Theory and design of a hardware stochastic reliability simulator.

MANN, D.P.

1982

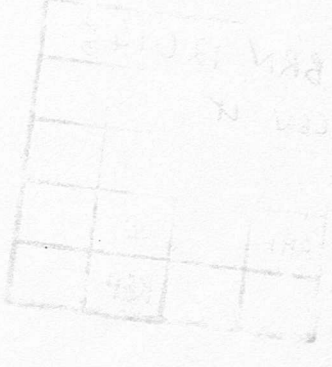
THEORY AND DESIGN OF A HARDWARE STOCHASTIC RELIABILITY SIMULATOR

By

Daniel Peter Mann BSc with
First Class Honours in Electronic Engineering.

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

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



August 1982

DECLARATION

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

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

(i) SERC Vacation School on 'Stochastic Processes in Control Systems' held at Warwick University, July 1981.

(ii) SERC Vacation School on 'Robot Technology' held at Hull University, September 1981.

(iii) SERC Graduate School on 'Industrial Management' at Durham University, April 1982.

(iv) 7th Advances in Reliability Symposium held at University of Bradford, April 1982.

(v) Fifth European Conference on Electronics held at Copenhagen, June 1982.

D P Mann
August 1982

Contents

ACKNOWLEDGMENTS

I would like to express my thanks to my supervisor Dr N D Deans for his guidance throughout the project.

My thanks also to Mr A J Bourne of The National Centre for Systems Reliability for his helpful discussions.

Finally I acknowledge the support of an SERC grant and thank Mr B Davidson for his help in the preparation of this thesis.

ABSTRACT

THEORY AND DESIGN OF A HARDWARE STOCHASTIC SIMULATOR

by

DANIEL P MANN

The work described in this thesis is concerned with the design of special purpose digital circuitry to achieve high speed simulation of system reliability.

Initially an investigation into alternative simulation methods is undertaken. Analytical and Monte-Carlo techniques are described and the advantages and disadvantages of each method are discussed. The operating principle of the new Simulator developed is placed in context with current simulator design.

Systems which contain reliable components, arranged in redundant configurations, possess high reliability. A reliability assessment for such systems requires fast simulation. Considerable parallel hardware operation is employed to produce simulation speeds faster than 100years/second, independent of the number of system components. A novel technique is also used to gain further speed via asynchronous time scaling.

Modelling of complex strategies of repair and preventative maintenance are catered for. Systems can be modelled where facilities for repair are limited and system components compete, according to some priority policy, for available repair men. The use of these additional modelling features is achieved without reducing simulation speed.

Components which make up the system under investigation are modelled by general purpose hardware modules. Each component module is equipped for considerable modelling detail, providing a simulator that can model systems with wide ranging characteristics. The operation of component modules is determined by a micro-programmable control unit.

The interconnection of components to form a system is dealt with by logical network techniques. Although logical networks are employed, observation of system behaviour is not solely by monitoring network events.

Important to the operation of the Simulator is the generation of random event signals of prescribed distribution. Considerable care has been taken in the design of the event signal generators, and their operation is both mathematically and statistically investigated to verify their performance.

Finaly the simulator is applied to a range of reliability simulation problems. The systems initially considered are analytically analysed to confirm the Simulator's operation. Later, systems containing complex repair schemes are modelled, and the economic or reliability performance of the system is optimised.

Review of Reliability Modeling

1.0 Introduction

Many systems which use a large number of components of equipment require high reliability from the overall system. During the development of these complex systems, decisions have to be made on possible maintenance policies and the organisation of components which make up the system. The inclusion of redundancy and complex strategies of repair or replacement, make the task of ensuring a low probability of system failure difficult.

Clearly there is a requirement for mechanisms that aid the designer and system operator with these problems. Many such mechanisms are built around models which do not adequately represent the system under investigation, or result in expensive time-consuming studies. This Chapter reviews and contrasts known techniques and finally introduces a new approach to the problem.

1.1 Investigative Techniques

The two basic approaches to calculation of system reliability use:

a) simulation

b) analytical techniques.

Whichever technique is employed, a mathematical description of the system must be developed. From this description, the two techniques follow separate paths to the solution of the system reliability problem. Analytical methods involve the solution of equations describing the system. This leads to an exact solution. Simulation methods involve experiments on the mathematical model. The results obtained are statistically analysed to determine system reliability.

At first sight, the analytical technique seems most attractive as it yields an exact solution. Also it might be expected to do so in a shorter time, as to increase the accuracy of values produced from statistical sampling, many experimental runs are required to gain a large number of samples. However an analytical solution may be intractable unless considerable simplification of the mathematical model is carried out, leading to results of questionable accuracy. The distribution form of results produced by simulation may also contain information not normally obtainable by analytical methods.

The following sections of this Chapter describe and compare the two methods. It will be seen that different paths to a solution can have similarities as indicated in Figure 1.1a.

1.1.1 Analytical Markov Approach

A system under investigation can take on a number of distinct system states. These states are connected by links corresponding to state transition probabilities. A state transition diagram aids in the development of such system models. Consider an example system of two active items of equipment for which one must work if the system is to be operational. Figure 1.1.1a gives the state transition diagram for the system. The items of equipment are identical, having a failure rate λ. Only one repair man is available, constituting a single item repair rate of μ. If single-step state transition probabilities are defined as $P_{i,j}$, which is the probability of moving from state i to state j, then transition probabilities can be arranged in matrix form. This matrix is known as the transition matrix P. The state of the system at time n is given by a state probability vector $(P_1,P_2,\ldots)^n = p^n$, where p_i^n is the probability of being in state i at the time n. To determine the probable state of the system at time m, given an initial starting vector p^o, the following equation must be solved

$$p^m = p^0 . \; p^m$$

The Markov approach to system reliability calculation proceeds by developing a state transition matrix. Techniques for the manipulation of transition matrices have been developed to calculate system characteristics such as mean time in a particular state or time-to-first state-encounter [1]. In reliability terms, these parameters correspond to mean down time and time to first system failure respectively. Computer programs have been constructed to generate state transition matrices and determine the solution [2,3].

A Markov process can be termed 'memoryless' as the state of the system at time m can be completely determined from the state at time m-1. No knowledge of the sequence of states leading to the state at m-1 is required. Another way of expressing this is that state transition probabilities are not time dependant. Limiting transition probabilities in this way allows times between repair and failure to be modelled only by exponential distributions. Here lies a disadvantage of Markov analysis. Either modelling distributions are limited to an exponential form, which may not be justified, or a non-Markov approach must be devised. Non-Markov processes are particularly difficult to analyse and are often avoided by applying methods to convert them to an approximate Markov process solution [2].

Tractable solutions to models employing non-exponential distributions have been found possible by semi-Markov methods [4]. Semi-Markov processes contain an embedded Markov process in that state transitions occur according to a transition matrix P. Delay times for transitions, say i to j, are given by $X_{i,j}$ and are determined from the conditional probability distribution $F_{i,j}(t)$ which may be non-exponential.

A further disadvantage of Markov methods is the intractability of systems containing large numbers of components, whose reliability is effected by varied maintenance and management policies. Such systems require techniques to be applied to reduce the large number of possible system states [5]. The problems outlined here have led to other methods of obtaining an analytical solution, principally the use of logical networks. Arguments for and against the use of logical networks are interesting [6,7] and are outlined in the section dealing with network techniques in general.

1.1.2 Simulation Approach

The technique of system simulation is a procedure based on experimentation carried out on a representative model system. It is an imprecise technique due to the stochastic nature of the model. That is, simulation does not provide exact solutions to system reliability problems. Better results are obtained by increasing the number of experimental observations. The technique relies upon the generation of random variables of known distributions to describe system properties. Bringing together these properties according to a mathematical model permits observations to be made about system properties of unknown distribution. The generation of random system parameters in this way is often referred to as Monte-Carlo Simulation.

The simulation approach to system reliability proceeds by developing a model which contains only the important features of the real system. This is not a simple task and requires a careful analysis of the system to be carried out if the simulator is to provide valuable information. Unnecessary model detail is to be avoided as it may obscure any understanding of model behaviour. Further detail results in longer development times, longer run times and greater cost. These pitfalls can be avoided if the initial system study is carefully investigated.

There are two principle methods for simulation time scaling. The choice of method employed has considerable bearing on the Simulator's implementation. The `event-by-event` method relies on updating the model at the occurrence of events [8]. Each event is decided on the basis of the shortest-time-to-occurrence, from a queue of events. According to the event chosen, the model is updated and a new random time generated for the next occurrence of the event, which is then returned to the event queue. The process of event selection continues until a prescribed condition (e.g. total simulated time) is reached. This type of time scaling is referred to as asynchronous, as the quantum of time that the simulation is incremented by does not directly relate to real time, but depends on the value selected from the queue. Simulators of this type appear to offer greater speed, as no time is wasted modelling system behaviour between events.

The second method of time scaling is known as `epoch-by-epoch`. Its principle is to divide real time into regular intervals (epochs). At the start of each new interval, all aspects of the model are brought up to date. This corresponds to synchronous simulation as the modelling process proceeds at a constant rate relative to real time. Unlike the previous method the generation of random time intervals between events is not required, rather the probability of event occurrence. However the time between events still follows some appropriate probability distribution. The accuracy of the distribution of events generated, in modelling a prescribed distribution, is dependent on the epoch size. This leads to a compromise situation not encountered by the event-by-event method, in that increased epoch size yields a faster simulation but reduced modelling accuracy.

The implementation of a simulator can be achieved using software or special purpose hardware. Software simulations involve the use of computers in conjunction with a program which models the system under consideration. Generally, special purpose simulation languages [9] are used to minimise modelling times and effort in software development. Such programs are often found to require substantial amounts of computer time for execution [10], unless considerable simplification of the model is permitted. With a view of improving simulation speed, programs can be constructed in machine language although in practice this is seldom done due to increased complexity. Software simulators written in special purpose languages or high-level compiler languages are invariably event-by-event because of the simulation speed problem.

An interesting technique applicable to event-by-event simulation and producing substantial speed gains is variance reduction [10]. The method allows a smaller number of statistical observations to be made by replacing the random time of duration for each random event occuring by the events expected duration. However the technique is limiting in the range of model characteristics observable.

Hardware simulators involve the construction of special purpose circuitry. This offers the opportunity of considerably increased simulation speed at a much reduced cost. Parallel modelling operations offer the main contribution to increased speed, a technique not normally available to the software simulator. By far the most popular method of implementation is by software. The reasons for this are as follows :

1. Convenience of computers and lack of experience in electronic circuit design.

2. The availability of special purpose simulation languages.

3. Adaptability of software programs. It is unusual to encounter a hardware simulator of reconfigurable nature.

4. The ability to construct a much more user-orientated simulator.

Most simulation studies are for unique systems, although much work has been directed to the more difficult problem of constructing a general purpose software simulator [11]. A general purpose Monte-Carlo hardware simulator is unknown.

1.2 Use of Logical Networks

Logical networks give a means of logically representing the interconnection of component aspects forming the system of interest. In reliability problems, the use of fault and success trees, which are subsets of logical networks, is frequently encountered. Tree analysis requires logical statements which describe the conditions necessary to bring about some undesired system event. Consider the logical flow diagram Figure 1.2a depicting a 5 component system. Flow diagrams or block diagrams are themselves logical networks which are useful at the system tree development stage, as they more closely resemble the functional system layout. The corresponding system success tree is given in Figure 1.2b. Components are restricted by a two state representation viz 'working' and 'failed'. In success tree notation, the occurrence of a component failure is known as a primary event. Primary events are connected by logical operators, generally AND and OR gates. These form logical sub events which lead towards system failure, the top event. It can be seen that a tree analysis provides a means of determining critical failure paths, as sub events contributing to the top event are logically related to the primary events, and as will be seen later can be quantified. Greater understanding of failure mechanisms offers an opportunity for system improvements.

In comparison with the Markov approach, the fault tree method has two disadvantages. It is unable to deal with degraded component operation, and analysis of system behaviour is restricted to statistics of the occurrence of the `working` or `failed` state.

Investigations into tree construction [12,13] are closely connected with realisation of the Boolean function describing the top event in terms of the primaries. The reason for this is due to the use of fault trees in determining analytical solutions to system reliability [14]. By this method, the top event is specified in terms of the minimum cut sets. Minimum cut sets are simply sets of primary events in which the presence of each component is necessary to bring about the top event. Replacing the events in the Boolean expression by their probability of occurrence permits the probability of the top event to be calculated from the minimum cut set equations. The method has found success and can tackle quite complicated systems, dealing with a range of probability distributions (unlike the Markov approach) and maintenance policies. Unfortunately, the cost of executing such software studies has been high and this has lead to the development of hardware modelling techniques [15,16]. Hardware techniques are generally employed in the calculation of system minimum cut sets, after which a computer is employed to determine system reliability.

Monte-Carlo simulations based on fault trees are also possible, and software implementations have been investigated [17]. Substantial numbers of runs would be required to simulate system operation if results are to be obtained with acceptable confidence limits. It is doubtful if software implementations can be achieved with reasonable running costs. However a Monte-Carlo simulation of logical trees does have a considerable advantage, in that the Boolean functions describing the top event and sub-events are not required. A hardware Monte-Carlo simulation employing logical trees is an area receiving little attention

[18]. High speed parallel hardware will make a valuable contribution to system reliability studies.

1.3 Proposed Simulator

A hardware Monte-Carlo simulator is proposed employing logical trees to define system configuration. The model developed is highly flexible, allowing varied system studies to be undertaken. System models are constructed from components. Each component employed by the model to simulate a particular system aspect may be programmed to characterise a range of behaviour. Components are equipped for considerable modelling detail, providing a simulator well suited to system studies containing small numbers of sophisticated components. The current design is for a simulator limited to 32 components.

Characteristic distributions selected during the programming operation are not limited to the exponential distributions. Maintenance and management policies are well catered for allowing manipulation of policies, the consequence of which can be studied in a controlled way. The use of logical networks enables system configurations to be easily altered. Further, they make possible the identification of sequences of events which have the potential to affect system operation.

Although logical networks are employed, statistical analysis of system behaviour is not only dealt with by monitoring the occurrences of logical tree events. The actual technique used permits information to be gathered about system behaviour which would be unobtainable via logical networks alone.

With a full knowledge of component reliability, it is possible to assess the probability of system failure in a very short time, as components operate in a parallel fashion. Further, the operation of each individual component is itself in a high speed parallel operation. The advantages of asynchronous simulation are also exploited to gain

additional simulation speed. When a numerical reliability goal is set for the system, the required redundancy of various components and subsystems can be found, and an optimum preventive maintenance policy produced.

Attention has been paid to operator usability and the proposed simulator is particularly convenient and efficient. A graphical specification of logical networks and display of experimental results has been incorporated.

1.4 Development of Simulator

The Simulator models the stochastic behaviour of system components by generating random binary signals at the occurrence of events, such as failure. Initially, interest is directed towards generating binary signals of uniform probability distribution. Later, methods of transforming the uniform distribution into any prescribed distribution are investigated.

Throughout the discussion of techniques to generate multiple streams of random event signals, statistical tests are employed to assess the various methods considered.

The construction of special purpose hardware to model, in parallel, the random and deterministic events which effect component behaviour are undertaken. The high speed hardware is interfaced to a computer to enable easy operator control.

After the construction of the simulator, systems of analyticaly determinable behaviour are modelled and the operation of the Simulator verified.

1.5 Conclusion

Many techniques have been used to achieve high simulation speeds. Modelling detail of systems under investigation is not compromised for simulation speed or cost. The simulator is particularly suited to sensitivity analysis and reliability improvement problems.

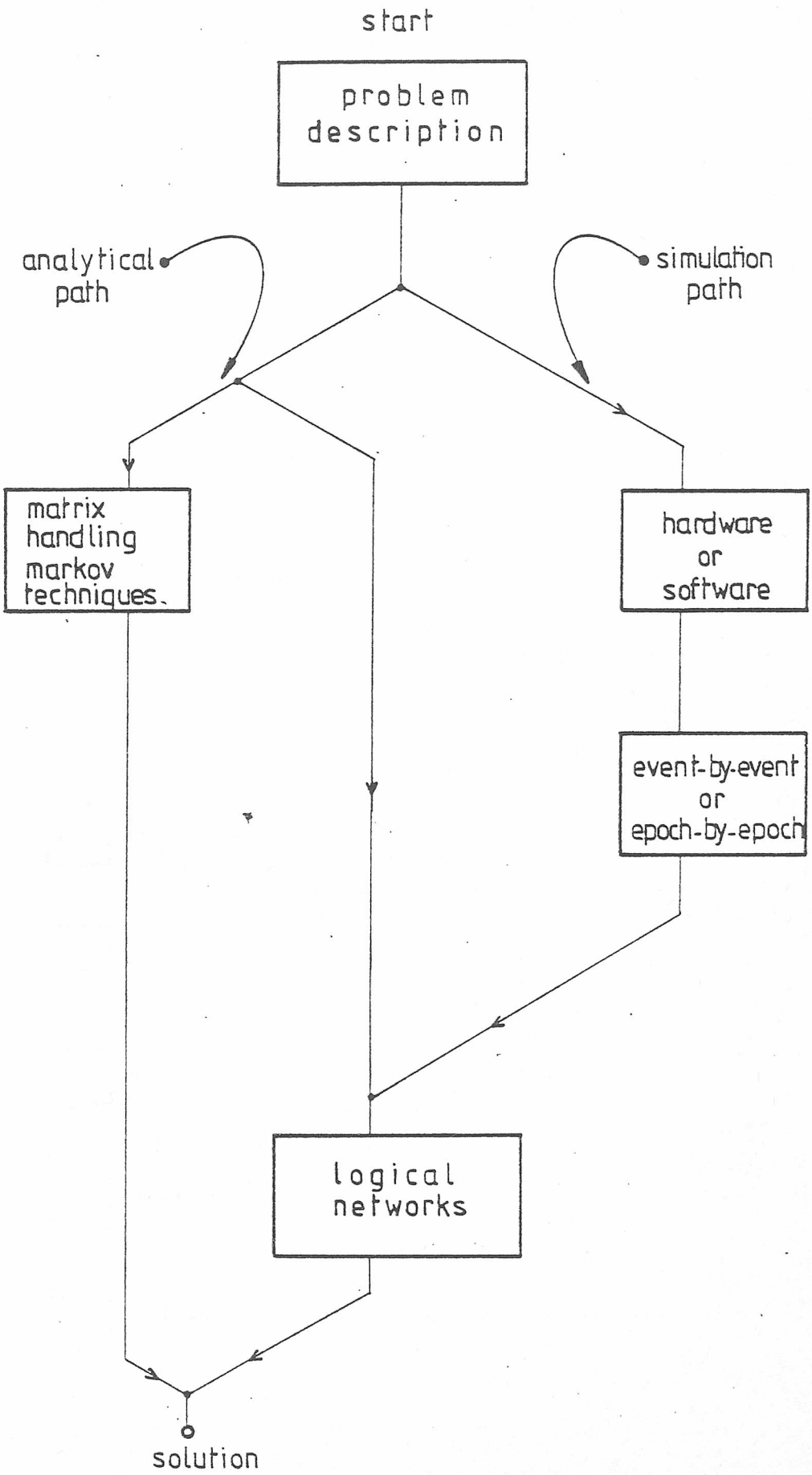


Figure 1.1a Alternative paths to a solution

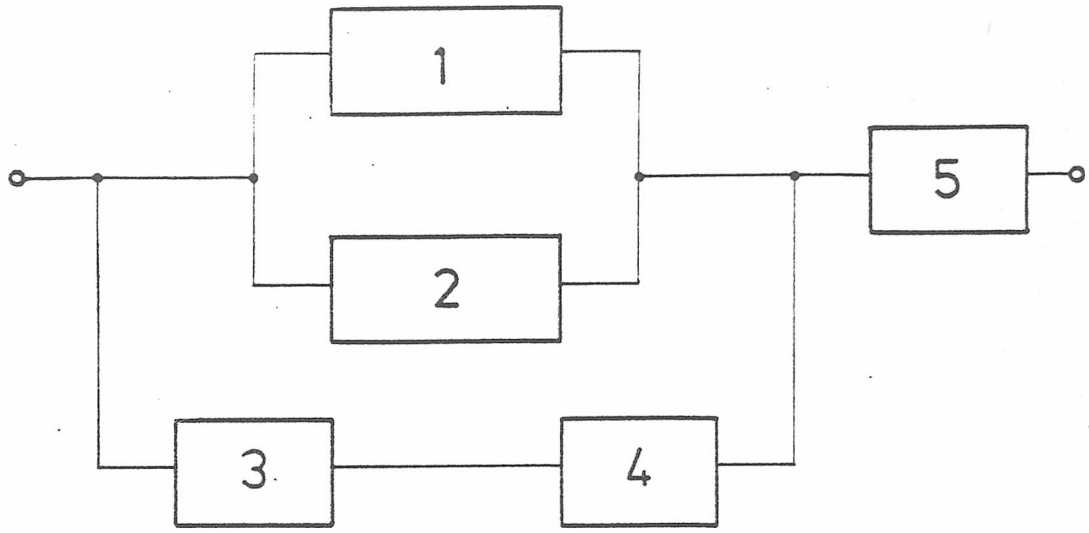


Figure 1.2a 5 component system

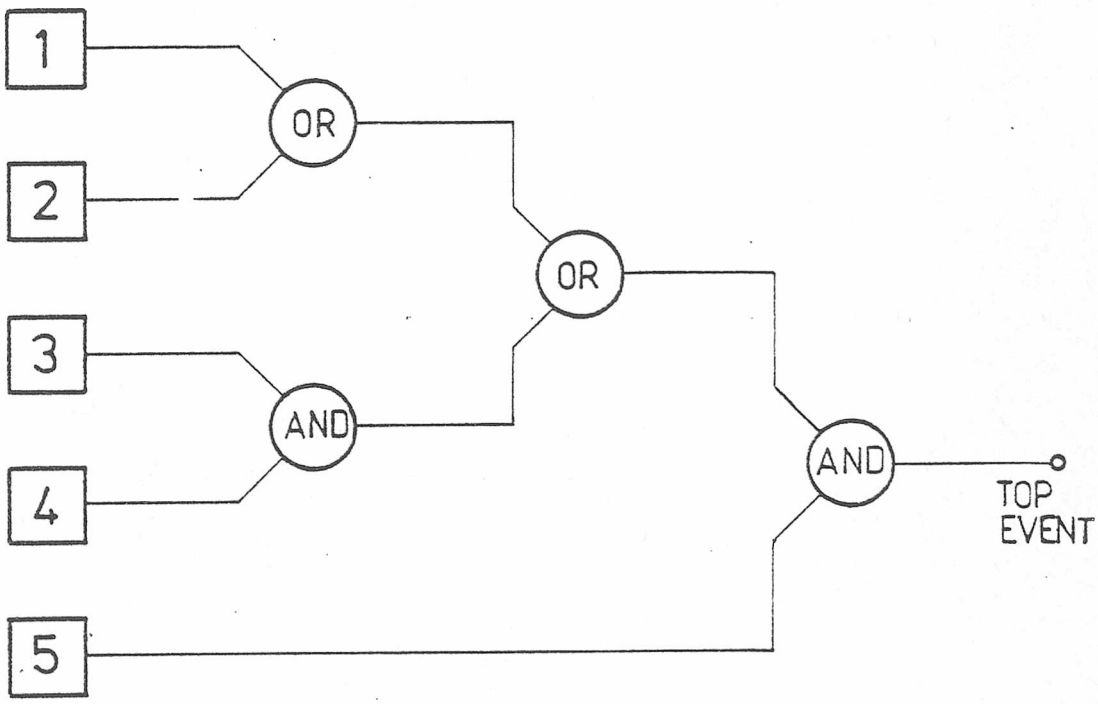
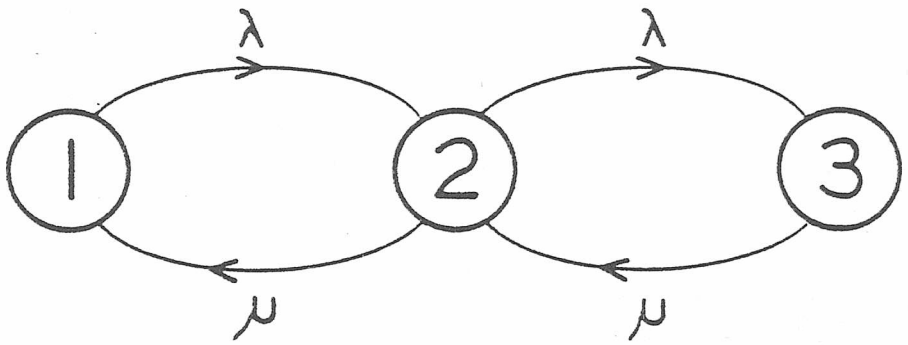


Figure 1.2b Success tree



state 1 = all working
state 2 = 1 working
state 3 = failed state

Figure 1.1.1a State transition diagram

2.0 Techniques of Random Number Generation

There are several ways of approaching the problem of generating random numbers. One technique involves the use of a device whose intrinsic operation is based on random phenomena. Some sort of signal conditioning is normally necessary to convert the device output into number form. Another method is to consult a table of numbers specifically produced to pass all the statistical tests for randomness when taken in a sequence. By far the most popular method of random number generation involves programming a digital computer to deterministically compute random numbers according to some simple algorithm. The resulting sequence of numbers is called pseudo-random since although the numbers might have excellent statistical properties, their origins are distinctly non-random.

The random number generator used in the simulator works on the principle that an n-bit random number can be constructed from n pseudo-random binary digits. The binary sequences are generated deterministically using shift registers with feedback. Such a generator is suitable for simultaneously producing multiple streams of random n-bit numbers. An analysis of the generator operation indicates that number sequences can be expected to have statistical independence. This Chapter describes the design of the generator and compares the method developed with other techniques for number generation. Statistical tests as well as mathematical analyses techniques are used to confirm the advantages offered by the design.

2.1 Theory of Pseudo-Random Binary Sequences

The principle of using shift registers to generate [19,20] sequences of 1's and 0's has been well explored. Figure 2.1a shows a typical digital circuit technique. The combinational feedback logic applies modulo 2 addition to certain bits of the register. The result is entered into the first bit of the register on the next clock pulse, during which the contents of the register are shifted one bit along.

Consider an N-bit register. The succession of states in the register is periodic, with period $P \leq 2^N - 1$. This is easily proven by considering that each state of the register is completely determined by the previous state. So if it ever happens that a state is the same as an earlier state then the sequence of states of the register would repeat. With an N-bit shift register there are only 2^N different states. Thus the sequence of states in the register must start to repeat somewhere between the first and 2^N clock pulse. Further, the all 0's state cannot be allowed to occur as this would result in only logic 0 being fed back, and the register would remain in the all 0's state. Thus $P \leq 2^N - 1$.

The sequence of 1's and 0's generated by the register is certainly of finite length and cannot be said to be truly random. The best that can be done is to single out certain properties as being associated with randomness, and to accept any sequence which has these properties as a random sequence.

The following properties are associated with randomness:

1. The number of logic 1 levels is approximately equal to the number of logic 0 levels.

2. Runs of consecutive logic 1 levels or logic 0 levels frequently occur, with short runs being more frequent than long runs. Half of the runs should be of length 1, a fourth of length 2, and an

eighth of length 3, and so on.

3. The auto-correlation is a measure of the amount of similarity between the sequence and the sequence phase shifted. The random sequence should have an auto-correlation function peaked at zero phase shift and near-zero for all other phase shifts.

These characteristics of randomness can be satisfied by a logic sequence produced by a shift register with appropriate feedback.

2.2 Feedback Circuitry

The particular logic required to generate the random '1's and '0's pattern is given by the factorisation of the equation governing the N-bit cycling shift register. If the contents of the i-th stage of the register is labelled Y_i (see Figure 2.2a) then it is easily seen that:

$$D^N Y_i = Y_i$$

where D is the well known delay operator, used to express the delay of one clock period. The above equation can also be expressed as:

$$(x^N - 1)Y_i = 0 \qquad (1)$$

where the symbol D has been replaced by x. The sequence produced by the register $\{Y_i\}$ satisfies equation (1). The same sequence may also satisfy the equation $(x^d - 1)Y_i = 0$ where $d<N$.

For this to happen $x^d - 1$ would have to be a factor of $x^N - 1$, and the roots of $x^d - 1$ would therefore also be roots of $x^N - 1$. Each factor of $x^N - 1$ is of degree less than N, and therefore the sequence produced could be generated on a shorter register. Not all the factors will describe sequences with the appropriate randomness properties. The suitability of each factor depends on its particular roots.

In the field of complex numbers the roots, a_k, of $x^N-1=0$ are $a_k=e^{j2\pi(\frac{k}{N})}$ (K=1,2, ... , N). The complex ones amongst these occur in conjugate pairs. The roots for which K is prime to N are called primitive N-th roots of unity. The number of primitive roots is given by the Euler function, $\phi(N)$ [20,pg23].

Consider N=15. The solution of the equation $x^{15}-1=0$ will give the 15 roots of unity ($a_k=e^{j2\pi(\frac{k}{15})}$,K=1,2,...,15), for which $a_k^{15}-1=0$, K=1,2,...,15.

Taking the case $a_{10}=e^{j2\pi(\frac{10}{15})}$ which can be expressed as $a_{10}=e^{j2\pi(\frac{2}{3})}$. Thus a_{10} is also a root of $x^3-1=0$. The divisors of 15 are 1,3,5 and 15 so $x^{15}-1$ has binominal factors x^d-1, where d may be any of these divisors. All the 15 roots of unity can be arranged into groups which satisfy the equation $x^d-1=0$ but not $x^{d'}-1=0$, where $d'<d$. Each root will satisfy an equation of the form $x^{15}-1=(x-a_k)$. Rearranging the roots into groups gives:

$$x^{15}-1 = P(1).P(3).P(5).P(15)$$

The root $a_{15}=e^{j2\pi(\frac{15}{15})}=1$ satisfies x-1=0. The conjugate pair a_5, a_{10} satisfies $x^3-1=0$, and so on.

$$P(1) =(x-a_{15}) = x-1$$

$$P(3) =(x-a_5)(x-a_{10})$$

$$P(5) =(x-a_3)(x-a_6)(x-a_9)(x-a_{12})$$

$$P(15) =(x-a_1)(x-a_2)(x-a_4)(x-a_7)(x-a_8)$$
$$X\ (x-a_{11})\ (x-a_{13})\ (x-a_{14})$$

The complex terms in the above equations occur in conjugate pairs and so can be expressed as polynominals with only real terms.

$$P(1) = x-1$$

$$P(3) = x^2+x+1$$

$$P(5) = x^4+x^3+x^2+x+1$$

$$P(15) = x^8 - x^7 - x^5 + x^4 - x^3 - x + 1$$

These polynominals are called cyclotomic polynominals [19] and the polynominal P(15) is termed primitive because it is a factor of $x^{15}-1$ and of no other x^d-1 where $d<15$. The roots of P(N) are the primitive N-th roots of unity. The number of primitive roots, and hence the degree of the polynominal is given by the Euler function, $\phi(N)$.

When x is restricted to binary values the above mathematics is applied Modulo 2. The polynominals describe binary sequences which will satisfy the equation $x^{15}+1=0$:

$$x^{15}+1 = (x+1)(x^2+x+1)(x^4+x^3+x^2+x+1)$$
$$X\ (x^8+x^7+x^5+x^4+x^3+x+1)$$

Note the polynominal $(x^8 + x^7 + x^5 + x^4 + x^3 + x + 1)$ can be factorised into $(x^4 + x^3 + 1)(x^4 + x + 1)$. Consider a sequence $\{Y_i\}$ which satisfies the polynominal P(5):

$$(x^4 + x^3 + x^2 + x + 1)\ Y_i = 0 \qquad (2)$$

The same sequence would satisfy the equation:

$$(x^{15} + 1)Y_i = 0$$

Remembering that x was defined as the delay operator, equation (2) can be set up on a 4-bit register (Figure 2.2b) using exclusive-OR gates to perform the modulo 2 addition.

The roots of P(5) are also the roots of $x^5-1=0$, and therefore the sequence $\{Y_i\}$ would also satisfy the equation.

$$(x^5 + 1)Y_i = 0$$

This proves that the sequence is only 5 bits in length. The same argument can be applied to the polynominal P(3). Roots of this polynominal are also roots of $x^3-1=0$ and a sequence, $\{Y_i\}$, which

satisfies:

$$(x^2 + x + 1)Y_i = 0$$

will also satisfy $(x^3+1)Y_i=0$ and therefore can only be 3 bits in length.

It can be seen that the equation which is satisfied by the longest sequence is the primitive polynominal. It has roots which are not solutions to any of the binominal factors of $x^{15}-1$, but only to the primitive polynominal and $x^{15}-1$. Consider the expression x^4+x^3+1, a factor of the primitive polynominal. A four bit register designed to generate the sequence $\{Y_i\}$ governed by:

$$(x^4 + x^3 + 1)\ Y_i = 0$$

is shown in Figure 2.2c. The sequence will be 15 bits long which is the maximum possible length for a 4-bit register. The equation $x^4+x^3+1=0$ is called the characteristic equation and the sequence is known as a maximum length sequence or m-sequence for short.

In general for an N-bit shift register to produce a maximum length sequence the period of the sequence must be $p=2^N-1$. This would require that the polynominal describing the generated sequence is irreducible (that is, cannot be factorised). M-sequences pass all the statistical tests for randomness previously listed [Section 2.1].

Consider an m-sequence of length $p=2^N-1$, produced from an N-bit register; the number of 1's in the sequence is 2^{N-1} and the number of 0's is $2^{N-1}-1$. The difference is due to the register never entering the all 0's state. The auto-correlation is given by C(t), where:

$$C(t) = 1 \qquad \text{if } t = Kp$$

$$C(t) = -\frac{1}{P} \qquad \text{if } Kp<t<(K+1)\ p$$

$$\text{for } K = 0,1,2,\ldots$$

The auto-correlation function of m-sequences is two valued: $C(t)=1$ for a delay which is a multiple of the period, and $C(t)=-1/p$ for all other values, since the number of 1's in each period exceeds the number of 0's by one.

2.3 Coset Structure

In Section 2.2, dealing with feedback circuitry of shift registers, the initial equation (x^N-1) was assumed. This equation has N roots. Some of the roots will also be roots of equation (x^d-1) where $d<N$. The rest will only be roots of (x^N-1) or some higher order expression.

Consider, as in Section 2.2, the case $(x^{15}-1)$. The 15 roots are given by $a_k=e^{j2\pi(\frac{k}{15})}$, $k=1,2,\ldots15$. It was shown that roots a_5, a_{10} were also roots of (x^3-1) and roots a_3, a_6, a_9, a_{12} were also roots of (x^5-1) and so on. The roots can be arranged into cyclotomic polynominals. For example, all the roots of (x^5-1) can be arranged into polynominal P(5):

$$P(5) =(x - a_3)(x - a_6)(x - a_9)(x - a_{12})$$

The integers from 1 to N-1 can also be grouped into cyclotomic cosets [19]. A coset contains the integers which 'correspond' to the roots of a particular cyclotomic polynominal. For example, the elements in the coset which correspond to P(5) above are

3, 6, 9, 12

If N is odd the integers $1,2,4,8,\ldots2^{m-1}$ (where $N=2^m-1$) are the elements of the first coset, known as the multiplier subgroup. The other cosets are obtained by multiplying, modulo N, any of the remaining integers from 1 to N-1 by each member of the subgroup. For the case N=15 the cosets are

C1	1	2	4	8	proper
C2	7	14	13	11	proper
C3	3	6	12	9	improper
C4	5	10			improper

Cosets which contain numbers prime to N (numbers which correspond to roots of the primitive polynominal or a factor of it) are termed proper cosets, the others are termed improper. Each proper coset corresponds to a particular characteristic equation with its own m-sequence [19]. The number of possible m-sequences is given by the number of proper cosets.

2.4 Decimation of N-bit Number

Decimation of an m-sequence $\{a_n\}$ is the process of generating a new sequence $\{b_n\}$ by selecting every q-th term of $\{a_n\}$.

If $\{a_n\}$ is an m-sequence of period r, and q is prime to r (proper decimation), $\{b_n\}$ also has period r and is either a phase shifted version of $\{a_n\}$ or a different m-sequence altogether.

Decimation can be understood by examining the coset structure. As stated above, if q is prime to r, then $\{b_n\}$ is an m-sequence. This limitation on q corresponds to q, modulo r, being a member of a proper coset. The sequence $\{b_n\}$ produced depends on the coset to which q belongs and not the exact value of q. If q is a member of the multiplier subgroup then $\{b_n\}$ is identical to $\{a_n\}$ except for a possible phase shift. However if q belongs to another proper coset, a different m-sequence is produced. Should q be a member of an improper coset, then $\{b_n\}$ is not a maximal length sequence.

The application of proper decimation is important when over-coming the problem of producing several random binary sequences.

2.5 N-bit Random Number

The problem of producing pseudo-random binary sequences has been dealt with. Consideration must next be given to how these random binary digits can be used to produce an n-bit random number.

Let the n-bit random number be denoted by E_i. The random variable should be uniformly distributed in the interval $[0,2^n-1]$, and each number in this range should be equally likely to appear at any point in the sequence. There should be no correlation between E_i and E_{i+k} for any non-zero integer k. That is each number in the sequence should appear to be independent of all previous numbers in the sequence.

This criteria cannot be met with sequences of pseudo-random numbers, because each number is strictly determined by the preceding number in the sequence. However the generator can be considered acceptable if the numbers produced pass certain statistical tests, based upon the properties of sequences of truly random numbers. A description of the tests used and the results obtained shall be given later in this chapter.

As for the auto-correlation function of E_i, it is known [21] that it is the existence of correlation variation rather than its absolute value that determines the quality of the random numbers.

The ideal generator should also be able to serve as a source of several random numbers. That is, an n-bit random number generator may have a demand put upon it to supply say p 'customers' with n-bit random numbers of the type described above. Further, the numbers received by any particular customer should have statistical independence from the numbers received by any other. (the first customer receives number sequence E_i^1, the second customer receives number sequence E_i^2, the Jth

customer receives the number sequence E_i^J where $1 \leq J \leq P$. The use of m-sequences to meet these requirements must be treated with caution.

The following subsections describe a number of possible n-bit number generators, each employing m-sequence techniques. A discussion of the 'quality' of the random numbers that would be produced is also given.

Work carried out in [22] has proved useful in understanding the problem of random number generation.

2.5.1 Multiple m-Sequences

This generator is constructed from n separate feedback shift registers. Figure 2.5.1a shows the register arrangement. All n registers are clocked together, therefore producing a new n-bit number at each clock pulse.

Each register is N bits long and produces an m-sequence of length 2^N-1. All m-sequences are different. That is, the operation of each register is described by a particular primitive polynominal which applies to only that register. The binary sequences produced at each of the n outputs share the same period, but the actual pattern of 1's and 0's is different in each case.

The auto-correlation functions of the individual sequences are two valued, and the individual sequences will pass the statistical tests for random binary sequences. Cross-correlation between the different m-sequences produced [19,pg82, 20,pg65] is not zero, but assumes a number of distinct values which is less than or equal to the number of cyclotomic cosets. This characteristic disqualifies the generator as a high quality pseudo-random number source.

The generator does have the advantage that the registers can be started from any initial condition. Also it can easily act as a source of several random numbers. Proper decimation of the output random number corresponds to a proper decimation of each m-sequence. Therefore the bit contributed to the random numbers from any particular register has a relative phase shift round the same m-sequence, as explained in Section 2.4. This results in the cross-correlation between different bits of different random number sequences having the same form as the cross-correlation function between different bits of the same random number sequence; or in the case where the bit is supplied by the same register, the auto-correlation function. It can be concluded from this that the independence between numbers in different sequences should have the same statistical quality as the independence between consecutive numbers in the same sequence. This type of generator has been tested [23] and the results of the test carried out on the numbers produced were considered adequate.

2.5.2 Two-feedback shift register pseudo-random sequences

This generator is very economical in hardware (see Figure 2.5.2a). Only two feedback shift registers, each producing an m-sequence, and a number of exclusive-or gates are required. The generator has been used in various applications as a pseudo-random number source. The Manchester University Institute of Science and Technology road-traffic simulator is one example of this kind of machine. It has been investigated in [24,25,21,26]. The auto-correlation function of the binary sequences, produced at the output of the exclusive-OR gates, is certainly not two valued and is of the form :

$$R(t) = \frac{1}{T_a T_b} \qquad t \neq K_1 T_a \neq K_2 T_b \qquad K_1, K_2 = 1,2,3,\ldots$$

$$R(t) = \frac{-1}{T_b} \qquad t = K_1 T_a \neq K_2 T_b \qquad K_1, K_2 = 1,2,3,\ldots$$

$$R(t) = \frac{-1}{T_a} \qquad t = K_2 T_b \neq K_1 T_a \qquad K_1, K_2 = 1,2,3,\ldots$$

$$R(t) = 1 \qquad t = K T_a T_b \qquad K = 0,1,2,\ldots$$

for $0 < t < T_a T_b$

Where T_a, T_b are the periods of the m-sequence produced by registers a and b respectively.

A plot of the function, R(t), is given in Figure 2.5.2b. A random binary sequence is produced at each exclusive-OR gate, resulting in a new n-bit number at every clock pulse. Each sequence is a delayed version of only one sequence, which has period $T_a T_b$. The exact amount of delay is determined by the particular pair of stages which are selected for modulo 2 addition.

It has been shown [21] that the auto-correlation of an n-bit random number produced by using n of these sequences is many valued. This means that this generator cannot be considered as a first class random number generator.

Decimation of the n-bit output number to produce several sequencees of n-bit numbers, can be considered as decimation of several phase shifted, but identical, pseudo-random sequences. A proper decimation of the type of sequence produced by this generator has the same result as decimation of an m-sequence. That is, the new bit sequences produced are identical to the original, and equally displaced from one another, but contain an additional phase shift from the original.

Thus the individual number sequences produced by decimation have the same statistical randomness quality as the number sequences produced from an undecimated generator. Cross-correlation between the number sequences is indeterminable as the actual phase shift, relative to the original, introduced to the bit sequences by performing decimation, depends on the particular decimation and position in the sequence the decimation starts. This could lead to random bits of one n-bit number correlating with different bits of an n-bit number from another sequence, or the same sequence, after a very short time. It can be concluded that decimation of an n-bit number which is produced from several phase shifted versions of the same sequence is an unreliable method of producing number sequences.

2.5.3 Cascaded shift registers

This type of generator, shown in Figure 2.5.3a, is very similar to the generator described in Section 2.5.2. There are n side registers, each M-bits long, and one main register of N-bits. The feedback circuitry on each register is such as to, if acting alone, produce m-sequences. Each side register is also coupled to the main register.

The generator has been statistically investigated in [27]. The random binary sequences produced have exactly the same form as in Section 2.5.2. Clearly the n-bit random numbers produced will have the same characteristics, and therefore are rejected. The difference between this generator and the generator employing only two feedback registers is in the way the phase shifts are achieved.

A phase shift of 2^N-1 or a multiple of this (where 2^N-1 is length of the main sequence) can be achieved by simply ensuring that the coupling space is one or more between any of the side registers. By this method, a phase shift can be ensured between all of the output sequences regardless of the initial conditions of the registers.

2.5.4 Decomposed Register

This generator (Figure 2.5.4a) has been investigated in [28] and more extensively in [29]. It consists of n registers of length N_i (i=1,2,...,n) linearly interconnected so that the characteristic polynominal which describes the composite system is primitive and of degreee $N = N_1+N_2+....N_n$. At each clock pulse, n new bits are formed and used as inputs to the n registers. These n-bits can be used to form an n-bit random number.

Each of the n binary sequences is a phase shift of a single m-sequence. It was shown in [29] that considerable care must be taken at the design stage to ensure that the phase shifted sequences are 'widely' spaced. The minimum phase shift difference sets a limit to the operation of the number generator before cross-correlation between the binary sequences occurs. Since each output sequence is identical, the cross-correlation function has the same form as the auto-correlation function except there is an initial delay present.

The networks f_i and g_i perform modulo 2 operations on the contents of the sub-registers. These networks must be carefully chosen to ensure an m-sequence is produced and relative phase shifts are achieved. Random numbers produced by this generator will appear statistically independent when taken in sequence, but the computation involved in evaluating the phase shifts of lone m-sequences is excessive. From the conclusion arrived at in Section 2.5.2, it is clear the technique of simply decimating the n-bit numbers produced to obtain several output number sequences would be unreliable.

2.5.5 Use of shifted m-sequences

The techniques described in Sections 2.5.2, 2.5.3 and 2.5.4 involve n identical but phase shifted sequences being used to produce an n-bit random number. If the binary sequences used are m-sequences, the random number sequence can be expected to have good statistical properties. However the decimation of such a random number sequencee does not produce several sequences with adequate statistical randomness. It is suggested that the problem of synchronously generating multiple schemes of random numbers can be overcome by the phase-shifting of pre-conditioned sequences.

Any advanced or delayed pseudo-random binary sequence produced by a shift register with feedback can be obtained by modulo 2 addition of selected stages of the shift register as shown in Figure 2.5.5a [30, 31].

Let $U_{k,j}$ be the binary output of the j-th stage of an N-stage register at time k. Let $\overline{b}_j$ be the output vector, i.e. a '1' in the j-th row of N vector $\overline{b}$ implies that the output sequence $\{U_{k,j}\}$ is taken from the j-th register stage. Let $\overline{V}_k$ be the state vector defining the contents of the register at time k. Then:

$$U_{k,j} = \overline{b}_j^T \overline{V}_k$$

also

$$U_{k,j} = \overline{b}_j^T \overline{T}^{k-1} \overline{V}_1$$

Where $\overline{V}_1$ is the initial state of the register and $\overline{T}$ is the register transition matrix. A delayed version of the sequence $\{U_{k,j}\}$ namely $\{U_{k\pm s,j}\}$, where s is the number of bits the sequence is shifted, may be considered as either the sequence $\{U_{k,j}\}$ operated on by the transition matrix s times, i.e.

$$U_{k\pm s,j} = U_{k,j} \overline{T}^{\pm s}$$

$$= \bar{b}_j^T \bar{T}^{k \pm s-1} \bar{V}_1 \qquad (3)$$

or as the weighted modulo 2 sum of the register states at time k

$$U_{k \pm s,j} = \sum_{i=1}^{n} d_i U_{k,i} \qquad (4)$$

Where d_i are the weighting coefficients.

Substituting for $U_{k,j}$ in (4) and equating (3) and (4) gives:

$$\bar{b}_j^T \bar{T}^{k \pm s-1} \bar{V}_1 = \sum_{i=n}^{n} d_i [\bar{b}_i^T \bar{T}^{k-1} \bar{V}_1]$$

$$\bar{b}_j^T \bar{T}^{\pm s} = \sum_{i=1}^{n} d_i \bar{b}_i^T \qquad (5)$$

The feedback stages of the shift register define matrix $\bar{T}$. The output stages of the register, to be used for the reference sequence $\{U_{k,j}\}$ define vector $\bar{b}_j$. From equation (5) it can be seen that by multiplying $\bar{b}_j$ by $\bar{T}$, s times will determine the stages of the register to be added to produce the required shift.

A computer simulation of the proposed technique revealed that a phase shift of approximately 2^J where J is an integer can be obtained with the addition of a small number of stages. This is an important point as the modulo 2 adders introduce excessive propagation delays in a hardware random number generator.

Phase shifted sequences produced by the method described can be used to produce an n-bit random number. Consider the case where a 31-stage shift register has feedback applied so as to produce a $2^{31}-1$ bit m-sequence. If an 8-bit random number is required, the relative phase shift between each binary sequence should be $(2^{31}-1)/8$. Thus the maximum displacement is approximated by 2^{28}. The required phase shifts $(S_0, S_1, \ldots S_7)$ are given by

$$S_i = i.2^{28} \quad , \; i = 0,1,\ldots.7 \quad \text{modulo } 8$$

Sequences for which J is non-integer can be generated using the method shown in Figure 2.5.5b. A single 31 stage register with feedback is used to produce an m-sequence $\{U_k\}$, and the appropriate stages are modulo 2 added to obtain sequences $\{U_{k-S1}\}$,$\{U_{k-S2}\}$ and $\{U_{k-S4}\}$. Now if a sequence shifted by S_i is further shifted by S_j (j=0,1,...7) the resulting sequence is displaced from the original by S_i+S_j. The sequence $\{U_{k-S1}\}$ is fed into another register which has the same modulo 2 addition operations performed on its contents as the first register. The resulting output sequences are $\{U_{k-S3}\}$ and $\{U_{k-S5}\}$. All eight sequences can be produced in this way. Random binary sequences produced by this technique have the same properties as the sequences produced by decomposed registers methods but the necessary phase shifts are much more simply achieved.

The problem of synchronously generating several n-bit random numbers can be overcome by employing the described technique to pre-conditioned sequences. Each number sequence is produced by a generator of the type described above. The phase shift between any two binary output sequences is calculated from a section of the m-sequence length and not the whole m-sequence. The length of a section is determined by the number of random number sequences. Consider the case where four random numbers each of 8-bits are required simultaneously. The length of a section should be 1/4 the length of the m-sequence employed and the phase shift between output bits should be 1/8 of the section. There is now maximum displacement between any two bits of any number. Figure 2.5.5c illustrates the example. In practice the method can be simply implemented where there are 2^I (I integer) number sequences. Each shift register, operating on its allocated position of the m-sequence, is preloaded with the m-sequence value at the start of its section. This is achieved by decimation of a source m-sequence. The hardware solution to the example above is shown in Figure 2.5.5d. There

is an increase in the amount of hardware required but the random numbers produced can be expected to show good statistical independence. The generator is very fast producing all bits of each number at every new clock pulse. A generator of this type is proposed for use in the reliability simulator[32].

2.6 Tests on Random Number Generator

In Section 2.5 the problem of producing an n-bit random numbers was investigated. The subsection which followed described particular generators which have been proposed as solutions to the problem. There were only three underlying methods present in the proposed generators;

1. Multiple m-sequences (Section 2.5.1). The generator of Figure 2.5.1a was built to test this method. It produces a new 8-bit random number at each clock pulse. The random number sequence produced was decimated to produce a source of sixteen random numbers ($E_i^1, E_i^2, \ldots E_i^{16}$). The results obtained for the method are shown in Table 2.6a.

2. Combination of two m-sequences (Section 2.5.2 and 2.5.3). Figure 2.5.3a shows the generator built to test this method. The number sequence was not decimated as it may have led to results unrepresentative of the generator. An analysis of this type of generator showed how it was only suited to the generation of a single number sequence. For the purpose of statistical tests the generator was run sixteen times and each of the number sequences produced was considered separately. The results obtained from the method are shown on Table 2.6b.

3. Shifted m-sequence techniques (sections 2.5.4 and 2.5.5). Three generators were constructed to test this method. Two were of the

type shown by Figure 2.5.5b. Different m-sequences were used in each example; the first had feedback from stages 18 and 31, the second from stages 24 and 31. Although the generators were constructed to produce 8-bit numbers the phase shifts were determined for a 16-bit number, thus enabling expansion to 16 bits at a later date. The stages required to be modulo 2 added to produce the phase shifts are given on Table 2.6f. Both the generators were decimated to produce sixteen number sequences. An analysis of this type of generator concluded that decimation was an unreliable method of producing several number sequences, but it may prove interesting to observe the effects (if any) on the statistical test results. The actual test results are given on Tables 2.6c and 2.6d. The third generator in this section is of the type shown by Figure 2.5.5d. It was constructed to produce 16 sequences of 8-bit numbers. Actual phase shifts were determined for thirty-two numbers of 16 bits, allowing expansion at a later date; the stages required for modulo 2 addition are given on Table 2.6f number sequences produced by this generator should be of the same quality as number sequences produced by an undecimated version of generator 2.5.5b. This effectively enables the results which would have been obtained from undecimated versions of the above two generators to be determined. The test results are given on Table 2.6e.

A number of statistical tests were used to examine the performance of the random number generators, with the aim of determining the quality of the numbers produced. The results prove interesting as two of the methods were rejected on the grounds that they would produce random numbers of insufficient statistical independence.

All tests were carried out for an 8-bit random number, that is E_i lay in the interval [0,255]. Although it was expected that a random number of greater amplitude variation would be required, limiting the range to a smaller value allowed a smaller sample size of four thousand numbers per sequence to be gathered. The actual tests carried out are discussed separately in the following subsections.

2.6.1 Empirical Mean

The empirical mean, the central tendency or location of the random variable, was calculated for each of the sixteen sources.

$$\hat{E} = \frac{1}{N}\sum_{i=1}^{N} E_i$$

expected value $\frac{2^8-1}{2} = 127.5$

The results are shown in the first column of each of the tables of results.

2.6.2 Variance

The variancee of each sequence was calculated. A measure of the dispersion of the random variable:

$$\sigma^2 = \sum_{i=1}^{N} (E_i-\hat{E})p(x_i) \qquad ,N=2^8$$

expected value [20], $\sigma^2 = \frac{(N^2-1)}{12} = \frac{2^{16}-1}{12}$

$$\sigma^2 = 5461.25$$

The results are shown in the second column of each table.

2.6.3 Distribution

The random numbers should be uniformly distributed in the interval [0,255]. To check this the chi-square (χ^2) test [33,34] was used to examine the goodness of fit between the observed data distribution and the theoretical expected distribution

The interval [0,255] was divided up into mutually exclusive and equal groups. Each random number produced falls into one of these groups. By applying the χ^2 test it can be determined whether a significant difference exists between the observed number in each group and the expected number in each group. The interval was divided into 64 groups and a sample of N numbers were taken. The number (θ_i, i=1,2,...64) of observations falling in each of the 64 groups was determined. The reason for combining the interval into 64 cells was to ensure that $\theta_i > 5$ for every i.

The test is given by:

$$\chi^2 = \sum_{i=1}^{64} \frac{(\theta_i - E_i)^2}{E_i}$$

where E_i is the number of observations expected in the i-th group. In this case, $E_i = \frac{N}{64}$ a constant value, because the random numbers should be uniformly distributed.

The hypothesis that the random numbers are uniformly distributed can be checked by using a rule to reject or not reject the hypothesis on the basis of the result obtained from the χ^2 test. The probability α to reject the hypothesis when it is true is called the level of significance of the test. If chosen to be $\alpha = 0.05$, then the critical value for $\chi^2_{63,0.95}$ can be found from tables (χ^2_{63} is a chi squared distribution with 64-1=63 degrees of freedom, 64=number of groups in test). If from the test results $\chi^2 > \chi^2_{63,0.95}$ then the hypothesis will be rejected at a 5% level of significance.

The results and critical values are shown in the third column of each table of results.

2.6.4 First Independence Test

An independence test on number pairs was carried out. The number pairs were:

$$(E_i^1, E_{i-1}^1), (E_i^2, E_{i-1}^2), \ldots (E_i^{16}, E_{i-1}^{16}) \qquad i=2,3,4,\ldots N$$

N is the number of samples in each of the sixteen sequences (For the tests carried out N=4000).

Consider the first number source producing the sequence of random numbers E_1. The interval [0,255] was divided into sixteen groups to allow the χ^2 test to be performed. Each random number produced falls into one of the 16 groups. The quantity of consecutive numbers (E_i^1, E_{i-1}^1) falling in the same group was observed, and compared with the expected value. In this case the expected value in each group is N/(16x16).

The χ^2 test determines whether a significant difference exists between the observed number in each group and the expected number in each group.

Results for this test and critical values of χ^2_{15} are shown on the fourth column in the tables of results.

2.6.5 Second Independence Test

A second independence test on the number pairs:

$$(E_i^2, E_i^1), (E_i^3, E_i^1), \ldots (E_i^{16}, E_i^1)$$ was done

As in the first independence test, the interval [0,255] was divided up into 16 groups. The χ^2 test was carried out esults are shown on the

fifth column in the tables of results. Note that this test has only been carried out on decimated number sequences.

2.6.6 Distribution (Kolmogorov-Smirnov test)

The Kolmogorov-Smirnov [34] test has been used to compare the experimental cumulative distribution function observed with the expected distribution. The test is based on the difference between the cumulative sample distribution calculated for each interval in [0,255] and the observed cumulative distribution. The value D obtained from the test is the difference with greatest absolute magnitude.

Test results produced by each of the sixteen sources are given in the sixth column in the table of results. The hypothesis is that the observed distribution matches the expected if the D value falls below the critical value of D.

$$D_{critical} = \text{multiplier} \times \frac{\sqrt{2}}{\sqrt{N}}$$

N is the number of samples in each sequence. The multiplier can be found from tables and is determined by the level of significance of the test. The critical values of D are also shown in the sixth column in the tables of results.

2.6.7 Runs Test

This test [33,pg55] can determine if there are long runs of large or small numbers. It is considered to be a very discriminating test, that is it 'fails' more sequences of random numbers than other tests.

The runs test is designed to deal with binary data, that is a variable which has only two values. To allow the test to be implemented, the random numbers greater than or equal to 128 are replaced by a '1'. Random numbers less than or equal to 127 are replaced by a '0'. A run is defined as a succession of identical 1's or identical 0's. The actual number of runs occuring in any of the sixteen random sequences is given the value r.

Let N_1, be the number of 0's in the sequence and N_2 the number of 1's. The variable r will have a normal distribution, with:

$$\text{mean} = u_r = \frac{2N_1N_2}{N_1+N_2}$$

$$\text{and standard deviation} = \sigma_r = \sqrt{\frac{2N_1N_2(2N_1N_2-N_1-N_2)}{(N_1+N_2)^2\,(N_1+N_2-1)}}$$

The test to be carried out is to find the value of Z where:

$$z = \frac{r-u_r}{\sigma_r}$$

Under the assumption that the original random numbers are uniformly distributed in the interval [0,255] the value Z should be normally distributed with zero mean and unit variance. The significance of the value of Z obtained from the test can be found from tables of the normal distribution. Actual values of Z obtained for the sixteen random, 8-bit number, sequences are shown in the last column of the tables of results. Critical values for the test are also given.

2.7 Conclusion

Various techniques for producing several statistically independent streams of random numbers have been investigated.

The use of multiple m-sequences was considered. This technique can be easily implemented, and the n-bit numbers produced by a single generator whose output sequence is decimated have the same statistical quality as the original single number stream. However cross-correlation characteristics between different m-sequences rule out this generator as a highly independent number source. Decimation was also shown to be unreliable and this is reflected in the poor results of Table 2.6d.

The use of a two-m-sequence combination was considered for two quite different implementations. Modulo 2 addition of m-sequences offers a economy of hardware although care has to be taken to ensure that an adequate phase shift exists between output bit sequences. The use of cascaded shift registers eliminates the phase shift problem but at the cost of additional hardware. Investigations of the auto-correlation function of an n-bit number produced by this technique indicated that the numbers are not of high statistical quality. In addition, proper decimation of such a number sequence may result in significant correlation between the output number sequences.

Finally the principle of producing n output sequences from a single m-sequence was considered. This may be achieved by decomposed register techniques. However considerable effort is involved in ensuring that the n sequences produced are widely spaced. Modulo 2 addition of selected stages of a feedback shift register also yields phase shifted sequences, and the necessary combination of stages is more simply calculated.

A new pseudo-random number generator is proposed for use with the reliability simulator [32]. The technique relies on phase shifting pre-conditioned sequences, drawn from a single m-sequence. Each output number sequence is simultaneously generated, making the technique well suited to high speed simulation. Statistical tests performed on the generator confirm the high statistical quality of the numbers produced [32].

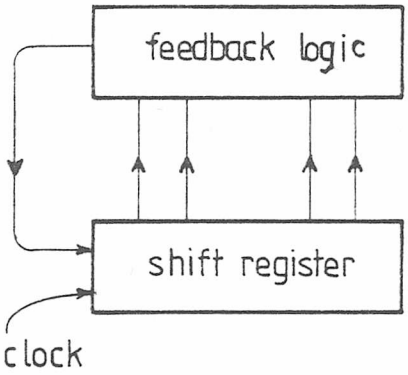


Figure 2.1a Shift register with feedback

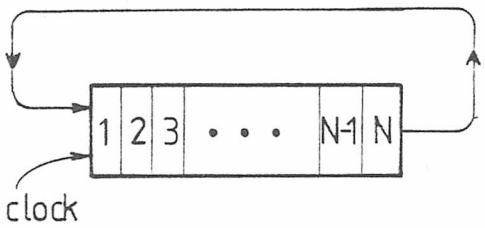


Figure 2.2 a N-bit cycling shift register

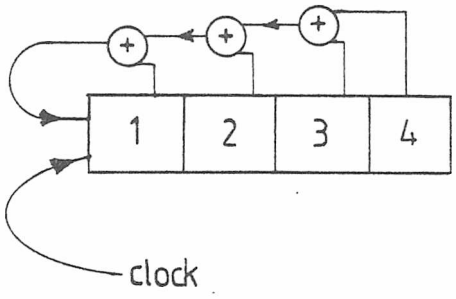


Figure 2.2b Exclusive-OR gates perform feedback operation

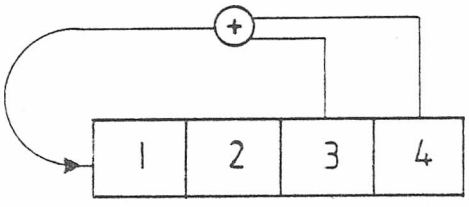


Figure 2.2c P R B S generator

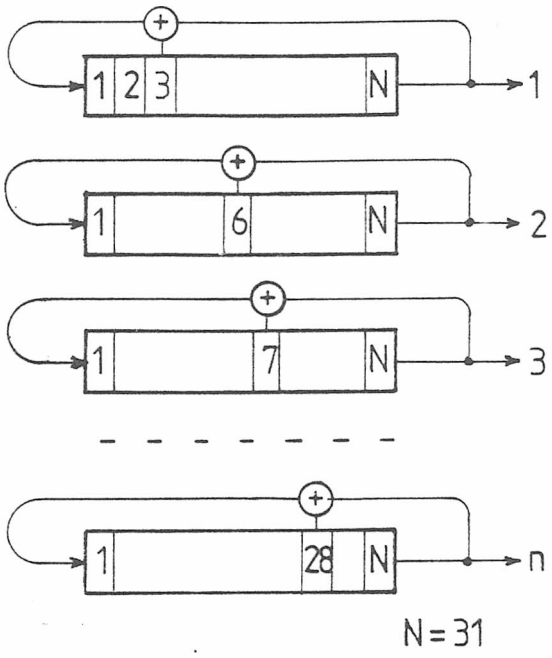


Figure 2·5·1a Multiple m-sequences

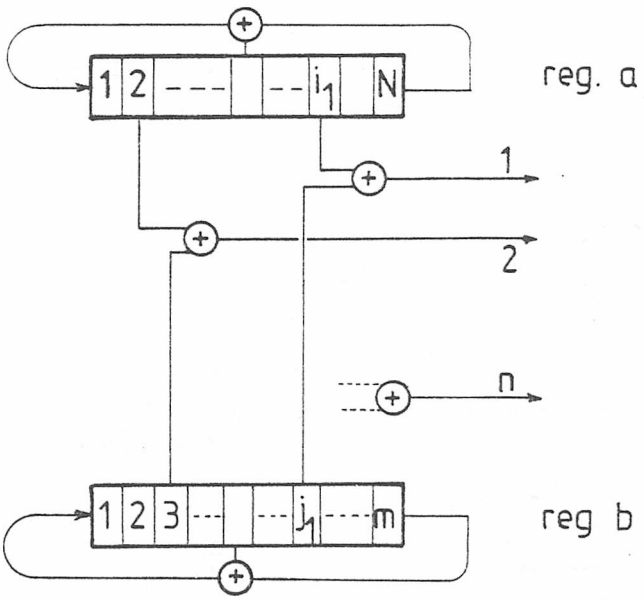


Figure 2·5·2a Two feedback shift register pseudorandom sequences

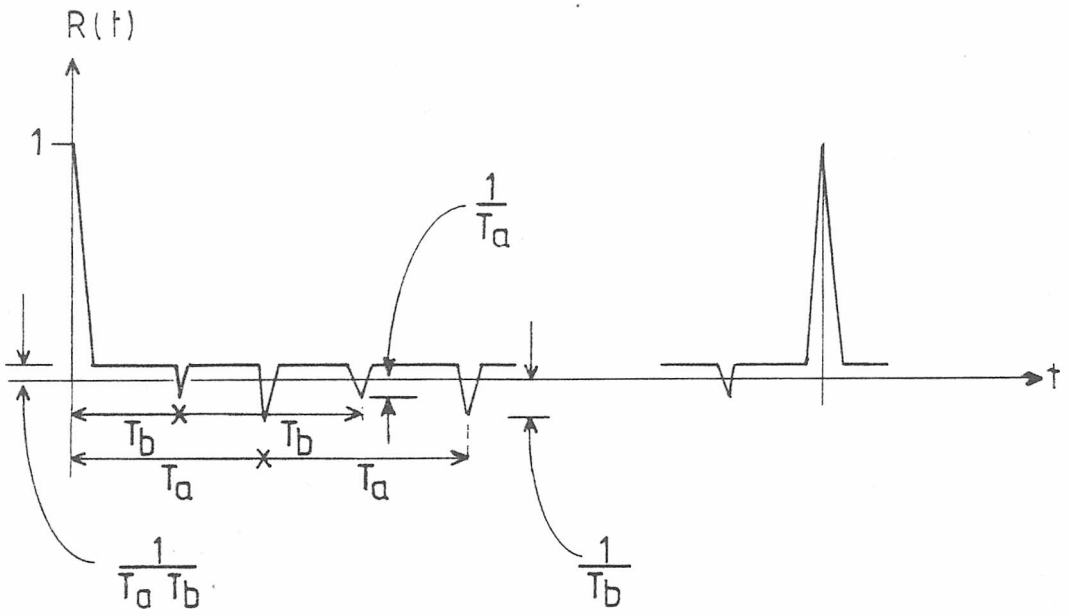


Figure 2·5·2b Auto-correlation function

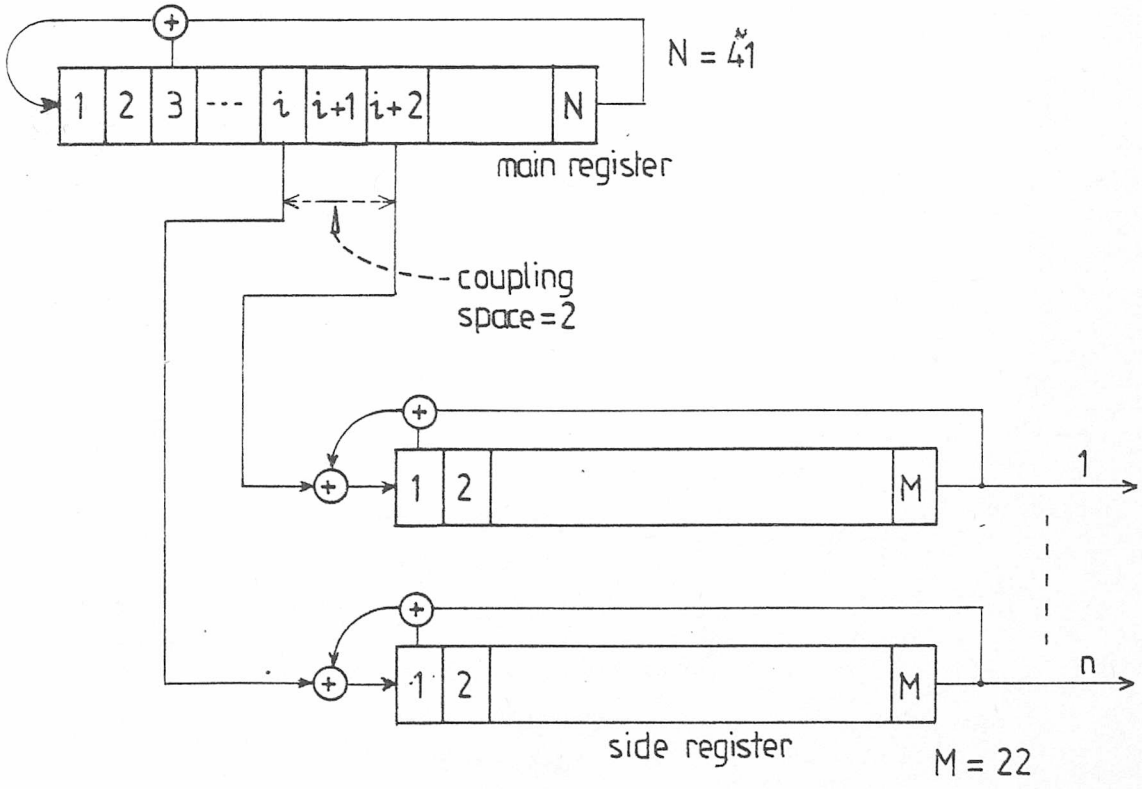


Figure 2·5·3a Cascaded shift registers

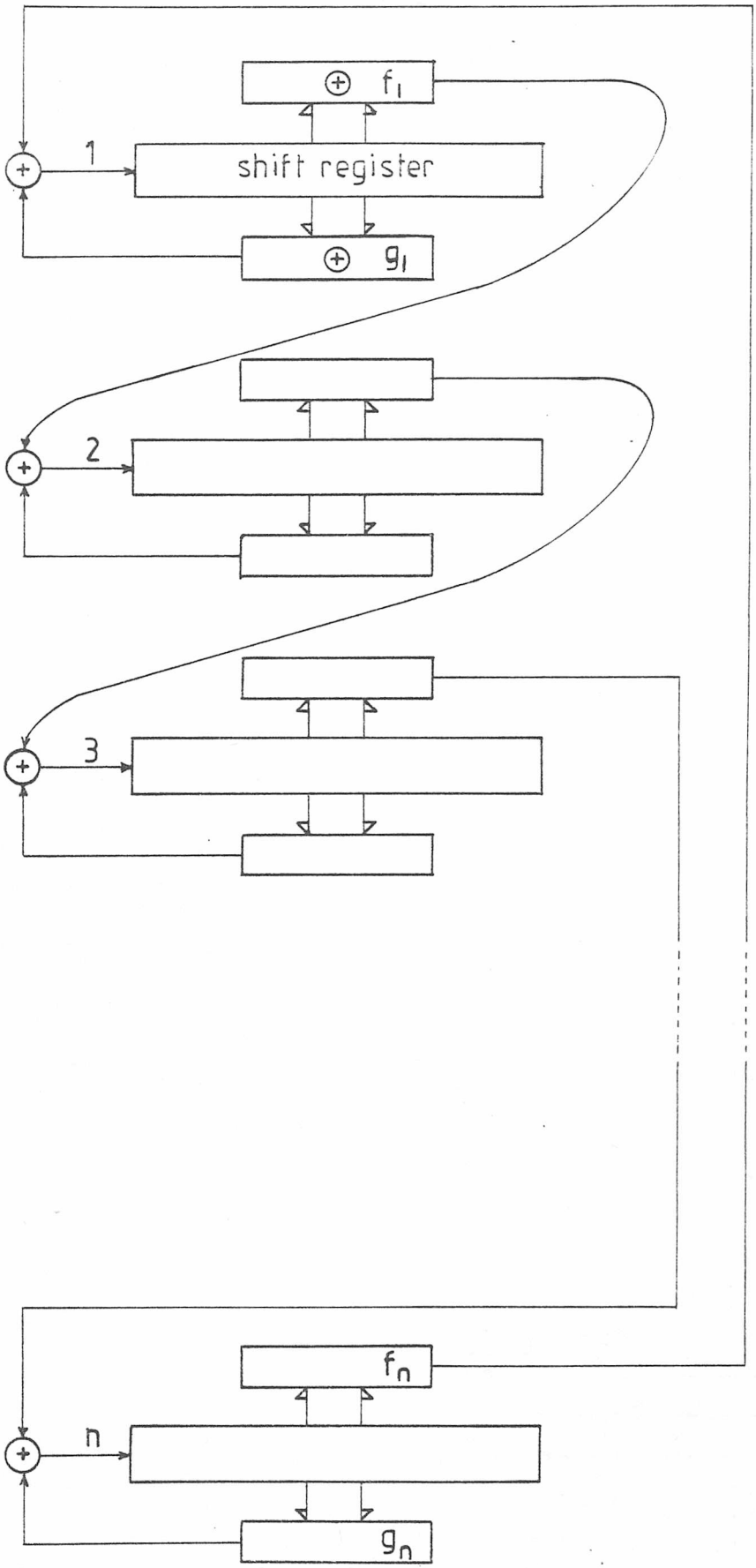


Figure 2.5.4a Decomposed shift register

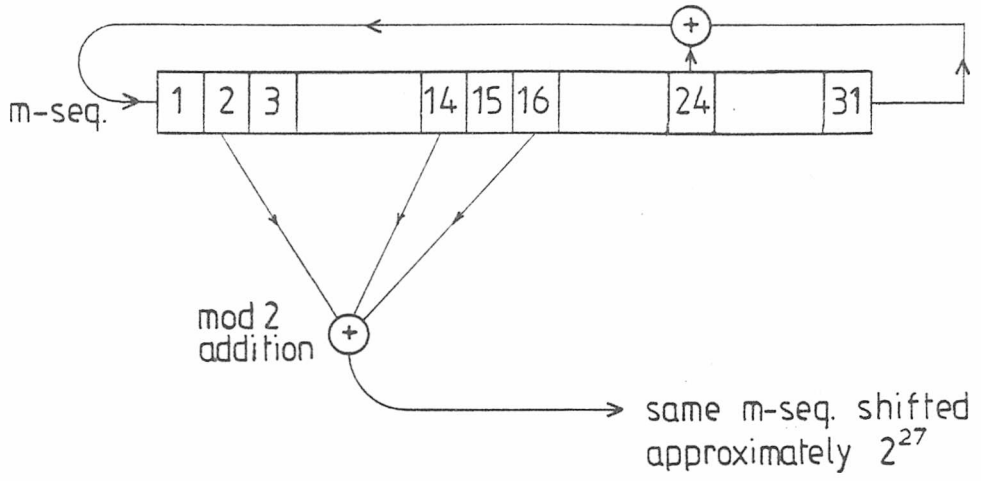


Figure 2.5.5a Producing a shifted m-sequence

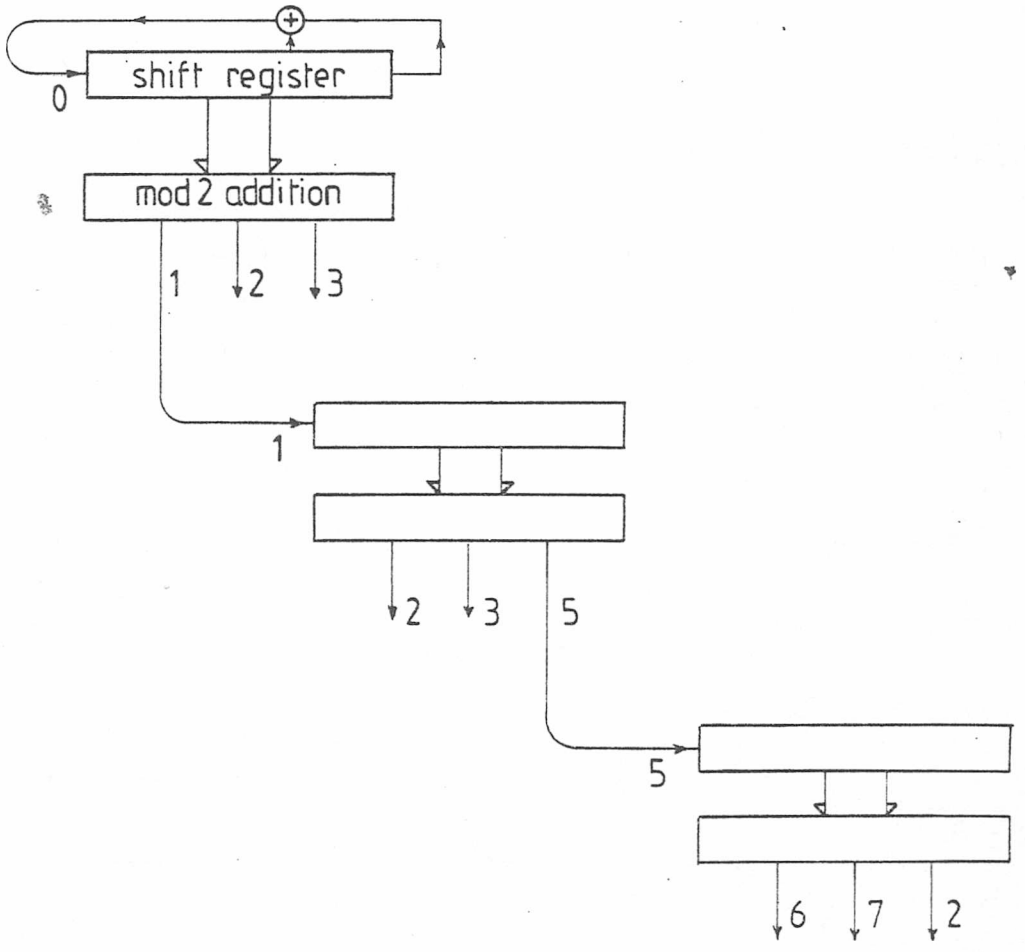


Figure 2.5.5b Implementation of 8 shifted m-sequences

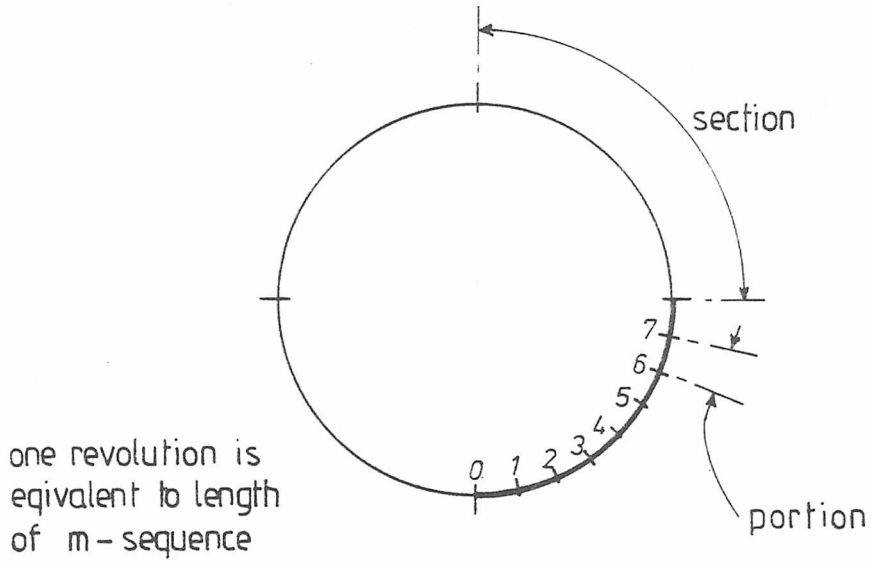


Figure 2·5·5c

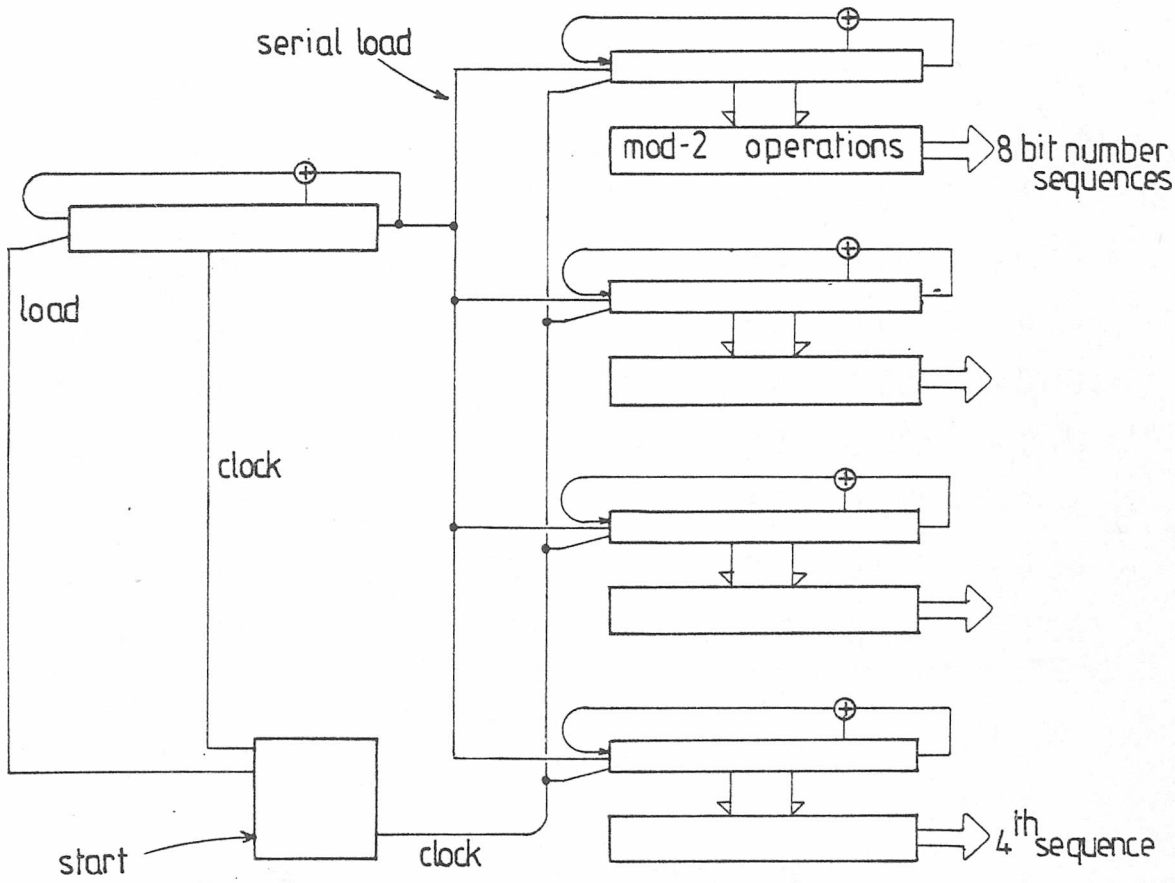


Figure 2·5·5d Multiple random number generator

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	TEST2	K-S	RUNS
1	126.4	5441.2	91.3	12.2	12.2	0.011	-0.06
2	126.7	5555.3	56.4	19.3	10.7	0.013	-0.22
3	126.6	5407.1	64.7	26.8	14.7	0.013	1.06
4	125.7	5529.2	64.2	17.2	12.4	0.016	0.52
5	125.5	5476.8	115.6	56.4	18.0	0.015	0.98
6	127.2	5397.8	55.6	28.5	25.7	0.012	0.77
7	126.1	5585.8	68.1	25.9	30.7	0.015	0.14
8	128.7	5505.1	64.6	12.3	17.1	0.014	0.48
9	127.9	5345.3	58.0	25.2	21.0	0.015	0.66
10	127.8	5482.0	64.0	28.0	24.0	0.009	1.77
11	127.0	5388.4	66.6	11.7	15.5	0.009	-0.19
12	128.4	5544.6	50.3	9.4	15.2	0.012	0.04
13	126.9	5433.2	61.1	20.0	11.2	0.008	-0.13
14	128.1	5457.4	44.9	15.9	18.7	0.012	-0.89
15	125.9	5383.2	53.3	19.6	16.1	0.017	1.02
16	127.6	5602.2	53.5	12.9	17.7	0.011	-0.38
	127.5	5400	82.5	24.9	24.9	0.030	±1.96

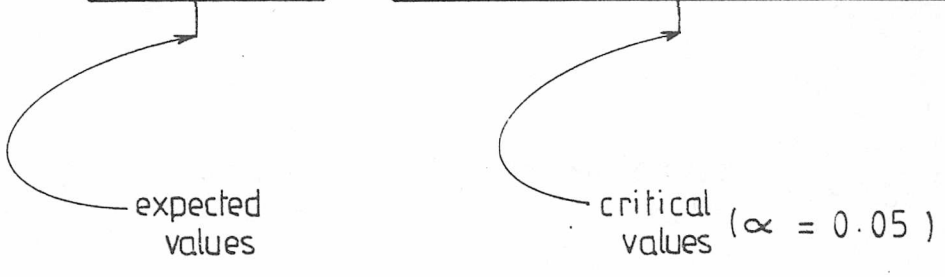


Table 2.6a Multiple m-sequence

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST 1	K-S	RUNS
1	128.5	5579.3	58.8	8.8	0.010	-0.91
2	125.7	5471.4	84.9	14.7	0.016	-0.39
3	127.4	5470.2	63.3	11.0	0.012	0.49
4	128.0	5486.5	82.8	73.4	0.009	1.36
5	127.4	5480.4	73.4	7.9	0.009	-0.73
6	127.7	5436.9	57.1	17.3	0.005	1.05
7	126.6	5388.9	70.2	11.5	0.012	-0.41
8	128.5	5548.2	79.7	15.7	0.013	-1.28
9	127.2	5453.8	54.0	14.5	0.010	0.85
10	127.4	5479.6	29.2	16.8	0.004	1.14
11	126.8	5376.9	78.5	32.8	0.011	-0.43
12	127.5	5553.1	48.2	26.3	0.011	-0.09
13	127.5	5411.6	71.9	15.1	0.012	2.30
14	129.4	5435.2	65.0	11.7	0.017	2.60
15	128.7	5494.5	72.4	22.0	0.011	-0.85
16	128.6	5506.4	61.7	18.1	0.012	-0.81
	127.5	5400	82.5	24.9	0.030	±1.76

expected values

critical values ($\alpha = 0.05$)

Table 2.6b Combination of two m-sequences

No of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST 1	TEST 2	K-S	RUNS
1	125.5	5465.5	71.3	19.7	19.7	0.019	0.50
2	128.4	5491.0	49.2	18.4	19.4	0.011	-0.32
3	126.8	5518.6	57.4	19.9	16.6	0.011	-1.26
4	127.1	5527.2	53.3	11.4	16.7	0.010	-0.92
5	127.2	5450.8	44.3	25.8	4.8	0.006	-0.57
6	128.2	5419.1	56.0	14.6	21.4	0.007	0.19
7	128.2	5394.2	64.3	15.3	10.3	0.010	-0.84
8	129.4	5489.7	49.2	18.8	14.9	0.016	-0.03
9	127.2	5559.0	55.4	12.4	16.4	0.011	-0.98
10	127.3	5466.4	43.8	8.3	17.1	0.007	-0.53
11	128.9	5433.9	66.8	20.9	16.2	0.022	-1.82
12	128.2	5380.9	52.9	13.4	11.2	0.011	0.39
13	126.0	5479.5	86.0	17.1	20.8	0.016	-1.77
14	127.5	5501.9	65.2	20.8	5.2	0.006	1.55
15	128.8	5535.8	58.8	22.3	14.0	0.014	3.08
16	128.0	5374.4	56.7	28.0	23.6	0.010	-0.36
	(127.5	5400)	(82.5	24.9	24.9	0.030	±1.76)

expected value

critical values ($\alpha = 0.05$)

Table 2.6c Shifted m-sequence technique

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	TEST2	K-S	RUNS
1	126.4	5519.7	55.6	26.3	26.3	0.012	-0.060
2	127.0	5416.6	66.3	8.9	14.6	0.009	-1.26
3	128.0	5510.4	69.6	9.7	24.4	0.010	0.22
4	127.1	5639.0	90.6	25.2	31.6	0.014	0.70
5	126.3	5590.1	47.0	33.3	6.2	0.012	0.26
6	126.4	5528.3	93.5	17.2	20.3	0.014	-0.50
7	126.1	5362.0	50.0	21.3	11.2	0.015	0.87
8	128.3	5429.9	65.6	30.1	52.8	0.009	-1.14
9	128.0	5420.5	38.1	14.8	24.8	0.008	-0.92
10	126.3	5599.4	93.5	14.4	17.5	0.016	-0.54
11	126.8	5399.0	46.7	18.6	13.3	0.011	-1.14
12	127.9	5570.6	106.8	20.2	21.3	0.013	-0.22
13	127.4	5395.7	52.7	7.6	26.6	0.006	-0.25
14	128.2	5536.4	175.9	40.1	16.0	0.014	-1.32
15	127.7	5336.1	94.0	17.8	8.7	0.010	1.14
16	126.2	5534.0	69.3	9.8	18.9	0.017	0.59
	(127.5	5400)	(82.5	24.9	24.9	0.030	1.76)

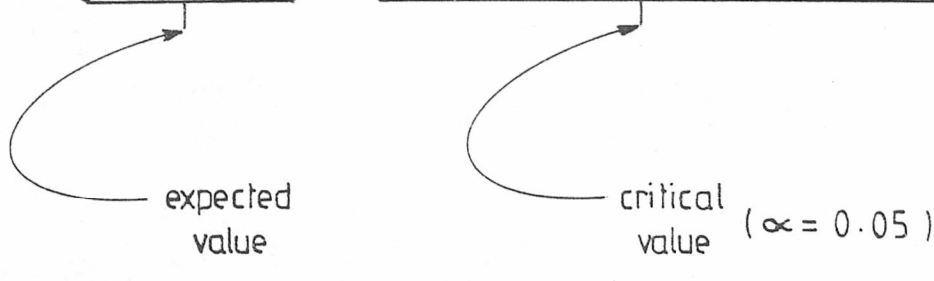


Table 2.6d Shifted m-sequence technique

No. of elements in each sequence = 4000

SEQ.	MEAN	VAR	CHI	TEST 1	K-S	RUNS
1	128.3	5390.4	67.4	21.0	0.010	-0.79
2	126.5	5387.9	45.7	22.2	0.012	-0.60
3	128.2	5485.7	52.7	25.3	0.009	0.62
4	126.3	5410.8	37.5	23.6	0.012	0.02
5	126.2	5627.7	60.8	8.8	0.015	2.44
6	127.1	5413.8	61.0	28.6	0.011	-2.33
7	127.0	5421.7	42.8	14.0	0.009	-0.28
8	128.8	5294.5	65.8	24.8	0.018	-1.57
9	128.7	5481.0	50.1	10.0	0.012	-0.19
10	125.8	5460.8	66.0	14.8	0.018	0.48
11	126.4	5539.1	54.0	7.5	0.012	-0.31
12	125.2	5372.2	63.0	19.6	0.018	-1.25
13	124.5	5426.5	84.1	28.4	0.026	0.56
14	121.2	5432.9	105.1	46.2	0.041	0.76
15	126.9	5412.9	78.2	13.6	0.012	-0.71
16	122.7	5445.7	74.8	17.9	0.028	0.55
	127.5	5400	82.5	24.9	0.030	±1.76

expected value

critical value ($\alpha = 0.05$)

Table 2.6e Proposed method

approx. phase shift	feedback from points 18 and 31 required MOD-2 addition stages
1×2^{27}	2, 8, 9, 18, 19, 28
2×2^{27}	4, 5, 16, 25
4×2^{27}	8, 10, 19
8×2^{27}	7, 16

approx. phase shift	feedback from points 24 and 31 required MOD-2 addition stages
1×2^{27}	2, 4, 16
2×2^{27}	2, 5
4×2^{27}	1, 7
1×2^{22}	1, 3, 4
2×2^{22}	1 5 7
4×2^{22}	1 9 13
8×2^{22}	1 16

Figure 2.6f Additions required to produce phase shifts

		multiple m-seq.	2-m seq.	shifted m-seq.		proposed method
A	Table	1	2	3	4	5
	CHI-test	2	2	1	6	2
	Test 1	6	3	2	5	3
	Test 2	2	-	1	4	-
	K-S test	0	0	0	0	1
	Runs test	0	2	3	0	2
	number of seq. which passed all tests	9	10	11	6	11

In section A each box contains the number of sequences failing the particular test ($\alpha = 0.05$)

Table 2.7a Analysis of results

Non-Uniformly Distributed Number Generation

3.0 Introduction

In the process of modelling the reliability behaviour of the system under investigation, the system is broken down into a number of components. Each component continually undergoes a change of state. At any time a component may fail and move into the non-operating state. Some time later it will be repaired and move back into the operating state. This Chapter is concerned with developing a mathematical model of the stochastic process undergone by components. By application of the model, an investigation of the accuracy of the modelling process is carried out, and increased control of the process is achieved.

The mathematical model is concerned with the generation of random numbers. The character and distribution of these numbers determines the behaviour of the system component. It can be seen that modelling the stochastic behaviour of many real systems requires the generation of random variables of a wide range of distributions. In fact a generator which can faithfully produce numbers of any distribution is desirable. Just such a generator, implementable in digital hardware, is described.

3.1 The Renewal Process

Consider the simplest stochastic point process known as the renewal process. This process may be defined as one which generates events, and since all such events are assumed to be identical, its essential interest resides in their times of occurrence. In this case, the word 'event' refers to a renewal point in the system.

Now consider a process which is characterised by a non-negative random variable, X, called its renewal time. The renewal time can be thought of as the amount of time the process has been running until the renewal event occurs. The random variable X has a probability distribution function (p.d.f.) f(x) given by:

$$f(x) = \lim_{\Delta x \to 0} \frac{\text{Prob } (x < X \leqslant x + \Delta x)}{\Delta x} \qquad , x \geqslant 0$$

with

$$\int_0^\infty f(x)\, dx = 1$$

The renewal times of the system $X_1, X_2, \ldots$ are mutually independent. Figure 3.1a shows a possible time schedule of a renewal process. $S_{i-1}, S_i, S_{i+1}, \ldots$ are renewal points of the system. $X_{i-1}, X_i, X_{i+1}, \ldots$ are renewal times.

The distribution of X_i can be completely determined from the p.d.f., f(x), but it is also convenient to use the cumulative distribution function (c.d.f.) F(x), giving the probability of renewal occurring by time x:

$$F(x) = \text{Prob } (X \leqslant x)$$

$$= \int_0^x f(u)\, du$$

clearly F(0) = 0 and $F(\infty) = 1$

Differentiation of the c.d.f. gives the p.d.f., thus:

$$f(x) = F'(x)$$

3.2 Renewal Rate

Consider a system, at a time x, at which renewal is known not to have occurred. The age-specific renewal rate, $\phi(x)$, is defined to be the limit of the ratio of the probability of renewal in $(x,x+\Delta x]$ to Δx. This can be expressed mathematically as:

$$\phi(x) = \lim_{\Delta X \to 0} \frac{\text{prob } (x<X\leqslant x+\Delta x \mid x\leqslant X)}{\Delta X}$$

$\phi(x)$ gives the probability that the end of the random time X lies in the interval $(x,x+\Delta x]$ given that NO renewal has occurred in $(0,x]$. The end of the random time X_i corresponds to the renewal point S_i. See Figure 3.1a.

Now for any two events A and B,

$$\text{prob } (A/B) = \frac{\text{Prob } (A \text{ and } B)}{\text{Prob } (B)}$$

But the event $(x<X\leqslant x+\Delta x$ and $x<X)$ is the same as the event $(x<X\leqslant x+\Delta x)$ since X starts at zero from the previous renewal point. Thus

$$\phi(x) = \lim_{\Delta x \to 0} \frac{\text{Prob } (x<X\leqslant x+\Delta x)}{\Delta x} \cdot \frac{1}{\text{Prob } (x<X)}$$

$$\phi(x) = \frac{f(x)}{1 - F(x)}$$

3.3 Reliability Applications

So far, the renewal process has been discussed as a sequence of events, renewal points. No physical interpretation has been given to the process. To gain an insight into the application of renewal theory, a process undergone by a component of equipment is defined. Initially the component is in working order, but after some time, X_1, it fails and is immediately replaced by a new component, which itself fails after an operational time of X_2, replacement is again carried out and the process continues. The failure time of the r-th component used is X_r

and the r-th failure occurs at time S_r, where:

$$S_r = Y_1+Y_2+\ldots+Y_r$$

The renewal points of the process occur at the failure of components. At the renewal points, the process returns to the start and is then completely independent of its previous behaviour, thus the process is time homogeneous. The distribution of life time of components is given by the p.d.f. of X , f(x), and completely determines the renewal process.

3.4 Alternating Renewal Processes

The renewal process has so far been used as a mathematical model of a component of equipment which is undergoing a failure replacement process. Replacement times have been considered to be negligible. The introduction of random replacement times, Y_i, distributed according to p.d.f. g(x), leads to the alternating renewal process. Renewal points now occur when the component fails or is repaired. To remove any confusion between the term renewal rate and repair, renewal rate is generally known as the hazard rate.

A mathematical model of random failure and replacement times of a component of equipment can now be produced. The alternating sequence undergone by the component is described by two embedded renewal processes. Figure 3.4a shows a possible time schedule for the component.

In the proposed Simulator, components can be any of eight states. Transition from state to state is dependent on both deterministic and stochastic processes. Embedded renewal processes are used to generate the random times between state transition.

3.5 Graphs of Hazard Rate

The age specific hazard rate $\phi(x)$ gives the probability of an immediate renewal point given that the time from the previous renewal is x. Figures 3.5a, 3.5b, 3.5c show some typical forms of $\phi(x)$.

Figure 3.5a is a graph of a constant hazard rate ($\phi(x)$ is monotonic). Monotonic hazard rates characterise a process in which the probability of immediate renewal is not dependent on the time since the last renewal. The exponential distribution is monotonic and is widely used in reliability theory. It will be seen later that the generation of a renewal process is considerably simplified if the process has a monotonic hazard rate.

Figure 3.5b shows $\phi(x)$ as an increasing function of x. This is called positive ageing. In reliability theory it corresponds to the fact that the older the component is, the more likely is its immediate failure. Negative ageing is shown in Figure 3.5c. In this case the older the component the less likely is its immediate failure.

It is well known that many components of equipment follow a relatively standard failure rate pattern as shown on Figure 3.5d. The pattern is known as the 'bathtub curve'. The distribution can be conveniently divided into three sections. Firstly the 'burn-in' or 'infant mortality' stage, which could be due to poor manufacture. Secondly the 'useful life' stage. Here the distribution is monotonic and it is by this portion that normal operation is described. Lastly the 'burn-out' stage where $\phi(x)$ increases rapidly. This stage represents the end of the useful life of the equipment as it begins to rapidly wear out.

3.6 Generator of Ordinary Renewal Processes in Discrete Time

The generator developed can simulate renewal processes in which the random time interval, X, between renewal points can take on only discrete values. The p.d.f. of a discrete random variable X is given by:

$$f_k(k\Delta x) = \text{Prob}\ (k\Delta x \leqslant X < (k+1)\Delta x)$$

and its c.d.f. is given by $F_k(k\Delta x)$.

The random time X can only have values $X=k\Delta x$, $k=0,1,2\ldots$ Figure 3.6a shows the generator block diagram. There are only three main elements :

1. A Generator of uniformly distributed random numbers E_i
2. A Function generator giving the quantity W_k.
3. A Comparator which compares E_i and W_k and gives an output pulse if $E_i < W_k$.

There are three assumptions made in the operation of the generator.

1. The random numbers are statistically independent and uniformly distributed in the interval [0,H). $H=2^m$, where m is the number of bits in which E_i is represented.
2. The quantity W_k can take on all values in the interval [0,H) i.e. W_k and E_i are not quantised.
3. The comparison between E_i and W_k is done instantaneously at the time points $i\Delta t$.

At each clock pulse, a random number E_i and a W_k value are presented to the comparator, which gives an output pulse if $E_i<W_k$. The c.d.f. of E_i is given by:

$$\text{Prob } (E_i \leqslant y) = \frac{y + 1}{H} \qquad ,0 \leqslant y < H$$

it can easily be seen that:

$$\text{Prob } (E_i < y) = \frac{y}{H}$$

Thus the probability of an output pulse is:

$$\text{Prob (output pulse)} = \text{Prob } (E_i < W_k) = \frac{W_k}{H}$$

Consider a process which started at time $k\Delta x=0$ with a renewal point, and no renewal has occured during $(0,(k-1)\Delta x)$, as shown in Figure 3.6b. The probability that a renewal point lies in the interval $(k\Delta x,(k+1)\Delta x]$ can be found from the age-specific hazard rate for the discrete random time interval X:

$$\phi(x) = \text{Prob } ((k-1)\Delta x < X \leqslant k\Delta x \mid (k-1)\Delta x < X) \; / \Delta x$$

$$= \text{Prob } \frac{((k-1)\Delta x < X \leqslant k\Delta x)}{\Delta x} \; \frac{1}{\text{Prob } (X > (k-1)\Delta x)}$$

$$= \frac{F_k(k\Delta x) - F_k((k-1)\Delta x)}{1 - F_k((k-1)\Delta x)}$$

A renewal point in the process corresponds to an output pulse from the comparator, and the probability of this happening is given by W_k/H Equating the two probabilities produces the relationship:

$$W_k = \frac{F_k(k\Delta x) - F_k((k-1)\Delta x)}{1 - F_k((k-1)\Delta x)} \cdot H \qquad ,k=1,2,3 \ldots$$

Thus for any given c.d.f., $F_k(k\Delta x)$, W_k values can be found enabling the simulation of the corresponding ordinary renewal process.

In many cases, the c.d.f. of the random time interval between renewal points can be completely continuous. Let $F(x)=\text{Prob}\ (X \leqslant x)$ be the continuous distribution of random time intervals. Time quantisation is necessary, thus $x=k\Delta x$, $k=0,1,2,\ldots$ By choosing Δx we can approximate $F(x)$ by $F_k(k\Delta x)$.

That is: $F_k(k\Delta x) = F(k\Delta x)$,at $k=1,2,3\ \ldots$

3.7 Probability Quantisation

The generator of renewal processes described in Section 3.6 is implemented using digital circuitry. It is now necessary to consider not only time but also probability quantisation. If m is the number of bits the random number E_i is represented by, then E_i and W_k can take on only values $0,1,2,\ldots 2^m-1$.

That is, the probability of renewal, $\emptyset(x)$, can take on values in the interval [0,1.0) in steps of $1/2^m$. If m is increased then the probability resolution is improved.

The W_k values calculated, necessary to simulate a renewal process with random time intervals distributed according to a particular discrete distribution, now have to be approximated by W_{kap}. Clearly this results in random time intervals which have a c.d.f. that differs from $F_k(k\Delta x)$. Let the expected c.d.f. of the random times be $F_{kex}(k\Delta x)$. From the results of Section 3.6:

$$W_{kap} = 2^m \frac{\left[F_{kex}(k\Delta x) - F_{kex}((k-1)\Delta x)\right]}{1 - F_{kex}((k-1)\Delta x)}$$

3.8 Calculation of W_k

Figure 3.8a shows a flow chart of the basic process involved in calculating the W_k values. It is assumed that the process starts at a renewal point, and therefore the first W_k value is calculated from:

$$W_1 = \frac{F_k(1\Delta x) - F_k(\phi\Delta x)}{1 - F(\phi\Delta x)} \cdot 2^m$$

where $F_k(0)=F(0)=0$, is the initial value of the c.d.f. of the random time intervals. W_1 must be approximated by an integer value, W_{lap}.

$$W_{lap}+0.5>W_1>W_{lap}-0.5$$

The value W_{lap} is then presented to the comparator. An output from the comparator results if $E_i<W_{lap}$, that is if the random number also presented to the comparator is less than W_{lap}. The comparator output pulse corresponds to a renewal point in the process, which results in the process returning to the start. Should no output pulse occur, then the process continues, and the next W_k value in the sequence is calculated. Each W_k value is compared with a random number E_i and the result determines whether the process returns to the start or continues to the next stage. Obviously at some time during the process a renewal point will occur.

Observing the process, it can be seen that the time intervals between renewal points (in the flow chart this time is given by k) are randomly distributed with distribution $F_{kex}(k\Delta x)$. Considering the probability quantisation, the distribution $F_{kex}(k)$ is an approximation to $F_k(k)$. Considering both probability and time quantisation $F_{kex}(k)$ is an approximation to $F(x)$ ($x=k\Delta x$, $k=0,1,2...$). Where $F(x)$ is the originally desired distribution of the time intervals.

Using the method described above to calculate W_k values, and assuming a 'perfect' generator of the random number E_i, tests have been carried out to estimate how good a fit $F_{kex}(k\Delta x)$ is to $F(x)$. The value D, associated with the Kolmogorov-Smirnov test, [see Section 2.6.6] is defined as the maximum absolute difference between the desired c.d.f., $F(x)$, and the expected c.d.f., $F_{kex}(k\Delta x)$. The quantity D has been plotted for two distributions considering different time and probability quantisations. The time quantisation is given by the value of Δx; the probability quantisation is given by the number of bits on which the comparator operates. Figure 3.8b gives the D value plotted for an exponential distribution of mean x=20 . Figure 3.8c shows the D value expected for a Weibull distribution. (λ=0.4, a=3.5).

It can be seen that the error (value of D) reduces as the quantum time step (Δx) is reduced. Considering Figure 3.6b this is certainly what would be expected. But this result only holds true for small probability quantum steps ($1/2^m$). There is a simple explanation for this. When the quantum time step is small, the values of W_k required to generate the distribution are also small. Integer representation of W_k by W_{kap} is more accurately achieved if the probability steps are small. This is particularly important at small values of W_k where the percentage error in the approximation to W_k can be large. An interesting characteristic, which can be observed on both figures, is that accuracy in the generation of the distributions is not always improved by reducing Δx. With large probability steps reducing the time step results in a decrease of accuracy.

3.9 Improved Calculation of W_k

Figure 3.9a shows a flow-chart of an improved method of calculating W_k values. It results in generating random time intervals with a distribution, $F_{kek}(k\Delta x)$, that is considerably closer to the desired c.d.f. $F(x)$. The method described here is much the same as the original method described in the previous section, but contains two alterations.

Firstly, during the calculation of the W_k value at time J the present c.d.f. value $F_k(J\Delta x)$, (represented by FN in the flow chart) and the previous c.d.f. value, $F_k((J-1)\Delta x)$, (represented by FO in the flow chart) are used. An improvement can be made here by replacing $F_k((J-1)\Delta x)$ by $F_{kex}((J-1)\Delta x)$. These two c.d.f. values are different due to the probability quantisation of the previous W_k values i.e. W_k, k=1,2,...,J-1. The expected p.d.f. $f(k\Delta x)$ values (represented by p in the flow chart) are calculated from the W_{kap} values and used to obtain $F_{kex}(k\Delta x)$, which is used in place of $F_k(k\Delta x)$.

$$W_{kap} = 2^m \cdot \frac{F_{kex}(k\Delta x) - F_{kex}((k-1)\Delta x)}{1 - F_{kex}((k-1)\Delta x)}$$

$$W_{kap} = 2^m \cdot \frac{f_{kex}(k\Delta x)}{1 - F_{kex}((k-1)\Delta x)}$$

using the terms of the flow chart, we have:

$$W_{kap} = 2^m \cdot \frac{P}{1 - FO}$$

$$P = W_{kap} \cdot \frac{(1-FO)}{2^m}$$

and also FN = FO + P, where $P = f_{kex}(k\Delta x) = f(k\Delta x)$

i.e. $F_{kex}((k\Delta x) = F_{kex}((k-1)\Delta x) + f_{kex}(k\Delta x)$

Thus by employing a form of feedback the error between the desired cumulative distribution and the expected distribution can be reduced, by reducing the error caused by approximating W_k by W_{kap}.

Secondly, to reduce the effects of time quantisation, a 'time shifted' distribution can be generated in preference to the actual distribution desired. See Figure 3.9b. Distribution $F_s(x)$ is the desired distribution shifted by $1/2\,\Delta x$ in time. That is,

$$F_s(x) = F(x + 0.5\Delta x)$$

Obviously this is not a proper distribution as $F_s(0)=0$. However if an attempt to generate $F_s(x)$ leads to a resulting distribution that is a better approximation to $F(x)$, then the process should be considered acceptable.

The programmed ordinary renewal process attempts to generate random time intervals with the discrete distribution $F_{sk}(k\Delta x)$, which is a discrete approximation to $F_s(x)$. Due to the probability quantisation of W_k, the random time intervals actually have a distribution $F_{skex}(k\Delta x)$. It is easily seen from Figure 9.6b that this distribution is a better approximation to $F(x)$ than would have been obtained by an attempt to generate $F_k(k\Delta x)$.

Using the methods described in this section to calculate the desired W_k values, tests have been carried out to estimate the goodness of fit of generated distributions. As in the tests carried out in the previous section, it was assumed that a 'perfect' generator of random number E_i was available. The Kolmogorov-Smirnov test has again been used as the discriminating test.

The value D has been plotted for a Weibull distribution, using the feedback technique to improve the process. Different time and probability quantisations have been considered and the results are shown in Figure 3.9c. The value D has also been plotted for the same Weibull distribution generated using both feedback and shifting in the generation process. The results from this test are shown on Figure 3.9d. It can be clearly seen, from the results obtained, that the error (value of D) reduces as the quantum time step (Δx) is reduced. Unlike the method first proposed to calculate W_k, this result holds true for any value of probability quantisation. The characteristic, observed with the previous method, of the reduction of Δx not necessarily resulting in a lower D value is certainly no longer true. It can also be seen that when using a very small probability step, feedback gives little improvement in D value. This is as would be expected, because an integer approximation of W_k should be more accurately achieved when the probability time step is small. The effects of feedback, as observed from the results obtained in this section show that feedback makes it possible to reduce the D value by reducing Δx, when a large probability step is being used.

The effect of time shifting the distribution can be clearly seen to have reduced the D value. The characteristics of the feedback process are unaffected by the shifting.

3.10 Tests on Number Generator

The random times between state transition are distributed according to known functions and the Simulator is expected to generate random time values which closely resemble those functions. Techniques have been developed which are expected to improve the accuracy of the generated distribution over a range of time and probability quantisation. This section presents statistical tests employed to confirm the random number

generation process and evaluate improvement techniques.

A requirement for the operation of the non-uniformly distributed random number generator (which is another name for the renewal process generator) is a source of uniformly distributed random numbers. This need can be fulfilled by the uniformly distributed random number generator proposed and tested in Chapter 2.

A description of the statistical tests carried out is given in Appendix A1, along with probability distributions frequently encountered in system modelling. A discussion of the application of the distributions is given, and the theory invoked to calculate the W_k values necessary to program the generator is explained. The results from the statistical tests carried out, along with a graph of p.d.f. for each distribution generated, are presented at the end of Appendix A1 sub-sections. A table of results along with critical values for tests carried out is also presented.

The methods proposed in Section 3.9 to improve the calculation of W_k values, and so improve the distribution 'fit' are investigated. The abbreviation F.B. (feedback) shown in the tables of results corresponding to the first improvement proposed in Section 3.9 viz. that of replacing $F_k((J-1)\Delta x)$ by $F_{kex}((J-1)\Delta x)$ in the calculation of W_k. Abbreviation T.S. (time shift) corresponds to the time shift improvement; that is generating $F_s(x)$, $[F_s(x)=F(x+0.5\Delta x)]$, in preference to $F(x)$. All other distributions are generated by the simpler method of calculating W_k [Section 3.8].

The effects of the quantisation of probability values are investigated by experimenting with 8-bit and 16-bit random numbers. This requires the W_k values to be calculated with an 8-bit and 16-bit representation respectively. The 8-bit generator is the actual generator tested in Chapter 2, and the 16-bit generator is simply an expanded version.

The effects of time quantisation, step size Δx, are investigated. Various distributions are generated with Δx values of 1, 1/2 and 1/4. Altering the step size and the random number size proves an interesting test for the various methods of calculating W_k values.

Each test contains 1600 or 3200 samples, the lower figure being chosen when small Δx values are being used. This is because of the resulting increase in time required to gather results.

3.11 Examination of Statistical Test Results

As explained in Appendix A1, for the frequency test, gaps were grouped together where the empirical frequency fell below five. A minimum grouping of five was a requirement of the chi-square test if accurate results were to be expected. When generating a random number of a particular distribution with $\Delta x=1.0$, the grouping arrangement led to a test with a particular number of degrees of freedom. (A further explanation of this is given in Section 2.6.3) When the same distribution was then generated with $\Delta x=0.5$ or $\Delta x=0.25$ the number of degrees of freedom for the test was maintained at the previous value. This was not necessary as a larger number of groups could have been formed each at frequency greater than five. The result of this is that the chi-square test did not fully test these distributions, though the loss of information at $\Delta x=0.5$ could not have been too great. The test was carried out this way because distributions generated with $\Delta x=0.25$ were few, and maintaining a fixed number of degrees of freedom for each distribution enabled a simpler comparison to be made between test results.

The Kolmogorov-Smirnov test, D value, did not suffer from the grouping arrangement of the chi-square test. All distributions were fully tested. A comparison of the test results was aided by keeping the number of samples in each distribution to 2 fixed values viz. 3200 and 1600.

There are several clear points which can be concluded from the test results. They are :

1. The feedback improvement to calculation of W_k values leads to an improvement in both chi-square and Kolmogorov-Smirnov test results.

2. The time-shift improvement to the W_k calculation causes a dramatic improvement in the Kolmogorov-Smirnov test result but has a disastrous effect on the chi-square result. The effect on chi-square result was reduced if feedback was also employed or if Δx was reduced. Certainly time shifting the distribution cannot be considered as an improvement. A reason for this can be seen by examining the p.d.f. of the time shifted distribution.

$$\text{Shifted p.d.f.} = f_s(x) = \frac{d}{dt}(1-e^{-(\lambda x + 0.5\Delta x)}) \quad , x \geqslant 0$$

$$f_s(x) = e^{-0.5\lambda\Delta x} . \lambda e^{-\lambda x}$$

This is obviously an illegal distribution, as was previously stated in Section 3.9. Integration over the complete range of x reveals:

$$F_s(x) = e^{-\frac{1}{2}\lambda\Delta x} . 1 \neq 1$$

Which is always less than 1.0, and diverges further as Δx is increased. Figure 3.11a shows the discrete p.d.f. $f_k(k\Delta x)$ which would be generated to represent the continuous p.d.f. $f(x)$.

Figure 3.11b shows the shifted p.d.f. $f_s(x)$ and the new discrete p.d.f. $f_{sk}(k\Delta x)$ relative to the desired p.d.f. $f(x)$. The figure displays the effect of T.S., and since the chi-square test is based on the 'fit' between the actual generated distribution and $f(x)$, it can be seen that the T.S. technique has a considerable effect on that fit.

3. Reducing the probability quantisation of W_k values is a sound method of improving both statistical test results.

4. If one factor had to be singled out as the most effective way of improving test results it must be that of reducing the Δx value. Generally, whenever Δx was reduced, the results improved. In Section 3.8 it was stated that, when the simple method of calculating W_k was employed, "accuracy in the generation of the distribution is not always improved by reducing Δx". The only evidence found for this was during the generation of the exponential distribution when Δx was reduced from 0.5 to 0.25; the D value reduced from 0.0465 to 0.0469 when a reduction in D value would have been expected. When feedback was employed for the calculation of W_k values during generation of the Erlang distribution, the D value went from 0.0548 to 0.039 when Δx was changed from 0.5 to 0.25. This is an improvement which was not observed with the exponential distribution where feedback was not employed. However it must be said that the D value improved when Δx went from 0.5 to 0.25 for the Erlang distribution without the aid of feedback.

It was decided that large samples would be generated for each distribution previously considered, and tests carried out. An 8-bit W_k value was chosen as it was considered to be the most practical size. The time step Δx was set at 1.0 to enable full use of the chi-square test. The feedback improvement technique was employed in the calculation of W_k, with all distributions except the exponential one. Using feedback for the exponential distribution results in loss of the memory-less ability in the generation of the W_k values, which is considered a great advantage. The sample size was 9600 and graphs of the p.d.f.'s generated are shown on Figures 3.11c to 3.11f. The desired p.d.f. has been shown for comparison. Table 3.11g contains all results and critical values.

All distributions passed the chi-square test. The poorer result gained for the exponential distribution is due to the simpler method of calculating W_k. The D values produced did not pass the Kolmogorov-Smirnov test but this was expected, and is a result of chosing a large Δx value. The D values which occurred are very close to the expected values.

3.12 Conclusion

Each component of equipment which makes up the reliability system undergoes a continual change of state. The random times between state transition are described by embedded renewal processes. A digital hardware simulator of the embedded renewal processes can be simply constructed from a digital comparator and a random number generator. The distribution of random time intervals, X, generated is completely controlled by the W_k values, presented to the comparator. W_k values range between 0 and 1 and correspond to the probability of instantaneous renewal. This value may vary with the age of the process. The ability to simulate $W_k(t)$, $t>0$ leads to generation of random times of any

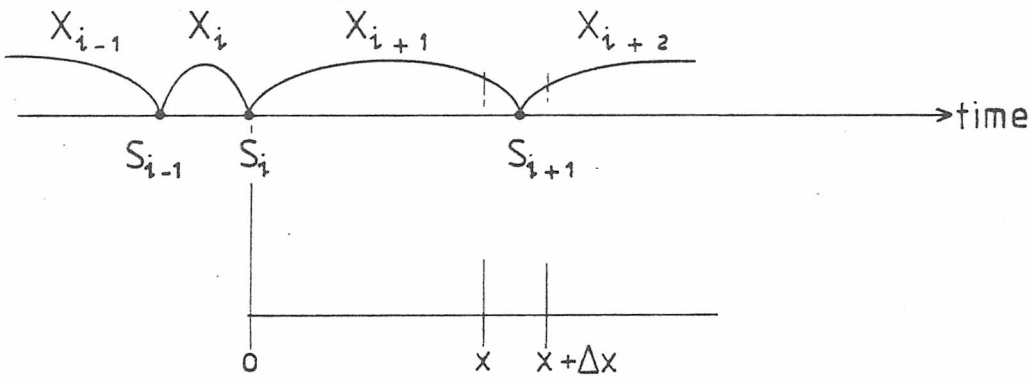
distribution.

Amplitude quantisation of W_k and time quantisation of X influence the generator's ability to accurately generate time intervals with a prescribed distribution. However techniques have been developed which achieve an accuracy at quantisation values previously considered poor.

The feedback technique of calculating W_k values was successful in improving both the p.d.f. and c.d.f. distributions. Time shifting achieved the improvements expected in Kolmogorov-Smirnov test results, but severely affected the p.d.f. distribution. This therefore cannot be regarded as a useful technique in improving random time interval distributions.

The time step Δx had a considerable influence over the statistical test results. As was expected, from previous investigations , when the feedback technique is employed, the c.d.f. 'fit' is consistently improved by reducing Δx.

The economic hardware design of the generator, coupled with its capacity for high speed operation, make it well suited to modelling reliability for multi-component systems. Within the Simulator a single random number generator is employed to model the behaviour of each system component. The parallel stream of independent uniformly distributed random numbers required to maintain the component processes is supplied by the generator proposed in Chapter 2.



• renewal point

A renewal Process

Figure 3.1a

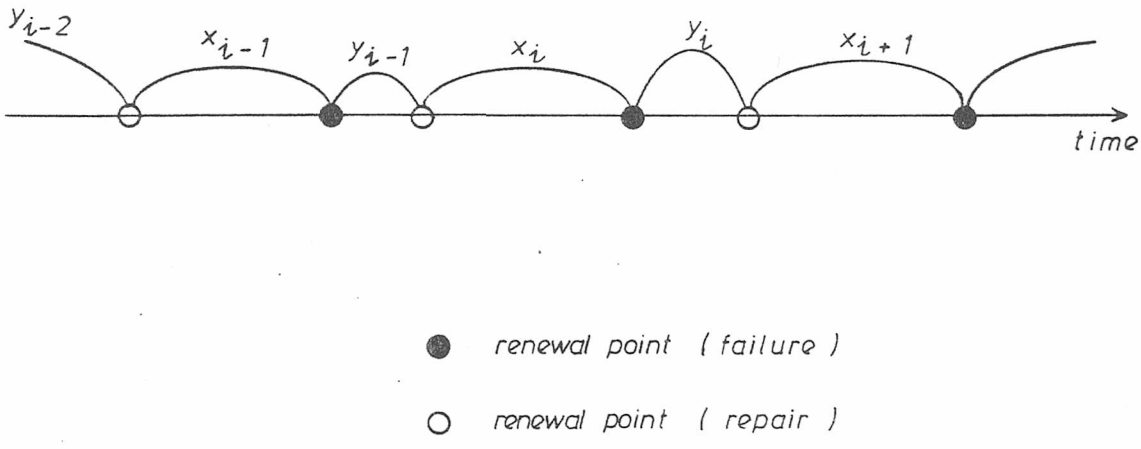


Figure 3.4a Alternating renewal process

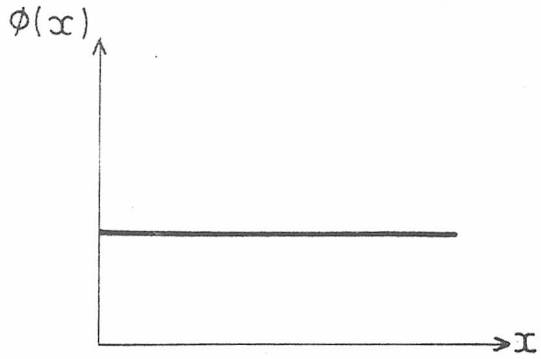


Figure 3.5 a

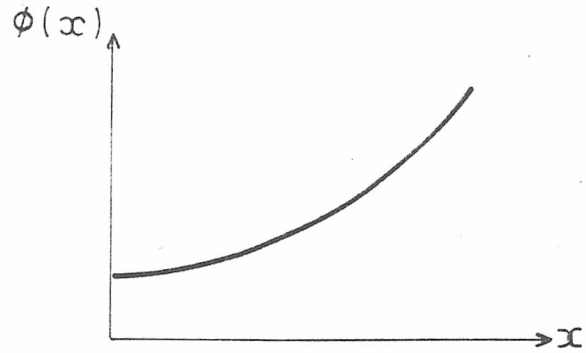


Figure 3.5 b

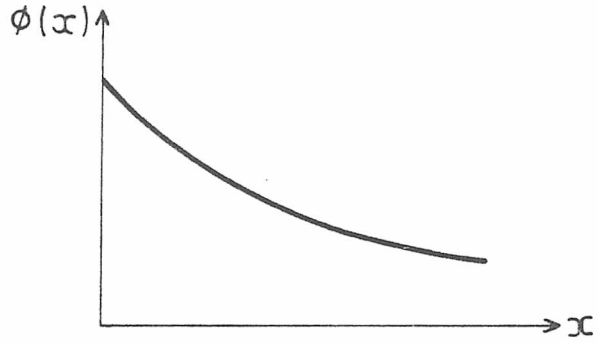


Figure 3.5 c

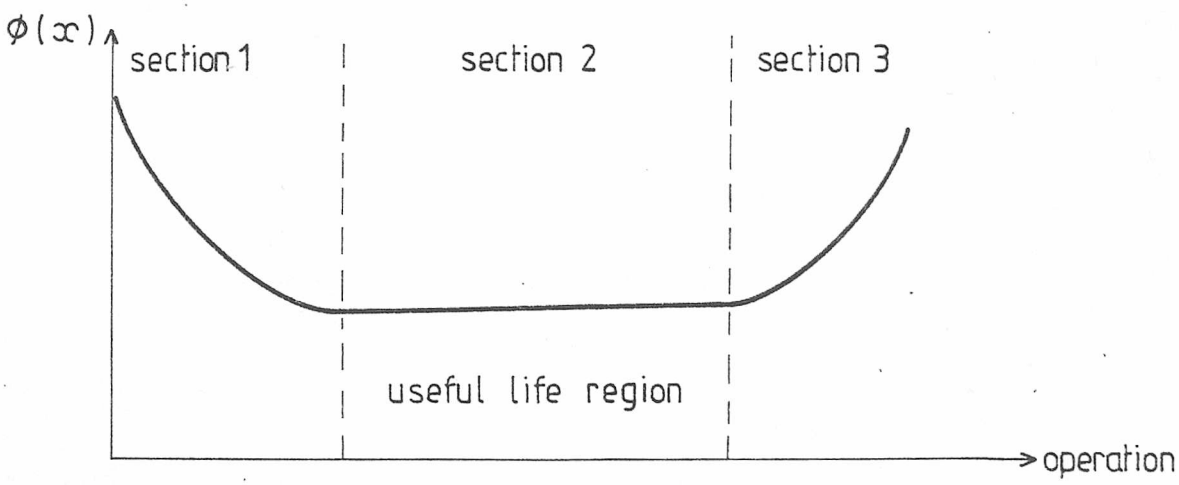
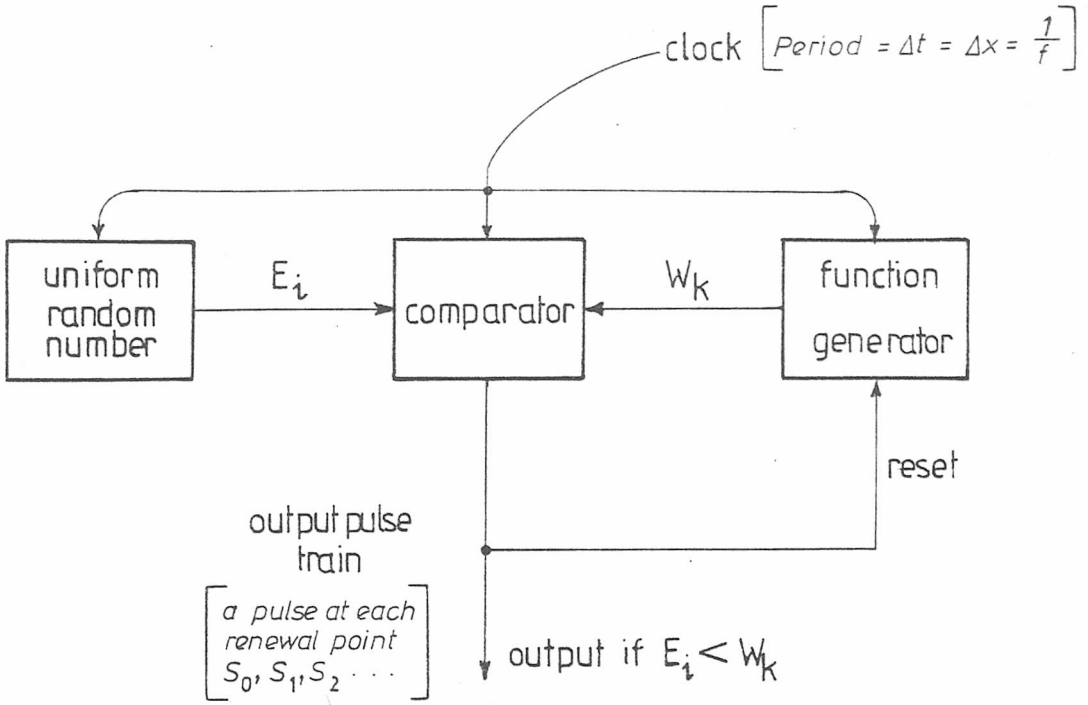


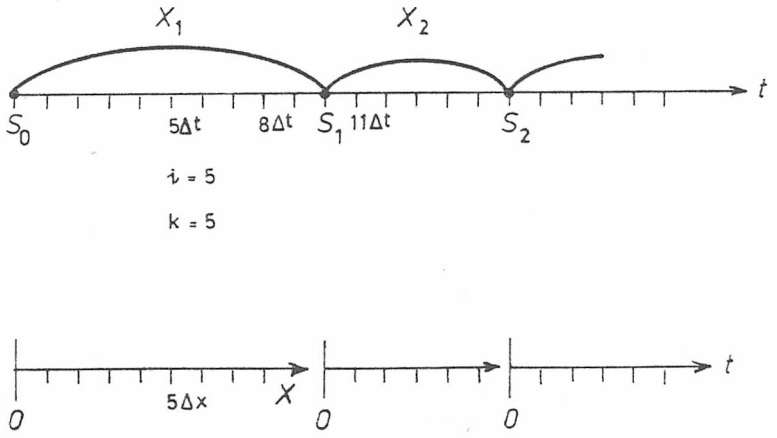
Figure 3.5d

Graphs of hazard rate



Renewal Process Generator

Figure 3.6a



A renewal process in discrete time

Figure 3.6b

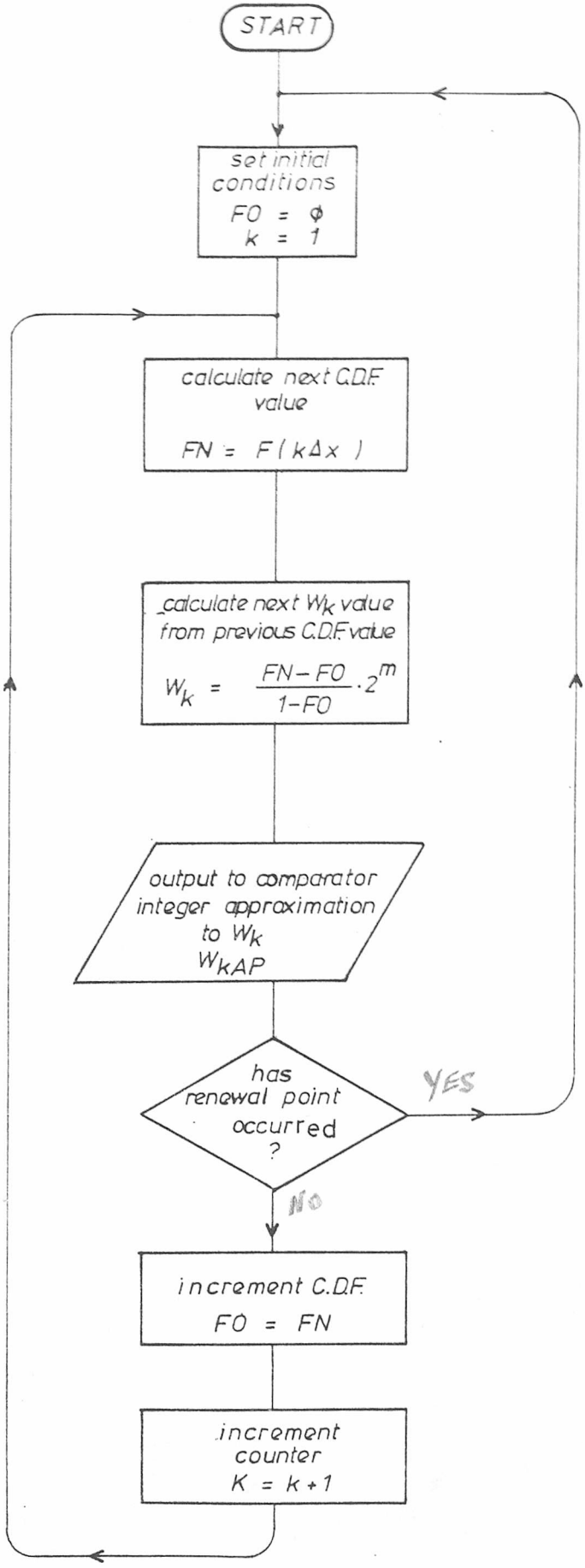


Figure 3.8a

simple method of calculating W_K

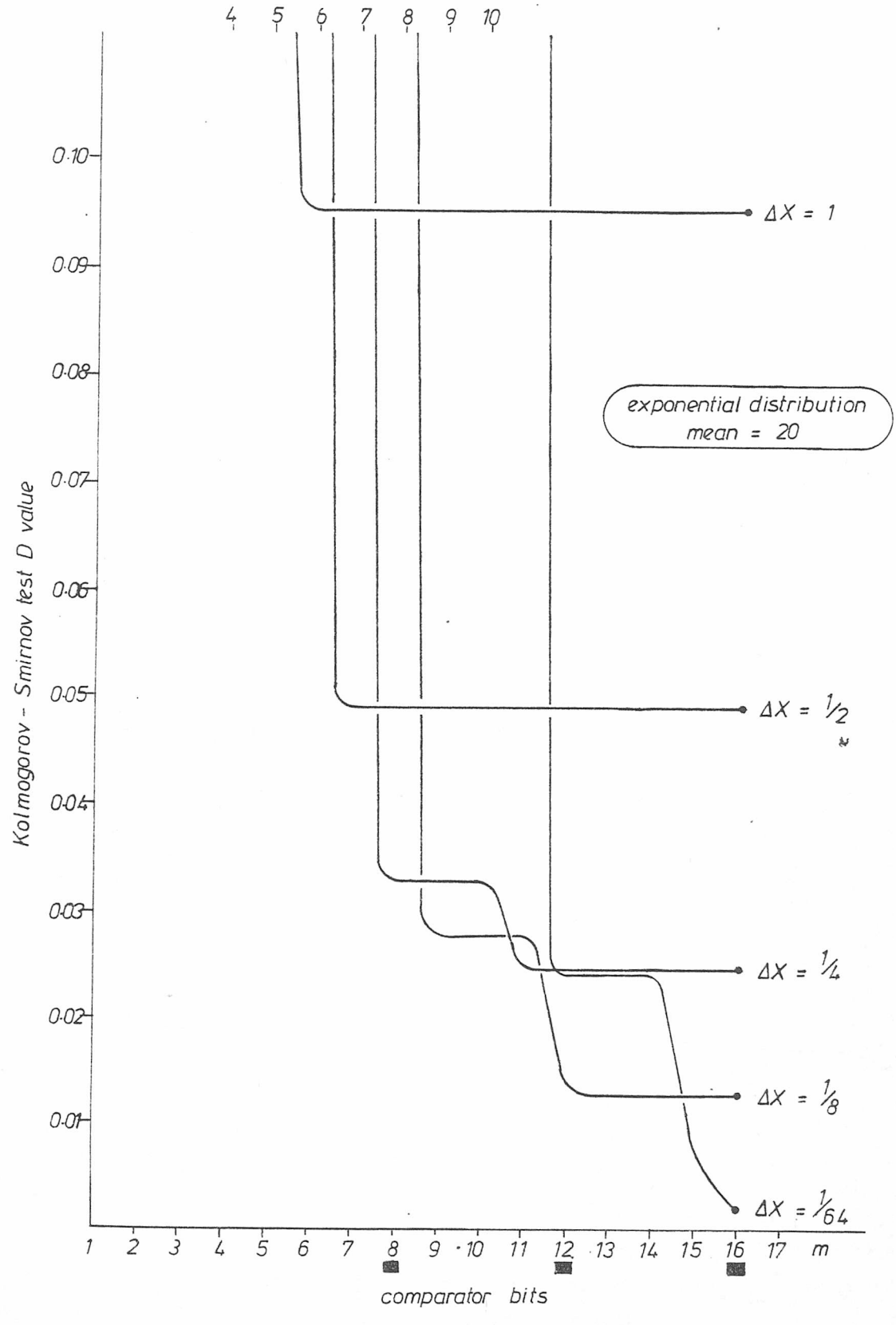


Figure 3·8b Simple method of calculating W_k

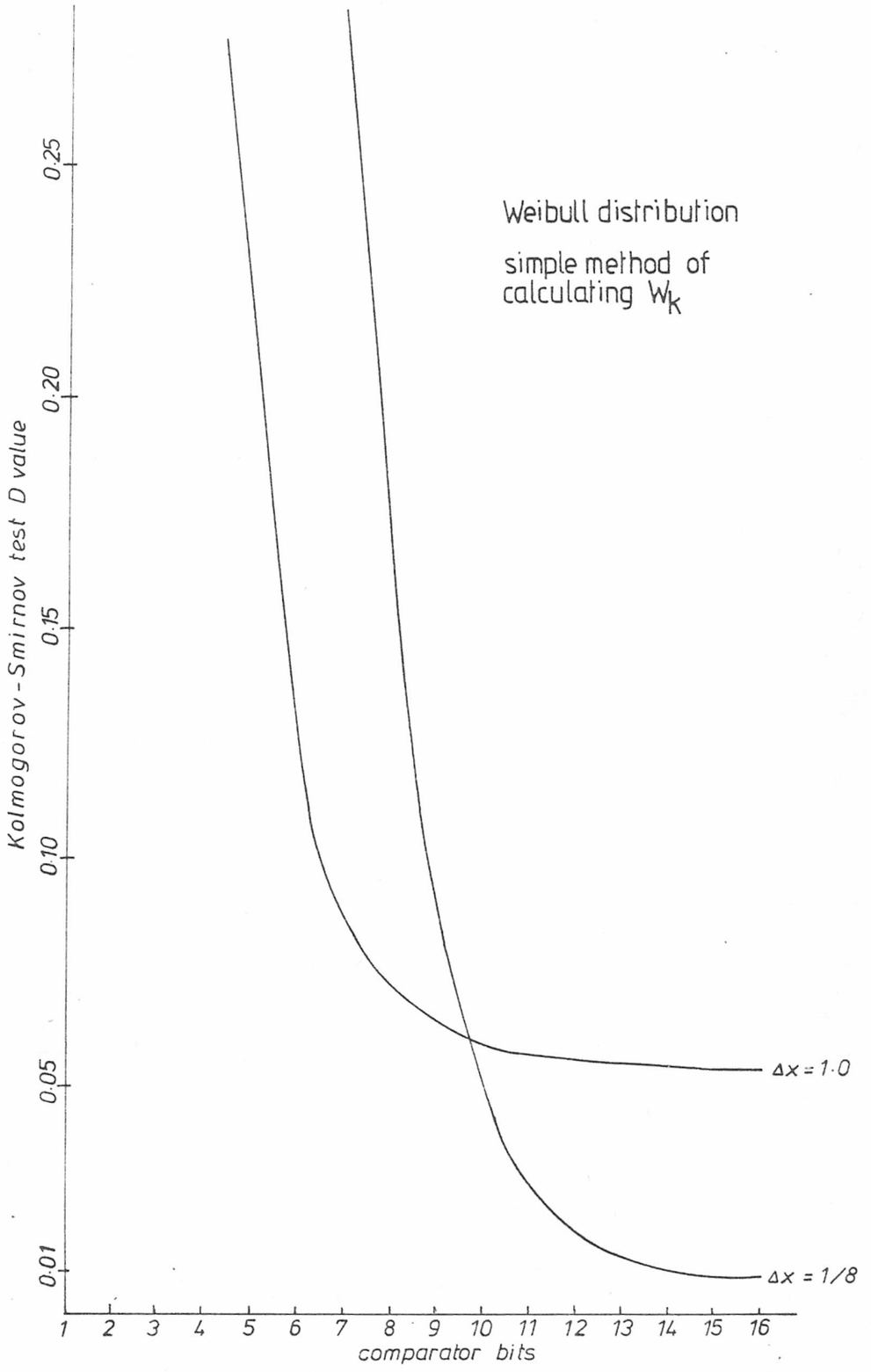

Weibull distribution
simple method of calculating W_k
Kolmogorov-Smirnov test D value
0·25
0·20
0·15
0·10
0·05
0·01
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
comparator bits
Δx = 1·0
Δx = 1/8

Figure 3·8c

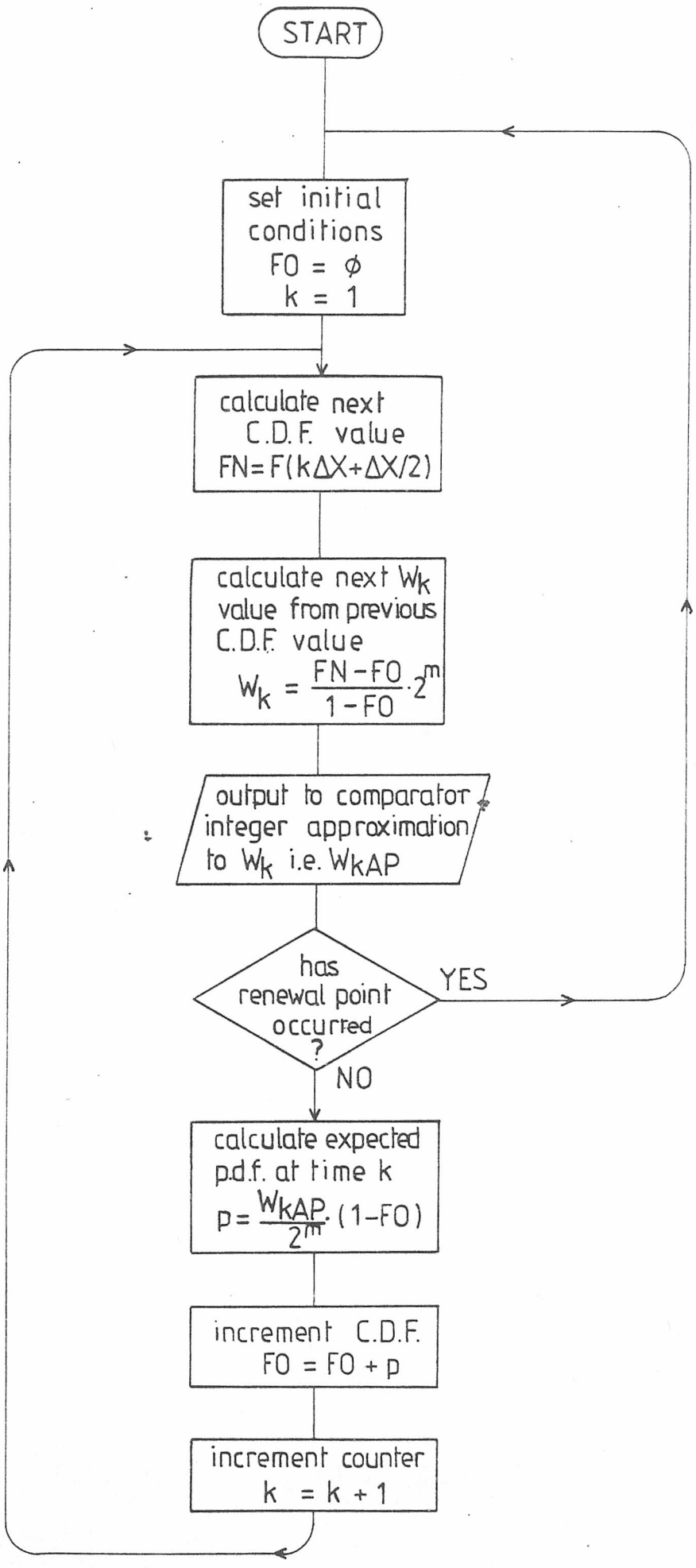


Figure 3.9a Improved calculation of W_k

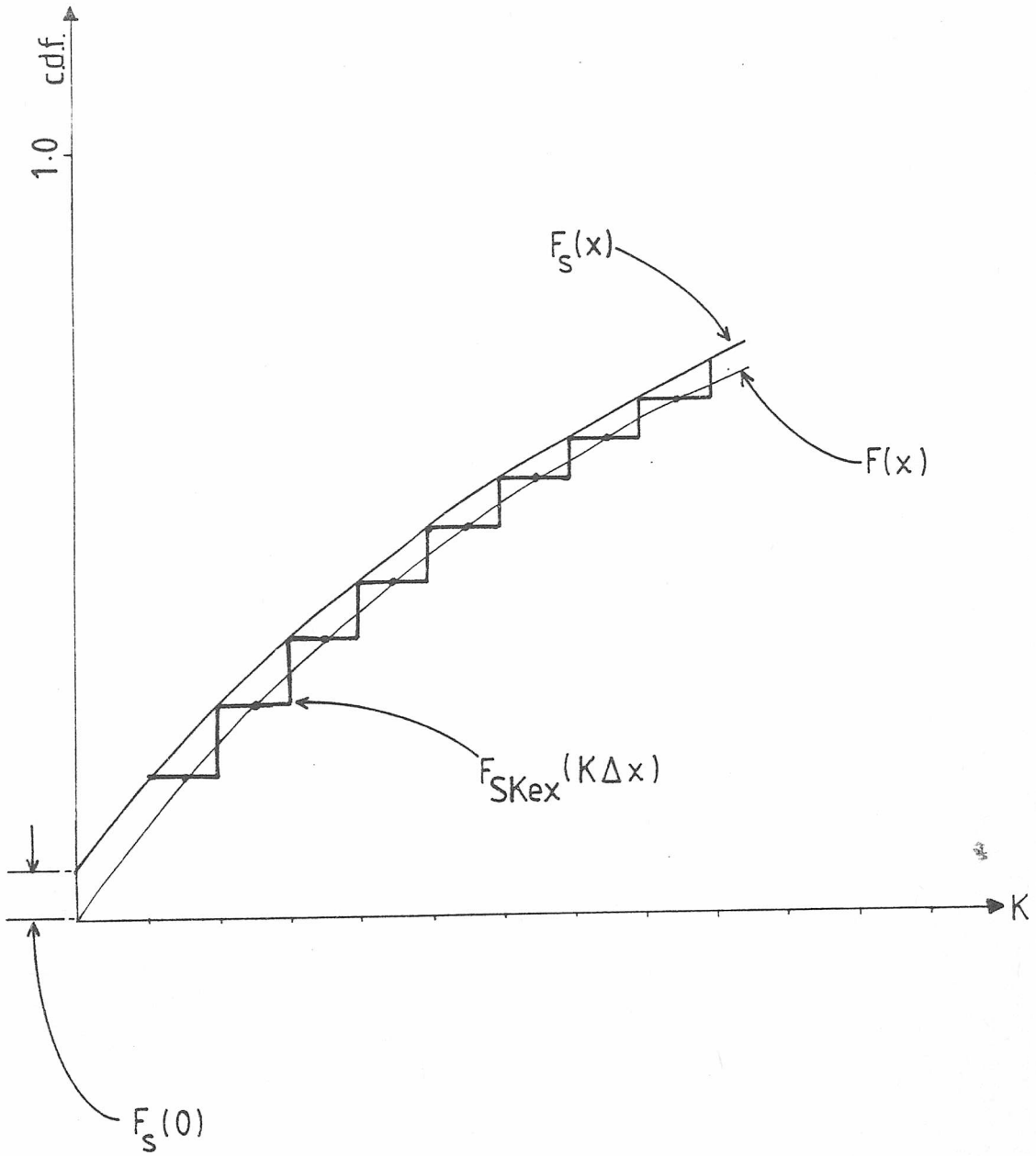


Figure 3.9b

Generation of $F_s(x)$ in place of $F(x)$

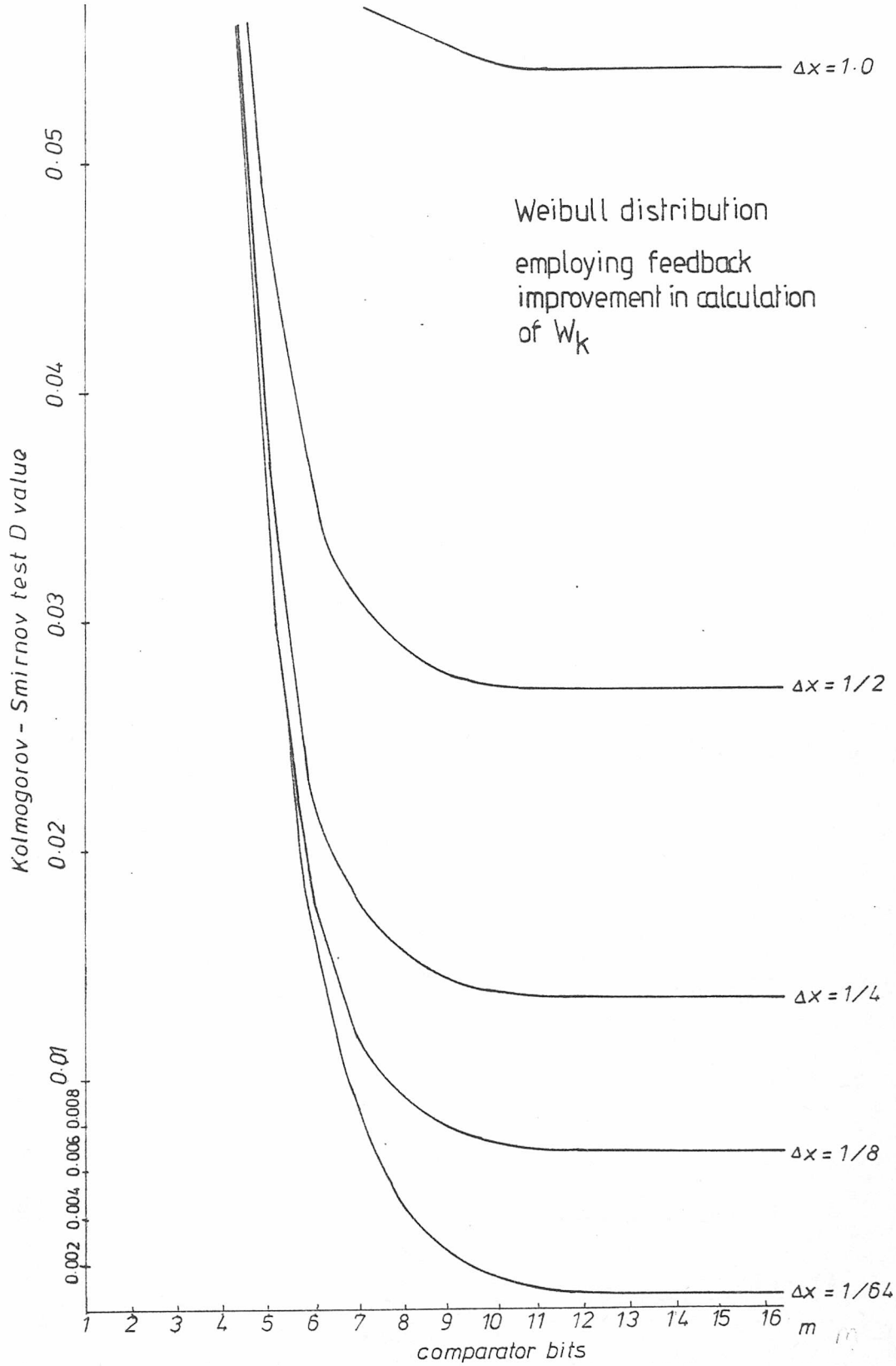

Δx = 1·0
Weibull distribution
employing feedback
improvement in calculation
of W_K
Δx = 1/2
Δx = 1/4
Δx = 1/8
Δx = 1/64
Kolmogorov - Smirnov test D value
0·05
0·04
0·03
0·02
0·01
0.008
0.006
0.004
0.002
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 m
comparator bits

Figure 3·9c

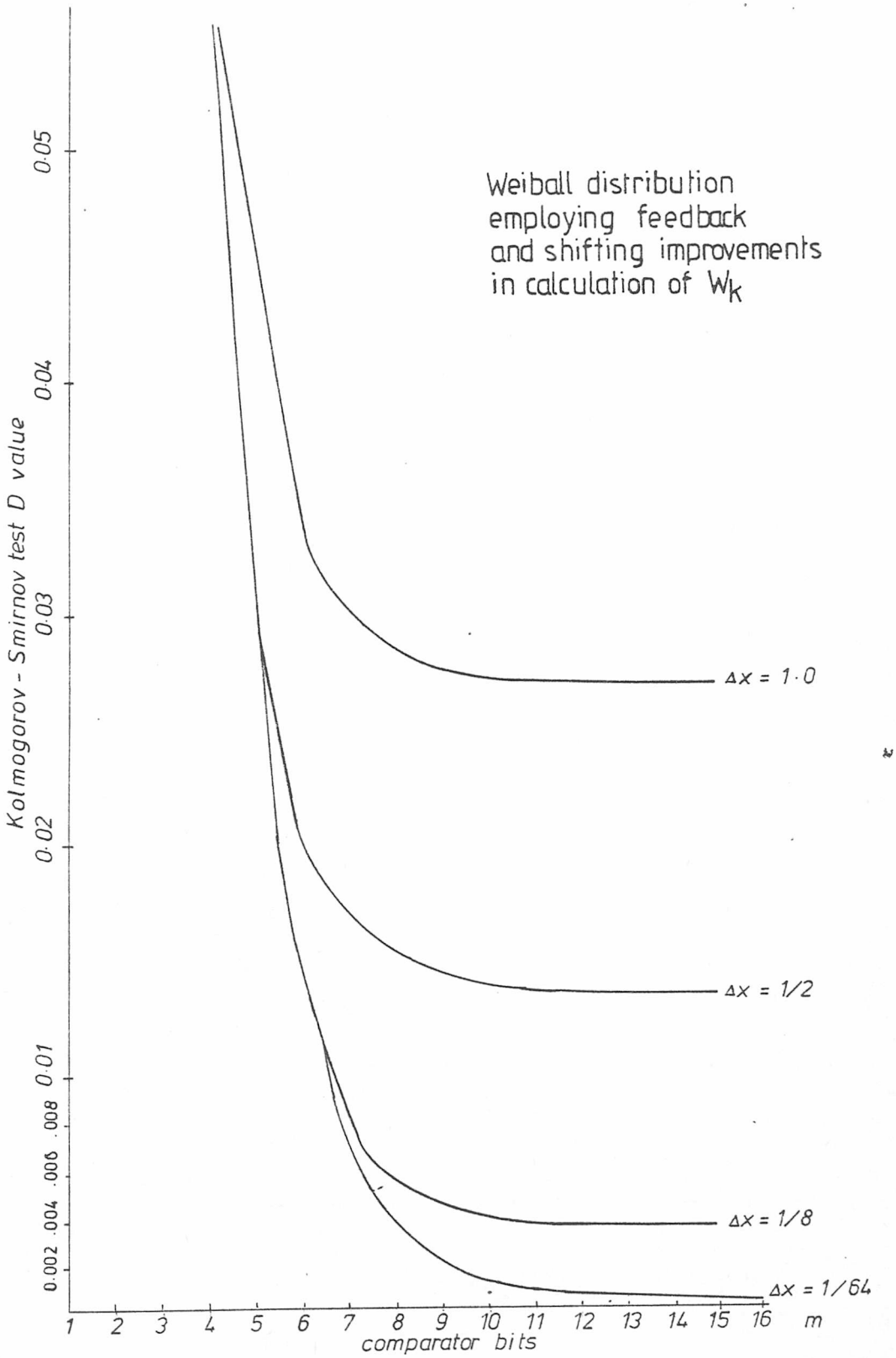

Weiball distribution
employing feedback
and shifting improvements
in calculation of W_K
Kolmogorov - Smirnov test D value
0.05
0.04
0.03
0.02
0.01
.008
.006
.004
0.002
Δx = 1·0
Δx = 1/2
Δx = 1/8
Δx = 1/64
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 m
comparator bits

Figure 3·9d

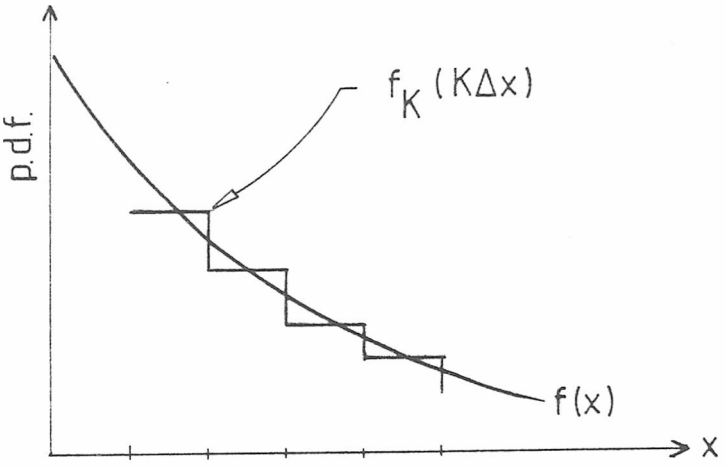


Figure 3·11a

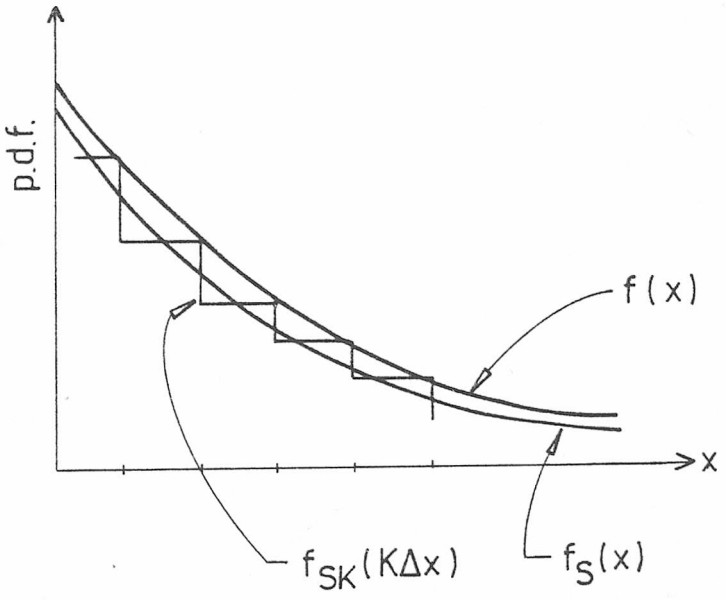
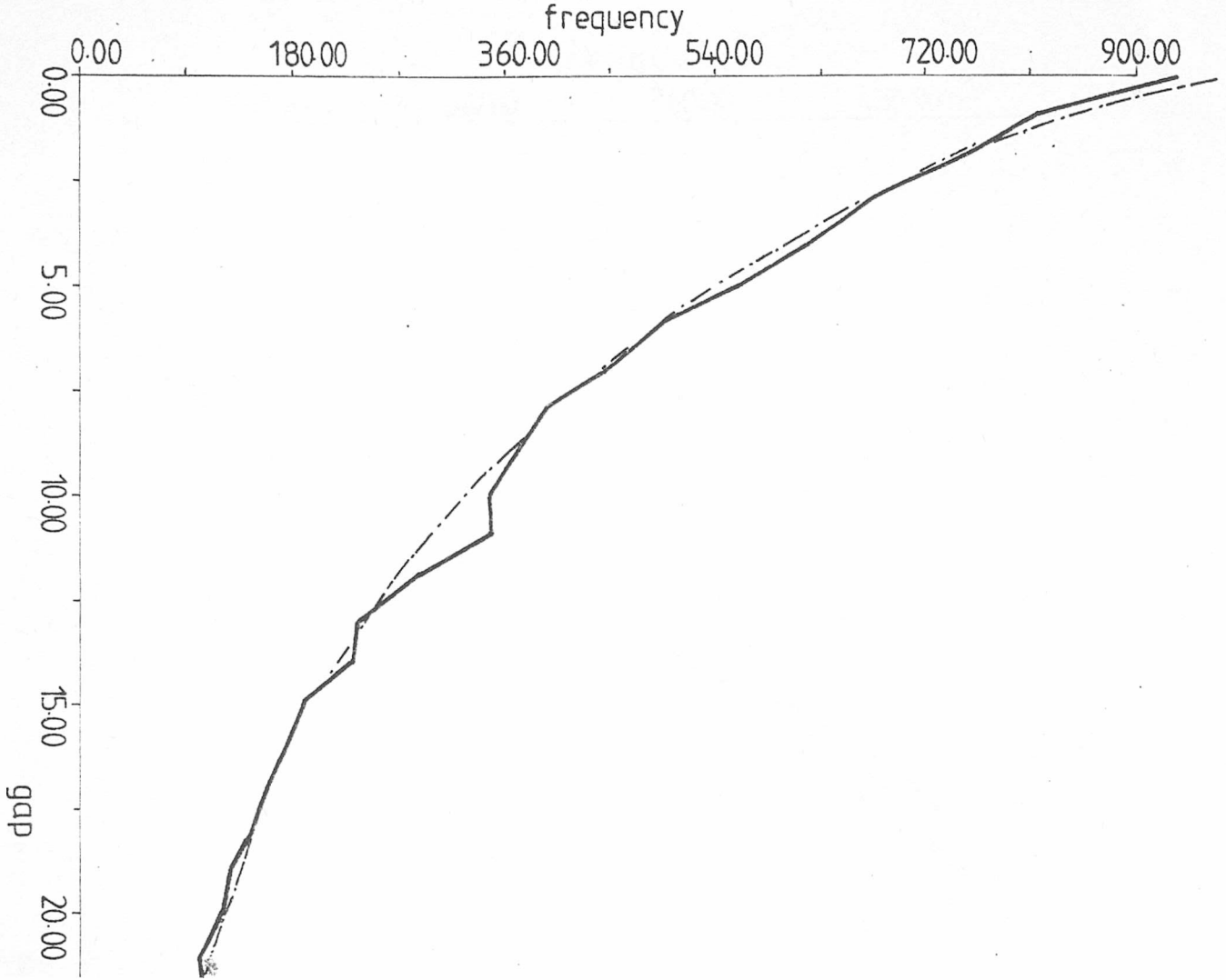


Figure 3·11b

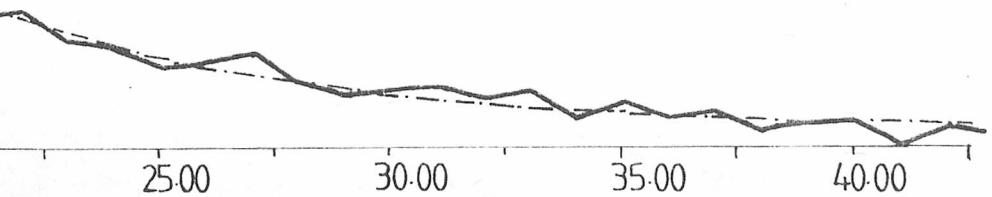
Effects of shifting distribution (T.S.)


frequency
0·00
180·00
360·00
540·00
720·00
900·00
gap
0·00
5·00
10·00
15·00
20·00

Graph No.	3·11c
Description:	Exponential

(data)

sample size =	*9600*
method =	*SIMPLE*
ΔX =	*1·0*
H =	*2*
χ^2 =	*40·0*
D =	*0·0827*



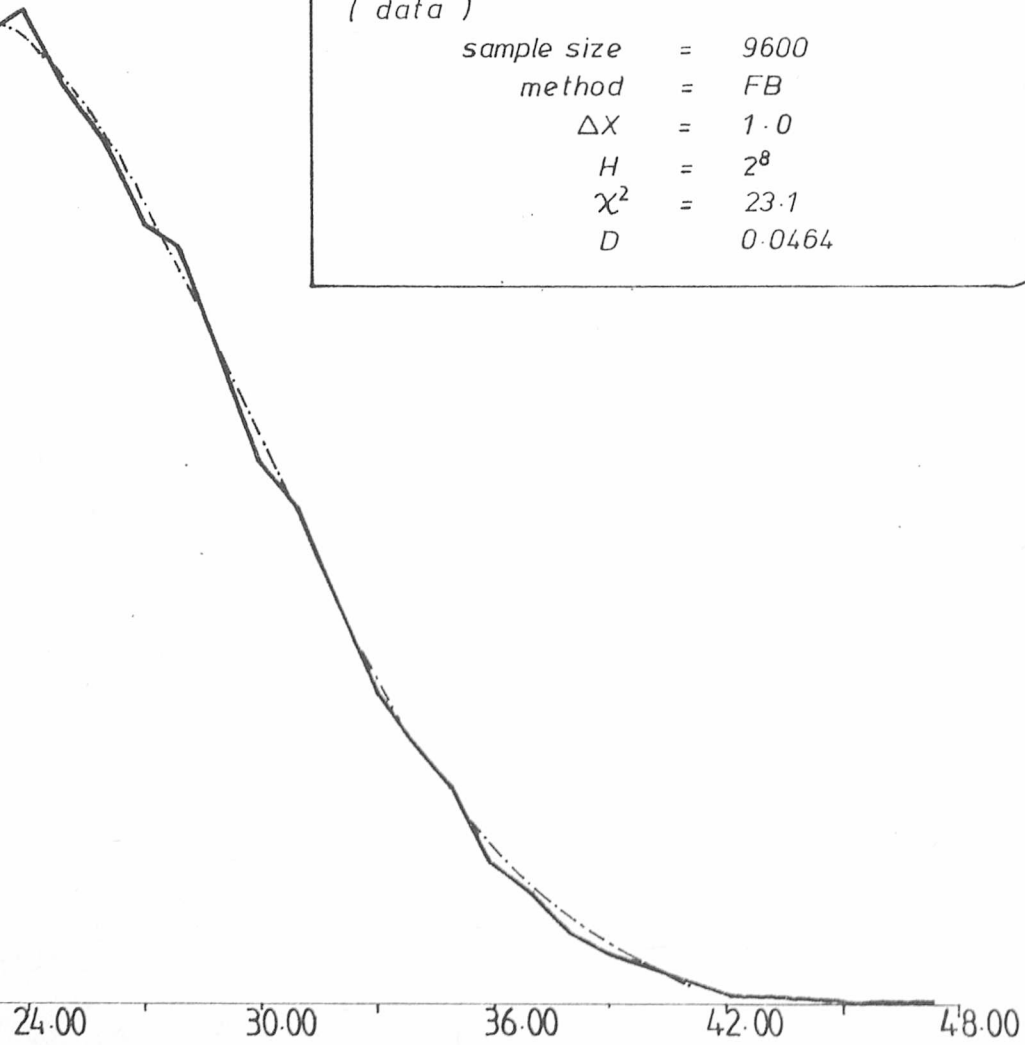
frequency
0.00
120.00
240.00
360.00
480.00
600.00
0.00
6.00
12.00
18.00
gap

Graph No. 3·11d

Description: Weibull

(data)

sample size	=	9600
method	=	FB
ΔX	=	1·0
H	=	2^8
χ^2	=	23·1
D		0·0464



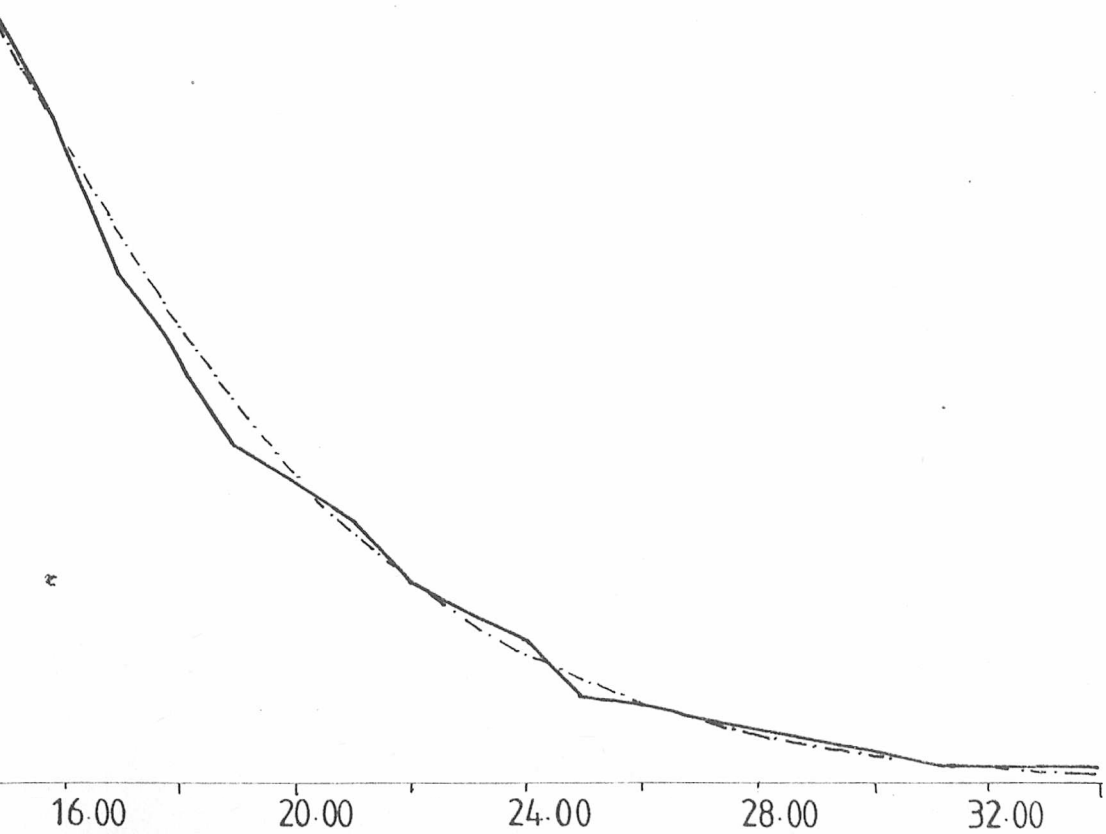
frequency
0.00
150.00
300.00
450.00
600.00
750.00
0.00
4.00
8.00
12.00
gap

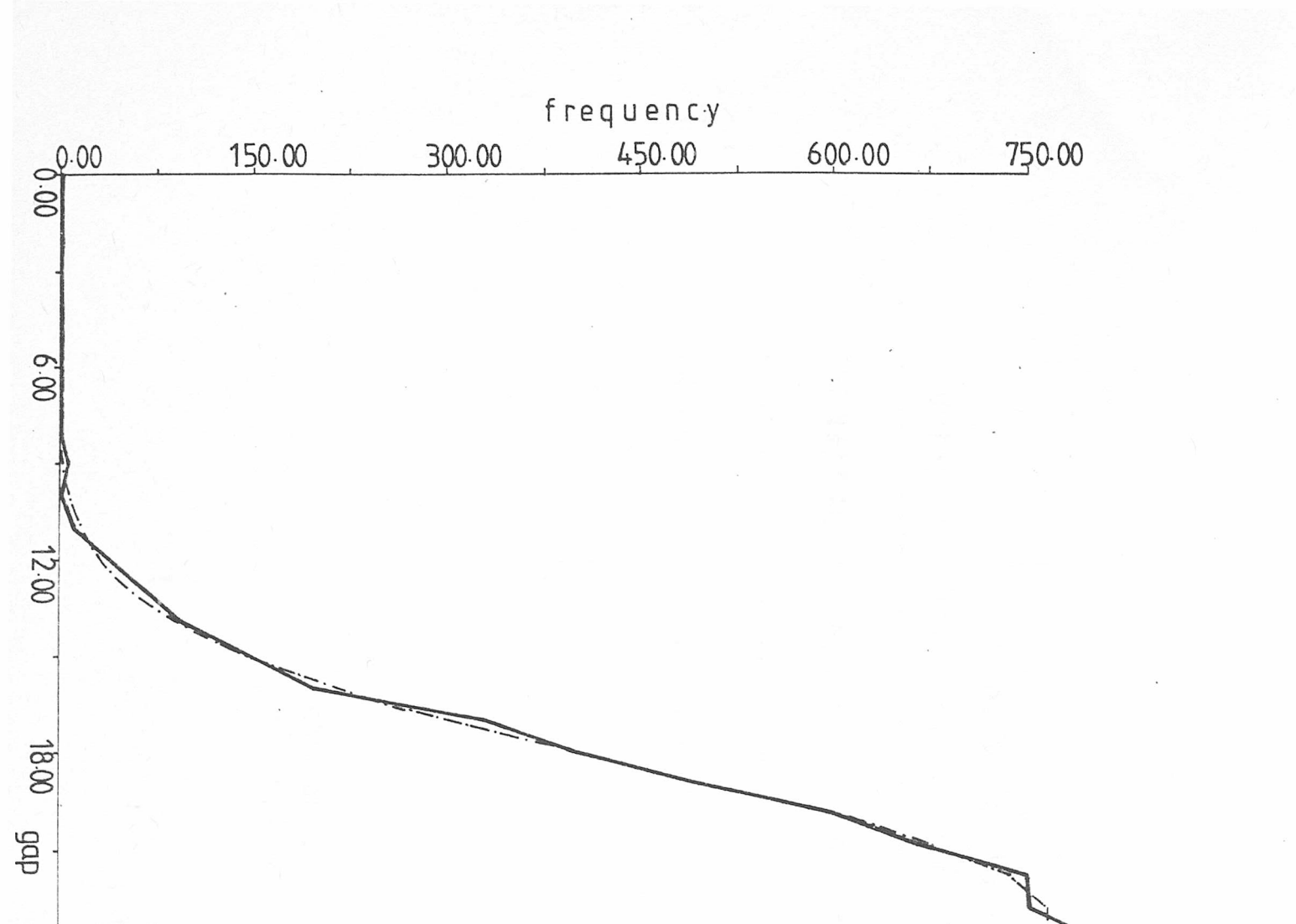
Graph No. 3.11e

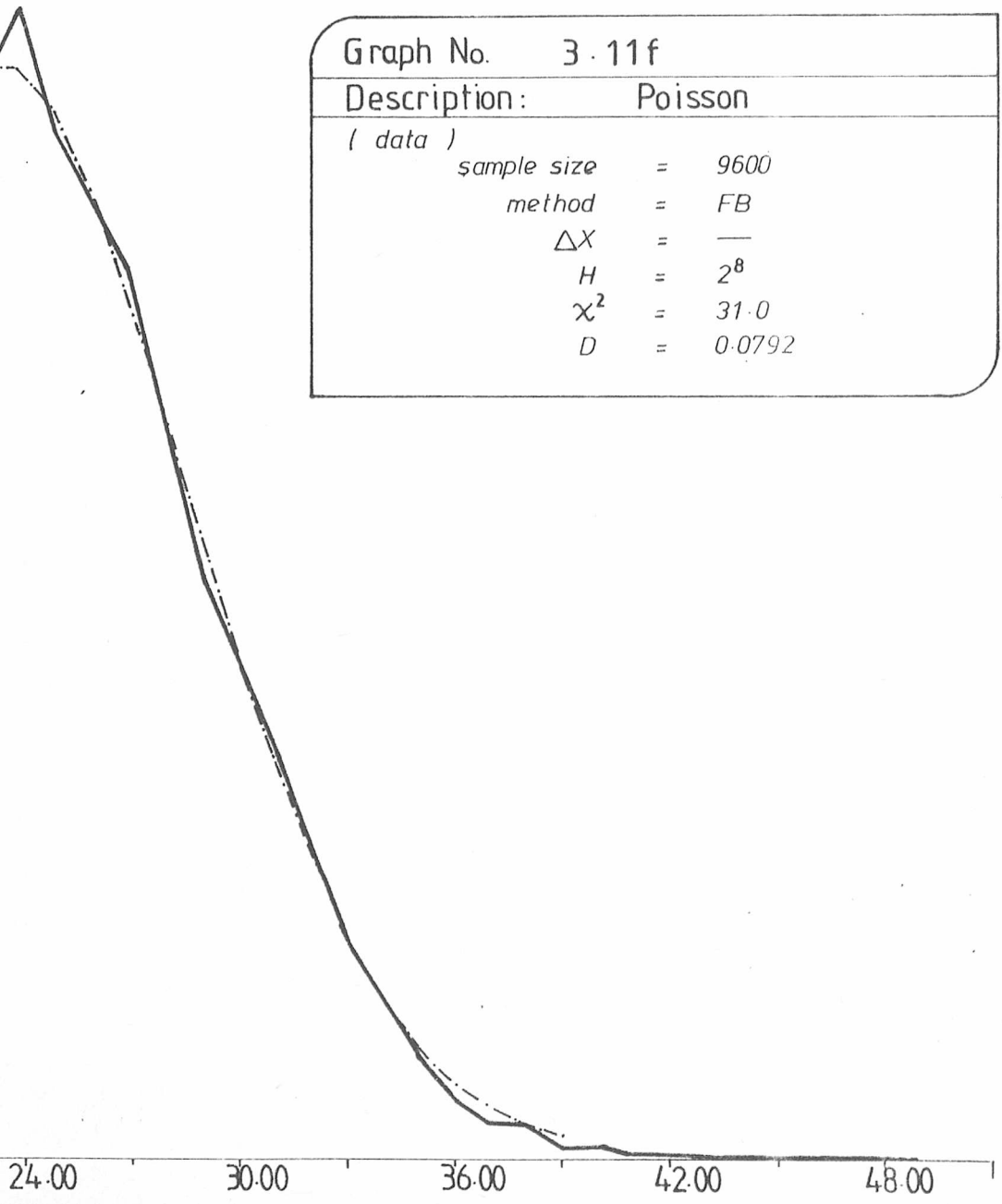
Description: Erlang

(data)

sample size	=	9600
method	=	FB
ΔX	=	1.0
H	=	2^8
χ^2	=	20.4
D	=	0.0745




frequency
0.00
150.00
300.00
450.00
600.00
750.00
0.00
6.00
12.00
18.00
gap


Graph No. 3·11f
Description: Poisson
(data)
sample size = 9600
method = FB
ΔX = —
H = 2^8
χ² = 31·0
D = 0·0792
24·00
30·00
36·00
42·00
48·00

distribution	figure	number of samples	number of bits of random number	Δx	K-S TEST			χ^2 TEST	
					test result	expected value	critical value	test result	critical value
EXPONENTIAL	4.6c	9600	8	1.0	0.0827	0.181	0.0196	40.0	56.8
WEIBULL	4.6d	9600	8	1.0	0.0464	0.0558	0.0196	23.1	52.2
ERLANG	4.6e	9600	8	1.0	0.0745	——	0.0196	20.4	42.6
POISSON	4.6f	9600	8	1.0	0.0792	——	0.0196	31.0	43.8

Table of results for distributions generated with large sample size

Table 3.11g

4.0 Introduction

This Chapter contains an introduction to the philosophy behind the design of the reliability simulator. Consequently it also serves as an examination of the development of an engineering system, comprising digital hardware and software, to implement design ideas. The basic philosophy behind the system design was to break down the tasks of reliability simulation into a number of sub-groups. With simulation aspects decomposed in such a way, there is a need for communication channels between the sub-groups to enable the groups to operate in a unitied fashion. This corresponds to bringing together all the aspects of a model system during simulation. The result of decomposing the simulator into separate sections is:

1. Modular design which lends itself better to system expansion.

2. Simple control of the total system operation due to an efficient data bus structure.

3. Greater efficiency and a reduction in the amount of hardware required in each sub-section.

4. Ease of sub-section testing, and of whole system testing when the sub-sections are brought together.

The attractions of decomposition can only be achieved if the function of each sub-group is carefully chosen. Decisions about these functions greatly influence the form of the interconnecting communication channels. If the features listed above are to be achieved, the interconnection must take the form of a simple, efficient bus structure.

The sub-groups chosen to represent separate aspects of reliability simulation are shown on the block diagram of the simulator, Figure 4.0a, and the bus structure is also indicated. The following sections of this Chapter describe and discuss the subsections and other features which make up the simulator.

4.1 Component

The simulator may contain any number of Component Modules. Each Component module represents a particular aspect of the real system being modelled. In the simplest form a component may represent an item of equipment such as a generator or relay. More abstract system aspects can also be represented in the same way, aspects such as computer software or human operator behaviour. Components interact with each other, and each component has to be versatile if it is to be possible for it to model such a wide range of system aspects. All simulator components acting together describe the characteristic of the real system under investigation.

The behaviour of the system components is governed by both probabilistic and deterministic events. For example an electric motor may have a random time between failures which is exponentially distributed. This means that there are stochastic processes going on at component level and that the processes are expected to satisfy some appropriate probability distribution.

In generating the random times associated with the stochastic processes a supply of random numbers is required. The numbers employed by each component should be of good statistical quality if the model behaviour is to appear truly random. Further the number sequences used by different components are also expected to show statistical independence. These specifications are difficult to meet, particularly in a high speed simulator requiring several sequences. A hardware

implementation could have led to excessive amounts of circuitry, but from investigations carried out, it was found possible to construct number generators for each component using only fourteen medium-scale-integrated (MSI) circuits.

The modelling aspect dealt with by each component is updated in asynchronous time steps. Different components may employ different time step values so as to achieve high speed and accurate modelling resolution. The asynchronous progressions undergone by components are kept relative in time to each other by instructions issued by a Control Module.

Component Modules take the form of special purpose hardware implemented with MSI and large-scale-integrated (LSI) circuits. The interaction between components is dealt with by both hardware and software features. The hardware features are used to deal with the most frequent and simpler actions, and the software features are used to give increased flexibility to the model, allowing it to describe greater, more complex variations of system behaviour. Each Component Module is constructed on a single printed circuit board and is automatically interfaced to the simulator system by insertion into the common bus.

The parallel operation of component hardware ensures a considerable speed of simulation. However, care had to be taken to ensure an economic design, as with a detailed model the hardware required can be excessive.

4.2 Simulation Control

At the start of a simulation the Control Module takes charge of the communication data bus from the host computer. To carry out its operation of controlling an asynchronous time simulation it also monitors a dedicated communications bus, called the Time Increment bus. Each Component Module is connected to the Time Increment bus, and on

request from the Control Module it will reveal it current time quantisation value for the random time process going on at component level. The Control Module determines the minimum time quantisation value currently in use by any component. A decision is then taken about how much the simulation should be incremented by in time, and this value is communicated to the components via the system data bus. Further control signals are then issued to affect the components which have the required time quantisation.

Gathering Statistics on the model's behaviour is also directed by the Control Module. Whilst replying to the components, the Control Module also indicates to the Statistical Gathering Module how much the simulation is being incremented by.

As noted in Section 4.1, some of the aspects modelled by the components are dealt with by software. Components, on occasion, make requests to this software which exists in the form of routines held in the host computer. The Control Module is involved in monitoring these requests and directing them towards the host.

The Control Module is constructed on a single printed circuit board from MSI and LSI integrated circuits.

4.3 Statistical Gathering and Network Specification

The function of the Statistical Gathering Module is to gather evidence about the model system behaviour. It consists of a number of binary counters. Each counter within the Statistical Gathering Unit is programmed, by the simulator operator, to gather evidence about some particular behaviour of the system. Programming of the counters is achieved by defining logical events about which condition monitoring is to take place.

Each component produces a binary output signal indicating that it is in an operating or non-operating state. A Boolean expression of combinations of component states which describe system events and sub-events can be developed. All logical event expressions are stored in the Network Specification Module, which is programmed before a simulation run starts. The Simulator is then in a position to decode the binary component signals and indicate to the Statistical Gathering Module the occurrence of a system event, thus enabling monitoring to take place. Clearly a Boolean function is required for each Statistical Gathering Unit counter. By making the Network Specification Module fully programmable, via the common bus, any interconnection of model components which describes a system under investigation can be entered into the Simulator. Further details of the techniques used to specify the system topology can be found in Chapter 5

4.4 Repair and Maintenance Policy

The Repair and Maintenance Policy Module is continuously informed of the condition of each component making up the model system. Should any component have to make a decision about its future behaviour, it is required to inform the Policy Module. It should be noted that only decisions which influence the overall state of the system, or other components in the system, are dealt with in this way. The Policy Module has been initially programmed with the system management scheme at the start of the simulation and therefore can make decisions effecting the global system. For example, consider a component representing an item of equipment which is due for maintenance. A maintenance request is issued to the Policy Module which decides if there are maintenance facilities currently available, or which could be released from a less important task. This request must be considered. Further the Policy Module may decide that the system is in a critical state and maintenance

should be queued for future consideration. Finally the Module informs each component of its decision.

The Policy Module takes the form of a large lookup table [39] which is addressed by component states and requests. The table contents are presented to the decision lines returning to the components.

4.5 Host Computer

The host computer has already been referred to in the Component Module section. It was stated that certain aspects of the component representation are dealt with in software. This is made possible by the host computer having access to the common bus which can be used to effect changes in the simulator hardware.

The ability to manipulate hardware is extended to allow system initialisation by the host. Examination of the Simulator's progress is also possible as the host can request the contents of any register or counter accessible via the common bus. These are important points and were carefully considered during the design of the separate modules.

The programs contained in the host computer play an important role in the overall simulator system. With the ability to set up the Simulator's hardware, and store the configuration for future use, repeated simulation of a particular problem is simply achieved. An analysis of statistical data, gathered by the hardware during simulation, is carried out by programs resident in the host computer.

To summarise, considerable flexibility is offered by the Simulator. Operator control of the modelling process is by a visual display unit, communicating directly to the host computer. Because the operator communicates with the Simulator via host software the system can be made much more user oriented and tedious control of hardware is eliminated. It is anticipated that the Simulator's hardware will be more fully exploited by such a system.

4.6 Host Computer and Simulator Interface

The interface is based on an 8085 microprocessor and support chips. The layout is indicated on figure 4.6a. There are two modes of operation for the interface. In the first the VDU is connected to the host computer, giving normal peripheral terminal operation. The second mode of operation enables interaction of the simulator hardware with host software to take place.

The necessary protocol between the Simulator and host is dealt with by the interface. Effectively the host behaves like a Simulator Module tied to the system common bus. In a similar way to the Control Module the host can take control of the bus and both enter data and extract data from it.

The circuitry for the interface is constructed on a single board which has all necessary input/output connections for insertion into the common bus.

4.7 System Bus Organisation

The main communication path within the simulator is the common bus. It comprises a 12-bit address/control portion and a 12-bit bi-directional data portion. On initialisation of the Simulator hardware, the bus is controlled by the host computer via the interface. During a simulation, the bus is controlled by the simulator Control Module. However the Control Module may pass control of the bus to the host computer. Bus contention is prevented by a hand-shaking arrangement between the interface and the Control Module.

An important bus for system operation is the Time Increment bus. It communicates to the Control Module the time quantisation presently used by each component. For reasons which will become clear later, high speed operation can be more easily achieved if it is implemented in open collector form. A 12-bit bus is used.

4.8 Conclusion

In conclusion, the Simulator described here in offers a powerful tool in determining system reliability. Detailed models can be quickly constructed to simulate a real system with considerable speed. The unrestricted modelling ability makes this Simulator particularly useful in varied modelling experiments.

The Simulator operator would find the model very convenient to deal with. All control and setting up is dealt with by programs in the host computer.

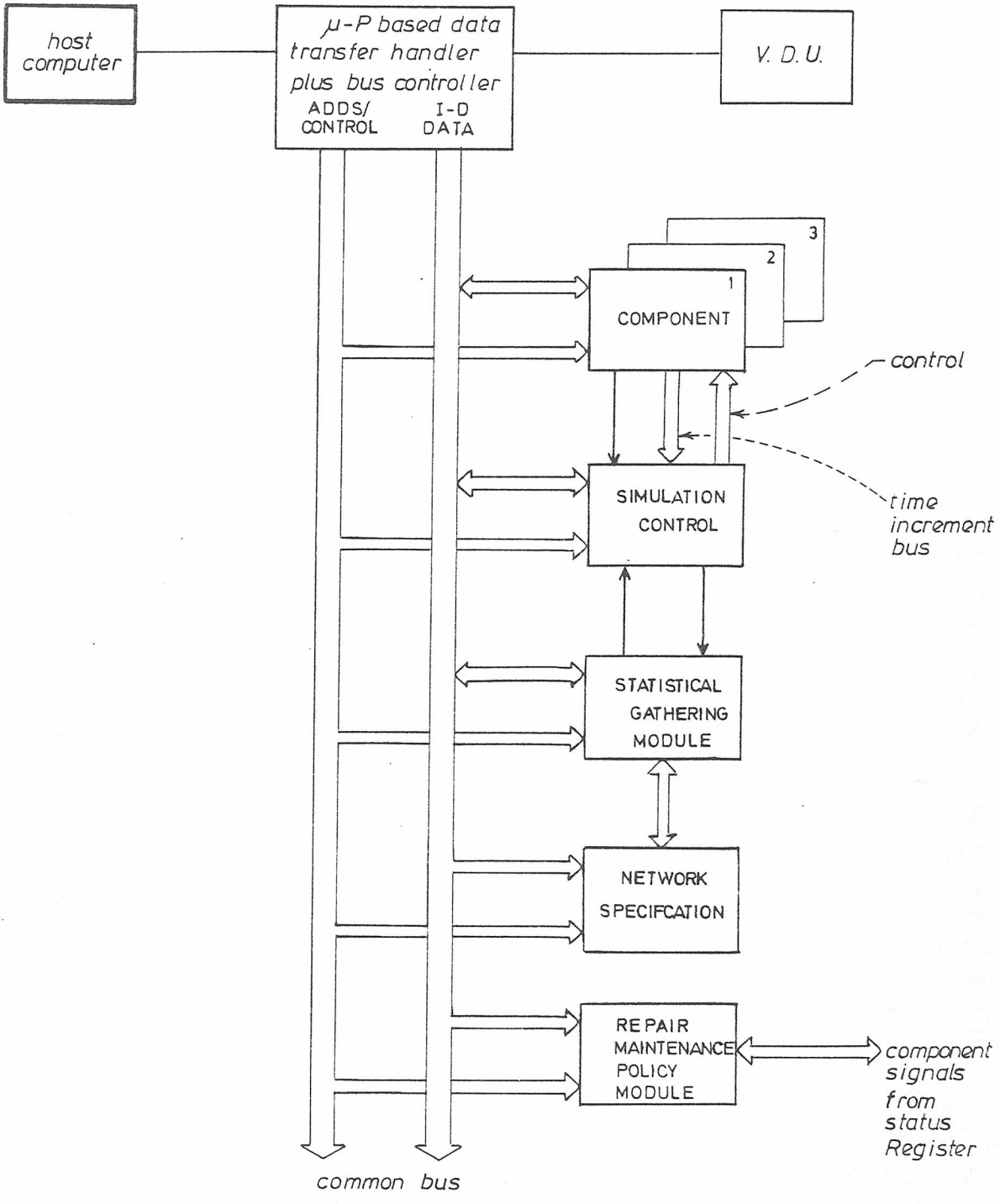


Figure 4.0a Simulator block diagram

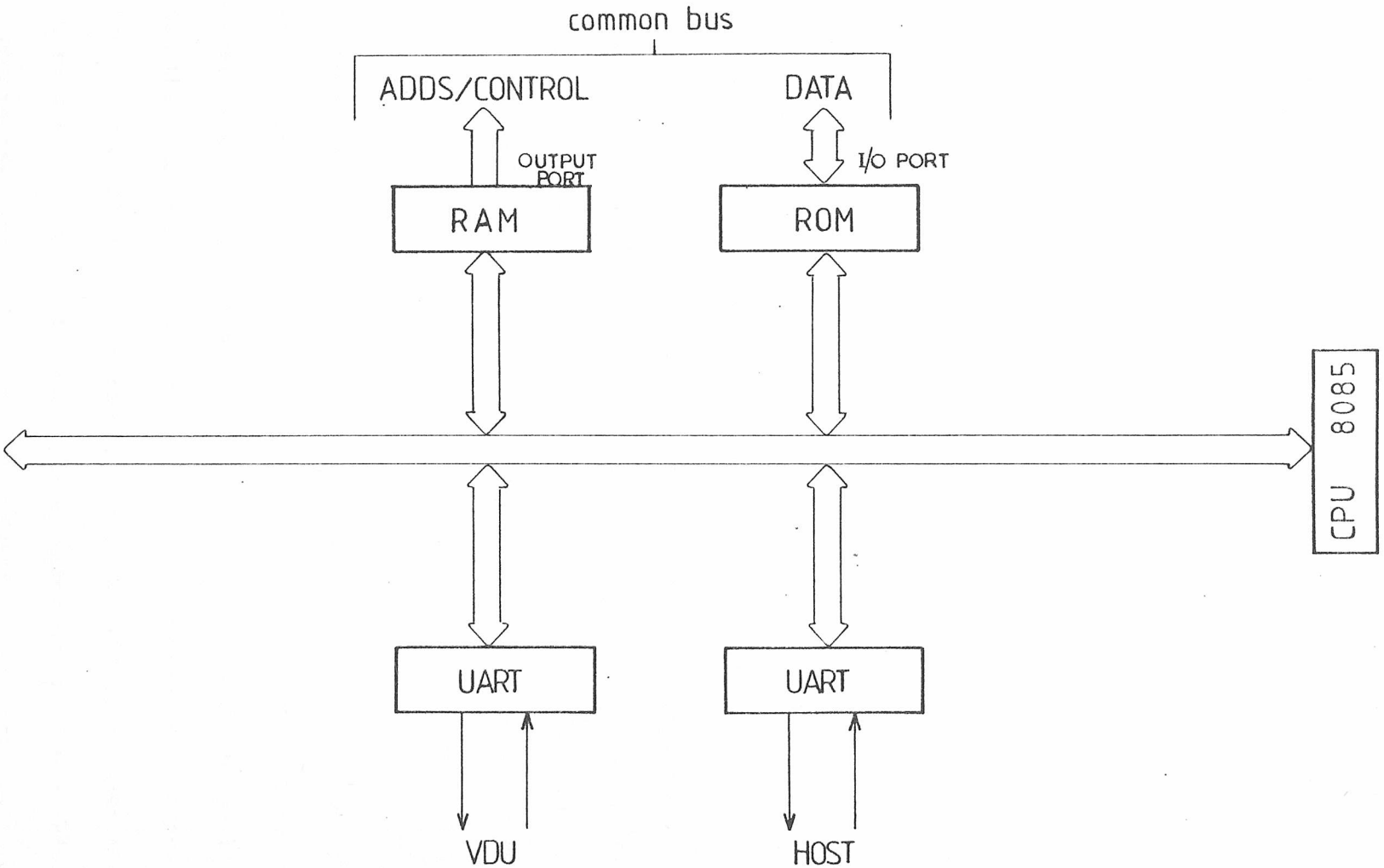


Figure 4.6a Host computer and simulator interface

5.0 Introduction

The Simulator hardware is totally implemented in digital circuitry. This necessitates quantisation of all time and amplitude values in the model. The effects of quantisation have been investigated and are reported in Chapter 3. From the results obtained, it was decided that a 12-bit data bus structure would be most suitable for hardware interconnection. With values represented in twelve binary bits, there are 2048 resolved levels. This is shown later to give an adequate range of time quantisation, and also a sufficiently small amplitude quantisation. An increase to more than 12-bits would give only small returns in modelling accuracy but would result in a considerable increase in hardware.

In Chapter 3 the process of generating random time intervals was introduced which in reliability modelling may correspond to life times of equipment. In essence, a random time value T_n can be generated in which T_n is a multiple of a quantum value ΔT. The minimum time value which T_n can have determines the resolution of the distribution of possible T_n values. If ΔT is chosen to be large, the generated distribution of T_n would not model the desired distribution. Further, the model becomes unrealistic in that essential data may be lost during a simulation. Selecting a small ΔT value would solve these problems. However when considering the problem of reliability modelling it is found that the distribution being used for modelling depends on the state of the model. For example, repair distributions have a different form from failure distributions. Time quantisation values which would be considered small for one distribution may not be for another. Running the whole simulation at the smallest ΔT required to model any of the random time values would introduce unnecessary detail to the

simulation. Also, the time required to carry out a simulation would be much greater than if the optimum ΔT value were being used at all times. It would be also impossible to operate the system with a 12-bit bus structure, as with very fine time resolution, an increase in amplitude resolution is required to maintain modelling accuracy. Clearly there is a need for a mechanism which can operate with the optimum ΔT value. The technique described in this chapter offers a system which not only achieves this but maintains several random processes all of different ΔT values in a parallel mode. This makes possible a simulation possessing both considerable speed and resolution.

This chapter explains the operation of minimum ΔT value control. It also describes in detail the hardware structure of the Component Modules,and other simulator modules, which play an important part in giving the Simulator its versatility.

5.1 ΔT Selection

In Chapter 4 it was reported that each Component Module takes on the job of modelling a particular aspect of the simulated system. Therefore a process is going on within each component for which a random time interval is being generated. All time values are quantised but the actual ΔT values used by components may be different.

Consider a simulator system in which each component is permitted to model its particular probability distribution with a time quantisation value of suitable resolution. There are three points to be noted here.

1. Investigations have shown that the range of ΔT values anticipated could all be represented within a 12-bit binary pattern. Also the amplitude resolution required for such ΔT values could be achieved with a 12-bit structure.

2. All time quantisation values must be integer multiples of an absolute minimum quantisation value, ΔT_{min}. This must be true if individually modelled aspects are to be kept relative in time to each other.

3. The update of the modelling process corresponds to incrementing the renewal process. Components would only be considered for update when the basic counter keeping track of simulated time had been incremented by an amount equal to their ΔT value. For example, if a component had time quantisation $\Delta 3T_{min}$ and another had $\Delta 7T_{min}$, then the first component would have its modelling process updated more than two times for every one update of the second component.

For the simulator considered, the job of controlling simulated time would involve determining the amount to increment the basic counter by to reach the next component update. The drawbacks of the system would be :

1. Determination of the basic counter increment could not be achieved at high speed for a multi-component system.

2. A situation could be reached in which all components currently had the same time quantisation but were out of phase. This would mean that the ability to rapidly increment the basic time counter would be lost, as component modelling would be taking place in a serial fashion although a parallel mode was possible.

To overcome the problems described in this section the constructed Simulator limits the range of selected ΔT values to binary multiples of ΔT_{min}. This makes possible a system which can rapidly determine the minimum time quantisation present among components. Secondly, all

components which currently have the same ΔT value, are updated together and never allowed to fall out of phase. Considerable speed of simulation can be expected.

5.2 High Speed ΔT Selection

Limiting time quantisation values to binary multiples of a basic minimum value, ΔT_{min}, enables a 'wired - OR' function to calculate the minimum ΔT value in use by any component. Figure 5.2a outlines the operation. Each component simultaneously presents its current ΔT value to the open-collector bus, known as the Time Increment Bus. At the Control Module, a wired-or function is performed on the bus and the minimum time value is found. This value is normally the amount the basic counter is incremented. The only diversion from this rule occurs when there is a change in minimum ΔT. In this case the simulation is incremented by the time required to reach the next scheduled update of whichever components have the new minimum ΔT. By this method a coherent simulation is maintained. A result of achieving in-phase simulation is that components experience an initial phase shift, to bring them in line with all other components of the same ΔT value. Shifting obviously only occurs when a component changes its ΔT value and therefore is relatively infrequent. If small ΔT values are used, the error introduced to the model should be negligible.

As an example to simulator operation, consider a system of components for which simulation has been progressing with $\Delta T = \Delta 2T_{min}$. The time counter has reached $\Delta 6T_{min}$ and it is found that the minimum ΔT is $\Delta 4T_{min}$. Figure 5.2b describes the process. It can be seen the simulation time counter should be incremented by $\Delta 2T_{min}$ to reach the next scheduled update. At that point, components of ΔT values $\Delta 4T_{min}$ and $\Delta 8T_{min}$ are updated. The Control Module performs the task described here. The hardware layout of the ΔT selector is shown on

Figure 5.2c.

To summarise, the operation of the ΔT selector portion of the Control Module:

1. components are chosen for update depending on their ΔT value
2. components of ΔT value ΔT_{min} are chosen twice as often as those of $\Delta 2T_{min}$, and four times as often as those of $\Delta 4T_{min}$ and so on.
3. By using the common bus, the controller indicates which components are to be updated.

5.3 Component Structure

The hardware structure of components is a limiting factor to Simulator's flexibility. Considerable effort has been directed towards obtaining an economic design without making major compromises in speed and flexibility. The final component structure presented here reveals a system of surprising flexibility with a high speed, parallel mode of operation. Flexibility is achieved by incorporating a micro-instruction controller which can be programmed by the Simulator operator. High speed is achieved by parallel hardware operation and careful design. The parallel philosophy to component operation ruled out a μ-processer implementation. A Simulator in which each component is modelled by a single μ-processor is certainly achievable but would result in simulation times possibly two orders of magnitudes longer.

A diagram of the component layout is given on Figure 5.3a. The subsections which follow deal with the structure in more detail.

5.3.1 Status Register

The status register ,Figure 5.3.1a, is a 7-bit register which defines the overall condition of the component at any time. The first 3-bits define eight possible component states. The remaining 4-bits are flags set during the modelling process.

The status bits of the register directly address the time quantisation memory. Therefore, on a request from the Control Module, the memory can transmit to the Time Increment bus, via the component bus, the time quantisation assigned to the current component state. The time quantisation memory described above also contains eight hazard rate values, one for each component state. These values can be presented to a comparator, along with a random number, via the component bus. This makes possible the generation of monotonically distributed random time values for each component state.

The flag bits are used to keep a record of the simulation process. They communicate to the Policy Module outwith the component, assisting it with global system decision-making. They also communicate with the micro-instruction controller contained within the component, enabling decisions to be taken about component modelling behaviour. Bit 7 in the register has the special function of interrupting the simulation Control Module with a request for software modelling features contained in the host computer.

5.3.2 Mask and Micro-instruction Control

Due to the parallel operation of components, to make possible high speed simulation, a considerable amount of decision making has been devolved to the components themselves. The decision-making policy is contained in a single 1024x8 bit EPROM which issues micro-instructions to hardware within the component. At all times the 'micro-controller' communicates with the status register, and is informed of global policy,

decisions. Clearly with this system, modelling aspects are contained within the micro-controller memory, and if an alteration to the model is required, then the memory must be re-programmed. A system operator would find this very inconvenient as each component contains a micro-instruction memory, and any number of these may require alteration. Also determination of the micro-instruction code is a time-consuming task. To overcome this problem, and maintain the desire to have a highly flexible simulator, an 8-bit mask register has been introduced.

The first three bits of the mask connect directly to the micro-controller, and the remaining five bits are used to operate on information passing to and from the controller. This makes possible the use of a small micro-controller memory. What is achieved is a micro-controller which contains the instructions to model a wide range of behaviour without the need for re-programming. The operator has now merely to define the mask register setting to select a particular model.

5.3.3 Component Counters

Each component contains three hardware counters. These counters are indicated on the component diagram, Figure 5.3a and also Figure 5.3.3a, as C_a, C_b and C_c.

Counters C_a, C_b are 12-bit programmable counters. Their contents are fed directly to digital comparators for comparison with values T_a, T_b respectively. When $C_a \geq T_a$ or $C_b = T_b$ occurs, a signal is sent to the micro-controller which takes appropriate action. The micro-controller has the ability to increment and clear C_a but only increment C_b. Counter C_b is automatically cleared when a match with T_b occurs. With these counters, control signals (which effect the modelling behaviour) can be programmed to be issued after fixed periods of time or prescribed number of events.

Value T_b may correspond to a 'block replacement' time. Under a block replacement policy [35] the component is replaced at the end of regular intervals of time, regardless of its operation during the time interval. By employing counter C_a to record operational time an 'age replacement' policy [35] can be implemented. The component is then replaced when it completes a working time T_a.

Counter C_c is intended for statistical gathering use. It is an 8-bit counter which can be incremented by the micro-controller. This permits a record to be kept of the number of occurrences of a particular event, such as visits to a particular component state. Use of counter C_c is essential for monitoring system behaviour not observable via the logical network (see Section 8.7 on Optimal Economic System Operation). The decomposed statistical gathering system formed by C_c counters and the Statistical Gathering Module enable information to be gathered about a systems performance in a way not possible by logical networks alone.

5.3.4 Non-Monotonic Distributions

Each component has the capability of modelling two non-monotonic renewal processes. It is known that during the generation of random time intervals which are not exponentially distributed, the hazard rate for the corresponding process is not constant, but changes with time. To enable high speed generation of such processes, the hazard rate must be calculated at regular intervals and the values obtained stored in a memory for use during the simulation.

For this purpose, a 256x8 bit read-write memory is provided on each component, see Figure 5.3.4a. The first seven bits of counter C_a are used as the address, with the eighth address bit giving selection between two stored distributions. With this organisation counter C_a has the special function of recording the age of the process. The selection of non-monotonic distribution memory in place of monotonic

distribution memory is carried out by the micro-controller. The micro-controller can also, if instructed, join the two distributions together forming a single 256 interval process. In this case the eighth bit of C_a is used in place of the distribution select line.

It should be noted that non-monotonic distributions are generated with an amplitude resolution of eight bit instead of the normal twelve bits. This does not lead to a poorer standard of modelling as investigations, reported in Chapter 3, have revealed techniques of calculating hazard rates which lead to increased modelling accuracy. These techniques make possible the use of an eight bit structure, which limits the hardware required.

5.3.5 Control of Component

Component control is devolved in two ways. Firstly during the simulation, the process undergone by each component is directed by its own micro-controller. Each of these micro-controllers is governed by the Control Module which issues a limited number of instructions causing the whole simulation to progress. A 3-bit control bus is used.

The second form of control is by the address/control bus section of the common bus. Instructions issued by this bus would normally be accompanied by a value on the 12-bit data bus portion. Components decode the bus, determining if the information is for them, and further where it is to be sent. A table of programming codes is given on Figure 5.3.5a. Each programming instruction is constructed from two octal characters. In this way micro-instructions can be issued by combining codes. An example of micro-programming can be seen when entering data into the 256x8-bit, read/write memory. Both the 'enable-memory' and 'write-to-memory' codes are required. The resulting instruction code is 44 octal.

5.4 Control Module

The Control Module layout is given on Figure 5.4a. Its basic function can be easily described. Firstly, a signal is issued to all components to transmit their current time quantisation value via the Time Increment Bus. After examining the values received, the Control Module determines the updating quantisation value. All components are simultaneously informed of the updating value via the common bus. At this point a second control instruction is issued enabling the micro-controller of components which have the correct current time quantisation. Finally a third control signal is issued allowing the micro-controllers to update the component status registers. It can be seen because of the efficient bus structure that the amount of controlling signals required by components is greatly limited.

As well as carrying out the basic function described, the Control Module has two other functions. It must generate the three-phase clocks used to record simulated time. The clocking system is fully described in the following section on statistical gathering. The second function is to stop the simulation at the required point. This is normally when the digital comparator connected to the 24-bit simulation counter identifies a match with the pre-programmed stop time. However the simulation could also be stopped by the simulator operator, host computer, Statistical Gathering Module or any of the Component Modules by an interrupt request. Whenever the simulation is required to stop, the control procedure is the same. That is, control signals are no longer issued after the completion of the current time quantisation update. This ensures that the statistical gathering is correctly accomplished.

The Master Control Unit, shown on the module layout, makes up only a small proportion of the Control Module hardware, but is the key to the overall operation. It consists of a 24-bit recirculating shift register which issues all the necessary control signals as a single bit circulates around the register. The register is clocked at 10MHz leading to an updating of the simulation model every 2.4 microseconds.

Control of the Module itself is via the common bus in the same way as any of the simulator modules (This procedure has been described in Control of Component section). The Control Codes are given on Figure 5.4b.

5.5 Statistical Gathering and Network Specification Modules

Statistical evidence gathered about the model performance is the end result of any simulation. Throughout the simulation, data is gathered at the occurrence of prescribed logical events in counters constructed on the Statistical Gathering Module. A module layout is shown in Figure 5.5a. There are twelve twelve-stage counters arranged in six groups. Each Statistical Gathering Module provides the Simulator with four data gathering probes. A probe is an arrangement of counters given over to gathering evidence about a single logical event. Each probe contains a 24-bit counter recording simulated time, and a 12-bit counter recording the number of occurrences of a logical event.

The operation of the Statistical Gathering Module is best described by an example. Figure 5.5b shows a system flow diagram containing five components. The corresponding success tree is shown in Figure 5.5c. Success trees are a means of graphically displaying the Boolean functions which describe logical events. It can be seen that probe P1 is connected to the final system output (the top of tree) representing a system failure event. Probe P2 is observing the condition of the two parallel signals corresponding to the logical sub event P2=(1+2)(3+4).

Whenever a probe detects a logical 0 it increments its occurrence counter by one and enables incrementation of its time counter by the amount the simulation is progressing until a logic 1 appears. At any time during the experiment, and certainly at the end, the host computer may update its knowledge of the model behaviour by examining the probe counter values.

The binary signal produced by each component, indicating an operational or non-operational state, form the input to the look-up table known as the Network Specification Module . The table output represents enabling signals to the Statistical Gathering probes. The Network Specification Module is constructed on a small printed circuit board which is inserted on to the common bus. Figure 5.5d shows the layout. Its operation is simply achieved by a RAM table, where the mapping between input and output is according to Boolean functions defining logical events. That is, each location in memory contains a binary word specifiying system top event and sub-event occurrence. With a large system containing many components this table may be very large and a RAM decomposition technique employed. This can be achieved by limiting components to groups which form the input to a sub-table. The logical sub-event signals produced by sub-tables would then form the input to a table monitoring the top event. No difficulty should exist in describing the top event in terms of sub-events, as sub-event probe points represent intermediate terms for the top event Boolean expression.

By employing a programmable Network Specification Module, the operator is freed from the tedious task of connecting components with AND and OR gate logic to describe each system. The operator need only define the Boolean expression of the top event at the VDU terminal, allowing the host computer to determine the contents and then program the Network Specification Module table [40]. Any system configuration can

be stored for re-use or modified during a simulation with this highly flexible system. Further development has made possible a graphical input of the model system success tree, eliminating the need to determine the Boolean expressions for system events.

Statistical probes are so called because they can be moved around the system tree, allowing a clearer picture to be formed of sub-events which lead to the top event. It should be noted that there is no need for all probes to be examining the same tree. Alternative system organisations leading to different trees, can be simultaneously observed by simply defining different system event expressions.

The size of the binary words contained in the RAM table determines the maximum number of statistical probes available for gathering results. With an N-bit word N-1 sub-events and the top event can be monitored for each combination of component state signals.

Probe number P1 has a special feature in that it can be used to stop the simulation. If hardware in the Control Module has been correctly prepared before the simulation starts (via host software) then a stop request is issued at the occurrence of the first logical event at probe number 1. This makes possible such experiments as an investigation of 'time to first system failure'.

The minimum value which any counter recording the simulated time can be incremented by is ΔT_{min}, corresponding to a single input clock pulse. When these counters are incremented they may be so by any number of clock pulses, within the limitation of the maximum time quantisation value of 2048. It would be impossible to clock the statistical counters up to 2048 times during the 2.4μ seconds cycle. A solution to the problem would have been to replace counters by adders which could have summed to stored values, the current time increments. This would have considerably increased the amount of circuitry required leading to a greater cost and a longer development time. The solution adopted was to

use a three phase clocking system shown on Figure 5.5e. All time increments are constructed from combinations of the three clocks viz. X1, X16, X256. For example, an increase of 38 time units is achieved by the combination 6x1+2x16. Care has to be taken to ensure that carries from early portions of counters are rippled to the inputs of succeeding sections. No more than sixteen clock pulses are ever required to increment the Statistical Gathering units. This can be achieved within the 2.4μ seconds cycle time.

5.6 Repair Policy Module

The repair Policy Module is in charge of all system repair and maintenance resources. Continually, components make demands upon these common resources. The Policy Module monitors all component requests and distributes the resources according to the pre-programmed management policy. A decision-table implementation of the module has considerable advantages. It offers a method of clearly defining complex conditions for which decisions must be taken.

The behaviour of components is interrelated, with respect to global resources and policy. The complexity of these relations does not affect the speed of the simulation, which is only concerned with the access time of the table. The table lists all combinations of component states, and for each combination provides the corresponding management action. Programming the table before a simulation starts requires a clear description of the rules governing any action. However once this has been carried out, modelling of complex systems is achieved with ease and confidence.

Consistent with the nature of the hardware simulator described in this report, the operation of the Policy Module is in parallel with Component Modules and the Statistical Gathering Module. It is expected that, for systems containing a large number of Component Modules, table decomposition will be necessary, particularly where components may be in more than two states. An increase in the number of possible component states permits more accurate modelling of management policy but would result in a considerable reduction in the number of components observed by each sub-divided table. The Policy Module could no longer be called global and would be simply a collection of localised decision-making teams. Fortunately, smaller tables should be more easily developed and quicker programmed.

Stochastically and deterministically varying management policies can be achieved by making the table contents a function of particular components which need not appear in the system tree. These special components may represent changes (not failure) of repair crew staffing or different policies operative only in crises. The module is constructed in a similar way to the Network Specification Module, employing a read-write memory.

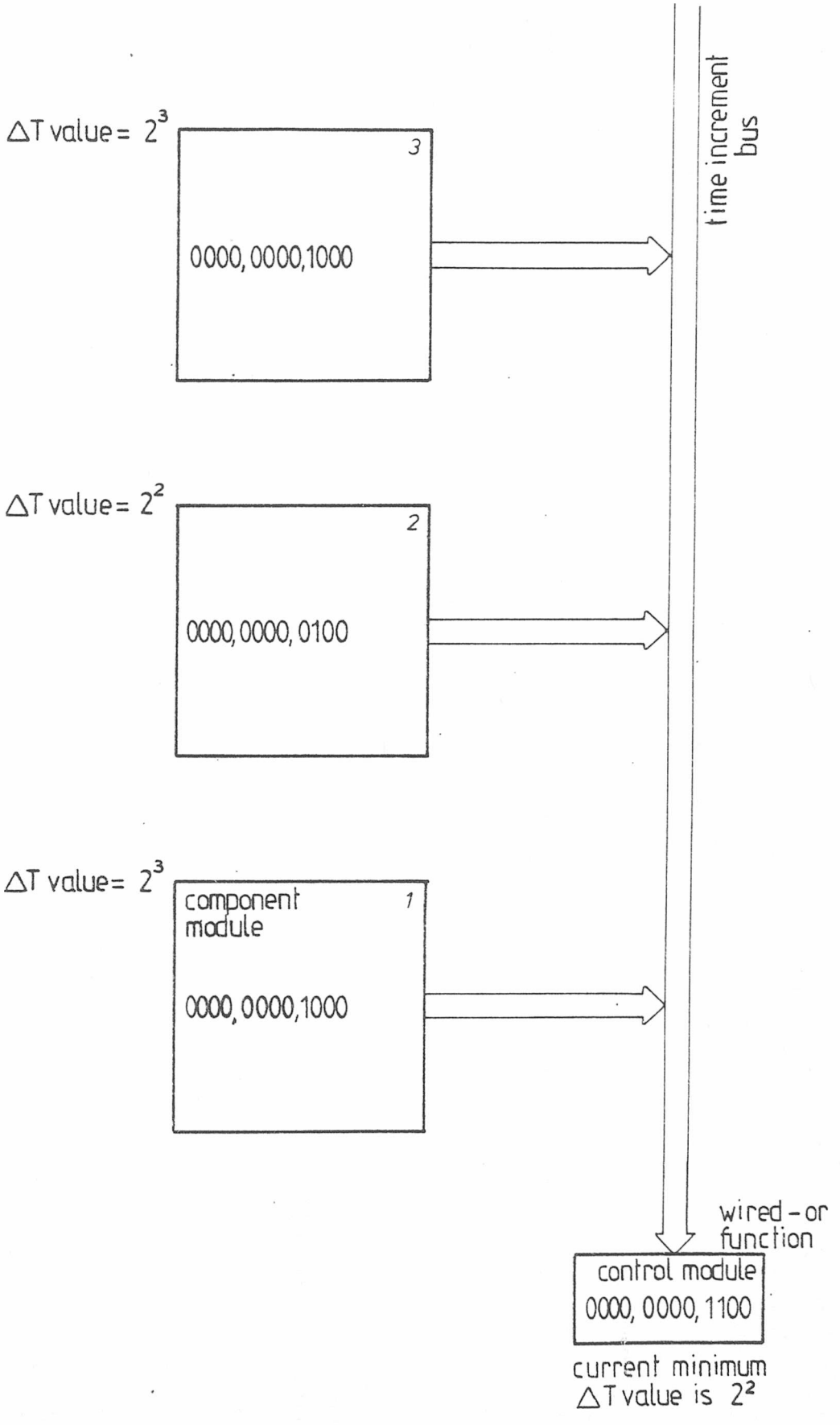


Figure 5.2a ΔT selection portion of control module

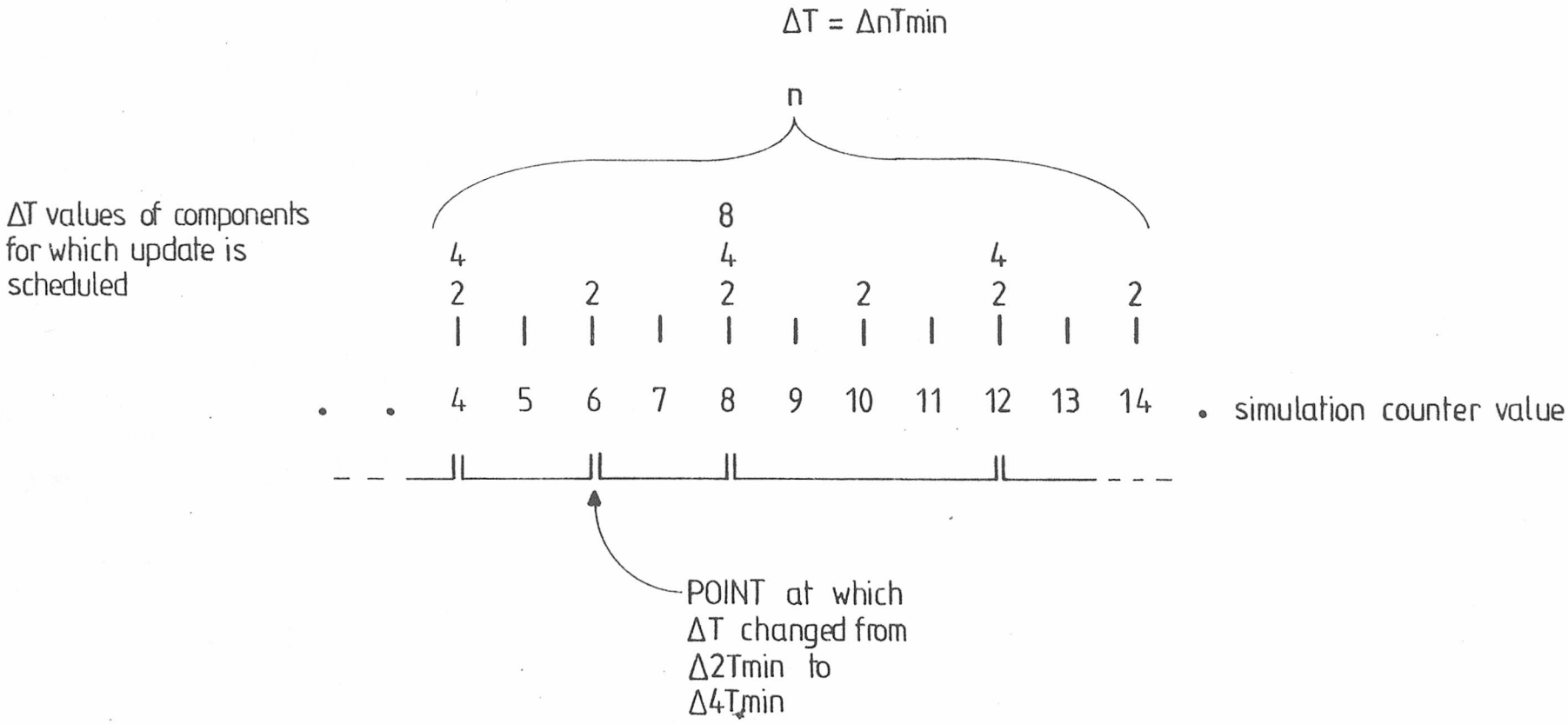


Figure 5·2b Example process

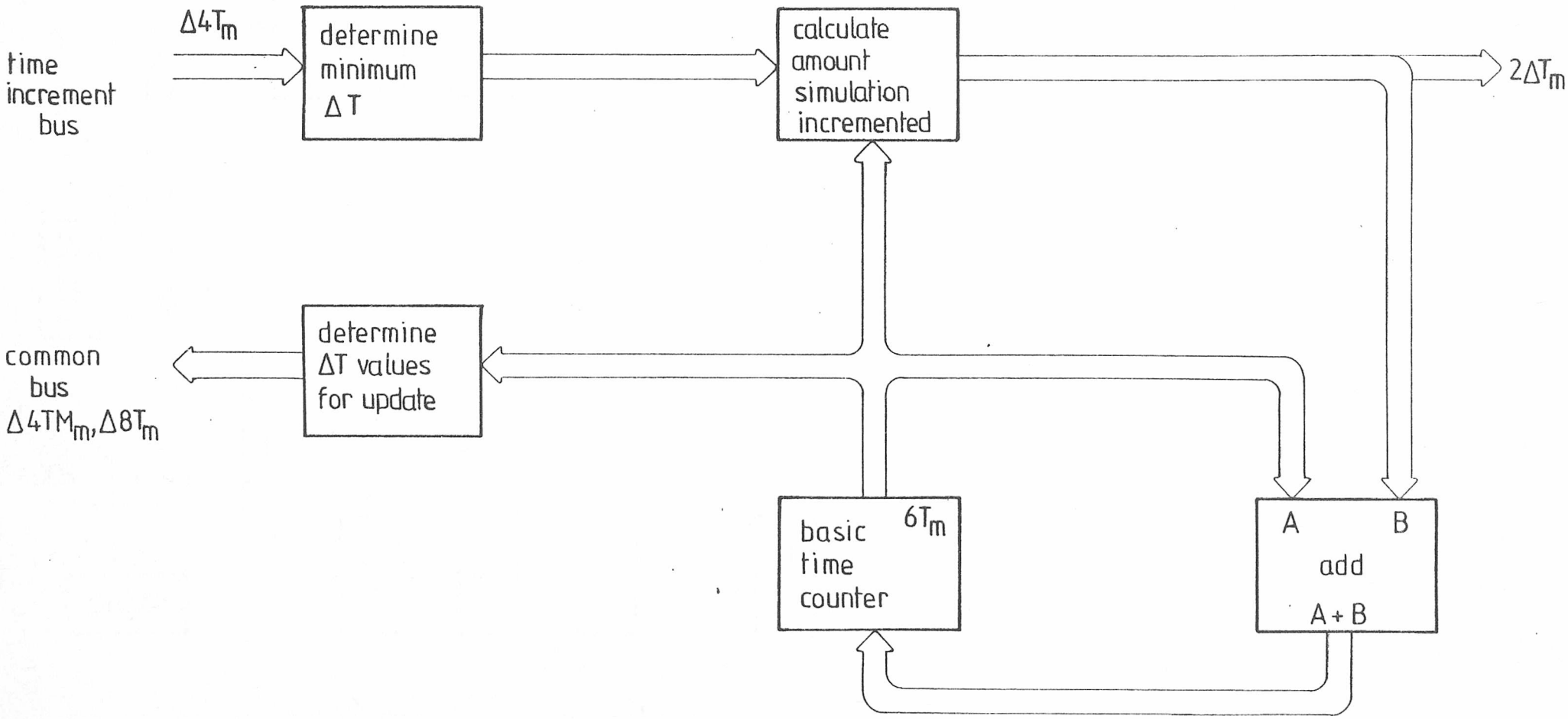


Figure 5.2c Example of ΔT selector operation (control module)

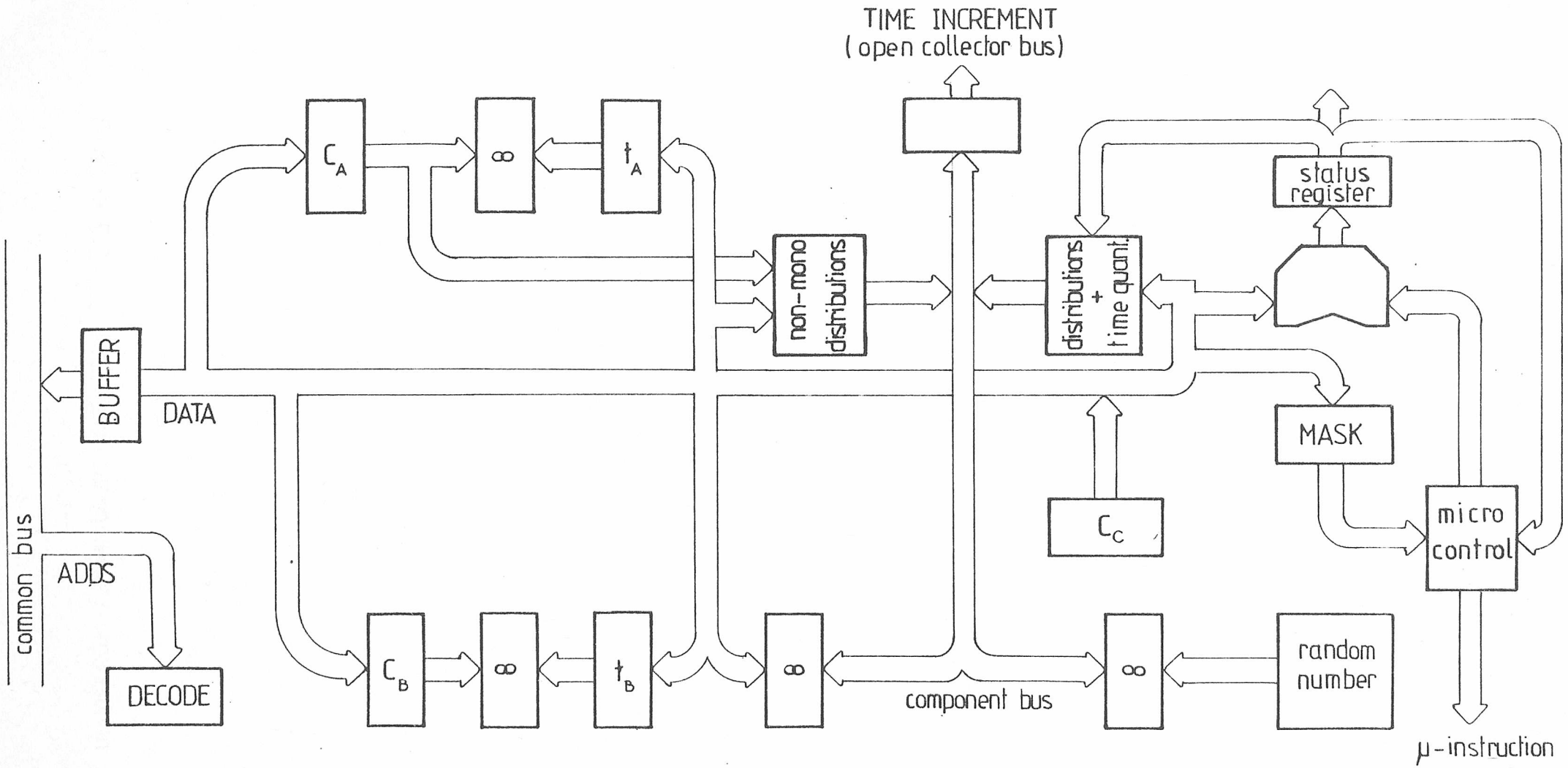


Figure 5.3a Component

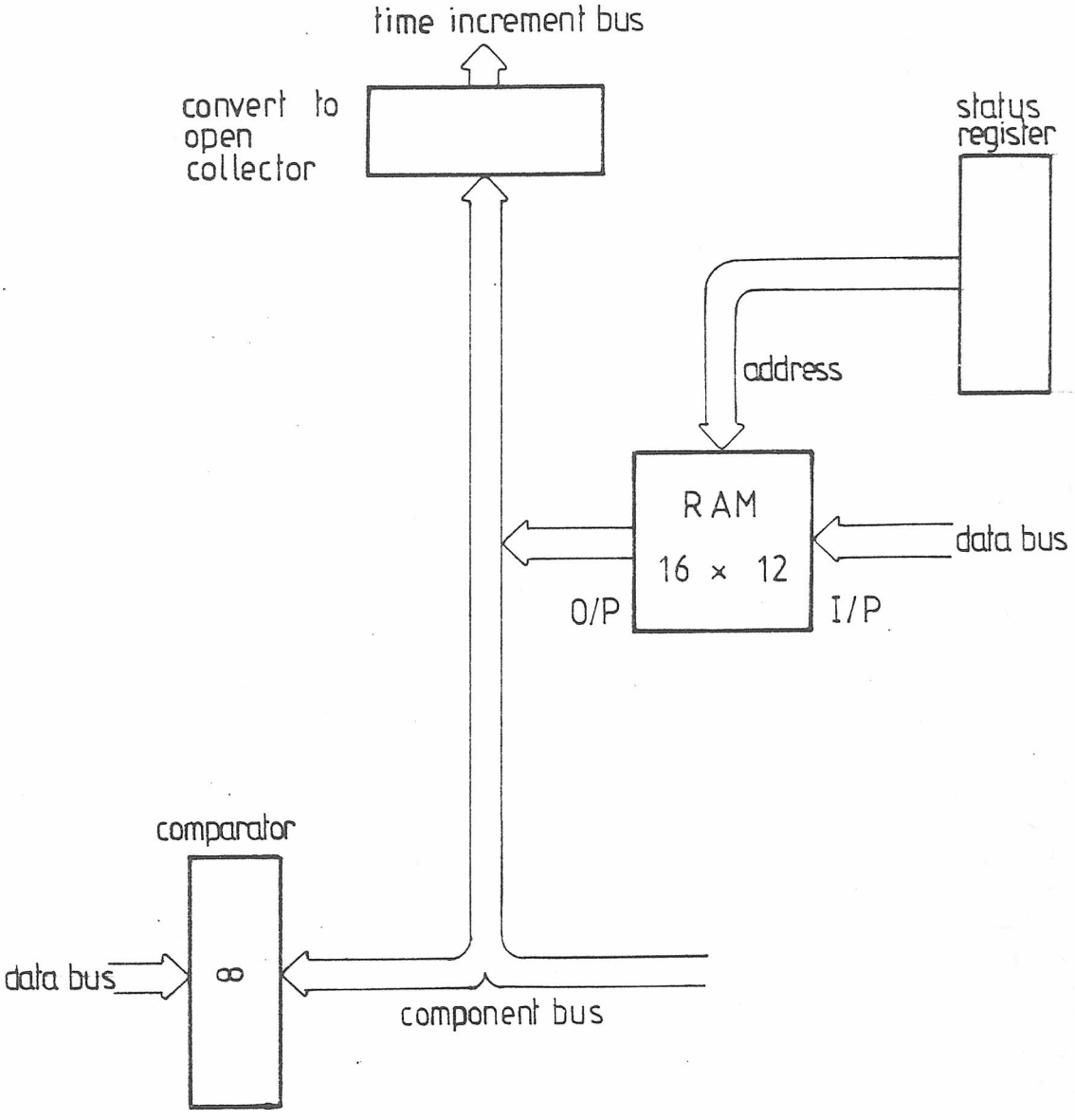
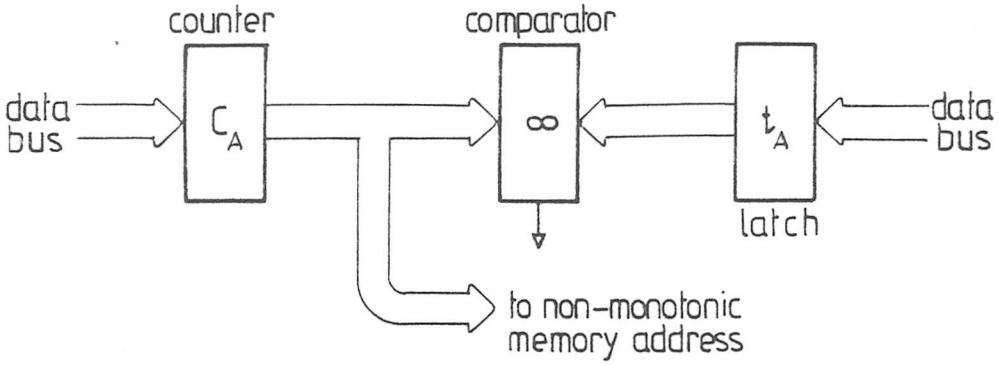
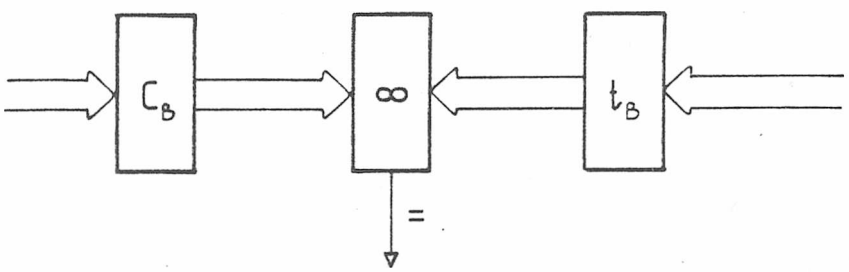


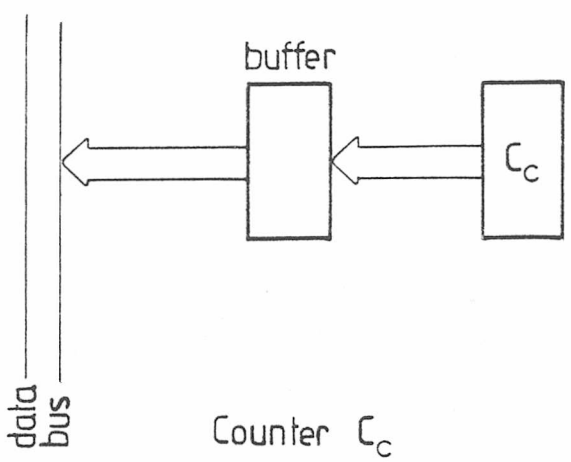
Figure 5.3.1a Time quantisation memory organisation



Counter C_A Organisation



Counter C_B Organisation



Counter C_C

Figure 5.3.3a Component module counters

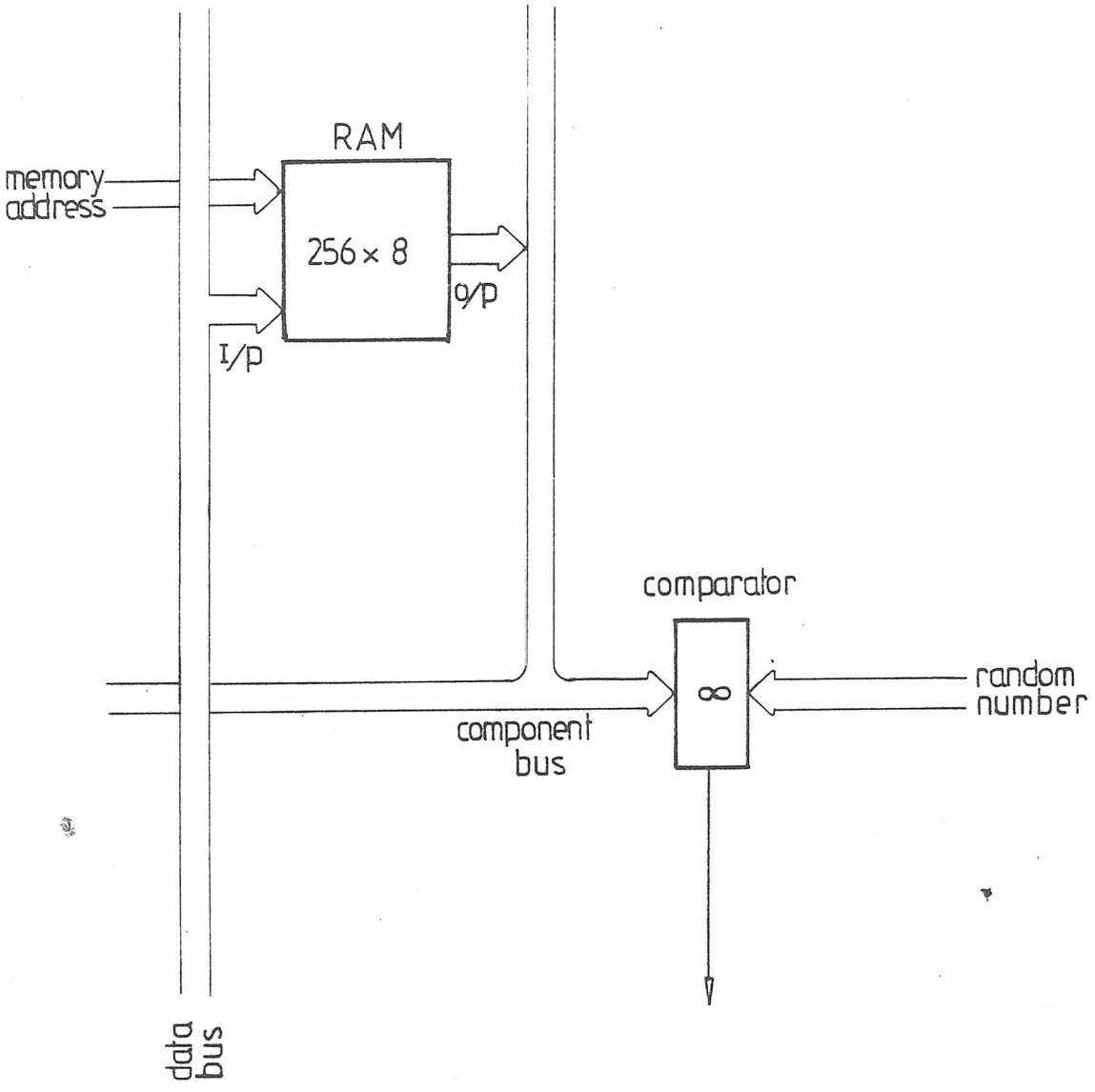


Figure 5.3.4a Non-monotonic distribution circuitry

OPERATION	CODE octal	CODE decimal
—	00	0
LOAD t_B	01	1
ENABLE C_C output buffer	02	2
LOAD MASK REGISTER	03	3
ENABLE 256 x 8 bit RAM	04	4
ENABLE 16 x 12 bit RAM/set 0 at RN i/pt.	05	5
LOAD Status register SR	06	6
LOAD t_A	07	7
LOAD C_B	10	8
CLEAR C_C	20	16
[connect SR to data bus] set 1 at RN i/pt. [prepare to program RN]	30	24
WRITE into 256 x 8 bit RAM	40	32
WRITE into 16 x 12 bit RAM	50	40
CLOCK random number generator RN	60	48
LOAD C_A	70	56

Figure 5.3.5a Programmable component instructions

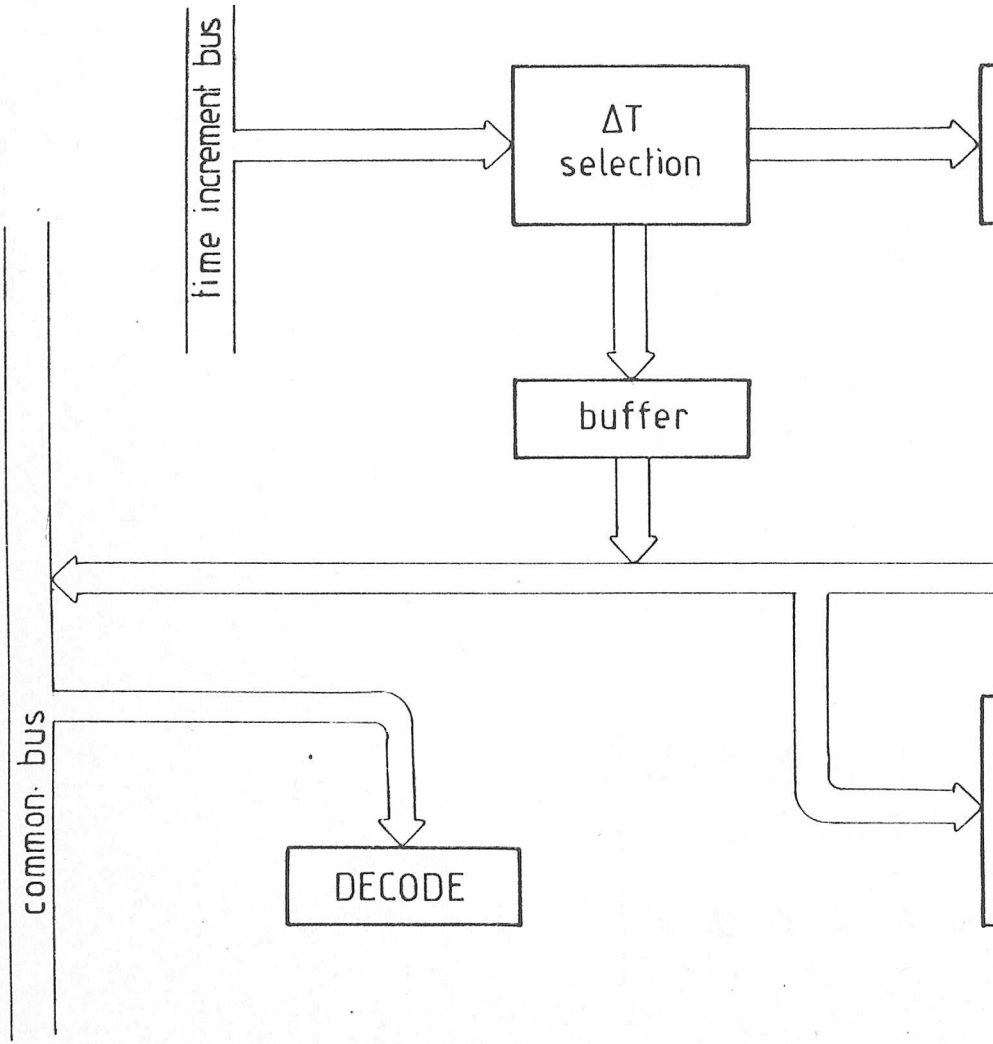
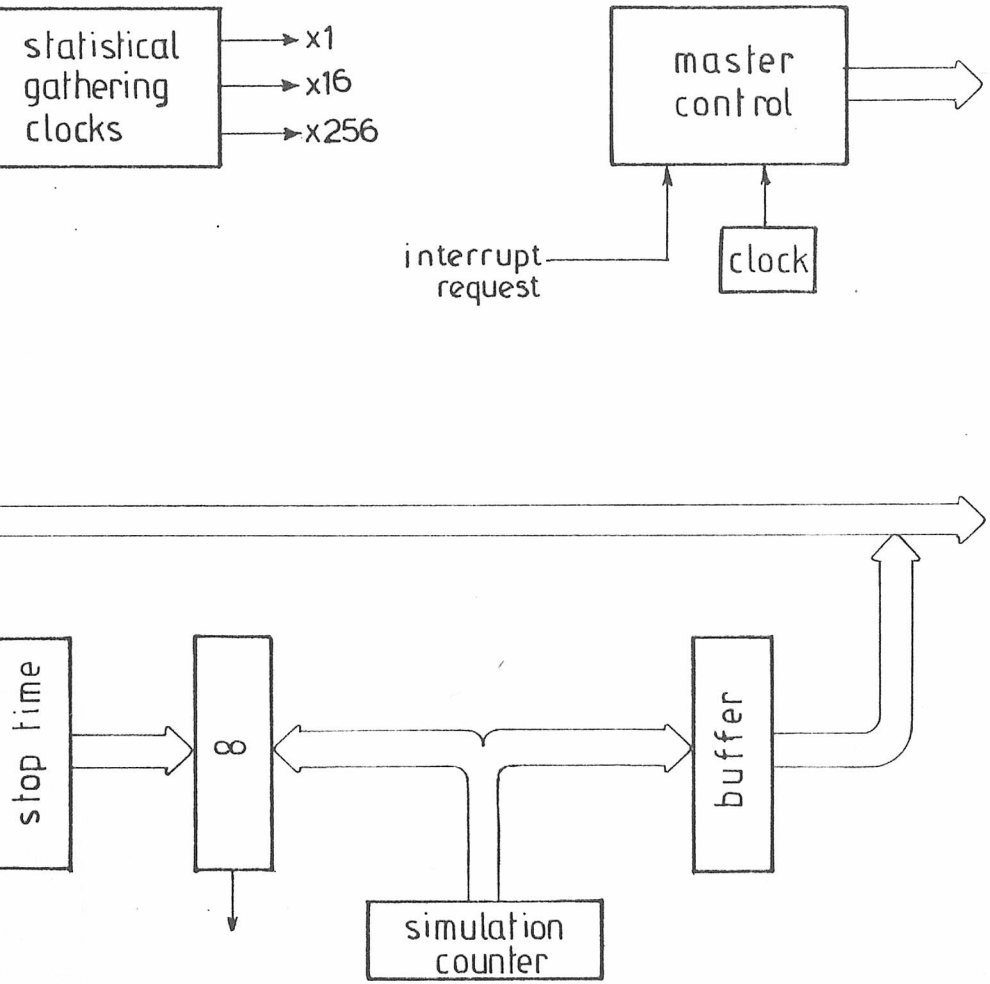


Figure 5.4a



Control module

OPERATION	CODE octal	CODE decimal
—	00	0
START SIMULATION	01	1
RESET MASTER CONTROL	02	2
STOP SIMULATION	03	3
COMMON BUS handshake	04	4
INITIALISE clock generator	05	5
INDICATE STOP at event P1	06	6
enable Component interrupts	07	7
LOAD STOP TIME (low byte)	10	8
LOAD STOP TIME (high byte)	20	16
CLEAR SIMULATION COUNTER	30	24
READ SIMULATION COUNTER (low byte)	40	32
READ SIMULATION COUNTER (high byte)	50	40
INDICATE STOP at time limit	60	48
(disenable Component interrupts)	70	56

Figure 5.4b Control module instructions

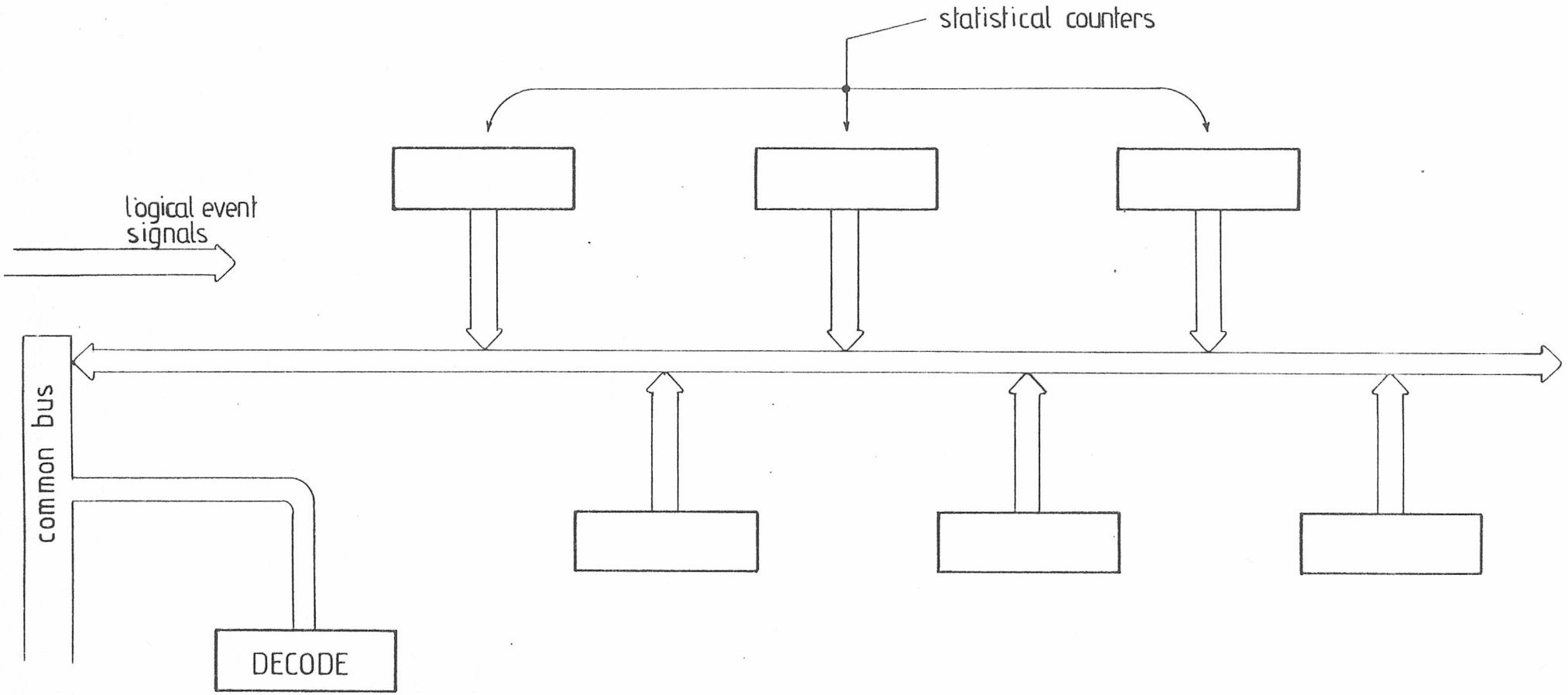


Figure 5.5a Statistical gathering module

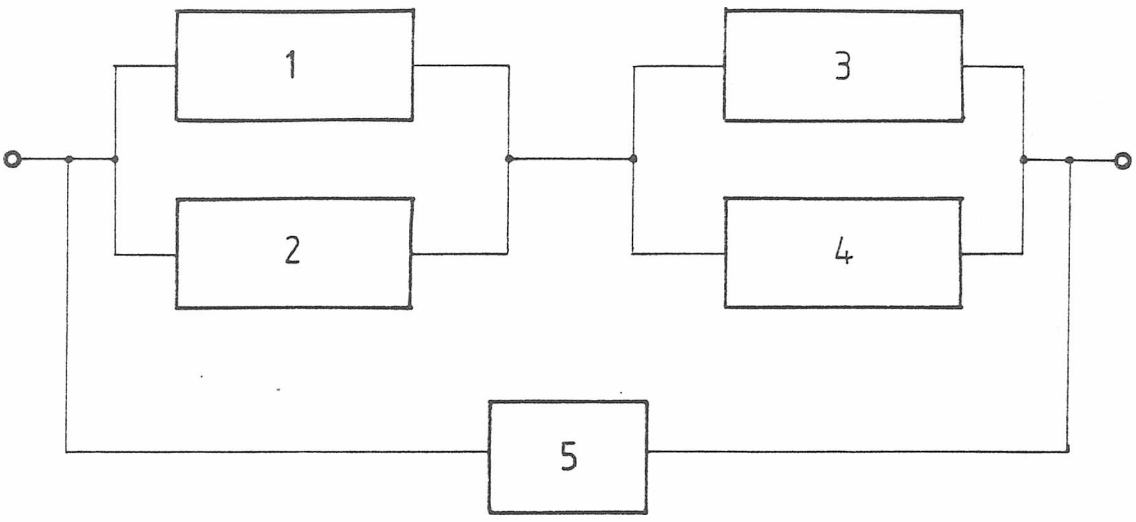


Figure 5·5b Model system flow diagram

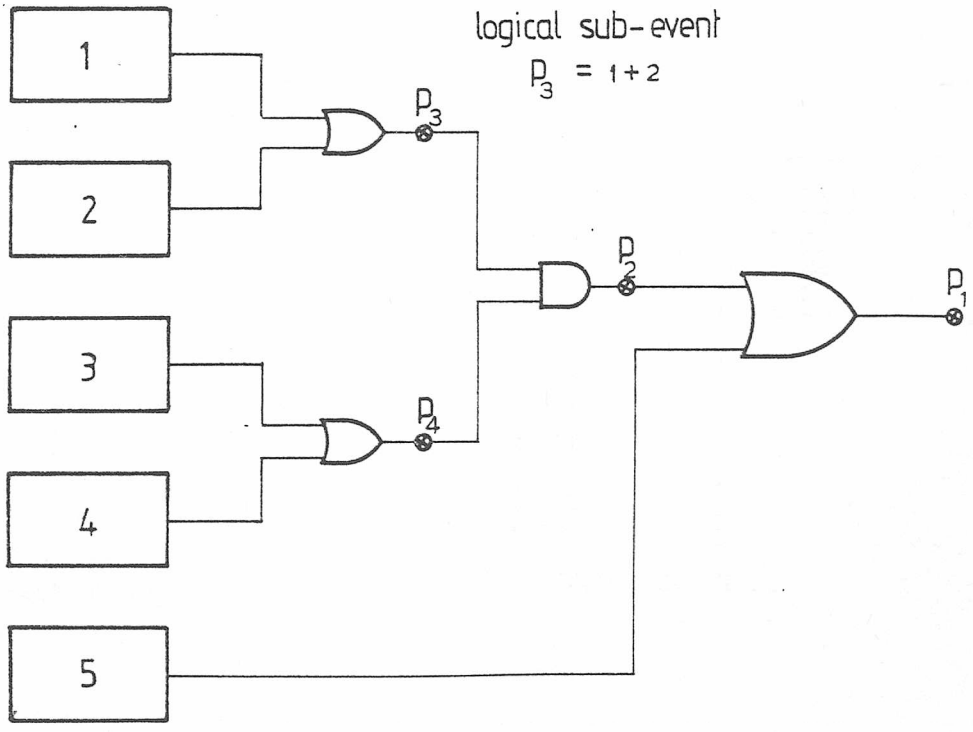


Figure 5·5c Success tree

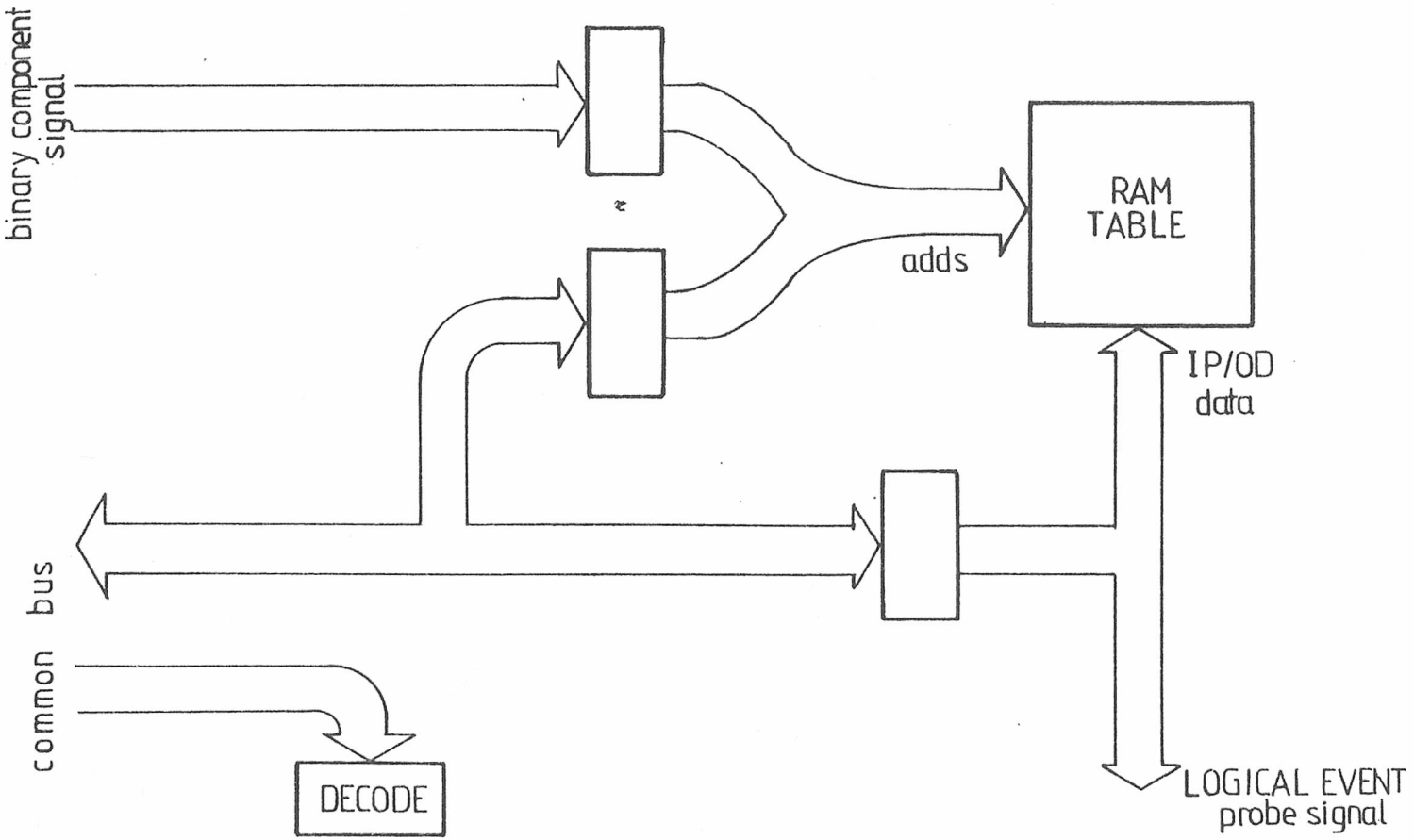


Figure 5·5d Network specification module

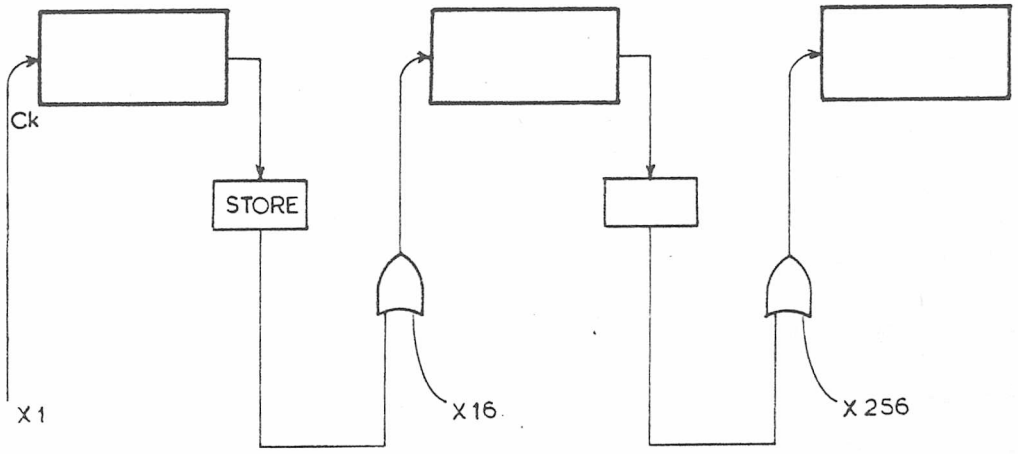


Figure 5.5e Statistical gathering counter recording simulated time

Reliability Modelling View of Simulator

6.0 Introduction

The description of the Simulator so far has been kept very general. No interpretation of model features, within the context of reliability simulation has been made. This was deliberate and is due to the wide ranging nature of engineering systems investigated for reliability performance.

Versatility of modelling components is achieved through programmable operation. Further modelling power is brought to the simulator by the global Policy Module, which makes it possible to implement complicated system management policies without affecting simulation speed. Finally components can, if required, interrupt the simulation with requests for software modelling features contained in the host computer. However it is not suggested that this feature should be generally employed, but only when modelling component behaviour outwith the scope of the programmable components. Repeated interrupts during a simulation would result in longer run times.

To determine the Simulator's performance and enable the investigation of engineering systems to be carried out with a view to improving their reliability, work has continued in the direction of developing a component arrangement which may be considered universal. That is, a component description usable in a wide range of reliability problems.

Coinciding with this development, the necessary software enabling the host computer to prepare the model for simulation, and examine the results obtained has also been written. This Chapter reports on this work and although the model presented is general purpose it should be considered as only one of the possible component arrangements. For

example the problem of modelling minimal repair, described in Chapter 8, is not dealt with by the general purpose micro-controller. To deal with such additional system characteristics, a modified micro-control memory was produced. The operator could then select from a library of control memories to achieve the necessary component operation.

Host software was developed to permit a high level of communication between the Component Modules and the host. The communication is in accordance with the general purpose micro-controller view of hardware operation, and any substantial change made to the micro-controller's direction of the hardware would require changes to the host programs. The higher level enables easy operator interaction with the hardware, and is made possible by the understanding of micro-controller and mask register operation. Throughout the development of the software, ease of operator use has been a prime consideration.

6.1 Status Register

The format of the status register layout is shown on Figure 6.1a. The lower three bits are the actual component status bits indicating the current condition of the component. The next three bits are flags set during modelling. Bits SR4,SR5 are repair (R.R.) and maintenance (M.R.) requests respectively and are made available to the Policy Module. A repair request is issued immediately a component fault is detected. This corresponds to entry into the requiring repair state. A maintenance request is issued on reaching a scheduled maintenance time. However this does not correspond to entry into the component maintenance state. Transition to this state only occurs when the Policy Module permits maintenance to start. Throughout the delay the M.R. bit remains set. With this method, maintenance can be queued during a 'system crisis' or when maintenance resources are in high demand. The Policy Module has a single reply line request allowed (R.A.), which

indicates to the component that the request is granted, corresponding to a release of global resources to meet component demand. At this point status register bit SR3 (ack.R.) is set, acknowledging the response from the Policy Module. The ack.R. bit remains set until the completion of the requested task.

The final status register flag, bit SR6, is the component interrupt request (int.). If the mask register setting enables interrupts then the micro-controller sets the flag when component failure occurs. At this point, the Control Module stops the simulation and indicates to the host all components requesting attention. No general purpose interrupt service routine is presented in this chapter. In later work, reported in Chapter 8, the facility was used to model reliability growth, for which a limited service routine was constructed.

Control of status register flags is simply achieved by the micro-instruction controller within each component. Any variation of operation from that described above is possible by re-programming the control memory.

The eight possible conditions which any component may be in are defined on Figure 6.1b. The state descriptions given are self explanatory. A selection of these states may be used by components to describe quite general systems. Indicated on Figure 6.1b are the two states for which non-monotonic transition rates are possible. The relevant states have been defined such as to allow non-exponential repair and failure distributions, which for reliability modelling are most useful. Also shown are the states which must have the same time quantisation if use of the programmable counters is made in the way described by this chapter. The reason for this is as follows. Counters within components are used to record component age and length of time since previous replacement. With this information the micro-controller can issue maintenance request signals (M.R.). Counter values must be

kept up to date and an incrementing clock pulse may be issued by the micro-controller during a Control Module cycle. Each pulse corresponds to a single quantum time step increment. This makes it impossible to maintain counter updating and unrestricted asynchronous simulation. To overcome this problem, counters are only incremented when the component is in a limited set of states, resulting in certain states having the same time quantisation. The system is acceptable because the time quantisation value is the largest used by the component, and the states restricted to this value are the most commonly visited states. Therefore the amount of simulated time spent in states for which counter incrementation is not possible is small.

6.2 Modelling Counters

Component counters C_a and C_b are used to implement a maintenance policy based on component operation times. The special task of recording the amount of time a component is resident in a particular state is assigned to counter C_a. This is necessary if non-monotonic renewal processes are to be simulated, as only counter C_a addresses the instantaneous hazard rate memory. Whenever a component fails or is repaired/replaced, counter C_a returns to zero. For components in working condition but not operating, such as cold standby, counter C_a is not incremented. At all times the value of counter C_a is compared with T_a. If $C_a \geq T_a$, a maintenance request (M.R.) signal is issued. This corresponds to age replacement after the component completes T_a time units of operation.

Counter C_b is used to generate maintenance request signals at periodic intervals of time. This is known as a block replacement policy. The actual block replacement time is held in register T_b, and once again a digital comparator identifies the maintenance time. The incrementation of counter C_b is continuous (within the limitations of

Section 6.1) and does not depend on the component working. Counter C_b is not reset when the component is repaired/replaced but only when a (M.R.) is issued by the component. It should be noted, that the action of both counters can be separately enabled or disenabled by setting the component mask register.

Counter C_c records the number of component failures. This information may be used when statistically analysing model performance.

6.3 Component Micro-Instructions

A component models a particular system aspect by drawing upon the range of available component states. These states have been listed in Section 6.1. The range of states which can be visited by a component are determined by its mask register setting. The contents of this register constitute inputs to the micro-controller which is responsible for generating state transition signals. Certain bits of the register enable modelling options to be selected more directly, operating on component hardware signals without the micro-controller. The layout of the mask register is given on Figure 6.3a. The effects of individual register bits are described in more detail below.

M0; cold standby. When this bit is set the component moves into the passive standby state if the component which it backs up is working (working states are defined as 3 and 7). The components are numbered 1,2,3,...n, and are positioned in a module card racking system in numerical order. Further, a passive component at position m backs up a working component at position m-1, a higher priority position. If a further passive back up component is added it must be inserted into position m+1.

Components which can be in a passive standby state observe a signal, failure allowed (F.A.), generated by its neighbouring component lower down the rack. Figure 6.3b illustrates the hardware implementation of this ripple signalling. The signal (F.A.) indicates to a passive component that no active or passive component, of higher priority, within the redundant group is working and therefore it is required to become active. It remains in a working state until a component, within the group, lower down the rack (of higher priority) is repaired.

Always the aim of the redundant group is to have the component of highest priority working and all others in a passive state. At all time the highest priority available component is the only component which may be working.

M1,M2; High start up failure, high start up delay. Setting these mask register bits selects the respective modelling options. Actual parameters must be entered into the component 16x12bit RAM. Deterministic start-up delays are implemented by programming a unit impulse p.d.f. for the state transition time distribution. The delay is then the time quantisation value chosen for the 'random' transit time.

M3; Interrupt. Component generated interrupts are enabled by setting this bit. The micro-instruction controller is programmed to generate an interrupt whenever a component failure occurs.

M4,M5; Age replacement, block replacement. To enable the programmable counters to generate maintenance request signals these bits must be set. If both bits are set then maintenance occurs at block intervals only if the component exceeds the age limit T_a.

M6,M7; These bits indicate that non-monotonic renewal processes are used to describe repair and failure times respectively. Setting any of these bits results in the storing of instantaneous renewal rates in the 256x8bit RAM in place of the normal 16x12bit RAM.

The micro-instruction input/output signal layout is given on Figure 6.3c. The output signals are self explanatory and show the control over component hardware. All input signals have been described except random event signal (R.E.) which is the random binary output from the renewal process comparator. Development of the micro-controller firmware must be carefully carried out if correct component operation is to result. Stipulation of control signals to be issued during state transition is somewhat simpler and is given in Appendix A2. The specification of the required input binary patterns causing state transition requires a clear understanding of the nature of component behaviour. A computer program was written to carry out the off-line programming of micro-controller memories. The program produces the micro-instructions to be issued during state transition. Appendix A3 presents the derivation of these signals. It also gives a clear description of the component state transition process.

6.4 Host Computer Software

The operator communicates with the simulator via a VDU terminal. The behaviour of the Simulator is completely controlled from this terminal via the host computer software. Entering key instructions enables the initialisation, execution and analysis of all reliability system studies. The software has been developed with the aim of simplifying terminal input/output format, resulting in efficient use of the Simulator hardware.

On powering up the Simulator and activating the control program at the host computer, the operator is presented with the following list of tasks that the host software is capable of undertaking:

```
INITIALISATION AND CONTROL OF SIMULATOR
 SELECT SUB-SYSTEM LEVEL TO BE AFFECTED
1. . . SIMULATION INITIATE
2. . . COMPONENT MODEL
3. . . STATISTICAL ANALYSIS
4. . . NETWORK SPECIFICATION
5. . . HARDWARE DEBUG
6. . . RE-ENTER SIMULATION
6. . . SYSTEM POLICY
8. . . FINISH SIMULATION

INPUT SELECTION  N=. .
```

To carry out experiments on a system, three files must be created (If these files have been previously generated then they may be recalled for repeated use). They are:

1. the component description file
2. network specification file
3. resource policy file.

Generation of new files can be selected at this point in the control program. All other operations of the host computer are similarly selected at this frequently recurring branch point. On completion of each control or file handling task, the program returns to the common branch point. The following sections outline the entire simulation process, as seen by the operator, from model preparation to analysis of results.

6.4.1 Model Description

A model description file determines the characteristics of components representing system features. When creating or editing the file, each component is selected and individually initialised. A maximum of six components is permitted by the hardware currently available. On first describing any component, a mask register setting

is developed by answering eight questions. The mask setting determines the modelling features to be used during simulation. On the basis of this setting, parameters are requested by the file handling routines. The listing given below shows the format of the parameter specification. The exact format varies for each mask setting and is unlikely to be as large as the one shown, which results from selecting every possible modelling feature.

```
INPUT NAME OF MODEL DESCRIPTION FILE . .MOD0

COMPONENTS IN USE BY MODEL ARE
  1  2  3  4  5  6



MINIMUM TIME QUANTISATION =      1.000 HOURS
INPUT NEW VALUE OR RETURN FOR NO CHANGE   . . . .


NOTE:    1. ALL TQ VALUES MUST BE A BINARY
            MULTIPLE OF TQMIN
         2. ALL VALUES IN HOURS
MAXIMUM TIME QUANTISATION =   2048.000 HOURS
SELECT COMPONENT TO BE ADJUSTED (1 TO  6)
ENTER RETURN FOR NO FURTHER ALTERATIONS N=. .3

IS COMPONENT TO BE REMOVED Y/N. .N

DO YOU WISH TO DEFINE A NEW MASK REGISTER
Y/N   ANS.=  Y

IS THE COMPONENT PASSIVE Y/N. .Y

IS THERE A HIGH START UP FAILURE Y/N. .Y

IS THERE A START UP DELAY TIME Y/N. .Y

ARE COMPONENT LEVEL INTERRUPTS POSSIBLE Y/N. .Y

AGE REPLACEMENT OPERATIVE Y/N. .Y

BLOCK REPLACEMENT OPERATIVE Y/N. .Y

NON-MONOTONIC DISTRIBUTION FOR REPAIR Y/N. .Y

NON-MONOTONIC DISTRIBUTION FOR FAILURE Y/N. .Y

NEW AGE REPLACEMENT TIME (    1000.000)
ENTER RETURN FOR NO CHANGE. . . .

NEW BLOCK REPLACEMENT TIME (    2000.000)
ENTER RETURN FOR NO CHANGES. . . .
```

```
TIME SINCE PREVIOUS BLOCK REPLACEMENT (        0.000)
ENTER RETURN FOR NO CHANGES  . . . . .

INITIAL AGE OF COMPONENT (        0.000)
ENTER RETURN FOR NO CHANGE. . .

DO YOU WISH TO CHANGE ANY DISTRIBUTION Y/N. .Y

FAILURE DISTRIBUTION INDICATOR =  1
SELECT DISTRIBUTION
 WEIBULL . . . . . 1
 ERLANG  . . . . . 2
 2 MODE WEIBULL  . 3
 ENTER RETURN FOR NO CHANGE . . . .

DISTRIBUTION PARAMETERS (  500.0 ,    2.00 )
 ENTER RETURN FOR NO CHANGE . . . . .
 TIME QUANTISATION (        8.000 )
 MINIMUM VALUE FOR 128 ELEMENT MEMORY =       8.000
 ENTER RETURN FOR NO CHANGE . . . . .

REPAIR DISTRIBUTION INDICATOR =  1
SELECT DISTRIBUTION
 WEIBULL . . . . . 1
 ERLANG  . . . . . 2
 2 MODE WEIBULL  . 3
 ENTER RETURN FOR NO CHANGE . . . .

DISTRIBUTION PARAMETERS (   15.0 ,    2.00 )
 ENTER RETURN FOR NO CHANGE . . . . .
 TIME QUANTISATION (        1.000 )
 MINIMUM VALUE FOR 128 ELEMENT MEMORY =       1.000
 ENTER RETURN FOR NO CHANGE . . . . .

REPLACEMENT DISTRIBUTION MEAN (       10.000)
  ENTER RETURN FOR NO CHANGE  . . . . . .
 TIME QUANTISATION VALUE (        1.000) . .

UNREVEALED FAULT DISTRIBUTION MEAN (     1000.000)
ENTER RETURN FOR NO CHANGE  . .

START UP FAILURE RATE (       10.000 % )
ENTER RETURN FOR NO CHANGE . .

IS START UP DELAY DETERMINISTIC OR RANDOM D/R
ENTER FOR NO CHANGE          . . . . . R

INPUT MEAN RANDOM DELAY (        5.000). .
 TIME QUANTISATION VALUE (        1.000). .

INPUT STARTING STATE OF COMPONENT (      71.000). .

SELECT COMPONENT TO BE ADJUSTED (1 TO  6)
ENTER RETURN FOR NO FURTHER ALTERATIONS N=. .
```

Where non-exponential distributions are in use, an indicator of distribution type must be entered. The 2-mode Weibull indicator refers to failure times which occur according to two modes, each mode described by its own distribution [see Section 8.9].

Time quantisation values for non-exponential distributions can be selected freely. However the host computer determines the minimum quantisation necessary for the distribution c.d.f. to reach 0.95 before the final hazard rate memory location is reached. The suggested value is indicated as 'minimum value for 128 element memory'.

6.4.2 System Success Tree

The second file necessary for complete system description is the network specification file. This file contains information for forming the system success tree, which must be entered into the look-up table in the Simulator. There are two separate methods of creating and editing this file. If a conventional operator terminal is in use then the system success tree is constructed from logical boxes. The operator specifies the connection of boxes by identifying each with a number. An example of the output format for a 6-component system is shown on Figure 6.4.2a. Describing system success trees by this method is simple for small systems but could be tedious and error-prone for larger systems. At the completion of the tree description by this method, the operator is presented with a table which is merely a record of box connections.

The second form of file handling requires an interactive graphics terminal. This method is by far preferable as the operator need only 'draw' the component boxes and their connections to specify the system tree. Figure 6.4.2b illustrates the tree drawn for the example system above. The software controlling graphical editing of the tree has been designed for simple operator use. Both tree construction methods

require specification of statistical gathering probe positions, corresponding to logical events about which information is to be gathered. However the graphical method is particularly convenient to use as previous probe points can be quickly 'rubbed out' and the probe redrawn in a new tree position.

6.4.3 Repair Policy

The policy file contains information about the utilisation of repair and maintenance men. A distinction is made between repair and maintenance actions performed on components. However the men available to carry out the work are drawn from a common group. The distinction is necessary as the response to requests for a man to carry out preventative maintenance is likely to be postponed during a crisis where repair men are better employed on urgent repair work. A consequence of this more accurate modelling of system management policy is that each component requires two binary address lines to describe its condition to the Policy Module.

With a priority repair and maintenance policy, it is to be expected that, on occasion, resources released to meet a component request may be recalled and re-issued to a higher priority component. Later when the initial component resource demands are of sufficient priority to continue they commence from the point at which they left off. In the case of interrupted maintenance the component remains in the unusable maintenance state. If the (ack.R.) signal is made available to the Policy Module then decisions can be made with the knowledge of previously incompleted repair/maintenance operations within the system components. To prevent an increase in the number of policy address lines component signals are coded as follows:

working order -- no request flags set

request for repair -- a R.R. but no ack.R.

request for maintenance -- a M.R. but no R.R. or ack.R.

task commenced -- an ack.R.

With large numbers of components, the policy table may become too large, with regard to realistic implementation. Grouping components into sub-policy tables of six components results in a practical table size of 4096x6 bits.

In programming the Policy Module table, it would be impossible to construct a table describing the actions to be taken for each combination of component states as the possible number of combinations could be very large. In practice, statements are made about component requests of particularly high priority. With the knowledge of these statements the host computer determines the decision table necessary to implement the policy and then programs the Policy Module. An example of programing a policy is given below.

```
INPUT NAME OF POLICY FILE . . .POL2

REQUESTS OF PRIORITY   1 ARE :
   COMPONENT   2 R
REQUESTS OF PRIORITY   2 ARE :
   COMPONENT   3 R
REQUESTS OF PRIORITY   3 ARE :
   COMPONENT   4 R
REQUESTS OF PRIORITY   4 ARE :
   COMPONENT   2 M
   COMPONENT   3 M
   COMPONENT   4 M
NUMBER OF POLICY "MEN" = 2
MINIMUM ACCEPTABLE PRIORITY = 6
COMMENCED TASKS HAVE OVERRIDING PRIORITY

ADJUST SYSTEM POLICY. REQUEST TYPES ARE:
M=MAINTENANCE
R=REPAIR

SELECT COMPONENT
ENTER RETURN TO END. . .2

INDICATE REQUEST TYPE . . .R

SELECT PRIORITY
```

```
ENTER RETURN TO REMOVE PRIORITY SETTING . . .1

SELECT COMPONENT
ENTER RETURN TO END. . .

DO COMMENCED TASKS HAVE OVERRIDING PRIORITY Y/N . . .

MINIMUM ACCEPTABLE PRIORITY ( 6 )
ENTER RETURN FOR NO CHANGE . . .

NUMBER OF POLICY "MEN" AVAILABLE . . .

SELECT OPTION
0.  EXIT
1.  DISPLAY POLICY
2.  PROGRAM MODULE . . .
```

Note that component requests of priority below the stated minimum acceptable priority do not receive attention from repair men. When a repair or maintenance task is acknowledged by the Policy Module the component enters the 'task commenced' state. This state has the same priority as the repair state since they both describe a non-operational component. However by stating that commenced tasks have overriding priority then repair or maintenance tasks, once initiated, cannot be discontinued by the occurrence of a request of higher priority.

6.4.4 Executing an Experiment

After the development (or recall) of the necessary system description files has been completed, the operator initiates experiments on the model system as shown below.

```
NAME OF FILE TO KEEP RESULTS . . RST1

NO. OF EXPERIMENTAL RUNS . .100

NUMBER OF WINDOWS . . .
ENTER RETURN TO STOP AT EVENT P1 . .2

TIME(HOURS) AT END OF WINDOW 1 =. .10000

TIME(HOURS) AT END OF WINDOW 2 =. .20000
```

A file is used to contain results, allowing examination at a later date.

The number of repeated experimental runs is entered, each run starting from the same specified initial system condition but with a different random number seed. During each run the Simulator is requested to stop at window points, at which the host computer updates its information about model behaviour. Any number of window points can be used, the actual times being entered by the operator. A request to stop at the first occurrence of logical event P1 can be selected. Clearly in this case no stop times are entered. The technique is particularly useful if event P1 is specified as 'system down'.

Experiments run to completion unless interrupted by the operator, an example of which is shown below.

```
COMPLETED  10 OF 100 RUNS
TIME TO COMPLETION =  2.06 SECONDS

SIMULATION STOPED BY OPERATOR
1.  . RETURN TO MAIN PROGRAM
2.  . CONTINUE SIMULATION

                              N=.  .  .
```

On interrupt, the simulation is suspended and the operator may continue the experiment or return to the main feature selection point. When the operator interrupts the experiment the simulation run may be pre-examined, aborted, pre-examined, restarted or re-continued from the suspended position. Further, on causing an interrupt, the operator is informed of the Simulator's progress and of estimated time to experiment completion, as determined by the host computer.

In the example interrupt, the time given is for a simulation using at all times only the minimum time quantisation value. In practice, the time to completion would be reduced due to selection of larger quantisation values. Set against this is the increase in experimental run times due to delays experienced in use of a time shared host computer.

The serial communication link between the host and the Simulator operates at its limit of 9600 bits/second. At this speed the problem of communication delays is reduced. It should be noted that run times reduce as the number of stop windows and statistical probes are reduced due to the reduced amount of data passing between the host computer and the Simulator. Probe control signals are not issued for probes unspecified at the Network Specification file. In practice, the estimated time to completion indicated to the operator takes into account both of the effects described above.

6.4.5 Results Analysis

On completion of an experiment the operator is returned to the main procedure selection point of the control program. At this point, a statistical analysis of a results file can be requested. The format of the statistical analysis is shown below.

```
ENTER NAME OF RESULTS FILE . .RST1

MINIMUM TIME QUANTISATION = 1.000
NO. OF EXPERIMENTAL RUNS  = 100
```

PROB	WINDOW	TOTAL DOWN-TIME	NO. OF EVENTS	EVENT DURATION	CRV
1	1	60305.2	592.1	101.85	0.012
1	2	59982.8	587.5	102.10	0.015
2	1	12491.6	826.2	15.12	0.011
2	2	12406.2	793.7	15.63	0.009

```
IDENTIFY PROB OF INTEREST
THERE WERE   2 IN USE          =
```

Results obtained for each probe are analysed separately. The values recorded by probe counters are averaged over the number of experimental runs performed to determine mean down time, mean number of events and mean event duration. Each window is dealt with separatly. Therefore the table of results presented lists moving average values monitored

between window points. In the final column of the table of results, the coefficient of relative variation (CRV) for down time is determined. This measure aids the experimentor in determining if results are a good statistical estimate of the true value [see Section 8.1]. If a graphics terminal is in use the operator may request a frequency distribution plot of event times or number of events occuring for each probe window. A hard-copy graph plotter option can also be selected.

No specific routines were written for analysing simulation results recorded in Component Module counter C_c. However each counter would be examined and results recorded.

6.4.6 Hardware Debug

A software control section not required by the experimenter is the hardware debug. This section played a considerable role in the testing of the simulator hardware modules. It enables specific hardware control signals to be issued requesting and entering data into individual circuits in the Simulator. Pre-programmed sequences of instructions can also be issued to further check hardware operation.

6.5 Conclusion

Two main conclusions can be drawn from this chapter.

The universal instruction set developed for the programmable controllers enables a detailed description of component features to be achieved. Its flexible operation should free the operator from the need to re-program the micro-controller. The operator merely prepares a mask register setting to select from a range of modelling characteristics which make full use of component modelling power.

If an interactive graphics terminal is used then graphical

specification of the model system tree is catered for, and immediate displays of result distributions can be obtained.

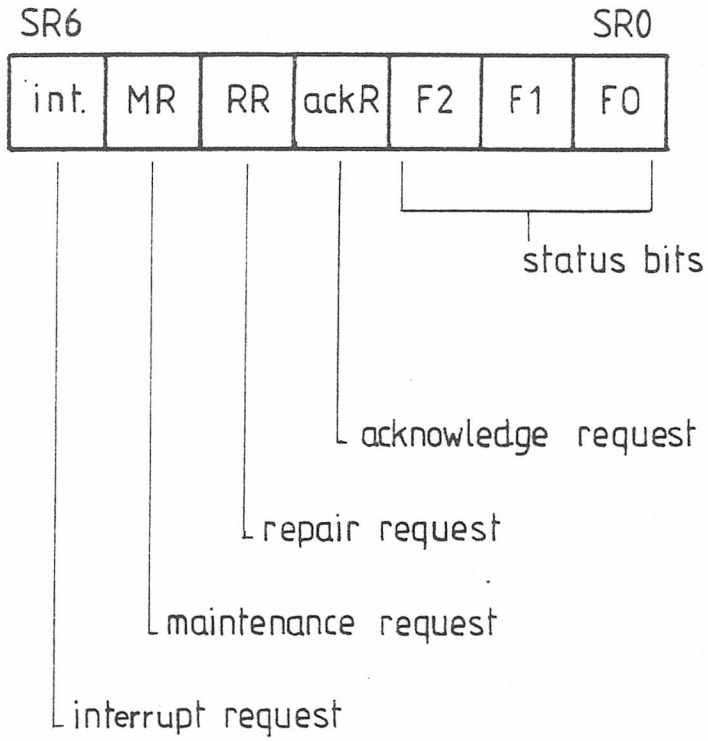


Figure 6.1a Status register format

	non-monotonic distribution available	status F2	F1	FØ	decimal	state definition
*		0	0	0	0	unattended failure
*		0	0	1	1	working order cold standby
	x	0	1	0	2	requiring repair
		0	1	1	3	suffering additional risk during starting (SUF)
		1	0	0	4	undergoing scheduled maintenance
		1	0	1	5	unavailable during starting delay (SUD)
*		1	1	0	6	unrevealed fault condition
*	x	1	1	1	7	working order

* states with same time quantisation

Figure 6.1b Universal state definition of micro-controller

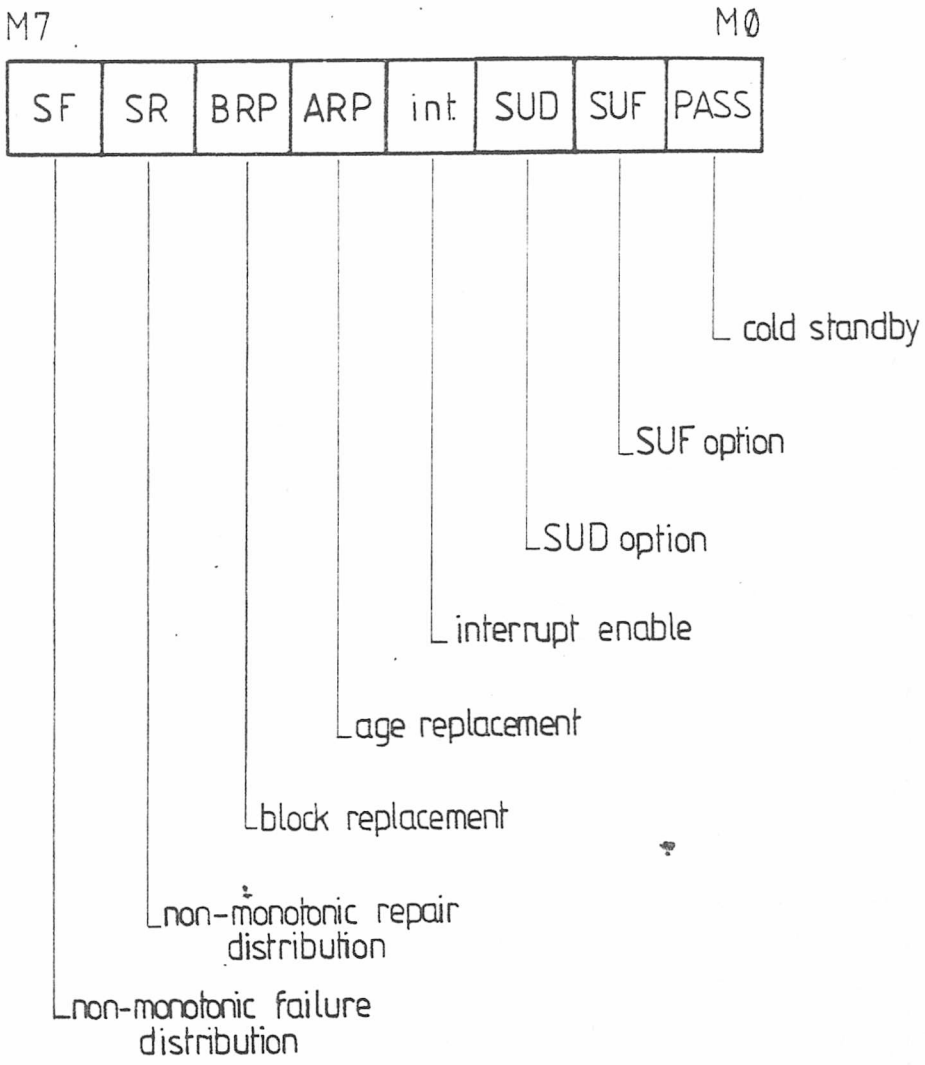


Figure 6.3a Mask register format

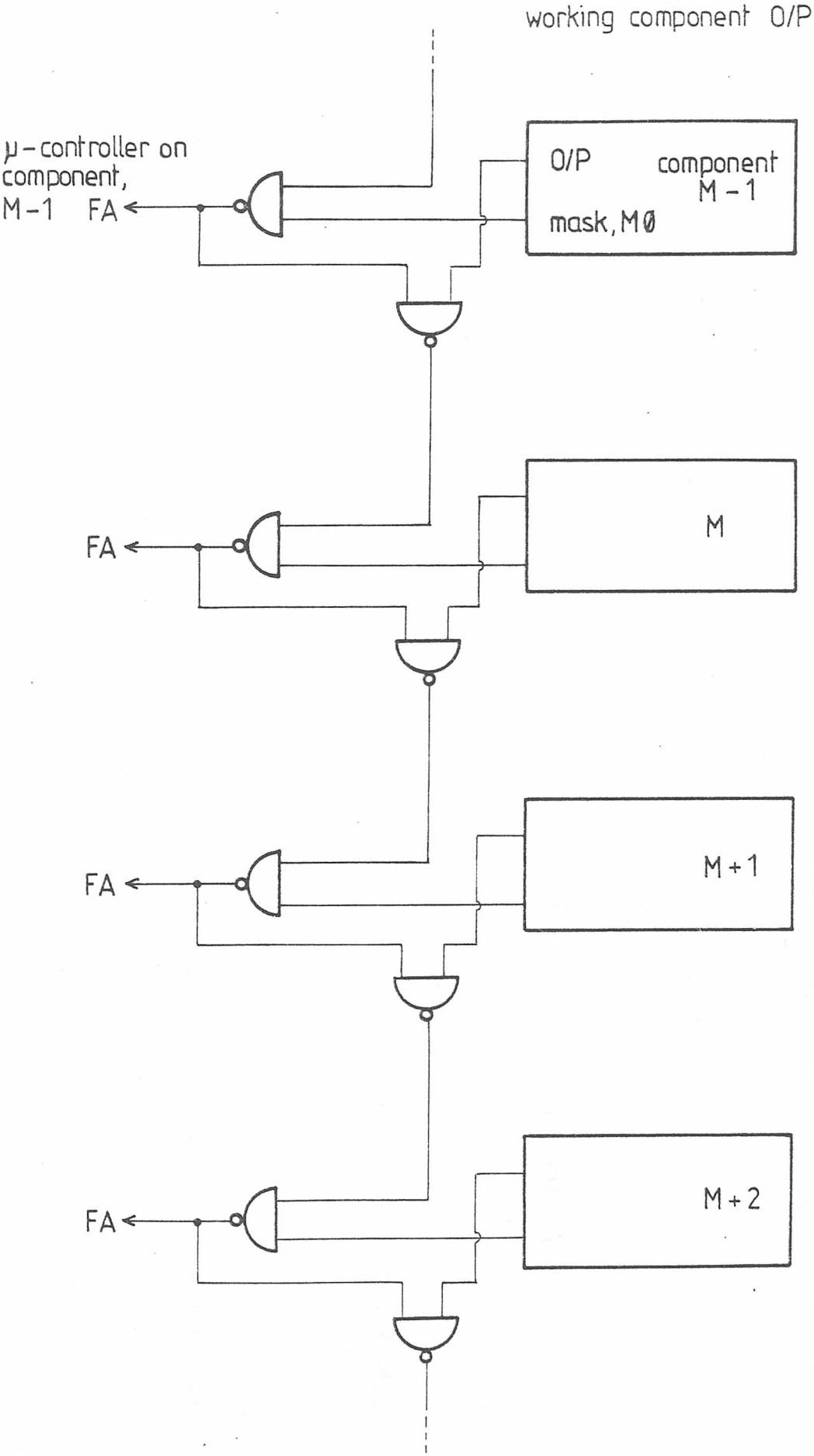


Figure 6.3b Passive redundancy arrangement

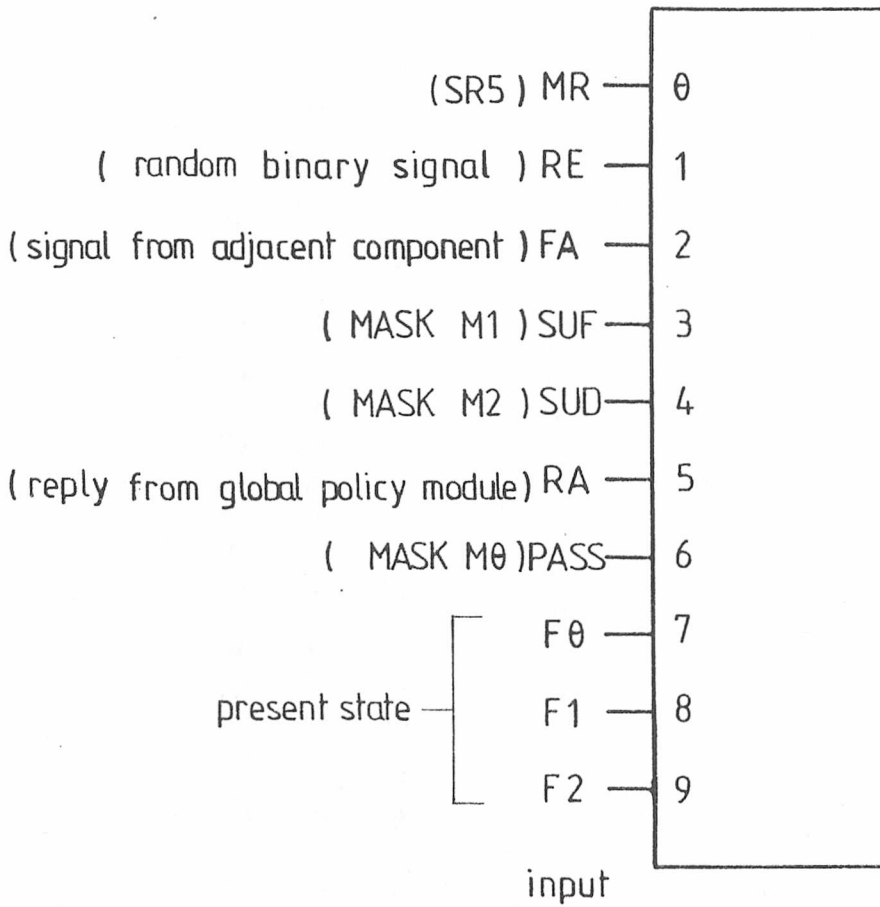
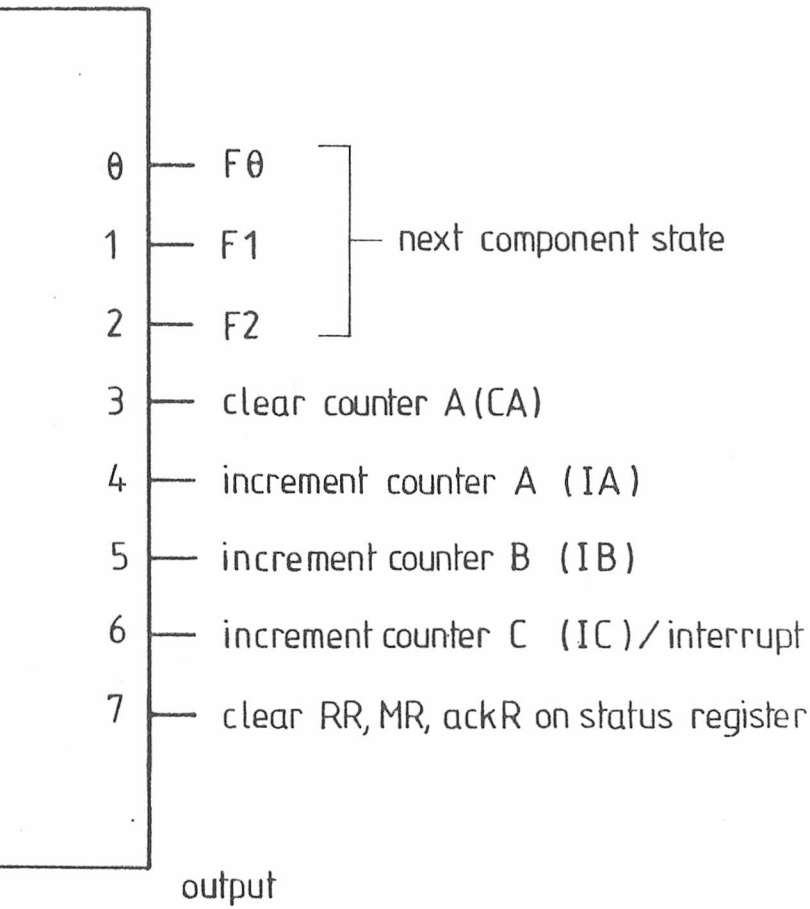
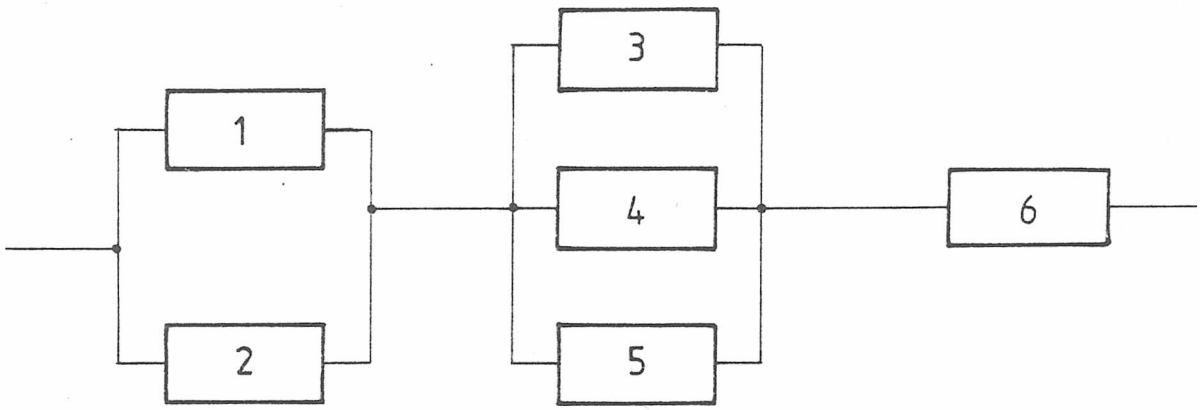


Figure 6.3c Component μ-



- instruction control



flow diagram

probe 1 connected to box 9

probe 2 connected to box 7

probe 3 connected to box 8

box 7 requires 1 signals and is fed from 1 2

box 8 requires 2 signals and is fed from 3 4 5

box 9 requires 3 signals and is fed from 6 7 8

Figure 6.4.2a Logical success tree

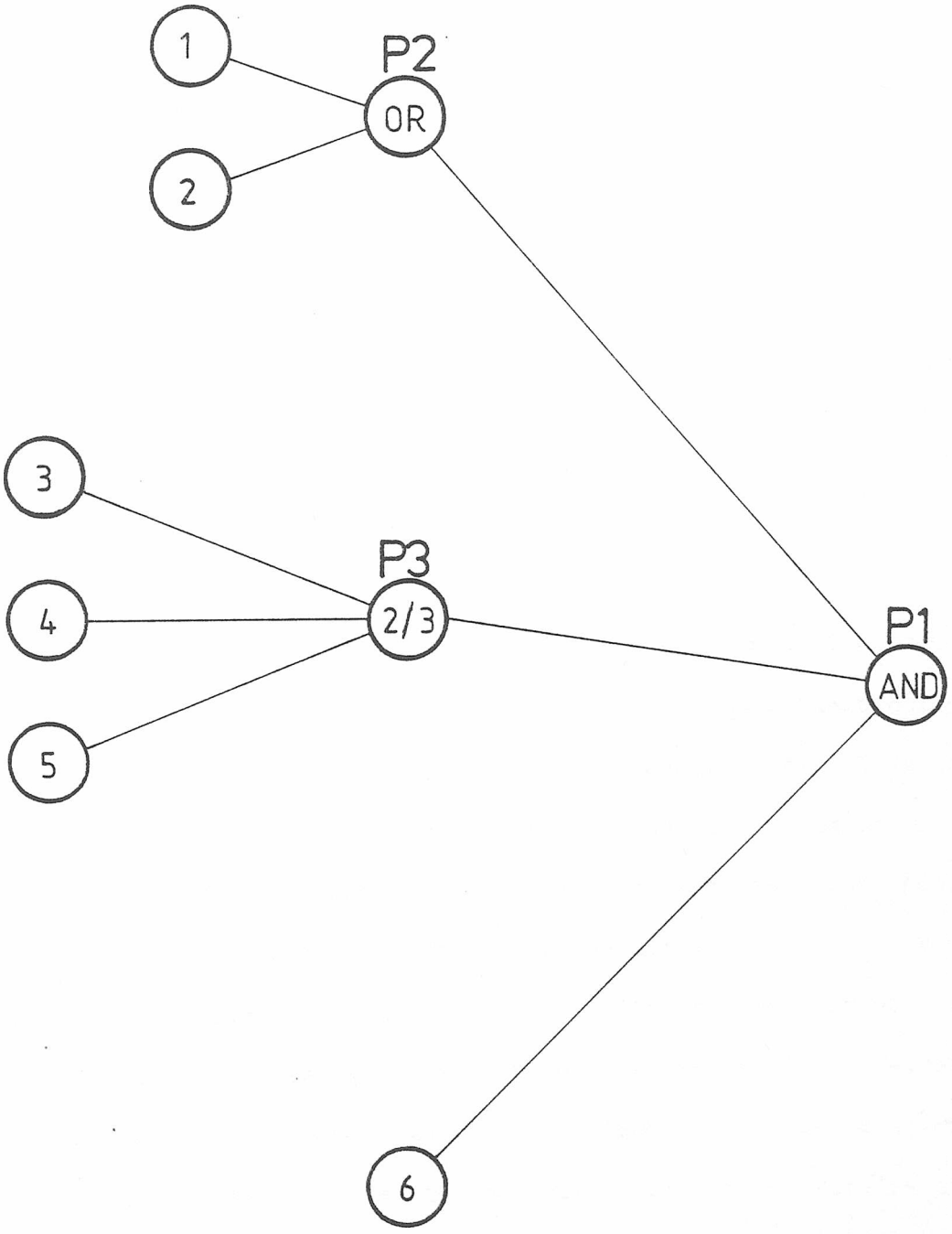


Figure 6.4.2b Graphical development and editing of system success tree

Verification of Simulator Operation

7.0 Introduction

To verify the operation of the Simulator, a collection of system arrangements has been studied. Each system contains some different aspect of reliability engineering, and the range of systems examined covers the principal system topologies. A further consideration during each experiment was the full evaluation of the Simulator's characteristics. For this reason, system parameters have been varied to aid with identification of any modelling problems. In particular, the distribution time quantisation TQ was widely varied, in many cases outwith the normal working range.

For each experiment carried out, the Policy Module was programmed to allow repair or maintenance requests to be acknowledged without delay. That is, system components did not compete for repair men. This is necessary as an investigation of policy dependency would not have permitted any analytical estimation of simulation results, and hence detracted from the verification of Simulator operation.

The following sections of this Chapter describe the experiments in detail and results are presented. In each case, a mathematical analysis of the system has been performed, an essential task if confidence in the operation of the Simulator is to be obtained. Many results are presented in the form of mean unavailability (u_D) or mean availability (u_A).

Where preventive maintenance was applied, the mathematical analysis of the systems considered maintenance to be instantaneous. This cannot be achieved by the simulator. In practise maintenance times were programmed to have their minimum values of ΔT_{min}. Consequently, to permit any comparison of the u_D values obtained by simulation with the

theoretically expected values, the results have been adjusted as follows:

$$\text{down time} = \text{measured down time} - \frac{T_s}{T_B} \Delta T_{min}$$

where T_s = simulated life time

T_B = interval between maintenance

7.1 Single Component with Periodic Replacement

The simplest system conceivable is that of a single component for which no repair takes place. Initially the component is in perfect working order. During its operation it suffers a probability of failure according to a specified probability distribution. Once failure occurs the component remains in a failed state until it is replaced. Component replacement occurs at regular test intervals regardless of the component still being operational. Experiments carried out on systems of this simple nature enable the behaviour of the Simulator to be more easily analysed. Complex systems may have produced results obscuring behavioural trends which are of important to this Chapter.

The replacement policy described is best simulated by specifying a block replacement time T_B. The Repair and Maintenance Policy Module must be programmed to only permit component maintenance requests to be allowed. Component mean maintenance times are set to a value less than ΔT_{min}, this results in an actual maintenance time of ΔT_{min}. All systems have been simulated over long time intervals permitting component unavailability to be estimated with little variance.

Consider a component for which T_B=1000 hours and ΔT_{min}=1 hour. The component failure distribution is exponential, with a mean value 1/0. This arrangement has been modelled for 1/θ values of 200 hours, 1000 hours and 4000 hours where the distribution time quantisation (TQ) has been varied from 1 hour to 128 hours. Selecting 1/θ values above

T_B enables a check to be made on the generation of the correct distribution shape. When lower $1/\theta$ values are used, the results are credible if only the distribution mean value is accurately modelled. However, a value of $1/\theta$=4000 hours where TQ=1 hour has been previously shown to be rather high for accurate modelling to be achieved. (Section 3.8 indicated the safe limit of $1/\theta$ to be approximately 1500 hours). Theoretically, the component unavailability is given by [41]:

$$u_D = \frac{1}{T}\int_0^T P(t)\,dt$$

$$= 1 - \frac{1}{T\theta}(1 - e^{-T\theta})$$

where $T = T_B$

$P(t) = 1 - e^{-t\theta}$, failure distribution c.d.f.

System life times of 10^7 hours have been modelled and the error in predicted mean unavailability determined. The results are shown on Figure 7.1a. As $1/\theta$ is increased there is a slight increase in error due to the greater importance of distribution shape and the difficulty of accurately representing W_k values. However clearly the most important trend revealed is that of increasing TQ values producing poorer results. This is explained by choosing TQ values too large to make full use of the probability resolution possible. In addition, T_B is not an integer multiple. Hence its integer representation must also introduce error most noticeable at high TQ values.

To further examine the effects of high TQ values, T_B was reduced to 100 hours and TQ varied from 1 hour to 16 hours for an exponential distribution. The mean component life time ranged from 50 hours, 200 hours, 500 hours and 1000 hours. The results are shown on Figure 7.1b. The error in mean unavailability has now increased at low

TQ values particularly where $1/\theta$ is greater than T_B, indicating some difficulty in achieving the correct distribution shape. The higher than previous error occuring at TQ=16 hours is due to T_B integration.

Experiments have also been carried out on a component described by a Special Erlang failure distribution. The mean unavailability is calculated as before, with $P(t)=1-(1+t\theta)e^{-\theta t}$:

$$u_D = \frac{1}{T}\int_0^T P(t)\,dt$$

$$= 1 - \frac{2}{T\theta}\left[e^{-\theta t}\left(1+\frac{\theta T}{2}\right) - 1\right]$$

A model where $1/\theta$=400 hours, T_B=200 hours and ΔT_{min}=1 hour has been considered. The time quantisation TQ was varied from 1 hour to 32 hours. The results obtained for the error in u_D are shown on Figure 7.1c. Two curves are presented, showing the advantage of the feedback improvement developed in Chapter 3, clearly very considerable where low TQ values are in use. This is explained by the increased demand put upon the reduced non-monotonic probability resolution by small time steps. In such cases good results are only achieved with the aid of feedback.

The suggested minimum TQ value shown is the value indicated to the experimenter (by host computer software predictions) at the time of experiment initialisation. It corresponds to the minimum TQ value necessary if the modelled distribution c.d.f. is to reach 0.95 before the final W_k memory address. (That is when C_A=127 units). Best results have been achieved at TQ values equal to half the minimum indicated. This is explained by the automatic use of the extended memory feature (described in Chapter 6) available when only one non-monotonic distribution is in use. All 256 W_k memory locations were available for distribution modelling.

Further experiments employing Special Erlang failure distributions have been carried out with T_B once again 200 hours and $1/\theta$ varied from 25hours, 100 hours to 400 hours. The results for the error in predicted unavailability are shown on Figure 7.1d, and compare well with the previous exponential distribution experiments. A similar trend of error at high TQ value is indicated.

Throughout the experiments described in this sub-section a wide range of parameters have been employed to highlight Simulator trends. Although the results presented are in the critical form of error between theoretical value and experimental value, they are believed to be within acceptable limits. In practice, TQ values would not be selected as high as those tested here. When non-monotonic distributions are in use, selection of TQ values according to the minimum indicated by host software can be used. However there appears no danger in selecting values below the minimum as the feedback technique employed maintains modelling accuracy.

A limitation to selection of TQ values not illustrated by the experiments of this section is the maximum size of the maintenance counter C_B. For example, selecting TQ to be 1 hour limits the maximum period between maintenance to be 4096 hours. Periodic replacement times longer than than 4096 hours require failure distributions to be modelled by TQ values greater than one hour.

7.2.1 Series Systems with no Repair

A series system of m components implies that all m components are required to be operational for the system to be operational. With non-repairable systems it is assumed that all components are initially in perfect working order. Each component suffers a probability of failure during its operation time, and if failure occurs with any component the system is rendered in the failed state. Periodically, all components are tested and replaced by new components regardless of any component still being operational. If the probability of component n being operational at time t is given by P_n, then the probability of the system being operational is given by:

$$P_x = \prod_{i=1}^{m} P_i$$

P_n is determined from $P_n = 1 - P_n(t)$ where $P_n(t)$ is the failure c.d.f. of component n.

Consider the case of a two-component system. If component failure distributions are given by $P_a(t)$ and $P_b(t)$, then the probability of successful system operation up to time t is given by:

$$P_x = [1 - P_a(t)][1 - P_b(t)]$$

To determine the mean system unavailability, the standard equation below can be used.

$$u_D = \frac{1}{T} \int_0^T P_x(t)\, dt$$

where $T = T_B$, periodic replacement time

$P_x(t) = 1 - P_x$, system failure c.d.f.

If components have failure distributions described by exponential distributions of mean $1/\theta_a$ and $1/\theta_b$ respectively, then the solution of u_D is simply:

$$u_D = 1 - \frac{1}{(\theta_a + \theta_b)} \left(1 - e^{-(\theta_a + \theta_b)T}\right)$$

Experiments have been carried out on a system where $1/\theta_a = 1/\theta_b = 1000$ hours, $\Delta T_{min} = 1$ hour and $T_B = 1000$ hours. The time quantisation TQ for the failure distribution, and consequently time spent awaiting maintenance when in the failed state, was varied from 1 hour to 64 hours. The Repair and Maintenance Policy Module was programmed to only permit component maintenance requests to be allowed. The results for system u_D are shown on Figure 7.2.1a, along with time necessary to simulate 10^6 hours of operation. All values of u_D are very close to the expected. The slight error occurring at high TQ values is due to :

1. Greater difficulty in accurately representing T_B by an integer value.

2. Not making full use of the probability resolution.

3. Difficulty in keeping component test points in phase. Component micro-controllers are responsible for the updating of the block replacement counters, C_B. Updating not only depends on particular state transitions but also time quantisation values for individual states, a factor not fully catered for. Future development of the micro-controller firmware may include changes improving simulation results at high TQ. The simulation time was restricted to 10^6 hours to prevent phase shifts becoming too large.

These problems pose no serious problem to Simulator operation, as the error in u_D amounted to 1.8% with TQ=64 hours. The time required by each experiment fell, as expected, with increasing TQ values. A noticeable 'flattening' of the result curve appears at TQ=16 hours. This is explained by a reduced probability of selecting a high TQ value for model up-date when high TQ values are in use.

The calculation of system unavailability is simpler for a series system than for a parallel system, due to the resultant system failure distribution being characterised by a hazard rate that is the sum of the individual hazard rates of each component [41]. Examining the solution for u_D above, it can be seen that $\theta_x=\theta_a+\theta_b$. Using this theory the above system can be represented by a single component having an exponential failure distribution, mean=500 hours. Such an experiment has been carried out and the result is given below:

$$\text{expected system} \quad u_D = 0.568$$

$$\text{single component model} \quad u_D = 0.565$$

Systems containing components with non-exponential failure distributions, can also be represented by a single component. The W_k values necessary for process modelling are calculated from the system failure c.d.f., $P_x(t)=1-(1-P_a(t))(1-P_b(t))$ ... This poses no problem to the initialisation software in the host computer.

7.2.2 Series Systems with Repair

The simplest system repair policy is that of commencing repair on any faulty unit immediately upon failure occurring. The Simulator can be instructed to carry out this policy by programming the Repair Policy Module to respond to all requests. Consider a single component undergoing a failure repair process. If the failure distribution has mean u_f, and the repair distribution has mean u_r, then the limiting availability of the component is given by the well known equation:

$$u_A = \frac{u_f}{u_f + u_r}$$

For a system containing m components each undergoing the above process, the limiting system availability is known to be :

$$u_A = \prod_{i=1}^{m} u_{Ai}$$

The Simulator has been tested on a system containing three identical components. All distributions were exponential, and the mean failure and repair times were 1000 hours and 50 hours respectively. Life times of 10^7 hours were modelled, with ΔT_{min}=1 hour, employing a range of failure distribution time quantisations (TQ) from 1 hour to 128 hours. The repair distribution time quantisation was constant at 1 hour. Results for system unavailability and simulation time required are shown on Figure 7.2.2a. The values obtained for u_D are acceptable for all TQ. However a slight error (3.5%) appears at TQ=128 hours. This may be due to :

1. Not making full use of the probability resolution

2. Cross-correlation effects apparent when TQ exceeds the mean repair time. When a component failure occurs, the system minimum time quantisation is 1 hour and remains so until all repair is complete. Further component failures are not possible until a time increment reaches the failure distribution TQ value. Expressed another way, components are unlikely to fail during the repair process of another component.

As expected, the simulation time required has fallen as TQ is increased. The usual flattening of the result curve is present, due to reduced probability of up-dating the model with high TQ values when high TQ values are in use.

Systems containing components in series can be represented by a single component. As discussed in Section 7.2.1, the hazard rate for the system failure distribution is the sum of the individual component hazard rates. For the above three-component system, the system failure c.d.f. is given by $1-e^{-3\theta t}$, that is exponential with a mean value of

333.3 hours. The repair distribution is not exponential for a series system but can be modelled by an exponential of equivalent mean. The mean repair time can be obtained by observing the actual three-component system. This was done during an experiment of 10 runs each 10^6 hours long. A shorter life time was chosen to prevent overflow of the 12-bit event counters within the Statistical Gathering Module. The mean values for system repair and failure times are given below.

mean repair time = 54.28 hours

mean failure time = 344.6 hours

Clearly the failure time was not the expected 333.3 hours, which is used to describe the single component failure distribution. The mean repair time observed was used to complete the single component description. Experimental results for the representative component u_D are given below along with the expected system u_D.

expected system u_D = 0.1362

single component model u_D = 0.1361

7.2.3 Series System with Staggered Testing

Where components within a series system are undergoing no repair, but receiving periodic replacement, system reliability can be reduced if replacement time between individual components is staggered. Unlike the non-repairable systems considered in Section 7.2.1, components within a system are then no longer of the same age.

Consider a two-component system which undergoes staggered replacement. Component A is initially new and is replaced at time T. Component B starts new at time kT ($0 \leq k < 1$) and is replaced at periods of T. As in Section 7.2.1 system unavailability is calculated from the system failure distribution c.d.f., which now changes at time kT. That

is:

$$u_D = \frac{1}{KT}\int_0^{kT} 1 - [1-P_a(t)][1-P_b(t+(1-kT))]$$

$$+ \frac{1}{(1-k)T}\int_{kT}^{T} 1 - [1-P_a(t)][1-P_b(t-kT)]$$

This is determined from $P_x(t)=1-P_x$ and $P_x=P_a P_b$.

If $P_a(t)$ and $P_b(t)$ are both exponential of mean $1/\theta$ then the above equation reduces to:

$$u_D = 1-\frac{1}{2\theta T}[(1-e^{-\theta t})(e^{-k\theta T}+e^{-(1-k)\theta T})]$$

Differentiating u_D with respect to k and equating to zero gives a worst case value k=1/2. This is as expected and it can be seen that k=1/2 is the critical value for any system containing two identical components regardless of there failure distribution. When k=1/2 and $P_a(t)=P_b(t)$ the system $P_x(t)$ is the same over both test intervals $0\rightarrow kT$ and $kT\rightarrow T$, enabling the system to be represented by a single component. In such case the system hazard rate is calculated from:

$$P_x(t) = 1 - [1-P(t)][1-P(t+T/2]$$

where $P_x(0) = P(T/2)$

In the case of the exponential distribution this reduces to:

$$P_x(t) = 1 - e^{-2\theta t}.e^{\frac{-\theta T}{2}}$$

That is, a failure process characterised (as in the case of the unstaggered replacement) by a hazard rate the sum of the individual component hazard rates, and suffers an additional initial probability of failure $1-e^{\frac{-\theta T}{2}}$. This initial probability of failure corresponds to replacement of a component not rendering the system operable. The system unavailability can now be calculated by integrating $P_x(t)$ over a

sub-interval.

$$u_D = \frac{2}{T}\int_0^{T/2} 1 - e^{-2\Theta t} e^{-\frac{\Theta T}{2}}\, dt$$

$$= 1 - \frac{e^{-\Theta T/2}(1-e^{-\Theta T})}{\Theta T}$$

Consider a system of two components described by exponential failure distributions with $1/\Theta$=1000 hours. Critical staggered replacement occurs for each component at intervals of 1000 hours. A simulation has been carried out over 10^6 hours of operation, all time quantisations being $TQ = \Delta T_{min}$=1 hour. Simultaneously an experiment was carried out on a single component programmed to represent the system with a mean failure time $1/\Theta$=500 hours and start up failure rate = 39.3%. The results are shown below

expected	u_D = 0.617
system model	u_D = 0.611
single component model	u_D = 0.621

The slightly increased single component u_D is possibly due to considering the start-up failure state as a non operational state.

A single component representation of the system could have been tackled in other ways, resulting in the desired u_D value. However in other parameters a difference would appear. Consider an alternative component programmed with T_B=500 hours and an exponential failure distribution such that u_D was correct over the interval T_B. Maintenance requests would be issued at the expected frequency but the distribution of down time and up time would not model the actual system, partly due to not including the possibility of properly scheduled maintenance leaving the system inoperable at time T_B. For comparison, an experiment comprising this alternative system representation has been carried out. The results are presented below.

	mean down time	mean up time	u_D
two-component model	502 hours	318 hours	0.611
Single component model	526	317	0.622
Poor alternative	304	198	0.614

7.3.1 Parallel System with No Repair

A parallel system of m components implies that only one of the m components need be operational for the system to be successful. Alternatively all components must be faulty to bring the system down. Each component's behaviour is described by a probability distribution of time to failure. The whole system is tested at regular intervals and components are replaced by new components regardless of their operating condition. If the probability of component n being operational at time t is given by Pn then the probability of the system being operational at time t is given by:

$$P_x = 1 - \prod_{i=1}^{m} P_i$$

where $\bar{P}_n = P_n(t)$, the failure c.d.f. of component n

As in the section on series systems with no repair, consider the case of a two component system. Where components are identical having an exponential failure distribution, the mean $= 1/\theta$. System unavailability is given by:

$$u_D = \frac{1}{T} \int_0^T (1 - e^{-\theta t})^2 \, dt$$

where $T = T_B$, periodic replacement time.

The solution is:

$$u_D = 1 - \frac{1}{2\theta T} (3 + e^{-2\theta T} - 4e^{-\theta T})$$

Unlike series component systems parallel systems are not described by a failure distribution function which has a hazard rate the sum of the individual component hazard rates. Limiting system component distributions to the exponential does not result in an exponential system failure distribution. If it is desired to represent the system with a single component then the problem of non-monotonic distribution modelling presents no difficulty. The necessary W_k values can be calculated from the system c.d.f. equation above.

A system containing two components each described by an exponential failure distribution mean 1000 hours has been investigated. The time between component replacement was 1000 hours and failure distribution TQ values were varied from 1 hour to 64 hours. System life times of 10^5 hours were programmed to be repeated fifty times permitting results to be averaged. Limiting the time to 10^5 hours prevents overflow of the Statistical Gathering Module event counters occurring. The values obtained for u_D are shown in Figure 7.3.1a. The increasing error at high TQ values has been previously explained for the series system case, in Section 7.2.2. For the particular case TQ=1 hour, the results are presented below.

u_D	mean down-time	mean up-time
0.161	227.9 hours	1181.0 hours

expected u_D value =0.168

Note that mean down time and mean up time do not add to make 1000 hours. This is explained by the difficulty of maintaining simultaneous replacement.

A single component representation has been carried out. The time quantisation was increased to 32 hours to allow the non-exponential system distribution to be fully stored in the component memory. The results are given below.

u_D	mean down-time	mean up-time
0.169	173.hours	849.7hours

The mean times still do not add to 1000 hours, but this has a new explanation. The quantisation of the replacement time T_B is now poorer at 32 hours. When T_B is not a binary multiple there is always an error in its approximation which increases with increasing TQ values.

7.3.2 Parallel Systems with Repair

As with any parallel system, each component within the system is required to be operational for system success. The simplest system repair policy is that of commencing repair on faulty units immediately upon failure occurring. Programming the Simulator to carry out this policy has been described in the section on series systems with repair. As pointed out for the series component arrangement, individual component unavailability is given by:

$$u_D = \frac{u_r}{u_f + u_r}$$

where u_f and u_r are mean failure and repair times respectively.
For a system containing m components each undergoing a failure repair process, the limiting system unavailability is known to be:

$$u_D = \prod_{i=1}^{m} u_{Di}$$

The Simulator has been tested for a system containing three identical components. The failure and repair distributions were exponential with mean values of 1000 hours and 50 hours respectively. System life times of 10^7 hours were modelled with ΔT_{min}=1 hour, employing a range of failure distribution time quantisation TQ from 1 hour to 128 hours. Repair distribution quantisation was constant at 1 hour. The results produced for system unavailability are shown on Figure 7.3.2a. The values obtained for u_D are acceptable, but as with the series system previously examined, errors occur at high TQ values. The reasons for this error are the same as for for the series case. However it is noticable that the effect of high TQ values is now more damaging. This is understandable considering the much lower probability of system failure for the parallel case. Cross-correlation occurring at high TQ values has two effects:

1. It reduces the likelihood of component failure during the repair process of another component, as explained in the series system case.

2. It increases the likelihood of simultaneous component failure.

It is this second reason given that causes the system unavailability to increase at high TQ values.

Systems containing components in parallel can be represented by a single component. The previous section showed that the system hazard rate is not the sum of each component hazard rate. However by a deduction similar to that of considering the hazard rate for the failure process of components in series it can be seen that the hazard rate for the repair process, in a parallel system, is the sum of the individual component hazard rates. For the above system, the repair c.d.f. is given by $1- e^{-3\theta t}$, θ=1/50 hours. The mean failure time is obtained by

observing the actual 3-component system. The values obtained are given below.

mean repair time = 17.8 hours

mean failure time = 164,629 hours

The repair time observed was not the expected 16.7 hours, to be used to describe the single component repair distribution. The failure time mean is very large and requires a large quantisation time if accurate modelling is to be achieved. A value of TQ=128 hours was chosen for an experiment carried out over a system life time of 10^7 hours. The results are below:

3-component model	$u_D = 1.081$
single component model	$u_D = 0.963$
expected	$u_D = 1.079$

7.3.3 Parallel Systems with Staggered Replacement

Where components within a parallel system are undergoing no repair, but receiving periodic replacement, the system reliability can be improved if the replacement times between individual components are staggered. Staggered replacement has already been considered for series systems. The example 2-component system given in Section 7.2.3 is now dealt with for a parallel configuration. As for non-staggered testing, the probability of system failure is given by:

$$\overline{P}_x = \overline{P}_a \overline{P}_b \qquad \text{,where } \overline{P}_a = \overline{P}_b = 1 - e^{-\Theta t}$$

This equation can be used to determine the system availability by integrating over the intervals $0 \rightarrow kT$ and $kT \rightarrow T$. The solution obtained would be complex. The problem is therefore simplified by making a few likely assumptions. Firstly, all components are considered identical. This results in the system availability being optimal for $k=1/2$, for

which availability is the same over each interval. The system mean unavailability is then given by:

$$u_D = \frac{1}{kT} \int_0^{kT} [1-e^{-\Theta t}][1-e^{-\Theta(t+\frac{1}{2}T)}]\, dt$$

$$= 1 + \frac{(1-e^{-\Theta T})}{\Theta T} \left(e^{\frac{-\Theta T}{2}} - 2\right)$$

An experiment was carried out for the parallel system, in the same manner as for the serial system ($1/\Theta$=1000 hours, TQ=1 hour, T_B=1000 hours). The results obtained are given below:

expected u_D = 0.1192

system u_D = 0.1187

A single component representation of the system is possible by applying the rules given in the section on parallel systems with no repair. For the example system, the single component failure distribution would be non-exponential. This would present no difficulty to the host initialisation routines. Difficulty would occur in modelling the probability of selecting a component for maintenance which resulted in the system remaining operational. (Noting that for the series case, a start-up failure risk was added to represent the chance of replacement not rendering the system operational). This problem cannot be overcome with the current Simulator design. Therefore it should be expected that the distribution of up and down times cannot accurately model the 2-component system. An experiment on a single component system has been carried out, using a failure distribution time TQ=16 hours. The results are given below.

	u_D	mean up time	mean down time
2-component model	0.1187	1155.5 hours	149.3 hours
single component	0.1176	445.6 hours	66.7 hours

7.4 Majority Voting System

A majority voting system is one consisting of m components arranged such that any n or more components are required to be working for the system to be operational (n<m). In examining such systems only non-reparable systems are studied. Experience of modelling series and parallel systems indicates that no difficulty should occur with simple repair policy modelling.

Consider a 3-component system A, B and C. The probability of a 2-out-of-3 system being operational is:

$$P = P_a P_b + P_b P_c - P_c P_a - 2P_a P_b P_c$$

Assuming all three components are identical then the probability of system failure becomes:

$$\overline{P}_x = 1 - 3P^2 + 2P^3$$

Integrating $\overline{P}_x$ over the operation time T gives the system mean unavailability:

$$u_D = \frac{1}{T} \int_0^T 1 - 3P^2 + 2P^3 \, dt$$

If the component failure distribution is the exponential, $P = e^{-\theta t}$, then:

$$u_D = 1 - \frac{3}{2\theta T}(1 - e^{-2\theta T}) + \frac{2}{3\theta T}(1 - e^{-3\theta T})$$

Staggered replacement can be performed on the components. Consider the case where component A has just been replaced and components are replaced in order A, B then C. It can be seen for an optimal replacement policy that:

$$P_a = e^{-\theta t} \quad , P_b = e^{-\theta(t + \frac{2}{3}T)} \quad , P_c = e^{-\theta(t + \frac{1}{3}T)}$$

Mean unavailability can now be calculated for the new replacement policy.

$$u_D = \frac{3}{T}\int_0^{T/3} \bar{P}_x \, dt$$

$$= 1 - \frac{3}{2\Theta T}(1-e^{-\Theta T 2/3})(e^{-\Theta T 1/3}+e^{-\Theta T 2/3}+e^{-\Theta T})+\frac{2}{\Theta T}e^{-\Theta T}(1-e^{-\Theta T})$$

Majority voting systems can be represented by a single component. The system failure process cannot be described by an exponential distribution, but the necessary W_k values are simply calculated from the system failure c.d.f.. Staggered replacement presents difficulty in the same manner as parallel component systems. The probability of selecting a component for maintenance not rendering the system inoperable cannot be modelled. A staggered policy in majority voting systems also contains the series system characteristic of maintenance possibly not rendering the system operable. However this can be modelled by the start-up failure feature. The actual value of SUF is given by the system c.d.f. at t=0.

Experiments have been performed on a 2-out-of-3 system. The components were identical with an exponential failure distribution whose mean was 1000 hours. Periodic replacement occurred at intervals of 1000 hours and time quantisation was 1h. System life times of 10^6 hours were programmed to be simulated ten times, allowing the results to be averaged. The results obtained for both staggered and non-staggered replacement are given below:

	simultaneous replacement	stagered replacement
expected u_D	0.336	0.299
3-component system	0.327	0.292
single component (TQ=8 hours)	0.333	0.303 (SUF=13.8%)

7.5 Standby Redundant Systems

A standby redundant system is essentially a parallel component arrangement where not all components are working at the same time. Certain components have a redundant inactive state and are not made operational until the failure of another component. The general purpose micro-controller within each component has the facility to model such passive standby operation.

Consider the case of a 2-component system. The main element (component) is modelled by a Component Board at position 'm' in the Simulator. The standby component is positioned at m+1. Whenever the main element is not operational this second element becomes active, and remains so until the main element is repaired.

It cannot be assumed that the change-over process is perfect. It is more likely that some probability can be associated with successful standby change-over. The change-over process can be viewed in two ways. Firstly change-over is accomplished by a change-over component, which suffers a probability of failure during its life-time. This results in the probability of a successful change-over reducing, the longer the standby component waits before coming into operation. Modelling such behaviour requires an additional component to represent the switch-over element. However this can be avoided by including the change-over component failure characteristic with the standby-failure characteristic of the standby component. The resultant distribution for failure at change-over cannot be exactly described by an exponential function, as the change-over and standby components form an OR function to failure at change-over. To enable modelling to take place, an exponential approximation is necessary. It is considered unlikely to introduce any significant error in any result.

The second method by which change-over failure can be represented by is constant failure probability. Failure now does not depend on the waiting time before change-over is required. This characteristic is modelled by a start-up failure (SUF) rate for the standby component.

For the purpose of verifying the Simulator operation, a two-component non-repairable system is examined. The main component is denoted by A and the standby by B. The probability of the system being operational is given by :

$$P_x = P_a + \bar{P}_a P_b$$

In terms of distribution functions;

$$P_x(t) = 1 - P_x = P_{ao}(t) - \bar{P}_{bs} \int_0^t \bar{P}_{bd}(u) . \bar{P}_{bo}(t-u) . f_{ao}(u)\, du$$

where subscripts o indicate operational state

d dormant or standby state

s start-up failure probability

The convolution integral, formed by the sum of two random variables, necessitates the use of Laplace transforms before any algebraic manipulation can produce a solution to $P_x(t)$. Considering all distributions to be exponential and $L[f_x(t)] = F_x(s)$ then :

$$\frac{1}{S} F_x(S) = \frac{1}{S} \frac{\theta_{ao}}{\theta_{ao}+S} - \frac{\bar{P}_{bs}\theta_{ao}}{S+(\theta_{ao}+\theta_{bd})} \frac{1}{S+\theta_{bo}}$$

now applying tables of inverse transforms

$$P_x(t) = 1 - e^{\theta_{ao} t} - \frac{\bar{P}_{bs}\theta_{ao}}{\theta_{ao}+\theta_{bd}-\theta_{bo}} \left(e^{\theta_{bo} t} - e^{-(\theta_{ao}+\theta_{bd})t}\right)$$

The system mean unavailability can be found by applying:

$$u_D = \frac{1}{T} \int_0^T P(t)\, dt$$

which yields:

$$u_D = 1 - \frac{(1-e^{\theta_{ao}T})}{\theta_{ao}T} + k\left[\frac{1-e^{(\theta_{ao}+\theta_{bd})T}}{(\theta_{ao}+\theta_{bd})T} - \frac{1-e^{\theta_{bo}T}}{\theta_{bo}T}\right]$$

where $$k = \frac{\bar{P}_{bs}\theta_{ao}}{\theta_{ao}+\theta_{bd}-\theta_{bo}}$$

Two standby system arrangements have been modelled. Each system contained two components whose parameters are below.

	$1/\theta_{ao}$	$1/\theta_{bo}$	$1/\theta_{bd}$	P_{bs}	T_B
First system	2000	1250	10000	0	1000
Second system	2000	1250	0	5%	1000

The mean values are given in hours. The Simulator ΔT_{min} was 1 hour and each system was programmed to be modelled one hundred times for 10^5 hours of operation. The simulations were then repeated with failure distribution quantisations TQ varied from 1 hour to 32 hours.

For the first system, the probability of successful change-over reduces with the delay before change-over is requested. It should be noted that the minimum time before a standby component can respond is TQ. Mathematically the standby component 'responds' immediately. To allow a comparison of results to be carried out, an adjustment to the measured down-time is therefore required

down-time = measured down-time - maintenance time
- delay in response

$$= \text{measured time} - \frac{10^5}{1000}[1+(1-e^{-\theta_{ao}T_B})TQ]$$

The results are presented on Figure 7.5a.

The second system has a constant change-over failure probability. For similar reasons to the above, the measured down time also requires adjustment.

$$\text{down-time} = \text{measured down-time} - \text{maintenance time} - \text{delay in response} - \text{time in SUF state}$$

$$= \text{measured time} - \frac{10^5}{1000}\left[1+(1-e^{-\Theta_{ao} T_B})(TQ+1)\right]$$

The results are presented in Figure 7.5b.

Both result curves show a dip at TQ=1 hour, due mainly from selecting a TQ value too small for accurate modelling of the main component failure distribution. The usual fall off in accuracy is also present at high TQ values. With TQ set to the practical value of 8 hours the error in u_D are 4.5% and 1.6% for systems 1 and 2 respectively.

7.6 Consideration of Common Mode Failure

A significant factor of the reliability of complex systems is common mode failures (CMF). The consideration of CMF for systems in which high reliability is obtained by application of redundancy techniques is most important. It is not satisfactory to consider parallel sub-system arrangements as statistically independent. Investigations [42] have shown that CMF can degrade a system's reliability by one or two orders of magnitude. Consequently, any simulator intended as a general purpose modelling tool must be capable of CMF modelling.

CMF can be dealt with by the Simulator in so far as once a failure mechanism has been identified, its affects on the system performance can be investigated and quantified. Within this section, CMF shall be considered as the failure of a single component, or group of components, which result in the loss of separate channels of a redundant system.

Two of the principal types of CMF are hereafter considered for inclusion in multi-component systems.

Recurring maintenance errors can be expected to be, in the long term, the predominant form of CMF. Maintenance errors arise from human factor involvement during such tasks as component testing, calibration and replacement. Statistical observation has indicated a task error rate of 10^{-2} per sub-system-year [42]. This value is believed to vary with certain system conditions although the exact nature of which is not clearly known.

Maintenance-originating sub-system failures can be modelled by including a common component in the system network specification. The common component is then specified to have a start-up failure rate (SUF) immediately following maintenance, representing CMF. Where a sub-system can be represented by a single component, then the system success tree may not need expanding, as the SUF can be modelled by the single component representation.

Random errors account for about 30% of CMF [42]. Such failures are caused by catastrophic events, environmental extremes and operator error. Their inclusion in a system model is via component representation.

A further method also useful in dealing with CMF representation is that of determining system success under CMF. This technique avoids the need for any expansion of the system tree with common mode components and has found use with network analysis techniques where difficulty exists in determining minimal cut-sets when allowing for CMF. The probability of system success is given by:

$$P_x \simeq P_o(1-C_1) + C_1P_1$$

where P_o = probability of system success with no CMF

C_1 = probability of CMF

P_1 = probability of system success with CMF present

Consider a system consisting of two temperature sensors and two pressure sensors arranged such that one temperature sensor and one pressure sensor are required for system success i.e. two parallel trains in series. Assume that the temperature sensors are identical each having an exponential failure distribution, mean $1/\theta_r$=2 years. Assume that the pressure sensors are also identical and are described by an exponential failure distribution with a mean value of $1/\theta_p$=1 year. Assume that the temperature sensors are tested and replaced simultaneously at intervals of twelve months and that the pressure sensors are maintained simultaneously at six monthly intervals. The system unavailability can be calculated from:

$$u_D = \frac{1}{T}\int_0^T \bar{P}_x \, dt \qquad , T=1 \text{ year}$$

where $P_x = 1-(2e^{-\theta_r t}-e^{-2\theta_r t})(2e^{-\theta_p t}-e^{-2\theta_p t})$ from t=0→6 months

$= 1-(2e^{-\theta_r t_e-\theta_r T/2}-e^{-2\theta_r t_e-\theta_r T})(2e^{-\theta_p t}-e^{-2\theta_p t})$ t=6 months→1 year

The solution is $u_D = 0.1120$

The system has been modelled for an operating life time of 438 years, where ΔT_{min}=8.76h (chosen from 8760 hours/year). All exponential distribution time quantisations (TQ) were T_{min}, and this resulted in a simulation time of twelve seconds. Later, each parallel train was represented by a single component with a non-exponential failure distribution. The higher TQ values necessary for memory storage resulted in simulation times of less than 1 second. The statistical gathering probes were positioned to observe both the parallel train and the total system behaviour. The results are presented below.

	System u_D	Single component u_D	expected u_D
temperature sensors	0.057	0.0587	0.0582
pressure sensors	0.0561	0.0587	0.0582
Total system	0.108	0.117	0.112

If information from sensors is arranged so that two cables each carry signals from a pressure sensor and temperature sensor, then a CMF in a single cable would result in a system configuration of two series components. During the above experiment this configuration was monitored to determine the system u_D. The results obtained were:

System u_D (under CMF)	expected
0.375	0.374

If the CMF can occur in each cable with probability $C_1=0.1$ then the system u_D is now given by

$$u_D \simeq 0.117 \times 0.8 + 0.375 \times 0.2$$
$$\simeq 0.169$$

7.7 Conclusion

The simulation results are very close to the theoretically predicted results for all system configurations examined. The time required to model even long system life times was sufficiently short as to allow sensitivity analysis to be performed on component parameters. Where time quantisation TQ values greater than ΔT_{min} have been used, the simulation times are further reduced. Large TQ values (say $TQ>8\,\Delta T_{min}$) do not lead to still greater speed gains due to the reduced probability of their selection for model updating. Certain problems can be associated with the use of large TQ values, viz:

1. Cross-correlation effects between components affects the system statistics. This is most noticeable with systems of high

reliability, due to parallel trains showing less than expected reliability.

2. The rounding of the system parameters to binary multiples becomes less precise e.g. the block replacement time T_B. This only appears significant where $T_B < 10TQ$.

3. Difficulty exists in keeping scheduled maintenance points in phase for multi-component systems. With the current Simulator design, component maintenance is kept in phase by micro-controller update signals. Updating not only depends on particular component state transitions but also on the TQ values used for individual states, a factor not fully catered for. This problem is considered to be the most damaging to the simulation results.

By limiting the TQ values to $TQ < 8\Delta T_{min}$, the above problems are avoided and speed gains in the order of X4 to X8 can be expected. However certain systems, due to their inherent high reliability or use of non-exponential distributions, require TQ values in the order $32\Delta T_{min}$. These systems usually have large T_B parameters and therefore do not suffer from binary rounding problems. The greater TQ values produce the required increased simulation speeds for such systems. In the case of short T_B values, the use of a 12-bit resolution for exponential distributions and feedback for non-exponential distributions can be relied upon to ensure high accuracy. When non-exponential distributions are in use, initialisation routine predictions are useful in selecting TQ values.

For each system examined, single component representation has been developed. This is intended for use where systems initially have more elements than the Simulator has components. In such cases, sub-systems, whose components do not compete for repair facilities, can be selected for reduced component representation. The results obtained indicate that reduction can be successfully applied to many system configurations. Only where sub-systems contain a parallel arrangement with a staggered maintenance policy was difficulty observed. The particular problem encountered cannot be considered limiting as it concerned matching the Simulator model to a mathematical model which itself was unrealistic in practical system terms.

Many reduced systems resulted in more accurate reliability estimates. This is explained by maintenance points being kept more in phase, particularly between components employing different failure distribution time quantisation values

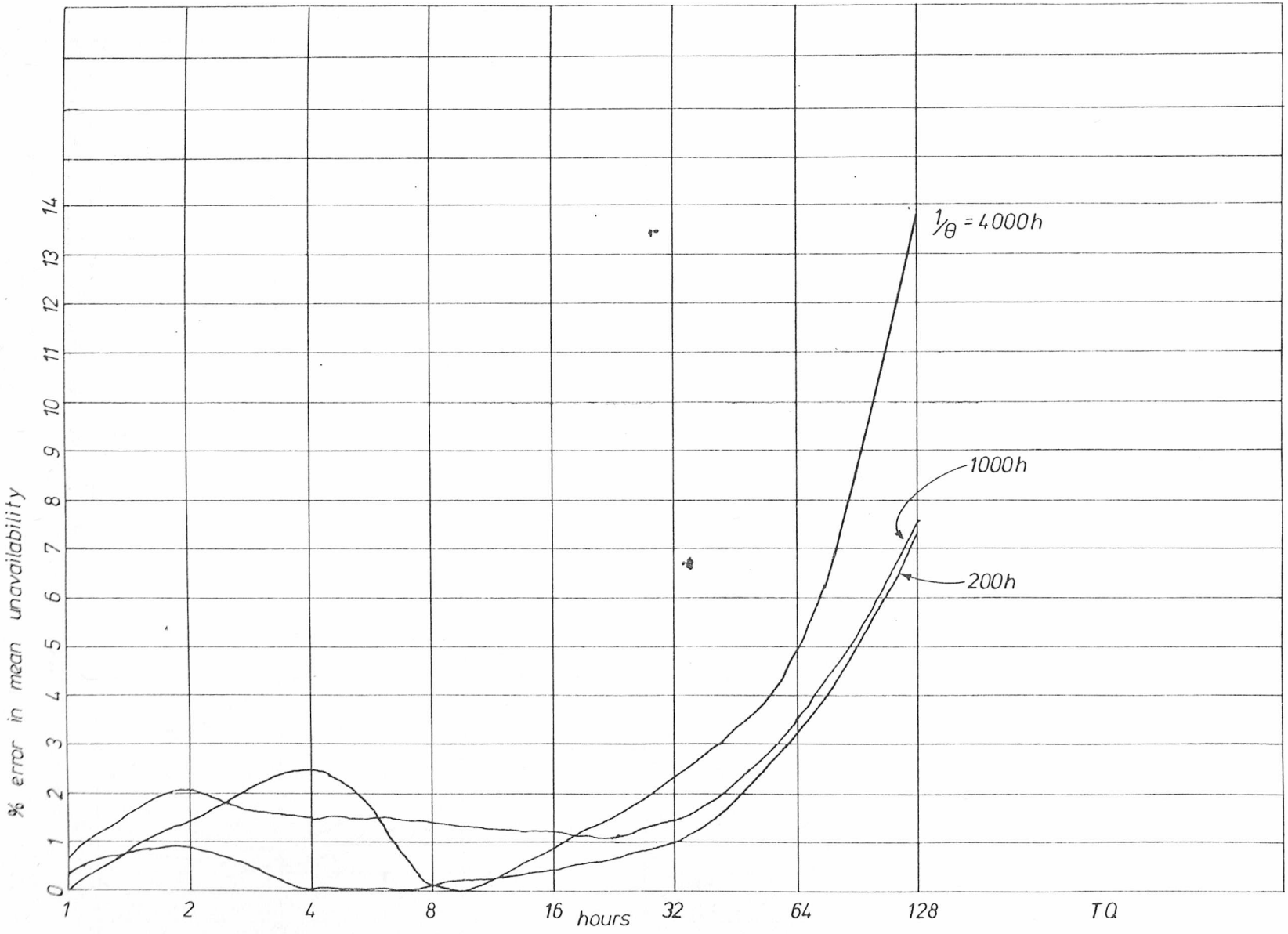


Figure 7.1a Single component with maintenance T_B = 1000h

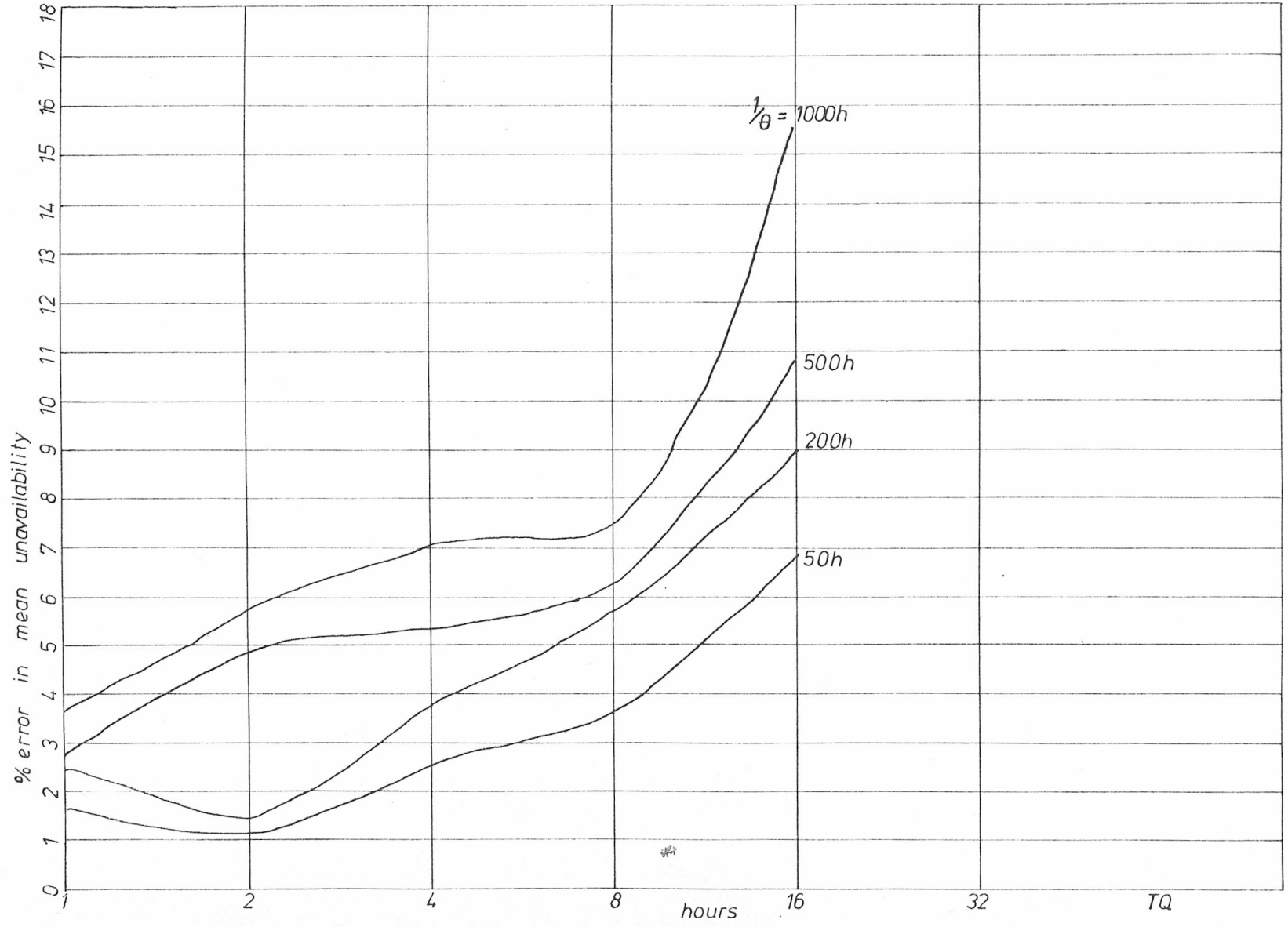


Figure 7·1b Single component with maintenance $T_B = 100h$

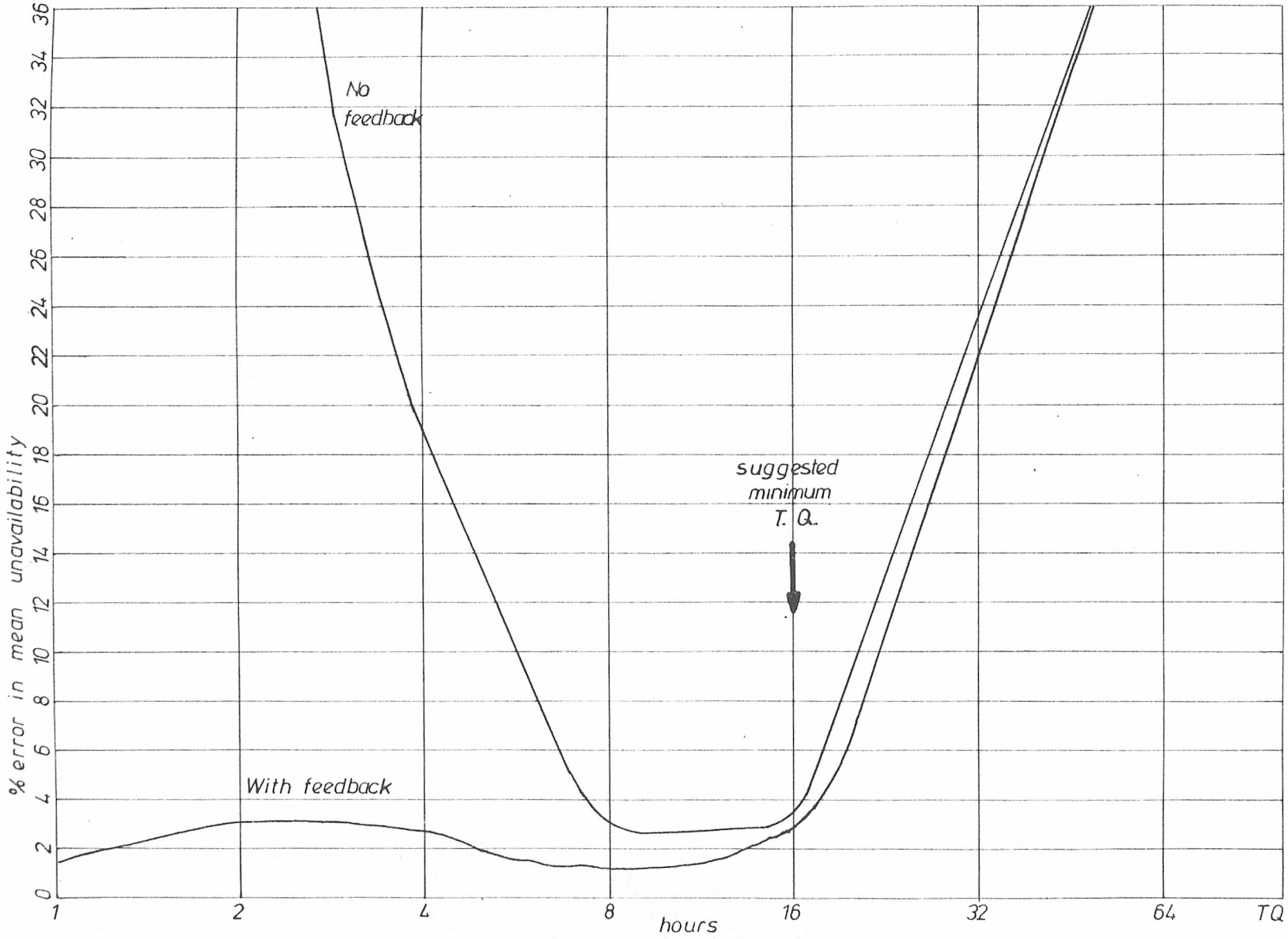


Figure 7·1c Single component spatial Erlang failure distribution maintenance at $T_B = 200h$

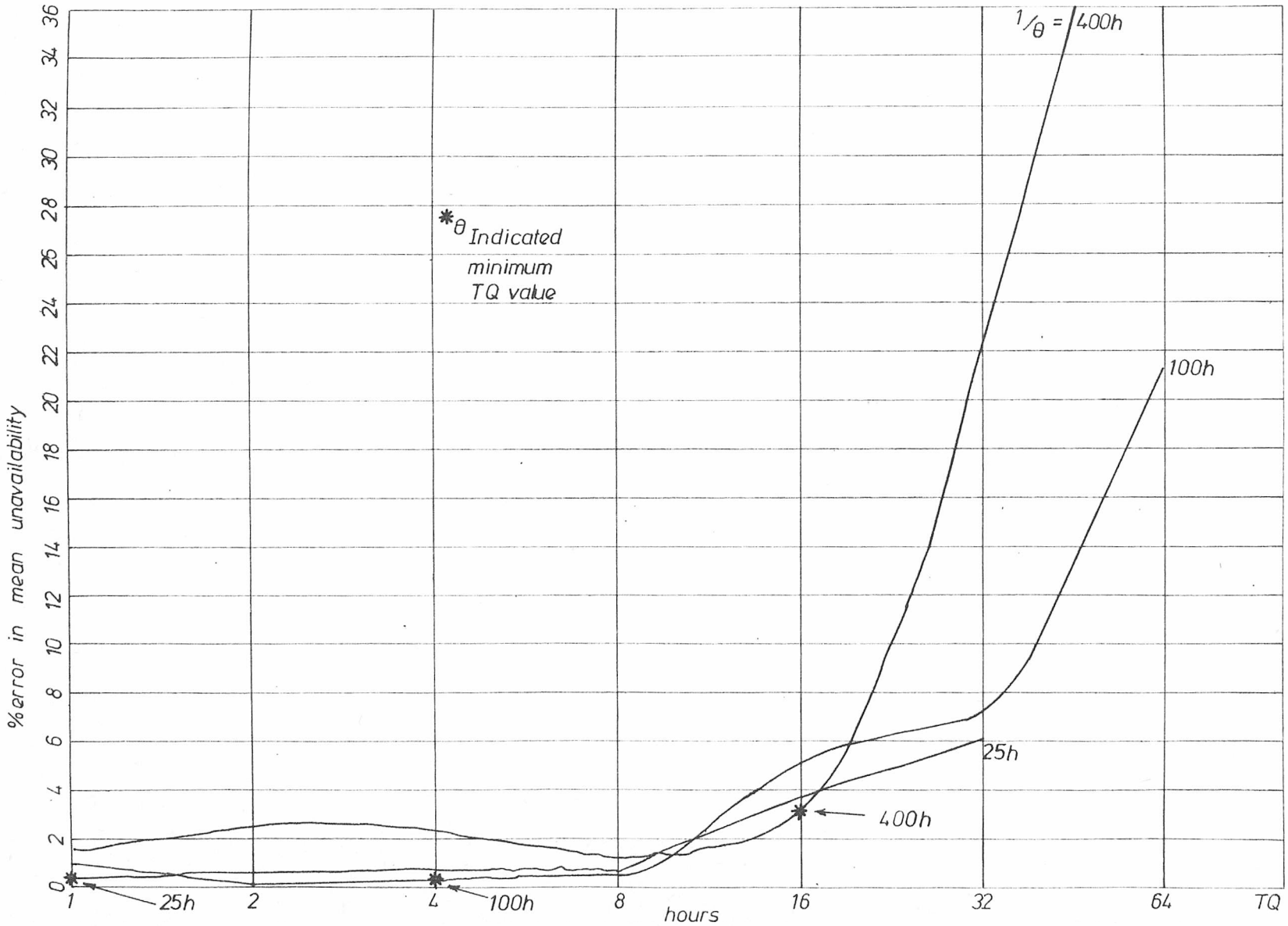


Figure 7·1d Single component with spectral Erlang failure distribution maintenance at T_B = 200h

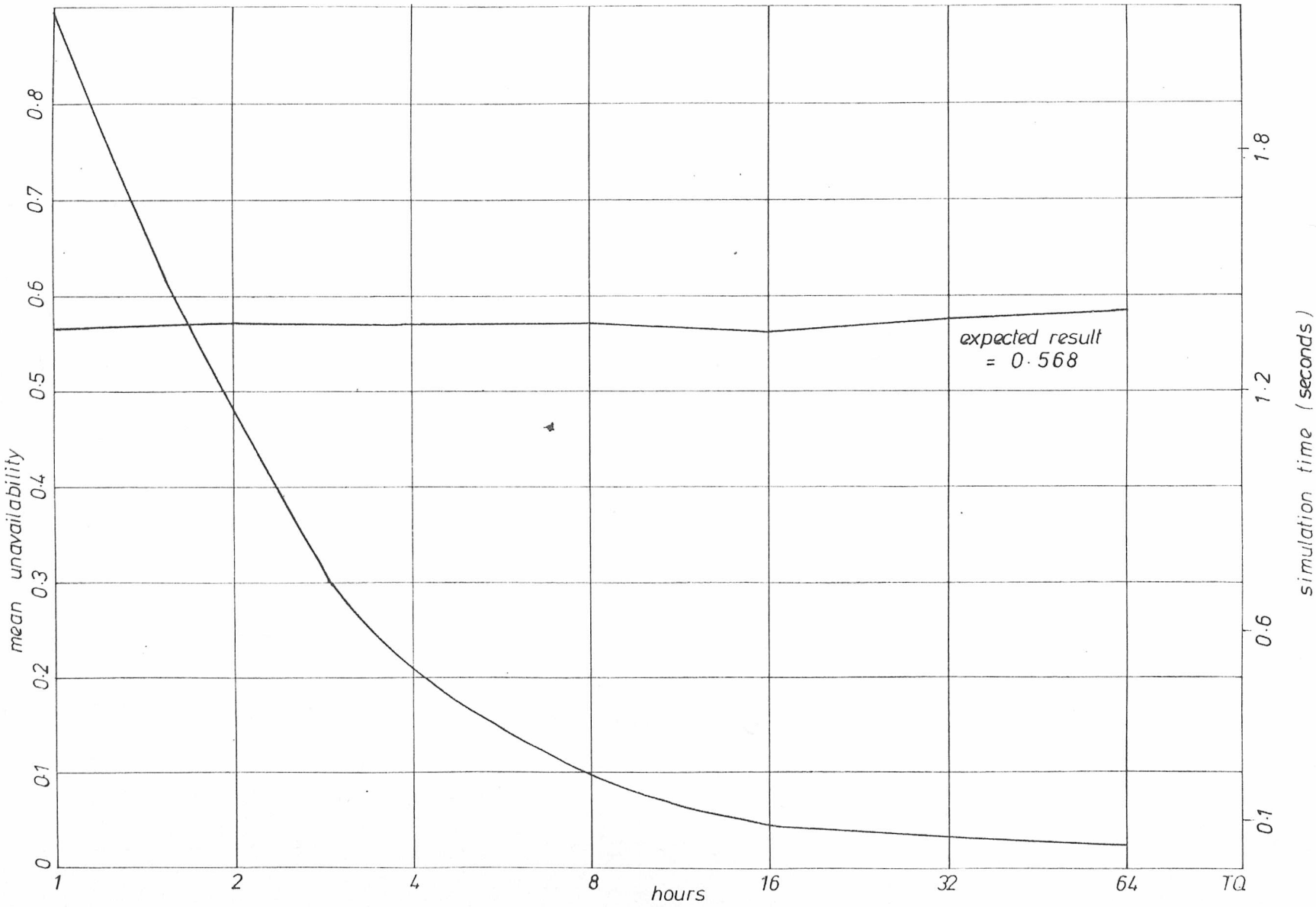


Figure 7.2.1a 2 components in series maintenance at $T_B = 1000$ hours

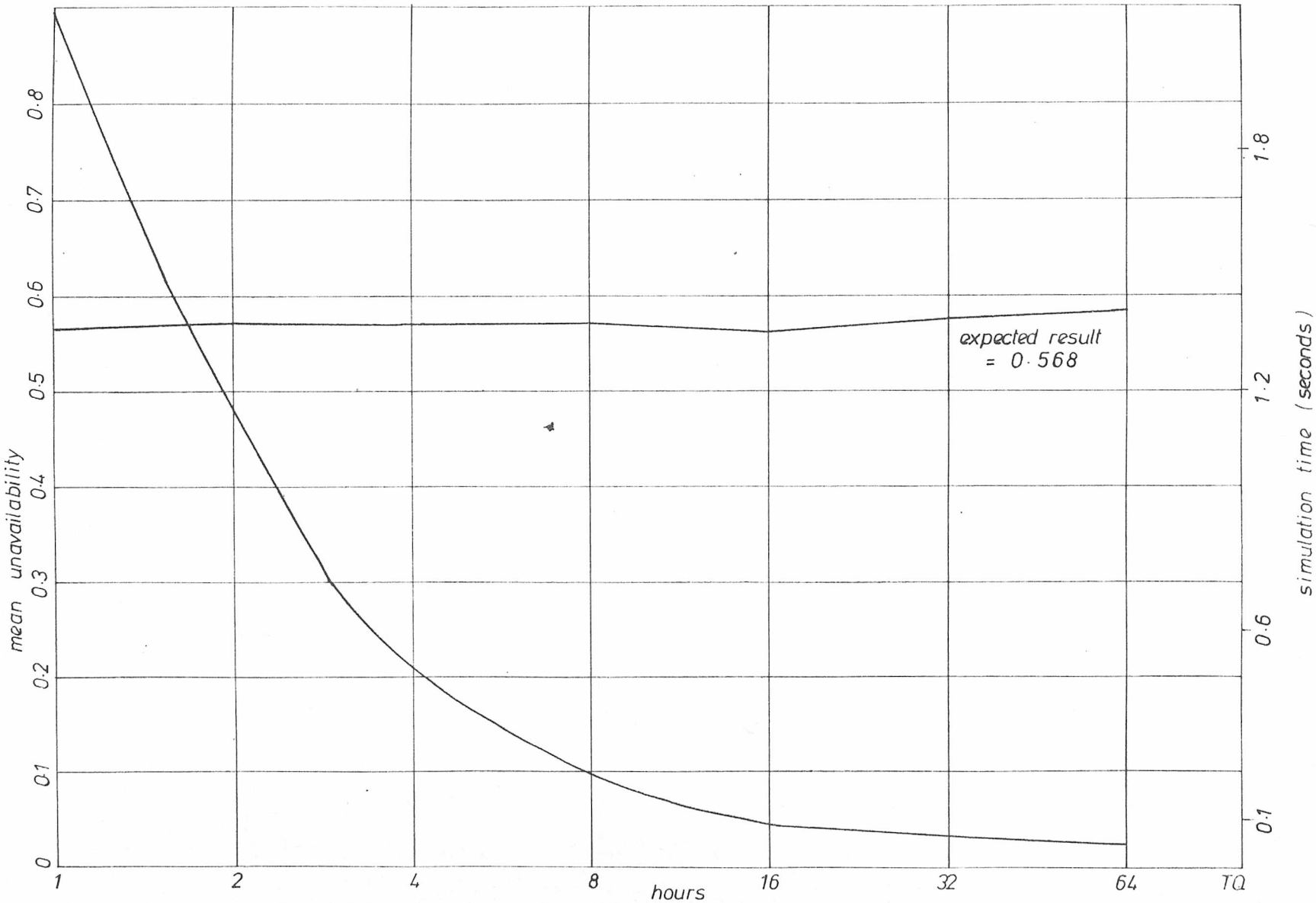


Figure 7.2.1a 2 components in series maintenance at $T_B = 1000$ hours

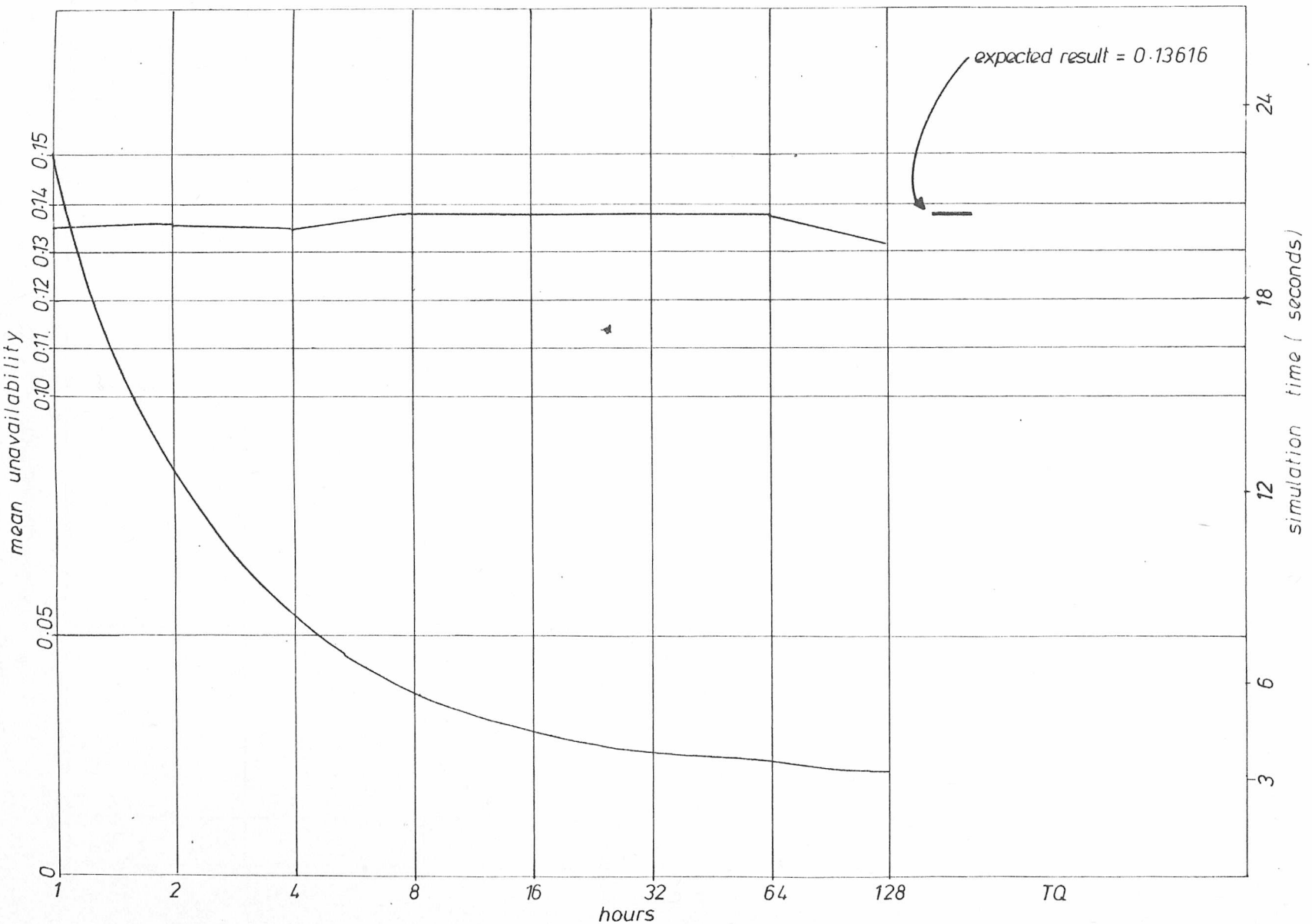


Figure 7·2·2a 3 components in series

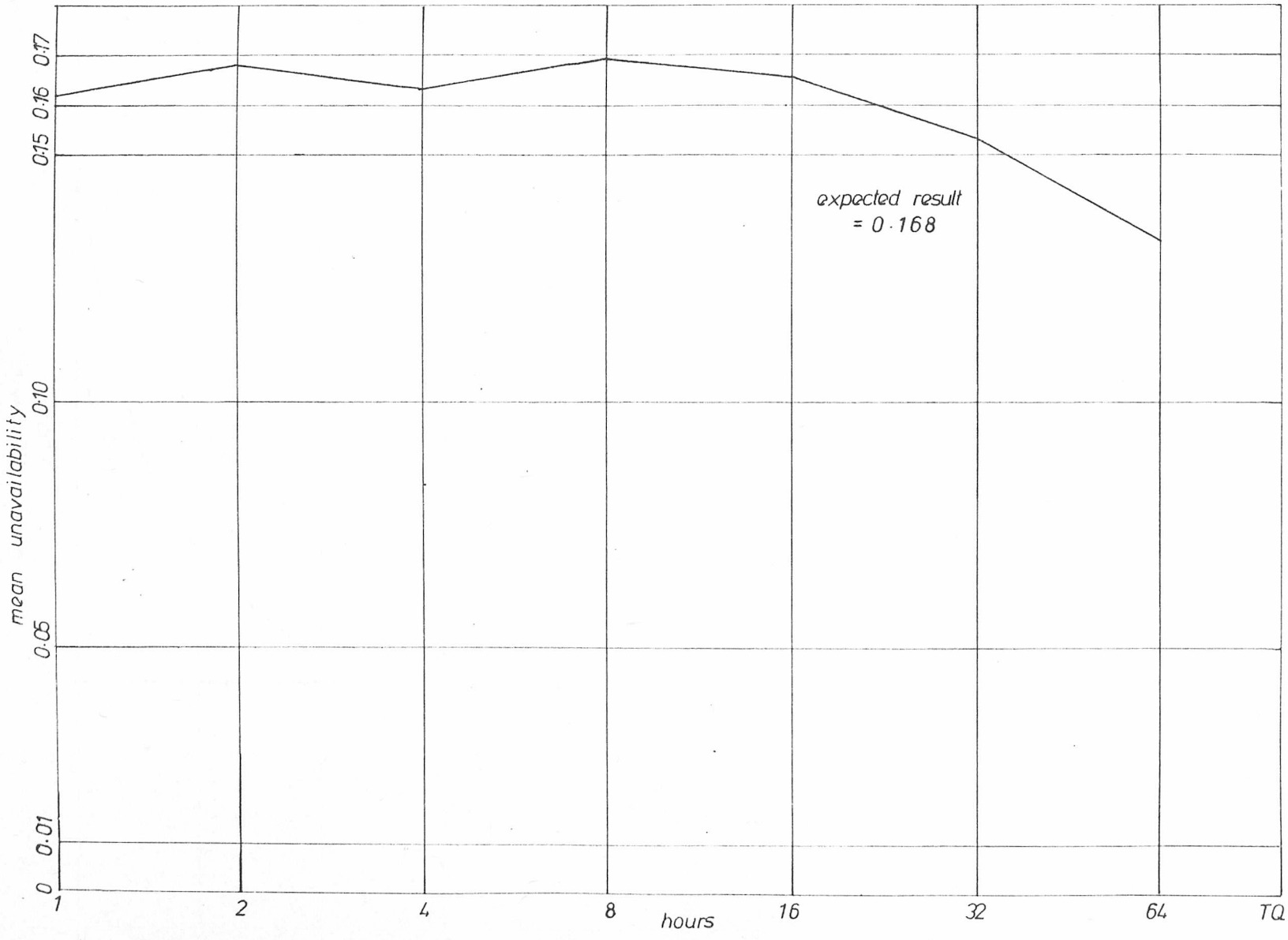


Figure 7·3·1a 2 components in parallel maintenance at T_B = 1000h

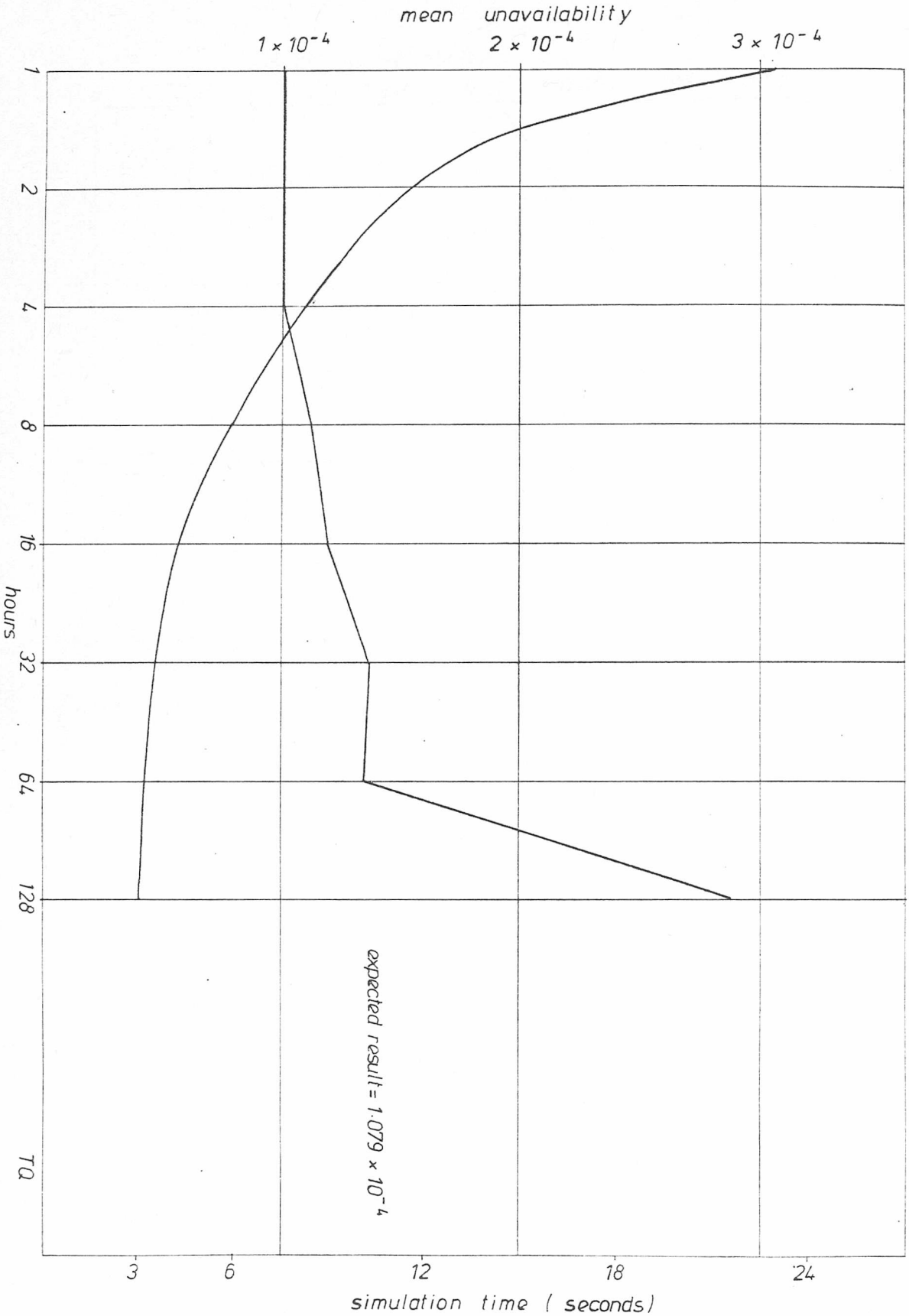


Figure 7.3.2a 3 components in parallel

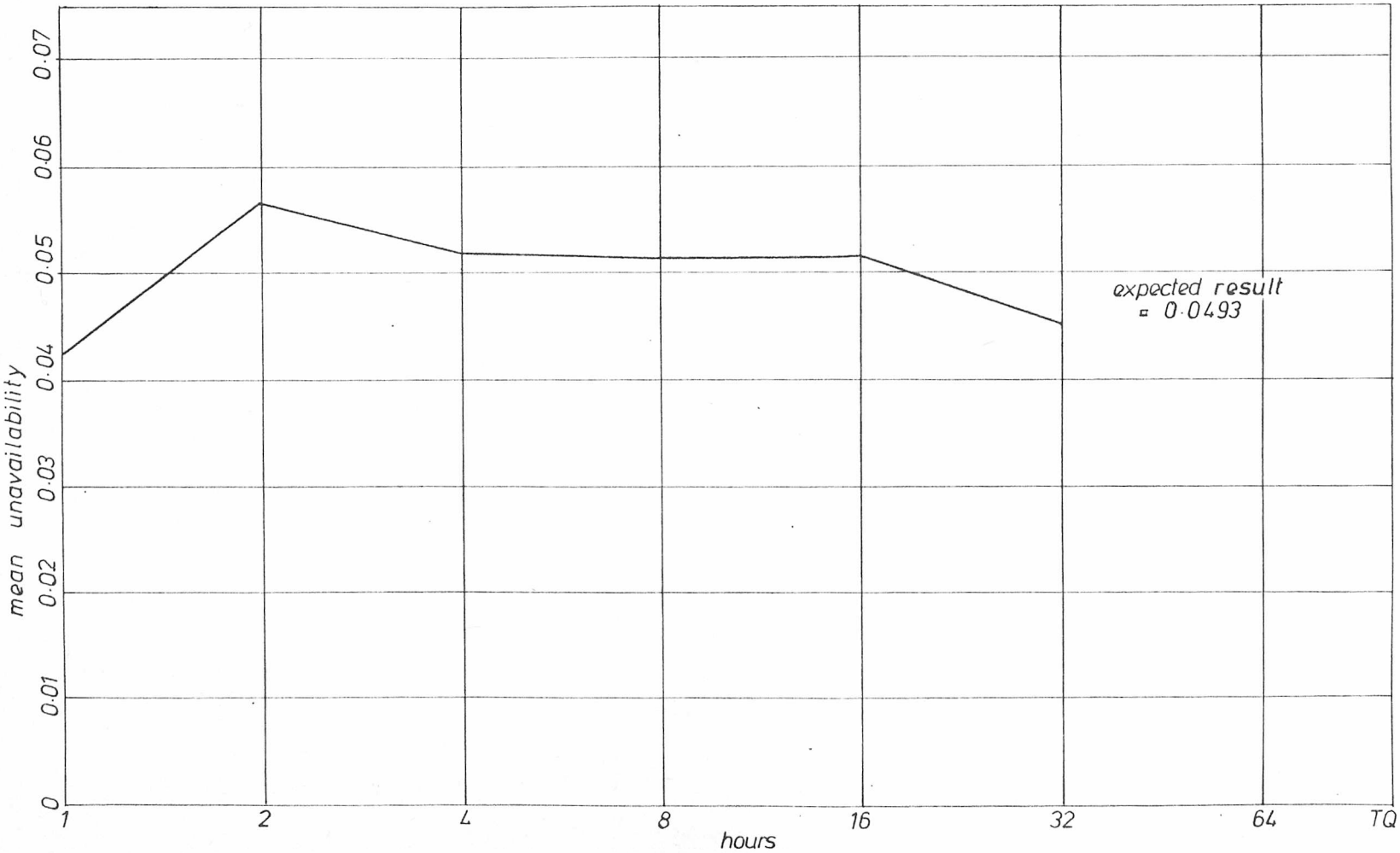


Figure 7·5a 2 component standby system change-over success reducing with delay

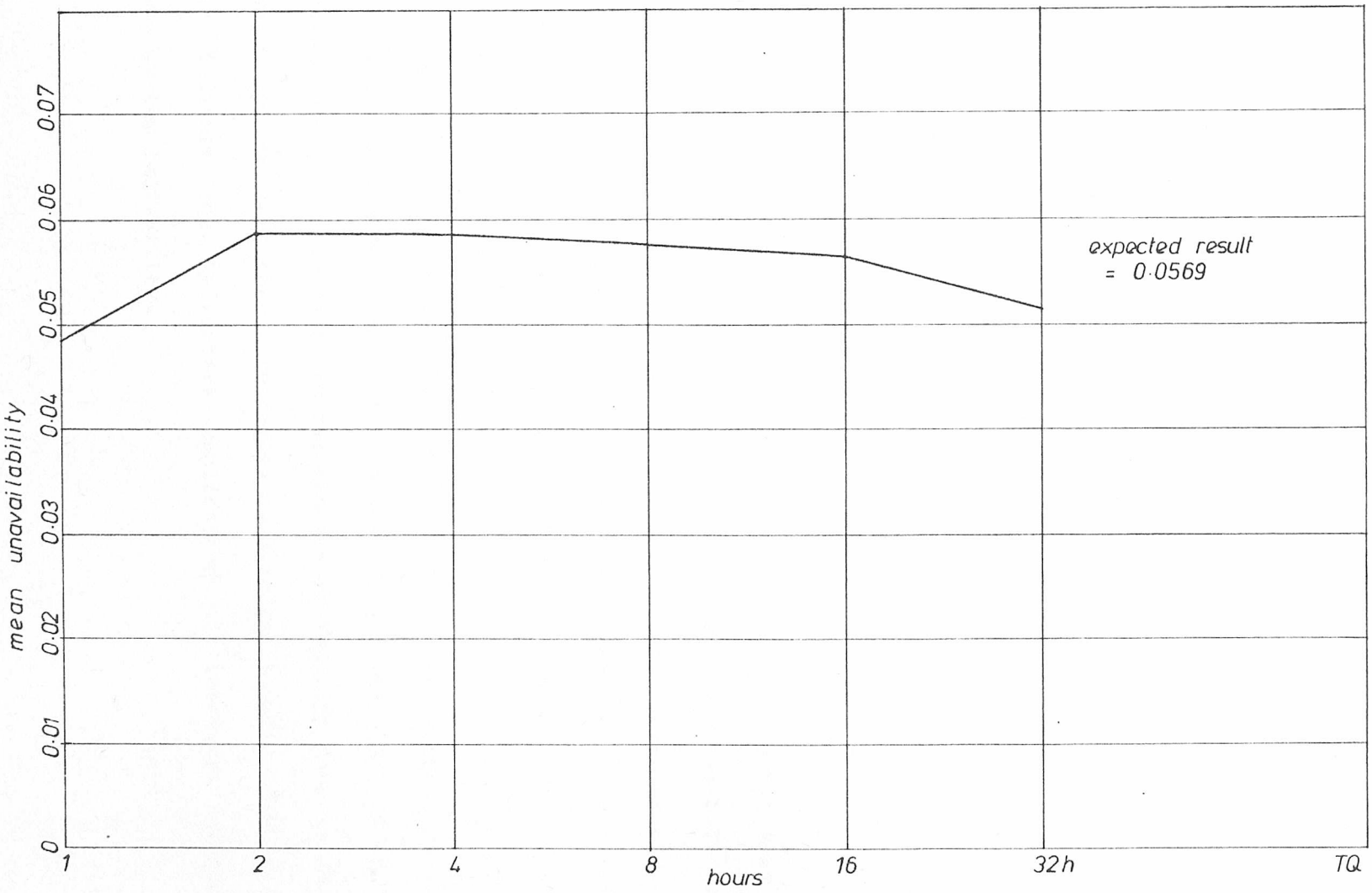


Figure 7·5b 2 component standby system constant switch-over failure

8.0 Introduction

This Chapter is concerned with the application of the Simulator to a selection of realistic system reliability problems, many of which would be unsolvable by analytical means. It will be seen that considerable flexibility has been achieved within the Simulator design, and that simulation times are unrestrictingly short. The universal component micro-instruction code is revealed to make effective use of component hardware, the need to re-programme the code only occurring once to enable minimal repair and economic operation to be dealt with.

Much of the Simulator's modelling ability is demonstrated by application to the example five component system, shown on Figure 8.0a. The system is considered to contain many interesting reliability aspects such as: series and parallel sub-systems, cold standby and non-exponential distributions, and various maintenance strategies are applied. The system success tree and probe positions about which information is to be gathered is given on Figure 8.0c, in the format acceptable by the Simulator.

Where maintenance schemes have been applied to the example system optimal maintenance strategies have been sought, and are shown to exist. The broader problem of unspecified maintenance policy with concern about necessary system maintenance resources is also dealt with. Simulation results indicate that effective application of resources produces improved system reliability.

8.1 Statistical Estimation and Simulation Time

Where system reliability is high, simulation times are necessarily 'long' to ensure sufficient information is gathered about system events. A question is posed by this problem, that is 'how long should a simulation last to ensure good statistical estimation?'. Application of the Simulator to determine the distribution of time-to-first-system-event poses a similar question 'how many such events should be observed?'.

To aid the experimenter in determining if results obtained are a good estimate, the software analysing the results derives a measure of result variation about its mean. The measure is the coefficient of relative variation CRV [43], and is given by:

$$CRV = \frac{\sigma}{\bar{x}}$$

where σ = standard deviation of result

$\bar{x}$ = mean value of result

If a simulation time is long, then it can be expected that the value for, say, mean unavailability u_D will have little variation about its mean value. An increase in simulation time should further reduce the variation in u_D producing a smaller CRV value, therefore increasing the confidence that u_D is a good estimate of the true value. No assumption need be made that unavailability is exponentially distributed, unlike alternative tests.

Consider the estimation of u_D for two sub-systems selected from the example system. The estimated u_D values at probe P9 and component #2 are 0.0039 and 0.074 respectively. An experiment to determine the CRV for increasing simulation time has been carried out for each u_D estimation. The results are presented on Figures 8.1a and Figure 8.1b. The lower probability event occuring at probe P9 required 0.27 seconds

of simulation time to achieve a confidence in u_D estimation of CRV=0.1. For the higher probability, the same confidence was achieved in 0.015 seconds.

Simulation of the sub-system at probe P9 required 8.8 seconds to model 10^7 hours of life-time. This indicates a three-fold reduction in the simulation time compared with a calculated synchronous time of 24 seconds. The gain is due to asynchronous time scaling produced by component failure distribution time quantisations of 4 hours and 8 hours.

The problem of Statistical Gathering Module event counter overflow was reported in Chapter 7. If overflow is to be prevented, simulation times may have to be chosen lower than is desirable with regard to CRV. In such a case experiments can be repeated several times to ensure a good estimation.

8.2 Probe Positioning

After defining the system success tree, events about which information is to be gathered can be specified. This corresponds to positioning a statistical gathering probe at a logical box in the success tree. Probes can be easily removed and re-positioned, offering an opportunity to gain additional information about the system.

For the example system, the four probes available to the simulator have been located at various positions of the system tree according to Figure 8.0c. The results obtained are presented in Figure 8.2a. It can be seen that probes have been connected directly to primary logical boxes, giving information about particular component behaviour. In such cases, the probe positions are indicated by the components logical name. The results reveal that the parallel paths formed by component #2 and group #3,#4,#5 have much the same u_D, but their mean up and down times differ considerably.

The hardware construction of probes specifies that down time is recorded. Thus information about up times can be calculated from mean unavailability and mean down time, or by employing an intermediate NOT box. For example, probe P2 monitors time between system failures where probe P1 monitors time-to-system-repair. In the case of a system, initially in a down state, and information about the distribution of time to system repair is to be gathered, then a logical NOT box is essential in defining the logical event at which a simulation run stops. The distribution of time-to-first-system-failure at probe position P3 has been determined for a sample of 1000 events, and is presented on Figure 8.2b.

8.3 Parameter Sensitivity

The high speed operation of the Simulator makes it well suited to sensitivity analysis of component parameters. Previous studies [44] have considered the sensitivity of system unavailability for changes in component distribution parameters and number of repairmen. It is proposed here to consider system unavailability, down-time and up-time sensitivity to varying component life times. Changes in the number of repairmen is dealt with in later sections of this Chapter.

For the example system, the sensitivity to component failure mean time (λ) is determined for a range of λ values. That is $0.5\lambda_n < \lambda < 3.0\lambda_n$ where λ_n is the normal parameter value. Each component is considered in turn, all other components remaining unchanged at their normal λ_n value, except in the case of component #4 and cold standby component #5 where their λ parameters are varied together. The system was observed at probe position P1. Results for u_D, mean up time, mean down time are presented on Figures 8.3a, 8.3b and 8.3c respectively.

The system unavailability is shown to reduce with improvement in component life times. Certain components effect the u_D more than others, in particular an order of component #1,#2,#3, (#4,#5) is revealed. An examination of the component arrangement in the system success tree intuitively confirms the order of component 'importance' indicated.

The mean time between system failure is shown to increase with increasing component life times. The order of importance is different from the u_D case. Component #2 and component #3 are the most and least important respectively. This is explained by the component repair characteristics. For example, component #2 has a relatively long repair time. Avoiding repair by improving the life time of component #2 produces a considerable increase in system up-time.

Results for system down-time are particularly interesting, as both an increase and decrease of mean-down time occurs for improvement in component life times. Throughout the modelling exercise no change of component repair characteristics was considered. The result for mean-down time indicates that the system down time distribution can increase in mean value for improvement in component life times. This does not reduce system reliability because system failure occurs less frequently.

8.4 Application of Preventative Maintenance

It is well known [35] that application of preventative maintenance to system components improves the availability of the components themselves and therefore the system. To investigate the application of the Simulator to the problem of reliability improvement, the example five-component system has been studied with age and block replacement policies.

Throughout, modelling maintenance and repair resources are sufficient to meet all component resource demands, without delay. A later section of this Chapter investigates the effect on maintenance strategies under the condition of limited resources. For the example, system maintenance was considered to require one hour. Component #1 has not undergone maintenance because of its exponential failure distribution resulting in a deteriorated performance when maintenance is applied alongside repair.

Component #2 unavailability has been monitored for varying age and block replacement times. The results are shown in Figure 8.4a. Both replacement policies are shown to have an optimal replacement time. At time values above 300 hours, block replacement produces a lower value than age replacement. From 300 hours down to the optimal time value, block replacement is less rewarding . This is explained by the component micro-controller not incrementing the block replacement counter when the component is non-operational. At time values below the optimal value, block replacement is also less rewarding due to the greater number of operational components removed for maintenance.

Unavailability of component #3 is shown on Figure 8.4b for varying age and block replacement times. The shorter mean repair time results in a more pronounced optimal time value. Also, due to the reduced amount of time spent in a non-operational state, block replacement is shown to be the better policy for all times above the optimal.

The sub-system formed by component #4 and cold standby component #5 has been observed by probe #6 to determine the mean unavailability. Two alternative actions have been considered at failure of the cold standby component viz. replacement (corresponding to a maintenance operation), or normal repair. The results for varying age and block replacement times, applied to component #4, are shown on Figure 8.4c and 8.4d respectively. Preventative maintenance of component #4 is shown to be

most rewarding in the case where the cold standby component undergoes repair at failure.

An age replacement policy has been applied to components #2, #3 and #4, and the system unavailability determined for varying replacement time. All components were considered to have the same replacement age. This reduces the graph of u_D to a one variate, two-dimensional, plot. The cold standby component was considered to undergo repair at failure. Results for the system u_D are shown on Figure 8.4e. Component #1 does not take part in the preventative maintenance scheme and causes an attenuation in the reliability improvements produced. Better observation of the effects of maintenance can be achieved by observing the system at probe P3, that is a sub-system not containing component #1. With the age replacement policy described above sub-system mean unavailability, mean time between failure and mean time to repair have been determined. The results are shown on Figures 8.4f, 8.4g and 8.4h respectively. A particularly interesting result is an optimal mean-up time produced at 300 hours. The down time is shown to vary from the 1 hour required to carry out maintenance to 7.2 hours, the value previously determined for the subsystem undergoing no maintenance.

The system as observed at probe P3 has been further investigated for a block replacement policy. Components #3 and #4 are scheduled for simultaneous maintenance at intervals of T_B. Component #2 also undergoes maintenance at intervals of T_B but starts with an initial time shift of kT_B, $0 \leqslant k < 1$. The system unavailability for T_B values of 1000 hours, 500 hours and 200 hours is shown on Figures 8.4i, 8.4j and 8.4k respectively. Reliability generally improves as T_B is reduced. The importance of the maintenance staggering parameter, k, is also greater at lower T_B values, and an optimal k value of 0.6 is indicated. A value in this region may be considered likely when remembering, during the repair-only experiments, the mean time between failure of

component #2 was found to be greater than the mean time between failure of group #3,#4,#5. Certainly the difference between the up time characteristics of the two parallel trains forming the sub-system at probe P3 is the explanation for the improved reliability at $k>0.5$ rather than $k<0.5$.

8.5 Maintenance and Minimal Repair

Repair of a failed component may not render the component in a condition 'as-good-as-new'. Consider a component, initially new, which fails at time t. After repair the component is younger but not necessarily as-good-as-new. The age after repair is given by mt, where $0 \leqslant m \leqslant 1$. For $m=0$ the repaired component is as-good-as-new (RGN). And for $m \neq 0$ the new component condition is better than repaired (NBR). Modelling of a NBR characteristic can be dealt with in two ways depending upon the m value. Where $0<m<1$ the component interrupt facility is required to instruct the host computer to re-programme the component age counter. The second modelling technique can only be applied where $m=1$. In such cases repair is considered minimal [35,45], and the repaired component hazard function continues from the point at which failure occurred. Therefore re-programming of the age counter is not required. Such a method can only be employed if the age counter remains unchanged during the component repair process. However, modelling a non-exponential repair process requires the age counter to update the process. A compromise condition therefore exists, that is re-programming of the age counter can only be avoided if repair distributions are limited to being exponential.

The example system, as observed by probe P3, has been examined for an optimal age replacement policy considering a minimal repair condition. Component micro-controllers were re-programmed to render the age counters ineffective during the repair processes. The repair distributions were modelled by exponential distributions of equivalent mean value. Preventative maintenance applied to components #2, #3 and #4 returned components to a good-as-new condition. All components were considered to have the same replacement age, and replacement was carried out in one hour exactly. The results for the system u_D and mean-time-between-failure are shown on Figures 8.5a and 8.5b. Also shown are the results, obtained in Section 8.4, for perfect repair (RGN). It can be seen where repair is not RGN that preventative maintenance has an increasing importance in maintaining high system reliability. The optimal age parameter appears unaffected by the success of repair.

8.6 Limited Repair and Maintenance Resources

The assumption of an unlimited resources of repair and maintenance men is frequently encountered in system reliability studies. When considering resources to be sufficient to meet all demands put upon them, without delay, the availability of each system component is statistically independent. Thus the evaluation of system reliability is made easier by assuming that no waiting for repair exists. Reducing the number of repairmen increases the waiting times, and therefore degrades the system reliability.

For systems described by non-exponentially distributed times for failure and repair, and containing limited repair facilities, the evaluation of system performance by analytical techniques is extremely difficult. However, system availability has been determined for such systems by applying software modelling [8]. The indicated method

assumed that repair is conducted on a first-come-first-served principle. This cannot be assumed to be true for all systems. It is anticipated that better use of a limited number of repair men can be achieved by investigating a range of repair policies. Two experiments are now described, where not only unavailability but mean up-time and mean down-time are determined for a range of repair men and policy. Consequently, a measure of sensitivity to system reliability policy is obtained.

The example system, as observed by probe P3, has been examined for an optimal age replacement policy considering minimal repair, NBR. With the condition new better than repaired, greater demand is put upon repair facilities, highlighting the repair policy under consideration. Preventative age replacement, which returns components to the as-good-as-new condition, was applied to components #2, #3 and #4. The cold standby component #5 underwent good as new repair, RGN, at failure. Replacement times were exponentially distributed and of mean value 10 hours. Individual component unavailability has been determined for a range of replacement times. The results are shown on Figure 8.6a. It can be seen, for statistically independent components, that the optimal age replacement time for components #2, #3 and #4 is 200 hours, 500 hours and 300 hours respectively.

The system unavailability u_D, mean up-time and mean down-time has been determined where components #2, #3 and #4 compete for a limited number of repairmen. Note that repairmen are also required to carry out component preventative maintenance. Each component undergoes optimal age replacement, according to the times given above.

Where three men are available, no priority policy is necessary as all component requests can be dealt with without delay. Limiting the resources to two men or one man, allowed three priority request policies to be considered.

Policy #1: Component requests are granted in order #2,#3,#4 where component #2 is of highest priority. Once a request has been granted, the repair man cannot be recalled for re-allocation until his current task is completed.

Policy #2: Component requests have priority as in policy #1. However a repair man can be recalled from a low priority task to be redirected to a higher priority request. After completion of the higher priority task, the repair man can return to his interrupted work, unless the incompleted task has been taken up by another repair man freed by the completion of some other system task. It should be noted that the component micro-controllers are instructed to commence interrupted repair or maintenance from the point at which it was discontinued.

Policy #3: Component requests have priority #4,#3,#2. Discontinuation of incompleted requests is not allowed.

Note: For all policies described above, repair requests of any priority are given preference over maintenance requests. However, after acknowledgement of a maintenance request has been made, the request assumes the priority of the component repair request. This achieves a low priority for commencing maintenance requests (enforcing a delay during crisis) but a high priority (and therefore less likely) for discontinuation.

The easy operator interaction with the Simulator enabled these varied policies to be specified and stored for future use in less than one minute.

The results for mean unavailability, Figure 8.6b, indicate that policy #2 and policy #1 are to be chosen in preference to policy #3. This is intuitively correct, as examination of the component arrangement indicates system reliability to be more sensitive to component #2 than component #4. A more interesting result is that policy #2 is better than policy #1, indicating gains in reliability from discontinuing low priority tasks. The results for mean up time, Figure 8.6c, contain a further two interesting observations. Where two repair men are available, policy #2 achieves a mean time between system failure as good as three men. With only one repair man, policy #2 produces a poorer mean up-time than policy #1. However it must be remembered that the unavailability is lower for policy #2. Results for mean down-time, Figure 8.6d, follow the same trend as for mean unavailability, that is policy #2 always produces the shortest time to system repair.

The system has further been investigated for a block replacement policy. Components #3 and #4 are scheduled for simultaneous maintenance at intervals of T_B. Component #2 also undergoes maintenance at intervals of T_B but starts with an initial time shift of kT_B, $0 \leqslant k < 1$. The maintenance times are again exponentially distributed and of mean value 10 hours. All repair is considered to be NGR. Components #2, #3 and #4 compete for repair and maintenance men, which are allocated according to either policy #1 or policy #2 As before component #5 undergoes repair at failure outwith delay from a repair man without the normal repair team.

With T_B=1000 hours, the system u_D has been determined for a range of k values under policy #1. The results for varied number of repair men are given on Figure 8.6e. T_B was then changed to 300 hours, and the system u_D determined for a one man and three men case under policies #1 and #2. The results are shown on Figure 8.6f. With T_B=300 hours, maintenance is applied too frequently with respect to optimal u_D. The one man policy achieves a better system availability than the three men policy, because many component maintenance requests cannot be immediately responded to, and are delayed for later consideration. With the practical T_B value of 1000 hours, system availability showed an improvement for greater numbers of repair men. The greater degradation in performance shown for a reduction of two men to one man, compared with three men to two men, is probably due to the reduced likelihood of components#3 and #4 being maintained simultaneously.

8.7 Optimal Economic System Operation

Preceding sections of this Chapter have dealt with determining the reliability of the example five-component system. Various age and block replacement policies were applied to the system with the aim of improving system reliability. Optimum implementations of chosen policies were derived with the view of minimising system unavailability and mean repair time or maximising mean time between failures. This section is concerned with obtaining maintenance stratagies which minimise total system cost, therefore obtaining best value from the system function.

Modelling a system's economic performance requires all the information about component behaviour associated with reliability modelling, and in addition information about the cost of repair and maintenance operations performed on components. The data about cost of repair men actions is not itself necessary to specify the system model, but is essential in analysing the cost of system up-keep determined at the end off a simulation. However, knowledge of the format of the component cost must be known before simulation can take place, as information about component behaviour not observable by the Statistical Gathering Module may be required to accurately determine component costs.

Consider a component undergoing preventative maintenance and repair at failure. When the component becomes unavailable, due to either repair or maintenance, the Statistical Gathering Module could only record a down event, as it is unable to distinguish between different kinds of events represented by the same binary logic level. Essentially the Network Specification Module, through which the Statistical Gathering Module is connected to the components, forms a window on the simulation, and not all forms of system behaviour can be observed through this window. To distinguish between repair and maintenance events, component micro-controllers have been programmed to increment component counter C_c at the completion of each maintenance task. Thus the host computer can examine the component counters at the end of a simulation enabling it to determine the total cost of repair and maintenance operations for each component.

Frequently investigations into the cost of system up-keep are needed to aid the selection of an age or block replacement policy [46], or to determine optimal replacement parameters [35]. Systems are normally single component with repair resulting in an as-good-as-new condition (RGN). Also repair and maintenance times are considered instantaneous.

Variation of this arrangement, such as the possibility of unsuccessful maintenance, has been considered [47]. The restrictions of RGN, instantaneous repair, and single component system, requires preventive maintenance to cost less than repair, due to the association of component failure with system failure.

Where systems consist of several components and repair and maintenance times are not instantaneous but random variables of known distribution, optimal maintenance policies may exist where maintenance is more costly than repair. For such systems individual component failure may not lead to system failure, and the optimal maintenance policy is the one which achieved the most economic availability of system components.

The cost of system failure is more likely determined from the total system out-time than the frequency of failure events. Whichever is the case, this additional 'penalty cost' and the cost of component repair will influence the optimal maintenance policy parameters. When repair is considered to be minimal, an economic maintenance policy [48] exists regardless of the nature of the repair and maintenance distributions.

The example five-component system is now considered with the aim of obtaining an optimal economic operating condition. The system is viewed as a production plant where loss of the system has a cost penalty per hour. A linear function for system down time cost simplifies the problem as only the mean down time value need be known to evaluate cost of lost production. However the problem is complex and considerable information is gathered about the system performance. System components undergo age replacement, and an optimal age is found when repair and maintenance resources are considered unlimited. Later a cost penalty is associated with repair men and the optimal number of repair men is determined for a varying system outage penalty cost.

For the example system under investigation component #1 has been removed due to its exponential failure distribution. A component with a non-increasing hazard rate does not reward preventive maintenance with improved reliability. If component #1 had been included but not taken part in the maintenance scheme, the sensitivity of system performance to maintenance parameters would have been reduced.

Components #2, #3 and #4 have been individually monitored for an age replacement policy, where minimal repair is applied at failure. The cost of preventative maintenance applied to any component was two units and the cost of repair was only one unit. Results obtained for the cost of component component up-keep per 10^4 hours operation are given on Figure 8.7a for various age replacement times. The cost for the cold standby component is shown to be dependent on the age replacement time of component #4, i.e. the component whose failure activates the cold standby. Optimal, with regard to cost, replacement times for components #2, #3 and #4 are revealed to be 800 hours, 800 hours and 400 hours respectively.

The system was then modelled where components underwent age replacement according to the most successful time parameters observed above. Components #2, #3 and #4 competed for available repair men which were varied from one to three men. Note that repair men were also required to carry out the preventative maintenance work necessary for economic system operation. The repair and maintenance facilities were utilised according to Policy #1 [Section 8.6]. Briefly, repair has priority over maintenance and component requests have priority #2,#3,#4 where component #2 has highest priority. Component #5 received repair without delay from a repair man outwith the normal repair team. The results for the system unavailability, mean repair time, mean time between failures, and number of failure events per 10^4 hours operation are given on Figure 8.7b. Also shown are the number of repair and

maintenance actions per 10^4 hours operation performed on each component.

It can be seen that as the number of repair men reduce, the overall system performance degrades and the number of repair and maintenance actions reduce as components are forced to wait for attention. The total cost of all repair and maintenance actions per 10^4 hours operation has been calculated and has been given. A cost of 230.1 units for actions taken by three men is significantly higher than the 90.4 units for the actions taken by one man.

When the cost of employing repair men is also considered then the cost penalty associated with lost production must be high to justify a 3-man policy. Let the cost of employing a repair man be 100 units per 10^4 hours, Figure 8.7c shows the total system cost (that is cost of repair men, cost of repair and maintenance actions and cost of lost production) for increasing outage cost per hour. Where down-time costs less than 0.3 units/hour, one man is shown to be most economic, and where the cost is higher than 0.53 units/hour, three men achieve best economy.

In the calculation of the system penalty cost, given here for the example system, certain information gathered about the system operation has not been included. However it is invisaged that other views of the economic operation of the system may require the additional information. For example when planning storage of system output, so as to maintain the supply of the finished product even when the system is non-operational, information about the mean system down time is essential.

8.8 Reliability Growth and Wearout

Throughout reliability modelling, the assumption that repair or maintenance operations return system components to their initial condition is frequently made. The assumption leads to the time homogeneous processes for modelling system behaviour. However it has been shown [49] that repair of a failed system often results in times to subsequent failures which are not independently and identically distributed. Further, considering systems where design modifications and other corrective actions are taking place, successive failures should not be expected to be independent. The use of homogeneous processes, whether Poisson or not, cannot accurately model the behaviour of such systems.

This modification at failure or as a result of failure applies to both hardware and software systems. In the case of software, statistical models have been developed with the aim of monitoring reliability growth and predicting future times of failures [50, 51]. The model parameters are estimated throughout system development. No truly satisfactory model has yet been developed.

Reliability growth models employing non-homogeneous process models have been considered [52, 53], where the times between failures are exponentialy distributed and determined from the number of errors remaining in the system. This results in a hazard function, z(t), reducing in steps as errors are removed at failure. In the case of [52] the hazard function is simply calculated from:

$$z_i = \emptyset[N-(i-1)]$$

where i=number of errors removed.

N=initial error content.

$\emptyset$=proportionality constant.

A more complicated function is given in [53], where the hazard rate is also a function of time.

Application of a non-Poisson process is described by [50] where z(t) is not only a function of number of system errors still present but also the amount of time spent debugging, measured from the previous failure. The hazard function is given by:

$$z(t_i) = \phi[N-(i-1)]t_i$$

Note, where non-Poisson processes exist in conjunction with reliability growth, the hazard function may be increasing between failures, while the times between successive failures are nevertheless getting longer.

Many reliability growth models have application in describing both growth and wearout of hardware and software systems. The general application of Poisson processes however must be treated with caution until statistical testing confirms a constant hazard function between system modification or repair. However an interesting development of the simple Poisson model described above has been made [54]. That is a hybrid geometric-Poisson process, taking into account random error and error removal/introduction which occurs in reducing/increasing groups. The hazard function is given by:

$$z_i = Dk^{i-1} + \lambda$$

where λ=random error rate.

D=initial failure rate of geometric process

k=growth/wearout ratio

Within the Simulator, the modelling of non-homogeneous processees is dealt with by host software modelling routines accessed by the interrupt facility incorporated in each Component Module. When the model is to be updated, the simulation is halted until the necessary component distributions are re-programed, after which simulation is continued. The method equally well copes with Poisson and non-Poisson processes. The generation of an interrupt can be micro-programmed to occur after a prescribed length of time or a specified component operation or a stipulated number of events such as failure.

The Simulator has been used to study a two-component system where each component may undergo a non-homogeneous Poisson process. Components are in a redundant arrangement and statistical information is to be gathered about the distribution of time to first system failure.

Consider first components undergoing reliability growth. Each component has initially an expected life time of $1/\theta_i$, but after a long geometric growth period components have life times of $1/\theta_i+1/\theta_g$, as shown on Figure 8.8a. The expected component life time is given by:

$$1/\theta = 1/\theta_i + (1-k^n)1/\theta_g \qquad ,n=0,1,2,\ldots$$

For the example system, components are identical with, $1/\theta_i$=500 hours, $1/\theta_g$=500 hours and a failure distribution time quantisation of 8 hours. The repair was carried out immediately upon failure and was exponentially distributed with a mean value of 50 hours.

The Simulator was programmed to gather 1000 samples of time to failure starting from an initial all-working state, for growth k values of 0.75 and 0.5. The results were plotted on a 50 bar histogram then 'smoothed' for clarity as shown on Figure 8.8c. Also shown are the distributions of time to first failure for systems with no growth but starting with initial component life times of 1000 hours and 500 hours.

It can be seen that reliability growth has a considerable influence on the mean time to failure, and as expected the more rapid growth of k=0.5 produces the better system reliability. Where growth is applied, the distributions tend to be initially peaky, corresponding to a large number of failures before significant reliability growth has occured.

Reliability wearout has also been considered for the above system. The components have initially expected life times of $1/\theta_f+1/\theta_w=1000$ hours which reduce geometricaly to $1/\theta_f=500$ hours, as shown on Figure 8.8b. At any instant the expected component life time is now given by:

$$1/\theta = 1/\theta_f + 1/\theta_w\, k^n \qquad ,n=0,1,2,\ldots$$

$$0<k<1$$

This wearout process, unlike the wearout processes previously described sets a limit to the deterioration of expected life time due to wearout. The failure distribution time quantisation is again 8 hours and the repair times are exponentially distributed of mean value 50 hours. The distribution of time to first system failure obtained from 1000 samples is shown on Figure 8.8d, for k values of 0.75 and 0.5. The expected system life time is shown to be considerably reduced by wearout. Where wearout is rapid, i.e. k=0.5, a peak is shown to develop at 3000 hours corresponding to extensive reduction in component life times. For the slower wearout process the peak is shallower and broader indicating better system survival.

8.9 Valve Failure on Gas compressor (Industrial Problem)

The Simulator is well suited to tackling problems which are mathematically difficult. To further confirm this aspect of the Simulator, an industrial problem encountered by an oil producing company operating in the North Sea has been studied.

The particular item of equipment chosen for study was a reciprocating gas compressor. In certain applications a significant portion of the repair effort applied to the compressor was directed towards valve failure. For this reason, modelling of preventative maintenance schemes was undertaken, with the object of assessing reliability improvement techniques, and most importantly determining strategies of repair and maintenance.

The compressor has six cylinders and each cylinder has a low and high pressure stage, termed first and second stages. All stages have suction and discharge ports, each consisting of two suction and two discharge valves. For the compressor to work successfully all valves must be operational. Thus in reliability terms, the valve system is series. The preventative maintenance scheme is block replacement and no staggering of the replacement policy is allowed due to the series system arrangement. Modelling of the entire valve system is not required to determine reliability-improving maintenance policies. Any improvement experienced by a single valve directly contributes to the overall system improvement. In addition the optimal maintenance parameters for a valve operating alone remain optimal when considering that particular valve's contribution to overall system reliability. The simulation has therefore been concerned with modelling single valve behaviour.

The operational life of any valve was found to be terminated by one of three possibilities:

1. The valve fails and is replaced by a new valve.

2. The valve is suspended during the repair process of another valve which itself has failed.

3. The valve is suspended at the block replacement time.

Only first-stage valve operation has been considered as second-stage data could not confidently be associated with a mathematical probability distribution. Two tables of data relating to suction valves and discharge valves respectively were constructed. For each table, a cumulative distribution plot was produced and a computer program used to determine the parameters of the best-fitting straight line through the data points. From the parameters obtained, the Weibull distribution describing valve life times was found to have a decreasing failure rate (DFR). Experience of the compressor operation indicated that the valves had a wearout characteristic, that is an increasing failure rate with time (IFR). The difference between the derived Weibull parameters and that anticipated from experience was confidently believed to be due to failure occuring via more than one type of failure mode. By assuming failures that occur within 200 hours of start-up to be due to first mode (early-death mode) failure, the data was re-examined, and a three-parameter IFR Weibull distribution obtained for second mode (wearout mode) failure.

To enable modelling to take place on the Simulator, data for a two-parameter DFR first mode failure and a two-parameter IFR second mode failure must be specified. The second-mode parameters were estimated from the graph of cumulative hazard rate for failure times greater than 200 hours. Figures 8.9a and 8.9b show the graphs for suction and

discharge valves respectively. The method [55] is not as accurate as the cumulative distribution plot, but due to the small number of samples it should be sufficient. First-mode failures were estimated to have a mean value of 100 hours and a cumulative distribution function reaching 90% at 200 hours. This results in a DFR at B=0.4 [56]. The cumulative distribution function of failure is given by:

$$F(t) = p_1F_1(t) + p_2F_2(t) \qquad ,p_1+p_2=1$$

where $F_1(t)$ = cumulative distribution function of first mode failures

$F_2(t)$ = cumulative distribution function of second mode failures

and, p_1 is the probability of first mode failure (M1). From the data tables, p_1 was estimated to be 28.6% and 25% for suction and discharge valves respectively.

A two-parameter Weibull distribution was used to describe the valve repair times. In particular, B=1.12, C=7.31 giving a mean value for repair of 7.01 hours. The minimum time value for the three-parameter distribution was one hour and the c.d.f. value at one hour for the two-parameter approximation is 10%.

The aim of simulating valve operation was to improve the quality of decision-making relating to maintenance policies. The three following questions were posed:

1. What is the optimal block replacement time?

2. Should suspension at block replacement time be foregone if the valve was replaced within the block interval?

3. Can suspension during the repair process of some other faulty value be recommended?

To examine the effect of block replacement time (T_B), valves have been

modelled for various T_B values ranging from 2000 hours to 12000 hours and the mean unavailability determined (maintenance is assumed to take place instantaneously). The results are shown on Figure 8.9c. Also shown is the unavailability achievable if first mode (M1) failures are removed.

The suction valve reliability is improved by reducing T_B, the improvement being most noticeable when M1 failures are removed. The discharge valve reliability also shows a large improvement when only wearout failures are considered, but the inclusion of early death failures in the model results in destructive maintenance for all replacement times. It is concluded that the generally more reliable discharge valves should not undergo preventative maintenance as it appears better to run the risk of wearout than to be subjected to possible early death of the replacement valve.

With regard to the second question above, a block replacement policy of T_B=4500 hours (6 months) has been modelled where replacement is only permitted if the valve operation time exceeds the operation time, T_A. The results are shown of Figure 8.9d. For suction valves, a small improvement in unavailability is achieved at T_A=1500 hours, and the reliability reduces as T_A is increased towards 4500 hours. Contrastingly, discharge valve reliability improves as T_A is increased. This, to some extent, confirms the block replacement results above, because as T_A approaches T_B, the probability of undergoing block replacement reduces and such replacement was previously shown to be unsuccessful.

To investigate the final question, that of suspending a valve at some random time point, an age replacement policy has been used. Under the policy, valves were instantaneously replaced at time T_A. The assumption that replacement is instantaneous can be justified by carrying out suspension during the repair process of the existing faulty

valve. The results for varying T_A values are shown on Figure 8.9e. The suction valve results indicate a considerable improvement in unavailability for T_A values above 2000 hours, and suspension at lifetimes greater than 2000 hours can be recommended. Suspension below 1000 hours is shown to degrade valve reliability. The more reliable discharge valves show a very small improvement due to suspension after 5000 hours, and suspension below 5000 hours leads to increased unavailability.

8.10 Standby Power Supply System (Industrial Problem)

The following section describes a problem concerning a standby power supply system [57]. The problem was initially encountered by the National Center of Systems Reliability and has been successfully investigated by thier reliability simulation program ALMONA. Consequently an analytical solution of the probability of system failure is available.

A schematic arrangement of the standby power supply system is shown on Figure 8.10a. The 3.3kV switchboard is normally supplied through the two transformers #1 and #2 operating in parallel. That is sections A and B are normally connected together

In the event of loss of the normal supply, it is necessary that sections A and B are separated and the emergency generators started to supply the essential loads on each section. Loss of the normal electrical supply is detected by two undervoltage (UV) relays, any one of which will initiate operation of the standby system. The events taking place following undervoltage detection are as follows:

1. Starting-up of both diesel generators.

2. Tripping of both transformer isolating circuit breakers.

3. Tripping both bus-section circuit breakers. Operation of any one circuit breaker is sufficient to isolate sections A and B.

4. Tripping all six load circuit breakers.

Following the completion of the above, the essential loads can be re-connected by:

1. Closing both diesel generator circuit breakers.

2. Closing the four circuit breakers associated with the essential loads.

The system is considered successful if any two of the essential loads are re-energised.

Each component in the standby system undergoes simultaneous inspection at yearly intervals i.e. T_B=8760 hours. Faults which have occurred between inspection times are detected and components are returned to an as-good-as-new condition. The inspection time is considered to be instantaneous and the simulation results have been adjusted accordingly.

By modelling the system using the stochastic Simulator, the probability of unsuccessful operation at one year after the system was last tested was found. Also the mean unavailability was determined.

Component failure distributions are assumed exponential and are listed below.

	Failure to trip	Failure to close (F/yr)
Transformer CB	0.005	---
Bus-section CB	0.005	---
Generator CB	---	0.01
Load CB	0.005	0.01

UV relay failure rate = 0.01 (F/yr)

Diesel generator probability of successful start = 0.98

The flow diagram for the system is shown on Figure 8.10b. The load circuit breakers are shown twice, to represent the probability of opening and closing so as to re-energise the essential loads.

The system contains eighteen components, therefore in its present configuration it cannot be modelled by the Simulator, which is limited by hardware to only five components. To overcome this problem the system was sub-divided into manageable sections, and the results from each section used to determine the overall system performance. Figure 8.10c shows the equivalent flow diagram.

Section A and B Isolation. This is accomplished if at least one UV relay detects a voltage drop and at least one bus-section CB trips. The probability of successful isolation at time t is given by:

$$P_I(t) = [1-(\bar{P}_u(t))^2][1-(\bar{P}_b(t))^2]$$

where $\bar{P}_u(t)$=probability of UV relay failure.

$\bar{P}_b(t)$=probability of bus-section CB failure.

Modelling this section would require four component modules. However bus-section circuit breakers cannot be modelled with confidence. Consider a minimum time quantisation TQ_{min}=8.76 hours. The expected life time of 200years for the CB would necessitate a distribution time quantisation of TQ=128 hours [Section 3.9]. The results from Chapter 7 indicated that block replacement time T_B must be T_B>10TQ. Therefore for successful modelling, T_B is required to be greater than the $8760/128$ hours in use.

Two component modules have been employed to simulate UV relay operation only. The theoretical mean unavailability, u_D, for the relays has been determined by integrating $P_u(t)$ over $0—T_B$. The analytically determined results, together with those obtained from the Simulator are given below:

	Expected	Simulator result
$\overline{P}_u$	9.99×10^{-5}	9.42×10^{-5}
u_{DI}	3.31×10^{-5}	3.40×10^{-5}

Because of the high reliability expected from bus-section CB operation, the mean unavailability of this section cannot be obtained with confidence. However the probability of failure at TB can be determined by combining $P_u(t)$ above with the theoretical $P_b(t)$:

	Expected	Simulator/Analytical
$\overline{P}_I$	1.24×10^{-4}	1.19×10^{-4}
u_{DI}	4.14×10^{-5}	---

Section A and Section B. These are identical, each consisting of five components connected in series. Consider section A. The probability of this section working at time t is given by:

$$P_A(t) = P_t(t)\ P_d\ (P_o(t))^3$$

where $P_t(t)$=probability of the transformer CB tribing.

P_d =probability of the diesel generator starting and CB closing.

$P_o(t)$=probability of a load CB opening.

The mean unavailability of the section, u_{DA}, is obtained by integrating $P_A(t)$ over $0—T_B$.

With a series arrangement the failure hazard rate is the sum of the individual component hazard rates. This enables the section to be simply modelled by a single component module. The results are given below.

	Expected	Simulator results
$\overline{P}_A$	4.9×10^{-2}	4.83×10^{-2}
u_{DA}	3.46×10^{-2}	3.42×10^{-2}

Closing the Load CB. If both section A and section B are isolated and successfully operating, at least two of the four load circuit breakers must close to supply the essential loads. The probability of the two-out-of-four operation is given by:

$$P_{V4}(t) = 6P_c(t)-8P_c(t)+3P_c(t)$$

where $P_c(t)$=probability of a load CB closing.

A four-component model has been used to determine the probability of success after one year and the mean unavailability.

	Expected	Simulator result
$\overline{P}_{V4}$	3.9×10^{-6}	5.12×10^{-6}
u_{V4}	9.84×10^{-7}	19.8×10^{-7}

Should section A or section B fail, the two circuit breakers on the remaining section must operate. A simulation of two circuit breakers in series produced the following results.

	Expected	Simulator result
$\overline{P}_{V2}$	1.98×10^{-2}	2.15×10^{-2}
u_{V2}	9.94×10^{-3}	10.75×10^{-3}

System Operation. To determine the mean unavailability for the complete system, seven component modules would be required, i.e. one for each of the above sections and one for the four essential load circuit breakers. A six component model is possible by representing the two-out-of-four system by three components. However, in a similar manner to the reduced sub-section isolation model, the component parameters would be outwith the modelling range allowable

The probability of system failure at the one-year point can be determined by combining the results of the above sections. If P_S is the probability of system success, then:

$$P_S = P_I[P_A P_B P_{V4} + P_A \overline{P}_B P_{V2} + \overline{P}_A P_B P_{V2}]$$

The analytical and simulation results are then:

Theoretical $P_S = 1-4.37\times10^{-3}$

Simulation $P_S = 1-4.43\times10^{-3}$

10.11 Conclusion

The Simulator has been applied to several reliability problems which would have been difficult to solve by analytical means. The systems studied were either derived from an interesting 5-component example system, or obtained from industry.

Extensive use of the preventative maintenance modelling features of the Simulator have been made. The results produced show that the Simulator is a convenient mechanism for deriving reliability-improving maintenance policies where the characteristics of the system under study are complex. Throughout the modelling exercises, the simulation times have been consistently short.

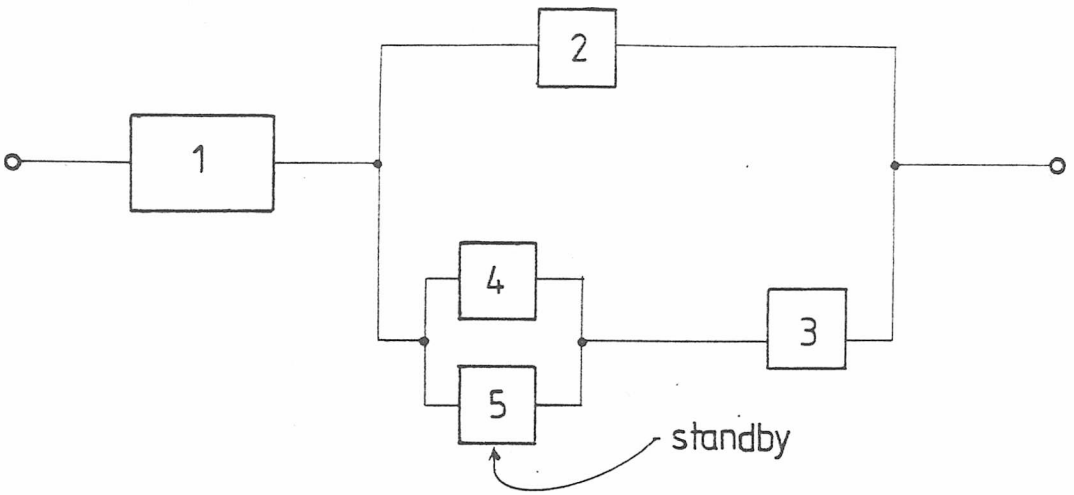


Figure 8.0a System under consideration

component	Failure Distribution	Repair Distribution	features
1	exponential λ = 2000h ΔT = 4h	exponential λ = 15h ΔT = 1h	
2	Weibull λ = 500h β = 2, ΔT=8h	Weibull λ = 40h β = 2, ΔT=1h	
3	Weibull λ = 500h β = 2, ΔT=8h	Weibull λ = 15h β = 2, ΔT=1h	
4	Weibull λ = 250h β = 2, ΔT = 4h	Erlang λ = 10h β = 2, ΔT = 1h	
5	Weibull λ = 250h β = 2, ΔT = 4h	Erlang λ = 10h β = 2, ΔT=1h	cold standby failure rate 1/1000h start-up failure risk 10%

Figure 8.0b Component specifications

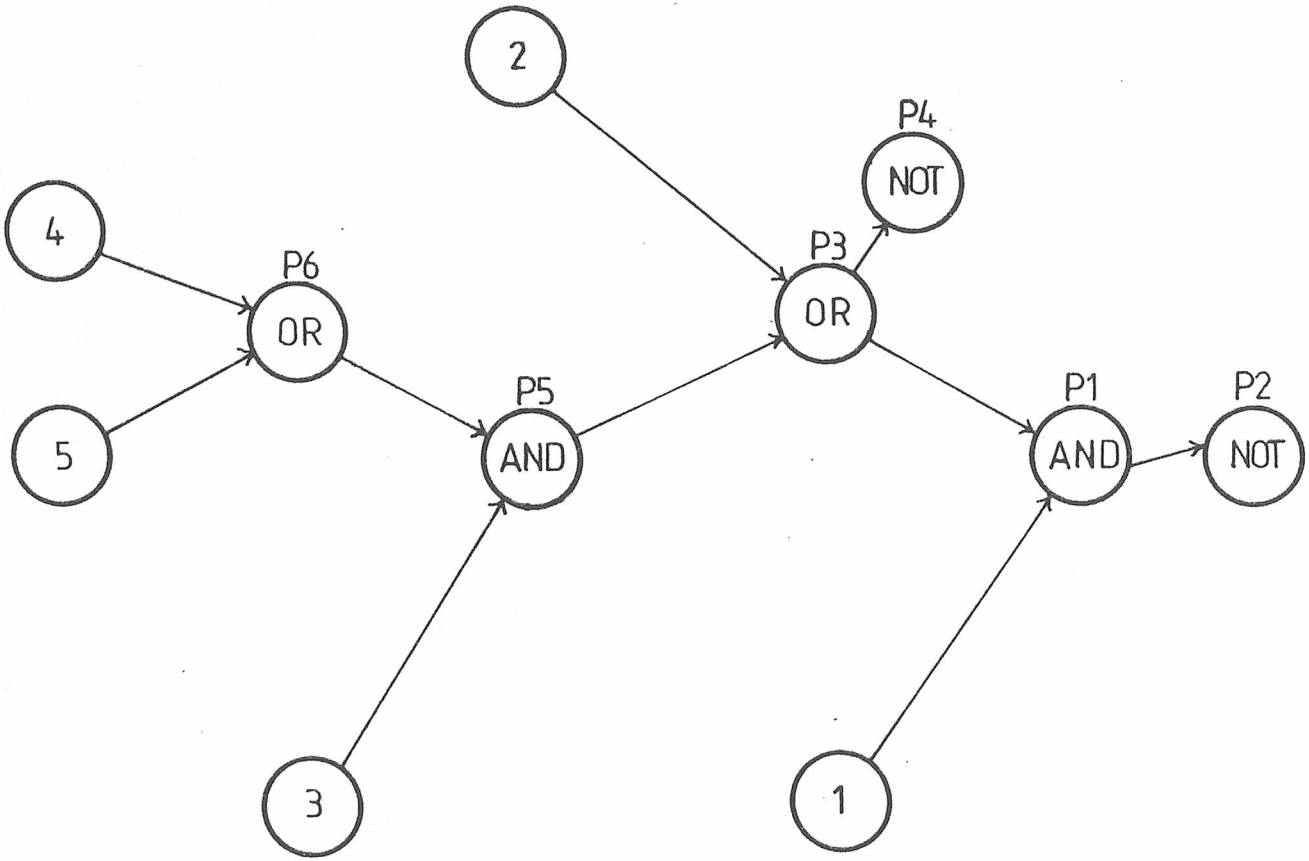


Figure 8·0c Success tree of example system

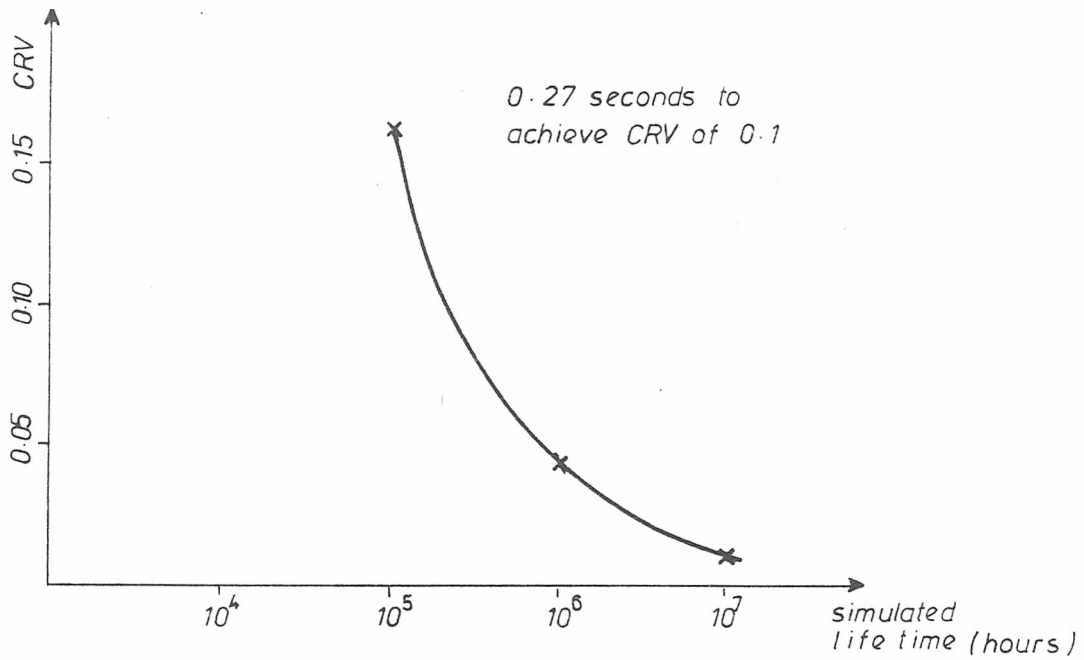


Figure 8.1a Estimated probability $\mu_D = 0.0039$

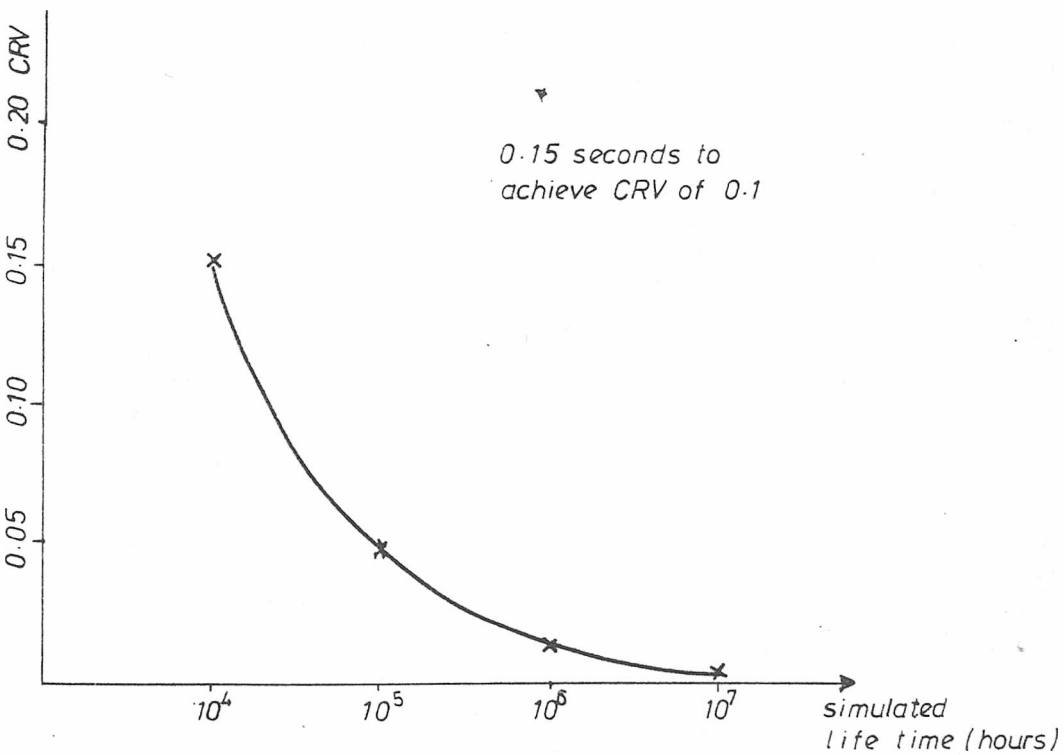


Figure 8.1b Estimated probability $\mu_D = 0.074$

CRV = Co-efficient Relative Variation

probe position	mean μ_D	mean down time	mean up time
component 1	0·0751	15·6	2037·9
component 2	0·0787	35·8	416·3
component 3	0·0296	13·7	438·4
probe P6	0·0397	5·9	—
probe P5	0·0549	8·6	—
probe P3, P4	0·00391	7·2	1809·6
probe P1, P2	0·0116	11·2	948·2

(all times in hours)

Figure 8·2a Result of varying probe position

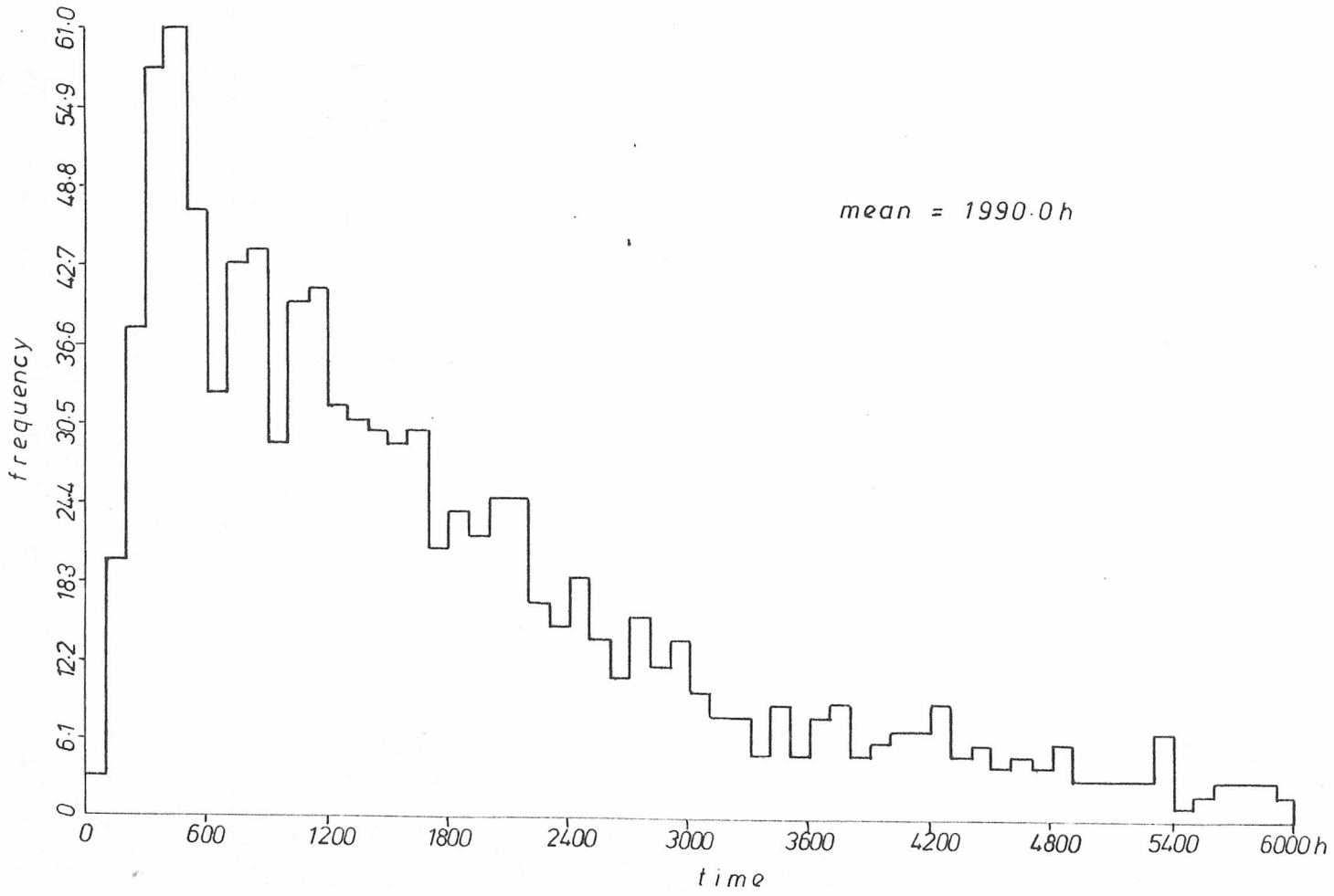


Figure 8·2b Distribution of time to first system failure at P3

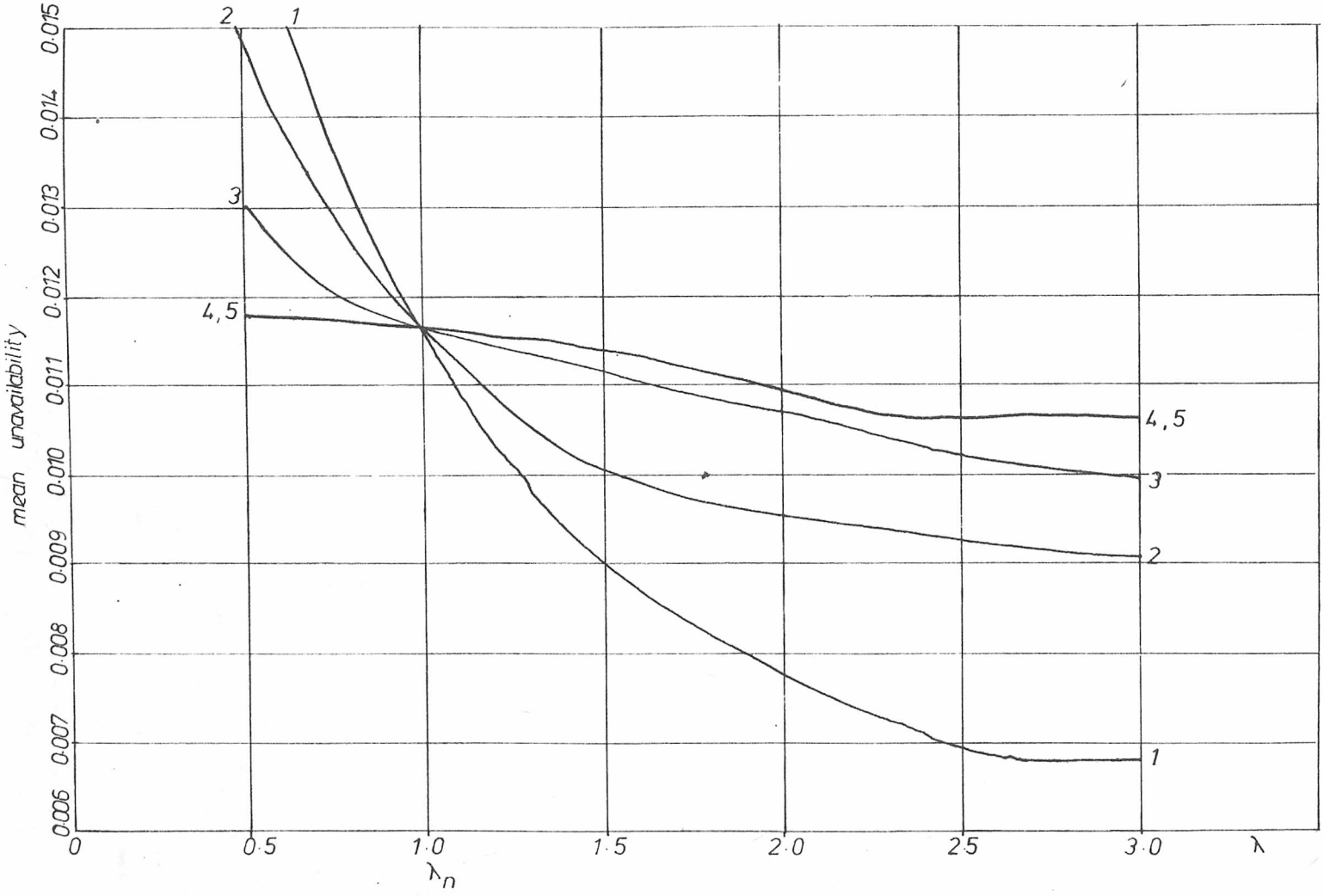


Figure 8·3a Mean unavailability sensitivity to component parameters

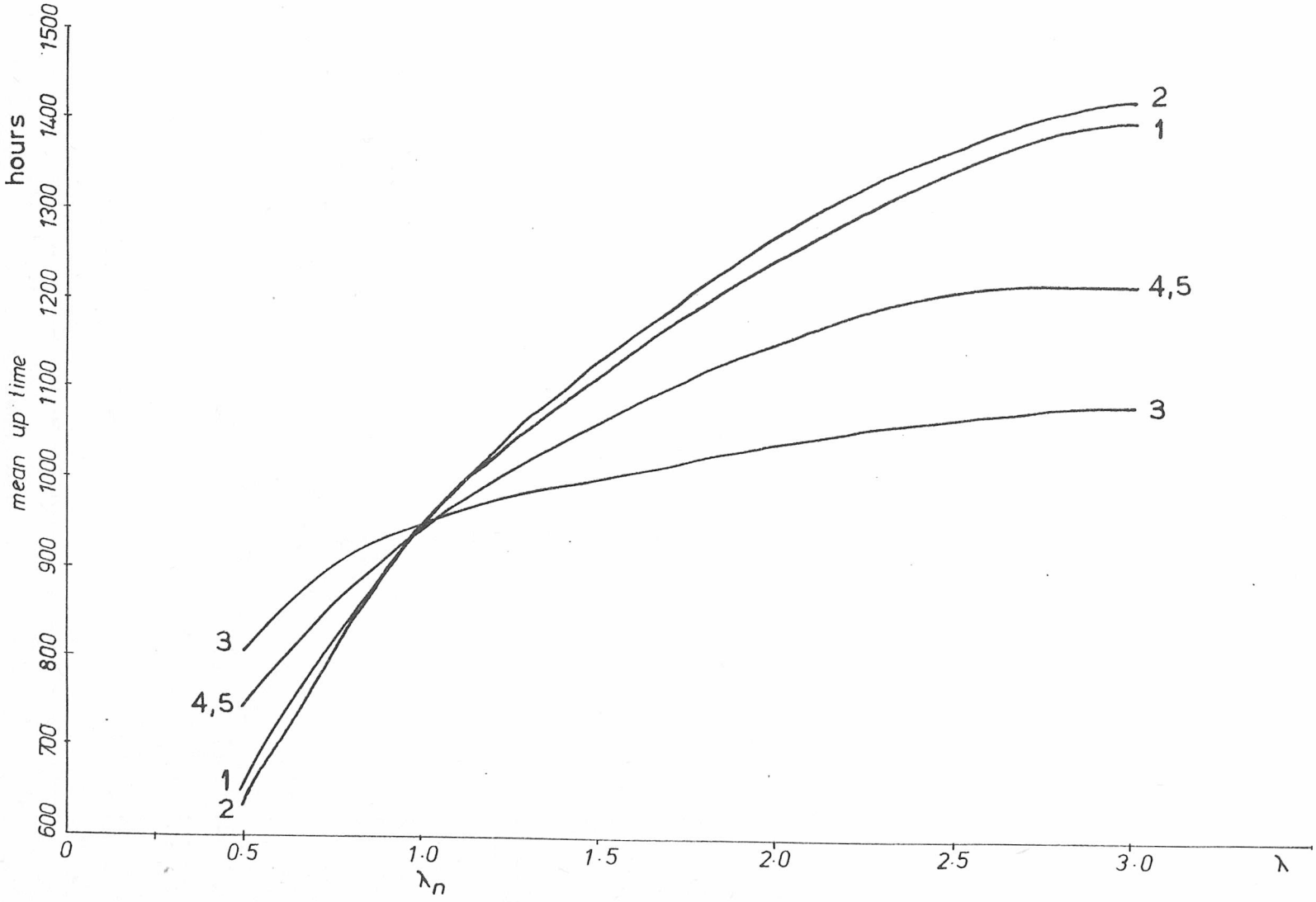


Figure 8·3b Mean up time sensitivity to component parameters

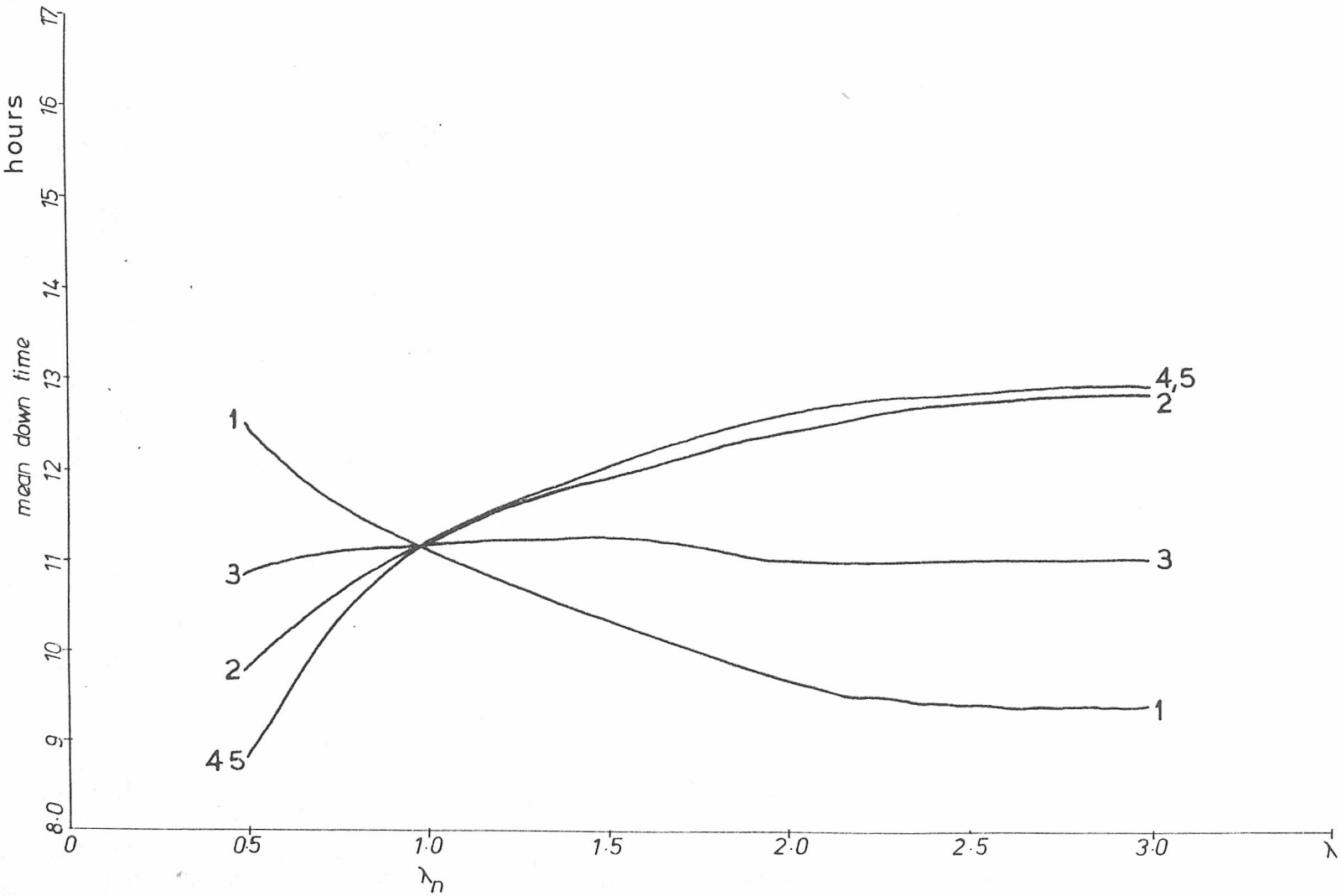


Figure 8·3c Mean down time sensitivity to component parameters

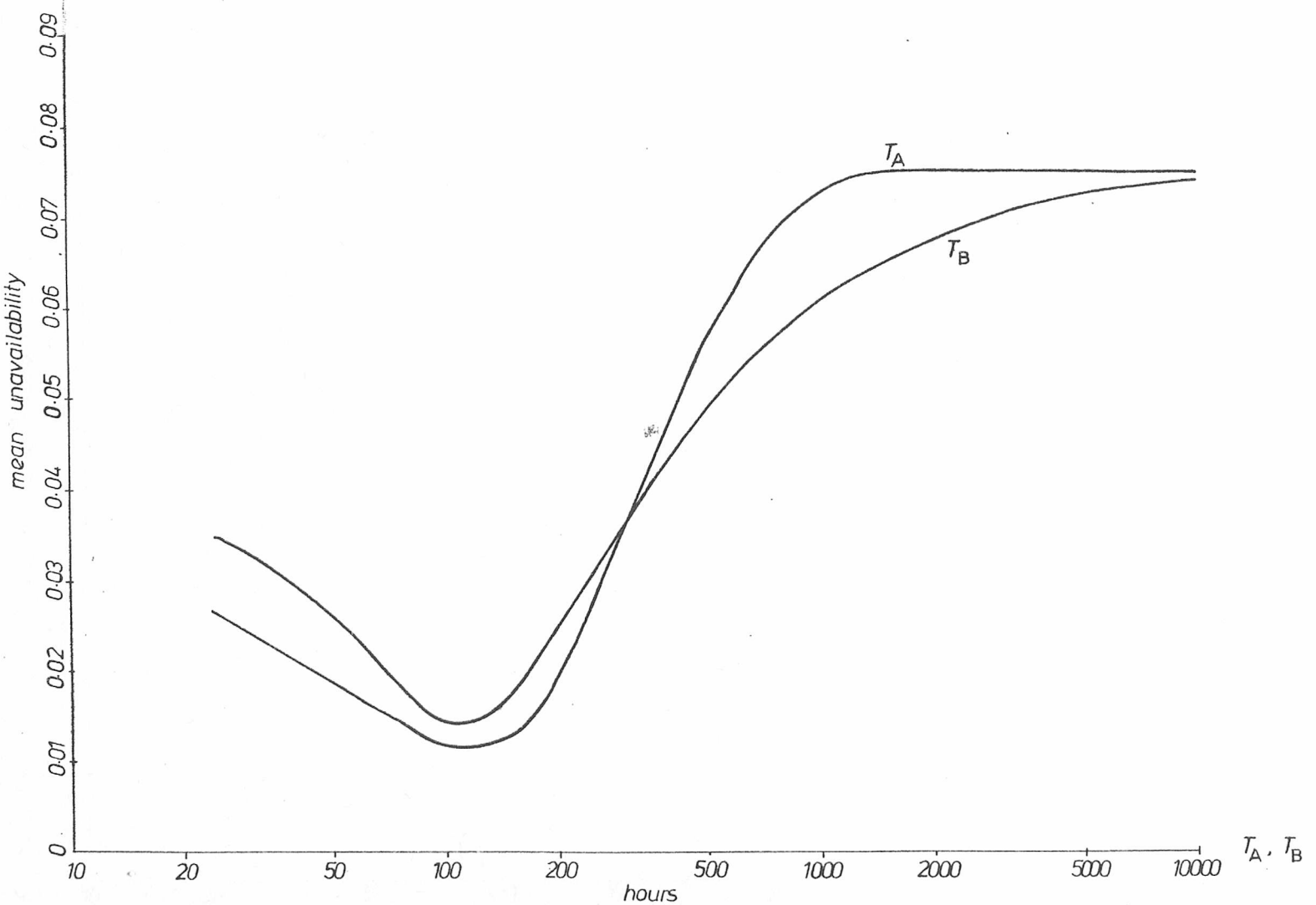


Figure 8·4a Component 2 unavailability for AGE and BLOCK replacement

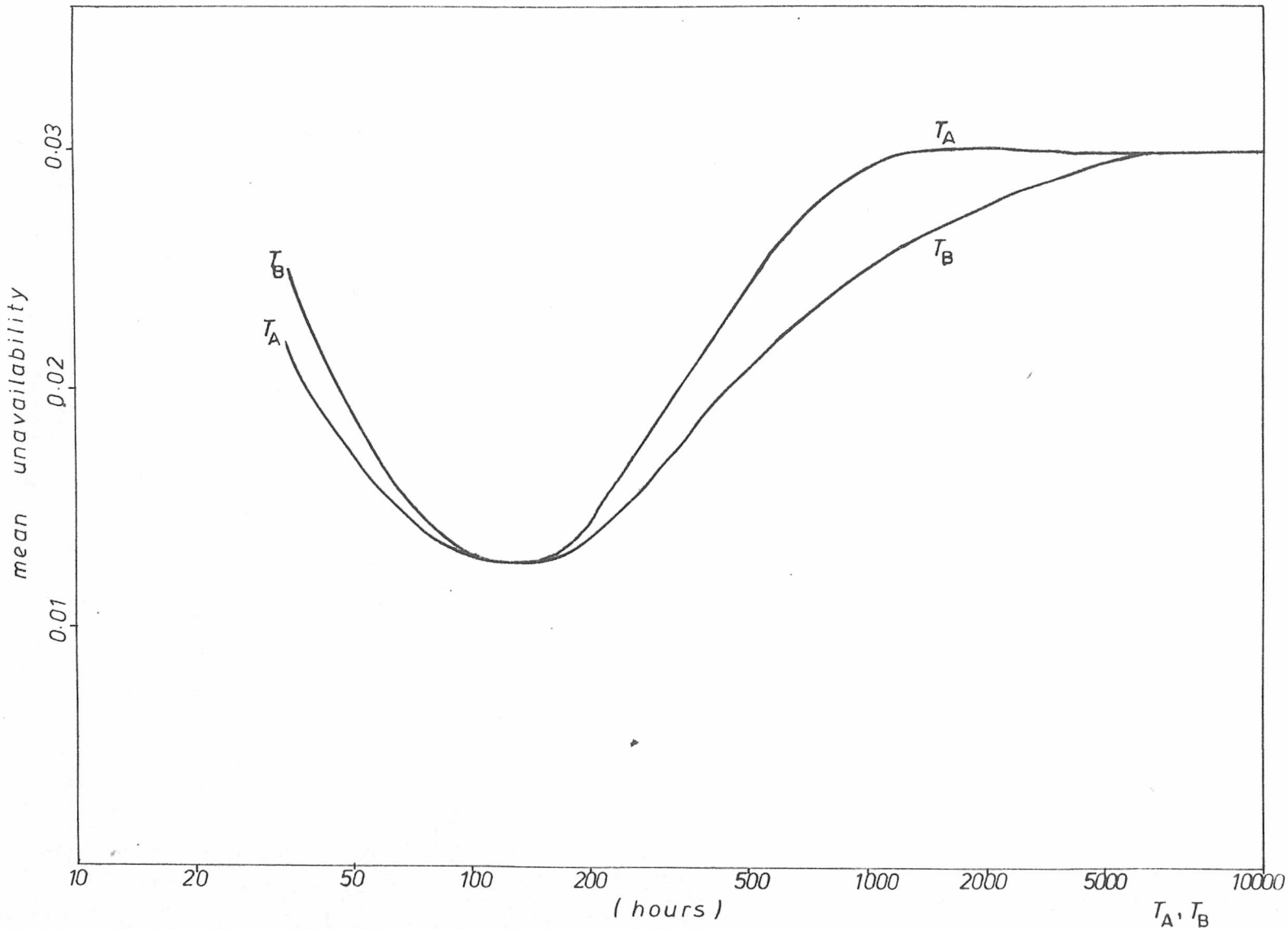


Figure 8.4b Component 3 unavailability for age and block replacement

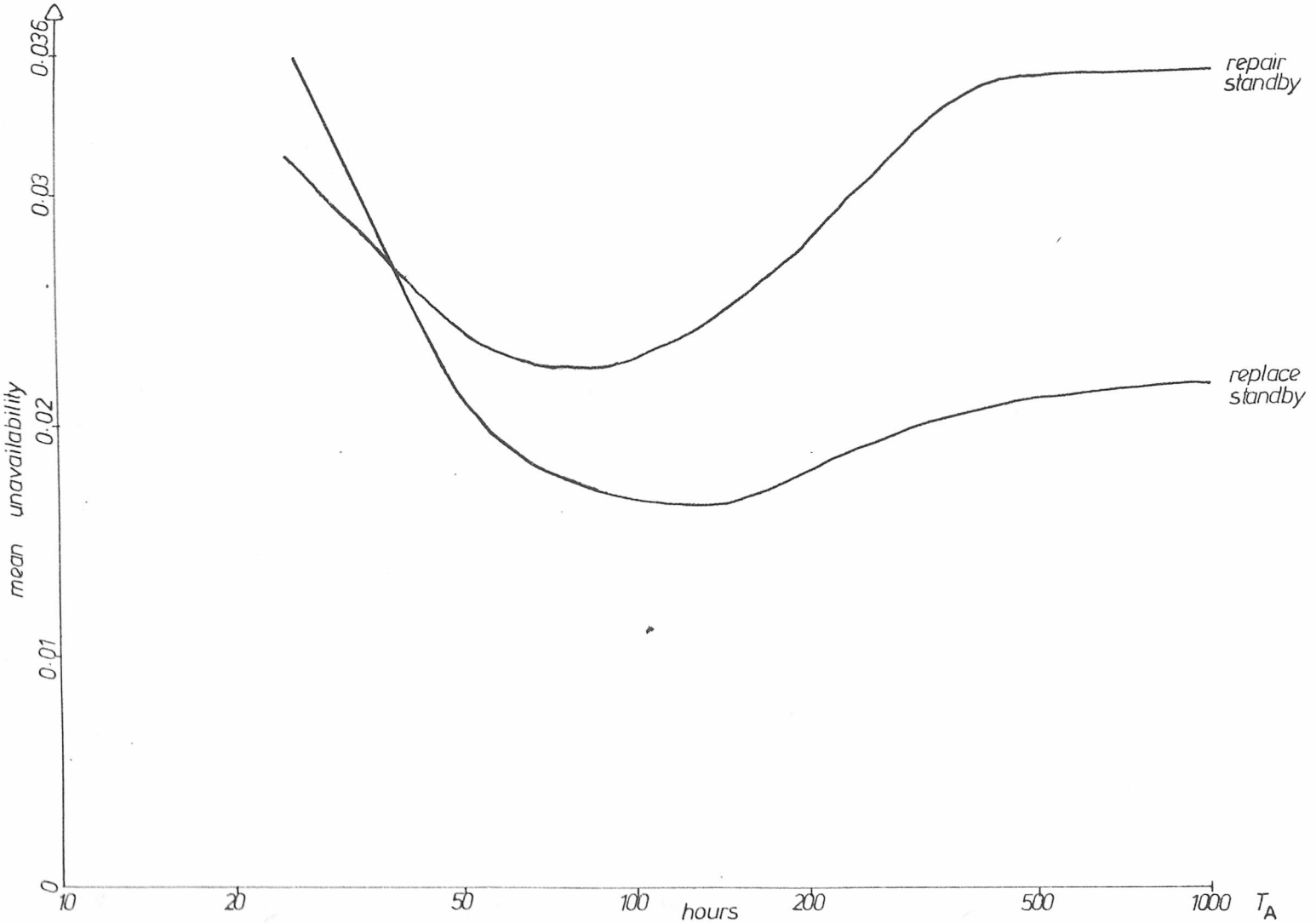


Figure 8·4c Unavailability of component 4 and cold standby component 5 with age replacement

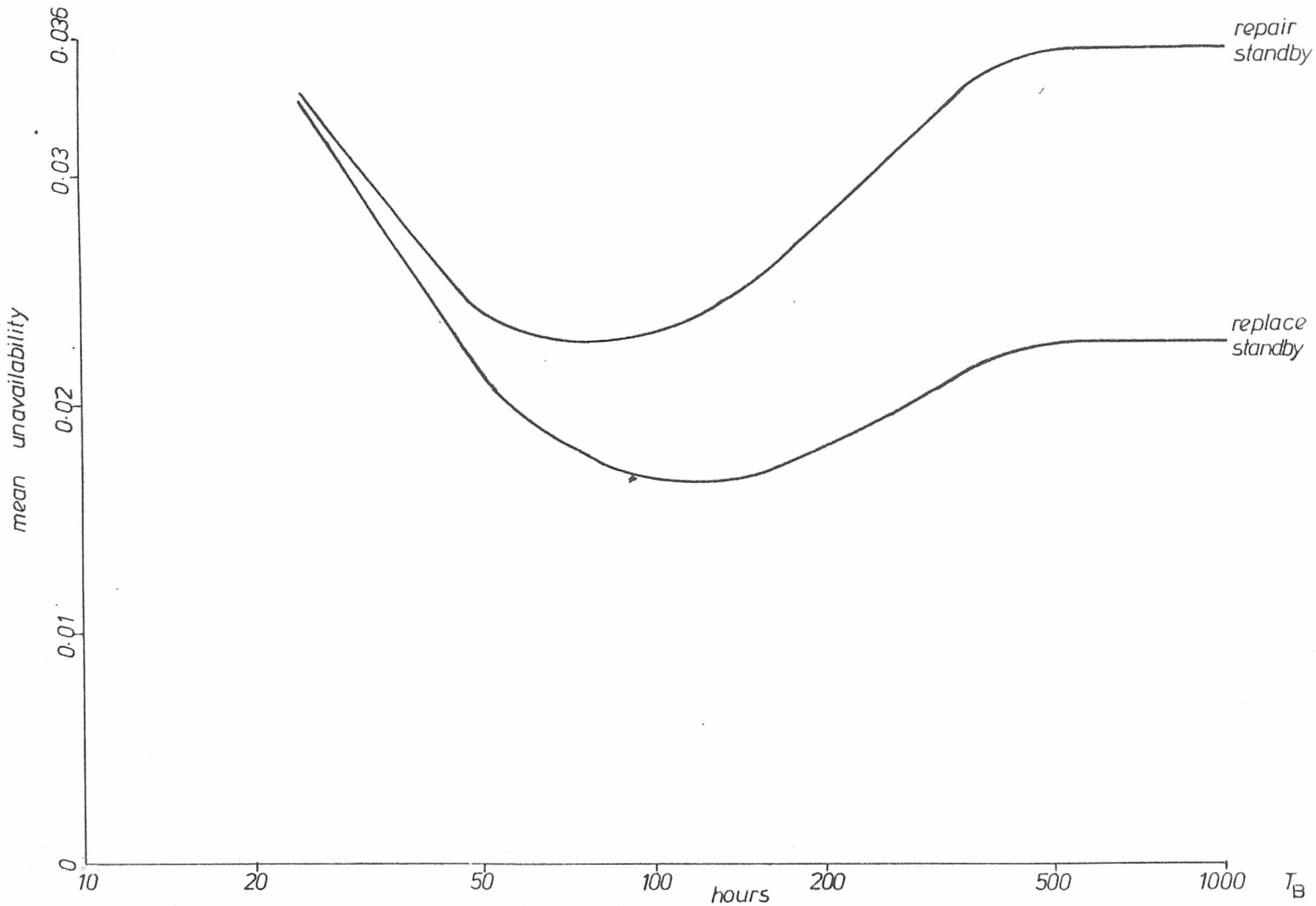


Figure 8·4d Unavailability of component 4 and cold standby component 5 with block replacement

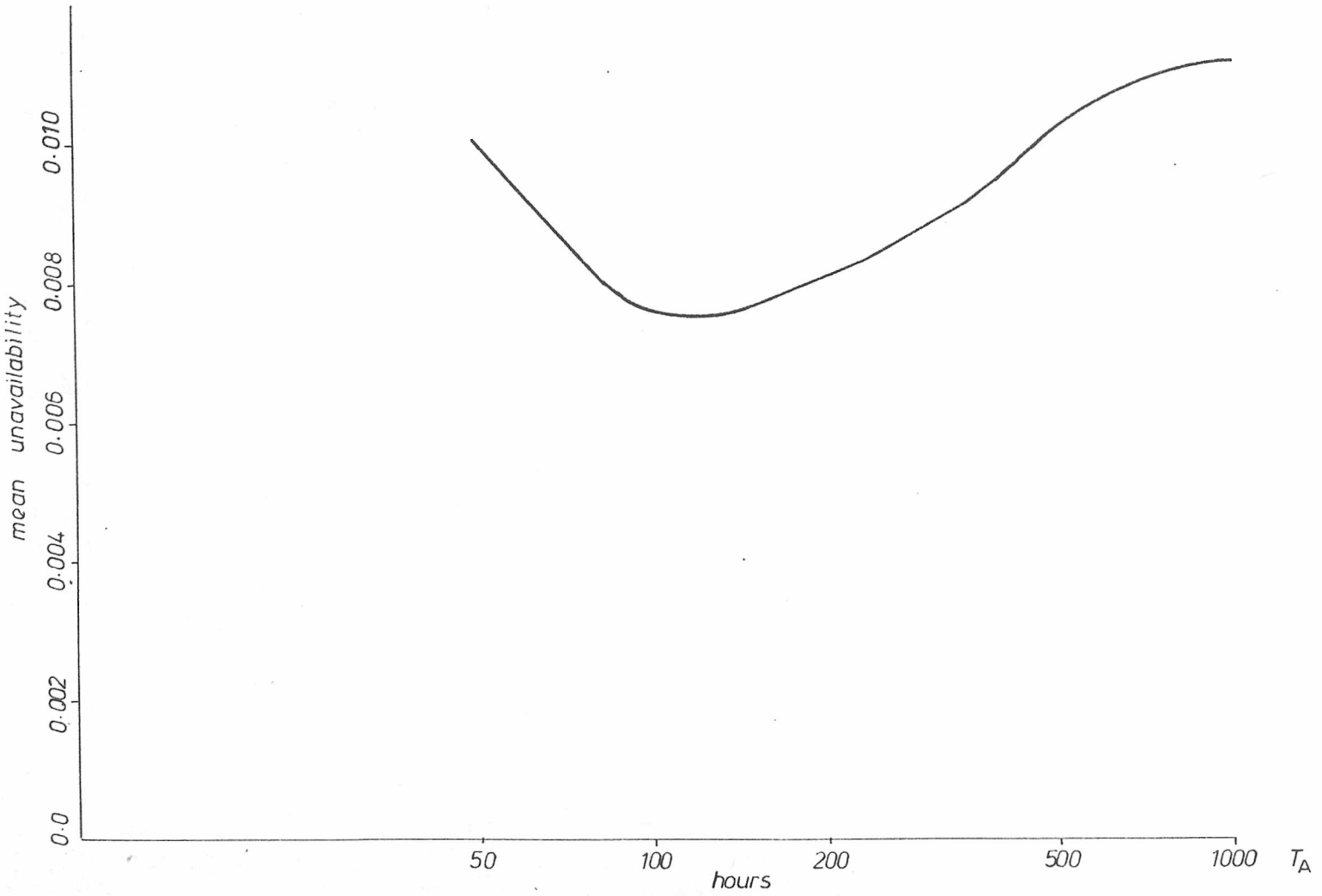


Figure 8·4e System unavailability with age replacement policy

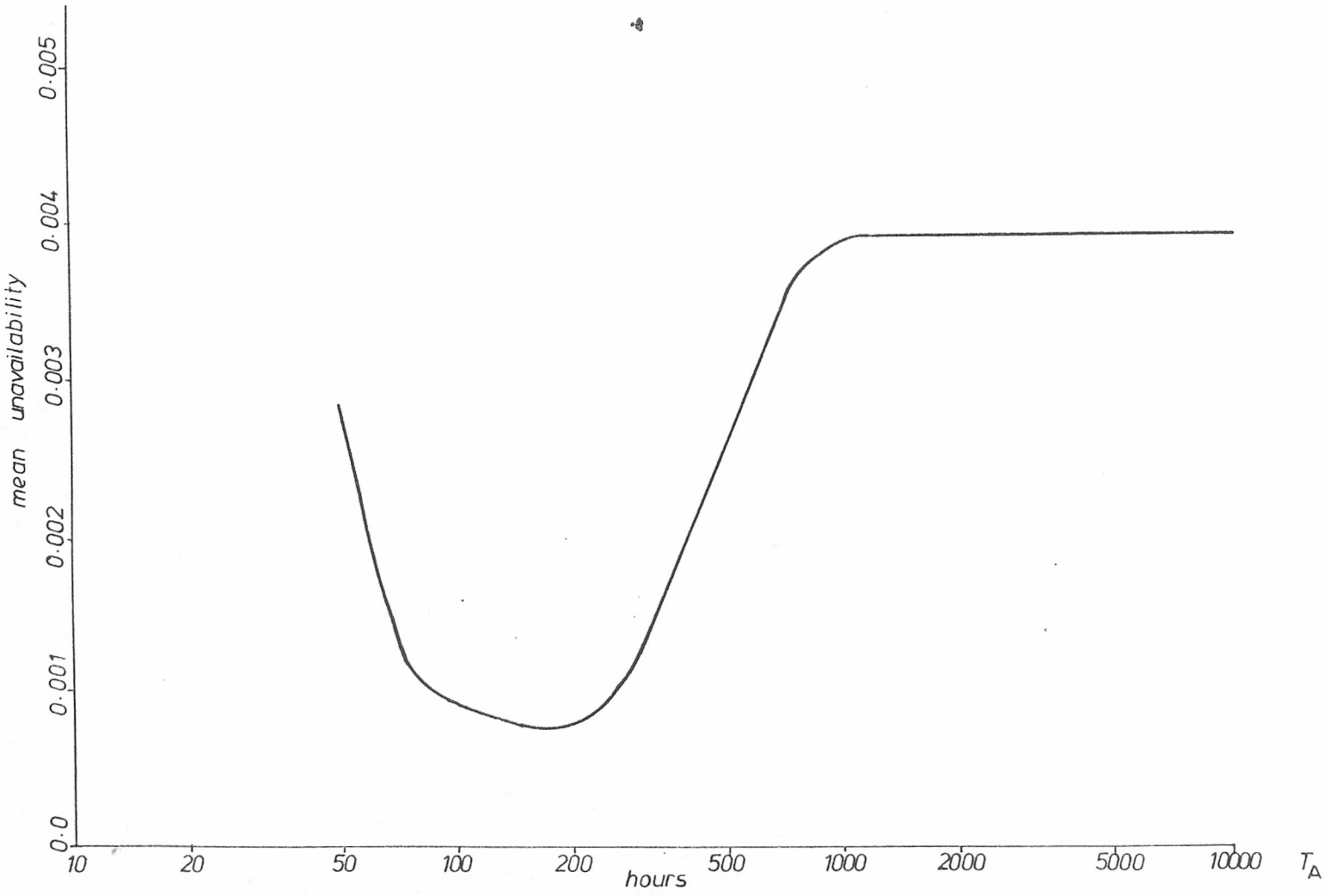


Figure 8·4f Mean unavailability at P3 with age replacement policy

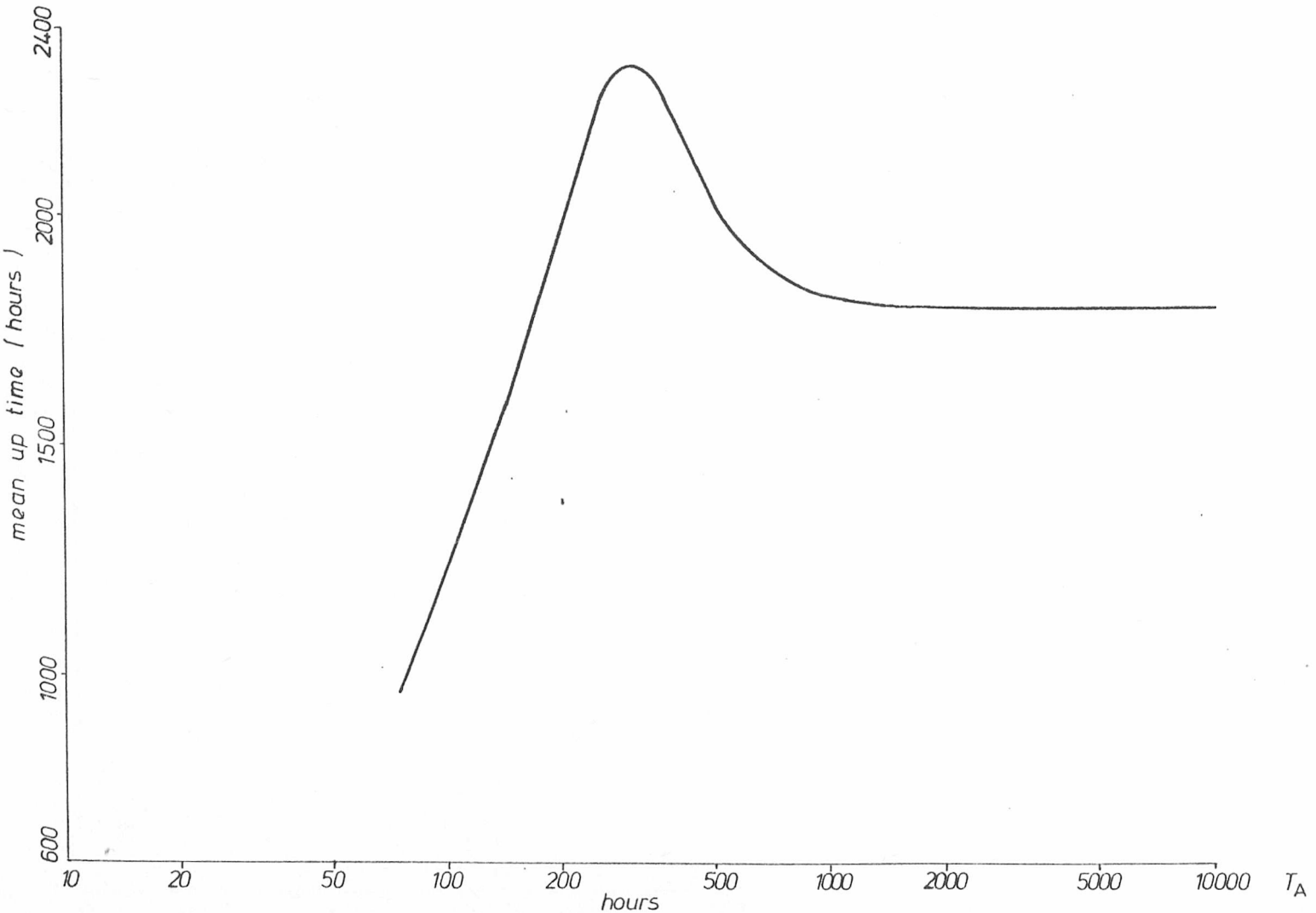


Figure 8.4g Mean up time at P3 with age replacement

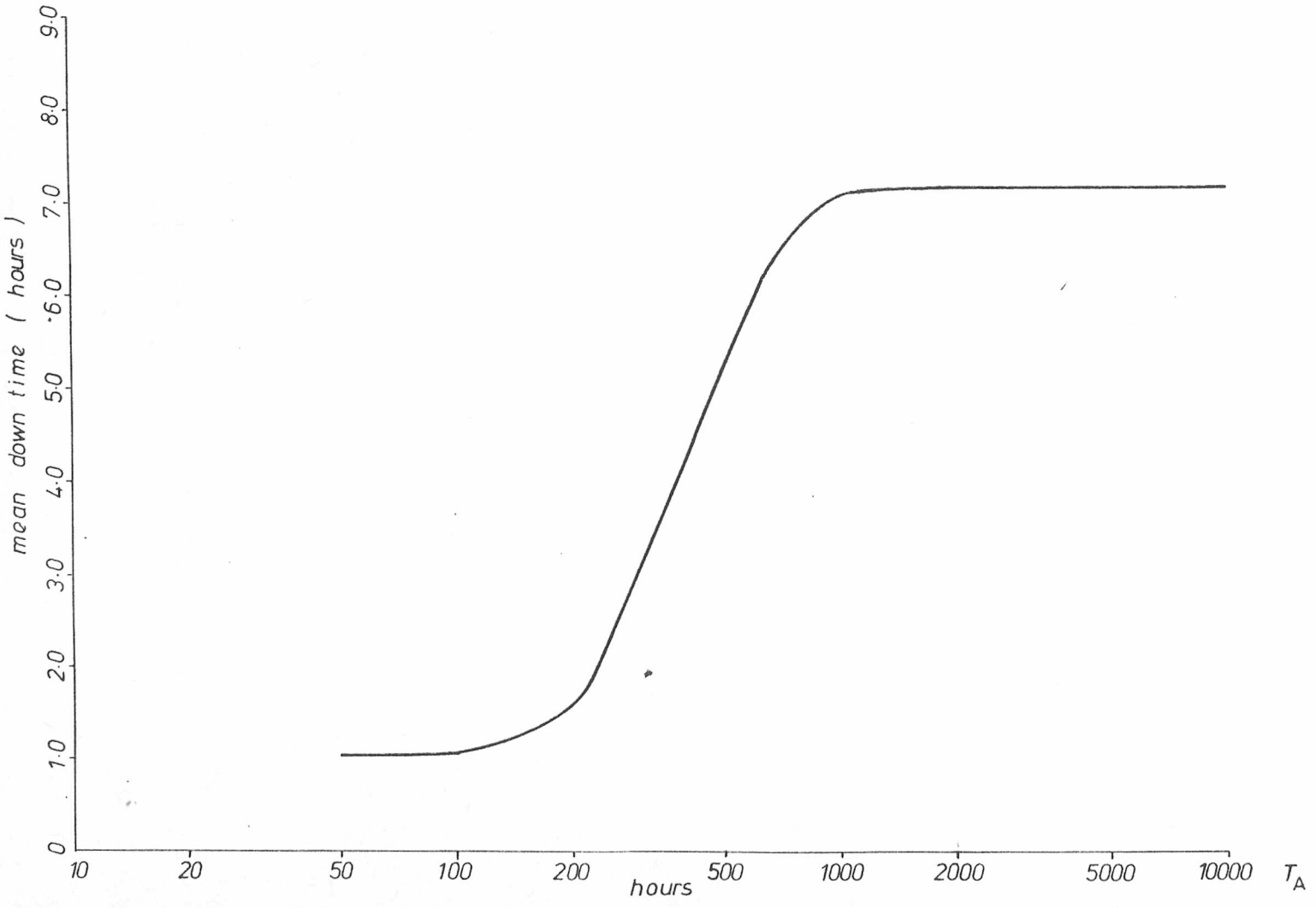


Figure 8·4h Mean down time at P3 with age replacement

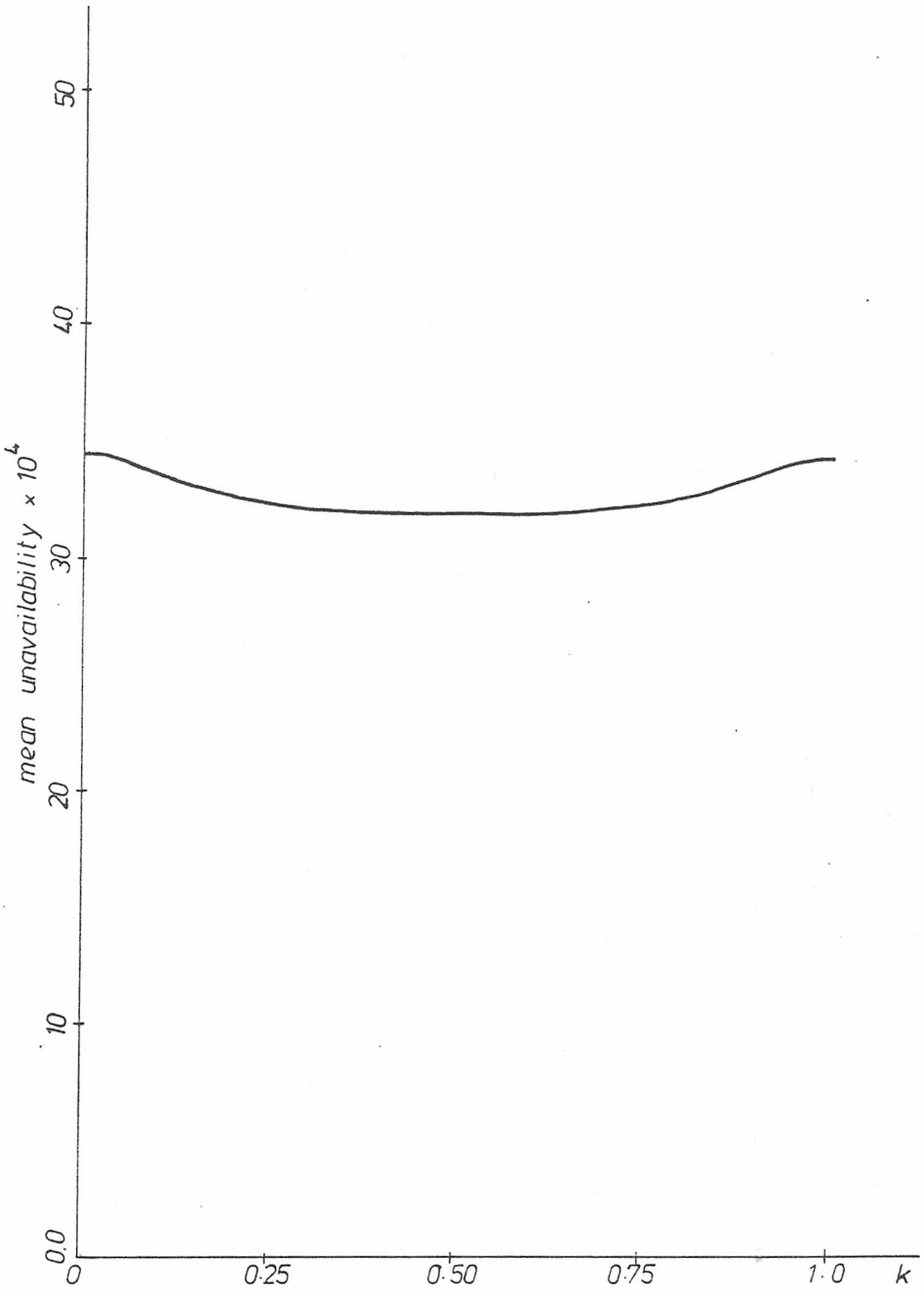


Figure 8.4i Mean unavailability with block replacement T_B = 1000 hours

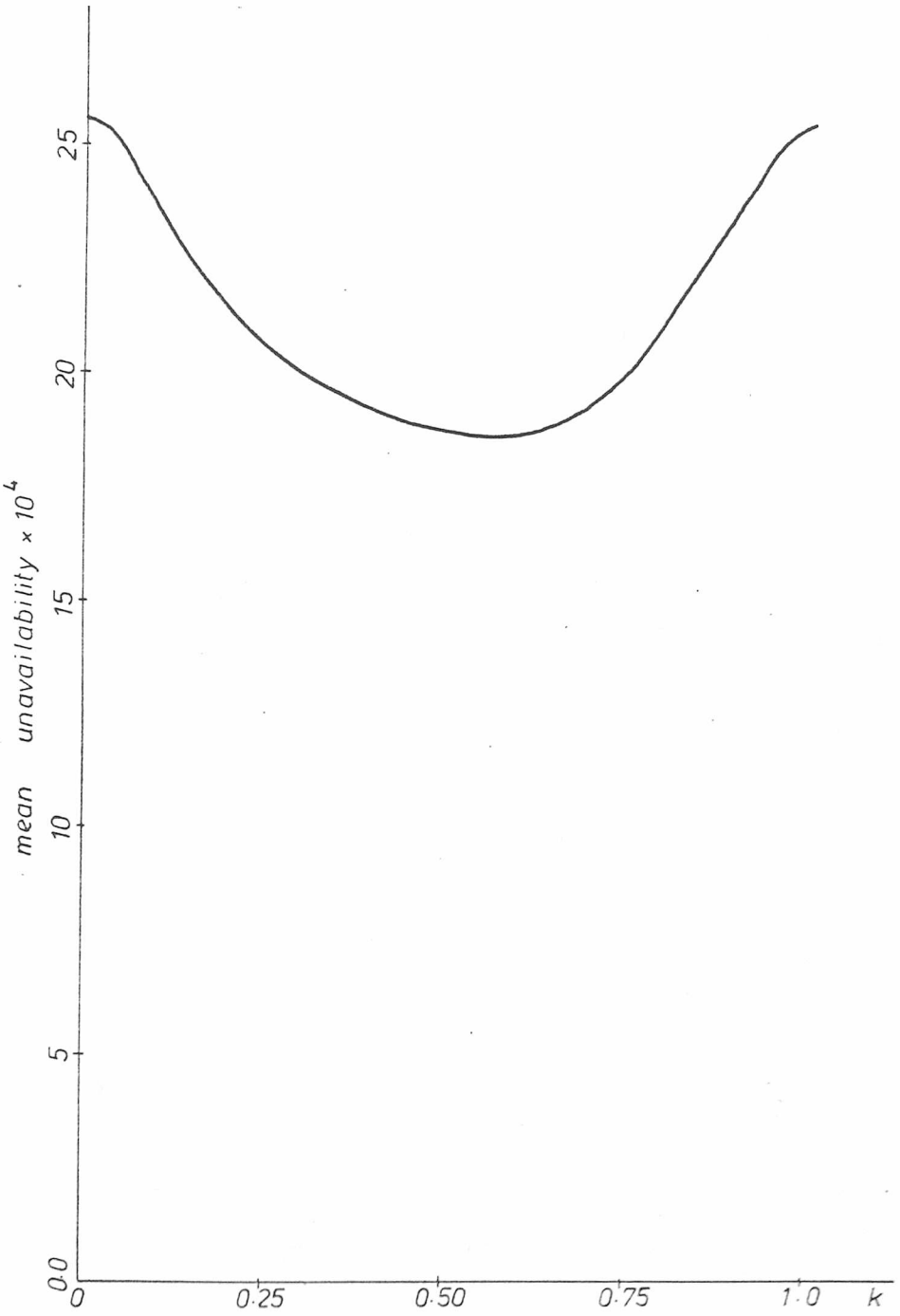


Figure 8.4j Mean unavailability with block replacement T_B = 500 hours

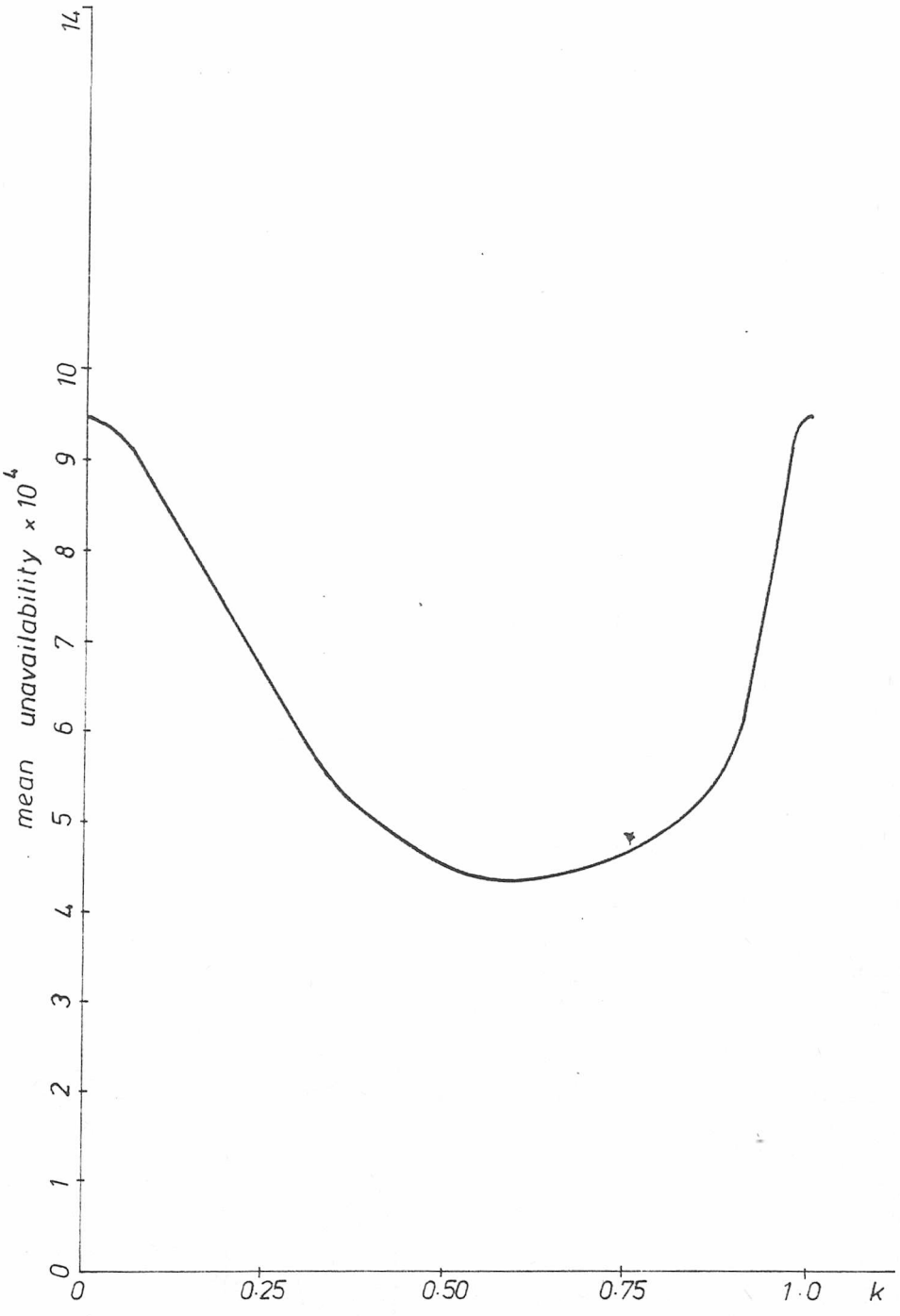


Figure 8.4k Mean unavailability with block replacement T_B = 200 hours

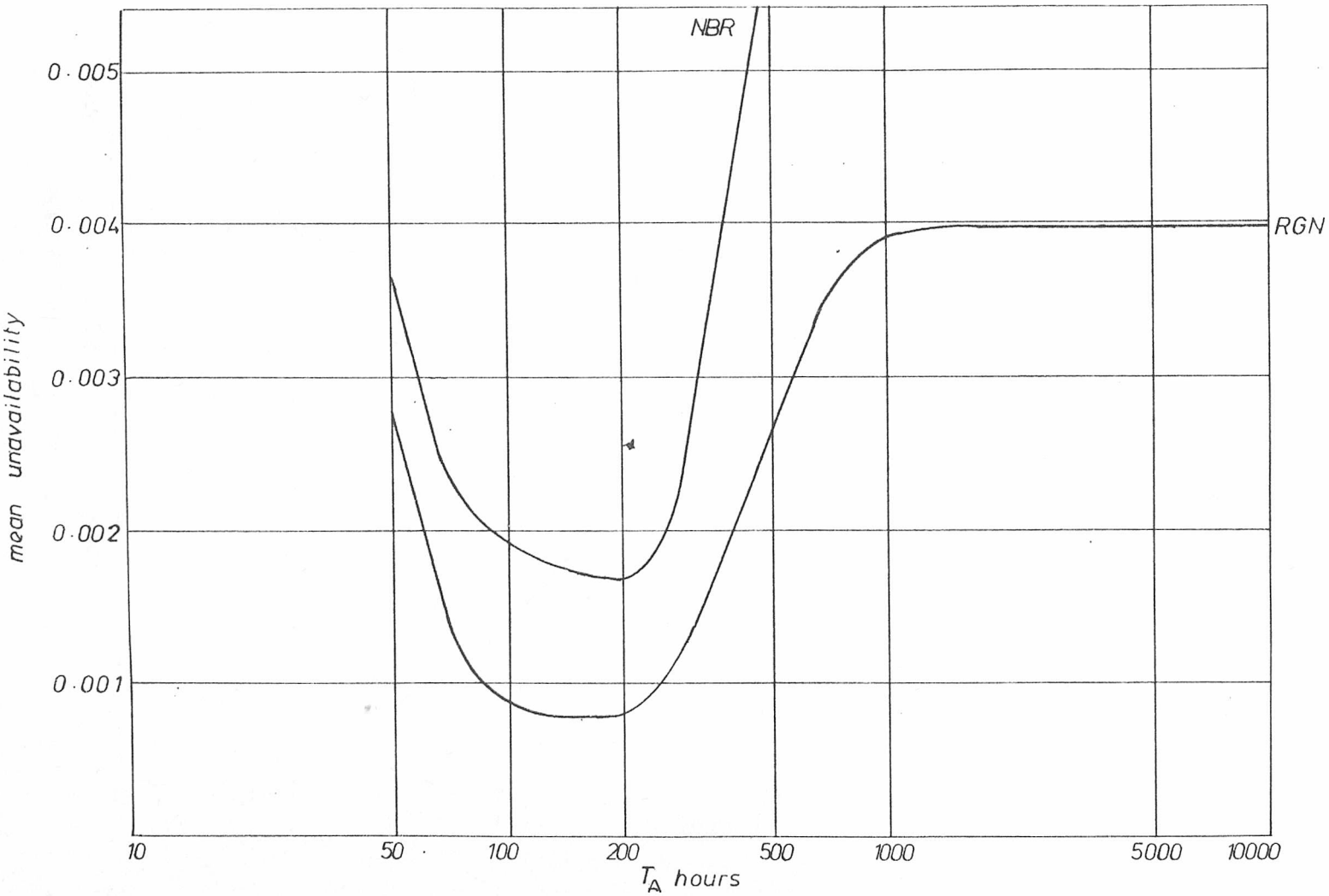
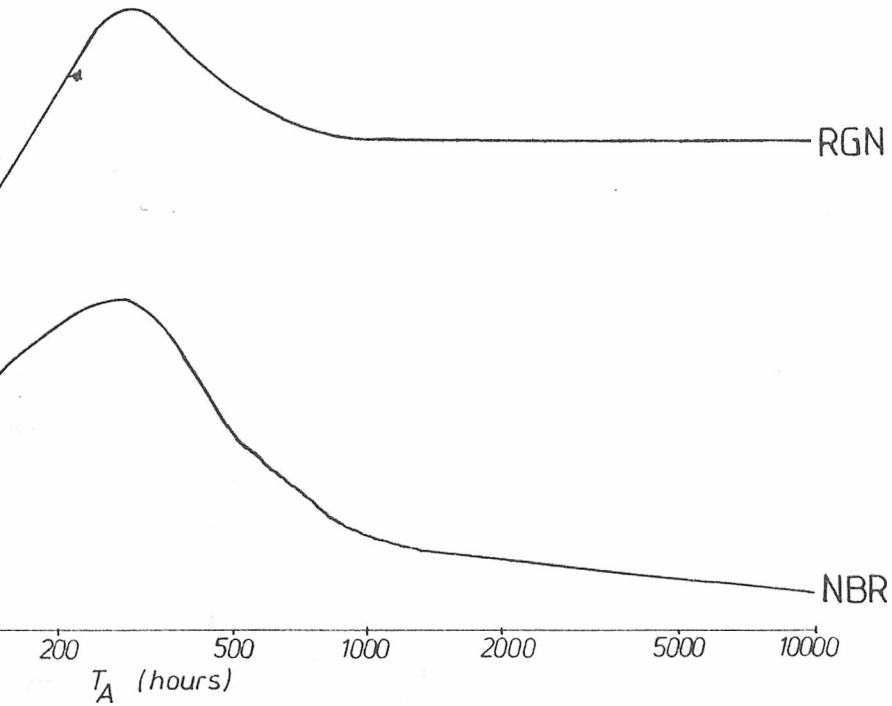


Figure 8.5a Age replacement with minimal repair

Figure 8·5b



Age replacement with minimal and perfect repair

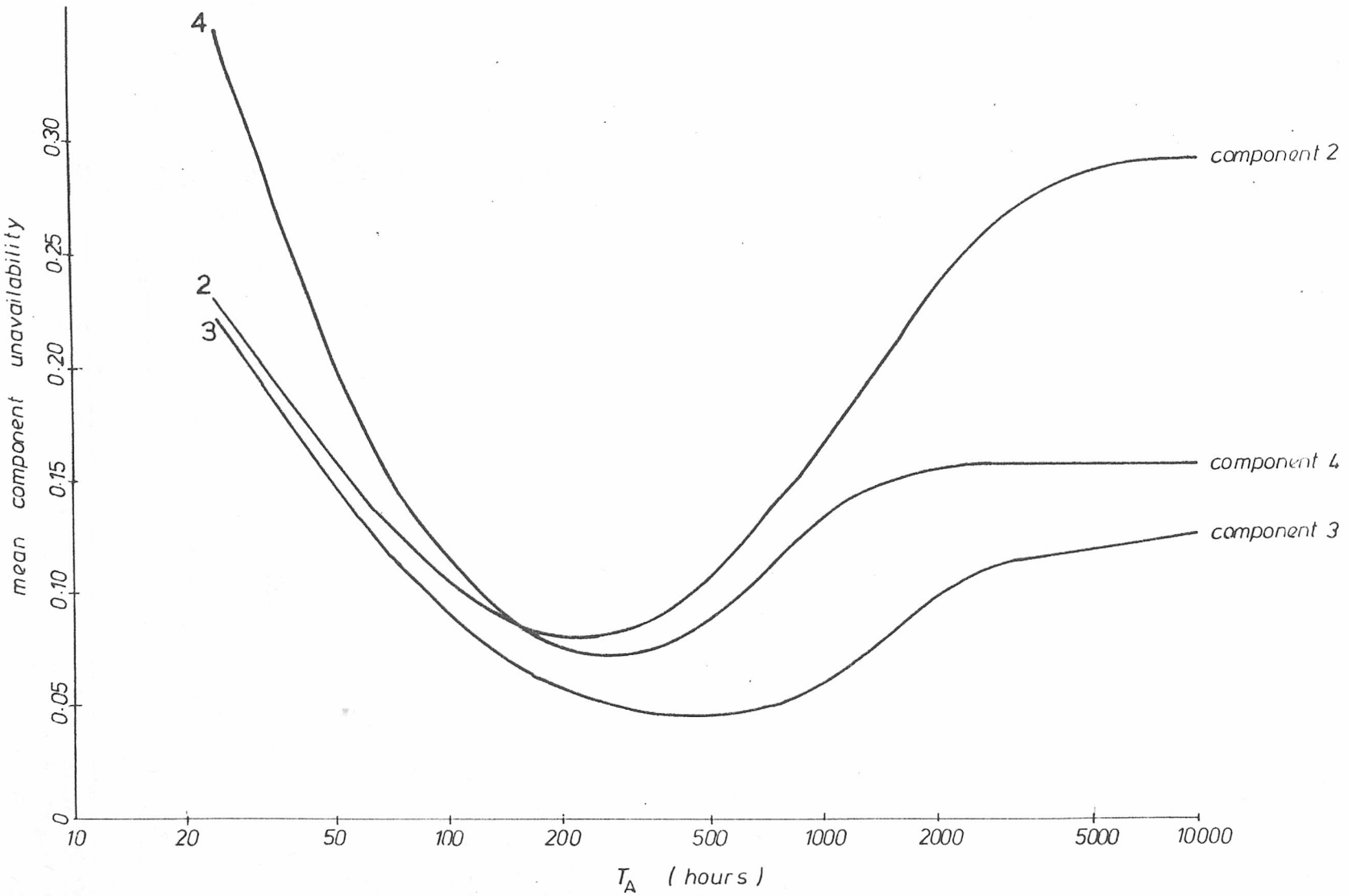


Figure 8·6a Component μ_D considering age replacement and NBR

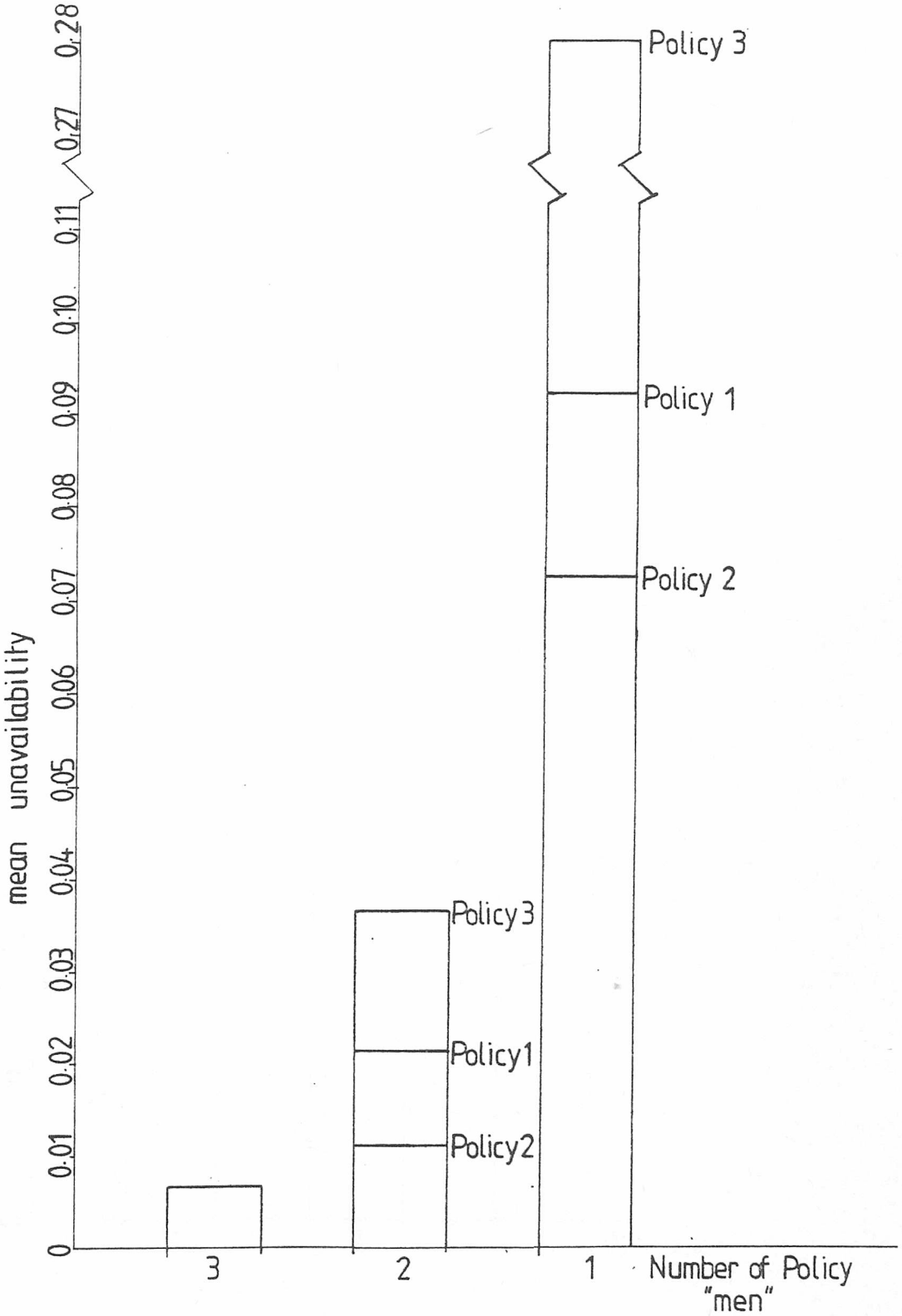


Figure 8.6b μ_D considering age replacement for various management policies

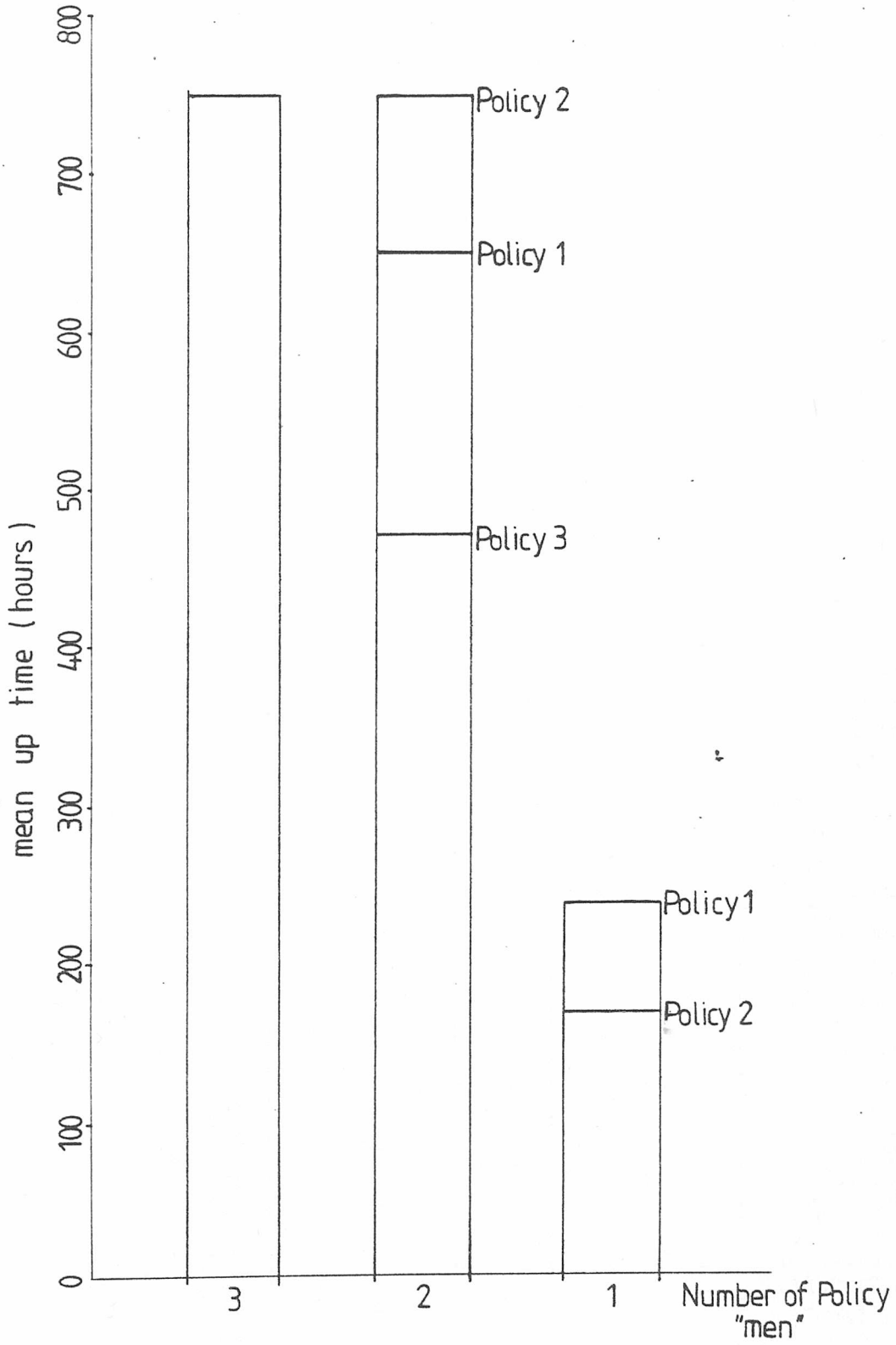


Figure 8.6c Mean up time considering age replacement for various management policies

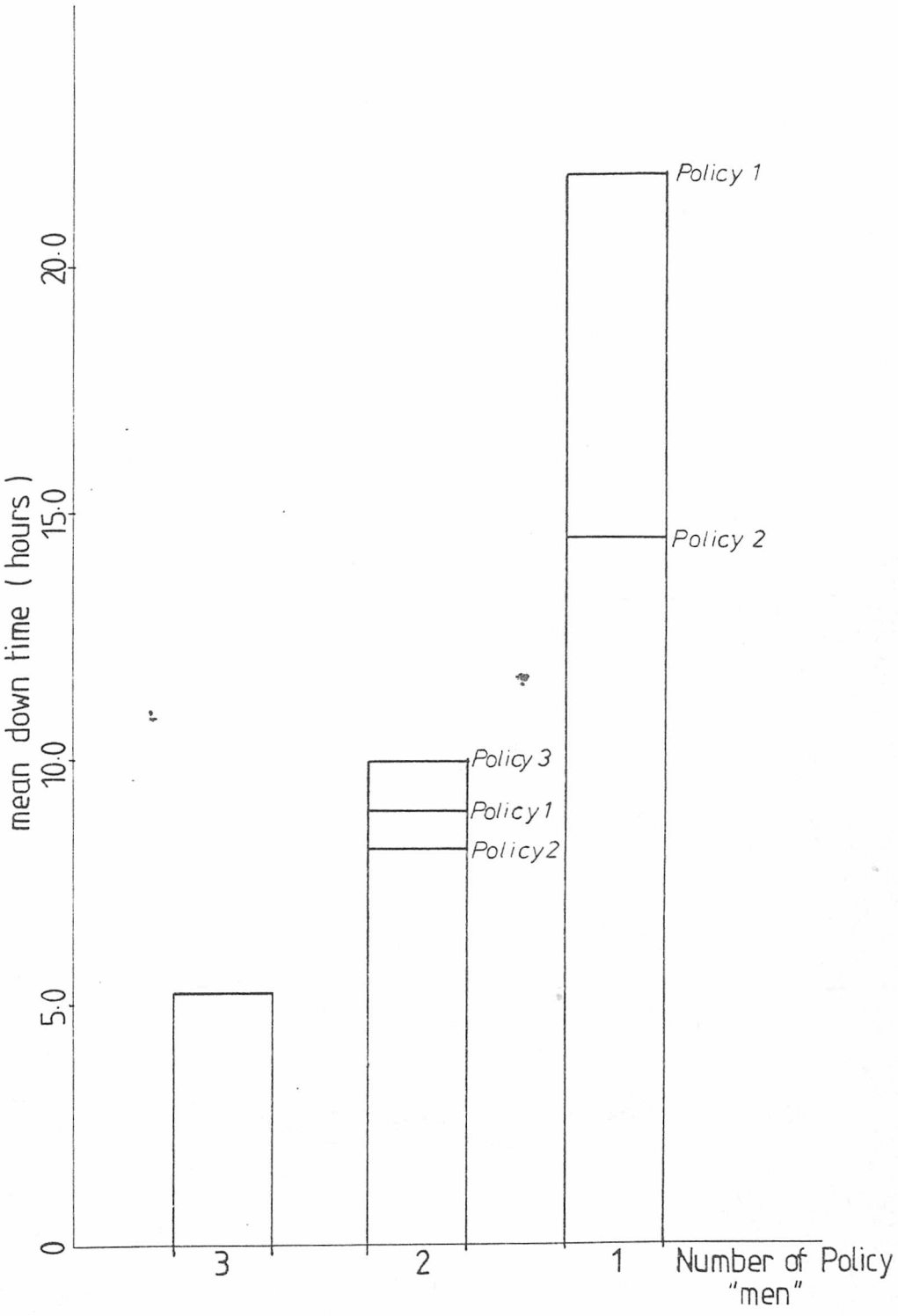


Figure 8.6d Mean down time considering age replacement for various management policies

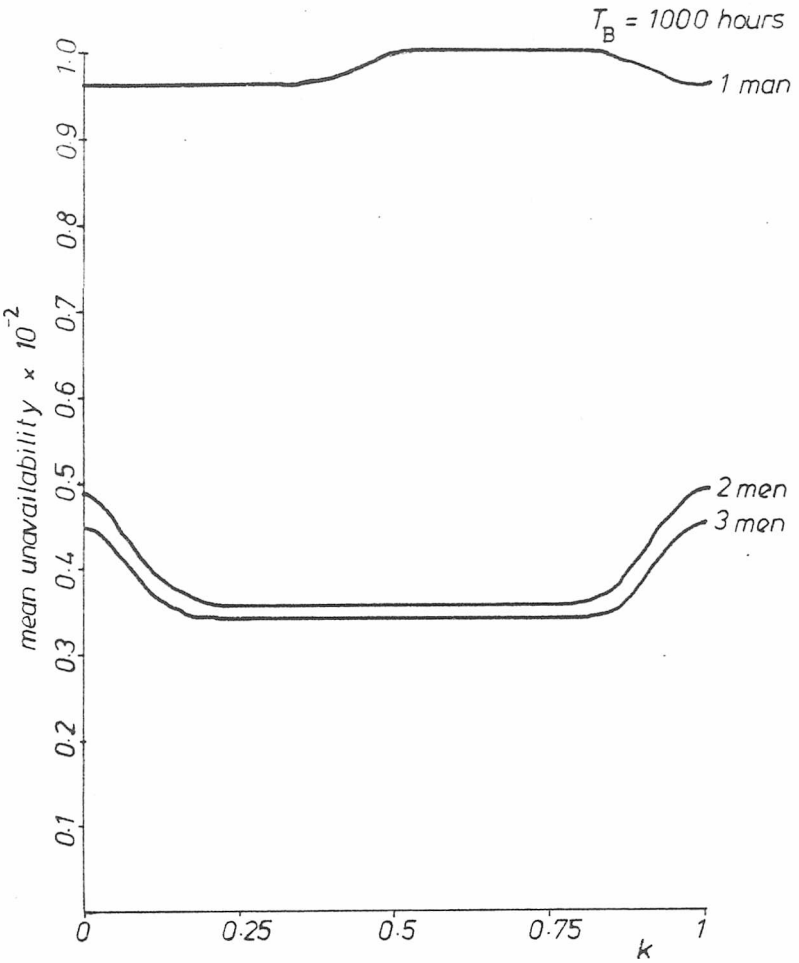


Figure 8·6e System unavailability considering block replacement and variable resources under Policy 1

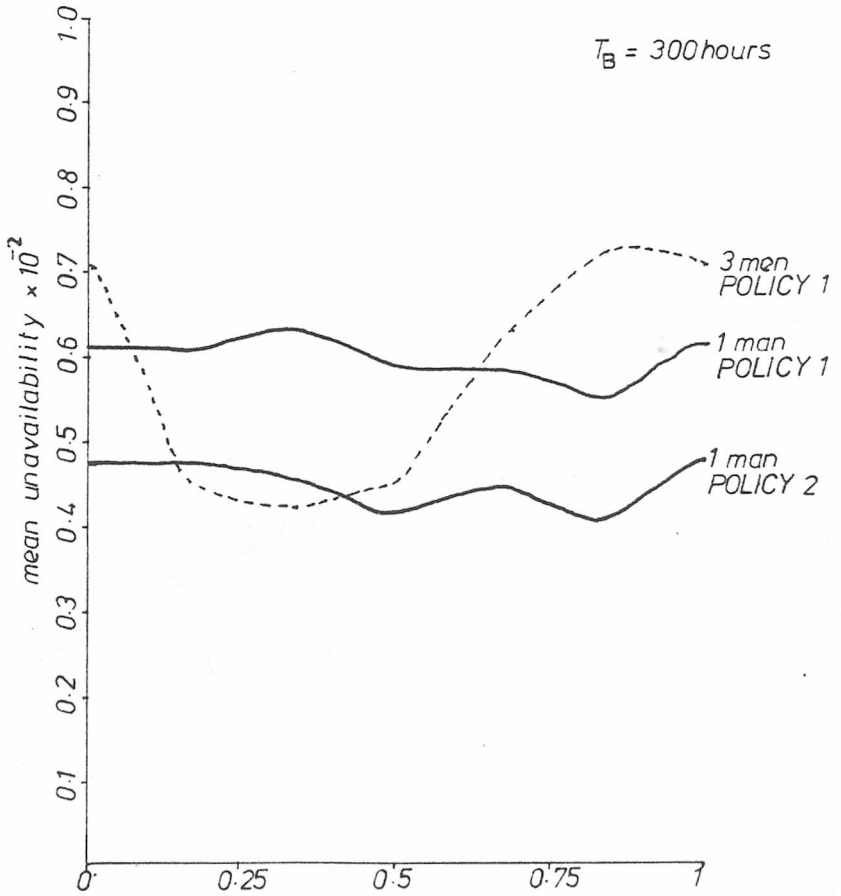


Figure 8·6f System unavailability considering block replacement and various Policy conditions

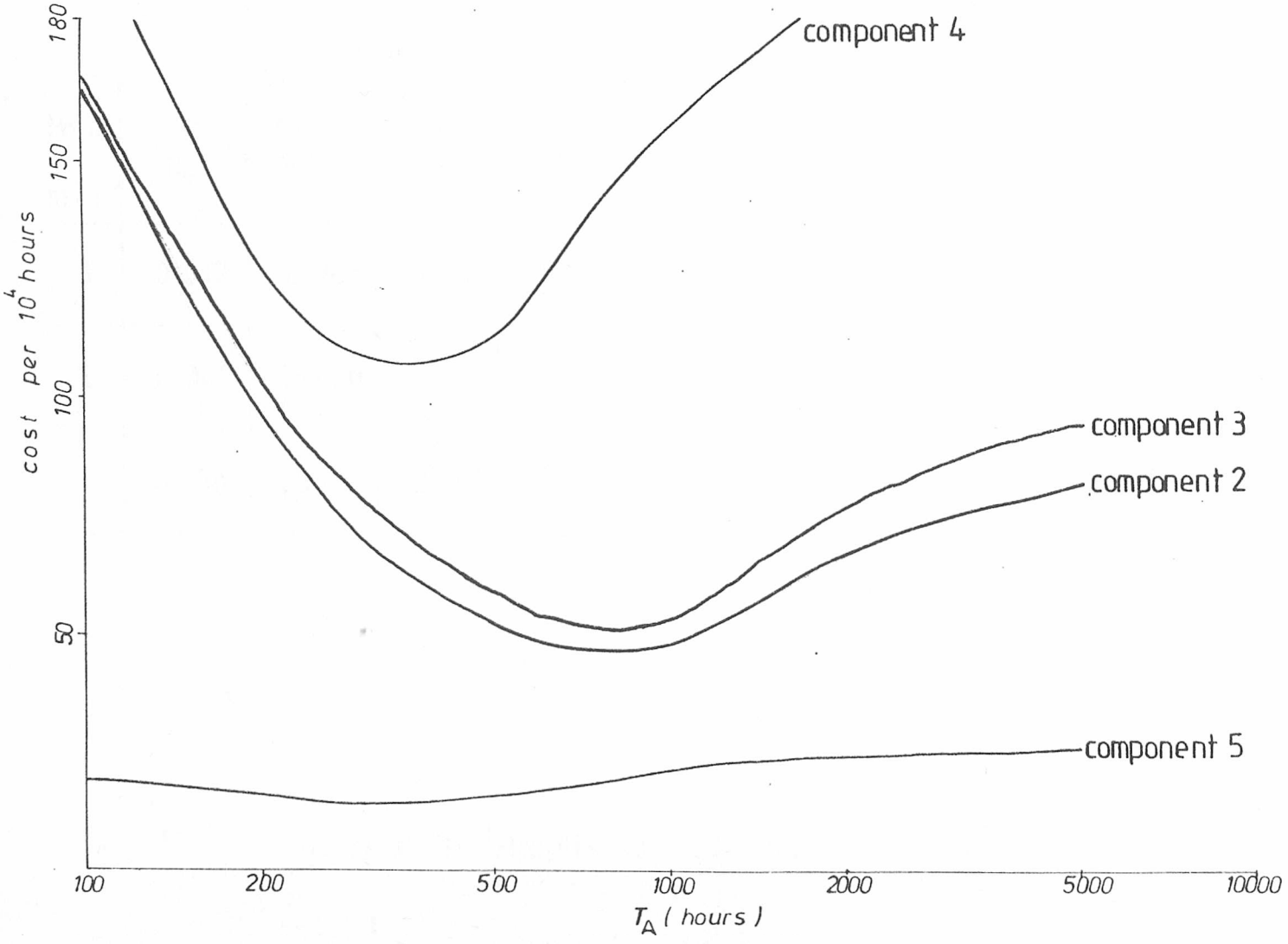


Figure 8.7a Cost of age replacement policy

No. of men	system behaviour: μ_D	mean down time	mean up time	No. of failure events	component behaviour: 2 m / R	3 m / R	4 m / R	5 m / R	action cost /10^4h	down time/10^4h operation
3	0·012	6·38h	522·7	189	11·3	11·2	22·2	-	217·1	121 h
					23·0	27·6	62·6	14·5		
2	0·044	16·06h	348·9	274	9·7	5·6	12·4	-	144·9	440h
					25·7	13·9	32·8	17·1		
1	0·100	25·75h	230·7	390	6·4	3·8	5·1	-	93·7	1004h
					18·4	9·5	13·3	21·9		

events are per 10^4 hours operation

m = component maintaince event

R = component repair event

Figure 8·7b Results for cost analysis of example system

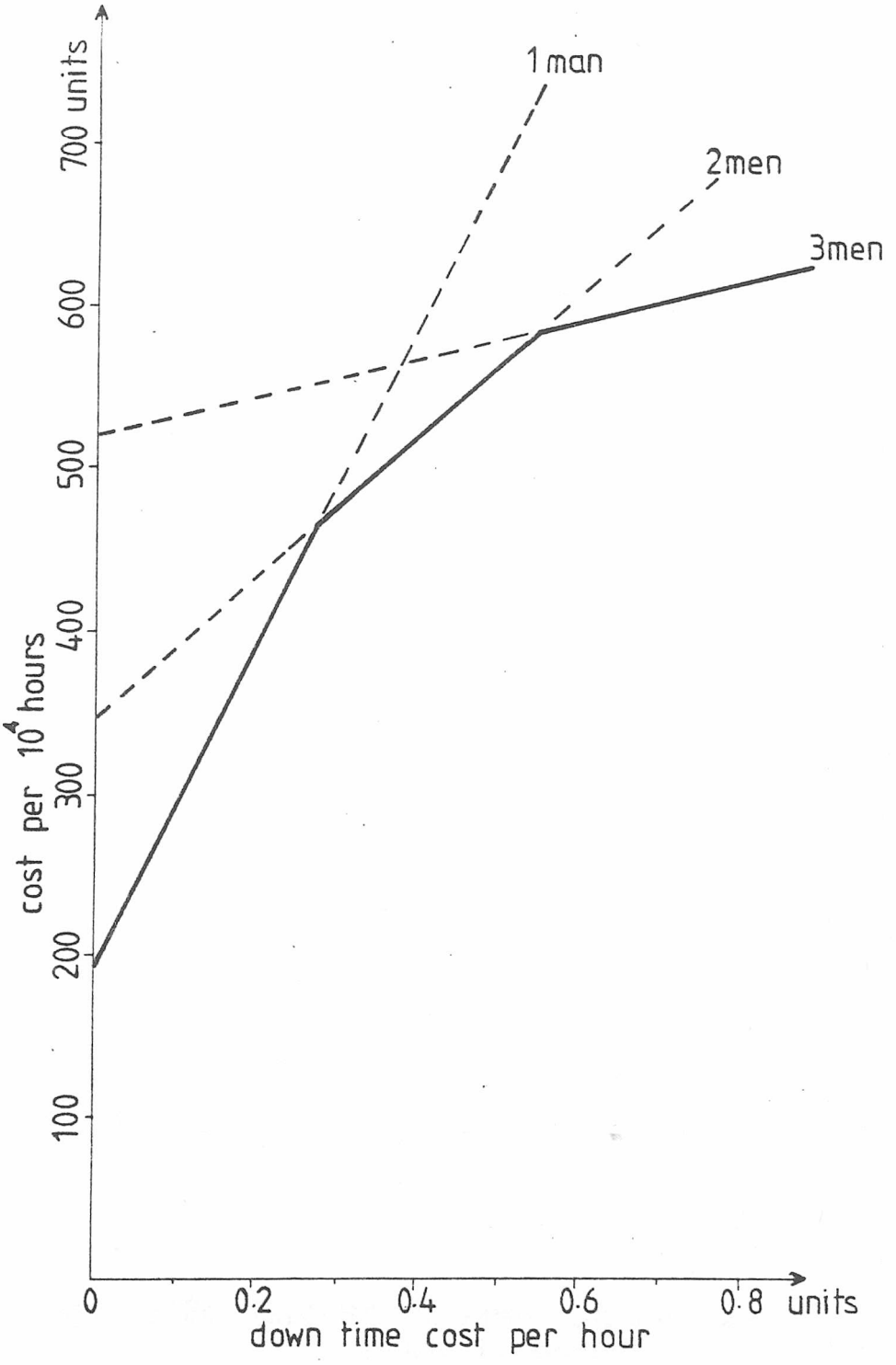


Figure 8·7c Cost of system operation

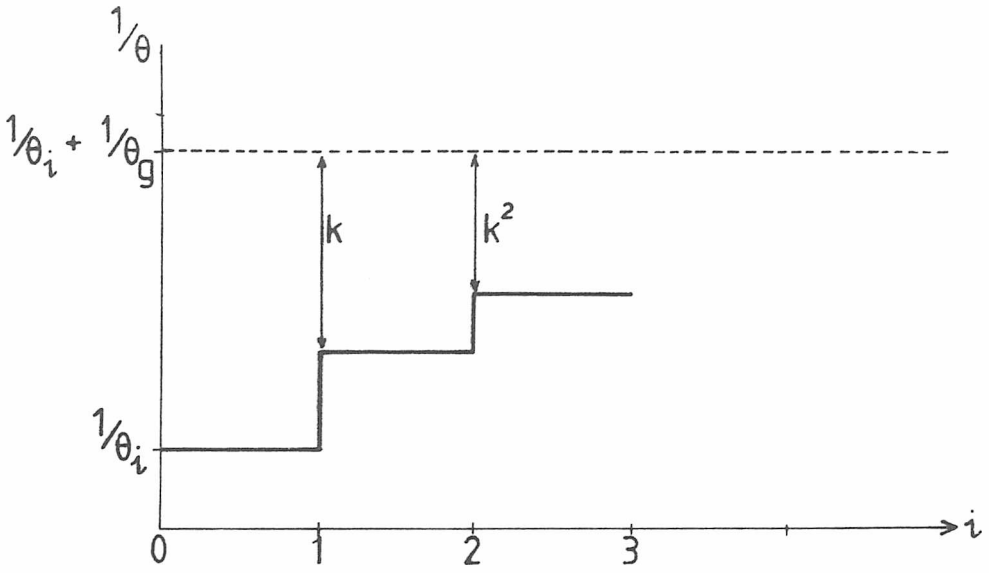


Figure 8·8a Expected component life time with reliability growth

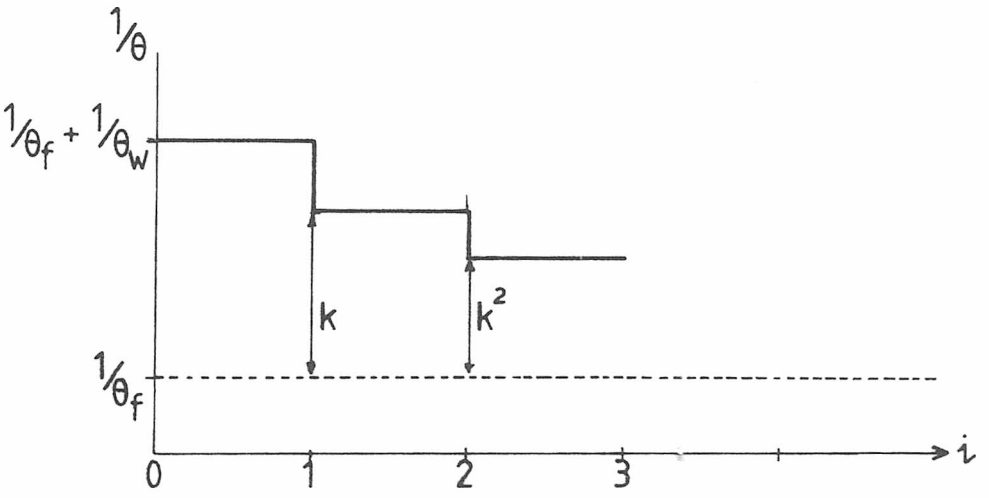
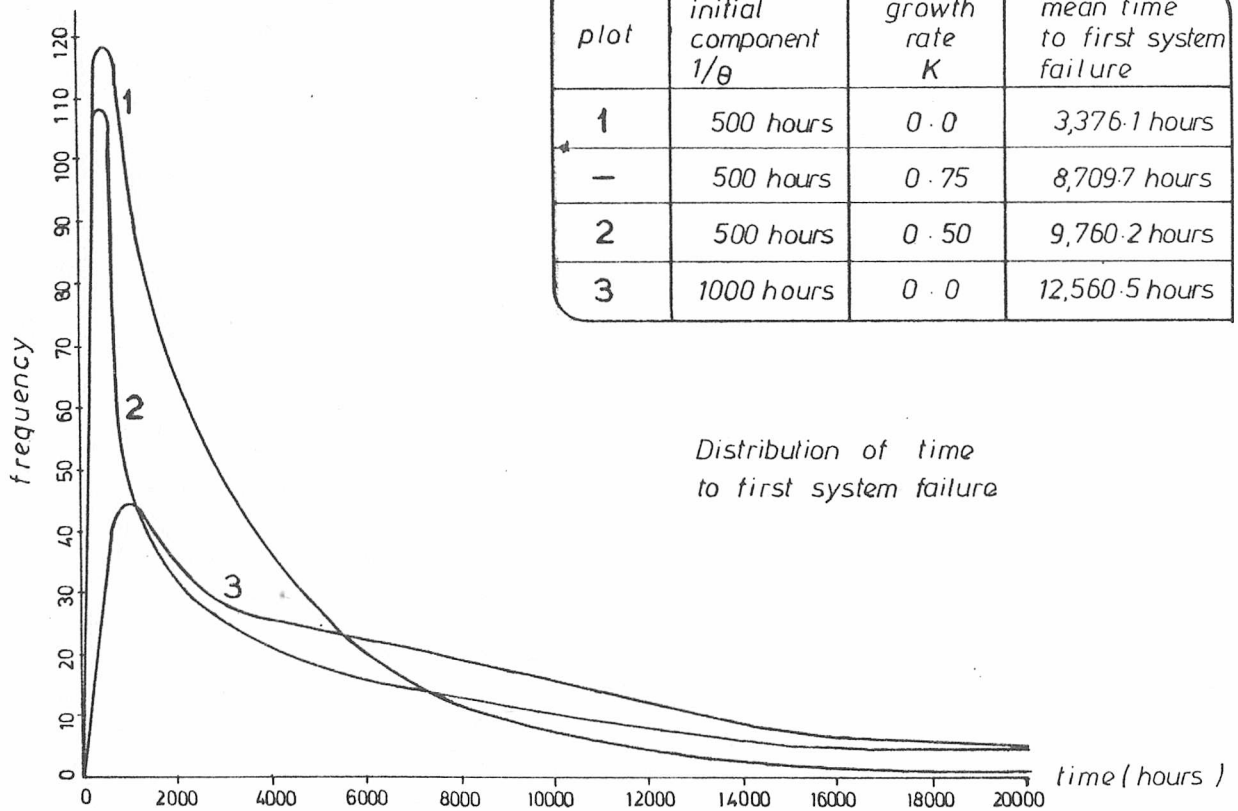


Figure 8·8b Expected component life time with reliability wearout

plot	initial component $1/\theta$	growth rate K	mean time to first system failure
1	500 hours	0·0	3,376·1 hours
—	500 hours	0·75	8,709·7 hours
2	500 hours	0·50	9,760·2 hours
3	1000 hours	0·0	12,560·5 hours



Distribution of time to first system failure

Figure 8·8c 2 component redundant system with reliability growth

plot	initial life time $1/\theta$ =	wearout rate K	mean time to first system failure
1	1000 hours	0 · 0	12,560·5 hours
2	1000 hours	0 · 75	5,776·2 hours
3	1000 hours	0 · 50	5,382·2 hours
4	500 hours	0 · 0	3,376·1 hours

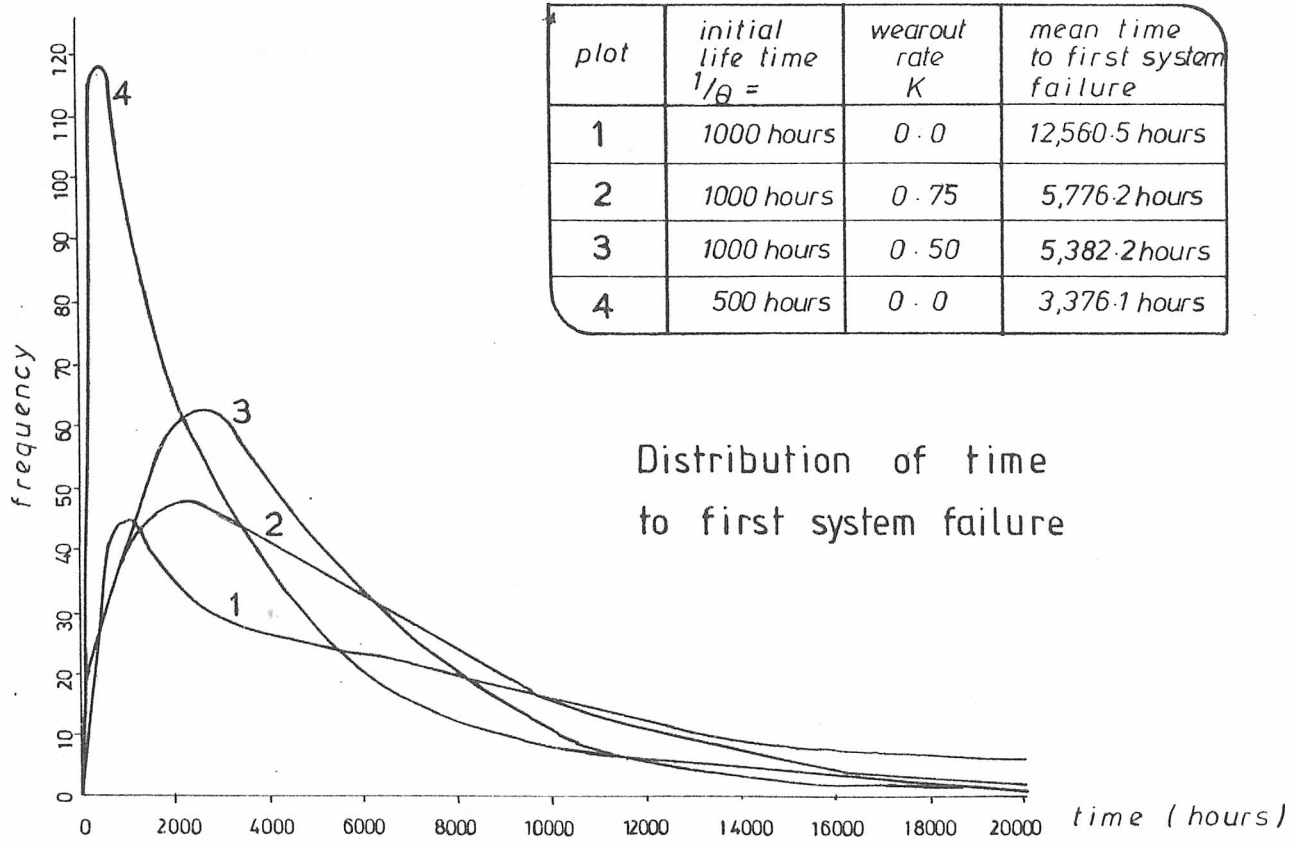


Figure 8·8d 2 component redundant system with reliability wearout

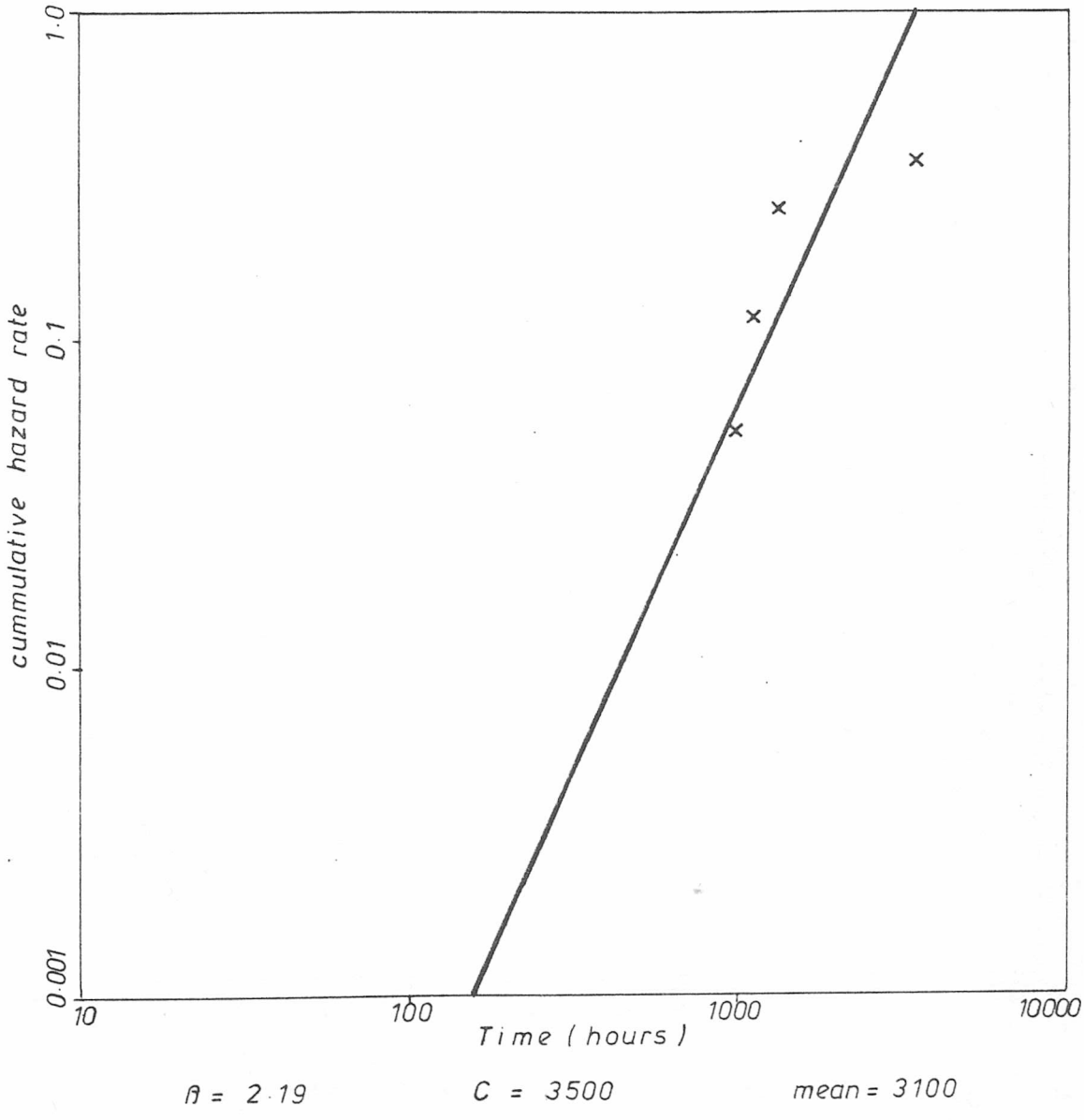
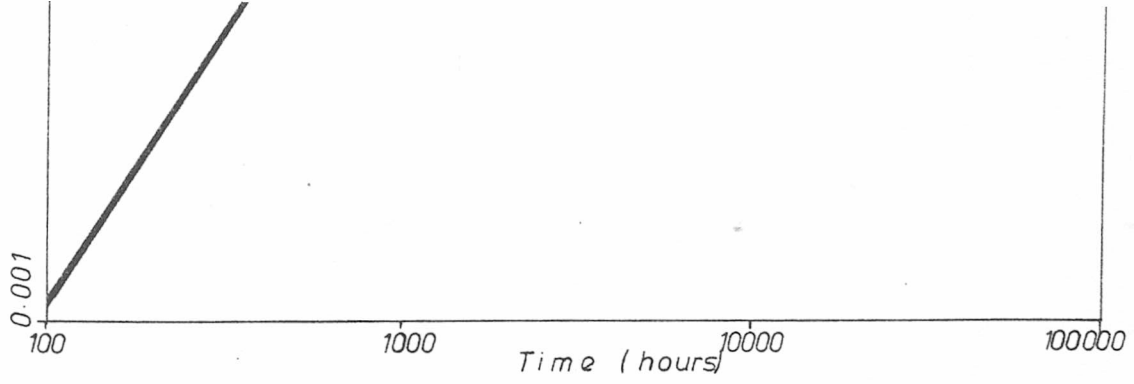
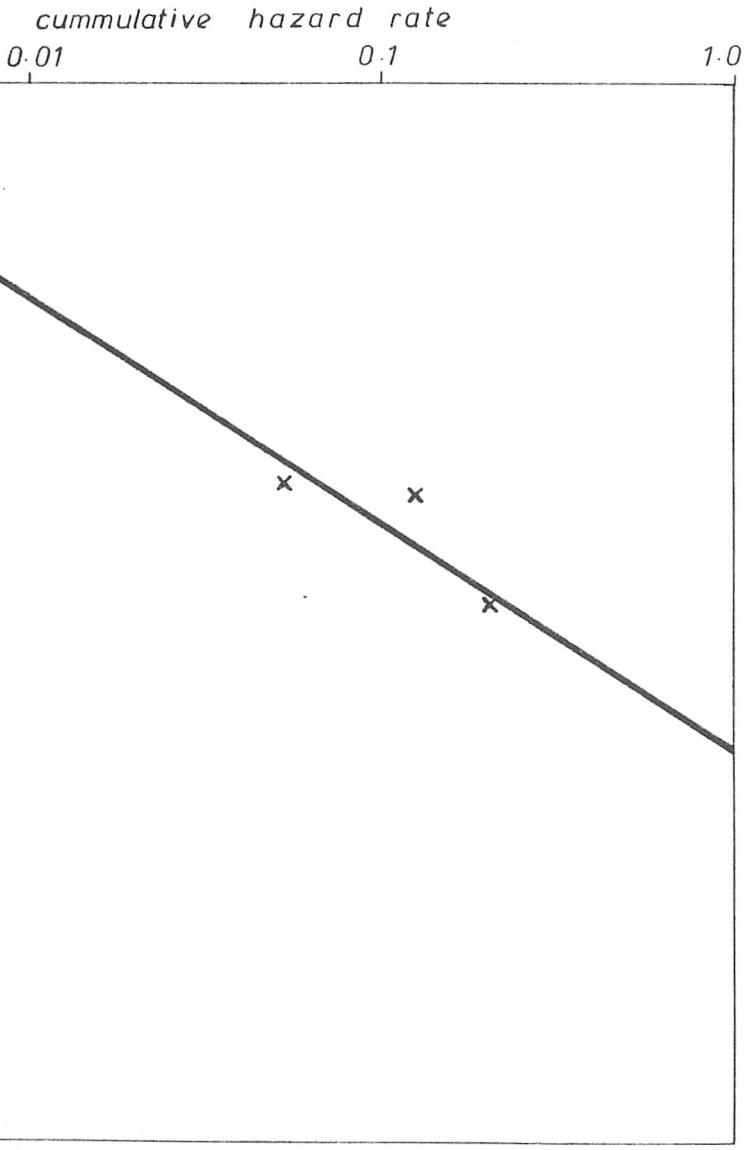


Figure 8·9a 1st stage suction 2nd mode only



$n = 1.53$ $C = 7800$ mean = 7030

Figure 8.9b 1st stage discharge 2nd mode only


cummulative hazard rate
0·01
0·1
1·0

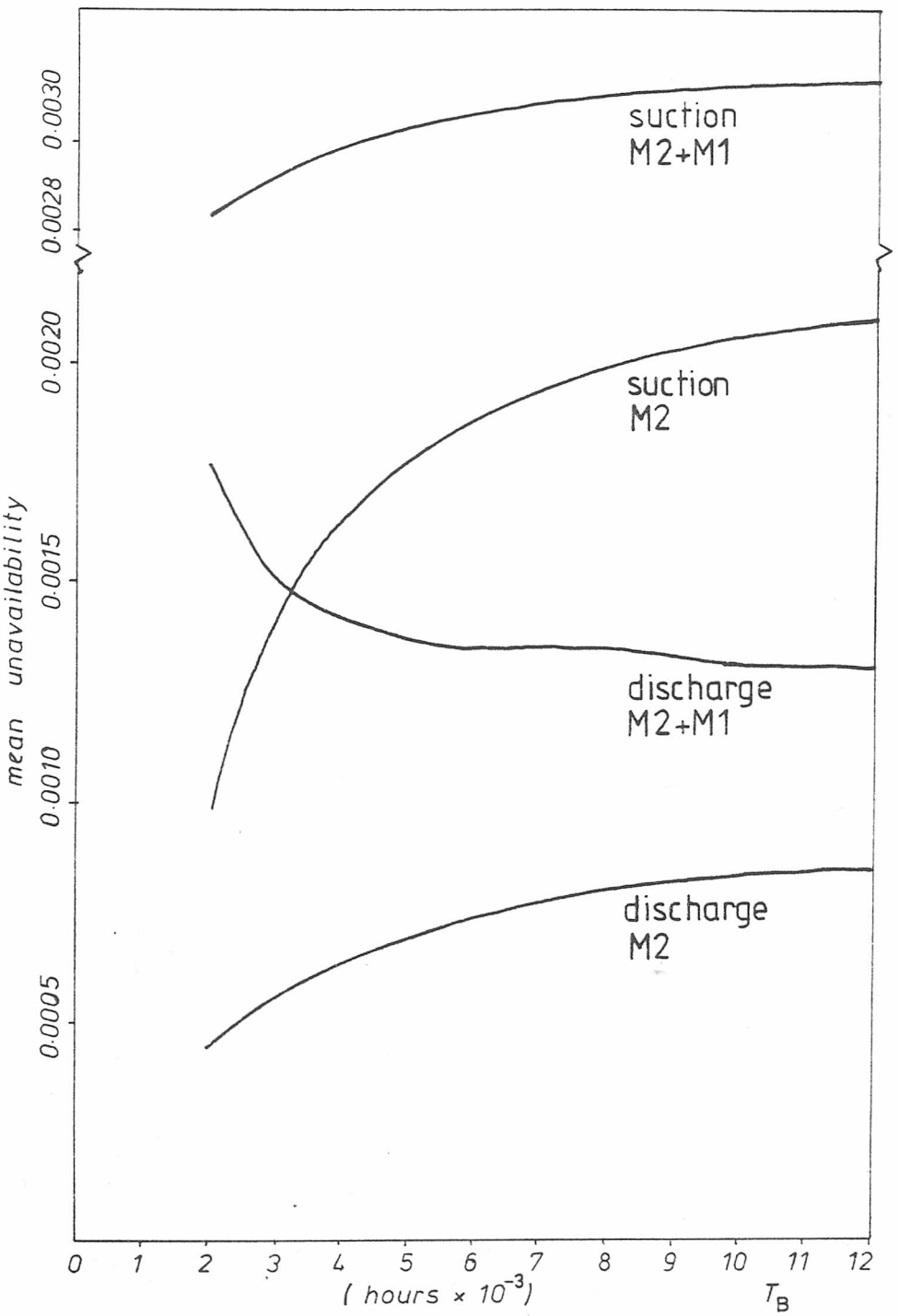


Figure 8·9c A single first stage valve with block replacement

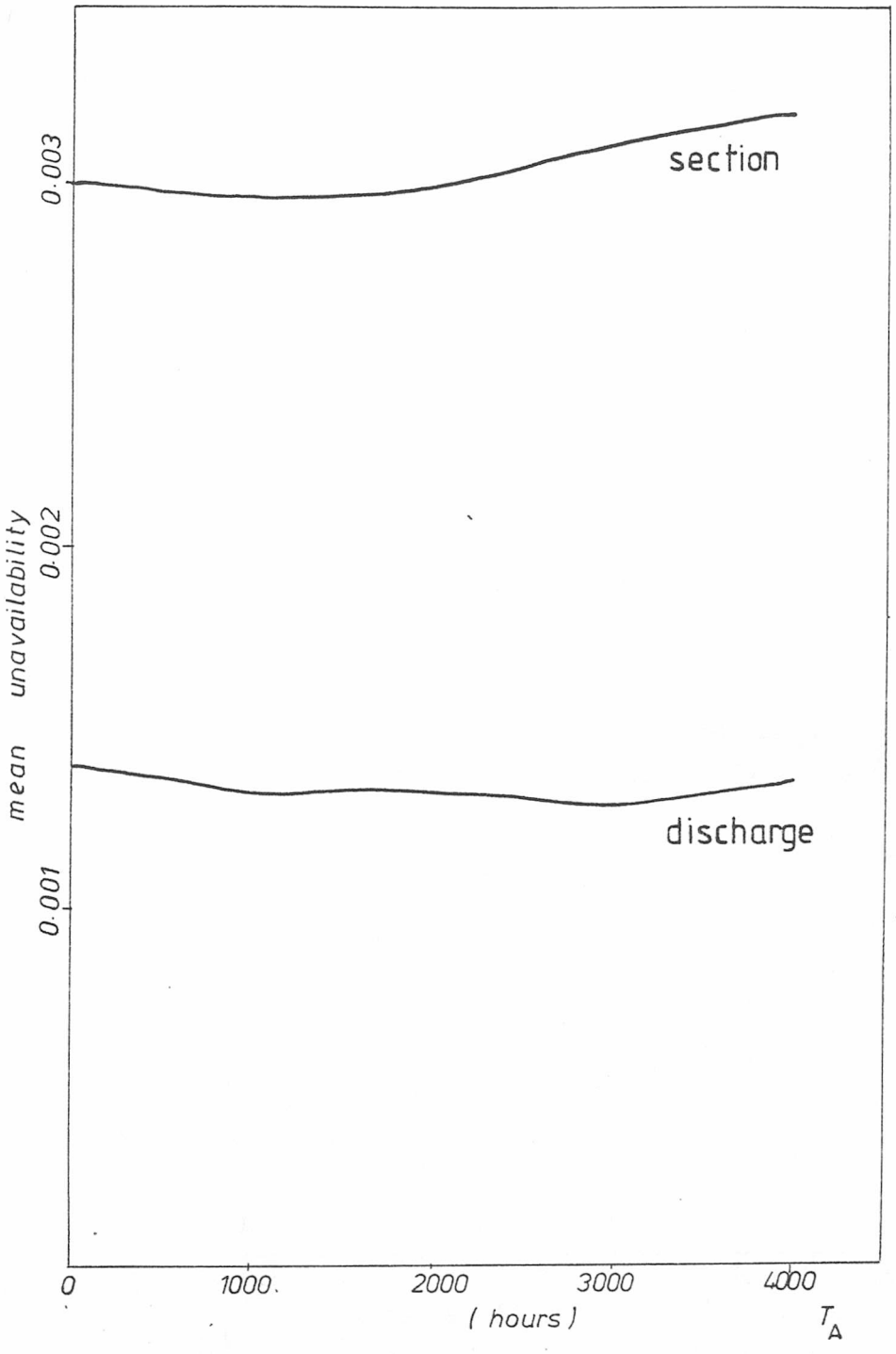


Figure 8.9d Block replacement at $T_B = 4500$ hours only if age over T_A

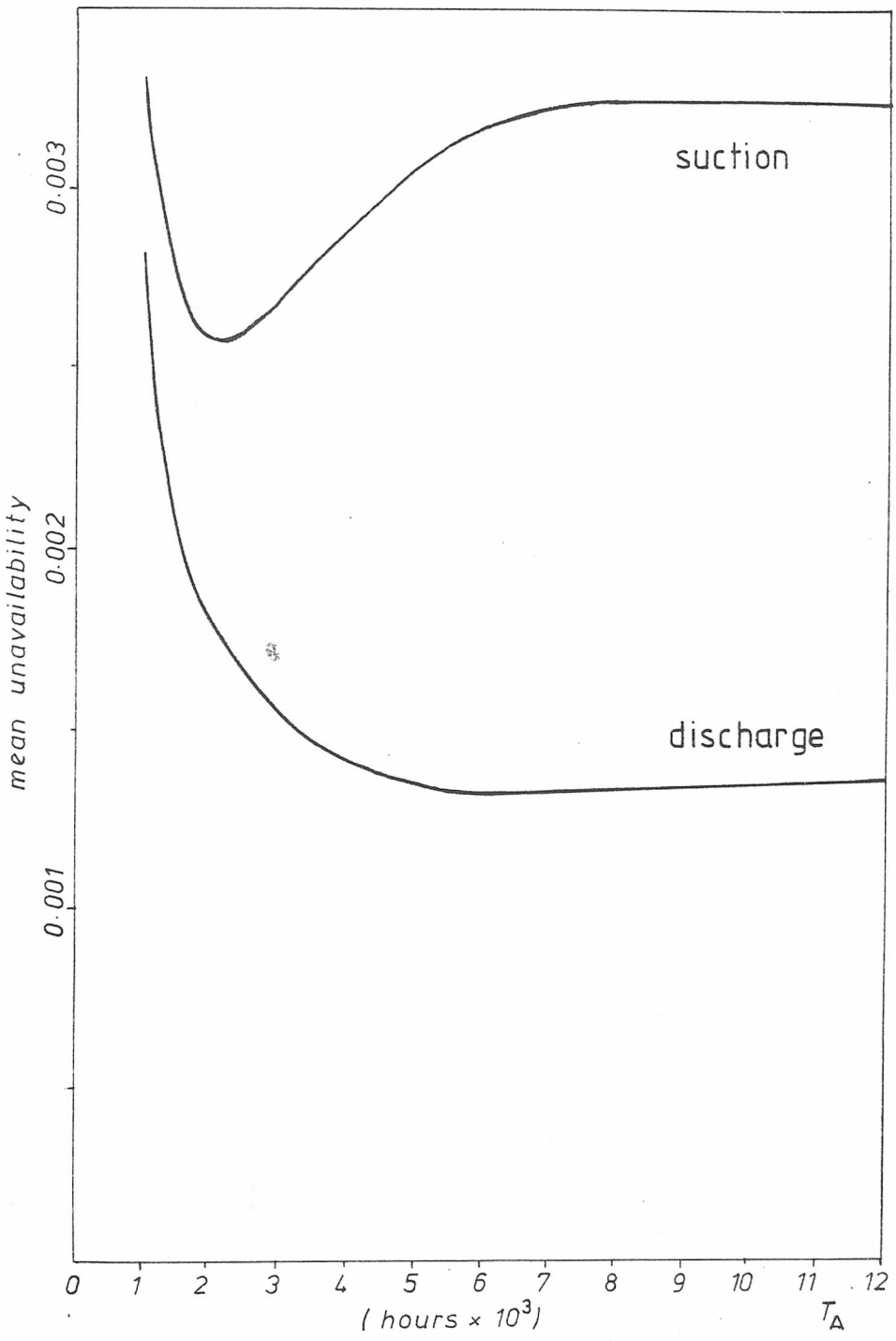


Figure 8·9e A single first stage valve with age replacement

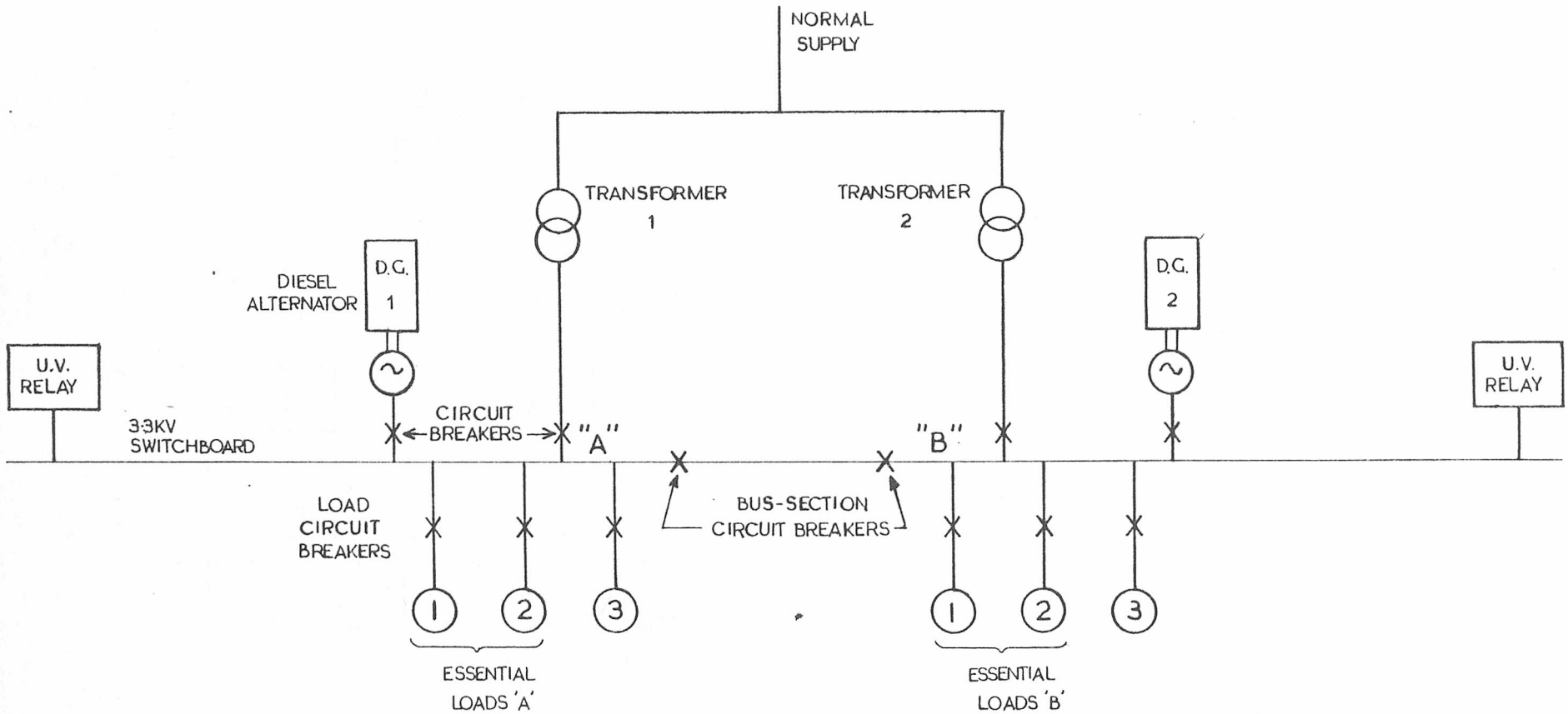


Figure 8·10a Schematic arrangement for standby power supply system

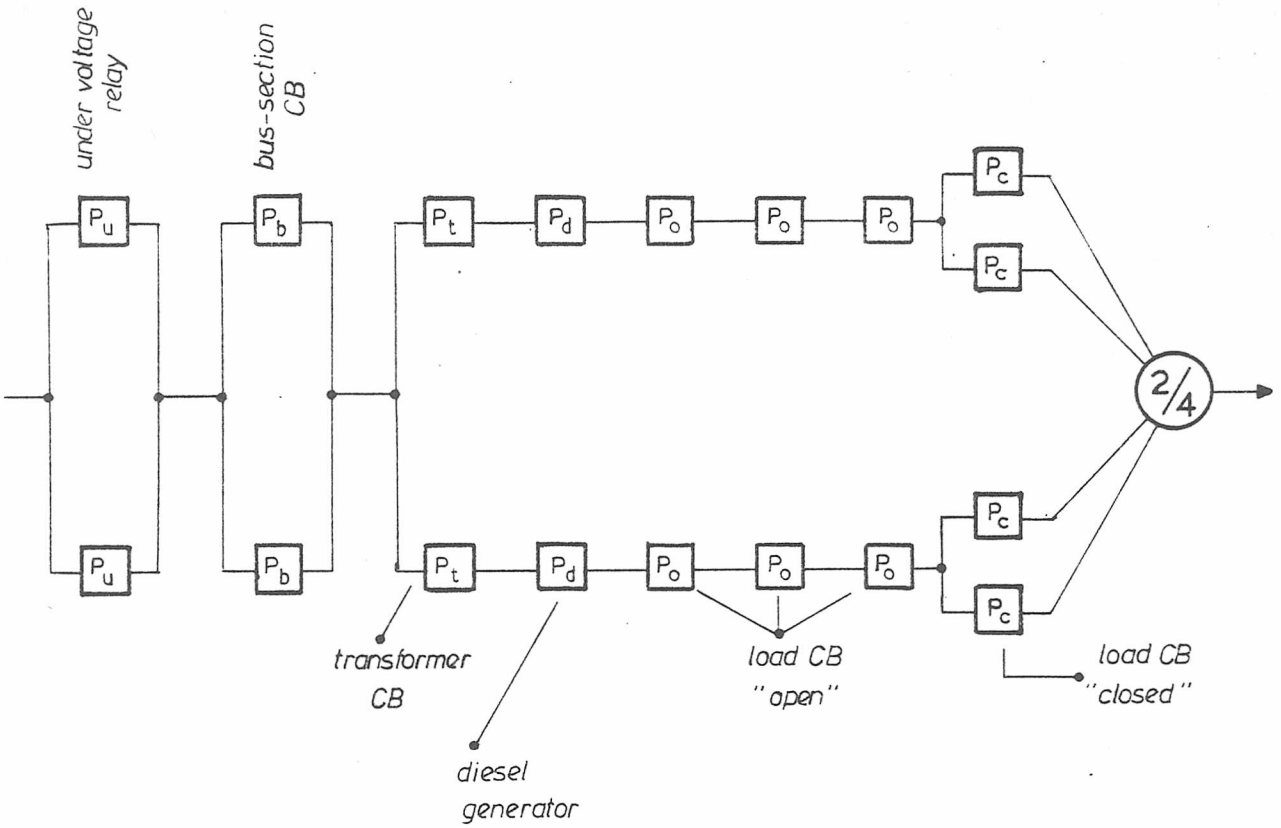


Figure 8·10b System flow diagram

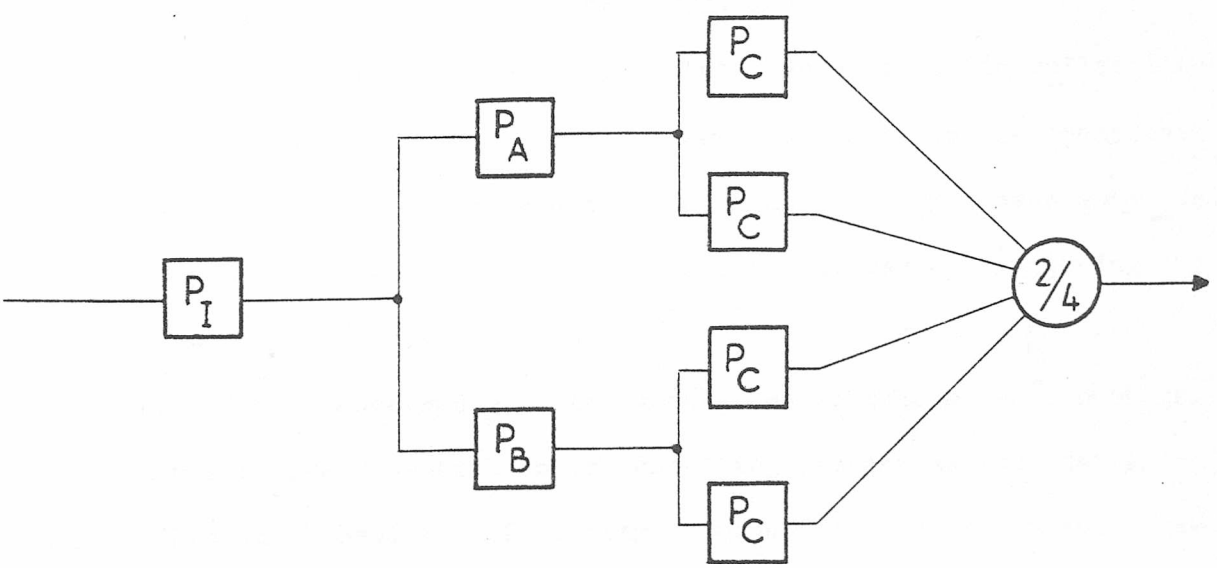


Figure 8·10c Flow diagram for reduced system

9.0 Overview of Project

The thesis begins with a discussion of the advantages and disadvantages of alternative reliability evaluation methods. The simulation approach is shown to be most useful where systems are complex and lead to analytical solutions which are intractable. For example, a system described by non-exponentially distributed random event times and limited repair facilities can only be investigated by simulation methods, unless the system is somehow 'simplified'. Complex systems generally require considerable system simplification before the system can be investigated by analytical means. This simplification, avoidable by simulation, can lead to results of questionable accuracy.

When modelling large complex systems, the simulation times experienced as a result of using conventional simulation methods are normally long. The Simulator developed avoids long simulation times by employing specially-constructed circuitry in place of the more conventional general purpose computer hardware.

Logical network techniques are frequently encountered in reliability studies, particularly due to their close resemblance to the functional system layout. Within the Simulator, the arrangement of components to form a system under study is conveniently achieved by specifying the system network.

Chapter 2 was concerned with the generation of random n-bit numbers. These numbers are essential to the modelling process as they determine the stochastic behaviour of system components. A technique was developed to generate multiple streams of statistically independent numbers. The necessary circuitry was limited by employing software techniques to initialise each number-generating circuit.

Statistical tests performed on software models of various number generating circuits indicated that the proposed technique yields highly random numbers. It is considered that the care taken during the development of the multiple random number sources has been reflected in the accuracy of the results obtained during later simulation experiments.

Chapter 3 described the use of the n-bit uniformly distributed random numbers, produced by the circuitry within each Component Module, to produce random event signals. These signals indicated the instant in time at which a change in the component state occurred i.e. from working to failed. A circuit that could generate event signals whose times between occurrence may follow any distribution was investigated. The effects of quantising both the probability and time values on the circuit operation were extensively researched, and a technique relying on error feedback was developed to gain greater modelling accuracy, whilst requiring less circuit memory storage.

Software modelling of the event signal circuitry was undertaken, and proved useful in determining the necessary binary word size by which component parameters are represented. A 12-bit representation was decided upon and this resulted in the 12-bit common bus for communication between the Simulator, host computer and the component modelling circuitry.

The modular format implementation of the Simulator was described in Chapter 4. Each of the key modelling elements required during simulation was explained and the role of the host computer made clear. Essentially the host enables the Simulator's operator to conveniently prepare the Simulator for a wide range of experiments, and after simulation is complete to extract the simulation results.

Although the detailed operation of the Simulator circuitry is complex, the Simulator's operator need only understand the system being studied in terms of its reliability features. That is, the operation of the circuitry is completely transparent to the operator.

The modular construction used throughout enables simple expansion of the Simulator to take place. Other advantages are that modules can be more easily tested during construction and the communication between modules is simplified by use of a common data/control bus.

A detailed description of the modules which make up the Simulator is given in Chapter 5. The operation of the Control Module in determining the time by which the simulation is to be incremented by, to reach the next minimum time quantisation value in use by any Component Module, is essential to the Simulator's operation, and is dealt with in detail. The use of a special bus to communicate each Component Module's time quantisation value to the Control Module simultaneously, ensures high simulation speeds.

The Network Specification Module and Repair Policy Modules are implemented by RAM decision tables. This allows complex networks and repair policies to be modelled at high speed, as the simulation time is partially determined by the access time of the table. A single examination of the respective tables produces all the necessary information for model updating to proceed.

The hardware structure of the Component Module is the limiting factor to the Simulator's flexibility. Considerable effort was directed towards obtaining an efficient design. The use of a re-programmable micro-instruction controller was very successful in achieving component flexibility. Re-programing of the control memory was avoided by employing a programmable mask register, which selects the component modelling features from a range of features contained in the memory. High speed component Module operation is achieved by parallel hardware

operation.

The reliability modelling aspects considered necessary by a 'general purpose' reliability simulator are presented in Chapter 6. The Component Module hardware was arranged to achieve just such a general purpose model, and the micro-control memories were programmed to carry out the control of the modules. From the description of the Simulator hardware operation required to sustain the general purpose model, the host computer software was developed. At this stage, the goal of making the Simulator's internal operations 'transparent' to the operator was achieved.

Use of an interactive graphics terminal greatly simplified the specification of system networks, and enabled the simulation results to be viewed in a graphical format immediately after the simulation was complete. A very convenient method of specifying a priority policy for component requests for repair men was also developed. This enabled changes in the policy or number of repair men to be easily made and the effects on system reliability to be rapidly determined.

To gain confidence in the Simulator's operation selected systems were studied in Chapter 7. Each system emphasised some particular aspect of reliability engineering, and mathematical analyses were performed on the system to confirm the simulation results. In all cases simulation results were very close to the theoretically predicted results, and the simulation times were short.

Two problems however were encountered. Firstly, the use of large time quantisation values lead to errors in the results, due to rounding error and cross-correlation effects. Secondly, the component scheduled block maintenance could vary, slightly, in time. These problems could be limited in their effect by careful use of the Simulator.

For each system examined, a single, or reduced number of component, representation was considered. The results obtained indicated that reduction can be successfully applied to many systems. By such methods the Simulator is able to tackle problems that have more components than the Simulator has Component Modules.

In Chapter 8 the Simulator is applied to a number of realistic system reliability problems, many of which would not be capable of solution by analytical means. The high speed of the Simulator is put to use in determining parameter sensitivity for a five component system. The actual system contains several interesting features and is employed for many of the simulation problems of Chapter 8.

Preventative maintenance schemes were investigated for system components, and optimal maintenance times were found where repair is considered good-as-new or minimal. Maintenance schemes where also investigated under the restriction of limited repair men. In such circumstances, priority request policies must be applied to obtain best performance from the repair men resource.

By extending the information gathered about system behaviour, the economics of system operation were investigated. Stratagies which minimise total system cost where found by assigning costs to:

1. component repair
2. component maintenance
3. repair men
4. loss of system

The results from the modelling of economic system operation proved very interesting and it is believed the Simulator is particularly useful when dealing with such problems.

After further development of the software at the host computer, non-time-homogeneous processes were able to be modelled. The Simulator was shown to be able to model components described by reliability growth or wearout. A two-component system was considered, the growth and wearout rates were varied, and the distribution of time to first system failure determined.

Finaly in Chapter 8 two industrial problems where considered viz. a standby power supply system and a reciprocating gas compressor. Each system contained features which are difficult to deal with analytically. The Simulator was shown to be a practical method of solving such reliability problems.

9.1 Hardware Versus Software Simulation

A hardware simulator is a system of electronic circuitry whose total system operation is affected by changing the interconnection between the circuit elements. The variations of interconnection must be performed by physicaly manipulating features of the circuitry.

Software simulators exist in the form of collections of code, viz programs. A program may be decoded to produce the instruction signals to direct a system of electronic hardware (computer) to perform the simulation task. The computer circuitry is different from the hardware simulator circuitry in that its operation is determined by coded instructions contained in a memory. Variation of the computer operation does not require any adjustment to the physical computer hardware, but only to the instructions contained in memory.

Within the definition given above the simulator developed during the research project may be defined as software, when viewed by the simulator operator. However, it must be remembered that the actual simulation is carried out by digital circuitry dedicated to reliability modelling. Although the operation of the circuitry is programmable, it

may only be varied in the context of reliability modelling. The title of the project arises from the investigation of the construction of the computer hardware.

The project has involved the construction of the simulator hardware and the development of a Fortran program, on another general purpose computer, which produces the code controlling the Simulator's circuitry. The general purpose computer is physically connected to the Simulator to allow the code to be entered into the Simulator's memory before simulation starts. The hardware of the Simulator is almost entirely in the form of special purpose circuitry, to achive considerable speed. The use of a network of general purpose microprocessors for the realisation of a hardware simulator was rejected because of the inherently slower operation that such a system would possess.

The arguments for and against software simulators intended for general purpose computers and hardware simulators, where the special hardware is controlled by a program contained in a general purpose computer associated with the hardware, favour the latter.

In terms of the financial cost to the experimenter, the amount spent on purchasing computer time for a large general purpose computer quickly exceeds the cost of constructing the special hardware. Including the cost of the small computer connected to the hardware still results in substantially lower simulation costs for all but the most trivial experiments.

Comparing the simulation speeds for complex reliability problems, the special purpose computer is significantly faster than even a high speed general purpose computer. The high speed is achieved by a multi-processor architecture and efficiency of programming code.

The simulator described in this thesis offers a very powerful tool for reliability investigations. The necessary hardware is inexpensive and forms a peripheral device to a general purpose (host) computer. The software controlling the operation of the special hardware is developed in a high level language and is run on the host computer.

Setting aside the logistics of supporting the special hardware, which has itself been effectively achieved, the varied simulation problems carried out during the research work indicate the flexible nature of the arrangement. Further, system models need not be compromised for simulation speed.

9.2 Possible Simulator Developments

The current simulator design offers a high degree of flexibility whilst maintaining considerable speed. The experiments reported in Chapters 7 and 8 confirm the flexible operation by their varied nature. Many of the systems studied could not have been analysed by analytical methods, endorsing the Simulator as a means of investigating complex systems. Throughout the reliability experiments performed, the design decisions made during the Simulator's construction have been questioned and the strengths and weaknesses of the Simulator determined. This section is intended to review the weakpoints in the design and suggest how they could be overcome in any future simulator.

The principle aim of the research undertaken was to investigate the design of a stochastic reliability simulator. The fundamental philosophy inherent in the design arrived at through the research work appears successful. In the most simplest terms the philosophy has been: Firstly the decomposition of the modelling task into separate modules which operate in parallel. Secondly the use of a host computer to achieve easy control and communication with modules. Most of the problems encountered with the Simulator have been in connection with

module implementation. Because of the special purpose hardware construction of the modules (so producing high speed operation) the elimination of certain problems would unfortunately require partial hardware re-design. Aspects to which attention should be given during any possible re-design are now dealt with separatlly below.

1. The component Modules are the most sophisticated hardware units within the Simulator. Each module contains seventyeight I.C. devices of which fiftyseven are soldered into printed circuit board locations (see Figure 9.2a). The remaining I.C. packages are located in wire-wrap holders in the top right hand corner of the board. Several Component Modules are required if the simulation of large systems is intended. This necessitates the repeated construction of component circuit boards which is a time consuming task. The elimination of the wire wrap area would considerably simplify the module construction and testing procedure.

2. During a simulation, errors in the expected results have been shown to occur where large time quantisation (TQ) values are used. The problem arises where components have long expected life times, and therefore require large TQ values to ensure:

   (a) good probability resolution of the hazard function (W_k) values.

   (b) the c.d.f. of component life time approaches unity before W_k memory storage runs out.

   When modelling life times which are exponentially distributed, a 12-bit W_k value is employed and (b) is not applicable. Investigations reported in Chapter 3 indicated that exponential distributions up to mean value $1/\lambda$=1500 hours could be modelled with TQ=1 hour. This is satisfactory for all but the most extreme cases. However when modelling non-exponential distributions, only 8-bit W_k

values are possible and memory storage is limited to 256 W_k values. Because of the feedback technique used to calculate non-exponential W_k values the selection of TQ values is dependent, to a great extent, on the memory size alone.

Consider a Weibull distribution with parameters λ,a (a=1). For the c.d.f. to approach 0.95 when the last memory location is reached then a memory containing $3x1/\lambda$ W_k values is required. This means that the maximum $1/\lambda$ value possible for a Weibull distribution (a=1) modelled on the Simulator is 85.3 hours, where TQ=1 hour. It can be seen that modelling of non-exponentially distributed time values frequently result in the use of large TQ values. Fortunately, where the distribution hazard rate is increasing, a>1, then a factor of less than $3x1/\lambda$ results.

A future simulator should contain more memory to store W_k values. Preferably as much memory as to achieve similarity between TQ values used by both exponential and non-exponential distributions. For exponential - distributions, TQ values are determined by the memory word size. If n is the word size measured in bits, and if a probability resolution of 0.5 is specified then the smallest W_k value representable is given by:

$$W_k = 2\ \frac{1}{2^n} = 1-e^{-k\lambda}$$

$$\Rightarrow \quad \frac{1}{\lambda} = 2^{n-1} \qquad \text{,where } \lambda << 1$$

This approximation can be used to determine the amount of memory required to 'match' memory size to memory word size. Figure 9.2b shows the memory size plotted against memory word size for the approximation. It can be seen that about 6K memory words will achieve similarity in TQ selection when considering a 12-bit W_k value.

3. When modelling with non-exponential distributions, the time quantisation values are selected by the host computer to ensure the distribution c.d.f. reaches 0.95 before distribution W_k value memory runs out. This does not always ensure selection of the best TQ value, as many components undergo replacement before their expected c.d.f. reaches 0.95. Host computer initialisation routines should take such effects into consideration.

4. The problem of maintaining block replacement times in phase for a group of components could be solved by removing the function of determining block replacement times from the components themselves. The Control Module should determine the block replacement times and inform the Component Modules. By this method, block replacement is centralised and the problems encountered with the present distributed scheme solved.

5. When modelling repair which results in a new component being better than repaired (NBR), repair distributions are limited to being of an exponential form unless use of the component interrupt facility is made. The only practical way of overcoming this problem is to employ a microprocessor in the Component Module design, to store age values during the repair process.

6. Programing the Simulator to run for a time t_s, results in the statistical gathering probes observing the number of occurrences of particular events, and also the total duration of all events. This enables the mean duration of an event to be determined but not the probability distribution of its duration. The probes were equipped with a 12-bit event counter, so that up to 4096 events could be observed before overflow of the event counter occurred. Probe counter overflow was a problem where very long simulation times

where used to ensure good mean value estimates. This suggests that the event counter should be increased in size or that the simulation should be stopped when the event counter limit is reached.

To enable distribution information to be gathered about event duration, some memory/counter circuitry should be incorporated in the Statistical Gathering Module. With as little as 4K memory words, a sufficient number of event durations could be observed to enable the distribution to be examined.

7. Systems which contain cold standby components suffer from the problem of a delay (TQ) before the standby component responds to a primary component failure. Design changes made to enforce immediate cold standby response, regardless of the time quantisation values in use.

An increase in simulator flexibility, as well as a solution to several of the problems listeed above, could be obtained by constructing Component Modules from microprocessor based circuitry. Such an implementation was previously rejected because of its slower operation. However the simpler circuit construction and increased flexibility may be regarded as more rewarding.

Considering a microprocessor based Component Module, the use of a micro-instruction memory would be retained to accomplish complex modelling decisions at high speed. The memory contents could be down loaded from the host computer prior to Simulation run, eliminating the need for a mask register, and offering greater flexibility for component operation. The program running in the microprocessor would itself be down loaded from the host, thus enabling the operation of the microprocessor to be varied whilst operating at any particular time with the minimum program size. Essentially the microprocessor program would be concerned with preparing the input signals to the micro-instruction

table and, following receipt of each model updating instruction, carrying out Component Module updating.

9.3 Conclusion

A general purpose programmable reliability simulator has been built largely according to designs originated during the three years of research. The Simulator, shown on Figure 9.3a, is constructed from special purpose digital circuitry to achive short simulation times. Stochastic signals, describing system characteristics, are manipulated by the circuitry in a parallel fashion.

The Simulator was shown to operate according to theoretical predictions. Systems which could not have been investigated by analytical techniques were studied. The Simulator was revealed to be a convenient and flexible method of studying complex systems. The special purpose circuitry employed enables several hundred years of system lifetime to be modelled in seconds.

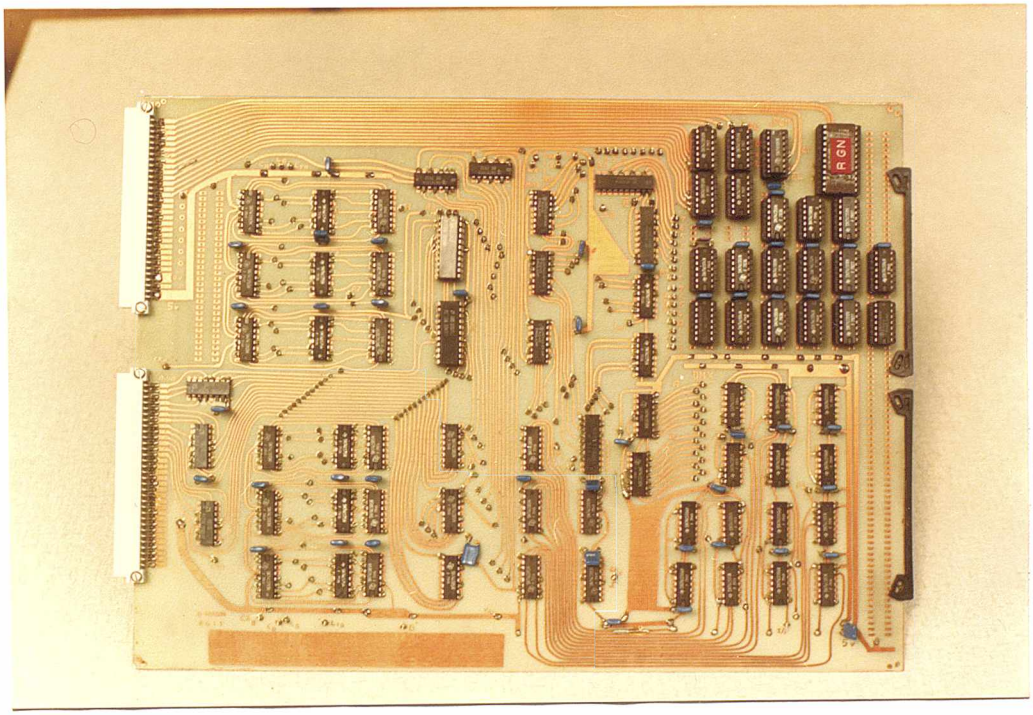


Figure 9·2a

Component Module Circuit Board

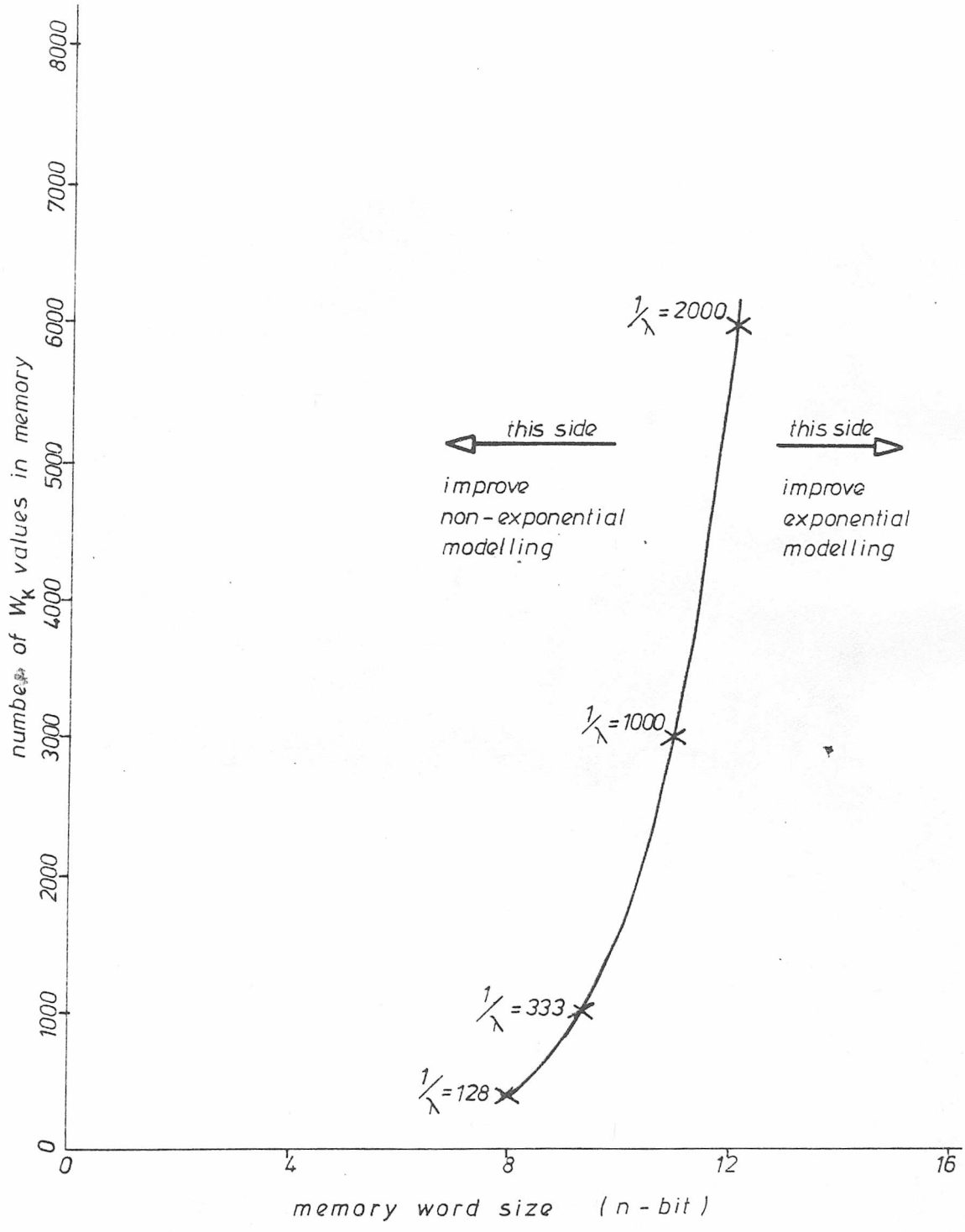


Figure 9.2b Selecting memory size

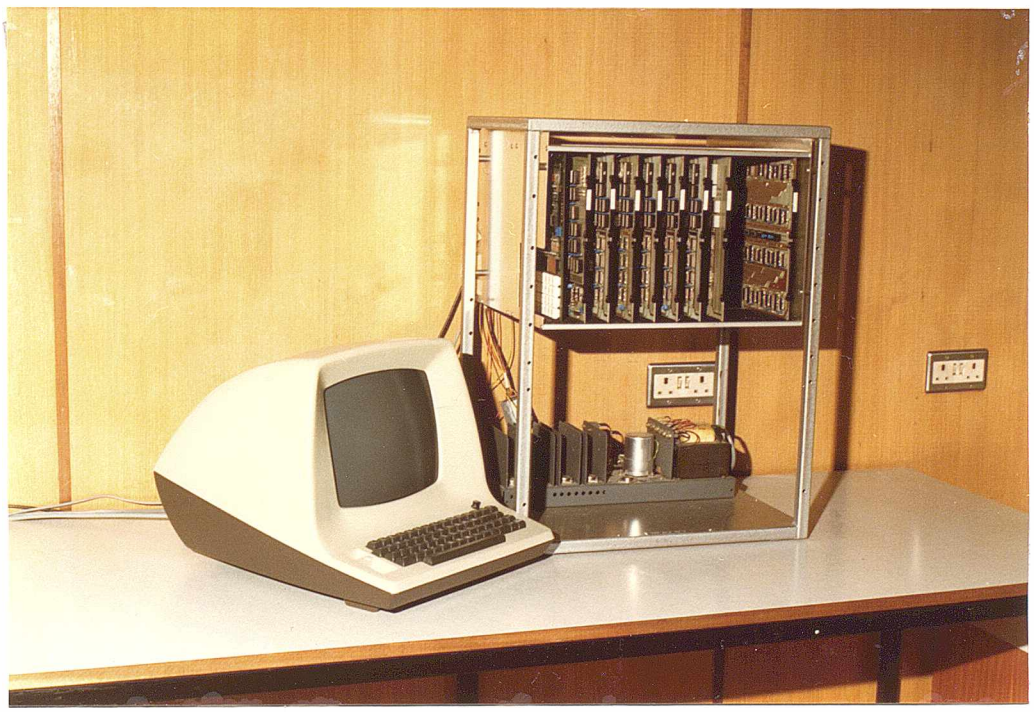


Figure 9·3a

Simulator and Operator Terminal

REFERENCES

1. Sinch C ,Billinton R ,'System Reliability Modelling and Evaluation', Hutchinson & Co. Ltd, 1977

2. Carradus R J ,'Analysis of Repairable Systems' The Marconi Review , Vol XLI No. 208 1978

3. Doyon L R ,'Solving Reliability Models of Nuclear Systems', Proceeding Annual Reliability and Maintainability Symposium 1977 pp 322-31

4. Tillman F A ,Lie C H ,Hwang C L ,'Simulation of Mission Effectiveness for Military Systems' IEEE Transactions Reliability Vol R-27 NO.3 pp 191-4

5. Singh C ,'Reliability Calculations of Large Systems', Proceedings of Annual Reliability and Maintainability Symposium 1975, pp188-193

6. Singh C ,Dhiion B S ,'On Fault Trees and Other Reliability Evaluation Methods' , Microelectronics and Reliability Vol 19, 1979 pp 57-63

7. Bazovsky I ,'Fault Trees, Block Diagrams and Markov Graphs ', Proceedings Annual Reliability and Maintainability Symposium 1977 pp 134-41

8. Basker B A ,Martin P ,'Availability Predictions by Using a Method of Simulation', Microelectronics and Reliability, 1977 Vol. 16, pp135-141

9. Alan A ,Pritsker B ,Kiviat P J ,'Simulation with GASP II',

10. Kumamoto H ,'State-Transition Monte Carlo for Evaluating Large, Repairable Systems', IEEE Trans. on Reliab. ,Vol R-29, Dec. 1980, pp376-380

11. Davidson A T G ,MacDonald I F ,'Evaluation of Reliability of Complex Systems' Advances in Reliability Technology , Symposium at Bradford 1976 , art. No. 19

12. Proctor C L ,Kothari A M ,'Fault Tree Analysis with Probability Evaluation ', Proc. Annual Reliability and Maintainability Symposium 1978 pp 306-11

13. Camarda P ,Corsi F ,Trentadue A ,' An Efficient Algorithm for Fault Tree Automatic Synthesis From the Reliability Graph ', IEEE Trans. Reliability Vol R-27 ,NO. 3 , 1978 pp 215-21

14. Brock P ,'The Reliability Analysis of Logical Networks by the Computer Program ALMONA', Nov. 1977 NCSR R14

15. Misra K B ,Raja A K ,'A Laboroty Model For System Reliability Analyzer ', Microelectronics and Reliability Vol 19 pp 259-64 1979

16. Laviron A ,Berard C ,Quenee R ,'ESCAF - failure Simulation and Reliability Calculation Device ' Second National Reliability Conference 1979 pp 6c/4/1 to 6c/4/10

17. Thakur R ,Misra K B ,'Monte Carlo Simulation for Reliability Evaluation of Complex Systems', Int. J. Systems Sci. ,1978 ,VOl 9 n0.11 pp1303-8

18. Rothbart G B ,Fullwood R R ,Bailey P C ,'Experiments with Stochastic Systems (ERMA)', Proceedings of the Annual Reliability and Maintainability Symposium, 1981 pp185-90

19. Golomb S W ,'Shift Register Sequences' , Holden-Day Inc. 1967

20. Hoffman de Visme G ,'Binary sequences' English University Press Ltd 1971

21. Maritsas D G ,'The Autocorrelation Function of Two Feedback Shift-Register Pseudorandom Source' IEEE Transactions on Computers 1973 pp 962-64

22. Tausworthe R C ,'Random Numbers Generated by Linear Recurrences Modulo Two', Mathematics of Computation 1965 Vol. 19 No. 90 pp 201-9

23. Birolini A ,'Hardware Simulation of Semi-Markov and Related Process' Mathematics and Computers in Simulation XIX(1977) 75-97 North-Holand Publishing Company

24. Maritsas D G ,Correspondence 'On the Statistical Properties of a Class of Linear Product Feedback Shift-Register Sequences', IEEE Transactions on Computers Oct.1973 pp961-2

25. Hartley M G ,'Development,Design and Test Procedure for Random generators using Chaincodes' Proc. IEE Vol 116 No. 1 1969 pp 22-34

26. Green D H ,'Nonlinear Product-Feedback Shift Registers ' Proc. IEE Vol. 177 No. 4 1970 pp 681-86

27. Schwind M ,'On Generating and Applicating a set of Independent Bernoulli-Sequences' Proceedings of 1st International Symposium on Stochastic Computing and its Applications, Toulouse, 1978 pp 103-12

28. Hurd W J ,'Efficient Generation of Statistically Good Pseudonoise by Linearly Interconnected Shift Registers' IEEE Transactions on Computers 1974 pp 146-52

29. Maritsas D G, Arvillias A C, Bounas A C ,'Phase Shift Analysis of Linear Feedback Shift Register Structures Generating Pseudorandom Sequences' IEEE Transactions on Computers c-27 No. 7 1978 pp 660-69

30. Mars P, Miller A J ,'Theory and Design of a Digital Stochastic Computer Random Number Generator' Mathematics and Computers in Simulation XIX 1977 pp 198-216 North-Holland Publishing Company

31. Latawiec K J ,Correspondence 'New methods of Generation of Shifted Linear Pseudorandom Binary Sequences'
Proc. IEE Vol.121 NO. 8 1974 pp 905-6

32. N D Deans, D P Mann, 'Design of High Speed Random Number Generator'
To be published in Mathematics and Computers in Simulation

33. Siegel S ,'Nonparametic Statistics for Behavioural Science'
New York , McGraw Hill Ltd. 1956

34. Maisel H ,Gnugnoli G ,' Simulation of Discrete Stochastic Systems', Kingsport Press Ltd. , 1972

35. Barlow R E ,Proschan F ,'Mathematical Theory of Reliability
Wiley 1965

36. Allan R N ,Antonopoulos C G ,'Modelling Non-Exponential Distributions in Systems Reliability Evaluation',
5th Symposium on Reliability Technology,
Bradford 1978

37. Mihram G A ,'Simulation Statistical Foundations and Methodology' Academic Press, 1972

38. Daellenbach H G ,George J A ,'Introduction to Operations Research Techniques' Allyn and Bacon Inc., 1978

39. Davies N R ,'Decision Tables in Discrete-System Simulation'
Simulation Journal 1974 pp 39-44

40. D P Mann, N D Deans, 'Programing a Logical Network Decision Table',
Submited to Computers and Digital Techniques

41. Green A J ,Bourne A E ,'Reliability Technology',
Wiley, 1972

42. Edwards G T ,Watson I A ,'A Study of Common-Mode Failures',
Safety and Reliability Directorate, SRD 146, 1779

43. Mueller K F ,Schuessler H L ,Costner H L ,'Statistical Reasoning in Sociology', Houghton Mifflin Company 1977

44. Basker B A ,'Sensitivity Analysis of Availability of unit in a Production/electrical System', Microelectronics and Reliability, Vol. 16 1977, pp259-71

45. Ohashi M ,et al ,'Optimum Preventive Maintenance Policy for Two-Unit Priority Standby Redundant with Minimal Repair',Microelectronics and Reliability,
Vol. 18 1979, pp535-38

46. Berg M ,Epstein B ,'Comparison of Age, Block, and Failure Replacement Policies', IEEE Transactions on Reliability,
Vol. R-27, 1978, pp25-29

47. Murthy D N P ,Nguyen D G ,'Optimal Age-Policy with Imperfect Preventive Maintenance',
IEEE Trans. Rel. ,Vol. R-30, No. 1 1981, pp80-81

48. Nakagawa T ,'Replacement Models with Inspection and Preventive Maintenance', Microelectronics and Reliability, 1980 Vol. 20, pp427-433

49. Ascher H ,Feingold H ,'Is There Repair After Failure', Annual Reliability and Maintainability Symposium, IEEE Proceedings 1978, pp 190-197

50. Schick J G ,Wolverton R W ,'An Analysis of Competing Software Reliability Models', IEEE Trans. on Software Eng. ,Vol. SE-4 1978 pp104-20

51. Soi I M ,Gopal K ,'A Comparative Study of Software Reliability Models', IE(I) Journal-ET, Vol. 59 1978, pp1-6

52. Moranda P B ,Software Reliability Research', Statistical Computer Performance Evaluation, Academic Press, 1972, pp 465-484

53. Musa J D ,'Validity of Execution-Time Theory of Software Reliability', IEEE Trans. on Reliab. Vol. R-28, No. 3, 1979, pp181-91

54. Moranda P B ,'Event-Altered Rate Models for General Reliability Analysis', IEEE Trans. on Reliab. , Vol. R-28, No. 5, 1979 pp376 381

55. British Standards Institution, BS5760:Part 2, 1981 Reliability of Systems, Equipments and Components

56. Janke E, Emde F ,'Tables of Functions with Formulae and Curves', Dover Publications, New York,

57. Bourne A E ,National Centre of Systems Reliability, Safety and Reliability Directorate, Culcheth

BIBLIOGRAPHY

Green A J ,Bourne A E ,'Reliability Technology',
Wiley, 1972

Maisel H ,Gnugnoli G ,' Simulation of Discrete Stochastic
Systems', Kingsport Press Ltd. , 1972

Barlow R E ,Proschan F ,'Mathematical Theory of Reliability
Wiley 1965

Siegel S ,'Nonparametic Statistics for Behavioural Science'
New York , McGraw Hill Ltd. 1956

Golomb S W ,'Shift Register Sequences' , Holden-Day Inc. 1967

Hoffman de Visme G ,'Binary sequences'
English University Press Ltd 1971

Sinch C ,Billinton R ,'System Reliability Modelling and
Evaluation', Hutchinson & Co. Ltd, 1977

Cox D R ,'Renewal Theory',
Butler and Tanner Ltd., 1962

Tocker K D ,'The Art of Simulation', English Universities
Press, 1963

Hartley M G, 'Digital Simulation Methods',
Peter Peregrinus Ltd. for the IEE, 1975

Davidson A T G ,MacDonald I F ,'Evaluation of Reliability of
Complex Systems'
Advances in Reliability Technology , Symposium at Bradford
1976 , art. No. 19

Walton D ,'P.c.b. Layout for High-speed Schottky t.t.l.',
Wireless World, Feb 1978

Appendix A1

Tests on non-uniform distributed random number generator

The probability distribution observed, for each random variable generated, has been plotted as a continuous distribution. The method used enables simple comparisons and judgments to be made on the generated distributions.

A1.1.1 Empirical Frequency Test

An empirical frequency test the χ^2 test, described in Section 2.6.3, is used to compare the generated p.d.f. against the programmed p.d.f $f_k(k\Delta x)$. The discrete time intervals (gap) between renewal points were recorded and the empirical frequency of each gap calculated. The theoretical absolute frequency for each gap was also determined and the χ^2 tests used as a measure of goodness of fit between the generated distribution and expected. Note that gaps have been grouped together where the theoretical absolute frequency has fallen below 5. The χ^2 values obtained for each distribution have been given on each p.d.f. graph. The tables at the end of each distribution section contain critical values for the χ^2 tests.

A1.1.2 Cumulative Distribution Test

The Kolmogorov-Smirnov test, described in Section 2.6.6 has been used to compare the experimental c.d.f. observed with the expected c.d.f. . The test is based on the value D which is the difference with greatest absolute magnitude between the two distributions. The value D obtained for each distribution has been given on the p.d.f. diagram, and on the table of results at the end of each distribution section. Critical values for the test are also given in the table of results.

Results for exponential and Weibull distributions also contain expected values of D. These values are based on the work of Sections 3.8 and 3.9. They are the D values predicted for the distribution $F_{kex}(k\Delta x)$ considering a 'perfect' random number generator is used.

A1.2 Exponential Distribution

The exponential distribution [1,34,35] is used widely to represent component operating times. Data collected on the life time of equipment has led to this assumption. The family of exponential distributions is now the best known and most thoroughly explored. Figure A1.2a shows the c.d.f. F(x), p.d.f. f(x) and hazard rate $\emptyset(x)$ for an exponential distribution. It has the property that if failure of a component has not occurred by time t then the probability distribution for its future life is the same as if the component were new. Only the exponential distribution has this property and it is explained by the monotonic hazard rate.

The exponential distribution has been applied to the repair times of components, though there is evidence that this is not an accurate description [36]. However, modelling both repair times and operating times with exponential distributions, leads to a Markovian process which enables an analytical solution to the reliability problem. The exponential distribution can be regarded as a continuous analogue of the discrete geometric distribution.

Calculation of W(k) values (simplified method):

Density function, $f(x)= \lambda e^{-\lambda x}$ $\quad ,x\geq 0$

mean interval time $= \frac{1}{\lambda}$

c.d.f. $$F(x) = \int_0^x \lambda e^{-\lambda t}\, dt$$

$$= 1 - e^{-\lambda x}$$

To form a discrete approximation to F(x), $F(k\Delta x)$

$$F(k\Delta x) = 1 - e^{-\lambda \Delta x k}$$

The W(k) values can now be calculated:

$$W(k) = \frac{F(k\Delta x)-F((k-1)\Delta x)}{1-F((k-1)\Delta x)}$$

$$= \frac{1-e^{-\Delta x k} - (1+e^{-\lambda \Delta x (k-1)})}{1-(1+e^{-\Delta x (k-1)})} \cdot H$$

$$W(k) = H\,(\,1-e^{-\Delta x \lambda}\,)$$

It can be seen that this method leads to a constant value of W(k), as would be expected from a monotonic failure rate. Generation of this distribution is greatly simplified as the memory ability to store W(k) values is not required.

Statistical tests have been carried out on an exponential distribution of mean=$1/\lambda$ =10. The p.d.f. generated for various methods are shown on Figures A1.2b to A1.2j , and a table of all results is given on A1.2k.

A1.3 Weibull Distribution

The Weibull distribution [1pg30, 36] is a continuous distribution which is especially applicable whenever the process at hand is yielding a random variable that is essentially the maximum (or minimum) value among a large set of random variables. For example, if a system, such as an electronic instrument, is deemed operational until such time as any one of its components has failed, then the time S between repairs will be a random variable of the Weibull family, since S is essentially

defined as the minimum of its components lifetimes.

Figure A1.3a shows the c.d.f. F(x), p.d.f. f(x) and hazard rate $\phi(x)$ for a Weibull distribution over a range of parameter (a) values. Weibull distributions with increasing hazard rate (a>1) have been found useful in simulating equipment lifetimes [14].

The following describes the calculation of W(k) values necessary to generate a Weibull distribution.

c.d.f. $\quad F(x) = 1 - e^{-(\lambda x)^a}$

discrete approximation $\quad F(k\Delta x) = 1 - e^{-(\lambda \Delta x\, k)^a}$

density function, $\quad f(x) = a\lambda^a x^{a-1} e^{-(\lambda x)^a} \quad , \quad x \geq 0$

calculation of W(k) value:

$$W(k) = \frac{F(k\Delta x)-F((k-1)\Delta x)}{1-F((k-1)\Delta x)} \cdot H$$

$$= \frac{1-e^{-(\lambda \Delta x)^a} -1+e^{-(\lambda \Delta x(k-1))^a}}{1-1+e^{-(\lambda \Delta x(k-1))^a}} \cdot H$$

$$W(k) = H.\left(1-e^{-(k^a-(k-1)^a)(\lambda \Delta x)^k} \right)$$

Statistical tests have been carried out on a Weibull distribution of parameter λ =0.04, a=3.5. The p.d.f. generated for various methods employed are shown on Figures A1.3b to A1.3j ,and a table of all results is given on Figure A1.3k.

A1.4 Erlang Distribution

It is possible to 'simulate' a number of arbitrary distributions by a compound system of negative exponential distributions. The resulting distribution is known as an Erlang distribution [36,37]. By experimenting with the number r, it is possible to match any observed distribution. Figure A1.4a shows the c.d.f., p.d.f. and hazard rate

for a range of parameter r values. For large values of r, the Erlang distribution approaches the normal distribution.

$$\text{density function } f(x) = \frac{\lambda^r x^{r-1} e^{-\lambda x}}{(r-1)!}$$

consider the case when $r = 4$

$$f_4(x) = \frac{\lambda^4}{3!} x^3 e^{-\lambda x} \qquad ,x>0$$

the corresponding c.d.f. is given by :

$$F_4(x) = \int_0^x f_4(t)\, dt$$

$$= \frac{1}{3!}\left[\frac{\lambda^4 t^3 e^{-\lambda t}}{\lambda} - \frac{\lambda^4 3t^2 e^{-\lambda t}}{\lambda^2} + \frac{\lambda^4 6 t e^{-\lambda t}}{\lambda^3} - \frac{\lambda^4 6 e^{-\lambda t}}{\lambda^4} \right]_0^x$$

$$= \frac{1}{3!}[6 - e^{-\lambda x}(\lambda^3 x^3 + \lambda^2 3x^2 + \lambda 6x + 6)]$$

In general:

$$F_r(x) = \frac{1}{(r-1)!}[(r-1)! - e^{-\lambda x}((\lambda x)^{r-1} + (\lambda x)^{r-2}(r-1) + \ldots + (r-1)!)]$$

$$F_r(x) = 1 - e^{-\lambda x} \sum_{n=0}^{r-1} \frac{(\lambda x)^n}{n!}$$

discrete approximation:

$$F(k\Delta x) = 1 - e^{-\lambda \Delta x k} \sum_{n=0}^{r-1} \frac{(\lambda \Delta x k)}{n!}$$

The W(k) values can be calculated:

$$W(k) = \frac{F(k\Delta x) - F((k-1)\Delta x)}{1 - F((k-1)\Delta x)}$$

$$= \frac{\sum_{n=0}^{r-1} \frac{(\lambda \Delta x)}{n!}[(k-1)^n - e^{\lambda \Delta x} k^n]}{\sum_{n=0}^{r-1} \frac{\lambda \Delta x (k-1)^n}{n!}}$$

Statistical tests have been carried out on an Erlang distribution of parameters $\lambda=0.3$, $r=5$. The p.d.f. generated for various generation methods are shown on Figures A1.4b to A1.4m, and a table of all results is given on Figure A1.2n.

A1.5 Poisson Distribution

The Poisson distribution [36] is another discrete distribution of importance in system simulation. It gives the probability of exactly x independent occurrences during a given period of time if events take place independently and at constant rate.

For example the simplest waiting line models assume that the number of arrivals occuring within a given interval of time, t, follows a Poisson distribution with parameter λt, where λt is the average number of arrivals in the interval of time t. Note λ represents the arrival rate, and x the number of arrivals in the interval t.

density function $f(x) = \dfrac{(\lambda t)^x e^{-\lambda t}}{x!}$,x=0,1,2...

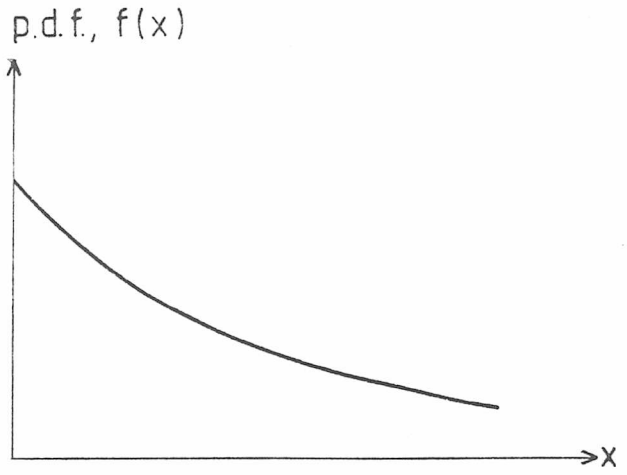
c.d.f $F(x) = \sum_{k=0}^{x} \dfrac{(\lambda t)^k e^{-\lambda t}}{x!}$

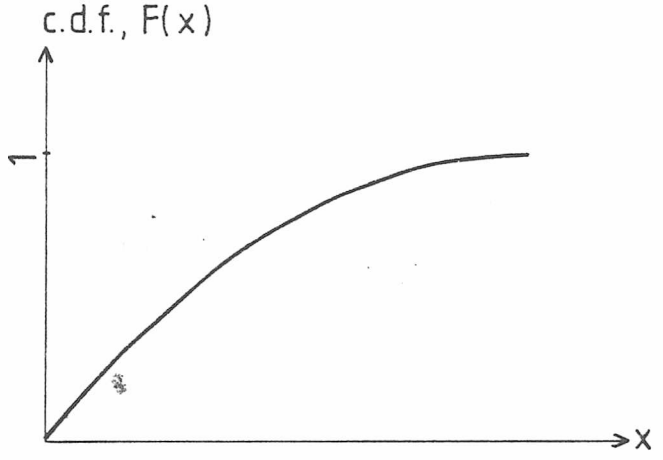
The W(k) values can be computed from :

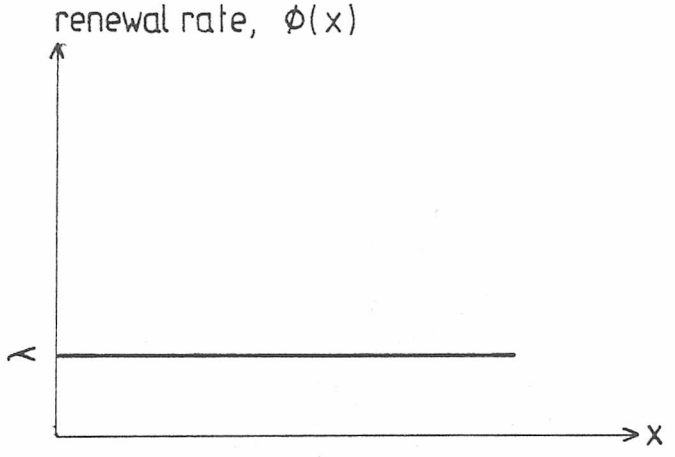
$$W(k) = \frac{F(k\Delta x)-F((k-1)\Delta x)}{1-F((k-1)\Delta x)}$$

$$= \frac{(t\lambda)^x / x!}{e^{-t} - \sum_{k=o}^{x-1} \frac{(\lambda t)^k}{k!}} \quad ,x=k\Delta x$$

Statistical tests have been carried out on a Poisson distribution of parameter $\lambda t=25$. The p.d.f. generated for each generation method employed are shown on Figures A1.5b to A1.5d, and a table of all results is given on Figure A1.5e.







exponential distribution $F(x) = 1 - e^{-\lambda x}$

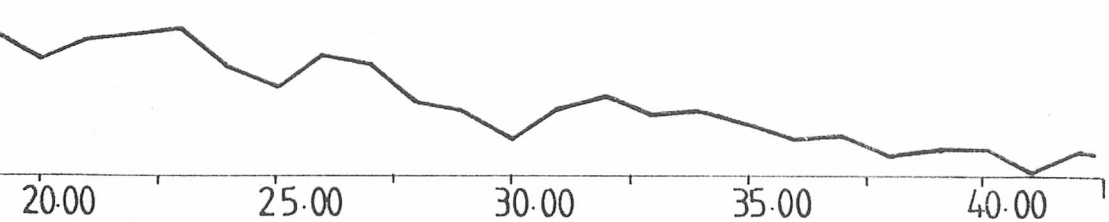
Figure A1·2a

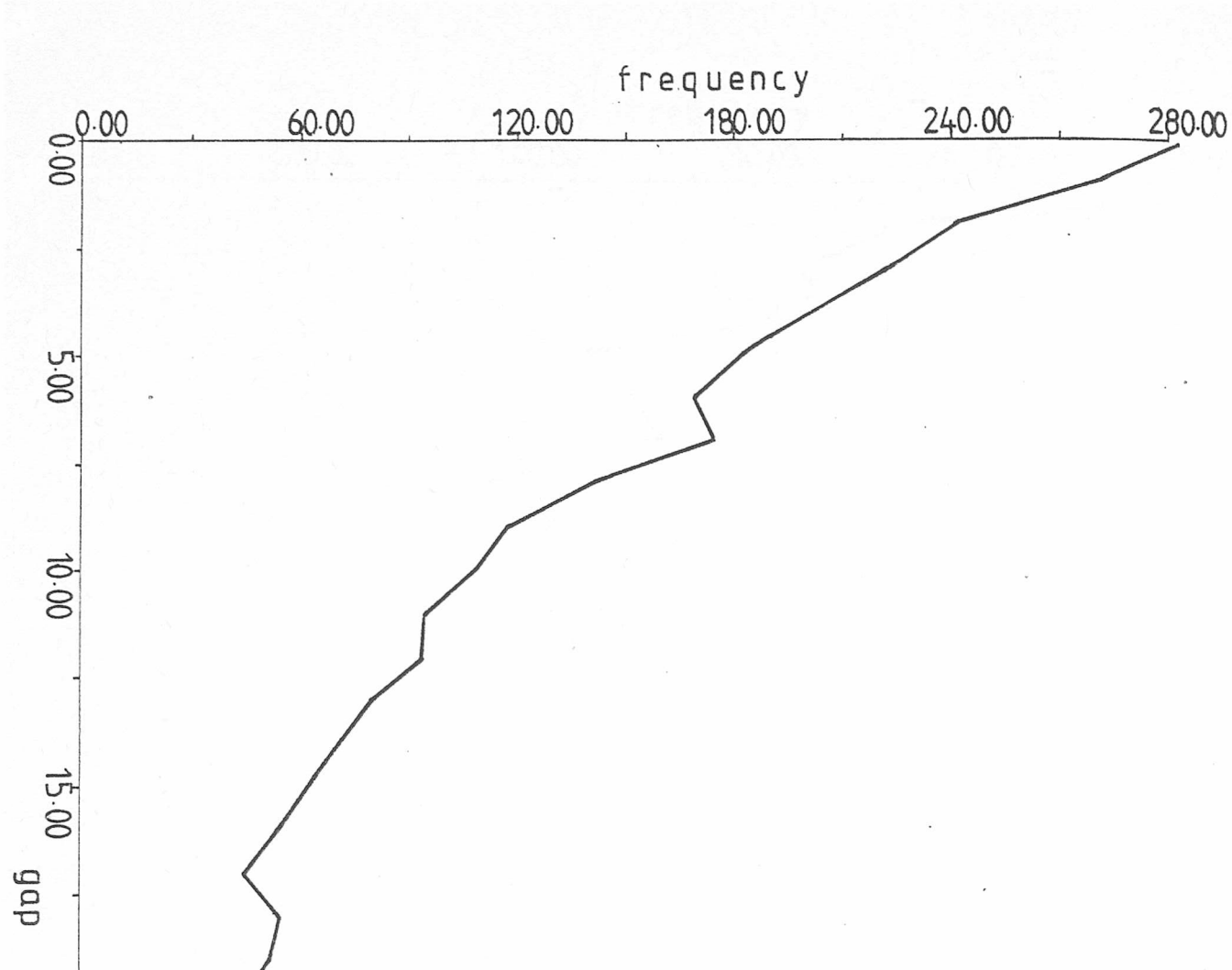
300.00
240.00
180.00
120.00
60.00
0.00
frequency
0.00
5.00
10.00
15.00
gap

Graph. No.	A1·2b
Distribution:	Exponential

(data)

sample size	=	*3200*
method	=	*SIMPLE*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*61·3*
D	=	*0·0875*

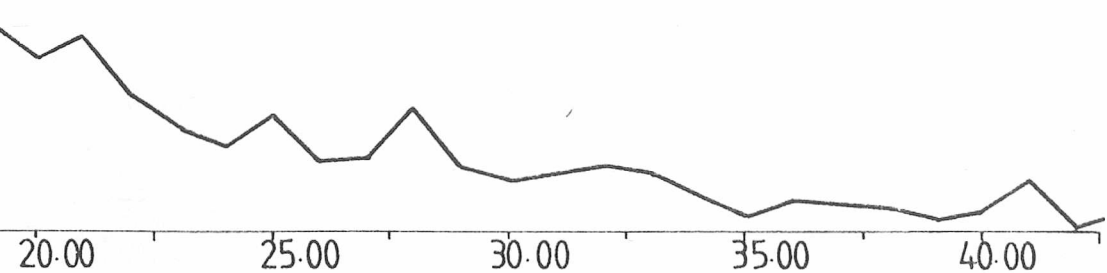


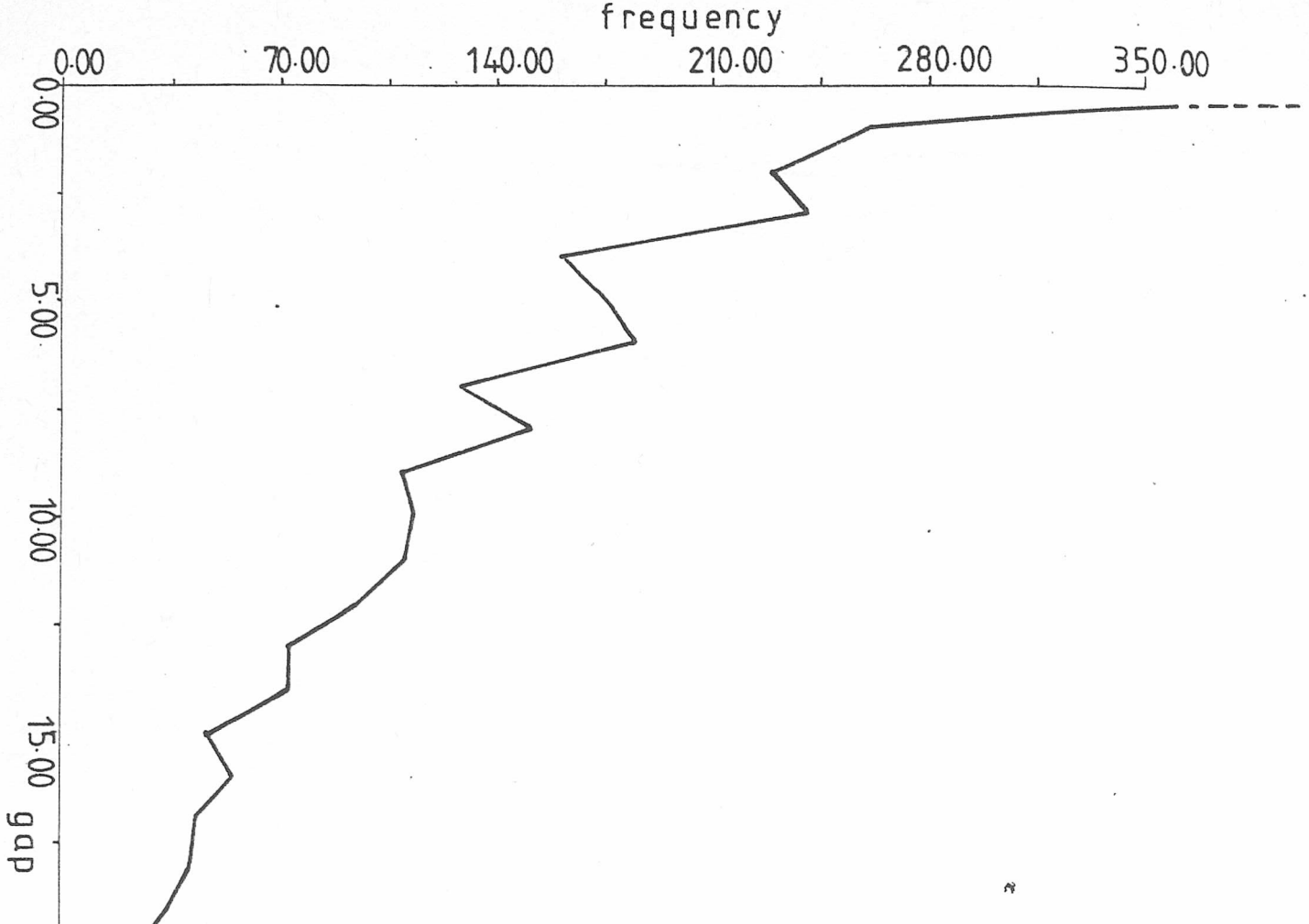

frequency
0.00
60.00
120.00
180.00
240.00
280.00
0.00
5.00
10.00
15.00
gap

Graph No.	A1·2 c
Distribution:	Exponential

(data)

sample size	=	*3200*
method	=	*SIMPLE*
ΔX	=	*1·0*
H	=	2^{16}
χ^2	=	*35·2*
D	=	*0·0850*

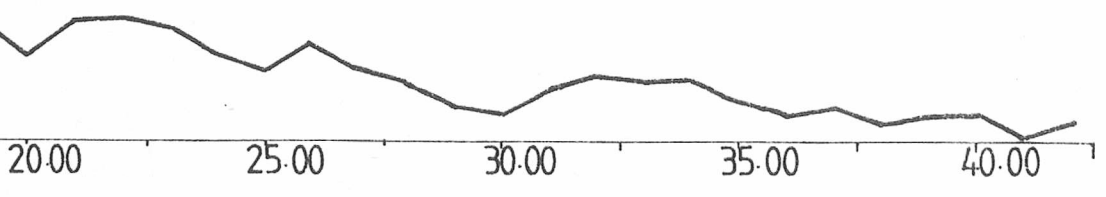



frequency
0.00
70.00
140.00
210.00
280.00
350.00
0.00
5.00
10.00
15.00
gap

Graph No.	A1·2d
Distribution:	Exponential

(data)

sample size	=	*3200*
method	=	*TS*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*135·8*
D	=	*0·0422*

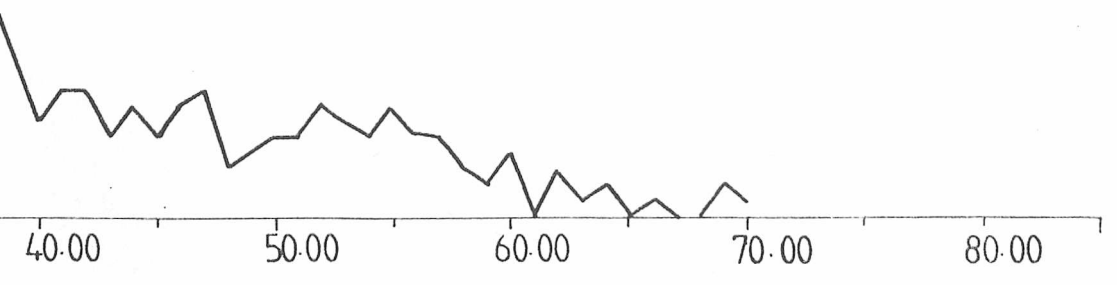


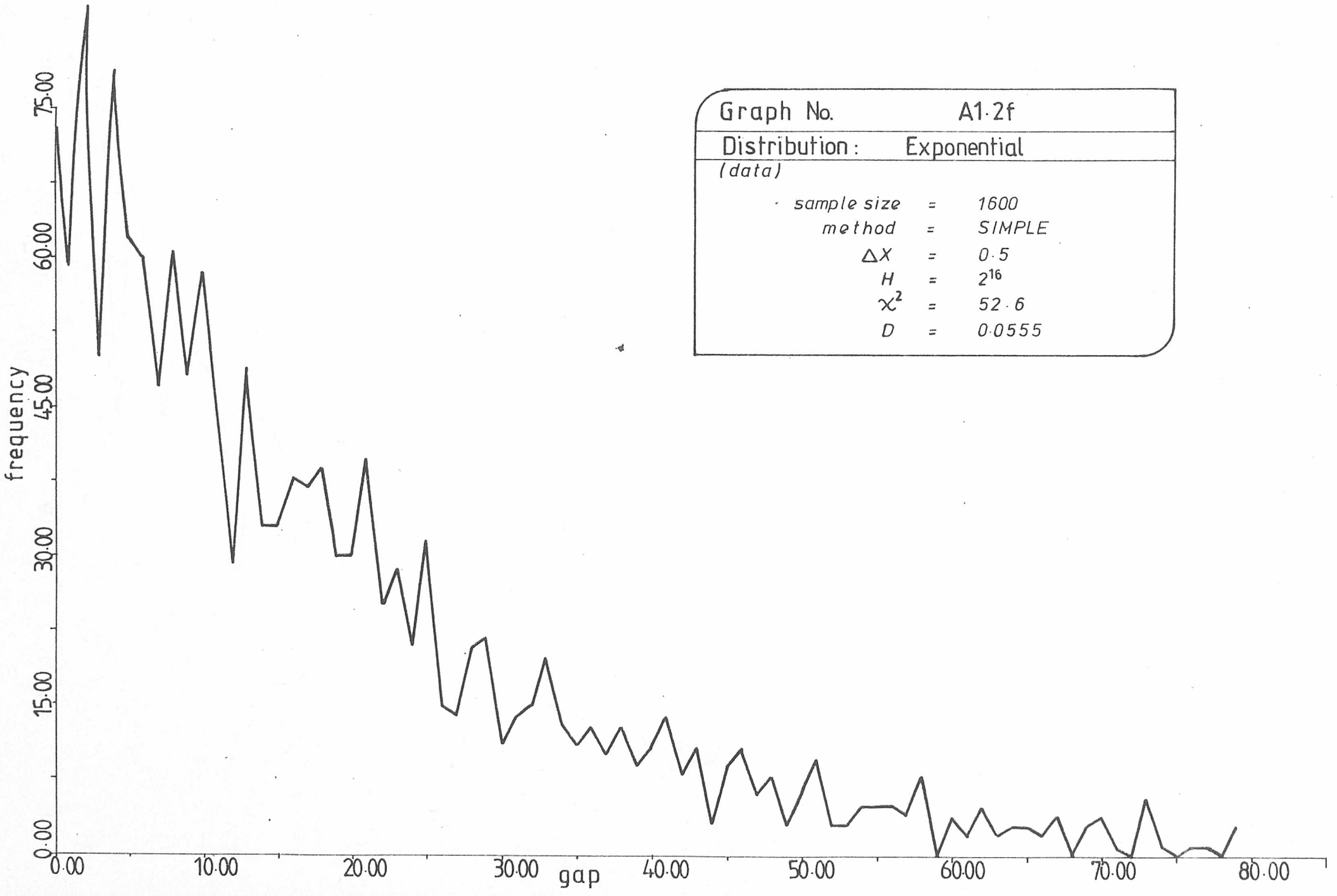
frequency
75.00
60.00
45.00
30.00
15.00
0.00
0.00
10.00
20.00
30.00
gap

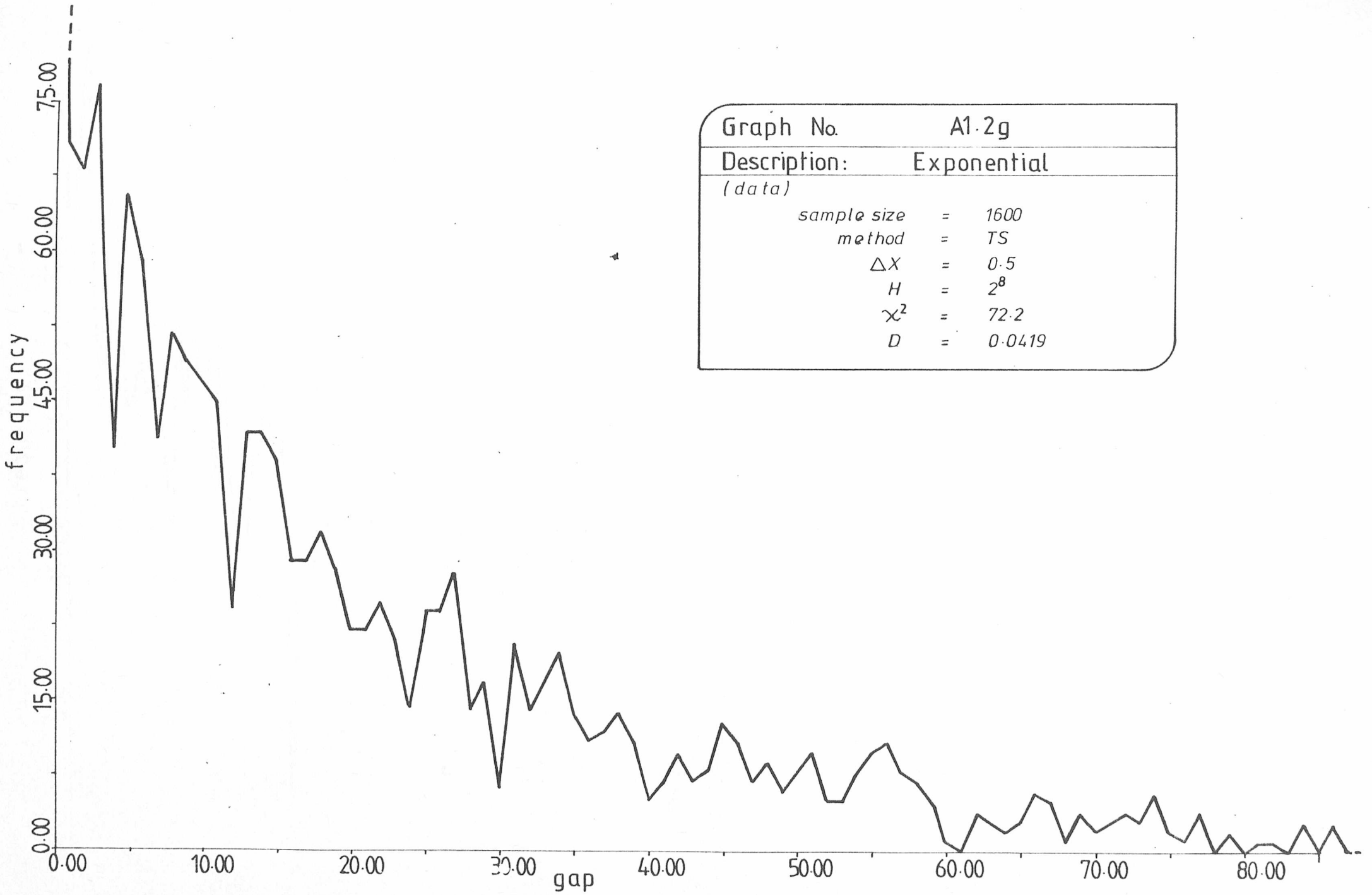
Graph No.	A1·2 e
Description:	Exponential

(data)

sample size	=	*1600*
method	=	*SIMPLE*
ΔX	=	*0·5*
H	=	2^8
χ^2	=	*53·9*
D	=	*0·0456*



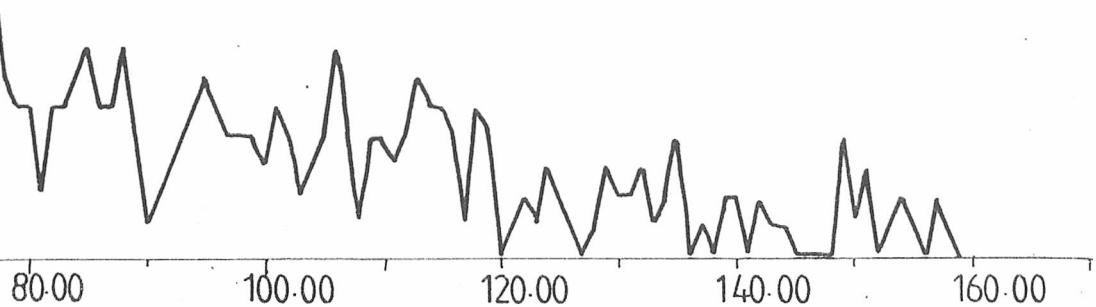

Graph No. A1·2f
Distribution: Exponential
(data)
sample size = 1600
method = SIMPLE
ΔX = 0·5
H = 2^16
χ² = 52·6
D = 0·0555
frequency
0·00
15·00
30·00
45·00
60·00
75·00
0·00
10·00
20·00
30·00
gap
40·00
50·00
60·00
70·00
80·00

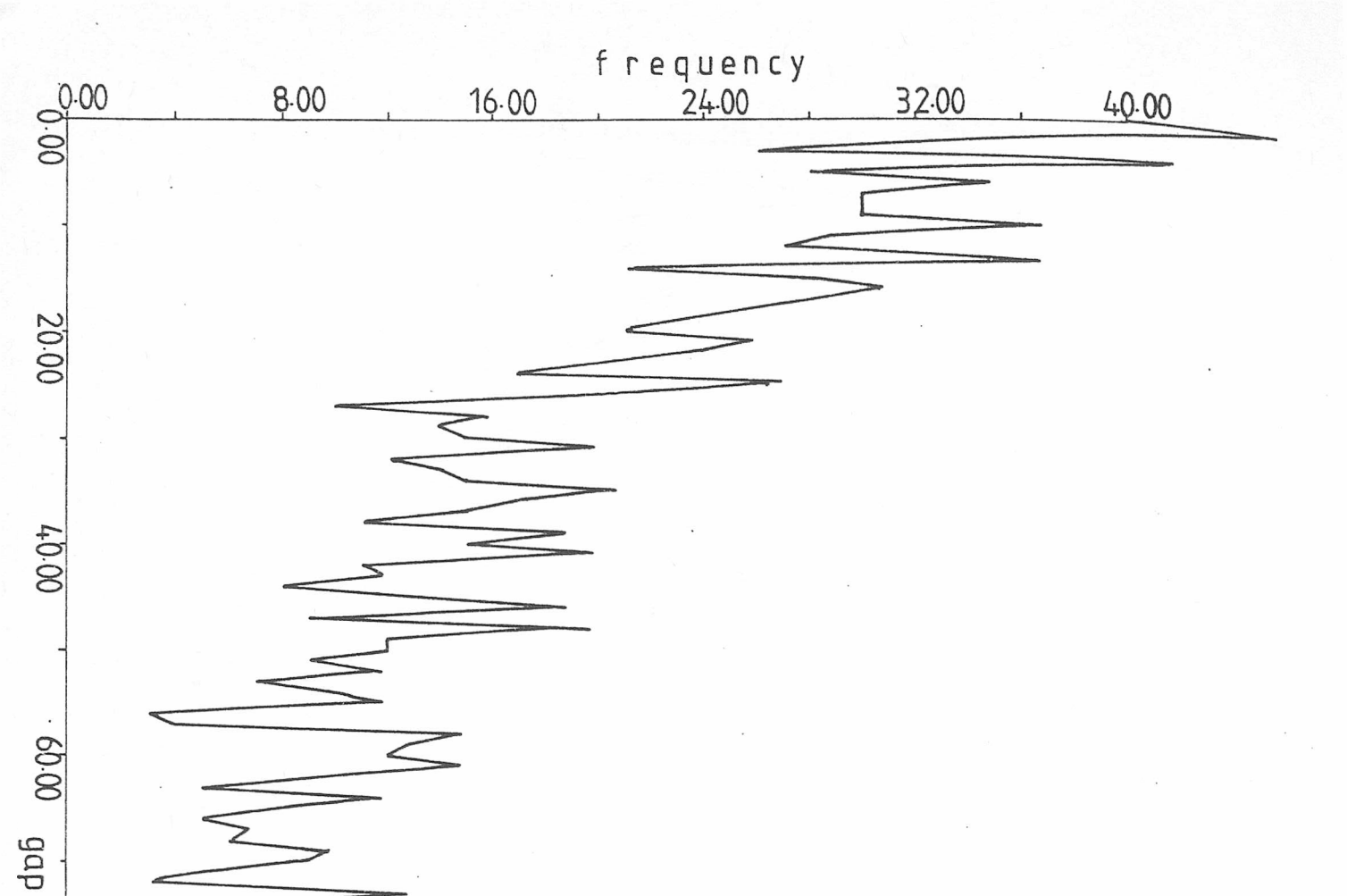

Graph No. A1·2g
Description: Exponential
(data)
sample size = 1600
method = TS
ΔX = 0·5
H = 2^8
χ² = 72·2
D = 0·0419
frequency
0·00
15·00
30·00
45·00
60·00
75·00
0·00
10·00
20·00
30·00
40·00
50·00
60·00
70·00
80·00
gap

frequency
0.00
8.00
16.00
24.00
32.00
40.00
0.00
20.00
40.00
60.00
gap

Graph No.	A1·2h
Description:	Exponential

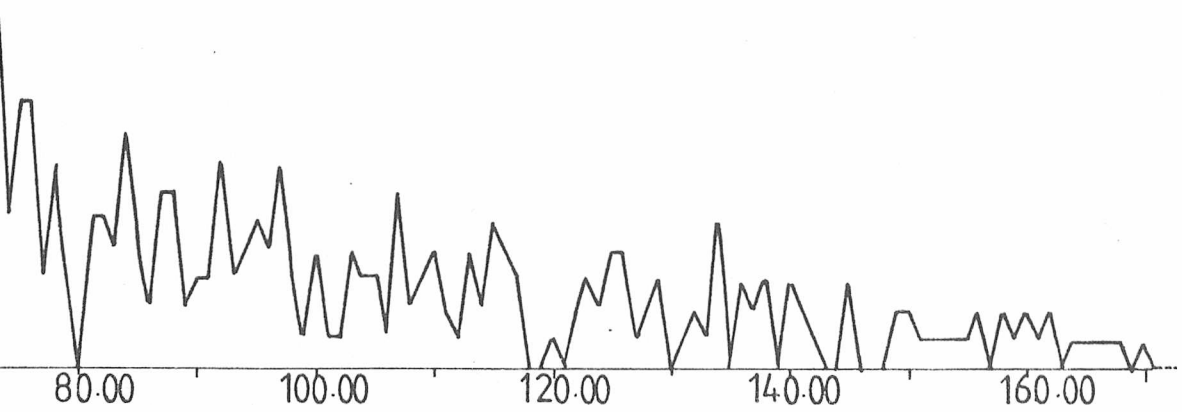
sample size	=	*1600*
method	=	*SIMPLE*
ΔX	=	*0·25*
H	=	2^8
χ^2	=	*48·0*
D	=	*0·469*




frequency
0.00
8.00
16.00
24.00
32.00
40.00
0.00
20.00
40.00
60.00
gap

Graph No.	A1·2i
Description:	Exponential

sample size	=	*1600*
method	=	*SIMPLE*
ΔX	=	*0·25*
H	=	2^{16}
χ^2	=	*34·2*
D	=	*0·0316*

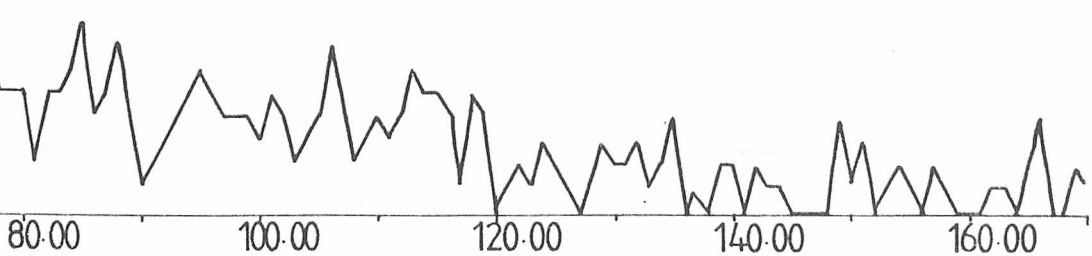


frequency
50.00
40.00
30.00
20.00
10.00
0.00
0.00
20.00
40.00
60.00
gap

Graph No.	A1·2j
Distribution:	Exponential

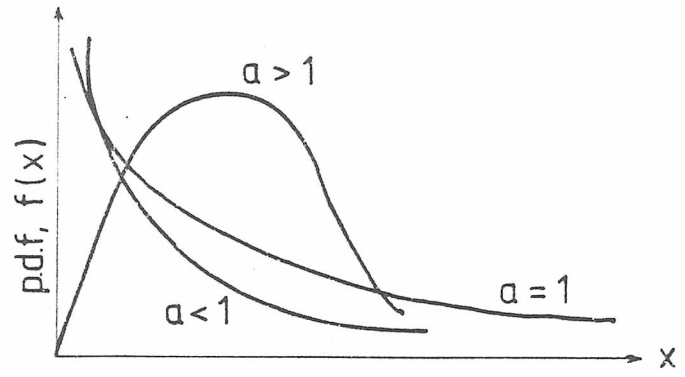
(data)

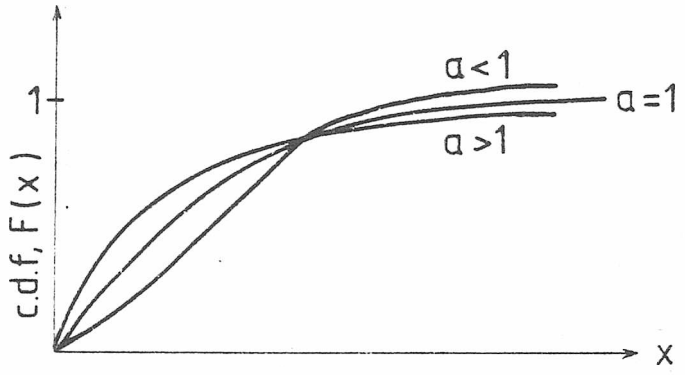
sample size	=	*1600*
method	=	*TS*
ΔX	=	*0·25*
H	=	2^8
χ^2	=	*61·2*
D	=	*0·0456*

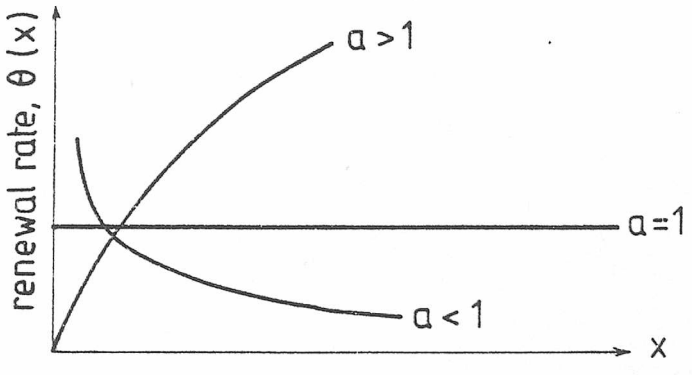


EXPONENTIAL DISTRIBUTION		number of samples	number of bits of random number	Δx	K-S TEST			$\chi^2_{n,\alpha}$ TEST	
method	figure				test result	value expected	critical value	result	critical value
simple	4.2b	3200	8	1.0	0.0875	0.181	0.034	61.3	56.9
simple	4.2c	3200	16	1.0	0.0850	0.181	0.034	35.2	56.9
TS	4.2d	3200	8	1.0	0.0422	——	0.034	135.8	56.9
simple	4.2e	1600	8	0.5	0.0456	0.095	0.048	53.9	56.9
simple	4.2f	1600	16	0.5	0.0555	0.095	0.048	52.6	56.9
TS	4.2g	1600	8	0.5	0.0419	——	0.048	72.2	56.9
simple	4.2h	1600	8	0.25	0.0469	0.0488	0.048	48.0	56.9
simple	4.2i	1600	16	0.25	0.0316	0.0488	0.048	34.2	56.9
TS	4.2j	1600	8	0.25	0.0456		0.048	61.2	56.9

Table A1.2k Table of results and critical values ($\alpha = 0.95$)



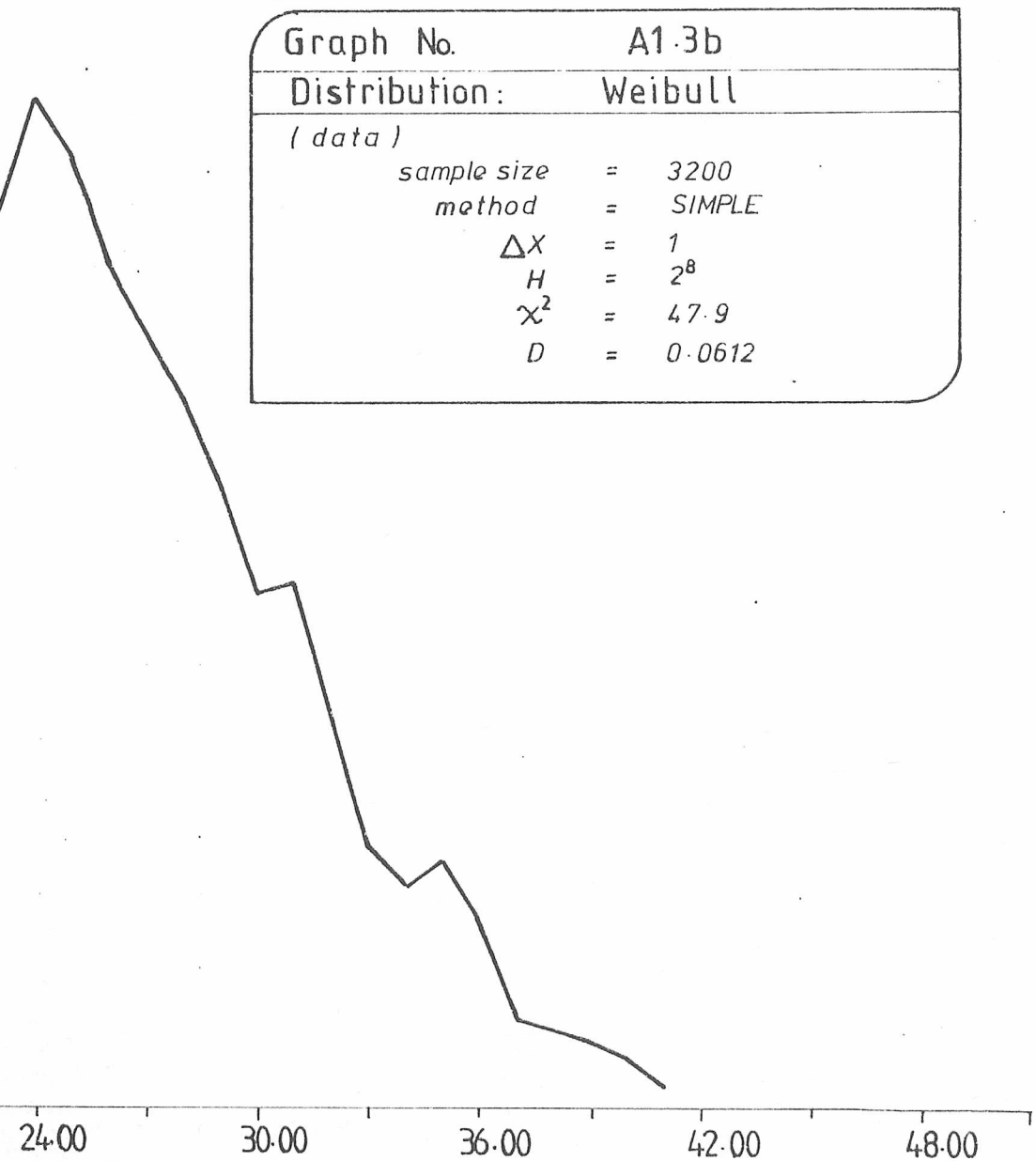




$$F(x) = 1 - e^{-(\lambda x)^{a}}$$

Figure A1·3a Weibull distribution

frequency
0·00
40·00
80·00
120·00
160·00
200·00
0·00
6·00
12·00
18·00
gap

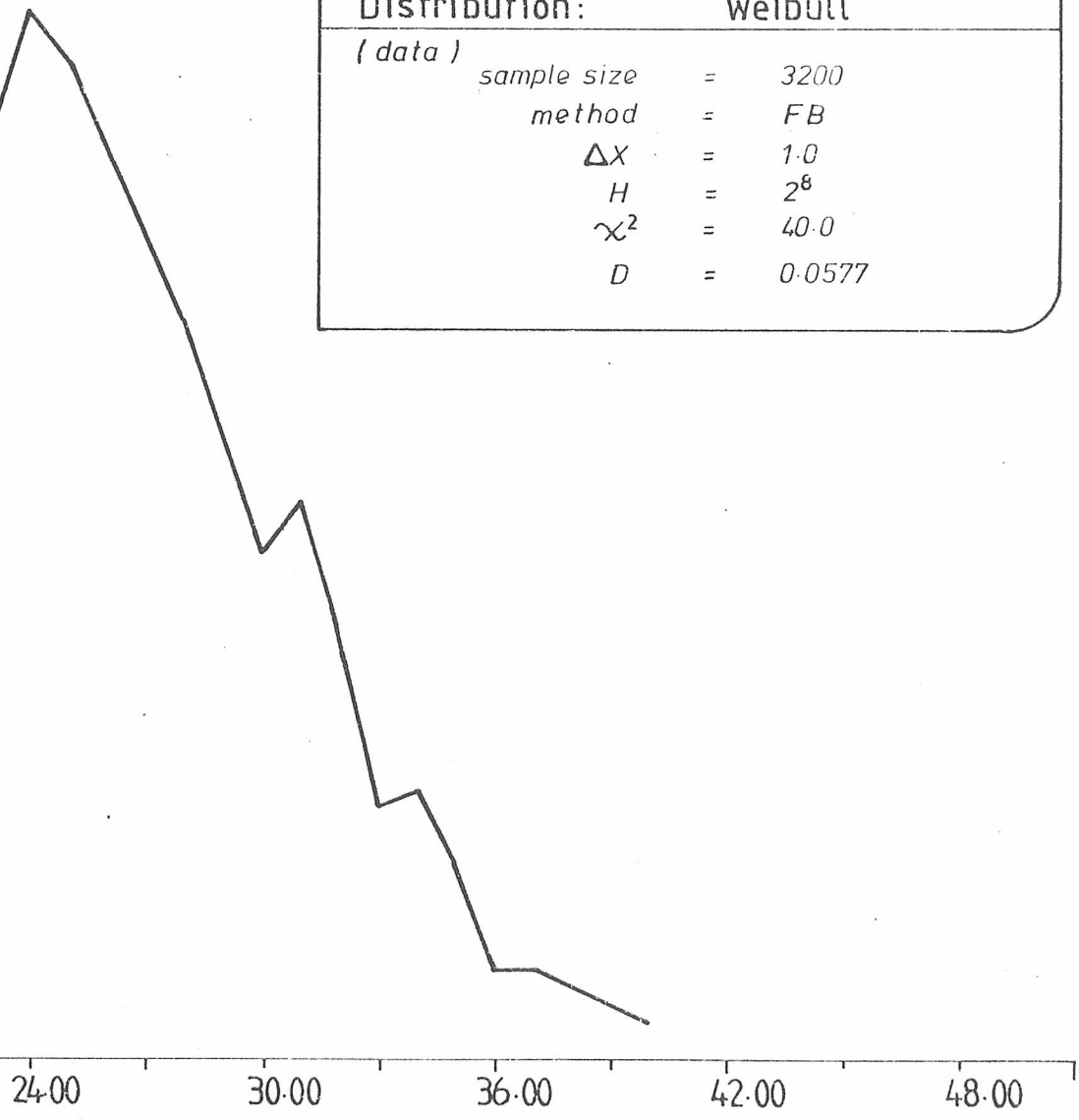

Graph No. A1·3b
Distribution: Weibull
(data)
sample size = 3200
method = SIMPLE
ΔX = 1
H = 2^8
χ² = 47·9
D = 0·0612
24·00
30·00
36·00
42·00
48·00

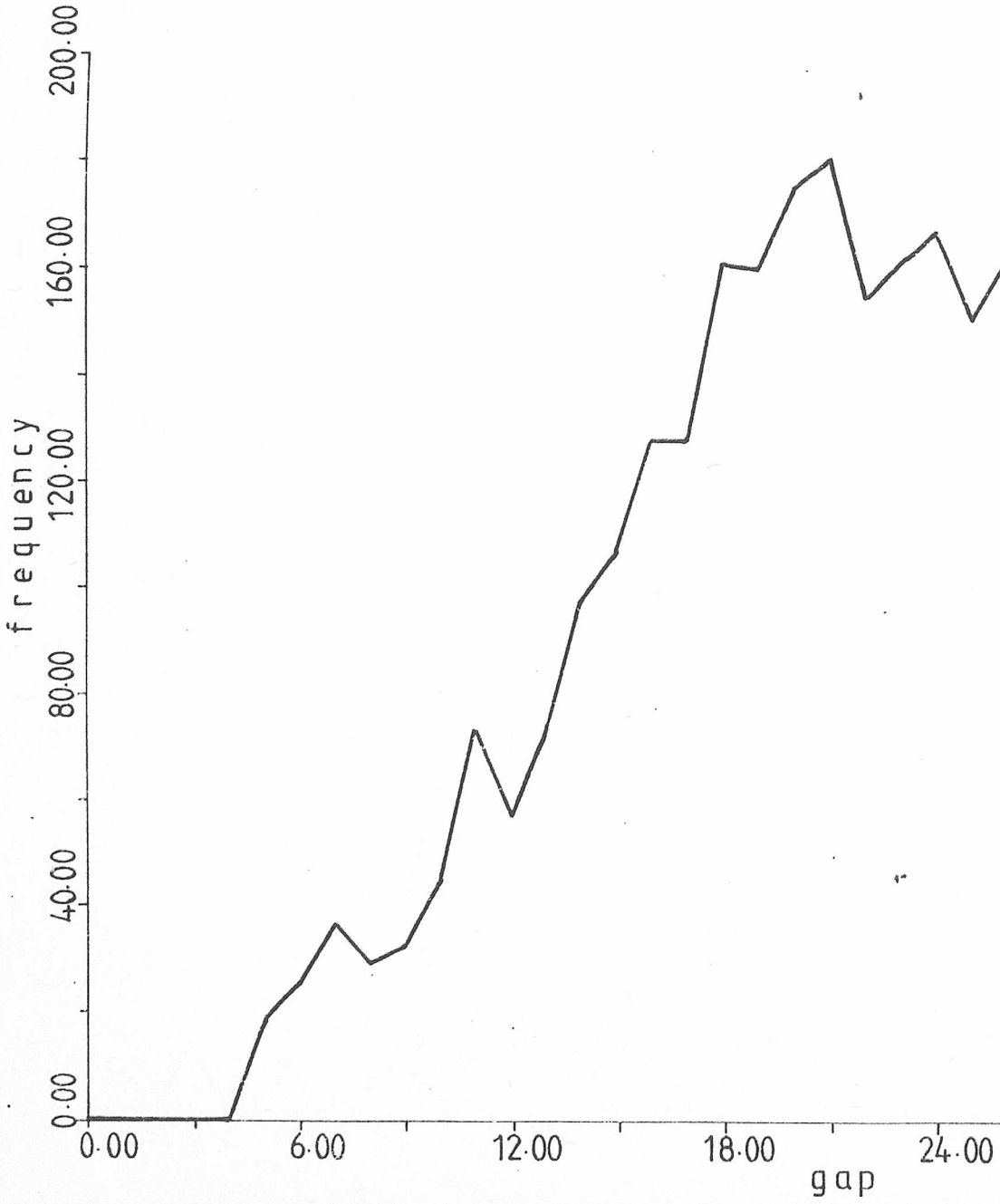
frequency
0.00
40.00
80.00
120.00
160.00
200.00
0.00
6.00
12.00
18.00
gap

Graph No.	A1·3c
Distribution:	Weibull

(data)

sample size	=	*3200*
method	=	*FB*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*40·0*
D	=	*0·0577*

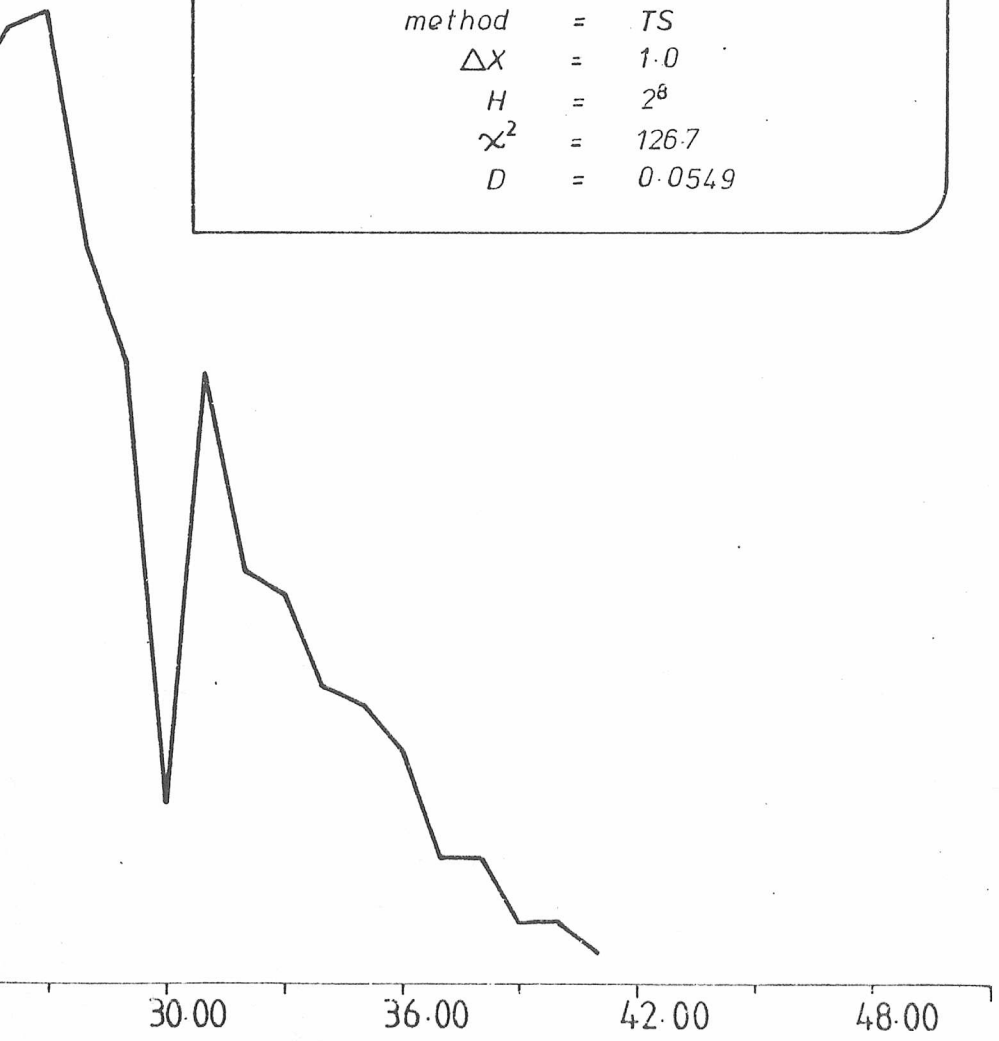



frequency
0.00
40.00
80.00
120.00
160.00
200.00
0.00
6.00
12.00
18.00
24.00
gap

Graph No.	A1·3d
Distribution:	Weibull

(data)

sample size	=	*3200*
method	=	*TS*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*126·7*
D	=	*0·0549*

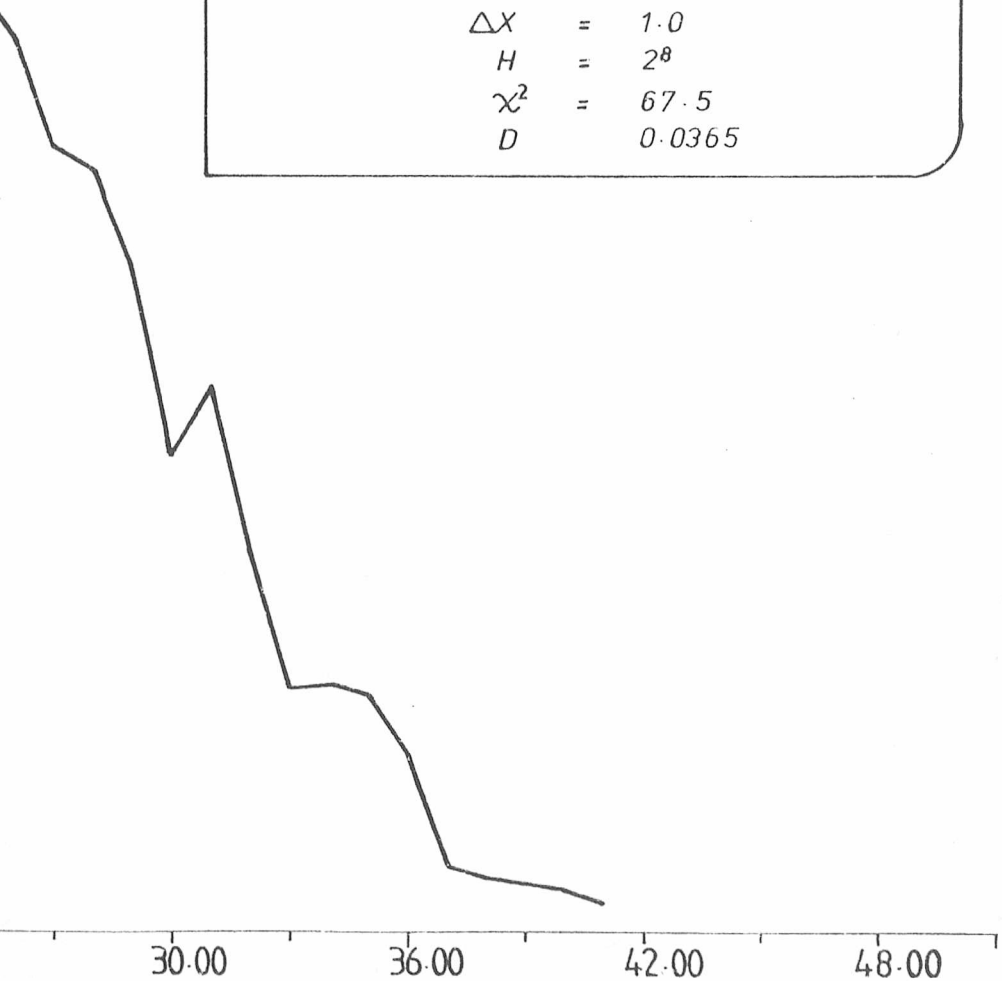


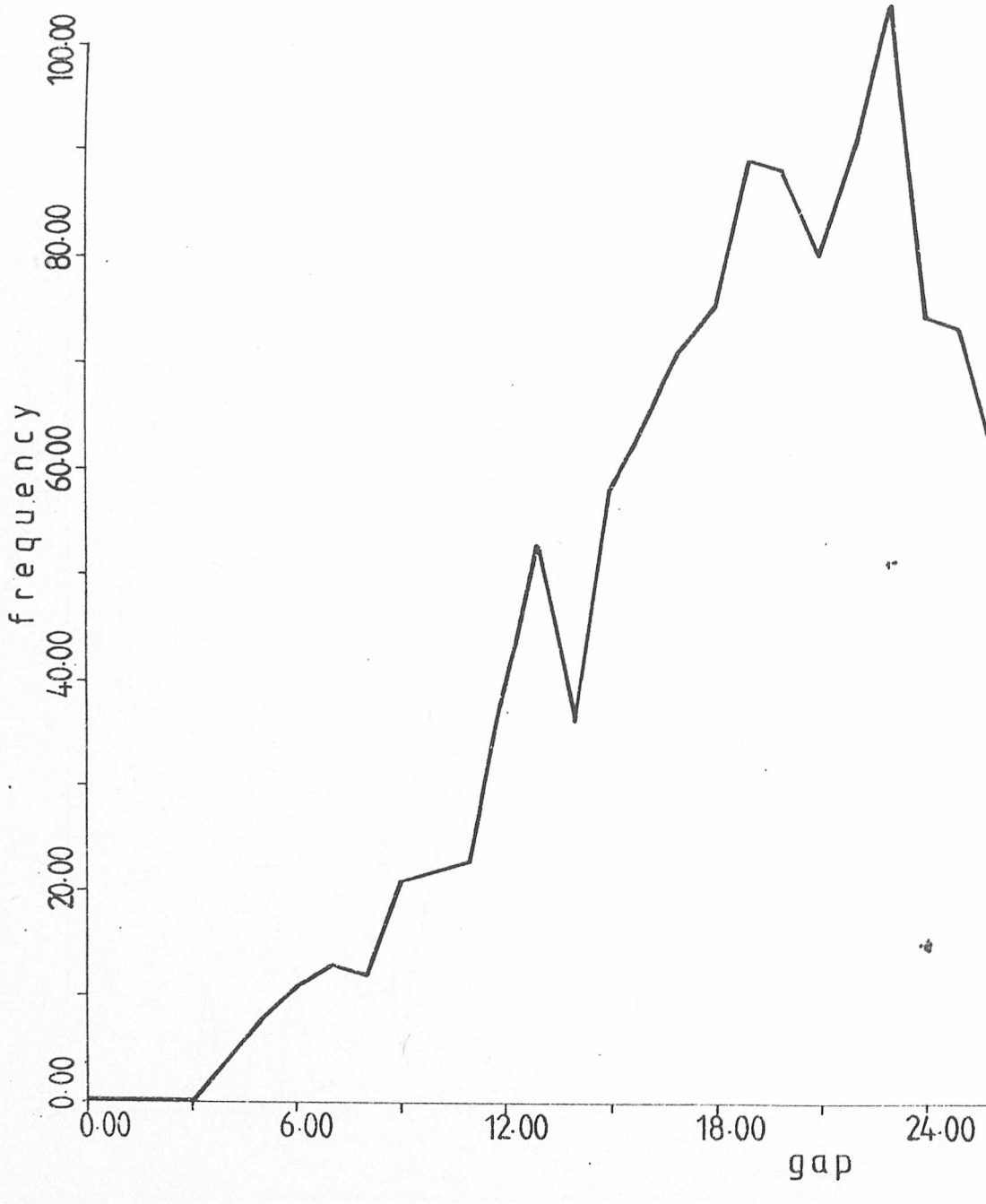
frequency
0.00
40.00
80.00
120.00
160.00
200.00
0.00
6.00
12.00
18.00
24.00
gap

Graph No.	A1·3e
Description:	Weibull

(data)

sample size	=	*3200*
method	=	*FB and TS*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*67·5*
D		*0·0365*

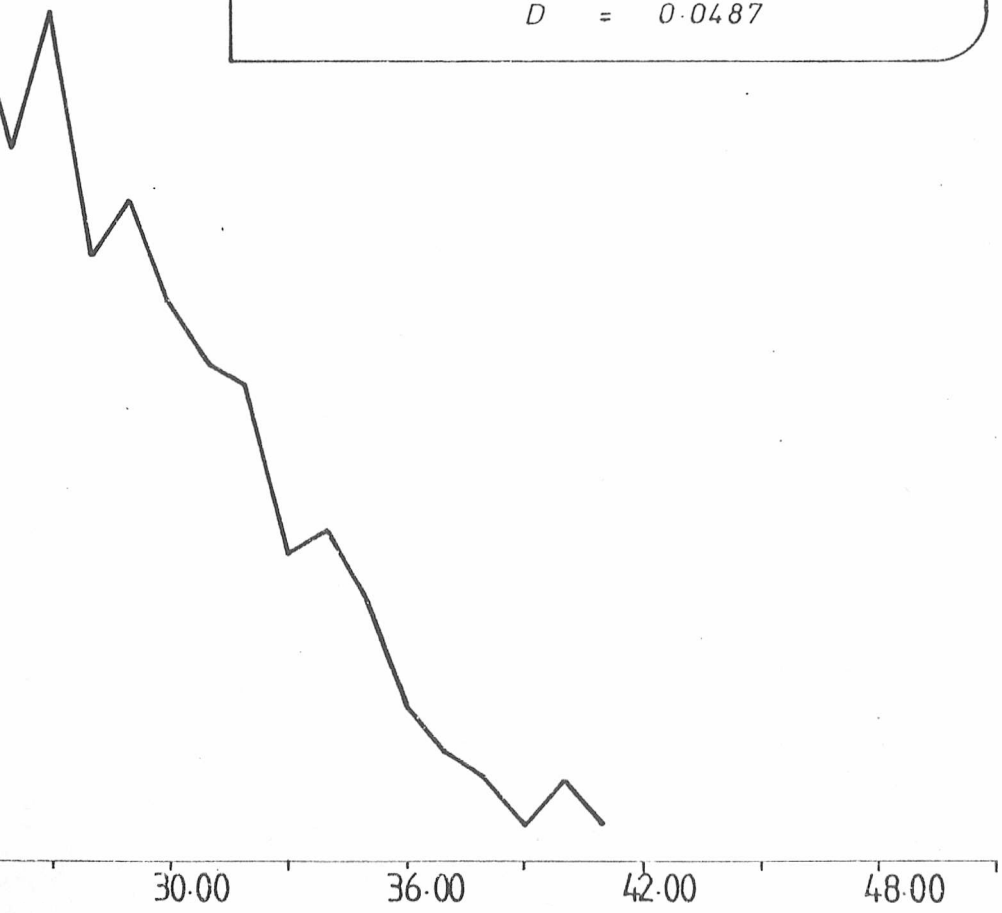



frequency
0·00
20·00
40·00
60·00
80·00
100·00
0·00
6·00
12·00
18·00
24·00
gap

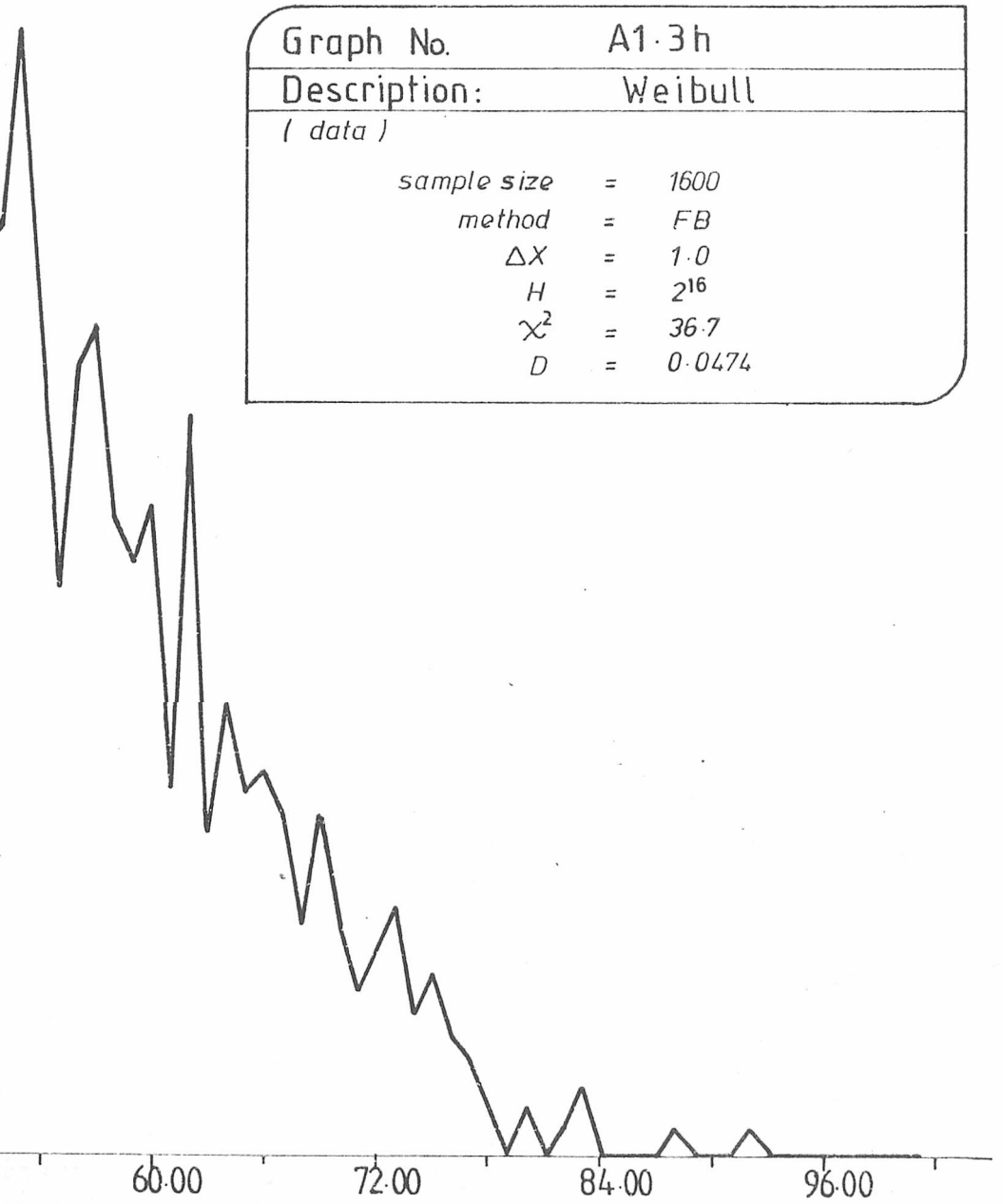
Graph No.	A1·3f
Description:	Weibull

(data)

sample size	=	*1600*
method	=	*FB*
ΔX	=	*1*
H	=	2^{16}
χ^2	=	*30·2*
D	=	*0·0487*



frequency
0.00
10.00
20.00
30.00
40.00
50.00
0.00
12.00
24.00
36.00
48.00
gap

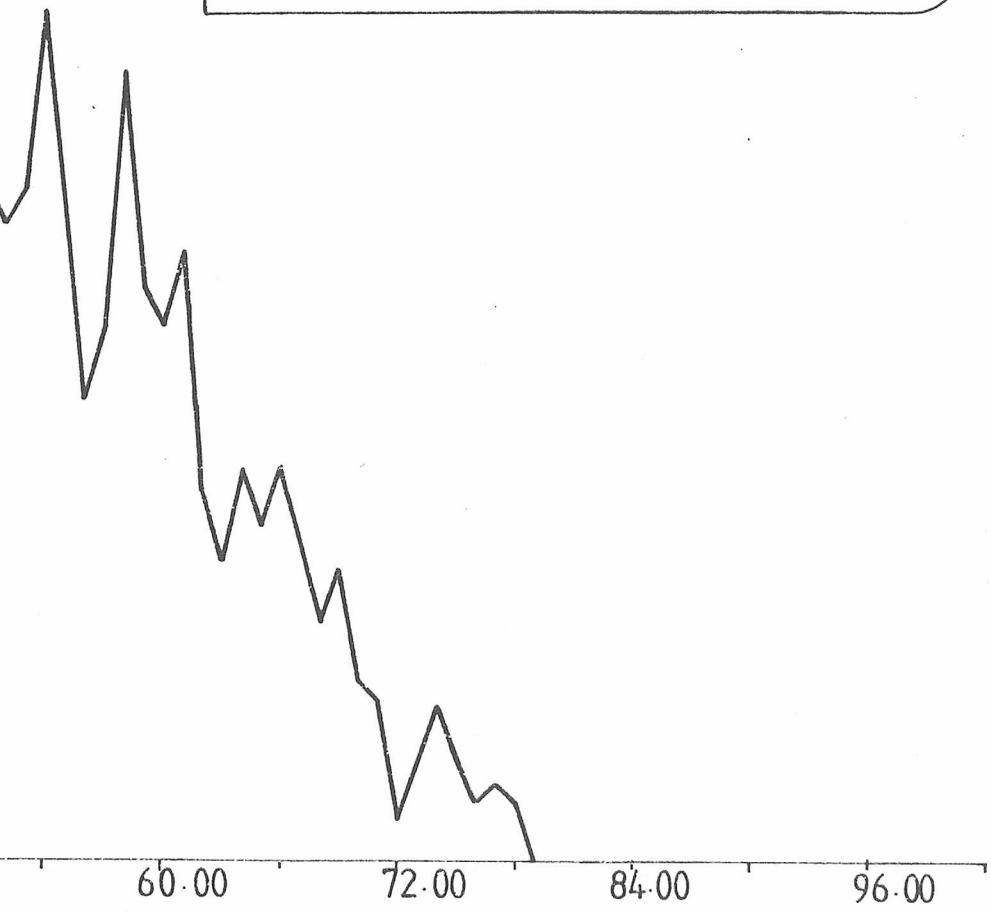

Graph No. A1·3h
Description: Weibull
(data)
sample size = 1600
method = FB
ΔX = 1·0
H = 2^16
χ² = 36·7
D = 0·0474
60·00
72·00
84·00
96·00

frequency
60.00
48.00
36.00
24.00
12.00
0.00
0.00
12.00
24.00
36.00
48.00
gap

Graph No.	A1·3i
Description:	Weibull

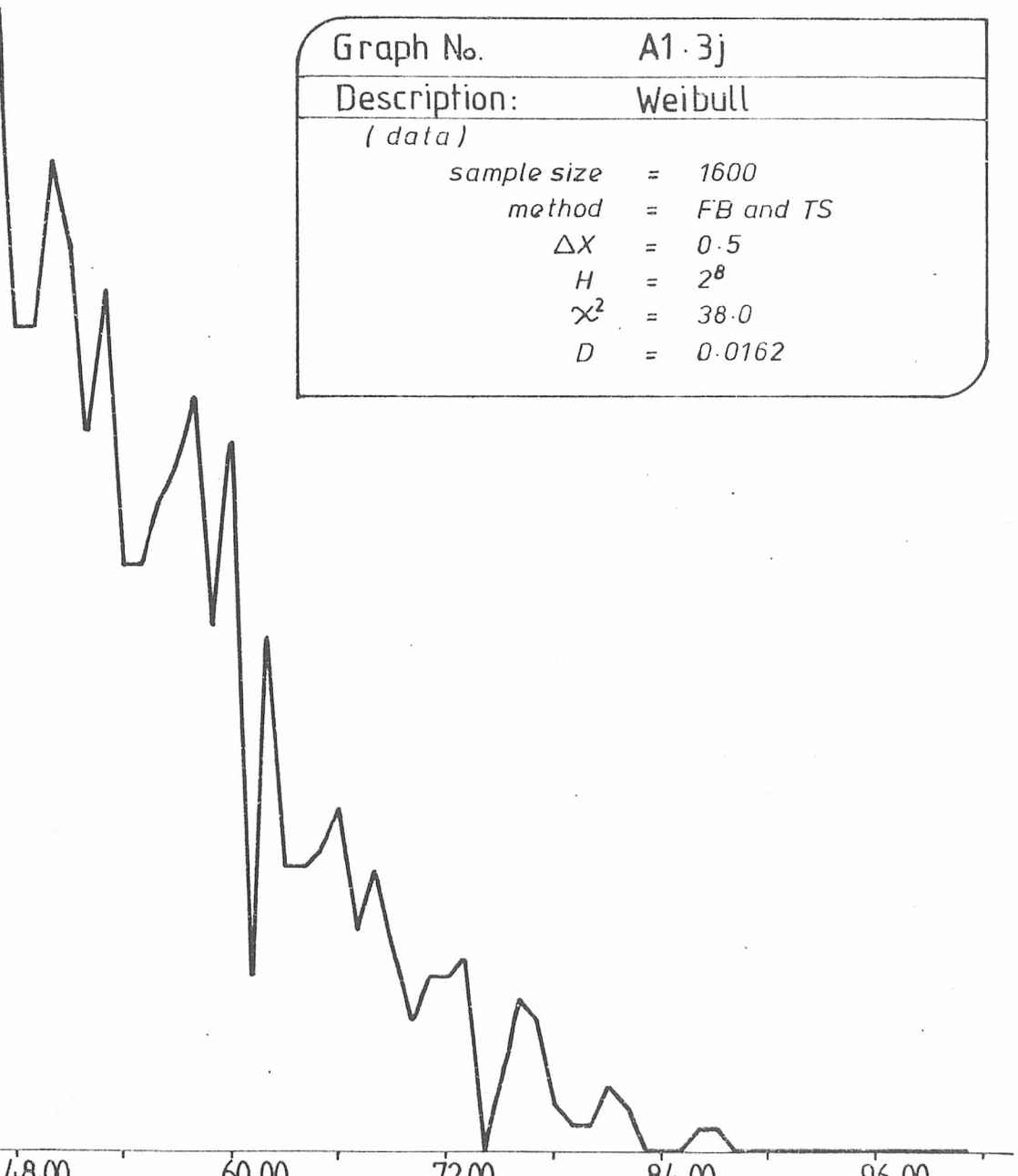
(data)

sample size	=	*1600 .*
method	=	*FB*
ΔX	=	*0·5*
H	=	2^{16}
χ^2	=	*27·5*
D		*0·0499*



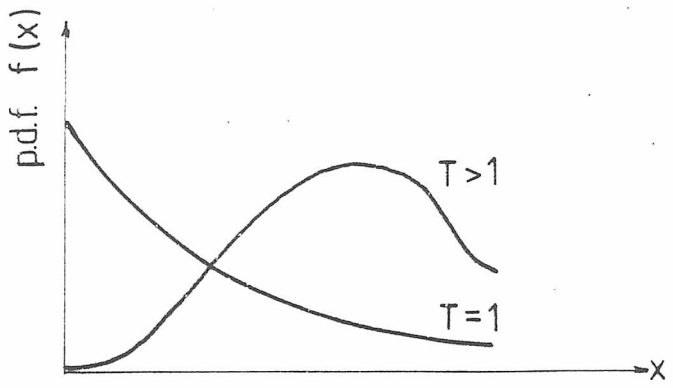
frequency
0.00
10.00
20.00
30.00
40.00
50.00
0.00
12.00
24.00
36.00
gap

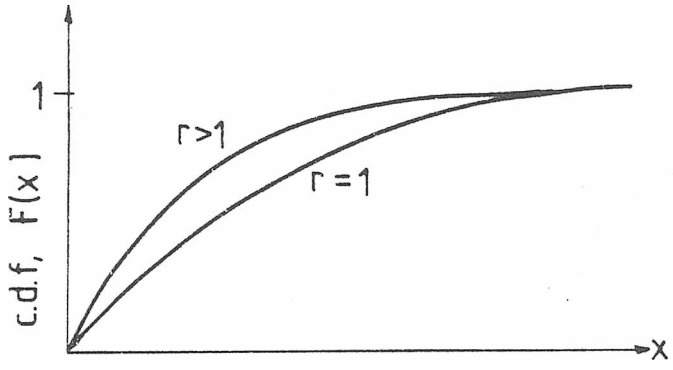
Graph No.	A1·3j
Description:	Weibull
(data)	
sample size	= 1600
method	= FB and TS
ΔX	= 0·5
H	= 2^8
χ^2	= 38·0
D	= 0·0162

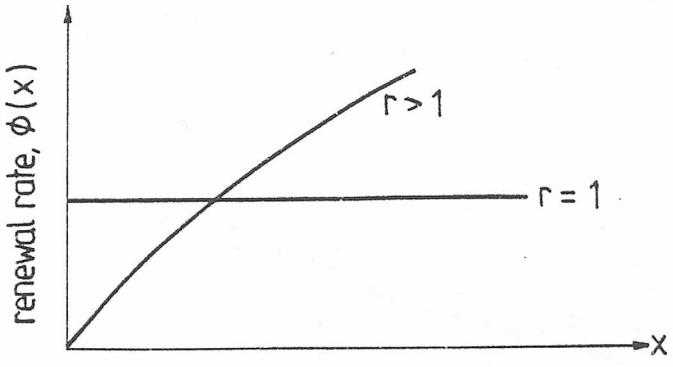


WEIBULL DISTRIBUTION		number of samples	number of bits of random number	Δx	K-S TEST			$\chi^2_{n,\alpha}$ TEST	
method	figure				value of result	value expected	value critical	result	critical value
simple	4.3b	3200	8	1.0	0.0612	0.0710	0.034	47.9	52.2
FB	4.3c	3200	8	1.0	0.0577	0.0558	0.034	40.0	52.2
TS	4.3d	3200	8	1.0	0.0549	——	0.034	126.7	52.2
FB & TS	4.3e	3200	8	1.0	0.0365	0.0286	0.034	67.5	52.2
FB	4.3f	1600	16	1.0	0.0487	0.0538	0.048	30.2	50.9
simple	4.3g	1600	8	0.5	0.0455	0.0693	0.048	34.8	40.1
FB	4.3h	1600	8	0.5	0.0474	0.0286	0.048	36.7	40.1
FB	4.3i	1600	16	0.5	0.0499	0.0268	0.048	27.5	40.1
FB & TS	4.3j	1600	8	0.5	0.0162	0.0155	0.048	38.0	40.1

Table A1.3k Table of results and critical values ($\alpha = 0.95$)

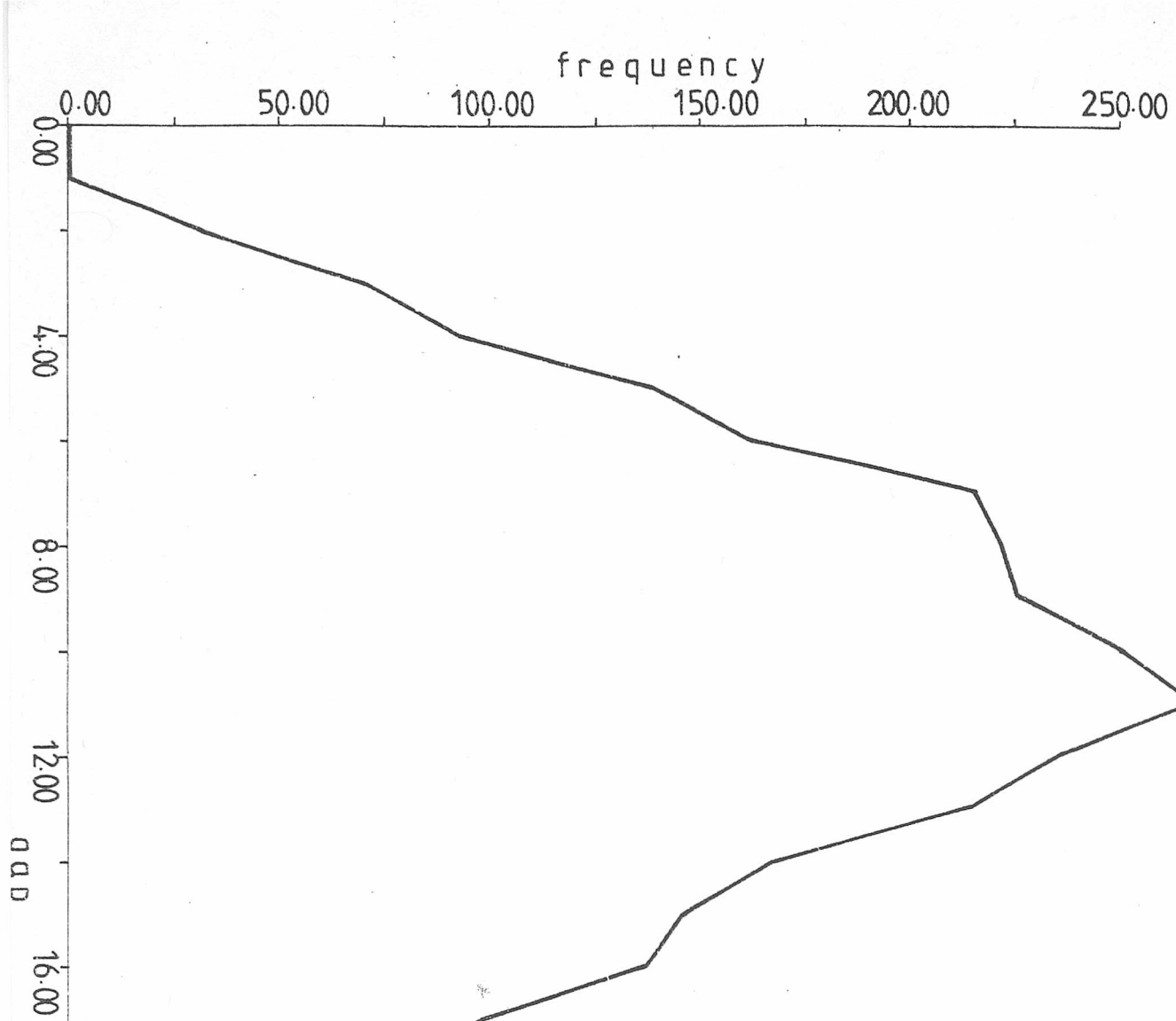






$$f(x) = \frac{\lambda^r \, x^{r-1} \, e^{-\lambda x}}{(r-1)} \qquad x > 0$$

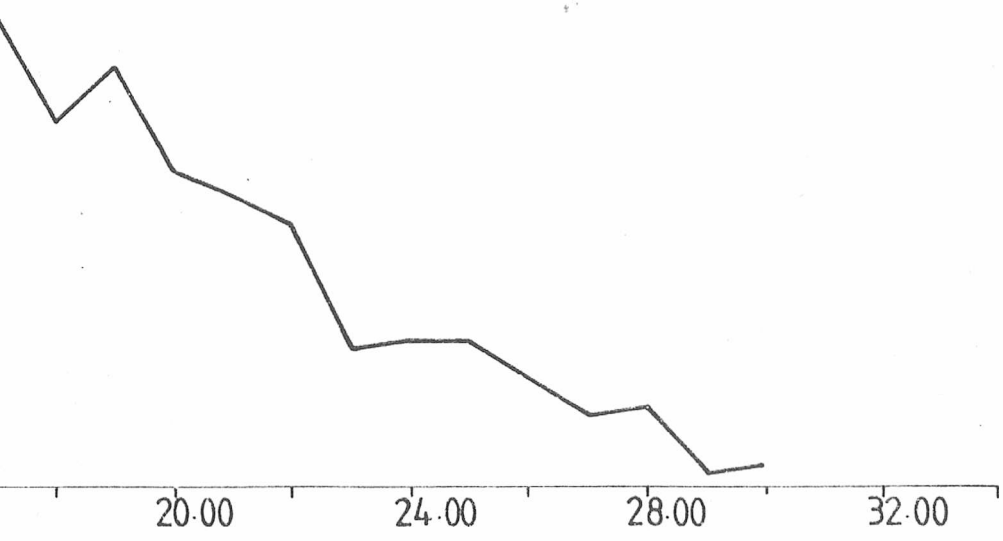
Figure A1·4a Erlang distribution


frequency
0.00
50.00
100.00
150.00
200.00
250.00
0.00
4.00
8.00
12.00
16.00
hour

Graph No.	A1 · 4b
Description:	Erlang

(data)

sample size	=	*3200*
method	=	*SIMPLE*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*58·4*
D	=	*0·0857*

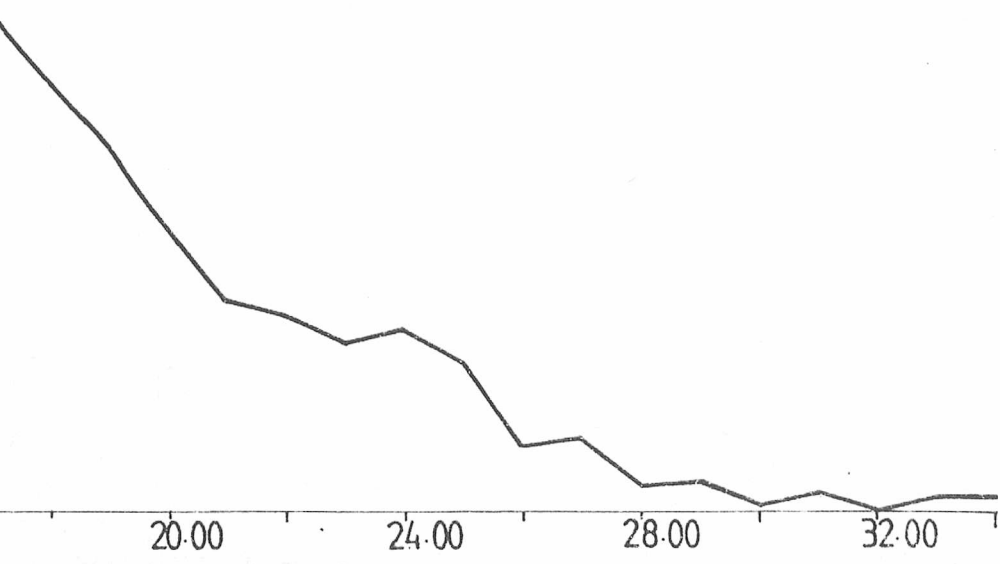


frequency
250.00
200.00
150.00
100.00
50.00
0.00
0.00
4.00
8.00
12.00
16.00

Graph No.	A1·4c
Description:	Erlang

(data)

sample size	=	3200
method	=	FB
ΔX	=	1·0
H	=	2^8
χ^2	=	38·4
D	=	0·0789

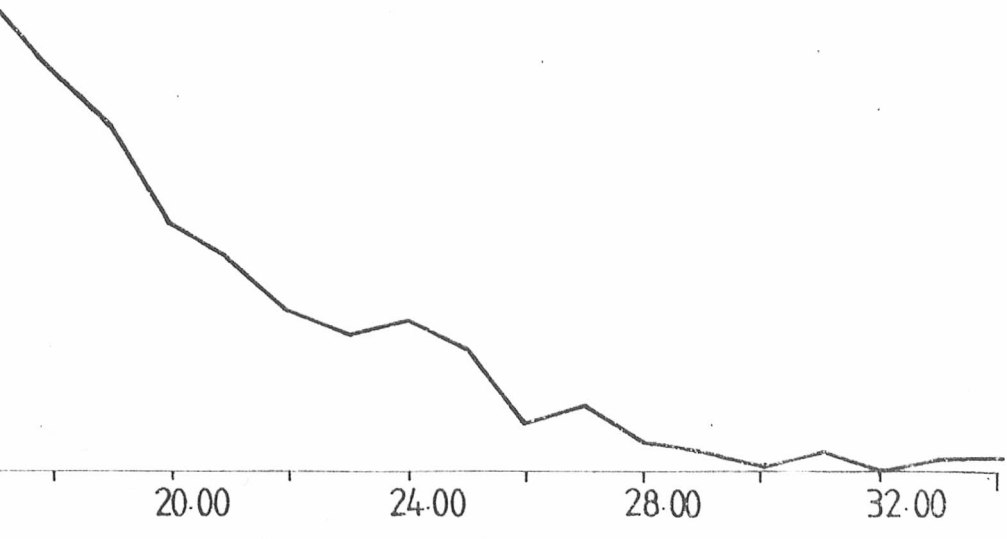


frequency
0.00
50.00
100.00
150.00
200.00
250.00
0.00
4.00
8.00
12.00
16.00
gap

Graph No.	A1·4d
Description:	Erlang

(data)

sample size	=	*3200*
method	=	*TS*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*76·2*
D	=	*0·0495*

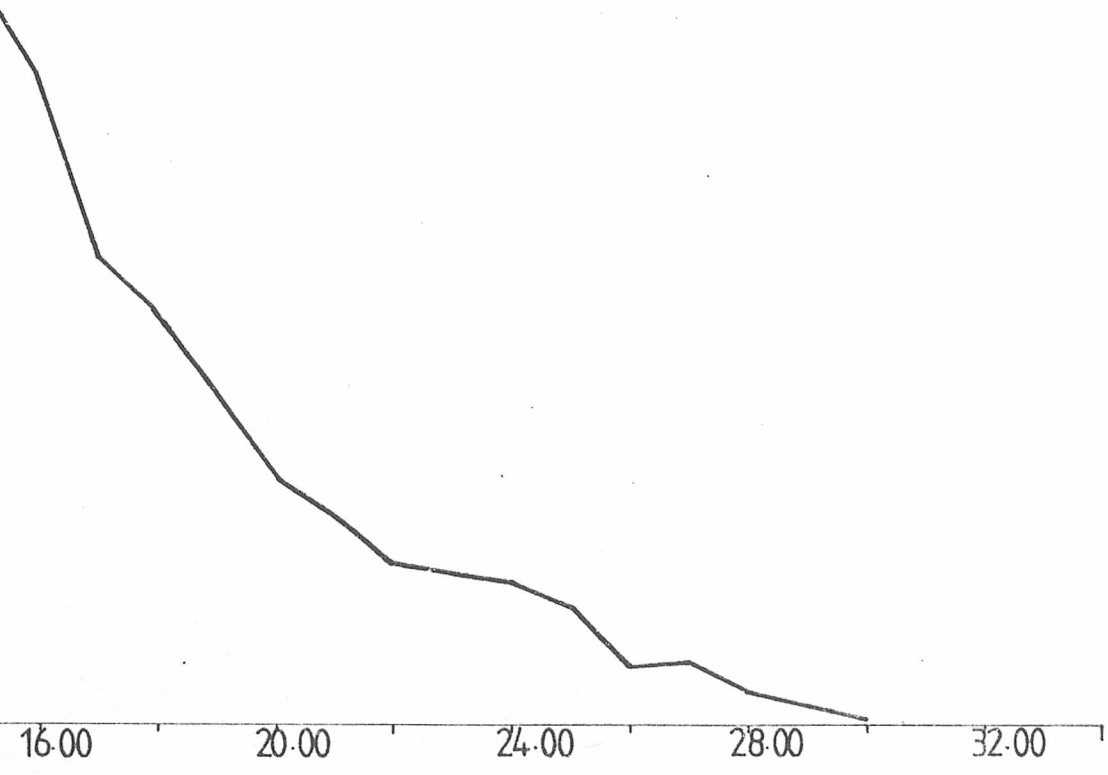


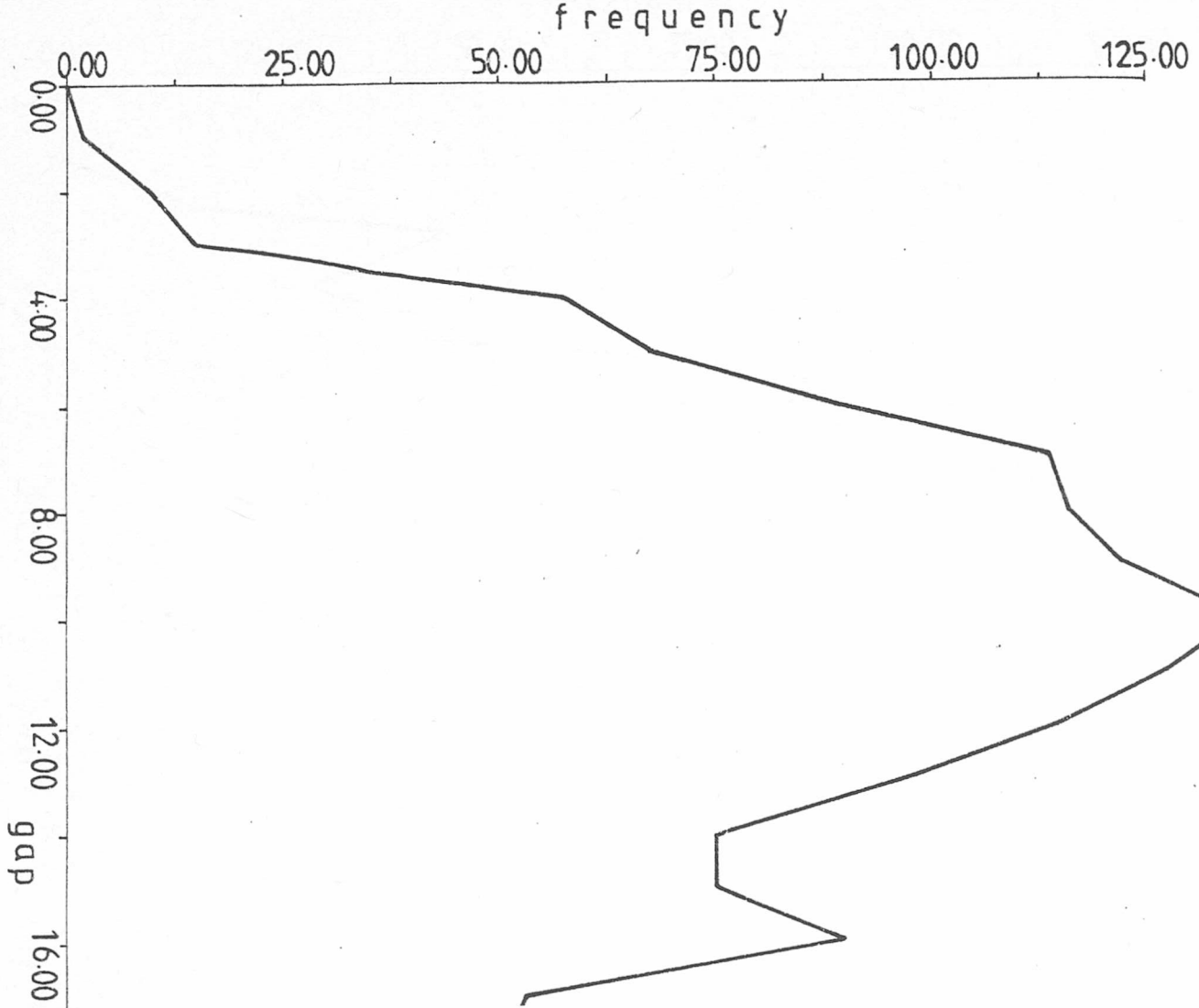
frequency
0.00
50.00
100.00
150.00
200.00
250.00
0.00
4.00
8.00
12.00
gap

Graph No.	A1·4e
Description:	Erlang

(data)

sample size	=	*3200*
method	=	*FB and TS*
ΔX	=	*1·0*
H	=	2^8
χ^2	=	*68·7*
D	=	*0·0504*

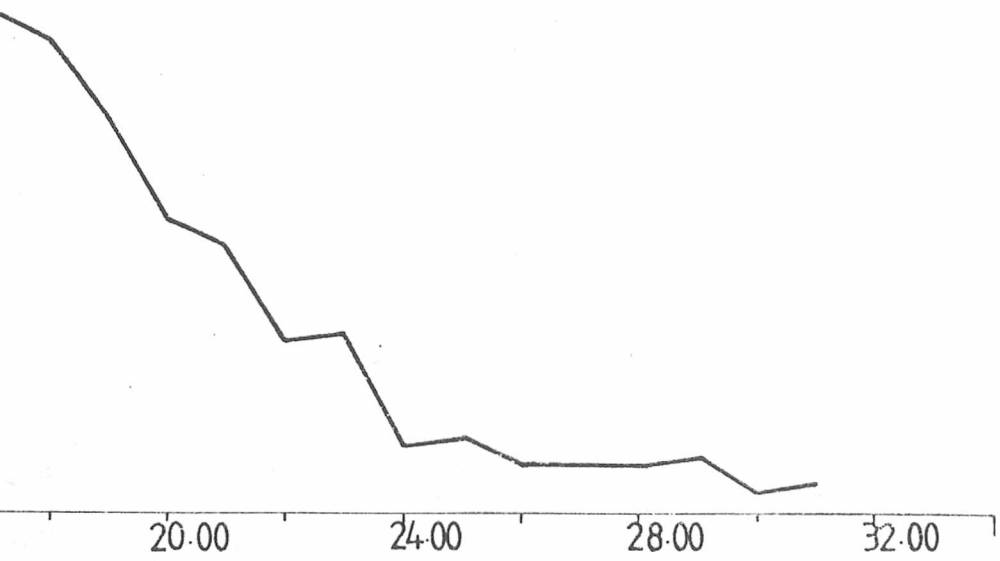


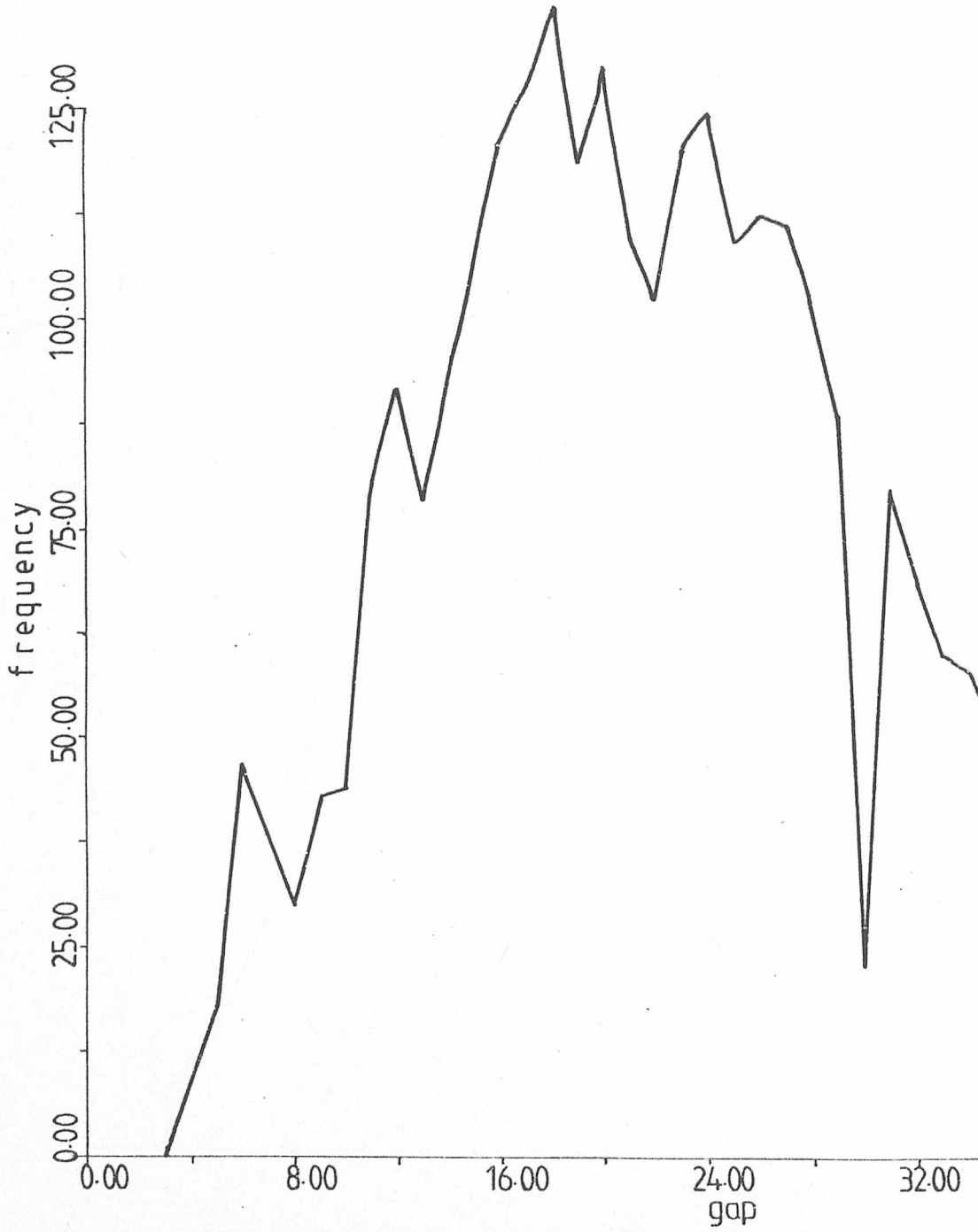

frequency
0.00
25.00
50.00
75.00
100.00
125.00
0.00
4.00
8.00
12.00
16.00
gap

Graph No.	A1·4f
Description:	Erlang

(data)

sample size	=	*1600*
method	=	*FB*
ΔX	=	*1·0*
H	=	2^{16}
χ^2	=	*27·6*
D	=	*0·0776*




frequency
0.00
25.00
50.00
75.00
100.00
125.00
0.00
8.00
16.00
24.00
32.00
gap

Graph No. A1 · 4g

Description: Erlang

(data)

sample size = 3200

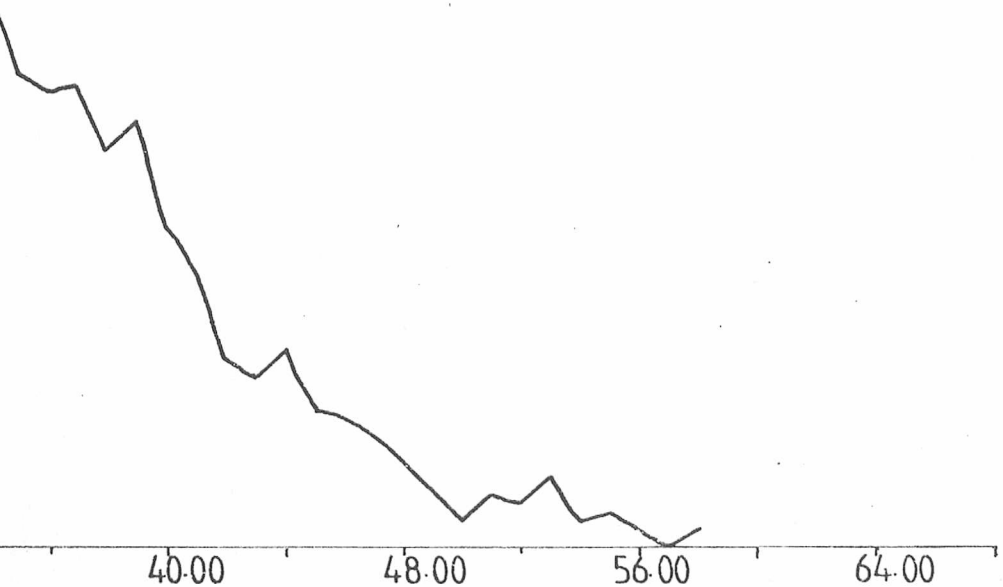
method = SIMPLE

ΔX = 0·5

H = 2^8

χ^2 = 110·8

D = 0·0469

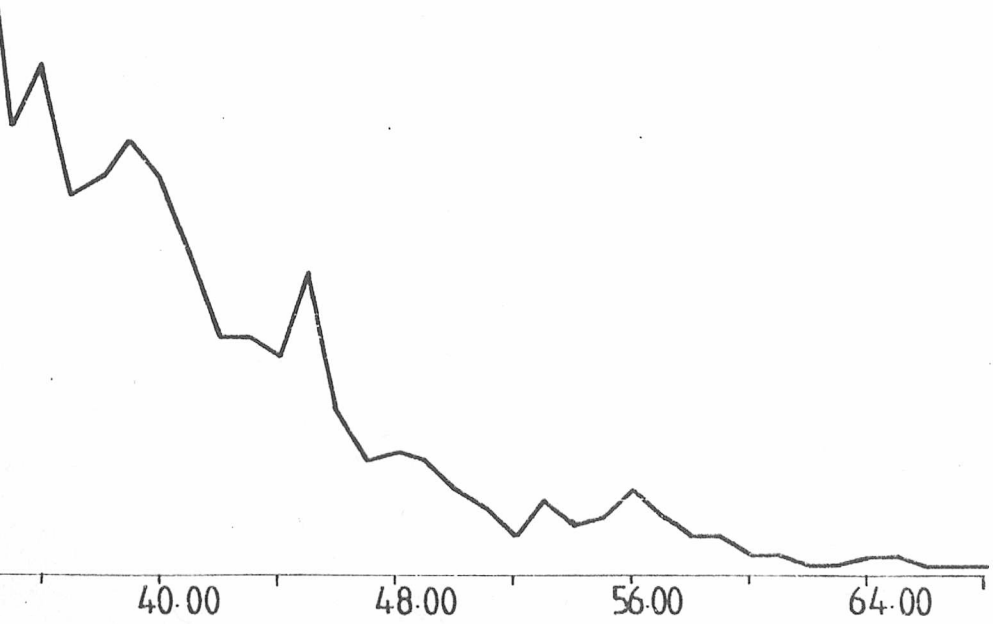


frequency
0.00
25.00
50.00
75.00
100.00
125.00
0.00
8.00
16.00
24.00
32.00
gap

Graph No	A1.4h
Description:	Erlang

(data)

sample size	=	3200
method	=	FB
ΔX	=	0.5
H	=	2^8
χ^2	=	36.6
D	=	0.0548



frequency
0.00
25.00
50.00
75.00
100.00
125.00
0.00
8.00
16.00
24.00
gap

Graph No.	A1·4i
Description:	Erlang

(data)

sample size = 3200

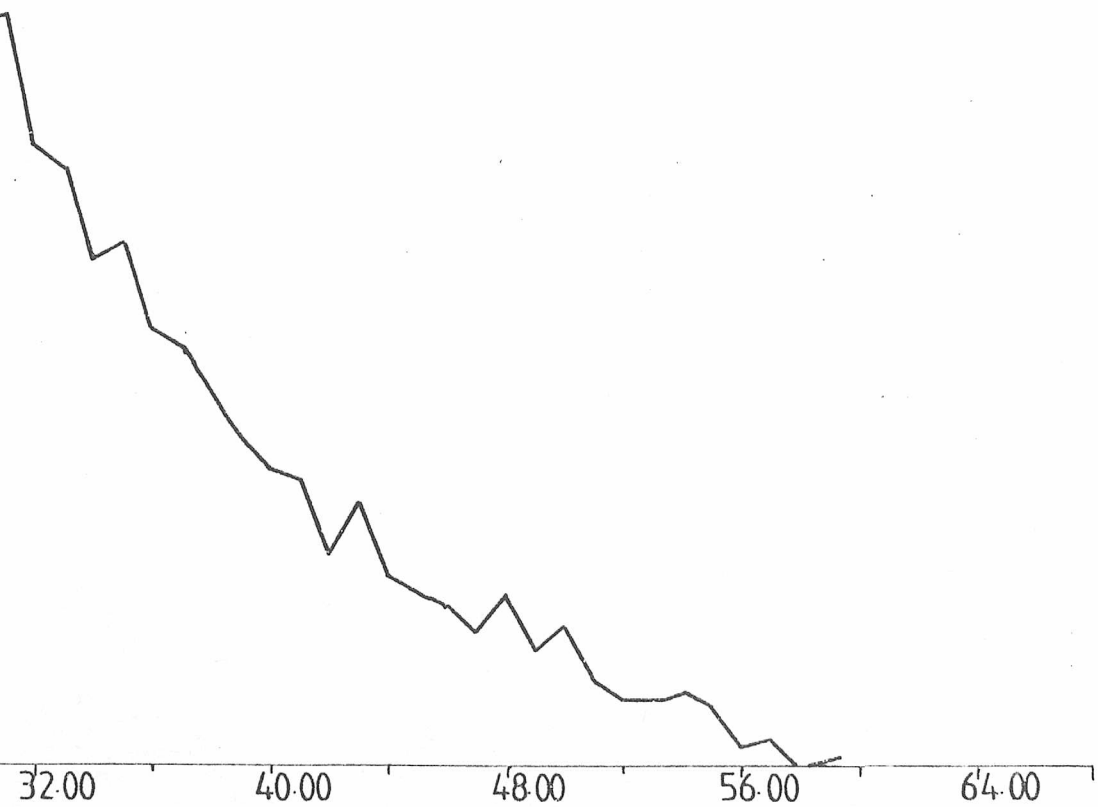
method = FB

ΔX = 0·5

H = 2^{16}

χ^2 = 41·2

D = 0·0525

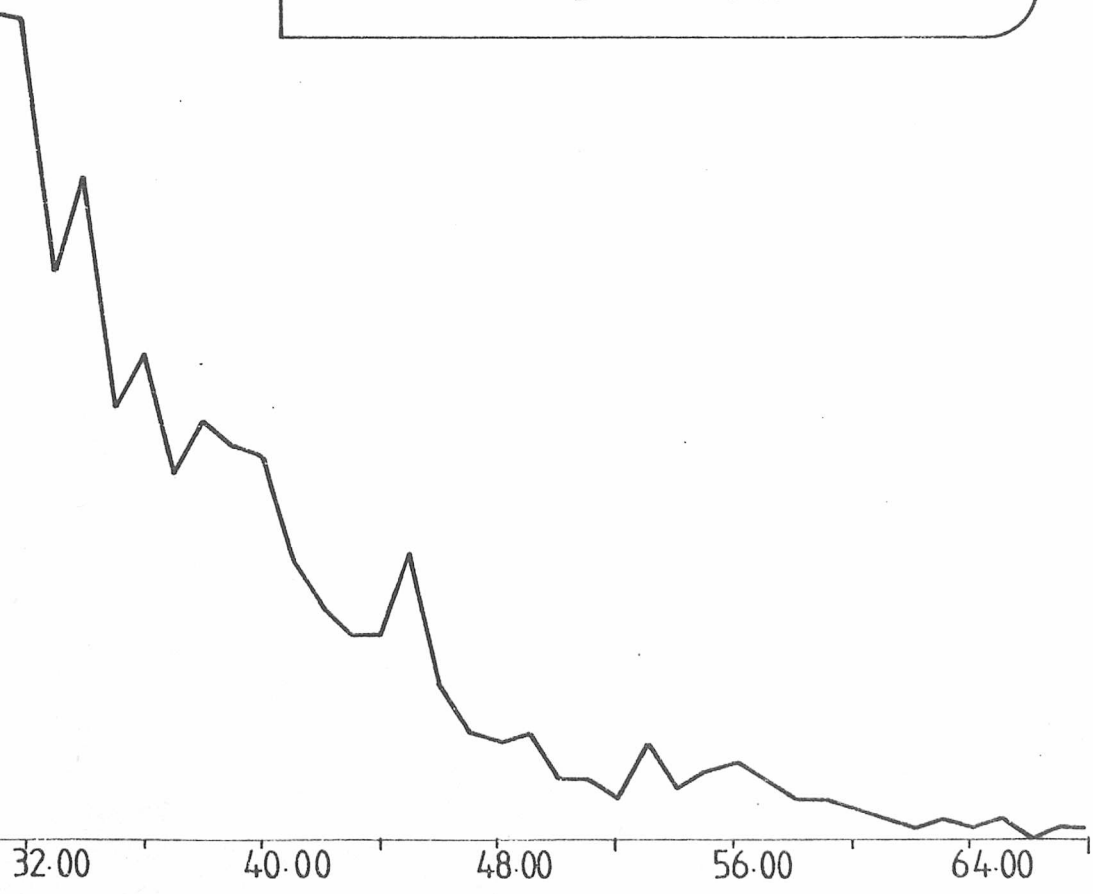


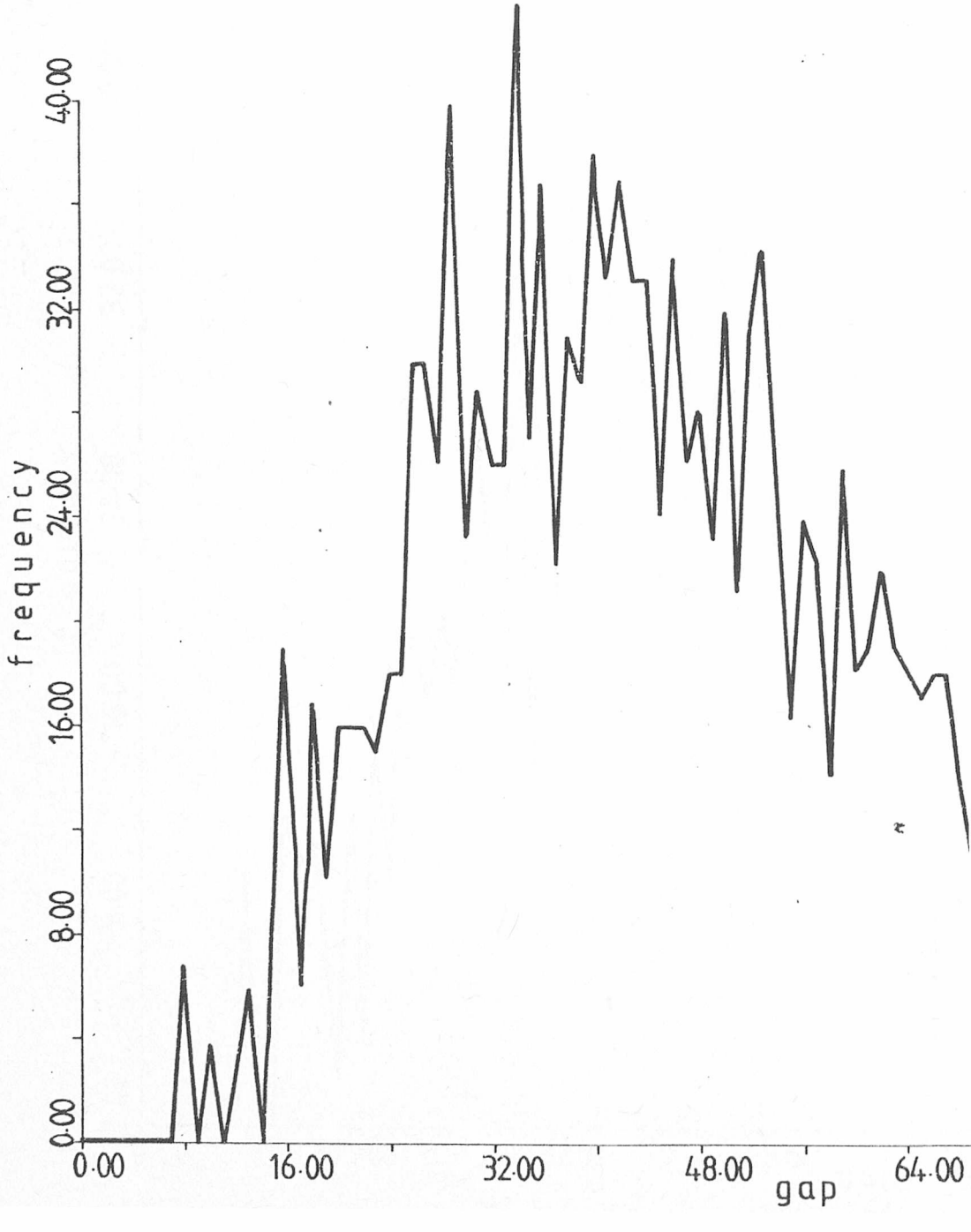
125.00
100.00
75.00
50.00
25.00
0.00
frequency
0.00
8.00
16.00
24.00
gap

Graph No.	A1 · 4j
Description:	Erlang

(data)

sample size	=	*3200*
method	=	*FB and TS*
ΔX	=	*0·5*
H	=	2^8
χ^2	=	*75·7*
D	=	*0·0269*

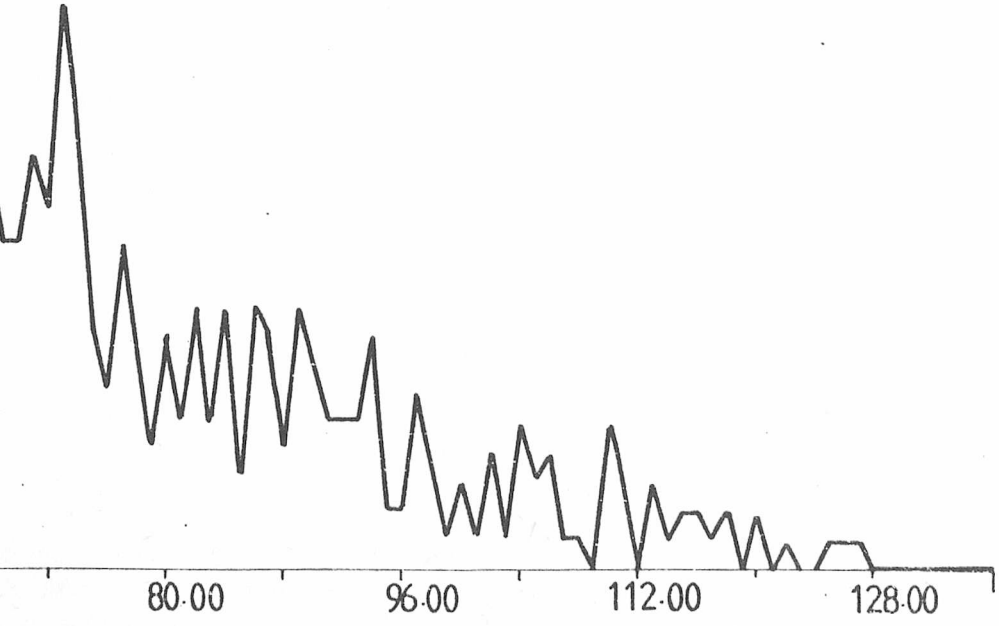



frequency
0.00
8.00
16.00
24.00
32.00
40.00
0.00
16.00
32.00
48.00
64.00
gap

Graph No.	A1 · 4ℓ
Description:	Erlang

(data)

sample size	=	*1600*
method	=	*FB and TS*
ΔX	=	*0·25*
H	=	2^8
χ^2	=	*35·6*
D	=	*0·024*

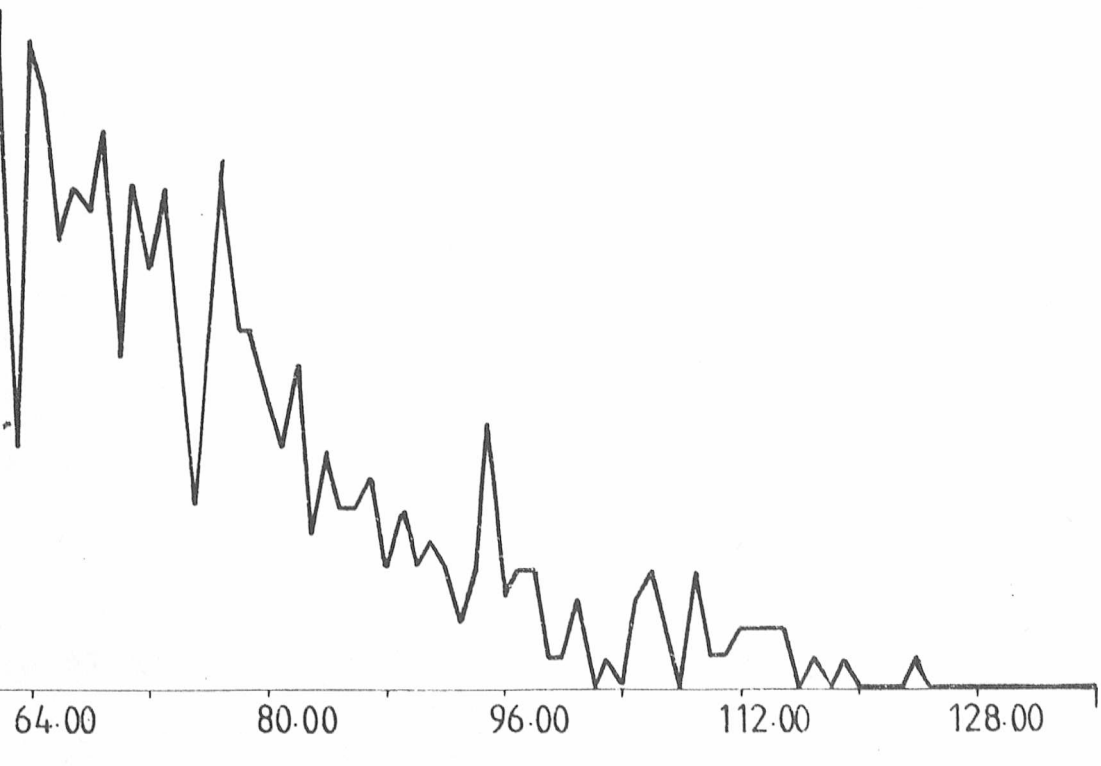


frequency
0.00
8.00
16.00
24.00
32.00
40.00
0.00
16.00
32.00
48.00
gap

Graph No.	A1 . 4m
Description:	Erlang

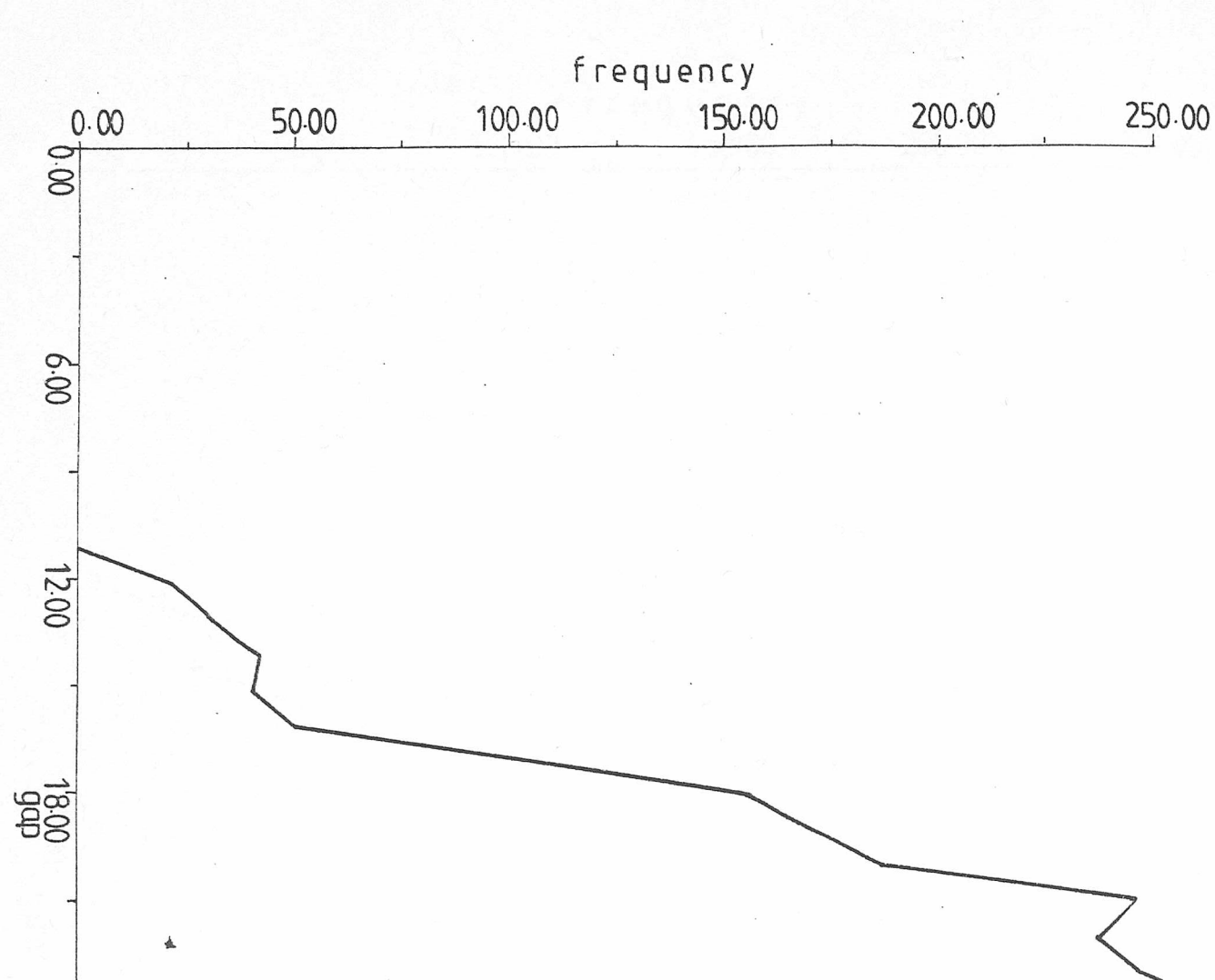
(data)

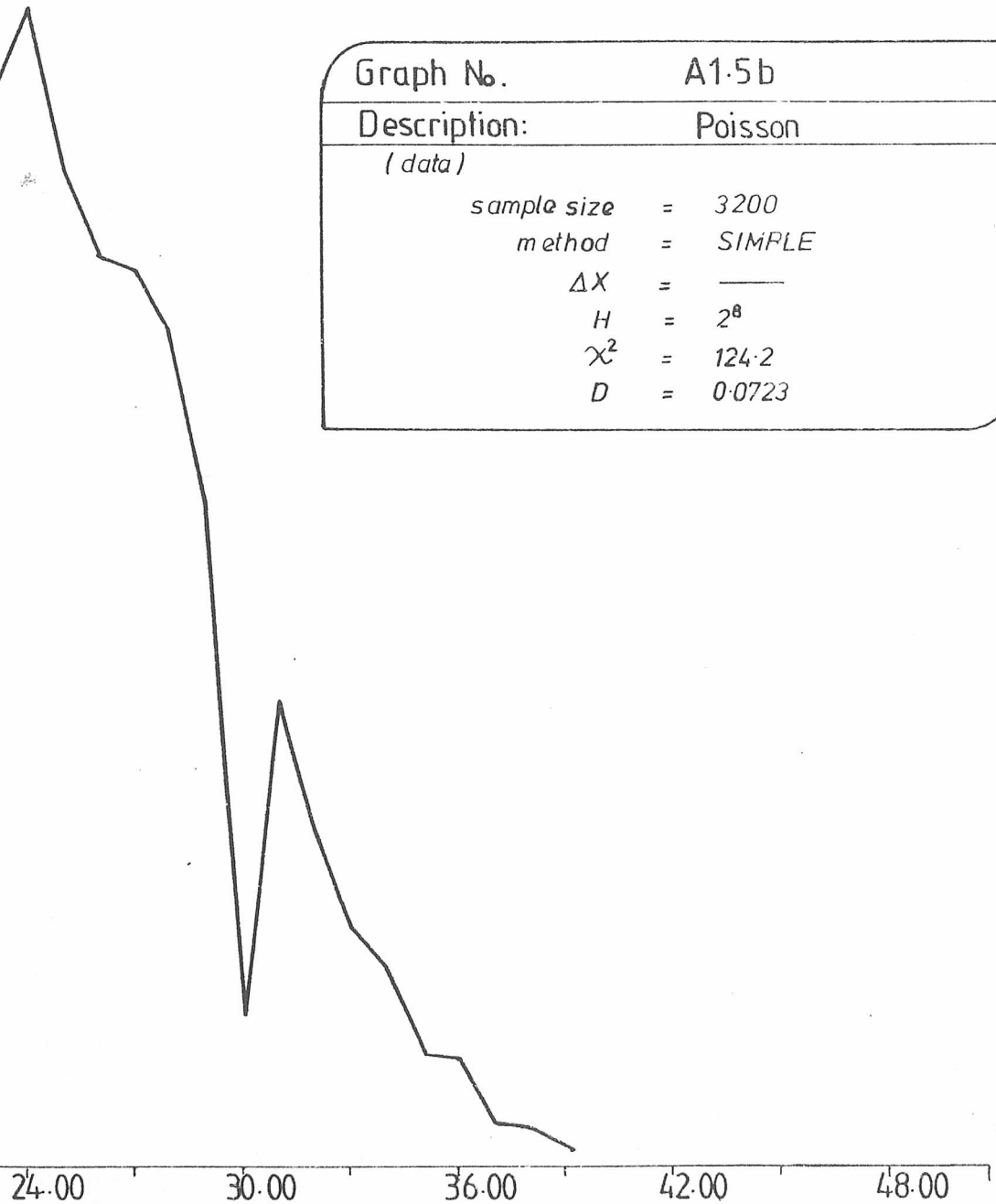
sample size	=	*1600*
method	=	*FB*
ΔX	=	*0·25*
H	=	2^8
χ^2	=	*45·1*
D	=	*0·0239*



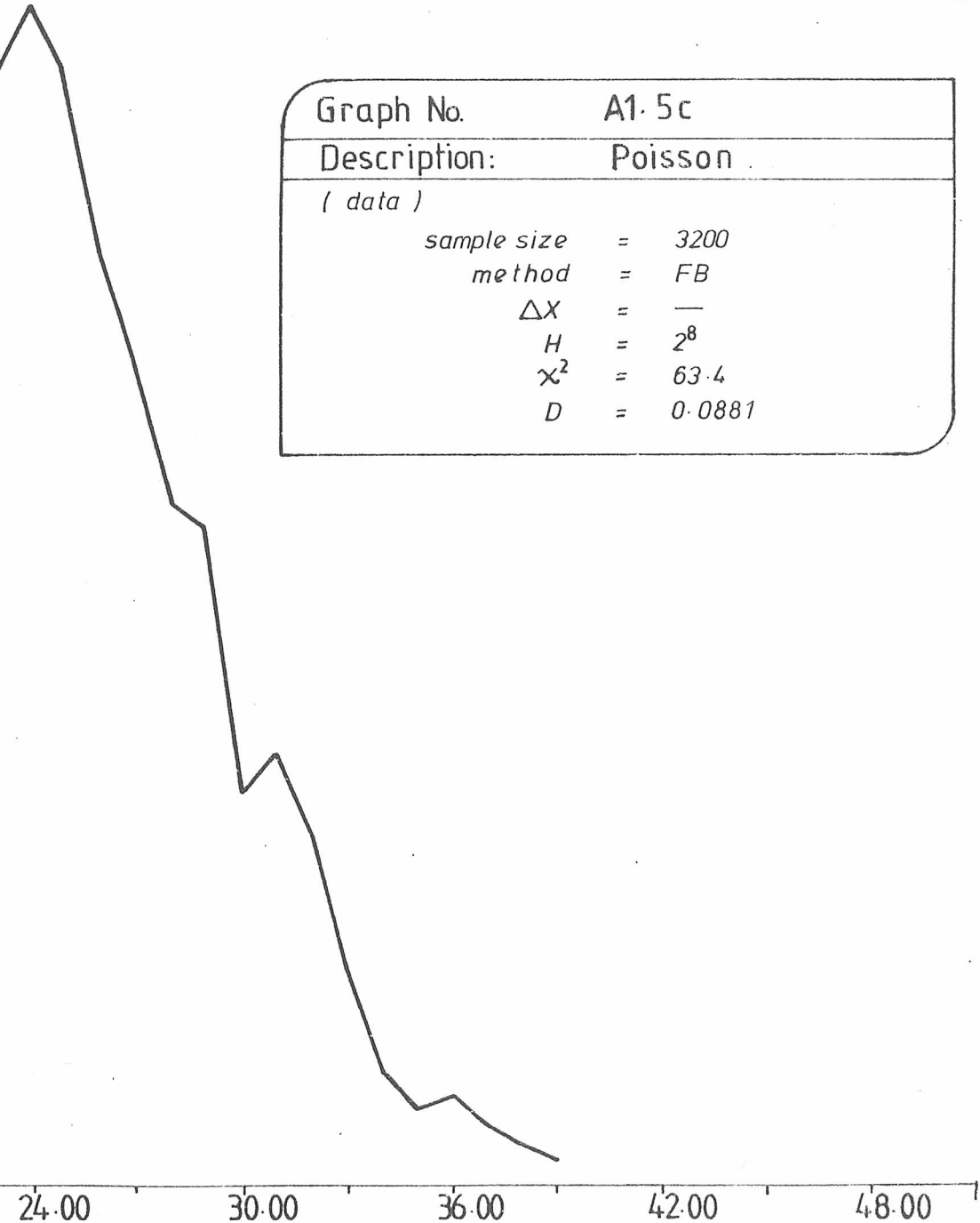
ERLANG DISTRIBUTION		number of samples	number of bits of random number	Δx	K-S TEST		χ^2 TEST	
method	figure				result	critical value	result	critical value
simple	4.4b	3200	8	1.0	0.0857	0.034	58.4	42.6
FB	4.4c	3200	8	1.0	0.0789	0.034	38.4	42.6
TS	4.4d	3200	8	1.0	0.0495	0.034	76.2	42.6
FB &TS	4.4e	3200	8	1.0	0.0504	0.034	68.7	42.6
FB	4.4f	1600	16	1.0	0.0776	0.048	27.6	42.6
simple	4.4g	3200	8	0.5	0.0469	0.034	110.8	40.1
FB	4.4h	3200	8	0.5	0.0548	0.034	36.6	40.1
FB	4.4i	3200	16	0.5	0.0525	0.034	41.2	40.1
FB &TS	4.4j	3200	8	0.5	0.0269	0.034	75.7	40.1
simple	4.4k	1600	8	0.25	0.0270	0.048	34.1	28.9
FB. & TS	4.4l	1600	8	0.25	0.0240	0.048	35.6	28.9
FB	4.4m	1600	8	0.25	0.0239	0.048	45.1	28.9

Table A1.4n


frequency
0.00
50.00
100.00
150.00
200.00
250.00
0.00
6.00
12.00
18.00
gap


Graph No. A1·5b
Description: Poisson
(data)
sample size = 3200
method = SIMPLE
ΔX = ——
H = 2^8
χ² = 124·2
D = 0·0723
24·00
30·00
36·00
42·00
48·00

frequency
0.00
50.00
100.00
150.00
200.00
250.00
0.00
6.00
12.00
18.00
gap

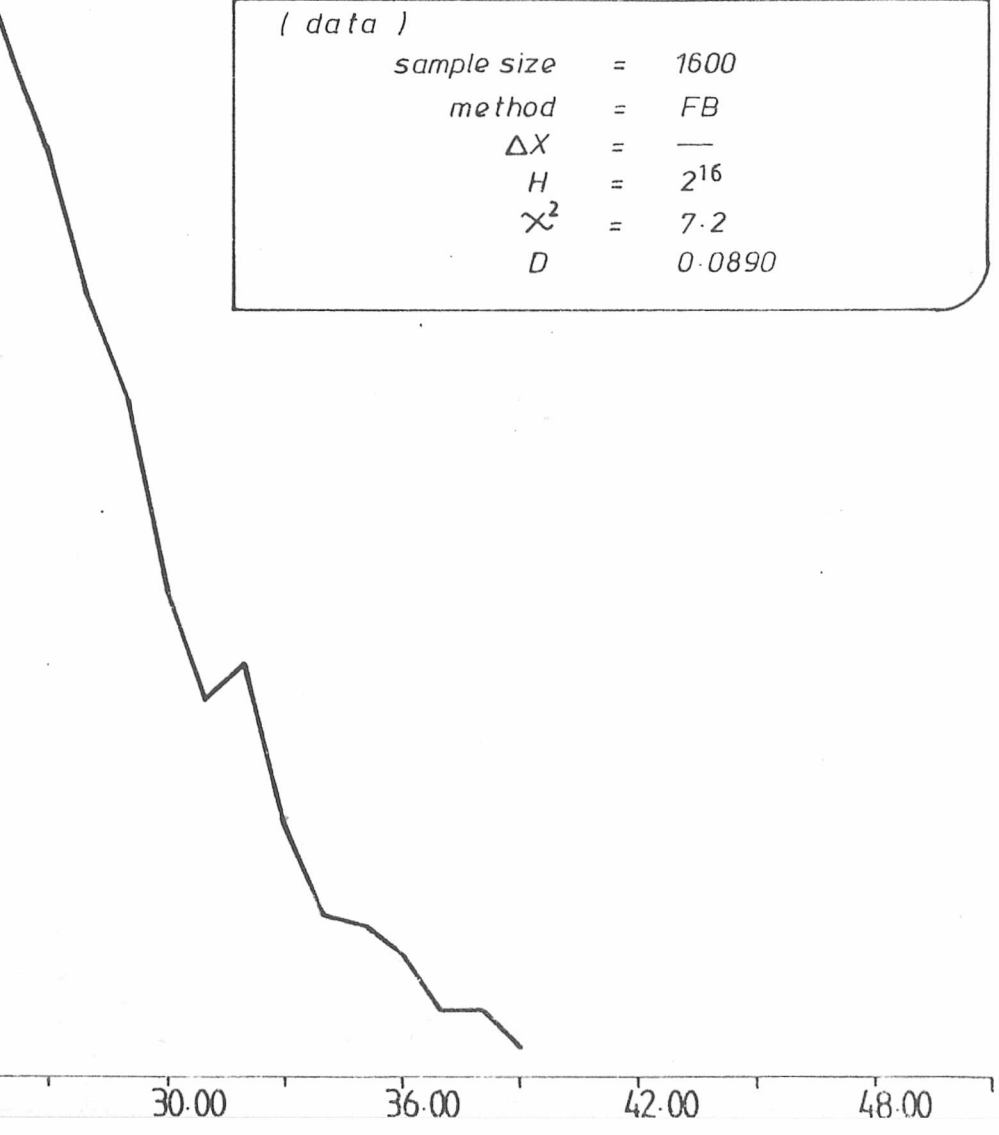

Graph No. A1·5c
Description: Poisson
(data)
sample size = 3200
method = FB
ΔX = —
H = 2^8
χ² = 63·4
D = 0·0881
24·00
30·00
36·00
42·00
48·00

frequency
0.00
25.00
50.00
75.00
100.00
125.00
0.00
6.00
12.00
18.00
24.00

Graph No.	A1·5d
Description:	Poisson

(data)

sample size	=	*1600*
method	=	*FB*
ΔX	=	—
H	=	2^{16}
χ^2	=	*7·2*
D		*0·0890*



POISSON distribution		No. of samples	No. of bits of random number	K-S test		$\chi^2_{n,\alpha}$ test	
method	figure			value result	critical value	test result	critical value
simple	4·5b	3200	8	0·0723	0·034	124·2	43·8
FB	4·5c	3200	8	0·0881	0·034	63·4	43·8
FB	4·5d	1600	16	0·0890	0·048	14·6	40·1

Table A1·5e Table of results and critical values ($\alpha = 0·95$)

APPENDIX A2

Control Signals Generated at State Transition

Appendix A3 presents the the necessary input signals to the micro-controler causing state transition. The output signals generated by the micro-controller to update the component model are presented here.

A * in the table of output signals indicates the generation of the particular control signal. The possible control signals are:

C_a -- Clear counter A.

I_a -- Increment counter A

I_b -- Increment counter B

I_c/int. -- Increment counter C and flag Component interrupt.

Clear -- Clear Policy Module request flags.

Transition to working state from:

	C_a	I_a	I_b	I_c/int.	Clear
working		*	*		
SUD maintenance	*				*
SUF repair	*				*
cold standby			*		

Transition to unrevealed fault state from:

	C_a	I_a	I_b	I_c/int.	Clear
cold standby			*		
unrevealed fault			*		

Transition to SUD state from:

	C_a	I_a	I_b	I_c/int.	Clear
SUD					
maintenance	*				*
repair	*				*
cold standby			*		

Transition to maintenance state from:

	C_a	I_a	I_b	I_c/int.	Clear
working	*		*		
unrevealed fault	*		*		
SUD	*				
maintenance					
SUF	*				
cold standby	*		*		
unattended failure	*		*		

Transition to SUF state from:

	C_a	I_a	I_b	I_c/int.	Clear
SUD					
maintenance	*				*
repair	*				*
cold standby			*		

Transition to repair state from:

	C_a	I_a	I_b	I_c/int.	Clear
working	*		*	*	
unrevealed fault	*		*		
SUF	*			*	
repair		O			
unattended failure			*		

Note O: I_a only if request allowed (R.A.)

Transition to cold standby state from:

	C_a	I_a	I_b	I_c/int.	Clear
working		*	*		
SUD					
maintenance					
SUF					
repair					*
cold standby		*	*		

Transition to unattended failure state from:

	C_a	I_a	I_b	I_c/int. Clear
repair		*		
unattended failure			*	

APPENDIX A3

State Transition Equations

Figure A3a presents all possible state transitions on a transition diagram.

7. Present state = working

		PRIORITY
possible next states:	undergoing repair	1
	maintenance	2
	cold standby	3
	working	4

[working→repair] transition requires RE.FA

There is no higher priority state

[working→maintenance] transition requires $MR.RA.(\overline{PASS}+PASS.\overline{FA})$

(Note: Passive components in operational state are not permited maintenance)

Hardware reductions reduce this to $MR.RA.(\overline{PASS}+\overline{FA})$

Working→maintenance AND (Not any higher priority state)

$=MR.RA.(\overline{PASS}+\overline{FA}).(\overline{RE}+\overline{FA})$

$=MR.RA.(\overline{PASS}.\overline{RE}+\overline{FA})$

[working→standby] transition requires FA

working→standby AND (Not any higher priority state)

$=\overline{FA}.(\overline{MR}+\overline{RA}+PASS.FA).(\overline{RE}+\overline{FA})$

$=\overline{FA}.(\overline{MR}+\overline{RA})$

working→working AND (Not any higher priority state)

$=(\overline{RE}+\overline{FA}).(\overline{MR}+\overline{RA}+FA.PASS).FA$

$=FA.\overline{RE}.(\overline{MR}+\overline{RA}+PASS)$

6. Present state = unrevealed fault

		PRIORITY
possible next states:	undergoing repair	1
	maintenance	2
	unrevealed fault	3

[unrevealed fault→repair] transition requires FA

There is no higher priority state

[unrevealed fault→maintenance] transition requires MR.RA

unrevealed fault→maintenance AND (Not repair)

$=MR.RA.\overline{FA}$

unrevealed fault→unrevealed fault AND (Not any higher priority state)

$=\overline{FA}.(\overline{MR}+\overline{RA})$

= start up delay (SUD)

5. Present state = start up delay (SUD)

		PRIORITY
possible next states:	maintenance	1
	cold standby	2
	SUF	3
	working	4
	SUD	5

[SUD→maintenance] transition requires $MR.RA.(\overline{PASS}+\overline{FA})$

There is no higher priority state

[SUD→standby] transition requires $\overline{FA}$

SUD→standby AND (Not any higher priority state)

$=\overline{FA}.(\overline{MR}+\overline{RA}+PASS.FA)$

$=\overline{FA}(\overline{MR}+\overline{RA})$

[SUD→start up failure risk (SUF)] transition requires RE.SUF

SUD→SUF AND (Not any higher priority state)

$=RE.SUF.FA.(\overline{MR}+\overline{RA}+PASS.FA)$

$=RE.SUF.FA(\overline{MR}+\overline{RA}+PASS)$

[SUD→working] transition requires RE

SUD→working AND(Not any higher priority state)

$=RE.(\overline{RE}+\overline{SUF}).FA.(\overline{MR}+\overline{RA}+PASS.FA)$

$=RE.\overline{SUF}.FA.(\overline{MR}+\overline{RA}+PASS)$

SUD→SUD AND (Not any higher priority state)

$=\overline{RE}.(\overline{RE}+\overline{SUF}).FA.(\overline{MR}+\overline{RA}+PASS.FA)$

$=\overline{RE}.FA.(\overline{MR}+\overline{RA}+PASS)$

4. Present state = scheduled maintenance

		PRIORITY
possible next states:	cold standby	1
	SUD	2
	SUF	3
	working order	4
	maintenance	5

[maintenance→standby] transition requires $RE.\overline{FA}$

There is no higher priority state

[maintenance→SUD] transition requires RE.SUD

maintenance→SUD AND (Not any higher priority state)

$=RE.SUD.(FA+\overline{RE})$

$=RE.SUD.FA$

[maintenance→SUF] transition requires RE.SUF

maintenance→SUF AND (Not any higher priority state)

$=RE.SUF.(FA+\overline{RE}).(\overline{RE}+\overline{SUD})$

$=RE.SUF.FA.\overline{SUD}$

[maintenance→working] transition requires RE

maintenance→working AND (Not any higher priority state)

$=RE.(\overline{RE}+\overline{SUD}).(\overline{RE}+\overline{SUF}).(FA+\overline{RE})$

$=RE.\overline{SUD}.\overline{SUF}.FA$

maintenance → maintenance AND (Not any higher priority state)

$=(\overline{RE}+\overline{SUD}).(\overline{RE}+\overline{SUD}).\overline{RE}.(FA+\overline{RE})$

$=\overline{RE}$

3. Present state = start up failure risk (SUF)

		PRIORITY
possible next states:	repair	1
	maintenance	2
	cold standby	3
	working	4

[SUF→repair] transition requires RE

There is no higher priority state

[SUF→maintenance] transition requires $MR.RA.(\overline{PASS}+\overline{FA})$

SUF→maintenance AND (Not any higher priority state)

$=MR.RA.(\overline{PASS}+\overline{FA}).\overline{RE}$

[SUF→cold standby] transition requires $\overline{FA}$

SUF→standby AND (Not any higher priority state)

$=\overline{FA}.\overline{RE}.(\overline{MR}+\overline{RA}+PASS.FA)$

$=\overline{FA}.\overline{RE}.(\overline{MR}+\overline{RA})$

SUF→working AND (Not any higher priority state)

$=FA.\overline{RE}.(\overline{MR}+\overline{RA}+PASS.FA)$

$=\overline{RE}.FA.(\overline{MR}+\overline{RA}+PASS)$

2. Present state = requiring repair

		PRIORITY
possible next states:	cold standby	1
	SUD	2
	SUF	3
	working	4
	repair	5
	unattended failure	6

[repair→standby] transition requires $RE.\overline{FA}$

There is no higher priority state

[repair→SUD] transition requires RE.SUD

repair→SUF AND (Not any higher priority state)

$=RE.SUD.(\overline{RE}+FA)$

$=RE.SUD.FA$

[repair→SUF] transition requires RE.SUF

repair→SUF AND (Not any higher priority state)

$=RE.SUF.(\overline{RE}+FA).(\overline{RE}+\overline{SUD})$

$=RE.SUF.FA.\overline{SUD}$

[repair→working] transition requires RE

repair→working AND (Not any higher priority state)

$RE.(\overline{RE}+FA).(\overline{RE}+\overline{SUD}).(\overline{RE}+\overline{SUF})$

$=RE.FA.\overline{SUD}.\overline{SUF}$

[repair→repair] transition requires RA

repair→repair AND (Not any higher priority state)

$=RA.(\overline{RE}+FA).(\overline{RE}+\overline{SUD}).(\overline{RE}+\overline{SUF}).\overline{RE}$

$=RA.\overline{RE}$

repair→unattended failure AND (Not any higher priority state)

$=\overline{RE}.\overline{RA}$

1. Present state = cold standby

		PRIORITY
possible next state:	unrevealed fault	1
	SUD	2
	SUF	3
	working	4
	maintenance	5
	standby	6

[standby→unrevealed fault] transition requires RE

There is no higher priority state

[standby→SUD] transition requires FA.SUD

Standby→SUD AND (Not any higher priority state)

$=FA.SUD.\overline{RE}$

[standby→SUF] transition requires FA.SUF

$=FA.SUF.(\overline{FA}+\overline{sud}).\overline{RE}$

$=FA.\overline{SUF}.\overline{SUD}.\overline{RE}$

[standby→working] transition requires FA

standby→working AND (Not any higher priority state)

$=FA.\overline{SUF}.\overline{SUD}.RE$

[standby→maintenance] transition requires $MR.RA.\overline{FA}$

standby→maintenance AND (Not any higher priority state)

$=MR.RA.\overline{FA}.(\overline{FA}+\overline{SUD}).(\overline{FA}+\overline{SUF}).\overline{FA}.\overline{RE}$

$=MR.RA.\overline{RE}.\overline{FA}$

standby→standby AND (Not any higher priority state)

$=(\overline{MR}+\overline{RA}+FA).\overline{FA}.(\overline{FA}+\overline{SUF}).(\overline{FA}+\overline{SUD}).\overline{RE}$

$=\overline{FA}.\overline{RE}.(\overline{MR}+\overline{RA})$

0. Present state = unattended failure

		PRIORITY
possible next state:	maintenance	1
	repair	2
	unattended failure	3

Note: hardware ensures a MR is not issued unless ack.R. = 0

This prevents a transition to maintenance when repair has been started but temporarily discontinued.

[unattended→maintenance] transition requires MR

There is no higher priority state

[unattended→repair] transition requires RA

unattended→repair AND (Not any higher priority state)

$=RA.\overline{MR}$

unattended→unattended AND (Not any higher priority state)

$=\overline{RA}.\overline{MR}$

$$\text{empirical mean} = \frac{1}{N} \sum_{i=1}^{N} E_i$$

$$\text{expected value} = \frac{2^8 - 1}{2} = 127.5$$

The results are shown in the first column of each table of results.

(ii) The variance of each sequence was calculated, a measure of the dispersion of the random variable

$$\sigma^2 = \sum_{i=1}^{N} (E_i - \hat{E}) \, p(x_i) \qquad , N = 2^8$$

Expected value [2], $\sigma^2 = \frac{(N^2 - 1)}{12} = \frac{2^{16} - 1}{12}$

$= 5461.25$

The results are shown in the second column of each table.

(iii) To test the generated probability distribution function of E_i, the chi-square (X^2) test [11, 12] was used to compare the observed frequency with the theoretical expected value. To carry out the test, the interval [0, 255] was sub-divided into sixtyfour groups giving sixtythree degrees of freedom for the test. Critical values of X^2 are given along with results in the third column of each table. The level of significance for the test, α, was chosen to be 0.05.

(iv) Independence tests were carried out on number pairs. Firstly number pairs,

$$(E_i, E_{i-1}), (E_i, E_{i-1}), \ldots (E_i, E_{i-1})$$

N = number of samples

$i = 2, 3, 4, \ldots N$

were chosen and compared to determine the quantity of consecutive numbers, (E_i, E_{i-1}) etc., following in the same group. For both independence tests applied, the interval [0,255] was subdivided into sixteen equal-sized groups. This meant that the expected number of observations in each group was $N/(16 \times 16)$. The second independence test was for number pairs,

$$(E_i, E_i), (E_i, E_i), \ldots (E_i, E_i)$$

Once again the numbers were observed for consecutiveness, that is both lying in the same group. The test was only carried out when the decimated number sequence technique was used. The X^2 test has been used to determine the goodness of the results. The X^2 values obtained and critical values for the test, $X^2_{15,0.05}$, are given in each table of results.

(v) The Kolmogorov-Smirnov test [11] was used to compare the experimental cumulative distribution function observed for E_i with the expected distribution. The value D, given in the sixth column in each table, is the difference observed with greatest absolute magnitude. Critical values for D, with $\alpha = 0.05$, are also given.

(vi) A runs test [11] was employed to determine if there were long runs of large or small numbers. It is considered to be a very discriminating test, that is it 'fails' more sequences of random numbers than other tests. Values obtained for the test and critical values obtained from tables of the normal distribution ($\alpha = 0.05$) are given on the last column of each table of results.

4 COMPARISON OF TEST RESULTS

The performance of number generators based on the following techniques has been considered :

(i) The multiple m-sequence method, described in Section 2.1. A circuit diagram of the generator built to test this method is shown in Figure 1, and the experimental results obtained are given in Table 1.

These indicate that the sequences produced possessed relatively poor independence characteristics. Only 38% of the sequences had values in excess of the critical value for the number-pairs tested and 13% failed the test for consecutiveness. On the other hand, sequences produced by this method are proved to have good distribution characteristics, 56% passing all tests.

(ii) The combination of two-m-sequences technique using as described in Sections 2.2 and 2.3. The generator used to test the method is shown in Figure 4. No decimation of the random number sequences was performed. However the generator was run sixteen times to produce the necessary output sequences. The results are shown on Table 2.

The sequences produced performed averagely well in both those tests examining their distribution characteristic and their independence characteristic. Only 13% failed to pass the former and 19% failed to meet the critical test values. Overall, 63% of the sequences passed all tests.

(iii) Shifted m-sequence techniques were investigated by testing three different generators. All of the generators employed the phase shifting techniques described in Section 2.5. The stages required to be modulo-2 added to produce the phase shifts are given in Figure 10. Only one, shown in Figure 9, has the

approx. phase shift	feedback from points 18 and 31 required MOD-2 addition stages
1×2^{27}	2, 8, 9, 18, 19, 28
2×2^{27}	4, 5, 16, 25
4×2^{27}	8, 10, 19
8×2^{27}	7, 16

approx. phase shift	feedback from points 24 and 31 required MOD-2 addition stages
1×2^{27}	2, 4, 16
2×2^{27}	2, 5
4×2^{27}	1, 7
1×2^{22}	1, 3, 4
2×2^{22}	1, 5, 7
4×2^{22}	1, 9, 13
8×2^{22}	1, 16

Figure 10 Additions required to produce phase shifts

proposed technique implemented to produce sixteen synchronous streams of random numbers. The results of the statistical tests on this generator are shown in Table 5. The other two generators, of the type shown in Figure 7, were decimated to produce sixteen sequences. The investigations concluded that decimation was an unreliable method of producing several number sequences, but it may prove interesting to observe the effects (if any) on the tests. The results obtained are given in Tables 3 and 4. It should be noted that the only difference between these two generators is in the choice of m-sequence used.

Of the techniques using shifted m-sequences, the proposed technique based on the use of preconditioned sequences yields consistently good results. Although two sequences failed the Chi and Kolmogorov-Smirnov distribution tests and four failed to pass the independence test, the majority, eleven, passed all tests successfully.

A summary of the results is given on Table 6. For each method tested the number of sequences which failed a particular test is indicated. Finally the number of sequences which passed all tests has been calculated for each method.

5 CONCLUSION

Various techniques for producing several statistically independent streams of random numbers have been investigated.

The use of multiple m-sequences was considered. This technique can be easily implemented, and the n-bit numbers produced by a single generator whose output sequence is decimated have the same statistical quality as the original single number stream. Cross-correlation characteristics between different m-sequences rule out this generator as a highly independent number source.

The use of a two-m-sequence combination was considered for two quite different implementations. Modulo-2 addition of m-sequences offers a economy of hardware although care has to be taken to ensure that an adequate phase shift exists between output bit sequences. The use of cascaded shift registers eliminates the phase shift problem but at the cost of additional hardware. Investigation of the auto-correlation function of an n-bit number produced by this technique indicated that the numbers are not of high statistical quality. In addition, proper decimation of such a number sequence may result in significant correlation between the output number sequences.

Finally the principle of producing n output sequences from a single m-sequence was considered. This may be achieved by decomposed register techniques. Considerable effort is involved in ensuring that the n sequences produced are widely spaced. Modulo-2 addition of selected stages of a feedback shift register also yields phase shifted sequences. The necessary combination of stages is more simply calculated. Proper decimation of a single number sequence produced by this technique may result in cross-correlation between the output number sequences.

A new pseudo-random number generator is proposed. Each output number sequence is simultaneously generated, making the technique well suited to high speed problems. The technique relies on phase-shifting pre-conditioned sequences, and lends itself to a MSI circuitry implementation. Statistical tests performed on the generator confirm the high statistical quality of the sequences produced vis-a-vis alternative techniques. Of the sequences produced by this new method, 69% passed all the statistical tests. Although

a similar pass rate was achieved using the circuit configuration illustrated in Figure 7, the results given in Table 4 indicate that this technique cannot be regarded as being reliable.

6 ACKNOWLEDGEMENTS

The support of the Science and Engineering Research Council, the National Centre of Systems Reliability, UKAEA and the assistance of Mr B Davidson and Miss K Craighead are acknowledged.

7 REFERENCES

[1] Golomb S W, 'Shift Register Sequences", Holden-Day Inc. 1967

[2] Hoffman de Visme G, 'Binary Sequences', English University Press Ltd 1971

[3] Birolini A, 'Hardware Simulation of Semi-Markov and Related Process', Mathematics and Computers in Simulation XIX (1977) 75-97 North-Holand Publishing Company

[4] Maritsas D G, Correspondence 'On the Statistical Properties of a Class of Linear Product Feedback Shift-Register Sequences IEEE Transactions on Computers Oct. 1973 pp 961-2

[5] Hartley M G, 'Development, Design and Test Procedure for Random Generators using Chaincodes', Proc. IEE Vol 116 No. 1 1969 pp 22-34

[6] Schwind M, 'On Generating and Applicating a set of Independent Bernoulli-Sequences' Proceedings of 1st International Symposium on Stochastic Computing and its applications 1978 Toulouse France pp 103-12 Computing and its Applications 1978 Toulouse France

[7] Hurd W J, 'Efficient Generation of Statistically Good Pseudonoise by Linearly Interconnected Shift Registers' IEEE Transactions on Computers 1974 pp 146-52

[8] Maritsas D G, Arvillias A C, Bounas A C, 'Phase Shift Analysis of Linear Feedback Shift Register Structures Generating Pseudorandom Sequences' IEEE Transactions on Computers c-27 No. 7 1978 pp 660-69

[9] Mars P, Miller A J, 'Theory and Design of a Digital Stochastic Computer Random Number Generator' Mathematics and Computers in Simulation XIX 1977 pp 198-216 North-Holland Publishing Company

[10] Latawiec K J, Correspondence 'New Methods of Generation of Shifted Linear Pseudorandom Binary Sequences' Proc. IEE Vol. 121 No. 8 1974 pp 905-6

[11] Siegel S, 'Nonparametic Statistics for Behavioural Science' New York, McGraw Hill Ltd. 1956

[12] Maisel H, Gnugnoli G, 'Simulation of Discrete Stochastic Systems', Kingsport Press Ltd. 1972

[13] Maritsas D G, 'The Autocorrelation Function of Two Feedback Shift-Register Pseudorandom Source', IEEE Transactions on Computers 1973 pp 962-64

[14] Green D H, 'Nonlinear Product-Feedback Shift Registers' Proc. IEE Vol. 177 No. 4 1970 pp 681-86

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	TEST2	K-S	RUNS
1	126·4	5441·2	91·3	12·2	12·2	0·011	-0·06
2	126·7	5555·3	56·4	19·3	10·7	0·013	-0·22
3	126·6	5407·1	64·7	26·8	14·7	0·013	1·06
4	125·7	5529·2	64·2	17·2	12·4	0·016	0·52
5	125·5	5476·8	115·6	56·4	18·0	0·015	0·98
6	127·2	5397·8	55·6	28·5	25·7	0·012	0·77
7	126·1	5585·8	68·1	25·9	30·7	0·015	0·14
8	128·7	5505·1	64·6	12·3	17·1	0·014	0·48
9	127·9	5345·3	58·0	25·2	21·0	0·015	0·66
10	127·8	5482·0	64·0	28·0	24·0	0·009	1·77
11	127·0	5388·4	66·6	11·7	15·5	0·009	-0·19
12	128·4	5544·6	50·3	9·4	15·2	0·012	0·04
13	126·9	5433·2	61·1	20·0	11·2	0·008	-0·13
14	128·1	5457·4	44·9	15·9	18·7	0·012	-0·89
15	125·9	5383·2	53·3	19·6	16·1	0·017	1·02
16	127·6	5602·2	53·5	12·9	17·7	0·011	-0·38
	(127·5	5400)	(82·5	24·9	24·9	0·030	±1·96)

expected values

critical values ($\alpha = 0.05$)

Table 1 Multiple m - sequence

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	K-S	RUNS
1	128·5	5579·3	58·8	8·8	0·010	-0·91
2	125·7	5471·4	84·9	14·7	0·016	-0·39
3	127·4	5470·2	63·3	11·0	0·012	0·49
4	128·0	5486·5	82·8	73·4	0·009	1·36
5	127·4	5480·4	73·4	7·9	0·009	-0·73
6	127·7	5436·9	57·1	17·3	0·005	1·05
7	126·6	5388·9	70·2	11·5	0·012	-0·41
8	128·5	5548·2	79·7	15·7	0·013	-1·28
9	127·2	5453·8	54·0	14·5	0·010	0·85
10	127·4	5479·6	29·2	16·8	0·004	1·14
11	126·8	5376·9	78·5	32·8	0·011	-0·43
12	127·5	5553·1	48·2	26·3	0·011	-0·09
13	127·5	5411·6	71·9	15·1	0·012	2·30
14	129·4	5435·2	65·0	11·7	0·017	2·60
15	128·7	5494·5	72·4	22·0	0·011	-0·85
16	128·6	5506·4	61·7	18·1	0·012	-0·81
	(127·5	5400)	(82·5	24·9	0·030	±1·76)

expected values

critical values ($\alpha = 0.05$)

Table 2 Combination of two m - sequences

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	TEST2	K-S	RUNS
1	125.5	5465.5	71.3	19.7	19.7	0.019	0.50
2	128.4	5491.0	49.2	18.4	19.4	0.011	-0.32
3	126.8	5518.6	57.4	19.9	16.6	0.011	-1.26
4	127.1	5527.2	53.3	11.4	16.7	0.010	-0.92
5	127.2	5450.8	44.3	25.8	4.8	0.006	-0.57
6	128.2	5419.1	56.0	14.6	21.4	0.007	0.19
7	128.2	5394.2	64.3	15.3	10.3	0.010	-0.84
8	129.4	5489.7	49.2	18.8	14.9	0.016	-0.03
9	127.2	5559.0	55.4	12.4	16.4	0.011	-0.98
10	127.3	5466.4	43.8	8.3	17.1	0.007	-0.53
11	128.9	5433.9	66.8	20.9	16.2	0.022	-1.82
12	128.2	5380.9	52.9	13.4	11.2	0.011	0.39
13	126.0	5479.5	86.0	17.1	20.8	0.016	-1.77
14	127.5	5501.9	65.2	20.8	5.2	0.006	1.55
15	128.8	5535.8	58.8	22.3	14.0	0.014	3.08
16	128.0	5374.4	56.7	28.0	23.6	0.010	-0.36
	(127.5	5400)	(82.5	24.9	24.9	0.030	±1.76)

expected values

critical values ($\alpha = 0.05$)

Table 3 Shifted m-sequence technique

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	TEST2	K-S	RUNS
1	126.4	5519.7	55.6	26.3	26.3	0.012	-0.060
2	127.0	5416.6	66.3	8.9	14.6	0.009	-1.26
3	128.0	5510.4	69.6	9.7	24.4	0.010	0.22
4	127.1	5639.0	90.6	25.2	31.6	0.014	0.70
5	126.3	5590.1	47.0	33.3	6.2	0.012	0.26
6	126.4	5528.3	93.5	17.2	20.3	0.014	-0.50
7	126.1	5362.0	50.0	21.3	11.2	0.015	0.87
8	128.3	5429.9	65.6	30.1	52.8	0.009	-1.14
9	128.0	5420.5	38.1	14.8	24.8	0.008	-0.92
10	126.3	5599.4	93.5	14.4	17.5	0.016	-0.54
11	126.8	5399.0	46.7	18.6	13.3	0.011	-1.14
12	127.9	5570.6	106.8	20.2	21.3	0.013	-0.22
13	127.4	5395.7	52.7	7.6	26.6	0.006	-0.25
14	128.2	5536.4	175.9	40.1	16.0	0.014	-1.32
15	127.7	5336.1	94.0	17.8	8.7	0.010	1.14
16	126.2	5534.0	69.3	9.8	18.9	0.017	0.59
	(127.5	5400)	(82.5	24.9	24.9	0.030	±1.76)

expected values

critical values ($\alpha = 0.05$)

Table 4 Shifted m-sequence technique

No. of elements in each sequence = 4000

SEQ	MEAN	VAR	CHI	TEST1	K-S	RUNS
1	128·3	5390·4	67·4	21·0	0·010	-0·79
2	126·5	5387·9	45·7	22·2	0·012	-0·60
3	128·2	5485·7	52·7	25·3	0·009	0·62
4	126·3	5410·8	37·5	23·6	0·012	0·02
5	126·2	5627·7	60·8	8·8	0·015	2·44
6	127·1	5413·8	61·0	28·6	0·011	-2·33
7	127·0	5421·7	42·8	14·0	0·009	-0·28
8	128·8	5294·5	65·8	24·8	0·018	-1·57
9	128·7	5481·0	50·1	10·0	0·012	-0·19
10	125·8	5460·8	66·0	14·8	0·018	0·48
11	126·4	5539·1	54·0	7·5	0·012	-0·31
12	125·2	5372·2	63·0	19·6	0·018	-1·25
13	124·5	5426·5	84·1	28·4	0·026	0·56
14	121·2	5432·9	105·1	46·2	0·041	0·76
15	126·9	5412·9	78·2	13·6	0·012	-0·71
16	122·7	5445·7	74·8	17·9	0·028	0·55
	(127·5	5400)	(82·5	24·9	0·030	±1·76)

expected value

critical value (α = 0·05)

Table 5 Proposed method

	multiple m-seq.	2-m seq.	shifted m-seq.		proposed method
Table	1	2	3	4	5
CHI - test	2	2	1	6	2
A Test 1	6	3	2	5	3
Test 2	2	-	1	4	-
K-S test	0	0	0	0	1
Runs test	0	2	3	0	2
number of seq. which passed all tests	9	10	11	6	11

In section A each box contains a number of sequences failing the particular test (α = 0·05)

Table 6 Analysis of results

THE DEVELOPMENT OF A NEW HARDWARE RELIABILITY SIMULATOR

N. D. Deans and D. P. Mann
Robert Gordon's Institute of Technology, Aberdeen

A new concept, based on the use of random signals, is proposed for a computing instrument for simulating the reliability aspects of a system. Random signals interact with deterministic signals. A Simulator has been designed and constructed using large-scale integrated circuits. A high degree of parallelism has been incorporated in the design, and the use of asynchronous time-scaling techniques have yielded operational speeds of the order of 10^8 computations per second. Experiments have been carried out on unreliable systems using the Simulator and typically, system lifetimes of 10^7 hours are simulated by the equipment in approximately one second.

INTRODUCTION

Two basic approaches to the calculation of the reliability of a system exist, viz :

1 Analytical methods in which the solutions to the equations describing a mathematical model of the system are sought. However, the investigation of systems whose behaviour cannot be accurately modelled by exponential distributions and/or which contain complex repair and maintenance policies presents difficulty.

2 Simulation techniques in which sampling experiments are carried out on the model. For acceptable accuracy of solution, such an analysis carried out on a digital computer with a von-Neumann architecture necessitates long computational times.

A new stochastic reliability simulator based on Monte-Carlo simulation techniques has been developed to enable the reliability of complex systems to be studied. The technique relies upon the generation of random variables with known statistical distributions to describe particular properties of the components which make up the system. By allowing these properties to 'interact' in accordance with a mathematical model describing the overall system, observations on the overall system reliability can be made.

The generation of number distributions with prescribed properties is described by Birolini (1) and can be easily realised with medium or large-scale integrated circuits. In addition the processing of random number sequences can be carried out at high speed using special purpose digital circuitry. By allowing the modelling of each component to proceed simultaneously with all other components, a high degree of parallelism can be achieved.

A block diagram of the Simulator is shown in Figure 1. It consists of a number of subsystems acting collectively as a special-purpose peripheral device to a host computer. In operation, the host computer initialises the Simulator specifying in particular :

a the failure distribution of each component

b the repair distribution of each component

c the component interconnection pattern

d details of age or block replacement policies for each component

e details of any start-up delays associated with any component

f information relating to any high start-up failure risk

g details of the time span over which the study is to be made and the time resolution factors to be used throughout the simulation during the time that the component is operational and during the time it is being maintained or repaired.

h details of policy relating to resources etc.

The host computer then relinquishes control of the Simulator and the latter proceeds to act autonomously. Individual components operate in a united fashion, each processing random number distributions in conjunction with the prescribed failure and repair distributions to determine its operational condition. During the simulation period, interactions between components may take place. In particular, requests for maintenance, repair etc are dealt with by the Policy Module. The Statistical Gathering Module is designed to collect statistics relating to the behaviour of the overall system model or to the performance of specific sub-units within the model; it acts in effect as a 'statistical probe'.

Subsequent to a simulation run, the host computer re-engages the Simulator and transfers into its main memory, data relating to the performance of the model and its component parts. A suite of programs within the host computer process this data and present it to the experimenter as tabulated numerical information or directly in graphical form on cathode-ray displays.

TIME QUANTISATION

The components which comprise the overall system are continually undergoing changes of state. At any time, a component may fail and fall into a 'non-operating' state; some time later it will be repaired and move back into an 'operating' state. Mathematically, this changing sequence is described by the changing renewal process, in which there are embedded renewal processes. The random time intervals used in the renewal processes are generated using pseudo-random number generators described by Deans and Mann (2). A random time interval T_n is generated in which T_n is a multiple of a quantum time value ΔT. The ΔT values used by different components will vary, and all ΔT values are considered to be integer multiples of an absolute minimum quantisation value $\Delta T(min)$. The updating of the modelling process corresponds to incrementing the renewal process. A component is only considered for updating when a basic counter keeping account of the time into the simulation has been incremented by an amount equal to its ΔT value. Thus a component with a time quantisation value of $\Delta T = 8\ \Delta T(min)$ would have its modelling process updated twice as often as the component for whom $\Delta T = 4\ \Delta T(min)$.

A COMPONENT MODULE

Each Component Module represents a particular aspect of the actual system being modelled. In its simplest form, a component may represent an item of equipment such as a generator or a relay. More abstract system aspects such as computer software or human behaviour can also be represented, and the design of the Component Module utilises microprogramming techniques to yield the degree of flexibility required. A block diagram of the Module is shown in Figure 2.

The age and block replacement times are held in registers Ta and Tb respectively. Their contents are continuously compared with the current values held in counters Ca and Cb. The former records the age of the component, the latter the time elapsed since the last block replacement operation. When equality of either comparison occurs, a signal is sent to a microcontroller which then takes the appropriate action.

A Status Register is used to maintain a bit pattern indicative of the state of the component. At any time the component may be in any one of the following states :

In working order

Undergoing repair

Undergoing scheduled maintenance

In a failed state - unattended

On cold standby

Possessing an unrevealed fault condition

Unavailable during starting delay

Suffering additional risk during start-up

In addition, the Status Register contains the following flag bits :

(i) A 'repair request' flag, set immediately a component fault is detected.

(ii) A 'maintenance request' flag that is set on reaching a scheduled maintenance time

(iii) An 'acknowledge request' flag indicating that the request for assistance has been recorded by the Simulator and that global resources are being released to satisfy the request.

A Mask Register and a Micro-instruction Control Unit contained within each Component Module allow a considerable amount of decision-making to be devolved to component level. The decision-making policy is stored as a series of controls in a 1024 × 8-bit programmable read-only-memory. In operation, requests by the component for attention are acted on by the microcontroller only after it carries out an examination of the state of the component in conjunction with the bit pattern held in the Mask Register. By setting different bit patterns in this register, particular features of component behaviour can be enabled or disabled.

Each component has the ability to model two non-exponentially distributed state transition times. This is achieved by storing the repair and maintenance distributions in read-write memories, these being written to by the host computer during the initialising phase and read from during the simulation phase.

CONTROL MODULE

Following the initialisation phase by the host computer, the Control Module takes control of the internal data buses of the Simulator. Its prime function is to determine the amount by which the system simulation is to be advanced in time. It achieves this by investigating each Component Module's time quantisation value (Δt) ie the time resolution on which that Component Module is currently operating. The Control Module selects the smallest ΔT value found, and advances the simulation in time by that amount. Thus the simulation proceeds by leaping forward in time eg in steps of days when nothing significant is happening, but advancing slowly eg in steps of minutes while a component is undergoing repair.

STATISTICAL GATHERING MODULE

During a simulation run, information relating to the number of times prescribed events occur, and their durations, is recorded in the Statistical Gathering Module. It consists of a number of binary counters each of which can be programmed by the host computer, during the initialisation phase, to record the occurence of specified events eg down time of a particular component or group of components, time to first system failure, number of maintenance requests.

At the conclusion of a simulation run, the host computer regains access to the various data buses within the Simulator and is able to access and read data from the counter recorders of the Statistical Gathering Module.

REPAIR POLICY MODULE

The behaviour and state of any component in a system is inter-related to that of each of the other constituent components in that system especially if limited maintenance and repair resources are available. The Module is programmed by the host computer during the initialisation phase with data relating to the character and extent of the resources available for use. Throughout a simulation run, components issue requests for repair and maintenance support, and it is the responsibility of the Repair Policy Module to accede to these requests within the bounds of the available resources.

EXPERIMENTAL PROCEDURE

During the initialisation phase, the experimenter defines the component and system specification by responding to a series of questions displayed on a terminal. In particular the following parameters are required for each component :

(i) the time quantisation values to be used

(ii) whether an age replacement policy is to be used

(iii) whether a block replacement policy is to be used

(iv) the age and block replacement times

(v) whether or not the component has a high start-up failure probability

(vi) whether or not the component has a start-up delay associated with it

(vii) the repair distribution

(viii) the failure distribution

(viiii) the initial age of the component

The configuration of the components which comprise the system must also be specified. A program within the host computer allows the experimenter to enter the topology of the system being studied on an interactive graphics terminal. The position of the components together with their connective relationships are entered as a success tree and the points in the network at which the Statistical Gathering Module will collect data are defined.

Finally, the time over which the behaviour of the system is to be observed, is specified.

EXPERIMENTAL RESULTS

To verify the operation of the Simulator, a selection of system arrangements have been studied. As an illustration of the Simulator's performance and asynchronous operation, the results of a two-component series system are presented.

The components are identical, each having a failure distribution described by an exponential function of mean $1/\theta$. Global maintenance resources are sufficient to ensure that a policy of simultaneous block replacement of components is successfully carried out at intervals of Tb. With no repair permitted, the theoretical mean unavailability, μ_D, for the system is given by :

$$\mu_D = 1 - \frac{1}{2\theta Tb}\left(1 - e^{-2\theta Tb}\right)$$

A block replacement time of 1000 hours was selected and $1/\theta$ chosen to be 1000 hours. The parameter μ_D is then equal to 0.568. The minimum time quantisation $\Delta T(min)$ was selected to be 1 hour and the failure distribution time quantisation (TQ) was varied from 1 hour to 64 hours. The results for the mean unavailability and the simulation time necessary to model 10^6 hours of operation are shown on Figure 3. High TQ values are shown to produce short simulation times. However,

very high TQ values are not rewarded with significant speed gains, as the probability of a component existing in a state employing such a TQ value reduces.

The example system described utilises little of the modelling features available. To demonstrate a selection of these features, consider the five-component system shown in Figure 4. The component failure and repair distribution parameters are shown in Figure 5 along with other component features. The system success tree and probe positions at which information is to be gathered are shown in Figure 6 in the format directly acceptable to the Simulator.

The sensitivity of system reliability to individual component failure distribution parameters can be quickly determined. The distribution parameter λ has been varied from its normal value λ_n through the range $0.5\ \lambda_n < \lambda < 3.0\ \lambda_n$ for each component in turn, all other components remaining unchanged. The system mean unavailability u_D has been determined by probe P1 for system life times of 10^6 hours. Each simulation lasted 0.9 seconds and the results are shown on Figure 7.

The system as observed by probe P3 has been investigated for an age replacement policy. Component 1 is removed because its exponential failure distribution results in an impaired performance when maintenance is applied alongside repair. The four-component system can then be modelled for life times of 10^6 hours in 0.88 seconds. Only components 2, 3 and 4 took part in the maintenance programme. The cold standby unit, component 5 underwent normal repair at failure. The system repair and maintenance resources were such that all requests made by components could be allowed. That is, three 'men' were available to carry out the repair and maintenance policy. Repair at failure for the components undergoing maintenance was considered to take two possible forms. Firstly repair resulted in a component being as good as new (R G N policy). Secondly, repair was minimal, that is the repair did not return the failure hazard function to its starting value. In this case, the new component condition achieved by maintenance was better than the repaired condition (N B R policy). The time required to carry out maintenance on any component was 1 hour, and the age at which maintenance was applied was varied over a wide range of values. The results for system u_D are given in Figure 8, and for the mean-time-between-system-failures on Figure 9.

The system has further been investigated for a block replacement policy. Components 3 and 4 are scheduled for simultaneous maintenance at intervals of Tb. Component 2 also undergoes maintenance at intervals of Tb but starts with an initial time shift of k Tb, $0 < k < 1$. The maintenance times are exponentially distributed with mean value of 10 hours, and repair is R G N. The maintenance and repair resources can be either 'one man', 'two men' or 'three men', applied according to two resource policies viz Policy 1 or Policy 2. These policies assign a priority order to component requests for use of the shared resources. Of highest priority is component 2 followed by component 3 then 4. The policies differ in that Policy 1 does not permit a low priority request to be discontinued by the arrival of a higher priority request after the low priority task has commenced.

With Tb = 1000 hours, the system u_D has been determined for a range of k values for Policy 1. The results for varied numbers of repair 'men' are given on Figure 10. Tb was then changed to 300 hours. The system u_D was determined for a 'one man' and 'three men' case under Policies 1 and 2. The results are shown on Figure 11. With T_B = 300 hours maintenance is applied too frequently with respect to optimal u_D. The 'one man' policy achieves a better system availability than the 'three men' one because many component maintenance requests cannot be immediately responded to, and are delayed for later consideration.

CONCLUSIONS

A design, based on the use of random signals is proposed for simulating the reliability aspects of a system. Random signals interact with deterministic signals representing features of a system in a stochastic digital processor. A Simulator has been designed and constructed using L S I devices. A high degree of parallelism has been incorporated in the design, and the use of asynchronous time-scaling techniques allows system lifetimes of 10^7 hours to be simulated in

approximately one second. The control logic of the Simulator is implemented using microprogramming techniques, enabling complex system features to be defined and modelled.

The Simulator has been designed to be general-purpose and has wide application.

REFERENCES

1 Birolini, A., 1977, Mathematics and Computers in Simulation, XIX, 75

2 Deans, N. D., and Mann, D. P., Mathematics and Computers in Simulation (To be published)

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support of the Science and Engineering Research Council and the National Centre for Systems Reliability in this work. In addition, the assistance of Mr B Davidson and Miss K Craighead in the preparation of this manuscript is acknowledged.

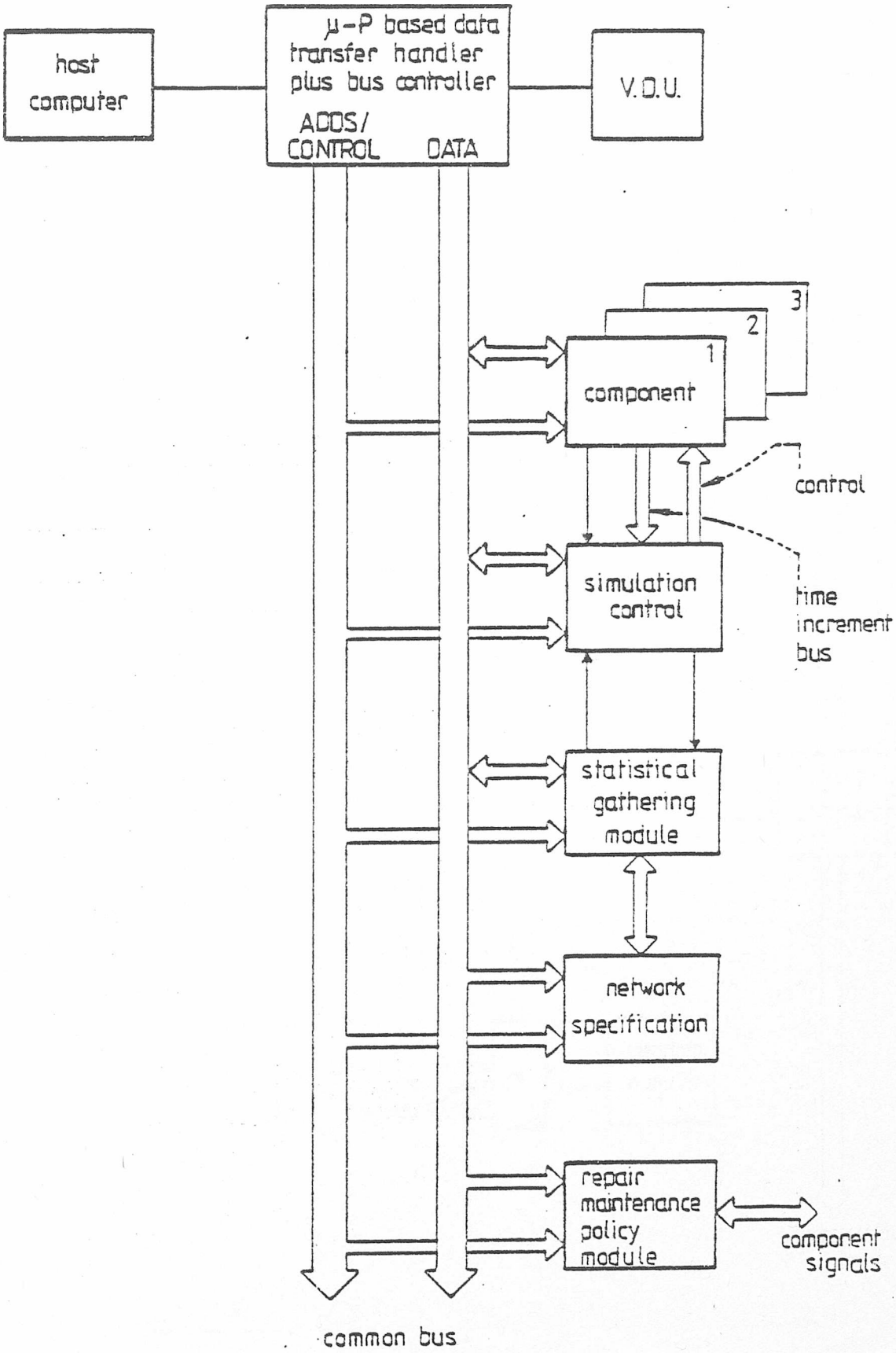


Figure 1. Block diagram of Simulator

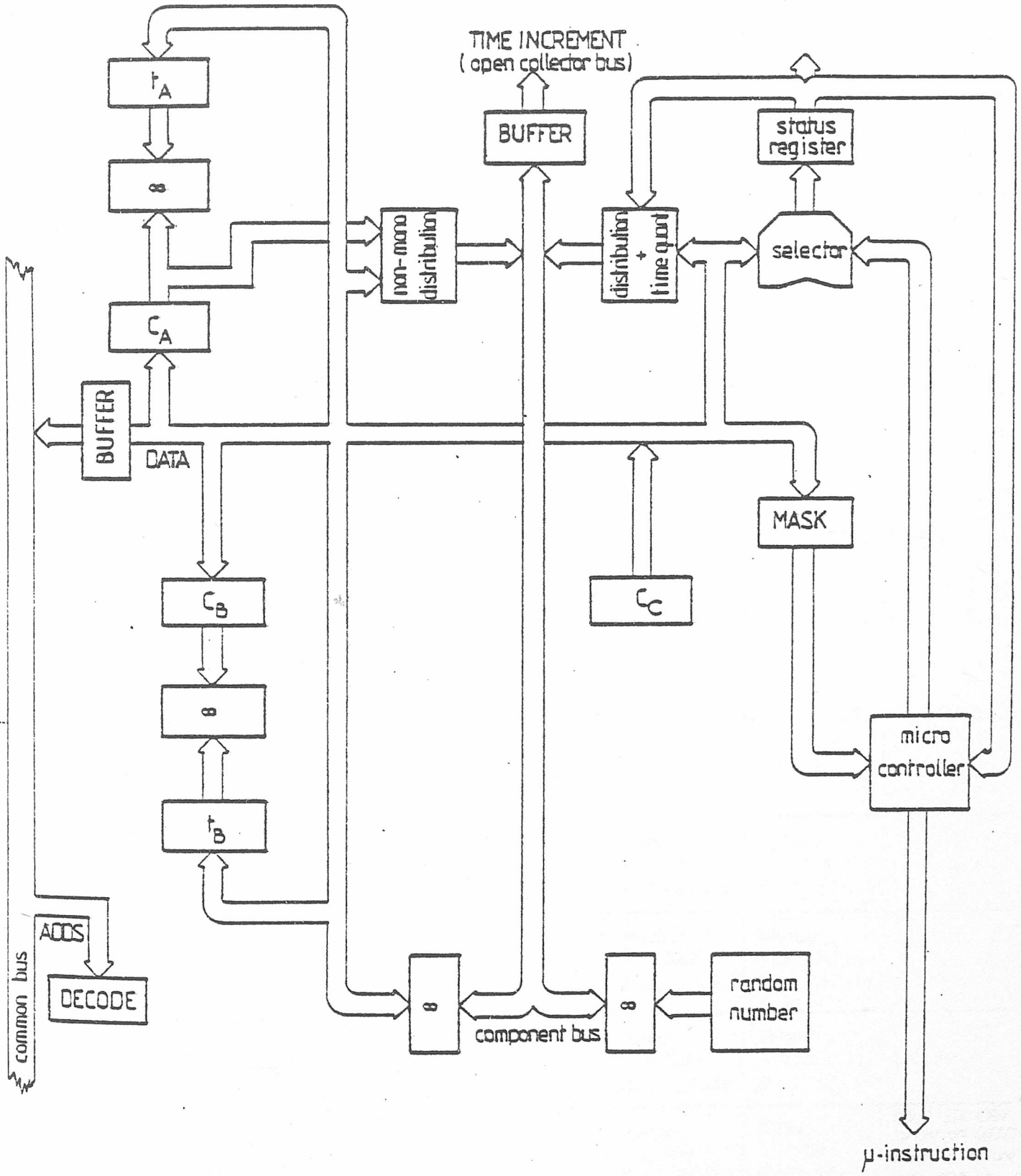


Figure 2 Block diagram of a Component Module

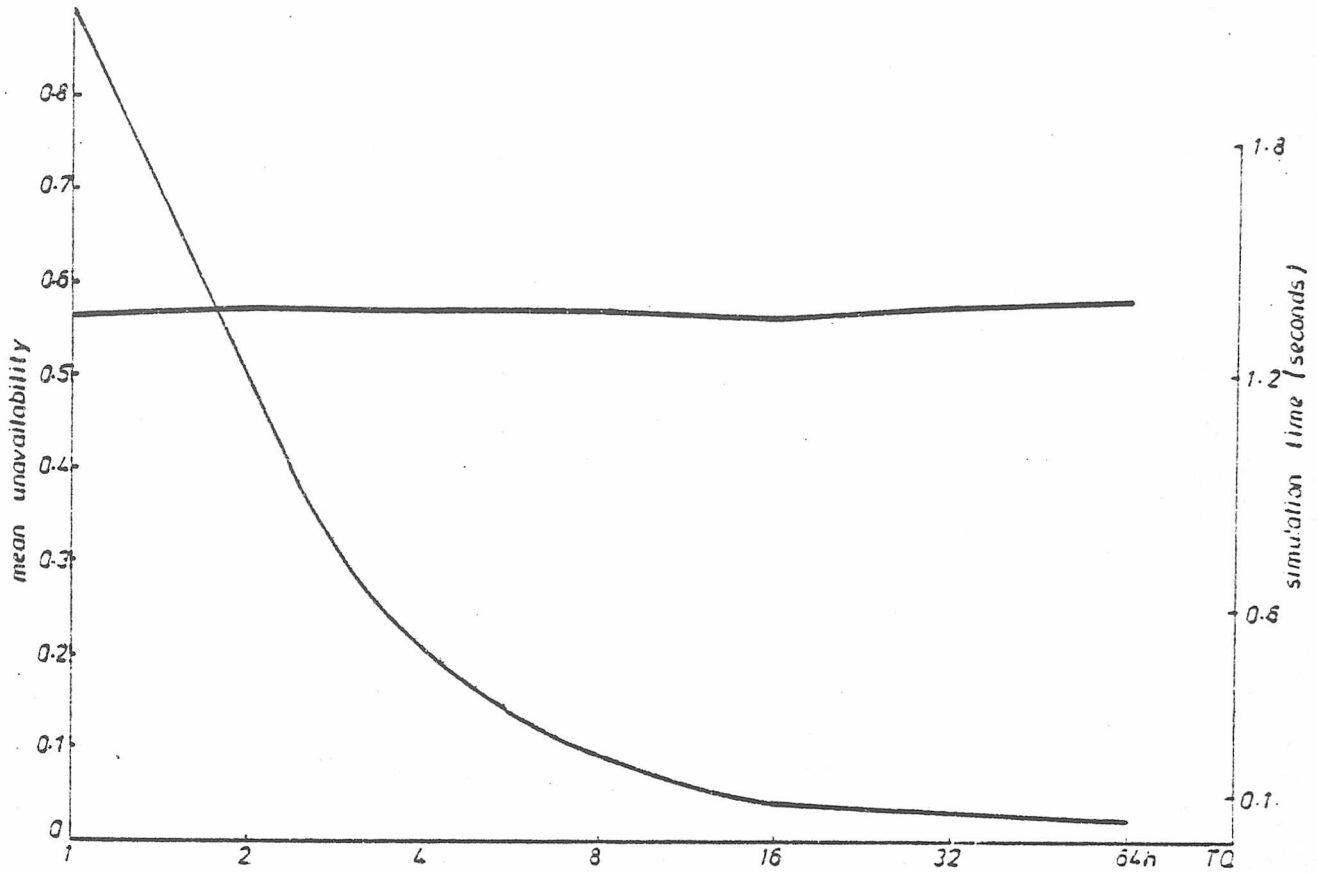


Figure 3 Two components in series Maintenance at Tb = 1000 hours

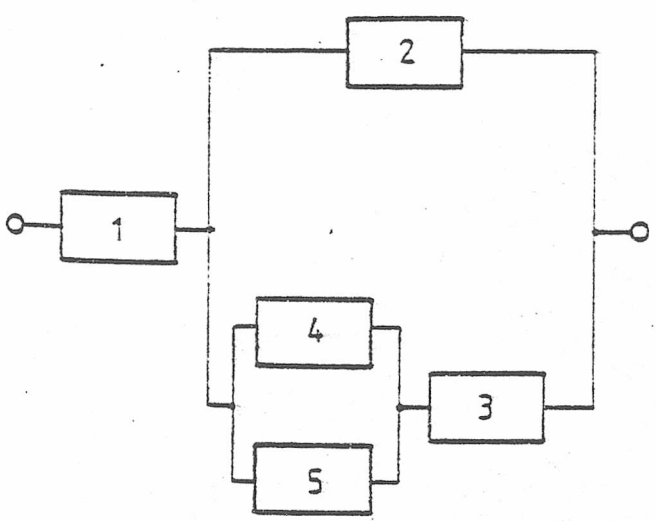


Figure 4 Five-component system

Component	Failure distribution	Repair distribution	Features
1	exponential λ = 2000h ΔT = 4h	exponential λ = 15h ΔT = 1h	—
2	Weibull λ = 500h β = 2, ΔT = 8h	Weibull λ = 40h β = 2, ΔT = 1h	—
3	Weibull λ = 500h β = 2, ΔT = 8h	Weibull λ = 15h β = 2, ΔT = 1h	—
4	Weibull λ = 250h β = 2, ΔT = 4h	Erlang λ = 10h β = 2, ΔT = 1h	—
5	Weibull λ = 250h β = 2, ΔT = 4h	Erlang λ = 10h β = 2, ΔT = 1h	COLD STANDBY failure rate 1/1000h start up failure risk 10%

Figure 5 Component Specifications

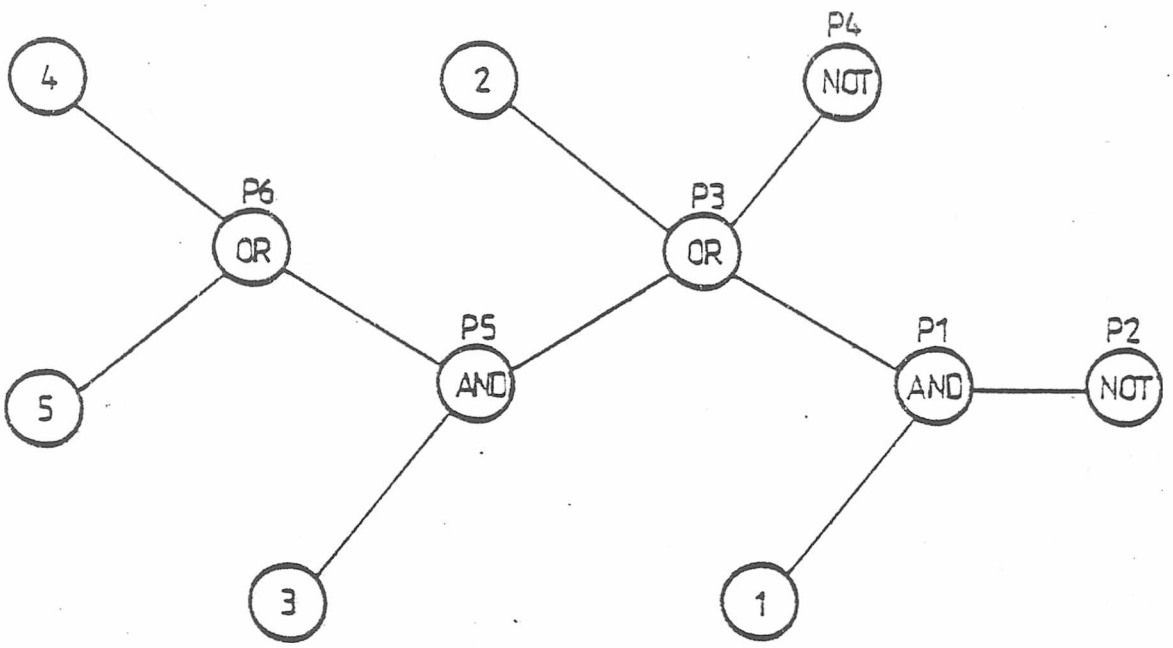


Figure 6 Success Tree of five-component system

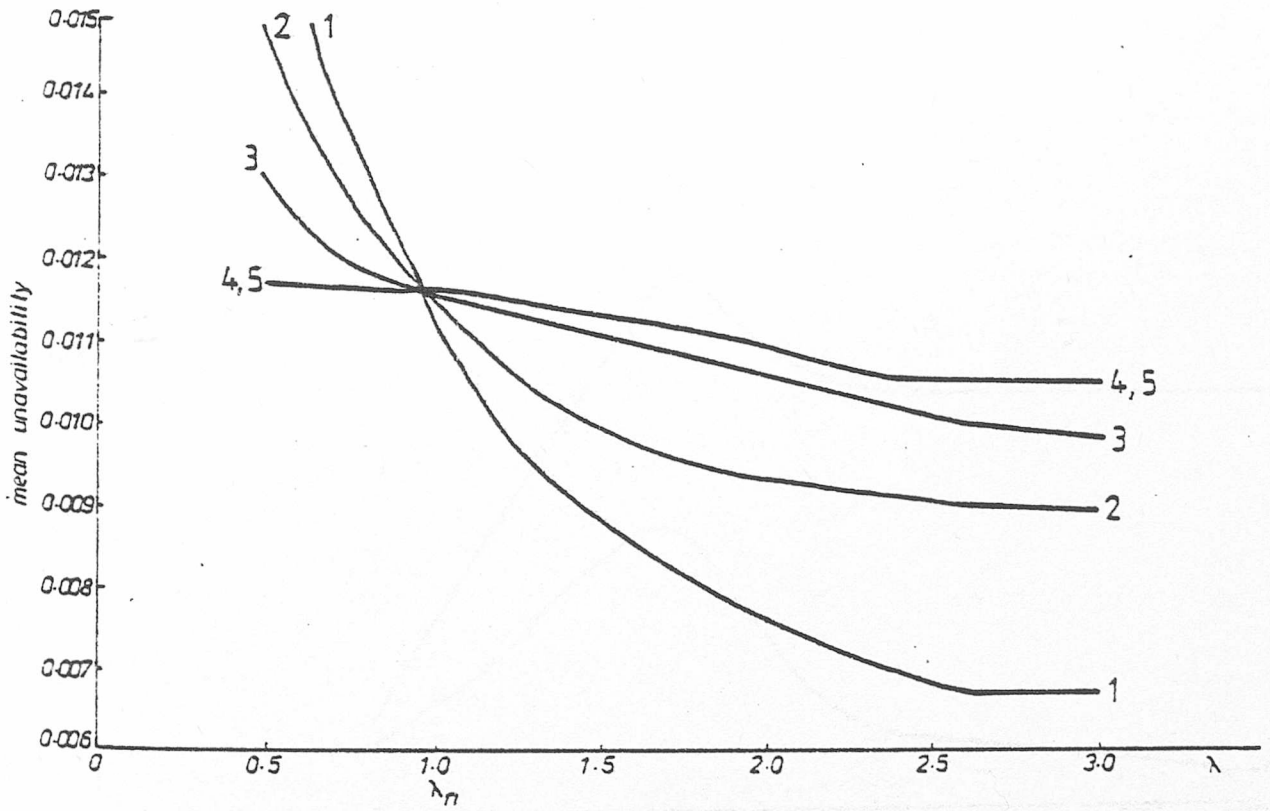


Figure 7 Sensitivity of μ_D to component parameters

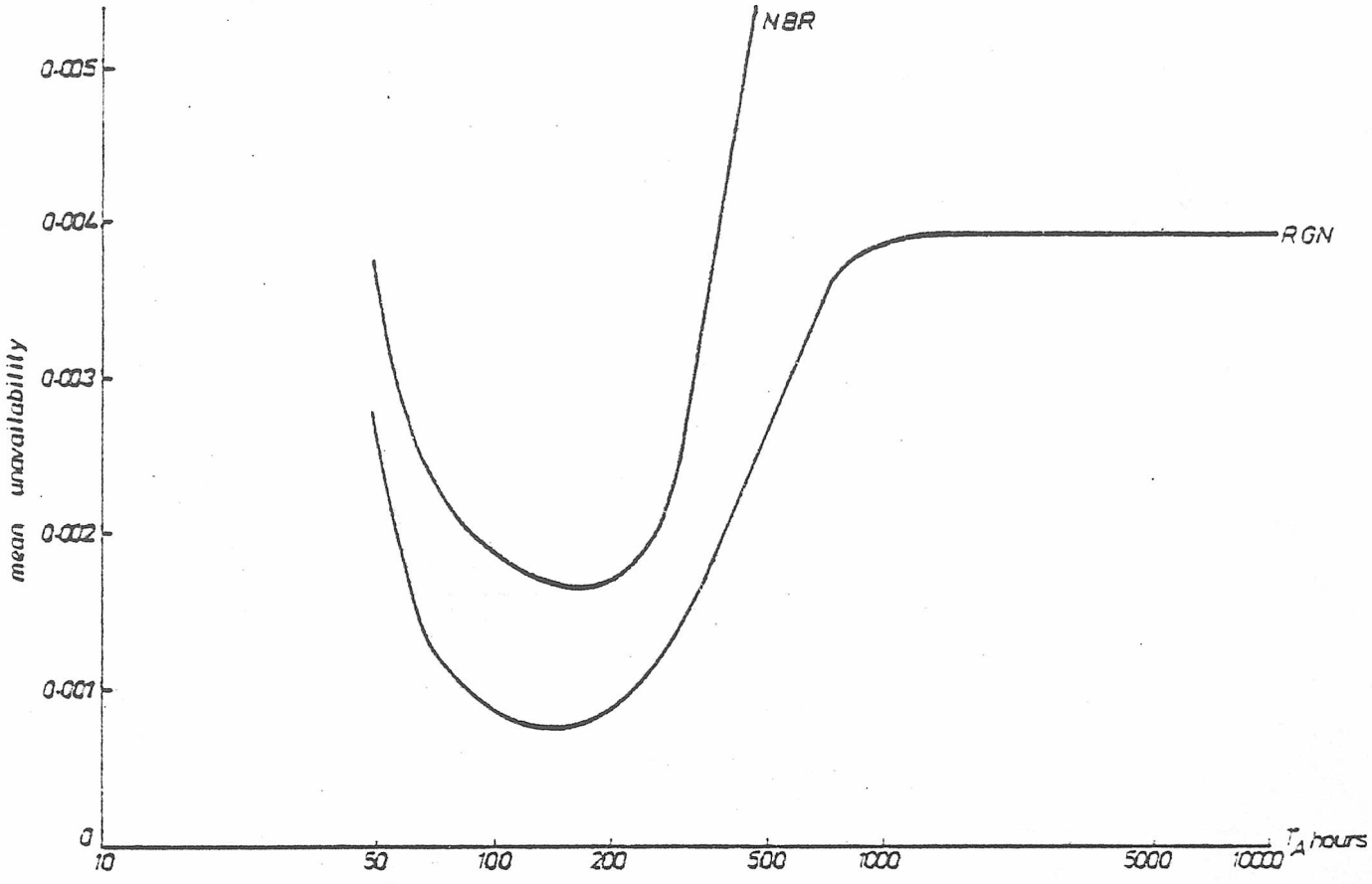


Figure 8 Five component system unavailability with age replacement

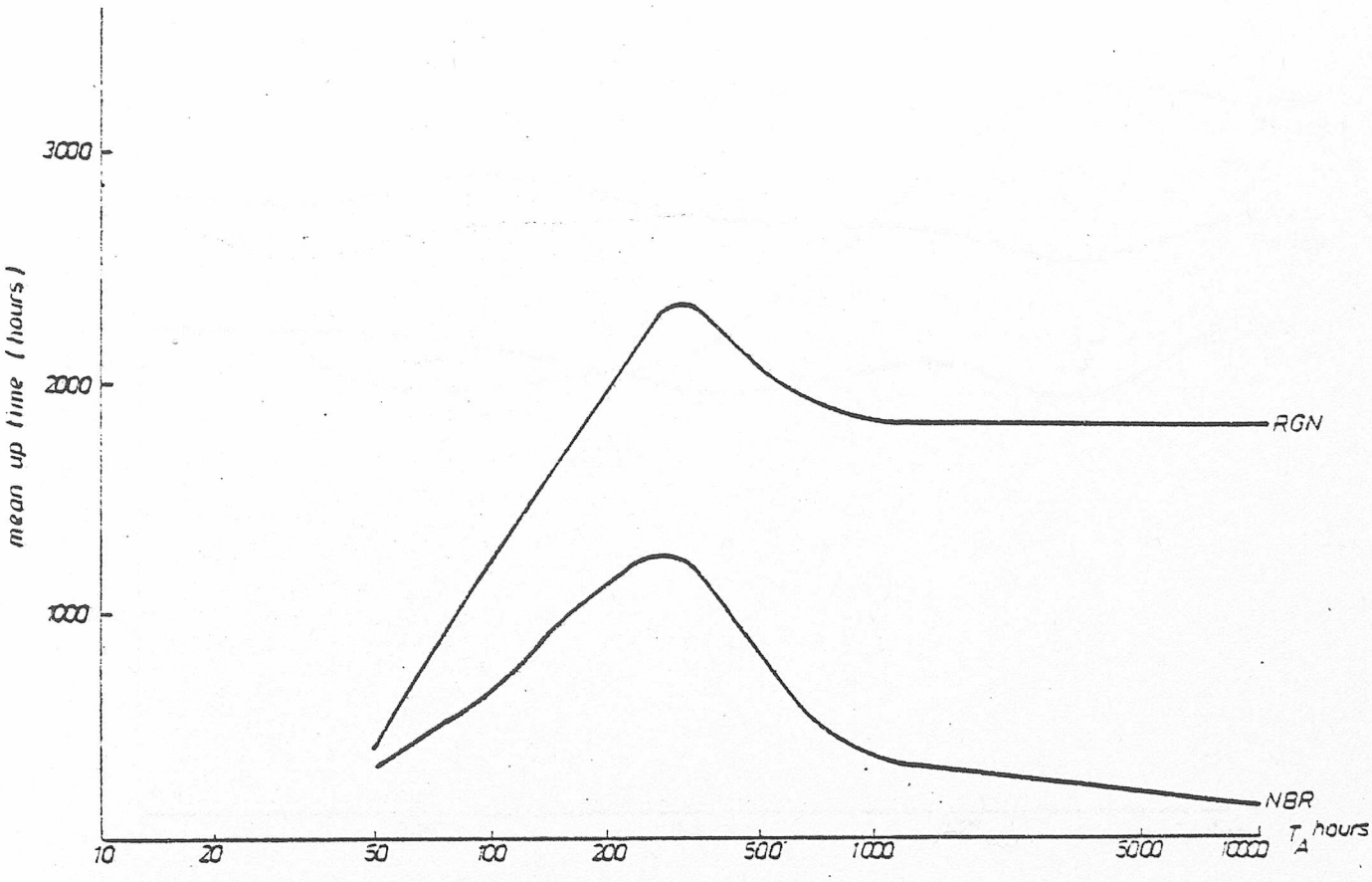


Figure 9 Five component system up-time with age replacement

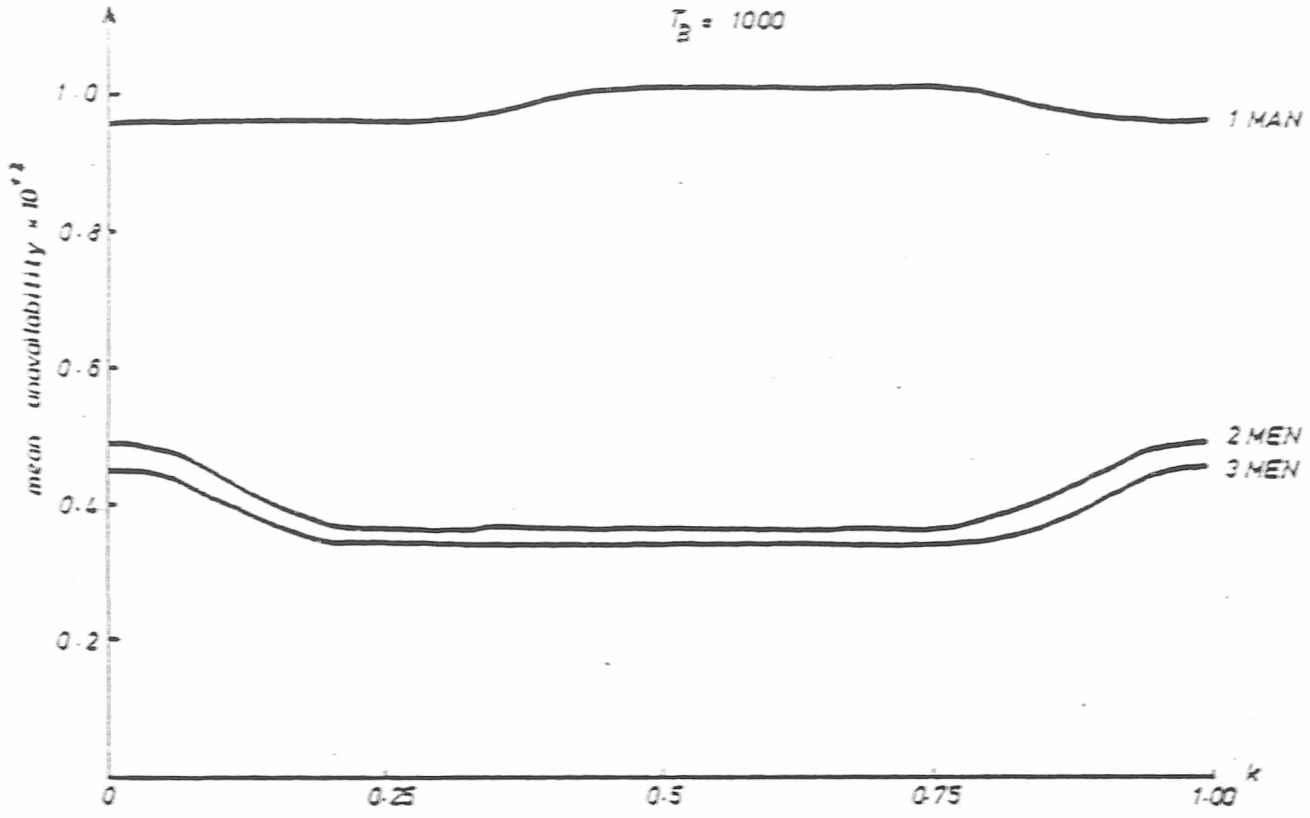


Figure 10 Five component system with fixed policy and block replacement

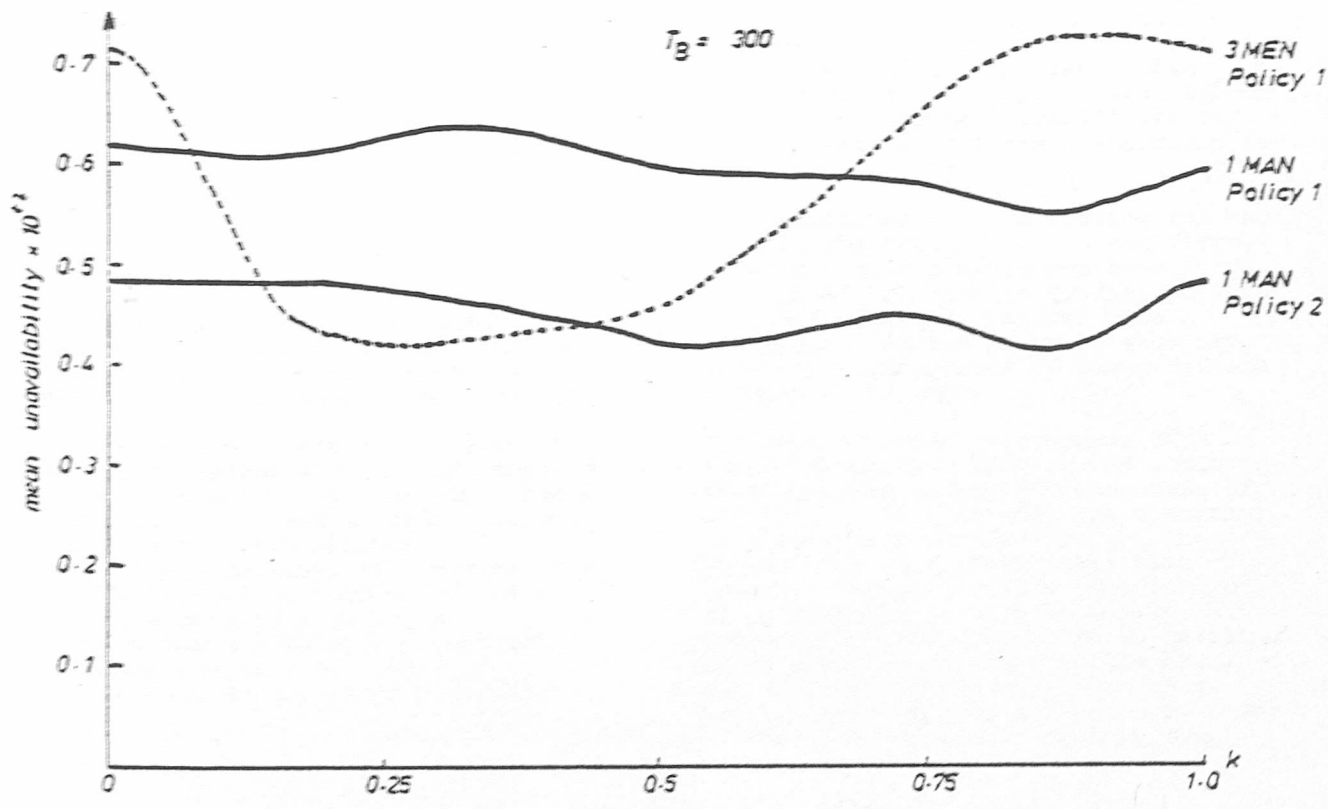


Figure 11 Five component system with variable policy and block replacement

FITH EUROPEAN CONFERENCE ON ELECTRONICS
1982

HIGH-SPEED STOCHASTIC RELIABILITY SIMULATOR

N D Deans and D P Mann
Robert Gordon's Institute of Technology, Aberdeen

A hardware Monte-Carlo simulator is proposed, employing logical trees to define the topology of system components. Component modules are employed by the simulator to model particular system aspects and they may be programmed to accommodate a wide range of behaviour. The highly flexible operation of the component modules yields a simulator well suited to the study of complex unreliable systems. Characteristic failure and repair distributions are selected during an initial programming phase, together with a definition of any maintenance and resource-management policies to be referred to during an experimental run. The design and operational behaviour of such a simulator is reported and the results of a series of experiments carried out using the simulator are presented.

1. INTRODUCTION

Conventional methods of calculating the reliability of a system require a mathematical model of the system to be developed. A measure of the reliability is then obtained by either solving the system equations by analytical means, or by carrying out simulation exercises on the model using a digital computer. Using these techniques, investigators face difficulties :

(i) in establishing appropriate mathematical models for anything other than simple systems.

(ii) in dealing with components whose characteristics of reliability cannot be modelled by exponential distributions.

(iii) in producing results within acceptable times.

Attempts are often made to minimise these difficulties by simplifying the original model, but this leads to results of questionnable accuracy.

A new computing instrument, based on the use of random signals, is proposed for simulating the reliability aspects of a system. Random signals interact with deterministic signals representing features of a system in a stochastic digital processor. As a result, a departure from the conventional von Neumann processing structure can be made, and a considerable degree of parallelism achieved.

The Simulator is in effect a high speed special-purpose digital system acting as a dedicated peripheral device to a host computer. It is comprised of a number of identical programmable 'Component Modules'. Each Component Module represents a particular aspect of the actual system being modelled. In its simplest form, this may be a resistor or transistor, but more complex items of equipment or abstract aspects of a system such as human behaviour can be represented, provided their particular operational characteristics can be defined.

Each component in the system undergoes, during a simulation run, a series of failure and repair processes. In operation, values with prescribed distributions are compared with random numbers, producing probabilities for the instantaneous state transitions for each component.

Two techniques of time-scaling can be used in the simulator. In the first, referred to as synchronous operation, the 'real' lifetime is subdivided into a number of equally spaced time intervals. At the start of each new interval, all aspects of the model are brought up to date.

The second method producing a much faster simulation time can be achieved however using an asynchronous mode of operation. In this mode the updating of a component occurs when a high probability of a state transition exists. The reliability behaviour of all components is simultaneously being modelled and the time at which updating takes place is selected on the basis of the shortest time-to-next-probable-event. According to the event selected, the model is updated. In this mode, the quantum of time that the simulation is incremented is not a constant; for example, during times when the system

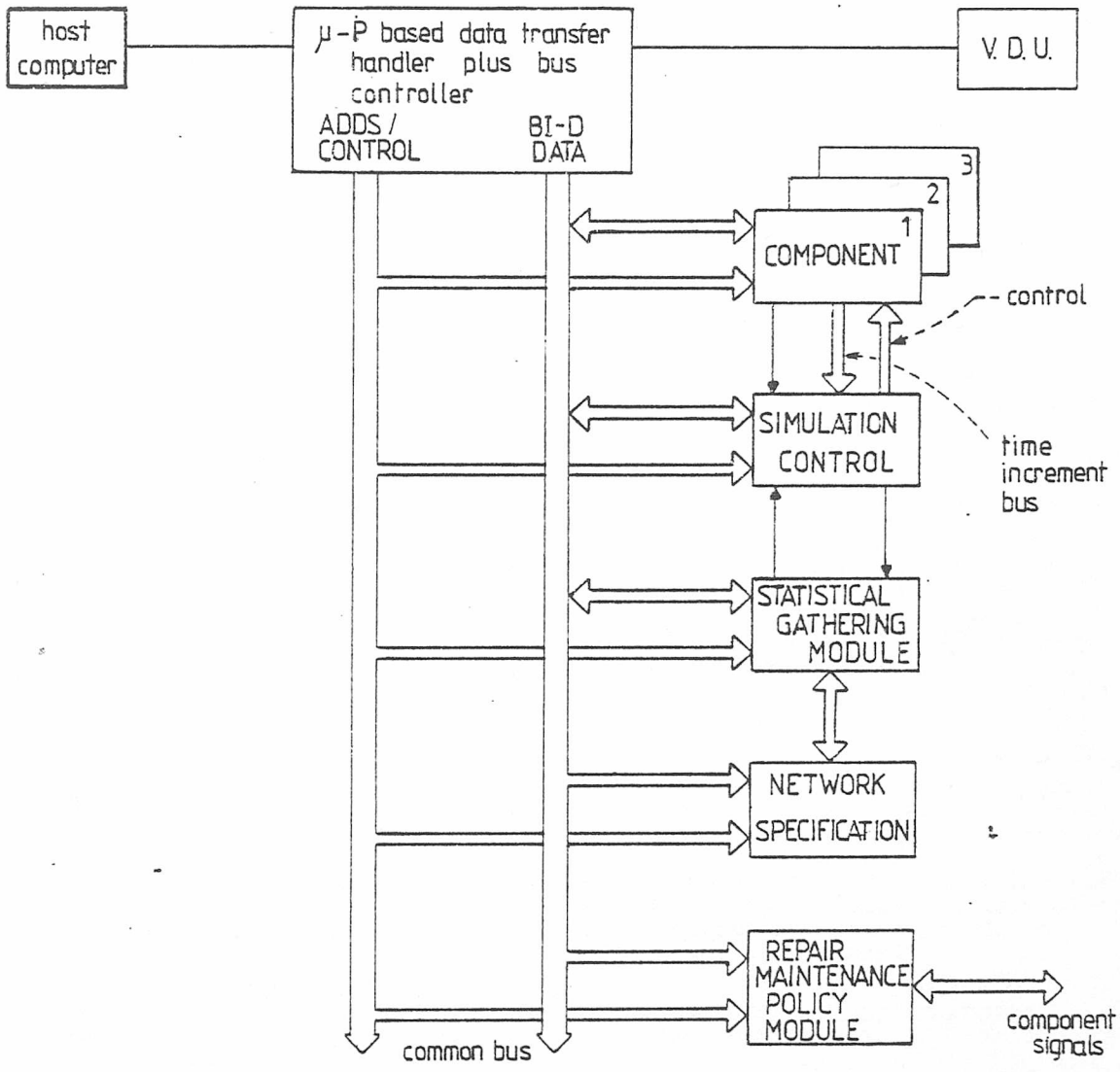


Figure 1 Simulator block diagram

is working, the simulator leaps forward in large time periods (e.g. weeks) whereas when the system is undergoing maintenance or repair, the simulator moves forward in relatively smaller periods (e.g. hours).

The simulation process is controlled by three 'levels' of definition viz

(i) At a Primary Level, special-purpose digital circuitry is used to store the distributions for the failure, maintenance and repair characteristics of each component, and these, together with an internally generated random number source, determine the state (operational, requiring repair, undergoing scheduled maintenance etc) of that component.

(ii) At a Secondary Level, reprogrammable memories are used to provide microprogrammed control of key system features. The particular states that any component can exist in can be individually defined, as can the power devolved to the component to enable it to unilaterally issue requests for maintenance and repair. In addition, particular features may be masked out of a simulation, restricting the range of behaviour of the component.

By redefining the system features, component behaviour can be altered dramtically.

(iii) At a Tertiary Level, a suite of computer programs exist in a main-frame (host) computer to which the

Simulator is connected. These are designed to allow the user to define, inter alia, the topology of the system being simulated, the characteristics of individual components, the specific type of information to be collected during the simulation, and the length of time that the simulation should continue for. Logical networks and tree analyses are used to define the system topology and operational conditions, and graphical entry techniques are used in the host computer to enter such data.

2. MODELLING HARDWARE (PRIMARY LEVEL)

The Simulator takes the form of a number of sub-units interconnected by a common data bus as shown on Fig 1, the whole being connected via a communications processor to a host computer and an operator terminal. A number of Component Modules, each capable of simulating a particular feature of the system being studied, model the reliability characteristics of that system. The remaining sub-units serve to monitor, support and control the Component Modules.

2.1 Component Module

Each Component Module is comprised of the following circuit units :

(i) Memory devices that are used to store the failure and repair distributions. These are reprogrammable, enabling a wide range of distribution types to be used.

(ii) A source of random numbers [1] to interact in a stochastic manner with the component reliability characteristics. The random number generator used [2] ensures a high degree of statistical independence between component behaviour.

(iii) A number of counters to record age replacement times, block replacement times, component age, number of event occurences etc.

(iv) A 'status register' defining the condition of the component at any time. As a result of entering certain states, the Component Module will issue requests for assistance via the Policy Module.

(v) A reprogrammable 'Mask Register' that allows the user of the Simulator to indicate whether or not certain states and features are required.

(vi) A microprogrammable control unit. This enables a considerable amount of decision-making to be devolved to individual Component Modules.

In operation, data from the memories interacts with binary patterns from the number generators, the results of these interactions determining the state of the component. At times specified by counters, certain major events eg the replacement of that component may take place.

2.2 Network Specification Module

Logical trees are used to provide the specification of the topology of the system whose reliability is being studied. The component interconnections are programmed by the user prior to a simulation run by writing data into look-up tables in this module. In addition, the nodes in the logical tree at which statistical data is to be collected are defined.

2.3 Statistical Gathering Module

This unit consists of a number of counters. Data is collected during a simulation run in these counters and transferred to the host computer at the end of a run for inspection and further processing.

2.4 Policy Module

This unit is initially programmed with data relating to the available repair and maintenance resources and to the management policies to be used for the distribution of these resources. The Policy Module is continuously informed by the individual Component Modules of their conditions, and responds to demands for assistance.

3. MICROPROGRAMMABLE FEATURES (SECONDARY LEVEL)

To enable the Component Modules to simulate a wide variety of system features, a number of key control and decision-making circuits within the Simulator are microprogrammed. In particular, the behaviour of the Mask Register and the control electronics in the Component Modules is determined by firmware contained in a series of reprogrammable memories. By writing different data patterns to these memories via the host computer, the response of the Simulator to component behaviour can be altered.

4. OPERATIONAL BEHAVIOUR (TERTIARY LEVEL)

The operation of the Simulator is controlled at the tertiary level by commands issued to the system software in the host computer via a visual display unit. Initially, the characteristics of each component must be entered. In a typical simulation,

the following parameters must be specified :

(i) the failure distribution

(ii) the repair distribution

(iii) the age replacement time (if applicable

(iv) the block replacement time (if applicable)

(v) whether or not the component has a high start-up failure risk

(vi) whether or not the component has a start-up delay time associated with it

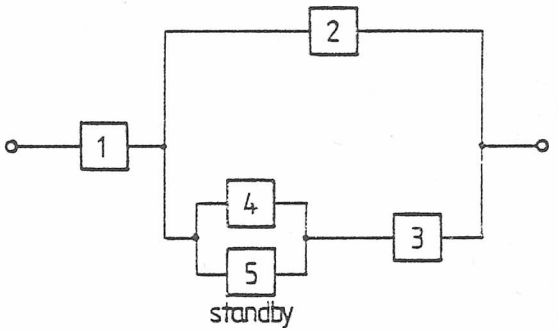


Figure 2 System under consideration

component	Failure distribution	Repair distribution	features
1	exponential λ =2000h ΔT = 4h	exponential λ = 15h ΔT = 1h	
2	Weibull λ = 500h β = 2, ΔT= 8h	Weibull λ = 40h β = 2, ΔT=1h	
3	Weibull λ = 500h β = 2, ΔT= 8h	Weibull λ = 15h β = 2, ΔT=1h	
4	Weibull λ = 250h β = 2, ΔT = 4h	Erlang λ = 10h β = 2, ΔT=1h	
5	Weibull λ = 250h β = 2, ΔT = 4h	Erlang λ = 10h β = 2, ΔT = 1h	cold standby failure rate 1/1000h start up failure risk 10%

Figure 3 Component specification

(vii) the initial age of the component

(viii) whether the component can interrupt the simulation with demands for attention.

The topology of the system is then entered. The component positions in the system and the connections that specify the success tree are defined using an interactive graphics terminal. In addition, the positions in the tree at which statistical information is to be collected during a simulation run are identified. This latter action causes a number of hardware counters in the simulator to be assigned to the task of recording data relating to system performance, e.g. the number of times a particular component was in a non-operational state. Finally the time for which the simulation should run is defined, either as a total elapsed time or as the time to the occurrence of a specified event.

Following an experimental run, the host computer re-engages the data buses of the simulator and gains access to the statistical gathering counters. These are read into the host computer and subjected to various analysing procedures prior to being presented to the experimenter as tabulated results or as graphical information.

5. EXPERIMENTAL RESULTS

A five-component system has been studied. The flow diagram for the system and component parameters are given on figures 2 and 3. When determining system reliability, 'long' simulation times are necessary where event probabilities are low if confidence in the simulation results is to be ensured. To aid the experimenter in determining sufficient simulation run times, the host computer

software determines the Coefficient of Relative Variation (CRV). The CRV can be used as a measure of good statistical estimation and is given by the ratio of standard deviation to mean value. For the example used, the CRV of the unavailability for the system and for component 2 have been determined for various simulation run times. The results are presented in figure 4.

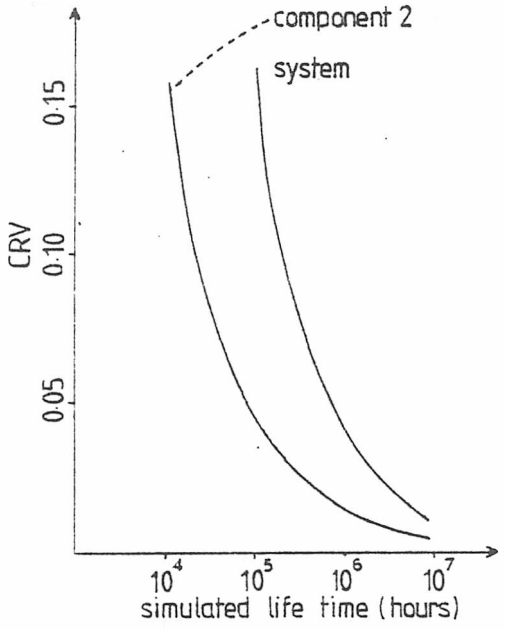


Figure 4 CRV of unavailability

The time required to carry out the investigation and reach a CRV of 0.1 was 0.27 seconds for the overall system and 0.015 seconds for component 2.

The reliability of the system with component 1 omitted has been investigated when a block-replacement policy is adopted.

Component 1 was removed because its particular exponential failure distribution characteristics concealed the significant effects of various preventative maintenance policies. In this truncated system, only Components 2, 3 and 4 took part in the maintenance programme. The cold-standby component (Component 5) underwent normal repair at failure without delay and by a repairman outwith the normal repair team.

Components 3 and 4 are scheduled for simultaneous maintenance at intervals of T hours but start with an initial time shift of kT, $0 \leq k < 1$. The system repair and maintenance resources were such that all requests made by components could be allowed. That is, three 'men' were available to form the normal repair team. The time to carry out maintenance on any component was 1 hour, and the time at which maintenance was applied was chosen to be 1000 hours, 500 hours and 200 hours. The system mean unavailability has been determined for a range of k values and is presented on Figure 5.

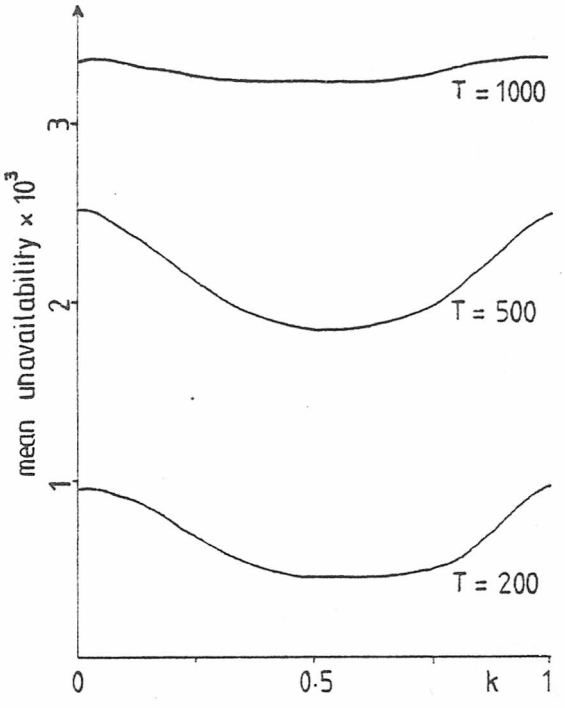


Figure 5 Block replacement

Further investigations have been carried out for an age replacement policy. The repair of components 2, 3 and 4 was considered to be minimal [3] i.e. repair did not return the failure hazard function to its starting value. In this case, the new component condition achieved by maintenance was better than the repaired condition (NBR). To implement this policy, component repair distributions were modelled by exponential functions of equivalent mean. The component maintenance times were exponentially distributed with a mean value of 10 hours. An investigation to determine best age replacement times was carried out. The results for component μ_D are shown in figure 6. From the results, maintenance times of 200 hours, 500 hours and 300 hours were chosen for components 2, 3 and 4 respectively. The maintenance and repair resources could be either 'one man', 'two men' or

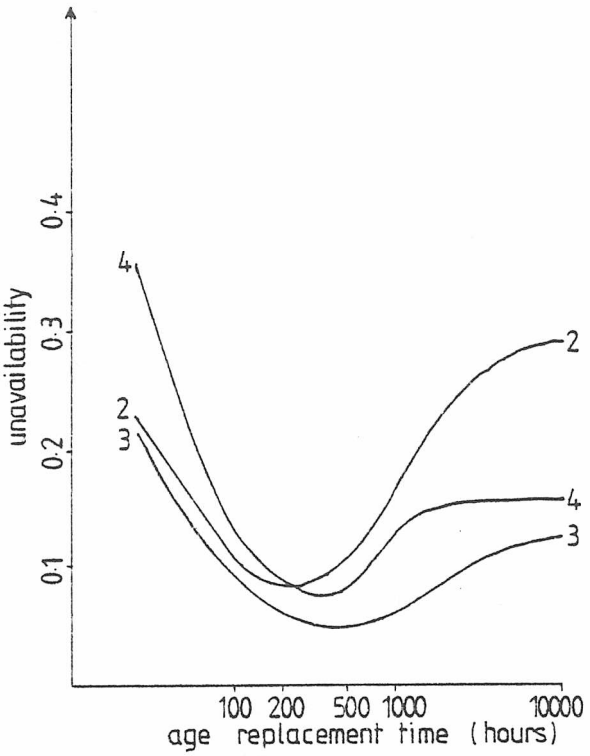


Figure 6 Component μ_0

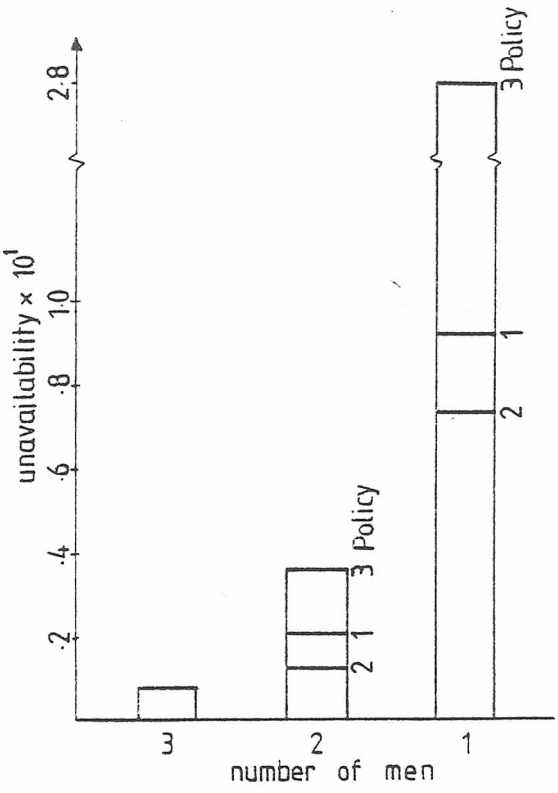


Figure 7

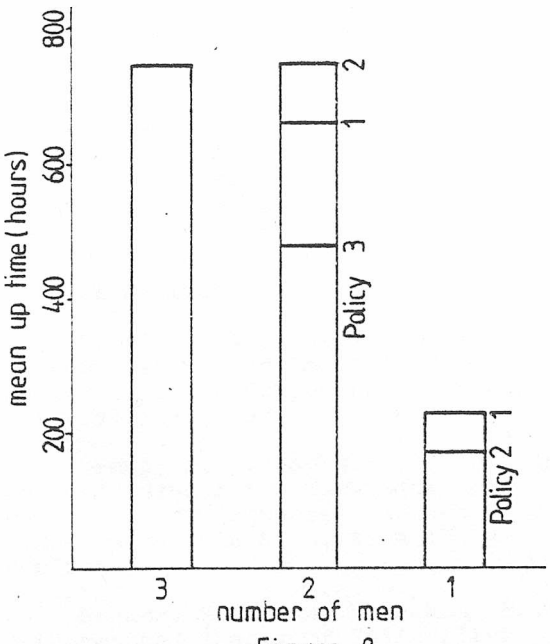


Figure 8

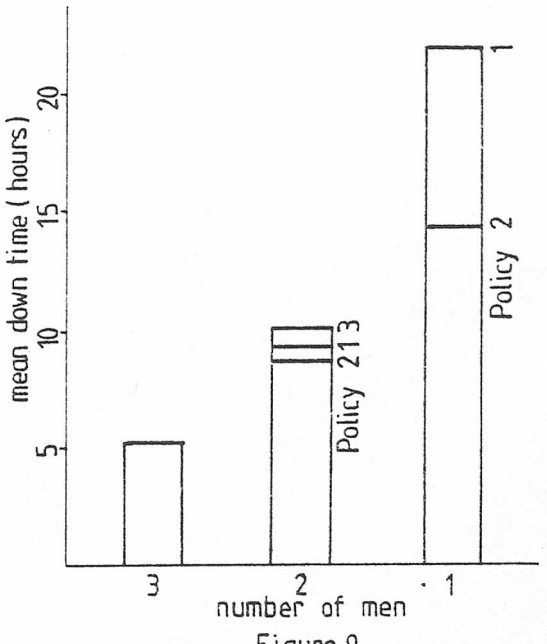


Figure 9

Figures 7, 8 and 9 System behaviour with age replacement and limited repair men

'three men'. applied according to three resource policies. These policies assigned a priority order to component requests for use of the shared resources. Policy 1 gave highest priority to component 2 followed by 3 then 4. Policy 2 was as policy 1, but allowed lower priority requests to be discontinued by the arrival of a higher priority request after the low priority task had commenced. Policy 3 gives components priority 4, 3, 2 with no discontinuing allowed. Varying the global policy and number of policy men, the system unavailability mean up time and mean down time has been determined. The results are presented in figures 7, 8 and 9.

6. CONCLUSIONS

The concept and design of a stochastic reliability simulator has been presented. Special-purpose circuitry, representing components in the system being studied, controls the interaction of deterministic and random number sequences. This interaction results in a definition of the components' states. The component and network specification is initialised by allowing a host computer to access memories installed in the Simulator, and that same computer re-accesses various registers and counters following a simulation run to determine the component and network's operational history.

A series of experiments are reported, confirming the speed and flexibility of the simulator. Lifetimes of 10^6 hours are shown to be simulated in less than one second and the ease with which global resource policies can be studied are demonstrated.

7. REFERENCES

[1] Birolini, A., Hardware Simulation of Semi-Markov and Related Processes, Mathematics and Computers in Simulation XIX (1977) 75 - 97

[2] Deans, N. D. and Mann, D. P., An Improved Generation Technique for Random Number Sequences, Mathematics and Computers in Simulation (To be Published)

[3] Barlow, R. E. and Proschan, F., Mathematical Theory of Reliability (Wiley 1965)

8. ACKNOWLEDGEMENTS

The support of the Science and Engineering Research Council, The National Centre of Systems Reliability, UKAEA and the assistance of Mr B Davidson and Miss K Craighead are acknowledged.

N Deans studied electrical engineering at Aberdeen University and received his B Sc(Eng) degree in 1966. Following a period as a research and development engineer at Ferranti, he carried on research work at Heriot-Watt University in Edinburgh for which he received a Ph D degree. In 1973 he was appointed as a lecturer in digital systems and became senior lecturer in microcomputing in 1979.

D Mann studied electronic and electrical engineering at Robert Gordon's Institute of Technology in Aberdeen, graduating with a B Sc degree in 1979. Since then, he has investigated the concept and design of special-purpose digital circuitry, based on the use of stochastic signals, to carry out signal processing, with special emphasis on the simulation of system reliability.