Some aspects of the design construction and applications of a digital stochastic computer.

BAXTER, T.

1975

SOME ASPECTS OF THE DESIGN CONSTRUCTION AND APPLICATIONS OF A DIGITAL STOCHASTIC COMPUTER

Thomas Baxter

September, 1975

Thesis submitted for the Degree of M.Phil.
at Robert Gordon's Institute of Technology

CONTENTS

-----oooOooo-----

SUMMARY

The object of this project has been the construction and some aspects of design of a digital stochastic computer, in particular the patching system, initial conditions of integrators and a study of a stochastic to analogue output interface.

In the latter stages of the project attention was turned to focus on the design and construction of a special purpose stochastic simulator, namely the Markov Chain and Random Walk simulator.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr P Mars for his guidance and encouragement throughout the course of this project. Also I am indebted to my colleagues Tony Miller and Angus Brown for their valuable contributions and enlightening discussions. Finally I would like to thank Miss Pam Swanson for her superb work in the typing of this thesis.

INTRODUCTION

Within the field of computer control there exists an increasing number of problems which cannot easily be solved using conventional computing systems. These problems arise for example in the real time control of large multivariable systems such as chemical processes, aircraft control systems, etc. Attempts to overcome these problems have led to the development of various arrangements of hybrid computers in an effort to obtain the advantages of the analogue and digital computers in one machine. Unfortunately the majority of these hybrid systems also incorporate the disadvantages of the two conventional computers.

The digital computer although very fast and accurate, performs all computations sequentially and in applications involving, for example, the solution of differential equations where numerical iterative techniques must be employed, the time taken to obtain a solution can be in the order of minutes or even longer. This may be acceptable in a process which requires correction in this time scale but in fast processes where correction is essential within seconds, this is unacceptable. One advantage of the use of a digital computer in control is that the size of the computer required does not increase significantly as the size of the process to be controlled.

Conversely the analogue computer, because of its parallel operation, can provide a solution to a differential equation almost instantaneously. However the complexity of an analogue computer increases greatly as the size of the process increases.

The /

The ideal hybrid computer should combine the advantages of the analogue and digital computers without incorporating any of their disadvantages.

A stochastic computer, although not a hybrid computer, does combine the advantages of analogue and digital computers. The stochastic computer has been defined as 'an analogue computer using digital techniques.' It is defined in this way because it operates in a parallel mode, which makes it very fast, and uses conventional digital circuitry which makes it very competitive as regards cost. One disadvantage of the stochastic computer is that, as in the case of the analogue computer, the complexity increases significantly as the problem size. However because of the nature of the circuitry of the stochastic computer this disadvantage can be eliminated by using LSI techniques for constructing the computer thus making it small and inexpensive.

The stochastic computer uses probability as its analogue quantity in the same way as the analogue computer uses voltage.

Because probability cannot be estimated instantaneously there is a delay in obtaining a solution to any problem. The delay is proportional to the accuracy required, ie, the more accurate a solution is to be then the longer will be the time taken to obtain this solution. Therefore a balance between accuracy and speed must be struck.

Nevertheless in applications involving complex problems where speed and accuracy are not critical, the stochastic computer is the ideal solution.

CHAPTER 1

FUNDAMENTALS OF STOCHASTIC COMPUTING

1.1 Representation of Variables[1,2,5,6]

Stochastic computers are similar to analogue computers in that both are parallel processing machines. The analogue computer employs voltage as its analogue quantity ie, a given range of voltage (±1 machine unit, usually ±100 volts) represents the range of a normalised variable. In contrast the stochastic computer employs probability as an analogue quantity, the probability of a sampled pulse being high, ie, ON. This probability cannot be estimated from one sample therefore a number of samples must be made. As is well known from elementary probability theory the accuracy of the estimation is in fact proportional to the number of samples. The value of this probability is defined as the ratio of the number of ON pulses recorded to the total number of samples. Thus the estimated probability must lie in the range 0 to 1.

To use probability as an analogue quantity it is necessary to scale variables within this range which is similar to the scaling procedure in an analogue computer.

There are three principal mapping procedures and these are:

(a) single line unipolar representation (SLUR)
(b) double line bipolar representation (DLBR)
(c) single line bipolar representation (SLBR)

Each of these procedures will now be described in detail.

1.2 Single Line Unipolar Representation

This is the simplest of the three mapping procedures, simple scaling being all that is required.

A quantity E can be represented by a probability

$$p(ON) \quad /$$

$$p(ON) = E/V \quad \text{where} \quad 0 \leq E \leq V.$$

At the maximum value of E, ie, E = V then p(ON) = 1 and this is represented by a continuously ON logic level. Conversely the minimum value of E, ie, E = 0 gives p(ON) = 0 which is represented by a continuously OFF logic level.

Figure 1.1(a) shows an example to demonstrate how an intermediate value can be represented. The diagram illustrates a sample of twenty pulses, seven of which are ON. An approximation of the probability may be obtained by taking the number of ON pulses as seven and the number of samples as twenty. This gives

$$p(ON) = \frac{E}{V} = \frac{7}{20} = 0.35$$

$$E = 0.35V$$

If for example, E were to represent velocity with a maximum value (V) of 13 m/s then the above stochastic sequence would represent

$$E = 0.35 \times 13 = 4.55 \text{ m/s}$$

It is important to note that the line which carries this stochastic sequence is always associated with the same variable as is also the case with an analogue computer.

With this mapping either positive or negative quantities can be represented, ie, unipolar. If both positive and negative quantities are to be represented then a bi-polar mapping must be used.

1.3 /

1.3 Double Line Bipolar Representation

With this representation two lines are used, one to represent positive quantities and one to represent negative quantities. The line with the positive weighting is called the UP line and the line with the negative weighting is called the DOWN line.

To represent a quantity E where $-V \leqslant E \leqslant V$ then

$$p(\text{UP=ON}) - p(\text{DOWN=ON}) = \frac{E}{V} \quad .$$

The maximum positive quantity is represented by the UP line being continuously ON and the DOWN line continuously OFF. Conversely the maximum negative quantity is represented by the UP line always OFF and the DOWN line always ON. For intermediate quantities there will be a stochastic sequence on both lines.

1.4 Single Line Bipolar Representation

A bipolar variable can be represented by a single line if the following mapping is used,

$$p(\text{ON}) = \tfrac{1}{2} + \tfrac{1}{2}\frac{E}{V} \quad \text{where} \quad -V \leqslant E \leqslant V$$

The maximum positive quantity occurs at E = V giving p(ON) = 1 and the maximum negative quantity occurs at E = -V giving p(ON) = 0. To represent E = 0 then p(ON) = 0.5.

1.5 Choice of Representation[5]

Some advantages and disadvantages of the above three mapping procedures will now be considered so as to enable a choice of method to be made.

With SLUR scaling of variables is easily performed. However, this method will be ruled out because only unipolar quantities can be represented and this was considered unsuitable for a general purpose stochastic computer.

The /

The second method, DLBR has the disadvantage of requiring more hardware than any of the single line mappings. Two lines are used in this method, one for positive quantities and the other for negative quantities. For each element, there is in effect two unipolar elements, one for negative and one for positive quantities. Thus compared to a single line mapping the hardware requirements and hence system cost, are greater for DLBR. A second hardware disadvantage of DLBR is that of the patching system requirements. With a single line representation each stochastic input and output consists of only one line. Therefore to patch two elements together it is only necessary to have one connecting path. However with DLBR there are two lines (UP and DOWN) associated with one stochastic input or output and hence the patching of two elements would require two connecting paths. In fact the patching system for DLBR will be two single line patching systems in parallel, ie, the hardware involved in the patch panel will be doubled.

One of the advantages of DLBR is that of variance. After a large number of samples the estimated probability of a sequence will lie within a range of values, ie, distribution curve. The narrower the range then the lower the variance or the greater the accuracy of the estimation. It can be shown[5] that for SLBR, maximum variance occurs when p(ON) = 0.5 whereas with DLBR this value of p(ON) coincides with minimum variance.

Because of the increased hardware requirements a DLBR was rejected as a possible method of mapping.

The method of SLBR was therefore adopted for DISCO and in the following descriptions of computing elements only this mapping procedure will be considered.

1.6 Input Interface[1,5]

Before considering individual operating elements we must consider how a weighted stochastic sequence is generated, ie, we must examine the input interface of the stochastic computer.

Consider Figure 1.2. A 12-bit binary input is presented to a digital comparator and is compared with a 12 bit digital random number. The random number generator must have a uniform distribution, ie, there must be an equal probability of generating any one of the 2^{12} possible numbers. This is achieved by using 12 noise lines, each one having p(ON) = 0.5. The method of generating these noise lines is demonstrated in the following section.

An ON pulse is delivered by the comparator output at each clock pulse if the digital number is greater than the random number. If the binary number is low then there will be fewer ON pulses delivered than would be the case if the binary number is high. For example, if every bit of N_b (the binary number) is low then no ON pulses will be delivered, ie, p(ON) = 0 and E = -V. Conversely if every bit of N_b is high then N_b is always greater than or equal to N_r (the random number) and a continuous stream of ON pulses is delivered giving a stochastic sequence with p(ON) = 1, ie, E = V. If the most significant bit of N_b is high and all other bits are low then there is an even chance of $N_b > N_r$ and the resulting stochastic sequence will on average have an equal number of ON and OFF pulses, ie, p(ON) = 0.5 and E = 0.

It is therefore possible to generate a stochastic sequence of any probability from a 12 bit digital number. This is called a stochastic comparator and it serves as an ideal interface between digital and stochastic computers.

1.7 /

1.7 Generation of Digital Noise[1,5]

The preferred method of providing digital noise is to use maximal length sequences (m-sequences) which are generated by a pseudo-random binary sequence (PRBS) generator. This generator is called pseudo-random because the sequence will repeat itself at periodic intervals although any two m-sequences will pass all necessary tests[5] for statistical independence, ie, randomness.

Figure 1.3 illustrates the method of generating digital noise. The exclusive-OR gate which generates the feedback signal performs modulo-2 addition with the carry neglected. If the stages of the shift register which feed the exclusive-OR gate are carefully selected then the register will cycle through each non-zero state in an apparently random fashion. This means that for an N bit shift register each m-sequence produced has a period of 2^N-1 clock intervals.

A single m-sequence can be delayed by x clock pulses where $1 \leqslant x \leqslant 2^N-1$, to give 2^N-2 additional statistically independent m-sequences. The first N m-sequences are taken directly from the shift register outputs and are used as noise lines.

The individual computing elements are now discussed in theory, only the single line bipolar representation being considered.

1.8 Inversion[1]

The inversion operation is performed very simply by a NOT gate.

Consider Figure 1.4(a). The stochastic sequence at A is representative of E where $-V \leqslant E \leqslant V$ whereas the sequence /

sequence at B is the inverted form of that of A. Thus these two sequences are mutually exclusive and their probabilities must sum to unity. Thus

$$p(B) = 1 - p(A)$$

but

$$p(A) = \tfrac{1}{2} + \tfrac{1}{2}.\frac{E}{V}$$

thus

$$p(B) = 1 - (\tfrac{1}{2}+\tfrac{1}{2}.\frac{E}{V}) = \tfrac{1}{2} - \tfrac{1}{2}\frac{E}{V} = \tfrac{1}{2} + \tfrac{1}{2}\frac{E^*}{V}$$

The stochastic sequence at B is representative of the quantity E* and it is clearly seen that

$$E^* = -E$$

1.9 Multiplication[1]

In single line bipolar representation multiplication is achieved by an exclusive-NOR gate.

From Figure 1.4(b) it is seen that the output of the exclusive-NOR gate is

$$C = A.B + \bar{A}.\bar{B}$$

The stochastic sequences at A and B are given by $p(A) = \tfrac{1}{2}+\tfrac{1}{2}.\frac{E}{V}$ and $p(B) = \tfrac{1}{2}+\tfrac{1}{2}.\frac{E'}{V}$ and the output stochastic sequence represents the quantity E* such that $p(C) = \tfrac{1}{2}+\tfrac{1}{2}.\frac{E^*}{V}$.

But

$$p(C) = p(A).p(B) + [1 - p(A)][1 - p(B)]$$

$$p(C) = [\tfrac{1}{2} + \tfrac{1}{2}\frac{E}{V}][\tfrac{1}{2} + \tfrac{1}{2}\frac{E'}{V}] + [\tfrac{1}{2} - \tfrac{1}{2}\frac{E}{V}][\tfrac{1}{2} - \tfrac{1}{2}\frac{E'}{V}]$$

This reduces to

$$p(C) = \tfrac{1}{2} + \tfrac{1}{2}\frac{E\,E'}{V^2}$$

$$E^* = \frac{E\,E'}{V}$$

which /

which indicates the normalised multiplication of E by E'.

1.10 Squaring[5]

Basically a squarer is the multiplier described above. If however the stochastic sequence representing the quantity to be squared is applied simultaneously to both inputs of a multiplier then the output is continually ON. Therefore it is necessary to use two statistically independent stochastic sequences, both representing the same quantity. It can be shown[5] that if a stochastic sequence is delayed by more than twelve clock periods then the two sequences are statistically independent. Clearly the delayed sequence will have the same weighting as the original sequence which eliminates the need to generate two separate sequences. Thus the original stochastic sequence representing the quantity to be squared is multiplied by the delayed sequence resulting in normalised squaring, ie,

$$E^* = \frac{E^2}{V} \quad .$$

Figure 1.4(c) shows the basic circuit diagram for a squarer.

The delay is achieved by employing a shift register clocked at the same rate as the stochastic sequence.

1.11 Summation[1,5]

At first sight it would appear that summation of two stochastic sequences can easily be achieved using an OR gate. However, over a given number of samples, the sum of two stochastic sequences should have on average the same number of ON pulses as the sum of the ON pulses of both the sequences. This introduces two problems.

Firstly, /

Firstly an OR gate makes no allowance for two coincident high level inputs and each time this occurs an ON pulse will be lost introducing an error into the output. Secondly if two stochastic sequences each have a probability of greater than 0.5 then the sum of these sequences is a stochastic sequence with a probability of greater than unity which by definition is impossible. Therefore normalised addition must be performed.

Consider Figure 1.4(d). The configuration is such that only one of gates A and B is enabled at any one time. If the internally generated noise (m-sequence) has a probability of 0.5 then A and B have an equal chance of being enabled resulting in the output of gate A being E/2 and that of B being E'/2. At no time can the stochastic sequences at the outputs of gates A and B both be ON simultaneously thus permitting an OR gate to accurately sum the two sequences. Thus output C is given as

$$p(C) = \tfrac{1}{2}p(A) + \tfrac{1}{2}p(B)$$

$$= \tfrac{1}{2}(\tfrac{1}{2} + \tfrac{1}{2}\frac{E}{V}) + \tfrac{1}{2}(\tfrac{1}{2} + \tfrac{1}{2}\frac{E'}{V})$$

$$= \tfrac{1}{2} + \tfrac{1}{2}\,\frac{\tfrac{1}{2}(E + E')}{V} = \tfrac{1}{2} + \tfrac{1}{2}\,\frac{E^*}{V}$$

$$E^* = \tfrac{1}{2}(E + E')$$

ie, normalised addition is performed.

1.12 Integration[1,5]

The basic integrator in a stochastic computer is a digital counter. In a bipolar representation the counter used must be reversible since both positive and negative quantities occur.

Figure 1.5 shows the block diagram of an integrator. For each clock pulse the counter will be incremented by /

by one if the UP line is ON and the DOWN line is OFF. Conversely, if the UP line is OFF and the DOWN line is ON the counter will be decremented by one.

The contents of the counter after the nth clock pulse is denoted C(n) such that

$$C(n) = \frac{\text{state of counter}}{N}$$

where N = number of states.

The average expected change of the counter after the nth clock pulse is

$$\delta C(n) = \frac{p(UP) - p(DOWN)}{N}$$

Over a period of q clock pulses

$$C(q) - C(0) = \sum_{n=0}^{q} \delta C(n)$$

where C(0) is the initial condition of the counter.

$$C(q) = C(0) + \sum_{n=0}^{q} \delta C(n)$$

Rewriting in integral form

$$C(q) = C(0) + \int_{0}^{q} \delta C(n)\,dn$$

$$= C(0) + \int_{0}^{q} \frac{p(UP) - p(DOWN)}{N}\,dn$$

$$= C(0) + \frac{1}{N} \int_{0}^{q} (p(UP) - p(DOWN))\,dn \quad \text{---- (1.1)}$$

To convert this to an integration with respect to time we proceed as follows:

AT $n = 0 \quad t = 0$

$n = q \quad t = q\tau_1$

where

where τ_1 = period of clock pulse

$$n = t/\tau_1$$

$$dn = \frac{1}{\tau_1} dt$$

Substituting in equation (1.1) we obtain

$$C(t) = C(O) + \frac{1}{N\tau_1} \int_O^t \{p(UP) - p(DOWN)\}dt \qquad \text{---- (1.2)}$$

The real time gain of the integrator is seen to be dependent on the counter size and clock frequency.

From Figure 1.5 we see that,

$$p(DOWN) = 1 - p(UP).$$

p(UP) is in fact the weighting of the stochastic sequence to be integrated.

Therefore

$$p(UP) - p(DOWN) = 2p(UP) - 1$$

$$= 2(\tfrac{1}{2} + \tfrac{1}{2}\frac{E}{V}) - 1$$

$$= \frac{E}{V}$$

Substituting in equation (1.2) we have

$$C(t) = C(O) + \frac{1}{N\tau_1} \int_O^t \frac{E}{V} dt \qquad \text{---- (1.3)}$$

The solution C(t) is in binary form and must be converted to a stochastic sequence so as to facilitate further computation. This is accomplished by comparing the constants of the counter with a 12 bit random digital number as previously explained in section 1.5.

By /

By definition C(t) must lie in the range 0 to 1, therefore the stochastic sequence delivered by the comparator will have a probability equal to C(t). ie,

$$\tfrac{1}{2} + \tfrac{1}{2}\frac{E^*(t)}{V} = \tfrac{1}{2} + \tfrac{1}{2}\frac{E^*(0)}{V} + \frac{1}{N\tau_1}\int_0^t \frac{E(t)}{V}\,dt$$

$$E^*(t) = E^*(0) + \frac{2}{N\tau_1}\int_0^t E\,dt \qquad \text{---- (1.4)}$$

This is the equation which describes the operation of an integrator.

1.13 Integrator With Summing Inputs[5]

If the UP and DOWN lines of an integrator are both the same, either ON or OFF at the incidence of a clock pulse then the counter state will be held constant. This is the HOLD mode and by utilising this condition it is possible to obtain summation of the inputs.

Figure 1.6 shows the scheme for a summing integrator. It is seen that

$$p(UP) = p(A)p(B)$$

and

$$p(DOWN) = [1 - p(A)][1 - p(B)]$$

$$p(UP) - p(DOWN) = p(A) + p(B) - 1$$

$$= (\tfrac{1}{2} + \tfrac{1}{2}\frac{E_1}{V}) + (\tfrac{1}{2} + \tfrac{1}{2}\frac{E_2}{V}) - 1$$

$$= \frac{E_1 + E_2}{2V}$$

Substituting in equation (1.2) we have

$$C(t) = C(0) + \frac{1}{N\tau_1}\int_0^t \frac{E_1 + E_2}{2V}\,dt$$

giving /

giving

$$E^*(t) = E^*(O) + \frac{1}{N\tau_1} \int_O^t (E_1(t) + E_2(t))dt \qquad \text{---- (1.5)}$$

This is the integrator configuration adopted for DISCO.

1.14 Output Interface[1,5]

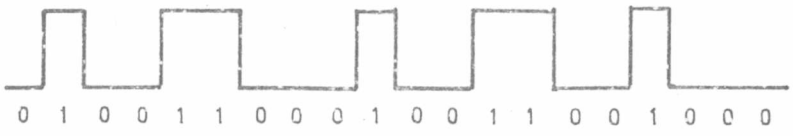
The solution to any computation executed by the stochastic computer will be represented by a stochastic sequence (except in the case of an integrator counter containing the desired solution). It is necessary to translate this stochastic sequence into a binary number or an analogue voltage so as to enable the solution to be stored or displayed in some form.

Conversion of a stochastic sequence to a binary number is achieved by an integrator with negative feedback, the nature of which may be probabilistic or deterministic. This is discussed in Chapter 5.

A stochastic to digital convertor is called an ADDIE (ADaptive DIgital Element), one form of which, called a noise ADDIE is shown in Figure 1.7(a). The quantity represented by the stochastic sequence is exponentially averaged by the ADDIE giving the averaged solution in binary form as C(t).

A stochastic to analogue (S/A) convertor is shown as a simple R-C low pass filter in Figure 1.7(b). The voltage at the output of the filter is proportional to the weighting of the stochastic sequence being smoothed.

Both the noise ADDIE and the S/A convertor are analysed in detail in Chapter 5.



(a) p (ON) = $^{7}/_{20}$ = 0 35 for the above diagram

(b) Symbolic representation of Single Line Bipolar Mapping

FIGURE 1.1 Representation of variables

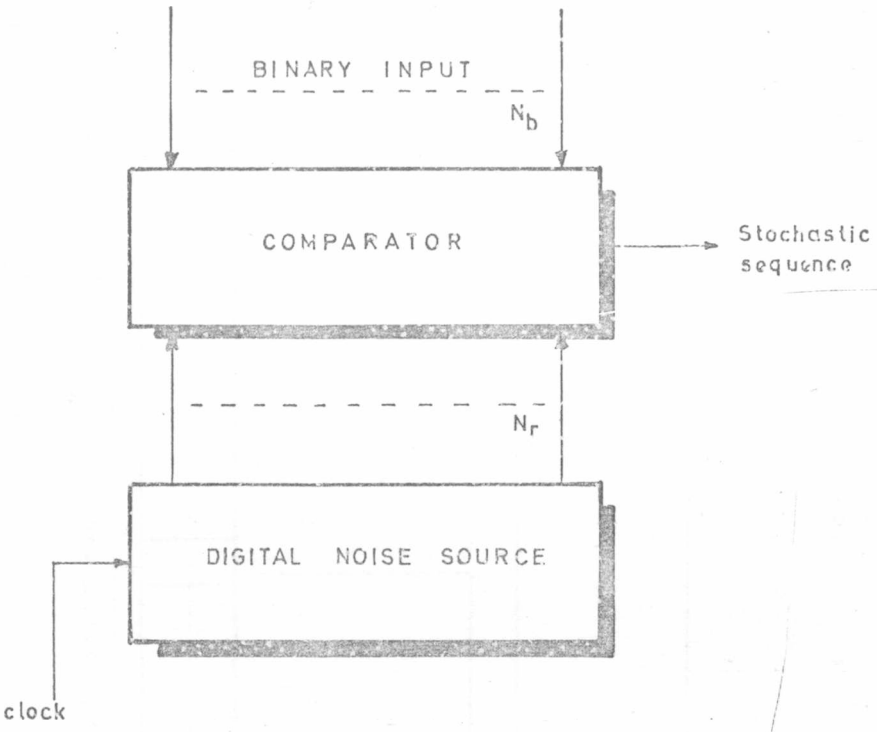


FIGURE 1.2 Stochastic Comparator

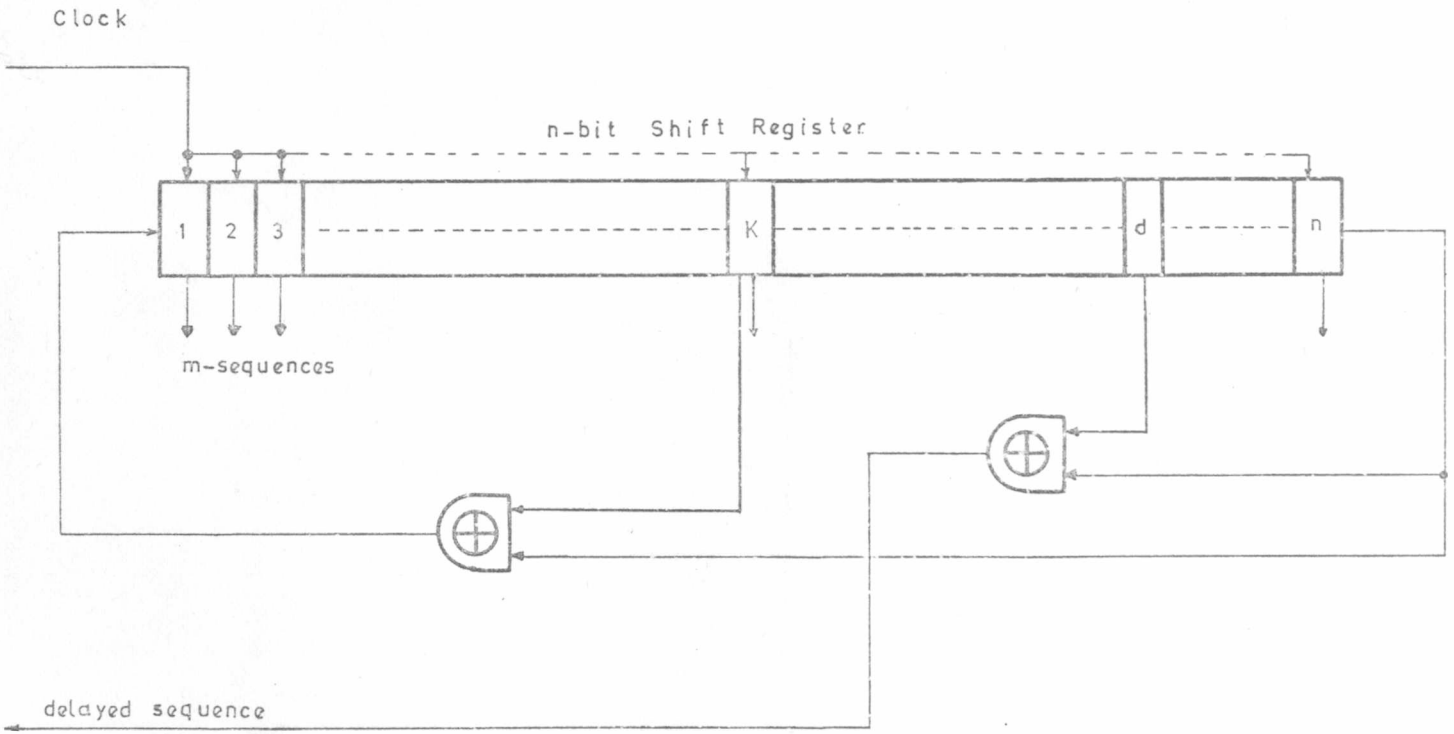
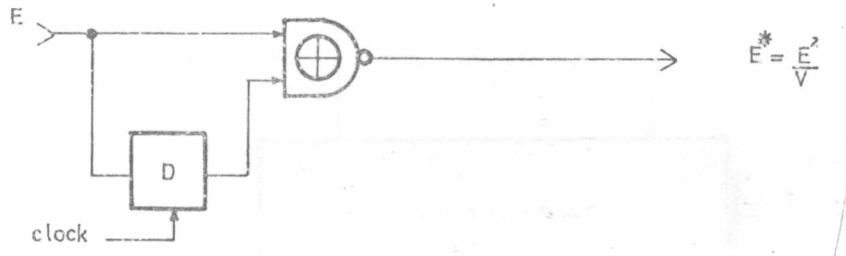


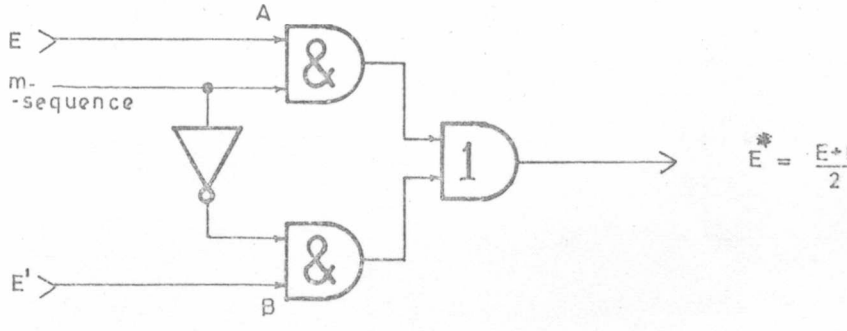
FIGURE 1.3 Generation of Digital Noise

(a) Invertor

(b) Multiplier



(c) Squarer



(d) Summer

FIGURE 1.4 Stochastic Computing Elements

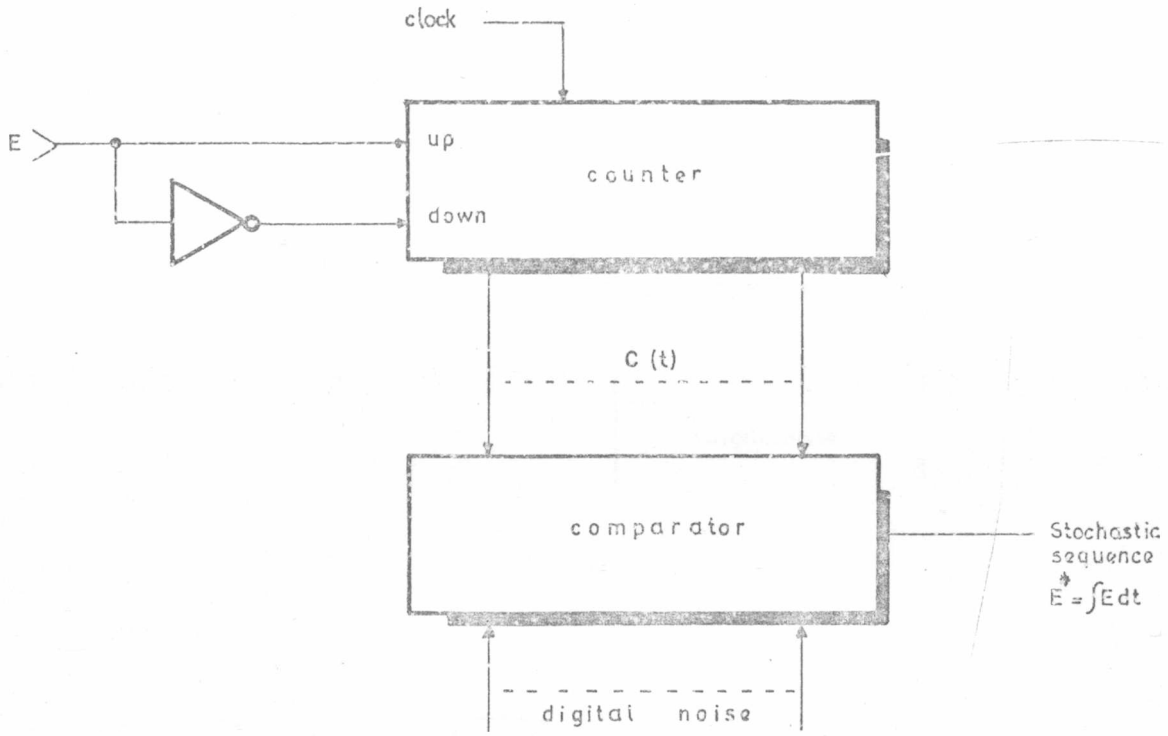
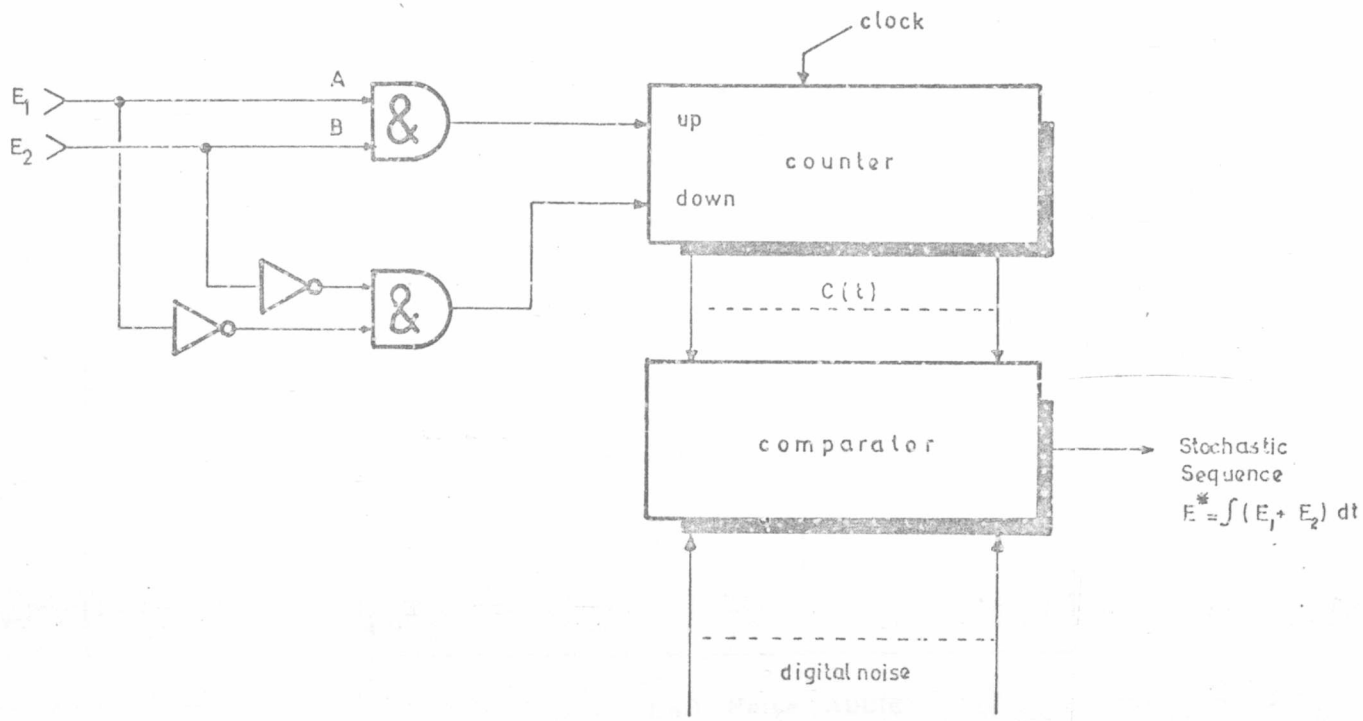
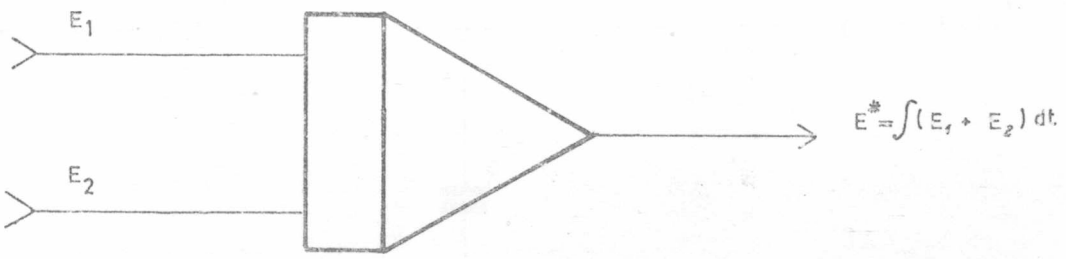


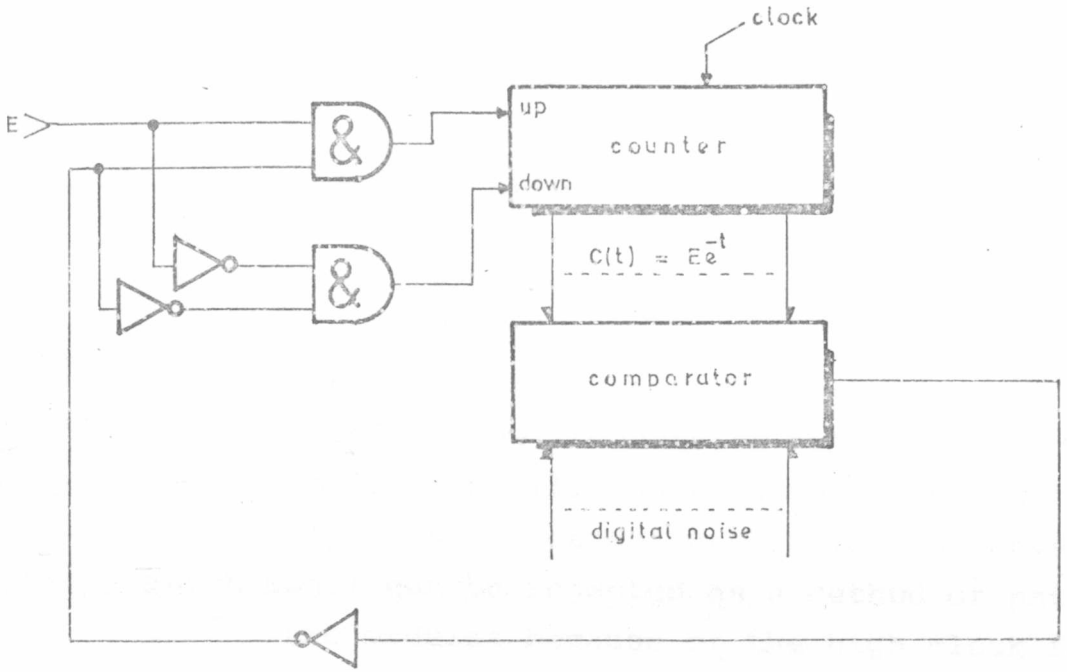
FIGURE 1.5 Integrator



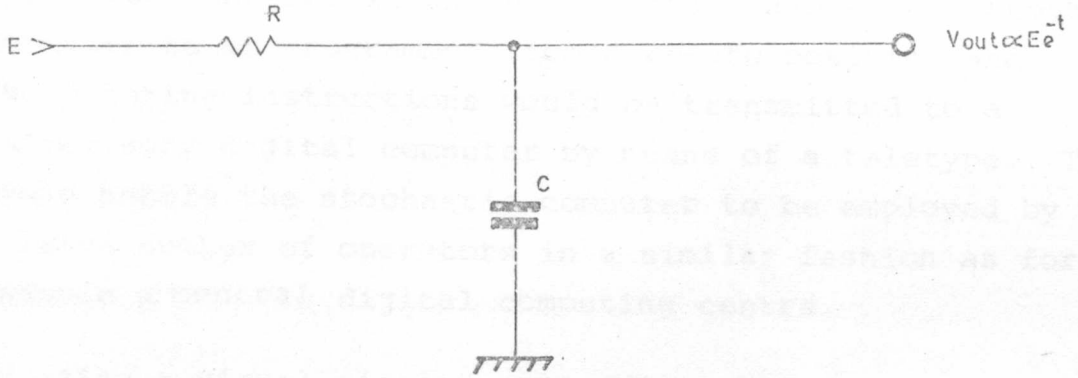


SYMBOL

FIGURE 1.6 Summing Integrator



(a) Noise ADDIE



(b) S/A convertor

FIGURE 1.7 Output interface

CHAPTER 2

DESIGN OF PATCH PANEL

2.1 Necessity For An Automatic Patch Panel

The individual operating elements have been described in Chapter 1. It is clear that these elements must be interconnected in some fashion so as to implement a program as is the case in an analogue computer. In this case the interconnecting of computing elements, or patching, is achieved by employing a patch panel in which all patch wires used will be interlaced with each other. Also there will be a small capacitance between the two plates in each patch socket. This may be neglected at low frequencies but at high frequencies these two factors will cause false switching because of crosstalk and stray capacitance. Thus a conventional patch panel can be rejected as a method of patching the stochastic modules because of the high clock frequency involved.

Some of the other advantages of using an automatic patch panel will now be described.

A computer controlled patching system enables the stochastic computer to be programmed from a remote position and the patching instructions would be transmitted to a supervisory digital computer by means of a teletype. This would enable the stochastic computer to be employed by a large number of operators in a similar fashion as for example a central digital computing centre.

By using a visual display unit (VDU) the programmer will simply have to type the numbers of the inputs and outputs to be connected. This ease of operation will result in a very short programming time compared to physically connecting the required inputs and outputs as would be done with an analogue computer.

2.2. /

2.2 Definition of Input and Output Nodes

Before considering the design of the patch panel some basic definitions will be considered.

The first definition is that of an output node. Each element of the stochastic computer has a stochastic output (except of course the output interface which has either a digital or analogue output) and the line from this output is defined as an output node.

The second definition is that of an input node. Again, every element (except the input interface) has at least one stochastic input. The line to this input is defined as an input node and there is an input node for each input.

Each input and output node will be uniquely described in Chapter 4.

To avoid confusion it must be borne in mind that the inputs to the patch panel are output nodes and the outputs of the patch panel are input nodes, ie, the terms input and output nodes refer to the computing elements and not to the patch panel.

2.3 Specification For Patching System

To specify the requirements for the patch panel the number of computing elements to be catered for must be given.

The number of each type of element to be used is given in Table 2.1 and from this it can be seen that the patching system must cater for 64 output nodes and 96 input nodes.

In fact this table has subsequently been invalidated by the modular arrangement adopted. (See Chapter 4). Nevertheless Table 2.1 is included so as to demonstrate the original design philosophy.

2.4 /

2.4 Summary of Previous Systems

There have been a number of previous systems concerned with the problem of patching N output nodes to M input nodes. In all cases the problem has not been solved easily and the hardware involved is substantial.

Three of the most important systems will now be described in detail.

The first system[3] to be described was designed for application in a hybrid system using conventional analogue and digital computers where high speed operation was necessary to enable the system to be time shared.

The heart of the system is an NxM switching array and is shown in Figure 2.1. Each of the switches shown is an insulated gate field effect transistor (IGFET). IGFETs were chosen because of their high speed operation, low cost, availability in integrated circuit blocks as multiplexed arrays and finally because they were compatible with the analogue computers used. An IGFET has an ON resistance of 200 ohms and so operational amplifiers are used as buffers to reduce the output impedance.

There must be some form of memory incorporated within the patching system so as to enable the data pertaining to the state of each switch to be stored and the necessary switches to be closed. The memory capacity can be greatly reduced by considering the following. For each input there is one column of switches (see Figure 2.1) and at any one time only one of the switches in each column need be closed. If more than one were closed then more than one output node would be connected to the same input node which is impermissible. Thus the number of memory cells associated with the ith input node is given by

$$2^Z \quad /$$

$$2^Z = N + 1$$

where Z = Number of memory cells

$N + 1$ = N output nodes plus one for no connection

$$Z = \log_2(N + 1)$$

The total number of memory cells required for an NxM matrix is ZM, ie, a ZM bit shift register. To allow the correct switch to be closed, the information contained in the memory must be decoded.

Although this system was designed for the patching of analogue signals, digital signals can easily be catered for.

The main disadvantage of this system is that the IGFETs are incompatible with TTL and buffer and level shifting circuitry would have to be employed thus increasing cost, physical size and power supply requirements.

There are however important points to be noted from this system and these are:

(i) at any one time only one of the N output nodes is required to be connected to the ith input. This is very important and leads to a sizeable reduction in hardware and computer memory storage.

(ii) the number of memory cells required for an NxM matrix is ZM where $Z = \log_2(N+1)$.

As will be seen later this system is similar to that adopted for use in DISCO.

The next system[4] to be considered makes use of telephone switching theory. It is the objective of this system to reduce the number of switches in an automatic patch panel to a reasonable level.

The /

The ratio of input nodes to output nodes (outputs to inputs of the patch panel) is defined as the expansion factor E, ie,

$$E = \frac{M}{N}$$

where M = number of input nodes
N = number of output nodes.

For an MxN matrix the number of switches (S) required to connect any input to any output is

$$S = MN = EN^2$$

The number of switches required varies as the square of the number of inputs. This is called the 'N^2 problem' and it is desirable to reduce this factor of N^2 so as to decrease the hardware required for large values of N.

Figure 2.2 shows a three stage matrix with a trunk line between each input block and each middle block, and a trunk line between each middle block and each output block. Each block is a matrix which will connect any input to any output.

The number of input blocks is N/n where N is the number of output nodes and n is the number of inputs to each block. There will be M/m output blocks where M is the number of input nodes and m is the number of outputs from each block. The number of middle blocks is Y. Thus each input block has n inputs and Y outputs, ie, A contains nY switches. The total number of switches in all input blocks is thus $(\frac{N}{n})$ x nY = NY. Similarly the total number of switches contained in all output blocks is MY. There are N/n inputs to each middle block (one from each input block) and M/m outputs from each middle block (one to each output block). Therefore the total number of switches contained in all middle blocks is /

is

$$\frac{N}{n} \times \frac{M}{m} \times Y = \frac{NM}{nm} Y.$$

Thus the total number of switches required by a three stage matrix is

$$S = NY + MY + \frac{NM}{nm} Y$$

$$= Y(N + M + \frac{NM}{nm}) \qquad \text{---- (2.1)}$$

The problem now is to find the optimum values of Y, n and m for a given N and M.

To allow every input to each input block to be utilised then there must be at least the same amount of outputs as inputs, ie, $Y \geqslant n$. Similarly $Y \geqslant m$.

By inspection of equation (2.1) it is seen that $S \propto Y$ and $S \propto \frac{1}{nm}$. Therefore the conditions $Y \geqslant n$ and $Y \geqslant m$ suggest an optimum value of Y such that

$$Y = m = n \qquad \text{---- (2.2.)}$$

Substituting (2.2) in (2.1)

$$S = n(N + M + NM/n^2)$$

Elementary calculus gives the optimum value

$$n = (MN/(M + N))^{\frac{1}{2}}$$

and the optimum number of switches is therefore

$$S = 2(MN(M + N))^{\frac{1}{2}}$$

or rewriting using $E = \frac{M}{N}$

$$S = 2[E(1 + E)]^{\frac{1}{2}} N^{\frac{3}{2}} \qquad \text{---- (2.3)}$$

Thus /

Thus the number of switches varies as $N^{\frac{3}{2}}$ which is a significant improvement upon N^2.

From Table 2.1 it is seen that N = 64 and M = 96

$$E = 1.5 .$$

Substituting in equation (2.3)

$$S = 2076 \quad \text{and} \quad Y = m = n = 6.$$

ie, the number of input blocks is 11, the number of middle blocks is 6 and the number of output blocks is 16. This system actually caters for a value of N = 66. If N = 64 then Y, m and n have non-integer values which is meaningless.

Using one large matrix

$$S = 96 \times 66 = 6336$$

Clearly this three stage matrix has effected a considerable saving of some 66% of switches.

This analysis has ignored the possibility of fan out, ie, connecting an output node to more than one input node and in practice this will increase the value of S although there will still be considerable savings in hardware.

One significant problem arises from the use of this system. Once a trunk line has been used then another route must be found for the patch to be implemented. This may become a tedious process if a large scale problem is to be programmed.

However this system certainly offers advantages which could be utilised if the programming difficulty is overcome and could be constructed using digital techniques.

The /

The final system[7] to be described was built at Heriot-Watt University in Edinburgh. This system is virtually identical to the first system applied to DISCO although both were developed independently. Because of the similarity between the two systems the Heriot-Watt system will not be considered here but the difference between the two will be outlined at the end of the next section.

2.5 Initial Patching System

The first patching system to be designed was partially constructed and tested. One board was built, with the facility of having one output node patched to any one or more input nodes. Another 63 similar boards would have been required for a complete 64x96 patching system (one board for each output node).

The patching system must have the ability to accept and retain information pertaining to the nodes to be patched. This information must control some gating arrangement. These two factors suggest the use of a shift register as a means of entering and storing the patching data. Each bit or cell of the shift register will enable or inhibit one AND gate.

Consider Figure 2.3. The circuitry required for each output node consists of a 96 bit shift register with one AND gate associated with each bit of the shift register. Information is entered into the shift register in serial form from the supervising digital computer. The state of the jth bit of the shift register will determine whether the jth AND gate is enabled or inhibited. For example consider the shift register associated with the ith output node. If the jth bit of this register is logic '1' then the jth AND gate is enabled. Thus a patch is effected between the ith output node and the jth input node. Conversely, if the jth bit of the ith shift register contained a logic '0' then no patching will occur between the ith output node and jth input node. /

node. In this way the ith output node can be connected to any one or more input nodes. If the outputs of the jth AND gate in each of the 64 shift registers are ORed together then any output node can be connected to the jth input node. This is done for all 96 input nodes. To implement this 64 input OR gate, open collector NAND gates were used in place of AND gates. This enables a 'wired OR' arrangement to be used, Figure 2.4 showing that the 'wired OR' arrangement is equivalent to that described above using AND gates.

This patching system is in fact the realisation of the switching matrix of Figure 2.1 using logic gates as switches.

The 64 shift registers associated with the output nodes are connected in series, ie, the complete patch panel would have a 6144 (96x64) bit shift register. This shift register would be loaded in serial fashion by the digital computer from its memory. A PDP8/E computer is used as the supervising digital computer. The PDP8/E uses a 12 bit word and so 512 memory locations would be used. In fact this is one-sixteenth of the present memory capacity of 16k words. This is a significant amount considering 4k words are reserved for operation of the video display unit alone.

Table 2.2 gives the estimated cost of this patching system in terms of hardware only. It is seen that this system is very expensive. Because each of the circuits associated with each ouptut node must be constructed using one complete circuit board then 64 separate boards must be built. This would be very large and would in fact be three times the size of the stochastic computer itself.

As stated previously this system is virtually identical to the system constructed at Heriot-Watt University. The major difference between the two systems is that the patch panel constructed at Heriot-Watt University is /

is built using 8x8 patching modules, ie, each module has the ability to connect any of 8 output nodes to any of 8 input nodes. The only advantage offered by this modular approach is that the size of the patching system is flexible and can easily be expanded by adding more modules.

Owing to the large size and cost of this system further thought was given to the problem and an improved system was devised.

2.6 Final System Design

From section 2.5 the following points should be restated.

Firstly, for any given input node, one and only one output node may be patched at any one time. Secondly, the number of memory cells (bits) required to store the patching information for an NxM matrix is ZM where N = number of output nodes, M = number of input nodes, and $Z = \log_2(N+1)$.

Thus for each input node a circuit is required which will select one and only one of 64 output nodes for connection to the input node, ie, a 64 to 1 line data selector. One of these 64 to 1 line data selectors is required for each input node and so 96 data selectors are required. The first output node would be common to each of the 64 to 1 line data selectors as would be the second, third, etc. output nodes.

Figure 2.5 shows the scheme for one 64 to 1 line data selector.

To understand the operation of the circuit it is necessary to understand the operation of the SN74150 and SN74151 integrated circuits (IC).

The /

The SN74150 is a 16 to 1 line data selector. Upon application of a 4 bit binary code to the address inputs of the device, one and only one of 16 inputs is connected through to the output. Which one of the inputs to be connected is determined by the address code. For example if the code were 0000 then E_0 would be connected and if the code were 1111 then E_{15} would be connected.

The SN74151 is identical in operation to the SN74150 except that it is an 8 to 1 line data selector. By setting the most significant code bit to logic '0' the SN74151 operates as a 4 bit data selector.

Consider Figure 2.5. The 64 output nodes are connected to 4 SN74150s; output nodes 1 to 16 being assigned to IC1 etc. The address code, ABCD is common to IC1-4. For any code one from each group of 16 output nodes will be selected, eg, if the address code for the first stage were 0000 then output nodes 1, 17, 33 and 49 would be selected by the first stage. The second stage serves to select one of these 4 output nodes, eg, if the address code for the second stage were 00 then output node 1 would be selected and would effectively be patched to the jth input node. Thus by using a 6 bit code any one of the 64 output nodes may be patched to the jth output node.

A 6 bit serial in-parallel out shift register is used to store the 6 bit code. Each of the 96 6 bit shift registers are connected in serial form to form a 576 bit shift register which functions as the memory for the patching information for the entire patching system.

A number of advantages are offered by this system over the initial system, one of which is that it can be constructed using less ICs. This results in a patch panel of one third of the size and approximately one half of the cost of the initial system as may be verified by comparing Table 2.2 to Table 2.3 which shows the hardware costs of the final system. Secondly, only 48 12 bit memory locations are utilised which is a very small fraction

of the 16k words available in the PDP8/E. Finally the power supply requirements are half that of the initial system (an estimated 30A compared to 50A supply current). In fact the final system requires a supply current of 21A, the estimated figure being based on the IC manufacturers maximum ratings. The initial system has no advantages over the final system.

All control programs for the PDP8/E are listed in reference 6.

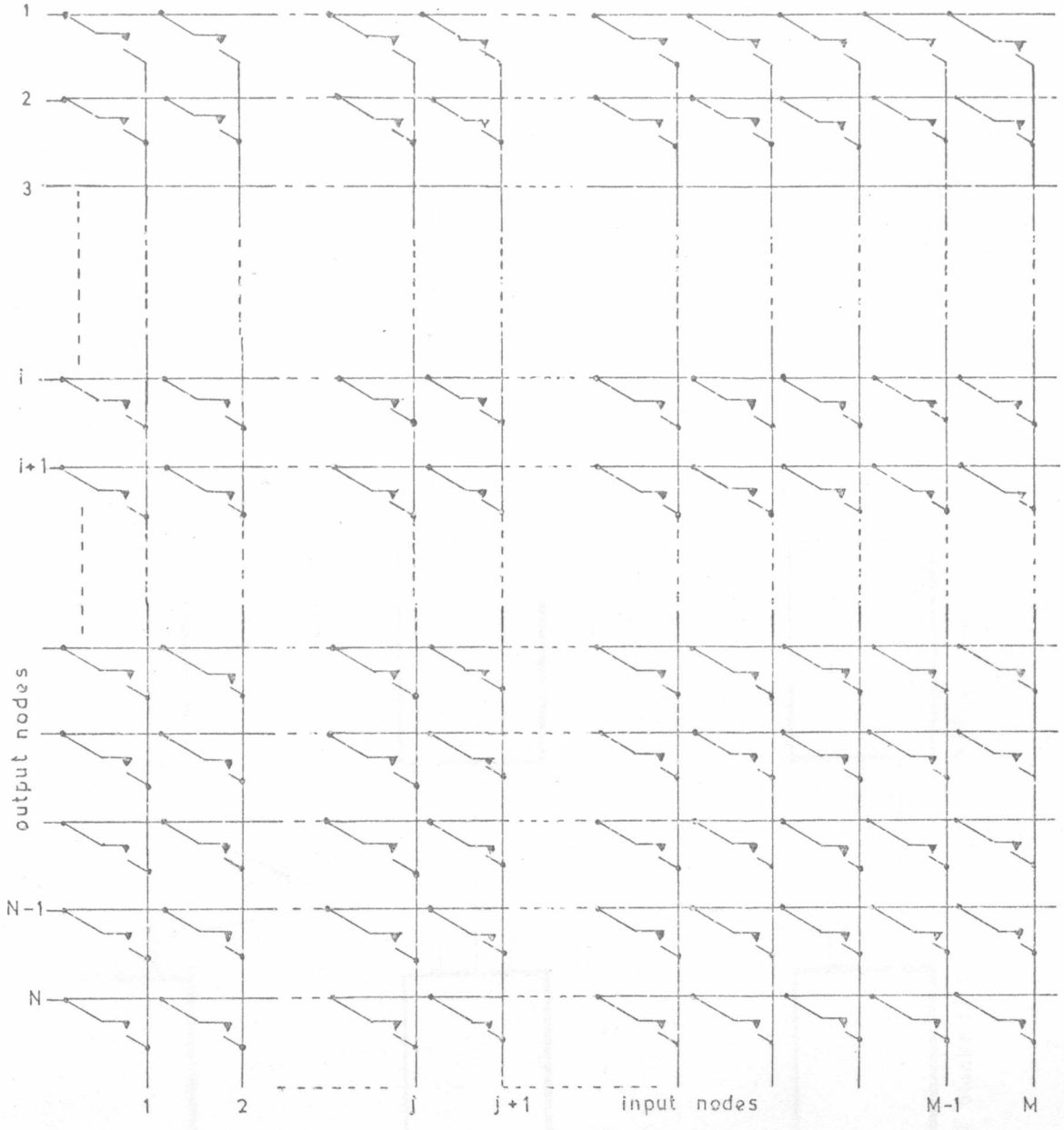


Figure 2.1 N x M switching matrix

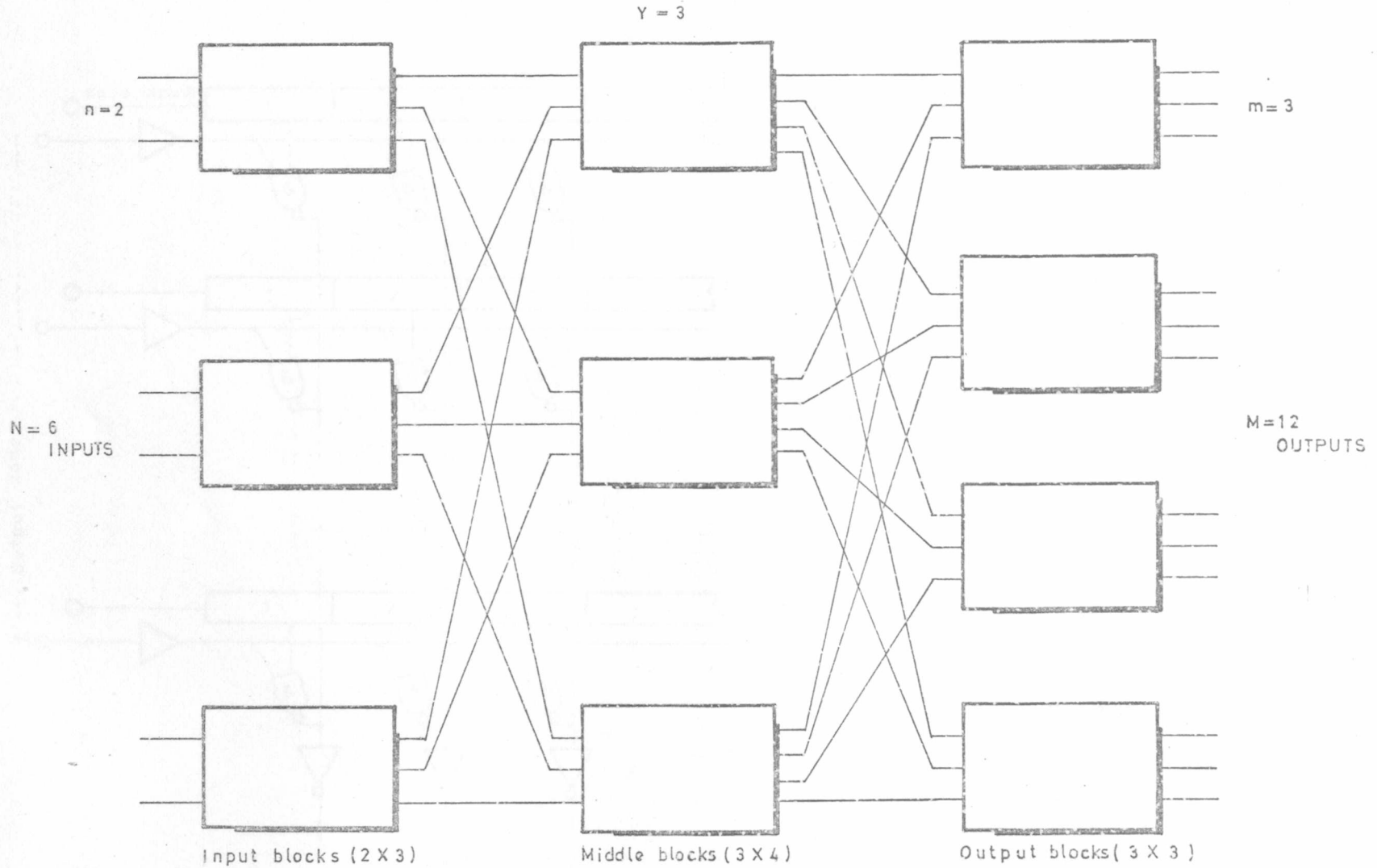


FIGURE 2.2 Three stage Matrix

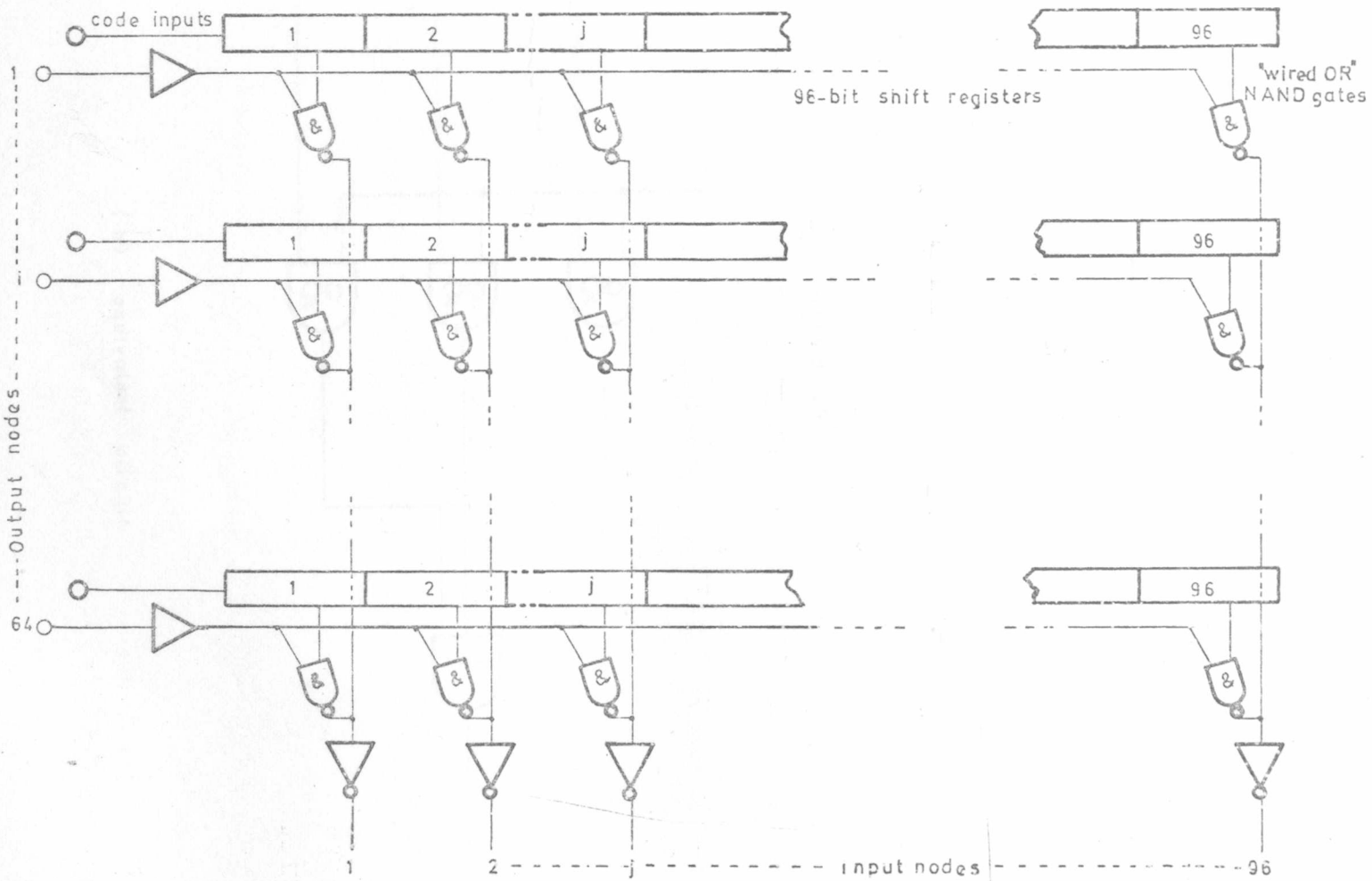
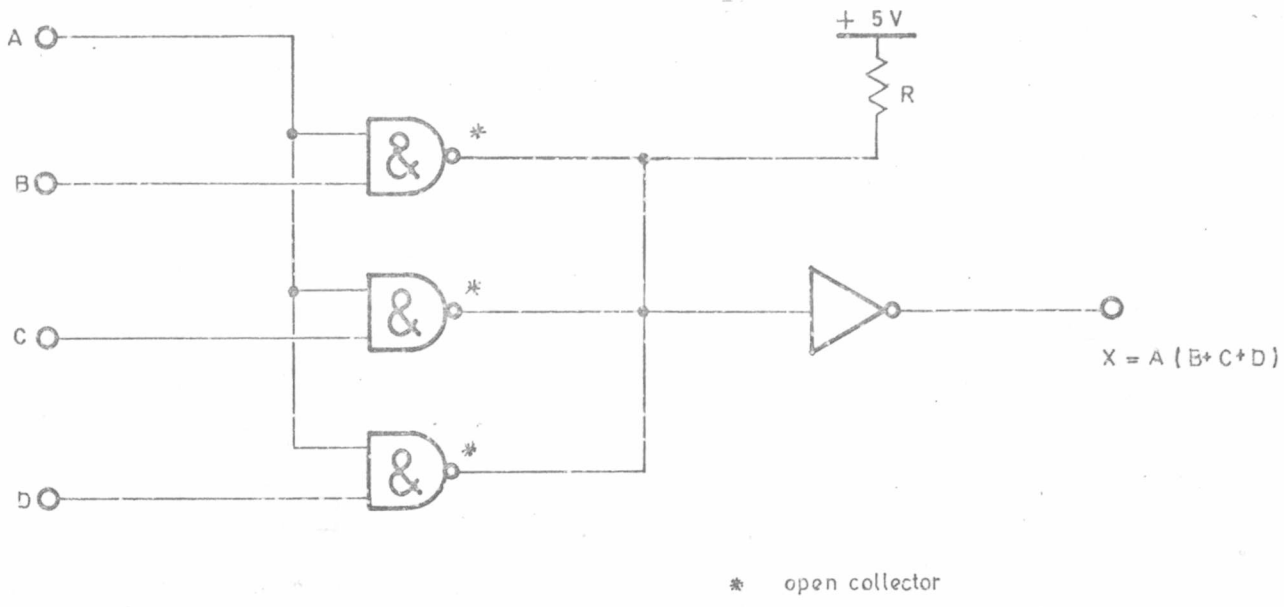
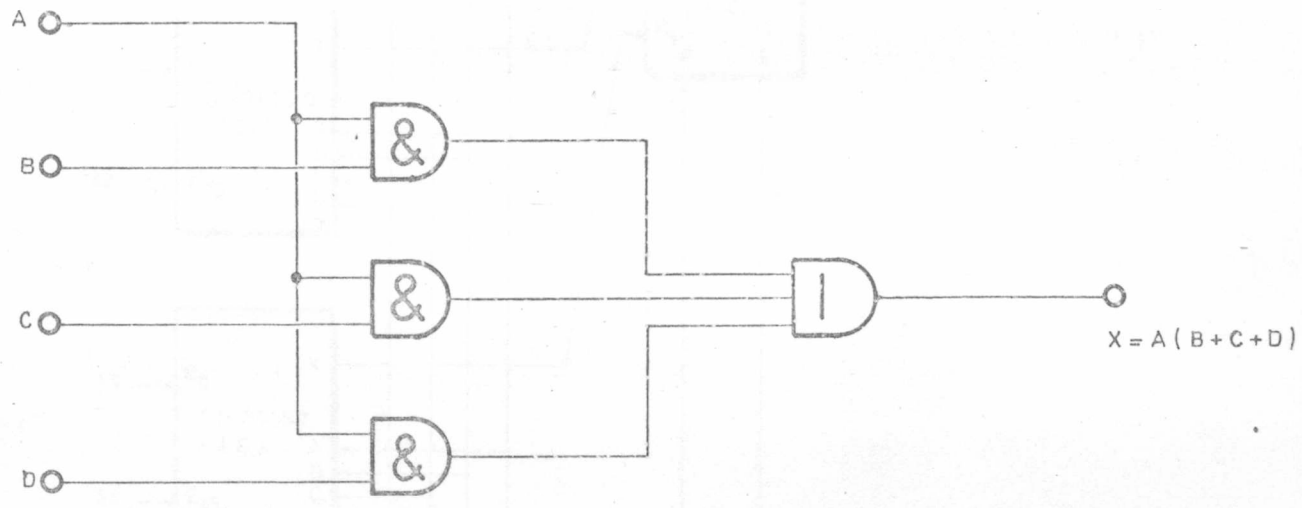


FIGURE 2.3 Initial patching system



(a) "wired OR"



(b) equivalent circuit

FIGURE 2.4 "wired OR" arrangement

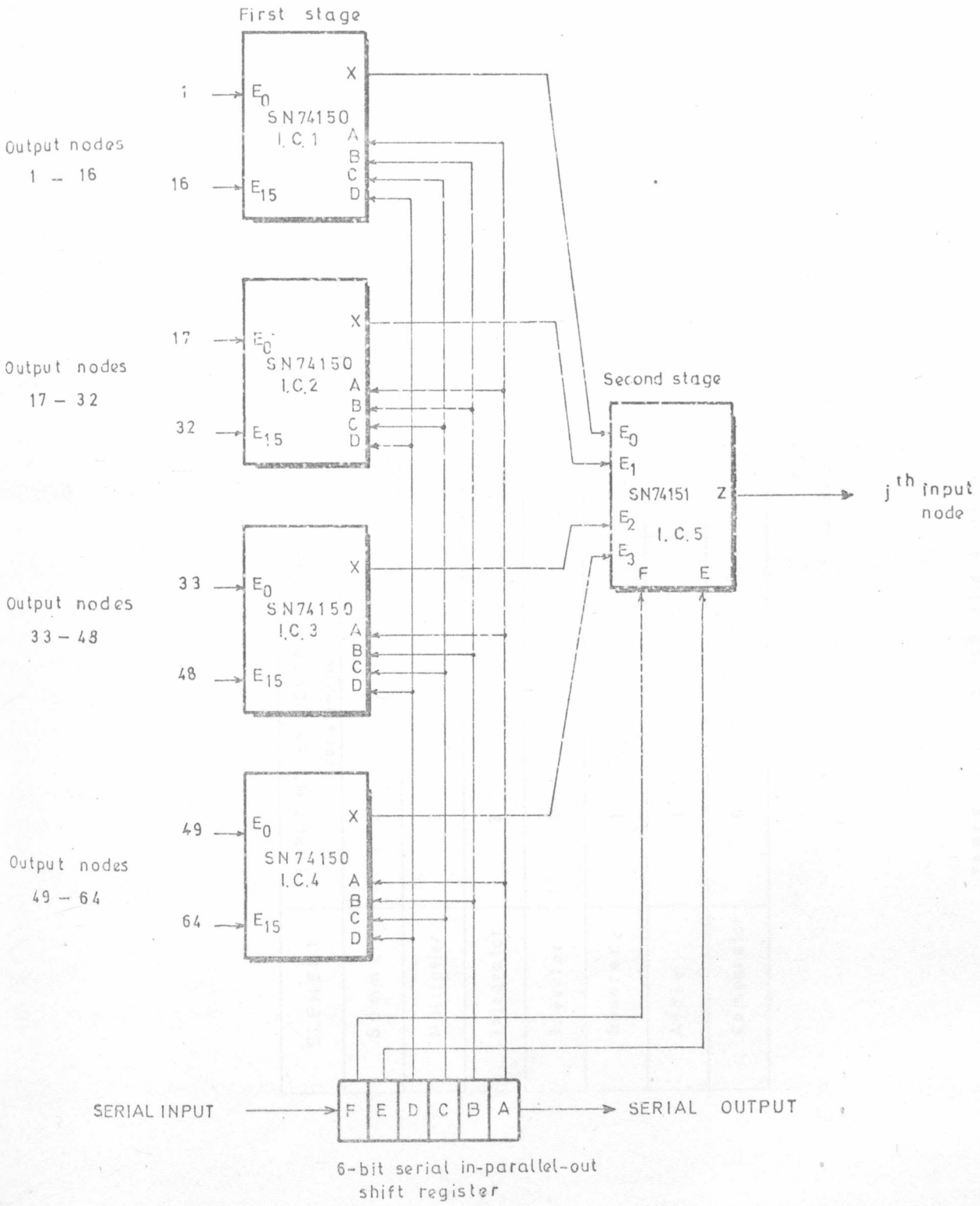


FIGURE 2.5 Final system design

ELEMENT	INPUT NODES per element	OUTPUT NODES per element	NUMBER OF ELEMENTS	INPUT NODES	OUTPUT NODES
Summer	2	1	10	20	10
Multiplier	2	1	20	40	20
Integrator	2	1	10	20	10
Invertor	1	1	6	6	6
Squarer	1	1	4	4	4
Addie	1	0	6	6	0
Comparator	0	1	14	0	14
			TOTAL	96	64

TABLE 2.1 Stochastic elements to be catered for.

I. C's USED	NUMBER PER CIRCUIT	TOTAL NUMBER	COST PER UNIT	COST
SN 7401	24	1536	0·16	245·76
SN 7404	——	96	0·12	11·52
SN 74164	12	768	2.10	1612.80
SN 7440	2	128	0.12	15.36
96 Veroboards			2.00	192.00
3 Veroracks			25.00	75.00
			TOTAL	£2152.44

TABLE 2.2 Estimated cost of initial system

I. C's USED	NUMBER PER BOARD*	TOTAL	COST PER UNIT	COST
SN74150	16	384	2.50	960.00
SN74151	4	96	0.90	86.40
SN7404	11	264	0.12	31.68
SN74164	3	72	2.10	151.20
Note 1 refers SN7440	17	34	0.12	4.08
26 Veroboards			2.00	52.00
1 Vero rack			25.00	25.00
			TOTAL	£1310.36

* One board contains 4 data selectors.

Note 1. On the 2 buffer boards only.

TABLE 2.3 Final system cost

CHAPTER 3

INITIAL CONDITIONS FACILITY

3.1 Introduction to Problem

Because a stochastic computer is a parallel processing machine many of its applications will involve the use of integrators, eg, solution of differential equations. It is therefore necessary to have the facility of setting the state of each integrator to some value at the beginning of a program. Consider equation (1.5) which gives the output of an integrator as

$$E^*(t) = E^*(0) + \frac{1}{N\tau} \int_0^t (E_1(t) + E_2(t))dt$$

where $E^*(0)$ is the quantity represented by the integrator output at time $t = 0$, ie, the initial conditions. This quantity $E^*(0)$ will be known from the particular problem to be implemented and may have any value in the range $-V$ to V, thus it is essential to program the integrators accordingly.

The initial state of an integrator may be loaded by a hardware or software method but a software implementation was rejected for the following reasons. A software system would have involved major alterations to the master program and this was considered undesirable. Secondly the stochastic computer was becoming increasingly dependent on the PDP8/E and if the control of DISCO were to become excessively complex then this would rule out the possibility of employing a microprocessor as the supervisory computer in the future. One advantage of using a microprocessor as opposed to a mini-computer is that considerable savings in cost may be achieved because a microprocessor can operate with a read only memory (ROM) as a programming device. If the control programs become too large then the use of a ROM would be impractical and this would mean the addition of expensive magnetic core as /

as a stored program medium. As the supervisory programs increase in complexity then the memory requirements will increase and hence costs will increase.

Having decided to employ a hardware implementation the operation of the system must now be defined. This may be subdivided into "WRITE" and "READ" operations.

(i) WRITE operation; this involves the transfer of information from the memory of the PDP8/E to a memory incorporated within the framework of the initial conditions circuitry. It is necessary to perform this operation only once for a given problem, prior to the programming of DISCO.

(ii) READ operation; in this mode of operation the initial conditions are loaded into the integrators upon instruction from the PDP8/E. This operation must be performed each time the problem is run on DISCO.

3.2 System Design

With the modular arrangement described in Chapter 4, it is possible to have an integrator in any one of 34 positions, these positions being uniquely numbered from 1 to 34. It is therefore necessary to store 34 12 bit words, each word being the representation of the initial conditions of its corresponding integrator. If any of the slots 1 to 34 do not contain an integrator then this event is regarded as being equivalent to an integrator with an initial condition of -V, ie, all bits of the 12 bit word are zero.

The circuitry involved is centred upon the Signetics 2519 integrated circuit which is detailed in Figure 3.1. This is a 6x40 bit MOS static shift register and by using two of these ICs a memory capable of storing 40 12 bit words can be obtained. Although only 34 12 bit words need be catered for this IC was chosen for convenience /

convenience of cost and operation. It is possible to operate this device in a recirculate or write mode by application of a logic '1' to the recirculate pin (see Figure 3.1). The diagram of the system is shown by Figure 3.2. Throughout the circuitry the following abbreviations have been used.

C_C: clock from PDP8/E for entering information into the 12 bit serial in 1 parallel out shift register (SR).

W : write command from PDP8/E.

C_M: master clear from PDP8/E. This is also used to clear the integrators.

C_D: DISCO master clock which operates integrator counter.

I : 'count up' signal to integrators.

The write line and the master clear line determine the mode of operation of the circuitry. If the write line is high when the master clear pulse is received then information is transferred from the PDP8/E to the MOS shift registers. This is the WRITE operation. Conversely if the write line is low at the time of the clear pulse then the initial conditions are deposited in the integrator. This is the READ operation. The WRITE operation will be considered first.

For reasons of clarity the generation of some control signals is not detailed in Figure 3.2. The logic equations realised by the combinational logic throughout the circuit are given in Table 3.1.

When the WRITE line is high, the following will occur upon receipt of the clear pulse C_M.

(i) the ÷12 counter is reset to zero.

(ii) FF1 is cleared, ie, Q_{FF1} becomes logic '0'. This ensures that only the output of the ÷12 counter (C_{12}) will clock the MOS shift registers.

The initial conditions, in binary form, of the first integrator (the integrator in slot 1) are fed into the 12 bit SR. This requires 12 clock pulses from the PDP8/E. After 12 clock pulses, one clock pulse is delivered from C_{12} and is used to clock the information in the 12 bit SR into the MOS registers. The initial conditions of the first integrator are now contained in the MOS memory and this sequence is repeated for the initial conditions of the first, second, etc through to the fortieth integrator. As previously mentioned only 34 integrators need be catered for but because there are 40 locations within the MOS memory there are considered to be 40 integrators for the purpose of the WRITE operation. If there is no integrator in a particular slot then the corresponding 12 bit word in the MOS memory is set to binary zero. Because the initial conditions of the fortieth integrator were the last to be stored in the MOS memory, the initial conditions of the first integrator have been clocked through to the output stages of the MOS shift registers and the WRITE operation is now complete.

During the READ operation the WRITE line is low and MOS shift registers will be in the recirculate mode, ie, the information stored in the MOS memory will be retained. When the WRITE line is low at the time of the clear pulse the following will occur.

(i) FF1 is preset, ie, Q_{FF1} becomes logic '1' which enables the MOS memory to be clocked from the comparator. The clocks of the twelve bit counter and the HOLD shift register are also enabled by Q_{FF1}.

(ii) the HOLD register is preset, ie, the hold lines are all high except for hold line one which is Q_{FF2} which ensures that only integrator 1 can count up.

(iii) /

(iii) the 12 bit counter is reset to zero.

The 12 bit counter is in effect a simulation of the integrator counter because both are reset to zero by C_M and both will count UP at the same rate and so the 12 bit counter may be thought of as the integrator counter.

The comparator will give an output when the state of the integrator counter is greater than the 12 bit word occupying the output stage of the MOS memory. At the beginning of the READ operation this 12 bit word is the binary representation of the initial conditions of the first integrator. Thus the first integrator will count up until it is one state greater than the required initial state at which point an output pulse is delivered by the comparator and this is used to clock the MOS memory, the HOLD register and to clear the dummy integrator counter. The circuitry is now ready to set the initial conditions of the second integrator which are presented to the comparator and the hold line of the second integrator now contains logic '0'. This sequence will be repeated until the 40 12 bit words in the MOS memory have been recirculated, ie, all initial conditions have been loaded. At this point the logic '0' in the HOLD register will occupy Q_{40}. The logic '0' in Q_{40} is used to clear the HOLD register, thus enabling all integrators, and to clear FF1 so as to prevent the comparator output from clocking the MOS memory. If, during the above sequence, there is no integrator in a particular slot then only one master clock pulse (C_D) will be required to produce an output from the comparator and so preparing the next integrator for the setting up of its initial conditions.

It is necessary to add some simple circuitry to the integrators so as to ensure that each integrator will count UP. This is achieved by setting both integrator inputs to logic '1'. Figure 3.3 shows the additional circuitry. The 'count UP' line Q_{FF1} is always high during /

during the READ operation and is used to ensure that both inputs E_1 and E_2 to the integrator are high, ie, it will count UP. The outputs E_1 and E_2 are the stochastic sequences A and B respectively providing Q_{FF1} is low. Otherwise E_1 and E_2 are both high.

The flowchart for the PDP8/E program is given in Figure 3.4 and the program is listed in reference 6.

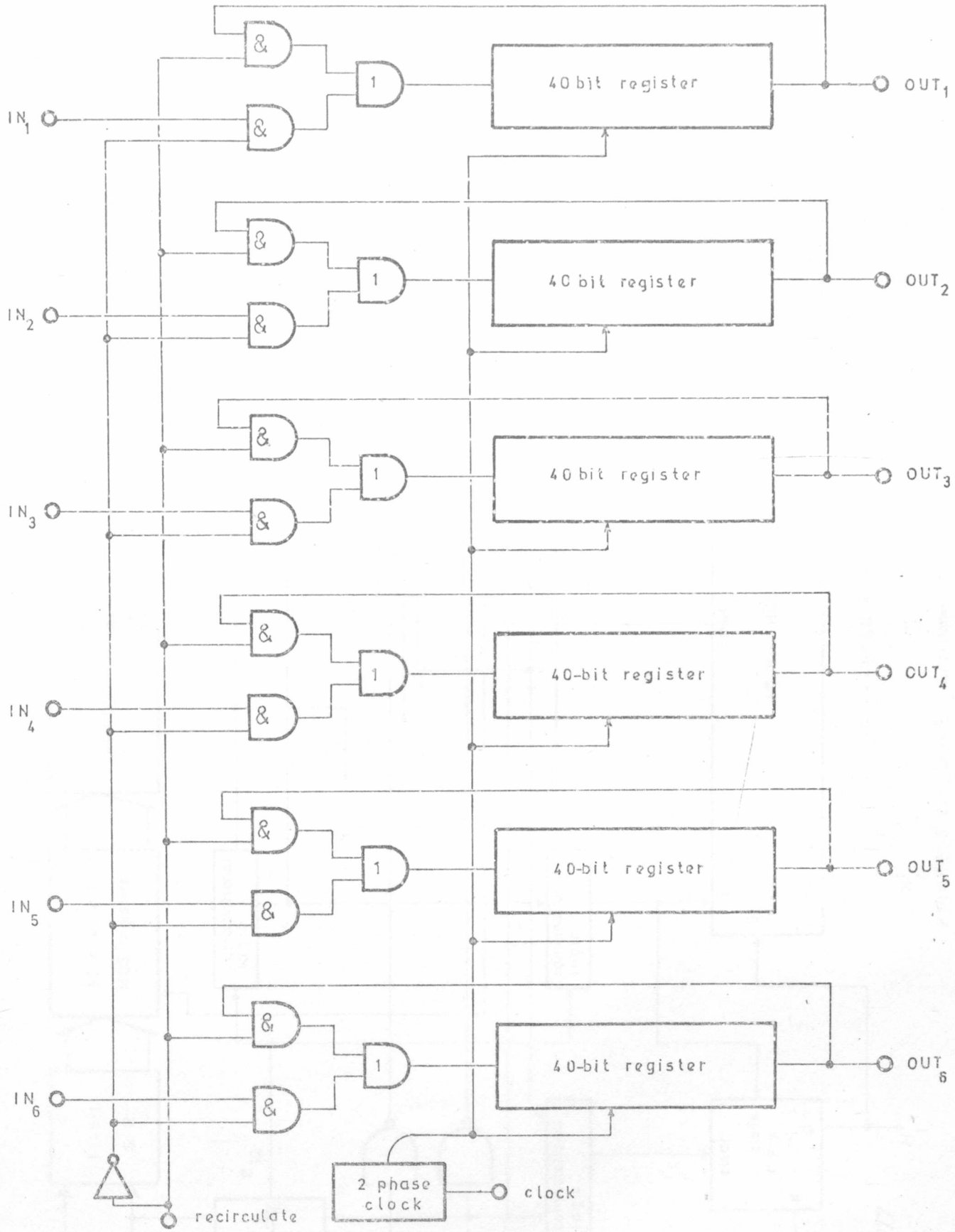


FIGURE 3.1 Internal circuitry of Signetics 2519

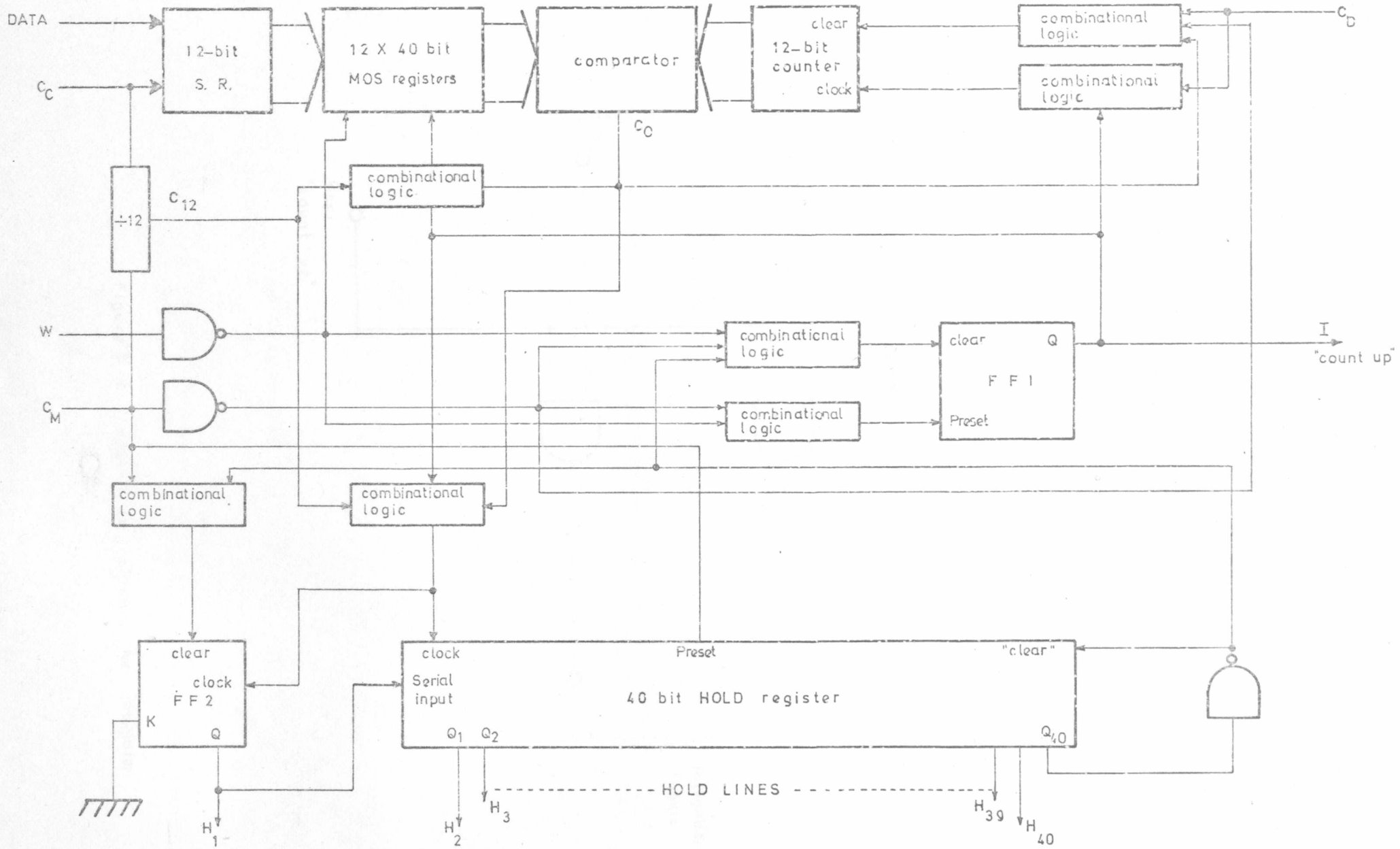


FIGURE 3.2 Initial conditions circuitry

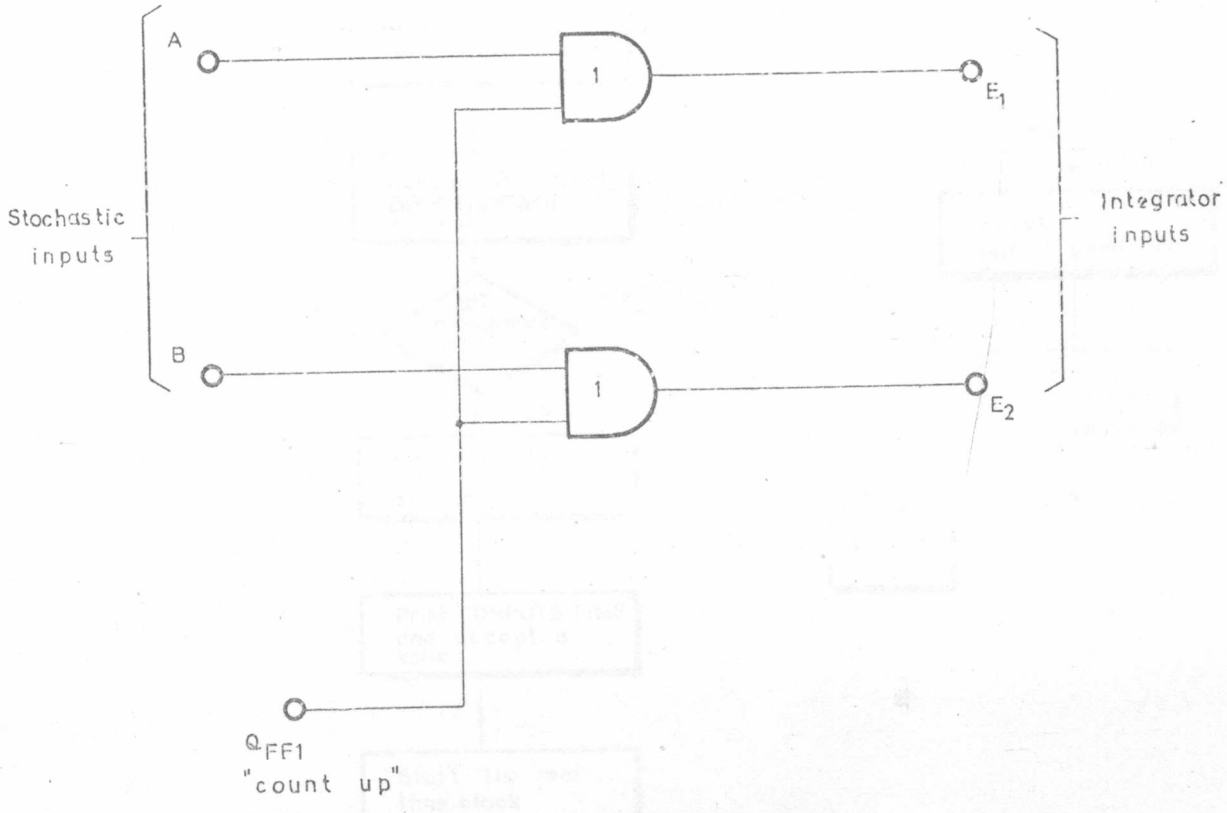


Figure 3.3 "count – up" circuit for integrator

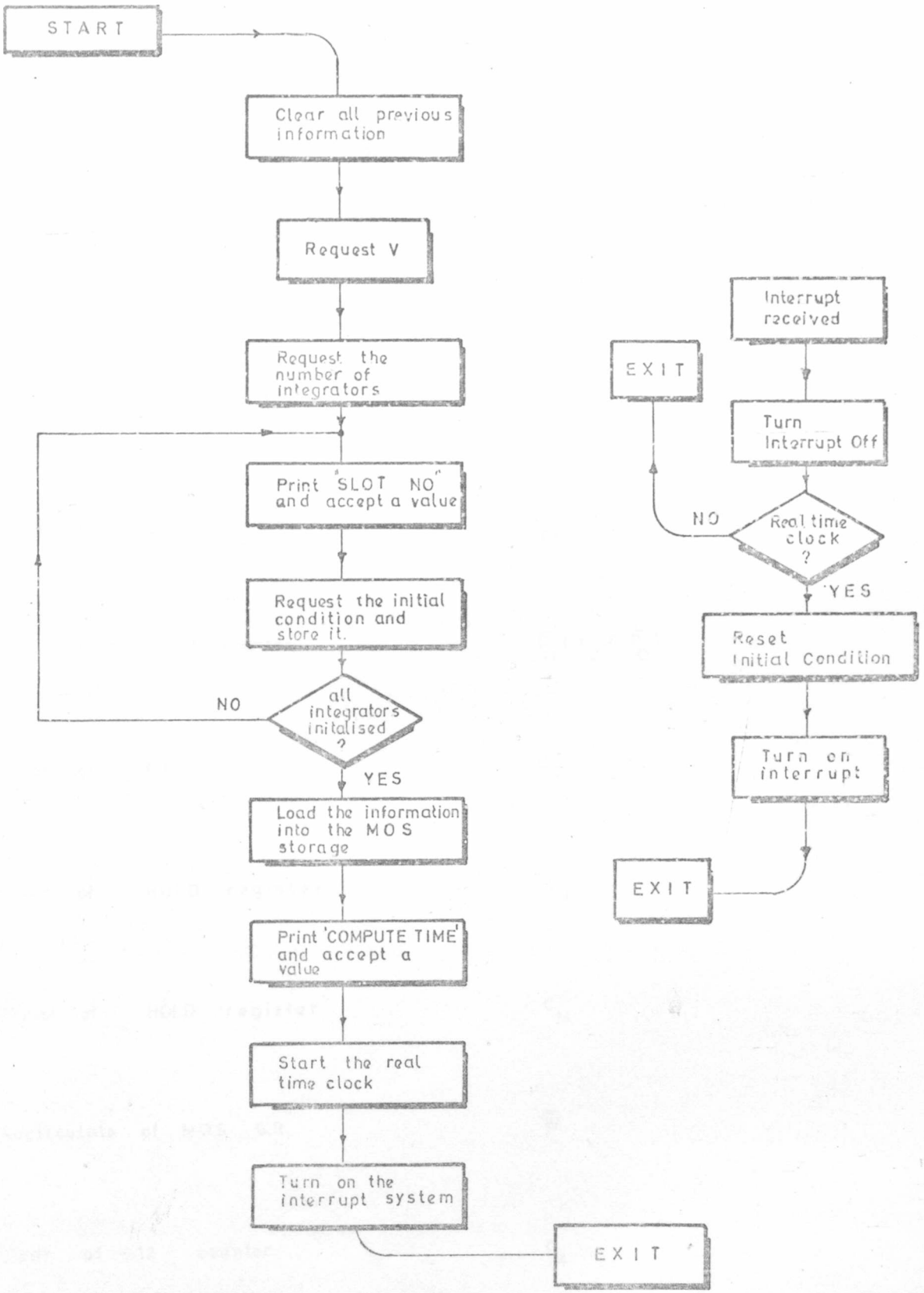


Figure 3.4 Setting up of the Initial conditions

CONTROL INPUT	ALGORITHM
Clock of MOS S.R. and HOLD register	$C_{12} + C_0 \cdot Q_{FF1}$
Clear of FF 1	$Q_{40}(\overline{W} + \overline{C}_M)$
Preset of FF 1	$W + \overline{C}_M$
Clock of 12 bit counter	$Q_{FF1} \cdot C_D$
Clear of 12 bit counter	$\overline{C}_M (C_D + \overline{C}_0)$
Clear of FF 2	$\overline{C_M} \cdot \overline{Q_{40}}$
Clear of HOLD register	$\overline{Q_{40}}$
Preset of HOLD register	C_M
Recirculate of MOS S.R.	$\overline{W}$
Clear of ÷12 counter	C_M

TABLE 3.1

CHAPTER 4

GENERAL SYSTEM ORGANISATION OF DISCO[6,7]

4.1 Scaling of Integrator[5,6]

In many problems which are to be solved using a stochastic computer it is necessary to have the facility of scaling an integrator. This facility is incorporated within an analogue computer, ie, choice of nose gains. With an analogue computer scaling is achieved by selecting one of a number of possible time constants for an integrator and this basic method is also suitable for the stochastic computer.

Consider equation (1.5), which gives the output of an integrator as

$$E^*(t) = E^*(0) + \frac{1}{N\tau_1} \int_0^t (E_1(t) + E_2(t))dt$$

where

$E^*(0)$ is the initial state of the integrator,

E_1 and E_2 are the deterministic equivalents of the stochastic inputs,

τ_1 is the period of a clock pulse, and

N is the number of states of the integrator counter.

The time constant of an integrator is thus $C_\tau = \frac{1}{N\tau_1}$ or $C_\tau = f_c/N$ where f_c is the clock frequency. Therefore the time constant of an integrator may be varied by varying f_c or N. In DISCO scaling is achieved by selecting one of five values of N. The number of states of an integrator is

$N = 2^n$ where n is the bit capacity of the integrator counter.

If n is increased by unity then

$$2^{n+1} = 2N \quad \text{and} \quad C_\tau = \frac{f_c}{2N}$$

and /

and conversely,

$$2^{n-1} = N/2 \quad \text{and} \quad C_\tau = \frac{2f_c}{N}$$

As n is reduced by unity then C_τ doubles for a fixed f_c.

Each integrator within DISCO may have a programmable counter length of 12, 11, 10, 9 or 8 bits giving the scaling factors as 1, 2, 4, 8 or 16 respectively.

Because of the nature of the UP/DOWN counters used, the integrator inputs have been rearranged from that in Figure 1.6 and a practical form of integrator is shown in Figure 4.1. The scaling arrangement is detailed in Figure 4.2.

The four least significant bits of the 12 bit counter may be programmed by the scaling code $X_1X_2X_3X_4$ so as to constitute a 0, 1,2, 3 or 4 bit UP/DOWN counter. Table 4.1 shows the effective integrator counter length for the corresponding scaling code. It is seen that only one code bit may be logic '0' at any one time. Gates A, B, C and D in Figure 4.2 control the length of the counter. If for example X_3 was logic '0' then the control signals from the two least significant bits are inhibited and the J and K inputs to FF3 both become logic '1' thus making FF3 change state at each clock pulse, ie, FF3 becomes the least significant bit of the counter.

The next eight bits of the counter are unaffected by scaling and require only an UP/DOWN line and an enable signal from the previous stages.

A major disadvantage of this method of scaling is that a loss of resolution occurs due to the reduced counter length /

length. An alternative method of scaling is discussed in section 9.1.

4.2 Modular Arrangements of Elements [6]

From the discussion in Chapter 2 it is clear that as the system capacity (number of computing elements) increases then the patching system increases at a far greater rate. This introduces a practical limit on the system computing capacity. In fact the final patching system described in Chapter 2 is the largest feasible system available, a fact which may be demonstrated by the following example. At present the patching system is capable of connecting any one or more of 64 outputs to any one or more 96 inputs, this corresponding to 64 computing elements. Now if the requirement were to patch the inputs and output of 128 elements, this would double the system capacity. For 128 elements (outputs) there will on average be 192 inputs. It will be remembered from section 2.6 that one 64 to 1 line data selector is necessary for each input node. In the case of 192 input nodes there will be 192 data selectors which immediately means a doubling of hardware. Each data selector must now have 128 inputs, ie, the hardware involved in the construction of one data selector is doubled. Thus for a 192 by 128 patching system the hardware involved is four times that required for a 96x64 system. It can thus be seen that a doubling of system capacity means increasing the patching system hardware by a factor of four. Clearly this would be very expensive and so an alternative must be sought for increasing system capacity.

A second important point concerning the patching is that each input and output is committed to a computing element. Very few problems would use all of the elements at any one time. Therefore there will be a considerable degree of redundancy involved in the stochastic /

stochastic computer. The degree of redundancy may be reduced by careful selection of the computing elements available. For example in the solution of simultaneous equations the elements required would be comparators, invertors, summers, multipliers and ADDIEs and the number of each type of element required could be estimated by studying some typical problems. Therefore a certain fraction of the 64 elements would be summers, a certain proportion would be multipliers and so on for each element required. Unfortunately different classes of problem will require different proportions of each element and so this is impractical for a general purpose stochastic computer.

It would be preferable if the inputs and outputs of the patching system were not committed to a computing element but were associated with a slot position which could accommodate any type of element. This modular arrangement was adopted for the construction of DISCO.

Using this arrangement, up to 64 computing elements (each one being selected by the operator) can be used in any one problem whereas previously the level of redundancy meant that less than 64 elements would be utilised. Thus the computing capacity of DISCO is effectively increased by the use of a modular arrangement.

To allow any type of element to be inserted in any one slot, some convention must be used for the edge connections of all boards containing computing elements and is given in Table 4.2.

Because of construction and wiring difficulties the modular arrangement has been reduced to 34 modular positions in which any element may be housed. The remaining 30 elements are fixed and consist of 8 invertors, 10 multipliers and 12 comparators. Each of these elements has its input and output nodes committed to specified patching inputs and outputs. The actual nature of these fixed elements were determined /

determined by a brief examination of typical examples of different classes of problems applicable to a stochastic computer. From this examination it was clear that some types of elements would be utilised regardless of the type of problem to be solved. These elements were selected as being the most suitable, as regards the minimum of redundancy, for fixed positions.

Figure 4.3 shows the modular arrangement in slots 1 to 34, each modular slot in DISCO being numbered. The dedicated patching inputs and outputs are given with the associated fixed elements. The information given in Figure 4.3 is necessary for the programming of any problem and this is in fact reproduced on the front panel of DISCO as may be seen from Plate 4.1.

In the event of more invertors, multipliers or comparators then those in fixed positions being required some extra elements are available in modular form and may be accommodated in the modular positions.

When an element with only one input, eg, an invertor, is used in a modular position then the lowest number of the two associated inputs is the one to be patched. There are two inputs to the patch panel which have not been allocated and may be used as a means of entering external signals into the system, for example a stochastic sequence from another system.

Finally the output interface elements are compatible with this modular arrangement and can be located in any one of the modular slots with the digital or analogue outputs being taken directly from the board containing the output interface.

4.3 Interface with PDP8/E

The digital computer used to supervise the operation of the stochastic computer is a DEC PDP8/E. Operation and programming of DISCO is controlled from a visual display /

display unit (VDU). This makes programming extremely simple and is explained in more detail in the following section.

Plate 4.1 shows the complete system which consists of the VDU, the PDP8/E and the stochastic computer.

The operations performed by the PDP8/E are:

(i) control of the patching system. This entails the loading of the shift register controlling the data selectors.

(ii) control of initial conditions. In this operation the initial condition of each integrator is entered into the memory contained within the initial conditions circuitry. The PDP8/E must send a clear pulse to all integrators and the initial conditions board to start each run of any problem.

(iii) control of scaling. This enables the bit capacity of each integrator to be set according to the scaling required.

(iv) control of comparators. Each comparator is loaded with a 12 bit binary word which is the equivalent of the weighting of the stochastic sequence to be produced.

(v) reading of ADDIE. The 12 bit binary word representing the solution of a problem is taken from the ADDIE and is displayed in decimal form on the VDU.

(vi) plotting of distribution curves. This is a useful operation and is used to show the mean value of a stochastic sequence and its associated variance over a range of samples. A distribution curve gives an insight into the accuracy and bias of the stochastic sequence because a deviation from the expected mean can be seen (bias) and also the range in which a sample may be expected to fall (accuracy).

(vii) graph on X-Y plotter. Again this is a useful function and can be used to give a distribution curve in graphical form.

These operations are demonstrated below in the form of a programming example.

4.4. Programming Procedure

Before describing the procedure for programming a problem on DISCO the equations describing the operation of each computing element will be restated below.

(i) For an invertor with input E the output is

$$E^* = -E$$

(ii) The output of a multiplier with inputs E_1 and E_2 is

$$E^* = \frac{E_1 E_2}{V}$$

and in the special case of $E = E_1 = E_2$

$$E^* = \frac{E^2}{V}$$

which is the squaring operation.

(iii) For inputs E_1 and E_2 the summer output is

$$E^* = \tfrac{1}{2}(E_1 + E_2)$$

(iv) Finally the output of an integrator is given as

$$E^*(t) = E^*(O) + \frac{1}{N\tau_1} \int_O^t (E_1(t) + E_2(t))dt$$

The problem chosen to demonstrate the programming of DISCO is that of a second order system with zero damping, ie, a sine wave generator.

Consider /

Consider the equation

$$\frac{d_2\, x(t)}{dt^2} + \omega^2\, x(t) = 0 \qquad \text{---- (4.1)}$$

which may be rewritten using the convention

$$\ddot{x} = \frac{d_2 x}{dt^2} ,$$

as

$$\ddot{x} + \omega^2 x = 0 \qquad \text{---- (4.2)}$$

Equation (4.2) has the standard solution

$$x(t) = \frac{\dot{x}(0)}{\omega} \sin \omega t \qquad \text{---- (4.3)}$$

where $\dot{x}(0)$ is the value of $\frac{dx}{dt}$ at $t = 0$.

ie, a sine wave of natural frequency ω and peak value $\frac{\dot{x}(0)}{\omega}$ subject to the initial condition $x(0) = 0$.

To establish the flow diagram the procedure is identical to that for an analogue computer.

Rewriting equation (4.2) as

$$\ddot{x} = -\omega^2 x$$

easily allows the flow diagram to be established and is shown in Figure 4.4.

To determine the relationship between ω and $\dot{x}(0)$ of equation (4.3) and the stochastic elements, the output of the flow diagram of Figure 4.4 must be given.

From Figure 4.4,

$$E_2 = E_2(0) + \frac{S_1}{N\tau_1} \int_0^t 2E_1 dt$$ where S_1 is the scaling factor,

and /

and

$$\frac{dE_2}{dt} = \frac{2E_1S_1}{N\tau_1}$$

$$E_1 = \frac{N\tau_1}{S_1^2} \cdot \frac{dE_2}{dt} \qquad \text{---- (4.4)}$$

Also

$$E_1 = E_1(0) + \frac{S_2}{N\tau_1} \int_0^t 2\,E\,dt$$

$$\frac{dE_1}{dt} = \frac{2S_2E}{N\tau_1}$$

$$E = \frac{N\tau_1}{2S_2} \frac{dE_1}{dt} \qquad \text{---- (4.5)}$$

Substituting (4.4) in (4.5),

$$E = \frac{N\tau_1}{2S_2} \frac{d}{dt} \left(\frac{N\tau_1}{2S_1} \frac{dE_2}{dt}\right)$$

$$= \left(\frac{N\tau_1}{2S_1S_2}\right)^2 \frac{d_2E_2}{dt^2} \qquad \text{---- (4.6)}$$

but $E = -E_2$

(4.6) becomes

$$-E_2 = \left(\frac{N\tau_1}{2S_1S_2}\right)^2 \frac{d_2E_2}{dt^2}$$

Rearranging

$$\frac{d_2E_2}{dt^2} + \left(\frac{2S_1S_2}{N\tau_1}\right)^2 E_2 = 0 \qquad \text{---- (4.7)}$$

This is seen to be identical in form to equation (4.1).

Thus /

Thus for a stochastic implementation of equation (4.1),

$$\omega = \frac{2S_1S_2}{N\tau_1}$$

and E_2 would represent $x(t)$. It is a necessary condition that $x(0) = 0$, ie, the initial conditions of integrator 2 are zero. The initial conditions of integrator 1 will determine the amplitude of oscillation.

Taking $S_1 = S_2 = 1$, ie, both integrators have 12 bit counters we have

$$\omega = \frac{2}{N\tau_1}$$

and peak value $= \frac{\dot{x}(0)}{\omega}$.

Consider now a numerical example

$$x(t) = 0.5 \sin 100t$$

$$\dot{x} = 50 \cos 100t$$

$$\dot{x}(0) = 50 \quad \text{and} \quad \dot{x}_{max} = 50$$

$$\omega = 100$$

but $\omega = \frac{2}{N\tau_1}$ where $N = 2^{12} = 4096$

$$\tau_1 = \frac{2}{4096 \times 100} = 4.8 \times 10^{-6} S$$

ie, a clock frequency of 204.3 kHz.

To implement this sine wave generator the following operations must be performed.

(i) The elements must be patched. Figure 4.3 shows the patching information beside each input and output. The integrators occupy slots 26 and 27.

(ii) Scaling of integrators must be achieved with both scaling factors set to unity.

(iii) /

(iii) The required initial conditions of the integrators must be established, ie, $\dot{x}(0) = 50$ and $x(0) = 0$. The value $V = 100$ is the maximum range and so the sinewave will be represented by a varying probability with maximum value 0.75 and minimum value 0.25. To enable the problem solution to be repeated at 5 second intervals the compute time is set to 5.

This sequence is shown in Plate 4.3. The required solution x(t) is represented by the stochastic sequence at E_2 and is patched to a stochastic to analogue convertor which will allow the solution to be displayed on an oscilloscope.

Plate 4.3 shows the code letters for each operation and these are detailed in Table 4.3.

An indication of the output derived from the sine wave generator is given in Figure 4.5. The waveform of Figure 4.5 was reproduced from the oscilloscope display of the output voltage of the S/A convertor. In the vertical direction the scale is 0.2 v/cm and in the x direction the scale is 20 mS/cm. The output voltage of the S/A convertor has the range ±1v which corresponds to ±V. Thus the waveform of Figure 4.5 can be seen to have the correct amplitude and period, ie, it represents the sine wave described by 0.5 sin100t.

Finally an explanation of the detail of Plate 4.2 will be given so as to give an idea of the size of DISCO. Plate 4.2 in fact shows a close up of DISCO with the larger bottom rack containing the patching system. The centre rack is the housing for the modular elements and contains 34 positions or slots. Above this rack is the rack for the fixed elements (comparators, invertors and multipliers) which also contains the boards required for the generation of the m-sequences. Underneath the oscilloscope is the panel which controls the master clock and /

and a special purpose stochastic simulator which is discussed in Chapters 6 and 7.

From Plate 4.2 it is seen that the patch panel is as large as DISCO itself as was mentioned in Chapter 2.

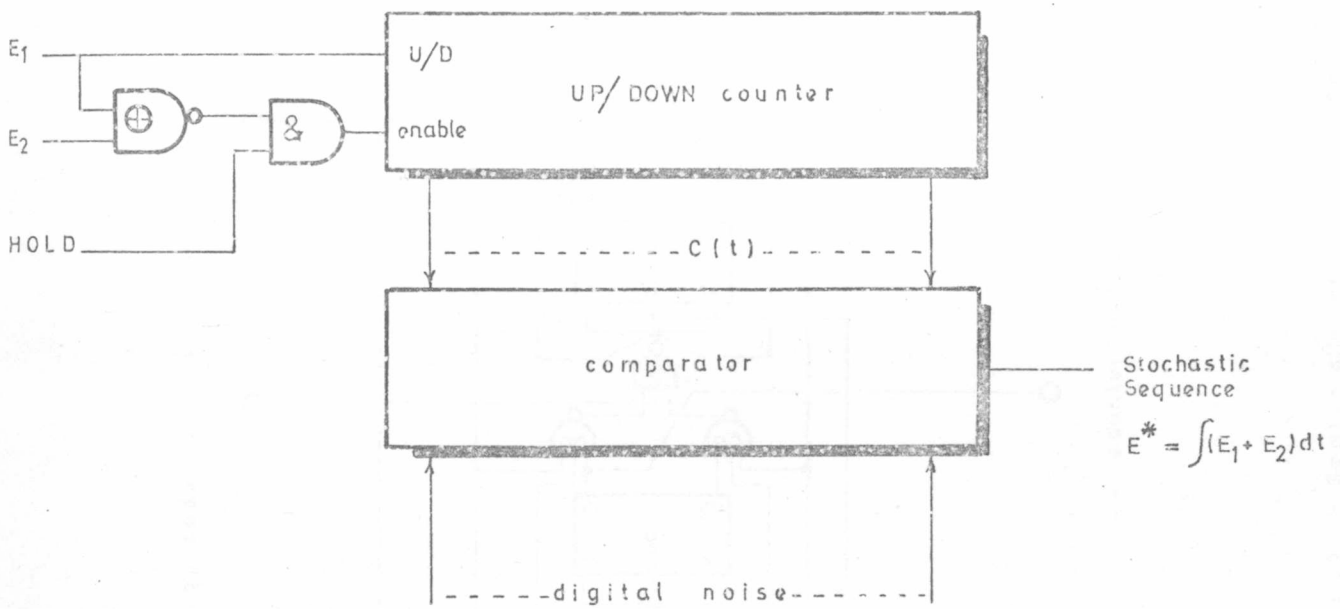


FIGURE 4.1 Practical form of Integrator

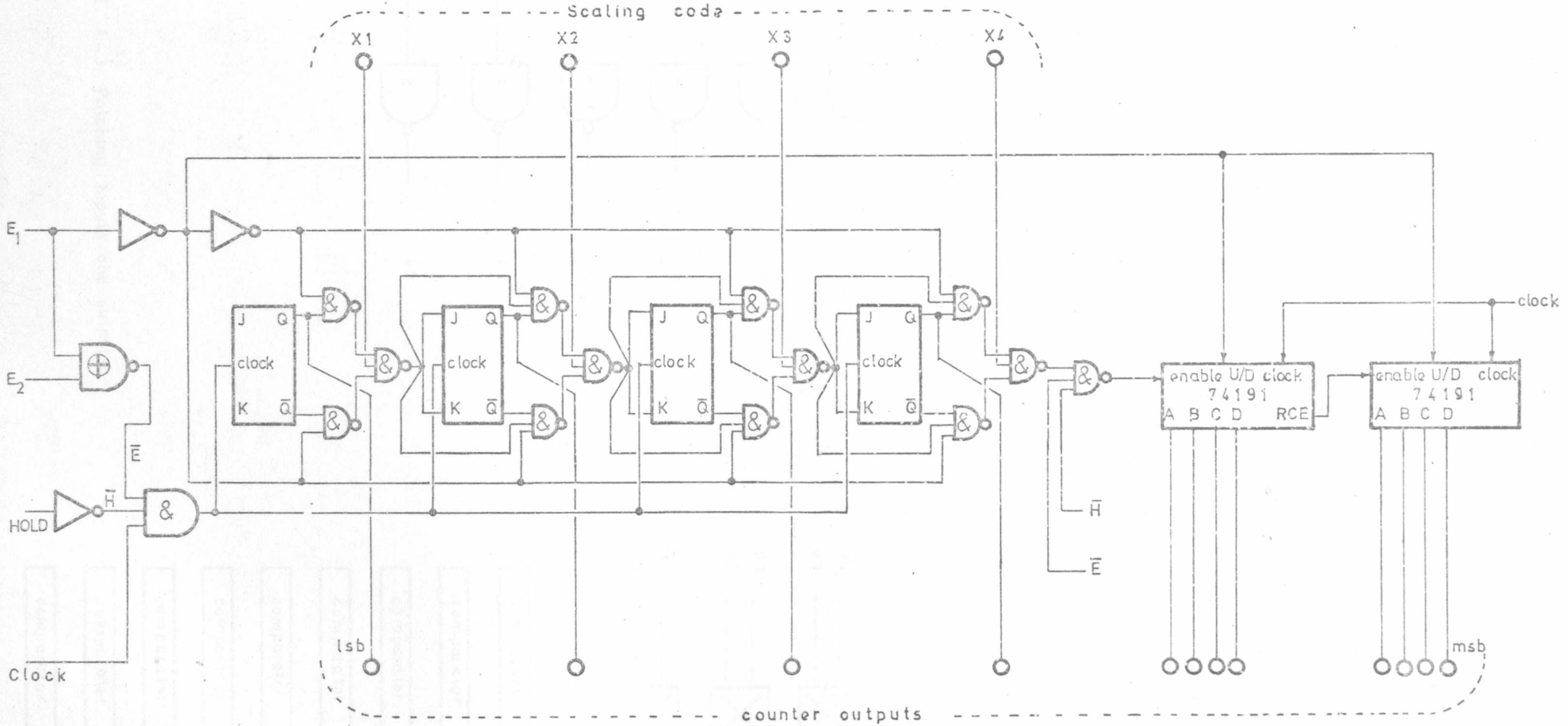


FIGURE 4.2 Scaling of Integrator

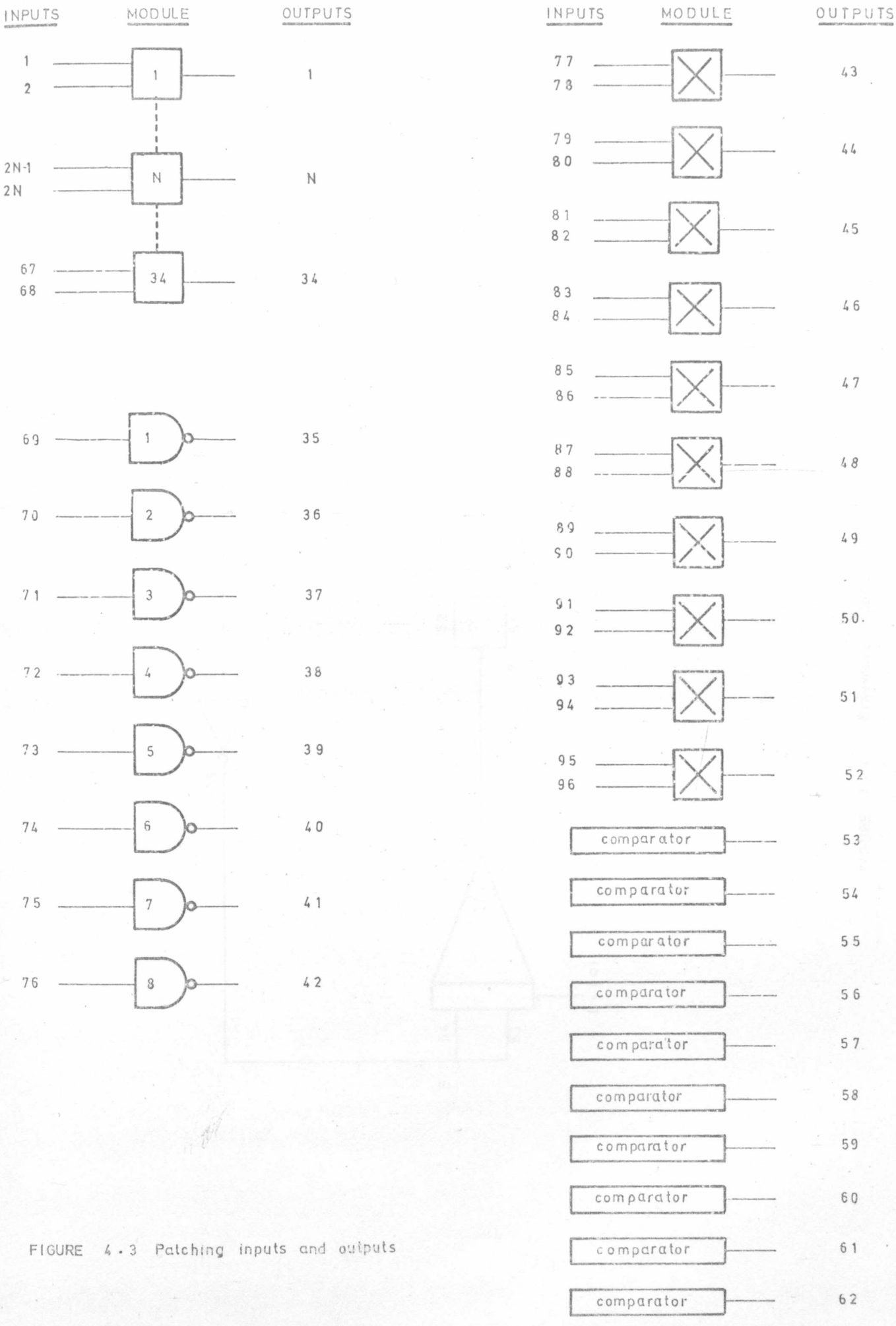


FIGURE 4.3 Patching inputs and outputs

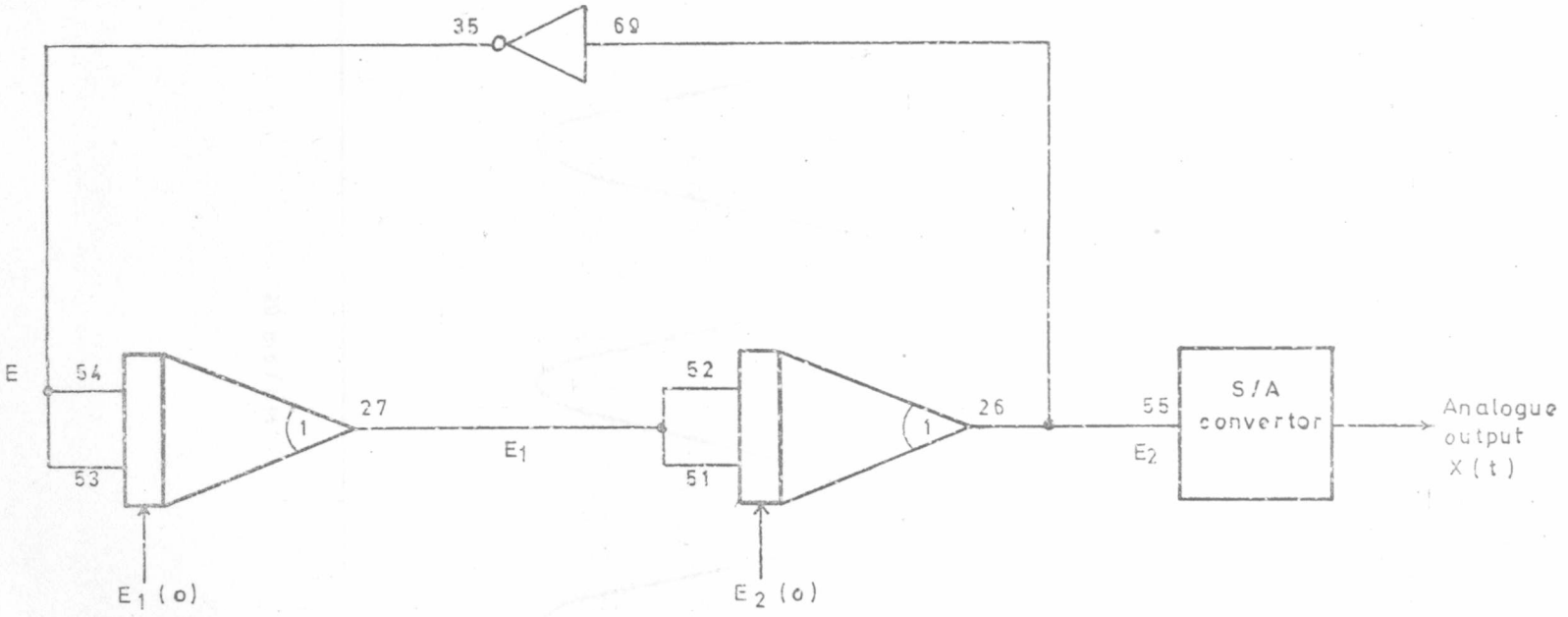


FIGURE 4.4 Sinewave generator

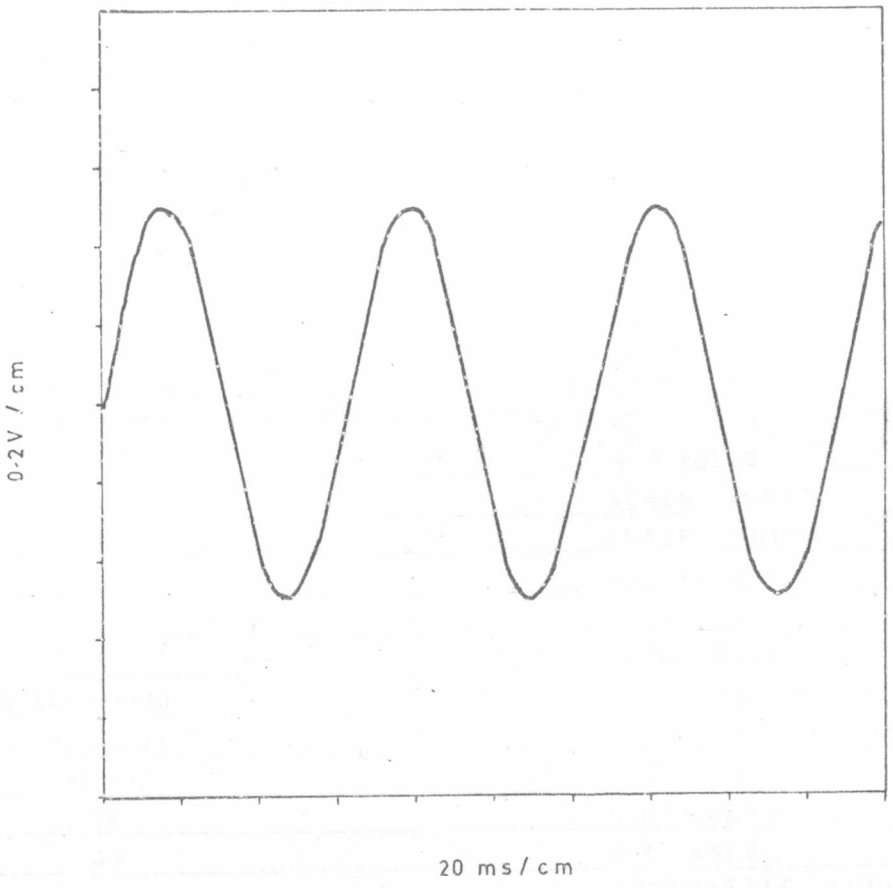


FIGURE 4.5 Output of Sinewave Generator

CODE	SCALING FACTOR	EFFECTIVE LENGTH OF INTEGRATOR
1 1 1 1	1	12
0 1 1 1	2	11
1 0 1 1	4	10
1 1 0 1	8	9
1 1 1 0	16	8

TABLE 4.1 Scaling code

PIN NUMBER	CONNECTION
1	+ 5 VOLTS
2	SCALE INPUT
3	SCALE OUTPUT
4	MASTER CLOCK
5	HOLD
6	SCALE CLOCK
8, 10, 12,----30	12-BIT NOISE INPUT
9, 11, 13,-----31	12 BIT DIGITAL OUTPUT
32---37	not used
38	STOCHASTIC OUTPUT
39	not used
40	STOCHASTIC INPUT
41	not used
42	STOCHASTIC INPUT
43	GROUND

TABLE 4.2 Pin connections

CODE LETTER	OPERATION
C	Patching
I	Comparator inputs
S	Scaling
R	Read number in ADDIE
D	Distribution curve
$	Transfer information from PDP8/E to DISCO
G	Graph on X—Y plotter
P	Presets Initial Conditions

TABLE 4.3 Program Code Letters

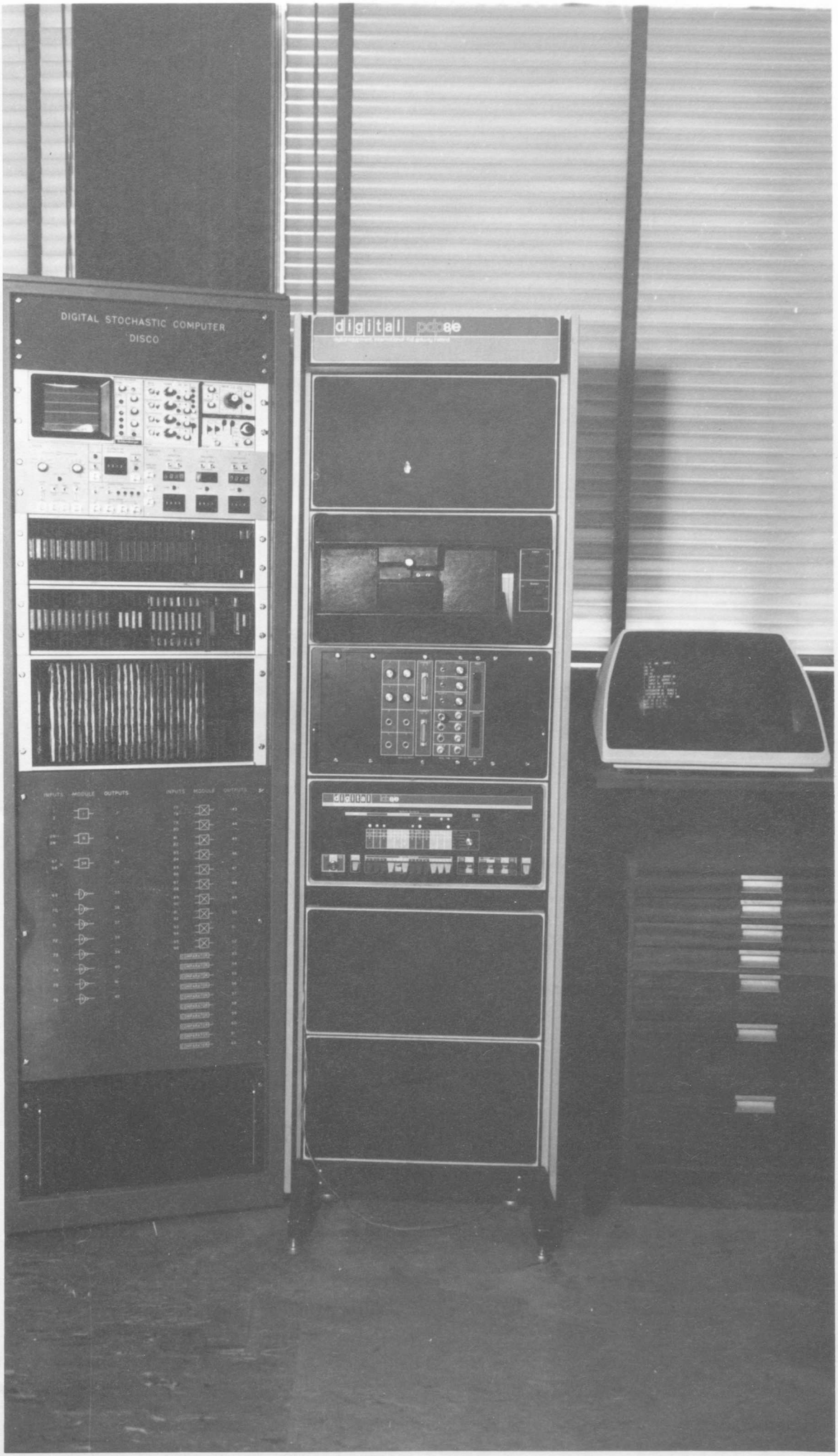


PLATE 4.1 General view of system

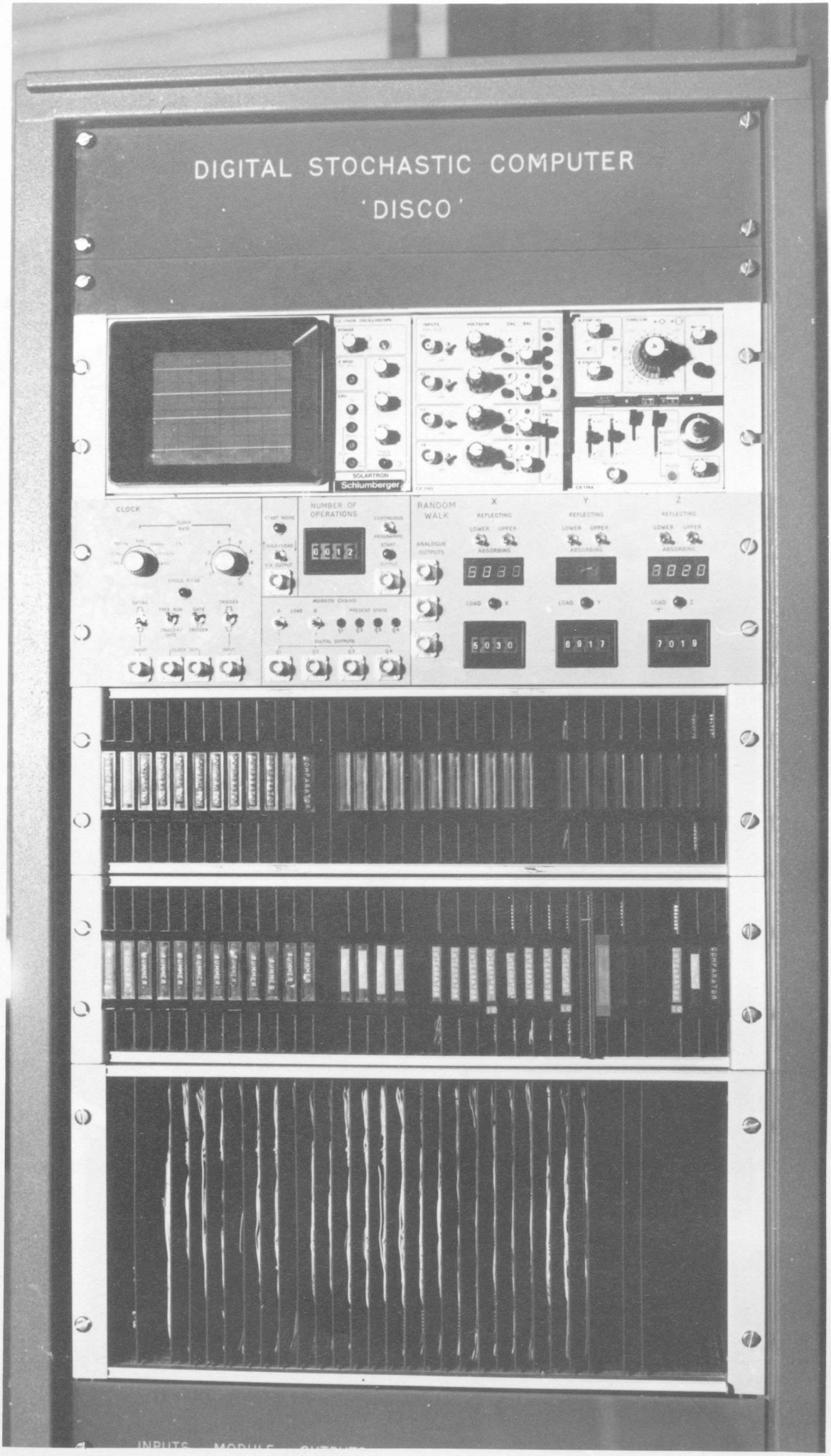


PLATE 4.2 Close up of DISCO

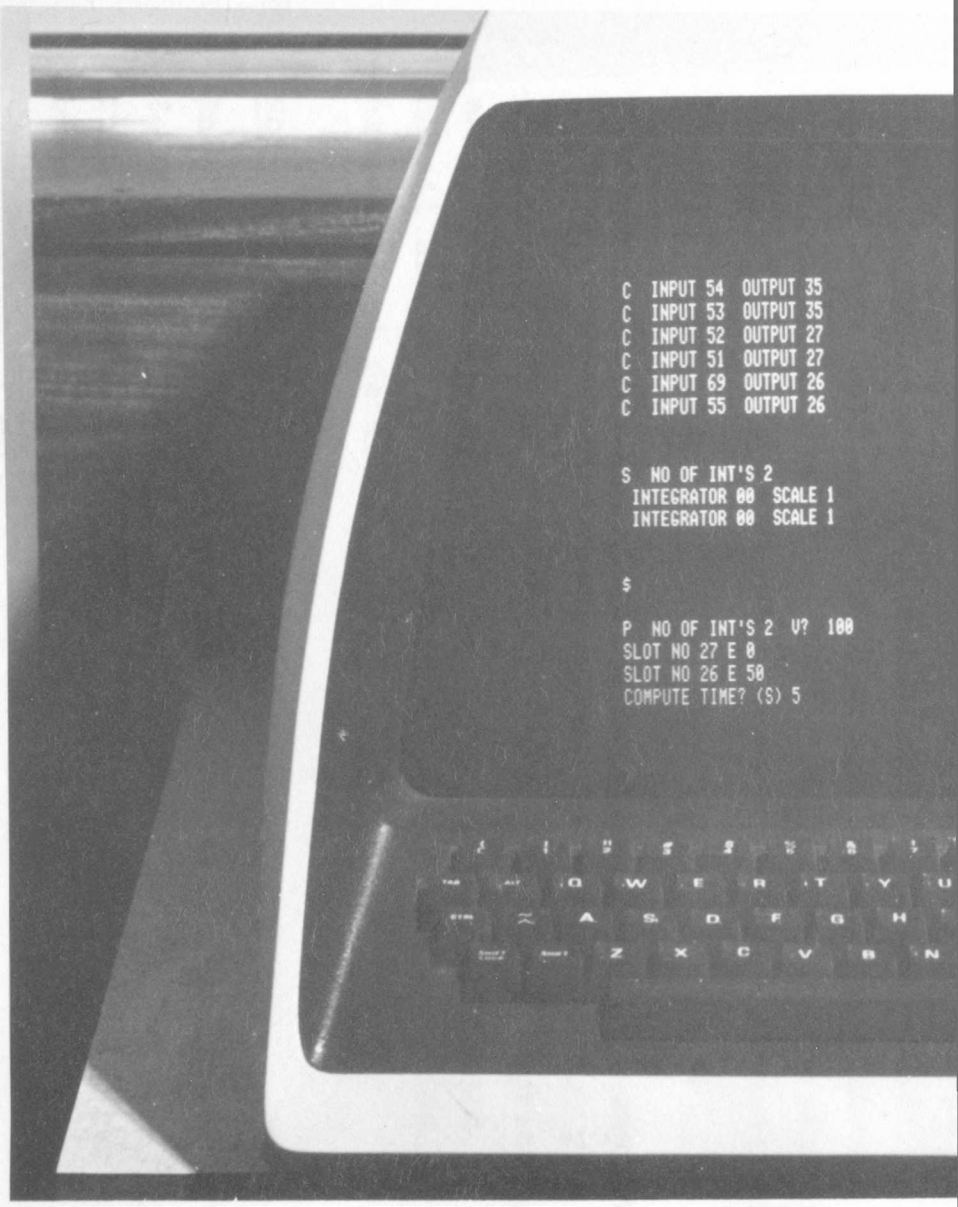
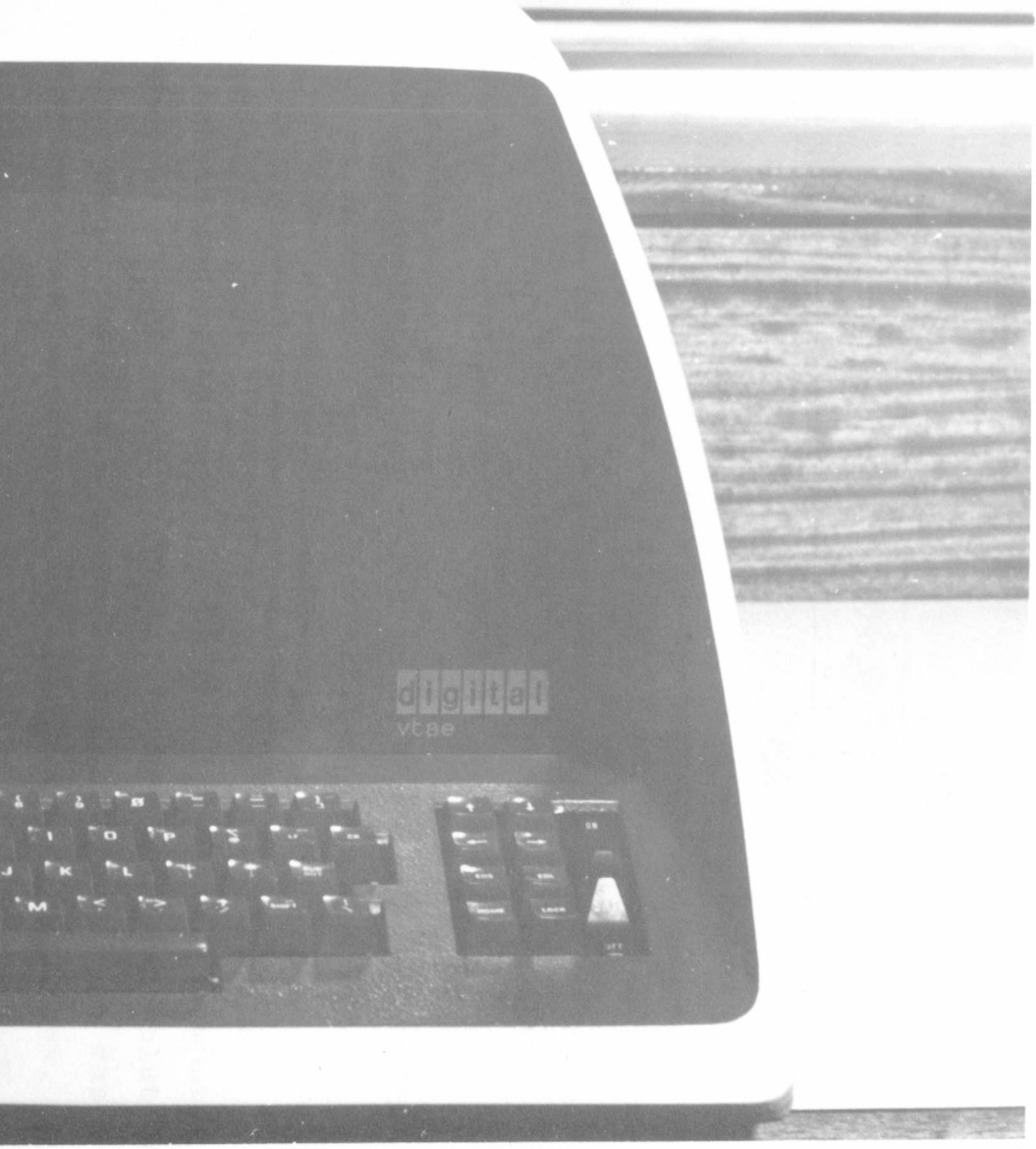

C INPUT 54 OUTPUT 35
C INPUT 53 OUTPUT 35
C INPUT 52 OUTPUT 27
C INPUT 51 OUTPUT 27
C INPUT 69 OUTPUT 26
C INPUT 55 OUTPUT 26
S NO OF INT'S 2
INTEGRATOR 00 SCALE 1
INTEGRATOR 00 SCALE 1
$
P NO OF INT'S 2 V? 100
SLOT NO 27 E 0
SLOT NO 26 E 50
COMPUTE TIME? (S) 5

PLATE 4.3



Close up of V. D. U.

CHAPTER 5

OUTPUT INTERFACE

5.1 Introduction

It is an essential requirement of a stochastic computer that it has the facility of presenting the solution of a problem in digital or analogue form. This requires the design of a stochastic to analogue (S/A) convertor and a stochastic to digital convertor, the latter being called an ADDIE (ADaptive DIgital Element).

As was previously mentioned in section 1.12 the output of an integrator may be represented by a stochastic sequence or a binary number. Thus an integrator has a digital output which serves as an ideal interface between the stochastic computer and a digital computer. However this is only practical if the required solution appears at the output of an integrator. In some cases, for example the solution of simultaneous equations, it is not possible to obtain the solution of a problem from the output of an integrator. It is therefore desirable to have a separate output interface element which can be accommodated in a modular position (see section 4.2) thus allowing the output of any element to be patched to it.

There are three ADDIE structures which have been examined and these are discussed in detail in reference 5. The simplest of these structures was mentioned in section 1.14 and is called a Noise ADDIE. This is now discussed in detail and the theoretical results will be verified by experimental data.

5.2 Noise ADDIE[5]

An ADDIE is basically an integrator with 100% negative feedback and the noise ADDIE is shown in Figure 5.1. It will be seen that a noise ADDIE is simply an integrator with /

with one input consisting of the negated stochastic output signal p(F) and the second input being the stochastic sequence to be converted, p(A). Therefore the ADDIE may be analysed in the same way as an integrator. In section 1.12 the state of the integrator counter was given as a function of the two input sequences. This is described by equation (1.2) which states

$$C(t) = C(0) + \frac{1}{N\tau_1} \int_0^t [p(UP) - p(DOWN)]\,dt$$

where C(0) is state of counter at t = 0
N = number of counter states
τ_1 = period of clock pulse.

The probabilities p(UP) and p(DOWN) are derived from the input probability p(A) and the feedback sequence which is p(F) where

$$p(F) = 1 - p(C(t)) = 1 - C(t)$$

(it was explained in section 1.13 that C(t) lies in the range 0 to 1 and hence can be substituted for the weighting of the stochastic output p(C(t))).

The UP and DOWN lines are given as

$$p(UP) = p(A)p(F)$$

$$p(DOWN) = \overline{p(A)}.\overline{p(F)} = [1 - p(A)][1 - p(F)]$$

$$p(UP) - p(DOWN) = p(A) - C(t) \qquad \text{---- (5.1)}$$

Substituting (5.1) in (1.2) we obtain

$$C(t) = C(0) + \frac{1}{N\tau_1} \int_0^t [p(A) - C(t)]\,dt$$

and differentiating both sides

$$\frac{dC(t)}{dt} = 0 + \frac{1}{N\tau_1} [p(A) - C(t)]$$

Rearranging /

Rearranging,

$$\frac{dC(t)}{dt} + \frac{1}{N\tau_1} C(t) = \frac{p(A)}{N\tau_1}$$

and taking Laplace transforms of both sides

$$(S + \frac{1}{N\tau_1})C(s) = \frac{\mathcal{L}\{p(A)\}}{N\tau_1} + C(0) \qquad \text{---- (5.2)}$$

where C(0) is the initial condition of the counter.

If we consider p(A) as a step input of step size p(A) then

$$\mathcal{L}\{p(A)\} = \frac{p(A)}{S}$$

and substituting in (5.2)

$$(S + \frac{1}{N\tau_1})C(0) = \frac{p(A)}{SN\tau_1} + C(0)$$

$$C(S) = \frac{SC(0) + p(A)/N\tau_1}{S(S + \frac{1}{N\tau_1})}$$

Expanding by partial fractions

$$C(S) = \frac{p(A)}{S} - \frac{p(A) - C(0)}{S + 1/N\tau_1}$$

and taking inverse transforms we have

$$C(t) = p(A) - (p(A) - C(0))\, e^{-\frac{t}{N\tau_1}}$$

$$= p(A)(1 - e^{-\frac{t}{N\tau_1}}) + C(0)e^{-\frac{t}{N\tau_1}} \qquad \text{---- (5.3)}$$

This solution gives an exponential response to a step input and also shows the exponential decay of the initial conditions term

ie as $t \to \infty$ $C(t) \to p(A)$

which is the required result.

To /

To verify this analysis equation (5.3) can be used to estimate the time taken for a solution to be obtained to a given accuracy.

Rearranging equation (5.3),

$$\frac{C(t)}{p(A)} = 1 - e^{-\frac{t}{N\tau_1}} + \frac{C(0)}{p(A)} e^{-\frac{t}{N\tau_1}}$$

For example the time taken to obtain a solution to within 10% is found as follows:

$$\frac{C(t)}{p(A)} = 0.9 \quad \text{(assuming } p(A) > C(0)\text{)}.$$

$$0.9 = 1 - e^{-\frac{t}{N\tau_1}} + \frac{C(0)}{p(A)} e^{-\frac{t}{N\tau_1}}$$

which can be expressed as

$$\frac{t}{\tau_1} = N \ln\left(10\left[1 - \frac{C(0)}{p(A)}\right]\right) \quad \text{where } \frac{t}{\tau_1} = \text{number of clock pulses.} \qquad \text{---- (5.4)}$$

Similarly for a 5% accuracy

$$\frac{C(t)}{p(A)} = 0.95$$

and

$$\frac{t}{\tau_1} = N\left[\ln\left(20\left[1 - \frac{C(0)}{p(A)}\right]\right]\right] \qquad \text{---- (5.5)}$$

and for a 1% accuracy

$$\frac{t}{\tau_1} = N\left[\ln 100\left[1 - \frac{C(0)}{p(A)}\right]\right] \qquad \text{---- (5.6)}$$

These estimates can be checked by measuring the time taken for the ADDIE to reach these limits.

Graph 5.1 shows the measured response of a noise ADDIE to step inputs of $p(A) = 0.25, 0.5, 0.75$ and 1 with $C(0) = 0$. These results may be used to confirm the estimates of equations (5.4), (5.5) and (5.6). A comparison of the estimated and measured numbers of clock /

clock pulses for varying accuracy and p(A) is shown in Table 5.1. From the comparison it is seen that the estimates are in good agreement with the experimental results. Equations (5.4), (5.5), and (5.6) are equally applicable to the S/A convertor discussed in the next section.

A further check on the result of equation (5.3) is made by considering the bandwidths of the ADDIE. By inspection of (5.2) the cut off frequency is given by

$$\omega_{3dB} = \frac{1}{N\tau_1}$$

For N = 4096 and τ_1 = 1 s

$$\omega_{3dB} = 244 \text{ rad/s}$$

or

$$f_{3dB} = 38.8 \text{ Hz}$$

Figure (5.2) shows the experimental configuration for measuring the frequency response of the noise ADDIE and the results are shown later in Graph 5.3 which also shows the response of the simple S/A convertor and an improved 2nd order S/A convertor.

5.3 Stochastic to Analogue Convertor

The simplest form of S/A convertor was mentioned in section 1.14 and is a simple R-C low pass filter as shown in Figure 5.3. It is easily shown that the output voltage v(t) for a step input of magnitude A is

$$v(t) = A(1 - e^{-\frac{t}{RC}}) + V_o e^{-\frac{t}{RC}} \qquad \text{---- (5.7)}$$

where V_o is V(t) at t = O.

Over /

Over a given time t the change of voltage v(t) is

$$\delta v = v(t) - V_o$$

$$= (A - V_o)(1 - e^{-\frac{t}{\tau_2}}) \qquad \text{---- (5.8)}$$

where $\tau_2 = RC$.

Consider now one single pulse of the stochastic sequence to be smoothed. If an ON pulse is present at time t = O, ie, at the beginning of the pulse then A ≒ 3v, V_o may have any value between Ov and 3v and t will be the period of one pulse, ie, $t = \tau_1$. Using these values in equation (5.8) the change in output voltage after one sample will be given in terms of the voltage after the previous sample. This will also be the case where no pulse is present, ie, A = Ov.

For a given network and clock period the term

$$(1 - e^{-\frac{t}{\tau_2}})$$

will be a constant and this will be written as K, ie,

$$K = (1 - e^{-\frac{\tau_1}{\tau_2}})$$

where τ_1 is the clock period.

Therefore equation (5.8) becomes

$$\delta v = (A - V_o)K$$

and therefore

$$v(t) = (A - V_o)K + V_o$$

$$= V_o(1 - K) + AK \qquad \text{---- (5.9)}$$

Equation (5.9) may be written in the general form

$$v_n = v_{n-1}(1 - K) + A_n K \qquad \text{---- (5.10)}$$

where /

where v_n is the output voltage after the nth sample, v_{n-1} is the output voltage after the (n-1)th sample and A_n is the value of the nth sample (either ON or OFF which corresponds to voltages of approximately 3.3v and Ov respectively).

Using equation (5.10) we have

$$v_1 = A_1K + V_O(1 - K) \qquad \text{---- (5.11)}$$

and in this case V_O is the output voltage at the beginning of the first sample, ie, the initial condition of v(t). The output voltage v_1 (and in the general case v_n) will be normalised, varying between O and 1, if the value of A_1 (and A_n) is taken as being O or 1 and not as Ov or 3.3v. After the second sample A_2 the output voltage is

$$v_1 = A_2K + V_1(1 - K)$$

$$= A_2K + A_1K(1 - K) + V_O(1 - K)$$

and extending this operation to n samples we obtain

$$v_n = V_O(1-K)^n + A_1K(1-K)^{n-1} + A_2K(1-K)^{n-2} + \ldots.A_nK$$

$$= V_O(1-K)^n + \sum_{j=O}^{n} A_{n-j}K(1-K)^n \qquad \text{---- (5.12)}$$

The summation term of equation (5.12) is in fact a generating function and this will give an unbiased estimate of p(A) if the weighting terms sum to unity ie,

$$\sum_{j=O}^{n} K(1-K)^j = 1$$

Summing all weighting terms we have

$$K(1-K)^{n-1} + K(1-K)^{n-2} + K(1-K)^{n-3} + \ldots.$$

$$= K(1 + (1-K) + (1-K)^2 + \ldots + (1-K)^{n-1})$$

and /

and using the binomial theorem

$$\sum_{j=0}^{n-1} K(1-K)^j = K(1 - (1-K))^{-1}\Big|_{n\to\infty}$$

Therefore for large n

$$\sum_{j=0}^{n-1} K(1-K)^j \doteqdot K(K)^{-1} \doteqdot 1$$

which is the required result for an unbiased estimate of p(A).

A time solution to a step input, for v_n is derived in Appendix 1 and the result approximates to

$$v(t) = V_o e^{-\frac{Kt}{\tau_1}} + p(A)(1 - e^{-\frac{Kt}{\tau_1}}) \qquad \text{---- (A1.7)}$$

As previously stated v(t) is the normalised output voltage and the true output voltage is approximately 3.3v(t).

Equation (A1.7) may be compared to equation (5.3) which gives response of the noise ADDIE to a step solution as

$$C(t) = p(A)(1 - e^{-\frac{t}{N\tau_1}}) + C(0)\, e^{-\frac{t}{N\tau_1}}$$

From this comparison it is seen that the two equations are identical if $K = 1/N$ and $C(0) \equiv V_o$, ie, the response of the S/A counter and the noise ADDIE are identical. This can be verified by examining Graph 5.2 which gives the step response of both output interfaces for the same step input with the clock frequency such that $K = \frac{1}{N}$. The step responses were recorded on a high speed ultra-violet X-Y plotter.

It /

It will now be shown that the S/A convertor is more accurate than the noise ADDIE.

5.4 Variance of Output Interfaces[5]

It has been shown[5] that the variance of a noise ADDIE is given as

$$\text{variance} = \sigma^2 = \frac{p(1-p)}{N} \qquad \text{---- (5.13)}$$

where

p = probability weighting of input sequence.

The variance of the simple R-C network will now be desired.

Consider equation (5.10) which gives the filter output after the nth sample as

$$v_n = v_{n-1}(1-K) + A_n K$$

Squaring both sides we obtain

$$(v_n)^2 = (1-K)^2 (v_{n-1})^2 + 2(1-K)A_n v_{n-1} K + A_n{}^2 K^2$$

Taking expected values and using the relationship

$$\varepsilon(A_n{}^2) = \varepsilon(A_n) = p$$

the above equation becomes

$$\varepsilon(v_n{}^2)(2-K) = p^2(2-K) - p^2 K + pK$$

$$\varepsilon(v_n{}^2)^2 - p^2 = K(p-p^2)\ \frac{1}{2-K}$$

and therefore by definition of variance

$$\sigma^2 = \frac{p(1-p)}{2/K-1} \qquad \text{---- (5.14)}$$

In the case of the S/A convertor and the noise ADDIE having identical time constants then

$$\sigma^2 = \frac{p(1-p)}{2N-1} \qquad \text{---- (5.15)}$$

By /

By comparing the above equation to equation (5.13) it is clear that for large N (and hence small k) the S/A convertor has a variance equal to half that of the noise ADDIE.

To confirm that this is the case it is necessary to transform the variance into some other more meaningful parameter. The square root of the variance is called the standard deviation of the distribution function and is easier to measure than the variance. In the case of a Binomial distribution 64% of all samples fall within the range $p-\sigma$ and $p+\sigma$ where p is the expected value or mean and σ is one standard deviation. It is therefore convenient to talk of the error of an output interface as one standard deviation. Thus the accuracy of a noise addie can be defined as

$$\varepsilon_1 = \sigma_1 = \left(\frac{p(1-p)}{N}\right)^{\frac{1}{2}} \qquad \text{---- (5.16)}$$

and the accuracy of a S/A convertor as

$$\varepsilon_2 = \sigma_2 = \left(\frac{p(1-p)}{2N-1}\right)^{\frac{1}{2}} \qquad \text{---- (5.17)}$$

Substituting (5.16) in (5.17) and approximating we obtain

$$\varepsilon_2 = \varepsilon_1/\sqrt{2} \qquad \text{---- (5.18)}$$

which shows that the accuracy of the S/A convertor is greater than that of the noise ADDIE.

Therefore a simple R-C filter with K = 1/N and an N state noise ADDIE will have the same response curves (both step and frequency) but the analogue filter has a greater accuracy, ie, the analogue filter offers greater accuracy for the same bandwidth. Alternatively, for the same accuracy, the analogue filter will have a greater /

greater bandwidth than the noise addie. In fact for the same accuracy the bandwidth of the analogue filter is approximately twice that of the noise ADDIE.

Graphs 5.4 through to 5.6 show, for varying input probabilities, the sample distributions for both the S/A convertor and the noise ADDIE. From these graphs it is seen that the error in each (one standard deviation) is in agreement with the error calculated from equations (5.16) and (5.17) for each value of input probability p. In all cases K = 1/N, ie, the bandwidths of both output interfaces is equal. Thus the superior accuracy of the R-C filter can clearly be seen from the distributions.

5.5 Practical Form of S/A Convertor

Although the results given in the previous sections have been for a simple R-C low pass filter, this configuration is not ideal and may be improved upon by using an active filter followed by a calibration stage. This is shown in Figure 5.4. The active filter is a 2nd order Butterworth filter which has the transfer characteristic

$$Av(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n + \omega_n^2}$$

where $\omega_n = 1/RC$, $\xi = \frac{3 - Av_O}{2}$ and Av_O is given as $Av_O = 1 + R_2/R_1$.

By using the same analysis used for the simple R-C network the output voltage for a step input of magnitude p(A) and damping ratio $\xi = 1$ can be shown to be approximately

$$v(t) \quad /$$

$$v(t) = p(A)(1 - (1 + K/\tau_1 t)e^{-\frac{Kt}{\tau_1}}) + V_o(1 + K/\tau_1 t)e^{-\frac{Kt}{\tau_1}} \quad \text{---- (5.19)}$$

where $K = 1 - e^{-\omega_n \tau_1}$ and τ_1 = clock period.

The frequency response of this S/A convertor is shown in Graph 5.3 and is compared with the noise ADDIE and simple R-C filter frequency response. From this graph the slope of the S/A convertor response is seen to be 40 dB/decade at frequencies greater than ω_n. This leads to a reduction of the noise components of the output voltage, especially at the clock frequency. The value of the damping ratio ξ was made less than unity giving rise to the expected curvature of the response at the cut off frequency.

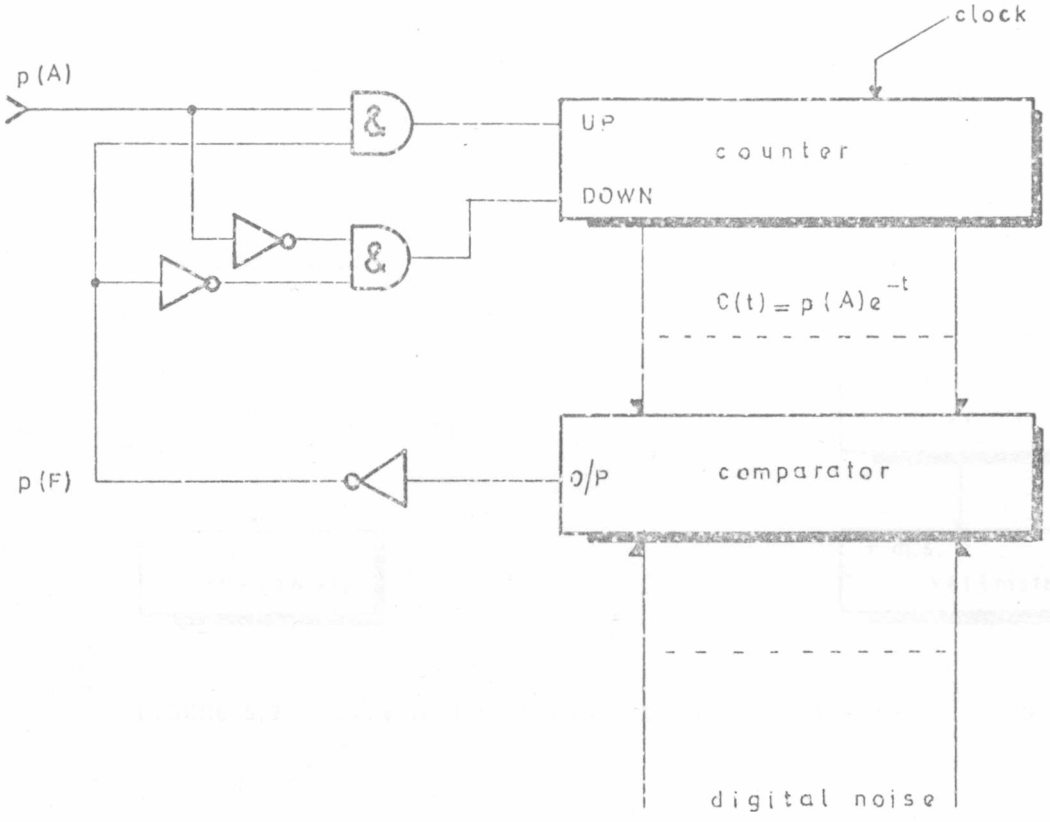


FIGURE 5.1 Noise ADDIE

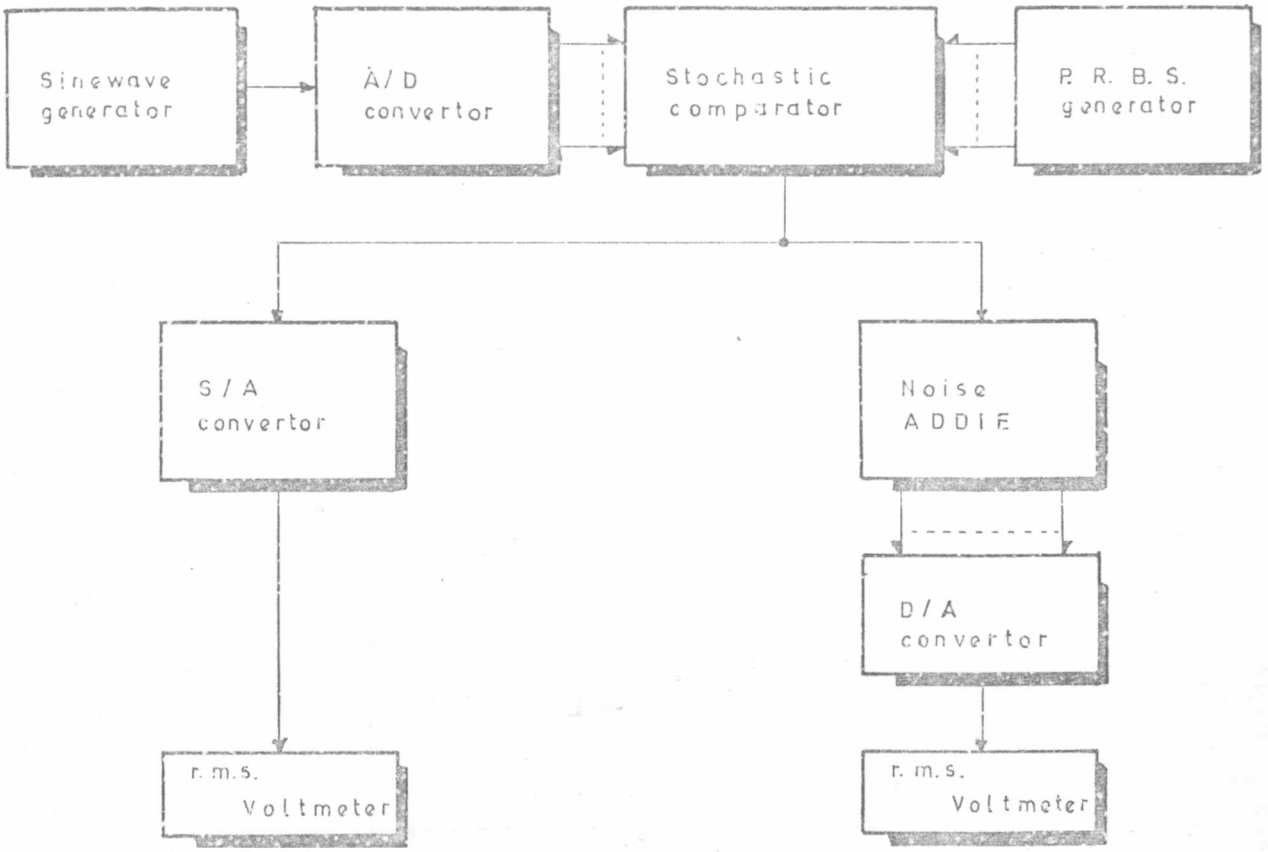


FIGURE 5.2 Scheme for measuring Output Interface Frequency Response

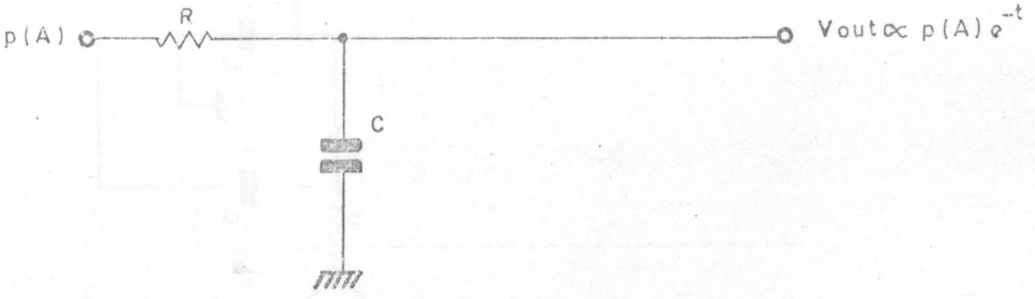


FIGURE 5.3 Simple R-C Low Pass Filter

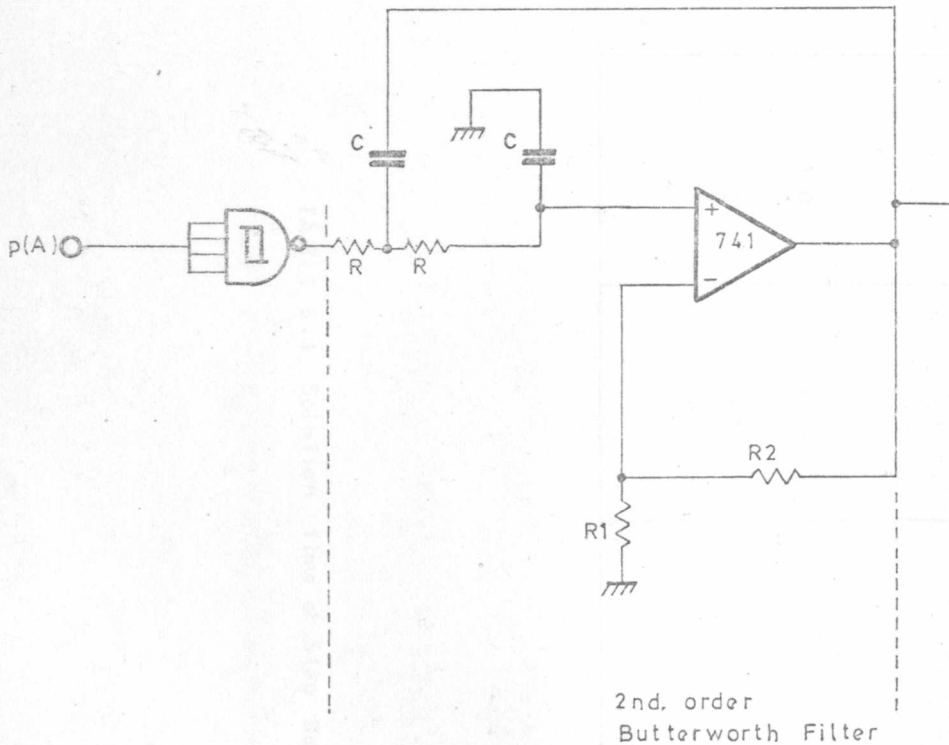
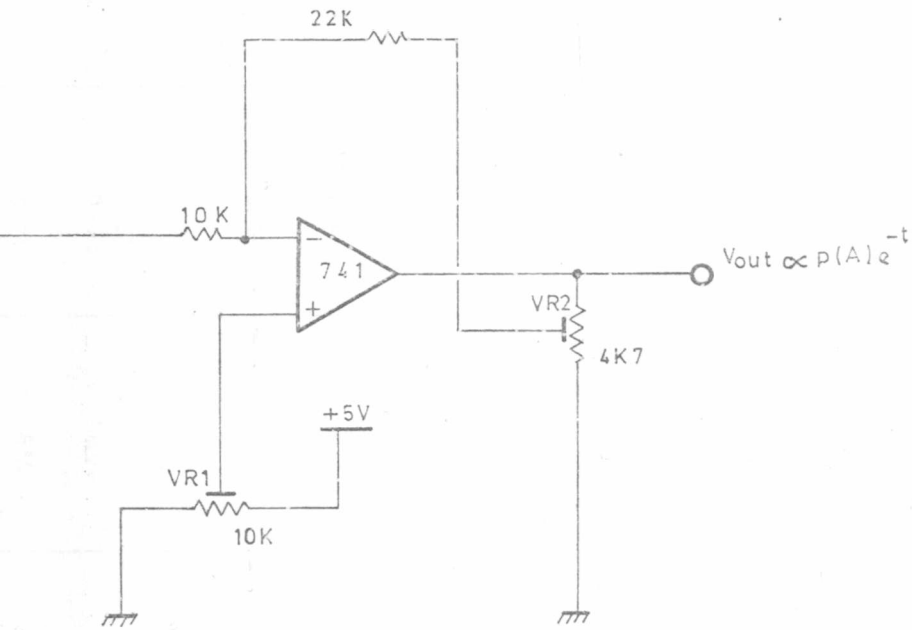


FIGURE 5·4 Second

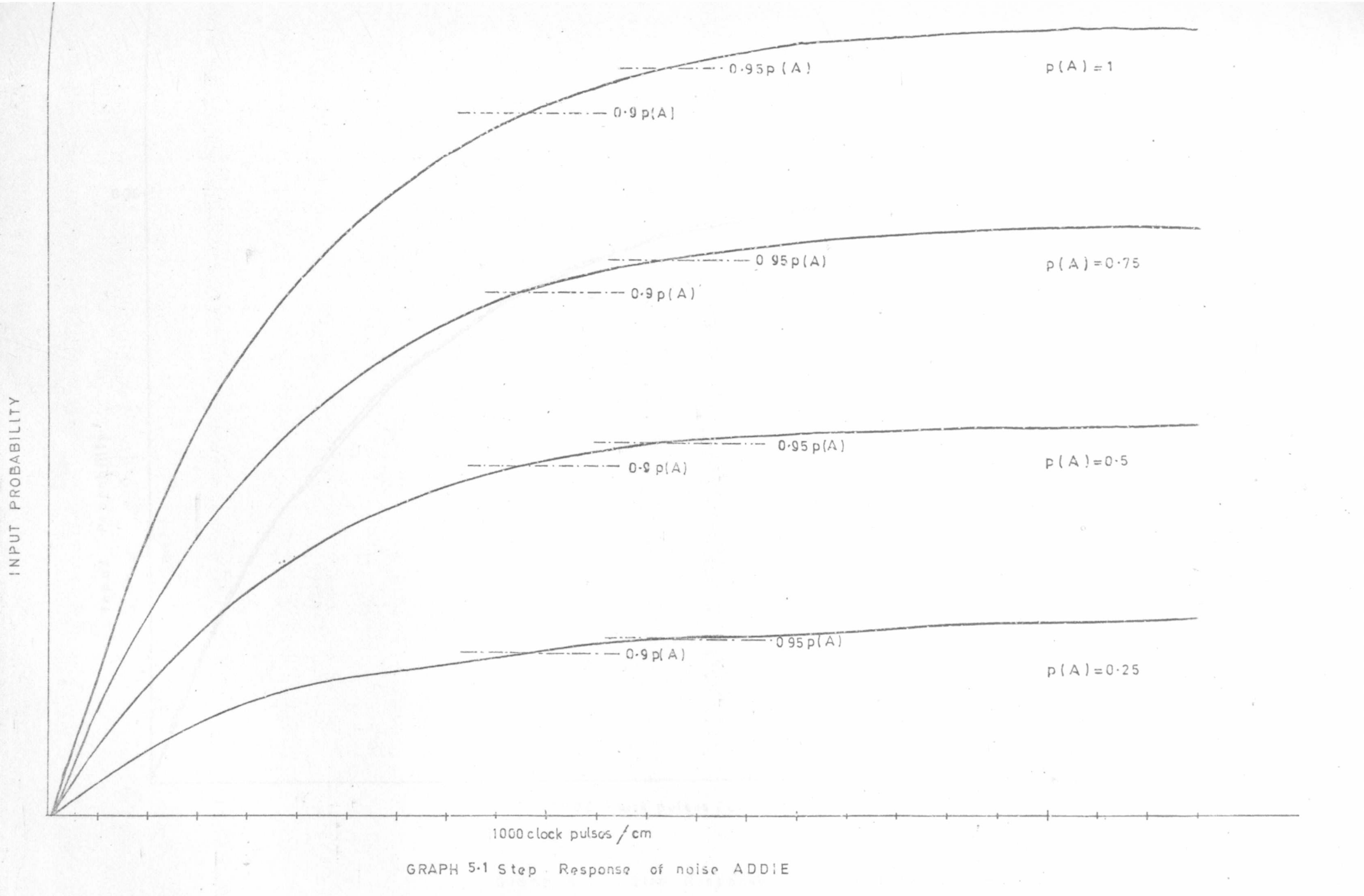


Calibration and
level shifting

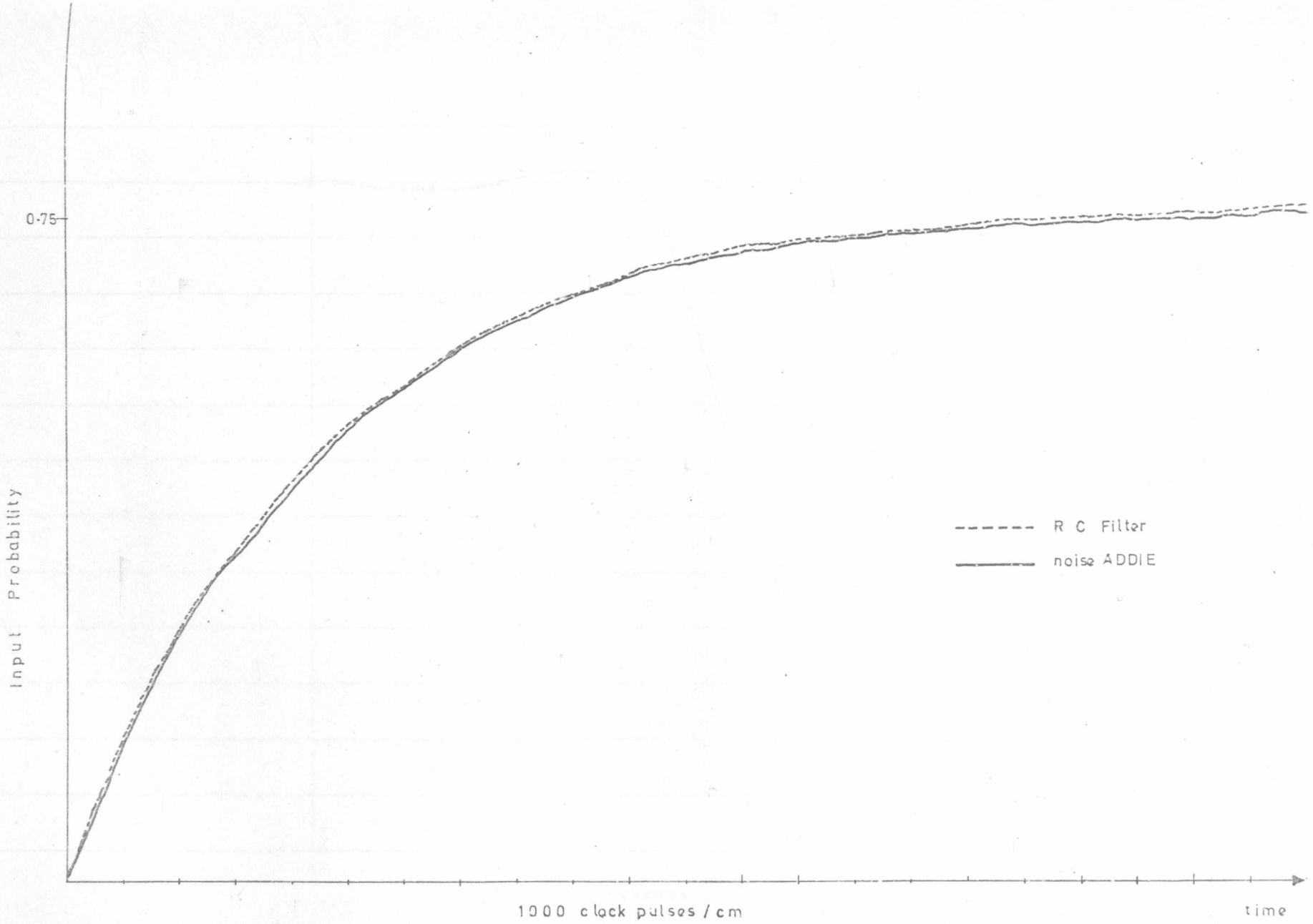
Order S/A Convertor

STEP SIZE p(A)	ACCURACY	TIME TAKEN (in clock periods) Theoretical	Measured
0·25	10%	9400	9500
0·25	5%	12300	12400
0·5	10%	9400	9600
0·5	5%	12300	12300
0·75	10%	9400	9500
0·75	5%	12300	12400
1·0	10%	9400	9600
1·0	5%	12300	12400

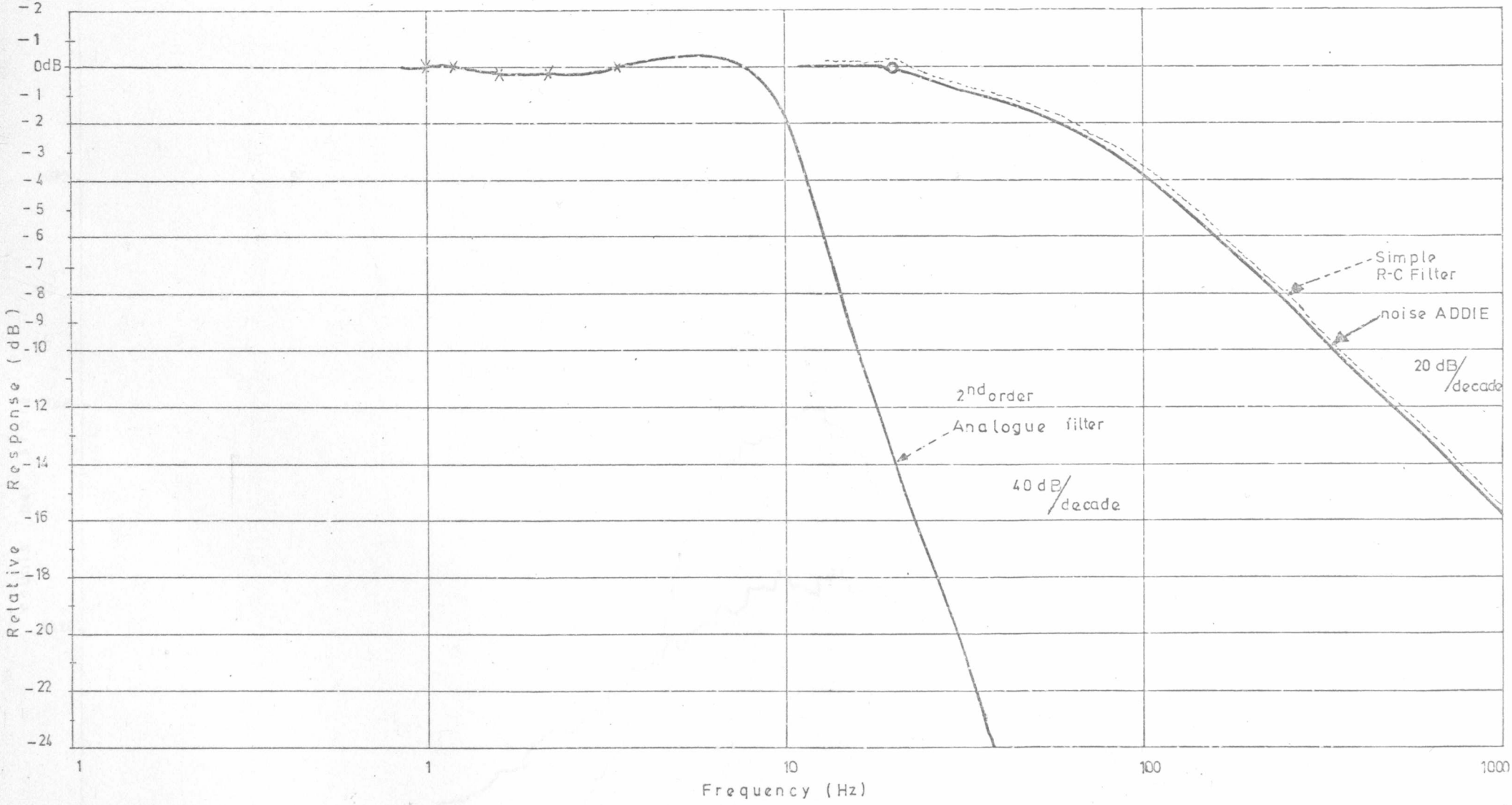
TABLE 5.1 Solution Time of Step Response to a given Accuracy



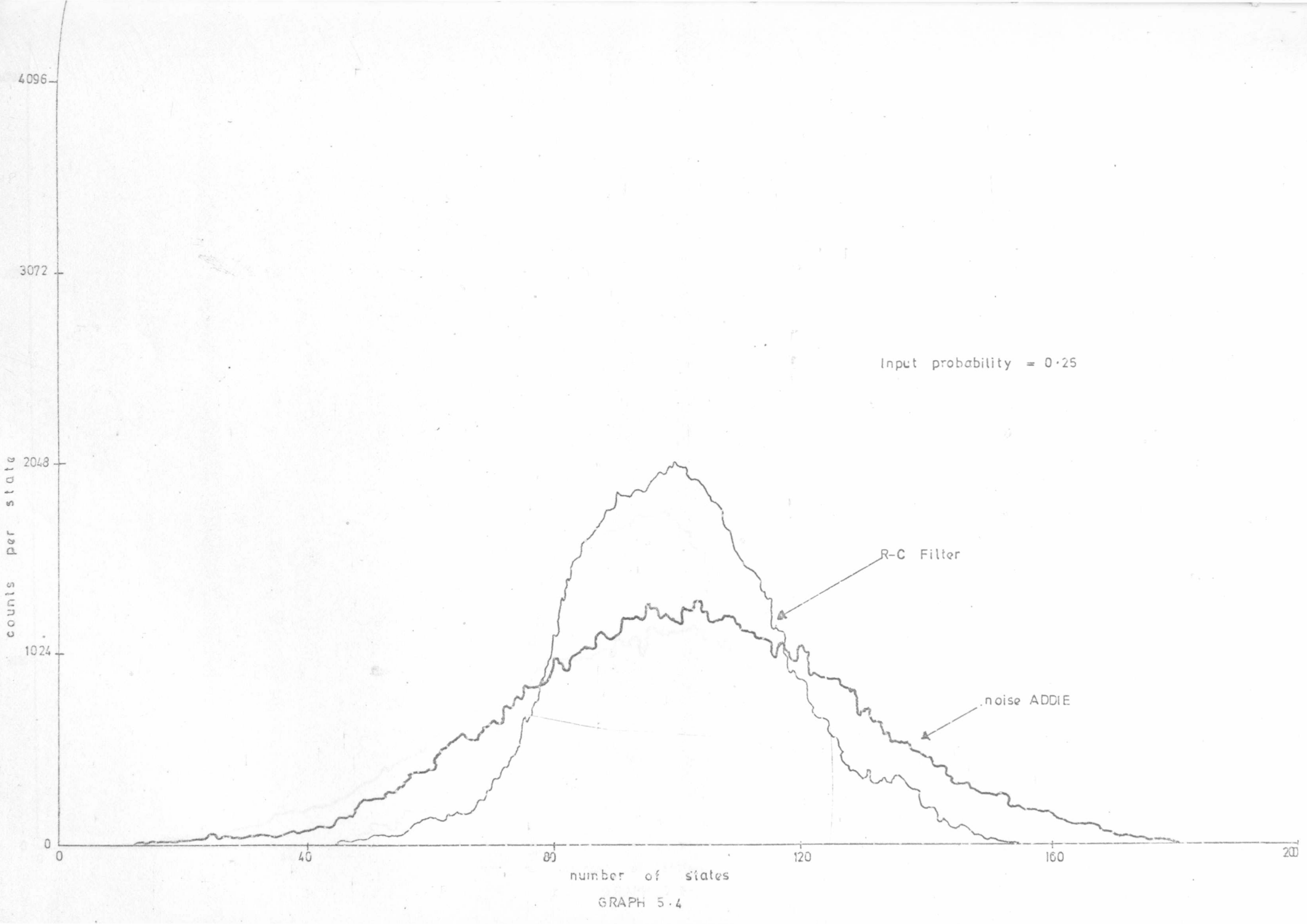
GRAPH 5·1 Step Response of noise ADDIE



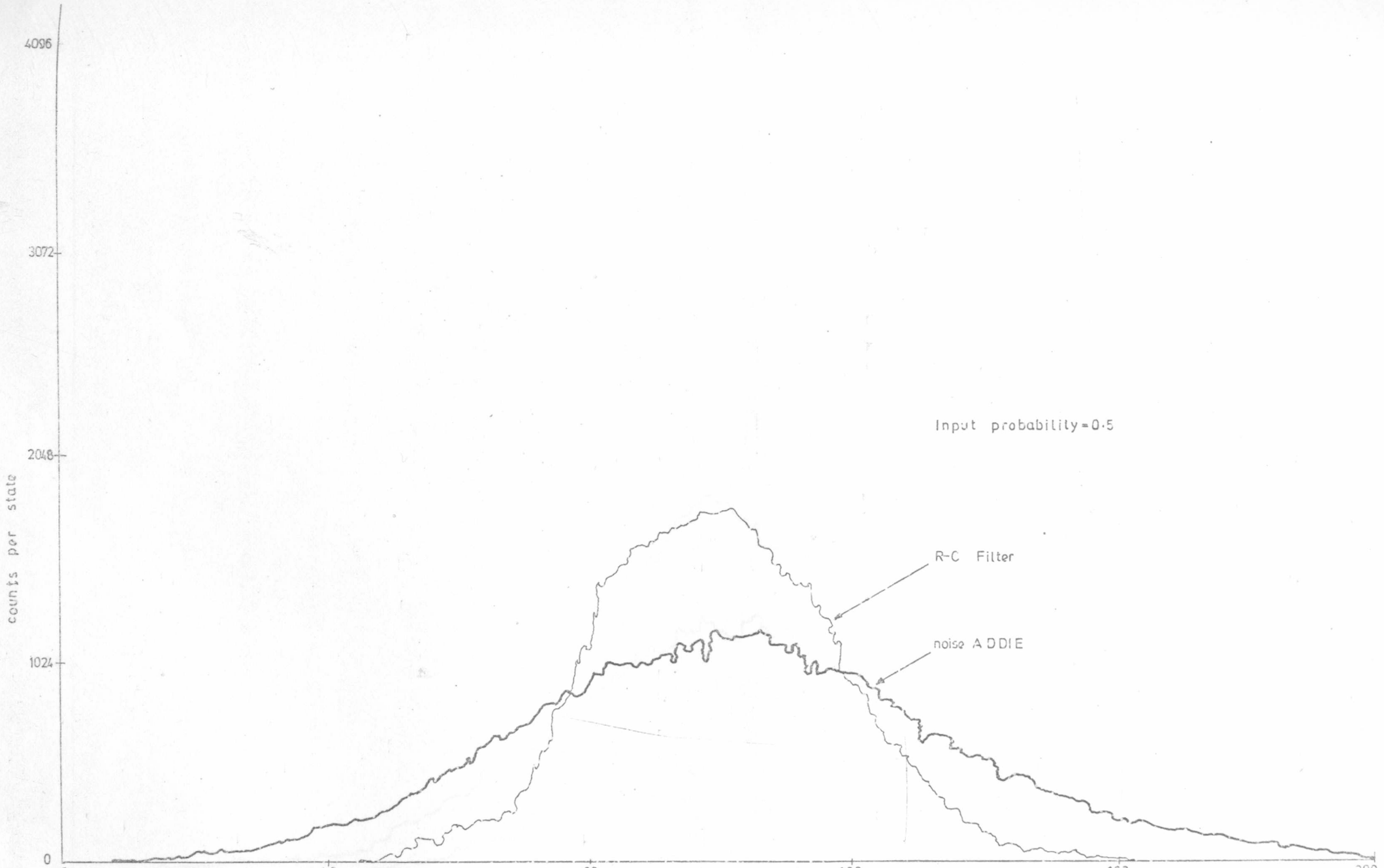
Graph 5·2 Step Response of S/A and noise ADDIE



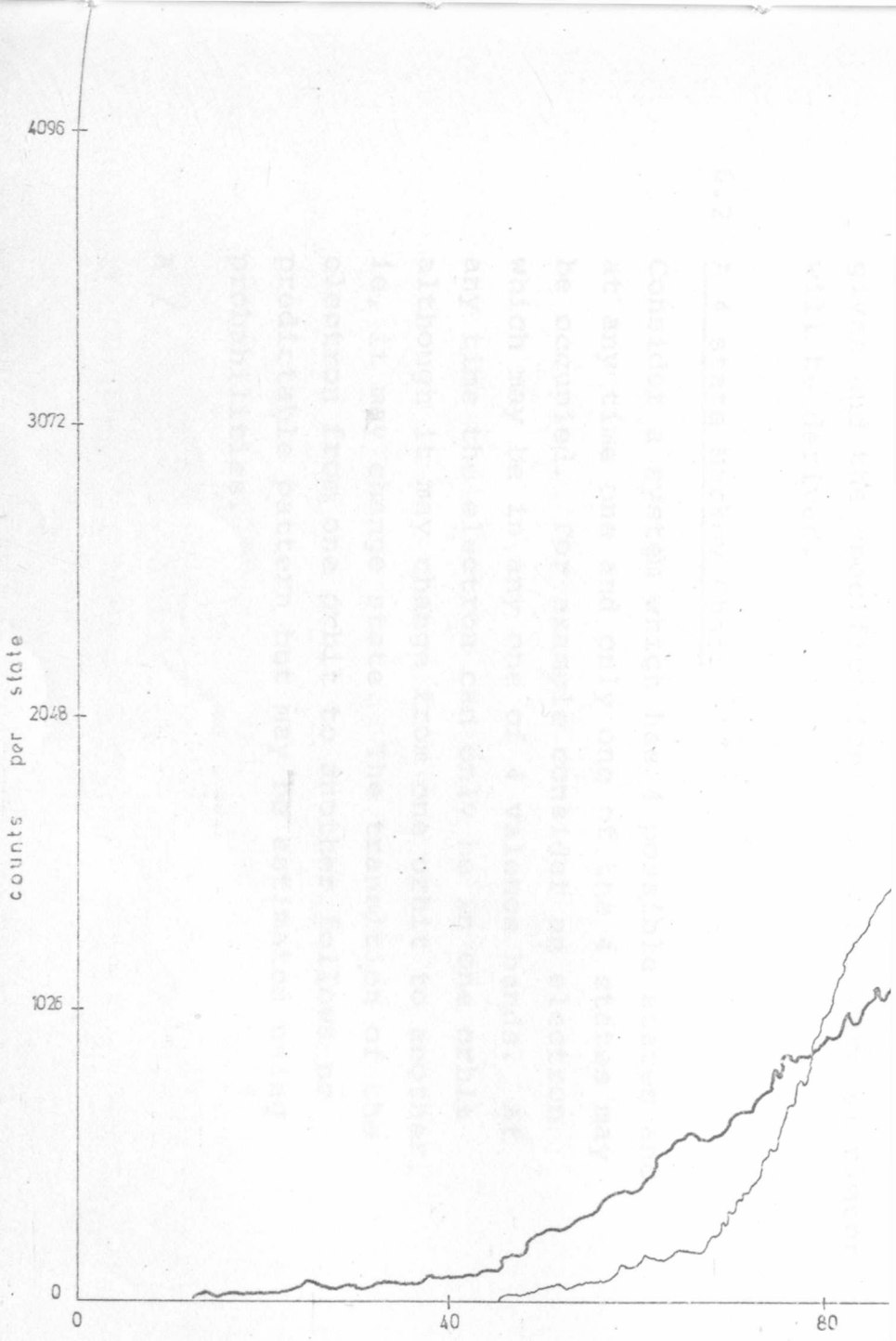
Graph 5·3 Frequency Response of Output Interface


4096
3072
2048
1024
0
counts per state
Input probability = 0·25
R-C Filter
noise ADDIE
0
40
80
120
160
200
number of states

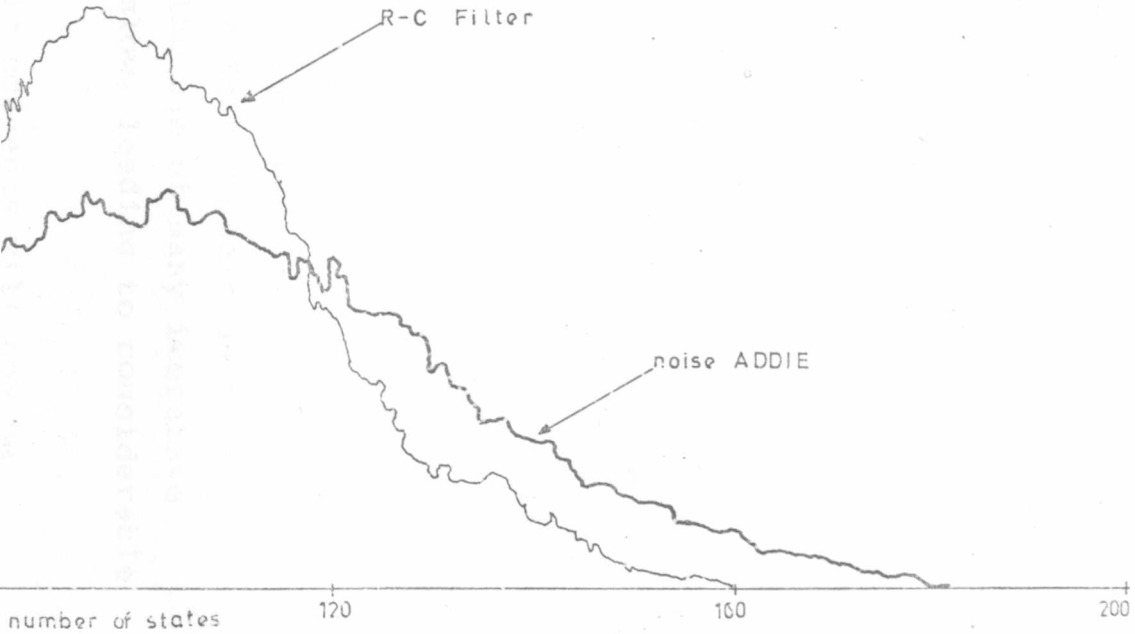
GRAPH 5·4


4096
3072
2048
1024
0
counts per state
Input probability = 0·5
R-C Filter
noise ADDIE
0
40
80
120
160
200
number of states

GRAPH 5·5


4096
3072
2048
1026
0
counts per state
0
40
80

Input probability = 0·75



GRAPH 5·6

CHAPTER 6

DESIGN AND OPERATION OF A MARKOV CHAIN SIMULATOR

6.1 Introduction to Markov Chains

Stochastic models are being used to an ever increasing extent by those who wish to investigate phenomena that are essentially concerned with a flow of events in time, especially those exhibiting such highly variable characteristics as birth, death, queueing, evolution, etc. One such stochastic model is that of Markov Chains which have been used extensively in the field of operations research for many years. While Markov Chains have been applied successfully to many areas in operational research, no high speed simulation models exist. To simulate a Markov Chain using a digital computer requires the use of many iterative procedures which are very slow, leading to considerable solution times.

An introduction to the basic concepts will now be given and the specifications for a hardware simulator will be derived.

6.2 A 4 state Markov Chain[8,10]

Consider a system which has 4 possible states and at any time one and only one of the 4 states may be occupied. For example consider an electron which may be in any one of 4 valence bands. At any time the electron can only be in one orbit although it may change from one orbit to another, ie, it may change state. The transition of the electron from one orbit to another follows no predictable pattern but may be estimated using probabilities.

A /

A transition probability is defined as being the probability of, starting in state i, being in state j at the next sample and is written P_{ij}. Each state has four associated transition probabilities, one to each of the other three states and a fourth, of remaining in the same state. Because there are only four possibilities then the sum of the four transition probabilities must be unity since we are dealing with probabilities. This is shown in the form of a state transition diagram in Figure 6.1.

At this point the four states will be designated s_1, s_2, s_3 and s_4 and the probabilities of being in each state as q_1, q_2, q_3 and q_4.

We can write the probability of being in the nth state after a transition as

$$q_n = q_1P_{1n} + q_2P_{2n} + q_3P_{3n} + q_4P_{4n}$$

where P_{1n}, P_{2n}, P_{3n} and P_{4n} are the probabilities of the four transitions to that state.

This may be written for each state giving the four equations

$$q_1 = q_1P_{11} + q_2P_{21} + q_3P_{31} + q_4P_{41}$$

$$q_2 = q_1P_{12} + q_2P_{22} + q_3P_{32} + q_4P_{42}$$

$$q_3 = q_1P_{13} + q_2P_{23} + q_3P_{33} + q_4P_{43}$$

$$q_4 = q_1P_{14} + q_2P_{24} + q_3P_{34} + q_4P_{44}$$

which may be rewritten in matrix form giving

$$\begin{bmatrix} q_1^{t+1}, q_2, q_3, q_4 \end{bmatrix} = \begin{bmatrix} q_1^{t}, q_2, q_3, q_4 \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix}$$

where /

where the square matrix is known as a stochastic matrix.

This represents to operation of a four state Markov chain and may be written in the form

$$\bar{Q}_{t+1} = \bar{Q}_t . S \qquad \text{---- (6.1)}$$

where $\bar{Q}_{t+1}$ and $\bar{Q}_t$ are the row vectors and S is the stochastic matrix. If the starting state is $\bar{Q}_O$ then after n transitions equation (6.1) becomes

$$\bar{Q}_O \cdot S^n$$

By evaluating the matrix $\bar{Q}_O S^n$ the probabilities of being in each state after n transitions is found. For large n this is clearly a tedious process and a hardware simulator would be a considerable advantage. The requirements for a simulator are:

1. a four state sequential network with programmable inputs and state detection;
2. the facility to program the 16 probabilities of the stochastic matrix;
3. a programmable counter which will determine the number of transitions n;
4. the ability to estimate the probability of being in any one state after n transition periods.

6.3 System Design

Of the above specifications only the second exists within the framework of the stochastic computer, ie, comparators. The first requirement is met by using a two stage sequential network consisting of two flip-flops and appropriate combinational logic. This means that one change of state or transition will occur /

occur at every clock pulse, ie, one matrix multiplication will be accomplished by each clock pulse. The components of the stochastic matrix are generated by stochastic comparators, each one giving a stochastic sequence with a variable probability of delivering an ON pulse. This means that for the duration of any clock period the stochastic matrix will be composed of 'ones' and 'zeros' with no intermediate values possible. Intermediate values are in fact represented by the probabilities of finding a 'one' in each matrix position. Therefore, because each row in the matrix must sum to unity, there can only be one 'one' in each row and so probability transformers must be used to ensure that this is the case. Secondly it is meaningless to have a non-zero value for any component of a row other than that row which corresponds to the state occupied at the time of the clock pulse. This would mean that the system would be required to vacate a state which it does not occupy at the time of the clock pulse. Because of these last two conditions only one of the transition probabilities can be 'one' at any time.

There are 16 transition probabilities in the stochastic matrix but because each row must sum to unity then it is only necessary to generate 12 probabilities (3 per row). The transition probabilities not generated are P_{11}, P_{22}, P_{33} and P_{44} which are taken to be 'one' when the other probabilities in the appropriate row are zero. In practice this means that logic '0's are applied to the J and K inputs of the sequential network which means that no change will occur.

The truth table for the sequential network is given in Table 6.1 for each possible value of the 12 generated transition probabilities. From this truth table we can find the expressions for the control inputs. /

inputs. These are

$$J2 = P_{13} + P_{14} + P_{23} + P_{24}$$

$$K2 = P_{31} + P_{32} + P_{41} + P_{42}$$

$$J1 = P_{12} + P_{14} + P_{32} + P_{34}$$

$$K1 = P_{21} + P_{23} + P_{41} + P_{43}$$

Figure 6.2 shows the hardware implementation of the four state system and the state assignment adopted is

$$S_1 = \overline{Q2} \cdot \overline{Q1}$$

$$S_2 = \overline{Q2} \cdot Q1$$

$$S_3 = Q2 \cdot \overline{Q1}$$

$$S_4 = Q2 \cdot Q1 \quad .$$

The probabilities P_{11}, P_{12}, P_{13}, ... P_{44} are the transition probabilities of the stochastic matrix. These are derived from the comparator outputs C_1, C_2, ..., C_{12} in such a way as to ensure that only one of the transition probabilities is a 'one' at any clock pulse. The outputs of the state detection circuitry ensure that only the transition probabilities associated with the occupied state have the possibility of being high.

Switches S1 and S2 are centre-off toggle switches which are used to set the starting state.

The probabilities C_1, C_2, ..., C_{12} are the outputs from the 12 programmable stochastic comparators and can be calculated in the following manner. Consider P_{12}, P_{13}, P_{14}, C_1, C_2 and C_3.

From /

From Figure 6.2 it is seen that the following is true:

$$P_{12} = C_1(1 - C_2)(1 - C_3)$$

$$P_{13} = C_2(1 - C_3) \quad \text{and}$$

$$P_{14} = C_3$$

since $\overline{C_2} = 1 - C_2$ and $\overline{C_3} = 1 - C_3$

Therefore,

$$C_2 = \frac{P_{13}}{(1 - C_4)}$$

but since $C_3 = P_{14}$ and $P_{11} + P_{12} + P_{13} + P_{14} = 1$ then

$$C_2 = \frac{P_{13}}{P_{11} + P_{12} + P_{13}} .$$

Also,

$$C_1 = \frac{P_{12}}{(1 - C_2)(1 - C_3)}$$

$$= \frac{P_{12}}{(1 - \frac{P_{13}}{1 - C_3})(1 - C_3)}$$

$$= \frac{P_{12}}{1 - C_3 - P_{13}}$$

therefore,

$$C_1 = \frac{P_{12}}{P_{11} + P_{12}}$$

Probabilities $C_4, C_5, \ldots, C_{12}$ are found in the same way. Thus the values of comparator output probabilities can be found in terms of transition probabilities of the stochastic matrix and the actual relationships are shown in Table 6.2.

Using /

Using this circuitry it is now possible to simulate a four state Markov Chain with the stochastic comparators being programmed to give the appropriate values of stochastic sequences. The system may be clocked any number of times and this will simulate a possible sequence of state occupation. However if this simulation were to be repeated using the same starting state then it is unlikely if the same sequence of state occupation would be observed. It is therefore necessary to evolve additional circuitry which will give the probability of being in any state after any number of clock pulses.

Firstly it is necessary to have the facility of delivering any preset number of clock pulses to the system. This is achieved using the system shown in Figure 6.3.

The required number of clock pulses is set by means of thumbwheel switches and this may vary between 1 and 9999 inclusive. Switch S2 provides the choice of having a continuous clock or operation in the programme mode. In the programme mode the clock sequence is initiated by S1, the output of its associated contact bounce eliminator being used to trigger a monostable. This is necessary to ensure that the initialising pulse has a shorter duration than one clock period otherwise the sequence of clock pulses delivered may be completed before the pulse has been removed. As a result another cycle of clock pulses would be given which would lead to an error. The output of the monostable presets a flip-flop and clears the four decade counter and the output of the flip-flop enables the master clock (C_m) to clock the four digit decade counter. Because the system clock is the same clock as is applied to the counters, the contents of the counter will represent the number of clock pulses delivered to the system. /

system. When the preset number of clock pulses have been delivered, the contents of the counter and the outputs of the switches will be the same. This will result in an output from the comparator which is used to clear the flip-flop thus disenabling the clock output and sequence is complete. Although the Markov Chain simulator may be clocked any number of times required this still does not give the probability of being in a given state after n clock pulses.

To find this, the initial state must be set and the sequence initiated with the state occupied after n clock pulses being recorded. Then the initial state must be reset and the clock sequence repeated with the final state again being recorded. This must be repeated a number of times until an estimate of the probability of being in a given state can be obtained. To find the probability of being in a given state the number of times the simulator ended in this state would be divided by the total number of runs.

This procedure would clearly be time consuming if it were to be performed manually hence it must be made fully automatic. The following requirements are necessary for automatic operation.

Firstly the initial conditions must be reset automatically at the beginning of each cycle of clock pulses. Secondly at the end of each cycle the contents of each state must be read and used to estimate the probability of being in that state. Finally the period between runs must be greater than the maximum number of clock pulses per cycle multiplied by one clock period, ie, 10^4-1 clock periods. The hardware implementation of these requirements is shown in Figures 6.4 and 6.5.

Figure /

Figure 6.4 shows the method of resetting the initial state. The initial state is set by means of switches S1 and S2 and this information is entered into flip-flops FF1, FF1', FF2 and FF2'. Flip-flops FF1' and FF2' serve as a memory for the initial state. On each subsequent cycle or run, the information contained within this memory must be transferred to the sequential network consisting of FF1 and FF2. This is done upon receiving a pulse from the monostable within the programmable clock pulse generator (Figure 6.3), this pulse being delivered whenever a cycle is initiated.

Now the requirement for the automatic initiation of a cycle must be implemented. As seen from Figure 6.5, this is accomplished by dividing the mastercclock frequency by 10^4 which means that one pulse will be delivered for each 10^4 master clock pulses. The figure of 10^4 was chosen because it is possible to programme a sequence of 10^4-1 clock pulses, ie, the period between cycles must be greater than the maximum period of one cycle. This single pulse is fed, via an open collector output to point A in Figure 6.3 which means that one cycle will be initiated for each 10^4 master clock periods.

The condition of the system after the prescribed number of clock pulses (one cycle) must be examined and the result used to estimate the probability of being in any one state. Figure 6.5 shows that FF1 is cleared on each 'high' half cycle of the master clock and the Q output will therefore be low until the flip-flop is preset. This occurs when the output of the comparator of Figure 6.3 is high, ie, when the preset number of clock pulses per cycle have been delivered to the simulator which coincides with the rising edge of the master clock. Thus the Q output of FF1 is high for one half clock period (FF1 merely serves to increase the period of the pulse from the comparator). The /

The output of FF1 is used to clock FF2 to FF5 which are used to store the condition of each state after each cycle. Therefore the outputs of flip-flops 2 to 5 are stochastic sequences with a clock rate of 10^{-4} times the master clock frequency. Each stochastic sequence represents the probability of being in each state after n clock pulses. Four S/A convertors are used to convert each of the four sequences into a voltage which is proportional to their weighting. Only one of the flip-flops FF2 to FF5 and S/A convertors is shown in Figure 6.5, the other three being identical. The final stage of each convertor is an amplifier and level shifter which is necessary to convert the TTL levels of 0.2v (logic 0) and 3.3v (logic 1) to 0v and 1v respectively. Because the final stage is an inverting amplifier the inverted outputs of flip-flops FF2 to FF5 are used to compensate. The time constant of the S/A convertor must be as large as possible because the stochastic sequence will have a pulse period in the region of milli-seconds. From Chapter Five it is clear that the time constant must be in the region of 1 second for 1% accuracy.

To summarise, the operation of the system will now be described using the block diagram of Figure 6.6.

The programmable pulse generator will deliver a preset number of clock pulses for each 10^4 master clock pulses, this being output B. Upon reception of a pulse from output A the initial state of the sequential network is loaded from the memory. This pulse is delivered immediately preceding the first clock pulse of each cycle.

After /

After the required number of clock pulses have been delivered the synchronous network will be in its final state. At this point, a pulse is received by the state sampling network resulting in the final state being stored by this network.

The system stays in this condition until the next start pulse is received from the 'divide by 10^4' circuit hence the cycle will be repeated after each 10^4 clock pulses. After about 10^3 cycles the voltage outputs of the S/A convertors will represent the required probabilities to within 1% for a time constant of 1 second for the S/A convertors (see equation (5.6) with N = 1/K). Thus the probability of being in any state can be estimated to within 1% in 1 second for a master clock frequency of 10 MHz.

6.4 Examples of A Four State Markov Chain

Some experimental results will now be given to demonstrate the capabilities of the simulator. The theoretical results were calculated using program 1 which is listed in Appendix 2. This program simply multiplies the initial state vector by the stochastic matrix to give the probability of being in each state after one transition (clock pulse). The resulting state vector is multiplied by the stochastic matrix giving the probabilities of being in any state after two transitions. This process is repeated until a steady state condition is reached, usually less than fifteen transitions.

As mentioned in Chapter 1 the variance of a stochastic sequence is greatest at p(ON) = 0.5. Therefore the first example of a four state Markov Chain will have a stochastic matrix such that all comparator output probabilities are 0.5 which will be the worst case for errors in the simulator. With all 12 comparators programmed to generate probabilities of 0.5 the

resulting /

resulting stochastic matrix is

$$S = \begin{bmatrix} .125 & .125 & .25 & .5 \\ .125 & .125 & .25 & .5 \\ .125 & .25 & .125 & .5 \\ .125 & .25 & .5 & .125 \end{bmatrix}$$

The experimental results for this matrix are shown in Graphs 6.1 to 6.4 and are compared to the predicted results. Each graph gives the results for different starting states. It is interesting to note that the steady state values of the probabilities are independent of the starting states and only the transient behaviour will vary.

It must be mentioned at this point that although the graphs are shown as continuous curves, these curves are included only as guide lines to the response of the simulator. This is because the probability of being in any state is not defined for non integer values of clock pulses, ie, the response is discrete and not continuous.

As shown by the graphs the Markov Chain Simulator gives an accurate estimation of the probability of being in any state after n clock pulses even at the worst case of variance in the driving probabilities.

In practice the probabilities generated by the comparators will be calculated from the components of the stochastic matrix and not the reverse as was the case in the above example. The second example[9] will illustrate the method of solving a four state Markov Chain and will give a physical interpretation of the concept of Markov Chains.

In /

In Washington, D.C. taxicab fares are based on zones arranged in a pattern of concentric circles. A taxi may start the day in one zone and the zone of destination of the first passenger then determines the zone in which the driver cruises for his next fare and so on.

This process may be modelled as a Markov chain if there are four fare zones, S_1, the centre zone, S_2, S_3 and the outer zone S_4. The transition probabilities between zones are assumed to be stationary over time and the stochastic matrix is given as

$$S = \begin{bmatrix} 0.8 & 0.14 & 0.05 & 0.01 \\ 0.6 & 0.2 & 0.18 & 0.02 \\ 0.5 & 0.4 & 0.05 & 0.05 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{bmatrix}$$

Graphs 6.5 to 6.8 show the probabilities of being in each zone after n fares, each graph having a different starting zone. Using this simulation model the following questions may be answered:

(a) If a taxicab driver lives in the centre zone (S_1) and starts the day in the outer zone (S_4) what is the probability of being in his hom e zone after four fares?

(b) If the driver starts in his home zone what is the probability of his returning to his home zone after two fares?

(c) The driver usually stops for the morning after he has driven ten passengers. What is the probability of being in his home zone and thus not have far to drive to reach his house?

The /

The solutions to the above questions may be obtained from the graphs 6.5 to 6.8. The solutions to each question are as follows:

(a) After four fares, ie, transitions or clock pulses, and starting state S_4 the probability of being in state S_1 is 0.73 which compares well with the theoretical value of 0.725.

(b) The probability of being in zone S_1 after two transitions is 0.76 which again compares well with 0.752.

(c) For this problem it is noticed that the probability of being in any zone after ten fares is independent of the starting zone and in the case of q_1 (the probability of being in zone S_1) the value is 0.74. Once again this compares well with the calculated value of 0.734.

One interesting feature evident from the graphs is that as the zones move out from the city centre then the probability of a taxicab being in that zone decreases. For example, the probability of being in the centre zone (q_1) is 0.74 and the probability of being in the outer zone is 0.02. A physical interpretation of this feature is that if the driver lives in zone S_1 then he will on average spend 74% of his working day in his home zone. Another implication of this feature is that on average 74% of all taxicabs in Washington are to be found in the city centre (zone S_1), 18% in zone S_2, 9% in zone S_3 and 2% in the outer zone.

The above examples show that the Markov Chain simulator performs very well and gives results accurate to within 1% full scale. However the circuitry described in this chapter forms only the basis of a Markov Chain simulator, some relatively simple circuitry being required for a powerful, fully comprehensive Markov Chain system. This is /

This is discussed under future developments in Chapter 8.

Finally Plate 6.1 shows the Markov Chain simulator in close-up with the random walk simulator, which is discussed in the next chapter, and the master clock generator.

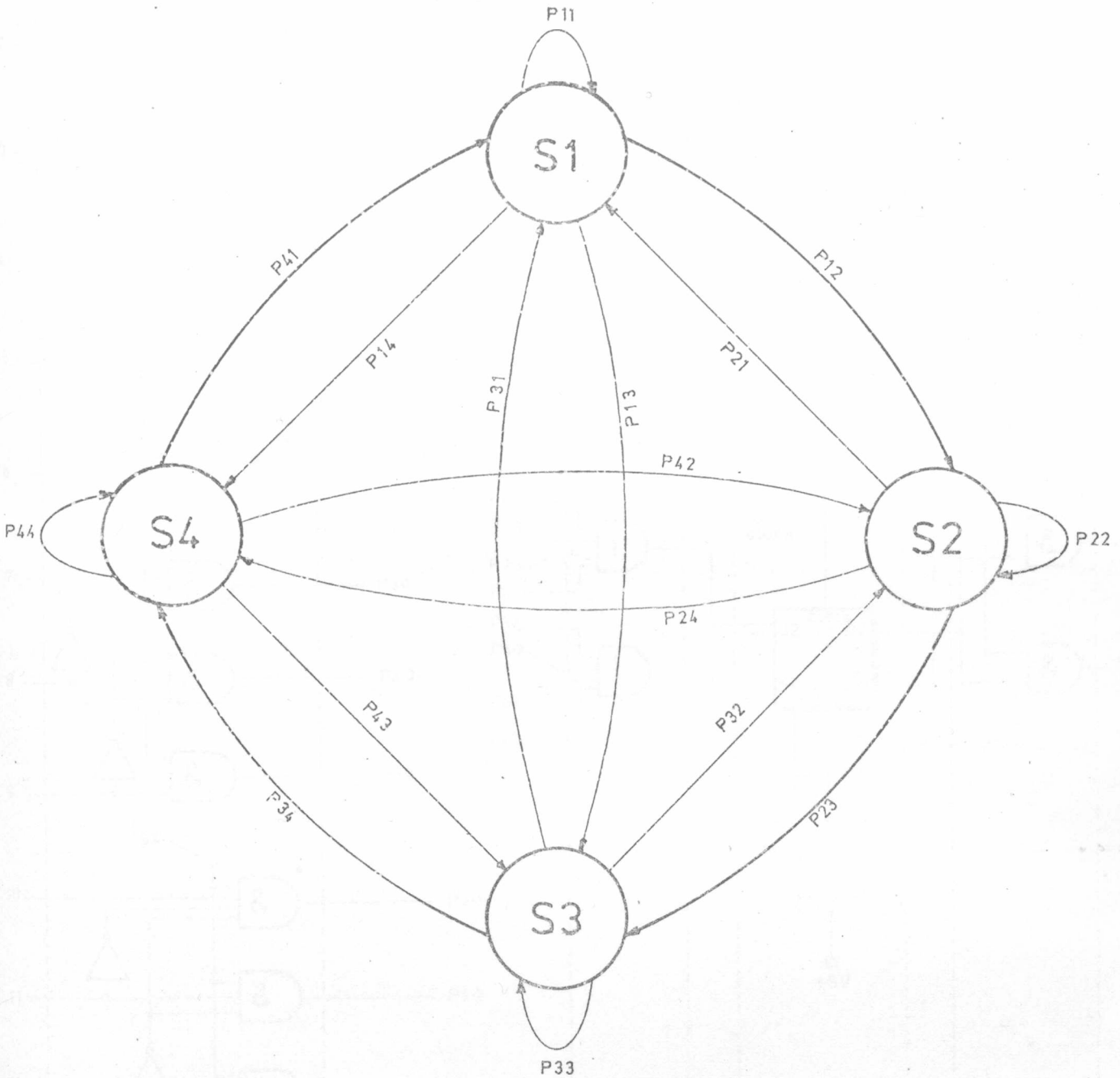


FIGURE 6.1 Markov Chain State Transition diagram

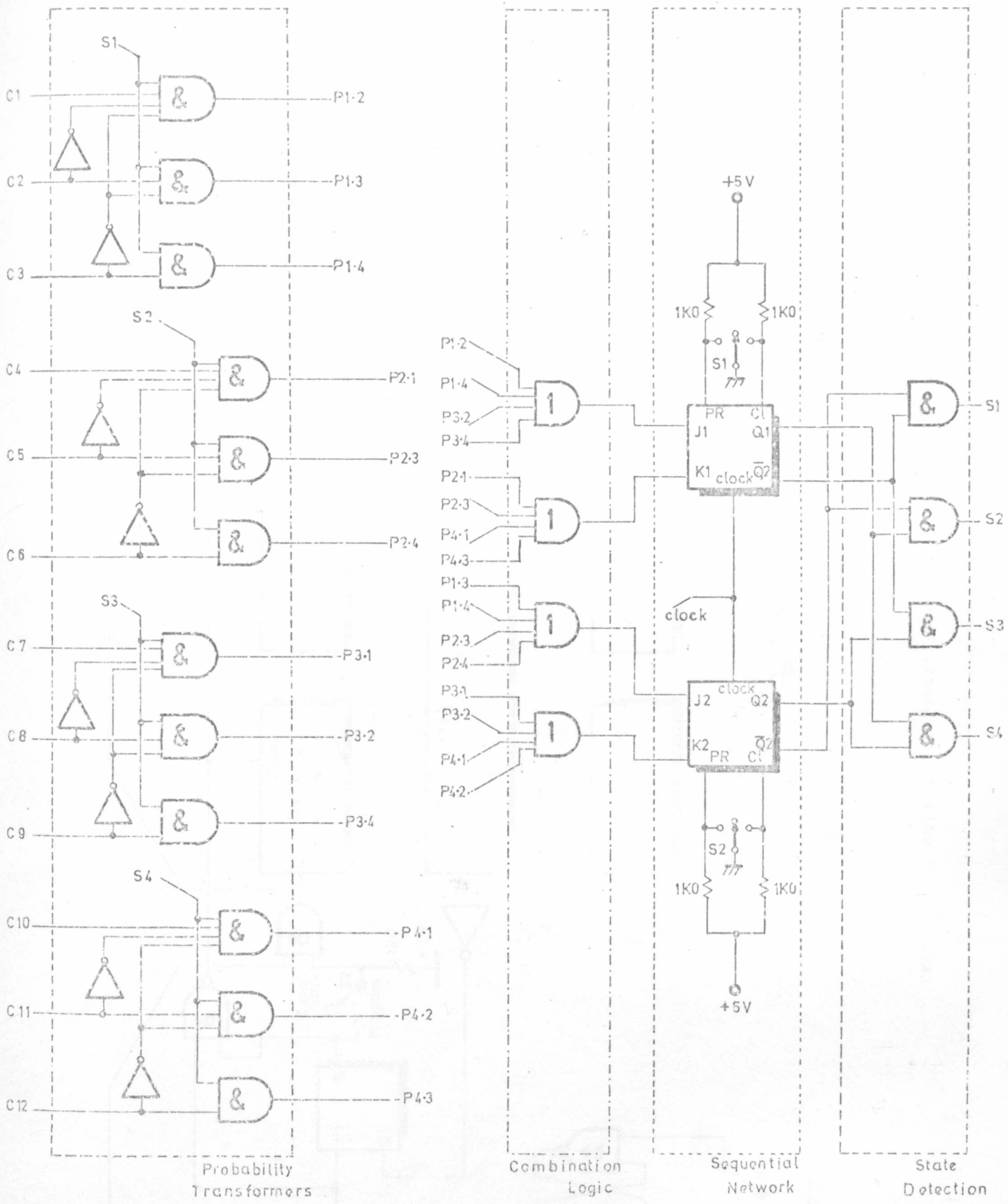


FIGURE 6·2 Circuitry for Simulation of 4-State Markov Chain

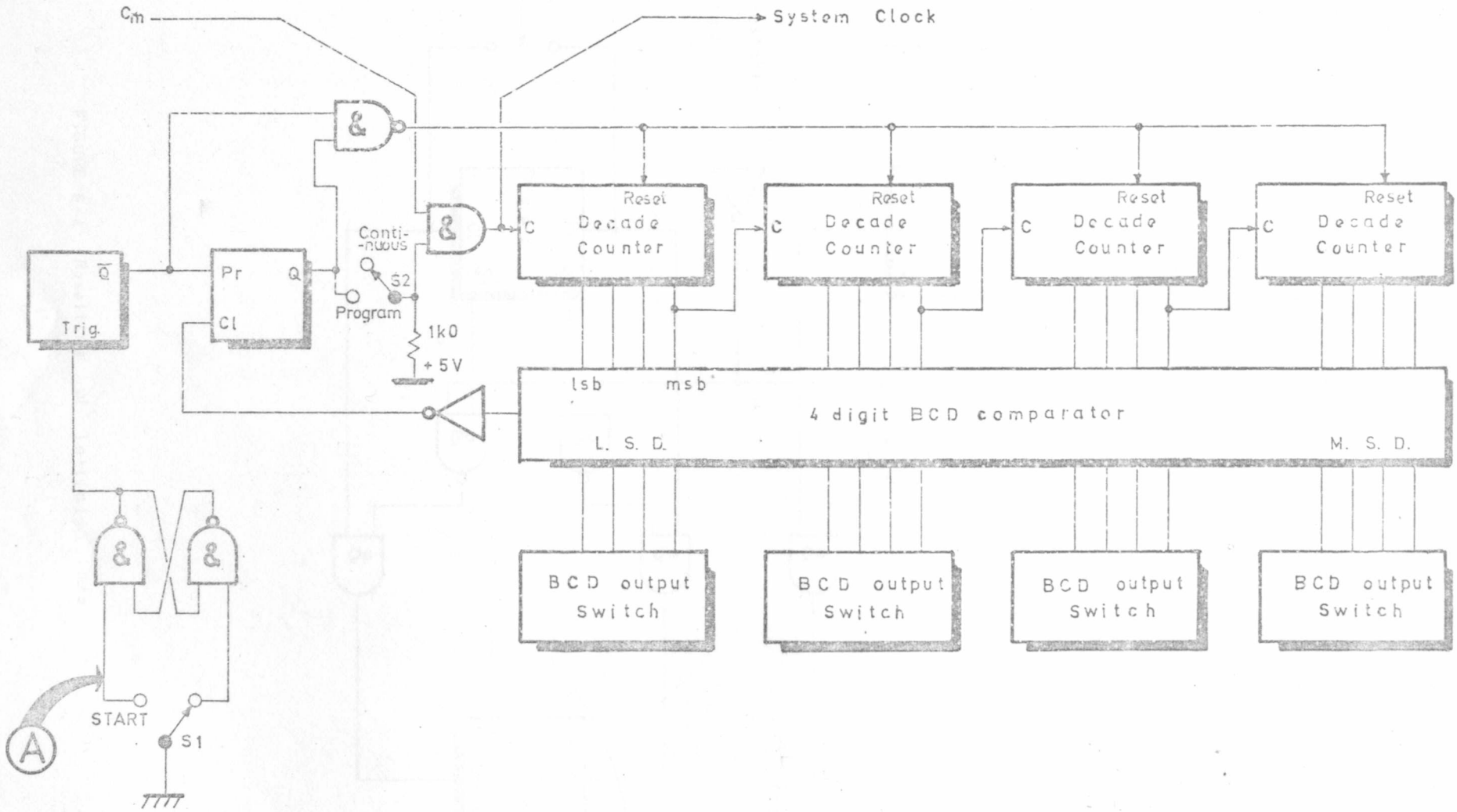


FIGURE 6·3 Circuitry for Preset Number of Clock Pulses

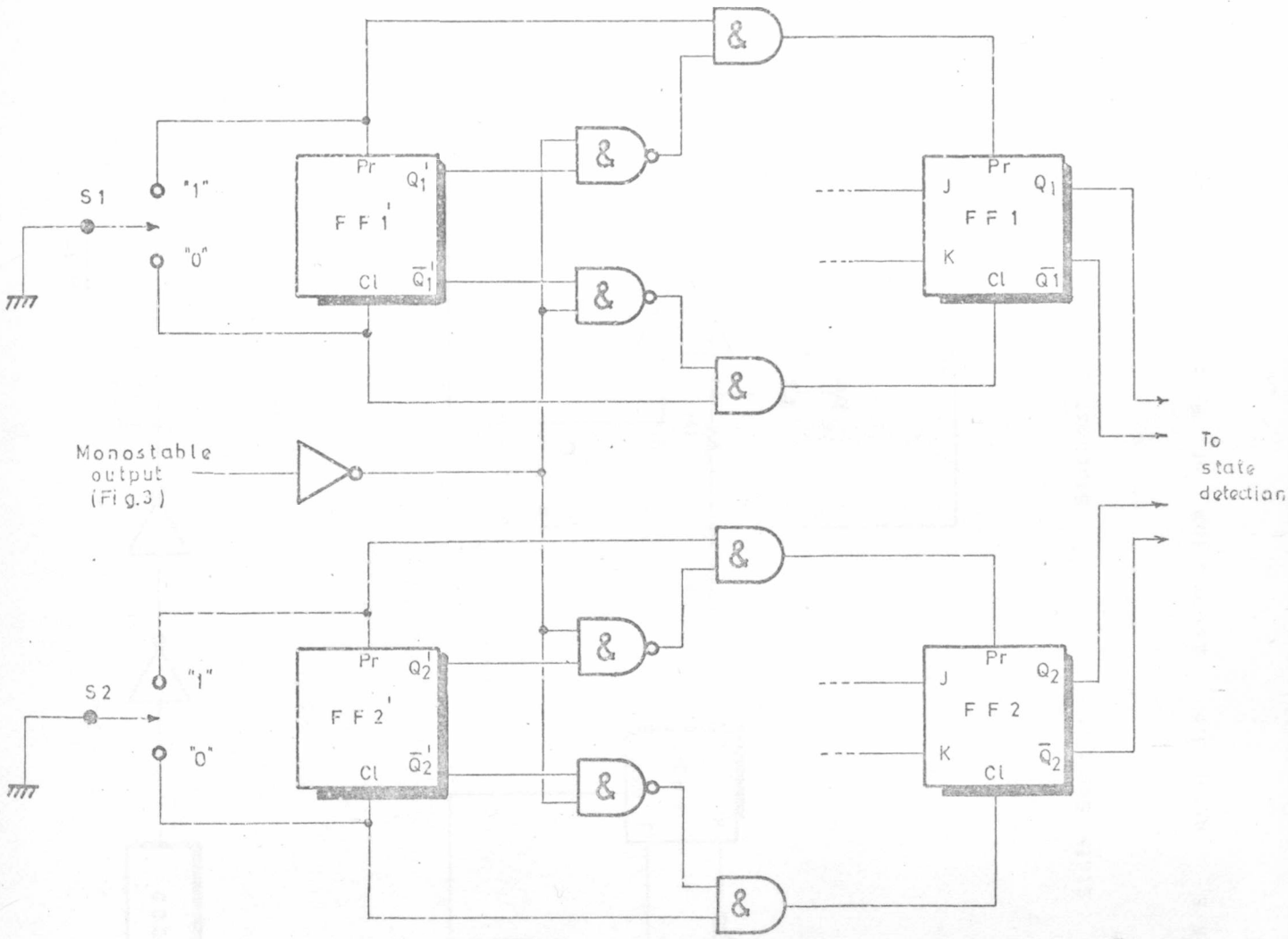


FIGURE 6·4 Resetting of Initial State

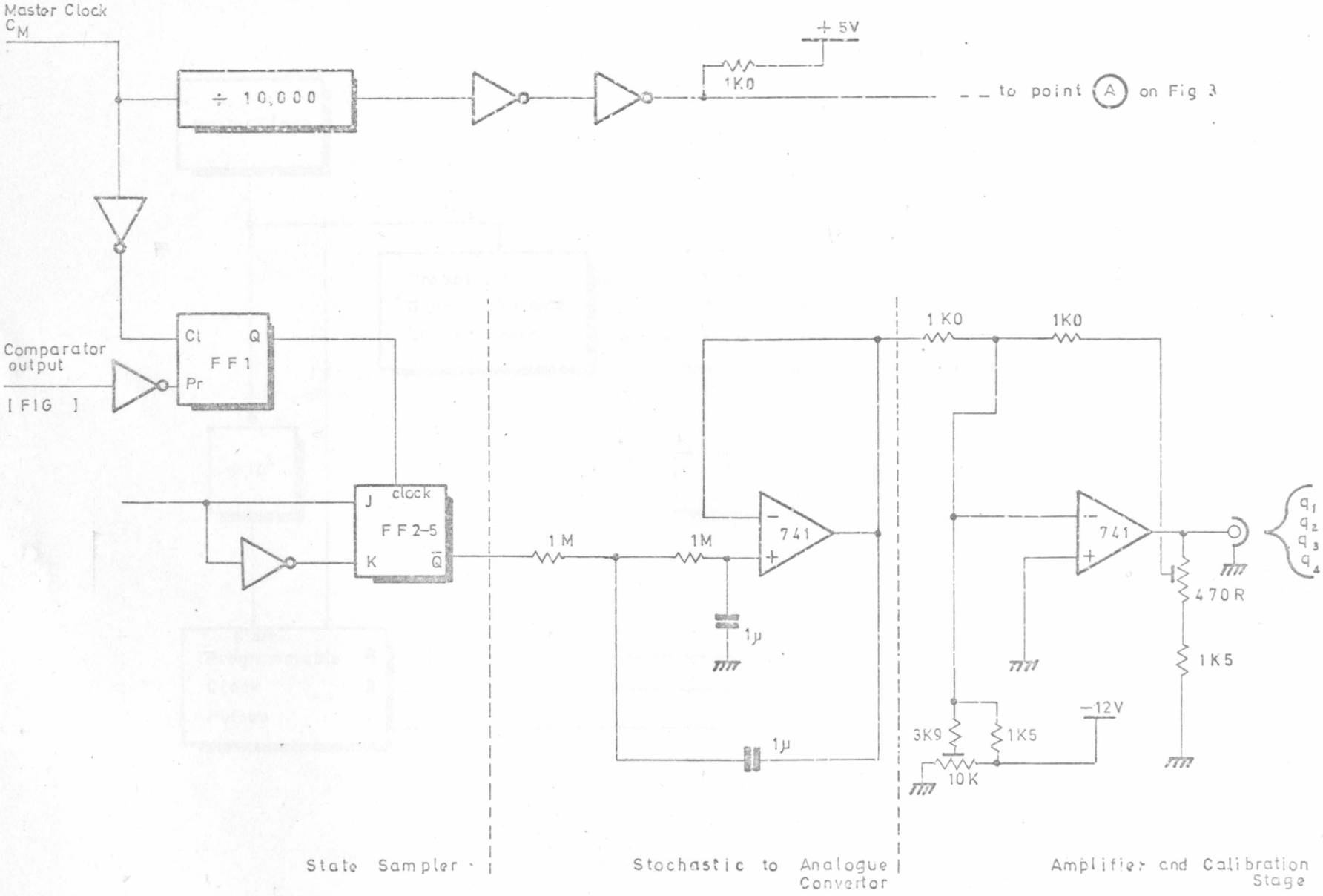


FIGURE 6·5 Automatic Estimation of Probability of being in any State

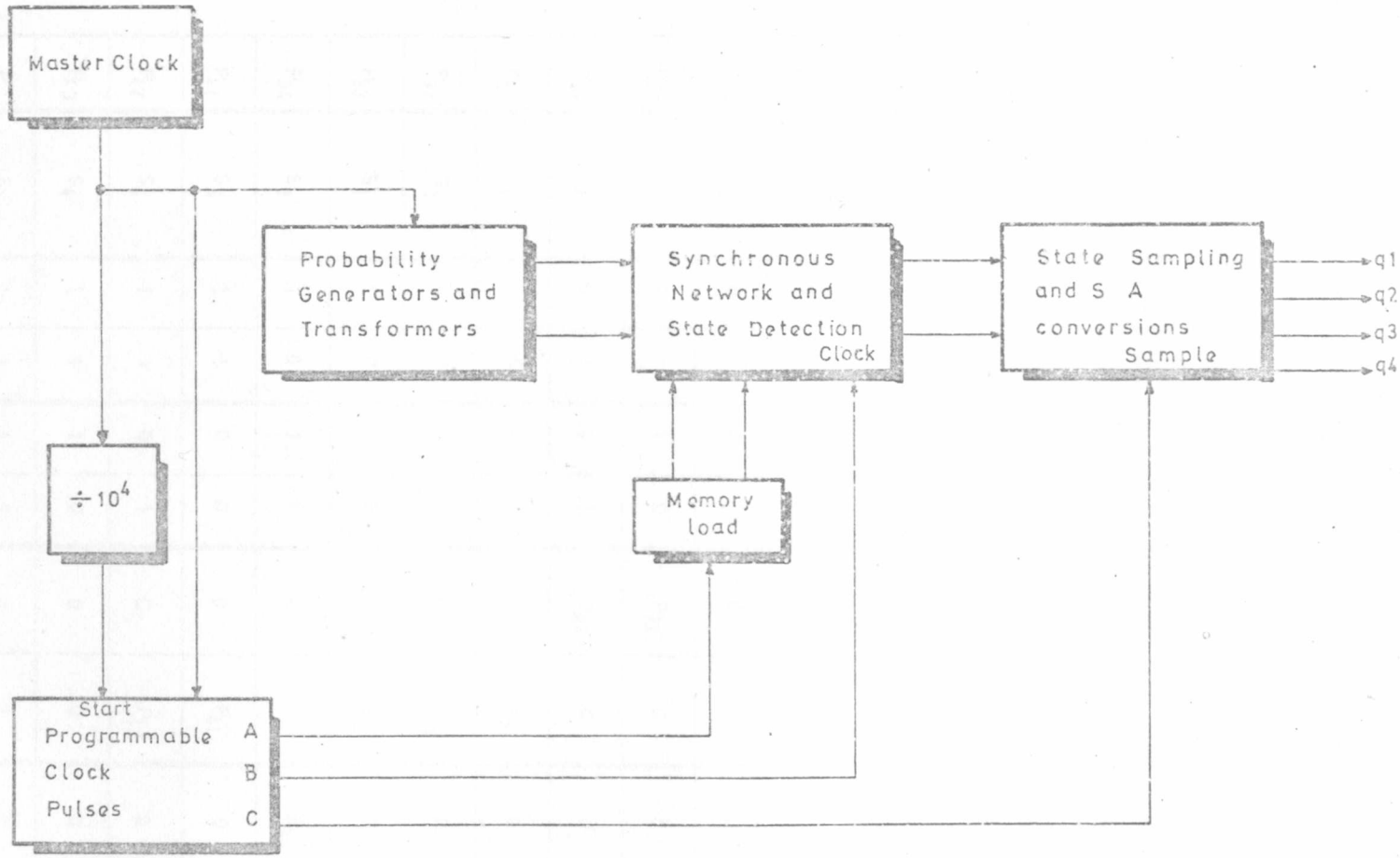


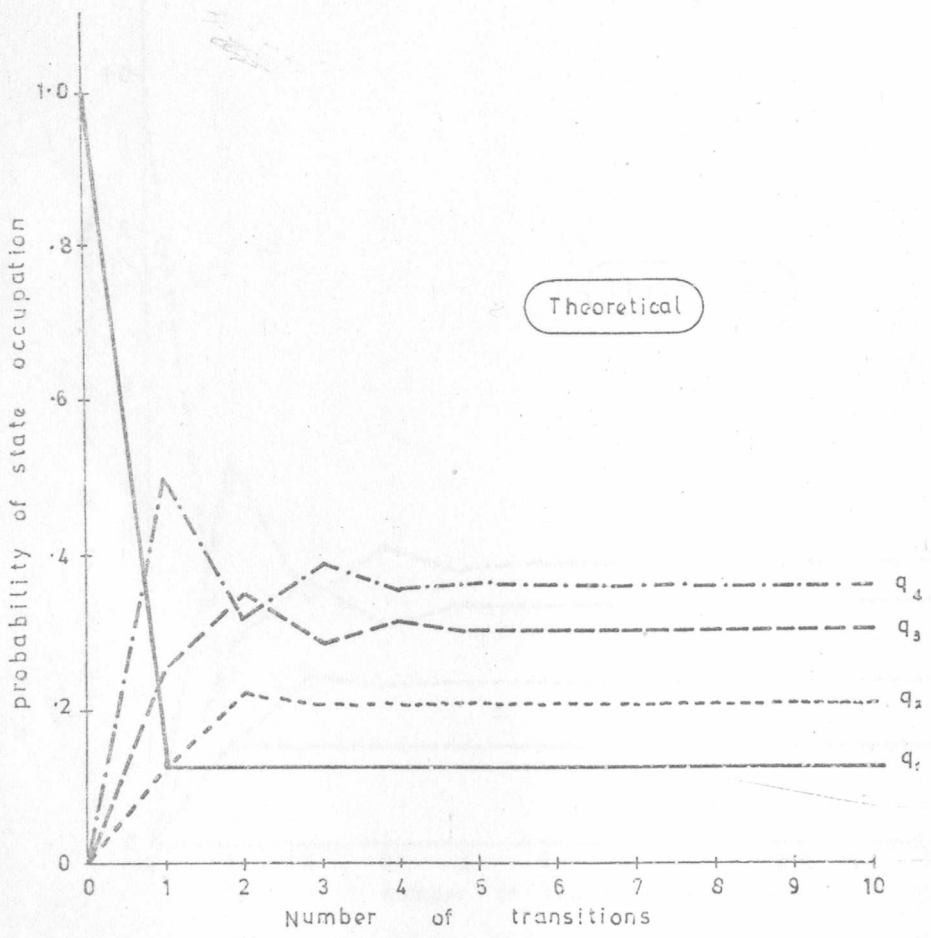
FIGURE 6·6 Block Diagram of Markov Chain Simulator

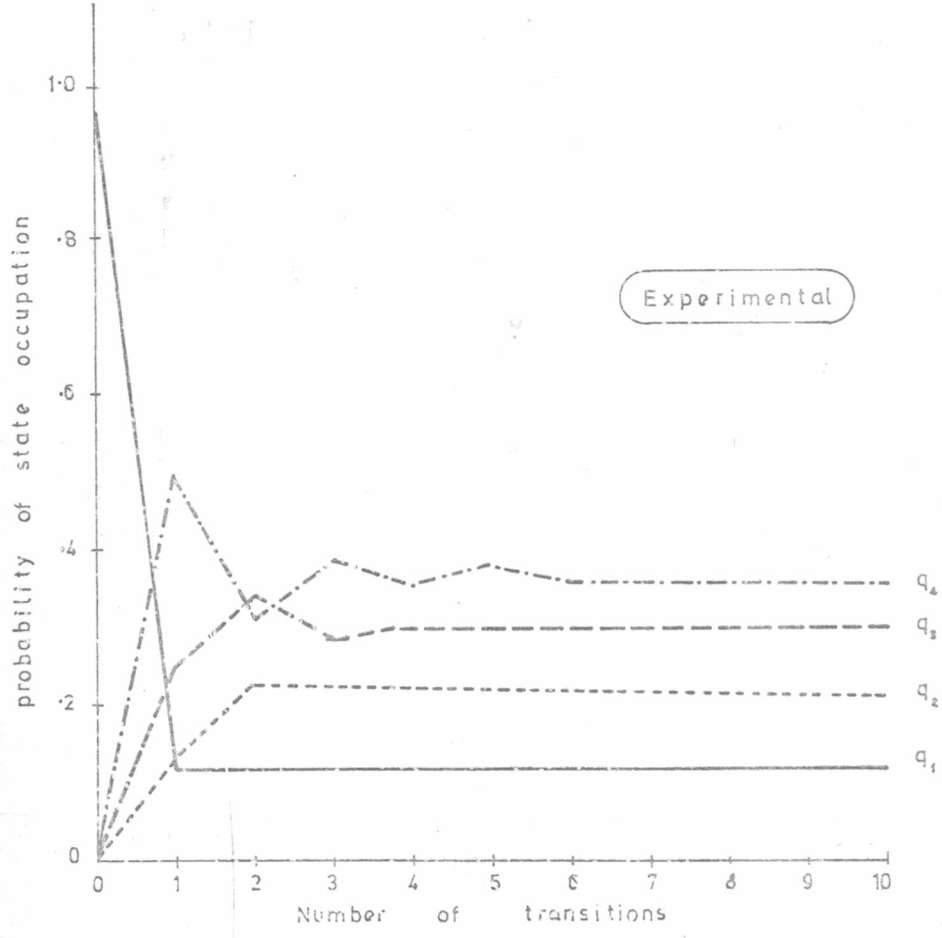
	Present State	time t		time t+1		time t			
		Q2	Q1	Q2	Q1	J2	K2	J1	K1
P_{11}	S_1	0	0	0	0	0	0	0	0
P_{12}	S_1	0	0	0	1	0	0	P_{12}	0
P_{13}	S_1	0	0	1	0	P_{13}	0	0	0
P_{14}	S_1	0	0	1	1	P_{14}	0	P_{14}	0
P_{21}	S_2	0	1	0	0	0	0	0	P_{21}
P_{22}	S_2	0	1	0	1	0	0	0	0
P_{23}	S_2	0	1	1	0	P_{23}	0	0	P_{23}
P_{24}	S_2	0	1	1	1	P_{24}	0	0	0
P_{31}	S_3	1	0	0	0	0	P_{31}	0	0
P_{32}	S_3	1	0	0	1	0	P_{32}	P_{32}	0
P_{33}	S_3	1	0	1	0	0	0	0	0
P_{34}	S_3	1	0	1	1	0	0	P_{34}	0
P_{41}	S_4	1	1	0	0	0	P_{41}	0	P_{41}
P_{42}	S_4	1	1	0	1	0	P_{42}	0	0
P_{43}	S_4	1	1	1	0	0	0	0	P_{43}
P_{44}	S_4	1	1	1	1	0	0	0	0

TABLE 6·1 Truth Table for Sequential Network

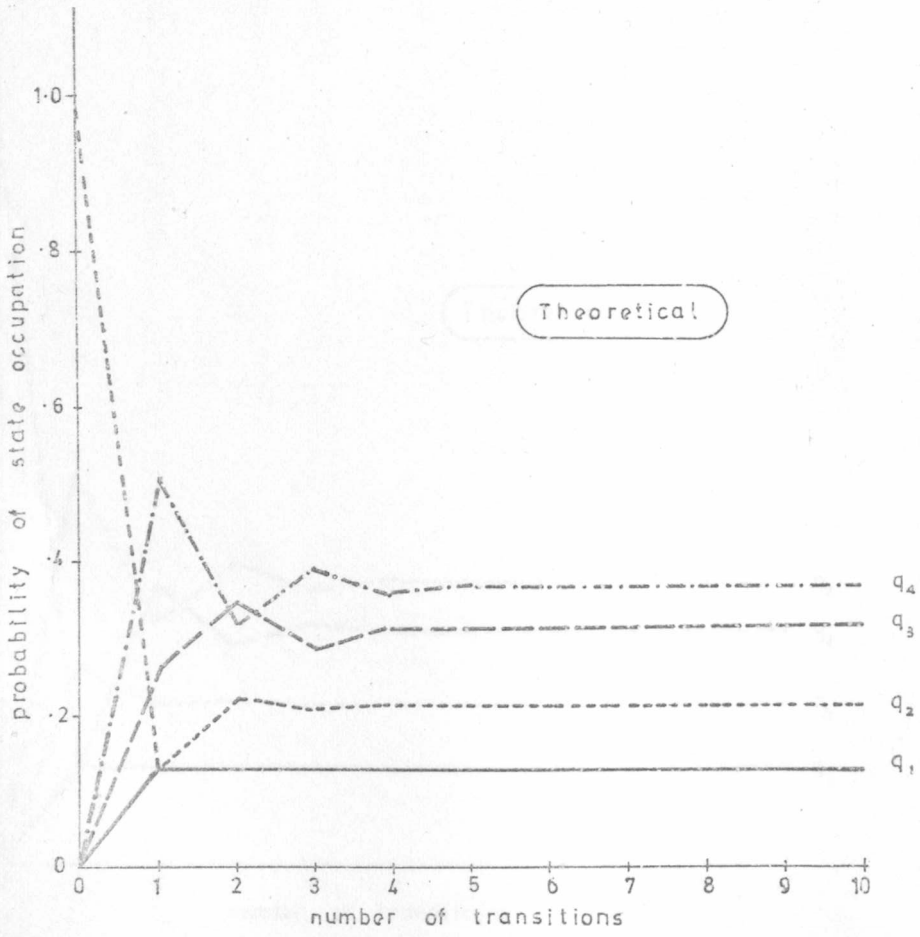
Stochastic Sequence	Value
C_1	$\frac{P_{12}}{P_{11} + P_{12} + P_{13}}$
C_2	$\frac{P_{13}}{P_{11} + P_{12}}$
C_3	P_{14}
C_4	$\frac{P_{21}}{P_{21} + P_{22} + P_{23}}$
C_5	$\frac{P_{23}}{P_{21} + P_{22}}$
C_6	P_{24}
C_7	$\frac{P_{31}}{P_{31} + P_{32} + P_{33}}$
C_8	$\frac{P_{32}}{P_{31} + P_{33}}$
C_9	P_{34}
C_{10}	$\frac{P_{41}}{P_{41} + P_{42} + P_{44}}$
C_{11}	$\frac{P_{42}}{P_{41} + P_{44}}$
C_{12}	P_{43}

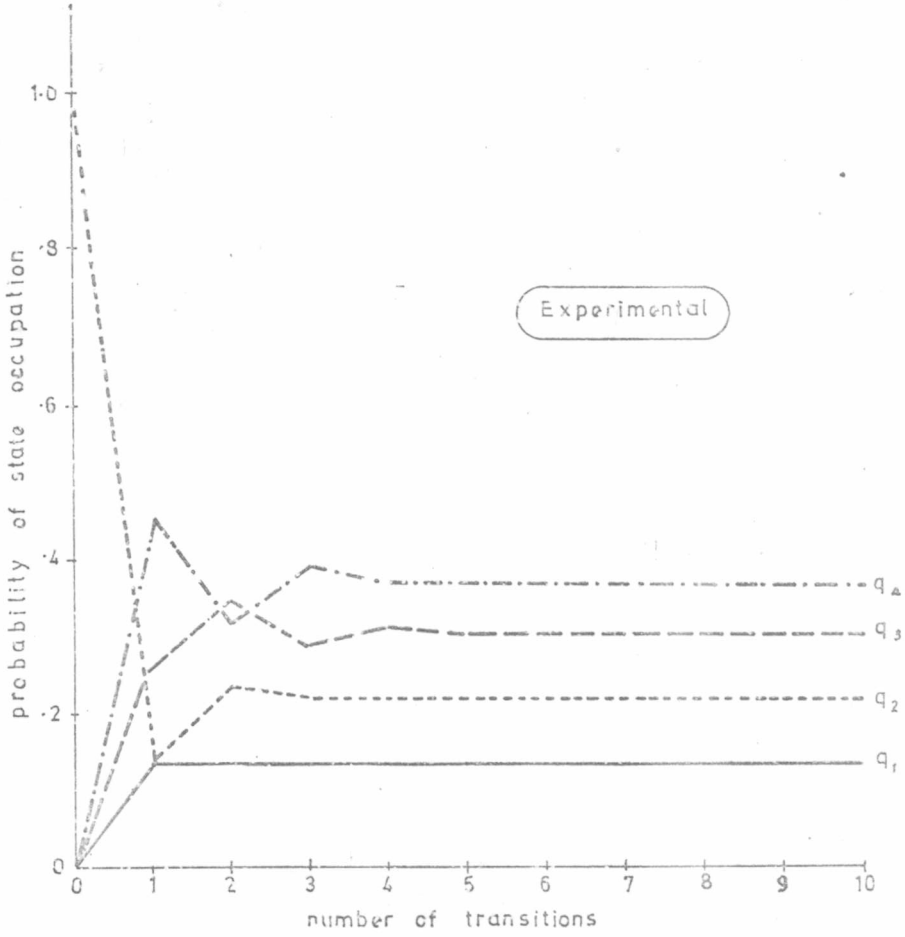
TABLE 6·2 Values of Comparator Output Probabilities



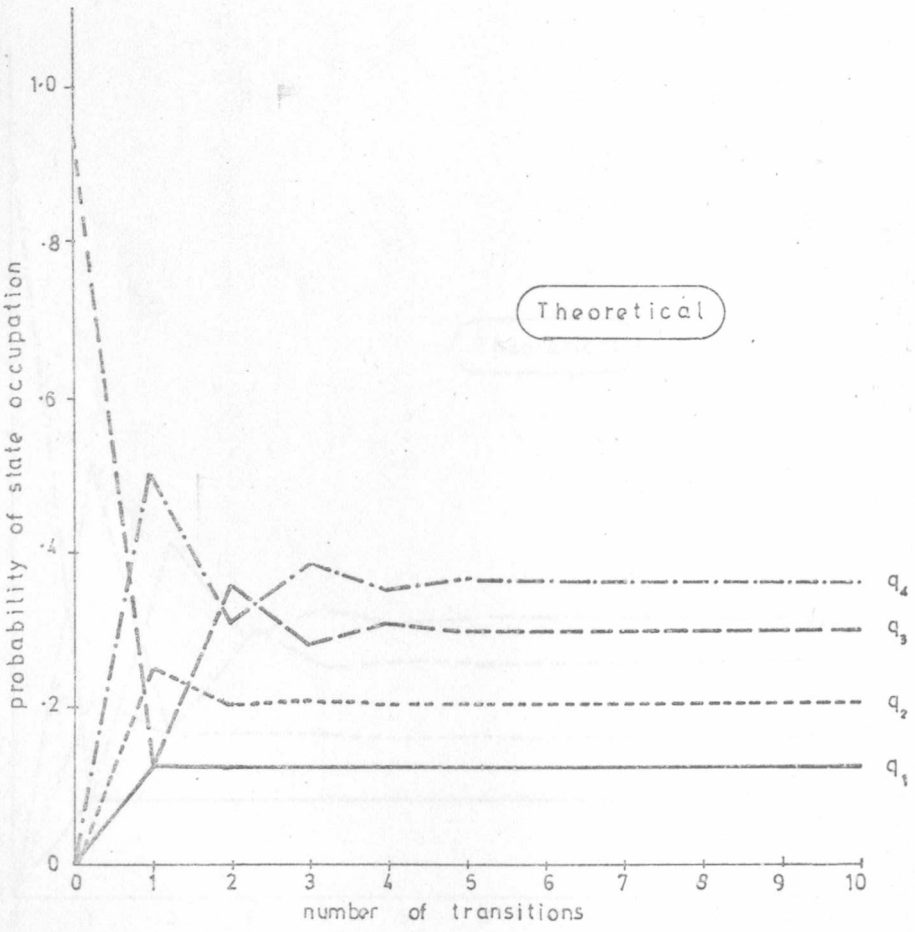


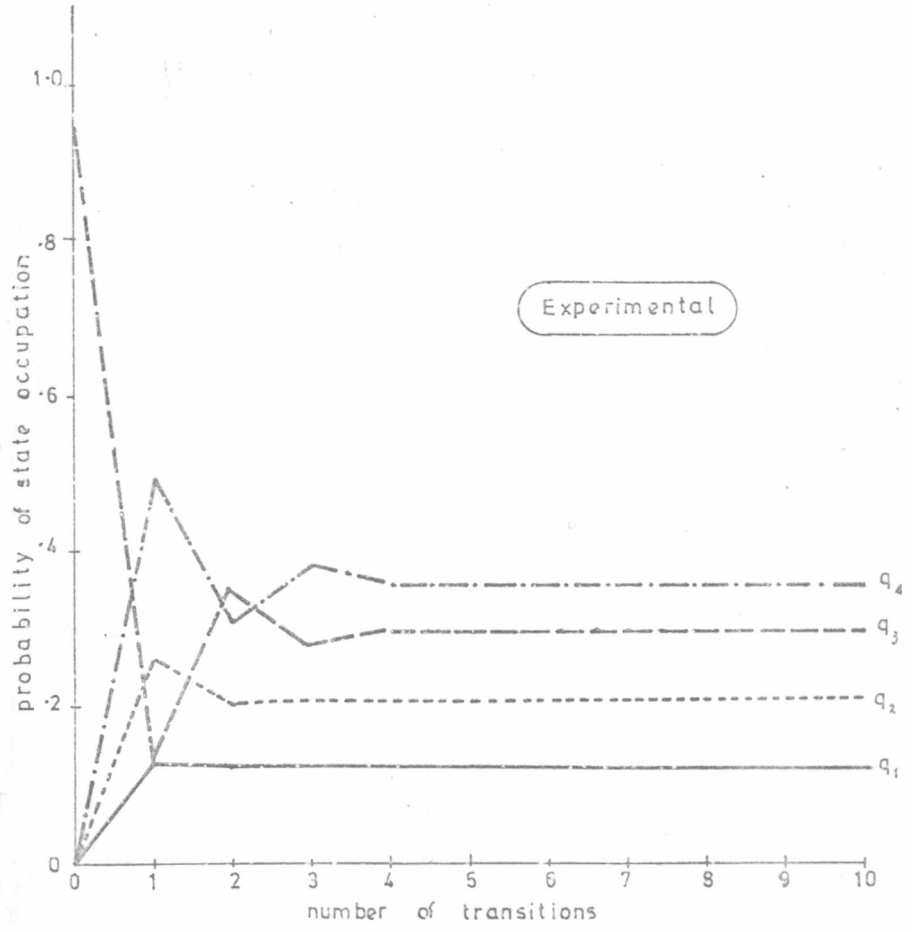
Graph 6·1 Starting State S_1



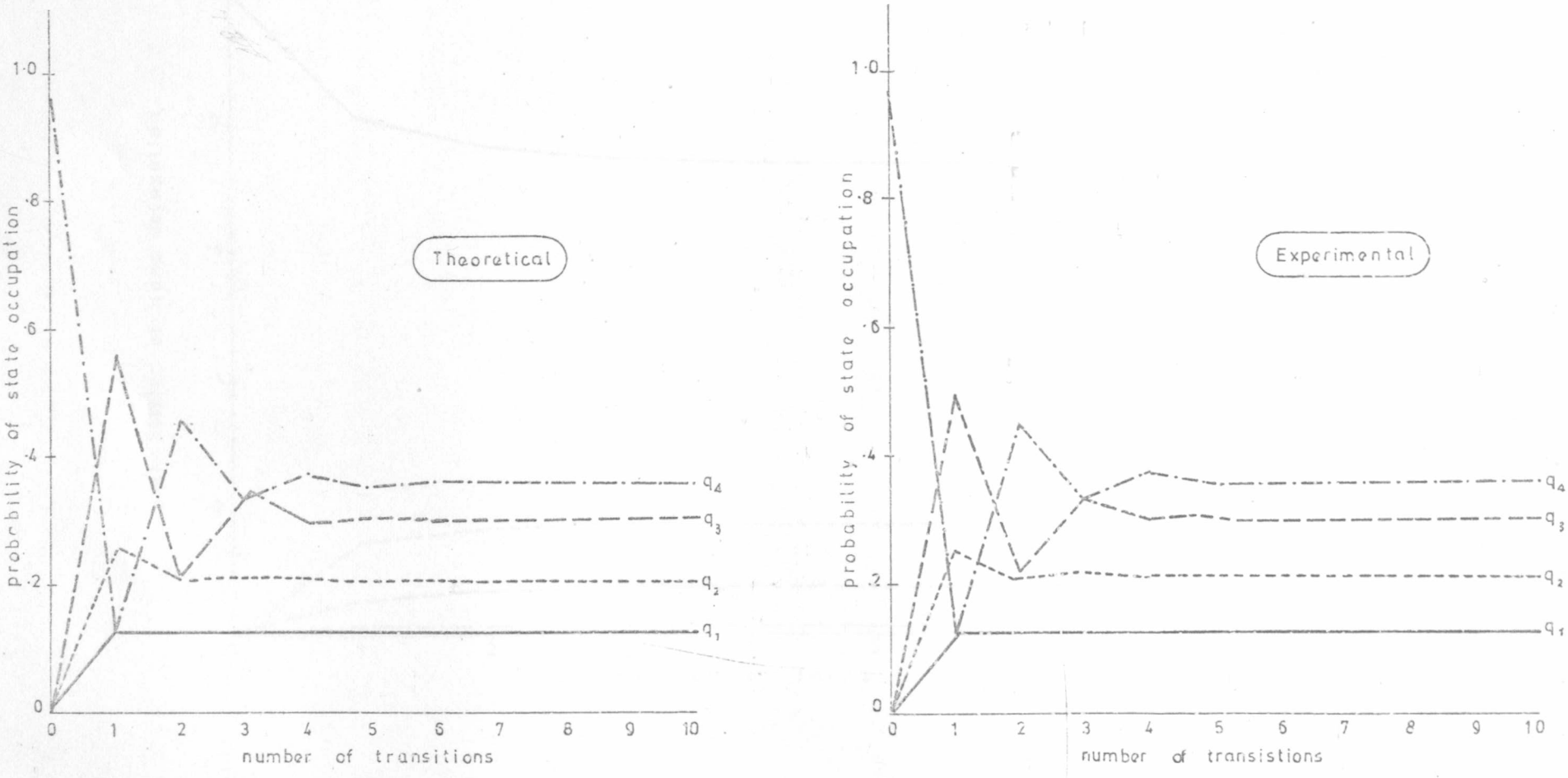


Graph 6·2 Starting State S_2

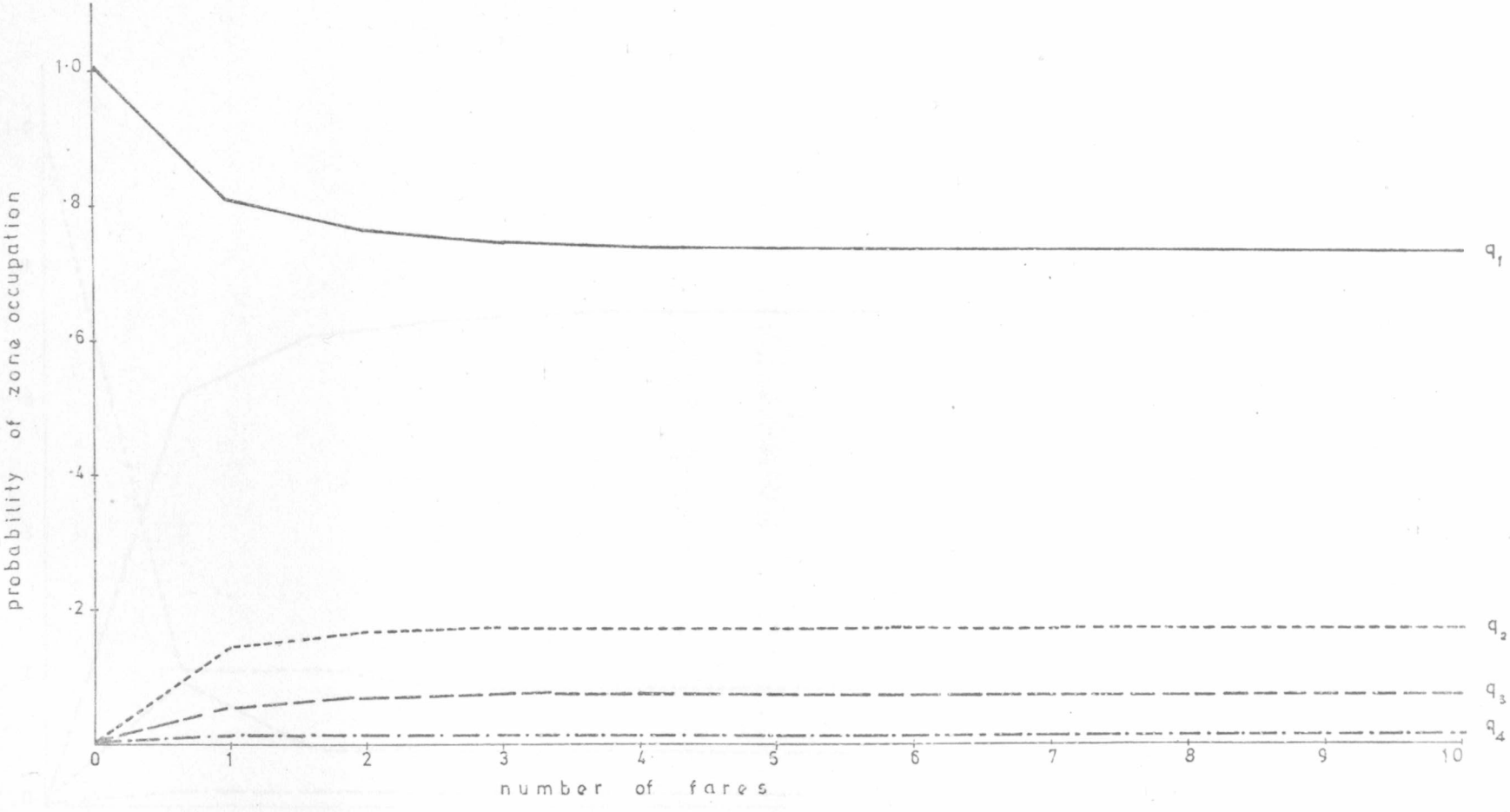




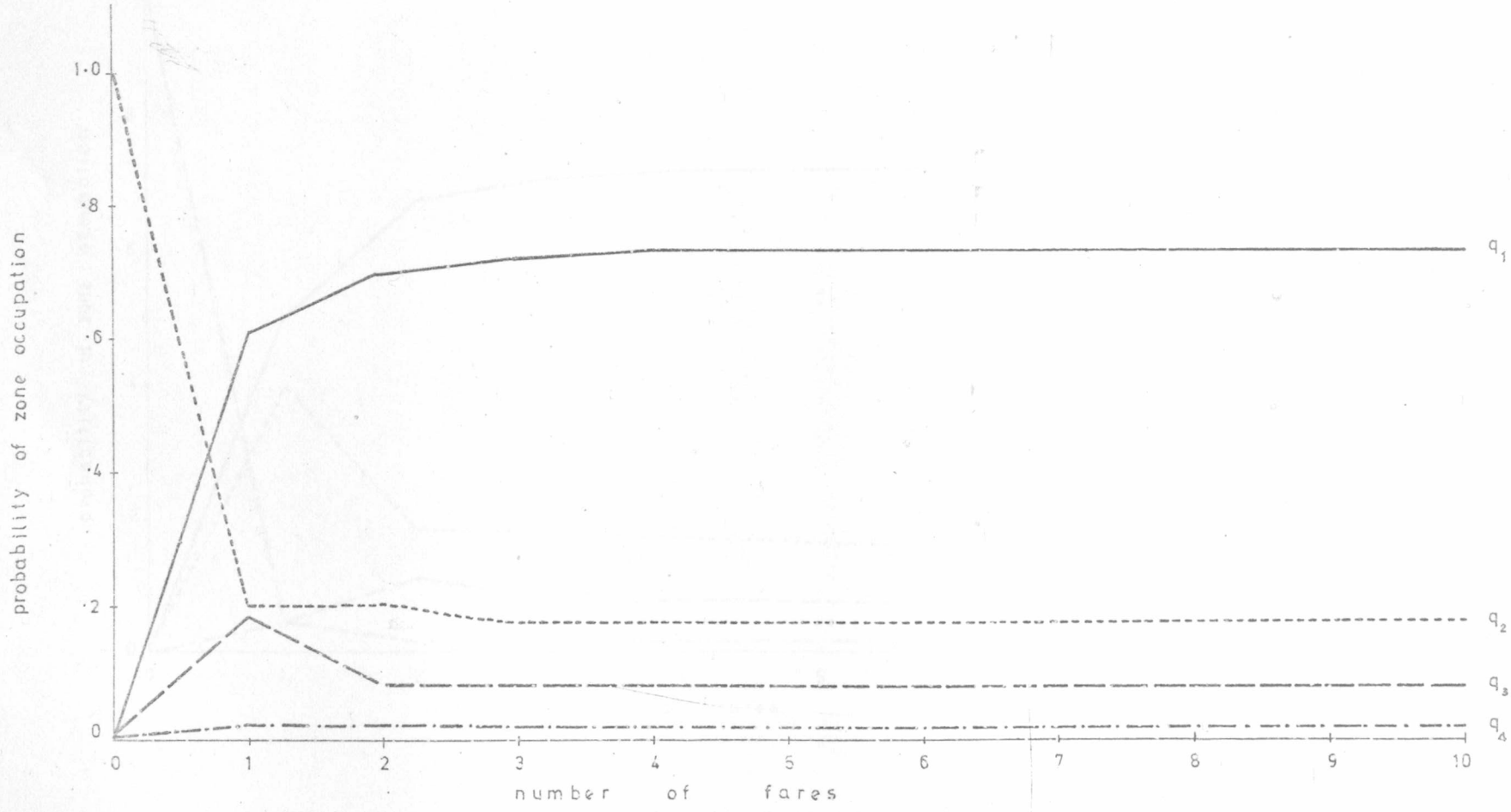
Graph 6-3 Starting State S_3



Graph 6·4 Starting State S_4

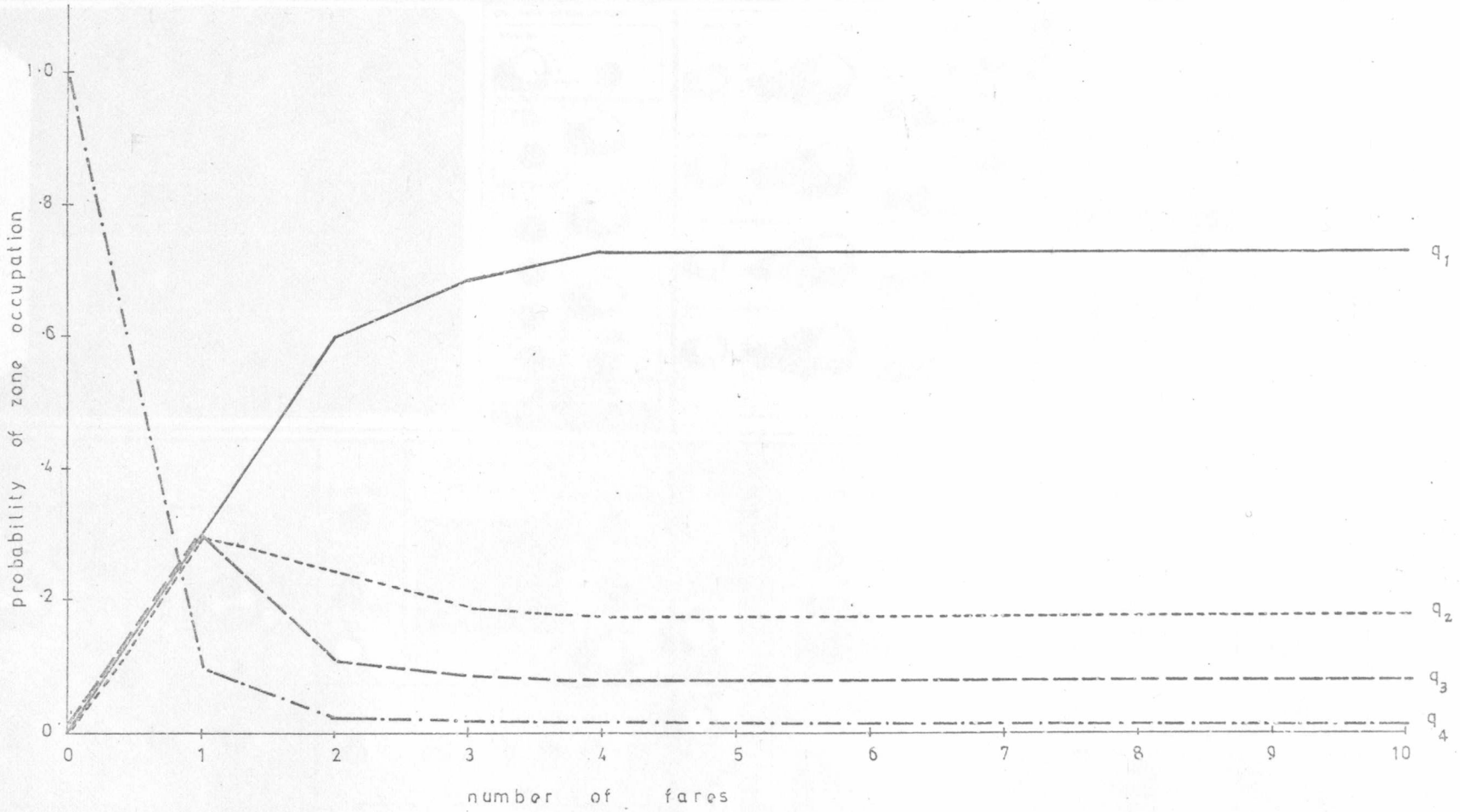


Graph 6·5 Starting Zone S_1



Graph 6·6 Starting Zone S_2

Graph 6·7 Starting Zone S_3



Graph 6.8 Starting Zone S_4

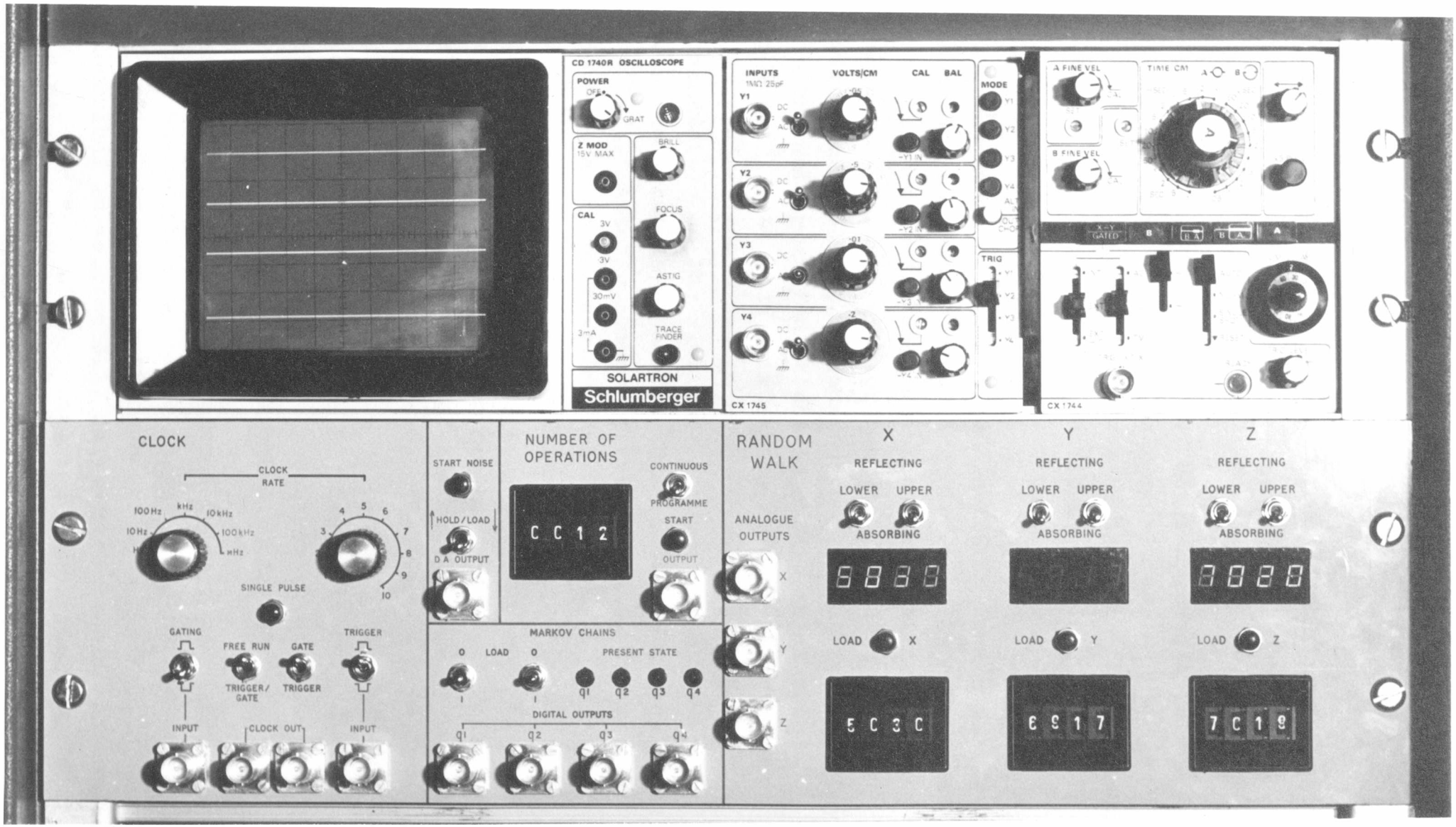


PLATE 6·1 Random Walk and Markov Chain Simulator

CHAPTER 7

RANDOM WALK SIMULATION

7.1 Introduction and Definition of Random Walks [8,10]

As was the case with Markov Chains discussed in the previous chapter, random walk models are widely used in Monte Carlo methods especially in the solution of partial differential equations, multiple integrals and in the study of diffusion processes.

A random walk may be defined in the following manner. Consider the motion of a particle which is restricted to motion in a single dimension. The particle may move to the right with probability p and to the left with probability q. Assuming the particle cannot stay in the same position at the time of a trial then $p + q = 1$. At each trial the particle will vacate its present state and move to one of the states immediately to the right or left and at no time can the particle move more than one state to the right or left. After a number of trials the particle will move in a random fashion thus constituting a random walk. If the particle is assumed to be travelling in the x-axis, boundaries may be introduced thus limiting the number of states which may be occupied. Consider the arbitrary boundaries of the origin ($x = 0$) and at some point in the positive direction ($x = a$), thus confining the motion of the particle in the range 0 to a. The random walk may be started at any point in this range and the starting state is given as k. A diagram of this model is shown in Figure 7.1.

It may be observed that a random walk is a special case of a Markov Chain with the restriction that it cannot 'jump' states. Using the concept of a stochastic matrix described in section 6.1, only the transition probabilities $P_{n,n+1}$ and $P_{n,n-1}$ can exist, ie, the particle can only move /

move to an adjacent state. The transition probabilities $P_{n,n+1}$ and $P_{n,n-1}$ will be constant for all n and will have values of p and q respectively. All other transition probabilities are zero. Thus the transition matrix may be found easily and is shown in Figure 7.2. Two cases are shown and these cater for two boundary conditions, absorbing and reflecting. In the absorbing case, if the particle enters either boundary then the random walk is terminated. Conversely with the reflecting barriers the particle may leave either boundary state but cannot move outside the range 0 to a and if the particle attempts to exceed the range then it will remain in the boundary state.

A useful analogy to this random walk is the classical 'gamblers ruin problem'[8] and will be useful in demonstrating some aspects of a random walk. Using the notation of the above random walk consider the case of a gambler with initial capital k. He plays against an opponent whose initial capital is a-k and the game consists of a number of trials. In each trial the gambler has a chance p of winning one unit per trial and a chance q of losing one unit. In practice such a situation would be a random walk with absorbing boundaries at 0 and a, ie, the game would cease whenever the gambler had no capital left (x = 0) or had won all his opponents capital (x = a). If this random walk had reflecting barriers this would mean that if the gambler had lost all his capital (x = 0) and if he won the next trial he would receive one unit otherwise he would stay with no capital and the contest would last for an infinite number of trials.

Some of the basic concepts will now be developed using the example of the gambler.

7.2 'Gambler's Ruin Problem'[8,10] /

7.2 'Gambler's Ruin Problem'[8,10]

In this section all formulae developed will assume that the random walk model has absorbing boundaries at 0 and a.

Consider firstly the probability of the gambler's ruin which will be calculated using the method of difference equations. After the first trial the gambler's capital is either k+1 or k-1 depending on the outcome of the game. Hence the probability of ruin is

$$q_k = pq_{k+1} + q\,q_{k-1} \qquad 1 < k < a-1$$

and

$$q_1 = pq_2 + q \qquad k = 1$$

$$q_{a-1} = q\,q_{a-2} \qquad k = a-1$$

which may be written in the general form as

$$q_k = pq_{k+1} + q\,q_{k-1} \qquad 1 \leqslant k \leqslant a-1 \qquad \text{---- (7.1)}$$

with the limits of $q_0 = 1$ and $q_a = 0$.

If we put $q_k = w^k$ in equation (7.1) we obtain the auxiliary equation

$$pw^2 - w + q = 0 \qquad \text{---- (7.2)}$$

which has roots $w = 1$ and $w = q/p$. For $q \neq p$ equation (7.2) has separate roots and the general solution of equation (7.1) becomes

$$q_k = A(1)^k + B(q/p)^k$$

Using the boundary conditions $q_0 = 1$ and $q_a = 0$ we may solve for the constants A and B giving

$$q_k = \frac{(q/p)^a - (q/p)^k}{(q/p)^a - 1} \qquad \text{---- (7.3)}$$

To /

To evaluate the probability of the gambler's success, p_k (his opponents ruin) it is simply a case of interchanging p and q and writing a-k for k in equation (7.3). This gives

$$p_k = \frac{(q/p)^k - 1}{(q/p)^a - 1} \qquad \text{---- (7.4)}$$

Combining (7.3) and (7.4) shows that

$$p_k + q_k = 1 \qquad \text{---- (7.5)}$$

which indicates that the possibility of an unending contest is zero.

If p = q then the auxiliary equation (7.2) has two equal roots of w = 1 and the general solution is of the form

$$q_k = C(1)^k + Dk(1)^k$$

and again using the boundary conditions to evaluate the constants C and D gives

$$q_k = 1 - k/a \qquad \text{---- (7.6)}$$

and hence

$$p_k = k/a \qquad \text{---- (7.7)}$$

The second equation to be formulated is that of the expected duration of the game (d_k). Again the starting capital of the gambler is k. If the gambler wins the first trial the conditional duration is d_{k+1} and so the expected duration is $1 + d_{k+1}$. Similarly the expected duration of the game if the gambler loses the first game is $1 + d_{k-1}$. Therefore

d_k /

$$d_k = p(1+d_{k+1}) + q(1+d_{k-1}) \qquad 1 \leq k \leq a-1$$

$$= 1 + pd_{k+1} + qd_{k-1} \qquad \text{---- (7.8)}$$

with the boundary conditions $d_0 = 0$ and $d_a = 0$. Equation (7.8) is simply a non-homogeneous case of (7.1) and the general solution of (7.1) may be used provided we add the particular solution of (7.8). The general solution of equation (7.8) is thus,

$$d_k = \frac{k}{q - p} + A + B\left(\frac{q}{p}\right)^k \quad \text{where} \quad q \neq p$$

Evaluating the constants A and B using the boundary conditions gives

$$d_k = \frac{k}{q - p} - \frac{a}{q - p} \cdot \frac{1 - (q/p)^k}{1 - (q/p)^a} \qquad \text{---- (7.9)}$$

If $p = \frac{1}{2}$ equation (7.9) becomes

$$d_k = k(a-k) \qquad \text{---- (7.10)}$$

This equation shows that for trials of equal skill, ie, $p = q$ then the duration of the game is longer than would be expected. For example, if both players start with 5000 units ($k = 5000$, $a = 10000$) then the expected duration of the game is 25 million trials. As will be seen later this presents a considerable problem in the experimental verification of predicted results.

Finally the probability distribution of ruin at the nth trial will be dealt with. For a starting state k the probability of ruin at the nth trial is $q_{k,n}$. Again the method of difference equations may be applied giving

$$q_{k,n+1} = pq_{k+1,n} + qq_{k-1,n} \qquad 1 \leq k \leq a-1, \quad n \geq 0$$

with /

with the boundary conditions

$$q_{0,n} = q_{a,n} = 0 \qquad n \geqslant 1$$

$$q_{0,0} = 1; \quad q_{k,0} = 0 \quad k > 0$$

The solution of the above difference equation is

$$q_{k,n} = a^{-1} 2^{n+1} p^{(n-k)/2} q^{(n+k)/2} \sum_{j<a/2} \cos^{n-1} \frac{\pi j}{a} \sin\frac{\pi j}{a} \sin\frac{\pi kj}{a} \qquad \text{---- (7.11)}$$

where the summation term extends over the positive integers $<a/2$ and for large n only the first few terms are significant.

As was the case for the probability of ruin, the probability of ruin for the gambler's opponent, $p_{k,n}$, is found by using equation (7.11) with p and q interchanged and a-k written for k. The probability of the game ending at the nth trial is therefore $p_{k,n} + q_{k,n}$.

Equations (7.3), (7.4), (7.6), (7.7), (7.10) and (7.11) are the most important formulae relating to the concept of random walks and any simulation model would have to be shown to behave according to these equations.

7.3 System Design

A hardware simulation of a random walk may clearly be performed by a reversible counter with appropriate control logic. The specifications for a random walk simulator were as follows:

(a) the reversible counter had to have programmable probabilities of counting UP or DOWN, ie, p and q. This is easily accomplished using a stochastic comparator.

(b) /

(b) the counter had to have the facility of remaining in the same state after a trial. In the terms of the gambler's ruin example this means that the outcome of any trial could be a draw. Again this is simply accomplished using a stochastic comparator with its output applied to the enable of the counter.

(c) the random walk could be started in any state.

(d) a visual indication of the occupied state was to be given using 7-segment displays.

(e) both boundaries had the choice of being absorbing or reflecting and one boundary could reflect while the other could absorb.

(f) the number of states of the random walk would be sufficiently large for higher resolution and hence accuracy.

(g) the simulator would be capable of dealing with problems in three dimensions which means three separate random walks.

The final system is shown in Figure 7.3 and as can be seen the basis of the random walk simulator is the four digit UP/DOWN decade counter. A decade counter was chosen in preference to a binary counter because of the simplicity in driving a display. In fact the contents of the counter are displayed at all times by simply decoding the four BCD digits and driving four seven segment displays. The counter has a probability of counting UP determined by P_U and this is simply routed to the UP/DOWN lines. Because of requirement (b) above the counter has a probability of staying in the same state equal to P_H which is applied to the enable input of the counter. Whenever P_H is at logic '1' then the counter will remain unchanged regardless of the state of P_U. Thus the probability of staying in the same state is truly P_H. However in the case of P_U, the counter will count UP if P_U is logic '1' and will count DOWN if P_U is logic '0', if and only if P_H is /

is logic '0'. Thus in the event of both P_H and P_U being logic '1' at the same time then the counter will not count UP. Therefore the true probability of counting UP (p) is

$$p = P_U - P_U P_H$$

and

$$q = (1-P_U) - (1-P_U)P_H$$

or

$$p + q + r = 1$$

where

q = probability of counting down
r = probability of staying in same state
p = probability of counting up

which is the expected result. The above equations may be rearranged to give the values P_U and P_H in terms of p, q and r, thus

$$P_U = \frac{p}{p + q} \qquad \text{---- (7.12)}$$

$$P_H = r \qquad \text{---- (7.13)}$$

The starting state of the random walk (k) is set by means of 4 thumbwheel switches which give BCD outputs and upon application of the initial conditions switch, the switch outputs are loaded into the decade counter in parallel fashion.

At any of the two boundary states, ie, 0000 and 9999, an output is received from the gate which combines the four Max + Min outputs, if the counter will overflow at the next clock pulse. For example if the random walk state is 9999 and P_U is logic '1' then this corresponds to an attempt to cross a boundary which must be prevented. This combined Max + Min output is used to hold the counter in the boundary state. If the counter tries to return to the region between the boundaries then the Max + Min signals are removed, eg, if /

if the random walk is in the 0000 state and P_U changes to logic '1'. This is a random walk with reflecting boundaries and to have a choice of reflecting or absorbing boundaries it is necessary to detect the occupation of each boundary state. This is easily done by using D_d which is the most significant bit of the most significant digit and will indicate the actual boundary occupied at the time of the RCE signal. If switch S1 is closed at the time of the Max + Min signal and D_d is logic '1', ie, the upper boundary state, then the flip-flop is cleared and its output holds the counter in the upper boundary state until the initial state is reloaded, ie, an absorbing boundary. Similarly if switch S2 is closed the lower boundary will be absorbing. Switches S1 and S2 may be operated independently thus allowing the choice of different boundary conditions for each boundary.

Finally the simulator was given 10,000 states because this gives high resolution and hence accuracy.

7.4 Experimental Verification of Performance

As was previously mentioned in section 7.2 the maximum expected duration of a random walk with 10,000 states is 25×10^6 or 25 seconds at a clock rate of 1 MHz. Because most experimental results involve probability then a large number of random walks must be performed and for 25 seconds duration per walk then the time involved would be astronomical. For this reason the size of the random walk has been reduced to 100 states for all experimental results. This gives a maximum average duration of 2.5 ms at 1 MHz which is more practical.

The /

The various predictions of performance derived in section 7.2 will now be tested experimentally starting with the probability of absorption at either boundary for varying starting state k. In fact for values of p and q other than 0.5 (r is taken as 0 for all experimental results) then the probability of being absorbed at one boundary is virtually unity and at the other boundary it is virtually zero and so only the one case has been examined. Graph 7.1 shows the probability of the gamblers ruin (the probability of absorption at the lower boundary q_k) for varying k and for p = q = 0.5. The solid line indicates the expected values which were calculated using program 2 in Appendix 2 and the crosses indicate the experimental results. As can be seen the experimental results are very close to the expected values. Although only the values of p and q of 0.5 were examined in this case, it will be seen from Chapter 8 that these are the values of most interest. The value of p_k is easily obtained from $p_k = 1 - q_k$.

The results of Graph 7.1 were obtained by performing a large number of random walks and recording the number of times the walk terminated at each boundary.

Now the expected duration of the game will be examined. To find this value a large number of random walks were executed and the number of trials before each absorption were recorded, the arithmetic mean value giving the expected or mean duration of the game. The theoretical results were evaluated using program 3 in Appendix 2. A comparison of predicted and experimental expected durations is given in Table 7.1 for varying p, q and k. For p = q = 0.5 1000 random walks were performed and for all other values of p and q only 500 walks were executed. Only the value of p is indicated in Table 7.1 because q is always 1-p.

Again /

Again the experimental results are in agreement with the predicted values, the worst discrepancy occurring at $p = q = 0.5$. This is because the variance of the distribution curve is greatest at this value and 1000 results is not sufficient to provide a more accurate estimation of duration. Because this particular experiment was performed manually, ie, the duration of each random walk was recorded manually, it was not practical to increase the number of readings. However the values are still reasonably accurate and show that the average simulator behaviour is predictable.

The final experiments with the random walk simulator were concerned with the probability of absorption at the nth trial. Equation (7.11) gives the theoretical value of the probability of absorption at the nth trial and may be evaluated for a range of n using program 4 in Appendix 2. This evaluation will give the probability distribution and is evaluated for a range of value of p, q and k. The experimental distribution curves were obtained using the results for the duration of the game and these are shown in Graph 7.2 to Graph 7.10 for different values of p, q and k.

Each graph has been shown in histogram form with the area under each bar representing the probability of being absorbed in the range of trials associated with the bar. In the cases of $p = 0.4$ and 0.6 each bar represents a range of 10 trials although only in 5 of these trials is absorption possible. For example if $p = 0.4$ and $k = 25$ then the probability of being absorbed at the upper boundary is virtually zero and so all absorptions will occur at the lower boundary. Therefore because $k = 25$ (an odd number) then absorption cannot occur in an even number of trials. Thus for $p = 0.4$ and k odd then only odd values of n may be allowed for in the averaging necessary to calculate the probability of absorption in each averaged range. Conversely if k is even then absorption will occur only at /

at even trials. The reverse is true if p = 0.6 when all absorptions will occur at the upper boundary. In this case, because a = 99, for even values of k then only at odd numbers of trials can absorption occur and for odd values of k absorption can occur only at even numbers of trials.

For p = q = 0.5 absorption is possible at any trial and both $p_{k,n}$ (the probability of absorption at the upper boundary at the nth trial) and $q_{k,n}$ (the probability of absorption at the lower boundary at the nth trial) must be evaluated. The theoretical distribution curves shown in Graphs 7.5, 7.6, and 7.7 (p = q = 0.5) are representative of the sum of $p_{k,n}$ and $q_{k,n}$. In this case of p = q = 0.5 each bar has been averaged over 100 trials, the larger average being required to condense the scale of the graph.

In all cases of p, q and k the experimental results form a skewed binomial distribution curve the mean of which gives the expected duration of the game. Again in all cases the peak and mean of the experimental distribution curves coincide closely with the predicted curves. In the case of p = q = 0.5 the magnitudes of both the experimental and theoretical curves coincide exactly although in all other values of p, ie, p = 0.4 and 0.6, there is a discrepancy in the magnitudes of the experimental and theoretical probabilities of absorption. This discrepancy may be due to the low number of samples (only 500) although the implication of the discrepancy is that the simulator is more accurate than predicted. This is evident by examining Graphs 7.2 to 7.4 and 7.8 to 7.10 which show that both the theoretical and experimental curves have approximately the same area but the experimental distribution is the sharper of the two distributions, ie, the variance of the experimental curve is less than that of the predicted curve.

These /

These experimental results together with the earlier results show clearly that the random walk simulator functions well and therefore will be a valuable asset in the solution of, for example, partial differential equations as will be demonstrated in Chapter 8.

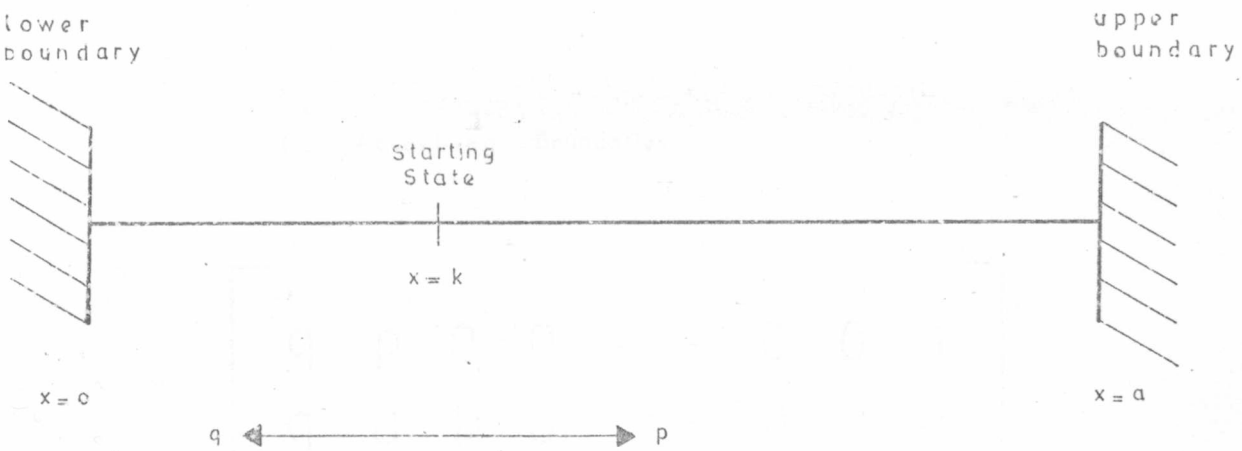


Figure 7·1 Symbolic Representation of Random Walk

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ q & 0 & p & 0 & \cdots & 0 & 0 & 0 \\ 0 & q & 0 & p & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdots & q & 0 & p \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

(a) Absorbing Boundaries

$$S = \begin{bmatrix} q & p & 0 & 0 & \cdots & 0 & 0 & 0 \\ q & 0 & p & 0 & \cdots & 0 & 0 & 0 \\ 0 & q & 0 & p & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdots & q & 0 & p \\ 0 & 0 & 0 & 0 & \cdots & 0 & q & p \end{bmatrix}$$

(b) Reflecting Boundaries

FIGURE 7·2 Random Walk Stochastic Matrix

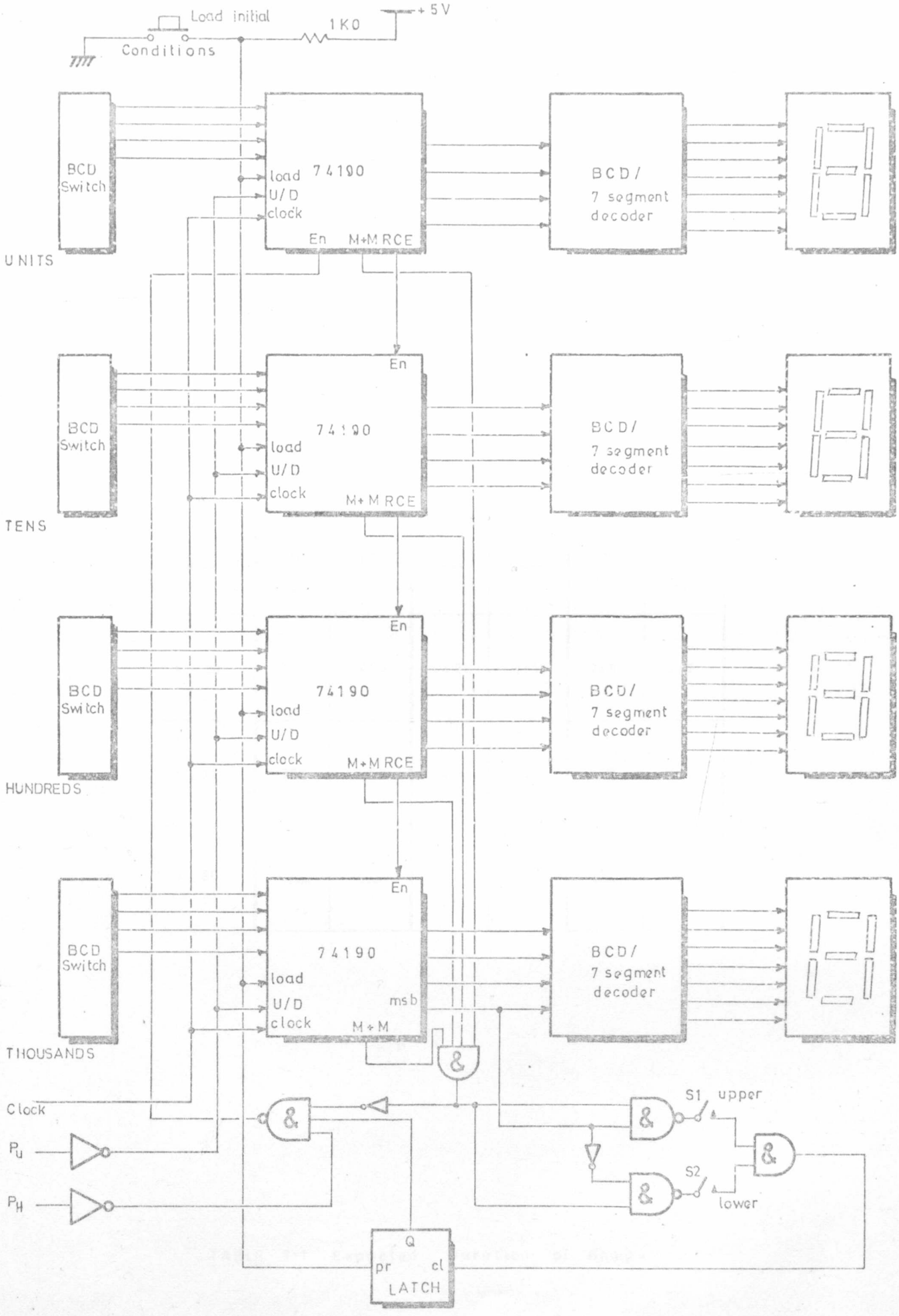
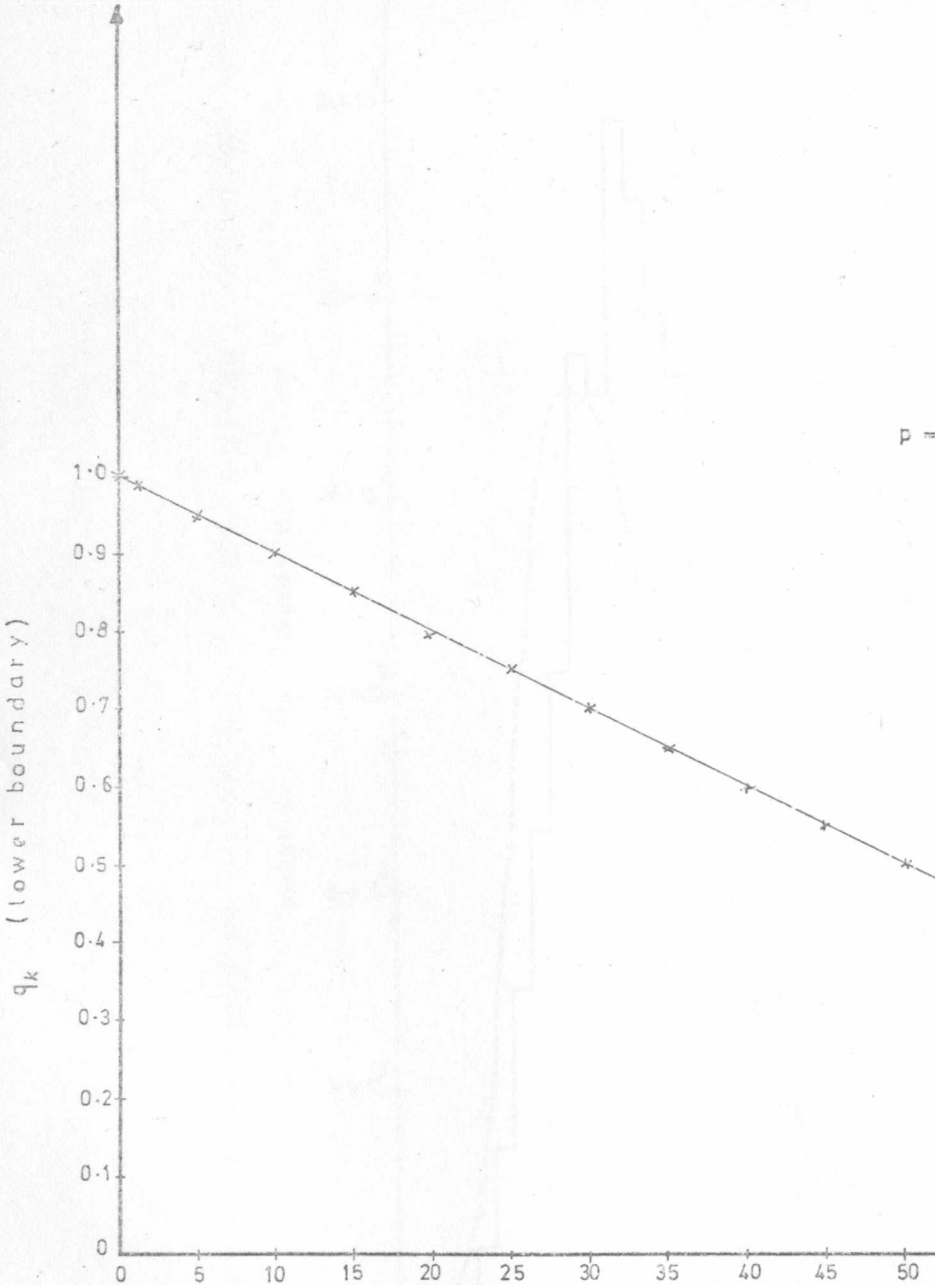


FIGURE 7·3 Random Walk Circuitry

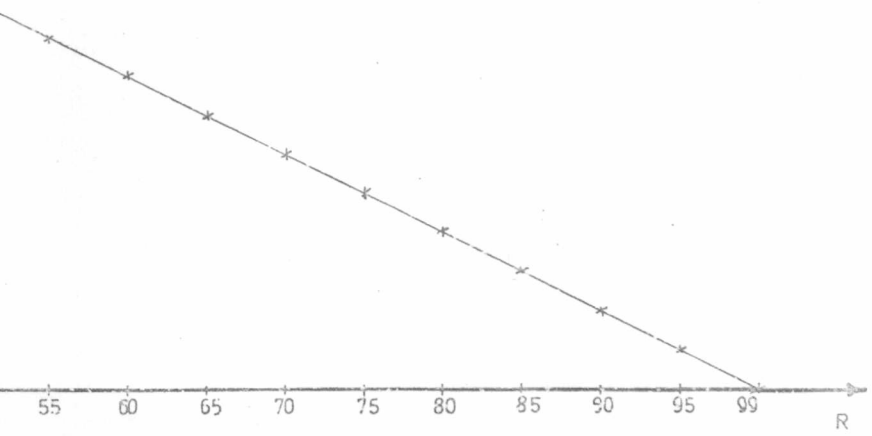
	p = 0·4		p = 0·5		p = 0·6	
	Theo.	Exp.	Theo.	Exp.	Theo.	Exp.
k = 20	—	—	1580	1655	—	—
k = 25	125	125	—	—	370	370
k = 50	250	246	2450	2516	245	247
k = 75	375	372	—	—	120	118
k = 80	—	—	1520	1418	—	—

TABLE 7·1 Expected Duration of Game



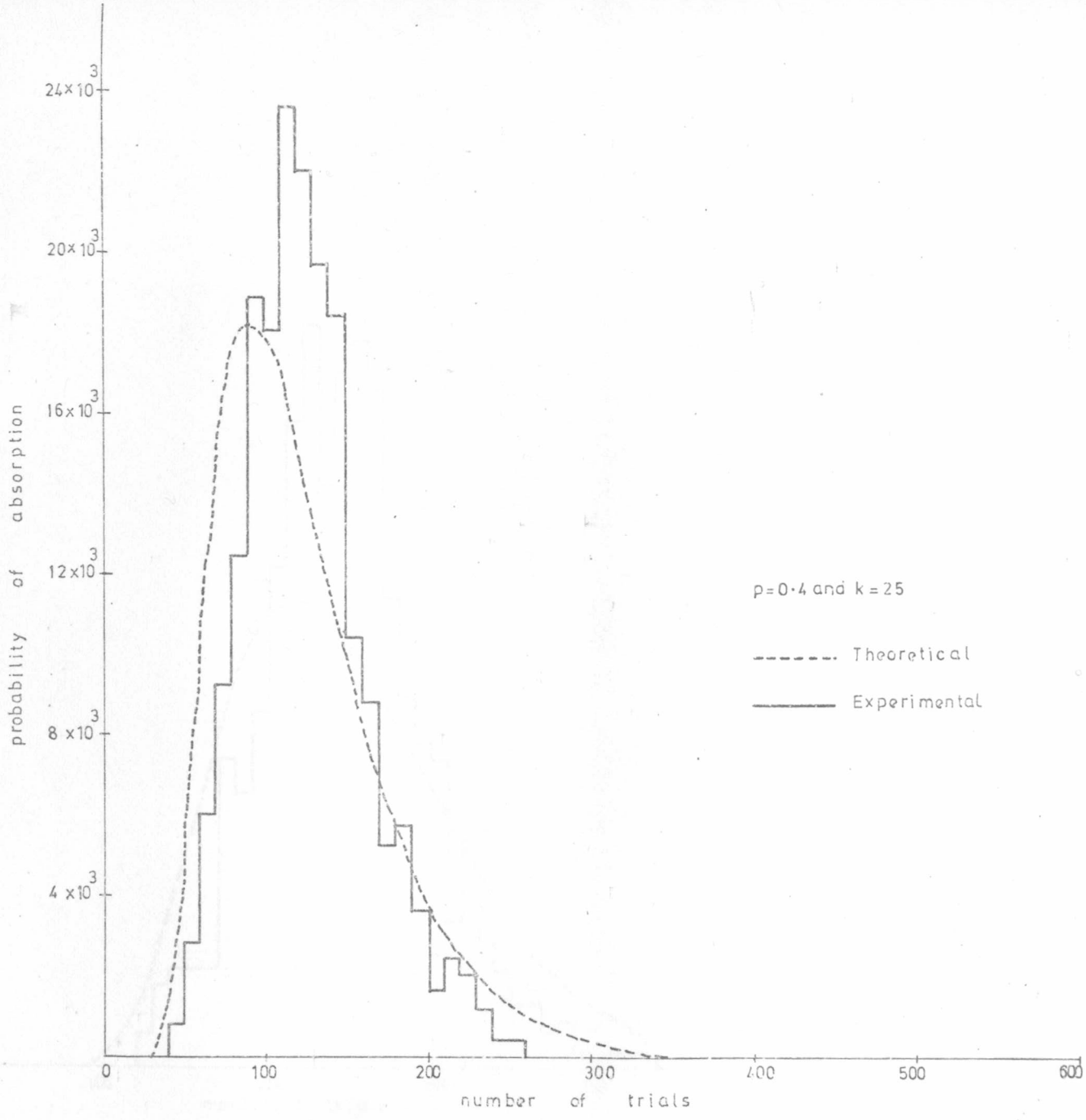
GRAPH 7·1 Probability

$q = 0.5$

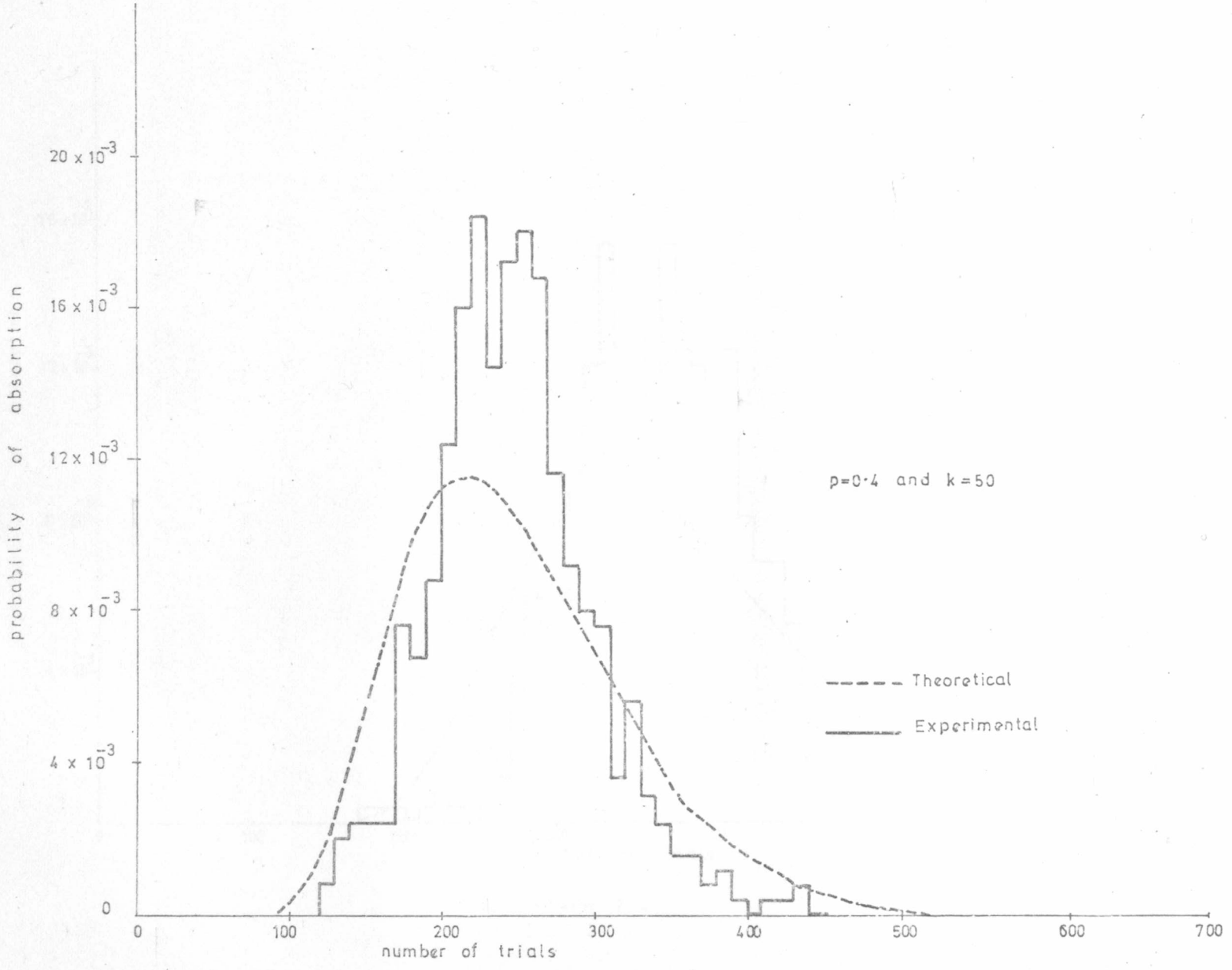


(k)

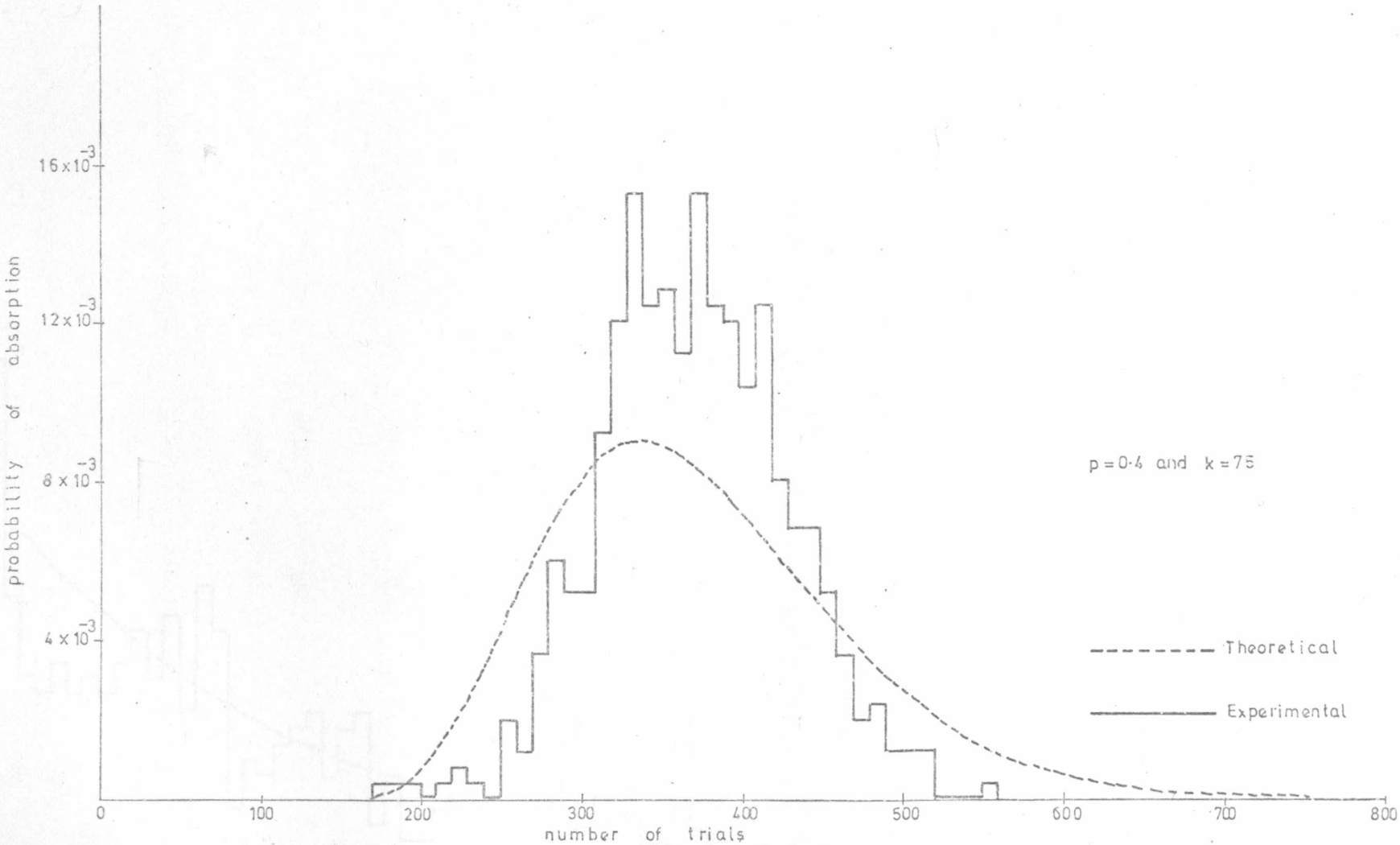
of Gambler's Ruin


24×10^3
20×10^3
16×10^3
12×10^3
8×10^3
4×10^3
probability of absorption
0
100
200
300
400
500
600
number of trials
p=0·4 and k=25
Theoretical
Experimental

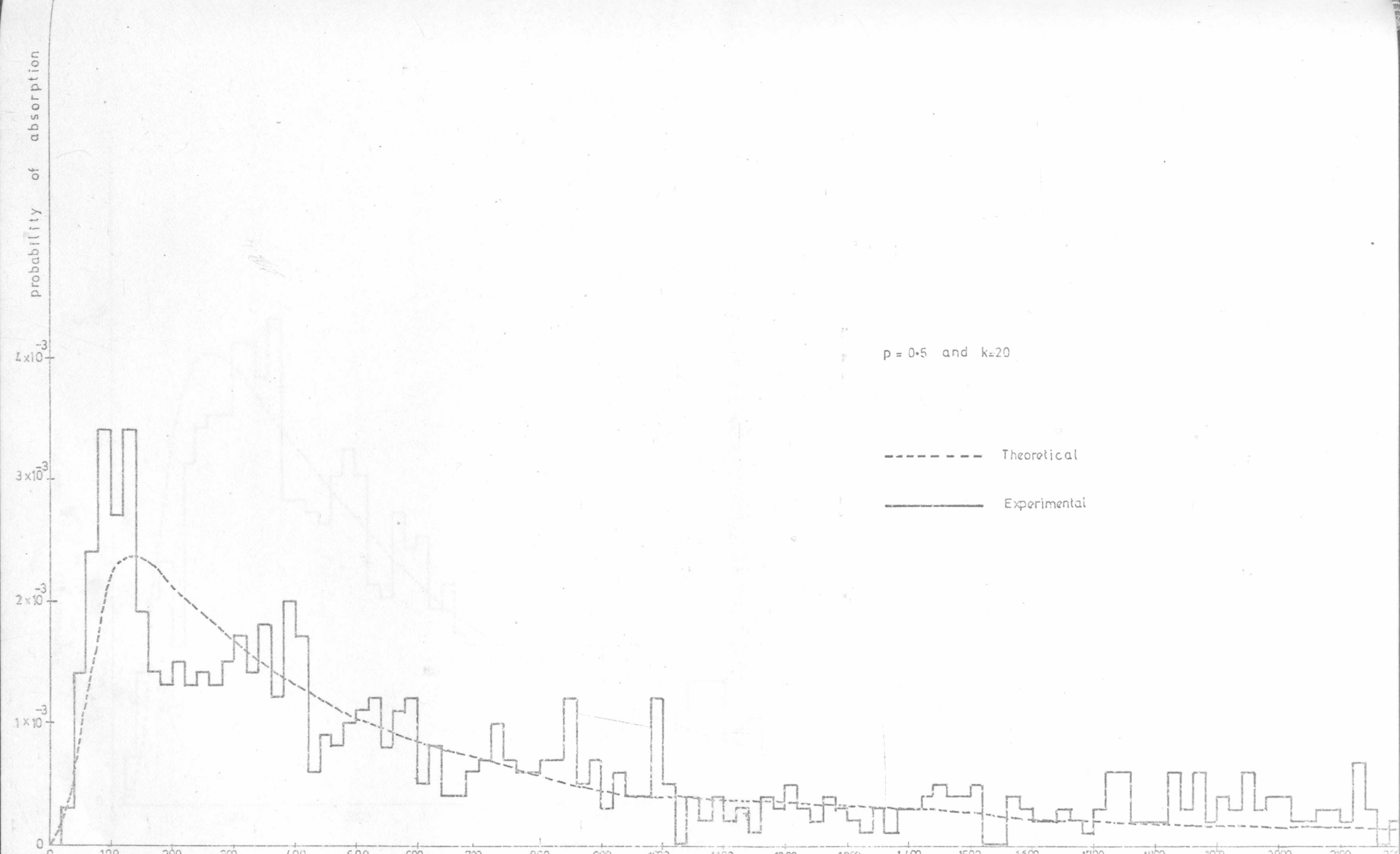
GRAPH 2·2


probability of absorption
20 x 10⁻³
16 x 10⁻³
12 x 10⁻³
8 x 10⁻³
4 x 10⁻³
0
0
100
200
300
400
500
600
700
number of trials
p=0·4 and k=50
Theoretical
Experimental

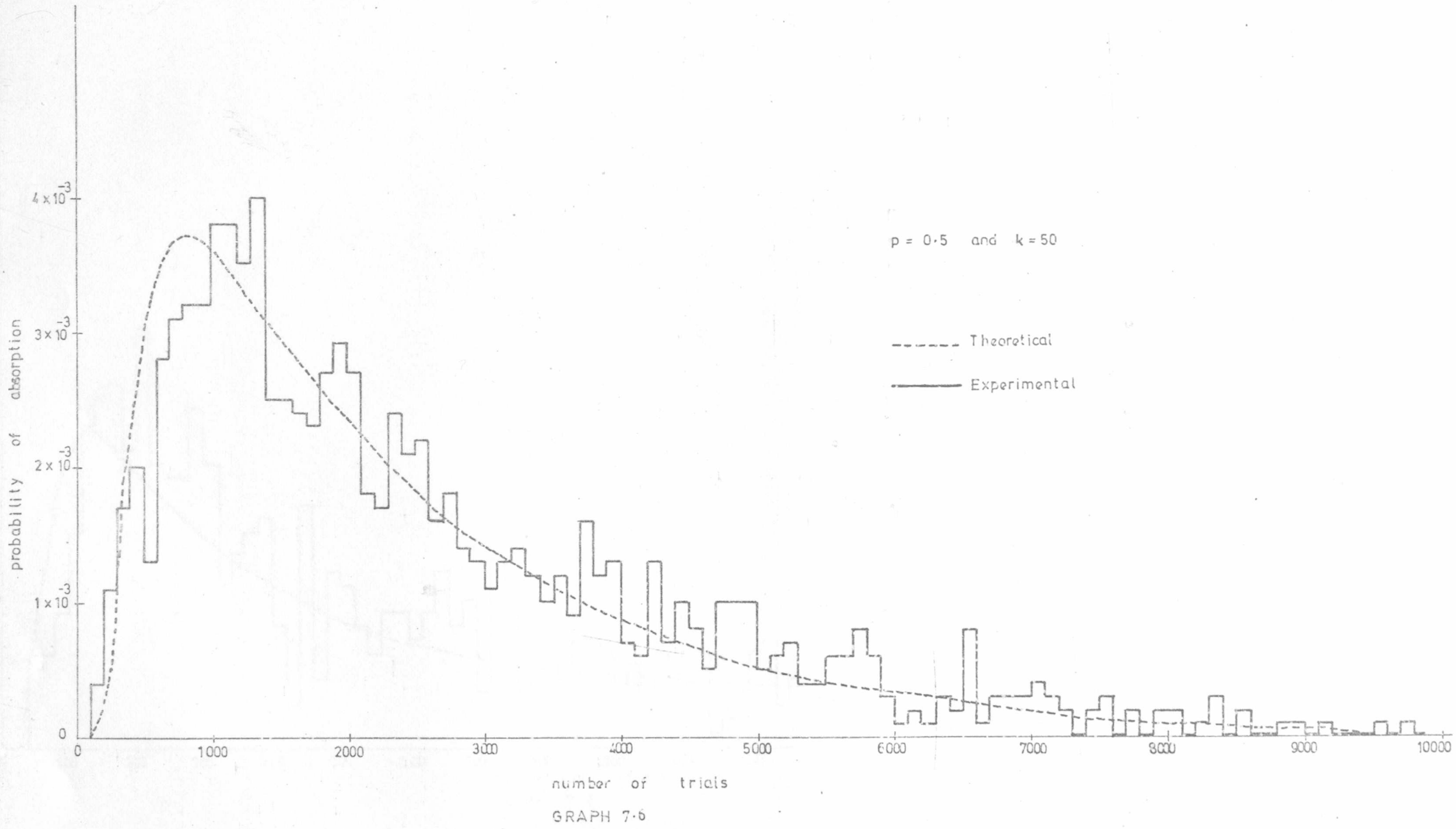
GRAPH 7·3


probability of absorption
16 x 10⁻³
12 x 10⁻³
8 x 10⁻³
4 x 10⁻³
0
100
200
300
400
500
600
700
800
number of trials
p = 0·4 and k = 75
Theoretical
Experimental

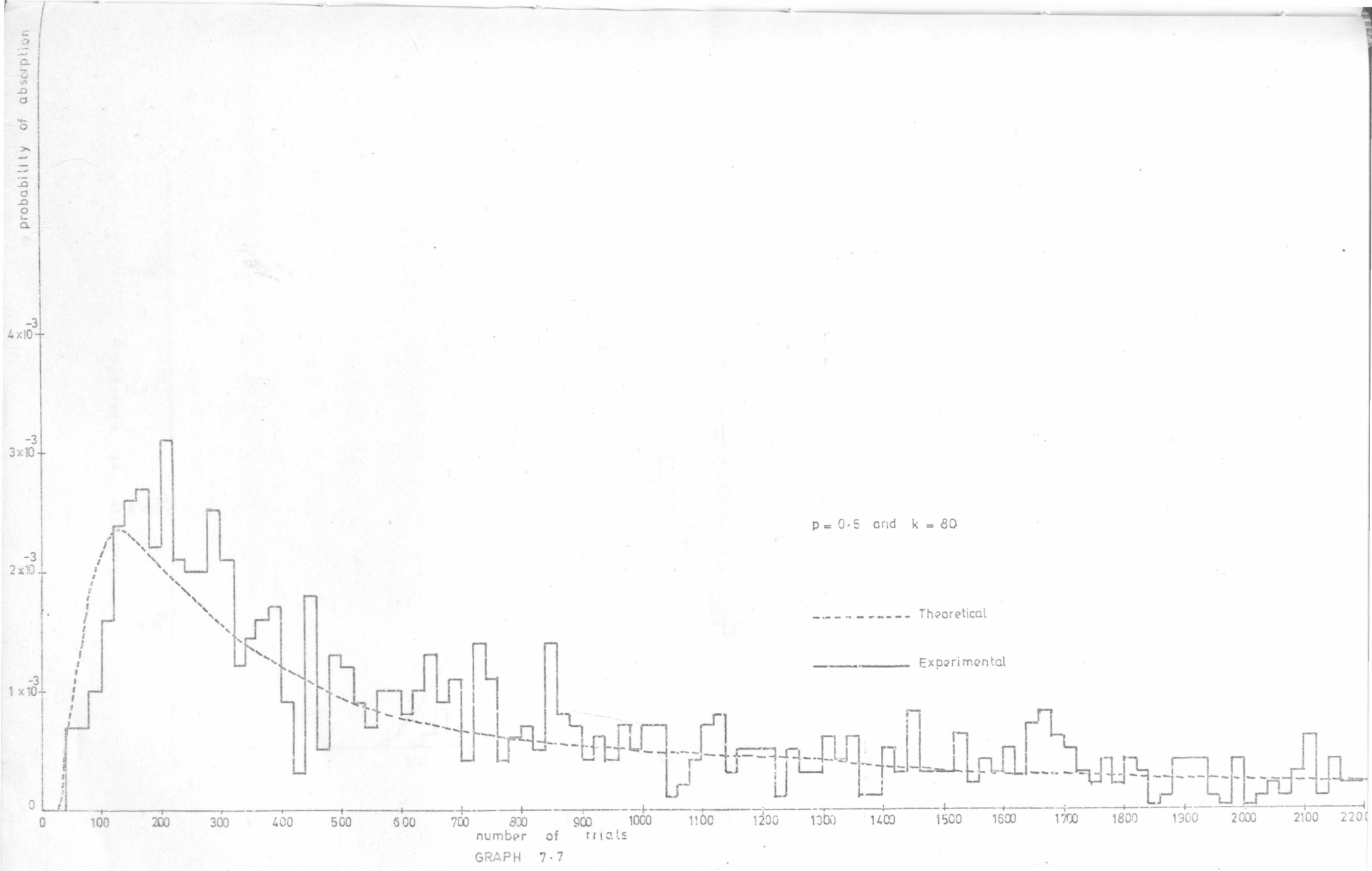
GRAPH 7·4


probability of absorption
4x10⁻³
3x10⁻³
2x10⁻³
1x10⁻³
0
p = 0·5 and k=20
Theoretical
Experimental
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000
number of trials

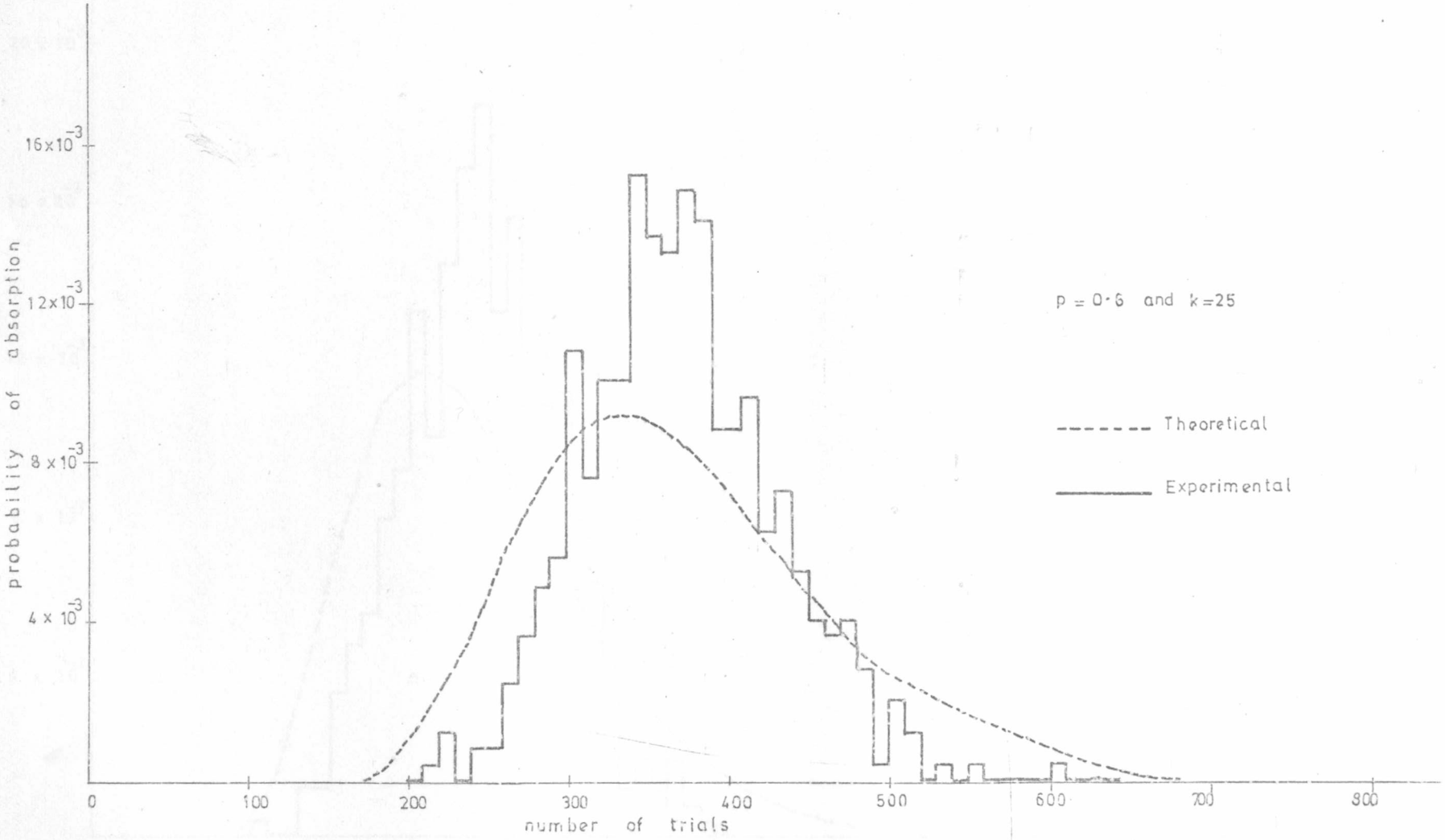
GRAPH 7 · 5


p = 0·5 and k = 50
Theoretical
Experimental
probability of absorption
4 x 10⁻³
3 x 10⁻³
2 x 10⁻³
1 x 10⁻³
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
number of trials

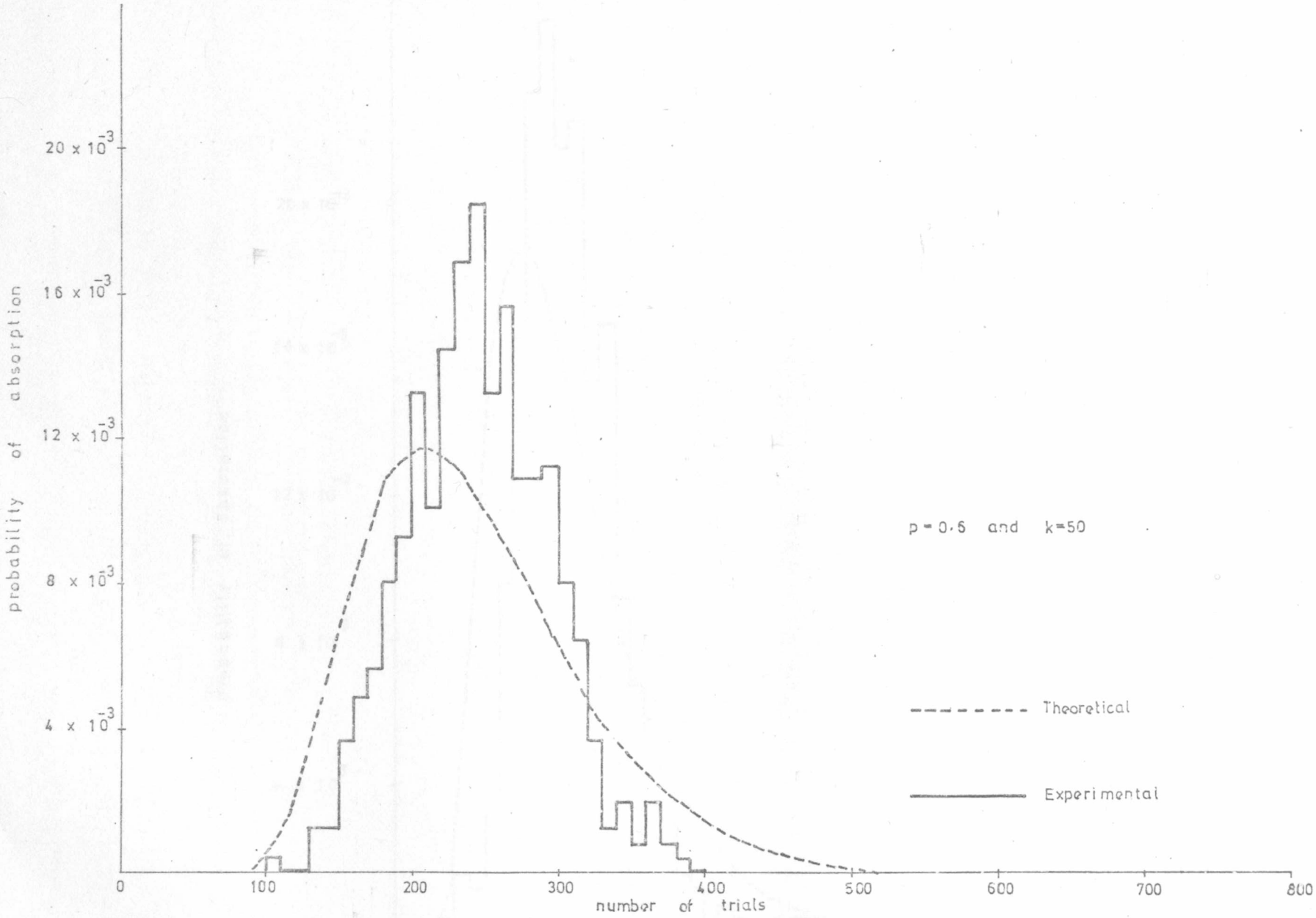
GRAPH 7·6


probability of absorption
4x10⁻³
3x10⁻³
2x10⁻³
1x10⁻³
0
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000
2100
2200
number of trials
p = 0·5 and k = 80
Theoretical
Experimental

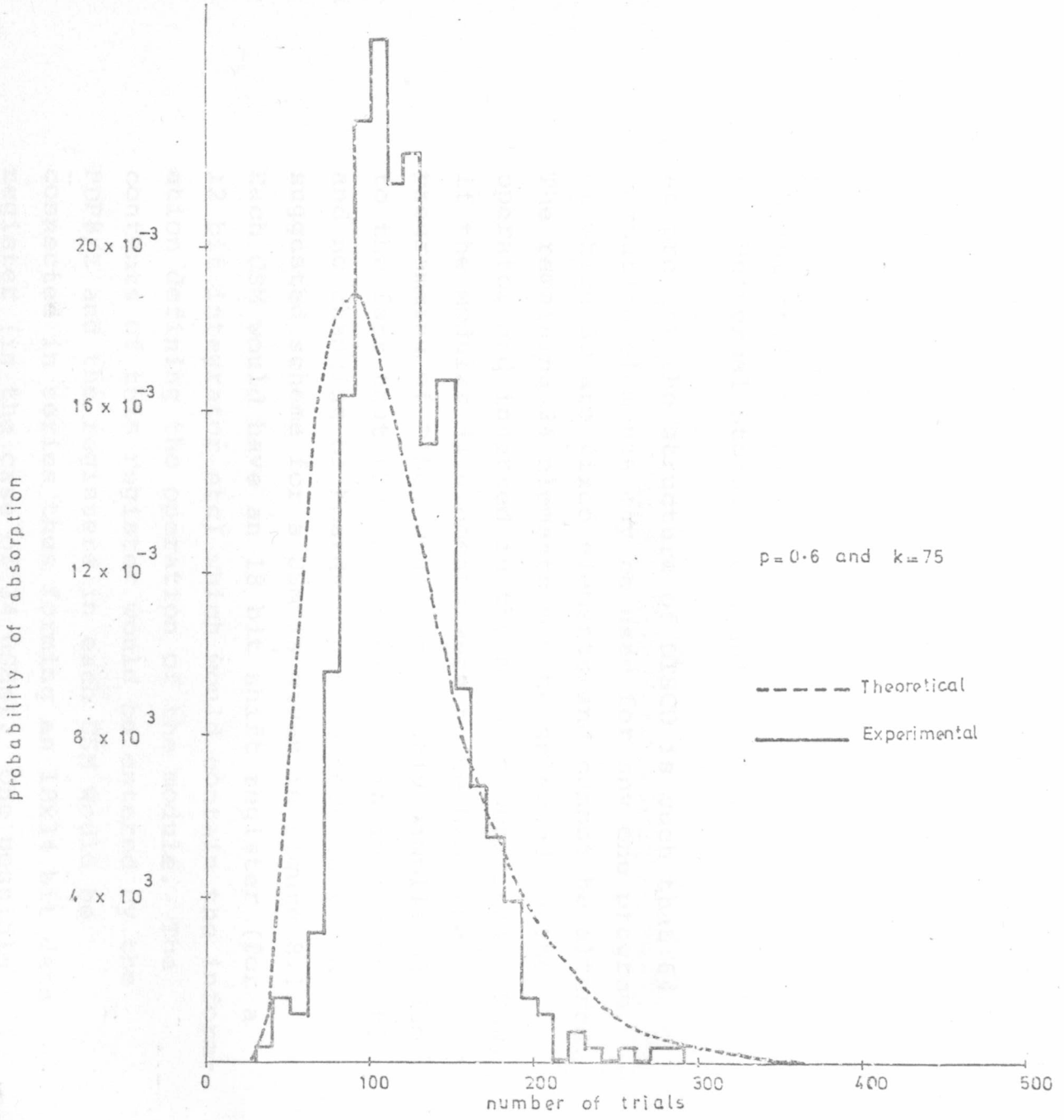
GRAPH 7·7


probability of absorption
16x10⁻³
12x10⁻³
8 x10⁻³
4 x 10⁻³
0
100
200
300
400
500
600
700
800
number of trials
p = 0·6 and k=25
Theoretical
Experimental

GRAPH 7·8


probability of absorption
20 x 10⁻³
16 x 10⁻³
12 x 10⁻³
8 x 10⁻³
4 x 10⁻³
0
100
200
300
400
500
600
700
800
number of trials
p = 0·6 and k=50
Theoretical
Experimental

GRAPH 7·9


20 x 10^-3
16 x 10^-3
12 x 10^-3
8 x 10^3
4 x 10^3
probability of absorption
0
100
200
300
400
500
number of trials
p = 0·6 and k = 75
Theoretical
Experimental

GRAPH 7·10

CHAPTER 8

FUTURE DEVELOPMENTS AND CONCLUSIONS

8.1 Universal Stochastic Module

As was seen from section 4.2 the use of a modular arrangement of the stochastic computing elements had the effect of increasing the computing capacity of DISCO. This modular approach although increasing the computing capability suffers from the disadvantage of the operator having to physically plug-in the various computing elements in the modular positions which for the solution of large scale problems may be time consuming. One solution to this problem, whilst still retaining the modular approach, is to use Universal Stochastic Modules (USM).

At present the structure of DISCO is such that 64 computing elements may be used for any one program of which 30 are fixed elements and cannot be altered. The remaining 34 elements may be selected by the operator and inserted in the modular positions. Now, if the modular 34 elements were to be USMs the programming of DISCO would be greatly simplified due to the fact that the USMs could remain in any position and no changing of boards would be necessary. A suggested scheme for a USM is given in Figure 8.1. Each USM would have an 18 bit shift register (for a 12 bit integrator etc) which would contain the information defining the operation of the module. The contents of this register would be entered by the PDP8/E and the registers in each USM would be connected in series thus forming an 18x34 bit data register (in the case of 34 USMs). One possible assignment code for the mode of operation is given in Table 8.1. The code bits m_0, m_1, and m_2 are the first three bits of the 18 bit data word associated with /

with each USM. As is seen from Figure 8.1 these three element code bits are decoded giving 8 control signals c_0, c_1, ..., c_7. The element code in Table 8.1 wa arranged such that if m_2 is zero then the element chosen does not use the n bit counter, thus simplifying the decoding circuitry required.

Each USM has two stochastic inputs E_1 and E_2 and the following signals are required to be derived from these two inputs for a given element. These signals may be verified by examination of the element equations listed in section 4.4.

(a) Invertor: $\bar{E}_1$ is simply derived from E_1 with E_2 not being used.

(b) Multiplier: again $\bar{E}_1$ and $\bar{E}_2$ are simply derived from E_1 and E_2.

(c) Squarer: in this case E_1 is in fact multiplied by a delayed version of itself (D) using the signals E_1, $\bar{E}_1$, D and $\bar{D}$.

(d) Summer: only a noise line (N) and its inverse $\bar{N}$ are required with E_1 and E_2.

(e) Noise ADDIE: E_1 and E_2 only are required.

(f) Integrator: again only E_1 and E_2 are required.

(g) Comparator: no derivations of E_1 and E_2 are required.

Thus the following signals have to be derived from E_1 and E_2 and these are $\bar{E}_1$, $\bar{E}_2$, D and $\bar{D}$. This is shown in Figure 8.1.

At /

At this point an alternative method of scaling to that of section 4.1 will be described. With the present method of scaling the effective length of an integrator counter may be varied thus varying the time constant of the integrator. As was suggested in section 4.1 the time constant may be varied by altering τ_1 the clock frequency. Thus to decrease the time constant by a factor of two, τ_1 would be doubled. Then to give scaling factors of 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$, $\frac{1}{64}$ and $\frac{1}{128}$ the clock may be divided by 1, 2, 4, 8, 16, 32, 64 and 128, ie, successive division by two, with the relevant clock rate being selected for operating the integrator counter. Each division of the clock rate by two is equivalent to increasing the counter length by one bit. Figure 8.1 shows the clock being divided successively by 2 and the outputs of each stage is applied to an 8 to 1 line data selector. The clock rate selected is determined by the scaling code which is the final 3 bits of the USM data word. These 3 bits S_2, S_1 and S_0 are applied to the code inputs of the data selector and the appropriate clock rate is selected. A possible scaling code is shown in Table 8.2 with the master clock rate being written as f_m.

The operation of the USM would proceed as follows. All programming of the USMs would be accomplished by means of the PDP8/E computer and its VDU terminal. For each USM to be used in a program the relevant information would be required by the PDP8/E so as to enable it to correctly establish the 18 bit data word. For example if the USM in position 10 were to be a summer then the bits m_2, m_1 and m_0 would have to be set to 0 1 1 respectively with all other bits of the 18 bit data word being set to zero. Thus the PDP8/E would only require the type of element providing the 12 bit counter is not required, ie, for the codes c_0, c_1, c_2 and c_3. If the 12 bit counter were required then /

then further information would be required. For example if the USM in position 15 were to be a noise ADDIE then the element code would have to be established initially. After this the initial condition of the counter would be required and would be used to set the contents of the next 12 bits of the data word. Finally the scaling factor would be typed, encoded and placed in the final 3 bits of the data word. The programming procedure for the integrator would be identical to that for the noise ADDIE. In the remaining case of a USM being required to function as a comparator the element code would again be established first. Secondly the value of the required comparator output would be placed in the next 12 bits of the data word, with the final 3 scaling bits being set to zero because they are not required.

All information pertaining to the USMs would be stored within the PDP8/E memory during programming and once this is completed the 18 bit data words would be transferred serially and in the correct order, to the data registers in each USM.

Once the data word has been entered into the data register the element code would be decoded and the relevant control signal would activate the appropriate gating. In the case of summation, invertion, multiplication or squaring the relevant derivations are routed to the output X to give the correct function. In the remaining cases the 12 bit counter must be incorporated. For the USM to function as an integrator the signals E_1 and E_2 must be combined and routed to the UP/DOWN and ENABLE inputs of the counter, the contents of which are compared with the 12 bit noise number. The output of the comparator is representative of /

of the integration of the sum of E_1 and E_2 and this signal is passed to output X. To load the initial conditions a control signal would be received from the PDP8/E and used to transfer, in parallel fashion, the contents of the appropriate 12 bits of the data word to the counter. The scale code would select the correct clock rate and so set the time constant of the integrator. Upon reception of a clear signal from the PDP8/E the contents of the 12 bit counter would be set to zero and the integration may be stopped and held at any time by application of a HOLD signal. This HOLD signal is applied to the enable input of the data selector, which selects the clock rate, thus having the effect of stopping the clock. The contents of the 12 bit counter would be available in binary form.

In the case of the USM operating as a noise ADDIE the operation is identical to the integrator described above, the only difference being that the inverse of the comparator output replaces E_2 with the counter contents thus being representative of the weighting of the stochastic sequence E_1. The scaling code in the case of the noise ADDIE could still be applicable if a choice of time constants were desirable.

Finally in the case of the USM operating as a stochastic comparator the weighting of the required output sequence is contained in the centre 12 bits of the data word. The relevant control signal c_6 continually applies a parallel LOAD control signal to the 12 bit counter thus transferring the 12 bits of weighting information to the comparator inputs. This LOAD command overrides all other counter control signals thus leaving the counter contents unchanged.

Having programmed the relevant USMs the patching operation would be performed as previously described.

One major advantage of this system is that by careful rearranging /

rearranging of circuit elements it is possible to construct the above USM using only one standard Verocard. This means that this approach could be adopted for use in the present stochastic computing system.

Another interesting feature of the USM is that providing the noise generation is incorporated within the module itself then only 22 inputs and outputs are required. Using modern LSI techniques it would not be a difficult task to produce an integrated circuit in a standard 24 pin package to perform the role of a USM. If this were to be accomplished then DISCO mkII would be very small indeed both in size and power requirements. This coupled with an LSI patching system and the fact that computing elements may be selected without the need to physically handle the system means that a future stochastic computer could be reduced to one or two circuit boards, which may easily be accommodated within a future digital computer. As a result a powerful hybrid computer no larger than a present mini-computer would become available, replacing very large, power consuming and most importantly, expensive hybrid systems.

8.2 Extensions to Markov Chain Simulator

The simulator described in Chapter Six has the ability to estimate the probability of state occupation after any number of transitions. However a brief glance at the theory of Markov Chains indicates that the operation of the simulator must be improved to enable the simulator to become a valuable and fully comprehensive system. For example, two of the more important features of a Markov Chain are the first return time and the first transition time. In /

In the case of the first return time it would be necessary to perform a large number of runs and in each run to record the number of transitions occurring before the simulator returns to its initial state. The recorded values would then be averaged to give an estimate of the first return time. This can easily be incorporated within the structure of the existing system. In the existing system the initial state is stored in a small memory (see Figure 6.6) and after each clock pulse, ie, transition, the contents of this memory could be compared to the outputs of the sequential network thus giving an indication when the simulator returns to the original state. A further addition to the simulator would be a counter to count the number of clock pulses and would be reset at the beginning of each run. When the system has returned to its original state the comparator output would be used to transfer the contents of the counter to the averaging circuitry. It is desirable to have the facility of measuring a number of parameters simultaneously and so when the first return has occurred the next run will not be initiated but the simulation will be allowed to run for its required time. (This time would be determined by the measurement of some other parameter, eg, measurement of the probability of being in any state after n clock pulses.) In this case it would be necessary to differentiate between the first return time and the second, third, etc. return time. This may easily be accomplished using a latch which would be cleared at the same time that the counter contents would be transferred. If this were not done the parameter which would be measured would be the average return time, a quantity which could be useful in future work.

In/

In the case of the estimation of the first passage time, the number of transitions required before a given state is reached, the procedure would be similar to that of the first return time. Again a counter would be required to record the number of transitions for each run and this would be reset at the beginning of each run. A second small memory would be required to store the state which is to be reached. The contents of this memory would be compared to the state of the sequential network and an output would be given when the required state has been reached, resulting in the transfer of the counter contents to the averaging circuitry. In this case it is important that a latch be incorporated to differentiate between the first passage time and any other passage time. If this were not done the parameter which would be measured would be the average return time to the state indicated by the contents of the second memory.

The two parameters discussed above may easily be measured using the methods described, resulting in a fully comprehensive and powerful simulator, with very little additional circuitry being required.

As was mentioned in section 6.1 Markov Chain theory has been well documented and developed and there are already a large number of possible applications in the field of Operational Research. One field in which the Markov Chain simulator could prove successful is that of learning systems. With these systems the parameters describing the system are continually being changed in accordance with punishment and reward criteria. Such a system could easily be simulated using the Markov Chain simulator, it being a simple task to alter the 12 driving probabilities in accordance with the system behaviour.

8.3 Solution of Partial Differential Equations Using the Random Walk Simulator[11,12]

The random walk simulator described in Chapter Seven will prove most useful in the solution of multi-dimensionable partial differential equations (PDEs) using Monte Carlo techniques.[11] In fact the solution of a single dimension PDE has already been found in the course of the experimental results taken to verify the performance of the simulator (see Graph 7.1). The results shown in Graph 7.1 will now be shown to be the solution of a single dimension PDE.

Consider Laplace's equation in one dimension

$$\frac{\partial^2 u}{\partial x^2} = 0 \qquad \text{---- (8.1)}$$

with boundary conditions

$$u(-10) = +10$$

$$u(10) = -10$$

Equation (8.1) is an elliptical PDE and an approximate solution may be obtained[12] by using the difference equation

$$\frac{\partial^2 u}{\partial x^2} \doteq \frac{1}{h^2}\left[u(x+h) - 2u(x) + u(x-h)\right] \qquad \text{---- (8.2)}$$

where h is the step size and in the case of equation (8.1) we have

$$\frac{\partial^2 u}{\partial x^2} \doteq \frac{1}{h^2}\left[u(x+h) - 2u(x) + u(x-h)\right] = 0 \qquad \text{---- (8.2)}$$

From section 7.2 the difference equation used to develop /

develop the expression for the probability of ruin was

$$q_k = pq_{k+1} + qq_{k-1}$$

where p is the probability of a win, q is the probability of a loss and q_k is the probability of ruins with starting state k. If $p = q = \frac{1}{2}$ then

$$q_k = \tfrac{1}{2}(q_{k+1} + q_{k-1})$$

and therefore

$$\tfrac{1}{2}(q_{k+1} - 2q_k + q_{k-1}) = 0 \qquad \text{---- (8.3)}$$

Equations (8.2) and (8.3) are identical in form and evaluating q_k will give a solution for u(x) where k will be representative of x. Using the notation of section 7.2 the maximum value of x will be a and the minimum value of x will be 0.

In the case of equation (8.1) the boundary conditions are u(-10) = +10 and u(10) = -10. Thus the value of x lies in the range -10 to +10 which corresponds to k = 0 to k = a and the value of u(x) lies in the range -10 to +10 which corresponds to q_k = 0 to q_k = 1. As was previously mentioned the graph of q_k against k (Graph 7.1) gives the value of $\frac{\partial^2 u}{\partial x^2}$ for the range of x. Graph 7.1 is shown with rescaled axes as Graph 8.1 and the experimental values are seen to be in close agreement with the theoretical results.[11]

This technique for the solution of PDEs may easily be extended to three dimensions simply by using three random walk simulators and some additional circuitry to record the number of absorptions at each of the six boundaries.

One /

One important future application of random walk simulators is the solution of multi-dimensional integrals[12] which is an extremely complicated problem and indeed no acceptable computing technique exists for such problems.

A restricting feature of the present random walk simulator is that the boundaries are fixed and so only problems with fixed rectangular boundaries (in two dimensions) may be solved. This problem is discussed in the next section and two methods of introducing variable boundaries are discussed.

8.4 Random Walk Simulator with Variable Boundaries

At present the random walk simulator operates within rectangular boundaries for the two dimensional case and within cubical boundaries for the three dimensional case. For the solution of some PDEs it is desirable to have variable boundaries and this feature may be incorporated using one of two methods. Firstly the boundary conditions would be stored within a Read Only Memory (ROM) the output of which would indicate when a boundary has been reached. For example in the two dimensional case the coordinates of the present state are contained in the two simulator counters. Then for a 100 state simulator there would be two eight bit data words, one for each coordinate, which would be fed to the ROM address inputs. The ROM would function as a 'look up' table and a logic '1' would be stored in each memory location corresponding to each boundary state, all other locations containing logic '0'. If the address inputs are presented with the coordinates of a boundary state then the logic '1' output from the ROM would indicate this fact. The main disadvantage of this method is that, for each required boundary a separate ROM would be required or at best a single ROM would be required to be reprogrammed for each PDE to be solved. This would

either /

either be very expensive or time consuming.

An alternative method of defining the boundaries of a random walk is to use the stochastic computer to define the boundary conditions. This is only possible if the boundaries are mathematically definable, eg, a circle, rectangle, ellipse, rhomboid, etc in the two dimensional case. Consider the two dimensional random walk with the required boundary being a circle. The coordinates of the present state of the random walk would have to be represented by two stochastic sequences, one for each coordinate. This may easily be accomplished by comparing the contents of each counter with a random number as described in section 1.6. The simulator counter would have to be a binary counter and not, as is the present case, a decade counter in order to give accurate stochastic sequences. Both stochastic sequences would be manipulated and used to determine whether or not a boundary has been reached.

Consider the equation of a circle which is

$$(x-x_0)^2 + (y-y_0)^2 = r^2$$

where x and y are the present coordinates, (x_0, y_0) is the coordinates of the centre of the circle, and r is the radius.

The quantity $(x-x_0)^2 + (y-y_0)^2$ is easily generated using the configuration shown in Figure 8.2. Each x and y coordinate is represented by the stochastic sequences X and Y respectively which are generated by comparing the contents of the simulator counters to random numbers. The centre of the circle is defined by the values of X_0 and Y_0 which are generated by stochastic comparators. In each case where the output of an element is given, allowance has been made for the normalising of the output signal, eg, the output of the summer is a stochastic sequence representing one half of the true sum. Thus the output stochastic sequence represents the quantity

quantity

$$[(x-x_0)^2 + (y-y_0)^2]/8V$$

which is a scaled version of the required result. The quantity V is the maximum value, ie, the range of a stochastic sequence (probability 0 to 1) represents the range V. If this is compared to the value of $r^2/8V$ then the output of the comparator will indicate when a boundary has been reached. This would be achieved by converting the output stochastic sequence to binary form, ie, by using an ADDIE, and comparing this to the binary representation of $r^2/8V$. The value of $r^2/8$ would be contained in for example a shift register loaded by the PDP8/E. A separate board could easily be constructed to perform this function and provided the correct pin convention is adopted this board could be treated as another element and could thus be accommodated in a modular position. The value of $r^2/8V$ could be loaded by regarding the element as a stochastic comparator for the purpose of loading information.

This method can be adopted for any mathematically definable boundary and in the case of for example an ellipse the value of the radius r would vary and so further computation would be required. Nevertheless with the exception of the comparator all the computing elements have been built and so there would be no extra cost of any significance.

The major disadvantage of this method of providing variable boundaries is that a finite time would be required to convert the contents of the simulator counter to a stochastic sequence and to convert the required stochastic sequence to binary form. One means of achieving this would be to use two clock frequencies, one for the stochastic computer and a slower one for the random walk simulator. The likely optimum ratio of the two clock frequencies would be in the region of 1000:1. If the simulator clock frequency were too

fast /

fast compared to the stochastic computer clock frequency then insufficient time would be available to estimate whether or not a boundary had been reached resulting in the possible crossing of a boundary by the simulation, ie, overshoot. On the other hand if the simulator clock frequency was relatively too slow then although the boundary would be accurately described, the time taken for a single random walk would be great. This would clearly be undesirable if a large number of random walks were required which would be the case in the solution of PDEs. Thus there must be a trade-off between speed of operation and accuracy of defining the boundaries and further work is required to determine an optimum ratio of clock pulses. This disadvantage does not occur with the use of a ROM to define the boundaries.

Another disadvantage of using the stochastic computer is that the random walk simulator would have to be rebuilt using a binary counter as opposed to a decimal counter because of the need to generate a stochastic sequence representing the counter contents. This would mean that a display of the occupied state could not be presented using 7-segment displays. For the solution of PDEs this is not important and the simulator for each dimension could be constructed on one board which could be accommodated in a modular position. This would result in the existance of several new modules which would then be used as special purpose elements for the solution of PDEs.

Thus the use of the stochastic computer as a means of defining the boundaries of a random walk is preferable except in the case of high speed together with high accuracy being required and in the case where the boundaries cannot be mathematically described.

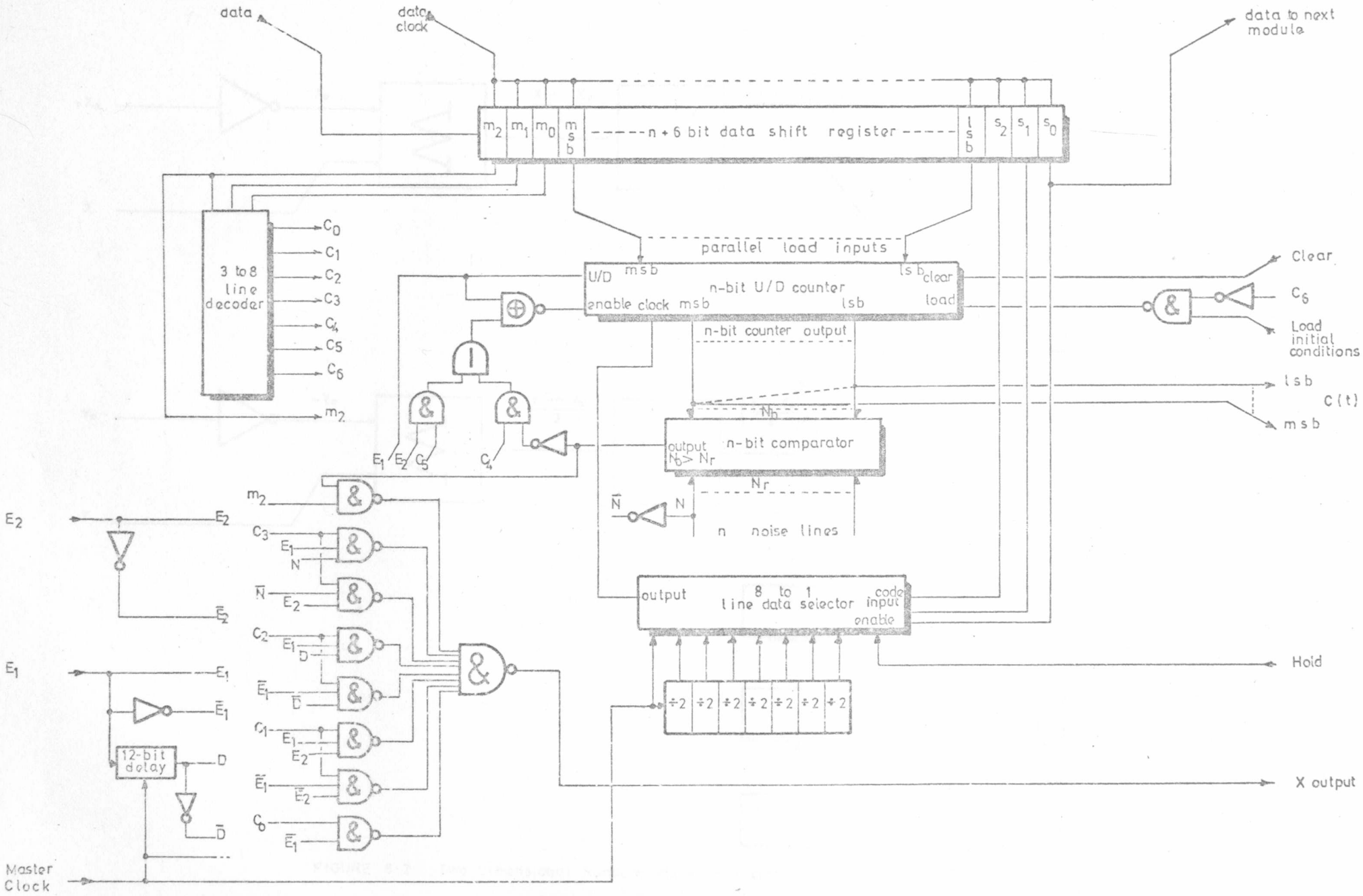


Figure 8·1 Universal Stochastic Module

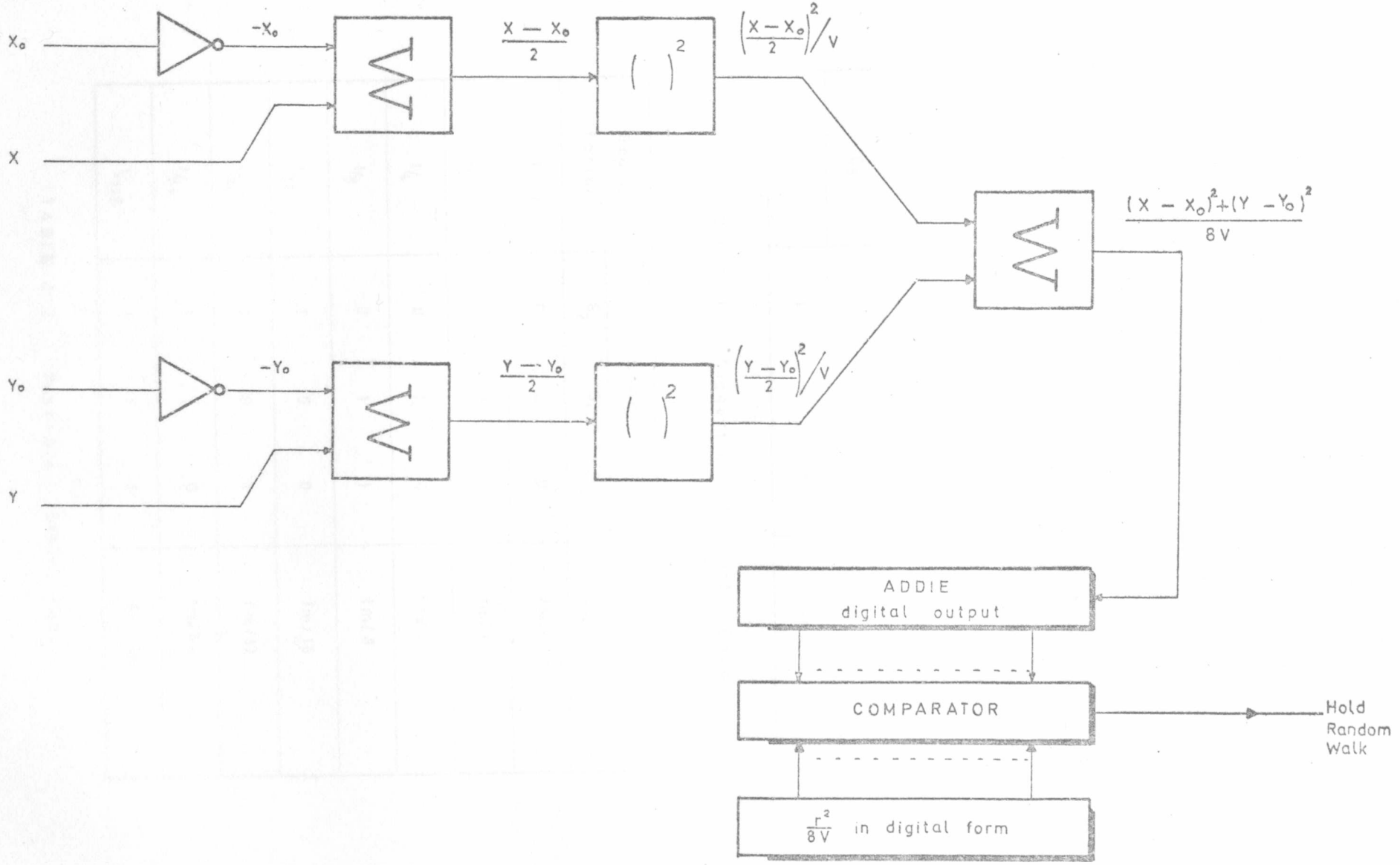


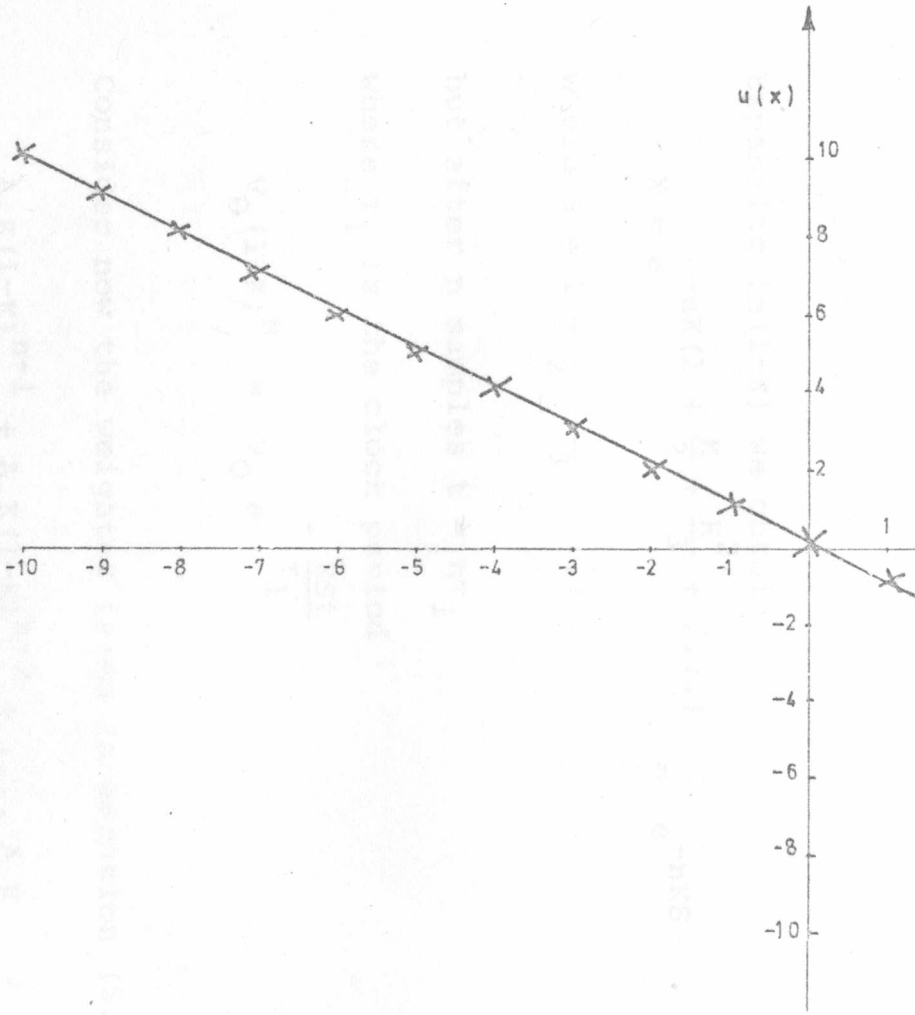
FIGURE 8·2 Two Dimensional Random Walk with circular boundaries

Control Signal	code			Element
	m_2	m_1	m_0	
C_0	0	0	0	invertor
C_1	0	0	1	multiplier
C_2	0	1	0	squarer
C_3	0	1	1	summer
C_4	1	0	0	noise ADDIE
C_5	1	0	1	integrator
C_6	1	1	0	comparator
C_7	1	1	1	none

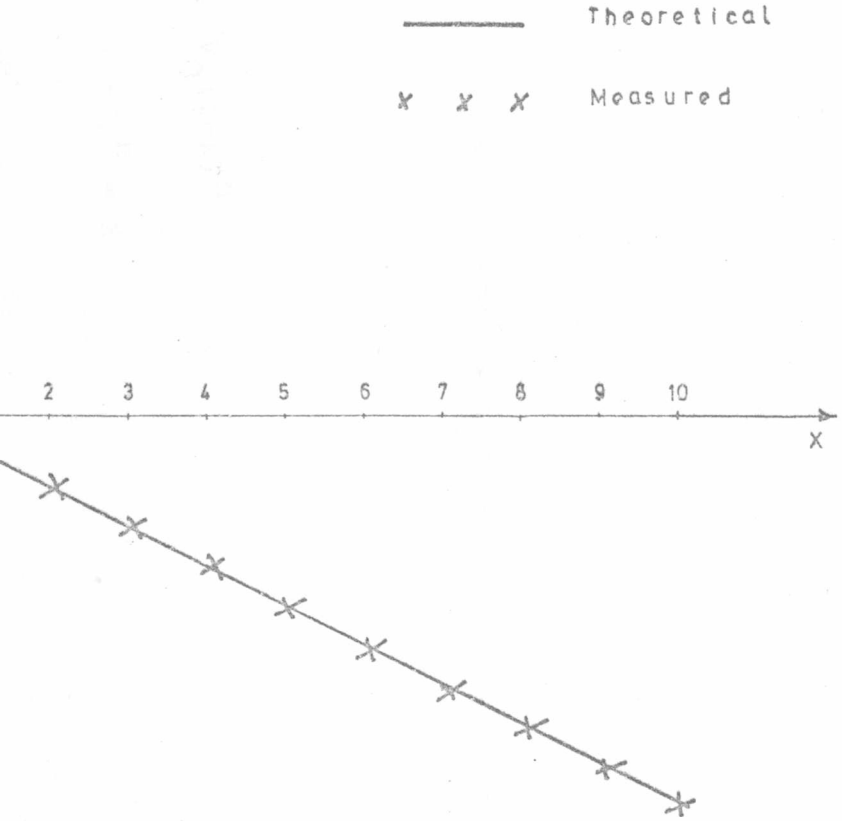
TABLE 8.1 Suggested Element Code

Scale factor	code			Selected Clock Rate
	S_2	S_1	S_0	
1	0	0	0	fm
$1/2$	0	0	1	fm/2
$1/4$	0	1	0	fm/4
$1/8$	0	1	1	fm/8
$1/16$	1	0	0	fm/16
$1/32$	1	0	1	fm/32
$1/64$	1	1	0	fm/64
$1/128$	1	1	1	fm/128

TABLE 8.2 Possible Scale Code



GRAPH 8 1 Solution of One



Dimensional P. D. E.

APPENDIX 1

TIME SOLUTION TO STEP RESPONSE OF STOCHASTIC TO ANALOGUE CONVERTOR

From section 5.3 the normalised voltage output of the smoothing network was

$$v(n) = V_o(1-K)^n + A_1K(1-K)^{n-1} + A_2K(1-K)^{n-2} \ldots\ A_nK \qquad \text{---- (5.10)}$$

Consider the term due to the initial voltage

$$V_o(1-K)^n \qquad \text{---- (A1.1)}$$

Let $(1-K)^n = X$

$$\ln X = n\ln(1-K)$$

$$X = e^{n\ln(1-K)}$$

Expanding $\ln(1-K)$ we obtain

$$X = e^{-nK(1 + \frac{K}{2} + \frac{K^2}{3} + \ldots.)} = e^{-nKS}$$

where $S = 1 + \frac{K}{2} + \frac{K^2}{3} + \ldots..$

but after n samples $t = n\tau_1$

where τ_1 is the clock period

$$V_o(1-K)^n = V_o\, e^{-\frac{KSt}{\tau_1}} \qquad \text{---- (A1.2)}$$

Consider now the weighted terms in equation (5.2)

$$A_1K(1-K)^{n-1} + A_2K(1-K)^{n-2} + \ldots. A_nK$$

If we take $A_1 = A_2 = A_3 = p(A)$

where /

where p(A) is the probability of the stochastic sequence to be converted, then the sequence will be averaged over n clock pulses, ie, we take expected values.

Thus

$$v(n) = p(A)\ K[\ (1-K)^{n-1} + (1-K)^{n-2} \ldots . (1-K) + 1)]$$

ignoring the initial conditions term.

$$v(n) = p(A)\ K\left(\sum_{i=0}^{n-1} (1-K)^{i}\right)$$

and converting this to an integral we obtain

$$v(n) = p(A)\ K \int_{0}^{n-1} (1-K)^{n}\, dn \qquad \text{---- (A1.3)}$$

To convert this integration to one with respect to time we have $t = (n-1)\tau_1$ where τ_1 is one clock period

$dn = \frac{dt}{\tau_1}$ and (A1.3) becomes

$$v(t) = \frac{p(A)K}{\tau_1} \int_{0}^{\frac{t}{\tau_1}} (1-K)^{\frac{t}{\tau_1}}\, dt$$

$$= \frac{p(A)K}{\tau_1 \ln(1-K)^{\frac{1}{\tau_1}}} \left[\ (1-K)^{\frac{t}{\tau_1}} - 1\right]$$

but

$$\ln(1-K) = -K\left(1 + \frac{K}{2} + \frac{K^2}{3} + \ldots . \right)$$

For convenience let $S = \left(1 + \frac{K}{2} + \frac{K^2}{3} + \ldots . \right)$

$$v(t) = \frac{p(A)\ K\tau_1}{\tau_1\ (-KS)} \left[\ (1-K)^{t/\tau_1} - 1\right]$$

$$= \frac{p(A)}{S} \left[\ 1 - (1-K)^{t/\tau_1}\right] \qquad \text{---- (A1.4)}$$

Let /

Let $(1-K)^{t/\tau_1} = Y$

then

$$\ln Y = t/\tau_1 \ln(1-K)$$

$$Y = e^{t/\tau_1 \ln(1-K)} = e^{-t/\tau_1 KS}$$

Substituting in (A1.4)

$$v(t) = \frac{p(A)}{S}[1 - e^{-t/\tau_1 KS}] \qquad \text{---- (A1.5)}$$

Combining (A1.5) and (A1.2) we obtain a complete time solution

$$v(t) = \frac{p(A)}{S}[1 - e^{-\frac{KS}{\tau_1}t}] + V_o e^{-\frac{KS}{\tau_1}t} \qquad \text{---- (A1.6)}$$

The series S is given as

$$S = 1 + \frac{K}{2} + \frac{K^2}{3} + \ldots\ldots$$

If K is very small

$$S = 1.$$

This is justified by considering the following example. The value of K is

$$(1-e^{-\frac{\tau_1}{\tau_2}})$$

where τ_2 is the time constant of the filter and τ_1 is the period of a clock pulse.

Say $\tau_2 = .001S$ (bandwidth 150 Hz)

and $\tau_1 = 1\ \mu S$ (clock frequency 1 MHz)

then $K = .001$

$$S = 1 + \frac{10^{-3}}{2} + \frac{10^{-6}}{3} + \ldots\ldots$$

$$= 1.0005.$$

Thus it can be seen that a typical value of K gives $S = 1$ with no appreciable error.

The approximate solution to a step response is therefore

$$v(t) = p(A)(1 - e^{-\frac{Kt}{\tau_1}}) + V_o e^{-\frac{Kt}{\tau_1}} \qquad \text{---- (A1.7)}$$

APPENDIX 2

PROGRAM 1

EVALUATION OF MARKOV CHAIN TRANSIENTS

```
 10 PRI "DETERMINATION OF MARKOV CHAIN TRANSIENTS USING THE"
 20 PRI "FOLLOWING MATRIX."
 30 FOR I=1 TO 4\FOR J=1 TO 4
 40 READ P(I,J)\PRI P(I,J),\NEX J
 50 PRI\NEX I
 60 PRI\PRI
100 FOR Q=1 TO 4
110 FOR I=1 TO 4\A(I)=Ø
120 NEX I
130 A(Q)=1\N=Ø
140 PRI\PRI "NO. OF OPS.";
150 PRI "     Q1          Q2          Q3          Q4"
160 PRI "       ";N,A(1),A(2),A(3),A(4)
170 N=N+1
180 FOR J=1 TO 4\S(J)=Ø
190 FOR I=1 TO 4\S(J)=S(J)+A(I)*P(I,J)
200 NEX I
210 NEX J
220 FOR J=1 TO 4\A(J)=S(J)
230 NEX J
240 IF N<41 THEN 160
250 NEX Q
260 STOP
```

PROGRAM 2

EVALUATION OF PROBABILITIES OF ABSORPTION AT Ø AND A

```
 50 PRI "INPUT A,P"
 60 INP A,P
 70 Q1-P
 80 IF P=0.5 THEN 140
 90 FOR K=5 TO 95 STEP 5
100 Q1=((Q/P)^A-(Q/P)^K)/((Q/P)^A-1)
110 PRI K;TAB(20);Q1;TAB(40);1-Q1
120 NEX K
130 GOTO 190
140 FOR K=5 TO 95 STEP 5
150 Q1=1-K/A
160 P1=K/A
170 PRI K;TAB(20);Q1;TAB(40);P1
180 NEX K
190 PRI\PRI\PRI\PRI
200 PRI"TYPE 1 TO RUN AGAIN"
210 INP Y
220 IF Y=1 THEN 50
230 STOP
```

PROGRAM 3

EVALUATION OF DURATION

```
 50 PRI "INPUT A,P"
 60 INP  A,P
 70 Q=1-P
 73 PRI\PRI\PRI\PRI
 74 PRI TAB(10);"  K";TAB(25)"DURATION FOR P=";P
 75 IF P=.5 THEN 140
 80 FOR K=5 TO 95 STEP 5
 90 D=K/(Q-P)-A/(-P)*((1-(Q/P)^K)/(1-(Q/P)^A))
110 PRI TAB(10);K;TAB(30);D
120 NEX K
130 GO TO 200
140 FOR K=5 TO 95 STEP 5
150 D=K*(A-K)
170 PRI TAB(10);K;TAB(30);D
180 NEX K
200 PRI\PRI\PRI\PRI
210 PRI "TYPE 1 TO RUN AGAIN"
220 INP Y
230 IF Y=1 THEN 50
240 STOP
```

PROGRAM 4

PROBABILITY OF ABSORBTION AT EACH TRIAL

```
10 PRI "INPUT A,P,Q,K"
20 INP A,P,Q,K
30 PRI "MIN N,MAX N,STEP SIZE"
40 INP M,X,S
50 PRI "SUMMATION LIMIT"
60 INP Y
65 I=3.141593
70 FOR N=M TO XSTEP S
80 Z1=(4*P*Q)^(N/2)*(Q/P)^(K/2)/A*2
85 Z2=Z1*((4*P*Q)^.5)
90 B=(P/Q)^(A/2)
93 S1=0
94 S2=0
95 S3=0
96 S4=0
97 C1=0
98 C2=0
100 FOR J=1 TOY
110 E1=(COS(I*J/A))^(N-1)*SIN(I*J/A)*SIN(I*J*K/A)
120 E2=E1*(-COS(I*J))
130 S1=S1+E1
140 S2=S2+E2
150 E3=(COS(I*J/A))^N*SIN(I*J/A)*SIN(I*J*K/A)
160 E4=E3*(-COS(I*J))
170 S3=S3+E3
180 S4=S4+E4
190 NEX J
195 G1=N+K
196 G1=G1-2
200 IF G1=0 THEN C1=1
205 IF G1=0 THEN 230
210 IF G1=1 THEN 230
220 GO TO 196
230 Q1=Z1*S1*C1
240 Q2=Z2*S3*(1-C1)
245 G2=N+A-K
246 G2=G2-2
250 IF G2=0 THEN C2=1
255 IF G2=0 THEN 280
260 IF G2=1 THEN 280
270 GO TO 246
280 P1=Z1*S2*B*C2
290 P2=Z2*S4*B*(1-C2)
300 PRI N,Q1,P1
310 PRI N+1,Q2,P2
320 NEX N
330 STOP
```

BIBLIOGRAPHY

1. GAINES, B R 'Stochastic Computer Thrives on Noise', Electronics Vol 40 No 14 July 10, 1967, pp 72-79.

2. GAINES, B R 'Stochastic Computing', AFIPS 30 SJCC 1967, pp 149-156.

3. SHOUP, J F and ADAMS, W S 'A Practical Automatic Patching System for a Time Shared Hybrid Computer', Simulation April 1972, pp 142-148.

4. HANNAVER, G 'Automatic Patching for Analogue and Hybrid Computers', Simulation May 1969, pp 219-232.

5. MILLER, A J 'Digital Stochastic Computing', PhD Thesis, University of Aberdeen, 1975.

6. BROWN, A W 'Design of a Digital Stochastic Computer', CNAA MPhil Thesis, 1975.

7. HOWARD, B Dept of Electronic and Electrical Engineering, Heriot Watt University (Private Correspondence).

8. FELLER, W 'An Introduction to Probability Theory and its Applications', Wiley International Press.

9. DERMAN, C, GLESER, L J, OLKIN, I 'A Guide to Probability Theory and Application', Holt, Rinehart and Winston Inc. Press.

10. BAILEY, N T J 'The Elements of Stochastic Processes', Wiley International Press.

11. HANDLER, H 'Monte Carlo Solution of Partial Differential Equations Using a Hybrid Computer', EES series report number 16. Engineering Experiment Station, College of Engineering, The University of Arizona, Tucson, Arizona.

12. FROBERG, C E 'Introduction to Numerical Analysis', Addison Wesley Publishing Company.