)$$

is zero. Thus the right-hand side of (B.10) is known at time T-1 with the exception of the term e(T) which is independent of everything that has happened up to time T-1.

Therefore, the natural predictor is [B.7]:

$$\hat{y}(T) = [1 - B(z^{-1})/C(z^{-1})]\ y(T) + A(z^{-1})/C(z^{-1})\ u(T) \qquad (B.11)$$

which gives

$$C(z^{-1})y(T)= [C(z^{-1}) - B(z^{-1})]y(T) + A(z^{-1})\ u(T) \qquad (B.12)$$

This is a convenient finite difference equation for calculating y [B.7]. The initial condition is often $y(T)=0$ for $T<0$, corresponding to an assumption that $y(T)= u(T)=0$ for $T<0$. The error is then

$$e(T)= y(T) - \hat{y}(T) \qquad (B.13)$$

From Eq.(B.12)

$$C(z^{-1})\ e(T)= B(z^{-1})y(T) - A(z^{-1})u(T) \qquad (B.14)$$

which can be written as

$$e(T)= y(T) - \underline{e}^T\underline{q}(T) \qquad (B.15)$$

with

$$\underline{e}^T=[b_1 \ldots b_{nb}\ a_1 \ldots a_{na}\ c_1 \ldots c_{nc}]$$

and

$$\underline{q}^T(T)=[-y(T-1)/C(z^{-1}),\ \ldots,\ -y(T-n_b)/C(z^{-1}),\ u(T-1)/C(z^{-1}),$$
$$\ldots,\ u(T-n_a)/C(z^{-1}),\ e(T-1)/C(z^{-1}),\ \ldots,\ e(T-n_c)/C(z^{-1})] \qquad (B.16)$$

A reasonable criterion of how well the model $\underline{e}^T\underline{q}$ performs is to minimise some function of the prediction errors e(T). One procedure is to minimise a cost function

J which is defined as the sum of the squares of the errors

$$J = 1/2 \sum_{T=1}^{N} e(T)^2 \qquad (B.17)$$

where

$$e(T) = y(T) - \hat{y}(T)$$

B.4 RECURSIVE LEAST SQUARES (RLS)

B.4.1 General

The LS method was first used by Gauss in 1809. Its recursive version has apparently been found independently by several authors [B.7]. The original reference seems to be Plackett (1950) [B.23].

The RLS method assumes a model of the form (B.1) with $H(z^{-1})=1$:

$$B(z^{-1})y(T) = A(z^{-1})u(T) + e(T) \qquad (B.18)$$

We shall define

$$\underline{f}^T=[-y(T-1)....-y(T-n_b)\ u(T-1)....u(T-n_a)] \qquad (B.19)$$

and

$$\underline{e}^T=[b_1....b_{nb}\ a_1....a_{na}] \qquad (B.20)$$

and y(T) as the predicted value of y at time T-1. The model is rewritten in terms of

the predicted value y(T) as

$$y(T)= \hat{y}(T) + e(T) \qquad (B.21)$$

since $C(z^{-1})=1$ then $\underline{f}(T)= \underline{g}(T)$ and (B.21) is

$$y(T)= \underline{\theta}^T\underline{g}(T) + e(T) \qquad (B.22)$$

The RLS method is a simple and easily applicable method. Its rate of convergence is very high [B.17]. There is only one real disadvantage, namely the assumption $H(z^{-1})=1$. If this does not hold in the system being identified, RLS will in general give a biased estimate. This drawback is the motivation for use of other methods [B.8].

B.4.2 The algorithm

The function J in Eq.(B.17) is minimised by the parameter $\underline{\theta}$ that obeys

$$[\sum_{t=1}^{N}\underline{g}(T)\underline{g}^T(T)]\ \underline{\theta}(N) = \sum_{t=1}^{N}\underline{g}(T)\ y(T)$$

if the matrix $\sum_{t=1}^{N}\underline{g}(T)\underline{g}^T(T)$ is nonsingular, the minimum is unique

and given by [B.1]:

$$\underline{\theta}(N)= [\sum_{t=1}^{N}\underline{g}(T)\underline{g}^T(T)]^{-1} \sum_{t=1}^{N}\underline{g}(T)\ y(T) \qquad (B.23)$$

This solution requires all the data recorded up to the Nth sample to be stored and calculated at once in a batch execution process. Every time a new sample is recorded the same process is repeated for the whole data.

In real-time applications this solution is not practical for two reasons. A lot of memory storage is required since all the sampled data has to be recorded. The second reason is that matrix inversion process takes a considerable time, this is a problem when fast sampling is required.

To avoid these problems a recursive solution is obtained in which the estimated value, $\underline{\theta}_N$, after N samples is a linear sum of the estimated value, $\underline{\theta}_{N-1}$, obtained after N-1 samples plus a corrective term based on the new information y_N and $\underline{q}_N$ received at the Nth sampling instant. This solution does not require matrix inversion, consequently the execution time is exessively reduced. Furthermore, it requires much less memory storage.

The least-squares estimate $\underline{\theta}$ satisfies the recursive equation [B.9]

$$\underline{\theta}(T) = \underline{\theta}(T-1) + \underline{k}(T)[y(T) - \underline{\theta}^T(T-1)\underline{q}(T)] \qquad (B.24)$$

where

$$\underline{k}(T) = P(T-1)\underline{q}(T)\ [1 + \underline{q}(T)^T P(T-1)\underline{q}(T)]^{-1} \qquad (B.25)$$

and

$$P(T) = [I - \underline{k}(T-1)\underline{q}^T(T)]\ P(T-1) \qquad (B.26)$$

where P(T) is the variance-covariance matrix of the identified parameters.

B.5 RECURSIVE EXTENDED LEAST SQUARES (RELS)

B.5.1 General

The recursive extended least squares was developed by Panuska (1968) and Young (1968) [B.23]. It can be considered as a straight forward extension of the RLS and also as a special case of the recursive maximum likelihood method as we shall see later. The general model Eq.(B.1) is assumed to have $H(z^{-1})=C(z^{-1})$, as as shown in Fig.B.2:

$$B(z^{-1})y(T) = A(z^{-1})u(T) + C(z^{-1})e(T) \qquad (B.26)$$

In RELS

$$\underline{\theta}^T = [b_1 \ldots b_{nb} \quad a_1 \ldots a_{na} \quad c_1 \ldots c_{nc}]$$

and

$$\underline{f}^T = [-y(T-1) \ldots u(T-1) \ldots e(T-1) \ldots]$$

Therefore,

$$e(T) = y(T) - \underline{f}^T \underline{\theta} \qquad (B.27)$$

Clearly this method is more general since the restrictive assumption $H(z^{-1})=1$ is no longer in play. Convergence requires some restrictions on the value of the $C(z^{-1})$ polynomial (its roots have to be within the unit circle) [B.4].

B.5.2 The algorithm

The cost function Eq.(B.17) is rewritten

$$J = 1/2 \sum_{T=1}^{N} e^2(T)$$

where

$$e(T) = y(T) - \underline{q}^T\underline{\theta}$$

and

$$\underline{q}(T) = \underline{f}(T)/C(z^{-1})$$

If a good estimate of $C(z^{-1})$ is available, it may be possible to use a fixed filter based on available a priori knowledge. The use of a fixed filter will then lead to an approximate form of the sequential prediction error algorithm [B.4]. Let $D(z^{-1})$ be a fixed a priori estimate of $C(z^{-1})$. In the case of RELS the algorithm is simplified even further by putting $D(z^{-1})=1$ [B.4]. Therefore:

$$\underline{q}(T) = \underline{f}(T)$$

The algorithm is then:

$$\underline{\theta}(T) = \underline{\theta}(T-1) + \underline{k}(T-1)\,[y(T) - \underline{q}^T\underline{\theta}(T-1)]$$

$$\underline{k}(T-1) = P(T-1)\underline{q}(T)\,[1 + \underline{q}^T(T)P(T-1)\underline{q}(T)]^{-1}$$

$$P(T) = [I - \underline{k}(T-1)\ \underline{g}^T(T)]\ P(T-1)$$

B.3 RECURSIVE MAXIMUM LIKELIHOOD (RML)

B.3.1 General

The recursive maximum likelihood method, originally developed by Soderstrom (1973), is based on an idea by Astrom [B.8]. The assumed model is the same as in RELS

$$B(z^{-1})y(T) = A(z^{-1})u(T) + C(z^{-1})e(T) \qquad (B.28)$$

or

$$y(T) = \underline{q}^T\underline{e} + e(T) \qquad (B.29)$$

where

$$\underline{e}^T = [\, b_1 \ldots b_{nb} \;\; a_1 \ldots a_{na} \;\; c_1 \ldots c_{nc}]$$

From Eq.(B.16)

$$\underline{q}^T = [-y(T-1)/C(z^{-1}) \ldots u(T-1)/C(z^{-1}) \ldots$$

$$e(T-1)/C(z^{-1}) \ldots \,] \qquad (B.30)$$

The RML is a superior method and has been shown to converge for all values of $C(z^{-1})$ [B.8]. The RML method is superficially the same as RELS except that the $\underline{q}(T)$ vector of RELS is filtered, where the filter used is $1/C(z^{-1})$. The RML algorithm needs more computation than RELS due to the extra filtering as shown in Table.B.1.

B.3.2 The algorithm

The cost function (B.17) is rewritten

$$J = 1/2 \sum_{T=1}^{N} e^2(T)$$

where

$$e(T) = y(T) - \underline{q}^T \underline{\theta}(T)$$

The algorithm is

$$\underline{\theta}(T) = \underline{\theta}(T-1) + \underline{k}(T)e(T)$$

$$\underline{k}(T) = P(T-1)\underline{q}(T)\ [1 + \underline{q}^T(T)P(T-1)\underline{q}(T)]^{-1}$$

$$P(T) = [\ I - \underline{k}(T)\underline{q}^T(T)]\ P(T-1)$$

B.4 RECURSIVE INSTRUMENTAL VARIABLE (RIV)

B.4.1 General

The recursive instrumental variable method is a modification of the least squares method and is designed to overcome the biased estimate problem of the RLS, if $H(z^{-1}) \neq 1$[B.10]. The model assumed is the same as that in RLS i.e.

$$B(z^{-1})y(T) = A(z^{-1})u(T) + e(T)$$

and

$$y(T) = \underline{q}^T \underline{e} + e(T) \qquad \text{(B.31)}$$

A disadvantage with the least square estimate is that in general $\underline{q}(T)$ and $e(T)$ will be found to be correlated but, as a condition for $\underline{e}(N)$ to converge to the true $\underline{e}$ in the RLS is that $\underline{q}(T)$ and $e(T)$ should not have any correlation, $\underline{e}(N)$ will not converge to the true $\underline{e}$ if this condition is not satisfied. To overcome this problem we might replace $\underline{q}(T)$ in (B.23) by a vector $\underline{x}(T)$, such that $\underline{x}(T)$ and $e(T)$ are uncorrelated and (B.23) becomes:

$$\underline{e}(T) = [\, \underline{x}(T)\underline{q}^T]^{-1}\, \underline{x}(T)\, y(T) \qquad \text{(B.32)}$$

and $\underline{e}(T)$ will tend to the true $\underline{e}$ as t tends to infinity under the following three conditions [B.7]:

$\underline{x}(T)$ and $e(T)$ are uncorrelated (B.33)

$e(T)$ has zero mean, (B.34)

the matrix $[\, \underline{x}(T)\underline{q}^T(T)]^{-1}$ is non-singular (B.35)

The estimate Eq.(B.32) is known as the Instrumental Variable (IV) estimate. The vector $\underline{x}(T)$ is referred to as the instrumental variable [B.10].

B.4.2 The algorithm

A common choice to satisfy the three conditions (B.33), (B.34), and (B.35) is [B.7]:

$$\underline{x}^T(T)= [-y_m(T-1) \ldots -y_m(T-n_b)\ u(T-1) \ldots u(T-n_a)] \qquad (B.36)$$

where $y_m(T)$ is the output of a deterministic system driven by the actual input $u(T)$ [B.7]:

$$y_m(T)+b_1y_m(T-1)+\ldots+b_{nb}y_m(T-n_b)= a_1u(T-1)+\ldots+ a_{na}u(T-n_a) \qquad (B.37)$$

For the recursive algorithm (B.39) a frequently used approach is to let b_i and a_i be time-dependent. Then the current estimates $b_i(T)$, $a_i(T)$ obtained from (B.39) can be used at time t in (B.37) [B.7]. Eq.(B.37) can be written as:

$$y_m(T)= \underline{\theta}^T\underline{x}(T) \qquad (B.38)$$

This approach was suggested by Mayne (1967), Wong and Polak (1967), and Young (1965) [B.7].

The algorithm is then

$$\underline{\theta}(T)= \underline{\theta}(T-1) + \underline{k}(T)\ [y(T) - \underline{a}^T(T)\underline{\theta}(T-1)] \qquad (B.39)$$

$$\underline{k}(T-1) = P(T-1)\underline{x}(T)\,[1 + \underline{q}^T P(T-1)\underline{x}(T)]^{-1} \qquad \text{(B.40)}$$

$$P(T) = [I - \underline{k}(T)\underline{q}^T(T)]\,P(T-1) \qquad \text{(B.41)}$$

B.5 STOCHASTIC APPROXIMATION (STA)

B.5.1 General

There are several recursive identification methods based on stochastic approximation, e.g. [B.11], [B.12], [B.13].

The model assumed is the same as that in RLS, i.e.

$$B(z^{-1})y(T) = A(z^{-1})u(T) + e(T)$$

or

$$y(T) = \underline{q}^T\underline{\theta} + e(T)$$

The methods considered above can be simplified to yield algorithms of this type by essentially substituting the matrix P(T) by a scalar p(T), e.g. c/T or $1/\mathrm{tr}P(T)^{-1}$ [B.7]. This means that the general description of the algorithms is transformed into

$$\underline{\theta}(T) = \underline{\theta}(T-1) + p(T)\underline{q}(T)e(T) \qquad (B.42)$$

This reduces the time of computation per iteration considerably, as shown in Table.B.1, and simplifies the procedure. However, in general the convergence will be slower than if the original algorithm is applied [B.8].

B.5.2 The algorithm

The cost function Eq.(B.17) is rewritten

$$J = 1/2 \sum_{T=1}^{N} e(T)^2$$

and

$$e(T) = y(T) - \underline{q}^T \underline{\theta}(T)$$

Robbins and Monro (1951) suggested the following recursive scheme as time evolves [B.7]:

$$\underline{\theta}(T) = \underline{\theta}(T-1) + p(T)\ \underline{q}(T)\ e(T) \qquad \text{(B.43)}$$

The sequence $p(T)$ in Eq.(B.43) is the gain sequence. Consideration of how to choose this sequence is given in [B.7].

B.6 INITIAL VALUES

In order to start all recursive algorithms, it is necessary to have initial values for $\underline{\theta}(0)$ and P(0). If no apriori information is available, then algorithm can be started with $\underline{\theta}(0)=0$ [B.9]. Since there is no confidence in this choice, the coveriance matrix P(0) should be large; an appropriate choice is P(0)= r.I where r is a big number e.g. 10^4 [B.14].

The choice of $\underline{\theta}(0)=0$ and P(0) = r.I is applicable to all methods. Note that the choice of the initial values have influence only on the transient behaviour of the methods.

B.7 REAL TIME VERSIONS

Real time identification allows time varying parameters to be tracked [B.8]. The ordinary algorithm gain ($\underline{k}$) reduces quickly when the P matrix gets small after a few iterations (typically 10 to 20) [B.7]. Therefore the algorithm cannot track any subsequent large change in the system parameters. The solution is to prevent the P matrix from getting too small. It is possible to modify the general algorithm to handle such systems. Essentially three kinds of extensions are used in the literature [B.8].

One way is to include a weighting factor or forgetting factor g. The modified algorithm is then:

$$\underline{\theta}(T) = \underline{\theta}(T-1) + \underline{k}(T)\, e(T) \qquad (B.44)$$

$$\underline{k}(T) = P(T)\underline{q}[\, g + \underline{q}^T P(T)\underline{q}\,]^{-1} \qquad (B.45)$$

$$P(T) = [\, I - \underline{k}(T-1)\underline{q}^T(T)]\, P(T-1)/g \qquad (B.46)$$

By choosing g smaller than 1 (e.g. as 0.99) old residuals will have small influence on the estimates [B.8,B.15]. This approach is described in [B.16]. An adaptive control algorithm with variable forgetting factor was implemented by Fortescue, Kershanbaum and Ydstie [B.15].

Another approach is to reset the covariance matrix P at various times. In this case, the obvious time to reset P is when one suspects that a significant parameter change has occurred [B.8]. Therefore:

$$P(0) = h.I, \text{ where } h > 0$$

Let $Z_s = [T_1 \quad T_2 \quad T_3 \quad \ldots]$ be the times at which resetting occurs, otherwise for $T \in [Z_s]$ an ordinary algorithm is used, that is:

$$P(T) = [\, I - \underline{k}(T-1)\underline{a}^T(T)]\, P(T-1) \qquad (B.47)$$

otherwise, for $T = T_i \in [Z_s]$, $P(T)$ is reset as follows:

$$P(T_i) = K_i.I \qquad (B.48)$$

where $0 < K_{min} < K_i < K_{max} < \infty$

The third approach is to make use of the interpretation of the RLS method as a Kalman filter [B.8]. Inclusion of process noise in the model will lead to the algorithm:

$$\underline{\theta}(T) = \underline{\theta}(T-1) + \underline{k}(T)\, e(T) \qquad (B.49)$$

$$\underline{k}(T) = P(T)\underline{q}[\ 1 + \underline{q}^T P(T)\underline{q}\ ]^{-1} \qquad (B.50)$$

$$P(T) = [\ I - \underline{k}(T-1)\underline{q}^T(T)]\ P(T-1) + R \qquad (B.51)$$

where R is a positive semi definite matrix [B.7].

The extensions above have the desired property that the gain vector $\underline{k}(T)$ does not tend to zero.

B.8 IMPROVEMENT OF THE CONVERGENCE RATE

In [B.17] several ways to improve the convergence rate were discussed for the RML method. A very common choice is to use the algorithm [B.8]:

$$\underline{\theta}(T) = \underline{\theta}(T-1) + \underline{k}(T-1)\, e(T) \qquad (B.52)$$

$$\underline{k}(T) = P(T-1)\underline{q}[\ g(T) + \underline{q}^T P(T-1)\underline{q}\ ]^{-1} \qquad (B.53)$$

$$P(T) = [\ I - \underline{k}(T-1)\underline{q}^T\ ]\ P(T-1)/g(T) \qquad (B.54)$$

where

$$g(T) = g_0\, g(T-1) + (\ 1 - g_0) \qquad (B.55)$$

The number g_0 is chosen close to 1. e.g. as 0.99, and so is the initial value $g(0)$ [B.8].

The weighting factor g(T) is constructed to tend to 1 as time goes on.

B.9 COMPARISONS

These methods have been examined and compared in [B.5], [B.6], [B.7], [B.8], [B.20], [B.21], and [B.22]. The following results are obtained:

B.9.1 Overall comparison of the performance

For long identification time the parameter estimation methods with unbiased estimates are (RIV, RELS, RML, STA) and they have approximately the same accuracy of the input/output model. For short identification all methods, including RLS show little difference. In general the estimates of the process parameters converge quicker than the estimates for the noise parameters.

B.9.2 Properties of each method

Table.B.1 shows a comparison of the methods in terms of computation time and rate of convergence.

(a) Recursive Least Squares (RLS)

Biased estimates for coloured noise. Applicable for short identification time if noise acts on the process. Relatively small computational expense. Good starting method for RIV or RML. RLS reliable convergence for white noise.

(b) Recursive Extended Least Squares (RELS)

Similar to RML but it doesn't converge for some type of systems (i.e. if the roots of the $C(z^{-1})$ polynomial are outside the unit circle).

(c) Recursive Maximum Likelihood (RML)

Good performance for special noise model C/B. Large computational expense due to the filtering of $\underline{q}(T)$. The convergence is more reliable than RELS.

(d) Recursive Instrumental Variable(RIV)

Good performance for wide range of noise models. Medium computational expense. A priori factors: 1 matrix, filter factors. Reliable convergence. It is recommended to start with RLS first and then continue with RIV.

B.9.3 Conclusion

(a) - The RLS method gives biased estimates for noisy systems.

(b) - The RIV, RELS and RML methods often give good results. Notice, however, that the RIV method sometimes produces unacceptable estimates.

(c) - In general the RML method is the most accurate method.

(d) - The RML method seems to be superior to the RELS method, in particular concerning the estimation of the parameters of C.

(e) - For adaptive control RLS is often adequate.

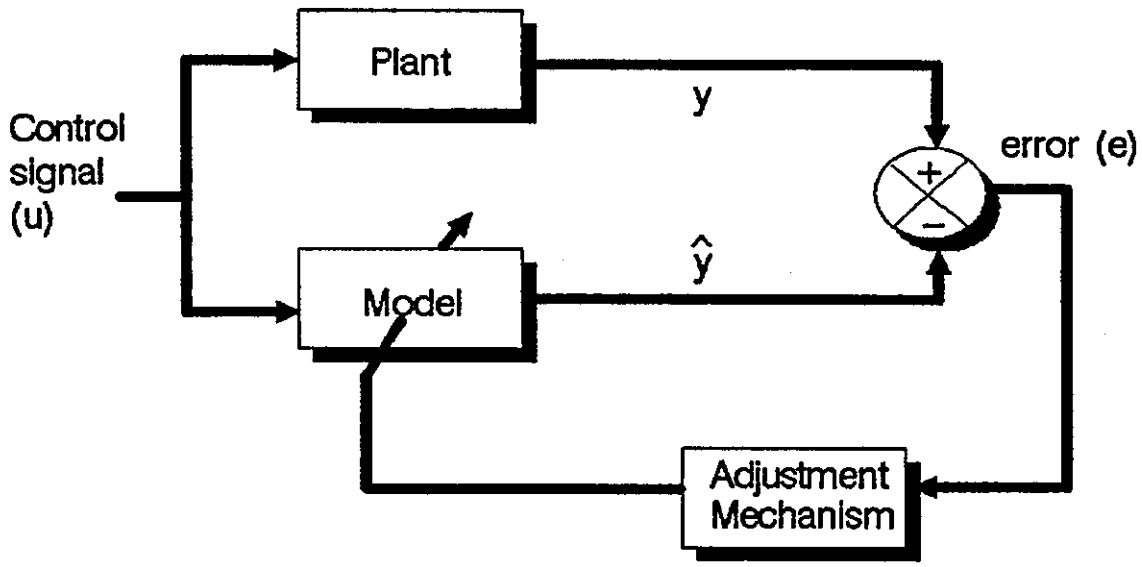


Fig . B . 1
On – line System Identification Process

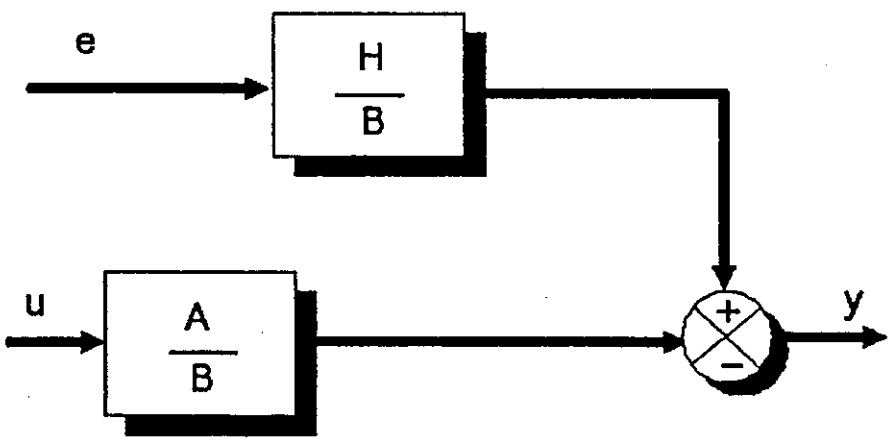


Fig . B . 2
General Model Form

Method	Computation Time	Convergance Rate
RLS	Small	Fast
RELS	Medium	Fast
RML	Large	Medium
RIV	Large	Medium
STA	Very small	Very slow

Table . B . 1

Specifications of the methods

APPENDIX - C

C - CONTROLLER DESIGN TECHNIQUES

C.1 - ONE-STEP-AHEAD CONTROL

The one-step-ahead controller minimises a cost function I which is defined as:

$$I = [\, y(T+1) - w(T)]^2 \qquad (C.1)$$

where $y(T+1)$ is the output at time T+1 and $w(T)$ is the desired value.

The transfer function (4.4) is rewritten here:

$$G(z) = \frac{a_1 z^{-1}}{1 + b_1 z^{-1}} \qquad (C.2)$$

where $a_1 = kh/\tau$ and $b_1 = -1$

The discrete time domain equivalent of Eq.(C.2) is:

$$y(T) = -b_1\, y(T-1) + a_1\, u(T-1) \qquad (C.3)$$

To minimise I in (C.1) $\dfrac{\partial I}{\partial u(T)}$ is equal to zero. Thus

$$\partial I / \partial u(T) = 2\, [\, y(T+1) - w(T)]\, a_1 = 0 \qquad (C.4)$$

and substituting for $y(T+1)$ from Eq.(C.3) in Eq.(C.4) we have

$$-b_1\, y(T) + a_1\, u(T) - w(T) = 0$$

The control signal is then

$$u(T) = \frac{1}{a_1} w(T) + \frac{b_1}{a_1} y(T) \qquad (C.5)$$

Eq.(C.5) can be written in the general form of Eq.(4.5) as:

$$u(T) = f_0 w(T) - s_0 y(T)$$

where $f_0 = 1/a_1$ and $s_0 = -b_1/a_1$. Since $b_1 = -1$ then $f_0 = s_0$ and the controller is simply a pure gain:

$$u(T) = K [w(T) - y(T)] \qquad (C.6)$$

where $K = 1/a_1 = \gamma/kh$.

C.2 - WEIGHTED ONE-STEP-AHEAD CONTROL

The cost function I is defined as:

$$I = [\, y(T+1) - w(T)]^2 + \lambda\, u^2(T) \qquad (C.7)$$

where the value of λ determines the compromise between bringing y to w and the amount of effort expended.

To minimise I in (C.7), set $\partial I / \partial u(T) = 0$

$$\partial I / \partial u(T) = 2\, [\, y(T+1) - w(T)]\, a_1 + 2\lambda\, u(T) = 0 \qquad (C.8)$$

substituting for y(T+1) from Eq.(C.3) in Eq.(C.8) then:

$$[-b_1\, y(T) + a_1\, u(T) - w(T)]\, a_1 + \lambda\, u(T) = 0$$

$$u(T)[\, a_1 + \lambda / a_1\,] = w(T) + b_1\, y(T)$$

and

$$u(T) = \frac{1}{a_1 + \lambda / a_1}\, w(T) + \frac{b_1}{a_1 + \lambda / a_1}\, y(T) \qquad (C.9)$$

Eq.(C.9) can be written in the general form:

$$R(z)\, u(T) = F(z)\, w(T) - S(z)\, y(T)$$

and in this R, F, and S for our example of polynomials of degree zero, then

$$u(T) = f_0\, w(T) - s_0\, y(T) \qquad (C.10)$$

where $f_0 = 1/[a_1 + \lambda/a_1]$ and $s_0 = -b_1/[a_1 + \lambda/a_1]$

C.3 - POLE/ZERO CANCELLATION

C.3.1 - General

The design considers a servo problem which is expressed in terms of a model that gives the desired response to command signals [C.12]. Fig.C.1 and Fig.C.2 show block diagrams of the system and the desired model respectively.

The desired closed-loop pulse-transfer is given by

$$G_m = \frac{A_m}{B_m} \qquad (C.11)$$

where A_m and B_m do not have any common factors. In general it is not sufficient to specify G_m [C.2]. With output feedback, the command signal does not excite all the states of the system; therefore the states that are not measured have to be observed. This is done by introducing a polynomial B_0 called an observer [C.2].

The plant model(4.4) is written as:

$$B(z)y(T) = A(z)u(T) \qquad (C.12)$$

where u is the control signal and y is the measured output signal of the plant. The problem is to find a control law of the form (4.5) such that the closed-loop system has the input-output relation given by the pulse-transfer function of (C.11).

The input-output relationship for the closed-loop system is obtained by eliminating u between Eq.(4.5) and Eq.(C.12), then

$$(B R + A S)y = A F w \qquad (C.13)$$

Comparing Eq.(C.11) and Eq.(C.13) gives:

$$\frac{A\,F}{B\,R + A\,S} = \frac{A_m}{B_m} \qquad (C.14)$$

The problem now is to find the polynomials R, S, and F that satisfy Eq.(C.14).

The zeros of the closed-loop system are the zeros of the polynomials A and F. First consider the open-loop zeros of the plant, i.e. the zeros of the polynomial A. If a factor of A is not a factor of A_m then it must be a factor of BR+AS, so it must be cancelled by a closed-loop pole. Since the closed loop system must be stable, only zeros which lie in the complex domain unit circle stability region (defined here as "stable zeros") may be cancelled. We can factor A as:

$$A = A^+ A^-$$

where A^+ represents all zeros inside the stability region and A^- represents all zeros outside the stability region. To get a unique factorisation the coefficient of the highest power in A^+ is fixed to unity [C.12]. The polynomial A^+ is said to be monic [C.2].

Since A^- cannot be a factor of BR+AS, it follows that it must divide A_m, i.e.,

$$A_m = A^- A_m' \qquad (C.15)$$

This implies that unstable plant zeros cannot be changed, but must be included in A_m. If A^+ (stable zeros) is a factor of BR+AS, it follows that it is also a factor of R. Hence

$$R = A^+ R' \qquad (C.16)$$

Now we can write Eq.(C.14) as:

$$\frac{A^+ A^- F}{B A^+ R' + A^- A^+ S} = \frac{A^- A_m'}{B_m}$$

$$\Rightarrow \frac{A^+ A^- F}{A^+(B R' + A^- S)} = \frac{A^- A_m'}{B_m} \qquad \text{(C.17)}$$

Clearly Eq.(C.17) reduces to

$$\Rightarrow \frac{F}{B R' + A^- S} = \frac{A_m'}{B_m} \qquad \text{(C.18)}$$

and this shows that B_m is a factor of $BR'+A^-S$. Furthermore, as discussed in [C.2] the observer polynomial (B_o) will not appear in the transfer function, which relates the output to the command signal. It follows that the observer polynomial is cancelled in the transfer function from the reference signal to the output. Therefore B_o is a factor of $BR+AS$ and F and the following conditions are obtained:

$$F = A_m' B_o \qquad \text{(C.19)}$$

and

$$B R' + A^- S = B_o B_m \qquad \text{(C.20)}$$

Equation (C.20) must therefore be satisfied by R' and S to achieve the required closed-loop plant model.

C.3.2 - The design of the plant controller

The transfer function (4.4) is rewritten in the forward shift form:

$$G(z) = \frac{a_1}{z + b_1} \qquad \text{(C.21)}$$

where $a_1 = kh/\tau$ and $b_1 = -1$

We shall consider the desired closed-loop behaviour to be given by the classical continuous time second order system:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

since such system has known rise time and overshoot as explicit functions of ζ and ω_n.

The equivalent discrete form of the continuous transfer function, G(s), is determined using the Bilinear z-transform [C.1]. The poles of the corresponding discrete time system are given by the characteristic equation:

$$z^2 + p_1 z + p_2 = 0 \qquad \text{(C.22)}$$

where

$$p_1 = \frac{2T^2\omega_n^2 - 8}{4 + 4T\zeta\omega_n + T^2\omega_n^2} \qquad \text{(C.23)}$$

$$p_2 = \frac{4 - 4T\zeta\omega_n + T^2\omega_n^2}{4 + 4T\zeta\omega_n + T^2\omega_n^2} \qquad \text{(C.24)}$$

Fig.C.3 shows the definition of the rise time T_r, settling time T_s, and overshoot M_p. The values of and are calculated according to the specified rise time, T_r, and overshoot, M_p, using the following relations [C.6,C.7]:

$$T_r = \frac{\exp(\phi/\tan \phi)}{\omega_n} \qquad (C.25)$$

and

$$\omega = \frac{\exp(\phi/\tan \phi)}{T_r} \qquad (C.26)$$

where $\cos \phi = \zeta$.

The overshoot is

$$M_p = \exp(- \pi\zeta/ \sqrt{1 - \zeta^2}) \qquad (C.27)$$

and

$$\zeta = \ln M_p / \sqrt{\pi^2 + (\ln M_p)^2} \qquad (C.28)$$

the settling time T_s is

$$T_s = \ln x / \omega\zeta \qquad (C.29)$$

where x is the tolerance of the output response to the steady state value.

The desired closed-loop system is characterised by the pulse-transfer function:

$$G_m = \frac{A_m}{B_m} = \frac{P(1)\ A^-(z)}{A^-(1)\ P(z)} \qquad (C.30)$$

where $A^-(z)$ are the unstable or poorly damped zeros of the plant. $P(1)$ and $A^-(1)$ ensure zero steady state error [C.2].

The pulse transfer function (C.21) has no zero. In this case no zero is cancelled and $A(z)$ is factorised as:

$$A^+ = 1$$

$$A^- = a_1$$

The desired pulse-transfer function of the system is then:

$$G_m = \frac{1 + p_1 + p_2}{z^2 + p_1 z + p_2} \qquad (C.31)$$

$$A'_m = \frac{A_m}{A^-} = \frac{1 + p_1 + p_2}{a_1} \qquad (C.32)$$

It is proved in [C.2] that for the pole-placement design to have a causal solution then,

the degree of $B_0 \geq 2 \deg B - \deg B_m - \deg A^+ - 1 = -1$

Thus with $\deg B_0 = 0$ and choosing $B_0 = 1$

$$\deg S = \deg B - 1 = 0$$

and

$$F = A'_m B_0 = (1 + p_1 + p_2)/a_1 = f_0 \qquad (C.33)$$

$$\deg F = 0$$

$$\deg R' = \deg B_0 + \deg B_m - \deg B = 1$$

then $R' = z + r_1$ [C.2].

Substituting for F, R and S in the design Eq.(C.12)

$$(z - 1)(z + r_1) + a_1 s_0 = z^2 + p_1 z + p_2 \qquad (C.34)$$

which can be solved for the unknown coefficients. Let $z = 1$ in (C.34) then:

$$a_1 s_0 = 1 + p_1 + p_2$$

and

$$s_0 = \frac{1 + p_1 + p_2}{a_1}$$

equating equal powers of z in Eq.(C.34) to get:

$$(r_1 - 1) = p_1$$

and

$$r_1 = 1 + p_1$$

Substituting for these values in the general control Eq.(4.5):

$$(z + r_1) u(T) = f_0 w(T) - s_0 y(T)$$

then

$$u(T) = f_0 w(T-1) - s_0 y(T-1) - r_1 u(T-1) \qquad (C.35)$$

C.3.3 - The design algorithm

The following data must be provided in the design of pole/zero cancellation: a plant model specified as a pulse-transfer function A/B, an observer polynomial B_0, the desired closed-loop pulse-transfer function A_m/B_m, and the stability region.

The degree of the observer polynomial is [C.2]

$$\deg B_0 \geq 2 \deg B - \deg B_m - \deg A^+ - 1$$

Step 1. Factor A and A_m as

$$A = A^- A^+ , \qquad A_m = A^- A_m'$$

where A^+ is monic and has all its zeros inside the stability region, and A^- has all its zeros outside the stability region.

Step 2. Solve the equation

$$B R' + A^- S = B_0 B_m$$

for the polynomials R' and S. The solution should satisfy the conditions:

$$\deg R' = \deg B_0 + \deg B_m - \deg B$$
$$\deg S = \deg B - 1$$

To solve the equation introduce polynomials R' and S with unknown coefficients and given order and equate equal powers of z to determine the

coefficients of R' and S.

Step 3. Substitute in the control law

$$R\ u(t) = F\ w(t) - S\ y(t)$$

where $R = A^{+}\ R'$, $F = A'_m\ B_0$

[C.2].

C.4 - PID CONTROLLER

The idealised equation of a continuous time PID controller is [C.13]:

$$u(t)= K \left[e(t) + \frac{1}{T_I} \int_0^t e(t)\, dt + T_D \frac{de(t)}{dt} \right] \qquad (C.36)$$

where the parameters

K = proportional gain

T_I = integration time

T_D = derivative time

For a given sample time T, Eq.(C.36) can be turned into a difference equation by discretisation. The derivative is simply replaced by a first order difference and the integral by a sum [C.14]. Applying backward rectangular integration gives [C.13]:

$$u(k)= K \left[e(k) + \frac{T}{T_I} \sum_{i=0}^{k-1} e(i) + \frac{T_D}{T} (e(k)- e(k-1)) \right] \qquad (C.37)$$

This is a non-recursive control algorithm, which means that all past error values e(k) have to be stored. However a recursive algorithm is more suitable for programming on computers since past error values do not have to be stored. This algorithm is characterised by the calculation of the current control signal u(k) based on the previous value u(k-1) and correction terms [C.13]. The recursive algorithm is derived by subtracting from Eq.(C.37)

$$u(k-1)= K[\ e(k-1) + \frac{T}{T_I} \sum_{i=0}^{k-2} e(i-1) + \frac{T_D}{T}(e(k-1)- e(k-2))] \qquad (C.38)$$

and one obtains

$$u(k)- u(k-1) = q_0\, e(k) + q_1\, e(k-1) + q_2\, e(k-2) \qquad (C.39)$$

with parameters

$$q_0 = K(1 + \frac{T_D}{T}) \qquad (C.40)$$

$$q_1 = -K(1 + 2\frac{T_D}{T} - \frac{T}{T_I}) \qquad (C.41)$$

$$q_2 = K \frac{T_D}{T} \qquad (C.42)$$

Fig.C.4 shows the recursive algorithm of the digital PID controller. Fig.C.5 shows the control loop of the controller. The Z-transfer function of the controller is

$$G_c(z)= \frac{u(k)}{e(k)} = \frac{Q}{J} = \frac{q_0 + q_1\, z^{-1} + q_2\, z^{-2}}{1 - z^{-1}} \qquad (C.43)$$

$$(1 - z^{-1})u(k) = (q_0 + q_1\, z^{-1} + q_2\, z^{-2})[w(k)- y(k)] \qquad (C.44)$$

The following characteristic coefficients can be defined:

$K = q_0 - q_2$	gain	(C.45)
$C_D = q_2/ K$	lead coefficient	(C.46)
$C_I = (q_0 + q_1 + q_2)/ K$	integration coeff	(C.47)

They have been defined so that for small sample times they are related to the parameters of the continuous PID controller as follows:

$$K = K \; ; \quad C_D = T_D/T \; ; \quad C_I = T/T_I$$

C_D is the ratio of the lead time to sample time and C_I the ratio of sample time to integration time. If these characteristic coefficients are substituted in Eq.(C.43) the Z-transfer function becomes

$$G_c(z) = \frac{K\,[\,(1+C_D) + (C_I - 2C_D)z^{-1} + C_D\,z^{-2}\,]}{1 - z^{-1}} \qquad (C.48)$$

If $C_D = 0$ in Eq.(C.46), then $q_2 = 0$ and Eq.(C.43) becomes

$$G_c(z) = \frac{q_0 + q_1\,z^{-1}}{1 - z^{-1}} \qquad (C.49)$$

and the difference equation is

$$u(k) = u(k-1) + q_0\,e(k) + q_1\,e(k-1) \qquad (C.50)$$

and this is equivalent to a PI controller.

From Fig.C.5 the closed loop equation is found to be:

$$y(T) = \frac{Q\,A}{B\,J + Q\,A}\; w(T) \qquad (C.51)$$

The general control input form shown in Fig.C.6 can be compared with Fig.C.7, the PID control input function given by Eq.(C.44). The control input of Eq.(C.44) can be associated with the general controller form Eq.(4.5) if

$$F/R = S/R \text{ , and } F = S$$

then

$$R = J = 1 - z^{-1} \text{ and } \quad F = S = Q = q_0 + q_1 z^{-1} + q_2 z^{-2} \qquad (C.52)$$

The coefficients of the Q polynomial q_0, q_1 and q_2 must be selected to meet the desired performance objective, hence the K, T_I and T_D coefficients will be chosen. The pole assignment criterion is used, in which the denominator of the closed loop Eq.(C.51) is equated to P, an a priori selected polynomial. The equation required to meet the pole assignment objective is therefore:

$$B\ J + Q\ A = P \qquad (C.53)$$

from which the coefficients of the Q polynomial are found.

The apriori defined polynomial P corresponds to the continuous time polynomial

$$s^2 + 2\zeta\omega_n\ s + \ \omega_n^2 \qquad (C.54)$$

and, as shown in Eq.(C.22), using the backward shift operator is:

$$1 + p_1\ z^{-1} + p_2\ z^{-2} \qquad (C.55)$$

where

$$p_1 = \frac{2T^2\omega_n^2 - 8}{4 + 4T\zeta\omega_n + T^2\omega_n^2} \qquad (C.56)$$

$$p_2 = \frac{4 - 4T\zeta\omega_n + T^2\omega_n^2}{4 + 4T\zeta\omega_n + T^2\omega_n^2} \qquad (C.57)$$

The values of ω_n and ζ are calculated according to the specified rise time and overshoot as discussed in section (C.3).

Substituting for A, B, J, Q and P from Eq.(C.2),Eq.(C.52) and Eq.(C.55) respectively in the design Eq.(C.53) we have

$$(1-z^{-1})(1+b_1)+a_1z^{-1}(q_0+q_1z^{-1}+q_2z^{-2})= 1+p_1z^{-1}+p_2z^{-2} \qquad (C.58)$$

Equating equal powers of z gives

$$q_0 = (2 + p_1)/a_1 \qquad (C.59)$$

$$q_1 = (p_2 - 1)/a_1 \qquad (C.60)$$

$$q_2 = 0 \qquad (C.61)$$

and this is equivalent to a PI controller. The control signal function is then

$$u(t)= q_0 [w(t)- y(t)] + q_1 [w(t-1)- y(t-1)] + u(t-1) \qquad (C.62)$$

C.5 - EXECUTION TIMES

Fig.C.8 shows the hardware used to measure the execution time for each algorithm.

The computational time required to execute each of the following algorithms:

- One Step Ahead controller (OSA)
- Weighted One Step Ahead controller (WOSA)
- Pole/Zero Assignment controller (PZA)
- PID controller

is as follows:

OSA	-- 2.475	mSec.
WOSA	-- 3.05	mSec.
PZA	-- 4.35	mSec.
PID	-- 4.40	mSec.

Table.C.1

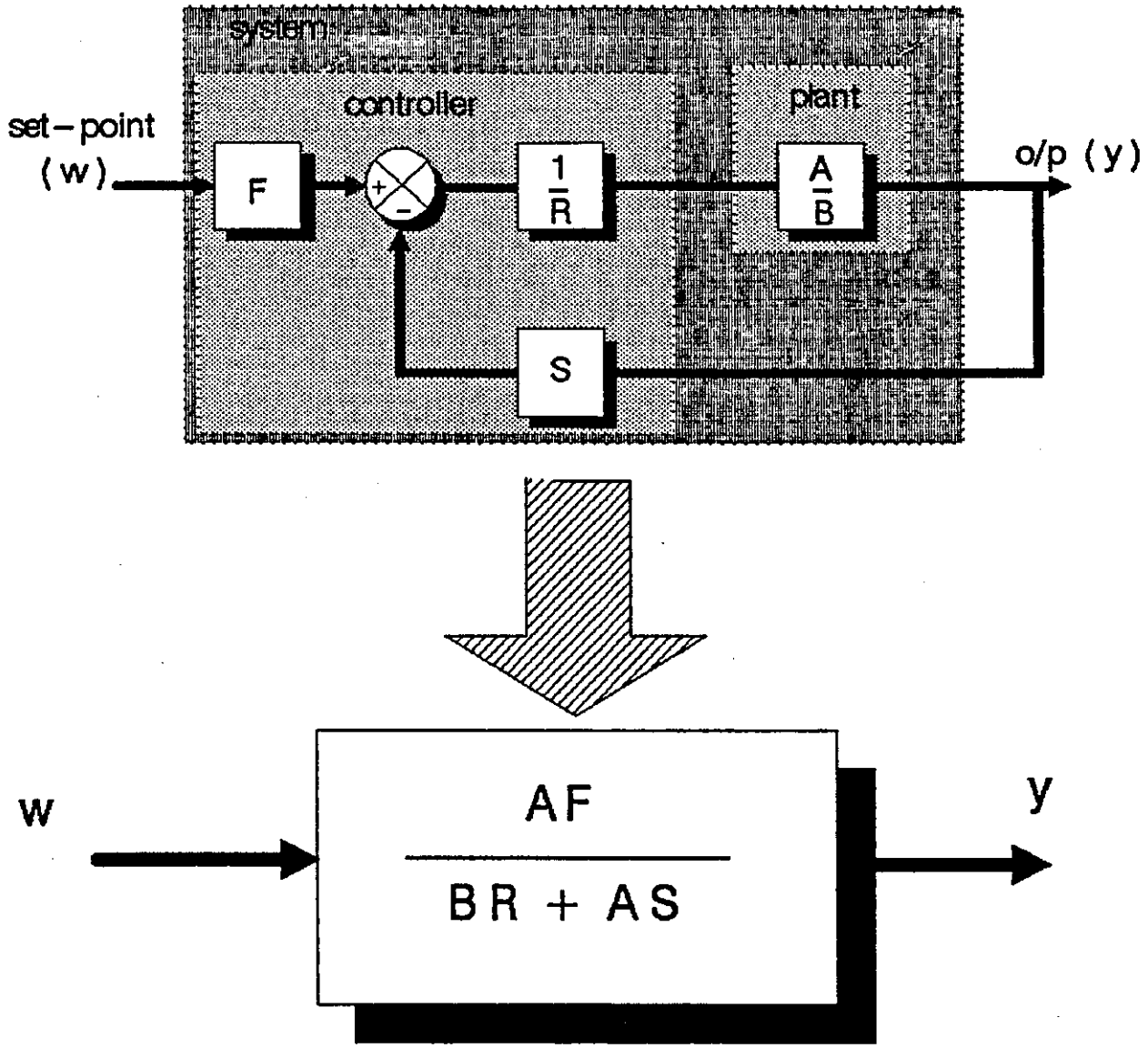


Fig . C . 1
System closed loop transfer function

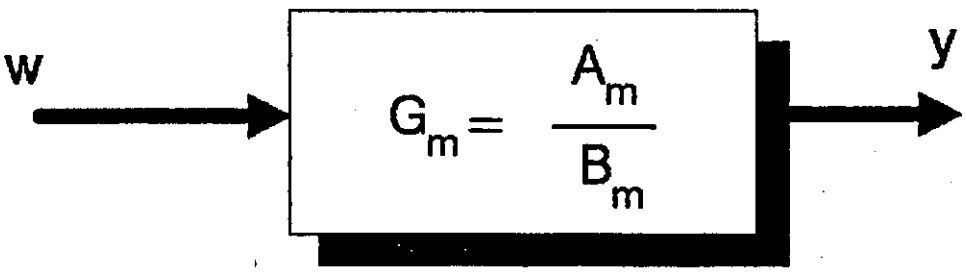


Fig . C . 2
Generalised closed loop transfer function

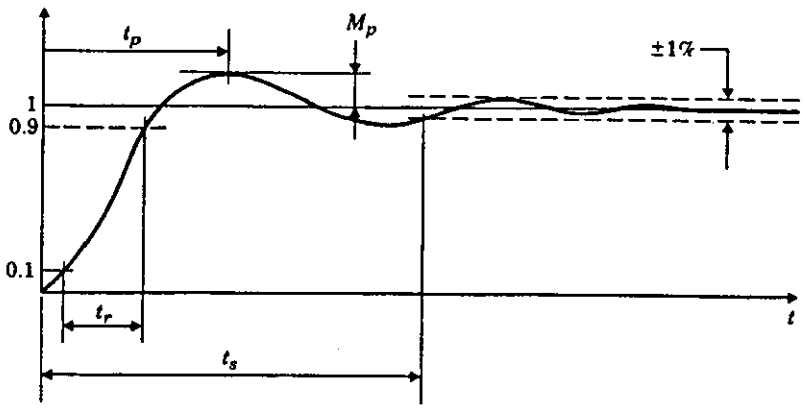


Fig .C.3

Definition of rise time t_r , settling time t_s , and overshoot M_p

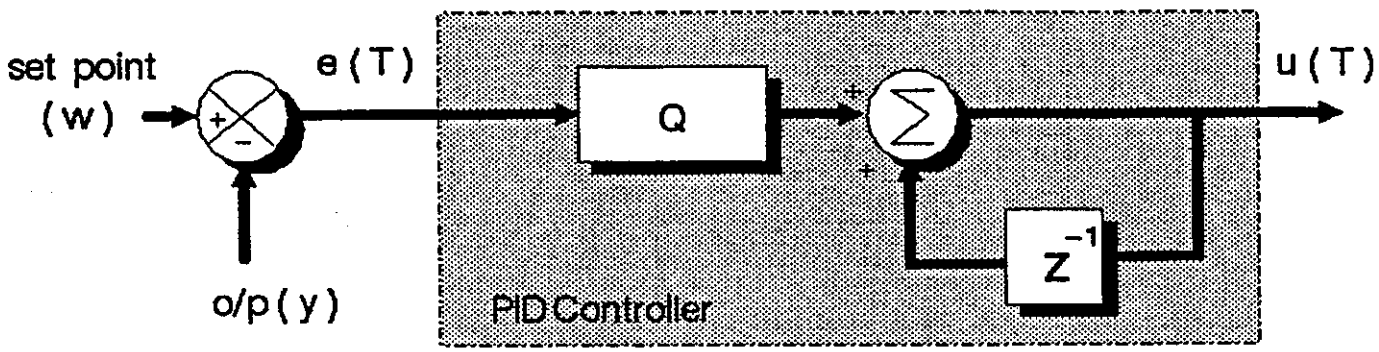


Fig . C . 4

Digital PID Controller – Recursive Algorithm Implementation

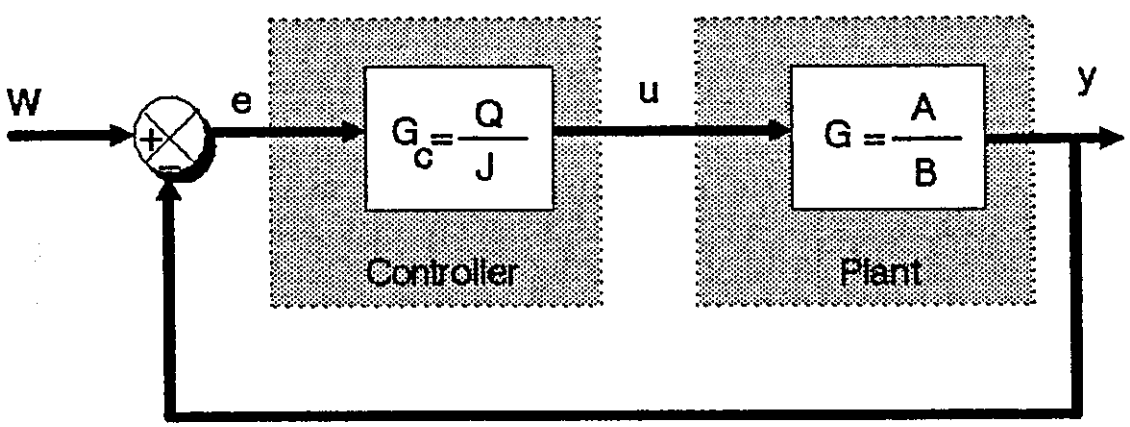


Fig . C . 5

Complete Control Loop

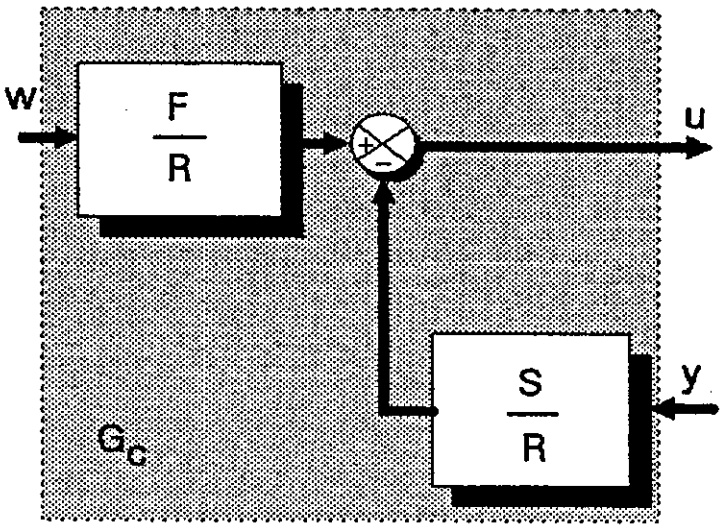


Fig . C . 6

The general controller

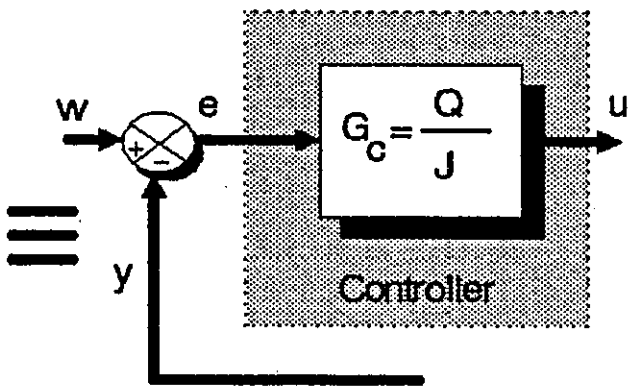
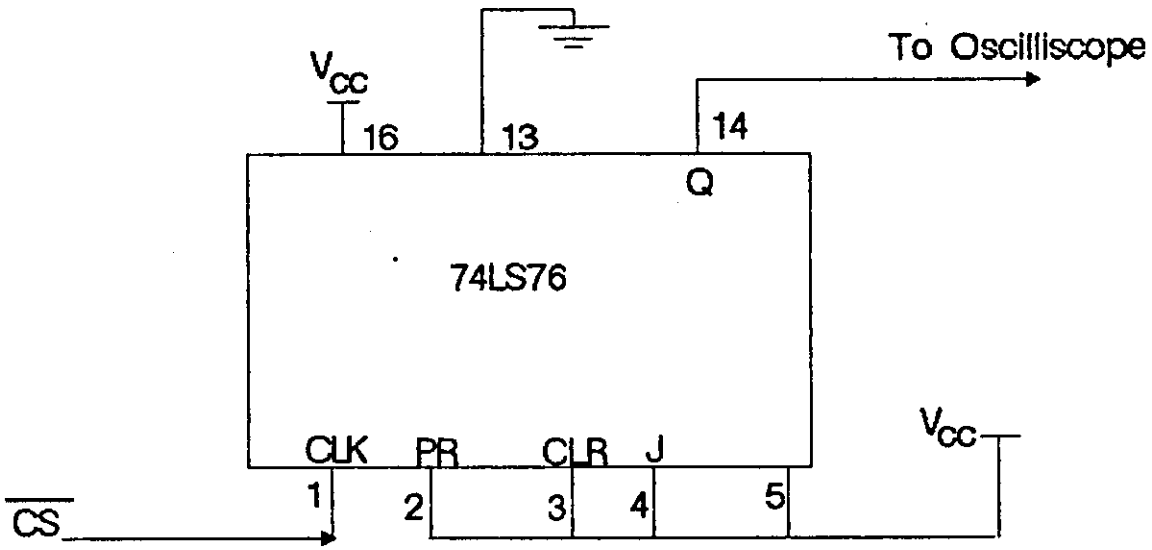


Fig . C . 7

The PID controller



Flg . C . 8

Toggle Circuit

APPENDIX-D

D SELF TUNING CONTROLLER - HARDWARE

D.1 TARGET BOARD SYSTEM DESCRIPTION

D.1.1 Overall structure

Fig.D.1 shows the overall structure of the target system controller, this being implemented on two printed circuit boards. It is based on the use of a general purpose single board computer (SBC), augmented by an analogue I/O section mounted on a piggy-back board.

(a) Central processing unit

The Central Processing Unit (CPU) is an Intel 8088 (5-MHz version). The system operates in maximum mode at a clock frequency of 4 MHz.

(b) Numerical data processor

The Numerical Data Processor (NDP) is the Intel 8087 Co.Processor which works in parallel with the system processor to execute arithmetic operations.

(c) Bus controller

The Intel 8288 Bus Controller is incorporated since the CPU has to work in maximum mode.

(d) Analogue to digital converter

A 12-bit successive approximation type Analogue to Digital Converter (ADC) is used to digitise plant (and other) analogue signals.

(e) Digital to analogue converter

A 12-bit Digital to Analogue Converter (DAC) is used to handle the digital to analogue conversion for the output control signal.

(f) Memory

Three sockets are provided, organised in BYTE-WIDE fashion (JEDEC standard). Each socket is capable of holding either:

* 2764 Eprom (8K x 8 Bits)
* 27128 Eprom (16K x 8 Bits)
* 4802 Ram (2K x 8 Bits)
* 2016P Ram (8K x 8 Bits)

The system has a 64 kByte address range.

(g) Timers

A single programmable timer I.C. is included in the design, housing three separate timers. One is used as the serial communications baud rate generator (see section D.1.h), enabling software selection of the baud rate. Two timers are available for control loop timing.

(h) Interrupts

A multi-level interrupt structure is implemented using a programmable interrupt controller device.

(i) Serial communications

A serial data communications interface is incorporated for use with a console. Full duplex signal transmission and programmable baud rate (See Sec.D.1.g) facilities are included, the signal levels complying with EIA RS-232C standards.

(j) Backplane bus

This consists of address, data and control buses, being compatible with the IEEE P1000 bus standard. Connections are made via a 64 way (DIN 41612C) two-part connector.

(k) Power supplies

Three d.c. voltages are required by the system. A 5 volt line feeds the digital circuits whilst ±15 volt lines power the analogue sub-systems.

(l) Circuit board

This is Extended double Eurocard format.

D.1.2 System design - Details

Fig.D.2 shows a block diagram of the target system.

(a) System bus

(i) Bus signals

The system bus consists of 16 address and 8 data lines plus the following control lines:

* MRDC/ (Read)
* MWTC/ (Write)
* READY
* RESET
* CLK (Clock)
* PCLK (Peripheral Clock)
* REQUEST/,GRANT 0/
* REQUEST/,GRANT 1/
* DEN/(Data Bus Enable)
* DT/R (Data Transmit/Receive)
* ALE (Address Latch Enable)

Note: A signal that is active low is designated by a slash(/) following its title. The address and data buses are fully buffered.

(ii) MRDC/ and MWTC/ Lines

These signals are issued by the Bus Controller to addressed memory or

peripheral devices commanding them to perform a data write (MWTC/) or data read (MRDC/) function.

(iii) RESET and READY lines

* RESET is a signal generated by the Clock generator. It is applied to the CPU and peripheral devices as a general hardware reset command.

* READY is a CPU input signal which is generated by on-board devices, the purpose being to force the processor into a wait-state condition. This allows a "slow" device to lengthen the read and write cycles. This line is sampled by the processor during each read and write cycle; if it is "not ready" (low) the CPU holds all its control lines in their current state, continuing only when ready returns high.

(iv) Clock and peripheral clock

The clock (CLK) signal is used as the CPU's timing reference. It is a square wave signal having a 33% duty cycle.

The peripheral clock (PCLK) is a square wave at half the CLK frequency with a 50% duty cycle. It is used as the timing reference for the peripheral devices.

(v) REQUEST/,GRANT/

Request-grant operations are provided to support multiprocessor operation. "REQUEST" is an input signal generated by another processor in the system which requires use of the system bus. "GRANT" is the response by the receiving processor, this being sent back on the same line as REQUEST. Fig.D.3

shows the Request/Grant timing diagram. On receiving a request the CPU completes its current instruction and then sends a grant pulse to the requesting processor on the same line. At the same time as issuing GRANT the processor releases control of the system bus (i.e. "floats" it). The second processor takes control of the system on receiving the GRANT/ signal.

(vi) DEN/, DT/R, and ALE

The 8088 uses a multiplexed address and data bus structure; the lower 8 bits of the address bus being shared with the 8 bit data bus.

At the beginning of each cycle the required address is presented on the address pins of the CPU. This is then latched using external latches with ALE as the control signal (generated by the Bus Controller). The lower 8 bits of the CPU's address bus now become the bi-directional data bus. In most cases this requires buffering so that it can drive the system data bus; hence the Bus Controller generates the required buffer chip control signals, data enable (DEN) and data transmit/receive (DT/R).

(b) System timing

All system timing is related to the system clock as shown in Fig.D.4. Each processor bus cycle consists of at least 4 CLK cycles, referred to as Tl, T2, T3, and T4. If the READY line is low (not ready) at the end of T2 or T3, extra CLK cycles called wait states (Tw) are inserted between T3 and T4 [D.1].

(c) Interrupt operation

Interrupts are used extensively in the system to allow devices to work independently to the CPU, but be able to request service from it via an interrupt. Interrupts can be either software or hardware initiated.

The 8088 processor can support 256 interrupt vectors. A 256-entry vector table, which contains address pointers to the interrupt routines, reside in absolute locations 0 through 3FFH. As shown in Fig.D.5, the first five interrupt vectors are associated with the software-initiated interrupts and the hardware non-maskable interrupt (NMI). The next 27 interrupt vectors are reserved by Intel. The remaining interrupt vectors (vectors 32 through 255) are available for user interrupt routines.

(i) Software interrupts

These originate directly from program execution (i.e. execution of a "cause interrupt" instruction) or indirectly through program logic (e.g. attempting to divide by zero). An interrupt can be initiated under software control by issuing the "cause interrupt" instruction followed by the appropriate type vector.

(ii) Hardware interrupts

Hardware interrupts, originating from external device, fall into two catagories, maskable and non-maskable (NMI).

A maskable interrupt is triggered by raising the INTR pin from low to high. The processor completes its current instruction and then executes the interrupt acknowledge sequence of Fig.D.6. The CPU causes the bus controller to switch the data bus into the read mode and then branches into the interrupt service routine. To do this it requires that the address of the interrupt vector (the "type vector") be placed on the data bus. This type vector is used as a pointer to a "look-up table"

(Fig.D.5) located in memory which contains the addresses of the interrupt service routines. The processor then executes the appropriate routine; on completion it returns to the main program.

A non-maskable interrupt is caused by raising the NMI pin from low to high. In this case there is no interrrupt acknowledge sequence; instead the CPU goes directly to position 2 in the look-up table to find the address of the service routine. There is only one hardware non-maskable interrupt.

The INTR pin may be masked (disabled) under software control. However, the NMI interrupt will always be serviced, even while another interrupt is being serviced, and can not be disabled.

(d) Memory organisation

Fig.D.7 shows the memory map of the system. The system has a 64 kByte address range with memory locations being labelled from 0H to FFFFH (H indicates a hexadecimal number). Some locations are reserved for the processors use, these being:

0H to 40H for the interrupt look-up table.

FFF0H to FFFFH for the power-on reset routine.

The processor starts executing the program at FFF0H every time it receives a valid reset signal. In this design the reset signal is generated on power-on or when the "reset" switch is operated. Hence the program at FFF0H must be located in read only memory (ROM). This interrupt look-up table can be stored in either ROM or read/write memory (RAM). RAM is chosen so that the same device may be used for the system stack (which must be in RAM) and also to store program

variables.

2 kByte of memory is set aside to map the peripheral devices into memory.

(e) Address decoding

The memory map for this system is shown in Fig.D.7. This shows the system with one 32K EPROM chip, two 8K RAM chips and 2K set aside to map some programmable devices into the memory.

Selection of devices is performed by the chip select logic. Each of the memory sockets accepts a range of memory devices; hence the chip select logic must be able to accommodate these different devices. To provide this flexibility, a small PROM is used to generate the major chip select logic signals. As a result the memory map can be changed simply by programming a new PROM. Additional minor decoding is required to select system peripheral devices. These include the interrupt controller, USART, PIT, PIC etc.. For each device a minimum of 2 and a maximum of 128 memory locations are provided, this extra decoding being done with a 3-8 line decoder. The PROM selects the decoder and it in turn further decodes the address bus further, as shown in Fig.D.8.

(f) Bus controller

The Intel 8288 Bus Controller is used when the 8088 processor in the max. mode to provide command and control signals. Its command logic decodes the 8088 CPU status lines (S0/S1/,S2/) to determine what command is to be issued. The bus controller has two operational states, these being I/O Bus mode and System Bus mode. In this design the 8288 is set to System Bus mode, commands being issued accordingly.

(g) Numerical data processor

The Intel 8087 Numerical Data Processor (NDP) is included in the system as a co-processor to perform fast mathematical operations on a variety of numeric data types. The NDP works in parallel with the main CPU (as shown in Fig.D.9), that is, it is not perceived as a separate device. By using this device the computational capabilities of the CPU are greatly expanded.

(h) Analogue input

The input signal is converted to a digital code using an Analogue to Digital Converter (ADC). Before conversion takes place the signal passes through interfacing circuitry (Fig.D.10) which conditions the signal before it is fed to the ADC [D.2].

(i) Analogue output

The output control signal is required in analogue voltage form. Digital to analogue conversion is carried out using a 12 bit Digital to Analogue Converter (DAC) supplemented by a booster amplifier as shown in Fig.D.11. The signal is fed to an output low pass filter amplifier before leaving the microcontroller to;

* eliminate high frequency components and transient spikes introduced by the DAC
* give the control signal the drive capability [D.3]
* allow for offsetting the analogue output.

(j) Serial communications

The RS232 system transmits and receives data in serial form. Interfacing to the serial I/O system is performed using a Universal Synchronous/Asynchronous Receiver/Transmitter (USART). This device also adds various control bits to the serial data as required by the RS232 standard.

As far as the system is concerned, the serial port appears as a single memory location. One USART is required for each serial channel. The baud rate reference is generated by a programmable timer device.

(k) Backplane bus

This is a 64 way bus as shown in Table.D.1. Its full functional description is given in the IEEE P1000 specification [D.4].

(l) Hardware interrupt structure

As the CPU can only deal with one hardware generated interrupt (excluding the NMI) it needs to be supplemented for general purpose working. In this design system interrupts are handled by a programmable interrupt controller (PIC) which supports 8 hardware initiated interrupts. These 8 interrupts form the input signals to the PIC, this producing a single output interrupt request for the CPU. When it receives any interrupt signal the PIC interrupts the CPU; on receipt of the "interrupt acknowledge" response from the bus controller it supplies the processor with the interrupt type vector, as shown in Fig.D.6.

The PIC also resolves priorities when more than one interrupt occurs at the same time. Table.D.2 shows the interrupt structure, with IR7 being the highest priority and IR0 the lowest.

(m) Watchdog timer

This provides a hardware check of the program's operation, causing a system reset in the event of fault conditions. The program addresses the watchdog timer at regular intervals; if for any reason the program fails to do this a NMI is generated to force the program to a restart state. It could also be used to institute an external alarm function.

The watchdog timer is mapped into memory at one location only, its output being connected to the Non Maskable Interrupt (NMI) pin of the CPU. When the watchdog is addressed for the first time it starts a timer and sets its output to a non-interrupt condition. Provided it is addressed within a specified time limit the timer resets and starts timing again. The output signal therefore does not change state. Hence, as long as the timer is addressed at regular intervals, program operation continues in its normal fashion. However, if it is not re-addressed before the timer completes its cycle it generates a non maskable interrupt.

(n) Single step control

This is an invaluable hardware/software debugging tool. It provides a way to stop the CPU in mid cycle (by signalling not-ready on the READY line). Hence the program may be executed one step at a time, allowing the tester to examine the system state at carefully controlled intervals.

D.2 HARDWARE DESIGN

D.2.1 Overview

The target system controller consists of two boards, a digital board and an analogue one. The circuit diagrams of the digital and analogue subsystems are shown in Figs.D.12 and D.13 respectively.

D.2.2 Central processing unit

(a) Processor and clock generator

Fig.D.14 shows the circuit diagram of the CPU system, this being based on the Intel 8088 microprocessor. The processor has two modes of operations, minimum mode and maximum mode. In minimum the processor generates all bus control signals (RD, RW etc.) whilst in maximum mode a separate bus controller chip performs this function. The maximum mode is intended to be used when there is more than one processor in the system (e.g. the Intel 8087 numeric processor). The modes are selected with the MIN/MAX pin.

In this system the 8088 works in maximum mode. It can be seen from Fig.D.14 that the 8088 deals directly with the RQ/GT lines, providing the 8088/8087 handshake control function.

The 8088 requires an external clock signal, this being supplied by the 8284 clock generator. In addition to generating the primary (system) clock signal, this device provides both the hardware reset interface and the insertion of wait states in the bus cycle [D.1].

The clock generator requires an external frequency source. This is shown in Fig.D.14 as a series-resonant crystal input, the CLK frequency being one third of the crystal frequency, with a 33% duty cycle. The crystal frequency is 12MHz, giving a CLK frequency of 4MHz. To run the system slower than 4MHz the PCLK line must be used as the system clock.

The 8284 has two input signals (RDY1 and RDY2) which control the CPU "READY" signal. "RDY1" is used in this design; it is connected to output of the single step circuit.

To cause a valid processor reset the RESET line must be raised high for at least four clock cycles [D.1] which are used for an internal reset by the processor. When reset returns low the processor restarts execution at location FFF0H.

The 8284 provides for a power-on a reset facility. Its RES/ line (an input) is connected to its reset line (an output) via an internal Schmitt inverter. A capacitor-resistor network is connected to the RES/ line as shown in Fig.D.14. On power-on the capacitor has zero volts across it. It then charges up via the resistor until RES/ reaches a trigger potential, at which time the RESET line goes low. This causes the CPU to fetch and execute the instruction at location FFF0H.

The RESET signal is also used by various other devices in the system. It is activated synchronously with the CLK signal and must be active for four clock cycles.

(b) Bus control logic

The 8088 has a multiplexed address and data bus as mentioned in section D.1. Two 74LS373 latches and one 74LS245 transceiver integrated circuits (IC's) shown in Fig.D.14 comprise the bus control logic. The two 74LS373 are used to latch the address onto the address bus and the 74LS245 is used to buffer the data bus.

(c) Bus controller

In the maximum mode the CPU does not provide all control signals since some of these pins are used for other functions. Instead a Bus Controller is used, providing the following control signals: MRDC/ (read), MWTC/ (write), DEN/ (data bus enable), DT/R (data transmit/receive), ALE (address latch enable), and INTA (interrupt aknowledge). Fig.D.15 shows the bus controller.

D.2.3 Numeric data processor

As a co-processor to the 8088, the 8087 is wired directly to the CPU as shown in Fig.D.14. The CPU's queue status lines (QSl and QS1) enable the NDP to obtain and decode instructions in synchronisation with the CPU. The 8087 NDP is invoked directly by the programmer's instructions. There is no need to write instructions that address the NDP as an "I/0" device [D.5]. The NDP's BUSY signal informs the CPU that the NDP is executing code. For normal operations the NDP uses one of the CPU's REQUEST/, GRANT/ lines to obtain control of the bus for data transfers (reads and writes to memory). The NDP utilizes the same clock generator and system bus interface components.

The NDP can interrupt the CPU when it detects an exception condition such as divide by zero, overflow etc.. This is done through the 8259A Interrupt Controller.

D.2.3 Memory

(a) Address decoding

Fig.D.7 shows the memory map of the system and Fig.D.16 shows the chip select logic. A 32 word by 8 bit PROM is used, which provides 8 chip select lines,

each representing at least 2K of system memory. Table.D.3 shows a listing of the contents of the programmed Prom. Three of these lines are used for the memory sockets and a fourth is used to select the area used by the system devices. Table.D.4 shows the memory map of these devices.

This logic produces 3 CS/lines for use by the system memory devices, 7 lines for the system devices and one line for the watchdog timer.

The watchdog timer requires a unique address in the system. Thus one of the system chip select lines is further decoded to produce a single output goes active (low) only when one specific address is present.

(b) Memory sockets

A memory socket is shown in Fig.D.17. It shows which connections have to be made for each memory type.

D.2.4 Interrupts

(a) Interrupt controller

Fig.D.18 shows the 8259A programmable interrupt controller (PIC). This device provides 8 interrupt request lines (IRO - IR7) for use by the system. It also has an interrupt request (INT) and an interrupt acknowledge (INTA) line which are connected to the corresponding lines on the CPU and the Bus Controller respectively.

When the PIC receives an interrupt request on one of its IR lines it raises the INT line to signal the processor. As a result of this the Bus Controller sends the interrupt acknowledge pulses, as shown in Fig.D.6. On receiving the second INTA

pulse the PIC places the type vector, corresponding to the interrupt requested, onto the data bus. The processor reads this and then jumps to the appropriate service routine.

The CPU masks the INT line on entering an interrupt service routine, so the service routine must set it again if interrupts are to be continuously serviced.

The PIC removes its interrupt request on receiving the second pulse from the bus controller and resumes monitoring of the IR lines. If two or more interrupt requests are received simultaneously they are processed according to their priority in the sysem. IR7 is the highest priority in this system and IR0 the lowest. However, other priority structures are available by reprogramming the PIC.

An interrupt is caused by raising the required IR line from low to high thus supplying it with a positive edge. It is possible to use the PIC in the level triggered mode.

The PIC is mapped into memory as two locations (Table.D.4).

(b) Watchdog timer

Fig.D.19 shows the circuit of the watchdog timer. It is based on the use of a 75LS123 retriggerable monostable flip flop. When the timer receives a negative going edge on its A input (from the CS/ logic) the output Q/ goes low. It stays low for a time determined by the capacitor C1 and the resistor R1. If the timer is re-selected before this time it will retrigger; this keeps Q/ low but resets the timer. If the timer is not retriggered within the time-out period Q/ goes high as soon as the time is up, and then interrupts the CPU. This generates a NMI as described in Sec.D.1.2-m.

(c) Single step logic

Fig.D.20 shows the single step logic. The two NAND gates are used to debounce the push button, so that when pressed and then released, a single positive going pulse is produced. The Two D-type flip flops are used to synchronise the ready and clock signals. As designed here, when the "step" button is pressed the ready line goes to a high state for one clock cycle and then returns low.

When flip-flop A receives the positive going edge from the switch debounce circuit it "latches" the logical '1' on its D input to its Q output. This takes the D input of flip-flop B to a logical '1'. On the next positive edge of the clock is latched to the Q output of B. This then resets the '1' of the output of flip-flop A. On the next positive clock edge A '0' will be latched by B. Thus the Q output of B has gone high for one clock cycle and then returned low. This signal can be switched onto the READY line. When the push button is pressed the processor will complete the current instruction and continue to the next. However, READY will have gone low again, so after T3 the processor will wait for READY to go high again before completing the new instruction. Thus a single instruction has been executed.

D.2.5 Programmable timer

Fig.D.21 shows the 8254 programmable interval timer (PIT) I.C. This has three independent timers, each of which can be used in one of five modes.

Each timer has two inputs and one output. The CLK input provides the timing reference for that timer and can be at a different frequency to the system clock. The OUT line is the output from the counter.

In Table.D.4 the PIT is shown to occupy four locations in memory, one device command register and three count registers, one for each timer. The device is

programmed by writing a byte to the command register. This byte indicates which channel it refers to, what mode it is to be set to, and how the count data is to be entered.

The two modes used in this system are square wave mode (baud rate generator) and rate generator mode. In the baud rate mode GATE is tied high to enable counting. OUT goes high for half the count and then low for the other half. This timer then resets to the original count and repeats the above. The system clock is used as the CLK input. This produces a square wave with a 50% duty cycle at the system clock frequency divided by the contents of the count register. Channel 0 is in this mode and is used as the programmable clock for the serial communications channels.

In rate generator mode the timer acts as a "divide by N" counter, generating a single (low) pulse at end-of-count. It automatically reloads its counter and repeats the sequence. In this system its output is connected to the PIC, so providing an accurately timed regular interrupt signal. This is used to set the control loop timing.

D.2.6 Serial communications port

Fig.D.22 shows the serial communications port. The USART (8251A) is mapped into memory as a memory location as shown in Table.D.4.

The clear to send line (CTS/) enables transmission to commence and is tied low. This line is intended for use with a full modem system.

The serial data appears on the TXD line, with each bit shifted out on the falling edge of the TXC. Similarly the received data is input to RXD line. The RXC line is used as the sampling reference. The RXC and TXC are tied together so that the transmit and receive baud rates are the same. The programmable timer (PIT)

acts as the baud rate generator.

The TXRDY line indicates whether the 8251A is ready to accept a byte for transmission. A high indicates that the USART is ready to accept data. On writing data to the port (A0 low) TXRDY goes low to signal a "not ready" status. When the USART can accept the next byte for trasmission TXRDY goes high. This low to high transition is used to signal an interrupt which informs the processor that the USART is now ready to accept the next byte.

When the USART has received a complete byte, it raises the RXRDY line to indicate that it has data ready to be read. This line is used to signal the processor, via an interrupt, to read this data from the USARTs data port.

The serial I/O signals of the UART are TTL compatible. These are processed so that they conform with the required line driving standard of RS232C.

A 741 operational amplifier and a simple transistor switch circuit provide the interface between the UART and the data lines. The receiver circuit requires only the standard 5V power line; however the transmitter circuit needs +15V, -15V and 5V supplies.

The signal lines are connected to a 25 way 'D' connector as shown in Table.D.5.

D.2.7 Analogue input

Fig.D.13 shows the circuit diagram of the analogue input section. It consists of

Signal conditioning

Channel multiplexer

Sample/hold

Analogue to Digital Converter (ADC)

(a) Signal conditioning

Fig.D.23 shows the signal conditioning circuit. It consists of first, a differential amplifier which amplifies the signal to a level that can be handled by the ADC. An OP07 amplifier is used in this circuit. The input amplifiers are protected against transient overvoltage by employing four diodes as shown in Fig.D.23. The input amplifier is followed by a 3rd order low-pass anti-alias filter, incorporated to eliminate high frequency components.

(b) Channel multiplexer

This device provides and controls the sharing of the ADC. Fig.D.24 shows the circuit diagram of the multiplexer. In this design the Hi-508A multiplexer is used. It can handle 8 input channels, they being used for

* Input signal
* Reference voltage
* Output signal

The 8 input channels are mapped to the memory as shown in Table.D.4.

(c) The sample/hold

Fig.D.25 shows the sample/hold device. This is a Harris HA-2420 device which has an acquisition (sample) time of 5 uSec. The output of the analogue

multiplexer is fed to the sample/hold input. Sample/hold commands are issued from the ADC using the EOC line. When the ADC starts conversion it issues a command to the S-H unit to hold the signal; at the end of conversion it automatically issues the sample command.

(d) Analogue to digital converter

The Analogue to Digital converter (Fig.D.26) is a 12-bit successive approximation analogue to digital converter with a conversion time of 25 usec. This device, Hybrid Systems AD574A type, is set for bipolar operation, for a ±5V range. It is mapped into the memory as 4 locations as shown in Table.D.4.

The control signals CE, CS/, and R,C/ control the operation of the converter. Table.D.6 shows the ADC truth table relating to system operation.

D.2.8 Analogue output

Fig.D.13 shows the circuit diagram of the analogue output. It consists of

Digital to Analogue Converter (DAC)

Output filtering and amplification

(a) The digital to analogue converter

Fig.D.27 shows the circuit diagram of the AD7542 [D.6], DAC, this being a precision 12-bit multiplying DAC. The DAC is set for bipolar mode having an output range of ±10 volts (when fitted with a buffer amplifier).

To operate the DAC the following sequence of memory write instructions are

executed.

* Load the LOW byte data register.
* Load the MIDDLE byte data register.
* Load the HIGH byte data register.
* Load the 12-bit DAC register.

Address lines A0 and A1 determine the operation of the AD7542. These lines are decoded internally in the DAC to point to the desired loading operation (i.e. load high byte, middle byte, low byte or DAC register). Table.D.7 shows the AD7542 truth table. These operations are identical to the write cycle of a RAM as shown in Table.D.7. Therefore the DAC appears to the memory as 4 locations as shown in Table.D.4.

The DAC requires a stable reference voltage (10V) which is provided by the REF-01.

(b) Output filtering and amplification

The analogue signal from the DAC is fed to a buffer amplifer to improve its drive capability. It is then fed to low-pass amplifier which minimises the effects of DAC glitch spikes and high frequency components. Both short circuit and transient overvoltage protection are included for the output amplifiers using diodes as shown in Fig.D.27.

D.2.9 Power supplies

The 5V and ±15V power lines from the host system are fully stabilised. To

suppress power line "spikes" due to fast switching logic a 0.1 uF disk ceramic capacitor is placed between the +VE supply pin to each I.C. and its ground pin. A 100 uF electrolytic capacitor is placed across the supply to deal with any sudden, short term high current demands.

A voltage reference REF-01 is used to provide the analogue circuits with a 10 V reference voltage as shown in Fig.D.13.

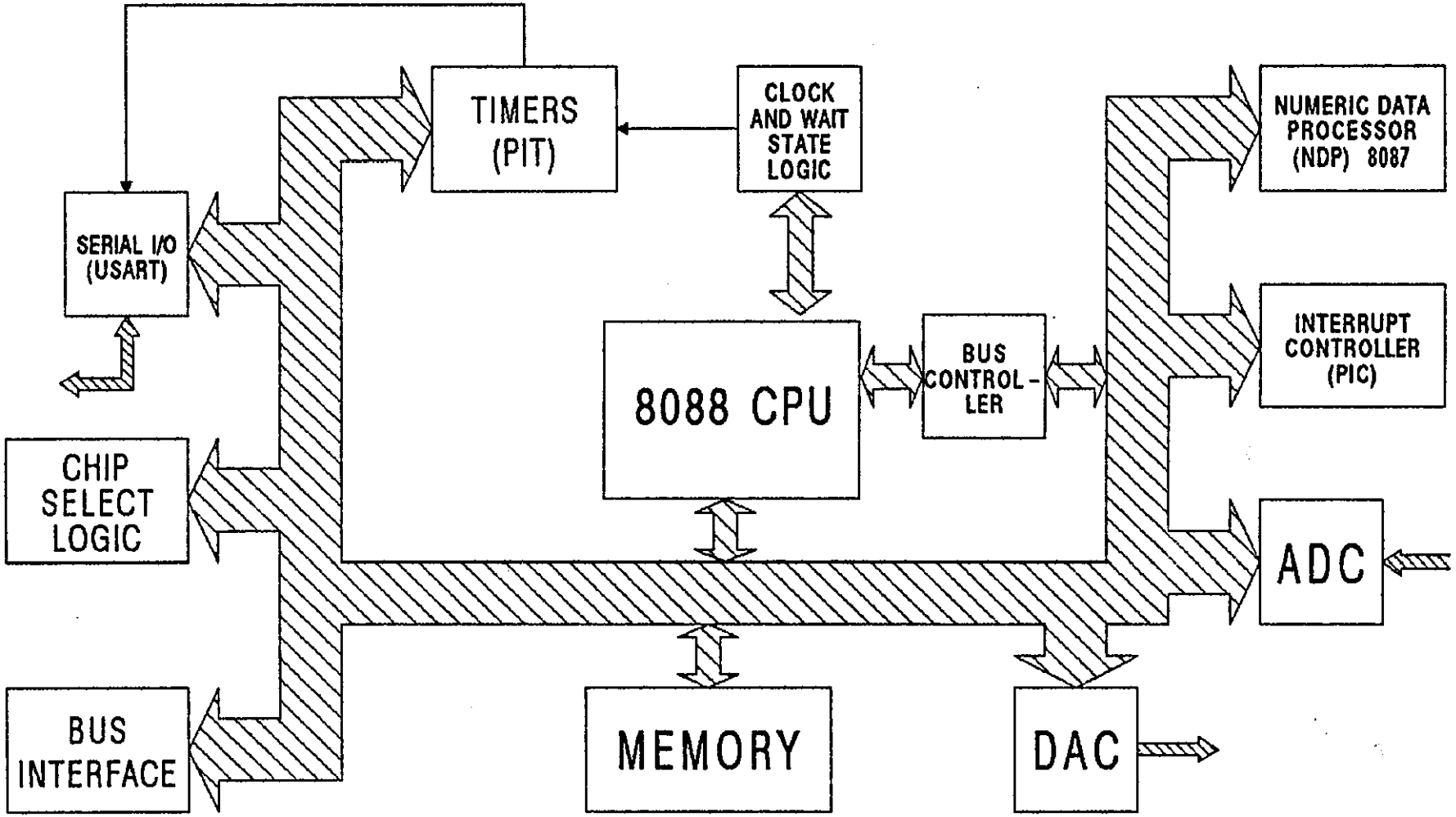


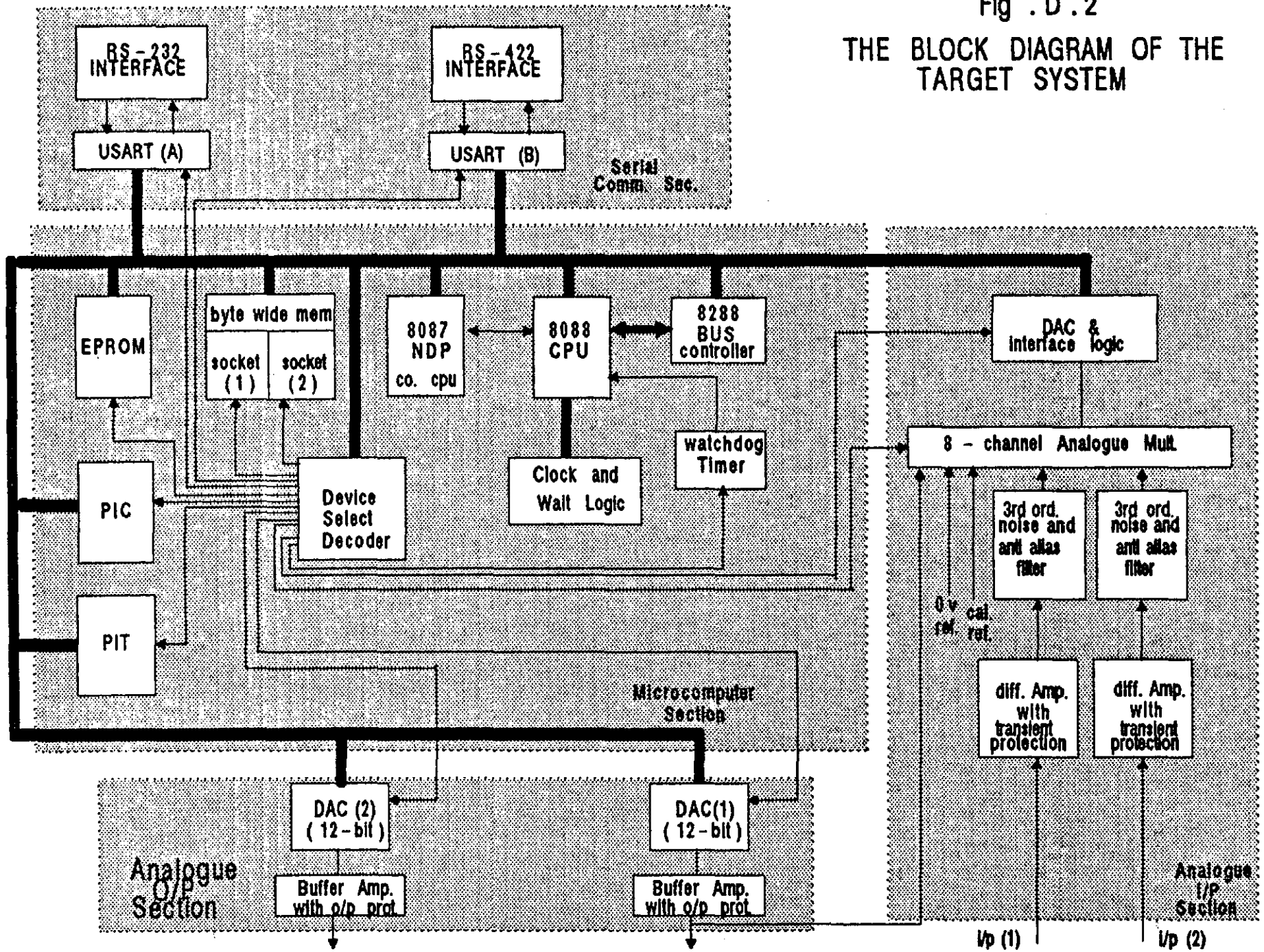
Fig . D . 1

TARGET SYSTEM CONTROLLER

OVERALL STRUCTURE

Fig . D . 2

THE BLOCK DIAGRAM OF THE TARGET SYSTEM



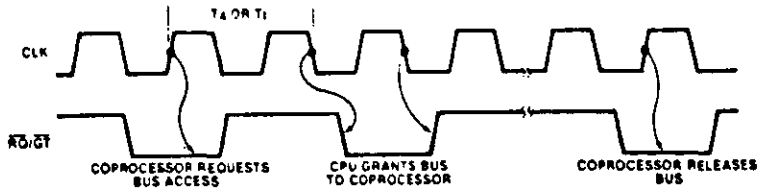


Fig . D . 3

Request / Grant Timing

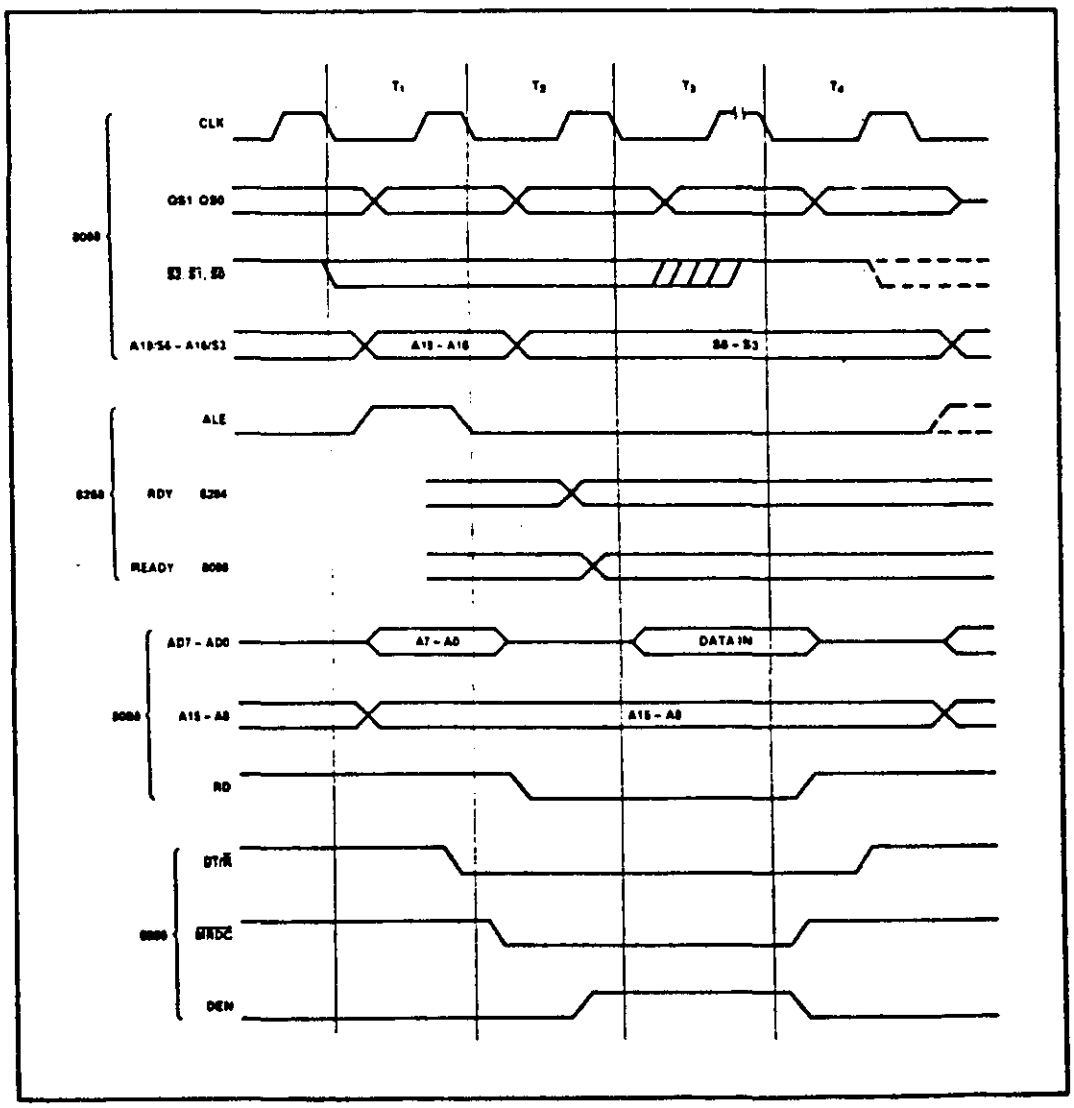


Fig . D . 4

System Timing

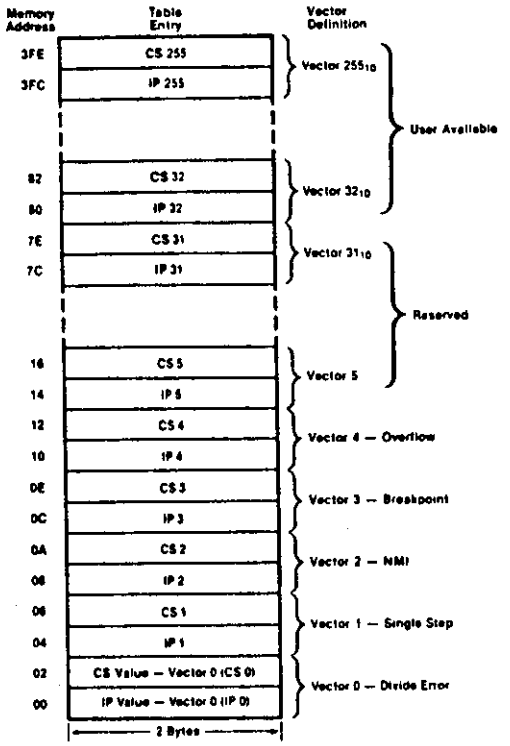
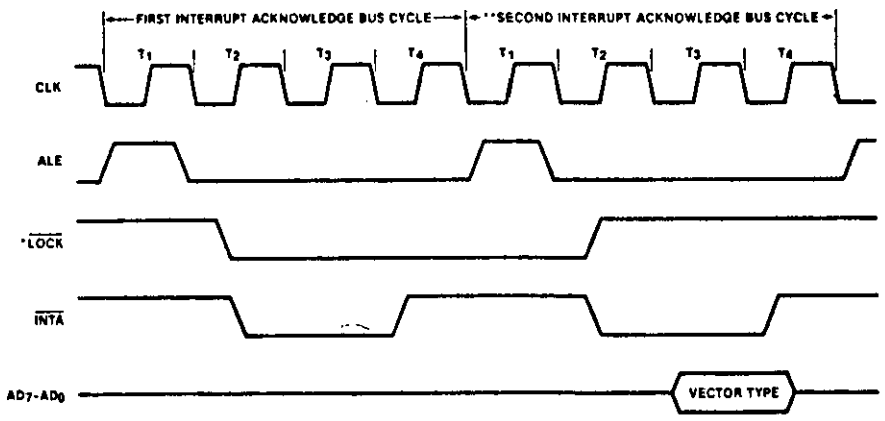


Fig . D . 5

Interrupt Vector Table



*MAXIMUM MODE ONLY

**SEVERAL (3 TYPICAL) IDLE CLOCK STATES OCCUR BETWEEN THE FIRST AND SECOND INTERRUPT ACKNOWLEDGE BUS CYCLES IN THE 8086 CPU (DURING THIS INTERVAL THE BUS IS DRIVEN). INTERRUPT ACKNOWLEDGE BUS CYCLES OCCUR BACK-TO-BACK IN THE 8088 CPU.

Fig . D . 6

Interrupt Acknowledge Sequence

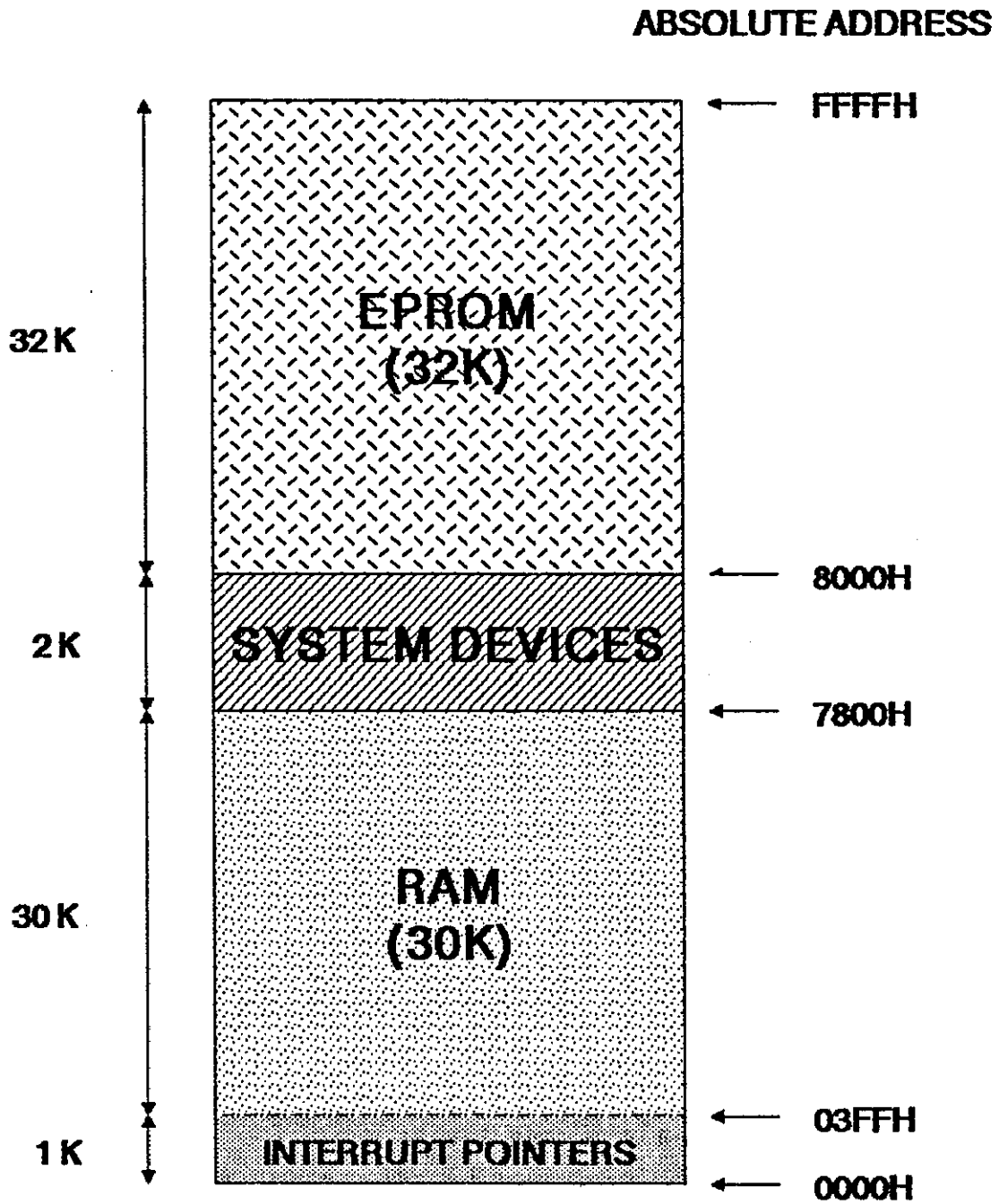


Fig . D . 7

THE MEMORY MAP OF THE SYSTEM

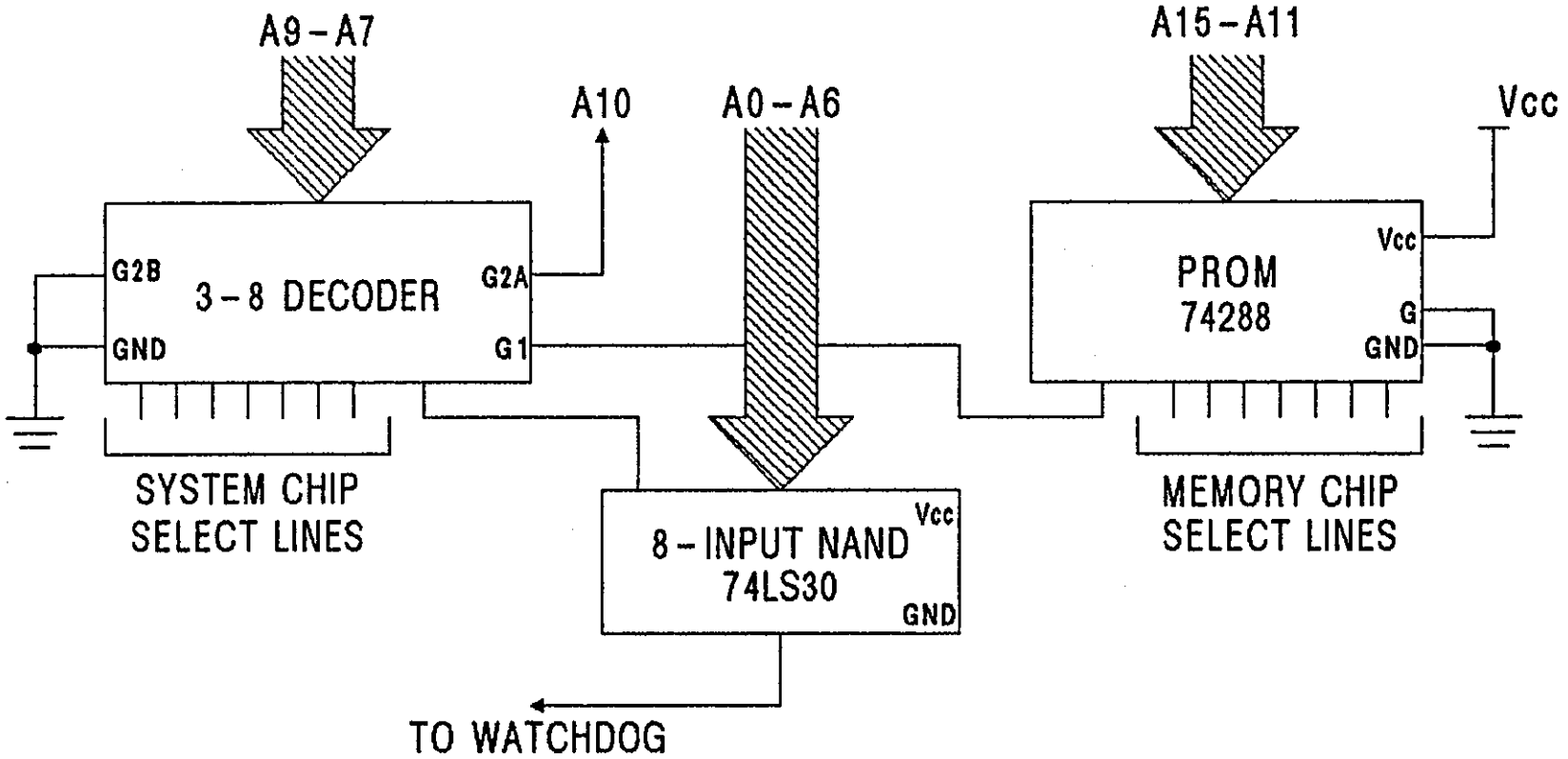


Fig.D.8

ADDRESS DECODING

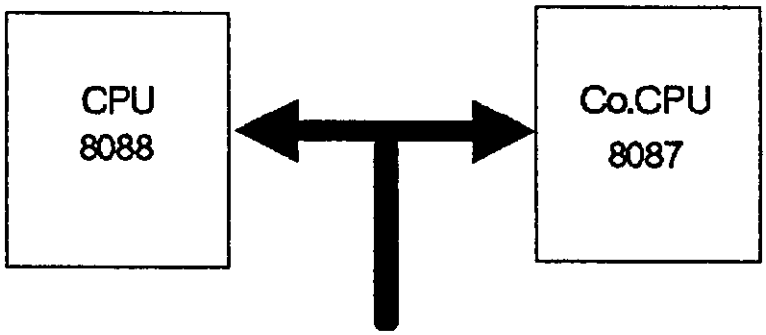


FIGURE D.9

NUMERICAL DATA PROCESSOR

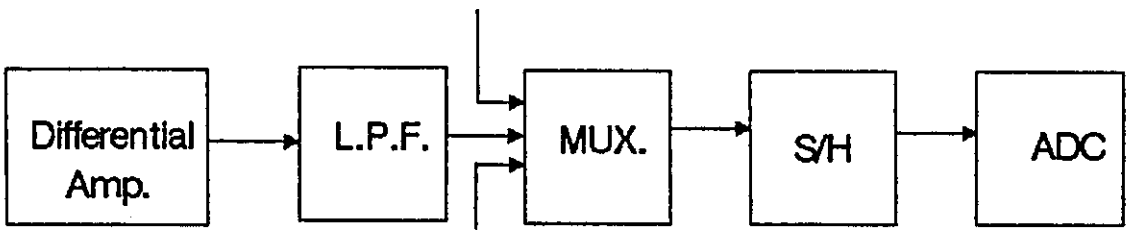


FIGURE D.10

BLOCK DIAGRAM OF ANALOGUE I/P

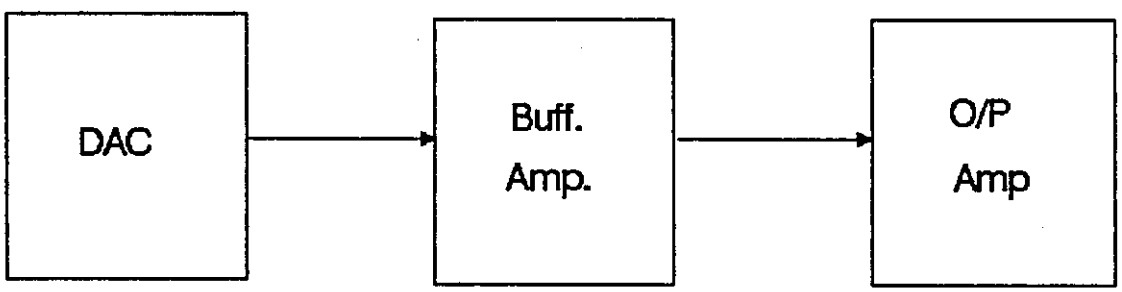
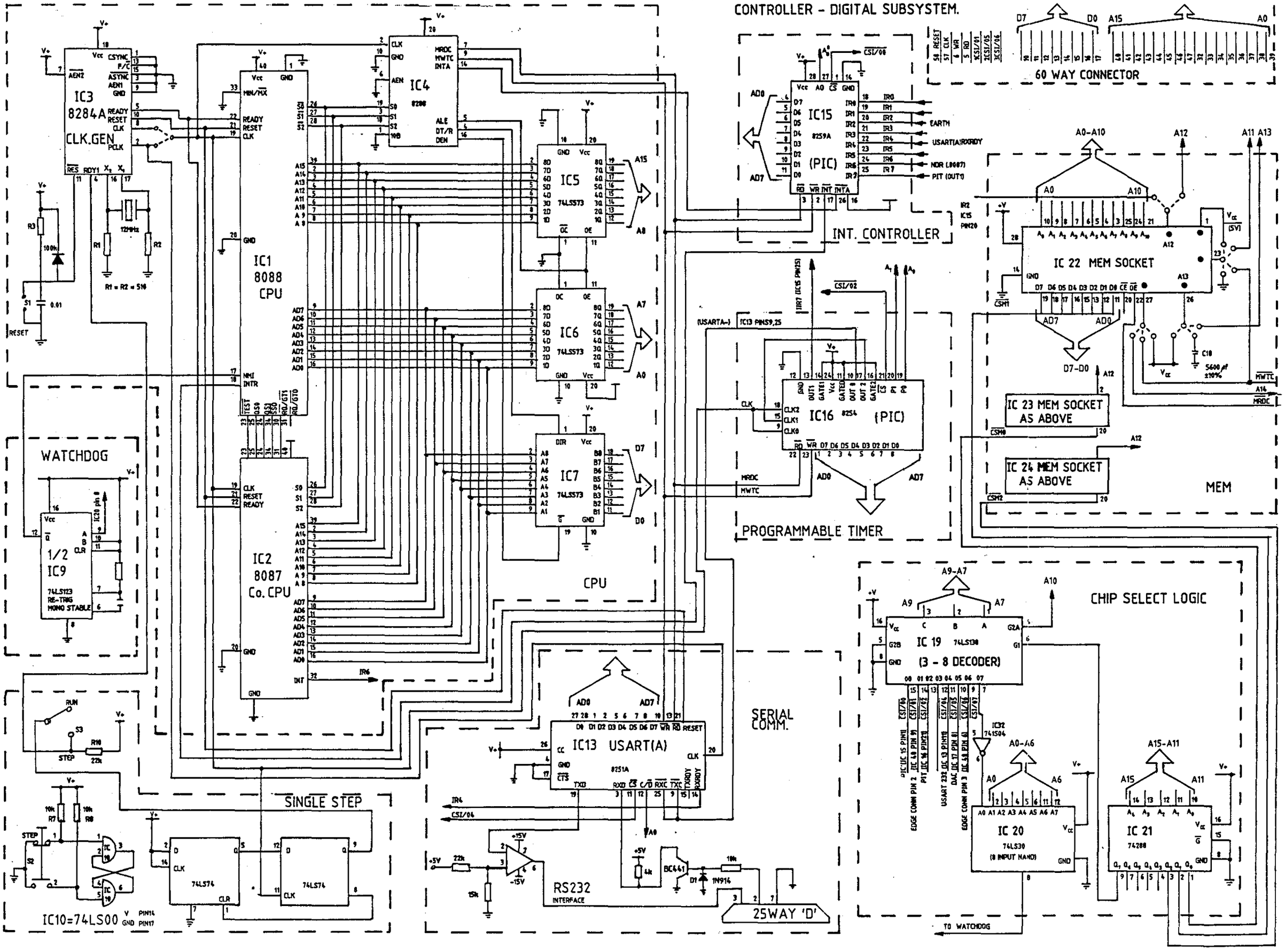


FIGURE D.11

BLOCK DIAGRAM OF ANALOGUE O/P

FIG. D.12. CIRCUIT DIAGRAM OF THE SELF-TUNING CONTROLLER - DIGITAL SUBSYSTEM.



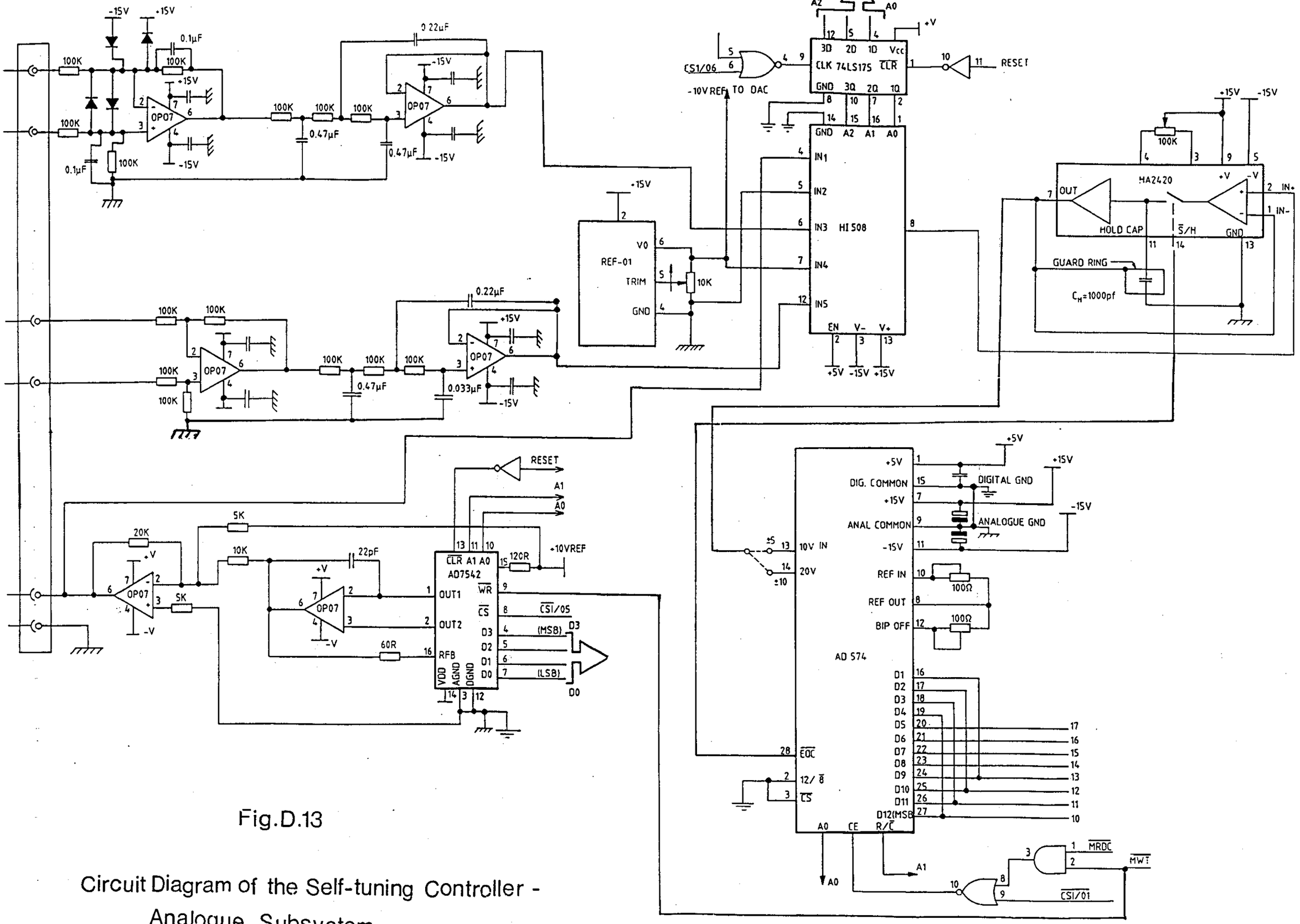


Fig.D.13

Circuit Diagram of the Self-tuning Controller - Analogue Subsystem

NOTE..
Digital and analogue grounds are to be connected together only at the ADC./DAC

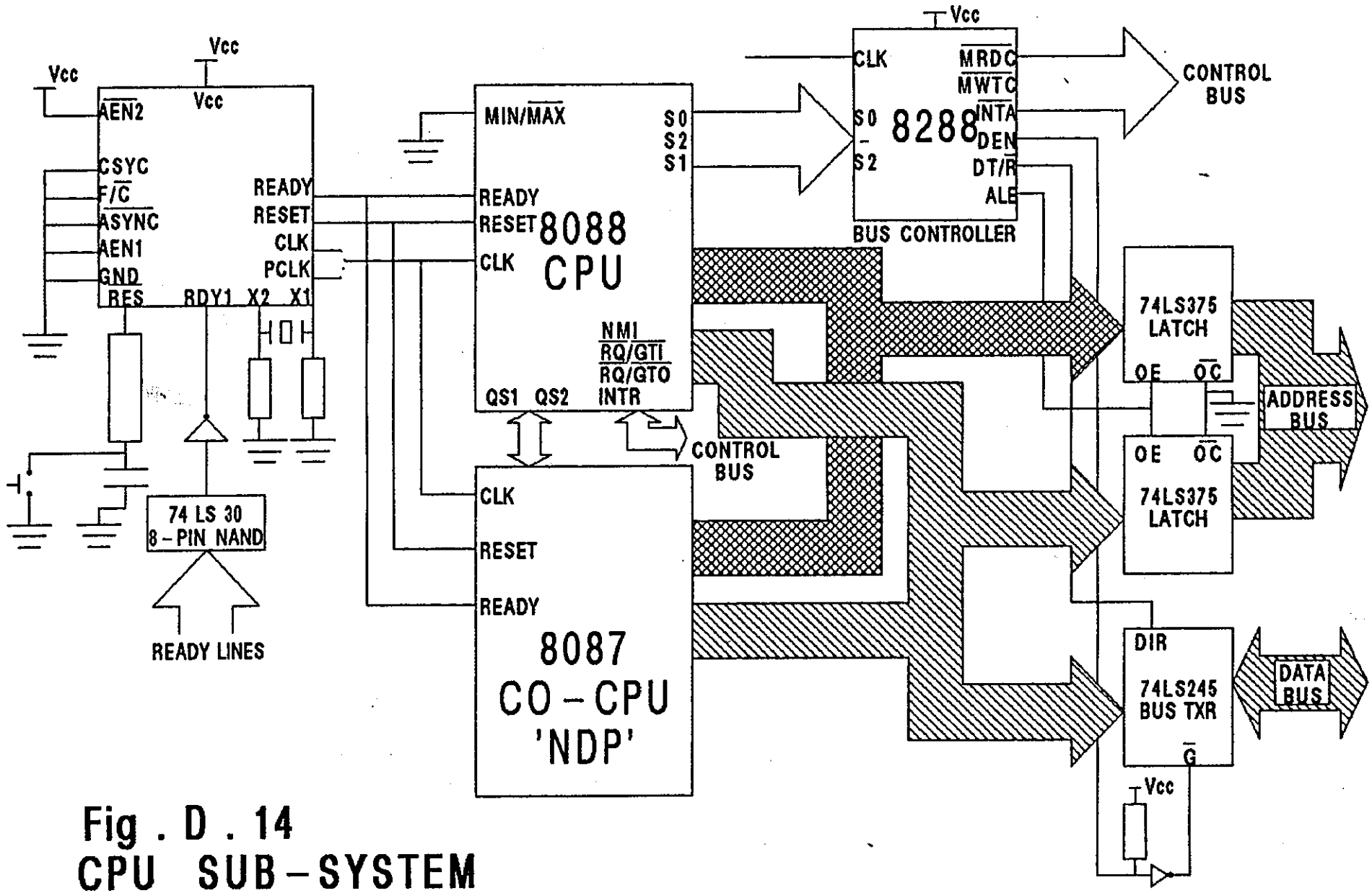


Fig . D . 14
CPU SUB-SYSTEM

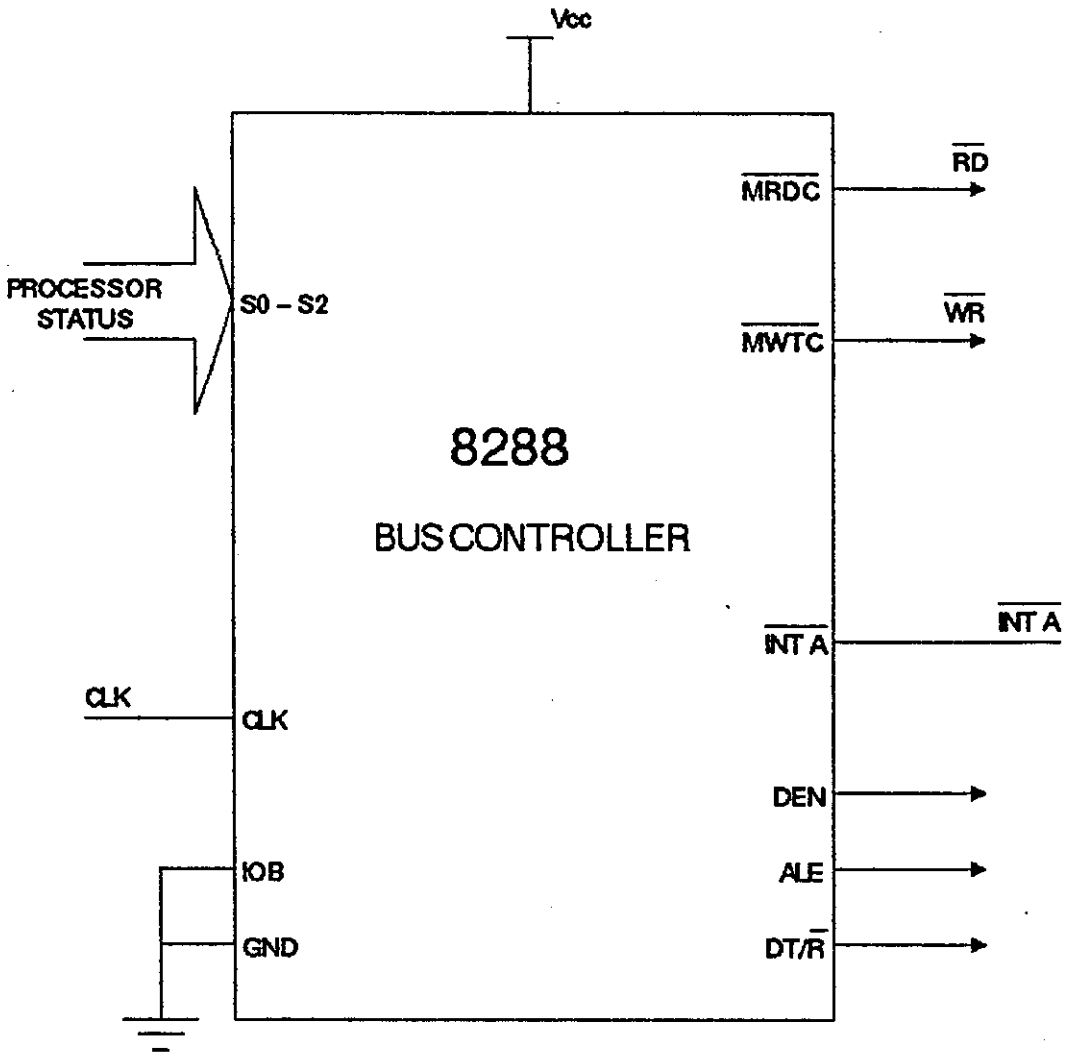


Fig . D . 15

THE 8288 BUS CONTROLLER

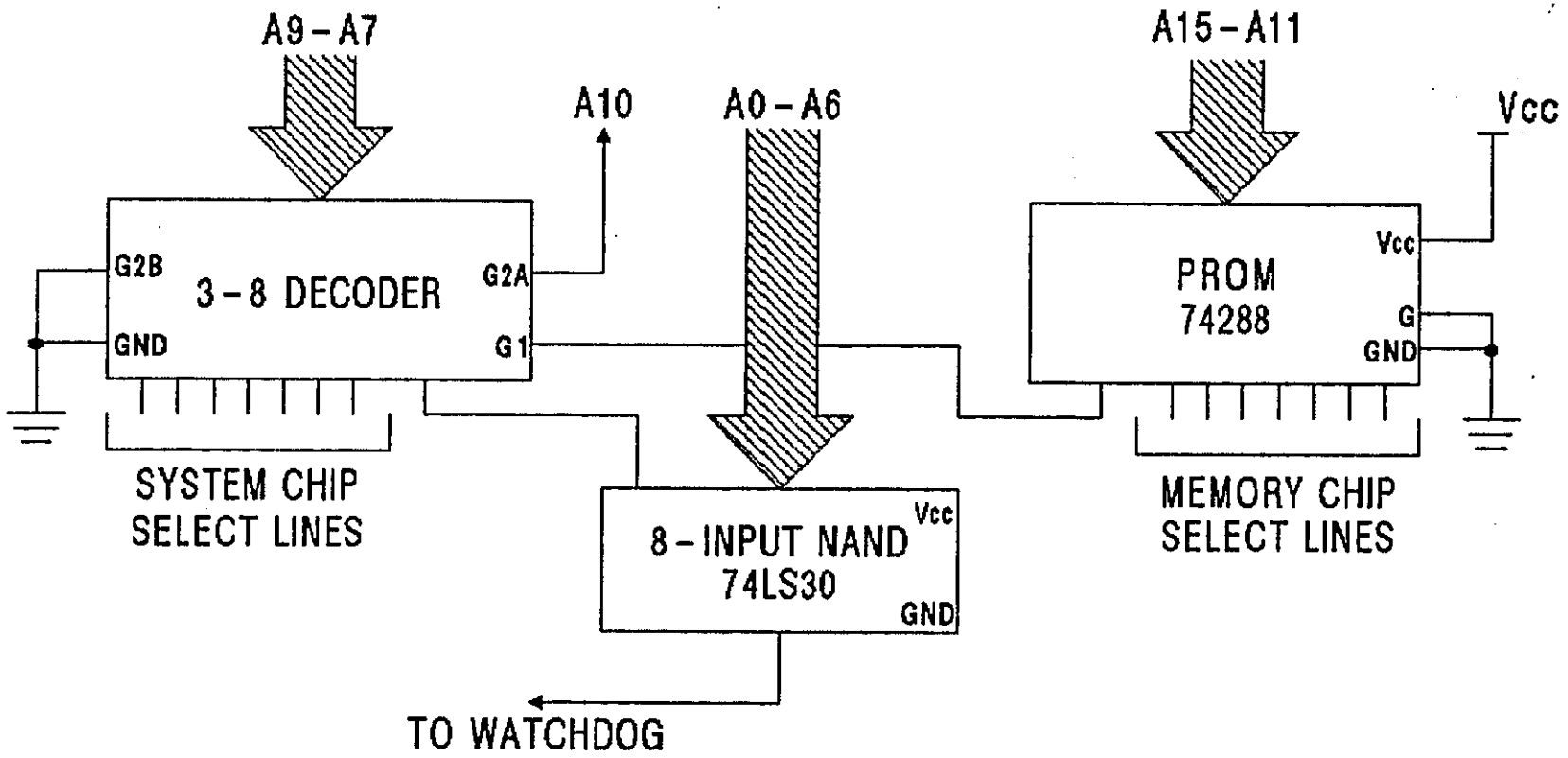
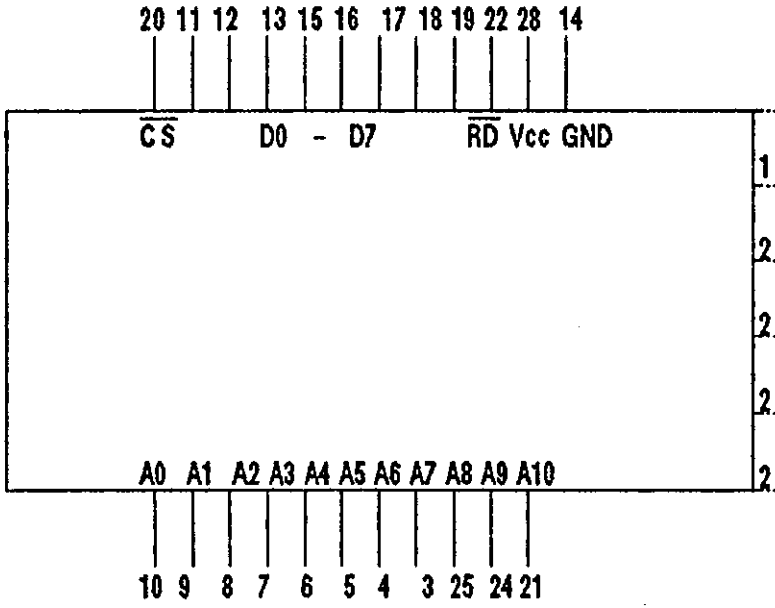


Fig . D . 16

ADDRESS DECODING



	EPROM 8K 2764	EPROM 16K 27128	EPROM 32K 27235	RAM 2K x 8 4802	RAM 8K x 8
1	Vcc	Vcc	Vcc	X	Nc
2	A12	A12	A12	X	A12
23	A11	A11	A11	$\overline{WR}$	A11
26	X	A13	A13	Vcc	Vcc
27	Vcc	Vcc	A14	X	$\overline{WR}$

Fig . D . 17

THE MEMORY SOCKET CONNECTIONS

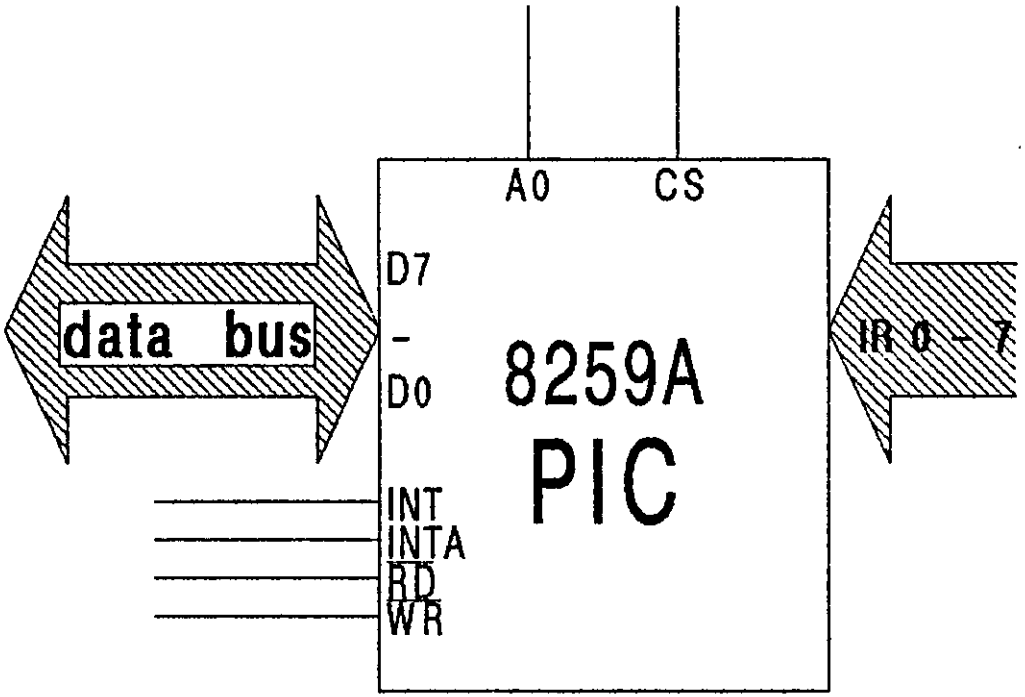


Fig . D .18

INTERRUPT CONTROLLER

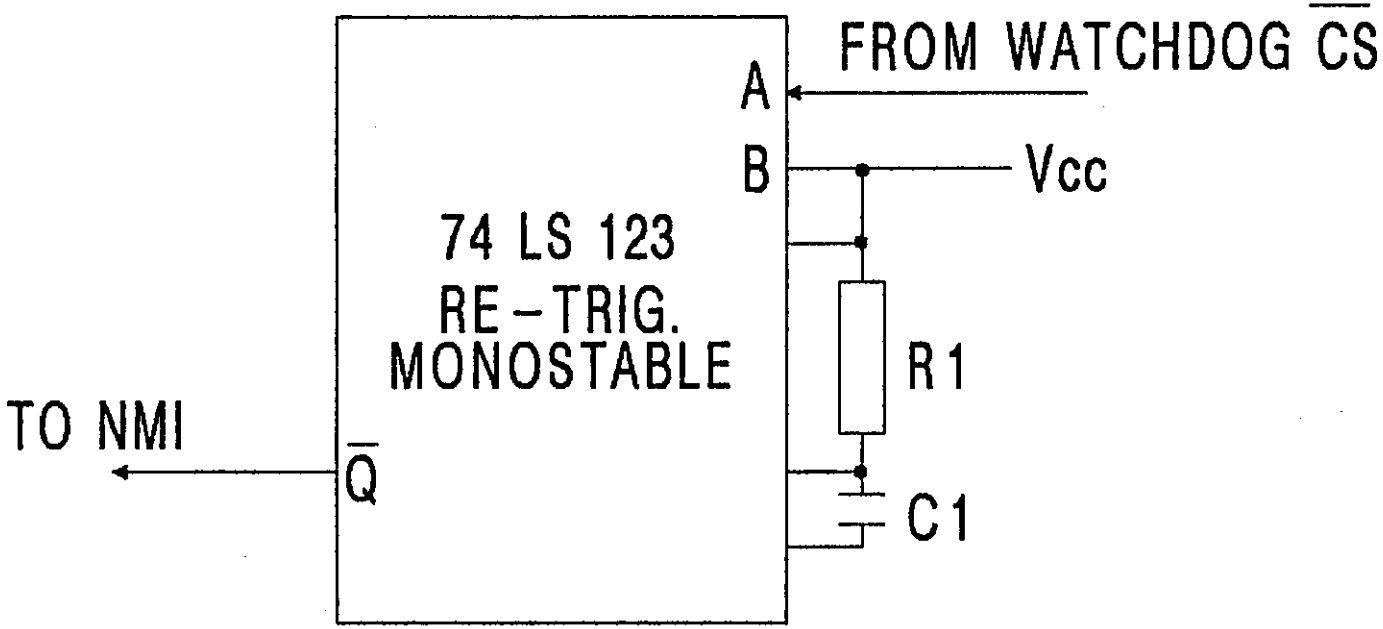


Fig . D . 19

WATCHDOG TIMER

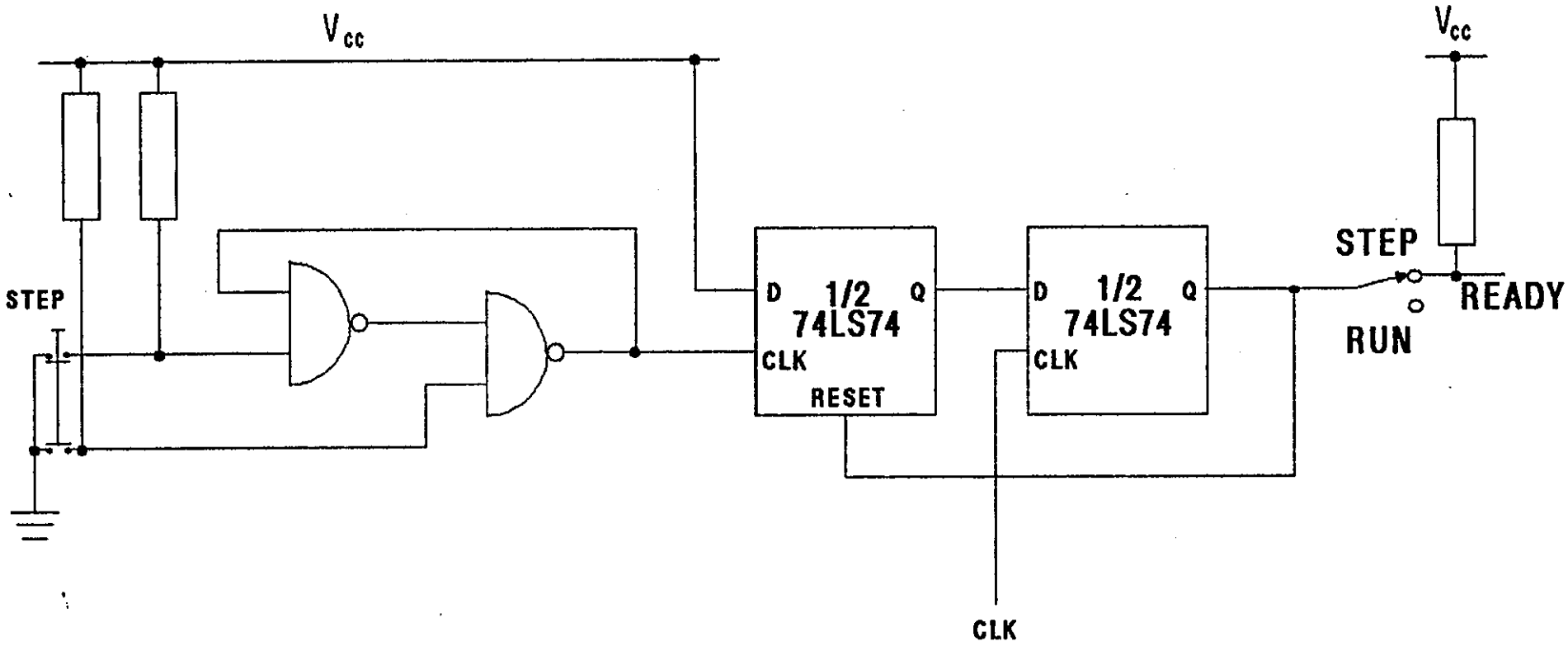


FIGURE D.20

SINGLE STEP LOGIC

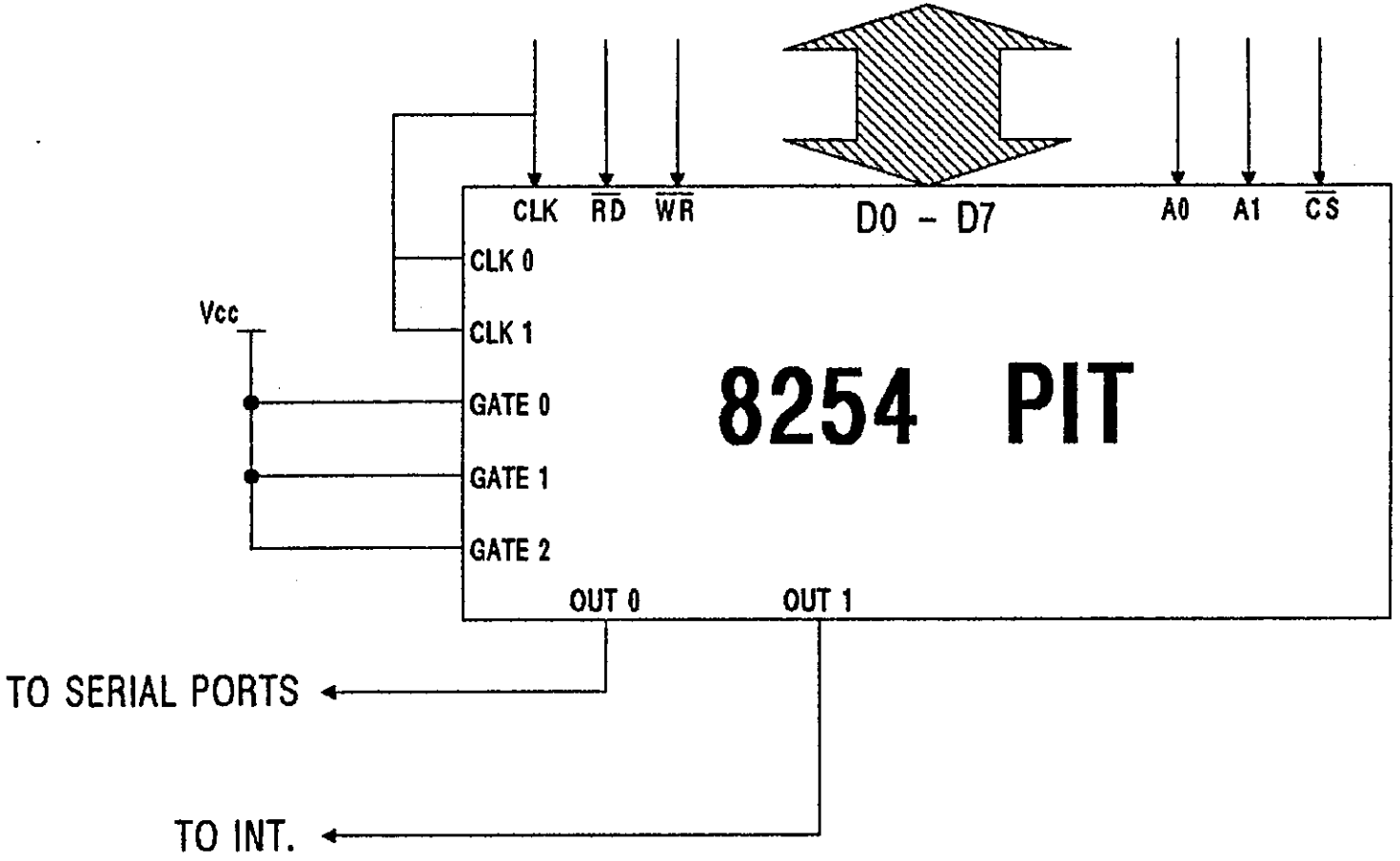


Fig . D . 21

PROGRAMMABLE TIMERS

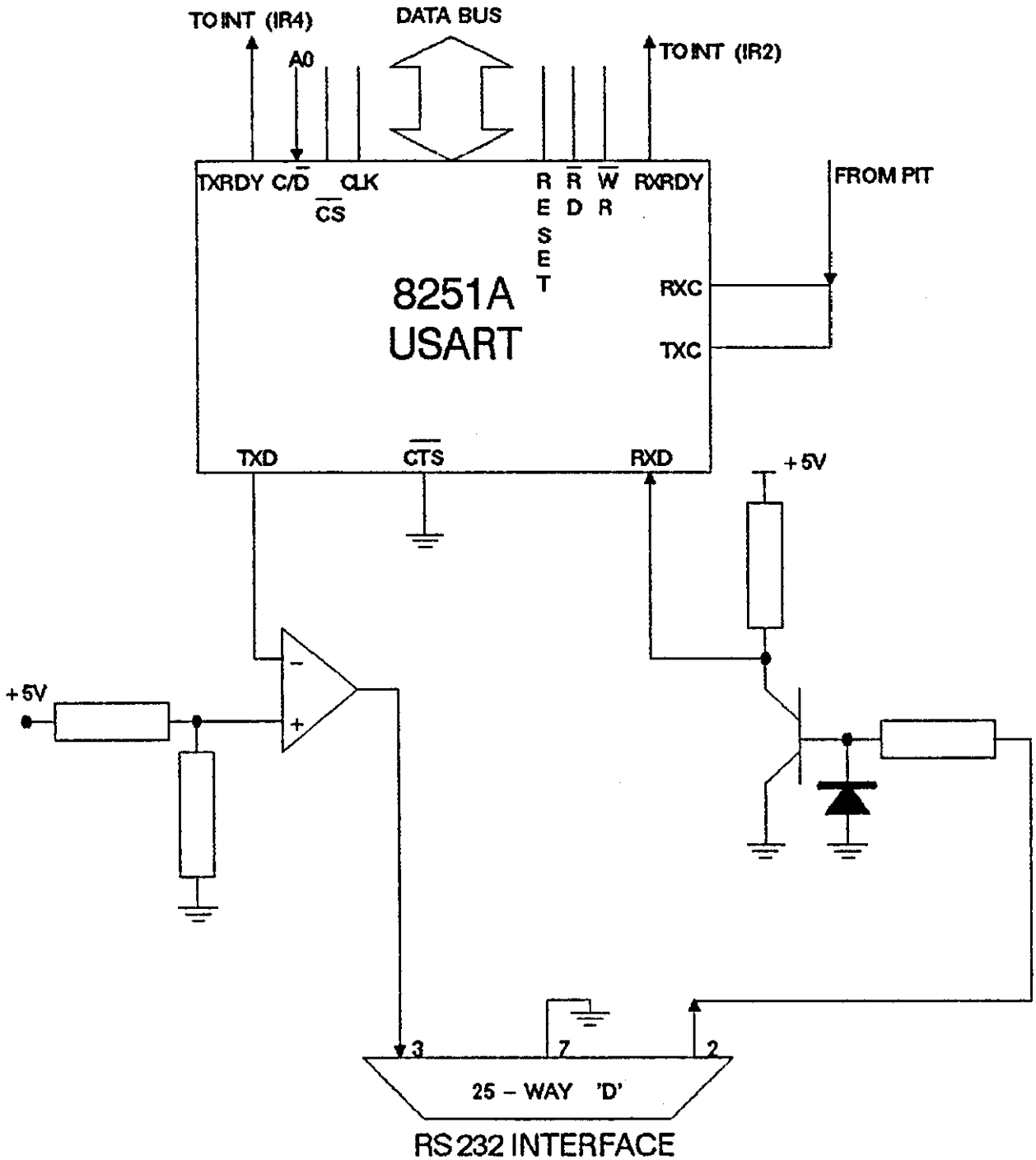


Fig . D . 22

SERIAL COMMUNICATIONS

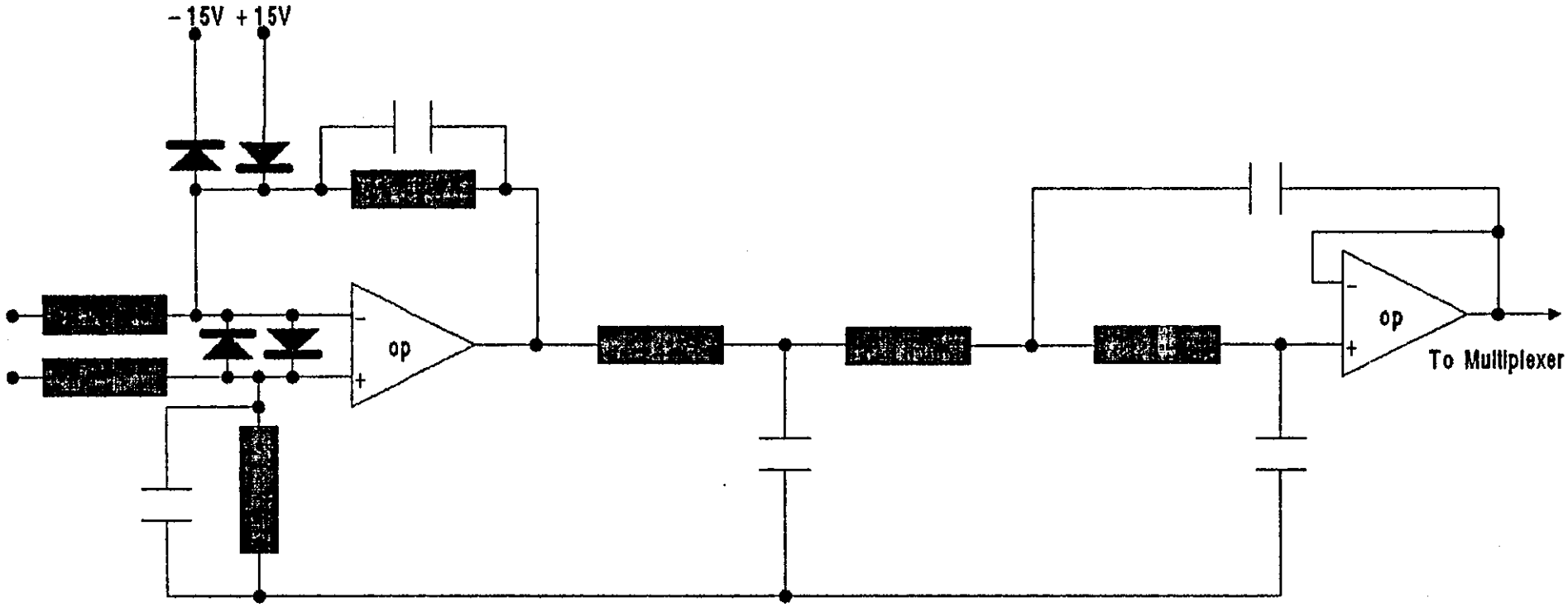


Fig . D . 23

SIGNAL CONDITIONING CIRCUIT

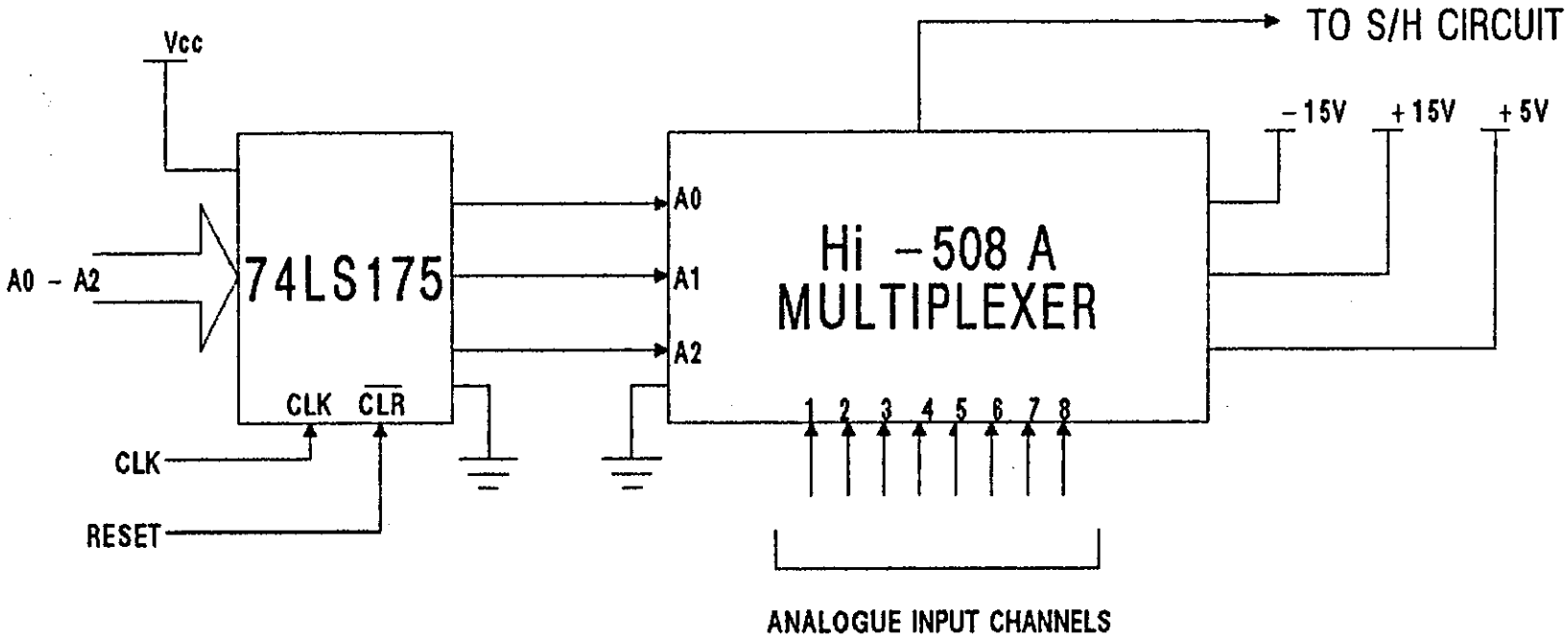


Fig . D . 24

ANALOGUE MULTIPLEXER

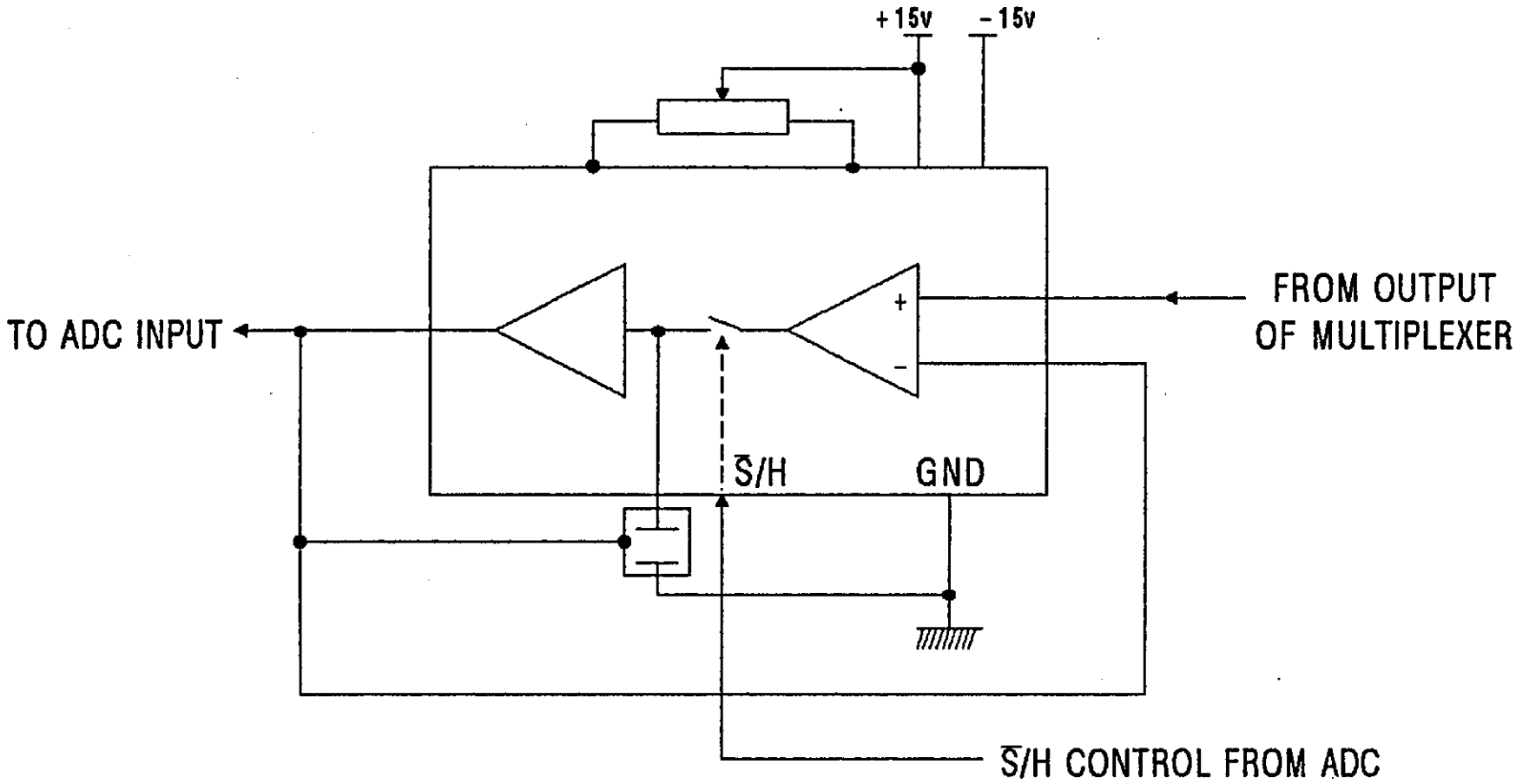


FIGURE D.25

THE SAMPLE AND HOLD DEVICE

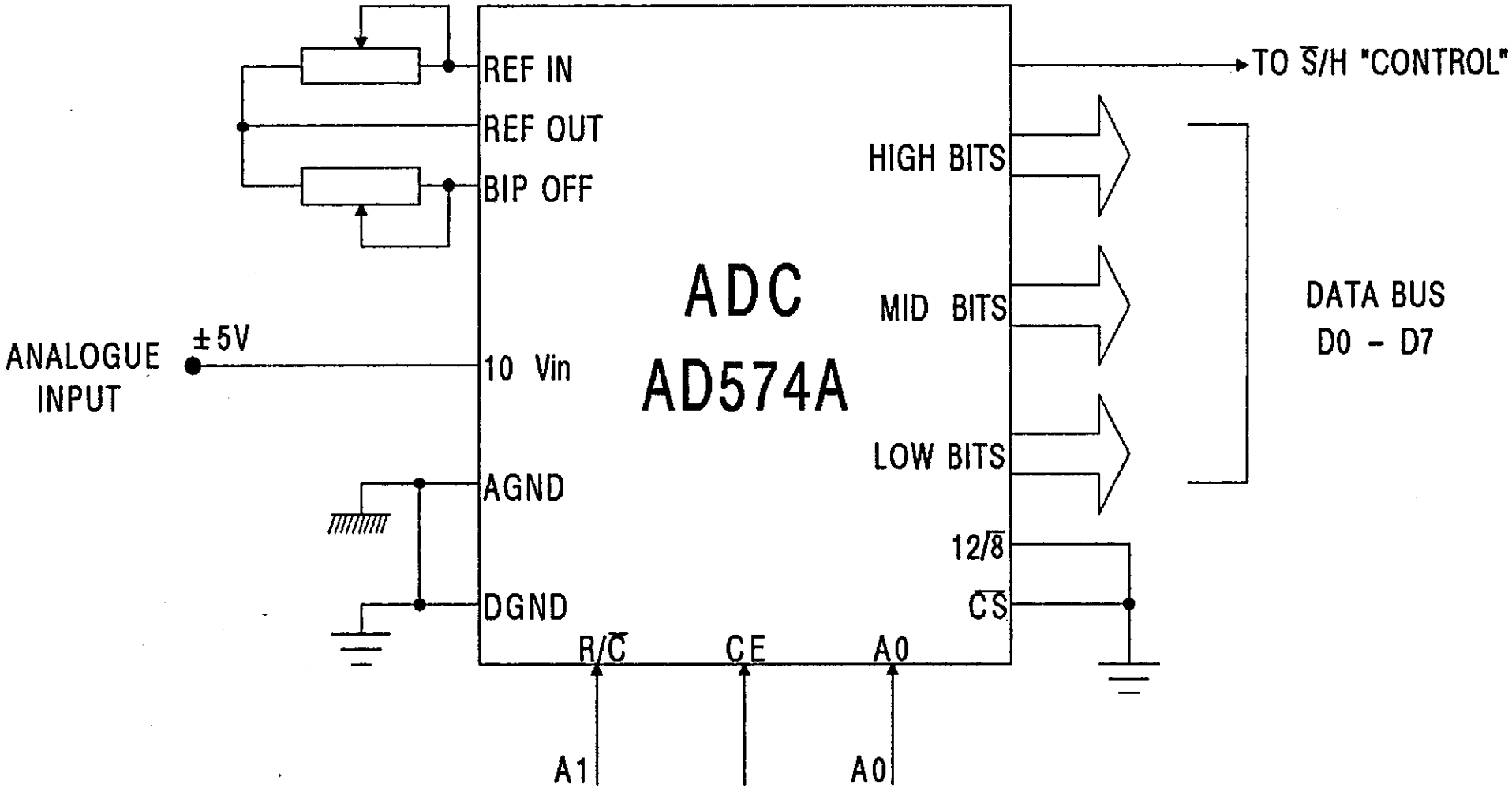


FIGURE D.26

BIPOLAR INPUT CONNECTIONS FOR ADC

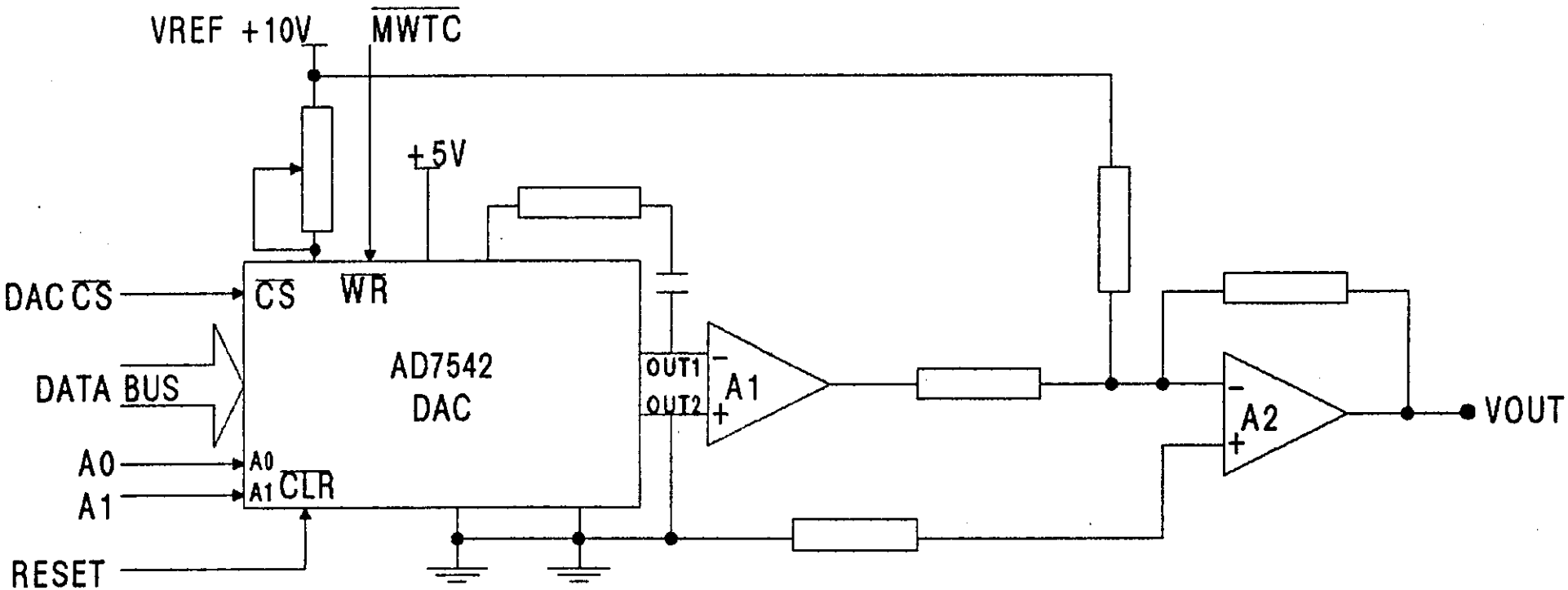


FIGURE D-27

ANALOGUE OUTPUT CIRCUIT

TABLE. D-1

IEEE P1000 Bus Pin Designations

	a	b	c
1	0V		0V
2	+5V		+5V
3	D0		D1
4	D2		D3
5	D4		D5
6	D6		D7
7	A0		0V
8	A2		A1
9	A4		A3
10	A6		A5
11	A8		A7
12	A10		A9
13	A12		A11
14	A14		A13
15	0V		A15
16	A16		A17
17	A18		A19
18	SWRITE*		SIOREQ*
19	ADRSTB*		0V
20	XFRACK*		DATSTB*
21	INTRQ1*		0V
22	INTRQ3*		INTRQ2*
23	INTACK*		INTRQ4*
24	BUSREQ*		BUSBSY*
25	BUSERR*		RESET*
26	SYSCLK*		+V STBY
27	IACKIN*		IACKOT*
28	BACKIN*		BACKOT*
29	0V		0V
30	-AUX V		+AUX V
31	+5V		+5V
32	0V		0V

Int. Type	Device
IR 0	X
IR 1	X
IR 2	X
IR 3	USART(A – B)TXRDY
IR 4	USART(A) RXRDY
IR 5	USART(B) RXRDY
IR 6	NDP (8087)
IR 7	PIT (OUT – 1)

TABLE . D.2

INTERRUPT VECTOR TABLE

HEX ADDRESS	HEX CONTENTS	MEMORY SOCKET NO.
00	7E	CSM0 (1) (8K RAM)
01	7E	
02	7E	
03	7E	
04	7D	CSM1 (2) (8K RAM)
05	7D	
06	7D	
07	7D	
08	7F	RESERVED FOR RAM
09	7F	
0A	7F	
0B	7F	
0C	7F	
0D	7F	
0E	7F	
0F	FF	G1 (SYSTEM DEVICES)
10	7B	CSM2 (32K EPROM)
11	7B	
12	7B	
13	7B	
14	7B	
15	7B	
16	7B	
17	7B	
18	7B	
19	7B	
1A	7B	
1B	7B	
1C	7B	
1D	7B	
1E	7B	
1F	7B	

TABLE D·3

THE LISTING OF THE PROM CODE FOR MEMORY MAP

DEVICE	REGISTER	HEX. ADDRESS
PIC	COMMAND 0 COMMAND 1	7800 7801
ADC	12 – BIT CONVERT 8 – BIT CONVERT READ 8 – MSB READ 4 – MSB	7880 7881 7882 7883
PIT	COUNT CH. 0 COUNT CH. 1 COUNT CH. 2 CONTROL	7900 7901 7902 7903
232 – USART	DATA COMMAND	7A00 7A01
DAC	LOW 4 – BIT BYTE MID. 4 – BIT BYTE HIGH 4 – BIT BYTE LOAD 12 – BIT BUFFER	7A80 7A81 7A82 7A83
INPUT CHANNELS	CH. 1 (OUTPUT OF DAC) CH. 2 CH. 3 (INPUT – 1) CH. 4 (REFERENCE) CH. 5 (INPUT – 2) CH. 6 CH. 7 CH. 8	7B00 7B01 7B02 7B03 7B04 7B05 7B06 7B07
WATCHDOG		3BFF

TABLE D.4

THE MEMORY MAP OF THE SYSTEM DEVICES

'D' CONNECTOR	
PIN	FUNCTION
2	RS 232 TRANSMIT
3	RS 232 RECEIVE
7	GROUND

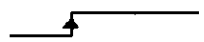
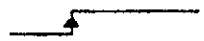
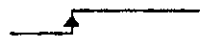
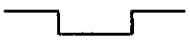
TABLE D·5

THE CONNECTIONS OF THE SERIAL PORT

CE	$\overline{CS}$	R/$\overline{C}$	12/$\overline{8}$	Ao	OPERATION
0	X	X	X	X	None
X	1	X	X	X	None
1	0	0	X	0	Initiate 12 – bit Conversion
1	0	0	X	1	Initiate 8 – bit Conversion
1	0	1	Pin1 (5v)	X	Enable 12 – bit parallel output
1	0	1	Pin15 (Gnd)	0	Enable 8 Most Significant Bits
1	0	1	Pin15 (Gnd)	1	Enable 4 LSBs and 4 trailing zeros

Table D·6

TRUTH TABLE FOR ADC

A1	A0	$\overline{CS}$	$\overline{WR}$	$\overline{CLR}$	OPERATION
X	X	X	X	0	Reset DAC 12 – bit reg. to 000H
X	X	1	X	1	No op.
0	0	0		1 *	Load low byte data register on edge
0	1	0		1 *	Load mid byte data register on edge
1	0	0		1 *	Load high byte data register on edge
1	1	0		1	Load 12 – bit DAC register with data in low, mid, and high byte data registers. This control signal is level triggered

* MSB xxxx xxxx xxxx LSB
high mid low
byte byte byte

1 indicates logic High
0 indicates logic Low
x indicates don't care


indicates low to high transition

Table D·7

TRUTH TABLE FOR DAC

APPENDIX - E

E MODEL DETERMINATION

E.1 Overview

In this appendix the mathematical details required in model determination are presented. These are

* the effect of choosing a model order that is higher than the actual order,
* static gain calculation,
* improved techniques for model order reduction,
* the effect of noise on the model in the Z-domain and the S-domain, and
* a model trimming technique.

E.2 The effect of assuming a higher order model

If the order of the assumed model in the identification scheme is higher than the order of the plant, the following will occur:

(a) Zero coefficients will occur in the numerator polynomial of the transfer function when it is in the rational expansion form in the s-domain. This will take place until the ratio of the numerator to the denominator of the assumed model is equal to the ratio of the numerator to the denominator of the plant. Then,

(b) Pole-Zero cancellation will take place until the assumed model matches the plant. This is very clear when the model is in factored form.

Therefore from the theoretical point of view the choice of a higher model order does not affect the estimated model since it will match the real plant. This will be

illustrated by a simulated example.

In practice this does not occur, but instead negligible, rather than zero, coefficients appear in the numerator and poles and zeros that should cancel are not exactly equal. This is due to the fact that there are different noise sources in the system (e.g. quantisation effect). Practical data obtained from the controlled plant is used to show these concepts.

E.2.1 Simulation results

Frequency and step response tests previously carried out show that a first order model of the form:

$$\frac{k}{1 + \tau s} \qquad \text{(E.1)}$$

adequately describes the actuator system. This has a closed-loop time constant (τ) of approximately 4.3 sec and unity gain (k=1).

A simulation using this model was carried out and the data produced from this simulation were used to identify the transfer function. When a second order model was assumed in the identification scheme the following estimated transfer function was produced:

$$G(z) = \frac{0.022727278 + 0.0340663364\ z^{-1} + 0.01133915\ z^{-2}}{1 - 0.455626\ z^{-1} - 0.4762413\ z^{-2}}$$

Transforming G(z) to the S-domain then

$$G(s) = \frac{0.27253 + 0.00911\ s + 0.0\ s^2}{0.27253 + 1.18099\ s + 0.03918\ s^2}$$

Notice the occurrence of a zero coefficient in the numerator.

Hence

$$G(s) = \frac{0.27253 + 0.00911\ s}{0.03918(\ s + 0.23256)(\ s + 29.91012)}$$

$$= \frac{(\ 1 + 0.03343\ s\ )}{(\ 1 + 4.3\ s\ )(\ 1 + 0.03343\ s\ )}$$

Pole-zero cancellation within the transfer function reduces the order to match the original model.

$$G(s) = \frac{1}{1 + 4.3\ s}$$

This shows that (in theory) the assumption of a higher order model in the identification scheme does not affect the estimated model since it will match the real plant.

E.2.2 Practical results

In this test the data used was collected from the real plant as mentioned in section 3.4. When a second order model was assumed in the identification scheme the estimated transfer function became:

$$G(s) = \frac{0.27658 + 0.00894\ s + 0.00168\ s^2}{0.27845 + 1.17530\ s + 0.03969\ s^2}$$

Notice the coefficient of s^2 is not zero here, but is small and will be ignored.

Hence

$$G(s)= \frac{0.27658 + 0.00894\ s}{0.27845(\ 1 + 4.2\ s)(\ 1 + 0.034\ s)}$$

$$G(s)= \frac{0.27658(\ 1 + 0.0323\ s)}{0.27845(\ 1 + 4.2\ s)(\ 1 + 0.034\ s)}$$

It is clear that the pole and the zero which should cancel each other are not exactly equal. This is caused by a variety of noise sources in the system (e.g. quantisation effect). The transfer function, assuming pole/zero cancellation and ignoring the difference, is

$$G(s)= \frac{0.993}{1 + 4.2\ s}$$

E.3 Static gain

The static gain of a discrete transfer function G(z) is its value when z=1. Let G denote the static gain, therefore

$$G = \frac{\sum_{i=0}^{m} a_i}{\sum_{j=0}^{n} b_j} \qquad \text{(E.2)}$$

where m and n are the order of the numerator and the denominator respectively.

E.4 Techniques to improve Pole/Zero cancellation

The pole/zero cancellation technique proposed by Soderstrom produces a reduced order model that is incorrect by a scaling factor and hence give rise to a static gain difference between the original model and the reduced model. Several solutions to this problem are introduced. These are;

* Scale the reduced order model
* Retain static gain during optimisation
* Amend a zero or a pole

E.4.1 Scale the reduced order model

The static gain of both the original and the reduced order models are calculated using Eq.E.2. The proposed improvement in the pole/zero cancellation method is to multiply the reduced order transfer function $G_r(z)$ by the ratio of the static gain of the original model to the static gain of the reduced order model. Therefore the new model is

$$G_{new}(z) = \frac{G_o}{G_r} G_r(z)$$

where G_o and G_r are the static gain of the original and the reduced models respectively.

E.4.2 Retain static gain during optimisation

(a) Force a pole to have specific value

The static gain of the original model is determined using Eq.E.2. The objective is to have the static gain of both the original and the reduced models equal. This is achieved as follows. Let

$$G_r = \frac{\sum_{i=0}^{m} a_i}{\sum_{j=0}^{n} b_j} = G_o \qquad \text{(E.3)}$$

then

$$G_o \sum_{j=0}^{n} \tilde{b}_j = \sum_{i=0}^{m} \tilde{a}_i$$

$$\sum_{j=0}^{n} \tilde{b}_j = \frac{1}{G_o} \sum_{i=0}^{m} \tilde{a}_i$$

and

$$1 + \tilde{b}_1 = \frac{1}{G_o} \sum_{i=0}^{m} \tilde{a}_j - \sum_{j=2}^{n} \tilde{b}_j$$

$$\tilde{b}_1 = \frac{1}{G_o} \sum_{i=0}^{m} \tilde{a}_j - \sum_{j=2}^{n} \tilde{b}_j - 1$$

This value is assigned to b_1 every iteration during the optimisation procedure to insure the retainment of the static gain.

(b) Force a zero to have specific value

The above procedure is repeated but this time $\tilde{a}_0$ is factored out and assigned the value

$$\tilde{a}_0 = G_0 \sum_{j=0}^{n} \tilde{b}_j - \sum_{i=1}^{m} \tilde{a}_j$$

E.4.3 Amend a zero or a pole

The pole/zero cancellation technique is used to produce the reduced order model and then either a zero or a pole is amended. This is done by changing the value of a_0 to be

$$\tilde{a}_0 = G_0 \sum_{j=0}^{n} \tilde{b}_j - \sum_{i=1}^{m} \tilde{a}_j$$

or replacing the value of b_1 by

$$\tilde{b}_1 = \frac{1}{G_0} \sum_{i=0}^{m} \tilde{a}_j - \sum_{j=2}^{n'} \tilde{b}_j - 1$$

E.5 Effect of negligible coefficients

The simulation results in section E.2.1 showed that zero coefficients should appear in the numerator of the estimated transfer function until its structure matches the actual model. Practical results in section E.2.2 showed that negligible coefficients will appear instead of zero coefficients due to noise. These coefficients should be eliminated to match the structure of the actual plant.

Assume a first order transfer function of the form

$$G(s) = \frac{k(1 + \epsilon s)}{(1 + T_1 s)} \qquad \text{(E.4)}$$

where ϵ is a small coefficient (negligible) due to noise in the system. The aim here is to show how this coefficient transfers to the Z-domain. Using the Bilinear Z-transform method then

$$G(z) = \frac{k\left(1 + \epsilon \frac{2}{T} \frac{(1 - z^{-1})}{(1 + z^{-1})}\right)}{1 + T_1 \frac{2}{T} \frac{(1 - z^{-1})}{(1 + z^{-1})}}$$

$$= \frac{k\,T(1 + z^{-1}) + 2\,k\,\epsilon(1 - z^{-1})}{(1 + z^{-1}) + 2\,T_1(1 - z^{-1})}$$

$$= \frac{k\,T + 2\,k\,\epsilon - 2\,k\,\epsilon\,z^{-1} + k\,T\,z^{-1}}{1 + z^{-1} + 2\,T_1 - 2\,T_1\,z^{-1}}$$

and

$$G(z) = \frac{k(T + 2\epsilon) + k(T - 2\epsilon)\,z^{-1}}{(1 + 2T_1) + (1 - 2T_1)\,z^{-1}} \qquad \text{(E.5)}$$

It is shown in Eq.E.5 that the noise coefficient contributes in every coefficient of the numerator in the Z-domain. Therefore, it is not possible to eliminate this coefficient if the transfer function is in the Z-domain, while it is possible in the S-domain since it will be a separate coefficient.

E.6 Model trimming

The model determined using system identification as shown in Chapter-3 is in the Z-domain. The pole/zero cancellation method tests for pole/zero cancellation in the Z-domain but does not test for negligible coefficients which should appear if the model order is over estimated (Sec.E.2). The controlled plant is continuous, for more accurate model determination we have to trim the produced model to eliminate all small coefficients and match the actual structure of the plant. This is done in the S-domain for the reason mentioned in section E.5.

(a) The problem formulation

The objective of this method is to test for negligible coefficients in the numerator of the transfer produced from model order reduction process.

The problem that is considered can be formulated as follows: The two polynomials that form the plant transfer function (A/B) are

$$A(z^{-1}) = a_0 + a_1 z^{-1} + \ldots a_{na} z^{-na}$$

$$B(z^{-1}) = 1 + b_1 z^{-1} + \ldots b_{nb} z^{-nb}$$

where the values of parameters are the ones produced in the previous test (section E.4). The technique takes into account the uncertainty of the parameters estimated. Therefore, the covariance matrix (P) of these parameters is used. The problem now is to test whether the polynomial A has small coefficients or not. The test is carried out for M coefficients by starting with M=1, repeat the test for M=2,3,..., etc. as long as small coefficients are found.

(b) The criterion

Let the n-dimensional vector $\hat{\underline{x}}$ consist of the estimated values of the coefficients of the polynomials $\hat{A}$ and $\hat{B}$. Introduce a vector $\underline{x}$ that has the same dimension as $\hat{\underline{x}}$ corresponding to two polynomials A and B. The problem now is to look for a vector $\underline{x}$ in the same domain of $\hat{\underline{x}}$ such that the corresponding polynomials (A and B) have at least N common factors.

The technique achieves this by minimising a cost function $J(\underline{x})$ of the form

$$J(\underline{x}) = (\underline{x} - \hat{\underline{x}})^T P^{-1} (\underline{x} - \hat{\underline{x}}) \qquad \text{(E.6)}$$

(c) The algorithm

Introduce the polynomials $\tilde{A}(z^{-1})$, $\tilde{B}(z^{-1})$ and $\tilde{D}(z^{-1})$ where

$$\tilde{A}(z^{-1}) = \tilde{a}_0 + \tilde{a}_1 z^{-1} + \ldots + \tilde{a}_{na-M} z^{-(na-M)}$$

$$\tilde{B}(z^{-1}) = \tilde{b}_0 + \tilde{b}_1 z^{-1} + \ldots + \tilde{b}_{nb} z^{-nb}$$

$$\tilde{D}(z^{-1}) = 1 + 10^{-4} z^{-1} + \ldots + 10^{-4*M} z^{-M}$$

where M is the number of small coefficients. Consider the polynomials $A(z^{-1})$ and $B(z^{-1})$ have the following form

$$A(z^{-1}) = \tilde{A}(z^{-1}) \tilde{D}(z^{-1}) ,$$

$$B(z^{-1}) = \tilde{B}(z^{-1}).$$

The coefficients of the two polynomials ($\tilde{A}$, and $\tilde{B}$) are collected in a vector $\underline{y}$. Thus $\underline{x}$ can be written as a function of $\underline{y}$, $\underline{x}=f(\underline{y})$. Then the optimisation problem is to find the global minimum without constraints of

$$V[f(\underline{y})] = [f(\underline{y})- \hat{\underline{x}}]^T P^{-1} [f(\underline{y})- \hat{\underline{x}}]$$

The resulting coefficients of $\tilde{A}$ and $\tilde{B}$ represent the coefficients of the trimmed model.

This is a non-linear optimisation procedure and hence may yield several minima, therefore the selection of the initial values of the $\underline{y}$ vector is important. A reasonable set of initial values can be found by using the values of the reduced model.

The quasi-Newton method is used to handle the optimisation process involved in model trimming.

REFERENCES

CHAPTER-1

1.1 Goodwin, G. and Sin, K. "Adaptive Filtering Prediction and Control", Prentice-Hall, 1984.

1.2 Noton, M." Modern Control Engineering", Pergamon Press, 1972.

1.3 Franklin, G., Powell, J. and Emami-Naeini, A. "Feedback Control of Dynamic Systems", Addison-Wesley, 1986.

1.4 Isermann, R. "Digital Control Systems", Springer-Verlag, 1982.

1.5 Warwick, K. "Further Development in Self-Tuning Control", Self-Tuning and Adaptive Control: Theory and Applications, ed. C.J. Harris and S.A. Billings, IEE Control Engr. Series 15, 2nd edition, 1985. 332-360.

1.6 Clarke, D. "Model reference and pole-placement self-tuners", Optimal Control Applications and Methods, vol.3, 1982, 323-335.

1.7 Cegrell, T. and Hedqvist, T. "Successful adaptive control of paper machine", Automatica, vol.11, 1975, 53-59.

1.8 Buccholt, F. and Kummel, M. (1979). "Self-tuning control of a pH-neutralization process. Automatica 15, 665.

1.9 Wertz, V. "Application of adaptive predictive control for the bottom temperature of a glass furnace", 2nd IFAC Workshop on Adaptive System in Control and Signal Processing, 1986, 113.

1.10 Daley, S. "A study of a fast self-tuning control algorithm", Proc. Instn. Mech Engrs, vol.200, No.C6, 1986, 425-430.

1.11 Hope, O. and Sheirah, M. "Digital implementation and test results of a self-tuning speed regulator", CAN. ELEC. ENG. J., vol 6, 1981, 9-15.

1.12 Jones, A. and Porter, B. "Expert tunners for PID controllers", Proc. IASTED Conference on Computer-Aided Design and Applications, Paris. June 1985.

1.13 Numerical Analysis Group, Mayfield House, 256 Banbury Rd., Oxford OX2 7DE, U. K.

1.14 Integrated Systems Inc., 101 University Ave., Palo Alto, CA 94301- 1695, isi Matrixx UD.

1.15 Ismail, Z. M." Microprocessor Control of Electro-Mechanical Actuator", Ph.D. thesis, 1986, Loughborough University of Technology.

1.16 "The use of personal computers in control systems analysis", IEE Colloquium, 23rd May 1986, London. Colloquium digest No. 1986/83.

CHAPTER-2

2.1 MacFarlane, A.G. "The Development of Frequency Response Methods in

Automatic Control", IEEE Trans. on Automatic Control, April 1979.

2.2 Hort, W. "Die Entwicklung des Problems der stetigen Kraftmaschinen - Regelung nebst einem Versuch der Theorie unstetiger Regelungsvorgange", Zeitschrift Math. Phys., vol. 50, 1904, 233-279.

2.3 Von Mises, R. "Regulierung des Maschinenganges", Encyk. Math. Wissenschaften, vol.4, part 2, article 10, 1911, 254-296.

2.4 Tolle, M. "Die Regelung der Kraftmaschinen", Berlin: Springer, 1905.

2.5 Trinks, W. "Governors and the Governing of Prime Movers", New York: Van Nostrand, 1919.

2.6 Minorsky, N. "Directional stability of automatically steered bodies", J. Amet. Soc. Naval Eng., vol.42, 1922, 250-309.

2.7 Hazen, H.L. "Theory of servomechanisms", J. Franklin Inst., vol.218, 1934, 279-331.

2.8 Minorsky, N. "Automatic steering tests", J. Amer. Soc. Naval Eng., vol.42, 1930, 285-310.

2.9 Callender, A. et al."Time lag in a control system", Phil. Trans. Roy. Soc. London, vol.235A, 1936, 415-444.

2.10 Franklin, G., Powell, J. and Emami, A. "Feedback Control of Dynamic

Systems", Addison-Wesley, 1986.

2.11 Ziegler, J. and Nichols, N. "Optimum settings for automatic controllers", Trans. ASME, vol.64, 1942, 759-769.

2.12 Sage, A. "Linear Systems Control", Pitman, 1978.

2.13 Harris, H. "The analysis and design of servomechanisms", OSRD Rep. No. 454, 1942.

2.14 James, H. Nicholls, N. and Phillips, R. "Theory of Servomechanisms", Ney York: McGraw-Hill, 1947.

2.15 Linvill, W. "Sampled-data control system studies through comparson of sampling with amplitude modulation", AIEE Trans., vol.70, part II, 1951, 1779-1788.

2.16 Tsypkin, Ya. Z. "Frequency method of analysing intermittent regulating systems", in Frequency Response, R. Oldenburger. Ed. New York: Macmillan, 1956.

2.17 Ragazzeni, J. and Franklin, G. "Sampled-Data Control Systems", New York: McGraw-Hill, 1958.

2.18 Jury, E. "Sampled-Data Control Systems", New York: Wiley, 1958.

2.19 Jury, E. "Theory and Application of the Z-Transform Method", New York:

Wiley, 1964.

2.20 Freeman, H. "Discrete Time Systems", New York: Wiley, 1965.

2.21 Wiener, N. "Generalized harmonic analysis", Acta Math., vol.55, 1930, 117-258.

2.22 Astrom, K. "Process control - Past, present, and future", IEEE Control System Magazine, Aug. 1985, 3-10.

2.23 Horowitz, I. "History of personal involvement in feedback control theory", IEEE Control System Magazine, Nov. 1984, 22-23.

2.24 Truxal, J. "Automatic Feedback Control System Synthesis", Ch.6, McGraw-Hill, 1955.

2.25 Herwald, S. "Recollections of early development of servomechanism and control systems", IEEE Control System Magazine, Nov. 1984, 29-32.

2.26 Astrom, K."Design Principles for self-tuning regulators", in Methods and Applications in Adaptive Control, ed. by Unbehauen, H., Springer-Verlag, 1980.

2.27 Clarke, D. "Introduction to self-tuning controllers", in Self-Tuning and Adaptive Control, Ed. by Harris, C. and Billings, S., IEE series 15, 1981, 36-69.

2.28 Mahmoud, M.S. and Singh, M. "Discrete Systems", Springer- Verlag, 1984.

2.29 Narendra, K. "Recent development in adaptive control", in Method and Applications in Adaptive Control, Ed. by Unbehauen, H., Springer-Verlag, 1980.

2.30 Jacobs, O. "Introduction to adaptive control", in Self-Tuning and Adaptive Control, Ed. by Harris, C. and Billings, S., IEE series 15, 1981.

2.31 Landau, Y. "Adaptive Control - Model Reference Approach", Dekker, 1979.

2.32 Ljung, L. and Landau, I, MRAC sys. and self-tuning regulators - some connections", Proc. 7th IFAC Congress, 1978, vol.3, 1973-1980.

2.33 Narendra, K. "Stable discrete adaptive control", IEEE Trans. Automatic Control, AC-24, 1980, 456-460.

2.34 Gawthrop, P. "Some interpretations of the self-tuning Controller", Proc. IEE, vol.124, No.10, 1977, 889.

2.35 Goodwin, G. and Sin, K. "Adaptive Filtering Prediction and Control", Prentice-Hall, 1984.

2.36 Wellstead, P. and Zanker, P. "Techniques of self-tuning", Optimal Control Applications and Methods, vol.3, 1982, 305-322.

2.37 Isermann, R. "Parameter adaptive control algorithms - a tutorial", Automatica, vol.18, 1982, 513.

2.38 Clarke, D. "Model reference and pole-placement self-tuners", Optimal Control Applications and Methods, vol.3, 1982, 323-335.

2.39 Cegrell, T. and Hedqvist, T. "Successful adaptive control of paper machine", Automatica, vol.11, 1975, 53-59.

2.40 Buccholt, F. and Kummel, M. (1979). "Self-tuning control of a pH-neutralization process. Automatica 15, 665.

2.41 Wertz, V. "Application of adaptive predictive control for the bottom temperature of a glass furnace", 2nd IFAC Workshop on Adaptive System in Control and Signal Processing, 1986, 113.

2.42 Daley, S. "A study of a fast self-tuning control algorithm", Proc. Instn. Mech Engrs, vol.200, No.C6, 1986, 425-430.

2.43 Jones, A. and Porter B. "Expert tuners for PID controllers", Proc IASTED Conference on Computer-Aided Design and Application, Paris, June 1985.

2.44 Ljung, L. and Soderstrom, T. Theory and Practice of Recursive Identification, MIT press, 1983.

2.45 Unbehauen, H. "Systematic design of discrete model reference adaptive systems", in Self-Tuning and Adaptive Control, IEE series 15, 1981, 166-191.

2.46 Warwick, K. "Further development in self-tuning control", in Self-Tuning and Adaptive control: Theory and Applications, ed. C.J. Harris and S.A.

Billings, IEE Control Engr. Series 15, 2nd edition, 1985, 332-360.

CHAPTER-3

3.1 Astrom, K., Wittenmark, B. "Computer-Controlled Systems", Prentcie-Hall, 1984.

3.2 Davis, M. and Vinter, R. "Stochastic Modelling and Control", Chapman and Hall, 1985.

3.3 Franklin, G. and Powell, J. "Digital Control of Dynamic Systems", Addison-Wesley, 1980.

3.4 Franklin, G., Powell, J. and Emami, A. "Feedback control of Dynamic Systems", Addison-Wesley, 1986.

3.5 Leigh, J., R. "Modelling and Simulation", IEE topics in control series 1, 1983.

3.6 Ljung, L. and Soderstrom, T. "Theory and Practice of Recursive Identification", MIT, 1983.

3.7 Schwartzenbach, J. and Gill, K. "System Modelling and Control", 2nd ed., Arnold, 1984.

3.8 Sinha, N. and Kuszta, B. "Modelling and Identification of Dynamic systems, Van Nostrand Reinhold, 1983.

3.9 Soderstrom, T. "Test of Pole-Zero cancellation in estimated models", Automatica, Vol.11, 1975, 537-541.

CHAPTER-4

4.1 Astrom, K. "Simple Self-Tuners I". Internal report, LUTFD2, Lund Institute of Technology, Department of Automatic Control, 1979.

4.2 Astrom, K. and Wittenmark, B. "Computer-Controlled Systems Theory and Design", Prentice-Hall, 1984.

4.3 Clarke, D. "Model Following and Pole-Placement Self-Tuners", Optimal Control Applications & Methods. Vol 3, 323-335, 1982.

4.4 Doebelin, E. O. "Control System Principle and Design", John-Wiley, 1985.

4.5 Franklin, G., Powell, J. and Emami-Naeini, A. "Feedback Control of Dynamic Systems", Addison-Wesley, 1986.

4.6 Goodwin, G. and Sin, K. "Adaptive Filtering Prediction and Control", Prentice-Hall, 1984.

4.7 Houpis, C. and Lamont, G. "Digital Control Systems", McGraw-Hill, 1985.

4.8 Isermann, R. "Digital Control Systems", Springer-Verlag, 1982.

4.9 Katz, P. "Digital Control Using Microprocessors", Prentice-Hall, 1981.

4.10 Kuo, B. "Digital Control Systems", Holt Saunders, 1980.

4.11 Leigh, J. R. "Applied Digital Control", Prentice-Hall, 1985.

4.12 Palm III, W. "Modelling, Analysis and Control of Dynamic Systems", John Wiley, 1983.

4.13 Phillips, C. and Nagle, H. "Digital Control System Analysis and Design", Prentice-Hall, 1984.

4.14 Shinners, S. "Control System Design", John-Wiley, 1964.

4.15 Warwick, K. "Further Development in Self-Tuning Control", Self-Tuning and Adaptive Control: Theory and Applications, ed. C.J. Harris and S.A. Billings, IEE Control Engr. Series 15, 2nd edition, 1985. 332-360.

4.16 Wittenmark, B. "A Self-Tuning Regulator", Internal Report, Lund Institute of Technology, Division of Automatic Control.

4.17 iAPX 86,88 User's Manual, Intel, 1981.

CHAPTER-5

5.1 Astrom, K. "Self-Tuning Regulators - Design Principles and Applications", Application of Adaptive Control, 1-68.

5.2 Astrom, K. "Theory and Application of Adaptive Control - A survey". Automatica, Vol.19, 1983, 471.

5.3 Clarke, D. "Introduction to Self-Tuning Controllers", in Self-Tuning and Adaptive Control: Theory and Applications, ed. C.J. Harris and S.A. Billings, IEE Control Engr. Series 15, 1985, 36-71.

5.4 Clarke, D. "The Application of Self-Tuning Control". Trans. Inst. M.C., Vol 5, 1983, 59-69.

5.5 Goodwin, G. and Sin, K. "Adaptive Filtering Prediction and Control", Prentice-Hall, 1984.

5.6 Isermann, R. "Parameter Adaptive Control Algorithms. a Tutorial". Automatica, Vol 18, 1983, 513-528.

5.7 Isermann, R. "Digital Control Systems", Springer-Verlag, 1982, 384-386, 414-420.

5.8 Ogata, K. " Modern Control Engineering". Prentice-Hall, 1970, 791-795.

5.9 Soderstrom, T., Ljung, L. and Gustavsson, I. "Comparative Study of

Recursive Identification Methods", Internal report, Lund Institute of Technology, 1974.

5.10 Soderstrom, T. and Ljung, L. "Theory and Practice of Recursive Identification", MIT press, 1983.

5.11 Warwick, K. "Further development in self-tuning control", in Self-Tuning and Adaptive control: Theory and Applications, ed. C.J. Harris and S.A. Billings, IEE Control Engr. Series 15, 2nd edition, 1985, 332-360.

5.12 Wittenmark, B. "Stochastic Adaptive Control - A Survey". Int. J. Control, Vol 21, 1975, 705-730.

CHAPTER-6

6.1 "iAPX 86/88, 186/188 User's Manual, Hardware Reference", Intel Corporation 1985, 3065 Bower Avenue, Santa Clara, CA95051, USA, ISBN 0-917017-36-6.

6.2 RS-232C standard.

6.3 RS-422 standard.

CHAPTER-7

7.1 Gilbert, F. "Software Design and Development", Science Research Associates Inc., Chicago, 1983. ISBN 0-574-21430-5.

7.2 Jensen, R. W and Tonies, C. C. "Software Engineering", Prentice-Hall, 1979.

7.3 Jones, G. "Structured Programming Design", Hodder and Stoughton, 1985. ISBN 0-340-36448-3.

CHAPTER-10

10.1 Ismail, Z. M." Microprocessor Control of Electro-Mechanical Actuator", Ph.D. thesis, 1986, Loughborough University of Technology.

10.2 Dettmer, R."Digital signal processors", Electronics & Power, Feb. 1986, 124-128.

APPENDIX-A

A.1 Ismail, Z. M." Microprocessor Control of Electro-Mechanical Actuator", Ph.D. thesis, 1986, Loughborough University of Technology.

APPENDIX-B

B.1 Astrom,K., Introduction to Stochastic Control Theory, Acadimic press, 1970.

B.2 Astrom,K., and Wittenmark,B., Computer Controlled System, Prentice-Hall, 1984.

B.3 Franklin,G.,and Powell,D., Digital Control of Dynamic Systems, Addison-Wesley,1980.

B.4 Goodwin,G. and Sin,K., Adaptive Filtering Prediction and Control, Prentice-Hall, 1984.

B.5 Isermann,R., "Practical Aspects of Process Identification", Automatica, vol.16, pp 575-587, 1980.

B.6 Isermann,R. et al, "Comparison Of Six On-Line Identification and Parameter Estimation Methods", Automatica, vol.10, pp 81-103, 1974.

B.7 Ljung,L., and Soderstrom,T., Theory and Practice of Recursive Identification. MIT press, 1983.

B.8 Soderstrom,T., Ljung,L. and Gustavsson,I, Comparative Study of Recursive Identification Methods, Internal report, Lund Institute of Technology, 1974.

B.9 Young, P. "Recursive approach to time series analysis", The Institute of Mathematics and its Applications, May 1974. 209-224.

B.10 Young, P. "Recursive Estimation and Time Series Analysis", Springer-Verlag, 1984.

B.11 Sakrison, G. "The use of stochastic approximation to solve the system identification problem", IEEE Trans. on Automatic Control, vol AC-12, 563-567.

B.12 Saridis, G. and Stein G."Stochastic approximation algorithms for discrete time system identification", IEEE Trans. on Automatic Control, vol AC-13, 515-523.

B.13 Tsypkin, Ya."Foundation of The Theory of Learning Systems", Academic Press, N.Y., 1973.

B.14 Strejec, V."Least squares parameter estimation", Automatica, vol 16, 535-550.

B.15 Fortiscue, T,"Implementation of self-tuning regulator with variable forgetting factor", Automatica, vol 17, 1981.

B.16 Tsypkin, Ya."Adaptive and Learning in Automatic Systems", Academic Press, N.Y., 1971.

B.17 Soderstrom, T."On the accuracy of identification and the design of identification experiments", Lund Institute of Technology, Internal report, Dec. 1974.

B.18 Talmon, J. and Van den Boom,"On the estimation of transfer fuction parameters of process and noise dynamics using a single stage estimator", Proc. 3rd IFAC Symposium on Identification and System Parameter

Estimation, the Hague, 1973.

B.19 Fuhrt, B."New estimation for the identification of dynamic processes", IBK Report, Institute Boris Kidric Vinca, Belgrade, 1973.

B.20 Kurz, H. and Isermann, R."Experimental comparison and application of various parameter-adaptive control algorithms", Automatica, vol 16, 1980, 117-133.

B.21 Saridis, G."Comparison of six on-line identification algorithms", Automatica, vol 10, 1974, 69-79.

B.22 Kurz, H. and Isermann, R."Methods for on-line proc. identification", 6th Conf. IFAC, 1975.

B.23 Plackett, R."Some theorems in least squares", Biometrika, vol 37, 1950, 149.

B.24 Young, P."The use of linear regression and related procedures for the identification of dynamic process", Proc. 7th IEEE Symposium on Adaptive Processes, UCLA.

APPENDIX-C

C.1 Astrom, K. "Simple Self-Tuners I". Internal report, LUTFD2, Lund Institute of Technology, Department of Automatic Control, 1979.

C.2 Astrom, K. and Wittenmark, B. "Computer-Controlled Systems Theory and Design", Prentice-Hall, 1984.

C.3 Clarke, D. "Model Following and Pole-Placement Self-Tuners", Optimal Control Applications & Methods. Vol 3, 323-335, 1982.

C.4 Doebelin, E. O. "Control System Principle and Design", John-Wiley, 1985.

C.5 Franklin, G., Powell, J. and Emami-Naeini, A. "Feedback Control of Dynamic Systems", Addison-Wesley, 1986.

C.6 Goodwin, G. and Sin, K. "Adaptive Filtering Prediction and Control", Prentice-Hall, 1984.

C.7 Houpis, C. and Lamont, G. "Digital Control Systems", McGraw-Hill, 1985.

C.8 Isermann, R. "Digital Control Systems", Springer-Verlag, 1982.

C.9 Katz, P. "Digital Control Using Microprocessors", Prentice-Hall, 1981.

C.10 Kuo, B. "Digital Control Systems", Holt Saunders, 1980.

C.11 Leigh, J. R. "Applied Digital Control", Prentice-Hall, 1985.

C.12 Palm III, W. "Modelling, Analysis and Control of Dynamic Systems", John Wiley, 1983.

C.13 Phillips, C. and Nagle, H. "Digital Control System Analysis and Design", Prentice-Hall, 1984.

C.14 Shinners, S. "Control System Design", John-Wiley, 1964.

C.15 Warwick, K. "Further Development in Self-Tuning Control", Self-Tuning and Adaptive Control: Theory and Applications, ed. C.J. Harris and S.A. Billings, IEE Control Engr. Series 15, 2nd edition, 1985. 332-360.

C.16 Wittenmark, B. "A Self-Tuning Regulator", Internal Report, Lund Institute of Technology, Division of Automatic Control.

C.17 iAPX 86,88 User's Manual, Intel, 1981.

C.18 Lancaster, D. "Active-Filter Cookbook", Howard W. Sams & Co., Inc., 1977.

APPENDIX-D

D.1 Cooling, J.E. "Analogue Data Acquisition Systems for Microcomputers, Lecture notes, Loughborough University of Technology, U.K.

D.2 Astrom, K. and Wittenmark, B. "Computer Controlled Systems", Prentice-Hall, 1984.

D.3 "IEEE P1000 specifications Draft 3.1".

D.4 "iAPX 86/88, 186/188 User's Manual, Hardware Reference", Intel Corporation 1985, 3065 Bower Avenue, Santa Clara, CA95051, USA, ISBN 0-917017-36-6.

D.5 "iAPX 86/88 User's Manual", Intel Corporation 1981, 3065 Bower Avenue, Santa Clara, CA95051, USA.

D.6 "Data-Acquisition Databook", Analogue Devices, 1982.

D.7 Cooling, J.E. "Microcomputer Systems, Analogue Output Signals", Lecture notes, Loughborough University of Technology, U.K.