Artificial intelligence in control of real dynamic systems.

Assilian, S

ARTIFICIAL INTELLIGENCE IN THE CONTROL OF REAL DYNAMIC SYSTEMS

BY

S. ASSILIAN

Thesis presented for the degree of
Doctor of Philosophy in the Faculty of
Engineering in the University of London.

August 1974

ABSTRACT

A real dynamic plant is used to compare, test and assess the present theoretical techniques of adaptive, learning or intelligent control under practical criteria. Work of this nature has yet to be carried out if "intelligent control" is to have a place in everyday practice.

The project follows a natural pattern of development, the construction of computer programmes being an important part of it.

First, a real plant - a model steam engine - and its electronic interface with a general purpose digital computer are designed and built as part of the project. A rough mathematical model of the plant is then obtained through identification tests.

Second, conventional control of the plant is effected using digital techniques and the above mentioned mathematical model, and the results are saved to compare with and evaluate the results of "intelligent control".

Third, a few well-known adaptive or learning control algorithms are investigated and implemented. These tests bring out certain practical problems not encountered or not given due consideration in theoretical or simulation studies. Alternatively, these problems materialise because assumptions made on paper are not readily available in practice. The most important of these problematic assumptions are those relating to computational time and storage, convergence of the adaptive or learning algorithm and the training of the controller. The human operator as a distinct candidate for the trainer is also considered and the problems therein are discussed.

Finally, the notion of fuzzy sets and logic is viewed from the control point and a controller using this approach is developed and implemented. The operational advantages and the results obtained, albeit preliminary, demonstrate the potential power of this notion and provide the grounds for further work in this area.

CONTENTS

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

PATTERN RECOGNITION AND ADAPTIVE CONTROL

CHAPTER 5

ADAPTIVE CONTROL OF THE STEAM ENGINE

CHAPTER 6

FUZZY LOGIC AND CONTROL

CHAPTER 7

FUZZY LOGIC CONTROL OF THE STEAM ENGINE

APPENDIX B

APPENDIX C

APPENDIX D

CHAPTER 1

INTRODUCTION

1.1 ARTIFICIAL INTELLIGENCE AND CONTROL: MOTIVATION

The study reported in this dissertation concerns the faculty of control engineering. It is only natural, therefore, to start by asking what is 'artificial intelligence' and what is implied by 'intelligent control'?

Basically, the scope of artificial intelligence (AI) embodies any study or research which has as its prime purpose the construction of either a computer programme (software) or a physical device (hardware) which exhibits intelligent behaviour (Feigenbaum and Feldman, 1963). The term 'intelligent' here implies not only a behaviour exhibiting the faculties of understanding and reasoning, according to its usual definition in the English language, but also one which is par excellence 'goal-seeking' through the acquisition of 'experience' and the process of 'learning'. In other words, implicit in the meaning of intelligence is the power of 'appropriate selection'. Some workers in the field of AI hold the strong view that such behaviour can be found only in humans, by virtue of their natural endowments, and therefore an 'intelligent machine' (software or hardware) cannot be constructed (Feigenbaum and Feldman, 1963). On the other hand, for whatever reason that has motivated it, much research has been done, and is still being done in the simulation of 'cognitive processes' and the construction of intelligent machines (Samuel, 1959, 1967; Ernst and Newell, 1969; Rosenblatt, 1962; Gaines and Andreae, 1966; Nilsson, 1969). For the control engineer it suffices to go just this far on the philosophical aspects of intelligent behaviour before turning attention to a more practical, and perhaps a more urgent question: (MOTIVATION) The applicability of the knowledge and techniques available in the field of AI to control problems.

If an intelligent machine is defined as a system which gathers information (experience), and processes it appropriately and with efficiency (learning), so as to achieve a high intensity of appropriate selection (control),

then the possibility of applying techniques of AI to control becomes obvious. It is misleading and perhaps unfortunate, however, when the expression 'intelligent control' is used sometimes in the literature according to the above definition (Fu, 1971). For, if appropriate selection is an act of intelligence, then is not a conventional controller intelligent?

1.2 THE MAIN OBJECTIVES OF THE STUDY

The applicability of AI techniques to control problems has been extensively studied in the past and many encouraging results have been reported in the literature (Widrow and Smith, 1964; Gaines and Andreae, 1966; Mendel, 1967; Fu, 1970). However, there is one obtrusive gap in these studies which is due to the general neglect of supplementing theoretical and simulation work with practical experimentation. It is indeed very natural in any science to develop the theory first and test the validity of the theory in a simulated environment next. But in engineering applications, it is also as important to have a third stage:(OBJECTIVE) Test for validity on the real system as a conclusive step. This constitutes the main objective of this study.

There are certain characteristics about process control which could make it unsuitable for the application of current AI techniques:

(a) Timing is critical and solutions must be reached in minutes or even seconds.
(b) The solution must be economical in terms of hardware cost (for example, the size of a computer-controller).
(c) Implementation of the controller must be quick and easy, (for example, the rate of convergence of the adaptive or learning control algorithms and the method of training these controllers - Chapter 4).

Investigation of the practical limitations arising out of the above three points constitutes the other objective of this study.

1.3 ACHIEVEMENTS AND CONTRIBUTION

In order to satisfy the first objective above, a real system had to be made available for experimentation. Unfortunately though, one is almost always faced with a problem here since real systems are not readily available when required for research work. The use of an industrial plant which is already in production is very unlikely for obvious reasons. The only alternative, therefore, was to have a laboratory-size plant.

Prior to the commencement of this work, another piece of research, which was designed on similar lines as the present, had just been completed at Queen Mary College (Carter et al, 1971). The experimental part of Carter's work was conducted on a toy steam engine which was found to be inadequate in many ways, and the need for a robuster and more sophisticated system had been shown to be essential for further work. Thus a new, model steam engine (photographs in Figs.2.3 and 2.4) was built in the workshop of Queen Mary College, with which the author had little to do apart from partaking in some aspects of its design. However, the design and building of the electronic interface between the steam engine and the digital computer, which simulated all the controllers considered in this study, was completed by the author. This was an achievement in its own right and inevitably a considerable amount of time was spent in order to bring it to its final, operating and reliable form.

Having thus made a start with a new system, a carefully planned programme of identification tests was completed next and a simple mathematical model of the steam engine was established. These tests evinced the non-linear nature of the system.

A major justification in considering the applicability of AI techniques in control is the situation where conventional control techniques are inadequate. Nevertheless, a simple conventional controller was designed and implemented in order to provide a benchmark against which the current state or power of AI control techniques could be assessed or measured.

The direct digital controller implemented achieved very satisfactory control of the steam engine, which was another achievement in its own right. Furthermore, this exercise proved the controlability of the steam engine, thus providing the thesis that learning controllers ought to be able to learn and achieve the same quality of control.

The investigation of AI control techniques was started by reconsidering the work of Carter (1971). An identical controller to his was implemented, but experimentation indicated that his procedure was too naive, since essentially it involves the building up of a memory where for every possible state of the plant an associated action is stored. Obviously, this is a trivial solution to the control problem with no generalising ability, and this part of the work is not described in this dissertation.

The relatively better established techniques of adaptive and learning control (Chapter 4) stimulated the implementation of such controllers next, but two special features, which were introduced on purpose into the steam engine, brought out certain difficulties that one can expect to encounter in a realistic situation. These features are the multi-variable nature of the plant, and the multi-valued (non-bang-bang) nature of the inputs to the plant. A critical survey of adaptive control techniques from the view of such systems is presented. The main difficulties that were encountered in the experimental part of the work can be summarised as follows (Assilian and Mamdani, 1974a):

(a) With human-supervised learning, very poor convergence is achieved with adaptive controllers, and the rate of convergence is unacceptably slow.

(b) With non-supervised learning, the specification of a useful performance criterion is difficult, and the storage required with this scheme is exorbitant (one of the intentional restrictions (Carter et al,1971) in this work was the use of a small digital computer - a PDP8/S with only 8K of core memory).

Little could have been done with non-supervised schemes in this study. However, the important conclusion with human-supervised machine learning is that the real obstacle is not having the means for linguistic communication between man and machine. The notion of 'fuzzy logic' (Zadeh, 1965) seemed to provide a solution in the attempt to overcome this obstacle. Thus a fuzzy logic controller was designed and implemented, which, surpassing all expectations, achieved comparable control of the steam engine with the direct digital controller (Assilian and Mamdani, 1974b; Mamdani and Assilian, 1974). This work is a distinct contribution; although the control applications of fuzzy logic have been suggested (Chang and Zadeh, 1972), to date this work represents the first attempt of an actual application in a real system.

1.4 CONTENTS OF CHAPTERS

The length of a chapter in this dissertation bears no relation to the amount of effort spent on the work described in that chapter. In general, the material which is presented is chosen either because it is not documented elsewhere, or because it is important in comparing and discussing this work in relation to other previous work.

Chapter 2 is devoted to a description of the complete steam engine control system. This includes the design of the electronic interface between plant and computer, and a brief description of the general software at the end of the chapter. An account of the modelling of the steam engine is also presented in this chapter, leading to a simple mathematical model of the plant.

Chapter 3 is concerned with the design and implementation of a conventional feedback controller for the steam engine, based on the mathematical model established in Chapter 2. Two simple controllers are considered; PI and PID controllers, and the so-called 'single-term' controller. The tuning of these controllers is described at the end of the chapter.

Chapter 4 is devoted to a review of the relevant earlier work carried out on adaptive control techniques. The chapter starts on some fundamental aspects of adaptive control and the relationship between pattern recognition and control. Two widely studied adaptive control techniques are reviewed; those based on the adaptive threshold logic element and those based on decision theory. The review is critical in nature as seen from the point of view of a real, multi-variable control system. Mathematics is kept to the essential minimum. The main training algorithms and main training modes for adaptive controllers are examined in detail.

In Chapter 5 the results of adaptive control of the steam engine are presented. Three different teachers are considered in the training of the adaptive controllers; the human operator, the conventional controller developed in Chapter 3, and the fuzzy logic controller developed later in the dissertation (Chapter 7). The problems encountered with each teacher are pointed out and discussed in detail. Some aspects of various linearly independent codes are also examined at the end of the chapter.

Chapter 6 describes the formulation of the general fuzzy logic controller. The chapter starts with certain formal definitions in fuzzy set theory, which are required in the implementation of such a controller. Central to the execution of the fuzzy logic control algorithm is the 'compositional rule of inference' which is also defined and discussed. Throughout this chapter, simple illustrative examples are given in order to clarify all the definitions and computations involved in the formulation and implementation of the controller.

In Chapter 7 the results of fuzzy logic control of the steam engine are presented and discussed. First, a non-interactive control algorithm is implemented and an efficient method of tuning this controller is described. Second, it is demonstrated that the fuzzy logic controller is equally suitable for implementing interactive, or more complex control policies.

Finally, in Chapter 8, a comparison of all the controllers considered in this research is presented. This is followed by a summary of the important conclusions drawn throughout the thesis, leading to suggestions and recommendations for future work.

CHAPTER 2

THE CONTROL SYSTEM - HARDWARE AND SOFTWARE

2.1 INTRODUCTION

It is noteworthy that the choice of the steam engine bears no relevance to the development of theoretical ideas in this research. In fact, outside the presentation of experimental results, all discussions are held without reference to any particular system, thus indicating the general applicability of the theory. Expressed otherwise, the steam engine serves only as a vehicle for the investigation of AI techniques in control.

The engine and boiler to be described were designed and built in Queen Mary College. The author had little to do with this part of the hardware. However, the design and building of the electronic interface between the plant and digital computer were completed by the author, and this exercise proved to be both painstaking and time consuming in promoting it to its final, operating form. A natural follow-up to this task was the identification or modelling of the yet unknown system. Various experiments were carried out mainly to test for linearity/non-linearity, and to produce a mathematical model which is required in the implementation of a conventional feedback controller. These tests also helped the author to familiarise himself thoroughly with the operational characteristics of the plant, an experience which was useful later in the development of trainable adaptive and 'fuzzy logic' controllers.

2.2 STEAM ENGINE-COMPUTER INTERFACE DESIGN

In order to keep the length of presentation within reasonable bounds, only a brief description of the interface is given in this section. Where it is deemed useful, certain details are included in Appendix A.

2.2.1 TOTAL CONTROL SYSTEM

A block-diagrammatic view of the complete system is given in Fig.2.1. The hybrid computer, a SOLARTRON HS7-6D, behaves as part of the interface between plant and digital controller, a PDP8/S digital computer. Due to its design specifications, the SOLARTRON is readily coupled to any PDP8 series computer allowing very convenient and easy communication with the digital computer for both logic and analogue signals. Equally useful are the real time clock and the twelve interrupt channels provided, which in this project are used mainly for timing sampling intervals and initiating special, 'panic-action' routines when the plant inadvertently enters 'dangerous states' (see Fig.2.12.)

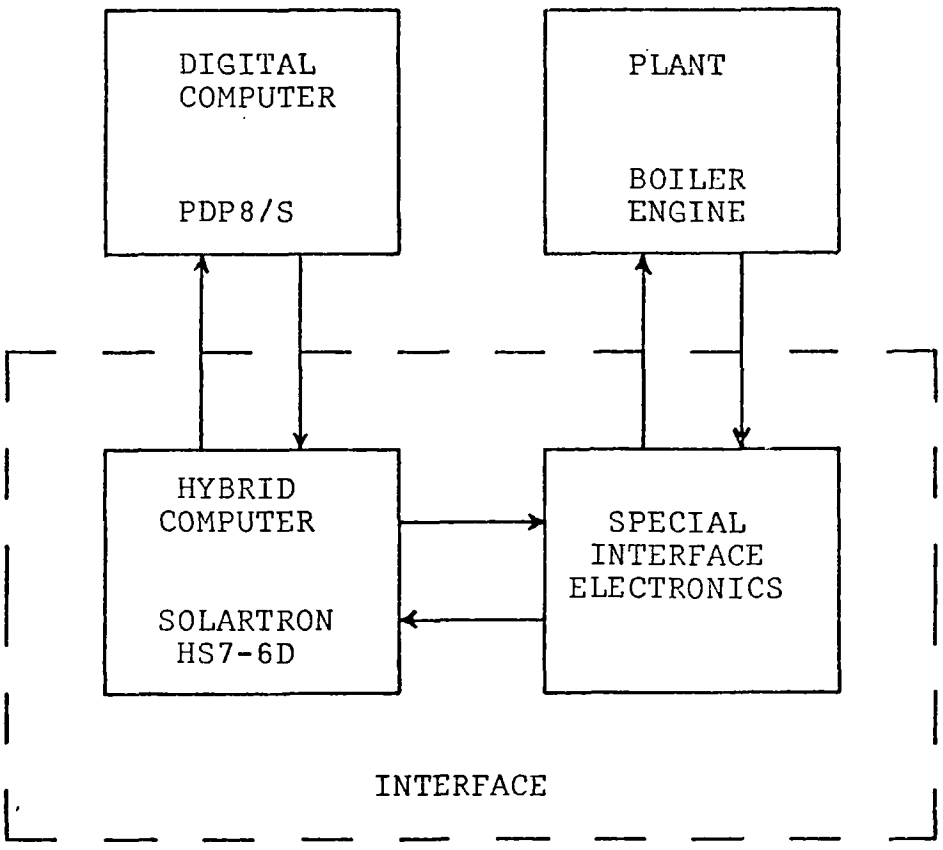


Fig. 2.1 The System

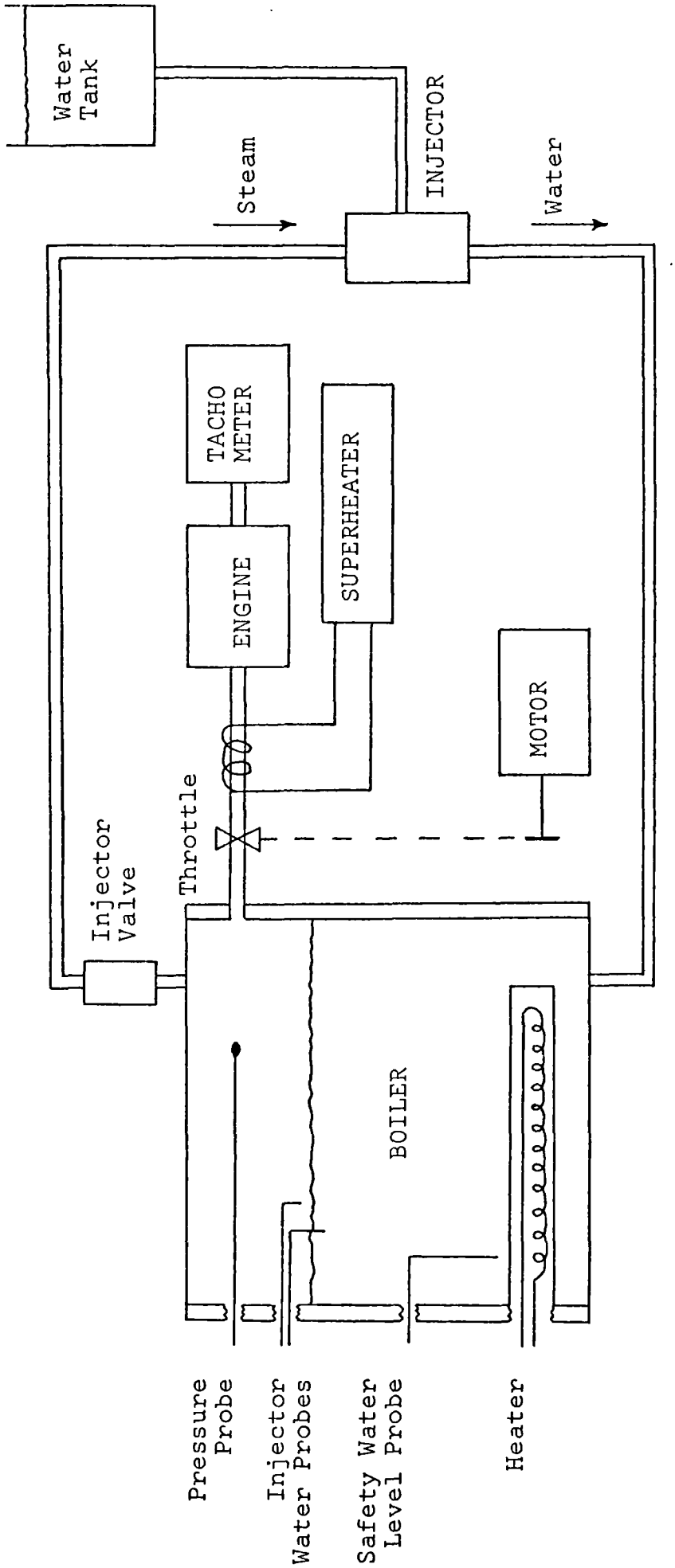


Fig. 2.2 The Plant

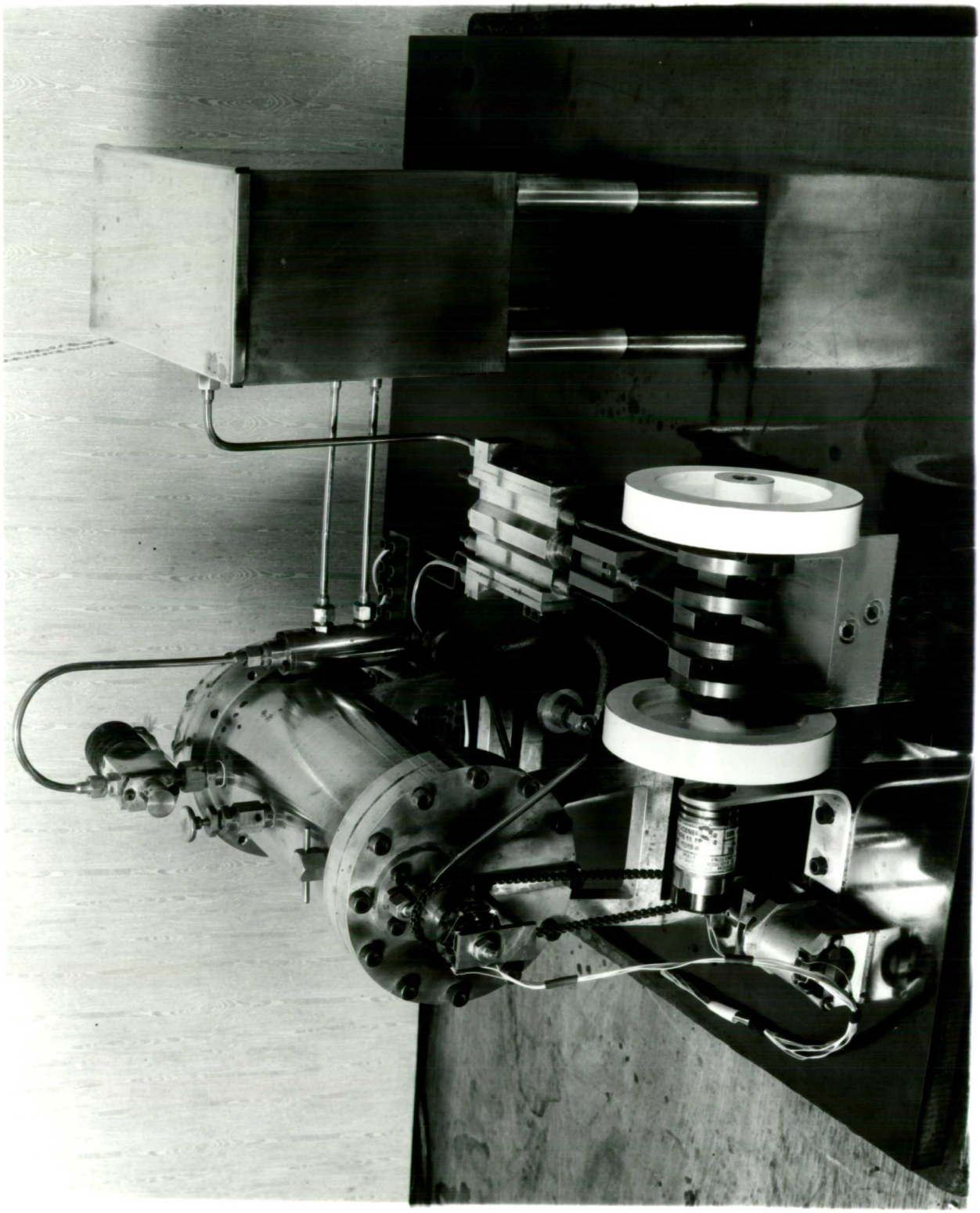


Fig. 2.3 The Steam Engine

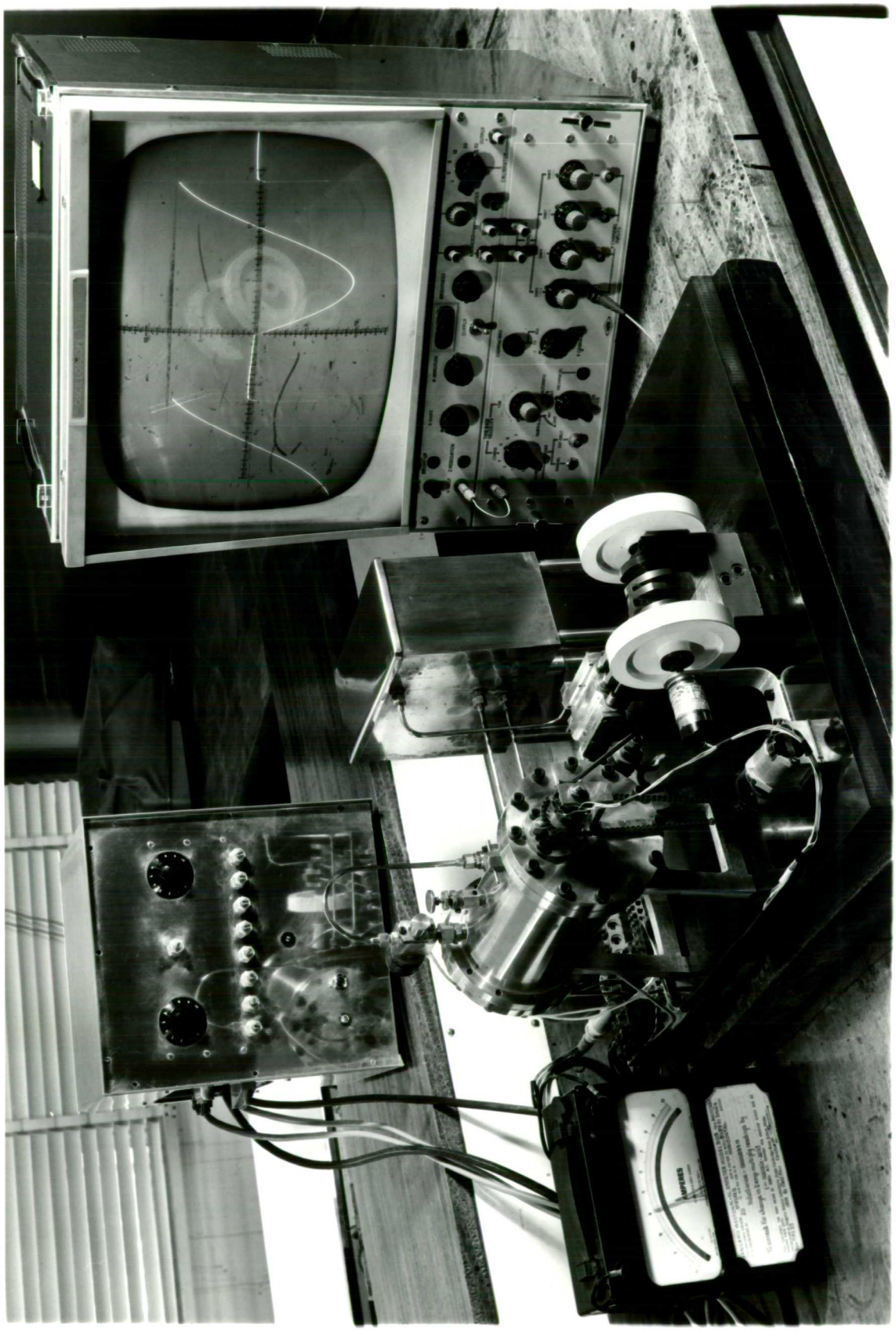


Fig. 2.4 The Steam Engine

A schematic view of the plant is shown in Fig.2.2, supplemented with two photographs in Figs.2.3 and 2.4. The plant comprises a steam engine-boiler combination. The main inputs to the plant are heat and throttle, with pressure and speed as outputs. Heat to the boiler is provided using two electric heaters and the pressure in the boiler is estimated from the resistance of a thermistor in contact with the steam. The speed of the engine is measured using a tachometer which is directly coupled to the crank rod of the engine. The speed is controlled via the throttle which is driven by a small electric motor. Of course, the speed can also be controlled by varying the heat input, since any heat change varies the pressure which in turn varies the speed. Similarly, the throttle can be used to vary the pressure; in fact, there is coupling between each input and each output as described in Section 2.3.

The special interface electronics, which is housed in the box seen behind the plant in Fig.2.4, performs the special functions required in this project which cannot be achieved on the hybrid computer alone. Besides providing for the abovementioned inputs and outputs, it carries out additional signal processing for safety purposes and for better running of the plant. A safety water level detector with its associated circuitry makes sure the boiler never runs completely out of water lest the heaters are caused to burn out. The steam operated injector with its associated circuitry keeps the water level in the boiler within two limits close to each other, corresponding to the two level detectors shown in Fig.2.2. In fact, the circuitry performing this task is independent of the digital computer. Finally, a superheater is incorporated to dry the steam in an attempt to increase the running efficiency of the engine. Although the degree of superheat is kept constant in this project, it could be made another controlled variable by additional circuitry similar to that of the heater variable.

2.2.2 HEATER CONTROL

2.2.2.1 The Circuitry

The heat input is designed to be discrete because of the nature of the project and 32 levels or steps are provided from no power to full power. A schematic view of the complete heater control circuit is shown in Fig.2.5.

Central to the control scheme is a 4-bit binary counter which indicates the present or current level of input. Clearly such a counter can only indicate 16($=2^4$) levels, however, by separating the control of the positive-half cycle of the mains input from the negative-half cycle, it is not necessary to provide for each step of the full 32 levels. The separation of control of the two half-cycles, which is actually necessitated by the operational characteristics of the silicon controlled rectifier (SCR), is achieved by using a different SCR to fire each half. Thus, the first 15 steps are obtained by controlling the positive-half cycle, keeping the negative-half cycle switched off. At step 16, half power, the negative-half cycle is fully switched on via a Schmitt trigger and the positive-half cycle initiated to zero. For steps greater than 16, the positive-half cycle is controlled identical to the first 15, only this time having the negative-half cycle switched on.

The above scheme simplifies greatly the commands to be given by the digital computer in order to manipulate the heat input. From the point of view of the digital computer, heater control is achieved using three commands; 'UP n steps', 'DOWN n steps' and 'CLEAR' counter. Every 'UP' or 'DOWN' command corresponds to putting a logic pulse on the appropriate line, n steps obviously requiring a train of pulses. The 'CLEAR' command is provided for initiating to zero with a single step instead of counting down. Command frequencies of up to 30MHz can be obtained because of the fast switching time of integrated circuits. See Appendix A for further details.

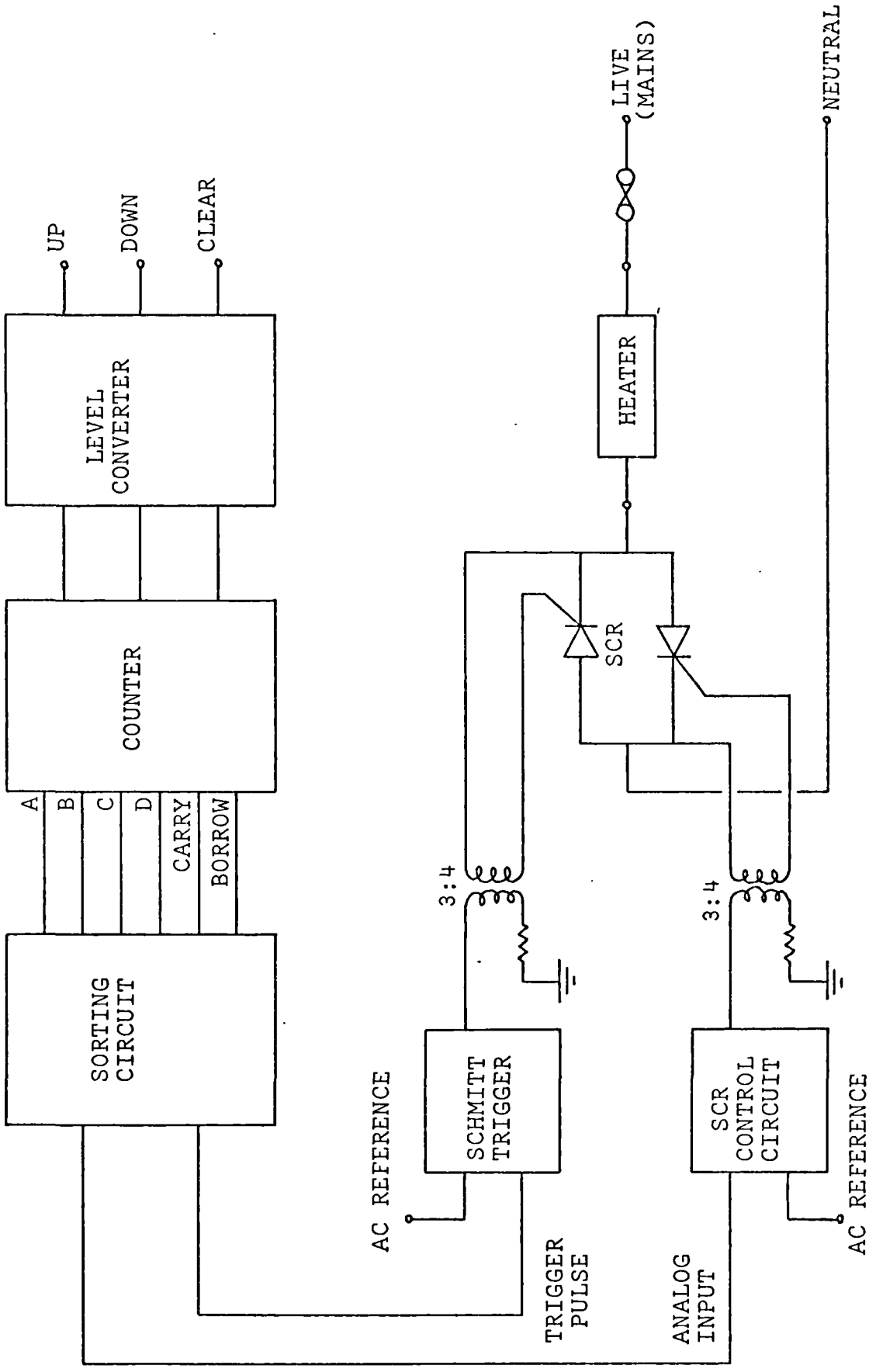


Fig. 2.5 Heater Control

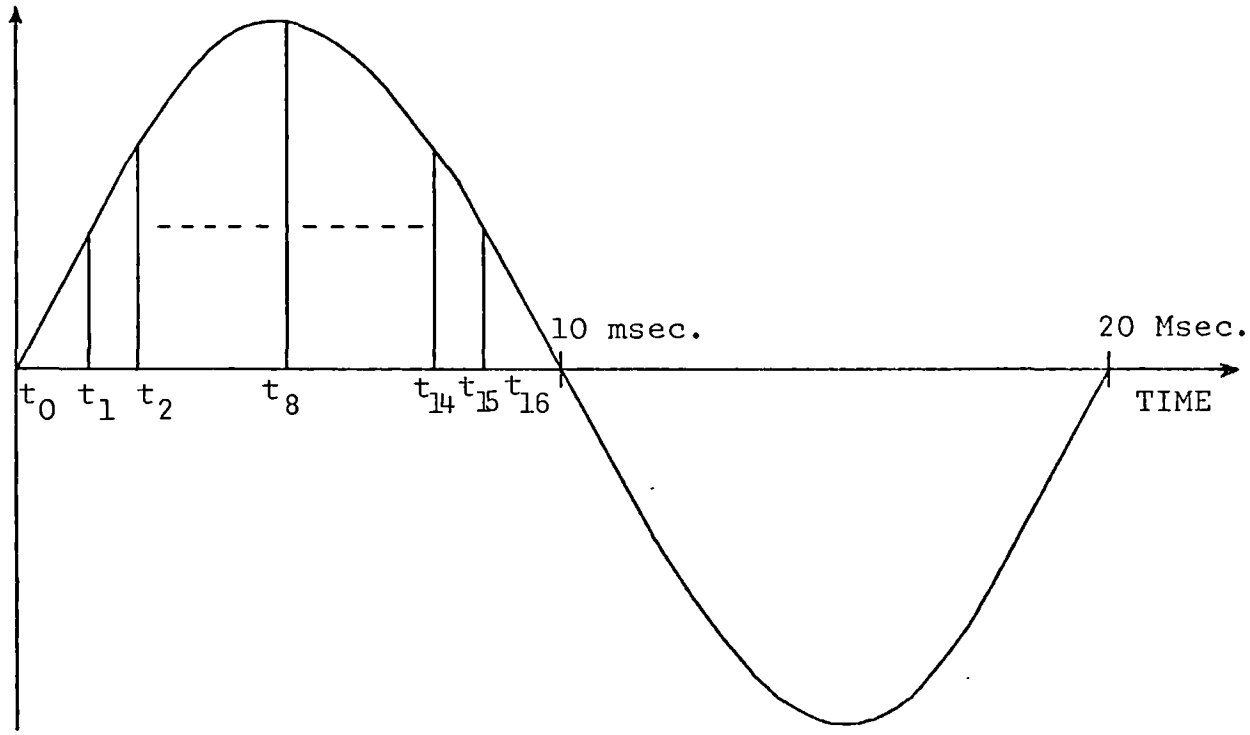


Fig. 2.6 Adjustment of Heater Steps

n	t_n(msec.)	n	t_n(msec.)
1	1.60	9	5.40
2	2.30	10	5.80
3	2.85	11	6.33
4	3.33	12	6.66
5	3.77	13	7.15
6	4.20	14	7.70
7	4.60	15	8.40
8	5.00	16	10.00

Table 2.1 Values for t_n in Fig. 2.6

2.2.2.2 Adjustment of Heat Steps

The 32 steps on the heat input are adjusted to give equal power per step. This is achieved by adjusting appropriately the firing angle of the SCR for each step, as described in Appendix A.

The power input at any instant is given by the area under the sinewave shown in Fig.2.6 (see also Fig.2.4). A simple integral equation gives the values for t_1, t_2,---,t_{16} in terms of time as follows.

$$\int_{t_o=0}^{t_{n+1}} \sin\omega t \, dt - \int_{t_o=0}^{t_n} \sin\omega t \, dt = \frac{1}{16} \int_{t_o=0}^{t_{16}=1/100} \sin\omega t \, dt \qquad (2.1)$$

Simple manipulation of Eqn.(2.1) yields

$$\cos\omega t_n - \cos\omega t_{n+1} = 1/8 \qquad (2.2)$$

whereupon substituting values for n from 0 to 15 gives the values for t_n in Table 2.1.

2.2.3 THROTTLE CONTROL

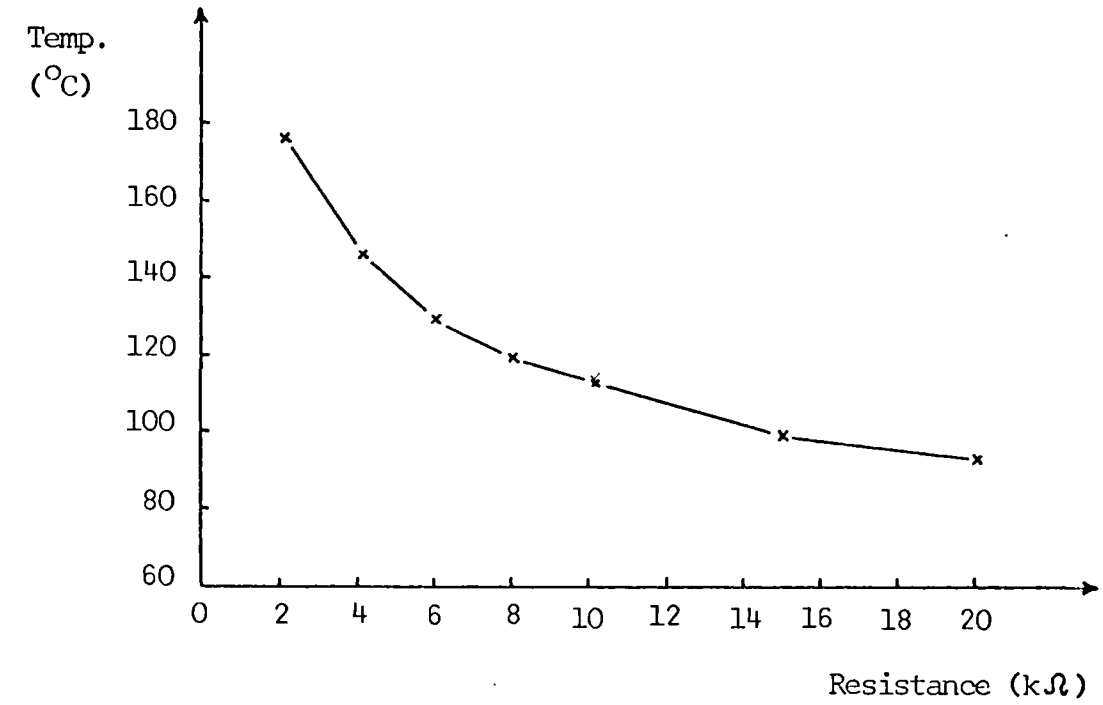
The throttle input is designed to be discrete for the same reason given in the design of the heater control. 10 steps are provided from the fully-shut position to the fully-open position.

The throttle is driven via a small electric motor as shown in Fig.2.2. Basically, the throttle is just a small brass plate on which 10 holes, increasing equally in area, are drilled on a circular line and spaced out 30^o from each other. Every step then amounts to turning the throttle through 30^o in the appropriate direction, left or right, to bring the corresponding hole on the plate against the opening on the boiler.

The motor drive and throttle circuits are also designed to simplify the commands to be given by the digital computer to achieve throttle control. Three commands are provided for similar to the heat variable; 'LEFT n steps', 'RIGHT n steps' and (effectively) 'SHUT FULLY'. Every 'LEFT' or 'RIGHT' command corresponds to putting out the right levels or pulses on the appropriate line and taking n steps requires repeating the command after the previous step has been completed. The 'SHUT FULLY' command is provided for initiating the throttle. Details of applying a step, testing the completion of a step, and the hardware to achieve these are presented in Appendix A.

2.2.4 PRESSURE MEASUREMENT

The pressure of the steam in the boiler is estimated indirectly through the use of a thermistor. The circuit designed allows measurements of up to 10 atmospheres, the measured variable being a voltage signal proportional to the pressure. The circuit diagram of the electronics associated with the thermistor is given in Apprndix A. This circuit is designed such that the relationship between the measured output voltage and pressure can be chosen and established by the designer. Clearly a linear relationship is desirable in this application and the transposition from the various variables involved to the final linear relationship between measured output voltage and pressure is shown in Fig.2.7.



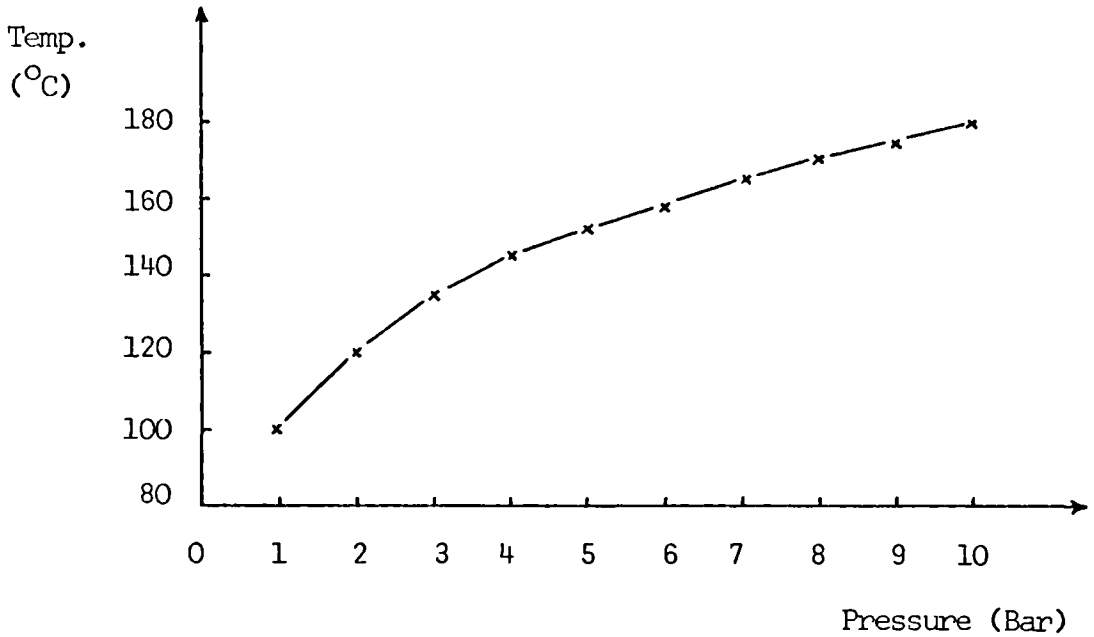
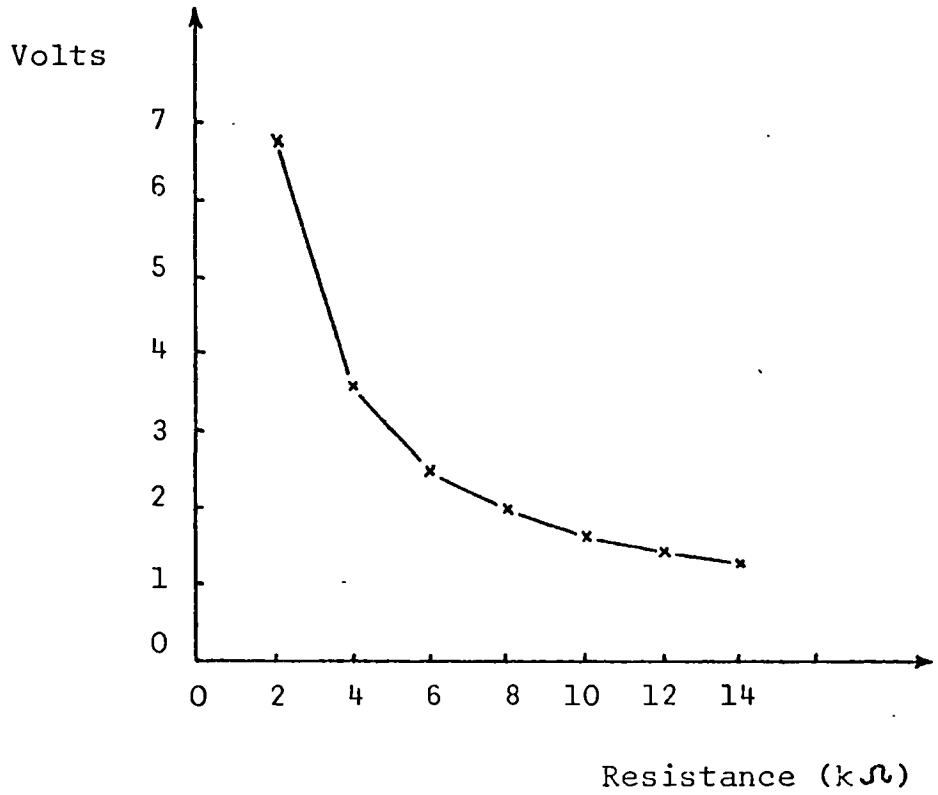


Fig. 2.7 Pressure Measurement



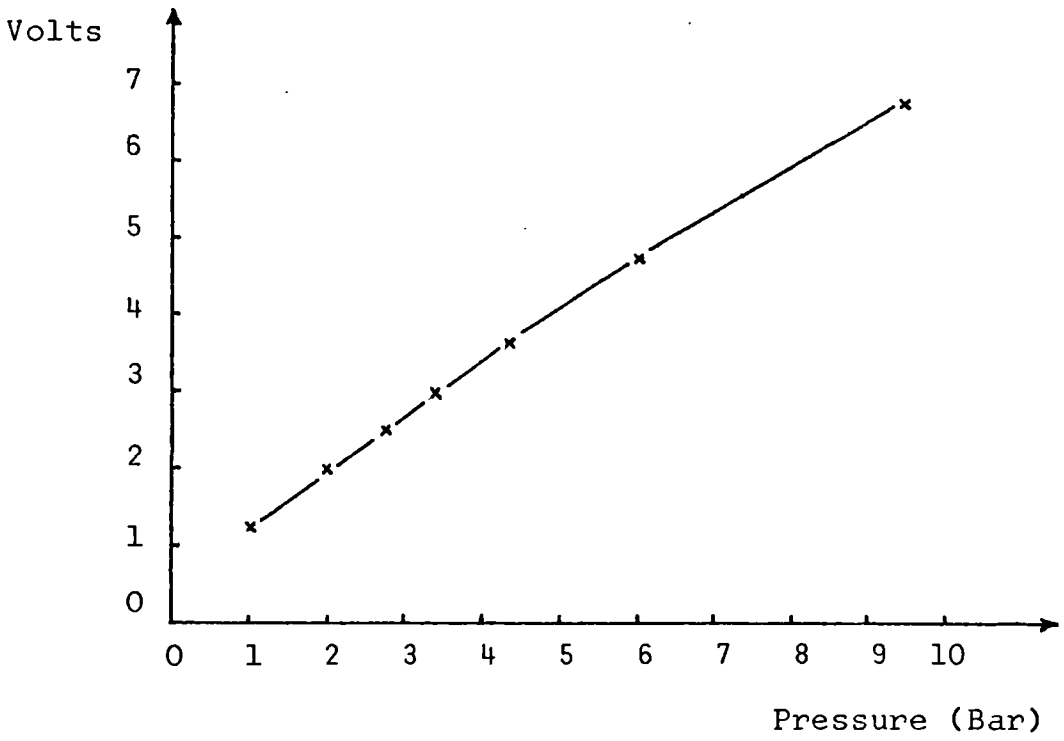


Fig. 2.7 (cont.) Pressure Measurement

2.2.5 SPEED MEASUREMENT

The speed of the engine is represented by the output voltage of the tachometer which is coupled directly to the engine crank rod, as shown in Fig.2.2. The relationship between the output voltage of the tachometer and the speed of the engine is shown in Fig.2.8 and can be seen to be linear except at very high speeds.

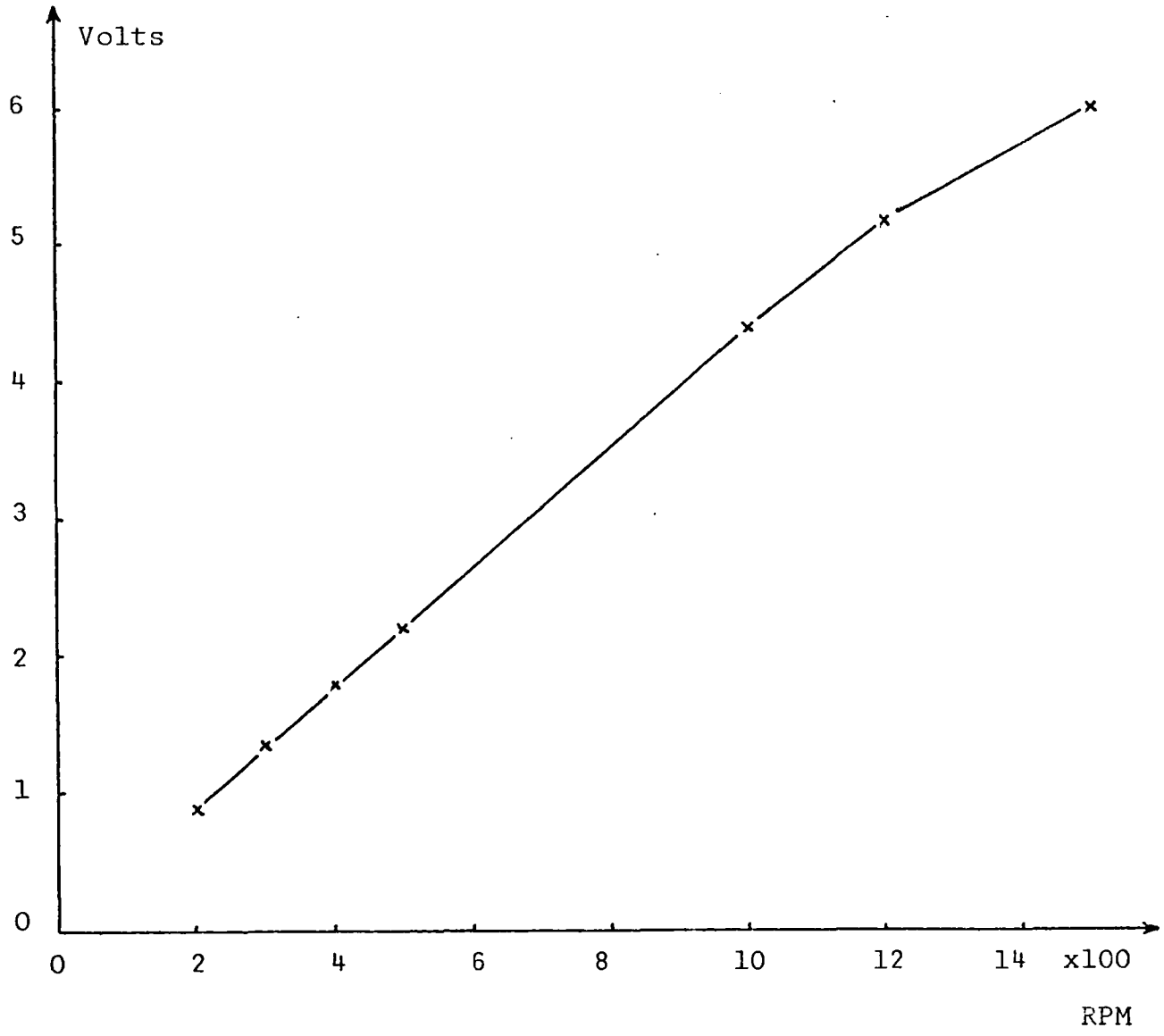


Fig. 2.8 Speed Measurement

2.3 DYNAMIC MODELLING OF THE PLANT

The modelling of a real dynamic process normally constitutes a project on its own right. The process of modelling can conveniently be segmented into two stages. In the first stage, the boundaries of the system to be modelled must be established and the variables falling within these boundaries must be examined and classified according to their role in the process; that is, whether they are dependent or independent, controlled or uncontrolled, or measured or unmeasured variables. Clearly, this stage of the modelling is influenced by the purpose that the model is to serve. For example, in the case of the steam engine, a model including the superheater would have different boundaries than the model developed below; there will possibly be some variables common in both models, but it will also include different variables (Savas, 1965).

The second stage of modelling involves the development of relationships between the variables. These relationships, or process equations, are usually derived by a combination of both theoretical analysis and empirical observation of the process - a good example of analysis and synthesis. The theoretical basis for the process model rests on scientific knowledge about the fundamental chemical and physical phenomena which govern the process. Ordinarily, theoretical analysis of the process is the starting point in model development and is pursued as far as practicable, before turning to empirical techniques to supplement or verify the theoretical model. Empirical supplementation involves the determination of the numerical values of the parameters or constants in the model which cannot realistically be derived by theoretical analysis. In any case, empirical observation is a necessary and important aspect of modelling in order to establish whether the theoretical model does in fact adequately describe the actual behaviour of the process.

2.3.1 THE MODEL

An attempt to develop a theoretical model of the steam engine system will invariably involve complicated thermodynamic and energy conservation equations (Evans and Fry, 1964). Having established these equations, it is then necessary to introduce numerical values relating to physical quantities, for example, boiler dimensions, in order to calculate the parameters or constants of the model. Because such an approach reflects process fundamentals, the resultant model can be very reliable. However, apart from its other limitations, such an approach may yield an unnecessarily over-complex model for the control purposes of this project. Therefore, rather than attempting to develop a full theoretical model, a simple model is considered here involving single time constants only.

All the controllers to be considered in this dissertation will use the input and output variables shown in Fig.2.9. A set of linearised perturbation equations of these variables is now proposed.

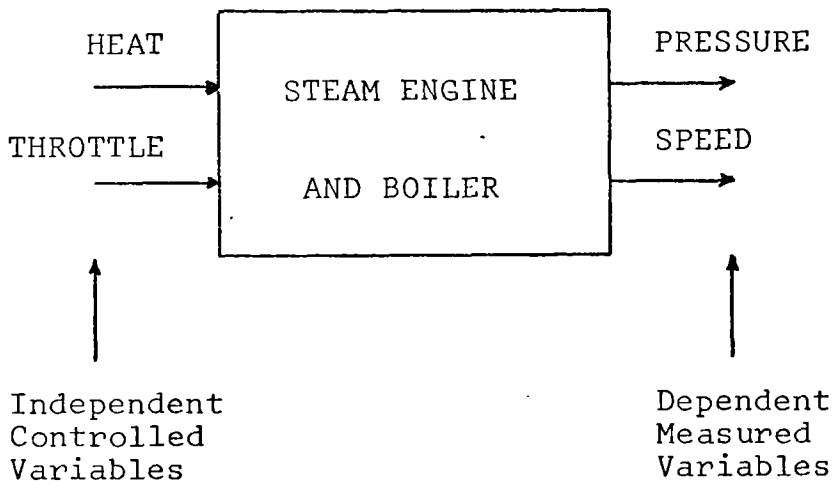


Fig. 2.9 Process Variables

The pressure in the boiler is assumed to be directly related to heat and throttle and as first approximations:

$$\frac{P_1(s)}{H(s)} = G_1(s) = \frac{G_1 \exp(-s\tau_1')}{1+s\tau_1} \qquad (2.3)$$

$$\frac{P_2(s)}{T(s)} = G_2(s) = -\frac{G_2 \exp(-s\tau_2')}{1+s\tau_1} \qquad (2.4)$$

where P_n = pressure variation
H = heat variation
T = throttle variation
$G_n(s)$ = transfer function
G_n = gain
τ_n = time constant
τ_n' = time delay
s = Laplace operator w.r.t. time

It has been assumed, from knowledge about other similar systems, that $G_1(s)$ and $G_2(s)$ have approximately the same time constant.

From Eqns.(2.3) and (2.4)

$$P(s) = P_1(s) + P_2(s)$$

$$= \frac{G_1 \exp(-s\tau_1')}{1+s\tau_1} H(s) - \frac{G_2 \exp(-s\tau_2')}{1+s\tau_1} T(s) \qquad (2.5)$$

where P = total pressure variation

The total steam flow to the engine can be considered to be the sum of that due to the throttle position at constant pressure, and that due to up-stream (boiler) pressure at constant throttle position, i.e.

$$W(s) = G_5 \exp(-s\tau_5')T(s) + G_6 \exp(-s\tau_6')P(s) \qquad (2.6)$$

where W = steam flow variation

Lastly, the speed of the engine is related to the steam flow and a single time constant is introduced to represent the inertia of the flywheel, i.e.

$$S(s) = \frac{G_7 \exp(-s\tau_7')}{1+s\tau_2} W(s) \qquad (2.7)$$

where S = speed variation

Substituting for W(s) from Eqn.(2.6)

$$S(s) = G_3(s)P(s) + G_4(s)T(s)$$

$$= \frac{G_3 \exp(-s\tau_3')}{1+s\tau_2} P(s) + \frac{G_4 \exp(-s\tau_4')}{1+s\tau_2} T(s) \qquad (2.8)$$

where $G_3 = G_6 G_7$

$G_4 = G_5 G_7$

$\tau_3' = \tau_6' + \tau_7'$

$\tau_4' = \tau_5' + \tau_7'$

A block diagram of the model is shown in Fig.2.10.

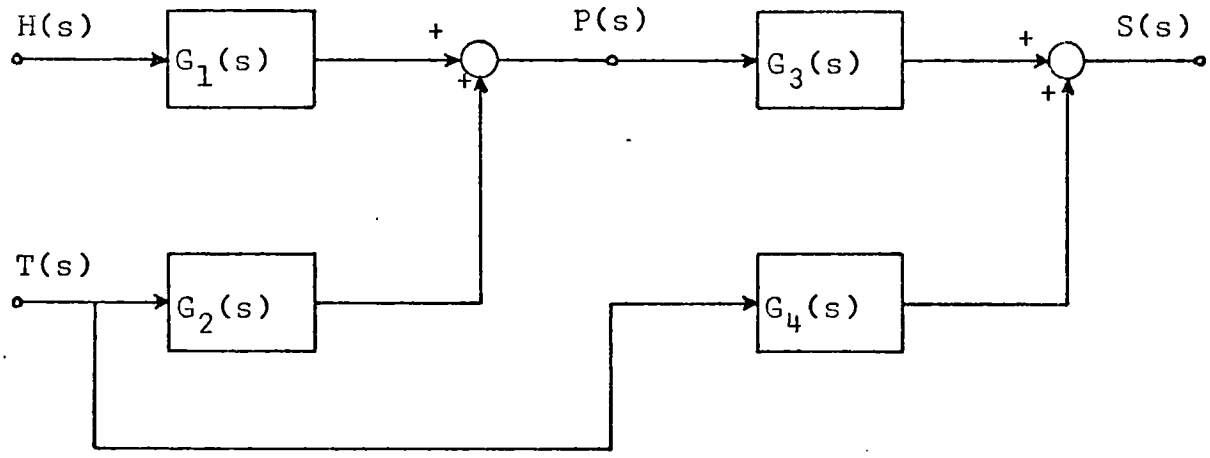


Fig. 2.10 Block Diagram of Model

2.3.2 DESIGN OF EXPERIMENTS

When an engineer is confronted with a multivariable system it is important to have a carefully planned programme of experiments in order to eliminate bias and to control the errors of estimate, of the parameters of the model, in relationship to measurement errors (Savas, 1965). For the purposes of this project, experiments were designed bearing in mind the following points:

(1) Does the simple model proposed in the last section describe adequately the actual behaviour of the process, and if so, what are the values of the various parameters;

(2) Does the assumption of linearity hold:
   (a) For different amplitudes of the test signal,
   (b) For opposite polarities of the test signal, and
   (c) For (a) and (b) with different initial conditions.

The test signal used was the step function. In most applications it is a simple matter to generate a step function. Furthermore, step response tests are amenable to convenient and easy analysis by graphical methods. However, because of the very noisy nature of the environment, the generalized least squares method (Clarke, 1967) was used to estimate the parameters of the model.

2.3.3 ANALYSIS OF EXPERIMENTS

The long series of experiments carried out are not described in this dissertation, as space limitations do not allow this. Some of the results are included in Appendix B.

The important conclusions of the tests are summarized below in the same order as the questions raised in the last section.

(1) A single time constant is adequate in all four transfer functions of Fig.2.10. Furthermore, the time delay is negligible in all four functions. The estimated transfer functions are shown in Fig.2.11 and it can be seen that the time constants of $G_1(s)$ and $G_2(s)$ are equal, as anticipated in Eqns.(2.3) and (2.4). However, the time constants of $G_3(s)$ and $G_4(s)$ are different, in contrary to Eqn.(2.8). It is suggested that this is due to the assumption of Eqn.(2.6). Finally, as a result of the non-linearities existing in the system, the gains quoted in Fig.2.11 are averages of the different estimates obtained for variations of the test signal and the initial conditions.

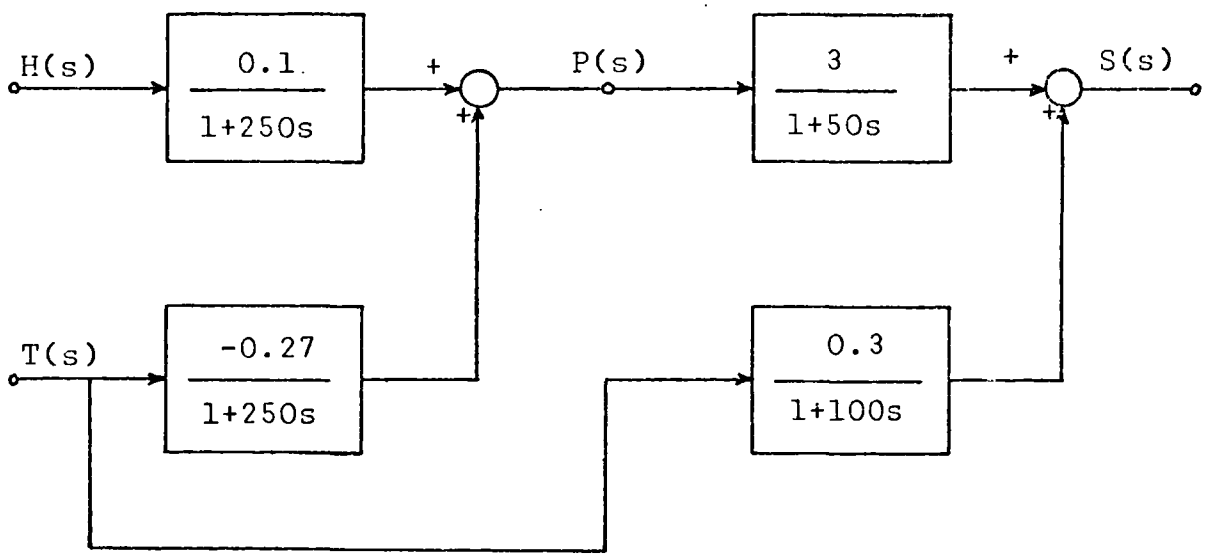


Fig. 2.11 Estimated Steam Engine Model

(2) In general, the system is non-linear with both magnitude and polarity of the test signal. It also exhibits different characteristics in different areas of the operating space. In view of this observation there arises the question regarding the usefulness of the model given in Fig.2.11, since it is not associated with any particular operating point. However, as pointed out in Chapter 3, this model is intended to serve as a starting point in the design of the controller which is further tuned on-line.

As a final remark, the speed variable was observed to be rather noisy compared to the pressure variable. In addition, the speed variable exhibited either poor sensitivity or extra-sensitivity to throttle variations, instead of the desired uniform response. However, rather than cure these situations, it was felt that they could present an interesting environment for the learning and fuzzy logic controllers.

2.4 THE SOFTWARE SYSTEM

The development and writing of the software for the digital computer played an important role throughout this project since ultimately all theoretical notions had to be implemented and tested on the experimental set-up. Ordinarily, the writing of software does not present too big a task, specially when experiments are carried out on simulated systems so that a high-level language can be conveniently and adequately used. In this project, however, the assembler language had to be used because of the real-time nature of the system, and in order to cope with the workload imposed on the system, full use had to be made of the 'interrupt' mechanism available on any general purpose computer.

A flow-chart of the software is shown in Fig.2.12. On the occurrence of an interrupt, originating either in the plant or from the human operator, the interrupt service routine is activated which passes control to the special routines or the foreground programme depending on the source and meaning of the interrupt. The special routines

respond to stimuli like 'water below safe limit in boiler', or 'pressure beyond safe limit in boiler', and they initiate certain panic-actions, like the total shut-down of the system. The foreground programme executes at interlock level (non-interruptible) and it responds to all other, normal requests, like an input to the plant at the end of a sampling interval. On finishing its execution, the foreground activity passes control to the background segment which is responsible for the general 'house-keeping' of the system; logging the events taking place, processing any data queued by the foreground activity (for example, updating the memory of adaptive controllers) and outputting all this information to the printer.

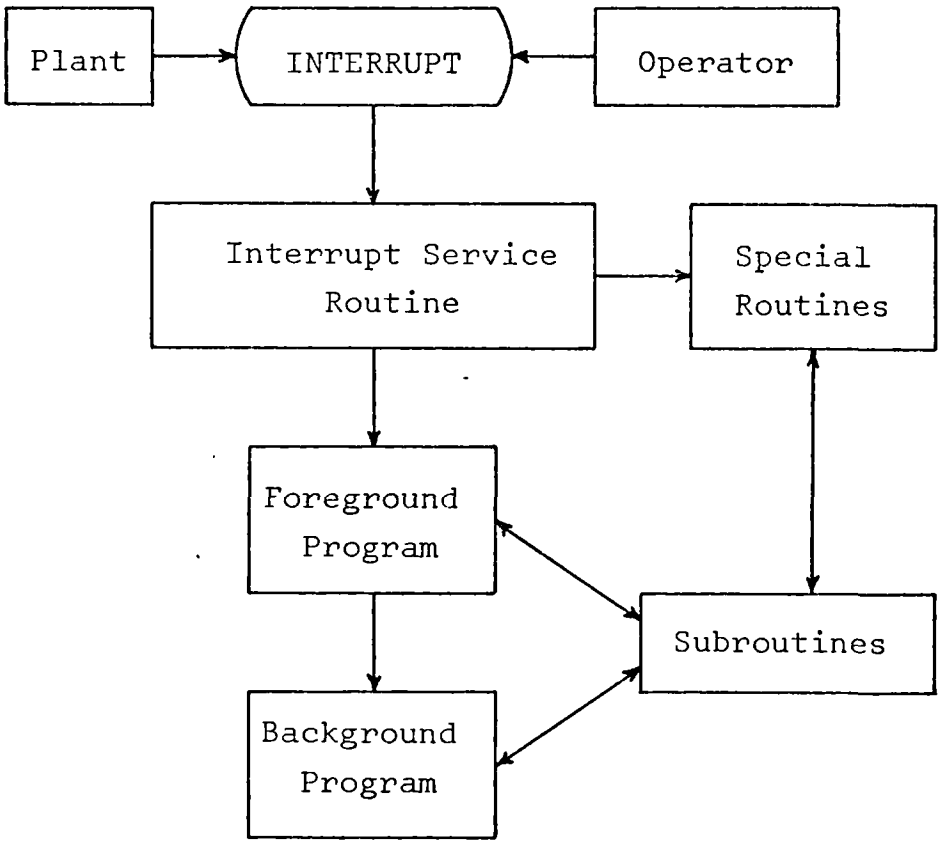


Fig. 2.12 Software Flow-Chart

The length of the software is over 3000 machine words and in order to minimize the changes necessary between one series of experiments and another, the foreground and background segments are written in the form of calls to subroutines. With the use of this type of modularity introduced into the software, the implementation of the modelling programme, the conventional feedback controller, the adaptive controllers and the 'fuzzy logic' controller, amounted to compiling practically the same collection of routines with only few lines of coding altered amongst them.

CHAPTER 3

DIRECT DIGITAL CONTROL OF THE STEAM ENGINE

3.1 INTRODUCTION

Although the main aim in this project is the investigation of the applicability of 'artificial intelligence' - adaptive and fuzzy logic theory - in control, it deemed worthwhile and justifiable to examine in the first place the quality of control achievable with a simple conventional controller. Since we are starting with a new plant, at worst the performance of the conventional controller could prove the system to be uncontrollable, at least with the tools available in classical control theory. At best, when compared with the performance of learning or fuzzy logic controllers, it could prove that classical controllers are still far superior, possibly bringing out at the same time any differences which can aid in the improvement of the other controllers. In any case, the performance to be obtained by the conventional controller is intended to provide a benchmark in the final assessment of the quality of learning and fuzzy logic control.

The implementation and testing of the conventional controller was again a lengthy exercise like the mathematical modelling of the system. This chapter gives a brief account, including important details only.

3.2 BRIEF STATEMENT OF THE CONTROL PROBLEM

Using vector-state notation, the behaviour of the linear and stationary plant can be described by the vector equation

$$\dot{X}(t) = AX(t) + BU(t) + FD(t) \qquad (3.1)$$

where X=n-vector representing the state of the plant

$\dot{X}$=first time derivative of X

U=r-vector representing controllable inputs to the plant

D=s-vector representing uncontrollable inputs(disturbances)

t=time

X,U,D=functions of time

A=nxn (constant) system matrix
B=nxr (constant) distribution matrix
F=nxs (constant) disturbance matrix

The plant output vector, which is not necessarily identical with the system state vector, is given by

$$Y=CX \tag{3.2}$$

where Y=q-vector representing the outputs of the plant
C=qxn output or measurement matrix

The performance of the plant at any time can be expressed as some scalar quantity

$$P(t)=P(X,U,D,t) \tag{3.3}$$

where t is included to allow for the possibility that the performance measure may be time-varying.

In general, the control objective is to maximize or minimize, as the case may be, the integral of P(t) over all time, i.e. the performance index is

$$I=\int_{t_o}^{\infty} P(t)dt \tag{3.4}$$

where t_o is the initial time when control is applied.

Any such optimization must, of course, be carried out within the operating limits of the plant. Therefore, the control problem statement has to include restrictions

$$\text{all } X\in R_1(t) \text{ and all } U\in R_2(t) \tag{3.5}$$

where $R_1(t)$ and $R_2(t)$ are vector spaces.

3.3 THE SOLUTION TECHNIQUE

Most control situations assume that a model (Eqn.3.1) of the process is available and simple feedback is sufficient to compensate for variations in process characteristics and other uncertainties. The optimum control vector (U(t)) is then computed from current process input and output measurements using a prescribed, fixed procedure based on the model. This procedure may involve solving a set of differential or algebraic equations depending on the problem formulation (Eqns.3.3 and 3.4) and mathematical approach.

The solution for the optimum control vector in the above manner for single-input, single-output processes is fairly well understood, at least for the linear and stationary case, and relatively simple and easy-to-implement procedures or techniques are available in the control literature. However, the solution is by no means trivial for multivariable feedback systems (Macfarlane, 1972). Obviously, the difficulty in multivariable feedback systems arises out of the interacting effects between the multiple loops. Recalling the results of the identification tests carried out in Section 2.3, the steam engine must be placed in the class of interacting multivariable systems.

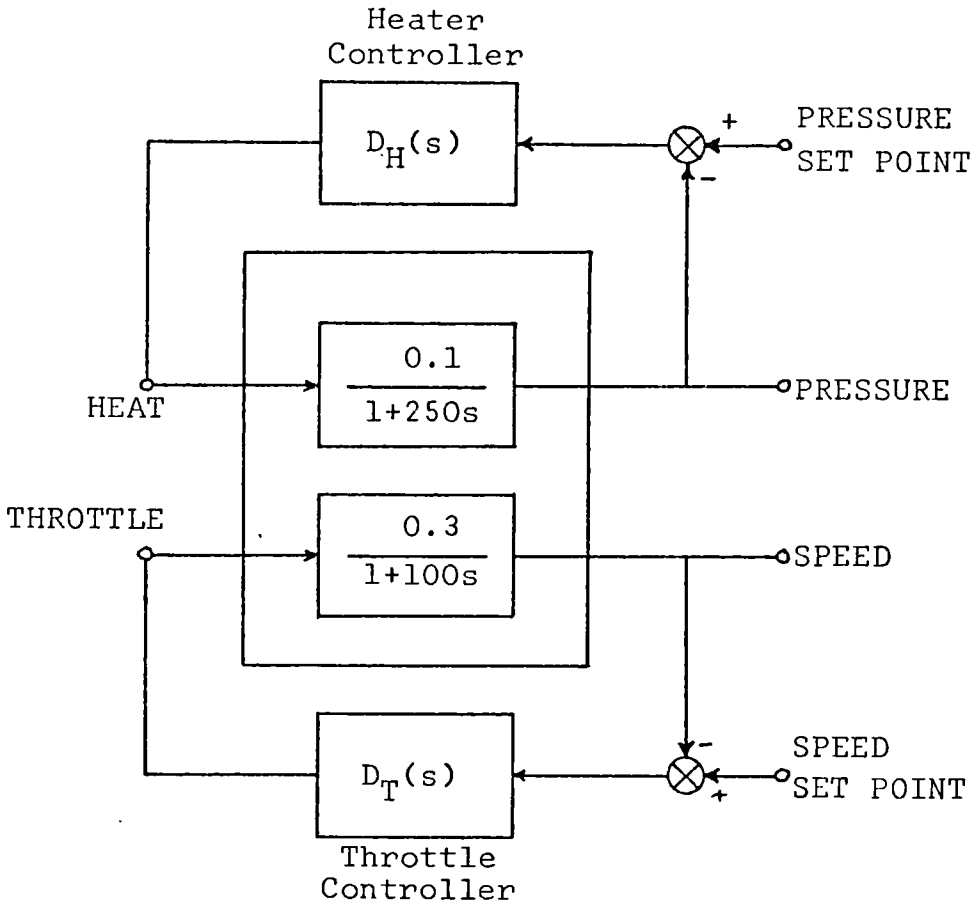


Fig.3.1 Steam Engine Control Configuration

The method, or to be precise, the notion therein, adopted in designing a controller for the steam engine is that of 'non-interacting control technique'. The principle of non-interactive control is to convert the multivariable feedback system into a family of independent, single-input, single-output loops, each of which can be handled by conventional feedback theory. It is needless to say that this technique does suffer certain disadvantages (Macfarlane, 1972). However, in the face of the very approximate model obtained for the steam engine, so that any precise mathematical derivation of a controller would be meaningless, the simplicity of the underlying principle of non-interactive control is attractive enough to implement the control configuration shown in Fig.3.1.

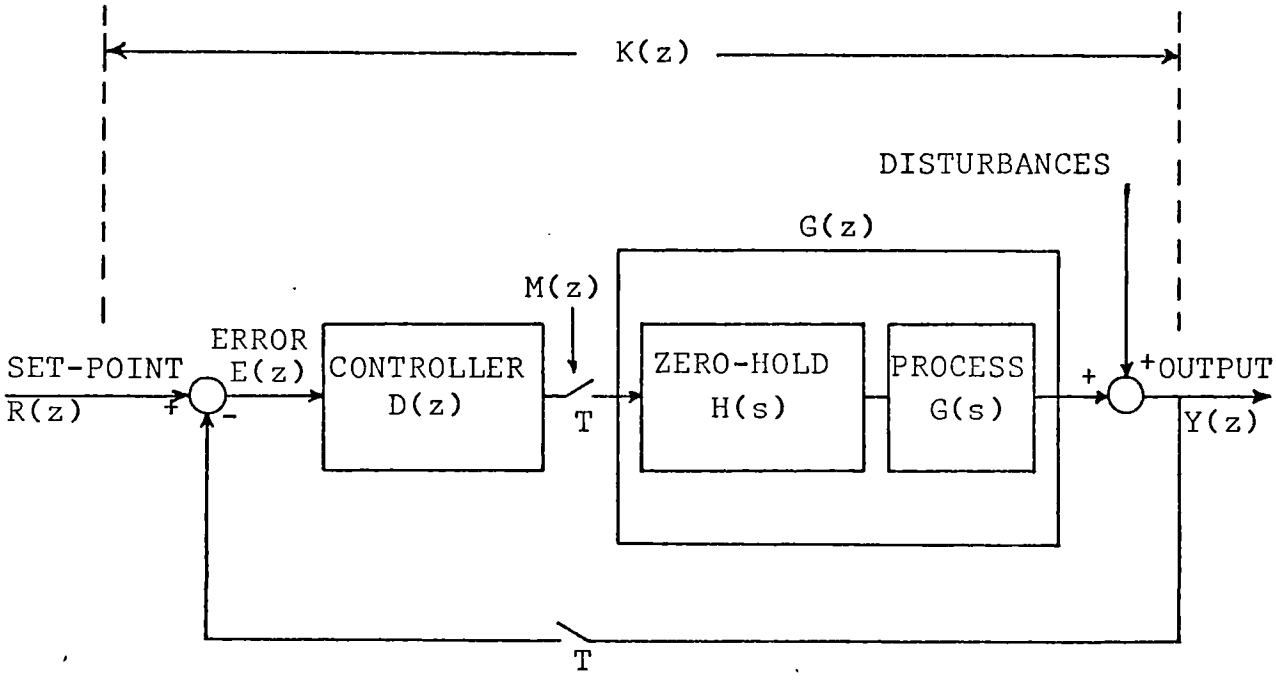


Fig. 3.2 Closed-Loop Control Configuration

Comparing Fig.3.1 with Figs.2.10 and 2.11 in the last chapter, it can be seen that the transfer functions used in the control configuration are $G_1(s)$ and $G_2(s)$. Although the controller settings of $D_H(s)$ and $D_T(s)$, derived using these transfer functions, do not promise good control, it is intended that these controller settings serve as the starting point for further on-line tuning to obtain the best possible control of the plant.

3.4 CONTROLLER DESIGN

Having reduced the multivariable feedback control system to independent, single-input, single-output loops, two simple digital algorithms are derived below for the heater and throttle controllers, $D_H(s)$ and $D_T(s)$ respectively in Fig.3.1. The closed-loop configuration and the definitions of general terms to be used for either controller are shown in Fig.3.2. It is noted that the output of the controller, that is the digital computer, passes through a first-order sample-and-hold which is an adequate description of the interface between plant and computer. Using block diagram algebra, it can be shown that

$$D(z) = \frac{1}{G(z)} \frac{K(z)}{1-K(z)} \tag{3.6}$$

3.4.1 PI AND PID CONTROLLER DESIGN

The well-known general form of the controller here is

$$D(s) = \frac{M(s)}{E(s)} = K_C\left(1 + \frac{1}{T_I s} + T_D s\right) \tag{3.7}$$

where the three terms correspond respectively to the proportional, integral and derivative actions of the controller.

Optimum settings for the three constants were originally suggested by Ziegler and Nichols (1942) from empirical studies. However, the Ziegler-Nichols settings

necessarily include a time delay of the process, which in the case of the steam engine is negligibly small. The other alternative for calculating the optimum values for the three constants above is to use analytical methods involving the minimization of error functional integrals to define optimality (Eqn.3.4). The most often used error integrals are the error squared integral (ISE), absolute error integral (IAE), and integral of the product of time and absolute error (ITAE) (Lopez et al, 1969). In this project the ITAE criterion is used. The relationships between the optimum values of K_C, T_I, T_D, the system parameters G, τ (Eqn.3.11), and the sampling interval T, which must be considered in sampled-data systems, can be found in the above reference.

To implement Eqn.(3.7) in the form of a digital controller, it is first put in digital form, thus:

$$m_n = K_C \left[e_n + \frac{1}{T_I} \sum_{i=o}^{n} (e_i + e_{i-1}) \frac{T}{2} + T_D \left(\frac{e_n - e_{n-1}}{T} \right) \right] \quad (3.8)$$

The equivalent of Eqn.(3.8) in z-transform notation is

$$D(z) = \frac{M(z)}{E(z)} = \frac{\alpha_o + \alpha_1 z^{-1} + \alpha_2 z^{-2}}{1 - z^{-1}} \quad (3.9)$$

where

$$\alpha_o = K_C \left[1 + \frac{T}{2T_I} + \frac{T_D}{T} \right]$$

$$\alpha_1 = -K_C \left[1 - \frac{T}{2T_I} + \frac{2T_D}{T} \right] \quad (3.10)$$

$$\alpha_2 = K_C T_D / T$$

The form of Eqn.(3.9) is the one most suitable for implementation on a digital computer.

3.4.2 'SINGLE-TERM' CONTROLLER DESIGN

In this controller, due to Higham (1968), one parameter is used to adjust the closed-loop performance of the system. Although it was originally designed for processes with dead time, the algorithm is easily modified for processes with no dead time. The single parameter also overcomes mis-measurements of the system parameters, which can cause poor control and ultimate instability.

Consider the general process having dead time kT (Fig.3.3) such that

$$G(s) = G\,\frac{\exp(-skT)}{1+s\tau} \qquad (3.11)$$

Then

$$\begin{aligned} G(z) &= \mathcal{Z}\,\{H(s).G(s)\} \\ &= \mathcal{Z}\,\{\frac{1-\exp(-sT)}{s}\,.\,G\,\frac{\exp(-skT)}{1+s\tau}\} \\ &= \frac{GLz^{-(k+1)}}{1-(1-L)z^{-1}} \qquad (3.12) \end{aligned}$$

where $L = 1-\exp(-T/\tau)$ (3.13)

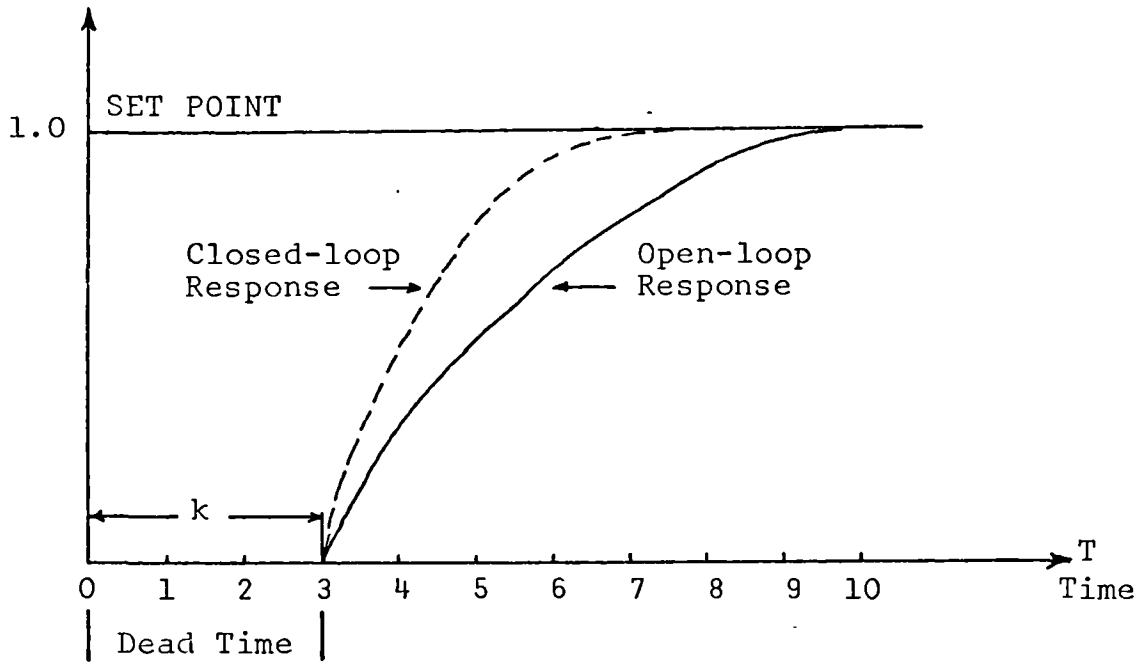


Fig. 3.3 Controlled Response for Unit Change in Set Point.

The next step is to define the overall closed-loop transfer function K(z) of the control loop. By looking at the open-loop transfer function of G(z) in Eqn.(3.12) and Fig.3.3, K(z) can be expressed as

$$K(z)= \frac{Qz^{-(k+1)}}{1-(1-Q)z^{-1}} \qquad (3.14)$$

where $Q=1-\exp(T/\tau_c)$ (3.15)

τ_c=time constant of controlled response

Substituting for G(z) and K(z) from Eqns. (3.12) and (3.14) into Eqn.(3.6)

$$D(z)= \frac{1}{G} \cdot \frac{Q/L-z^{-1}Q(1-L)/L}{1-(1-Q)z^{-1}-Qz^{-(k+1)}} \qquad (3.16)$$

For a process with no dead time, i.e. k=0,

$$D(z)= \frac{1}{G} \cdot \frac{Q/L-z^{-1}Q(1-L)/L}{1-z^{-1}} \qquad (3.17)$$

Eqn.(3.17) can be recognised to be the discrete form of a PI controller - compare with Eqn.(3.9).

3.5 PERFORMANCE

The mathematical expressions for the controllers referred to by way of curves A through J in the discussion below are given in Appendix C. The unit on the vertical axis of the graphs in Figs.3.4 through 3.7 is voltage, which is proportional to pressure or speed accordingly.

3.5.1 PI AND PID CONTROLLERS

Refer to Fig.3.4. Curve A shows the controlled response of the pressure for a step demand, using a PI heater controller with settings based on the transfer function shown in Fig.3.1. Curve C shows the same response but with the added effect of derivative action of the controller; that is, the PID controller. There is very little improvement on the response with the addition of derivative action. The explanation for this is that, for the first-order process with no

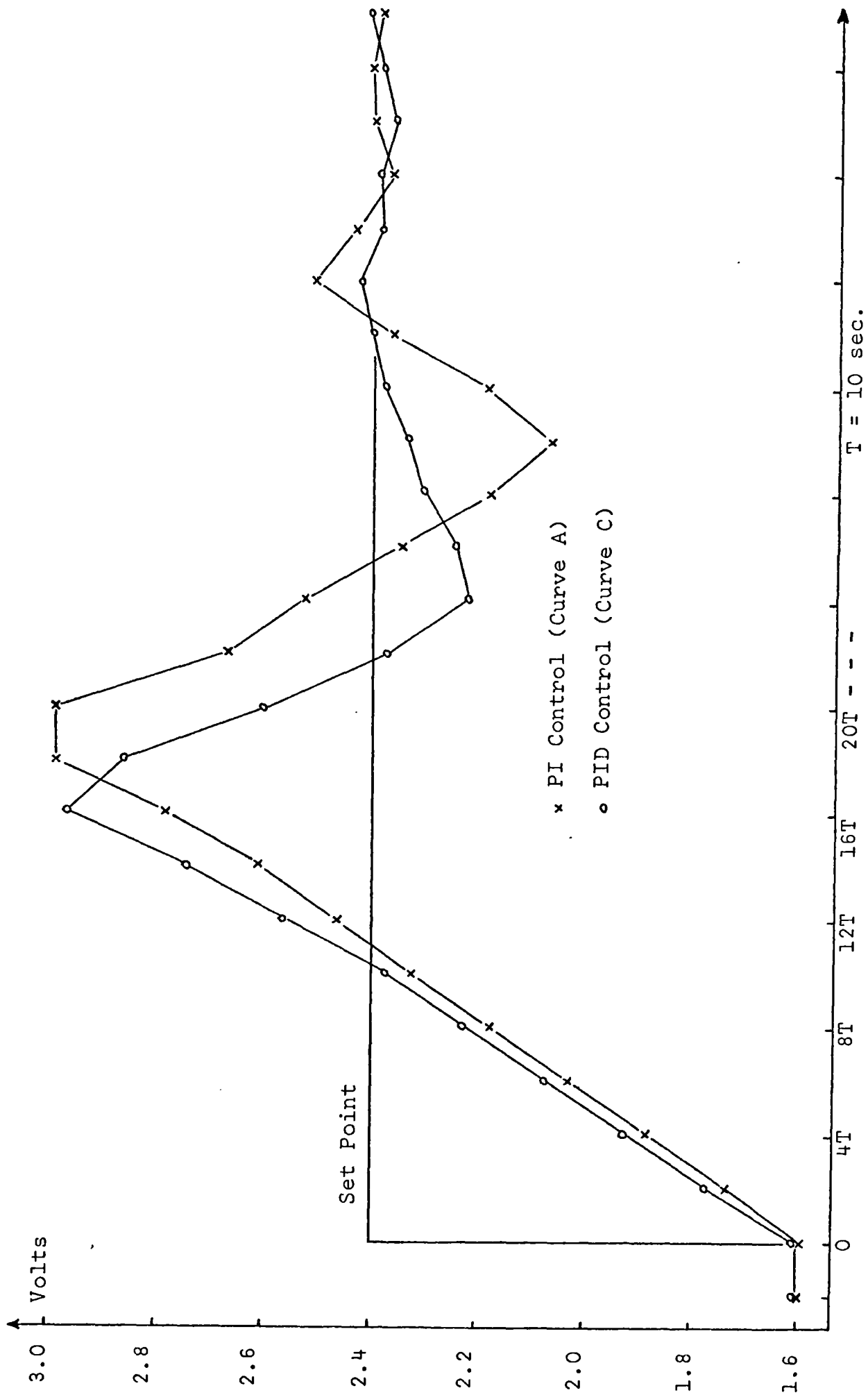


Fig. 3.4 Pressure Response

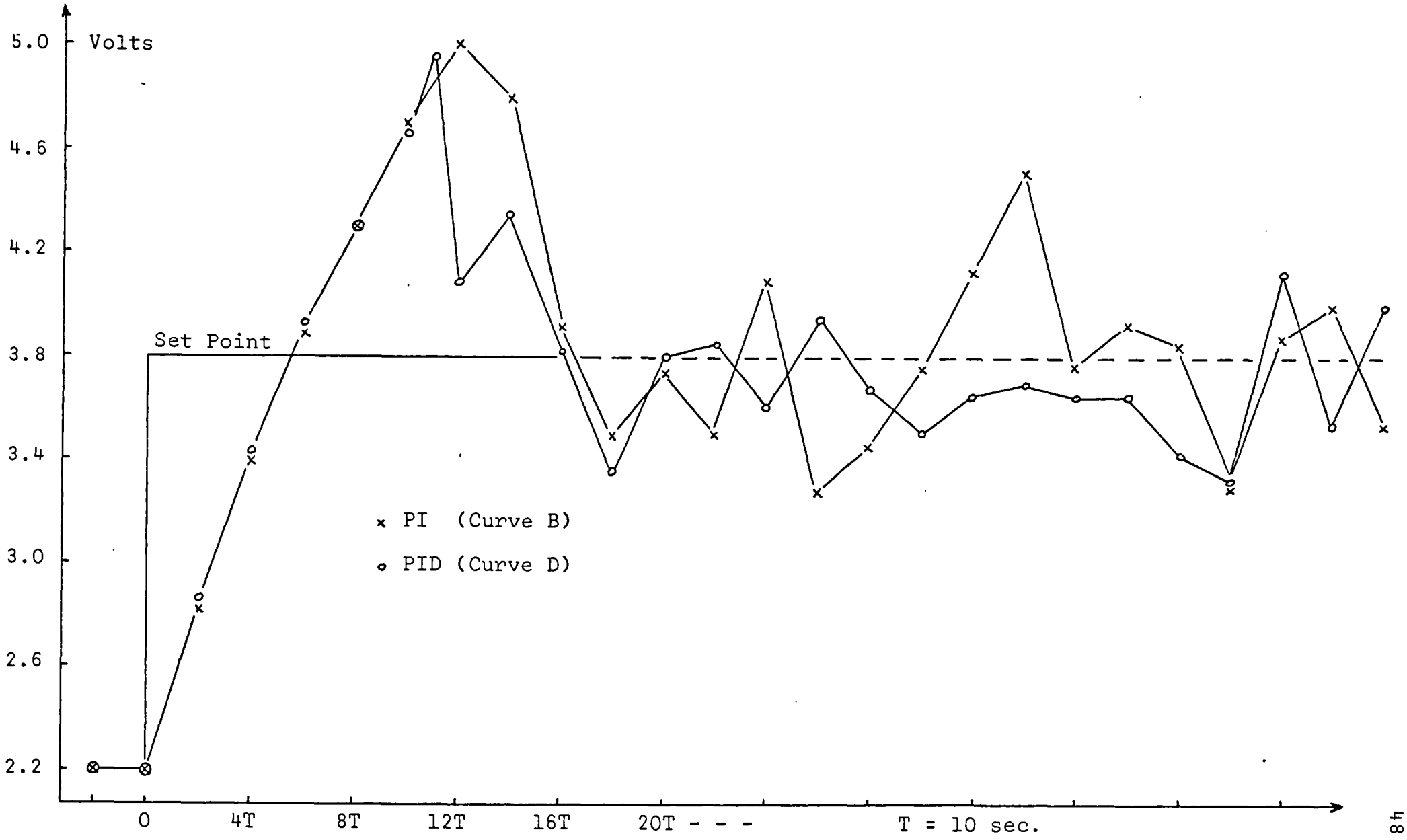


Fig. 3.5 Speed Response

delay time, the PI algorithm is optimum (Lopez et al, 1969). Apart from the above observation, both controllers produce a big first overshoot, although, admittedly, a big step demand is made in both cases. In common practice, the step demand is usually around fifteen per cent of the total operating range. The reason for making a big step demand here is to enable a comparison of the results of DDC with the responses to be obtained using adaptive (Chapter 5) and 'fuzzy logic' (Chapter 7) controllers. In the latter two cases, the nature of the quantized operating space induces the necessity of making rather big step demands.

The above arguments apply in their entirety to the speed variable as well, the controlled response of which is shown in Fig.3.5. The occasional big random errors appearing in the speed variable should not be misinterpreted as poor control. It has been mentioned in Section 2.3 that the speed variable is inherently very noisy, in addition to being poorly sensitive to throttle control in certain areas of the operating space.

To summarise, it can be said that fairly good control has been achieved with PI and PID controllers. Of course, further on-line tuning of both controllers is necessary, and having done this, it is reasonable to assume that improvement in the quality of control will be obtained. However, the tuning of the 'single-term' controller is preferred as it is easier to assess the effect of varying one parameter only, as opposed to two or three parameters. Furthermore, by looking at Fig.3.3, the effect of varying Q in the single-term controller has more to offer to 'insight' (which after all is what the human relies on for on-line tuning) than the effects of varying K_C, T_I and T_D, one or two or three at a time, in the PI and PID controllers.

3.5.2 'SINGLE-TERM' CONTROLLER

For Q=L, the closed-loop response of K(z) is the same as the open-loop response. This is a convenient value of Q to use as an initial value in the process of on-line tuning. In order to achieve tighter control, it is required to make Q>L. Therefore by incrementing the value of Q iteratively and judiciously, the best possible controller can be arrived at. This procedure is followed below.

Refer to Fig.3.6. Curve E shows the controlled response of the pressure for a step demand, with the parameters of the heater controller calculated for Q=L. Curves F, G and H show the same response for Q(H)>Q(G)>Q(F)>Q(E)=L. The expediency and efficiency of the single-term controller is clearly evinced in these responses. The final control achieved, curve H, can be considered to be 'very good'.

Although the same degree of success cannot be reported in the case of the speed variable, the quality of control can still be considered to be good, as shown in Fig.3.7. Curve I in Fig.3.7 corresponds to the response obtained with the throttle controller having its parameters calculated for Q=L, and curve J is the best control achieved, which shows some improvement over the former.

Comparing the responses obtained here with those of PI and PID control, it is observed that in general the single-term controller produces better results. The first big overshoot obtained with PI and PID control is not present in the single-term control. As mentioned at the end of the last section, however, had the PI and PID controllers been subjected to on-line tuning, they would possibly have converged to the same controllers as in this case.

Using the same, best heater and throttle controllers produced, further tests were carried out with different initial conditions, spanning much of the operating space, and with different step demands both in magnitude and direction. The good quality of control was retained in the majority of the tests. These results are not presented here.

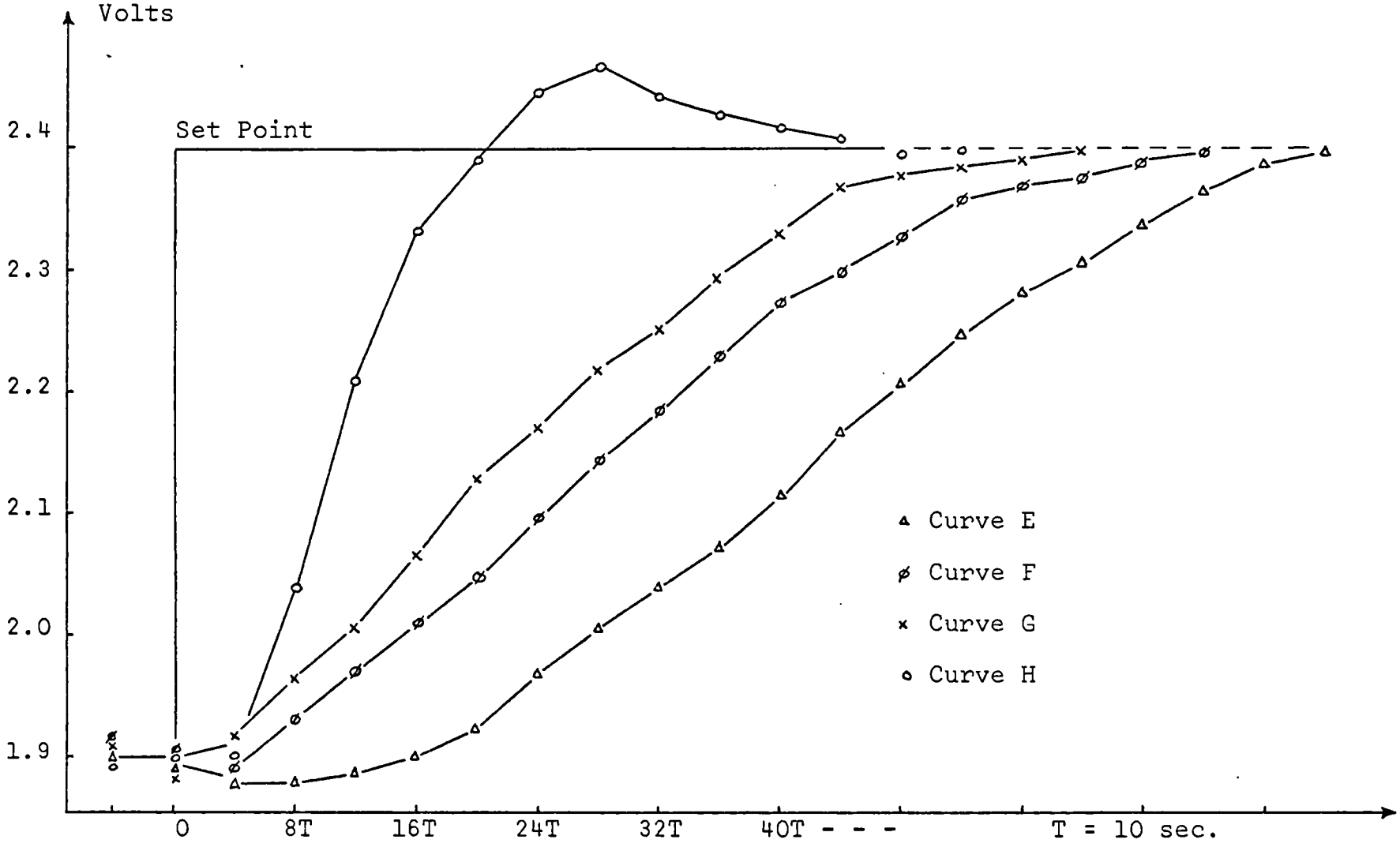


Fig. 3.6 Pressure Response

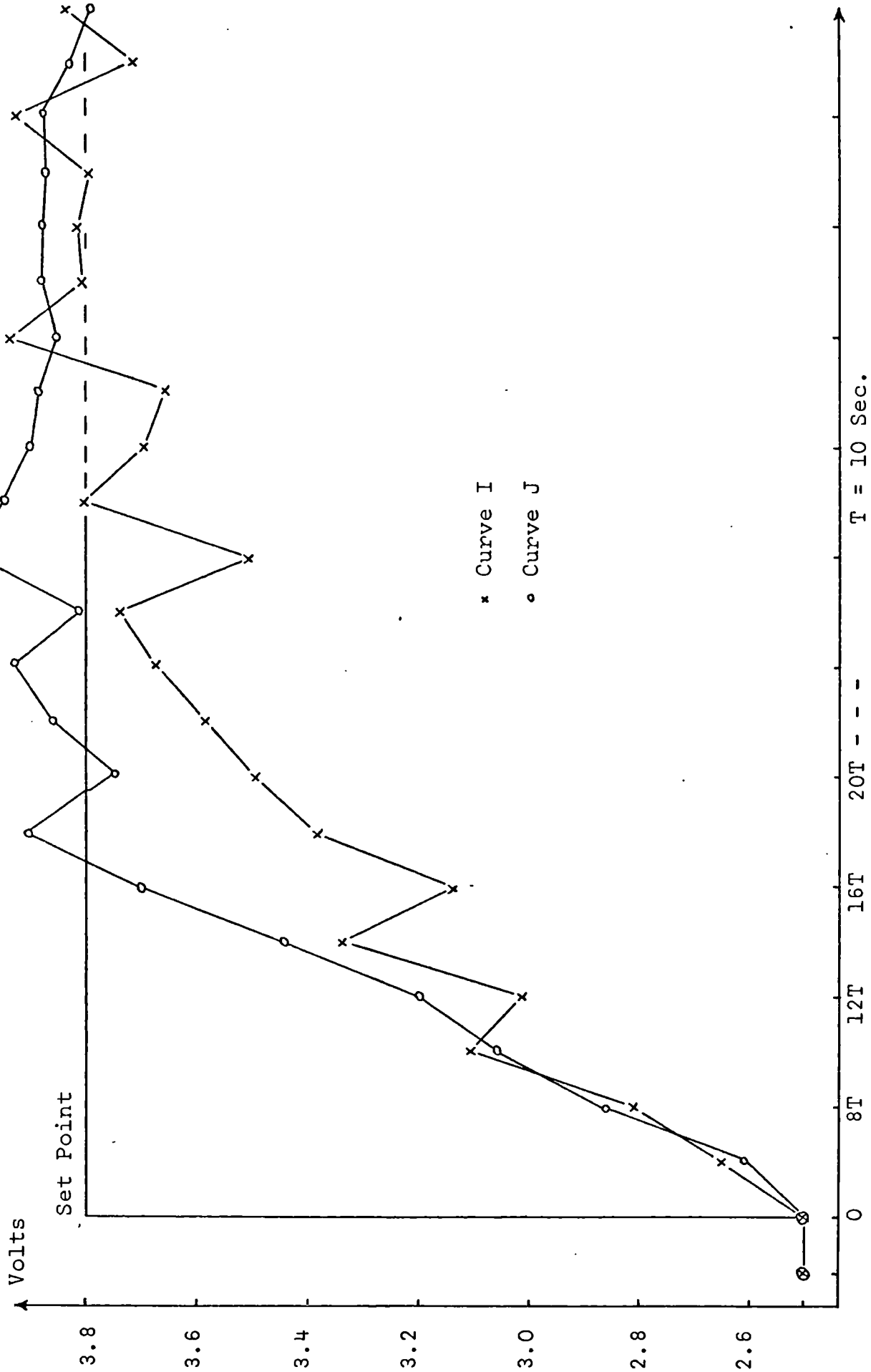


Fig. 3.7 Speed Response

3.6 CONCLUSIONS

Good control of the steam engine has been achieved using a single conventional controller. It ought to be feasible, therefore, for a learning controller to learn how to control with the same degree of success from monitoring the same information as the conventional controller design demands. The experimental work has also demonstrated the effectiveness of on-line tuning. This is important since accurate process models are very difficult to obtain with real systems. Furthermore, good control has been achieved using the same controller over a wide range of the operating space. Since the plant characteristics can vary a great deal under these circumstances, it is indeed ideal to have a controller which is not affected, at least not to a great extent, by changes in plant parameters. In the same way, a learning controller ought to be able to learn to be insensitive to the non-linearities that obviously exist in the system.

CHAPTER 4

PATTERN RECOGNITION AND ADAPTIVE CONTROL

4.1 INTRODUCTION

4.1.1 FUNDAMENTAL ADAPTIVE AND LEARNING CONTROL

Before embarking on the details of techniques available in the realm of adaptive and learning control it is first deemed appropriate to try and answer two fundamental questions: (a) What is meant by adaptive and learning control, and (b) The raison d'etre for adaptive and learning control.

For some time, the first question had the reputation of being the foremost controversy in the subject. Numerous papers were published advancing arguments in support of a particular definition of the terms 'adaptive' and 'learning' (Thorpe, 1950; Pask, 1963; Gibson, 1963), and inevitably the argument went even further to question whether a machine could at all 'think' (Turing, 1950). On the other hand, adaptive and learning control are different from any other science in that there is no unifying theory which subsumes the various, and in the majority of cases, only seemingly different activities in the general field of AI (Blake & Uttley, 1959; Cherry, 1961; Feigenbaum and Feldman, 1963; Tou and Wilcox, 1964; Mendel and Fu, 1970). However, the lack of a unifying theory cannot be attributed to the lack of a universally accepted definition of the terms adaptive and learning (and other synonymous terms like self-organising (Yovits et al, 1962), machine intelligence (Meltzer and Michie, 1969), etc.) but rather to the lack of knowledge on a more fundamental level, the act of learning itself - What are the mechanisms of learning? In view of the fact that the present study is an engineering application, the attitude taken here is best described by quoting Truxal (1963):

> "While the literature of control theory is replete with arguments re the definition, progress in adaptive control theory has not been impeded by the failure of purists to reach universal agreement on an appropriate definition."

There are two connotations of the terms adaptive and learning; behavioural and structural. Speaking generally from the behaviouristic point of view, where terminology is kept at the state-transition level (Gaines and Ardreae, 1966), no real distinction is made between adaptive and learning control. The typical situation is considered to consist of three constituents, the plant, the controller and a performance measure, and the expected behaviour of the controller when coupled to the plant is that, if its control policy is not satisfactory for that plant, then it will eventually become so (Gaines, 1971, 1972). In other words, the controller adapts itself or learns to improve its performance. Gaines has taken a step forward towards a formalized behavioural theory of adaptive and learning control through appropriate abstraction of the basic concepts of adaption. The obvious advantage of viewing adaptive behaviour in abstract terms is the possibility of bringing together the various activities in the general field of AI.

A distinction between adaption and learning is introduced when the structure of the controller is considered. It is at least generally accepted that a ~~universal~~ versatile learning machine (controller) has to be associated with a hierarchical structure of multi-level feedback loops (Sklansky, 1966). Thus, from this point of view, the structure of an adaptive controller is a two-level hierarchy, termed as the dual control mode (Feldbaum, 1963, 1965). A learning controller, on the other hand, would have at least one additional feedback loop resulting in a three-level or multi-level hierarchy. This concept is also in agreement with another school of thought where a 'memory' is associated with a learning controller over the adaptive controller (Mendel, 1967). The implication here is that when the plant or environment parameters change sufficiently often so that purely adaptive action (Elgerd, 1967)

could not optimize the performance of the controller, then a memory is necessary to store pertinent information about a particular situation in order that on consequent confrontation with that same situation the corresponding state of the controller can be immediately reinstated by accessing the memory. Finally, the concept of a hierarchical structure is also in accord with the so-called self-organising systems (Yovits et al, 1962), a term which has been associated with the structure of systems as opposed to behaviour (Fu, 1970). The word structure here does not necessarily refer to physical considerations, but a mathematical set of relationships can also be regarded as a systems structure (Mesarovic, 1962).

To recapture, the descriptives adaptive and learning in control are used synonymously in this dissertation when behaviouristic aspects are of interest, remaining cognizant at the same time of the multi-level hierarchical structure of such controllers. The importance of the upper level in this hierarchy, in which verbal communication takes place, is discussed in a later section.

Turning attention to the second question raised at the beginning of this section, the typical situation which justifies the need of a new art of learning controllers is when all or part of the a priori information necessary about the controlled process and its environment is unknown to the designer of the controller. Taking it one step further, the need also arises when the designer is faced with difficult sensitivity problems. In many of the above instances, it is reasonable to expect that a learning controller, which continually searches for the optimum within its allowed class of possibilities via estimating the unknown information, will have an eventual performance superior to that of a fixed controller which has been designed using only partial or incomplete information available.

There are many other realistic situations as well where the availability of learning systems is desirable; remote manipulation (Freedy et al, 1971; Whitney, 1969) to perform handling operations in environments which are hazardous

to human life,for example, underwater retrieval and the handling of radioactive materials; space vehicle control systems (Mendel, 1967); intelligent Robots (Nilsson and Raphael, 1967; Doran, 1969; Munson, 1971; Ejiri et al, 1971) to be used both for space exploration and on earth, for example, in automatic assembly; and indeed in situations where the complexity of the mathematical model of the environment renders the physical realisation of a controller impractical and uneconomical, if not impossible. However, where applicable, learning machines will be useful only if they can solve significant problems faster, cheaper or with greater programming ease than conventional machines. From this point of view, there is a trade-off between developing new techniques for modelling and recognition of complex environments by mathematical equations, which are suitable for applying known optimization methods in the design of controllers, and developing learning machines. The real advantage of learning controllers over fixed ones is when learning is to be carried on ad infinitum. Otherwise, if the environment is stationary, and it has been so in most cases studied thus far, then once learning is complete the learning controller can be replaced with a fixed system.

Although the varied uses that an ideal, general-purpose learning machine offers are attractive, and very often romanticized in the literature
(to quote a classic statement (Widrow, 1964):

> "It is expected that pattern recognizing control systems will be extremely flexible---- ultimately including processes whose complexities defy detailed mathematical description and analysis."),

one must be aware at the same time, as a control engineer, of the practical limitations that exist with the present state of the art. The purpose of this study is to investigate some of these limitations.

4.1.2 ADAPTIVE PATTERN RECOGNITION AND CONTROL

To date, most studies of learning machines have largely concentrated on adaptive pattern classifiers rather than adaptive controllers. The complexity of true learning machines, for example, STELLA (Gaines and Andreae, 1966), has restricted their applicability in practice and they remain only as a vehicle for further theoretical research in the study of learning machines. However, it is of overwhelming importance that learning machines include classification techniques in their skills, since to deal with a problem in general the machine has to recognise first what kind of problem it is faced with in order to choose between actions (Minsky, 1961).

Pattern recognition or classification can be viewed as a process by which an association, that is, an input-output relationship is formed between two sets of variables. A feature that distinguishes pattern recognition is the high dimensionality of the input space, and the reduction by grouping of this space through to the output is the task expected of the classifier. Because of their generality, pattern classifiers have been used for a great variety of tasks (Nagy, 1968); character recognition (Minneman, 1966), speech recognition (King and Tunis, 1966), weather prediction (Hu,1963), interpretation of aerial photographs (Welch and Salter, 1971) and control tasks (Widrow and Smith, 1964) are only a few examples.

The process of recognition can be conveniently segmented into two parts (Fig.4.1), the receptor and the categorizer (Marill and Green, 1963). In the receptor the input is subjected to a number of tests in predicate form (Minsky and Papert, 1969), each of which indicates whether a certain feature is present or not in the input pattern. Based upon the features present or absent, represented by a 'feature vector' X, the categorizer then assigns the input pattern to one of a finite number of categories using a decision procedure. In certain tasks, specially in character recognition,

it is expedient to carry out a pre-processing (with respect to noise, translation, rotation and size) of the input pattern in order to modify it in such a way that the probability of correct recognition is increased (Alcorn and Hoggar, 1969). Such processing is not necessary in most control systems as the input pattern is represented by a vector rather than a matrix.

It is obvious that for any recognition task, system performance depends upon effective solution of both segments of the pattern classifier. Although the performance of the categorizer can be optimized given a set of features, the selection of features in the receptor is very problem-dependent by necessity. Feature selection in character recognition, for instance, is primarily based upon the designer's ingenuity and intuition (Bomba, 1959; Lewis, 1962; Chow, 1962; Akers and Rutter, 1964). To the relief of the control engineer, the situation is less critical in control problems where a natural set of features is available as the best choice. Obviously this choice constitutes the set of

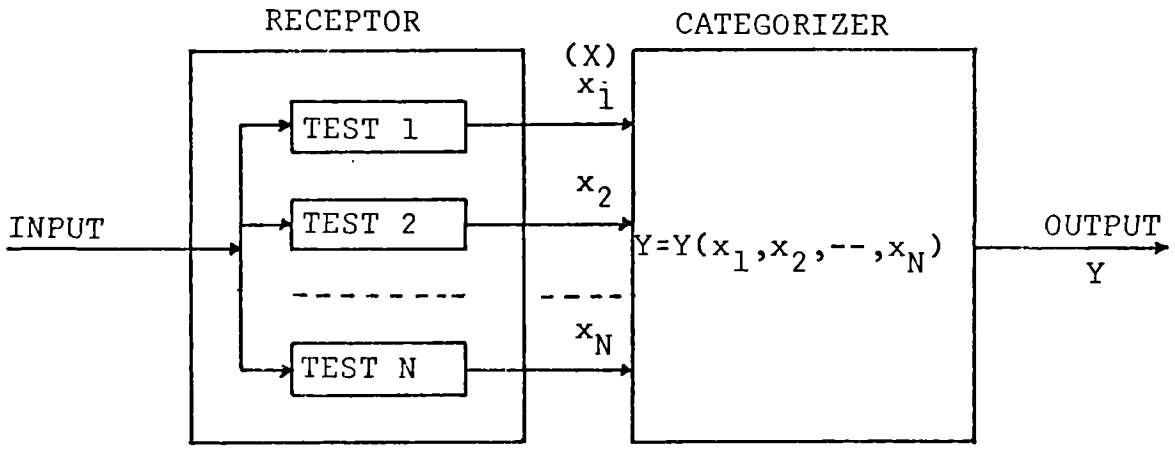


Fig. 4.1 A Recognition System

state variables of the process. Nevertheless, difficulties can still arise as not all of the state variables of a plant may be observable or measurable. Moreover, in the absence of a model of the plant, the state variables may not even be identified conceptually. In this case, the absence of a state variable, for example, pressure velocity in the steam engine, would render the pattern classifier useless (Mays,1964).

To summarise, it is possible to use an adaptive pattern classifier as an adaptive controller, since, when the state of a control system is represented as a pattern, then learning to make control decisions actually becomes the same as learning to classify the patterns.

The main techniques used for the design of classification mechanisms (the categorizer segment in Fig.4.1) are threshold logic and decision theory. The rest of this chapter investigates these methods, including ways of training such machines.

4.2 PATTERN RECOGNITION I - THRESHOLD ELEMENT

In the course of time, two general approaches have been established in the study of machine learning. One method, known as the neural-net approach, deals with the possibility of inducing learned behaviour into a switching net, possibly connected randomly, as a result of a reward/punishment routine (McCulloch et al, 1962; Rosenblatt, 1964). The second method tries to produce the equivalent of learning behaviour in the form of a heuristic computer program (Minsky, 1959, 1961; Waterman, 1968, 1970; Samuel, 1959,1967; Ernst and Newell, 1969; Newell and Simon, 1963, 1972). In this section, the first approach is considered which is mostly built around the adaptive threshold logic element (ATLE).

4.2.1 RATIONALE OF THRESHOLD ELEMENT APPROACH

The ATLE has been studied under the guise of various names, for example, Perceptron (Rosenblatt, 1962), Adaline (Widrow, 1962), and generally, learning machines (Nilsson, 1965). In spite of its practical limitations to be described later in this chapter, the ATLE is still perhaps the most powerful pattern classifier available to be utilized as the basic building block of complex, hierarchical control systems. The reason for this could well be its origin - the neuron, which is the basic human nervous system building block. The ATLE can be looked upon as an engineering version of attempts to model the human brain. These models are based on the 'neuron-doctrine' of brain functioning and they are influenced both by the physical structure of the neuron and known physiological phenomena. For mathematical convenience the properties of the neuron are considerably abstracted and simplified in constructing these models, however, the essential features are still retained. Good historical reviews on the neuron can be found in (Hawkins, 1961) and (Rosenblatt, 1962); a noteworthy contribution to the field is the original work of McCulloch and Pitts (1943) who showed the possibility of applying Boolean algebra to nerve net behaviour.

4.2.2 THE ADAPTIVE THRESHOLD LOGIC ELEMENT

The ATLE is capable of dichotomous classification of the input patterns. A typical ATLE is shown in Fig.4.2. The input pattern, X, is encoded into a binary N-vector with elements x_i=0,1 (or alternatively, $\pm$1), $1 \leq i \leq N$. Stored within the ATLE there is a N-vector of weights, W, with elements w_i, $1 \leq i \leq N$. The dichotomous classification is a numerical function - the linear threshold function - of the scalar product between the input pattern vector and the weight vector, and a threshold w_o. Denoting the threshold input by x_o, which is held fixed at +1, so that

$$W.X = \sum_{i=o}^{N} w_i x_i \qquad (4.1)$$

then the linear threshold function is

$$Y = \begin{cases} +1 \text{ if } W.X > o \\ -1 \text{ if } W.X < o \end{cases} \qquad (4.2)$$

By considering patterns X to be points in a space (the feature space) having dimensions x_i, $0 \leq i \leq N$, it is easy to see that the ATLE implements a linear decision hyperplane which separates patterns belonging to one class (Y=+1) from patterns belonging to the second (Y=-1). For many real recognition problems the linear decision boundary is not the optimum one. However, the linear boundary has been the subject of many experimental investigations because it is optimum in certain idealized cases, convenient to implement in hardware, and convergent adaptive algorithms exist to search for it iteratively. Additionally, the ATLE classifier can produce useful information, such as the important features in the classification task and the important samples in the training task (Greenberg and Konheim, 1964).

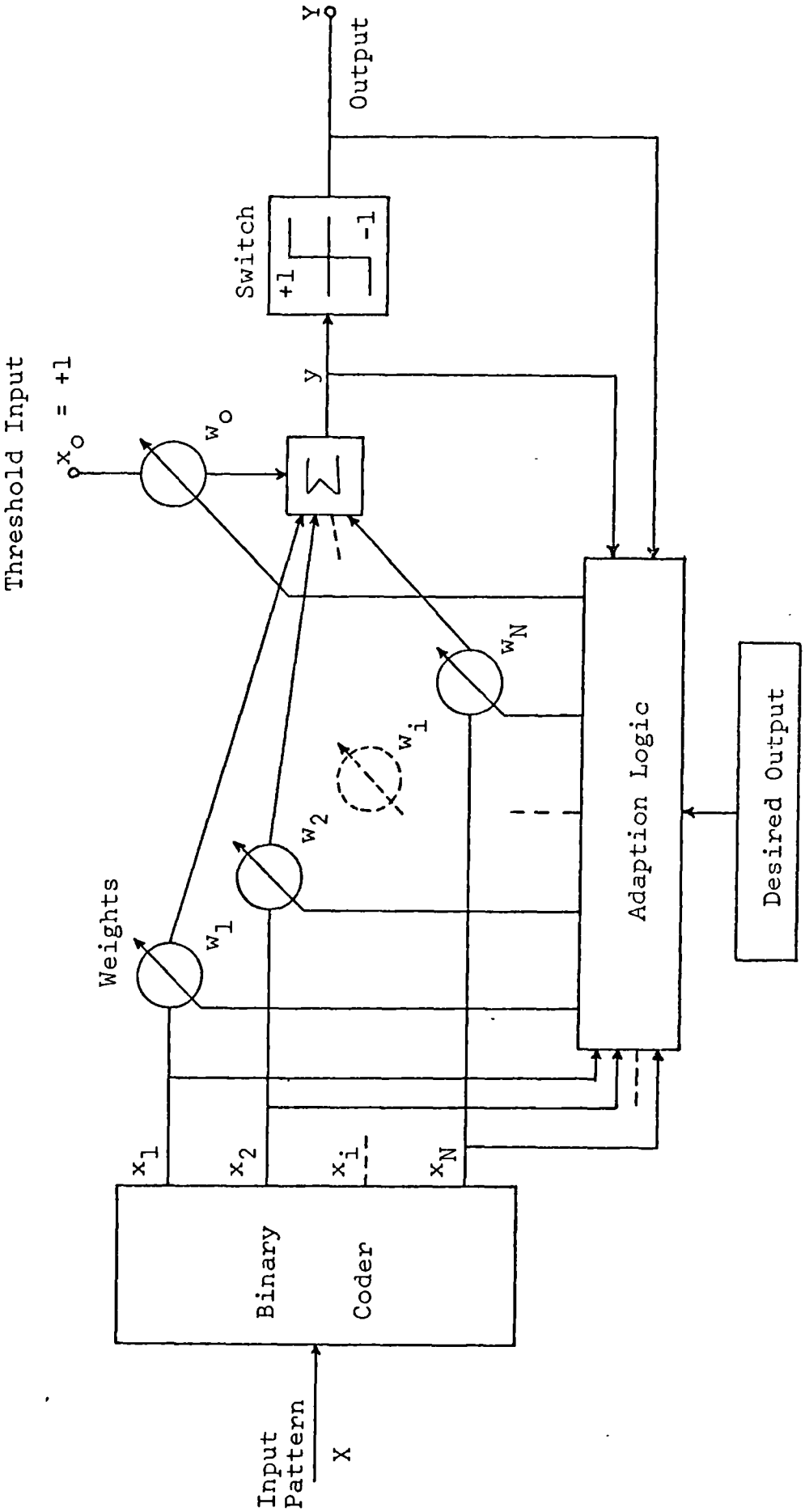


Fig. 4.2 The Adaptive Threshold Logic Element

4.2.3 WEIGHT ADAPTATION PROCEDURES

In Section 4.1.1, the concept of adaption was segmented into two phases; after an initial phase of unsatisfactory performance, the controller becomes satisfactory in the second phase. This transition of behaviour is usually described in anthropomorphic terms as 'training' or 'teaching'. Training is usually a major part of the implementation of an ATLE pattern classifier. The training procedures are iterative processes, whereby patterns from the 'training set' are shown to the classifier one at a time with the desired output, which the weight adjustment procedure tries to match with the actual output of the classifier. It will be assumed for the moment that the desired response is known.

The two well known and most often used weight adjustment procedures (next two sub-sections) can be formulated as hill-climbing procedures (Minsky, 1961). Interpretation of the problem in hill-climbing terms enables a unified approach to a number of seemingly different adaptation procedures (Mays, 1965; Smith, 1969). Furthermore, this interpretation gives insight into a number of problems encountered in complex networks of threshold logic elements by relating such concepts as 'local minima' and the 'mesa phenomenon' (Minsky, 1961; Minsky and Selfridge, 1961; Mays, 1964; Sklansky, 1966).

4.2.3.1. Error Correction Procedure

As the procedure name suggests, training in this case is done only on those patterns in the training set which give erroneous classification. It must be noted that erroneous classification differs from the notion of rejection, where a pattern is rejected by the machine as being unrecognizable if the recognition decision is unreliable in some sense (Highleyman, 1962). This is usually a safeguard against the undefined condition in Eqn.(4.2) when $W.X=0$. The situation can be avoided by putting $\pm\theta$ bounds on either side of the zero threshold, which defines the bounds within which a pattern is rejected (Griffin et al, 1963).

The error correction procedure adjusts every weight component, w_i, by an amount proportional to the corresponding component of the pattern, x_i. That is:

$$
\begin{aligned}
W_{n+1} &= W_n && \text{if decision was correct} \\
W_{n+1} &= W_n - c_n X_n && \text{if decision was incorrect and } W_n . X_n > 0 \\
W_{n+1} &= W_n + c_n X_n && \text{if decision was incorrect and } W_n . X_n < 0
\end{aligned}
\qquad (4.3)
$$

where $c_n > 0$ is called the correction increment, possibly depending on n which describes the iteration steps.

There are several types of error correction procedures in the literature. These differ solely in the interpretation to be given to the value of the correction increment. The one which is most often used for its simplicity is the fixed-increment rule. In this case, $c_n = c > 0$ is taken to be a constant not depending on n. The resulting adjustment may or may not be enough to correct the error for pattern X_n depending on the value of $W_n . X_n$ in relation to c. Usually the value of c is taken to be equal to 1, in which case computation of Eqn.(4.3) simply involves addition or subtraction of the pattern vector X_n from the weight vector W_n. Moreover, recalling that the input pattern has binary components, if the initial weight vector is also chosen to be an integer, including zero, then all succeeding weights will be integers thus making digital implementation very convenient.

The other two rules, the absolute correction and the fractional correction rules (Nilsson, 1965; Fu, 1970), yield non-integer weights. There is no real advantage in employing these rules except that the error for a particular pattern is corrected in a single adjustment step. However, the same effect can also be achieved with the fixed increment rule if the pattern is presented repeatedly to the machine until classification is correct. All three rules will

eventually produce correct classification provided the sets to be classified are linearly separable (Section 4.2.4). Finally, it is noteworthy that originally the above solutions of Eqn.(4.2) were not motivated by studies of learning machines, but by studies of numerical methods for solving linear inequalities (Agmon, 1954; Motzkin and Schoenberg, 1954).

4.2.3.2 LMS Error Criterion

The least-mean-square (LMS) training procedure was developed by Widrow and Hoff (1960). Training here is carried out on all patterns, unlike the error correction procedure. The weight adjustments are such as to minimise the mean-square error,

$$\overline{\varepsilon}^2 = \frac{1}{L} \sum_{\ell=1}^{L} \left[d_\ell - W.X_\ell \right]^2 \qquad (4.4)$$

where L is the number of patterns in the training set and d_ℓ is the desired binary output for pattern X_ℓ. With this criterion the value of the correction increment in Eqn.(4.3) becomes

$$c_n = |\lambda \varepsilon| / X_n.X_n \qquad (4.5)$$

where $\varepsilon = d_n - W_n.X_n$ is the error, and λ is a proportionality constant.

The LMS algorithm will give a unique solution, that is, a unique weight vector W. However, it will not necessarily minimise the number of classification errors even with linearly separable pattern sets, that is, with pattern sets which can be classified correctly by means of a linear decision surface. A good description of the LMS algorithm can be found in (Widrow, 1971). Widrow describes the search for a minimum of expression (4.4) in terms of hill-climbing techniques as mentioned previously. Widrow also estimates the time constant of the learning curve associated with the LMS procedure. However, because this procedure will not necessarily produce the optimum decision boundary even with linearly separable pattern sets, it is not widely used.

4.2.3.3 Selective Bootstrapping

One of the main difficulties encountered in control problems is the question of supplying the learning controller with the desired output for every single input pattern. This problem does not arise in character recognition situations, for instance, where the trainer, who is usually a human, can identify clearly what category (A, B, C, etc.) every input pattern belongs to. The difficulty can be partly solved using the concept of bootstrap learning (Widrow, 1966; Widrow et al, 1973) where a sequence of decisions by the machine are evaluated as opposed to single decisions.

In bootstrap learning, when a sequence of decisions by the machine is rewarded, then every decision in that sequence is rewarded. This is similarly true in the case when a sequence of decisions is punished. Adopting such a policy constitutes an approximate solution to the credit assignment problem (Minsky, 1961). It is an approximate solution since even incorrect (or correct) decisions in a sequence which has been rewarded (or punished) have their probability of occurrence increased (or decreased). Consequently, bootstrap learning takes place at a slower rate than learning with an ideal teacher who provides correct information for every decision made by the machine.

At the end of the last but one paragraph above, it was stated with care that bootstrap learning only partly solves the question of performance evaluation of the learning machine. Although the criterion used for performance evaluation in bootstrap learning is global, as expressed by the time integral in Eqn.(3.4), in control systems where an error-functional performance criterion is adopted, it is only possible to evaluate a sequence of decisions relative to some past sequence. Thus qualification in control problems is in terms of 'better' or 'worse', and not 'good' or 'bad' (Gaines, 1972). This, of course, very correctly reflects the endeavour of searching for the optimum performance. If the optimum performance was known a priori, only then it would be possible to categorically evaluate a sequence of decisions as

'good' or 'bad'. Illustrations of this point can be found in (Hill et al, 1964; Waltz and Fu, 1965; Nikolic and Fu, 1966), where for every possible plant state the controller remembers the past performance of every possible action applied in response to that particular state. Therefore, when requested to act upon a particular state of the plant, the controller chooses that action which has associated with it the best performance thus far, and the experience of this new action is added to its memory.

In any case, bootstrap learning is a good heuristic training method in the absence of a better one. It has been shown that "using selective bootstrap adaptation, a single threshold element is able to learn to play blackjack very well without knowing the rules or the objectives of the game" (Widrow et al, 1973). Bootstrap adaptation is straightforward in blackjack, of course, since an immediate 'win' or 'lose' evaluation is available for every round of play. Another advantageous application of bootstrap adaptation could be in multilayered and more generally connected (nontrivial) networks of adaptive threshold elements, about which very little is known at present in so far as training procedures and convergence of training are concerned (Mays, 1964).

4.2.3.4 A Heuristic Criterion

In an attempt to individualise performance evaluation to single decisions of the learning controller, a simple heuristic criterion has been found to be sufficient for convergence in a particular control problem (Gaines, 1971, 1972; Witten and Corbin, 1973). The criterion adopted in this case is to accept a decision as good if the error modulus, between desired and actual output of the plant, decreases at some given future time. Conceivably such a heuristic is useful when on-line learning of bang-bang control systems is of interest. However, the situation with a multi-class pattern recognizer controller, that is, a plant whose inputs can have more than two values, is more complex and this is discussed in Section 4.5.

4.2.4 CONVERGENCE OF ADAPTATION

The discussion of this section refers to convergence proofs of the error correction adaptation procedures. Convergence of the LMS algorithm has already been discussed, and as regards bootstrap adaptation no theoretical convergence proof exists to date. Bootstrap adaptation relies on the heuristic that when better-than-average performance is rewarded and poorer-than-average performance is punished, then useful learning will be achieved.

It will be convenient to state the convergence theorem first and then discuss the main points of interest. The proof of the theorem, which can be found in the references cited below, is not given here.

Theorem: If there exists a weight vector which will correctly classify all possible patterns, then the fixed-increment error-correction procedure will converge to a solution weight vector (not unique) which will give the correct classification for every pattern.

This theorem, perhaps worded differently by different authors, has been proved over and over in the literature, possibly because it is one of the few tangible points in the 'theory of adaptive threshold elements' (Rosenblatt, 1962; Novikoff, 1963; Nilsson, 1965; Minsky and Papert, 1969). The theorem looks simple at a first glance, however it subsumes a few important assumptions, which are as follows.

First, the theorem assumes that there exists a weight vector which will correctly classify all possible patterns. With a single ATLE of N inputs, in general there are 2^{2^n} different input-output relationships or truth functions by which the N input variables can be mapped into the single output variable, and only a subset of these, the linearly separable logic functions, can be realised by ~~all~~ any possible choices of the weights. Two sets A and B are linearly

separable if the intersection of the convex hulls of sets A and B is the empty set (Greenberg and Konheim, 1964). Furthermore, the points in a feature space which are identified with a particular class by a linear decision function, that is the ATLE, form a convex set (Highleyman, 1962). Hence the connotation of linear separability in the theorem, of the pattern sets that are being trained into the ATLE.

When the assumption of linear separability is not valid, there is no theoretical justification that the weights converge to values which guarantee a minimum number of incorrect classifications among the training samples, although empirical results reported indicate that the adaptation procedure generally does produce errors close to the minimum. Several methods of testing for linear separability have been suggested (Singleton, 1962; Wee and Fu, 1968), however, the general approach is to implement the threshold element and observe if the weights will converge after a period of training. The symptoms that suggest non-separability are oscillations in the weights. The main disadvantage of these tests from the point of view of on-line real dynamic systems,is that it is assumed the training set, with the desired output for every sample in that set, is available.

Second, the theorem does not provide the number of steps of adaptation needed to achieve convergence. The upper bound obtained in the course of the proof of the theorem unfortunately assumes that at least one solution to the problem be known before the length of the training sequence can be estimated. Nagy (1968) has shown this upper bound to be very sensitive by way of a counter-example, in which convergence is achieved in 45 steps as opposed to the theoretically calculated 284 steps by Singleton (1962). In more realistic applications, the number of training steps required is actually exhorbitant; for instance, some 10^4 odd samples were quoted at a colloquium by Witten (1973).

Further assumptions of the convergence theorem are discussed in the following sections. In concluding this section it is noted that although the theorem has been stated for the fixed-increment error correction procedure, slightly modified theorems can also be proved to apply to the absolute and fractional error correction procedures (Nilsson, 1965).

4.2.5 GENERALIZATION

A third postulate in the convergence theorem is that every sample in the set which is to be dichotomized is present in the training set. That is, the training set should contain all 2^N possible pattern vectors. If only a proportion of these, γ, is included in the training set, then the theorem says nothing about the remaining $(2^N-\gamma)$ samples. This assumption is stronger than necessary in fact since the constraint of linear separability implies that having dichotomized a subset of the complete set into A and B, then a new pattern X may not necessarily be assignable at will to either A or B (Gaines, 1971). In other words, training with the set (A+B) and (A+B+X) leads to the same dichotomization, and the ATLE is said to have generalizing property.

There are no theoretical methods for calculating the minimum number of samples to be included in the training set which will cause correct classification of the complete set through generalization. The general rule of thumb is that the training set should be statistically representative of the environment, so that the structure of the recognition network reflects in some sense the structure of the environment with which it must deal. This is a useful heuristic to induce generalizing properties into the ATLE. Otherwise, if a machine is trained on a limited number of samples, then it may converge to a very non-optimum solution because the amount of data is not sufficient to define an optimum solution. The constraints on the training set which are distinct in real dynamic control problems are discussed in Section 4.5.

4.2.6 CONSIDERATIONS OF THE WEIGHTS

Yet another assumption in the ATLE convergence theorem is that the values of the weights should be unbounded in magnitude. Again this condition is stronger than necessary and it can be shown (Gaines, 1967, 1968) that the weights need take only a finite range of values but that the range necessary for convergence is greater than the range necessary for separability. Hence, with bounded weights one runs the risk of not achieving convergence even though the two pattern sets to be dichotomized are linearly separable. An illustration of this

is given in the above references, where adaptation results in limit cycling of the weights without attaining a solution. Gaines also points out that in an ATLE with bounded weights the convergence is a function of the initial values of the weights, unlike the assumption in the main theorem where the initial weights are arbitrary.

Turning attention to the physical realisation of pattern classifiers, the implementation of fixed threshold logic is relatively easy, but adaptive threshold elements place restrictions on the phenomena that can be used to represent the weights. Various techniques for storage have been proposed, including those based on the electrochemical variable resistance cell (Widrow, 1962), the magnetic core (Brain, 1961), the MOS transistor (McConnell and Meadows, 1966) and the voltage of a stabilized capacitor (Smith and Harbourt, 1968), but these have all had their defects in cost, size, reliability or complexity. For want of a simpler and more economical implementation,a current summing device (Highleyman, 1962; Griffin et al, 1963) is suitable, although the latter is only applicable in the implementation of a fixed threshold element.

The basic difficulty in the above devices is that analog weights are considered. It was indicated in the fixed-increment error correction training procedure that integer weights are desirable since these can be fabricated at a low cost and high reliability using digital integrated circuits. It has further been shown (Ide et al, 1968) that digital storage requirements can be relaxed, so that good performance can still be obtained with fewer quantizations of the weights. This supports the fact that the solution weight vector referred to in the convergence theorem is not unique. However, reducing the number of quantizations cannot be considered before the training is complete, as this creates the problem of having bounded weights discussed above.

4.2.7 CODING OF INPUT PATTERNS

It has already been mentioned that the convergence theorem applies only if a solution exists, and the latter does so if and only if the two sets of patterns to be dichotomized are linearly separable. On the other hand, linear separability applies to the pattern vectors generated by the coder in Fig.4.2 and not to the original stimuli, which might not even have numeric connotations.

A method of coding, the linearly independent code (Smith, 1964, 1966), has proved to be particularly useful in control problems. Smith has shown that when the state variables, x_k, of the plant are each encoded using the linearly independent code, a single ATLE will approximate to an arbitrary degree of accuracy switching functions of the form

$$f(x_1 x_2, ---, x_k)=0 \qquad (4.6)$$

provided that no cross-product terms are included in the expression. Expressions containing cross-product terms can be realised by encoding additional analogue variables which are linear combinations of the original state variables. This is a standard method used in other coding schemes as well, since a non-linear decision boundary, that is, a decision boundary between two sets which are not linearly separable in the original sample space, appears as a linear boundary in the space spanned by the products of the original variables (Greenberg and Konheim, 1964). In general, nonlinear combinations other than products yield the same result. Various alternative methods for classifying non-separable sets have also been proposed, for example, by introducing multiple linear planes with some output logic via AND and/or OR gates (Mattson, 1959), and by cascading threshold gates (Cadzow, 1968).

It is an obvious desire to have a pattern vector of minimum dimensions consistent with the separability requirement. From this point of view, although the linearly independent code produces a separable pattern set, it requires a large amount of

storage for the weights which are redundantly used (clearly not all of the 2^N possible combinations of N features occur with the linearly independent code). For instance, to represent a total of 8 patterns, a binary coder requires 3 bits whereas the linearly independent coder requires 8 bits! In general, the linearly independent coder requires as many bits (features) as there are patterns. As a consequence, the ATLE loses its ability to generalise as well. However, this is a price that must be paid in order to achieve the desired classification.

4.2.8 EXTENSION TO MULTICATEGORY CASE

Up to now the properties of a single ATLE have been considered thus limiting their application to dichotomous classification. The extension to multicategory pattern classifiers, by building a network containing more than one ATLE, is more interesting, although the majority of control problems studied in the light of learning machines have been limited to bang-bang systems requiring dichotomous classification.

The immediate problem facing the designer of the multicategory pattern classifier is the number of threshold elements to be included in the network. Three possibilities have been considered, each one being expedient under separate conditions. The first possibility is to have less number of threshold elements than the number of classes. In this case the need arises to code (usually binary code) the output in some manner, which, unfortunately, requires pre-knowledge of the clustering properties, or the topology of the input space with respect to classes. Therefore, the selection of a code is not arbitrary but rather is dictated by the environment. A simple illustration is given in (Mattson and Dammann, 1965). Mattson and Dammann also present a method for determining and coding clusters in an unknown, multi-clustered space. However, one of the disadvantages of their technique from the point of view of control problems, is that it is an off-line procedure requiring all samples to be classified to be available concurrently.

The second method of separating multiple classes is to have C(C-1)/2 threshold elements, where C is the number of classes (Highleyman, 1962). This method, referred to as class-pair separation (Griffin et al, 1963; Greenberg and Konheim, 1964; Ide et al, 1968), is based on the principle that any multiclass problem can be treated as a number of two-class problems involving the separation of each class from the remainder of the universe. Although the use of this method produces good performance, its disadvantage is that a large number of threshold elements are involved not all of which may be necessary.

The third method is a compromise between class-pair separation and the first method mentioned. Referred to as the 1-out-of-C procedure (see references cited above), it employs one threshold element for each class. The implication is, therefore, that each linear plane of each threshold element separates one class from all the others, and at any one time only one of these planes is 'on', giving the class chosen. Inevitably, this method has its disadvantage too, in that it may not be possible to separate one class from all others by means of a single linear plane. Obviously, the convexity or linear separability requirements of the 1-out-of-C procedure are more stringent than those of the class-pair procedure which produces better performance. On the other hand, after a comprehensive experimental study of all the possible code assignments, Ide, Kiessling and Tunis (1968) conclude that the performance of the 1-out-of-C coded classifier is exceedingly close to the performance of the class-pair classifier. In general, the 1-out-of-C code is preferred considering the large number of threshold elements that the class-pair classifier requires.

The most often used weight adjustment method in multicategory situations is the fixed-increment error correction procedure. A convergence proof for the 1-out-of-C coded classifier using the fixed-increment error correction weight adjustment procedure is given in (Nilsson, 1965), which is a slightly modified version of the convergence proof for the single threshold element. As regards the class-pair classifier it is easy to see that its convergence requirements are identical to those of the single threshold element.

4.3 PATTERN RECOGNITION II - DECISION THEORY

4.3.1 RATIONALE OF DECISION-THEORETIC APPROACH .

Statistical decision theory establishes rules for choosing an action based on a set of measurements so that a loss is minimised. This formulation alone evinces the applicability of decision theory in control problems via pattern classifiers.

Mathematically, the decision-theoretic approach is more elegant than that of the threshold element. To be specific, classification in a decision-theoretic learning system is based on a gradually updated estimate of the joint probabilities of recently observed features. The generality of this formulation enables it to handle a wider variety of problems. However, the resulting computational load in computing the joint probability distributions of many variables in a real system could easily overwhelm storage availability, even in a large digital computer. Furthermore, the time required to find these distributions can be excessive (Assilian and Mamdani, 1974a). Because of these detrimental factors there has been a general lack of interest in decision-theoretic pattern classifiers, particularly among control engineers, and the majority of applications considered have been limited to cases where it has been possible to assume simple statistics of the environment.

4.3.2 THE MAXIMUM-LIKELIHOOD CLASSIFIER

Let the R pattern classes be denoted C_i, $1 \leq i \leq R$, each having a priori probability of occurrence $p(C_i)$, where

$$\sum_{i=1}^{R} p(C_i) = 1 \qquad (4.7)$$

Sometimes the categorizer may be allowed to reject a pattern as being unrecognizable if the recognition decision is unreliable in some sense (Section 4.2.3.1). A rejection may or may not be considered as an error and the rejection category will be indexed by zero, as C_0.

Central to the decision-theoretic treatment is the specification of a loss funtion $\lambda(C_i,C_j)=\lambda_{ij}$, $0\leq i, j\leq R$, which represents the loss or cost incurred when the machine places a pattern actually belonging to category Cj into category C_i. The usual case requires that $\lambda_{ij} > \lambda_{io} > \lambda_{ii}$, where λ_{io} is the loss associated with rejection. If the machine classifies patterns such that the average value of λ_{ij} is minimised, then it is said to be optimum with respect to minimising the loss, and is known as a Bayes machine.

Given the conditional probability $p(C_j|X)$, the conditional average loss, denoted $L(C_i)$, is expressed by

$$L(C_i) = \sum_{j=1}^{R} \lambda_{ij} p(c_j|X) \qquad (4.8)$$

By definition a Bayes machine assigns a pattern X to category C_{i_B}, where for $i = i_B$

$$L(C_{i_B}) \leq L(C_i) \text{ for } 1\leq i\leq R \qquad (4.9)$$

The expression for $L(C_i)$ can be simplified using:

Bayes' rule

$$p(C_j|X) = p(X|C_j)p(C_j)/p(X) \qquad (4.10)$$

and the special loss function,

$$\lambda_{ij} = (1-\delta_{ij})\lambda \qquad (4.11)$$

where δ_{ij} is the Kronecker delta function. Although the use of this loss function is primarily for computational ease, in certain applications it is also a reasonable assumption to make; a loss of λ units is incurred for an erroneous classification, but no loss is assumed for a correct classification. Furthermore, it can be shown that a Bayes machine using this loss function minimises also the probability of erroneous classification (Highleyman, 1961; Patrick, 1972).

Substitution of Eqns. (4.10) and (4.11) into Eqn. (4.8) and simple manipulation yields,

$$L(C_i) = \lambda[p(X|C_i)p(C_i)]/p(X) \qquad (4.12)$$

Finally, minimising the above expression for $L(C_i)$ is equivalent to maximising the expression

$$p(X|C_i)p(C_i) \qquad (4.13)$$

A decision based on such a computation is known as a maximum-likelihood decision.

4.3.3 ASSUMPTION OF STATISTICAL INDEPENDENCE

The mathematical derivation in the last section shows that the implementation of the decision-theoretic categorizer requires numerical values for $p(X|C_i)$ and $p(C_i)$. Of course, these probabilities are unknown at the outset and it is the task of the learning controller to estimate their values, the computations of which will depend on the assumptions made about the form of the distributions. It is with respect to these assumptions that the computational load can be excessive. For instance, the most general but trivial case, when there are N binary features, would require 2^N registers merely to store the probability distribution of one class. Clearly, this approach must be rejected outright since it is impractical for large systems, and therefore some simplification of the statistics underlying the environment is necessitated.

The number of distributions to be computed, and the difficulty of the computation itself is greatly reduced if statistical independence between the features $(x_1, x_2, \cdots, x_N)$ that make up a pattern X can be assumed. This assumption is frequently made as a theoretical constraint in the study of decision-theoretic learning classifiers. However, how well the assumption of statistical independence describes many realistic situations is very application dependent and care must be exercised before utilizing it. When it can be assumed, then it is possible to express $p(X|C_i)$ as

$$p(X|C_i) = p(x_1|C_i)p(x_2|C_i)---p(x_N|C_i)$$

$$= \prod_{j=1}^{N} p(x_j|C_i) \qquad (4.14)$$

From expression (4.13), the task of the classifier now becomes that of maximising

$$\prod_{j=1}^{N} p(x_j|\ C_i)p(C_i) \qquad (4.15)$$

or, since the logarithm is a monotonic function of its argument,

$$\sum_{j=1}^{N} \log\ p(x_j|C_i) + \log\ p(C_i) \qquad (4.16)$$

Depending on the application, other assumptions about the statistics underlying the environment are also possible. For example, if it is known that the distribution is Gaussian, then the task of the classifier reduces to that of estimating the parameters of the Gaussian functional form (Nilsson, 1965).

4.3.4 PROBABILITY ESTIMATION PROCEDURES

Similar to the weight adjustment procedures for the threshold element, various methods for estimating the probabilities involved in a decision-theoretic classifier have been proposed.

4.3.4.1 Laplace's Rule of Successions

Laplace's rule of successions (Parzen, 1960) estimates the nth iteration of $p(x_j|C_i)$ as

$$p^n(x_j|C_i) = \frac{1 + \sum_{k=1}^{n} \phi\ (C_i, x_j^k)}{2 + \sum_{k=1}^{n} \psi\ (k, C_i)} \qquad (4.17)$$

where

$$\Phi\ (C_i, x_j^k) = \begin{cases} 1, & \text{if } x_j^k = 1 \text{ in pattern which is classified as } C_i \\ 0, & \text{otherwise} \end{cases} \qquad (4.18)$$

and

$$\psi\ (k, C_i) = \begin{cases} 1, & \text{if } C_i \text{ is chosen} \\ 0, & \text{otherwise} \end{cases} \qquad (4.19)$$

That is, $\sum_{k=1}^{n} \psi\ (k, C_i)$ is the total number of times class C_i is chosen in n training steps. Similarly, the nth iteration of $p(C_i)$ is

$$p^n(C_i) = \frac{1 + \sum_{k=1}^{n} \psi\ (k, C_i)}{2 + \sum_{i=1}^{R} \sum_{k=1}^{n} \psi(k, C_i)}$$

$$= \left[1 + \sum_{k=1}^{n} \psi(k, C_i)\right] / (2+n) \qquad (4.20)$$

By writing Laplace's rule in short as (1+a)/(2+b), then it becomes obvious why it is preferred to the simpler and ordinarily used expression, (a/b), for estimating probabilities. It may happen that even after a long period of training, a = 0 and b ≥1, in which case (a/b) = 0, so that the product in Eqn.(4.15) of the probabilities is zero. Clearly this is not appropriate, and since

$$\lim_{n \to \infty} \frac{1+a}{2+b} \to \frac{a}{b}$$

Laplace's rule of estimating probabilities is better suited in on-line applications. A variant of Laplace's rule has also been suggested (Symons, 1968), however the difference is unlikely to be felt operationally.

4.3.4.2 Heuristic Estimators

Dropping the descriptives in brackets in Eqn.(4.17) and denoting by d the denominator, it is easy to see that an iterative procedure for estimating the probability is:

$$p^{o} = 1/2$$
$$p^{n} = [(d-1)p^{n-1} + \phi]/d \qquad (4.21)$$

The disadvantage of this procedure is that it has to keep record of d. This can be avoided by a simple heuristic. Rewriting Eqn.(4.21) in the form

$$p^{n} = (1-1/d)p^{n-1} + \phi/d \qquad (4.22)$$

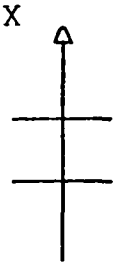
the following procedure is suggested:

$$p^{o} = 1/2$$
$$p^{n} = (1-\alpha)p^{n-1} + \alpha\phi \qquad (4.23)$$

It can be shown (Minsky and Papert, 1969) that p^{n} approaches the true probability as a limit but with the effect of past iterations decaying exponentially, which could be advantageous in some applications. A similar estimator can be given for Eqn.(4.20) with Ψ replacing ϕ. The procedure of Eqn.(4.23) is also discussed in (Minsky, 1961; Minsky and Selfridge, 1961), where it is pointed out that essentially the procedure is that used in 'reinforcement learning' (Bush and Mosteller, 1955; Estes, 1959).

4.3.5 CODING OF INPUT PATTERNS

A caution was given to treat the property of statistical independence carefully and this point is elaborated here. When a state variable of a plant in a control problem is encoded using the linearly independent code, it can no longer be assumed that the generated features in this fashion are statistically independent. For example, consider the case where a state variable is quantized into three levels and the single-spot linearly independent code is used:



It is easy enough to see that $p(X) \neq p(x_1)p(x_2)p(x_3)$, but simply:

$$p(X) = \begin{cases} p(x_1) & \text{when } X = 001 \\ p(x_2) & \text{when } X = 010 \\ p(x_3) & \text{when } X = 100 \end{cases} \qquad (4.24)$$

When the linearly independent code is used, statistical independence can only be assumed between any two state variables, X_1 and X_2, which have been combined to form the input pattern X. Thus,

$$p(X) = p(X_1)p(X_2) \qquad (4.25)$$

but again each partial vector, X_1 or X_2, follows the rules in Eqn.(4.24). In the general case, where there are M state variables quantized into N_1, N_2, ---, N_M levels, the input pattern X is composed of $N_1 + N_2 + \text{---} + N_M$ features,

$$X = (X_1; X_2; \text{---}; X_M)$$

$$= (x_{11}, x_{12}, \text{---}, x_{1N_1}; x_{21}, x_{22}, \text{---}, x_{2N_2}; \text{---}; x_{M1}, x_{M2}, \text{---}, x_{MN_M}) \qquad (4.26)$$

and the probability of the input pattern is

$$p(X) = p(X_1)p(X_2)\text{---}p(X_M)$$
$$= p(x_{1a_1})p(x_{2a_2})\text{---}p(x_{Ma_M}) \qquad (4.27)$$

where a_i corresponds to that position in the partial vector X_i such that $x_{a_i} = 1$. It must be noted that the number of terms in Eqn.(4.27) is a great deal less than it would otherwise be - see Eqn.(4.14). Consequently, use of the linearly independent code reduces computational load in terms of computation time, but the use of storage is still excessive.

Similar considerations must also be applied to the multi-spot code, and an example of the misuse of statistical independence when linearly independent coding is used can be found in (Freedy et al, 1971).

4.4 THRESHOLD ELEMENT VERSUS DECISION THEORY

Although there seems to be little in common, at least in mathematical representation, between an adaptive threshold element and a Bayes pattern classifier, under certain circumstances the structure of the classifier can be shown to be identical for both cases. Any disparity arises only out of the different methods of determining the values of the components in that structure.

4.4.1 AN EQUIVALENCE

When the features comprising the input pattern to the classifier can be assumed to be statistically independent, and when the features take on binary values, it can be shown that the decision boundary of a Bayes classifier is linear, same as that of the threshold element. This striking equivalence has been pointed out often in the literature (Minsky and Selfridge, 1961; Papert, 1961; Minsky and Papert, 1969; Assilian and Mamdani, 1974a). An excellent exposition is given in (Chow, 1965).

To give a brief outline, let

$$p(x_j=1|C_i)=p_{ji} \tag{4.28}$$

$$p(x_j=0|C_i)=1-p_{ji}=q_{ji} \tag{4.29}$$

Assuming statistical independence, expression (4.16) can then be reduced to

$$\sum_{j=1}^{N} x_j \log(p_{ji}/q_{ji}) + \sum_{j=1}^{N} \log q_{ji} + \log p(C_i) \tag{4.30}$$

which is equivalent to the linear threshold function

$$\sum_{j=1}^{N} x_j w_{ji} + w_{oi} \qquad (4.31)$$

where $w_{ji} = \log(p_{ji}/q_{ji})$ (4.32)

$$w_{oi} = \sum_{j=1}^{N} \log q_{ji} + \log p(C_i) \qquad (4.33)$$

4.4.2 DIFFERENCES

Even though the decision boundary of a Bayes classifier is linear under the assumption of statistical independence the orientation or the positioning of this boundary is different than that of the threshold element because of the different ways of determining the weights associated with each boundary. The Bayes boundary tends to position itself as a perpendicular bisector to the line joining the centres of gravity of the two classes of points (Minsky and Papert, 1969). Hence it is possible for the Bayes classifier to make some errors even when the two sets of points are linearly separable. On the other hand, when the two sets are not linearly separable, the Bayes classifier is known to minimise the number of errors, whereas very little is known how the ATLE will behave. It was mentioned, however, that empirical results indicate that in general the latter will adapt to a configuration which also gives an almost minimal number of errors. When no prior knowledge exists about separability, it seems the safer approach is to use a Bayes classifier. Nevertheless, the simplicity of the threshold element appears to have had more appeal to most workers in the field.

4.5. TRAINING OF PATTERN RECOGNIZERS

Up to now very little has been said about the training aspects of pattern recognizers, and when control problems are of interest, two main problems can be isolated; first, the configuration to be used involving the plant, controller and trainer, and second, the choice of the trainer itself.

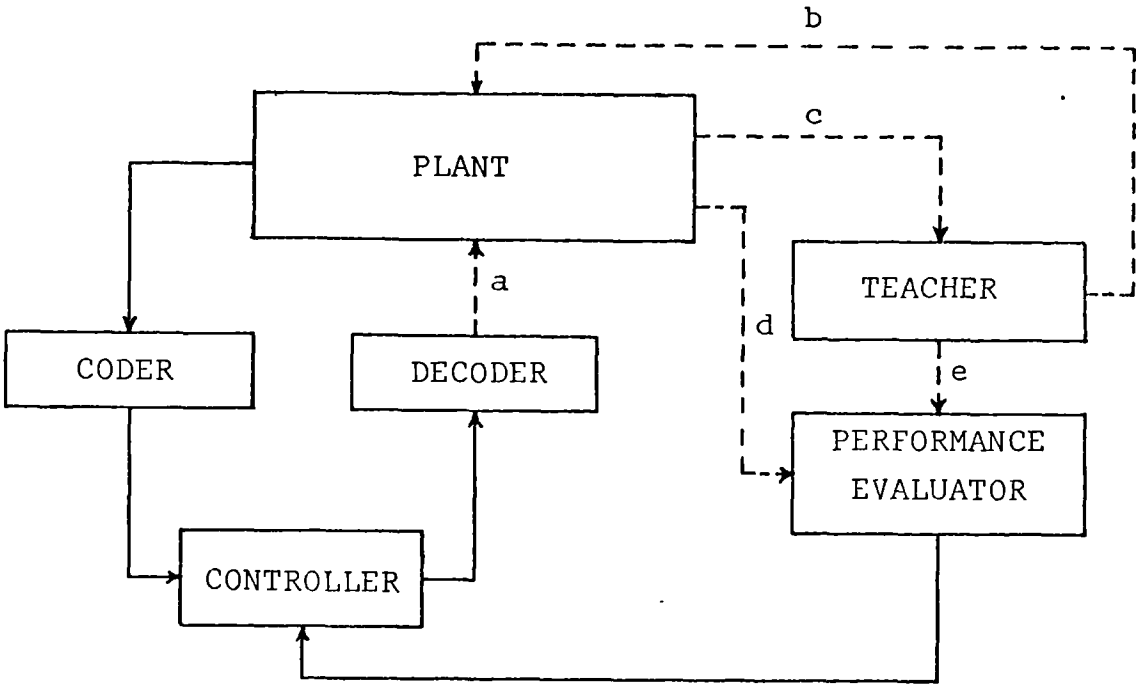


Fig. 4.3 Training Configurations

4.5.1 OPEN- AND CLOSED-LOOP TRAINING MODES

Considerations of the configuration between plant, controller and trainer lead to two main types of training, open- and closed-loop. The distinguishing feature between the two modes of training is the loop-closing link between plant and controller as shown by the dotted line a in Fig.4.3.

When there is no feedback from the controller to the plant, so that an independent source is influencing the patterns generated by the plant, the mode of training is called open-loop (configuration b-c-e). The control actions applied to the plant in this case are usually those of the teacher. With this configuration it is possible therefore for the teacher which would usually have to be the human operator, to give the controller a 'guided-tour' of the entire state space of the plant, thus providing a typical or a statistically representative training set necessary to produce an optimum controller. Where this cannot be done, for example, with plants which are

already in production, then the generalising properties of the controller become important. Similar views related to these points are discussed at length by Gaines (1971, 1972) as the 'sub-environment' phenomenon. Effectively, the behaviour of the controller in the open-loop configuration can be looked upon as that of mimicing the actions of the trainer, and this situation can give rise to a ridiculously sub-optimal controller if the trainer is non-optimal itself.

In the closed-loop configuration where the controller is part of the control loop, two possibilities arise depending on whether a teacher is also included (configuration a-c-e) or not (configuration a-d). When a teacher is included to guide adaptation through performance evaluation, the problem of producing a sub-optimal controller still exists, and even with an optimal teacher, for example, in the case of a fixed optimal controller acting as the teacher, it may be forced to operate outside its normal range of inputs thus again showing sub-optimal behaviour. On the other hand, when the criterion for adaptation is based on typical performance measures such as minimisation of some error functional over an interval, the designer is faced with the problems associated with localising performance evaluation to single actions of the controller, as discussed in Section 4.2.3.3.

4.5.2 SUPERVISED TRAINING

4.5.2.1 A Fixed Controller as Trainer

An obvious choice for the teacher in Fig.4.3 is to use an already existing fixed controller which is known to operate successfully on the plant. It would be reasonable to ask at this point the reason for wanting a second controller when one already exists. However, the use of a fixed controller to train an adaptive one is normally encountered in feasibility studies of learning machines where the objective of the investigation is the adaptive capabilities of such machines. It is still conceivable though for the need to arise for replacing a fixed controller by an adaptive one. For example, it can happen that theoretical design efforts yield a fixed optimal controller which is too expensive, or even impossible to realise in hardware. In such cases, adaptive pattern-

recognising controllers offer an alternative solution since they can be cheaply implemented after an off-line training phase carried out in a general purpose digital computer.

A few of the problems associated with the use of fixed controllers to train adaptive ones have already been mentioned. One of the main disadvantages is that fixed controllers are usually designed to be optimal only in a very small region of the entire operating space of the plant. This region is the immediate neighbourhood of the normal operating point of the plant, and if large changes in the operating point are to be demanded it becomes necessary to design a completely new controller with different characteristics. This has the consequence that when the same fixed controller is used for training in the entire operating space of the plant, it will teach non-optimal control policies for the most part of that space.

Another disadvantage of a fixed controller, which is being assumed here to be designed using classical or conventional control theory, is that usually their inputs are analogue signals, whereas the inputs of most adaptive controllers are quantized values of these analogue signals. This difference between the two can result in the fixed controller giving contradictory teaching information within the same quantized region of the input space (see Chapter 5).

4.5.2.2 Human Operator As Trainer

Having a human operator as the trainer solves some of the problems mentioned in the last section since, in general, the human can be more flexible. The human operator also offers the only alternative when no exemplifying fixed controller is available. As of this date, the human has played an important role in studies of learning machines. In character recognition, speech recognition and studies of the like, the human has exclusively played the role of the teacher. Similarly, in the majority of work carried out within the wide scope of AI, for example, game-playing and theorem-proving situations, the influence of the human has been extensive.

The case is somewhat different in control situations specially when the control of real dynamic plants is concerned, where transient responses (dynamics) rather than end-states are of importance. Static manipulatory control tasks are more readily formulated and studies have been carried out where the human operator has successfully played the role of the teacher (Whitney, 1969; Freedy et al, 1971). However, the environment when the human is required to control transient response is more complex, and his behaviour less predictable and less understood. The problems arising in these circumstances are discussed in more detail in Chapter 5 in the light of the experience gained in the training of adaptive controllers for the steam engine.

4.5.3 UNSUPERVISED TRAINING

Undoubtedly, supervised training can create many problems whether the trainer is another machine or a human, and as an alternative solution unsupervised training has been suggested where the controller guides its own learning through some performance index (configuration a-d in Fig.4.3). An immediate advantage of unsupervised training, therefore, is that labelling of patterns is eliminated. In fact, unsupervised training is usually considered when this labelling information is not available. Another advantage is that unsupervised learning endows the controller with tracking ability, that is, the controller is capable of tracking any changes in class distributions due to data changes or hardware degradation (Nagy, 1968). This is because learning in this mode of training can be carried on ad infinitum unlike the case in supervised mode, where after an initial phase of learning, teaching is usually discontinued.

Nevertheless, the unsupervised training mode has its disadvantages too, particularly in the case of a multi-class system, for example, the steam engine in this study. The discussion below is mainly concerning such systems. The two main problems in this case are the specification of an appropriate performance index and the size of storage required to hold the information necessary for the learning process, which otherwise is 'remembered' by the teacher.

The difficulty that arises in evaluating the performance is due to the credit assignment problem (Minsky, 1961) as discussed in Section 4.2.3.3. In two-class systems, this problem can be solved using simple heuristic criteria, for instance, the one described in Section 4.2.3.4. In multi-class systems, however, the situation is not as simple since at every sampling instant, or for every possible state of the plant, one has the choice between more than two actions, so that the effect of each of these actions need to be evaluated in order to choose the optimum. Such unsupervised training schemes have been suggested and limited simulation results have indicated the viability of the scheme (Hill et al, 1964; Waltz and Fu, 1965; Nikolic and Fu, 1966). The 'local' performance index that is specified in all three studies cited above is heuristic in nature and it is suggested (Waltz and Fu, 1965) therefore to have a separate 'learning loop' to search for the optimum performance index. This, of course, is reminiscent of the hierarchic structure of the general learning system. More to the point though, the inclusion of a secondary learning loop can increase the learning process to an unacceptably long period in practice, even though its presence is seen to be important, if not necessary. Indeed, the learning rate in the unsupervised mode is in itself very slow without this additional loop (Nikolic and Fu, 1966).

The second problem in unsupervised learning arises because of the need to retain vast amount of information about every possible state of the plant. The formulations of the training schemes given in the above cited references indicate clearly the vast space of memory required to deal even with small systems. Again, the problem is not as serious with two-class systems as it is with multi-class systems.

To conclude, in general the complexity of the unsupervised mode of training cannot be reduced even though the scheme is known to be feasible. A good discussion of the inherent problems in unsupervised learning is given by Spragins (1966). He points out that theoretically developed schemes (Hilborn and Lainiotis, 1969) very often cannot be implemented in practice because of their highly complex formulation.

4.6 CONCLUSIONS

Two important techniques in adaptive or learning control have been presented and discussed in detail in this chapter. These techniques have been criticised particularly in the light of what might be expected with real systems. Possible training modes for such controllers have also been presented and discussed. In retrospect, it is obvious that a large and complex set of interrelated problems exist whatever the configuration adopted for training. It would be virtually impossible to try and associate each problem mentioned in the chapter with only one configuration. Further discussion is given in the next chapter where a few particular training schemes are considered for the control of the steam engine.

CHAPTER 5

ADAPTIVE CONTROL OF THE STEAM ENGINE

5.1. INTRODUCTION

The lengthy review presented in the last chapter of the techniques available in the art of adaptive control was in preparation for possible experimentation to be carried out on the steam engine. Classical control methods proved the engine as controllable and now the way was open to test the applicability of adaptive control methods. The true value of these tests lie in the fact that a real dynamic system is being used in assessing the effectiveness of adaptive controllers. Apart from this fact, no innovation is claimed in any of the learning controllers, which have been extensively studied, mostly through simulation work, in the literature.

5.2 EXPERIMENTAL CONSIDERATIONS

5.2.1 INPUTS TO THE ENGINE

It is recalled that both the heat and throttle inputs to the plant are quantized into 32 and 10 levels respectively. Because the adaptive controllers to be described consider errors and changes in errors of the state variables, similar to the direct digital controller of Chapter 3, the actions applied by these controllers were accordingly changes in heat and throttle. The heat input was allowed to take $0 \rightarrow \pm 7$ steps, and the throttle $0 \rightarrow \pm 2$ Steps. The choice of using these values was made after observing that the direct digital controller also confined its actions within these ranges. Of course, any action demand by the controller which would take the inputs beyond their maximum limits, for example, a +6 step demand on the heat input which is already at absolute level 30, is detected by the software and truncated to the maximum possible (+2 in the above example).

5.2.2 OUTPUTS FROM THE ENGINE

The state variables monitored by all the adaptive controllers are the very same ones considered by the direct digital controller, that is, pressure error (PE), change in pressure error (CPE), speed error (SE) and change in speed error (CSE). Naturally, the errors refer to the difference between the value of a state variable and the set point for that variable. And a change in error is defined as the difference between the magnitudes of error in a variable at either end of a sampling interval. Since the sampling interval is kept constant, this information is equivalent to having the first derivative with respect to time, that is, velocity of a variable.

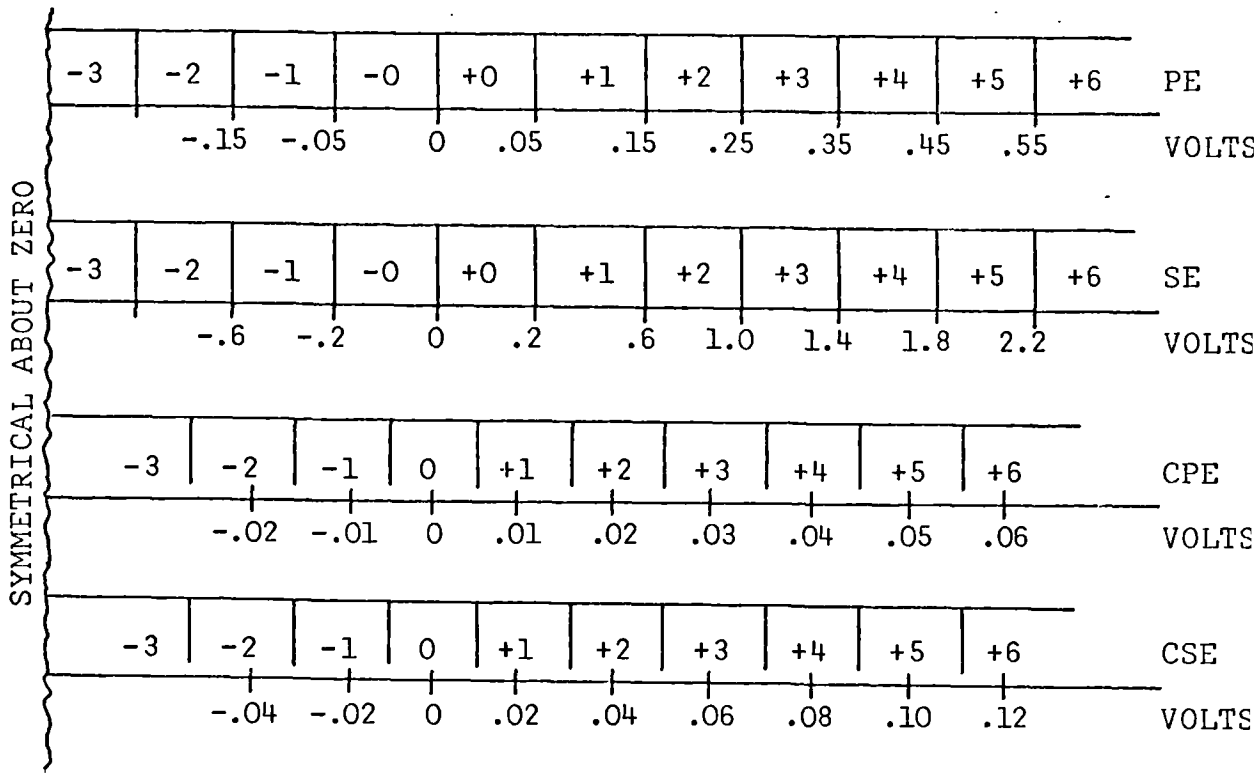


Fig 5.1 PE, SE, CPE, CSE Definitions
(Not To Scale)

The pressure and speed set points are defined at the beginning of an experiment and using these values the software quantizes the PE and SE variables into 14 points. Similarly, the CPE and CSE variables are quantized into 13 points as shown in Fig.5.1. Because a change in error has been defined as the difference between magnitudes of errors, a negative change indicates a velocity directed towards the set-point, and a positive change indicates a velocity pointing away from the set-point. Note also that the zero states of the PE and SE variables have been further divided into +0 and -0 states about the set-point in an effort to present inputs to the adaptive controllers as possibly the same as those presented to the direct digital controller.

5.2.3 SAMPLING

The time interval of sampling the state of the plant was chosen as 10 secs. The choice of this value was reckoned as suitable after considering the time constants of the pressure and speed variables, as established in the modelling of the plant. A second influence in this choice was the desire to make timing compatible with that of the direct digital controller, which is included as the teacher of the adaptive controller in one of the experiments. Using a different sampling interval would have necessitated changing the already established parameters of the direct digital controller since they depend on the sampling period.

5.3 EXPERIMENT I: HUMAN OPERATOR AS TEACHER

Two series of experiments were conducted where the human operator took on the role of the trainer. The first series involved the training of a Bayes controller for the engine and the second that of an ATLE-network controller. Because the observations made were similar for both controllers, only the training of the Bayes controller will be described here.

The irput pattern to the learning controllers consisted of PE, CPE, SE and CSE where each variable was encoded using the single-spot linearly independent code. Thus each input pattern was a (14+13+14+13)=(54)-dimensional vector. Two probability matrices were defined next, the first (54x15) matrix for heat control and the second (54x5) matrix for throttle control. These probabilities were calculated using Laplace's Rule of successions as given by Eqn.(4.17). Two probability vectors were also defined to keep record of the probability of applying each possible heat and throttle input. These probabilities were calculated using Eqn.(4.20). The output of the Bayes controller was calculated using a similar expression to (4.16); the actual expression used accounted for the arguments advanced in Section 4.3.5 concerning the interrelationship between statistical independence and the linearly independent code.

The state of the plant was displayed to the human operator every 10 secs. on a teletype and in the form of numbers. At first, all four variables PE, SE, CPE, CSE were displayed but the operator found this as too much information to absorb in the given time. Also, because of the quantized nature of the displayed data, and specially so in the case of CPE and CSE, the operator had great difficulty in getting a 'feel' for the dynamics of the plant. For this reason, only PE and SE were displayed to him on the teletype and at the same time he was allowed to observe the digital voltmeter, which monitored the pressure and speed variables on a time-shared basis of 5 secs. each, in order to estimate the velocity of the two variables.

Finally, the actions applied to the plant were those of the human operator as shown by the configuration b-c-e in Fig.4.3. In order to present a statistically representative set of training patterns to the learning controller, training runs were systematically started from different points in the state space.

5.3.1 EXPERIMENTAL RESULTS

In anticipation of the conclusion for this series of experiments, very poor convergence was exhibited by the Bayes controller even after many days of training. For this reason alone, instead of presenting the final, and less interesting, poor quality of control learned by the Bayes controller by way of state-trajectory diagrams which compare very unfavourably with those of the direct digital controller, results will be presented and discussed which point at some of the reasons accounting for the discouraging outcome.

The main reason which can be attributed to this discouraging result was the inconsistency of the human operator in his actions or control policies. Three separate runs performed on different days are shown in Fig.5.2.

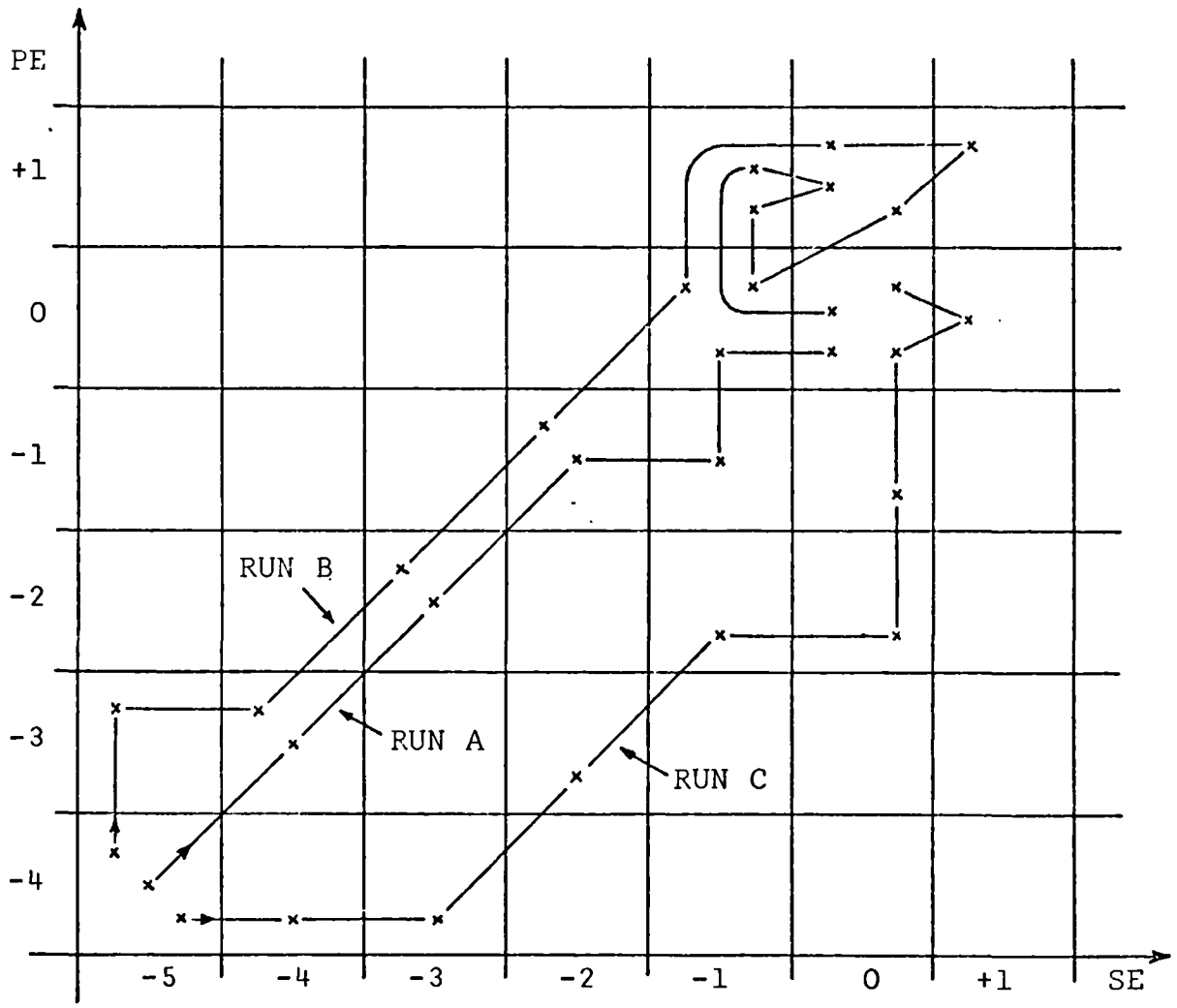


Fig. 5.2 Teaching Trajectories

SAMPLING TIME	PE	SE	OPERATOR HEAT	BAYES HEAT	OPERATOR THROTTLE	BAYES THROTTLE
0	-4	-5	-	-	-	-
1	-4	-5	+7	+3	+2	+2
5	-3	-4	-1	-1	0	0
6	-3	-4	-1	+1	0	0
8	-2	-3	-2	-1	0	0
12	-1	-2	-1	0	0	0
15	-1	-1	-1	0	-1	0
18	0	-1	-1	0	0	0
21	0	0	0	+3	0	0
25	0	0	-1	+3	0	0
29	0	0	-1	+4	0	0
32	0	+1	-1	+3	-1	-1
33	+1	0	-1	-5	0	0
36	+1	+1	-4	0	0	0
37	+1	0	0	+7	0	0
38	0	0	+3	+2	0	-1
39	0	0	-1	+3	0	+2
47	0	0	0	+4	-1	+1
50	0	-1	0	0	0	+1
52	0	-2	+1	0	+1	+1
53	0	-1	0	0	0	0
54	0	0	0	-1	0	+2

Table 5.1 Sample Run Data (Run A Fig.5.2)

They all start with the same state of the plant and yet quite different trajectories are followed in each run, to say nothing about the quality of control. Although run A in Fig.5.2 appears to be of good control quality, the run data given in Table 5.1 indicate in what way this is not true. To elaborate further on the inconsistency of the control policies of the human operator (the author, in this case), a protocol taken at the time run A was conducted is given below.

Sampling Time	Protocol
1	"Both P(pressure) and S(speed) are way below their set points - maximum action."
5	"They are coming up now - decrease H(heat) by one to avoid overshoot - leave T(throttle) as it is since S still low."
6	"Decrease H by one."
8	"P going up fast - will overshoot - decrease H by 2."
12	"P still going up - take one from H - let S go up still."
15	"Decrease H another step - S almost at set-point - decrease T by one."
18	"P is there (at set-point) - but still creeping up - take one."
21	"Both there - no action."
25,29	"P still creeping up (this information is from observing the voltmeter) - take one from H."
32	"P still moving - minus one on H - S has overshoot now - minus one on T."
33	"P has overshoot - have decreased H so many steps! - take another one - S O.K."
36	"P still above set-point! - take 4 this time from H - S has overshoot again - but since P is high now, I think S will come down with P - leave T as it is."
37	"P still one above (set-point) - but coming down - no H action - S O.K. as hoped."

38 "P at set point but going down too fast now - take plus 3 on H."

39 "Maybe plus 3 was too much on H - before I decreased H so much and P still didn't come down - so take one from H."

47 "P is at lower edge (of quantized zone - observed on voltmeter) - S is at upper edge and going up slowly to overshoot - will try to rectify both by controlling T - take one."

50 "S one below (set-point) but will wait in case it comes up again with P since P should increase with my closing of T."

52 "Didn't work - S too low now - to recover quick increase both H and T by one."

53,54 "Both P and S doing O.K. - no action."

5.3.2 DISCUSSION

Little need be said about the inadequacy of the human operator in his duty as the teacher. His control policies were found to be so varied at times that the Bayes controller was left in utter confusion. The situation was very much the same in the training of the threshold logic controller. Although the human operator knows the basic principles that govern the operation of the steam engine, he finds it very difficult to convert his qualitative knowledge into numerical quantities. From the point of view of the adaptive controllers this constant variation in action for the same plant state is unacceptable. The convergence theorem for the ATLE assumes that every pattern in the training set is labelled with only one category that it belongs to. A few mistakes in this labelling can be tolerated and averaged out even though the training time is increased correspondingly. However, inconsistency beyond a certain limit will not produce any systematic convergence in the ATLE. Similar arguments apply to the Bayes controller.

There are several factors that can influence the performance of the human operator and some of these problematic areas were brought out in the experiments. The first problem is the manner in which information about the plant is displayed to the human operator. With single-input, single-output systems this is usually done by displaying the state of the system as a spot on a cathode-ray tube, with the two axes corresponding to the position and velocity of that state variable. This sort of visual aid is suited best to the human to follow the dynamics of a system. However, in the case of the steam engine, and multi-variable systems in general, a display of this nature could not be utilized and instead information had to be presented in a discretized form. Unfortunately, this form of information is not as effectively absorbed and can cause degradation in the performance of the human operator (Witten and Corbin, 1973).

The protocol given in the results exemplifies a second problem facing the human operator, which is his constant uncertainty about the response time of the plant for his actions. When the state of the plant is displayed visually to the human, empirical studies have shown that there is an optimum frequency, around 0.7 Hz, at which he can best follow a motion (Hall, 1963; Sugie, 1971), and that at very low frequencies the human loses his sense of continuity given him by his short term memory. Clearly, such factors come into play irrespective of the form of display adopted. In simulation studies it is possible, of course, to match the dynamics of the system near to the optimum frequency for the human. However, with a real plant there is nothing that can be done but be faced with the natural dynamics of that particular plant, and most industrial plants actually have a slow reaction time as illustrated even by the model steam engine.

Another important problem facing the human operator is fatigue, which again may effect his performance after a certain period depending on the amount of concentration demanded by the control task. Because of the slow dynamics of the steam engine runs lasting many hours and many days had to be performed if any convergence was to be achieved. Under these circumstances fatigue very often gave way to frustration which in turn encouraged drastic changes in the control policy

adopted by the operator in an effort to accomplish the control task with the least effort and in minimum time. A few other and similar problems associated with the human controller have been extensively studied by Gaines (1971, 1972).

5.4 EXPERIMENT II: DIRECT DIGITAL CONTROLLER AS TEACHER

The problems and difficulties discussed in the last section in connection with the human operator can be bypassed if a fixed controller takes the role of the teacher in the training of the adaptive controllers. This was the next natural step to take in this study and further two series of experiments were carried out involving the training of the Bayes and threshold logic controllers. Results are presented only for the case of the Bayes controller again since similar observations were made in both cases. The fixed controller used was the direct digital controller developed in Chapter 3 and the training configuration was the same as that used with the human operator (configuration b-c-e of Fig.4.3).

5.4.1. EXPERIMENTAL RESULTS

The results obtained in these experiments were more successful as regards inducing some learning into the adaptive controller. However, even after a very long training period, the Bayes controller still exhibited indecisiveness and constant fluctuations in its parameters. This was the result of incompatibility between the operational natures of the direct digital controller and the Bayes controller.

Two sample run data are given in Tables 5.2 and 5.3. Both runs show that, in general, the output of the Bayes controller roughly follows the trainer's actions; not always the same magnitude is predicted but something close and at least in the same direction. Closer examination reveals the cause for these disparities. For example, the seventh input pattern in run A and the seventh in run B are identical and yet the direct digital controller has applied different throttle actions. This has the same detrimental effect as the inconsistency of the human operator. However the incon-

PE	CPE	SE	CSE	DDC HEAT	BAYES HEAT	DDC THROT.	BAYES THROT.
+6	+5	-4	-2	-5	-6	0	+1
+6	+6	+4	-2	-7	-5	-1	-1
+6	+2	+4	-6	-6	-4	0	-1
+6	-3	+4	+6	-3	-2	-1	-1
+6	-6	+4	-6	-2	-2	-1	-1
+5	-6	+4	-6	0	-2	0	-1
+4	-6	+3	-6	+1	+1	+1	-1
+3	-6	+3	-6	+1	+1	0	-1
+2	-6	+3	-1	+1	+1	0	-1
+2	-5	+2	-2	+1	+1	-1	-1
+1	-4	+2	-6	0	+1	+1	+1
+1	-3	+1	-6	0	+1	+1	+1
+1	-3	+1	-1	+1	0	0	0
+1	-3	+1	-1	+1	0	0	0
+0	-2	+1	-2	0	+1	0	0
+0	-2	+1	+2	+1	+1	0	0
+0	-1	+1	+1	0	0	-1	-1
+0	-1	+1	-5	0	0	+1	+1
-0	+1	+1	+3	+1	+1	-1	-1
-0	0	+1	-2	0	0	0	0

Table 5.2 Sample Run Data A

PE	CPE	SE	CSE	DDC HEAT	BAYES HEAT	DDC THROT.	BAYES THROT.
+6	+6	+4	+4	-7	-6	0	-1
+6	+5	+4	+1	-6	-6	-1	-1
+6	+0	+4	-4	-5	-3	0	0
+6	-3	+4	0	-3	-6	-1	-1
+6	-6	+3	-6	-1	-2	0	-1
+5	-6	+3	+2	0	-1	0	-1
+4	-6	+3	-6	+1	+1	0	-1
+3	-6	+3	-4	+1	+1	-1	0
+2	-6	+2	-6	+1	+1	+1	-1
+2	-5	+2	-6	+1	+1	0	-1
+1	-4	+2	-1	0	+1	0	0
+1	-3	+2	+5	+1	0	-1	-1
+1	-3	+2	-3	0	+1	0	0
+1	-2	+2	+2	0	+1	0	+1
+0	-1	+2	-6	0	0	0	-1
+0	-2	+1	-1	+1	0	0	0
+0	-2	+1	-1	+1	0	0	0
-0	+2	+1	-1	+1	+1	0	0
-0	+1	+1	-1	0	+1	-1	0
-0	0	+1	-6	0	0	+1	-1

Table 5.3 Sample Run Data B

sistency in this case is on a much smaller scale and therefore some learning has taken place on behalf of the adaptive controller.

The inconsistency in the direct digital controller is because of the nature of its inputs and not because of changes in control policy, as the case was with the human operator - notice the almost identical PE and SE state trajectories in Tables 5.2 and 5.3. The inputs to the direct digital controller are analogue signals as opposed to quantized values, and this makes it sensitive to changes in variables even within a quantized region. Therefore, although the seventh samples in both runs appear as the same input pattern to the Bayes controller, in the eyes of the direct digital controller these two samples must have looked different. On these grounds, further experimentation involving the direct digital controller as the teacher were discontinued since it could not represent the ideal teacher desired.

As a final and side remark, these experiments bring out one noteworthy point about evaluation of results in adaptive learning controllers. It is observed from run A in Table 5.2 that only 50% of the heat actions of the Bayes controller agree with those of the direct digital controller. On the other hand, judging from the closeness of the two columns of actions, the quality of control of the former is very unlikely to be only half as good as that of the latter. Therefore in control studies, qualitative evaluation is more meaningful and informative than quantitative evaluation, and evaluation of results in terms of "% correct recognition", as is usually done in character recognition for instance, should be avoided.

5.5 EXPERIMENT III: FUZZY LOGIC CONTROLLER AS TEACHER

The experiments to be described next were carried out after the development of the fuzzy logic controller (see the next chapter), but the results are presented at this point in the thesis since they concern the training of adaptive controllers. Anticipating the success of the fuzzy logic controller, the reason for conducting further experiments with

the adaptive controllers was to find out how well they can perform with an ideal teacher. The main difficulty with both the human operator and the direct digital controller was inconsistency in teaching. The human operator was unsuitable because he changes his control policy too often. The direct digital controller has a fixed control policy but disparities still arose because of the analogue nature of its inputs. The fuzzy logic controller has a fixed control policy and uses the same quantized inputs as both the threshold logic and Bayes controllers. For this reason, this series of experiments were deemed important.

5.5.1 TRAINING OF BAYES CONTROLLER

Training in this series of experiments was carried out off-line where the procedure was to generate all the possible states of the plant in a cyclic manner and use these as the training samples. Training with one complete set of all the possible states will be referred to as "one cycle of training". After every cycle of training, the complete set of inputs were again presented to the controller, this time without any adaptation, and the number of disagreements in its predicted actions with those of the fuzzy logic controller were calculated as a percentage. This percentage will be referred to as "cyclic percentage". A "running percentage" of misrecognition, calculated on a per sample basis as training was taking place, was also recorded in every training cycle.

The control policies of the fuzzy logic controller are shown in Figs.5.3 and 5.4. After only one cycle of training the Bayes controller reached its final form and its respective learned control policies are shown in Figs. 5.5 and 5.6. The cyclic and running percentages of misrecognition in heat actions are shown in Fig.5.7.

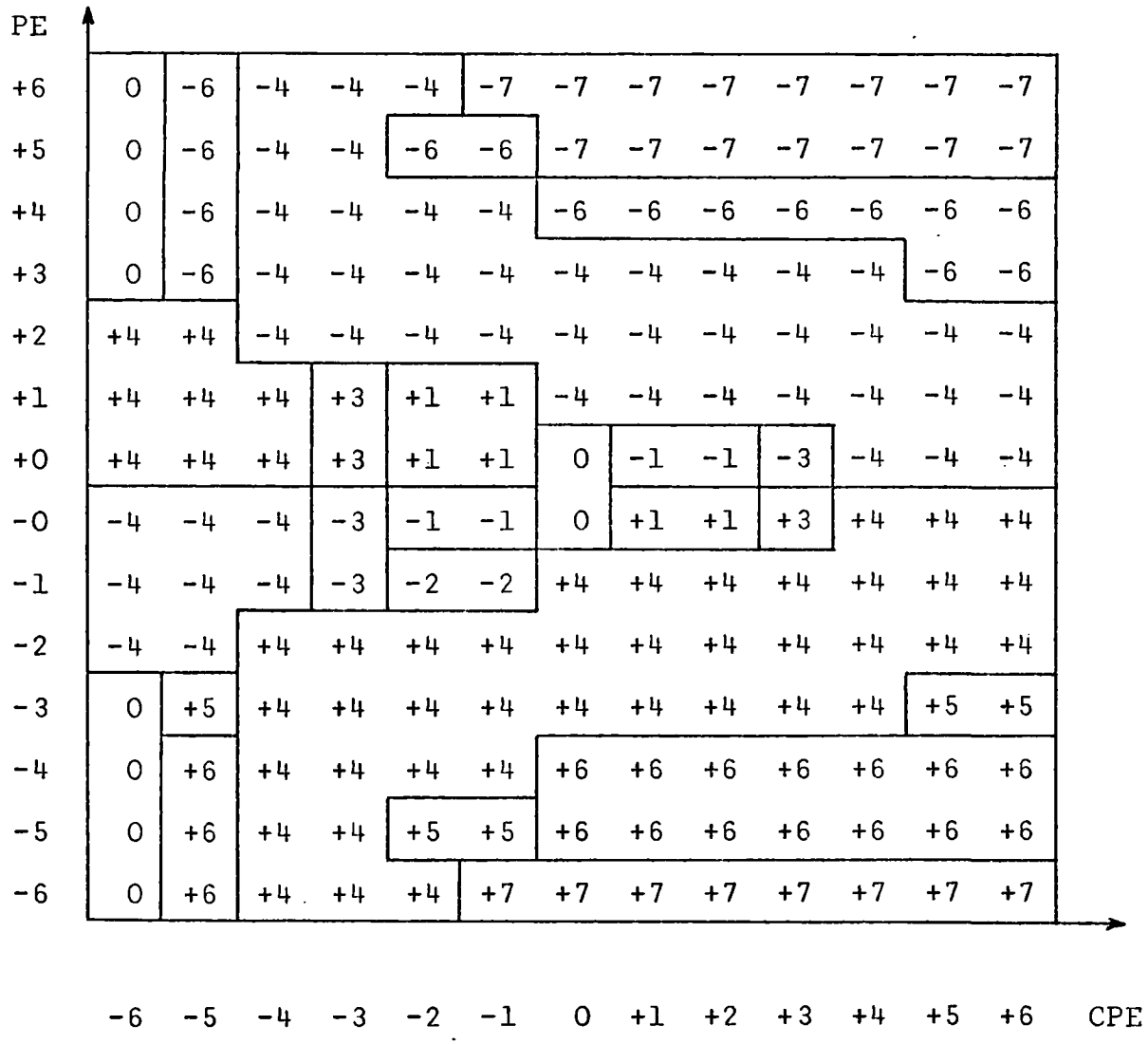


Fig.5.3 Fuzzy Logic Heat Control Policy

SE													
+6	0	-2	0	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
+5	0	-2	0	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
+4	0	-2	0	-2	-2	-2	-2	-1	-1	-1	-1	-1	-1
+3	0	-2	0	-2	-2	-2	-2	-1	-1	-1	-1	-1	-1
+2	0	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
+1	0	0	0	0	0	0	0	0	0	-1	-1	-1	-1
+0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1
-0	0	0	0	0	0	0	0	0	0	0	+1	+1	+1
-1	0	0	0	0	0	0	0	0	0	+1	+1	+1	+1
-2	0	0	0	0	0	0	0	+1	+1	+1	+1	+1	+1
-3	0	+1	0	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
-4	0	+1	0	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
-5	0	+1	0	+1	+2	+2	+2	+2	+2	+2	+2	+2	+2
-6	0	+1	0	+1	+2	+2	+2	+2	+2	+2	+2	+2	+2
	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6 CSE

Fig.5.4 Fuzzy Logic Throttle Control Policy

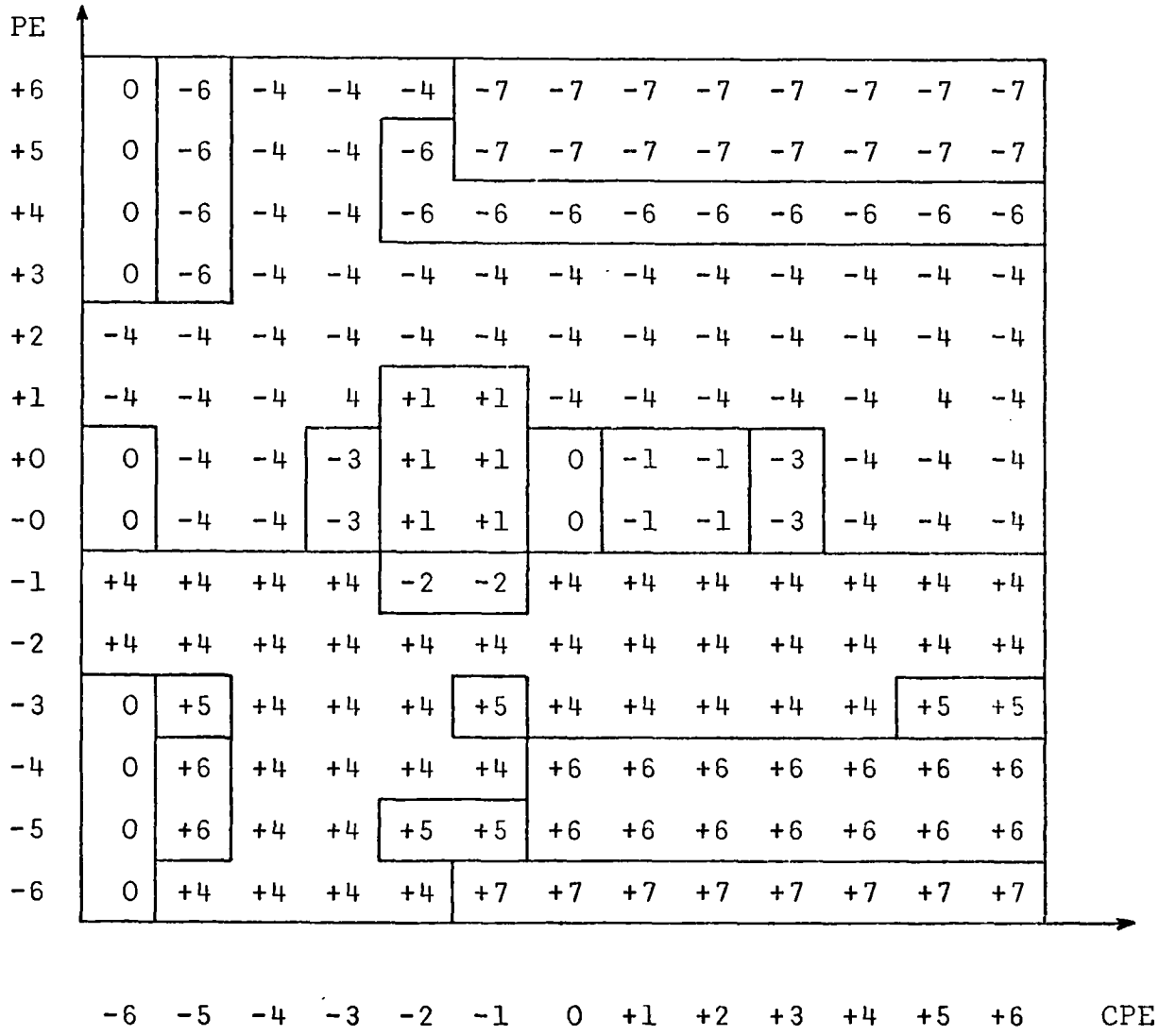


Fig.5.5 Bayes Heat Control Policy

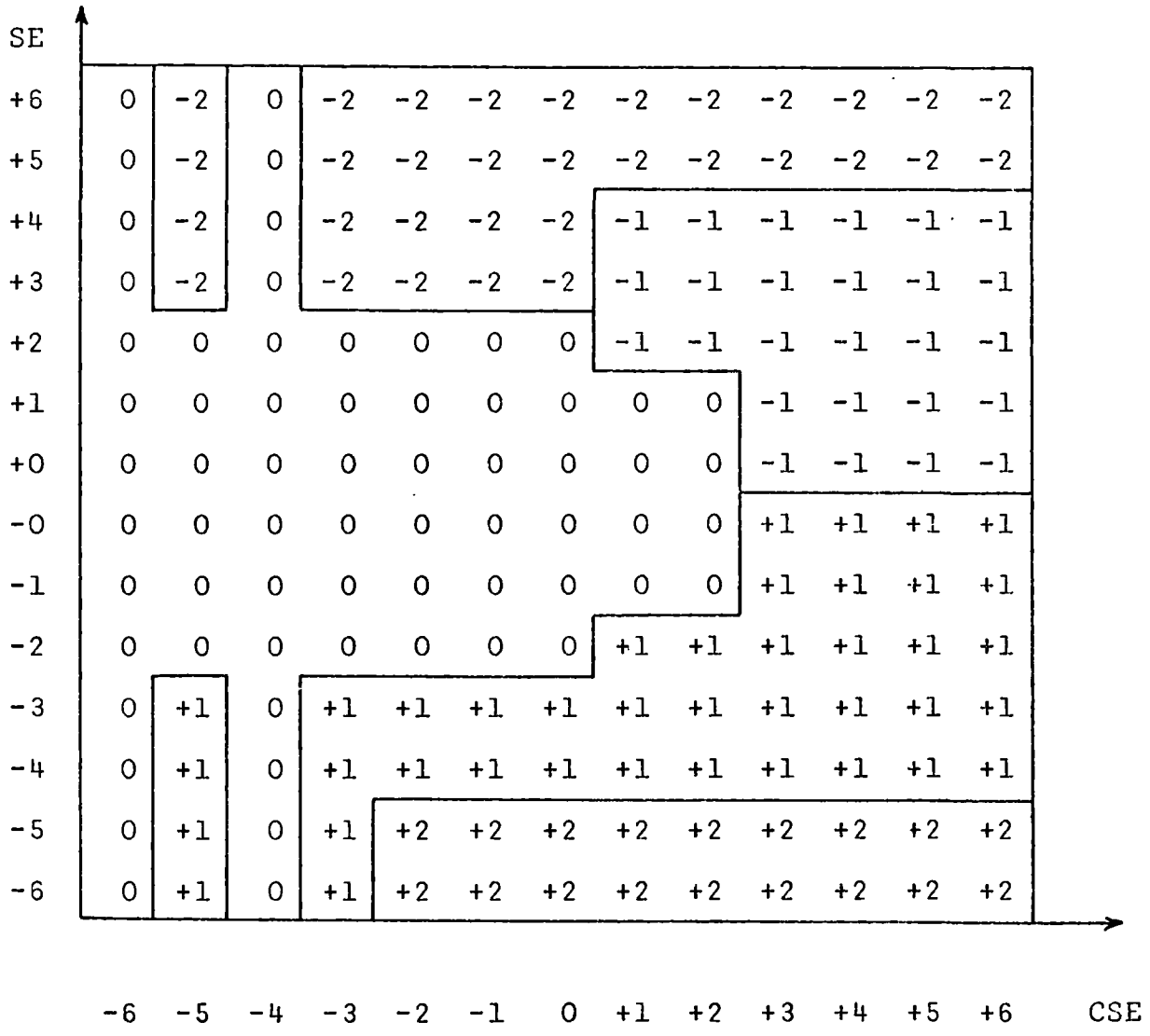


Fig.5.6 Bayes Throttle Control Policy

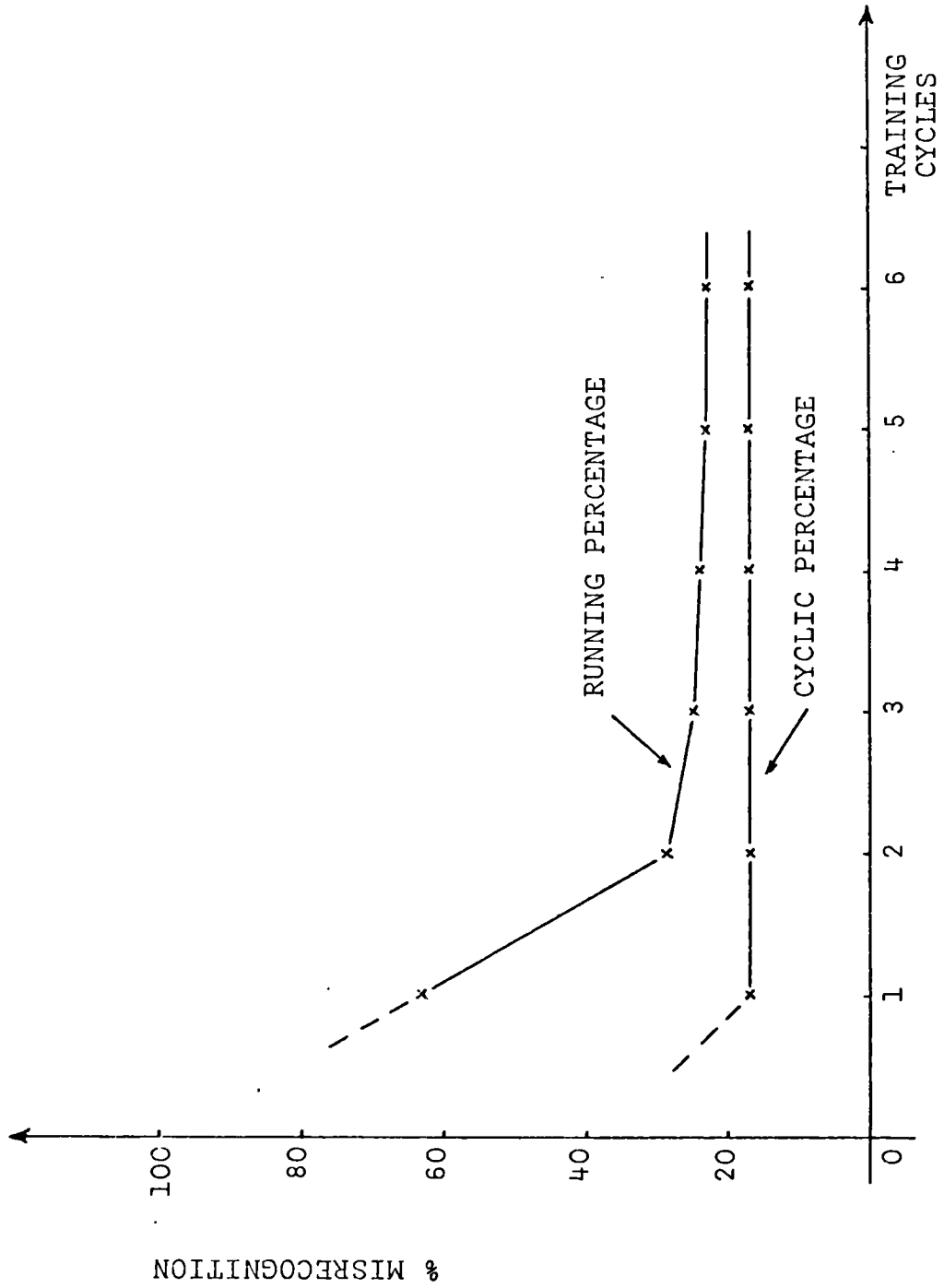


Fig. 5.7 Bayes Heat Learning Curve

5.5.2 TRAINING OF THRESHOLD LOGIC CONTROLLER

In the implementation of the threshold logic controller the 1-out-of-C method (Section 4.2.8) of separating the multiple classes was employed and the training procedure used was that of the fixed-increment error correction rule (Section 4.2.3.1). The experimental procedure was identical to the one described for the Bayes controller.

5.5.2.1 Single-Spot Code

Two methods of coding the input pattern were tested, one using the single-spot code, and the second, the multiple-spot code. Note that both are linearly independent codes (Smith, 1964, 1966). With single-spot coding the learned control policy for heat is shown in Fig.5.8 which was arrived at after 50 training cycles. The learning of the throttle control policy was perfect and therefore Fig.5.4 is not duplicated unnecessarily.

5.5.2.2 Multi-Spot Code

Two different multi-spot codes were tested as shown in Fig.5.11. The control policies learned using these codes were very similar to those obtained with the use of the single-spot code and therefore these results are not presented. A more interesting aspect is brought out instead in Figs.5.9 and 5.10 where the rates of adaptation using the different coding methods are compared for the heat and throttle control respectively.

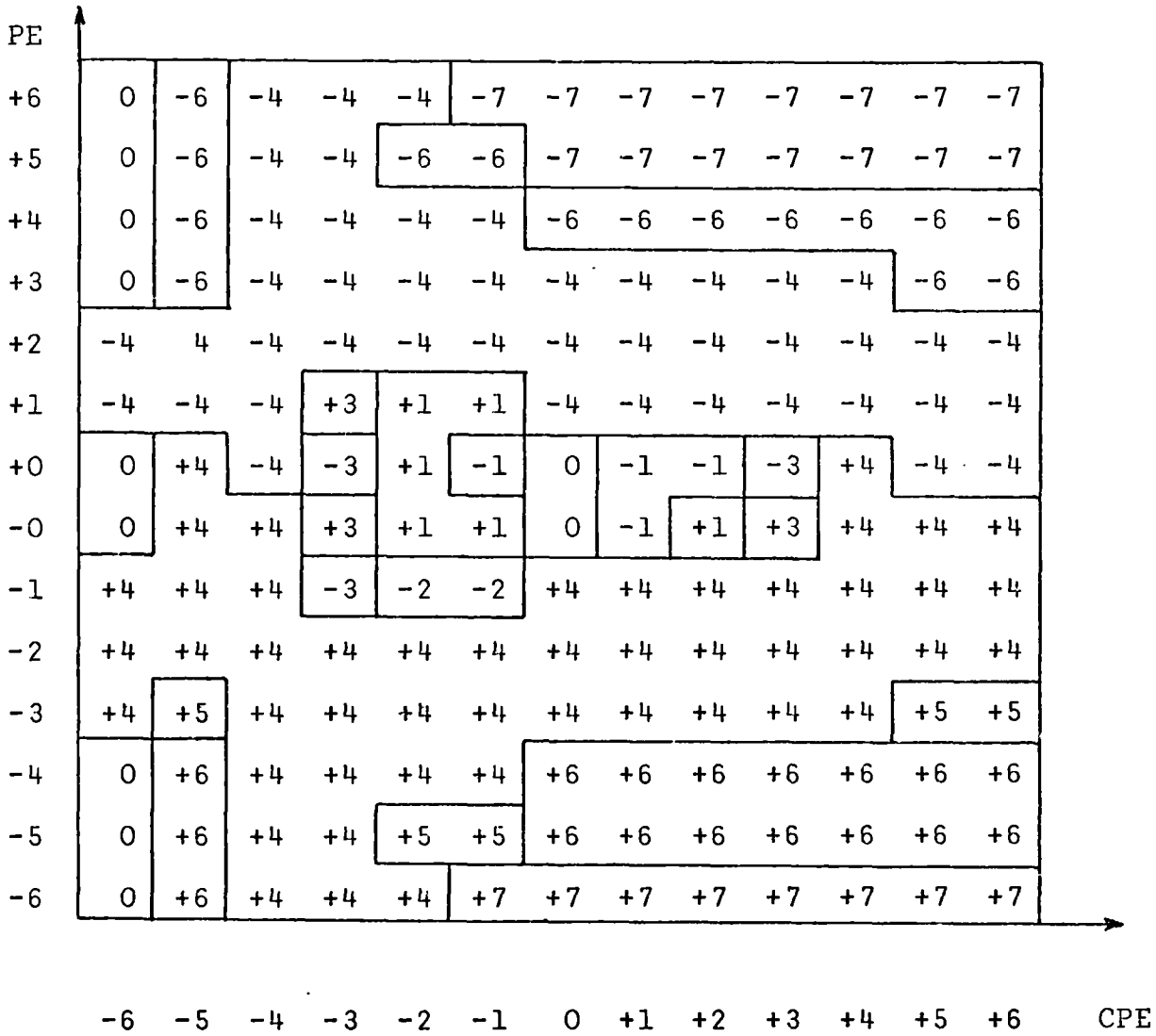


Fig.5.8 Threshold Logic Heat Control Policy

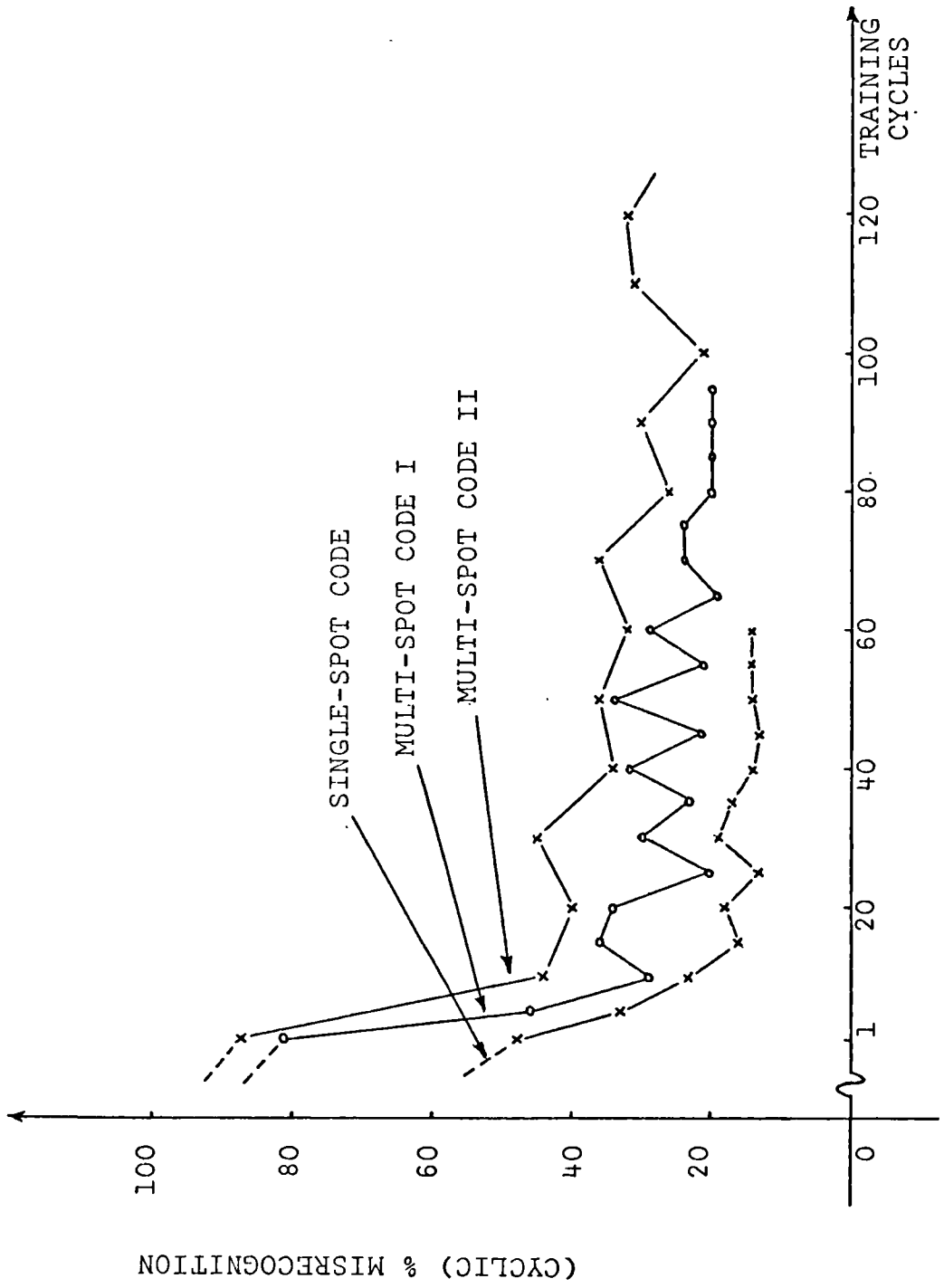


Fig. 5.9 Threshold Logic Heat Learning Curve

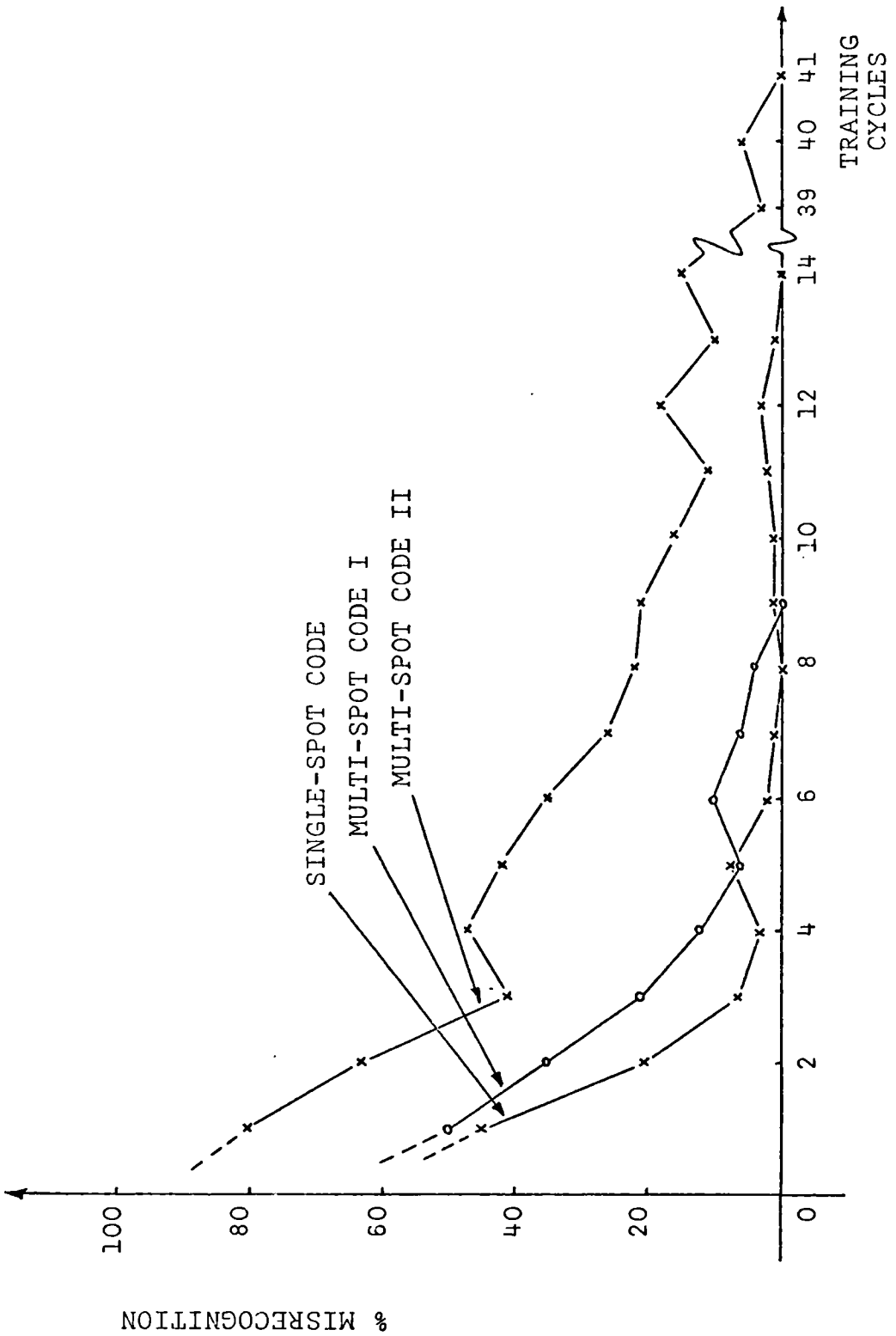


Fig. 5.10 Threshold Logic Throttle Learning Curve

VARIABLE	Single-Spot	Multi-Spot I	Multi-Spot II
x_1	0,0,0,1	+1,+1,+1,+1	1,1,1,1
x_2	0,0,1,0	+1,+1,+1,-1	1,1,1,0
x_3	0,1,0,0	+1,+1,-1,-1	1,1,0,0
x_4	1,0,0,0	+1,-1,-1,-1	1,0,0,0

Fig. 5.11 Linearly Independent Codes

5.5.3 DISCUSSION

The first observation in these experiments is that all the adaptive controllers considered achieved to learn perfectly (there are only two misrecognitions in the case of the Bayes controller - see point made below) the throttle control policy of the fuzzy logic controller, but not in the case of heater control. By comparing the clustering properties of these two policies in Figs. 5.3 and 5.4 perhaps this fact is not too surprising. Since the threshold logic controller achieved 100% classification as dictated by the fuzzy logic controller, it is concluded that the throttle control classes are linearly separable. It must be pointed out here that in designing the fuzzy logic controller no consideration was given to properties of clustering or linear separability, and any such outcome was quite accidental. The two trivial misrecognitions in the Bayes throttle controller, states (SE = -0, CSE = +3) and (SE = +0, CSE = +3), illustrate the fact pointed out in Section 4.4.2 that even with linearly

separable sets of patterns it is still possible for the Bayes classifier to make errors.

In view of the arguments given above, it must also be concluded that the heat control classes of the fuzzy logic controller are not linearly separable. As a result both the threshold logic and Bayes controllers have similar differences in adapted policy with respect to the fuzzy logic controller. Although in terms of per cent misrecognition the figure is only about 15% in both adaptive controllers, the different policy adapted to can have a detrimental effect which is far removed from one's interpretation of a ratio like 15%. It is best to illustrate this point by considering an example. It can be seen from Fig.5.3 that one of the policies of the fuzzy logic controller is: "When the pressure is just below the set-point and its velocity is high and directed towards the set-point, then decrease the heat input a little in order to avoid overshooting." For example, the state (PE = -1, CPE = -6). However, because of the clustering properties of the adaptive controllers, this policy has been corrupted in the latter, and instead the heat is further increased which is very likely to produce a big overshoot of the pressure variable. Close examination of Figs.5.3, 5.5 and 5.8 reveal other similar mal-adaptations in the adaptive controllers.

Another observation to be made in these experiments is concerning the influence of different codes on the rate of adaptation. One of the implications of Smith's (1964) work, is that when the sets of patterns to be classified are linearly separable, then any linearly independent code used to represent the inputs will produce correct classification of these sets. This is indeed the case as the results of Fig.5.10 show. However, two questions still remain open: first, what happens when the pattern sets are not linearly separable, and second, which of the several linearly independent codes is more efficient regarding rate of adaptation. The first question is perhaps the harder of the two to answer, and although the results in Fig.5.9 suggest the superiority of the single-spot code, no such conclusion can be drawn here on the grounds of insufficiency of evidence.

As regards the second question raised, both Figs.5.9 and 5.10 again suggest the superiority of the single-spot code. Without drawing such a conclusion either, an intuitive explanation is offered on the other hand, which points towards this possibility. Consider the variable shown in Fig.5.11. Recalling the weight adjustment procedure of Eqn.(4.3) for the ATLE, it will be observed that the occurrence of the variable in the four different regions x_1, x_2, x_3 and x_4, will affect only one different weight (of evidence) at a time when the single-spot code is used, whereas the same weight can be influenced by all four values of the variable when one of the multi-spot codes is used. This interaction between the weights across different input patterns can conceivably slow down the rate of adaptation when the multi-spot code is used.

Remaining on the question of rate of adaptation, the situation is quite different in the case of the Bayes controller. The Bayes controller achieved its final form in one training cycle (Fig.5.7) compared with those shown for the threshold logic controller in Fig.5.9. Of course, this is expected considering the mathematical formulation of the Bayes controller; given that a pattern X belongs to a class A, the probability of X belonging to A cannot change if on every occurrence of X it is assigned to A in the process of training. What can change, as indicated by the running percentage of misclassification in Fig.5.7, is the probability of choosing a class (or taking an action). From Eqn.(4.13) it is obvious that the probability of a class that is occurring more often can outweigh the probability of the true class of an input pattern by an amount which is enough to cause misclassification. It is regarding these considerations in giving the rule of thumb that the training set of patterns should be statistically representative of the environment.

Finally, it was interesting to notice the variation in the range of weights acquired by the threshold logic controllers, using the three different codes. The eventual ranges established were:

Single-Spot Code	-6 → 7
Multi-Spot Code I	-35 → 39
Multi-Spot Code II	-40 → 55

Evidently, the single-spot code is more economical in terms of the number of binary bits required to represent the weights. In any case, combined with the results for throttle control, the differences in the weights also prove the non-uniqueness of the solution weight vector in the convergence theorem of the ATLE.

5.6 CONCLUSIONS

A number of experiments have been described in this chapter concerning the supervised training of adaptive controllers for the steam engine. The results, which have been discussed in some depth in each experiment, bring out certain problems that can arise when dealing with a realistic system. Some of the main conclusions drawn from these problems that were encountered can be grouped together as follows.

(a) Assuming the trainer is perfect, or near-perfect in his task of labelling the training pattern sets, then it is known that an appropriate ATLE network will learn to classify correctly, provided the pattern sets to be classified are linearly separable. Almost the same arguement is true for the Bayes classifier. On the other hand, it is very seldom that prior knowledge about linear separability exists with patterns generated in a real system. Furthermore, it is reasonable to assume that in most real systems the classification task is of such nature that the original stimuli generated do not constitute a linearly separable set of patterns. Thus, with the present techniques of adaptive control, the controllers produced in most, realistic situations are (very often ridiculously) sub-optimal. This suggests at least three alternatives for future investigation; first, evaluation of present coding schemes and possibly finding better ones - it is felt that little can be gained here since any new coding scheme is very likely to introduce much redundancy, like the linearly independent code; second, investigation of more complex ATLE networks - although it is generally agreed that more complex networks are necessary in order to solve complex problems, such a task is by no means

easy (Rosenblatt, 1962); and third, investigation of other techniques than threshold logic and decision theory - in fact, a number of other techniques have been studied in the literature, for example, stochastic controllers (Gaines and Andreae, 1966; Fu, 1970), and recently, the fuzzy logic approach (Chang and Zadeh, 1972).

(b) The human operator represents an important class of controllers in real systems. Taken on his own, and given a certain amount of freedom, he can be considered to be an apt controller albeit irregular in some aspects. On the other hand, the experiments described in this chapter have shown that he is unsuitable as a teacher in adaptive control systems. In view of this, limited studies have been made and procedures which allow for an imperfect teacher have been suggested (Shanmugam and Breipohl, 1971). Such procedures, however, are in general difficult to implement, and it is felt that the real need is in providing a more effective channel for direct linquistic communication between man and machine. This channel would constitute the most upper level in the hierarchic structure of the general learning system.

(c) Unsupervised learning is feasible, and when it is implemented some of the problems arising with the human teacher are bypassed. However, two main problems exist with this scheme; first, the vast amount of storage, and second, the very long time that are both necessary for the learning process. In fact, one of the reasons for not considering unsupervised learning experiments in this study is because the amount of storage that was available in the PDP8/S was not sufficient for the implementation of an unsupervised learning procedure. A second reason is that it was believed research effort is better spent on the problem of man-machine interaction, and this is the theme of the next chapter.

CHAPTER 6

FUZZY LOGIC AND CONTROL

6.1 INTRODUCTION

One of the important objectives of this study was the training of adaptive or learning controllers by the human operator, and yet very discouraging results were obtained in the investigation and experimentation of this possibility. One of the fundamental reasons accounting for this failure was the incapability of the human operator in translating his linguistically expressed control strategies into quantitative control actions. In designing a controller, quantitative languages supporting arithmetic are the natural ones to the control engineer. On the other hand, most control engineers would accept intuitively that the mathematical computations they perform in translating their concept of a control strategy into an automatic controller are far removed from their own approach to the manual performance of the same task. Therefore, there feels to be a pressing need in providing a direct path between a loose linguistic statement of a control strategy and its implementation as a quantitative control algorithm.

The influence and importance of both direct verbal communication and linguistic representation/expression have also been studied and pointed out in various other contexts. In the field of psychology, Luria (1961) has investigated the important role of speech in the organization of human behaviour, and in specific, in a child's development of his mental activities. He illustrates clearly the influence of verbal instructions given to a child in formulating his own perception of the physical objects surrounding him and in his learning of simple perceptual-motor skills. In the field of artificial intelligence, Winograd (1972) has given an impressive demonstration of

the possibility of direct linquistic control of a robot with a mechanical arm performing manipulatory tasks with toy blocks, although his main concern has been the understanding of natural language by machines as one of the fundamental aspects of intelligence. Becker(1969, 1970) has attempted to formalize the structure of a semantic memory, his work originating again out of the need for intelligent machines to be able to converse with humans. In their studies of adaptive controllers and adaptive trainers, Gaines (1971, 1972) and Pask (1971) have brought out the importance of the interaction arising out of verbal instruction between trainer and trainee. Similarly, in the research carried out by Waterman (1968, 1970) in the machine learning of heuristics, linguistics has played an important role.

A distinguishing difference between the present study and the ones cited above is the desire of maintaining the level of imprecision or 'fuzziness' in translating a linguistic instruction into a quantitative control law. Thus, in the implementation of the control strategy "when PE is large and negative, and CPE is large and towards the set-point, then increase the heat by a medium amount," it is not wished to give precise numerical definitions of the descriptives 'large' and 'medium'. Gaines (1971, 1972), for instance, has translated similar strategies into precise stimulus-response training pattern vectors in 'priming' ATLE controllers with an initial control policy which overcomes the sub-environment phenomenon. Waterman (1968, 1970), on the other hand, preserves some imprecision by dealing with a range of values, instead of a single numerical value in defining a descriptive like 'high'. The means of maintaining fuzziness in this study is through the use of Zadeh's (1965) calculus of fuzzy logic.

6.2 FUZZY SETS AND FUZZY LOGIC

The notion of a theory of fuzzy sets was first introduced by Zadeh (1965) out of his contention that the real physical world is an aggregate of classes of objects which do not have a precisely defined criteria of membership, and that human thinking and reasoning, be it in the domain of pattern recognition, communication of information, decision-making, control or abstraction, is based not on the traditional two-valued or even multivalued logic, but a logic with fuzzy truths and fuzzy rules of inference. Zadeh thus exemplifies his contention by asking the members of "the class of all beautiful women," when the question might well have been "the class of all large PE". As regards human behaviour, he argues that one of the most important facets of human thinking is his ability to summarize information, and that by its nature, a summary is an approximation to what it summarizes (Zadeh, 1973). Thus in performing a task, the human retains only 'task-relevant' or 'decision-relevant' information with a maximal degree of fuzziness which is just sufficient for him to be able to repeat that particular task in the future.

The intuitive, informal definition of a fuzzy set is that class of objects in which transition from membership to non-membership is gradual rather than abrupt. This definition suggests a distinction between fuzziness and randomness, which must be emphasized. Essentially, randomness has to do with uncertainty concerning membership or non-membership of an object in a non-fuzzy set. Thus in dealing with this type of uncertainty, usually the concepts and techniques of probability theory are employed. For example, "the probability that a transition will take place between two states S_1 and S_2 given an action A is 0.7". Fuzziness, on the other hand, deals with uncertainties of the type "the grade of membership of state S_1 in the class of large PE is 0.7".

Since its first introduction, a fair amount of theoretical work has been done to extend the concept of fuzziness in several directions. The work has covered such concepts as fuzzy algorithms (Zadeh, 1968; Santos, 1970), fuzzy relations and orderings (Zadeh, 1971a), fuzzy functions (Kandel, 1973), fuzzy systems (Zadeh, 1971b) and quantitative fuzzy semantics (Zadeh, 1971c). Bellman and Zadeh (1970) have investigated decision-making in a fuzzy environment, where a fuzzy decision is viewed as an intersection of the given fuzzy goals and fuzzy constraints. Wee and Fu (1969) have formulated a fuzzy automaton and used this as a model of learning systems in automatic control and pattern recognition applications. Similarly, Chang and Zadeh (1972) have treated the control problem based on fuzzy mappings and functions. And lastly, Zadeh (1970) has investigated in a preliminary way the importance of fuzzy languages in human and machine intelligence.

6.3 FORMAL DEFINITIONS OF FUZZY SETS AND THEIR PROPERTIES

The definitions given below are due to Zadeh (1965, 1973). The order of presentation has been chosen such as to indicate the way in which fuzzy logic can be used in the translation of linguistic control strategies into quantitative algorithms which can be implemented in a computer programme. Definitions are always illustrated by examples to clarify the mathematical operations and computations involved in the procedure.

6.3.1 THE FUZZY SET

Let U be the universe of discourse or the space of points, with a generic element of U denoted by u. Thus, a finite universe of discourse

$$U = \{u_1, u_2, ---, u_n\}$$

$$= \{u_i\} \quad i = 1, 2, ---, n \qquad (6.1)$$

$$= \sum_{i=1}^{n} u_i$$

where Σ stands for union.

Example. The PE universe of discourse from Fig.5.1 is

$$PE = \{-6,-5,---,-0,+0,---,+5,+6\} \tag{6.2}$$

A fuzzy set A in U is characterised by a membership function $\mu_A: U \rightarrow [0,1]$ which associates with each element u of U a number $\mu_A(u)$ in the interval $[0,1]$ which represents the grade of membership of u in A. The fuzzy set A of U in Eqn.(6.1) will be denoted

$$A = \sum_{i=1}^{n} \mu_A(u_i)/u_i \tag{6.3}$$

Example. The fuzzy set 'positive big' (PB) of PE in Eqn.(6.2) is

$$PB = (0.1/+3) + (0.4/+4) + (0.8/+5) + (1.0/+6) \tag{6.4}$$

where + stands for union. Note that at the remaining points, -6, -5, ---, +2, in the universe PE the grade of membership of PB is assumed to be zero.

It therefore follows that fuzzy sets can be viewed as an extension to ordinary sets, since, when $\mu_A(u_i)$ is constrained to only two values 0 and 1, from Eqn.(6.3) A reduces to the definition of an ordinary set. More significant, however, is the ease with which intuitive meanings can be attached to words in normal use. Thus in the example, PE is a fuzzy variable, and PB, which is the label of a fuzzy set, is a value of that fuzzy variable where the fuzzy set represents the meaning of PB.

6.3.2 BASIC OPERATORS

6.3.2.1 UNION

The union of two fuzzy sets A and B of the same universe of discourse U is denoted A+B and is defined by

$$A+B = \sum_{i=1}^{n} (\mu_A(u_i) \vee \mu_B(u_i))/u_i \tag{6.5}$$

where V stands for maximum (abbreviated to max). Intuitively, the union corresponds to the connective OR in the algebra of ordinary sets.

Example. Define the fuzzy set 'positive medium' (PM) of PE as

PM = (0.2/+2)+(0.7/+3)+(1.0/+4)+(0.7/+5)+(0.2/+6) (6.6)

Then the union of PB and PM from Eqns.(6.4) and (6.6) is

PB+PM = (0.2/+2)+(0.7/+3)+(1.0/+4)+(0.8/+5)+(1.0/+6) (6.7)

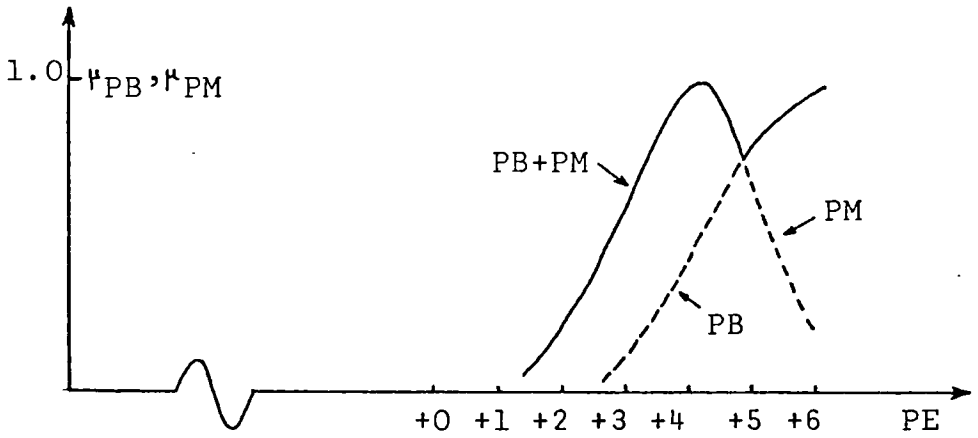


Fig.6.1 Union of Fuzzy Sets.

The extension of the union to more than two sets is obvious. For instance,

$$A+B+C = \sum_{i=1}^{n} \max\{\mu_A(u_i), \mu_B(u_i), \mu_C(u_i)\}/u_i \qquad (6.8)$$

6.3.2.2 INTERSECTION

The intersection of two fuzzy sets A and B of the same universe of discourse U is denoted $A \cap B$ and is defined by

$$A \cap B = \sum_{i=1}^{n} (\mu_A(u_i) \wedge \mu_B(u_i))/u_i \qquad (6.9)$$

where $\wedge$ stands for minimum (abbreviated to min). Intuitively, the intersection corresponds to the connective AND in the algebra of ordinary sets.

Example. The intersection of PB and PM from Eqns.(6.4) and (6.6) is

$$PB \cap PM = (0.1/+3)+(0.4/+4)+(0.7/+5)+(0.2/+6) \qquad (6.10)$$

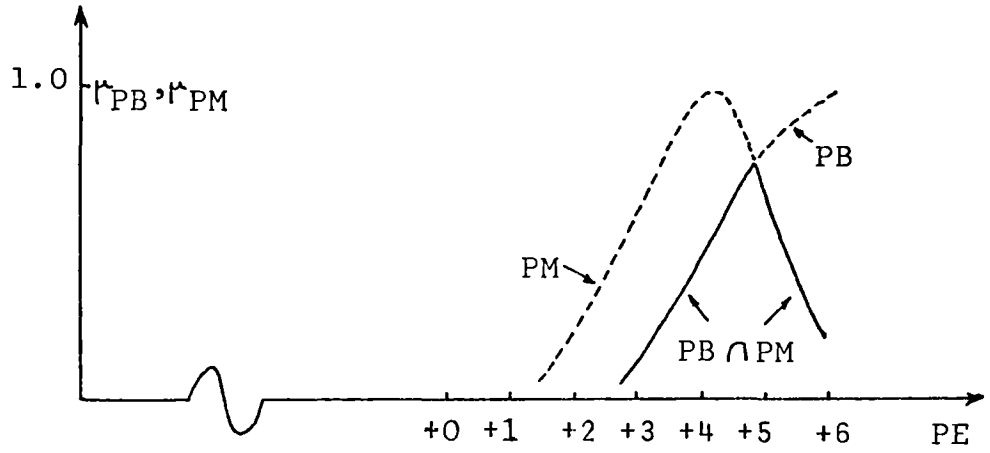


Fig.6.2 Intersection of Fuzzy Sets.

The extension of the intersection to more than two sets is obvious. For instance,

$$A \cap B \cap C = \sum_{i=1}^{n} \min\{\mu_A(u_i), \mu_B(u_i), \mu_C(u_i)\}/u_i \qquad (6.11)$$

6.3.2.3 COMPLEMENT

The complement of a fuzzy set A of the universe of discourse U is denoted $\neg A$ and is defined by

$$\neg A = \sum_{i=1}^{n} (1-\mu_A(u_i))/u_i \qquad (6.12)$$

The operation of complementation corresponds to negation - NOT.

Example. The complement of PM (NOT PM) from Eqn.(6.6) is

$$\neg PM = (1.0/-6)+(1.0/-5)+(1.0/-4)+(1.0/-3)+(1.0/-2)+(1.0/-1)+(1.0/-0)+(1.0/+0)+(1.0/+1)+(0.8/+2)+(0.3/+3)+(0.0/+4)+(0.3/+5)+(0.8/+6) \quad (6.13)$$

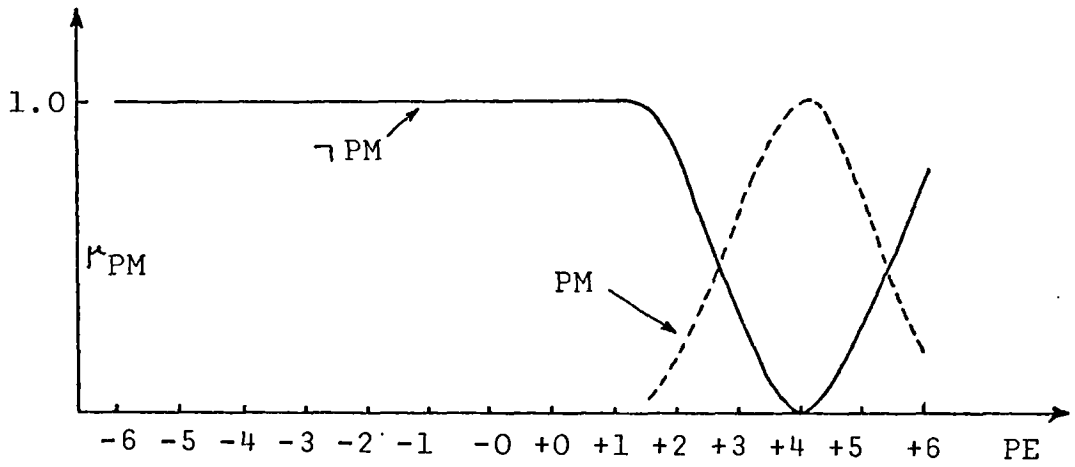


Fig.6.3 Complement of Fuzzy Set

It can be seen, therefore, that the three basic operators AND, OR and NOT permit the representation of expressions of words. Thus, it is possible now to give a value to the meaning of expressions like 'positive big OR positive medium AND NOT positive small'. Zadeh (1973) considers also the introduction of hedges, such as 'very', 'much', 'slightly', etc., into expressions and by viewing these as operators similar to the three above, he gives intuitive definitions for them. However, the use of such linguistic hedges was left out in this study for two reasons; first, in order to keep the level of the final product simple, with the intention of introducing them later if it became necessary to do so (which it didn't - see the results of Chapter 7); second, the effect of operating a linguistic hedge on a fuzzy set becomes significant only when a large universe of discourse is assumed, which is not the case in this study.

6.3.2.4 CARTESIAN PRODUCT

The cartesian product of two fuzzy sets A and B, of possibly different universes of discourse $U = \{u_1, u_2, ---, u_n\}$ and $V = \{v_1, v_2, ---, v_m\}$ respectively, is denoted AxB and is defined by

$$AxB = \sum_{i=1}^{n} \sum_{j=1}^{m} \min\{\mu_A(u_i), \mu_B(v_j)\}/(u_i, v_j) \qquad (6.14$$

The cartesian product of two sets AxB can be conveniently represented by a matrix of n rows and m columns.

Example. The 'heat change'(HC) universe of discourse from Section 5.2.1 is

$$HC = \{-7, -6, ---, -1, 0, +1, ---, +6, +7\} \qquad (6.15$$

Define the fuzzy set 'negative big' (NB) of HC as

$$NB = (1.0/-7)+(0.8/-6)+(0.4/-5)+(0.1/-4) \qquad (6.16$$

Then the cartesian product of PE=PB and HC=NB from Eqns. (6.4) and (6.16) is

$$\begin{aligned} PE_{PB} x HC_{NB} = {} & 0.1/(+3,-7)+0.1/(+3,-6)+0.1/(+3,-5)+0.1/(+3,-4) \\ & +0.4/(+4,-7)+0.4/(+4,-6)+0.4/(+4,-5)+0.1/(+4,-4) \\ & +0.8/(+5,-7)+0.8/(+5,-6)+0.4/(+5,-5)+0.1/(+5,-4) \\ & +1.0/(+6,-7)+0.8/(+6,-6)+0.4/(+6,-5)+0.1/(+6,-4) \end{aligned} \qquad (6.17$$

which can be represented by the matrix

$$PE_{PB} \begin{array}{c} \\ (+3) \\ (+4) \\ (+5) \\ (+6) \end{array} \begin{array}{c} HC_{NB} \\ \begin{array}{cccc} (-7) & (-6) & (-5) & (-4) \end{array} \\ \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.4 & 0.4 & 0.4 & 0.1 \\ 0.8 & 0.8 & 0.4 & 0.1 \\ 1.0 & 0.8 & 0.4 & 0.1 \end{bmatrix} \end{array} \qquad (6.18)$$

The extension of the cartesian product to more than two sets is obvious. For instance, if C is a fuzzy subset of the universe $W=\{w_1, w_2, ---, w_\ell\}$, then

$$AxBxC = \sum_{i=j}^{n} \sum_{j=1}^{m} \sum_{k=1}^{\ell} \min\{\mu_A(u_i), \mu_B(v_j), \mu_C(w_k)\}/(u_i, v_j, w_k) \qquad (6.19)$$

6.3.3 RELATIONS AND FUZZY CONDITIONAL STATEMENTS

In essence, the cartesian product of two fuzzy sets can be looked upon as a relation between two fuzzy variables. Thus, the cartesian product AxB establishes a relation from U to V. Similarly, in the example given above, Eqn.(6.17) or the matrix of (6.18) define the relation of applying a big negative heat action when the pressure error is positive big. A relation will be denoted R.

An alternative view is to regard the cartesian product as a representation of conditional statements of the form

IF A THEN B = AxB (6.20

Thus Eqn.(6.17) represents the conditional statement that "IF the pressure error is positive big THEN apply a negative big heat action". More generally, conditional statements can have the form

IF A THEN (IF B THEN(IF C THEN(IF---))) = AxBxCx--- (6.21

or, further

IF A_1 THEN (IF B_1 THEN(IF C_1 THEN(IF---)))
ELSE IF A_2 THEN (IF B_2 THEN(IF C_2 THEN(IF---)))
ELSE ----------

$= (A_1xB_1xC_1x---) + (A_2xB_2xC_2x---) + ---$ (6.22

where + stands for union.

6.3.4 COMPOSITIONAL RULE OF INFERENCE

To summarise the development thus far, numerical values can be given to the meaning of words, these words can be combined to form expressions, and these expressions can be used to make (conditional) statements. One last question remains: How can one infer relevant information from these statements?

To be specific, let R be a fuzzy relation from U to V as given by Eqn.(6.14). Then, given a fuzzy subset a of U, the fuzzy subset b of V which is induced by a is given by the composition of R and a, that is,

$$b = aoR \tag{6.23}$$

Composition is interpreted as the max-min product of a and R, and therefore from Eqns.(6.14) and (6.23)

$$b = \sum_{j=1}^{m} [\max_i [\min\{\mu_a(u_i), \min\{\mu_A(u_i), \mu_B(v_j)\}\}]]/(v_j)$$

$$= \sum_{j=1}^{m} [\max_i [\min\{\mu_a(u_i), \mu_A(u_i), \mu_B(v_j)\}]]/(v_j) \tag{6.24}$$

where $\max_i$ denotes maximum over all i.

Example. Consider the following:

i) If PE is PB then make HC equal NB.

ii) If PE is PM, then what is HC?

The conditional statement in i) is given by Eqn.(6.17)

The definition of PE=PM is given by Eqn.(6.6)

Therefore, from Eqn.(6.24)

$$HC_{PE=PM} = (0.7/-7)+(0.7/-6)+(0.4/-5)+(0.1/-4) \tag{6.25}$$

It is worthwhile to look a little closer at this result.

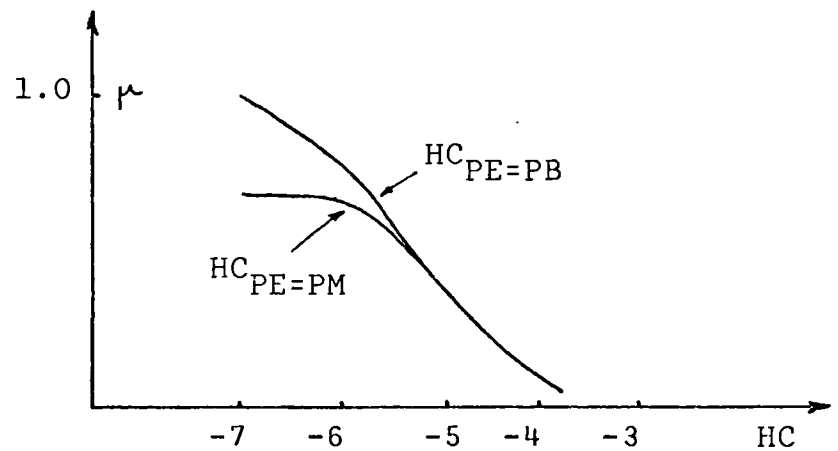


Fig.6.4 Inference

By looking at Fig.6.4, Eqn.(6.22) effectively says "when the pressure error is positive medium then do not apply a negative big heat action with the same certainty as when the pressure error is positive big". In other words, a negative medium heat action is given preference. Furthermore, the more different PE is from PB, the less close HC is to NB, and at the same time the less sharply defined is HC. Thus, with a PE far removed from PB, virtually nothing can be inferred about HC from the statement in i). Of course this is not unreasonable, and it becomes necessary in this case to consider other statements which are relevant to that situation.

The extension of the compositional rule of inference to cover conditional statements of the forms given in Eqns.(6.21) and (6.22) is straightforward. The following two examples give simple illustrations of each case.

Example. Consider the conditional statement

IF A THEN (IF B THEN C) (6.26)

where A, B and C are fuzzy subsets of the universes U, V and W respectively. The relation R=AxBxC is given by Eqn.(6.19). Then, given a fuzzy subject a of U and a fuzzy subset b of V, what is the fuzzy subset (c) of W induced by a and b. The answer to this question can be found in two parts. Analogous to Eqn.(6.24), the fuzzy subset a induces the two-dimensional fuzzy set BC of VxW as

$$BC = \sum_{k=1}^{\ell} \sum_{j=1}^{m} [\max_{i}[\min\{\mu_a(u_i),\mu_A(u_i),\mu_B(v_j),\mu_C(w_k)\}]]/(v_j,w_k) \quad (6.27)$$

And now, the fuzzy subset b induces the fuzzy set c as

$$c = \sum_{k=1}^{\ell} [\max_{j}[\min\{\mu_b(v_j),\max_{i}[\min\{\mu_a(u_i),\mu_A(u_i),\mu_B(v_j),\mu_C(w_k)\}]\}]]/(w_k$$

$$= \sum_{k=1}^{\ell} [\max_{j}[\max_{i}[\min\{\mu_a(u_i),\mu_b(v_j),\mu_A(u_i),\mu_B(v_j),\mu_C(w_k)\}]]]/(w_k) \quad (6.28$$

Example. Consider the conditional statement

$$\begin{aligned} &\text{IF } A_1 \text{ THEN } B_1 \\ \text{ELSE} \quad &\text{IF } A_2 \text{ THEN } B_2 \end{aligned} \tag{6.29}$$

where A_1 and A_2 are values of the fuzzy variable A, and similarly, B_1 and B_2 are values of the fuzzy variable B. The question is this: If the value of the fuzzy variable A is a, then what is the value (b) of the fuzzy variable B induced by a. From Eqn.(6.24), the first conditional statement in (6.29) yields

$$b_1 = \sum_{j=1}^{m} [\max_i [\min\{\mu_a(u_i), \mu_{A_1}(u_i), \mu_{B_1}(v_j)\}]]/(v_j) \tag{6.30}$$

Similarly, the second conditional statement in (6.29) yields

$$b_2 = \sum_{j=1}^{m} [\max_i [\min\{\mu_a(u_i), \mu_{A_2}(u_i), \mu_{B_2}(v_j)\}]]/(v_j) \tag{6.31}$$

Now, the connective ELSE is equivalent to the connective OR, and therefore corresponds to the max operation. That is,

$$\begin{aligned} b &= \max(b_1, b_2) \\ &= \sum_{j=1}^{m} [\max\{\max_i [\min\{\mu_a(u_i), \mu_{A_1}(u_i), \mu_{B_1}(v_j)\}], \\ &\qquad \max_i [\min\{\mu_a(u_i), \mu_{A_2}(u_i), \mu_{B_2}(v_j)\}]\}]/(v_j) \\ &= \sum_{j=1}^{m} [\max_i [\min\{\mu_a(u_i), \mu_{A_1}(u_i), \mu_{B_1}(v_j)\}, \\ &\qquad \min\{\mu_a(u_i), \mu_{A_2}(u_i), \mu_{B_2}(v_j)\}]]/(v_j) \end{aligned} \tag{6.32}$$

With the above examples, all the tools necessary for a linguistic synthesis of a controller are complete.

6.4 FUZZY LOGIC CONTROL ALGORITHMS AND THEIR EXECUTION

In its strict sense, an algorithm is an ordered set of instructions which upon execution yield a solution to a specified problem. Here a fuzzy logic control algorithm is a collection of control policies of the form given in Eqn.(6.22) which upon execution in response to an input to it, corresponding to a state of the controlled plant, yields an action to be applied to that plant. Both the algorithm and its execution exhibit certain features which are noteworthy.

A very significant feature of the algorithm of Eqn.(6.22) is that it is an unordered or unstructured set of instructions. This fact can be verified by the execution of the algorithm given in Eqns.(6.24), (6.28) and (6.32). The important implication of this is, of course, that modification of the algorithm can be achieved with great convenience; policies can be deleted or added anywhere in the set without affecting the result of execution. In contrast, the 'production rules' of Waterman (1968, 1970), which are very similar to the conditional statements referred to in this study, form an ordered set and a great deal of rearrangement must be carried out if any rule is deleted, modified or added to the set. A rough contrast can also be made with the vast amount of work which incorporate a 'decision tree', as a model of the environment, from which to extract a decision (for instance, Rae, 1968)). The inconvenience of deleting or adding a node in such trees is obvious.

An interesting feature of the execution of the algorithm is that more than one control policy can contribute to the computation of a control action. This, of course, arises out of the fuzzy nature of the control policies, and as it happens, it has its assets. For one, it is not necessary to specify a long and exhaustive set of rules or instructions since a situation can still be covered by rules which apply to similar situations. Secondly, and more important, the output of the algorithm can indicate if there

are any contradictory, or even any weak or bad rules in the set. For example, consider the illustrations in Fig. 6.5.

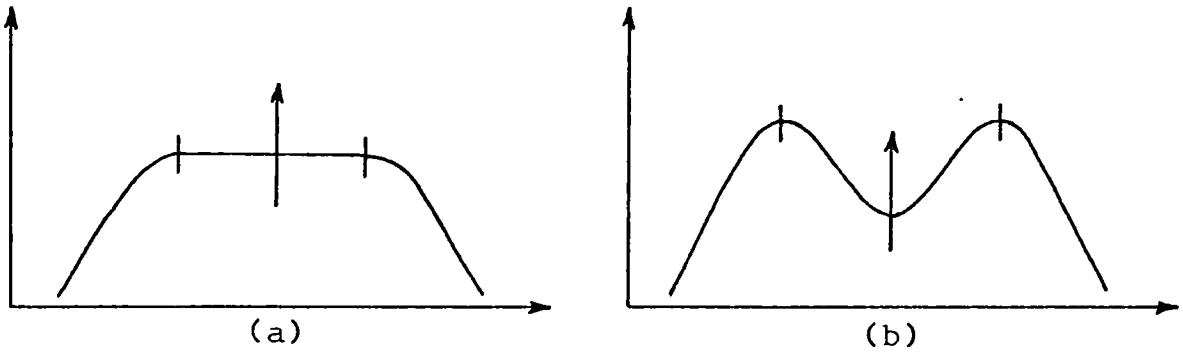


Fig.6.5 Fuzzy Outputs of Algorithm

An output of the form in (a) indicates the absence of a strong or good set of rules. An output of the form in (b), on the other hand, indicates the presence of at least two contradictory rules. Therefore it is possible by monitoring the execution of the algorithm to locate and modify control policies of such bad nature. This point is discussed further in the next chapter.

Finally, it is clear that the output of the fuzzy logic control algorithm is a fuzzy set which corresponds to a fuzzy action. Of course, only deterministic actions can be applied to a real plant, and therefore some procedure must be employed to reduce a fuzzy action to a deterministic one. The specification of such a procedure depends on the particular application and it is usually at the designer's discretion. Zadeh (1968) has considered a few possibilities and the particular procedure used in this study is given in the next chapter.

6.5 CONCLUSIONS

Using the notion of fuzzy sets and fuzzy logic a procedure has been developed for translating linguistic control policies into a quantitative, executable and coherent control algorithm. The procedure is fairly simple, and more important, has an intuitive appeal to the human way of thinking.

Central to the scheme is the fuzzy conditional statement of the form IF A THEN B, which is so simple in what it says, that its significance can be easily overlooked when it is isolated to only one study like this one. It is remarkable, however, that it is used as the underlying mechanism in so many different studies. In psychology, one of the two major categories into which learning theories fall is the stimulus-response (S-R) theory (Hilgard and Bower, 1966). In studies of problem solving, the expression 'condition → action' has been used to characterise the problem-solving process occurring in a human subject as she solves a problem (Newell and Simon, 1972). In studies of strategy or heuristics learning, heuristics have been represented as an ordered set of production rules which again have the form 'condition → action' (Waterman, 1968, 1970). Similarly, the concept of the production has proved useful in the analysis of compiler writing (Floyd, 1964). And as a last instance, the same concept is implied by the expression 'state → action' which is used in control studies (Gaines and Andreae, 1966).

The conditional statement IF A THEN B is also studied in propositional calculus as the implication ($\Rightarrow$) connective, where A and B are propositional variables. From this point of view, the compositional rule of inference can be regarded as an approximate extension of the rule of modus ponens. Informally, the rule of modus ponens states that having previously established the propositions "$A \Rightarrow B$" and "A", then "B" may be inferred. Thus, the formulation of the fuzzy logic controller is in fact based on well established concepts and theories.

CHAPTER 7

FUZZY LOGIC CONTROL OF THE STEAM ENGINE

7.1 INTRODUCTION

The fuzzy logic control algorithm developed in the last chapter was implemented on the PDP8/S and experiments were carried out to test the applicability of the fuzzy logic approach to the control of the steam engine. To the best knowledge of the author, this is the first case study where fuzzy logic is used in the control of a real plant.

The experimental features were the same as those described in Section 5.2 for the adaptive controllers. That is, there were four inputs to the controller, PE, SE, CPE and CSE, quantized as specified in Fig.5.1, and two outputs, heat change (HC) and throttle change (TC), which were allowed to take $0 \rightarrow \pm 7$ and $0 \rightarrow \pm 2$ steps respectively. The sampling period was 10 secs.

The implemented control algorithm is a collection of control policies that the human operator (the author, in this case) would have used had he been controlling the steam engine. These control policies were established first by imagining the entire state space (PExCPExSExCSE) to be divided into a number of areas, and second, writing down a control policy for each of these areas. Obviously, the first set of rules obtained in this manner does not necessarily produce the best quality of control possible and therefore further modifications of the algorithm must be considered. A method of modifying the control algorithm, or 'tuning' the controller, and features to facilitate it are described below.

Finally, it must be made clear that the fuzzy logic controller does not exhibit any learning in the sense implied in the case of adaptive or learning controllers. The fuzzy logic controller merely implements the control policy of the human operator, and nothing else. The fabrication of a learning fuzzy logic controller (Wee and Fu, 1969) is a possibility, of course, and this will be discussed later.

7.2 PRELIMINARY CONSIDERATIONS

7.2.1 SPECIAL FEATURES OF THE CONTROLLER

The fuzzy logic controller developed in Chapter 6 is in its most general form and it becomes necessary to introduce certain special features when one is dealing with a real plant. These features concern the inputs and outputs of the controller.

7.2.1.1 The Inputs

The computation of the compositional rule of inference assumes that the input to the controller has a fuzzy value. In the present study, the inputs were chosen to be non-fuzzy vectors, where only one element of a partial input vector has a membership grade of one, the rest being all zero. Input vectors constructed in this manner are actually identical to those when the single-spot linearly independent code is used.

Example. When PE = +2, then the partial non-fuzzy PE input vector is

-6	-5	-4	-3	-2	-1	-0	+0	+1	+2	+3	+4	+5	+6
[0,	0,	0,	0,	0,	0,	0,	0,	0,	1,	0,	0,	0,	0]

With this constraint on the input, the computation involved in calculating an action is considerably reduced. Thus, if

$$\mu_a(u_i)=1 \text{ for } i=p \quad 1\leqslant p\leqslant n$$
$$=0 \text{ for } i\neq p \qquad (7.1)$$

and

$$\mu_b(v_j)=1 \text{ for } j=q \quad 1\leqslant q\leqslant m$$
$$=0 \text{ for } j\neq q \qquad (7.2)$$

then, Eqn.(6.28) reduces to

$$c = \sum_{k=1}^{\ell} \min\{\mu_A(u_p),\mu_B(v_q),\mu_C(w_k)\}/(w_k) \qquad (7.3)$$

and Eqn.(6.32) reduces to

$$b = \sum_{j=1}^{m} \max\{\min\{\mu_{A_1}(u_p), \mu_{B_1}(v_j)\}, \min\{\mu_{A_2}(u_p), \mu_{B_2}(v_j)\}\}/(v_j) \qquad (7.4)$$

It would have been quite feasible, of course, to fuzzify (Zadeh, 1973) the input once it was sampled. In certain applications such a procedure may be advantageous if the input variables are inherently very noisy, or if the measuring instruments used are not reliable.

7.2.1.2 The Outputs

It was mentioned in Section 6.4 that some procedure must be employed to reduce the fuzzy output of the controller to a deterministic one in order to apply it to a real plant. The procedure used in this study is the following:

(a) If there is a distinct peak in the fuzzy output spread, then apply the action corresponding to that peak.

(b) Otherwise, if there are two equal peaks or a plateau in the output spread, then take an action which is midway between the two peaks or at the centre of the plateau.

Thus, in Fig. 6.5 the actions taken are indicated by the arrows pointing upwards.

There are at least two important points about this procedure, which must be mentioned. First, it is clear that in the absence of any output spread, which can happen if none of the rules contribute anything to the output, the controller will take no action. This is a reasonable policy, of course, since an output spread is not produced only when the controller does not recognize the input state of the plant. The second point is that, even when the grade of membership is as low as 0.1 at a peak, the controller will still take the action corresponding to that peak. Applying such an action can produce unsatisfactory control since a membership grade of 0.1 suggests great uncertainty in the choice of that action. However, it can also happen that the chosen action is the right one, so that eventually it is up to the designer to decide, depending on the particular application.

7.2.2 THE VALUES OF THE FUZZY VARIABLES

The six fuzzy variables PE, SE, CPE, CSE, HC and TC were assigned values using nine basic fuzzy sets. These values were:

(a) PB - Positive Big
(b) PM - Positive Medium
(c) PS - Positive Small
(d) PO - Positive Zero (in the case of PE and SE)
(e) NO - Negative Zero (in the case of PE and SE)
- Zero (in other cases)
(f) NS - Negative Small
(g) NM - Negative Medium
(h) NB - Negative Big
(i)ANY - Any value.

The subjective fuzzy sets defining these values are given in Tables 7.1 to 7.4.

	-6	-5	-4	-3	-2	-1	-0	+0	+1	+2	+3	+4	+5	+6
PB	0	0	0	0	0	0	0	0	0	0	.1	.4	.8	1.
PM	0	0	0	0	0	0	0	0	0	.2	.7	1.	.7	.2
PS	0	0	0	0	0	0	0	.3	.8	1.	.5	.1	0	0
PO	0	0	0	0	0	0	0	1.	.6	.1	0	0	0	0
NO	0	0	0	0	.1	.6	1.	0	0	0	0	0	0	0
NS	0	0	.1	.5	1.	.8	.3	0	0	0	0	0	0	0
NM	.2	.7	1.	.7	.2	0	0	0	0	0	0	0	0	0
NB	1.	.8	.4	.1	0	0	0	0	0	0	0	0	0	0
ANY	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.

Table 7.1 Values for PE and SE

	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6
PB	0	0	0	0	0	0	0	0	0	.1	.4	.8	1.
PM	0	0	0	0	0	0	0	0	.2	.7	1.	.7	.2
PS	0	0	0	0	0	0	0	.9	1.	.7	.2	0	0
NO	0	0	0	0	0	.5	1.	.5	0	0	0	0	0
NS	0	0	.2	.7	1.	.9	0	0	0	0	0	0	0
NM	.2	.7	1.	.7	.2	0	0	0	0	0	0	0	0
NB	1.	.8	.4	.1	0	0	0	0	0	0	0	0	0
ANY	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.

Table 7.1 Values for PE and SE

	-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6	+7
PB	0	0	0	0	0	0	0	0	0	0	0	.1	.4	.8	1.
PM	0	0	0	0	0	0	0	0	0	.2	.7	1.	.7	.2	0
PS	0	0	0	0	0	0	0	.4	1.	.8	.4	.1	0	0	0
NO	0	0	0	0	0	0	.2	1.	.2	0	0	0	0	0	0
NS	0	0	0	.1	.4	.8	1.	.4	0	0	0	0	0	0	0
NM	0	.2	.7	1.	.7	.2	0	0	0	0	0	0	0	0	0
NB	1.	.8	.4	.1	0	0	0	0	0	0	0	0	0	0	0

Table 7.3 Values for HC

	-2	-1	0	+1	+2
PB	0	0	0	.5	1.
PS	0	0	.5	1.	.5
NO	0	.5	1.	.5	0
NS	.5	1.	.5	0	0
NB	1.	.5	0	0	0

Table 7.4 Values for TC

7.2.3 SPECIAL FEATURES OF THE SOFTWARE

7.2.3.1 The Monitor

Several points have been mentioned which make it desirable to have a fairly easy way of modifying or 'tuning' the fuzzy control algorithm. To facilitate this, a monitor was incorporated in the software which, for every action chosen by the algorithm, stored away (a) the fuzzy spread of the action, and (b) the rules contributing to the decision of this action. This information in conjunction with the state-trajectory of the engine, which were both printed out after every run, formed the basis for making any modifications to the control algorithm. The important role of the monitor, without which the tuning process would not have been as systematic and as efficient, will be highlighted in Section 7.4.

7.2.3.2 Separation of Programme and Data Space

To facilitate further the tuning process of the controller, certain features were introduced in the structure of the software. First, it is clear from Eqns.(7.3) and (7.4) that it is only necessary to store away the nine basic values of the six fuzzy variables, and not the relations that are implied by every rule that make up the algorithm. The significant reduction in data space here is obvious. Second, the rules comprising the algorithm were written as calls to the same subroutine, which worked on the data space of the basic definitions of the variables. Therefore, with this scheme the programme and data spaces were separated so that changes could be made independently to either space. Further, by having the algorithm as a chain of subroutine calls, and recalling that this chain of rules is unordered, deleting, modifying or adding a rule amounts to changing only a few lines of coding in the programme.

The subroutine that processes the rules has four parameters passed to it, which are PE, CPE, SE and CSE. Although in one of the controllers implemented not all four inputs are necessary (non-interactive control), the subroutine was made as general as possible so that it could be used unaltered for any configuration (interactive control).

7.3 THE FUZZY LOGIC CONTROLLER: NON-INTERACTIVE CONTROL

The first controller implemented was based on the principle of non-interactive control. Thus, two separate controllers were implemented, one for the heater and one for the throttle. The inputs to the heater controller were PE and CPE (with SE and CSE equal to ANY) and the inputs to the throttle controller were SE and CSE (with PE and CPE equal to ANY). The two algorithms are given below, where the rule numbers attached will be referred to in the discussions and in the tuning of these controllers.

7.3.1 HEATER ALGORITHM

```
Rule H1     If          PE = NB
               then if  CPE = not (NB or NM)
               then if   SE = ANY
               then if  CSE = ANY
            Then        HC = PB
      Else
Rule H2     If          PE = NB or NM
               then if  CPE = NS
               then if   SE = ANY
               then if  CSE = ANY
            Then        HC = PM
      Else
Rule H3     If          PE = NS
               then if  CPE = PS or NO
               then if   SE = ANY
               then if  CSE = ANY
            Then        HC = PS
      Else
Rule H4     If          PE = NO
               then if  CPE = PB or PM
               then if   SE = ANY
               then if  CSE = ANY
            Then        HC = PM
      Else
Rule H5     If          PE = NO
               then if  CPE = NB or NM
               then if   SE = ANY
               then if  CSE = ANY
            Then        HC = NM
      Else
Rule H6     If          PE = PO or NO
               then if  CPE = PS or NS or NO
               then if   SE = ANY
               then if  CSE = ANY
            Then        HC = NO
      Else
```

Rule H7 If PE = PO
then if CPE = NB or NM
then if SE = ANY
then if CSE = ANY
Then HC = PM

Else

Rule H8 If PE = PO
then if CPE = PB or PM
then if SE = ANY
then if CSE = ANY
Then HC = NM

Else

Rule H9 If PE = PS
then if CPE = PS or NO
then if SE = ANY
then if CSE = ANY
Then HC = NS

Else

Rule H10 If PE = PB or PM
then if CPE = NS
then if SE = ANY
then if CSE = ANY
Then HC = NM

Else

Rule H11 If PE = PB
then if CPE = not (NB or NM)
then if SE = ANY
then if CSE = ANY
Then HC = NB

7.3.2 THROTTLE ALGORITHM

Rule T1 If PE = ANY
then if CPE = ANY
then if SE = NB
then if CSE = not (NB or NM)
Then TC = PB

Else

Rule T2 If PE = ANY
then if CPE = ANY
then if SE = NM
then if CSE = PB or PM or PS
Then TC = PS

Else

Rule T3 If PE = ANY
then if CPE = ANY
then if SE = NS
then if CSE = PB or PM
Then TC = PS

Else

Rule T4 If PE = ANY
then if CPE = ANY
then if SE = NO
then if CSE = PB
Then TC = PS

Else

Rule T5 If PE = ANY
then if CPE = ANY
then if SE = NO
then if CSE = NB
Then TC = NS

Else

Rule T6 If PE = ANY
then if CPE = ANY
then if SE = PO or NO
then if CSE = PS or NS or NO
Then TC = NO

Else

Rule T7 If PE = ANY
then if CPE = ANY
then if SE = PO
then if CSE = NB
Then TC = PS
Else
Rule T8 If PE = ANY
then if CPE = ANY
then if SE = PO
then if CSE = PB
Then TC = NS
Else
Rule T9 If PE = ANY
then if CPE = ANY
then if SE = PS
then if CSE = PB or PM
Then TC = NS
Else
Rule T10 If PE = ANY
then if CPE = ANY
then if SE = PM
then if CSE = PB or PM or PS
Then TC = NS
Else
Rule T11 If PE = ANY
then if CPE = ANY
then if SE = PB
then if CSE = not (NB or NM)
Then TC = NB

7.3.3 EXPERIMENTAL RESULTS

The quality of control obtained for the pressure variable is shown by curve P1 in Fig.7.1. The quality of control obtained for the speed variable is shown by curve S1 in Fig.7.2. Both responses shown are typical of a series of runs performed. The printout of the monitor for this run (A) is given in Appendix D.1.

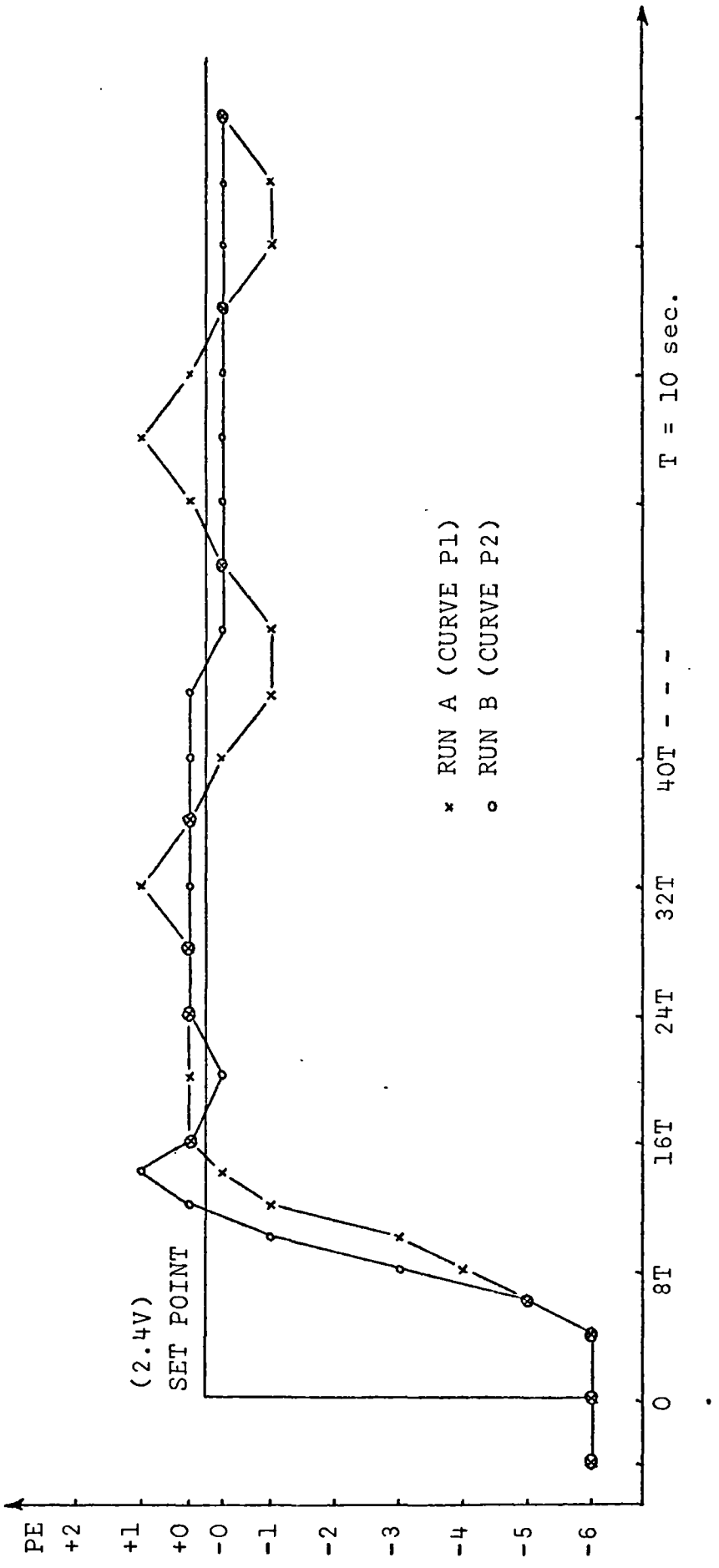


Fig. 7.1 Pressure Response

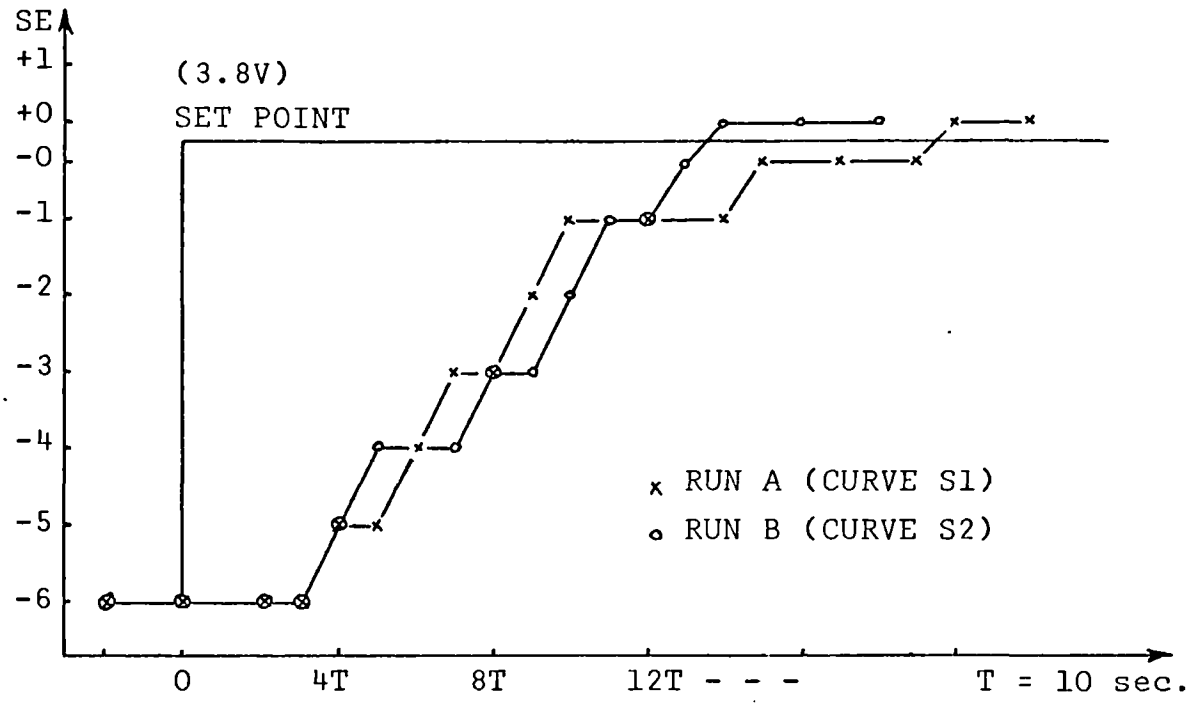


Fig. 7.2 Speed Response

7.3.4 DISCUSSION

Consider the heat-pressure loop. By looking at curve P1 in Fig.7.1, two main observations can be made about the pressure variable. First, the climb towards the set-point is very satisfacotry, and second, there seems to be poor control around the set-point once the pressure gets there. Therefore the rules that come into play around the set-point need to be modified, and if necessary, new ones added.

After analysing the information printed out by the monitor (Appendix D.1), three rules were found which accounted for the poor control around the set-point. The rule that was central to the problem was H6. In words, rule H6 says that "if the pressure is just below or just above the set-point, and if its velocity is zero or very small in either direction, then take no heat action". There are two possibilities in this rule which can give rise to oscillations around the set-point. First, when the pressure is just below the set-point and its velocity is small but

away from the set-point, then it will be allowed to continue on this course without taking any action to stop or reverse it. The same is true when the pressure is just above the set-point. Therefore modification of this rule was necessary, which will be considered in the next section.

The analysis further showed that once the pressure left the vicinity of the set-point, the two rules H3 and H9 did not take strong enough actions to return the pressure to the set-point as quickly as possible. It can be seen from curve P1 that this is specially true when the pressure is below the set-point. Therefore rules H3 and H9 required to be modified also. Apart from this, it was remarkable that the pressure variable behaved in the way that it did, specially when considering the small number of rules that were put down, and moreover, at a first attempt. Different initial conditions of the pressure were also considered and identical observations were made.

Turning attention to the throttle-speed loop now, it can be seen from curve S1 in Fig.7.2 that good control has been obtained. Although it is not indicated on curve S1, there were occasions when the speed tended to get trapped either just above or just below the set-point. The rules that caused this situation were T5 and T7 (Appendix D.1). For instance, when the speed was just below the set-point and going up fast, then rule T5 would put a negative step on the throttle, thus never allowing the speed to get to the set-point unless it was approaching at a slow rate. Therefore rules T5 and T7 required revision. Otherwise, the same general remarks can be made about the speed variable as in the case of the pressure.

7.4 TUNING OF FUZZY LOGIC CONTROLLER: NON-INTERACTIVE CONTROL

The weak rules discovered in the analysis of the last section were rectified and a modified controller was implemented as described below.

7.4.1 MODIFIED HEATER ALGORITHM

Apart from modifying rules H3, H6 and H9, four more rules were added to the algorithm to give even tighter control around the set-point. The modified and the new rules are given below, with the changed lines indicated to the right.

Rule H3a	If	PE = NS	
	then if	CPE = PS or NO	
	then if	SE = ANY	
	then if	CSE = ANY	
	Then	HC = PM	(changed)
Rule H6a	If	PE = PO or NO	
	then if	CPE = NO	(changed)
	then if	SE = ANY	
	then if	CSE = ANY	
	Then	HC = NO	
Rule H9a	If	PE = PS	
	then in	CPE = PS or NO	
	then if	SE = ANY	
	then if	CSE = ANY	
	Then	HC = NM	(changed)
Rule H12	If	PE = NO	
	then if	CPE = PS	
	then if	SE = ANY	
	then if	CSE = ANY	
	Then	HC = PS	
Rule H13	If	PE = NO	
	then if	CPE = NS	
	then if	SE = ANY	
	then if	CSE = ANY	
	Then	HC = NS	

Rule H14 If PE = PO
then if CPE = NS
then if SE = ANY
then if CSE = ANY
Then HC = PS

Rule H15 If PE = PO
then if CPE = PS
then if SE = ANY
then if CSE = ANY
Then HC = NS

7.4.2 MODIFIED THROTTLE ALGORITHM

The only modification made to the throttle algorithm was to delete rules T5 and T7 to avoid the situation described in Section 7.3.4. The decision to delete rather than modify them, was made after verifying in the analysis that they did not make any contribution otherwise.

7.4.3 EXPERIMENTAL RESULTS

The quality of control obtained with the modified version of the controller is shown by curve P2 in Fig.7.1 for the pressure variable, and by curve S2 in Fig.7.2 for the speed variable. Again, the responses shown in both Fig.7.1 and 7.2 are typical of the results obtained. The monitor printout for this run (B) is given in Appendix D.2.

7.4.4 DISCUSSION

It can be seen from curve P2 in Fig.7.1 that control of the pressure around the set-point has improved considerably with the modified heater algorithm. The increased sensitivity of the heater controller can be seen by comparing the monitor outputs before and after tuning, given in Appendix D. Similarly, deleting the throttle rules T5 and T7 allowed settling of the speed variable at the set-point, and the response of both variables can be considered as very satisfactory now. Thus, no further modification of the two control algorithms

was necessary. It is remarkable in fact that such good quality of control is achieved after only two attempts. This only goes to suggest that there can be a great value and potential in the technique of using fuzzy logic whenever a humanistic problem is being tackled.

The importance of the monitor in providing feedback to the human operator is clearly evinced in these experiments. Without this information the human operator would probably have great difficulty in modifying his control policies. An off-line execution of the modified control algorithms and the monitor produced the summary shown in Figs. 7.3 to 7.6. The control policies of these algorithms were given in Figs.5.3 and 5.4 in Chapter 5. The entries in Figs.7.3 and 7.5 are the rules which produce the highest peaks in the output spreads, with corresponding membership grades shown in Figs.7.4 and 7.6. It is not suggested, though, that data obtained in this manner can be used to modify the algorithms, since no information is apparent about the quality of control implied in these data. The 'tuning' process must still be carried out on-line. The data illustrate, on the other hand, the two points mentioned in Section 7.2.1.2 concerning the output decision procedure.

As a final remark, it is useful to note that it is the rules that are modified in the tuning process and not the nine basic, but subjective definitions of the values assigned to the variables involved in the rules. Given a simple, say, 14-point universe of discourse, any subjective definition of 'positive big' is hardly likely to be too contentious. Stated otherwise, a set of rules is rather insensitive to the definition of an individual fuzzy subset. Similarly, when sets with a large number of members are considered, it is again expedient to manipulate the rules, possibly using hedges (Zadeh, 1973), since manipulating the basic definitions interacts with the policy implied in more than one rule. Obviously, this can perpetuate into a vicious circle of modifying a definition, followed by examining all the rules affected by this change.

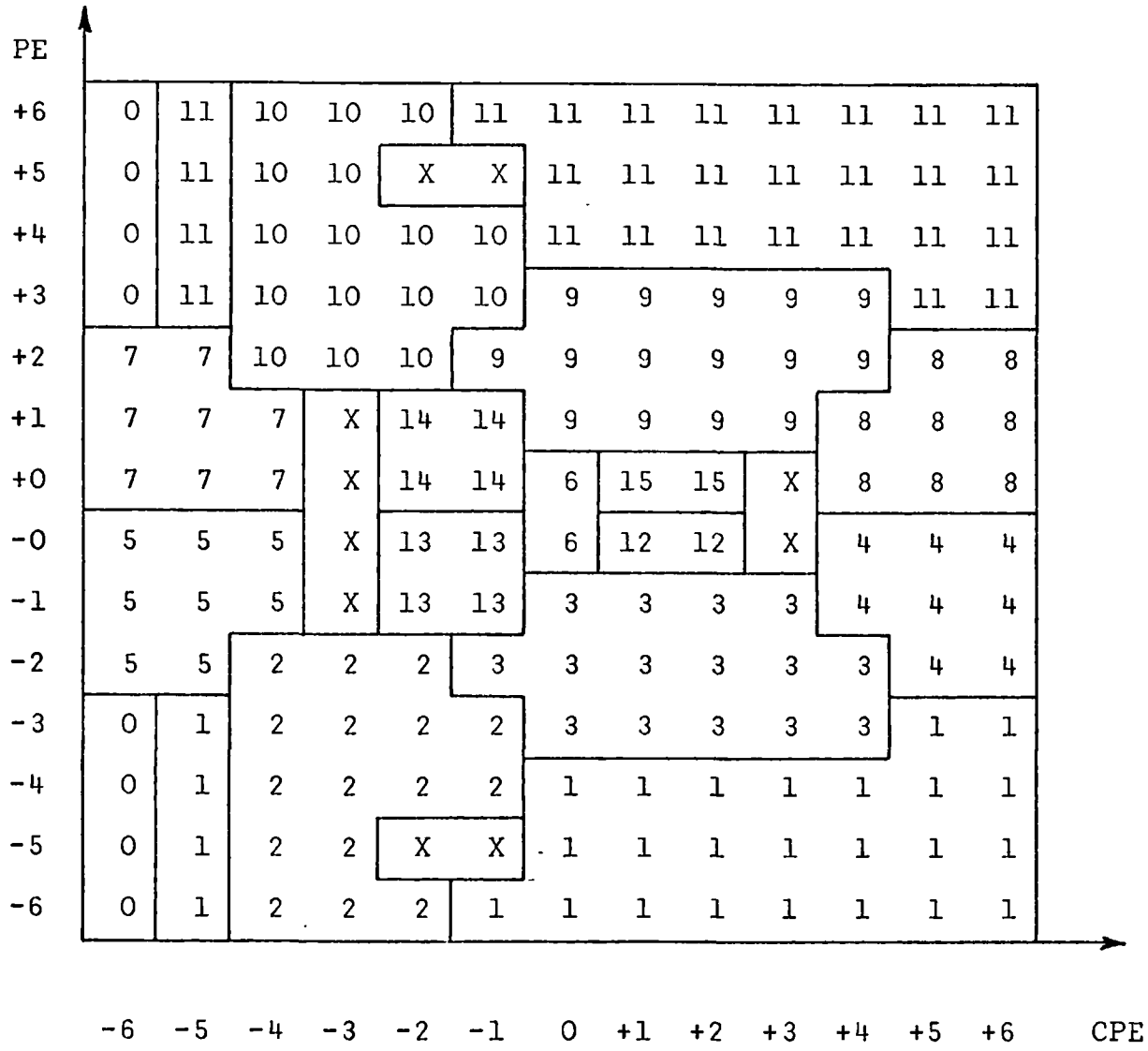


Fig.7.3 Heat Rules Determining Actual Output
(X implies more than one rule)

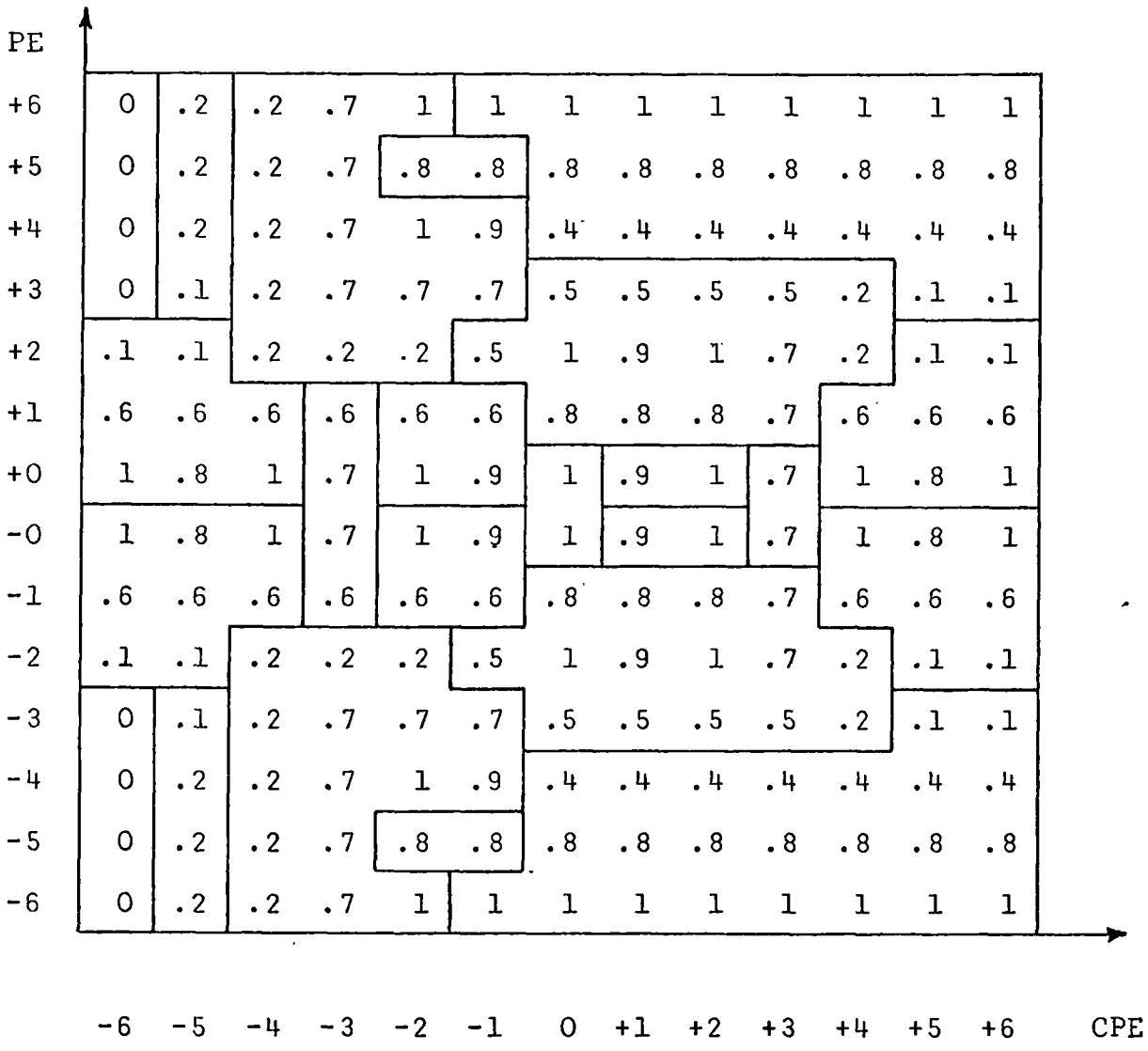


Fig.7.4 Membership Grade of Heat Output

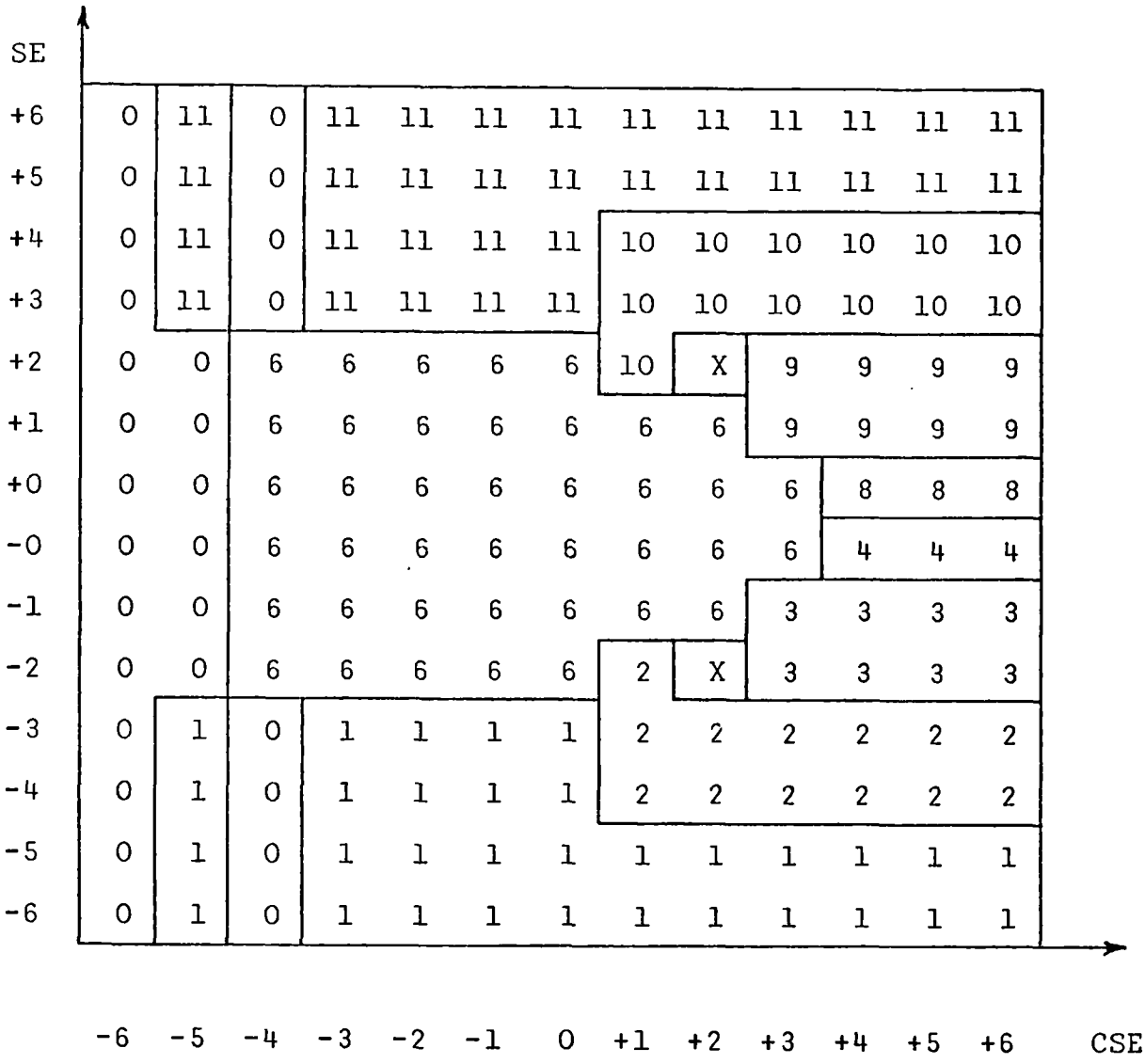


Fig.7.5. Throttle Rules Determining Actual Output
(X implies more than one rule)

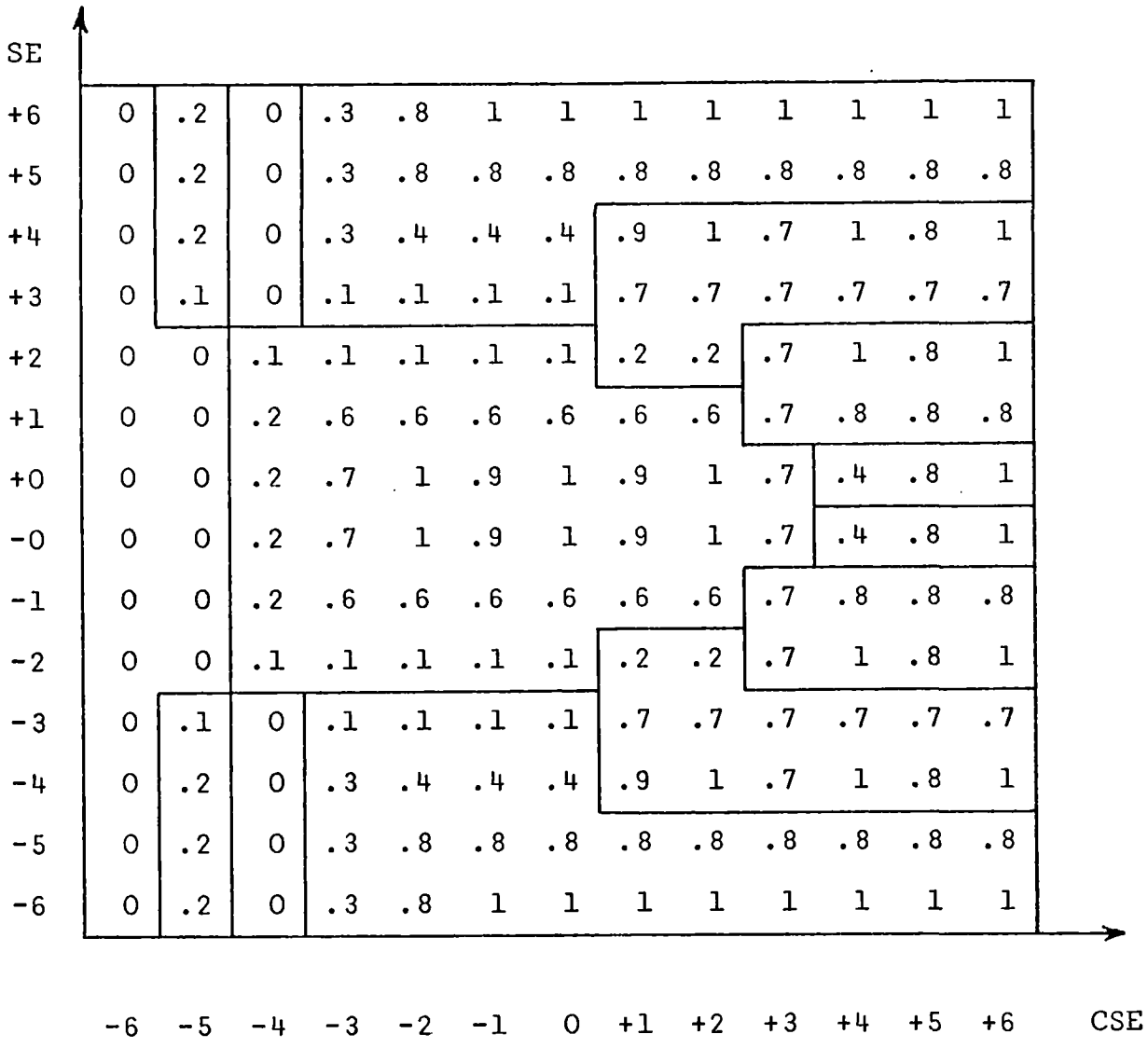


Fig.7.6 Membership Grade of Throttle Output.

7.5 INTERACTIVE FUZZY LOGIC CONTROL

This section is primarily intended for further demonstration of the effectiveness of fuzzy logic in translating linguistic instructions into an executable control algorithm. The two control algorithms described in the last section were non-interactive, that is, the choice of heat action was influenced by the pressure only, and the choice of throttle action was influenced by the speed only. Here, the mode of operation is changed, so that both pressure and speed influence the choice of both the heat and throttle actions. The new desired control policies are described below.

7.5.1 HEATER ALGORITHM

It is known that when the heat input is increased, then this causes the pressure to rise, which in turn increases the speed. Therefore, it is desirable to consider the speed as well when intending to increase the heat input. This policy is introduced by the two modified and the two new rules given below. It will be observed that the nature of the change is 'precautionary', and not for better control of the speed variable.

Rule H1a	If		PE = NB	
		then if	CPE = not (NB or NM)	
		then if	SE = not (PB or PM)	(changed)
		then if	CSE = ANY	
	Then		HC = PB	
Rule H2a	If		PE = NB or NM	
		then if	CPE = NS	
		then if	SE = not (PB or PM)	(changed)
		then if	CSE = ANY	
	Then		HC = PM	

Rule H16

If PE = NB
then if CPE = not (NB or NM)
then if SE = PB or PM
then if CSE = NB or NM
Then HC = PB

Rule H17

If PE = NB or NM
then if CPE = NS
then if SE = PB or PM
then if CSE = NB or NM or NS
Then HC = PM

7.5.2 THROTTLE ALGORITHM

The desired policy for throttle control is the following. It is known that the faster the steam engine is running, the greater is the consumption of steam from the boiler, and therefore the greater must the heat input be in order to maintain the pressure - conservation of energy. Thus, when the pressure is very low, it may be desirable to allow the pressure to reach a certain level first before opening the throttle to gain speed. With this procedure, one would expect a faster rise in the pressure, than when consumption of steam is allowed during the build-up in pressure. Such a policy is normally adopted in the operation of a real steam engine, for example, a steam train, although more for start-up purposes rather than direct control. This policy was implemented by writing a completely new set of rules for throttle control, which is given below.

Rule T12

If PE = NB
then if CPE = ANY
then if SE = NB or NM
then if CSE = not NB
Then TC = NO
Else

Rule T13 If PE = NB
then if CPE = ANY
then if SE = NB or NM
then if CSE = NB
Then TC = NS
Else
Rule T14 If PE = NB
then if CPE = ANY
then if SE = not (NB or NM)
then if CSE = ANY
Then TC = NS
Else
Rule T15 If PE = NM
then if CPE = not (NB or NM)
then if SE = ANY
then if CSE = ANY
Then TC = NO
Else
Rule T16 If PE = not NB
then if CPE = NB or NM
then if SE = NB or NM
then if CSE = ANY
Then TC = PS
Else
Rule T17 If PE = not NB
then if CPE = ANY
then if SE = NM or NS
then if CSE = NB
Then TC = NO
Else
Rule T18 If PE = not NB
then if CPE = ANY
then if SE = NS
then if CSE = not (NB or NM)
Then TC = PS
Else

Rule T19 If PE = not NB
then if CPE = ANY
then if SE = NO
then if CSE = PB or PM
Then TC = PS

Else

Rule T20 If PE = not NB
then if CPE = ANY
then if SE = PO or NO
then if CSE = PS or NS or NO
Then TC = NO

Else

Rule T21 If PE = not NB
then if CPE = ANY
then if SE = PO
then if CSE = PB or PM
Then TC = NS

Else

Rule T22 If PE = not NB
then if CPE = ANY
then if SE = PS
then if CSE = not (NB or NM)
Then TC = NS

Else

Rule T23 If PE = not NB
then if CPE = ANY
then if SE = PM
then if CSE = PB or PM or PS
Then TC = NS

Else

Rule T24 If PE = not NB
then if CPE = ANY
then if SE = PB
then if CSE = not (NB or NM)
Then TC = NB

7.5.3 EXPERIMENTAL RESULTS AND DISCUSSION

The response obtained with the interactive control policy is compared with that of non-interactive control in Figs.7.7 and 7.8. Curves P2 and S2 are the best responses obtained in Section 7.4. Curves P3 and S3 are typical pressure and speed responses respectively, obtained with interactive control. It is observed that no improvement has been achieved in the quality of control, if minimum-time is the criterion of performance. However, the speed response in Fig.7.8 reflects clearly the intended change in control policy; the speed is not allowed to increase until the pressure reaches a certain level (roughly NM in this case, as it can also be confirmed by the throttle algorithm).

It is observed that the pressure response P3, shown in Fig.7.7 does not exhibit the expected faster rise towards the set-point. There is an explanation for this if the monitor print-out, given in Appendix D.3 for this run (C), is studied closely. It can be seen from the printout that the heat input has reached maximum power (32 steps) at T=4 and has remained at that level until T=9, when it has started decreasing. Comparing this with the printout data for run B in Appendix D.2, it can be seen that the same path has been taken in that run. Therefore, on the one hand, the physical limitations of the system can explain the identical, initial responses obtained in Fig.7.7. On the other hand, one still would expect a lag in the pressure since in run B the speed is higher at all times, thus using up more steam. However, two points must be borne in mind here; first, the rate of consumption of steam at very low speeds is unlikely to effect the pressure in the boiler, where steam is being produced at a much faster rate (at maximum power) than at which it is being consumed, and second, because of the quantized nature of the displayed variables, any small difference that could have existed within a quantized region is obviously not brought out.

In Fig.7.7, the lag in pressure that appears starting at T=10, is because the speed in run C is allowed to increase, by opening the throttle, without compensating for the steam that is used in doing this. It is assumed here that at the sort of

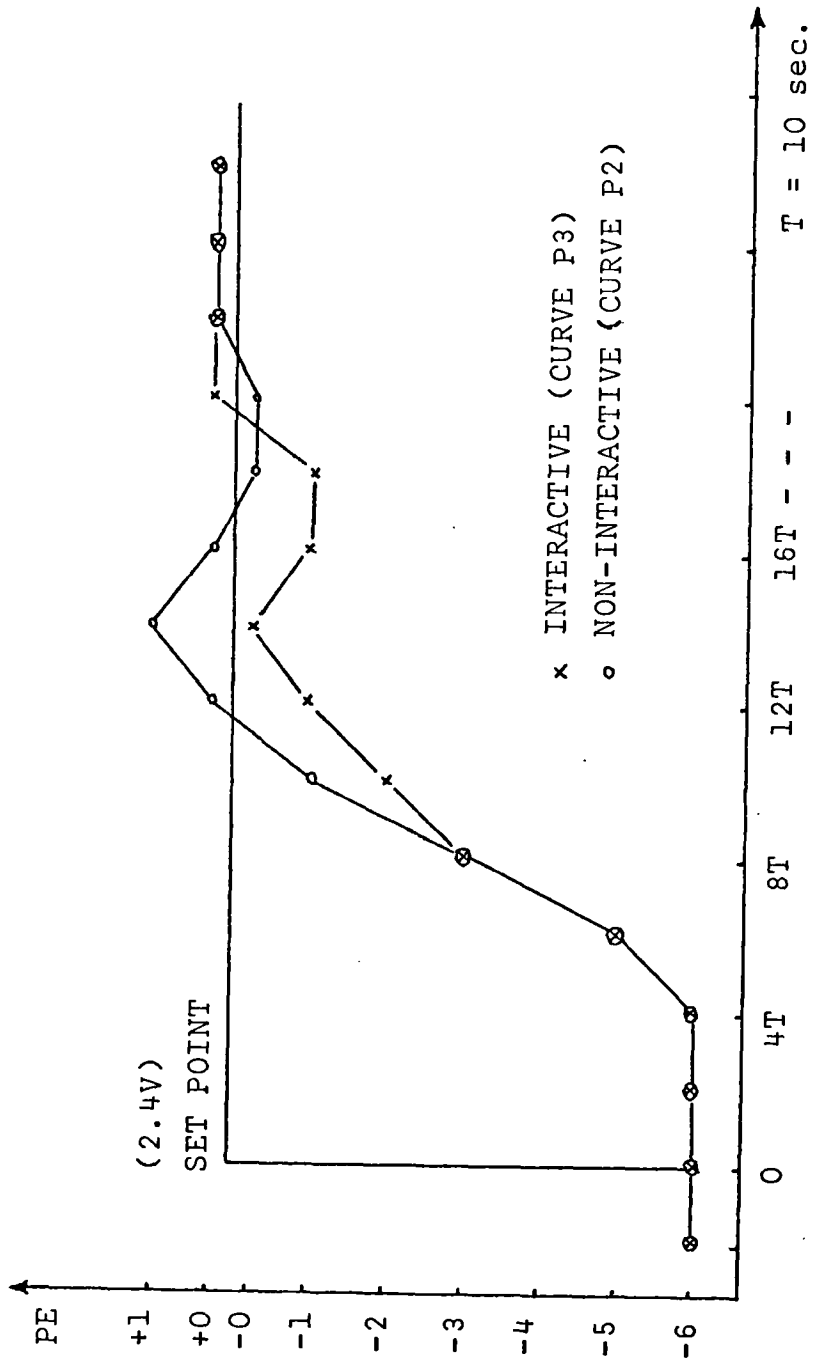


Fig. 7.7 Pressure Response

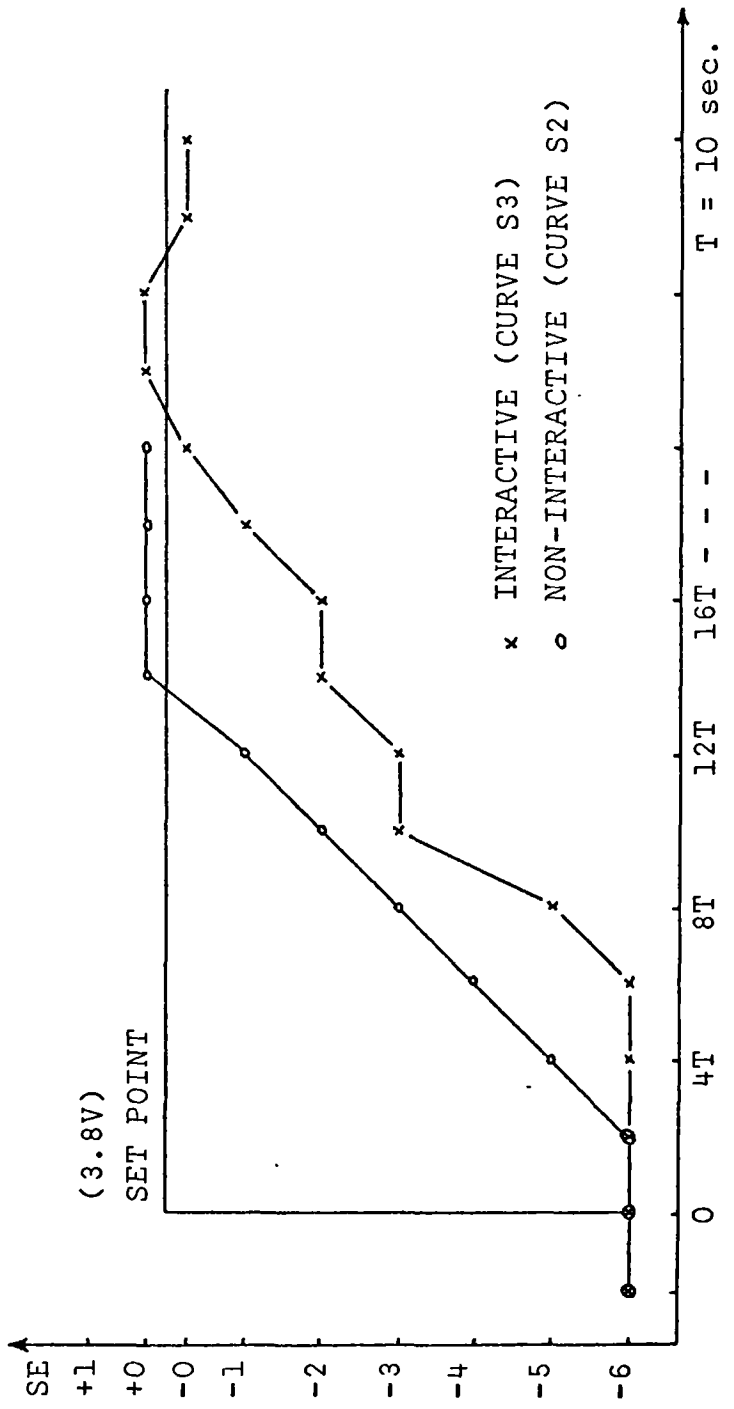


Fig. 7.8 Speed Response

pressure and speed found around the set-points in run C, the heat input that is present at T≥10 is roughly right to maintain a stable state (this can also be verified in the tests carried out for the modelling of the steam engine - see, for example, the initial values of the pressure, speed and heat input given in Tables B.1 and B.2 in Appendix B for Test 3, and compare these values with those given for run C here). This implies that the rate of steam production at T≥10 is comparable with the rate of consumption, so that a better policy at that point in time would be to increase the heat input a little at the same time that the throttle is opened, in order to avoid the drop in pressure.

The implementation of the above suggested modification in the control policy was not tried. However, the argument leading to that suggestion was presented primarily to demonstrate the richness in the 'heuristics' that the human operator can impart to a (learning) machine controller. At the same time, the complexity demanded of an 'intelligent' controller becomes clear. It would be expedient, for example, in the above case, to augment the input vector to the controller by its outputs, so that a policy like "when opening the throttle, increase the heat appropriately" can be implemented conveniently. The consequences of increasing the dimensionality of the input vector in this manner, when adaptive (ATLE or Bayes) controllers are considered, are hardly to be wished for in the light of the arguments presented in Chapters 4 and 5. On the other hand, the formulation of the fuzzy logic controller and its implementation, indicate the great ease with which complexities of this nature can be accommodated in the fuzzy logic approach.

7.6 CONCLUSIONS

It is difficult to draw many conclusions from the experimental results described in this chapter, because no other similar work is available in the literature, apart from certain theoretical investigations and suggestions arising therein (Chang and Zadeh, 1972), so that a comparison can be made with this case study. The question may arise, therefore, as to whether the fuzzy logic deterministic controller described represents a non-trivial alternative to other approaches. Unfortunately, it is difficult to see any direct relationship

between such a controller and any other deterministic controller, such as the direct digital control algorithm or some logic circuit, which has the same input-output capability. Nevertheless, some obvious observations can still be made.

The experimental work described demonstrates the excellent applicability of fuzzy logic theory to the design of controllers. The power of this approach derives from the fact that it enables translation of an entirely unstructured set of heuristics expressed linguistically into an executable control algorithm. The results obtained indicate further, that a controller designed in this manner can produce very effective and very satisfactory control, if some sort of feedback is provided to the designer (the human operator) through which he can systematically modify or optimise the performance of the controller.

The only comparable studies in the literature, where linguistic instructions are used in a similar manner, appear to be those of Waterman (1968, 1970) and Gaines (1972). Waterman uses linguistic instructions for teaching heuristics to a computer programme which is learning to play draw poker. The similarity between Waterman's production rules and the fuzzy logic conditional statement used here has already been pointed out. Gaines, on the other hand, describes a procedure for 'priming' an ATLE-based controller by giving it instructions which it interprets by 'mentally rewarding' itself when 'imagining' itself carrying out the instructions. Remarkably, Gaines further adds that he could not choose a good set of instructions in advance, and that feedback information as to the effect of different instructions formed the basis of choosing the best set. This finding of Gaines and the function that the monitor is designed to provide in this study are clearly very closely related to each other.

As a last remark, it may be desirable in certain applications to have the means of modifying, deleting or adding rules to the fuzzy logic control algorithm on-line, instead of having to recompile the whole programme every time any such alteration is made in it. This enhancement can be provided very simply by writing an 'interpreter' programme between man and computer. Basically, this interpreter needs to have two modes of operation; in the first mode it enquires whether the alteration desired is a modification or deletion of an already existing rule,

or an addition of a new rule, and in the second mode it accepts the corresponding modified or new rule to make the actual alteration.

CHAPTER 8

CONCLUSIONS

8.1 COMPARISON OF CONTROLLERS

Three basic types of controllers have been considered in this study for the control of the steam engine; conventional controllers, adaptive or learning controllers, and controllers based on the fuzzy logic approach. It is interesting now to try and compare these controllers in the light of the experimental work done, bearing in mind the three aspects of main importance to the control engineer, which are implementation, operational characteristics and performance.

Comparing the adaptive (threshold logic and Bayes) controllers with the fuzzy logic controller, it is true to say that both require fairly simple data structures when they are implemented in the form of a computer programme. The data which is stored is either in the form of matrices of weights in the adaptive controllers, or in the form of vectors of membership grades defining fuzzy subsets in the fuzzy logic controller. It is assumed here that in the implementation of the fuzzy logic controller it is the basic definitions that are stored and not the relation matrices, which can be of many dimensions in the general case. However, there is one important difference in the two approaches, in that as the size of the system which is being controlled increases, the storage requirements of the adaptive controller grows enormously, whereas in the fuzzy logic controller this expansion in storage is not as significant.

To illustrate this point, consider the hypothetical case where a system has 10 inputs and 10 outputs each of which is coded into a 10-bit pattern, or in the case of the fuzzy logic approach, each is represented by a fuzzy subset in a universe of 10 elements, with 10 basic values defined for each input or output variable. Therefore, the adaptive controller would require (10x10)x(10x10)=10,000 weights,whereas the

fuzzy logic controller would require 10x((10+10)x10)=2,000 grades of membership. Similarly, with a system having 100 inputs and 100 outputs, the adaptive controller would require (100x10)x(100x10)=1,000,000 weights, whereas the fuzzy logic controller would require 10x((100+100)x10)=20,000 grades of membership. Of course, hidden in the calculation of these figures are a number of assumptions, for example, the number of bits required to represent each weight or each grade of membership. For the sake of argument it is assumed here that the same number of bits can represent both, if the weights refer to those stored in an ATLE - 4 bits are necessary to represent grades of membership 1→10, and compare this with the range of weights obtained for the ATLE in Section 5.5.3 when the single-spot linearly independent code is used. In the case of the Bayes controller, of course more than 4 bits are required normally, since to represent a probability or the logarithm of a probability, real numbers, as opposed to integer numbers must be stored if some degree of accuracy is to be maintained.

It is evident from the implementation described in Chapter 7, that further reduction in the data space of the fuzzy logic controller is conceivable in practice since the same fuzzy value can be assigned to more than one fuzzy variable. For example, the same fuzzy NM value is assigned to both PE and SE variables in the steam engine. Similarly, with the configuration adopted in this study, where a separate matrix is kept for each output variable of the controller, it may not be necessary to include all the measured outputs of the plant in the input pattern of every matrix in the adaptive controller. For example, the input pattern to the heat matrix in the steam engine system may not necessarily include any information about the speed or its time derivative. Clearly, the data space required in the example given in the last paragraph is a great deal reduced in this case. On the other hand, this separation of the multiple control action loops may necessitate the inclusion of other variables in the input pattern, specifically the control variables themselves. An example where this might be desirable was given in Section 7.5.3.

Considerations as those given above lead to a more basic question which arises out of the multi-output nature of the controller that was introduced in this study. In the case of the steam engine, therefore, an alternative configuration, or internal structure that can be adopted in the implementation of the adaptive controllers is to have one matrix of weights, where the implication is that the two actions HC and TC are combined, through same coding, to give one 'composite' action. Thus, action 1 could represent (HC = -7, TC = -2), action 2 could represent (HC = -7, TC = -1), ---, action 75 could represent (HC = +7, TC = +2). One of the advantages in this structure is that dependencies between the control variables can be directly embedded. Refer again to the example given in Section 7.5.3 where it is desired to "Increase the heat input at the same time that the throttle is opened". The main disadvantage, on the other hand, is the considerable increase in data space; with separate matrices for the two control variables the total number of ATLEs is (15+5)=20, whereas here it is (15x5)=75. It is clear that either structure has both its advantages and disadvantages. But if storage requirements are to be minimised, then the structure with separate loops must be preferred.

When the structure of the adaptive (threshold logic or Bayes) controller is described in terms of matrices, a striking similarity becomes apparent between the operational characteristics of the fuzzy logic and adaptive controllers. In both cases the input to the controller is represented by a vector, and if the min operation in the compositional rule of inference (Section 6.3.4) is replaced by the product, and the max operation is replaced by the sum, then the calculations involved in the two controllers are identical. The implications of this remarkable similarity are not immediately obvious, except it is felt that the similarity originates in the fact that both approaches are based on the stimulus-response (S-R) structure. On the other hand, one suggestion that may be made because of this similarity is not recommended to be pursued. Thus it may be suggested to develop procedures similar to the weight adjustment algorithms in the adaptive controllers, which will enable the

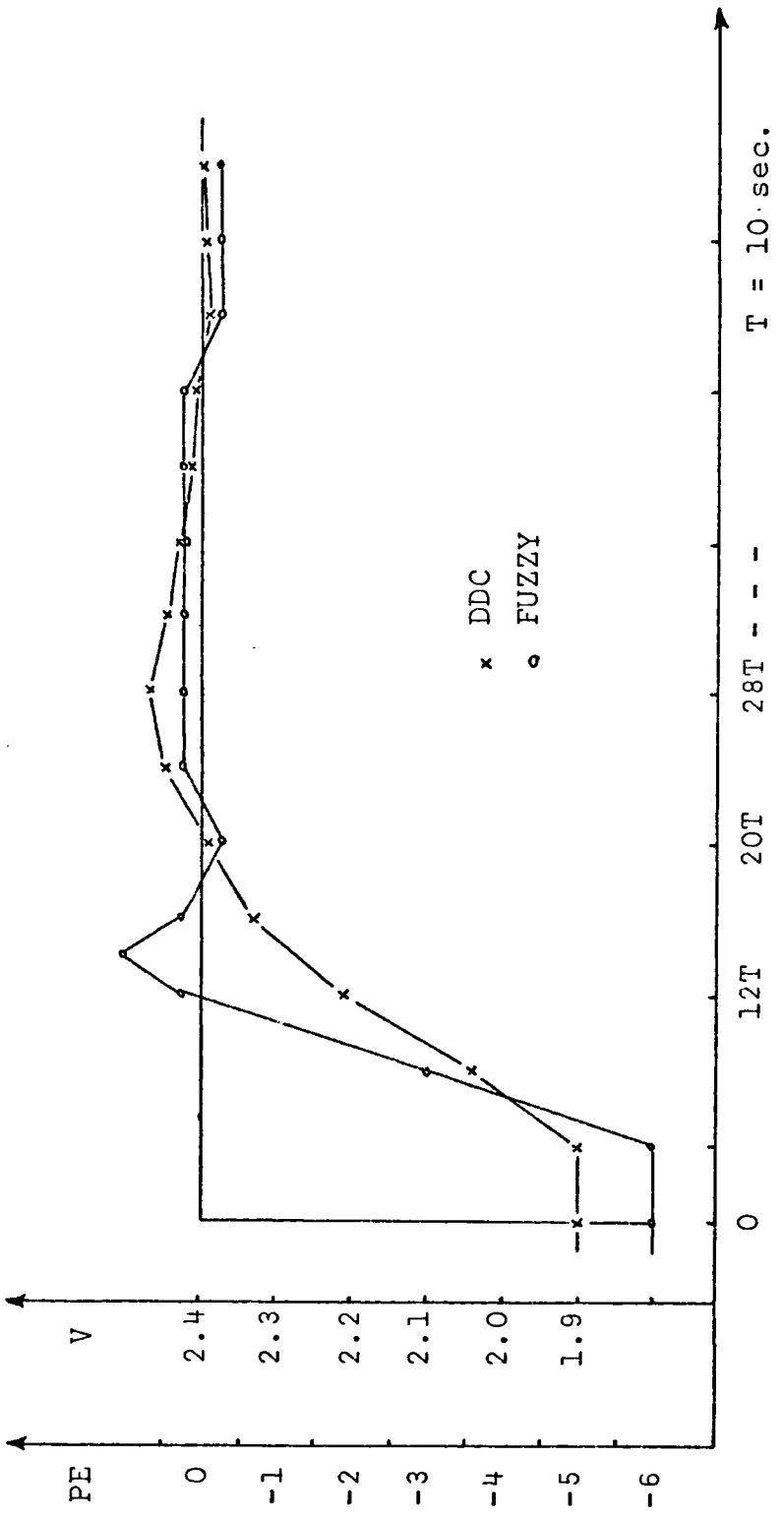


Fig. 8.1 Comparison of control achieved with Conventional and Fuzzy logic controller

the 'learning' of the grades of membership that are involved in a relation matrix. Although feasible, this suggestion defeats the original purpose - the linguistic nature - in the fuzzy logic approach to the problem. It should be preferred, therefore, not to reduce the learning process to the parameter level, but to maintain it at the linguistic level. The possibility of having a self-learning fuzzy logic controller is discussed further in Section 8.3.

In comparing the performance achieved with each of the three controllers that were considered, it is evident from the results reported that the fuzzy logic controller can challenge the conventional controller, whereas the adaptive controllers fall behind in this respect. Fig.8.1 shows the best pressure responses obtained with the fuzzy logic (curve P2 in Fig.7.1) and the conventional (curve H in Fig.3.6) controllers; a choice between the two is difficult. The same is true in the control of the speed variable. The adaptive controller which can be expected to yield the best performance is the one which was trained by the fuzzy logic controller in Section 5.5. But even then, that performance is not likely to be of the high quality appearing in Fig.8.1.

8.2 SUMMARY

The research study described in this dissertation investigates the applicability and utility of AI techniques in control problems. Emphasis is laid on supporting this investigation with evidence obtained through experimentation which is carried out on a real control system. Thus, theoretical investigation establishes the applicability of the proposition, and the practical work provides the evidence of its utility. A critical survey of the relevant literature (Chapter 4) indicates that a few of the better known adaptive or learning control techniques are restricted in their applicability when the general multi-variable and non-bang-bang control system is considered. Such systems are not widely studied in the light of adaptive or learning control theory and very little documentation is available in the literature. The most important restrictions that can be expected in these systems are summarised below.

Prerequisite to the success of an ATLE-based controller is the assumption of linear separability of the input pattern set. This can be a serious limitation in real systems, even when the variables are encoded using techniques, in particular, the linearly independent code, which are known to improve the chances of meeting the above prerequisite. When the performance of a controller is assessed in terms of 'per cent correct recognition' of the input patterns, it is indeed true that even with pattern sets which are not linearly separable, the controller will minimise the number of incorrect classifications. On the other hand, it was indicated (Section 5.4) that such assessment of performance is not really meaningful in control systems with non-bang-bang inputs, where the 'degree' of misrecognition can have much deeper implications than the obvious but simple division into 'correct' and 'incorrect' classifications. Similar arguments apply to the Bayes controller, since the efficiency of this also relies on the clustering properties of the input space. Obviously, with both controllers, clustering features in the input space start fading away, or grow in complexity as the dimensionality of the input space increases, or even more important, as the number of categories increases.

Although the use of the linearly independent code improves the chances of separability in the input space, at the same time it introduces much redundancy in the data that is stored in the memory of adaptive controllers. With multi-variable systems, this can result in the requirement of a vast storage space in a digital computer. Furthermore, the processing of this data can present problems in process control where timing is critical. For example, when the controller is implemented by a computer programme where processing is done serially, the calculation of control actions and the updating of the memory, when on-line training is taking place, can take a long time.

The training of adaptive or learning controllers is perhaps one of the most difficult tasks in the implementation of such controllers. Basically, two modes of training are available, supervised and unsupervised. In the supervised training mode the obvious choice of the teacher is the human operator. Although the use of another, already existing fixed controller is often considered in

theoretical or simulation studies, this would not be done normally in actual practice, since very rarely one would want a second controller. In the unsupervised mode, apart from the storage limitations and the specification of an appropriate performance criterion (Section 4.5.3), a strong objection may also be raised against this method in certain applications. Implicit in this method of learning is the requirement of 'exploration' of the environment in order to find out the best choice of an action for a given state. Clearly, when it is an industrial process that is being controlled, the risk in allowing this can be too high. For example, when the pressure in the boiler of the steam engine is at the extreme high end, the consequences of applying full power to the heater can be disastrous.

The human operator, therefore, is the most suitable, and probably the only teacher to be considered in the training of learning controllers. Indeed, in most studies reported in the literature he has been exclusively given this role, or at least he has had an influence in one way or another. However, the human operator can present serious problems too in practice. In the first place, he is not well suited to the discretized and quantized nature of the environment that he is required to operate in. Secondly, he soon shows the effects of fatigue which make his control actions or policies appear as vagarious. Under these circumstances the adaptive controllers do not exhibit systematic convergence towards a definite control policy, and even if they may do eventually, the time taken for this would be unacceptably long.

The vagaries that the human operator exhibits are not entirely due to fatigue or his unsuitable environment however. Another, and perhaps more significant factor accounting for this is his difficulty in translating linguistically conceived control policies into their numerical equivalent. Thus, the human operator has knowledge of powerful control policies, but he cannot convey this useful knowledge to a machine, the controller, efficiently and uncorrupted in context. The provision of a medium which enables direct verbal communication between man and machine can obviously be seen to be of great importance in many other fields than just control.

It has been shown in this dissertation that one efficient method of translating linguistic instructions into their numerical equivalent is through the use of the semi-quantitative calculus of fuzzy logic. Using this approach, a controller has been formulated which is found to produce excellent performance when controlling the steam engine. Furthermore, it has been shown that by providing the necessary feedback to the human operator to assist in his evaluation of the effects of different instructions, he can very systematically and quickly modify his instructions to produce the optimum controller.

Finally, the generality or flexibility of the fuzzy logic controller has been elucidated by injecting such control policies into its operation which are not conceivable with conventional controllers. For example, apart from fine control around the set-point of the plant, the fuzzy logic controller is equally proficient in start-up or shut-down operations, and moreover, interaction between any input or output variables does not present the problems that are encountered in the design of conventional controllers. In this respect, the adaptive controllers are conceivably capable too, but the additional problems introduced can be quite serious. It is argued sometimes that the great value of adaptive (threshold logic and Bayes) controllers lies in their generality, or capability of solving many varied types of problems, with excellence taking second place. On the other hand, when a control engineer is faced with a particular problem, he is concerned more with excellence, or how 'special-purpose' he can make a technique that is available to him, rather than how 'general-purpose' that technique is.

8.3 RECOMMENDATIONS FOR FUTURE WORK

The results obtained in this single case study of the fuzzy logic approach to control are very encouraging. However, it is evident that the nature of the work is rather heuristic, and furthermore, the study has concentrated solely on the synthesis of controllers. An important and useful extension to this work is therefore to consider the analysis of such controllers, similar to the procedures

followed in classical control theory. Thus, stability and sensitivity analysis should be possible since the actual operation of the method is based on a well-defined calculus.

Another interesting area where further research can be done is the synthesis and analysis of self-learning fuzzy logic controllers. In a sense, the procedure of tuning the fuzzy logic controller as described in this study can be viewed as unsupervised training or learning. The formulation of a self-learning fuzzy logic controller must have at least three distinct functions available:

(a) A method of evaluating the efficiency or 'goodness' of rules.
(b) A method of generating rules.
(c) A method of merging rules.

The work done by Waterman (1968,1970), which is very similar in nature, can be useful in this investigation (Assilian, 1971).

Of course, there are many other questions too that can stimulate further research. For example, can similarly successful results be obtained with systems much larger and more complex than the steam engine in this study? In what way, specially hardware, can a fuzzy logic controller be physically realised other than a computer programme? From the commercial point of view, can a general package be developed which can, through 'initialization', be tailored to the needs of particular applications?

Finally, both the analysis and synthesis of controllers in classical control theory require a mathematical model of the control system. Thus modelling is an important aspect in control studies. In the same way, it should be possible to establish a fuzzy logic model of a system, which can later be used to design a fuzzy logic controller. It is obvious in the synthesis of the fuzzy logic controller described in this study, that the control rules specified by the human operator are based on such a model of the system. However, it is very likely, this model as conceived by the human is not very accurate. Furthermore, in certain areas of the operating space, or indeed for the whole of the control system, the human may not even have a model to work from. Therefore, the development of techniques for building a fuzzy logic model of a system can be of great importance.

BIBLIOGRAPHY

AGMON, S. (1954) "The Relaxation Method for Linear Inequalities," Canadian J. of Math., 6, 3, 382-392.

AKERS, S.B., Jr., RUTTER, B.G. (1964) "The Use of Threshold Logic in Character Recognition," Proc. IEEE, August, 931-938.

ALCORN, T.M., HOGGAR, C.W. (1969) "Pre-processing of Data for Character Recognition," The Marconi Review, First Quarter, 61-81.

ASSILIAN, S. (1971) "A Heuristic Learning Program for the Direct Control of a Plant," Proc. 7th DECUS European Seminar, Amsterdam.

ASSILIAN, S., MAMDANI, E.H. (1974a) "Learning Control Algorithms in Real Dynamic Systems," 4th IFAC/IFIP Conf. on Digital Computer Applications to Process Control, Zurich.

ASSILIAN, S., MAMDANI, E.H. (1974b) "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," Int. J. of Man-Machine Studies (to be published).

BECKER, J.D. (1969) "The Modelling of Simple Analogic and Inductive Processes in a Semantic Memory System," Proc. 1st Int. Joint Conf. on Artificial Intelligence.

BECKER, J.D. (1970) "An Information-Processing Model of Intermediate-Level Cognition," Stanford A.I. Project, Memo AI-119, Stanford Univ. Calif.

BELLMAN, R.E., ZADEH, L.A. (1970) "Decision-Making in a Fuzzy Environment," Management Science, 17, 4, B141-B164.

BLAKE, D.V., UTTLEY, A.M. (Eds.) (1959) Mechanisation of Thought Processes, Symposium, NPL, Teddington, England, London: H.M. Stationery Office.

BOMBA, J.S. (1959) "Alpha-Numeric Character Recognition Using Local Operators," Proc. EJCC, 218-224.

BRAIN, A.E. (1961) "The Simulation of Neural Elements by Electronic Networks based on Multi-Aperture Magnetic Cores," Proc. IRE, 49, 49-52.

BUSH, R.R., MOSTELLER, F, (1955) Stochastic Models for Learning, Wiley, New York.

CADZOW, J.E. (1968) "Synthesis of Nonlinear Decision Boundaries by Cascaded Threshold Gates," Trans. IEEE Computers, C-17, 12, 1165-1172.

CARTER, G.A., MAMDANI, E.H., EVANS, F.J. (1971) "Direct Hybrid Control of an Engine using Adaptive Logic, Adaptive Programming and Self-Organising Storage," Proc. 4th UKAC Congr. on Multivariable System Design and Applications, Manchester, IEE Conf. Pub. No. 78.

CHANG, S.S.L., ZADEH, L.A. (1972) "On Fuzzy Mapping and Control," Trans. IEEE Systems, Man, and Cybernetics, SMC-2, 1, 30-34.

CHERRY, C. (Ed.) (1961) Information Theory, 4th London Symposium, Butterworths.

CHOW, C.K. (1962) "A Recognition Method Using Neighbor Dependence," Trans. IRE, Electronic Computers, EC-11, 683-690.

CHOW, C.K. (1965) "Statistical Independence and Threshold Functions,"Trans. IEEE Electronic Computers (Short Notes), EC-14, 66-68.

CLARKE, D.W. (1967) "Generalized Least Squares Estimation of the Parameters of a Dynamic Model," IFAC Symp. on Identification in Automatic Control Systems, Prague.

DORAN, J. (1969) "Planning and Robots," in (Meltzer and Michie, 1969).

EJIRI, M., UNO, T., YODA, H., GOTO, T., TAKEYASU, K. (1971) "An Intelligent Robot with Cognition and Decision-Making Ability," Proc. 2nd Int. Joint Conf. on Artificial Intelligence, London.

ELGERD, O.I. (1967) Control Systems Theory, McGraw-Hill.

ERNST, G., NEWELL, A. (1969) GPS: A Case Study in Generality and Problem Solving, Academic Press.

ESTES, W.K. (1959) "The Statistical Approach to Learning Theory," in Psychology: A Study of a Science, Vol. 2, S. Koch (Ed.), McGraw-Hill.

EVANS, F.J., FRY, J.P. (1964) "Dynamic Analysis of a Boiler," Part 1, CEGB Computing Branch, Analogue Studies Group Report CD/CB/ASG/R49.

FEIGENBAUM, E.A., FELDMAN, J. (Eds.) (1963) Computers and Thought, McGraw-Hill.

FELDBAUM, A.A. (1963) "Dual Control Theory Problems," Proc. 2nd IFAC Int. Congr., Basle.

FELDBAUM, A.A. (1965) "Optimal Systems," in Disciplines and Techniques of Systems Control, J. Peschon (Ed.), Blaisdell.

FLOYD, R.W. (1964) "The Syntax of Programming Languages - a Survey," Trans. IEEE Electronic Computers, 346-353.

FREEDY, A., HULL, F.C., LUCACCINI, L.F., LYMAN, J. (1971) "A Computer-based Learning System for Remote Manipulator Control," IEEE Trans. Systems, Man, and Cybernetics, SMC-1, 4, 356-363.

FU, K.S. (1970) "Learning Control Systems-Review and Outlook," IEEE Trans. Automatic Control (Expository Papers), SC-15, 210-221.

FU, K.S. (1971) "Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control," IEEE Trans. Automatic Control (Technical Notes and Correspondence), AC-16, 70-73.

GAINES, B.R. (1967) "Techniques of Identification with the Stochastic Computer," Proc. IFAC Symp. on The Problems of Identification in Automatic Control Systems, Prague.

GAINES, B.R. (1968) "Stochastic Computing Systems," Dept. of Elect. Eng. Science, Essex Univ., U.K.

GAINES, B.R. (1971) "The Human Adaptive Controller," Ph.D. Thesis, Psychology Dept., Cambridge Univ.

GAINES, B.R. (1972) "The Learning of Perceptual-Motor Skills by Men and Machines and its Relationship to Training," Instructional Science, 263-312.

GAINES, B.R.,ANDREAE, J.H. (1966) "A Learning Machine in the Context of the General Control Problem," Proc. 3rd IFAC Int. Congr. London.

GIBSON, J.E. (1963) Nonlinear Automatic Control, McGraw-Hill, Chap. 11.

GREENBERG, H.J., KONHEIM, A.G. (1964) "Linear and Nonlinear Methods in Pattern Classification," IBM J., July, 299-307.

GRIFFIN, Jr., J.S., KING, Jr., J.H., TUNIS, C.J. (1963) "A Pattern Identification System using Linear Decision Functions," IBM Systems J., 248-267.

HALL, I.A.M. (1963) "Study of the Human Pilot as a Servo Element," J. Roy, Aero. Soc., 67, 351-360.

HAWKINS, J.K. (1961) "Self-Organising Systems - A review and Commentary," Proc. IRE, 49, 31-48.

HIGHAM, J.D. (1968) "'Single-Term' Control of First - and Second-order Processes with Dead Time," Control, 135-140.

HIGHLEYMAN, W.H. (1961) "A Note on Optimum Pattern Recognition Systems, "Trans. IRE Electronic Computers, EC-10, 287-288.

HIGHLEYMAN, W.H. (1962) "Linear Decision Functions, with Application to Pattern Recognition," Proc. IRE, 50, 1501-1514.

HILBORN, C.G. Jr., LAINIOTIS, D.G. (1969) "Unsupervised Learning Minimum Risk Pattern Classification for Dependent Hypotheses and Dependent Measurements," Trans. IEEE Systems Science and Cybernetics, SSC-5, 2, 109-115.

HILGARD, E.R., BOWER, G.H. (1966) Theories of Learnings Appleton-Century-Crofts, New York.

HILL, J.D., MCMURTRY, G.J., FU, K.S. (1964) "A Computer Simulated On-line Experiment in Learning Control Systems," Proc. SJCC, 315-325.

HU, M.J. (1963) "A Trainable Weather-Forecasting System," Stanford Elect. Labs., Tech. Dept. 6759-1, Stanford Univ., Stanford, Calif.

IDE, E.R., KIESSLING, C.E., TUNIS, C.J. (1968) "Some Conclusions on the Use of Adaptive Linear Decision Functions," Proc. FJCC, 1117-1123.

KANDEL, A. (1973) "On Minimization of Fuzzy Functions," Trans. IEEE Computers, C-22, 9, 326-332.

KING, J.H. Jr., TUNIS, C.J. (1966) "Some experiments in Spoken Word Recognition," IBM Journal of Research and Development.

LEWIS, P.M. (1962) "The Characteristic Selection problem in Recognition Systems," Trans. IRE, Information Theory, IT-8, 171-178.

LOPEZ, A.M., MURRILL, P.W., SMITH, C.L. (1969) "Tuning PI and PID Digital Controllers," Instruments and Control Systems, 42, 89-95.

LURIA, A. (1961) The Role of Speech in the regulation of Normal and Abnormal Behaviour, Oxford:Pergamon Press.

MACFARLANE, A.G.J. (1972) "A Survey of some Recent Results in Linear Multivariable Feedback Theory," Automatica, 8, 455-492.

MAMDANI, E.H., ASSILIAN, S. (1974) "A Case Study on the Application of Fuzzy Set Theory to Automatic Control," Proc. IFAC Symp. on Stochastic Control, Budapest.

MARILL, T., GREEN, D.M. (1963) "On the Effectiveness of Receptors in Recognition Systems," IEEE Trans. Information Theory, IT-9, 1, 11-17.

MATTSON, R.L. (1959) "A Self-Organising Binary System," Proc. EJCC, 212-217.

MATTSON, R.L., DAMMANN, J.E. (1965) "A Technique for Determining and Coding Subclasses in Pattern Recognition Problems," IBM J. of R. and D., 9, 4, 294-303.

MAYS, C.H. (1964) "Comments on Learning and Adaptive Machines for Pattern Recognition," Proc. FJCC, 623-630.

MAYS, C.H. (1965) "The Relationship of Algorithms used with Adjustable Threshold Elements to Differential Equations" Trans IEEE Electronic Computers (Short Notes), EC-14, 62-63.

McCONNELL, J.W, MEADOWS, R.A. (1966) "MOS Adaptive Memory Elements as Weights in an Adaptive Pattern Classifier," J. IEEE Solid-State Circuits, SC-1, 94-99.

McCULLOCH, W.S., ARBIB, M.A., COWAN, J.D. (1962) "Neurological Models and Integrative Processes," in (Yovitz et al, 1962).

McCULLOCH, W.S., PITTS, W. (1943) "A Logical Calculus of the Ideas Immanent in Nervous Activity," Bull. Math. Biophys., 5, 115-133.

MELTZER, B., MICHIE, D. (Eds.) (1969) Machine Intelligence, 5, Edinburgh Univ. Press.

MENDEL, J.M. (1967) "Applications of Artificial Intelligence Techniques to a Spacecraft Control Problem," NASA Contractor Report, CR-755.

MENDEL, J.M., FU, K.S. (Eds.) (1970) Adaptive, Learning and Pattern Recognition Systems: Theory and Applications,Academic Press.

MESAROVIC, M.D. (1962) "On Self Organizational Systems," in (Yovits et al, 1962).

MINNEMAN, M.J. (1966) "Handwritten Character Recognition Employing Topology, Cross Correlation, and Decision Theory," IEEE Trans. Systems Science and Cybernetics, SSC-2, 2, 86-96.

MINSKY, M. (1959) "Some Methods of Artificial Intelligence and Heurisitc Programming," in (Blake and Uttley, 1959).

MINSKY, M. (1961) "Steps Toward Artificial Intelligence," Proc. IRE, 49, 8-30.

MINSKY, M., PAPERT, S. (1969) Perceptrons, MIT Press.

MINSKY, M., SELFRIDGE, O.G. (1961) "Learning in Random Nets," in (Cherry, 1961).

MOTZKIN, T.S., SCHOENBERG, I.J. (1954) "The Relaxation Method for Linear Inequalities," Canadian J. of Math., 6, 3, 393-404.

MUNSON, J.H. (1971) "Robot Planning, Execution, and Monitoring in an Unknown Environment," 2nd Int. Joint Conf. on Artificial Intelligence, London.

NAGY, G. (1968) "State of the Art in Pattern Recognition," Proc. IEEE, 56, 5, 836-862.

NEWELL, A., SIMON, H.A. (1963) "Computersin Psychology," in Handbook of Mathematical Psychology, Vol. 1, R.D. Luce, R.R. Bush, E. Galanter (Eds.), John Wiley and Sons.

NEWELL, A., SIMON, H.A. (1972) Human Problem Solving, Prentice-Hall.

NIKOLIC, Z.J., FU, K.S. (1966) "An Algorithm for Learning Without External Supervision and its Application to Learning Control Systems," Trans. IEEE Automatic Control, AC-11, 3, 414-421.

NILSSON, N.J. (1965) Learning Machines, McGraw-Hill.

NILSSON, N.J. (1969) "A Mobile Automaton: An Application of Artificial Intelligence Techniques," Proc. 1st Int. Joint Conf. on Artificial Intelligence, Washinton, D.C.

NILSSON, N.J., RAPHAEL, B. (1967) "Preliminary Design of an Intelligent Robot," in Computer and Information Sciences 2, J. Tou (ed.), Academic Press.

NOVIKOFF, A. (1962) "On Covergence Proofs for Perceptrons," Proc. Symp. on Mathematical Theory of Automata, Brooklyn, N.Y.

PAPERT, S. (1961) "Some Mathematical Models of Learning," in (Cherry, 1961).

PARZEN, E.A. (1972) Modern Probability Theory and its Applications, Wiley.

PASK, G. (1963) "Learning Machines," Survey Paper, Proc. 2nd IFAC Int. Congr., Basle.

PASK, G. (1971) "A Cybernetic Experimental Method and its Underlying Philosophy," Int. J. Man Machine Studies, 3, 4, 279-337.

PATRICK, E.A. (1972) Fundamentals of Pattern Recognition, Prentice-Hall.

RAE, R. (1968) "On an Automaton in a Non-static Environment," Memo. MIP-R-44, Edinburgh Univ.

ROSENBLATT, F. (1962) Principles of Neurodynamics, Spartan.

ROSENBLATT, F. (1964) "A Model for Experimental Storage in Neural Networks," in (Tou and Wilcox, 1964).

SAMUEL, A.L. (1959) "Some Studies in Machine Learning using the game of checkers," IBM J., 3, 210-220.

SAMUEL, A.L. (1967) "Some Studies in Machine Learning using the game of checkers, II-Recent Progress," IBM J., 11, 601-617.

SANTOS, E.S. (1970) "Fuzzy Algorithms," Information and Control, 17, 326-339.

SAVAS, E.S. (1965) Computer Control of Industrial Plants, McGraw-Hill.

SHANMUGAM, K., BREIPOHL, A.M. (1971) "An Error Correcting Procedure for Learning with an Imperfect Teacher," Trans. IEEE Systems, Man, and Cybernetics, SMC-1, 3, 223-229.

SINGLETON, R.C. (1962) "A Test for Linear Separability as Applied to Self-Organizing Machines," in (Yovitz et al, 1962).

SKLANSKY, J. (1966) "Learning Systems for Automatic Control," IEEE Trans. Automatic Control, AC-11, 1, 6-19.

SMITH, F.W. (1964) "Contractor Control by Adaptive Pattern Recognition Techniques," Stanford Electronics Labs., Stanford, Calif., Tech. Rept. TR 6762-1.

SMITH, F.W. (1966) "A Trainable Nonlinear Function Generator," Trans. IEEE Automatic Control, AC-11,2, 212-218.

SMITH F.W. (1969) "Design of Multicategory Pattern Classifiers with Two-Category Classifier Design Procedures," Trans. IEEE Computers (Short Notes), C-18, 548-551.

SMITH, J.R. Jr., HARBOURT, C.O. (1968) "An Adaptive Threshold Logic Gate Using Capacitive Analog Weights," Trans. IEEE Computers (Short Notes), C-17, 78-81.

SPRAGINS, J. (1966) "Learning Without a Teacher," Trans. IEEE Information Theory, IT-12, 2, 223-229.

SUGIE, N. (1971) "A Model of Predictive Control in Visual Target Tracking," Trans. IEEE Systems, Man, and Cybernetics, SMC-1, 2.

SYMONS, M. (1968) "A New Self-Organizing Pattern Recognition System," Proc. IEE/NPL Conf. on Pattern Recognition.

THORPE, W.H. (1950) "The Concepts of Learning and their Relation to those of Instinct," Symposium of the Society for Experimental Biology, No. 4.

TOU, J.T., WILCOX, R.H. (Eds.) (1964) Computer and Information Sciences, Spartan.

TRUXAL, J.G. (1963) "Adaptive Control," Survey Paper, Proc. 2nd IFAC Int. Congr., Basle.

TURING, A.M. (1950) "Computing Machinery and Intelligence", Mind, 59, 433-460.

WALTZ, M.D., FU, K.S. (1965) "A Heuristic Approach to Reinforcement Learning Control Systems," Trans. IEEE Automatic Control, AC-10, 4, 390-398.

WATERMAN, D.A. (1968) "Machine Learning of Heuristics," Ph.D. Dissertation, Stanford Univ. Report No. CS118, A.I. 74.

WATERMAN, D.A. (1970) "Generation Learning Techniques for Automating the Learning of Heurisitics," Artificial Intelligence, 1, 121-170.

WEE, W.G., FU, K.S. (1968) "An Adaptive Procedure for Multiclass Pattern Classification," Trans. IEEE Computers (Short Notes), C-17, 178-182.

WEE, W.G., FU, K.S. (1969) "A Formulation of Fuzzy Automata and its Application as a Model of Learning Systems," Trans. IEEE Systems Science and Cybernetics, SSC-5, 3, 215-223.

WELCH, J.R., SALTER, K.G. (1971) "A Context Algorithm for Pattern Recognition and Image Interpretation," IEEE Trans. Systems, Man, and Cybernetics, SMC-1, 1, 24-30.

WHITNEY, D.E. (1969) "State Space Models of Remote Manipulation Tasks," Proc. 1st Int. Joint Conf. on Artificial Intelligence, Washington, D.C.

WIDROW, B. (1962) "Generalization and Information Storage in Networks of Adaline "Neurous"," in (Yovits et al, 1962).

WIDROW, B. (1964) "Pattern Recognition and Adaptive Control," IEEE Trans. Applications and Industry, 83, 269-277.

WIDROW, B. (1966)""Bootstrap Learning" in Threshold Logic Systems," Proc. 3rd IFAC Int. Congr., London.

WIDROW, B. (1971) "Adaptive Filters," in Aspects of Network and Systems Theory, R.E. Kalman and N. DeClaris (Eds.), Holt Rinehart Winston.

WIDROW, B., GUPTA, N.K., MAITRA, S. (1973) "Punish/ Reward: Learning with a Critic in Adaptive Threshold Systems," Trans. IEEE Systems, Man, and Cybernetics, SMC-3, 5, 455-465.

WIDROW, B., HOFF, M.E. Jr. (1960) "Adaptive Switching Circuits," IRE Wescon Conv. Record, pt. 4, 96-104.

WIDROW, B., SMITH, F.W. (1964) "Pattern Recognising Control Systems," in (Tou and Wilcox, 1964).

WINOGRAD, T. (1972) Understanding Natural Language, Edinburgh Univ. Press.

WITTEN, I.H. (1973) "Human and Automatic Adaptive Controllers," IEE, Colloquium on Computer Structures for Artificial Intelligence, Digest 1973/16.

YOVITS, M.C., JACOBI, G.T., GOLDSTEIN, G.D. (Eds.) (1962) Self-Organising Systems, Spartan.

ZADEH, L.A. (1965) "Fuzzy Sets," Information and Control, 8, 338-353.

ZADEH, L.A. (1968) "Fuzzy Algorithms,"Information and Control, 12, 94-102.

ZADEH, LA. (1970) "Fuzzy Languages and their Relation to Human and Machine Intelligence," Proc. Int. Conf. on Man and Computer, Bordeaux.

ZADEH, L.A. (1971a) "Similarity Relations and Fuzzy Orderings," Information Sciences, 3, 177-200.

ZADEH, L.A. (1971b) "Toward a Theory of Fuzzy Systems," in Aspects of Network and Systems Theory, R.E. Kalman and N. DeClaris (Eds.), Holt Rinehart Winston.

ZADEH, L.A. (1971c) "Quantitative Fuzzy Semantics," Information Sciences, 3, 159-176.

ZADEH, L.A. (1973) "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," Trans. IEEE Systems, Man, and Cybernetics, SMC-3, 1, 28-44.

ZIEGLER, J.G., NICHOLS, N.B. (1942) "Optimum settings for Automatic Controllers," Trans. ASME, 64, 759-768.

APPENDIX A

DETAILS OF STEAM ENGINE - COMPUTER INTERFACE DESIGN

A.1 HEATER CONTROL

Referring to Fig.2.5, the design and functions of the various components shown are as follows.

The level converters merely match the different operating logic levels of the hybrid computer and the integrated circuit chips.

The function of the sorting circuitry can best be explained in three stages. The first stage determines whether the negative-half cycle of the mains input should be on or off. The circuit achieving this is shown in Fig.A.1. The condition above is given at the output of bistable BI1. The output of the second bistable, BI2, is used for the switching of the 32nd step. It is reminded that a 4-bit binary counter can give 15 counts excluding the zero. Therefore BI2 gives relative count 16 which corresponds to absolute step 32 with the negative-half cycle switched on. Absolute step 16 is of course achieved by simply switching the negative-half cycle on, as described in Section 2.2.2.1.

The second stage of the sorting circuitry simply analyses the state of the counter to determine at what count or step the positive-half cycle of the mains input is out of the possible 15. This is achieved by having 15 replicas of the circuit shown in Fig.A.2, with inputs $A\bar{B}\bar{C}\bar{D}$, $\bar{A}B\bar{C}\bar{D}$, $AB\bar{C}\bar{D}$, etc.

The third and final stage of the sorting circuitry is mainly for scaling purposes of the 32 steps, as described in Section 2.2.2.2. An identical circuit, shown in Fig.A.3, is provided for each of the 15 outputs in Fig.A.2 and the output of bistable BI2 in Fig.A.1. Clearly, at any given instant only one of these 16 steps is 'active', thus switching the appropriate voltage onto the output line, which acts as the input to the SCR control circuit to be described below. Scaling is achieved by adjusting the potentiometers to give the required voltage.

Referring back to Fig.2.5, the SCR control circuit, shown in Fig.A.4a, provides an output to a SCR gate variable from 0 to 10 msec., thus giving control of the conduction angle from 0 to 180°. The variation is controlled by the positive d.c. signal coming from the output of stage three of the sorting circuitry. The operation of the SCR control circuit can be followed by referring to the waveforms sketched in Fig.A.4b.

The last component in Fig.2.5 is a standard Schmitt trigger used for switching the negative-half cycle of the mains.

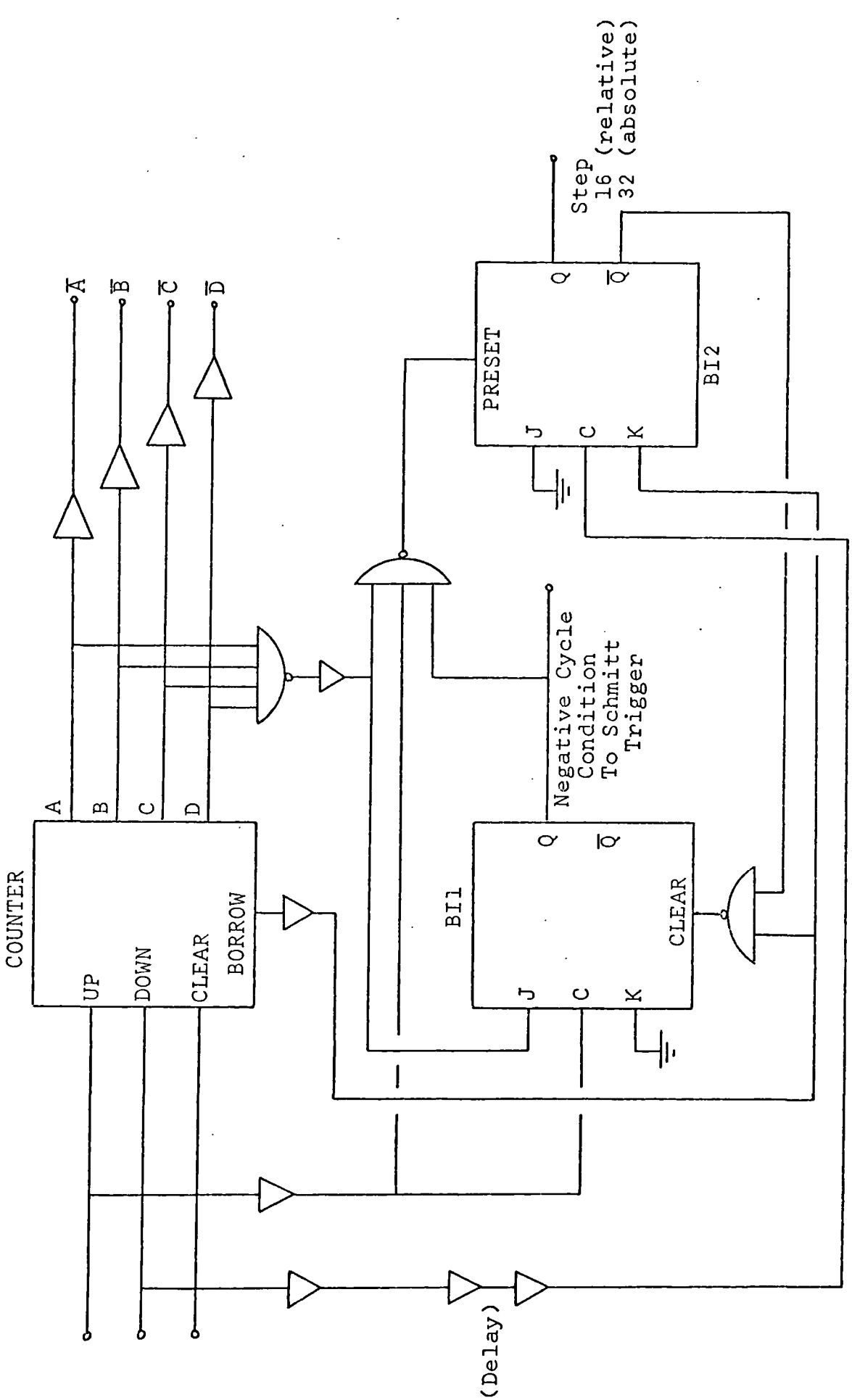


Fig. A.1 Sorting Circuit - Stage 1

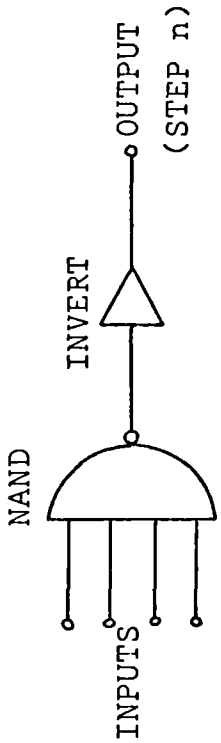


Fig. A.2 Sorting Circuit - Stage 2

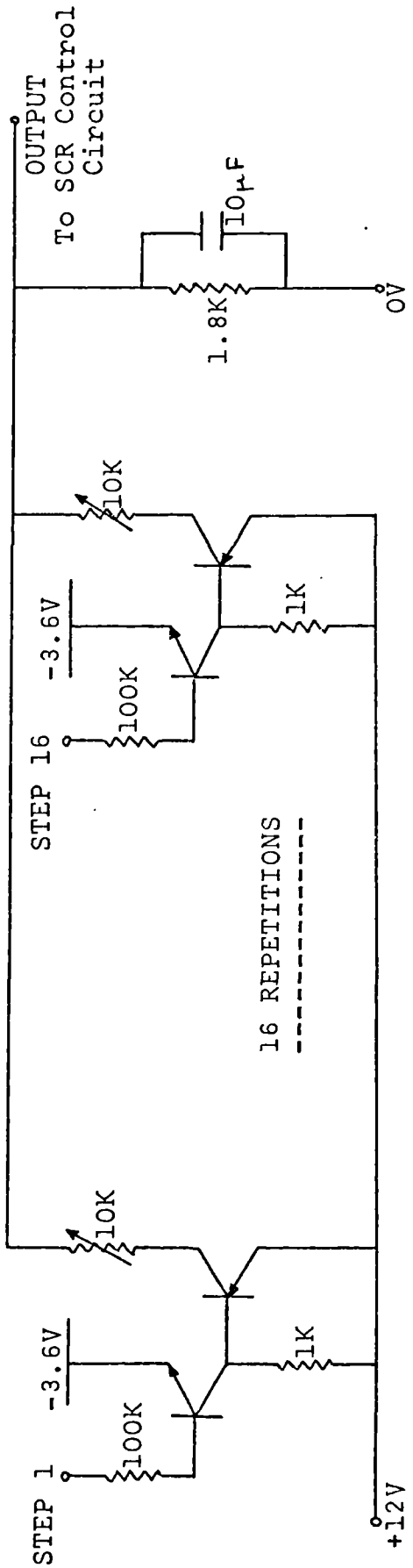


Fig. A.3 Sorting Circuit - Stage 3

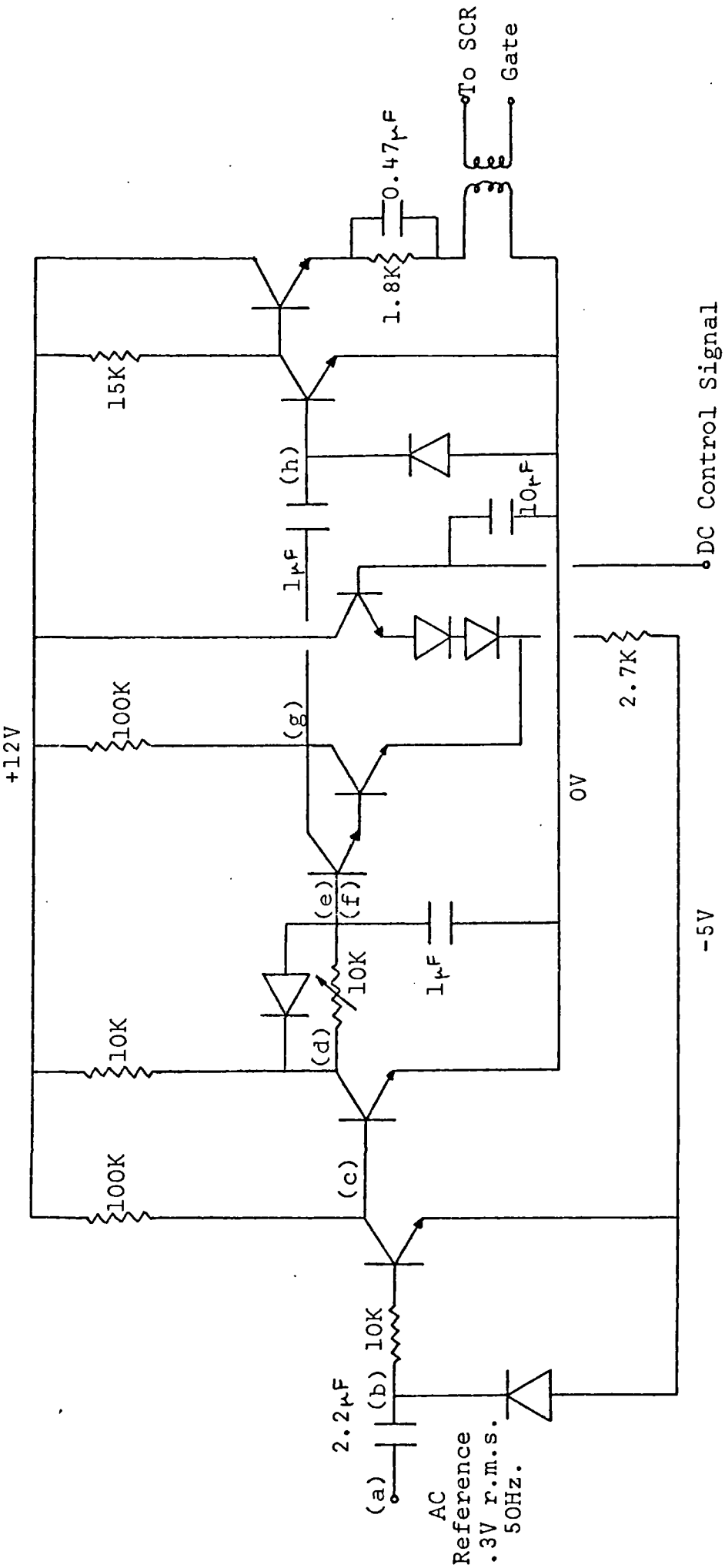


Fig. A.4a SCR Control Circuit

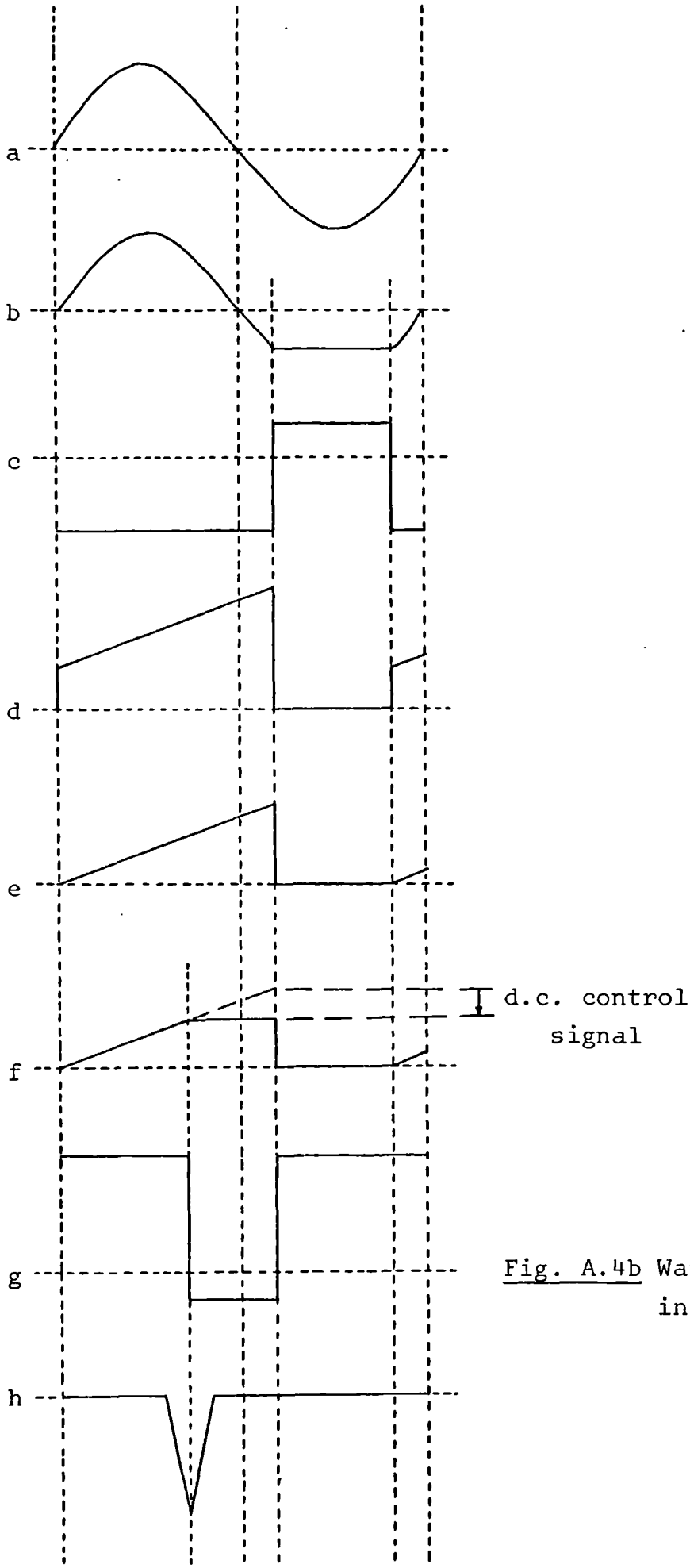


Fig. A.4b Waveforms in Fig. A.4a

A.2 THROTTLE CONTROL

There are two main circuits associated with throttle control; a motor drive circuit and a throttle circuit.

From the description given in Section 2.2.3, the function requested from the motor is that of a stepping-motor. Such a function could be achieved by energizing the motor with a fixed-length pulse which correspondingly turns the throttle through 30°. This was tried but found to be unsatisfactory as the load frictional forces on the motor were too big and varied to give a rotation of the throttle the same number of degrees with every step. The accuracy was also badly affected because transmission of rotational torque from the motor shaft to the throttle plate is through a steel chain which cannot be kept equally taut throughout its length. The design given here allows for these variations by stopping the motor, after energizing it to take a step, only when 30° of rotation is completed by the throttle. The detection of these 30° points is accomplished by the throttle circuit described below. The motor drive circuit is shown in Fig.A.5 and it is noted that the method used for stepping the motor through 30° allows any convenient speed of the motor.

When 30° of rotation is completed by the throttle plate, this condition is detected by the throttle circuit, at which point a pulse is generated and fed back to the 'energize' gate of the motor drive circuit in order to stop it. In addition, the throttle circuit generates a separate pulse when the throttle is in the fully-shut position. This pulse is eventually transmitted to the digital computer to inform it that the throttle is at the origin. The throttle circuit is shown in Fig.A.6. It consists of a 12-way switch which is mounted and kept fixed concentric with the throttle plate. The two brushes shown rotate with the throttle plate and they are aligned such that they close the gap between two adjacent contacts at the same time when one of the holes on the throttle plate is in line with the opening on the boiler. It can be seen from the diagram that the terminal indicated by 'COUNT' gives a positive pulse at the 30° marks, and the terminal

marked 'ORIGIN' gives a positive pulse when the throttle is fully shut.

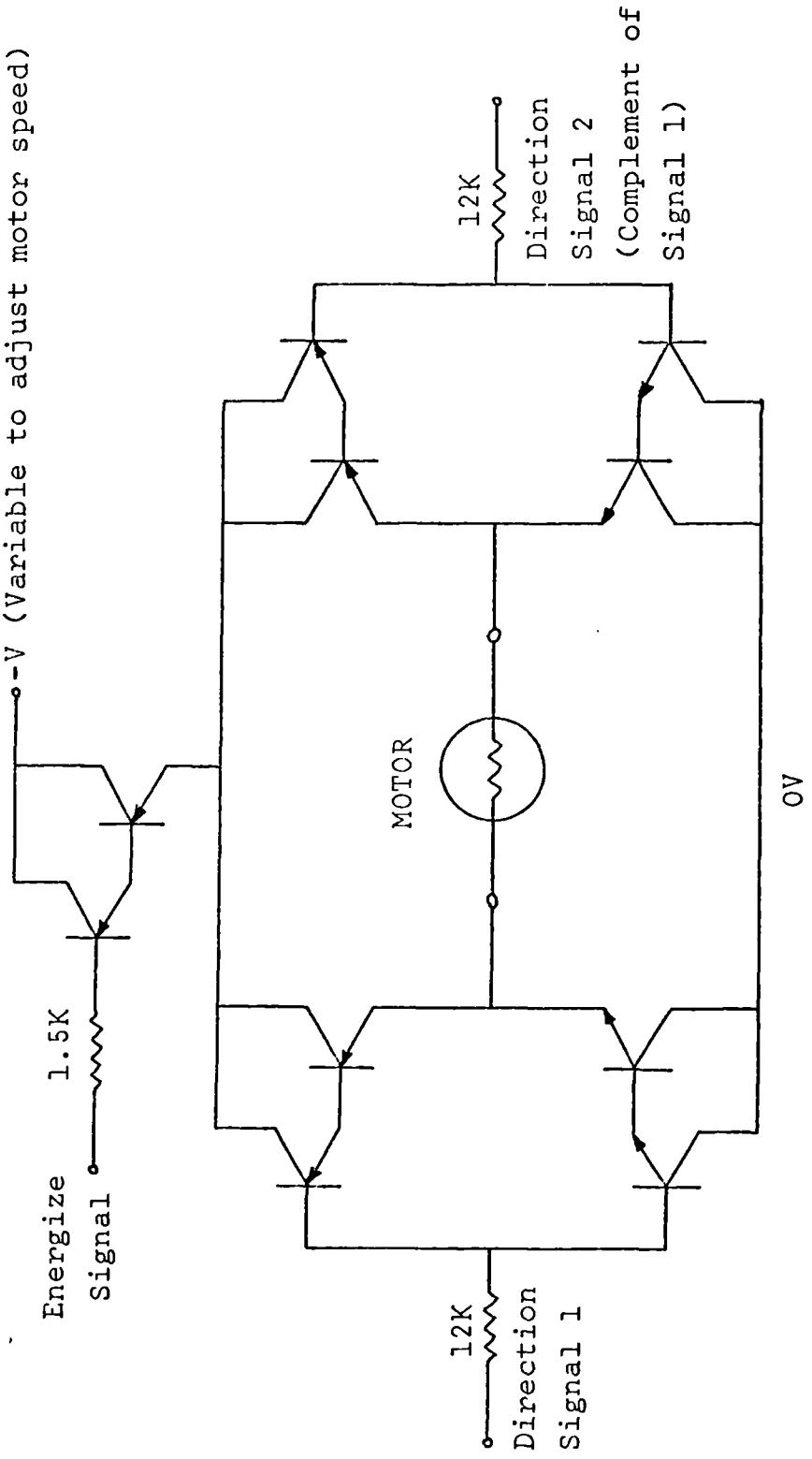


Fig. A.5 Motor Drive Circuit

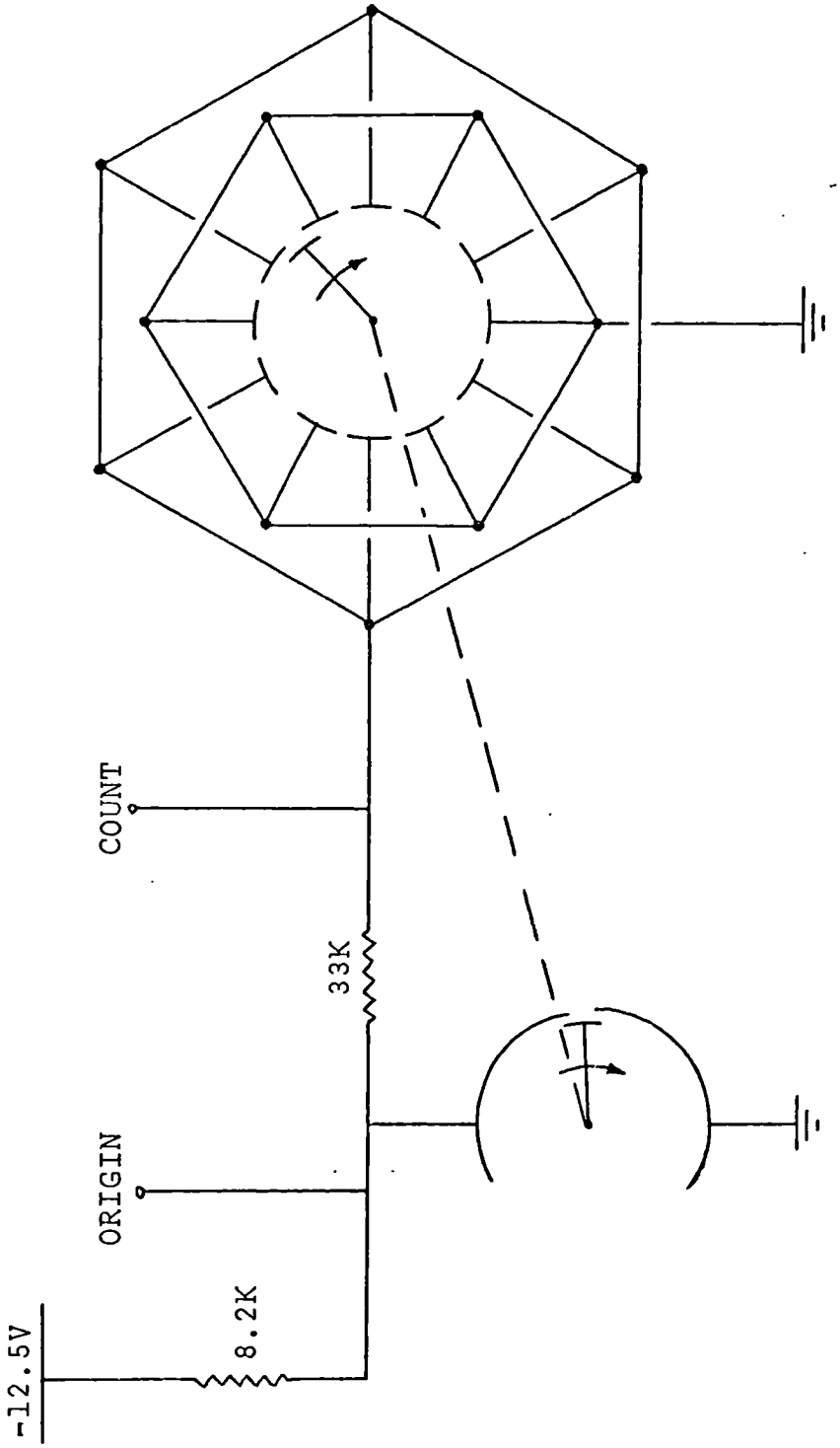


Fig. A.6 Throttle Circuit

A.3 PRESSURE MEASUREMENT

The resistance change of a thermistor is produced by a change of temperature in the semiconductor itself. This may be achieved either by a change in temperature of its surroundings, or internally by heat resulting from dissipation of power in the element. Because the interest here is only in changes of the first type above, it is imperative to keep the power dissipated in the thermistor below the safe maximum quoted by the manufacturers. The thermistor circuit is shown in Fig.A.7. The two potentiometers are used to adjust the relationship between measured voltage and pressure, as described in Section 2.2.4.

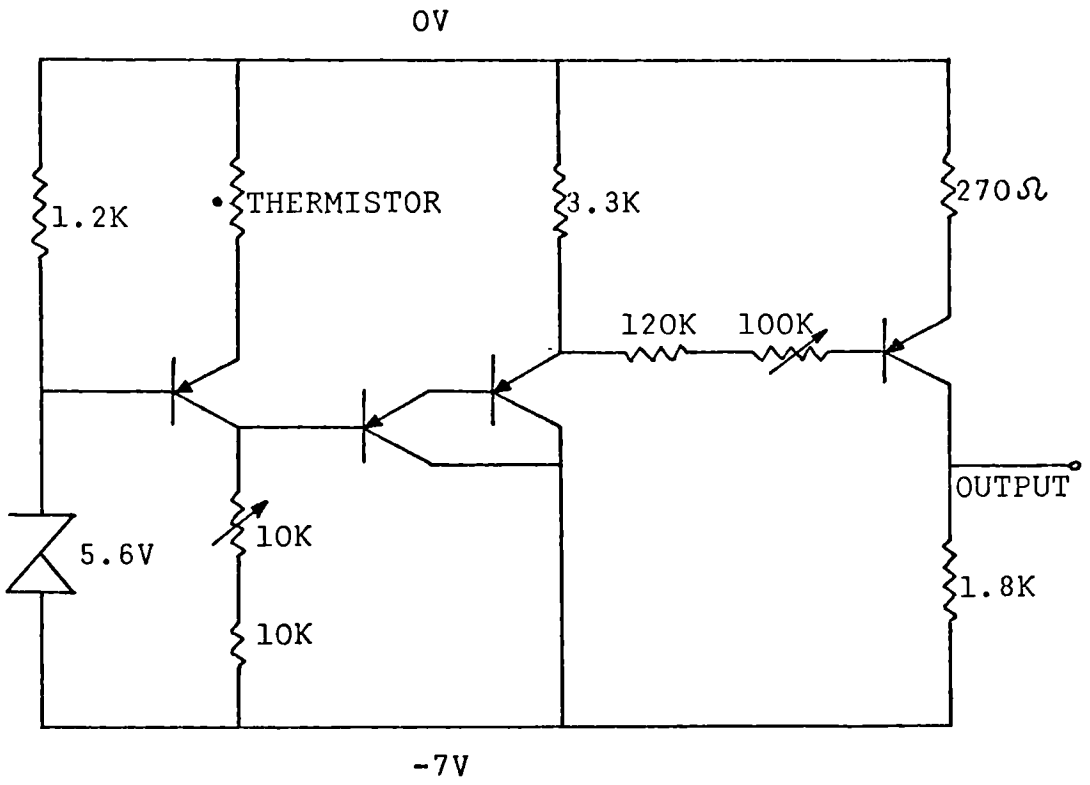


Fig. A.7 Thermistor Circuit

A.4 THE INJECTOR CIRCUIT

Boiler feed injectors require steam for their operation and this is usually provided directly from the boiler as shown in Fig.2.2. A steam valve on the boiler has to be opened to allow the steam to flow into the injector. The purpose of the injector is to replace the water in the boiler that is used up in running the engine. Two water level probes operate the injector by sending make/break signals to the circuit which controls the steam valve. A schematic view of this circuit is given in Fig. A.8.

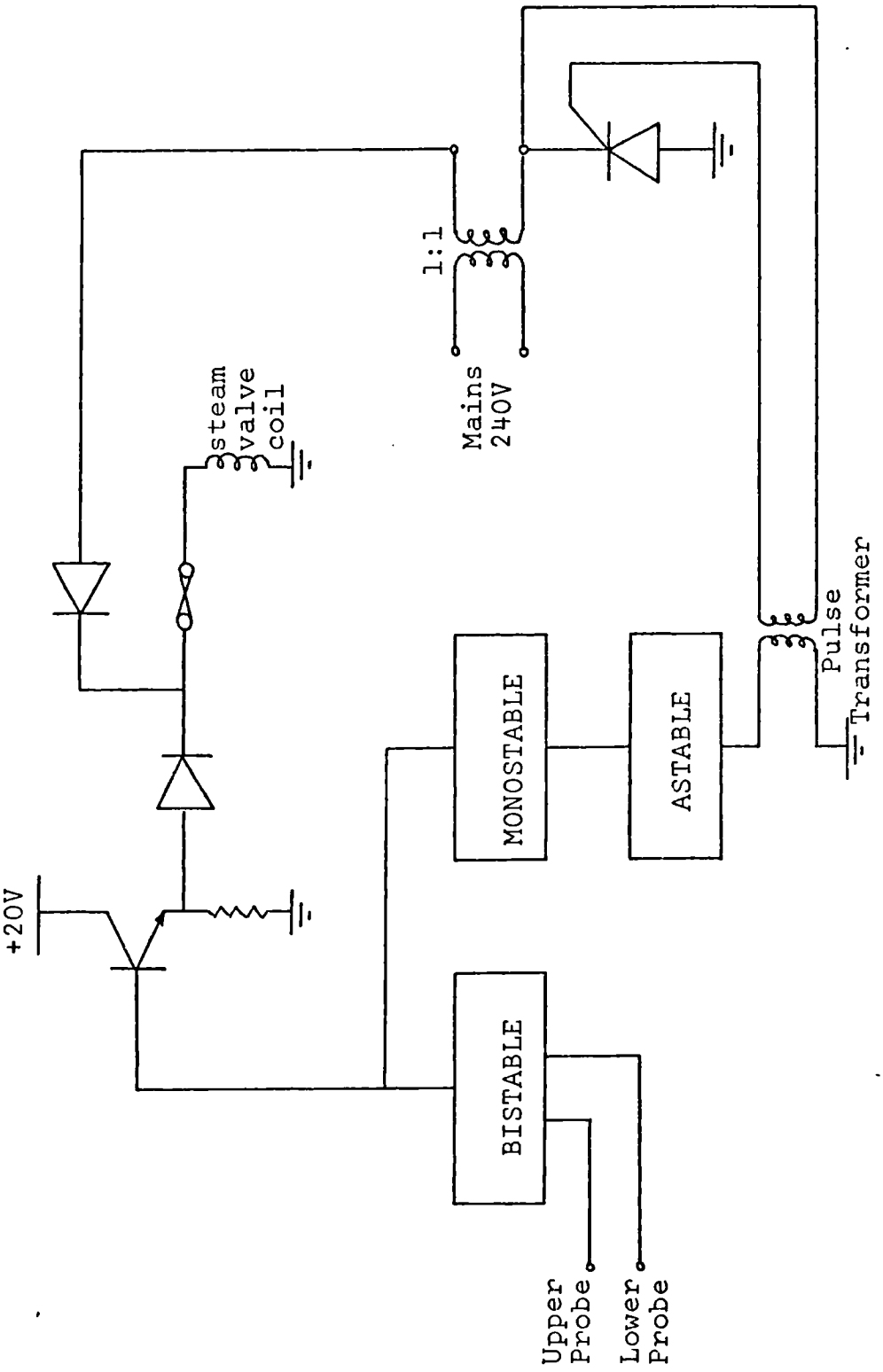


Fig. A.8 Operation of Injector

A.5 SAFETY WATER LEVEL DETECTOR

It is important to make sure that on no account the boiler runs out of water completely as this can cause the heaters to burn out. A probe is provided to detect this 'danger state' and send a warning - an interrupt - to the digital computer to switch the heaters off. The detecting circuitry is shown in Fig.A.9.

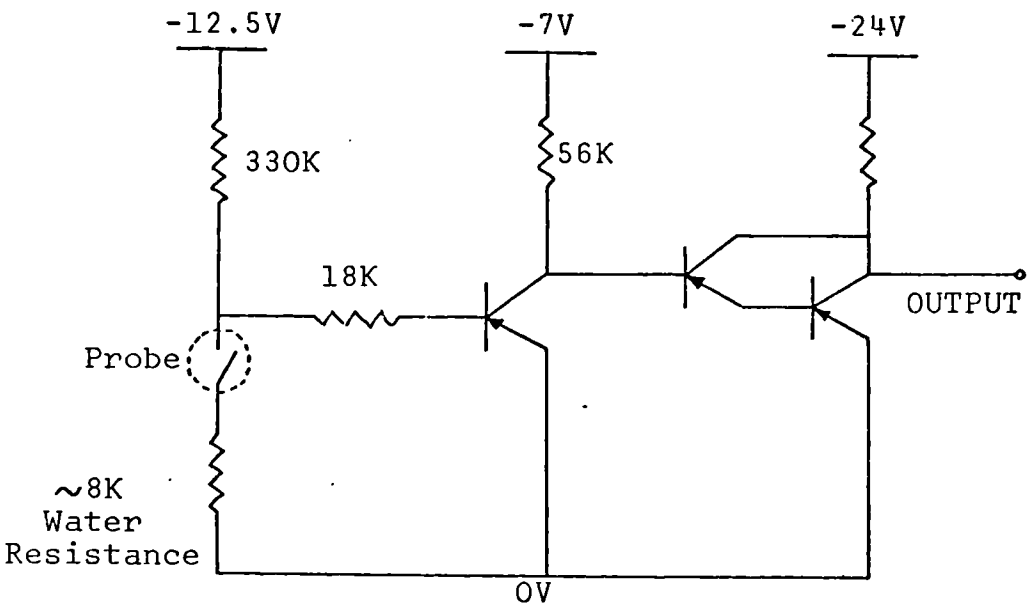


Fig. A.9 Safety Water Level Detector

APPENDIX B

SOME RESULTS OF THE IDENTIFICATION TESTS IN THE DYNAMIC MODELLING OF THE STEAM ENGINE

Some of the typical results obtained in the tests carried out for the dynamic modelling of the steam engine are summarized in Tables B.1 to B.4. Graphical illustrations are given in Figs.B.1 to B.4. Referring to Fig.2.10, the tests summarised in Table B.1 are to estimate $G_1(s)$, Table B.2 for $G_3(s)$, Table B.3 for $G_2(s)$ and Table B.4 for $G_4(s)$.

The abbreviations used are H=Heat, T=Throttle, P=Pressure and S=Speed. The units of heat and throttle used correspond to the quantized steps, as described in Section 2.2. The unit of pressure and speed is voltage. The unit of gain (per step of heat or throttle) is voltage, which is proportional to either pressure or speed from the linear relationships shown in Figs. 2.7 and 2.8.

Finally, it is also noted that in the tests summarised in Table B.4 the pressure was kept constant by closing the heat-pressure loop through the heater controller described in Chapter 3.

Initial Inputs (H/T)	Positive Steps					Negative Steps				
	TEST	Initial State (P/S)	Step	Gain	τ	TEST	Initial State (P/S)	Step	Gain	τ
14/3	01	2.2/4.2	+3	0.08	250	02	2.2/4.1	-3	0.15	260
12/3	03	2.0/3.5	+3	0.11	280	05	2.0/3.6	-2	0.14	280
12/3	04	1.9/3.5	+6	0.11	300					
18/3	06	2.7/4.8	+3	0.15	200	07	2.7/5.0	-3	0.09	300
15/3	08	2.5/4.4	+3	0.08	200	10	2.4/4.6	-3	0.13	275
15/3	09	2.4/4.2	+6	0.06	125					
15/5	11	2.2/4.5	+3	0.08	250	12	2.2/4.4	-3	0.12	250

Table B.1 Step on Heat; Throttle Constant; Pressure as Output

Initial Inputs (H/T)	Positive Steps					Negative Steps				
	TEST	Initial State (P/S)	Step	Gain	τ	TEST	Initial State (P/S)	Step	Gain	τ
14/3	01	2.2/4.2	+3	0.13	260	02	2.2/4.1	-3	0.40	265
12/3	03	2.0/3.5	+3	0.28	125	05	2.0/3.6	-2	0.54	330
12/3	04	1.9/3.5	+6	0.22	300					
18/3	06	2.7/4.8	+3	-	-	07	2.7/5.0	-3	-	-
15/3	08	2.5/4.4	+3	0.16	120	10	2.4/4.6	-3	0.24	200
15/3	09	2.4/4.2	+6	0.12	120					
15/5	11	2.2/4.5	+3	0.08	170	12	2.2/4.4	-3	0.27	340

Table B.2 Step on Heat; Throttle Constant; Speed as Output

Initial Inputs (H/T)	Positive Steps					Negative Steps				
	TEST	Initial State (P/S)	Step	Gain	τ	TEST	Initial State (P/S)	Step	Gain	τ
13/3	13	2.1/4.0	+1	-0.08	250					
15/3	14	2.5/3.7	+1	-0.12	300	15	2.5/4.5	-1	-0.41	320
15/2	16	2.9/4.4	+1	-0.46	220					
15/2	17	2.9/4.3	+2	-0.30	240					

Table B.3 Step on Throttle; Heat Constant; Pressure as Output

Initial Inputs (T)	Positive Steps					Negative Steps				
	TEST	Initial State (P/S)	Step	Gain	τ	TEST	Initial State (P/S)	Step	Gain	τ
3	18	2.2/4.1	+1	0.29	150	19	2.2/4.4	-1	0.88	60
3	20	2.2/4.0	+2	0.15	85					
3	21	2.2/3.7	+2	0.46	110					
4	22	2.2/4.1	+1	0.07	125					

Table B.4 Step on Throttle; Pressure Constant; Speed as Output

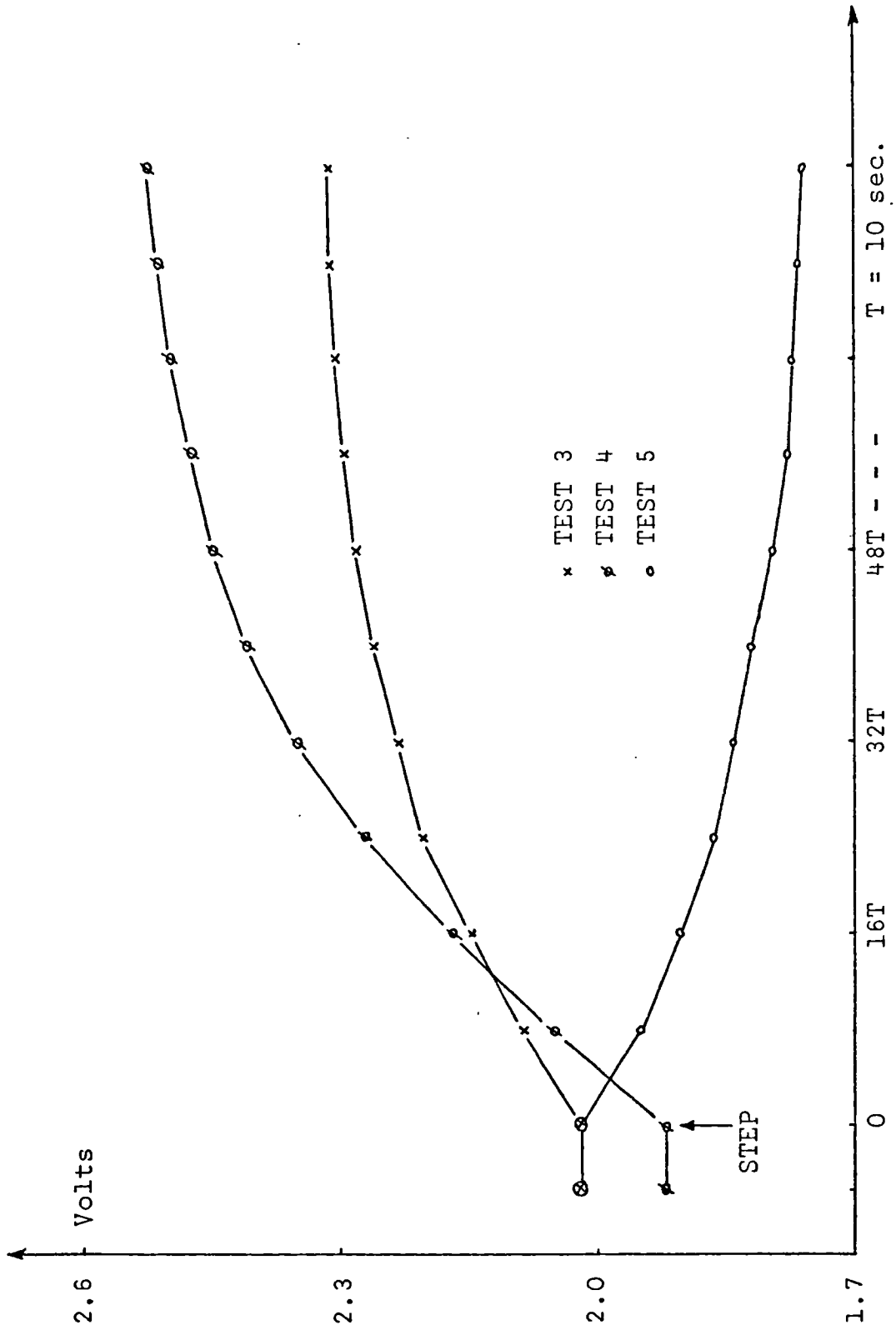


Fig. B.1 Pressure Response (Tests 3, 4, 5)

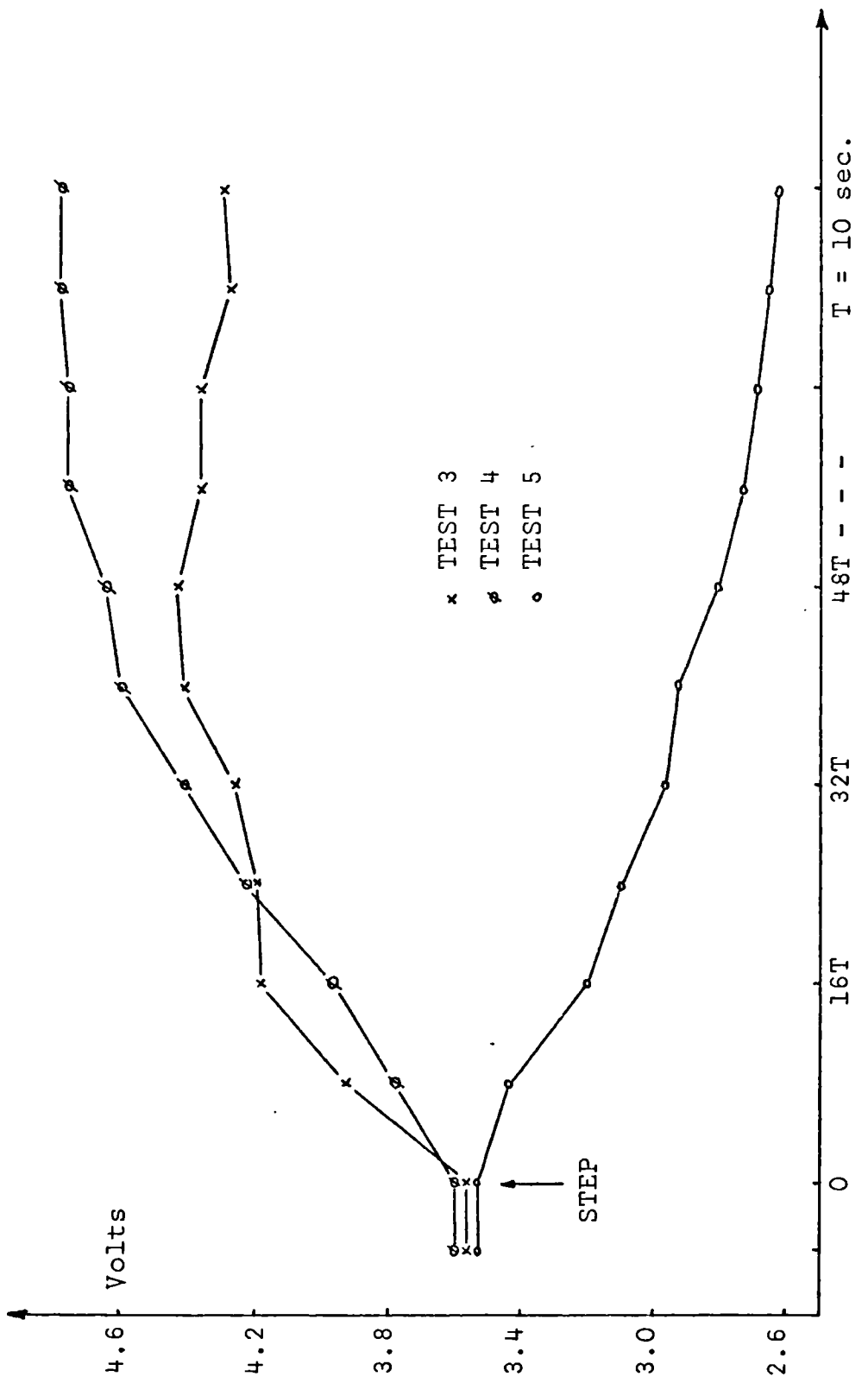


Fig. B.2 Speed Response (Tests 3, 4, 5)

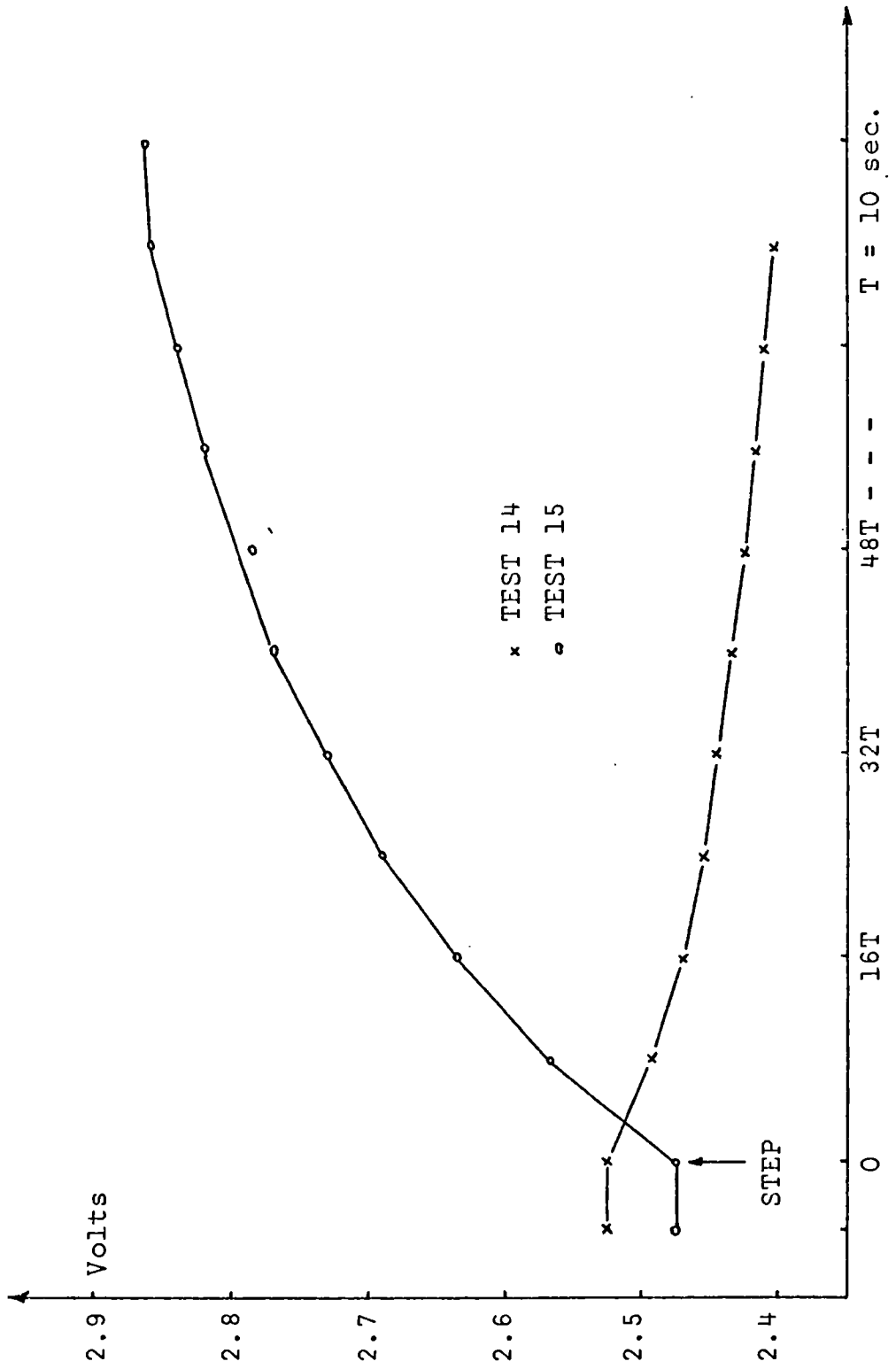


Fig. B.3 Pressure Response (Tests 14, 15)

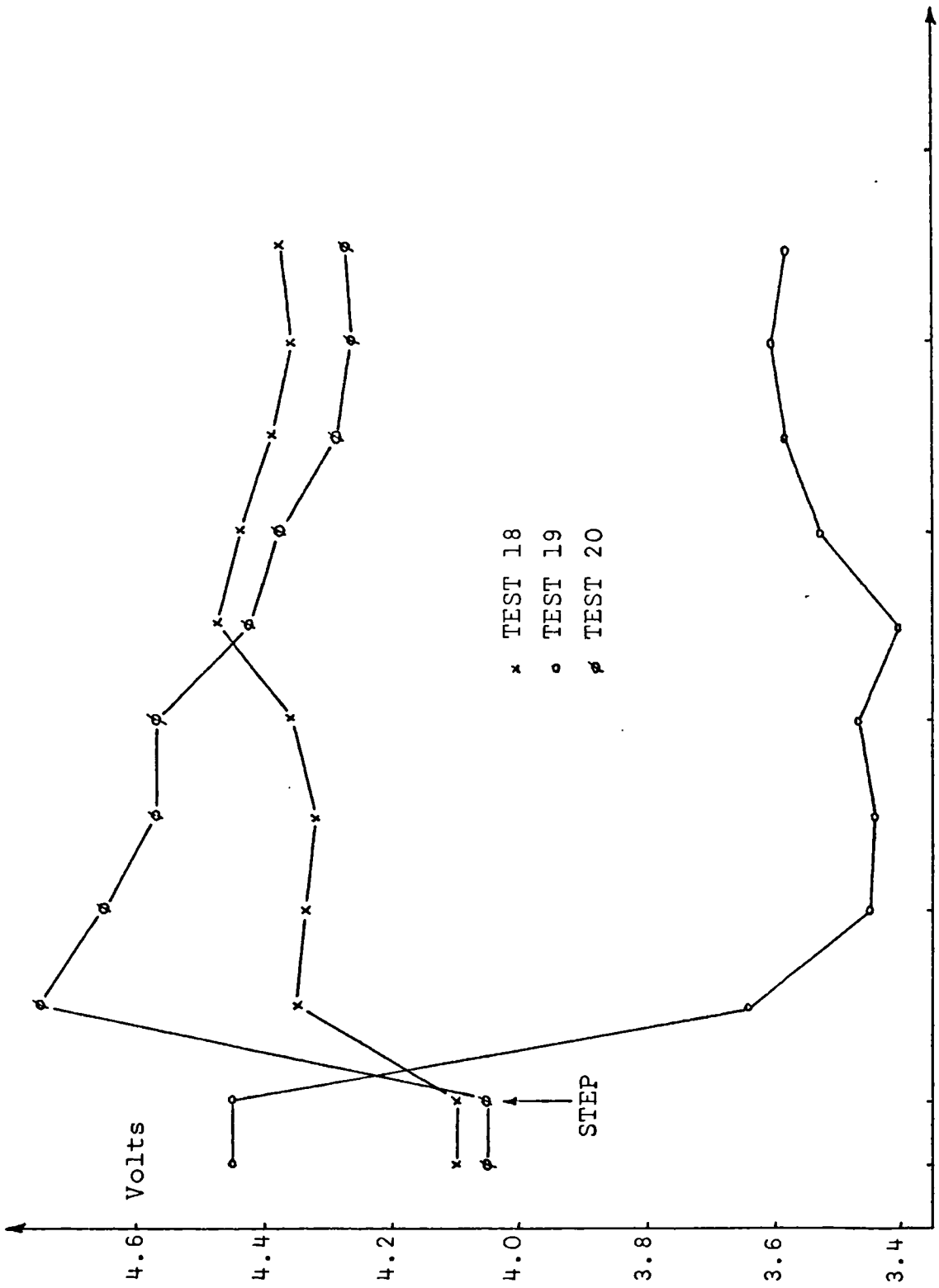


Fig. B.4 Speed Response (Tests 18, 19, 20)

APPENDIX C

DDC CONTROLLER PARAMETERS

C.1 PI CONTROLLER

a. Curve A: Heat-Pressure Loop.

$G = 0.1 \qquad K_c G = 13 \qquad T = 10$

$\tau = 250 \qquad \tau/T_I = 11$

$$\alpha_o = K_c \{1 + \frac{T}{2T_I}\} \simeq 160$$

$$\alpha_1 = -K_c \{1 - \frac{T}{2T_I}\} \simeq -100 \qquad \} \quad \text{(Eqn.3.10)}$$

$$D_H(z) = \frac{160 - 100z^{-1}}{1 - z^{-1}} \qquad \text{(Eqn.3.9)}$$

b. Curve B: Throttle-Speed Loop.

$G = 0.3 \qquad K_c G = 8 \qquad T = 10$

$\tau = 100 \qquad \tau/T_I = 7$

$\alpha_o \simeq 36 \qquad \alpha_1 \simeq -17 \qquad$ (Eqn.3.10)

$$D_T(z) = \frac{36 - 17z^{-1}}{1 - z^{-1}} \qquad \text{(Eqn.3.9)}$$

C.2 PID CONTROLLER

a. Curve C: Heat-Pressure Loop.

$G = 0.1 \qquad K_c G = 15 \qquad T = 10$

$\tau = 250 \qquad \tau/T_I = 10 \qquad T_D/\tau = 0.05$

$$\alpha_o = K_c\{1+\frac{T}{2T_I} + \frac{T_D}{T}\} \simeq 375$$

$$\alpha_1 = -K_c\{1-\frac{T}{2T_I} + \frac{2T_D}{T}\} \simeq -495 \qquad \} \qquad \text{(Eqn.3.10)}$$

$$\alpha_2 = \frac{K_c T_D}{T} \simeq 187$$

$$D_H(z) = \frac{375-495z^{-1}+187z^{-2}}{1-z^{-1}} \qquad \text{(Eqn.3.9)}$$

b. Curve D: Throttle-Speed Loop

$G = 0.3 \qquad K_c G = 11 \qquad T = 10$

$\tau = 100 \qquad \tau/T_I = 7 \qquad T_D/\tau = 0.06$

$$\alpha_o \simeq 52$$

$$\alpha_1 \simeq -48 \qquad \alpha_2 \simeq 16 \qquad \text{(Eqn.3.10)}$$

$$D_T(z) = \frac{52-48z^{-1}+16z^{-2}}{1-z^{-1}} \qquad \text{(Eqn.3.9)}$$

C.3 'SINGLE-TERM' CONTROLLER

a. Heat-Pressure Loop:

$G = 0.1$ $\quad T = 10$

$\tau = 250$ $\quad L = 1-\exp(-T/\tau) = 0.04$

i) Curve E. Choosing $Q = L = 0.04$

$$\Delta D_H(z) = 10 - 9.6z^{-1}$$

ii) Curve F. Choosing $\tau' = 100$

$Q = 0.1$

$$\Delta D_H(z) = 25 - 24z^{-1}$$

iii) Curve G. Choosing $\tau' = 50$

$Q = 0.18$

$$\Delta D_H(z) = 45 - 43z^{-1}$$

iv) Curve H. The controller was chosen as

$$\Delta D_H(z) = 45 - 40z^{-1}$$

b. Throttle-Speed Loop:

$G = 0.3$ $\quad T = 10$

$\tau = 100$ $\quad L = 1-\exp(-T/\tau) = 0.1$

i) Curve I. Choosing $Q = L = 0.1$

$$\Delta D_T(z) = 3.3 - 3z^{-1}$$

ii) Curve J. Choosing $\tau' = 80$

$Q = 0.12$

$$\Delta D_T(z) = 4 - 3.6z^{-1}$$

APPENDIX D

MONITOR PRINTOUTS OF FUZZY LOGIC CONTROLLER

The format of the printout is as follows :

Line 1 - PE CPE HC SE CSE TC
Line 2 - Heat rules contributing to output with highest peak due to every rule.
Line 3 - Spread of heat output vector.
Line 4 - Throttle rules contributing to output with highest peak due to every rule.
Line 5 - Spread of throttle output vector.

This format is repeated at each sampling instant which is 10 secs. If any of lines 2 to 5 are missing, it implies that no rules 'caught' the input state at that sampling instant. The membership grades shown vary [0→10] instead of [0→1]. The pressure and speed set-points and the initial heat and throttle input values are also given at the beginning of each run.

D.1 PRINTOUT FOR RUN A (NON-INTERACTIVE CONTROL)

Pressure Set-Point = 2.4 Volts
Speed Set-Point = 3.8 Volts
Initial Heat = 0 (Steps)
Initial Throttle = 3 (Steps)

```
-6   +3   +7   -6   +0   +2
   HO1 = 10
   00 00 00 00 00 00 00 00 00 00 00 01 04 08 10
      TO1 = 10
      00 00 00 05 10
-6   +4   +7   -6   +3   +2
   HO1 = 10
   00 00 00 00 00 00 00 00 00 00 00 01 04 08 10
      TO1 = 10    TO2 = 2
         00 00 02 05 10
```

```
-6   +0    +7    -6   -6    +0
   HO1 = 10
   00 00 00 00 00 00 00 00 00 00 00 01 04 08 10


-6   -2    +4    -5   -6    +0
   HO1=08  HO2=10
   00 00 00 00 00 00 00 00 00 02 07 10 07 08 08

-6   -3    +4    -5   -6    +0
   HO1=03  HO2=07
   00 00 00 00 00 00 00 00 00 02 07 07 07 03 03

-5   -5    +6    -4   -6    +0
   HO1=02
   00 00 00 00 00 00 00 00 00 00 00 01 02 02 02

-5   -6    +0    -3   -6    +0

-4   -6    +0    -3   -6    +0

-3   -6    +0    -2   -6    -1
     TO5=01
     01 01 01 00 00

-3   -6    +0    -1   -6    -1
     TO5=06
     05 06 05 00 00

-2   -6    -4    -1   -6    -1
   HO5=01
   00 01 01 01 01 01 00 00 00 00 00 00 00 00 00
     TO5=06
      05 06 05 00 00

-1   -6    -4    -1   -5    -1
   HO5=06
   00 02 06 06 06 02 00 00 00 00 00 00 00 00 00
     TO5=06
      05 06 05 00 00

-0   -6    -4    -1   +1    +0
   HO5=10
   00 02 07 10 07 02 00 00 00 00 00 00 00 00 00
     TO6=06
      00 05 06 05 00

-0   -4    -4    -1   -3    +0
   HO5=10 HO6=02
   00 02 07 10 07 02 02 02 02 00 00 00 00 00 00
     TO5=01   TO6=06
      01 05 06 05 00
```

```
+0   +1    +0    -0   -3    +0
   H06=09 H09=03
   00 00 00 01 03 03 03 09 02 00 00 00 00 00 00
      T05=01 T06=07
      01 05 07 05 00

+0   +0    +0    -0   -1    +0
   H06=10 H09=03
   00 00 00 01 03 03 03 10 02 00 00 00 00 00 00
      T06=09
      00 05 09 05 00

-----------------------------------------------

-----------------------------------------------

+0   +1    +0    -0   +6    +1
   H06=09 H09=03
   00 00 00 01 03 03 03 09 02 00 00 00 00 00 00
      T03=03 T04=10
      00 00 05 10 05

+0   +1    +0    -0   +0    +0
   H06=09  H09=03
   00 00 00 01 03 03 03 09 02 00 00 00 00 00 00
      T06=10
      00 05 10 05 00

+1   +1    -2    +0   +6    -1
   H06=06 H09=08
   00 00 00 01 04 08 08 06 02 00 00 00 00 00 00
      T08=10 T09=03
      05 10 05 00 00

+1   +1    -2    -0   +6    +1
   H06=06 H09=08
   00 00 00 01 04 08 08 06 02 00 00 00 00 00 00
      T03=03 T04=10
      00 00 05 10 05

-----------------------------------------------

-----------------------------------------------

-0   +1    +0    -1   -1    +0
   H03=03 H06=09
   00 00 00 00 00 00 02 09 03 03 03 01 00 00 00
      T06=06
      00 05 06 05 00

-0   +2    +0    -0   -6    -1
   H03=03 H04=02 H06=10
   00 00 00 00 00 00 02 10 03 03 03 02 02 02 00
      T05=10
      05 10 05 00 00
```

```
-1   +2    +1    -1   +6    +1
   HO3=08  HO4=02  HO6=06
   00 00 00 00 00 00 02 06 08 08 04 02 02 02 00
      TO3=08  TO4=06
      00 00 05 08 05

-1   +1    +1    -1   -1    +0
   HO3=08  HO6=06
   00 00 00 00 00 00 02 06 08 08 04 01 00 00 00
      TO6=06
      00 05 06 05 00

-1   +1    +1    -0   -6    -1
   HO3=08  HO6=06
   00 00 00 00 00 00 02 06 08 08 04 01 00 00 00
      TO5=10
      05 10 05 00 00
```

D.2 PRINTOUT FOR RUN B (NON-INTERACTIVE CONTROL)

```
Pressure Set-Point = 2.4 Volts
Speed Set-Point    = 3.8 Volts
Initial Heat       = 10 (Steps)
Initial Throttle   = 3  (Steps)
```

```
-6   -1    +7    -6   -5    +1
   HO1=10  HO2=09
   00 00 00 00 00 00 00 00 00 02 07 09 07 08 10
      TO1=02
      00 00 00 02 02

-6   +1    +7    -6   -3    +1
   HO1=10
   00 00 00 00 00 00 00 00 00 00 00 01 04 08 10
      TO1=03
      00 00 00 03 02

-6   -4    +4    -6   -6    +0
   HO2=02
   00 00 00 00 00 00 00 00 00 02 02 02 02 02 00

-6   -5    +6    -5   -6    +0
   HO1=02
   00 00 00 00 00 00 00 00 00 00 00 01 02 02 02

-6   -6    +0    -4   -6    +0

-5   -6    +0    -4   -6    +0

-4   -6    +0    -4   -6    +0

-3   -6    +0    -3   -6    +0
```

```
-2   -6    -4    -3   -6    +0
   H05=01
   00 01 01 01 01 01 00 00 00 00 00 00 00 00 00

-1   -6    -4    -2   -6    +0
   H05=06
   00 02 06 06 06 02 00 00 00 00 00 00 00 00 00

-0   -6    -4    -1   -6    +0
   H05=10
   00 02 07 10 07 02 00 00 00 00 00 00 00 00 00

+0   +5    -4    -1   -6    +0
   H08=08
   00 02 07 08 07 02 00 00 00 00 00 00 00 00 00

+0   +2    -1    -0   -6    +0
   H08=02  H09=03  H15=10
   00 02 03 03 04 08 10 04 00 00 00 00 00 00 00

+1   +2    -4    +0   +6    -1
   H08=02  H09=08  H15=06
   00 02 07 08 07 06 06 04 00 00 00 00 00 00 00
      T08=10  T09=03
      05 10 05 00 00

+1   +1    -4    +0   +0    +0
   H06=05  H09=08  H15=06
   00 02 07 08 07 06 06 05 02 00 00 00 00 00 00
      T06=10
      00 05 10 05 00

+0   -3    +3    +0   -2    +0
   H07=07  H14=07
   00 00 00 00 00 00 00 04 07 07 07 07 07 02 00
      T06=10
      00 05 10 05 00

+0   -5    +4    +0   -0    +0
   H07=08
   00 00 00 00 00 00 00 00 00 02 07 08 07 02 00
      T06=09
      00 05 09 05 00

-0   +3    +3    +0   -1    +0
   H03=03  H04=07  H12=07
   00 00 00 00 00 00 00 04 07 07 07 07 07 02 00
      T06=09
      00 05 09 05 00

-0   -1    -1    +0   +1    +0
   H03=03  H06=05  H13=09
   00 00 00 01 04 08 09 05 02 02 03 03 03 02 00
      T06=09
      00 05 09 05 00
```

D.3 PRINTOUT FOR RUN C (INTERACTIVE CONTROL)

```
        Pressure Set-Point =  2.4 Volts
        Speet Set-Point    =  3.8 Volts
        Initial Heat       = 10  (Steps)
        Initial Throttle   =  3  (Steps)

-6   -1    +7    -6   -1    +0
   H01=10  H02=09
   00 00 00 00 00 00 00 00 00 02 07 09 07 08 10
      T12=10  T15=02
      00 05 10 05 00

-6   +0    +7    -6   -3    +0
   H01=10
   00 00 00 00 00 00 00 00 00 00 00 01 04 08 10
      T12=09  T13=01  T15=02
      01 05 09 05 00

-6   -3    +4    -6   -4    +0
   H01=03  H02=07
   00 00 00 00 00 00 00 00 00 02 07 07 07 03 03
      T12=06  T13=04  T15=02
      04 05 06 05 00

-6   -4    +4    -6   -6    -1
   H02=02
   00 00 00 00 00 00 00 00 00 02 02 02 02 02 00
      T13=10
      05 10 05 00 00

-6   -6    +0    -6   -1    +0
      T12=10
      00 05 10 05 00

-5   -6    +0    -6   -5    -1
      T12=02  T13=08  T16=02  T17=02
      05 08 05 02 02

-4   -6    +0    -6   -4    +1
      T12=04  T13=04  T16=06  T17=02
      04 04 05 06 05

-3   -6    +0    -5   -6    +1
      T13=01  T14=01  T16=08  T17=07
      01 05 07 08 05

-2   -6    -4    -4   -6    +0
   H05=01
   00 01 01 01 01 01 00 00 00 00 00 00 00 00 00
      T16=10  T17=10
      00 05 10 10 05
```

```
-2   -6    -4    -3   -6    +0
   H05=01
   00 01 01 01 01 01 00 00 00 00 00 00 00 00 00
      T16=07  T17=07
      00 05 07 07 05

-1   -6    -4    -3   -6    +0
   H05=06
   00 02 06 06 06 02 00 00 00 00 00 00 00 00 00
      T16=07  T17=07
      00 05 07 07 05

-1   -4    -4    -3   -6    +0
   H05=06  H13=02
   00 02 06 06 06 02 02 02 00 00 00 00 00 00 00
      T16=07  T17=07
      00 05 07 07 05

-1   -1    -2    -2   -5    +0
   H03=05  H06=05  H13=06
   00 00 00 01 04 06 06 05 02 02 05 05 05 02 00
      T17=08  T18=02
      00 05 08 05 02

-0   -1    -1    -2   -2    +1
   H03=03  H06=05  H13=09
   00 00 00 01 04 08 09 05 02 02 03 03 03 02 00
      T18=08  T20=01
      00 01 05 08 05

-0   +0    +0    -2   -3    +1
   H03=03  H06=10
   00 00 00 00 00 00 02 10 02 02 03 03 03 02 00
      T17=01  T18=03  T20=01
      00 01 03 03 03

-1   +1    +4    -2   -6    +0
   H03=08  H06=05  H12=06
   00 00 00 00 00 00 02 05 06 06 07 08 07 02 00
      T17=10
      00 05 10 05 00

-1   +1    +4    -1   -6    +0
   H03=08  H06=05  H12=06
   00 00 00 00 00 00 02 05 06 06 07 08 07 02 00
      T17=08
      00 05 08 05 00

-1   +0    +4    -1   -6    +0
   H03=08  H06=06
   00 00 00 00 00 00 02 06 02 02 07 08 07 02 00
      T17=08
      00 05 08 05 00
```

```
-0   -2    -1    -1   -6    +0
   H05=02  H13=10
   00 02 02 02 04 08 10 04 00 00 00 00 00 00 00
      T17=08
      00 05 08 05 00

+0   +5    -4    -0   -6    +0
   H08=08
   00 02 07 08 07 02 00 00 00 00 00 00 00 00 00
      T17=03
      00 03 03 03 00
```

ACKNOWLEDGEMENTS

The Author is grateful to QMC/Drapers Company for support through a Research Studentship in the course of this work; to Professor M.W. Humphrey Davies for providing the opportunites for carrying out this research; to Dr. E.H. Mamdani for his valuable advice and many fruitful discussions concerning this work; to Dr. F.J. Evans for his interest and encouragement throughout the period of this research; to Mr. T.W. Hobbs for his help and advice in the design and building of the electronic interface between the steam engine and computer; to the Sperry Univac Company for financing in great part the preparation of this thesis; and to Mrs. E. James for typing this thesis.

The Author is especially grateful to his parents for their continued support and encouragement throughout the period of this research.