Synthesis of stochastic learning automata.

NEVILLE, R.G.

1980

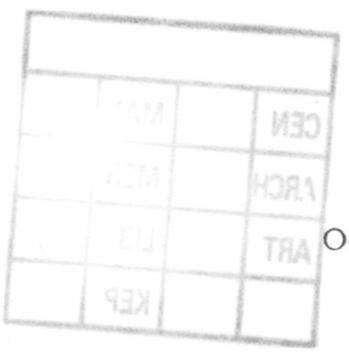
SYNTHESIS OF
STOCHASTIC LEARNING AUTOMATA

by

RICHARD GRAHAM NEVILLE B Sc with
First Class Honours in Electronic Engineering.

A thesis submitted in partial fulfilment of the
requirements of the Council for National Academic
Awards for the Degree of Doctor of Philosophy (Ph D).

School of Electronic and Electrical
Engineering
Robert Gordon's Institute of Technology
Schoolhill
Aberdeen
AB9 1FR



October 1980.

to my Mother .....

DECLARATION

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

In the course of this research, the following were included in an approved programme of advanced studies:

(i) SRC Vacation School on "System Modelling and Optimisation' held at Cambridge University, March 1977.

(ii) SRC Vacation School on 'Stochastic Processes in Control Systems' held at Warwick University, April 1978.

(iii) Workshop on 'Fuzzy Reasoning - Theory and Applications' held at Queen Mary College, London, August 1978.

(iv) 1st International Symposium on "Stochastic Computing and its Applications' held at the Institut National Polytechnique de Toulouse, France, 29 November - 1 December 1978.

(v) Pre-Doctoral Fellowship in the Department of Engineering and Applied Science at Yale University, Connecticut, U S A, August - December 1979.

R G Neville
October 1980

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Professor Philip Mars, for his guidance and encouragement throughout the course of this project.

I would also like to thank various colleagues in the department for numerous helpful discussions, and in particular technicians Colin Nicol and John Still for their assistance with hardware developments.

I hereby acknowledge the Science Research Council for their support of this project, including my appointment as a Research Assistant, and the provision of a travel grant to visit the Department of Engineering and Applied Science at Yale University, U S A. In this connection, I greatly appreciate the co-operation and hospitality extended by Professor Kumpati Narendra and his colleagues.

Finally, I thank Mrs Anne Hobbs for her superb efforts in deciphering my handwriting and typing this thesis.

CONTENTS

ABSTRACT

SYNTHESIS OF STOCHASTIC LEARNING AUTOMATA

by

RICHARD G NEVILLE

Over the past two decades, considerable interest has developed in the field of stochastic learning automata theory and, consequently, the application areas for learning systems. In control engineering, they are viewed as a means to implement optimal adaptive controllers for situations where little or no a priori information on the plant is available.

Stochastic automata with a variable structure operate by means of a global random search, interacting with the environment to improve the action strategy towards optimum performance. They represent therefore a novel and attractive solution to a large class of problems involving high order uncertainties.

At the same time, research has progressed in digital stochastic computing, in which variables are represented by random pulse trains, enabling analogue functions, effectively transformed into Boolean logic operations, to be performed at high speed by conventional digital hardware. These techniques were seen as ideally suited to the practical implementation of stochastic learning automata.

This project is seen as the convergence of these two lines of activity, developing hardware automaton designs and devising applications to simulated and real system parameter optimisation problems. To provide continuity with previous theoretical studies and also lay the necessary foundations of hardware system design experience, basic two-state systems were designed and constructed initially. A standard modular design evolved which was incorporated in a hierarchical structure. This design philosophy enabled large state order automata to be implemented, providing a powerful tool for the optimisation of multivariable, multimodal systems.

A prototype hierarchical structure 128-state automaton has been constructed and tested in both static experiments and a real process control application, based on a small-scale thermal system. The hardware learning automaton approach has been shown here to permit the effective, economic realisation of high-speed real-time system controllers.

CHAPTER 1

BASIC CONCEPTS OF STOCHASTIC LEARNING AUTOMATA

1.1 Introduction

One of the potential application areas for stochastic computing research[1 - 4] is the implementation of learning systems for optimal control by means of stochastic automata.[5] For many control system problems, the characteristics of the process are well understood and a complete mathematical description of both process and control strategy is possible. Conventional programmed-computer control techniques[6] can then be used.

A large class of problems exists, however, which cannot be solved by these techniques, due to incomplete knowledge of plant dynamics, or operation in a random or "noisy" environment. If the probabilistic nature of the uncertainties can be determined, stochastic control theory may provide a solution, but if, as is often the case, the uncertainties are of a high order, no conventional theoretical framework exists. It is in just such a problem area that a "learning" controller, which actually develops the requisite control action by way of real-time on-line interaction with the plant, finds its application.

Stochastic automata have been shown previously[7, 8] to be suitable for the modelling of learning systems in general, and more recently they have been introduced into the field of control engineering.[9 - 16] This project aims to combine the results of earlier work in hardware stochastic computing systems[17, 18] and extensive simulation studies of learning automata[19 - 21] in order to realise their potential in practical control applications for the first time.

Before proceeding to describe the synthesis techniques which /

which have brought about the implementation of hardware stochastic learning automata systems, it is first of all necessary to define what exactly is meant by "automaton", the nature of the "environment" in which it operates, and the manner in which it can be said to "learn". This chapter also contains an exposition of the relevant techniques of digital stochastic computing systems. This in turn will lay the foundations for the following chapters which detail the use of these techniques in the actual synthesis and operation of hardware automata. In general, the notation used here will follow closely that which is standard practice in the literature.

1.2 The Automaton

The concept of "automaton" in the context of the work reported here can be defined as follows. An automaton is essentially a device which is capable of receiving input signals or responses at discrete intervals of time and determining one of a finite number of output actions by means of some intermediate decision-making process acting on its internal structure or state.

The various elements of this broad definition can be stated more precisely as follows:

(i) The input to the automaton, denoted $x(n)$, is an element of the set

$$X = \left\{ x_1, \; x_2, \; - - - \; x_k \right\}$$

where k may be finite or infinite.

(ii) The state of the automaton, denoted $\phi(n)$, is an element of the set

$$\Phi = \left\{ \phi_1, \; \phi_2, \; - - - \; \phi_s \right\} \; ; \; s \text{ is finite}$$

(iii) /

(iii) The output action of the automaton, denoted a(n), is an element of the set

$$A = \left\{a_1, \; a_2 - - - a_r\right\} ; \; r \text{ is also finite}$$

In addition, two functional relationships exist which relate the above variables and complete the definition of the automaton

(iv) The transition function F relates the current state and input at stage n to the next state at stage n+1.

i.e., $\phi(n+1) = F\left\{\phi(n), \; x(n)\right\}$

(v) The output function G relates the current state of the automaton to the resulting output action at stage n.

i.e., $a(n) = G\left\{\phi(n)\right\}$

The automaton is therefore defined mathematically by a quintuple $\left\{X, \; \phi, \; A, \; F, \; G\right\}$, as summarised in Figure 1.1. The functions F and G may be deterministic or stochastic mappings. If F and G are both deterministic, the automaton is denoted a "deterministic automaton", in which case the next state and output action are uniquely defined for a given current state and input. The work to be described here, however, will concentrate on the stochastic automaton, in which F or G, or both, are stochastic functions. In this case, there are only probabilities associated with the succession of states and output actions.

State transition probabilities are defined as follows:

$$p_{ij}^{x} = Pr\left\{\phi(n+1) = \phi_j \; / \; \phi(n) = \phi_i, \; x(n) = x\right\}$$

denotes the probability that the automaton moves from state ϕ_i to state ϕ_j for a given input x. The p_{ij} are the elements /

elements of an $s \times s$ transition matrix T,

i.e., $\phi(n+1) = T\,\phi(n)$

Since $\sum_j p_{ij} = 1$ for all i, in order to preserve probability measure, it follows that T is a stochastic matrix.

In the case of a fixed structure automaton, the p_{ij} have constant values. However, it is frequently found useful to update the transition probabilities at each stage so that the automaton can improve its performance in some respect. In this case, the p_{ij} are a function of n, and the automaton is said to have "variable structure".

An alternative representation of the structure of the automaton can be given in terms of the total state probabilities:

$$\pi_i(n) = \Pr\left\{\phi(n) = \phi_i\right\}$$

or the total action probabilities:

$$p_i(n) = \Pr\left\{a(n) = a_i\right\}$$

Again, to preserve probability measure, it follows that $\sum_i \pi_i = \sum_i p_i = 1$. It is frequently the case that G denotes a one-to-one mapping between states and actions, in which case $\pi(n)$ and $p(n)$ are equivalent.

The choice of representation depends on whether transition or total state probability information is more important for a given problem. However, an important consideration to bear in mind is that the updating process for the latter case involves fewer quantities (s, rather than s^2), which becomes a significant factor in the implementation of large structures.

1.3 /

1.3 The Environment

Within the field of learning automata theory, the "environment" can be defined as the general medium in which the automaton itself is required to operate. The environment thus encompasses all those external factors which influence the structure or behaviour of the automaton. It accepts the output actions of the automaton as inputs, and produces output responses which are in turn fed back to the automaton. The environment is therefore characterised by three sets of variables forming the triple $\{A, C, X\}$ where A and X are respectively the action and input sets of the automaton as defined above, and C is a set of "penalty probabilities", $C = \{c_1, c_2 - - - c_r\}$. In practice, it is convenient to concentrate on the particular automaton-environment configuration in which the set X has just two elements, i.e., $X = [0, 1]$. By convention, $x = 0$ denotes a favourable response or "reward" and $x = 1$ denotes an unfavourable response or "penalty". The work reported here will, in fact, concentrate on this example of a binary environment response, which is classified as the "P-model" (see Figure 1.2).

Each element c_i of C is associated with an element a_i from the action set A, and is defined as follows:

$$c_i = Pr\left\{x(n) = 1 \quad / \quad a(n) = a_i\right\}$$

In general, the normalised environment response, as a measure of some performance index, may be quantised into discrete levels, giving the "Q-model", or represented as a continuous element on the interval $[0, 1]$, giving the "S-model". Although the Q and S-models permit a greater degree of discrimination in specifying the environment response, the P-model has the advantage of greatest simplicity. Also, from the point of view of the research /

research reported here, the inherent binary form is ideally suited to practical implementation with digital circuitry. It was assumed above that the c_i have fixed values, thus defining a "stationary environment". However, a great deal of interest is centred on environments which are non-stationary in some respect, in which case the c_i are functions of time.

1.4 The Concept of Learning

The concept of "learning" is applied here to describe the behaviour of a variable structure automaton operating in an environment as defined above. A learning automaton is capable of determining the success of each action in eliciting a reward from the environment, and, in the specific case of a variable structure device, ordering its structure so as to increase the probability of selecting a more successful action.

Clearly, if the c_i were already known, the strategy of the automaton would be simply to select the action a_m corresponding to the minimum penalty probability c_m. The elements $\{c_i\}$ of C are therefore assumed to be unknown at all times.

1.5 Reinforcement Schemes

A variable structure automaton modifies its policy for selecting output actions by the application of a reinforcement scheme, denoted $f^n \left\{p_1(n) - - - p_r(n)\right\}$, such that

$$p_i(n+1) = p_i(n) + f_i^n \left\{ \right\}, \quad i = 1 - - - - r$$

Again, to preserve probability measure, all such schemes must ensure that

$$\sum_{i=1}^{r} f_i^n \left\{ \right\} = 0.$$

Extensive /

Extensive simulation studies of stochastic automata have been carried out, with a view to application in diverse fields, and much attention has been focussed on the performance of a variety of reinforcement schemes.[22 - 30] In particular, Narendra and Viswanathan have undertaken an exhaustive survey[20, 21] of learning algorithms in order to quantify their performance both in terms of transient response, i.e. learning time, and steady-state behaviour, i.e. asymptotic probability of performing the chosen output action.

As described above, the reinforcement scheme is a function of the total state, or alternatively, the transistion probabilities, and may be linear or non-linear. In addition, schemes have been proposed which combine linear and non-linear forms of updating, usually depending on the current value of $p_i(n)$, in order to obtain the best overall convergence. These are termed hybrid schemes.

As an example, one of the most widely investigated, and indeed earliest proposed schemes, the linear reward-penalty scheme, denoted L_{R-P}, will now be described. The algorithm, stated in total probability form, is as follows:

(a) Reward (action a_i)

$$p_{j=i}(n+1) = \alpha\, p_j(n)$$

$$p_i(n+1) = 1 - \sum_{j=i} p_j(n+1)$$

(b) Penalty (action a_i)

$$p_i(n+1) = \beta\, p_i(n)$$

$$p_{j=i}(n+1) = p_j(n) + \left\{\frac{1 - \beta}{r - 1}\right\} p_i(n)$$

where $0 < \alpha,\ \beta < 1$

In /

In the case of the two-state SLA, which has been widely considered in theoretical and simulation studies, the learning algorithm has a particularly simple form:

(a) Reward (action a_1)

$$p_1(n+1) = 1 - \alpha\, p_2(n)$$

$$p_2(n+1) = \alpha\, p_2(n)$$

(b) Penalty (action a_1)

$$p_1(n+1) = \beta\, p_1(n)$$

$$p_2(n+1) = 1 - \beta\, p_1(n)$$

Similar expressions hold for the case when action a_2 is performed. Reinforcement algorithms will be discussed in more detail in the chapters which follow on hardware synthesis.

1.6 The Automaton/Environment Configuration

The fully annotated automaton-environment interconnected system is as shown in Figure 1.2. This depicts an arrangement, analogous to a closed-loop feedback system, in which automaton actions become environment inputs and output responses from the environment in turn become inputs to the automaton. Also represented as an "input" to the system are the random disturbances, about which the designer has little knowledge and over which he has no control.

Starting from some initial state $\phi_i(0)$, the automaton performs the corresponding action $a_i(0)$. This elicits a response $x(0)$ from the environment, which in turn evokes a change in the state of the automaton to $\phi_j(1)$. In the case of a variable structure automaton, the application of the reinforcement /

reinforcement scheme at each stage alters the total state, or transition, probabilities themselves, so that the relationship between $\phi(n+1)$ and $\phi(n)$ will be updated also.

The iterative process just described, whereby the automaton interacts with a random environment in such a way as to improve its performance, as judged by some criterion, characterises the learning behaviour. In this way, the automaton, if it is to be of some use, should settle into a steady-state condition such that its policy for selecting output actions minimises the received penalty. This corresponds to selecting a_m, i.e., the action with lowest penalty probability c_m. It should be noted that there is always a non-zero probability of penalty even for the optimum action, the important point being that the strategy of selecting this action guarantees the lowest probability of an unfavourable response.

1.7 Criterion of Performance

As described above, the automaton performs a sequence of actions on the environment in the course of its operation, and is deemed to "learn" in the process if its performance can be seen to improve in some respect. A useful criterion for judging the performance of a learning automaton is the average received penalty, denoted by:

$$M(n) = \sum_{i=1}^{r} p_i(n)\, c_i$$

In a "pure chance" situation, an automaton selects each action at all times with constant, equal probability, $p_i(n) = \frac{1}{r}$, $i = 1, 2 - - r$. The value of $M(n)$ in this case is simply the arithmetic mean of the c_i, denoted by:

$$M_o = /$$

$$M_o = \frac{1}{r} \sum_{i=1}^{r} c_i$$

For an automaton to be said to learn, it must achieve a level of performance at least better than pure chance.

A learning automaton is termed "expedient" if the asymptotic average received penalty, in the expected sense, is less than M_o i.e.,

$$\lim_{n \to \infty} E\left[M(n)\right] < M_o$$

A learning automaton is termed "optimal" if the asymptotic value of received penalty is absolutely minimised, i.e.,

$$\lim_{n \to \infty} E\left[M(n)\right] = c_m \; ; \; c_m = \min\left\{c_i\right\}$$

Optimality implies that action a_m is chosen with probability one, representing ideal and in fact impractical conditions. Indeed, in a non-stationary environment it would be highly undesirable for the automaton to lock-on irrevocably to one particular action. Accordingly, a third class of automaton with sub-optimal behaviour is defined. This class of automaton is termed "ϵ-optimal", and achieves a level of performance described as follows:

$$\lim_{n \to \infty} E\left[M(n)\right] = c_m + \epsilon$$

where $\epsilon > 0$ is an arbitrary constant, which may be as small as desired.

It has become general practice to describe a reinforcement scheme in terms of the resulting behaviour of the automaton. That is, if for example a certain reinforcement algorithm produces ϵ-optimal performance from the system, that scheme is referred to as an "ϵ-optimal scheme".

1.8 /

1.8 Digital Stochastic Computing

Stochastic computing techniques[1 - 4, 17, 18, 31] provide an ideal basis for the practical simulation of random processes. Digital stochastic circuits use the probability of switching a digital circuit to represent an analogue quantity. Using this principle, it is possible to implement all the standard operations of summation, inversion, multiplication and integration found on the analogue computer. In addition, a highly flexible, programmable interconnection system can be constructed using standard digital multiplexing elements.[17, 32] Much work has already been done on the development of stochastic-to-digital interface elements,[33 - 36] and the problems connected with the generation of uncorrelated digital noise.[17, 37 - 40]

The great advantage of the digital stochastic machine lies in its unique speed-economy combination, matching the low-cost circuitry of the digital computer with the high-speed parallel processing inherent in the analogue machine.[41 - 44] The result of applying these techniques to the synthesis of stochastic learning automata is that simulations of learning processes which normally require several seconds or even minutes of computer time can be performed in the order of milliseconds on a dedicated hardware system.

Several means of representing variables for the purposes of stochastic computing have been described.[1 - 4] However, since the variables associated with the learning automaton are generally probabilities or constants in the range 0 to 1, it is sufficient to consider the simple unipolar representation. As illustrated in Figure 1.3, a variable is represented here by the probability of a high logic level in a stochastic or random pulse train. This particular example shows two sequences having eight 1's in an interval of twenty clock pulses, representing the value 0.4.

1.9 /

1.9 Stochastic Computing Elements

Using the representation of variables just described, it is found that the resulting stochastic computing elements have particularly simple forms. The multiplication function, as shown in Figure 1.4(a), is performed by a single AND-gate. Since the output of the gate is high only if pulses coincide on the inputs, the probability of a '1' in one pulse train is effectively multiplied by the corresponding probability represented by the other pulse train.

The special case of the squaring circuit is shown in Figure 1.4(b). In this case, a delay of one clock interval is introduced into one of the inputs by means of a D-type flip-flop triggered by the same clock as the noise generator. This preserves the requisite property of statistical independence between the noise characteristics of the actual multiplier inputs which is essential for valid operation.

Inversion, as illustrated in Figure 1.4(c), is simply achieved by passing the signal through a standard inverter, but summation is a rather more complicated function. Although the basic operation involved is logical-OR, it is clear that if just a single gate is used, the output cannot be correct for the condition where pulses coincide on the inputs. It is necessary to maintain the mapping of variables into the range 0 to 1, which in turn dictates that the inputs to the (two-input) summer are limited to a maximum value of 0.5. This is achieved by the circuit shown in Figure 1.4(d). A noise line is used to switch at random between the two inputs, which effectively performs the multiplication by 0.5 and prevents pulse coincidence at the OR-gate inputs. It is evident that repeated summation operations involve successive attenuation of the problem variables, placing an inevitable limitation on the accuracy of the system.

The /

The function of integration is performed basically by feeding the pulse train to a counter (Figure 1.5). Clearly, the sum of pulses accumulated over a certain time interval, or number of clock pulses, represents the time integral of the variable represented by the signal.

The interface between a deterministic signal, presented in digital form, and the stochastic machine is shown in Figure 1.6. The input signal is compared with a random number sequence in a standard digital comparator. Provided that the noise source has a uniform distribution, the probability of a logic '1' output from the comparator is directly proportional to the value of the input number as a fraction of full scale.

The output of the stochastic machine can be converted to either an analogue or a digital value. The analogue output interface in its simplest form consists of a first-order R-C low pass filter, since all that is required is to derive the average value or 'd-c' component of the bit stream.

Stochastic-to-digital conversion involves a similar averaging process, though the circuitry is of necessity rather more complex. The standard configuration, as shown in Figure 1.7, is called a noise ADDIE, abbreviated from "Adaptive Digital Logic Element".[1, 33 - 36] An up-down counter is connected to a noise comparator, and a feedback loop arranged as shown. As a result, the counter will count up or down until $p(F) = p(A)$, effectively providing a continuous estimate of the input probability value.

In conclusion, it should be stressed once again that all the computing elements described above are implemented using entirely standard hardware components.

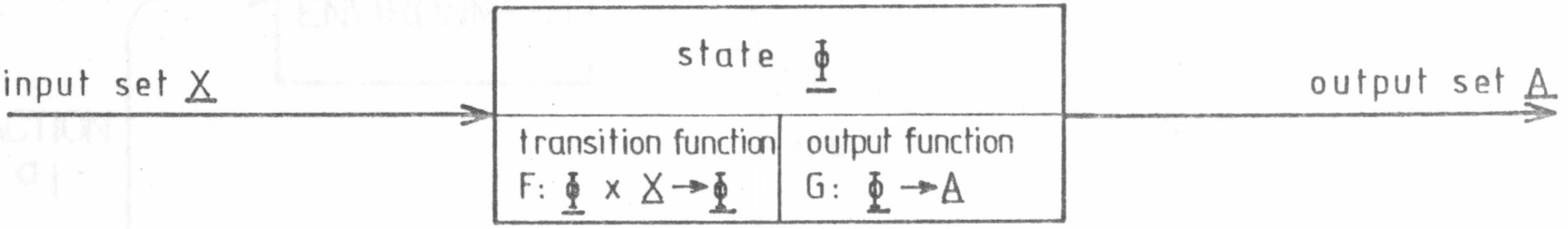


Figure 1.1 The stochastic learning automaton

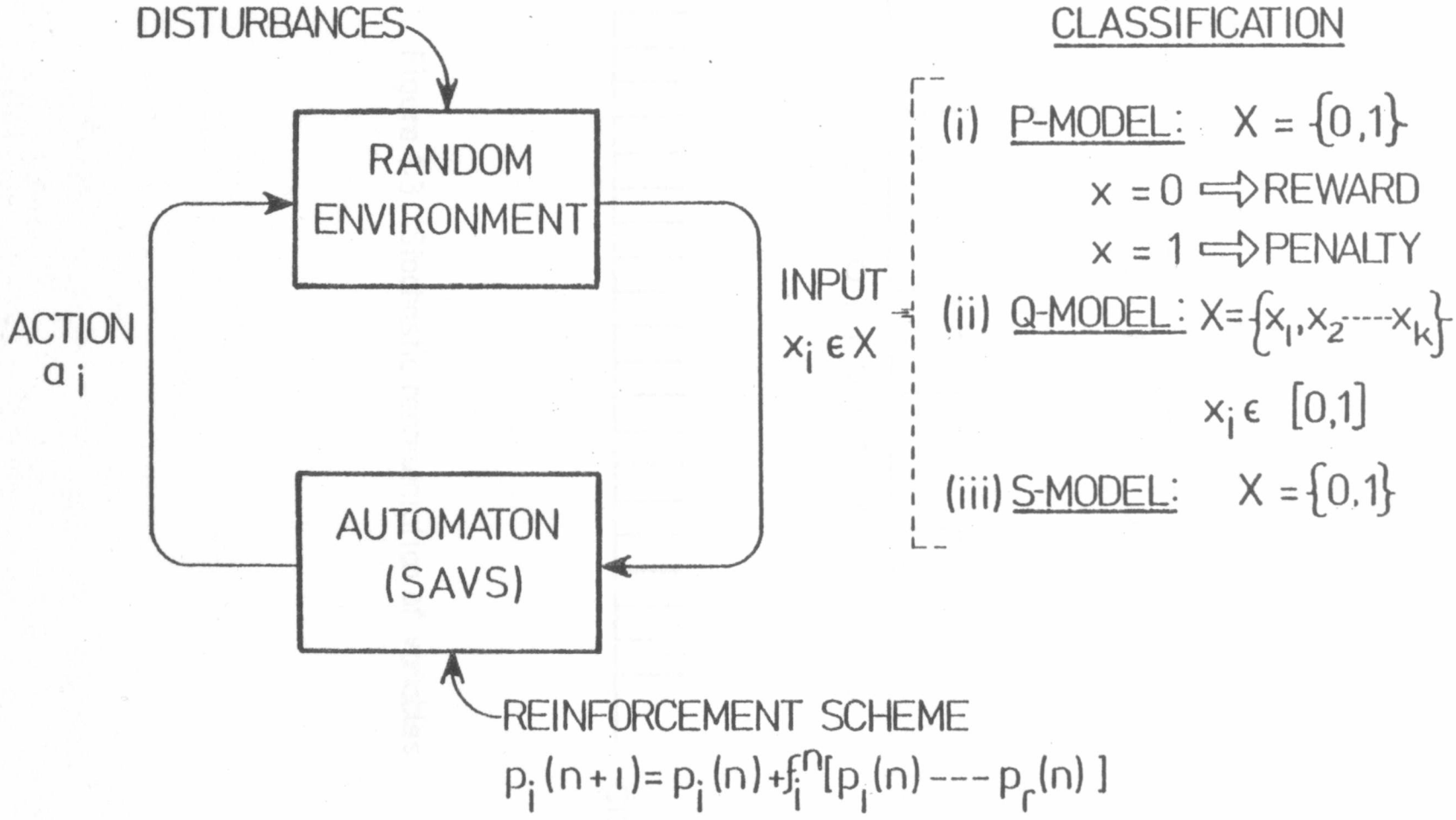


Figure 1.2 Automaton - environment configuration

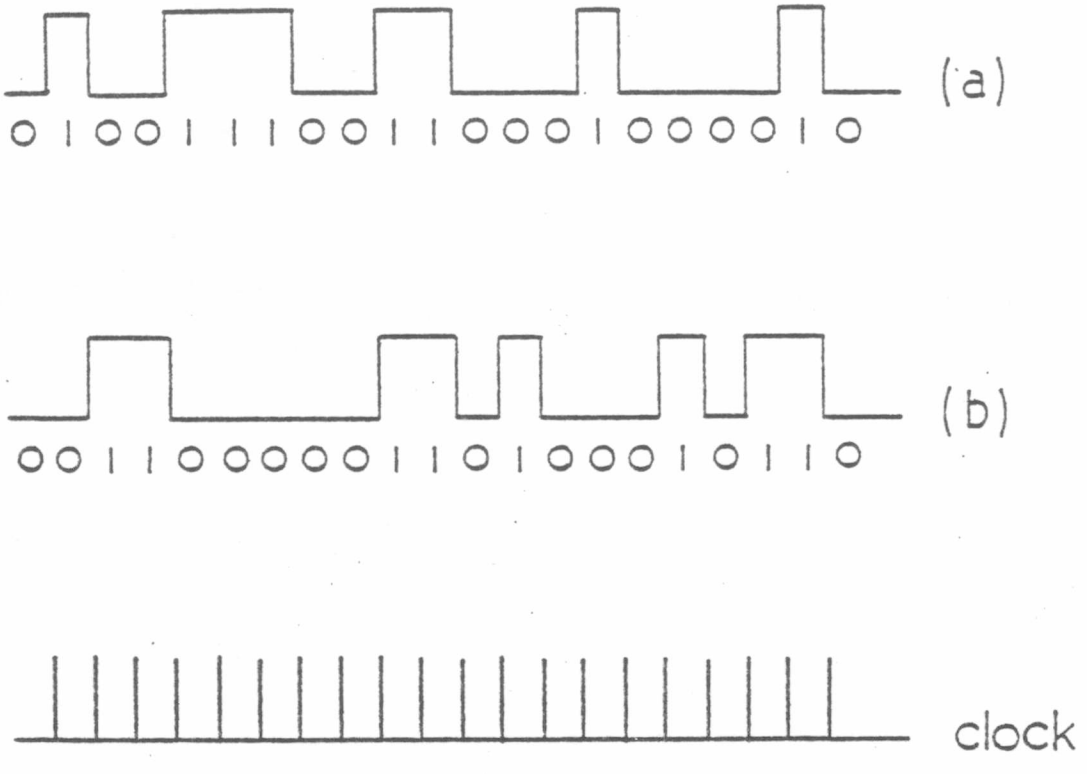
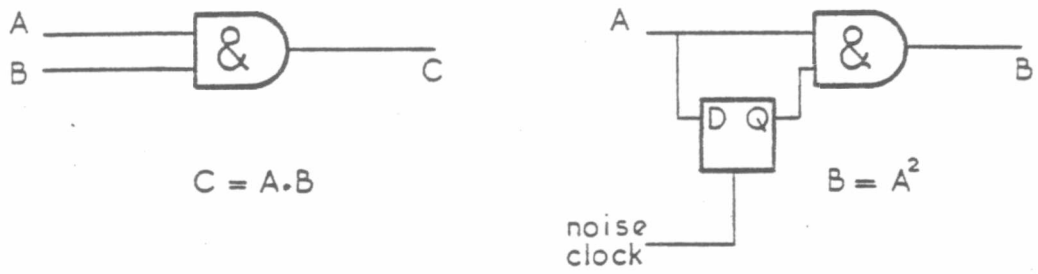
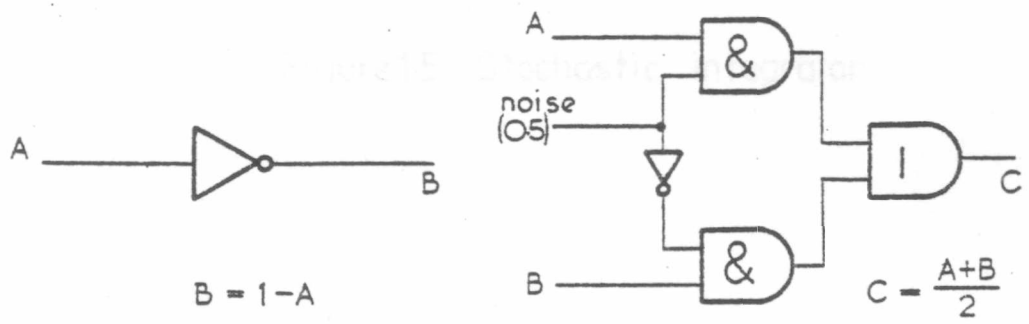


Figure 1.3 Stochastic representation of variables



(a) multiplier (b) squarer



(c) inverter (d) summer

Figure 1.4 Stochastic computing elements

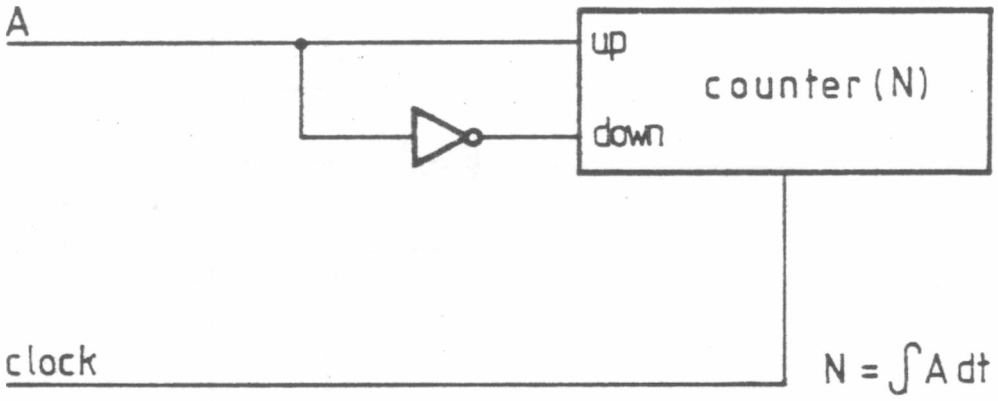


Figure1.5 Stochastic integrator

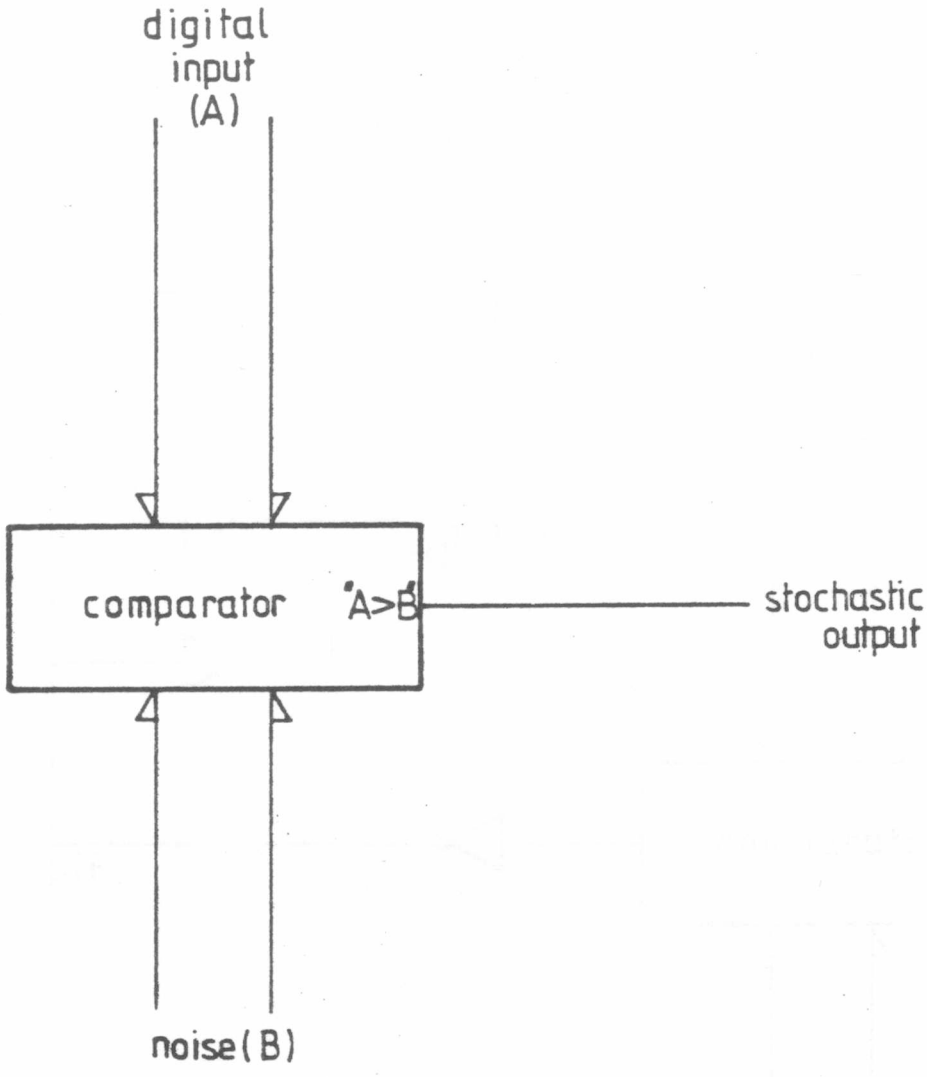


Figure 1.6 Digital-to-stochastic conversion

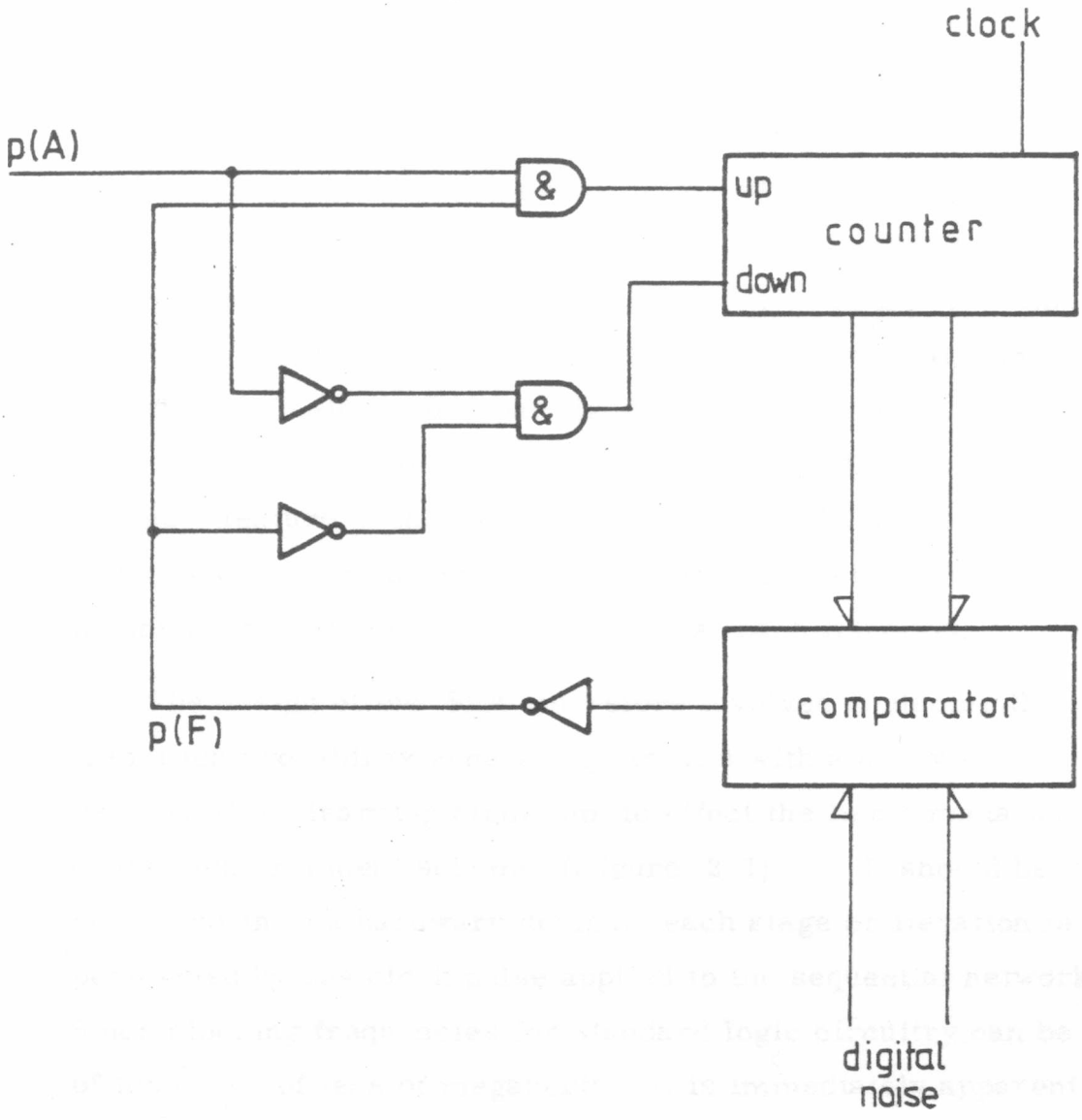


Figure 1.7 The noise ADDIE

CHAPTER 2

SYNTHESIS AND OPERATION OF A TWO-STATE SYSTEM

2.1 Initial Design Considerations

The starting point for the synthesis of a hardware system was taken to be the Markov Chain simulator described by Baxter et al.[18, 32] This is a special purpose simulator designed for high speed calculation of the n^{th} power of the transition matrix of a Markov Chain process. The circuitry consists basically of a sequential network to process signals representing the transition probabilities which are derived from stochastic comparators and steered via appropriate pulse-steering logic. The difference between this system and the design requirements for the SLA is essentially that the Markov Chain system has fixed transition probabilities, i.e., the elements of the transition matrix, whereas the intrinsic feature of the variable structure SLA is the updating of the transition (or total state) probabilities, by means of the reinforcement scheme, at each iteration.

The design of the SLA therefore involved replacing the transition probability generating circuits with a hardware version of the learning algorithm to effect the implementation of the reinforcement scheme (Figure 2.1). It should be noted that in this hardware design, each stage or iteration is performed by one clock pulse applied to the sequential network. Since clocking frequencies for standard logic circuitry can be of the order of tens of megahertz, it is immediately apparent how fast such a system can potentially operate.

2.2 Two-State System

It was decided to design initially a two-state system, in view of the simplicity of the corresponding learning algorithms. This /

This would enable valid comparisons to be made with the simulation work reported previously[20, 21] on two-state SLA's, and allow opportunity for generalisation to the r-state case when design experience had been built up. As explained earlier, the choice of P-model automaton is logical here, because of the natural affinity between a binary reward/penalty response and the operation of digital circuitry.

The sequential network in the case of a two-state system consists of a single flip-flop, with the transition diagram as shown in Figure 2.2. If a J-K flip-flop is used with the state assignment "$p_1(n) = Q$, $p_2(n) = \bar{Q}$", the input signals to the J and K lines represent $p_1(n+1)$ and $p_2(n+1)$ respectively. Therefore, the operation of the system consists of clocking the flip-flop to produce an output '1' or '0', i.e., action a_1 or a_2, with a certain probability depending on the current input condition. This output is then transformed via the algorithm circuit and reappears at the input with a revised probability to be clocked through on the next cycle. As the learning process evolves, the probability of one of the output lines being 'high' should tend towards unity. It can be seen that an inherent feature of the system is a race-around or regenerative condition. This is a consequence of the reinforcement scheme which by its very nature represents a positive feedback system.

The reinforcement scheme used was the linear reward-penalty (L_{R-P}) which is reproduced below, in total state probability form, for the two-state case:

(a) Reward (action a_1)

$$p_1(n+1) = 1 - \alpha\, p_2(n)$$

$$p_2(n+1) = \alpha\, p_2(n)$$

(b) /

(b) Penalty (action a_1)

$$p_1(n+1) = \beta\, p_1(n)$$

$$p_2(n+1) = 1 - \beta\, p_1(n)$$

Similar expressions hold for action a_2.

It is convenient to express the learning algorithm in the form of a "truth table", indicating the terms which must be generated for each combination of current action and reward/penalty response.

$a_1(n)$	$a_2(n)$	P/R	$p_1(n+1)$	$p_2(n+1)$
0	1	0	$\alpha\, p_1(n)$	$1 - \alpha\, p_1(n)$
0	1	1	$1 - \beta\, p_2(n)$	$\beta\, p_2(n)$
1	0	0	$1 - \alpha\, p_2(n)$	$\alpha\, p_2(n)$
1	0	1	$\beta\, p_1(n)$	$1 - \beta\, p_1(n)$

The various algorithm terms $\alpha\, p_1$, etc., are simply formed using the stochastic computing elements outlined in Chapter 1.

It was found subsequently during initial circuit tests that a design embodying a J - K flip-flop would not work correctly. The problem stemmed from the complementary nature of the signals on the J and K lines. As in the Markov Simulator, pulse-steering logic was used to ensure that $p_2(n) = 1 - p_1(n)$ at all times. The resulting waveforms are shown in Figure 2.3, indicating that a steady-state condition is established whereby the output is alternately set and reset at each clock pulse. A fresh design procedure was therefore adopted to exploit the feature of complementary signals while avoiding this drawback.

2.3 /

2.3 Revised Design

The approach used in the revised design was to employ a D-type flip-flop, and generate $p_1(n+1)$ alone as the input. Q and $\overline{Q}$ represent $p_1(n)$ and $p_2(n)$ as before, but $p_2(n+1)$ exists only as the implicit complement of $p_1(n+1)$. It was realised that using the current state signals to select algorithm terms as well as actually form them would lead to anomalies; for example, $1 - \alpha p_2$ would become simply 1 if action a_1 is performed because p_2 is then zero for the duration of the cycle.

The revised system was therefore designed with two dependent but separately clocked loops: one with a fast clock for the algorithm computation cycle, the other with a slower clock representing the main system cycle. The full circuit diagram, incorporating algorithm, sequential network and simulated plant, is shown in Figure 2.4.

Algorithm terms formed by the stochastic computing elements are selected according to the current action and the resulting reward/penalty signal (P/R). A standard TTL four-to-one line data selector provided a convenient and compact method for selecting the appropriate term to represent $p_1(n+1)$, with the fast feedback loop, associated with FF1, feeding the algorithm with current action probability values (p_1 and p_2) and the slow loop, associated with FF2, establishing the address for the data selector (a_1). Therefore FF2 can be identified as the principal "system flip-flop". A 10:1 ratio of clock frequencies was found in practice to give satisfactory results. The reward/penalty signal was derived from a simulated "environment" consisting simply of two penalty probability signals, c_i, selected by the appropriate action a_i. The generation of these signals is described below.

2.4 /

2.4 Noise Sources

The various noise-related constants required for the above system were derived from a common central digital noise source, consisting of a standard shift-register PRBS generator.[37, 45] A 31-bit shift register, with exclusive -OR feedback connected to bits 3 and 31 as shown in Figure 2.5, will cycle at random through every state bar "all-zeroes" before repeating. This is referred to as a maximal or m-length sequence.[46] The total number of states is $2^n - 1$, which in the case of $n = 31$ exceeds 2×10^9. Each cell of the register is thus in effect a digital noise source, producing a stochastic bit stream with a "value" of 0.5. By virtue of the clock pulse delay between each cell, the noise lines are statistically independent, which guarantees the validity of computations performed between them, and also between any signals derived from them. Also incorporated in the noise generator is a "one-shot" circuit which ensures that the "all-zeroes" condition cannot occur at switch-on and prevent the sequence from starting.

A wide range of factors based on fractions of $\frac{1}{2}$ or their complements can be generated by combining noise lines via arrays of AND-gate multipliers, together with inverters. The arrangement adopted here is shown in Figure 2.6.

As reported by Narendra et al, the degree of expediency of the L_{R-P}, and other reinforcement schemes, depends on the relative values of the step-size factors α and β. The reward-penalty ratio $(1-\alpha)/(1-\beta)$, denoted by γ, is thus an important element in the classification, and performance, of individual schemes. The noise sources described above enabled a variety of c_i and γ factors to be selected, so that a useful range of experiments with the L_{R-P} scheme could be carried out.

2.5 /

2.5 Output Interface

Some form of output interface circuit was required as a means of observing the behaviour of the SLA. It was decided that the most suitable way of studying the learning characteristics was to use a storage oscilloscope to display individual "learning curves". The learning curve is a plot of action probability against time from the commencement of the learning period. Typical results obtained in this way were then photographed and reproduced in a format similar to that of previously published results.[20, 21] The system output, which is a stochastic pulse train containing frequency components of the order of the clock frequency of the main system flip-flop, was passed through a filter to produce an analogue measure of the action probability value.

The filter circuit used for this particular stochastic-to-analogue conversion process was the well-proven 2^{nd} order Butterworth type, together with a level-shifter and calibration stage to convert the TTL signal to a convenient 0 - 10 volts output swing. The circuit is shown in Figure 2.7. The choice of time constant or cut-off frequency for the output interface clearly influences the observed response time, and a compromise exists between learning time and variance in the steady-state for a given clock frequency. The values shown gave reasonably low variance at a clock rate of 1 MHz, as demonstrated by the results detailed below, while permitting a response time of approximately 5 ms.

The complete two-state system, including noise sources and simulated environment, was assembled on two small circuit cards and fitted in a cabinet with a front-panel "patching" facility for the selection of c_i, α and β. A photograph of this unit is reproduced as Figure 2.8.

2.6 /

2.6 Experimental Results

For the first experiments, the simulated environment response was set up with $c_1 = 0.75$ and $c_2 = 0.25$, while γ was varied from 1 to 64 using combinations of the available noise sources. The system master clock was set at 1 MHz, resulting in a clock frequency of 100 kHz for the main system flip-flop.

The family of learning curves illustrated in Figure 2.9 clearly shows how the degree of expediency increases as γ increases. In each case, the system flip-flop was preset initially to a_1, i.e., the "wrong" action, and the output subsequently converged to a steady-state condition with a high probability of choosing a_2, the action with the lower penalty probability. When $\gamma = 1$, the system converged to a level corresponding approximately to the reward probability for the better action. This is to be expected in a situation where reward and penalty factors are of equal magnitude.

The overall characteristics of the system are well summarised by the results presented in Figure 2.10. These curves show how the SLA can lock-on to whichever action carries the lower penalty probability, from either initial state, using in this case a scheme with $\gamma = 8$.

The influence of environment characteristics on learning behaviour is illustrated in Figure 2.11. This shows the results from two separate trials, again using the "$\gamma = 8$" scheme, in which the effect of changing the penalty probabilities was investigated. While c_1 was fixed at 0.75, c_2 was increased from 0.25 to 0.5. The degree of expediency, as represented by the steady state value of p_2, is clearly reduced for the latter case.

A significant feature of all these results is that learning times /

times of the order of milliseconds were consistently recorded. Moreover, an investigation of the actual flip-flop output showed a very rapid transition to the steady-state pulsing situation, so the learning times indicated here are in fact dominated by the output filter time constant. A learning period of 5 ms with a main system clock frequency of 100 kHz implies that approximately 500 cycles are used to reach steady-state conditions. This order of magnitude of "stage number" is in accordance with software simulation results reported previously.[20, 21]

2.7 Parameter Optimisation

An application area of particular importance for the SLA is the multimodal parameter optimisation problem.[47-51] The general problem is that of identifying the extremum of a noisy multimodal performance surface, which can be described by a function $g(\underline{a}, z)$. Here, $\underline{a}$ denotes the vector of system parameters, and z represents the superimposed noise amplitude, i.e., $g(a, z) = f(a) + z$ (see Figure 2.12).

Such problems are frequently insoluble by existing techniques, because of either a lack of sufficient a priori information concerning plant structure and dynamics, or mathematical difficulties involving computation time and problem complexity. In addition, plant and controller variables are of course subject to random disturbances, the intrusion of noise making reliable prediction or measurement impossible. Even in such cases where a solution by conventional techniques[52, 53] can be envisaged, this will tend to result in convergence to some "local" optimum unless prior knowledge can somehow be acquired to enable the selection of a suitable starting value for /

for the search.

The use of the SLA avoids all these problems, because of the inherently random nature of its search in the parameter space. Automaton actions are assigned, in some arbitrary fashion, to represent respective values of system parameter, so that each point on the P I surface has a non-zero probability of being selected during the initial learning period, regardless of contour irregularities or even discontinuities. In addition, it is not necessary to have any knowledge of the distribution of noise on the surface. These features give the automaton the ability to locate the global optimum at all times, which will of course yield the lowest probability of received penalty. The SLA is, in effect, "altitude" rather than "gradient" sensitive.

This particular optimisation problem has received much attention in previous simulation studies,[47 - 50, 54, 55] and clearly represents an important area of investigation with the hardware system.

2.8 Learning Controller

The general configuration of a learning controller is shown in Figure 2.13. A key feature of this system is the "evaluation section", which represents the interface between the environment and the SLA. The performance index (P I) measure has to be translated to a reward/penalty response, which must have a binary form if the P-model is to be used. Narendra[49] described how this translation problem could be side-stepped by employing an S-model automaton to handle a suitably normalised measure of the P I as a direct input. Also, an adaptive method was proposed to cater for the situation of unknown P I bounds, which would often be the case in practice.

Since /

Since the hardware design effort has been concentrated on the P-model automaton, for reasons already discussed, the problem of implementing a P I evaluator suitable for this model was investigated. For the sake of simplicity, a non-adaptive system was initially considered, and the result was the configuration shown in Figure 2.14. This consists essentially of a standard digital-to-stochastic interface (see Section 1.9) which compares the digitised input with a random noise source. The output of the comparator is thus a stochastic bit stream whose value i.e., the time averaged ratio of 1's to 0's, represents the value of the digital input as a fraction of full scale.

The P I is therefore presented as a digital value, assumed to be normalised, feeding directly to the comparator. The resulting output pulse train represents an instantaneous measure of $g(a_i)$ and hence also of c_i. This will reach an absolute minimum value, c_m, when the search locates $g(a_m)$ corresponding to action a_m, which is the global minimum. If a flip-flop is placed on the comparator output as shown and clocked at some instant, the situation is analogous to tossing a weighted coin, in that the probability of a '1' at the output depends on the average ratio of 1's to 0's in the input pulse train. Using this deceptively simple configuration, it is therefore possible to perform a truly global search of the P I surface, and obtain a reward/penalty response in binary form as required for the P-model.

The basic system as described assumes that the bounds of the P I are known, to avoid exceeding the capacity of the comparator. However, with the P I presented in digital form, it should be possible to design an adaptive interface employing some form of "autoranging" technique, as used in standard multirange DVM's for example, to map any /

any incoming signal into the range of the comparator. This approach would be essentially similar to Narendra's method of handling unknown P I boundary values, referred to above.

2.9 Plant Simulator

The ability of the prototype flip-flop SLA to perform with a simulated "noisy" plant was then investigated. In the case of a two-state system, the performance "surface" consists of merely two discrete points, with superimposed noise causing random fluctuations in their values.

Translating this into hardware terms, the two "P I" points were represented by two numbers stored in binary counters. The counter contents were then converted to c_i signals using the above comparator method. The effect of superimposed noise was then simulated by allowing the counters to undergo a "random walk" between set limits. This was achieved by feeding the up/down lines of the counters with noise signals from the PRBS generator, resulting in an equal probability of counting up or down. The chosen end-points for the counters were detected by combinational logic feeding back to inhibit the count as necessary. A latch was placed between each counter and its comparator to allow the counter to cover its full permitted range between samples. At each sampling instant, the resulting c_i value could be anywhere within these set limits. Two clock frequencies were therefore required: a fast clock for the counter, and a slower clock for the latch. The system was designed with 4-bit counters, giving the P I representation shown in Figure 2.15.

The circuit configuration of the plant simulator is shown in Figure 2.16, and the "end-points" logic in Figure 2.17.

In /

In practice, it was found convenient to use the two clock lines from the SLA to run the plant. A simple change in the end-point detection circuit produced the alteration in the noise boundary shown by the dotted line in Figure 2.15. This enabled the ability of the SLA to deal with the case of overlapping noise with asymmetrical distribution on the P I surface to be investigated. The SLA should be capable of locking on to the correct action provided that, on average, one of the c_i signals is found to be lower, despite the fluctuations caused by superimposed noise.

2.10 Simulator Results

The assembled plant simulator circuit was incorporated within the existing flip-flop SLA box (Figure 2.8), and tests were made using an L_{R-P} reinforcement scheme with $\gamma = 8$. Learning curves were obtained as before with both possible assignments of c_i to a_i, favouring first one action, then the other. The result is shown in Figure 2.18. The "low-noise" curves illustrate the ability of the SLA to identify the correct action despite the noise on the c_i.

The noise boundary was then altered, as shown in Figure 2.15, to increase the variation on one of the c_i. There is still, on average, a difference between them, but a greater deal of overlap exists, making discrimination much more difficult. The experiment was then repeated, and the resulting "high noise" curves are again shown in Figure 2.18, superimposed for the purposes of comparison with the first result. It is significant that, although the quality of convergence obtained is less expedient, as would be expected from previous results with similar, fixed values of c_i, the SLA is nevertheless able to identify the correct action.

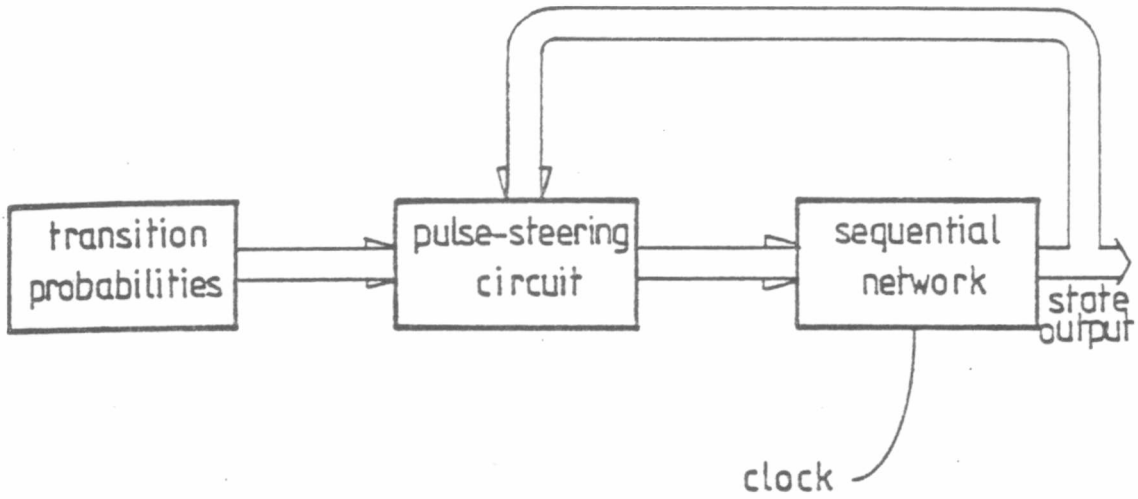
2.11 /

2.11 Conclusions

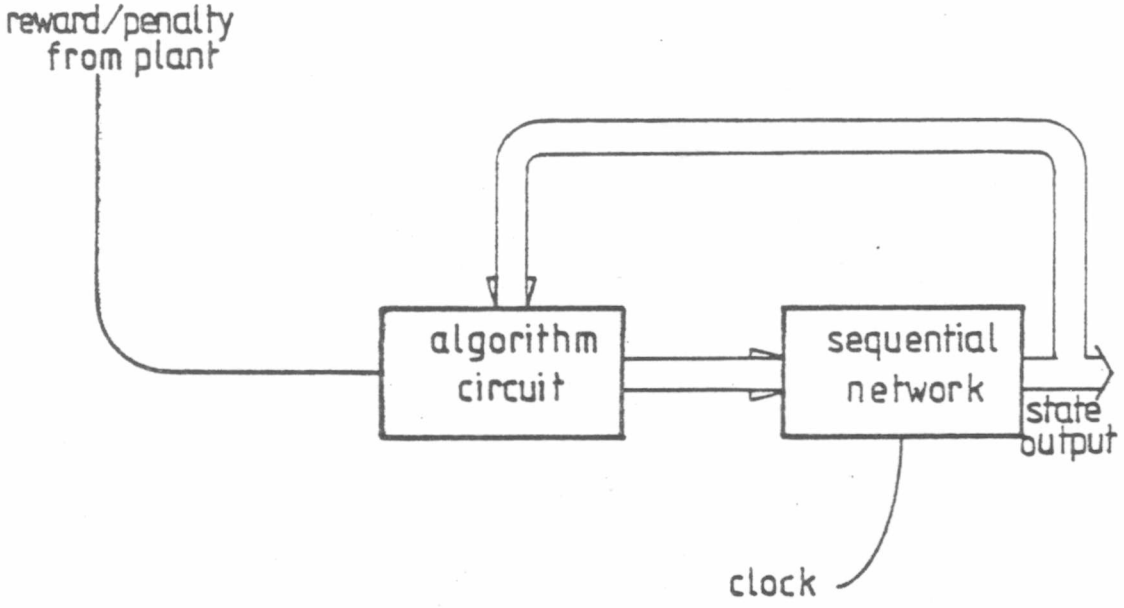
The above results clearly demonstrate the potential of the SLA as a means of achieving optimisation even under extreme conditions of intrinsic noise, while the fast learning times obtained indicate that practical, on-line operation is feasible with the hardware system.

Although this simple prototype did provide satisfactory simulation of the expedient L_{R-P} scheme, its usefulness was limited in particular by its inability to implement ϵ-optimal schemes such as the linear reward-inaction, or L_{R-I} scheme. Inspection of the circuit shows that setting $\beta = 1$ (i.e., infinite γ) would simply cause the system flip-flop to remain in whichever state was initially selected. This was easily demonstrated in practice.

These considerations, together with the need to develop larger systems, led to the design of a more sophisticated SLA circuit, capable of operating with a comprehensive range of reinforcement schemes. This development is described in the following chapter.



[a] Markov Simulator



[b] Stochastic learning automaton

Figure 2.1 Evolution of SLA design

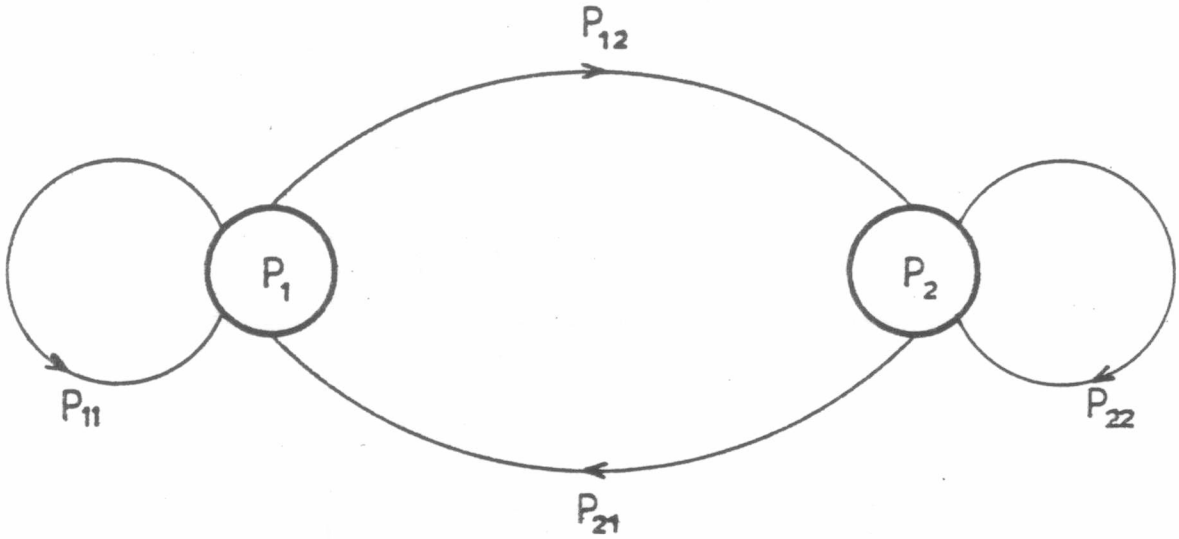


Figure 2.2 Two-state system transition diagram

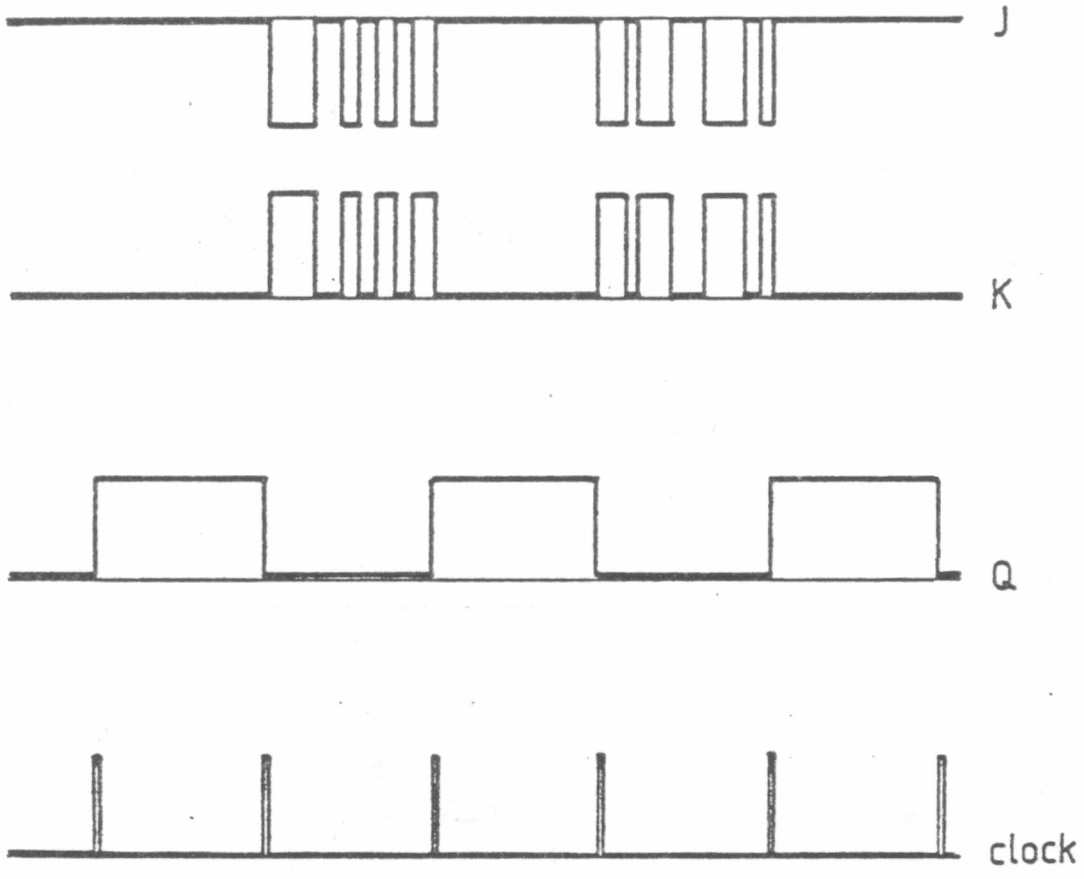


Figure 2.3 J-K system waveforms

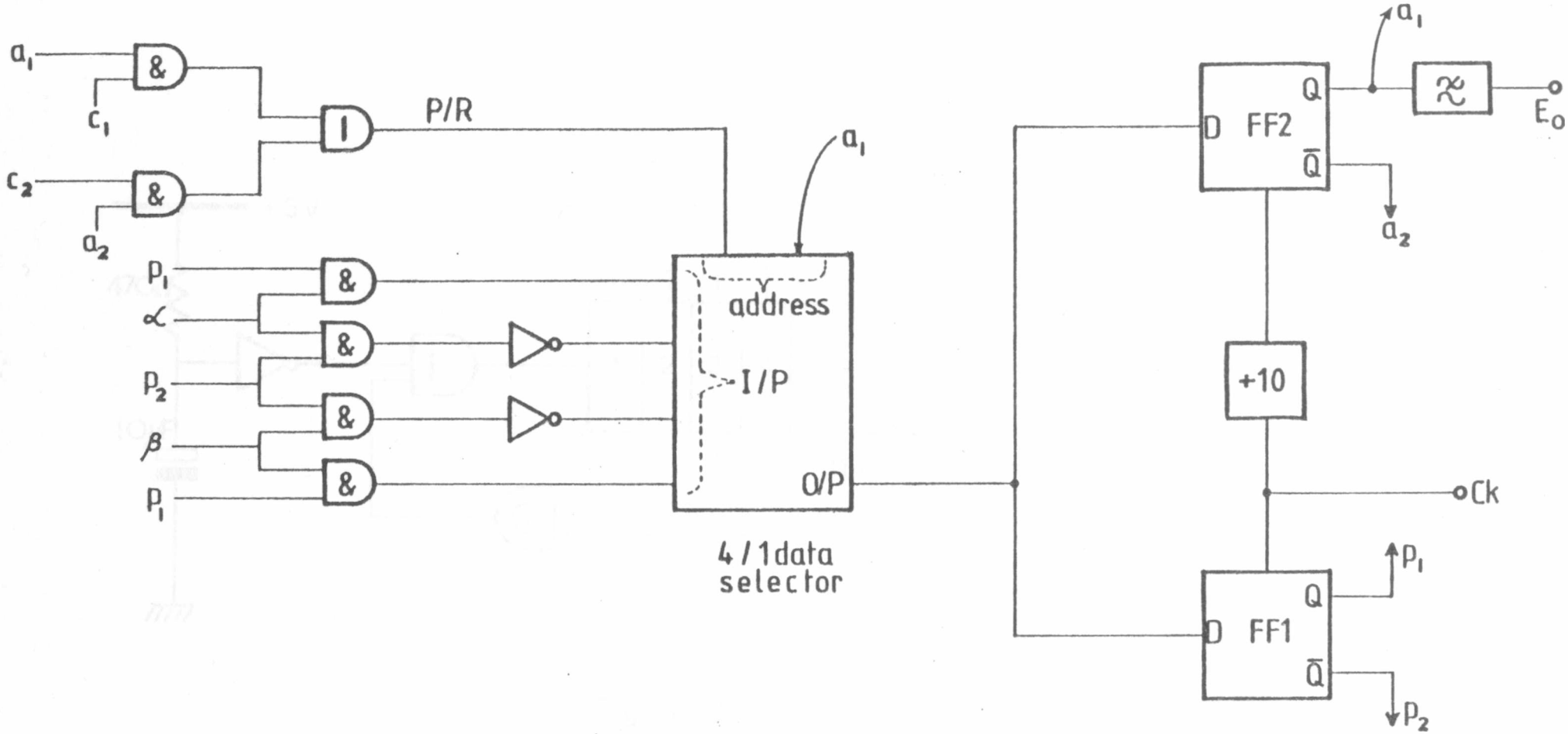


Figure 2·4 Flip-flop SLA circuit

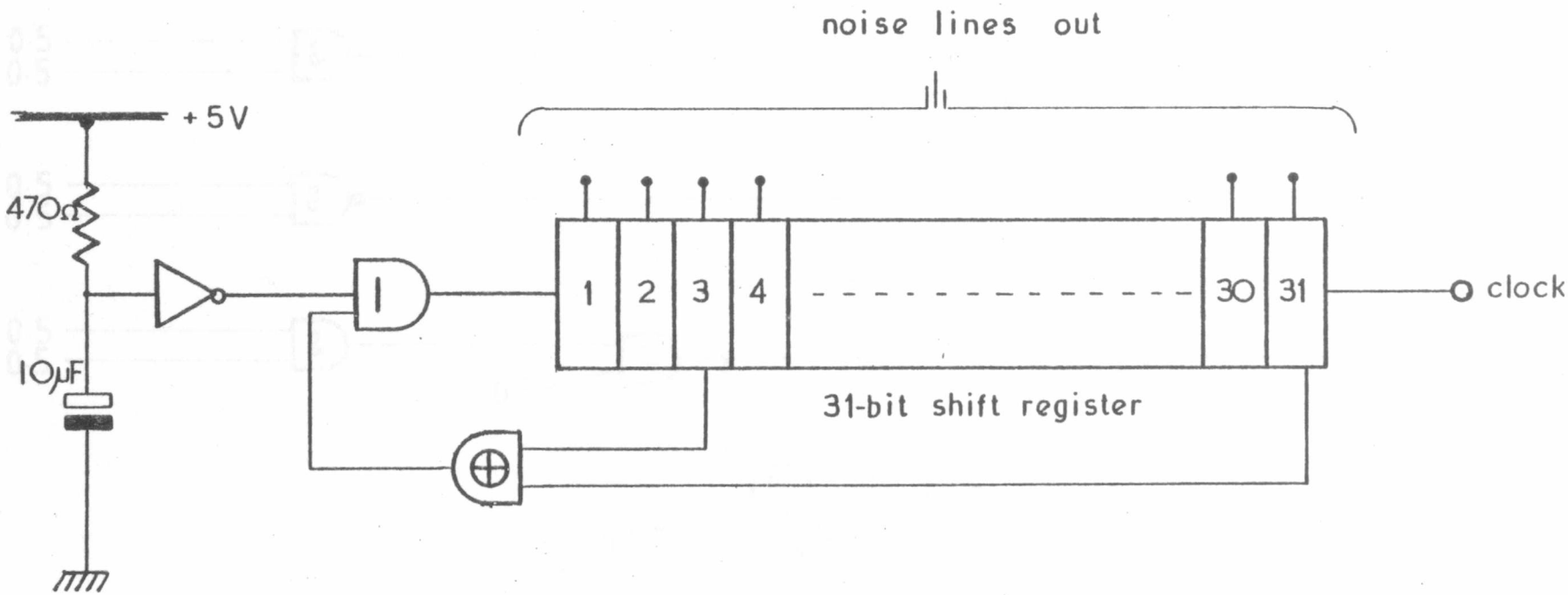


Figure 2.5 PRBS generator

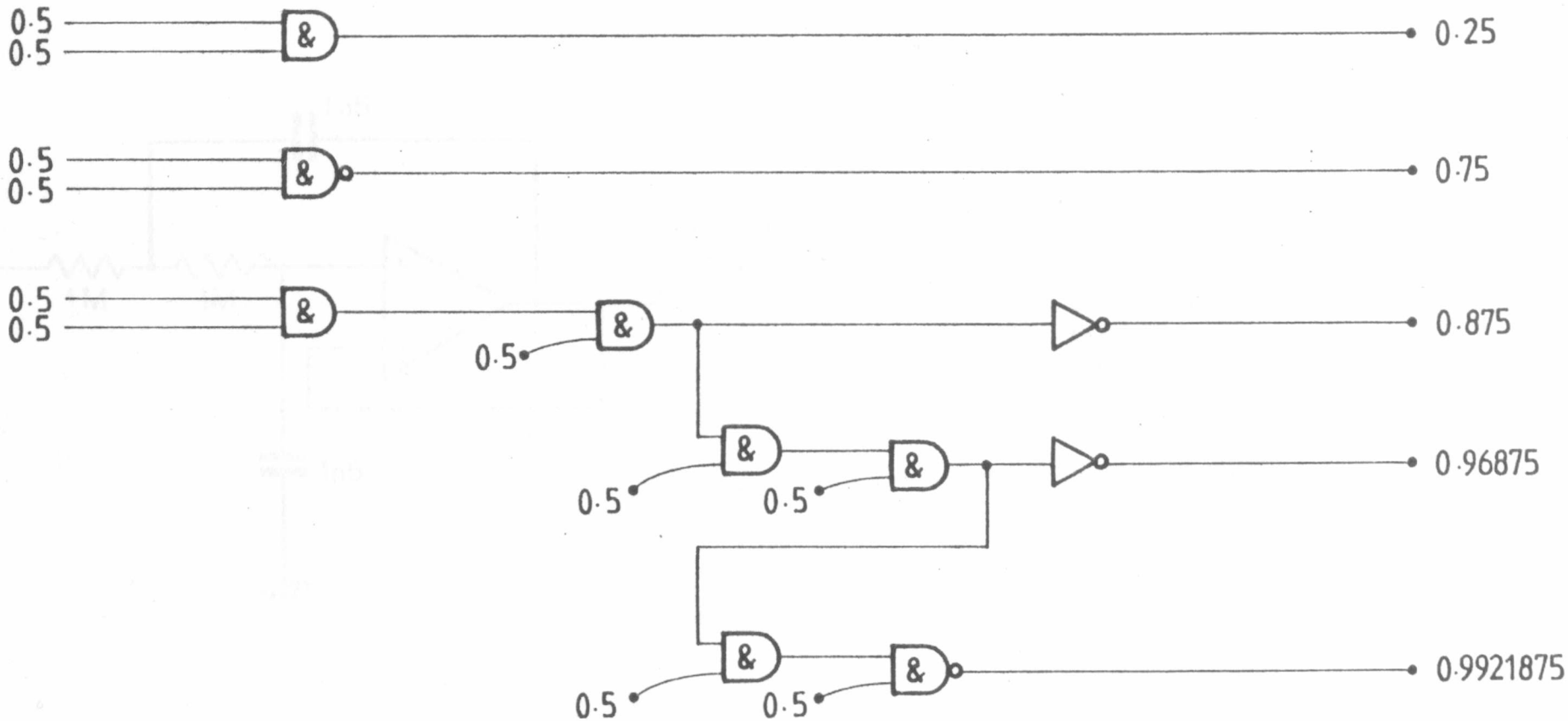


Figure 2·6 Generation of stochastic variables

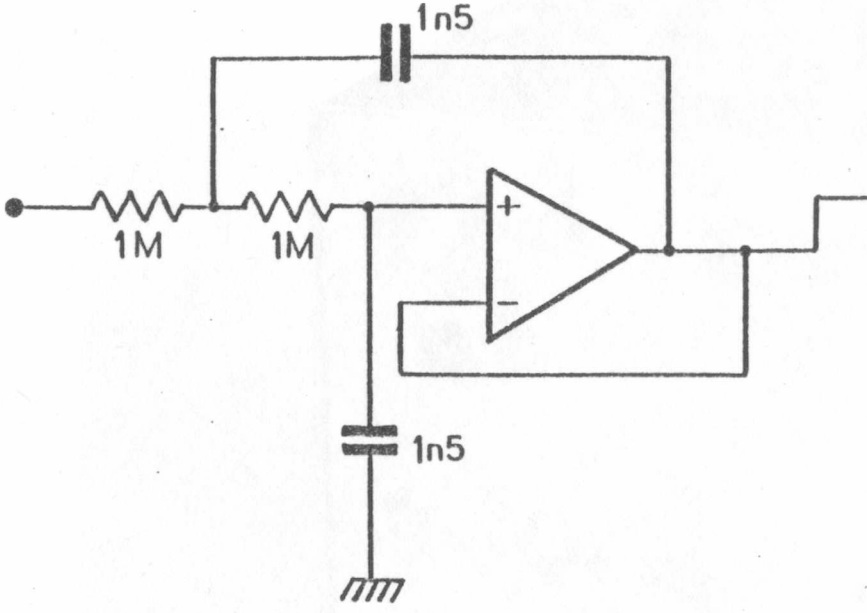
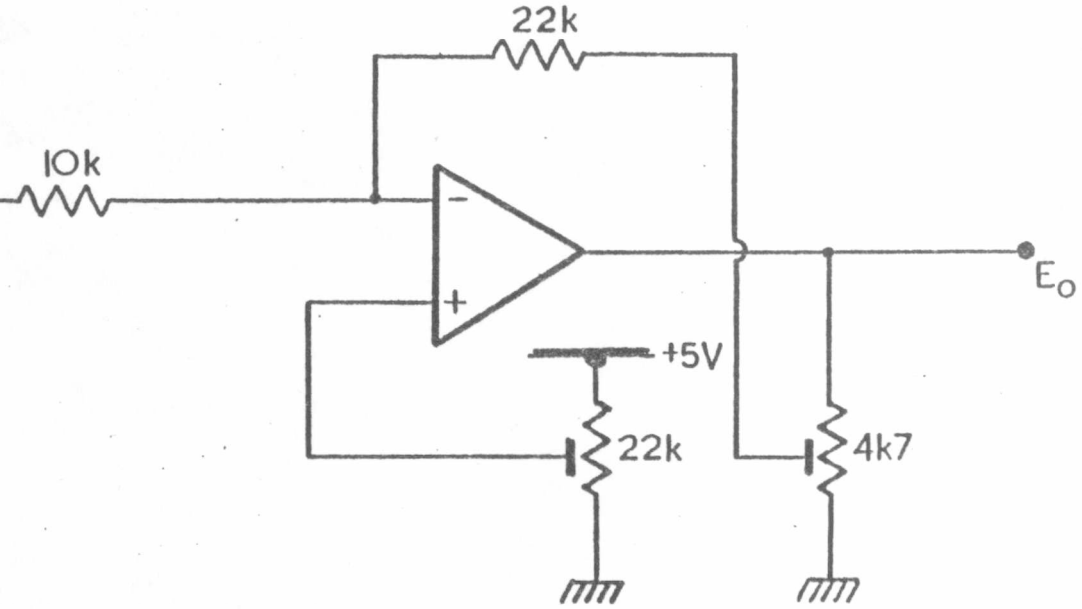


Figure 2.7



SLA output interface

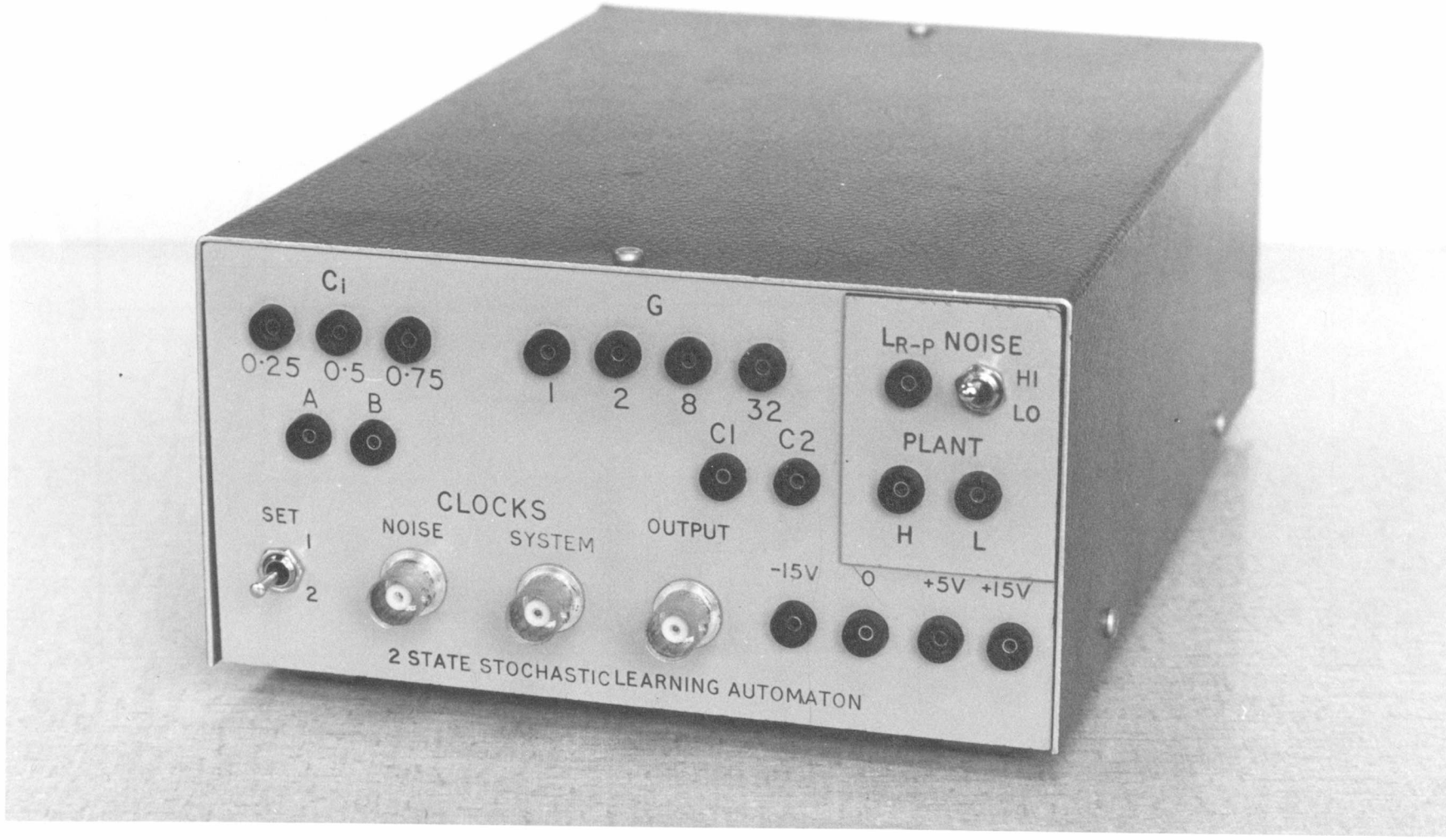


Figure 2·8 Flip-flop SLA

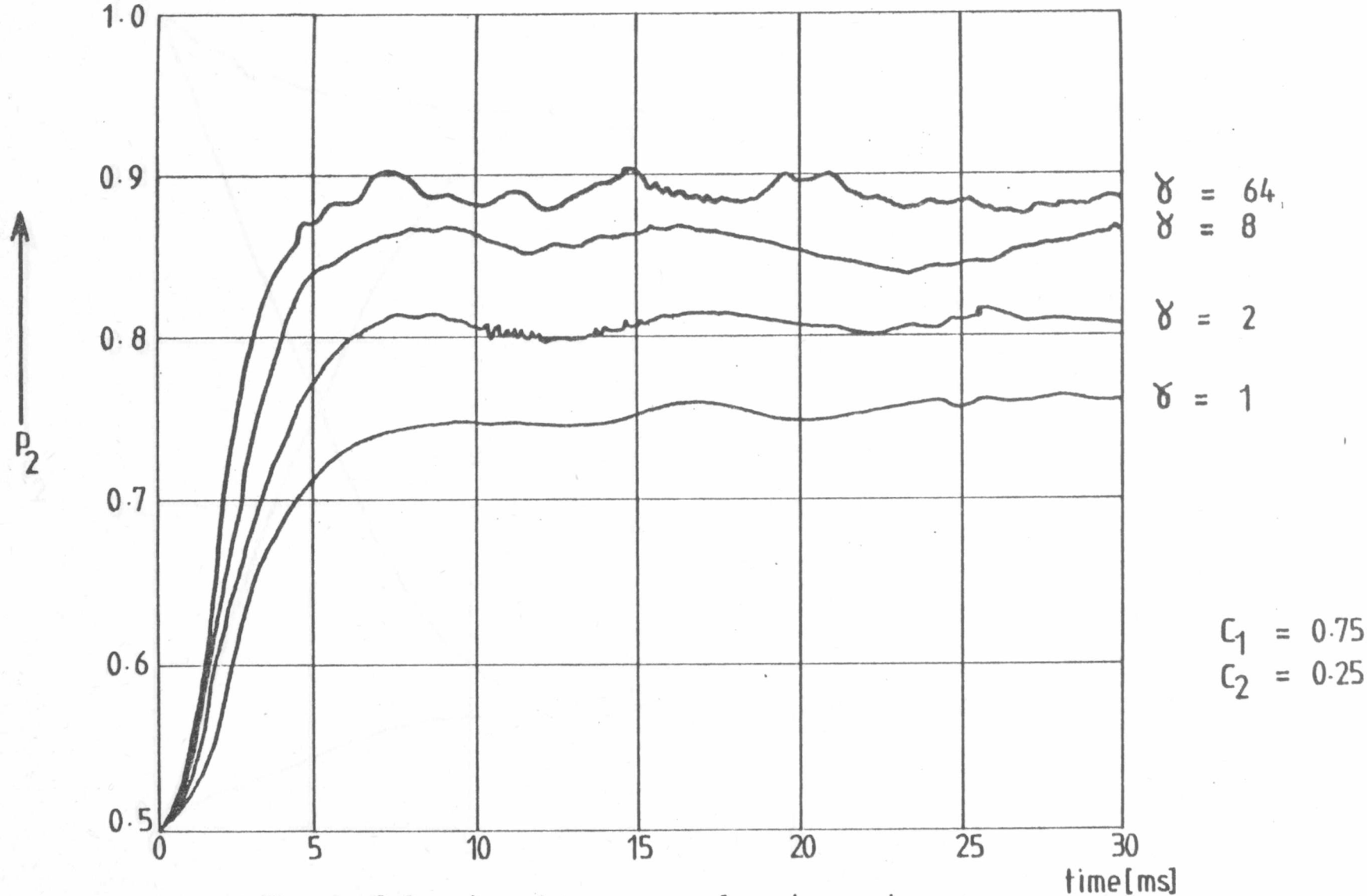


Figure 2.9 Learning curves for L_{R-P} scheme

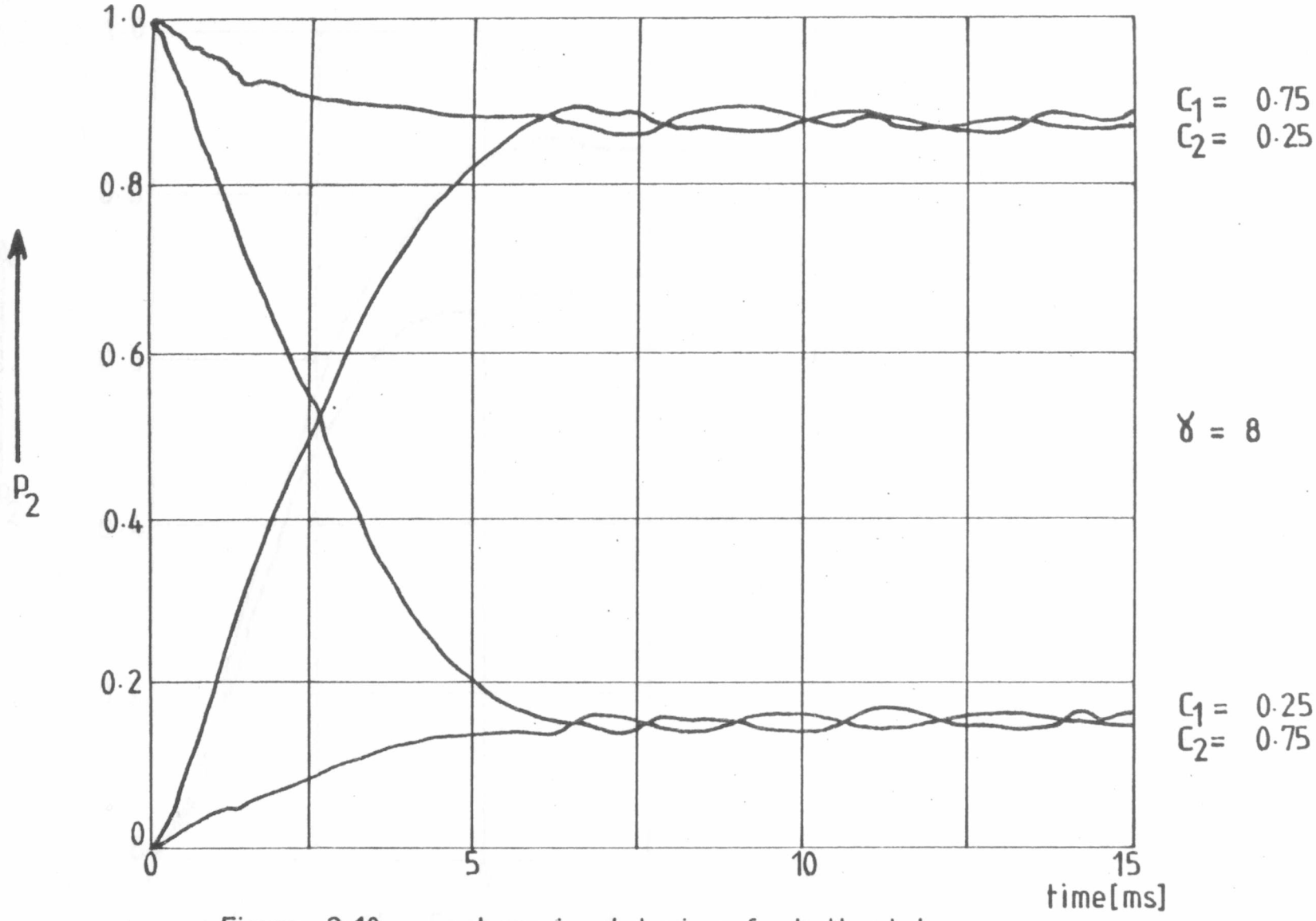


Figure 2·10 Learning behaviour for both states

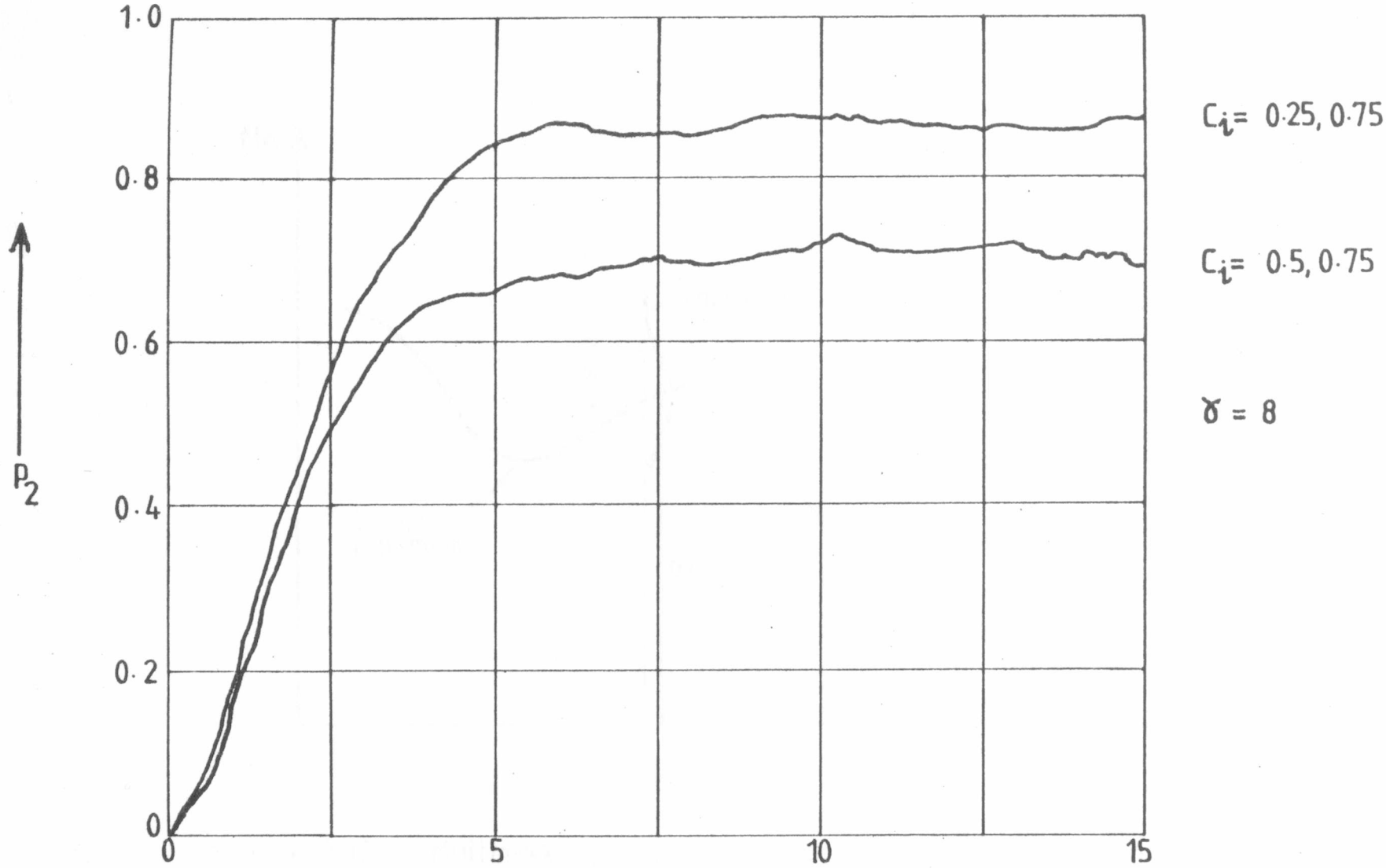


Figure 2·11 Comparison of expediency for different c_i

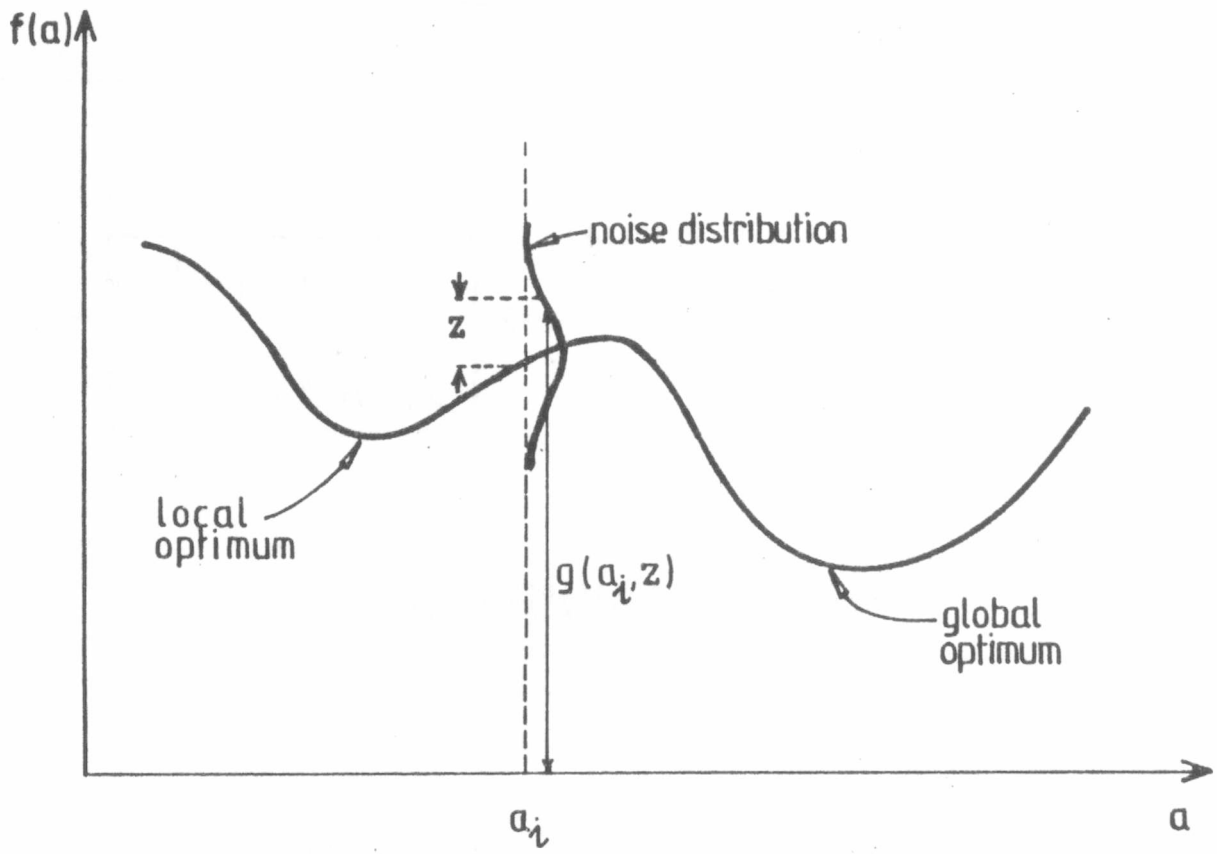


Figure 2.12 Multimodal performance index with superimposed noise

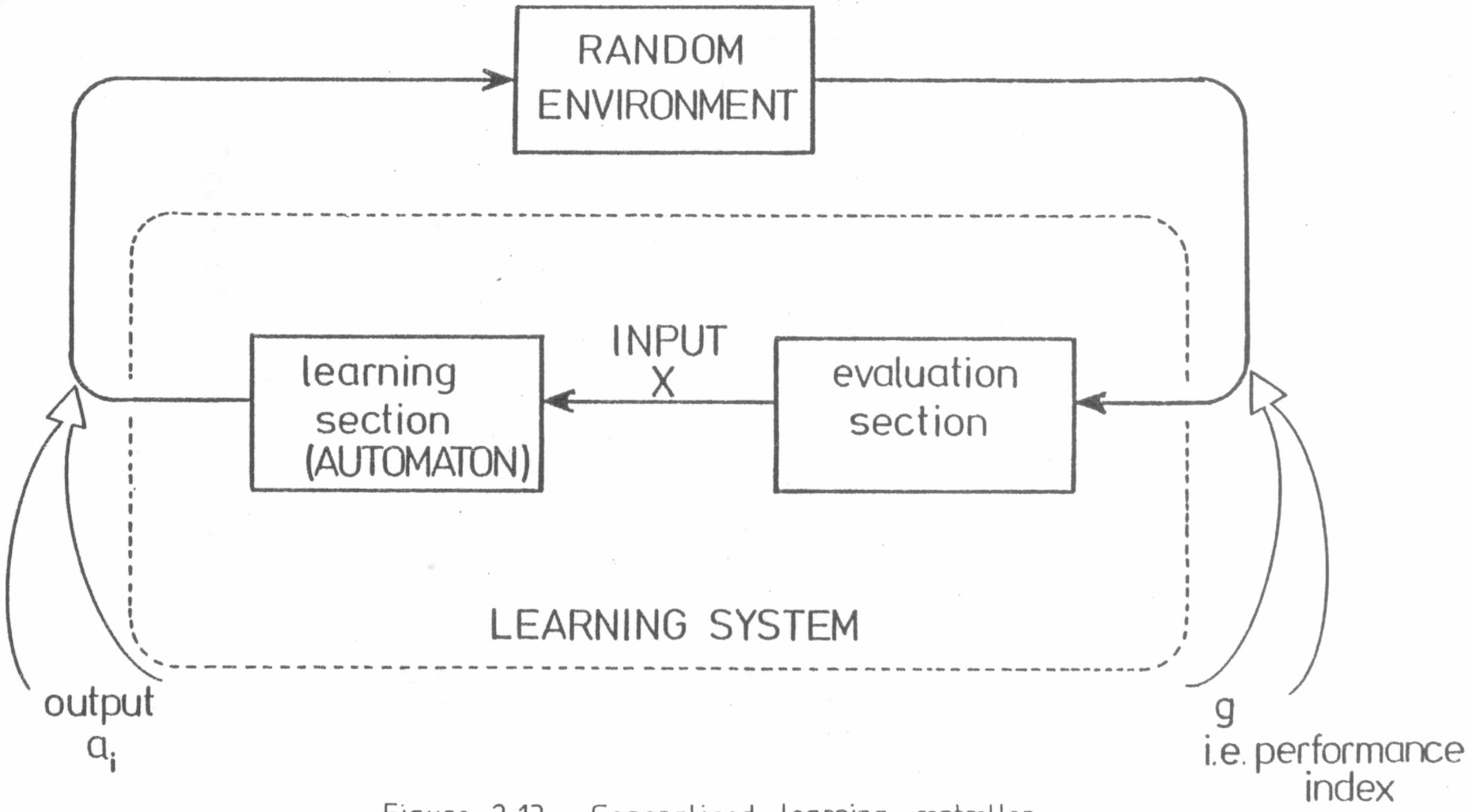


Figure 2·13 Generalised learning controller

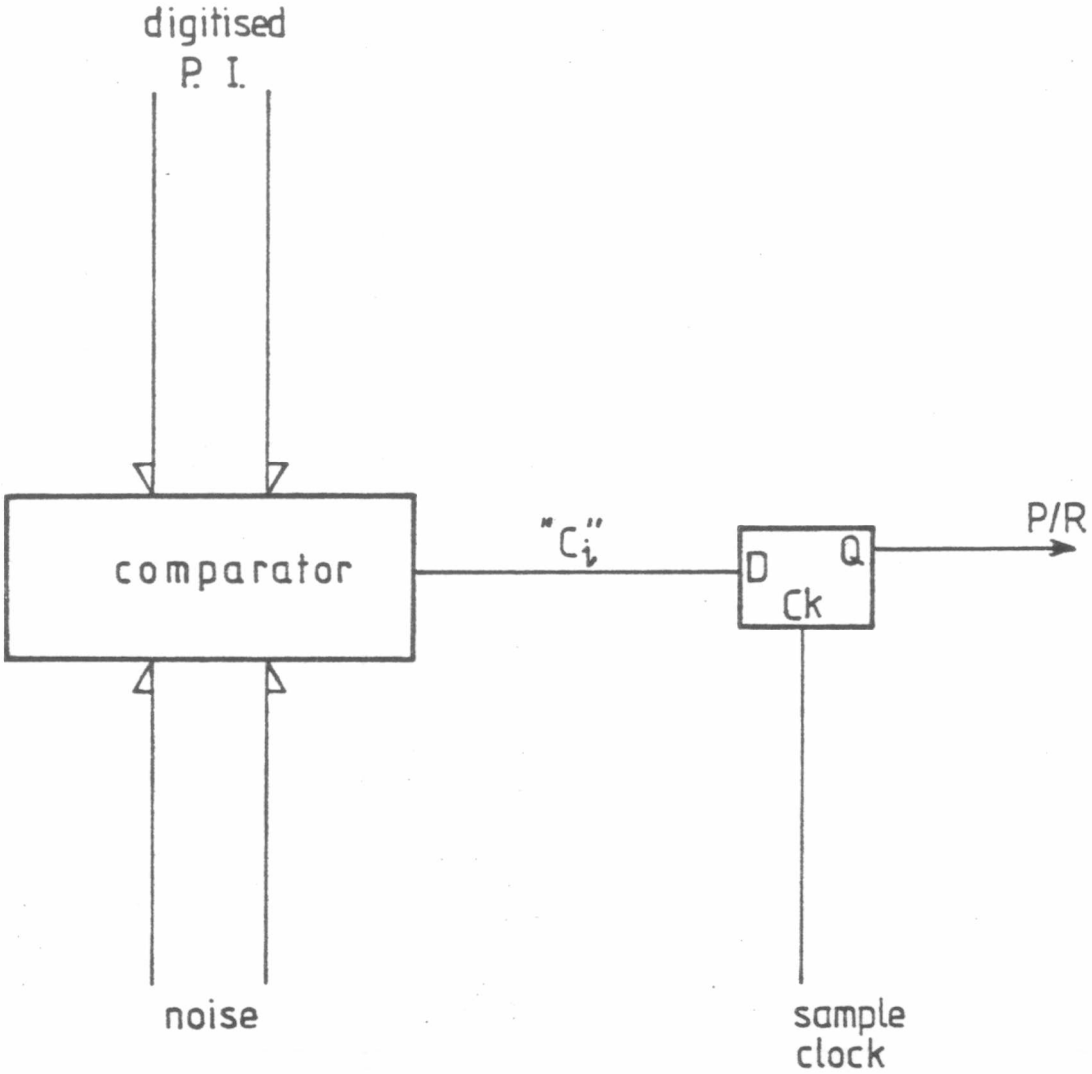


Figure 2.14 Plant - SLA interface

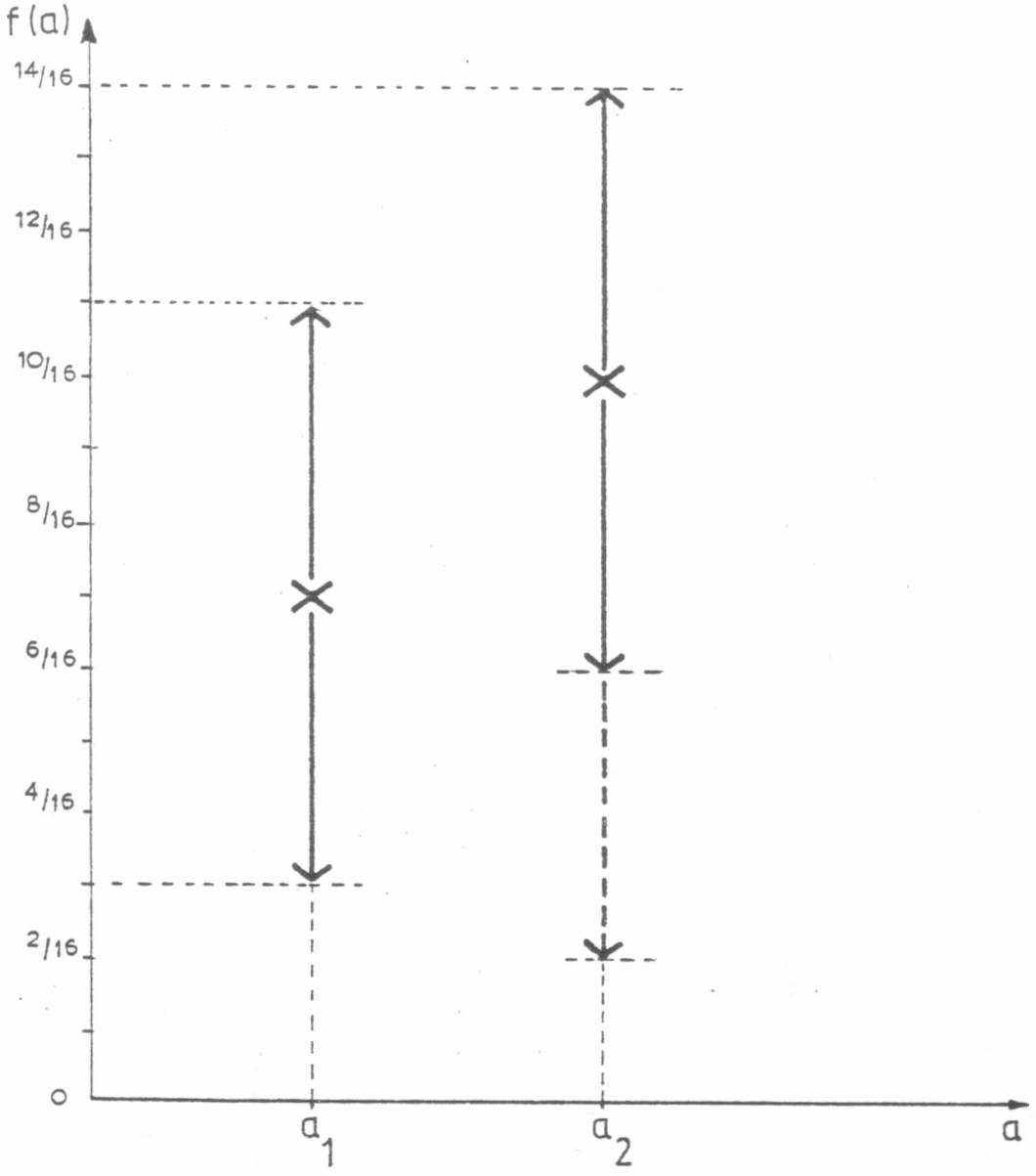


Figure 2.15 Simulated two-state 'noisy' environment

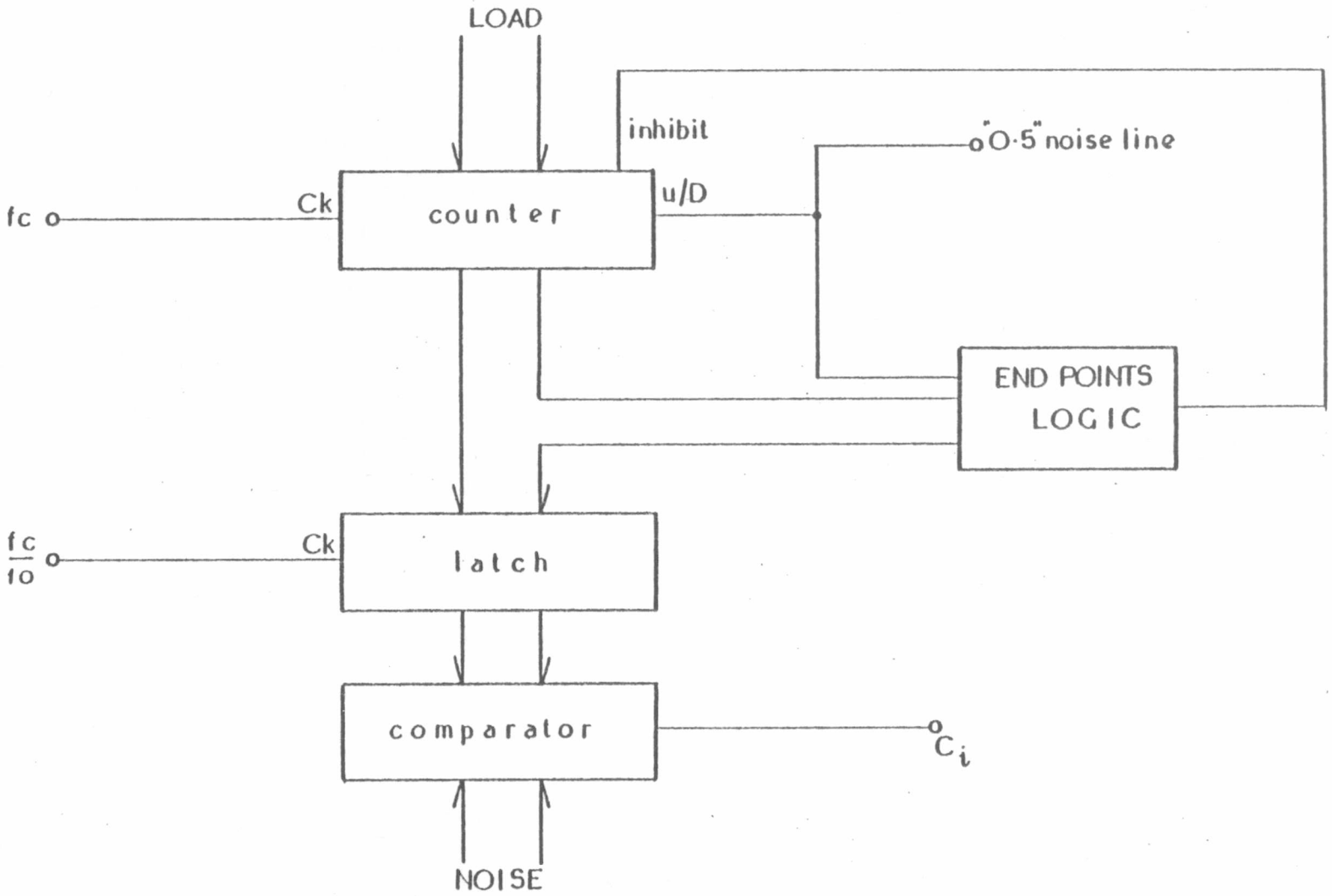
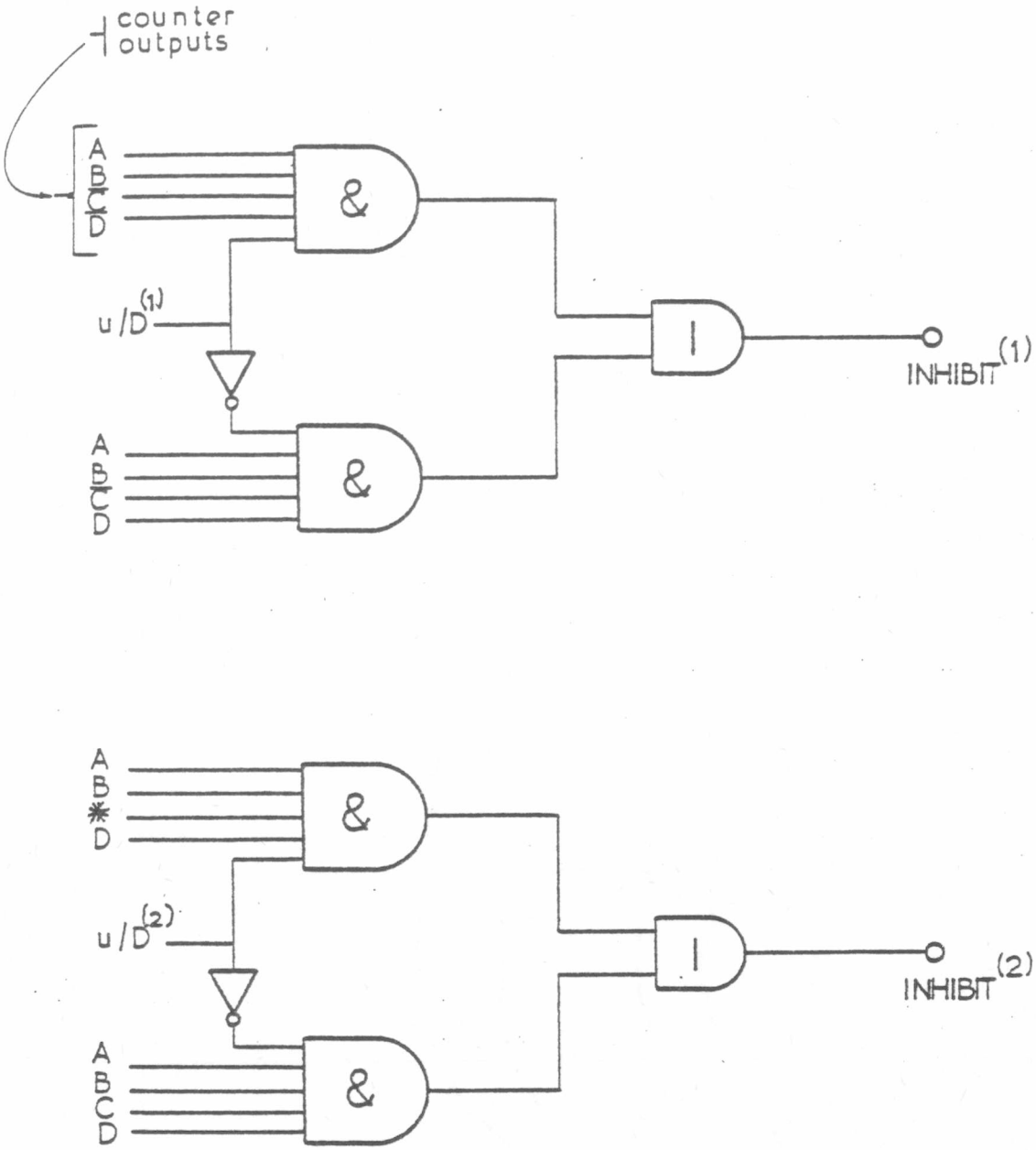


Figure 2·16 Plant simulator circuit



* C for "low noise"
$\bar{C}$ for "high noise"

Figure 2.17 End-points logic

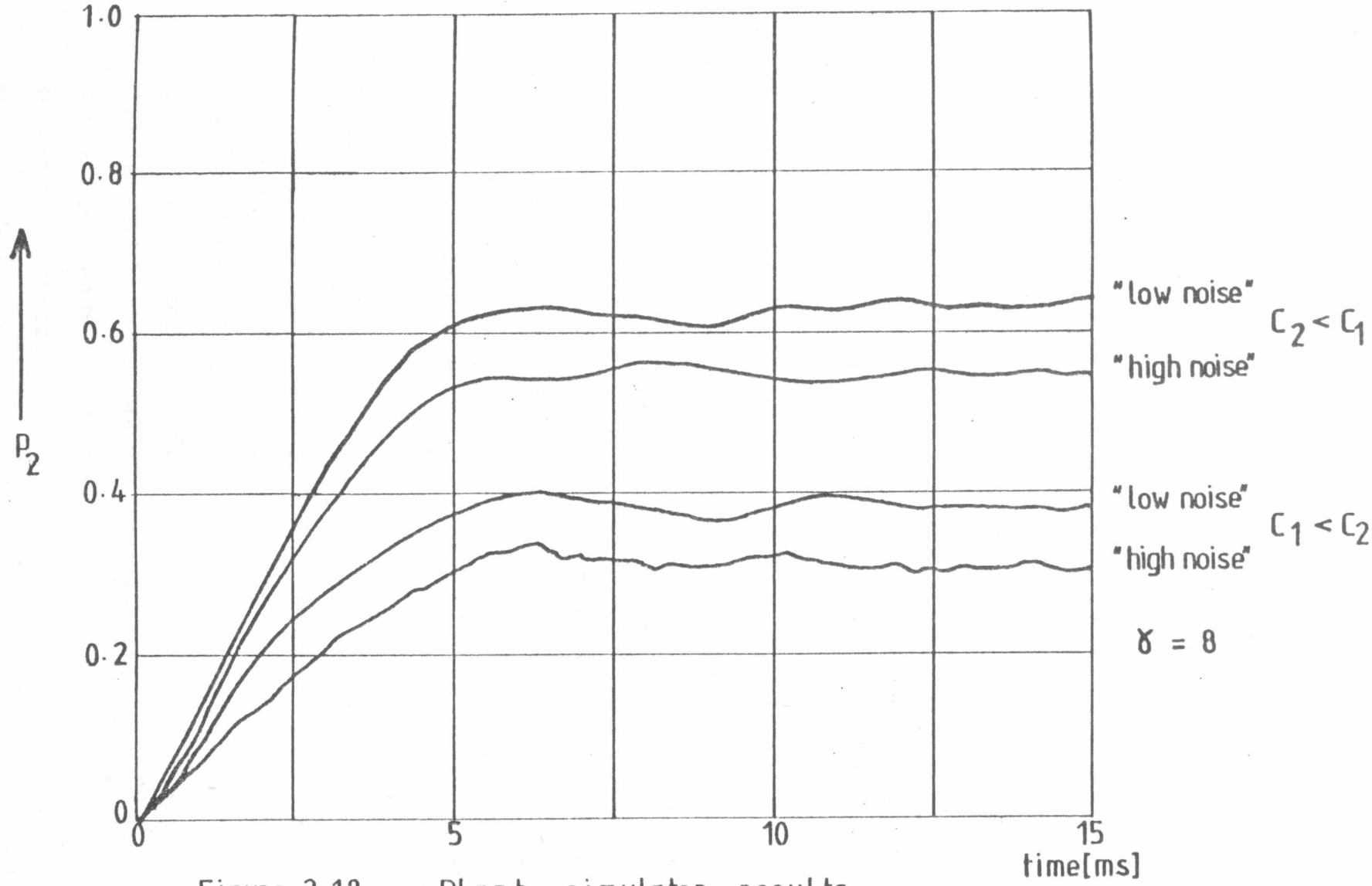


Figure 2·18 Plant simulator results

CHAPTER 3
DEVELOPMENT OF AN ADDIE SLA

3.1 Design Requirements

As was stated at the end of Chapter 2, the simple flip-flop SLA was very restricted in its range of application, in particular by its inability to operate with superior reinforcement schemes such as the ϵ-optimal L_{R-I} scheme. Attention was therefore focussed on improving the hardware design, with a view also to the intended development of large state order systems.

On considering the reinforcement algorithms in their general, r-state form, it is clear that there must be a memory capability built into the automaton structure, so that some priority of state or action probabilities is established during the learning period. Unless this were so, the past experience of the SLA would be wiped out after each system cycle (or clock pulse). This consideration led to the idea of representing the total state probabilities of an automaton, not by the probability of a flip-flop being in a certain state at the occurrence of a clock pulse, but by a number stored in a counter. Conversion from this representation to a single-line bit stream can be easily achieved by the noise comparator method described earlier.

3.2 Basic Configuration

The basic configuration of an r-state SLA using the counter implementation is shown in Figure 3.1. Associated with each counter is a flip-flop, and it should be noted that again pulse-steering logic is required to ensure that, at each sampling instant (system cycle) only one of the D-types can be triggered to indicate the state of the automaton at that particular stage. It is immediately apparent /

apparent that a large state order system, say $r > 10$, calls for a considerable amount of hardware and interconnection, a point which will be raised again later.

The implementation of the algorithm involves operating on the contents of the counters as determined by the reinforcement scheme. Although parallel computations offer the attraction of the highest operating speed, they suffer the disadvantage of considerable complications in terms of circuit design and overall system timing. It was therefore decided to apply the reinforcement scheme in serial form, operating on the count-up and count-down lines of the counter. The resulting configuration began to bear a strong resemblance to the noise ADDIE (adaptive digital logic element) of Figure 1.7, which is the standard stochastic-to-digital interface element in digital stochastic computing as described earlier.

In view of this, consideration was given to integrating the circuitry for the automaton and the reinforcement scheme in the form of an ADDIE structure. The result was the evolution of a new design for a hardware learning automaton,[56] which is referred to accordingly as the ADDIE SLA, in order to distinguish it from the earlier flip-flop SLA. It should be noted that the use of ADDIE structures has also been proposed for implementing the related "two-armed bandit" controller.[57, 58]

3.3 Design of the ADDIE SLA

As before, a two-state system was considered initially, for the sake of simplicity. It follows that results obtained can be extended to the general r-state case. Calling on the design philosophy of the flip-flop SLA, a further simplification can be made here in that only one counter is actually /

actually required, representing one state probability explicitly; the other probability is simply the complement The counter-comparator arrangement of Figure 3.1 is configured as a true ADDIE, measuring the updated state probability from the algorithm circuit. The algorithm therefore retains a single input, single output format, and can be designed using exactly the same principles as outlined previously in Chapter 2.

The current state probability value must be stored for the duration of each cycle to maintain a constant input signal $p_i(n)$ for the algorithm circuit while the ADDIE converges to an updated value $p_i(n+1)$. This task is performed by a latch which is loaded with the counter contents at the start of each cycle, and is itself connected to a noise comparator to generate $p_i(n)$ as a bit stream. The full circuit configuration is shown in Figure 3.2.

The output of the ADDIE is sampled by a D-type flip-flop which represents the current state at each stage and thus corresponds to FF2 of the flip-flop SLA (Figure 2.4). The environment is simulated, as before, by the selection of one of two locally-generated c_i signals. This plant simulator also incorporates a flip-flop which samples the selected c_i signal to generate a single binary reward/penalty response. As before, this has the function, along with the system flip-flop output, of providing the address for the algorithm circuit data selector, which is then fixed for the duration of one system cycle. The clock waveform illustrated on the diagram allows a brief setting-up period for this address, followed by a much longer adjustment or learning period.

3.4 /

3.4 Testing and Development

The two-state ADDIE SLA was implemented in 8-bit form, although the ADDIE's, integrators etc. reported elsewhere[41, 42] for applications in digital stochastic computing have been, in general, 12-bit devices. It was felt that an 8-bit system would have sufficient resolution for the purpose, and would also have the advantage of extra speed, since the duration of each iteration, apart from any considerations of environment response time, is dictated primarily by the time taken for the ADDIE counter to adjust to its "new" state. The output filter (stochastic-to-analogue convertor) and noise sources used were similar to those developed for the flip-flop SLA. The 8-bit noise inputs required for the ADDIE and latch comparators were tapped off an additional PRBS generator.

It was found during initial testing that the system exhibited a marked asymmetry in its characteristics. Considerably higher variance was noted when the ADDIE approached the state corresponding to "all-ones". This behaviour was traced subsequently to the basic characteristics of the comparator. If the input from the counter is "all-zeroes", then this cannot of course exceed the value of the noise input, and hence the "A > B" output will always be low. However, if the input is all-ones, the comparator output will not always be high, as it should be, because whenever the noise input is all-ones, the A > B condition no longer holds, and the comparator gives an erroneous low output.

The solution adopted initially was to detect the all-ones condition in the noise lines and set the most significant bit to zero under these circumstances using the circuitry shown in Figure 3.3. This was felt to be a reasonable compromise, since the resulting slight distortion produced in the probability distribution of the noise would occur at 0.5, /

0.5, which should not destroy the symmetry and hence affect adversely the performance at either extreme.

However, a more serious problem presented itself when the dynamic behaviour of the SLA was studied in detail. It was found that the counter in the ADDIE was liable to jump states in an unpredictable manner, causing severe distortion of the learning curve and steady state trajectories. In order to avoid spurious pulse-beating effects, it was necessary to synchronise the PRBS and ADDIE clocks to a central master clock. It was found that merely placing an inverter in the clock feed to the latch PRBS generator considerably reduced the severity of the state-jumping effect. This in turn suggested that the root cause of the problem was pulse coincidence effects in a system containing an unavoidable mixture of leading and trailing edge-triggered devices.

The overall solution to this problem was applied in two parts. The synchronous counters in the ADDIE were replaced with asynchronous types so that no input condition constraints would be violated in operation. The ADDIE was then modified to include end-point detection circuits which stopped the counter just short of all-ones or all-zeroes using the asynchronous parallel load facility to temporarily "freeze" the ADDIE at the boundary value. Satisfactory operation then ensued.

A useful feature of the ADDIE SLA is that it permits a realistic initial condition to be established for the learning process, i.e., $p_1(0) = 0.5$, corresponding to random state selection at time t_o. This was achieved in practice by loading "one-all-zeroes" via a data selector on the parallel inputs. The circuit of this fully-developed ADDIE is shown in Figure 3.4.

3.5 /

3.5 Operating Sequence

The full operating sequence for the ADDIE SLA is now described. The initial load operation sets up the requisite value of $p_1(0) = 0.5$, as above. The output of the ADDIE comparator is therefore a bit stream with an equal probability of 1's and 0's. At the first system clock pulse, this signal is sampled by the system flip-flop, and simultaneously the counter contents are copied into the latch. The action a_i represented by the system flip-flop output selects a reward/penalty or c_i signal from the environment. On the falling edge of the clock pulse, this signal is sampled, setting up the algorithm address, while the ADDIE clock is enabled, allowing the learning period to commence. During this time, the ADDIE adjusts to the updated value of $p_1(1)$, which in turn forms the basis for the next cycle of operation.

The fundamental advantage of this design is that no hardware race-around or locking-on problems can occur. The design is such that several of the reinforcement schemes reported previously,[19] in particular the more suitable ϵ-optimal schemes, can be successfully employed, utilising the established method of algorithm circuit design described in Chapter 2.

3.6 Algorithm Circuits

The design for the L_{R-P} scheme has already been described, and the obvious development from here is the implementation of the ϵ-optimal linear reward-inaction (L_{R-I}) and linear reward-reward (L_{R-R}) schemes. The L_{R-I} scheme is particularly simple to accommodate, since the only modification required is to set $\beta = 1$.

The L_{R-R} scheme, in which the penalty response is replaced /

replaced by a lesser reward, is given below in two-state form:

(a) Reward (action a_1)

$$p_1(n+1) = 1 - \alpha\, p_2(n)$$
$$p_2(n+1) = \alpha\, p_2(n)$$

(b) Penalty (action a_1)

$$p_1(n+1) = 1 - \beta\, p_2(n)$$
$$p_2(n+1) = \beta\, p_2(n)$$

where $0 < \beta < \alpha < 1$

As before, a truth-table can be constructed to enable this algorithm to be translated into a hardware design:

$a_1(n)$	$a_2(n)$	P/R	$p_1(n+1)$
0	1	0	$\alpha\, p_1(n)$
0	1	1	$\beta\, p_1(n)$
1	0	0	$1 - \alpha\, p_2(n)$
1	0	1	$1 - \beta\, p_2(n)$

Note that $p_2(n+1)$ need not be stated explicitly.

Comparing this with the truth table for the L_{R-P} scheme, it is evident that the L_{R-P} circuit can be converted to an L_{R-R} circuit simply by reversing the $\beta\, p_1(n)$ and $1 - \beta\, p_2(n)$ connections to the data selector. The circuit arrangements for these linear schemes are summarised in Figure 3.5.

3.7 Non-Linear Scheme

Although the "best" of the linear schemes, the L_{R-I} scheme, has been widely reported[20, 21, 59] as most suitable for many applications, considerable study has also been made[8, 19, 27] of non-linear reinforcement algorithms. These /

These tend to show higher rates of convergence, and indeed the chief motivation behind the investigation of these schemes has been the desire to obtain the best possible convergence times. This is especially the case if they are incorporated with a linear algorithm to form a hybrid scheme[19].

The simplest of the non-linear schemes is that denoted by Narendra et al as $N_{R-P}^{(1)}$, which has "square-law" non-linearity. It has been shown[8, 19] that this scheme is conditionally optimal, providing optimal convergence if $c_i < \frac{1}{2} < c_j$, and expedient otherwise. This scheme, again considering the two-state case, is given below:

(a) Reward (action a_1)

$$p_1(n+1) = p_1(n) + \alpha\, p_1(n)\,[1 - p_1(n)]$$

$$p_2(n+1) = p_2(n) - \alpha\, p_1(n)\,[1 - p_1(n)]$$

(b) Penalty (action a_1)

$$p_1(n+1) = p_1(n) - \beta\, p_1(n)\,[1 - p_1(n)]$$

$$p_2(n+1) = p_2(n) + \beta\, p_1(n)\,[1 - p_1(n)]$$

where $0 < \alpha,\ \beta < 1$

The truth-table for this scheme is as follows:

a_1	a_2	P/R	$p_1(n+1)$
0	1	0	$p_1 - \alpha\, p_2(1 - p_2)$
0	1	1	$p_1 + \beta\, p_2(1 - p_2)$
1	0	0	$p_1 + \alpha\, p_1(1 - p_1)$
1	0	1	$p_1 - \beta\, p_1(1 - p_1)$

The hardware implementation of this scheme as expressed here is complicated by two factors. The first is /

is the presence of terms of the form $p_i(n) \cdot [1 - p_i(n)]$, i.e., $p_1(n) \cdot p_2(n)$. This product cannot be formed simply by an AND-gate as before, because of the direct complementary relationship between the signals. The solution here is to interpose a delay of one clock interval on one of the multiplier inputs, along the lines of the stochastic squarer circuit shown in Figure 1.4(b). The other, more serious problem is that summation of stochastic variables involves an AND-OR configuration with random switching between the two signals by means of a separate noise line (Figure 1.4(d)). As explained earlier, this introduces an attenuation factor of 0.5.

It was therefore felt preferable to re-arrange the terms of the algorithm so that they involved only multiplication and inversion operations. Again, only $p_1(n+1)$ is required to be generated. The algorithm terms are then transformed as follows:

(i) $p_1 + \alpha\, p_1(1-p_1) = 1 - \left[p_2 - \alpha\, p_1(1-p_1)\right] = 1 - (1-\alpha\, p_1)\, p_2$

(ii) $p_1 - \beta\, p_1(1-p_1) = (1-\beta\, p_2)\, p_1$

etc.

This results in the following truth-table:

a_1	a_2	P/R	$p_1(n+1)$
0	1	0	$(1 - \alpha\, p_2)\, p_1$
0	1	1	$1 - (1 - \beta\, p_1)\, p_2$
1	0	0	$1 - (1 - \alpha\, p_1)\, p_2$
1	0	1	$(1 - \beta\, p_2)\, p_1$

Comparing this revised version with the table for the standard L_{R-P} scheme, an essential similarity is evident. In the $N^{(1)}_{R-P}$ scheme, the simple constants α, β are replaced with terms of the form $(1 - \alpha\, p_i)$ and $(1 - \beta\, p_i)$ /

$(1 - \beta p_i)$.

The hardware implementation for this scheme is shown in Figure 3.6, showing how two levels of multiplication are involved. The extra D-type flip-flop introduces a one bit delay on the αp_i and βp_i signals to preserve the property of statistical independence between multiplier inputs.

3.8 Construction

As in the case of the flip-flop SLA, the assembled system was fitted in a cabinet with a front panel patching arrangement to provide a choice of constants for α, β and c_i. Due to the similarity between the family of algorithm circuits, it was possible also to accommodate a simple switching arrangement to select the desired reinforcement scheme. The ADDIE SLA is illustrated in Figure 3.7, and the results obtained from this system are detailed in the following chapter.

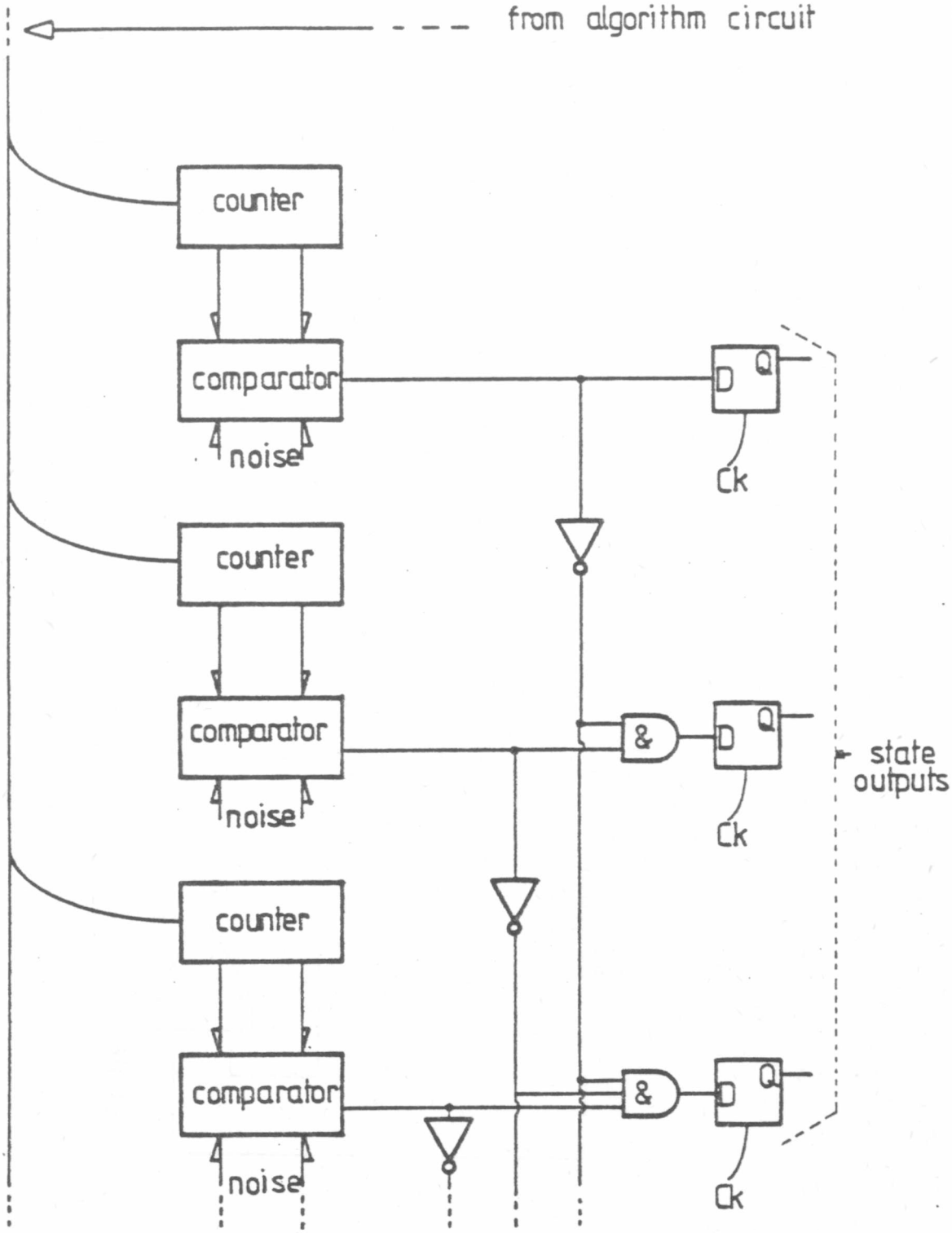


Figure 3.1 r-state SLA configuration

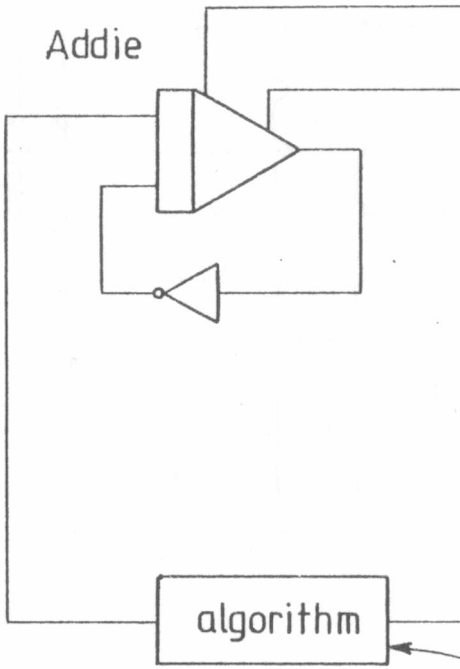
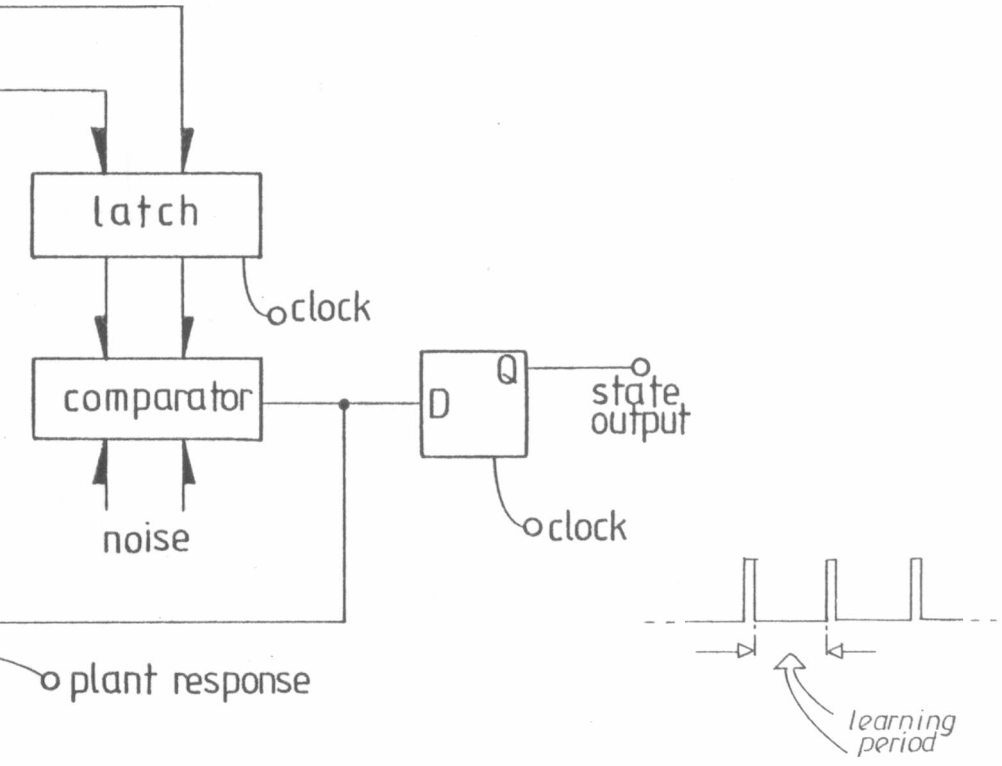


Figure 3.2



Two - state ADDIE SLA

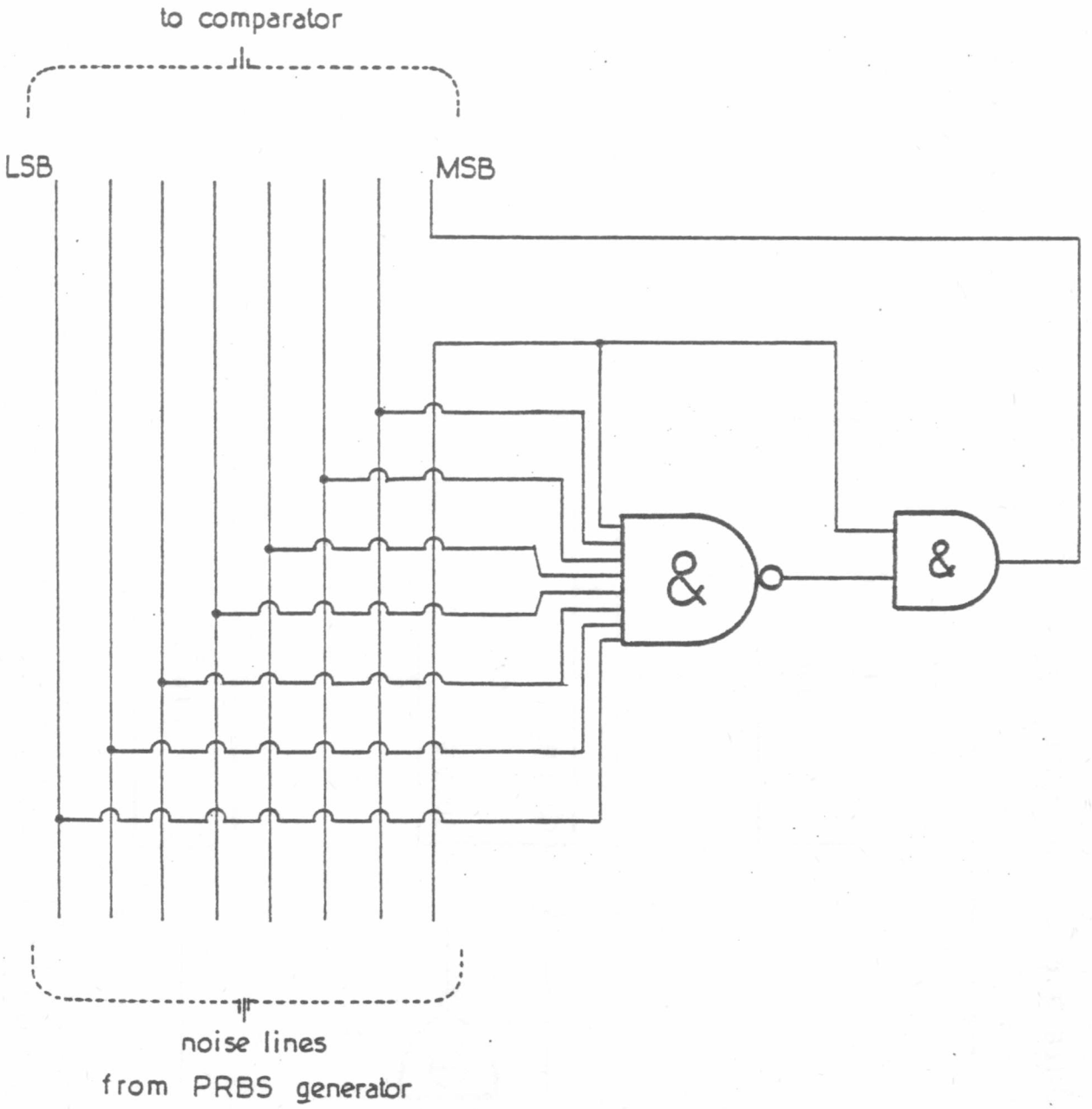


Figure 3.3 Modified noise supply

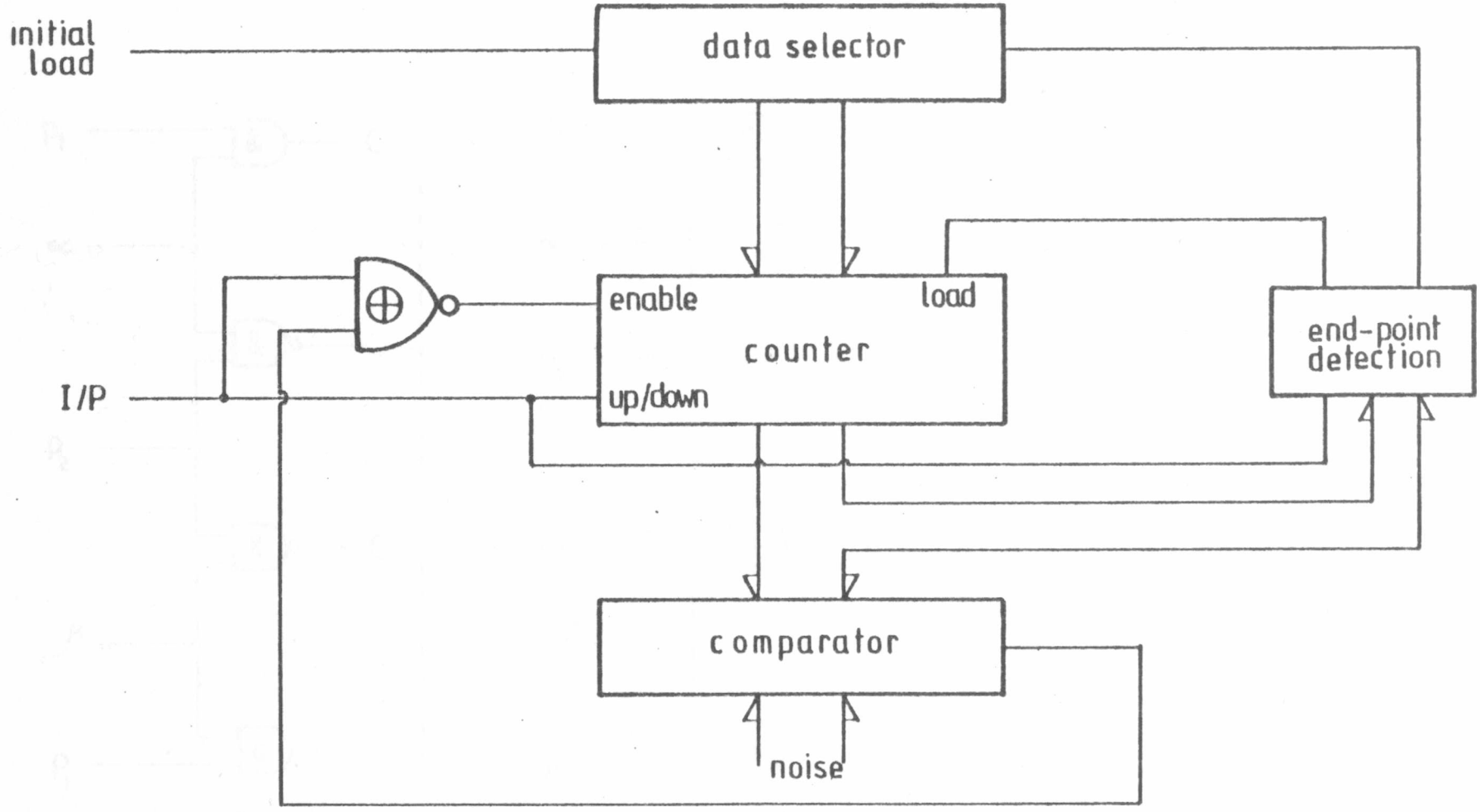
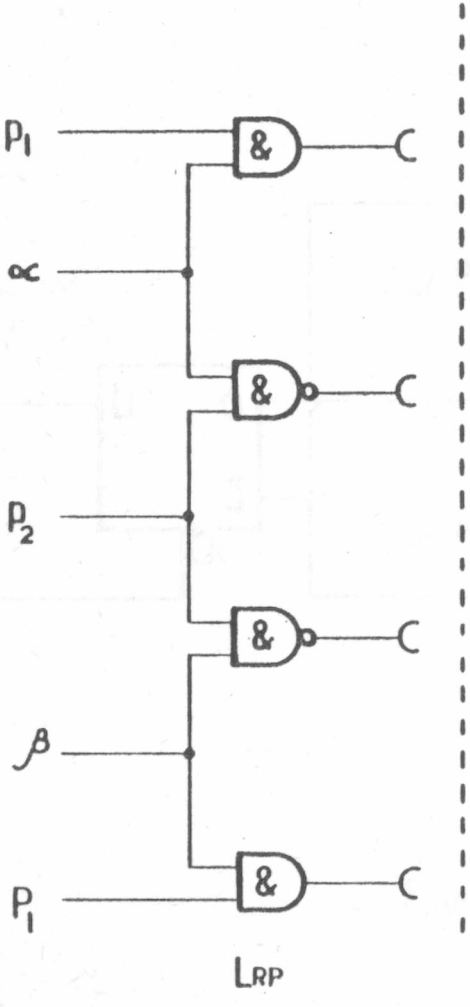


Figure 3.4 Practical ADDIE in SLA application



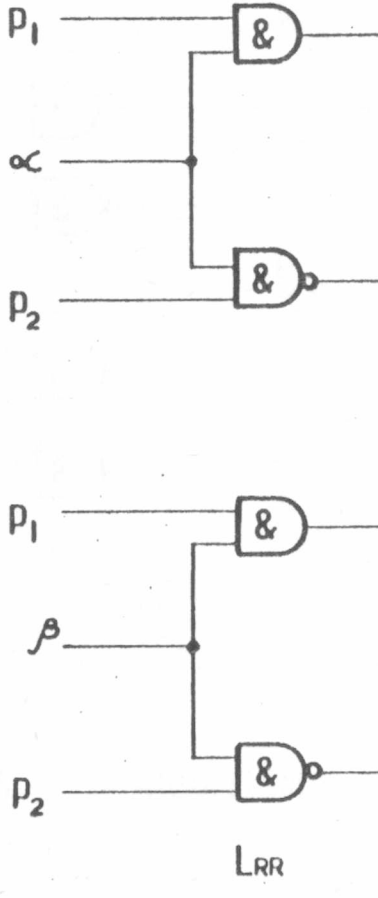
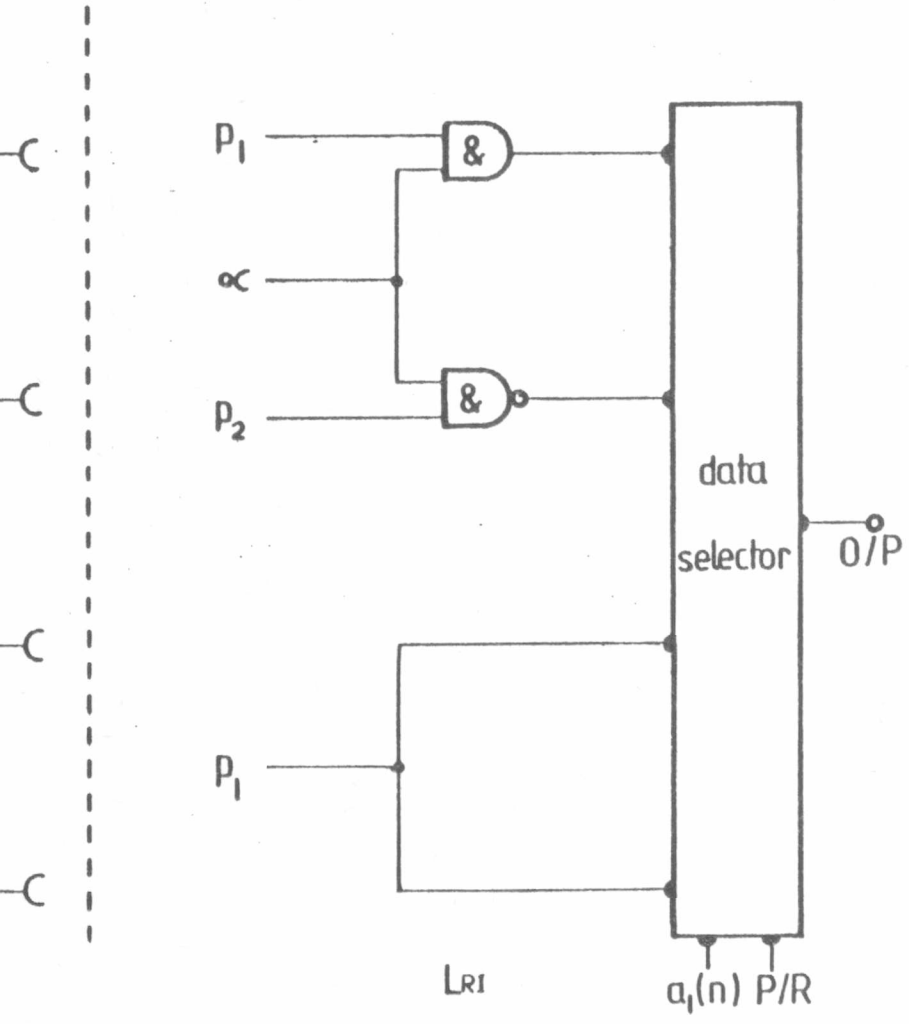


Figure 3.5 Algorithm circuits



for linear schemes

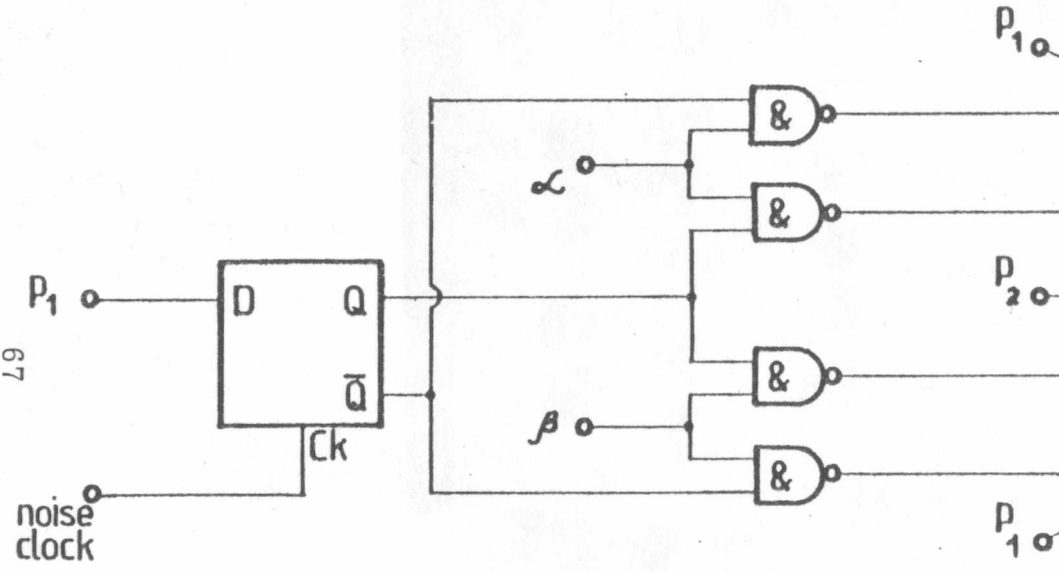
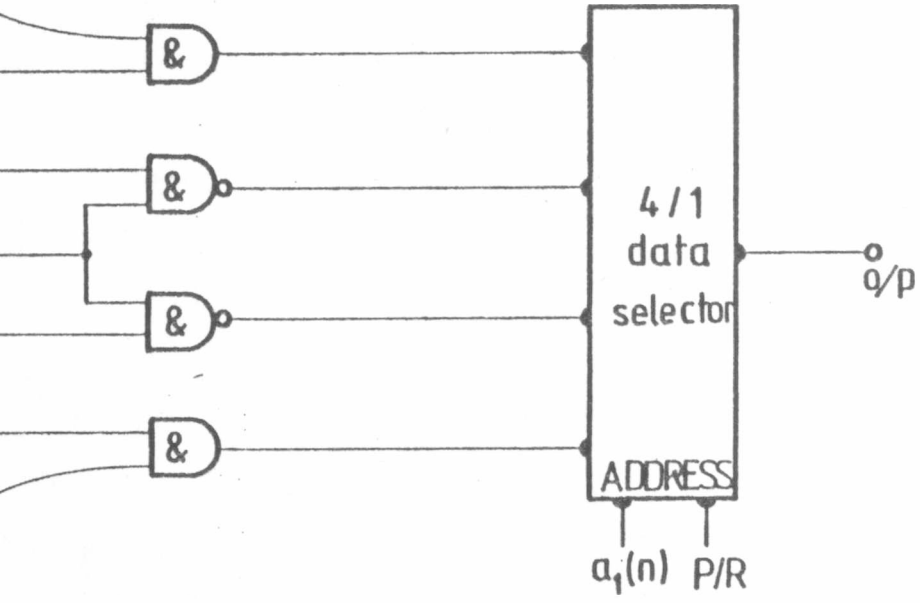


Figure 3.6 Algorithm



circuit for the $N_{R-P}^{(1)}$ scheme

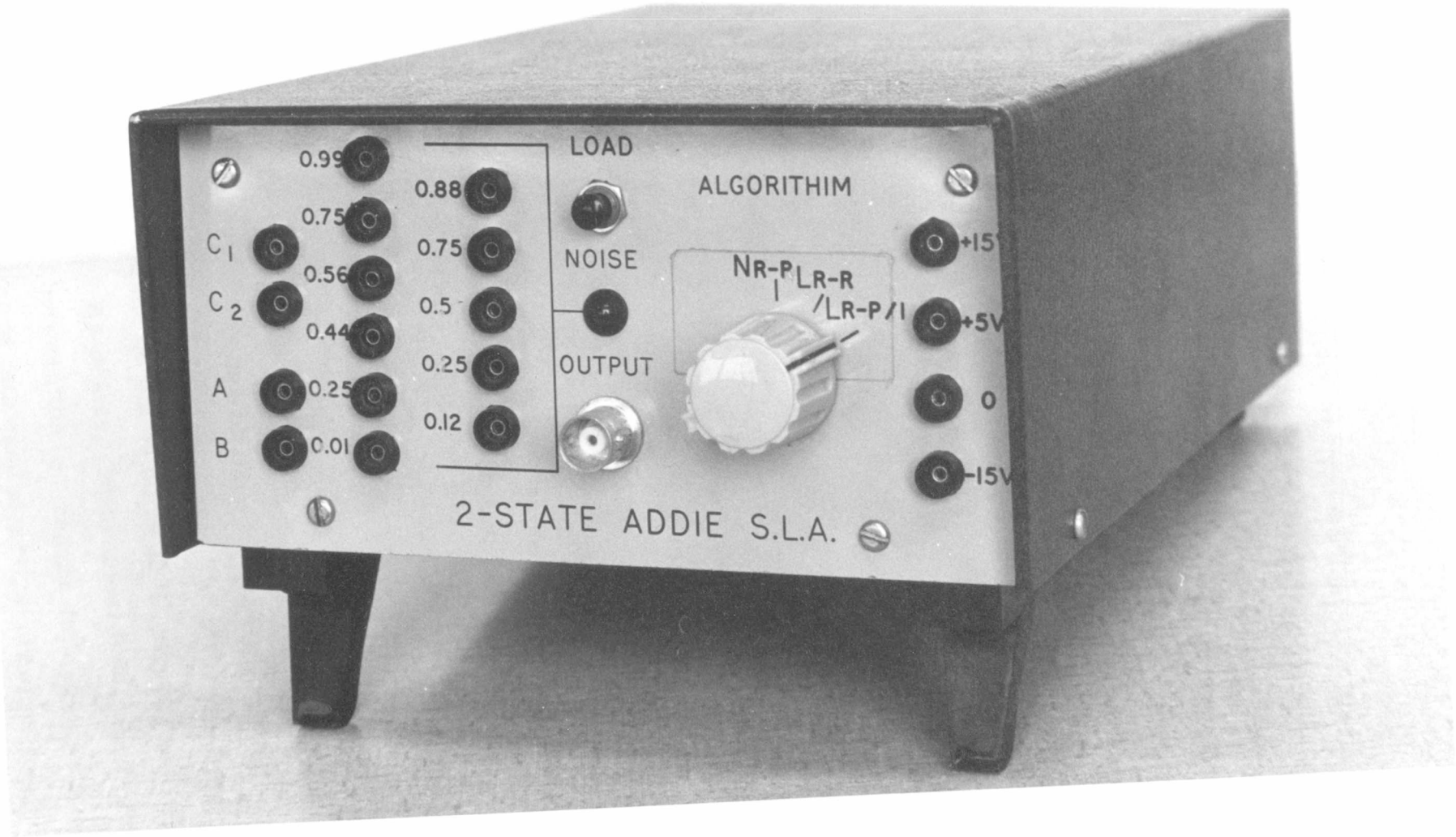


Figure 3.7 ADDIE SLA

CHAPTER 4
RESULTS FROM THE ADDIE SLA

4.1 Introduction

A series of experiments was carried out with the ADDIE SLA in order to assess its learning characteristics with a variety of reinforcement schemes in operation. As before, it was decided to present the results via storage oscilloscope display, using a filter on the output of the system flip-flop to provide the requisite stochastic-to-analogue interface. The master clock was set to run at 10 MHz, while the main sampling clock consisted of 100 ns pulses at a frequency f_s of 100 kHz, giving a system cycle time of 10 μs.

4.2 L_{R-P} Scheme

Learning curves obtained with the L_{R-P} scheme are reproduced in Figures 4.1 - 4.4. The first two results confirm that, as before, the effect of increasing reward-penalty ratio γ is to increase the degree of expediency. Comparing these curves with those from the flip-flop SLA (Figure 2.9) which used the same system clock rate, it can be seen that learning times and steady-state action probability values are virtually identical.

Two other experiments performed earlier on the flip-flop SLA were also repeated here. Figure 4.3 shows the characteristics for both possible allocations of penalty probabilities, using a $\gamma = 8$ scheme. In each case, the automaton can be seen to select the action which incurs the lower penalty probability. The influence of relative c_i values on the outcome is illustrated in Figure 4.4. As before, it turns out that reducing the discrimination between the c_i results in less expedient behaviour, using in this case a $\gamma = 2$ scheme.

4.3 /

4.3 L_{R-I} Scheme

The results obtained with the ϵ-optimal L_{R-I} scheme are illustrated in Figures 4.5 - 4.7. They show that with this scheme the automaton's performance is indeed almost optimal, with $p_1(n)$ (or $p_2(n)$) close to unity in the steady-state. There is a small probability of not performing the correct action, as indicated by the various "blips" on the trace, particularly evident in Figure 4.6, which are chiefly the result of ADDIE variance. A comparison of Figure 4.6 and 4.7, which in fact record the results of two separate trials with the same system parameters, gives some indication of the typical variations to be found in the transient and steady-state behaviour. Figure 4.7 is particularly interesting, showing a state trajectory with two clear reversals in learning behaviour during the initial learning period.

The choice of algorithm reward factor also influences the learning characteristics of the L_{R-I} automaton. Results obtained with $\alpha = 0.75$ exhibit a slightly higher steady-state value, i.e., higher expediency, than was obtained for the case with $\alpha = 0.25$ (Figure 4.5). This accords with the conclusions of previous simulation studies,[20, 21] showing that for best results with this scheme, the "step-size" factor, which is $1 - \alpha$ here, should be small.

4.4 L_{R-R} Scheme

The L_{R-R} scheme is comparable in expediency to the L_{R-I} scheme when γ is sufficiently high, though its rate of convergence has been found to be slower.[20] The results obtained here with this scheme are reproduced in Figure 4.8. As with the L_{R-P} scheme, the curves indicate a degree of expediency /

expediency which increases with γ.

The general conclusion to be gained from all the linear scheme results is that there is no significant difference in expediency, as far as this hardware implementation is concerned, between L_{R-I} schemes, and L_{R-P} or L_{R-R} schemes with high γ-factor. In addition, there appears to be little discernible difference between the learning times of the various schemes investigated so far. However, the chief determinant of transient response in the hardware is the output filter. As explained in section 2.5, this element has a bandwidth which is purposely restricted in order to achieve an acceptable compromise between the speed of convergence, and variance in the steady-state.

4.5 $N^{(1)}_{R-P}$ Scheme

The $N^{(1)}_{R-P}$ scheme is the most elementary of the non-linear schemes to have been reported, and is, as described in Section 3.7, quite straightforward to implement in hardware. It was possible to verify in practice the property of conditional optimality which is a prominent feature of this scheme.

Using penalty probabilities of 0.25 and 0.75, the system converged as shown in Figure 4.9, with generally lower variance than was evident with the best of the linear schemes. However, when the c_i values were altered to 0.75 and 0.875 (i.e., both greater than 0.5), the system displayed a rather low level of expediency with considerable variance. Figure 4.10 shows a typical result, with an "optimal" curve superimposed for comparison. The time scale here is 5 ms per division, which enables the fast learning time for the latter curve in particular to be appreciated. Figure 4.11 illustrates the results of experiments /

experiments with more widely separated c_i, which serves to emphasise the very high degree of expediency which can be achieved with this scheme, given the right conditions.

The scheme was further investigated by feeding the penalty probabilities via a switching circuit, thus enabling a direct comparison to be made of steady-state behaviour under "optimal" and "expedient" conditions. The result is shown in Figure 4.12. One of the penalty probabilities was set at 0.125, while the other was switched periodically about the critical value of 0.5, as indicated by the control waveform. It can be seen that with $c_i = \{0.125, \ 0.56\}$, the state trajectories yield optimal convergence, whereas with $c_i = \{0.125, \ 0.44\}$, the system degrades, as predicted, to expedient behaviour.

4.6 Plant Simulator

It was decided to test the ADDIE SLA also with the plant simulator circuit developed originally for the flip-flop SLA. This system, described in detail in Chapter 2, is effectively a generator of "noisy" c_i, and represents the more practical situation in which the reward/penalty signals are themselves subject to random variations.

The two systems were linked up and tested with two reinforcement schemes of high expediency: the L_{R-I} scheme ($\alpha = 0.75$) and the L_{R-R} scheme ($\gamma = 64$). The result for the L_{R-I} automaton is shown in Figure 4.13. The high noise boundary clearly has a degrading effect on the learning ability of the system, though the choice of action 1 does predominate.

The plant simulator was then tested on the L_{R-R} automaton, and a typical result is shown in Figure 4.14, this time favouring action 2. In this particular example, a /

a marked reversal of learning behaviour in the initial stages is evident, similar to the effect noted in Figure 4.7. Again, the result is characterised by lower expediency and higher variance, in comparison with those experiments which used invariant c_i values.

Both results indicate, however, that as in the case of the flip-flop SLA, the ADDIE automaton has the ability to discriminate in its choice of action even if the reward/penalty information is heavily noise-corrupted. The rather high steady-state variance observed in these plant simulator results is probably due in part to the fact that the sampling frequency for the noisy c_i, i.e., the latch clock of Figure 2.16, is just one-tenth that of the main system clock. This means that in practice the rate of fluctuation of c_i values from one iteration to the next cannot be very high, thereby causing considerable excursions from the main learning trend.

4.7 Performance Curves

As an adjunct to the learning curve results described above, it is instructive to consider also an alternative presentation of automaton behaviour in the form of a performance curve which shows the variation in average received penalty as the learning process evolves. As defined in Chapter 1, average received penalty is denoted by:

$$M(n) = \sum_i p_i(n)\, c_i(n)$$

Since p_1, p_2, c_1, c_2 are all present in the hardware simulation, it is a simple matter to generate a continuous signal representing the quantity M, and present the result on the oscilloscope via a filter as before.

Figures /

Figures 4.15 through 4.18 illustrate typical performance curves for the following reinforcement schemes: $L_{R-P}(\gamma = 7)$, $L_{R-I}(\alpha = 0.125)$, $L_{R-R}(\gamma = 7)$, $N^{(1)}_{R-P}$ respectively. In each case, $c_i = \{0.25, \ 0.75\}$, so that the asymptote of optimum performance is $M_o = 0.25$. As expected, the more expedient schemes are most successful in achieving performance verging on the optimum, whereas the L_{R-P} scheme falls somewhat short, and also shows characteristically higher variance.

4.8 Non-Stationary Environments

The results presented so far have concentrated on learning behaviour in a stationary environment. Since most physical systems are not time-invariant, however, it is important from a practical veiwpoint to evaluate the adaptive or "tracking" properties of the learning automaton.[60-64] In a non-stationary environment, the c_i are some function of stage number n; this relationship may be linear or non-linear, periodic or random in nature. Rather than consider excessively complex systems, it was decided to concentrate on a two-action system, undergoing some step-wise interchange in the relative order of the c_i (Figure 4.19).

An important criterion of performance for an SLA operating in a non-stationary environment is the mean adjustment time.[62] This parameter can be defined as the average number of iterations n_o such that:

$$p_1(n) > p_2(n), \text{ for all } n \in [0, \ n_o] \ ,$$

$$\text{and} \quad p_2(n_o) \geqslant p_1(n_o) \ ,$$

where it is assumed that the penalty probabilities were switched at $n = 0$.

Since /

Since the linear reinforcement schemes L_{R-P} and L_{R-I} are of primary interest, they form the basis for the following experiments. It is possible to calculate the mean adjustment time of an automaton as a function of the learning scheme parameters and the penalty probabilities, so that each experimental result can be compared with the corresponding theoretical value. In Appendix A, expressions are derived for the expected value of state probability for two-state L_{R-P} and L_{R-I} automata. It is shown that for L_{R-P}:

$$E\left[p_1(n+1)\right] = (c_2-c_1)(\alpha-\beta)(p_1(n))^2 + \left[1+(1-\alpha)(c_2-c_1) - 2(1-\beta)c_2\right] p_1(n)+(1-\beta)c_2$$

and for L_{R-I}:

$$E\left[p_1(n+1)\right] = \left[1+(1-\alpha)(c_2-c_1)(1-p_1(n))\right] p_1(n)$$

As expected, the L_{R-I} formula, though derived separately, turns out to be the particular case of the L_{R-P} formula with $\beta = 1$. Mean adjustment time is then deduced from the above by solving iteratively for n_o such that $p_1(n_o) \leqslant 0.5$, where $p_1(0)$ is the steady-state starting value prior to switching, e.g., 0.90.

4.9 Results

The dynamic characteristics of the SLA operating in a non-stationary environment can be readily observed by feeding the c_i via a changeover circuit with either periodic or random switching. Figure 4.20 shows typical behaviour observed in each case, using an L_{R-I} scheme with $\alpha = 0.125$. It can be seen that provided the minimum switching period of the environment exceeds the SLA adjustment time, successful tracking is achieved.

In /

In order to observe the actual state transition characteristics with greater accuracy, it was decided to dispense with the filter output interface and use instead a D/A converter connected directly to the ADDIE counter. Figure 4.21 shows a typical transition curve obtained in this way, complete from one state to the other, with the system clock waveform for the period up to $p_i(n) = 0.5$ superimposed. In this particular case, 27 iterations occurred in the adjustment period.

Past results have confirmed that there is considerable variation in the transient behaviour of the automaton from one learning cycle to another. It was therefore felt necessary to take large sample average measurements of adjustment time to obtain sufficiently reliable experimental values. This consideration led to the design of an automatic logging facility, outlined in Figure 4.22, which records a large, preset number of readings from a system with continuous, periodic switching of the environment. Each time the c_i are switched, SLA sampling clock pulses are passed to the totalising counter via gates 1 and 2 while the ADDIE state undergoes transition from 0.90 to 0.5. The occurrence of each burst of pulses is recorded by a separate logging counter controlling gate 2. After 5000 cycles, the totalising counter thus presents a direct reading of (mean adjustment time) x 5000.

A series of experiments was carried out with the L_{R-I} scheme, in which both the reward factor α and the sampling clock frequency f_s were varied. The results are plotted in Figure 4.23, with the theoretical values included for comparison. It can be seen that the mean adjustment time decreases as the algorithm step size $(1 - \alpha)$ increases, as would be expected. The variation in f_s does appear to have some /

some influence also, with the median value of 12.5 kHz returning the most consistent results. It appears that there is in practice an optimum duration for the updating phase, which must be of sufficient length to accommodate fully each ADDIE transition dictated by the reinforcement algorithm, while not permitting undue "drifting" errors caused by steady-state variance to build up.

Results for the L_{R-P} scheme are shown in Figure 4.24. The variation in reward-penalty ratio γ, which is of course proportional to $1 - \alpha$, produces a similar trend in mean adjustment time against step size to that observed above. It is also interesting to note that, comparing L_{R-I} and L_{R-P} schemes with the same reward parameter, e.g., $\alpha = 0.75$, $\gamma = 32$, the L_{R-P} gives faster adjustment. The presence of just a small penalty factor clearly enhances the ability of the automaton to recognise and adapt to changes in the environment.

4.10 Conclusions

The foregoing results have verified that the ADDIE SLA has very suitable learning characteristics, and are entirely consistent with previous theoretical predictions. Although a certain amount of serial processing is involved in its operation, the combination of 8-bit configuration and high clocking rate have enabled it to return similar learning times to the basic flip-flop SLA. Indeed, the results from that prototype system, using the L_{R-P} scheme, have been virtually duplicated here, with the additional advantage that a more comprehensive range of reinforcement algorithms can be implemented.

It has been shown also that the ADDIE SLA is well able to adapt to changes in the environment, again yielding results which /

which are in good agreement with the theoretical values. This has important implications for the use of the automaton with non-stationary systems, where most practical applications are likely to be found.

The ability to obtain convergence in the space of a few milliseconds represents a very significant advance, particularly in view of the subsequent extension to learning systems calling for much higher state order, but incorporating the same basic ADDIE SLA design. This development is detailed in the following chapter.

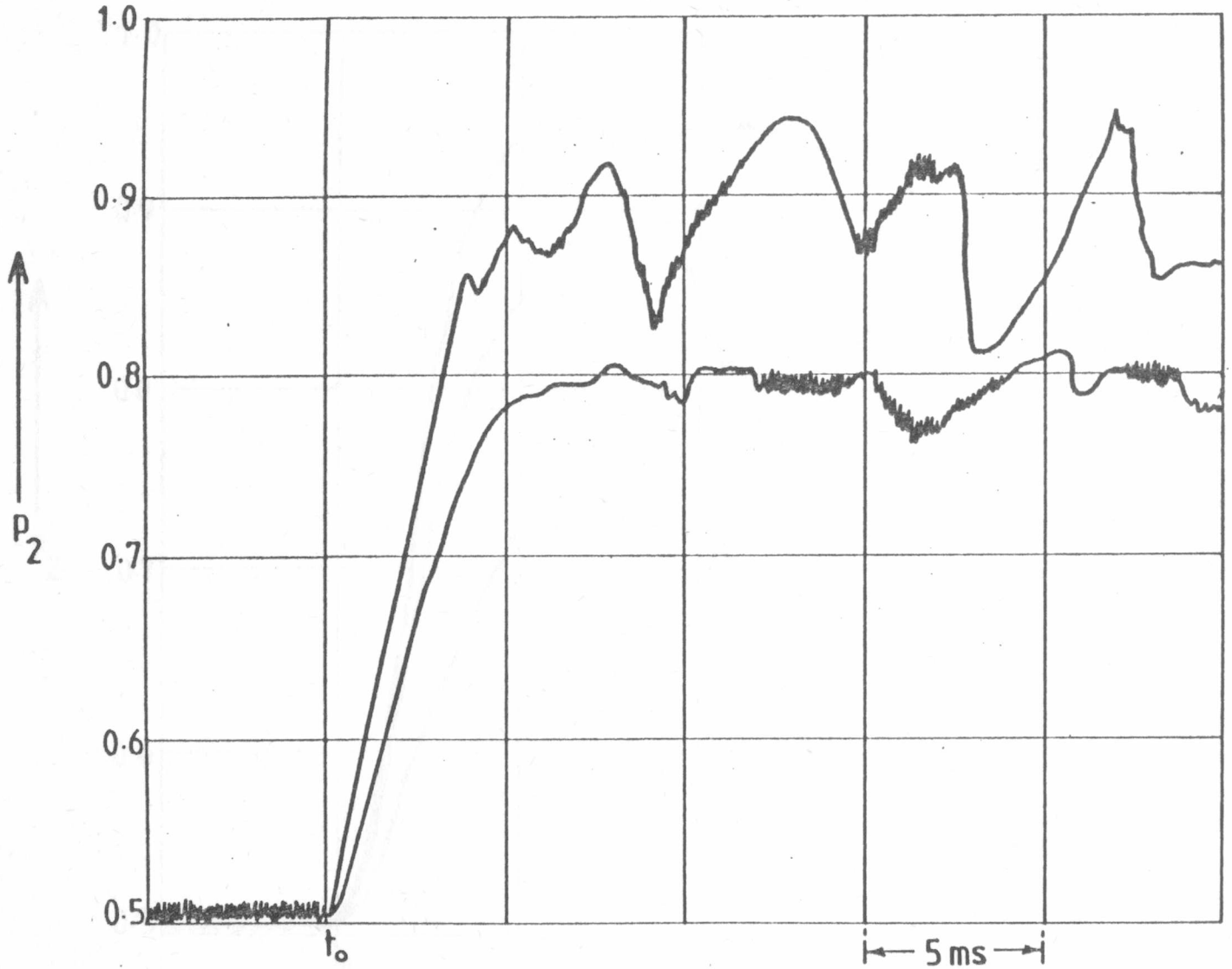


Figure 4.1 L_{R-P} scheme (1)

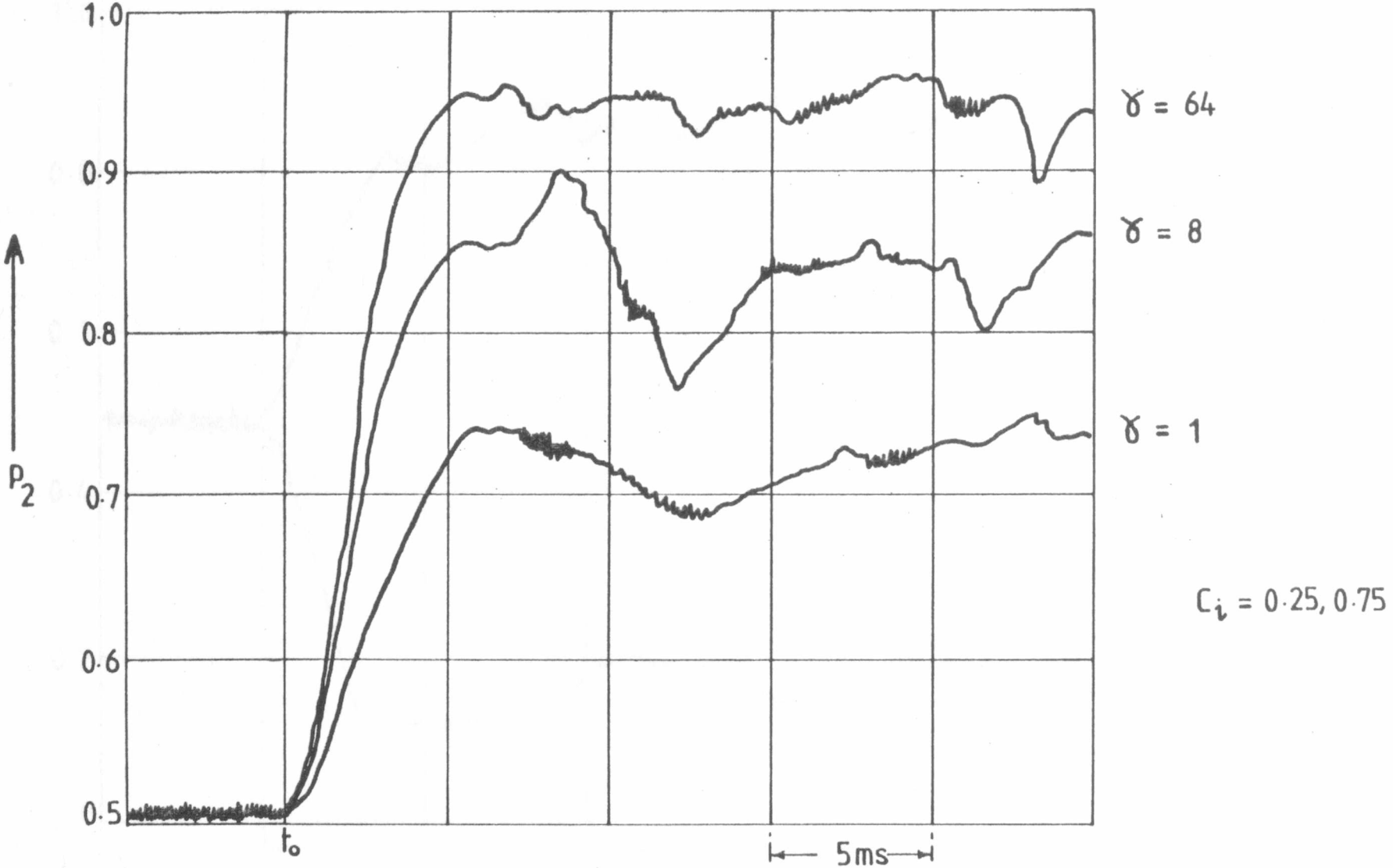


Figure 4.2 L_{R-P} scheme (2)

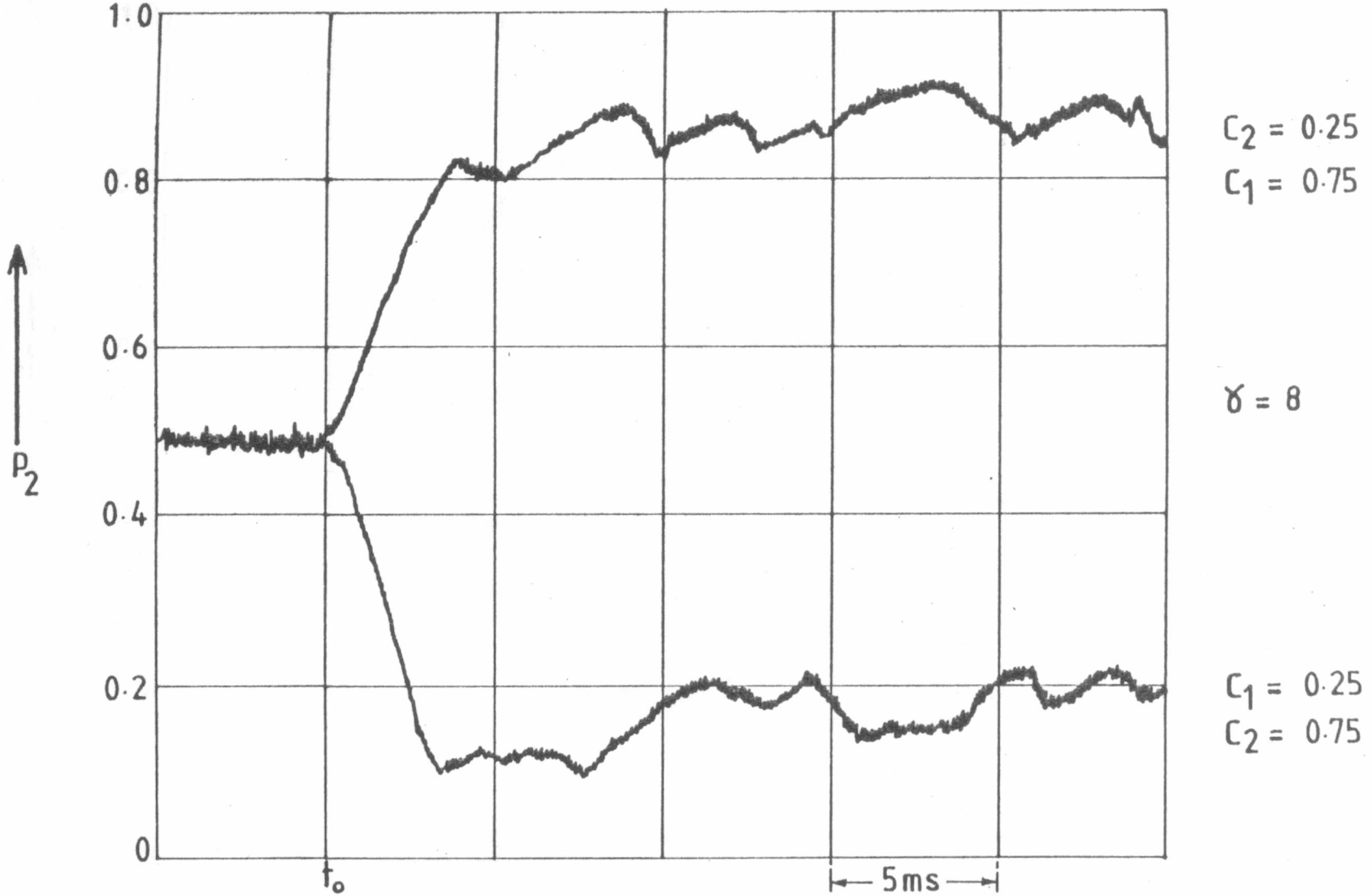


Figure 4·3 Learning behaviour for both states

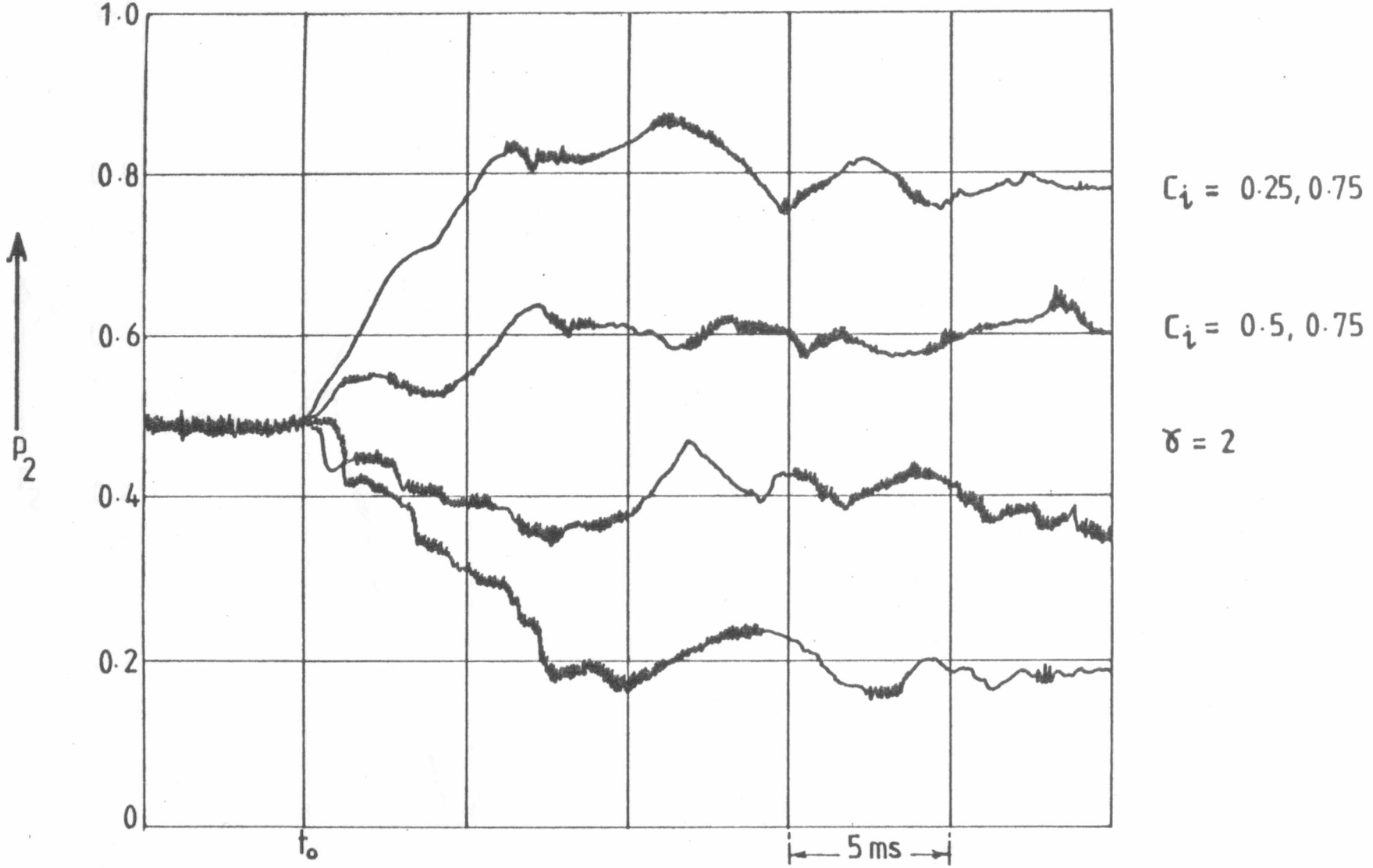


Figure 4.4 Comparison of expediency for different C_i

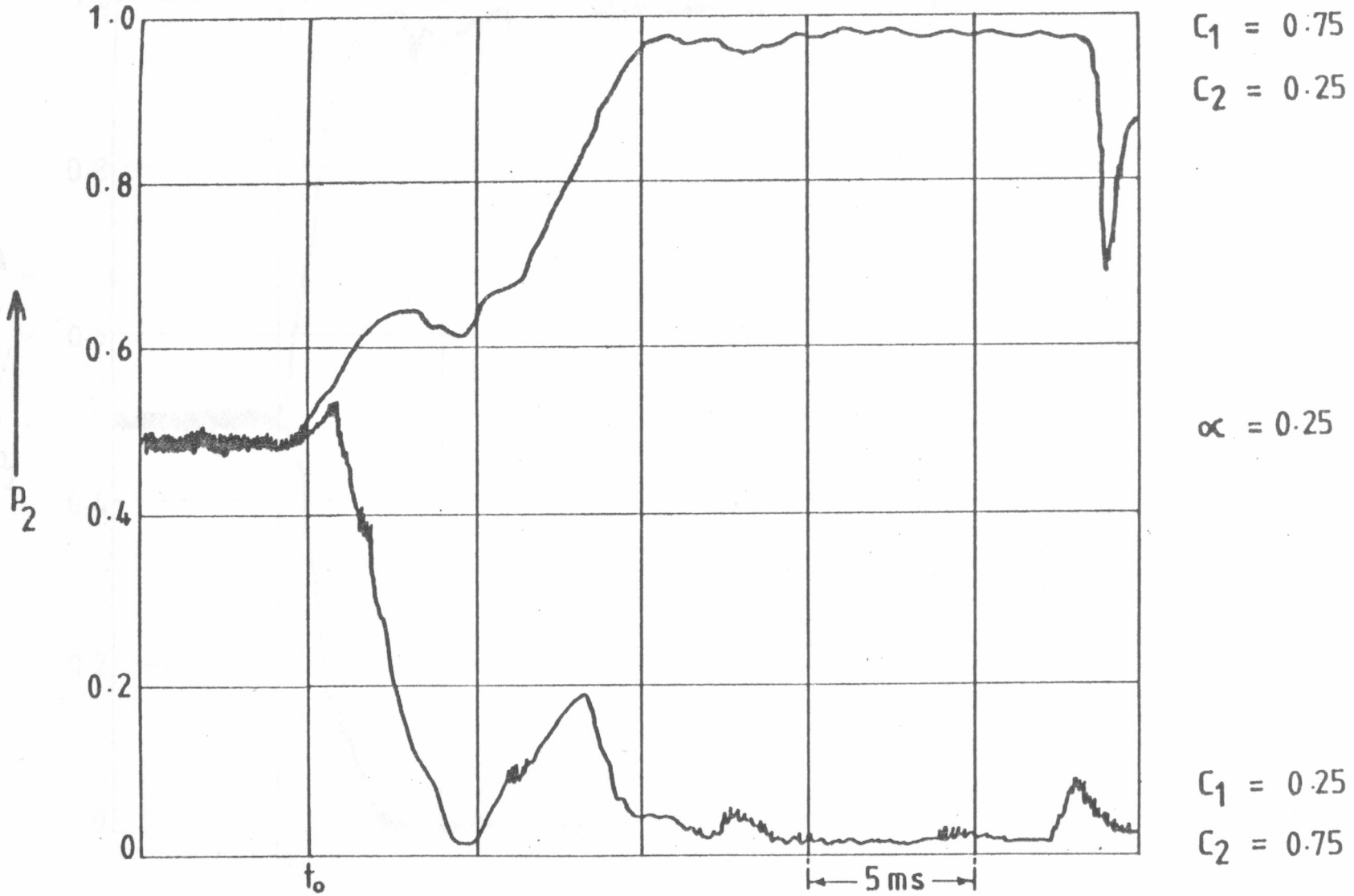


Figure 4·5 L_{R-I} scheme (1)

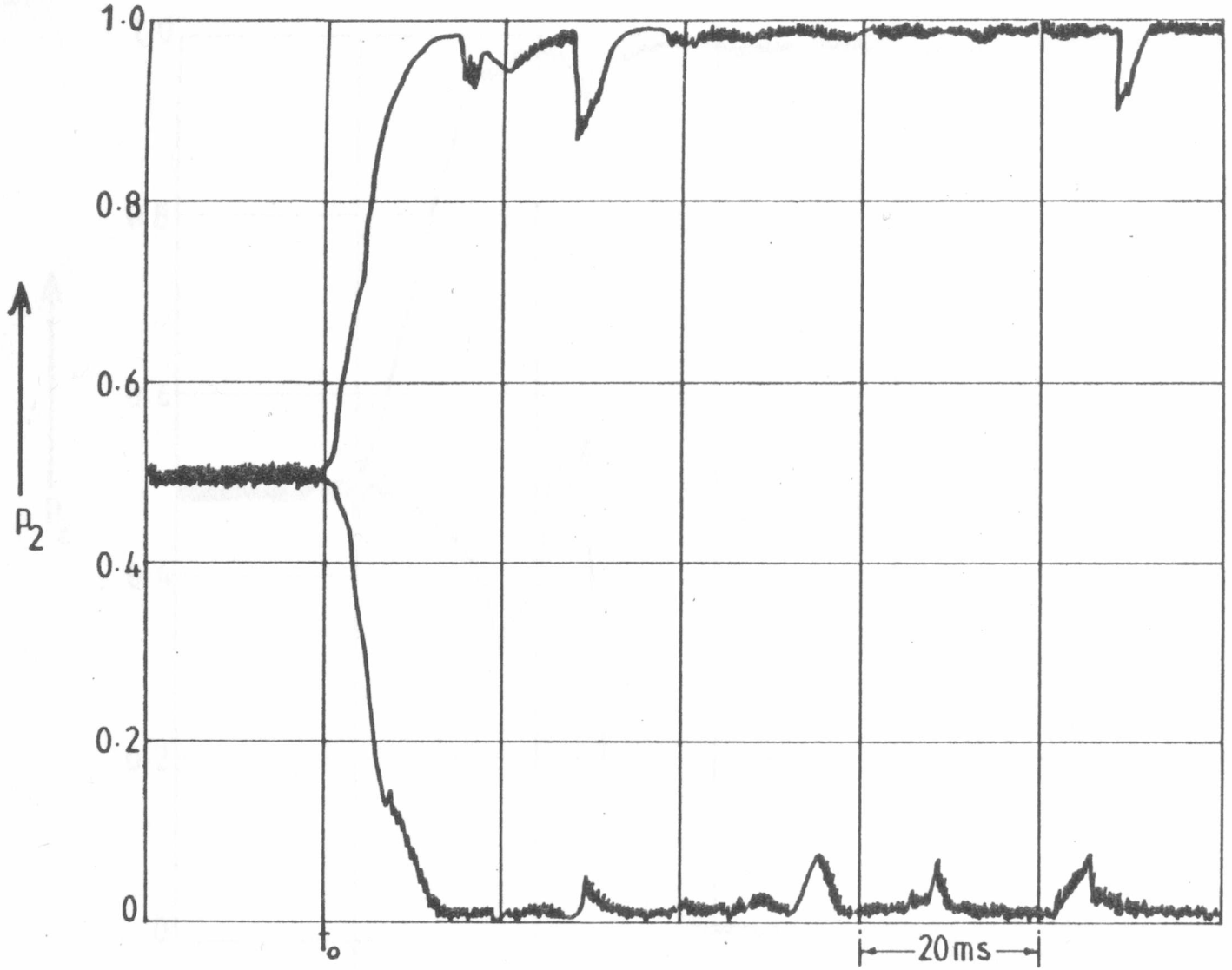


Figure 4·6 L_{R-I} scheme (2)

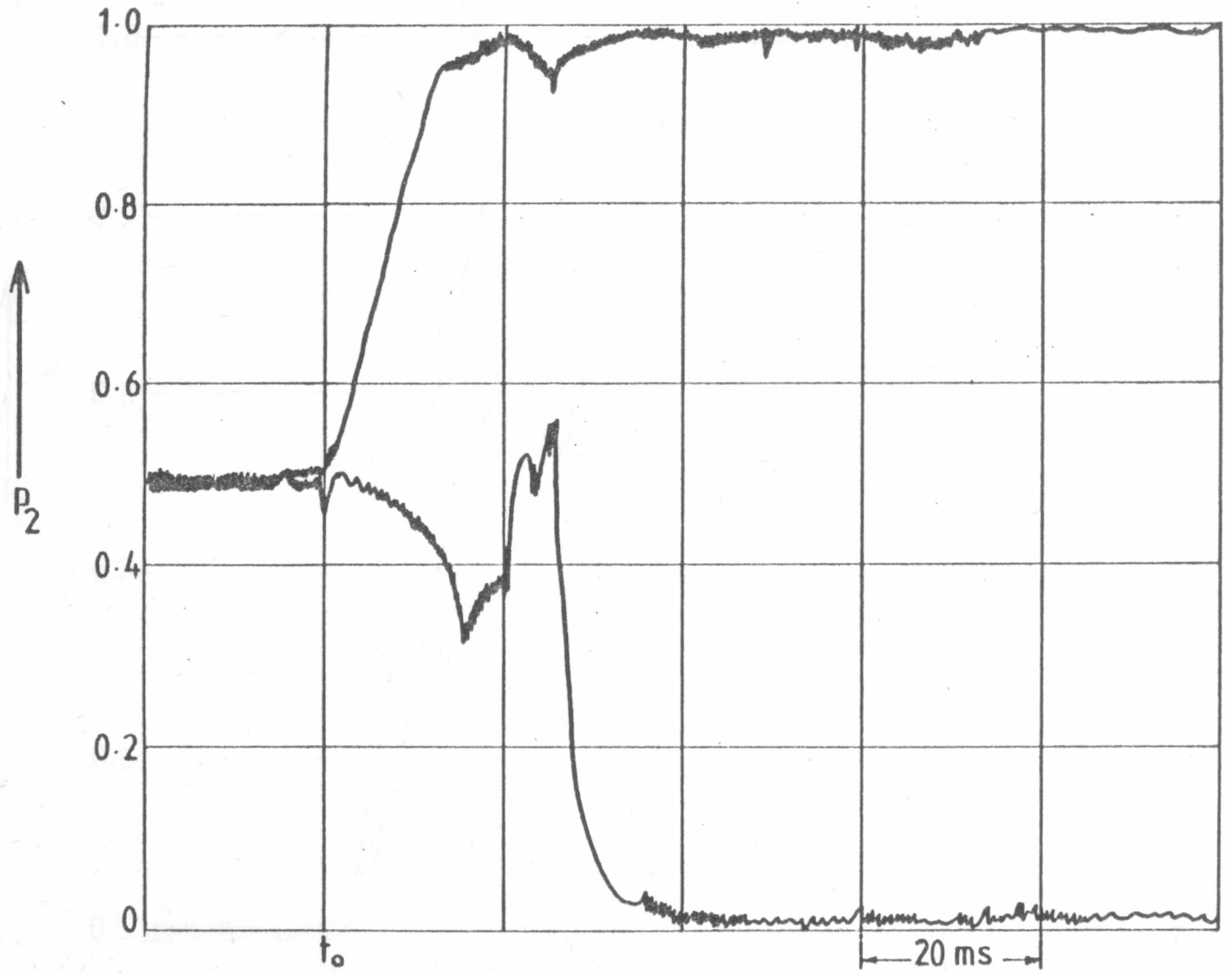


Figure 4·7 L_{R-I} scheme, showing initial reversal

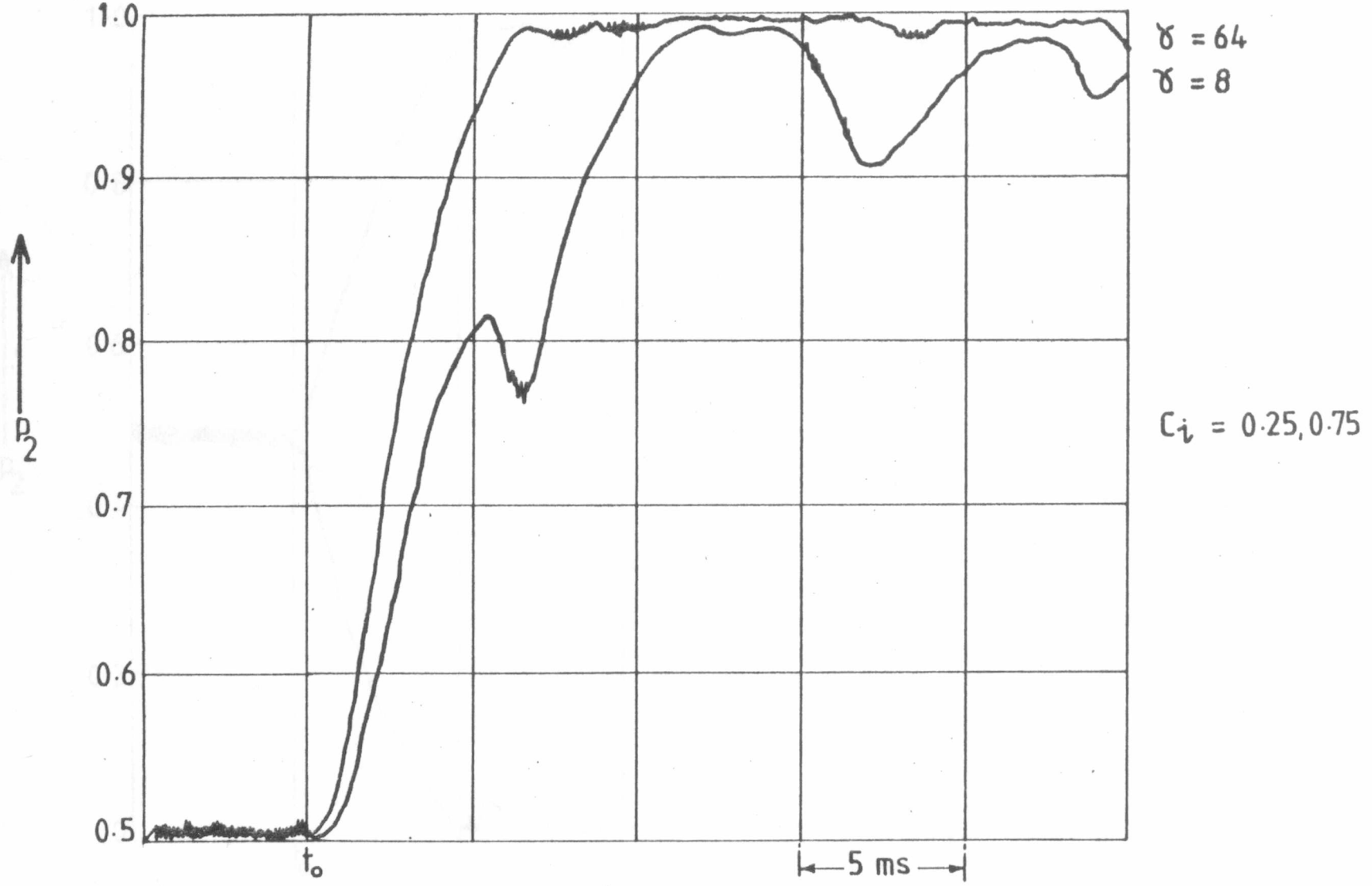


Figure 4.8 L_{R-R} scheme

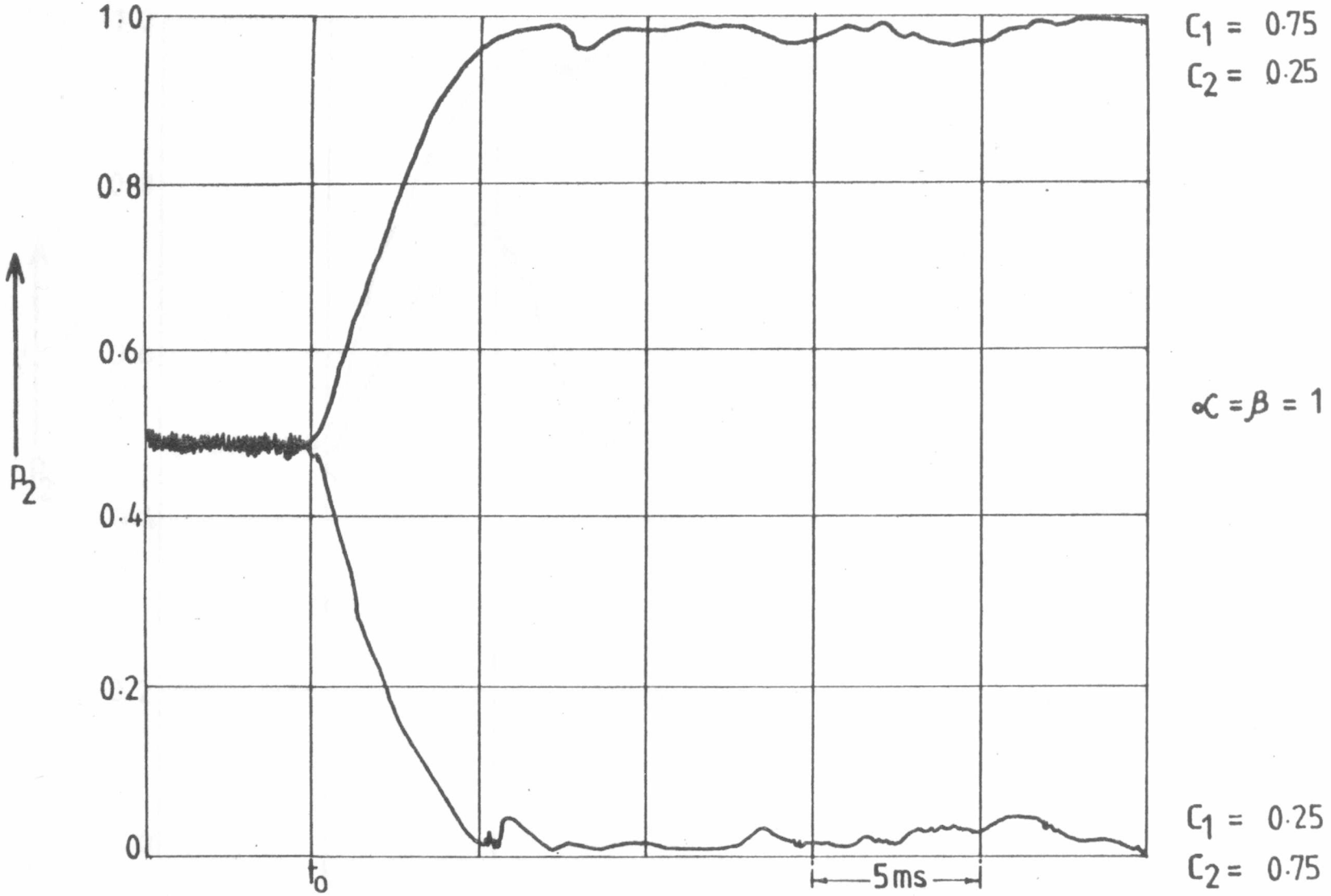


Figure 4.9 $N^{(1)}_{R-P}$ scheme (1)

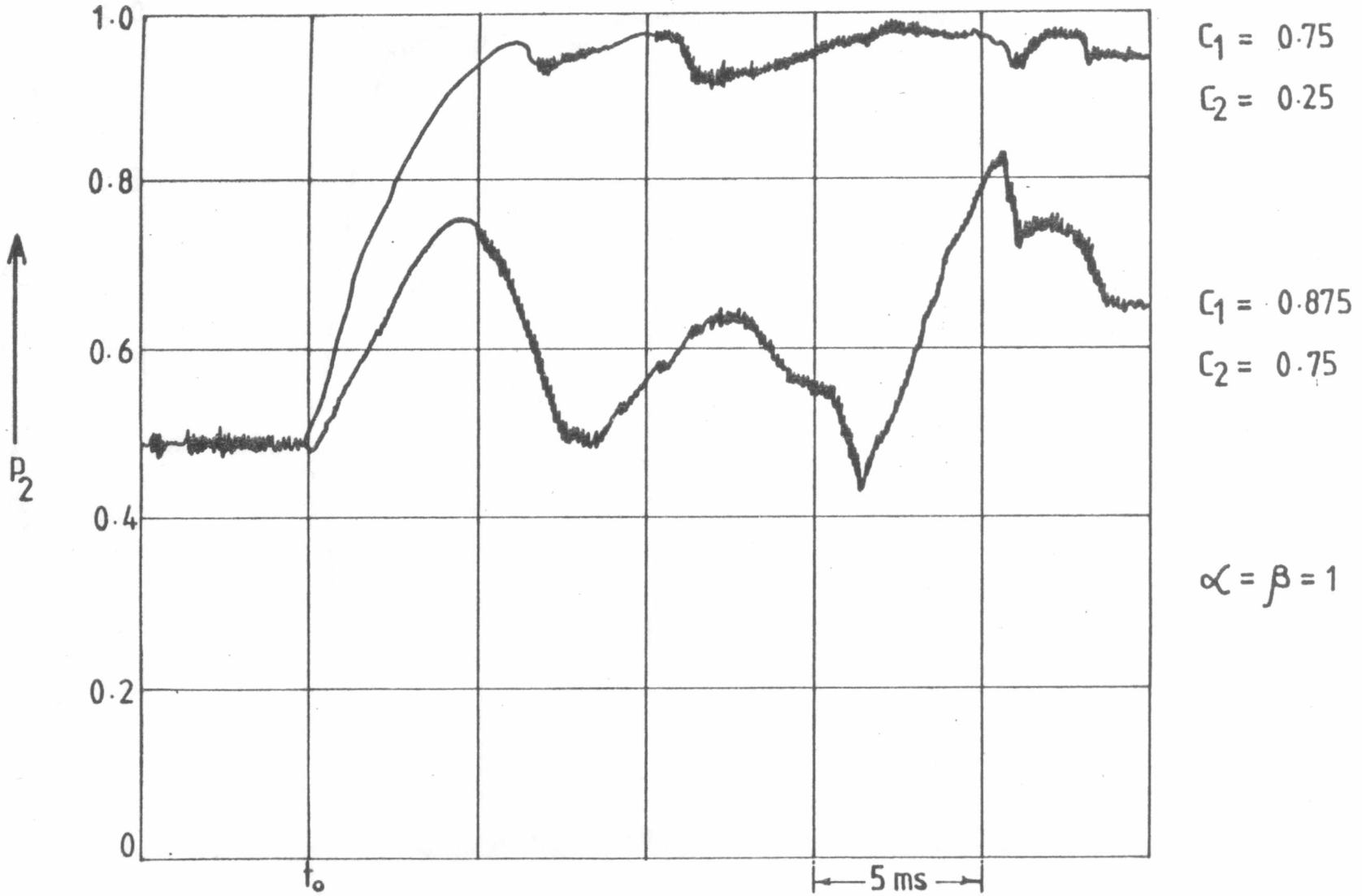


Figure 4·10 $N^{(1)}_{R-P}$ scheme, showing conditional convergence

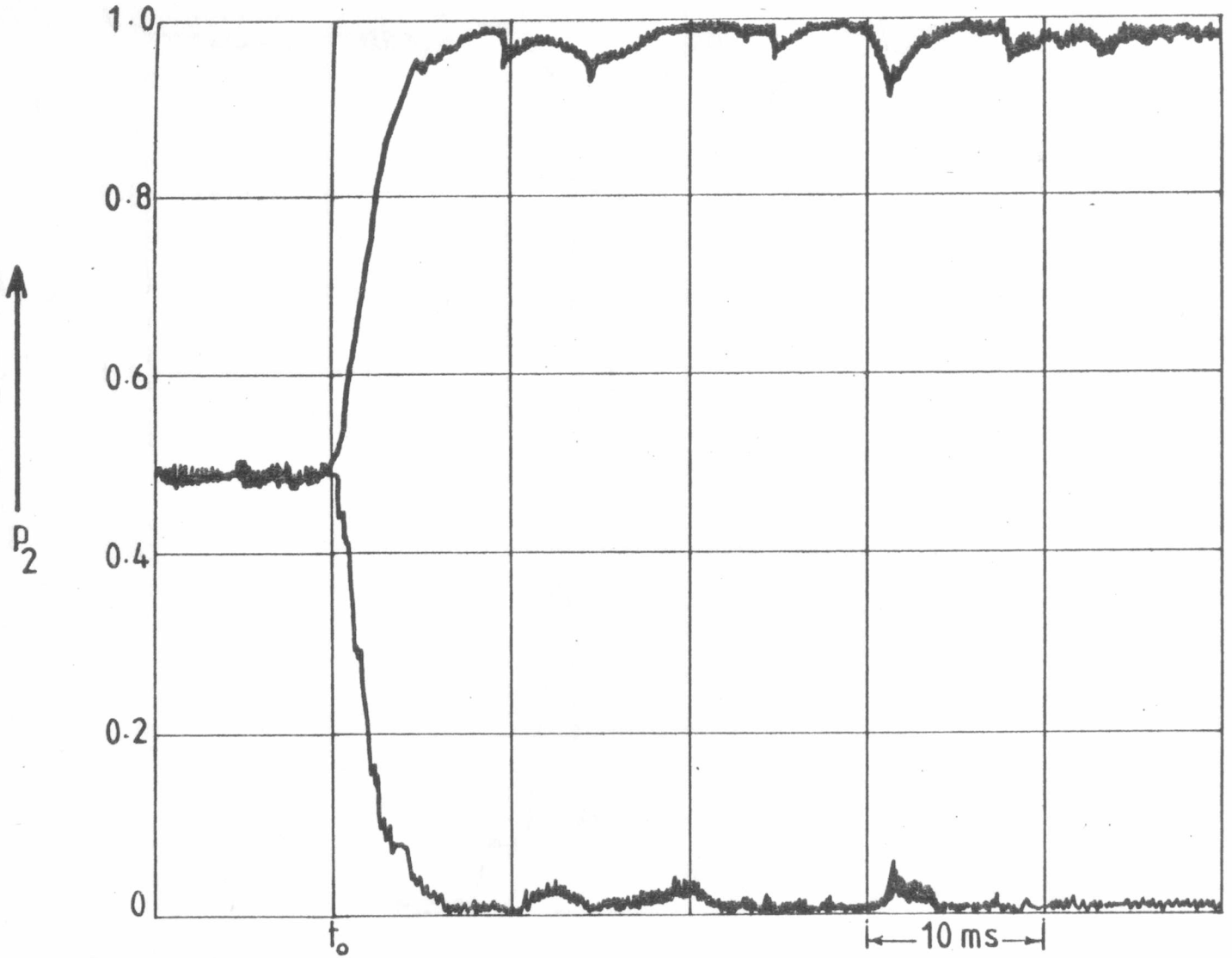


Figure 4·11 $N^{(1)}_{R-P}$ scheme (2)

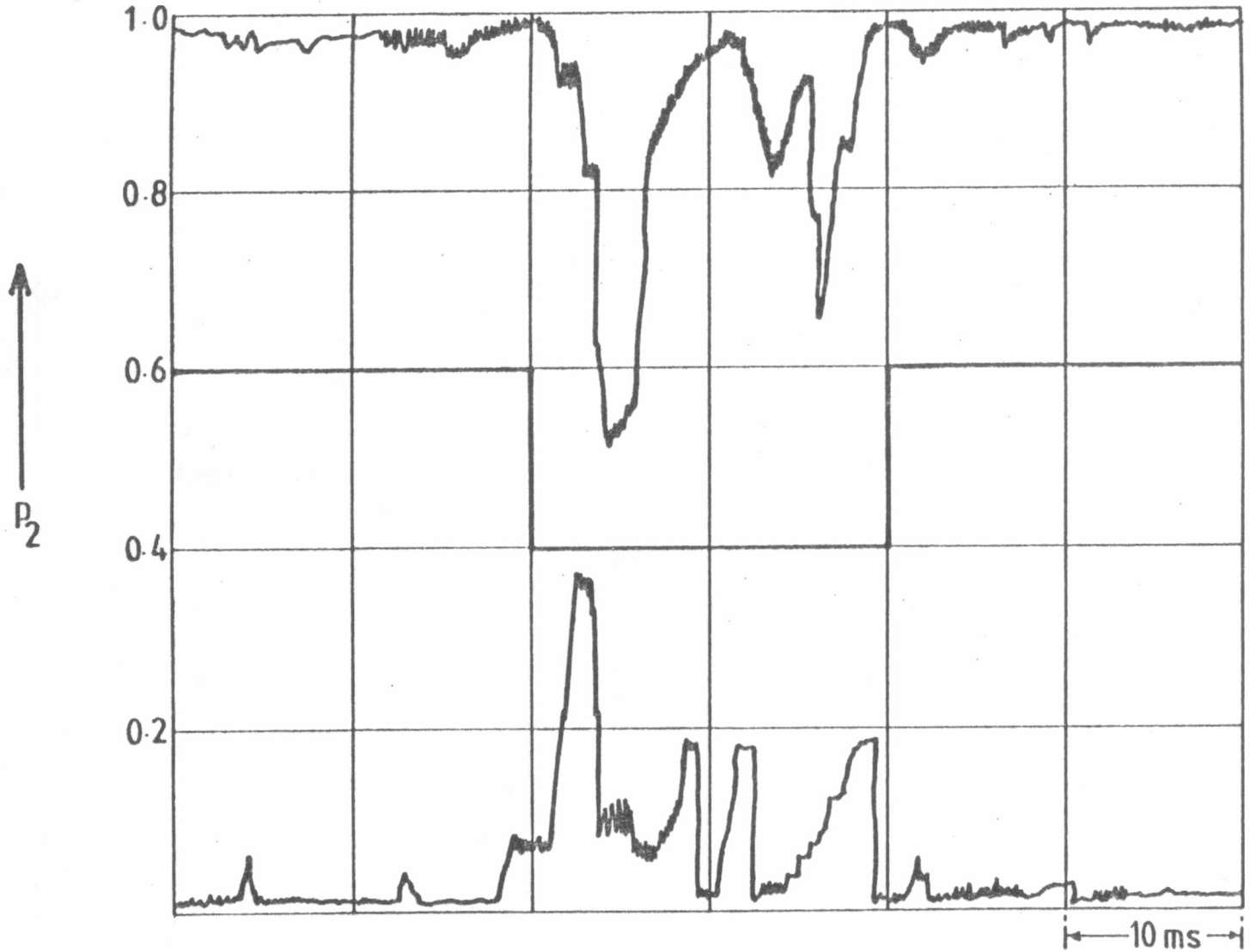


Figure 4·12 $N^{(1)}_{R-P}$ behaviour with switched conditions

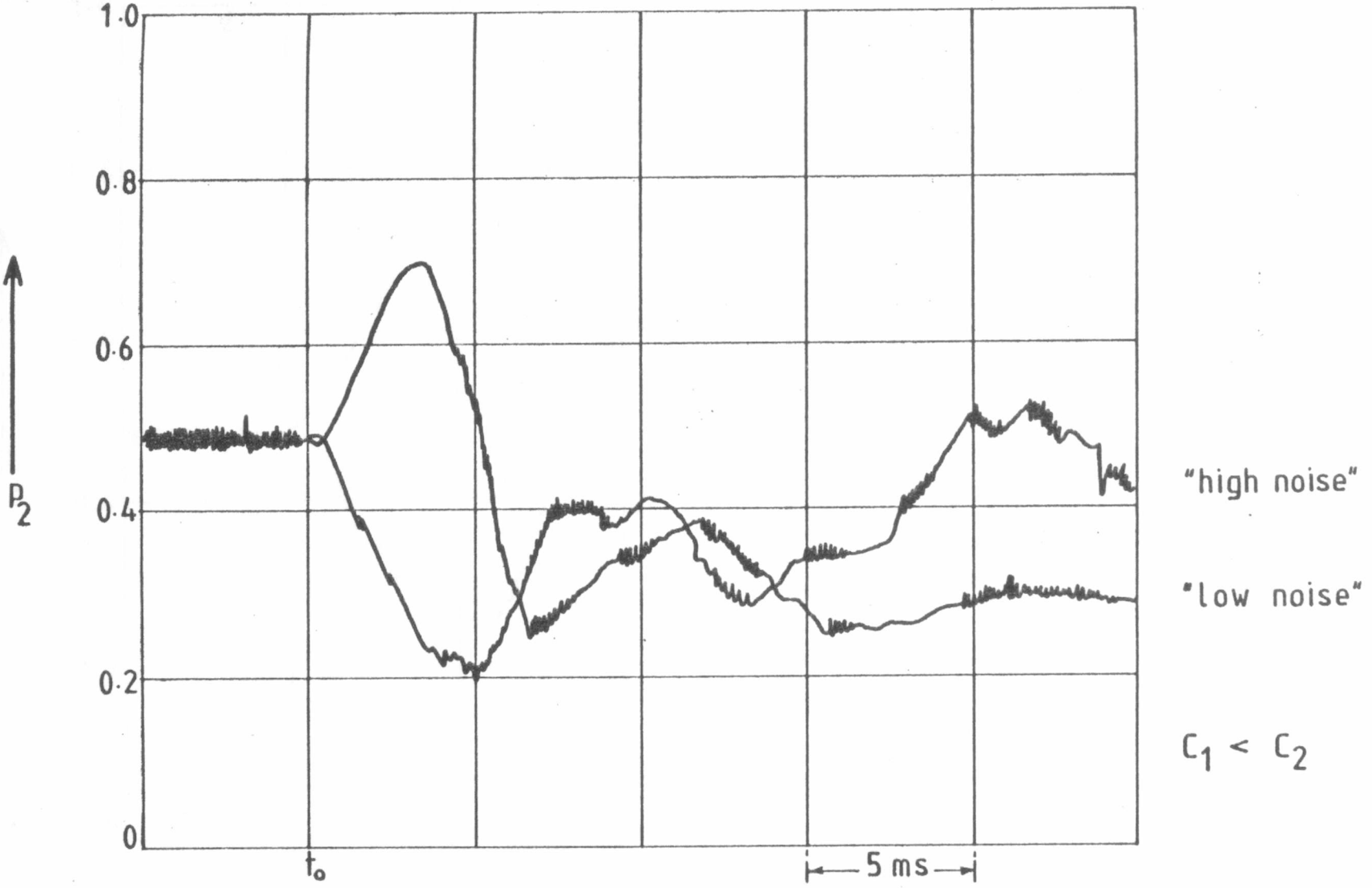


Figure 4·13 Plant simulator results : L_{R-I} scheme

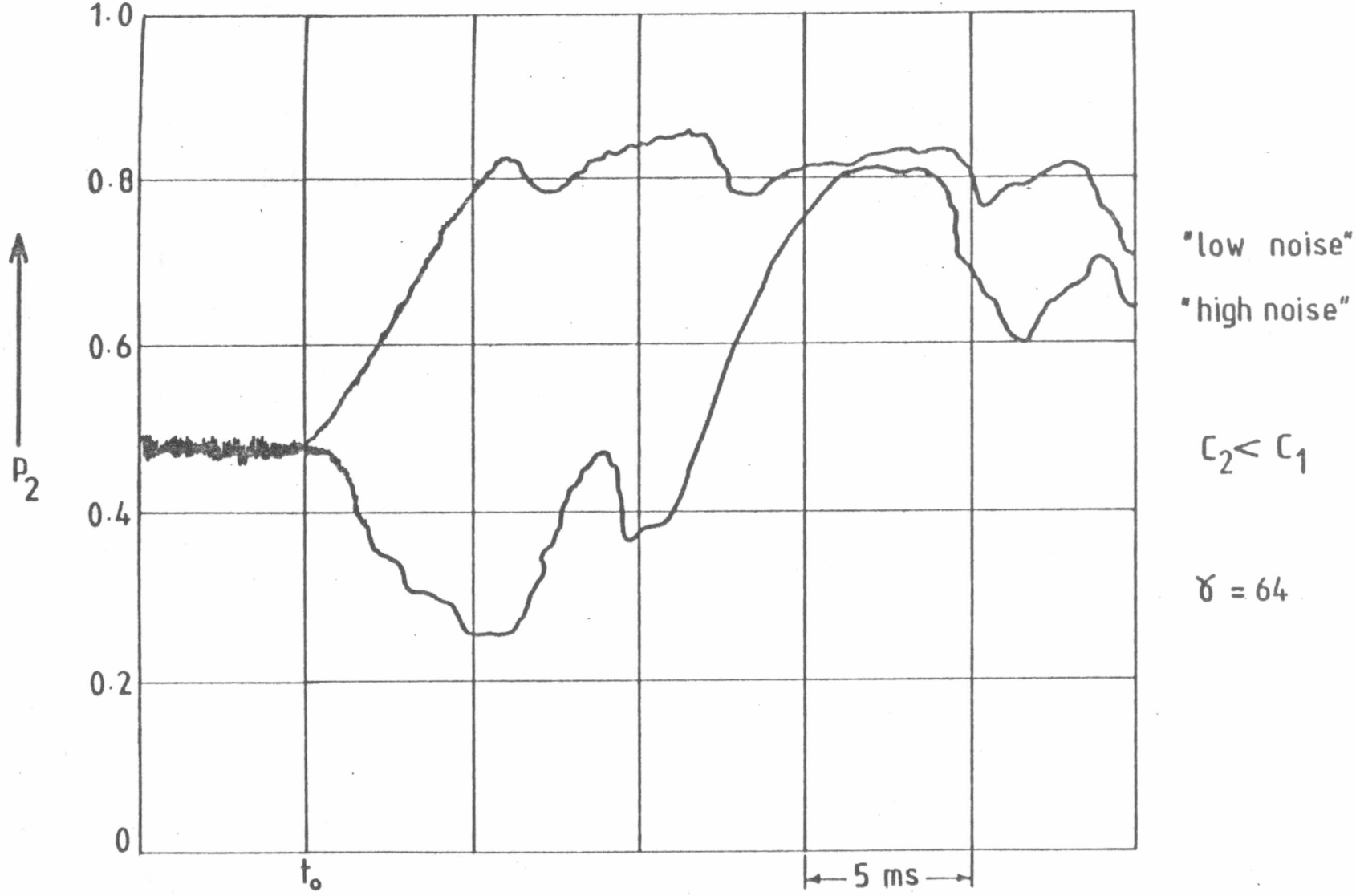


Figure 4.14 Plant simulator results: L_{R-R} scheme

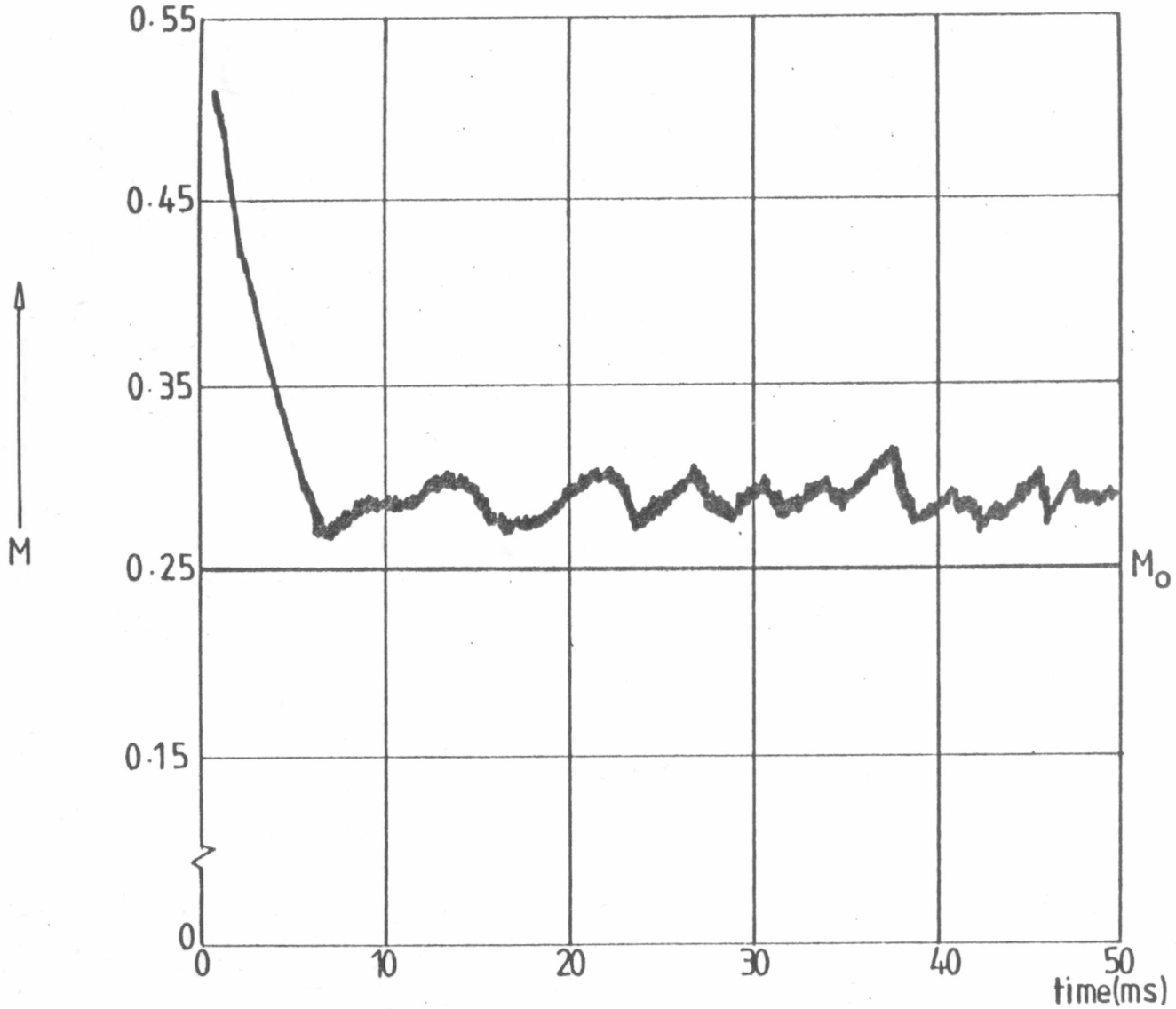


Figure 4.15 Performance curve: L_{R-P} scheme

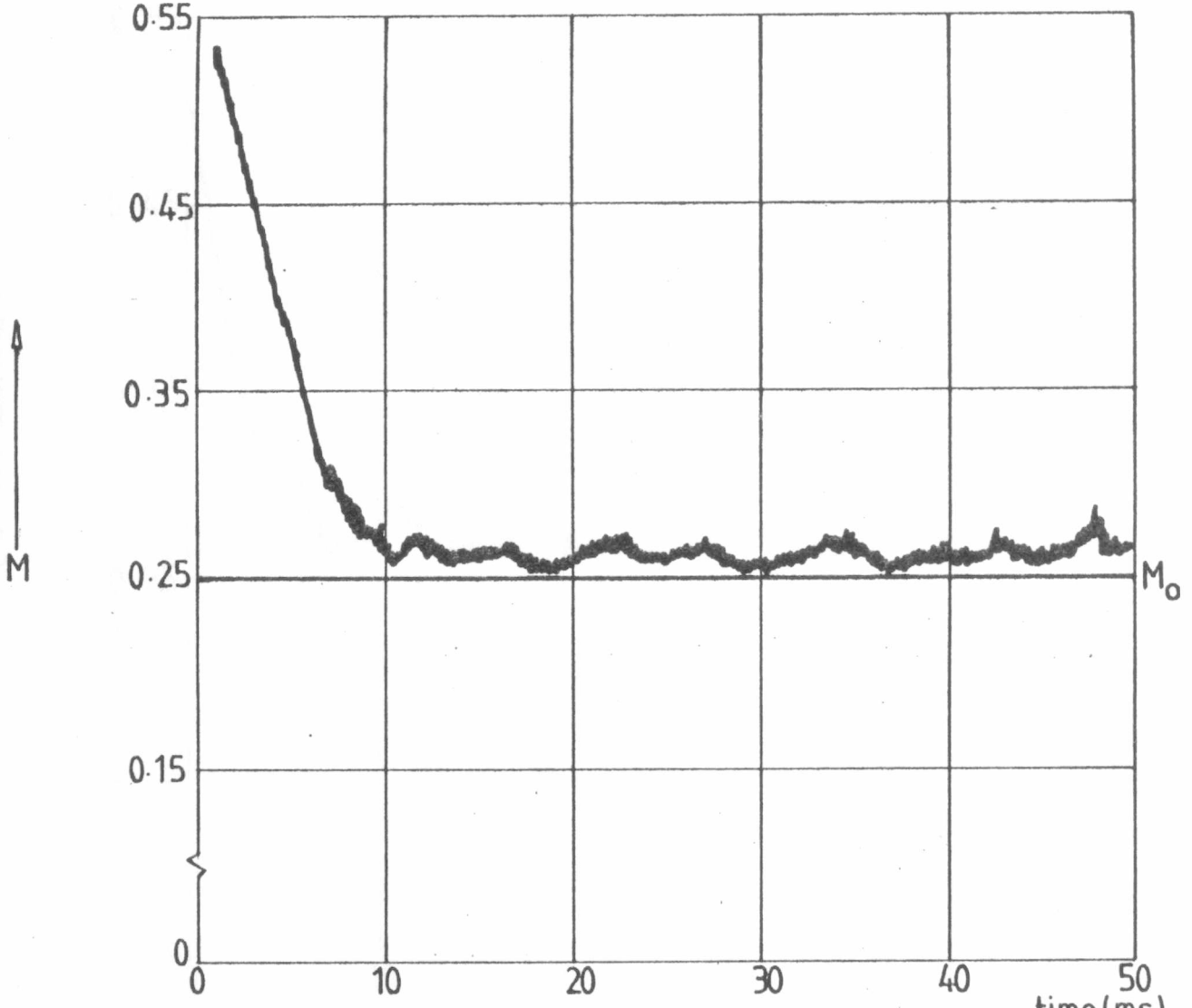


Figure 4·16 Performance curve: L_{R-I} scheme

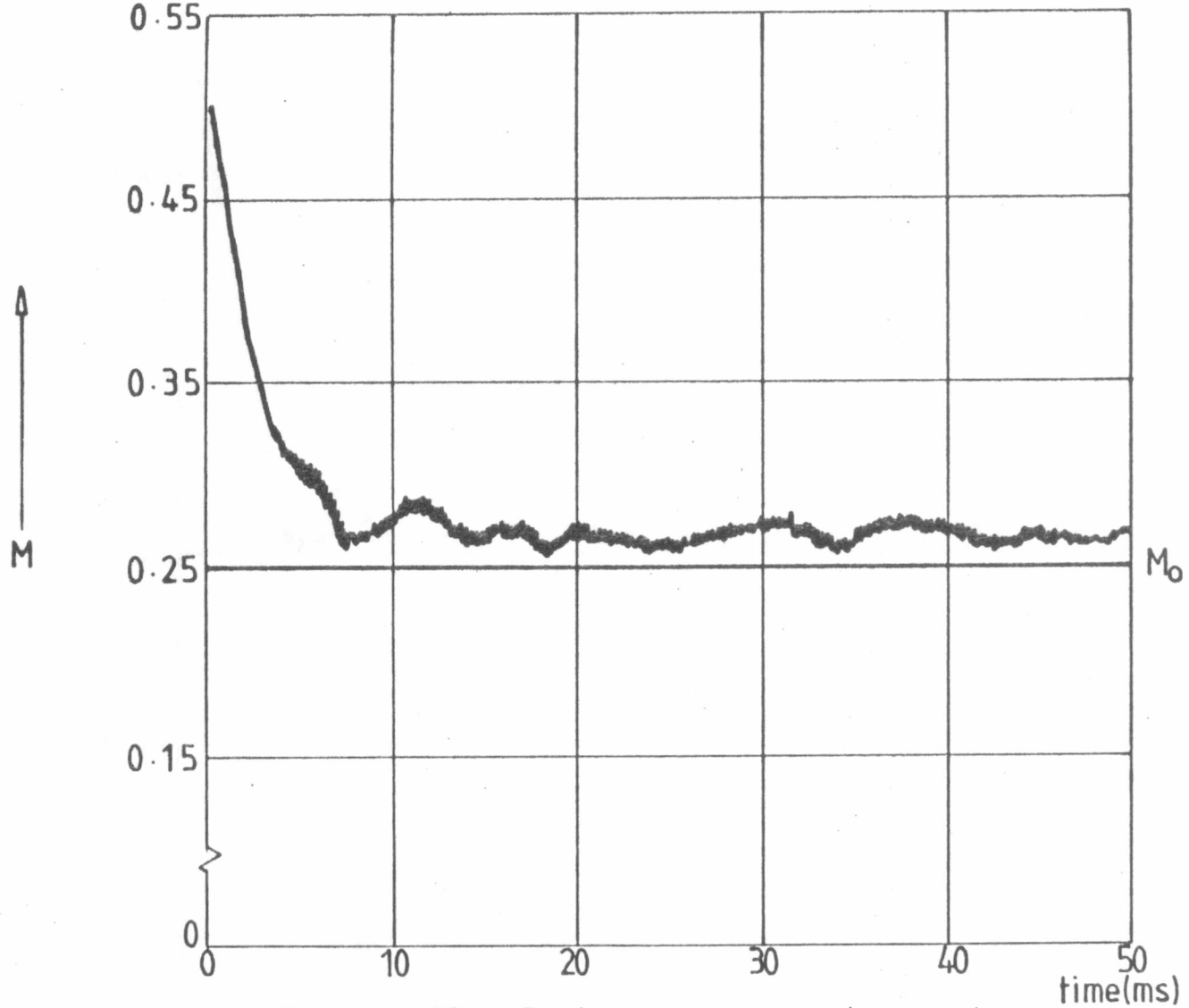


Figure 4·17 Performance curve: L_{R-R} scheme

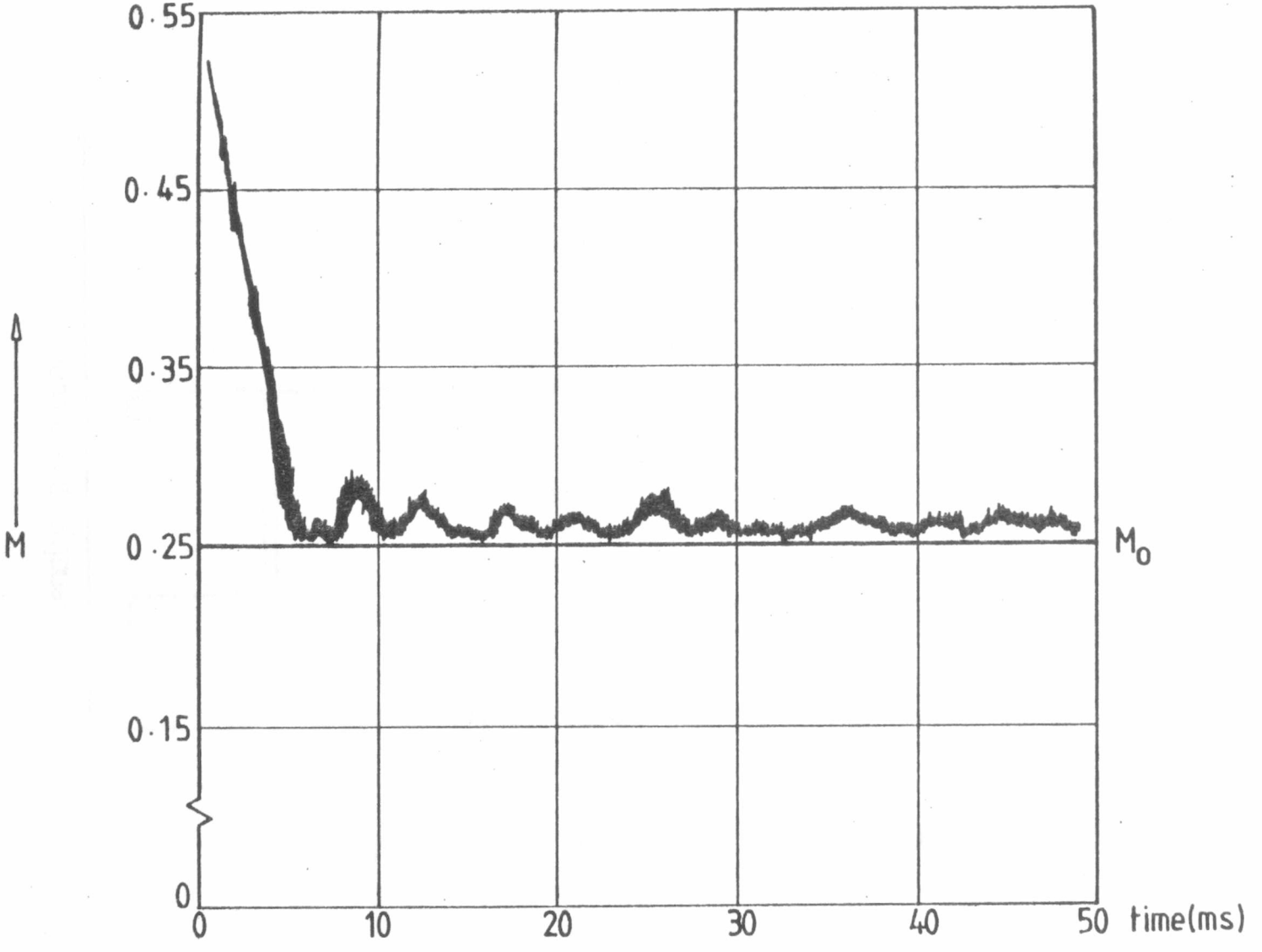


Figure 4·18 Performance curve: $N^{(1)}_{R-P}$ scheme

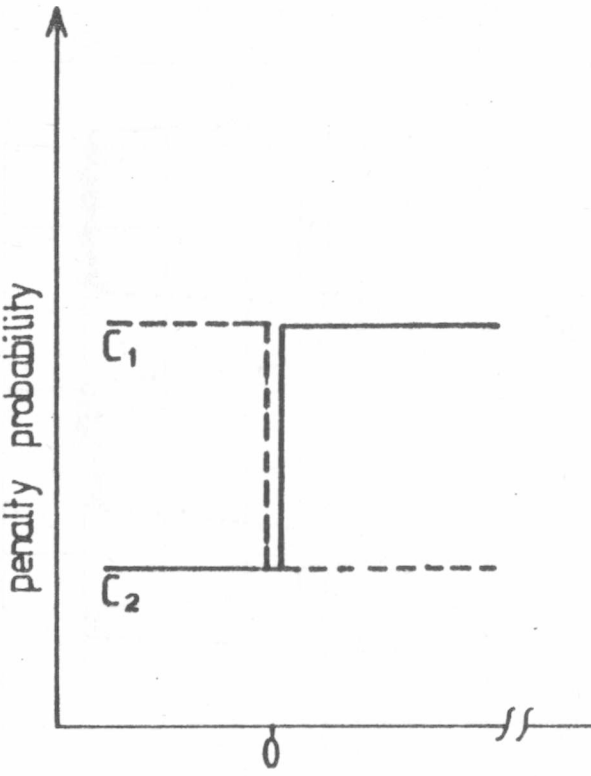
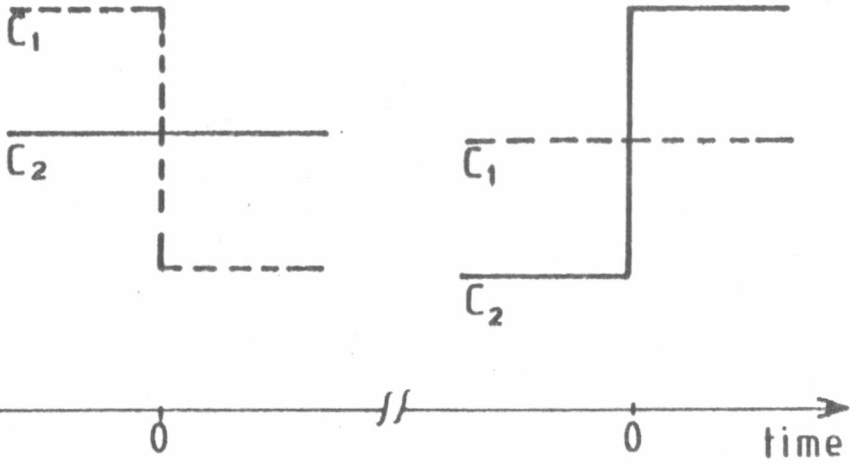
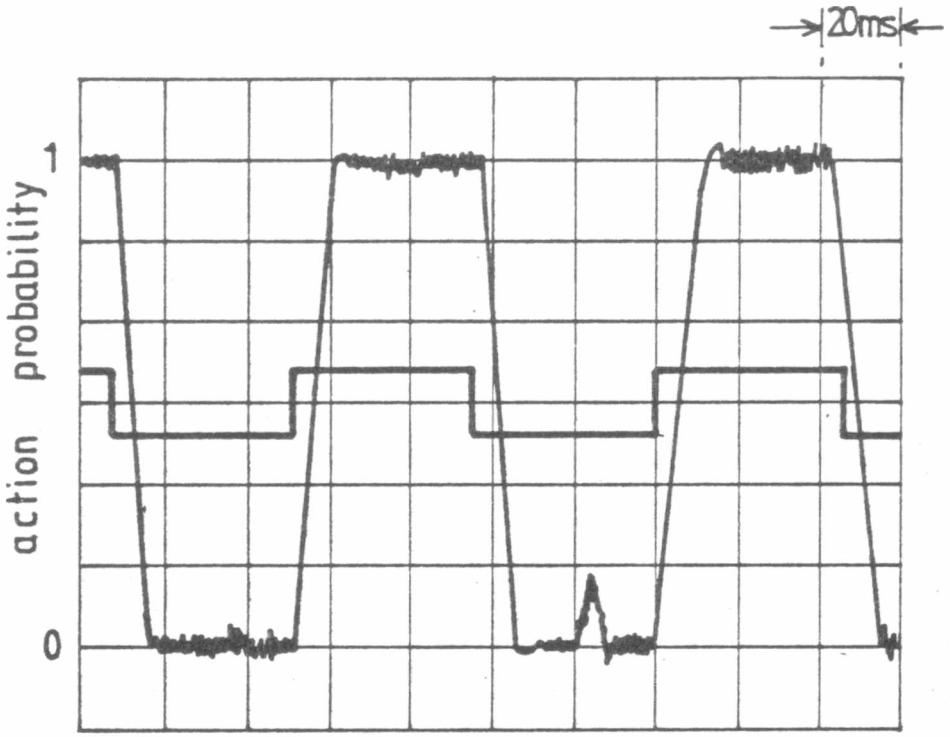


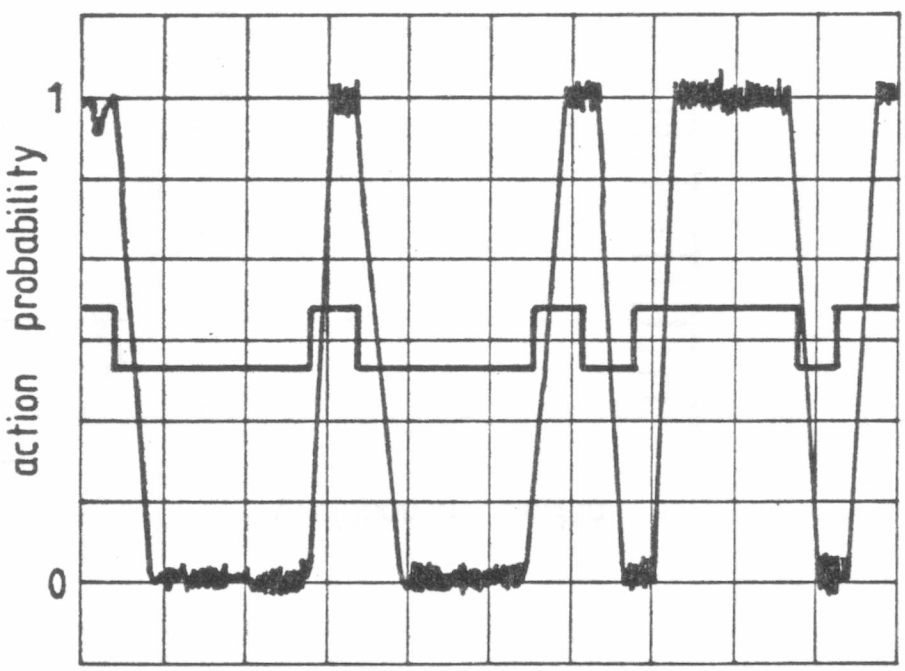
Figure 4.19



Step-wise switched environments



(a) Periodic switching



(b) Random switching

Figure 4.20 Switching behaviour of 2-state SLA

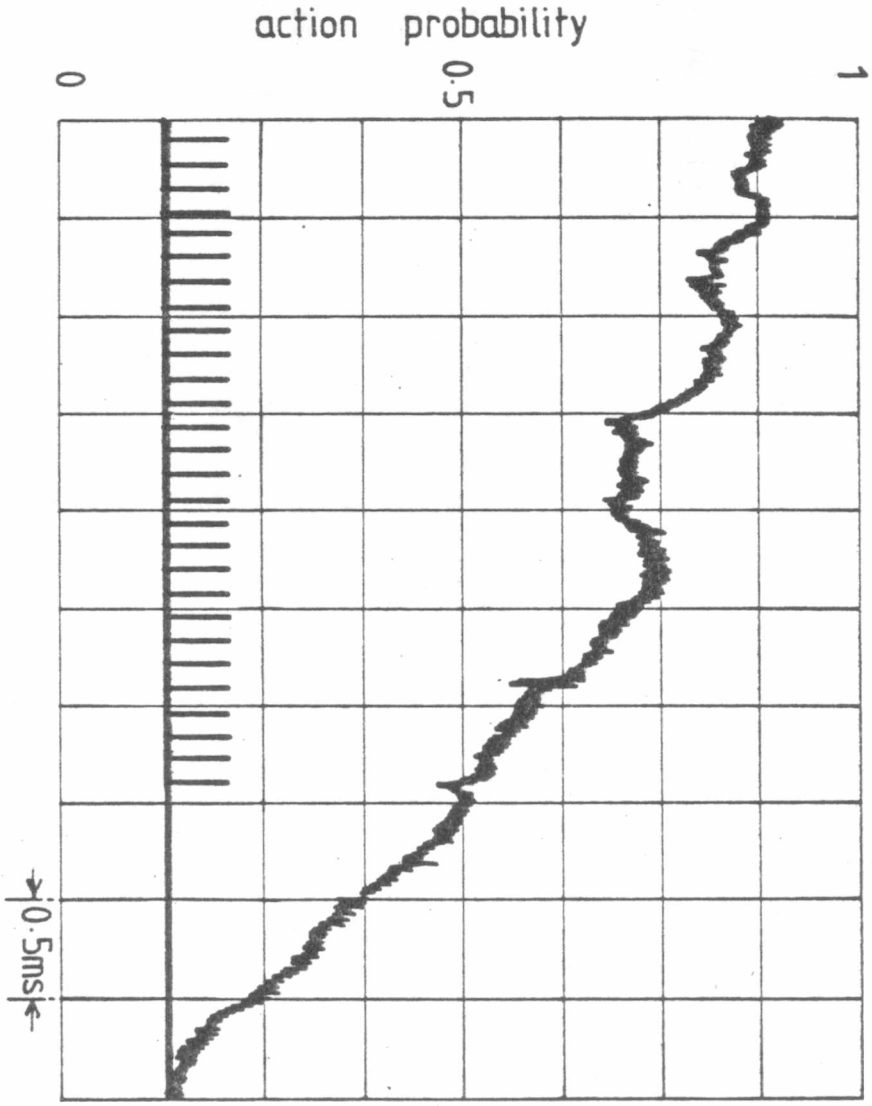


Figure 4.21 SLA transition curve

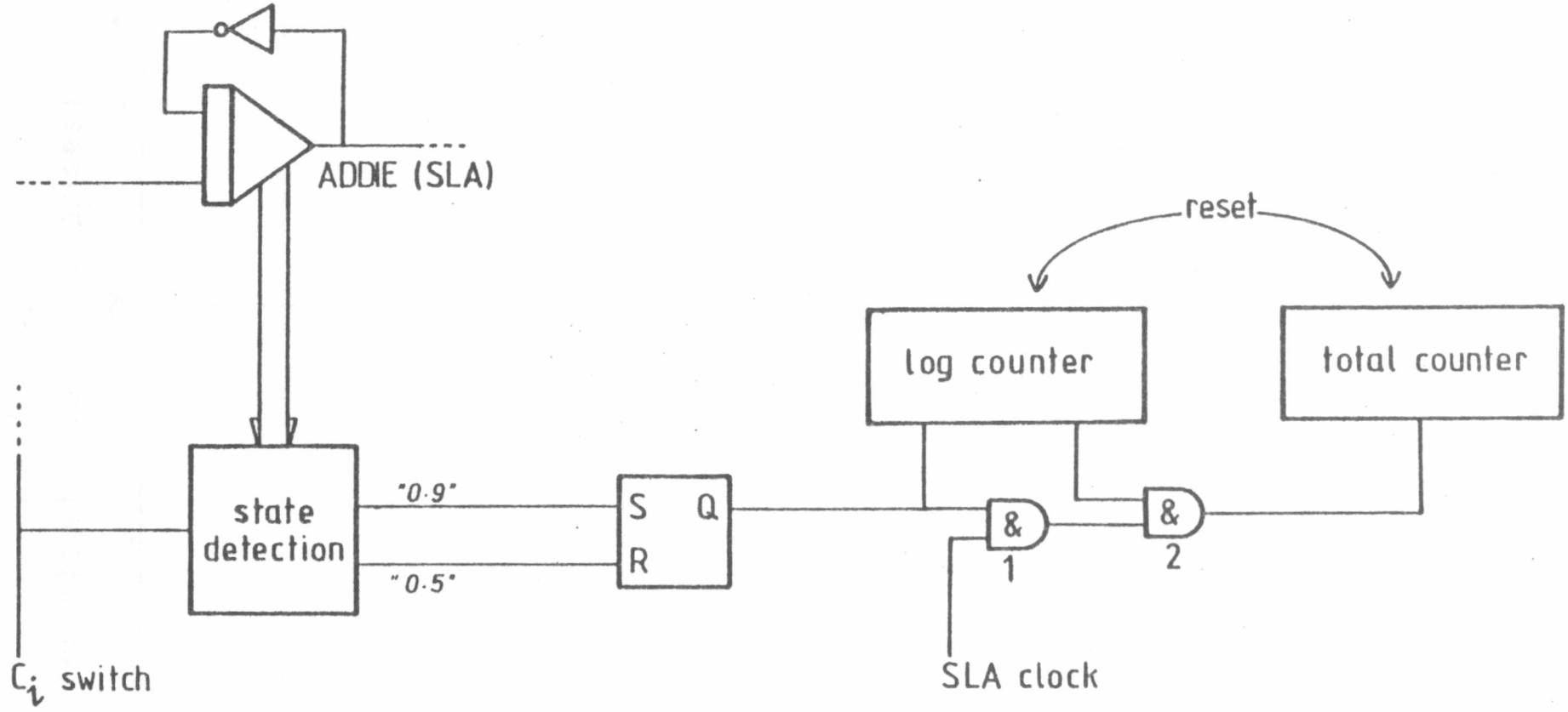


Figure 4.22 Automatic logging of mean adjustment time

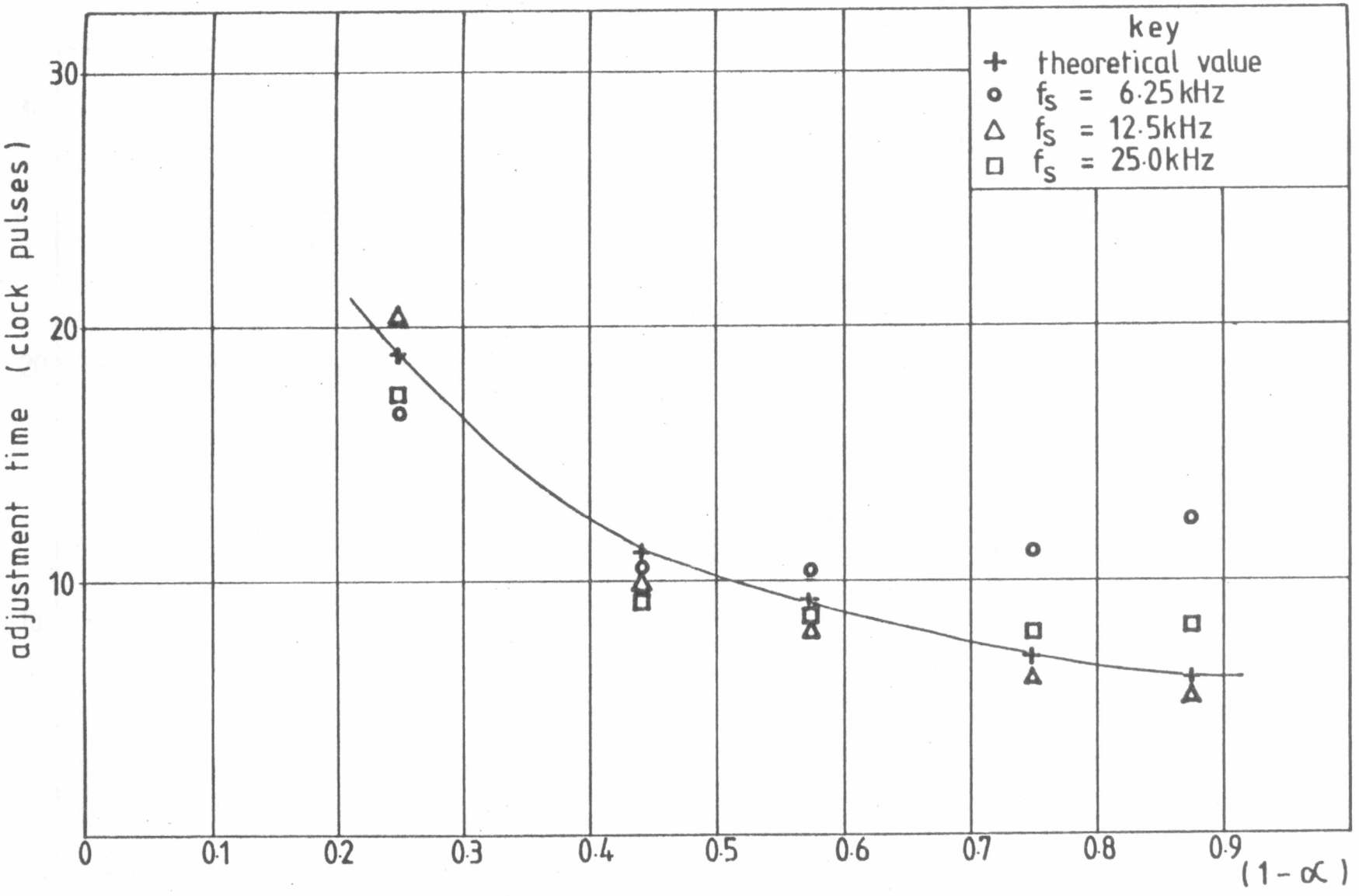


Figure 4·23 SLA adjustment characteristic : L_{R-I} scheme

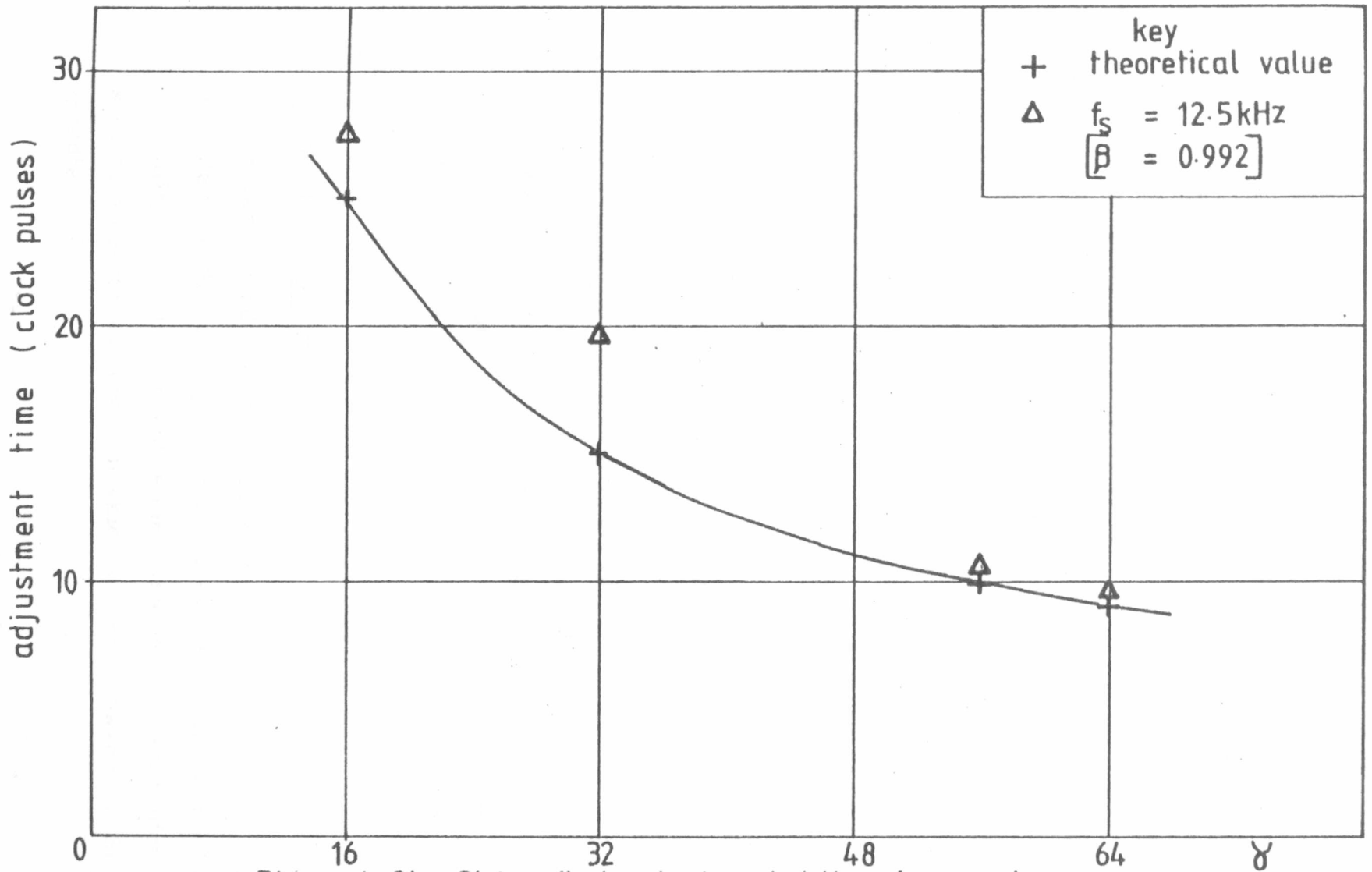


Figure 4·24 SLA adjustment characteristic: L_{R-P} scheme

CHAPTER 5

DEVELOPMENT OF A HIERARCHICAL STRUCTURE SLA

5.1 System Expansion

The results described in the foregoing chapters enabled useful experience of the performance of simple two-state automata to be built up. The next step was then to undertake the development of a much larger system, since it was felt that practical applications lay in that direction. Rather than simply consider extending the design to, say, a 10-state automaton, which has been investigated in previous simulation studies,[48, 49] it was decided to proceed with the implementation of a much larger system with state order $r > 100$. With this order of system size, there was a clear problem of minimising the amount of circuitry required, while at the same time preserving as much as possible the high operating speeds which are a feature of the two-state hardware systems.

It was decided that the most feasible solution to this problem would be to subdivide the state space and perform the random search between automaton states via a set of levels in a hierarchical structure. It has been suggested previously[5] that a multi-level design would be a suitable way of overcoming this very problem of high dimensionality, and the application of simple two-level systems has been considered.[65]

5.2 Hierarchical System

It was argued during the development of the ADDIE SLA (Chapter 3) that a large system must have a built-in memory capability. This concept is embodied in the hierarchical structure described here, in which a simple low-order SLA module /

module or "cell" is time-shared between each location in the "decision tree" and interfaced with a random access memory (RAM), as illustrated in Figure 5.1. Any one state or action probability is therefore represented by the product of decision probabilities at each node along the appropriate path through the tree.

The saving in hardware is immediately evident from the consideration that an m-level system based on a single r-state cell results in an automaton with r^m states. Each learning cycle, however, consists of m decisions between r states, i.e., a total of r x m decisions, rather than a single level decision between r^m states. This configuration does of necessity involve more serial processing operations, but the savings in hardware are felt to far outweigh the speed penalty. A further advantage is that a modular construction greatly simplifies the design requirements for even larger systems than those considered at present.

5.3 Design Evolution

It is recommended that the basic "cell" of the hierarchical system should have a binary multiple of state order, for suitable compatability with the organisation of standard random access memories. Clearly, the larger the cell, the fewer the levels which are required. For example, an 8-state cell in a three-level structure can cover 512 states, whereas a two-state cell needs nine levels to achieve the same coverage.

Several conflicting requirements arise when considering the size of the cell to be used. Small cells, such as two-state, give fast decisions at each level, but more levels are required for a given system size. On the other hand, /

hand, the control circuitry involved in timing, RAM addressing etc., has the least complexity.

In view of this, it is felt that the use of a two-state cell confers several advantages. In particular, the algorithm circuitry retains the standard 4-term format used previously with basic two-state systems, with its application time-shared now at each decision level. At a later stage, it may be found advantageous to modify the reinforcement scheme at different levels, or stages, during the learning process, to secure the best overall performance. Also, if each decision has just two outcomes, simple binary coding can be used to represent the decision paths through the hierarchical structure. This in turn renders the least complex design for the control or "housekeeping" circuitry.

The realisation of an automaton with a non-binary multiple of states is more complicated. One possible solution, yet to be tried, would be to insert a decoder between the SLA and the plant. The SLA would be organised with the next-highest binary number of states, with the decoder arranging for redundant states to be paired off with "active" states. This would cause some initial bias in learning behaviour, but the effect may not be too significant when averaged out over the whole learning period.

5.4 128-State System Design

The basic cell of the hierarchical system is the two-state ADDIE SLA described earlier. This single computing element effectively controls each decision at every node across and down the decision tree, with the ADDIE counter interfaced to a RAM to store intermediate decision probability values.

The /

The memory requirements are dictated by the number of levels used in the system. For each decision, the value stored in the counter at the relevant sampling instant represents, together with its implicit complement, the decision probabilities for each outcome. The first decision level thus requires one word of storage, the second level requires two words, the third, four words, and so on. Therefore, an r-state system requires a total RAM allocation of (r-1) x w bits, where w is the word length.

In order to preserve the highest possible operating speed, the transfer of information between the SLA cell and its memory should be a parallel process. This implies that the internal structure of the memory should be byte-oriented, with a common data bus interface. In the case of MOS devices, a good access time and TTL compatibility are also desirable. The device chosen initially to fulfil these requirements was the Motorola MCM 6810 AL, part of the "6800" microprocessor family. This is a 1 K static RAM with a 128 x 8 bit organisation, and is therefore directly suited for application in a 128-state system. Indeed, this device has available six "chip select" lines, giving ample scope for memory expansion if required. Subsequently, a TTL memory card was constructed with pin-for-pin compatibility, using an array of 74S200 type RAMs which are organised as 256 x 1 bit. As expected, these devices permitted a higher rate of data transfer.

The full configuration of the 128-state hierarchical system is shown schematically in Figure 5.2, with the corresponding seven-level decision tree in Figure 5.3. The two-state ADDIE SLA which forms the cell of the structure is essentially similar to that depicted in Figure 3.4. The main difference is that memory interface circuits /

circuits are now required, since the ADDIE no longer acts in a virtually continuous, self-contained cycle, but operates instead in a time-shared mode. A block diagram of this slightly modified system is shown in Figure 5.4.

The input and output buffers connected to the ADDIE are tri-state devices, designed for application with a common, bi-directional data bus. This considerably simplifies the design of the system, and of course reduces the "pin-out" requirements on the associated circuit cards.

5.5 Decision Path Control

As mentioned earlier, the use of a two-state cell enables simple binary coding to be used to track the decision path taken by the system in selecting an output action on each cycle. The means of achieving this are now described.

At each main sampling clock pulse to the cell, the resulting output from the flip-flop will be either 1 or 0, and this "decision bit" is stored in a 7-bit latch, each of whose locations corresponds to a particular decision level. As a result, at the end of a search through the decision tree, the contents of this "state latch" will define uniquely one of the 2^7 states, and also represent the output to be fed to the plant. When a second scan is made along the same path to apply the reinforcement scheme in accordance with the plant response and update each of the decision probabilities, the state latch effectively provides a "map" of the route to be taken through the decision tree. That is, one machine cycle must involve two traversals of the decision tree, the first in the nature of a "random walk", the second in turn guided by the outcome of the first. The formation of the state latch contents is simply accomplished by using a ring counter as a commutator to steer each decision bit to its correct /

correct location, as shown in Figure 5.5.

5.6 Memory Address

A central feature of the hierarchical system is the memory address procedure. The counter in the ADDIE SLA must be interfaced correctly to the memory location corresponding to its current position in the decision tree at every stage.

Consider an m-level system, with the memory partitioned as follows:

Decision level : 1, 2, 3, - - - - - m
Rows of memory : 1, 2, 4, - - - - - 2^{m-1}

When the cell is at level d , say, it will have made d - 1 decisions. The state latch therefore will contain d - 1 bits, which are sufficient to form all address codes required for the number of rows of memory at that level. The address code is derived accordingly from the state latch, itself in effect a small "scratch-pad" memory tracking the decision path.

The details of the addressing method are illustrated in Figure 5.6, using a 4-bit example for clarity (i.e., a 16-state system). In the table shown, the position of the circulating bit from the ring counter is marked with an asterisk, while state latch contents at each level are enclosed in dotted lines. At the upper levels, there are insufficient information bits from the state latch to assemble the requisite m-bit address word. The address words are therefore completed with the ring counter bit plus a string of zeroes as appropriate. The resulting progression of address codes can be seen to assume a well-ordered BCD type of layout, illustrating how easily the design philosophy can be extended to any desired number of levels.

Address /

Address words are formed, therefore, by suitably combining the ring counter output and state latch contents, with blocking inverters arranged to generate the necessary zeroes. The circuit configuration for this is shown in Figure 5.7. It can be seen that this arrangement also allows for the address lines to be cycled from an external counter to facilitate the loading of 0.5 into each memory location as an initial condition. The operation of this address circuit can be verified by considering the example illustrated in Figure 5.8. This shows the build-up of address codes at successive levels during one cycle. The procedure is identical for both "search" and "reinforcement" phases of operation. The ring counter is clearly an important component in the control circuitry, since it supervises both the loading of the state latch and the formation of the memory address.

5.7 System Clocks

Overall control of the sequence of operations which comprises one complete iteration of the hierarchical automaton is achieved by means of a multi-phase clock generator. Since each cycle consists of an ordered, repetitive sequence of events, it was decided to base the design on a ring counter, each of whose cells would act as a source of clock pulses for one particular operations. A logic array on the ring counter outputs enables certain portions of each waveform to be blanked off as appropriate. By coincidence, it turned out that a seven stage counter was called for, matching the seven-bit commutator described earlier. One of the functions of the primary ring counter is of course to clock the commutator; at the same time, the first and last commutator bits are used for certain masking and reset operations on the primary clock generator. /

generator. The two ring counters can therefore be considered as a nested pair (Figure 5.9).

The full timing diagram for the 128-state hierarchical system is illustrated in Figure 5.10. For convenience, each waveform has been assigned a mnemonic. In the detailed description of system operation which follows, these will be explained as necessary, together with references to appropriate elements of the circuitry.

5.8 Operating Sequence

The master clock, denoted MC, is the central source for all the clock waveforms, and all events are therefore synchronised to it. While synchronisation is primarily required for the sequential processing operations, it is also necessary, as explained earlier, to apply it to all associated noise sources. The same master clock is therefore used to supply the four PRBS generators used in the design.

SLA operation is initiated by a reset pulse (RST) which may be automatically, manually, or "power-on" activated. This clears all registers, latches and counters. The next phase, which occurs just once in the learning process, is the establishment of the initial conditions. For this operation, the memory read/write line (R/W) is set to "write" mode, and the data bus select line (DBS (2)) arranged to feed the "one all-zeroes" initial condition onto the data bus. Figure 5.11 shows the principal elements of the data bus interface. The initial condition counter then starts to count up, cycling through all the address codes as explained earlier. When the final carry or overflow bit (ICCOB) goes high, the counter stops, the primary ring counter is enabled, and the main multi-phase clock sequence commences. The output of a flip-flop, denoted "sequence flag" (SF) then /

then goes low, indicating that the automaton has entered the "search" phase.

During this phase, the memory is held in "read" mode, and the secondary ring counter is clocked (COMCK). This sets up the first level address and prepares for steering the decision bit to the first cell of the state latch. The current or "old" value of $p_i(n)$ held in memory is clocked into a latch (OPLCK), and a decision bit is obtained by sampling with the associated system flip-flop (SFFCK). This sequence is performed seven times in the case of a 128-state system. The state latch contents are then loaded into a system state output latch (SSOLCK), which acts as the digital output interface, and also a parallel-in serial-out (PISO) register from which the decision bits will be recovered in correct sequence for the operation of the algorithm circuit (see below). Figure 5.12 illustrates the components involved in this data transfer.

This single clock pulse (SSOLCK) also sets a flip-flop whose output (CKSTP) effectively "freezes" the primary ring counter (Figure 5.11) by maintaining a high logic level on one of its feedback lines. This facility allows the automaton to wait for the plant response, which must be assumed to be totally outwith the control of the automaton clock. The plant response is assumed to include a signal pulse (PLTRDY) which resets this flip-flop and allows the cycle to continue. If a simulated plant with essentially "instant" response is used, however, this facility is not required and the CKSTP signal can be held low all the time.

The system is now ready to enter the reinforcement phase. The first commutator clock pulse changes the state of the sequence flag, thereby enabling the appropriate clock signals for this phase. The plant response is sampled by /

by the reward/penalty flip-flop (PRFFCK), and the memory contents at the first level address are read into the ADDIE (DBS (1)) and the latch (OPLCK). Another holding operation on the primary ring counter is then initiated, using a counter denoted "learning period timer". As before, a high logic level is fed back until the most significant bit ($LPTQ_D$) changes state. In the interval, the ADDIE adjusts to the updated value of $p_1(n+1)$ supplied by the algorithm circuit. The address for the algorithm data selector consists of the output from the reward/penalty flip-flop, and successive decision bits clocked out from the PISO register, mentioned earlier, using the DBS (1) clock signal. The memory is then switched to "write" mode, and the ADDIE output buffer control line (DBC (2)) switched accordongly, to allow the updated ADDIE contents to be dumped in the memory. After seven such operations, one system iteration is completed. The automaton is then ready to re-enter the search phase and repeat the process. If at any time the system is reset, the memory will be "scrambled" by the initial condition phase and the whole learning process will start over again.

The learning experience of the automaton is represented by the reinforcement of the path leading to the optimum state as the decision probabilities at each node on the path approach unity. The chief determinant in the cycle time, again setting aside consideration of the plant time constant, is the ADDIE adjustment period during the reinforcement phase. Since seven updating steps occur in each iteration, it can be provisionally estimated that the learning time for the 128-state system will be at least seven times that of the two-state ADDIE SLA with its single updating step.

5.9 /

5.9 Variable Size Facility

Since the last bit of the commutator controls the length of the primary clock sequence, it is possible to alter the effective size of the SLA simply by moving the tap-offs on the secondary ring counter and masking appropriate feedback lines, as shown in Figure 5.13. In this way, the hierarchical system can be set up with 7, 6, 5 or 4 levels, i.e., 128, 64, 32 or 16 states. A set of data selectors is used to facilitate switching between the various combinations of control signals, so that just two external address lines require patching-up by the operator.

5.10 Construction

The complete system was assembled on a total of ten circuit cards, each representing a distinct function: clock generator, state latch and address, memory (MOS or TTL), ADDIE (8 or 12 bit), data latch, noise (1), noise (2), algorithm, output interface, display functions. A cabinet was constructed providing all the requisite input/output and control facilities for system operation on the front panel. Photographs of the unit are reproduced in Figures 5.14 and 5.15.

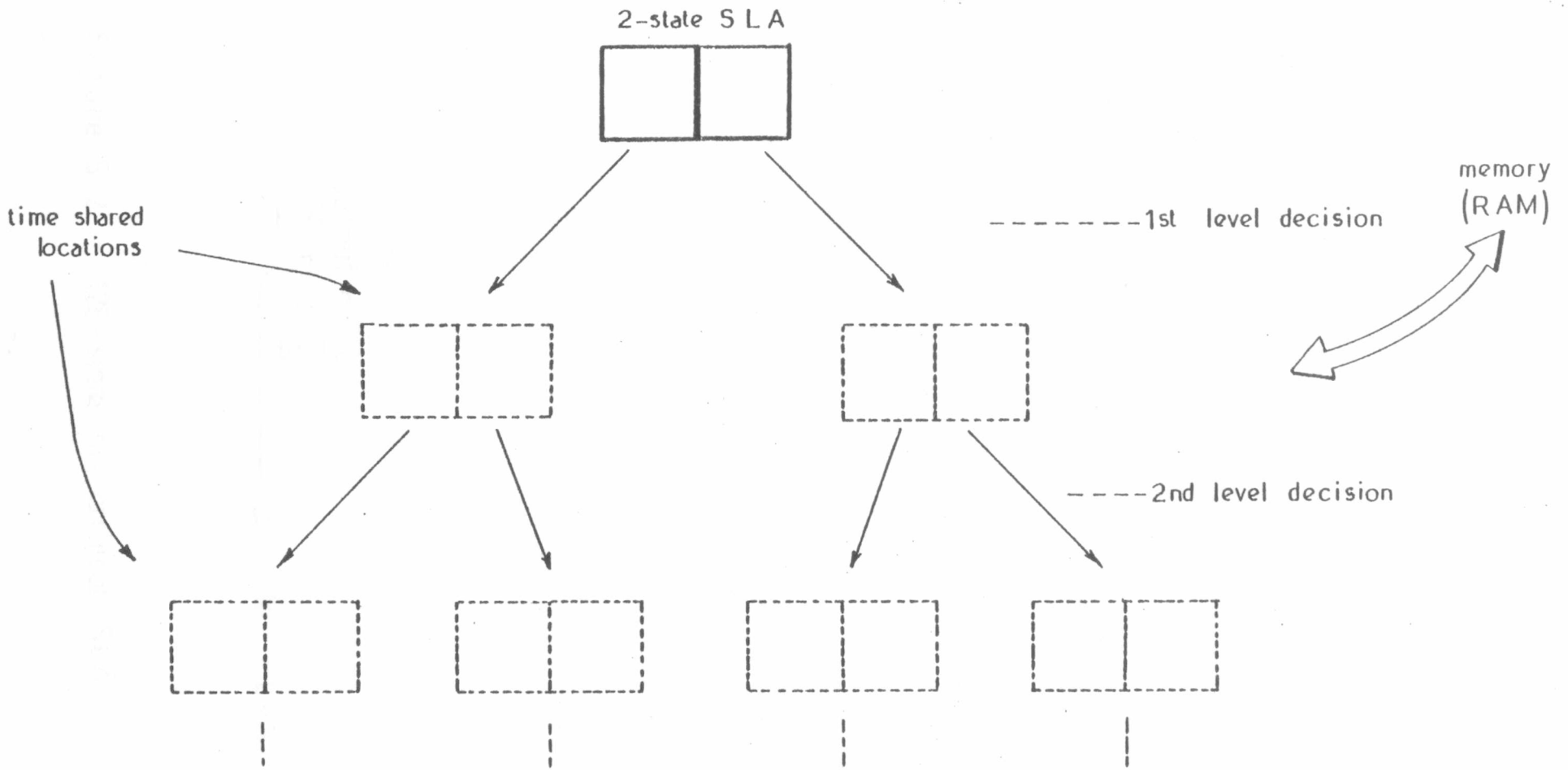


Figure 5.1 Hierarchical structure with 2-state cell

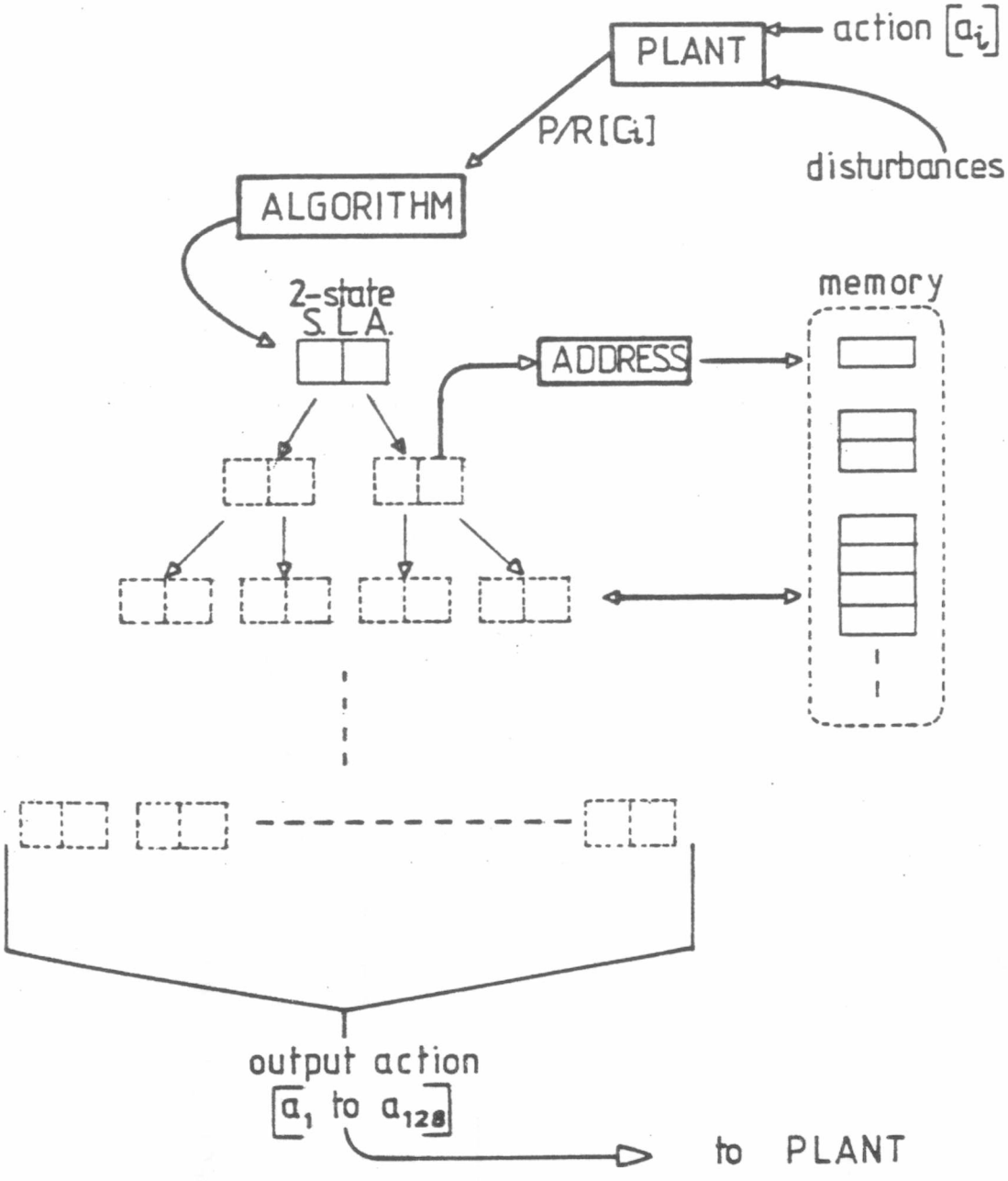


Figure 5.2 128-state hierarchical SLA

DECISION 'TREE'

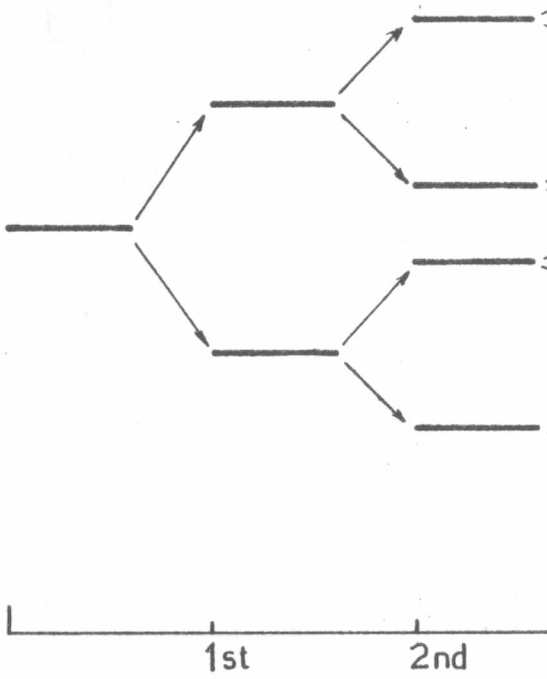
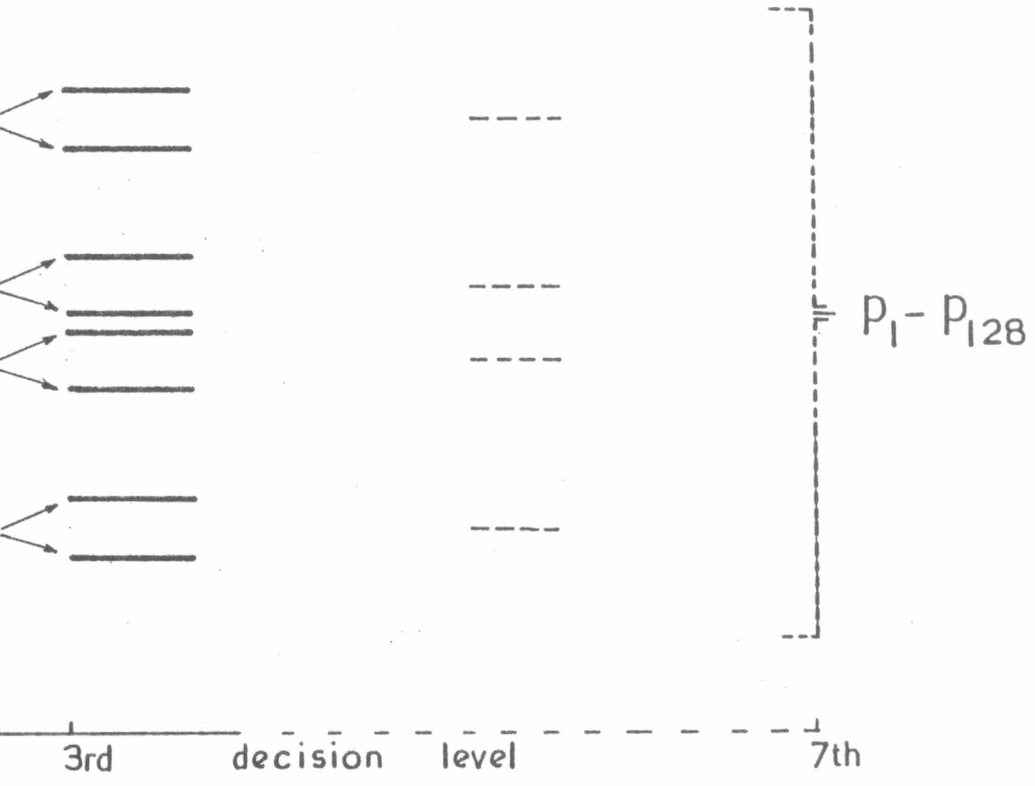


Figure 5.3



7 - level decision tree

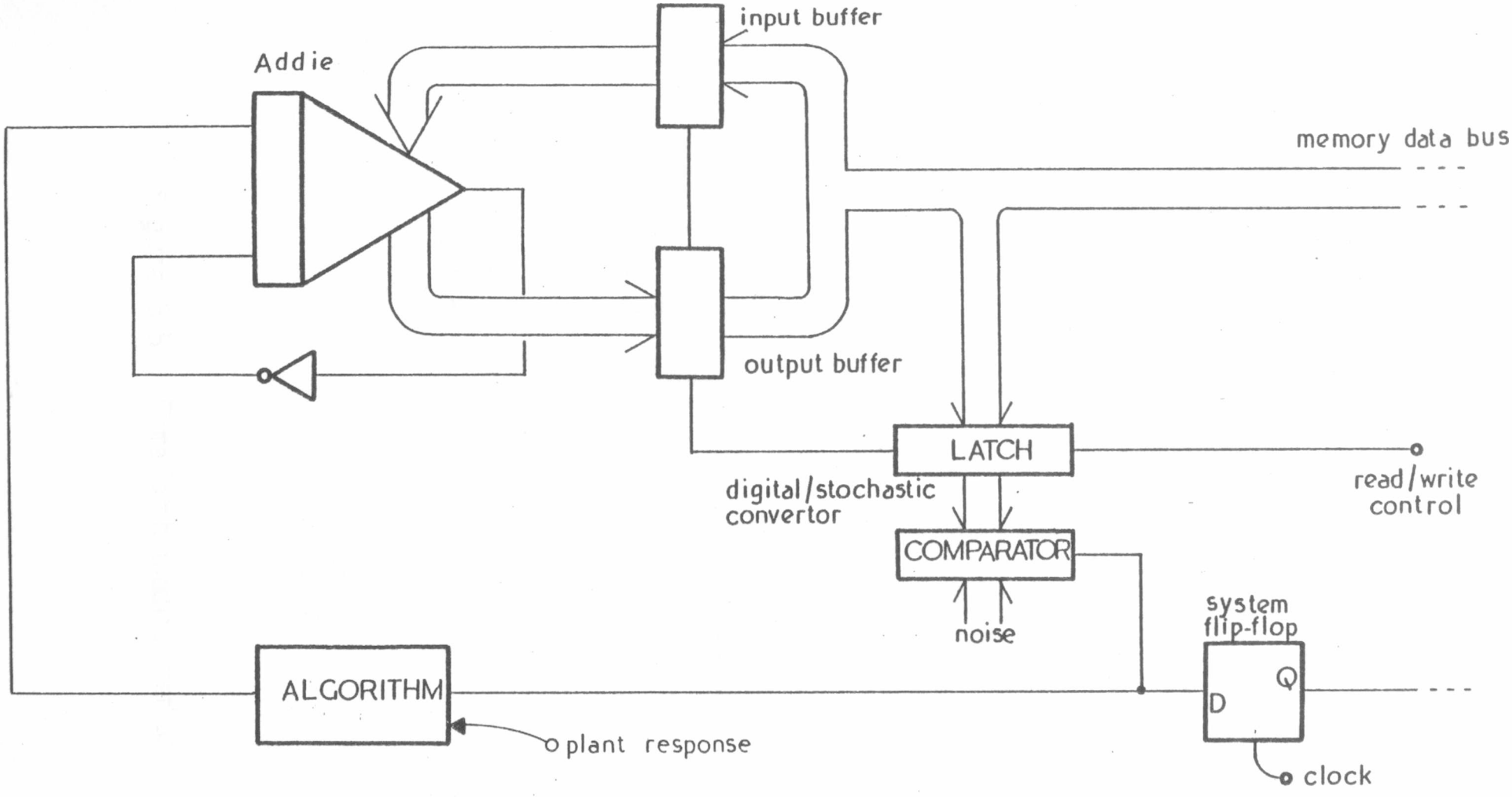


Figure 5.4 ADDIE SLA with memory interface

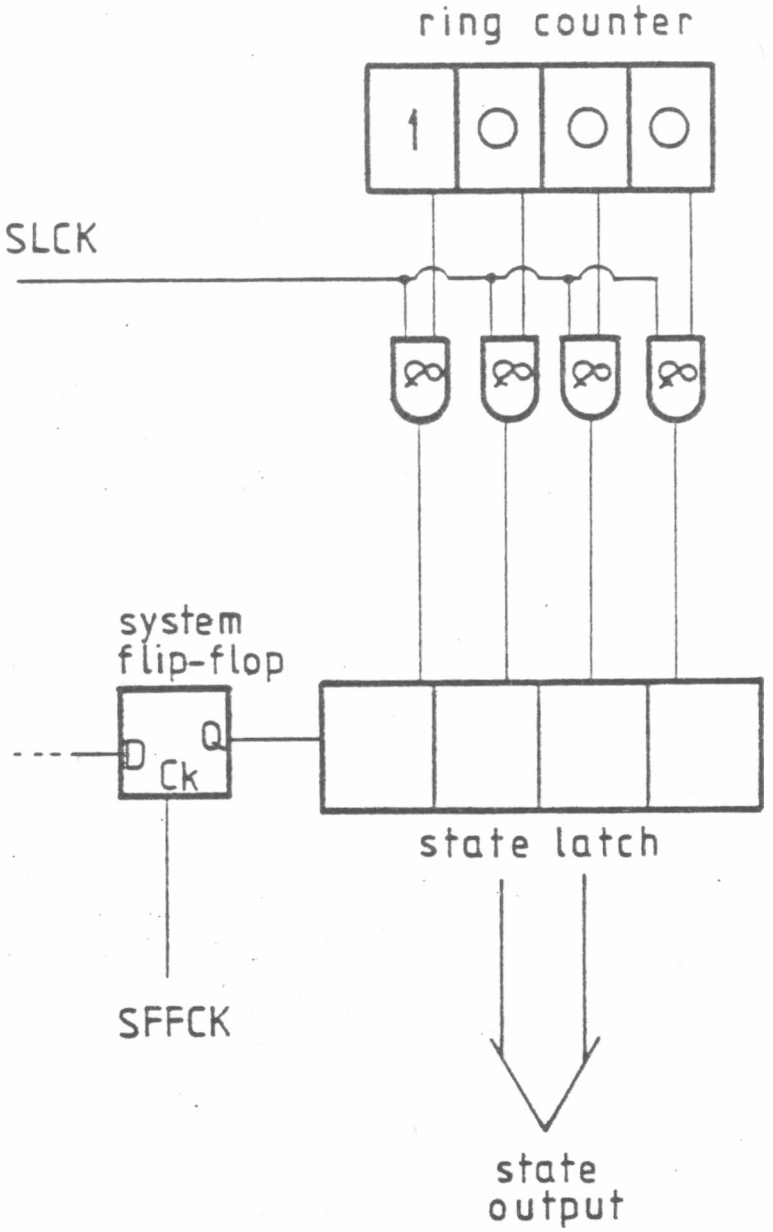


Figure 5.5 State latch loading method

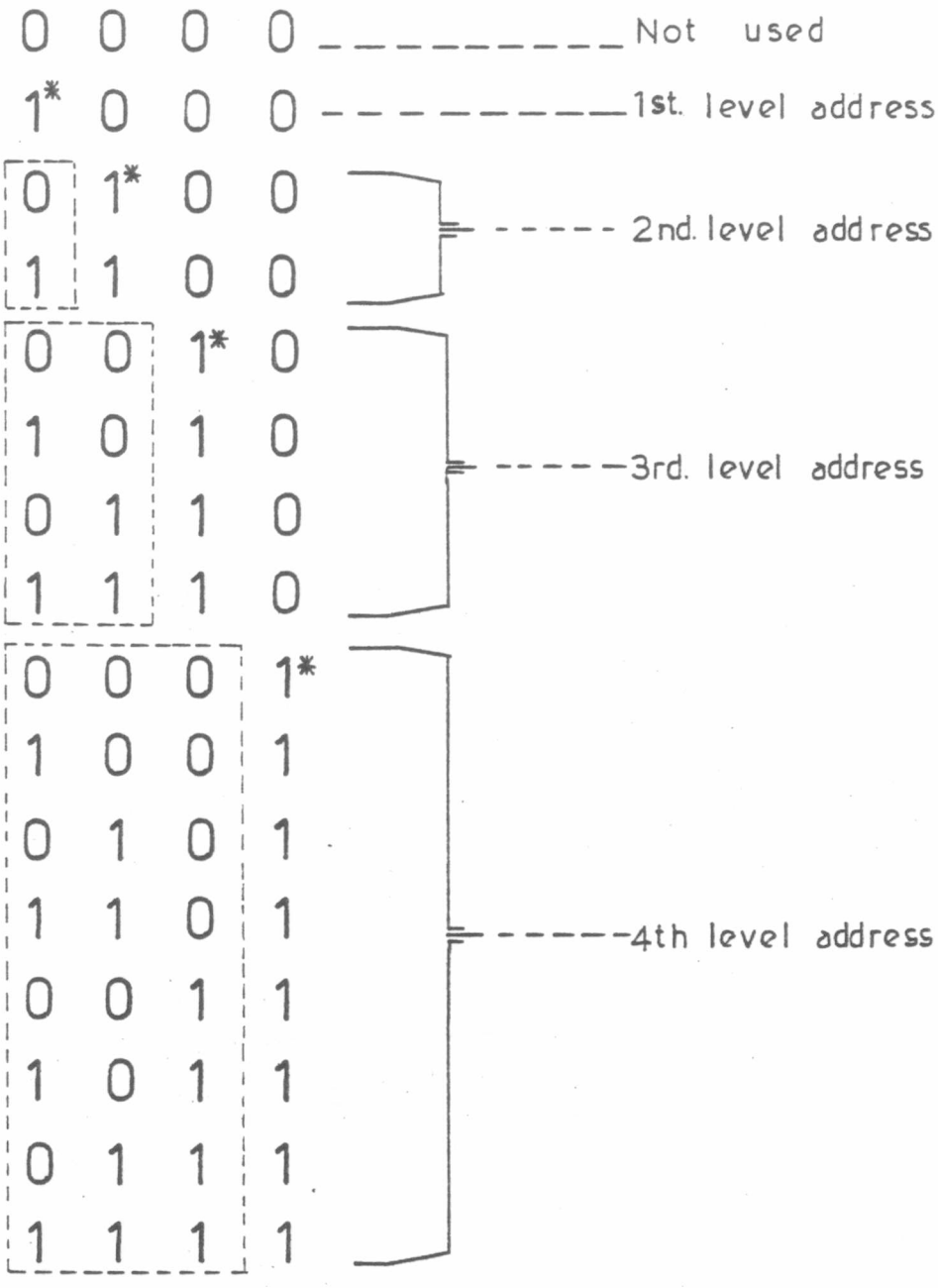


Figure 5.6 Partitioning the RAM address

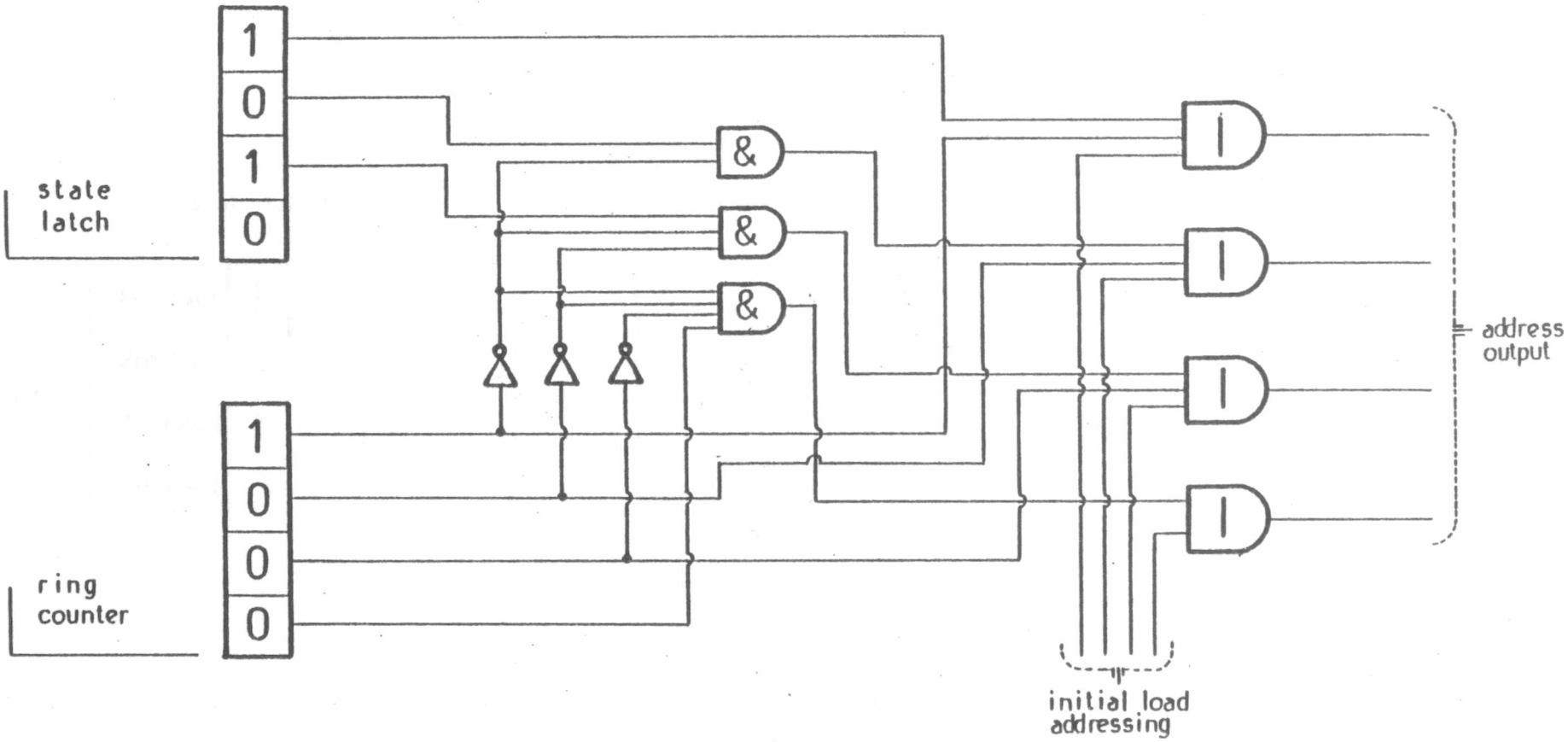


Figure 5.7 Memory address circuit

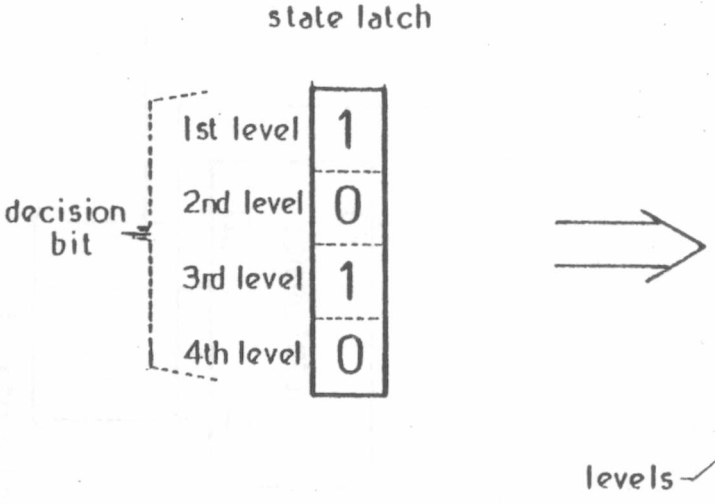
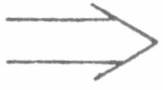

state latch
1st level
2nd level
3rd level
4th level
1
0
1
0
decision bit
levels

Figure 5.8

ring counter states

1st	2nd	3rd	4th
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1



address outputs

1st	2nd	3rd	4th
1	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

levels

Memory address example

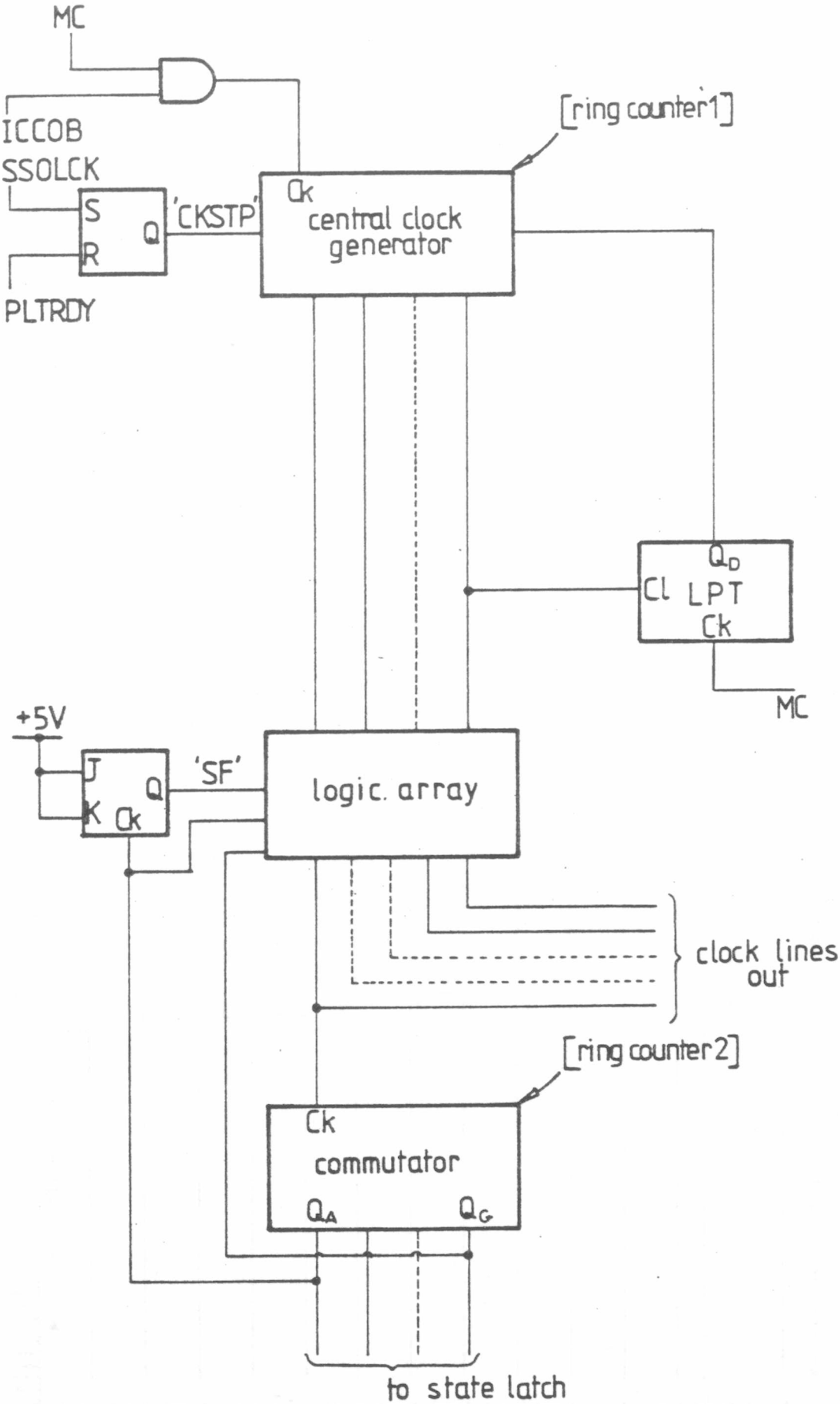


Figure 5.9 Clock generation in the hierarchical system

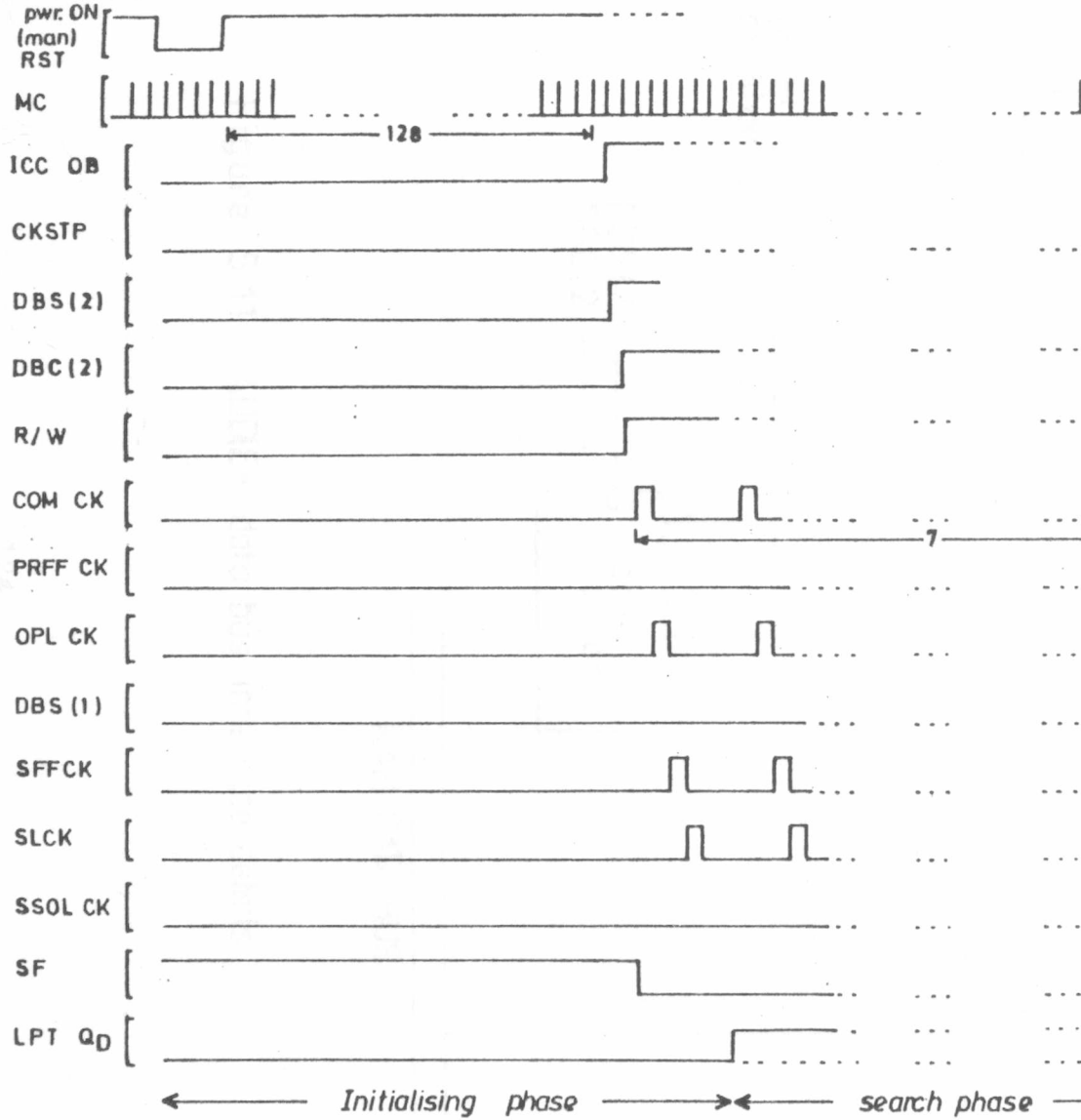
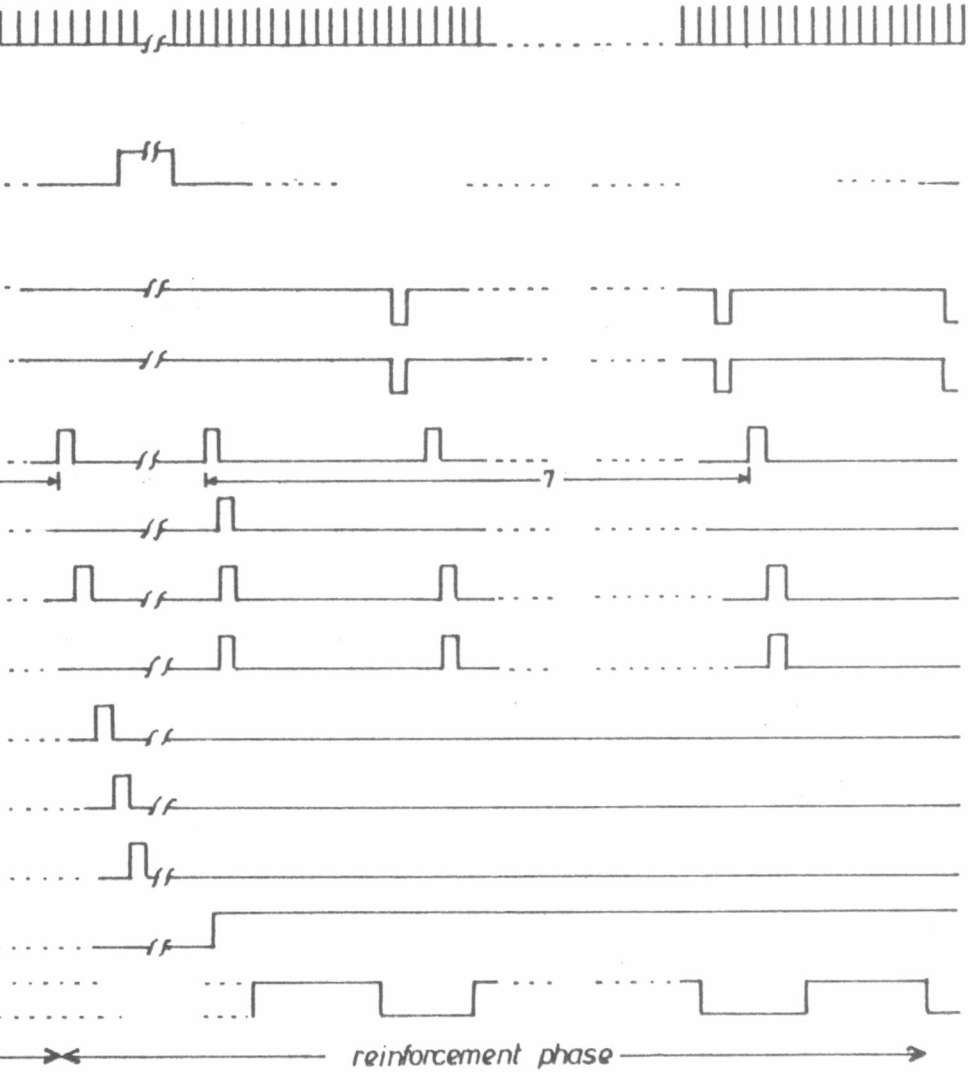


Figure 5·10 System ti



ming diagram

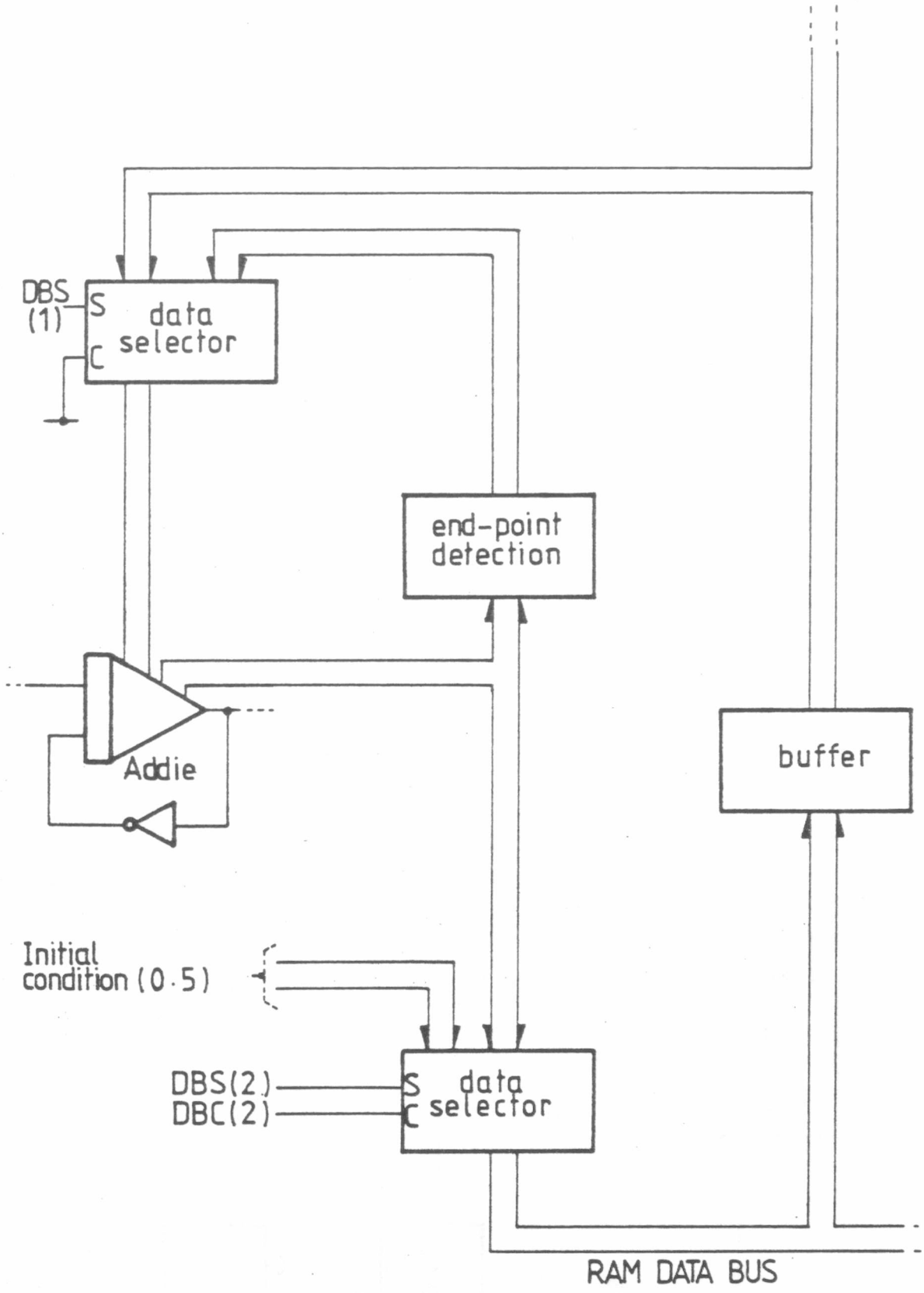


Figure 5.11 ADDIE- data bus interface details

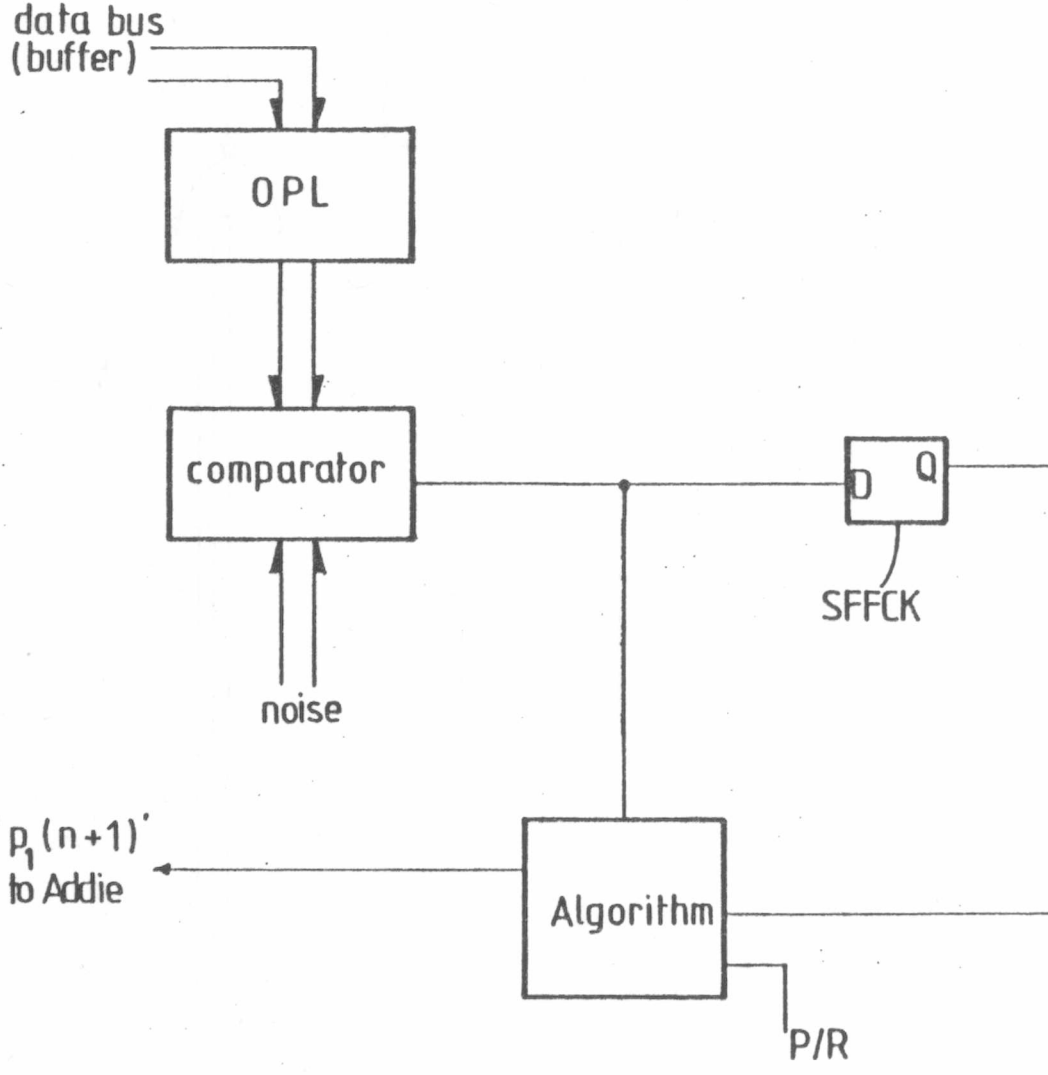
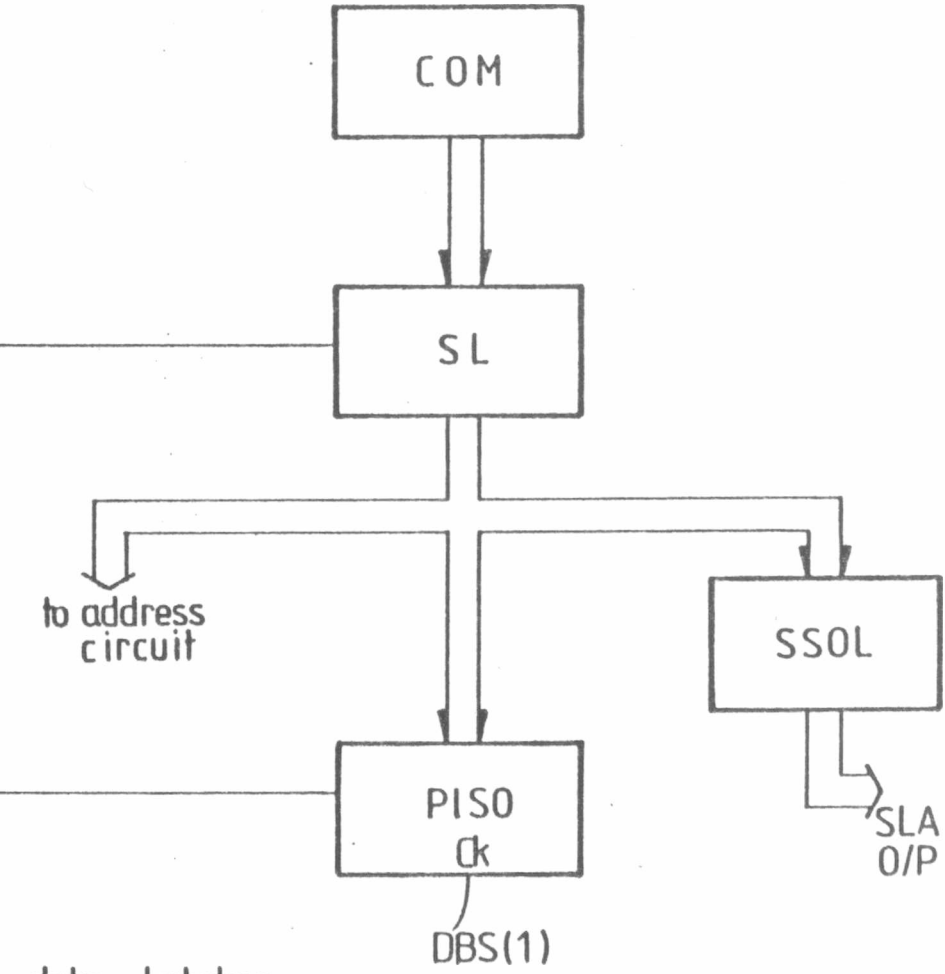


Figure 5.12 System


COM
SL
to address
circuit
SSOL
PISO
Ck
SLA
O/P
DBS(1)
data latches

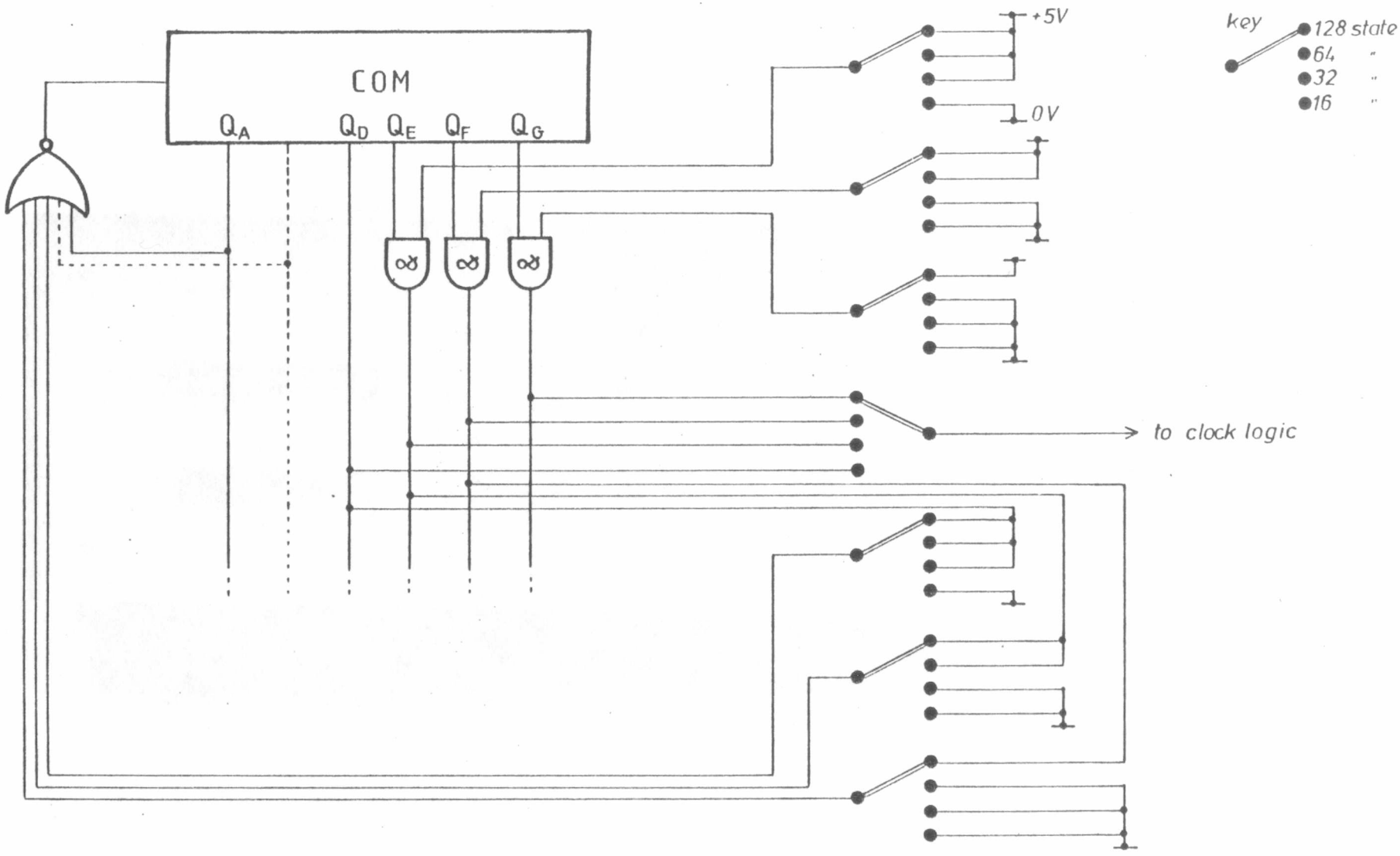


Figure 5.13 Variable size facility

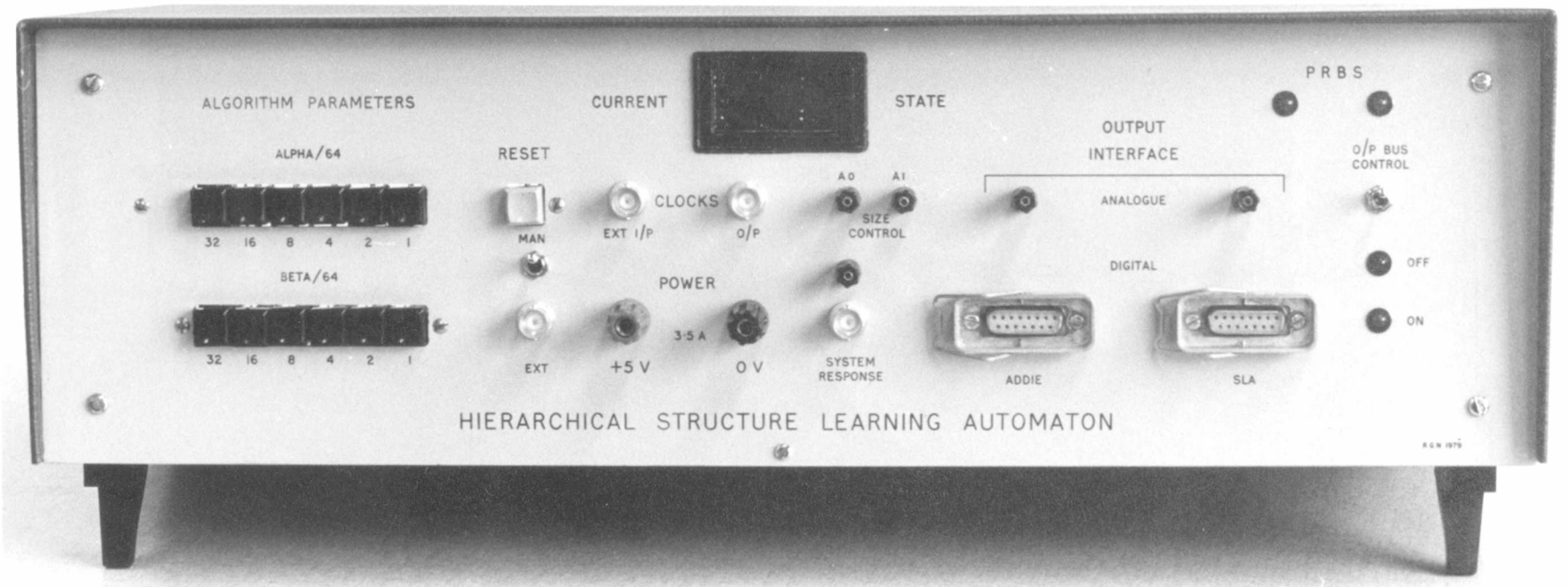


Figure 5·14 128 - state system - front

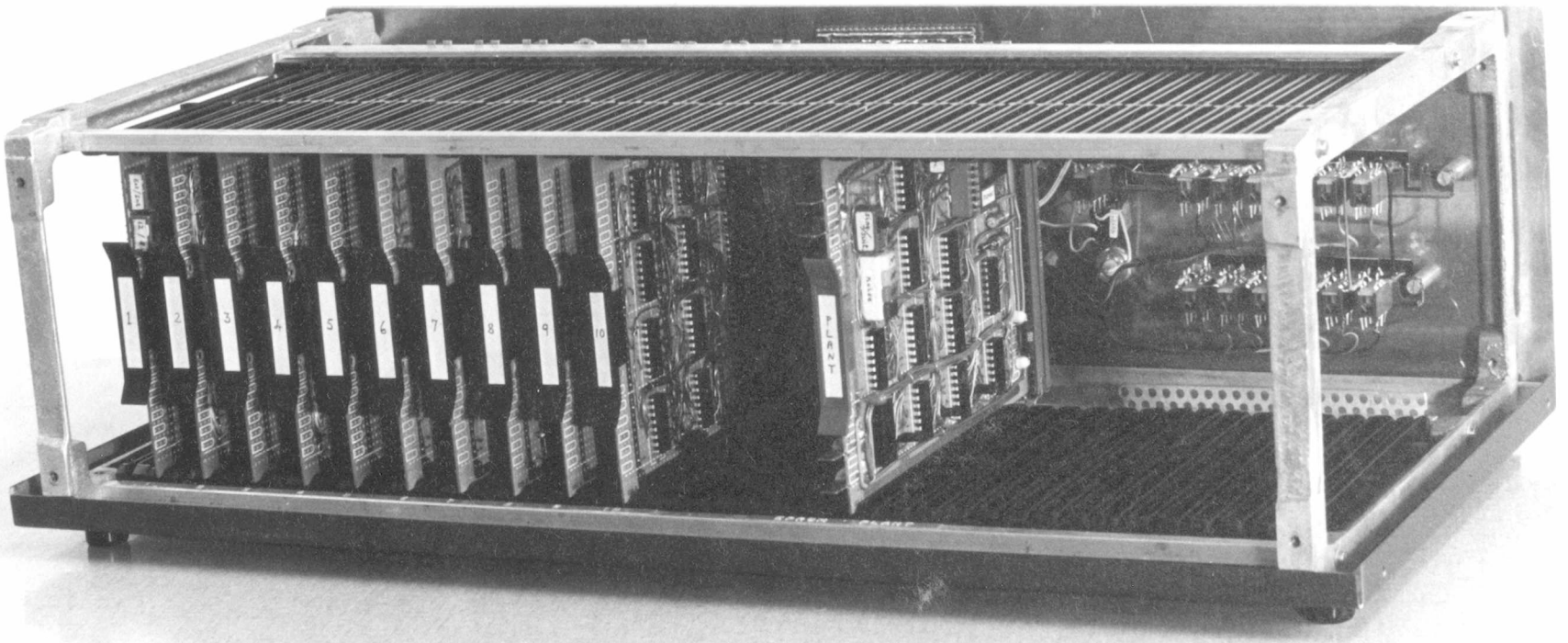


Figure 5.15 128-state system - rear

CHAPTER 6
RESULTS FROM THE HIERARCHICAL SLA

6.1 Preliminary Results

During the initial assembly of the hierarchical automaton in prototype form, it was found convenient to test the various sub-sections separately, as far as was possible, and then to operate the full system initially at a low number of levels as an aid to fault-finding. Figure 6.1 shows the learning characteristic subsequently obtained from a 3-level, 8-state arrangement converging to state 4. The discrete, nested learning curves give a good illustration of the operating principle of the system. The learning time here is typically three times that of a comparable 2-state system, as would be expected from the time-shared nature of the learning process. The hierarchical SLA, it is felt, should be considered akin to a system of co-operative games[66, 67] between 2-state automata, one at each level. Each automaton decides, in turn, which of the two locations below its own current position in the decision tree should be selected, having been steered to that position by the cumulative efforts of the automata above (c.f. Figure 5.1).

Initial optimisation experiments with the 128-state system were performed with the most elementary of simulated plant circuits, as shown in Figure 6.2. This is merely an extension of the 2-state system plant, in that one selected action, in this case number 41, results in a low penalty probability, $c_\ell = 0.25$, and all others a high one, $c_h = 0.875$. Figure 6.3 shows the typical learning behaviour, presented here in the form of an 'output map' of the state latch contents at successive sampling instants, obtained by D/A conversion and presentation on a storage oscilloscope.

The identity of the optimum action can easily be varied by /

by altering the front-end logic in the plant. To verify the absence of any bias in the system, the experiment was repeated for actions near each end of the range: number 5 and number 120. The results are shown in Figures 6.4 and 6.5 respectively. In each case, convergence is achieved in the range of 50 to 100 ms. The approximate length of one iteration, using the 8-bit ADDIE and a 2.5 MHz master clock, is 100 μs, so the indicated learning times correspond to between 500 and 1 000 iterations.

There is always a small probability of incorrect decisions beyond the initial learning period, at any level, due to variance inherent in the operation of the ADDIE. This results in the sporadic occurrence of incorrect output actions which show up on the output map as scatter points off the main trace.

As before, it is a simple matter to simulate a non-stationary environment by switching the identity of the optimum action. Figure 6.6 shows how the SLA can adapt to an environment in which the optimum action is switched between 58 and 106. It is significant that not only does the SLA adapt successfully, but also that the adjustment period is virtually the same as the initial learning response time.

6.2 Application to Multimodal Systems

Having established satisfactory operation of the hierarchical SLA with a very basic "plant", the next step was to consider its application to the more practical situation of multivariable systems with multimodal performance criteria. In order to simulate such an environment, a rather more sophisticated plant circuit was required. It was decided to use as an example a widely-reported P I function /

function[68, 69] which has a clearly defined global optimum, a local optimum and a saddle point.

The objective function describing this particular performance index is given by

$$f(x, y) = (1+8x - 7x^2 + \frac{7}{3}x^3 - \frac{1}{4}x^4) y^2 e^{-y}$$

where x and y are the two control variables. In order to match the P I to the SLA, a program, denoted "SLIG", was written which computed f(x, y) over the requisite range and provided a choice of output formats. Figure 6.7 shows the surface plotted on a fine grid to illustrate its general features. Figure 6.8 presents an alternative view, using this time a 16 x 8 grid to illustrate the partitioning of the surface into 128 discrete elements. SLIG also generates a table of penalty probability values (c_i) corresponding to each surface element, and writes these to a dump file. A third output option is to produce a punched tape of binary c_i values to feed a PROM programmer designed specifically for 2708-type PROMS (1 K x 8-bits).

A useful feature of the program is the provision for a choice of "compression factor" to be applied to the range of c_i values derived from normalisation of the P I objective function. This enables the final values of the penalty probabilities to be constrained between chosen limits within the full-scale range $[0, 1]$, as a test of the discriminatory powers of the SLA. For the experiments reported below, the compression factor was set at 0.95, giving a minimum c_i of 0.025 and a maximum of 0.85 for the full 128-state system.

The principal elements of the plant simulator circuit are shown in Figure 6.9. The PROM is at the centre, storing the c_i values as 8-bit numbers, each addressed by /

by the appropriate action output from the SLA. In order to evaluate the performance of the SLA under non-stationary conditions also, it was arranged that a "reflected" version of the P I be stored in the next 128 bytes of the PROM (i.e., x and y - axes reversed). In this way, simply switching the most significant address line abruptly changes the environment as seen by the SLA.

It was decided to dispense with the earlier method of simulating surface noise or observational error described in Chapter 2, which called for a two-phase clock and a rather slow sampling rate. Instead, the presence of noise is effected by introducing a full adder fed with random numbers in the form of noise lines tapped off the central PRBS source. The resulting noise-corrupted c_i value is then applied to a standard noise comparator arrangement which produces a stochastic pulse train representing the penalty probability. This is subsequently sampled by the penalty/reward flip-flop to provide the appropriate environment response (0: reward, 1: penalty) for use by the SLA in the reinforcement phase. The use of computer facilities in the preparations for these experiments is summarised in Figure 6.10.

6.3 Data Logging

In order to obtain maximum flexibility in the presentation of data from SLA learning runs, it was decided to use a computerised data logging system. A program, "SLOG", was written which recorded the output action after each iteration and stored the information in a data file, together with the relevant parameters for the particular experiment. A companion program, "SLAG", was then written to enable processing of the data, together with the c_i file provided by SLIG, to form an output map, a penalty curve (i.e., a plot of /

of received penalty against iteration number), and a set of cumulative distribution curves illustrating the evolution of automaton action at various stages during the learning process. Of this family of programs, SLAG and SLIG were written primarily for main-frame system operation (DEC-20), while SLOG was run on an LSI 11/03. All three programs are in FORTRAN, though some MACRO routines are called as appropriate. The application of SLOG and SLAG to the logging and presentation of experimental data is summarised in Figure 6.11.

The logging process requires the SLA to interrupt the 11/03 whenever an output action is available. The circuit used to accommodate this is shown in Figure 6.12. The clock pulse which triggers the system state output latch is differentiated to produce a narrow spike, sufficiently shorter than the computer's interrupt response time. This sends an interrupt request via the flip-flop, which is subsequently reset by the reply signal from the computer. The selected action, held in the seven bit output latch, is then read via the I/O ports to a disk file.

6.4 Results and Comments

A total of seven experiments performed with the hierarchical SLA is reported here, using the performance index described above with a superimposed noise component of $\pm z$ distributed uniformly over the surface. Four bits of noise were in fact added, so that z represented approximately 3% of the full-scale range (0-255) of the 8-bit c_i values. The results are detailed below with appropriate comments.

Experiment 1: /

Experiment 1: 128-state, L_{R-P} scheme

$\alpha = 0.437$, $\beta = 0.992$

The output map, penalty curve and distribution curves for this experiment are shown in Figures 6.13, 6.14, 6.15 respectively. The form of the output map bears a close resemblance to that of Figures 6.3 - 6.5, which were reproduced directly from an oscilloscope trace, and as before convergence is achieved in around 1 000 iterations. This particular learning run shows convergence to the optimum action (100), while the effect of spurious state transitions producing suboptimal actions, commented on earlier, shows up clearly on the penalty curve as transient spikes to a higher level of received penalty. Although the values of received penalty are only calculated at discrete points, it was decided to present the result as a continuous curve to illustrate more clearly the underlying trend.

Experiment 2: 128-state, L_{R-P} scheme

$\alpha = 0.25$, $\beta = 0.875$ (Figures 6.16 - 6.18)

This experiment was chosen to demonstrate the effect of a low ratio of reward to penalty, in this case $\gamma = 6$. The output map presents a rather chaotic picture, as does the penalty curve to some extent. However, the distribution curves prove that the SLA does, in fact, favour actions at or near the optimum, since a set of peaks is clearly evident, spaced at intervals of 8 in accordance with the partitioning of the surface into 16 x 8 elements for 128 possible actions.

This result, therefore, bears out the expected performance of an L_{R-P} automaton with a small measure of expediency, i.e., reliable convergence to a condition in which favourable actions are selected, though with probability somewhat /

somewhat short of unity.

Experiment 3: 128-state, L_{R-I} scheme
$\alpha = 0.25$ (Figures 6.19 - 6.21)

This result is a classic example of an L_{R-I} automaton locking-on to the wrong action. The learning characteristics are very similar to those of Experiment 1, but this time the system homes in on action 91. This illustrates the basic flaw in the L_{R-I} scheme, in that its very high degree of expediency (ϵ-optimality) can lead to an inability to escape from a situation where the chosen action turns out to be sub-optimal, which may well arise in the case of a non-stationary environment.

Experiment 4: 128-state, L_{R-P} scheme
$\alpha = 0.5$, $\beta = 0.992$ (Figures 6.22 - 6.24)

While the previous experiments were logged over 2 000 iterations using a static environment, this experiment was run for 5 000 iterations, with the plant switched after 2 048 iterations, as described earlier, under the control of a binary counter fed with state output latch clock pulses. In this particular case, the set-up was reversed so that the "reflected" plant was chosen first.

The result shows, first of all, convergence predominantly to actions 19 and 20, which are low penalty but suboptimal actions. After the plant is switched, there follows an interim adjustment period, culminating in convergence to the "new" optimum of action 100. Although the first half of this trial did not yield optimal behaviour, since action 28 would have been preferred, the overall result does nonetheless demonstrate the ability of the SLA to track a non-stationary environment without excessive /

excessive delay, provided a "suitable" (L_{R-P}) reinforcement scheme is employed.

The following three experiments were chosen to illustrate the flexibility provided by the hierarchical structure of this hardware SLA design. Using the front panel patching facility to alter the programmable system clock described previously, it is a simple matter to change the size of the structure immediately (in binary multiples).

Experiment 5: 64-state, L_{R-I} scheme
α = 0.125 (Figures 6.25 - 6.27)

In the case of the 64-state system, the automaton addresses only every second point on the P I surface (odd numbers). The optimum action here is 101, which becomes 51 in the nomenclature of the 64-state system. The result of this particular experiment demonstrates once again how an L_{R-I} scheme can lock-on to the wrong action, in this case number 50.

Experiment 6: 32-state, L_{R-I} scheme
α = 0.125 (Figures 6.28 - 6.30)

This result for a 32-state system is essentially similar to Experiment 5, showing convergence to action 24, whereas 26 is optimum.

Experiment 7: 16-state, L_{R-I} scheme
α = 0.125 (Figures 6.31 - 6.34)

This last experiment is significant, in that by addressing itself only to every eighth element of the P I surface, the 16-state automaton in effect sees only the rather shallow front edge (see Figure 6.8), reducing the problem to a simple two-dimensional system. However, the /

the corresponding penalty probability range here is only from 0.71 to 0.85, which clearly presents a severe test of discrimination. The result indicates convergence to action 15, whereas 13 is the optimum. However, the penalty curve verifies that a useful reduction in received penalty is achieved as a result of automaton action.

As the number of states is reduced, the learning times are clearly reduced also. Therefore, two sets of distribution curves were obtained from this experiment. The first set, covering 2000 iterations as before, shows that all significant acitvity is over within the first 1000 iterations. A second set (Figure 6.34) was accordingly obtained to cover this initial period and to illustrate in greater detail the evolution of the selected action.

In all of these experiments, a 12-bit ADDIE was used in the interests of greater precision and lower variance, though at the expense of operating speed. Actual learning times for the above results can be estimated in the context of an approximate iteration time of 1 ms.

6.5 Conclusions

As before, all the results are derived from "one-off" trials. It was felt that the nature and complexity of the 128-state system precluded the presentation of results averaged in some way over many experiments. Sufficient information can, in any case, be derived from representative samples such as those detailed above.

The results of these experiments clearly demonstrate the power of the hierarchical SLA as a means of achieving rapid optimisation of a multimodal system, irrespective of contour and despite the presence of measurement noise. Even /

Even in cases where sub-optimal convergence occurred, stemming from a reinforcement scheme of which such behaviour is a known hazard (L_{R-I}), the automaton succeeded in choosing actions which were adjacent to the optimum. It can therefore be said to have performed its allotted task of reducing the average received penalty. This in turn implies an improvement in system performance which at least approaches the ideal.

It must be stressed that at no time did convergence to the local optimum occur, demonstrating conclusively that the SLA has purely altitude sensitivity over the P I surface, as opposed to the gradient sensitivity characteristic of the conventional hill-climbing methods of optimisation. The successful results obtained with switched environments are particularly significant, since it is likely that most SLA applications will involve non-stationary plant.

The presence of measurement noise on the P I surface does not seem to impair significantly the performance of the SLA. Indeed, it can be argued that its perturbating effect on the value of received penalty on a short-term basis might help to dislodge a highly expedient learning scheme from an incorrect action to which it might otherwise lock-on. This would permit a slightly higher degree of expediency, which does have desirable features, to be catered for in the reinforcement process.

The development of a viable hardware automaton enables a fully operational learning controller to be implemented, and the following chapter is devoted to the study of a practical system and the results achieved.

Figure 6.1 Learning characteristic of an 8-state system

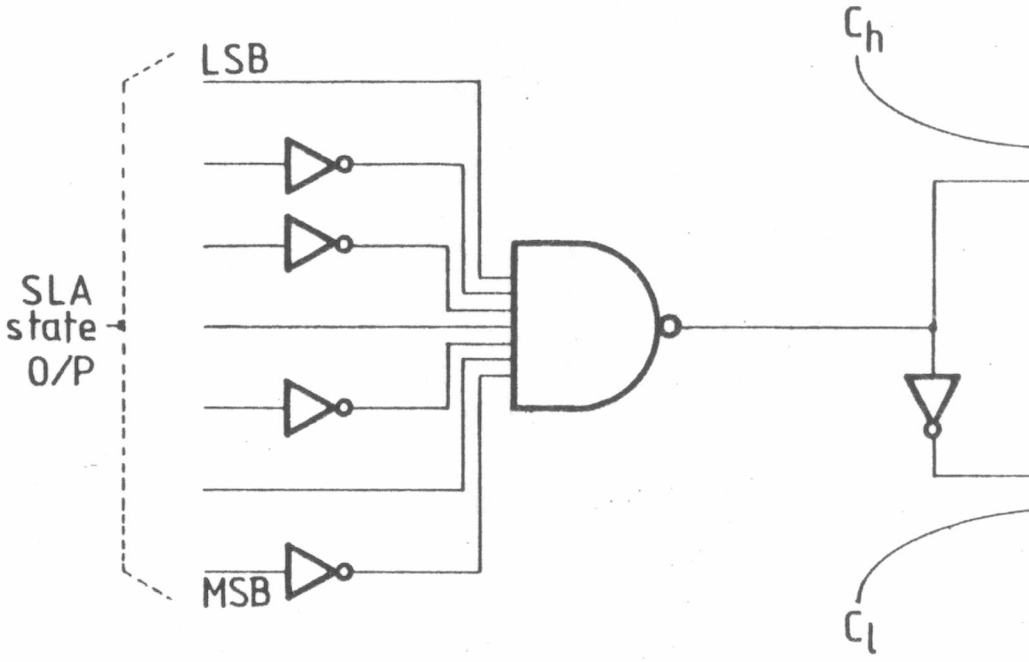
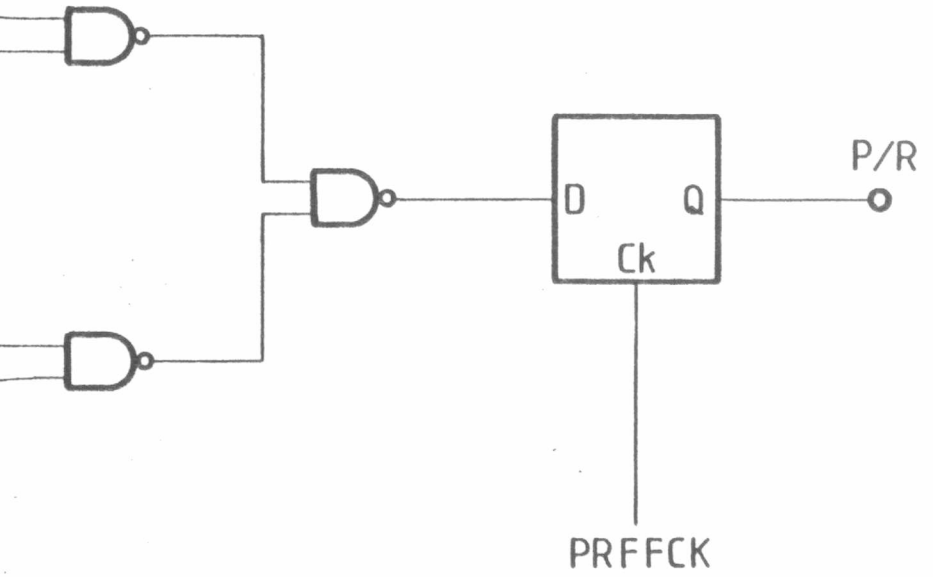


Figure 6.2 Simple plant



simulator (action 41)

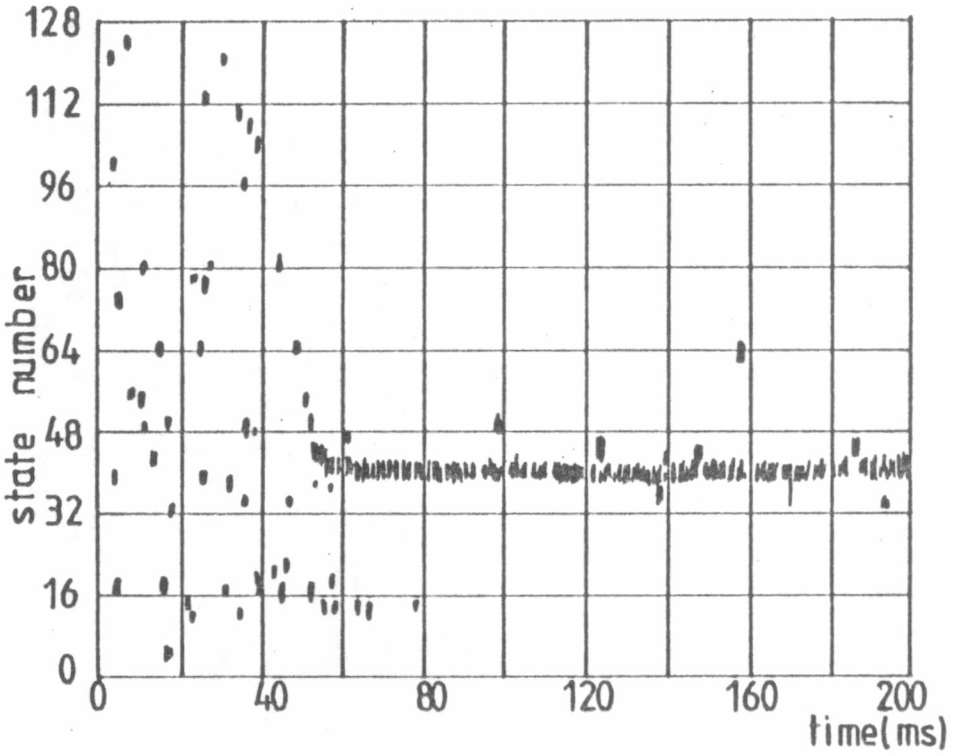


Figure 6.3 Output map (action 41)

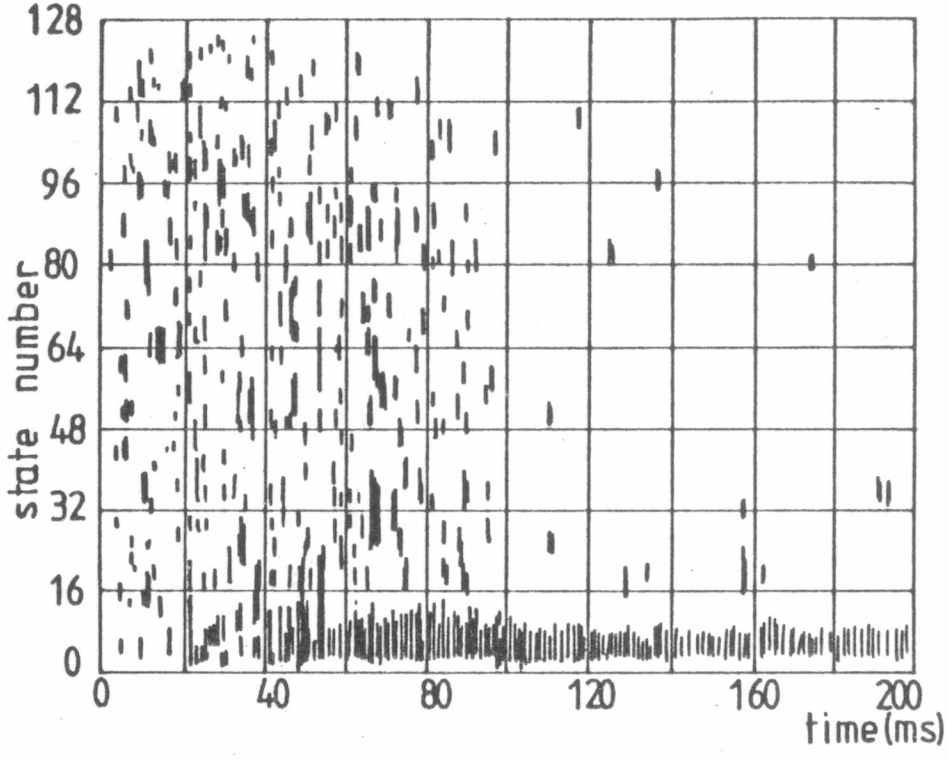


Figure 6.4 Output map (action 5)

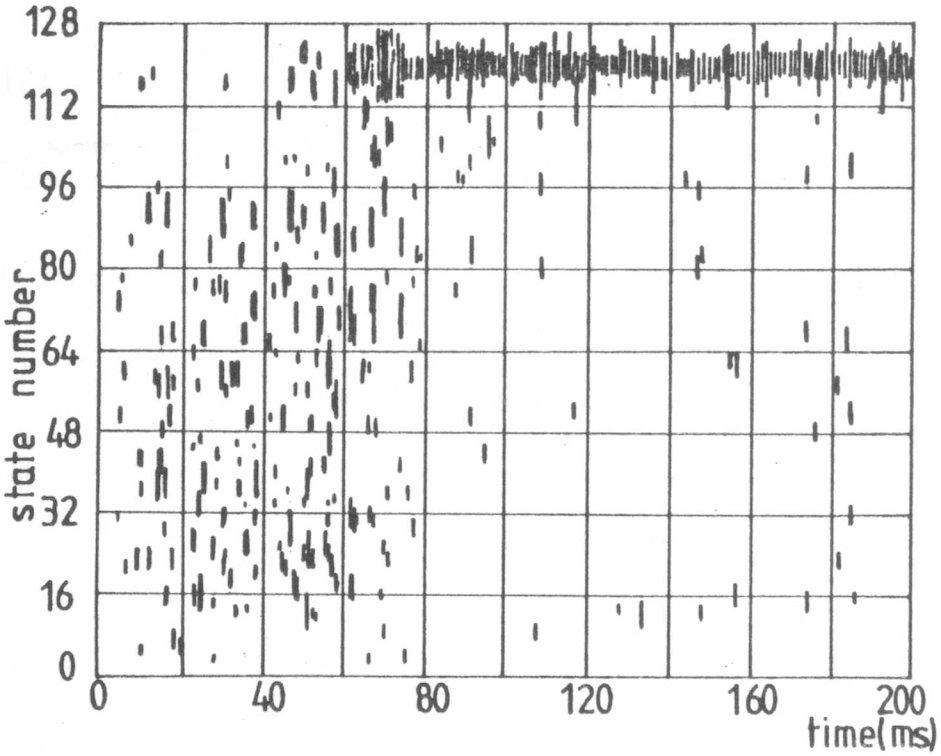


Figure 6.5 Output map (action 120)

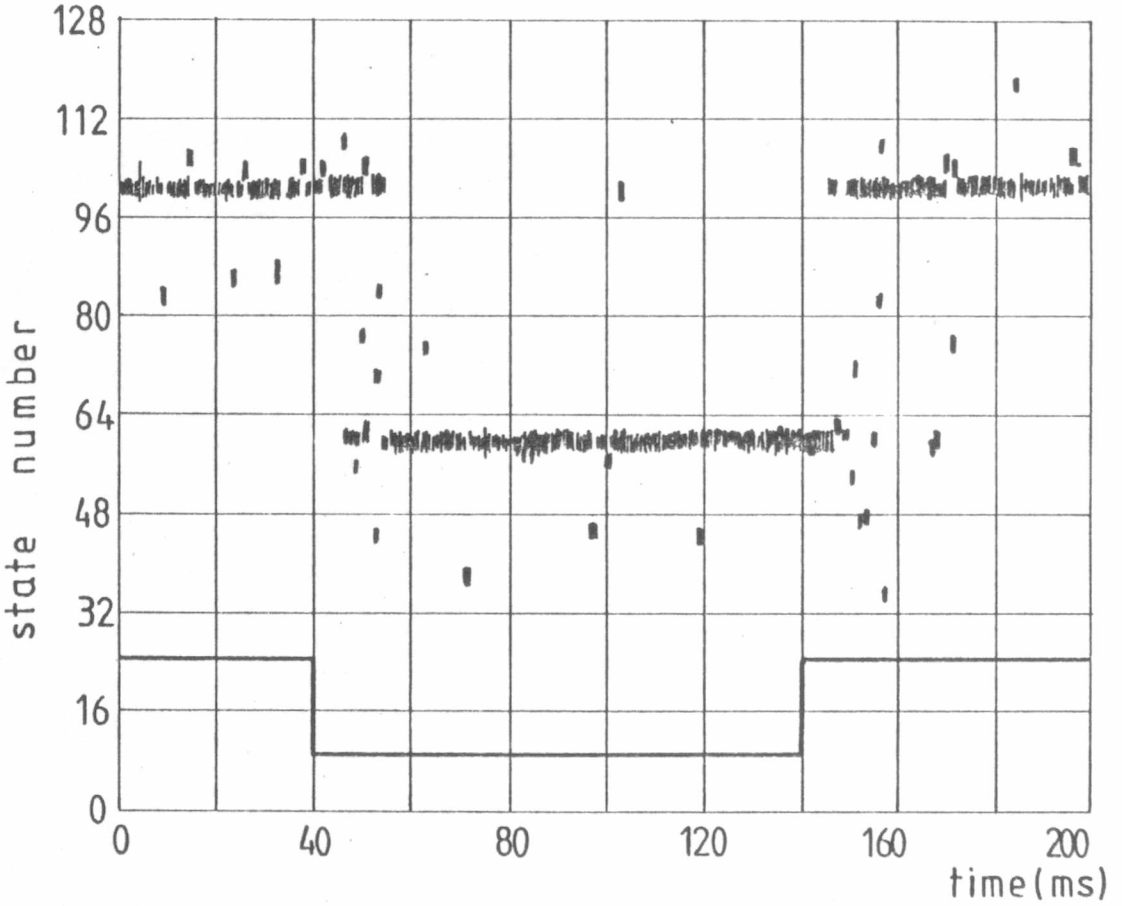


Figure 6.6 Output map for switched environment

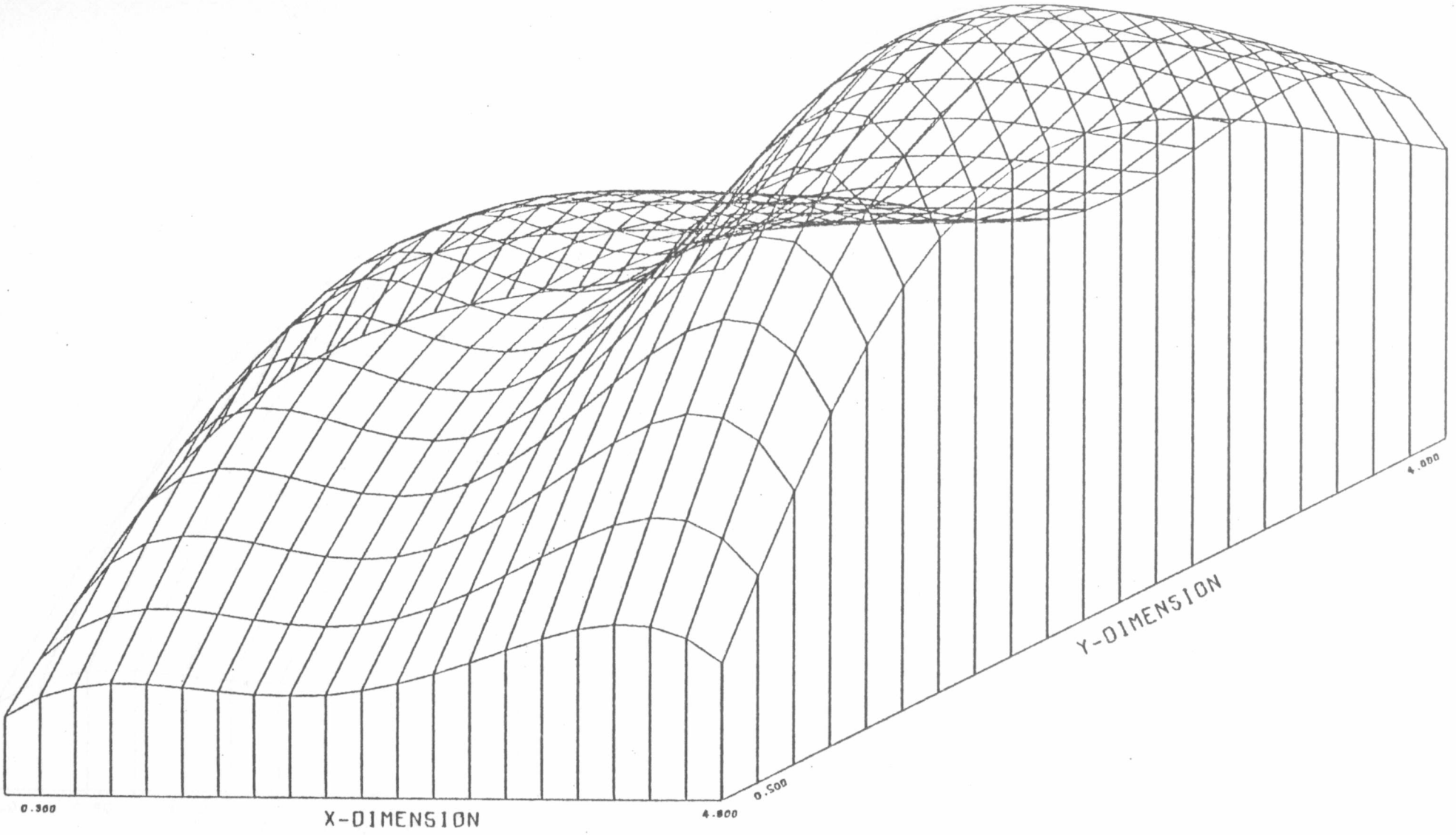


Figure 6.7 Computer-generated view of P. I. surface

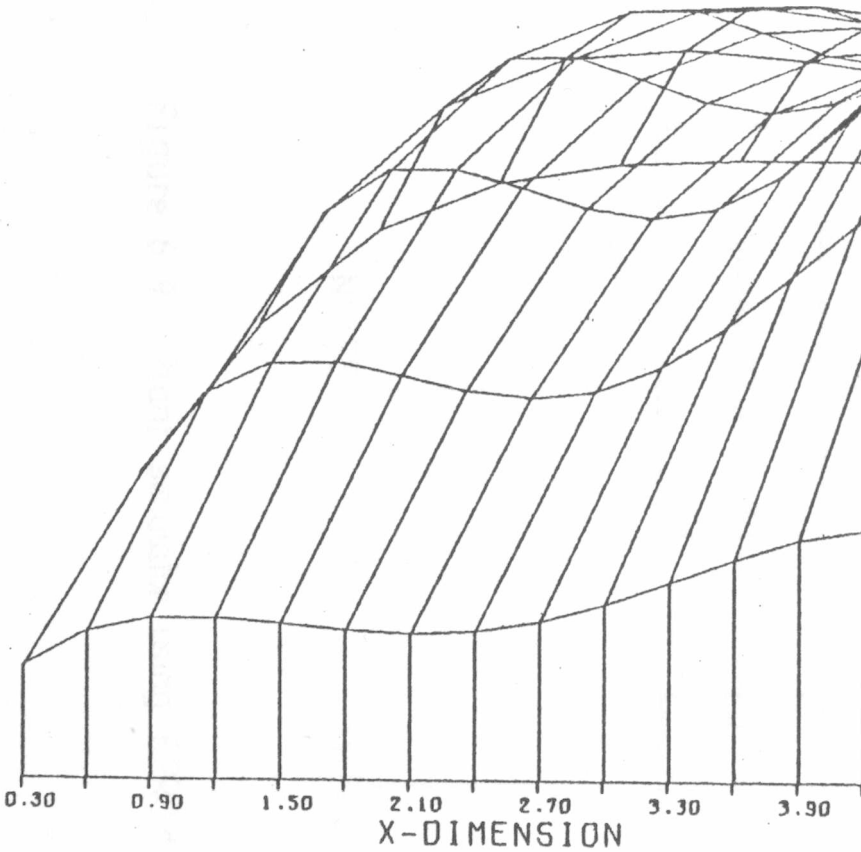
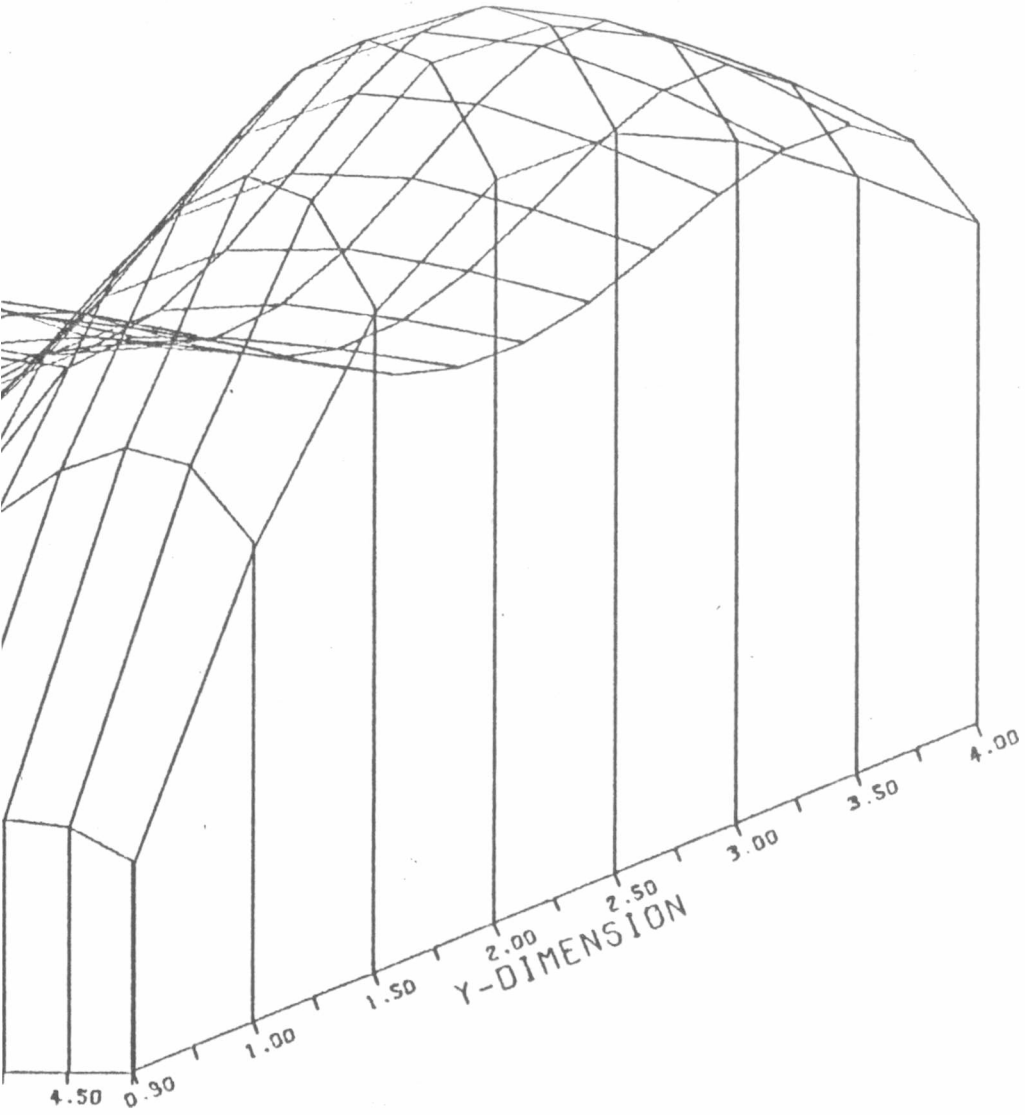


Figure 6.8 P-I.



surface with 128-point grid

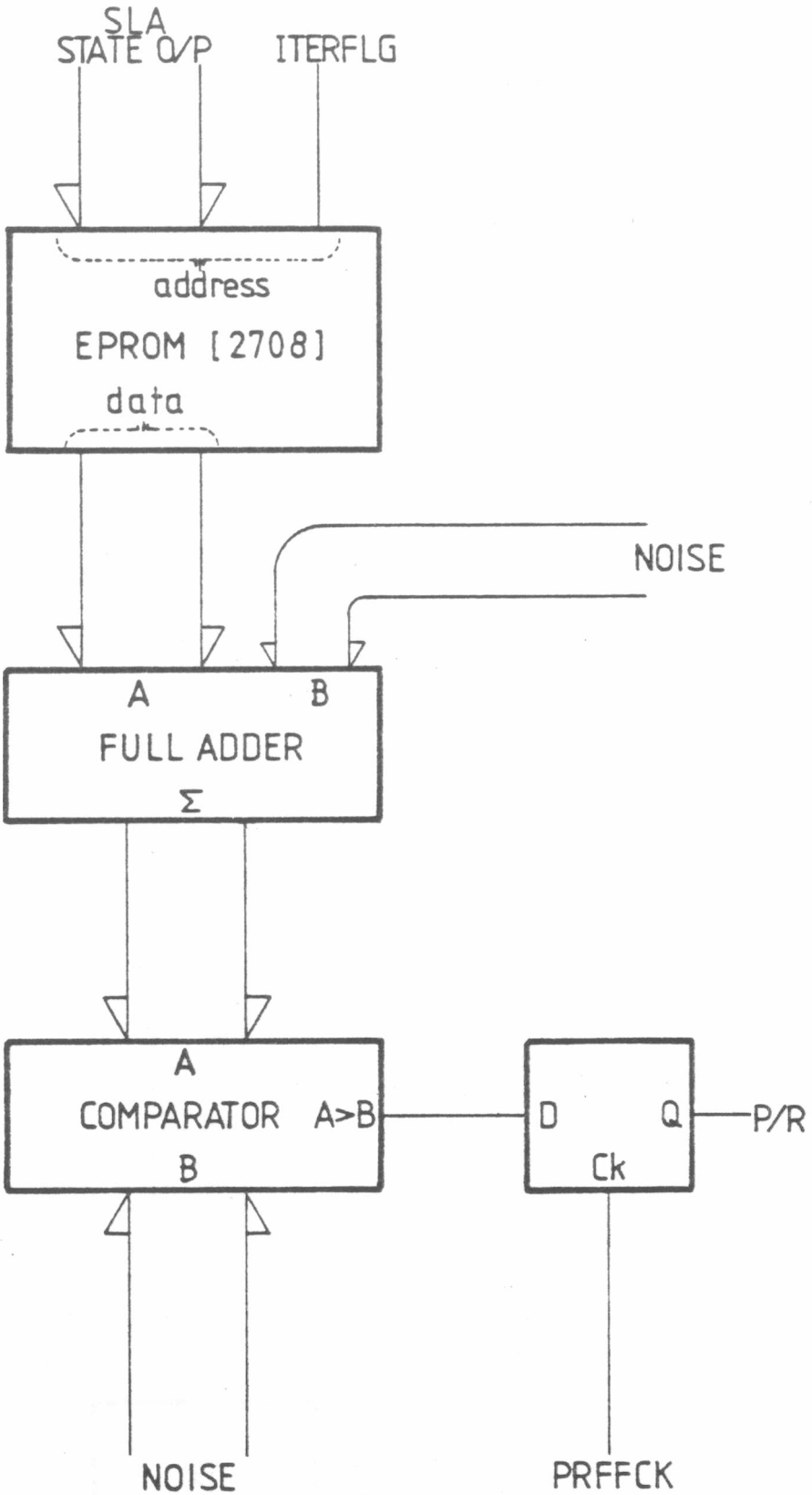


Figure 6.9 Plant simulator using EPROM

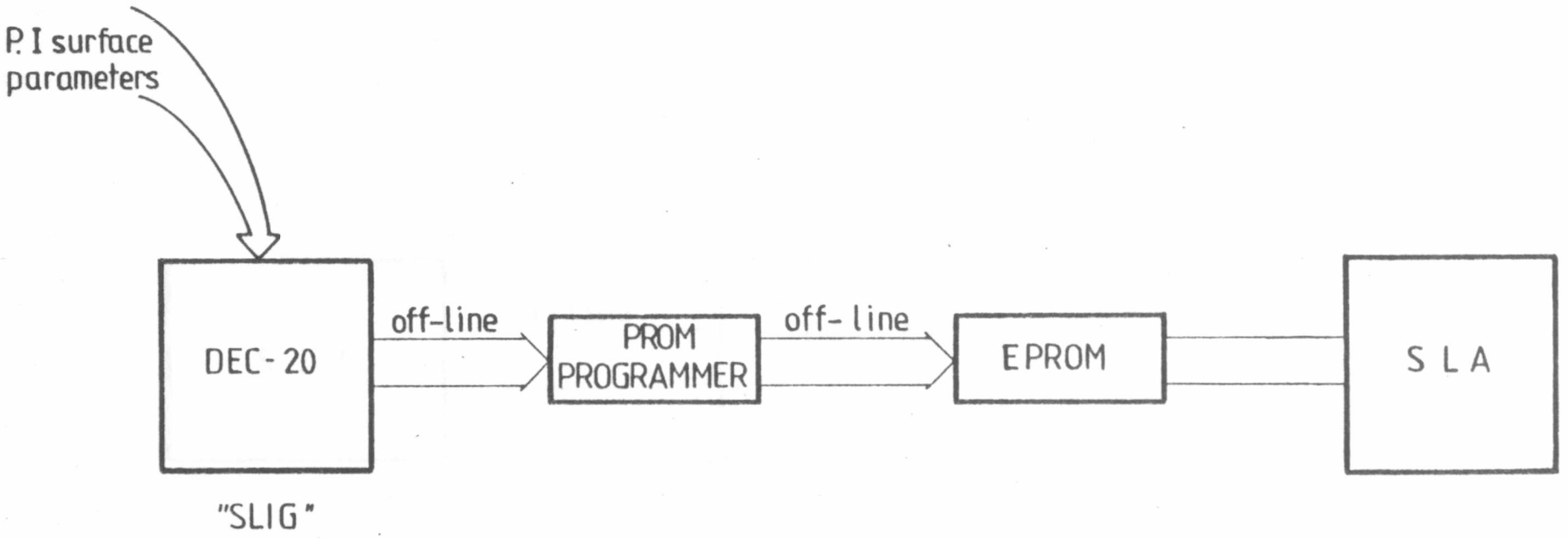


Figure 6.10 Data preparation for SLA experiments

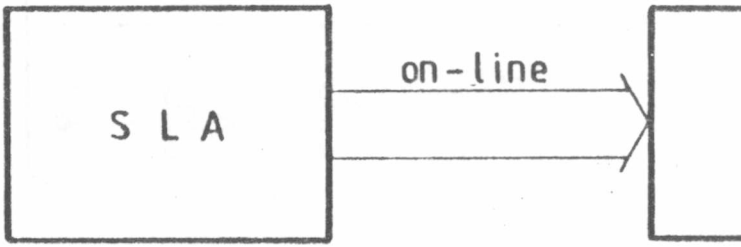
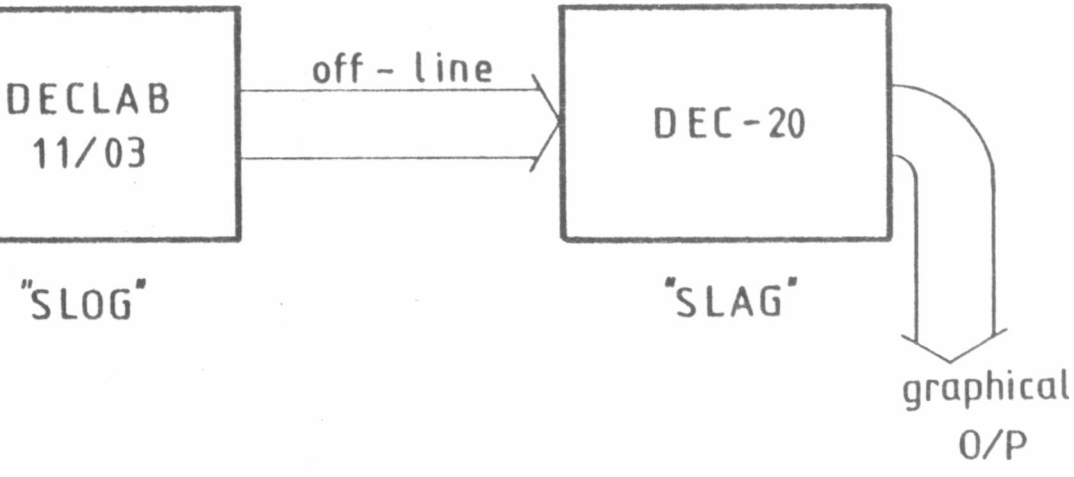


Figure 6.11 Logging



and presentation of SLA learning runs

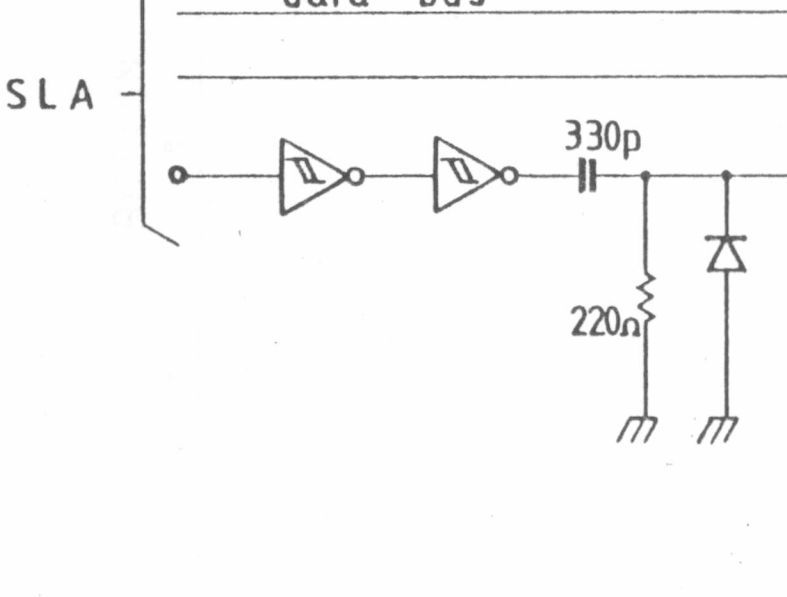
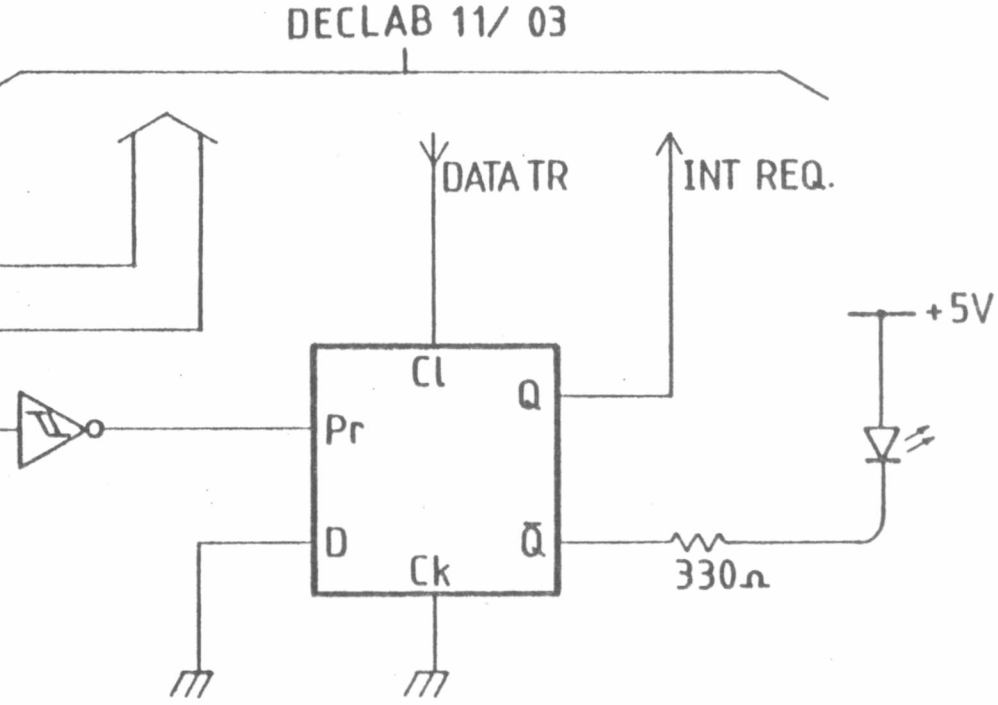


Figure 6.



12 Interrupt processing circuit

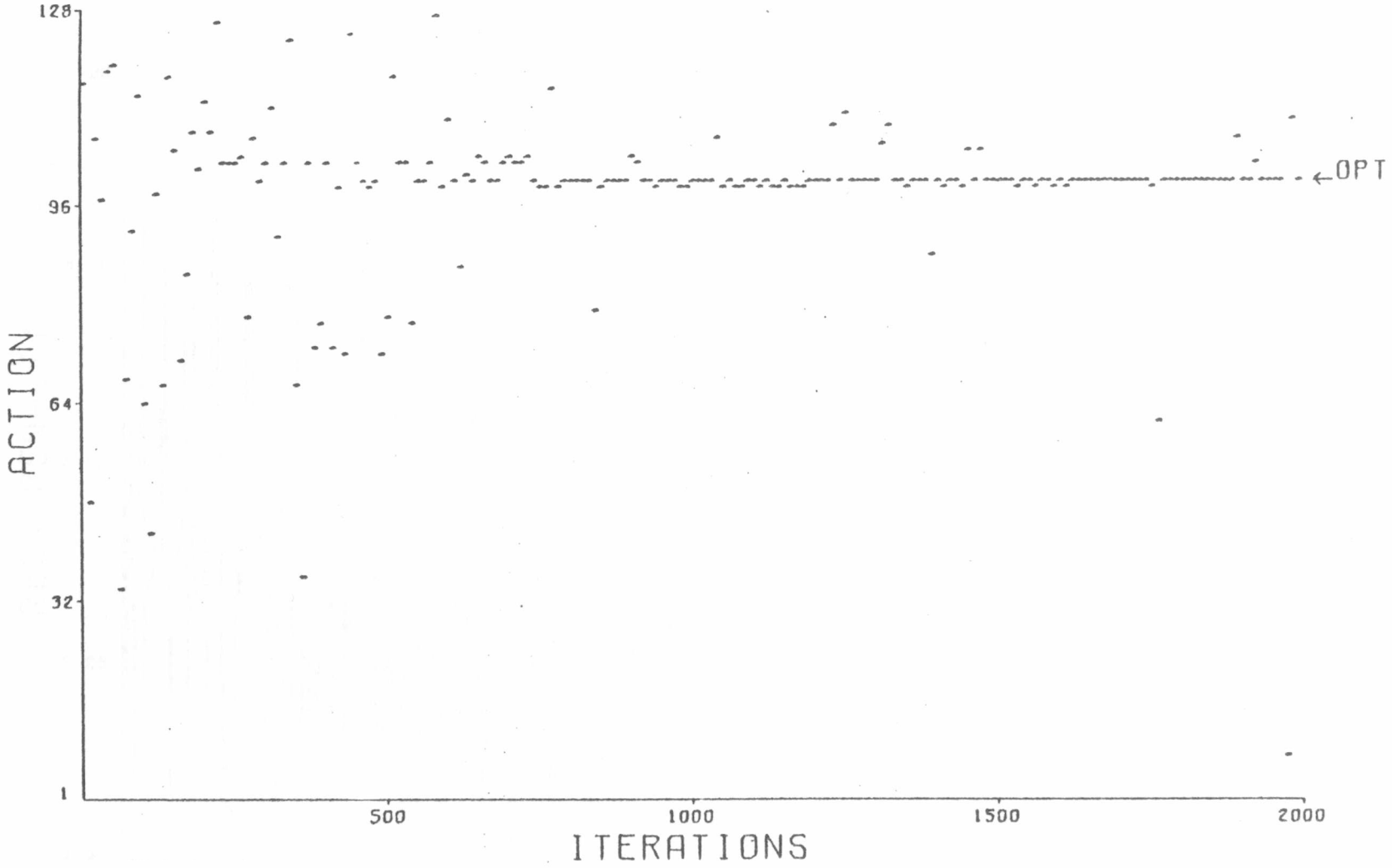


Figure 6·13 Output map (1)

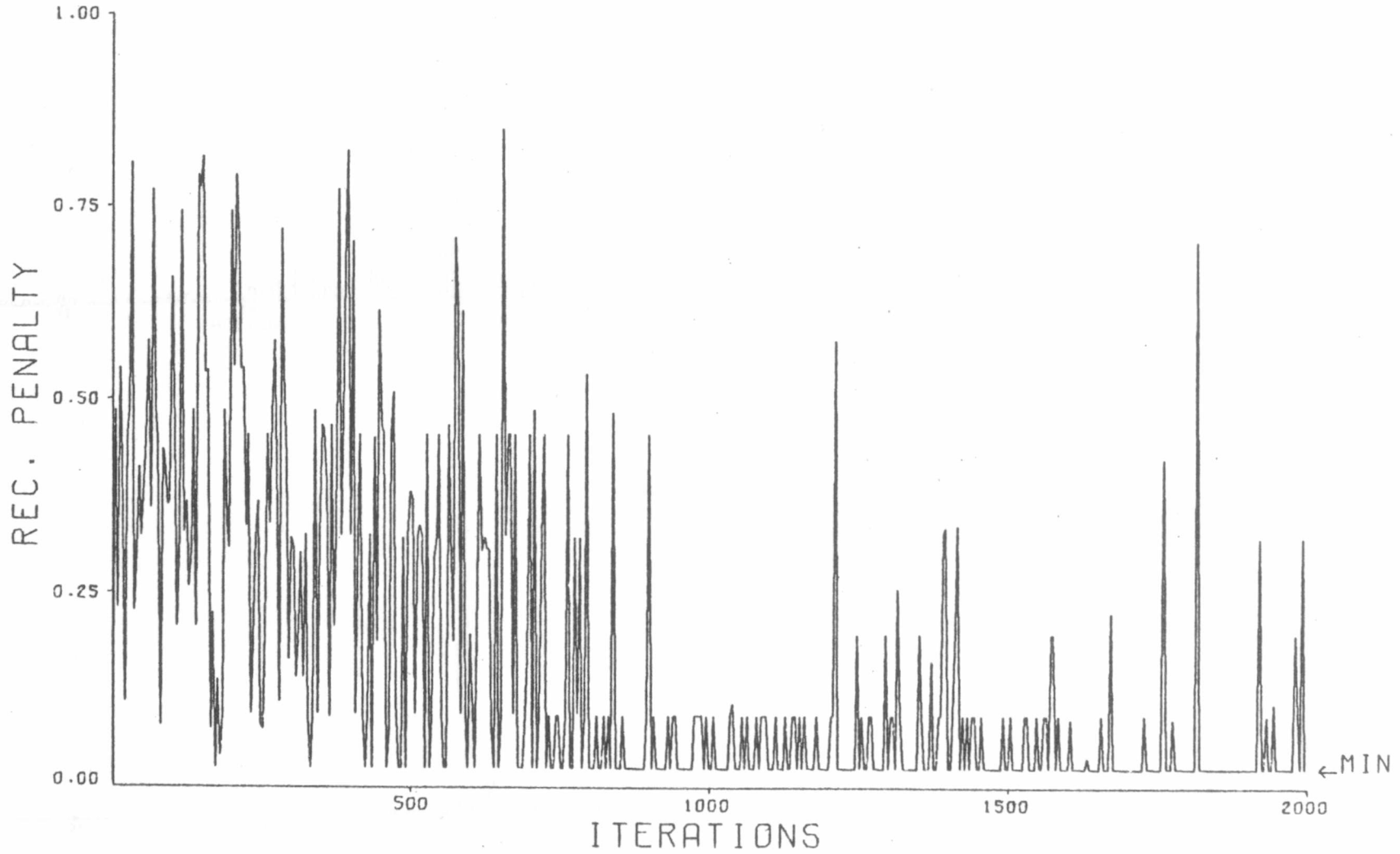


Figure 6.14 Penalty curve (1)

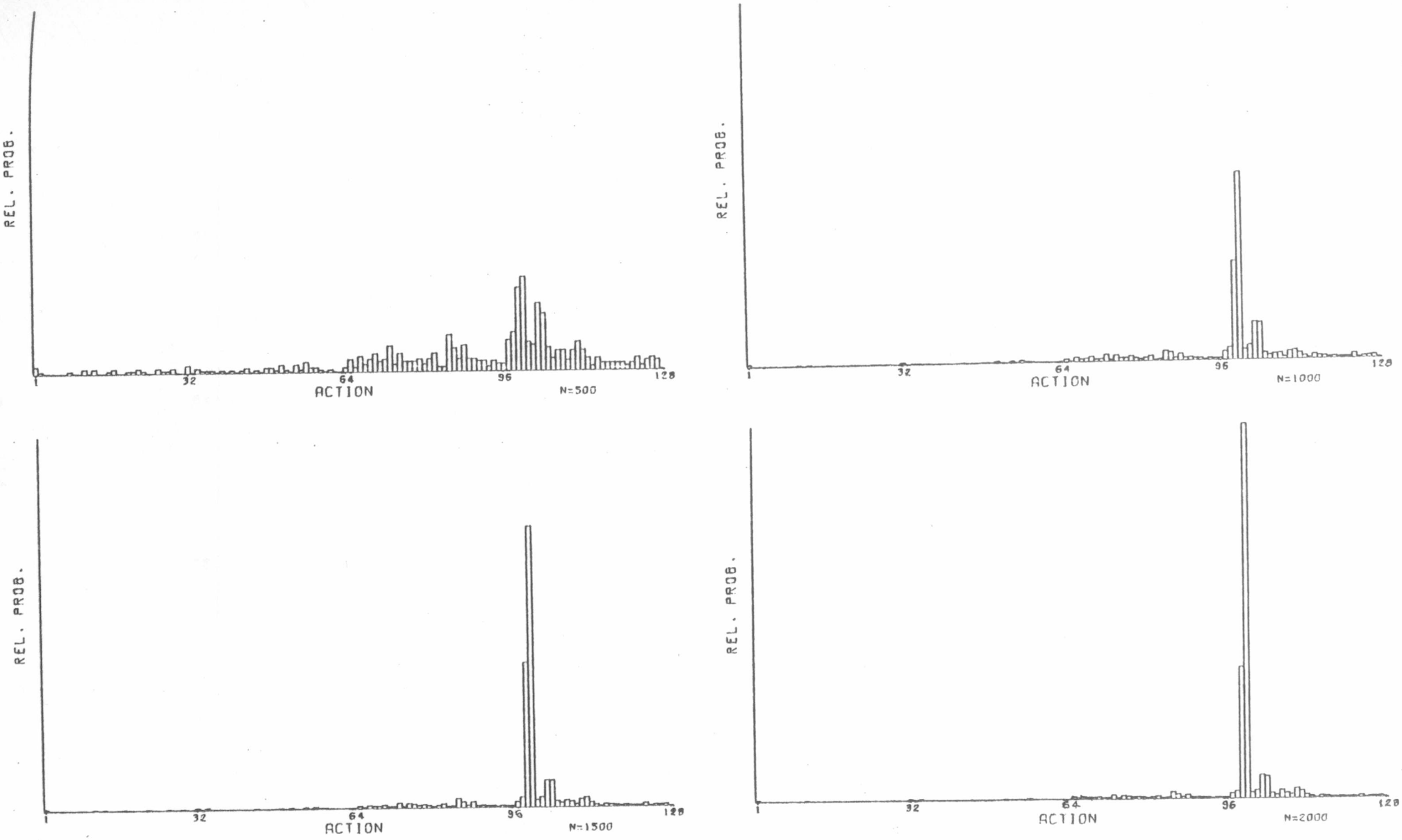


Figure 6.15 Distribution curves (1)

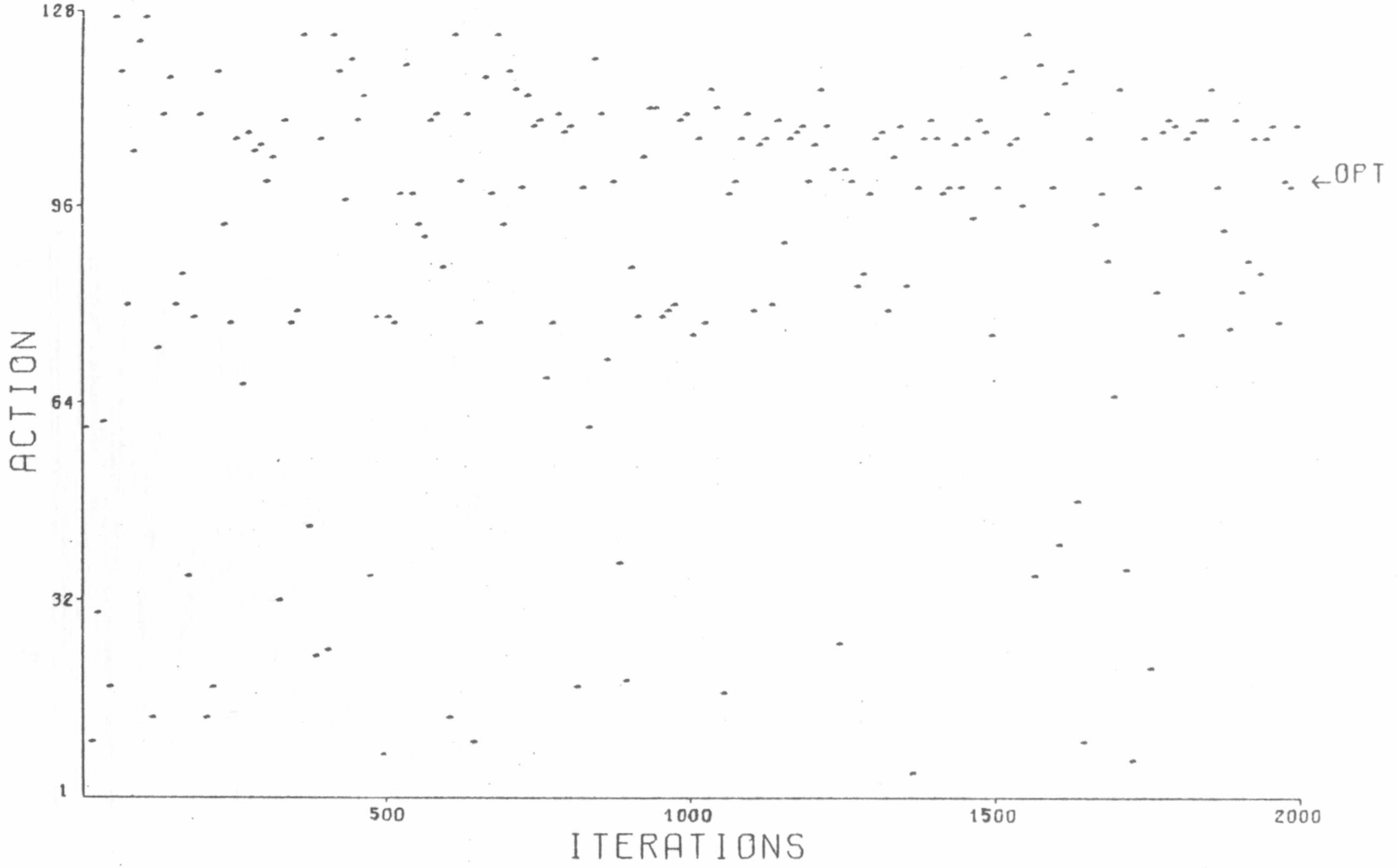
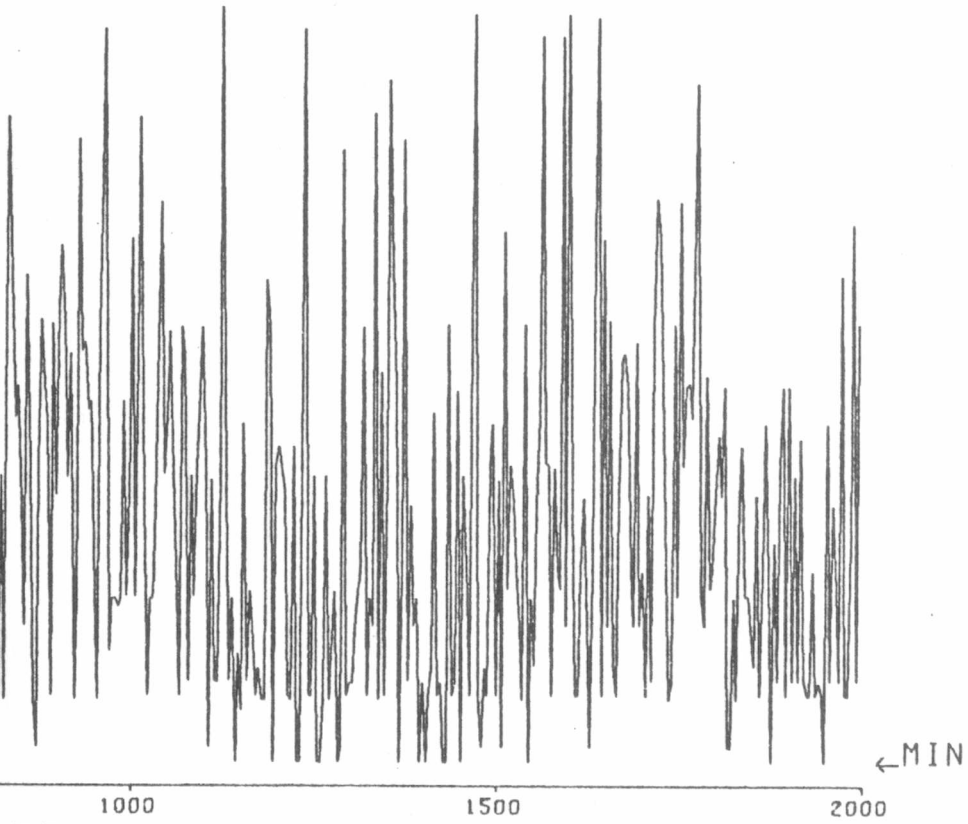


Figure 6.16 Output map (2)

Figure 6.



17 Penalty curve (2)

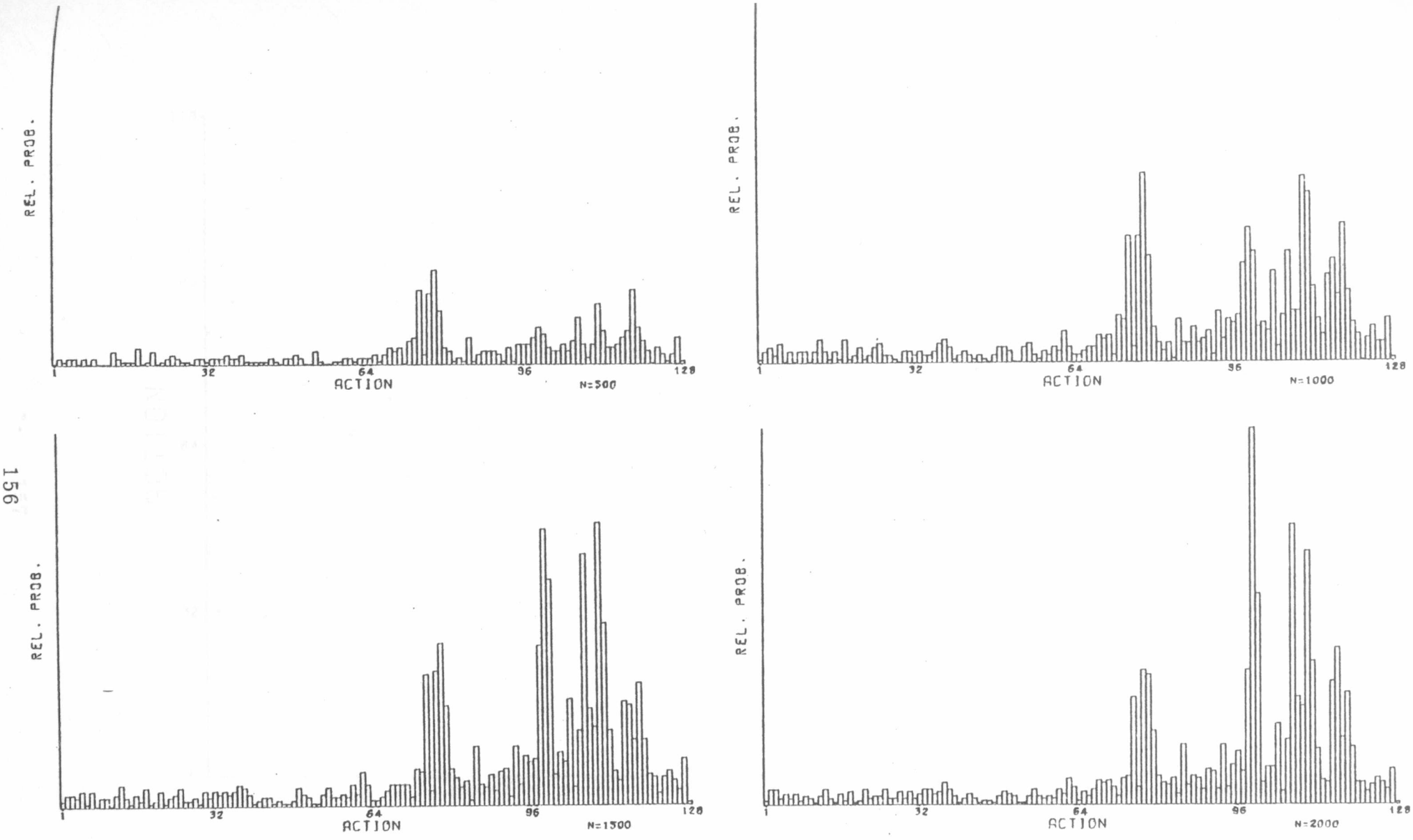


Figure 6.18 Distribution curves (2)

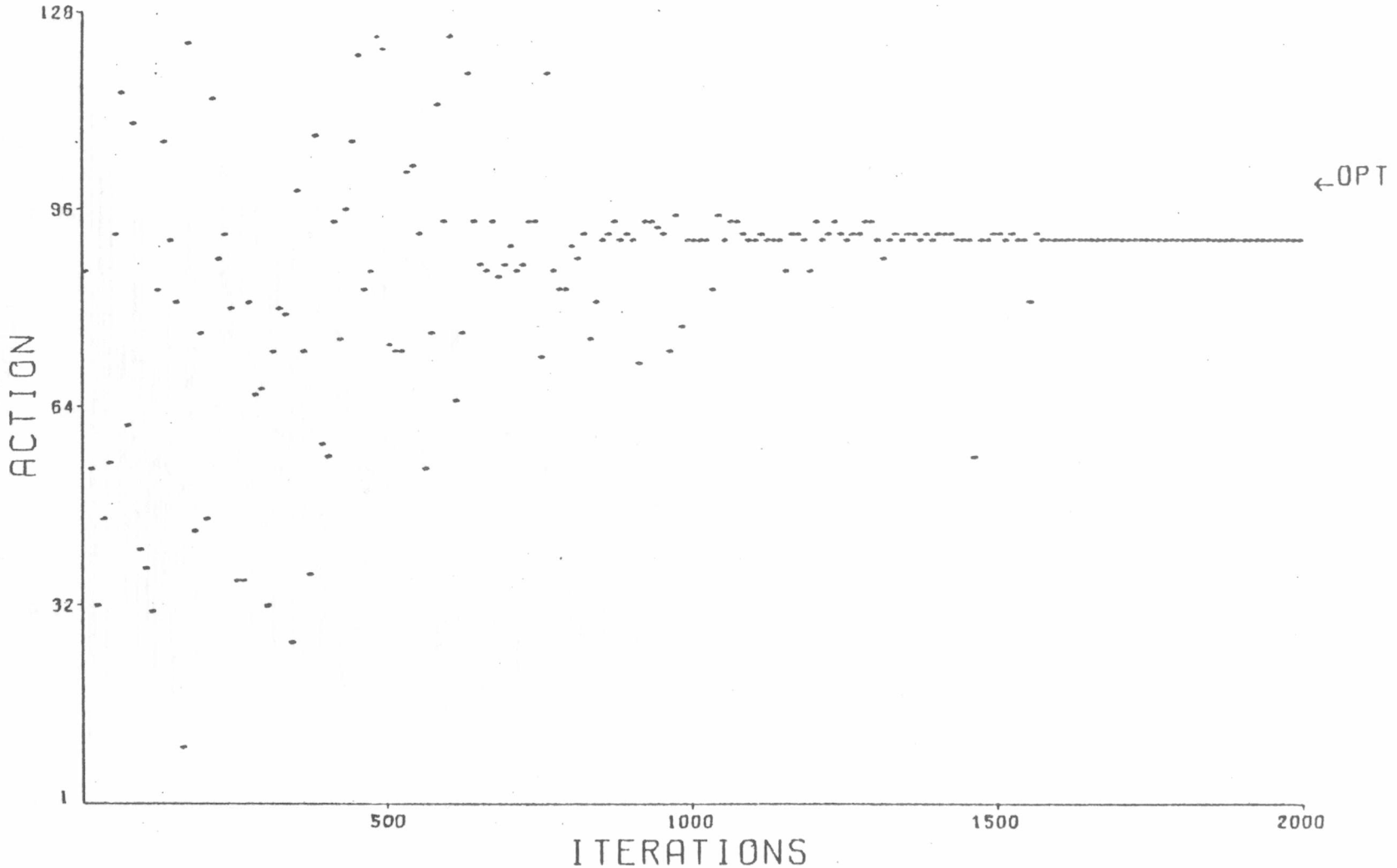


Figure 6.19 Output map (3)

1.00
0.75
0.50
0.25
0.00
REC. PENALTY
500

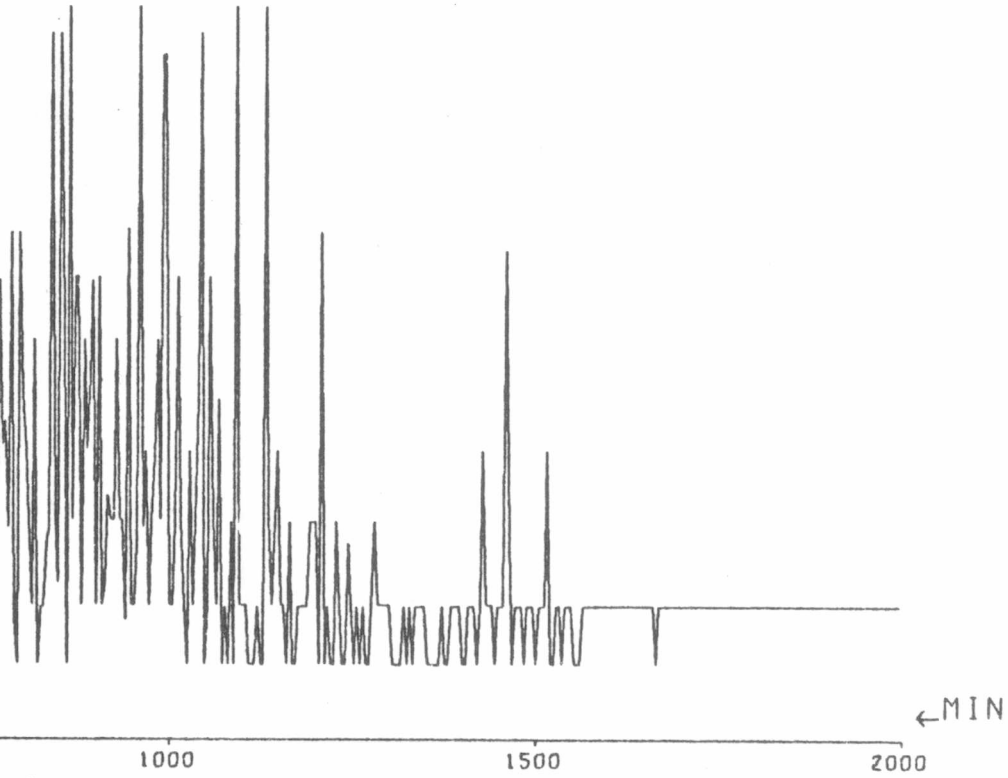
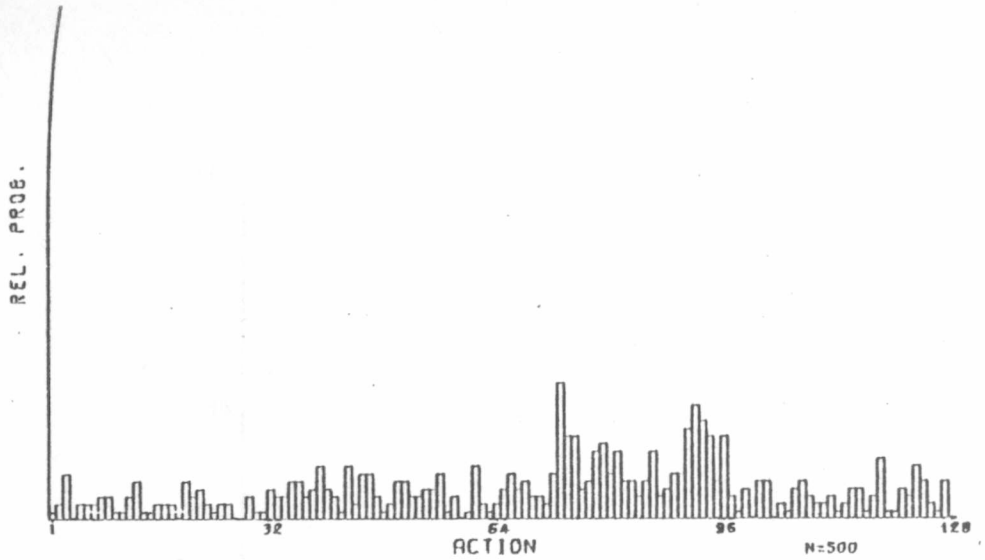
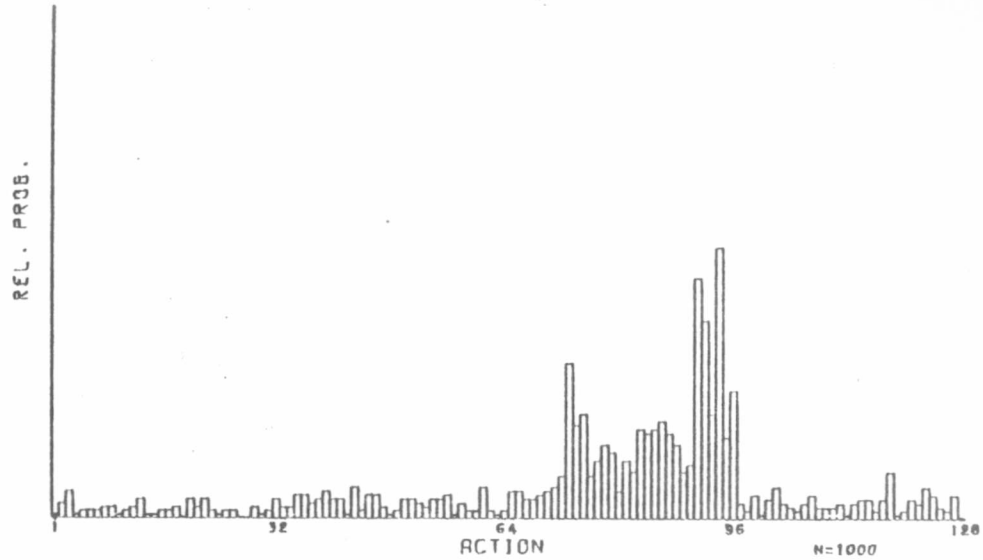
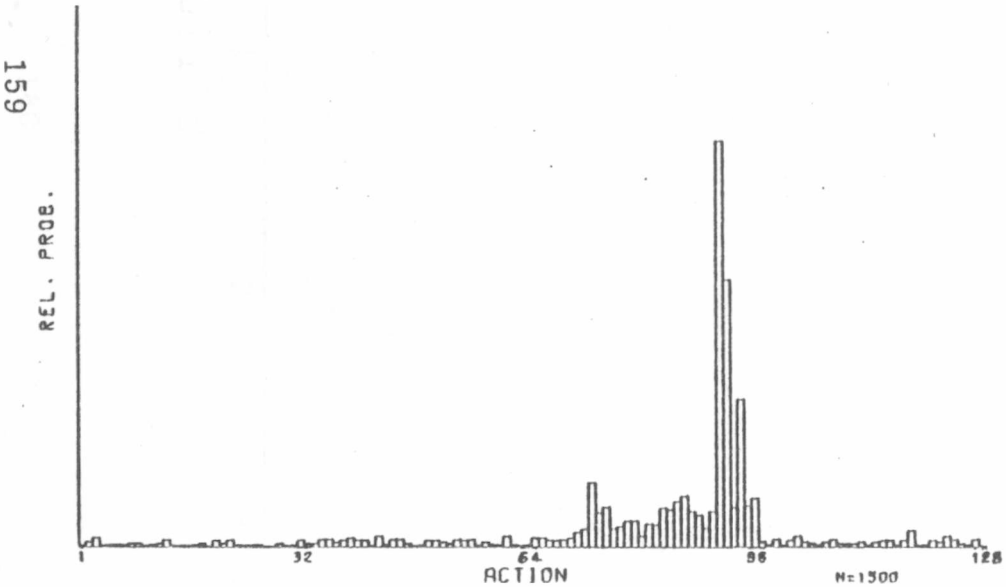


Figure 6.20 Penalty curve (3)







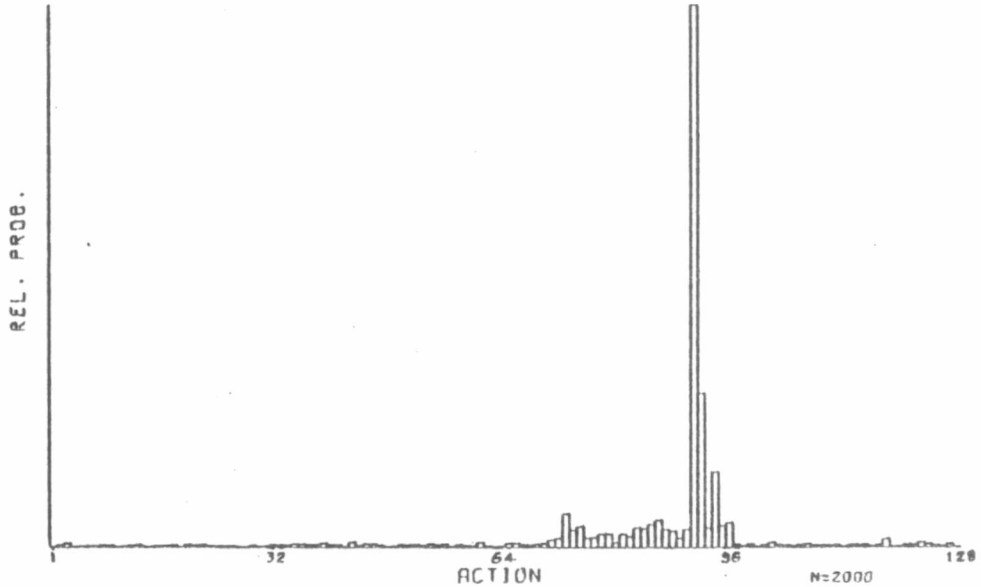


Figure 6.21 Distribution curves (3)

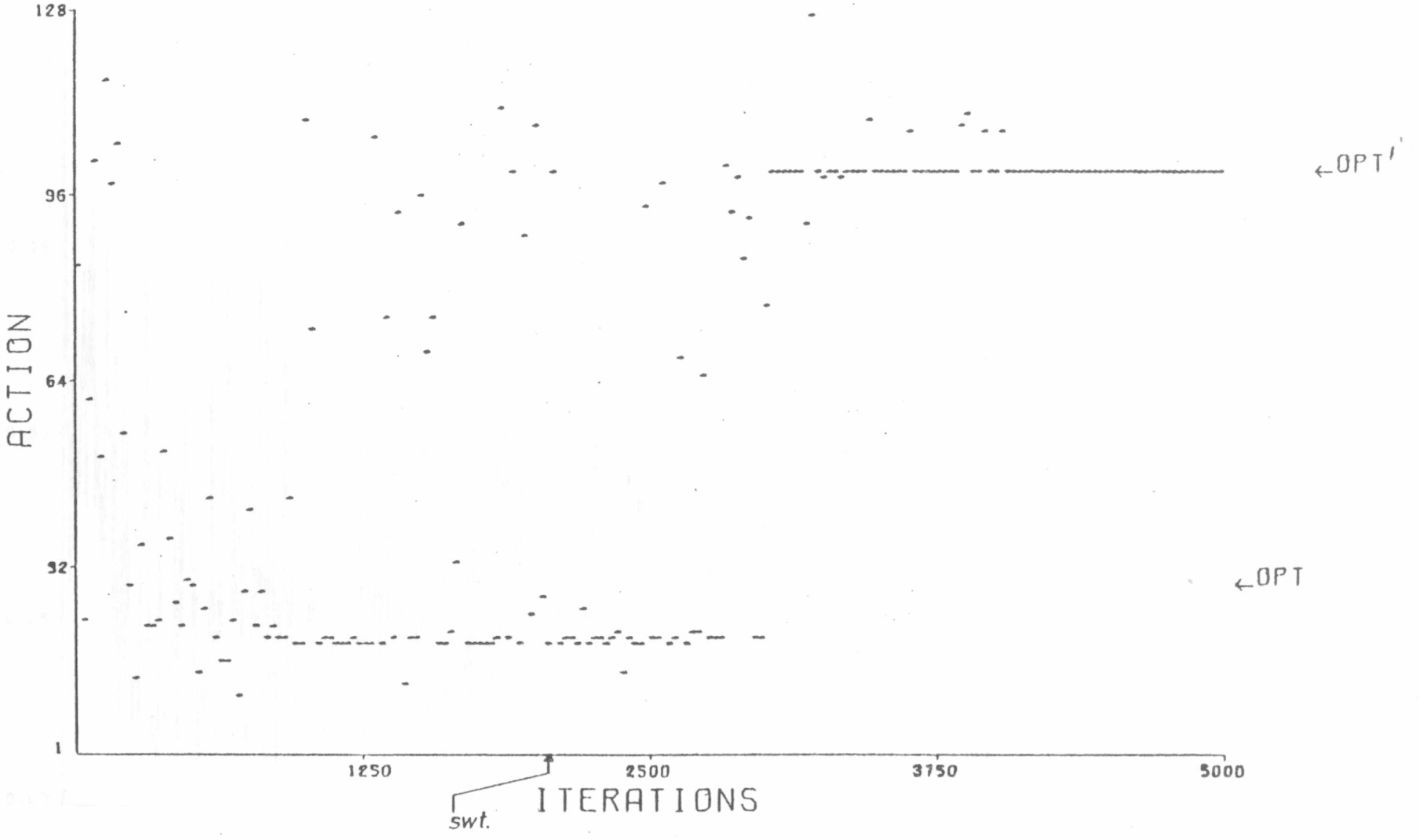


Figure 6·22 Output map (4)

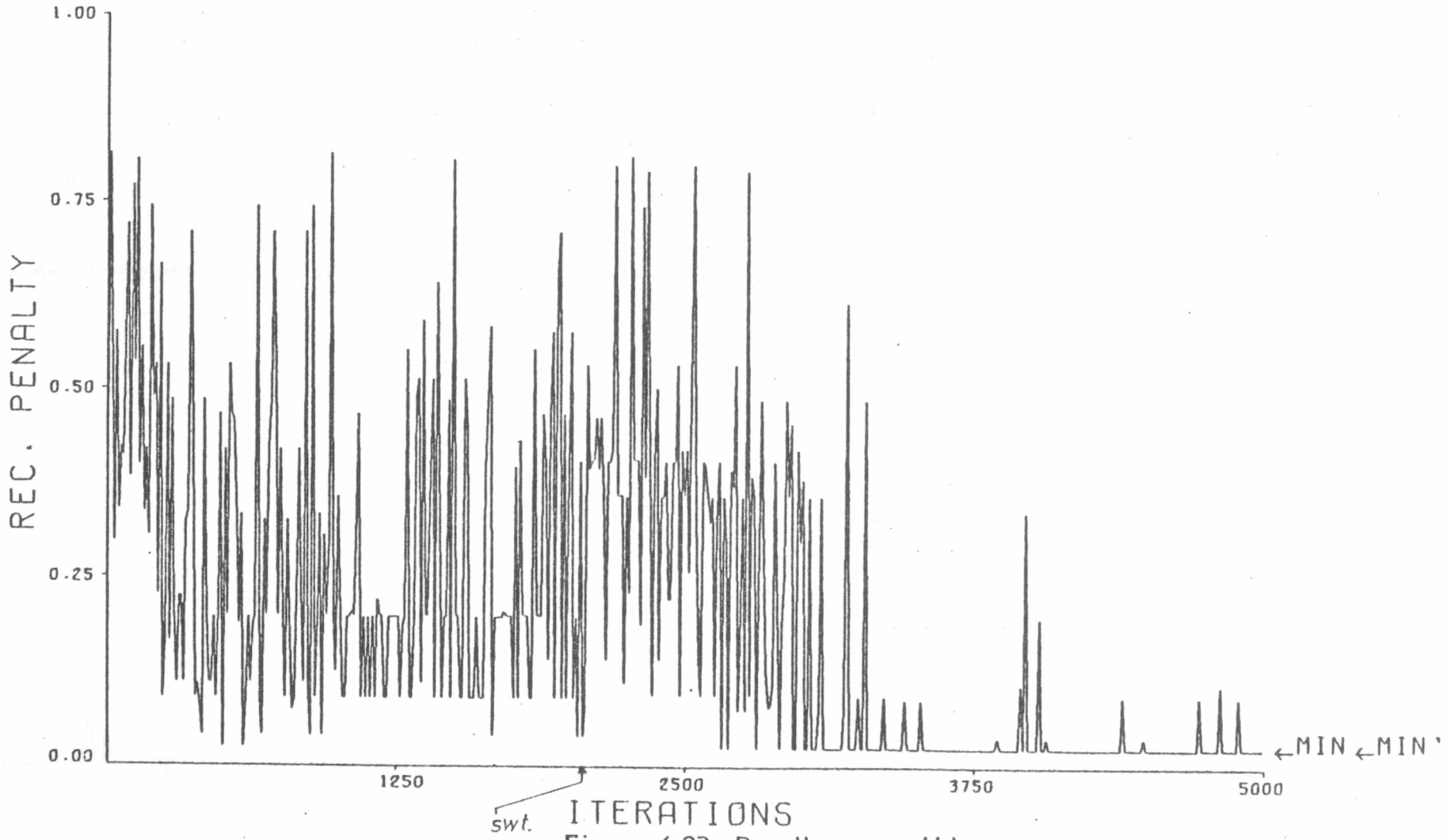


Figure 6.23 Penalty curve (4)

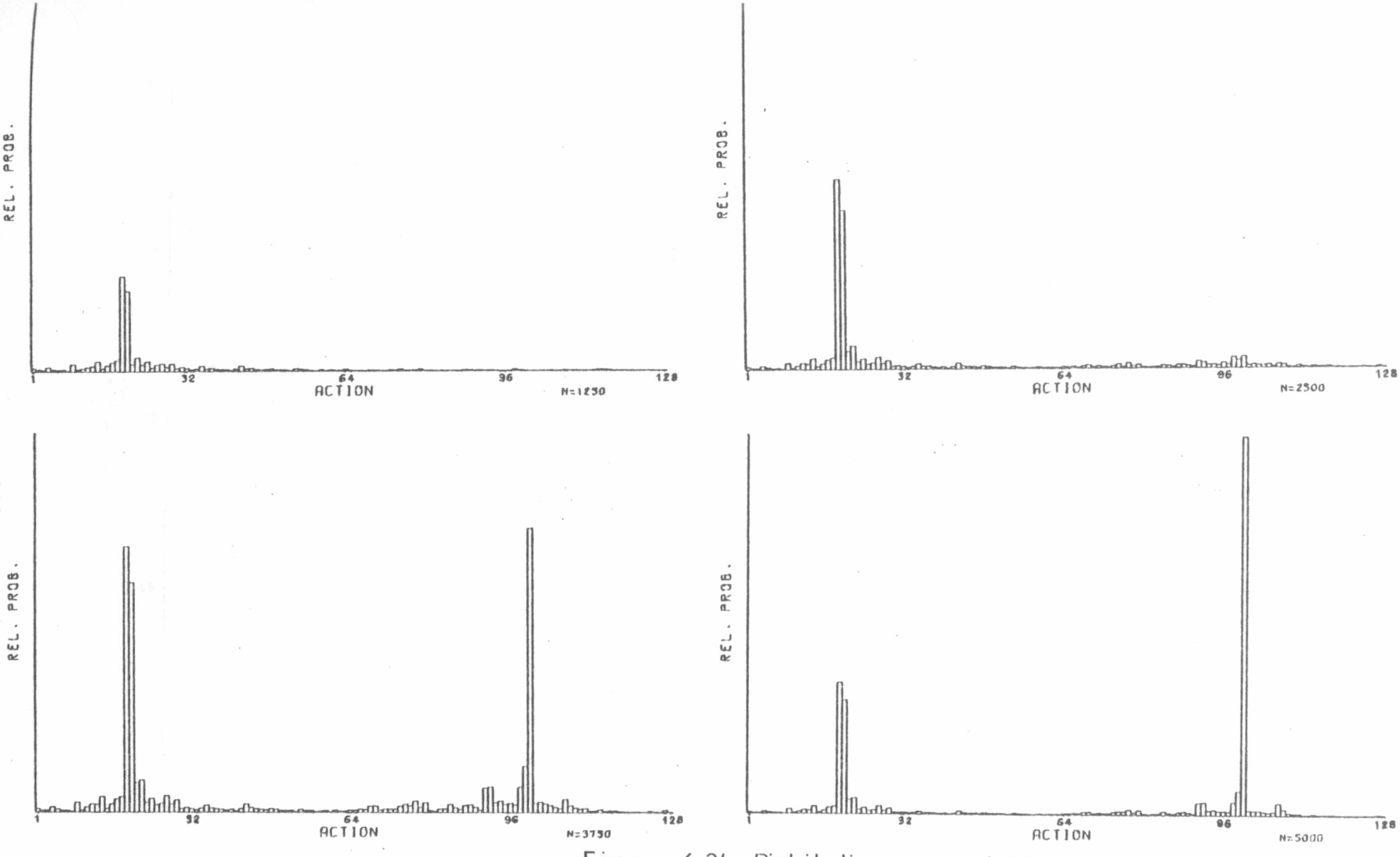


Figure 6.24 Distribution curves (4)

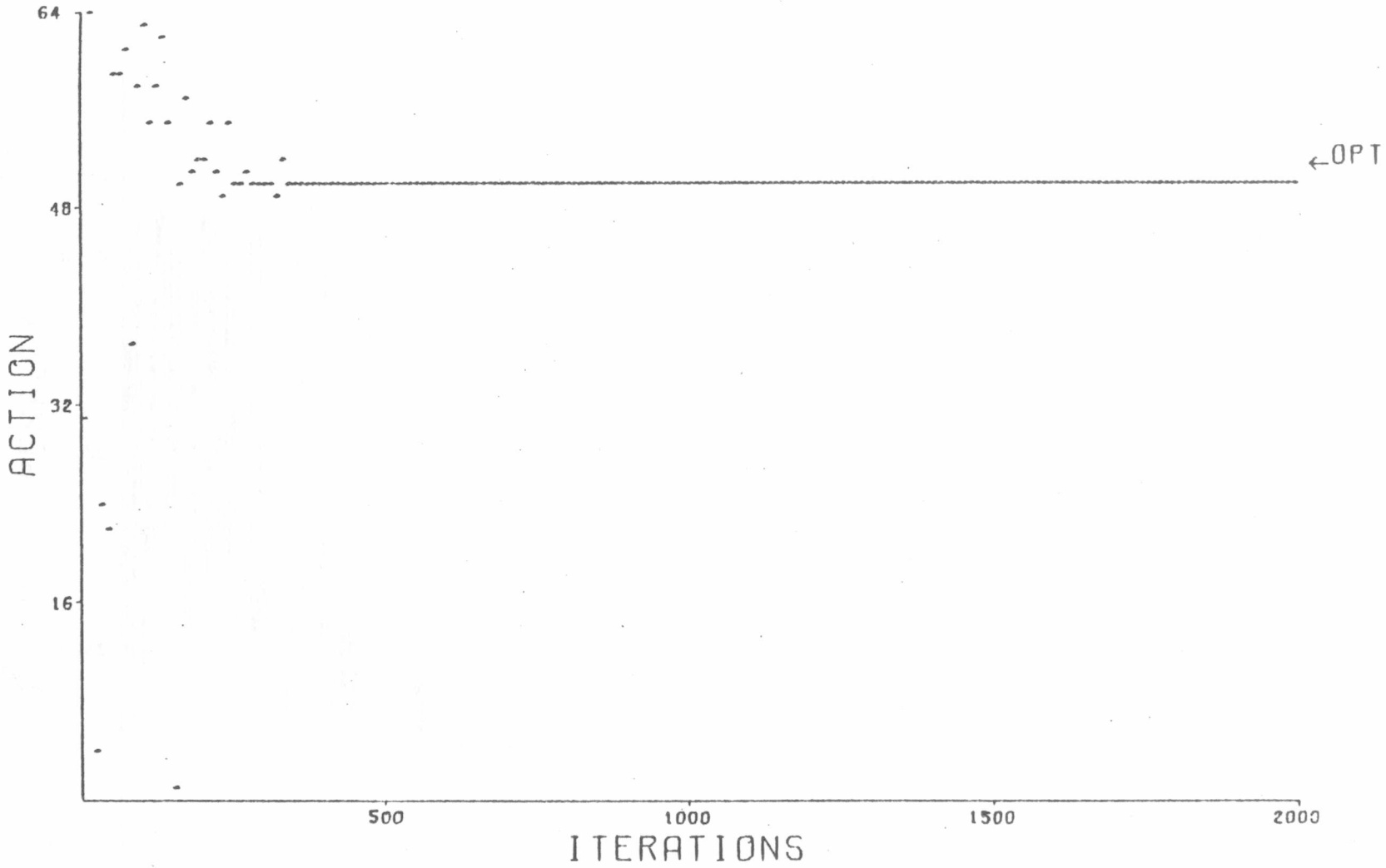
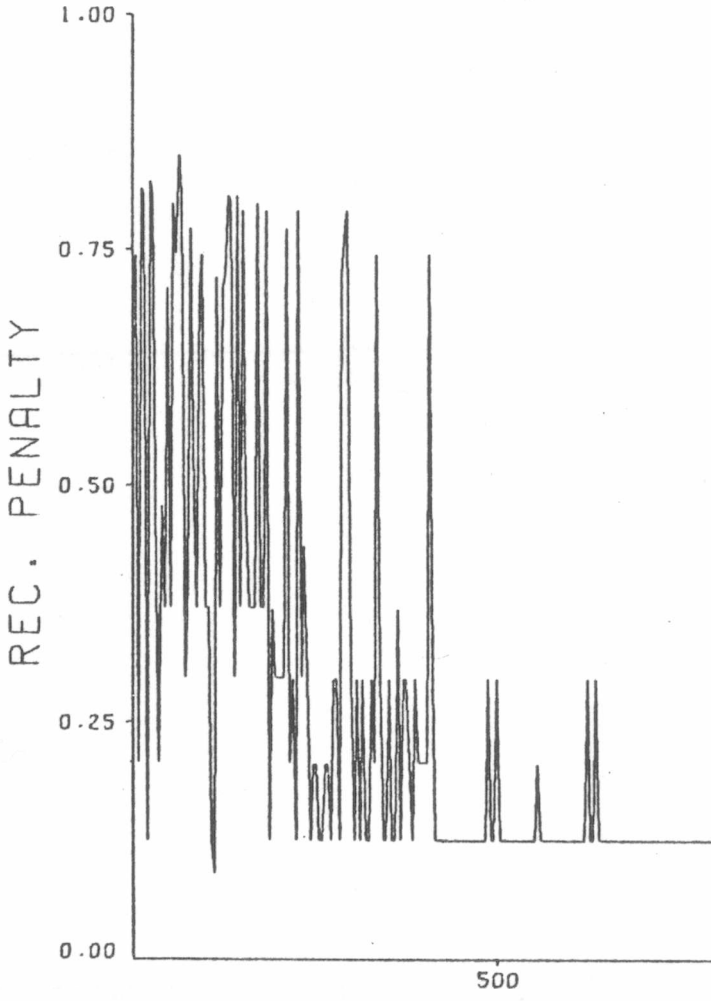
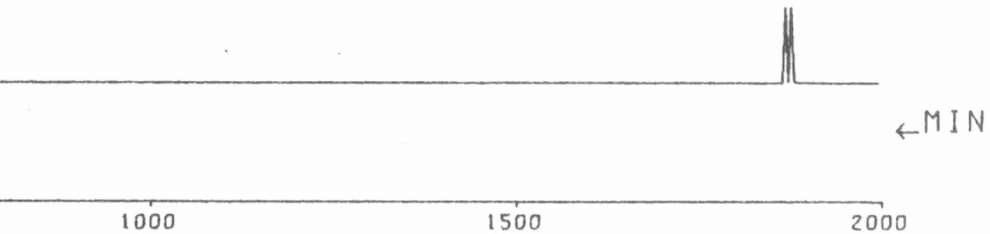


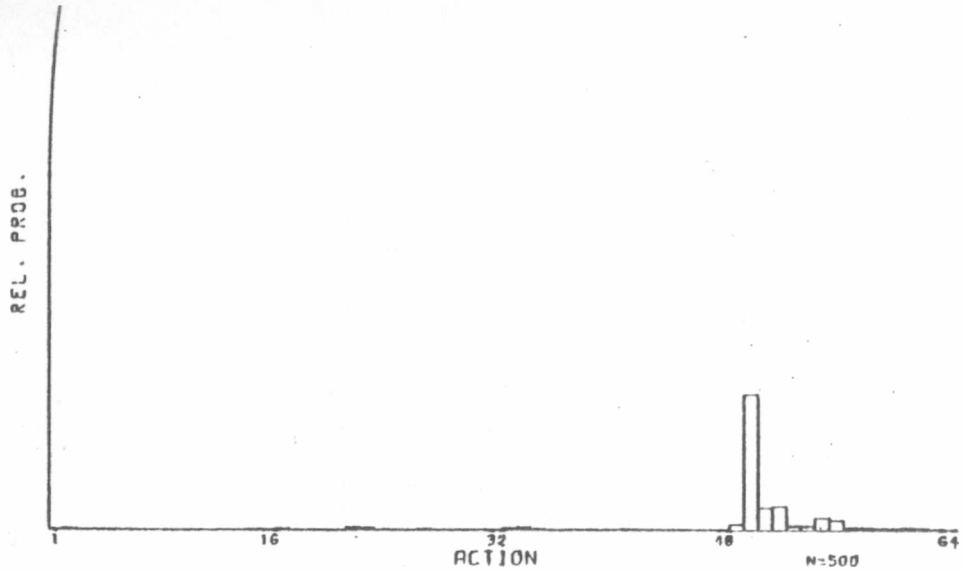
Figure 6.25 Output map (5)



Figure



6.26 Penalty curve(5)


REL. PROB.
1
16
32
48
64
ACTION
N=500

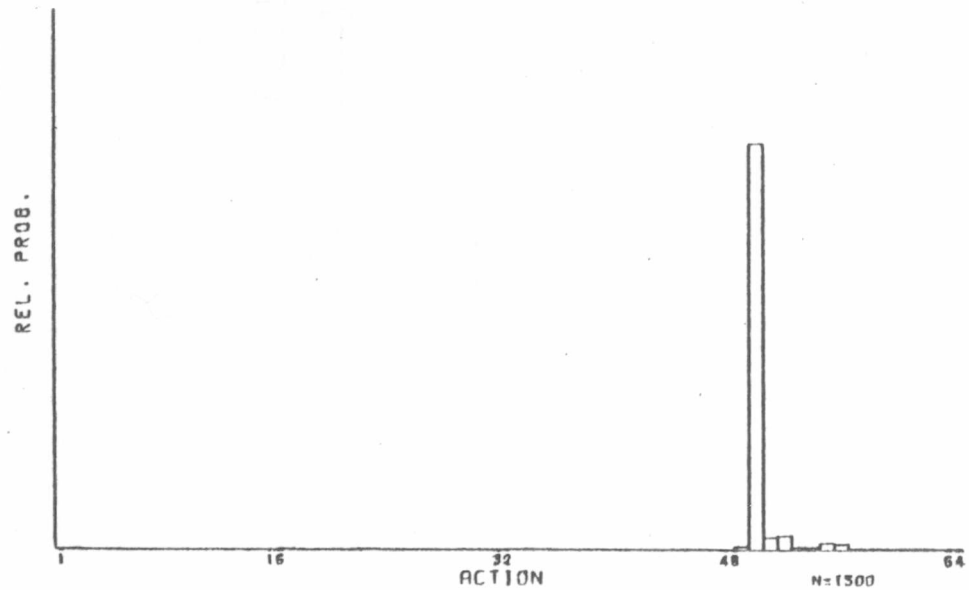
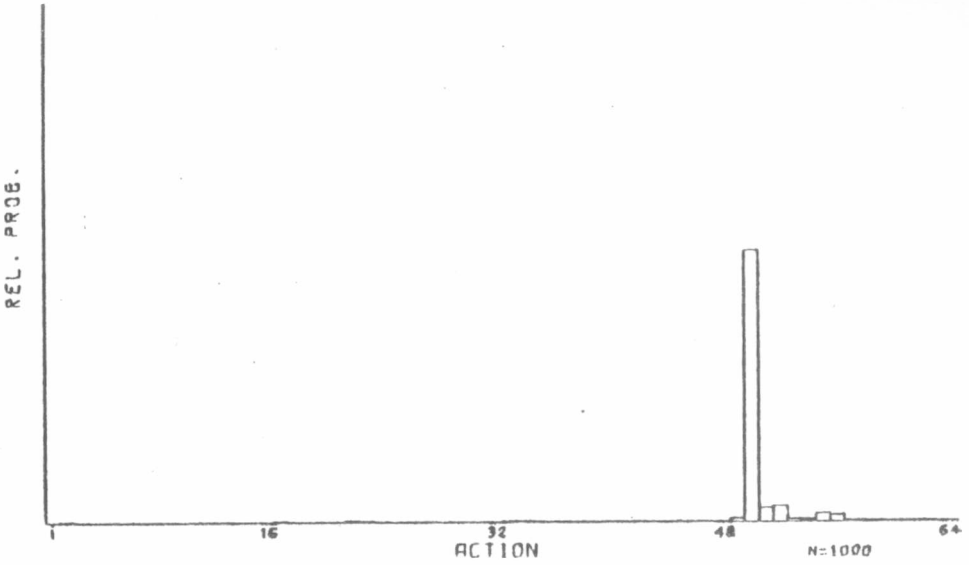
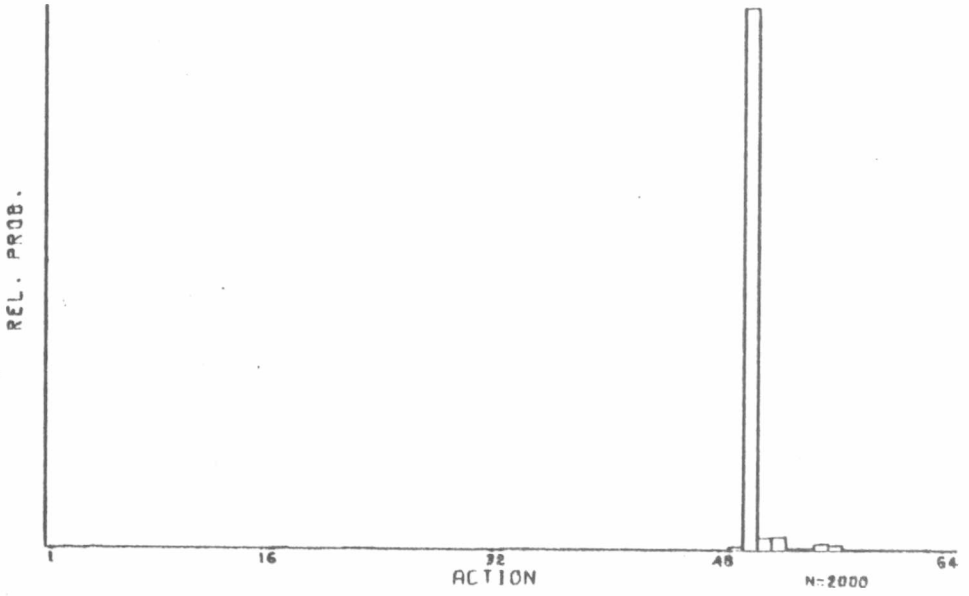

REL. PROB.
1
16
32
48
64
ACTION
N=1500

Figure 6.27





Distribution curves (5)

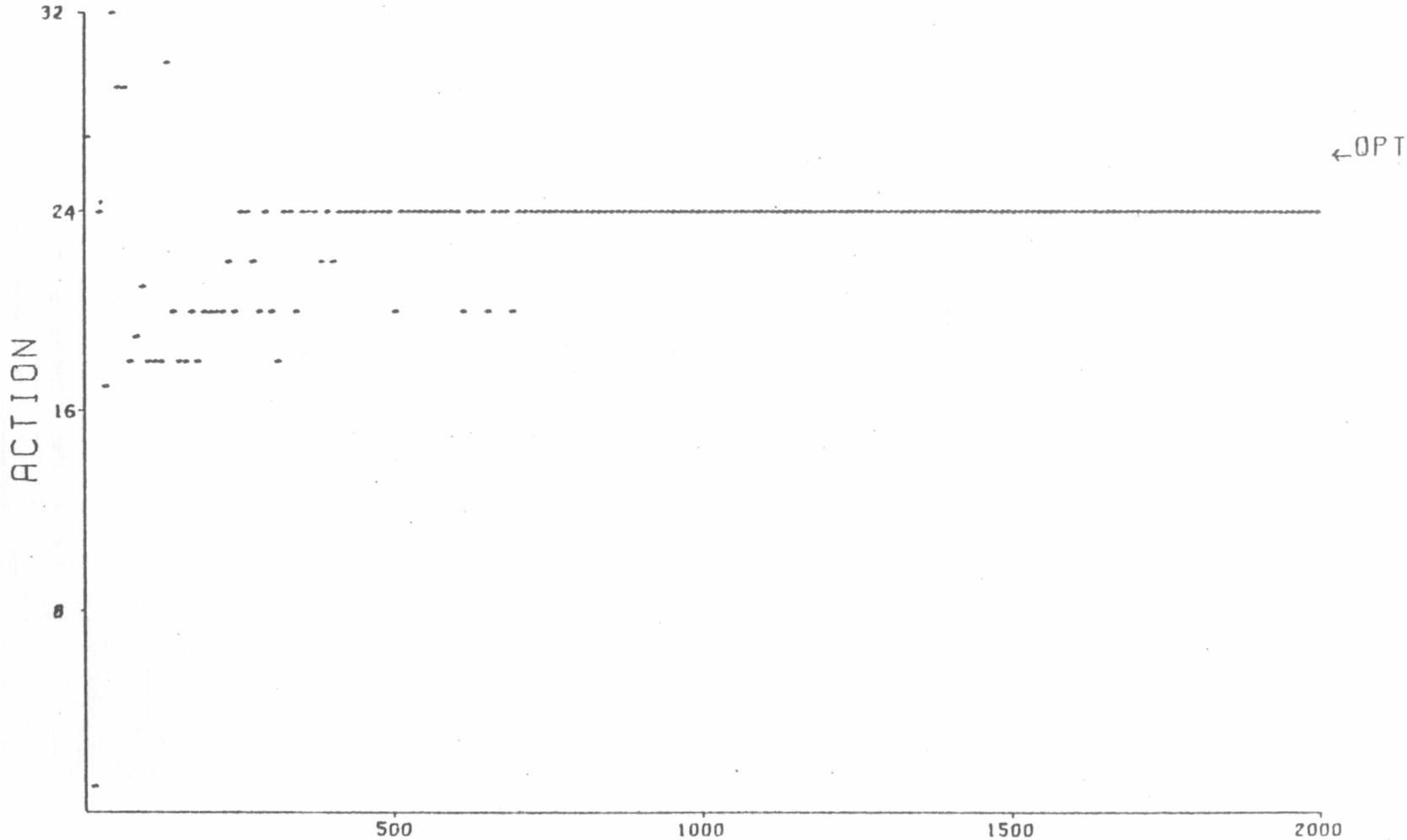


Figure 6.28 Output map (6)

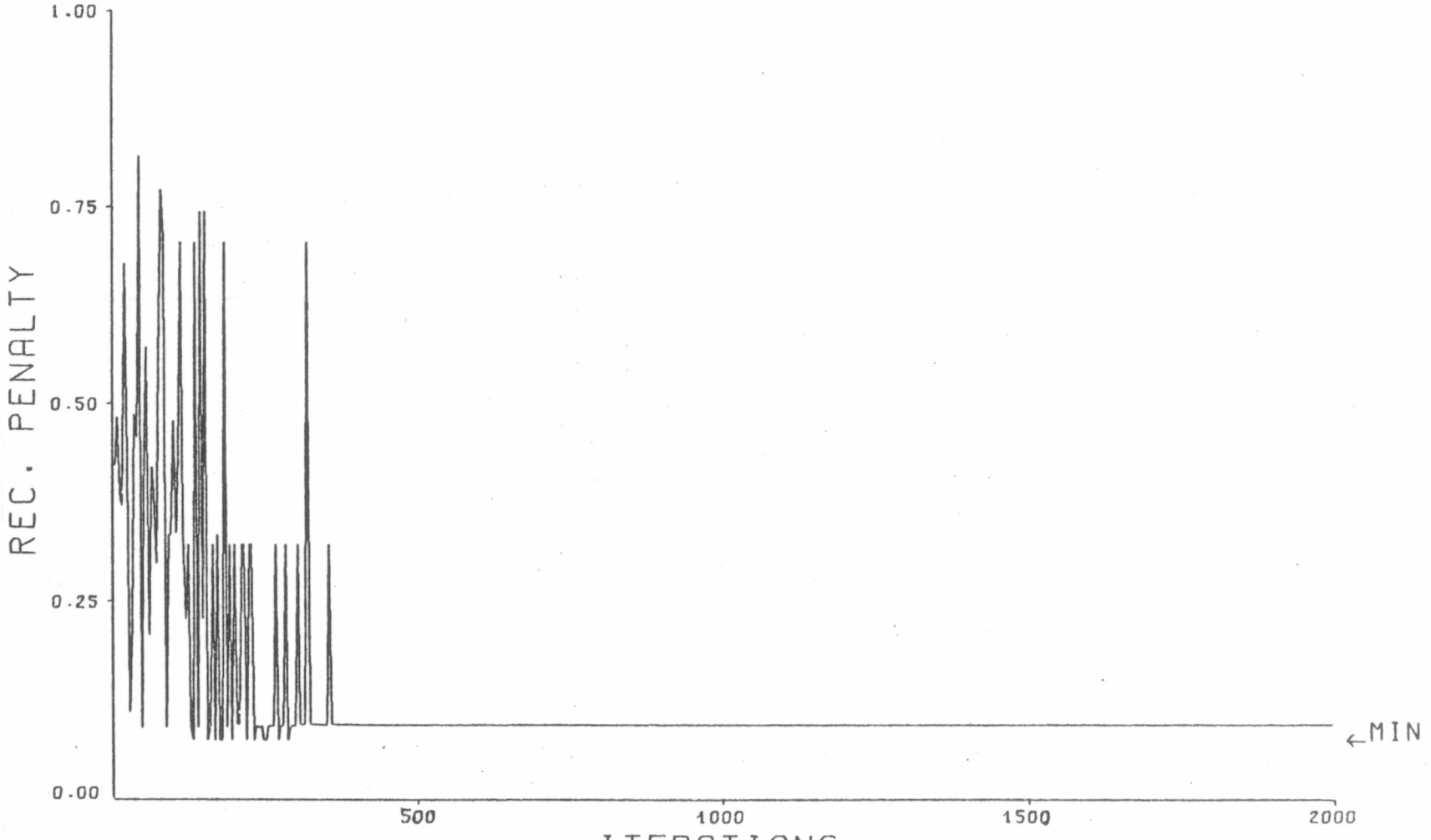
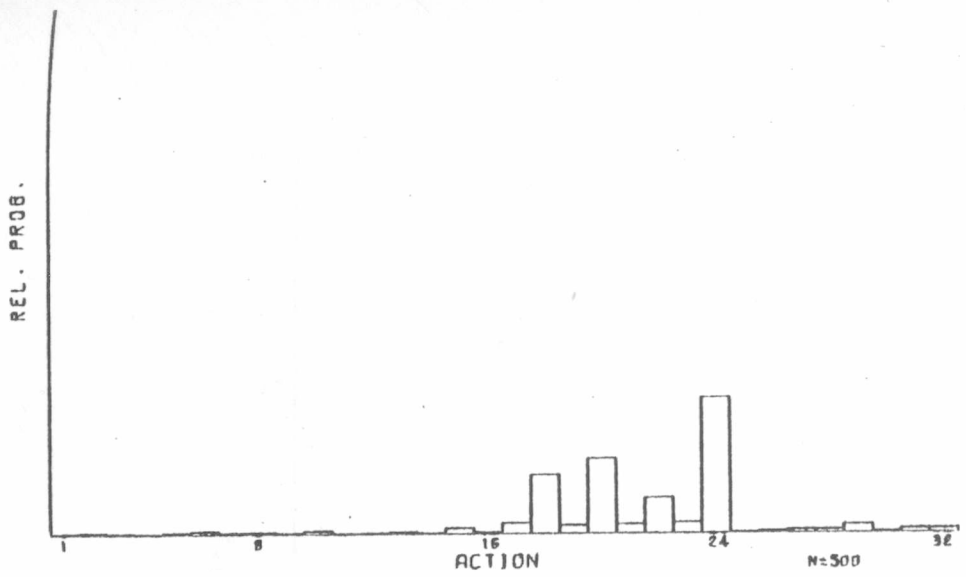
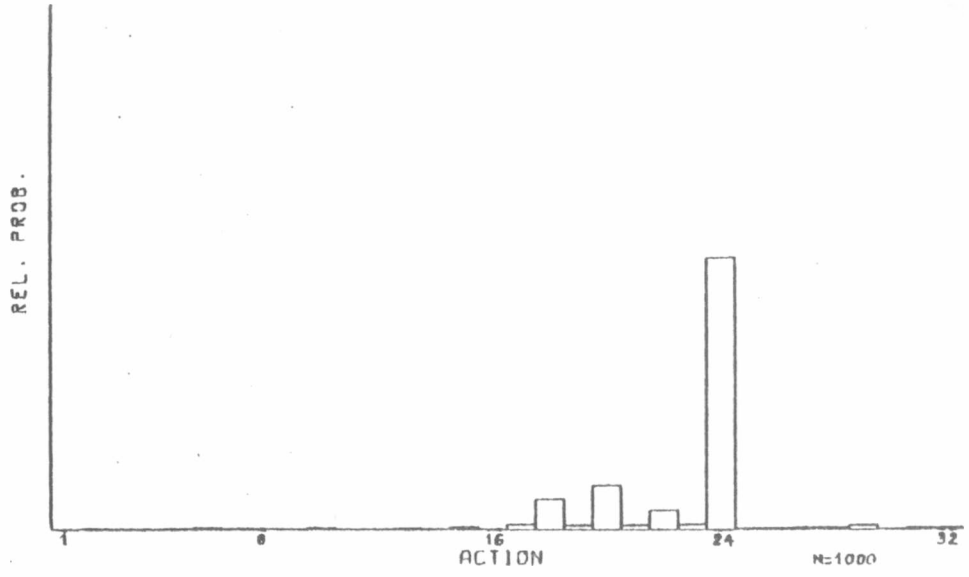
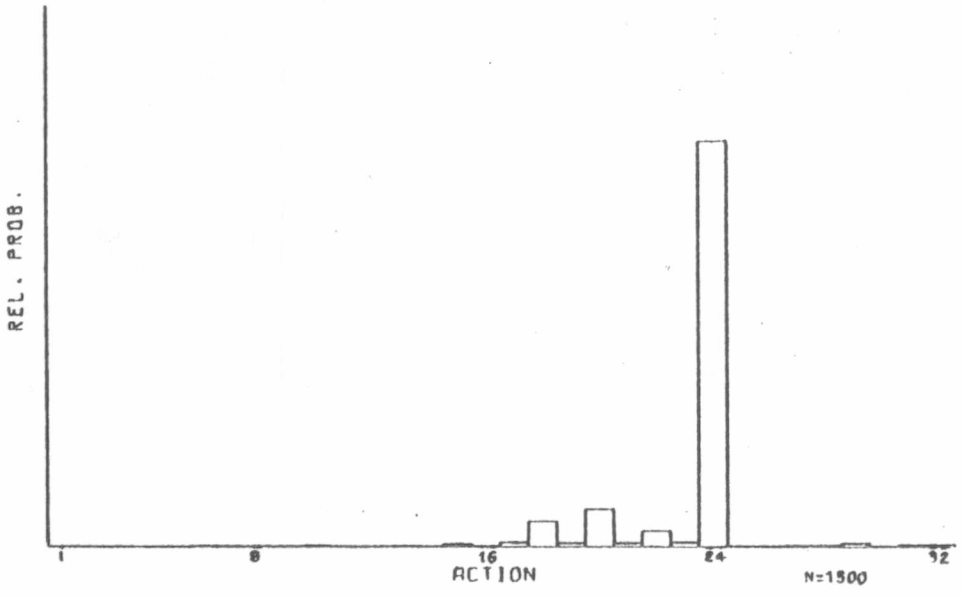


Figure 6.29 Penalty curve (6)







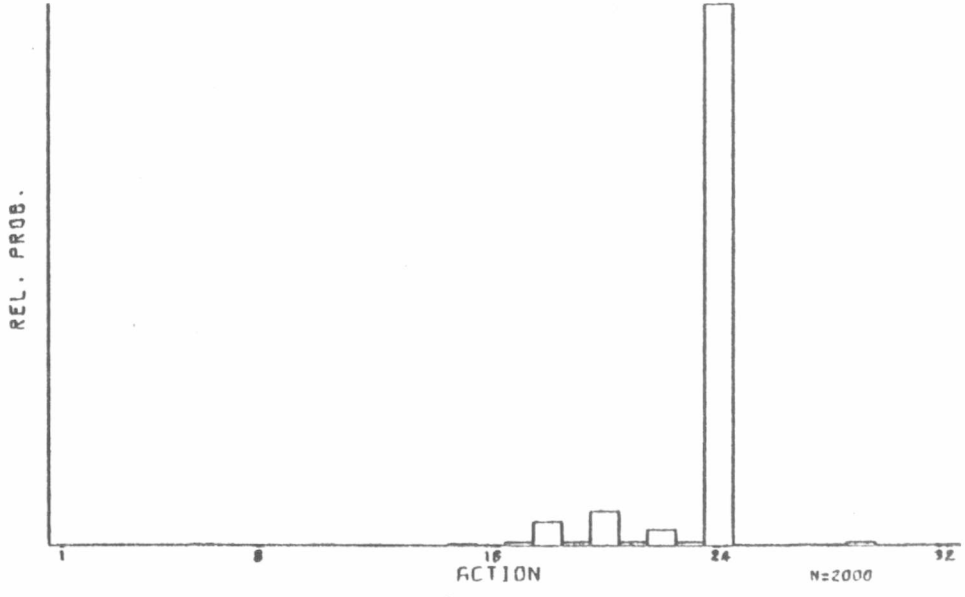


Figure 6.30 Distribution curves (6)

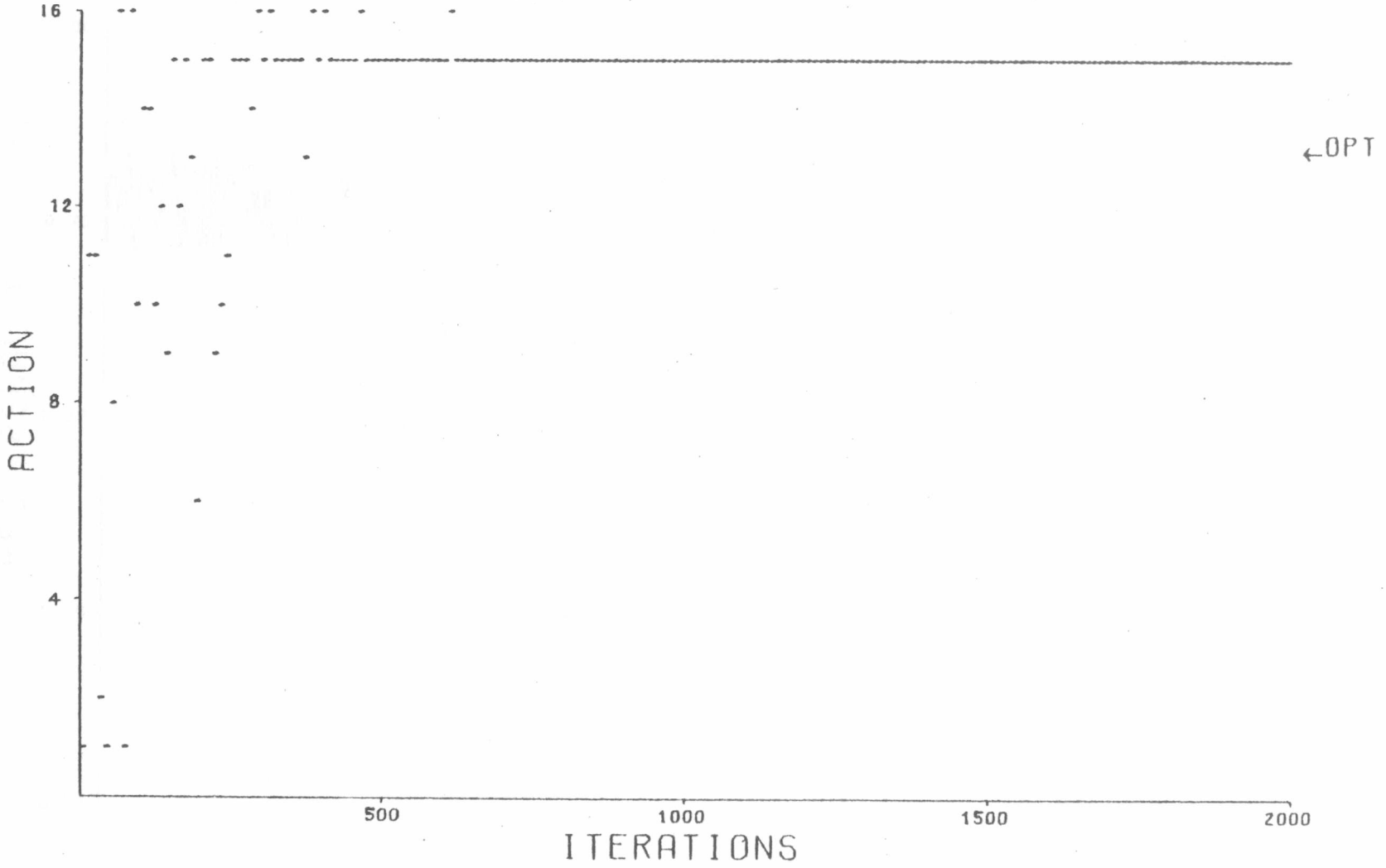


Figure 6.31 Output map (7)

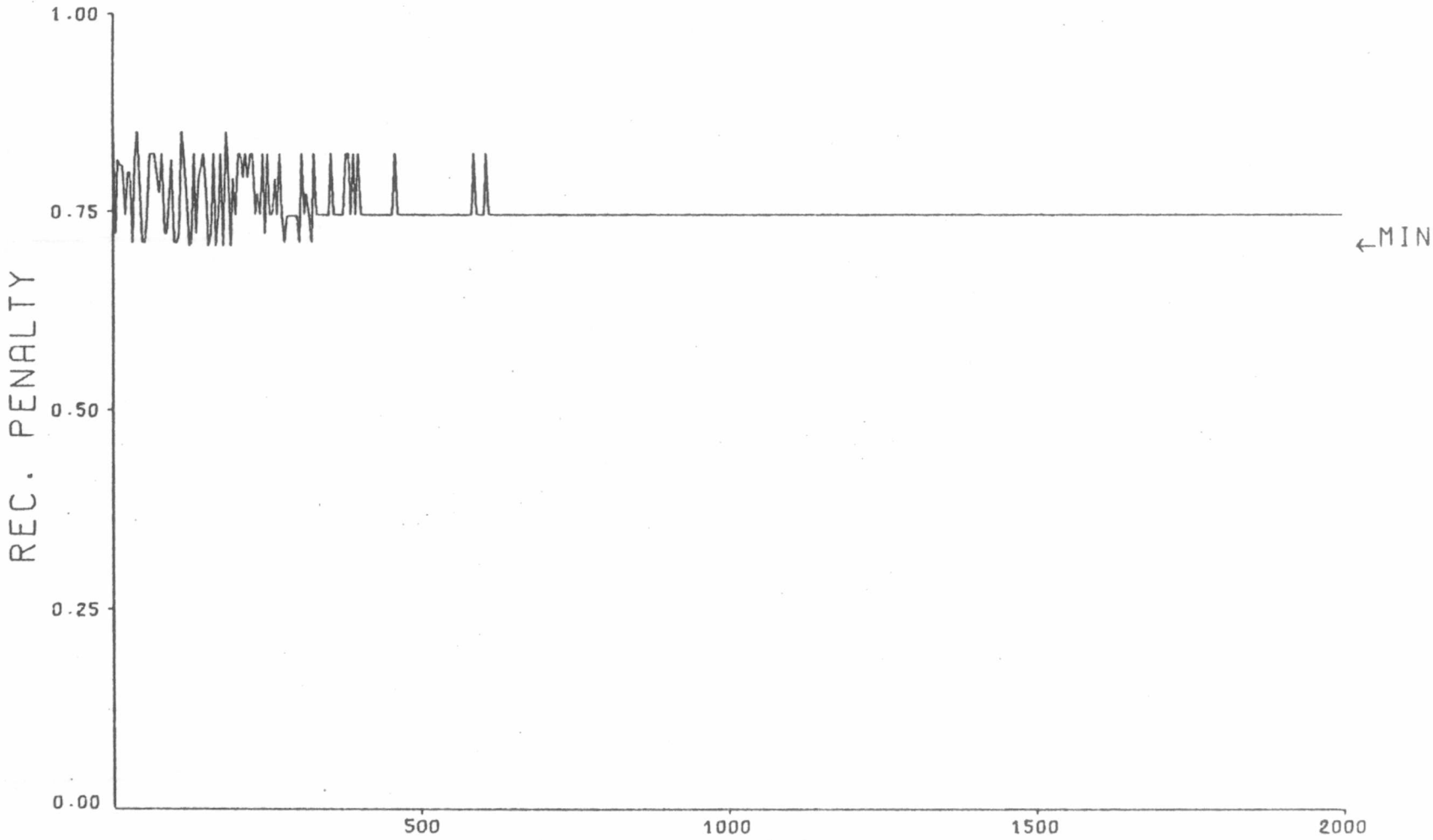
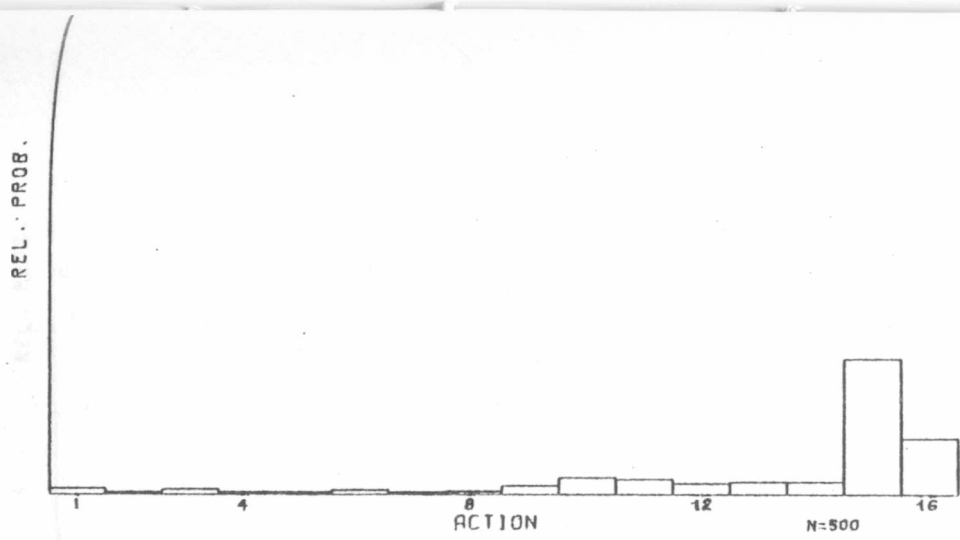


Figure 6.32 Penalty curve (7)



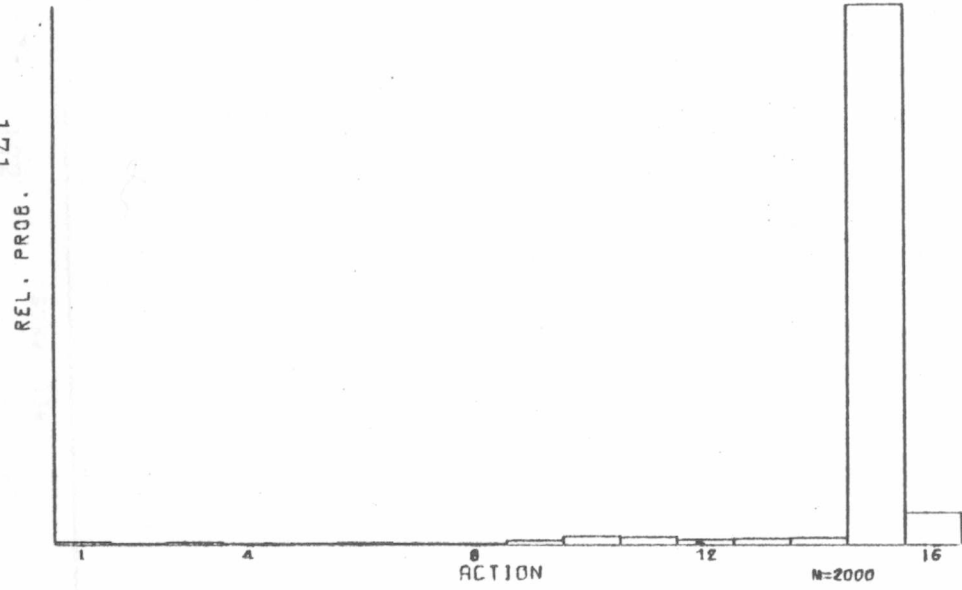
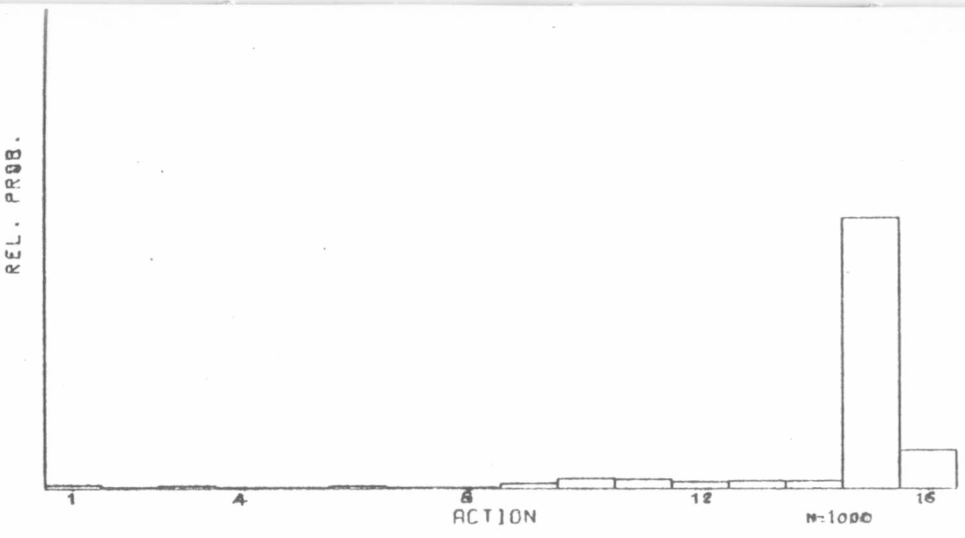
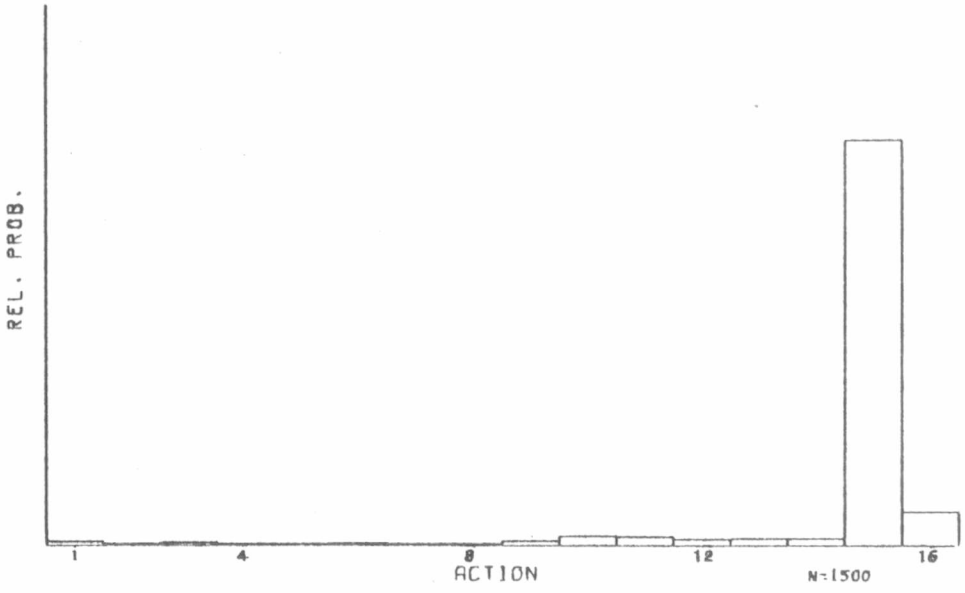
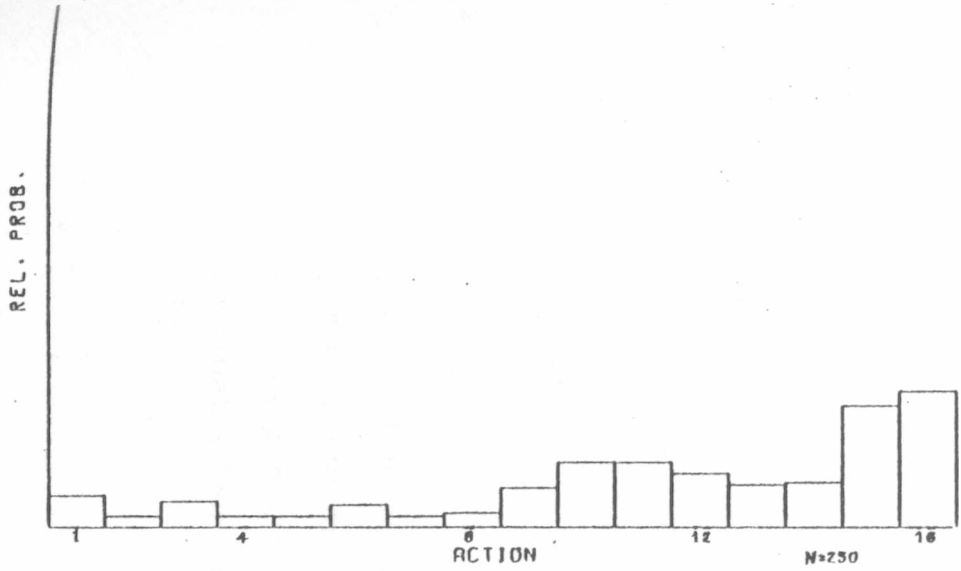


Figure 6.33





Distribution curves (7 a)



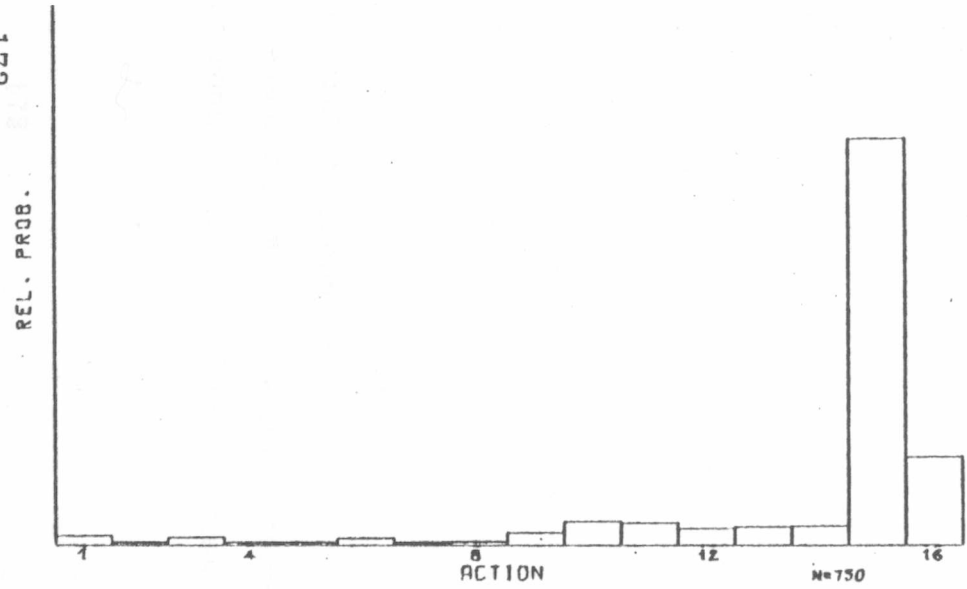
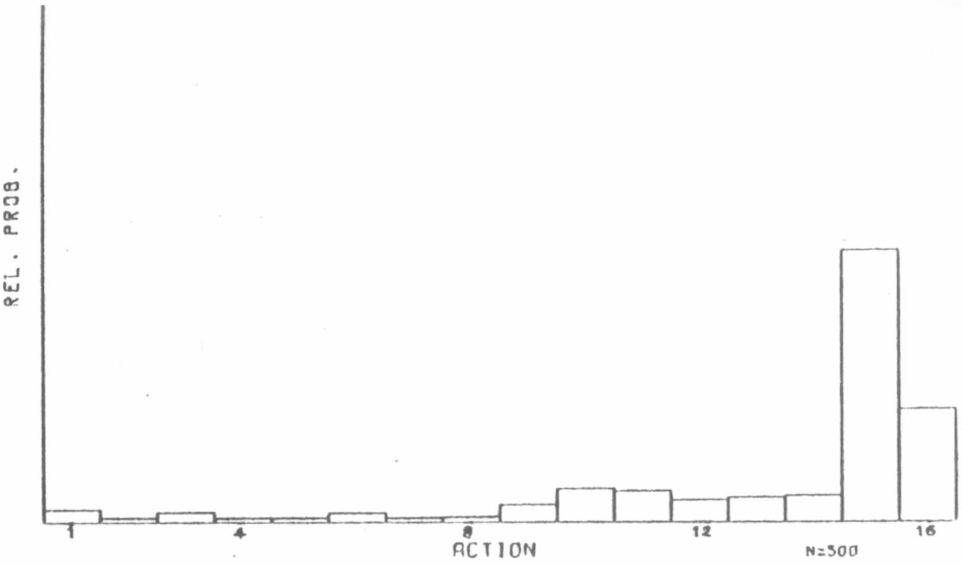
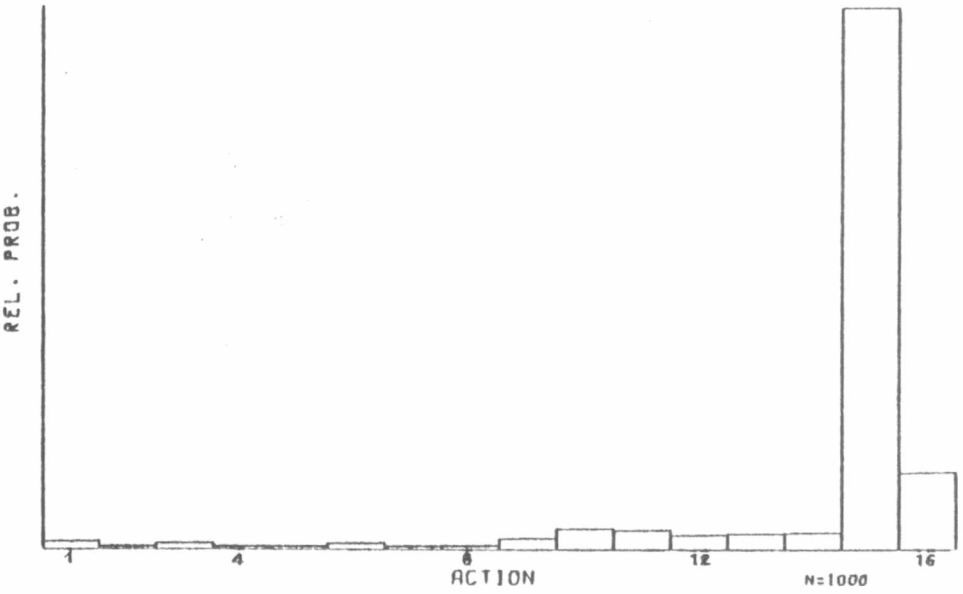


Figure 6.34





Distribution curves (7b)

CHAPTER 7

PROCESS CONTROL WITH A LEARNING AUTOMATON

7.1 Introduction

The results of the experiments presented in the previous chapters have confirmed that the hardware stochastic learning automaton has useful potential for applications in adaptive control systems. However, these were confined to simulated and of necessity artificial plant dynamics. It was felt, therefore, that a more convincing demonstration of automaton performance in this area would accrue from experiments which involved a real physical process.

While much theoretical study has been made of adaptive control strategies, conventional process controllers of the two term (proportional plus integral) or three term (proportional plus integral plus derivative) variety are still widely used in manufacturing industries. With these systems, there is usually a need for frequent, unpredictable adjustments to be carried out on-line. This "tuning" function, involving the simultaneous adjustment of several interdependent control parameters, could be performed by an automaton, and represents a fruitful area of application for a practical hardware system.

Adaptive control systems can be broadly classified as either plant measuring or performance measuring. Plant measuring implies the identification or observation of the parameters of the environment in order to develop a control strategy. Since the learning automaton is ideally suited to the situation where information about the environment is difficult or impossible to obtain, i.e., the classic "black box" problem, the control philosophy to be followed here will be performance measuring, where the only information available from the environment is in the form of some criterion of /

of performance, presented at the output port, in response to an input test signal such as a step or impulse.

It was eventually decided to base these applications studies on a small thermal process with a two term (proportional plus integral) controller. This was felt to be a suitable representation, on a laboratory scale, of the type of system commonly encountered in real-life situations.

7.2 Thermal Process

The thermal process used in these experiments was the laboratory tutor model PT 326 (manufactured by Feedback Ltd). This comprises of a centrifugal fan with an adjustable inlet orifice which blows air through a grid heater element, down a pipe, and out to the atmosphere. A miniature bead thermistor is used to monitor the exhaust air temperature, providing a measured value of process variable which is compared with the desired set-point in a detector bridge. This generates the resultant error signal or deviation which is fed to the controller. Control signal amplifier and heater drive circuits are incorporated within the unit, as is provision for external modulation of the set-point.

The characteristics of this process are approximated by a distance-velocity lag followed by two exponential lags, giving the following forward loop transfer function:

$$G(s) = \frac{e^{-sT_1}}{(1 + sT_2)^2} \left(a + \frac{b}{s}\right)$$

where $T_1 = 0.2$ sec , $T_2 = 0.26$ sec

A convenient measure of performance for this system is /

is the Integral of Squared Error (ISE) criterion resulting from the application of a step input. This criterion is particularly relevant here, since the error signal relates directly to the voltage applied to the heater. The ISE is therefore a measure of the energy consumption of the system in its response to a disturbance.

It was decided that a two-term controller would be used, to enable a useful range of parameter adjustment to be made with the hierarchical automaton. It is therefore instructive to consider the relationship between the ISE and the control parameters. A set of ISE contours for different settings of proportional band and integral time in the controller has been obtained[52], and is reproduced as Figure 7.1. The principal feature of this performance characteristic is a well-defined shallow ridge, which implies comparative insensitivity to proportional band setting in the region of the optimum: 69% proportional band, 0.85 sec integral time.

The complete system block diagram is presented in Figure 7.2. This shows the essential features of the process, together with its nested control system: an inner loop, consisting of a standard two-term controller, and an outer loop, containing the automaton, which evaluates the error separately and acts on the controller accordingly. In effect, the automaton sees the process and the controller combined as the 'environment'.

7.3 Control System

The two-term controller section is essentially straightforward, as shown by the circuit diagram in Figure 7.3. It consists of a front end buffer, followed by an integrator which connects via an inverter stage to the final /

final summing amplifier. Standard IC operational amplifiers are used throughout, with no special arrangements made for precision balancing or temperature compensation. In keeping with the spirit of the automaton approach, factors such as ambient temperature variations are placed in the category of random disturbances over which the designer can assume no control.

Integral time adjustment is achieved by varying the proportion of error signal which is integrated, while the proportional band setting is changed by altering the gain of the final summing amplifier. Both functions are controlled from the automaton by means of programmable attenuators (PA1 and PA2 respectively) as detailed in Figure 7.4. This shows a set of potential dividers (R_a:R_{bx}) activated by CMOS switches on the "earthy" side of each of the lower resistors. Selection is performed by means of two 3 to 8-line demultiplexers, addressed by the SLA output bus. Thus eight settings of proportional band (33% - 175%) and eight settings of integral time (0.5 sec - 8 sec) are available, controlled by the automaton in a 64-action (6-level) configuration.

7.4 Performance Evaluation

The performance evaluation section is of central importance to the automaton-plant interface. The function of this section, as discussed earlier in Chapter 2, is to quantify whatever measure of performance can be derived from the plant, and in turn formulate the requisite binary reward/penalty response for the automaton.

For these experiments, the performance evaluator is divided into three sub-sections, which are described with reference to Figure 7.5. The first section provides for the /

the calculation of ISE, and is implemented on a Solartron HS7 hybrid computer for accuracy and repeatability. The patch-up consists of a 4-quadrant multiplier unit in squaring mode, an integrator with facilities for external control of the reset and compute modes (via a comparator), and suitably scaled buffer amplifiers. The ISE output is then passed to an A/D converter via a potentiometer labelled "set range", which is adjusted initially to establish a convenient dynamic range for the converter. This would be supplemented in the case of a comprehensively adaptive system by an autoranging facility (see Section 2.8).

The overall operation of the system is supervised by the sequence control section, which consists basically of two monostables interacting with the SLA clock sequence to effect control of the integrator reset/compute modes in the HS7, operate the A/D converter, and generate a signal which is the source of the plant disturbance. The automaton itself can run at its normal fast clock frequency of the order of megahertz, while making use of the CKSTP/PLTRDY facility, described in Chapter 5, to arrest the system between each iteration.

The set-point input signal is a square wave with a period of 6 seconds, since it was found that a 3 second interval was sufficient for the plant to register its reaction to the disturbance, followed by 3 seconds in which to recover. A TTL signal from the sequence control section is applied to the MSB input of a D/A converter whose output voltage in turn provides the process set-point input. The addition of noise to the system is easily brought about by connecting lines from a PRBS generator to less significant bit inputs of the D/A converter. The effect of a step-wise disturbance on the process is shown in Figure 7.6. In /

In (a), no noise is present, and the plant response, mirrored in the received error signal, shows the typical features of lag and overshoot. In (b), noise has been superimposed on the set-point, and the additional disturbances, random in amplitude and duration, are clearly evident. The peak value of the noise envelope here is approximately 12% of the nominal step input amplitude.

7.5 Experimental Results

The overall performance of this learning control system was assessed by feeding the ISE output voltage from the HS7 to an X-Y chart recorder. A simple R-C low-pass filter was used to smooth out variations in the trace resulting from the long interval between each up-dated reading (6 secs), while the X-axis was driven by the output of a "system iterations" counter via a D/A converter to provide a suitable time scale.

A series of experiments was performed to illustrate various aspects of the learning controllers characteristics. The results are presented and discussed below. As before, these are "one-off" trials, chosen as representative examples of system behaviour.

Experiment 1: L_{R-P} scheme (Figure 7.7)

These error curves are the result of single learning runs obtained with the L_{R-P} reinforcement scheme. In (a), with no noise present, the system converged rapidly to a low level of steady state ISE after approximately 200 iterations, with little additional variance-induced error in the later stages. In (b), noise was superimposed on the set-point, causing the learning phase to be somewhat prolonged, /

prolonged, and a higher level of fluctuation in steady-state error.

Experiment 2: L_{R-I} scheme (Figure 7.8)

In similar fashion, these curves show the learning characteristics using the L_{R-I} scheme. Compared to the above, the initial learning phase was more erratic, but the system converged just as rapidly, with a virtually constant level of steady state error, which would be expected from this highly expedient reinforcement algorithm. In the case of superimposed noise (b), it is significant that the system has converged to a sub-optimal action, resulting in a higher final level of system error.

Clearly, the general features of these learning schemes which were observed in the earlier, static experiments are coming to the fore also in this practical system. The last result is particularly interesting, illustrating once again that the L_{R-I} automaton does lock on occasionally to a less than optimal action.

Further experiments were then carried out in which plant or controller disturbances were introduced, in order to simulate the situation of a non-stationary environment to which the automaton is particularly suited, and indeed which is most likely to be encountered in practice.

Experiment 3: L_{R-P} scheme, blower switch (Figure 7.9)

In this experiment, the blower inlet aperture was opened abruptly from 20^{o} to 60^{o}. The disturbance was set manually, when the initial learning phase was deemed to have advanced sufficiently to steady state conditions. In the no-noise case (a), the initial learning period was notably rapid. Immediately following the switch, there was a considerable /

considerable increase in received error, which subsequently receded as the automaton re-adjusted the controller. A similar result was obtained in (b) with noise added to the system. The main feature here is a generally higher average level of system error, though adjustment times in each case are virtually identical.

Experiment 4: L_{R-I} scheme, blower switch
(Figure 7.10)

The above experiment was then repeated with the L_{R-I} scheme. In the no-noise case (a), the reaction to the disturbance was rapidly controlled, and the final error value very low. With added noise (b), the system exhibited longer adjustment times and a higher overall level of ISE. Again, the almost constant level of steady state error with this scheme is apparent.

Experiment 5: L_{R-I} scheme, controller switch
(Figure 7.11)

As a further test of the adaptability of the automaton, a more drastic disturbance was introduced into the system by switching the controller characteristics. This was achieved by changing over two address lines at the inputs to the programmable attenuators, one for the proportional band selector and one for the integral time selector, thereby completely restructuring the relationship between automaton actions and controller settings.

In curve (a), the reaction to the controller disturbance was swiftly compensated. Furthermore, it appears that in this case the reconfigured controller was able to lock on to a better control setting, as indicated by the lower error level of the latter portion of the trace.

With /

With noise added (b), the reaction to the disturbance was noticeably more erratic. However, the time taken to re-adjust did not appear to be significantly longer than in the previous example.

Experiment 6: L_{R-I} and L_{R-P} schemes, controller switch (Figure 7.12)

This final experiment gives an excellent illustration of the pitfalls involved in using the L_{R-I} scheme, rather than L_{R-P}, in a non-stationary environment. In (a) controller switching was performed on an L_{R-I} system. It is clear that the automaton simply did not respond to the change in conditions. Since the current choice of action, and therefore the control setting, was no longer suitable, the ISE rose to a higher level, at which it remained locked within the confines of this experimental log (512 iterations). The jitter on the trace is mainly a result of the R-C filter on the chart recorder input.

In (b), a direct comparison can be made with the performance of an L_{R-P} system under identical conditions. The initial adjustment phase was longer than the above, while the initial reaction to the disturbance was very similar. However, the all-important difference is that the L_{R-P} automaton was able to re-adjust in the light of the changed conditions, thereby restoring the system operating error to its original level.

7.6 Conclusions

With these experiments, a practical process controller with stochastic learning automaton supervision has been shown to operate successfully. No a priori information was assumed in the application of the automaton to /

to the system, save for establishing the dynamic range at the input of the simplified, non-adaptive performance evaluator (Section 7.4).

The system gave a highly satisfactory demonstration of adaptive control, achieving minimisation of error under stationary and non-stationary plant conditions. In this context, the disturbances applied were quite severe. In most physical plant the system parameters vary slowly and continuously, calling for a tracking ability from the automaton rather than the comprehensive re-adjustment of which it has been shown nevertheless to be fully capable.

As before, the intrusion of externally derived noise produced only a marginal degradation in performance. Convergence times, though long compared with the process time constants, were notably shorter here than was the case for the experiments with static plant (ROM) reported in Chapter 6. This is probably a consequence of the particular contours of the process performance characteristic (Section 7.2), which would be expected to encourage rapid convergence to the general vicinity of the optimum. By comparison, the automaton computation cycle time (i.e., the search and reinforcement phases) is quite insignificant. This confirms the view that a large-scale multivariable controller could be constructed along the lines of the system described here, with no speed penalty from the automaton in operation.

The process under consideration was a comparatively simple one, but it did exhibit the general properties of a real large-scale system. Furthermore, by employing an automaton to tune a standard two-term process controller, an application has been devised which can be readily assimilated by the existing and established technology of process control.

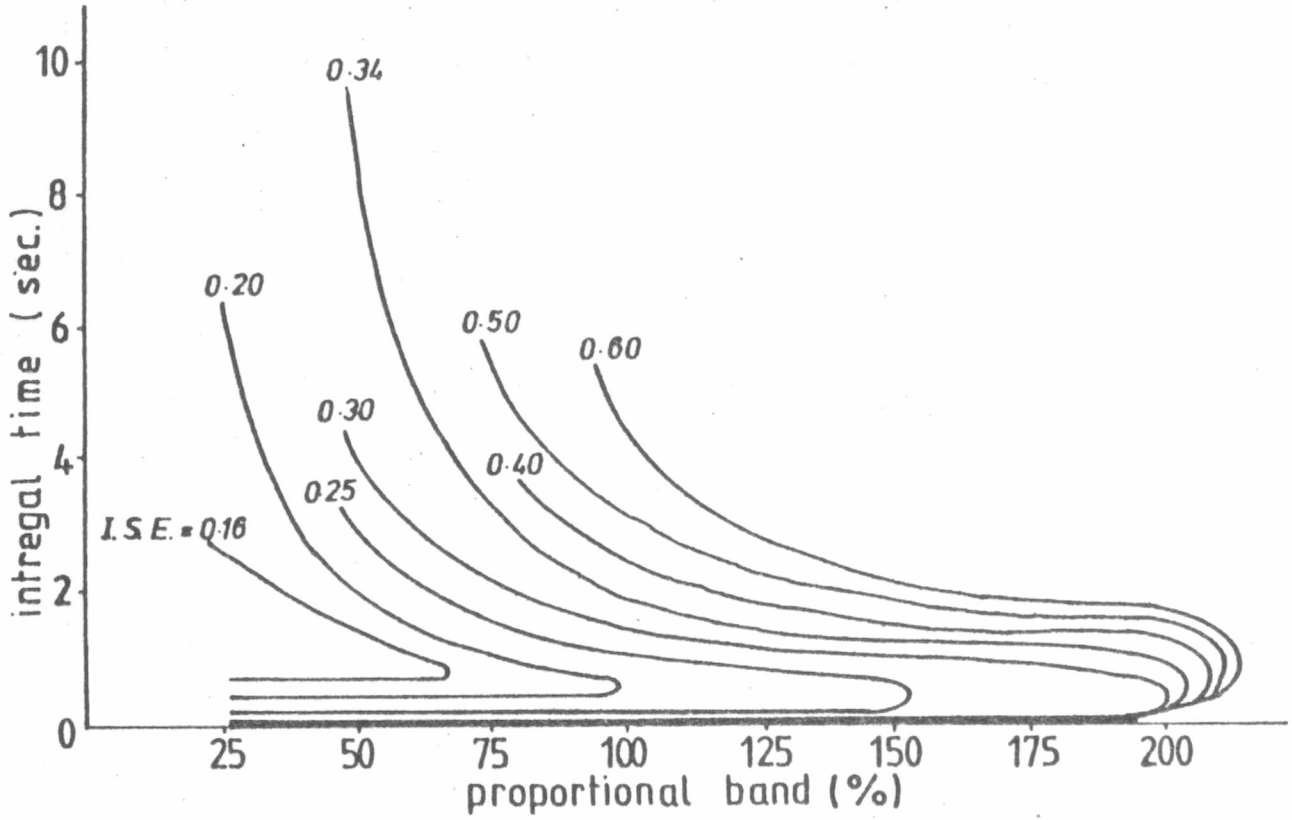


Figure 7·1 I.S.E. contours

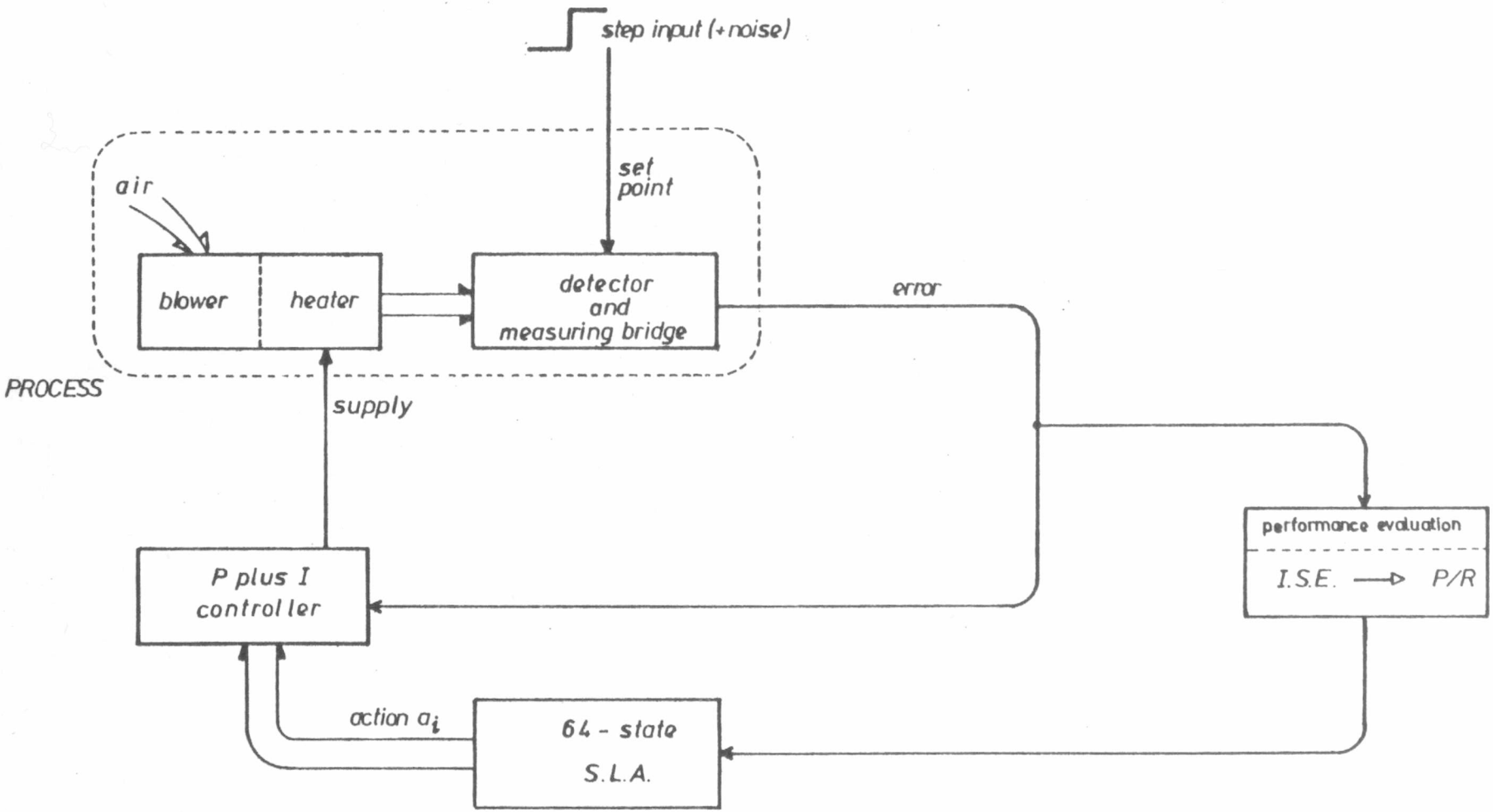


Figure 7.2 Block diagram of process control system

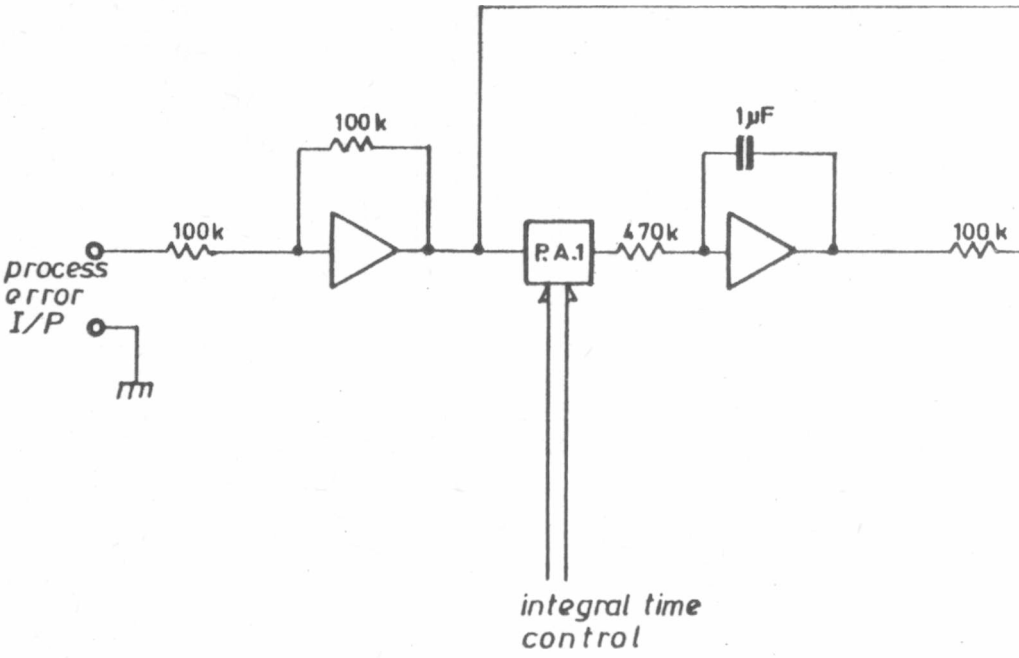
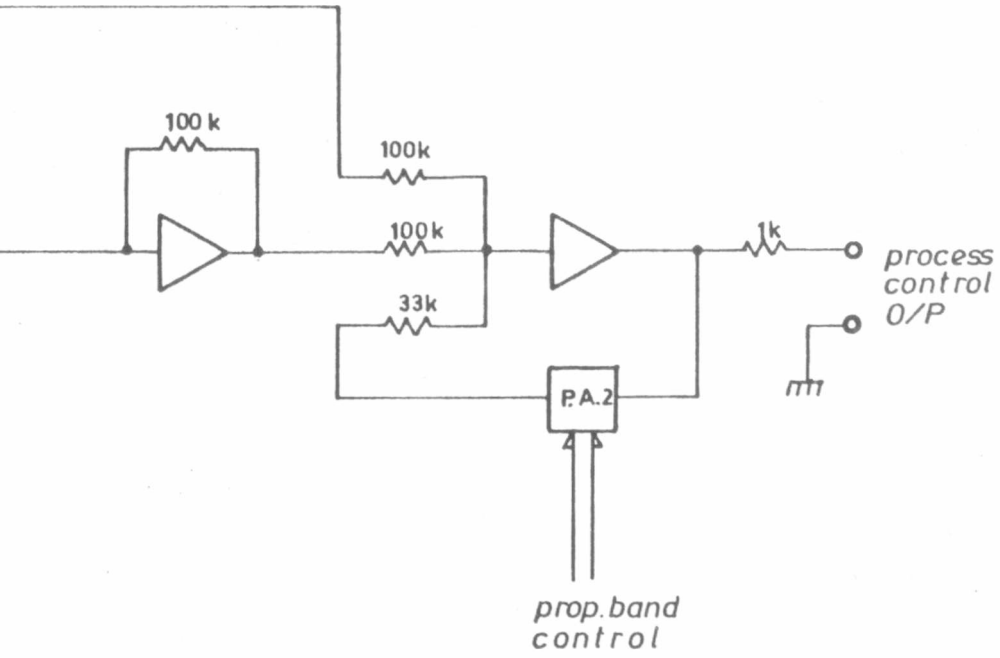

100k
100k
1μF
100k
P.A.1
470k
process
error
I/P
integral time
control

Figure 7.3



Controller circuit

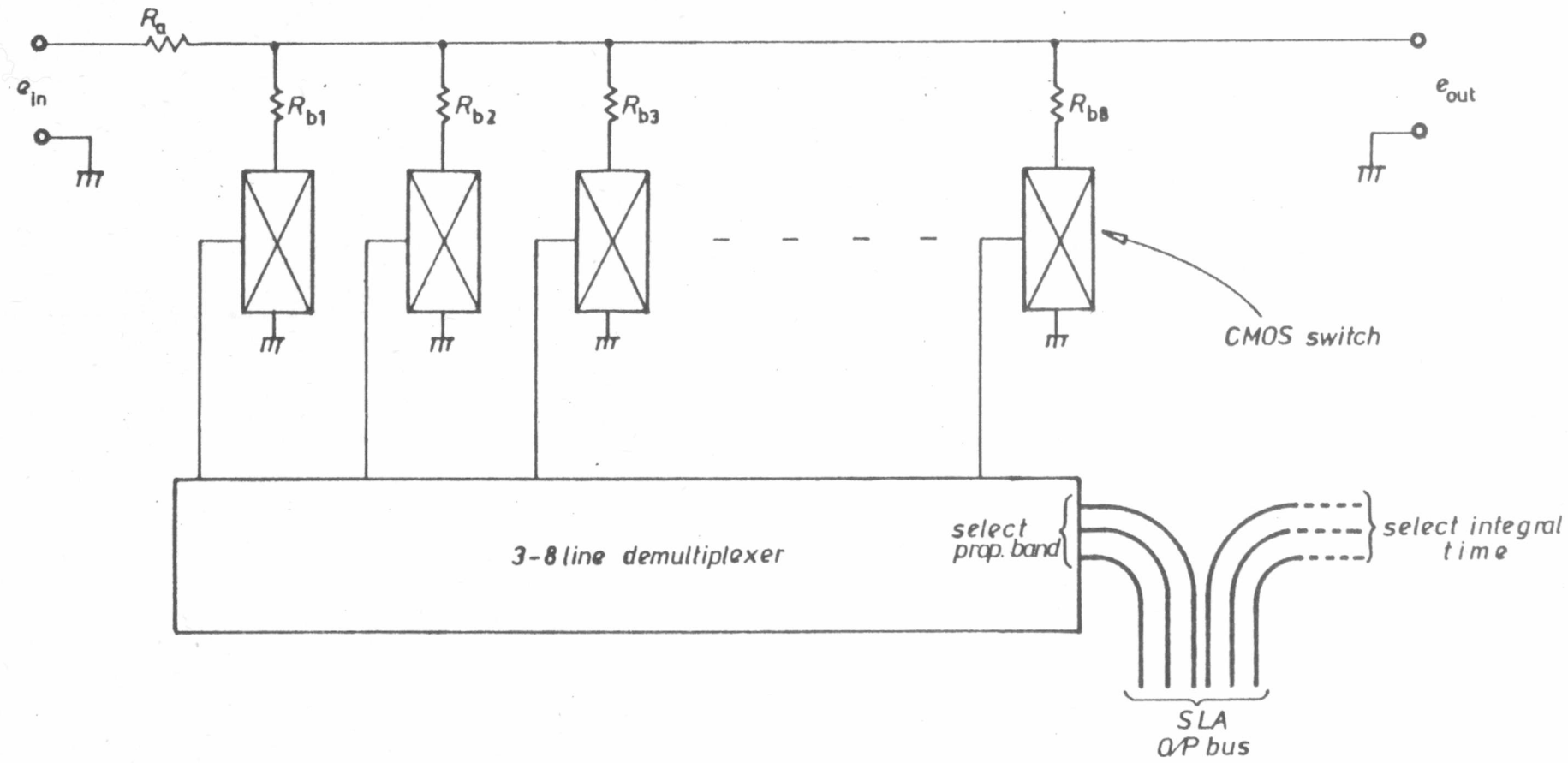


Figure 7.4 Programmable attenuator

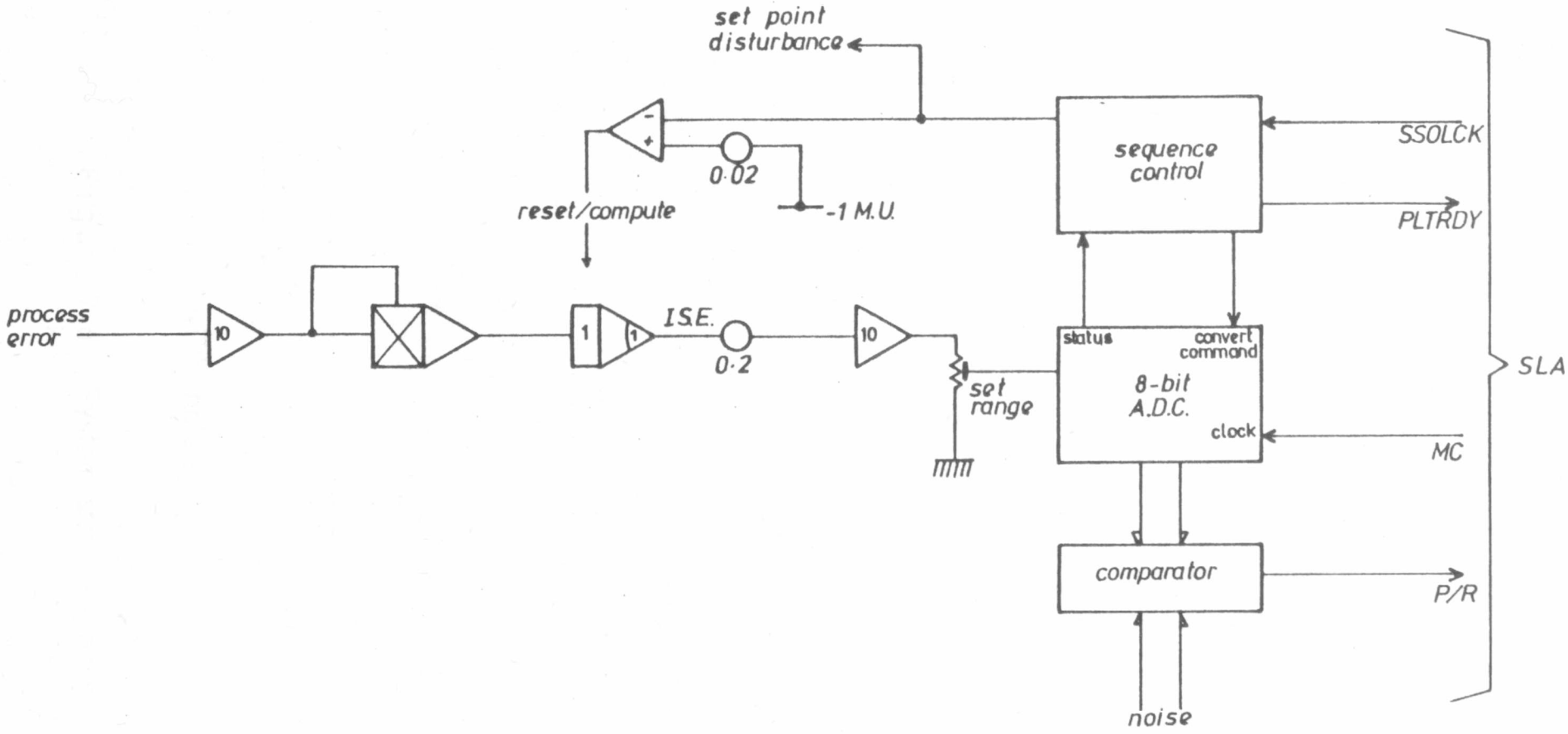
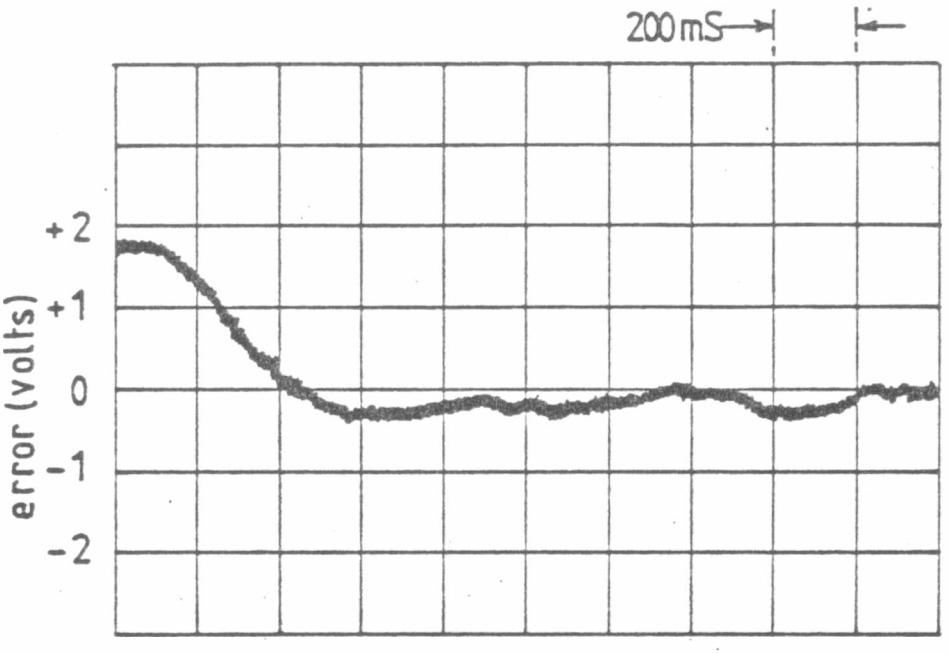
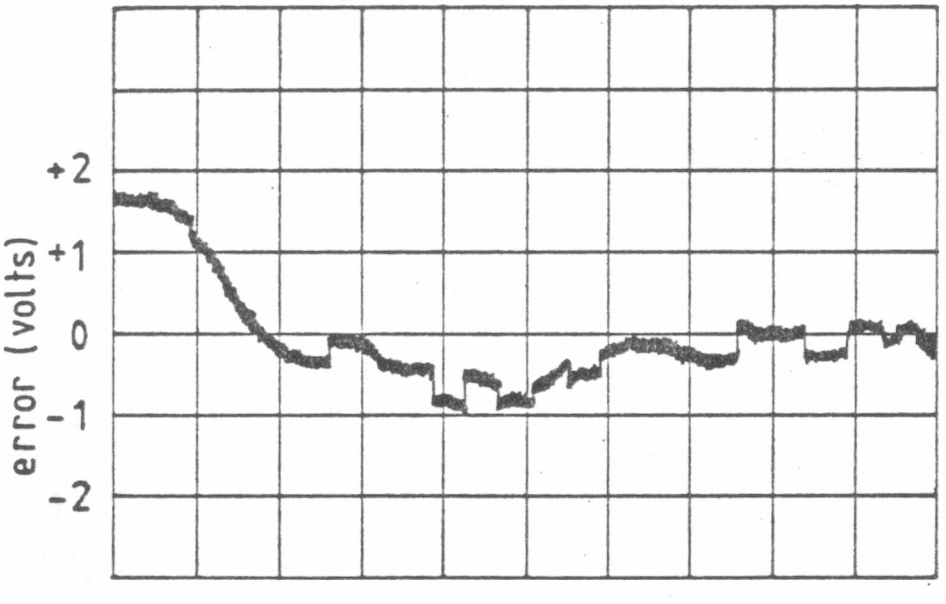


Figure 7·5 Performance evaluation section

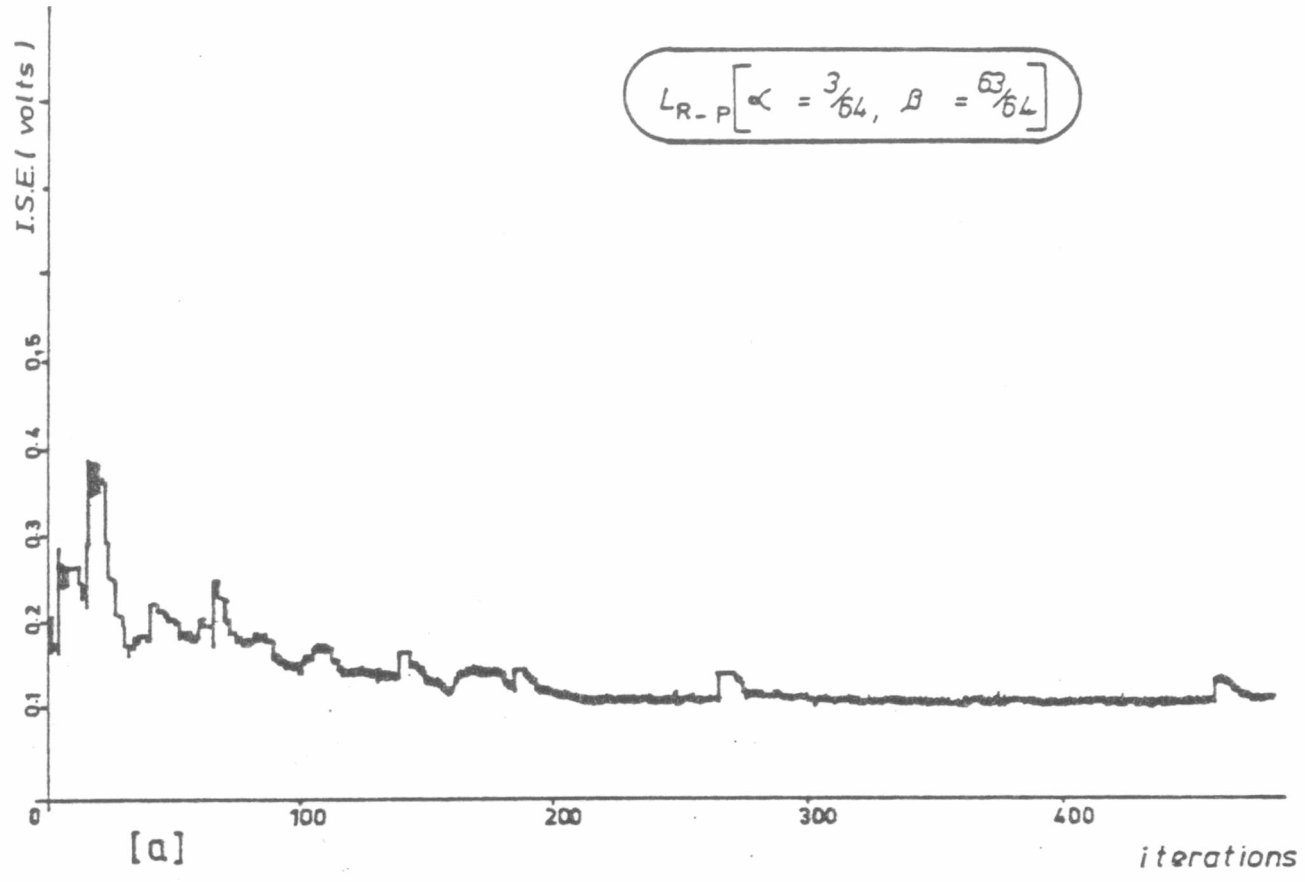


(a) without noise



(b) with noise

Figure 7.6 System step response



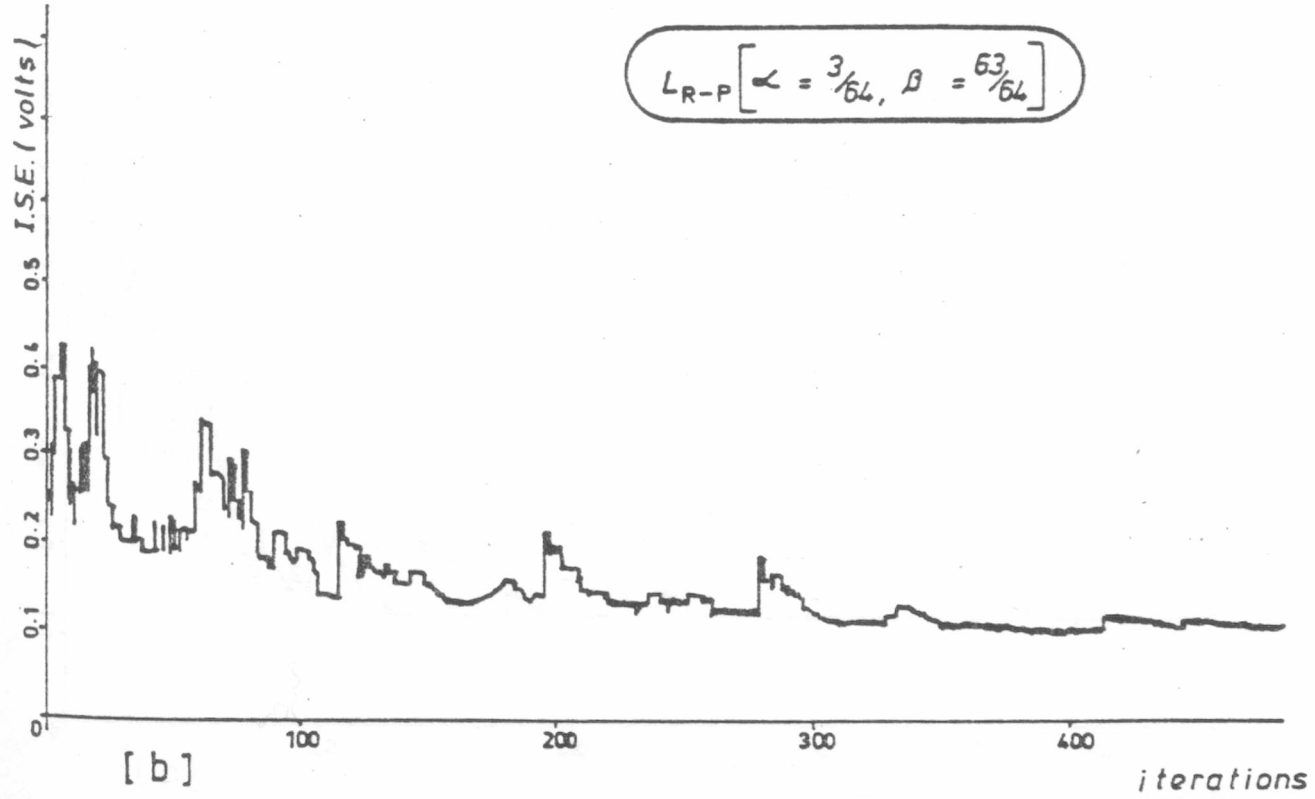
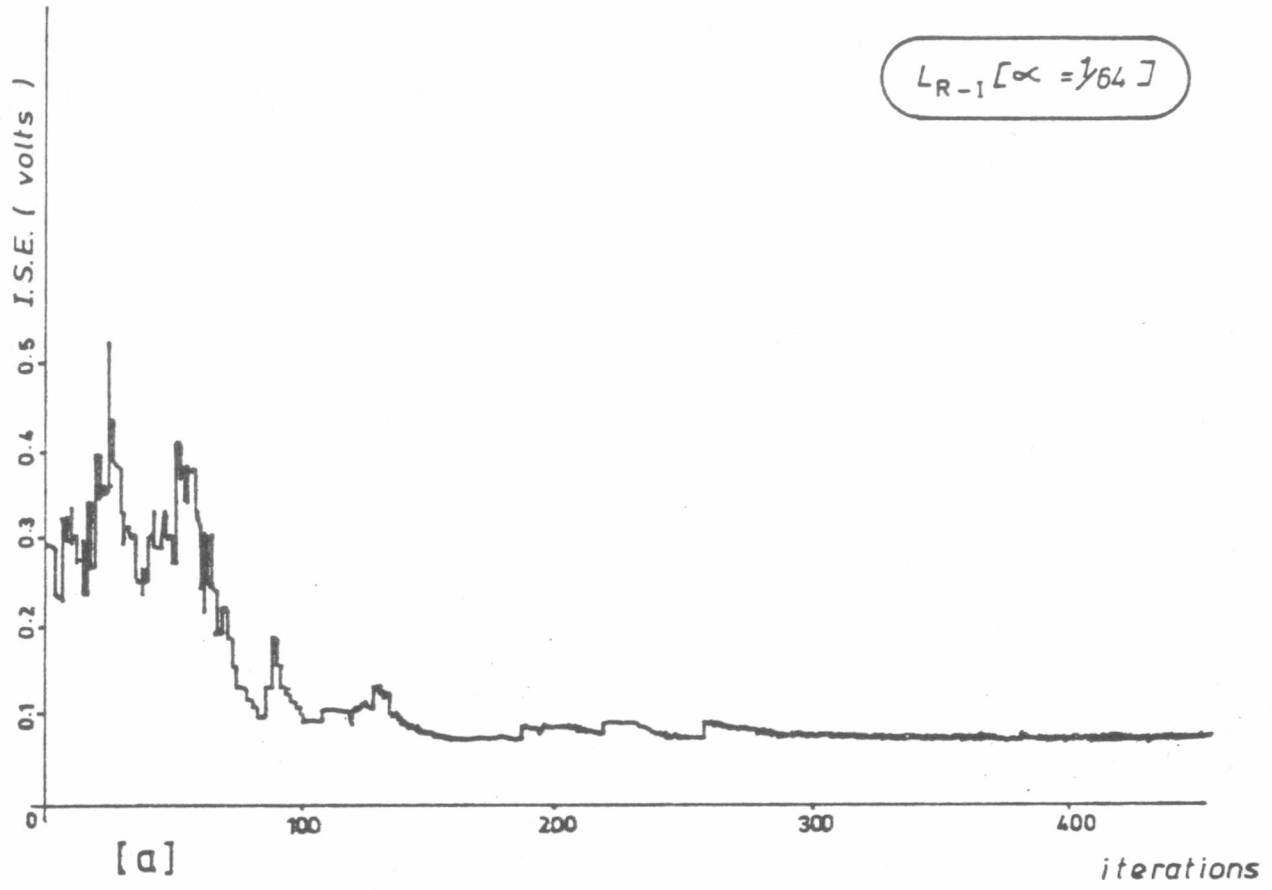


Figure 7.7 System error curves (1)



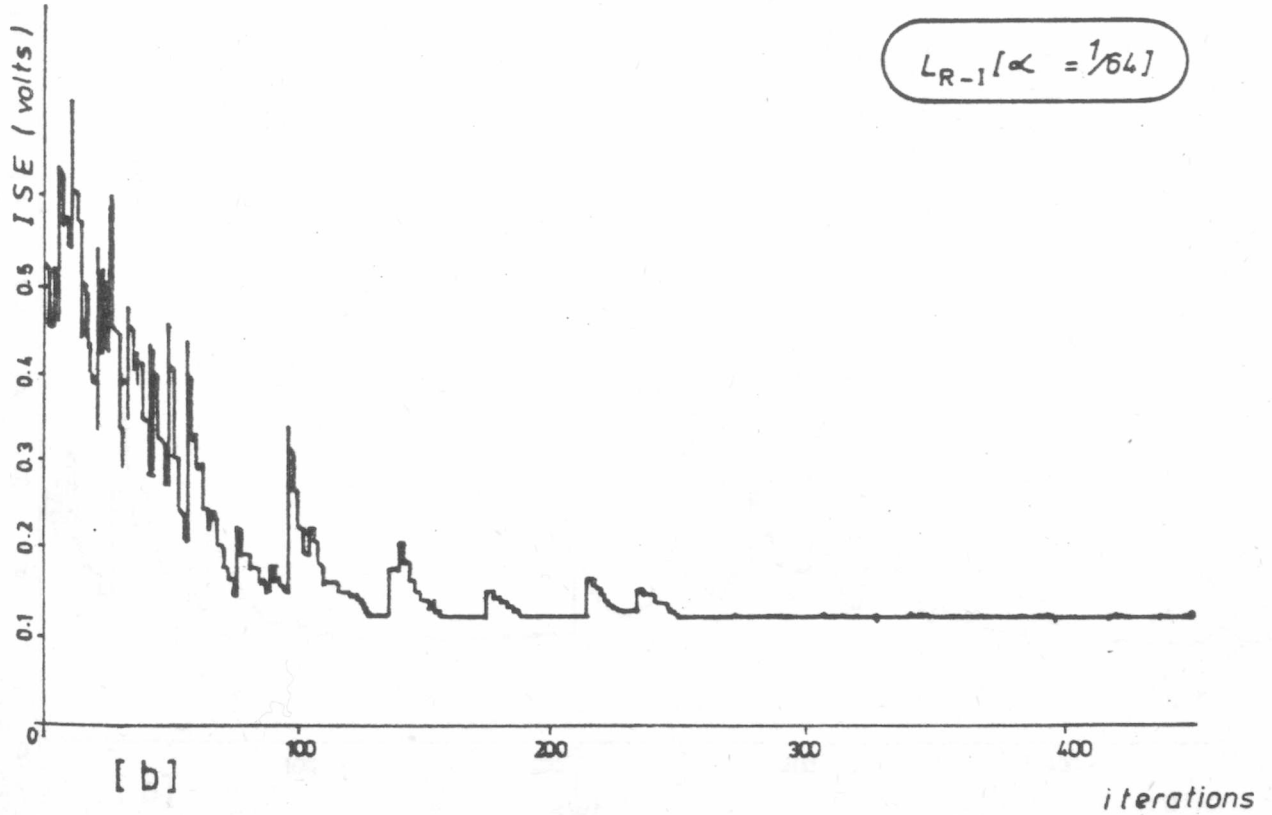
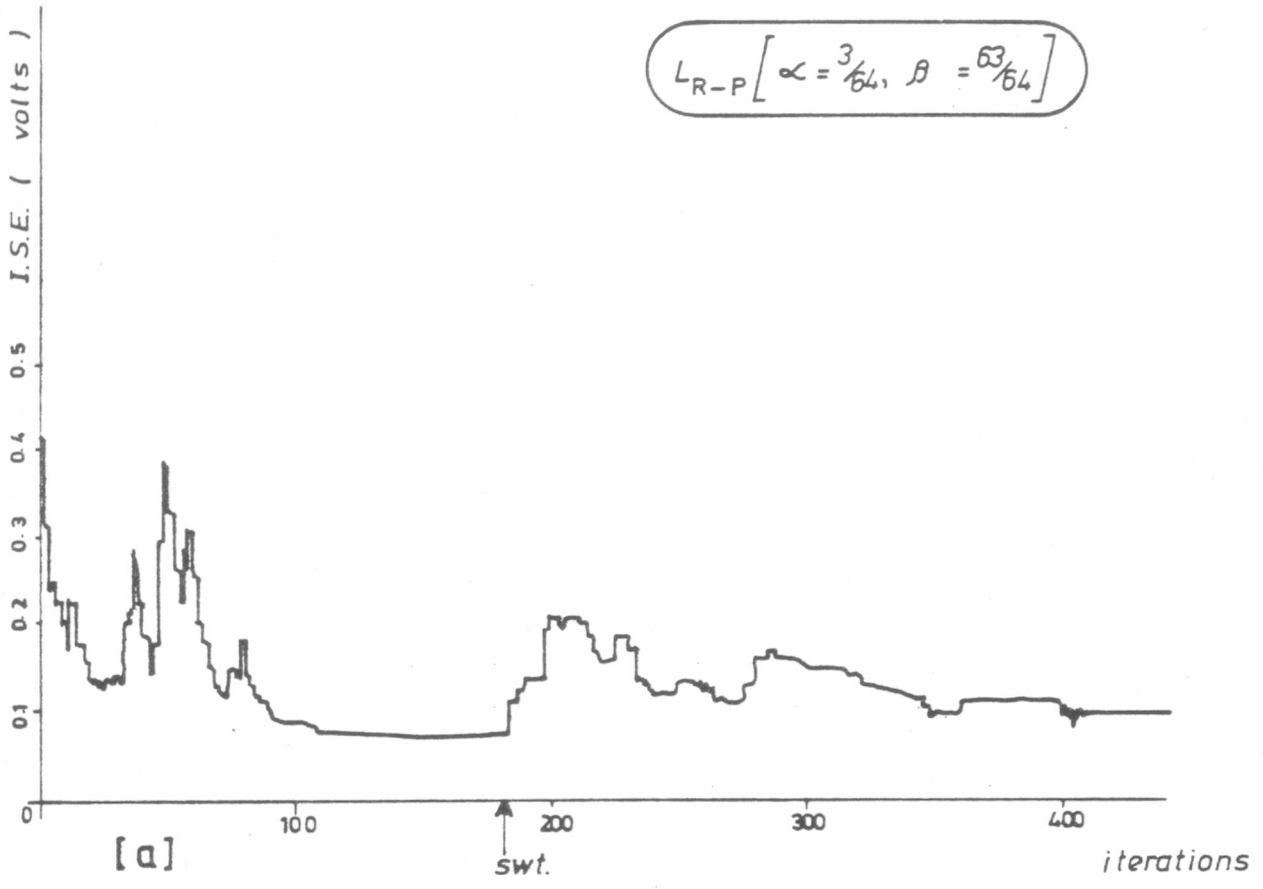


Figure 7.8 System error curves (2)



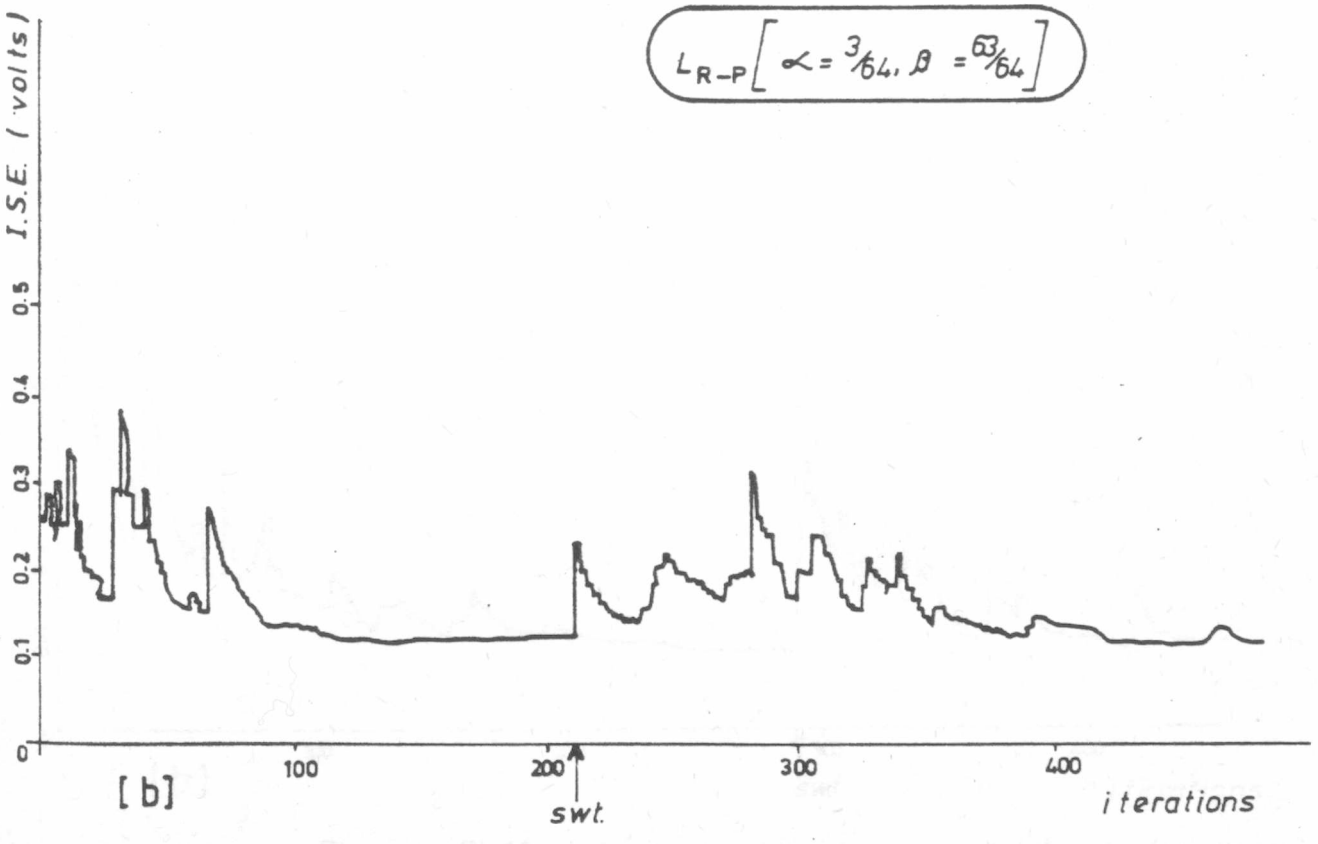
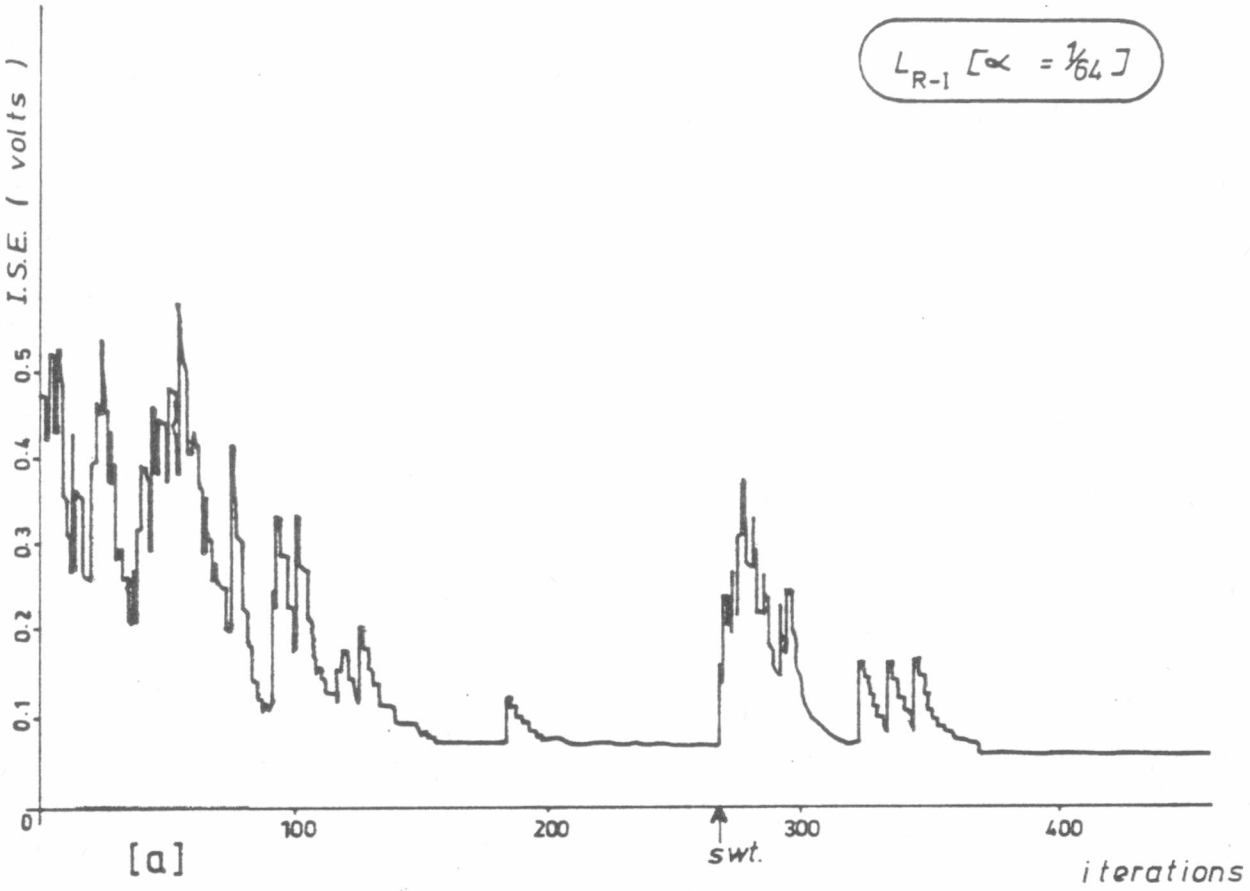


Figure 7.9 System error curves (3)



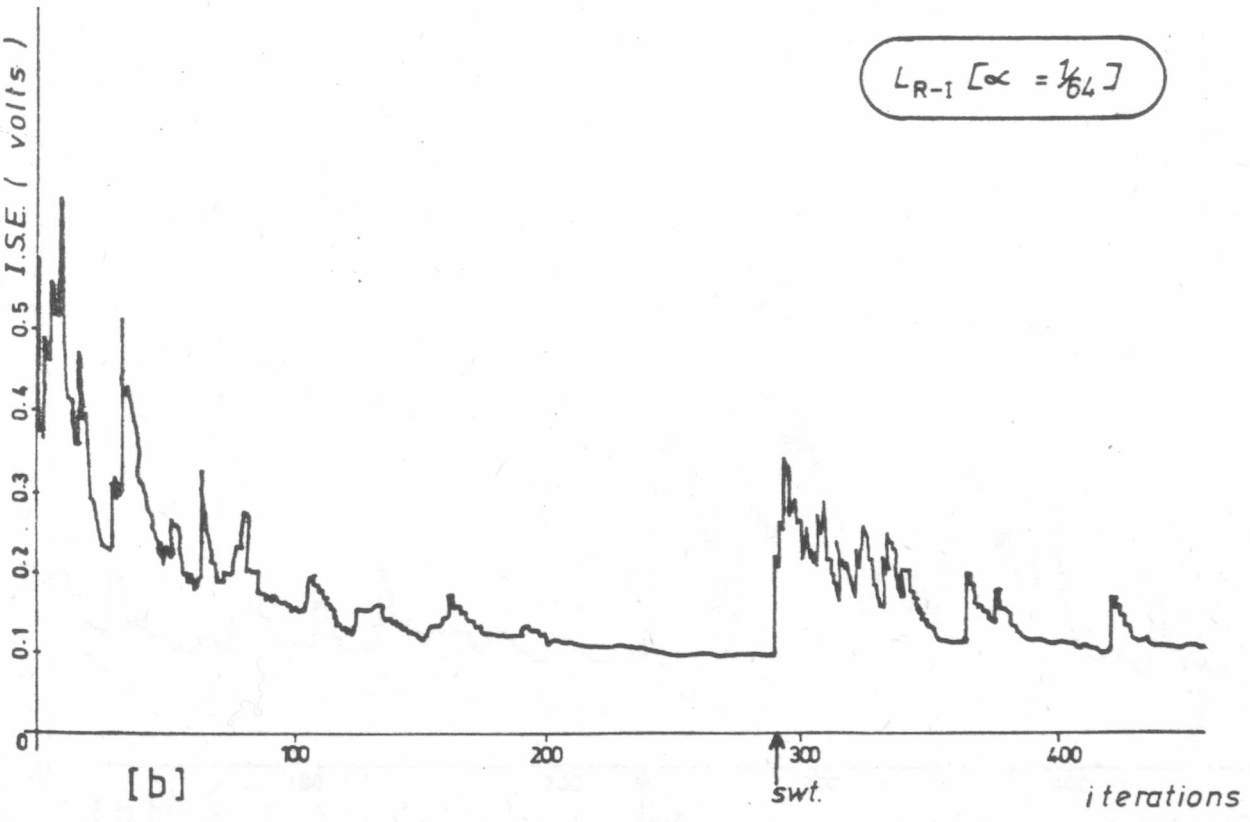
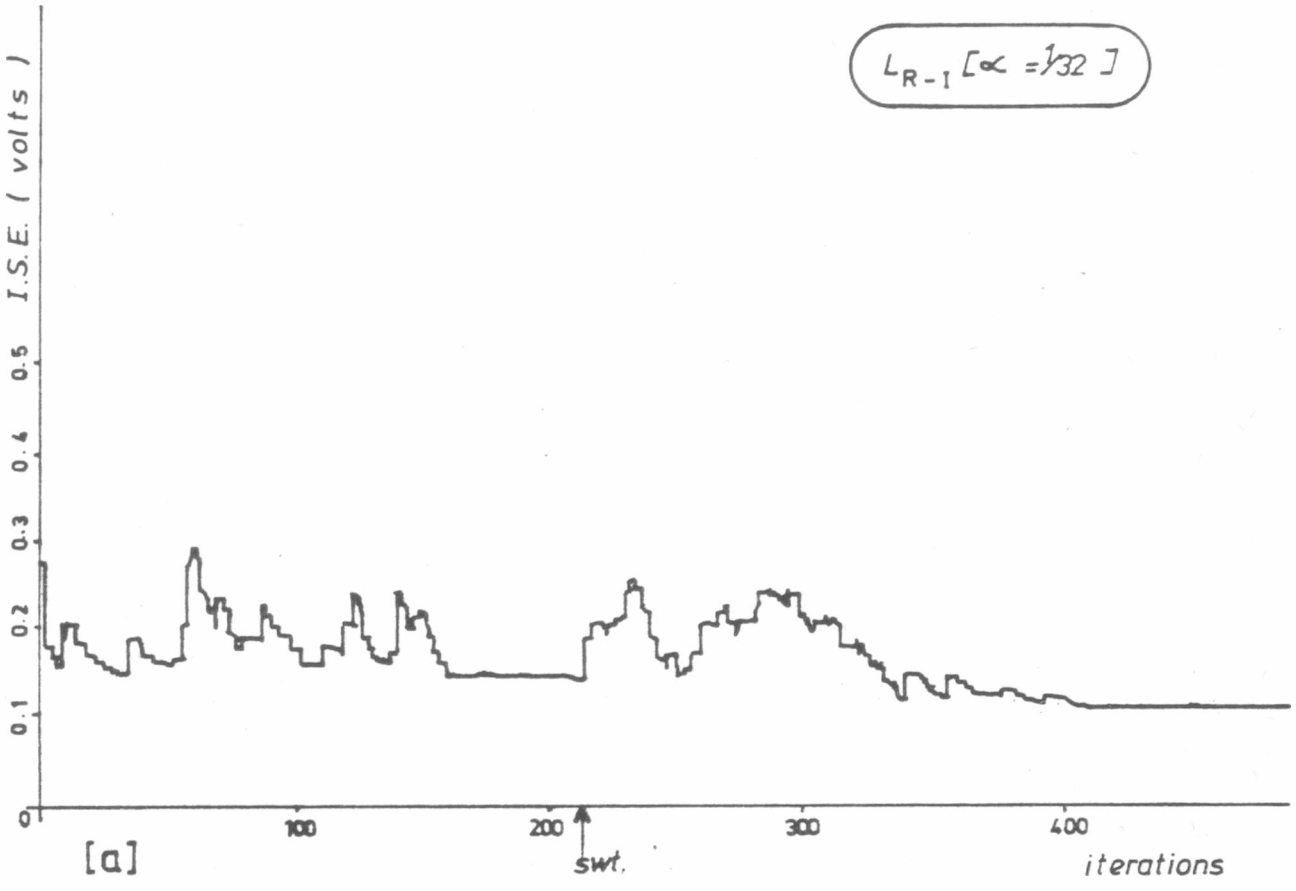


Figure 7.10 System error curves (4)



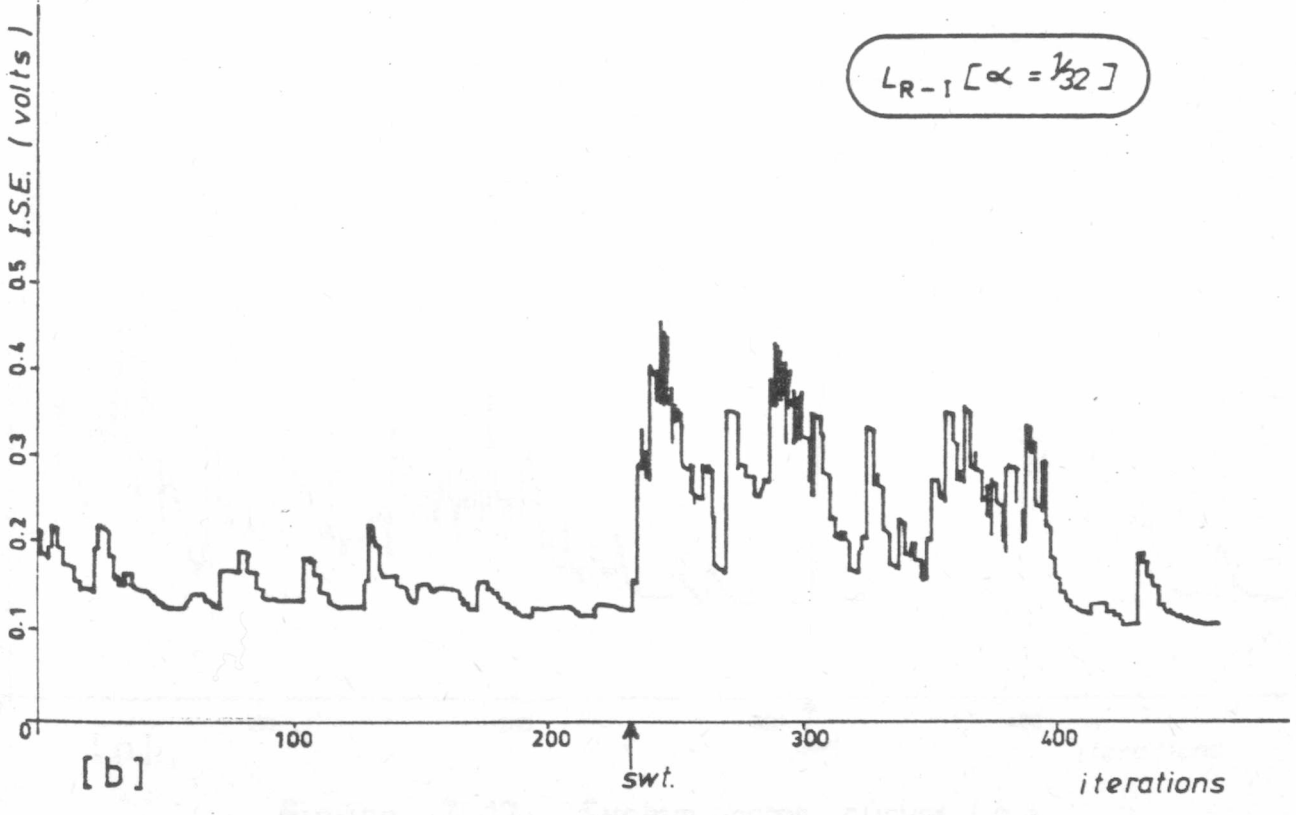
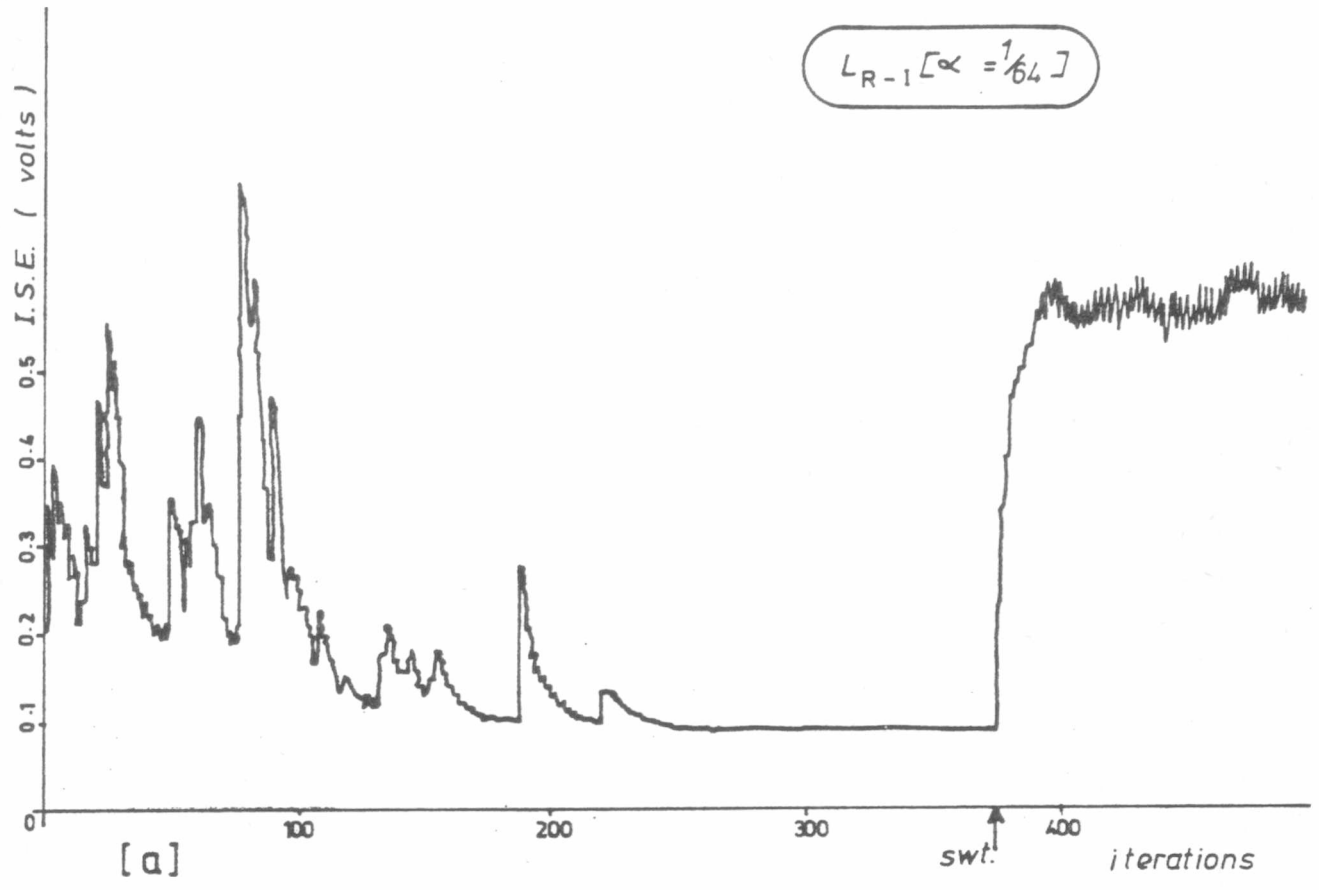


Figure 7.11 System error curves (5)



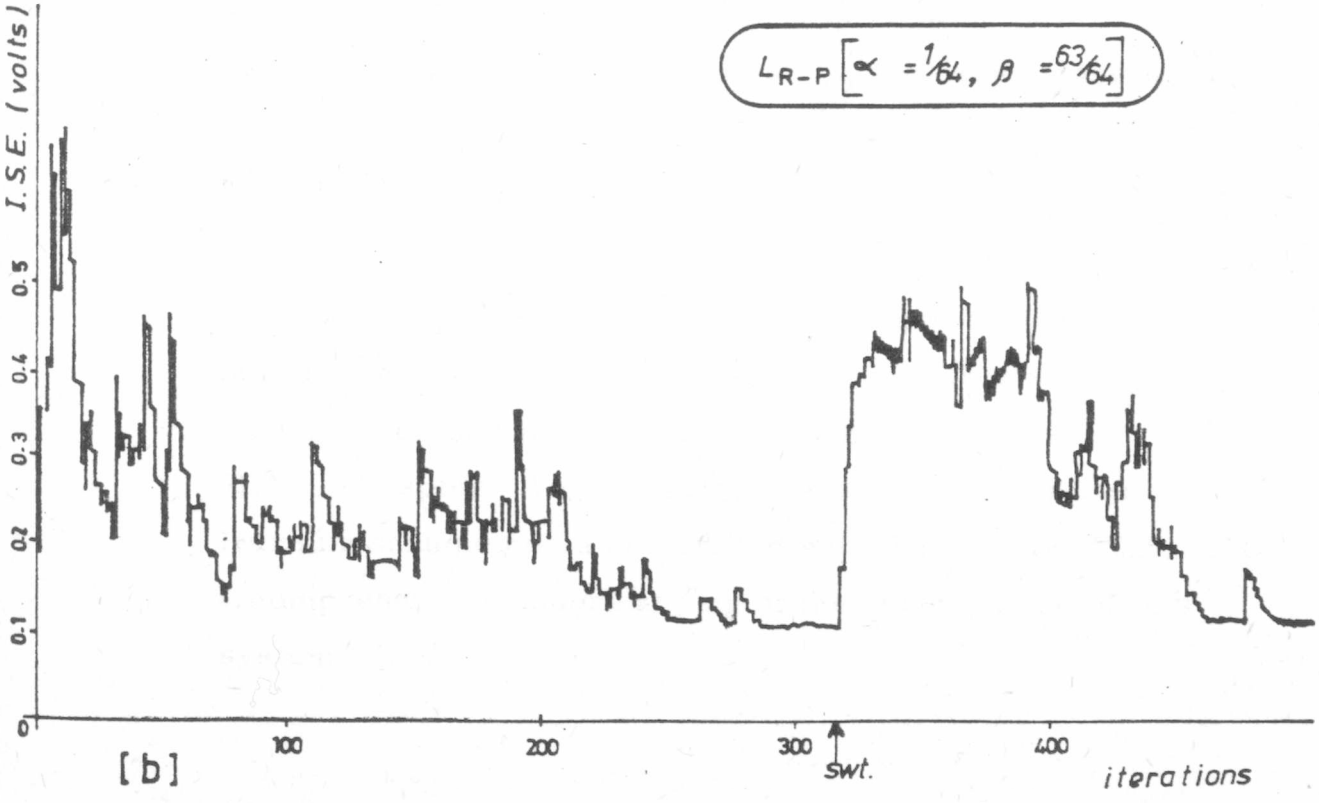


Figure 7·12 System error curves (6)

CHAPTER 8

CONCLUSIONS - REVIEW AND OUTLOOK

8.1 Review of the Project

This project was initiated against a background of considerable developments in learning automata theory[5, 7, 8]. Their behaviour was well understood, and application areas were being actively pursued[9-13, 16, 47-51]. In view of this, investigations began into the feasibility of developing hardware automata for practical engineering applications, with digital stochastic computing[1-4] providing the ideal medium. The vast majority of theoretical studies had concentrated on two-state automata; therefore they formed the basis for the initial experiments in hardware synthesis.

The first two-state design (flip-flop SLA - Chapter 2) was rather limited in scope, but realised quite a promising performance. The general learning characteristics were broadly similar to those reported previously[20, 21], both in transient and steady-state behaviour, i.e., learning time (in terms of system iterations), degree of expediency and variance. The results achieved here could not, by the nature of the hardware involved, approach the precision of these previous simulation studies, but the same general conclusions could be drawn. A simple plant simulator was developed also, and the degree of insensitivity of the system to superimposed noise was demonstrated.

The circuit design for the reinforcement algorithm turned out to be as flexible as it was simple, and was subsequently incorporated in a more advanced form of SLA design (ADDIE SLA - Chapter 3). Again, the salient features of each learning scheme could be verified with the ADDIE SLA, and a comprehensive understanding of the performance of this system /

system was built up. The experiments (Chapter 4), which dealt with a wide variety of reinforcement schemes, linear and non-linear, progressed a stage further with the consideration of non-stationary environments. In addition, the effect of superimposed noise in the environment was further investigated.

The ADDIE SLA had been devised using the concept of memory in the learning process, and acted as a prelude to the development of a large state order system in which a hierarchical structure based on a time-shared two-state automaton "cell" was employed (Chapter 5). Conservation of hardware and ease of operation were paramount in the design, but operating speed was not to be unduly compromised. The hierarchical SLA was initially tested with a simulated multimodal performance characteristic stored in a PROM (Chapter 6). The results obtained in these experiments, which could be considered much closer to real-life systems, indicated that the transition from simple two-state applications to the multivariable, multimodal problem had been successfully accomplished.

The applications studies culminated in the operation of a real small-scale thermal process controller under automaton supervision (Chapter 7). Successful control, reflected in the minimisation of error in the step response, was achieved in conditions which involved quite severe external disturbances to the system, demonstrating conclusively the viability of the automaton approach.

8.2 The Future

Although the hardware system described here has proved its worth, there is undoubtedly a case to be made for further development. A hierarchical array of two-state cells, constructed /

constructed with dedicated hardware using digital stochastic computing techniques, has been shown to be an eminently suitable solution to the high speed control of large-scale systems. However, the strict binary nature of the state order does represent a handicap. Some method of internally ordering the decision paths in a large hierarchical structure system would be useful in accelerating the convergence process by eliminating more rapidly the obviously less suitable directions for the "search" phase (see Section 5.8). Such an approach, wherein the structure of the automaton itself has adaptive properties, is in sympathy with the type of multilevel automaton arrangement envisaged by Narendra[5].

A possible solution for non-binary r-state systems was touched upon in section 5.3, in which it was suggested that a binary hierarchical automaton would have its output decoded down to the requisite number of actions for a particular application. However, it is likely that the best approach to the design of general r-state automata lies in the use of microprocessors. Preliminary work in this area has in fact been reported recently[71]. Such systems are ideally suited to the telephone traffic routing problem[59], particularly in view of the development of stored program exchange control techniques. In this context, the dedicated hardware system would still have a part to play, providing a means for the high speed simulation of learning schemes prior to the implementation of automata in communication networks via the controlling software.

For medium speed applications, the microprocessor is not at a severe disadvantage, since the algorithm calculations can be performed rapidly enough, using an external arithmetic processor if necessary. Delays arise, though, in using software random number generating routines. For fastest /

fastest operation, digital stochastic hardware has the inherent advantage that calculations are effectively performed at each single clock pulse. If the hardware stochastic computing approach is to remain viable in the long term, however, it would be preferable to have the requisite functions integrated in the form of a universal stochastic module[32]. This element would provide all the standard stochastic functions (Chapter 1) on a single programmable LSI chip, with all the attendant advantages of space and cost-effectiveness.

In the process control application area, speed is not usually a priority, given the long time constants which are often encountered. It may well become an important factor, however, when large multivariable systems are involved. Considerable development effort will be required in the detail design of performance evaluation systems for the plant-automaton interface, since communication at this point is vital to the overall effectiveness of the controller.

In conclusion, the union of digital stochastic computing techniques with learning automata theory has produced a form of learning machine which has real potential for on-line adaptive control systems.

REFERENCES

1 Gaines B R
'Stochastic Computing'
AFIPS 30 SJCC 1967 pp 149 - 156

2 Gaines B R
'Stochastic Computing Systems'
in 'Advances in Information and Systems Science'
ed J Tou 2 1969 p 37 - 172

3 Poppelbaum W J, Afuso C and Esch J W
'Stochastic Computing Elements and Systems'
AFIPS 31 FJCC 1967 pp 631 - 644

4 Ribiero S T
'Random Pulse Machines'
IEEE Trans on Electronic Computers
16 3 June 1967 pp 261 - 276

5 Narendra K S and Thathachar M A L
'Learning Automata - A Survey'
IEEE Trans on Systems, Man and Cybernetics
4 4 July 1974 pp 323 - 334

6 Bibbero R J
'Microprocessors in Instruments and Control'
Wiley 1977

7 Tsetlin M L
'On the Behaviour of Finite Automata in Random Media'
Avtomatika i Telemekhanika (Automation and Remote Control)
22 10 October 1961 pp 1345 - 1354

8 Varshavskii V I and Vorontsova I P
'On the Behaviour of Stochastic Automata with a Variable Structure'
ibid 24 March 1963 pp 353 - 360

9 Sklansky J
'Learning Systems for Automatic Control'
IEEE Trans on Automatic Control
11 January 1966 pp 6 - 19

10 Fu K S
'Learning Control Systems - Review and Outlook'
ibid 15 April 1970 pp 210 - 221

11 /

11 Riordon J S
'An Adaptive Automaton Controller for Discrete-Time Markov Processes'
Automatica 5 Pergamon Press 1969 pp 721 - 730

12 Riordon J S
'Optimal Feedback Characteristics from Stochastic Automaton Models'
IEEE Trans on Automatic Control
14 February 1969 pp 89 - 92

13 Jones L E and Fu K S
'On the Selection of a Sub-Goal and the Use of a Priori Information in Learning Control Systems'
Automatica 5 Pergamon Press 1969 pp 705 - 720

14 Tsypkin Y Z
'Adaptation and Learning in Automatic Systems'
Academic Press 1971

15 Jarvis R A
'Optimisation Strategies in Adaptive Control: A Selective Survey'
IEEE Trans on Systems, Man and Cybernetics
7 3 March 1977 pp 125 - 143

16 El-Fattah Y M and Najim K
'Practical Problems Related to the Use of Learning Models for Control of Industrial Processes'
Proc of the Workshop on Applications of Adaptive Control
Yale University Connecticut U S A
August 1979 pp 164 - 169

17 Brown A W
'Design of a Digital Stochastic Computer'
M Phil Thesis
Robert Gordon's Institute of Technology, Aberdeen
1975

18 Mars P, McIntosh F G and Baxter T
'High-Speed Simulation of Discrete Dynamic Probabilistic Systems'
Mathematics and Computers in Simulation
21 1979 pp 21 - 38

19 Viswanathan R and Narendra K S
'Expedient and Optimal Variable-Structure Automata'
Becton Centre Yale University Connecticut USA
Tech Rep CT-31 April 1970

20 /

20 Viswanathan R and Narendra K S
'Simulation Studies of Stochastic Automata Models'
ibid Tech Rep CT-45 December 1971

21 Viswanathan R and Narendra K S
'Comparison of Expedient and Optimal Reinforcement Schemes for Learning Systems'
J of Cybernetics
2 1 1972 pp 21 - 37

22 Viswanathan R and Narendra K S
'A Note on the Linear Reinforcement Scheme for Variable-Structure Stochastic Automata'
IEEE Trans on Systems, Man and Cybernetics
2 April 1972 pp 292 - 294

23 Glorioso R M and Grueneich G R
'A Training Algorithm for Systems Described by Stochastic Transition Matrices'
ibid 1 January 1971 pp 86 - 87

24 Mason L G
'An Optimal Learning Algorithm for S-Model Environments'
IEEE Trans on Automatic Control
18 October 1973 pp 493 - 496

25 Lakshmivarahan S and Thathachar M A L
'Absolutely Expedient Learning Algorithms for Stochastic Automata'
IEEE Trans on Systems, Man and Cybernetics
3 May 1973 pp 281 - 286

26 Sawaragi Y and Baba N
'A Note on the Learning Behaviour of Variable-Structure Stochastic Automata'
ibid 3 November 1973 pp 644 - 647

27 Sawaragi Y and Baba N
'Two ϵ-Optimal Non-Linear Reinforcement Schemes for Stochastic Automata'
ibid 4 January 1974 pp 126 - 131

28 Baba N
'On the Learning Behaviour of the SL_{R-I} Reinforcement Scheme for Stochastic Automata'
ibid 6 August 1976 pp 580 - 582

29 /

29 Lakshmivarahan S and Thathachar M A L
'Absolute Expediency of Q - and S-Model Learning Algorithms'
ibid 6 March 1976 pp 222 - 226

30 Lakshmivarahan S and Thathachar M A L
'Bounds on the Convergence Probabilities of Learning Automata'
ibid 6 November 1976 pp 756 - 763

31 Miller A J
'Digital Stochastic Computation'
Ph D Thesis
University of Aberdeen
1976

32 Baxter T
'Some Aspects of the Design, Construction and Applications of a Digital Stochastic Computer'
M Phil Thesis
Robert Gordon's Institute of Technology Aberdeen
1975

33 Miller A J, Brown A W and Mars P
'Adaptive Logic Circuits for Digital Stochastic Computers'
Electronics Letters
9 21 1973 pp 500 - 502

34 Miller A J, Brown A W and Mars P
'A Study of an Output Interface for a Digital Stochastic Computer'
Int J of Electronics
37 5 pp 637 - 655

35 Miller A J, Brown A W and Mars P
'Moving-Average Output Interface for Digital Stochastic Computers'
Electronics Letters
10 20 pp 419 - 420

36 Miller A J and Mars P
'Optimal Estimation of Digital Stochastic Sequences'
Int J of Systems Science
8 6 1977 pp 683 - 696

37 Miller A J and Mars P
'Theory and Design of a Digital Stochastic Computer Random Number Generator'
Mathematics and Computers in Simulation
19 1977 pp 198 - 216

38 /

38 Birolini A
'Hardware Simulation of Semi-Markov and Related Processes'
ibid 19 1977 pp 75 - 97

39 Albareda A and Castanie F
'Optimisation of the Structure of a Random Number Generator with Correlated Bits'
Proc of the 1st Int Symp
on Stochastic Computing and its Applications
INPT Toulouse France 1978 pp 89 - 102

40 Schwind M
'On Generating and Applicating a Set of Independent Bernoulli-Sequences'
ibid pp 103 - 112

41 McLean H R and Mars P
'High-Speed Matrix Inversion by Stochastic Computer'
Electronics Letters
12 18 September 1976 pp 457 - 459

42 McLean H R and Mars P
'Implementation of Linear Programming with a Digital Stochastic Computer'
ibid 12 20 September 1976 pp 516 - 517

43 Mars P and Grover D
'This Random Pulse will Speed up the Processing'
Computing Europe
October 7 1976 pp 14 - 15

44 Witten I H and Madams P H C
'A Low-Cost Long-Term Stochastic Integrator'
Electronics Letters
14 10 May 1978 pp 293 - 294

45 Damashek M
'Shift Register with Feedback Generates White Noise'
Electronics
May 1976 pp 107 - 109

46 Golomb S
'Shift Register Sequences'
Holden-Day 1967

47 McMurtry, G J and Fu K S
'A Variable Structure Automaton used as a Multimodal Searching Technique'
IEEE Trans on Automatic Control
11 July 1966 pp 379 - 387

48 /

48 Shapiro I J and Narendra K S
'Use of Stochastic Automata for Parameter Self-Optimisation with Multimodal Performance Criteria'
IEEE Trans on Systems Science and Cybernetics
5 4 October 1969 pp 352 - 360

49 Viswanathan R and Narendra K S
'Application of Stochastic Automata Models to Learning Systems with Multimodal Performance Criteria'
Becton Centre Yale University Connecticut U S A
Tech Rep CT-40 June 1971

50 Jarvis R A
'Adaptive Global Search by the Process of Competitive Evolution'
IEEE Trans on Systems, Man and Cybernetics
5 3 May 1975 pp 297 - 311

51 Baba N
'Theoretical Considerations of the Parameter Self-Optimisation by Stochastic Automata'
Int J of Control
27 2 1978 pp 271 - 276

52 Levy N M
'The Application of Hill-Climbing Methods to the Adaptive Control of Small-Scale Practical Systems'
IEEE Trans on Industrial Electronics and Control Instrumentation
24 1 February 1977 pp 74 - 80

53 Kubrusly C S and Curtain R F
'Identification of Noisy Distributed Parameter Systems Using Stochastic Approximation'
Int J of Control
25 3 1977 pp 441 - 455

54 Jarvis R A
'Adaptive Global Search in a Time-Variant Environment Using a Probabilistic Automaton'
Proc IREE (Australia) July 1969 pp 210 - 226

55 Jarvis R A
'Adaptive Global Search in a Time-Variant Environment Using a Probabilistic Automaton with Pattern Recognition Supervision'
IEEE Trans on Systems Science and Cybernetics
6 3 July 1970 pp 209 - 217

56 /

56 Neville R G, Nicol C R and Mars P
'Design of Stochastic Learning Automata Using Adaptive Digital Logic Elements'
Electronics Letters
14 11 May 1978 pp 324 - 326

57 Witten I H
'Finite-Time Performance of Some Two-Armed Bandit Controllers'
IEEE Trans on Systems, Man and Cybernetics
3 March 1973 pp 194 - 197

58 Witten I H
'Stochastic Implementation of Learning Controllers'
IEE Colloquium on Parallel Digital Computing Methods
1976 Digest No 1976/30 paper 6

59 Narendar K S and Thathachar M A L
'On the Behaviour of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing'
Proc of the 1st Int Symp on Stochastic Computing and its Applications
INPT Toulouse France 1978 pp 301 - 317

60 Tsuji H et al
'An Automaton in the Non-Stationary Random Environment'
Information Sciences
6 1973 pp 123 - 142

61 Baba N and Sawaragi Y
'On the Learning Behaviour of Stochastic Automata Under a Non-Stationary Random Environment'
IEEE Trans on Systems, Man and Cybernetics
5 March 1975 pp 273 - 275

62 Loui M C and Narendra K S
'Comparison of Learning Automata Operating in Non-Stationary Environments'
Becton Centre Yale University Connecticut U S A
Tech Rep CT-65 May 1965

63 Coutts M J and Mars P
'Theory and Applications of a Modified-Estimating Automaton'
Proc of the 1st Int Symp
on Stochastic Computing and its Applications
INPT Toulouse France 1978 pp 303 - 320

64 Mackie N J and Mars P
'Stochastic Automata in Non-Stationary Environments'
ibid pp 321 - 344

65 /

65 Narendra K S and Viswanathan R
'A Two-Level System of Stochastic Automata for Periodic Random Environments'
IEEE Trans on Systems, Man and Cybernetics
2 April 1972 pp 285 - 289

66 Viswanathan R and Narendra K S
'Games of Stochastic Automata'
ibid 4 January 1974 pp 131 - 135

67 Langholz G and Katz E
'Learning Automata in a Three-Move Zero-Sum Game'
ibid 9 5 May 1979 pp 304 - 309

68 Asai K and Kitajima S
'A Method for Optimising Control of Multimodal Systems Using Fuzzy Automata'
Information Sciences
3 1971 pp 343 - 353

69 Asai K and Kitajima S
'Optimising Control Using Fuzzy Automata'
Automatica
8 Pergamon Press 1972 pp 101 - 104

70 Luders G and Narendra K S
'An Adaptive Observer and Identifier for a Linear System'
IEEE Trans on Automatic Control 18 October 1973
pp 496 - 499

71 Swan G B
'Investigation and Design of an n-State Learning Automaton'
B Sc Thesis Robert Gordon's Institute of Technology
Aberdeen 1980

BIBLIOGRAPHY

Astrom K J and Wittenmark B
'On Self-Tuning Regulators'
Automatica 9 Pergamon Press 1973 pp 185 - 199

Atkinson R C, Bower G H and Crothers E J
'An Introduction to Mathematical Learning Theory'
John Wiley and Sons 1965

Bharucha-Reid A T
'Elements of the Theory of Markov Processes and their Applications'
McGraw-Hill 1960

Booth T L
'Sequential Machines and Automata Theory'
John Wiley and Sons 1968

Charlesworth A S and Fletcher J R
'Systematic Analogue Computer Programming'
Pitman 1967

Eveleigh V W
'Adaptive Control and Optimisation Techniques'
McGraw-Hill 1967

Gaines B R
'Stochastic and Fuzzy Logics'
Electronics Letters 11 9 May 1975 pp 188 - 189

Gould E E
'An Automatic Control System with Self-Adjustment of Two Parameters'
MSEE Thesis University of Washington Seattle 1960

Halliwell J
'Stochastic Computers Solve More Problems Faster'
Electronic Engineering October 1970 pp 63 - 65

Hasdorff L
'Gradient Optimisation and Non-linear Control'
John Wiley and Sons 1976

Healey M
'Principles of Automatic Control'
3rd ed Hodder and Stoughton 1975

Hillier F S and Lieberman G J
'Introduction to Operations Research'
Holden-Day Inc 1967

Howard R A /

Howard R A
'Dynamic Programming and Markov Processes'
MIT Press 1960

Ichikawa K
'Principle of Luders-Narendra's Adaptive Observer'
Int J of Control 31 2 1980 pp 351 - 365

Jackson A S
'Analogue Computation'
McGraw-Hill 1960

Karlin S and Taylor H M
'A First Course in Stochastic Processes'
Academic Press 1975

Kickert W J M and Mamdani E H
'Analysis of a Fuzzy Logic Controller'
Fuzzy Sets and Systems 1 1978 pp 29 - 44

Korn G A
'Random Process Simulation and Measurements'
McGraw-Hill 1966

LaCarna R J and Johnson J R
'A Learning Controller for the Megawatt Load-Frequency Control Problem'
IEEE Trans on Systems, Man and Cybernetics 10 1 January 1980 pp 43 - 49

Mackie N J, Chrystall M S and Mars P
'Some Aspects of Stochastic Learning Automata in Non-Autonomous Environments'
Robert Gordon's Institute of Technology Aberdeen Tech Rep 1979

Mahmoud M S
'Multilevel Systems Control and Applications: A Survey'
ibid 7 3 March 1977 pp 125 - 143

Maisel L
'Probability, Statistics and Random Processes'
Simon and Schuster 1971

Mamdani E H
'Application of Fuzzy Algorithms for Control of Simple Dynamic Plant'
Proc IEE 121 12 December 1974 pp 1585 - 1588

Mamdani E H and Assilian S
'An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller'
Int J of Man-Machine Studies 7 1975 pp 1 - 13

Narendra K S /

Narendra K S, Mason L G and Tripathi SS
'Application of Learning Automata to Telephone Traffic Routing Problems'
Becton Centre Yale University Conn USA Tech Rep CT-60 January 1974

Narendra K S and Wright EA
'Application of Learning Automata to Telephone Traffic Routing Problems'
ibid Tech Rep CT-69 May 1976

Narendra K S and Lakshmivarahan S
'Learning Automata - A Critique'
ibid Systems and Inf Sci Rep 7703 May 1977

Narendra K S and Thathachar M A L
'On the Behaviour of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing'
IEEE Trans on Systems Man and Cybernetics 10 5 May 1980 pp 262 - 269

Østergaard JJ
'Fuzzy Logic Control of a Heat Exchanger Process'
Technical University of Denmark Copenhagen Publication No 7601 January 1976

Poppelbaum W J
'Statistical Processors'
Department of Computer Science University of Illinois at Urbana Illinois USA Report May 1974

Saridis G N
'Self-Organising Control of Stochastic Systems'
Marcel Dekker Inc 1977

Thathachar M A L and John O B
'Discretized Reward-Inaction Learning Automata'
J of Cybernetics and Inf Sci 2 1979 pp 24 - 29

Tou J T (ed)
'Applied Automata Theory'
Academic Press 1968

Watanabe S
'Creative Learning and Propensity Automaton'
IEEE Trans on Systems Man and Cybernetics 5 November 1975 pp 603 - 610

Zadeh L A
'Outline of a New Approach to the Analysis of Complex Systems and Decision Processes'
ibid 3 1 January 1973 pp 28 - 44

APPENDIX A

THE EXPECTED VALUE OF ACTION PROBABILITY

(i) Linear Reward-Penalty

The L_{R-P} reinforcement algorithm can be re-stated linguistically as follows[62]:

$p_1(n+1) = \alpha p_1(n)+(1-\alpha)$, if action a_1 was performed and rewarded, with probability $1-c_1$,

or $\alpha p_1(n)$, if a_2 was performed and rewarded, with probability $1-c_2$,

or $\beta p_1(n)$, if a_1 was performed and penalised, with probability c_1,

or $\beta p_1(n)+(1-\beta)$, if a_2 was performed and penalised, with probability c_2.

From this, the expected value of $p_1(n+1)$ is obtained as:

$$E\left[p_1(n+1)\right] = \left[\alpha p_1+(1-\alpha)\right](1-c_1)p_1+\alpha p_1(1-c_2)(1-p_1) +\beta p_1^2 c_1+\left[\beta p_1+(1-\beta)\right]c_2(1-p_1)$$

$$= p_1\left[\beta p_1 c_1-c_2p_1+(1-\beta)c_2p_1-(1-\beta)c_2-\alpha p_1 +\alpha p_1c_2+\alpha p_1+1-\alpha-\alpha p_1c_1-(1-\alpha)c_1+\beta c_2 +\alpha-\alpha c_2\right]+(1-\beta)c_2$$

$$= p_1^2\left[c_1-(1-\beta)c_1-c_2+(1-\beta)c_2-\alpha+c_2-(1-\alpha)c_2 +\alpha-c_1+(1-\alpha)c_1\right]+p_1\left[1-\alpha-(1-\beta)c_2-(1-\alpha)c_1+c_2 -(1-\beta)c_2+\alpha+(1-\alpha)c_2\right]+(1-\beta)c_2$$

Hence

$$E\left[p_1(n+1)\right] = \underline{(c_2-c_1)(\alpha-\beta)}\ p_1^2(n) + \left[\underline{1+(1-\alpha)(c_2-c_1)-2(1-\beta)c_2}\right]p_1(n) +\underline{(1-\beta)c_2}$$

(ii) /

(ii) Linear Reward-Inaction

The L_{R-I} algorithm can be stated in similar fashion:

$p_1(n+1) = \alpha p_1(n)+(1-\alpha)$, if a_1 was performed and rewarded, with probability $1-c_1$,

or $\alpha p_1(n)$, if a_2 was performed and rewarded, with probability $1-c_2$

or $p_1(n)$, if a penalty response was received.

Then

$$E\left[p_1(n+1)\right] = \left[\alpha p_1+(1-\alpha)\right](1-c_1)p_1+\alpha p_1(1-c_2)(1-p_1) +p_1^2 c_1+p_1(1-p_1)c_2$$

$$= p_1\left[p_1c_1+(1-p_1)c_2+\alpha(1-p_1)(1-c_2) +\left[\alpha p_1+(1-\alpha)\right](1-c_1)\right]$$

$$= p_1\left[p_1c_1+(1-p_1)-(1-\alpha)(1-p_1)(1-c_2)+\alpha p_1+(1-\alpha) -\alpha c_1p_1-(1-\alpha)c_1\right]$$

$$= p_1\left[(1-p_1)\left[1-(1-\alpha)(1-c_2)\right] +(1-\alpha)(1-p_1)(1-c_1)+p_1\right]$$

Hence

$$E\left[p_1(n+1)\right] = \left[1+(1-\alpha)(c_2-c_1)(1-p_1(n))\right]p_1(n)$$

Note that if $\beta = 1$ in the L_{R-P} scheme, then

$$E\left[p_1(n+1)\right] = (c_2-c_1)(\alpha-1)\ p_1^2(n) + \left[1+(1-\alpha)(c_2-c_1)-0\right]p_1(n) +0$$

$$= \left[1+(1-\alpha)(c_2-c_1)-(1-\alpha)(c_2-c_1)p_1(n)\right]p_1(n)$$

$$= \left[1+(1-\alpha)(c_2-c_1)(1-p_1(n))\right]p_1(n)$$

i.e., the formula is reduced to the expression for the L_{R-I} scheme, as derived above.

PUBLICATIONS

1 Neville R G, Nicol C R and Mars P
'Synthesis of Stochastic Learning Automata'
Electronics Letters 14 1978 pp 206 - 208

2 Neville R G, Nicol C R and Mars P
'Design of Stochastic Learning Automata Using Adaptive Digital Logic Elements'
ibid 14 1978 pp 324 - 326

3 Neville R G, Nicol C R and Mars P
'Design of Non-Linear Stochastic Learning Automata'
ibid 14 1978 pp 396 - 397

4* Neville R G and Mars P
'Hardware Synthesis of Stochastic Learning Automata'
Proc 1st Int Symp on 'Stochastic Computing and its Applications
INPT Toulouse France 1978 paper 7.4
pp 345 - 365

5 Neville R G and Mars P
'Hardware Design for a Hierarchical Structure Stochastic Learning Automaton'
J of Cybernetics and Inf Sci 2 1979 pp 30 - 35

6* Neville R G, Chrystall M S and Mars P
'Application of a Hierarchical Structure Stochastic Learning Automaton'
Systems and Inf Sci Becton Centre Yale University USA
Report 7906 September 1979

7 Neville R G and Mars P
'Adaptive Control of Multimodal Stochastic Systems Using Learning Automata'
Paper submitted to the IEE Int Conf on 'Control and its Applications' Warwick University March 1981

*Not enclosed

The first solution has the advantage of a low surface doping, hence providing higher gate-drain avalanche breakdown voltage. The second solution, however, should have a better (lower) source and drain contact resistance, and is probably somewhat easier to approximate in practice. Both profiles may be approximated by epitaxial growth and ion implantation techniques, or a combination of both.

Although the details of the profiles might require alteration to account for the effects of contact resistances and the variation of the drain voltage along the load line excursion, it is believed that the basic features of the profiles will not be altered

ROBERT A. PUCEL

13th January 1978

Research Division
Raytheon Company
Waltham, Massachusetts 02154 USA

References

1 PERLOW, S. M. 'Third-order distortion in amplifiers and mixers', *RCA Rev.*, 1976, 37, pp. 234–267
2 STATZ, H., HAUS, H. A., and PUCEL, R. A.: 'Noise characteristics of gallium arsenide field-effect transistors', *IEEE Trans.*, 1974, ED–21, p. 549
3 PUCEL, R. A., HAUS, H. A., and STATZ, H.: 'Signal and noise properties of gallium arsenide field-effect transistors', *in* 'Advances in electronics and electron physics 38' (Academic, New York, 1975)
4 PUCEL, R. A., MASSE, D. J., and KRUMM, C. F.: 'Signal and noise properties of gallium arsenide field-effect transistors', *IEEE J. Solid-State Circuits*, 1976, SC–11, p. 243
5 WILLIAMS, R. E., and SHAW, D. W.: 'GaAs f.e.t.s with graded channel doping profiles', *Electron. Lett.*, 1977, 13, pp. 408–409

0013-5194/78/0847-0204 $1.50/0

SYNTHESIS OF STOCHASTIC LEARNING AUTOMATA

Indexing terms. Stochastic automata, Logic design, Special purpose computers

The application of digital stochastic computing techniques to the hardware synthesis of stochastic learning automata is considered. Experimental results are presented for a two-state automaton realised using a linear reward/punishment algorithm. The fast learning times obtained are believed to be of significance to the viability of direct on-line control of stochastic systems.

Introduction. In many process control problems, the characteristics of the process are fully known, and a complete mathematical description of the process and of the corresponding control strategy is possible. However, a large number of situations arise where uncertainties are present, either due to an incomplete mathematical model of the process, or due to operation in a random environment. Where the probabilistic nature of these uncertainties is known, stochastic control theory can be applied, but in the case of higher order uncertainties where the probabilistic characteristics cannot be easily ascertained, it is only possible to gain sufficient knowledge of the process by 'on-line' observation. Herein lies the application area for stochastic learning automata (s.l.a.)

A stochastic automaton with variable structure (s.a.v.s.) changes the probabilities of its actions in response to the random inputs from the environment within which it is operating. These changes are brought about by a 'reinforcement scheme' built into the automaton structure such that it tends to converge to a suitable state to satisfy the immediate control requirements of the environment (Fig. 1A). For an r-state automaton at time n,

$$p(n+1) = Tp(n)$$

where $p(n)$ is the vector of total state probabilities and T is a stochastic matrix whose i, jth element, p_{ij}, denotes the probability of transition from state i to state j. Total state probabilities and transition probabilities are thus both valid representations of the s.a.v.s.

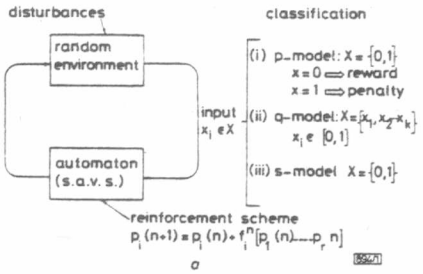


Fig. 1A *Automation-environment feedback configuration*

The techniques of digital stochastic computing[1] provide an ideal method for the practical synthesis of learning automata, and this letter describes the design of systems which will for the first time enable the hardware simulation of stochastic learning automata controllers.

Classification: The automaton/environment interaction can be classified as one of three types, depending on the nature of the environment response, as shown in Fig. 1A. An automaton functioning in a stationary environment is said to be:

(*a*) *expedient* if the asymptotic average penalty M is less than the arithmetic mean of the penalty probabilities

(*b*) *optimal* if the asymptotic average penalty equals the minimum of the penalty probability set, denoted by

$$\{C_i : i = 1, 2, \ldots, r\}$$

(*c*) *ε-optimal* if $M < C_{i(min)} + \epsilon$

The convergent behaviour of the automaton is determined by the algorithm employed in the reinforcement scheme.

Algorithms: A large number of algorithms have been described[2,3] for updating schemes, and their properties compared. Updating can be applied to total state probabilities or transition probabilities. The former may be preferred because updating is performed on a smaller number of quantities, but in other cases the actual transition information may be desired.

In the work described here, the algorithm employed was the simple linear reward-penalty scheme, denoted L_{R-P}. The L_{R-P} scheme applied to a P-model structure was considered to be a suitable system to implement in hardware form, since the binary system response and the stochastic multiplication involved are well-suited to digital circuitry.

The algorithm is stated as follows:

(i) Non penalty: (on action α_i)

$$p_{j \neq i}(n+1) = \alpha p_j(n), \qquad 0 < \alpha < 1$$

$$p_i(n+1) = 1 - \sum_{j \neq i} p_j(n+1)$$

(ii) Penalty: (on action α_i)

$$p_i(n+1) = \beta p_i(n), \qquad 0 < \beta < 1$$

$$p_{j \neq i}(n+1) = p_j(n) + \left\{\frac{1-\beta}{r-1}\right\} p_j(n)$$

An automaton using the above scheme is found to be expedient. In the case of a two-state s.l.a. the algorithm has a

particularly simple form

(a) Non-penalty: (on action α_1)

$$p_1(n+1) = 1 - \alpha p_2(n)$$

$$p_2(n+1) = \alpha p_2(n)$$

(b) Penalty: (on action α_1)

$$p_1(n+1) = \beta p_1(n)$$

$$p_2(n+1) = 1 - \beta p_1(n)$$

Hardware implementation: A 2-state automaton was designed in order to investigate its behaviour and compare results with previous software simulation work.[4] The algorithm circuit shown in Fig. 1B was designed with the aid of a 'truth-table' derived from the algorithm as follows:

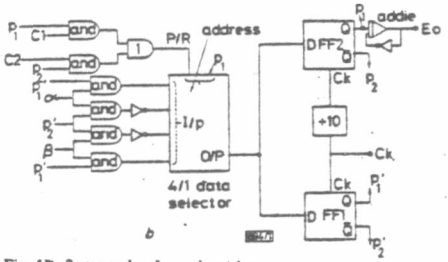


Fig. 1B *2-state s.l.a. L_{RP} algorithm*

$p_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$	$p_2(n+1)$
0	1	0	$\alpha p_1(n)$	$1-\alpha p_1(n)$
0	1	1	$1-\beta p_2(n)$	$\beta p_2(n)$
1	0	0	$1-\alpha p_2(n)$	$\alpha p_2(n)$
1	1	1	$\beta p_1(n)$	$1-\beta p_1(n)$

Clearly, only $p_1(n+1)$ need be formed, since $p_2(n+1)$ is always the complement. Simple AND gates are used to form the products $\alpha p_1(n)$ etc., the appropriate term being selected by the 4/1 data selector, whose 'address' consists of the present state signal $p_1(n)$ and the punishment/reward signal (P/R). The address must not change as rapidly as the algorithm inputs, or the correct factors will not be formed. Therefore a slower clocking rate than that of the algorithm cycle is used for the state output flip-flop (FF 2), from which the address derives.

Thus $p_1(n)$ is represented by the probability of this flip-flop input being logic 1 at the occurrence of a clock pulse, and this probability will either increase or decrease as the system is clocked depending whether that state is rewarded or penalised.

The requisite noise lines for the various constants were obtained from a 31-bit m-length sequence generator, consisting of a 31-bit shift register with exclusive-OR connected feedback from bits 3 and 31.[5] Factors other than 0·5 were simply obtained by AND-gate multiplication, enabling a variety of L_{R-P} schemes, denoted by $\gamma = (1-\alpha)/(1-\beta)$, to be implemented.

Some form of output interface circuit was required in order to enable the characteristics of the s.l.a. to be observed. It was decided that the most suitable way of studying the learning behaviour of the automaton was to display individual 'learning curves' on a storage oscilloscope. The system output which is a stochastic pulse train was converted to an analogue measure of probability by means of the standard adaptive digital logic elements described previously.[6-8]

Experimental results: In the initial experiments, the simulated environment response was set up with $C_1 = 0{\cdot}75$ and $C_2 = 0{\cdot}25$, while γ was varied from 1 to 64 using the available noise sources. The p.r.b.s. generator and s.l.a. clocks were set at 10 MHz and 1 MHz, respectively; the main system flip-flop clock was therefore 100 kHz.

The family of learning curves shown in Fig. 2A clearly shows the increasing expediency resulting from increasing γ. In each case, the system flip-flop was preset initially to p_1, i.e. the 'wrong' state, and the output subsequently converged towards p_2, the state carrying the lower penalty probability.

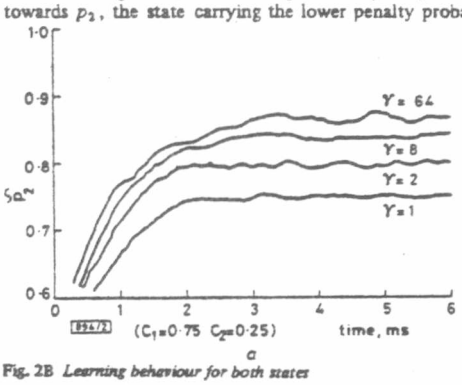


Fig. 2B *Learning behaviour for both states*

With $\gamma = 1$, the system converged to a level corresponding approximately to the reward probability for the state. This is to be expected in a situation where reward and penalty factors are of equal magnitude. The overall characteristics of the system are well summarised in Fig. 2B. This shows the ability of the s.l.a. to lock on whichever state carries the lower penalty probability, from either starting state, using in this case a '$\gamma = 8$' scheme.

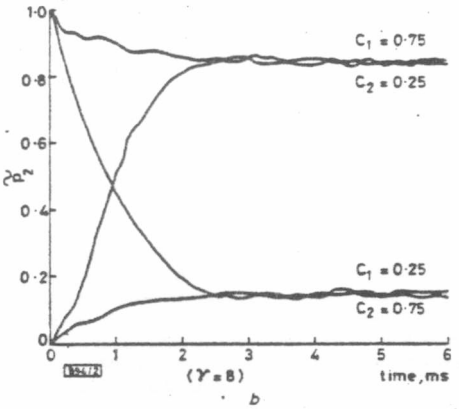


Fig. 2A *Learning curves–L_{RP} scheme*

It was realised that a reward-inaction (L_{R-I}) scheme cannot be implemented with this system, since inspection of the circuit shows that setting $\beta = 1$ (i.e. infinite γ) will cause the flip-flop to remain in whichever state is initially selected.

A significant feature of the results is that learning times of the order of 5 ms can be obtained. It is believed that learning times of this magnitude enable for the first time direct on-line control of several stochastic systems including multimodal stochastic system optimisation and the adaptive routing of telephone traffic systems.[9,10] Further development of the system clearly requires an extension in the number of states, without incurring the penalty of excessive hardware requirements. Work is now in progress on a 128-state hierarchical system using a two-state s.l.a. on a time-shared basis. These results will be reported subsequently.

Acknowledgment: The authors wish to gratefully acknowledge the support of a UK Science Research Council Grant.

R. G. NEVILLE
C. R. NICOL
P. MARS

School of Electronic and Electrical Engineering
Robert Gordon's Institute of Technology
Aberdeen AB9 1FR, Scotland

2nd February 1978

References

1 GAINES, B. R. 'Stochastic computing', *AFIPS SJCC*, 1967, **30**, pp. 149–156
2 NARENDRA, K. S., and THATHACHAR, M. A. L.: 'Learning automata–a survey', *IEEE Trans.*, 1974, **SMC-4**, pp. 323–334
3 VISWANATHAN, R., and NARENDRA, K. S.: 'Expedient and optimal variable structure stochastic automata'. Becton Centre, Yale University, 1970, Technical Report CT-31
4 VISWANATHAN, R., and NARENDRA, K. S.: 'Comparison of expedient and optimal reinforcement schemes for learning systems', *J. Cybern.*, 1972, **2**, pp. 21–37
5 MILLER, A. J., and MARS, P.: 'Theory and design of a digital stochastic computer random number generator', *Trans. IMACS*, 1977, **19**, pp. 198–216
6 MILLER, A. J., BROWN, A. W., and MARS, P.: 'Adaptive logic circuits for digital stochastic computers', *Electron. Lett.*, 1973, **9**, pp. 500–502
7 MILLER, A. J., BROWN, A. W., and MARS, P.: 'Study of an output interface for digital stochastic computers', *Int. J. Electron.*, 1974, **37**, pp. 637–655
8 MILLER, A. J., and MARS, P.: 'Optimal estimation of digital stochastic sequences', *Int. J. Syst. Sci.*, 1977, **8**, pp. 683–696
9 SHAPIRO, I. J., and NARENDRA, K. S.: 'Use of stochastic automata for parameter self-optimisation with multi-modal performance criteria', *IEEE Trans.*, 1969, **SSC-5**, pp. 352–360
10 NARENDRA, K. S., WRIGHT, E. A., and MASON, L. G.: 'Application of learning automata to telephone traffic routing and control', *IEEE Trans.*, 1977, **SMC-7**, pp. 785–792

ERRATA

VERRAZZANI, L.: 'Bandwidth and demodulation gain in q.s.s.b. f.m.', *Electron. Lett.*, 1978, **14**, pp. 18–19

The author would like to make the following corrections to his paper:

The third expression after eqn. 1 should read

$$R_s(\tau) = \tfrac{1}{2}\,\mathrm{Re}\,(\exp(j\omega_0\tau)\langle\exp\{-\alpha[\hat{f}(t)+\hat{f}(t+\tau)] - j\beta[f(t)-f(t+\tau)]\}\rangle)$$

In the first expression on p. 19 the first equality for P_s/P_t should read

$$\frac{P_s}{P_t} = \frac{R^+(0)}{R_+(0)}$$

The caption to Fig. 1 should read

Fig. 1 *Mean-square bandwidth of q.s.s.b. f.m. versus fraction of power in sidebands for lowpass rectangular input spectrum with unity cutoff frequency*

ADDENDA

SHAMASH, Y.: 'Computing the invariant zeros of multivariable systems', *Electron. Lett.*, 1977, **13**, pp. 722–723

The author would like to make the following addition to his letter:

In this letter, a method was suggested for computing the invariant zeros of multivariable systems. The author has since been made aware, through private communications with Professor B. Porter, of the fact that the essential idea of the method had been independently prepublished by Kwahernaak and Sivan.[1] The author wishes to thank Professor Porter for bringing this to his attention.

Reference

1 KWAHERNAAK, H., and SIVAN, R.: 'Linear optimal control systems' (Wiley, 1972)

CRANE, R. K., and DEBRUNNER, W. E.: 'Worst month statistics', *Electron. Lett.*, 1978, **14**, pp. 38–40

The authors would like to make the following addition to the acknowledgments section of their paper:

Dr. Crane's participation in the work of CCIR Study Group 5 was supported by the National Aeronautics and Space Administration under Contract NAS 5-24209.

recovered signal is proportional to the frequency deviation and is a differentiated version of the original modulating signal. Thus, for a small group delay τ, the multimode fibre acts as a frequency discriminator. In low dispersion graded-index fibres $\tau \approx 1$ ns/km, and hence the delay demodulation effect is negligible except at very high acoustic frequencies and large phase deviations.

(*c*) *Triple-transit echo:* p.m. to a.m. conversion may also occur if the optical fibre end reflections (which have made multiple transits of the fibre) interfere (or homodyne) with the direct signal. If we represent the direct signal by expr. 1, then, considering any one mode, we may write for the most significant optical reflection, i.e. the triple-transit echo,

$$\tfrac{1}{2}\sqrt{\{r_1 r_2 10^{-2\alpha l}\}}E_c \sin\,[\omega_c t + 3\Delta\theta \sin \omega_m t + \phi] \quad (13)$$

where $r_1 = r_2$ = optical intensity reflection coefficients ($\approx$3·5% for a silica-air interface), α = optical attenuation constant and l = length of fibre. Notice that the modulation index for the first echo is $3\Delta\theta$ since it has passed through the modulated region of the fibre three times. We shall assume that the propagation time for the echo ($\approx 5\,\mu$s/km) is insignificant with respect to the period of the modulating signal. Going through the same procedure as before and expanding in terms of Bessel functions, we may write for the photodiode current

$$\begin{aligned} i(t) = \tfrac{1}{2}\sqrt{\{r_1 r_2 10^{-2\alpha l}\}}\,[1 &+ J_0(2\Delta\theta)\cos\phi \\ &+ 2J_1(2\Delta\theta)\sin\phi \sin\omega_m t \\ &+ 2J_2(2\Delta\theta)\cos\phi \cos 2\omega_m t + \ldots] \end{aligned} \quad (14)$$

Again, for small $\Delta\theta$ and quadrature optical bias ($\phi = \pm\pi/2$), we may write

$$i(t) = \pm\sqrt{\{r_1 r_2 10^{-2\alpha l}\}}\,\Delta\theta \sin \omega_m t \quad (15)$$

Eqn. 15 reveals an important feature of optical homodyne or heterodyne systems, in that the echo return loss $\sqrt{\{r_1 r_2 10^{-2\alpha l}\}}$ has a minimum value of 29 dB (assuming 3·5% fibre end reflections, perfect temporal coherence and zero fibre attenuation), whilst in direct (incoherent) detection systems it has a minimum value of 58 dB, This difference arises from the fact that a square-law detector functions as a linear demodulator in coherent systems. Thus echo signals in coherent optical fibre systems are a potential source of interference to those signals produced by the differential phase discriminator effect (*a*). The relative sensitivity of discriminators (*a*) and (*c*) depends not only on coherence, echo return loss, number of propagating modes and mode-mixing effects, but also on the acousto-optic interaction length and as to how the fibre is subjected to stress.

Conclusions: The fibredyne system described and analysed above is a very simple device for exploiting the microphonic aspects of multimode fibres, where the fidelity of reproduction is not important, e.g. in intruder alarms. The analysis also indicates that problems may arise in conventional direct detection fibre systems if the laser is to coherent. Experimental evidence of these p.m. to a.m. phenomena will form the subject of the second paper in this series.

Acknowledgments: This work was supported by the UK Science Research Council. We are indebted to both GEC and STL for provision of optical fibres.

S. A. KINGSLEY
D. E. N. DAVIES

Department of Electronic & Electrical Engineering
University College London
Torrington Place
London WC1 7JE
England

17th April 1978

References

1 SMITH, L. W., and SNITZER, E.: 'Final development report for fibre optics information processor'. American Optical Corporation Report 600–TR–F, 1969
2 DAVIES, D. E. N., and KINGSLEY, S. A.: 'An optical fibre data collection highway'. Proceedings of Electro-optics/Laser International '76 UK, Brighton, 1976, pp. 64–72
3 KINGSLEY, S. A., and DAVIES, D. E. N.: 'Use of optical fibres as instrumentation transducers'. Digest of technical papers, Conference on laser and electro-optical systems, San Diego, Calif., 1976, pp. 24–25
4 NELSON, D. F., KLEINMAN, D. A., and WECHT, K. W.: 'Vibration induced modulation of fibreguide transmission', *Appl. Phys. Lett.*, 1977, **30**, pp. 94–96
5 BUCARO, J. A., DARDY, H. D., and CAROME, E. F.: 'Optical fibre acoustic sensor', *Appl. Opt.*, 1977, **16**, pp. 1761–1762
6 BUCARO, J. A., DARDY, H. D., and CAROME, E. F.: 'Fibre-optic hydrophone', *J. Acoust. Soc. Am.*, 1977, **62**, pp. 1302–1304
7 COLE, J. H., JOHNSON, R. L., and BHUTA, P. G.: 'Fibre-optic detection of sound', *ibid.*, 1977, **62**, pp. 1136–1138
8 CULSHAW, B., DAVIES, D. E. N., and KINGSLEY, S. A.: 'Acoustic sensitivity of optical-fibre waveguides', *Electron. Lett.*, 1977, **13**, pp. 760–761

0013-5194/78/1053-0322 $1.50/0

DESIGN OF STOCHASTIC LEARNING AUTOMATA USING ADAPTIVE DIGITAL LOGIC ELEMENTS

Indexing terms: Stochastic automata, Logic design, Special purpose computers, Learning systems

The hardware design of stochastic learning automata using adaptive digital logic elements is considered. Such techniques, based on digital stochastic computing, are shown to provide economical and fast learning-time computations. Experimental results are presented for a variety of linear learning algorithms.

Introduction: One of the potential areas for applying the results of stochastic computing research[1–3] is in the implementation of learning systems for optimal control using stochastic automata structures. A stochastic automaton with a variable structure (s.a.v.s.) changes the probabilities of its actions in response to inputs from a random environment.[4] A 'reinforcement scheme' built into the automaton causes updating of the action probabilities so as to improve performance and produce convergence to a suitable final structure.[5] Recently a simple flip-flop stochastic learning automaton based on a linear reward/penalty (L_{R-P}) algorithm has been described.[6] In order to incorporate superior learning algorithms and to improve the viability of large-state order systems attention has been focused on improving the original hardware design. A consideration of the various reinforcement algorithms shows that it is essential to include a memory capability within the automaton structure in such a manner as to establish priority of state probabilities during the learning period. If this is not so the past experience of the stochastic learning automaton is erased after each system cycle (or clock pulse). Such considerations led to the idea of representing the probability of state occupation not simply by the probability of a flip-flop being in a certain state at the occurrence of a clock pulse, but by a number stored in a counter, which may be subsequently converted to a stochastic sequence. The result

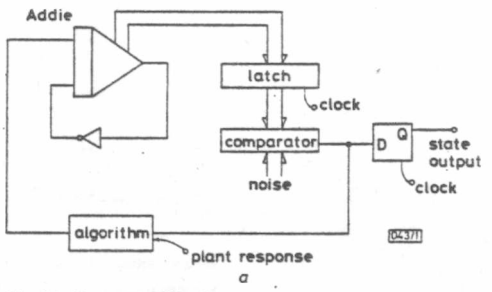


Fig. 1A *Two-state Addie s.l.a.*

is the evolution of a new design for a hardware learning automaton based on the adaptive digital logic elements (Addies) described previously.[7] It should be noted that the use of Addie structures has also been proposed for the related 'two-armed' bandit problem.[8]

Design of the Addie stochastic learning automaton: A 2-state stochastic learning automata can be implemented using a single Addie, as shown in Fig. 1A; The contents of the Addie counter represent state probability $p_1(n)$, while $p_2(n)$ is simply taken to be the complement.

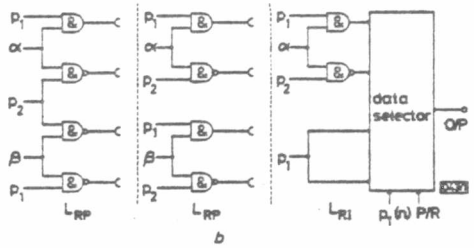


Fig. 1B *Algorithm circuits for linear schemes*

An essential feature of the operation of the automaton is the updating of state probabilities in accordance with the environment or plant response. This is achieved in the Addie s.l.a. by loading $p_1(n)$ from the Addie to a latch, and performing digital/stochastic conversion. The resulting stochastic pulse train is then transformed via the algorithm circuitry to an updated state probability $p_1(n+1)$. The Addie then reaches an estimate of $p_1(n+1)$, and, after a suitable settling time, the next cycle can commence. A flip-flop on the comparator output represents the present state occupied, and state trajectories can be observed by filtering the output, or by direct digital/analogue conversion of the Addie contents.

The operating sequence of the Addie s.l.a. is as follows: The initial load operation sets up the requisite value of 0·5 (i.e. 'one-all zeros') in the counter, so that the output of the comparator is a stochastic sequence with an equal probability of 1s and 0s, representing random state selection at initial time t_0. At the first clock pulse, this sequence is sampled and at the same time, the counter contents are copied into the latch. Then, when the clock pulse goes low, the punishment/reward signal resulting from the state of the *D*-type flip-flop is latched, and the Addie clock enabled, allowing the 'learning period' to commence. During this time, the Addie converges to the new value of $p_1(1)$, which is then used as the basis for the next cycle.

The advantage of this design is that, since no locking-on problems can occur, it is possible to implement the more suitable ϵ-optimal schemes, using the established method of algorithm circuit design based on stochastic computing techniques described previously.[6]

Algorithm circuit design: As mentioned earlier, the Addie stochastic learning automaton enables several of the reinforcement schemes described previously[5] to be implemented.

The linear reward/penalty scheme L_{R-P} may be described for the two state case as:

(*a*) Reward: (action a_1)

$$p_2(n+1) = \alpha p_2(n)$$
$$p_1(n+1) = 1 - \alpha p_2(n)$$

(*b*) Penalty: (action a_1)

$$p_1(n+1) = \beta p_1(n)$$
$$p_2(n+1) = 1 - \beta p_1(n)$$

where $0 < \alpha < 1$ and $0 < \beta < 1$.

Similar expressions hold for action a_2. The hardware implementation for the L_{R-P} scheme has been described and is shown in Fig. 1B (Reference 6). The obvious extension of this scheme is to implement the ϵ-optimal linear reward-reward (L_{R-R}) and reward-inaction (L_{R-I}) schemes. The L_{R-I} scheme is particularly simple to accommodate, since the only modification required is to set the factor $\beta = 1$. The L_{R-R} scheme, in which the penalty is replaced by a lesser reward, is given below in two-state form:

(*i*) Nonpenalty: (on action a_1)

$$p_1(n+1) = 1 - \alpha p_2(n)$$
$$p_2(n+1) = \alpha p_2(n)$$

(*ii*) Penalty: (on action a_1)

$$p_1(n+1) = 1 - \beta p_2(n)$$
$$p_2(n+1) = \beta p_2(n)$$

where $0 < \beta < \alpha < 1$.

As before a truth-table is constructed to enable the algorithm to be translated into a circuit design:

$p_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$
0	1	0	$\alpha p_1(n)$
0	1	1	$\beta p_1(n)$
1	0	0	$1 - \alpha p_2(n)$
1	0	1	$1 - \beta p_2(n)$

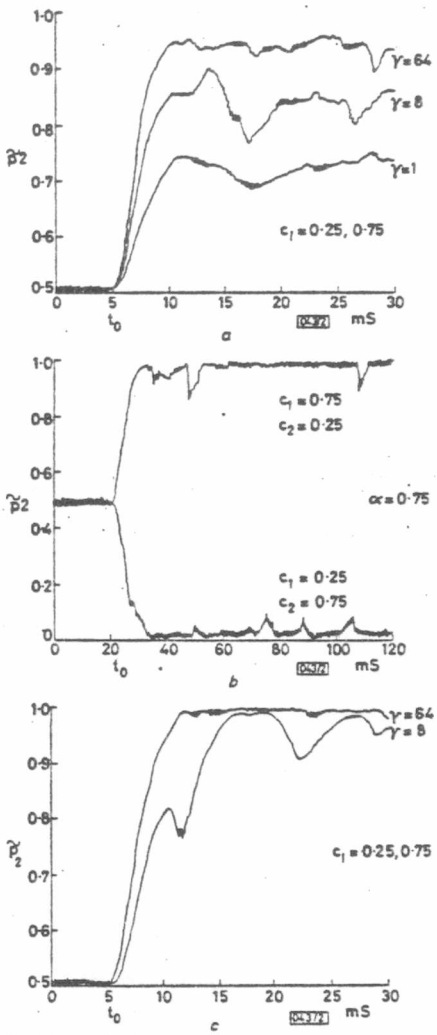


Fig. 2 *Learning curves*

a L_{R-P} scheme
b L_{R-I} scheme
c L_{R-R} scheme

By comparison with the truth-table for the L_{R-P} scheme,[6] it is evident that the L_{R-P} circuit can be converted to an L_{R-R} circuit simply by reversing the $\beta p_1(n)$ and $1-\beta p_2(n)$ connections. The circuit arrangements for these linear schemes are summarised in Fig. 1B.

Experimental results: The Addie stochastic learning automaton was tested using a storage oscilloscope to observe the state trajectories directly. Learning curves obtained with the L_{R-P} scheme are shown in Fig. 2*a*. The curves show how the degree of expediency increases with reward/penalty ratio γ ($\gamma = (1-\alpha)/(1-\beta)$). It should be noted that the counter implementation allows a proper initial condition to be set up, corresponding to random state selection at time t_0, i.e. $p_1(0) = 0{\cdot}5$. For these experiments, the Addie clock was set at 10 MHz and the noise generator clock at 1 MHz, while the main system clock consisted of 100 ns pulses with 100 kHz repetition rate.

The results obtained with the ϵ-optimal L_{R-I} scheme are illustrated in Fig. 2*b*. With this scheme the automaton exhibits virtually full convergence to $p_1 = 1$ or $p_2 = 1$. It is known that the L_{R-R} scheme is comparable in expediency to the L_{R-I} scheme when γ is high but the L_{R-R} scheme is expected to exhibit a slower rate of convergence. The results obtained with a L_{R-R} scheme are shown in Fig. 2*c*. As expected the degree of expediency increases with γ.

The general conclusion to be gained from the above results on linear schemes is that there is little difference in expediency between the schemes with high γ-factor. The criterion of relative convergence rates, which is stressed in the reported software simulation studies,[9] is less important in the work reported here. Indeed, there appears to be no discernible difference in learning time between the schemes investigated. The chief determining factor in the transient response of the hardware systems is the output Addie. This element has a restricted bandwidth in order to obtain an acceptable compromise between convergence speed and steady-state variance.

Conclusions: The above results have verified that the Addie stochastic learning automaton has very satisfactory learning characteristics. Although its operation involves a certain amount of serial processing, similar learning times to the flip-flop automaton have been obtained. Indeed the results obtained previously[6] for the L_{R-P} scheme have been virtually duplicated by the new circuit, which has of course the added advantage of being able to implement a more comprehensive range of reinforcement schemes. The ability to optimise in less than 10 ms represents a significant development, particularly in view of the proposed extension to systems of much larger state order, embodying the same basic Addie stochastic automaton within a hierarchical structure.

Acknowledgment: The authors wish to gratefully acknowledge the support of a UK Science Research Council grant.

R. G. NEVILLE
C. R. NICOL
P. MARS

Robert Gordon's Institute of Technology
School of Electronic & Electrical Engineering
Schoolhill, Aberdeen AB9 1FR
Scotland

12th April 1978

References

1 GAINES, B. R.: 'Stochastic computing', *AFIPS SJCC*, 1967, **30**, pp. 149–156
2 MILLER, A. J., and MARS, P.: 'Theory and design of a digital stochastic computer random number generator', *Trans. IMACS*, 1977, **19**, pp. 198–216
3 MILLER, A. J., and MARS, P.: 'Optimal estimation of digital stochastic sequences', *Int. J. Syst. Sci.*, 1977, **8**, pp. 683–696
4 NARENDA, K. S., and THATHACHAR, M. A. L.: 'Learning automata – a survey', *IEEE Trans.*, 1974, SMC-4, pp. 323–334
5 VISWANATHAN, R., and NARENDA, K. S.: 'Expedient and optimal variable structure stochastic automata', Becton Centre, Yale University, 1970, Tech. Rep. CT-31
6 NEVILLE, R. G., NICOL, C. R., and MARS, P.: 'Synthesis of stochastic learning automata', *Electron. Lett.*, 1978, **14**, pp. 206–208
7 MILLER, A. J., BROWN, A. W., and MARS. P.: 'Adaptive digital logic circuits for digital stochastic computers', *ibid.*, 1973, **9**, pp. 500–502
8 WITTEN, I. J.: 'Stochastic implementation of learning controllers', IEE colloquium on parallel digital computing methods, 1976 Digest 1976/30, Paper 6
9 VISWANATHAN, R., and NARENDA, K. S.: 'Simulation studies of stochastic automata models', Becton Centre, Yale University, 1971, Tech. Rep. CT-45

0013-5194/78/1043-0324 $1.50/0

MATERIAL DISPERSION IN LIGHTGUIDE GLASSES

Indexing terms: Optical fibres, Glass

Material dispersion measurements are reported on six characteristic lightguide glass compositions. The measurements were made on bulk specimens and cover the wavelength range from 0·8 to 1·5 μm. It is observed that in these silicate glasses the wavelength at which material dispersion is zero is in all cases greater than 1·2 μm.

Introduction: In order to increase the bandwidth of lightguides it is desirable to minimise material dispersion—the wavelength dependence of the light group velocity in the transmission medium. Measurements of material dispersion have been reported on certain bulk samples of lightguide glasses[1,2] and also on optical fibre specimens.[3–5] Both types of analyses are being pursued at this laboratory. Although bulk specimens may not duplicate the materials comprising those in lightguides, dispersion measurements on bulk specimens provide the following advantages: (*a*) accurate sample chemical composition determinations can be made on bulk material; (*b*) measurements are free of waveguide effects which may complicate dispersion analyses on fibres; (*c*) measurements can be performed on materials contemplated for optical waveguide claddings such as B_2O_3–SiO_2 compositions; (*d*) dispersion can be analysed in materials prior to the achievement of low-loss lightguides made from them.

The results of such measurements obtained on characteristic compositions of materials currently used in waveguides are reported herein.

Experimental: The glass compositions on which material dispersion is reported are listed in Table 1. Glass A–D were prepared at this laboratory by r.f. plasma fusion of vapour or presintered powder, glass E was prepared by Hereaus Quarzschmelze and glass F was prepared at this laboratory by a conventional high-purity crucible melting technique. The compositions reported are those indicated by chemical analysis of the prepared glass and are accurate to ±0·1 mole %.

Refractive indices were measured at the wavelengths given in Table 2. The measurement method utilised was the

Table 1 COMPOSITIONS AND ZERO MATERIAL DISPERSION FOR GLASSES STUDIED

Sample	Composition	Zero material dispersion
	Moles	μm
A	Quenched SiO_2	1·284
B	$13{\cdot}5GeO_2 : 86{\cdot}5SiO_2$	1·383
C	$9{\cdot}1P_2O_5 : 90{\cdot}9SiO_2$	1·274
D	$13{\cdot}3B_2O_3 : 86{\cdot}7SiO_2$	1·231
E	$1{\cdot}0F : 99{\cdot}0SiO_2$	1·284
F	$16{\cdot}9Na_2O : 32{\cdot}5B_2O_3 : 50{\cdot}6SiO_2$	1·283

References

1 JOSEPHY, R. D.: 'MOS-Transistoren zur Leistungsverstärkung im HF-Bereich', *Philips Tech. Rdsch.*, 1970/71, 31, pp. 262–269
2 MORITA, Y., TAKAHASHI, H., MATAYOSHI, H., and FUKUTA, M.: 'Si UHF MOS high-power FET', *IEEE Trans.*, 1974, ED-21, pp. 733–734
3 OAKES, J. G., WICKSTROM, R. A., TREMERE, D. A., and HENG, T. M. S.: 'A power silicon microwave MOS transistor', *ibid.*, 1976, MTT-24, pp. 305–311
4 REINDL, K.: 'Spun on arsenosilica films as sources for shallow arsenic diffusions with high surface concentration', *Solid-State Electron.*, 1973, 16, pp. 181–189
5 DECLERCK, G. J., HATTORI, T., MAY, G. A., BEAUDOUIN, J., and MEINDL, J. D.: 'Some effects of trichloroethylene oxidation on the characteristics of MOS devices', *J. Electrochem. Soc.*, 1975, 122, pp. 436–439

0013-5194/78/1112-0394 $1.50/0

DESIGN OF NONLINEAR STOCHASTIC LEARNING AUTOMATA

Indexing terms: Stochastic automata, Logic design, Special purpose computers, Learning systems

The hardware design of nonlinear stochastic learning automata using adpative digital logic elements is considered. Such techniques, based on digital stochastic computing, are shown to provide faster convergence rates than automata based on linear learning algorithms. Experimental results are presented and validation obtained for theoretical predictions concerning optimal convergence.

Introduction: A significant application area for digital stochastic computing techniques[1-3] is in the hardware synthesis of stochastic learning automata. Stochastic automata modify their action probabilities in response to inputs from a random environment. Previous work has demonstrated synthesis techniques for stochastic automata using a variety of linear reinforcement schemes including reward/penalty, reward/inaction and reward/reward.[4,5]

Although the best of the linear schemes, the L_{RI} (reward/inaction) scheme has been widely reported[6] as most suitable for many applications, investigations have also been made of nonlinear updating schemes.[7,8] These tend to show faster initial convergence rates, and indeed one reason for the emphasis placed on these schemes is to obtain optimum convergence times, especially when they are incorporated in hybrid schemes. The present work presents a synthesis technique for nonlinear stochastic learning automata and provides experimental results for learning characteristics including conditions for optimal convergence.

Nonlinear learning algorithm: Previous work[5] has demonstrated the design of a 2-state stochastic learning automaton using adaptive digital logic elements (Addies). The basic schematic of the system which has been described previously is shown in Fig. 1*a*.

The simplest of the nonlinear schemes is that denoted as $N_{R-P}^{(1)}$, which has 'square-law' nonlinearity. It has been shown[8] that this scheme is conditionally optimal, providing optimal convergence if $c_1 < \frac{1}{2} < c_2$ and expedient otherwise. (c_i represents penalty probabilities). This scheme for the two-state case, is given below:

(*a*) *Non-penalty* (on action a_1)

$$p_1(n+1) = p_1(n) + \alpha p_1(n)[1 - p_1(n)]$$

$$p_2(n+1) = p_2(n) - \alpha p_1(n)[1 - p_1(n)]$$

(*b*) *Penalty*

$$p_1(n+1) = p_1(n) - \beta p_1(n)[1 - p_1(n)]$$

$$p_2(n+1) = p_2 + \beta p_1(n)[1 - p_1(n)]$$

$$0 < \alpha, \beta < 1$$

The truth-table for this scheme is as follows:

$p_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$	$p_2(n+1)$
0	1	0	$p_1 - \alpha p_2(1 - p_2)$	$p_2 + \alpha p_2(1 - p_2)$
0	1	1	$p_1 + \beta p_2(1 - p_2)$	$p_2 - \beta p_2(1 - p_2)$
1	0	0	$p_1 + \alpha p_1(1 - p_1)$	$p_2 - \alpha p_1(1 - p_1)$
1	0	1	$p_1 - \beta p_1(1 - p_1)$	$p_2 + \beta p_1(1 - p_1)$

The hardware implementation of the scheme is complicated by two factors. The first is the presence of terms of the form $p_i(n)[1 - p_i(n)]$. This product cannot simply be formed by an AND gate because of the direct complementary relationship between the two signals. The solution here is to interpose a delay on one of the inputs, along the lines of the conventional stochastic squarer circuit.[1] The other difficulty is that summation in stochastic form requires an AND-OR configuration with random selection of the two signals by a separate p.r.b.s. noise line. This entails an overall division by two.

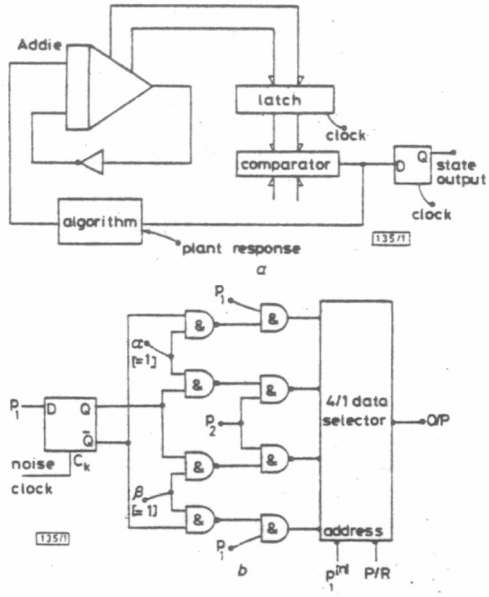


Fig. 1
a Two-state Addie s.l.a.
b Algorithmic circuit for $N_{R-P}^{(1)}$ scheme

It is therefore preferable to rearrange the algorithm terms so that they contain only multiplication and inversion operations. The algorithm circuitry is designed to generate $p_1(n+1)$ alone and the algorithm terms are then transformed as follows:

$$p_1 + \alpha p_1(1 - p_1) = 1 - [p_2 - \alpha p_1(1 - p_1)]$$

$$= 1 - [(1 - \alpha p_1)p_2]$$

$$p_1 - \beta p_1(1 - p_1) = (1 - \beta p_2)p_1, \text{ and so on.}$$

This leads to a revised truth-table:

$p_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$
0	1	0	$(1 - \alpha p_2)p_1$
0	1	1	$1 - [(1 - \beta p_1)p_2]$
1	0	0	$1 - [(1 - \alpha p_1)p_2]$
1	0	1	$(1 - \beta p_2)p_1$

The hardware implementation for this scheme, with $\alpha = \beta = 1$, is shown in Fig. 1*b*. Comparing this circuit with that for the L_{R-P} scheme described previously, an essential similarity is evident.[5] In the $N_{R-P}^{(1)}$ scheme, the simple constants α, β are replaced with terms of the form $(1 - \alpha p_i)$ and $(1 - \beta p_i)$.

Experimental results: The $N_{RP}^{(1)}$ scheme considered is the simplest of the nonlinear schemes to be reported, and is fairly easy to implement, as described above. It was possible to verify in practice the property of conditional optimality mentioned earlier by suitable choice of penalty probabilities c_i. With c_i values of 0·25 and 0·875, the system yielded the optimal convergence curves shown in Fig. 2*a*, with generally lower variance than was evident in the case of the best linear schemes.

The conditional optimality characteristic of this scheme was investigated by feeding the penalty probabilities via a switching circuit, so that the degree of convergence as the system was switched from 'optimal' to 'expedient' conditions could be observed. Fig. 2*b* shows the resulting state outputs, with the switching waveform superimposed (central trace).

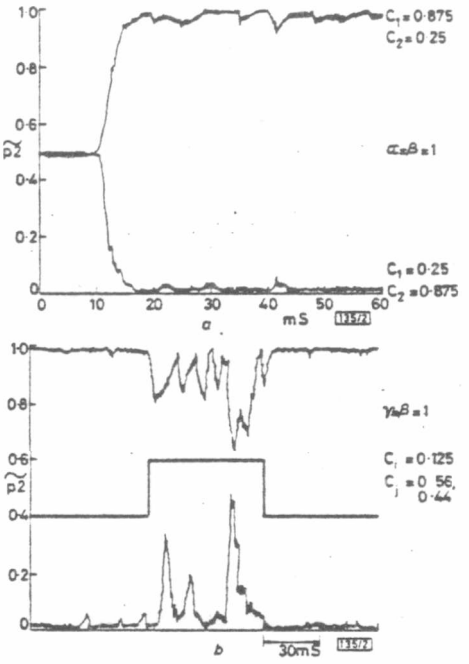


Fig. 2
a Learning curves for N_{R-P} scheme
b Optimal/Expedient switching characteristics

One penalty probability was fixed at 0·125, and the other was switched periodically about the critical point of 0·5 from 0·56 (switching waveform low) to 0·44 (switching waveform high). The results show that with $c_i = \{0{\cdot}125, 0{\cdot}56\}$, the state trajectories exhibit optimal convergence, whereas with $c_i = \{0{\cdot}125, 0{\cdot}44\}$, the system degrades to expedient behaviour. These results are entirely consistent with previous theoretical predictions.[8]

Conclusions: A design technique has been described for the synthesis of nonlinear stochastic automata based on digital stochastic computing methods. Experimental results have demonstrated the fast learning times obtained from nonlinear algorithms and proved the validity of previous theoretical predictions concerning conditions for optimal convergence.

Acknowledgment: The authors wish to gratefully acknowledge the support of a UK Science Research Council grant.

R. G. NEVILLE
C. R. NICOL
P. MARS

School of Electronic and Electrical Engineering
Robert Gordon's Institute of Technology
Schoolhill
Aberdeen AB9 1FR
Scotland

12th May 1978

References

1 GAINES, B. R.: 'Stochastic computing', *AFIPS SJCC*, 1967, 30, pp. 149–156
2 MILLER, A. J., and MARS, P.: 'Theory and design of a digital stochastic computer random number generator', *Trans. IMACS*, 1977, 19, pp. 198–216
3 MILLER, A. J., and MARS, P.: 'Optimal estimation of digital stochastic sequences', *Int. J. Syst. Sci.*, 1977, 8, pp. 683–696
4 NEVILLE, R. G., NICOL, C. R., and MARS, P.: 'Synthesis of stochastic learning automata', *Electron. Lett.*, 1978, 14, pp. 206–208
5 NEVILLE, R. G., NICOL, C. R., and MARS, P.: 'Design of stochastic learning automata using adaptive digital logic elements', *ibid.*, 1978, 14, pp. 324–326
6 VISWANATHAN, R., and NARENDRA, K. S.: 'Comparison of expedient and optimal reinforcement schemes for learning systems', *J. Cybern.*, 1972, 2, pp. 21–37
7 VISWANATHAN, R., and NARENDRA, K. S.: 'Simulation studies of stochastic automata models', Becton Centre, Yale University, Tech. Rept. CT-45, December 1971
8 VISWANATHAN, R., and NARENDRA, K. S.: 'Expedient and optimal variable-structure automata', Becton Centre, Yale University, Tech. Rept. CT-31, April 1970

0013-5194/78/1135-0396 $1.50/0

CYCLOTRON AND UPPER HYBRID RESONANCE FREQUENCY IN REFLECTED PULSE BY MAGNETOPLASMA HALF SPACE

Indexing terms: Cyclotron resonance, Plasma waves

Reflected waveforms of impulsive plane waves incident on a homogeneous lossless magnetoplasma half space (with d.c. magnetic field perpendicular to the propagation vector) are determined in the closed form by Bessel functions of first kind and fractional order. The cyclotron plasma frequency is related to the first normalised maximum excursion in the reflected waveforms. Cyclotron and upper hybrid resonance plasma frequencies are related to the delay of the first maximum excursion in the reflected waveforms.

Introduction: The reflection of an impulsive plane wave by a lossless magnetoplasma half space has been studied previously for small to moderate plasma anisotropy ($\alpha_B = \omega_B/\omega_p \leqslant 0{\cdot}3$) by the perturbation method (Stanić *et al.*[1]) and numerically for strong anisotropy (Jinno *et al.*[2]). In both cases the impulsive plane wave (with the wave vector perpendicular to the external d.c. magnetic field) was incident normally to a magnetoplasma half space with sharp boundary. Schmitt[3] made experiments with nanosecond pulses and used transient signals as a diagnostic tool. Here, the time domain solution is obtained in closed form by the standard Laplace transform technique for an arbitrary value of anisotropy parameter $\alpha_B = \omega_B/\omega_p$.

Theory: An electromagnetic pulse is incident to the free space ($z < 0$)– plasma ($z > 0$) interface ($z = 0$). The wave vector is perpendicular to the external d.c. magnetic field. The reflection coefficient in the frequency domain is

$$R(\omega) = \frac{k_p/k_0 - 1}{k_p/k_0 + 1} \tag{1}$$

where

$k_0 = \omega/c$ is the free space wave number

$k_p = k_0 \dfrac{(\omega^2 - \omega_1^2)(\omega^2 - \omega_2^2)}{\omega^2(\omega^2 - \omega_{UH}^2)}$ is the magnetoplasma wave number

$\omega_{UH}^2 = \omega_p^2 + \omega_B^2 = \omega_p^2(1 + \alpha_B^2)$ is the upper hybrid frequency

$\omega_{1,2}^2 = (\frac{1}{2})\omega_p^2(2 + \alpha_B^2 \pm \alpha_B(4\alpha_B^2 + 1)^{1/2})$

$\alpha_B = \omega_B/\omega_p$ is the anisotropy parameter

ω_B is the electron cyclotron frequency

ω_p is the electron plasma frequency

HARDWARE DESIGN FOR A HIERARCHICAL STRUCTURE STOCHASTIC LEARNING AUTOMATON

Richard G. Neville and Philip Mars

School of Electronic and Electronic Engineering
Robert Gordon's Institute of Technology
Schoolhill
ABERDEEN AB9 1FR, Scotland

The hardware design of stochastic learning automata using adaptive digital logic elements (ADDIES) is considered. Such techniques, based on earlier research into digital stochastic computing, are capable of providing usefully fast learning time computations, and experimental results are presented here for a variety of linear and nonlinear learning algorithms.

A hardware design for a 128-state stochastic learning automaton using a hierarchical structure is then described, and experimental results of static and dynamic optimisation are presented. This system is shown to be capable of fast, economical learning behaviour suitable for the practical implementation of on-line learning controllers.

1. INTRODUCTION

One of the potential areas for applying the results of stochastic computing research[1] [2] [3] is in the implementation of learning systems for optimal control using stochastic automata structures. A stochastic automaton with a variable structure (SAVS) changes the probabilities of its actions in response to inputs from a random environment[4]. A "reinforcement scheme" built into the automaton causes updating of the action probabilities so as to improve performance and produce convergence to a suitable final structure[5]. Recently a simple flip-flop stochastic learning automaton based on a linear reward/penalty (L_{R-P}) algorithm was described[6]. In order to incorporate superior learning algorithms and to improve the viability of large state order systems attention has been focussed on improving the original hardware design. A consideration of the various reinforcement algorithms shows that it is essential to include a memory capability within the automaton structure in such a manner as to establish priority of state probabilities during the learning period. If this is not so the past experience of the automaton is erased after each system cycle (or clock pulse). Such considerations led to the idea of representing the probability of state occupation not simply by the probability of a flip-flop being in a certain state at the occurrence of a clock pulse, but by a number stored in a counter, which may be subsequently converted to a stochastic sequence. The result is the evolution of a new design for a hardware learning automaton based on the adaptive digital logic elements (ADDIES) described previously[7]. It should be noted that the use of ADDIE structures has also been proposed for the related "two-armed" bandit problem[8].

2. DESIGN OF THE ADDIE STOCHASTIC LEARNING AUTOMATON

A 2-state stochastic learning automaton (SLA) can be implemented using a single ADDIE, as shown in Figure 1.

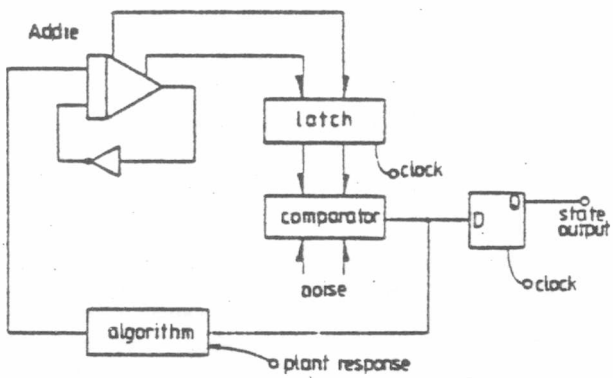


Figure 1 Two-State ADDIE SLA

The contents of the ADDIE counter represent state probability $p_1(n)$, while $p_2(n)$ is simply the complement.

An essential feature of the operation of the automaton is the updating of state probabilities in accordance with the environment or plant response. This is achieved in the ADDIE SLA by loading $p_1(n)$ from the ADDIE to a latch, and performing digital-to-stochastic conversion. The resulting stochastic pulse train is then transformed via the algorithm circuitry to an updated state probability $p_1(n+1)$. The ADDIE then reaches an estimate of $p_1(n+1)$, and, after a suitable settling time, the next cycle can commence. A flip-flop on the comparator output represents the present state occu-

pied, and state trajectories can be observed by filtering the output, or by direct digital/analogue conversion of the ADDIE contents.

The operating sequence for the ADDIE SLA is as follows. The initial load operation sets up the requisite value of 0.5 (i.e. "one - all zeros") in the counter, so that the output of the comparator is a stochastic sequence with an equal probability of 1's and 0's, representing random state selection at initial time t_0. At the first system clock pulse, this sequence is sampled and at the same time, the counter contents are copied into the latch. Then, when the clock pulse goes low, the punishment/reward signal resulting from the state of the D-type flip-flop is latched, and the ADDIE clock enabled, allowing the "learning period" to commence. During this time, the ADDIE converges to the new value of $p_1(1)$, which is then used as the basis for the next cycle.

The advantage of this design is that, since no locking-on problems can occur, it is possible to implement the more suitable ε-optimal schemes using the established method of algorithm circuit design, based on stochastic computing techniques, described previously[6].

3. ALGORITHM CIRCUIT DESIGN

As mentioned earlier, the ADDIE stochastic learning automaton enables several of the reinforcement schemes described previously[5] to be implemented.

The linear reward/penalty scheme L_{R-P} may be expressed in two-state form as:

(i) Reward : (action a_1)
$p_2(n+1) = \alpha\, p_2(n)$
$p_1(n+1) = 1 - \alpha\, p_2(n)$

(ii) Penalty : (action a_1)
$p_1(n+1) = \beta\, p_1(n)$
$p_2(n+1) = 1 - \beta\, p_1(n)$
where $0 < \alpha < 1$ and $0 < \beta < 1$

Similar expressions hold for action a_2. Algorithm circuitry was designed using a form of "truth-table":

$P_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$
0	1	0	$\alpha\, p_1(n)$
0	1	1	$1-\beta\, p_2(n)$
1	0	0	$1-\alpha\, p_2(n)$
1	0	1	$\beta\, p_1(n)$

The design can obviously be extended to cover the ε-optimal linear reward-reward (L_{R-R}) and reward-inaction (L_{R-I}) schemes. The L_{R-I} scheme is particularly simple to accommodate, since the only modification required is to set the factor $\beta = 1$. The L_{R-R} scheme, in which the penalty is replaced by a lesser reward, is given below in two-state form:

(i) Non-penalty : (on action a_1)
$p_1(n+1) = 1 - \alpha\, p_2(n)$
$p_2(n+1) = \alpha\, p_2(n)$

(ii) Penalty : (on action a_1)
$p_1(n+1) = 1 - \beta\, p_2(n)$
$p_2(n+1) = \beta\, p_2(n)$
where $0 < \beta < \alpha < 1$

As before a truth-table is constructed to enable the algorithm to be translated into a circuit design:

$p_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$
0	1	0	$\alpha\, p_1(n)$
0	1	1	$\beta\, p_1(n)$
1	0	0	$1 - \alpha\, p_2(n)$
1	0	1	$1 - \beta\, p_2(n)$

By comparison with the truth-table for the L_{R-P} scheme, it is evident that the L_{R-P} circuit can be converted to an L_{R-R} circuit simply by reversing the "$\beta\, p_1(n)$" and "$1 - \beta\, p_2(n)$" connections. The circuit arrangements for these linear schemes are summarised in Figure 2.

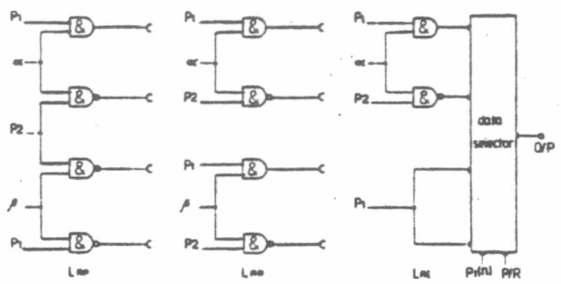


Figure 2 Algorithm Circuits for Linear Schemes.

Although the best of the linear schemes, the L_{R-I} scheme, has been widely reported as most suitable for many applications, considerable study has been made of nonlinear updating schemes[5]. These tend to show faster initital convergence rates, and indeed one reason for the emphasis put on these schemes is to obtain optimum covergence times, especially when they are incorporated in hybrid schemes.

The simplest of the nonlinear schemes is that denoted as $N^1{}_{R-P}$, which has "square-law" nonlinearity. It has been shown that this scheme is conditionally optimal, providing optimal convergence if $c_1 < \frac{1}{2} < c_2$ and expedient otherwise (c_i represents penalty probabilities). This scheme, again for the two-state case, if given below:

(i) Non-penalty (on action a_1)
$p_1(n+1) = p_1(n) + \alpha\, p_1(n)[1 - p_1(n)]$
$p_2(n+1) = p_2(n) - \alpha\, p_1(n)[1 - p_1(n)]$

(ii) Penalty (on action a_1)
$p_1(n+1) = p_1(n) - \beta\, p_1(n)[1 - p_1(n)]$
$p_2(n+1) = p_2 + \beta\, p_1(n) \quad [1 - p_1(n)]$
where $0 < \alpha\ ,\ \beta < 1$

The truth-table for the above scheme is as follows:

$p_1(n)$	$p_2(n)$	P/R	$p_1(n+1)$
0	1	0	$(1 - \alpha p_2)\, p_1$
0	1	1	$1-(1 - \beta p_1)\, p_2$
1	0	0	$1-(1 - \alpha p_1)\, p_2$
1	0	1	$(1 - \beta p_2)\, p_1$

The circuit diagram for this scheme is shown in Figure 3.

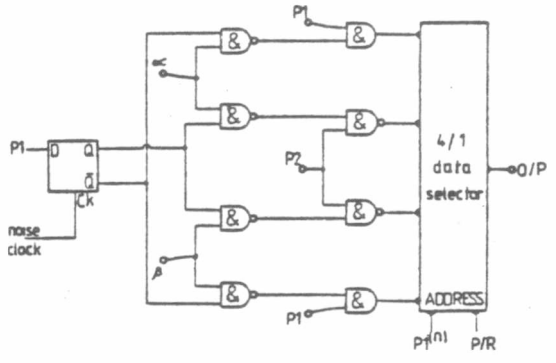


Figure 3 Algorithm Circuit for $N^{(1)}_{R-P}$ Scheme

The configuration is essentially similar to that of the L_{RP} circuit, except that in the case of the N^1_{RP} scheme, the constants α and β are replaced with terms of the form

$(1 - \alpha p_i)$ and $(1 - \beta p_i)$.

4. EXPERIMENTAL RESULTS

The ADDIE stochastic learning automaton was tested using a storage oscilloscope to observe state trajectories directly. Learning curves obtained with the L_{R-P} scheme using a main system cycle clock of 100 kHz are shown in Figure 4.

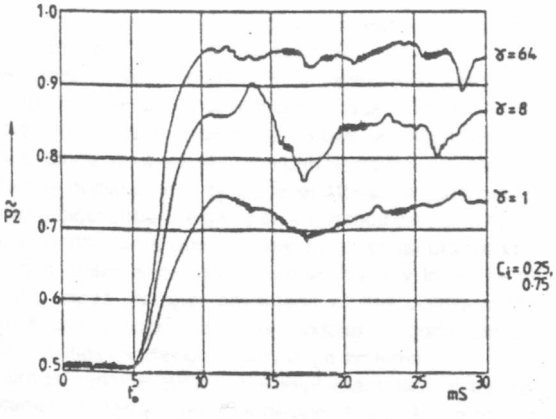


Figure 4 Learning Curves for L_{R-P} Scheme

The curves clearly show how the degree of expediency increases as the reward-penalty "Y" is increased from 1 to 64 ($\gamma = \frac{1-\alpha}{1-\beta}$), starting from an initial condition of random state selection at time t_0, i.e. $p_1(0) = p_2(0) = 0.5$.

Typical learning curves obtained from the ε-optimal L_{R-I} scheme with $\alpha = 0.75$ are illustrated in Figure 5.

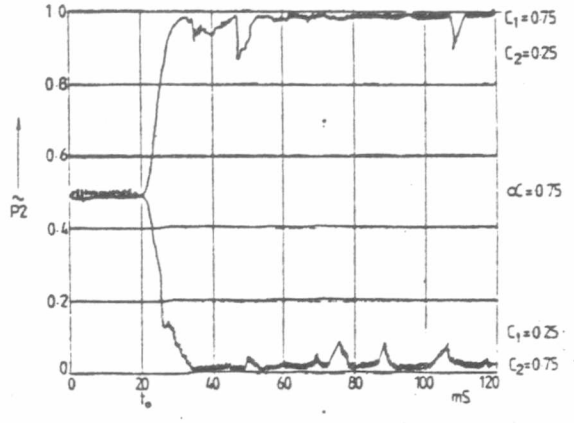


Figure 5 Learning Curves for L_{R-I} Scheme

With this scheme, the automaton exhibits virtually full convergence to $p_1 = 1$ or $p_2 = 1$, according to the relative values of the penalty probabilities. Figure 6 shows learning curves for the L_{R-R} scheme, again illustrating the higher degree of expediency obtained with high γ.

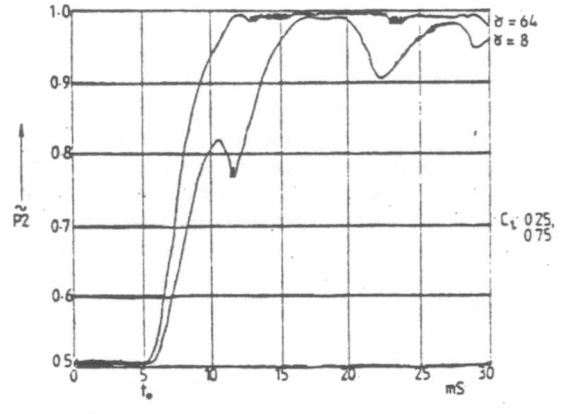


Figure 6 Learning Curves for L_{R-R} Scheme

These results are consistent with previous simulation studies,[5][9] although the actual convergence times are masked by the response time of the output ADDIE (stochastic-to-analogue converter). This element has a restricted bandwidth in order to provide a suitable compromise between learning time and variance in the steady-state output. Nevertheless, the results presented here indicate learning times of less than 10 ms, pointing to the feasibility of practical, on-line operation.

In the case of the N^1_{R-P} scheme, the conditional optimality property was investigated by feeding the penalty

probability signals via a switching circuit to compare the degree of convergence in "optimal" and "expedient" conditions. One penalty probability was fixed at 0.125 and the other was switched from 0.56 to 0.44, as indicated by the switching waveform in Figure 7.

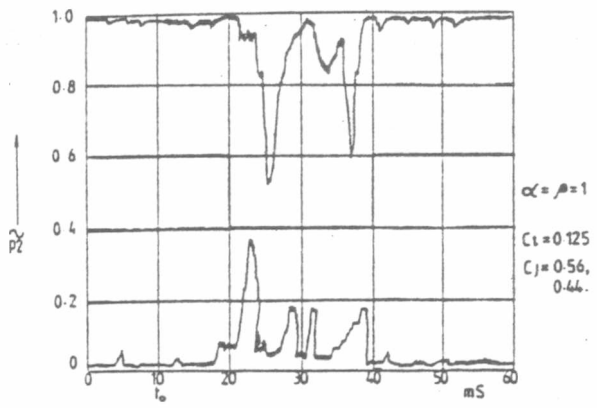


Figure 7 Learning Curves for N_{R-P} Scheme

This shows that with c_i = {0.125 , 0.56}, the state trajectories yield optimal convergence, whereas with c_i = {0.125 , 0.44}, the system degrades, as expected, to expedient behaviour.

5. 128-STATE SYSTEM

Having verified that the two-state ADDIE stochastic learning automaton had satisfactory learning characteristics, attention was subsequently focussed on the development of a much larger, practically useful system, and the attendant problem of minimising the amount of circuitry required while preserving high operating speeds.

The solution to this problem is to subdivide the state space and perform the random search between the states via a set of levels in a hierarchical structure. It has been suggested previously (4) that a multilevel approach can be used to overcome this problem of high dimensionality, and the application of simple two-level structures has been considered(10).

The requirement for a memory capability in the automaton structure to establish a priority of state order during the learning period is embodied in the 128-state hierarchical system described here. A two-state SLA "cell" is time-shared between each location in a seven-level "decision tree", and interfaced with a random-access memory (RAM) to store intermediate probability values, as illustrated in Figure 8. Since these seven two-state decisions are equivalent to one decision in a single-level 128-state automaton, the hierarchical structure gives an enormous saving in hardware, and although it does involve more serial processing operations, there is not an excessive penalty in terms of operating speed. Another advantage of this configuration is that it can be made entirely modular in construction, which simplifies the design of very large systems.

A further consideration is the implementation of the reinforcement scheme. With this design, it is possible to use the same two-state algorithm circuits as before, also time-shared between each decision level.

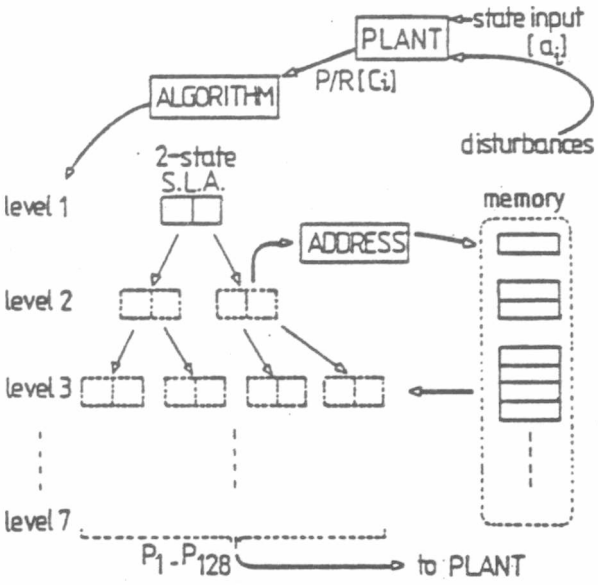


Figure 8 128-State Hierarchical System

6. SYSTEM OPERATION

The two-state ADDIE SLA which forms the "cell" of the structure is essentially similar to that described earlier. The main difference is that now memory interface circuits are required, since the SLA no longer acts in a continuous, self-contained cycle, but operates instead in a time-shared mode within the "decision tree". The full operating cycle for this system is as follows:

Initially, each memory location is set at 0.5, so that each decision has an equal probability of occurring; consequently the probability of selecting any one state is $(0.5)^7$, i.e. 1/128. At each main sampling clock pulse, the cell output will be either 1 or 0, and this "decision bit" is stored in a 7-bit "state latch". After a search through the decision tree, the state latch contents will therefore define uniquely one of 128 states, and also represent the output to the plant. The second half of the cycle consists of retracing the same path through the decision tree, this time applying a reinforcement scheme to alter the decision probabilities represented by the ADDIE in accordance with the plant response. Updating at each level is controlled by the punishment/reward signal (P/R) and the corresponding decision bit.

The next full cycle can then commence, with revised decision probabilities and consequently a revised set of total state probabilities. As the learning process evolves, the decision path leading to the optimum state will be reinforced until, in the limit, assuming the use of an optimal scheme, all the decision probabilities along that path tend to unity.

The memory requirements are determined by the number of levels in the system. For each decision, the binary word in the ADDIE represents, together with its implicit complement, the decision probabilities for each "direction". The first decision level thus requires one word of storage, the second level requires two words, the third, four words, and so on. Therefore, the 128-state system requires a total RAM allocation of 127 × n bits (n = 8 — 12). A central feature of the hierarchical system operation is the memory address

procedure. Since each decision is a binary one, the design of the control circuitry is greatly simplified, and the address code is derived from the state latch, which is effectively a small "scratch-pad" memory tracking the decision path.

7. RESULTS

The operating principal of the hierarchical system is illustrated in Figure 9.

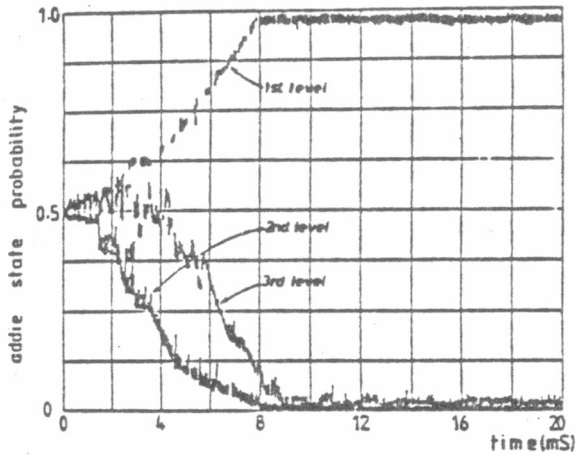


Figure 9 Learning Behaviour of 8-State System

This shows the learning behaviour of a three-level, 8-state system, with three simultaneous "learning curves", obtained via a D/A converter from the time-shared ADDIE, representing convergence to state 100_2 (i.e. state 4). The learning period is typically three times that expected of a single two-state system.

Static and dynamic optimisation experiments were carried out on the full 128-state system, using a simple simulated "plant" in which one selected state carried a low penalty probability (0.25), and all other states a higher one (0.875). For these results, the system master clock was set at 2.5 MHz, a limit governed by the RAM access time, and the system state was sampled every millisecond, with D/A conversion of the state latch contents providing a "map" of the state output.

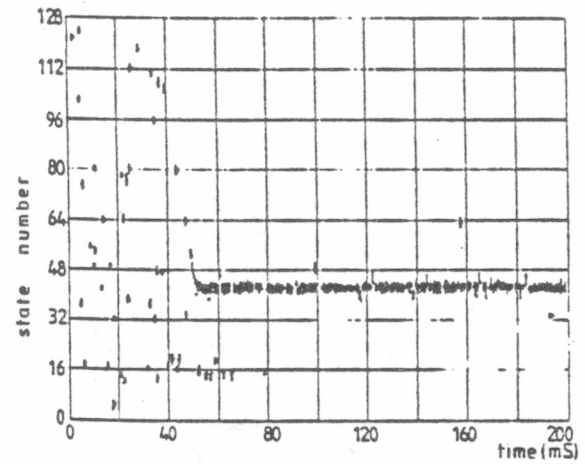


Figure 10 Convergence of Hierarchical System with Steady State Environment

Figure 10 shows a typical state output map of convergence to state 41, using an L_{RI} reinforcement scheme with $a = 0.25$. The significant feature of this result is that the indicated learning time is less than 50 ms. Due to the variance of the 8-bit ADDIE used here, there is a non-zero probability of incorrect decisions at any level, after the initial learning period, and this results in the sporadic "jumps" to other states seen in the output.

Figure 11, the optimum state is switched periodically from state 58 to state 106, and the automaton output is seen to follow the switching waveform within 50 ms.

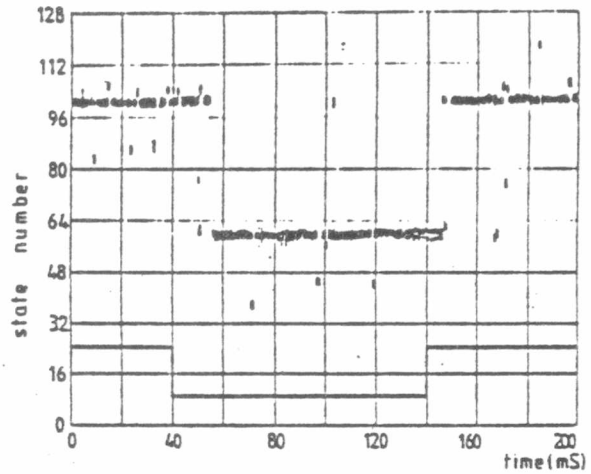


Figure 11 Hierarchical System with Switched Environment

The transition interval is characterised by random selection between "old" and "new" states, until the new state finally predominates.

8. CONCLUSIONS

The application of digital stochastic computing methods to the hardware synthesis of learning automata has been described. The use of the ADDIE as a basic building block for the system has enabled excellent learning characteristics to be achieved with algorithm circuitry capable of implementing a useful range of reinforcement schemes, and learning times of the order of milliseconds are demonstrated.

The design of large state order systems has been accomplished by incorporating the two-state ADDIE SLA in a hierarchical structure. The results obtained from the 128-state system described here have verified that fast, economical learning behaviour is possible under conditions of static and dynamic optimisation, and represent an important development in the implementation of on-line learning controllers. Work is currently in progress on the practical application of learning automata to stochastic systems with multimodal performance criteria.

ACKNOWLEDGEMENT

The authors wish to gratefully acknowledge the support of a U K Science Research Council grant.

REFERENCES

1. **Gaines, B. R.:** "Stochastic Computing" AFIPS SJCC, 1967, 30, pp 149-156.

2. **Miller, A. J. and Mars, P.:** "Theory and Design of a Digital Stochastic Computer Random Number Generator" Trans IMACS, 1977, 19, 3, pp 198-216.

3. **Miller, A. J. and Mars, P.:** "Optimal Estimation of Digital Stochastic Sequences". Int J of Systems Science, 1977, 8, 6, pp 683-696.

4. **Narendra, K. S. and Thathachar M. A. L.:** "Learning Automata — A Survey" IEEE Trans Syst, Man and Cybern, 1974, SMC-4, 4, pp 323-334.

5. **Viswanathan R. and Narendra K. S.:** "Expedient and Optimal Variable Structure Stochastic Automata" Becton Centre, Yale University, 1970, Tech Rep CT-31.

6. **Neville R. G., Nicol C. R. and Mars, P.:** "Synthesis of Stochastic Learning Automata"; Electronics Letters, 1978, 14, pp 206-208.

7. **Miller, A. J.M., Brown, A. W. and Mars, P.:** "Adaptive Digital Logic Circuits for Digital Stochastic Computers" electronics Letters, 1973, 9, pp 500-502.

8. **Witten, I. J.:** "Stochastic Implementation of Learning Controllers" IEE Colloquium on Parallel Digital Computing Methods, 1976, Digest No. 1976/30, Paper 6.

9. **Viswanathan R. and Narendra, K. S.:** "Simulation Studies of Stochastic Automata Models" Becton Centre, Yale University, 19171, Tech Rep CT-45.

10. **Narendra, K. S. and Viswanathan R.:** "A Two-Level System of Stochastic Automata for Periodic Random Environments" IEEE Trans, 1972, SMC-2, pp 258-289.

MARCO SOMALVICO

Marco Somalvico was born in Como, Italy, on October 10, 1941. He received the Dr. Ing. degree in electronic engineering from the Politecnico di Milano, Milan, Italy, in 1965.

From 1965, with a Postdoctoral Fellowship, he joined the Computer Science Laboratory of the Istituto di Elettrotecnica ed Elettronica of the Politecnico di Milano, Italy, where he worked as a Research and Teaching Associate in the area of computer-aided circuit analysis and design. In 1969 with a Fulbright-Hays Postdoctoral Fellowship, he joined, as a Visiting Scholar, the Artificial Intelligence Project of the Computer Science Department, Stanford University, Stanford, CA. In 1971 he was named Assistant Professor of Electrical Engineering and in 1974 Associate Professor of Computer Science at the Politecnico di Milano. His present research interests lie in artificial intelligence, mathematical theory of computation, and robotics. Dr. Somalvico is a member of the Association for Computing Machinery. He was awarded the Premium Somaini as the Best Undergraduate in the schools of Como in 1960, and the Gold Medal of the Politecnico di Milano as the Best Graduate in electronic engineering in 1965.

M.A.L. THATHACHAR

M.A.L. Thathachar was born in Mysore City, India in 1939. He received the B.E. degree in Electrical Engineering from the University of Mysore in 1959 and the M.E. and Ph.D. degrees from the Indian Institute of Science, Bangalore in 1961 and 1968 respectively. He was on the faculty of the Indian Institute of Technology, Madras during 1961-64. Since 1964, he has been with the Indian Institute of Science, Bangalore, where he is currently Professor in the Department of Electrical Engineering. He visited Yale University, New Haven, CT in 1973 as well as in 1978-79. His major research interests are in Learning Systems and Stability Theory.

OOMMEN B. JOHN

Oommen B. John was born on September 9, 1953 in the Nilgiris in India. He received his B. Tech. degree from the Indian Institute of Science in Bangalore, India, both in electrical engineering. He received his M.S. from Purdue University in Lafayette, IN in May 1979 where he is currently working on his Ph.D. in electrical engineering and is majoring in Computers.

His research interests include automata theory and formal languages, stochastic automata and some applications of artificial intelligence techniques.

R. G. NEVILLE

Richard G. Neville was born in Bishops Stortford, England on April 21, 1954. He received the B.S. degree with First Class Honours in Electronic and Electrical Engineering from Robert Gordon's Institute of Technology, Aberdeen, Scotland, in 1976. He is presently at Science Research Council Research Fellow in the School of Electronic and Electrical Engineering, Robert Gordon's Institute of Technology. His current research interests are mainly in the areas of of digital stochastic computing and learning systems.

PHIL MARS

Phil Mars was born in Liverpool, England, on July 30, 1941. He received the B.S. degree with honours in Electrical Engineering from Durham University, England, in 1964 and the M.S. and Ph.D. DEGREES FROM THE Universities of Newcastle upon Tyne and Liverpool, in 1969 and 1971 respectively. He is currently Head of the School of Electronic and Electrical Engineering at Robert Gordon's Institute of Technology, Aberdeen, Scotland. His research interests are in the area of stochastic automata and special purpose digital systems design. He is the author of numerous publications in the fields of digital and solid state electronics and is an industrial consultant in these fields. Dr. Mars is a member of the United Kingdom Council of National Academic Awards Electronic and Electrical Engineering Board.

ADAPTIVE CONTROL OF MULTIMODAL STOCHASTIC SYSTEMS USING LEARNING AUTOMATA

R G Neville and P Mars

School of Electronic and Electrical Engineering, R G I T, Aberdeen AB9 1FR, Scotland

In many process control problems, the characteristics of the process are fully known, and a complete mathematical description of the process and of the corresponding control strategy is possible. However, a large number of situations arise where uncertainties are present, either due to an incomplete mathematical model of the process, or due to operation in a random environment. Where the probabilistic nature of these uncertainties is known, stochastic control theory can be applied, but in the case of higher order uncertainties where the probabilistic characteristics cannot be easily ascertained, it is only possible to gain sufficient knowledge of the process by 'on-line" observation. Herein lies the application area for stochastic learning automata. Essentially stochastic learning automata provide a novel and attractive mode of solving a large class of problems involving uncertainties of a high order.

Many problems of adaptive control, pattern recognition, filtering and identification can, under proper assumptions, be regarded as parameter optimisation problems. A learning automaton can be fruitfully applied to solve such problems, especially under noisy conditions when the apriori information is small. For such problems, unlike stochastic approximation methods, the learning automaton has the desired flexibility not to get locked on to a local optimum, and this fact makes the automata approach particularly applicable to multimodal performance criteria systems.

The paper will consider experimental results obtained for the real-time control of noisy multimodal systems using recently developed hardware hierarchical structure learning automata. The approach will be shown to permit the effective economic realisation of high-speed controllers for real-time system control.

INTRODUCTION

One of the principal application areas for stochastic computing research is the implementation of adaptive control systems using stochastic learning automata (SLA) structures (1). A learning automaton is ideally suited to the problem of parameter optimisation of a noisy multimodal system, since the inherent principle of random search avoids the effect of locking-on to local optima unavoidable with normal gradient methods, for example. The automaton, by means of a suitable interface, interacts with the environment in a manner analogous to a conventional feedback control system to evolve a 'suitable' final structure (figure 1).

Through a combination of earlier work in hardware stochastic computing systems, and extensive simulation studies of learning automaton behaviour (2), it has been possible to synthesise practical learning systems capable of on-line operation. Hardware designs for 2-state systems have been described (3,4,5) which verified that suitably fast learning behaviour was possible.

In order to implement large-scale systems, a hierarchical structure automaton was developed, using the 2-state SLA in a time-shared mode. This system, reported previously (6), has been tested using simple simulated plant responses. The work described here also details applications to more practical examples of systems with multidimensional, multimodal performance criteria, and the adaptive control of a real, small-scale process.

The Automaton

The concept of "automaton" in the context of the work reported here can be defined as follows. An automaton is essentially a device which is capable of receiving input signals or responses at discrete intervals of time and determining one of a finite number of output actions by means of some intermediate decision-making process acting on its internal structure or state.

The various elements of this broad definition can be stated more precisely as follows :

(i) The input to the automaton, denoted x(n), is an element of the set

$$X = \{x_1, x_2, \text{---} x_k\}$$

where k may be finite or infinite.

(ii) The state of the automaton, denoted $\phi(n)$, is an element of the set

$$\Phi = \{\phi_1, \phi_2, \text{---}\phi_s\} \; ; \; s \text{ is finite}$$

(iii) The output action of the automaton, denoted a(n) is an element of the set

$$A = \{a_1, a_2\text{---}a_r\} \; ; \; r \text{ is also finite}$$

In addition, two functional relationships exist which relate the above variables and complete the definition of the automaton

(iv) The transition function F relates the current state and input at stage n to the next state at stage n+1

$$\text{i.e. } \phi(n+1) = F\ [\phi(n), x(n)]$$

(v) The output function G relates the current state of the automaton to the resulting output action at stage n.

$$\text{i.e. } a(n) = G\ [\phi(n)]$$

The automaton is therefore defined

mathematically by a quintuple $\{X, \phi, A, F, G\}$. The functions F and G may be deterministic or stochastic mappings. If F and G are both deterministic, the automaton is denoted a "deterministic automaton", in which case the next state and output action are uniquely defined for a given current state and input. The work to be described here, however, concentrates on the stochastic automaton, in which F or G, or both, are stochastic functions. In this case, there are only probabilities associated with the succession of states and output actions.

A representation of the structure of the automaton can be given in terms of the total state probabilities :

$$\pi_i(n) = Pr\ \{\phi(n) = \phi_i\}$$

or the total action probabilities :

$$p_i(n) = Pr\ \{a(n) = a_i\}$$

Again, to preserve probability measure, it follows that

$$\sum_i \pi_i = \sum_i p_i = 1.$$

It is frequently the case that G denotes a one-to-one mapping between states and actions, in which case $\pi(n)$ and $p(n)$ are equivalent.

The Environment

The environment encompasses all the external factors which influence the structure or behaviour of the automaton. It accepts the output actions of the automaton as inputs, and produces output responses which are in turn fed back to the automaton. The environment is therefore characterised by three sets of variables forming the triple $\{A, C, X\}$ where A and X are respectively the action and input sets of the automaton as defined above, and C is a set of "penalty probabilities",

$$C = \{c_1, c_2 \text{ ---- } c_r\}.$$

In practice, it is convenient to concentrate on the particular automaton-environment configuration in which the set X has just two elements, i.e. $X = [0, 1]$. By convention $x = 0$ denotes a favourable response or "reward" and $x = 1$ denotes an unfavourable response or "penalty".

Each element c_i of C is associated with an element a_i from the action set A, and is defined as follows :

$$c_i = Pr\ \{x(n) = 1 \ / \ a(n) = a_i\}\ .$$

The Concept of Learning

The concept of "learning" is applied here to describe the behaviour of a variable structure automaton operating in an environment as defined above. A learning automaton is capable of determining the success of each action in eliciting a reward from the environment, and, in the specific case of a variable structure device, ordering its structure so as to increase the probability of selecting a more successful action.

Clearly, if the c_i were already known, the strategy of the automaton would be simply to select the action a_m corresponding to the minimum penalty probability c_m. The elements c_i of C are therefore assumed to be unknown at all times.

Reinforcement Scheme

A variable structure automaton modifies its policy for selecting output actions by the application of a reinforcement scheme, denoted

$$f^n\ \{p_1(n) \text{ ---- } p_r(n)\}, \text{ such that}$$

$$p_i(n+1) = p_i(n) + f_i^n\ \{\ \},\ i=1 \text{ ---- } r$$

Again, to preserve probability measure, all such schemes must ensure that

$$\sum_{i=1}^{r} f_i^n\ \{\ \} = 0$$

An an example, one of the most widely investigated, and indeed earliest proposed schemes, the linear reward-penalty scheme, denoted L_{R-P}, will now be described. The algorithm, stated in total probability form, is as follows :

(a) Reward (action a_i)

$$p_{j\neq i}(n+1) = \alpha\ p_j(n)$$

$$p_i(n+1) = 1 - \sum_{j\neq i} p_j(n+1)$$

(b) Penalty (action a_i)

$$p_i(n+1) = \beta\ p_i(n)$$

$$p_{j\neq i}(n+1) = p_j(n) + \{\frac{1-\beta}{r-1}\}\ p_i(n)$$

where $0 < \alpha, \beta < 1$

If the penalty coefficient $\beta = 1$, the scheme is transformed to linear reward-inaction, denoted L_{R-I}.

Review of Hierarchical System

The hierarchical SLA evolved as a means of enabling a practical large-scale automaton to be constructed which would be capable of high-speed operation with the minimum of hardware. The approach adopted was to time-share a single 2-state SLA in a tree-structure, as shown in figure 2. The random access memory, which stores the intermediate decision probabilities, fulfils the requirement for a memory in the automaton to establish the priority of state order during the learning period. Any one state or action probability is given by the product of the decision probabilities along the appropriate path through the tree. This configuration does of necessity involve more serial processing operations, but the savings in hardware are felt to far outweigh the speed penalty. Another advantage is that a modular construction greatly simplifies the design requirements of much larger systems. The reinforcement algorithm circuit can retain the standard (4-term) format used previously with 2-state systems, and is also time-shared at each level.

Using these principles, a hardware system with up to 128 states was constructed, based on the availability of a suitable

commercial random access memory (RAM) with 128 bytes of storage. The memory requirements are determined by the number of levels in the system. The number of decision probabilities to be remembered at level p is 2^{p-1}; therefore, an r-level system (2^r states) requires a total RAM allocation of just $(2^r - 1)$ bytes.

Initial Experiments

Initial optimisation experiments were carried out using an elementary simulated plant, in which one selected action, in this case number 41, carries a low penalty probability, $c_l = 0.25$, and all others a higher one, $c_h = 0.875$. Figure 3 shows the resulting learning behaviour presented in the form of an 'output map', derived by D/A conversion of the automaton output latch contents. The approximate length of one system iteration is 50 μs, so that the indicated learning time of 50 ms corresponds to some 1 000 iterations.

Because of variance inherent in the hardware estimation of probability, there is always a slight chance of incorrect decisions, at any level, beyond the initial learning period. This results in the sporadic occurrence of incorrect output actions which can be seen on the output map.

Application to Multimodal System

In order to simulate a multimodal environment, a rather more sophisticated plant was required. It was decided to use as an example a P.I. function (7) which has a distinct global optimum, a local optimum and a saddle-point. The function is given by :

$$I(x,y) = (1 + 8x - 7x^2 + \frac{7}{3}x^3 - \frac{1}{4}x^4)y^2 e^{-y}$$

In order to present this P.I. surface to the SLA, the function was evaluated at discrete points and programmed into a read-only memory, thus storing the c_i values as 8-bit numbers, each addressed by the appropriate action output from the SLA. The presence of noise on the surface was simply effected by interposing a full adder fed with noise derived from the central pseudo-random binary noise source. In addition, a non-stationary environment was conveniently accommodated by storing alternative versions of the P.I. in different sectors of the PROM.

A representative result of the optimisation of this system by the hierarchical structure automaton is depicted in figure 4. This shows the output map for the case of a non-stationary environment (switched after 2 000 system iterations) using an L_{R-P} automaton with the following parameters :

$$\alpha = 0.5 \ , \ \beta = 0.992.$$

During the first learning phase, the system converges to actions 19 and 20, both near the optimum. An interim adjustment period follows the switch, culminating in convergence to the 'new' optimum of action 100. This result clearly illustrates the ability of the SLA to track a non-stationary environment without excessive delay.

Process Control

The application of the SLA to adaptive control was demonstrated with the aid of a small-scale thermal process. This consists of a centrifugal fan with an adjustable inlet orifice which feeds air past a grid heater and through an outlet pipe. The exhaust temperature is measured by a miniature bead thermistor and compared with the set-point to derive an error signal which in turn is used to control the heater power output.

A two-term (P + I) controller was developed, in which the proportional band and integral time coefficients were each adjustable over 8 steps (33% - 175% and 0.5 s - 8 s respectively). A 64-action SLA was therefore used to 'tune' the controller, as shown in the block diagram (figure 5). A convenient measure of performance for this system was the Integral of Squared Error or ISE criterion resulting from the application of a step input. The ISE was transformed into reward/penalty information for the automaton, while the direct error signal provided the input to the controller.

The performance of this adaptive controller is illustrated by the ISE plot of figure 6(a). After the initial adjustment phase, the system was disrupted by opening up the blower aperture. It can be seen that the automaton reacts to this disturbance and recovers control quite rapidly. Figure 6(b) shows a similar result in which noise was superimposed on the set-point, and is characterised by slightly longer adjustment time and higher levels of steady-state error.

Conclusions

These results clearly demonstrate the power of the automaton approach to the optimisation of non-stationary multimodal systems, irrespective of surface contours or of the presence of noise in the system. In particular, it should be stressed that at no time did convergence to the local optimum occur, indicating that the SLA has purely altitude sensitivity over a P.I. surface, as opposed to the gradient sensitivity of conventional methods.

Process controllers find widespread applications in manufacturing industries, and there is a need for frequent, often unpredictable adjustments to be made on-line. The success of the automaton in achieving adaptive control of the thermal process described above is felt to be of considerable importance, representing a fruitful area of application for practical hardware systems.

REFERENCES

1 Narendra, K.S., and Thathachar, M. A. L, "Learning Automata - A Survey", IEEE Trans. Syst., Man. and Cybern., 1974, SMC-4, 4, pp 323 - 334

2 Viswanathan, R., and Narendra, K. S., "Simulation Studies of Stochastic Automata Models", Technical Report, CT-45, 1971, Becton Centre, Yale University.

3 Neville, R.G., Nicol, C.R., and Mars, P, "Synthesis of Stochastic Learning Automata", Electronics Letters, 1978, 14, pp 206 - 208

4 Neville, R.G., Nicol, C.R., and Mars P.,

"Design of Stochastic Learning Automata using Adaptive Digital Logic Elements", Electronics Letters, 1978, 14, pp 324 - 326

5 Neville, R. G., Nicol, C.R., and Mars, P. "Design of Non-Linear Stochastic Learning Automata", Electronics Letters, 1978, 14, pp 396 - 397

6 Neville, R. G., and Mars, P., "Hardware Design for a Hierarchical Structure Stochastic Learning Automaton", J. of Cybern. and Inf. Sci., 1979, 2, 1, pp 30 - 35

7 Asai, K., and Kitajima, S., "A Method for Optimising Control of Multimodal Systems using Fuzzy Automata", Information Sciences, 1971, 3, pp 343 - 353.

Acknowledgement

One of the authors (R G Neville) wishes to acknowledge the support of an S.R.C. research assistantship.

The work reported here has been funded by S.R.C. grant GR/A53956.

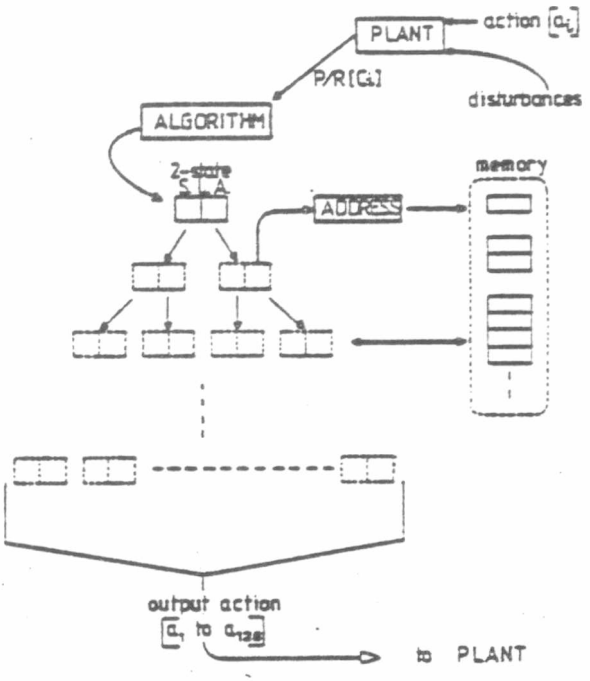


Figure 2 128-State Hierarchical Structure Automaton

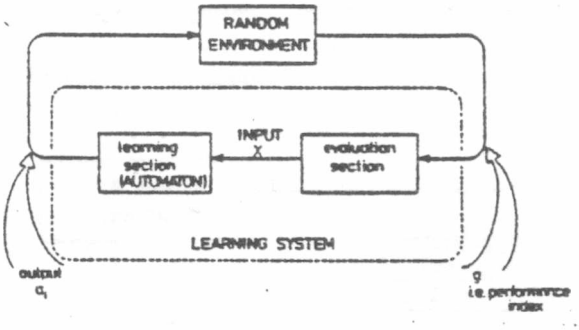


Figure 1 Automaton - Environment Configuration

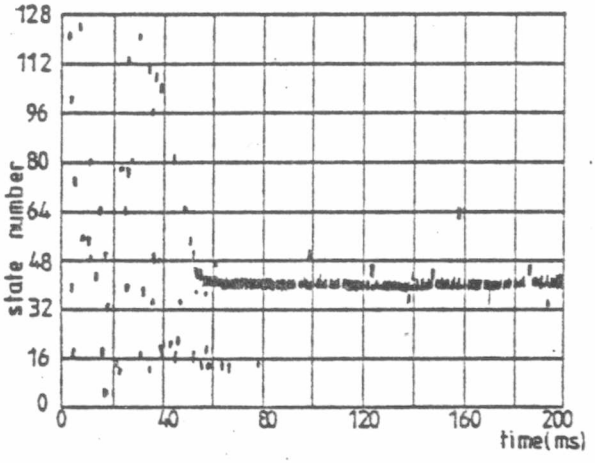


Figure 3 Output Map (Action 41)

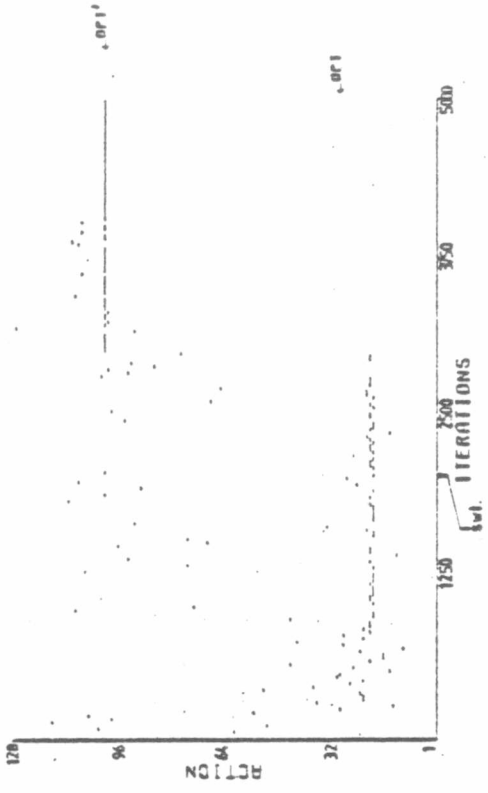


Figure 4 Output Map - Switched Environment

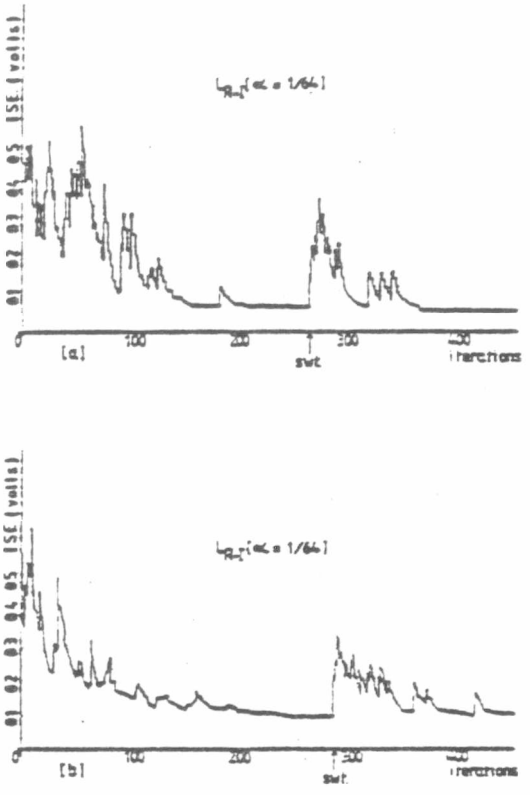


Figure 6 Error Curves from the Learning Controller

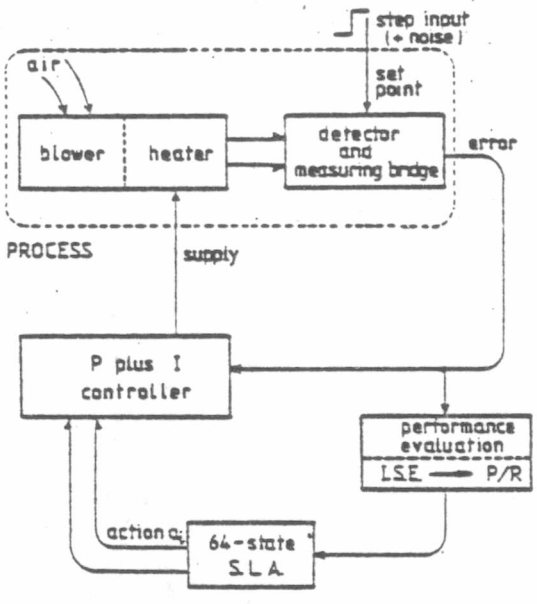


Figure 5 Thermal Process with SLA Control