# System simulation using digital stochastic computing structures.

MCLEAN, H.R.

1975

# SYSTEM SIMULATION USING DIGITAL STOCHASTIC COMPUTING STRUCTURES.

## ABSTRACT

HECTOR R. McLEAN

AUGUST 1975

This thesis is a detailed study of the potential applications of digital stochastic computers. In particular, this work has considered the simulation of stochastic networks using digital computer software written in FORTRAN. The study of these networks was aided by hybrid computer simulations which were used to check on the stability of stochastic networks.

The introduction to the thesis compares and contrasts analogue, digital, hybrid and stochastic computers. A close comparison is made between digital stochastic computers and other forms of parallel digital computers such as the Digital Differential Analyser and the phase computer. In chapter 1 the single line, symmetric, bipolar representation was chosen as the most economical method of representing problem variables in terms of hardware. Thus, the stochastic operators presented in this chapter are based on the bipolar mapping. The mathematics used is uncomplicated and the behaviour of digital circuits containing counters has been approximated by linear and non-linear differential equations in the main text. More precise analyses of digital circuits are to be found in the appendices but it was found that these studies yielded no more information than the approximate methods and were much more awkward to manipulate.

Chapter 2 is concerned with developing software written in FORTRAN to simulate the operation of the basic stochastic operators. The random number generator used in these simulations is based on the Lehmer Congruence method and a detailed account of its properties is given with particular reference to the ELLIOT 4120 digital computer for which the software was written. The stochastic operators simulated include the negator, summer, multiplier, squarer, integrator and output interface.

Some simple circuits involving the basic operators were investigated in chapter 3. These circuits included networks for square-root extraction, the solution of a linear equation, examining the transient behaviour of a second order stochastic system and sine/cosine generation. The second order system highlighted a problem which was not taken into account in the original definition of stochastic computation. Simple mathematical models are used to explain the transient behaviour of each circuit simulated.

In/

In chapter 4 two simple circuits for solving sets of linear equations were investigated. The first is based on an error criterion and the second circuit uses the method of steepest descent. Each circuit is analysed as a continuous system in the main text, but a discrete time analysis of each network is given in the appendices. Close attention is paid to the stability and convergence of each method.

A well known linear programming algorith is adapted for use on a stochastic computer in chapter 5. This study also demonstrates the way in which threshold switching is obtained in the stochastic computer. The problem examined in this chapter is a maximisation problem but the mathematics can be easily altered to cope with a minimisation problem.

Circuits for determining the parameters of first and second order systems were investigated in chapter 6. The circuit for identifying the parameters of a first order system revealed a difficulty in scaling when the method of steepest descent is used to identify system parameters but a procedure is adopted which overcomes this problem. An alternative algorithm for identifying a first order system was successfully demonstrated. The second order system was used to demonstrate the kind of difficulty which might be encountered when using a stochastic computer for parameter identification, namely induced oscillation arising from the random variance inherent in the stochastic computation method. These studies were extensively aided by hybrid computer simulations of the steepest descent algorithm. As a result of the simulation work carried out on the first order system identification a new output interface, the non-linear adaptive digital element, is proposed and this circuit is analysed in detailed in appendix 6.C.

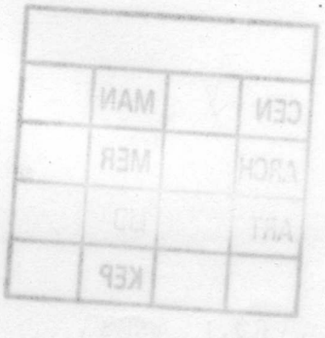
Chapter 7 is a review of the work discussed in the previous chapters and presents suggestions for further work with particular reference to Markov chains and systems which are inherently stochastic.

# SYSTEM SIMULATION USING DIGITAL STOCHASTIC COMPUTING STRUCTURES

Hector R. McLean

August 1975

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



## ACKNOWLEDGEMENTS

The author wishes to thank Dr P Mars for his help and guidance during the course of this project. Thanks are also due to Mr A Wilson of the School of Mathematics, the Punch Room staff and the technicians of the Computer Services Unit, and Miss Pam Swanson who typed this thesis.

## SUMMARY

The objective of this project has been a detailed study of the potential applications of digital stochastic computers. In particular, this work has considered the simulation of stochastic networks using digital computer software written in FORTRAN. Hybrid computer simulations were used to check on the stability of stochastic networks.

As a result of this work the circuits simulated include first and second order systems, networks for solving linear equations, matrix inversion and linear programming problems. Algorithms for identifying the parameters of first and second order systems were also investigated.

# CONTENTS

# INTRODUCTION

(A short historical survey[1] is made of analogue, digital and hybrid computers and comparisons are made between their various advantages and disadvantages.)

Simulation studies of weapons - systems such as aircraft dynamics and guided weapons, including fire control systems, started during the Second World War. These simulators were essentially analogue computers in which an electronic model was made of the system to be studied. That is, the circuit parameters behaved in a similar manner to the real system with respect to time.

Thus while weapon systems were being developed a new kind of computer, the electronic analogue computer, was coming into being as an offshoot of this research. These early machines could only simulate simple devices usually with few parameters, but the latest machines can process thousands of variables.

After World War II the idea of a stored programme digital computer was developed by Von Neuman and others. Since that time the basic digital computer architecture has showed little change from the stored programme philosophy.

In 1956 the first attempts were made to combine digital and analogue computing but these were not successful. These early hybrid computers employed distinct analogue and digital units with interfaces between them. The latest hybrid computers are a more subtle marriage of analogue and digital circuitry. Both the disadvantages and advantages of digital and analogue computation appeared in the hybrid system.

The analogue computer is fast but rather inaccurate while the digital computer is slow but precise. Careful programming was needed to overcome these disadvantages. Most analogue computers operated with digital circuits although some purely analogue machines were produced for special purposes. Despite the advances made in integrated circuit operational amplifiers the accuracy of an analogue computer still depends on passive components /

components which are not easily realised in integrated circuit form. In the analogue computer all the calculations are carried out in parallel while in a basic digital machine the same calculations would be performed a serial fashion. However, attempts are being made to produce a digital computer which can process information in a parallel fashion. Thus the analogue machine is many times faster than the conventional digital computer. With linear analogue computation 'time scaling' can be introduced which means that solution times are very much less, or greater, than those of the real system being simulated are possible. However, the cost of components limits the size of the system which can be simulated or controlled.

Advantages of the Analogue Computer

(a) They are faster than digital computers;

(b) Easy comparisons between computer models and real systems are possible;

(c) Interaction between man and machine is easy;

(d) Inputs and outputs do not need to be processed ie, they do not need to be converted to digital form, but they may need to be scaled;

(e) Easy on-line changes of variables can be made;

(f) The machine can be programmed from 'block' and circuit diagrams;

(g) No paper tape or card programmes are required.

Advantages of the Digital Computer

(a) They are precise (double precision numbers are available) and they can be very accurate if good algorithms can be implemented in the time available;

(b) Operations are either arithmetical or logical;

(c) /

(c) Good memory facilities for storing information, eg, ferrite stores, magnetic tapes, magnetic discs, etc;

(d) Variables do not have to be scaled;

(e) Permanent record of programmes are available on paper tape, cards or magnetic tape;

(f) The machine can handle mathematical, scientific and commercial problems easily.

Advantages of the Hybrid Computer

All the advantages of both types of machine combine in the one machine, but it is very much more versatile than either.

In 1965 research teams working independently in the UK and the USA on methods of pattern recognition proposed a new type of computer which could simulate large systems cheaply.

This machine is fully hybrid since it uses probability as an analogue quantity but the mechanisation of a problem is entirely digital. It has a speed/precision trade-off which cannot be matched by any previous machine. The advent of large scale integrated circuit technology will mean that large systems can be realised cheaply.

Other kinds of computer such as the Digital Differential Analyser (DDA) and the phase computer[2] have been developed along with the stochastic system. Like the stochastic computer the DDA and the phase computer use a relative frequency to represent information. In the stochastic computer information is represented by an unordered sequence of ON logic levels each of which are generated by a statistically independent process, and the computer operations are analogues of the system operations being simulated. Part of a digital stochastic sequence is illustrated in Figure 0(a) . The stochastic computer /

computer uses digital incremental UP/DOWN counters to store information about the problem variables. Information in a parallel Digital Differential Analyser is represented by patterned deterministic sequences of ON logic levels and one such sequence is illustrated in Figure 0(b). Unlike the stochastic computer, the DDA operations are not analogues of the system operations since it employs binary addition, subtraction and shifting processes. The DDA uses UP/DOWN counters and registers to store information. As with the DDA, the phase computer represents information by patterned deterministic sequences but information is stored in unidirectional counters which decreases hardware costs and simplifies the operations required to perform a calculation. The operation of these counters may be internally asynchronous although the overall operation of this computer may be synchronous. Unlike the stochastic computer and DDA the phase computer executes a programme in a sequential manner but this machine is designed to produce complex operations immediately on demand.

The representation of data by a probability means that the quantisation of analogue data necessary in other digital computers is avoided. However, a probability can only be accurately assessed if a sufficiently long clocked sequence is sampled and although there are no quantisation errors stochastic computation gives rise to another type of error, namely random variance. The effects of this error are discussed in APPENDIX 1A. The basic stochastic computer can perform the following arithmetical operations:- negation, multiplication, squaring, summation and integration. Other operations such as division and square-rooting can be extracted using combinations of the above operations. Switching functions can also be obtained as will be demonstrated in Chapter Five.

Advantages /

Advantages of the Stochastic Computer[7,8]

(a) Mappings can be produced which simply relate physical variables to the probabilities representing them (see Chapter One);

(b) Arithmetic operations are extremely cheap to produce compared with those used in a conventional hybrid computer;

(c) It is faster than a digital computer (see Figure 0(c)).

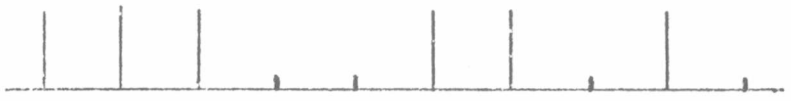
Disadvantages of the Stochastic Computer

(a) The representation of variables by sequences of randomly occurring pulses is the most inefficient method possible;

(b) Care must be exercised when setting up a stochastic computer circuit to prevent cross-correlation between the inputs of the various devices;

(c) It is not as fast as an analogue computer.

There are other advantages and disadvantages associated with the stochastic computer and these are brought out in the following sections when simulations of this machine are carried out on a conventional digital computer.



Part of a Stochastic Sequence of
Generating Probability 0.6

Figure 0(a)

Patterned Deterministic Sequence in a
Digital Differential Analyser

Figure 0(b)

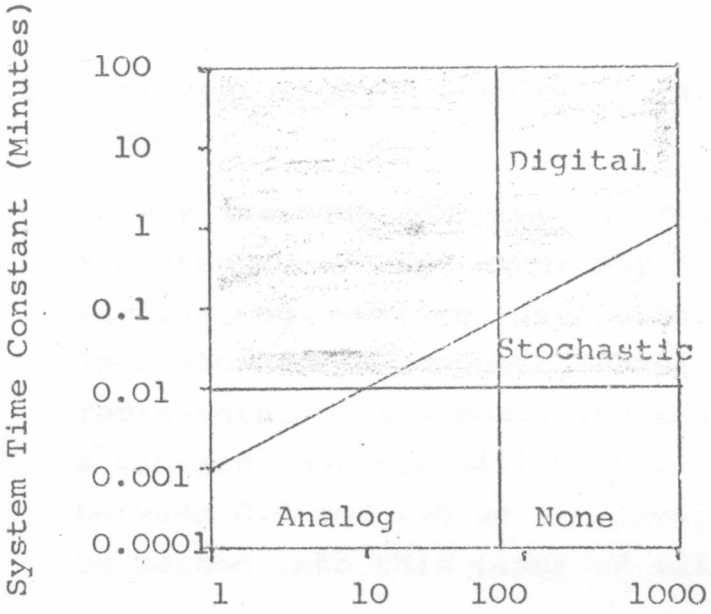



Figure 0(c)

Regions of Operation

Above the Diagonal - Digital Computers
Left of Vertical - Analogue Computers
Area Shaded - Stochastic Computers

# CHAPTER 1

(In this chapter the basic stochastic computing elements are discussed and are compared with the corresponding conventional analogue computer devices.)

## 1.1 Representation of Physical Variables in a Stochastic Computer

In any analogue computation, real variables have to be represented by some other physical variable in the machine, eg, voltage, displacement, or probability. In a stochastic computer, physical variables are represented by a probability that a logic level in a clocked sequence will be ON. Probabilities vary between 0.0 and 1.0 so all physical variables must be mapped into this range of values.

A number of possible mappings[6] for the representation of physical quantities have been proposed, and these include linear and non-linear mappings.

Non-linear mappings map variables on the domain $(-\infty,\infty)$ into the range (0,1). But these representations are difficult to deal with in terms of scaling and limitations in the resolution of a digital counter. Further, with some of these non-linear representations it is not possible (as yet) to obtain some of the basic arithmetic operations.

Linear mappings allow a more natural interpretation of a simulated problem and the three of most interest are:-

(i) unipolar
(ii) two-line bipolar
(iii) single-line bipolar

1.1 (a) /

1.1 (a) Unipolar Mapping

In this representation the physical quantities are assumed to be either always positive or always negative. The probability representation of a quantity, E, is simply:

$$p(ON) = \frac{E}{V}$$ where V is the maximum possible value of E and $0 \leq E \leq V$

1.1 (b) Two-line Bipolar Mapping

Both positive and negative quantities can be represented using this mapping by using two sequences of logic levels; one representing positive quantities and the other negative. The line in which the probability is weighted positively is called the UP line and the one weighted negatively is called the DOWN line.

Given a quantity, E, such that $-V \leq E \leq V$, ($V = E_{max}$) we have:

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

Thus V is represented by the UP line always ON and the DOWN line always OFF, and, vice versa for -V. Zero is represented by equal probabilitie of an ON logic level occurring on both lines.

1.1 (c) Single-line Bipolar Mapping

This method represents a quantity on one line without assuming anything about the sign of the variable. For some E such that $-V \leq E \leq V$ we have:

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

Thus, for E = V, p(ON) = 1.0
and E = -V, p(ON) = 0.0
and E = 0, p(ON) = 0.5

Thus /

Thus zero is represented by a logic level with an equal probability of being ON or OFF.

This mapping yields the simplest hardware synthesis of the basic arithmetical operations and so it will be used in the digital computer simulations discussed in subsequent chapters of this thesis.

In the following sections the basic stochastic computer hardware is reviewed. The operation of these devices assumes that all inputs are statistically independent and that all inputs are stationary Bernoulli sequences[6,7,8].

## 1.2 Negation[5]

Negation is performed by a logical inverter in which the output is logical complement of the input. The inverter is shown in Figure 1.2(a).

Let the probability that the input, $E_i$, will be ON be p(A) while that of the output, $E_O$, is p(B). If the two events are mutually exclusive we have:

$$p(B) = 1 - p(A) \qquad \text{---- (1.2.1)}$$

But $p(A) = \frac{1}{2} + \frac{1}{2}(\frac{E_i}{V}), \quad -V \leq E_i \leq V$

$$\Rightarrow \quad p(B) = \frac{1}{2} - \frac{1}{2}(\frac{E_i}{V}) = \frac{1}{2} + \frac{1}{2}(\frac{E_O}{V}), \quad -V \leq E_O \leq V$$

$$\Rightarrow \quad E_O = -E_i \qquad \text{---- (1.2.2)}$$

The conventional analogue computer circuit is an inverting operational amplifier, or one input summer. (See Figure 1.2(b).)

## 1.3 Multiplication[5]

Multiplication is achieved by means of an inverted EXCLUSIVE-OR gate. Using NAND logic the circuit takes on the form illustrated in Figure 1.3(b). If A and B are /

are logic inputs to this gate, and C is the logic output, then:

$$C = A.B + \bar{A}.\bar{B} \qquad \text{---- (1.3.1)}$$

This equation has the truth table detailed in Figure 1.3(c).

If the inputs to the multiplier are statistically independent events then:

$$p(C) = p(A).p(B) + [1 - p(A)][1 - p(B)] \qquad \text{---- (1.3.2)}$$

If $p(A) = \frac{1}{2} + \frac{1}{2}(\frac{E_1}{V_1})$ such that $-V_1 \leq E_1 \leq V_1$

and $p(B) = \frac{1}{2} + \frac{1}{2}(\frac{E_2}{V_2})$ such that $-V_2 \leq E_2 \leq V_2$

then $p(C) = \frac{1}{2} + \frac{1}{2}(\frac{E_1}{V_1})(\frac{E_2}{V_2}) = \frac{1}{2} + \frac{1}{2}(\frac{E_O}{V_O})$ ---- (1.3.3)

and $-V_O \leq E_O \leq V_O$

Hence $\frac{E_O}{V_O} = (\frac{E_1}{V_1})(\frac{E_2}{V_2})$ ---- (1.3.4)

This is an attenuative form of multiplication and gives similar results to the quarter-squares multiplication, and, the potentiometer methods used in conventional analogue computers.

The potentiometer has one mechanical and one electrical input while the stochastic multiplier has two electrical inputs. In a calculation one can use the stochastic multiplier in a similar manner to the way in which an analogue computer potentiometer would be employed. However, the stochastic multiplier does not have the mechanical disadvantages of wear on moving parts or the slow response times of a servo driven potentiometer. The precision potentiometer is very much more expensive than the stochastic device.

## 1.4 Squaring (5)

Squaring cannot be obtained by short circuiting the two inputs of the multiplier as this will always result in the output of the device being ON. See equation (1.3.1). This is the result of cross-correlation between the inputs of the device. If a sequence is delayed by at least one event then a statistically independent replica of this sequence is obtained and the two sequences can be multiplied together. Suppose $A_n$ is the logic value of the input to the squarer at the nth clock pulse then from equation (1.3.1):

$$C_n = A_n.A_{n-1} + \bar{A}_n.\bar{A}_{n-1} \qquad \text{---- (1.4.1)}$$

where $A_{n-1}$ is the logic value of the input at the previous clock pulse.

Then we can write:

$$P(A_n) = P(A_{n-1}) \qquad \text{---- (1.4.2)}$$

Let $P(A_n) = \frac{1}{2} + \frac{1}{2}(\frac{E}{V})$ , $-V \leqslant E \leqslant V$

Then $p(C_n) = p(A_n).p(A_{n-1}) + [1 - p(A_n)][1 - p(A_{n-1})]$

---- (1.4.3)

Hence $p(C_n) = \frac{1}{2} + \frac{1}{2}(\frac{E}{V})^2 = \frac{1}{2} + \frac{1}{2}(\frac{E_O}{V})$ ---- (1.4.4)

from which $E_O = \frac{E^2}{V}$ ---- (1.4.5)

A serial shift register will act as a delay and so gives stochastic isolation thus preventing cross-correlation. See Figure 1.4(a).

The analogue computer equivalent is a voltage multiplier with its inputs short-circuited together. See Figure 1.4(b).

1.5 /

## 1.5 Summation[5]

The process of summation is not straightforward. For example, consider a two input summer in which one input represents the maximum positive quantity and so is always ON, while the other, representing the maximum negative quantity is always OFF. Their sum must be zero which is represented by a random pulse train with a generating probability of 0.5. Clearly the two inputs are deterministic and if simply OR'd they cannot produce a random sequence.

Random behaviour is built into the summer by triggering a D-type flip-flop with digital noise of probability 0.5 so that either input line has an equal chance of being connected to the output. This is achieved as shown in Figure 1.5(a). The circuit has the truth table detailed in Figure 1.5(d). If the logic inputs are A and B with logic output, C, then:

$$C = Z.A + \bar{Z}.B \qquad \text{---- (1.5.1)}$$

and $p(Z) = 0.5$

Hence, $p(C) = \frac{1}{2}.p(A) + \frac{1}{2}p(B)$ ---- (1.5.2)

$$= \tfrac{1}{2}.(\tfrac{1}{2} + \tfrac{1}{2}(\frac{E_1}{V_1})) + \tfrac{1}{2}.(\tfrac{1}{2} + \tfrac{1}{2}(\frac{E_2}{V_2}))$$

$$= \tfrac{1}{2} + \tfrac{1}{2}(\frac{E_1}{2V_1} + \frac{E_2}{2V_2}) = \tfrac{1}{2} + \tfrac{1}{2}(\frac{E_O}{V})$$

if $V_1 = V_2 = V, \Rightarrow E_O = \frac{1}{2}(E_1 + E_2)$ ---- (1.5.3)

The variables $E_1$ and $E_2$ are effectively rescaled and this effect must be watched with care when many quantities are being summed.

## 1.6 Summation of Many Variables

The following discussions refer to two input summers.

(a) Let the number of inputs to be summed be S and let /

let $S = 2^n$ where n is a positive integer such that $n \geqslant 0$. Then we need (S-1) two input summers and the output is normalised to $2^n V$, eg, S = 4, see Figure 1.6(a), and $E_O = \frac{1}{4}(E_1 + E_2 + E_3 + E_4)$ ---- (1.6.1)

The complete summation requires three summers. Generalising, for a circuit which sums S quantities,

$$E_O = 2^{-n} \sum_{j=1}^{S} E_j$$

$$\Rightarrow \quad E_O = S^{-1} \sum_{j=1}^{S} E_j \qquad \text{---- (1.6.2)}$$

There is one great disadvantage in this method of summation since as S increases the output, $E_O$, converges to zero, if $\sum_{j=1}^{S} E_j \leqslant V$.

(b) If $S = \sum_{i=1}^{N} 2^{n_i}$ where the $n_i$ are integers such that $n_i \geqslant 0$ and let $n_1 > n_2 > n_3 > \ldots\ldots > n_N$, then we need (S-1) summers and the output is normalised to $2^{n_1+1} V$, eg, S = 6, and from Figure 1.6(b), it can be seen that:

$$E_O = \frac{1}{8}(E_1 + E_2 + E_3 + E_4 + E_5 + E_0) \qquad \text{---- (1.6.3)}$$

Five summers are required to implement this summation. Again, as S increases the output, $E_O$, converges to zero since the normalisation depends on $n_1$ which is defined as the largest of all the $n_i$.

In the above example a compensating multiplier is required to preserve the meaning of the summation, ie, it is used to alter the scaling (see Chapter Five and Six). It can be shown that (N-1) compensating multipliers /

multipliers are required. In the above example N = 2 so that one compensating multiplier has to be included in the one circuit to equalise the scaling between the two partial sums.

## 1.7 Division[5]

Division has to be used carefully since the output probability must be in the range

$$0.0 \leqslant p_O \leqslant 1.0 \ .$$

Division is extracted from a hill climbing algorithm which calculates the gradient of a criterion function and this result is used to drive the system continuously until the desired output is obtained.

Let $E_O/V$ be an approximation to $E_1/E_2$. We define the error:

$$\varepsilon = \frac{E_2 E_O}{V} - E_1 \qquad \text{---- (1.7.1)}$$

We use the square of the error as our criterion function:

$$\varepsilon^2 = \left(\frac{E_2 E_O}{V}\right)^2 - 2\,\frac{E_1 E_2 E_O}{V} + E_1^{\,2} \qquad \text{---- (1.7.2)}$$

The hill climbing technique requires that

$$E_O = -K\,\frac{d\,(\varepsilon^2)}{d\,E_O} \qquad K > 0$$

$$= -2K\,E_2$$

$$= -2K\,\frac{E_2^{\,2}\,E_O}{V^2} + \frac{2K\,E_2\,E_1}{V} \qquad \text{---- (1.7.3)}$$

In the steady state,

$$\frac{E_O}{V} \rightarrow \frac{E_1}{E_2} \quad \text{and} \quad \frac{\dot{E}_O}{V} = 0$$

A /

A circuit diagram of the divider is presented in Figure 1.7(a).

However, when $E_O/V = E_1/E_2$ the criterion function, $\varepsilon^2$, is zero and hence $\varepsilon$ is zero so that the optimum value of this index of performance is a null point. For the class of optimisation in which the optimum operating point gives rise to a null value for the criterion function we can adopt the following strategy:

$$\text{Let } \frac{E_O}{V} = -K\varepsilon \qquad \text{---- (1.7.4)}$$

$$\text{where } \varepsilon = \frac{E_2 E_O}{V} - E_1, \quad K > 0$$

$$\text{Hence } \frac{E_O}{V} = -K\frac{E_2 E_O}{V} + KE_1 \qquad \text{---- (1.7.5)}$$

Comparing this with equation (1.7.3) we have a much simpler optimisation strategy in terms of hardware, and, in the steady state

$$\frac{E_O}{V} \rightarrow \frac{E_1}{E^2} \quad \text{and} \quad \frac{\dot{E}_O}{V} = 0$$

A simulation of this second circuit was performed and the results are presented in Chapter Three. A circuit diagram is presented in Figure 3.2(a) and a discrete time analysis of this device is given in APPENDIX 1B.

## 1.8 Integration (5,6,7)

For this operation counters are used as memories, and stochastic automata theory provides a basis for analysing the behaviour of random pulses in sequential circuits.

Suppose the counter contains N+1 states $S_0, S_1, \ldots, S_N$. Let $S_i$ be a numerical value assigned to each state, ie $S_i$ is the output of the counter in its ith state. If $S_i$ lies in the range (0,1) then:

$S_i$ /

$$S_i = \frac{i}{N} \qquad \text{---- (1.8.1)}$$

At some clock pulse the counter will be in the state $S_i$ with associated output $S_i$. If the counter is driven by a stochastic input sequence, only $\pi_i$, the probability that the counter is in its ith state, is known. The output is a random variable with expected value:

$$\bar{S} = \sum_{i=0}^{N} \pi_i S_i \qquad \text{---- (1.8.2)}$$

The counter used is an UP/DOWN counter. Let $\omega$ be the probability that the UP line is ON and the DOWN line is OFF, and let e be the probability that the UP line is OFF and the DOWN line is ON. The expected change in the output of the counter at a clock pulse is:

$$\delta S = \frac{(\omega - e)}{N} \qquad \text{---- (1.8.3)}$$

If T is the clock interval, the expected change in the output of the counter over some time domain is:

$$\bar{S}(nT) - \bar{S}(0) = \sum_{m=0}^{n-1} \delta S(mT)$$

$$= \sum_{m=0}^{n-1} \frac{\omega(mT) - e(mT)}{N} \qquad \text{---- (1.8.4)}$$

This is a zero order numerical (discrete) summation for $\omega(t) - e(t)$. If N is large enough we can approximate the summation to an integral. Then,

$$\bar{S}(t) \simeq \bar{S}(0) + \frac{1}{NT} \int_0^t (\omega(t) - e(t)d\tau \qquad \text{---- (1.8.5)}$$

Thus the counter behaves like an integrator with respect to time having an effective gain of 1/NT. An integrator is realised by connecting the line representing the quantity to be integrated to the UP line and its inverted form to the DOWN line. See Figure 1.8(a). If the generating probability of the input sequence is $p_1$, (representing $E_1$), then/

then

$$\omega = p_1$$

$$\Rightarrow \quad e = 1 - p_1 = 1 - \omega \qquad \text{---- (1.8.6)}$$

$$\Rightarrow \quad \omega - e = 2p_1 - 1 = \frac{E_1}{V} \qquad \text{---- (1.8.7)}$$

$$\Rightarrow \quad E_O(t) = E_O(O) + \frac{2}{NT} \int_O^t E_1(\tau)\, d\tau \qquad \text{---- (1.8.8)}$$

Thus the gain of the integrator has been doubled. Used in this way the counter either increases by one or decreases by one each clock pulse. If we get a situation where no change occurs one can build a two input summing integrator.

Suppose $p_1$ and $p_2$ are the two input probabilities then:

$$\omega = p_1 \cdot p_2 \qquad \text{---- (1.8.9)}$$

$$\Rightarrow \quad e = (1-p_1)(1-p_2) \qquad \text{---- (1.8.10)}$$

$$\Rightarrow \quad \omega - e = p_1 + p_2 - 1 = \left(\frac{E_1+E_2}{2V}\right) \qquad \text{---- (1.8.11)}$$

$$\Rightarrow \quad E_O(t) = E_O(O) + \frac{1}{NT} \int_O^t [E_1(\tau) + E_2(\tau)]\, d\tau \qquad \text{---- (1.8.12)}$$

The circuit illustrated in Figure 1.8(b) achieves this type of integration. The output of the integrator is a Bernoulli sequence which is obtained by comparing the current stored value with a random number. This forms an input interface. A diagram of this process is given in Figure 1.8(d).

The conventional analogue computer has multiple input summing integrators although there is a sign inversion in the output. Multiple input stochastic summing integrators can be achieved by using a summing array before the integrator.

## 1.9 Input Interface[5,6,7]

The input interface is always obtained by comparing a deterministic input with a random number. An input signal, A, is mapped into the range (0,1) and compared with a noise signal, N, which randomly varies in the range (0,1). If A is greater than N then the output of the comparator is a logic '1' otherwise it is a logic '0'. The output signal is a Bernoulli sequence in which the probability of ON logic levels is directly proportional to the magnitude of the signal A.

Random numbers can be generated from natural noise sources or pseudo-random binary number generators. Natural noise sources include radio-active materials and noise diodes. The use of noise diodes as random number generators has the following disadvantages:

(a) experiments can never be exactly repeated;

(b) to build up an n bit binary number from n sequences of probability 0.5 large numbers of analogue to stochastic rate converters are required;

(c) the operation of a noise diode is affected by variations of temperature so a temperature controller would have to be included in the circuit to ensure the correct output generating probability;

(d) the output of the noise diode illustrated in Figure 1.9(a) gives rise to sampling rate problems which can ultimately limit the speed of a stochastic computer. Radio-active digital noise sources can be realised if silicon detectors are coated with long half-life radio-active materials and may be possible to produce many of these sources on a single chip. However, no suitable radio-active sources are yet available commercially.

The /

The pseudo random binary number generator[5,6,7,8,10,11] is a more convenient method which overcomes the above disadvantages. The device is basically a serial shift register with feedback. See Figure 1.9(b). The feedback element is an EXCLUSIVE-OR gate.

If the feedback connections are chosen correctly, the shift register will pass through all possible states with the exception of all zero. Should the shift register contain all zeros then it can never change its state. Thus for an n bit shift register the maximum number of possible unordered states generated using the correct feedback is $2^n-1$.

The correct feedback sequences can be determined from the characteristic polynomial[11] of the shift register. If the input to the register is A then the output of the first stage flip-flop will take on the value of A after a delay. This is represented by DA. Hence after n cycles of the clock the last bit of the shift register will take on the value of A and this is represented by $D^nA$.

For a three bit shift register the maximal length sequence is generated if the first and third bits are EXCLUSIVE-OR'd and fed back to the input.
Hence,

$$A = DA + D^3A \qquad \text{---- (1.9.1)}$$

EXCLUSIVE-OR both sides of this equation with A

$$A + A = A + DA + D^3A \qquad \text{---- (1.9.2)}$$

But $A + A = 0$

so that

$$(D^3 + D + 1)A = 0 \qquad \text{---- (1.9.3)}$$

which is the characteristic equation of this three bit PRBS generator.

For /

For a maximal length sequence to be generated the characteristic polynomial must be irreducible and it must not be a factor of $(D^m + 1)$ for any $m < 2^n-1$. Thus if any polynomial is not primitive the resulting sequence will not have a maximal length with an all zero loop but will have either more than two loops or two loops which do not contain all possible states of the counter. If only part of the shift register was used as an output it is possible to obtain an all zero state while satisfying the above conditions.

Simple EXCLUSIVE-OR gating can produce a delayed replica of the original sequence so that a number of uncorrelated generators can be derived from a master generator.

In a maximal length sequence of an n bit shift register there are $2^{n-1}$ ones and $2^{n-1}- 1$ zeros so that the generating probability is:

$$p = \frac{1}{2 - (\frac{1}{2^{n-1}})} \qquad \text{---- (1.9.4)}$$

Hence

$$\lim_{n\to\infty} p = \tfrac{1}{2} \qquad \text{---- (1.9.5)}$$

ie, as n increases the generating probability approaches 0.5.

Example

A thirty-three bit shift register with feedback from the thirteenth and the thirty-third stage has a maximal length sequence of $2^{33}-1$ numbers, ie, 8, 589, 934, 591 numbers. If the clock frequency is 1 MHz the cycle will repeat after approximately 2.4 hours.

The input interface of a stochastic computer is illustrated in Figure 1.9(a).

## 1.10 Output Interface[6,7,9]

Usually, the output of a stochastic computer will be a non-stationary Bernoulli sequence which can be considered as a deterministic signal with superimposed noise. The output interface must be insensitive to noise and provide a measure of the mean value of the input sequence's generating probability. This is accomplished by using an ADDIE (ADaptive DIgital Element). This is an averaging circuit which can be made out of an integrator with unity negative feedback, and it averages the input which is weighted by a decaying exponential term so that past values have less and less effect on the integral, ie,

$$E_O(t) = \frac{1}{NT} \int_0^t E_{in}\, e^{-Gt} dt \qquad \text{---- (1.10.1)}$$

where $E_{in}$ = constant and $e^{-Gt}$ is an exponential weight.

The ADDIE is dependent for its operation on the input stochastic sequence and the probability of a feedback sequence obtained from the contents of an UP/DOWN counter. Let the probability that the ADDIE is in state i at time t be $\pi_i(t)$. Then the probability of changing from state i to state j at time t is $\pi_{ij}(t)$.

The probability of being in state j at time (t+1) is then:

$$\pi_j(t+1) = \sum_{i=1}^{N} \pi_i(t).\pi_{ij}(t) \quad i = 1,2,..,j,...,N \qquad \text{---- (1.10.2)}$$

This is a non-stationary Markov process where $\pi_{ij}(t)$ is a probability matrix in which the rows sum to unity. However, UP/DOWN counters cannot jump states so that if the counter is in the initial state, i, it can stay in i or move to (i-1) or to (i+1), ie,

$\pi_{i,}$ /

$$\pi_{i,\ell-1}(t) + \pi_{i,\ell}(t) + \pi_{i,\ell+1}(t) = 1.0 \qquad \text{---- (1.10.3)}$$

Suppose the integrator is a two input device, then let p be the probability of the input sequence being ON at any clock interval, and the probability of it being OFF be q. Then the distribution function of the ADDIE states is binomial, ie,

$$p_n = p^n q^{N-n} \binom{N}{n} \qquad \text{---- (1.10.4)}$$

The states of the ADDIE will fluctuate about a mean value of:

$$M = Np \qquad \text{---- (1.10.5)}$$

and this mean value is an unbiased estimate of the generating probability of the input sequence.

The transient response[9] of the ADDIE to a step input in probabilistic terms is given by:

$$n(t) = Np\left[1 - \exp\left(\frac{-t}{\tau}\right)\right] \qquad \text{---- (1.10.6)}$$

where the time constant, $\tau$, is given by:

$$\tau = -f_c\left[\ln\left(1 - \frac{1}{N}\right)\right]^{-1} \qquad \text{---- (1.10.7)}$$

where $f_c$ = clock frequency
p = input probability
N = the number of ADDIE states.

The circuit diagram for this type of ADDIE, the noise ADDIE, is displayed in Figure 1.10(a).
We can define a bandwidth for a stochastic computing system in terms of the 3dB point of the ADDIE.

$\omega_{3dB}$ /

$$\omega_{3dB} = -f_c \ln(1-\frac{1}{N}) \qquad \text{---- (1.10.8)}$$

An error measurement for the ADDIE can be based on the normalised standard deviation of the state distribution function, ie,

$$\varepsilon = (\frac{pq}{N})^{\frac{1}{2}} \qquad \text{---- (1.10.9)}$$

and the maximum error occurs when $p = q = 0.5$, ie,

$$\varepsilon_{max} = 0.5(N)^{-\frac{1}{2}} \qquad \text{---- (1.10.10)}$$

Bandwidth is related to the error in the following manner:

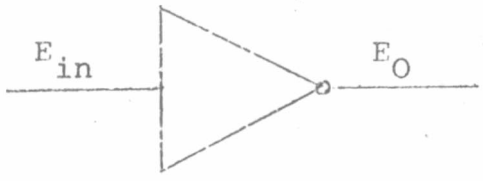
$$\varepsilon_{max} = 0.5\,[1 - \exp(\frac{-\omega_{3dB}}{f_c})] \qquad \text{---- (1.10.11)}$$

Thus, for a fixed bandwidth accuracy can only be improved by increasing the clock frequency, ie, more samples are taken in a given time so that the final result is sensibly independent of any one sample.

It has been shown that a deterministic feedback signal[9] can be employed instead of a random sequence. Figure 1.10(c) illustrates the way in which this is achieved. Even if one signal is deterministic and the other is random there is statistical independence between them so that computations are still valued in these cases. A deterministic signal can be generated using a binary rate multiplier (B.R.M). The output frequency, $f_b$, of the B.R.M. is determined by the current state of the UP/DOWN counter and the clock frequency, $f_c$. The generating probability of the feedback sequence is $f_b/f_c$. Because of the deterministic nature of the feedback signal the output distribution function is different to that of the noise ADDIE. With the B.R.M. ADDIE there is less random variance[9] /

variance[9] compared with the noise ADDIE, so improving the accuracy without any reduction of the interface bandwidth.

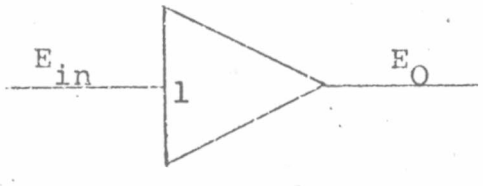
In the next chapter simulation models of the various stochastic computer devices are presented.




$$E_O = - E_{in}$$

Single Line Symmetric
Representation Inversion

Figure 1.2(a)




$$E_O = - E_{in}$$

Analogue Computer Inverter

Figure 1.2(b)

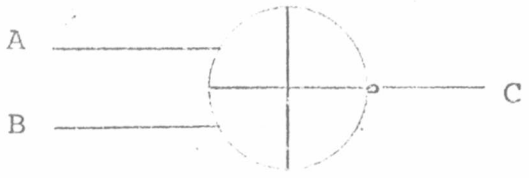
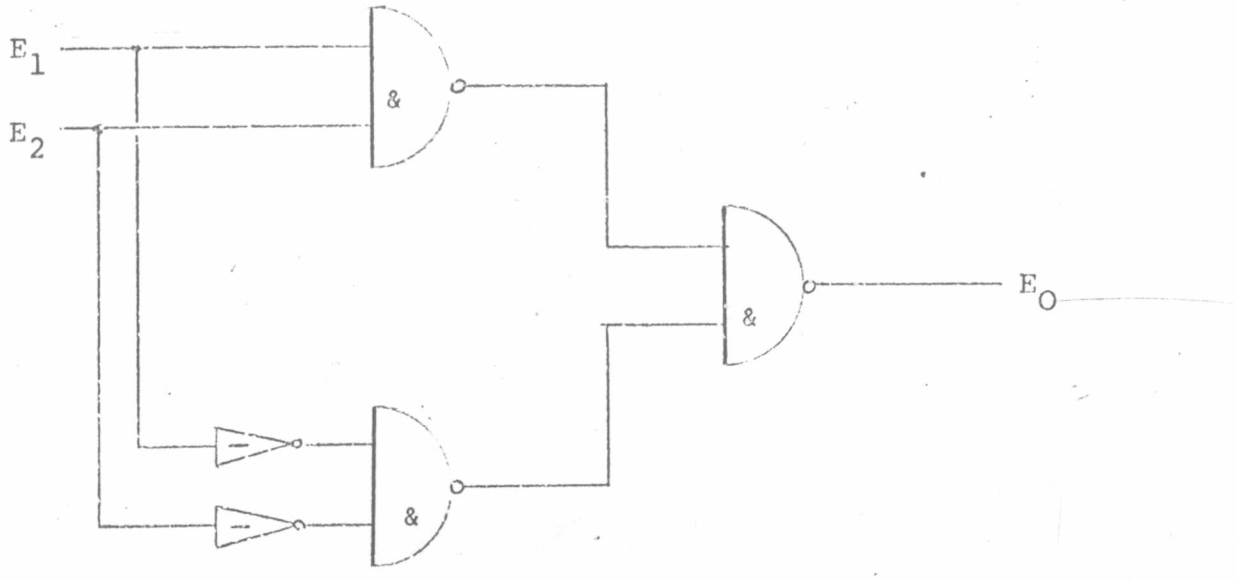



Figure 1.3(a) Inverted Exclusive-OR Gate




$$\frac{E_O}{V_O} = \frac{E_1}{V_1}\frac{E_2}{V_2}$$

Figure 1.3(b) Single Line Symmetric Representation Multiplier

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

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

Figure 1.3(c) Truth Table

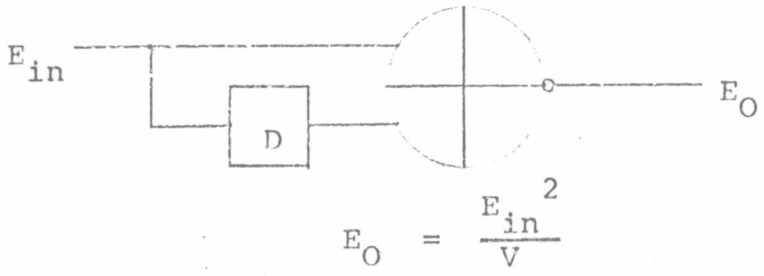



Figure 1.4(a) Stochastic Squarer

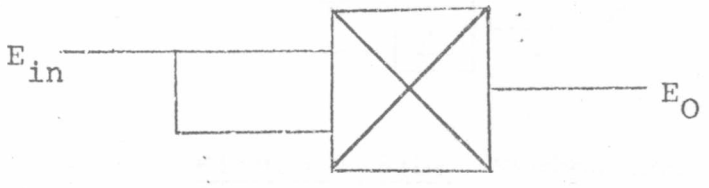



Figure 1.4(b) Analogue Computer Squarer

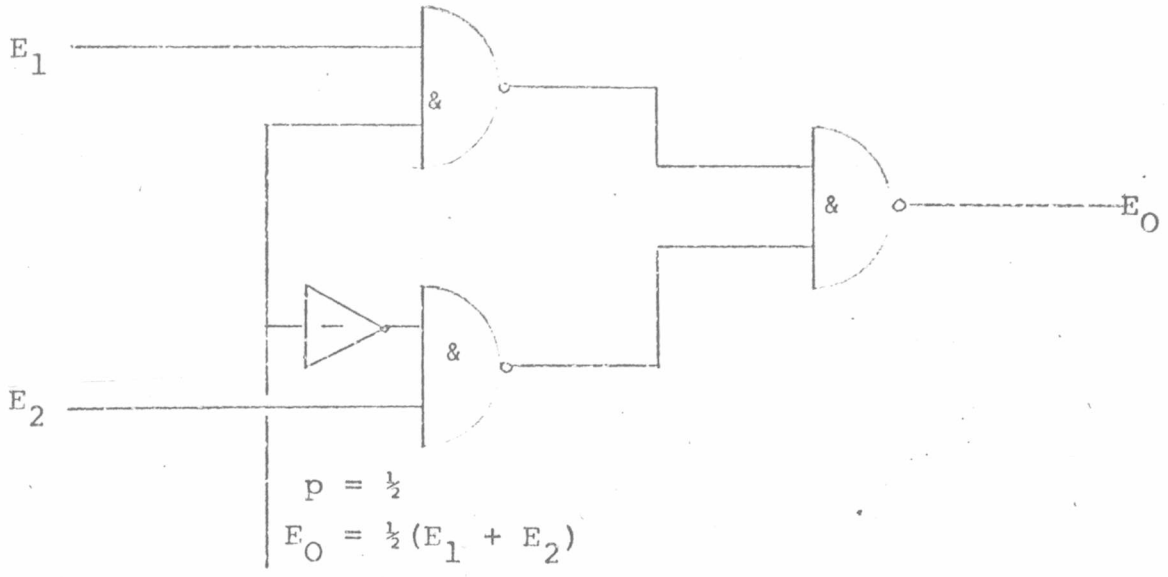



Figure 1.5(a) Stochastic Summer

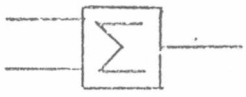
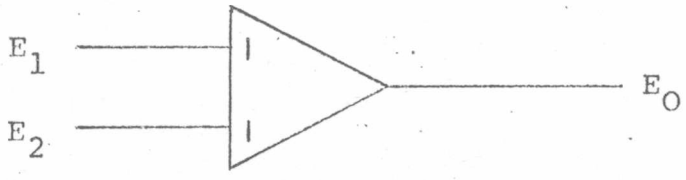



Figure 1.5(b) Symbol for Stochastic Summer




$E_0 = -(E_1 + E_2)$

Figure 1.5(c) Analogue Computer Summer

| A | B | Z | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Figure 1.5(d) Truth Table for The Stochastic Summer

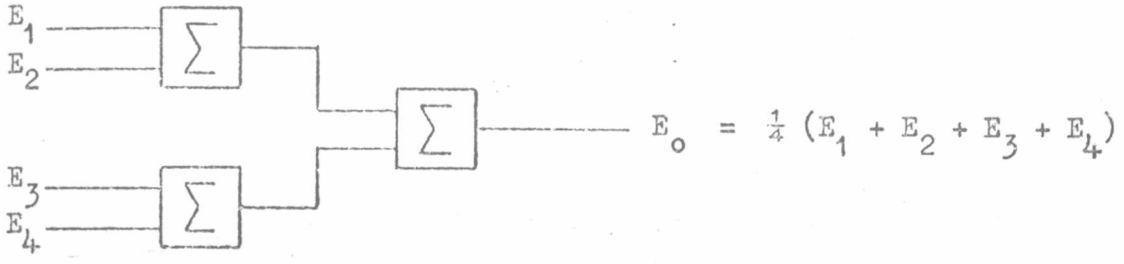



FIG 1.6 (a)

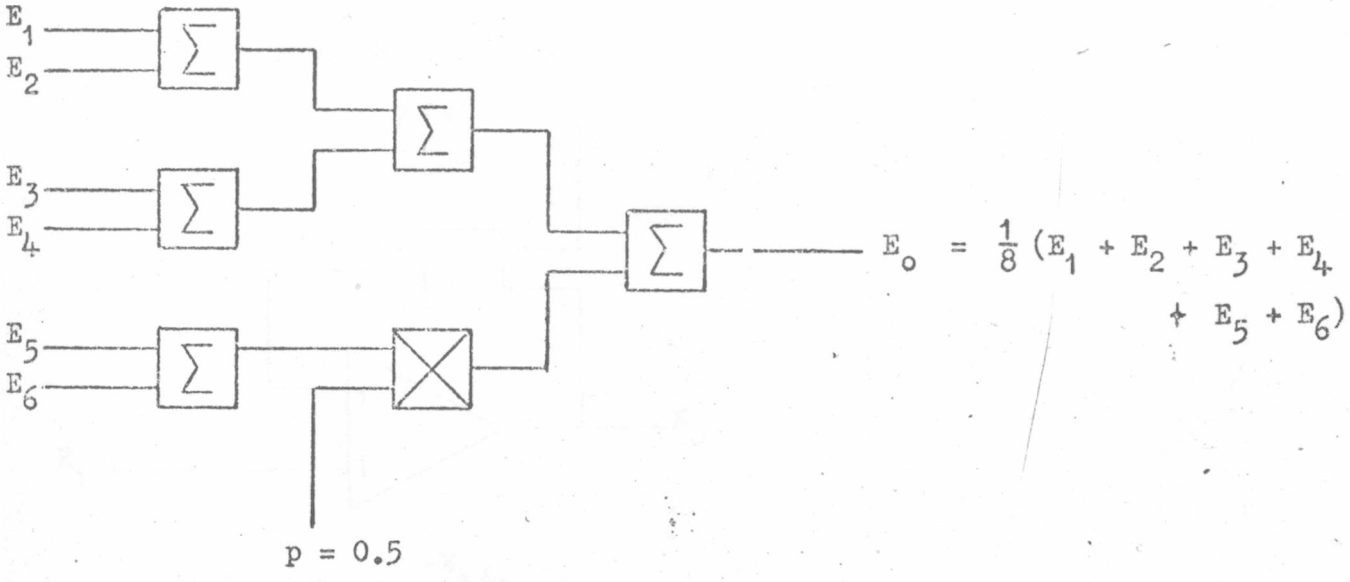
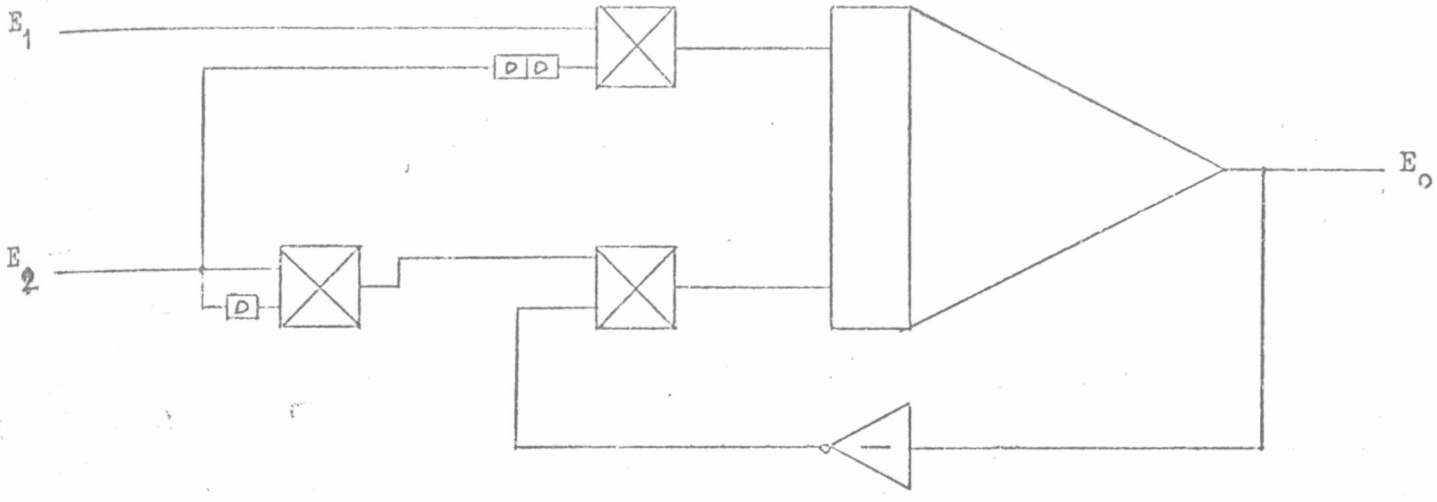



FIG 1.6 (b)

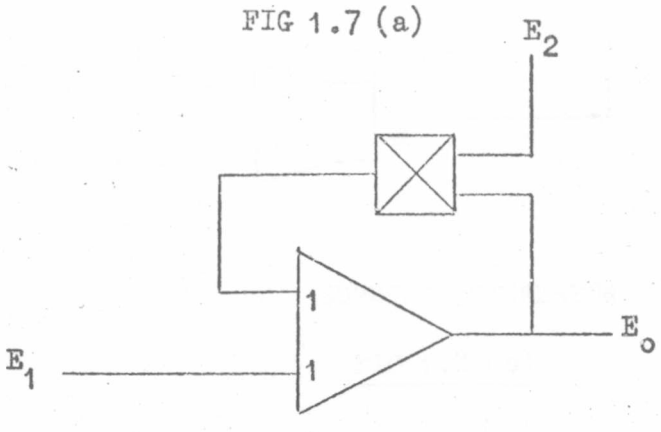
SUMMATION OF MANY VARIABLES




$$E_o = E_1/E_2$$

STOCHASTIC DIVIDER

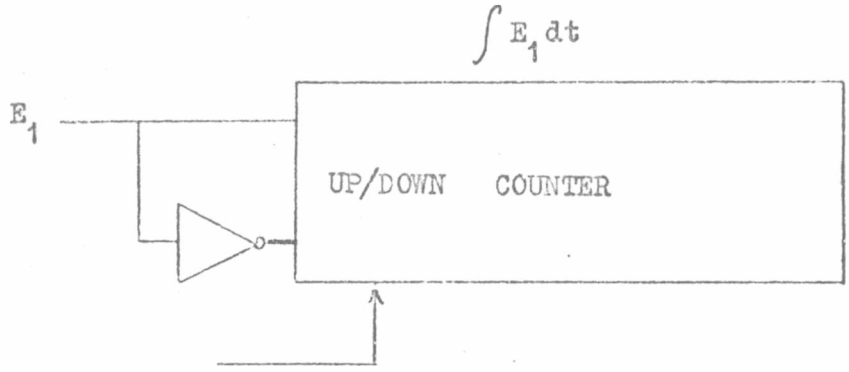
FIG 1.7 (a)




$$E_o = -E_1/E_2$$

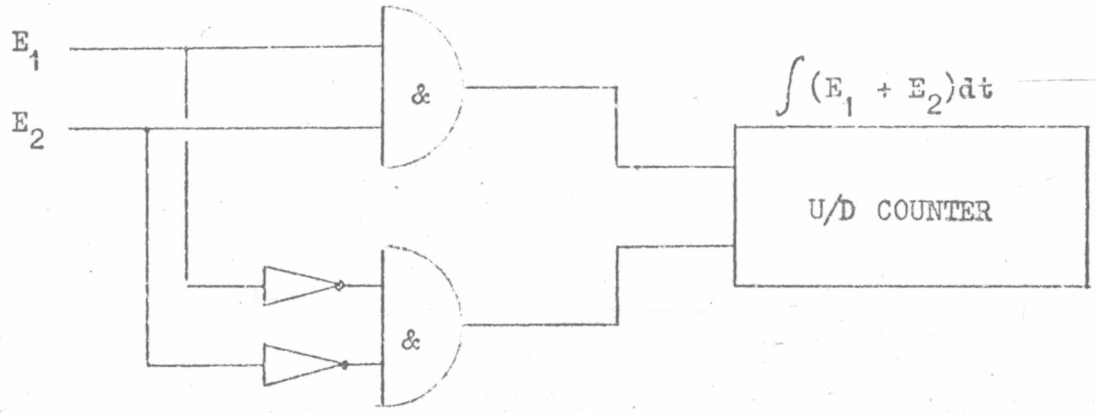
ANALOGUE COMPUTER DIVIDER

FIG 1.7 (b)



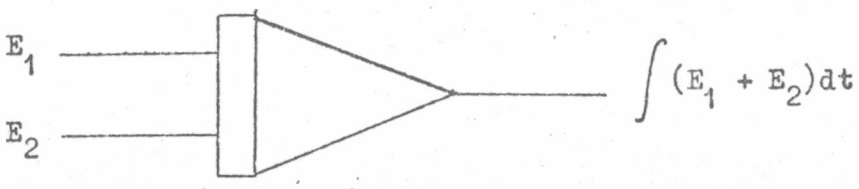

SLSR INTEGRATOR

FIG 1.8 (a)




SUMMING INTEGRATOR

FIG 1.8 (b)




SYMBOL FOR THE SUMMING INTEGRATOR

FIG 1.8 (c)

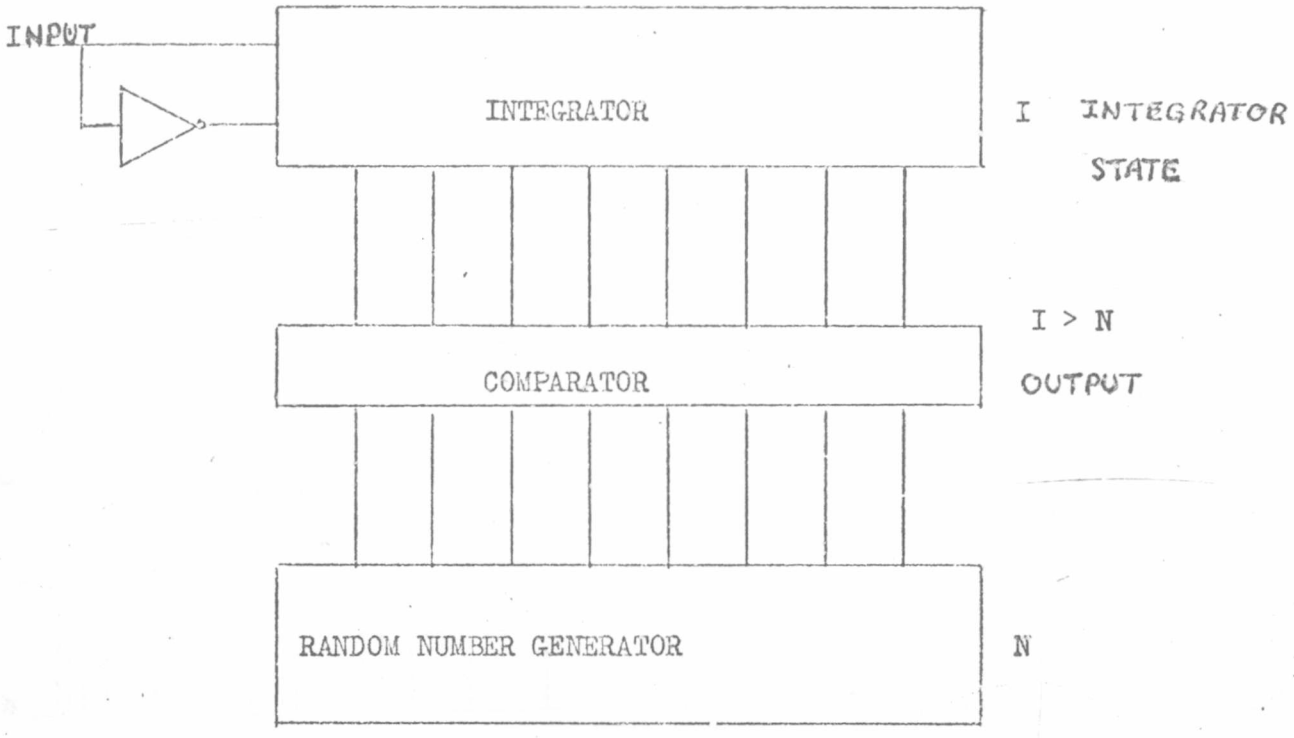
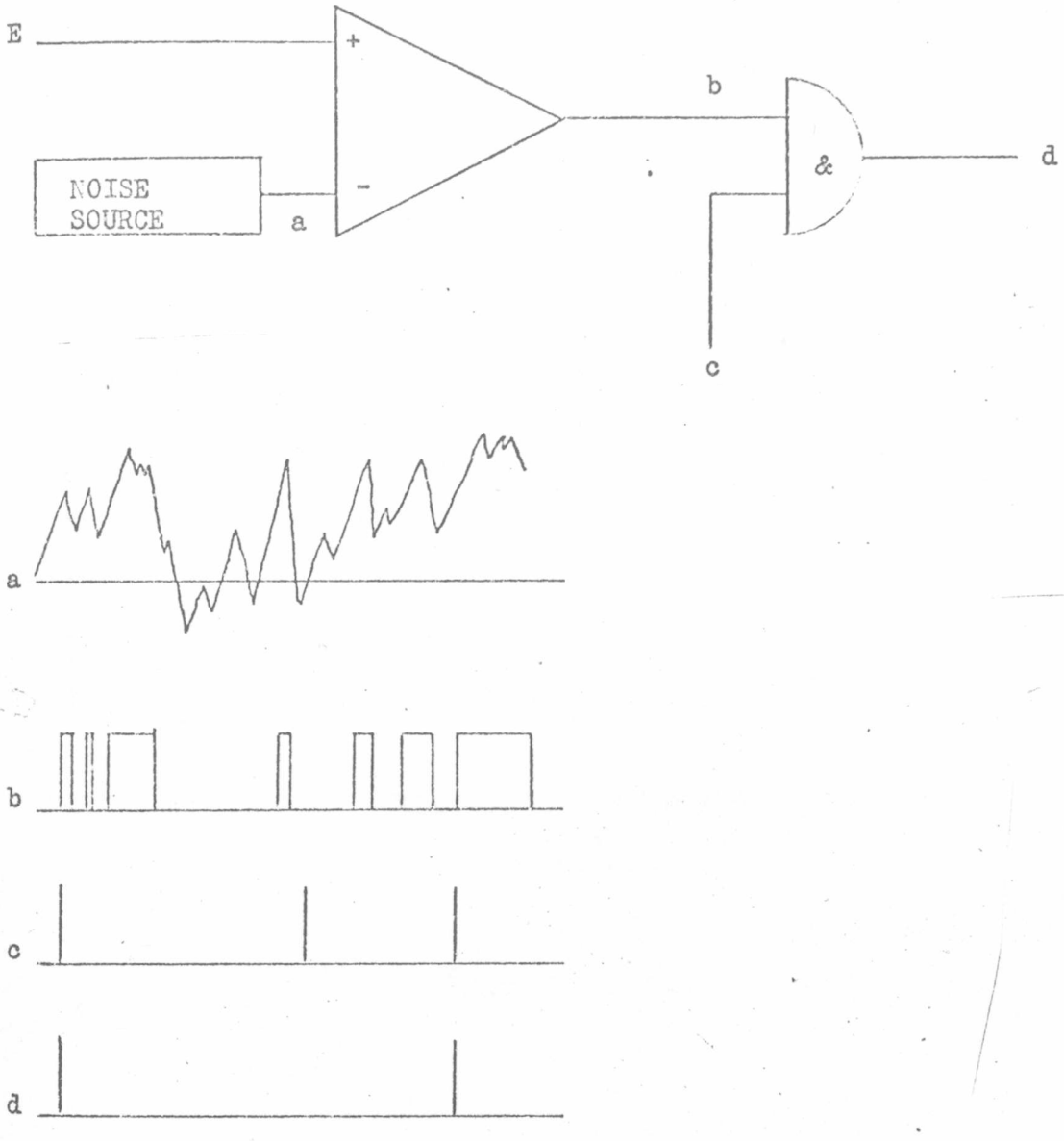
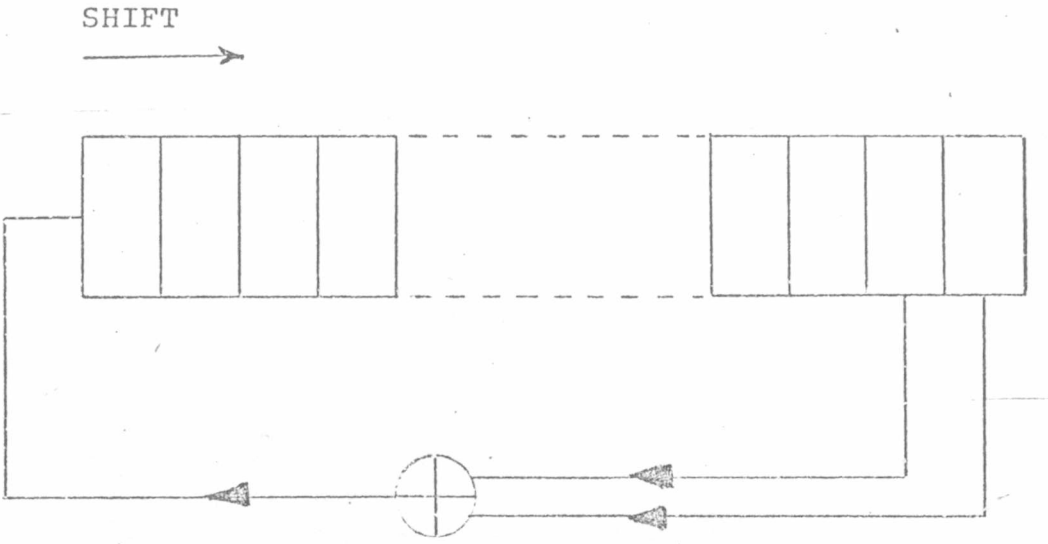

INPUT
INTEGRATOR
I INTEGRATOR STATE
I > N
COMPARATOR
OUTPUT
RANDOM NUMBER GENERATOR
N

FIG 1.8 (d)




ANALOGUE-STOCHASTIC RATE CONVERSION

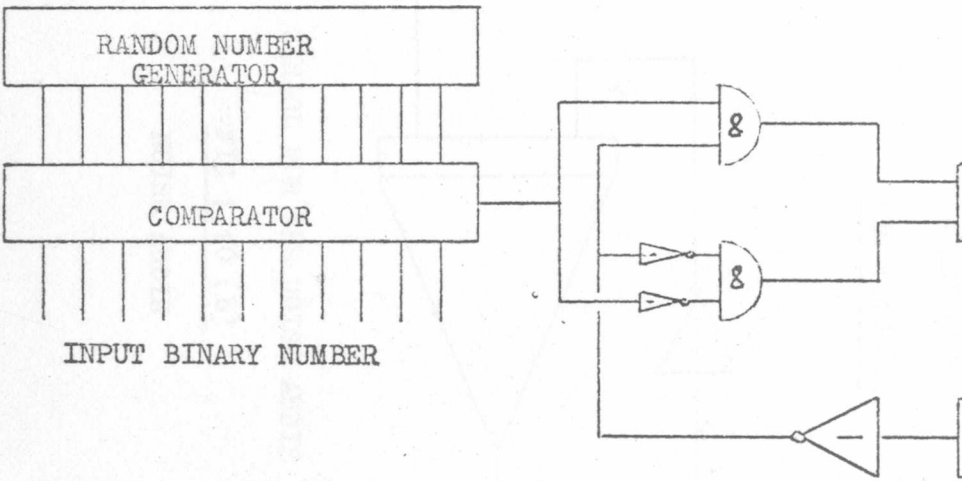
FIG 1.9 (a)




'Exclusive OR' Gate

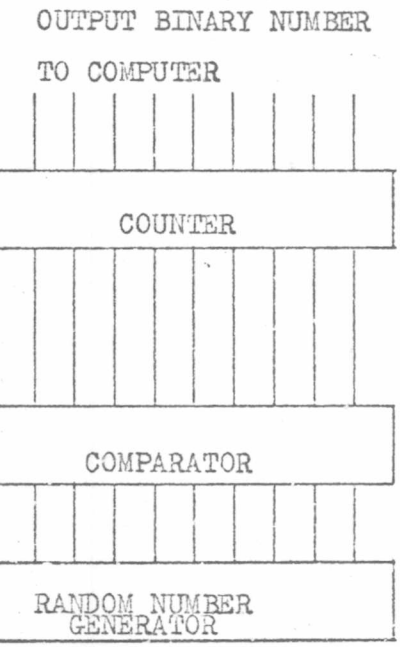
P.R.B.S. Generator

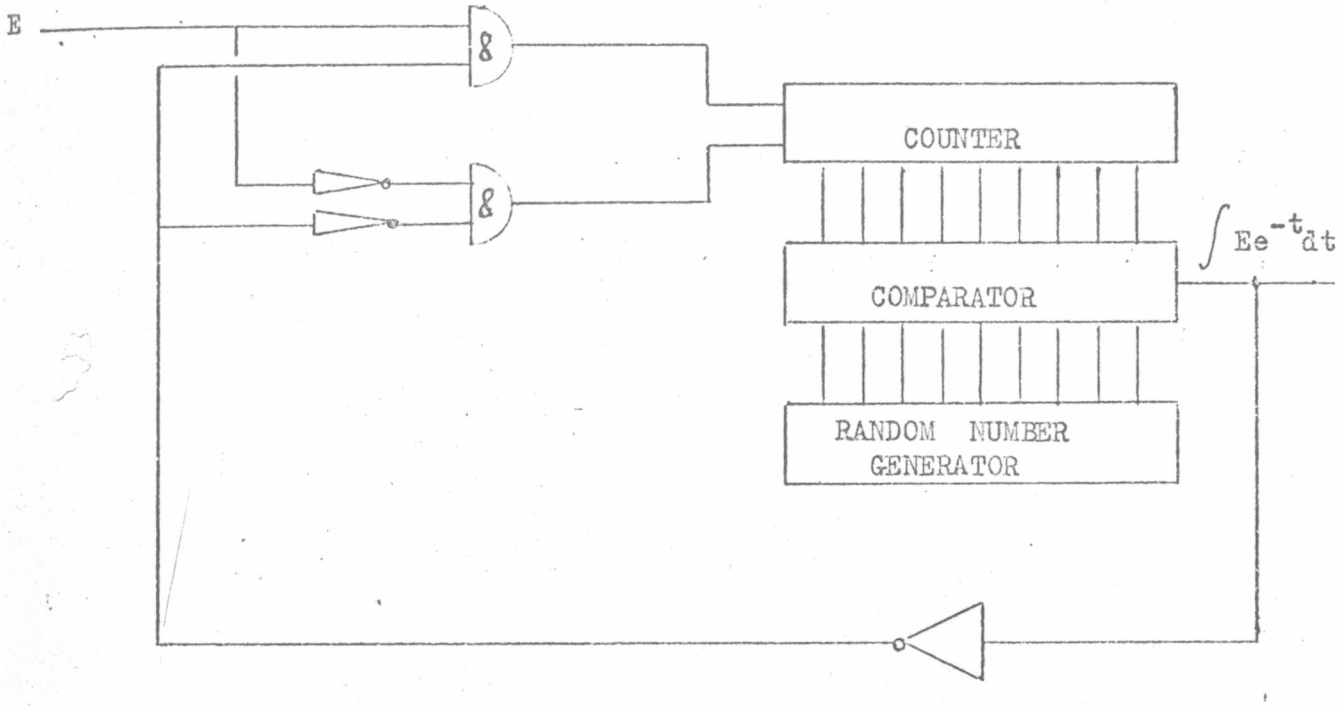
Figure 1.9(b)




INPUT INTERFACE

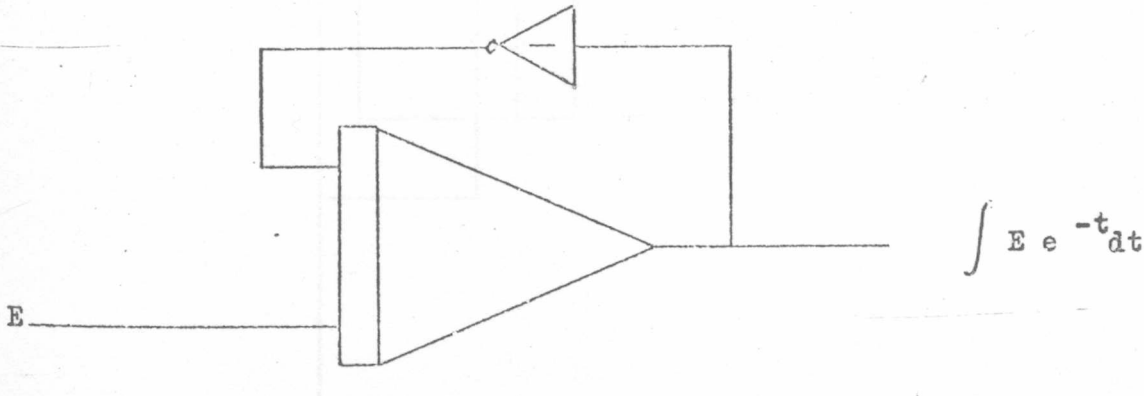
FIG 1.9 (c)


OUTPUT BINARY NUMBER
TO COMPUTER
COUNTER
COMPARATOR
RANDOM NUMBER
GENERATOR




NOISE ADDIE

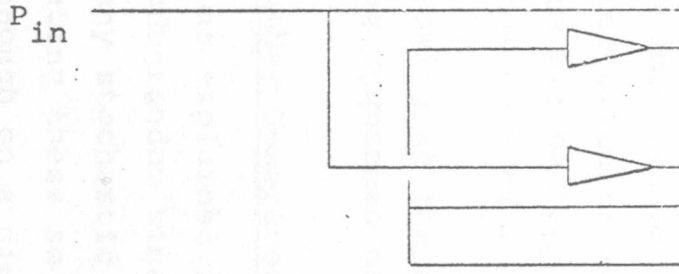
FIG 1.10 (a)

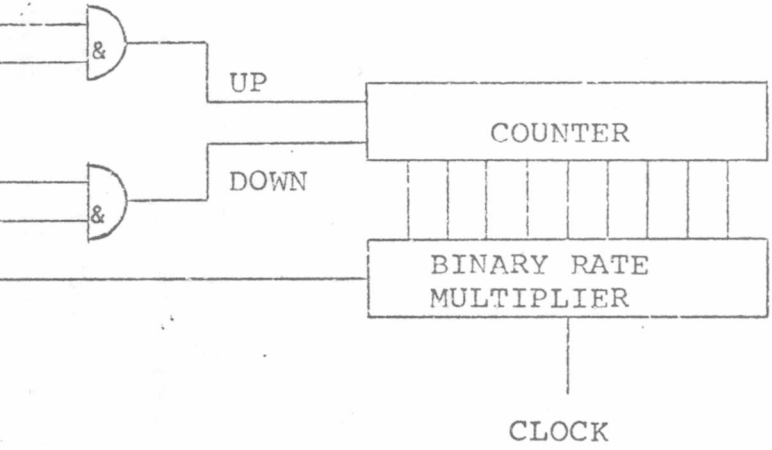



SYMBOL FOR THE NOISE ADDIE

FIG 1.10 (b)

NOISE ADDIE


$P_{in}$




B.R.M. ADDIE

Figure 1.10(c)

# CHAPTER 2

(In this chapter the basic stochastic computing devices are modelled in the FORTRAN IV high level programming language.)

## 2.0 Introduction

Since all the elements introduced in the last chapter can be defined by simple logic and arithmetic operations it is easy to model them, using a high level programming language, on a digital computer. The high level language chosen was FORTRAN IV[15, 16, 17, 18] because of the ease with which arithmetic and logic[19, 20] statements can be used.

Wherever possible, each computing unit will be modelled by a subroutine. This will save tedious duplication of statements in the main programme.

To enable complex systems to be simulated each type of subroutine is part of an array so that, for example, any one summer can be picked out from the others. For simpler circuits the same subroutine can be used to perform all particular kinds of computation regardless of how many of these particular devices there are.

Before looking at the computing devices, algorithms for generating sequences of random numbers are considered.

## 2.1 Random Number Generators[21, 22]

For reasons explained in the last section it is intended that pseudo-random binary number generators will be used in any stochastic computation. The actual method of generating these sequences cannot be simulated quickly enough on a digital computer so simpler algorithms are investigated here.

For /

For successful computations certain criteria are required of the sequences of random numbers produced by these methods:

1. Each element of the sequence is bounded and all possible values within the bounds will appear equally often in the sequence.
2. The auto-correlation of consecutive elements in the sequence is zero.
3. Introducing deterministic rules for forming sequences of bounded numbers will give rise to a cyclic sequence so we must work with sufficiently long cycles for our purposes.

Von Neumann suggested the 'Mid-Squares' technique in which a 'p' digit number, $x_0$, is squared and from the resulting 2p digits the middle digits are taken as $x_1$. The number $x_1$ is squared and the process is repeated. If the $x_i$ have a radix, r, then there are $r^{2p}$ possible values of $x_i$ so that the sequence must repeat some previous value. It is thus cyclic. The cycle length is actually considerably less than the theoretical maximum of $r^{2p}$.

The cycle length depends on the starting value $x_0$. Some of these values can lead to a zero term and when this happens the cycle length is one.

There is a tendency for successive numbers in these sequences to decrease in value since they do not satisfy the requirement that any value in the permitted range is equally likely to occur.

The /

The 'Mid Product Method'[21] is an extension of the 'Mid-Square' technique in which there are two starting values, viz, $x_1$ and $x_2$. The product, u, of $x_1$ and $x_2$ is formed. The middle digits of u are used as $x_3$ and the process is repeated using $x_2$ and $x_3$ to give $x_4$. This method has less bias and longer cycles than the 'Mid-Square' technique. Indeed all truncation methods lead to bias. To get a uniform distribution we require a transformation $f(x_n)$ of $x_n$ so that $x_{n+1}$ is given by the following analysis which considers front truncation only.

Let $u = f(x_n)$ ---------- (2.1.1)

and

$x_{n+1} = r^p (u \bmod r^{-p})$ ---------- (2.1.2)

Let $h = r^{-p}$ and $m = r^p = \frac{1}{h} \Rightarrow mh = 1$ ------ (2.1.3)

Let $x_{n+1} = \omega$ ---------- (2.1.4)

Suppose u has the distribution function $p(u)$ and the cumulative distribution function $P(u)$, then,

$$P(\omega) = h[\, p(\omega) + p(\omega+h) + p(\omega+2h) + \ldots\ldots + p(\omega + h(m-1))] \quad \text{--- (2.1.5)}$$

$$\Rightarrow \int_0^{(m-1)h} p(\omega+x)\,dx = P(\omega+1-h) - P(\omega) \quad \text{--- (2.1.6)}$$

Approximately,

$P(\omega+1-h) - P(\omega) = A$ ---------- (2.1.7)

for a suitable range of $\omega$ and some constant A, then over this range,

$P(\omega) = \alpha + \beta\omega$ ---------- (2.1.8)

for suitable $\alpha$, $\beta$

$\Rightarrow p(u) = \beta$ ---------- (2.1.9)

Thus /

Thus the distribution of u is uniform and the only permissible transformation is

$$f(x) = kx + \ell \qquad \text{---------- (2.1.10)}$$

where k and $\ell$ are constants.

Considering the case for $\ell = 0$, then we have the 'Lehmer Congruence Method'[21] in which the theory of numbers is used to devise cyclic sequences of maximum possible length. The theory of congruences supplies these sequences. The problem lies in calculating a suitable value of k to give a maximum length sequence. The sequence length is independent of the starting value $x_0$. However, the starting value must be non-zero and odd.

This method was used in a FORTRAN IV subroutine to generate random number sequences. Truncation was obtained by producing numbers far greater than can be represented by the twenty-four bit word of an 'ELLIOT 4120' digital computer. Any negative numbers were made positive so that all numbers were in the range $(0,2^{23})$. The maximum cycle length is $2^{21}$ and this was obtained by using a value of 4099 for k.

The subroutine below generates a random number and then provides a comparison with a deterministic number.

```
      CALL RANNUM(DE,LP,IA,ED)
           :
           :
      SUBROUTINE RANNUM(EE,LP,IA,DD)
      LOGICAL LP
C SUBROUTINE FOR VARIABLE COUNTER LENGTHS.
      IY=IX*4099
      IF(IY)1,2,2
    1 IY=IY+8388607+1
    2 RN=IY
      RN=RN/8388607.0
      IX=IY
      E=(EE+DD)/(2*DD)
      LP=.FALSE.
      IF(E-RN)4,4,3
    3 LP=.TRUE.
    4 RETURN
      END
```

## 2.2 Negation

Using the single line symmetric bipolar mapping, negation is simply a logical inversion and it is represented by a logic assignment statement.

```
B = .NOT.A
```

## 2.3 Multiplication

See Figure 2.3.

The following logic expression has to be evaluated:

$$C = A.B + \bar{A}.\bar{B} \qquad \text{---------- (2.3.1)}$$

Writing this in FORTRAN we have:

```
      SUBROUTINE MULT(SINP,K)
      LOGICAL SINP(2),SLLM(10)
      COMMON MU/SLLM
      SLLM(K)=(SINP(1).AND.SINP(2).OR.((.NOT.SINP(1)).
     C AND.(.NOT.SINP(2))))
      RETURN
      END
```

## 2.4 Squaring

This device is essentially the same as the multiplier except that one of the inputs is delayed by one clock pulse. A one bit delay is easily simulated in a programme by storing the current value of the input and using it again in the next iteration with its next logical value. Thus at the start of any simulation one of the inputs to the multiplier has to be initialised. See Figure 2.4.

This is simulated in the following way:

```
L=1 /
```

```
      L=1
      LLD(2)=.TRUE.
      DO 1 M=1,N

      LLD(1)=LL
      INP(1)=LLD(1)
      INP(2)=LLD(2)
      CALL MULT(INP,L)
      LLD(2)=LLD(1)

    1 CONTINUE
```

## 2.5 Summation

See Figure 2.5.

The following logic expression has to be evaluated:

$$C = Z.A + \bar{Z}.B \qquad \text{---------- (2.5.1)}$$

where Z is a logic variable representing a random sequence of generating probability 0.5. Thus a random number will have to be used to supply this random sequence.

This device is simulated by the following subroutine.

```
      SUBROUTINE SUM(SINP,K,IA)
      LOGICAL SINP(2),SLLS(10),LP
      COMMON /S/SLLS
      DE=0.0
      CALL RANNUM(DE,LP,IA)
      SLLS(K)=((LP.AND.SINP(1)).OR.((.NOT.LP).
     C AND.SINP(2)))
      RETURN
      END
```

## 2.6 Division

No model is given for division as it is a steepest descent algorithm incorporating the basic logic elements described in the last chapter.

2.7 /

## 2.7 Summing Integrator

The two input summing integrator counts up if both inputs are ON and down if both inputs are OFF. If the inputs are different the counter does not change. See Figure 2.7.

This is simulated by:

```
      SUBROUTINE INT(SINP,K,IA)
      LOGICAL SINP(2),SLLI(10),LP
      INTEGER SIS(10)
      COMMON /I/SIS,SLLI
      IF((SINP(1).AND..NOT.SINP(2)).OR.((.NOT.
     C SINP(1).AND.SINP(2)))GOTO 21
      SIS(K)=SIS(K)-1
      IF(SINP(1).AND.SINP(2))SIS(K)=SIS(K)+2
   21 DE=SIS(K)
      CALL RANNUM(DE,LP,IA)
      SLLI(K)=LP
      RETURN
      END
```

Unless it is necessary to store the logic value of the output of the integrator for other computations, SLLI(J) need not appear in the programme. Also, the random number comparison may be done in the main programme as part of the normal interfacing routine if this is more convenient. Thus we have:

```
      CALL INT(INP,J)
      DE=IS(J)
      CALL RANNUM(DE,LP,IA)
      LLI(J)=LP
```

## 2.8 ADDIE

Although this is a one input-one output device it can be put in the same format as the other units, ie, specifying an input and a device identification number, eg, K.

The /

The logic is expressed in terms of the current output of the ADDIE and the input. Thus we have to specify a starting value for the output of the ADDIE, LLA(J), before the actual stochastic computation commences. See Figure 2.8.

The following subroutine may be used:

```
      SUBROUTINE OUT(SINP,K,IA)
      LOGICAL SINP(2),SLLA(10),LP
      INTEGER SAS(6)
      COMMON /O/SAS,SLLA
      IF((SINP(2).AND.(.NOT.SLLA(K))).OR.((.NOT.SINP
     C (2)).AND.SLLA(K)))GOTO 31
      GOTO 32
   31 SAS(K)=SAS(K)-1
      IF(SINP92).AND.(.NOT.SLLA(K)))SAS(K)=SAS(K)+2
   32 DE=SAS(K)
      CALL RANNUM(DE,LP,IA)
      SLLA(K)=LP
      RETURN
      END
```

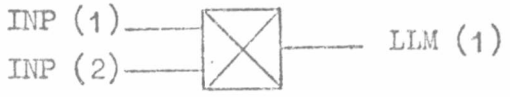
Again, the interface may be done in the subroutine or the main programme.

## 2.9 B.R.M. ADDIE

Because of the complex behaviour of the B.R.M. ADDIE no subroutine is offered as a simulation would take too long to perform.

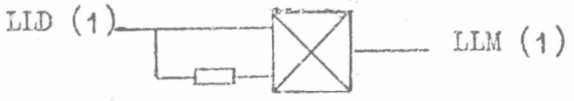
Having built up simple logic and arithmetical models of the computing units available we can interconnect these in a main programme to perform simulations of simple and complicated synchronous sequential networks.

Some simple circuits are investigated in the next chapter.




STOCHASTIC MULTIPLIER

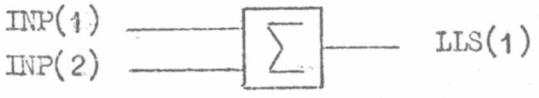
FIG 2.3




LLD (2)

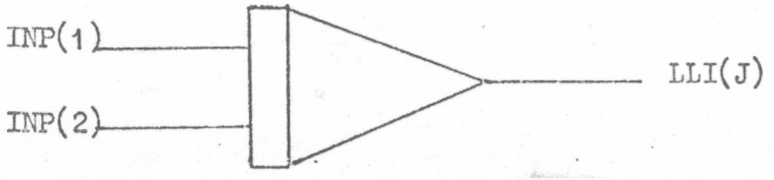
STOCHASTIC SQUARER

FIG 2.4




STOCHASTIC SUMMER

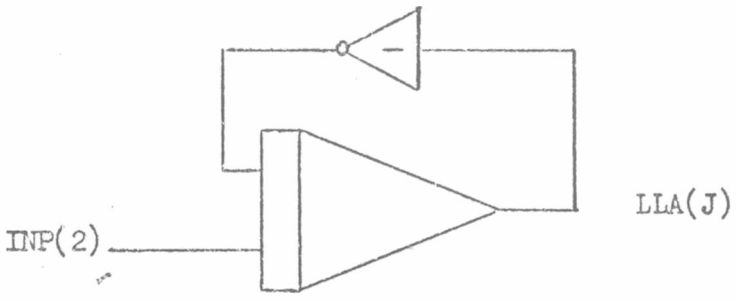
FIG 2.5




COUNTER STATE = IS(J)

SUMMING INTEGRATOR

FIG 2.7




COUNTER STATE = AS(J)

NOISE ADDIE

FIG 2.8

# CHAPTER 3

(In this chapter the simulation models are tested and some simple function generators are investigated.)

## 3. Introduction

In the last chapter simple arithmetic and logic models were developed to represent the basic computing units of the stochastic computer. These models were translated into FORTRAN IV subroutines which can be used to simulate large interconnected circuits.

The following analyses are based on conventional analogue computing algorithms. Stochastic automata theory has not been used, and instead emphasis has been placed on investigating such analogue effects as damping, transients and bandwidth. However, the effects of random variance have not been ignored and their effects on system stability will be discussed.

## 3.1 Square-Root Extraction

A circuit[6] has been proposed which yields the square-root of a positive number. The circuit is illustrated in Figure 3.1(a). This network solves the following non-linear differential equation:

$$E_O(t) = \frac{f_c}{M}\int_0^t (E_{in}(t) - \frac{E_O(t^2)}{V})dt + C \qquad \text{---- (3.1.1)}$$

The steady-state solution is:

$$\frac{E_O(t^2)}{V} = E_{in} \qquad \text{---- (3.1.2)}$$

where $E_{in}$ is a step input.

Thus, $E_O = \sqrt{(VE_{in})}$

$$\Rightarrow \quad \frac{E_O}{V} = \sqrt{(\frac{E_{in}}{V})} \qquad \text{---- (3.1.3)}$$

Hence the output probability is given by:

$$P_O = \tfrac{1}{2} + \tfrac{1}{2}\left(\frac{E_O}{V}\right) = \tfrac{1}{2} + \tfrac{1}{2}\sqrt{\frac{E_{in}}{V}} \qquad \text{---- (3.1.4)}$$

The input probability must be:

$$P_{in} = \tfrac{1}{2} + \tfrac{1}{2}\left(\frac{E_{in}}{V}\right) > \tfrac{1}{2} \qquad \text{---- (3.1.5)}$$

for this particular circuit configuration.

The complete solution of the following non-linear differential equation may be derived:

$$\frac{\dot{x}}{G} + x^2 = b^2 \qquad \text{---- (3.1.6)}$$

where $x = E_O/V$, $b^2 = E_{in}/V$ and $G = f_c/M$.

Let $[\cos[\theta(t)] = \frac{x(t)}{b}$ , $b \geqslant 0$ ---- (3.1.7)

and

$$[\sin^2[\theta(t)] = \frac{\dot{x}(t)}{Gb^2} \qquad \text{---- (3.1.8)}$$

But $\dot{x}(t) = -b\dot{\theta}(t)\ \sin[\theta(t)]$ ---- (3.1.9)

Replacing this expression for $x(t)$ in equation (3.1.8) we have:

$$\dot{\theta}(t) = -Gb\ \sin[\theta(t)] \qquad \text{---- (3.1.10)}$$

$$\Rightarrow \int \frac{d\theta}{\sin\theta} = -Gbt + C \qquad \text{---- (3.1.11)}$$

$$\Rightarrow \ln\{\tan[\frac{\theta(t)}{2}]\} = -Gbt + C \qquad \text{---- (3.1.12)}$$

at $t = 0$, $x(0) = 0$ and hence $\theta(0) = \pi/2$, thus $C = 0$

$$\Rightarrow \tan[\frac{\theta(t)}{2}] = e^{-Gbt} \qquad \text{---- (3.1.13)}$$

$$\Rightarrow \theta(t) = 2\ \arctan[e^{-Gbt}] \qquad \text{---- (3.1.14)}$$

From equation (3.1.7) we have

$$x(t) = b\cos\{2\ \arctan[e^{-bGt}]\} \qquad \text{---- (3.1.15)}$$

$$\Rightarrow \quad \frac{E_O(t)}{V} = \sqrt{\left(\frac{E_{in}}{V}\right)} \cos\{2 \arctan[e^{-bGt}]\} \qquad \text{---- (3.1.16)}$$

Since the circuit is very simple it is easy to simulate. The programme is listed in APPENDIX 3A. The simulation is equivalent to $12 \times 10^4$ clock periods and took forty minutes to run on an 'ELLIOT 4120' digital computer. For the example chosen, V is equivalent to 2048 states, using twelve bit counters, and $E_{in}$ is equal to 1024 states. Hence,

$$E_O = 32\sqrt{2048} \triangleq 1450 \text{ states}$$

The output was plotted against time. See Figure 3.1(b). This graph shows that the output takes a certain time to reach a steady state. The transient response agrees with the mathematical model and the output approaches a mean level of 1450 states as predicted by equation (3.1.11). The time taken for the output to reach 63.2% of its steady state value is $44.6 \times 10^3$ clock pulses. If the output is multiplied by a factor of $1/\sqrt{V}$, then the actual input number of states can be square-rooted.

## 3.2 Solution of a Linear Equation

One of the interesting features of the square-rooting circuit is that it has the effect of raising the input probability since,

$$\sqrt{\frac{E_{in}}{V}} > \frac{E_{in}}{V} .$$

This is achieved because the amount of negative feedback is being decreased. Since the element changing the amount of feedback is essentially a stochastic multiplier we can use one of its inputs to control the probability being fed back to the input of the integrator. See Figure 3.2(a). The probability fed back is,

$p_f/$

$$p_f = \tfrac{1}{2} - \tfrac{1}{2}\left(\frac{E_1 E_O}{V^2}\right) \qquad \text{---- (3.2.1)}$$

If $E_1$ is equal to V, then we have a noise ADDIE.

Essentially, the circuit is solving the linear equation

$$E_O\left(\frac{E_1}{V}\right) = E_{in} \qquad \text{---- (3.2.2)}$$

The circuit solves this equation in the following way for a step input.

$$E_O(t) = \frac{f_c}{M}\int (E_{in} - \left(\frac{E_1}{V}\right)E_O(t))dt \qquad \text{---- (3.2.3)}$$

Let $\frac{f_c}{M} = G$ and $\frac{E_1}{V} = \frac{1}{n}$

Equation (3.2.3) has the transient solution:

$$E_{OT}(t) = ce^{-\frac{G}{n}t} \qquad \text{---- (3.2.4)}$$

$$\frac{E_{Op}(t)}{n} = E_{in} \qquad \text{---- (3.2.5)}$$

If $E_O(0) = 0$, the complete solution is:

$$E_O(t) = nE_{in}\left[1 - e^{-\frac{G}{n}t}\right] \qquad \text{---- (3.2.6)}$$

and $\lim_{t\to\infty} E_O(t) = nE_{in}$, $n > 1.0$

Thus if $E_1 > 0$ always, we have a circuit which gives stable stochastic amplification by a factor of n. Thus on a bipolar mapping this circuit can both raise and decrease probabilities. Since in the steady state

$$E_O = \frac{V}{E_1} E_{in} \qquad \text{---- (3.2.7)}$$

$$\Rightarrow \quad \frac{E_O}{V} = \frac{E_{in}}{E_1} \qquad \text{---- (3.2.8)}$$

Hence, /

Hence,

$$p_O = \tfrac{1}{2} + \tfrac{1}{2}\left(\frac{E_O}{V}\right) = \tfrac{1}{2} + \tfrac{1}{2}\left(\frac{E_{in}}{E_1}\right) \qquad \text{---- (3.2.9)}$$

Thus, $E_{in}$ is effectively rescaled, but we must have:

$$\frac{E_{in}}{E_1} \leqslant 1.0 \quad \text{so that} \quad 0 \leqslant p_O \leqslant 1.0,$$

otherwise limiting will occur. This circuit is identical to the dividing circuit proposed in Chapter One.

However, there is a price to pay if the circuit is to be used as an amplifier. From equation (3.2.6), we can see that if we wish to multiply the input $E_{in}$ by a factor n, the system gain is attenuated by a factor of 1/n. This means that as n increases the circuit takes longer to attain its steady state operating point and hence the bandwidth is reduced by a factor of 1/n.

A circuit like this may be a possible answer to the cummulative attenuation resulting from stochastic summing arrays, since we can get amplification by any factor limited only by the accuracy with which $E_1$ can be set. This circuit then, behaves like an ADDIE, with variable feedback. By rearranging the network as shown in Figure 3.2(b) one can obtain an inverted output. All other stochastic amplifying circuits proposed so far have no bandwidth problems but can only amplify probabilities by a factor of two.

A programme simulating the operation of this circuit is listed in APPENDIX 3B. Two problems were run using this programme. One giving multiplication by 2.0 and the other by 3.06, ie,

(a) multiplication of +500 states by 2.0;
(b) multiplication of -500 states by 3.00;

Example (a) - Multiplication by 2.0.

The output clearly approaches a mean level of 1000 states as required. The simulation was carried out for an equivalent of $60 \times 10^3$ clock periods. For a clock frequency of /

of $10^6$Hz the real time simulated is 0.06 sec. The output probability is 0.744 with associated standard deviation of 27 states. See Figure 3.2(c).

Example (b) - Multiplication by 3.00.

In this case the integrator had an initial condition of -500 states and the output clearly approaches a mean level of -1500 states. This simulation was carried out for $120 \times 10^3$ clock periods which is equivalent to 0.12 sec at a clock frequency of $10^6$Hz. The output probability is 0.175 with a standard deviation of 25 states. See Figure 3.2(d).

Both curves demonstrate the effect of the multiplication factor on the transient response of the circuit, and they also indicate a rapid decrease in bandwidth as n increases. For example:

| Device | Bandwidth (radians/sec) |
|---|---|
| Noise Addie | 200 |
| Amplifier with a gain of 2.0 | 100 |
| Amplifier with a gain of 3.0 | 66 |

## 3.3 Response of a Second Order Stochastic System (Second Order ADDIE)

In this section we use our simulation language to investigate some of the properties of a second order system with variable damping. It has been suggested that higher order filters[6] be used as output interfaces and the particular circuit investigated here is a second order ADDIE. The circuit configuration is displayed in Figure 3.3(a).

In the same way as before a simplified analysis of the behaviour of this circuit is presented without considering the effects of random variance. From Figure 3.3(a) we have: /

have:

$$E_O(t) = \frac{f_c}{N}\int (E_1(t)-E_O(t))dt + c_1 \qquad \text{---- (3.3.1)}$$

and

$$E_1(t) = \frac{f_c}{M}\int (E_{in}(t)-E_O(t))dt + c_2 \qquad \text{---- (3.3.2)}$$

$$\Rightarrow \quad \frac{MN}{f_c^2}\ddot{E}_O(t) + \frac{M}{f_c}\dot{E}_O(t) + E_O(t) = E_{in}(t) \qquad \text{---- (3.3.3)}$$

Comparing equation (3.3.3) with the following well known representation of a second order system we have:

$$\frac{\ddot{E}_O(t)}{\omega_n^2} + \frac{2\zeta}{\omega_n}\dot{E}_O(t) + E_O(t) = E_{in}(t) \qquad \text{---- (3.3.4)}$$

where $\omega_n$ is the natural undamped frequency of the system and $\zeta$ is the damping ratio.

Hence,

$$\omega_n = \frac{f_c}{\sqrt{MN}} \Rightarrow f_n = \frac{1}{2\pi}\left(\frac{f_c}{\sqrt{MN}}\right) \qquad \text{---- (3.3.5)}$$

and

$$\zeta = \tfrac{1}{2}\sqrt{\frac{M}{N}} \qquad \text{---- (3.3.6)}$$

(a) Transient Solution

Let $D^2E_O + 2\zeta\omega_n DE_O + E_O = 0$ ---- (3.3.7)

Choosing $E_O = Ae^{\lambda t} \neq 0$

we have $\lambda_{1,2} = -\omega_n \pm \omega_n\sqrt{(\zeta^2 - 1)}$ ---- (3.3.8)

If $\zeta < 1.0$ then the $\lambda_i$ are complex conjugate roots, hence,

$$\lambda_{1,2} = -\omega_n \pm j\omega_n\sqrt{(1 - \zeta^2)} \qquad \text{---- (3.3.9)}$$

If $\zeta > 1.0$ the $\lambda_i$ are real and unequal, and if $\zeta = 1.0$ the roots are coincident in which case $\lambda_{1,2} = -\zeta\omega_n$.

(b) /

(b) Particular Integral

Then,

$$D^2E_O + 2\zeta\omega_n DE_O + E_O = E_{in} \qquad \text{---- (3.3.10)}$$

Considering a step input only the steady state solution is

$$E_O = E_{in} \qquad \text{---- (3.3.11)}$$

The complete solution for the underdamped case is:

$$E_O(t) = E_{in}\left[1 - e^{-\zeta\omega_n t}\left(\cos\omega_d t + \frac{\zeta}{\sqrt{(1-\zeta^2)}}\sin\omega_d t\right)\right] \qquad \text{---- (3.3.12)}$$

where

$$\omega_d = \omega_n\sqrt{(1-\zeta^2)}, \quad 0 \leqslant \zeta \leqslant 1.0, \quad t \geqslant 0$$

The transient solution is a damped oscillation and it can be shown that the time taken for $E_O(t)$ to reach its peak value is:

$$t_p = \frac{\pi}{\omega_d} \qquad \text{---- (3.3.13)}$$

and

$$E_{Omax} = E_{in}\left(1 + e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}\right) \qquad \text{---- (3.3.14)}$$

A programme was devised which simulates the behaviour of a second order stochastic system. Changes in damping ratio were achieved by varying the sizes of the counters used in the two integrators. The programme is presented in APPENDIX 3C.

Information is not represented using the bipolar mapping, but as simple probabilities so that statistical tests can be carried out on the results. In all cases the input probability is 0.5 (2048 states), and only the damping ratio was varied.

The /

The following cases were examined:

| Damping Ratio $\frac{1}{2}\sqrt{\frac{M}{N}}$ | N (states) | M (states) | I/P Probability (0.5) |
|---|---|---|---|
| 0.25 | 4096 | 1024 | 512 |
| 0.50 | 4096 | 4096 | 2048 |
| 1.00 | 1024 | 4096 | 2048 |
| 1.414 | 1024 | 8192 | 4096 |

FIGURE 3.3(b)

The results are displayed in Figure 3.3(c) - Figure 3.3(i).

The underdamped cases exhibit an interesting feature. The simple analysis led us to believe that all transients would die away, but they obviously do not. For example, $\zeta = 0.5$, the expected response is of the form shown in Figure 3.3(j) but instead we get the output shown in Figure 3.3(k). See also Figure 3.3(d), and Figure 3.3(e). Instead of the system settling down to a steady state level it has a small deterministic steady state oscillation.

These perturbations are not random because:

1. they are periodic;
2. the first stage integrator output also has a small oscillation leads the output;
3. statistical tests give a mean output probability of 0.5, (2048/4096), but the standard deviation is not what we would expect from a stationary Bernoulli sequence, and, it is very much greater than the theoretical value;
4. the theoretical standard deviation associated with a probability of 0.5 is 32 states, however, with a damping ratio of 0.25 the estimated standard deviation is 91 states;

5. /

5. for $\zeta = 0.28$ the amplitude of this oscillation can be anything up to 150 states which is far greater than three standard deviations when the probability is 0.5;

6. the frequency of oscillation is approximately that of the damped natural frequency, $\omega_n \sqrt{(1 - \zeta^2)}$;

7. the same effects can be seen for a damping ratio of 0.5 but they are much less severe and for $\zeta \geqslant 1.0$ there are no oscillations and the theoretical standard deviation is obtained (see Figure 3.3(ℓ));

8. the same circuit was set up on a conventional analogue computer and noise was injected into various parts of the system with the result that steady state oscillations were observed on the output.

The above investigations were based on simple statistical tests. For a random variable, X, and sample size, K, the mean was calculated in the following manner:

$$\mu = \frac{\sum_{\ell=1}^{K} X_\ell}{K} \qquad \text{---- (3.3.15)}$$

and the variance,

$$\sigma^2 = \frac{\sum_{\ell=1}^{K} (X_\ell - \mu)^2}{K - 1} \qquad \text{---- (3.3.16)}$$

The first analysis ignored the effects of random variance on the system. We can add a term $\phi(t)$ to the input, $E_i(t)$, to represent this variance, and in a similar manner, a term $\psi(t)$ can be added to the output signal to represent the effect of $\phi(t)$. If these corrections are added into equation (3.3.3), we have: /

we have:

$$\frac{MN}{f_c^2} D^2(E_o(t) + \psi(t)) + \frac{M}{f_c} D(E_o(t) + \psi(t)) + (E_o(t) + \psi(t)) = E_{in}(t) + \phi(t) \qquad \text{---- (3.3.17)}$$

Employing the principal of superposition we obtain:

$$\frac{MN}{f_c^2} \ddot{\psi}(t) + \frac{M}{f_c} \dot{\psi}(t) + \psi(t) = \phi(t) \qquad \text{---- (3.3.18)}$$

We know that $E_{in}$ is represented by a stationary sequence, and in this context $\phi(t)$ represents the local variation of the input probability with respect to time. It can be argued that $\phi(t)$ can take on values up to three standard deviations from the mean generating probability for short intervals, although its mean value is zero. If the variations in $\phi(t)$ are slow enough, $\psi(t)$ will try to follow them, but, the time variation of $\psi(t)$ must be governed by the circuit parameters such as the damping ratio and the natural frequency. Since the mean value of $\phi(t)$ is zero the mean value of $\psi(t)$ must also be zero.

Using our knowledge of the properties of a second order system and stationary Bernoulli sequences we can find an empirical expression for the maximum standard deviation to be expected from the counter of a second order ADDIE. From the experimental data we can approximate $\psi(t)$ to:

$$\psi(t) = \psi_{max} \sin \omega_d t \qquad \text{---- (3.3.19)}$$

But from our knowledge of the response of an under-damped second order system we can write:

$$\psi_{max} \triangleq \phi_{max} \left[ 1 + e^{\frac{-\pi\zeta}{\sqrt{(1-\zeta^2)}}} \right] \qquad \text{---- (3.3.20)}$$

but

$$\phi_{max} \triangleq 3\sigma_{in} \quad \text{where } \sigma_{in} = \sqrt{Np_{in}q_{in}}$$

Hence, /

Hence,

$$\psi_{max}(t) \triangleq 3\sigma_{in}\left[1 + e^{\frac{-\pi\zeta}{\sqrt{(1-\zeta^2)}}}\right] \sin(\omega_n\sqrt{(1-\zeta^2)})t \quad \text{---- (3.3.21)}$$

From equation (3.3.21) we see that the average of $\psi_{max}(t)$ is a time average and

$$\overline{\psi_{max}(t)} = 0 \quad \text{---- (3.3.22)}$$

The variance of $\psi_{max}(t)$ can be defined in the following way:

$$\sigma\psi_{max}^2 = \frac{1}{T}\int_0^T (\psi_{max}(t) - \overline{\psi_{max}(t)})^2 dt \quad \text{---- (3.3.23)}$$

where

$T = I(\frac{2\pi}{\omega_d})$ and I is a large integer.

Then we can say:

$$\sigma\psi_{max}^2 = \{3\sigma_{in}\left[1 + e^{\frac{-\pi\zeta}{\sqrt{(1-\zeta^2)}}}\right]\}^2 \overline{\sin^2\omega_d t} \quad \text{---- (3.3.24)}$$

$$\Rightarrow \sigma\psi_{max}^2 \triangleq \frac{\{3\sigma_{in}\left[1 + e^{\frac{-\pi\zeta}{\sqrt{(1-\zeta^2)}}}\right]\}^2}{2} \quad \text{---- (3.3.25)}$$

$$\Rightarrow \sigma\psi_{max} \triangleq 2.121\,\sigma_{in}\left[1 + e^{\frac{-\pi\zeta}{\sqrt{(1-\zeta^2)}}}\right] \quad \text{---- (3.3.26)}$$

Example:

Suppose the input probability of a second order ADDIE is 0.5 and one standard deviation is 32 states. If the damping ratio is 0.25 then the variance observed on the output counter is:

$$\sigma\psi_{max}^2 \triangleq \frac{\{100(1 + e^{\frac{-13x0.25}{(1-0.252)}}\}^2}{2}$$

$$\triangleq \frac{(143)^2}{2}$$

$$\Rightarrow \sigma\psi_{max} = 0.707 \times 143 = \underline{101 \text{ states}}$$

which /

which agrees well with the experimental value of 91 states in that the experimental value is less than the maximum predicted value.

A number of conclusions can be drawn from these studies.

1. Any stochastic automata theory which can be produced for the second order system will have to include the natural frequency and the damping ratio so that the resulting time domain expression for the output will be very much more complicated than that for a conventional machine.

2. Underdamped second order ADDIEs may be useless as output interfaces unless further filtering can be introduced and this could reduce the system bandwidth drastically.

3. The second order ADDIE can be adapted to provide stochastic amplification, See Figure 3.3(m). The equation governing this circuit is:

$$\frac{nMN}{f_c^2}\ddot{E}_O(t) + \frac{M}{f_c}\dot{E}_O(t) + E_O(t) = nE_{in} \qquad n \geqslant 1.0 \qquad \text{---- (3.3.27)}$$

where

$$\omega_n = \frac{f_c}{\sqrt{nMN}} \qquad \text{---- (3.3.28)}$$

and

$$\zeta = \tfrac{1}{2}\sqrt{\frac{M}{nN}} \qquad \text{---- (3.3.29)}$$

If $E_{in}$ is a step input then the steady state output is

$$E_O = nE_{in} \qquad \text{---- (3.3.30)}$$

Thus as n increases the damping is decreased and the stability is affected.

ie,

$$\zeta,\ \omega_n \propto \frac{1}{\sqrt{n}}$$

Improved /

Improved stability can be obtained by using the circuit illustrated in Figure 3.3(n). Here the damping is made proportional to n.

$$E_O(t) = \frac{f_c}{M} \int (E_O(t) - E_2(t)dt + c_1 \qquad \text{---- (3.3.31)}$$

and

$$E_2(t) = \frac{f_c}{N} \int (E_{in}(t) - \frac{E_O(t)}{n})dt + c_2 \qquad \text{---- (3.3.32)}$$

$$\Rightarrow \quad \frac{nMN}{f_c^2} \ddot{E}_O(t) + \frac{nM}{f_c} \dot{E}_O(t) + E_O(t) = nE_{in}(t) \qquad \text{---- (3.3.33)}$$

If $E_{in}$ is a step input then,

$$E_O(t) = nE_{in} \qquad \text{---- (3.3.34)}$$
$$n \geqslant 1.0$$

Transient Response

Suppose $E_O = e^{\lambda t}$, then

$$\lambda_{1,2} = -\frac{f_c}{2N} \pm j \frac{f_c}{nMN} \sqrt{(1 - \frac{nM}{4N})} \qquad \text{---- (3.3.35)}$$

For $n = \frac{4N}{M}$ we get a critically damped response, and if N = M, the overall gain is four. If n is greater than four the response is overdamped and the bandwidth is reduced.

This circuit was not simulated but was patched up on an actual stochastic computer. It gave the required amplification but the variance of the output was considerable - up to 5% of the dynamic range (2048 states) using the bipolar mapping.

## 3.4 Response of a Second Order Stochastic System with no Damping

The stochastic circuit illustrated in Figure 3.4(a) was simulated using a FORTRAN programme. See APPENDIX 3D. From the diagram we have:

$E_O$ /

$$\dot{E}_O = \frac{2f_c}{M}\int -E_O dt + c_1 \qquad \text{---- (3.4.1)}$$

and

$$E_O = \frac{2f_c}{N}\int \dot{E}_O \, dt + c_2 \qquad \text{---- (3.4.2)}$$

$$\Rightarrow \quad \frac{\ddot{E}_O}{\omega_n^2} + E_O = 0 \qquad \text{---- (3.4.3)}$$

$$E_O = (c_1 - c_2)\sin \omega_n t \qquad \text{---- (3.4.4)}$$

The results of the simulation are displayed in Figures 3.4(b) and 3.4(c). Both of the integrator states have been plotted against dimensionless time.

The second stage integrator follows a sine-wave law while the first follows a cosine-wave law.

The waveforms have a constant frequency, but both show amplitude instability. This can be explained quite simply in terms of the variances of the random sequences present in the circuit. The variance for a stationary random binomial sequence is:

$$\sigma^2 = Np(1 - p) \qquad \text{---- (3.4.5)}$$

Suppose p is a function of time, then since the sequence is still binomial, and if we consider an ensemble of such sequences we can write:

$$\sigma^2(t) = Np(t)(1 - p(t)) \qquad \text{---- (3.4.6)}$$

Let $\sigma_s^2$ be the variance on the output of the second stage integrator and $p_s(t)$ be the output generating probability at any instant.

Then

$$p_s(t) = \tfrac{1}{2} + \tfrac{1}{2}\,\frac{E_O \sin\omega t}{V} \qquad \text{---- (3.4.7)}$$

where $E_O$ is the amplitude of oscillator at the start of the first cycle.

Over the first cycle the variance is approximated to

$$\sigma_s^2(t) = N\left(\frac{1}{4} - \frac{1}{4}\left(\frac{E_O}{V}\right)^2 \sin^2\omega t\right) \qquad \text{---- (3.4.8)}$$

Let /

Let us examine the behaviour of $\sigma_s^2(t)$ as it varies with time. Then,

$$\frac{\partial \sigma_s^2(t)}{\partial \omega t} = N(0 - \frac{1}{4}(\frac{E_o}{V})^2 \sin 2\omega t) \qquad \text{---- (3.4.9)}$$

and

$$\frac{\partial \sigma_s^2(t)}{\partial \omega t} = 0 \quad \text{when } \omega t = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, \ldots, \frac{n\pi}{2}$$

when n is even $\sigma_s^2(t)$ has a maximum value of $\frac{N}{4}$ and the standard deviation is $\sqrt{\frac{N}{4}}$. When n is odd $\sigma_s^2(t)$ has a minimum value of $\frac{N}{4}(1 - (\frac{E_o}{V})^2)$ and the standard deviation is $\sqrt{\frac{N}{4}(1 - (\frac{E_o}{V})^2)}$ .

Let $\sigma_c^2(t)$ be the output variance from the first stage integrator, then,

$$\sigma_c^2(t) = \frac{N}{4}(1 - (\frac{E_o}{V})^2 \cos^2 \omega t) \qquad \text{---- (3.4.10)}$$

and

$$\frac{\partial \sigma_c^2(t)}{\partial \omega t} = 0 \quad \text{when } \omega t = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, \ldots, \frac{n\pi}{2}$$

when n is even $\sigma_c^2(t)$ has a minimum value of $\frac{N}{4}(1 - (\frac{E_o}{V})^2)$ and the standard deviation is $\sqrt{\frac{N}{4}(1 - (\frac{E_o}{V})^2)}$ and, when n is odd $\sigma_c^2(t)$ has a maximum value of $\frac{N}{4}$ and the standard deviation is $\sqrt{\frac{N}{4}}$. In this case both integrators have the same size of counter.

Thus for n even the second stage integrator inputs a maximum variance into the first stage integrator which in turn inputs a minimum variance into the second stage. Thus when the second stage integrator outputs a probability of 0.5 the first stage integrator counter shows large variations from the expected value. Hence the /

the new amplitude at the start of the second cycle is:

$$E_1 = E_0 \pm 3\sigma(0.5) \qquad \text{---- (3.4.11)}$$

and at the start of the third cycle the amplitude of oscillation is,

$$E_2 = E_1 \pm 3\sigma(0.5) \qquad \text{---- (3.4.12)}$$

Hence at the start of the jth cycle the amplitude is,

$$E_j = E_{j-1} \pm 3\sigma(0.5) \qquad \text{---- (3.4.13)}$$

This means that there is no way in which the amplitude of oscillation can be determined at the start of any cycle since the errors accummulate.

If N = 4096 states the variance is 1024 states so that three standard deviation is 96 states. It can be seen that consecutive amplitudes of both outputs vary by up to ±100 states. In the circuit simulated the integrators can only count up or down but they cannot remain stationary because the two inputs of each integrator are command. Hence the variance at the output will be worse than if each integrator had two uncorrelated inputs. For a one input integrator, there will always be uncertainty in the amplitude during the next cycle of $\pm 3/2\sqrt{N}$ states. If the normalised standard deviation is $\sigma/N$ then the uncertainty in the next amplitude is $\pm 3/2\sqrt{N}$. This error will decrease as the counter size, N, increases. Hence the per unit analogue uncertainty for a bipolar mapping is thus $\pm 3/\sqrt{N}$ so that the jth amplitude is given by:

$$\frac{E_j}{V} = \frac{E_{j-1}}{V} \pm \frac{3}{\sqrt{N}} \qquad \text{---- (3.4.14)}$$

Equation (3.4.14) indicates that small amplitude wave forms cannot be accurately produced.

We /

We can use the analogue model to predict the behaviour of a simulation. Consider the stochastic oscillator discussed above. We can predict the cycle length in terms of the clock periods of a synchronous sequential circuit, but without specifically bringing time into the calculation. From equation (3.4.3) we have:

$$\ddot{E}_O + \omega_n^2 E_O = 0$$

where

$$\omega_n = \frac{2f_c}{(MN)^{\frac{1}{2}}}$$

Let $$_N\omega_n = \frac{\omega n}{f_c} = \frac{2}{(MN)^{\frac{1}{2}}} \qquad \text{---- (3.4.15)}$$

if M = N = 4096 and one-input integrators are used,

$$_Nf_n = \frac{2}{2\pi \times 4096}$$

$$\Rightarrow \quad _NT_n = \frac{T_n}{T_c} = \frac{2\pi \times 4.096 \times 10^3}{2} \quad \frac{25}{2} \times 10^3$$

clock cycles in cycle length

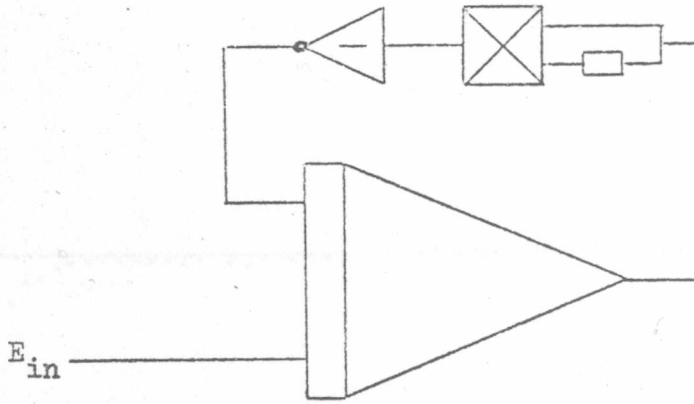
If these are 250 clock cycles to every printing interval (PI), then there are

$$\frac{25 \times 10^3}{500} = 50 \text{ PI/cycle length;}$$

and if there are 480 printing intervals to every experiment, there are

$$\frac{480}{50} \text{ cycle lengths} \triangleq 9.6 \text{ cycle lengths.}$$

Similar calculations may be done for overdamped and underdamped systems to determine the values of decaying exponential terms. See APPENDIX 3E.




SQUARE-ROOT EXTRACTION

FIG 3.1 (a)


$E_o$

Solution of a Non-linear Differential Equation $\frac{\dot{E}_O}{G} + E_O^2 = E_{in}$

(Square-root Extraction)

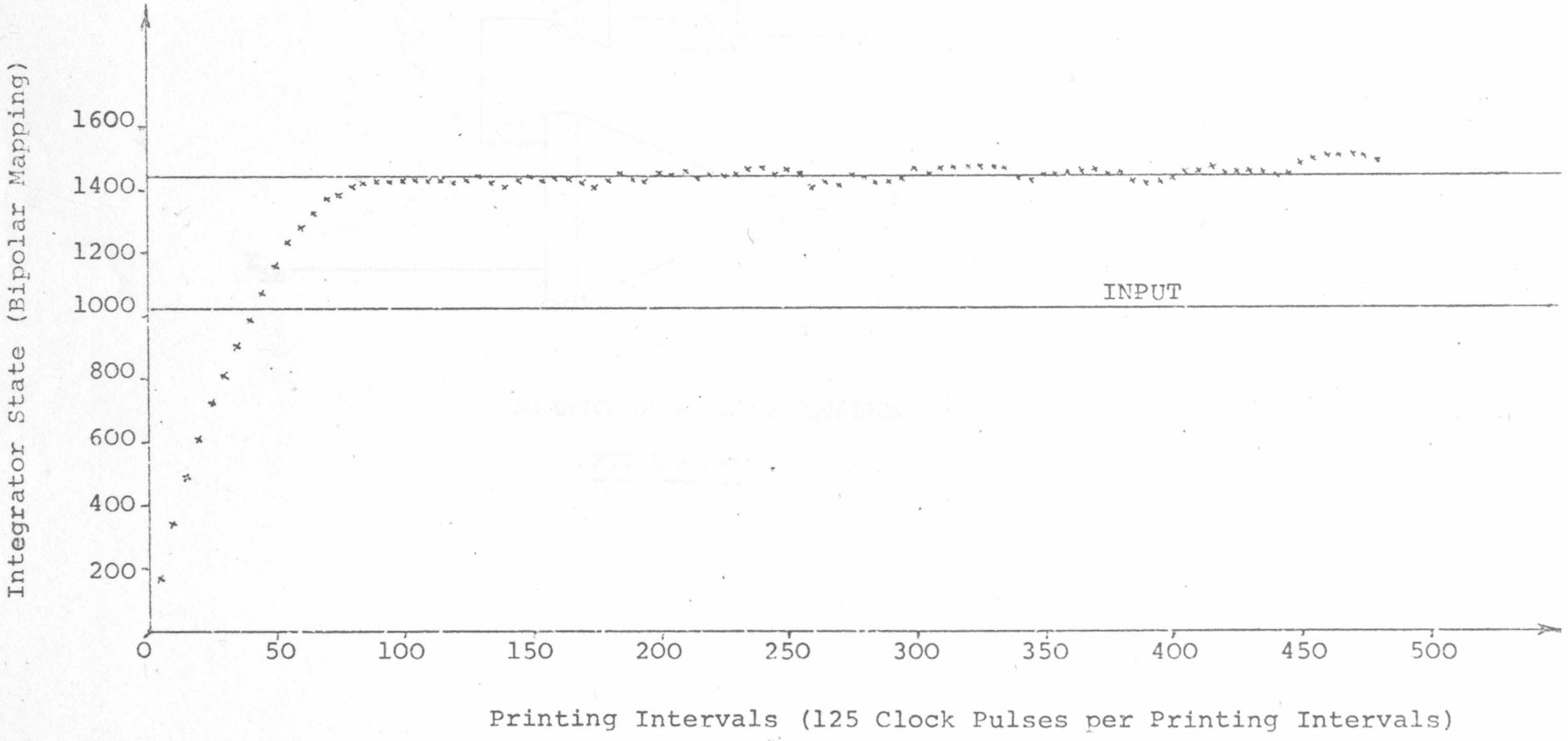
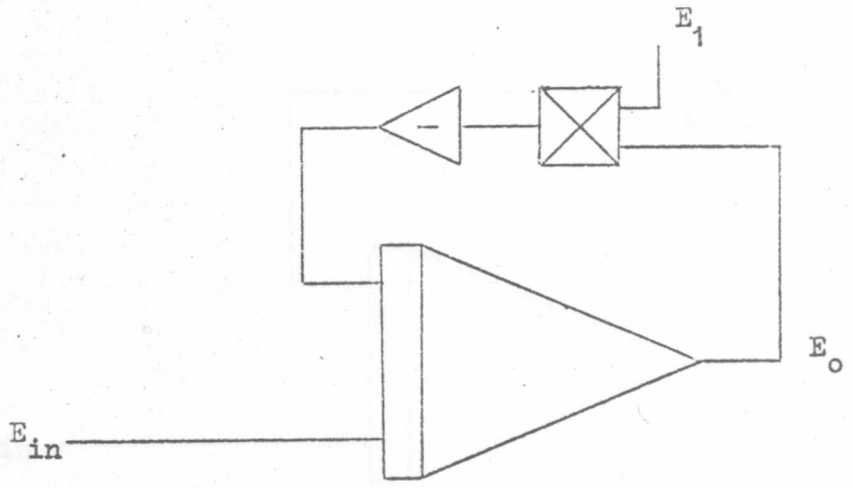
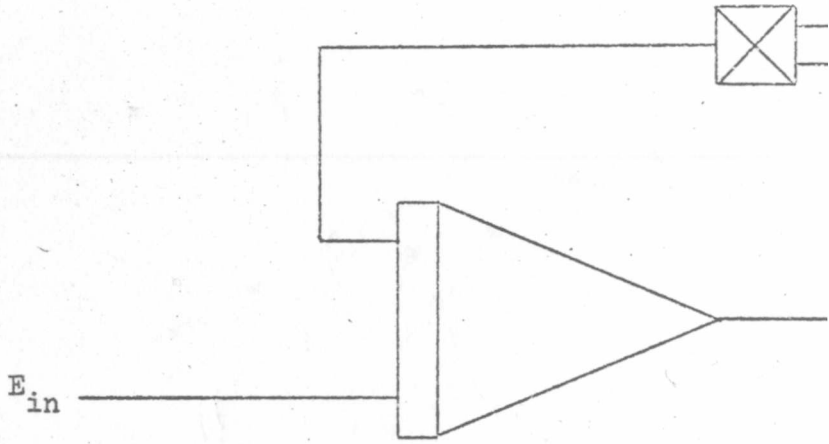



Figure 3.1(b)



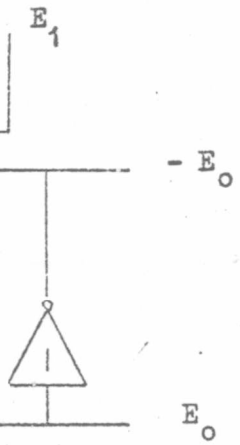

SOLUTION OF A LINEAR EQUATION

FIG 3.2 (a)



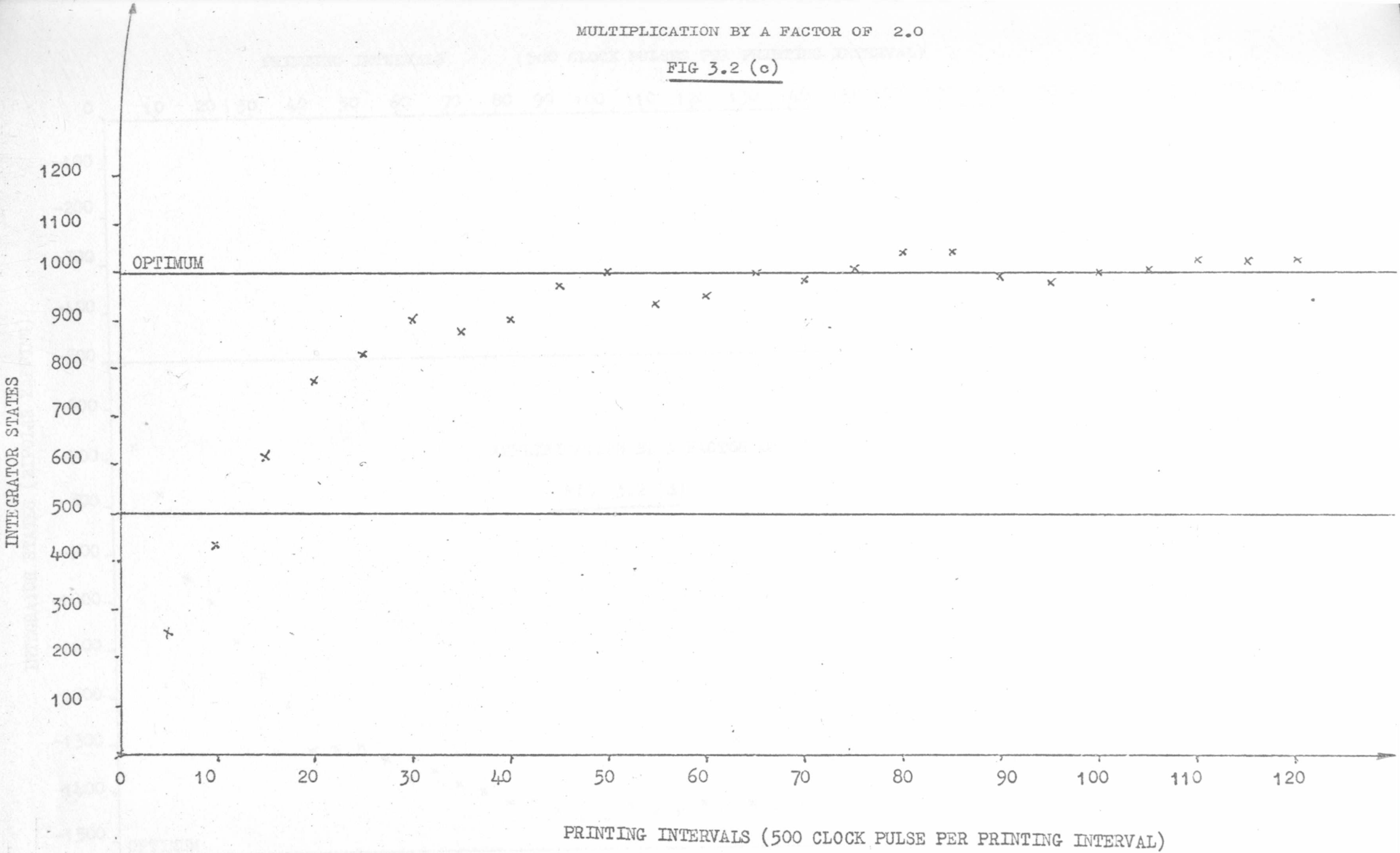

SOLUTION OF A LINEAR EQUATION

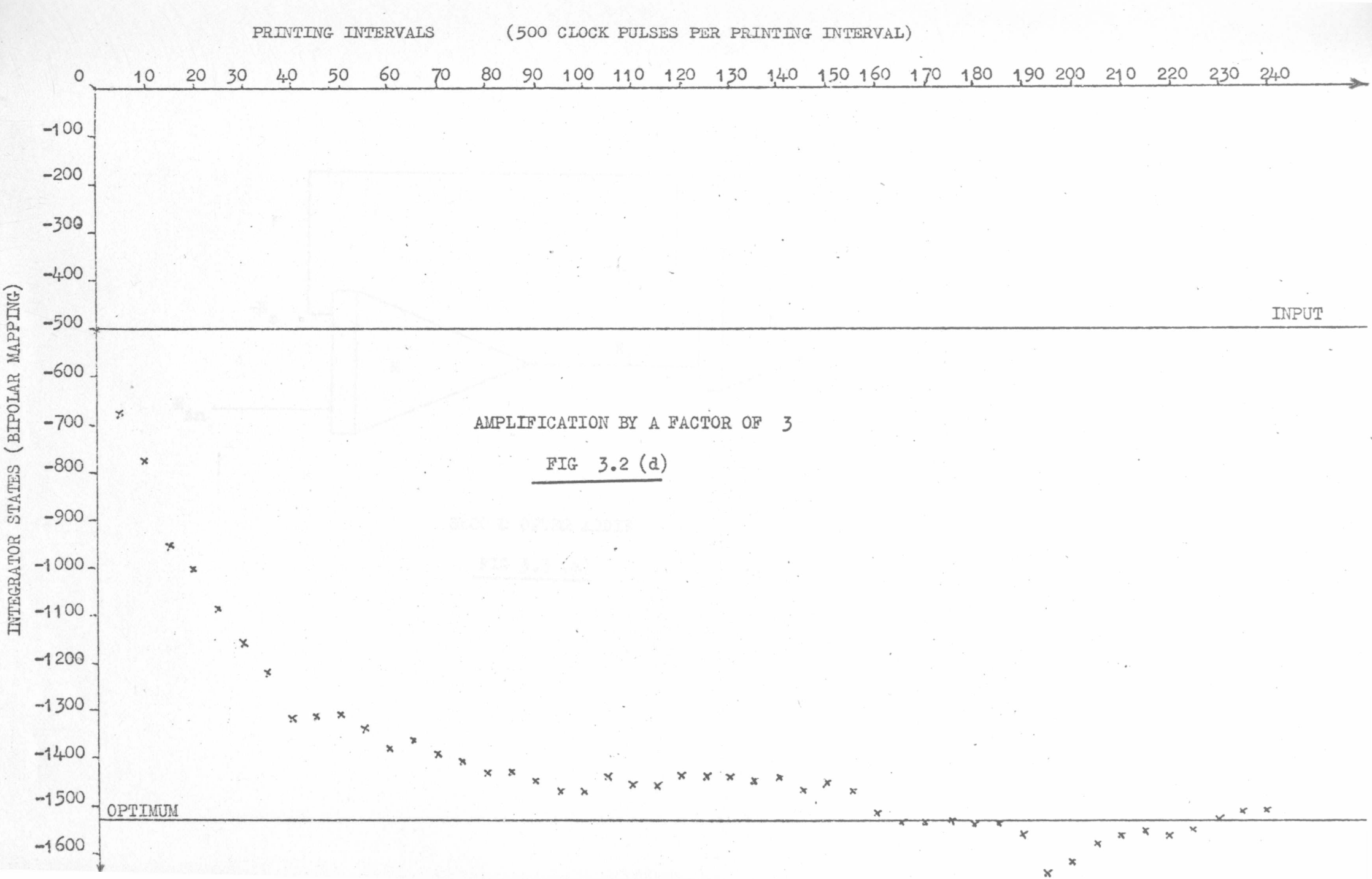
FIG 3.2 (b)


E1
- Eo
Eo

MULTIPLICATION BY A FACTOR OF 2.0

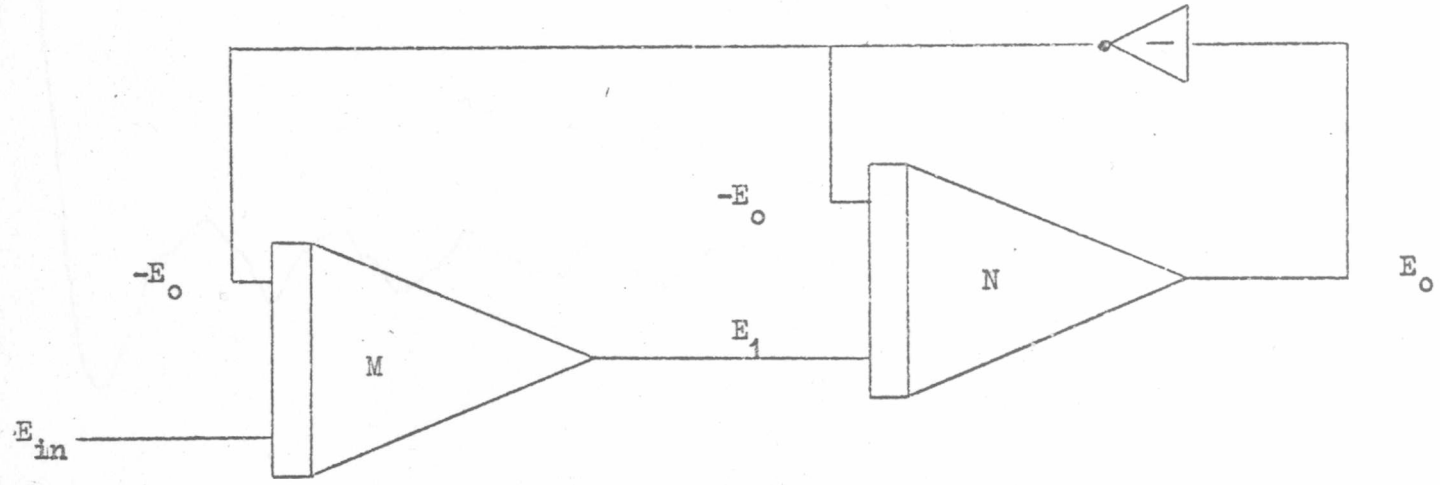
FIG 3.2 (c)





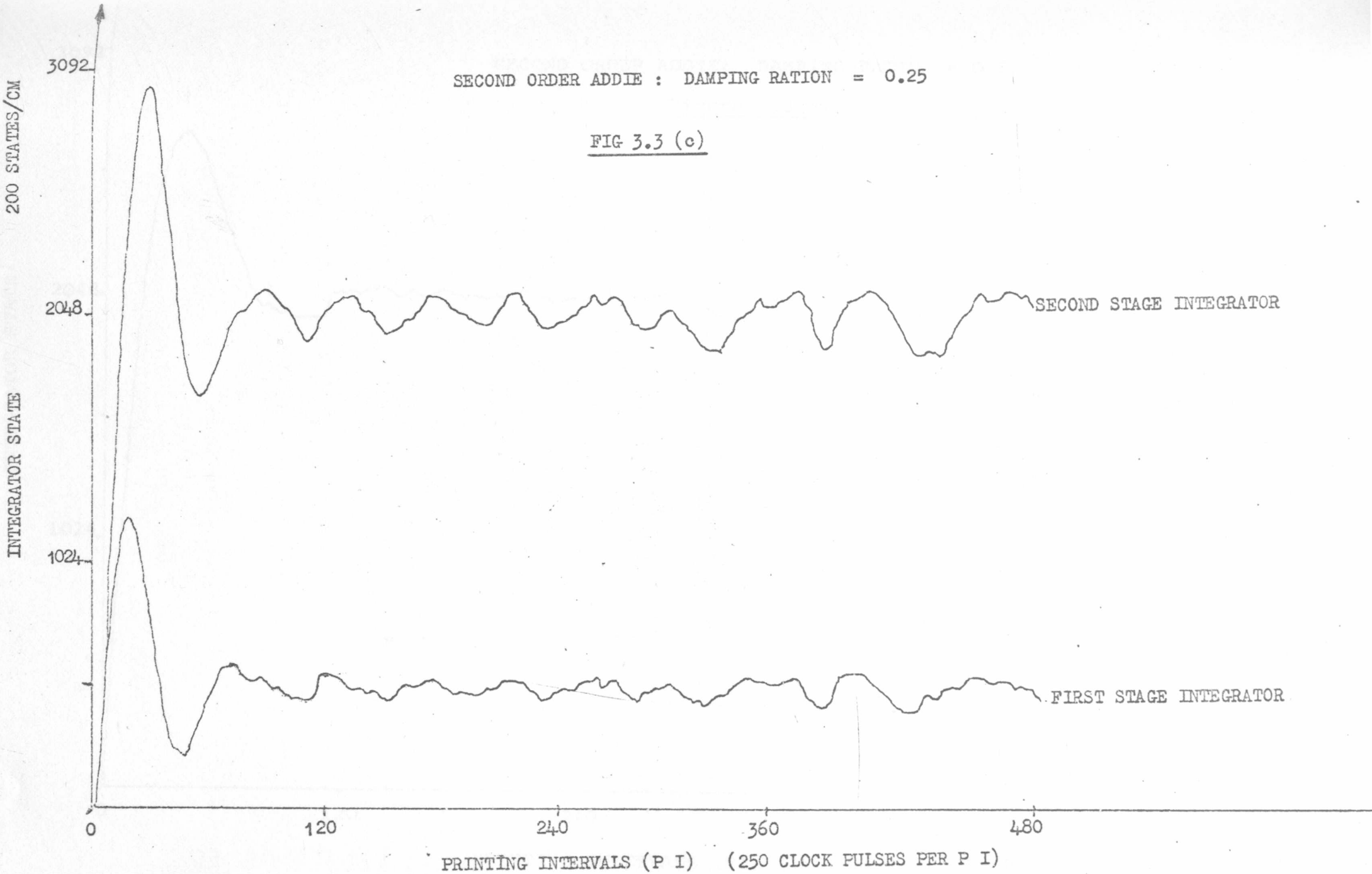

AMPLIFICATION BY A FACTOR OF 3

FIG 3.2 (d)



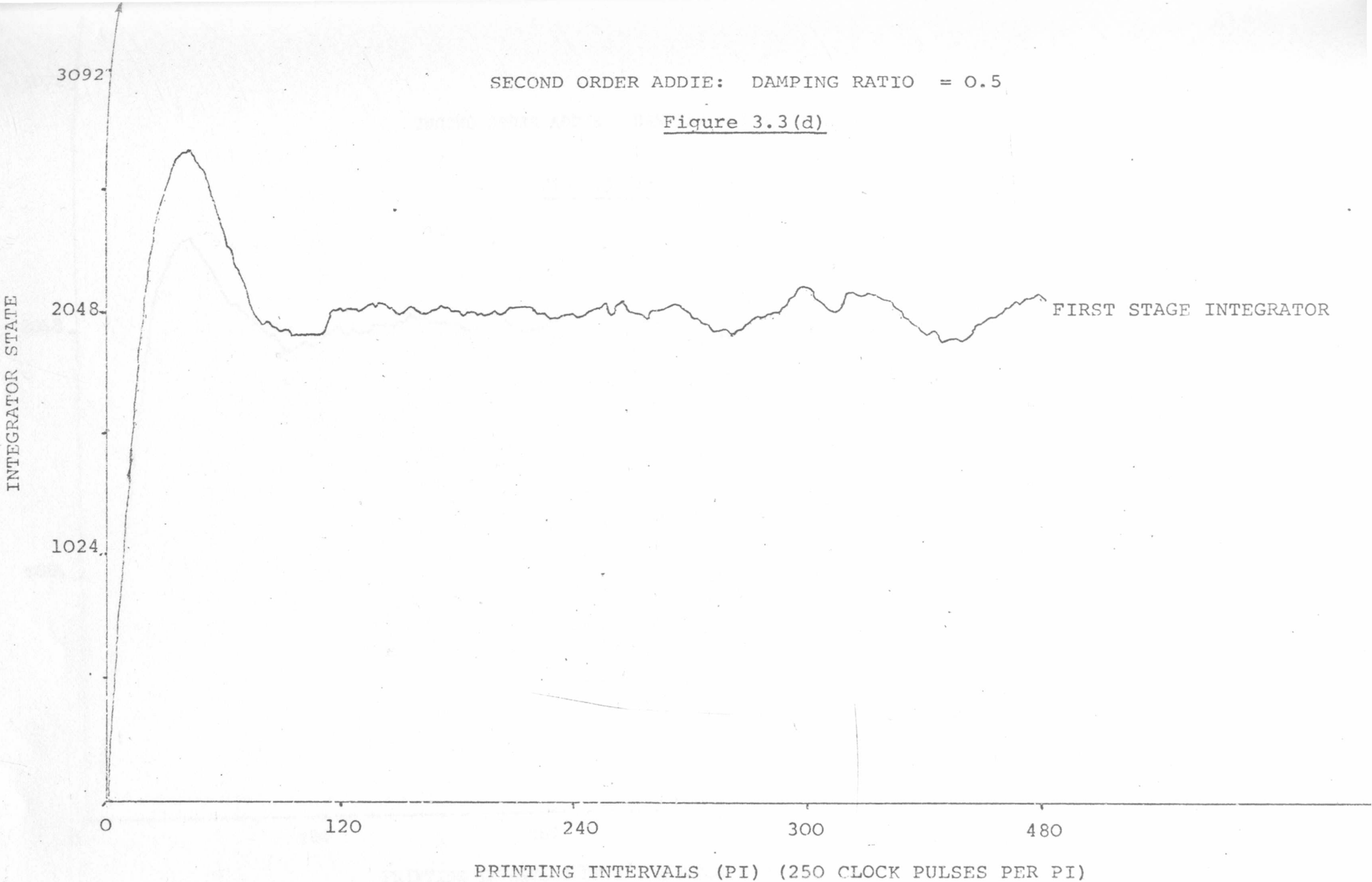

SECOND ORDER ADDIE

FIG 3.3 (a)


SECOND ORDER ADDIE : DAMPING RATION = 0.25
FIG 3.3 (c)
3092
2048
1024
200 STATES/CM
INTEGRATOR STATE
SECOND STAGE INTEGRATOR
FIRST STAGE INTEGRATOR
0
120
240
360
480
PRINTING INTERVALS (P I) (250 CLOCK PULSES PER P I)

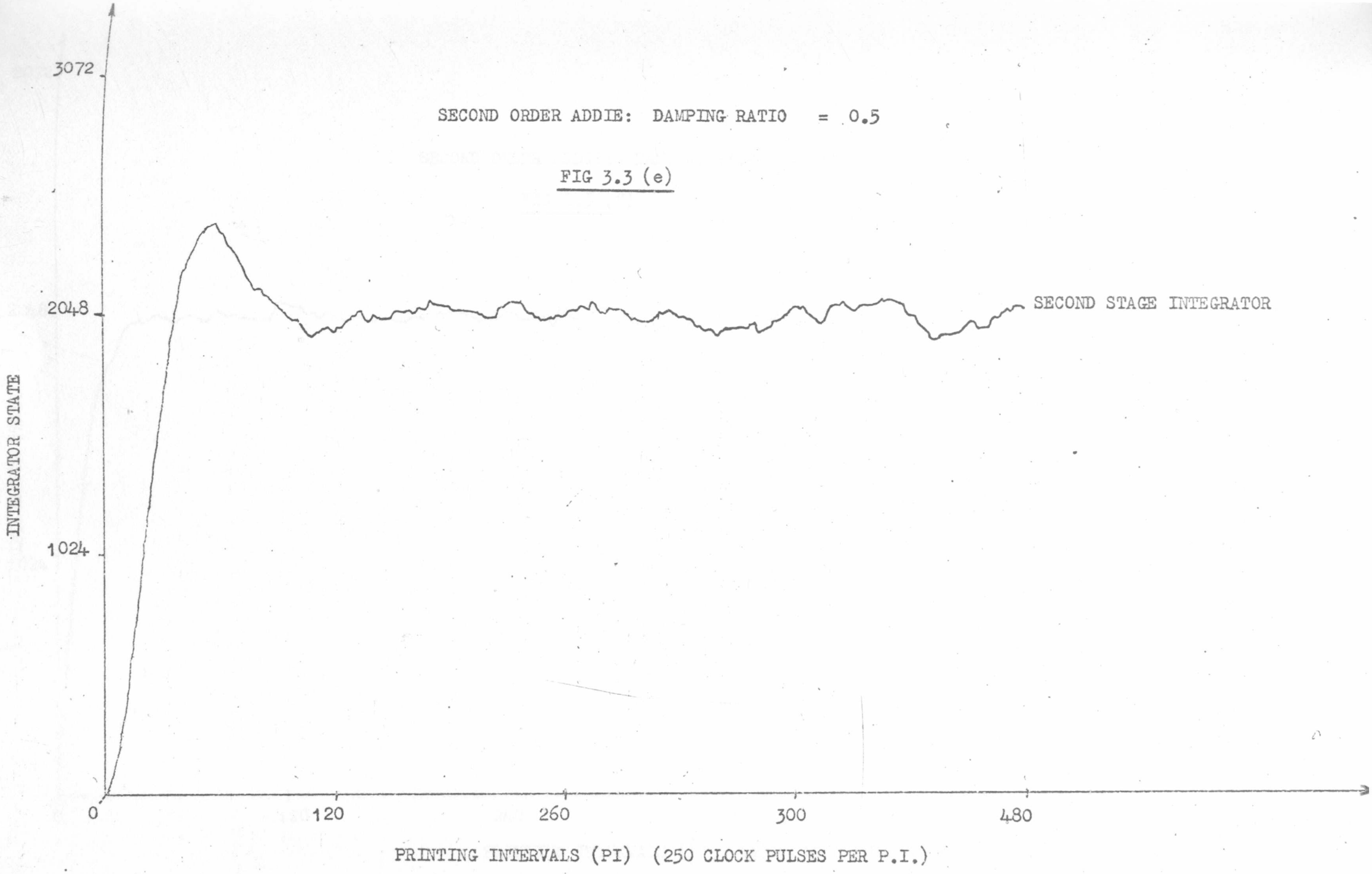
SECOND ORDER ADDIE: DAMPING RATIO = 0.5

Figure 3.3(d)



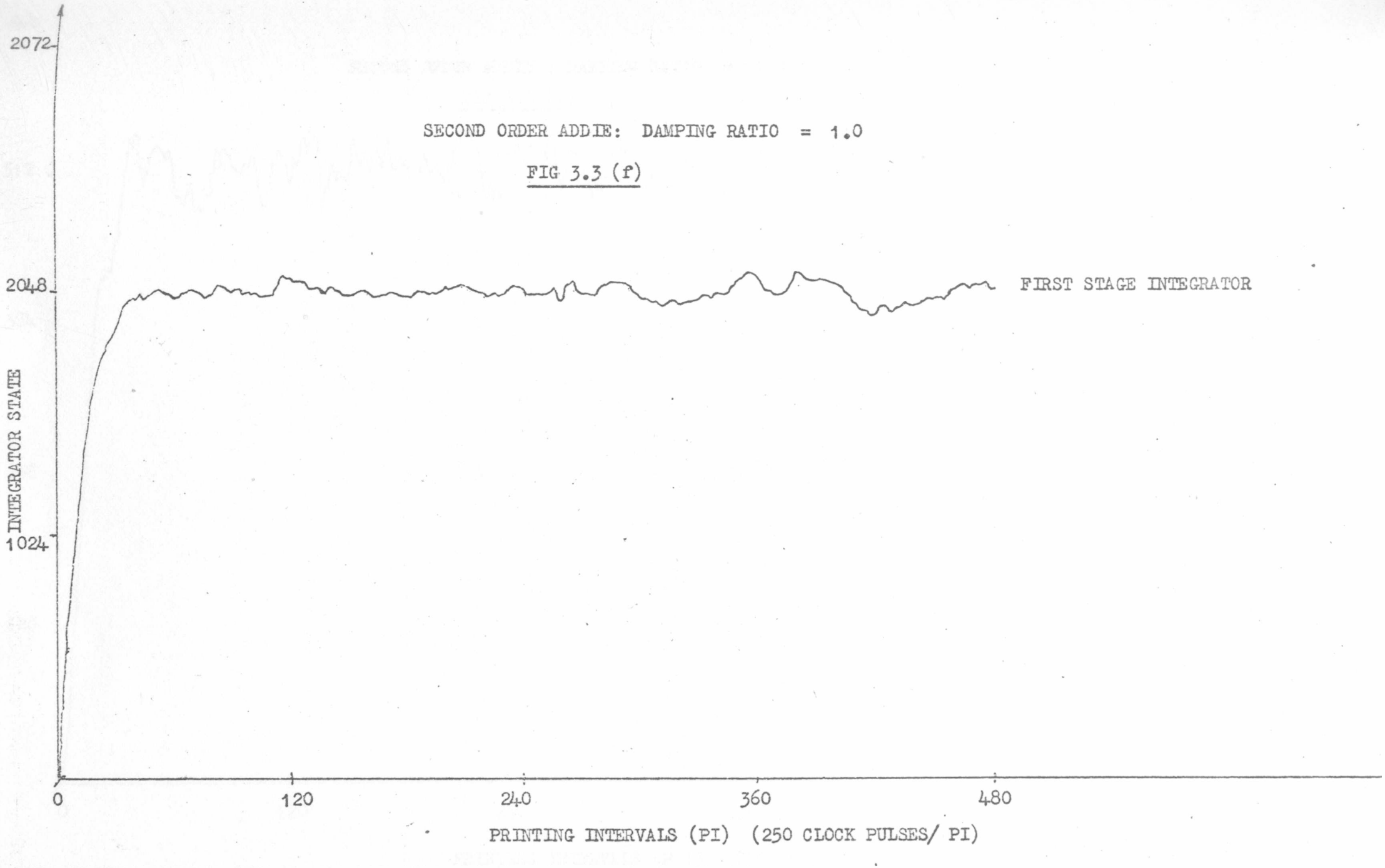
SECOND ORDER ADDIE: DAMPING RATIO = 0.5

FIG 3.3 (e)



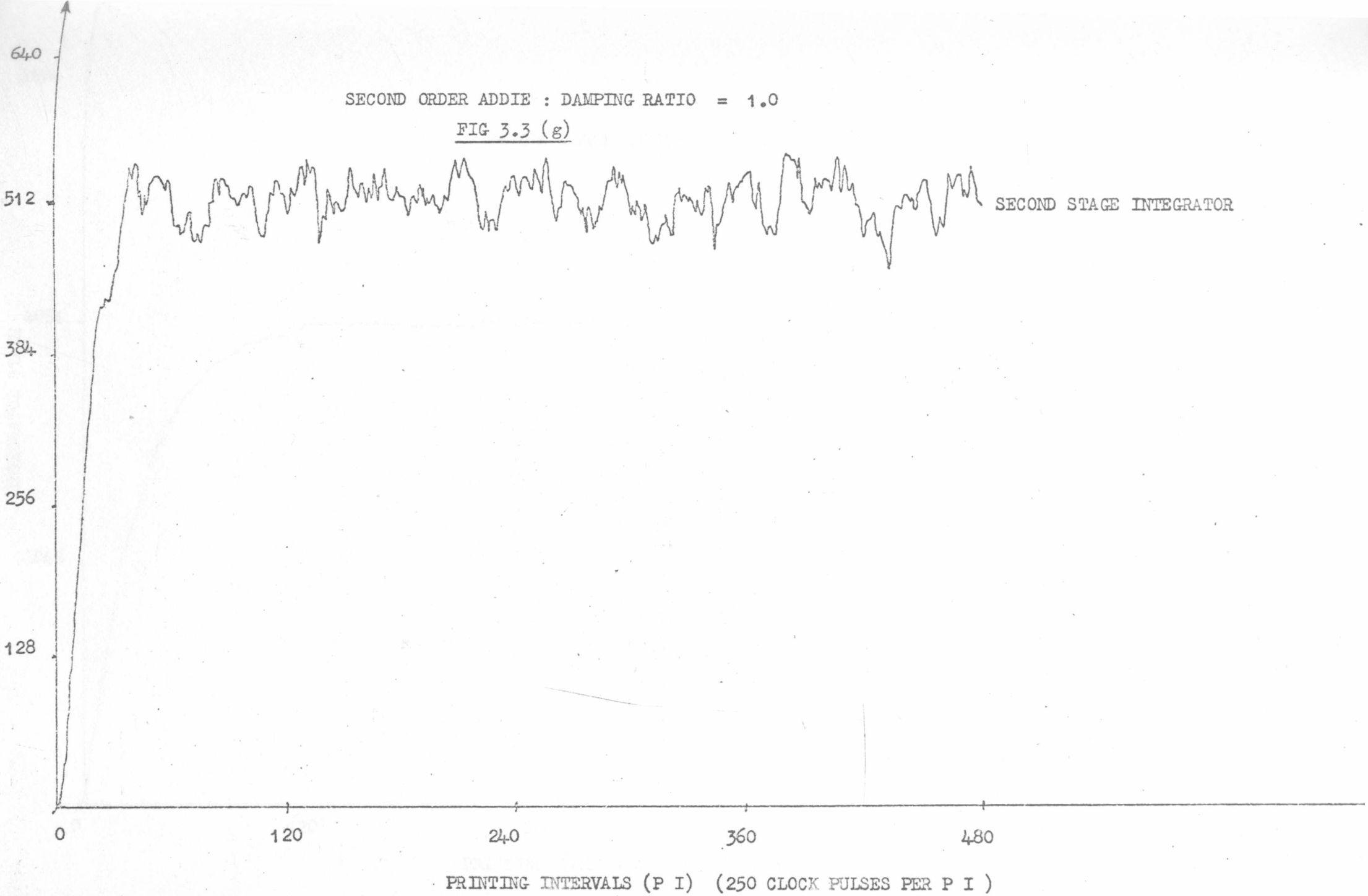
SECOND ORDER ADDIE: DAMPING RATIO = 1.0

FIG 3.3 (f)



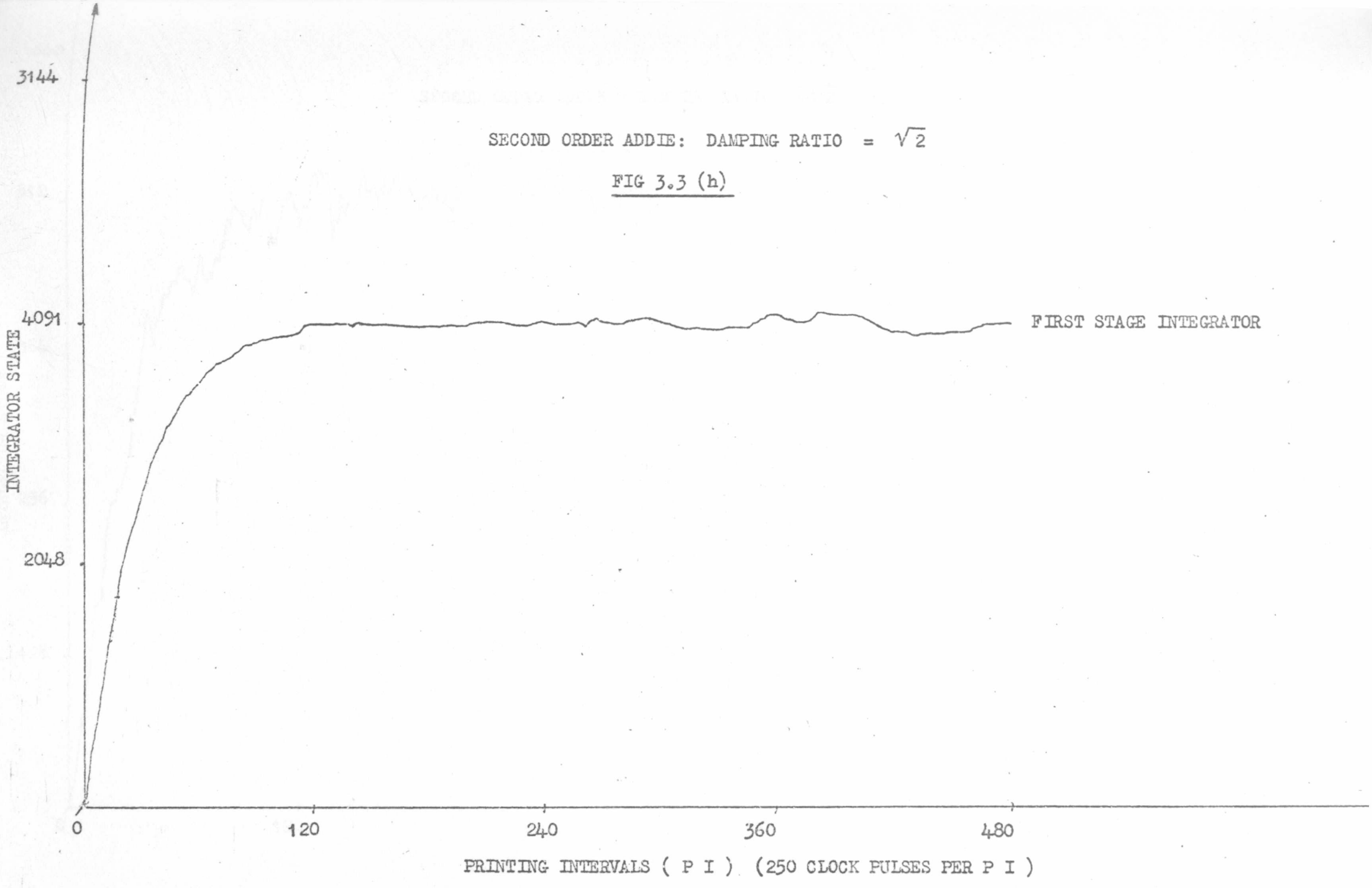
SECOND ORDER ADDIE : DAMPING RATIO = 1.0

FIG 3.3 (g)



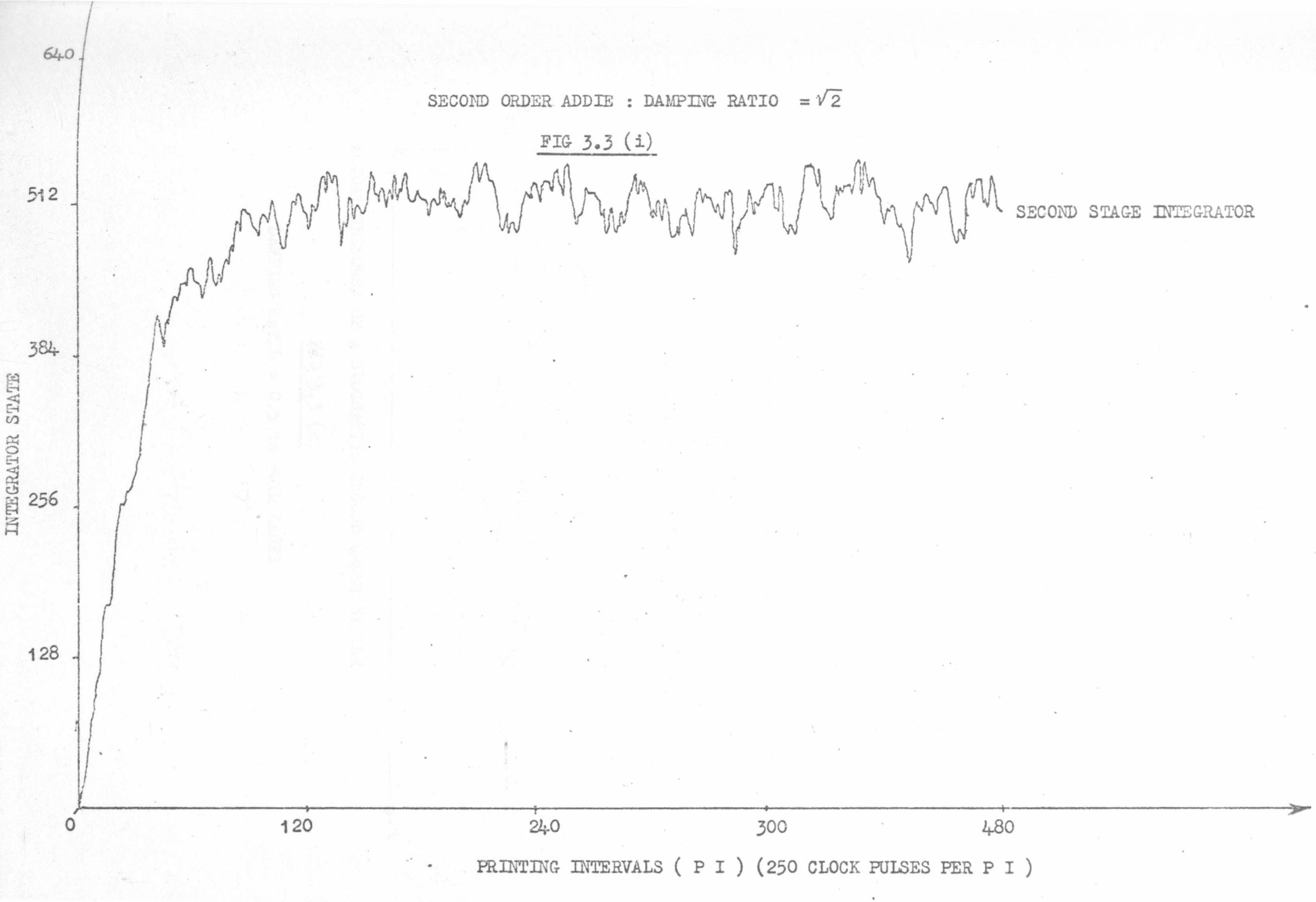
SECOND ORDER ADDIE: DAMPING RATIO = $\sqrt{2}$

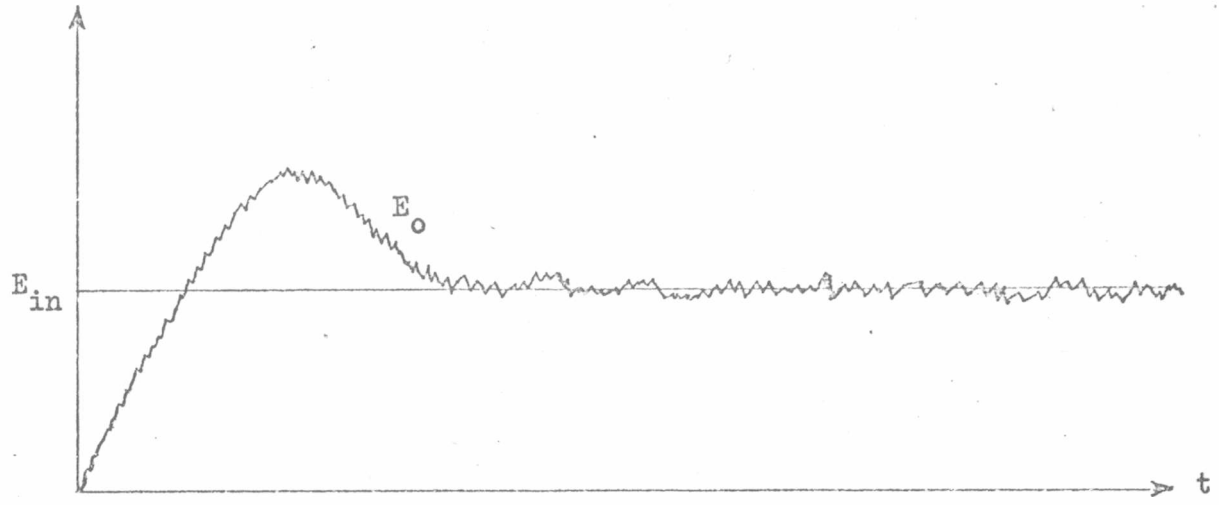
FIG 3.3 (h)



SECOND ORDER ADDIE : DAMPING RATIO $= \sqrt{2}$

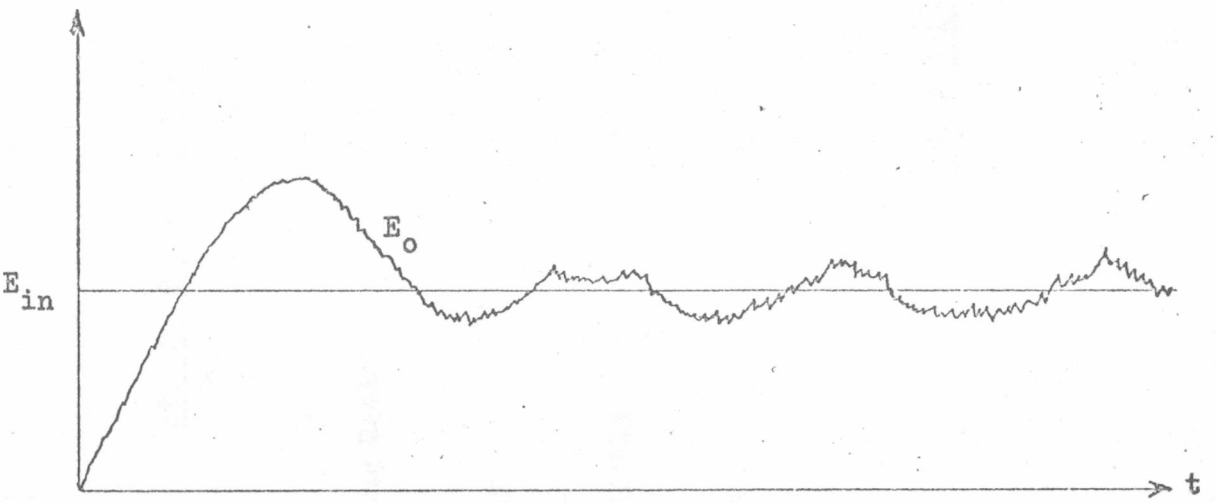
FIG 3.3 (i)






THEORETICAL RESPONSE OF A STOCHASTIC SECOND ORDER SYSTEM

FIG 3.3 (j)




ACTUAL RESPONSE OF A STOCHASTIC SECOND ORDER SYSTEM

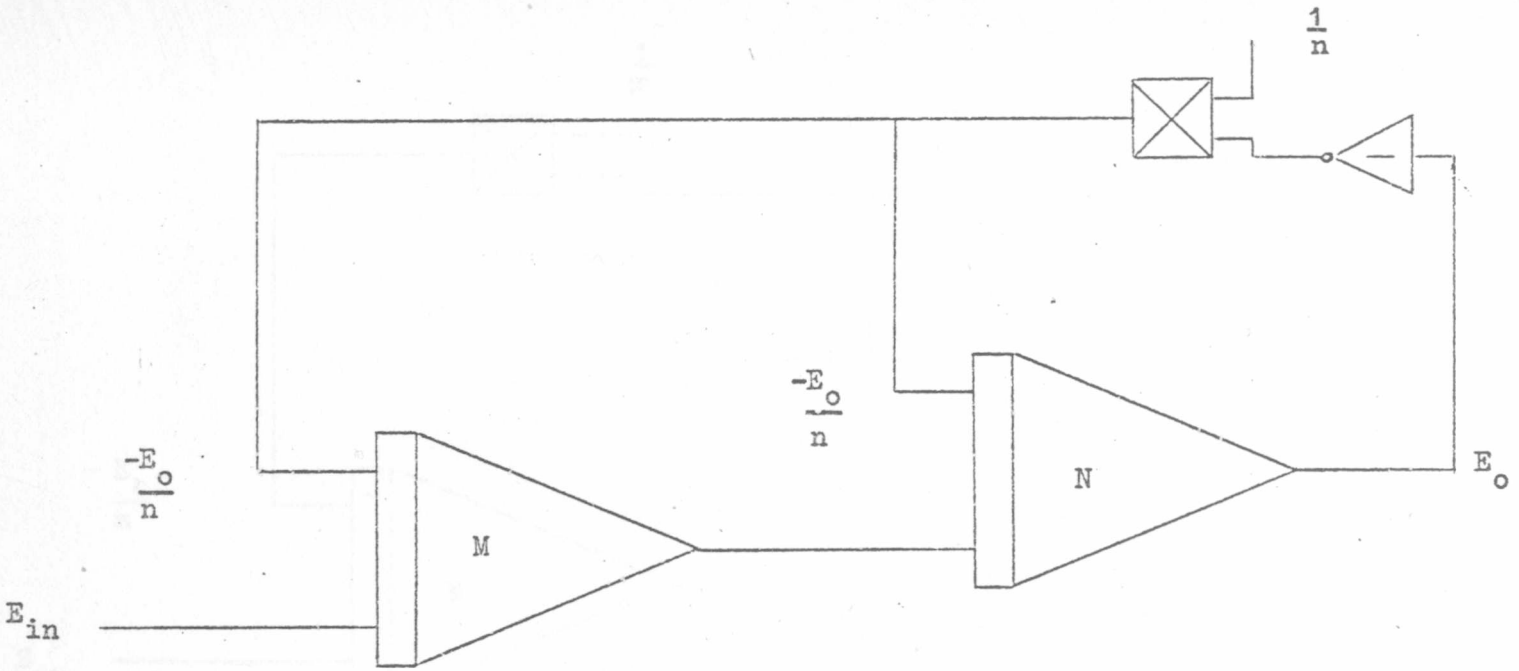
FIG 3.3 (k)

DAMPING RATIO = 0.5 in BOTH CASES

## SECOND ORDER SYSTEM - OVERDAMPED CASE

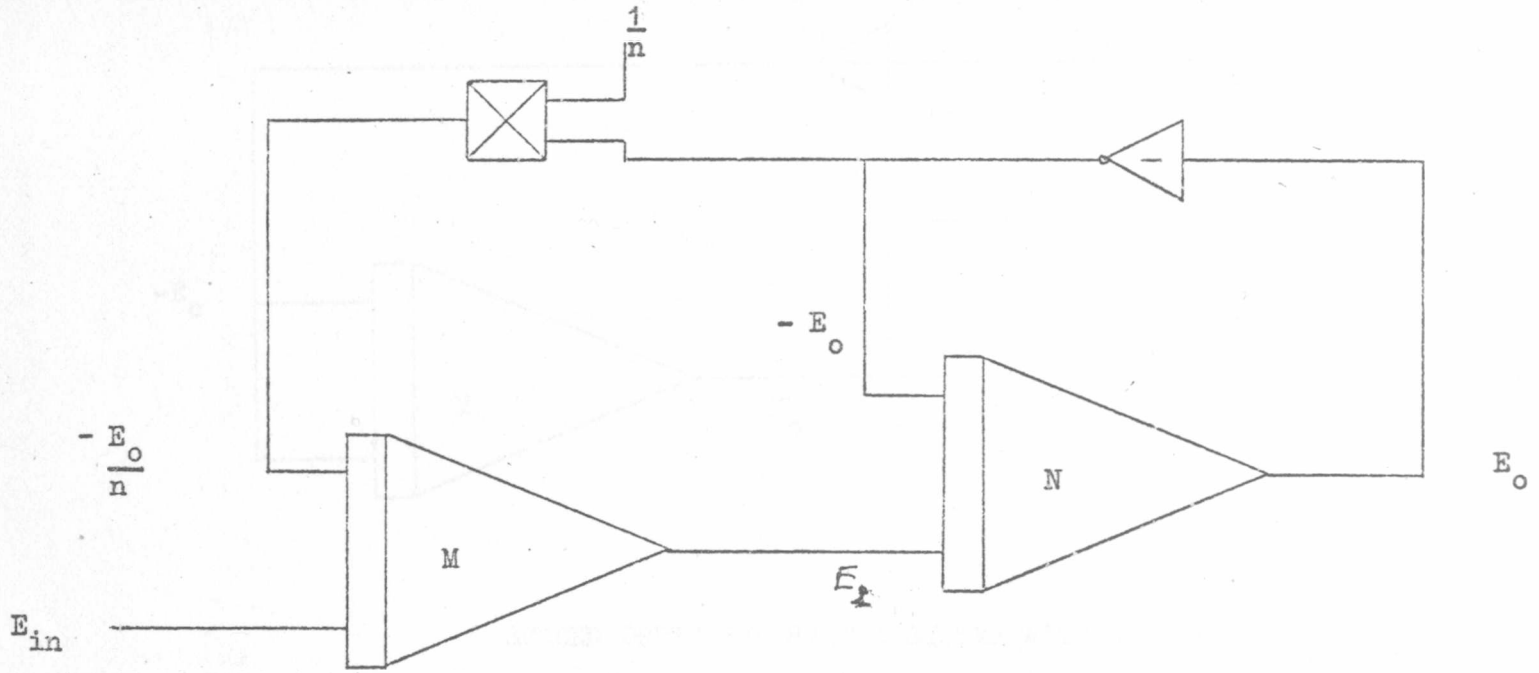
| Damping Ratio | 1.0 | 1.414 | Theoretical Prediction |
|---|---|---|---|
| MEAN | 509 | 511 | 512 |
| VARIANCE | 258 | 223 | 256 |
| S.D. | 16.06 | 14.9 | 16 |

Fig. 3.3 (1)



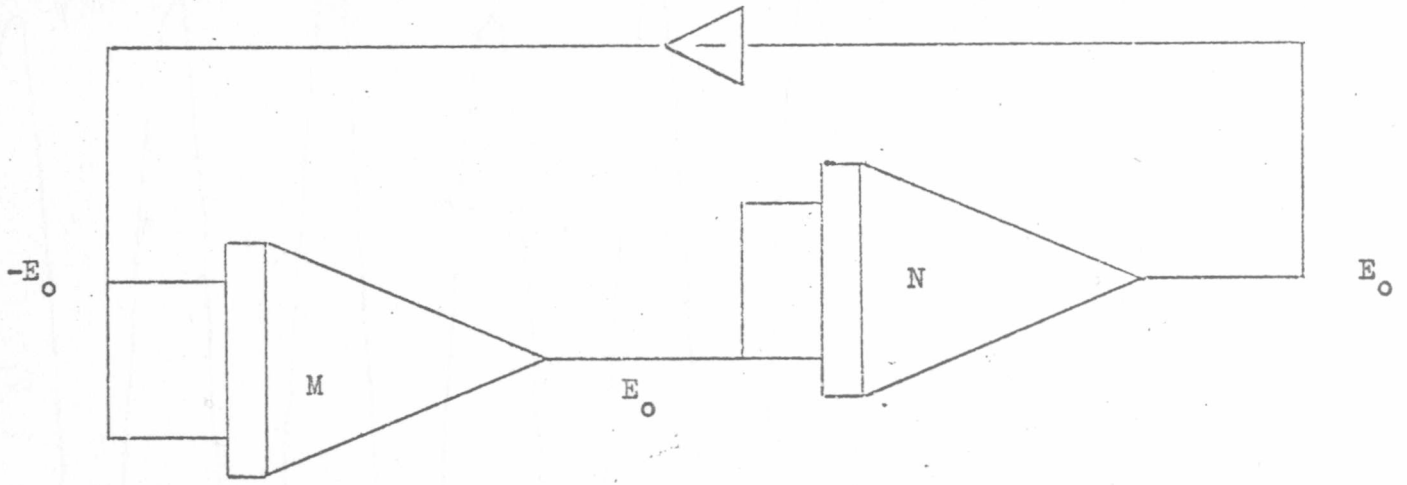

STOCHASTIC AMPLIFIER

FIG 3.3 (m)



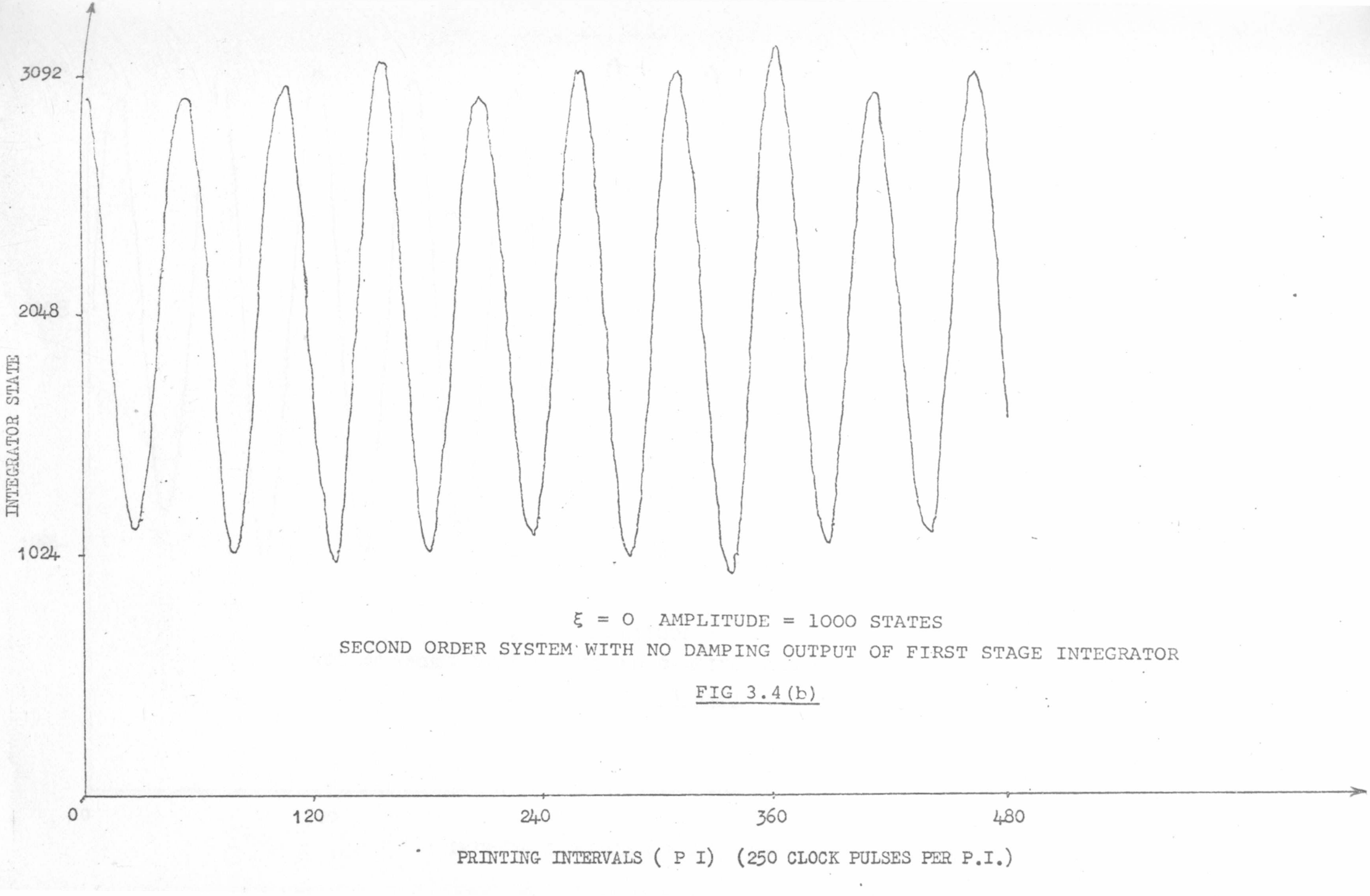

STOCHASTIC AMPLIFIER

Fig 3.3 (n)




SECOND ORDER STOCHASTIC SYSTEM WITH NO DAMPING

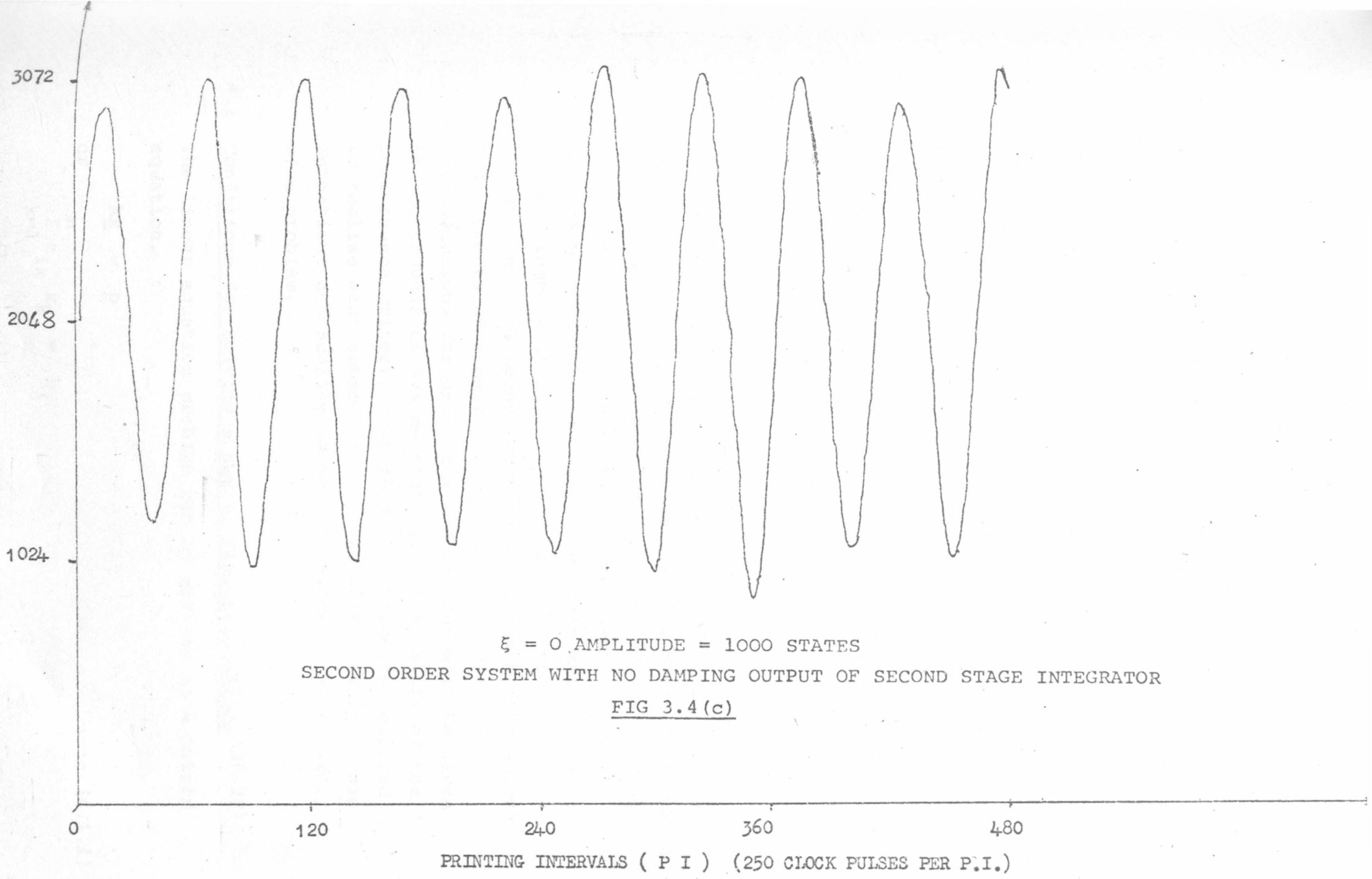
FIG 3.4 (a)




$\xi$ = 0 AMPLITUDE = 1000 STATES

SECOND ORDER SYSTEM WITH NO DAMPING OUTPUT OF FIRST STAGE INTEGRATOR

FIG 3.4(b)




$\xi$ = 0 AMPLITUDE = 1000 STATES

SECOND ORDER SYSTEM WITH NO DAMPING OUTPUT OF SECOND STAGE INTEGRATOR

FIG 3.4(c)

# CHAPTER 4

(In this chapter stochastic computer circuits which permit the solution of linear equations and matrix inversion are investigated.)

## 4. Introduction

In the last chapter some simple analogue circuits were investigated using the stochastic computing units available. These circuits involved the interconnection of few components to generate simple functions - usually an integrator with a simple feedback loop.

To solve sets of linear equations we need to be more ambitious in the number of computing units to be employed and interconnected. However, the models of the basic computing units produced for digital computer simulations allow one to synthesise programmes which solve these equations. Two methods are proposed for solving these sets of equations, namely, the error criterion and steepest descent methods. The first method relies on measurements of absolute errors while the second, being a 'hill climbing' technique, is based on gradient measurements of a performance index.[23,24,25] Justifications for using the two methods will be given on the grounds of system stability, the nature of the performance criteria, the amount of hardware required to realise each scheme, and the effects of large scale summations and scaling on the convergence and accuracy of a problem.

## 4.1 Conditions for Solving a Set of Linear Equations[26,27]

The linear equation problem can be written as a matrix equation.

$$A\underline{x} = \underline{b}$$

or

$$\sum_{k=1}^{n} a_{ik} x_k = b_i \qquad \text{---- (4.1.1)}$$

where /

where A is an order n matrix and $\underline{x}$ and $\underline{b}$ are n column vectors. The following points apply in a situation where the $a_{ij}$, $x_k$ and $b_i$ are real but many of the conditions carry over to a situation where these terms are complex.

(a) The above matrix equation can be solved for $\underline{x}$ if det(A) is not equal to zero so that $A^{-1}$ exists, then,

$$\underline{x} = A^{-1}\underline{b} \; . \qquad \text{---- (4.1.2)}$$

(b) There must be n equations in n unknowns.

(c) The coefficient matrix, A, must have a quadratic form which is positive definite, ie,

$$\underline{x}^T A\underline{x} > 0 \; . \qquad \text{---- (4.1.3)}$$

(d) The coefficients $a_{ij}$ and the constants $b_i$ must have modulii which are less than or equal to unity.

(e) The eigenvalues, $\lambda_i$, of the matrix A must have positive real parts.

(f) If A is not positive definite then the system of equations $A\underline{x} = \underline{b}$ can be converted to the system $H\underline{x} = \underline{c}$ where H is positive definite by multiplying both sides of the first equation by $A^T$, ie,

$$A^T A\underline{x} = A^T\underline{b} \qquad \text{---- (4.1.4)}$$

and

$$\underline{x} = A^{-1}\underline{b} = H^{-1}\underline{c} \qquad \text{---- (4.1.5)}$$

(g) If the vector, $\underline{x}$, is a function of time then at t = 0, $\underline{x}(0)$ is our first approximation to the required solution. Let $\underline{x}_{opt}$ be the required /

required solution, then we need methods which continuously adjust $\underline{x}(t)$ so that,

$$\lim_{t\to\infty} \underline{x}(t) = \underline{x}_{opt} \qquad \text{---- (4.1.6)}$$

and

$$\lim_{t\to\infty} \underline{e}(t) = \underline{0} \qquad \text{---- (4.1.7)}$$

where

$$\underline{e}(t) = A\underline{x}(t) - \underline{b} \qquad \text{---- (4.1.8)}$$

such that the $a_{ij}$ and $b_i$ are constants. All methods must converge to one $\underline{x}_{opt}$ no matter where $\underline{x}(0)$ is on the performance surface.

## 4.2 Error Criterion Method[27]

Here we take as our index of performance the absolute error measurement,

$$\underline{e}(t) = A\underline{x}(t) - \underline{b} \qquad \text{---- (4.2.1)}$$

The vector $\underline{e}(t)$ is taken as a measure of the velocity of $\underline{x}(t)$, hence,

$$\dot{\underline{x}}(t) = -K[\,A\underline{x}(t) - \underline{b}\,] \qquad \text{---- (4.2.2)}$$

where K is a gain term (>0) and the negative sign ensures that the system will adjust itself in the correct sense to decrease the error. Solving the above state equation we have,

$$\underline{x}(t) = e^{-KAt}[\,\underline{x}(0) - \underline{x}_{opt}] + \underline{x}_{opt} \qquad \text{---- (4.2.3)}$$

where $e^{-KAt}$ is an nxn transition matrix for the system. See APPENDIX 4A. From equation (4.2.3) we can see that,

$$\lim_{t\to\infty} \underline{x}(t) = \underline{x}_{opt} \qquad \text{---- (4.2.4)}$$

and /

and

$$\lim_{t \to 0} \underline{x}(t) = \underline{x}(0) \qquad \text{---- (4.2.5)}$$

The transition matrix can be evaluated using the Caley-Hamilton theorem and to ensure that

$$\lim_{t \to \infty} \exp(-KAt) = \mathbf{O} \qquad \text{---- (4.2.6)}$$

the eigenvalues of A must have positive real parts. We can be certain that the $\lambda_i$ do have positive real parts if we make the off-diagonal elements of A smaller than the diagonal ones, ie,

$$a_{ii} > a_{ij}\Big|_{i \neq j} \qquad \text{---- (4.2.7)}$$

## 4.3 Methods for Ensuring Stability[27]

(a) If the system of equations

$$A\underline{x} = \underline{b}$$

is unstable the following transformation can be made to ensure stability,

$$A^T A\underline{x} = A^T\underline{b}$$

The matrix $A^T A$ is a real, symmetric array whose eigenvalues have positive real parts. However, this method entails much calculation to produce the required transformation.

(b) Instead, a set of augmented equations can be solved:

$$\begin{bmatrix} I & A \\ A^T & O \end{bmatrix} \begin{bmatrix} \underline{\alpha} \\ \underline{x} \end{bmatrix} = \begin{bmatrix} \underline{b} \\ \underline{O} \end{bmatrix} \qquad \text{---- (4.3.1)}$$

where I is the identity matrix,
O is the null matrix, and
$\underline{\alpha}$, $\underline{x}$, $\underline{O}$, $\underline{b}$ are n vectors.

Twice /

Twice the number of equations have to be solved using this method but this system can be shown to be always stable. However the increase in hardware is not a great disadvantage in a stochastic computer.

## 4.4 Stochastic Computer Implementation of the Error Criterion Method

The system of equations to be implemented is:

$$x_i(t) = -K\left[\sum_{k=1}^{n} a_{ik}\, x_k(t) - b_i\right] \qquad \text{---- (4.4.1)}$$

In the general case, the following hardware is required:

Multiplication

$n^2 + n$ multipliers

Summation

$n(n + 1 - 1) = n^2$ summers

Depending on the value of (n+1), further multipliers will be required to adjust the values of the partial sums so that correct addition is achieved. See Section 1.

Invertors

If K is negative we require n invertors, but if K is positive 2n invertors are required.

Integration

n integrators

Comparators

| | |
|---|---|
| Coefficient matrix | $n^2$ |
| b vector | $n$ |
| Variable gain term, K | $n$ |
| Total | $n^2+2n$ |

The summation process will require $n^2$ independent sequences of probability 0.5.

A generalised flow diagram is given in Figure 4.4.

## 4.5 Hardware Savings

The stochastic computing units used are two input devices and all, except for the summer, depend on their inputs being statistically independent for correct operation, ie, there must be no cross-correlation between inputs. The error criterion circuit immediately branches into n limbs terminating at integrators which have the property of stochastic isolation by virtue of their internal random number comparison. Coefficients in different 'limbs' having the same value, may be fed from the same comparator. The number of noise lines to the summers may be similarly reduced. If K is chosen to be 1.0 then n multipliers and n summers can be saved by using the integrators as two input summing integrators. However 2n extra invertors will be required, but any summation will be greater by a factor of two and the resolution of smaller errors will be possible.

## 4.6 Speed of Computation

On a conventional analogue computer the speed of calculation depends on the integrator gains, K and the eigenvalues of A. The speed of an equivalent stochastic computer circuit depends on the above constants as well as the number of summations to be performed. This is due to the inherent attenuation in the summation process. Like the digital computer, the stochastic machine will take longer to calculate a result as the number of equations to be solved increases. Both machines will take longer than the conventional analogue computer.

4.7 /

## 4.7 System Resolution

The stochastic computer uses M bit counters to store variables. Each counter has $2^M$ possible states and using a bipolar mapping one machine unit is equivalent to $2^{M-1}$ states. Thus there is a limit to the resolution of the system. As M increases, variables whose mean magnitudes are smaller than $2^{1-M}$ have less and less significance. This limit on the resolution of a variable has to be taken into account when large numbers of random sequences are to be added since the summation process rapidly converges to zero (0.5 probability).

## 4.8 Scaling

Since the stochastic computer is essentially a digital mechanisation of an analogue computer - physical variables being represented by probabilities - problems are scaled in a similar manner to conventional machines. All problems have to be expressed in terms of normalised variables. Consider the following example:

Example

The unscaled system of linear equations is

$$\sum_{k=1}^{n} a_{ik} x_k = b_i \qquad \text{---- (4.8.1)}$$

Divide throughout by the modulus of the largest coefficient in each row, hence,

$$\sum_{k=1}^{n} \frac{a_{ik}}{|a_{ik}|_{max}} x_k = \frac{b_i}{|a_{ik}|_{max}} \qquad \text{---- (4.8.2)}$$

Normalise the $x_i$ to a base which is either the largest of them or some largest possible expected value. Then,

$$\sum_{k=1}^{n} \frac{a_{ik}}{|a_{ik}|_{max}} \frac{x_k}{|x_k|_{max}} = \frac{b_i}{|a_{ik}|_{max} |x_k|_{max}} \qquad \text{---- (4.8.3)}$$

$$\Rightarrow \sum_{k=1}^{n} \alpha_{ik_N} x_k = \beta_i \qquad \text{---- (4.8.4)}$$

where /

where $|\alpha_{ik}|, |x_i|, |\beta_i| \leq 1.0$

This is only one possible method of scaling, but any method should ensure that the diagonal elements are greater than the off-diagonal ones so that:

(a) the eigenvalues of the matrix $\{\alpha_{ij}\}$ have positive real parts;

(b) to ensure numerical stability since small errors in the diagonal elements with small modulii can mean large variations in the output, ${}_N\underline{x}$.

## 4.9 Numerical Example

A FORTRAN IV programme was devised to simulate a stochastic computer solving a set of linear equations for the case n = 3, K = 1.0. The programme is listed in APPENDIX 4B and a flowchart is presented in Figure 4.9. The unscaled problem is:

$$\begin{bmatrix} 2.0 & 1.0 & 1.0 \\ 1.0 & 2.0 & 1.0 \\ 1.0 & 1.0 & 2.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix} \qquad \text{---- (4.9.1)}$$

with solution

$$\underline{x}_{opt} = \begin{bmatrix} -0.5 \\ 0.5 \\ 1.5 \end{bmatrix} \qquad \text{---- (4.9.2)}$$

Normalising the $x_i$ to a base of 2.0, then the scaled problem is,

$$\begin{bmatrix} 1.0 & 0.5 & 0.5 \\ 0.5 & 1.0 & 0.5 \\ 0.5 & 0.5 & 1.0 \end{bmatrix} \begin{bmatrix} {}_N x_1 \\ {}_N x_2 \\ {}_N x_3 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.5 \\ 0.75 \end{bmatrix} \qquad \text{---- (4.9.3)}$$

In /

In the programme all coefficients and variables are represented by integer values so that the problem is expressed in the following way:

$$\begin{bmatrix} 2048 & 1024 & 1024 \\ 1024 & 2048 & 1024 \\ 1024 & 1024 & 2048 \end{bmatrix} \begin{bmatrix} {}_{N}x_1 \\ {}_{N}x_2 \\ {}_{N}x_3 \end{bmatrix} = \begin{bmatrix} 512 \\ 1024 \\ 1536 \end{bmatrix} \quad \text{---- (4.9.4)}$$

with solution

$$ {}_{N}\underline{x}_{opt} = \begin{bmatrix} -512 \\ 512 \\ 1536 \end{bmatrix} \quad \text{---- (4.9.5)}$$

where the variables are expressed in the range (-2048,2048).

## 4.10 Results

The results from the simulation were plotted against time. See Figures 4.10(a)-4.10(c). The following points were observed about the behaviour of the system.

(a) The trajectories show the deterministic behaviour of an analogue computer solution, but with superimposed random variance.

(b) The curves show evidence of 'hunting' around the optimum operating point so that the variances of the outputs are greater than expected.

(c) In the steady state the trajectories tend to lie to one side of optimum showing that the circuit cannot estimate extremely small errors.

(d) The trajectories are independent of each other as predicted by the mathematical model of the circuit. This indicates that one can relate stochastic computer circuits to some simple systems without recourse to stochastic automata theory.

(e) /

(e)

| Variable | Theoretical Value (states) | Mean Simulation Value (states | Error (states) |
|---|---|---|---|
| $x_1$ | - 512 | - 460 | +52 |
| $x_2$ | 512 | 450 | -62 |
| $x_3$ | 1536 | 1480 | -56 |

Figure 4.10(d)

| Variable | Maximum Variation at steady state (states) | Standard Deviation at steady state (states) |
|---|---|---|
| $x_1$ | ±50 | 31 |
| $x_2$ | ±50 | 31 |
| $x_3$ | ±80 | 21 |

Figure 4.10(e)

The error between theoretical and experimental values is under 3%. The effects of 'hunting' observed in the trajectories do not significantly alter the variances on the output sequences from the integrators.

(f) The same problem was run on an analogue computer and the results are displayed in Figures 4.10(f) - 4.10(h), and they show similar behaviour to the stochastic simulation.

## 4.11 Method of Steepest Descent[26,27,28]

The method of steepest descent measures the steepest slope of the criterion function and adjusts the system in that direction so that the fastest possible optimisation times are obtained. The parameter velocities are made equal to the criterion slopes in the corresponding directions. This is a powerful method which allows the optimisation of functions which have optimum operating points that are not null points.

## 4.12 Solution of Linear Equations by the Method of Steepest Descent

The performance criterion used is a scalar function of the error vector,

$$\underline{e}(t) = A\underline{x}(t) - \underline{b} \qquad \text{---- (4.12.1)}$$

The scalar function used is

$$f(\underline{x}(t)) = \tfrac{1}{2}\, \underline{e}(t)\, \underline{e}(t) \qquad \text{---- (4.12.2)}$$

Equation (4.12.2) has a minimum value of zero when

$$\underline{x}(t) = \underline{x}_{opt} \qquad \text{---- (4.12.3)}$$

and

$$A\underline{x}_{opt} = \underline{b} \qquad \text{---- (4.12.4)}$$

For all other $\underline{x}(t)$,

$$f(\underline{x}(t)) > 0 \qquad \text{---- (4.12.5)}$$

By making the $x_j(t)$ proportional to the partial derivative $-\partial f/\partial x_j$ convergence to the required solution is guaranteed. For a proof of this see APPENDIX 4B. Hence,

$$\dot{x}_j(t) = -K \sum_{i=1}^{n} a_{ij} \{ \sum_{k=1}^{n} a_{ik}\, x_k(t) - b_i \} \qquad \text{---- (4.12.6)}$$

where the $a_{ij}$ and $b_i$ are constants and K is a variable gain term. Equation (4.12.6) can be written in the form

$$\dot{\underline{x}}(t) = -KA^T [\, A\underline{x}(t) - \underline{b}\, ] \qquad \text{---- (4.12.7)}$$

This system always has eigenvalues which are stable and the above equation has to be implemented on the stochastic computer. In the same way as before we can /

can solve equation (4.12.7) to find $\underline{x}(t)$. Hence,

$$\underline{x}(t) = e^{-KA^{T}At} (\underline{x}(0) - \underline{x}_{opt}) + \underline{x}_{opt} \qquad \text{---- (4.12.8)}$$

Cayley-Hamilton theorem can be used to evaluate the transition matrix $\exp(-KA^{T}At)$.

## 4.13 Stochastic Computer Implementation of the Steepest Descent Method

The system of equations to be implemented is

$$\dot{x}_j(t) = -K \sum_{i=1}^{n} a_{ij} \{ \sum_{k=1}^{n} a_{ik} x_k - b_i \}$$

In the general case, the following hardware is required.

Multiplication

$n(2n+1)$ multipliers

Summation

$n(n+1-1) + n(n-1-1) = 2n(n-1)$ summers

The number of compensating multipliers required will depend on the sizes of the summations to be performed. See Chapter 1.

Invertors

2n invertors

Delays

One and two bit delays may be required at the output of the integrator to prevent any possibility of cross-correlation between elements of the error vector $\underline{e}(t)$. Hence we need 2n delays.

Integrators /

Integrators

n integrators

Comparators

| | |
|---|---|
| Coefficient matrix | $n^2$ |
| Transpose of coefficient matrix | $n^2$ |
| $\underline{b}$ vector | n |
| variable gain term | n |
| Total | $2n^2+2n$ comparators |

Other comparators are required for compensating multipliers in the summing arrays. These have to provide sequences with a generating probability of 0.75.

The summation process will require

$$n^2 + n(n-2) = 2n(n-1)$$

independent sequences of generating probability 0.5.

A generalised flow diagram is given which shows the circuit layout. See Figures 4.13(a) and 4.13(b).

## 4.14 Hardware Savings

Similar savings to those of the first method can be made in this circuit where it branches into limbs terminating at integrators which give stochastic isolation. In fact, the same error vector sequences can be used in the final matrix multiplication, ie,

$$\sum_{i=1}^{n} a_{ij}\, e_i, \qquad \text{---- (4.14.1)}$$

without any danger of cross-correlation between the $x_j$. The above equipment list is based on this assumption.

4.15 /

## 4.15 Speed of Computation

Again as n increases, the stochastic computer circuit will become slower than the conventional analogue machine due to the attenuation factor introduced by the summation process decreasing the system gain.

## 4.16 System Resolution and Scaling

The same principles apply in this case as for the error criterion method.

## 4.17 Numerical Example

The example presented is not a good one from the point of view of stability and suitable eigenvalues since the problem is just stable, but it shows how the precision of the UP/DOWN counters can avoid rounding errors in setting up the coefficients. Although K is chosen to be 1.0 the summation process introduces a further gain term of 0.125 which means that the estimated gradients are smaller than expected. The problem to be solved is:

$$\begin{bmatrix} 1.0 & -0.7143 & -1.0 \\ 1.0 & -0.5 & -0.75 \\ 0.7143 & 0.5714 & 0.5714 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -0.2286 \\ 0.0 \\ 1.0 \end{bmatrix} \quad \text{---- (4.17.1)}$$

where

$$\underline{x}_{opt} = \begin{bmatrix} 0.5996 \\ 0.60337 \\ 0.397 \end{bmatrix} \quad \text{---- (4.17.2)}$$

A programme which simulates the operation of the required circuit is listed in APPENDIX 4D. A flow diagram of the circuit is illustrated in Figures 4.17(a) and 4.17(b).

4.18 /

## 4.18 Results

The state trajectories are presented in Figures 4.18(a) - 4.18(c). Their graphs indicate good convergence of the system towards the predicted optimum solution. The results of the simulation are as follows:

$$ {}_{N}\underline{x} = \begin{bmatrix} 1160 \\ 1050 \\ 813 \end{bmatrix} \qquad \text{---- (4.18.1)} $$

and the theoretical solution is

$$ {}_{T}\underline{x} = \begin{bmatrix} 1228 \\ 1230 \\ 813 \end{bmatrix} \qquad \text{---- (4.18.2)} $$

The error between the theoretical and experimental values is:

$$ \underline{\varepsilon} = \begin{bmatrix} -68 \\ -180 \\ 0 \end{bmatrix} \qquad \text{---- (4.18.3)} $$

All trajectories show evidence of 'hunting', showing greater variances than would normally be expected at these probabilities. This drifting may be explained by the fact that the circuit cannot measure very small gradients so that it may drift out of control until the error is large enough to establish a controlling signal. If the clock frequency is 1MHz the solution time is approximately 60 msec, and the system time constant is 25 msec.

## 4.19 Matrix Inversion[26]

A problem often encountered in control engineering is to determine the inverse of a matrix. Let A be an order n matrix such that det(A) is not equal to zero so that its inverse exists.

Then, /

Then,

$$AY = I \qquad \text{---- (4.19.1)}$$

where I is the identity matrix.

Hence,

$$Y = A^{-1} \qquad \text{---- (4.19.2)}$$

Let

$$Y = [\underline{y}_1, \underline{y}_2, \underline{y}_3, \ldots, \underline{y}_j, \ldots, \underline{y}_n] \qquad \text{---- (4.19.3)}$$

and

$$I = [\underline{b}_1, \underline{b}_2, \underline{b}_3, \ldots, \underline{b}_j, \ldots, \underline{b}_n] \qquad \text{---- (4.19.4)}$$

where the $\underline{b}_j$ are the columns of I, the identity matrix if,

$$A[\underline{y}_1, \underline{y}_2, \ldots, \underline{y}_j, \ldots, \underline{y}_n] = [\underline{b}_1, \underline{b}_2, \underline{b}_3, \ldots, \underline{b}_j, \ldots, \underline{b}_n] \qquad \text{---- (4.19.5)}$$

Hence,

$$\begin{aligned} A\underline{y}_1 &= \underline{b}_1 \\ A\underline{y}_2 &= \underline{b}_2 \\ &\vdots \\ A\underline{y}_j &= \underline{b}_j \\ &\vdots \\ A\underline{y}_n &= \underline{b}_n \end{aligned} \qquad \text{---- (4.19.6)}$$

Thus the two circuits developed for solving linear equations can be used to determine the columns, $\underline{y}_j$, of the inverse of A. To determine $A^{-1}$ we have to solve n sets of linear equations in n unknowns n times so that computation times can become lengthy for large n. The error criterion and steepest descent methods can both be used to determine $A^{-1}$. However, to save time, the former method is used. Scaling is carried out in the same way as before. It is unlikely that the identity matrix can be used without it being scaled.

## 4.20 Numerical Example

The following system of equations has to be solved:

$$\begin{bmatrix} 1.0 & 0.5 & 0.5 \\ 0.5 & 1.0 & 0.5 \\ 0.5 & 0.5 & 1.0 \end{bmatrix} [\underline{Y}_1, \underline{Y}_2, \underline{Y}_3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

---- (4.20.1)

Suppose the maximum $y_{ij}$ is 2.0 then the scaled set of equations are:

$$\begin{bmatrix} 1.0 & 0.5 & 0.5 \\ 0.5 & 1.0 & 0.5 \\ 0.5 & 0.5 & 1.0 \end{bmatrix} {}_N Y = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} = {}_N I$$

---- (4.20.2)

The columns of $I_n$ are applied to the circuit in turn to yield the columns of ${}_N Y$.

## 4.21 Results

The inverse of A has nine elements and their trajectories are plotted in Figures 4.21(a) - 4.21(j). The three sets of equations are stable and approach the required values. All trajectories show evidence of 'hunting' over and above the normal random variance.

The required solution is

$${}_N A^{-1} = \begin{bmatrix} 0.75 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & 0.75 \end{bmatrix} \quad \text{---- (4.21.1)}$$

and ${}_N A^{-1}$ has to be denormalised to yield $A^{-1}$.

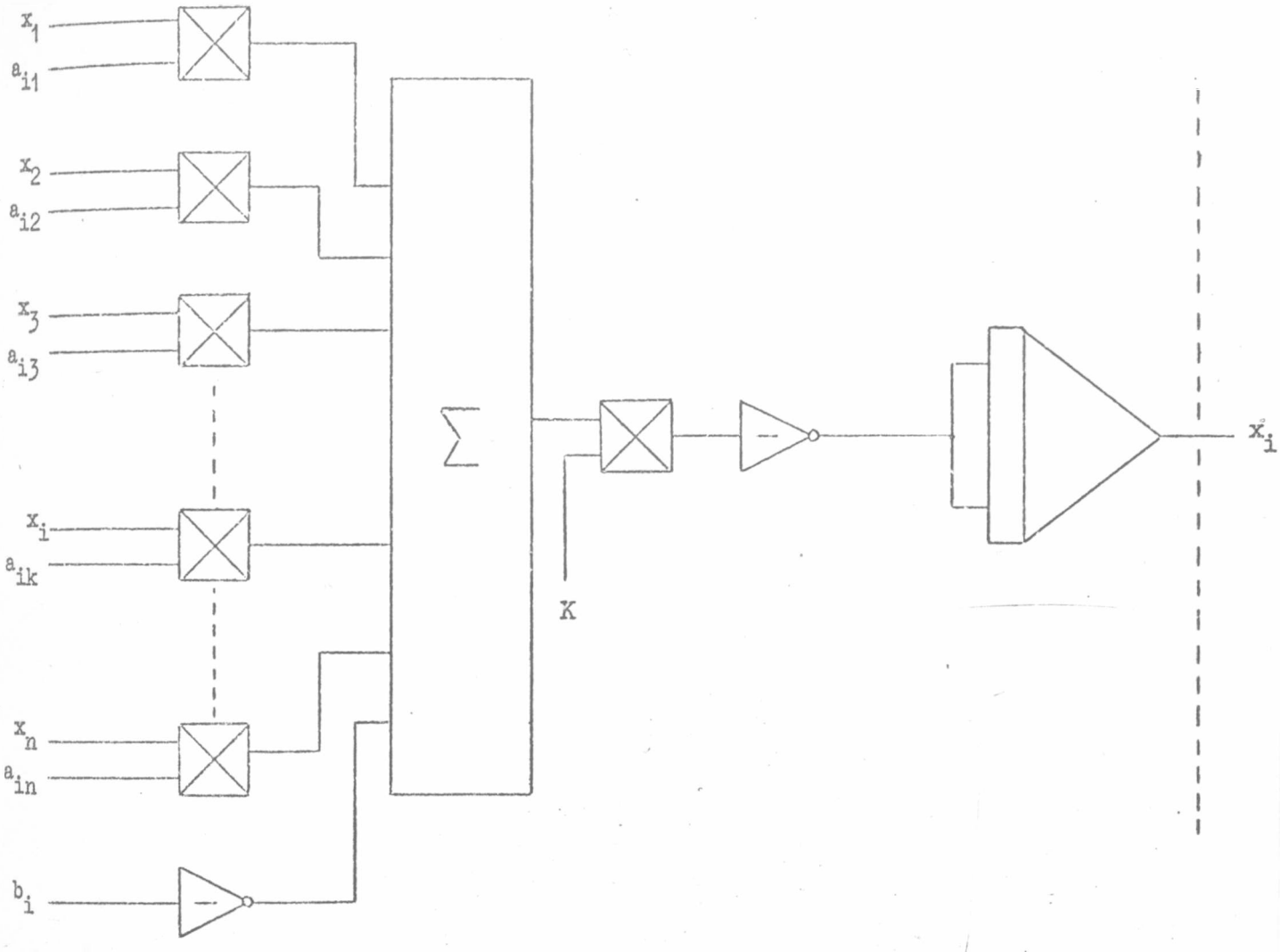
Steady state errors average 2.5% and this is again due to the inability of the network to estimate small errors. The counters employed in this circuit have a capacity of twelve bits.

4.22 /

## 4.22 Speed of Computation

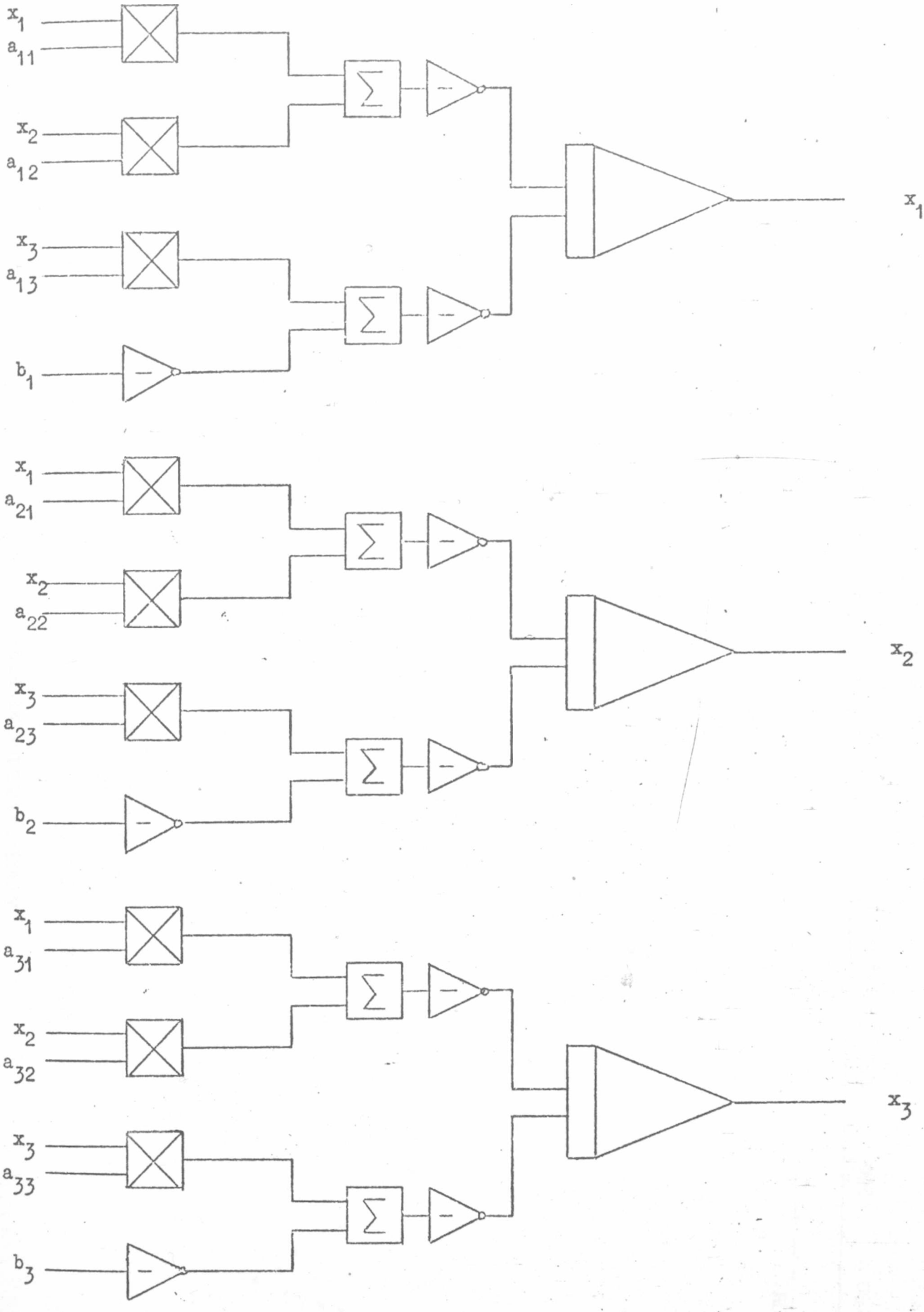
The digital computer programmes devised to simulate the operation of synchronous sequential circuits do not explicitly involve time. Instead they calculate the condition of a stochastic circuit at some clock pulse. In Chapter Three a technique was demonstrated for analysing the behaviour of a synchronous sequential circuit in terms of a normalised time domain.

Both the error criterion and steepest descent methods took about $6.75 \times 10^4$ clock pulses to attain steady state values. If the clock frequency had been 1MHz in each case, the solution time for both methods would have been 675 msec. A conventional digital computer solving these sets of equations by calculating the inverse of A and then performing a matrix multiplication, might take up to eight seconds to achieve the same results. Thus there is a great saving in computation time using a digital stochastic computer.




MECHANISATION OF THE ERROR CRITERION METHOD

FIG 4.4



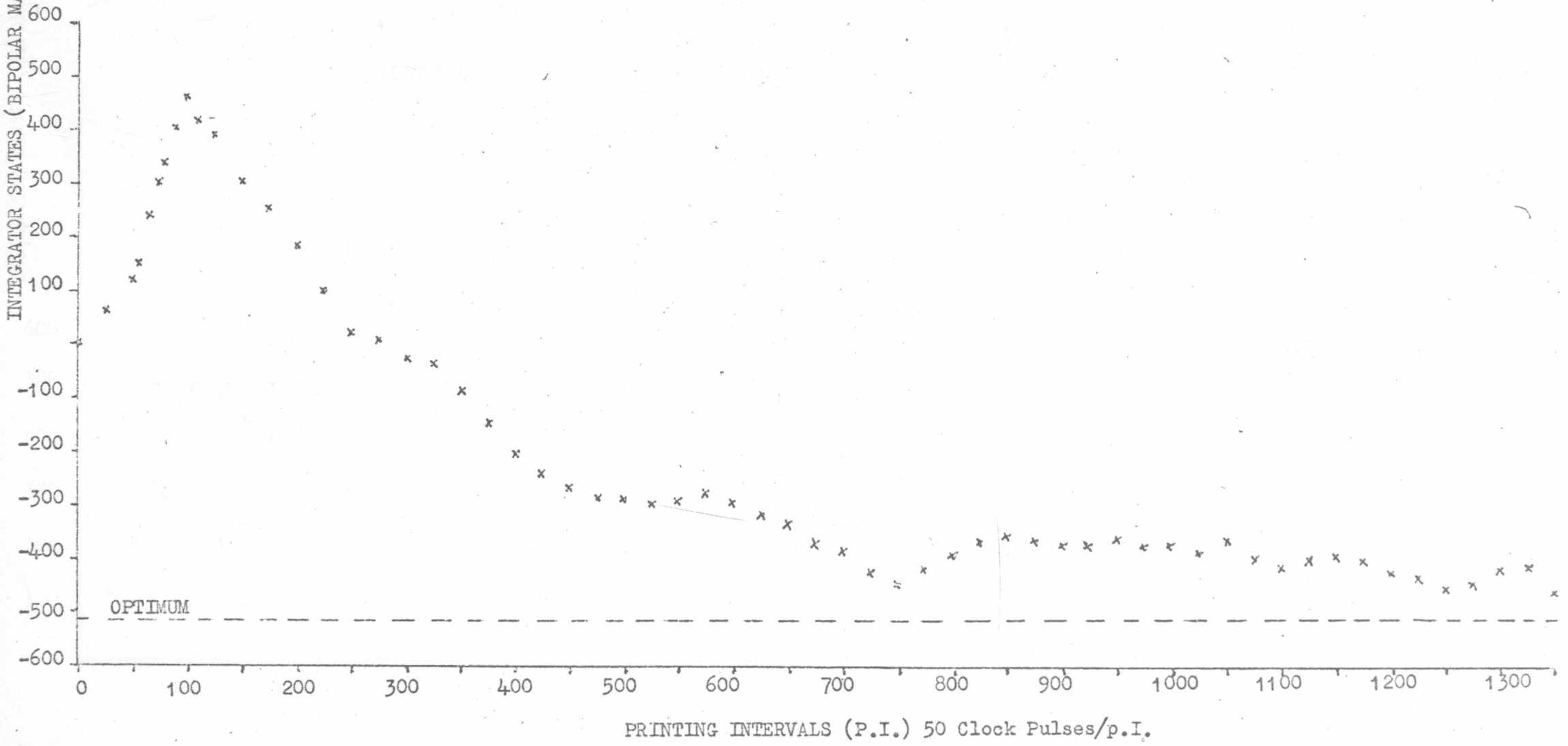

ERROR CRITERION METHOD FOR THE CASE n = 3

FIG 4.9

ERROR CRITERION METHOD: $x_1$

SIMULATION OF THE STOCHASTIC COMPUTER CIRCUIT

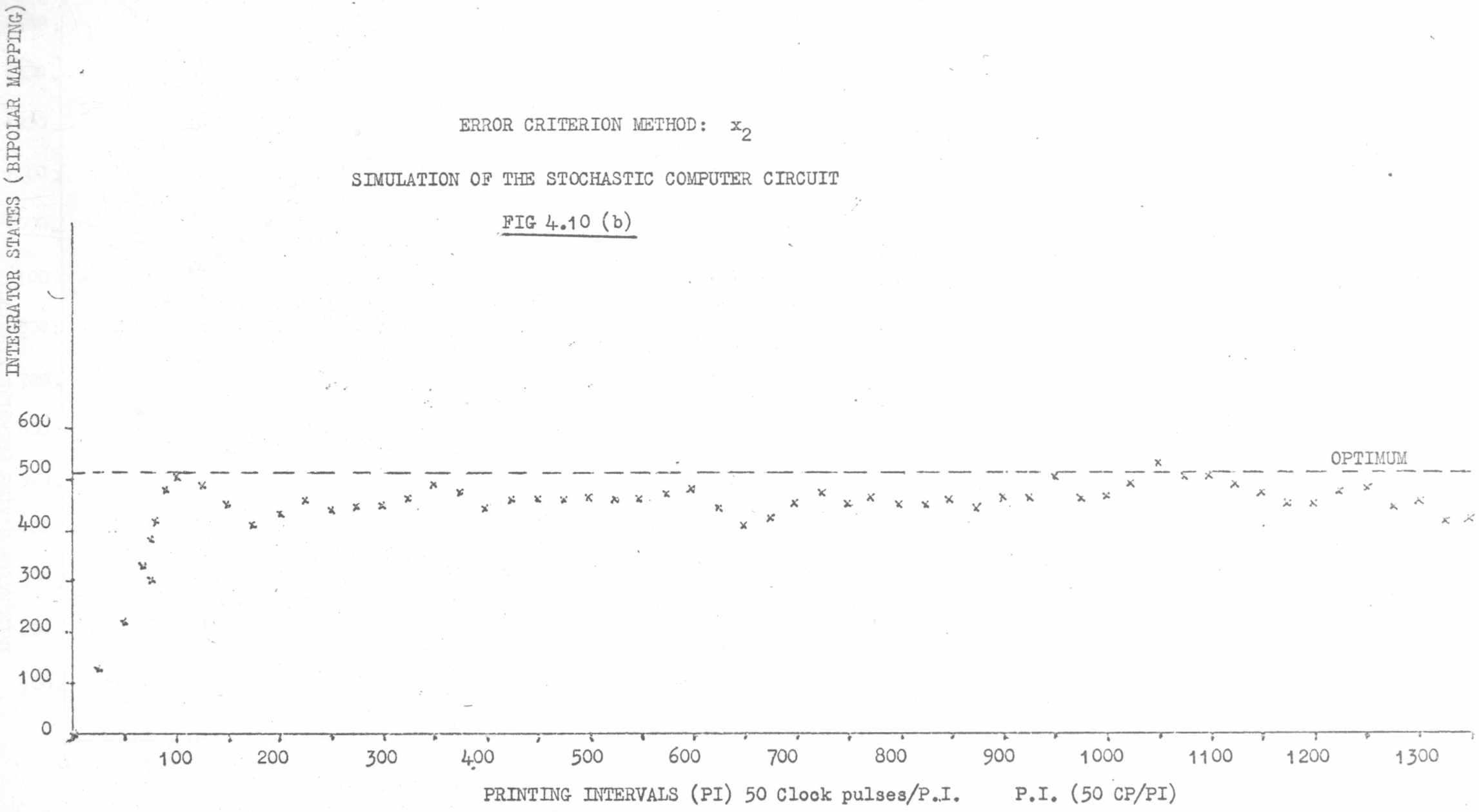
FIG 4.10 (a)

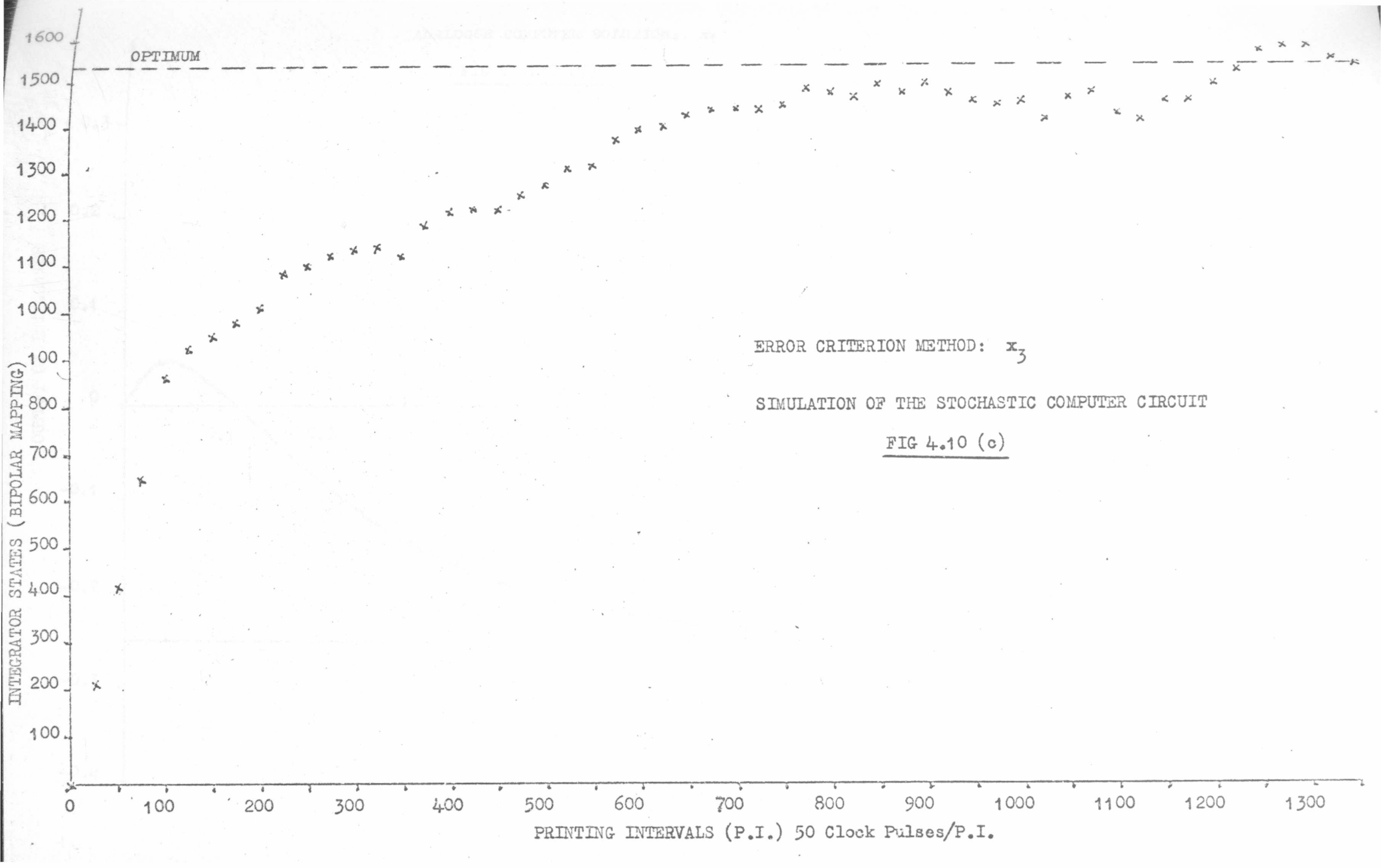


ERROR CRITERION METHOD: $x_2$

SIMULATION OF THE STOCHASTIC COMPUTER CIRCUIT

FIG 4.10 (b)





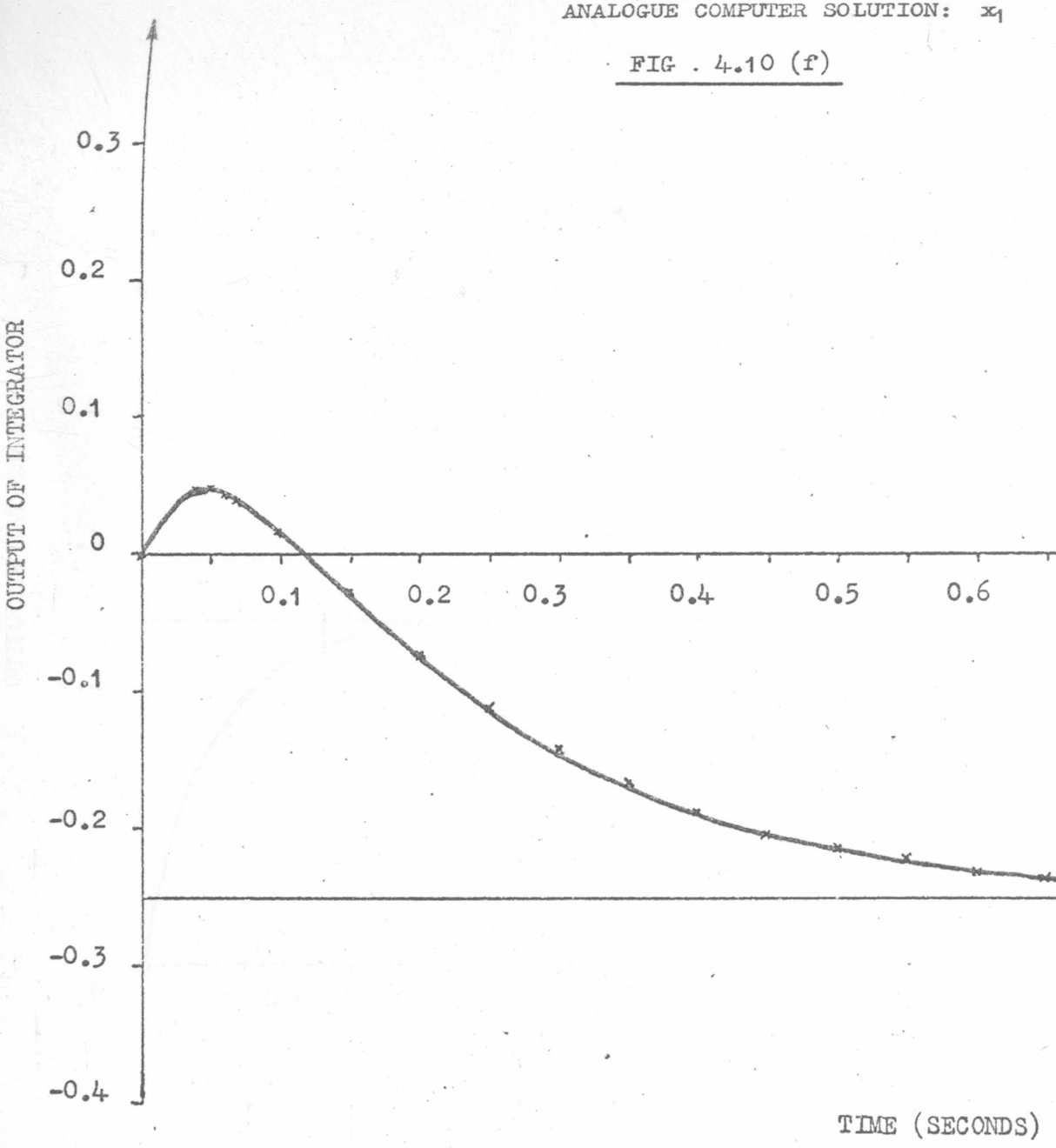

ERROR CRITERION METHOD: $x_3$

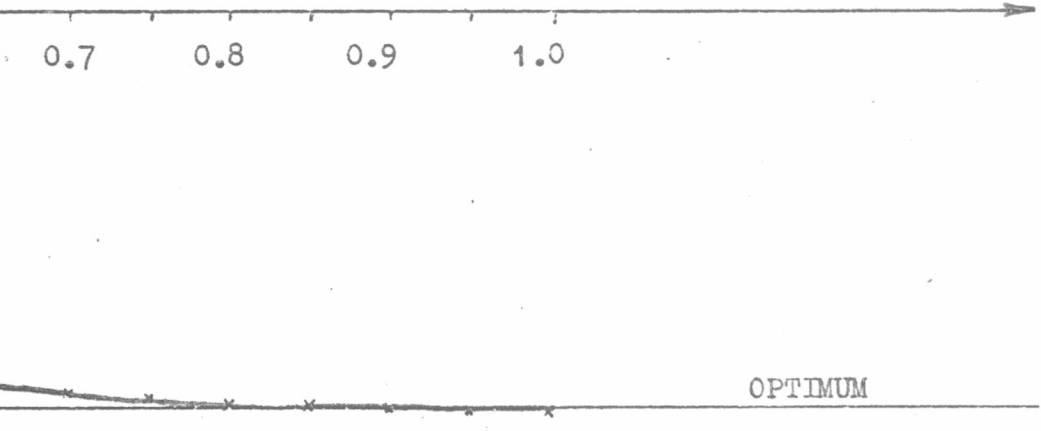
SIMULATION OF THE STOCHASTIC COMPUTER CIRCUIT

FIG 4.10 (c)

ANALOGUE COMPUTER SOLUTION: $x_1$

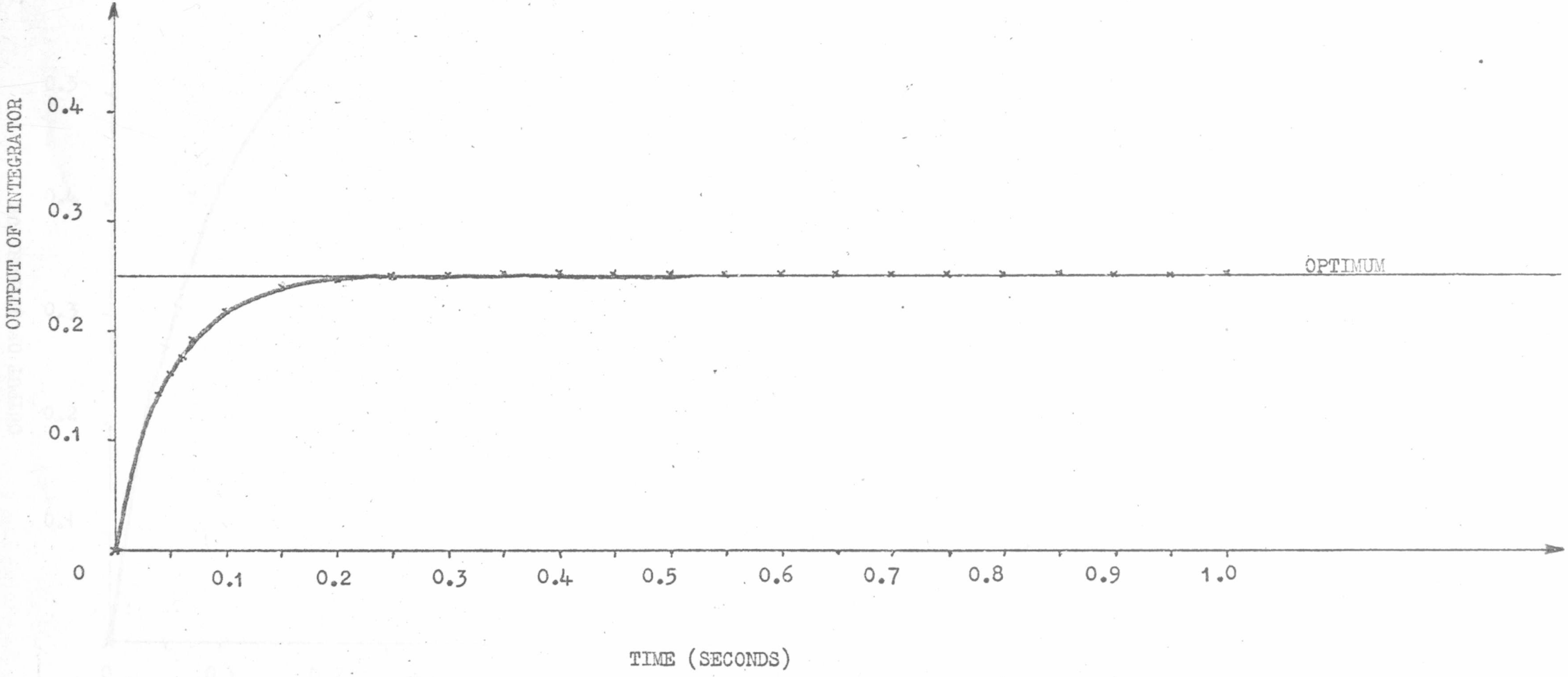
FIG . 4.10 (f)




0.7
0.8
0.9
1.0
OPTIMUM

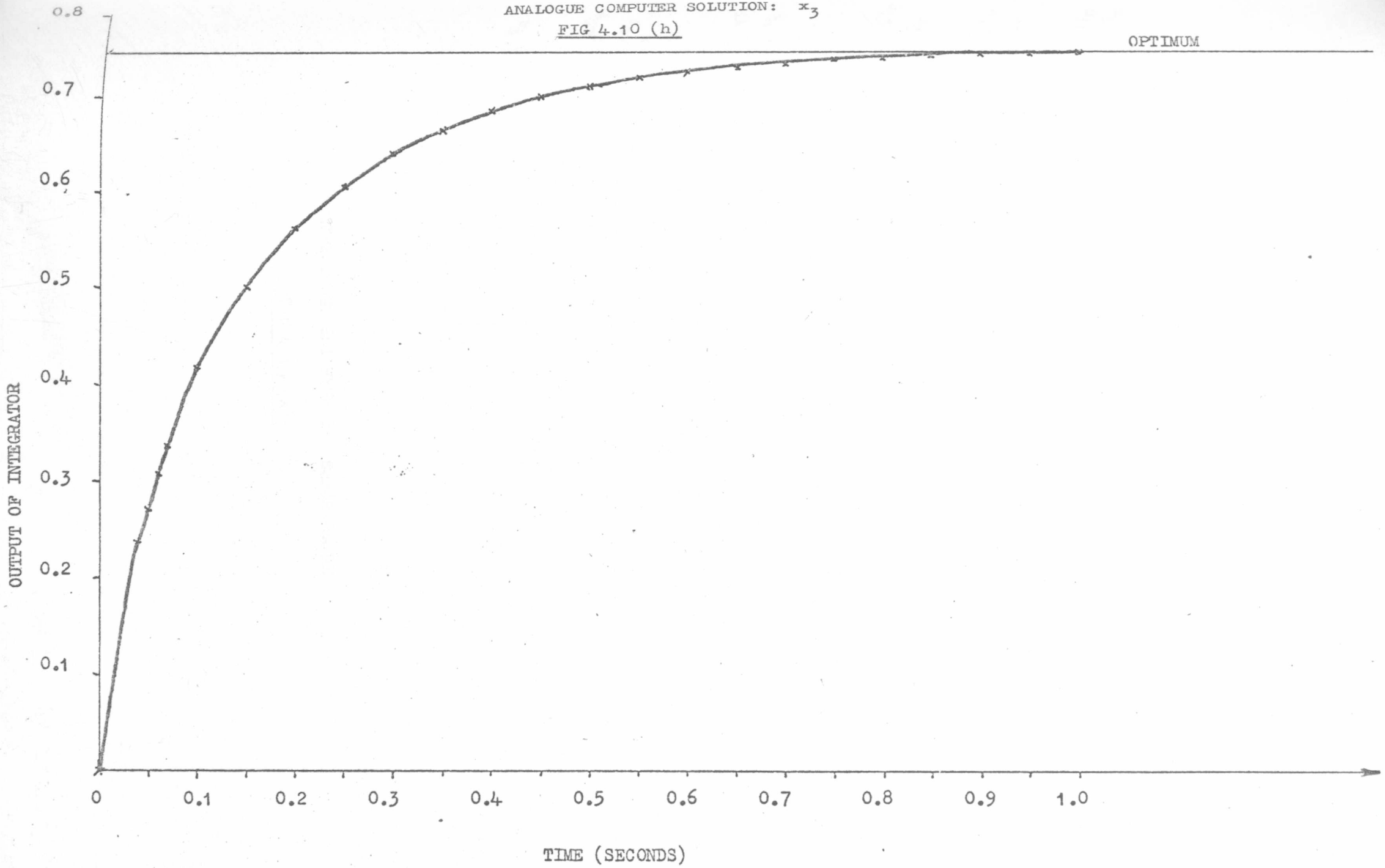
ANALOGUE COMPUTER SOLUTION: $x_2$

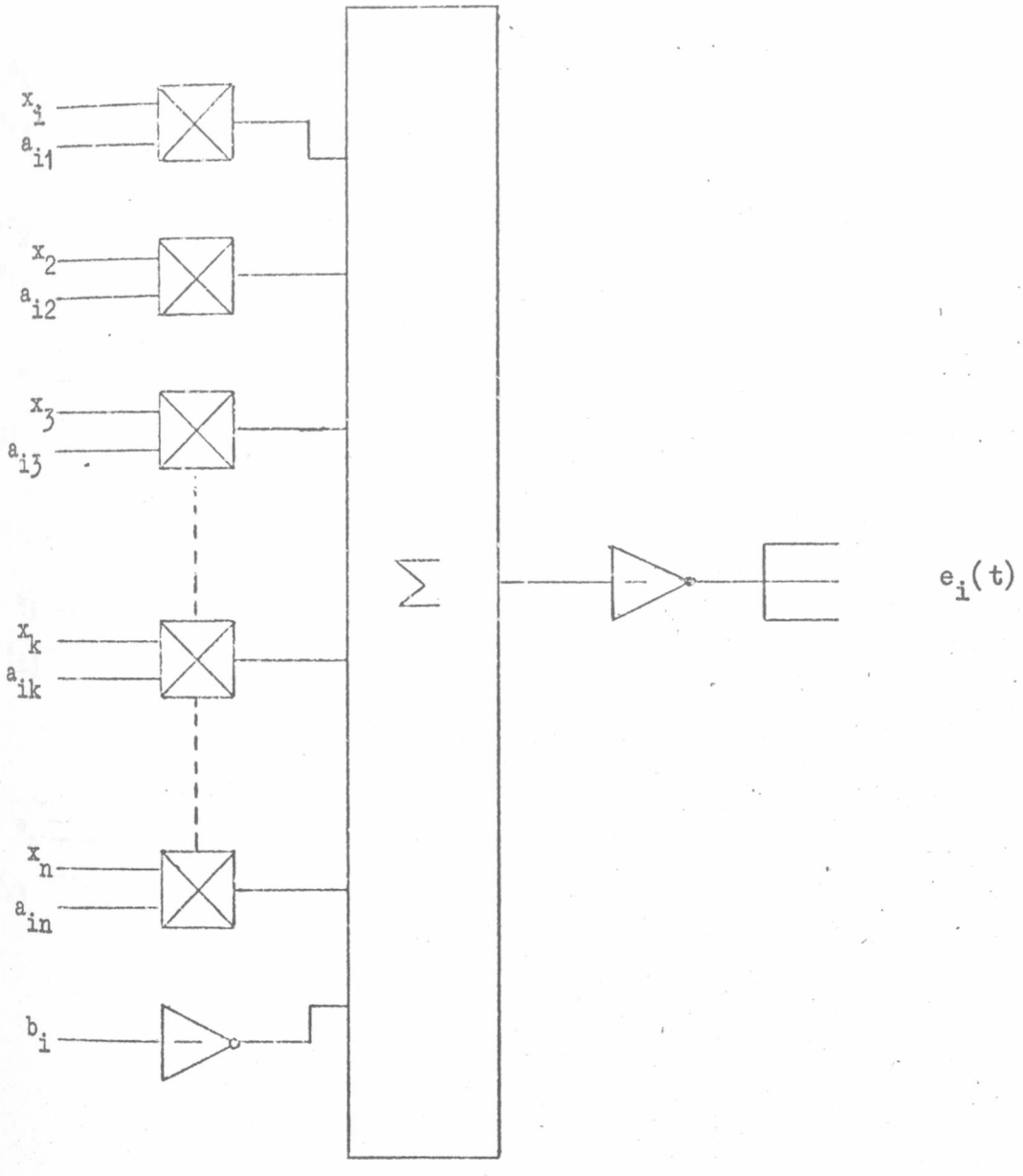
FIG 4.10 (g)



ANALOGUE COMPUTER SOLUTION: $x_3$

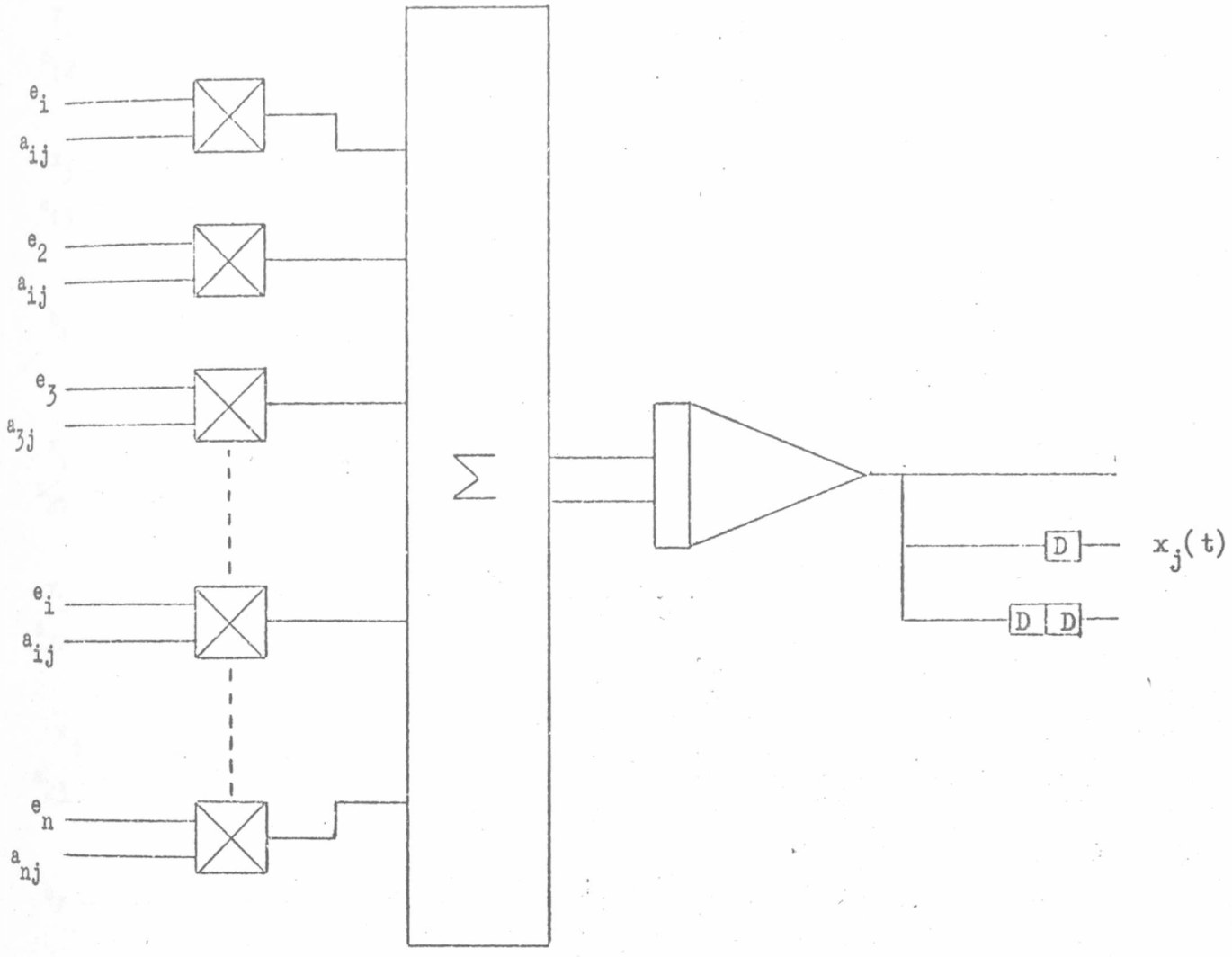
FIG 4.10 (h)





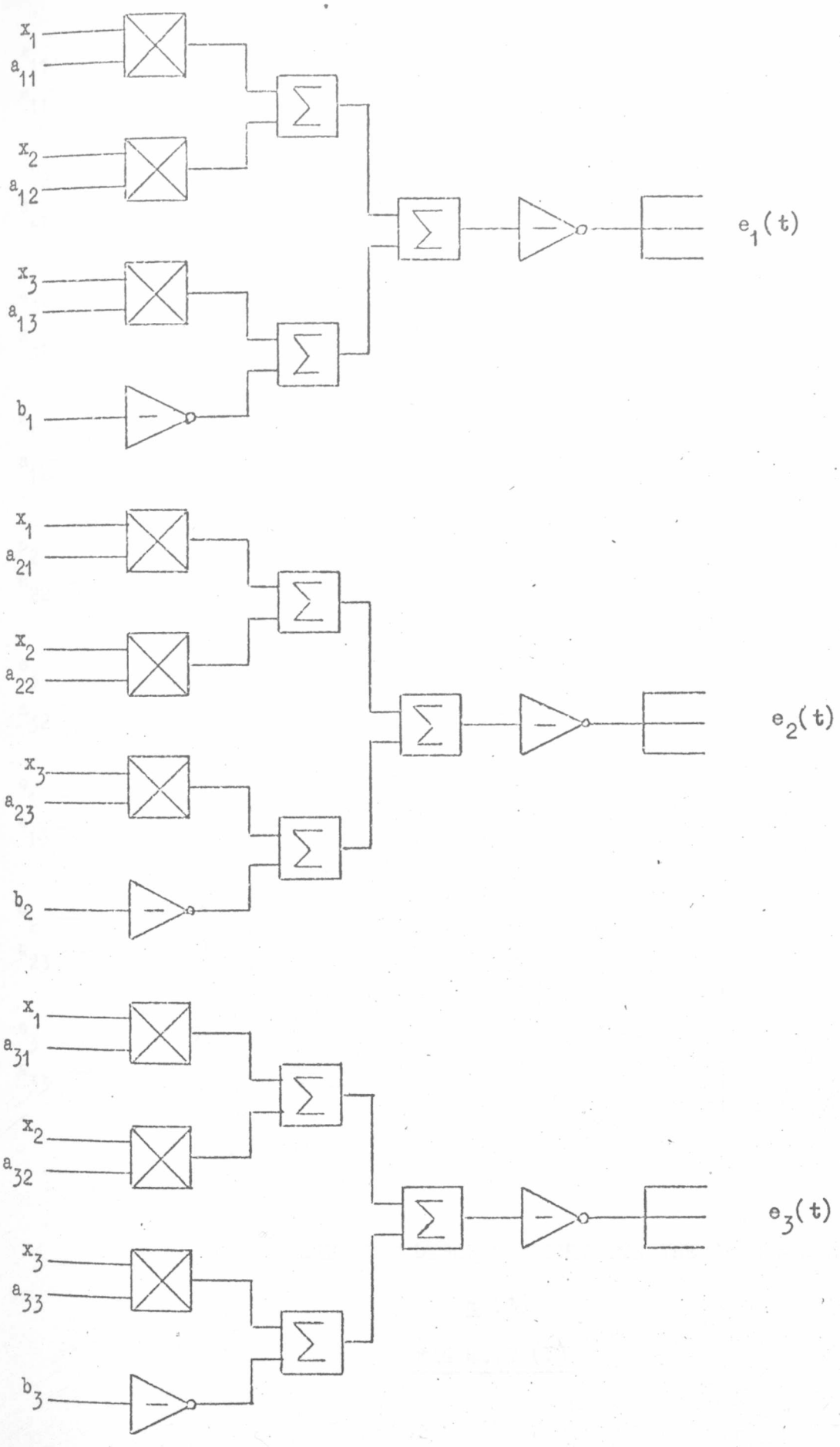

STEEPEST DESCENT METHOD: ERROR VECTOR

FIG 4.13 (a)




STEEPEST DESCENT METHOD: FORMATION of $\underline{x}$ (t)

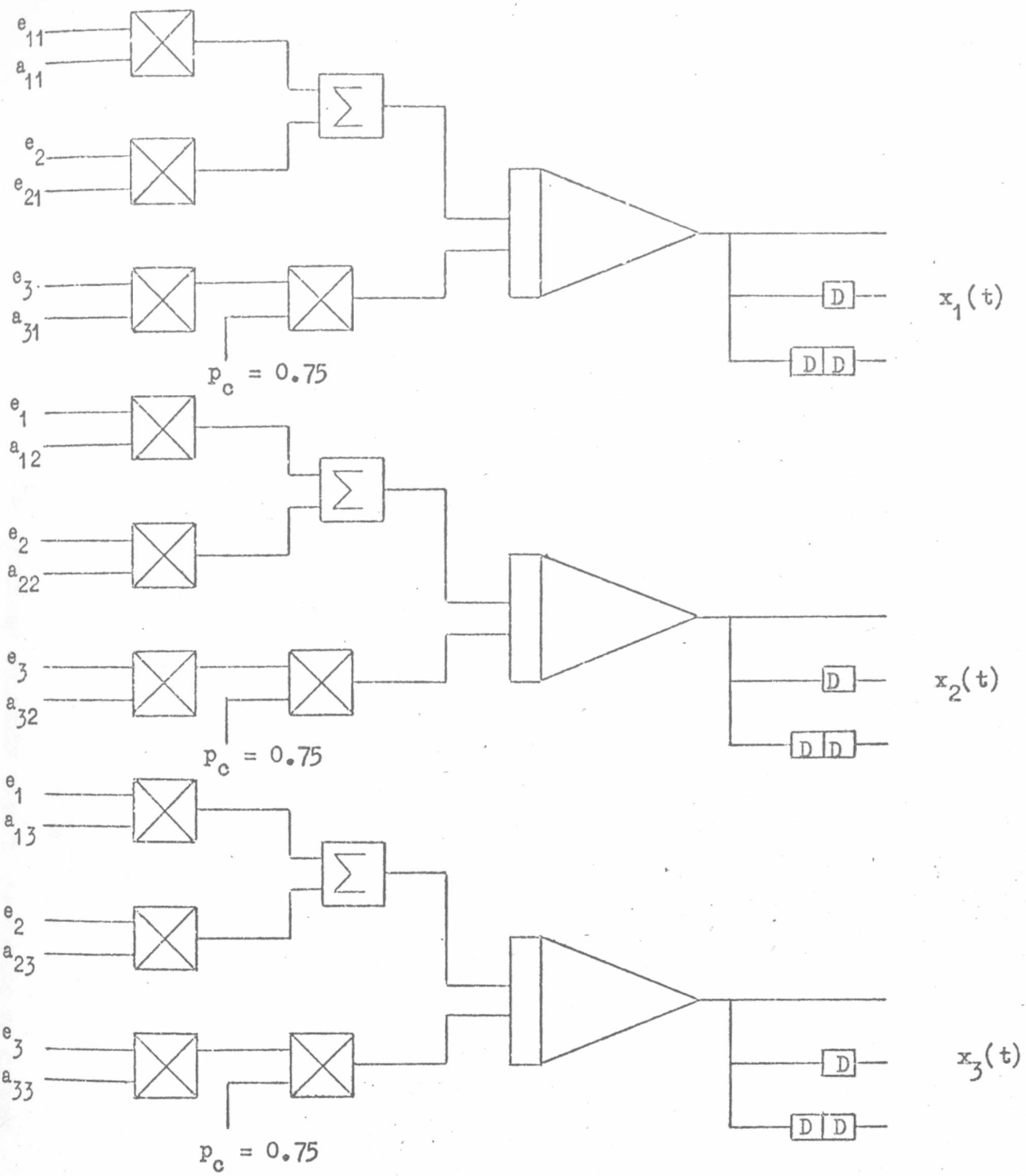
FIG 4.13 (b)




STEEPEST DESCENT METHOD FOR THE CASE n = 3

ERROR VECTOR

FIG 4.17 (a)



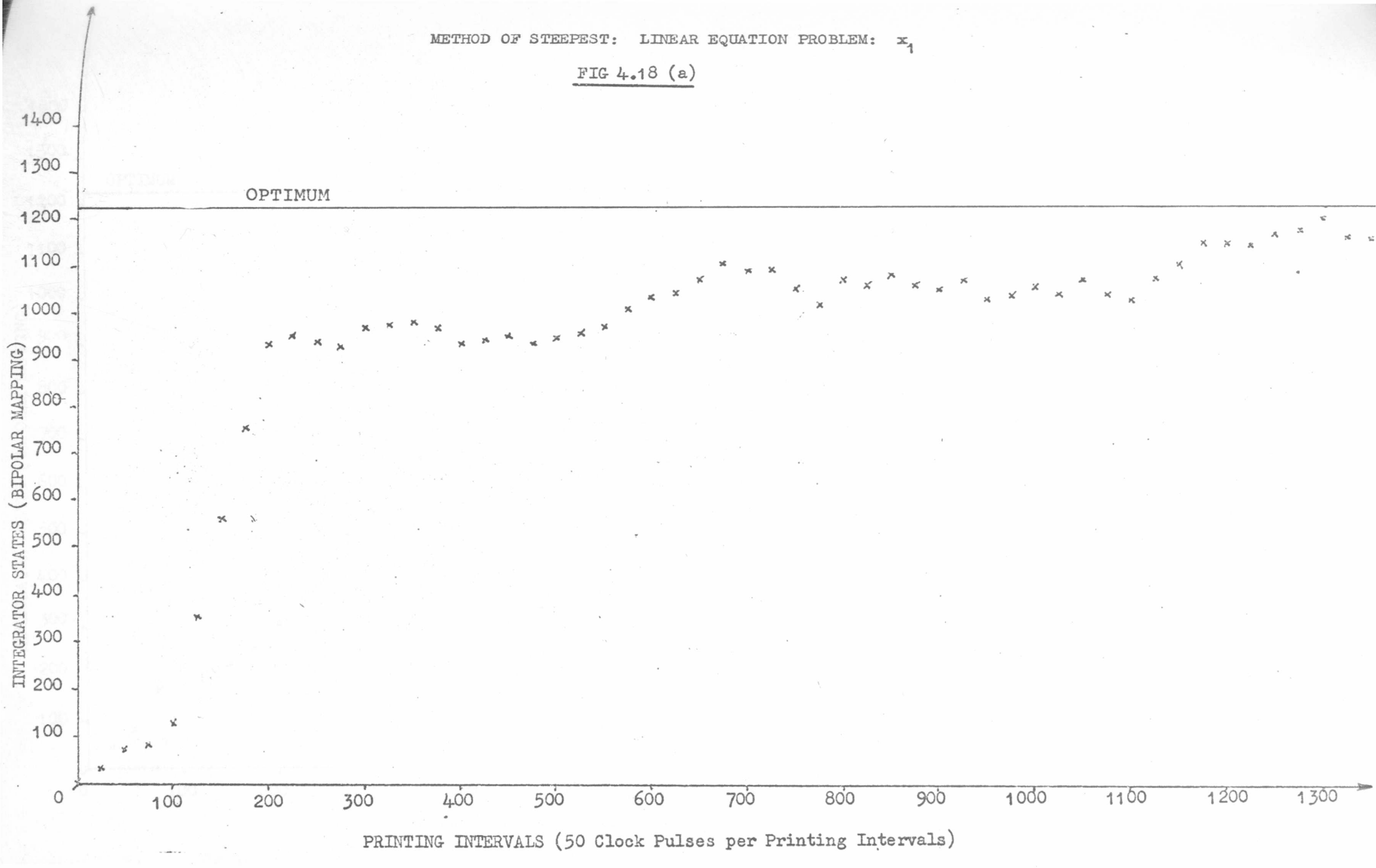

STEEPEST DESCENT METHOD FOR THE CASE n = 3

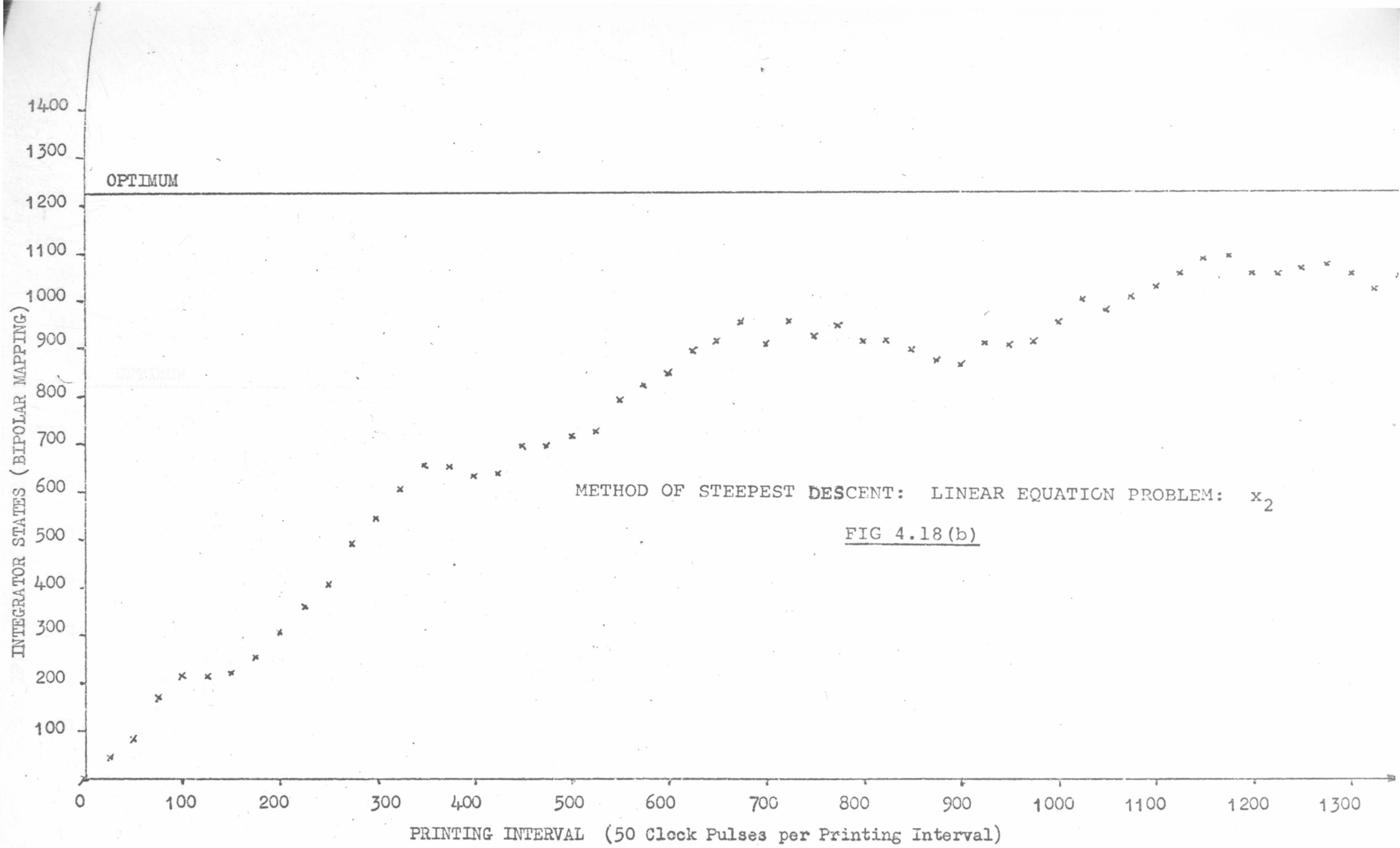
$\underline{x}(t)$

FIG 4.17 (b)

METHOD OF STEEPEST: LINEAR EQUATION PROBLEM: $x_1$

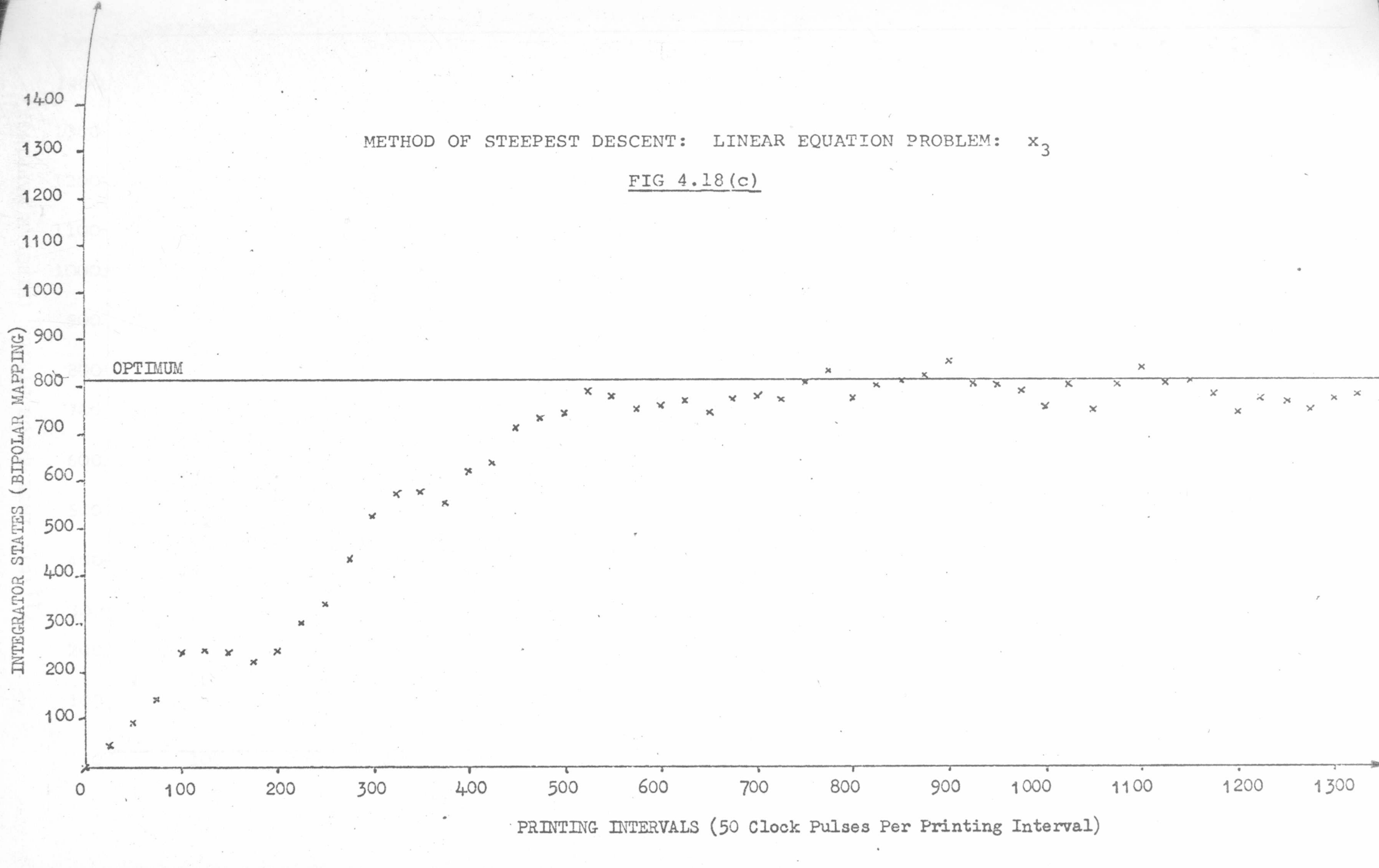
FIG 4.18 (a)

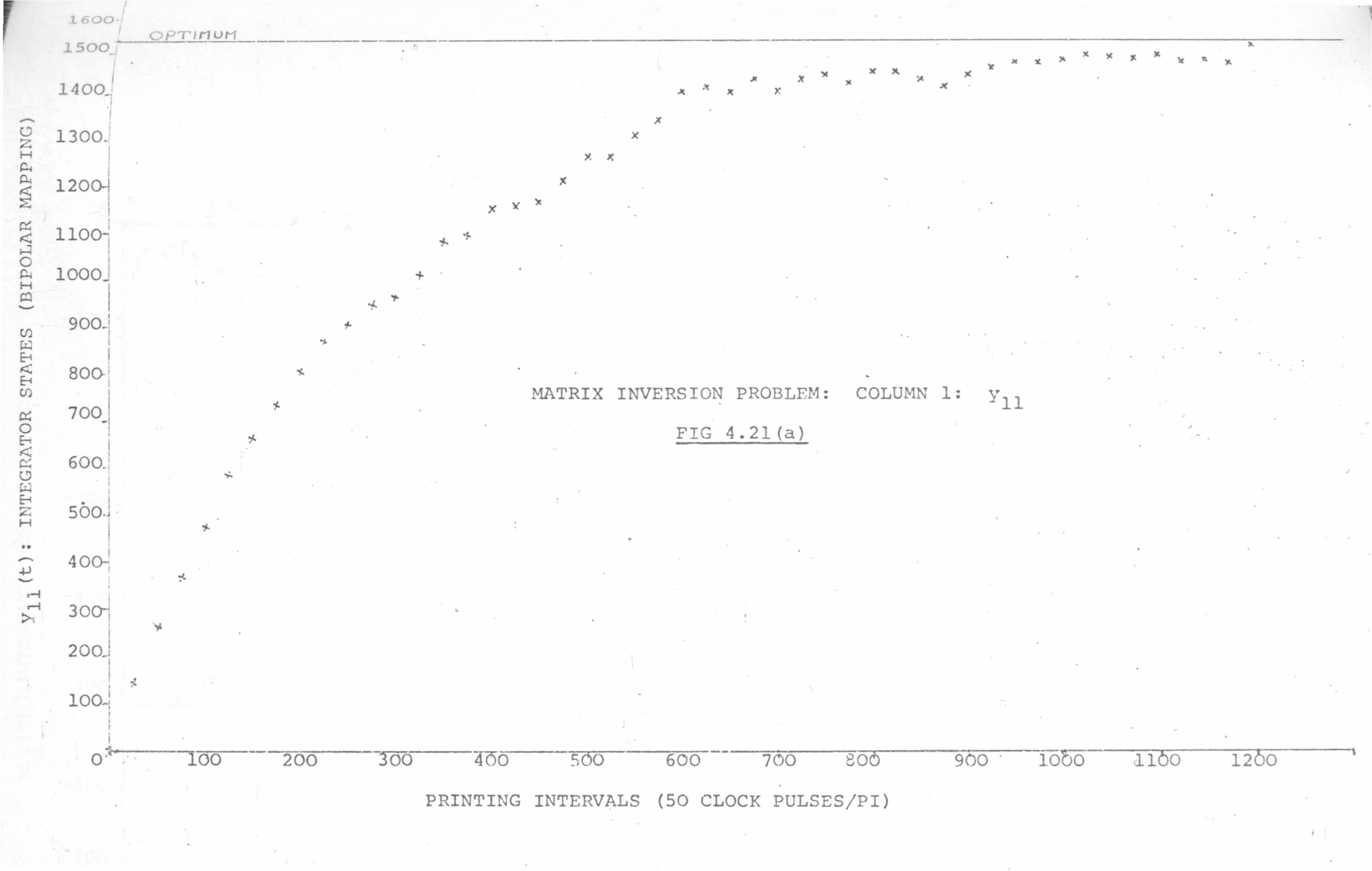





METHOD OF STEEPEST DESCENT: LINEAR EQUATION PROBLEM: $x_2$

FIG 4.18(b)


METHOD OF STEEPEST DESCENT: LINEAR EQUATION PROBLEM: $x_3$
FIG 4.18(c)
OPTIMUM
INTEGRATOR STATES (BIPOLAR MAPPING)
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
PRINTING INTERVALS (50 Clock Pulses Per Printing Interval)




MATRIX INVERSION PROBLEM: COLUMN 1: $y_{11}$

FIG 4.21(a)

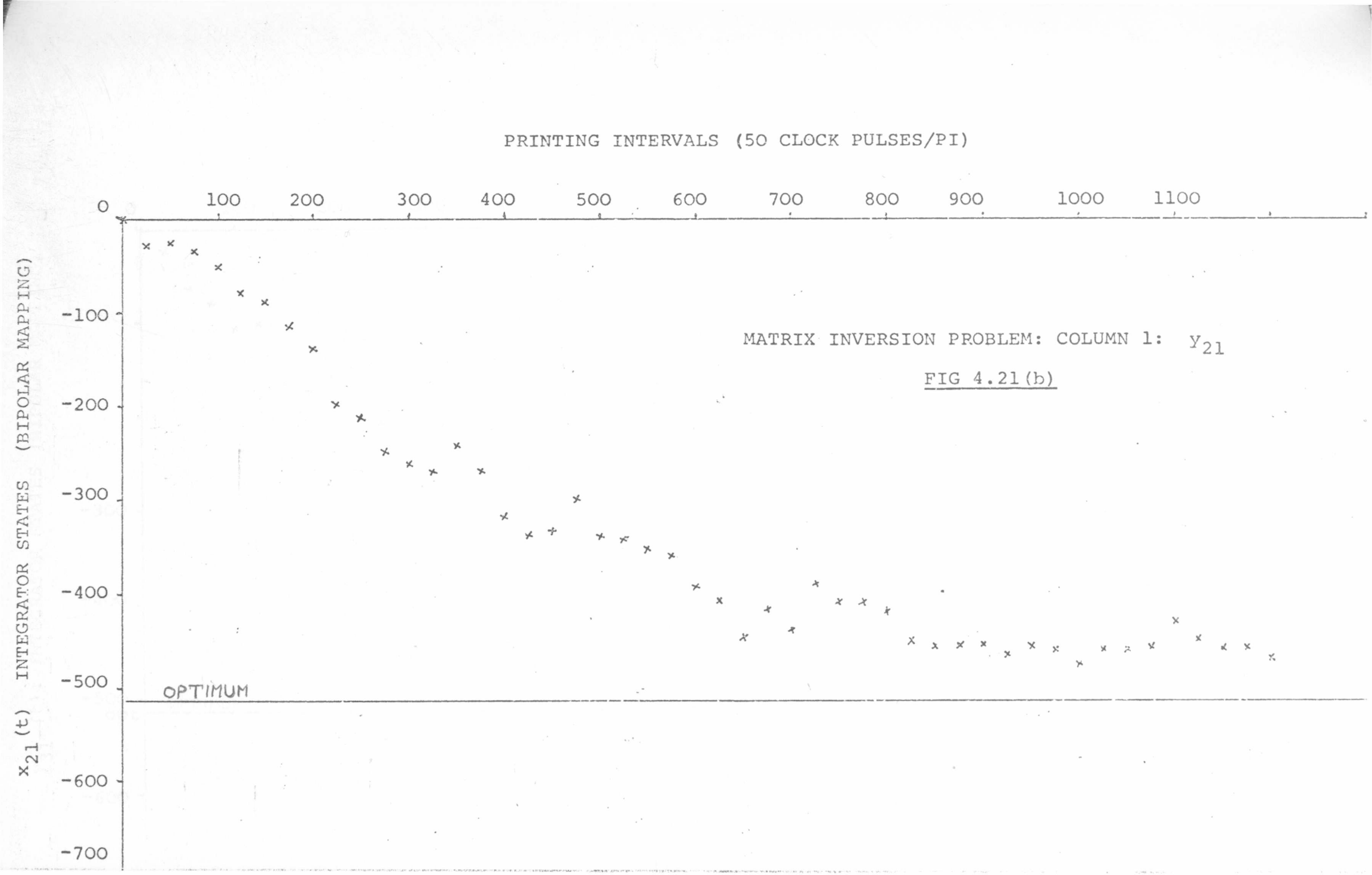

PRINTING INTERVALS (50 CLOCK PULSES/PI)
0
100
200
300
400
500
600
700
800
900
1000
1100
-100
-200
-300
-400
-500
-600
-700
$x_{21}(t)$ INTEGRATOR STATES (BIPOLAR MAPPING)
OPTIMUM
MATRIX INVERSION PROBLEM: COLUMN 1: $y_{21}$

FIG 4.21(b)

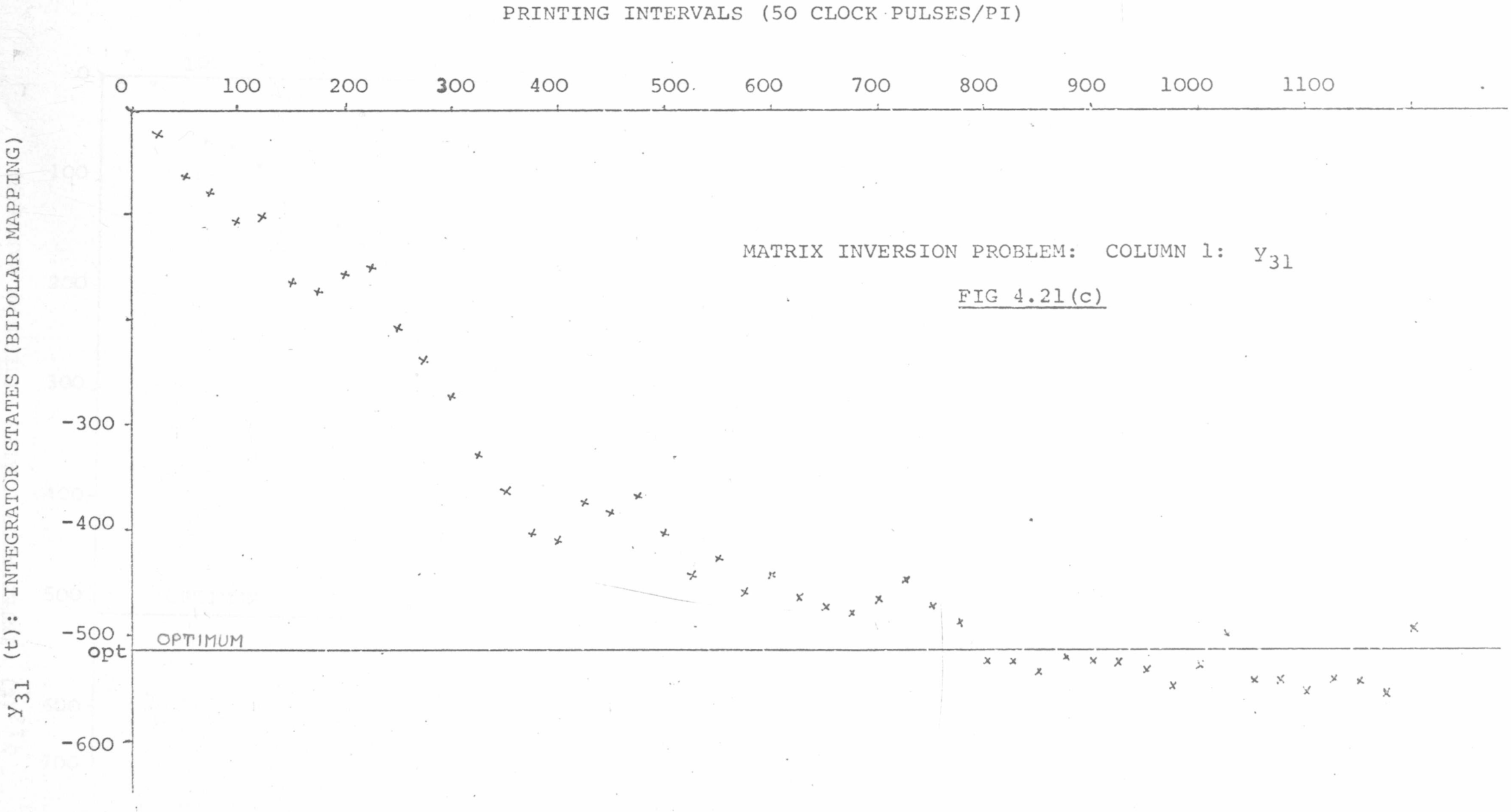

PRINTING INTERVALS (50 CLOCK PULSES/PI)
0
100
200
300
400
500
600
700
800
900
1000
1100
$y_{31}$ (t): INTEGRATOR STATES (BIPOLAR MAPPING)
-300
-400
-500
opt
OPTIMUM
-600
MATRIX INVERSION PROBLEM: COLUMN 1: $y_{31}$

FIG 4.21(c)

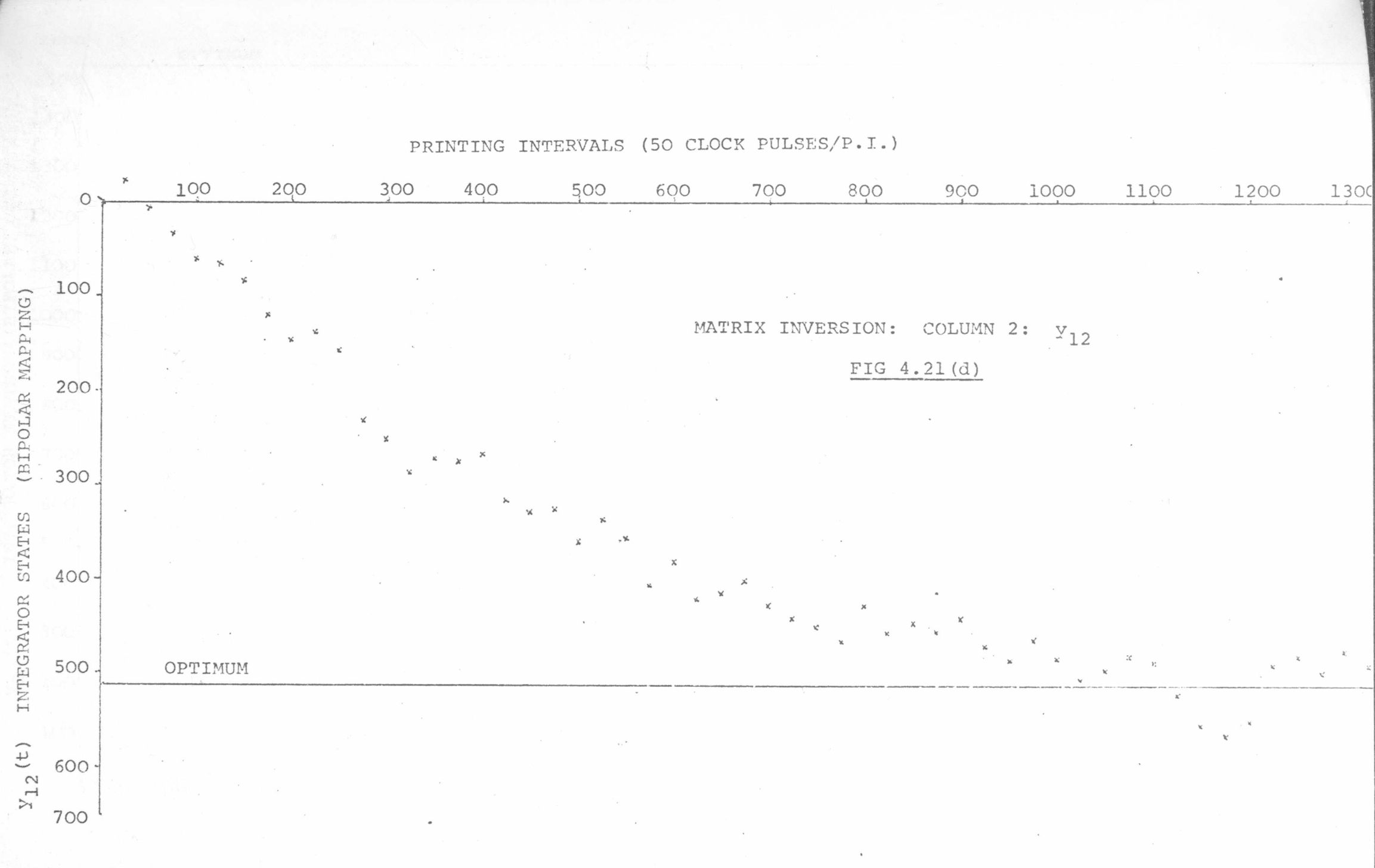

PRINTING INTERVALS (50 CLOCK PULSES/P.I.)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
$y_{12}(t)$ INTEGRATOR STATES (BIPOLAR MAPPING)
OPTIMUM
MATRIX INVERSION: COLUMN 2: $y_{12}$

FIG 4.21(d)

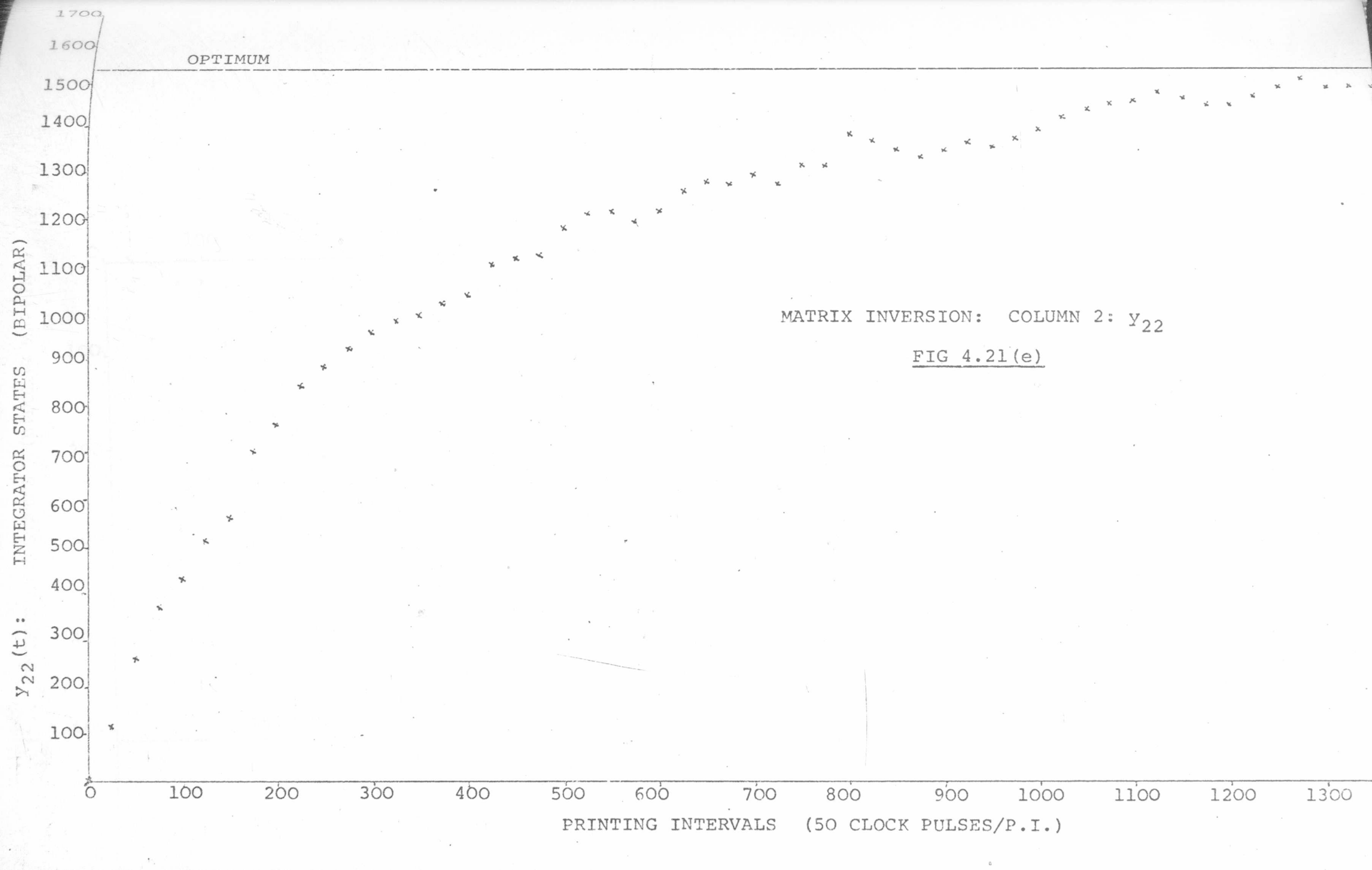
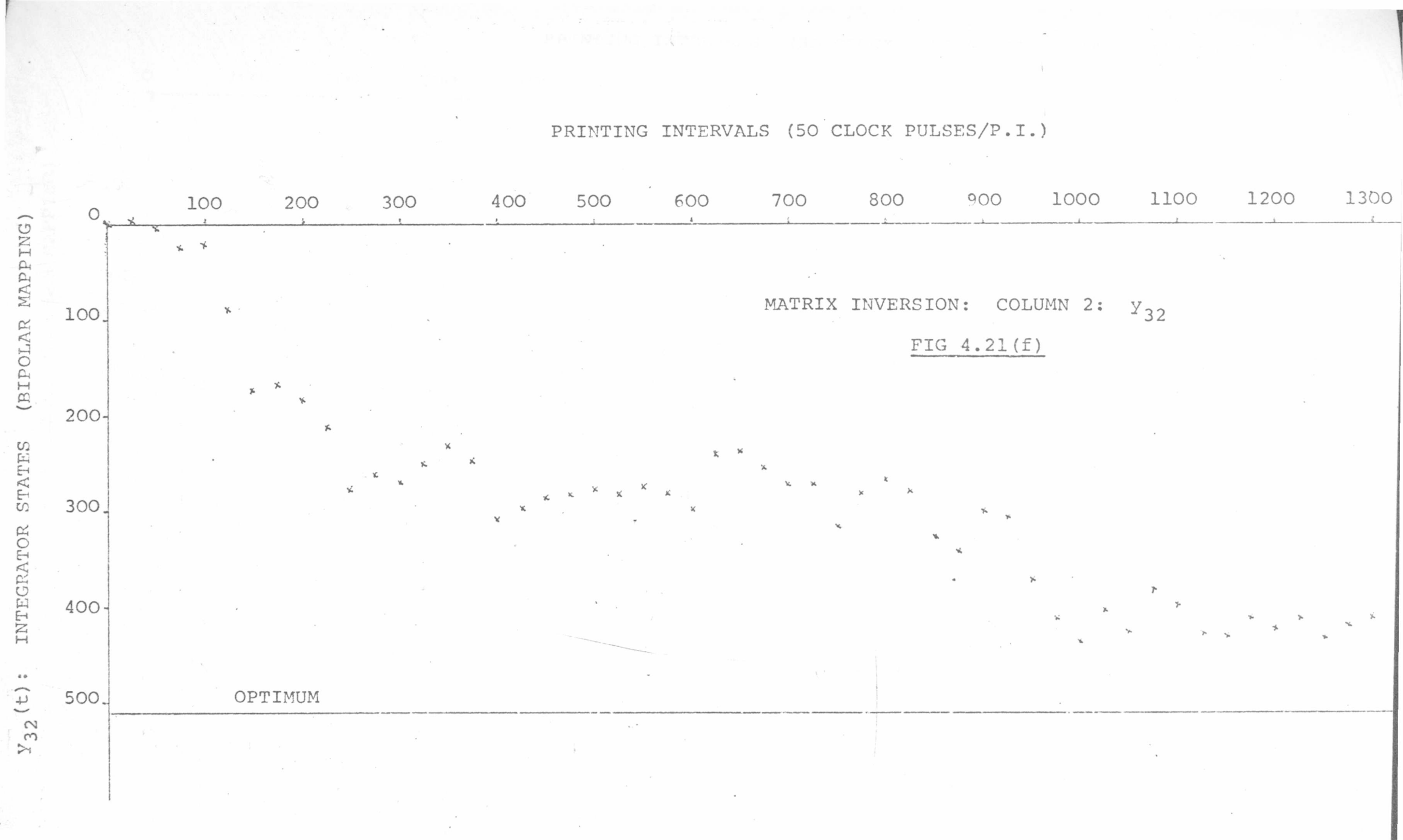

OPTIMUM
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
$y_{22}(t)$: INTEGRATOR STATES (BIPOLAR)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
PRINTING INTERVALS (50 CLOCK PULSES/P.I.)
MATRIX INVERSION: COLUMN 2: $y_{22}$

FIG 4.21(e)


PRINTING INTERVALS (50 CLOCK PULSES/P.I.)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
100
200
300
400
500
OPTIMUM
$y_{32}(t)$: INTEGRATOR STATES (BIPOLAR MAPPING)
MATRIX INVERSION: COLUMN 2: $y_{32}$
FIG 4.21(f)

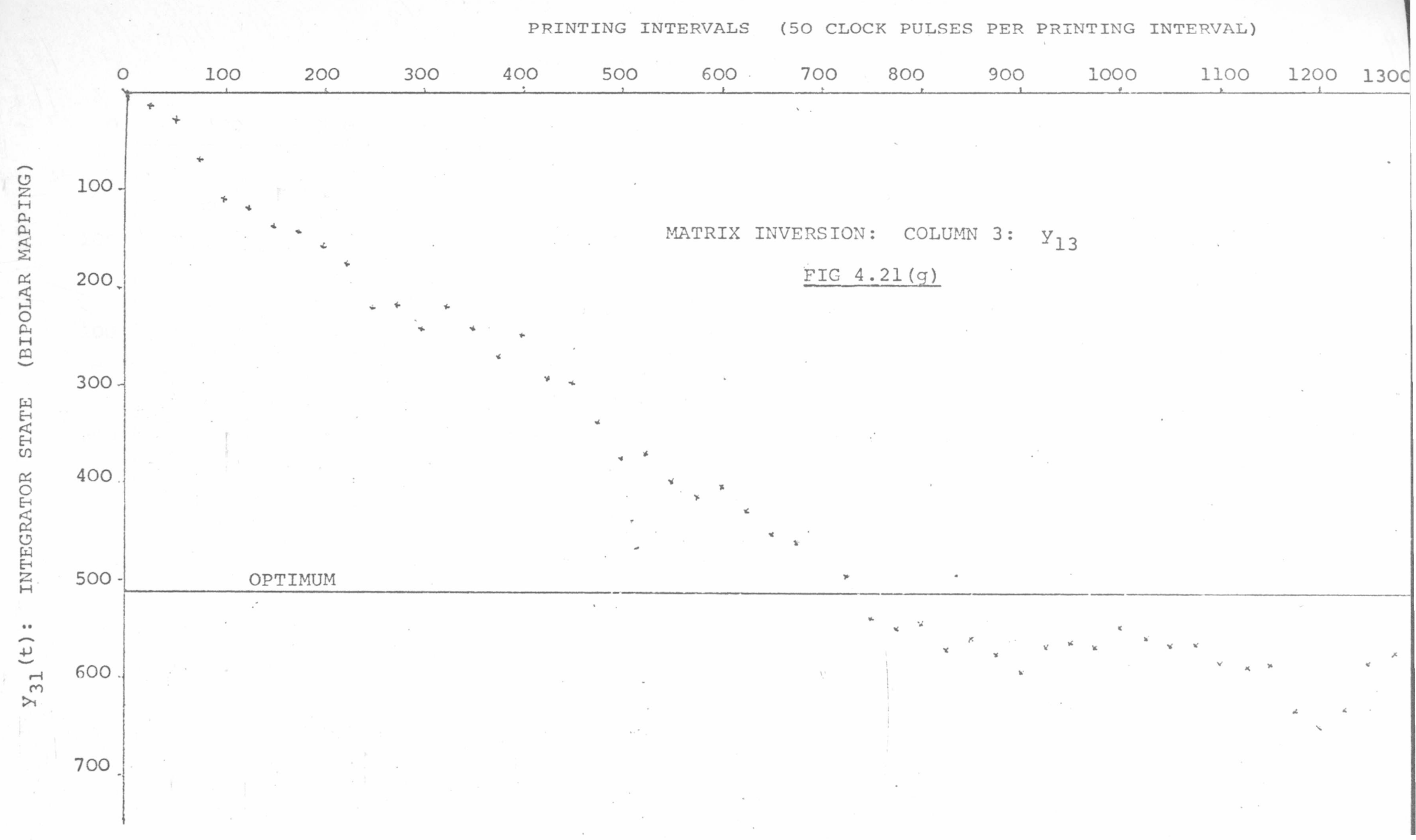
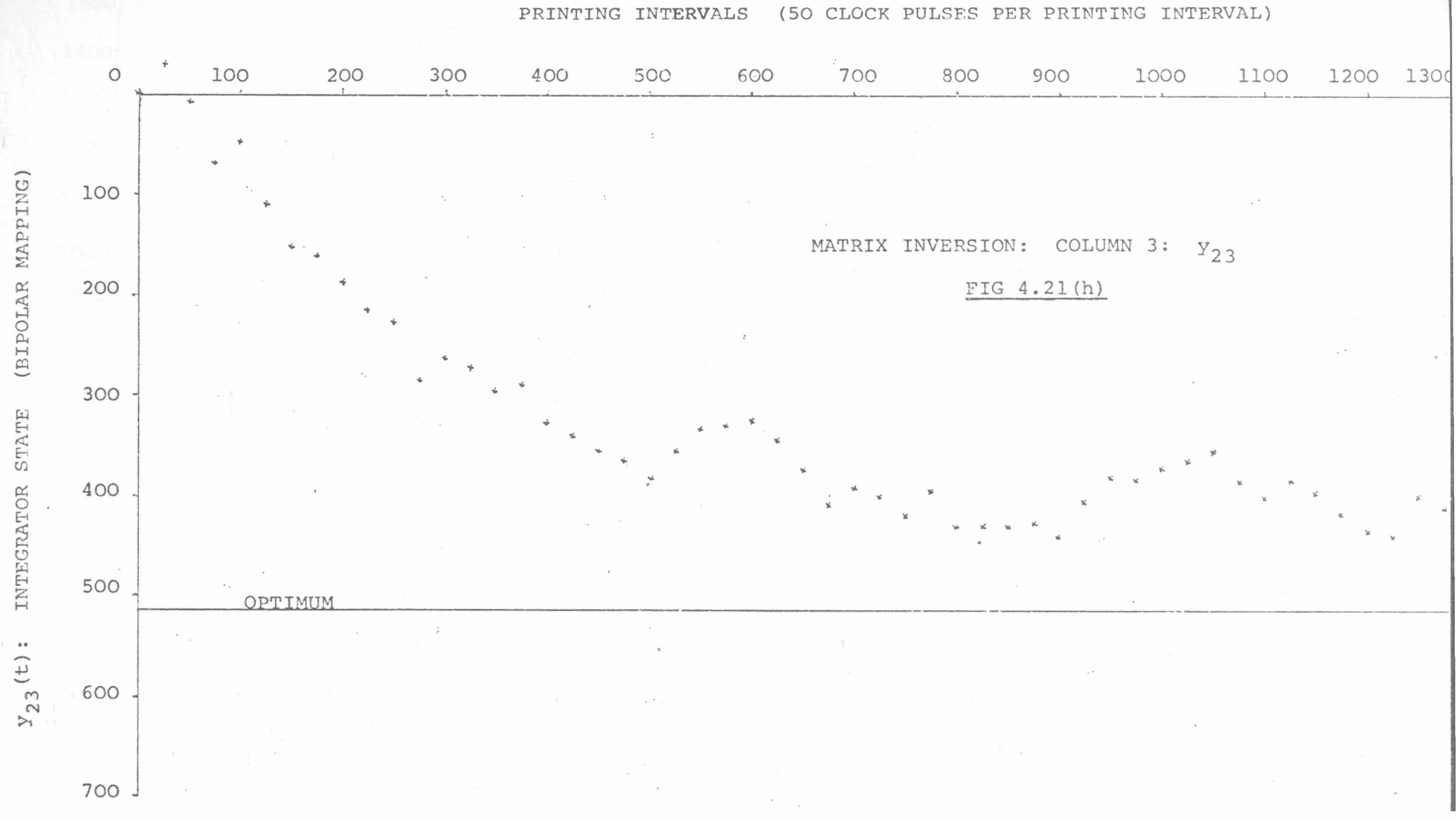

PRINTING INTERVALS (50 CLOCK PULSES PER PRINTING INTERVAL)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
100
200
300
400
500
600
700
OPTIMUM
$y_{31}(t)$: INTEGRATOR STATE (BIPOLAR MAPPING)
MATRIX INVERSION: COLUMN 3: $y_{13}$

FIG 4.21(g)


PRINTING INTERVALS (50 CLOCK PULSES PER PRINTING INTERVAL)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
100
200
300
400
500
600
700
$y_{23}(t)$: INTEGRATOR STATE (BIPOLAR MAPPING)
OPTIMUM
MATRIX INVERSION: COLUMN 3: $y_{23}$
FIG 4.21(h)

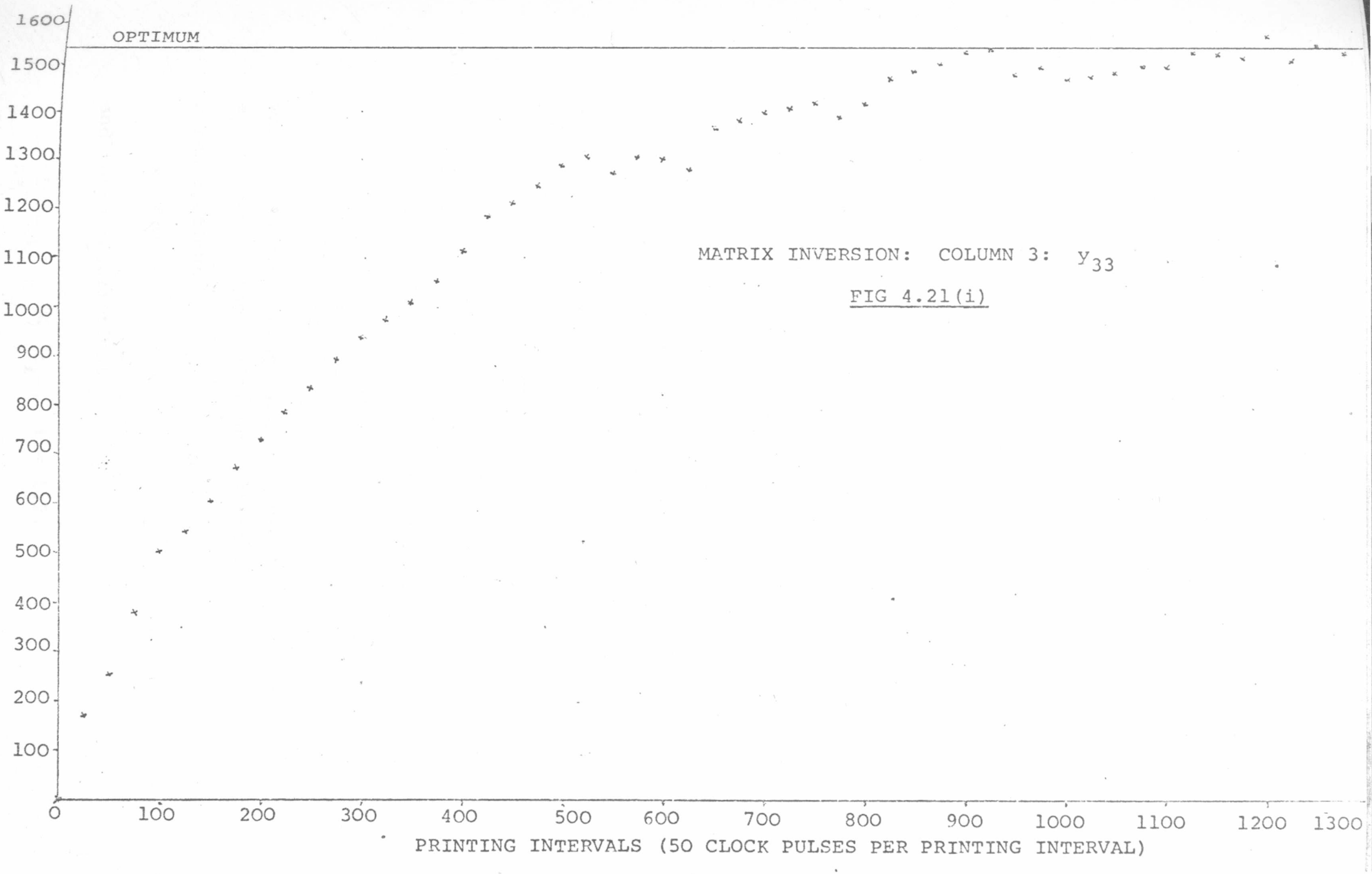

1600
OPTIMUM
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
$y_{33}(t)$: INTEGRATOR STATE (BIPOLAR MAPPING)
PRINTING INTERVALS (50 CLOCK PULSES PER PRINTING INTERVAL)
MATRIX INVERSION: COLUMN 3: $y_{33}$

FIG 4.21(i)

# CHAPTER 5

(In this chapter linear programming is examined and a possible stochastic computer circuit is investigated.)

## 5. Introduction

Another aspect of control engineering is the maximisation of a profit function or the minimisation of a loss function[31,32,33], and these objective functions may be subject to some constraints, eg, there are only limited quantities of the components available for some industrial process. We require a circuit which examines the constraints of a problem and finds a combination of the available components which will give the objective function an optimum value. The problem examined here is a maximisation problem, (the dual minimisation problem can also be defined), and a method[30] is proposed for the solution of this problem which introduces switching functions into the stochastic computer circuit. The circuit simulated is crude but it can be easily simulated in a digital computer programme. Track and store techniques can be simulated but computation times would be excessively long.

## 5.1 The Method of Steepest Ascent[30,31]

The following function of n variables is to be maximised:

$$z = \underline{b}^T \underline{x} = \sum_{k=1}^{n} b_k x_k \qquad \text{---- (5.1.1)}$$

which is a continuous single valued function of the $x_k$ subject to the following restrictions:

n restrictions

$$x_k \geqslant 0 \qquad \text{---- (5.1.2)}$$

and the m restrictions

$$\sum_{k=1}^{n} a_{\ell k} x_k \leqslant c_\ell \ , \quad \ell = 1,2, \ldots, m \qquad \text{---- (5.1.3)}$$

Thus /

Thus there are (m+n) restrictions.

Since z is differentiable we can define $\nabla z$:

$$\nabla z = \sum_{k=1}^{n} \frac{\partial z}{\partial x_k} i_k = \sum_{k=1}^{n} b_k i_k = \underline{b}^T \qquad \text{---- (5.1.4)}$$

where the $i_k$ are unit normal vectors. The function $\nabla z$ is everywhere normal to the hyperplanes of equal z, (z = constant), so it is in the direction of steepest ascent. Thus within the feasible solution space the objective point moves towards maximum values in the most direct route, but, the moving point must not violate the restrictions.

## 5.2 Examination of the Constraints (30)

The ith restriction is a hypervolume in n-space bounded by a hyperplane. Thus, the ith edge

$$\sum_{k=1}^{n} a_{ik} x_k = c_i$$

separates the region of space in which the inequality is satisfied from that which does not. The space in which the inequalities are satisfied is called the allowed region. The allowed region must be closed so that the objective function cannot increase without limit, ie, it cannot enter the restricted region. The $a_{ik}$ is an intercept of the ith edge with the kth axis. Inequalities lying completely in the restricted region are superfluous.

Let $N_i$ be a vector normal to the ith edge and directed toward the allowed region. There m of these vectors each associated with some particular restriction. Hence

$$N_i = -(a_{i1}, a_{i2}, \ldots, a_{in})^T \qquad \text{---- (5.2.1)}$$

and $$N = (N_1, N_2, \ldots\ldots, N_m) = -A^T \qquad \text{---- (5.2.2)}$$

We /

We define m quantities $\delta_1, \ldots, \delta_m$ such that

$$\delta_i = 0 \text{ if } \sum_{k=1}^{n} a_{ik} x_k \leq c_i$$

$$\delta_i = 1 \text{ if } \sum_{k=1}^{n} a_{ik} x_k > c_i \qquad \text{---- (5.2.3)}$$

Let $\Delta = [\delta_1, \ldots, \delta_m]$ ---- (5.2.4)

and let $\underline{f}$ be a function such that

$$\underline{f} = k\nabla z + \sum_{i=1}^{m} {}_iN_i = k\nabla z + \Delta N \qquad \text{---- (5.2.5)}$$

$$\Rightarrow \underline{f} = k\nabla z - \Delta A^T \qquad \text{---- (5.2.6)}$$

or

$$f_k = kb_k - \sum_{i=1}^{n} a_{ik}\delta_i \qquad \text{---- (5.2.7)}$$

Thus $\underline{f}$ depends on the position of the objective point because of the $\delta_i$'s. The position of the objective point is described by $\underline{x}$. Hence the velocity of the point in n-space is given by:

$$\underline{v} = \dot{\underline{x}} \qquad \text{---- (5.2.8)}$$

and $$\dot{\underline{x}} = \gamma \underline{f} \qquad \text{---- (5.2.9)}$$

$$\Rightarrow \dot{x}_k(t) = \gamma(Kb_k - \sum_{i=1}^{m} a_{ik}\delta_i) \qquad \text{---- (5.2.10)}$$

The objective point moves through the feasible solution space with a velocity of $\gamma K\nabla z$ until it reaches a restriction (say the ith). When this happens, the motion is defined by two vectors, viz., $\nabla z$ and $N_i$ (normal to the ith hyperplane). If $N_i$ is greater than the normal component of $K\nabla z$ the point is ejected from the restricted region. According to this model the rebound into the allowed region is infinitesimal in magnitude but this is not true in practice as will be demonstrated later. Thus, the vector ${}_iN_i$ vanishes and the gradient causes the motion of the objective point to reverse until it again enters the restricted region, ie, the objective point moves along the boundary in /

in the direction of $\nabla z$ on the ith hyperplane. See Figure 5.2. To prevent breakthrough into the restricted region we must have:

$$-K\nabla z\ N_i < N_i^2 \qquad \text{---- (5.2.11)}$$

$$\Rightarrow \quad K < \frac{\sum_{k=1}^{n} a_{ik}^2}{\sum_{k=1}^{n} a_{ik}b_k} \quad \text{for every } i \qquad \text{---- (5.2.12)}$$

## 5.3 Stochastic Hardware

The following discussions apply to two input devices.

### Introduction

The model proposed for solving this class of linear programming problem requires the use of level sensing methods and switching functions. In a conventional analogue computer information is represented by one continuous signal level so that magnitudes are estimated almost instantaneously. Level sensing in a stochastic computer means evaluating the probability of a random sequence of pulses using UP/DOWN counters, and this takes time to achieve using ADDIES. The level to be sensed is loaded into a shift register before the computation begins. During computation the content of the shift register is compared with the state of the ADDIE counter at each clock pulse and depending on the result of this test certain courses of action are possible. (See Figure 5.3(a) in connection with the following discussion.) For example, a binary sequence representing zero is loaded into the shift register. If the ADDIE state is greater than zero the circuit outputs a pulse train of probability 1.0 (equivalent to one machine unit). On the other hand, if the ADDIE state is less than or equal to zero the circuit outputs a random pulse train of probability 0.5. It can be arranged for this circuit to output other pulse trains if so desired.

However /

However the random variance on the input to the switch will increase as the moving point approaches a boundary so that the output random sequence from the switching ADDIE will gradually increase in probability rather than giving a sharp step output. This will show up the optimisation process due to 'misfires' from the circuit. The output probability function is essentially a cumulative distribution function. The problem of random variance giving poor switching may be slightly alleviated if the $c_i$ are loaded into this shift register and the sequences representing the

$$\sum_{k=1}^{n} a_{ik}\, x_k$$

are applied to the inputs to the ADDIES. Switching characteristics will improve as the modulus of

$$\sum_{k=1}^{n} a_k\, x_k$$

increases since the random variance in the system will decrease.

To obtain the fastest possible switching action small ADDIES with eight bit counters are used to sense the closeness of the moving point to the problem boundaries. To make switching even faster it was decided not to allow the counter to go below a certain number, eg, -15 or 113 states for an eight bit ADDIE.

The constraints imposed on the values of the $x_i$ are dealt with in a similar manner to the boundary sensing problem (see Figure 5.3(b)). The limiting value of each $x_i$ is loaded into a shift register and the contents of this register are compared with the contents of the corresponding stochastic integrater at each clock pulse. If the constraint is not violated the output probability sequence represents the contents of the integrater counter. However, if the constraints is violated the output sequence represents the contents of the shift register.

Twelve /

Twelve bit integrators are used to form the vector $\underline{x}$ so that the moving point will not move outside the permitted region before the system can decide whether or not it should eject this point. If the restrictions allow the variables to take a value of zero, which is represented by a probability of 0.5 or a mean of 2048 states in a twelve bit counter, the integraters must be allowed to go below 2048 states to generate an output random pulse train with the required probability. Since in a density function most points lie within ±3 standard deviations, the integraters should not be allowed to count below 1948 states.

The linear programming algorithm adopted requires the $\delta_i$ to be represented by probabilities of 0.5 or 1.0. See Figure 5.3(c) for details of the symbols chosen to represent the switching ADDIE and the limiting integrater.

## 5.4 Implementation of the Algorithm

The equations and inequalities to be implemented are:

$$z = \sum_{k=1}^{n} b_k x_k \qquad \text{---- (5.4.1)}$$

$$\varepsilon_\ell = \sum_{k=1}^{n} a_{\ell k} x_k - c_\ell, \quad \ell = 1,2,\ldots,m \qquad \text{---- (5.4.2)}$$

$$\dot{x}_k = Kb_k - \sum_{i=1}^{m} a_{ik}\delta_k, \text{ assuming } \gamma = 1.0 \qquad \text{---- (5.4.3)}$$

For the generalised case the following hardware elements are required:

Multiplication

$n + mn + n + mn = 2n(m+1)$ multipliers

Summation

$n + nm + n(m-1) = 2nm$ summers

Depending on the value of 2nm further compensating multipliers may be required.

Invertors /

Invertors

m + m = 2m invertors

ADDIES

1 twelve bit ADDIE to evaluate the objective function, z. m eight bit switching ADDIES to evaluate the $\varepsilon_\ell$.

Integrators

n limiting integrators

Comparators

This estimate excludes those used in the integrators and ADDIES.

2n(m+1) for coefficients

2nm noise lines of generating probability 0.5.

Hardware Savings

Because there are many branches in the network terminating at counters it is possible to make considerable hardware savings, eg, n(m+1) comparators can be saved because all the coefficients appear twice in different branches of the circuit. In the same way it may also be possible to save nm noise lines of probability 0.5 by feeding summers in different branches with the same sequence.

## 5.5 System Resolution and Scaling

The same remarks apply to linear programming.
See Chapter Four.

## 5.6 Numerical Example

Maximise $z = 0.5x_1 + x_2 + 0.333x_3$ subject to the following constraints:

$$\begin{aligned} x_1 + 0.666x_2 + 0.166x_3 &\leqslant 0.666 \\ 0.666x_1 + x_2 + 0.666x_3 &\leqslant 0.666 \\ x_1 &\geqslant 0 \\ x_2 &\geqslant 0 \\ x_3 &\geqslant 0 \end{aligned} \qquad \text{---- (5.6.1)}$$

Solving /

Solving this problem using the Simplex Method [32] for linear programming problems we have,

$$\text{Max } z = 0.6$$

$$\underline{x}_{opt} = \begin{bmatrix} 0.0 \\ 0.666 \\ 0.0 \end{bmatrix} \qquad \text{----(5.6.2)}$$

The problem is in a form which can be immediately implemented on a stochastic computer. A flowchart is given in Figures 5.5(a)-5.5(c). A digital computer programme which was written to simulate this class of problem is listed in APPENDIX 5A. The objective function, z, is not evaluated in the programme to save time.

## 5.7 Results

The results are expressed by numbers in the interval (-2048,2048). The trajectories of $x_1$, $x_2$ and $x_3$ are illustrated in Figures 5.6(a)-5.6(c). Results were printed out every fifty clock-pulses of the stochastic machine. The solution yielded by the simulation is,

$$\underline{x}_{opt} = \begin{bmatrix} 0.000 \\ 0.625 \\ 0.000 \end{bmatrix} \qquad \text{---- (5.6.3)}$$

which is a convincing demonstration of the usefulness of a stochastic circuit for solving linear programming problems. However, $x_2$ has come to rest below its optimum value of 0.666. This has happened because of the nature of the boundary sensing circuits. These use eight bit ADDIES which cannot resolve values less than 1/128. The error in $x_2$ is 0.0416 but the error detected by the constraint circuit is 0.0104 because of the attenuation factor of 0.25 in the summation process. This error arises because the moving point encounters a purely reflecting barrier. This means that the switches have no backlash and as a result the moving /

moving point will be located within the allowed region rather than on the boundaries as required for an optimal solution.

Each trajectory shows a pronounced 'zig-zag' and these 'zig-zags' occur at the same instant due to the boundary searching technique employed in this simulation. This 'zig-zag' occurs because the ADDIES cannot follow the moving point quickly enough. This means that there is inertia in the circuit resulting in a considerable time lag between the moving point crossing a boundary and the ADDIES switching. Further, since the $x_i$ are represented by non-stationary sequences the standard statistical tests do not apply in this case.

The solution to this linear programming problem could be considerably improved by using twenty-four bit integrators and twelve bit switching ADDIES. The 'zig-zag' effect would not be so pronounced and computations would be more accurate as smaller deviations from the boundaries could be measured. However, the system would be much slower. This investigation has shown that the variables $x_1$ and $x_3$ came to rest at a probability of 0.5 with very little variance while $x_2$ converged to a probability of 0.8 with considerable variance. These variances are not entirely random but are largely due to the action of the switching ADDIES.

These results show that it is possible to implement standard linear programming algorithms on a stochastic computer although accurate answers will only be achieved by using very long counters, and, the integrator counters must be very much longer than those used in the ADDIES.

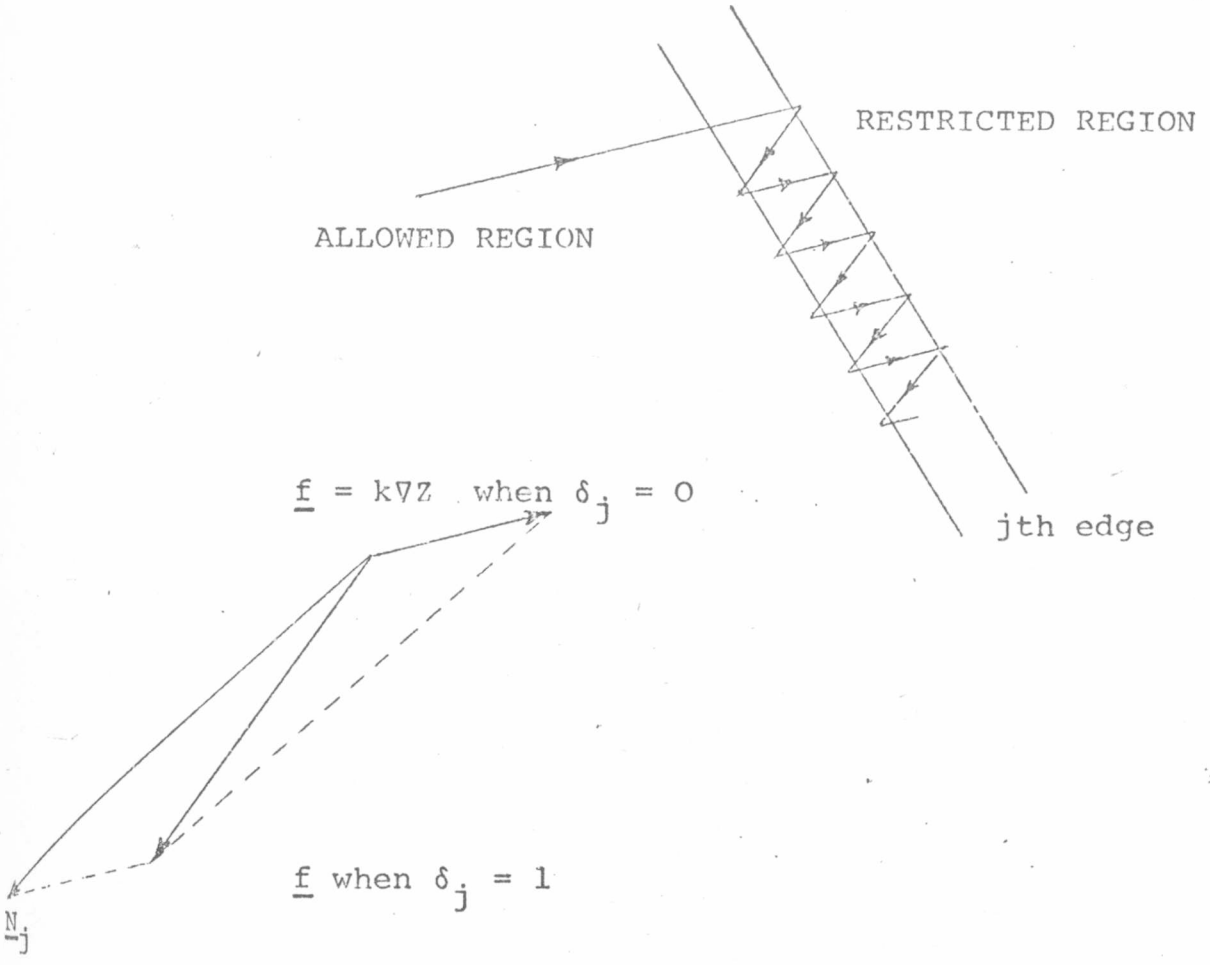
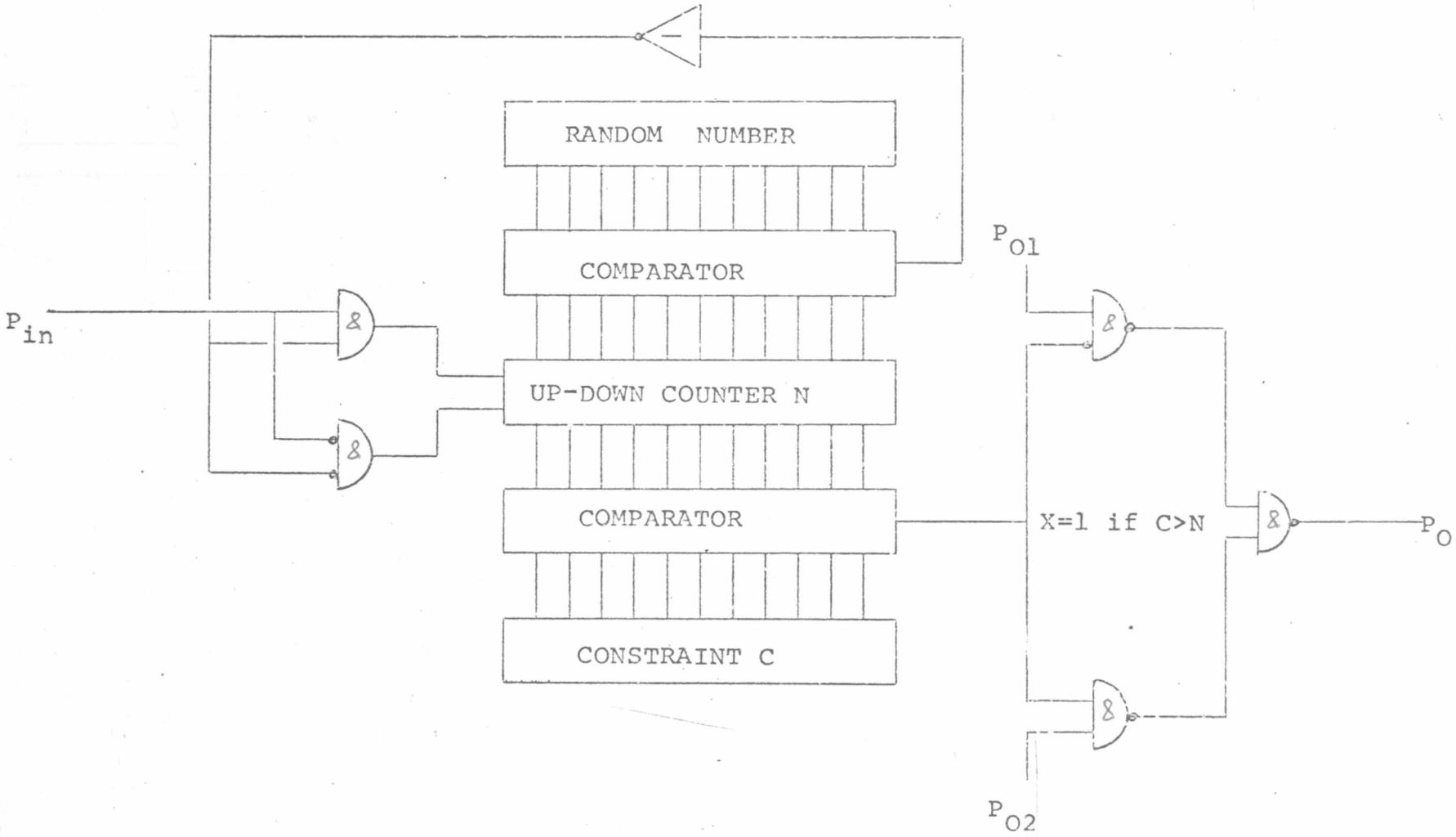
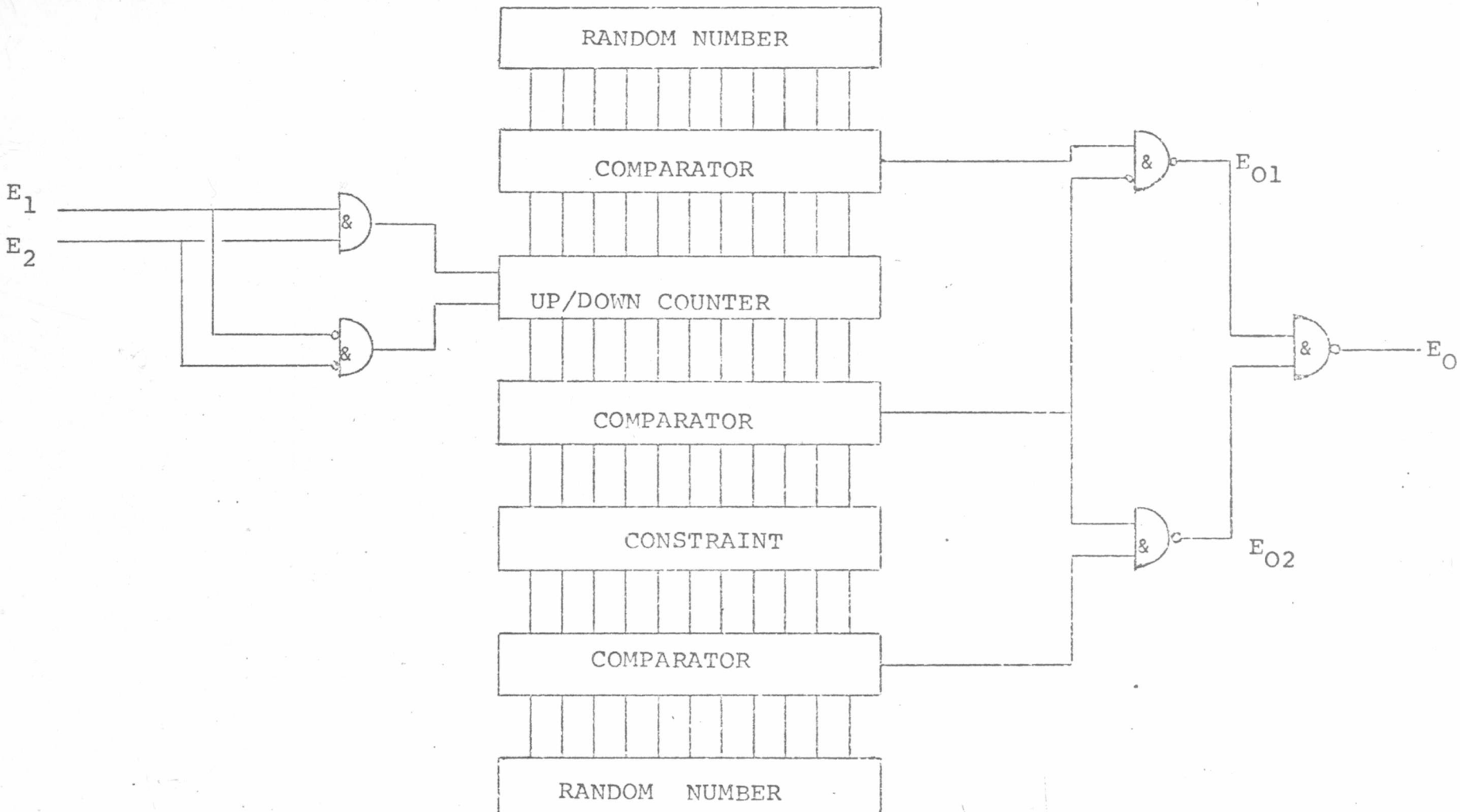

RESTRICTED REGION
ALLOWED REGION
$\underline{f} = k\nabla Z$ when $\delta_j = 0$
jth edge
$\underline{f}$ when $\delta_j = 1$
$\underline{N}_j$

FIG 5.2




PROBABILITY THRESHOLD SWITCH BASED ON THE NOISE ADDIE

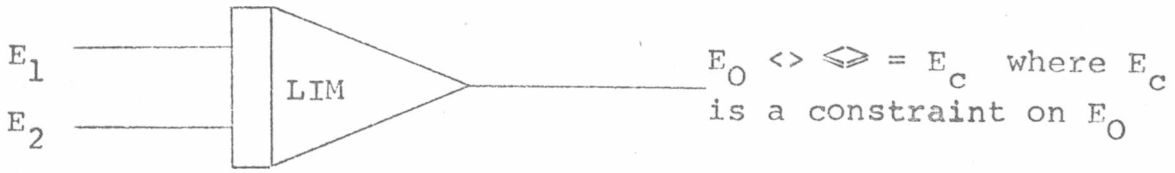
FIG 5.3(a)




INTEGRATOR WITH CONSTRAINT

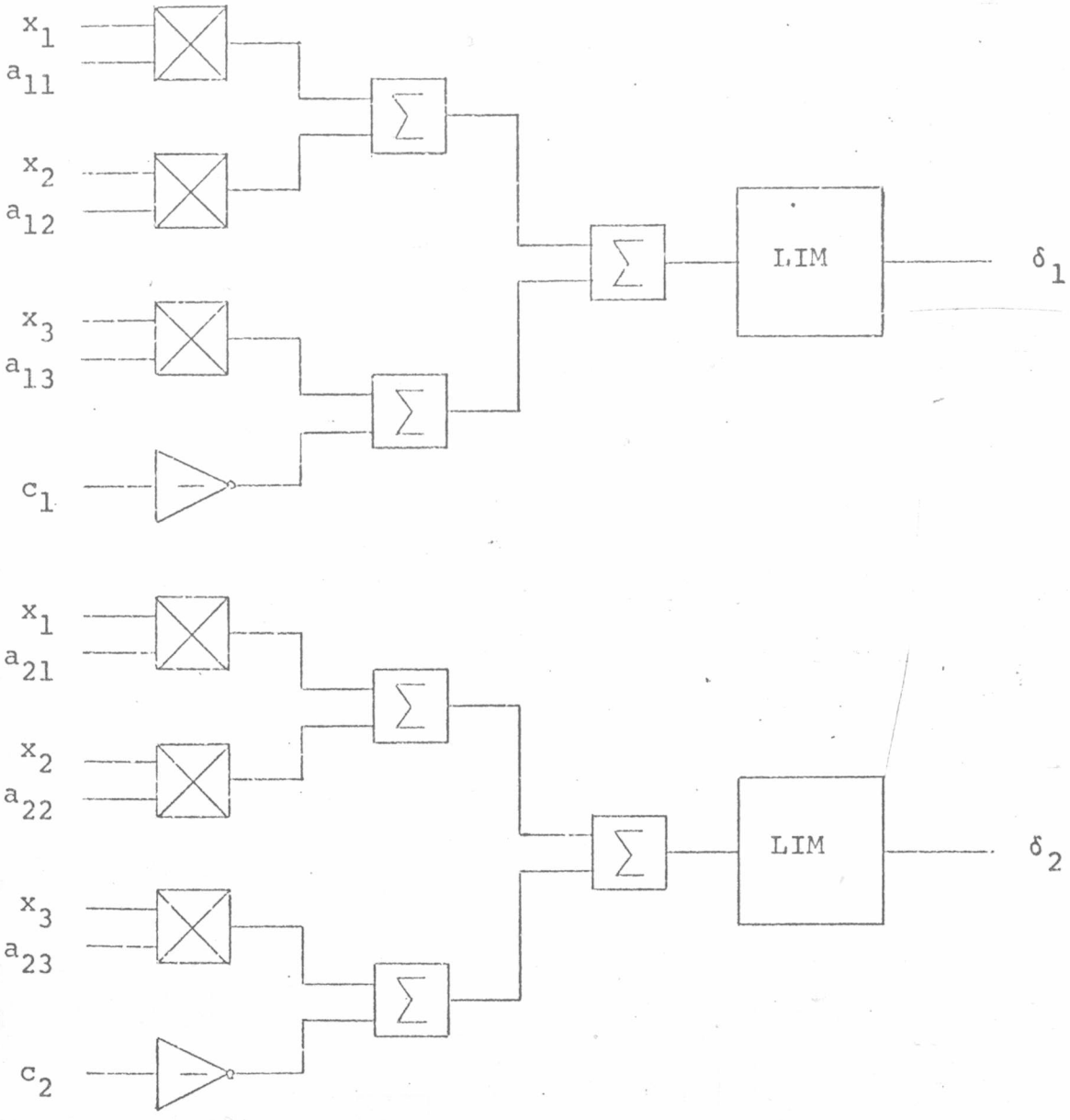
FIG 5.3(b)


SWITCHING ADDIE



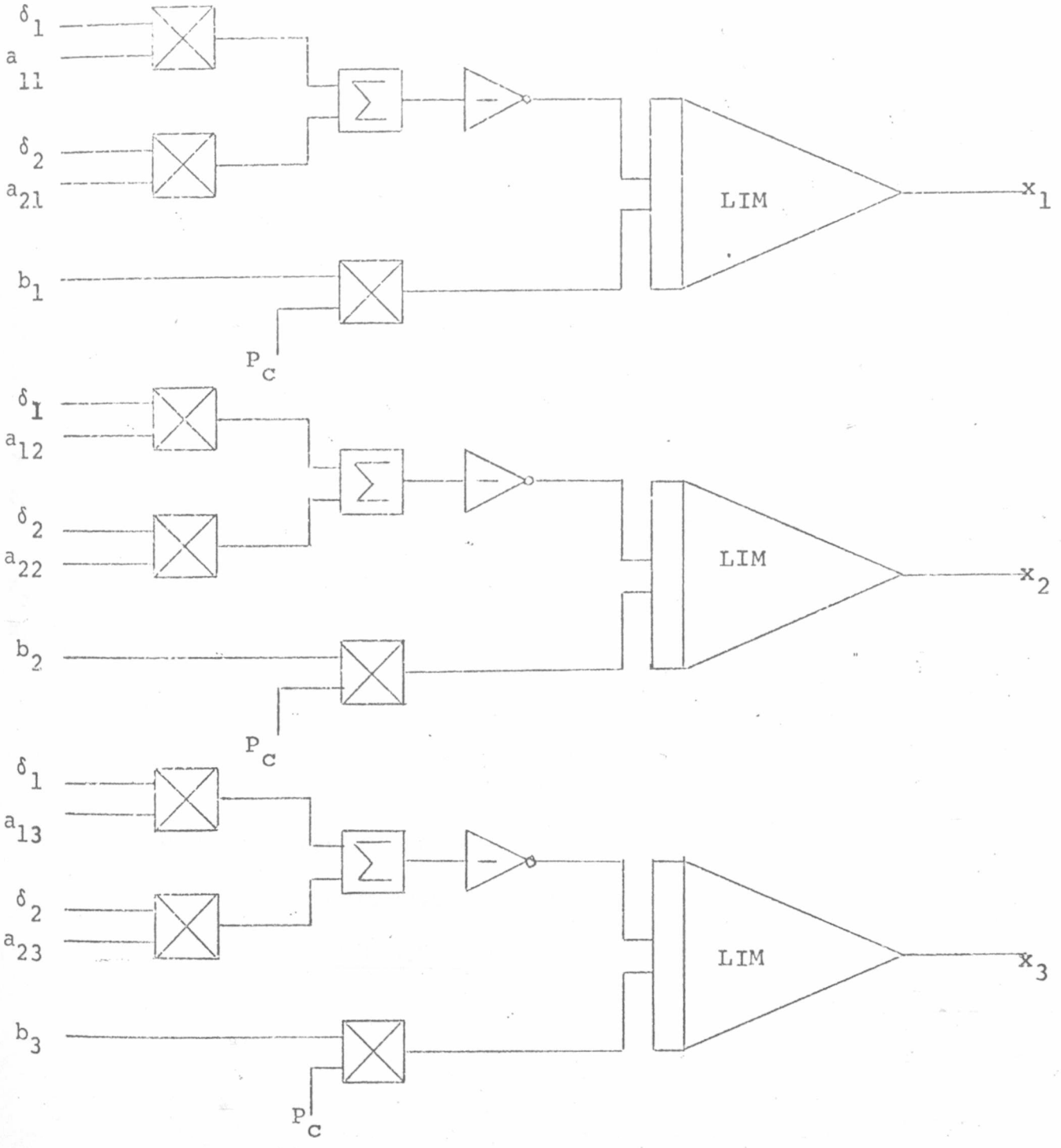

LIMITING INTEGRATOR

FIG. 5.3(c)




LINEAR PROGRAMMING PROBLEM: METHOD OF STEEPEST ASCENT: CONSTRAINTS

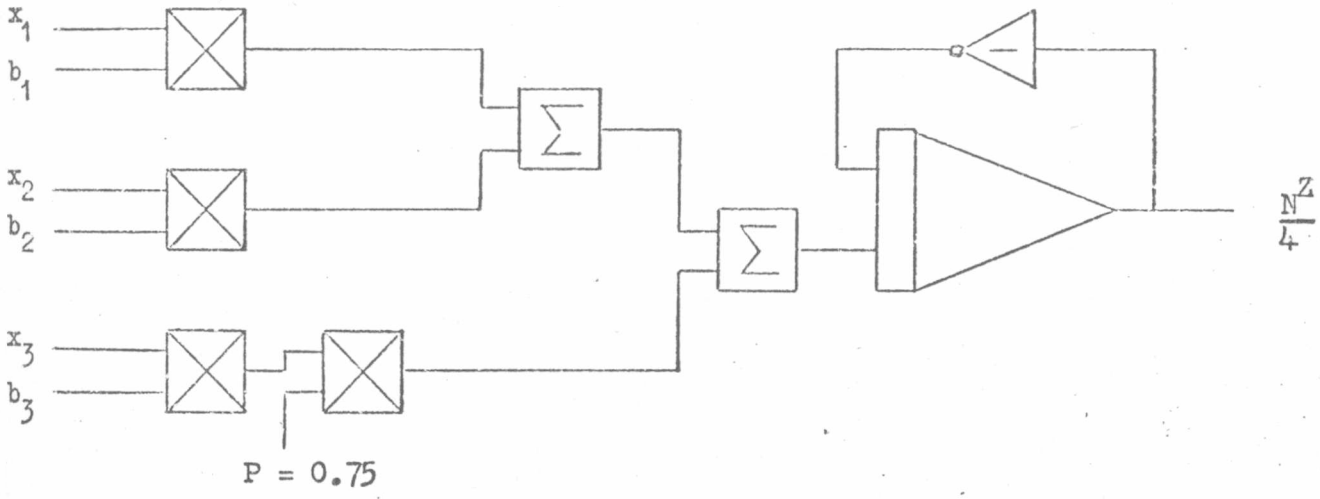
FIG 5.5(a)




$$P_c = \tfrac{1}{2} + \tfrac{1}{2}\left(\frac{K}{2}\right)$$

LINEAR PROGRAMMING PROBLEM: STEEPEST ASCENT EQUATIONS

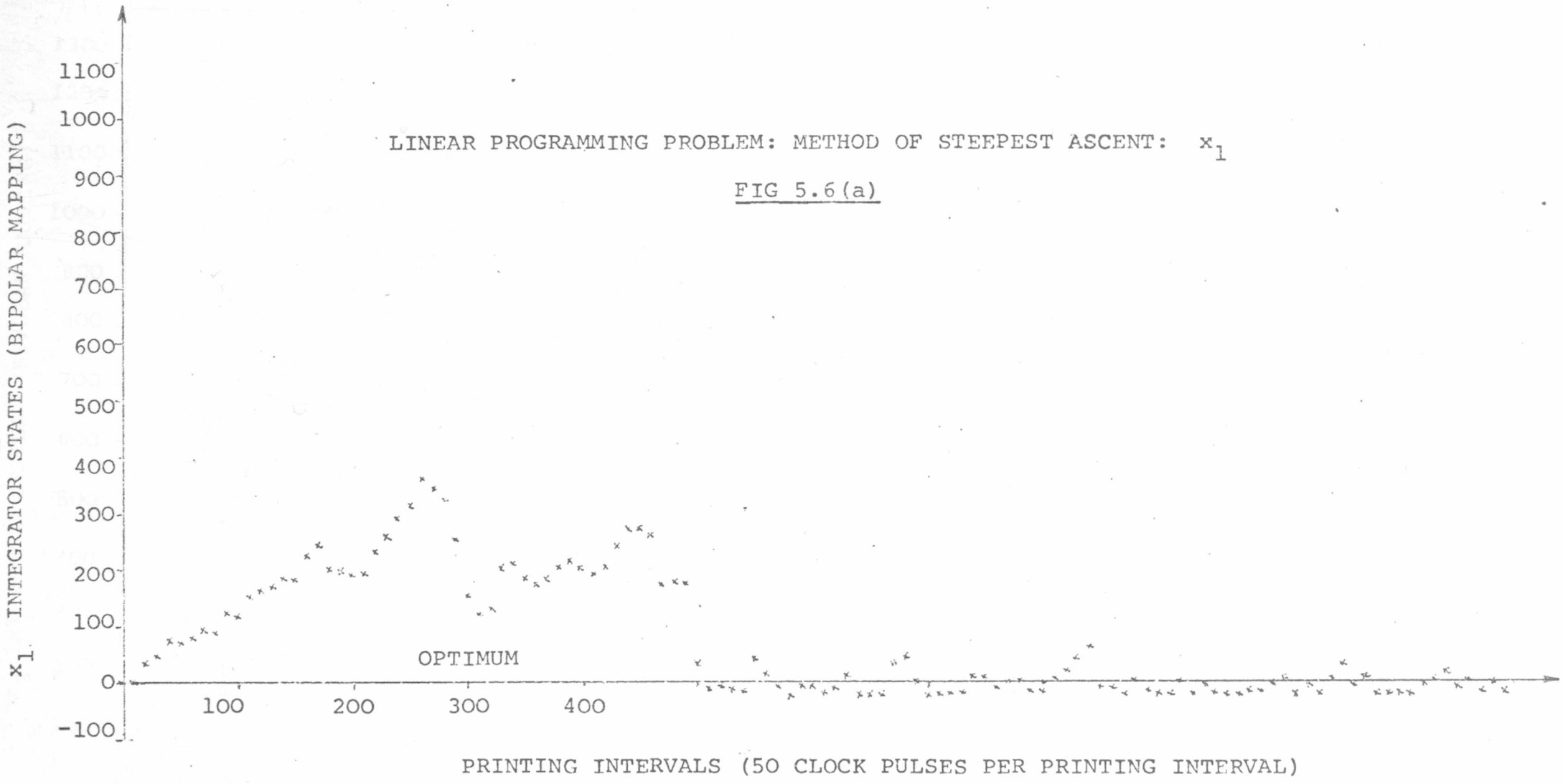
FIG 5.5(b)




LINEAR PROGRAMMING PROBLEM: METHOD OF STEEPEST DESCENTS

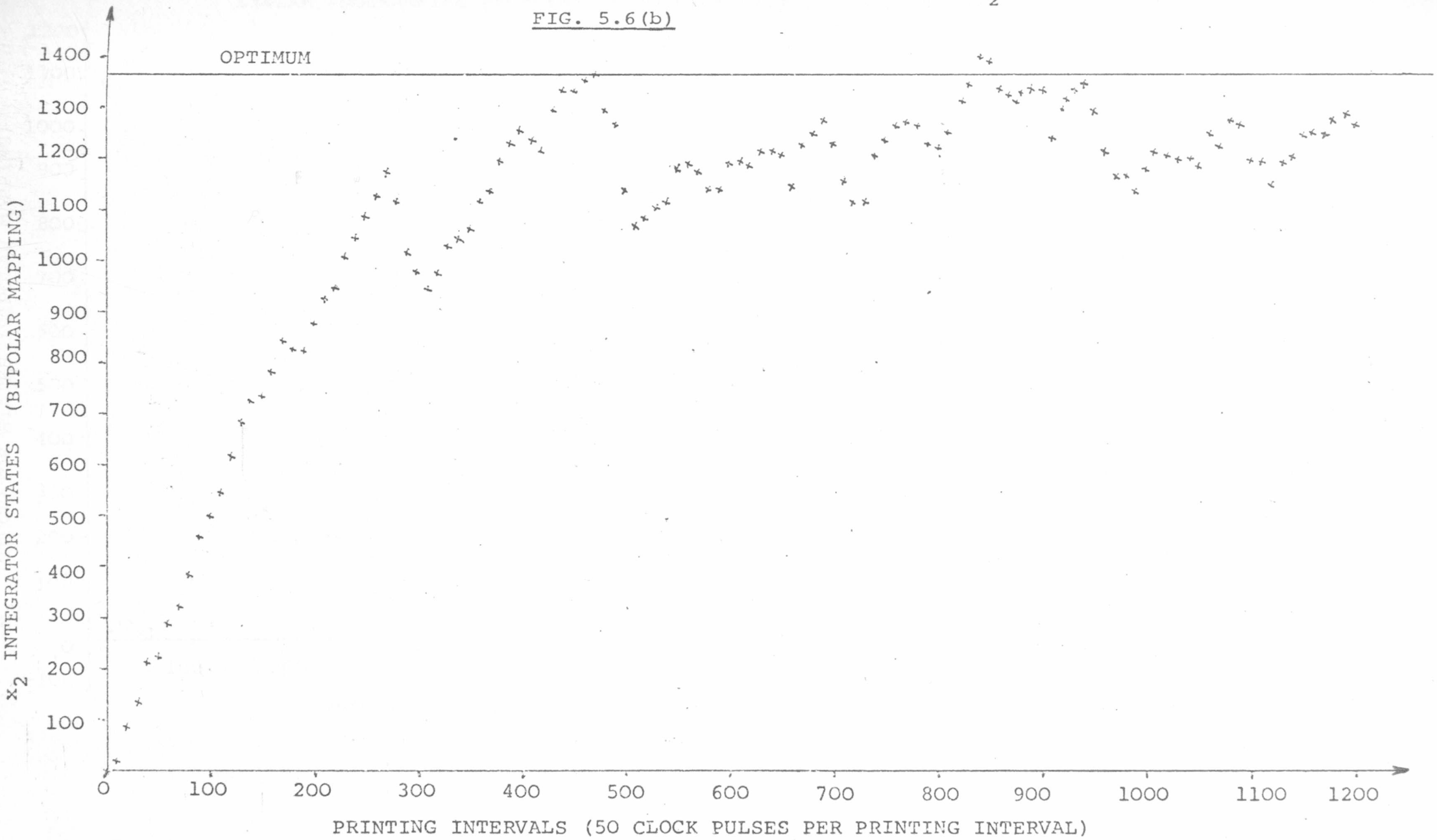
OBJECTIVE FUNCTION

FIG 5.5 (c).


LINEAR PROGRAMMING PROBLEM: METHOD OF STEEPEST ASCENT: $x_1$
FIG 5.6(a)
$x_1$ INTEGRATOR STATES (BIPOLAR MAPPING)
1100
1000
900
800
700
600
500
400
300
200
100
0
-100
OPTIMUM
100
200
300
400
PRINTING INTERVALS (50 CLOCK PULSES PER PRINTING INTERVAL)

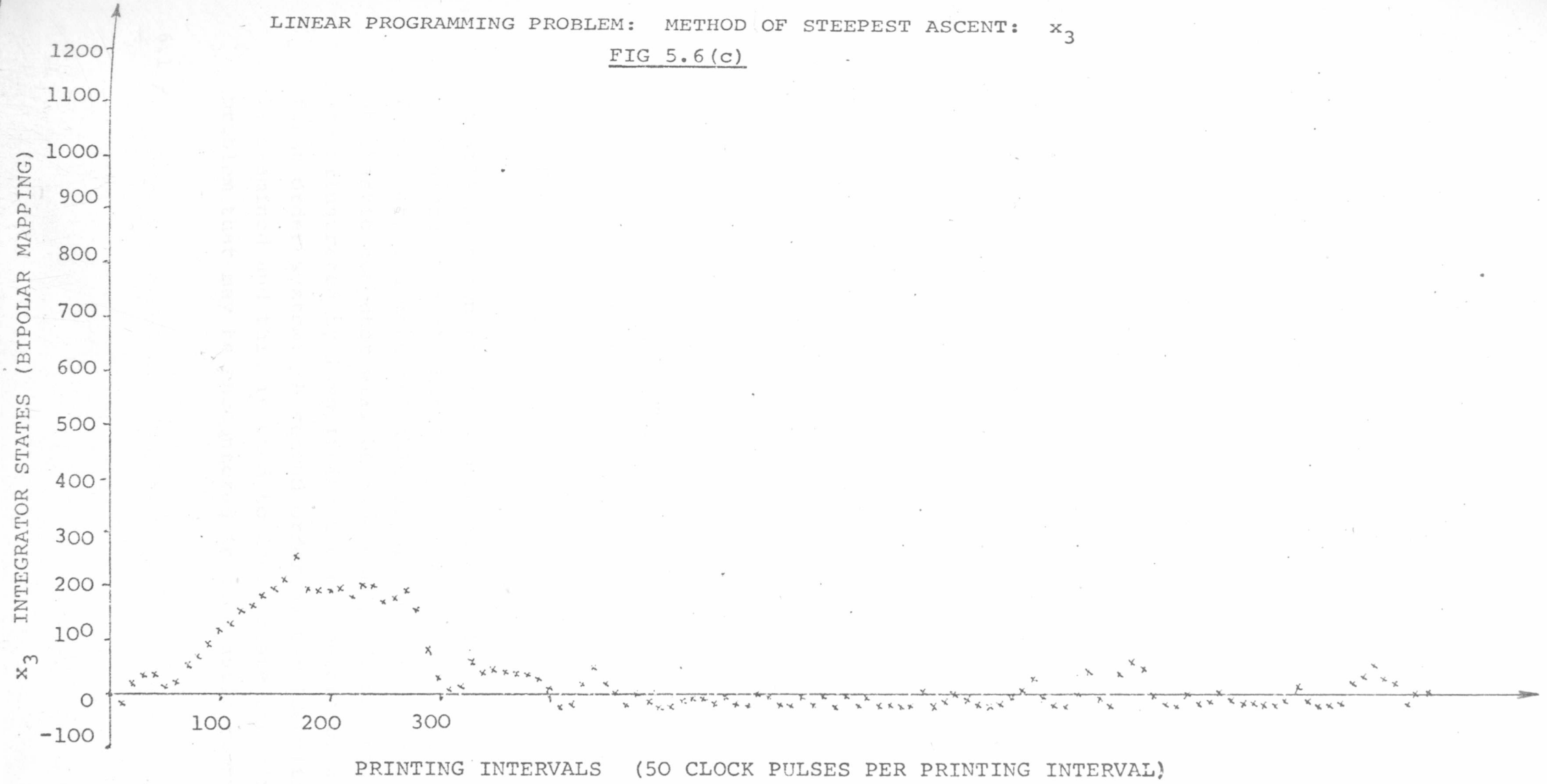
LINEAR PROGRAMMING PROBLEM: METHOD OF STEEPEST ASCENT: $x_2$

FIG. 5.6(b)



LINEAR PROGRAMMING PROBLEM: METHOD OF STEEPEST ASCENT: $x_3$

FIG 5.6(c)



# CHAPTER 6

(In this chapter circuits are proposed which may be used to identify the parameters of first and second order systems.)

## 6.0 Introduction

One important application of stochastic computing might be the implementation of algorithms which determine the parameters of a plant while it is in operation [28,36] If it can be assumed that a plant can be described by a linear differential equation with constant coefficients, then these coefficients can be evaluated knowing only the input to, and the output from the system. A schematic diagram of this identification technique is presented in Figure 6.0.

The criterion for correct identification is that the output of the model, m(t), is equal to the output of the actual plant, z(t). Thus we must find the minimum value for e(t) where,

$$e(t) = m(t) - z(t) \qquad \text{---- (6.0.1)}$$

One such method which achieves this is the method of steepest descent which minimises the scalar function.

$$f(t) = \tfrac{1}{2}\, e(t)^2 \qquad \text{---- (6.0.2)}$$

It will be shown how this particular solution can be mechanised on a stochastic computer. Then from this first method a more suitable algorithm for a digital stochastic computer will be developed. These ideas are illustrated by identifying the parameters of a first order system. A second order system identification is examined and this is used to demonstrate the kind of problem that may be encountered in stochastic computing.

6.1 /

## 6.1 Identification of the Parameters of a First Order System[36]

If it is assumed that some plant can be described by a first order linear differential equation then the model output, m(t), satisfies the differential equation,

$$\dot{m}(t) + \hat{\alpha}m(t) = \hat{\beta}x(t) \qquad \text{---- (6.1.1)}$$

where $\alpha$ and $\beta$ are the parameters to be identified and it is also assumed that they are time invariant. If the excitation, x(t), is a step function of value X and the initial conditions are zero then,

$$m(t) = \frac{\hat{\beta}X}{\hat{\alpha}}(1 - e^{-\hat{\alpha}t}) \qquad \text{---- (6.1.2)}$$

For a stable plant, $\hat{\alpha} > 0$.

To determine the values of the parameters $\alpha$ and $\beta$ we use the method of steepest descent to minimise equation (6.0.2). Hence,

$$\dot{\alpha}(t) = -K \frac{\partial}{\partial \alpha(t)}(\tfrac{1}{2} e(t)^2) = -Ke(t). \frac{\partial m(t)}{\partial \alpha(t)}, \quad K > 0$$

$$\text{---- (6.1.3)}$$

similarly,

$$\dot{\beta}(t) = -K\, e(t) \frac{\partial m(t)}{\partial \beta(t)}, \quad K > 0 \qquad \text{---- (6.1.4)}$$

From equation (6.1.2) we can see that the steady state solution is given by:

$$\lim_{t\to\infty} m(t) = \frac{\hat{\beta}}{\hat{\alpha}}X = \lim_{t\to\infty} z(t) \qquad \text{---- (6.1.5)}$$

Thus /

Thus the required condition $e(t) = 0$ can be satisfied by an infinite number of $\hat{\alpha}$ and $\hat{\beta}$ so that the values of the parameters obtained from the experiment may not be the correct ones. Only the ratio $\hat{\beta}/\hat{\alpha}$ is unique and we have,

$$\frac{\beta_{opt}}{\alpha_{opt}} = \frac{\hat{\beta}}{\hat{\alpha}} \qquad \text{---- (6.1.6)}$$

Before this identification method can be used the true value of either $\alpha$ or $\beta$ must be known.

The identification scheme is summarised below:-

$$\dot{\alpha}(t) = -Ke(t) \frac{\partial m(t)}{\partial \alpha(t)} \quad \text{given} \quad \beta = \beta_{opt} \qquad \text{---- (6.1.7)}$$

or

$$\dot{\beta}(t) = -Ke(t) \frac{\partial m(t)}{\partial \beta(t)} \quad \text{given} \quad \alpha = \alpha_{opt} \qquad \text{---- (6.1.8)}$$

Before we can implement the steepest descent equations we have to evaluate the partial derivatives $\partial m(t)/\partial \alpha(t)$ and $\partial m(t)/\partial \beta(t)$. These may be generated as the solutions of the sensitivity equations of the model.

## 6.2 Identification of $\alpha$ given the true value of $\beta$

For a system described by equation (6.1.1) the following equations have to be mechanised to determine $\alpha_{opt}$ given $\beta_{opt}$.

(i) $\beta(t) = \beta_{opt}$ ---- (6.2.1)

(ii) $\dot{\xi}_{\beta}(t) + \alpha(t)\, \xi_{\beta}(t) = x(t)$ ---- (6.2.2)

where $\xi_{\beta}(t) = \dfrac{\partial m(t)}{\partial \beta_{opt}}$

and equation (6.2.2) is derived from equation (6.1.1) in the following manner:

$$\frac{\partial}{\partial \beta_{opt}}(\dot{m}(t) + \alpha(t)m(t)) = \frac{\partial}{\partial \beta_{opt}}(\beta_{opt}\, x(t)) \qquad \text{---- (6.2.3)}$$

where $\beta_{opt} \neq 0$, and,

$$\lim_{t\to\infty} \xi_{\beta}(t) = \frac{x(t)}{\alpha_{opt}}, \quad \alpha_{opt} > 0 \qquad \text{---- (6.2.4)}$$

Also equation (6.2.2) always forms the basis of the model of the plant to be identified.

(iii) $$\dot{\xi}_{\alpha}(t) + \alpha(t)\, \xi_{\alpha}(t) = -m(t) \qquad \text{---- (6.2.5)}$$

where $\xi_{\alpha}(t)$ is derived in a similar manner to equation (6.2.2). Hence,

$$\frac{\partial}{\partial \alpha(t)}(\dot{m}(t) + \alpha(t)m(t)) = \frac{\partial}{\partial \alpha(t)}(\beta_{opt}\, x(t)) \qquad \text{---- (6.2.6)}$$

and

$$\lim_{t\to\infty} \xi_{\alpha}(t) = -\frac{m(t)}{\alpha_{opt}}, \quad \alpha_{opt} \neq 0 \qquad \text{---- (6.2.7)}$$

(iv) $$m(t) = \beta_{opt}\, \xi_{\beta}(t) \qquad \text{---- (6.2.8)}$$

(v) $$-e(t) = z(t) - m(t) \qquad \text{---- (6.2.9)}$$

(vi) $$\dot{\alpha}(t) = -K\, e(t)\, \xi_{\alpha}(t) \qquad \text{---- (6.2.10)}$$

The following computing units are required to implement this algorithm in a stochastic computer:

(a) 1 x summer;
(b) 4 x multipliers;
(c) 3 x inverters;
(d) 3 x integrators;
(e) 3 x comparators.

The /

The complete circuit is illustrated in Figure 6.2(a). If a conventional analogue computer is used the same hardware committment is required as shown in Figure 6.2(b).

The transient behaviour of this identification technique can be analysed using matrix algebra techniques. Equations (6.2.2) and (6.2.5) can be combined as a matrix differential equation. Hence,

$$D\begin{bmatrix}\xi_\alpha(t)\\ \xi_\beta(t)\end{bmatrix} + G\begin{bmatrix}\alpha(t) & 0\\ 0 & \alpha(t)\end{bmatrix}\begin{bmatrix}\xi_\alpha(t)\\ \xi_\beta(t)\end{bmatrix} = G\begin{bmatrix}-m(t)\\ x(t)\end{bmatrix} \quad \text{---- (6.2.11)}$$

where G is the gain of both integrators.
But $m(t) = \beta_{opt}\xi_\beta(t)$.

$$\Rightarrow D\begin{bmatrix}\xi_\alpha(t)\\ \xi_\beta(t)\end{bmatrix} + G\begin{bmatrix}\alpha(t) & \beta_{opt}\\ 0 & \alpha(t)\end{bmatrix}\begin{bmatrix}\xi_\alpha(t)\\ \xi_\beta(t)\end{bmatrix} = G\begin{bmatrix}0\\ x(t)\end{bmatrix} \quad \text{---- (6.2.12)}$$

Equation (6.2.12) is very like the method for solving sets of linear equations discussed in Chapter 4, although the coefficients are time varying in this case.

Let $\underline{y}(t) = \begin{bmatrix}\xi_\alpha(t)\\ \xi_\beta(t)\end{bmatrix}$, $A(t) = \begin{bmatrix}\alpha(t) & \beta_{opt}\\ 0 & \alpha(t)\end{bmatrix}$

and

$$\underline{b}(t) = \begin{bmatrix}0\\ x(t)\end{bmatrix},$$

hence, /

hence,

$$\dot{\underline{y}}(t) + A(t)\,\underline{y}(t) = \underline{b}(t) \qquad \text{---- (6.2.13)}$$

By analogy with the state variable method for solving sets of linear differential equations described in APPENDIX 4A we have:

$$\underline{y}(t) = [\underline{y}(0) - \underline{y}_{opt}(t)]\ F(-GA(t)) + \underline{y}_{opt}(t) \qquad \text{---- (6.2.14)}$$

and

$$\lim_{t\to\infty} F(-GA(t)) = 0 \qquad \text{---- (6.2.15)}$$

so that

$$\lim_{t\to\infty} \underline{y}(t) = \underline{y}_{opt}(t) \qquad \text{---- (6.2.16)}$$

and $G >> K$.

Thus if G is large enough $\underline{y}(t)$ will converge to almost its correct current value.

Let $\dot{\underline{u}}(t) = D\begin{bmatrix}\alpha(t)\\ \beta(t)\end{bmatrix} = \begin{bmatrix}\dot{\alpha}(t)\\ 0\end{bmatrix}$ if $\beta(t) = \beta_{opt}$

and $e(t) = m(t) - z(t)$, then,

$$\dot{\underline{u}}(t) = -Ke(t)\,\underline{y}(t) \qquad \text{---- (6.2.17)}$$

$$\Rightarrow \quad \dot{\underline{u}}(t) = -Ke(t)\{[\underline{y}(0) - \underline{y}_{opt}(t)]\ F(-GA(t)) + \underline{y}_{opt}(t)\} \qquad \text{---- (6.2.18)}$$

$$\Rightarrow \quad u(t) = -K\int_0^t e(\tau)\{[\underline{y}(0) - \underline{y}_{opt}(\tau)]\,F(-GA(\tau)) + \underline{y}_{opt}(\tau)\}d\tau \qquad \text{---- (6.2.19)}$$

and

$$\lim_{t\to\infty} \underline{u}(t) = \underline{u}_{opt} \qquad \text{---- (6.2.20)}$$

If G >> K, then,

$$\dot{\underline{u}}(t) \doteq -Ke(t)\ \underline{y}_{opt}(t) \qquad \text{---- (6.2.21)}$$

Equation (6.2.21) is the required form of the steepest descent equations presented earlier in this section.

The stochastic computer circuit was simulated in a FORTRAN IV digital computer programme and this is listed in APPENDIX 6A. The excitation, x(t), was chosen to be a step function since this is the easiest one to simulate.

## 6.3 Numerical Example

In this section a numerical example is chosen which demonstrates the kind of thing that can go wrong when the stochastic computer is used to identify the parameters of a system. Suppose a plant is described by the following scaled differential equation:

$$\dot{m}(t) + \alpha_{opt}\, m(t) = 0.5\, x(t) \qquad \text{---- (6.3.1)}$$

where x(t) is a step input of magnitude 0.5, and

$$\lim_{t\to\infty} z(t) = 0.5, \quad \beta_{opt} = 0.5$$

$$\lim_{t\to\infty} \xi_{\beta}(t) = \lim_{t\to\infty} \frac{x(t)}{(t)} = 1.0$$

and

$$\frac{G}{K} = 16 \quad \text{where} \quad K = f_c/4096.$$

This particular problem was simulated and the results are displayed in Figure 6.3(a). The same problem was solved on a conventional analogue computer and the corresponding graph of $\alpha(t)$ versus time is plotted in Figure 6.3(b). The stochastic computer simulation obviously does not converge to the required value of $\alpha_{opt} = 0.5$ while the conventional machine does.

Suppose the stochastic computer is sitting at its correct optimum value, then from Figure 6.3(a) it can be seen that if $\alpha(t)$ decreases slightly $\xi_\beta(t)$ should increase. But since $\xi_\beta(t) = 1.0$ and this is represented by a probability of 1.0 no increase is possible so that the error remains zero and no correcting signal can be applied to $\alpha(t)$ to push it back to the optimum value. Hence $\alpha(t)$ may drift downwards with no possibility of this accumulated discrepancy being removed. However, the voltage analogue computer can go slightly outside its dynamic range to produce the necessary correcting signal. The problem lies in choosing a scaling technique such that,

$$0 < \lim_{t\to\infty} \xi_\beta(t) < 1.0.$$

We can rewrite the first order differential equation in the form:

$$\dot{m}(t) + k\alpha(t)m(t) = k\,\beta_{opt}x(t), \quad k > 1.0 \qquad \text{---- (6.3.2)}$$

and let,

$$\beta_* = k\,\beta_{opt}$$

$$\alpha_* = k\,\alpha_{opt} \qquad \text{---- (6.3.3)}$$

Hence,

$$\lim_{t\to\infty} m(t) = \frac{\beta_* X}{\alpha_*} = \frac{\beta_{opt} X}{\alpha_{opt}} \qquad \text{---- (6.3.4)}$$

where X is the value of the step function.

If we choose k to be 4/3 then $\beta_* = 2/3$ and hence $\alpha_*$ should be 2/3 represented by a probability of 5/6. The problem was rerun using this new scaling and the /

the results are presented in Figure (6.3(c)). The graph clearly shows that $k\alpha(t)$ converges to a mean value of 1365 states which represents 2/3. Hence,

$$\alpha_{opt} = \frac{\alpha_*}{k} = 0.5 \quad \text{as required.}$$

The possibility of choosing a wrong scaling must always be guarded against by repeating the experiment with different values of k even though the original scaling is apparently valid.

## 6.4. Identification of β given the true value of α

In this case the following equations have to be implemented:

(i) The basic model of the system which is given by:

$$\frac{\partial}{\partial\beta(t)}(\dot{m}(t) + \alpha_{opt}m(t)) = \frac{\partial(\beta(t)x(t))}{\partial\beta(t)} \qquad \text{---- (6.4.1)}$$

$$\Rightarrow \dot{\xi}_\beta(t) + \alpha_{opt}\xi_\beta(t) = x(t) \qquad \text{---- (6.4.2)}$$

(ii) $m(t) = \beta(t)\xi_\beta(t)$ ---- (6.4.3)

(iii) $-e(t) = z(t) - m(t)$ ---- (6.4.4)

(iv) $\dot{\beta}(t) = -Ke(t)\xi_{\beta D}(t)$ ---- (6.4.5)

where $\xi_{\beta D}(t)$ is represented by the one bit delayed sequence used to represent $\xi_\beta(t)$. This prevents cross-correlation between $e(t)$ and $\xi_\beta(t)$.

The following computing units are required to implement this algorithm on a stochastic computer:

(a) 1 x summer;
(b) 3 x multipliers;
(c) 2 x inverters;
(d) /

(d) 2 x integrators;
(e) 3 x comparators;
(f) 1 x one bit delay.

The complete circuit is illustrated in Figure (6.4(a)) and the corresponding analogue computer circuit is presented in Figure (6.4(b)). The transient behaviour of this circuit is analysed in APPENDIX 6B and the behaviour of $\beta(t)$ is described by:

$$\beta(t) = \beta_{opt}\left(1 - \exp\left[-K\left(\frac{X}{\alpha_{opt}}\right)^2\left(t + \frac{2e^{-G\alpha_{opt}t}}{G\alpha_{opt}} - \frac{e^{-G2\alpha_{opt}t}}{G2\alpha_{opt}} - \frac{3}{2G\alpha_{opt}}\right)\right]\right) \quad \text{---- (6.4.6)}$$

A digital computer programme which simulates the above system is listed in APPENDIX 6D.

## 6.5 Numerical Example

In this case a plant is described by the scaled linear differential equation:

$$\dot{m}(t) + \alpha_{opt}\, m(t) = \beta(t)x(t) \quad \text{---- (6.5.1)}$$

where $x(t)$ is a step input of value 0.5, $\alpha_{opt} = 0.5$, $\lim_{t\to\infty} z(t) = 0.5$ and $G = K = \frac{f_c}{4096}$.

Results of the simulation are displayed in Figure (6.5(a)), and a conventional analogue computer solution is presented in Figure (6.5(b)). The stochastic computer circuit clearly converges to a mean value of 0.5, (1024 states), as required. There are some local variations in probability /

probability of up to 1.6% but these have very little significance in a statistical sense and we can conclude that the convergence error is zero. For this problem $\beta_{opt}$ is represented by a probability of 0.75 and the associated standard deviation is approximately 27 states so 99.8% of points should lie within ±8 states.

The transient response of the circuit agrees with equation (6.4.6) and the system converges after $2\times10^4$ clock pulses, and if the clock frequency is $10^6$ Hz, this is equivalent to 0.02 seconds. The time constant of this circuit is 0.013 seconds at 1 MHz clock frequency which means that the effective bandwidth is 77Hz. Equation (6.4.6) indicates that the control circuit used to evaluate $\beta(t)$ effectively introduces a delay into the feedback loop to give a transient response which looks like a ramp. Thus by purposely introducing delays or non-linearities into the feedback loop of an ADDIE it may be possible to produce an output interface which has a greater bandwidth than a simple noise ADDIE. One such device is investigated in APPENDIX 6C.

## 6.6 Development of an Alternative Algorithm for First Order System Identification

In section 6.1 it was stated that the constant parameter, $\alpha$, was evaluated using the following equation:

$$\dot{\alpha}(t) = -Ke(t)\,\frac{\partial m(t)}{\partial \alpha(t)}$$

We can interpret $\partial m(t)/\partial\alpha(t)$ as a time varying weight which is multiplied with $e(t)$ to produce a driving signal which adjusts $\alpha(t)$ towards its optimal value. Much computational labour may be involved in evaluating the partial derivative $\partial m(t)/\partial\alpha(t)$ and it is possible to produce an equally effective weighting function with much less effort. This partial derivative is evaluated using the following sensitivity equation:

$$D\{\frac{\partial m}{\partial \alpha}\} + G\alpha \frac{\partial m}{\partial \alpha} = -Gm, \quad G > 0 \qquad \text{---- (6.6.1)}$$

and

$$\lim_{t\to\infty} \frac{\partial m}{\partial \alpha} = -\frac{\beta_{opt} X}{\alpha_{opt}} .$$

If G is large enough we have

$$\frac{\partial m(t)}{\partial \alpha(t)} = -\frac{m(t)}{\alpha(t)} ,$$

but,

$$\frac{\partial m(t)}{\partial \alpha(t)} = \frac{|m(t)|}{|\alpha(t)|} \operatorname{sgn}(-m(t)).\operatorname{sgn}(\alpha(t)) \qquad \text{---- (6.6.2)}$$

However, $\alpha(t)$ must be positive to produce a stable system. Hence,

$$\frac{\partial m(t)}{\partial \alpha(t)} = W(t) \operatorname{sgn}(-m(t)), \quad W(t) > 0 \qquad \text{---- (6.6.3)}$$

If W(t) is made constant then there only remains the problem of determining the sign of $-m(t)$. This is easily achieved using the threshold switches discussed in Chapter 5. If $\operatorname{sgn}(-m(t))$ is positive then a probability representing W is obtained at the output otherwise -W is given. The steepest descent equation now becomes,

$$\dot{\alpha}(t) = -Ke(t)\, W \operatorname{sgn}(-m(t)), \quad W > 0 \qquad \text{---- (6.6.4)}$$

and since both K and W are constants we have:

$$\dot{\alpha}(t) = -W_1\, e(t) \operatorname{sgn}(-m(t)), \quad W_1 > 0 \qquad \text{---- (6.6.5)}$$

The size of $W_1$ will affect the time taken for the system to reach optimum. To determine $\alpha_{opt}$ given $\beta_{opt}$ the following equations have to be mechanised:

(i) /

$$\text{(i)}\quad \beta(t) = \beta_{opt} \qquad \text{---- (6.6.6)}$$

$$\text{(ii)}\quad \ddot{\xi}_{\beta}(t) + \alpha(t)\dot{\xi}_{\beta}(t) = x(t) \qquad \text{---- (6.6.7)}$$

$$\text{(iii)}\quad W_1\ \mathrm{sgn}(-m(t)) \qquad \text{---- (6.6.8)}$$

$$\text{(iv)}\quad m(t) = \beta_{opt}\xi_{\beta}(t) \qquad \text{---- (6.6.9)}$$

$$\text{(v)}\quad -e(t) = z(t) - m(t) \qquad \text{---- (6.6.10)}$$

$$\text{(vi)}\quad \dot{\alpha}(t) = -W_1 e(t)\,\mathrm{sgn}(-m(t)) \qquad \text{---- (6.6.11)}$$

The following stochastic operators are required to implement this algorithm on a stochastic computer:

(a) 1 x summer;
(b) 3 x multipliers;
(c) 2 x inverters;
(d) 2 x integrators;
(e) 1 x threshold switch;
(f) 3 x comparators.

The complete stochastic computer circuit is illustrated in Figure (6.6(a)) and this circuit was simulated in a FORTRAN programme which is listed in APPENDIX 6E.

## 6.7 Numerical Example

The example used here is the one used in section 6.3, and the same scaling technique was adopted to prevent limiting in the circuit. The results are presented in Figure (6.7(a)) and the graph clearly shows that $K\alpha(t)$ converges to a mean value of 1365 states although it has a deterministic oscillation about its steady state value. This oscillation has an amplitude of 100 states or 5% of the dynamic range. From previous investigations into the behaviour of underdamped circuits /

circuits subjected to random excitations it can be concluded that the oscillations observed in the above identification circuit are due to the random variance inherent in the stochastic computing method. As before, the random variance, and hence the induced oscillations, can be reduced by increasing the capacities of the integrators used in the circuit but this will slow up the identification process.

## 6.8 Identification of the Parameters of a Second Order System (36)

The ideas developed in sections 6.1, 6.2, 6.4 and 6.6 can be extended to second order systems. However, there is the danger that errors introduced by random variance may prevent the identification algorithm from converging to any solution. The random variance could be reduced by increasing the capacity of the integrators used; but to obtain reasonable solution times all the simulations performed so far have used integrator capacities of up to twelve bits.

It is assumed that some system can be described by the second order linear differential equation

$$\ddot{m}(t) + \hat{\alpha}\dot{m}(t) + \hat{\beta}m(t) = \hat{\nu}x(t) \qquad \text{---- (6.8.1)}$$

where $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\nu}$ are the parameters to be identified. For the purposes of this investigation we assume that $\hat{\nu}$ is known and is equal to unity so that $\hat{\alpha}$ and $\hat{\beta}$ have to be identified. Hence,

$$\ddot{m}(t) + \alpha\dot{m}(t) + \beta m(t) = x(t) \qquad \text{---- (6.8.2)}$$

where

$$\hat{\alpha} = 2\hat{\xi}\omega_n \qquad \text{---- (6.8.3)}$$

and

$$\hat{\beta} = \hat{\omega}_n^2 \qquad \text{---- (6.8.4)}$$

Applying /

Applying steepest descent techniques we have:

$$\dot{\alpha}(t) = -Ke(t)\xi_{\alpha}(t) \quad , \quad K > 0 \qquad \text{---- (6.8.5)}$$

$$\dot{\beta}(t) = -Ke(t)\xi_{\beta}(t) \quad , \quad K > 0 \qquad \text{---- (6.8.6)}$$

$$\nu = \nu_{opt} = 1.0 \qquad \text{---- (6.8.7)}$$

where

$$\xi_{\alpha}(t) = \frac{\partial m(t)}{\partial \alpha(t)} \quad , \qquad \text{---- (6.8.8)}$$

$$\xi_{\beta}(t) = \frac{\partial m(t)}{\partial \beta(t)} \quad , \qquad \text{---- (6.8.9)}$$

and

$$e(t) = m(t) - z(t) \qquad \text{---- (6.8.10)}$$

The weights $\xi_{\alpha}(t)$ and $\xi_{\beta}(t)$ can be derived by solving the sensitivity equations for the system. In the steady state:

$$\hat{\alpha} = \alpha_{opt} \qquad \text{---- (6.8.11)}$$

$$\hat{\beta} = \beta_{opt} \qquad \text{---- (6.8.12)}$$

The same results can be obtained without having to solve the sensitivity equations if we replace $\xi_{\alpha}(t)$ by

$$W_{\alpha}(t) = -K_{\alpha}\dot{m}(t) \quad , \quad K_{\alpha} > 0 \qquad \text{---- (6.8.13)}$$

and $\xi_{\beta}(t)$ by,

$$W_{\beta}(t) = -K_{\beta}m_{D}(t), \quad K_{\beta} > 0 \qquad \text{---- (6.8.14)}$$

where $m_{D}(t)$ is a one bit delayed version of $m(t)$, $\alpha(t)$, $\beta(t) > 0$, and

$$K_{\alpha} : K_{\beta} = 0.25:1 \qquad \text{---- (6.8.15)}$$

The /

The model of the system is represented by:

$$\ddot{\xi}_\nu + \alpha(t)\,\dot{\xi}_\nu + \beta\xi_\nu = x \qquad \text{---- (6.8.16)}$$

where

$$\xi_\nu(t) = \frac{\partial m(t)}{\partial \nu} \qquad \text{---- (6.8.17)}$$

but, since $\nu = 1$,

$$\xi_\nu(t) = m(t) \qquad \text{---- (6.8.18)}$$

A stochastic computer circuit representing the plant and the identifying circuit is illustrated in Figure (6.8(a)), and the corresponding analogue computer circuit is given in Figure (6.8(b)).

A FORTRAN programme which simulates the identification procedure is listed in APPENDIX 6F.

The following numerical example was chosen to test the algorithm:

$$\ddot{m} + 5\dot{m} + 50m = 10\sin 2\pi t \qquad \text{---- (6.8.19)}$$

$$\ddot{m} + 0.5(10\dot{m}) + 0.5(100m) = 10\sin 2\pi t \qquad \text{---- (6.8.20)}$$

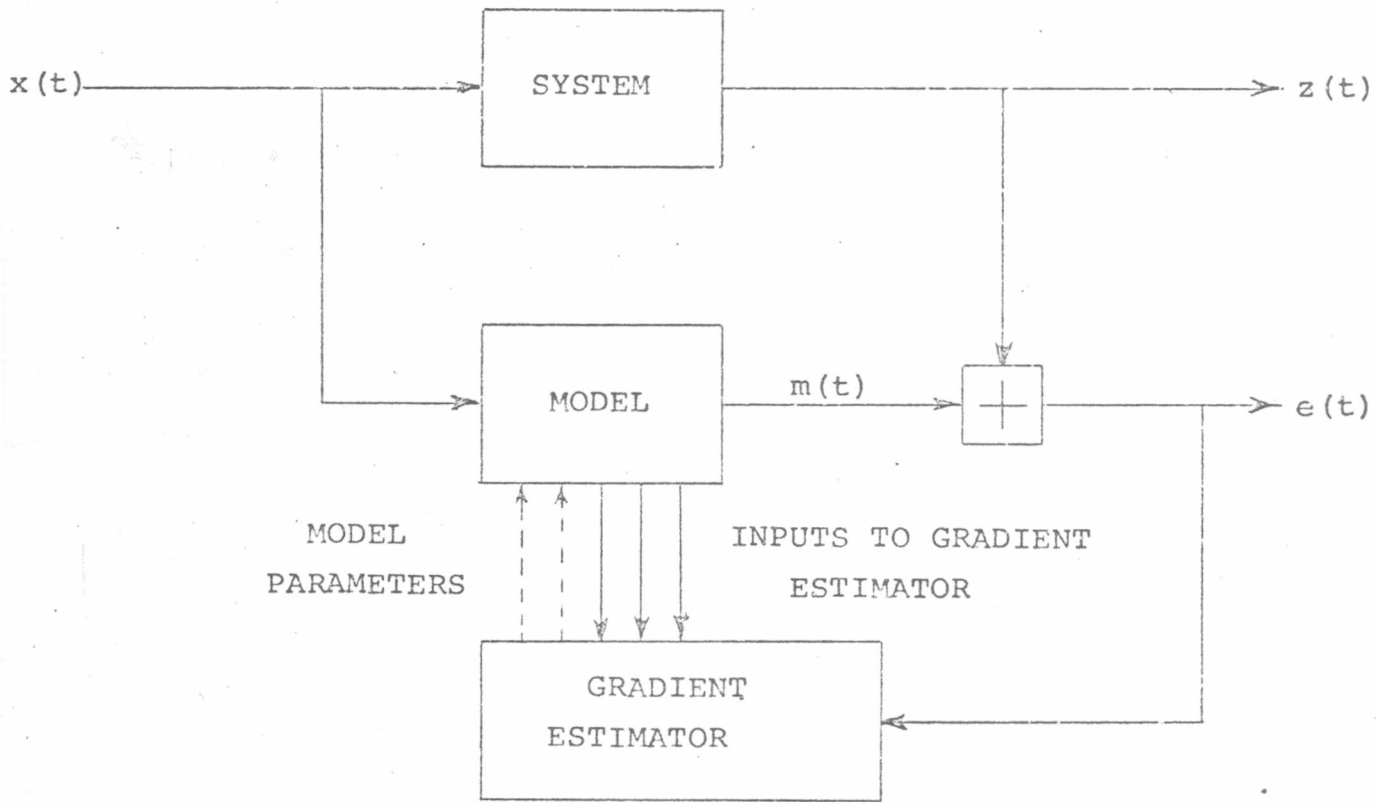
$$\ddot{m} + \alpha_{opt}(10\dot{m}) + \beta_{opt}(100m) = 10\sin 2\pi t \qquad \text{---- (6.8.21)}$$

The input, $x(t)$, can be sinusoidal, square wave or random, but not a step. Results from the analogue computer simulation for sinewave, squarewave and random excitations are presented in Figures (6.8(c)), (6.8(d)), and (6.8(e)), respectively. These results are presented in the form of a phase portrait with $\alpha(t)$ plotted against $\beta(t)$. A phase portrait of the corresponding stochastic computer simulation is illustrated in Figure (6.8(f)).

However, /

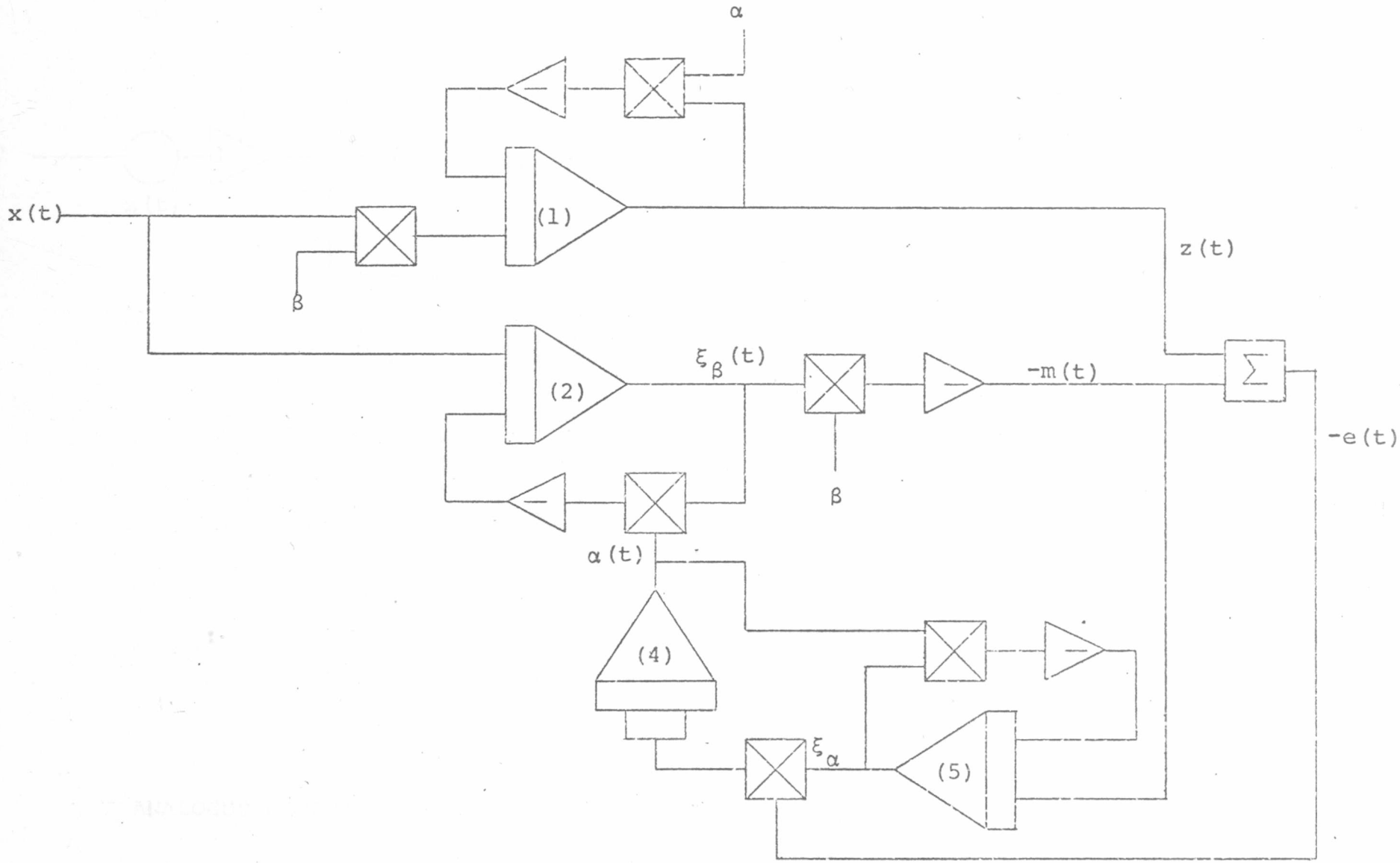
However, the corresponding stochastic simulation failed to converge to a steady state. This phenomenon can be explained by the fact that an underdamped system will oscillate when it is excited by noise. In this case the uncertainty in the outputs of the system and the model is ±10% of the dynamic range but the maximum error magnitude is only about 20% of the dynamic range so that there are no significant figures in the error measurement at all. One obvious way to overcome this problem is to use much larger counter sizes but this will drastically reduce the bandwidth of the identification circuit. The digital computer facilities available to this project do not permit one to simulate circuits using counter sizes of more than twelve bits because of the long computation times involved.

The experiment was repeated on an analogue computer but this time noise was injected into various parts of the circuit. Oscillations similar to those observed in the stochastic computer simulation were produced by the injected noise. Further tests showed that the circuit is very sensitive to slow variations of the injected noise. The above experiments on the analogue computer can only yield qualitative information about the stochastic simulation and this is because in a stochastic computer there is an intimate relationship between the gain of an integrator and the speed with which the variance occurs.



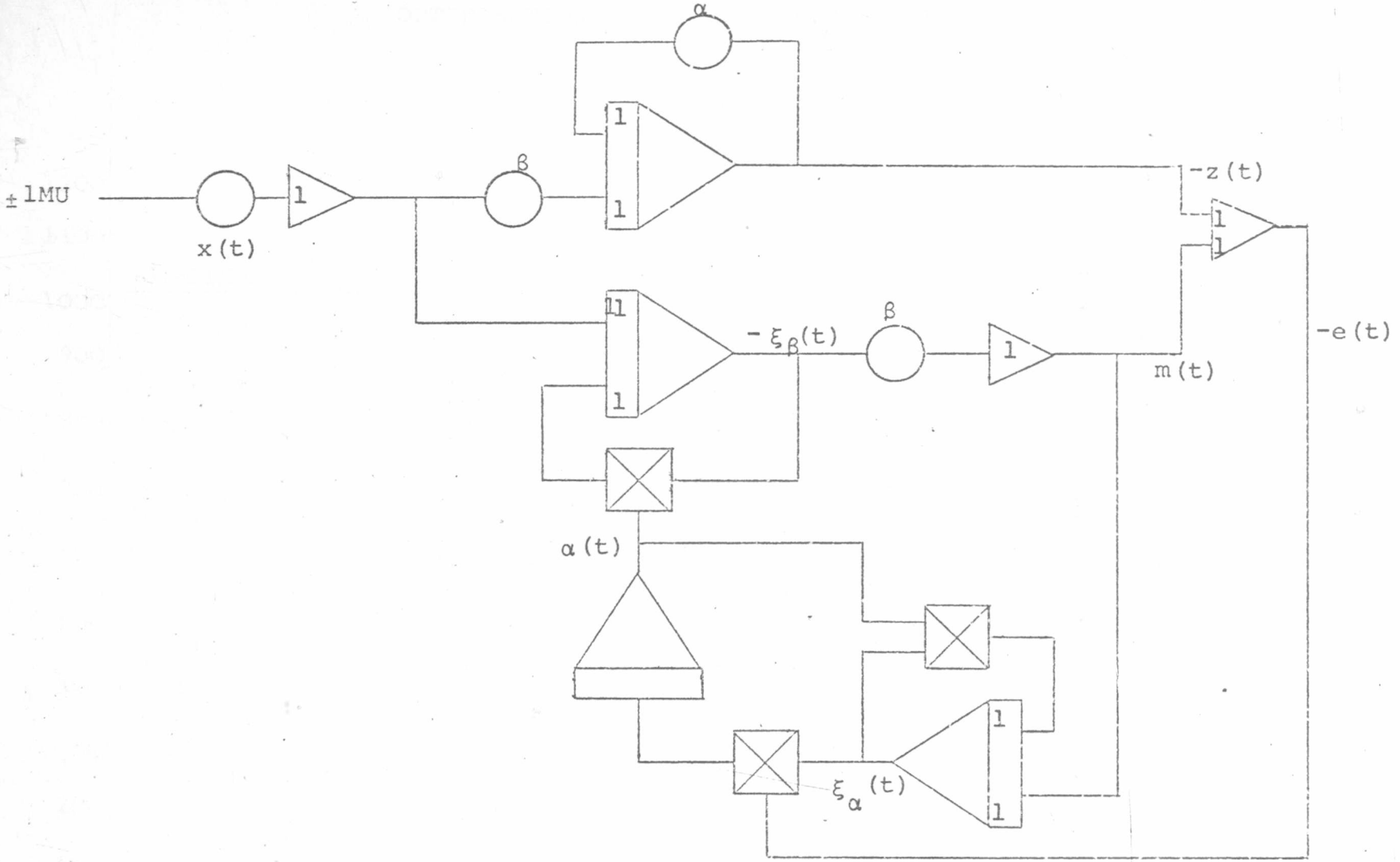

STEEPEST DESCENT METHOD OF SYSTEM IDENTIFICATION

FIG 6.0




STOCHASTIC COMPUTER CIRCUIT FOR FIRST ORDER SYSTEM IDENTIFICATION
IDENTIFICATION OF α

FIG 6.2(a)



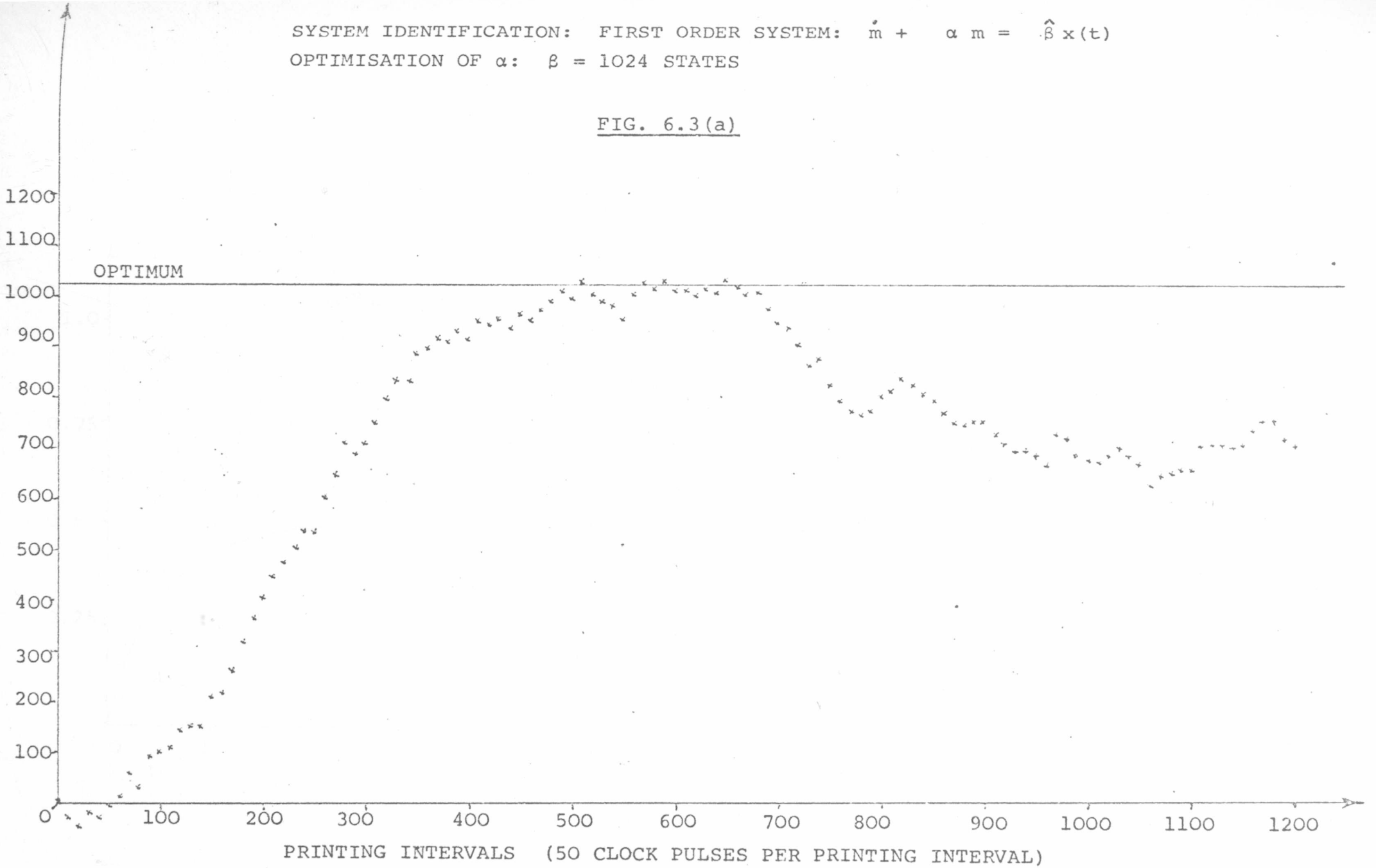

ANALOGUE COMPUTER CIRCUIT FOR FIRST ORDER SYSTEM IDENTIFICATION: IDENTIFICATION OF $\alpha$

FIG 6.2(b)

SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM: $\dot{m} + \alpha m = \hat{\beta} x(t)$

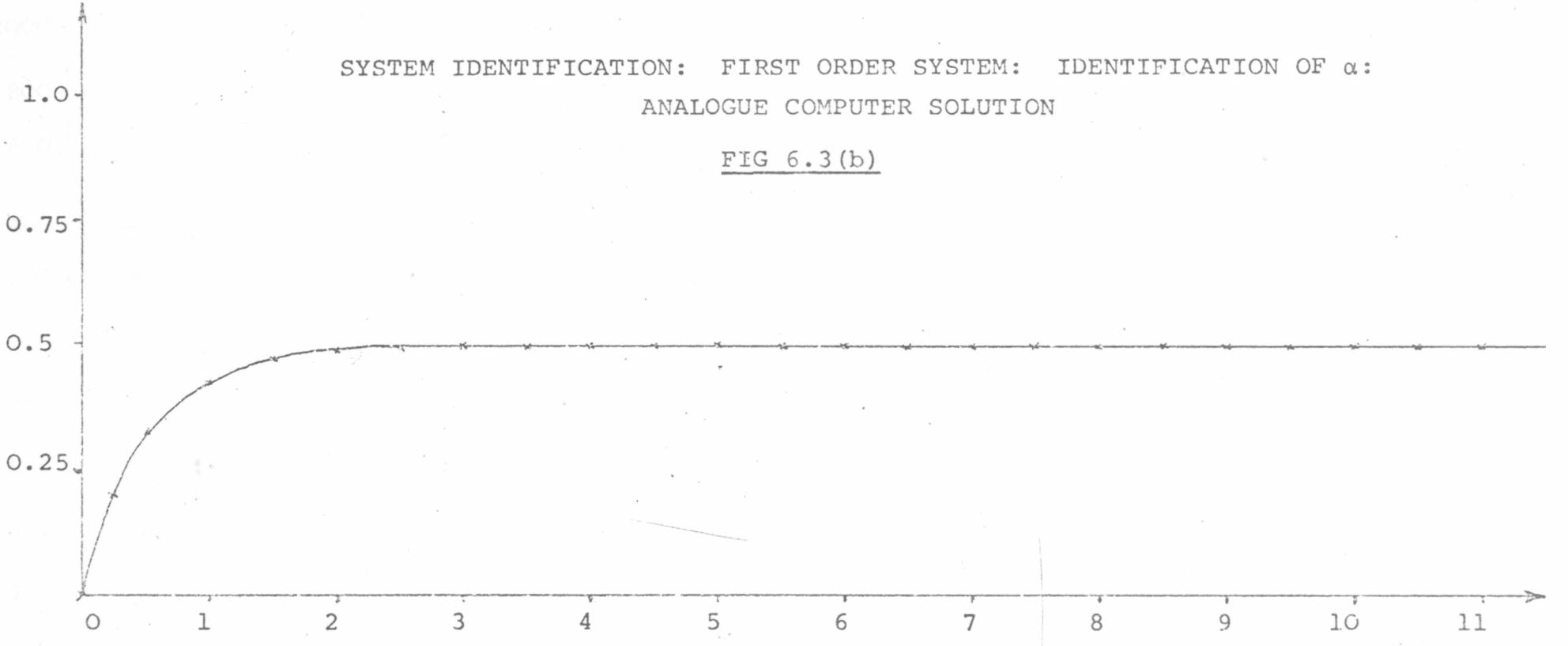
OPTIMISATION OF $\alpha$: $\beta$ = 1024 STATES

FIG. 6.3(a)



SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM: IDENTIFICATION OF $\alpha$:
ANALOGUE COMPUTER SOLUTION

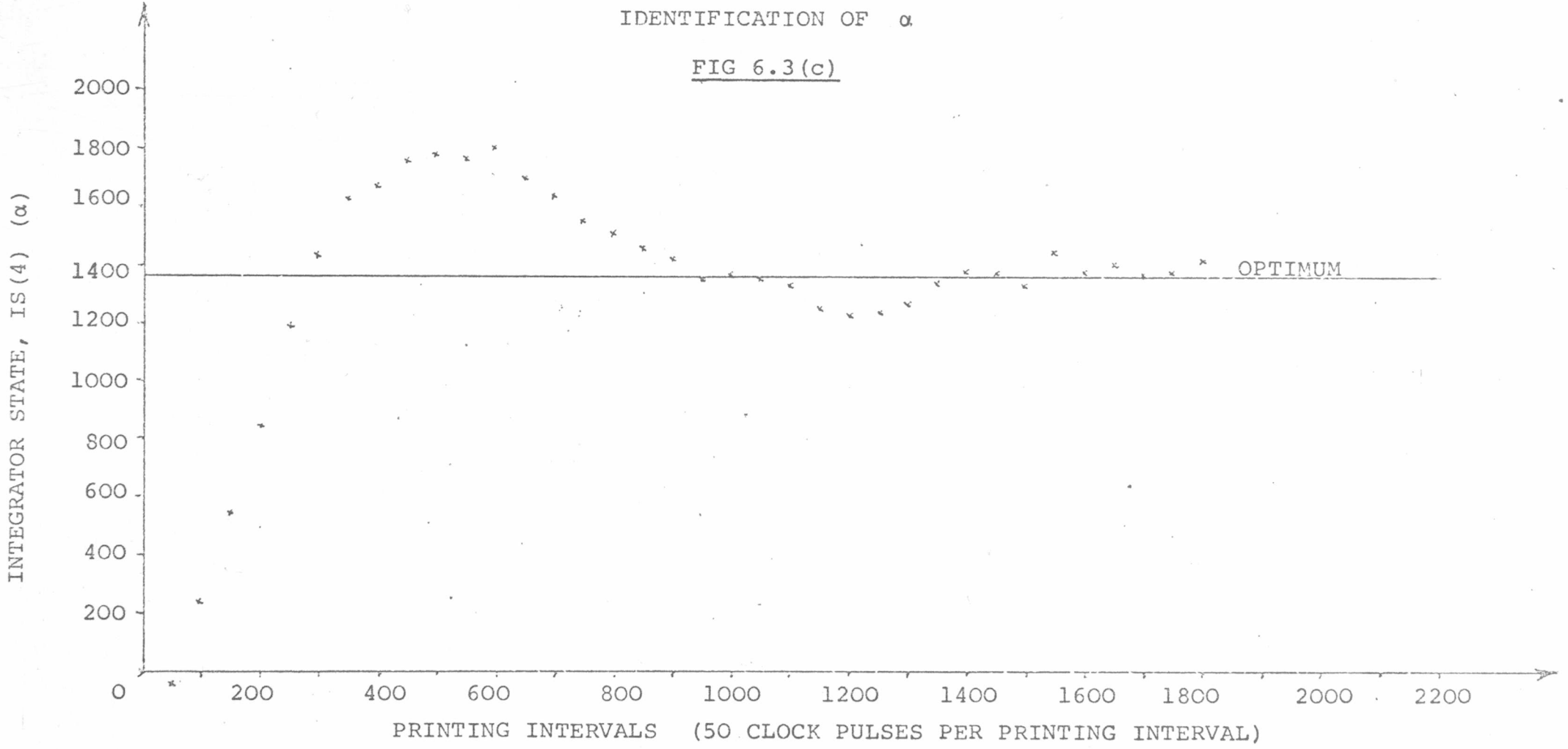
FIG 6.3(b)



SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM $\dot{m} + k\alpha m = k\beta x$

IDENTIFICATION OF $\alpha$

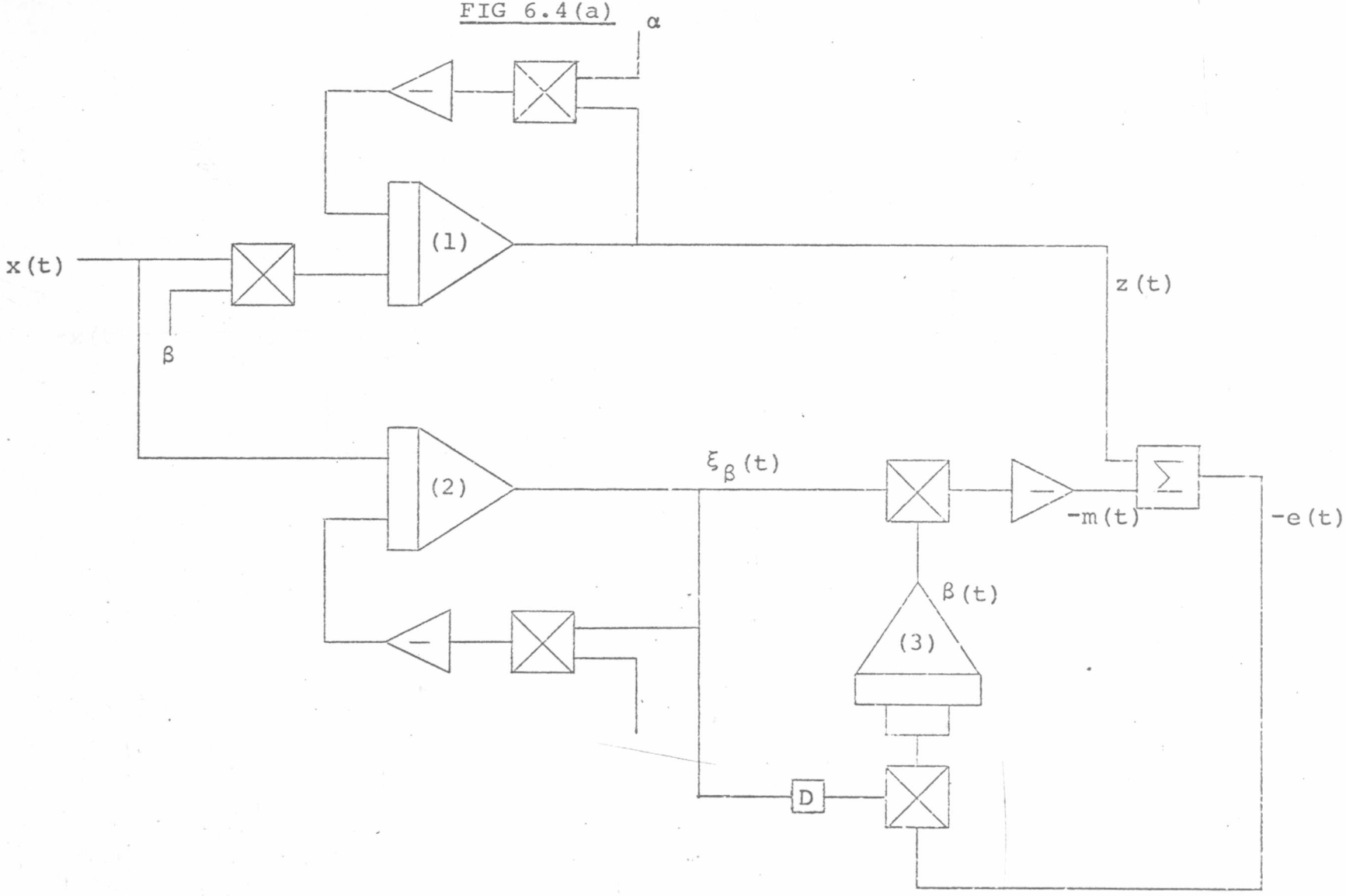
FIG 6.3(c)



SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM $\dot{m} + \hat{\alpha}m = \beta x$

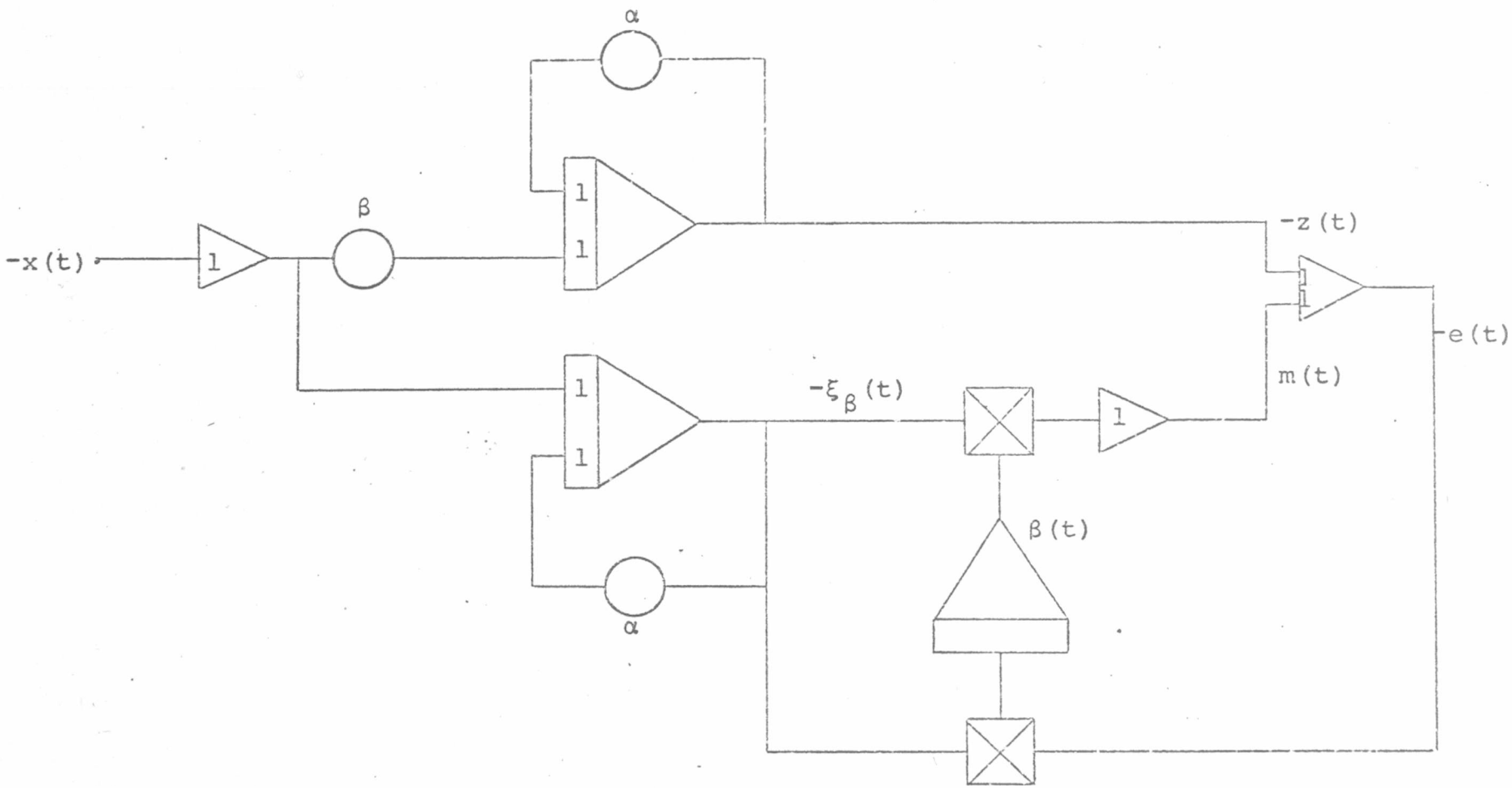
IDENTIFICATION OF $\beta$ USING A STOCHASTIC COMPUTER CIRCUIT

FIG 6.4(a)



ANALOGUE COMPUTER CIRCUIT FOR FIRST ORDER SYSTEM IDENTIFICATION:
IDENTIFICATION OF β

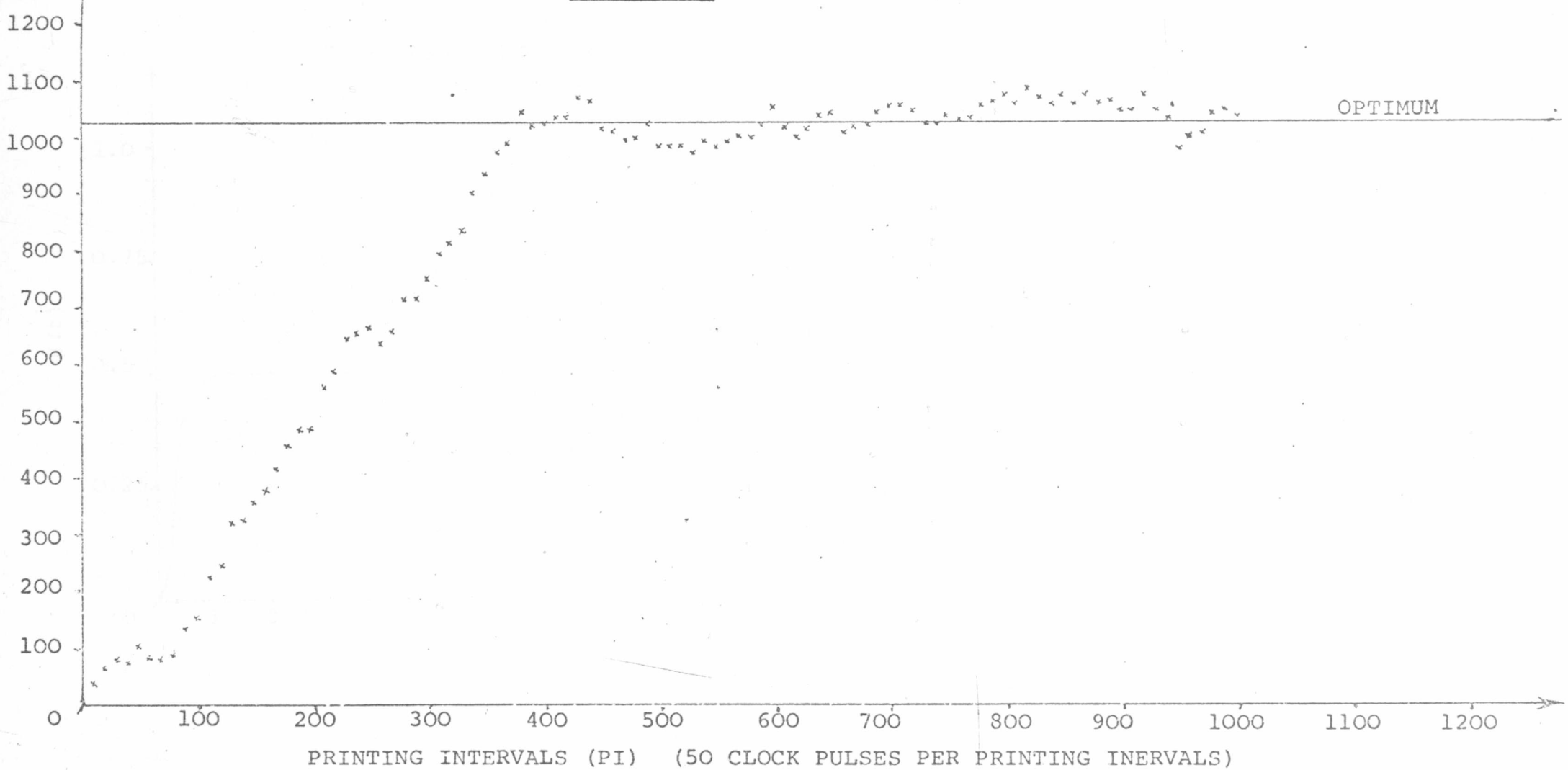
FIG 6.4(b)



SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM: $m + \hat{\alpha}m = \beta x(t)$

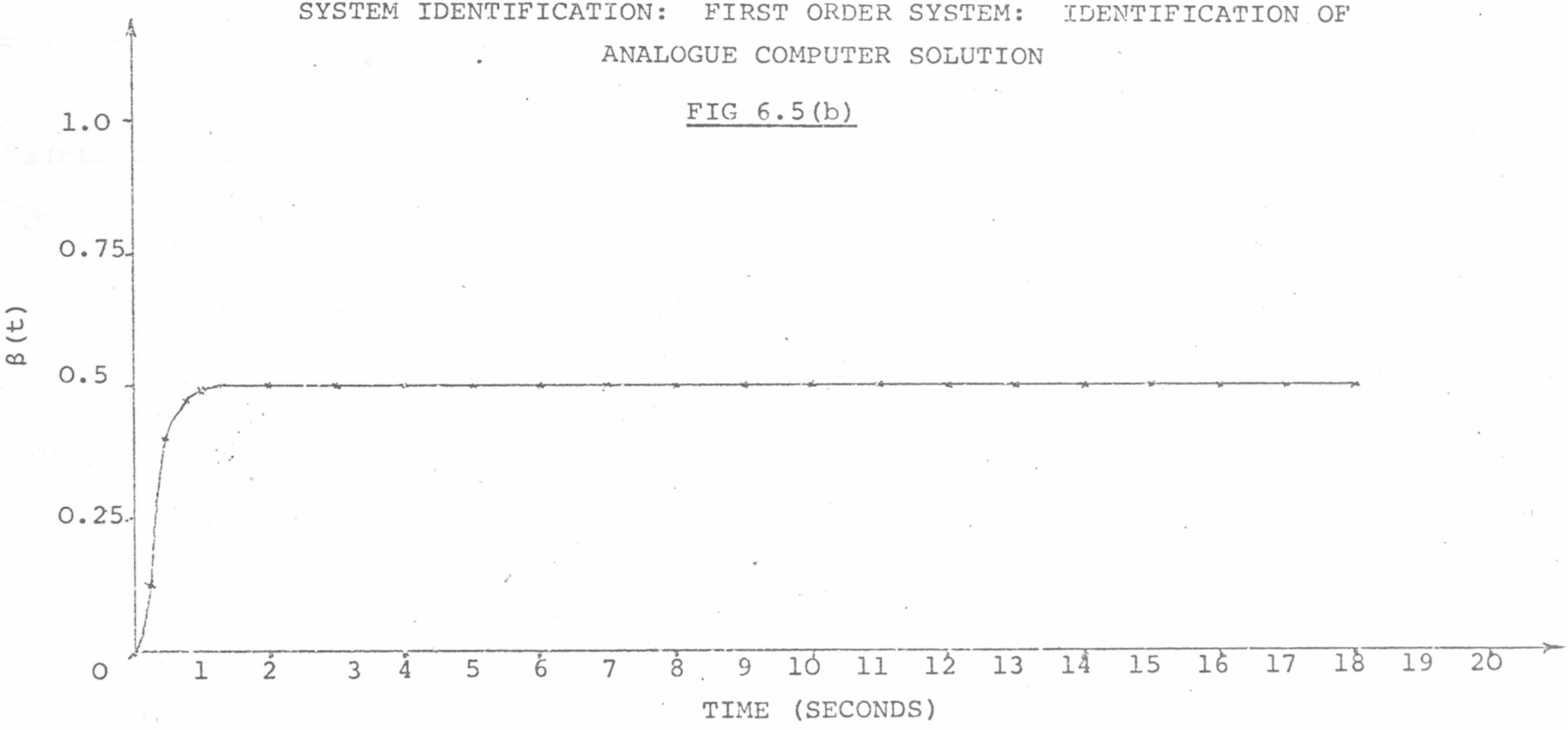
OPTIMISATION OF $\beta$: $\alpha \equiv 1024$ STATES

FIG 6.5(a)



SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM: IDENTIFICATION OF ANALOGUE COMPUTER SOLUTION

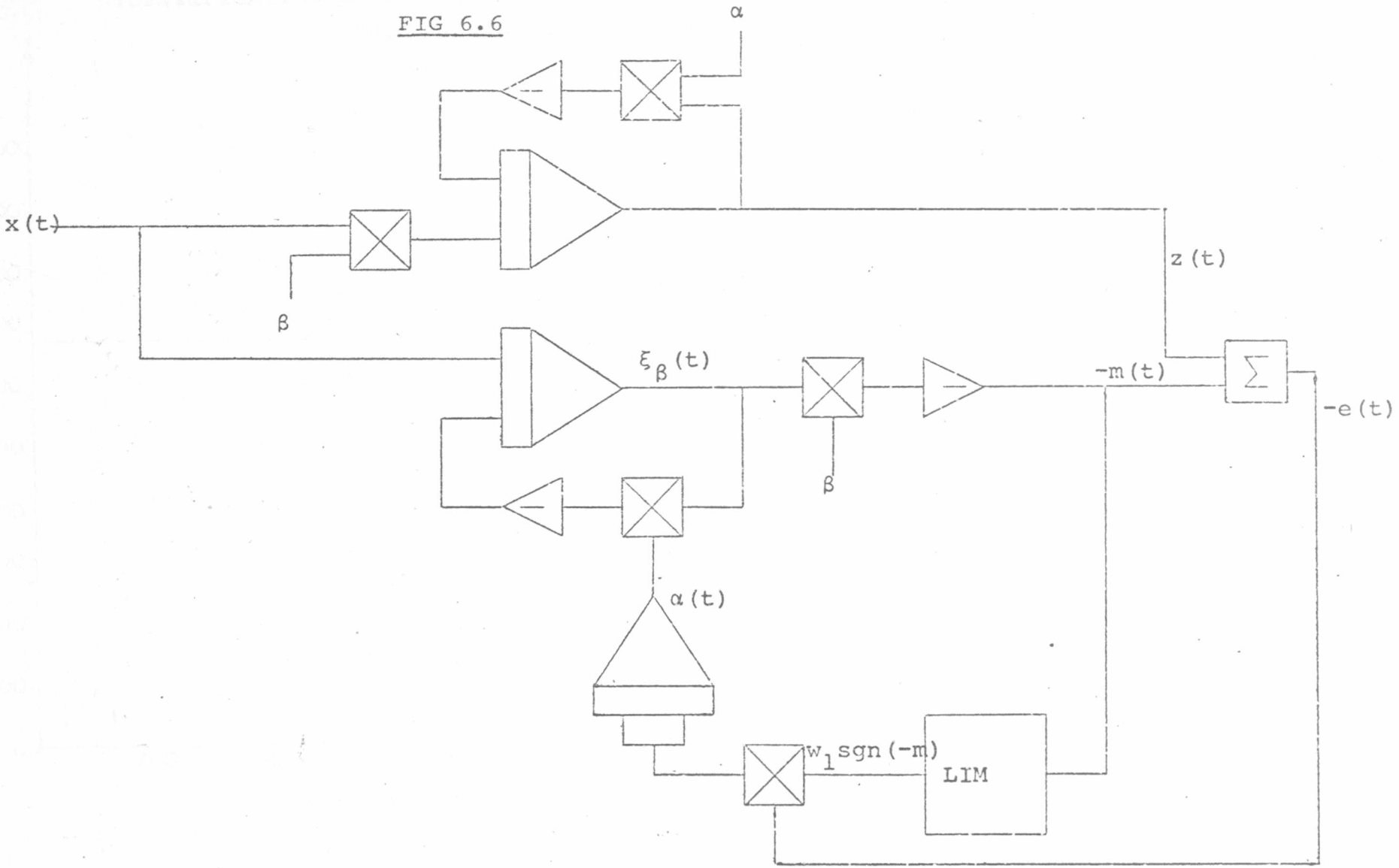
FIG 6.5(b)



SYSTEM IDENTIFICATION: FIRST ORDER SYSTEM $\dot{m} + \alpha m = \beta x$

THIS AVOIDS THE USE OF SENSITIVITY EQUATION

FIG 6.6



IDENTIFICATION OF THE PARAMETERS OF A FIRST ORDER SYSTEM: EVALUATION OF x

THIS METHOD USES sgn(-m) INSTEAD OF $\xi_x$

FIG 6.7

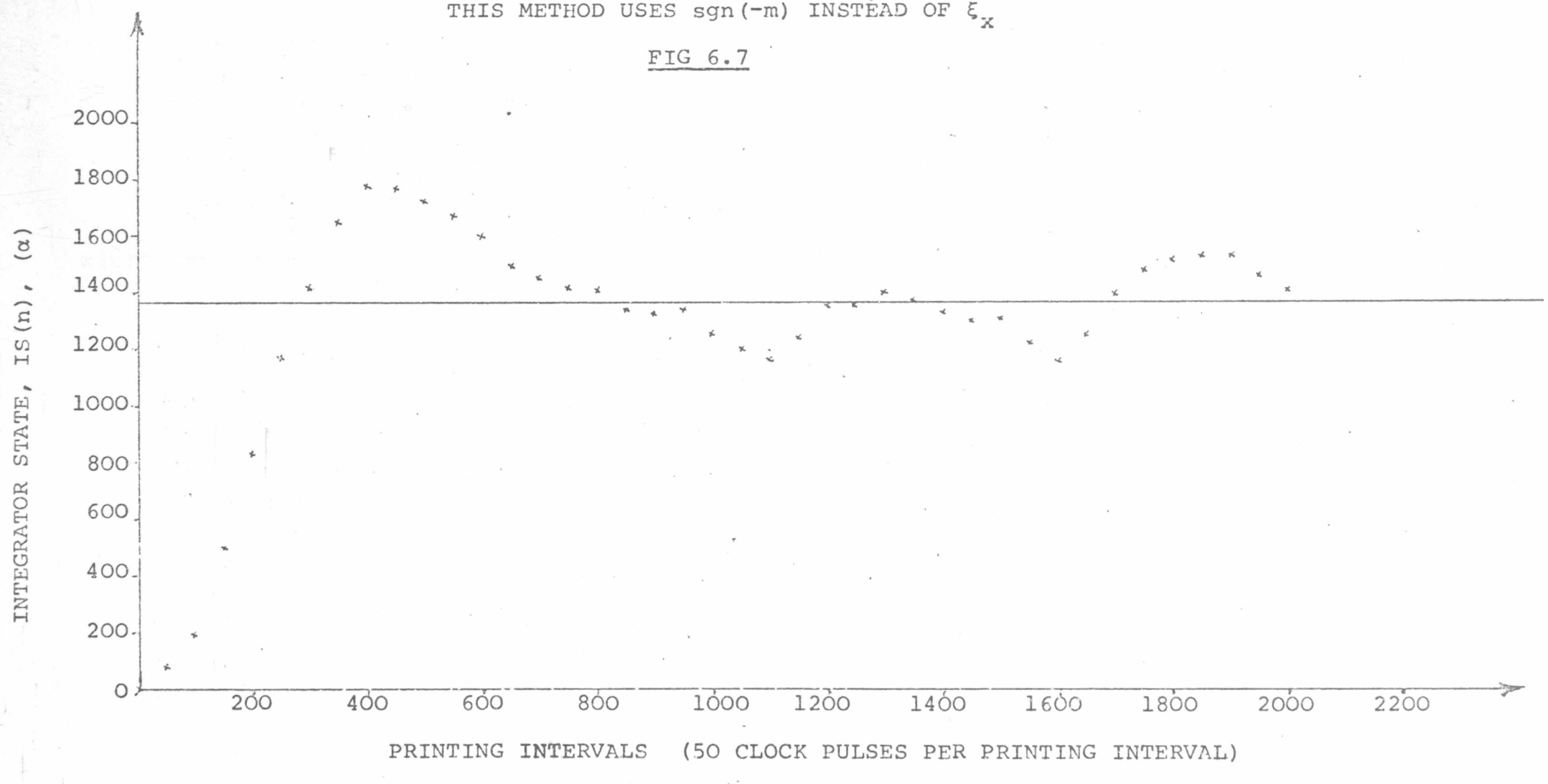
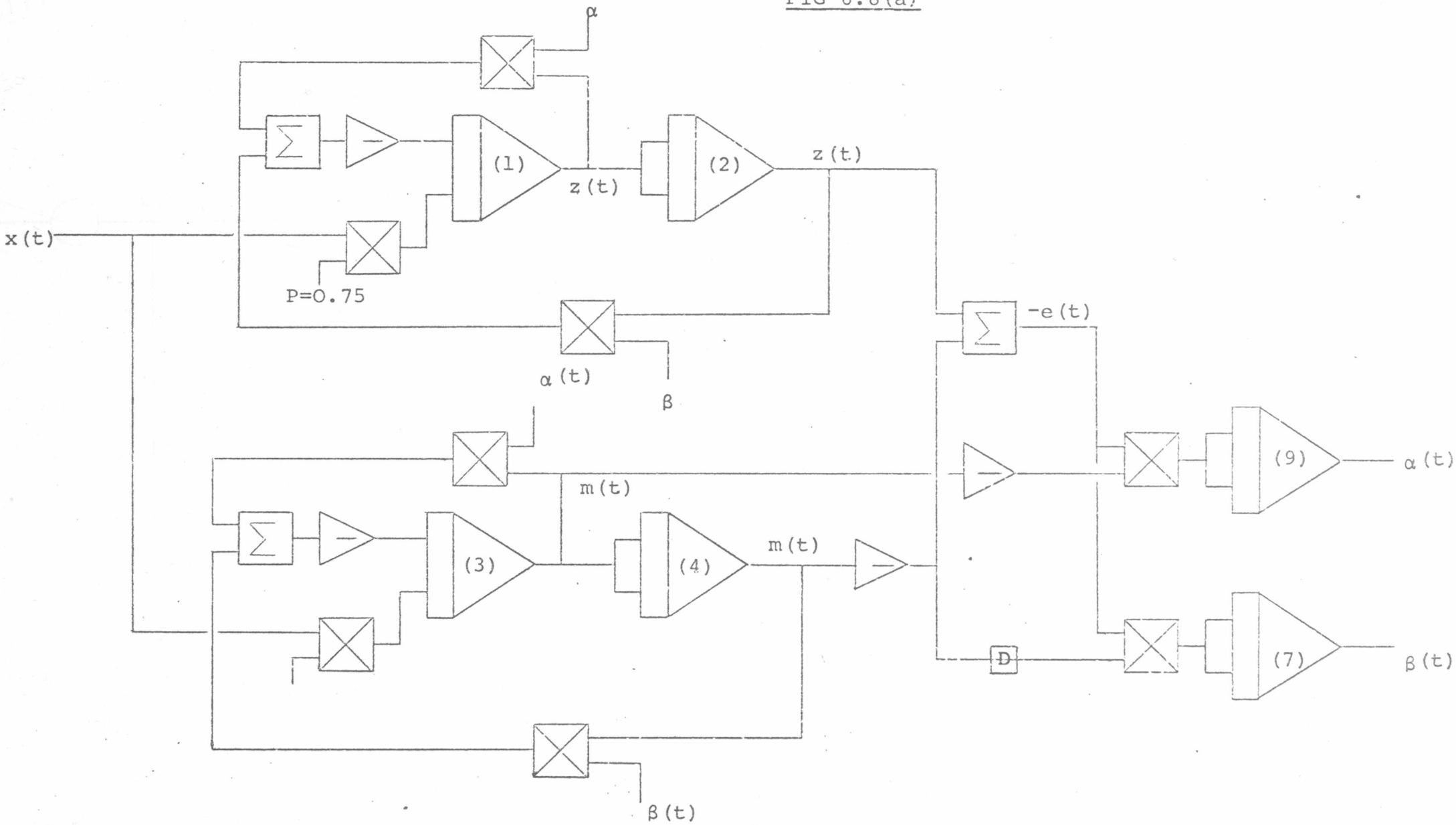
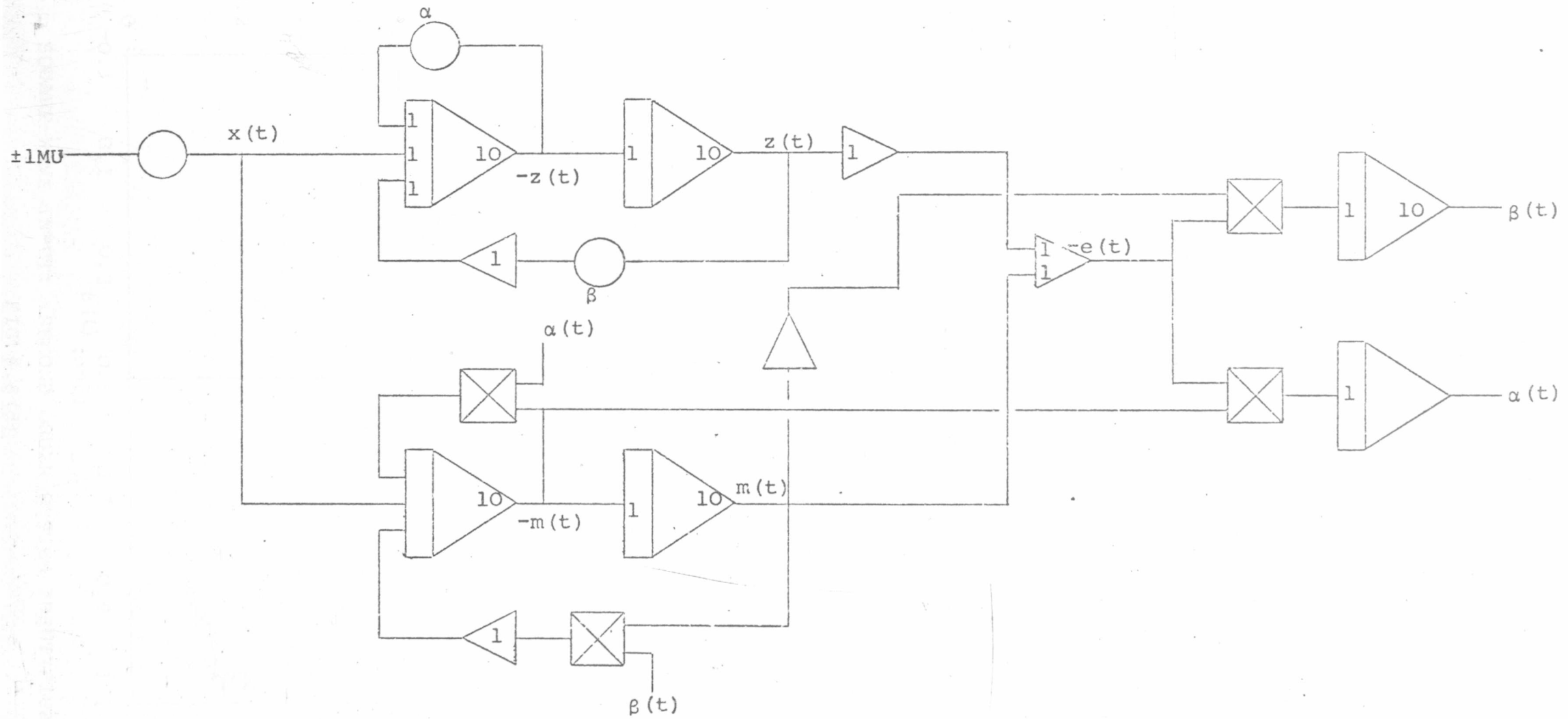


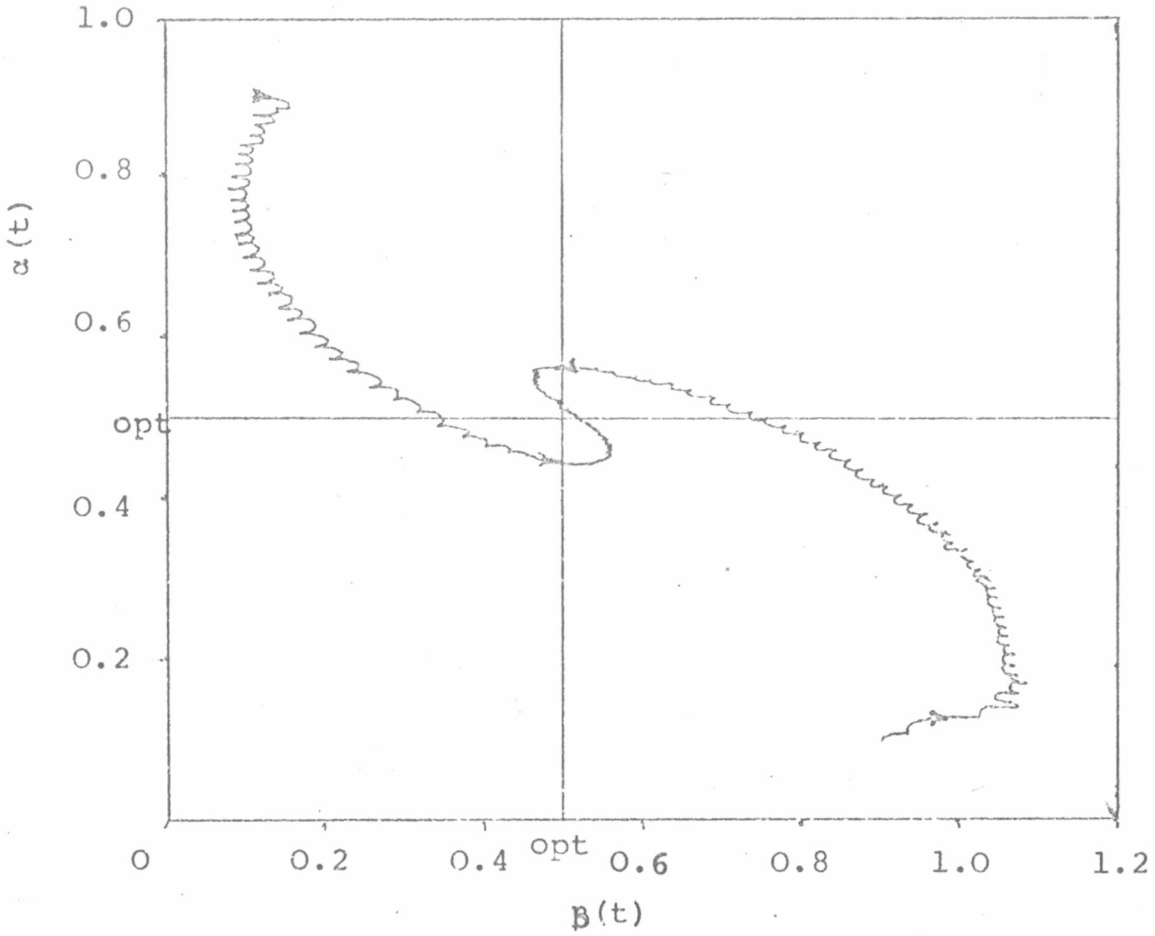
FIG 6.8(a)



IDENTIFICATION OF THE PARAMETERS OF A SECOND ORDER SYSTEM USING AN ELECTRONIC ANALOGUE COMPUTER

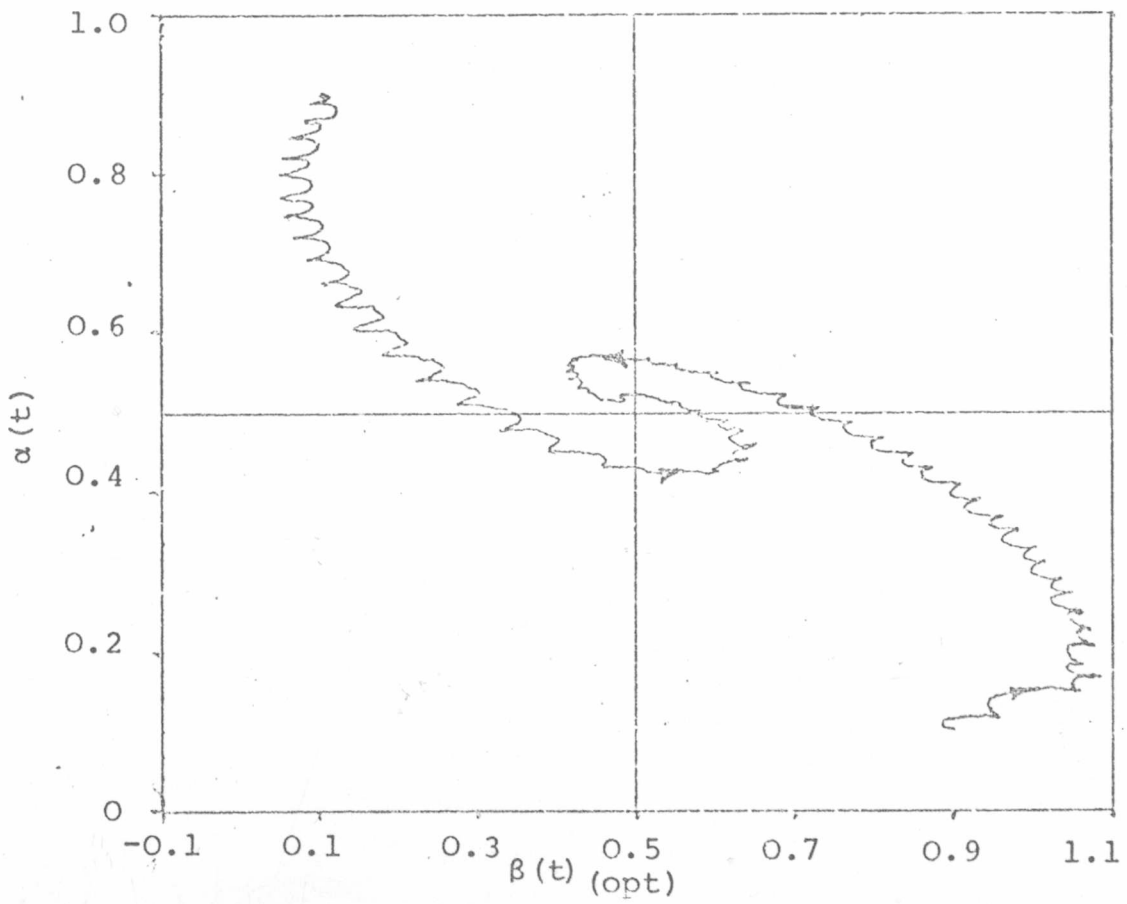
FIG 6.8(b)





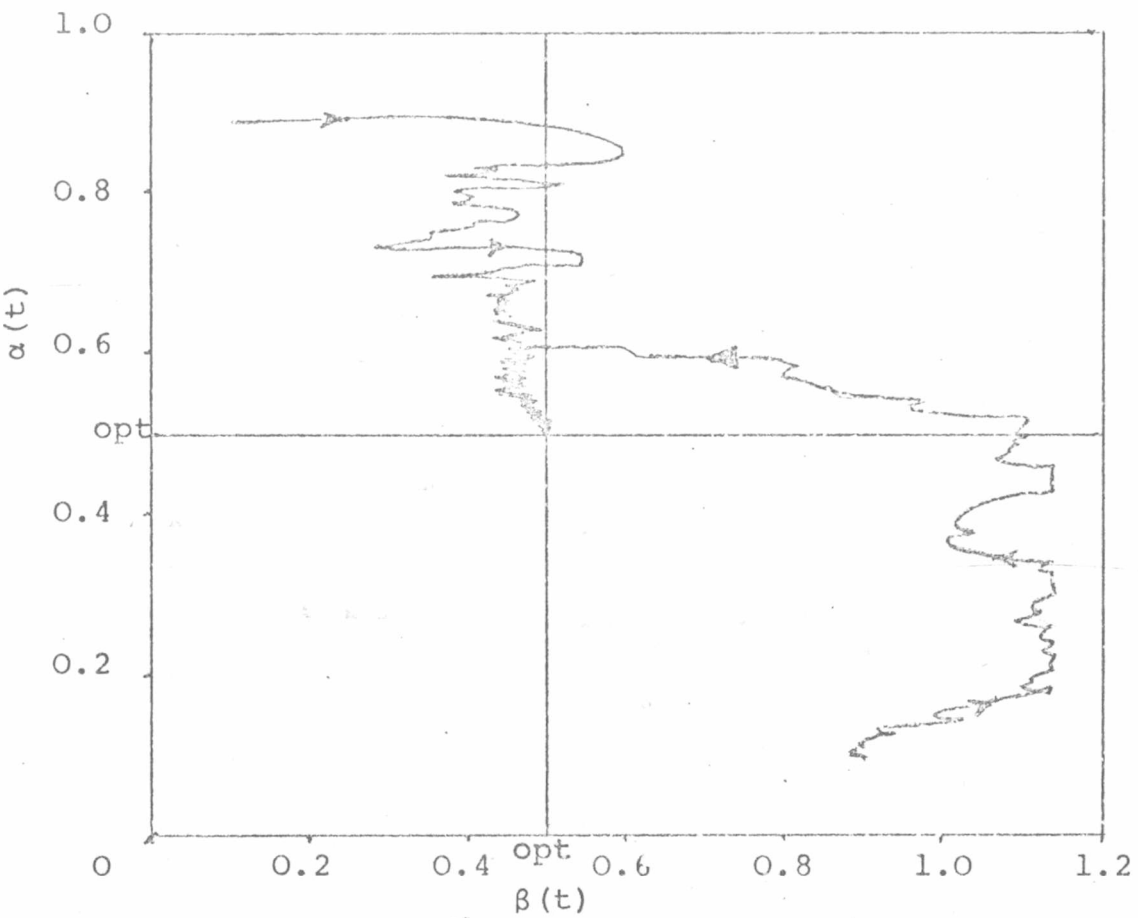

SINUSOIDAL INPUT: SECOND ORDER SYSTEM IDENTIFICATION

FIG 6.8(c)



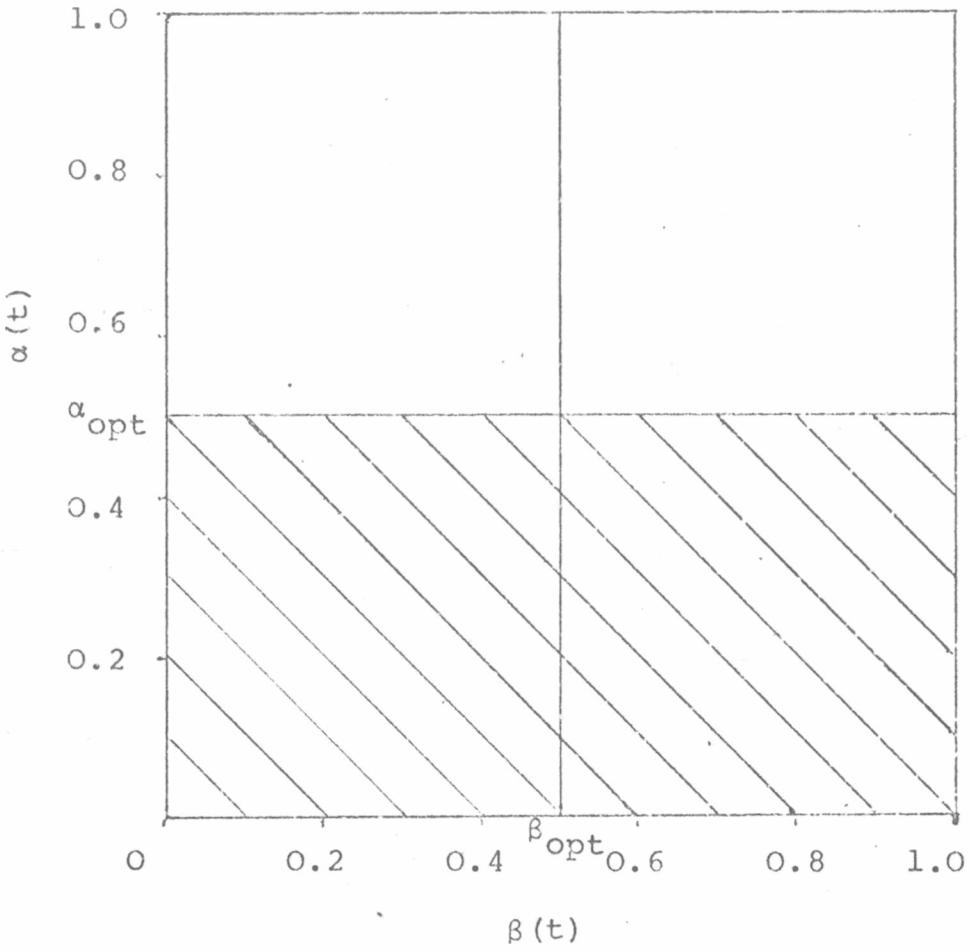

SQUARE-WAVE INPUT: SECOND ORDER SYSTEM IDENTIFICATION

FIG 6.8(d)




NOISE INPUT: SECOND ORDER SYSTEM IDENTIFICATION

FIG 6.8(e)




SECOND ORDER SYSTEM IDENTIFICATION

IDENTIFICATION OF $\alpha$ AND $\beta$. USING A STOCHASTIC COMPUTER CIRCUIT. AREA SHADED INDICATES THE RANGE OF VALUES $\alpha(t)$ AND $\beta(t)$ MAY TAKE.

FIG 6.8(f)

# CHAPTER 7

(This Chapter provides a summary of the work carried out so far and presents suggestions for future work on stochastic computing.)

## 7.0 Conclusions

The FORTRAN IV simulation programmes developed over the past two years permit one to follow the behaviour of a synchronous sequential network from one clock pulse to the next. Using this technique it is possible to simulate the behaviour of complicated circuits over long time periods. Essentially, these programmes give a method of stochastic computation which is independent of the clock frequency. There is also the advantage that one can examine a stochastic computation in detail without having to consider the effects of an output interface. A twelve bit ADDIE forms the output interface of the stochastic computer, but it has a bandwidth of about 250 rad/sec at a clock frequency of 1 MHz, so that rapidly changing phenomena in a circuit may be completely filtered out at the output stage. Thus, these programmes provide a means of closely checking problems run on a stochastic computer which has recently been built at RGIT.

## 7.1 First and Second Order Systems

In Chapter 3 some simple function generators were investigated using programmes based on simple mathematical models of the basic stochastic operators. These functions included division, square-root extraction and second-order exponential smoothing. Simple mathematical analyses were used to predict the behaviour of these systems based on the somewhat naive assumption that one could consider probabilities without their associated random variances. However, second order underdamped systems exhibited behaviour which was not explained by the simple mathematics used. A physical explanation of the small oscillations observed on the output /

output of a second order ADDIE was given which stated that randomly excited underdamped systems cannot attain a steady-state output. On the other hand, first order and overdamped second order systems did converge to steady-state outputs when excited by step inputs. A sine-cosine generator was simulated but this showed amplitude instability owing to the random variances of the two integrators used in the circuit.

## 7.2 Solution of Linear Equations and Matrix Inversion

In Chapter 4 some well known algorithms for solving sets of linear equations were implemented using a stochastic computer circuit. Typical simulations of problems run on these circuits yielded results which were within 5% of full scale. Both algorithms used showed evidence of 'hunting' or drifting round their optimum operating points. However, it was difficult to estimate errors from the experimental curves because the simulations were not continued for long enough to enable meaningful statistical tests to be performed on the results. Each set of curves required four hours of computing time on an 'ELLIOT 4120' digital computer to reach a steady state and this is only equivalent to $65 \times 10^3$ clock pulses. Any statistical tests would have to be carried out over hundreds of thousands of clock pulses. The simulations of the error criterion and steepest descent methods show that these methods are stable and convergent. The optimisation times quoted in Chapter 4 cannot be said to be representative of all third order systems since speed of convergence is governed by the eigenvalues of the coefficient matrix. A different coefficient matrix will yield different convergence times.

7.3 /

## 7.3 Linear Programming

The simulation of a linear programming problem demonstrated that it is possible to generate threshold switching functions in a stochastic computer. However, it is not possible to obtain a very fast switching action since it takes a relatively long time to estimate the values of generating probabilities using an ADDIE which is used as the basis for the switch. Thus there is a time lag between the input being applied to the ADDIE and the output taking on the value of the input. Therefore there is the possibility that the moving point may have entered the restricted region but the ADDIEs may not measure this immediately and the system may become unstable. To obtain good switching the ADDIE must employ shorter counters than those used in the integrators. The use of small counter lengths in the switches will mean that very small errors cannot be measured with the result that there will be poor convergence to optimum. Improved results may be obtained if twelve bit ADDIEs are used with twenty-four bit integrators. The test problem chosen was a maximisation and the curves yielded values which were within 5% of full scale compared with the theoretical solution. As with the linear equation problem discussed in Chapter 4, no specific estimations can be made about the speed of computation so that different problems will have different convergence times. The choice of scaling will affect the speed with which the algorithm searches the constraints assuming that the problem is not degenerate in the first place.

## 7.4 System Identification

Stochastic computer circuits which could identify the parameters of first and second order systems were devised in Chapter 6. The circuits for first order system identification worked successfully and identified the /

the system parameters exactly. A difficulty in scaling was pointed out and a procedure for preventing limiting was explained. The first order circuit also demonstrated the need to include delays as stochastic operators in their own right since their presence in a circuit can avoid cross-correlation effects.

Although the results from the second order identification were disappointing they demonstrated the way in which errors can arise in stochastic computation. Random variance in underdamped stochastic circuits give rise to small oscillations in the output probabilities and these perturbations may lead to catastrophic errors in a large interconnected network. These oscillations can be reduced by using integrators with much larger bit capacities to reduce the effective random variance in the circuit. Since the stochastic computer operations are analogues of the system being simulated, this computer must reflect the behaviour of the problem under noisy conditions.

## 7.5 Digital Stochastic Computer

A digital stochastic computer has recently been built at Robert Gordon's Institute of Technology, Aberdeen. It comprises ten integrators, twelve comparators, nine multipliers, six summers and eight inverters. However, up to forty devices can be used at any one time. Another research student has devised an automatic patching system and this enables a programmer to interconnect the stochastic operators via a teletype or a visual display unit with a keyboard. A PDP/8E minicomputer is used in conjunction with the stochastic computer to control all input/output operations. Other team members have devised procedures for scaling integrators and loading initial conditions into these devices before a problem is run. A digital-to-analogue converter permits the stochastic computer to control a small plant. The complete system is illustrated in Figure 7.5.

7.6 /

## 7.6 Future Work

It is intended to repeat all the simulations carried out so far on the stochastic computer described above to confirm the results presented in this thesis. However, before Linear Programming problems can be implemented on the stochastic computer, practical designs for threshold switches will have to be realised and their properties investigated.

The investigations carried out so far have indicated that the stochastic computer may not be as good as an electronic analogue computer in solving differential equations because of the presence of random variance. The stochastic computer may be more successfully used in the simulation of processes which are inherently stochastic. Many problems in operations research can be classified as 'MARKOV CHAINS'.[41,42,43,45,46] This idea can be exemplified if we consider the following problem. A number of computers are to be serviced by one technician and, to help with planning, the service company wants to know how much time the technician is going to spend at base and at each computer installation, assuming he started out from base. We wish to find a row vector, $\nu^{(r)}$, the elements of which are the probabilities of being at each location at some time, r. The term 'r' can represent the numbers of days or half days during which the technician is on duty. A probability transition matrix, P, can be defined which describes the probability of moving from one installation to another or staying in the present location. At any time we can predict the probability of being in a particular location (say the jth), ie,

$$\pi_j^{(r+1)} = \pi_1^{(r)} p_{1j} + \pi_2^{(r)} p_{2j} + \ldots + \pi_j^{(r)} p_{rj} + \ldots + \pi_n^{(r)} p_{nj} \quad \text{---- (7.6.1)}$$

$$\Rightarrow \pi_j^{(r+1)} = \sum_{k=1}^{n} n_k^{(r)} p_{kj}, \quad j = 1,2,\ldots n \quad \text{---- (7.6.2)}$$

or /

$$\text{or} \quad \underline{\nu}^{(r+1)} = \underline{\nu}^{(r)} P \qquad \text{---- (7.6.3)}$$

where

$$\underline{\nu}^{(r)} = [\pi_1^{(r)}, \pi_2^{(r)}, \ldots, \pi_n^{(r)}] \qquad \text{---- (7.6.4)}$$

Aging characteristics can be introduced into the system by making the transition probabilities vary with respect to r. Then we have:

$$\underline{\nu}^{(r+1)} = \underline{\nu}^{(r)} P(r) \qquad \text{---- (7.6.5)}$$

Since the technician must be at some location,

$$\sum_{k=1}^{n} \pi_k^{(r)} = 1.0 \qquad \text{---- (7.6.6)}$$

and after half a day he has to move somewhere else or stay where he is, hence:

$$\sum_{i=i_j}^{n} P_{ij}^{(r)} = 1.0 \qquad \text{---- (7.6.7)}$$

We can derive a recursive formula for $\underline{\nu}^{(r+1)}$ in the following way:

$$\underline{\nu}^{(1)} = \underline{\nu}^{(0)} P(0) \qquad \text{---- (7.6.8)}$$

$$\underline{\nu}^{(2)} = \underline{\nu}^{(1)} P(1) = \underline{\nu}^{(0)} P(0) P(1) \qquad \text{---- (7.6.9)}$$

$$\underline{\nu}^{(3)} = \underline{\nu}^{(2)} P(2) = \underline{\nu}^{(0)} P(0) P(1) P(2) \qquad \text{---- (7.6.10)}$$

$$\vdots$$

$$\underline{\nu}^{(r+1)} = \underline{\nu}^{(r)} P(r) = \underline{\nu}^{(0)} \prod_{i=0}^{r} P(i) \qquad \text{---- (7.6.11)}$$

If each of the $P^{(i)}$ are identical[45] and have at least one eigenvalue equal to unity then:

$$\lim_{r \to \infty} \prod_{i=0}^{r} P^{(\ell)} = Q \qquad \text{---- (7.6.12)}$$

and /

and

$$\lim_{r\to\infty} \underline{\nu}^{(r+1)} = \underline{\nu}^*, \qquad \text{---- (7.6.13)}$$

and we have

$$\underline{\nu}^* = \underline{\nu}^* P = \nu^{(0)} Q \qquad \text{---- (7.6.14)}$$

or

$$QP = Q, \qquad \text{---- (7.6.15)}$$

where the rows of Q are identical.

The Markov chain is then stationary and ergodic and the final vector $\underline{\nu}^*$ is independent of $\underline{\nu}^{(0)}$. However, equation (7.6.11) need not represent a stationary, ergodic process and can be used to model catastrophic events, eg, the cancellation of a maintenance contract in the problem described above.

If the eigenvalues of $P^{(r)}$ are distinct then modal[41] matrix techniques can be used to analyse the transient behaviour of $\underline{\nu}^{(r)}$, but if $P^{(r)}$ is doubly stochastic (both rows and columns of $P^{(r)}$ sum to unity), the eigenvalues are indistinct and z-transform methods have to be used.

For modelling non-stationary processes $P^{(r)}$ can be represented by:

$$P^{(r)} = C + D^{(r)}, \quad 0 \leqslant p_{ij}{}^{(r)} \leqslant 1.0, \quad 0 \leqslant 1d_{ij}{}^{(r)} \leqslant 1.0 \qquad \text{---- (7.6.16)}$$

where the elements of C are constant and those of $D^{(r)}$ vary from step to step. The rows of C sum to unity while those of $D^{(r)}$ sum to zero. A digital computer programme (listed in APPENDIX 7A) was written in BASIC to illustrate this idea. The numerical example chosen in this case was:

$P^{(r)}$ /

$$P^{(r)} = \begin{bmatrix} 0.4 & 0.35 & 0.25 \\ 0.2 & 0.6 & 0.2 \\ 0.3 & 0.3 & 0.4 \end{bmatrix} + e^{-Gr} \begin{bmatrix} 0.4 & -0.3 & -0.1 \\ -0.1 & 0.3 & -0.2 \\ 0.2 & 0.2 & -0.4 \end{bmatrix}$$

---- (7.6.17)

with $\underline{\nu}^{(0)} = [1,0,0]$ ----- (7.6.18)

and $\underline{\nu}^{(r)}$ converges to the row vector

$$[0.2835, 0.4490, 0.2675]$$

The $d_{ij}^{(r)}$ can be any function except that

$$\sum_{k=1}^{n} d_{ik}^{(r)} = 0 \qquad \text{---- (7.6.19)}$$

Hence one of the $d_{ik}^{(r)}$ must be expressed in terms of the other (n-1) row elements.

The idea of time weighting the transient probability matrix will allow programmers to produce more realistic models of stochastic processes. It is hoped that these algorithms can be used to model queuing systems,[37,38] to aid weather forecasting, and to help solve problems in nuclear physics.

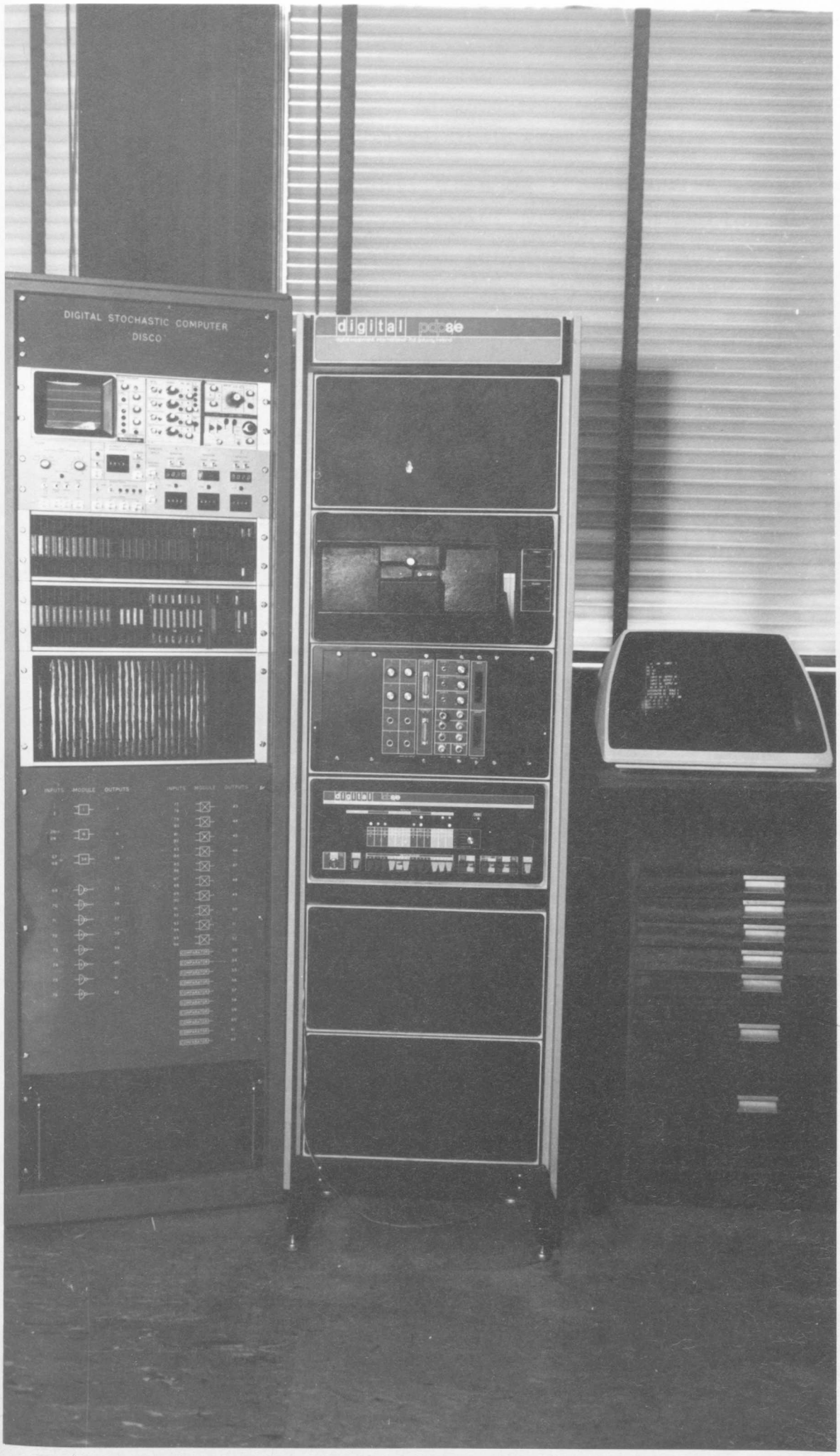



FIGURE 7.5

APPENDIX 1A

I The Variance of a Binomial Distribution

Since generating probabilities of random pulse sequences are used to represent real variables, information about these variables can only be achieved by measuring these probabilities. Probability can only be accurately messured over an infinite period of time. The shorter the sample length the more inaccurate the estimate will be of the probability. If the output of the input interface is a stationary Bernoulli sequence, and it is regularly sampled, all the samples will be distributed binomially[3] about a mean value, p, which is the unbiased estimate of the sequences generating probability. Let N be the sample size then,

$$\mu = Np$$

which is the mean number of ON pulses, with variance,

$$\sigma^2 = Np(1-p) \qquad \text{---- (1A.1)}$$

$$\Rightarrow \quad \sigma = \sqrt{Np(1-p)} \qquad \text{---- (1A.2)}$$

Normalising the standard deviation, $\sigma$, to a base N,

$$\sigma_N = \sqrt{\frac{p(1-p)}{N}} \qquad \text{---- (1A.3)}$$

The maximum standard deviation occurs when

$$\frac{d\sigma_N}{dp} = 0$$

and this occurs when $p = 1/2$.

Thus when the generating probability is 0.5 we have maximum variance and so the greatest uncertainty in estimating values. If a bipolar mapping is used we have zero represented by a probability of 0.5 so that maximum uncertainty is associated with minimum absolute value of a variable. For example, $p = 0.5$ and $N = 4096$, then,

$$\sigma^2 = 2^{10}$$

and /

and

$$\sigma = 32 \text{ states.}$$

Thus for a large enough sample 68.8% of all samples lie in the range (2016, 2080).

To obtain an estimate of the errors due to random variance it is more meaningful to express the standard deviations in terms of per unit analogue quantities rather than in terms of generating probabilities.

Let

$$p_i = \tfrac{1}{2} + \tfrac{1}{2} \left(\frac{E_i}{V}\right) \qquad \text{---- (1A.4)}$$

$$\Rightarrow \quad \frac{E_i}{V} = 2p_i - 1 \,. \qquad \text{---- (1A.5)}$$

Let

$$\frac{E_i}{V} \pm \Delta i = 2(p_i \pm \sigma_N i) - 1 \qquad \text{---- (1A.6)}$$

$$\Rightarrow \quad |\Delta i| = 2|\sigma_N i| \qquad \text{---- (1A.7)}$$

$$\Rightarrow \quad |\Delta i| = 2\sqrt{\left|\frac{p_i(1 - p_i)}{N}\right|} \qquad \text{---- (1A.8)}$$

Thus the standard deviation of a normalised analogue quantity is twice that of the standard deviation associated with the probability representing the variable.

## II Errors Incurred in the Basic Stochastic Computing Elements[40]

In order to estimate the total error in a stochastic system we have to consider the standard deviations resulting from each computation. Let $\Delta_0$ be the output standard deviation of the normalised analogue variable while $\Delta_1$ and $\Delta_2$ are the standard deviations of the input variables. The system errors are as follows:

Inversion /

Inversion

$$\Delta_o = \Delta_1 \qquad \text{---- (1A.9)}$$

Multiplication

$$\Delta_o^2 = \Delta_1^2 + \Delta_2^2 - N\Delta_1^2\,\Delta_2^2 \qquad \text{---- (1A.10)}$$

Summation

$$\Delta_o^2 = \tfrac{1}{2}\Delta_1^2 + \tfrac{1}{2}\Delta_2^2 + \frac{N}{4}\left(\frac{E_1}{V} - \frac{E_2}{V}\right)^2 \qquad \text{---- (1A.11)}$$

Addie

$$\sigma_N^2 = \frac{p_i(1 - p_i)}{N}$$

where N is the number of ADDIE states.

If the number of states in the ADDIE is increased by n times, the output accuracy is improved by a factor of $\frac{1}{\sqrt{n}}$.

APPENDIX 1B

DISCRETE TIME ANALYSIS OF THE SIMPLIFIED DIVIDING CIRCUIT

The integrator in the simplified dividing circuit illustrated in Figure 3.2(a) is really a digital UP/DOWN counter. This circuit can be analysed as a synchronous network, and the method of analysis[9] is similar to that used for investigating the behaviour of the noise ADDIE. Let the input probability be, p, and the output probability be n(t)/N. The output sequence is EXCLUSIVE-OR'd with a stochastic sequence of generating probability, $p_z$. If the counter has N states, the output of the EXCLUSIVE-OR gate is n'(t)/N at the tth step, hence,

$$\frac{n'(t)}{N} = \frac{n(t)}{N} q_z + \left(\frac{N-n(t)}{N}\right) p_z \qquad \text{---- (1B.1)}$$

where $q_z = 1-p_z$ and $p_z = \frac{1}{2} + \frac{1}{2}\left(\frac{1}{z}\right)$ where $\left|\frac{1}{z}\right| \leq 1.0$

Thus, the probability fed back to the input of the counter is

$$\frac{n'(t)}{N} = p_z - \frac{n(t)}{zN} \qquad \text{---- (1B.2)}$$

The expected change in the state of the counter between steps (t-1) and t is given by:

$$E\{n(t) - n(t-1)\} = [\text{Probability of counting up}] - [\text{Probability of counting down}] \qquad \text{---- (1B.3)}$$

$$= p[p_z - \frac{n(t-1)}{zN}] - q[\frac{n(t-1)}{zN} + q_z] \qquad \text{---- (1B.4)}$$

$$= p - [\frac{n(t-1)}{zN} + q_z] \qquad \text{---- (1B.5)}$$

The average value of n(t) is

n(0) /

$$n(0) + E\{n(1) - n(0)\} + E\{n(2) - n(1)\} + \ldots\ldots$$
$$+ E\, n(t) - n(t-1)$$

The expected state of the counter after the first step is

$$E\{n(1)\} = E\{n(0)\} + p - \frac{E\{n(0)\}}{zN} - q_z \qquad \text{---- (1B.6)}$$

$$= p-q_z + (1 - \frac{1}{zN})n(0) \qquad \text{---- (1B.7)}$$

After the second step the expected state of the counter is

$$E\{n(2)\} = E\{n(1)\} + p-q_z - \frac{E\{n(1)\}}{zN} \qquad \text{---- (1B.8)}$$

$$= p-q_z + (1 - \frac{1}{zN})E\{n(1)\} \qquad \text{---- (1B.9)}$$

$$= (p-q_z) + (1 - \frac{1}{zN})(p-q_z) + (1 - \frac{1}{zN})^2 n(0)$$

$$\text{---- (1B.10)}$$

Generalising we have:

$$E\{n(t)\} = (p-q_z)\,[1 + (1-\frac{1}{zN}) + \ldots + (1-\frac{1}{zN})^{t-1}]$$

$$+ [1-\frac{1}{zN}]^t\, n(0) \qquad \text{---- (1B.11)}$$

$$= zN(p-q_z)\,[1 - (1-\frac{1}{zN})^t] + [1-\frac{1}{zN}]^t\, n(0)$$

$$\text{---- (1B.12)}$$

$$= zN(p-q_z) - [zN(p-q_z) - n(0)]\,(1-\frac{1}{zN})^t$$

$$\text{---- (1B.13)}$$

If $p_z = 1.0$, then z is unity and the device becomes a noise ADDIE. As z increases the circuit takes longer to attain a steady-state output. In the steady state,

$$E\{n(t)\} = zN(p-q_z) \qquad \text{---- (1B.14)}$$

Let /

Let $p = \frac{1}{2} + \frac{1}{2}\left(\frac{E_{in}}{V}\right)$ ---- (1B.15)

and

$$q_z = \frac{1}{2} - \frac{1}{2}\left(\frac{1}{z}\right) \quad \text{---- (1B.16)}$$

Hence, in the steady state,

$$E\{n(t)\} = zN\left[\frac{1}{2} + \frac{1}{2}\left(\frac{E_{in}}{V}\right) - \frac{1}{2} + \frac{1}{2}\left(\frac{1}{z}\right)\right] \quad \text{---- (1B.17)}$$

$$= \frac{zN}{2}\left[\frac{zE_{in} + V}{zV}\right] \quad \text{---- (1B.18)}$$

But, $V = \frac{N}{2}$, hence

$$E\{n(t)\} = z\,E_{in} + V \quad \text{---- (1B.19)}$$

Let $n(t) = E_o(t) + V$ ---- (1B.20)

Hence, $E\{E_o(t)\} = z\,E_{in}$ ---- (1B.21)

Thus, if a bipolar mapping is used, the network amplifies the input by z. However, as z increases, the bandwidth decreases.

It is obvious that the circuit is solving a linear equation, but if we had to study a stochastic computer network which could solve a set of linear equations the above method of analysis would be extremely tedious. It is easier to approximate the behaviour of sequential networks by differential equations and no less accurate if large counters are used.

# APPENDIX 3A

```
&JOB;HMCLEAN;NONLINEAR DIFFERENTIAL EQUATION;
&FORTRAN;
&LIST;

      LOGICAL LL(10),LLI(10),LLM(10),INP(10),LLD(2)
      INTEGER IS(10),IP(10)
      COMMON /I/LLI,IS/MU/LLM
C SOLUTION OF A NONLINEAR DIFFERENTIAL EQUATION.
      IS(1)=0
      IP(1)=100
      IA=111111
      J=1
      LLD(2)=.FALSE.
      DO 102 NN=1,120
      DO 103 MM=1,500
      ID=IS(1)
      CALL RANNUM(ID,LL,IA)
      LLD(1)=LL(1)
      INP(2)=LLD(1)
      INP(1)=LLD(2)
      CALL MULT(INP,J)
      INP(1)=.NOT.LLM(1)
      LLD(2)=LLD(1)
      ID=IP(1)
      CALL RANNUM(ID,LL,IA)
      INP(2)=LL(1)
      CALL INT(INP,J)
  103 CONTINUE
      WRITE(2,33)NN,IS(1)
  102 CONTINUE
   33 FORMAT(I6,2X,I6)
      STOP
      END
```

## APPENDIX 3B

```
&JOB;HRMCL M2;MULTIPLYING CIRCUIT;
&FORTRAN;
&LIST;

      LOGICAL LL(10),LLI(10),LLM(10),INP(10)
      INTEGER IS(10),IP(10)
      COMMON /I/LLI,IS/MU/LLM
      J=1
      IA=111111
      IP(1)=-500
      IP(2)=1352
      IS(1)=500
      DD=4096.0
      DO 102 NN=1,120
      DO 103 MM=1,500
      ID=IS(1)
      CALL RANNUM(ID,LL,IA,DD)
      INP(1)=LL(1)
      ID=IP(2)
      CALL RANNUM(ID,LL,IA,DD)
      INP(2)=LL(1)
      CALL MULT(INP,J)
      INP(1)=.NOT.LLM(1)
      ID=IP(1)
      CALL RANNUM(ID,LL,IA,DD)
      INP(2)=LL(1)
      CALL INT(INP,J)
  103 CONTINUE
      WRITE(5,33)NN,IS(1)
  102 CONTINUE
   33 FORMAT(I6,2X,I6)
      STOP
      END
```

# APPENDIX 3C

```
&JOB;HRMCL A9;SECOND ORDER ADDIE;
&FORTRAN;
&LIST;


      LOGICAL LL(10),LLA(10),LLI(10),INP(10)
      INTEGER AS(10),IS(10),IP(10)
      COMMON /I/LLI,IS/O/LLA,AS
C SECOND ORDER SYSTEM.
C THE UNDERDAMPED CASE WITH ZETA EQUAL TO 0.25
C DEFINE THE INTEGRATOR AND ADDIE STATES.
      IP(1)=512
      AS(1)=0
      IS(1)=0
C INITIALISE THE RANDOM NUMBER GENERATOR WITH AN ODD NUMBER.
      IA=111111
      DO 17 MM=1,480
      DO 9 NN=1,250
C ONE RUN THROUGH THE PROGRAMME IS EQIVALENT TO ONE CLOCK PULSE IN D
C INITIALISE THE CLOCK.
C THE PROGRAMME STARTS AT THE OUTPUT END.
C MAIN PROGRAMME FOLLOWS.
      C=4096.0
      ID=AS(1)
      CALL RANNUM(ID,LL,IA,C)
C THE RANDOM NUMBER GENERATOR SIMULATES THE INPUT INTERFACE.
      LLA(1)=LL(1)
      C=1024.0
      ID=IP(1)
      CALL RANNUM(ID,LL,IA,C)
      INP(1)=.NOT.LLA(1)
      INP(2)=LL(1)
C INTEGRATION STEP.
      CALL INT(INP,J)
      ID=IS(1)
      CALL RANNUM(ID,LL,IA,C)
      LLI(1)=LL(1)
      INP(2)=LLI(1)
C ADDIE STAGE.
      CALL OUT(INP,J)
    9 CONTINUE
      WRITE(5,90)MM,IS(1)
   17 CONTINUE
   90 FORMAT(I6,2X,I6)
      STOP
      END
```

# APPENDIX 3D

```
&JOB;HRMCL D1;SINE/COSINE GENERATOR;
&FORTRAN;
&LIST;

      LOGICAL LL(10),LLI(10),INP(2)
      INTEGER IS(10)
      COMMON /I/LLI,IS
C SINE/COSINE GENERATOR.
      IA=111111
      IS(1)=2548
      IS(2)=2048
      INP(1)=.FALSE.
      INP(2)=.FALSE.
      DO 102 MM=1,240
      DO 103 NN=1,500
      J=1
      CALL INT(INP,J)
      ID=IS(1)
      CALL RANNUM(ID,LL,IA)
      INP(1)=LL(1)
      INP(2)=LL(1)
      J=2
      CALL INT(INP,J)
      ID=IS(2)
      CALL RANNUM(ID,LL,IA)
      INP(1)=.NOT.LL(1)
      INP(2)=.NOT.LL(1)
  103 CONTINUE
      WRITE(5,90)MM,IS(1),IS(2)
  102 CONTINUE
   90 FORMAT(I6,2X,I6,2X,I6)
      STOP
      END
```

APPENDIX 3E

I Time Scaling by Variation of the Clock Frequency

Linear differential equations solved using stochastic computer simulations take on the following form:

$$\frac{a_n}{f_c^n} D^n E_o(t) + \frac{a_{n-1}}{f_c^{n-1}} D^{n-1} E_o(t) + \ldots + \frac{a_1}{f_c} D E_o(t) + a_o E_o(t) = E_{in}(t) \quad \text{---- (3.5.1)}$$

Consider the homogeneous case, then,

$$\frac{a_n}{f_c^n} D^n E_o(t) + \frac{a_{n-1}}{f_c^{n-1}} D^{n-1} E_o(t) + \ldots + \frac{a_1}{f_c} D E_o(t) + a_o E_o(t) = 0 \quad \text{---- (3.5.2)}$$

Choosing a solution for this equation for which $E_o(t) = Ae^{\lambda t}$ and let $\lambda = f_c\xi$, then since $e^{f_c\xi t} \neq 0$,

$$a_n \xi^n + a_{n-1}\xi^{n-1} + \ldots + a_1\xi + a_o = 0 \quad \text{---- (3.5.3)}$$

This is a polynomial in $\xi$ which can be solved. Hence,

$$(\xi-\xi_n)(\xi-\xi_{n-1}) \ldots\ldots (\xi-\xi_1) = 0 \quad \text{---- (3.5.4)}$$

$$\Rightarrow \xi = \xi_1, \xi_2, \ldots\ldots, \xi_n$$

Since

$$_cE_o(t) = \sum_{i=1}^{n} A_i e^{f_c\xi_i t} \quad \text{---- (3.5.5)}$$

$$\Rightarrow {}_cE_o(t) = \sum_{i=1}^{n} A_i e^{\xi_i t_N} \quad \text{---- (3.5.6)}$$

where /

where

$$t_N = f_c t = \frac{t}{T_c} \qquad \text{---- (3.5.7)}$$

and the $A_i$ are arbitrary constants.

This is the normalised transient solution of an nth order linear differential equation with constant coefficients.

Let

$$L = \{ \frac{a_n}{f_c^n} D^n + \frac{a_{n-1}}{f_c^{n-1}} D^{n-1} + \ldots + \frac{a_1}{f_c} D + a_0 \} \qquad \text{---- (3.5.8)}$$

Then in the non-homogeneous case,

$$L \{ {}_p E_O(t) \} = E_{in}(t) \qquad \text{---- (3.5.9)}$$

$${}_p E_O(t) = L^{-1}[ E_{in}(t)] \qquad \text{---- (3.5.10)}$$

Hence,

$${}_p E_O(t_N) = {}_N L^{-1}[ E_{in}(t_N)] \qquad \text{---- (3.5.11)}$$

where $E_{in}(t_N)$ is the normalised form of $E_{in}(t)$ with respect to the clock period.

Let $${}_N L = \{a_n \mathcal{D}^n + a_{n-1} \mathcal{D}^{n-1} + \ldots + a_1 \mathcal{D} + a_0\} \qquad \text{---- (3.5.12)}$$

where

$$\frac{D}{f_c} = \mathcal{D} \qquad \text{---- (3.5.13)}$$

II Examples

$$t_N^n = \frac{D}{f_c} f_c^n t^n = f_c^{n-1} n t^{n-1} = n t_N^{-1} \qquad \text{---- (3.5.14)}$$

and if

$$\omega t = \omega_n f_c t = \omega_N t_N ,$$

then

$$\sin \omega t = \sin \omega_N t_N$$

and /

and

$$D \sin\omega t = f_c \mathcal{D} \sin\omega_N t_N = f_c \omega_N \cos \omega_N t_N$$

$$= \omega \cos \omega t \qquad \text{---- (3.5.15)}$$

and

$$\mathcal{D} \sin\omega_N t_N = \frac{D}{f_c} \sin\omega_N t_N = \frac{\omega}{f_c} \cos \omega_N t_N$$

$$= \omega_N \cos\omega_N t_N \qquad \text{---- (3.5.16)}$$

Hence a model which predicts the deterministic behaviour of a stochastic simulation can be produced without referring to any specific units of time.

## APPENDIX 3F

```
&JOB;HRMCLSTI;STATISTICAL TESTS;
&FORTRAN
&LIST;
```

```
      DIMENSION IX(200),K(200)
      WRITE(2,5)
      ISUM1=0
      LIM=141
      DO 1 I=1,LIM
      READ(3,2)K(I),IX(I)
      ISUM1=ISUM1+IX(I)
    1 CONTINUE
      MEAN=ISUM1/LIM
      ISUM2=0
      DO 3 I=1,LIM
      ISUM2=ISUM2+(IX(I)-MEAN)**2
    3 CONTINUE
      VAR=ISUM2/(LIM-1)
      SD=SQRT(VAR)
      WRITE(2,4)MEAN,VAR,SD
    2 FORMAT(I6,2X,I6)
    4 FORMAT(I6,2F9.4)
    5 FORMAT(24H  MEAN       VAR        SD)
      STOP
      END
```

# APPENDIX 3G

## LIST OF FORTRAN IN SUBROUTINES

```
      SUBROUTINE RANNUM (EE,LP,IX,DD)
      LOGICAL LP
      IY=IX*4099
      IF(IY)1,2,2
    1 IY=IY+8388607+1
    2 RN=IY
      RN=RN/8388607.0
      IX=IY
      E=(EE+DD)/(2*DD)
      LP=.FALSE.
      IF(E-RN)4,4,3
    3 LP=.TRUE.
    4 RETURN
      END
```

```
      SUBROUTINE RANNUM (EE,LP,IX)
      LOGICAL LP
      IY=IX*4099
      IF(IY)1,2,2
    1 IY=IY+8388607+1
    2 RN=IY
      RN=RN/8388607.0
      IX=IY
      E=(EE+2048.0)/4096.0
      LP=.FALSE.
      IF(E-RN)4,4,3
    3 LP=.TRUE.
    4 RETURN
      END
```

```
      SUBROUTINE MULT (SINP,K)
      LOGICAL SINP(2),SLLM(2)
      COMMON /MU/SLLM
      SLLM(K)=(SINP(1).AND.SINP(2)).OR.((.NOT.SINP(1)).AND.(.NOT.SI
     9))
      RETURN
      END
```

```
      SUBROUTINE SUM (SINP,K,IA)
      LOGICAL SLLS(2),SINP(2),LP
      COMMON /S/SLLS
      DE=0.0
      ED=2048.0
      CALL RANNUM (DE,LP,IA,ED)
      SLLS(K)=(SINP(1).AND.LP).OR.(SINP(2).AND.(.NOT.LP))
      RETURN
      END
```

```
      SUBROUTINE INT (SINP,K,IA)
      LOGICAL SLLI(6),SINP(2),LP
      INTEGER SIS(5)
      COMMON /I/SLLI,SIS
      IF((SINP(1).AND.(.NOT.SINP(2))).OR.((.NOT.SINP(1)).AND.SINP(2
     9T021
      SIS(K)=SIS(K)-1
      IF(SINP(1).AND.SINP(2))SIS(K)=SIS(K)+2
   21 RETURN
      END


      SUBROUTINE OUT (SINP,K)
      LOGICAL SLLA(5),SINP(2)
      INTEGER SAS(5)
      COMMON /O/SLLA,SAS
      IF((SINP(2).AND.(.NOT.SLLA(K))).OR.((.NOT.SINP(2).AND.SLLA(K)
     1T015
      GOT013
   15 SAS(K)=SAS(K)-1
      IF(SINP(2).AND.(.NOT.SLLA(K)))SAS(K)=SAS(K)+2
   13 RETURN
      END
```

APPENDIX 4A

I Solution of a Set of Linear Equations Using The Error Criterion Method

This optimisation scheme requires that

$$\dot{\underline{x}}(t) = -K \{A \underline{x}(t) - \underline{b}\} , \quad K > 0 \qquad \text{---- (4.A.1)}$$

and K is a function of n, the number of variables in the problem. The value of K depends on the size of the integrators n and the number of summations to be performed, and as a result, K decreases as the size of the problem increases.

An expression may be determined for $\underline{x}(t)$ in the following manner. First, equation (4.A.1) is Laplaced transformed.

$$s\underline{X}(s) - \underline{x}(0) = -KA\underline{X}(s) + \frac{K}{s}\underline{b} \qquad \text{---- (4.A.2)}$$

$$\Rightarrow (sI + KA)\underline{X}(s) = \underline{x}(0) + \frac{K}{s}\underline{b} \qquad \text{---- (4.A.3)}$$

$$\Rightarrow \underline{X}(s) = (sI + KA)^{-1} \{\underline{x}(0) + \frac{K}{s}\underline{b}\} \qquad \text{---- (4.A.4)}$$

The inverse Laplace transform is applied to equation (4.A.4) to yield $\underline{x}(t)$. However,

$$\mathcal{L}^{-1} [(sI + KA)^{-1}] = e^{-KAt} \qquad \text{---- (4.A.5)}$$

which is an nxn matrix called the transition matrix, and it is a matrix function corresponding to:

$$e^{-Kzt}$$

where z is a scalar variable.

Hence,

$$e^{-KAt} = [I - KAt + \frac{[KAt]^2}{2!} - \frac{[KAt]^3}{3!} + \ldots\ldots] \qquad \text{---- (4.A.6)}$$

$$= \sum_{i=0}^{\infty} \frac{[-KAt]^i}{i!} \qquad \text{---- (4.A.7)}$$

Thus /

Thus,

$$\underline{x}(t) = e^{-KAt}\underline{x}(0) + K\int_0^t e^{-KAt}\underline{b}\,dt \qquad \text{---- (4.A.8)}$$

but,

$$\underline{x}_{opt} = A^{-1}\,\underline{b}$$

$$\Rightarrow \quad \underline{x}(t) = e^{-KAt}(\underline{x}(0) - \underline{x}_{opt}) + \underline{x}_{opt} \qquad \text{---- (4.A.9)}$$

which is the required solution because

$$\lim_{t\to\infty} e^{-KAt} = 0 \qquad \text{---- (4.A.10)}$$

and

$$\lim_{t\to\infty} \underline{x}(t) = \underline{x}_{opt} \qquad \text{---- (4.A.11)}$$

also,

$$\lim_{t\to 0} e^{-KAt} = I \qquad \text{---- (4.A.12)}$$

$$\Rightarrow \quad \lim_{t\to 0} \underline{x}(t) = \underline{x}(0) \quad . \qquad \text{---- (4.A.13)}$$

Thus in the steady state $\underline{x}_{opt}$ is independent of $\underline{x}(0)$.

The transition matrix $\exp(-KAt)$ can be evaluated using the Caley-Hamilton theorem provided the eigenvalues of A are distinct. As K decreases the transition matrix takes longer to reach the limit defined by equation (4.A.10). If K is too small the system may fail to converge to the required answer. The attenuation due to the summation process may prevent convergence even if the integration gains are large.

II Caley-Hamilton Theorem

This states that a square matrix, A, satisfies its own characteristic equation, ie,

$$|A - \lambda I| = f(\lambda) = 0 \qquad \text{---- (4.A.14)}$$

$$\Rightarrow \quad f(A) = 0 \qquad \text{---- (4.A.15)}$$

Thus, /

Thus, terms such as $A^n$, $A^{n+1}$, $A^{n+2}$, ....., may be expressed in terms of I, A, $A^2$, ...., $A^{n-1}$. Suppose,

$$g(A) = \sum_{i=0}^{\infty} k_i A^i = \sum_{i=0}^{n-1} \alpha_i A^i \qquad \text{---- (4.A.16)}$$

then by Caley-Hamilton theorem,

$$g(\lambda) = \sum_{i=0}^{\infty} k_i \lambda^i = \sum_{i=0}^{n-1} \alpha_i \lambda^i \qquad \text{---- (4.A.17)}$$

If the $g(\lambda_i)$ and $\lambda_i$ are known all the coefficients $\alpha_i$ can be evaluated provided that the $\lambda_i$ are distinct. Hence, g(A) can be evaluated. Thus exp(-KAt) is expressed in terms of the $\exp(-K\lambda_i t)$ so that for the transition matrix to be stable and convergent the $\lambda_i$ must have positive real parts.

## III Discrete Time Analysis of the Error Criterion Method

The stochastic computer circuit for solving sets of linear equations illustrated in Figure can be analysed as a discrete system. If A is the scaled coefficient matrix then let

$$P = \{p_{ij}\} = \{\tfrac{1}{2} + \tfrac{1}{2} a_{ij}\} \qquad \text{---- (4.A.18)}$$

where $A = \{a_{ij}\}$ and $q_{ij} = 1 - p_{ij}$.

Similarly, if x(t) is the scaled output of the circuit then

$$\Pi i(t) = \tfrac{1}{2} + \tfrac{1}{2} x_i(t) \qquad \text{---- (4.A.19)}$$

If the integrator counters each have N states, then

$$\Pi i(t) = \frac{Z_i(t)}{N} \qquad \text{---- (4.A.20)}$$

and /

and if $\underline{b}$ is the scaled input vector, then

$$p_{bi} = \tfrac{1}{2} + \tfrac{1}{2} b_i \qquad \text{---- (4.A.21)}$$

The output probability of each multiplier is then

$$p_O = a_{ij}\Pi_j \div q_{ij} \qquad \text{---- (4.A.22)}$$

If $\nu$ is the attenuation factor due to the summation process, then one of the inputs to the ith integrator is:

$$1 - \nu \left[ \sum_{k=1}^{\ell} (a_{ik}\Pi_k(t) + q_{ik}) \right] \qquad \text{---- (4.A.23)}$$

where there are (n+1) quantities to be summed, and (n+1) is an integer power of 2. The probability of the other input is

$$1 - \nu \left[ 1 - P_{bi} + \sum_{k=\emptyset+1}^{n} (a_{ik}\pi_k(t) + q_{ik}) \right] \qquad \text{---- (4.A.24)}$$

The expected change in state of the ith integrator between steps t and (t-1) is

$$E\{Z_i(t)-Z_i(t-1)\} = \left[ \begin{array}{c}\text{probability of the ith counter} \\ \text{counting up}\end{array} \right] - \left[ \begin{array}{c}\text{probability of the ith counter} \\ \text{counting down}\end{array} \right]$$

$$= 1 - \nu \left[ 1 - P_{bi} + \sum_{k=1}^{n} q_{ik} + \sum_{k=1}^{n} a_{ik}\pi_k \right] \qquad \text{---- (4.A.25)}$$

The average value of the ith counter is:

$$Z_i(0) + E\{Z_i(1) - Z_i(0)\} + E\{Z_i(2) - Z_i(1)\} + \ldots\ldots + E\{Z_i(t) - Z_i(t-1)\} \qquad \text{---- (4.A.26)}$$

The /

The expected change in $Z_i$ after the first step is

$$E\{Z_i(1)\} = E\{Z_i(0)\} + 1 - \nu \left[ 1 - P_{bi} + \sum_k q_{ik} + \sum_k a_{ik} \frac{E\{Z_k(0)\}}{N} \right]$$

---- (4.A.27)

Writing this in matrix form we have:

$$E\{\underline{Z}(1)\} = E\{\underline{Z}(0)\} + \underline{m} - \nu \left[ \underline{c} + A. \frac{E\{\underline{Z}(0)\}}{N} \right]$$

---- (4.A.28)

where $\underline{Z}(t)$ is the n vector $[Z_1(t), Z_2(t), \ldots, Z_n(t)]^T$

---- (4.A.29)

$\underline{m}$ is the n vector $[1, 1, \ldots\ldots, 1]^T$ ---- (4.A.30)

$$\underline{c} = \underline{m} - \underline{P}_b + \underline{q}_a$$ ---- (4.A.31)

where $\underline{q}_a$ is the n vector $[\sum_k q_{1k}, \sum_k q_{2k}, \ldots, \sum_k q_{nk}]^T$

---- (4.A.32)

and, $\underline{d} = \underline{m} - \nu\underline{c}$ ---- (4.A.33)

Hence, $E\{\underline{Z}(1)\} = \underline{d} + (I - \frac{\nu}{N}A)E\{\underline{Z}(0)\}$ ---- (4.A.40)

where I is the identity matrix, then

$$E\{\underline{Z}(2)\} = E\{\underline{Z}(1)\} + \underline{d} - \frac{\nu}{N}A\, E\{\underline{Z}(1)\}$$

$$= \underline{d} + (I - \frac{\nu}{N}A)E\{\underline{Z}(1)\}$$

$$= \underline{d} + (I - \frac{\nu}{N}A)\underline{d} + (I - \frac{\nu}{N}A)^2 E\{\underline{Z}(0)\}$$

--- (4.A.41)

Generalising, /

Generalising,

$$E\{\underline{Z}(t)\} = [I + (I - \tfrac{\nu}{N}A) + \ldots + (I - \tfrac{\nu}{N}A)^{t-1}]\ \underline{d} + (I - \tfrac{\nu}{N}A)^{t} E\{\underline{Z}(O)\}$$

$$= [I - (I - \tfrac{\nu}{N}A)^{t}]\ \tfrac{N}{\nu}A^{-1}\ \underline{d} + (I - \tfrac{\nu}{N}A)^{t} E\{\underline{Z}(O)\}$$

$$= \tfrac{N}{\nu}A^{-1}\underline{d} - (I - \tfrac{\nu}{N}A)^{t} * [\tfrac{N}{\nu}A^{-1}\underline{d} - E\{\underline{Z}(O)\}] \quad \text{---- (4.A.42)}$$

As the number of summations increases $\nu$ decreases and equation (4.A.42) takes longer to converge to a limit. In the steady state,

$$E\{\underline{Z}(t)\} = \tfrac{N}{\nu}\ A^{-1}[\underline{m} - \nu(\underline{m} - \underline{p}_b + \underline{q}_a)] \quad \text{---- (4.A.43)}$$

$$= \frac{NA^{-1}}{\nu}\ [(1-\nu)\underline{m} + \nu(\underline{p}_b - \underline{q}_a)] \quad \text{---- (4.A.44)}$$

provided the matrix function $(I-\frac{\nu}{N}A)^{t}$ converges to the null matrix.

But,

$$E\{\underline{Z}(t)\} = \frac{NA^{-1}}{\nu}\ [(1-\nu)\underline{m} + \nu(\tfrac{1}{2}\underline{m} + \tfrac{1}{2}\underline{b} - n\underline{m} + \underline{p}_a)] \quad \text{---- (4.A.45)}$$

where $P_{ai} = \sum_{k=1}^{n} P_{ik}$ ---- (4.A.46)

$$E\{\underline{Z}(t)\} = \frac{NA^{-1}}{\nu}\ [(1-\nu)\underline{m} + \nu((\tfrac{1-n}{2})\underline{m} + \tfrac{1}{2}\underline{b} + \tfrac{1}{2}A\underline{m})] \quad \text{---- (4.A.47)}$$

$$= \frac{NA^{-1}}{\nu}\ [(1-\nu+\nu(\tfrac{1-n}{2}))I + \tfrac{\nu}{2}A]\underline{m} + \tfrac{N}{2}A^{-1}\underline{b} \quad \text{---- (4.A.48)}$$

where $\underline{x}_{opt} = A^{-1}\underline{b}$.

$$E\{\underline{Z}(t)\} = \tfrac{N}{2}\ [(2(\tfrac{1-\nu}{\nu}) + 1-n)A^{-1} + I]\ \underline{m} + \tfrac{N}{2}A^{-1}\underline{b} \quad \text{---- (4.A.49)}$$

But /

But

$$2\left(\frac{1-\nu}{\nu}\right) + 1-n = 0 \qquad \text{---- (4.A.50)}$$

since n+1 is an integer power of 2.

Hence,

$$E\{\underline{Z}(t)\} = \frac{N}{2}(\underline{m} + A^{-1}\underline{b}) \qquad \text{---- (4.A.51)}$$

which is the required solution. Equation (4.A.42) can be rewritten as

$$E\{\underline{Z}(t)\} = \frac{N}{2}(\underline{m}+A^{-1}\underline{b}) - (I-\frac{\nu}{N}A)^{t}[\frac{N}{2}(\underline{m}+A^{-1}\underline{b}) - \underline{Z}(0)] \qquad \text{---- (4.A.52)}$$

The eigenvalues of $(I-\frac{\nu}{N}A)$ must have modulii which are less than unity for the system to be stable. If the value of (n+1) is not an integer power of 2 then extra stochastic operators have to be introduced into the summing network to equalise the normalisation of the variables. Stochastic summers can be used for this purpose and the effects of these extra operators can be included in the above analysis if we add an extra term, $\frac{\varepsilon}{2}\underline{m}$, to equation (4.A.31) where $\varepsilon$ represents extra inputs into the summing array each having a probability of 0.5. The number of extra inputs must be such that $(n+\varepsilon+1)$ is an integer power of 2. This correction term can also be used to represent the effect of compensating multipliers.

# APPENDIX 4B

```
&JOB;HM LE3AB;LINEAR EQUATION PROBLEM;
&FORTRAN;
&LIST;

      DIMENSION D(7,7),X(5),IS(5)
      LOGICAL LLX(5),LL(6,6),LLKA(6,6),LLM(2),LLS(2),LP,LLT(6)
      LOGICAL LLB(6),LLAX(6,6),LLS1(6),LLI(5),INP(2)
      COMMON /I/LLI,IS/MU/LLM/S/LLS
      LIM1=3
      LIM2=5
      L=1
      N=3
      IA=111111
      WRITE(2,16)
      DO 11 I=1,LIM1
      DO 11 J=1,LIM2
      READ(7,12)D(I,J)
      WRITE(2,14)I,J,D(I,J)
   11 CONTINUE
      IS(1)=-512
      IS(2)=512
      IS(3)=1536
      DO 9 MM=1,100
      DO 17 NN=1,50
      DO 1 I=1,N
      DO 1 K=1,N
      DE=IS(K)
      CALL RANNUM(DE,LP,IA)
      INP(1)=LP
      DE=D(I,K)
      CALL RANNUM(DE,LP,IA)
      INP(2)=LP
      CALL MULT(INP,L)
      LLAX(I,K)=.NOT.LLM(1)
```

```
    1 CONTINUE
      DO 8 I=1,N
      K=1
      INP(1)=LLAX(I,K)
      K=2
      INP(2)=LLAX(I,K)
      CALL SUM(INP,L,IA)
      LLS1(1)=LLS(1)
      K=3
      DE=D(I,4)
      CALL RANNUM(DE,LP,IA)
      INP(1)=LP
      INP(2)=LLAX(I,K)
      CALL SUM(INP,L,IA)
      LLS1(2)=LLS(1)
      INP(1)=LLS1(1)
      INP(2)=LLS1(2)
      CALL INT(INP,I,IA)
    9 CONTINUE
   10 FORMAT(I4,2X,I5,2X,I5,2X,I5)
   12 FORMAT(F8.4)
   14 FORMAT(2I3,F8.4)
   16 FORMAT(14H   I   J   D(I,J))
      STOP
      END
```

APPENDIX 4C

## Solution of a Set of Linear Equations Using the Method of Steepest Descent

This method of optimisation finds the minimum value of a scalar function, $f(\underline{x}(t))$, where,

$$f(\underline{x}(t)) = \tfrac{1}{2}\, \underline{e}(t)^T\, \underline{e}(t) \qquad \text{---- (4.C.1)}$$

and

$$\underline{e}(t) = A\underline{x}(t) - \underline{b} \qquad \text{---- (4.C.2)}$$

Equation (4.C.1) has a minimum value when $\underline{x}(t) = \underline{x}_{opt}$ and $A\underline{x}_{opt} = \underline{b}$. Further, the criterion function, $f(\underline{x})$ must be continuous and differentiable everywhere, and $f(\underline{x}) \geqslant 0$. This index of performance can be thought of as being made up of an infinite number of level surfaces bounded by countours. Let $\hat{c}$ be one of these level surfaces, then,

$$\hat{c} = \hat{\underline{e}}^T\, \hat{\underline{e}} \qquad \text{---- (4.C.3)}$$

The fastest way to reduce $f(\underline{x})$ is to move normally to c in the direction of the gradient of $f(\underline{x})$, ie,

$$\dot{f}(\underline{x}) = \nabla f(\underline{x}).\dot{\underline{x}} \qquad \text{---- (4.C.4)}$$

$$= |\nabla f| \,.\, |\dot{\underline{x}}| \cos\theta \qquad \text{---- (4.C.5)}$$

Thus $\dot{f}(\underline{x})$ is a maximum when $\theta = 0$ and so $\nabla f$ and $\dot{\underline{x}}$ are parallel.

The steepest descent algorithm requires that

$$\dot{\underline{x}}(t) = -K\nabla f(\underline{x}(t)), \quad K > 0 \qquad \text{---- (4.C.6)}$$

The constant K is a gain term which depends on the number of summations which have to be performed and the capacity of the integrators used. The negative sign in equation (4.C.4) ensures that $f(\underline{x}(t))$ is minimised. Thus,

$\underline{x}(t)$ /

$$\dot{\underline{x}}(t) = -\frac{K}{2} \nabla \underline{e}^{T}(t)\, \underline{e}(t) \qquad \text{---- (4.C.7)}$$

where

$$\nabla = \begin{bmatrix} \partial/\partial x_1 \\ \partial/\partial x_2 \\ \vdots \\ \partial/\partial x_n \end{bmatrix} \quad \text{and} \quad \underline{e}(t)^{T}\, e(t) = \sum_{k=1}^{n} e_k(t)^2$$

$$\Rightarrow \quad \dot{x}_j(t) = -K \frac{\partial}{\partial x_j} \sum_{i=1}^{n} e_i(t)^2 \qquad \text{---- (4.C.8)}$$

$$= -K \sum_{i=1}^{n} a_{ij} \{ \sum_{k=1}^{n} a_{ik} x_k - b_i \} \qquad \text{---- (4.C.9)}$$

or

$$\dot{\underline{x}}(t) = -KA^{T} \{A\underline{x}(t) - \underline{b}\} \qquad \text{---- (4.C.10)}$$

The state variable method of analysis can be used to determine the steady state and transient response of $\underline{x}(t)$, hence,

$$\underline{x}(t) = e^{-KA^{T}At} (\underline{x}(0) - \underline{x}_{opt}) + \underline{x}_{opt} . \qquad \text{---- (4.C.11)}$$

Since $A^{T}A$ always has eigenvalues with positive real parts,

$$\lim_{t \to \infty} e^{-KA^{T}At} = 0 \qquad \text{---- (4.C.12)}$$

and

$$\lim_{t \to \infty} \underline{x}(t) = \underline{x}_{opt} \qquad \text{---- (4.C.13)}$$

as required for an optimisation.

As stated previously, K depends on the attenuation introduced in the summation process and the gain of the integrators. If the size of the problem is too large, the attenuation due to summing may be so great that the initial driving signal applied to the input of an integrator may be so small that convergence to the optimal solution is poor. Also, as the size of a problem increases, convergence will be slower. An alternative explanation of poor convergence can be given in terms of the properties of the criterion function. If the problem /

problem size increases, K will decrease and hence the $\dot{x}_j(t)$ will be smaller, and the optimisation may stop without being anywhere near the true optimum.

The effect of shallow gradients in a stochastic computer circuit will be characterised by large excursions of the integrator counters. There will be an upper limit to the size of problem which can be solved on a stochastic computer.

## II Discrete Time Domain Analysis of the Method of Steepest Descent

The steepest descent circuit for solving a set of linear equations illustrated in Figures (4.13(a)) and (4.13(b)) can be analysed as a digital network. Let A be the scaled coefficient matrix where

$$p_{ij} = \tfrac{1}{2} + \tfrac{1}{2}a_{ij} \qquad \text{---- (4.C.14)}$$

and let $\underline{b}$ be the scaled input vector where

$$p_{bi} = \tfrac{1}{2} + \tfrac{1}{2}b_i \qquad \text{---- (4.C.15)}$$

If $\underline{x}(t)$ is the scaled output vector, then

$$\underline{z}(t) = \frac{N}{2}(\underline{m} + \underline{x}(t)) \qquad \text{---- (4.C.16)}$$

and

$$\pi_i(t) = \frac{z(t)}{N}$$

where N is the number of counter states, and $\underline{m}$ is the n vector $[1,1,\ldots,1]^T$. We define the n vector $\underline{P}_a$ such that:

$$\underline{P}_a = \tfrac{1}{2}(\hbar I + A)\underline{m} \qquad \text{---- (4.C.17)}$$

where I is the identity matrix,

$\underline{q}_a$ /

$$\underline{q}_a = \tfrac{1}{2}(nI - A)\underline{m} \qquad \text{---- (4.C.18)}$$

Also, let $\underline{P}_a'$ be an n vector such that

$$\underline{P}_a' = \tfrac{1}{2}(nI + A^T)\underline{m} \qquad \text{---- (4.C.19)}$$

and

$$\underline{q}_a' = \tfrac{1}{2}(nI - A^T)\underline{m} \qquad \text{---- (4.C.20)}$$

The probability vector representing the error at step t is

$$\underline{P}_c(t) = \nu_1 \left[\underline{m} - \underline{P}_b + A\underline{\pi}(t) + \underline{q}_a + \frac{\varepsilon_1}{2}\underline{m}\right] \qquad \text{---- (4.C.21)}$$

where $\nu_1$ is the attenuation introduced by the summation process and $\varepsilon_1$ represents the number of extra inputs required to equalise the normalisation of the variables being summed. If there are n+1 variables being summed, then

$$\nu_1 = \frac{1}{n + \varepsilon_1 + 1}$$

where $(n+\varepsilon_1+1)$ is an integer power of 2.

The average value of the jth counter is

$$z_j(0) + E\{z_j(1) - z_j(0)\} + E\{z_j(2) - z_j(1)\} + \ldots + E\{z_j(t) - z_j(t-1)\} \qquad \text{---- (4.C.22)}$$

The expected change in state of the jth integrator between steps t and (t-1) is

$$E\{z_j(t) - z_j(t-1)\} = [\text{probability of the jth counter counting up}] - [\text{probability of the jth counter counting down}]$$

ie $E\{\underline{z}(t) - \underline{z}(t-1)\}$ ---- (4.C.23)

= /

$$= \underline{m} - \nu_2\{\nu_1 A^T [\underline{m} - \underline{P}_b + A\underline{\pi}(t) + \underline{q}_a + \frac{\varepsilon_1}{2}\underline{m}]$$

$$+ \underline{q}_a' + \frac{\varepsilon_2}{2}\underline{m}\} \qquad \text{---- (4.C.24)}$$

where $\nu_2$ is the attenuation factor introduced when forming the gradient vector and,

$$\nu_2 = \frac{2}{n + \varepsilon_2}$$

where $(n+\varepsilon_2)$ is an integer power of 2.

Hence we can determine an expression for $E\{\underline{z}(t)\}$ ie,

$$E\{\underline{z}(t)\} = \frac{N}{2}[\underline{m} + \underline{x}_{opt}] - (I-\frac{\nu}{N} A^T A)^t (\frac{N}{2}[\underline{m}+\underline{x}_{opt}] - \underline{z}(0))$$

$$\text{-----(4.C.25)}$$

where $\nu = \nu_1\nu_2$ and $\underline{x}_{opt} = A^{-1}\underline{b}$

As the problem size increases, $\nu$ decreases and the problem takes longer to converge. The eigenvalues of $(I-\frac{\nu}{N} A^T A)$ must have modulii which are less than unity for the problem to converge to a solution.

```
&JOB;HM  LE4H;LINEAR EQUATION PROBLEM;.
&FORTRAN;
&LIST;


      DIMENSION D(7,7),X(5),IS(5)
      LOGICAL LLX(5),LL(6,6),LLKA(6,6),LLM(2),LLS(2),LP,LLT(6)
      LOGICAL LLB(6),LLP(6,6),LLS1(6),LLY(6),LLI(5),INP(2)
      COMMON /I/LLI,IS/MU/LLM/S/LLS
C SOLUTION OF A SET OF LINEAR EQUATIONS.
      LIM1=3
      LIM2=5
      L=1
      N=3
      IA=111111
C READ AND WRITE THE DATA FIELD, D(I,J) .
      WRITE(2,16)
      DO 11 I=1,LIM1
      DO 11 J=1,LIM2
      READ(7,12)D(I,J)
      WRITE(2,14)I,J,D(I,J)
   11 CONTINUE
C INITIALISE THE "X" VECTOR, AND THE INTEGRATOR STATES.
      IS(1)=0
      IS(2)=0
      IS(3)=0
C COMPARISON WITH THE RANDOM NUMBER GENERATOR AND INPUT INTERFACE.
      DO 9 MM=1,100
      DO 17 NN=1,50
C CALCULATION OF "A(I,K)*X(K)"
      DO 1 I=1,N
      DO 1 K=1,N
      DE=IS(K)
      CALL RANNUM(DE,LP,IA)
      INP(1)=LP
      DE=D(I,K)
      CALL RANNUM(DE,LP,IA)
      INP(2)=LP
      CALL MULT(INP,L)
      LLAX(I,K)=LLM(1)
    1 CONTINUE
C FORMATION OF THE ERROR CRITERION.
C SUMMATION OF THE "A(I,K)*X(K)'S", ADDING -B(I)
      DO 8 I=1,N
      K=1
      INP(1)=LLAX(I,K)
      K=K+1
      INP(2)=LLAX(I,K)
      CALL SUM(INP,L,IA)
      LLS1(1)=LLS(1)
      K=K+1
      DE=D(I,4)
      CALL RANNUM(DE,LP,IA)
      INP(1)=.NOT.LP
      INP(2)=LLAX(I,K)
      CALL SUM(INP,L,IA)
      LLS1(2)=LLS(1)
      INP(1)=LLS1(1)
```

```
      INP(2)=LLS1(2)
      CALL SUM(INP,L,IA)
      LLT(I)=LLS(1)
    8 CONTINUE
C MULTIPLY BY "A(J,I)"
      DO 17 J=1,N
      DO 5 I=1,N
      INP(1)=LLT(I)
      DE=D(I,J)
      CALL RANNUM(DE,LP,IA)
      INP(2)=LP
      CALL MULT(INP,L)
      LLY(I)=LLM(1)
    5 CONTINUE
C SUM OVER "I" TO OBTAIN THE DERIVATIVE OF "X(J)".
      I=1
      INP(1)=LLY(I)
      I=I+1
      INP(2)=LLY(I)
      CALL SUM(INP,L,IA)
      LLS1(1)=LLS(1)
      I=I+1
      INP(1)=LLY(I)
      DE=D(3,5)
      CALL RANNUM(DE,LP,IA)
      INP(2)=LP
      CALL MULT(INP,L)
      LLS1(2)=LLM(1)
      INP(1)=.NOT.LLS1(1)
      INP(2)=.NOT.LLS1(2)
      CALL INT(INP,J,IA)
   17 CONTINUE
C WRITE THE VALUES OF "X(J)."
      WRITE(5,10)MM,IS(1),IS(2),IS(3)
    9 CONTINUE
   10 FORMAT(I4,2X,I5,2X,I5,2X,I5)
   12 FORMAT(F8.4)
   14 FORMAT(2I3,F8.4)
   16 FORMAT(14H  I   J   D(I,J))
      STOP
      END
```

```
&JOB;HMLPICA;L.P.PROBLEM;

&FORTRAN;

&LIST;

      DIMENSION D(7,7),IS(5)
      INTEGER AS(5)
      LOGICAL LLM(2),LLS(2),LP,LLT(6),LLR(6),LLAX(7,7),LLS1(6),LLY(6),
     2I(6),LLA(5),INP(2)
      COMMON /I/LLI,IS/O/LLA,AS/MU/LLM/S/LLS
C LINEAR PROGRAMMING PROBLEM.
      LIM1=3
      LIM2=5
      L=1
      IA=777111
C READ AND WRITE THE DATA FIELD,D(I,J) .
      WRITE (2,16)
      DO 11 I=1,LIM1
      DO 11 J=1,LIM2
      READ(7,12)D(I,J)
      WRITE(2,14)I,J,D(I,J)
   11 CONTINUE
      WRITE(2,24)
C INITIALISE THE INTEGRATOR AND ADDIE STATES.
      DO 2 J=1,3
      IS(J)=0
      AS(J)=0
      LLA(J)=.TRUE.
    2 CONTINUE
C COMPARISON WITH THE RANDOM NUMBER GENERATOR AND INPUT INTERFACE.
      DO 9 MM=1,200
      DO 17 NN=1,50
      ED=2048.0
      DO 1 I=1,3
      DO 1 K=1,2
      DE=0.0
      IF(AS(K).GT.0)DE=2048.0
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      DE=D(K,I)
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      CALL MULT(INP,L)
      LLAX(K,I)=LLM(1)
    1 CONTINUE
```

```
C FORMATION OF THE "X" VECTOR.
      DO 8 I=1,3
      K=1
      INP(1)=LLAX(K,I)
      K=K+1
      INP(2)=LLAX(K,I)
      CALL SUM (INP,L,IA)
      LLS1(1)=LLS(1)
      DE=D(I,4)
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      DE=512.0
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      CALL MULT(INP,L)
      INP(1)=.NOT.LLS1(1)
      INP(2)=LLM(1)
      CALL INT(INP,I,IA)
      IF(IS(I).LT.-20)IS(I)=-20
    8 CONTINUE
C FORMATION OF THE ERROR VECTOR.
      DO 5 I=1,2
      DO 5 J=1,3
      DE=D(I,J)
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      DE=IS(J)
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      CALL MULT(INP,L)
      LLAX(I,J)=LLM(1)
    5 CONTINUE
      DO 17 I=1,2
      ED=2048.0
      K=1
      INP(1)=LLAX(I,K)
      K=K+1
      INP(2)=LLAX(I,K)
      CALL SUM(INP,L,IA)
      LLS1(1)=LLS(1)
      K=K+1
      INP(1)=LLAX(I,K)
      DE=D(I,5)
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=.NOT.LP
      CALL SUM(INP,L,IA)
      LLS1(2)=LLS(1)
      INP(1)=LLS1(1)
      INP(2)=LLS1(2)
      CALL SUM(INP,L,IA)
      INP(2)=LLS(1)
      CALL OUT(INP,I)
      IF(AS(I).LT.-15)AS(I)=-15
      ED=128.0
      DE=AS(I)
      CALL RANNUM(DE,LP,IA,ED)
      LLA(I)=LP
   17 CONTINUE
C WRITE THE VALUES OF X(J) AND THE OBJECTIVE FUNCTION.
      WRITE(2,4)MM,IS(1),IS(2),IS(3),AS(1),AS(2)
```

```
  9 CONTINUE
  4 FORMAT(I4,2X,I5,2X,I5,2X,I5,2X,I5,2X,I5)
 12 FORMAT(F8.2).
 14 FORMAT(2I3,F8.2)
 16 FORMAT(14H   I   J   D(I,J))
 24 FORMAT(39HTIME      X1      X2      X3      A1      A2)
    STOP
    END
```

## APPENDIX 6A

```
&JOB;EFR04B;SYSTEM IDENTIFICATION;
&FORTRAN;
&LIST;

      LOGICAL LL(10),LLI(6),LS(2),LLM(2),INP(2),LP
      DIMENSION IS(5)
      COMMON /I/LLI,IS/S/LLS/MU/LLM
C IDENTIFICATION OF THE PARAMETERS OF A FIRST ORDER SYSTEM.
      IA = 999177
      L= 1
      READ(7,10)A,B,X,Z
      DO 1 I=5
      READ(7,11)IS(I)
    1 CONTINUE
      WRITE(2,14)
      DO 2 MM=1,200
      DO 3 NN=1,50
C SOLUTION OF SENSITIVITY EQUATION, ZB.
C THIS EQUATION FORMS THE MODEL O THE SYSTEM.
      ED=2048.0
      DE=IS(2)
      CALL RANNUM(DE,LP,IA,ED)
      LL(2)=LP
      INP(1)=LP
      DE=IS(4)
      CALL RANNUM(DE,LP,IA,ED)
      LL(5)=LP
      INP(2)=LP
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      DE=X
      CALL RANNUM(DE,LP,IA,ED)
      LL(1)=LP
      INP(2)=LP
      J=2
      CALL INT(INP,J,IA)
      IF(IS(2).GT.2148)IS(2)=2148
      IF(IS(2).LT.-2148)IS(2)=-2148
      DE=IS(3)
      CALL RANNUM(DE,L,IA,ED)
      INP(1)=LP
      INP(2)=LL(2)
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      LL(6)=INP(1)
      DE=IS(1)
      CALL RANNUM(DE,LP,IA,ED)
```

```
      INP(2)=LP
      LL(8)=LP
      CALL SUM(INP,L,IA)
      LL(3)=LLS(1)
      ED=128.0
C SOLUTION OF SENSITIVITY EQUATION,ZA.
      DE=IS(5)
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      LL(4)=LP
      INP(2)=LL(5)
      CALL MUL(INP,L)
      INP(1)=.NOT.LLM(1)
      INP(2)=LL(6)
      J=5
      CALL INT(INP,J,IA)
      IF(IS(5).GT.178)IS(5)=178
      IF(IS(5).LT.-178)IS(5)=-178
      INP(1)=LL(4)
      INP(2)=LL(3)
      CALL MULT(INP,L)
C STEEPEST DESCENT CALCULATION TO DETERMINE ALPHA.
      INP(1)=LLM(1)
      INP(2)=LLM(1)
      J=4
      CALL INT(INP,J,IA)
C FIRST ORDER SYSTEM BEING IDENTIFIED.
      ED=2048.0
      DE=B
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      INP(2)=LL(1)
      CALL MULT(INP,L)
      LL(7)=LLM(1)
      INP(1)=LL(8)
      DE=A
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      INP(2)=LL(7)
      J=1
      CALL INT(INP,J,IA)
    3 CONTINUE
      WRITE(2,12)MM,IS(1),IS(2),IS(3),IS(4),IS(5)
    2 CONTINUE
   10 FORMAT(4F8.2)
   11 FORMAT(I5)
   12 FORMAT(I5,2X,I6,2X,I6,2X,I6,2X,I6,2X,I6)
   14 FORMAT(45H TIME      Z      ZB      B      A      ZA)
      STOP
      END
```

APPENDIX 6B

Transient Behaviour of a Parameter Identification Circuit

The transient behaviour of the identification circuit discussed in section 6.2(c) can be analysed.
If $x(t)$ is a step input of value, X, equation (6.2(c).2) has the following solution assuming zero initial conditions:

$$\xi_\beta(t) = \frac{X}{\alpha_{opt}} (1 - e^{-G\alpha_{opt}}), \quad G > 1.0 \qquad \text{---- (6.B.1)}$$

and G is the integrator gain.

From equation (6.2(c).5) we have,

$$\dot{\beta}(t) = -K(m(t) - z(t)) \frac{X}{\alpha_{opt}} (1 - e^{-G\alpha_{opt}t}) \qquad \text{---- (6.B.2)}$$

assuming zero initial conditions. Then,

$$\dot{\beta}(t) = -K(\beta(t) - \beta_{opt}) \frac{X}{\alpha_{opt}} (1 - e^{-G\alpha_{opt}t})^2 \qquad \text{---- (6.B.3)}$$

$$\Rightarrow \dot{\beta}(t) + \beta(t) \frac{X\sqrt{K}}{\alpha_{opt}} (1 - e^{-G\alpha_{opt}t})^2$$

$$= \beta_{opt} \frac{X\sqrt{K}}{\alpha_{opt}} (1 - e^{-G\alpha_{opt}t}) \qquad \text{---- (6.B.4)}$$

and at $t = 0$, $\beta(t) = 0$ and $\dot{\beta}(t) = 0$.

The transient solution is derived from:

$$\dot{\beta}(t) + \beta(t) \frac{X\sqrt{K}}{\alpha_{opt}} (1 - e^{-G\alpha_{opt}t})^2 = 0 \qquad \text{---- (6.B.5)}$$

Let $\beta(t) = Ae^{\lambda(t)}$ ---- (6.B.6)

But $\frac{d(\beta(t))}{dt} = \frac{d\lambda(t)}{dt} \cdot \frac{d(\beta(t))}{d\lambda(t)} = \dot{\lambda}(t) Ae^{\lambda(t)}$

$$\Rightarrow \dot{\lambda}(t)Ae^{\lambda(t)} + Ae^{\lambda(t)} \frac{X\sqrt{K}}{\alpha_{opt}}(1 - e^{-G\alpha_{opt}t})^2 = 0 \qquad \text{---- (6.B.7)}$$

$$\Rightarrow \dot{\lambda}(t) = -\frac{X\sqrt{K}}{\alpha_{opt}}(1 - e^{-G\alpha_{opt}t})^2 \qquad \text{---- (6.B.8)}$$

Integrating over the interval (0,t) and assuming $\lambda(0) = 0$, we have,

$$\lambda(t) = -K\left(\frac{X}{\alpha_{opt}}\right)^2\left(t + \frac{2e^{-G\alpha_{opt}t}}{G\alpha_{opt}} - \frac{e^{-2G\alpha_{opt}t}}{2G\alpha_{opt}} - \frac{3}{2G\alpha_{opt}}\right) \qquad \text{---- (6.B.9)}$$

$$\Rightarrow \beta(t) = A\exp\left[-K\left(\frac{X}{\alpha_{opt}}\right)^2\left(t + \frac{2e^{-G\alpha_{opt}t}}{G\alpha_{opt}} - \frac{e^{-2G\alpha_{opt}t}}{2G\alpha_{opt}} - \frac{3}{2G\alpha_{opt}}\right)\right] \qquad \text{---- (6.B.10)}$$

The steady state solution is $\beta(t) = \beta_{opt}$, hence,

$$\beta(t) = \beta_{opt}\left[1 - \exp\left[-K\left(\frac{X}{\alpha_{opt}}\right)^2\left(t + \frac{2e^{-G\alpha_{opt}t}}{G\alpha_{opt}} - \frac{e^{-2G\alpha_{opt}t}}{2G\alpha_{opt}} - \frac{3}{2G\alpha_{opt}}\right)\right]\right] \qquad \text{---- (6.B.11)}$$

The observed transient response agrees with equation (6.B.11).

APPENDIX 6C

I Non-Linear Noise Addie

The analysis of the linear system presented in APPENDIX 6B suggests that it may be possible to produce output interfaces with improved bandwidth characteristics by employing non-linear filters. One such filter is illustrated in Figure 6C(a) and it is the same as a conventional noise ADDIE except that there is an m bit delay in the feedback loop which introduces a time delay of $m/f_c$. This circuit is described by the following differential equation:

$$\frac{\dot{E}_o(t)}{G} = -(F(t) - E_{in}(t)) \qquad \text{---- (6.C.1)}$$

Let H(t) be the Heaviside unit step function and let $a = mT_c$. Then F(t) is defined as:

$$\begin{aligned} F(t) = {} & f_1(t)\,[H(t) - H(t-a)] \\ & + f_2(t-a)\,[H(t-a) - H(t-2a)] \\ & + f_3(t-2a)\,[H(t-2a) - H(t-3a)] \\ & + f_4(t-3a)\,[H(t-3a) - H(t-4a)] \\ & + f_5(t-4a)\,[H(t-4a) - H(t-5a)] \\ & + f_6(t-5a)\,[H(t-5a) - H(t-6a)] + \ldots \\ & \ldots + f_j(t-(j-1)a)\,[H(t-(j-1)a) - H(t-ja)] + \ldots \end{aligned} \qquad \text{---- (6.C.2)}$$

$$= \sum_{i=1}^{\ell} f_i(t-(i-1)a)\,[H(t-(i-1)a) - H(t-ia)] \qquad \text{---- (6.C.3)}$$

and $$\lim_{i\to\infty} F(t) = E_{in} \qquad \text{---- (6.C.4)}$$

ie /

ie, if

$$f_j(a) - f_{j-1}(a) = 0 \text{ and } \dot{E}_0(t) = 0 \qquad \text{---- (6.C.5)}$$

Thus the series is convergent if

$$\left|f_{j+1}(a) - f_j(a)\right| < \left|f_j(a) - f_{j-1}(a)\right| \qquad \text{---- (6.C.6)}$$

II Example

Suppose the counter has N states, m = N/2, and $E_{in}(t)$ is a step input. Initially, the counter is set to zero and the m bit shift register is empty. Hence,

$$a = \frac{NT_c}{2} \qquad \text{---- (6.C.7)}$$

and

$$G = \frac{f_c}{N} \qquad \text{---- (6.C.8)}$$

$$\Rightarrow \quad aG = 1/2 \qquad \text{---- (6.C.9)}$$

Time period 0 - a

$$\frac{\dot{E}_0(\tau)}{G} = E_{in} - 0$$

$$\Rightarrow \quad E_0(\tau) = G\,\tau E_{in}$$

and at $\tau = a$, $E_0(a) = \frac{E_{in}}{2}$

Function stored in register at $\tau = a$ is $G\tau E_{in} + 0$

Time period a - 2a

$$\frac{\dot{E}_0(\tau)}{G} = E - G\,\tau\,E_{in}$$

$$\Rightarrow \quad E_0(\tau) = G\,\tau\,E_{in} - \frac{(G\tau)^2}{2} E_{in} + C_a$$

and at $\tau = 2a$, $E_0(2a) = \frac{E_{in}}{2} - \frac{E_{in}}{8} + C_a = \frac{3}{8}E_{in} + \frac{E_{in}}{2} = \frac{7E_{in}}{8}$

Function /

Function stored in the shift register at $\tau = 2a$ is

$$G\tau E_{in} - \frac{(G\tau)^2}{2}E_{in} + C_a$$

Time period 2a - 3a

$$\frac{\dot{E}_O}{G}(\tau) = \frac{E_{in}}{2} - G\tau E_{in} + \frac{(G\tau)^2}{2}E_{in}$$

$$\Rightarrow \quad E_O(\tau) = \frac{G\tau E_{in}}{2} - \frac{(G\tau)^2}{2}E_{in} + \frac{(G\tau)^3}{6}E_{in} + C_{2a}$$

and at $\tau = 3a$, $E_O(3a) = \frac{E_{in}}{4} - \frac{E_{in}}{8} + \frac{E_{in}}{48} + \frac{7}{8}E_{in} = 1.021E_{in}$

Function stored in register at $\tau = 3a$ is

$$\frac{G\tau}{2}E_{in} - \frac{(G\tau)^2}{2}E_{in} + \frac{(G\tau)^3}{6}E_{in} + C_{2a}$$

Time period 3a - 4a

$$\frac{\dot{E}_O}{G}(\tau) = \frac{E_{in}}{8} - \frac{G\tau}{2}E_{in} + \frac{(G\tau)^2}{2}E_{in} - \frac{(G\tau)^3}{6}E_{in}$$

and

$$E_O(\tau) = \frac{G\tau}{8}E_{in} - \frac{(G\tau)^2}{4}E_{in} + \frac{(G\tau)^3}{6}E_{in} - \frac{(G\tau)^4}{24}E_{in} + C_{3a}$$

and at $\tau = 4a$, $E_O(4a) = 1.04E_{in}$

Function stored in the register is:

$$\frac{G\tau E_{in}}{8} - \frac{(G\tau)^2}{4}E_{in} + \frac{(G\tau)^3}{6}E_{in} - \frac{(G\tau)^4}{24}E_{in} + \frac{49E_{in}}{48}$$

Similarly, $C_{5a} = 1.02175\ E_{in}$ $C_{6a} = 1.00565\ E_{in}$, $C_{7a} \triangleq E_{in}$. These values of integrator state were plotted against /

against time in Figure 6.C.(b). The time response of the linear noise ADDIE is presented in the same diagram for comparison. The non-linear filter clearly converges to the same value as the input and convergence is independent of the initial state of the counter so that this circuit is an adaptive element, ie, it is an ADDIE.

The response curve shows that there is a 4% overshoot if the shift register length is half the number of counter states. If the delay is increased the overshoot will be greater. On the other hand, as the delay is decreased the response of the non-linear filter approaches that of a conventional noise ADDIE. There will be one critical delay for which there is no overshoot. During the first time period the output is a ramp while during the second interval the response is quadratic. Similarly, successive segments of the output are cubic, quartic, fifth power, etc. Thus the device generates a convergent series which eventually cancels the input. The graph shows that the non-linear response is continuous, ie,

$$\dot{f}_j(0) = \dot{f}_{j-1}(a) \qquad \text{---- (6.C.10)}$$

This non-linear ADDIE has a much greater bandwidth than the noise ADDIE. The bandwidth has been increased by a factor of 1.58.

Since this ADDIE is a sequential network we can study its behaviour using a discrete time analysis[9]. Let $n(t)$ be the state of the non-linear ADDIE after t steps and let $d(t)$ be the contents of the last stage of the shift register after the same t steps. We assume that initially the shift register is empty and the UP/DOWN counter is zeroed. The expected change in the ADDIE state between steps (t-1) and t is given by:

$$E\{n(t) - n(t-1)\} = [\text{probability of counting up}] - [\text{probability of counting down}] \qquad \text{---- (6.C.11)}$$

$$= p\{\frac{N - d(t-1)}{N}\} - q\{\frac{d(t-1)}{N}\} \qquad \text{---- (6.C.12)}$$

$$= p - \frac{d(t-1)}{N} \qquad \text{---- (6.C.13)}$$

where p is the input probability and q = (1-p).
The average value of n(t) is:

$$n(0) + E\{n(1) - n(0)\} + E\{n(2) - n(1)\} + \ldots\ldots + E\{n(t) - n(t-1)\} \qquad \text{---- (6.C.14)}$$

Time Period ($0 \leq \tau \leq N/2$)

The expected state of the ADDIE after the first step is

$$E\{n(1)\} = E\{n(0)\} + p - \frac{E\{d(0)\}}{N}$$

$$= p + n(0) - \frac{d(0)}{N}$$

$$E\{n(2)\} = E\{n(1)\} + p - \frac{E\{d(1)\}}{N}$$

$$= 2p + n(0) - \frac{d(0) + d(1)}{N}$$

Generalising

$$E\{n(\tau)\} = \tau p + n(0) - \frac{\sum_{t=0}^{\tau-1} d(t)}{N}$$

$$\frac{N}{2}-1$$

$$\text{at } \tau = N/2, \quad E\{n(\frac{N}{2})\} = \frac{Np}{2} + n(0) - \frac{\sum_{\tau=0} d(\tau)}{N}$$

Let n(0) = 0 and the $d(\tau) = 0$, ie, the contents of the shift register reflects the contents of the counter initially.

$E\{n(\frac{N}{2})\}$ /

$$\Rightarrow \quad E\{n(\tfrac{N}{2})\} = \frac{Np}{2}$$

ie, $c_a = \frac{Np}{2}$

Hence, the probability function distributed in the shift register during the time interval (O,N/2) is

$$d(\tau) = p\tau$$

Time Period $(N/2 \leq \tau \leq N)$

$$E\{n(\tfrac{N}{2}+2)\} = E\{n(\tfrac{N}{2}+1)\} + p - \frac{2p}{N}$$

$$= \frac{Np}{2} + 2p - \frac{3p}{N}$$

$$E\{n(\tfrac{N}{2}+3)\} = E\{n(\tfrac{N}{2}+2)\} + p - \frac{E\{d(\frac{N}{2}+2)\}}{N}$$

$$= \frac{Np}{2} + 2p - \frac{3p}{N} + p - \frac{3p}{N}$$

$$= \frac{Np}{2} + 3p - \frac{6p}{N}$$

Generalising

$$E\{n(\tfrac{N}{2}+\tau)\} = c_a + \tau p - \frac{\tau p}{2N}(\tau+1)$$

$$\Rightarrow \quad E\{n(N)\} = E\{n(N-1)\} + p - \frac{E\{d(N-1)\}}{N}$$

$$= \frac{Np}{2} + \frac{Np}{2} - \frac{p}{N}\sum_{j=1}^{N/2} j$$

$$= \frac{Np}{2} + \frac{Np}{2} - \frac{p}{N}\cdot\frac{N}{4}(\tfrac{N}{2}+1)$$

$$\Rightarrow \quad E\{n(N)\} \doteq Np - \frac{Np}{8} = \frac{7}{8}Np \quad \text{if } \frac{N}{2} \text{ is very large.}$$

$$\Rightarrow \quad c_{2a} \doteq \frac{7}{8}Np$$

Hence /

Hence the probability function distributed along the shift register during the time interval (N/2,N) is

$$d(\tau) = c_a + p\tau - \frac{p\tau}{2N}(\tau+1)$$

Time Period ($N \leq \tau \leq 3N/2$)

$$E\{n(N+2)\} = E\{n(N+1)\} + p - \frac{E\{d(N+1)\}}{N}$$

$$= \frac{7}{8}Np + 2p - \left(\frac{\frac{Np}{2} + 3p - \frac{4p}{N}}{N}\right)$$

$$E\{n(N+3)\} = E\{n(N+2)\} + p - \frac{E\{d(N+2)\}}{N}$$

$$\frac{7}{8}Np + 2p - \left(\frac{\frac{Np}{2} + 3p - \frac{4p}{N}}{N}\right) + p - \left(\frac{\frac{Np}{2} + 3p - \frac{6p}{N}}{N}\right)$$

$$= \frac{7}{8}Np + 3p - \left(\frac{\frac{3Np}{2} + 6p - \frac{10p}{N}}{N}\right)$$

Generalising

$$E\{n(N+\tau)\} = \frac{7}{8}Np + \tau p - \frac{\tau Np}{2N} - \frac{\tau p}{2N}(\tau+1) + \frac{\tau p}{6N^2}(\tau+2)(\tau+1)$$

Hence,

$$E\{n(\tfrac{3}{2}N)\} = E\{n(\tfrac{3N}{2}-1)\} + p - \frac{E\{d(\frac{3N}{2}-1)\}}{N}$$

$$E\{n(\tfrac{3}{2}N)\} \triangleq \frac{7}{8}Np + \frac{Np}{2} - \frac{Np}{4} - \frac{Np}{8} + \frac{Np}{48} = 1.021Np$$

$\Rightarrow$ $c_{3a} \triangleq 1.021Np$ if $\frac{N}{2}$ is large.

Hence the probability fundion distributed in the shift register during the time interval (N,3N/2) is:

$$d(\tau) = c_{2a} + \frac{\tau p}{2} - \frac{\tau p}{2N}(\tau+1) + \frac{\tau p}{6N^2}(\tau+2)(\tau+1)$$

Time /

Time Period $(\frac{3}{2}N \leq \tau \leq 2N)$

$$E\{n(\tfrac{3}{2}N+2)\} = E\{n(\tfrac{3}{2}N+1)\} + p - \frac{E\{d(\frac{3}{2}N+1)\}}{N}$$

$$= 1.021Np + 2p - \left(\frac{2\times\frac{7}{8}Np + 3P/2 - 4P/N + 5P/N^2}{N}\right)$$

$$E\{n(\tfrac{3}{2}N+3)\} = E\{n(\tfrac{3}{2}N+2)\} + p - \frac{E\{d(\frac{3}{2}N+2)\}}{N}$$

$$= 1.021Np + 3p - \left(\frac{2\times\frac{7}{8}Np + 3P/2 - 4P/N + 5P/N^2}{N}\right) -$$

$$- \left(\frac{\frac{7}{8}Np + \frac{3p}{2} - \frac{6p}{N} + \frac{10p}{N^2}}{N}\right)$$

$$\triangleq 1.021Np + 3p - \left(\frac{\frac{21}{8}Np + \frac{3p}{1} - \frac{10p}{N} + \frac{15p}{N^2}}{N}\right)$$

$$\Rightarrow E\{n(\tfrac{3}{2}N + \tau)\}$$

$$\triangleq 1.021Np + \tau p - \frac{\frac{7p\tau}{8} + \frac{\tau p}{4}(\tau+1) - \frac{\tau p}{6N}(\tau+2)(\tau+1)}{N}$$

$$- \frac{\tau p}{24N^3}(\tau+3)(\tau+2)(\tau+1)$$

Hence,

$$E\{n(2N)\} = E\{n(2N-1)\} + p - \frac{E\{d(2N-1)\}}{N}$$

$$\triangleq 1.021Np + \frac{Np}{2} - \frac{7Np}{16} - \frac{Np}{16} + \frac{Np}{48} + 0 \quad \text{if} \quad \frac{N}{2} >> 3$$

$$\triangleq 1.021Np + \frac{Np}{48} \triangleq 1.042Np$$

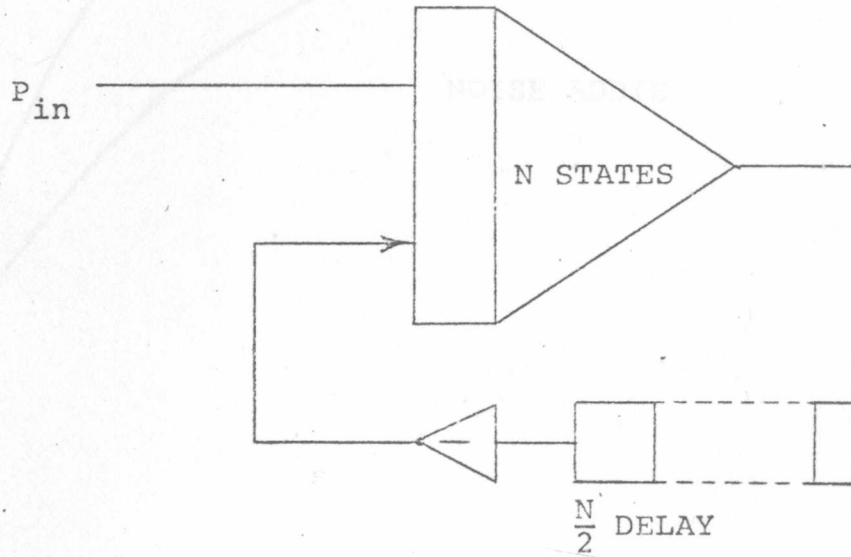
ie $c_{4a} \triangleq 1.042Np$, $\frac{N}{2}$ large.

Similarly, $c_{5a} \triangleq 1.02175Np$, $c_{6a} \triangleq 1.00565Np$ and $c_{7a} \triangleq Np$. The $c_{ja}$ of the discrete time analysis are virtually identical to those of the continuous time analysis provided that N/2 is large.

A /

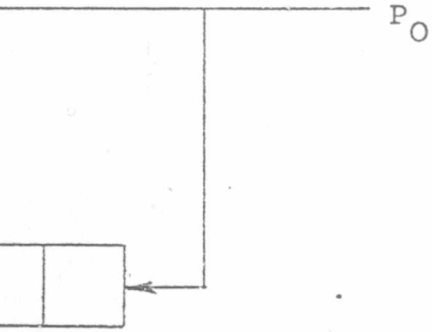
A digital computer simulation of the non-linear ADDIE was performed and the programme is listed in Figure 6CP(d). The response of the ADDIE to a range of input probabilities is shown in Figure 6C(c). The results clearly agree with the theoretical values except that random variance tended to obscure the overshoot predicted by the two analyses. Statistical tests were carried out on the results of the simulations and these results are presented in Figure 6C(e). The statistical tests show that the ADDIE states are distributed binomially about the mean, the value of which is predicted by the above analyses.

The final behaviour of the non-linear ADDIE is independent of its initial state. This is illustrated in Figure 6C(f) where initially the probability stored in the shift register is 0.25 and the initial counter state is 1024. In the steady state the output probability is equal to that of the input as predicted. Even if the contents of the shift register does not reflect that of the counter initially, the output of the ADDIE will still follow the input in the steady state. This is demonstrated in Figure 6C(g) where initially the counter is zeroed and the probability stored in the shift register is 0.25. The output probability eventually converges to 0.5. Thus the non-linear element described above is adaptive.



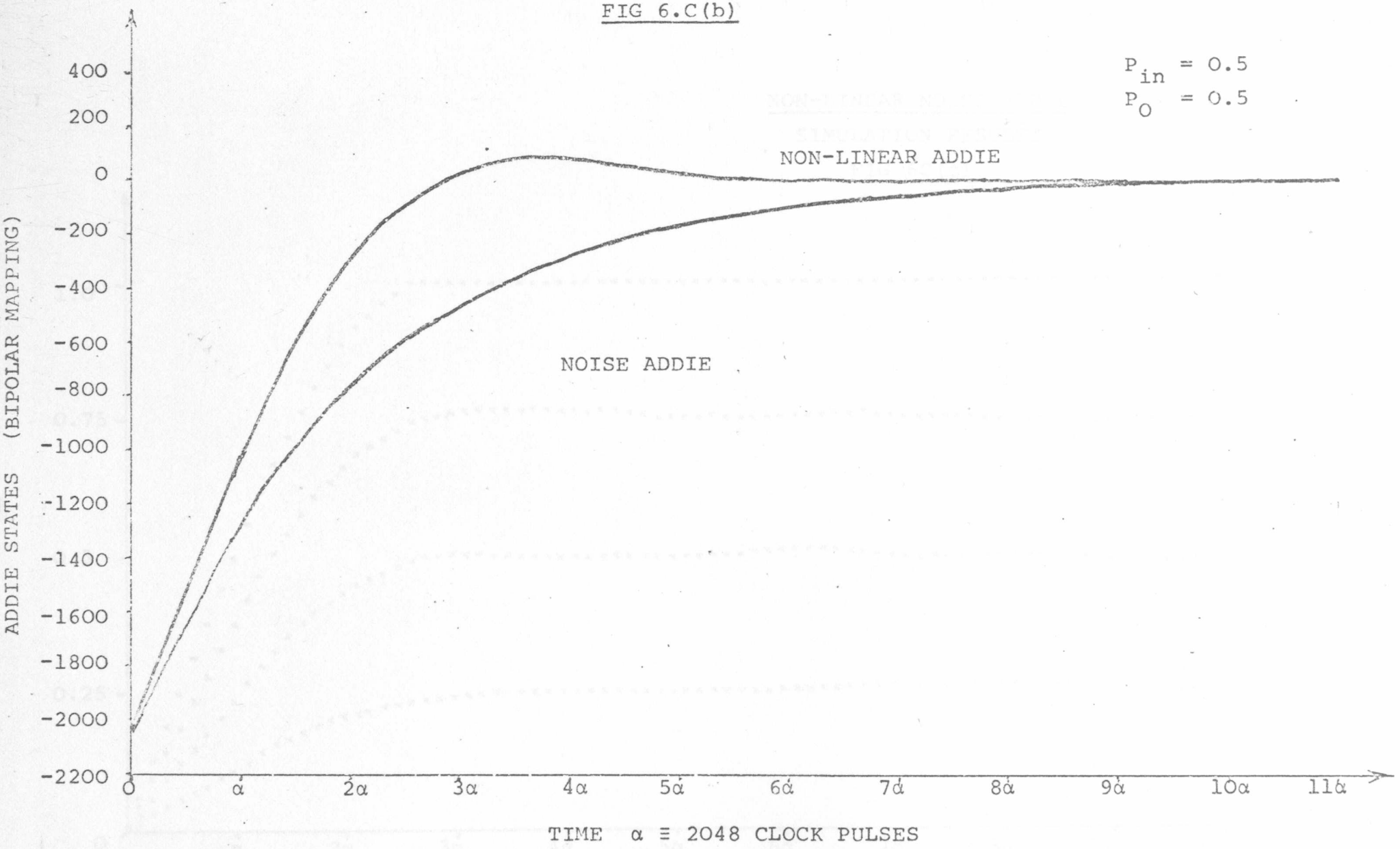

NON-LINEAR NOISE ADDIE

FIG 6.C(a)


$P_O$

NON-LINEAR NOISE ADDIE
THEORETICAL RESPONSE

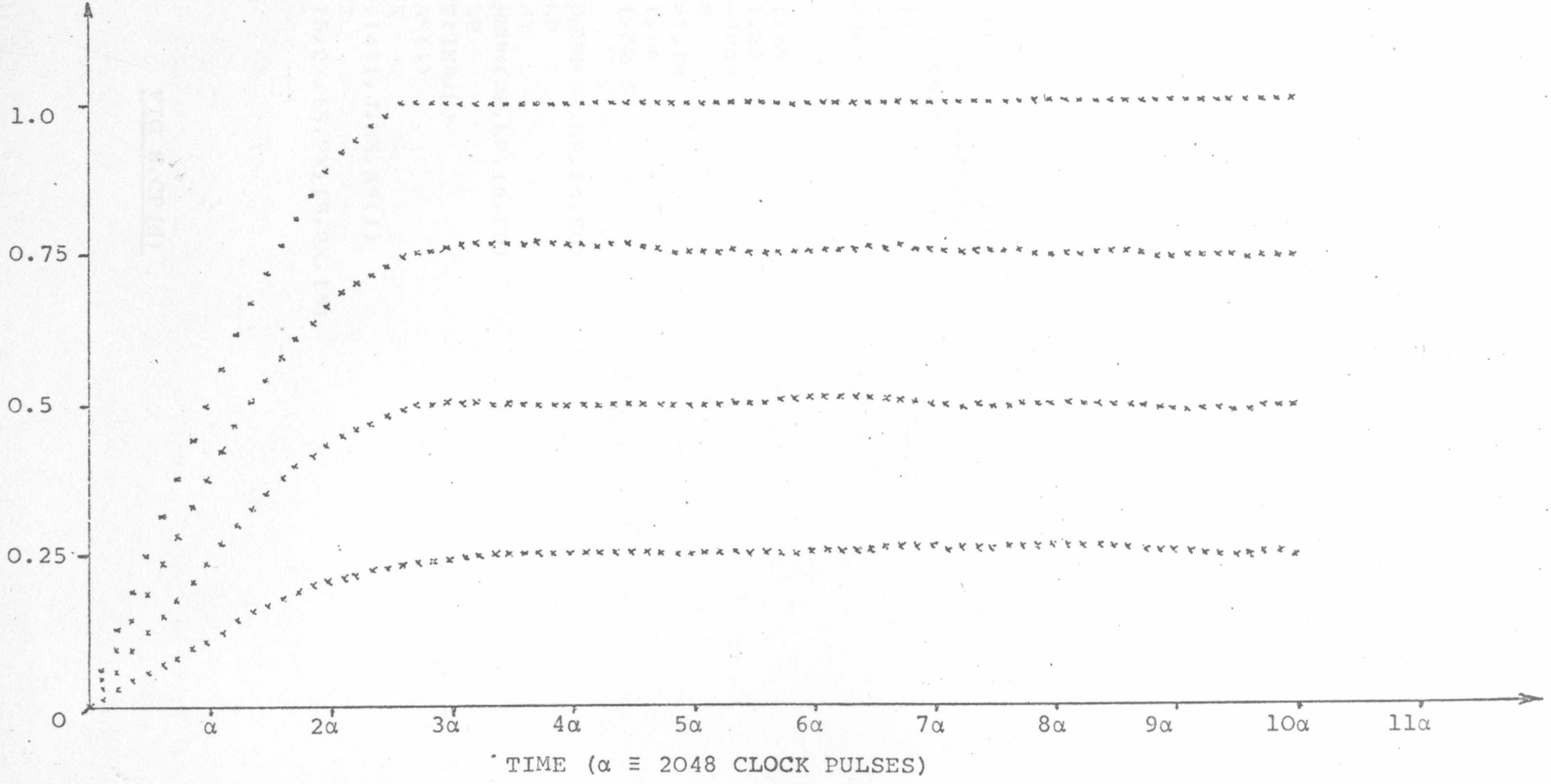
FIG 6.C(b)



NON-LINEAR NOISE ADDIE

SIMULATION RESULTS

FIG 6C(c)



```
&JOB;EER04I;NON-LINEAR ADDIE;
&FORTRAN;
&LIST;

      LOGICAL LLA(5),INP(2),LP
      INTEGER A(64,32),AS(5)
      COMMON /0/LLA,AS
      IA=717311
      ED=2048.0
      AS(1)=-2048
      L=1
      IP=0
      DO 6 I=1,64
      DO 6 J=1,32
      A(I,J)=-2048
    6 CONTINUE
      DO 1 MM=1,10
      DO 1 I=1,64
      DO 3 J=1,32
      DE=IP
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      DE=A(I,J)
      CALL RANNUM(DE,LP,IA,ED)
      LLA(1)=LP
      CALL OUT(INP,L)
      A(I,J)=AS(1)
    3 CONTINUE
      WRITE(2,14)I,J,MM,AS(1)
    1 CONTINUE
   14 FORMAT(I5,2X,I5,2X,I5,2X,I5)
      STOP
      END
```

FIG 6.CP(d)

| Input Probability | Output No. of States | | Variance | | Standard Deviation | |
|---|---|---|---|---|---|---|
| | Theoretical | Non-Lin ADDIE | Theoretical | Non-Lin ADDIE | Theoretical | Non-Lin ADDIE |
| 0.25 | 1024 | 1039.04 | 768 | 627.42 | 27.7 | 25.05 |
| 0.5 | 2048 | 2045.07 | 1024 | 839.69 | 32.0 | 28.98 |
| 0.75 | 3072 | 3089.18 | 768 | 798.97 | 27.7 | 28.27 |
| 1.0 | 4096 | 4096 | 0 | 0 | 0 | 0 |

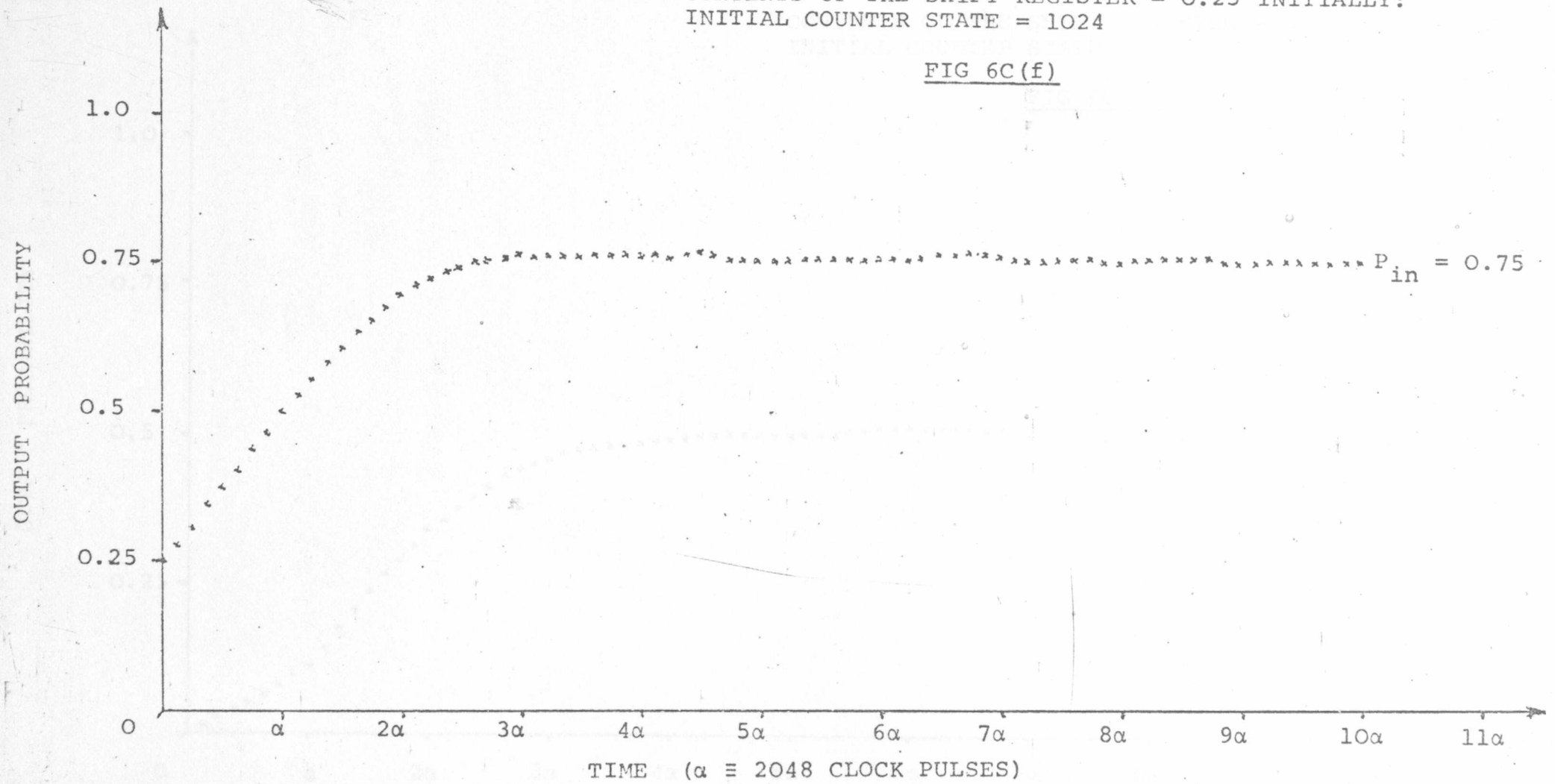
RESULTS OF STATISTICAL TESTS ON THE NON-LINEAR NOISE ADDIE

FIG 6C(e)

NON-LINEAR NOISE ADDIE

CONTENTS OF THE SHIFT REGISTER = 0.25 INITIALLY:
INITIAL COUNTER STATE = 1024

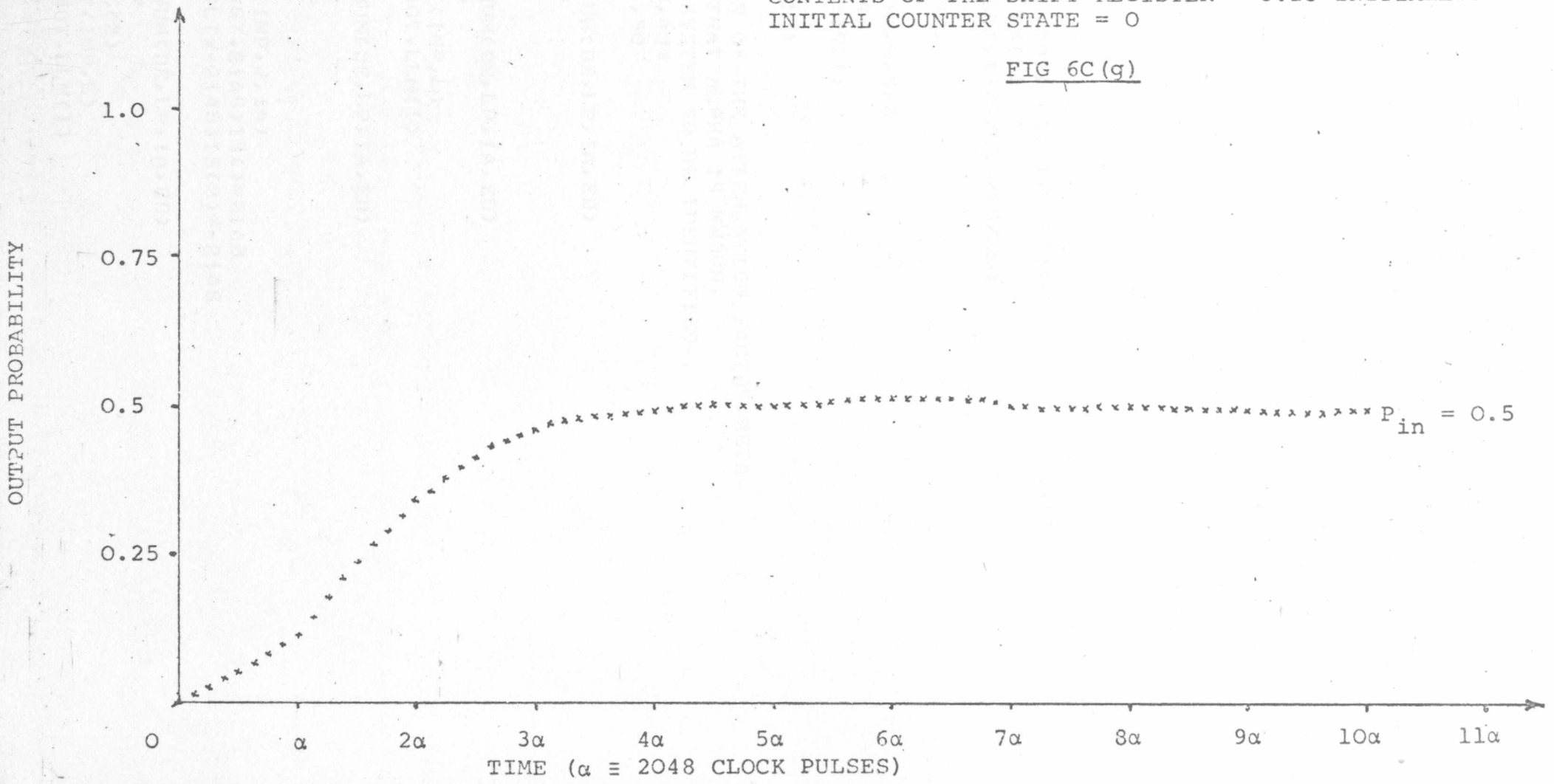
FIG 6C(f)



NON-LINEAR NOISE ADDIE

CONTENTS OF THE SHIFT REGISTER = 0.25 INITIALLY:
INITIAL COUNTER STATE = 0

FIG 6C(g)



## APPENDIX 6D

```
&JOB;EER04C;SYSTEM IDENTIFICATION
&FORTRAN;
&LIST;

      LOGICAL LL(10),LLI(6),LLS(2),LLM(2),INP(2),LP
      DIMENSION IS(5)
      COMMON /I/LLI,IS/S/LLS/MU/LLM
      IA=111773
      LL(9)=.TRUE.
      L=1
      READ(7,10)A,B,X,Z
      DO 1 I=1,4
      READ(7,11)IS(I)
    1 CONTINUE
      WRITE (2,14)
      ED=2048.0
C IDENTIFICATION OF THE ATTENUATION FACTOR,BETA.
C THIS ASSUMES THAT ALPHA IS KNOWN.
C MODEL OF THE SYSTEM TO BE IDENTIFIED.
      DO 2 MM=1,400
      DO 3 NN=1,50
      DE=IS(2)
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      LL(2)=LP
      DE=IS(4)
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      DE=X
      CALL RANNUM(DE,LP,IA,ED)
      LL(1)=LP
      INP(2)=LP
      J=2
      CALL INT(INP,J,IA)
      IF(IS(2).GT.2148)IS()=2148
      IF(IS(2).LT.-2148)IS(2)=-2148
      DE=IS(3)
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      INP(2)=LL(2)
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      DE=IS(1)
```

```
      CALL RANUMM(DE,LP,IA,ED)
      INP(2)=LP
      LL(8)=LP
      CALL SUM(INP,L,IA)
C STEEPEST DESCENT CALCULATION TO DETERMINE BETA.
      INP(1)=LLS(1)
      INP(2)=LL(9)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LLM(1)
      J=3
      CALL INT(INP,J,IA)
      LL(9)=LL(2)
      DE=B
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      INP(2)=LLM(1)
      CALL MULT(INP,L)
      LL(7)=LLM(1)
      INP(1)=LL(8)
      DE=A
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      INP(2)=LL(7)
      J=1
      CALL INT(INP,J,IA)
    3 CONTINUE
      WRITE(2,12)MM,IS(1),IS(2),IS(3),IS(4)
    2 CONTINUE
   10 FORMAT(4F8.2)
   11 FORMAT(I5)

   12 FORMAT(I5,2X,I6,2X,I6,2X,I6,2X,I6)
   14 FORMAT(37H TIME      Z      ZB      B      A)
      STOP
      END
```

## APPENDIX 6E

```
&JOB;EER04E;SIMPLIFIED IDENTIFICATION CIRCUIT;
&FORTRAN;
&LIST;

      LOGICAL LL(11),LLI(6),LLM(2),LLS(2),INP(2),LP
      DIMENSION P(4),IS(5)
      COMMON /I/LLI,IS/S/LLS/MU/LLM
      L=1
      IA=733311
      WRITE(2,14)
      READ(7,10)P(1),P(2),P(3)
      DO 1 I=1,5
      READ(7,11)IS(I)
    1 CONTINUE
      ED=2048.0
      DO 4 MM=1,400
      DO 3 NN=1,50
      DO 6 I=1,5
      DE=IS(I)
      CALL RANNUM(DE,LP,IA,ED)
      LL(I)=LP
    6 CONTINUE
C SIMULATION OF A FIRST ORDER SYSTEM.
      DE=P(3)
      CALL RANNUM(DE,LP,IA,ED)
      LL(10)=LP
      DE=P(2)
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      INP(2)=LL(10)
      CALL MULT(INP,L)
      LL(9)=LLM(1)
      DE=P(1)
      CALL RANNUM(DE,LP,IA,ED)
      INP(2)=LP
      INP(1)=LL(1)
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      INP(2)=LL(9)
      J=1
      CALL INT(INP,J,IA)
```

```
C MODEL OF THE FIRST ORDER SYSTEM.
      INP(1)=LL(2)
      INP(2)=LL(4)
      CALL MULT(INP,L)
      INP(1)=.NOT.LLM(1)
      INP(2)=LL(10)
      J=2
      CALL INT(INP,J,IA)
      INP(1)=LL(2)
      INP(2)=LL(3)
      CALL MULT(INP,L)
      LL(6)=.NOT.LLM(1)
      INP(1)=LL(6)
      INP(2)=LL(1)
      CALL SUM(INP,L,IA)
      INP(2)=.NOT.LL(5)
      J=5
      CALL INT(INP,J,IA)
      LL(7)=.TRUE.
      IF(IS(5).LT.0)LL(7)=.FALSE.
      INP(1)=LL(7)
      INP(2)=LLS(1)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LLM(1)
      J=4
      CALL INT(INP,J,IA)
      DO 3 I=1,5
      IF(IS(I).GT.2148)IS(I)=2148
      IF(IS(I).LT.-2148)IS(I)=-2148
    3 CONTINUE
      WRITE(2,12)MM,IS(1),IS(2),IS(3),IS(4),IS(5)
    4 CONTINUE
   10 FORMAT(3F8.2)
   11 FORMAT(I5)
   12 FORMAT(I4,2X,I5,2X,I5,2X,I5,2X,I5,2X,I5)
   14 FORMAT(4HTIME      Z      ZB      B      A SGN(-M))
      STOP
      END
```

APPENDIX 6F

```
&JOB;EER04F;SECOND ORDER SYSTEM IDENTIFICATION;
&FORTRAN;
&LIST;

      LOGICAL LL(20),LX(4),LLI(10),LLM(2),LLS(2),INP(2),LSW(2),LP
      DIMENSION IS(10),P(5)
      COMMON /I/LLI,IS/MU/LLM/S/LLS
C IDENTIFICATINN OF THE PARAMETERS OF A SECOND ORDER SYSTEM.
      L=1
      LL(6)=.TRUE.
      IA=111137
      READ(7,29)P(1),P(2),P(3),P(4)
      READ(7,28)IS(1),IS(2),IS(3),IS(4),IS(5),IS(7),IS(9)
      WRITE(2,27)
C MAIN PROGRAMME STARTS.
      DO 4 MM=1,200
      DO 3 NN=1,50
      ED=4096
      DE=IS(9)
      CALL RANNUM(DE,LP,IA,ED)
      LL(9)=LP
      ED=1024
      DO 70 J=1,7
      IF(J.EQ.6)GOTO 70
      DE=IS(J)
      CALL RANNUM(DE,LP,IA,ED)
      LL(J)=LP
   70 CONTINUE
C THREE PARAMETERS IN THIS SECOND ORDER SYSTEM.
      DO 6 J=1,4
      DE=P(J)
      CALL RANNUM(DE,LP,IA,ED)
      LX(J)=LP
    7 CONTINUE
C SYSTEM TO BE IDENTIFIED.
      INP(1)=LX(1)
      INP(2)=LL(1)
      CALL MULT(INP,L)
      LL(20)=LLM(1)
      INP(1)=LX(2)
      INP(2)=LL(2)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LL(20)
```

APPENDIX 6F CONTINUED

```
      CALL SUM(INP,L,IA)
      LL(12)=.NOT.LLS(1)
      INP(1)=LX(4)
      INP(2)=LX(3)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LL(12)
      J=1
      CALL INT(INP,J,IA)
      INP(1)=LL(1)
      INP(2)=LL(1)
      J=2
      CALL INT(INP,J,IA)
C MODEL OF THE SYSTEM TO BE IDENTIFIED.
      INP(1)=LL(9)
      INP(2)=LL(3)
      CALL MULT(INP,L)
      LL(14)=LLM(1)
      INP(2)=LL(7)
      INP(1)=LL(4)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LL(14)
      CALL SUM(INP,L,IA)
      LL(15)=.NOT.LLS(1)
      ED=2048.0
      DE=1024.0
      CALL RANNUM(DE,LP,IA,ED)
      INP(1)=LP
      INP(2)=LX(4)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LL(15)
      J=3
      CALL INT(INP,J,IA)
      INP(1)=LL(3)
      INP(2)=LL(3)
      J=4
      CALL INT(INP,J,IA)
      INP(1)=LL(4)
      INP(2)=LL(5)
      CALL MULT(INP,L)
      LL(17)=.NOT.LLM(1)
      INP(1)=LL(17)
      INP(2)=LL(2)
C FORMATION OF THE ERROR.
      CALL SUM(INP,L,IA)
      LL(19)=LLS(1)
C THE SIGN OF ALPHA IS ASSUMED TO BE POSITIVE.
C SOLUTIONS OF THE STEEPEST DESCENT EQUATIONS.
      INP(1)=LL(19)
C USE A ONE BIT DELAYED SEQUENCE TO REPRESENT -M.
```

```
      INP(2)=.NOT.LL(6)
      CALL MULT(INP,L)
      LL(6)=LL(4)
      INP(1)=LLM(1)
      INP(2)=LLM(1)
      J=7
      CALL INT(INP,J,IA)
      INP(1)=LL(1))
      INP(2)=.NOT.LL(3)
      CALL MULT(INP,L)
      INP(1)=LLM(1)
      INP(2)=LLM(2)
      J=9
      CALL INT(INP,J,IA)
C THIS ROUTINE IMPOSES LIMITING ON THE INTEGRATORS.
      DO 3 J=1,7
      IF(J.EQ.6)GOTO 3
      IF(IS(J).GT.1024)IS(J)=1024
      IF(IS(J).LT.-1024)IS(J)=-1024
    3 CONTINUE
      WRITE(2,30)IS(1),IS(2),IS(3),IS(4),IS(5),IS(7),IS(9)
    4 CONTINUE
   29 FORMAT(4F8.2)
   28 FORMAT(7I5)
   27 FORMAT(5HTIME      DZ       Z     DZG      ZG       G       B
   30 FORMAT(I4,2X,I5,2X,I5,2X,I5,2X,I5,2X,I5,2X,I5,2X,I5,2X,I5)
      STOP
      END
```

## APPENDIX 7A

```
  5 PRI "**NON STATIONARY MARKOV CHAIN SIMULATOR**"
 10 PRI
 15 REM READ IN THE STARTING VECTOR.
 20 READ N,G
 25 FOR J = 1 TO N
 30 READ X(J)
 35 NEX J
 40 REM READ IN THE ROWS OF THE STOCHASTIC MATRIX "A"
 45 FOR I = 1 TO N
 50 FOR J = 1 TO N
 55 READ A(I,J)
 60 NEX J
 65 NEX I
 70 REM READ IN THE ROWS OF THE STOCHASTIC MATRIX "B"
 75 FOR I = 1 TO N
 80 FOR J = 1 TO N
 85 READ B(I,J)
 90 NEX J
 95 NEX I
100 REM FORMULATE THE COMPOSITE STOCHASTIC MATRIX "C"
105 PRI " INPUT THE NUMBER OF ITERATION STEPS " \ INP S
110 FOR L = 1 TO S
115 FOR I = 1 TO N
120 FOR J = 1 TO N
125 C(I,J) = A(I,J) + B(I,J) * FNA(D)
130 NEX J
135 NEX I
136 DEF FNA (Y) = EXP(-G*L)
140 REM VECTOR - MATRIX MULTIPLICATION.
145 FOR J = 1 TO N
146 FOR K = 1 TO N
150 S(J) = S(J) + X(K) * C(K,J)
155 NEX K
160 NEX J
165 FOR K = 1 TO N
170 X(K) = S(K)
180 NEX K
185 REM PRINT THE ITERATION STEP AND THE STATE VECTOR.
190 PRI L,S(1),S(2),S(3),S(4)
191 FOR K = 1 TO N
192 S(K) = 0
193 NEX K
195 NEX L
220 DATA 3,0.1
225 DATA 1,0,0
230 DATA 0.4,0.35,0.25,0.2,0.6,0.2,0.3,0.3,0.4
235 DATA 0.4,-0.3,-0.1,-0.1,0.3,-0.2,0.2,0.2,-0.4
```

# BIBLIOGRAPHY


1. ALT, F L and RUBINOFF, M; 'Advances in Computers'; Vol 9, Academic Press, 1968.

2. GAINES, B R; 'Varieties of Computer - Their Applications and Inter-Relationships';

3. HOEL, P G; 'Introduction to Mathematical Statistics'; 4th Edition, Wiley International Edition.

4. PAPOULLIS; 'Probability, Random Variables and Stochastic Processes'; McGraw-Hill.

5. MARS, P; 'Digital Stochastic Computation'; UKSC, Seminar meeting, University of Glasgow, 1972.

6. GAINES, B R; 'Stochastic Computing Systems'; Advances in Information System Science, Vol 2 pp 37-172; Plenium Press, 1969.

7. GAINES, B R; 'Stochastic Computing'; Spring Joint Computer Conference; STL, 1967.

8. GAINES, B R; 'Stochastic Computer Thrives on Noise'; Advanced Technology, Electronics; July 10th, 1967.

9. MILLER, BROWN and MARS; 'Study of an Output Interface for a Digital Stochastic Computer'; Int. Journal of Electronics, pp 637-655, Vol 37, nO 5, 1974.

10. RIBEIRO, S T; 'Random Pulse Machines'; IEEE Transactions on Electronic Computers; Vol EC-16, No 3, June, 1967.

11. MARTIN, C; 'A Pseudorandom Number Generator'; Third year project report; University of Essex, April, 1972.

12. OHTERU, S, KATO, T, NUSHIHARA, Y, and KINOUCHI, Y; 'Stochastic Differential Analyzer'; Dept of Applied Physics, Woseda University, Tokyo, Japan.

13. MARS, P; 'Proposal for a Universal Stochastic Module'; Internal Paper, RGIT, Aberdeen.

14. GOLOMB, S W; 'Shift Register Sequences'; Holden Day Inc. Part I, pp 1-19, Part II pp 22-37.

15. DIGITAL CORP; 'Introduction to Programming'; PDP-8 Handbook, Vol 1.

16. DIGITAL CORP; 'Programming Languages'; PDP-8 Handbook, Vol 2.

17. /

17. O'BRIEN J, and WHITEHEAD; 'A Fortran Programming Course'; Prentice-Hall Inc., 2nd Edition.

18. McCRACKEN, D D; 'Fortran with Engineering Applications'; John Wiley and Sons Inc.

19. ALEKSANDER, I; 'Introduction to Logic Circuit Theory'; Engineering Series Monographs; G G Harrap & Co Ltd.

20. MORRIS, N M; 'Logic Circuits'; McGraw-Hill, London.

21. TOCHER, K D; 'The Art of Simulation'; English Universities Press Ltd, Electrical Engineering Series.

22. JANSSEN, B; 'Random Number Generators', Pethersons; 1966.

23. FLETCHER, R (Editor); 'Optimization'; Academic Press, London and New York.

24. WILDE and BEIGHTER; 'Foundations of Optimization'; Prentice Hall.

25. ZAHRADNIK, R L; 'Theory and Techniques of Optimization'; PECDS, Barns and Noble.

26. HUSKEY and KORN; 'Computer Handbook'; McGraw-Hill Book Company, Ltd, 1st Edition.

27. FIFER, S; 'Analogue Computation'; Vol III, McGraw-Hill Book Company, Ltd.

28. KORN and KORN; 'Electronic Analog and Hybrid Computers'; McGraw-Hill Book Company Ltd.

29. ROGERS and CONNOLLY; 'Analogue Computation in Engineering Design'; McGraw-Hill Book Company Ltd.

30. PYNE, I B; 'Linear Programming on an Electronic Analogue Computer'; AIEE Transactions; Communications and Electronics.

31. JENNES, R R; 'Analogue Computation and Simulation Laboratory Approach'; Allyn and Bacon.

32. TAHA, H A; 'An Introduction to Operations Research'; MacMillan.

33. GASS, S I; 'Linear Programming'; 3rd Edition, McGraw-Hill Book Company Ltd.

34. KIM, C; 'Introduction to Linear Programming'; Holt, Rinehart and Winston Inc.

35. /

35. HADLEY, G; 'Nonlinear and Dynamic Programming'; Addison Wesley Publishing Co.

36. BELL, D and GRIFFIN, A W J (Editors); 'Modern Control Theory and Computing'; McGraw-Hill.

37. PRABHU, N U; 'Queues and Inventories'; John Wiley and Sons, Inc.

38. NEWELL, G F; 'Applications of Queuing Theory Monographs on Applied Probability and Statistics'; Chapman and Hall, Ltd.

39. BEKEY and KARPLUS; 'Hybrid Computations'- John Wiley and Sons Inc.

40. HOSSACK, R A G; 'Simulation of a Stochastic Computer'; Honours Thesis; RGIT, May, 1974.

41. GRAY, J R; 'Probability'; University Mathematical Texts; Oliver and Boyd Ltd, pp 176-265.

42. ARTHURS, A M; 'Probability Theory'; Library of Mathematics, Routledge and Kegan Paul Ltd.

43. DRAKE, A W; 'Fundamentals of Applied Probability Theory'; McGraw-Hill Book Company Ltd., pp 163-185.

44. PFEIFFER, P E and SCHUM, D A; 'Introduction to Applied Probability'; Academic Press Inc, pp 349-370.

45. WHITTLE, P; 'Probability'; Library of University Mathematics; Penguin Books, pp 128-147.

46. BREIMAN, L; 'Probability and Stochastic Processes: With a View toward Applications'; Houghton Mifflin Company, pp 152-194.