EVALUATION OF ALTERNATIVE DISCRETE-EVENT SIMULATION EXPERIMENTAL METHODS

ALAN JAMES WARN

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

June 2003

Bournemouth University

ALAN JAMES WARN

ABSTRACT

The aim of the research was to assist non-experts produce meaningful, non-terminating discrete-event simulations studies. The exemplar used was manufacturing applications, in particular sequential production lines. The thesis addressed the selection of methods for introducing randomness, setting the length of individual simulation runs, and determining the conditions for starting measurements. "Received wisdom" in these aspects of simulation experimentation was not accepted. The research made use of a Markov Chain queuing model and statistical analysis of exhaustive computer-based experimentation using test models. A specific production-line model drawn from the motor industry was used as a point of reference.

A distinctive, quality control like, process of facilitating the controlled introduction of "representative randomness" from a pseudo random-number generator was developed, rather than relying on a generator's a priori performance in standard statistical tests of randomness. This approach proved to be effective and practical.

Other results included:

- The distortion in measurements due to the initial conditions of a simulation run of a queue was only corrected by a lengthy run and not by discarding early results.
- Simulation experiments of the same queue, demonstrated that, a single long run gave greater accuracy than having multiple runs.
- The choice of random number generator is less important than the choice of seed. Notably, RANDU (a "discredited" MLCG) with careful seed selection was able to outperform in tests both real random numbers, and other MLCGs if their seed were chosen randomly, 99.8% of the time. Similar results were obtained for Mersenne Twister and Descriptive Sampling.
- Descriptive Sampling was found to provide the best samples and was less susceptible to errors in the forecast of the required sample size.
- A method of determining the run length of the simulation that would ensure the run was representative of the true conditions was proposed.

An interactive computer program was created to assist in the calculation of the run length of a simulation and determine seeds so as to obtain "highly representative" samples, demonstrating the facility required in simulation software to support these selected methods.

TABLE OF CONTENTS

TABLES

GRAPHS

DIAGRAMS

ACKNOWLEDGEMENTS

I would like to thank Professors Brian Hollocks and Colin Armistead of the Bournemouth Business School for their help and assistance. Also to the many simulation experts who answered my questions about their work and especially John Ladbrook who provided me with information on transfer lines.

I am very grateful to all at Lanner for their assistance and for providing me with a copy of WITNESS.

Similarly I would like to thank my wife Gretel and the other members of my family for their forbearance.

I would like to give special thanks to Mr Frederick Dyke for allowing my wife and myself to stay in his house while we were visiting Bournemouth.

Alan J Warn

1. INTRODUCTION

1.1 THE IMPORTANCE OF SIMULATION

Simulation is the construction and use of a computer-based representation, or *model*, of some part of the real world as a vehicle for experimentation, where to experiment on the real world would be impractical, costly, or impossible. The purpose of the experimentation is to enable a user-organisation to develop and validate plans in advance of committing to some change (such as new investment in plant or new operating practice).

Discrete-event simulation is that form of simulation where the factors being modelled may be regarded as changing only at discrete points in time (known as "events"), for example a supermarket checkout, a car-body assembly line, or an airport terminal.

Continuous simulation is the form of simulation where the factors being modelled vary continuously over time. Examples are the temperature changes within an iron ingot and the relative position of a sprung axle travelling over a rough surface.

This research is concerned with discrete-event simulation.

An example of such a study is one that was made when a major vehicle manufacturer was proposing a major investment in a test facility for motorcar engines. A discrete-event simulation model was constructed of the proposed design. One of the decisions that faced the engineers was which of two types of material handling devices (called wait tables) was to be used. One wait table had complicated operating logic and would store more than one engine while the alternative had only simple operating logic and was only able to store one engine.

The more complex equipment needed more maintenance, had a shorter expected time between failure and a longer expected repair time due to the greater number of moving parts and complex logic. It did however offer greater buffering capability. Much to the disappointment of the Design Engineers, when the two designs were compared using the simulation model, the simpler equipment gave a higher throughput. The savings, made from using the simpler material handling equipment, allowed for more conveyors to be installed, which reduced the impact of breakdowns, and thus an even higher long-term throughput was obtained.

Mathematical analysis of such real industrial situations is often intractable and requires the analyst to make many simplifications to enable a solution to be obtained. The applicability of any results obtained from such analysis is thus restricted to situations where the simplifications are acceptable. The ability of simulation to examine real situations with the minimum of simplification, has thus given it great appeal and it is reported as the most frequently used simulation technique in Operational Research (Harpell et al. 1989). It is also reported as being successful (Simulation Study Group 1991).

It is surprising that simulation is not as widely used throughout industry as would be expected. There is a popular view that simulation is only suitable for the larger companies. The DTI initiated an investigation into why it was not used in the middle size companies (Simulation Study Group 1991). The study concluded that one of the main reasons was the need for an expert analyst. Such an expert is not always available even in a large company.

By way of comparison with simulation it is instructive to reflect on the use of Statistical Experimental Design. Grove and Davis (1992) attribute the "extraordinary progress in quality improvement made by the Japanese," to the ability of the Japanese engineers to perform experiments based on statistical methods, through the creation by Genichi Taguchi of statistical experimental methods that are both easy to understand and to apply.

There may be many who doubt some of the theoretical basis of Taguchi's methods but his "cookbook-like" guides have enabled this revolution. The older standard texts on experimental design often reflected agricultural experiments. This was probably due to the origins of the methodology in the 1920s (Fisher 1960). It was generally considered that a statistician was required to apply the techniques. The move to totally industry based and "easily understood" procedures has led to statistical experimental design being available to all engineers and to a much wider use. A similar establishment of a set of simple basic rules for simulation studies could give the technique a similar stimulus.

Indeed Wadsack and Tobias (1994) stated that there was a need for a framework giving assistance "beyond automatic run generation and report generation" to "help practitioners to ensure experimental validity." Hollocks (1995) also expressed the need to provide the practitioner with a robust experimental framework that will at least prevent erroneous results from incomplete experimentation and if possible enable the identification of the optimum design to be made within the existing time pressures.

Wadsack and Tobias confirmed that they found, from their interviews, that many organisations were operating under time pressures. Wadsack and Tobias expressed concern with the limited experimentation and analysis performed by these organisations. They found that many organisations were, in their view, "not allowing their models to 'warm-up' sufficiently" and "not running their models for a long enough period of time" and they reported that they found

some instances where the organisation "only performs a single simulation run at the 'eleventh hour'."

1.2 THE AIM OF THIS RESEARCH

Although the use of simulation may be broken down to a large number of phases, it is possible to group them into three major phases:

1. Construction of the model to reflect the real life system,
2. Using the model to resolve the questions being asked of the real life system
3. Applying the results.

In this research, decisions within the second phase were studied (the phase highlighted as a concern by Wadsack and Tobias, 1994). The decisions investigated were those made at the commencement of a simulation "run."

To provide the required information from a model using simulation, a number of "runs" are made in which various factors are changed. In the example of the test facility for motorcar engines described earlier the question asked was, "What type of material handling equipment should be used?" To answer this question two sets of runs were made. One set was with the simple material handling equipment and the other set was with the complex material handling equipment. From the results of the two sets of runs the most effective material handling equipment was then selected.

But it is not that straightforward. In real life there is variability in performance and there are unpredictable breakdowns. For the simulation model to be realistic some form of randomness needs to be introduced to reflect the stochastic nature of the various real life processes. Thus any analysis of the measurements from such a system must be based on statistical methods.

In the example above any conclusion about the benefits of one material handling system over the other must be based on enough information, which means, in practical terms, making sufficient number of runs of adequate length in order to determine that any measured difference is real.

It is also necessary, in order to achieve trustworthy results, not only to have runs long enough to truly reflect the performance of the system but to have suitable conditions at the start of the measurements so that all the measurements are relevant. The relevant conditions that need to be considered may include for example whether machines or men are busy or idle or how many parts should be in the storage areas.

The aim of this research was to select simple methods for use by non-specialists who wish to perform discrete event simulation runs. These methods would later be included in a total framework, which when finally created, will enable the non-expert user to obtain accurate and meaningful results with the least use of his or her resources. In this thesis the aim was to evaluate the choices made when establishing the actual simulation run. This is only a part of the total framework required. However the decisions made while setting up the computer run may be seen to be crucial, as it is unlikely that any action taken later would be able to eliminate any errors introduced at this stage. For example, if the sampling procedures are non-representative or if the run length is too short to give an accurate reflection of the performance, incorrect conclusions may be made on the accuracy obtained in the simulation and the significance of the results.

The aim of the research was also to determine what changes are required in the specialist software to enable the selected methods to operate in their systems.

To produce a tighter definition of the aims of the research it was necessary to determine the decisions required to establish a simulation run.

1.3 THE DECISIONS REQUIRED TO ESTABLISH A SIMULATION RUN

It has been assumed that the analyst engaged in a simulation study is at the stage where there exists a plan of experimentation runs, probably based on statistical methods, to test the various alternative designs. The requirement then is to perform the individual set of tests such as to obtain meaningful results.

As previously mentioned, one of, if not the most, critical requirement is a sound method of introducing some form of randomness into the simulation to accurately reflect the true stochastic nature of real life. In practical terms this has usually meant using a random number generator. Such a random number generator is an arithmetic calculation whose resulting sequence of numbers is deemed random. Construction of simulation models has been made easier by the availability of specialist packages. The specialist packages normally have imbedded generators that often cannot be changed by the user of the package. The choice is often, as in Witness (Lanner Group 1998), one of the most popular simulation software packages, to select from a limited number of a predetermined seeds. The imbedded generator may not be satisfactory, as is demonstrated by the concern with Microsoft's EXCEL whose random number generator has indeed been deemed unsatisfactory (McCullough and Wilson 1999, L'Ecuyer 2001). Although EXCEL is software designed for creation of spreadsheet rather than being a specialist simulation software package, it is widely used and indicates that any embedded software within a package must be reviewed with caution. Indeed the lack of a random number generator that is universally accepted is discussed later.

There has been a revival of interest in real random number generators (these are not based on an arithmetic procedure but instead use some naturally occurring "chaotic" process to create the random number) and these will be discussed later. Indeed a set of real random numbers was used in the thesis to perform a mathematical calculation (in this case, a calibration) using a numerical analysis method, Monte Carlo Method, which is based on random numbers, (see Hammersly and Handscomb 1964) where traditional analytical methods were considered not to be feasible. As yet (2003) real random number generators have not been seriously proposed for use in present day discrete-event simulation.

The arithmetic procedures require a number, or numbers, to be selected in order to initiate the sequence. These numbers are called "seeds". It will be seen that the choice of these seed values has received virtually no analysis in the literature. Indeed the choice of seed or seeds has usually been considered as unimportant.

Most writers considered it is important to determine a point or time in the simulation run when measurements should begin to be taken and this is not usually the beginning of the simulation run. This is because it is usual after having developed a model for it to be empty, that is it is without intermediate stocks of partly worked parts, and has machines and men idle. Such a position may be very unusual in real life and if the simulation was to start from this position and measurements were taken from the start, the initial readings would be correctly considered as untypical. It is generally supported in the literature (see for example Law and Kelton 2000) that when measuring the "on-going" performance of a system that it is necessary to have a "warm up" period, and not to take measurements until this period has passed so as to overcome any distortion due to the actual starting position. Indeed many writers state that measurements should only be taken when "steady state" is obtained. As will be

discussed later, "steady state" is difficult to detect and the need to discard early readings has been questioned.

The length of run after the "warm up" period must be long enough so that the results obtained are statistically sound and thus meaningful. This may be considered as not a vital decision at the start of a simulation run as measurements during the run can be made and these may be used to determine if the run was of sufficient length and, if not, the run could be extended. Such an analysis is not so straightforward, as most systems will have a degree of autocorrelation in any frequently measured metric of performance, since most industrial systems will have queues and machines whose state will be dependent on their condition when the previous measurement of the metric of performance was made. Thus the measurements taken over time will not be statistically independent, as is required by many statistical tests. There is information in the literature on techniques to determine the adequacy of a run that take account of autocorrelation and a reference to them is given later. As stated by Robinson (1994) these methods still rely on the initial run being sufficiently long as to be representative otherwise any measurement to determine if the run length has been sufficient would be flawed. In this thesis, the determination of a suitable run length before any simulation runs have made was considered. There is virtually no analysis in the literature available to assist in this decision. However they may exist a number of rules of thumb. However they do not appear to have been published. Robinson (1994) made reference to one such rule of thumb:

> "during a run, at least ten to twenty samples should
> be taken from every distribution of the model"

In a personal communication Robinson (2003) stated that he now considered that this would not lead to an adequate run time.

For this research the following fundamental basic design decisions were rigorously investigated:

- The choice of method of introducing randomness into the simulation
- Determination of the initial condition and warm up period
- The initial setting of the duration of the computer run.

2. THE METHODOLOGY USED

For this research, a real life model of a part of a production plant producing engines for a medium size luxury car was used as the test vehicle. This consists of a machining line and an engine assembly line. Ladbrook (1998) reports that five similar facilities (four machining lines and one engine assembly line) would require an investment of over 400 million US dollars.

The majority of analysis in this thesis was made using mathematical techniques that included simulation, Monte Carlo Methods, and use of Markov Chain models. Analytical mathematical analysis (e.g. Markov Chain analysis) of even this part of the production plant, without excessive simplification, was not considered feasible due to the large number of state variables and associated values (See for example Kouikoglou and Phillis 2001 pages 1-7) and such analysis was restricted to single elements (queues) within the model.

Alternative methodologies based on studying practices currently employed by users of simulation, such as surveys or ethnographics were considered to be less likely to be successful due to the limited range of statistical procedures either used or able to be used by the majority of modellers (Wadsack and Tobias 1994, Hollocks 1995). One of the causes of this limitation is that most efficient model-builders use specialised simulation software that restrict the statistical methods available to the user. An example that will be discussed later is the imbedded random number generators. Thus any results from such a study using these methodologies would tend to perpetuate the weaknesses that exist in current practice.

Action learning techniques were also rejected as using a real life development would have imposed problems if techniques being tested proved to be inefficient, since almost all projects in industry are subject to time pressures and any wasted time would be unacceptable (See Wadsack and Tobias 1994).

3. LITERATURE REVIEW

3.1 FOCUS OF THE REVIEW

As stated in the earlier section, the following design decisions were investigated:

- The choice of method of introducing randomness into the simulation
- Determination of the initial condition and warm up period
- The initial setting of the duration of the computer run.

The current literature was reviewed to determine what was the current understanding of the best methods of making these decisions.

The first review was to examine the methods proposed to introduce randomness. Since randomness especially in the form of random numbers is used in numerous fields other than simulation, notably the fields of Monte Carlo Methods and Cryptography, the literature is widespread. The designers of generators of random numbers appear to be aiming to design them so that they are universally applicable rather than for discrete-event simulation studies. The review has concentrated on literature that apparently is for, although not exclusively for, the simulation community.

3.2 DEFINITION OF RANDOMNESS

In order to understand the difficulties facing the designers of methods of introducing randomness it is instructive to consider the definition of randomness and random numbers.

3.2.1 Definition of Randomness and Random Numbers

Bennett (1998) gives a brief description of the main attempts to define randomness but states that the search for a universally accepted definition has not yet been successful. The usual form of randomness used in simulation studies is a stream of random numbers where the numbers take random values between 0 and 1.

In developing methods of creating random numbers in discrete-event simulation, most writers have assumed a pragmatic definition for randomness. They consider a sequence of numbers is to be acceptable as being random if it passes a set of established tests (see for example Knuth 1998). The term "pseudorandom numbers" is a convenient term for a string of random numbers generated by some arithmetic process. Such "pseudorandom numbers" would not suffice for all purposes; they cannot for example replace the randomising device used in the British National Lottery.

Bennett states, that such practical definitions "are not without their deficiencies." In such a pragmatic approach a generator is rejected if and only if it creates a string of numbers that is unlikely to be random to such an extent as to give incorrect results in our proposed use.

This reference to "proposed use" is important since random numbers are not only used in simulation. They are useful in mathematical methods referred to as Monte Carlo methods where the mathematics of statistics is used to solve problems that are deterministic (Hammersly and Handscomb 1964). Often Monte Carlo Methods are used where "traditional" analytical methods are impractical or even impossible. These methods typically require large sequences of random numbers.

Another area where random numbers are required is in cryptography. Here the desire is for the sequence to be unpredictable and that the generating function cannot be discovered in any reasonable time (see for example Blum et al.1986).

The development of methods of creating random numbers and the development of tests have proceeded in parallel. As one method of creating random numbers is developed, a test has been developed which the method fails. For this study the development of tests will first be considered then the development of methods.

3.2.2 Tests to Ensure a Stream of Numbers is Acceptable as Random

At present no professional or learned body has established a standard list of tests, a recommended random number generator, or a standard set of tables. Knuth (1980 and 1998) however gives a range of empirical tests (see table 3.1 on page 26).

Tocher (1960) gives several of the tests given later by Knuth. Several of the tests were devised by Kendall and Babington-Smith (1938,1939). These are indicated within table 3.1. The Babington-Smith tests were originally designed to be applied to the individual digits in the random sequence. Tocher also gives another test that Knuth did not include in his list. Tocher states that Yule (see Yule 1938) devised it, and refers to it as the "Yule's test". The test consisted of determining the distribution of sums of sets of non-overlapping 5 digits within the sequence and testing the actual against the theoretical distribution. This test seems to have been forgotten, maybe due to the need to determine the theoretical distribution by a "practical arithmetic process."

Test Name	Brief Description
Equidistribution test**	The spread of the random numbers over the whole range
Serial test**	The frequency that a certain of pairs of numbers occur
Gap test**	The length of sequences between certain range of values
Poker test**	The values of sets of five sequential numbers
Coupon collector's test	The length of sequence required to obtain all the values
Permutation test	The frequency of the different relative orderings of sets of a fixed number of sequential numbers
Runs test**	Test the frequency of run lengths of numbers increasing (or decreasing)
Maximum-of-t test	Test the frequency of the maximum number in a set of t sequential numbers
Collision test	Test distribution of n-tuples in n-dimensional space. Description given later.
Birthday spacing test*	Test the frequency of the difference in the numbers of a fixed length sequence after sorting. A number of sequences need to be tested
Serial correlation test	Test the serial correlation between n sequential numbers and the same sequence cyclically shifted sequence

* Not in the second edition 1980 but included in the third edition 1998.

** Included by Tocher (1960) devised by Kendall and Babington-Smith (1938)

Table 3.1 The List of Empirical Test Given by Knuth

Another test described by Tocher, the D^2 test, was devised by Gruenberger and Mark (1951), in which four consecutive random numbers are used to construct two points within a square and the distance between the points is calculated. The test consists of comparing the distribution of the actual distances with the theoretical distribution. This test is special since it is aimed at determining if the random number stream is suitable for its purpose, which in this case is evaluating integrals. Tocher considered it would be ideal to use tests that were "tailor-made" for the proposed use of the random numbers. He stated that in 1960, "This tailor-made approach, of course, is not very realistic in practice."

Knuth classifies the tests for random number generators as being either empirical or theoretical. He states that, in a theoretical test, the statistic can be calculated from the values used in the function that creates the sequence. This requires there to be developed an analytical method to calculate the statistic, which even Knuth himself states is difficult.

In empirical tests, the actual sequence is analysed. Knuth warns that even if a generator passes a theoretical test, the sequence to be used should be tested for short-term non-random behaviour by empirical tests.

Marsaglia (1985) states that the list of tests given by Knuth is not to be taken as the standard, however in practice many writers still refer to the tests in Knuth's list. (See for example Law and Kelton 2000). Knuth states that of the tests in Table 3.1, the "spectral test", a theoretical test is the most powerful, and any generator with high "accuracy" scores is acceptable. The spectral test is only applicable to one type of random number generator, but it is the most popular (This generator, the Mixed Linear Congruent Generator or MLCG, and the "spectral test" are described later).

Marsaglia gives a set of more stringent tests and other sets of tests exist. Dudewicz and Ralley (1981) created TESTRAND, although not now available, which is basically the same as Knuth's. Marsaglia has made his tests available

as a battery of tests called DIEHARD, which is available on the Internet (Marsaglia 2002). Most tests, as can be seen from Table 3.1, are based on simple concepts, for example the length of sequences of ascending or descending numbers. Dudewicz and van de Meulen (1983) (also Walker 1998) have developed a more radical approach based on the thermodynamic measure of disorder, which is termed entropy, and introduced the concept of "the entropy of the sequence". However, the use of this "entropy measure" was put into doubt by Bernhofen et al. (1996) who showed that RANDU, a discredited generator (see for example Knuth 1998 and Appendix 2), is "recommended" by the Dudewicz and van de Meulen's "entropy measure"!

The concerns regarding the validity of existing random number generators have led to this growth in the number of tests. As Marsaglia (1985) states, there is no problem in creating more tests for determining whether a sequence is random. This reflects the statement of the earlier workers, Kendall and Babington-Smith (1938), that the number of tests that can be developed is only limited by human ingenuity and, as Tocher noted, "their implication that a test may be able to be found that will fail any particular sequence of random numbers".

Currently many workers in the field are considering tests based on creating points in n-dimensional space. They consider that these points are created by the sequence of random numbers; a sequence of n random numbers being used to define a single point in n-dimensional space. A lengthy sequence of random numbers will create many points and various tests have been design to check if the distribution of these points is random. An early version was one by Davis and Rabinowitz (1956). Here the distribution of the number of points falling within an n-dimensional sphere is compared with the theoretical distribution. A direct use of this concept is the measure of merit, μ, used by Knuth in determining the performance of generators when analysed by the "spectral test". Here μ_n is related to the number of points, formed by the

sequence of "pseudorandom numbers" created by the MLCG being tested, in an ellipsoid in n-dimensional space. (See page 105 Knuth 1997)

Marsaglia (2000), although fully supporting the use of this type of test, states that these tests require a large number of random numbers.

Two recently described tests of this form that are often failed by the most popular random number generator, the Mixed Linear Congruent Generator (described later), are the Collision and Birthday Spacing tests (both now included in Knuth's list). The Collision test is similar to the test of Davis and Rabinowitz. Descriptions of the Collision test and the Birthday Spacing test, which was devised by Marsaglia (1985), and the test statistic applied, the measurement of discrepancy, are given by L'Ecuyer (2001).

A brief description of these two tests is that they measure the difference between the positions of points in n-dimensional space. This is performed by dividing the axes of the space into m equal divisions, thus forming m^n cells. Each cell has n sides. The tests consist of determining how many cells have two points (Collision test) and the distribution of the distance between occupied cells (Birthday Spacing test). The distribution of the number of expected collisions and the distance between occupied cells can be calculated from statistical theory.

Statistical methodology states that the null hypothesis "that the sequence is a random sequence" should be rejected if the probability of obtaining the result actually obtained, if the null hypothesis were true, would be small. Thus if statistical methodology is strictly applied, generators giving evenly distributed points, which may have a low probability of occurring if the points were really random, will be rejected equally as generators that give "severely bunched" points. Indeed L'Ecuyer (2001) strictly applied the methodology and when using the collision test rejected random number generator that had too few, as well as those having too many collisions. Therefore he proposed rejecting

random number generators based on the spread of random numbers in n-dimensional space being too uniform! Indeed, Niederreiter (1992) had stated that MLCGs produced pseudo-random numbers with too regular in structure and thus were consistently failing these tests.

There is however a valid case for selecting a generator where the spread would be considered by a statistical test as being "too uniform" since it has been noted that a generator that gives an even spread of random numbers gives more accurate results (Saliby 1990a). This is considered to be true for Monte Carlo analyses where "quasi-random numbers" have been to obtain greater accuracy (Niederreiter 1978). These "quasi-random numbers" are specially generated to give evenly spread random numbers. Indeed in a paper addressing Monte Carlo methods based on quasi-random numbers, L'Ecuyer and Lemieux (1999) contradicts both the need for a long cycle, that L'Ecuyer usually advocates (see discussion later), and for randomly distributed points. In this paper they state,

> "What we suggest is the opposite: Take a small random number generator with only n states, and let P_n be the set of all vectors of t successive output values generated by the generator, from all its initial states (i.e. over all its cycle). If the generator is designed so that P_n covers the unit hypercube more evenly than random points, it appears plausible that Q_n could be a better approximation than Q_n obtained by random points."

Q_n is here the approximation of an integral using a Monte Carlo method. The idea of quasi-random numbers is not new and will be discussed later.

A measure of the evenness within the n-dimensions is given by a metric called discrepancy. This measures the distribution of the number of points in predefined cells shapes in the n-dimensional space. This concept is explained further by L'Ecuyer (2000).

As previously discussed the development of these tests has been alongside a development of random number generators.

3.3 METHODS OF SUPPLYING THE STREAM OF RANDOM NUMBERS

3.3.1 Physical Devices and Tables

A brief history of the creation of random numbers is given by Bennett (1998) and Knuth (1998). In the early days of computing, devices based on natural random processes were proposed to supply the required random numbers (Tocher 1960).

Knuth reports that,

> "the Ferranti Mark 1 computer, first installed in 1951, had a built-in instruction that put 20 random bits into the accumulator using a resistance noise generator; this feature had been recommended by A. M. Turing."

Tocher (1960) stated that he had never seen details of the mechanism nor of any description of any studies made using it. He went on to state no further Ferranti machine had incorporated such a feature.

Details of a number of such natural random process devices are given by Tocher (1960). However the devices were found to be too slow. Tocher reports the device used by the RAND Corporation produced and printed random decimal digits at the rate of one a second. Instead tables of random number were published (for example RAND Corporation, 1955).

These tables (usually also available on tape) would then be made available for use in the computer based simulation model.

The early researchers found even accessing tables of data (possibly held on a magnetic drum) extremely slow, and arithmetic methods of generating random numbers, or to be more accurate "pseudorandom numbers", were sought (Tocher 1960). Two of the criteria that were used then, and are still considered important, for selecting a suitable method were speed and cycle or period length. Since computer have a limited set of numbers, all arithmetic processes used to produce random numbers will repeat. Too short a cycle was considered undesirable as it was considered that the experimental results would be questionable if the same set of random numbers were repeated. Due to the high cost of fast memory they also had to consider the size of the code.

Tocher states that there are advantages in using arithmetic process for generating "pseudorandom" numbers as the sequence is repeatable and experiments can therefore be reproduced.

The developments within data processing that have enabled large amounts of data to be quickly read, have lead to resurgence of the idea of tables. Marsaglia has produced a CD-ROM containing what he intended to be "an unassailable source" of real random numbers. This he created by,

> "..a combination of several of the best deterministic random number generators (RNGs), together with three sources of white noise, as well as black noise (from a rap music digital recording)" (Marsaglia 2002).

There are a number of web sites that offer tables that can be downloaded. Such a web site has been created by Haahr (1999). It produces a random stream of bits by tuning into a frequency where nobody is broadcasting. Many of the bits are discarded and then the remaining stream is amended to remove

any bias and is regularly tested by a measure based on entropy (Walker 1998). The site, as well as allowing real time sampling of the stream, also provides random streams created earlier. The real time speed of creation in 2002 was only 8 KB of raw data a minute but a faster version is planned (Haahr 2002).

For discrete-event simulation the most viable source at the present time is still considered to be one based on an arithmetic process and no specialised simulation software package was found that did not have one as its standard RNG.

3.3.2 Linear Congruent Generators

The most commonly used arithmetic based random number generators are those based on the congruential generator. The most popular, by far, is the linear congruential generator (LCG). This method, exploiting number theory, was first proposed by Lehmer (1951).

There are now a number of formulations of the generator. A definition embracing these formulations is:

$X_n \equiv (X_{n-1}a + c) \bmod(m)$

where X_0 is a seed given by the user

$c \geq 0$

if $c = 0$ $X_0 > 0$

if $c > 0$ $X_0 \geq 0$

$a >> 0$

$m > a$

The term r_n is the nth random number.

if r_n is not to take the values 0 or 1

$r_n = X_n/m$ $\quad c = 0$ or

$r_n = (X_n + 1)/(m + 1)$ $\quad c > 0$

if r_n is not to take the value 0 but can take the value 1

$r_n = X_n/(m-1)$ $\quad c = 0$ or

$r_n = (X_n + 1)/m$ $\quad c > 0$

if r_n is not to take the value 1 but can take the value 0

$r_n = (X_n - 1)/(m-1)$ $\quad c = 0$ or

$r_n = X_n/m$ $\quad c > 0$

if r_n can take the values 0 and 1

$r_n = (X_n - 1)/(m-2)$ $\quad c = 0$ or

$r_n = X_n/(m-1)$ $\quad c > 0$

The usual selection of c and calculation of r_n has:

$$c = 0$$

$$r_n = \frac{X_n}{m}$$

With such a choice it is obviously not advisable to have a seed of zero.

This general form is usually referred to as the "mixed linear congruent generator" or sometimes "multiplicative linear congruent generator", with the abbreviation MLCG, to distinguish it from the linear congruent generator (LCG) where there was no c or "offset" term. In this study the abbreviation MLCG will be used for both. Fishman and Moore (1986) have made an exhaustive search for the best set of mixed linear congruent generators with a modulus of 2^{31}-1.

L'Ecuyer (1999) produced a table of moduli and multipliers with good performance in the spectral test. Knuth has published on the Internet (Knuth 1999) that the multiplier 2650845021 obtained by Killingbeck after an exhaustive search of all multipliers of the form $a \equiv 1 \bmod 4$ for an MLCG based on a modulus of 2^{32} performs very well. Knuth went on to state "its spectacular μ scores (3.61, 4.20, 5.37,8.85, 4.11) exceed all values μ_2, μ_3, μ_4, and μ_5 for any modulus in the entire table." The term μ is, as described earlier, Knuth's metric of merit that he uses when applying the "spectral" test.

An extension of the MLCG with more than one multiplier is the Multiple Recursive Generator (MRG). A k^{th}-order MRG has the form:

$$X_n \equiv (a_1 X_{n-1} + \ldots\ldots\ldots\ldots a_k X_{n-k}) \bmod m$$

Kao and Tang (1997) searched for the best MRG with a modulus of 2^{31}-1 and selected a third order MRG. (Unfortunately the abbreviation MRG is also used for random generators created by combining MLCGs!)

3.3.3 Serious Concerns with the Linear Congruential Generator

Confidence in the linear congruential generator was seriously shaken by the paper "Random Numbers Fall Mainly in the Planes" by Marsaglia (1968). In this paper he proved that Lehmer's random number generator (MLCG) produced results with a "crystalline" structure. That is the points, created by sets of m successive "pseudorandom numbers", lie on a limited number of hyperplanes.

Marsaglia gives the example of using a modulus of 2^{32}. Fewer than 2,953 hyperplanes will contain all the 3-tuples. (3-tuples are the points created by three sequential "pseudorandom numbers" in a three dimensional space.) If the numbers were truly random Marsaglia stated that it could be theoretically shown

(but not in his paper) that the smallest number of planes that would contain all the points is about 10^8. Based on this, Marsaglia stated that MLCGs were unsatisfactory.

He restated this view in 1985 (Marsaglia 1985) and introduced his more stringent tests, which MLCGs fail.

Knuth (1998) does not share Marsaglia's view. He states

> "At first glance we might think that such systematic behaviour is so non-random as to make congruential generators quite worthless; but more careful reflection, remembering that m is quite large in practice, provides a better insight."

He goes on to state that if we take truly random numbers between 0 and 1 and round or truncate them (he did not give an indication to what degree of truncation he meant) then we would obtain an extremely regular structure when viewed under a microscope. One must imagining he is considering plotting the points, following Marsaglia's scheme, in space and that he is using a scale that provides a dense mass of points such that a microscope is needed to see the structure! Ignoring this imprecise imagery it is clear that Knuth's view is that MLCGs with hyperplanes that are close together are acceptable (His measure of acceptability is that the hyperplanes are close together that is the generator passes the so called spectral test, which measures the distance between the hyperplanes). Knuth defined the reciprocal of the distance between the hyperplanes as the "accuracy". Thus he states that the larger the accuracy the "better" is the generator. Here "better" means that the lattice structure is finer and thus the objection raised in Marsaglia's paper has less validity. Knuth also has a metric of merit μ, as discussed earlier, for evaluating if a generator passes the "spectral" test. James (1994), when criticising Marsaglia for apparently accepting MLCG as "still pretty good", stated, "I hold that a random

number generator which unknowingly gives you an incorrect result is not only bad, it is catastrophic."

During this search of literature, no published paper was found that stated an incorrect result had been obtained, or bad decision had been made due to the poor random number generator used in a real life discrete-event simulation study. This indirectly gives support to Knuth's view, but later in this thesis an account of a "biased" result in a computational physics study is discussed. Also it may be considered that the lack of such reports of incorrect results within discrete-event simulation maybe due to a reluctance to report such an error or simply that the error would not have been detected, as feared by James. It is also important to note the criticisms have often come from users of random numbers in application not connected to discrete-event simulation, for example James's interest in random numbers appears to be with Monte Carlo Method calculations within computational physics.

3.3.4 Alternative Psuedo-Random Number Generators

MLCGs are still widely used and marked improvements in the values of *m*, and *a* have been achieved, but there is a drive to create generators without their weaknesses.

To examine every type of RNG ever suggested is a large task and is beyond the needs of this thesis, as only the more significant RNGs are likely to be selected and therefore only they need to be considered. However of the many alternative RNGs that have been developed, and are being developed, they may be classified into two main forms:

- Fibonacci and Lagged Fibonacci Sequences
- Shift-Register Generators

It is also true that improvements in already existing RNGs have been made by combining RNGs.

These will now be discussed.

3.3.5 Fibonacci and Lagged Fibonacci Sequences

An early alternative method of obtaining "pseudorandom numbers" (Knuth 1998, suggested the method was considered in the early 1950s) was by using the sequence of numbers of already generated. That is:

$X_{n+1}=f(X_{n-m}, X_{n-(m-1)}, ...,X_n)$

An example of such a "Fibonacci Sequence" is

$X_{n+1} = (X_{\Omega}+ X_{n-1}) \bmod m$

In order to improve the performance in the test for randomness the sequence was "lagged" i.e.

$X_n=(X_{n-r}+X_{n-s}) \bmod m, \quad n>r, n>s$

Knuth (1998) states that values of r and s (the lags), as suggested by Mitchell and Moore (unpublished), of 24 and 55 respectively, and m having an even value, are particularly good values.

Marsaglia and Zaman (1991) state that all "additive" Fibonacci and Lagged Fibonacci sequences fail the "rigorous tests" (tests suggested by Marsaglia 1985) and although they stated that the "multiplicative" form:

$$X_n=(X_{n-1} \cdot X_{n-r}) \bmod m$$

passes these stringent tests, Marsaglia and Zaman (1991) suggested a new class of generator.

This new class of generator consisted of improved versions of the "additive" Lagged Fibonacci sequences. In these improved versions, the carry or borrow bit is used to amend the next number in the sequence. (They appear to have followed this development as they wished to avoid the slower operation of "multiply"). These new methods are called "Add with Carry" and "Subtract with Borrow" or "ACSB" (Marsaglia and Zaman 1991). The "Add with Carry" is used when the numbers in the function are added and the "Subtract with Borrow" is used when the numbers are subtracted. The terms come from "schoolboy" arithmetic.

These generators (ACSBs) found favour in applications where a large number of random numbers were required and speed of generation was important. Ferrenberg et al. (1992) have reported that there have been problems with this generator due to "long-term autocorrelation". When the generator was used in their Monte Carlo studies in theoretical physics, the results were found to be biased.

Ferrenberg et al. found that an old congruential random number generator, originally proposed by Lewis et al. (1969) was more precise in its estimates.

Lüscher (1994) states that to avoid these long-term autocorrelation problems, both in ACSBs and other RNGs, 24 random numbers should be generated then the next (p-24) should be discarded. The value of p may be changed to give

different level of protection against autocorrelation. This has been implemented into a computer program (RANLUX) by James (1994), where the default value of p is 223 and the highest selectable value is 389. The random generator used by this implementation was the original "ACSB" of Marsaglia and Zaman. [The implementation, by James, of this original "ACSB", without the discarding of "pseudorandom numbers", was called RCARRY (see Lüscher 1994).]

This stratagem of throwing away the majority of the created sequence is one reason for seeking longer cycle lengths.

Further to the problems of long-term autocorrelation found when using the "Add with Carry and Subtract with Borrow" generators, Tezuka et al. (1993) and Couture & L'Ecuyer (1994) showed that these RNGs also produced lattice structures and a Mixed Linear Congruent Generator could be found that would produce the same random number streams.

Knuth (1998) was thus able to include one of these ACSB generators in his tables of "spectral" test results. At the time of publication of Knuth's book (1998), the "ACSB" RNG gave the highest value of "accuracy." Since then Knuth has, as previously stated, declared that using the "spectral test," the random number generator of Killingbeck out performs all the generators in his table 1 (page 106 of Knuth 1998), including the "ACSB".

3.3.6 Shift-Register Generators

A popular alternative RNG is the Shift-Register Generators (see Golomb 1967), of which a well-known form is that suggested by Tausworthe (1965). Here the random sequence is a stream of bits that are generated by a logic operation (e.g. XOR) on earlier portions of the bit sequence. Knuth (1998) states a commonly considered method to create a random number from the bit stream is to consider the bits b_1, b_2 simply concatenated thus:

$X_n=(0.b_1b_2.....)_2$

But Knuth states that such random numbers do not pass the necessary statistical tests even if the individual bits are random.

The method of using logic operations on the string of bits has been improved by including register shifts and by using a number of streams of bits that are then combined to form a single number. In more complex implementations the logical operations are also performed across the streams. This is termed "Twisting".

The apparent need for computational physics to have uniform density of large numbers of random numbers in multi-dimensional space has led to a development of a spectacular "twisted generalised feedback shift register" (TGFSR) pseudorandom number generator called the Mersenne Twister (Matsumoto and Nishimura 1998). (Mersenne is the name of a prime number of the form 2^n-1). This "pseudorandom number" generator has 624 bit streams and thus requires 624 seeds. To create the seeds another random number generator is used. The FORTRAN version of the Mersenne Twister (Takano 1999) uses a random number generator suggested by Marsaglia (line 15 in Table 1, Knuth 1998) to generate the 624 seeds from a single seed. There have been reports that in an earlier version of the generator there was a problem with certain seed selection (Mersenne Twister Home Page, 2002). This has led to a rejection of some of the earlier methods of producing the seeds.

3.3.7 Combining Existing RNGs

One method of improving on the performance of a single random number generator is to combine two or more random number generators. A number of writers have claimed success in combining RNGs. For example L'Ecuyer (1988), when wishing to create an acceptable random number generator for a

16-bit computer, combined three multiplicative linear congruent generators to form a generator with a much longer cycle then the three original MLCGs (increasing from approximately 3×10^4 to 8×10^{12}). Knuth (1998) however states that at present the theoretical basis for such combinations seems complex and obscure, and he warns of the danger that if the choice of random number generator is based on an inability to understand, then it may be considered that the random number generator is itself being selected at random. Knuth concludes sagely, that the choice of a random number generator should never be made at random.

3.4 ALTERNATIVE METHODS OF SUPPLYING "RANDOMNESS"

In the field of Monte Carlo Methods there have been developments where the "randomness" has been replaced by specially constructed sequence of numbers called "quasi-random numbers" (see for example Niederreiter 1992). This has been shown to be more accurate in evaluating integrals (Niederreiter 1978). In the area of simulation, Brenner (1963) and Saliby (1990a) both proposed methods of constructing a sequence of random variates of fixed length, having a perfect mean and distribution. Their methods are very similar. A description of this method will be described later when it is tested. This method has not been widely used. Pidd (1992) suggested it should be given more attention. When Saliby presented his method to the Simulation Study Group of the Operational Research Society in 1988 it was heavily criticised as being contrary to "well known good practice." Kleijnen and van Groenendaal (1992) rejected the method as they considered it would give biased results.

3.5 CYCLE LENGTH

One of the driving forces to develop new generators is to increase the cycle length. One of the reasons for requiring a large cycle length is to accommodate the suggestion to reject large portions of the generated sequence. This

suggestion is not only made for "Add with Carry and Subtract with Borrow" random number generators when long-term autocorrelation was being removed but there is also a wish to reject most of a MLCG sequence (L'Ecuyer 1994) since it is claimed that MLCGs are too evenly distributed (Niederreiter 1992). However does not appear to be the main reason for requiring longer cycle lengths. The main reason seems to be to meet the needs of those studies that require millions, or as suggested by Marsaglia and Zaman (1991) billions, of random numbers. Such requirements do not arise from normal discrete-event simulation. As discussed below the cycle lengths discussed seem rather extreme.

L'Ecuyer (1994) discusses that the period of 2^{20} of a MLCG with a modulus 2^{32}, "can be exhausted in a few minutes of CPU time of a small workstation" and that a period length of "at least 2^{60}" is the minimum now required. This seems excessive since even though a modest workstation can now in 2003, create 2^{20} (approximately one million) "pseudorandom numbers" in not a few minutes, but one second, it would still take nearly 36 thousand years to create 2^{60}.

Kao and Tang (1997) state that their generator has a cycle of 2^{93}. If it only takes one second to create 2^{20} then the Kao and Tang generator would take over 3×10^{14} years (this is 30,000 times the life of the Universe so far) to run through the whole cycle! The Mersenne Twister generator has a cycle length that is a vast number of orders larger ($\times 10^{5952}$) than the one of Kao and Tang!

3.6 SEED CHOICE

All the arithmetic methods require one or more numbers (seeds) to be provided by the user to start the process of creating "pseudorandom numbers". The MLCG require a single number. Using the lags suggested by Mitchell and Moore, their lagged Fibonacci series would need 56 seeds and as previously stated the Mersenne Twister generator requires 624.

If more than one seed is required a MLCG can be used to generate them. In such a situation, the user has only to supply one seed (as in the FORTRAN version of the Mersenne Twister written by Takano 1999).

The literature (for example: Knuth 1998, Law & Kelton 2000 and Fishman & Moore 1986) reports that the choice of seed is not important. Fishman (1978) gave tables of "pseudorandom numbers" created 10,000 apart. These were to be used for seeds that would then have sequences that did not overlap, unless more then 10,000 samples were required. There is no suggestion that they would perform any better than any randomly chosen seed, just that the sequences would not overlap.

In reporting results produced during the testing of RNGs, the seeds used are not normally stated. However Marsaglia (1993), when discussing his Monkey tests, does state the seed he used in a set of tests. It was always 1234567! His choice of seed may be considered as indicating he did not consider the value of the seed important.

As previously noted, it has been recently reported that the Mersenne Twister has produce very non-random behaviour with poor selection of seeds. This at present has not been fully discussed.

A few authors have stated that the choice is important and a number of workers in the field, who unfortunately have not published their findings, have stated that they have had very poor results when certain seeds were used. Two such internal reports are Hollocks (1966) and Robson (1970).

It is however reasonable to state that the major writers have the view that the choice of seeds is not important. The lack of any evidence that this is true, especially for the smaller samples sizes used in discrete-event simulation, indicates this must be tested.

3.7 CONCERNS WHEN CREATING RANDOM VARIATES FROM OTHER DISTRIBUTIONS

Within simulation models, "pseudorandom numbers" are not usually used as such but only after transformation to produce a sample from a particular statistical distribution (such as Normal or Negative Exponential) with the parameters (e.g. mean) supplied by the user. Using the pseudorandom number string produced by MLCGs to produce random variates from other distributions has led to another problem. Ripley (1987) states that Nieve while using the modified Box and Muller polar method of producing normal deviates (the essential point is that this method uses pairs of sequential "pseudorandom numbers") found that when he plotted the nth value against the n+1th value he obtained a spiral pattern. Ripley points out that Nieve used a poor MLCG but the concern still exists that even with a sequence of what would otherwise be considered "perfect" random numbers, the random variates created by a particular transformation procedure using a certain RNG may be unacceptable since they may not be a representative sample.

An example of a transformation where the random deviates would be unacceptable even if the random numbers were "perfect" was the method of producing Normal deviates from a stream of random numbers used by the Subroutine Gauss in IBM's Scientific Subroutine Library for the IBM 360. (IBM 1970) The method used by the subroutine consists of adding a sequence of 12 random numbers then subtracting 6. The result is an approximation to normal distribution with a mean of 0 and a variance of 1. However the approximation is poor in the regions of the "tails".

For this thesis combinations of RNG and transformation must be considered. Thus when deciding on the method of producing a random input into a

simulation, the actual sample values need to examined rather than the random numbers themselves.

3.8 SUMMARY OF THE LITERATURE ON METHODS OF INTRODUCING RANDOMNESS

There are a number of "pseudorandom" number generators available. All appear to have been criticised and rejected by theoreticians. This includes the most popular generator MLCG. Even one of the most recently suggested RNG, the Mersenne Twister, has already been criticised and the authors themselves have stated that it is unsuitable for one particular application, that of cryptography (Mersenne Twister Home Page, 2002).

Since no attempt appears to have been made to match the statistical tests to the actual requirement of discrete-event simulation, the validity of the criticism was uncertain.

The move to long cycles beyond any practical requirement is of concern as there was some evidence from research made concerning the effectiveness of Monte Carlo Methods that smaller cycle lengths may give more accurate results.

Methods of producing sequences of random variates with perfect mean and distribution have been described but have no extensive use. A concern is that the method appears to require the sample size to be known.

The alternative of using "unassailable real" random numbers is a possibility but they are still slow to create.

The general advice from most writers is that all generators should be tested for suitability for the intended purpose and as already stated Knuth (1997) recommends testing the actual sequence to be used. This is not very helpful as there is little guidance on what test would be relevant for what simulation tasks. Some of the tests need large samples and thus the sequences actually going to be used in the simulation cannot be used in these tests, as they are too short.

It is also known from the literature that there exists a concern that the actual sample may be unacceptable even if the random number generator is acceptable. Thus testing the samples rather than the sequence of random numbers seems to be important but this has not been reflected in the literature.

As previously discussed, the general view that the seed selection is unimportant, needed to be investigated.

To meet our requirement for a simple method of providing randomness into the simulation to accurately reflect real life, it was necessary to determine:

- What RNGs provide the quality samples required?
- Can MLCGs still be used?
- Is the choice of seeds important?
- Can Descriptive Sampling out perform RNGs?
- Is our choice of a RNG or Descriptive sample affected by the need to forecast sample size?

The first concern was how to determine if a sample is a quality sample in terms of the needs of discrete-event simulation.

3.9 SELECTING THE INITIAL CONDITION AND WARM UP PERIOD

3.9.1 Terminating and Non-Terminating Simulations

In the literature when discussing such factors as the initial conditions required to commence measurements, and run length, a number of writers have found it useful to define two types of simulation study. These are normally referred to as, terminating and non-terminating simulations. Law and Kelton (2000) defined the two forms such that a terminating simulation is one where a 'natural' event specifies the length of each run (e.g. the end of a day) while no such event exists for a non-terminating simulation. Law and Kelton consider that with terminating simulations their initial conditions and run length are defined by the description of the problem. Thus the selection of a suitable initial condition and run length is not a problem. The remaining problem in this situation is how many replicates are needed to obtain the required accuracy. Law and Kelton (Page 515, 2000) offer the following rule of thumb, "Regardless of the cost per replication, we recommend at least three to five replications."

In this research the non-terminating simulations are being considered. Here methods of determining when to start recording, how long to record and number of number of runs need to be evaluated.

3.9.2 The Problem of Setting the Initial Conditions and Determining the Warm Up Period

Since many stochastic processes will have a degree of autocorrelation, the state of the model when measurements commence will affect the early readings in the run.

Pidd (1992) states there are two acceptable states for initialising the measurements:

- A representative state
- The state after a warm up period.

Wilson and Pritsker (1978) examined two systems, a single server queue with a capacity of 15 and a machine-repair system with 3 men and 14 machines, to determine from the available procedures what was the best setting for the initial condition when measurements would start to be made. They concluded that the best initial condition was the mode of the "steady state" system. This is in line with Pidd's representative state. No paper was found that determined a general procedure to obtain a representative state.

Most writers only discuss "the warm up period" method of setting the initial condition. This may be due to the difficulty of defining or even setting the model to a representative state.

Since in many stochastic systems the behaviour of the system is independent of the initial condition when "steady state" is reached, it is frequently considered that the warm up period should only end when the system has reached a "steady state". Indeed most writers consider that, for non-terminating simulations, the "steady state" values are required to describe the performance

of the system. They suggest that inaccurate results will be obtained if measurements are taken outside of this "steady state" condition.

Having a warm up period that enables the system to reach "steady state" and only starting the measurements at that point in time, appears to be commonly accepted as meeting the two requirements of starting from a representative position and taking measurements only at "steady state" conditions.

The detection of reaching "steady state" is not easy. Conway (1963) has pointed out that there are difficulties in determining when the warm up period has finished and when measurements should commence and Gafarian et al (1978) found that none of the methods of detecting steady state available in 1978 performed well.

Welch (1983) has suggested a graphical method but it is both time consuming and expensive in resources. (Law and Kelton 2000 state on page 522 "The major difficulty in applying Welch's procedure is that the number of replicates .. may be large if the process ... is highly variable" and on page 521, they state, "m should be much larger than the anticipated value of l", where m is the run length of each replicate and l is the time to get to "steady-state".) Hollocks (1995) reports that graphical methods are used for determining warm-up periods and some specialised simulation packages (e.g. AutoMod) offer automatic calculation of moving averages (see Wadsack and Tobias, 1994), however the time pressures reported both by Hollocks and by Wadsack & Tobias would question the common use of this technique in its correct form due to the resources required.

Law and Kelton (1983) developed a method for determining the length of the warm up period and the total simulation run length in order to get the best results. They state however that it depends on the expected value of the characteristic being measured (which varies over simulated time), monotonically converging on the long-term expected value (or mean of the steady state

system). Kelton (1985), Kelton & Law (1985), and Murray & Kelton (1988) demonstrated that for some simple queues this assumption is violated if the simulation started with certain initial queue sizes.

Kelton and Kelton & Law, from an analytical analysis of the expected queue size of simple queues, considered that the "steady state" was obtained quickest, and therefore the greatest accuracy could be obtained from a certain total run length, by starting with relatively large queues. Here they were considering that the steady state condition was not known and the system could not be set to such a condition due to lack of knowledge. They did however consider that there might exist a starting condition that would require the shortest warm up period to enable the model to reach steady state conditions and thus the measurements to commence. Thus they introduced the concept of two initial conditions: one to start the model and one to start the recording. They, and many other writers, assumed that the "steady state" is reached when the expected value of queue size is steady.

The concept that more accurate results will be obtained by discarding the initial readings is contradicted by the analytical result of Blomqvist (1970) who showed that for a whole range of types of simple queues, the accuracy of the measured expected queue size was greatest when none of the results are discarded. It should be noted that since it was a simple queue, with no delays before the first part arrived to be processed, the initial condition can be considered acceptable as a representative condition. That is no warm-up period to fill the system was required.

The two sets of studies however used different measures of accuracy. Kelton, and Kelton & Law used the absolute difference between the expected queue size and the steady state average queue size, while Blomqvist used the mean square error of the expected measured queue size from the steady state queue size. Other writers (for example Kleijnen 1974) consider that Blomqvist's result was only true for very long computer runs or they suggested that using mean

square error is inappropriate (Kleijnen and van Groenendaal 1992) but they offer no quantitative measurements. Blomqvist result does assume that the length of run is long enough as to get accurate results. Previously it had been shown that to get accurate results it was necessary to have lengthy runs (see for example Daley 1968). The minimum run length required by Blomqvist analysis is such that a point is reached when the accuracy of measurement is increasing with run length. For short experiments with certain starting queue sizes the accuracy initially reduces (See Graph 7.5 page 180)

Other methods have been suggested that use a single run to detect the "steady state" thus avoiding the use of resources required by the graphical method of Welch. Goldsman et al. (1994) suggested a method that incorporates earlier methods of Schruben (1982) and Schruben et al. (1983). The methods consist of batching the values of the output of the simulation and comparing estimates of their variances. If the variance of a batch is statistically significantly different to the variance of a later batch, then it is considered still to contain a bias from the initial conditions. There is obviously a problem in determining if the variances are statistically different. With the assumption of large batch sizes the standard F test may be applied. There are other similar techniques all attempting to provide estimates of the variance of the batches but require, according to Goldsman et al., more complex statistics.

3.10 SUMMARY OF THE LITERATURE ON SELECTING THE INITIAL CONDITION AND WARM UP PERIOD

Although there is no doubt that there needs either to be an initial setting to a realistic condition, either by direct intervention or by a warm up period, the need to obtain "steady state" is not commonly agreed. If it is accepted that a "steady state" has to be obtained, a widely accepted method that does not requires heavy use of resources to determine the required warm up period has not yet been found.

Clarification and a practical solution are required in determining how to set the initial condition and when measurements should commence in order to use the minimum of resources whilst obtaining the required accuracy.

3.11 SETTING OF THE DURATION OF A COMPUTER RUN

In order to obtain the required degree of accuracy the computer simulation could be run for a single long run or a number of shorter runs. There is disagreement on what is the best policy.

Most of the literature assumes that the simulation is set to run for a predetermined period and greater accuracy is obtained by repeating the experiment with identical design choices (replicates) but with different random number streams (see for example Lanner Group 1998). This is in line with the statistical designs of Fisher (1960). Such a design has a large body of supporting statistical literature. The results from the various runs can be considered as independent estimates.

In most statistical experiments, such as agriculture field trials or drug trials, there is limited scope in increasing accuracy by prolonging the duration of the experiment. (Replicates are also used so as to have a full range of values for the factors not able to be controlled in such experiments, another feature not relevant to computer simulation where the user inputs all "randomness".) Terminating simulation also cannot be extended beyond the natural conclusion. However in non-terminating simulation this is not the usual case. The only physical limit on a single computer simulation run for non-terminating simulations is the available computer time and the cycle length of the random number generator.

It is simpler however to perform a single lengthy run than to construct replicates. Replicates require different seeds for the random number generators or the result would be identical (Tocher 1960). Without the use of a simulation package this is difficult or time consuming to set up.

The discussions on whether the best design should be a number of short runs (with different seeds) or one long run in order to get an accurate estimate of the important variables is not conclusive. Whitt (1991) fails to clarify the discussion and only rules out a large number of very short runs. Whitt's analysis accepted the need for a lengthy warm up period and thus would be invalid if long warm up runs proved unnecessary.

Kleijnen and van Groenendaal (1992) state that their favoured approach for non-terminating simulations is to have a single long run with a number of readings spaced so as to give negligible autocorrelation.

Law and Kelton (page 528, 2000) have produced a summary of the various methods of constructing an experiment for a non-terminating simulation so as to get meaningful statistics.

They have identified six methods:

- Multiple Replication each with a separate warm up period
- Single run considered as a set of sequential batches with only one warm up period (spaced measurements)
- Autoregressive
- Spectral
- Regenerative
- Standardised Time series

Law and Kelton give detailed descriptions of the methods and suitable references. Only the first case has replicates. The other methods have a single

run. Since the application of these methods is not part of this research, the literature on these methods will not be discussed further.

The first two methods of the six, suggested by Law and Kelton, have been already mentioned. Law and Kelton state that the problem with the Multiple Replication procedure is there is no sound way of knowing what is a suitable run length and warm up period. The difficult in determining the set up period has already been discussed. Law and Kelton (page 537, 2000) state that no procedure in which the run length is fixed before the simulation begins can be relied on to give the required accuracy.

Law and Kelton state that the shortcomings with the spaced measurements is the difficulty in determining how long there should be between measurements. Although not stated specifically in the literature, this problem is equivalent to determining the warm up period required to reach "steady state", since the starting position of each batch (and thus the ending position of the previous batch) should be independent of the starting point of the previous batch. Otherwise the batches would be correlated. Thus the spaced measurement method may well be seen as a run of n+1 equal periods with a total length of n+1 times the length of the required warm up period with the first period discarded and the measurement at the end of the next n periods used in the analysis.

The remaining methods consist of fitting statistical models to the measurements. Here the general concern is how to determine if the statistical model is appropriate and how to measure any required parameter of the model. In summary all the methods have concerns and in the case of "Regeneration" are difficult to apply to a real life situation (see page 533 Law and Kelton 2000).

3.12 SUMMARY OF THE LITERATURE ON THE NUMBER OF RUNS REQUIRED AND THE NEED TO DETERMINE THE INITIAL LENGTH OF THE COMPUTER RUN

The literature supports the idea of a single run if there is a requirement for a lengthy warm up period. However the methods of analysis of the outputs is problematic. If there is no need for a lengthy warm up period then the Multiple Replicate procedures may be preferred as there are sound methods of analysis in the existing statistical literature.

The choice of a single or multiple runs both require an initial estimate of the run length of the individual computer runs. The only information on making the choice is that stated above by Law and Kelton that there is no reliable method and thus they suggest a sequential sampling method of determining when the run has been long enough. Even if their advice is taken there is still a requirement that the initial length of the first run must be long enough as to be representative of the performance of the system.

Hence a method of determining a run length that will be representative of the system and thus will have sufficient number of infrequent events and values in the tails of the distributions of the real life variables so that any measurement taken from the simulation will be valid.

3.13 CONCLUSION OF THE LITERATURE SURVEY

The literature survey fails to obtain clear guidance on the decisions:

- The choice of method of introducing randomness into the simulation
- Determination of the initial condition and warm up period
- The initial setting of the duration of the computer run.

The advice is confused, including contradictions, misdirections, and omissions.

To obtain the information necessary to make the decisions on what method will be selected, as previously stated, it was necessary

1) For selection of the method of inputting the required randomness it was necessary to select what method should be used to generating the samples by determining:

- What RNGs provide the quality samples required?
- Can MLCGs still be used?
- Is the choice of seeds important?
- Can Descriptive Sampling out perform RNGs?
- Is our choice of a RNG or Descriptive sample affected by the need to forecast sample size?

and for this, the first question was how to determine if a sample is a quality sample.

2) For setting the Initial condition and determining the warm up period it was necessary to devise a practical solution to determine how to set the initial condition and when measurements should commence in order to use the minimum of resources whilst obtaining the required accuracy.

3) To determine if there should be one run or many replicates a study was needed to determine which method, given a certain resource, provided the most accurate answer.

4) To ensure that no error arises from too short a computer run, a practical method of ensuring sufficient infrequent events and values in the tails of the distributions of the real life variables was required.

4. SELECTION OF THE SOURCE OF RANDOMNESS (CREATING THE TEST)

4.1 INTRODUCTION

As discussed in the conclusion of the literature review, in order to answer the questions

- What RNGs provide the quality samples required?
- Can MLCGs still be used?
- Is the choice of seeds important?
- Can Descriptive Sampling out perform RNGs?
- Is our choice of a RNG or Descriptive sample affected by the need to forecast sample size?

It is necessary to be able to determine if a sample is a quality sample. For this a suitable test needs to be selected and if necessary developed.

This is done in the next section.

4.2 REQUIREMENT FOR A DISCRIMINATING TEST TO SELECT HIGH QUALITY SAMPLES

The most straightforward conclusion from reading the literature is that no generator has been proved to be acceptable based on the concept that a generator should pass all the tests. The simplistic view of only accepting a generator that passes all the tests is however easily discarded. In Appendix 8 a test is devised which will fail any generator of the form:

$$X_n = F[X_{n-1}, X_{n-2}, \ldots\ldots, X_{n-k}]$$

where the nth term of a random number sequence is determined from k earlier values. It will be noted that such a sequence would need k seeds.

Such a test may be considered "contrived" but without a justification of applicability to a real situation all tests are "contrived". The policy of rejecting any RNG if it fails a particular test, can be seen to be seen to be unacceptable, as all RNGs will fail a number of tests. This is shown in a general proof given by L'Ecuyer (1998).

Thus attempting to create a new RNG that would pass all known tests was rejected and the RNG to be selected was to be found within those already developed.

It is equally true to accept a RNG that fails a test by simply ignoring the test result without any justification for such an action would seem to be foolhardy.

In the literature survey it was clear that the tests to decide if a RNG was acceptable were not aimed directly at the needs of discrete-event simulation but at testing for random behaviour. To perform a verification of any RNGs suitability for use within discrete-event simulations, the various tests first need to be evaluated in terms of their relevance to the actual requirements of discrete-event simulation. A procedure on selecting which tests are relevant appears to be as elusive as finding a random number generator that will be acceptable to everyone. For any test to be discarded it would need to be shown that there is no real life situation where the test would apply. Thus in selecting candidate RNGs their ability to pass the established tests is not irrelevant but their failure to pass certain of the more esoteric tests may be.

In this thesis the attempt to select a generator based solely on its ability to pass tests for randomness was abandoned in favour of an approach more akin to the testing of a product. That is a form of "Quality Control" was applied. In this approach the requirement is redefined as *the samples used in a discrete-event*

simulation will be those samples that pass a quality standard rather than to select a RNG based on its ability to pass test and to accept the samples it creates indiscriminately.

The transfer of the emphasis of the selection to choosing representative samples rather than RNGs leads to some samples from a particular RNGs being rejected whilst others are accepted. It may well be that many RNGs, even ones considered inferior may produce a number of "perfect samples" whilst a highly regarded RNG, (e.g. one passing the "spectral" test with "flying colours") may produce a number of samples that are unacceptable. If there were to be a selection of a RNG to be used in specialised simulation software, then using this methodology, it would be selected on its ability to create a larger percentage of quality samples than other RNGs.

Traditionally in applying "Quality Control" to screen parts in an industrial process, a "Go/Fail" test is often applied. In this quality control, a part is accepted if it passes the test and is rejected if it fails. To apply such a "Quality Control", in selected or rejecting samples created by a RNG a test must be applied. The ideal test is one based on statistics. Thus a discriminating test based on statistics is required to screen the very good samples from the others.

4.2 SELECTION OF A SUITABLE TEST

As previously discussed, the first priority is to make sure our source of randomness will enable the simulation analyst to get accurate results. In the previous section it was assumed that a good sample, one passing the test, would produce better results than a sample that is less representative of the true distribution.

As already discussed, the importance of a good sample was shown in the studies that led Saliby to suggest Descriptive Sampling (Saliby 1980). In this

work he demonstrated that the average queue size measured in a simulation was closer to that predicted by theory if the actual traffic intensity produced by the samples was used in the theoretical calculation rather than using the values of the underlying distribution from which the samples were purported to have been drawn. He went on to demonstrate that he obtained superior results by forcing the samples from the distribution to have the "correct" mean and standard deviation. Saliby demonstrated by this work and his later studies (Saliby 1990a) that if good representative samples are obtained then more dependable simulation results should ensue.

But large differences in the quality of samples can be obtained even with a RNG considered by many as acceptable.

As a demonstration, two samples of 1000 random negative exponential samples values were created using different seeds with same pseudorandom number generator (the method to convert pseudorandom numbers to negative exponential samples values is described later). The generator used to create the pseudorandom numbers was a widely used RNG described by Lewis et al. (1969).

Two samples were created using two different seeds. These seeds were determined by procedure described later. The samples were tested to see if their values followed the expected distribution. The measure of goodness of fit was made by using the Chi Squared Test (see for example page 42-47 Siegel 1956).

The following results were obtained:

Seed	χ^2 Value	Probability of χ^2 Value
64689	154.8	99.112%
39170	262.8	0.163%

Degrees of Freedom = 199

RNG used: Lewis at al. (1969) with a Log_e transform used to create negative exponential distribution samples.

Table 4.1: The χ^2 Values of 1000 Samples Using with Two Different Seeds

The results in table 4.1 clearly show that for these very carefully selected seeds, there is a significant difference in the ability for the two samples to represent a negative exponential distribution. Seed value 64689 give 1000 samples values which were distributed as expected, while a seed of value 39170 give a set of values which would be rejected at the 0.2% "confidence" level and be considered as not having come from a negative exponential distribution. (Rejecting at the 0.2% "confidence" level is very conservative and rejection would be certain in all usual circumstances, as the risk that the distribution was in fact a negative exponential distribution was equivalent to the risk that an event had occurred that had a chance of *1/500 or less* of occurring.)

If 1000 sample values were required for a single simulation run then the simulation analyst would not want to use a seed of 39170 with the Lewis et al. random number generator but he or she would be very satisfied with the sample if a seed of 64689 was used with that random number generator.

It is not only simulation runs that can be affected by samples that are considered far from the normal behaviour, real life studies can suffer from sampling problems.

Warn (1972) when collecting "real life data" for a simulation of a spare parts depot using a day long industrial engineering study covering the whole plant, was informed at the end of the day by the management of the depot, that the day chosen was most unusual. Such comments from the management of an area being studied are usual, but on investigation it was found to be the lowest demand on the depot for ten years!

Woodward (1961) when allocating at random the towns to be allocated to the research students taking part in a large study of the occupations of English towns' inhabitants, was horrified to discover that they had been allocated a large number of seaside towns, which would tend to have a high proportion of retired people.

In both cases the sampling was repeated. This is in contrast to the view of Hacking (1965) who, wishing to preserve the integrity of any table of random numbers, considered that any set of random tables should include a string of zeros and that he would not wish to exclude them, although he realised that such a part of the table should be avoided by anyone using the table for sampling.

The requirement for truly representative samples so that the results can be used with confidence is similarly true for discrete event simulation. The need for a test to ensure the sample is a quality sample was thus well established.

Thus it was helpful to restate the actual requirement as:

> " An RNG should provide, after transformation, a sample of stochastic variables that are truly representative of the real life variable both in value and sequence" (See also Saliby 1990a)

rather than the definition of L'Ecuyer (2001),

> "The aim of (pseudo)random number generators (RNGs) is to implement an imitation of the abstract mathematical concept of mutually random variables uniformly distributed over the interval [0,1] (i.i.d.*U*[0,1], for short)".

Although both definitions may select the same RNGs as acceptable, there is a strong possibility that a RNG that will produce satisfactory, or even the best, samples according to the first definition will fail the definition stated by L'Ecuyer.

In order that the results of the simulation should be accepted, the choice of the test to be used to perform the selection of quality samples should be such that it appears to an engineer as straightforward and completely relevant to his or her needs. Thus a test that evaluates whether the sequence of random numbers obeys some, if what in certain cases of the more complex tests may be considered by non-mathematicians, esoteric behaviour of true random numbers would not be acceptable.

From a standpoint of testing for random behaviour, Knuth's tests (1997) appear to be a very reasonable method of selecting suitable RNGs and many RNGs pass these tests. None of these tests, other than the "Equidistribution test" meet the requirement of being directly relevant. The "Equidistribution test" alone is insufficient. None of these tests were selected as the discriminating test.

Other possible candidates for a practical discriminating test were those where the sequences being evaluated are actually used in simulation models where the correct result is well known. The quality of each sequence is considered to be given by the accuracy of the results obtained. Saliby (1990b) compared the quality of samples created by Descriptive Sampling to certain other RNGs by such a method.

Use of such simulation models as quality measures is open to the criticism that the measurements made in any particular model may not be sensitive to certain errors in the sample. For example in two of the examples used by Saliby (1990b), the simple queue, and the "newsboy problem", there are such concerns. It is possible that the average queue length in a simple queuing model may be similar for a range of arrival patterns (see the expression for the average queue size for a queue with a general service-time distribution on page 99 of Tanner 1995) and that the "newsboy problem" is insensitive both to the form of the demand distribution (see Pearson 2000 for the expression of the optimum order quantity) and as Saliby noted, to the sequence of the random deviates. Thus a sample that would perform badly in another example may be selected by such a test. Such a method of testing for quality was thus rejected.

For this study a discriminating test that directly measures the quality of the fit of the sample to the required data distribution both in distribution of values and sequence, was used. The test chosen was a development of the simple test described by Tocher (page 47, 1960) as the "Yule's test".

4.3 SELECTION OF TARGET DISTRIBUTION AND TRANSFORMATION METHOD

Before describing the test, the selection of a target distribution and transformation method are made. This makes description of the parameters within the test easier to describe.

There is no obvious reason why one RNG would be acceptable for all distributions or that every *method* of transforming the pseudorandom numbers into sample values from other distributions will be equally as good (for a comprehensive description of methods of creating sample values for various distributions see Devroye 1986). For example, some methods of creating a

normally distributed sample value require two consecutive random numbers while other methods require only a single random number. Thus any generator where there is a correlation between successive pseudorandom numbers may lead to a poor sample of normal deviates using the first set of methods.

In this research only one combination of target distribution and method of transformation was considered.

Providers of RNGs within general software have a difficulty in that the future use of the routine is unknown. However in discrete event simulation the use is likely to be more limited and predictable. In one of the most, if not the most, popular software packages, WITNESS, (Lanner Group, 1998) there were, in 1998, 14 inbuilt distributions, namely:

Distribution Name	**Type of Variable**	**Description**
BINOMIAL	Integer	Binomial Distribution
INORMAL	Integer	Uniform Distribution
POISSON	Integer	Poisson Distribution
BETA	Real	Beta Distribution
ERLANG	Real	Erlang
GAMMA	Real	Gamma Distribution
LOGNORML	Real	Lognormal Distribution
NEGEXP	Real	Negative Exponential Distribution
NORMAL	Real	Normal Distribution
RANDOM	Real	Uniform Distribution between 0.0 and 1.0
TNORMAL	Real	Truncated Normal Distribution
TRIANGLE	Real	Triangular Distribution
UNIFORM	Real	Uniform Distribution over a stated range
WEIBULL	Real	Weibull Distribution

Table 4.2: Standard Distributions in WITNESS (Release 9)

Although it is easy for the modeller using WITNESS to add user-defined distributions, these appear to cover most of the design requirements of WITNESS users.

For the target distribution, the distribution of the major variable in the real life example was chosen. Breakdown, or machines unplanned stopping, is the most important disruption in the target real life example, which is a transfer line (Ladbrook 1998, Kouikoglou and Phillis 2001). Thus the important factors are the time between breakdowns and the time to repair.

Ladbrook describes how difficult it is to obtain accurate information. Some of the reasons he gives are:

For manual recording:

- Not all stoppages reported
- Exaggerated reporting of times
- Incorrect data input due to typing errors

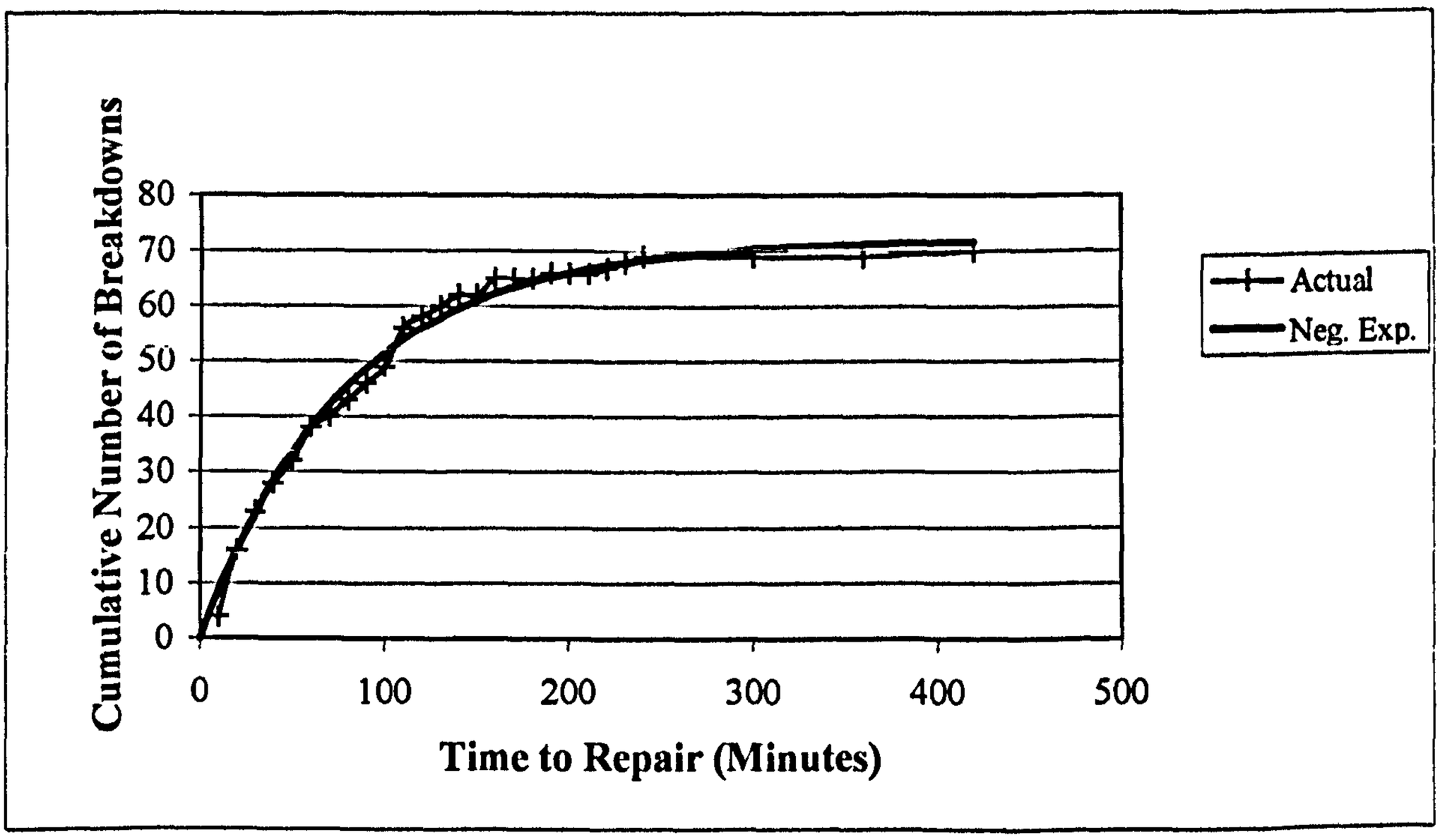
For electronic data collection:

- Data Collection Equipment not operating (repair completed outside normal working time)
- Try-outs (short runs to check repair) recorded as multiple breakdowns.
- Attempt to repair halted (e.g. over weekend, missing parts)

Ladbrook reports that it is usual for many companies to use the negative exponential distribution for both time between breakdowns and time to repair, even when the data that is collected does not appear to come from a negative exponential distribution. In his thesis, Ladbrook gives data for "operation 80" on a transfer line within the Dagenham Plant but does not attempt to fit a distribution to the data.

In this research, a negative exponential distribution has been successfully fitted to the data using Microsoft Excel (Microsoft 1994). The square error between each real life data values and the equivalent point on the theoretical cumulative curve was calculated, from which the total squared error was then calculated. Using the inbuilt optimising function 'Solver' to minimise the squared error by changing the value of the mean of the theoretical curve, a fit was obtained. The fit was then tested using a Chi Square test (see for example page 42 of Siegel 1956). It was found to be a remarkable good fit having a Chi Square value of only 3.6 for 79 degrees of freedom (Note: There is a 99.99% chance of getting a value of 34 or more by chance.)

The fit may be visually observed by examining graph 4.1.



Graph 4.1: Fitting of Real Life Data from Operation 80 to a Negative Exponential Distribution

Since the negative exponential distribution has such importance both within the real life model and in general simulation, it was chosen as the target distribution.

The method chosen for creating negative exponential variates using a random number was as follows (see for example Ahrens and Dieter 1972):

$$d_d = -\mu \ln(1 - r_n)$$

Where

d_n is the nth negative exponential sample

μ is the mean time to breakdown

r_n is the nth random number $0 < r_n < 1$

Since it is a precise inversion function the only disadvantage of this method is the need for a "log" to be calculated. Ahrens and Dieter stated that when programmed in FORTRAN, 72 percent of the time to convert the random numbers to negative exponential variates was required to calculate the "log". But as may be seen from the fact that it now needs less that 35 seconds elapse time to convert a stream of 10^{10} random numbers between 0 and 1 into 10^{10} negative exponential sample values, using a 600MH PC, that this is no longer a problem[§]. To put the number 10^{10} in perspective, if one was to simulate a system with 1000 random negative exponentially distributed processes, each with a mean time of only 10 seconds, that operated for 8 hour days and 5 day weeks, then with that many sample values the system to be able to be simulated for over 10 years operation.

[§] The Program used to measure the time taken was LNTIME. The listing is in Appendix 1.

4.4 DESCRIPTION OF THE TEST AND QUALITY MEASURE

4.4.1 Calculation of Test Variables and Test Statistics

The test is applied to single samples. An example of a "single sample" is the set of 1000 "sample values" produced when a section of 1000 sequential real random numbers are taken from the random number stream and converted into 1000 negative exponential sample values, or in the case of an RNG when 1000 psuedo-random numbers are created and converted into 1000 sample values of the negative exponential distribution. (A description of the terminology used is given in Appendix 5. This includes definitions of Sample, Sample Value, Test Variable, and Test Statistic.)

To perform the test, the test variables first have to be calculated. For this test there are several sets of test variables created from a single sample. Since one of the quality measurements required is that the individual sample values in a sample should appear to come from the correct distribution, the first set of test variables is the actual samples values themselves.

The sequence of the sample values is important and other sets of test variables are created so as to reflect the sequence. Each set of test variables is created by summing a fixed number of sample values. The sample values summed are sequential but the sequences do not overlap. By not overlapping the sample values when forming the test variables, the test variables created are then independent and thus able to be tested by the standard statistical tests that require such independence.

Thus after having formed the first set of test values from the actual sample values, the second set of test variables was created by summing the sample values in pairs, the third set of test variables was produced by summing threes and so on until ten sequential sample values were summed. (The number ten

was chosen as the default value and was used throughout this study but the computer programs allow the value of the maximum number of sequential values to be summed to be changed.)

N	Sample Value Number																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.39	0.70	0.74	0.63	0.45	0.65	0.49	0.85	0.46	0.48	0.94	0.83	0.99	0.82	0.52	0.67	0.38	0.50	0.40	0.53
2	1.09		1.37		1.1		1.34		0.94		1.77		1.81		1.19		0.88		0.93	
3	1.83			1.73			1.8			2.25			2.33			1.55				
4	2.46				2.44				2.71				3.00				1.81			
5	2.91				2.93				4.1				2.48							
6	3.56				4.05				3.88											
7	4.05				5.37															
8	4.9				5.71															
9	5.36				6.13															
10	5.84				6.58															

Where N is the number of sequential sample values used to calculate the test variable

Diagram 4.1: Illustrating the Non-overlapping Sums for the First Twenty Sample Values

Diagram 4.1 illustrates, for the first 20 sample values, the calculations of the sets of test variables. On line one of Diagram 4.1, is shown the individual sample values.

On line two, the sample values are summed in pairs, and on line three they are summed in triplets and so on until there are two values for the sum of ten sequential values.

If there were 1000 sample values then there would be 1000 single values, 500 pairs, 333 triplets, 250 sets of fours, 200 set of fives, 166 set of sixes, 142 set of sevens, 125 set of eights, 111 set of nines and 100 set of tens.

The test variables may be defined by the following expression:

For our sequence of N generated sample values and when the sums of sub-sequences of m sample values are being considered, then:

$V_{1,m} = S_1+S_2+...S_m$

$V_{2,m}=S_{m+1}+.....S_{2m}$

.....

.....

.....

$V_{k,m} =S_{mk-(m-1)} +.....S_{mk}$

Where:

$V_{i,m}$ is the ith. test variable for sequences of m sample values

S_i is the ith of the generated sample value

k is the largest integer such that $mk \leq N$

Having created 10 sets of test variables these are then converted to 10 test statistics. The test statistic chosen was χ^2. (This is pronounced chi-squared.)

This is defined in mathematical terms:

$$\chi^2 = \sum_{i=1}^{i=k}\left(\frac{\text{Actual Number in cell i - Expected Number in cell i}}{\text{Expected Number in cell i}}\right)^2$$

Where

k is the number of cells

$1 \leq i \leq k$

The definition of the cells is discussed later. The degrees of freedom for the statistic, when the mean is known, is the number of cells-1.

The distribution of χ^2 is known to be approximated by the chi-square distribution. The chi-square distribution for k degrees of freedom is the gamma

distribution with shape parameters k/2 and 2 (see Page 174 Arnold 1990). In statistics it is usual to name the test statistic, the distribution it follows and the test all by the same name. In this case it is confusing as the test statistic here is useful as a general measure of fit and it is necessary to have a quick method of referring to it. The convention being observed within this thesis is that χ^2 is used for the statistic calculated from the above equation, and chi-square is used for the distribution or variates drawn from the distribution.

The test is well known and has already been used in this study. (A simple "cookbook" description is given by Siegel page 42, 1956).

4.4.2 The Determination of the Number of Cells

The basic requirement for the Chi-Square test is to count the numbers of values of the test variables that fall within certain ranges. These ranges are termed the cells or categories. The Chi-Square test requires that the cell size be such that the *expected* number in a cell must be at least 5 (Chochran, 1954).

If the minimum value of 5 is used the maximum number of cells for the single values if there are 1000 sample values is 1000/5, that is 200. Thus the maximum number of cell for the different numbers of sequential sample values summed is shown in Table 4.3.

Number of Sequential Sample Values in the Test Variable	Number of Test Variables	Number of Cells
1	1000	200
2	500	100
3	333	66
4	250	50
5	200	40
6	166	33
7	142	28
8	125	25
9	111	22
10	100	20

Table 4.3: Maximum Number of Cells for the Chi Test
for 1000 sample values

The cell size in terms of the expected count was calculated so that the cells with the larger expected count are in the mid-values of the variables.

In the table 4.3 it can be seen that when 7 sequential sample values are being summed to provide the test variable, the 142 test variables have to be allocated to 28 cells. Thus the lower 13 values cells were dimensioned to have an expected count of 5, the middle two cells were dimensioned to have a count of 6, and the upper 13 have a size such that the expected count is 5. The use of integer values for the cell size is not essential. In this example, if non-integer values had been used, the cell size could have been identical with an expected count of 5.0714. However the use of integers made the testing of the whole system simpler.

The determination of the number of cells and expected number in each cell (as an integer) is performed by the program "SMALL". (See Appendix 1). The actual dimension of each cell is determined next.

4.4.3 Definition of the Cell Limits

In the computer programs the limits on each cell are stated as upper values. Consider the case where there are n cells. For cell 1, all calculated test variables with a value equal or less than the cell limit are counted in cell 1. The count in cell 2 is when the value of the test variable is greater than the limit on cell 1 but equal or less than the limit on cell 2. This is continued for all cells until the nth cell. The nth cell would not have a specified limit and the count in the nth cell is the number of times the value of the test variable is greater than the limit for cell n-1. Thus where there are n cells there are n-1 values of cell limits.

4.4.4 Calculation of the Cell Limits for the Single Sample Values

For this study, as the actual mean of the negative exponential distribution acts as a scaling factor, the mean has been taken here as 1. (The unit of time has therefore been standardised.)

As the unit of time is standardised and thus the mean of the negative exponential distribution is 1, the cumulative distribution of the negative exponential is then given by (see for example Law and Kelton 2000):

$$P(t) = 1 - e^{-t}$$

Where:

$P(t)$ is the probability of an arrival or completion of an event by time t and $t \geq 0$

The cell limits are calculated thus:

$$\begin{cases} C(k) = \sum_{i=1}^{i=k} E(i) \\ A(k) = -\log_e[1 - C(k)] \end{cases}$$

Where

$1 \leq k < n$

n is the number of cells

$E(i)$ is the Expected number of entries in cell i

$A(k)$ is the cell limit for cell k

A FORTRAN subroutine (FIXPB) calculates the values of n and k. The program SETUP calls this subroutine. They are listed in Appendix 1.

4.4.5 Calculation of the Cell Limits for the Test Variable created by the Addition of Sequential Sample Value

The distribution of a sum of k sample values from a negative exponential is the gamma distribution (see Arnold 1990). The gamma distribution has two shape parameters, α and β. The shape factor α has in this case the integer value k and since we are standardizing time, then β is equal to 1.

Thus

$$P(t) = 1 - e^{-t} \sum_{j=0}^{k-1} \frac{t^j}{j!}$$

The FORTRAN function GAMCUM (listing in Appendix 1) employs logarithms to calculate this function.

The limit for cell k, $A(k)$ is calculated from the expression:

$$\begin{cases} C(k) = \sum_{i=1}^{i=k} E(i) \\ C(k) = 1 - e^{-A(k)} \sum_{j=0}^{k-1} \frac{[A(k)]^j}{j!} \end{cases}$$

$$\text{Where } 1 \leq k < n$$

The value of $A(k)$ is determined by a search. In the routine SERGAM (in Appendix 1) this is accomplished by a binary search.

4.4.6 The Selection Criterion and Quality Measure

Using the expressions above, a set of cell definitions for each set of test statistics was calculated. To apply the test the sample values for a particular sample were used to create the 10 sets of test statistics and the number of values falling in the cells was determined. The χ^2 value for each set of test statistic was determined. For clarity the χ^2 value obtained when n sequential sample values are summed will be termed χ^2_n. If the fit were perfect the test statistic χ^2_n would have the value of 0. If the fit were poor then χ^2_n would be large. What is considered large is dependent only on the degrees of freedom, which in this case where the mean is known, is the number of cells minus one.

The sample is considered to be of a suitable quality if every χ^2_n value is less than its specific rejection level. Since each test statistic will have a different number of cells, and thus degrees of freedom, the rejection level is dependent on the value of n. The determination of the individual rejection levels is made later.

For ranking purposes a single metric was required so the sum of the χ^2_n values was used. However since the values of χ^2_n have different scales, the values of χ^2_n were multiplied by suitable weightings. The metric thus obtained was used as the "quality measure". It should be noted that the best quality sample has the smallest value of this quality measure.

4.4.7 The Computer Programs

The program SETUP is a housekeeping program that interactively allows the user to set the number of observations, the maximum number of consecutive sample values to be combined and the minimum number of observations required in each cell. These parameters have default values of 1000, 10 and 5. SETUP then runs the subroutines SMALL and FIXPB that in turn calls the subroutine SERGAM that uses the function GAMCUM. A file giving the specification of the cells and their limits is created for use in the tests. SETUP uses subroutines that are specific to the Exponential Negative distribution. If a different distribution were to be used then different subroutines would be needed.

SETUP needs only to be run if the number of observations, the maximum number of consecutive sample values to be combined, or the minimum number of observations required in each cell need to be changed.

A diagram of the programs and their listings is given in Appendix 1. All were programmed and were compiled using Microsoft FORTRAN version 4 (conforming to FORTRAN 77) for a computer with a "maths" processor.

The test is performed in two stages. In the first stage a computer program that was specially constructed for each source of random numbers is used. REALTST was developed for real random numbers. It is listed in Appendix 1.

It uses standard subroutines GAPC2 and CHICAL which in turn call subroutines INSERT and CHISQ. These standard subroutines are used in all the versions of the test and are independent of the source of the random numbers. The output from REALTST and the versions created for the other sources of randomness is a file where there is a record for each sample being tested. Each record has the sample's ten values of χ^2_n.

The second part uses a program SELECT. This selects the best samples and gives them a quality measure.

4.5 CALIBRATION AND TESTING USING REAL RANDOM NUMBERS

4.5.1 Introduction

In the section the file from REALTST will be used to verify the performance of its standard subroutines and then to calibrate the values that were later used by SELECT.

Although the subroutines used in REALTST and in the other versions of the first part of the test were tested at all stages of development a "systems" test was made by using real random numbers and determining the percentage of samples, for a certain value of n, have a value of χ^2_n greater than some critical value. Since χ^2_n approximately follows the chi-square distribution, the theoretical value can be calculated. Thus the actual values can be compared with the theoretical values and tested statistically to check the validity of the calculations.

As described in section 4.4.6, the operation of the method of rejecting the poorer samples, that is the "filter", that was used in SELECT compared all the calculated χ^2_n against a set of critical values. The setting of the critical values for the filter in SELECT is not amenable to a theoretical calculation so

"calibrating" using real random numbers was required. But first a source of real random numbers had to be obtained. The method of obtaining the real random numbers is described in Appendix 6.

4.5.2 Determining the Rejection Rates for Real Random Numbers

As previously discussed the first use of the test in the form of REALTST, was on the real random numbers obtained from the stream of random digits downloaded from "http://www.random.org" and extracted as described in Appendix 6. As already stated the output from REALTST was used to determine the percentage of samples that have for a certain value of n, χ^2_n greater than some critical value (or are "rejected"). Table 4.4 give the critical value of χ^2_n for certain "confidence" levels. Normal "confidence" levels of 0.1%, 0.5%, 1%, and 5% are given and also 95% and 50% values for reference and will be used later in constructing the filter.

Number of Sequential Sample Values n	Number of Cells k	Probability of χ^2_n Being Above the Table Value					
		95.0%	50.0%	5.0%	1.0%	0.5%	0.1%
1	200	167.36	198.33	232.91	248.33	254.13	266.39
2	100	77.05	98.33	123.23	134.64	138.99	148.23
3	66	47.45	64.33	84.82	94.42	98.10	105.99
4	50	33.93	48.33	66.34	74.92	78.23	85.35
5	40	25.70	38.34	54.57	62.43	65.48	72.06
6	33	20.07	31.34	46.19	53.49	56.33	62.49
7	28	16.15	26.34	40.11	46.96	49.65	55.48
8	25	13.85	23.34	36.42	42.98	45.56	51.18
9	22	11.59	20.34	32.67	38.93	41.40	46.80
10	20	10.12	18.34	30.14	36.19	38.58	43.82

Table 4.4: The Critical Values of the Chi-Square Distribution

If both the program is correct and the extract acts as if it contains real random numbers, then the percentage "rejected" should be equal to the "confidence" level. This reflects the weakness in statistical methodology that errors from rejection of correct distributions will occur.

This test using real random numbers is the same as an evaluation using a classical Monte Carlo method. In this case it is measuring the actual "confidence" level of each of the stated critical Chi-Square values. Since the Chi-Square distribution is a good approximation to the real case the estimate should be close. As with Monte Carlo studies it is possible to construct limits to our estimate.

These were calculated from the standard Normal approximation when the sample size is large (see Page 144 Arnold 1990):

$$\hat{r} = \bar{r} \pm \frac{3.08\sqrt{\bar{r}(1-\bar{r})}}{\sqrt{N}}$$

Where

$\hat{r}$ is the estimate

$\bar{r}$ is the measured mean

N is the sample size

From the four 10 Megabyte files available to be downloaded, it was possible to create over 40 million random numbers. This enabled a sample size of 40,000 to be used.

The tables AP7.1 to AP7.4 in Appendix 7 give the measured percentage of the samples "rejected" at the theoretical 5%, 1%, 0.05%, and 0.01% value. In all but two of the cases the measured rejection rate lies in the expected range forecast from the Chi-Square distribution. Those in which the measurement lies outside the expected range are marked with an asterisk. Since the "confidence" limit used to construct the estimate (the value in the formula above is 3.08) was 0.1% the probability of more than one or more measurements lying outside the expected range if there are 40 independent measurements is 3.9%. However the measurements are not independent. It should also be note that the Chi-Squared distribution is an approximation and the constructed limits were also based on an approximation.

Thus the hypothesis that the deviates are a random sequence of deviates from a negative exponential distribution is not rejected. It is also noted that no problem had been discovered in the previous testing of the real random numbers (Foley 2001).

This result gave the confidence that the program was correctly calculating χ^2_n values. The task of calibrating the filter for the simultaneous testing of the 10 χ^2_n values and using the filter to evaluating the ability of the real random number to create quality samples was then able to start.

4.5.3 Calibration of the Filter

As already discussed a method of determining the best samples was required. The sample, in order to pass the quality check, was required to be acceptable for all the test statistics. That is, the test was to be applied to all the sample's χ_n^2 values.

Since the χ_n^2 values within a sample cannot be considered as independent, a method of constructing a rule to be applied simultaneously to all the χ_n^2 values, based on statistical behaviour, is mathematically difficult. The most straightforward method is to construct a "suitable" rule and then to calibrate it using the results from real random numbers.

The proposed rule is to reject all samples that have one or more χ_n^2 value greater than some specified "confidence" level. For simplicity the "confidence" level was set to be the same for all values of n. The rate at which the samples constructed from the real random numbers "pass" is shown in table 4.5.

Number of Criteria Passed	"Confidence" Level					
	5.0%	1.0%	0.5%	0.1%	95.0%	50.0%
10	61.230%	90.143%	94.583%	98.853%	0.000%	0.195%
9	29.553%	9.198%	5.215%	1.140%	0.000%	1.475%
8	7.505%	0.608%	0.188%	0.008%	0.000%	5.620%
7	1.390%	0.043%	0.013%	0.000%	0.000%	12.375%
6	0.250%	0.005%	0.003%	0.000%	0.000%	19.498%
5	0.040%	0.003%	0.000%	0.000%	0.010%	21.970%
4	0.018%	0.003%	0.000%	0.000%	0.163%	18.758%
3	0.008%	0.000%	0.000%	0.000%	1.168%	11.955%
2	0.005%	0.000%	0.000%	0.000%	7.425%	5.648%
1	0.003%	0.000%	0.000%	0.000%	29.055%	2.045%
0	0.000%	0.000%	0.000%	0.000%	62.180%	0.463%

Table 4.5: The Measured Pass Rate of Samples of 1000 Real Random Numbers

The table demonstrate that if the common "confidence" level were set at 5% over 60% of the samples would pass at all 10 values of n.

If only the best samples are required the "confidence" level may be altered to the 95% or 50% levels. These would, if applied in a statistical test, lead to many erroneous decisions but here the aim is to develop a method to obtain the most representative sample.

The measured acceptance rate using these higher "confidence" levels is also given in table 4.5. At 95% most samples fail all the confidence levels and over 90% fail to pass eight of the ten levels. This is obviously too severe a filter. At 50%, 0.195% of the samples pass at every value of n. To get 100 best samples to pass the test there would need to be a total of 51823 samples. Thus for 60,000 samples there would be 117 samples expected to pass at all values of n when the level is the 50% level.

A confidence level of 50% was selected and it was expected that 60,000 samples would need to be examined in order to obtain 100 "highly representative samples".

4.5.4 Ranking the Samples

Any of the samples of 1000 values that pass the filter may be considered as a quality sample. But if the best samples, and in terms of RNG that require a seed the best seeds, are required then the samples need to be ranked. This also provides a means to determine which of the various methods of introducing randomness gives the best representative samples.

The method of ranking that was chosen was by producing for each sample a single quality measure created by adding a weighted sum of the χ_n^2 values.

$$S = \sum_{i=1}^{i=10} w_i \chi_i^2$$

The lower the value of this quality measure, the higher the quality of the sample.

The simplest set of weights was for all the weights to be one. Since the mean of the χ_n^2 values is not the same this in fact weights χ_1^2 greater than the χ_{10}^2 value.

To provide an estimate of the inequality of using a constant weight of 1, the size of the critical values of the chi-square value at the 50% confidence level may used to give the equivalent weighting. These values are given in table 4.6.

n	Equivalent Weight
1	10.8
2	5.4
3	3.5
4	2.6
5	2.1

n	Equivalent Weight
6	1.7
7	1.4
8	1.3
9	1.1
10	1

Table 4.6: The Equivalent Weightings or Bias using a 50% Confidence Level

The bias towards the smaller n may be removed by making w_i equal to the reciprocal of the equivalent weighting and scaling so that w_1 is 1. The weightings are as follows:

I	w_i
1	1
2	2
3	3.086
4	4.154
5	5.143

I	w_i
6	6.353
7	7.714
8	8.308
9	9.818
10	10.8

Table 4.7: The Weightings Required to remove Bias

If a sample was at the limit of each value of the filter, the weighted value for each test statistic would be 198.33, thus the highest quality measure that a sample could have and still pass the filter is 1983. This procedure of filtering and ranking is performed by the program SELECT.

Thus a procedure has been created that will select high quality samples and give them a quality measure. Before the test was applied a further evaluation was made. This was to determine if the test would reject poor samples that had been specially constructed and may "fool" it.

4.6 PERFORMANCE OF TEST ON PATTERNS

There must always be a concern that a constructed sample may pass an empirical test even though it has obvious characteristics that would make it unsuitable for use in discrete-event simulation. For the test and selection procedure to be acceptable it must be able to distinguish between these constructed sets of data and a real representative sample of the variable.

4.6.1 The Patterns

One method of systematically testing such a procedure is to produce patterns. Two basic patterns that should be always be used in such an evaluation are the constant value and the ramp. The constant value used was 0.5. The ramp consisted of the 1000 values growing linearly from 0.0005 to 0.9995.

A standard method to introduce patterns with strong autocorrelations is to produce saw tooth patterns. To create such saw tooth patterns the following expression was used:

Value(n) = 1-[|mod(n,1000/k)-(1000/2k)|/(1000/2k)]
If Value(n) was zero it was set to 0.0000001.

The value of k decides how many peaks there are in the series. Following are shown three examples, with k equal to 1, 3, and 10.

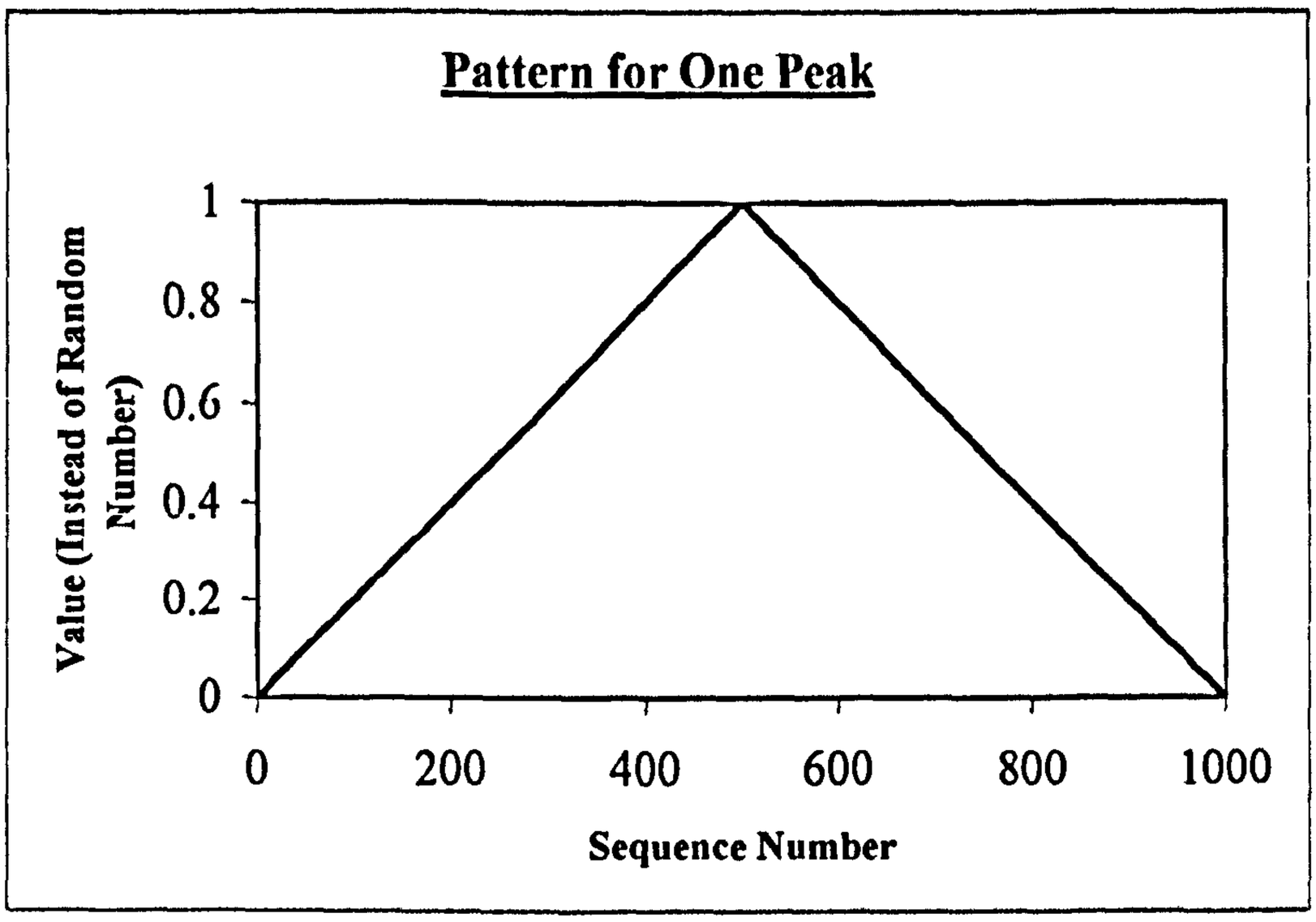


Diagram 4.2: Pattern for One Peak

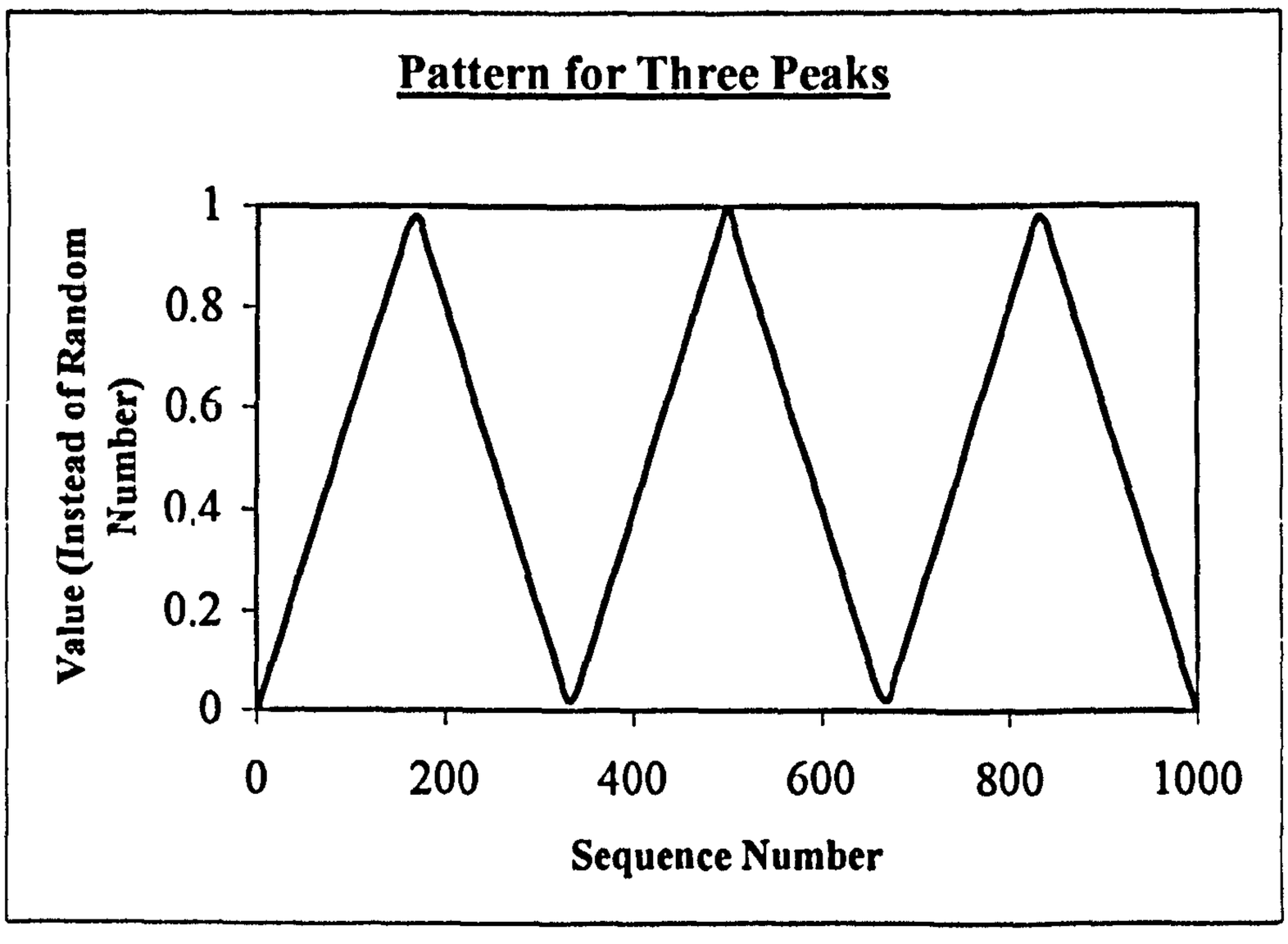


Diagram 4.3: Pattern for Three Peaks

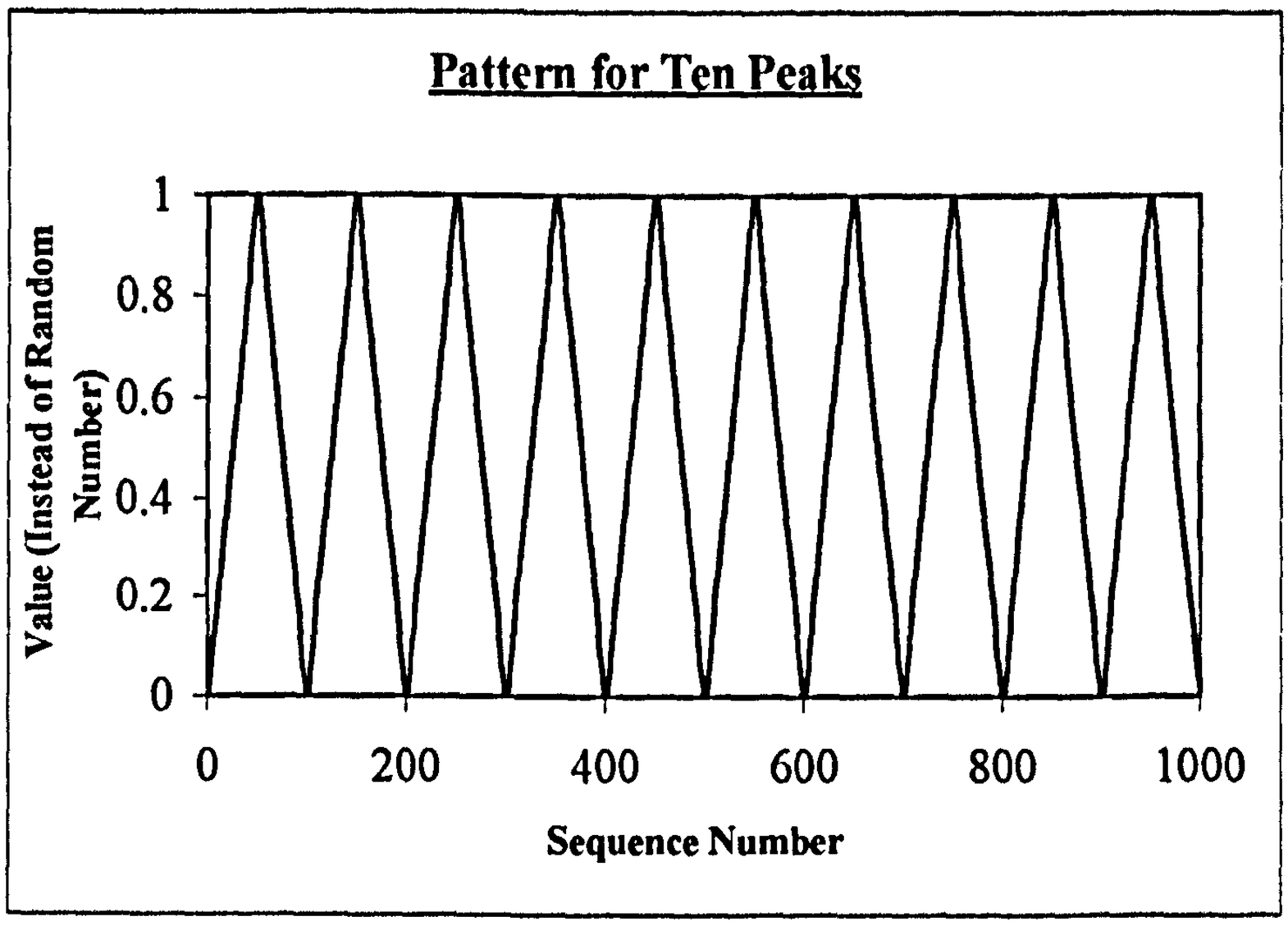


Diagram 4.4: Pattern for 10 Peaks

For the evaluating of the power of the test to discriminate between randomness and patterns of data, saw tooth patterns varying from one peak to twenty-five were used.

The patterns of values varying between 0 and 1 were converted into negative exponential samples and the test was applied using the computer program NONRAND. (See Appendix 1)

4.6.2 The Results Obtained

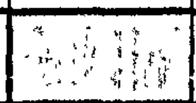
Number of Consecutive Samples	1	2	3	4	5	6	7	8	9	10
Pattern										
Flat	199,000	49,500	21,845	12,250	7,800	5,345	3,891	3,000	2,353	1,900
Ramp	0	302	582	663	636	575	508	454	398	359
1 Peak	40	302	567	664	655	583	508	472	398	352
2 Peaks	118	308	575	674	664	584	519	454	414	362
3 Peaks	200	354	739	681	676	565	517	473	386	359
4 Peaks	594	342	601	669	677	585	487	454	391	345
5 Peaks	990	380	587	720	690	588	494	560	383	370
6 Peaks	1,394	458	616	718	634	572	526	450	414	368
7 Peaks	1,796	553	707	695	660	612	508	473	422	375
8 Peaks	2,187	578	688	717	1,093	601	502	463	408	454
9 Peaks	2,588	727	1,303	724	725	622	541	488	386	375
10 Peaks	2,960	780	652	970	880	608	524	440	400	380
11 Peaks	3,393	876	993	785	813	695	509	486	488	474
12 Peaks	3,791	1,039	894	783	731	616	581	472	427	362
13 Peaks	4,192	1,041	735	913	741	616	550	520	438	383
14 Peaks	4,596	1,280	951	896	723	660	585	486	414	358
15 Peaks	4,966	1,112	1,335	790	779	893	537	466	477	380
16 Peaks	5,344	1,222	897	807	760	654	526	512	439	381
17 Peaks	5,776	1,348	969	762	724	660	627	506	441	392
18 Peaks	6,209	1,573	1,192	911	1,738	722	573	484	426	518
19 Peaks	6,542	1,576	1,186	1,176	808	614	586	536	444	402
20 Peaks	6,840	1,660	878	970	1,080	648	531	460	398	460
21 Peaks	7,424	1,978	1,128	989	878	692	576	600	433	390
22 Peaks	7,793	2,099	2,463	1,080	1,481	945	613	546	778	422
23 Peaks	8,198	1,908	1,330	1,076	842	838	678	596	479	360
24 Peaks	8,642	2,031	1,413	888	840	732	547	481	437	401
25 Peaks	8,750	2,000	1,289	1,250	1,050	674	491	750	415	650

Table 4.8: The χ^2_n Values Obtained for Certain Patterns

4.6.3. Discussion of the Results

The discrimination was completely successful. Every sample produced by the patterns fails to be selected by SELECT

Even if the filter is amended to have a most lenient criterion, that is a very low "confidence" level of 0.1%, all the patterns fail.

Using the critical values of the Chi-Square Distribution given in Table 4.4 and examining the χ^2_n in Table 4.8 shows that virtually all the values of χ^2_n are rejected at the 0.1% confidence level. Only the χ^2_1 values for the Ramp and the Saw Tooth patterns of up to three peaks were acceptable at the 0.1% "confidence" level of 266.39. All other χ^2 values are above the 0.1% "confidence" level.

The Ramp obviously provides a perfect set of values for the distribution but the sequence is of course is very regular and not random, and thus it fails at every combination of samples.

5 SELECTION OF METHOD OF PRODUCING RANDOMNESS (SELECTING THE METHOD)

Four methods of creating samples of a stochastic negative exponential deviates were considered. The methods were:

- Real Random Numbers (or to be more precise ones created from a random physical device)
- MLCG
- Mersenne Twister
- Descriptive Sampling

In each case the random number created was converted to a negative exponential deviate by the "natural log" transform.

The basis of the selection was that

- Real random numbers are often considered as the "ideal",
- MLCGs are still the most popular,
- Mersenne Twister is the one now often given in a list of suggested RNG (see for example L'Ecuyer 1998).
- Descriptive Sampling has in the past been shown for certain simulation problems to more efficient than MLCGs. (Saliby 1990b)

5.1 REAL RANDOM NUMBERS

As discussed in the Literature Review, the availability of faster methods of retrieving saved random numbers from storage devices that in turn are able to hold large amounts of data, has lead to the revival of the idea of using real random numbers for discrete-event simulation.

It is assumed in most texts that the ideal source of randomness for simulation would be real random numbers and thus the search continues for generators

that mimic as closely as possible the behaviour of random numbers. However, as discussed in the Literature Review, the concept of quasi-random numbers for use in Monte-Carlo analysis has gained favour among some workers as they provide more accurate results. Thus non-random methods of introducing stochastic behaviour should also be considered. However, to provide a basis of comparison, the first step was to measure and evaluate the ability of real random numbers to create highly representative samples.

The source of real random numbers used in the evaluation was that already used for the calibration of the filter. Indeed the same set of random numbers was used. The results are a therefore a reflection of that calibration. The values used in the filter were selected to give a relatively small number of very high standard samples. The low number of good samples obtained is a reflection of this high standard but also the ability of the real random number source to create them. Thus the number of samples selected from 40000 samples was able to be used as a measure of the ability of the source of randomness to produce high quality samples.

The distribution of the values of the "quality measure" is independent of the values used in the filter measure and is thus a measure of the quality of the samples.

Quality Sequence	Section Number	Weighted Sum
1	629	1467.25
2	38865	1469.84
3	24076	1477.70
4	20378	1483.38
5	32489	1484.21
6	22823	1490.54
7	8814	1501.17
8	35000	1515.09
9	19444	1517.56
10	1796	1522.55
11	10337	1528.24
12	3697	1535.34
13	6816	1537.14
14	31482	1547.93
15	9716	1553.89
16	964	1556.73
17	11176	1561.11
18	26732	1561.12
19	38813	1564.94
20	15853	1567.74
21	5921	1568.94
22	21144	1569.80
23	6472	1572.00
24	31283	1577.98
25	29263	1580.14
26	20492	1582.39
27	32334	1599.89
28	7230	1610.72
29	15600	1620.35
30	5129	1620.93
31	37942	1625.24
32	2928	1625.82
33	8507	1626.48
34	10366	1626.92
35	27197	1627.00
36	8759	1628.67
37	13223	1630.75
38	2189	1634.98
39	15365	1637.12
40	31701	1640.34
41	17748	1643.85
42	9066	1645.06
43	31267	1645.48
44	23017	1647.78
45	25250	1648.98
46	30121	1649.56
47	14685	1655.42
48	7058	1663.00
49	32818	1664.92
50	2903	1668.82
51	17555	1669.14
52	7679	1669.20
53	15370	1670.53
54	24064	1672.48
55	3300	1678.71
56	11556	1682.06
57	35797	1682.87
58	30391	1682.91
59	23006	1688.36
60	5925	1688.56
61	19514	1694.20
62	19361	1697.04
63	38583	1697.96
64	30605	1703.05
65	6379	1712.11
66	30870	1718.48
67	19904	1719.62
68	22306	1721.89
69	20673	1725.38
70	4709	1730.69
71	16978	1731.60
72	10100	1731.89
73	34612	1732.88
74	25585	1735.12
75	34155	1739.52
76	27714	1739.85
77	23594	1740.10
78	28117	1789.26

Table 5.1: The Selected Sections of the Real Random Numbers in "Quality" Sequence

5.1.1 The Number of Quality Samples.

As described in Appendix 6, the real random numbers available on the web site (Random.org) enabled 40,000 samples of 1000 negative exponential deviates to be created. The choice of the values used in the filter lead to 78 samples being selected as highly representative from the 40,000 created.

5.1.2 The Distribution of the Quality of the Samples.

Table 5.1 gives the distribution of the Quality Measure for the selected samples. The samples are identified by the sequence number of the section of 1000 random numbers used to create their sample values. The samples were created from consecutive non-overlapping sections of 1000 random numbers. The sample with the highest quality is the one with the lowest quality measure of 1467. The sample with the lowest quality, but still selected as highly representative, has a quality measure of 1789.

5.1.3 Discussion

The use of real random numbers has a number of practical difficulties. They would only be chosen if the other methods of providing randomness created inferior samples. The results from the real random numbers were used as the standards that had to be equalled or surpassed.

5.2 MLCG

5.2.1 Selection of MLCGs

In the "Methods of Supplying the Stream of Random Numbers" section the continuing popularity of MLCGs was discussed. This popularity still exists even though there have been serious concerns about their use due to the easily demonstrated non-random behaviour that can be observed when consecutive

samples values are converted into points in n-dimensional space. Some writers have used cautionary phrases when discussing this failure, such as "if this is important to you" (Bennett 1998) but none have reputed the findings as being irrelevant, nor has any researcher produce a measurement that demonstrated that if the MLCG had a certain "good result" in the spectral test (Knuth 1988) that they were acceptable. Knuth offers an opinion on the acceptability of certain results from the spectral test but not an opinion based on measurement.

In order to examine if MLCGs are acceptable for use within discrete-event simulation, a number of MLCGs were tested using the discriminating test that has been developed in the previous section, and the results obtained with the MLCGs were compared with those achieved using real random numbers. The following seven MLCGs were considered:

Proposer	Reference	Line in Table 1 Page 106 of Knuth (1998)
Fishman and Moore	Fishman G. S. and Moore L. R., (1986)	18
L'Ecuyer	L'Ecuyer P., (1988)	21
Lewis et al.	Lewis P A W, Goodman A S and Miller J M (1969)	19
Knuth	Knuth (1998)	17
Killingbeck L	Knuth (1999)	No Entry
IBM	Knuth (1998	12
Marse and Roberts	Marse and Roberts (1968)	No Entry

Table 5.2: The MLCGs Considered

The choice of the first five MLCGs was based on their good performance in the tests specified by Knuth, especially the spectral test. Their performance in the spectral test is documented by Knuth in his Table 1 (Knuth 1998) and for the one proposed by Killingbeck in his proposed future amendments to the table (Knuth 1999). The modulus of the selected MLCGs made them easy to port to many computers and thus suitable choices.

Two other MLCGs that were not recommended by Knuth were also included in the study. The first was the one proposed by Marse and Roberts (1968) called UNIRAN, which was designed as a "portable" random number generator and as speed was then a major concern, had an alternative method of computation of the "pseudorandom" number so as to avoid the division and thus performed the calculation quicker. In this thesis the divide was made in the form of a "mod" operation and was used for all the generators, since the concern with speed that existed in 1968 is no longer as relevant.

The other MLCG added to the list was RANDU. This RNG was introduced by IBM, and was once frequently used, being in the original IBM FORTRAN Scientific Subroutine Library for the IBM 360 (IBM 1970), but was later discredited (see Knuth and Appendix 2) and replaced in IBM's IMSL Scientific Subroutine Package (1978) by the Lewis et al. generator (Lewis et al., 1969). One of the main reasons for it being discredited is that it is considered to fail the spectral test in 3-dimensional space. Knuth (Page 107,1997) states that, "it should never have been used."

The definitions of the generators as used in the study are given in the following table.

Proposer	Name Used in Study	Modulus	Multiplier
Fishman and Moore	F & M	2147483647	62089911
L'Ecuyer	L'Ecuyer	2147483399	40692
Lewis et al.	Lewis	2147483647	16807
Knuth	Flying	2147483647	314159269
Killingbeck L	Killingbeck	4294967296	2650845021
Marse and Roberts	UNIRAN	2147483647	630360016
IBM	RANDU	2147483648	65539

Table 5.3: The Definitions of the MLCGs Being Considered

The MLCGs had no offsets and thus were strictly speaking LCGs.

The spectral test ignores the offset value c (see page 34), or as it is sometimes called "the additive constant", so no value for any offset is given in Knuth's table 1 (page 106 of Knuth 1998). However, if in the MLCG referenced here by the name Flying (this is because it is stated by Knuth as being one of the MLCG that passing his tests with flying colours), an offset of the value 1 is used, the MLCG no longer has the maximum cycle length. For all but one seed, the cycle is one less than the maximum and that seed, which has a value of 1,395,119,659, has a cycle of only one (See Appendix 4).

It was stated by L'Ecuyer and Lemieux (1999) that it may be better in Monte Carlo studies to choose a RNG with a reduced cycle size as the accuracy of the result may be improved by having a more even spread set of sample values. This may be true for discrete-event simulation as well. A brief study was therefore made of other MLCGs with much smaller cycle sizes to determine if this suggestion was correct.

5.2.2 Performing the Test

The tests were made to determine:

- If the MLCGs produce the same number of highly representative samples as the real random numbers. This was performed by creating for each MLCG 40,000 samples using seeds values from 1 to 40000 and comparing how may of the 40000 were selected as by SELECT with the 78 selected from the 40000 samples created from the real random numbers.
- If the MLCGs produce the same Quality of samples as the real random numbers. This was determined by measuring the distribution of the quality measure of the selected samples.

- If one of the MLCGs was superior in creating good samples. This was determined by checking all the samples created by the MLCGs, to see if the distribution of any MLCG samples was significantly different from the average. The check was one able to determine location as well as shape of distribution differences.

- If the seed selection was important. This was determined by comparing the quality measure obtained by samples created by different seeds for the same MLCG.

For testing the MLCGs, the program LCGTEST was used (see Appendix 1). This program calls a subroutine MCG that is a double precision version of a general MLCG and it is thus able to use all the moduli of the actual generators being tested. As previously stated, it performs a standard double precision "mod" operation. A listing of the subroutine MCG is given in Appendix 1.

5.2.3 Removal of Overlapping Sets of Random Numbers

It is possible to select two seeds that will produce 1000 random numbers that overlap. This will occur in an MLCG if the 1000 random numbers created by one of the seeds contains the other seed. This may not be considered a problem in actual use, since the simulation model will most likely be in a very different situation when the numbers are repeated. But in order that there is no remaining concern a program COMB (see appendix1) was created that would detect and remove any overlapping seed.

5.2.4. Result of the test if MLCGs produce the same Number of Highly Representative Samples as the Real Random Numbers

The following table give the number samples selected (high quality or highly representative samples) from the 40,000 samples of 1000 sample values created by the real random numbers and the corresponding results from the first 40,000 seeds of the MLCG random number generators.

In none of the cases did the COMB program reduce the number of samples selected by SELECT. Thus there was no overlap.

Name Used in Study	Number Selected	Number of Samples Having Quality Measure Less than:							
		1500	1550	1600	1650	1700	1750	1800	1850
Real Random Numbers	78	6	14	27	46	63	77	78	78
F & M	80	4	10	17	40	59	70	78	80
L'Ecuyer	84	2	10	26	41	62	78	83	84
Lewis	63	1	3	12	27	40	59	61	63
Flying	82	2	6	16	37	65	77	81	82
Killingbeck	64	2	6	13	23	49	61	63	64
UNIRAN	68	2	6	16	32	49	63	67	68
RANDU	78	1	11	24	41	59	74	78	78

Table 5.4: The Number of Samples Selected and the Distribution of their Quality Measure (Comparing Real Random Numbers and MLCGs)

The results for the real random numbers and the MLCGs appear similar, but it was necessary to test them statistically.

The values of "Number Selected" obtained from the MLCGs and the real random numbers must be considered as sample values, as the results using different sets of 40000 seeds or from another stream of real random numbers would be different but may be assumed to be distributed according to some statistical distribution. To test if the number selected is statistically significantly different a test not dependent on the distribution of the "Number Selected" was used. The test used was the chi-square test. (See pages 42-47 Siegel 1956). In this application of the test all the "Number Selected" values for all the MLCGs were tested simultaneously and were considered to give values of "Number Selected" that had the same distribution as would be given by real random numbers. It was assumed that the expected number to be selected was 78. Only the cells for "selected" were used in the calculation since the "non-selected" cells would not add any measurable value to the χ^2 value. Thus there are really 14 cells and since there are 7 totals there are (14-7), that is 7 degrees of freedom. Table 5.5 details the calculation.

Name Used in Study	Number Selected	Expected (From Random)	Calculation
F & M	80	78	0.05
L'Ecuyer	84	78	0.46
Lewis	63	78	2.88
Flying	82	78	0.21
Killingbeck	64	78	2.51
UNIRAN	68	78	1.28
RANDU	78	78	0.00
		$\chi^2 =$	7.40

Table 5.5: Calculation of χ^2

The 30% "confidence" limit for 7 degrees of freedom is 8.38 (Table C, Siegel 1956). A "confidence level" of 30% is a severe test. Thus the null hypothesis that, the "Number Selected" values are statistically the same as the values obtained from real random numbers, cannot be rejected. Thus the MLCGs

were able to create the same number of highly representative samples (i.e. approximately 1 from 500 candidates) as the real random numbers.

5.2.5 Comparing the Quality of the Selected Samples with those from Real Random Numbers

Having determined that the number of quality samples selected is the same as that created by the real random numbers, the next question that was asked was "are the selected samples of the same quality?" To answer this question the distribution of the quality measure of the samples selected was compared. This was performed using the Kolmogorov-Smirnov two-sample test. (See pages 127-136 Siegel 1956). To perform this test the cumulative fraction of the sample having a sample measure less than a certain value was calculated from Table 5.4. The maximum absolute deviation between the result from the real random numbers and the individual MLCG was calculated. The critical 10% value calculated from the expression given in table M of Siegel 1956 was also calculated. The results are shown in the following table 5.6.

All except the results from Killingbeck's RNG passed the severe 10% "confidence" level and the null hypothesis that they are the same distribution is accepted. Thus it is considered that they all come as if from the same distribution as the real random samples. Killingbeck's passes at the more liberal but acceptable 2.5% "confidence" level (critical value 0.25) and thus also must also be taken as coming as if from the same distribution as the real random numbers. There is a probability of 52% that in seven independent tests, at least one null hypothesis that should not be rejected will fail at the 10% "confidence" level.

Name Used in Study	Number Selected	Fraction of Selected Samples Having Quality Measures Less than:								Maximum Absolute Deviation From Random	10% Critical Value
		1500	1550	1600	1650	1700	1750	1800	1850		
Real Random Numbers	78	0.08	0.18	0.35	0.59	0.81	0.99	1.00	1.00		
F & M	80	0.05	0.13	0.21	0.50	0.74	0.88	0.98	1.00	0.13	0.19
L'Ecuyer	84	0.02	0.12	0.31	0.49	0.74	0.93	0.99	1.00	0.10	0.19
Lewis	63	0.02	0.05	0.19	0.43	0.63	0.94	0.97	1.00	0.17	0.21
Flying	82	0.02	0.07	0.20	0.45	0.79	0.94	0.99	1.00	0.15	0.19
Killingbeck	64	0.03	0.09	0.20	0.36	0.77	0.95	0.98	1.00	0.23	0.21
UNIRAN	68	0.03	0.09	0.24	0.47	0.72	0.93	0.99	1.00	0.12	0.20
RANDU	78	0.01	0.14	0.31	0.53	0.76	0.95	1.00	1.00	0.06	0.20

Table 5.6: The Fraction of the Selected Seeds having a Certain Quality Measure

The statistical test was able to detect any difference in quality between the selected samples from the MLCGs and the real random numbers

5.2.6 Discussion of the Comparison of MLCGs and Real Random Numbers

All the MLCGs that were considered in the study produced highly representative samples at the same rate and quality as the real random numbers.

This supports the contention that the MLCGs are acting like real random numbers. Thus, for this purpose the MLCGs, may be used instead of real random numbers. The ability to select a seed (either by inputting the seed value directly or by using a stream number) and always get the same sample gives them a convenience over "on-line" real random numbers.

The surprising conclusion that can be taken from the results, is that the "discredited" RANDU, if used with certain selected seeds, can create samples of the negative exponential distribution that are as good as the best created by the other MLCGs, that unlike RANDU, are considered to pass the spectral test.

Since the chance of selecting a seed at random, or a sequence of real random numbers that will create a sample that would rival the performance of RANDU with one of the carefully selected seeds, is approximately 1/500 or 0.2%, it may be said that indeed RANDU, used in these conditions would in the majority of cases (99.8%), outperform the real random numbers and other MLCGs with randomly chosen seeds.

The next question is which if any of the MLCGs outperforms the others.

5.2.7 Determining the Relative Performance of the MLCGs

In order to improve the accuracy of the study, the number of samples of 1000 sample values, and thus the number of seeds, was increased to 60,000.

The values of the quality measure of all the samples are given in Table 5.7.
It must be noted that there has been no removal of poor quality samples and every one of the 60000 seeds is included. Thus any difference will not be due to the action of the filter.

The results from each MLCG can be seen to be similar. The distributions were then examined statistically.

To perform the analysis, an average distribution for MLCGs was created by pooling all the samples. This was therefore based on 420,000 samples. With such a large number of samples this distribution was taken as the true

population values. Cumulative percentages of the quality measures were calculated for this population distribution, each of the MLCGs and also the 40000 samples from the real random numbers.

Name Used In Study	Up to 1500	1600	1700	1800	1900	2000
F & M	151	967	4013	8965	12852	12921
L'Ecuyer	146	1062	4018	8812	12884	13013
Lewis	155	1000	3915	8979	12968	13071
Flying	156	1082	3963	9034	13049	12939
Killingbeck	132	1015	4078	8839	12906	13114
UNIRAN	155	1075	4062	8992	12944	12836
RANDU	158	1121	3919	8883	12927	13020

Name Used In Study	2100	2200	2300	2400	2500	Above 2500
F & M	9611	5657	2833	1206	506	318
L'Ecuyer	9611	5647	2719	1270	490	328
Lewis	9630	5544	2794	1167	466	311
Flying	9563	5536	2733	1189	445	311
Killingbeck	9626	5577	2815	1160	459	279
UNIRAN	9693	5467	2736	1238	475	327
RANDU	9670	5588	2685	1197	495	337

Table 5.7: The Quality Measure Counts from 60,000 seeds for each MLCG

The distribution of the samples from the real random numbers was included to enable a check to be made that, using all the samples rather than just those passing the filter, would not invalidate the analysis. The result is shown in the following table.

Name Used In Study	Up to 1500	1600	1700	1800	1900	2000
F & M	0.25%	1.86%	8.55%	23.49%	44.91%	66.45%
L'Ecuyer	0.24%	2.01%	8.71%	23.40%	44.87%	66.56%
Lewis	0.26%	1.93%	8.45%	23.42%	45.03%	66.81%
Flying	0.26%	2.06%	8.67%	23.73%	45.47%	67.04%
Killingbeck	0.22%	1.91%	8.71%	23.44%	44.95%	66.81%
UNIRAN	0.26%	2.05%	8.82%	23.81%	45.38%	66.77%
RANDU	0.26%	2.13%	8.66%	23.47%	45.01%	66.71%
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Average	0.25%	1.99%	8.65%	23.54%	45.09%	66.74%
Real Random Numbers	0.28%	1.99%	8.44%	23.41%	44.73%	66.38%

Name Used in Study	2100	2200	2300	2400	2500	Above 2500
F & M	82.47%	91.90%	96.62%	98.63%	99.47%	100.00%
L'Ecuyer	82.58%	91.99%	96.52%	98.64%	99.45%	100.00%
Lewis	82.86%	92.10%	96.76%	98.71%	99.48%	100.00%
Flying	82.98%	92.20%	96.76%	98.74%	99.48%	100.00%
Killingbeck	82.85%	92.15%	96.84%	98.77%	99.54%	100.00%
UNIRAN	82.93%	92.04%	96.60%	98.66%	99.46%	100.00%
RANDU	82.83%	92.14%	96.62%	98.61%	99.44%	100.00%
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Average	82.78%	92.07%	96.67%	98.68%	99.47%	100.00%
Real Random Numbers	82.44%	91.89%	96.64%	98.68%	99.47%	100.00%

Table 5.8: The Percentage of the 60,000 Seeds for each MLCG falling in certain Quality Measure Ranges

Since the average distribution of the MLCGs is being used as the population distribution against which all other distributions will be measured, the one sample Kolmogorov-Smirnov test was used.

The first test was to determine if the distribution created by the real random numbers was the same as the population. The maximum absolute deviation of the cumulative distribution of the samples' quality measures created by the real random numbers from the cumulative distribution of the population was 0.0036; the critical value at 20% "confidence" level is 0.0054 (See Table E, Siegel 1956). Thus the distributions of the quality measures obtained from real random numbers and the distribution of the population created by the average quality measures obtained from the MLCGs can be considered as being the same. Thus the average quality measure that is being used as the population distribution may be used in the further tests with confidence.

The individual MLCGs were next tested to see if any were significantly better or worse than the average performance. The one sample K-S test was used. In this case as there were 60000 samples rather than the 40000 samples created by the random numbers, the critical value is smaller and at 20% is 0.0044.

Table 5.9 gives the maximum absolute deviation for each MLCG. All were within this severe critical value and therefore it can be considered that they all share the same distribution of quality measures, which was just shown to be the distribution of quality measure created by the real random numbers.

The result shown in table 5.9 show that none of the MLCGs were seen to be superior. Even RANDU was not seen as inferior even though "failing" the spectral test.

Name Used in Study	Maximum Absolute Deviation
F & M	0.0032
L'Ecuyer	0.0022
Lewis	0.0020
Flying	0.0038
Killingbeck	0.0016
UNIRAN	0.0029
RANDU	0.0014

Table 5.9: Maximum Absolute Deviation Between the Average and the individual MLCGs

The next investigation was to determine how important was the selection of the seed.

5.2.8 The Importance of the Seed Selection

The program BAD100 (see Appendix 1) selects the 100 seeds that created the worst samples (that is the 100 samples with highest values of the quality measure), from the 60,000 seeds. Table 5.10 gives the top 10 and bottom 10 seeds for each MLCG.

It can easily be seen that selecting a seed from the top rather than the bottom 10 makes a major difference in the quality of the sample created for every one of the MLCGs. The smallest change in quality measure by changing from the best seed to the worst for any MLCG is an increase of 1771. The increase in the quality measures by keeping the same ranking of seed but changing the selected MLCG is at most 74. This could be tested for statistical significance but the difference is so large as to make such a calculation redundant.

MLCG Rank	F & M	L'Ecuyer	Lewis	Flying	Killingbeck	UNIRAN	RANDU
1	1430	1448	1430	1442	1447	1444	1443
2	1431	1484	1449	1479	1470	1445	1481
3	1463	1504	1482	1499	1476	1458	1501
4	1470	1508	1527	1544	1501	1504	1504
5	1482	1514	1535	1548	1503	1518	1507
6	1497	1515	1537	1548	1520	1528	1511
7	1497	1517	1537	1548	1525	1529	1517
8	1506	1518	1552	1551	1536	1533	1520
9	1512	1533	1561	1560	1539	1538	1524
10	1536	1534	1565	1561	1548	1538	1529
59991	3061	3082	3031	3008	3006	2989	2970
59992	3090	3083	3055	3051	3012	2990	2973
59993	3092	3083	3055	3063	3022	3011	2978
59994	3121	3102	3058	3067	3052	3033	2983
59995	3154	3105	3106	3075	3076	3036	3003
59996	3192	3168	3280	3108	3085	3040	3006
59997	3203	3171	3281	3148	3091	3059	3012
59998	3365	3175	3322	3188	3160	3099	3182
59999	3504	3271	3373	3287	3376	3121	3193
60000	3593	3339	3544	3553	3767	3214	3325

Table 5.10: The Best and Worst Seeds for Each MLCG

It is clear that when a sample of 1000 negative exponential sample values are required the selection of the seed values is much more important than the choice of MLCG.

The test demonstrated that the MLCGs produce quality samples at the same rate and distribution as real random numbers. The performance difference between MLCGs is small but the choice of seed is very important.

However all the MLCGs had a large modulus when compared with the sample size. Such MLCGs are used to give large cycle sizes and to pass the spectral test. L'Ecuyer and Lemieux (1999) suggested better results could be obtained with a smaller modulus. This was next investigated.

5.2.9 Reduced Cycle MLCGs

It was stated by L'Ecuyer and Lemieux (1999) that it may be better in Monte Carlo studies to choose a RNG with a reduced cycle size as the accuracy of the result may be improved by having a more "even spread" set of sample values over the sample space being investigated. This is possibly true for discrete-event simulation as well. A study was therefore made of MLCGs with much smaller cycle sizes then have been considered so far. These were taken from Table 2 in the paper by L'Ecuyer (1999) entitled "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure." They are shown in Table 5.11.

The method of analysis was as before but in this case because the cycles for the MLCG were small, the full cycles were analysed.

The results of the computer runs are given in Table 5.11

5.2.10 The Quality Measures Obtained

In this case with the smaller cycle sizes, there were many overlaps. When the lower quality seed of an overlapping pair was removed the number of seeds available greatly reduced. In the case of the Modulus 524287 and multiplier 283741, there were initially 1155 seeds that met the quality criteria, but when overlaps were removed this reduced to 187.

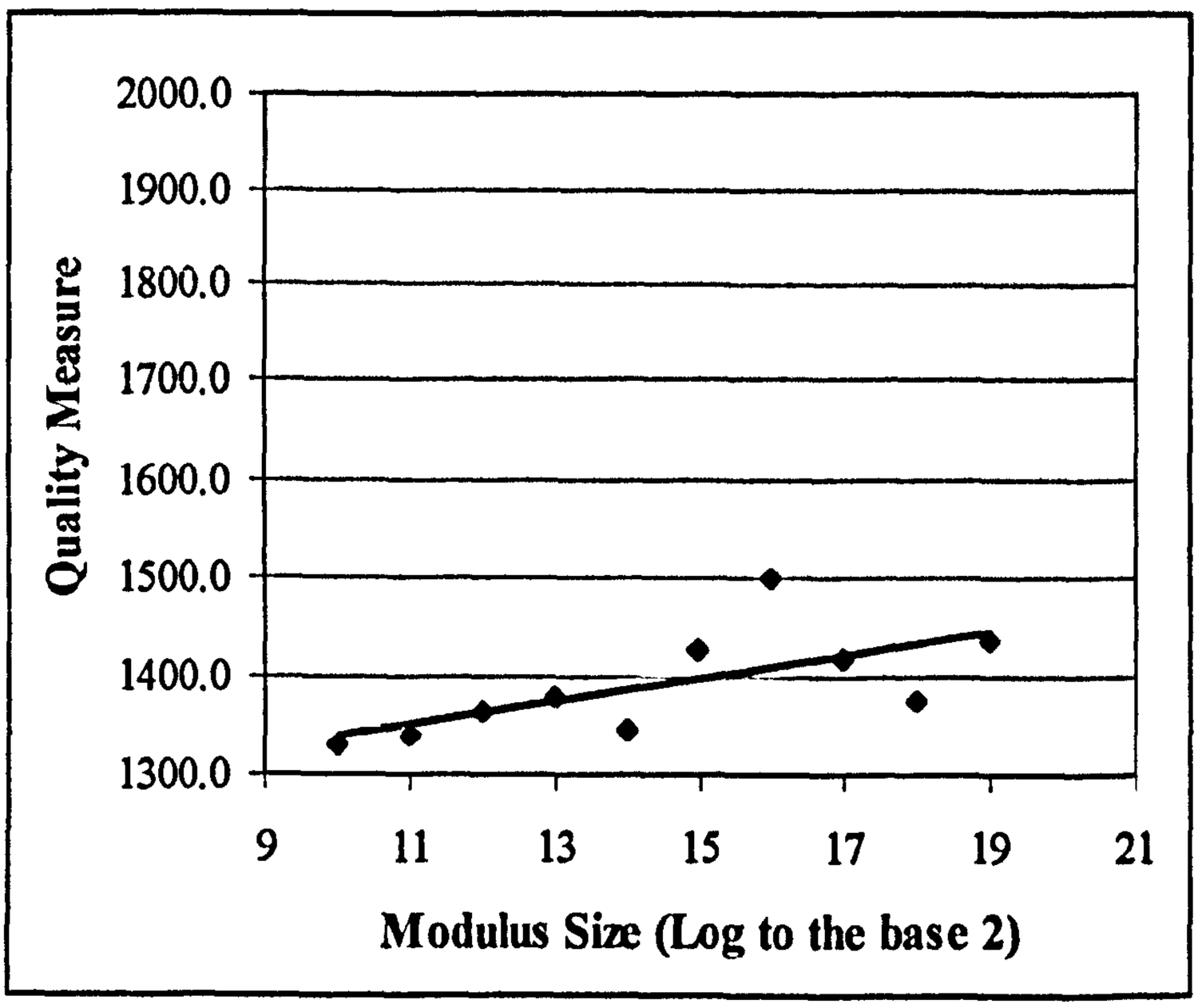
Modulus	Multiplier	Total Number of Seeds Selected	Number of Seeds After Overlaps Removed	Range of Quality Measure	
1021	65	44	1	1331.0	-
	331	0	0	-	-
2039	995	10	1	1430.2	
	328	69	1	1339.0	
	393	13	1	1390.8	
4093	209	12	2	1398.6	1596.3
	235	27	2	1463.2	1636.9
	219	78	3	1401.2	1572.1
	3551	30	2	1364.2	1540.2
8191	884	10	4	1603.0	1727.5
	1716	56	3	1379.6	1677.0
	2685	48	5	1439.9	1588.2
16381	572	74	8	1449.0	1679.2
	3007	44	8	1470.1	1812.4
	665	90	9	1458.3	1608.4
	12957	82	7	1345.3	1586.0
32749	219	110	13	1427.3	1759.2
65521	2469	126	23	1499.5	1752.6
131071	43156	301	45	1416.3	1796.2
262139	92717	423	96	1376.3	1791.7
524287	283741	1155	187	1436.5	1822.1

Table 5.11: The Number of Seeds Meeting the Quality Criteria using MLCGs with Relatively Small Modulus

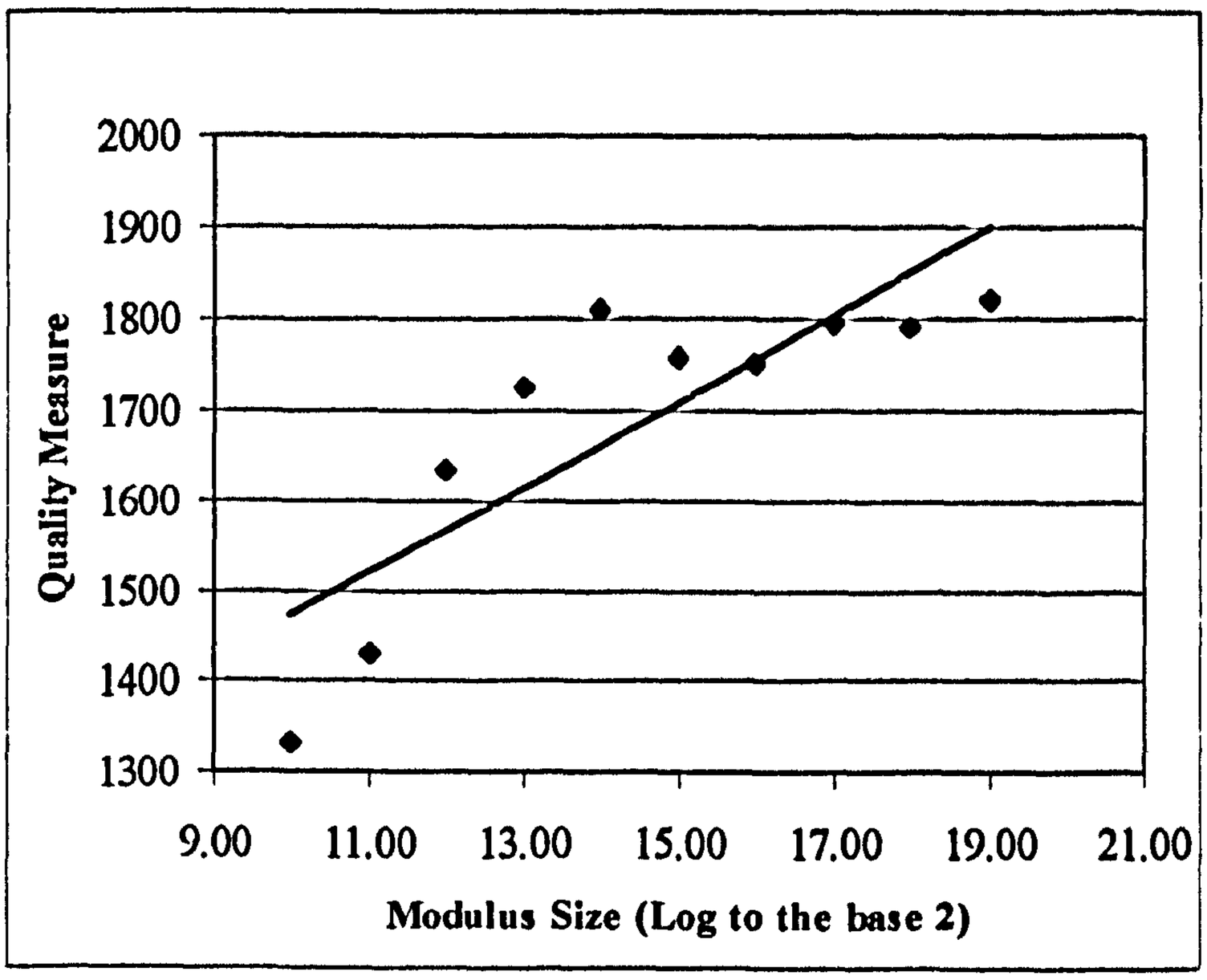
Table 5.11 shows that as predicted, the MLCGs with the smaller modulus tend to give a higher quality of sample. The following graphs indicate that the best and worst seeds for any modulus tended to have higher numerical quality values, which indicate a poorer quality sample, as the size of the modulus increased. (Note: The quality measure is the size of deviation from the ideal

and thus grows in value as the quality reduces.) In the graphs the size of the modulus was measured as log to the base 2. (The conversion to log is used here is to reflect "a rate of growth" and the base 2 is used to obtain a convenient scale.)

It should be noted that due to different sample sizes and theoretical limitations of statistics of the form "best" and "worst", any confidence limits being placed on slopes or correlation factors may only be taken as indications.



Graph 5.1: Best Seed's Quality Measure against Modulus Size



Graph 5.2: Worst Seed's Quality Measure against Modulus Size

The "r coefficient" for the best seeds performance against modulus size is 0.66 and for the worst seeds performance against modulus size is 0.83. These "r coefficient" values would indicate acceptable at the 3.7% and 0.3% level respectively (Calculated using Excel; also see table 6.2 Neave 1989). If in the "best" plot the rather poor result for 65521 is removed, the "r coefficient" value increases to 0.78. This passes the more serve 1% "confidence" level.

Two factors are present in the plot of worst seeds. The modulus increase and so does the number of seeds in the set being considered. Thus for very small modulus where there was only one seed passing the quality criteria, the worst and best are the same seed whilst for 524287 there were 187 seeds. It must also be noted that the "worst seeds" are the worst drawn from the set of 187 *passing* the quality criteria and are certainly not the worst seeds when all the possible seeds are considered.

However a Multiple Regression Analysis of the worst (largest numeric value) quality measures against the multiplier and the number of seeds in the set gave the coefficient for the number of seeds in the set as –1.5. This is obviously a spurious result and the result would be taken as zero, thus the number of seeds in the set can be rejected as a factor in the reduction in the quality of the samples.

The use of the smaller moduli is however limited as it could only provide few seeds that both met the criteria and did not overlap. They would also unlikely to be suitable for providing larger sample sizes. However the two largest moduli in the table, 262139 and 524286 with their respective multipliers 92717 and 283741 are very acceptable MLCGs for when only 1000 sample values are needed. The results for these MLCGs are given in Appendix 3.

This result may be of great importance when small samples are being considered. If the "rule of thumb" for run length that at least 10-20 samples are taken from a distribution were used, then samples of as small as 10-20 values would need to be considered.

5.2.11 Summary of the Evaluation of MLCGs

The test proved successful in isolating seeds that gave highly representative samples. The behaviour was as predicted by the application of the test on real random numbers.

All the MLCGs in table 5.3, when used with certain selected seeds gave highly representative samples with 1000 sample values. If only a small number of seeds are required then the MLCGs with smaller moduli, again with selected seeds can also be recommended.

These seeds and MLCG combinations that are recommended are given in Appendix 3.

Although the results for RANDU and UNIRAN are not included in the results in Appendix 3, they did not show any performance problems. This indicates that the spectral test may not be as relevant as the quality check of the sample for selecting RNGs for use within discrete event simulation, at least in the situations where the sample is created from a single random number. In cases where two or more random numbers are needed to create a single sample value there may still be a performance problem and until that has been resolved the spectral test may remain an essential test. Since the number of generators that pass the spectral test is sufficiently large it may remain prudent to use only MLCG that pass this test and select the seed to be used by the tests developed here.

5.3 EVALUATION OF THE MERSENNE TWISTER

5.3.1 Description of the Mersenne Twister

A RNG that is gaining interest in the simulation and computational analysis communities is the Mersenne Twister (MT) of Matsumoto and Nishimura (1998) which is a modification of a "Generalised Feedback Shift Register Generator" (see for example L'Ecuyer 1998 and also the Mersenne Twister Home Page 2002 http://www.math.keio.ac.jp/~matumoto/emt.html).

L'Ecuyer (1998) states that in his view it is one of the few RNGs that can be recommended. In their paper Matsumoto and Nishimura give the scientific bases for the 623 "dimensional equidistribution" and state that it passes all the tests that are available, including Marsaglia's Diehard. Thus MT avoids all the criticism aimed at the MLCGs. They state it can be tested more rigorously than MLCGs, as the spectral test is limited in the dimensionality able to be tested

due to the existing computational difficulties (dimensions greater than 100 are claimed to be too lengthy in computer time). Thus they claim MT is not open to the criticism that it is chosen due to the difficulty in demonstrating non-randomness due to an inability to analyse its behaviour. They also make the important claim that its formulation corrects the deficiencies that exist in a similar simple form of this type of RNG, which were reported by Ferrenberg et al. (1992). However, as already stated, it has been reported that certain seeds can lead to non-random behaviour. It cannot be assumed that the excellent behaviour of the generator over a full cycle, determined by a theoretical analysis, will give any guarantee for short, or even what may be considered in practical terms long, term behaviour. Thus empirical tests on the sample sizes that are used in practical investigations are still essential.

.

5.3.2 Concerns with the Mersenne Twister

One of the concerns is that MT required 624 seeds and thus there is a start up cost in terms of processing time and time to generate a random number. MT was designed to be used in Monte Carlo studies where often there is only a single stream of a very large number of random numbers used. Therefore this need to generate 624 seeds at the start of a simulation run is not a serious concern for Monte Carlo studies and that by their nature require lengthy computer runs or very fast computers. Discrete-event simulation will typically required many streams as this gives the ability to repeat the behaviour of any variable (Tocher 1960). Therefore a number of sets of 624 seeds will need to be generated. Although in practice the number of streams required is not normally that great, as may be seen from the fact that WITNESS, a leading simulation package, has 100 built in random number streams (Lanner 1998). (The package has actually 200 but 100 use the full word size and these 100 streams have larger cycles and are recommended to be chosen; the first 100 remain available in the package for backward compatibility.)

The MT generator has, as already stated, an enormous cycle length. This brings great benefits from the standpoint of the creation of as nearly as perfect as possible imitation of real random numbers. But in the study just described, it was shown that, where a small sample is required, the MLCGs with the smaller cycles give the more representative samples. This leads to the concern that MT, with its large cycle, may create fewer high quality small samples and thus not be an ideal choice for discrete event simulation where there may frequently be a requirement for small samples.

5.3.3 Creation of the 624 Seeds

The first problem in implementing MT is its requirement for 624 seeds. As already discussed in the implementation of MT by Nishimura (Mersenne Twister Home Page, 2002), a MLCG suggested by Marsaglia (described in Knuth 1999 page 108) was used to create the 624 seeds from a single seed. Thus from the user's viewpoint the total procedure, when using this implementation, only requires the one seed. The full set of values 1 to 2^{32}-1 are available for seeds to be input into the Marsaglia MLCG, which has a modulus of 2^{32}. The study used the same implementation and thus the samples created for this investigation were those created using the seeds generated by Marsaglia's MLGC.

5.3.4 Performing the Tests

The tests were designed to determine:

- If MT produce the same number of highly representative samples as the real random numbers. This was performed by creating 40,000 sets of 624 seeds using Marsaglia's MLGC with seeds values 1 to 40000. Then creating from these sets of seeds, 40,000 samples of 1000 sample

values using the MT generator and comparing how many of the samples thus created were chosen by SELECT with the number selected from the 40000 samples created from the real random numbers, that is 78.

- If MT produce the same quality of samples as the real random numbers. This was determined by comparing the distribution of the quality measure of the selected samples from the MT generator and real random numbers.

- If MT produces higher quality of samples than the MLCG. In this case the number of samples created using MT was increased to 60,000. Thus seeds of 1 to 60000 were used in the Marsaglia's MLGC to create the sets of seeds for the MT generator. The samples selected by SELECT were compared with the samples similarly selected from the 60,000 created by the MLCGs. The distribution of the "Quality measure" for the MT selected samples was compared with the average distribution of the MLCGs selected samples. The measurement of whether the MLCGs were better, or even just different, was made.

- If the seed selection is more important than the choice between using MT instead of MLCG RNGs. This was determined by measuring the difference in Quality Measure from changing to MLCGs from MT against the change that occurs by changing seeds in the MT generator.

To perform the above set of tests, a program MTTEST was written. Details of the program are in Attachment 1. The subroutines creating the initial seeds and the "pseudorandom" numbers use the coding of Nishimura. (The only adaptation was the introduction of a subroutine argument list and the ability to change the seed for the MLGC generating the 624 initial seeds.) As already stated, the effect of using a different MLCG from that chosen by Nishimura for producing the initial set of 624 seeds was not considered in this study. The

choice of this MLGC was probably based on its ease of implementation and speed as well as having a leading expert as its designer.

The samples created were as in the other studies; they each consisted of 1000 sample values from a negative exponential distribution. MT created the "pseudorandom" numbers and the negative exponential deviates were created by the "log" transformation. SETUP was run with the default values and thus the calibration of the tests previously made was use.

The probability that there would be any overlap (that is any of 624 sequence of pseudo random numbers created would be the same as one of the sets of 624 seeds) is so small that no test was made for any overlap.

5.3.5 Comparison of the Number of Highly Representative Samples Created

Name Used in Study	Number Selected	Fraction of Selected Seeds (or Sets) Having a Quality Measure Less than: 1500	1550	1600	1650	1700	1750	1800	1850	Maximum Deviation From Random	10% Critical Value
Real Random Numbers	78	0.08	0.18	0.35	0.59	0.81	0.99	1.00	1.00		
MT	71	0.11	0.15	0.30	0.46	0.76	0.87	0.96	1.00	0.13	0.20

Table 5.12: Distribution of the Quality Measure for Highly Representative Samples

The probability of obtaining 71 or less when the expected number is 78 is greater than 20% so the null hypothesis that the number of high quality samples

created is the same as with real random numbers cannot be rejected. Thus it may be assumed that the number of highly representative created by real random numbers and MT is the same.

5.3.6 Comparing the Quality of the Selected Samples with those from Real Random Numbers

To determine if the distribution of the quality measures in the selected samples was the same as obtained when using the real random numbers the Kolmogorov-Smirnov two-sample test (See Siegel 1956 pages 127-136 table M) was used. The null hypothesis that the two distributions are the same cannot be rejected even at the severe "confidence" limit of 10%. Thus the results from the MT RNG indicate the results are as if from real random numbers.

5.3.7 Comparison of the MT and MLCG Results

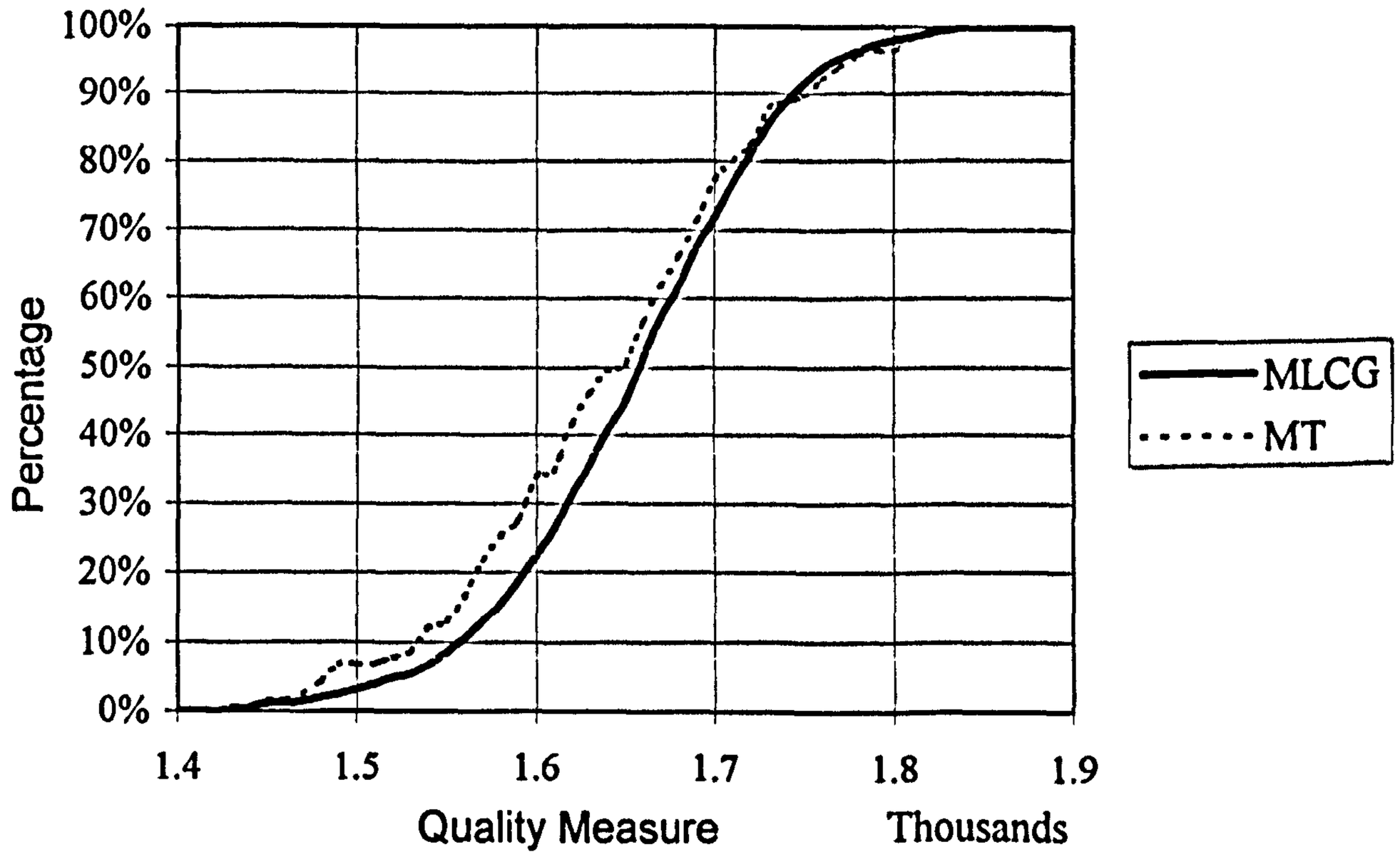
Since it was shown in section 5.2.9 that there was an improvement in the Quality Measures as the cycle size reduced it may be considered that the samples from MT could be of lower quality (that is larger numerical quality measure) than from the relatively small cycle MLCGs.

To determine if the results of the MT are inferior to those obtained from the MLCG previously analysed, the distributions of the quality measures were compared.

The results from selecting the highly representative samples from the 60000 samples were used. From the 60,000 samples created by MT, SELECT chose 115; for the 300,000 samples created by the five MLCGs (F&M, L'Ecuyer, Lewis, Flying and Killingbeck), 570 samples were chosen. These 570 were

considered as coming from the same distribution. The two cumulative distributions were calculated and the results compared.

The Kolmogorov-Smirnov two-sample test was again used. This time a single-tailed test was used. (Since there were a large number of samples, a transform was made on the maximum deviation of the MLCGs' distribution from the MT distribution such that the created test statistic follows the chi-square distribution with 2 degrees of freedom, (see page 131 of Siegel 1956). The null hypothesis was that the distributions were as if from the same distribution, while the alternative hypothesis is that MLCG were from a "better" distribution. The visual examination of the plots (Graph 5.3) clearly indicates that the alternative hypothesis is false.



Graph 5.3: Comparison of Performance of the MLCGs and MT

The largest deviation in the correct direction for the alternative hypothesis (a positive difference) is 2% at a Quality Measure of 1750. This is too small to see

on the plot. Using the tabulated values of chi-squared, the test shows that there is a 99.7% chance of getting this result or better from the null hypothesis and that the null hypothesis cannot be rejected in favour of the alternative hypothesis. Thus there is no evidence of the MT producing inferior samples.

A two-tail test also fails to find a significant difference; the null hypothesis that the distributions were the same would only be rejected at the 8% level. Thus the quality of the accepted samples from the MT generator was statistically the same as from the MLCGs,

Thus it has been shown that MT is not inferior to MLCGs in producing highly representative samples and indeed they appear to be equal in performance.

5.3.8 The Effect of Seed Selection

Using the results of the 60000 seeds the best 10 and worst 10 for the MT RNG are tabulated in table 5.13. The corresponding results from the five MLCGs are also shown.

As with the MLCGs, the difference between the results of the top seeds and bottom seeds is much greater than obtained by changing the RNG. In this case, to change between a MLCG and MT is at most 117 whilst changing from the top seed to the bottom of the values for MT is a change of over 2000, and even changing from the top seed to the 10^{th} (a change of 101) gave a difference greater than could be expected from changing generator.

RNG Rank	MT		F & M (MLCG)	L'Ecuyer (MLCG)	Lewis (MLCG)	Flying (MLCG)	Killingbeck (MLCG)
1	1429		1430	1448	1430	1442	1447
2	1449		1431	1484	1449	1479	1470
3	1466		1463	1504	1482	1499	1476
4	1473		1470	1508	1527	1544	1501
5	1479		1482	1514	1535	1548	1503
6	1488		1497	1515	1537	1548	1520
7	1488		1497	1517	1537	1548	1525
8	1489		1506	1518	1552	1551	1536
9	1518		1512	1533	1561	1560	1539
10	1530		1536	1534	1565	1561	1548
59991	3061		3061	3082	3031	3008	3006
59992	3070		3090	3083	3055	3051	3012
59993	3073		3092	3083	3055	3063	3022
59994	3091		3121	3102	3058	3067	3052
59995	3116		3154	3105	3106	3075	3076
59996	3160		3192	3168	3280	3108	3085
59997	3174		3203	3171	3281	3148	3091
59998	3205		3365	3175	3322	3188	3160
59999	3374		3504	3271	3373	3287	3376
60000	3442		3593	3339	3544	3553	3767

Table 5.13: Best and Worst 10 seeds for the MT Generator

5.3.9 Conclusion of the Study of the MT

The test again was able to determine a number of seeds that produce high quality samples.

It has been shown that even with its massive cycle size, the MT RNG produces as least as good samples of 1000 sample values as the MLCGs. But it does not produce better samples. When selecting a RNG based on an arithmetic process, the MT generator, because it lacks the shortcomings of the MLCGs, may be considered the preferred RNG of the two. However there is a concern that the MT generator requires processing time to initialise and is slower to produce the random numbers. This is discussed later when the timing of all the methods was considered.

5.4 EVALUATING DESCRIPTIVE SAMPLING

Descriptive Sampling is not really a random number generator but a method of constructing samples. The method of construction and the method of introducing randomness will now be described.

5.4.1 Description of Descriptive Sampling

Descriptive Sampling, or as it was originally called Selective Sampling, which was referred to in the Literature Survey, is a method of constructing a sample such that it will have the same distribution as the required distribution. That is the sample values have the same moments as the distribution of the variable. The problem in the method is to obtain a suitable sequence. In Saliby's description of the method the procedure is:

1. Determine the Number of Samples Values, N
2. Construct an Array with N values varying from 0.5/N to (N-0.5)/N in steps of 0.5/N
3. Convert each value to the required sample distribution with an inverse transformation

4. Sort the array into a random sequence or as it may be termed, "shuffle" the array.

Later Saliby and Paul (1993) moved the shuffle so that a single interchange, one step in the shuffle, is performed as the sampling takes place. This amended method is described later.

A similarity of the sample created by DS and the ramp used as one of the test patterns is that the sample values are calculated in the same way. Since both are a perfect sample in terms of sample values, both their χ_1^2 values are zero. The important difference is the sample values when used in Descriptive Sampling are shuffled in an attempt to give a representative sample in sequence as well as in sample values. If the shuffling were considered perfect then the values of χ_n^2 for all the values of n would be zero. This is not likely to be achieved.

5.4.2 Performance with Existing Tests

It may be noted that it is possible to change the sequence of steps to create the sample. If steps 4 and 3 were transposed, that is shuffling the unconverted values before the conversion, the resultant sample would be the same as before. This changed sequence within the algorithm provides after the sort, a set of what may be considered "pseudorandom" numbers that can then be tested using the established tests such as Knuth's or Marsaglia's list of tests (but not the "spectral" test which is only for MLCGs). However, if there are only a relatively small number of "pseudorandom" numbers then some tests will be impossible to apply.

Although from the method of construction of the sample some simple test such as a test for uniform spread of numbers will be met, it is very likely that the method of construction will also guarantee that such a set of "pseudorandom"

numbers would fail some tests, for example they would fail the tests in the "Monkey Tests" suite of Marsaglia (1993). (In these tests, ranges of numbers are assumed to represent individual keys on a keyboard, thus the sequence of numbers is converted to a string of letters, and these are considered to have been typed by a monkey). The tests consist of comparing the theoretical sequences of letters with the theoretical rate that the monkey should have typed. (Is "GOD" ever mentioned and does "SIN" occur too often?) As may be easily seen, in a sample of 1000 sample values constructed using Descriptive Sampling, there is no possibility of having more than one sample value in the range 0-0.001, while this Monkey Test would expect there to be a chance of 26.4% of having more than one value (probability of 1 or more from a Binomial Distribution of 1000 trials each with a chance of success of 1/1000). Similarly if all 1000 ranges of width 0.001 between 0 and 1 are considered, the probability if the "pseudorandom" numbers were true random numbers that every interval would have one occurrence has a probability of 10^{-432}. This is a near impossibility.

It may be noted that if N is the number of ranges (the number of sample values in the example above) then $(N!/N^N)$ is the probability that each range would have one occurrence. The $(N! /N^N)$ value can be derived by calculating the probability of success by considering the probability of placing sequential random numbers into an empty range. The probability of the first random number being so placed is (N/N), the second is ((N-1)/N) and if that was successfully placed the probability of the third would be ((N-2)/N) and so on.

But the sample was construct to have one value in each range. Thus it fails the test and therefore is shown clearly not to be a random sample. This is not a concern since our requirement is for a highly representative sample. The evaluation of the performance of Descriptive Sampling to produce highly representative samples will be made here by testing the set of samples it creates in the same manner as was performed with samples created by the real random numbers and the other generators.

5.4.3 Concerns with Descriptive Sampling

Since the requirements for many standard statistical tests will not be met if the samples are not drawn randomly, care would need to be taken when making any statistical analysis of the outputs from a simulation model in which this was not the case. Descriptive Sampling does not provide independent samples. The samples are related by having the exact mean and standard deviation, not the situation that would be obtained by independent samples. As was discussed previously the statistical analysis of simulation models is difficult and more so if the inputs are not independent samples and that statistical rigour may have to here be sensibly sacrificed for more dependable results. This is the case for all attempts to reduce variability in the output of simulation studies. Descriptive Sampling would therefore be unsuitable for some Monte Carlo studies, such as determining the distribution of a stochastic variable

A more fundamental problem is the requirement to sort the constructed values into a random sequence or in terms of a deck of cards, to shuffle them. Knuth stated that the fundamental problem of shuffling into random order can be seen if it is remembered that the number of possible permutations for the results of a random shuffle is massive. For 1000 samples there are 1000!, or over 10^{2567} possible permutations. If a billion permutations could be created in one second it would still take over 10^{2550} years to create them all. It can easily be seen that to shuffle by selecting at random from all the permutations by the use of a RNG such as an MLCG is impossible since they do not have such a range of random numbers. A RNG would need to be used with a very large cycle size.

An alternative approach is to have an algorithm that gives each sample value an equal chance of being put in any position in the sequence. Such an algorithm is used in the program SHUFFLE given in Appendix 1. This requires

a source of random numbers but "pseudorandom" numbers would suffice and the RNG need only have a cycle length of the number of sample values.

5.4.4 Selecting a Method to Shuffle

Searches of the literature, and direct requests for information have not provided any information on what shuffles have been proposed for Descriptive Sampling or to be more precise, since shuffling algorithms do exist (such as SHUFFLE), the source of randomness to be to be used by the shuffling algorithm. As stated above all shuffling algorithms require a source of randomness that is usually provided by a supply of "pseudorandom" numbers.

As previously mentioned Saliby and Paul (1993) suggested that the shuffle should not be made when the list of sample values is created but that one step of the shuffle should be made whenever a sample is taken. Thus in their implementation of the method, at the moment of taking a sample there is a random selection from the remaining values which one can visualised as being at the lower end of the list while those already chosen are at the top. The chosen value is swapped with the top of the unused values. The dividing line between chosen and un-chosen values then moves down one place. If more samples are required than are in the list all the values in the list are made available and the shuffle starts again with the top value in the list being swapped for the chosen value. When more than N samples are required there is a deviation from the perfect set of values but this will be investigated later. Their paper does not give information on how the random number is actually created. The assumption being the inbuilt random number generator of the simulation language is used. Brenner (1963) when describing a selection procedure (which he termed Selective Sampling), which is identical to Descriptive Sampling, used a look-up table created from the values in the book of tables created and published by the RAND Corporation (1955).

For use in the evaluation, the shuffling subroutine called SHUFFLE was written (see Appendix 1).

This algorithm operates in the same way and gives the same result as the procedure used by Saliby and Paul. (The version of SHUFFLE given by Durstenfeld (1964) appears similar and has no advantages.)

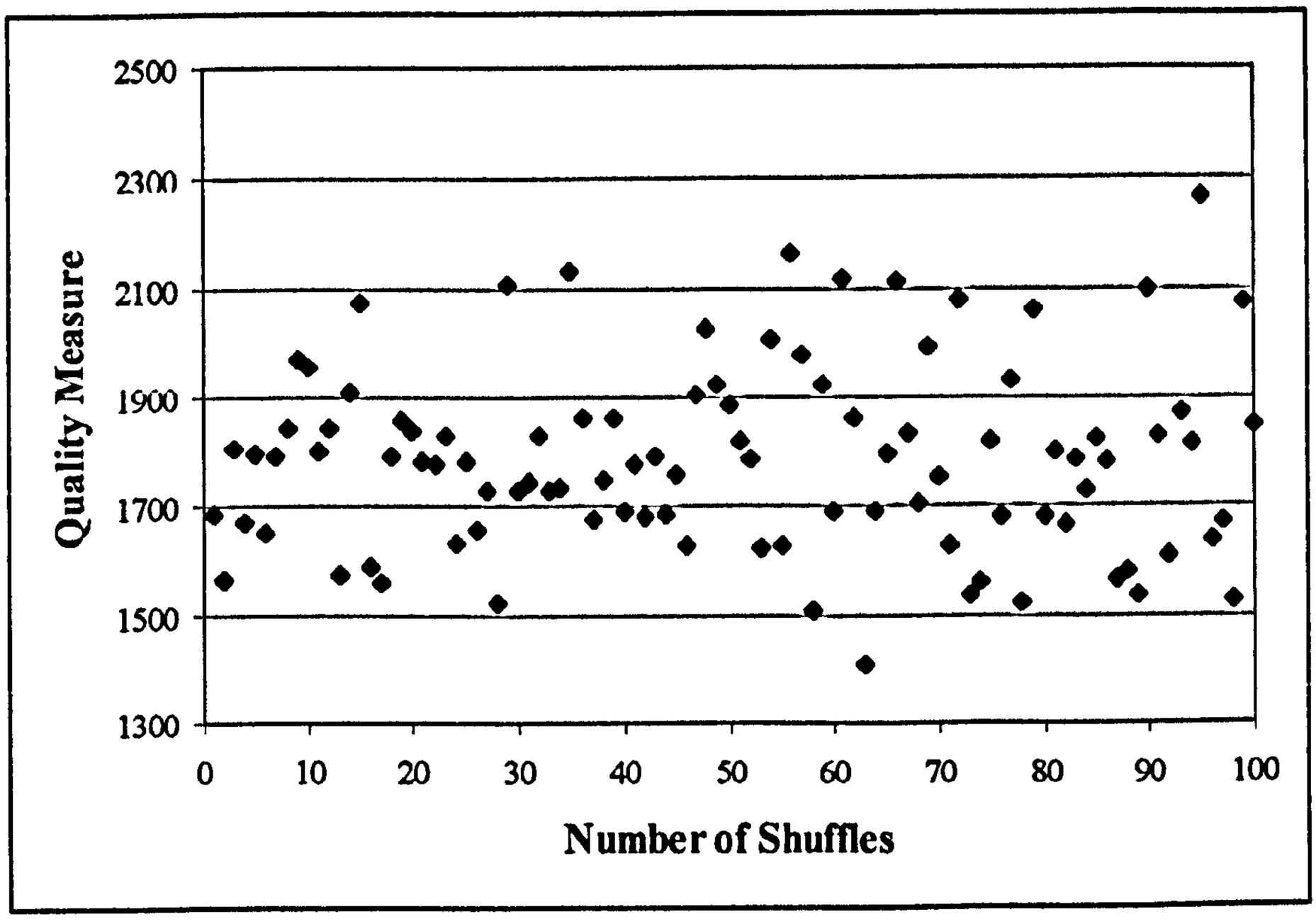
As already mentioned, if a source of randomness within SHUFFLE was a true random source, instead of the subroutine MCG, it can easily be shown that every member of the list would have an equal chance of being in any position in the shuffled array. However many permutations would be unwanted as sequences for our "highly representative sample." A practical shuffle that can create such "highly representative samples" is required. Thus the use of a MLCG to produce the randomness may be more suitable since, once a suitable combination of MLGC and seed is determined, the quality of the sequence of a certain number of values can be assured. For this study, the SHUFFLE program was used since any results may then be used within the implementation suggested by Saliby and Paul. The randomness required was provided by the MLCGs that have been used in the previous section. The MLCGs, seeds, and number of shuffles were examined to see which gave "highly representative samples."

The first set of tests was to determine if repeated shuffles would give better samples. The five MLCGs previously used were used to provide the randomness for the SHUFFLE program. The seed used for each MLCG was the seed that had the top ranking for its MLCG in the results in Appendix 3.

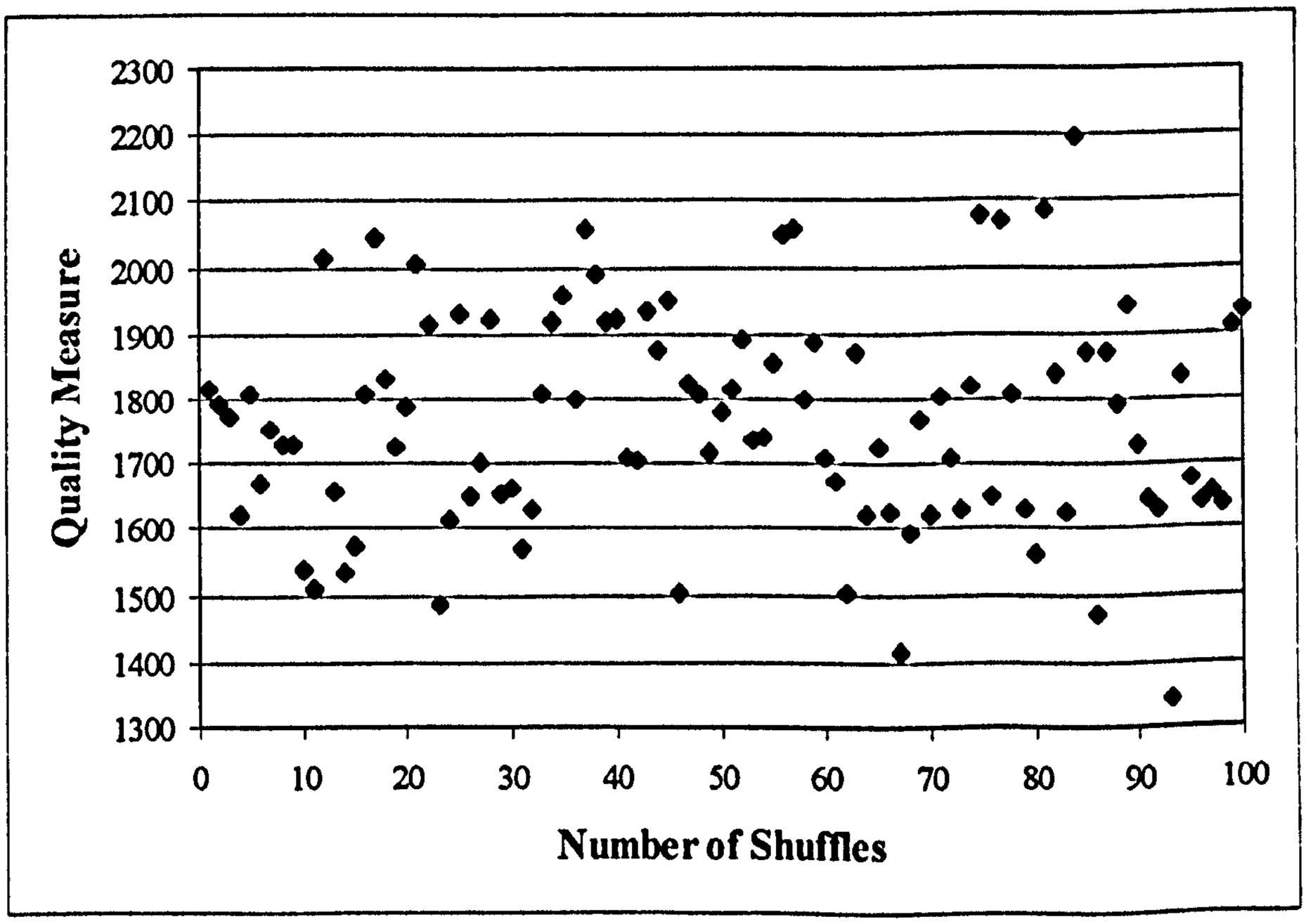
The sequences were tested after each shuffle. The sequences were shuffled up to 100 times. The list was sorted from its last sequence using the same seed.

5.4.5 Results of the Multiple Shuffles

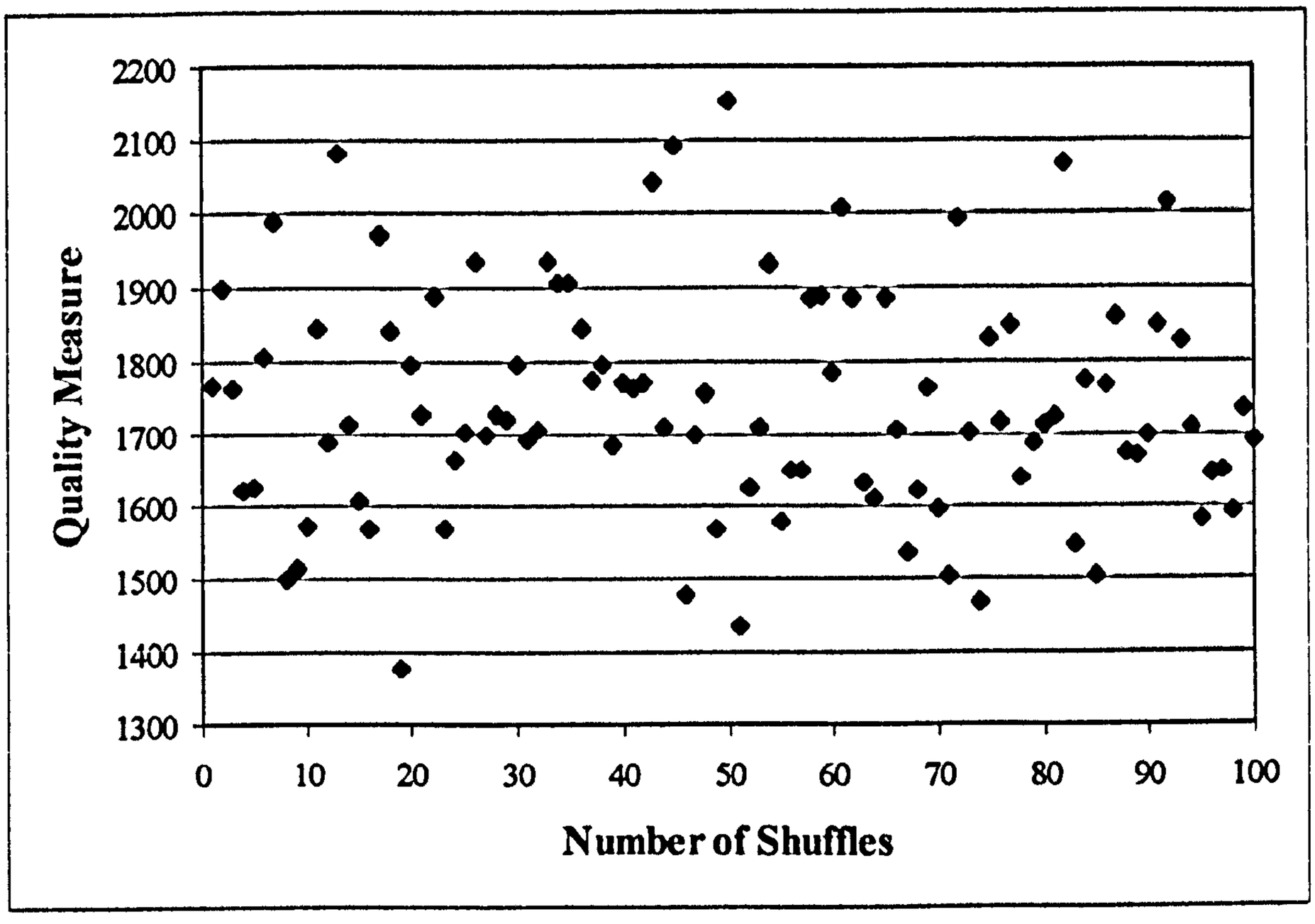
The results of the quality measures are given in the following five graphs.



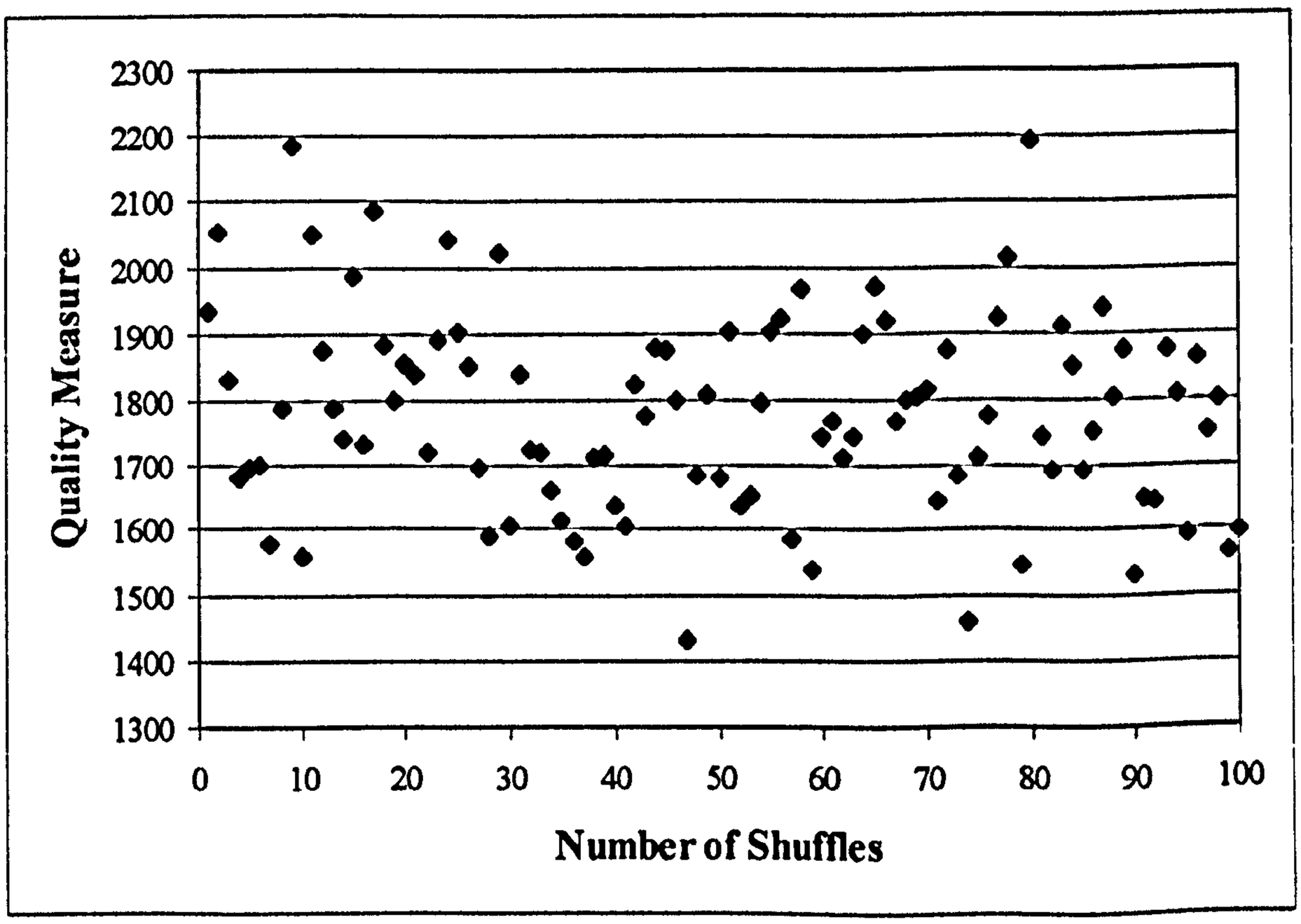
Graph 5.4: The Quality Measures from Repeated Shuffles using F & M



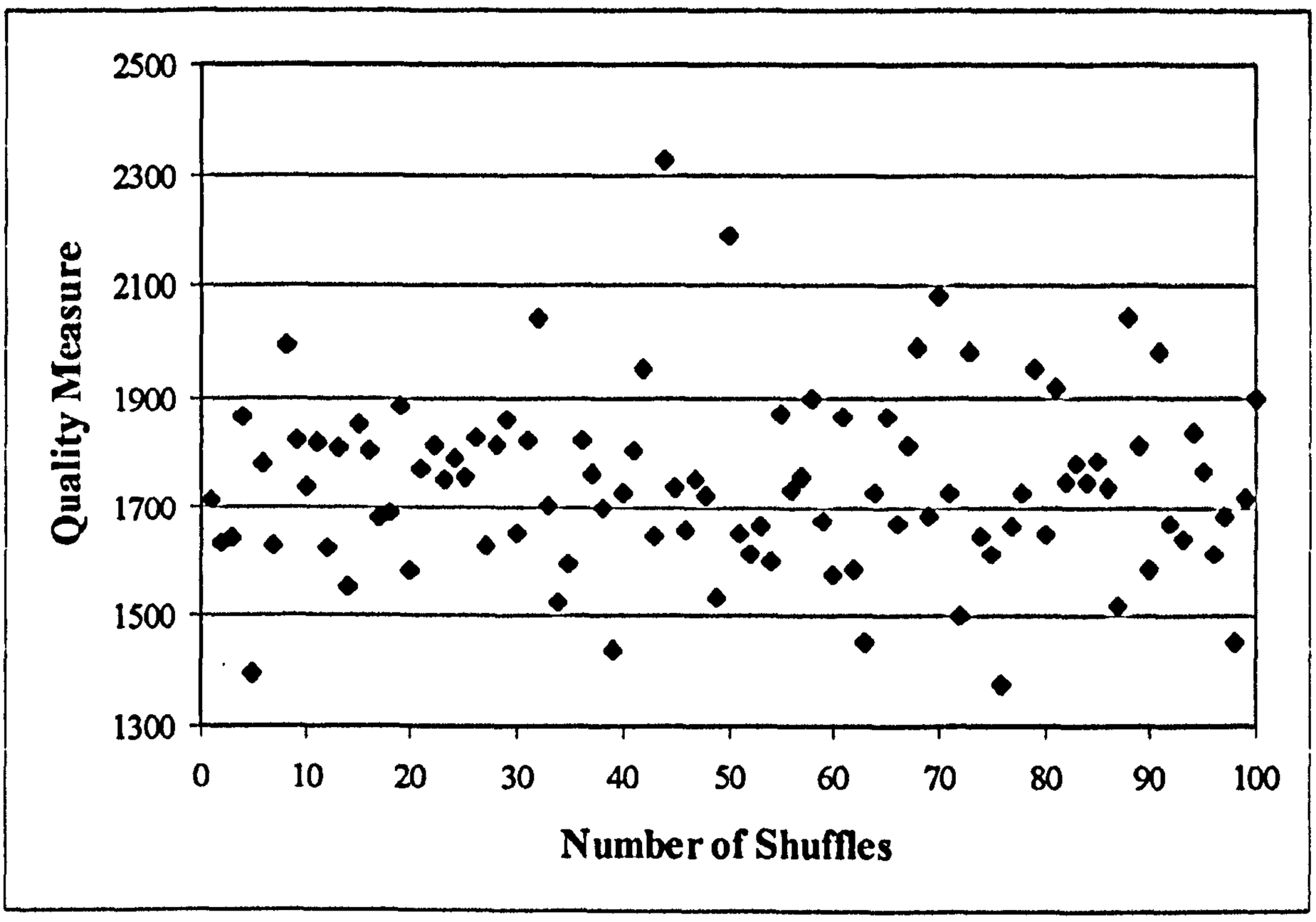
Graph 5.5: The Quality Measures from Repeated Shuffles using L'Ecuyer



Graph 5.6: The Quality Measures from Repeated Shuffles using Lewis



Graph 5.7: The Quality Measures from Repeated Shuffles using Killingbeck



Graph 5.8: The Quality Measures from Repeated Shuffles using Flying

The results clearly show that there is no guaranteed improvement by repeating the shuffle (often it became worse). Table 5.14 gives the best number of shuffles and the worst. There is no way to predict the number of shuffles to produce the best sample for a RNG and seed combination.

RNG used in SHUFFLE	Seed	Number of Shuffles		Quality Measure	
		Best	Worst	Best	Worst
F & M	52260	63	95	1411.9	2266.5
L'Ecuyer	33180	93	84	1343.1	2196.5
Lewis	36670	19	50	1377.9	2152.9
Flying	8528	76	44	1375.3	2326.4
Killingbeck	36120	47	80	1430.1	2193.2

Table 5.14: Summary of the Results from Repeated Shuffles

Thus in the investigation of the effectiveness of various MLCGs and seed combinations when used in the program SHUFFLE a single shuffle was used.

5.4.6 Performing the Tests

The tests were designed to determine:

- If DS produce the same number of highly representative samples as the real random numbers. This was performed using DS to create 40,000 samples of 1000 sample values. Five sets of 40,000 samples were created using the five MLCGs that passed the spectral test, each time using seeds of value 1 to 40000. The numbers of samples selected by SELECT were compared with the number selected from the 40,000 samples created from the real random numbers, that is 78.

- Does DS produce the same quality of samples as the real random numbers. This was determined by comparing the five distributions of the quality measure of the selected samples from the DS generator with the distribution when the real random numbers was used.

- If the choice of MLCG in the SHUFFLE algorithm is important. In this case the number of samples created using DS was increased to 60,000. Thus seeds of 1 to 60000 were used in the MLCGs used in SHUFFLE. The distributions of the Quality measures of the samples selected by SELECT were compared with their average distribution. Thus any MLCG that was superior was able to be detected.

- If the seed selection is more important than the choice of the MLCG used in SHUFFLE. This was determined by measuring the difference in Quality Measure from changing MLCG against the change that occurs by changing seeds.

The tests were performed using DESCTST, which in turn called subroutine DES that includes a call to SHUFFLE. (See Appendix 1)

5.4.7 Evaluation of DS's ability to produce the same number of highly representative samples as the real random numbers

In order to compare the performance of DS with the results from real random numbers, sets of 40000 samples of 1000 sample values were created. As previously stated, the five originally chosen MLCGs were used as the source randomness for each set with seed values of 1 to 40000. The program SELECT, again with the 50% "confidence" level filter, was used to determine the samples that were considered highly representative. Table 5.15 gives the number of selected samples and the number of times certain quality measures were obtained. It clearly shows that DS produces a higher number of selected samples than when using real random numbers. To determine if the rate of creating the highly representative samples using DS is statistically the same as for real random numbers a critical value has first to be calculated. If each sample created is considered as an independent trial, then the chance of getting 106 samples or more selected as "highly representative" (successes) from 40,000 samples (or trials), given a success rate of 1/500, is less than 0.01%. In this case the numbers selected are all greater than 106. Therefore it may be stated with confidence that DS produces more quality samples than obtained either from real random numbers or from MLCGs and MT. (Since MLCG and MT give identical distributions to the real random numbers.)

Name Used in Study	Number Selected	Number of Seeds used in SHUFFLE (or Sets of Real Random Numbers) Having Quality Less than:						
		1500	1550	1600	1650	1700	1750	1800
Real Random Numbers	78	6	14	27	46	63	77	78
F & M	291	213	266	287	291	291	291	291
L'Ecuyer	306	223	282	305	306	306	306	306
Lewis	271	194	245	270	271	271	271	271
Flying	286	206	251	279	286	286	286	286
Killingbeck	254	171	224	248	254	254	254	254

Table 5.15: Distribution of the Quality Measures of Selected Seeds

5.4.8 Comparing the Quality of the Selected Samples created by DS with those from Real Random numbers

Table 5.15 also indicates that the quality from DS was superior to that from real random numbers. It is so superior any statistical test is redundant.

This clearly shows the non-random nature of the samples and the benefit of forcing the χ_1^2 term to zero. DS is a significantly better method of producing highly representative samples.

The next investigation was to determine if the choice of MLCG as the provider of randomness for the shuffle was important.

5.4.9 Effect of the Choice of MLGC for use in the SHUFFLE

In this study again the seeds of range 1-60000 were used and a single shuffle was performed. Samples of 1000 sample values were again created. All the five MLCG in the above table were used to provide the randomness for the shuffle.

Again SELECT was used to determine the required samples. The distributions of the quality measure were determined. They are shown in table 5.16.

MLCG used in SHUFFLE	Number of Samples	Quality Measure							
		Below 1301	1301 to 1349	1350 to 1399	1400 to 1449	1450 to 1499	1500 to 1549	1550 to 1599	Above 1600
F & M	416	17	31	60	95	103	74	29	7
L'Ecuyer	435	14	39	67	95	109	79	31	1
Lewis	407	12	20	57	86	104	88	33	7
Flying	425	10	29	71	84	108	71	43	9
Killingbeck	402	8	26	65	88	102	72	35	6

Table 5.16: Distribution of the Quality Measure for the Selected DS Seeds

Using the average result to create the "expected values", the following χ^2 values were created:

MLCG used in SHUFFLE	χ^2 Values
F and M	3.71
L'Ecuyer	7.74
Lewis	5.45
Flying	5.53
Killingbeck	1.74

Table 5.17: Calculated χ^2 Values for the Different MLCG used in SHUFFLE

For 7 degrees of freedom the critical value at a severe "confidence" level of 10% is 12.02. Thus the null hypothesis that there is no difference cannot be rejected and thus it can be stated that the choice of MLCG, for use in SHUFFLE, from the five used was not important.

The next investigation was to determine if the choice of seed was important.

5.4.10 Effect of Changing the Seed

The data created for the previous analysis was used. For this analysis the best and worst performing seeds were determined. The selection of the worst seeds was made by using BAD100 (see Appendix 1).

The best 10 and worst 10 seeds for each of the MLCGs are shown in table 5.18.

MLCG Rank	F & M	L'Ecuyer	Lewis	Flying	Killingbeck
1	1231	1221	1172	1119	1172
2	1232	1246	1223	1155	1239
3	1241	1249	1258	1237	1247
4	1253	1250	1265	1265	1261
5	1258	1250	1272	1277	1263
6	1260	1254	1284	1279	1264
7	1269	1270	1297	1288	1285
8	1270	1272	1300	1291	1289
9	1277	1272	1307	1296	1293
10	1277	1273	1307	1298	1296
59991	2427	2441	2430	2424	2424
59992	2440	2452	2460	2433	2433
59993	2444	2453	2463	2438	2438
59994	2449	2457	2469	2445	2445
59995	2450	2488	2481	2452	2452
59996	2459	2501	2492	2460	2460
59997	2461	2511	2509	2464	2464
59998	2478	2517	2541	2468	2468
59999	2533	2585	2560	2473	2473
60000	2568	2591	2574	2487	2487

Table 5.18: Worst and Best Seeds for Each MLCG used in SHUFFLE

Table 5.18 shows that the choice of MLCG to provide the randomness to the SHUFFLE procedure is not as important as the choice of seed. The change in Quality Measure at the same rank for a change in the MLCG is at most 112 whilst the difference from the top seed to the worst seed for a particular MLCG is at least 1314. The tables of the seeds that produced sorted sequences that passed the 50% confidence filter (and thus were "selected") and then sorted in quality measure sequence are given in appendix 3.

A summary of the number of seeds that passed, and their highest and lowest quality measure are shown in table 5.19. The top 120 seeds for the MLCG listed are given in Appendix 3.

MLCG used in Shuffle	Number of Seeds	Quality Measure	
		Highest	Lowest
F & M	416	1231	1641
L'Ecuyer	435	1221	1612
Lewis	402	1172	1675
Flying	425	1119	1651
Killingbeck	407	1172	1637

Table 5.19: Summary of the Results of Selecting Seeds for DS

5.4.11 Summary of the Evaluation of Descriptive Sampling

As may be seen from table 5.19, Descriptive sampling gave significantly more and better quality of samples than can be expected from the real random numbers or processes that mimic true randomness (MLCG and MT).

It is not sensitive to the choice of the five MLCG used to provide the random numbers required by the shuffle algorithm but it is sensitive to the seed used, at least when 1000 sample values are being created.

5.7 SELECTION FROM DS, MT, AND MLCG

As stated in the introduction, the aim of this thesis is to select methods that will enable inexperienced users to produce accurate simulations. To provide further assistance, a computer program was developed that will assist the user to select seeds and, during the time before the implementation of the RNG within the specialised simulation software, to create files of random deviates. Most

specialised software will allow either the introduction of a user defined RNG or a file of random deviates.

The decision as to which of the RNGs should be used must be based on their ease of use, their practicability, as well as their capability to produce accurate simulations. All the generators examined in the previous section are capable of producing highly representative samples. The diagram 5.1 on page 141 illustrates their relative performance. (Note: The diagram has the highest quality at the top of the drawing.) It shows that DS is the best performer. This can be accounted for by the fact that it does not attempt to mimic randomness but "constructs" its samples. The diagram shows that 90% of the acceptable samples created by DS were broadly as of as high or higher quality as the top 10% of the acceptable samples provided by MT and the MLCGs. The highest quality sample from DS with a quality measure of 1221 easily outperforms the best sample from MT that had a quality measure of 1429. The number of samples produced by DS was also greater than three times than the number produced by MT or any of the MLCGs.

The diagram also shows that there is little difference between the performance of MT and the two MLCGs shown on the diagram. (It will be remembered that there was little difference in the performance of the seven MLCGs examined.)

There remain other aspects that need to be considered. These are:

- Speed of Operation
- Requirement for Computer Memory
- Sensitivity to the Accuracy of the Forecast of the Number of Samples Required
- Ease of Use and Acceptability

The first to be considered is the relative speed of operation.

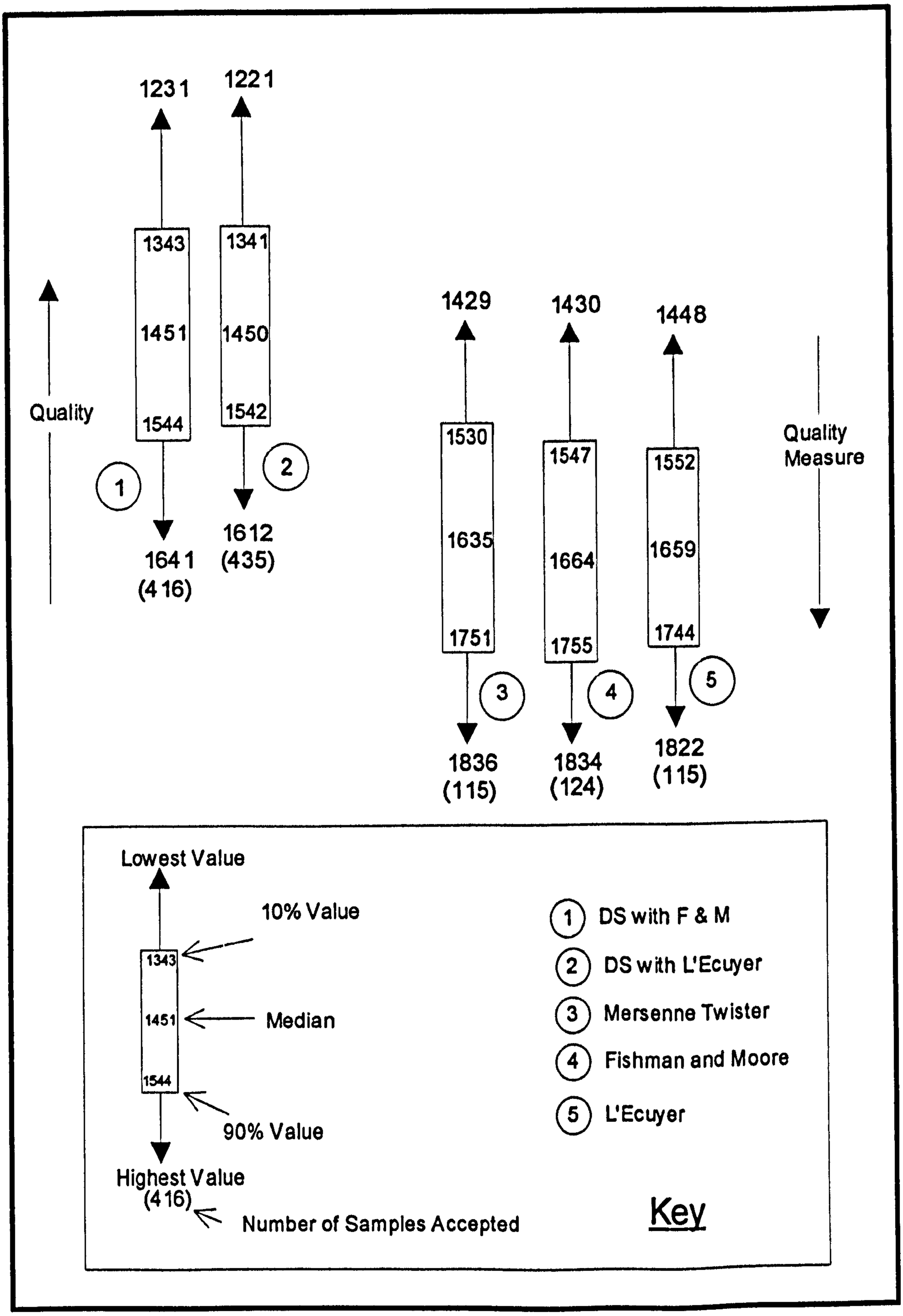


Diagram 5.1: Illustration of the Performance of the RNGs

5.7.1 Relative Speed of Operation

The usefulness of providing any timing measurements is limited since there are many factors that will affect the timings. Some of the more important factors are the computer's processor speed, type of memory and memory management, and the efficiency of the coding. However if all the elements are kept the same for all the generators such measurements can give an indication of the relative speed.

All the timings reported here were performed on a relatively slow computer having only a 600MHz AMD Athlon processor with 128MB SDRAM. The operating system was MS DOS operating under Windows ME and the algorithms were coded in Microsoft FORTRAN. It may be reasonably claimed that any future implementation will beat the timings given.

The timings in table 5.20 are for the creation of 1,000,000 random numbers. There are different timings given for different number of streams. The values in the table are not meant to be representative of "normal" simulations studies but to illustrate the fact that both DS and MT have a set up cost for each stream and thus, with the same total number of random numbers generated, the total elapse time for these generators increases with the number of streams.

RNG	Number of Streams	Number of Random Numbers per Stream	Elapse Time (seconds)
MLCG (F & M)	100	10,000	1.15
	1000	1,000	1.15
MT	100	10,000	134.2
	1000	1,000	134.7
DS	100	10,000	2.4
	1000	1,000	3.6

Table 5.20: Timings for the RNGs to Create One Million Random Numbers

As may be seen from table 5.22, MT is the slowest by far. The time to produce 1000 samples in 1000 streams is 100 times longer than the MLCG and 40 times DS. However in absolute terms, the timing of 134.7 seconds is not a real concern and thus it may be stated that none of the timings is so long as to make the speed of calculation a deciding factor in selecting a RNG.

5.7.2 Memory Requirements

The processing code for all the RNG may be considered to be small. If the memory to hold the program and the seeds is considered, but not the data values, the largest requirement for memory is for MT. MT requires 1.2KB while the MLCG requires only 128 bytes. However the greatest requirement for memory is DS as it does require storage of an array of data values. This would have been a serious concern when limitations on memory sizes were severe and accessing data from storage devices was slow, but in modern processing environments such requirements are easily met even with desktop computers. It would be simple to develop an implementation that did not need to hold the full sample values but simply a list of "pseudorandom" numbers that could be held in a more condensed form. (The speed of conversion is now very fast.) If the number of samples for any stream is less than 64000 then each number could be held in two bytes. Thus a model requiring a total of 10,000,000 sample values would need (what in modern processing terms would be considered as "only") 20MB of memory for these data. Modern memory control software will enable such data to be held on disk until needed and will use disk caching to reduce the time to retrieve the data.

In order to give a measure of the time penalty that could be expected from access a file of data, an experiment was performed. Using a FORTRAN program an unformatted file of 10,000,000 numbers held as INTEGER*2 (that is 2 bytes per number) was created both on Hard Disk and a ZIP drive. With the

numbers written as individual records there is an overhead leading to the file being 40,000,006 bytes long. The file was read using a second FORTRAN program. The time to read all the numbers on the file when on a hard disk took 14.72 seconds. When the file was on a ZIP drive, which is a much slower device, it took 3 minutes 55.2 seconds to be read. If the records on the file were written and read 1000 numbers at a time such that a record in the file contained 1000 numbers, the space required reduces to 20,323,034 bytes and the time to read from the hard disk reduces to 2.25 seconds and if written and read from the slower ZIP drive it becomes 1 minute 58.5 seconds. The programs were run using a 600MHz AMD Athlon processor with 128MB SDRAM and under MS DOS operating within Windows ME.

These experiments indicate that the penalty of having the "pseudorandom" numbers for use in DS held in a file is acceptable especially if the file is held on a hard disk.

5.7.3 Sensitivity to the Accuracy of the Forecast of the Number of Samples Required

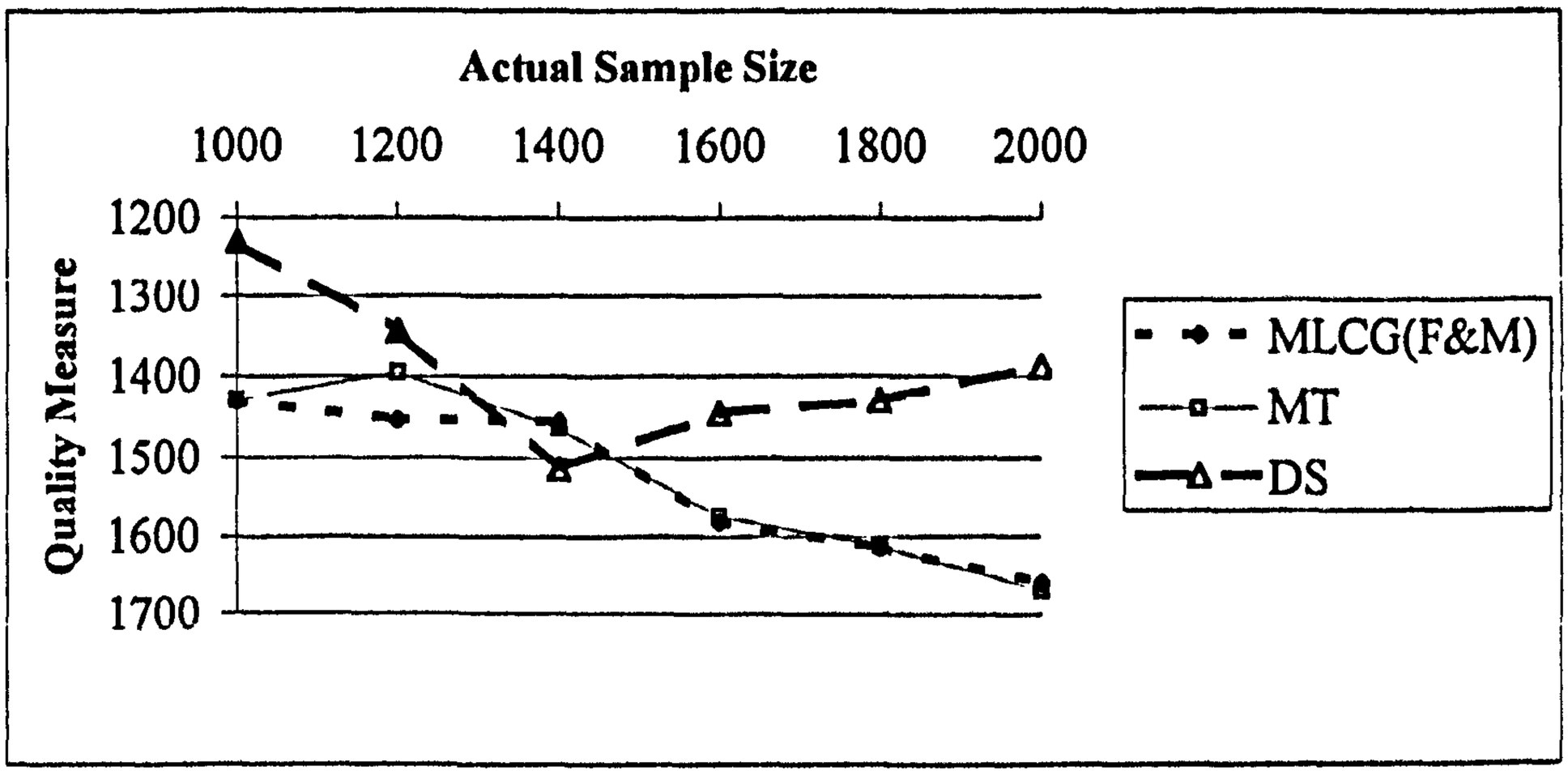
In all the types of RNGs examined and DS, it has been shown that the choice of seed is important. In MLCGs it is the actual seed, for MT it is the seed to enable the large set of seeds to be generated, and in DS it is for initialising the MLCG used in the shuffling algorithm. The selection of the best seeds has in all the studies been based on the number of sample values required. If the performance of a RNG to create a good sample were to drop dramatically if the forecast of the number of sample values required was inaccurate, then this RNG would not be a suitable choice, as it would not meet the requirement of being a robust procedure suitable for the non-expert.

Graph 5.9 shows the deterioration of the quality measure of the samples due to a poor forecast of the required number of samples. It assumes a forecast of 1000 was made but the actual requirement was higher. For each generator the best seed to provide a sample size of 1000 was chosen and a set of samples of sizes of 1000, 1200, 1400, 1600, 1800, and 2000 were created using that seed and the quality measure for each sample was measured. In this study it has been assumed that the simulation run will be such as to met or exceed design requirements and thus samples sizes smaller than that forecast have not been considered. This analysis must be one for further research.

The Fishman and Moore generator, which in the previous analysis had the best two seeds (52260 and 56242 with quality measures of 1430 and 1431 respectively), was used as the example of the MLCG.

The MT and the MLCGs did not need any extra coding to enable more samples than the forecast number to be generated. In DS the forecasted number of sample required is the number of sample values created and if more were required the array of created values would need to be sorted for a second time. In the implementation suggested by Paul and Saliby, there would be a need to reset the pointer to the top of the array of created values and again perform a swap for each value selected. In the implementation used in this study, the array was sorted at the beginning of the simulation and, if afterwards a sample value was required in excess of the forecast number, then the array was sorted when the first sample in excess of the forecast was requested. Subsequent sample values are then taken in sequence from the top of the re-sorted array.

The results are shown in graph 5.9 on the following page. For all but a short part of the range of 1000-2000 sample size, DS is still the best. (The graph 5.9 has been plotted such that the highest quality is at the top of the graph.) For both the MLCG and MT there was a steady deterioration in the quality of the samples as the forecast became more inaccurate.



Graph 5.9: The Effect of Error in Sample Size on the Quality Measure of the Sample

The result for the DS indicates a dip and climb. The DS result reflects that at sample size 2000 all the 1000 created sample values have been used twice and at that point χ_1^2 again becomes zero. It will be remembered that it was originally zero at a sample size of 1000. The quality at 2000 is not as good as at 1000 for as was seen earlier, a second shuffle does not guarantee an improvement in the sample and using a seed chosen to give a highly representative sample it would be expected that the second shuffle would lead to less representative sample.

In many cases forecasts of the number of samples values required by the simulation will be quite accurate. The actual number of sample values required and thus the ability to accurately forecast this number will vary according to the generator selected. As an example of how the accuracy of the forecasts of sample sizes can vary according to the RNG selected, is the case where breakdowns are modelled to occur as a function of the production time. If the run was for 1000 units of time and the average time between breakdowns was 1, then using the generator "F and M" with the seed value of 52260, (the best seed for 1000 sample size in the range 1-60000) the actual number of sample

values taken during the 1000 units of time is 996. With DS the number will be exactly 1000 whatever the choice of seed. In a situation where the number of sample values is dependent on some value being measured and this value is difficult to forecast accurately (e.g. number of parts processed), then the forecast of the number of sample values required will naturally also be more inaccurate. In many cases there will a stated design requirement for this value (e.g. number of parts required to be produced) and this may be used as the forecast. Finally if all methods fail, a short run may give an estimate from which an acceptable forecast may be made. In practice this is unlikely to be more inaccurate than the values that have been used in the study to measure the sensitivity to the forecast, of which the results were reported earlier (graph 5.9).

5.7.4 Must be easy to use and be acceptable

None of the methods of producing samples that have been examined are difficult to use. MT has been made easy by the implementation suggested by Nishimura. There are two points to consider when considering how easy the method is to apply. Since the seed choice has been shown to be important then it must be easy to determine suitable seeds. The test described previously is easy to run but the advantage obtained with DS is that there are more than three times as many suitable seeds in any range of possible seed values.

It will be seen later that when using DS it is easier to predict how many extreme sample values will actual occur during a simulation.

One advantage with DS is in making the results from simulation more acceptable. In particular when DS is used to generate the sample values for breakdown, since the sample values were constructed to have the required mean, the actual breakdown rate in the simulation is exactly the requested rate.

5.8 CONCLUSION

DS with a MLCG as the random input to SHUFFLE is the first choice:

- it produces the best samples,
- its speed of operation is acceptable,
- it is easy to implement and to use,
- it enables results to be obtained that have used the exact design values for such factors as breakdown rate.

Its disadvantage is that it requires more memory, which for the very large models may require storage on hard disk but if the file is held on hard disk it is an acceptable small increase in elapsed time.

It is also a major disadvantage that at this time (2003), none of the specialised simulation software offer, as a built in feature, the use of Descriptive Sampling. However most of the specialised simulation software packages are able to accept a file of random deviates thus enabling Descriptive Sampling to be used until the packages make it available.

The analyst needs to know how to set the model so that only valid measurements are made. This will be achieved by developing a simple method that will enable the analyst to determine the initial setting when measurement should commence.

6 DETERMINING THE INITIAL SETTING AND WARM UP PERIOD OF THE SIMULATION MODEL

6.1 DETERMINING THE INITIAL SETTING OF THE SIMULATION MODEL

As was discussed in the literature survey, the correct initial setting of the computer model so that accurate measurements can be obtained has been the subject of much debate. In the literature review it was noted that it is standard practice, in cases of non-terminating simulations, to run the model until it reaches "steady state" before any measurements are made. Thus using this procedure the initial condition for the measurements are the conditions the system happens to reach at the end of this warm up period.

As previously discussed there are two main reasons given to justify the requirement for an initial warm up period during which no measurements are made. The first arises from the fact that at the start of the simulation many computer models will be empty; that is the buffers will be empty and the machines will all be idle and, if modelled, the labour will be unassigned. This will often represent the system being in a state that will never be achieved during normal production. In such cases it may be legitimately considered that no meaningful measurements can be taken until the model represents the facility in a "normal" condition. For example, if the production facility is empty and the rate of production is to be measured then it may take some simulated hours (although only a short amount of actual computer time) before the first product is made and it is doubtful that anyone would suggest that this period should be included in the measurement.

It may reasonably be stated that the only time that one would commence measurement at the start, when the model of the production facility is empty, is

if the study is treated as a terminating simulation and the selected performance criterion is the time it takes to complete the manufacture of a certain large number of parts starting with such an empty system. When deciding on the best method to determine when to commence readings, it is being assumed that the requirement is to measure the "on going" production capability. Thus a warm up period is required so as to obtain "normal" conditions.

The second reason given to support a warm up period is that it is insufficient just to fill the system and obtain "normal" conditions, but that it is necessary to extend the warm up to enable the system to achieve a state where if measurements commence they will be undistorted by the starting condition and thus according to this argument, be valid and unbiased and give a measurement reflecting a true on-going capability. This is the state referred to as "steady state". The term is borrowed from other fields. This term is frequently used in physical experiments where external factors are artificially held constant and the equipment is allowed to settle until all measurements appear steady thus enabling a simpler mathematical analysis to be applied. But even in such experiments the term "steady state" is challenged. Feller (page 395, 1968) found the use of the term in such circumstances misleading.

Feller who actual uses the term equilibrium rather than "steady state" (in his index the entry steady state refers the reader to equilibrium distribution), when discussing a stationary Markov chain process and the invariant distribution obtained after a long period states:

> "....called *equilibrium distribution.* Unfortunately this term distracts from the important circumstances that it refers to a so-called *macroscopic equilibrium* that is, an equilibrium maintained by a large number of transitions in opposite directions."

In the industrial situation the movements of the parts, vehicles and such items are the "transitions" and they are not a large number compared with the number of electrons in the physical experiment. However there are many non-industrial systems where there are a large number of items. A number of such systems are of the form of "birth and death" processes where individuals are "born", live through many states, and finally "die". If there are a large number of individuals and equilibrium exists, then the number in each state will be approximately the same through time. No individual is in equilibrium only the whole system. But many industrial situations do not have a large number of items and cannot be accurately described as reaching such equilibrium.

The property of certain Markov chains to move to an equilibrium distribution, which is independent of the initial conditions, is the basis on which the statement that measurements should only be made when the simulation reaches this equilibrium or "steady state" appears to be made. The justification is if the simulation acts like a Markov chain, the behaviour of the simulation from that point must be independent of the starting condition.

If this concept of a "steady state" existing at a certain instance in time is accepted, it follows there ought to exist a state that, if the system could be set to it, then the system would be at steady state and thus no warm up would be required. In the physical experiment it may well be that the temperatures of the various parts of the apparatus could be set to their equilibrium or "steady state" temperature and thus no more time would be required to obtain an equilibrium. In the birth and death model the population numbers for each state could be set to their equilibrium number and thus "steady state" would be achieved.

In the simulation context, it may be reasonably assumed that the time required to move from an unusual condition would also be unnecessary as the system would already be in a "normal" condition.

One of the reasons that it is not be possible to set the system to such a state may be due to lack of knowledge. This appears to be the view of Kelton and Law (1985) as they stated,

> "The problem stems from the inability to initialize the simulation according to a steady state distribution, which is presumably unknown"

Given this belief that the inability to set the system to a steady state was due to lack of knowledge, Kelton and Law attempted, whilst investigating the transient behaviour of queues, to determine the best queue size to commence a study so as to reach the steady state in the shortest time. Again, in terms of the physical experiment, if the equilibrium temperatures are not accurately known, then the temperatures of the apparatus may be set to values close to what is considered to be the equilibrium temperatures and thus the time to obtain the real equilibrium values would be expected to be less.

It was first assumed that such a state exists and it is only lack of knowledge that stopped Kelton and Law setting the queue to this state. If however such a state does <u>not</u> exist then no warm up, however lengthy, would ever achieve it!

Kelton and Law investigated the transient behaviour of queues with random arrivals, negative exponential service time, and a number of servers. They measured the queue size by calculating the average time the parts had to wait in the queue. In this study, a simple queuing system was analysed. Queues are simple examples of sub-systems that appear within most production models. They exist in the target real life model of this research. Here they are in front of machines on the transfer line, the queues acting as buffers where parts wait to be processed.

The particular form of queue being considered here is one where there is a single machine with a constant cycle time (time to process a part) and that the

parts arrive in a random fashion due to disturbances in the progress of the part caused by failures during earlier processing. This is reasonable approximation to the situation in the real life model.

The measure of queue length used in the study reflects this type of production and is the queue length at the time a part finishes processing. This is relevant when the design problem is to determine the size of buffering required as at the point when the machine finishes processing a part the queue is at it longest.

However as already described, this was not the form of queue investigated by Kelton and Law nor did they use the same measure of queue size. Their queue with its negative exponential service time is not a realistic assumption for a transfer line or other production systems where the cycle time of machines is precise.

The steady state properties of this simple queue are particularly easy to analyse. The queue size is the only stochastic variable and this is what must be set to initialise the system into a "steady state."

6.1.1 The Long Term Expected Queue Size and Steady State Probabilities

As stated above, the queue size measured was that of the queue when an item had just completed being processed, the processing time was constant and the arrival of the parts was random. This enables a relatively straightforward analysis to be made (this is shown in Cox and Smith, 1979).

Cox and Smith give the steady state expected size of this measure of queue size as:

$$L_L = \frac{\rho\left(1-\frac{1}{2}\rho\right)}{1-\rho}$$

Where

L_L = the average queue left when processing of an item is complete and before a part leaves the queue.

ρ = machine utilisation $\rho < 1$

For ρ=0.9, the expected queue size is 4.95.

Cox and Smith also give the equations that define the steady state probabilities. Using their equations a more convenient set of recurrence equations was derived from which an EXCEL program was created to calculate the steady state probabilities.

The expressions derived were:

$$Z(0) = 1 - \rho$$

$$Z(1) = \frac{Z(0)[1 - p(0)]}{p(0)}$$

$$Z(2) = \frac{Z(1) - [Z(0) + Z(1)]p(1)}{p(0)}$$

$$Z(n) = \frac{Z(n-1) - [Z(0) + Z(1)]p(n-1) - \left\{\sum_{i=2}^{n-1} Z(i).p(n-i)\right\}}{p(0)} \quad n \geq 3$$

Where:

Z(n) is the steady state probability of a queue size of n.

p(i) is the probability of i arrivals during the processing time.

ρ is the machine utilisation.

The values for 90% machine utilisation are given in table 6.1 for queue sizes (Q in the table) of up to 54. (The EXCEL program measures the probability of a having larger queue sizes, for example a queue size of 216 is 1.3×10^{-16}).

Q	Probability	Q	Probability	Q	Probability	Q	Probability	Q	Probability
0	0.100000	11	0.022002	22	0.002254	33	0.000231	44	0.000024
1	0.145960	12	0.017886	23	0.001832	34	0.000188	45	0.000019
2	0.137640	13	0.014539	24	0.001489	35	0.000153	46	0.000016
3	0.115050	14	0.011819	25	0.001211	36	0.000124	47	0.000013
4	0.093804	15	0.009608	26	0.000984	37	0.000101	48	0.000010
5	0.076255	16	0.007810	27	0.000800	38	0.000082	49	0.000008
6	0.061985	17	0.006349	28	0.000650	39	0.000067	50	0.000007
7	0.050387	18	0.005161	29	0.000529	40	0.000054	51	0.000006
8	0.040960	19	0.004195	30	0.000430	41	0.000044	52	0.000005
9	0.033296	20	0.003410	31	0.000349	42	0.000036	53	0.000004
10	0.027066	21	0.002772	32	0.000284	43	0.000029	54	0.000003

Table 6.1: Steady State Probabilities of The Remaining Queue (Q) with Machine Utilisation of 90%

From the table 6.1 the mode for this measurement of queue size can be seen to be 1.

6.1.2 Using a Markov Chain Model to Investigate Initial Queue Sizes

In order to determine the rate at which the queue will obtain steady state with certain initial queue sizes a Markov Chain model (see Feller 1968) was constructed. This was based on the transitional probabilities depicted in Diagram 6.1 (on Page 157) The program was written in FORTRAN and is listed as MARKOV in Appendix 1.

This is similar to a study by Kelton and Law (1985) but they used the "memorylessness" feature of the service time and arrival distributions to produce simple functions for the probabilities of average waiting times.

The first results measured were the changes in the expected queue size that occurred as the parts were processed for four starting conditions. The four starting queue used were 0, 5, 10 and 15. The results are shown in table 6.2 and in graph 6.1.

The results show that the expected queue size after 700 parts have been processed is equal to the long term expected queue size to an accuracy of two decimal places for starting queue sizes of 0, 5 and 10. At 900 parts processed, all the expected queue sizes for all the starting queues had become the long term expected queue size.

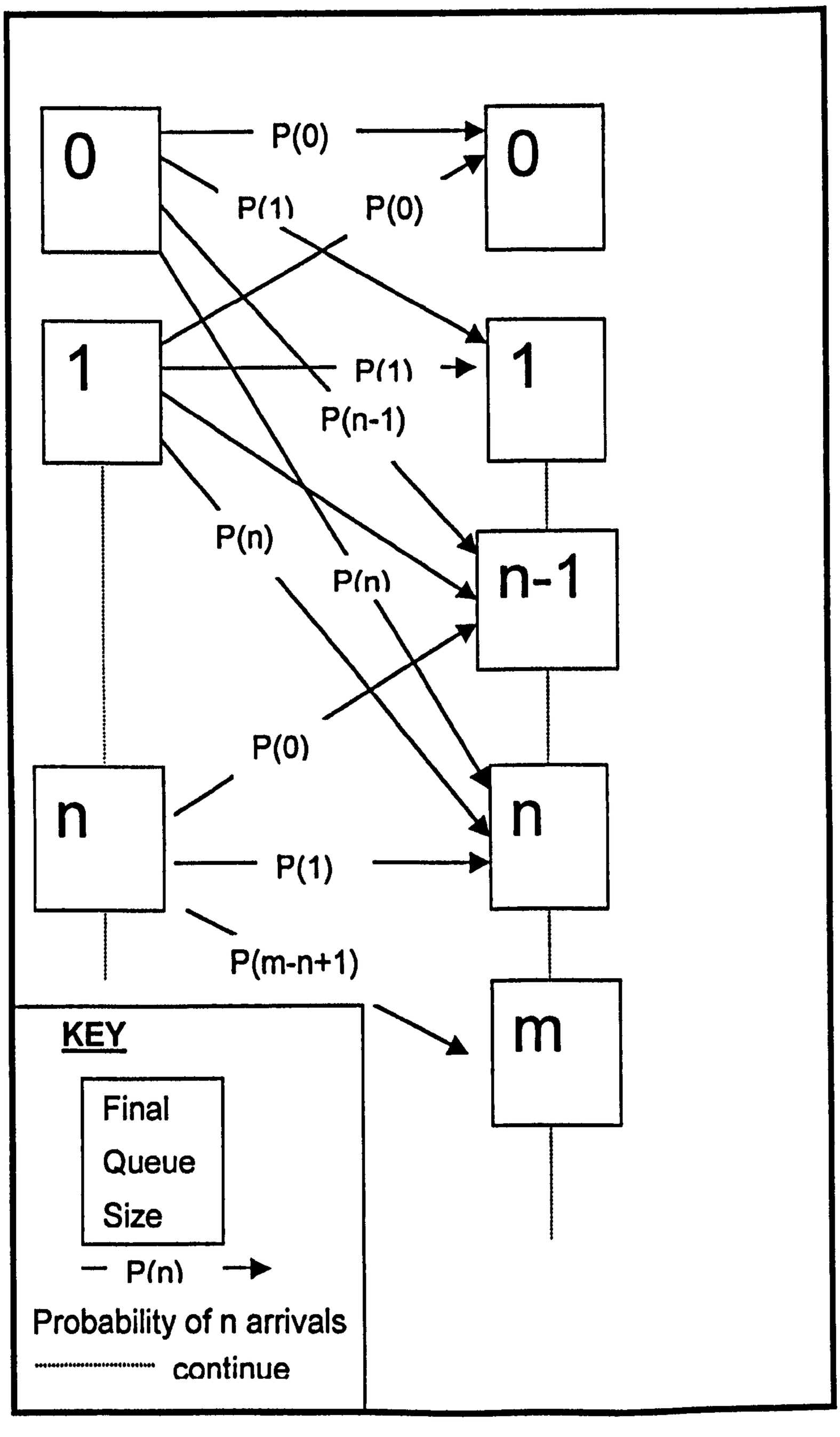
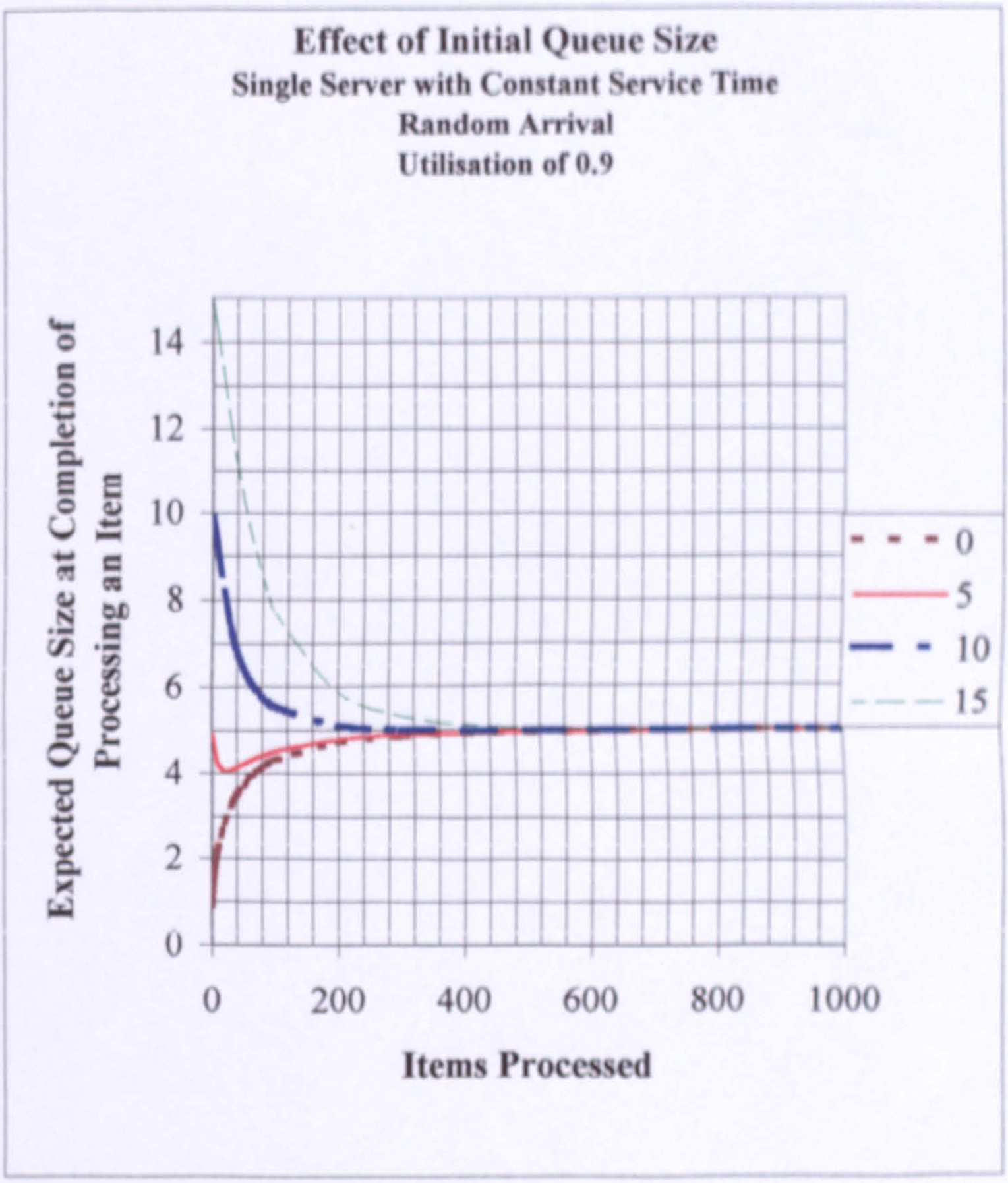


Diagram 6.1: Diagram of Markov Chain Model of Queue

Items Processed	Initial Queue Size 0	5	10	15
1	0.90	4.90	9.90	14.90
2	1.21	4.80	9.80	14.80
3	1.42	4.70	9.70	14.70
4	1.59	4.60	9.60	14.60
5	1.73	4.50	9.50	14.50
6	1.86	4.41	9.40	14.40
7	1.97	4.34	9.30	14.30
8	2.07	4.27	9.20	14.20
9	2.16	4.22	9.10	14.10
10	2.24	4.18	9.00	14.00
20	2.84	4.04	8.06	13.00
30	3.22	4.07	7.33	12.00
40	3.49	4.14	6.81	11.11
50	3.70	4.21	6.43	10.30
60	3.87	4.29	6.15	9.60
70	4.01	4.35	5.93	9.01
80	4.12	4.41	5.76	8.51
90	4.22	4.46	5.63	8.07
100	4.30	4.51	5.53	7.70
200	4.72	4.78	5.09	5.86
300	4.85	4.88	5.00	5.30
400	4.91	4.92	4.97	5.10
500	4.93	4.93	4.96	5.02
600	4.94	4.94	4.96	4.98
700	4.95	4.95	4.95	4.97
800	4.95	4.95	4.95	4.96
900	4.95	4.95	4.95	4.95
1000	4.95	4.95	4.95	4.95

Table 6.2: Expected Queue Size after a Number of Items Processed for Different Initial Queue Sizes



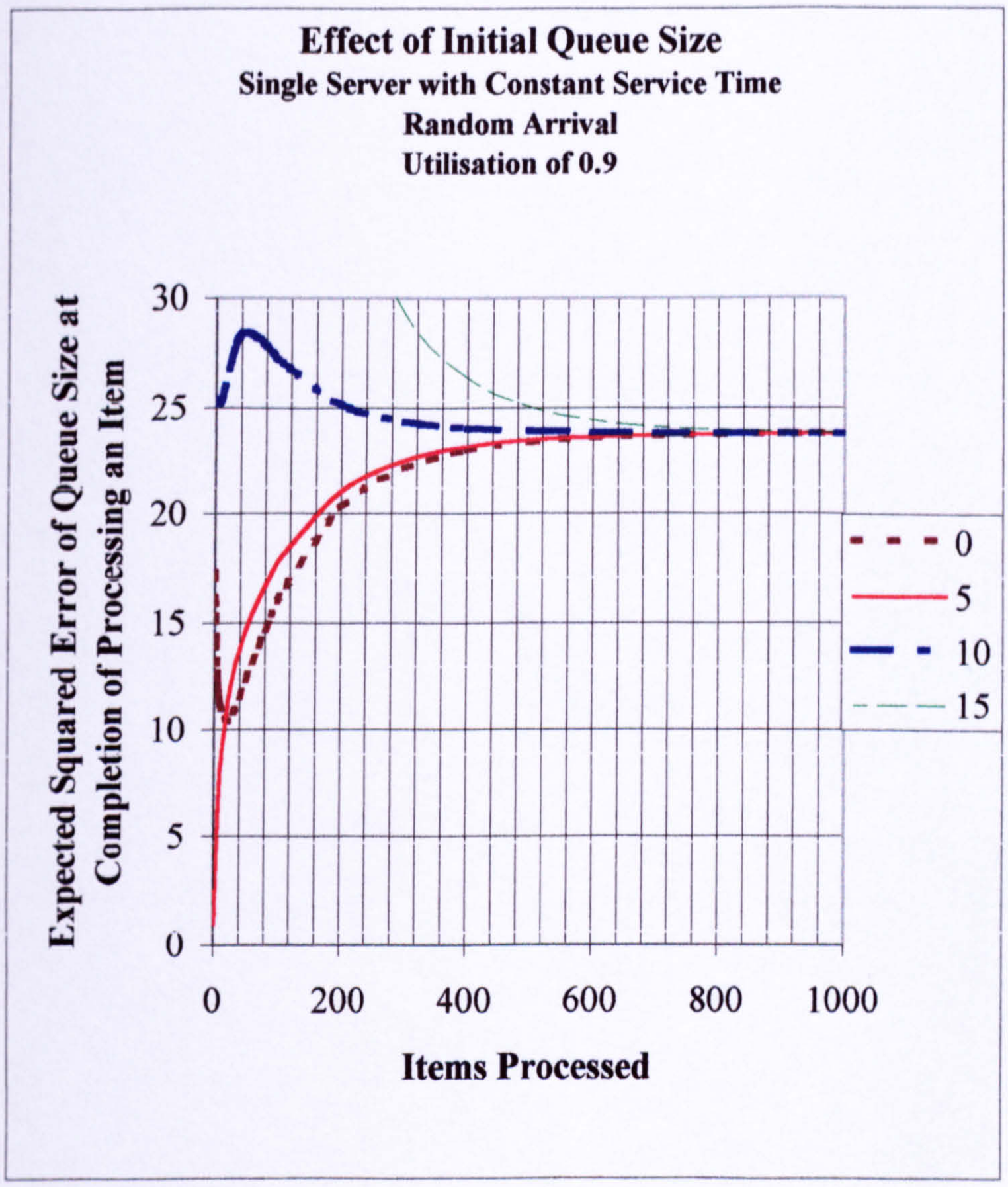
Graph 6.1: The Change in the Expected Queue Size

Kelton and Law (1985) created similar graphs when they also were attempting to determine the best starting position and length of run required to obtain "steady state" conditions. Based on a measure of convergence rate they used the absolute difference between the long term average and the expected waiting time after a certain elapse time from initiating the queue, they concluded that the larger queue size (an initial queue size of 15 for when there was one machine at 90% utilisation) appears to enable the steady state to be obtained quickest. In the case of the queue with constant processing time the convergence appears to be for a range of initial queue sizes of 0 to 10.

Although it is true that at steady state the expected mean queue size has a predictable value, the fact that the measured average mean of the queue has that value after a certain number of parts have been processed is not a dependable indicator that steady state has been reached. It is false logic to state that since, at the steady state, the expected mean is the long term average, then if the measured mean is the same as the long term average then the system must be at steady state and such a state must exist.

Kelton and Law noted that their study had concentrated on the expected delay time and suggest further studies with criteria based on the probability of delay time being greater than a certain value. This reflects the concern that the mean is insufficient an indicator of the whole distribution.

The Markov model enabled other criteria of convergence onto steady state to be examined. The model calculates the distribution of the probability of each possible queue size remaining after a certain number of parts have been processed. The expected square difference of the queue size from the expected mean at steady state (or expected square error) was measured. The results are shown in graph 6.2. (Page 161)



Graph 6.2: The Change in the Squared Error of the Queue

The result confirms the expected value result that after 700 parts being processed, the expected squared error of the queue size for the initial queue sizes of 0, 5 and 10, converged onto the long-term value, whilst the curve for the initial queue size of 15 converged to this long-term value at a higher value of items processed (840).

In this case, the initial queue size that apparently was the fastest to converge is a queue size of zero as it had the smallest variance from the long-term average after 40 parts had been processed.

The study indicates that the initial queue sizes of 0 to 10 do not give significantly different times to converge.

The assumption used in the study just described and in Kelton and Law (1985) is that the steady state exists but is unknown. However the steady state probabilities are known and thus it should be possible to set the queue so as to be at steady state without any warm-up period if such a state exists.

6.1.3 Attempting to set the Initial Queue to a Steady State Value

If the concept that a steady state does exist is true then, as discussed before, the queue size in the model described above should be able to be initialised to a state such that the probabilities of obtaining a certain queue size after the first part is processed are the steady state probabilities,

The situation being examined is an experiment consisting of a single run. The situation where there are a large number of replicates and each replicate could be set to a different initial queue size is not being examined.

The queue sizes examined were all those that have been considered in the literature as candidates for suitable starting values for steady state measurements. These were:

- the system in an empty condition with an initial queue of size 0
- the system in its initial queue size set to the mode (which for a machine utilisation of 0.9 is a queue size of 1)
- the long term average queue size (here a queue size of 4.95) represented by the two nearest integers that are queue sizes of 4 and 5.

Table 6.3 gives the probability of obtaining queue sizes of 0, 1, 2, 3, 4 and 5, when the first, second, third, fourth and fifth part are processed.

Initial Queue	Queue Left after 1 Part Processed					
	0	1	2	3	4	5
0	0.4066	0.3659	0.1647	0.0494	0.0111	0.0020
1	0.4066	0.3659	0.1647	0.0494	0.0111	0.0020
2	0.0000	0.4066	0.3659	0.1647	0.0494	0.0111
3	0.0000	0.0000	0.4066	0.3659	0.1647	0.0494
4	0.0000	0.0000	0.0000	0.4066	0.3659	0.1647
5	0.0000	0.0000	0.0000	0.0000	0.4066	0.3659

Initial Queue	Queue Left after 2 Parts Processed					
	0	1	2	3	4	5
0	0.3141	0.3496	0.2075	0.0879	0.0297	0.0085
1	0.3141	0.3496	0.2075	0.0879	0.0297	0.0085
2	0.1653	0.2975	0.2678	0.1607	0.0723	0.0260
3	0.0000	0.1653	0.2975	0.2678	0.1607	0.0723
4	0.0000	0.0000	0.1653	0.2975	0.2678	0.1607
5	0.0000	0.0000	0.0000	0.1653	0.2975	0.2678

Initial Queue	Queue Left after 3 Parts Processed					
	0	1	2	3	4	5
0	0.2698	0.3272	0.2209	0.1112	0.0464	0.0168
1	0.2698	0.3272	0.2209	0.1112	0.0464	0.0168
2	0.1882	0.2782	0.2395	0.1551	0.0819	0.0364
3	0.0672	0.1815	0.2450	0.2205	0.1488	0.0804
4	0.0000	0.0672	0.1815	0.2450	0.2205	0.1488
5	0.0000	0.0000	0.0672	0.1815	0.2450	0.2205

Initial Queue	Queue Left after 4 Parts Processed					
	0	1	2	3	4	5
0	0.2427	0.3083	0.2244	0.1254	0.0597	0.0252
1	0.2427	0.3083	0.2244	0.1254	0.0597	0.0252
2	0.1896	0.2681	0.2275	0.1525	0.0873	0.0438
3	0.1011	0.1906	0.2202	0.1938	0.1383	0.0827
4	0.0273	0.0984	0.1771	0.2125	0.1912	0.1377
5	0.0000	0.0273	0.0984	0.1771	0.2125	0.1912

Initial Queue	Queue Left after 5 Parts Processed					
	0	1	2	3	4	5
0	0.2240	0.2929	0.2238	0.1343	0.0699	0.0327
1	0.2240	0.2929	0.2238	0.1343	0.0699	0.0327
2	0.1861	0.2600	0.2206	0.1514	0.0912	0.0493
3	0.1186	0.1963	0.2074	0.1778	0.1303	0.0829
4	0.0511	0.1180	0.1719	0.1909	0.1711	0.1282
5	0.0111	0.0500	0.1125	0.1687	0.1898	0.1708

Table 6.3: The Probabilities of Remaining Queues after 1-5 Parts Processed

None of the initial queue sizes led to the steady state probabilities given in table 6.1, even when the time after the initial queue size was extended to five parts produced. It has therefore been demonstrated that it is not possible to initialise the system at a particular queue size and immediately obtain the steady state probabilities after the processing of the first item even when the steady state probabilities are known. It will be able to be noted that 1 and 0 were the only viable candidates since they were the only initial queue sizes that could, after the next part is processed, lead to a queue size of 0 and that for any stable queue there is a finite probability of having a queue of 0.

It is also shown by this experiment that it is not possible to warm the system up and obtain a queue size so that the probabilities of leaving certain queue sizes after the next part processed will be the "steady state" probabilities since there is no such queue size.

It may also be noted that the initial queue sizes examined are normal queue sizes and would be expected to occur 66% of the time with a machine utilisation of 90%. Indeed nearly a quarter of the time the queue size may be expected to be 1 or 0. This gives a strong case to support the argument that there is no reason to ignore measurements before "steady state" is achieved since there is a reasonable probability of obtaining after a warm up period, one of these normal queue sizes that would have been the starting value anyway. Even worse, if measurements are delayed until "steady state" is considered to have been reached, there is a probability of 10% of starting with a queue size of over 10, which may be considered not a "normal" condition.

This result means that for all systems containing queues (the queue examined does exist or an approximation to this queue exists in most industrial systems including the target real-life case) cannot be set to a "steady state" either by initialising to certain queue sizes or by running a lengthy warm up period.

6.1.4 Discussion on the Failure to set the Initial Queue to a Steady State Value

The Markovian property of "steady state" does exists in a mathematical sense, as the steady state probabilities were able to be calculated and it is known that the probabilities of the queue size after a long time will equal these steady state probabilities and thus at this point the initial conditions are unimportant. This means that if a very large number of experiments with different real random numbers were run and the queue size examined after a large number of items had been processed the frequency of obtaining a certain queue size will be found to be independent of the initial queue size.

This "steady state" condition is a probabilistic relationship between an initial state and a state after a long lapse of time. It would be useful to have a term to describe this time necessary to elapse to get the independence from the starting condition or what might be termed to "forget" the starting condition. For convenience the term "forgetting time" will be used.

There is no justification for stating that the measurements made only after a warm up period equal to this "forgetting time" are more valid than those made following a warm up period that was sufficient to simply fill the system.

6.2 DISCUSSIONS ON THE INITIAL CONDITIONS AND WARM UP PERIOD

Thus the need for an extension of the warm up period beyond that which is necessary to give system fill has been discounted. For certain production systems it is usual for each machine to be processing a part. To give general rules with allowances for these special cases is difficult and would defeat the fundamental idea of providing simple rules. The selected method was to start

with an empty system and run for sufficient time to ensure a system fill, since it is unlikely that in a production system this would consume much computer time.

The system may be deemed to have been filled when the first part is fully processed. In most practical systems where production volumes are significant and computer processing times required to simulate the production of one part after system fill has been achieved is small, the warm up may be extended beyond this minimum to some convenient time (e.g. the nearest hour).

7 NUMBER AND LENGTH OF RUNS

7.1 NUMBER OF RUNS

7.1.1 Simulation Compared to Traditional Trials

As discussed in the literature review, simulation studies do not have the same need as experiments in real life for replications (repeated trials). In studies such as agriculture or drug investigations, the use of replicates within well-planned experiments enables the effects of the uncontrollable external randomness to be isolated. Real life studies are also not easy to extend. Crops grow in a year and sick people die. Machines in simulation studies, unlike real life, only breakdown if the analyst has programmed them to. Computer models can run for as long as is wanted and the randomness is in control of the analyst.

Thus for non-terminating discrete-event computer simulation there is a choice for the user between having one long run and having many short runs. The most important factor in deciding between having a single long run or a number of shorter runs is most likely to be the accuracy that is obtained using a constant resource of analyst time and computer resources. The choice based on the accuracy obtained is examined in the next section.

7.1.2 Description of the Experiment

The question of whether there is any advantage in terms of accuracy by only performing one run (rather than a number of runs) was investigated by comparing the accuracy of the result obtained from making a number of simulation runs (replicates) with that obtained from a longer single run of the model of the queue previously described in section 6.1 page 152. The total number of items or parts processed in the simulations is the same. The smaller runs consisted of two, four and eight replicates. The number of items process was 1000, 500 and 250 respectively. The long run consisted of 2000 items. Thus in every case the total number of items processed in each measurement

was 2000. The total experiment was repeated for machine utilisation of 50%, 60%, 70%, 80%, and 90%.

Previously it has been discussed that a warm up period for such a queue is not required even for a system fill. However in this case a lengthy warm up period was made so as to satisfy any view that such a warm up period is required and thus isolate that discussion from the subject of the number of runs required. A single warm up run was made for each machine utilisation, and the final queue size was used as the starting point for each replicate or run with the respective machine utilisation.

Since the warm up resource was identical for the short runs and the long run, and the same number of items was processed (2000) in all the runs, the total resource used for the long run and the short runs were the same.

Although it has been shown in section 5 that the use of Descriptive Sampling or MLCGs with specially selected seeds would have given more accurate results, this approach was not followed. This was to prevent any distortion of the results on how many runs should be made from the effects of such a variance reduction technique.

For this study the MLCG suggested by Fishman and Moore (see table 5.3) and that was the most successful in creating quality samples, was used.

In the base experiments a single run at each machine utilisation was performed in which, as was stated above, 2000 parts were processed. This experiment was repeated 10,000 times to give a measure of the variability of the measurement from such an experiment. The experiments for the multiple replicates consisted, as described above, of 2, 4, and 8 replicates, the average of the replicates being taken as the measurement from each experiment. The seeds values were set to the next sequential value at the start of each replicate or run.

7.1.3 The Results from the Warm-up

The warm up run was started with a seed of 1 and run for 2000 items, which was considered to be longer than the "forgetting time".

The queue size remaining after different number of items processed is given in table 7.1.

		Remaining Queue Size				
Items Processed	Machine Utilisation	50%	60%	70%	80%	90%
100		1	1	1	1	1
200		0	0	0	0	0
300		0	1	1	1	1
400		1	1	1	1	1
500		0	0	1	2	2
600		0	0	1	5	10
700		0	0	0	0	1
800		2	2	2	2	5
900		0	0	0	1	2
1000		1	1	1	2	2
1100		1	1	2	2	9
1200		1	1	1	1	6
1300		2	2	2	3	14
1400		2	2	2	3	3
1500		0	0	0	0	1
1600		1	1	2	4	6
1700		0	0	0	2	2
1800		1	1	1	1	3
1900		0	1	1	2	4
2000		0	0	0	0	0

Table 7.1 The Remaining Queue Size for F&M MLCG with a Seed of 1

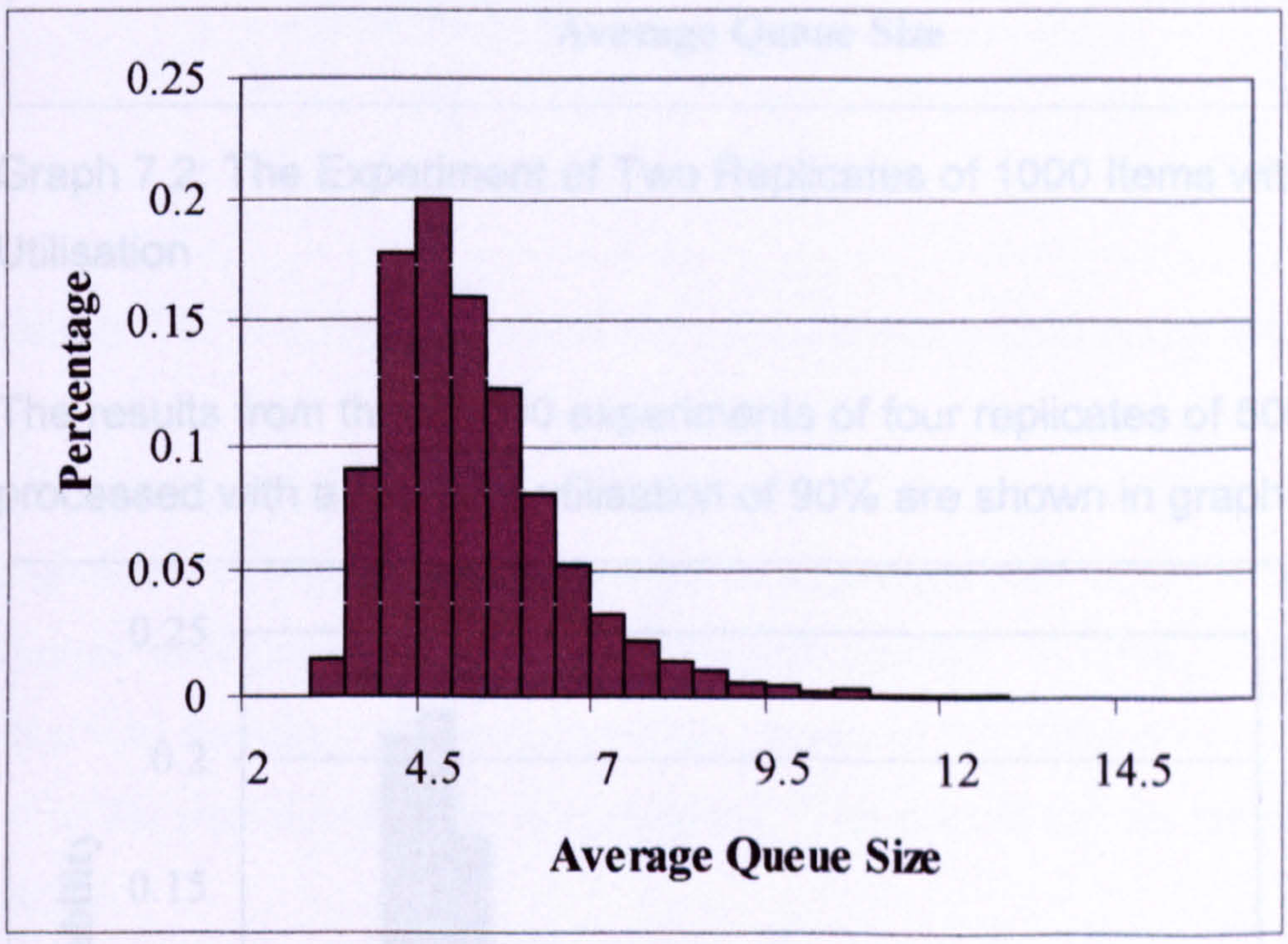
It can be seen that the remaining queue size after a warm up of 2000 items was 0 for all the machine utilisations examined. If the warm up had stopped at 1300

items processed the remaining queues, therefore the initial queue sizes for the replicates, and computer runs would have been different. For 90% utilisation the initial queue size would have been 14. This would have been a less "normal" queue size than 0.

7.1.4 The Results

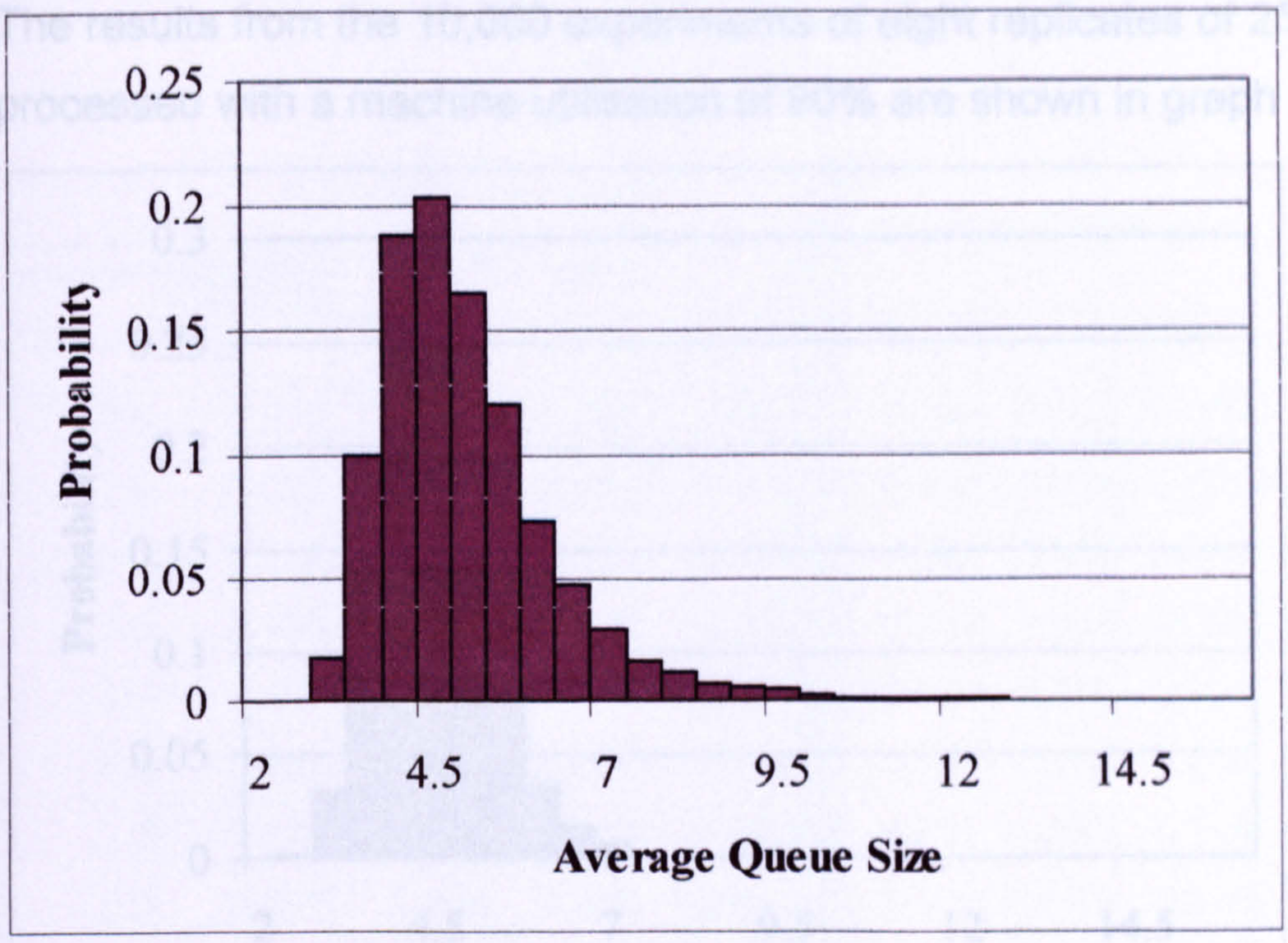
7.1.4.1 90% Machine Utilisation

The results from the 10,000 base experiments of 2000 items processed with a machine utilisation of 90% is shown in graph 7.1.



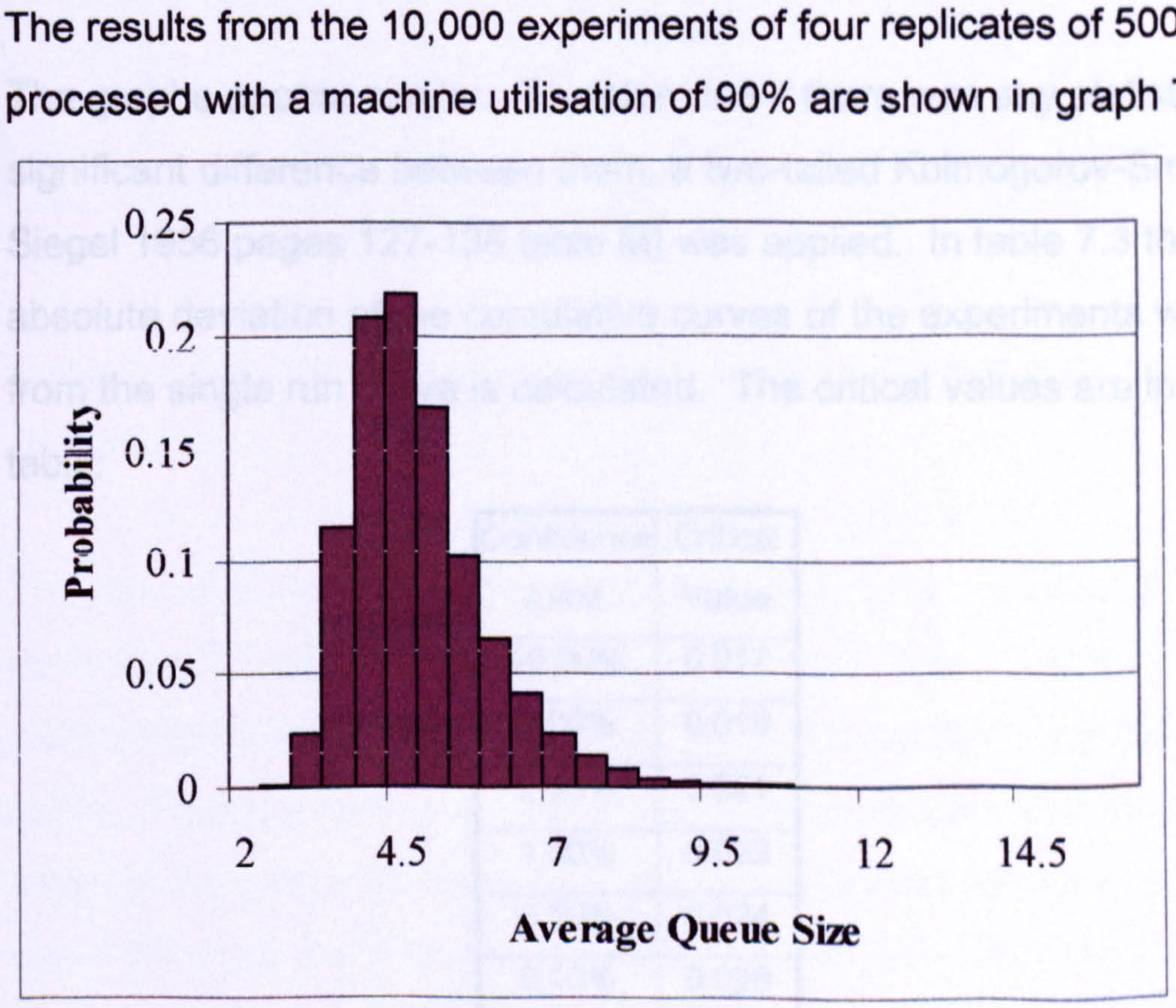
Graph 7.1: The Base Experiment of a Single Run of 2000 Items with 90% Machine Utilisation

The results from the 10,000 experiments of two replicates of 1000 items processed with a machine utilisation of 90% are shown in graph 7.2.



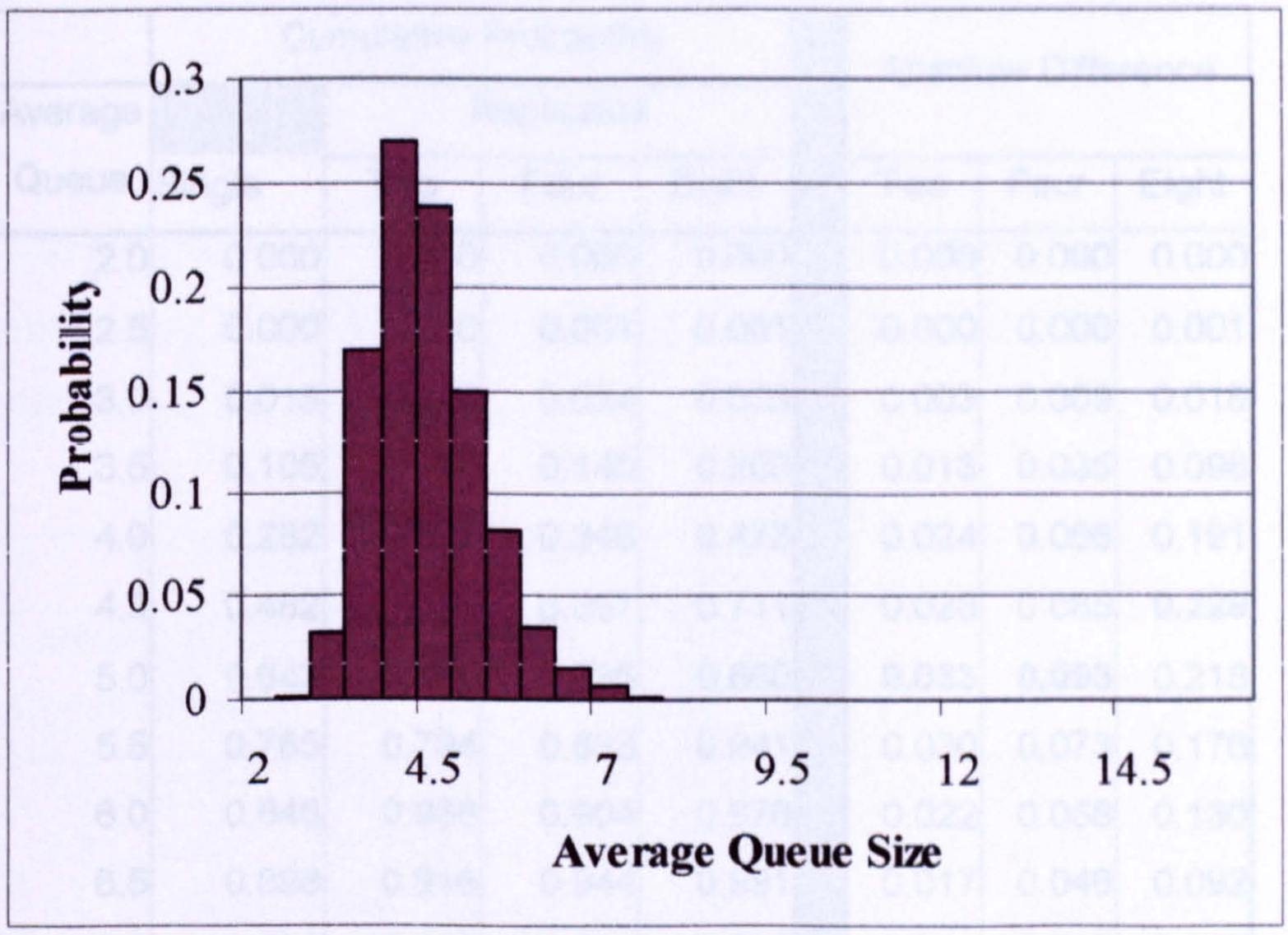
Graph 7.2: The Experiment of Two Replicates of 1000 Items with 90% Utilisation

The results from the 10,000 experiments of four replicates of 500 items processed with a machine utilisation of 90% are shown in graph 7.3.



Graph 7.3: The Experiment of Four Replicates of 500 Items with 90% Utilisation

The results from the 10,000 experiments of eight replicates of 250 items processed with a machine utilisation of 90% are shown in graph 7.4.



Graph 7.4: The Experiment of Eight Replicates of 250 Items with 90% Utilisation

The graphs appear similar. To determine if there was any statistically significant difference between them, a two-tailed Kolmogorov-Smirnov test (See Siegel 1956 pages 127-136 table M) was applied. In table 7.3 the maximum absolute deviation of the cumulative curves of the experiments with replicates from the single run curve is calculated. The critical values are in the following table:

Confidence Limit	Critical Value
10.00%	0.017
5.00%	0.019
2.50%	0.021
1.00%	0.023
0.50%	0.024
0.10%	0.028

Table 7.2 Critical Values for the Kolmogorov-Smirnov Test for 10,000 Values

Average Queue	Cumulative Probability				Absolute Difference		
		Replicates					
	Single	Two	Four	Eight	Two	Four	Eight
2.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2.5	0.000	0.000	0.001	0.001	0.000	0.000	0.001
3.0	0.015	0.018	0.024	0.033	0.003	0.009	0.018
3.5	0.105	0.118	0.140	0.203	0.013	0.035	0.098
4.0	0.282	0.306	0.348	0.473	0.024	0.066	0.191
4.5	0.482	0.510	0.567	0.711	0.028	0.085	**0.229**
5.0	0.642	0.675	0.735	0.860	**0.033**	**0.093**	0.218
5.5	0.765	0.794	0.838	0.941	0.030	0.073	0.176
6.0	0.846	0.868	0.904	0.976	0.022	0.058	0.130
6.5	0.898	0.916	0.944	0.991	0.017	0.046	0.092
7.0	0.931	0.945	0.968	0.998	0.014	0.037	0.067
7.5	0.953	0.961	0.982	0.999	0.008	0.029	0.046
8.0	0.967	0.973	0.991	1.000	0.006	0.024	0.033
8.5	0.978	0.980	0.994	1.000	0.002	0.016	0.022
9.0	0.984	0.986	0.997	1.000	0.002	0.012	0.016

Table 7.3 Calculation of the Maximum Absolute Deviation for 90% Machine Utilisation

The maximum absolute differences are 0.033, 0.093, and 0.229 for the 2 replicates, 4 replicates, and 8 replicates respectively. These values when referred to table 7.2 are too great for the null hypothesis that the curves are the same. Thus the curves are different. The theoretical value for the long-term mean of the queue size is 4.95. The value obtained from the 10,000 single runs was 4.85, from the 10000 two replicates it was 4.74, from the 10,000 four replicates it was 4.54 and from the 10,000 eight replicates it was 4.16.

It may be seen that the larger run has a closer estimate of the mean value and thus the shorter runs are more inaccurate for the 90% machine utilisation.

7.1.4.2 80% Machine Utilisation

The results from these experiments produced similar curves and it was necessary to statistically test them. The maximum absolute difference was calculated as for the 90%.

	Cumulative Probability					Absolute Difference		
Average Queue	Single	Replicates Two	Four	Eight		Two	Four	Eight
2.1	0.198	0.202	0.216	0.251		0.004	0.018	0.053
2.2	0.317	0.323	0.348	0.395		0.006	0.032	0.078
2.3	0.446	0.456	0.485	0.542		0.010	**0.039**	0.096
2.4	0.570	0.578	0.604	0.671		0.009	0.034	**0.101**
2.5	0.679	0.694	0.712	0.771		0.015	0.033	0.093
2.6	0.765	0.785	0.797	0.848		**0.019**	0.031	0.082
2.7	0.835	0.848	0.859	0.903		0.013	0.024	0.068
2.8	0.887	0.895	0.901	0.937		0.008	0.014	0.050
2.9	0.923	0.928	0.934	0.960		0.006	0.011	0.037
3.0	0.947	0.951	0.957	0.975		0.004	0.010	0.028
3.1	0.965	0.966	0.972	0.984		0.000	0.007	0.019
3.2	0.975	0.976	0.980	0.990		0.001	0.004	0.014
3.3	0.984	0.982	0.987	0.993		0.001	0.003	0.010
3.4	0.989	0.988	0.991	0.995		0.001	0.002	0.007

Table 7.4 Calculation of the Maximum Absolute Deviation for 80% Machine Utilisation

In this case the null hypothesis that 2 replicates curve and the Single curve are the same cannot be rejected at the 5% confidence level, since the maximum absolute difference is 0.019 which is the 5% confidence level (table 7.2). The other curves have too large absolute differences for the null hypothesis not to

be rejected. Thus the 4 and 8 replicates are statistically different curves to the curve for the single run. The theoretical long-term queue size is 2.40 (see expression on page 154). The measured average queue size in the experiments were for the 10,000 single run, 2.39, for the 2 replicate run it was 2.38, for the 4 replicate run it was 2.36, and for the 8 replicate run it was 2.31.

Thus it may still be said that the single run was more accurate.

7.1.4.3 70% Machine Utilisation

Again the results from these experiments produced similar curves and it was necessary to statistically test them. The maximum absolute difference was calculated as before.

	Cumulative Probability					Absolute Difference		
Average Queue	Single	Replicates Two	Four	Eight		Two	Four	Eight
1.45	0.353	0.356	0.377	0.401		0.003	0.024	0.048
1.48	0.423	0.430	0.448	0.480		0.006	0.025	0.057
1.50	0.495	0.506	0.520	0.553		0.011	**0.025**	0.058
1.53	0.564	0.573	0.589	0.630		0.009	0.025	**0.066**
1.55	0.634	0.640	0.650	0.691		0.007	0.017	0.057
1.58	0.698	0.701	0.713	0.746		0.004	0.015	0.049
1.60	0.753	0.754	0.764	0.797		0.001	0.011	0.044
1.63	0.797	0.802	0.807	0.839		0.005	0.011	0.043
1.65	0.834	0.846	0.845	0.876		**0.012**	0.011	0.042
1.68	0.866	0.875	0.876	0.904		0.009	0.010	0.037
1.70	0.894	0.898	0.903	0.928		0.004	0.008	0.034

Table 7.5 Calculation of the Maximum Absolute Deviation for 70% Machine Utilisation

Referencing table 7.5 shows that the curves for both the 2 replicates and the 4 replicates may be considered to be statistically the same as the single run. The curve for the 8 replicates however is statistically not the same. The theoretical long-term average queue size for 70% machine utilisation is 1.52. The measure queue size for the single run, the 2 replicates and the 4 replicates was 1.51, while the value for the 8 replicates was 1.49. Thus for 70% machine utilisation the accuracy for the single, 2 replicates and 4 replicates was statistically identical whilst the accuracy for the 8 replicates was lower.

7.1.4.4 60% Machine Utilisation

As before the results from these experiments produced similar curves and it was necessary to statistically test them. The maximum absolute difference was calculated as before.

Average Queue	Cumulative Probability					Absolute Difference		
		Replicates						
	Single	Two	Four	Eight		Two	Four	Eight
0.98	0.195	0.194	0.201	0.211		0.001	0.006	0.016
1.00	0.251	0.252	0.257	0.272		0.001	0.006	0.021
1.01	0.315	0.313	0.323	0.338		0.002	0.008	0.023
1.02	0.383	0.380	0.392	0.406		0.004	0.009	0.023
1.03	0.451	0.448	0.461	0.482		0.003	0.010	0.031
1.05	0.519	0.518	0.530	0.549		0.001	0.011	0.031
1.06	0.581	0.588	0.596	0.619		**0.008**	0.015	**0.038**
1.07	0.645	0.647	0.656	0.679		0.002	0.011	0.034
1.09	0.700	0.704	0.714	0.735		0.004	0.014	0.035
1.10	0.752	0.755	0.766	0.787		0.003	0.014	0.035
1.11	0.797	0.803	0.814	0.829		0.005	**0.016**	0.031
1.13	0.842	0.844	0.850	0.866		0.002	0.008	0.024

Table 7.6 Calculation of the Maximum Absolute Deviation for 60% Machine Utilisation

In this case all the curves are statistically the same. The theoretical long-term average queue size for 60% machine utilisation is 1.05. The measured average queue size for the single, 2 replicates, and 4 replicates was 1.05 and for the 8 replicates it was 1.04. For 60% machine utilisation, the result was as with 70%, the accuracy for the single, 2 replicates and 4 replicates was statistically identical whilst the accuracy for the 8 replicates was lower although in this case only slightly.

7.1.4.5 50% Machine Utilisation

In this case the curves are very similar and again it was necessary to statistically test them. Again the maximum absolute difference was calculated:

Average Queue	Cumulative Probability					Absolute Difference		
		Replicates						
	Single	Two	Four	Eight		Two	Four	Eight
0.69	0.084	0.087	0.091	0.083		0.003	0.006	0.001
0.70	0.118	0.125	0.126	0.121		0.007	0.009	0.003
0.70	0.158	0.168	0.171	0.165		0.009	**0.013**	0.006
0.71	0.214	0.218	0.218	0.219		0.004	0.004	0.005
0.72	0.274	0.279	0.275	0.278		0.005	0.000	0.004
0.73	0.332	0.343	0.343	0.345		0.010	0.011	0.013
0.74	0.401	0.407	0.409	0.417		0.006	0.009	**0.016**
0.74	0.478	0.472	0.482	0.488		0.006	0.005	0.011
0.75	0.547	0.541	0.553	0.558		0.005	0.006	0.011
0.76	0.615	0.608	0.617	0.629		0.007	0.002	0.013
0.77	0.682	0.670	0.678	0.691		**0.012**	0.004	0.009
0.78	0.737	0.727	0.737	0.744		0.010	0.000	0.007
0.78	0.781	0.778	0.786	0.796		0.004	0.004	0.014
0.79	0.824	0.821	0.826	0.839		0.004	0.002	0.015

Table 7.7 Calculation of the Maximum Absolute Deviation for 50% Machine Utilisation

As in the previous case all the curves are statistically the same. The theoretical long-term average queue size is 0.75. The actual measured value was 0.75 for every case. However if the results are examined at three decimal places the following table is obtained:

Type of Experiment	Average Queue Size
Theoretical	*0.7500*
Single run	0.7498
2 Replicates	0.7496
4 Replicates	0.7490
8 Replicates	0.7477

Table 7.8 Actual Average Queue Size Measured with Machine Utilisation of 50%

The accuracy may be seen to be higher for the single run. Similar results were seen for all the machine utilisation examined.

7.1.5 Conclusion

The most accurate forecasts were obtained from a single run. The multiple runs were able to reached statistically identical performance when the variability in the process was less. (The variability reduces as the machine utilisation reduces. See the queue sizes variability in the warm up, table 7.1)

7.2 LENGTH OF RUN

In section 6 when considering the setting of the initial conditions the change in the expected queue size as parts were processed was measured. This was to establish the expected queue values at the various times of production. However in a simulation study the measurement of average queue size is

usually made by computing the average queue size based on all the measured queue sizes (as was used in section 7.1). That is:

$$\text{Average Queue Size} = \frac{\sum_{i=1}^{i=n} q_i}{n}$$

thus

$$\text{Expected Cumulative Queue Size} = \frac{\sum_{i=1}^{i=n} Eq_i}{n}$$

Where:

q_i is the measured queue size after producing the ith part or item

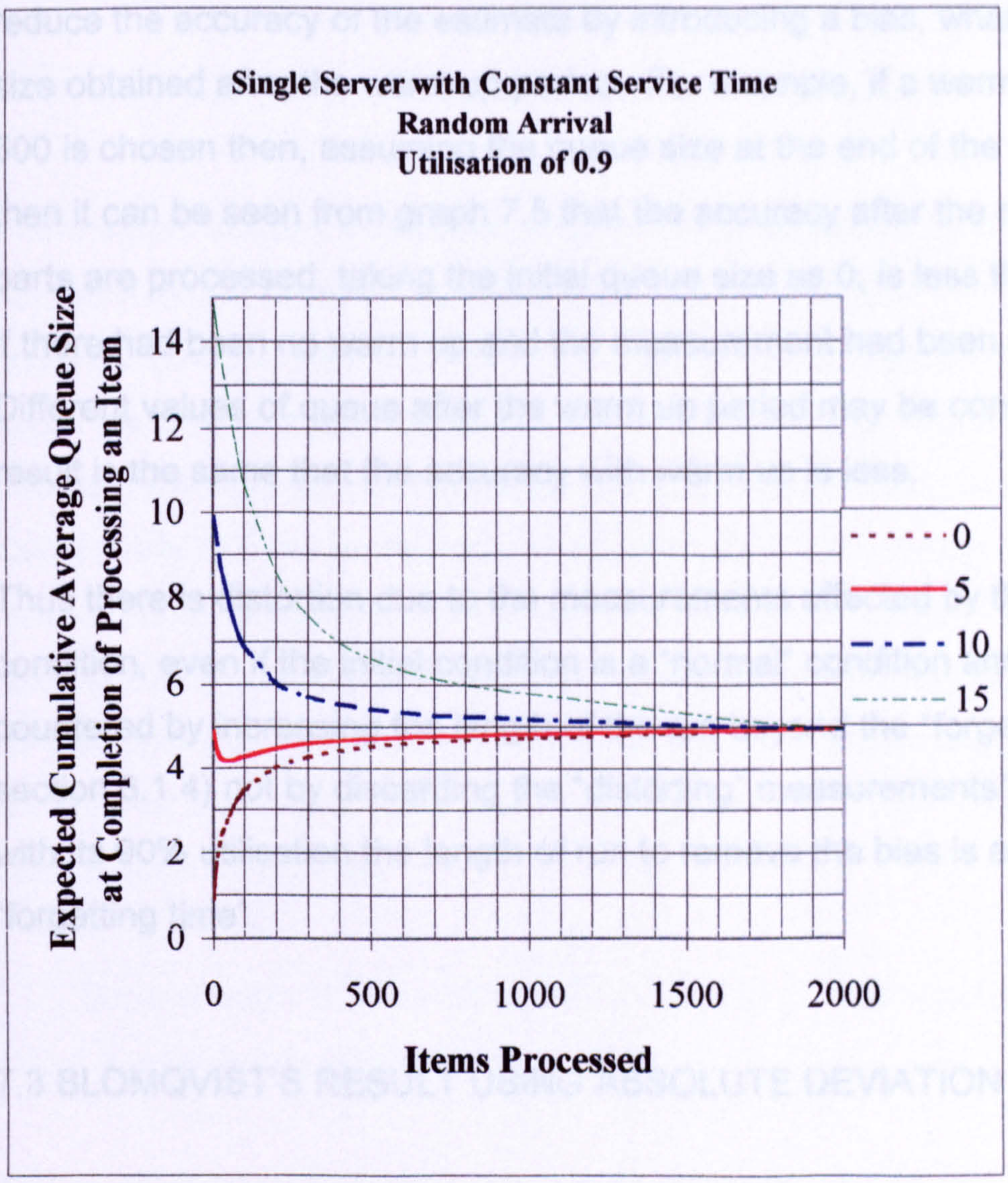
n is the number of parts

Eq_i is the expected queue size after producing the ith part or item

The term Expected Cumulative Queue Size is used for clarity.

In graph 7.5 on the following page the value of the expected cumulative average queue size is shown as it changes with production (simulation run time). The study was performed using the Markov model of section 6.1.2.

Previous studies in section 6 had indicated that 1000 parts processed is sufficient for the "forgetting time" (see section 6.1.4), graph 7.5 indicates that 1000 parts processed is not sufficient for the bias to be removed from the estimate of the long term average created by the average of the queue sizes obtained. In order not to have any bias, 2000 items are required to be processed with any initial queue size in the range 0 to 15.



Graph 7.5: The Change in the Estimate of Average Queue Size

This result may at first be considered to undermine the statement that, for a queue any warm up time is detrimental to the accuracy of the final estimate. It may be considered that the extended run required to remove the bias is only necessary to overcome the distortion caused by including the measurements that were affected by the initial value of the queue, which in the case of a queue of zero was an empty system. But graph 7.5 demonstrates that Blomqvist's result that no warm up is required if the simulation is long enough to get an accurate result is correct. If it is considered that computer power for 2000 parts is to be used, then any warm up period would mean less than 2000 parts could be used to provide the estimate. Any reduction below 2000 can be seen to

reduce the accuracy of the estimate by introducing a bias, whatever the queue size obtained after the warm up period. For example, if a warm up period of 500 is chosen then, assuming the queue size at the end of the warm up is 0, then it can be seen from graph 7.5 that the accuracy after the remaining 1500 parts are processed, taking the initial queue size as 0, is less than the accuracy if there had been no warm up and the measurement had been for 2000 parts. Different values of queue after the warm up period may be considered but the result is the same that the accuracy with warm up is less.

Thus there is distortion due to the measurements affected by the initial condition, even if the initial condition is a "normal" condition and this can only be countered by increasing the length of the run beyond the "forgetting time" (see section 6.1.4) not by discarding the "distorting" measurements". For our queue with its 90% utilisation the length of run to remove the bias is equal to twice the "forgetting time".

7.3 BLOMQVIST'S RESULT USING ABSOLUTE DEVIATION

Since graph 7.5 is not indicating square error but a visual estimate of the absolute error, Blomqvist's result can therefore be seen to hold for absolute deviation as well as a squared error measure of accuracy, at least for this type of queue and with a 90% machine utilisation. Thus the rejection of the result based on the choice of a squared deviation measure of error is unfounded as it holds for absolute deviation as well.

The result indicates that the run length for systems containing queue needs to be well in excess of the "forgetting time" (possibly double) and that any time spent on warm up past the initial obtaining of "normal" conditions is detrimental to the accuracy obtained. A means of obtaining some estimate of the actual run length that should be used in the first run of the model is required. This will be discussed next.

7.4 A METHOD OF DETERMINING LENGTH OF RUN

In the earlier section it has been shown that there should be one run of sufficient length to allow any initial distortion to be reduce to insignificance however Law and Kelton (2000) claim that, before any experimentation has taken place, it is impossible to determine the run length required to obtain the necessary accuracy. As discussed in the literature review, developments are being made to provide methods of establishing if a run needs to be extended to obtain greater accuracy. These methods are not discussed here. Law and Kelton (2000) do not give a method of determining the initial computer run but do give some guidance on run time in a section on determining what they considered is the necessary warm up period in which they state that the computer run should be:

".....large enough to allow infrequent events (e.g., machine breakdowns) to occur a reasonable number of times."

A method of establishing a run length that will provide at least a firm basis to determine such action as extending a run or to determine if the run already made is sufficient is required. It is proposed that the length of run should be chosen with the same criteria as was used in selecting the method of producing randomness and that reflects the requirement stated by Law and Kelton. Applying the concept that the requirement is that the run should be highly representative of the environment of the system being studied, leads directly to the requirement, noted by Law and Kelton, that the length of run should be such that there are a representative number of important events occurring during the simulation run. Only with such a representative run can any reasonable statements be made of the effectiveness of the design of the facility and indeed, as stated above, if a longer run is required to obtain the necessary accuracy.

In many manufacturing facilities there is an important stochastic event, which although it may not be the major variable controlling the performance of the system, will dominate the run length of any simulation attempting to accurately model the facility. This is normally because it a relative rare event and for the run to be considered realistic enough occurrences have to occur. In our real-life target model the number of machines, their layout and buffering may have the major effect on throughput, but there are disruptions in the process due to parts jamming or machines breaking down. In order for the measurement of throughput to be representative there must be an adequate number of machine breakdowns within the computer run. Indeed in order that the effect of these disruptions is correctly reflected, there must be sufficient number of the longer duration machine breakdowns within the computer run.

In this study, a method will be examined for determining the minimum run time of a simulation run based on ensuring that a representative number of these events occur in the simulation. There may exist other concepts, philosophies and methods (as yet not published) of determining the initial run length but this need for a representative number of events is a basic requirement and must be met.

In practical cases it may be sensible to exclude some very lengthy breakdowns. In the target example, the management excluded breakdowns of over 8 hours. They did not expect these to be included in the measurement of expected throughput. Such breakdowns would be seen as "reasonable excuses" for failure to produce the required output and they would use "overtime" to recover such production losses.

It has been assumed that, within the simulation model the stochastic factor controlling the length of the simulation is machine breakdowns and it is considered, as before, that both the duration of the disruption and the time between occurrences follow negative exponential distributions.

For the analysis a machine is selected that is significant in the production and has significant but infrequent breakdowns. This is termed, for this discussion, the "target machine".

The required run length may be seen to be determined by the number of breakdowns of the "target machine" required to give the requisite number of larger duration breakdowns. This is equivalent to determining the number of samples required for determining the durations of the breakdowns of the target machine. This is discussed in the next section.

7.5 SELECTION OF A SUITABLE NUMBER OF SAMPLES

In this section a suitable method will be discussed for determining a run length in terms of the number of samples required for an important, although infrequent, event, so that representative behaviour is ensured. It is assumed that the initial warm up period (or as previously discussed system fill) has been completed and the run length is measured from that position.

As a measure of achieving reasonable representation the metric chosen was the number of sample values that will lie in the critical areas of the statistical distribution during the simulation run. This critical area in the reference real-life model is the duration of a disruption so larger that it would be considered a major disruption to production.

If the underlying distribution of the disruption is negative exponential (a not-uncommon characteristic) the probability P of being within a certain range, say t_1 to t_2, when the mean duration time is λ, is given by

$$P = e^{-\lambda t_2} - e^{-\lambda t_1}$$

As stated the critical area in the case of disruptions within the target real-life model is the tail of the long durations. This upper tail may be defined by a

single parameter time T where the tail is defined by the area that contains all duration times greater than T and is defined by the value of T. (In the case of our real life model the duration of a major breakdown would have been considered to be 5 hours when the average breakdown is an hour.)

If true random numbers are generating the times of the disruptions then the probability P of a single sample being in the tail is:

$$P = e^{-\lambda T}$$

Thus if the mean of the negative exponential distribution is 1 time unit, then the probability of a sample being in the tail is given in table 7.9.

T	Probability
3	4.9787%
4	1.8316%
5	0.6738%
6	0.2479%
7	0.0912%
8	0.0335%
9	0.0123%
10	0.0045%

Table 7.9: Probability of a Sample Value falling into the Tail

If the run size is to be such that there will be M disruptions the expected number of disruptions E(N), with durations in the tail N, is given by

$$E(N) = MP$$

However the choice of M may be made to give a certain probability of having at least N in the tail.

The problem is now in terms of defining the tail and the number of occurrences required for the users, e.g. Production Engineers, to be able to accept the result. They are likely to feel more comfortable in deciding the number of major disruptions to define a representative run length rather than just to attempting to estimate a suitable time. Any final confirmation run (the computer run to confirm the final design) would be much longer and have available, statistical measurements from earlier runs.

If the samples are considered independent of each other and the probability of being in the tail is P then the count of occurrences in the tail M from a total sample of N will follow a Binomial distribution.

That is:

Probability of M disruption with durations in the tail from a total of N disruptions where P is the probability of an individual disruption sample falling in the tail

$$= \binom{N}{M} P^M (1-P)^{N-M} \text{ where } 0 \le M \le N$$

If N is large and P is small then thus may be approximated to by

$$\approx \frac{e^{-NP}(NP)^M}{M!}$$

Therefore the probability of at least N in an interval from a sample M where P is the probability of an individual sample falling within the required interval

$$= \sum_{i=M}^{i=N} \binom{N}{i} P^i (1-P)^{N-i} \text{ where } 0 \le i \le M$$

$$\approx \sum_{i=M}^{i=N} \frac{e^{-NP}(NP)^i}{i!}$$

Table 7.10 (page 188) gives the values for a tail of 5. That is if the mean of the duration of the breakdowns is 1 hour then the tail contains all observations of over 5 hours.

Table 7.10 shows that if we wished to be 99% certain of having 5 observations of over 5 hours duration when the average duration is 1 hour then we would need 1800 disruptions in our run. If a confidence of 90% would suffice, then 1200 disruptions would be needed.

These results only apply to true random numbers. Since, as previously discussed, in a simulation the random numbers are not usually real random number but "pseudo-random" numbers or a constructed sample, the actual number of samples that will appear in the tail may be exactly established before the simulation run.

If an RNG such as a MLCG or MT is being used, then it is possible to confirm that the number of samples is acceptable once the seed is selected by making a run of the actual random number generator. This may be performed before the full run of the model. If a selected seed is used it may be expected that the number in the tail would be closer to the expected value. In the case of the Fishman and Moore generator, the seed 52260 (rank 1 in the range 1-60,000) had the fifth deviate in the tail at the 642nd random number, for the seed 37846 (rank 10) the 635th deviate was the fifth in the tail and the seed 1562 (the rank 100th) had the 1010 deviate in the tail. In all these three cases the sample size required would need to be bigger than 542, 635 and 1010 respectively but would be less that the 1200 to 1800 required for real random numbers.

Samples	500	600	700	800	900
Expected	3.37	4.04	4.72	5.39	6.06
90.0%	1	2	2	3	3
95.0%	1	1	1	2	2
99.0%	0	0	1	1	1
99.5%	0	0	0	1	1

Samples	1000	1100	1200	1300	1400
Expected	6.74	7.41	8.09	8.76	9.43
90.0%	4	4	5	5	6
95.0%	3	3	4	4	5
99.0%	2	2	2	3	3
99.5%	1	1	2	2	3

Samples	1500	1600	1700	1800	1900
Expected	10.11	10.78	11.45	12.13	12.80
90.0%	6	7	7	8	8
95.0%	5	6	6	7	7
99.0%	4	4	4	5	5
99.5%	3	3	4	4	5

Samples	2000	2100	2200	2300	2400
Expected	13.48	14.15	14.82	15.50	16.17
90.0%	9	9	10	11	11
95.0%	8	8	9	9	10
99.0%	6	6	7	7	8
99.5%	5	5	6	6	7

Table 7.10: Number of Samples Values in a Tail of 5

The computer program TAILMLCG (listing in Appendix 1) gives the number of values falling in a range of definitions of the tail for a certain sample size.

If DS is used, the number of samples in the tail can be determined before the seed to be used in the shuffle is determined. The result of how many samples are in the tail is independent of the seed selection and is only dependent on the total number of samples, since the sample values are construct by a simple rule. The following expression gives the sample size M required to give N observations from a negative exponential distribution with a mean a, in a tail defined by t (in this case not standardised to a mean of 1):

$$M \geq e^{t/a}(N - 0.5) < M + 1$$

Thus five observations in a tail of 5 with a mean of 1 require a sample of 668.

The smaller run lengths required by DS and by MLGC and MT with selected seeds reflect their higher quality of sample as compared with MLCG or MT with a random choice of seed or seeds and a true random sample. DS with its constant required sample size is again superior in ease of use.

7.6 CALCULATION OF THE SIMULATION TIME AND SAMPLE SIZES OF OTHER EVENTS

The duration of the simulated time of the simulation can then be calculated from the mean of the time between the breakdowns and the mean of the duration of the breakdown.

Simulation Time = Number of Samples x (mean of time between disruptions + mean time of duration of a breakdown)

If DS is used, the actual mean of the samples values in a sample will be the design values, so the calculation of the number of samples that are required for a certain simulation time is accurate.

The samples sizes of the events, where the time between occurrences are determined by other stochastic events, cannot be calculated from the mean of the distribution of the time between events, as this will not be known. For those events, where the occurrence is dependent on the behaviour of the model, the forecast of number of samples would (as discussed earlier) be based on design values.

8 RESULTS FROM THE RESEARCH

8.1 THE TARGET REAL LIFE MODEL

In this research the concept was that the discrete-event simulation was for the simulation of a transfer line and thus the proposed methods had the needs of such a production system in mind.

8.2 CONCERNS WITH CURRENT DEVELOPMENTS

The latest research into new RNGs and tests, seem to be out of line with the requirements of discrete-event simulation. The cycle lengths of some of the latest RNGs being proposed are very large and far removed from the samples sizes indicated by the rule of thumb such that "an adequate run length in discrete-event simulation is when there is at least 10 to 20 samples from each distribution" (Robinson 1994).

8.3 THE USE OF QUALITY CONTROL ON THE SAMPLES

In this thesis a new methodology of selecting RNGs or other proposed methods of introducing randomness was developed. It used the concept of Quality Control. In this methodology the sample created after the random numbers were transformed was tested for its "quality" not the string of random numbers. The driving force for such a decision being that a poor sample was to be avoided (It had been seen in early work by Saliby (1980) that samples that were highly representative gave closer result to the theoretical results) and there was some evidence that use of some transforms may lead to poor samples even with a RNG that passes the statistical tests. The quality being determined was

the degree by which the sample represented the true-life distribution, both in sample values and their sequence. The control was applied by using a development of Yule's test, where the actual distribution of the average values for sets of sequential numbers drawn from the total sequence is tested against the theoretical distribution (see Tocher 1960). In the thesis, sets of up to 10 sequential sample values were used. This requirement to use samples rather than the raw sequence of pseudo-random numbers required a transform to be identified to be used in the research. The decision was to use samples of negative exponential distribution since the literature indicated it had wide use and a real life sample of breakdown durations of a machine similar to those in the target real life case being used to give realism to the research were seen to be negative exponential distributed. (The negative exponential also has a precise transform.)

Thus for the research the sample was defined as 1000 sample values from a negative exponential distribution. The transform used to convert random numbers to the distribution was the inverse log transformation.

The quality control process was successful in obtaining a number of samples that were deemed "highly representative", as was shown by their high probability (reflected in their χ^2 values), of coming from the correct distribution.

8.4 CALIBRATION OF THE QUALITY CONTROL

To calibrate the test and also to evaluate how real random numbers would perform in the quality test, a source of random numbers was obtained. It was noted that a 4-byte integer was required to be produced from the stream of binary bits to meet the stringent statistical performance required (see Appendix 6). The test was set to accept the best 0.2% of the samples created by the real random numbers. Thus only "highly representative" samples would be accepted. The performance in terms of the percentage of sets of real number

that give acceptable "highly representative" samples is thus defined by the parameters of the test, but the calibration did not dictate distribution of the "quality measure". The "quality measure" was calculated by a weighted sum of the χ^2 values for the ten sequence sizes.

8.5 PERFORMANCE OF THE MLCGS

When the quality control was applied to the five MLCGs that passed Knuth's "spectral" test, the result was that they gave similar results to the real random number both in the percentage passing the quality test and in the distribution of the "quality measure". Two other MLCGs, including a discredited random number generator RANDU, gave similar results. There was no statistical difference in the quality of the samples produced by all the seven MLCGs. However there was a vast difference in quality in the samples depending on the particular seed chosen.

The benefit of the MLCG over using a stream of real random numbers is that the seed may be selected from the list of high performing seeds (particular to that MLCG).

It was demonstrated, contrary to the statements in the general literature, the choice of seed was very important and apparently more important than the choice of MLCG (at least for samples sizes of 1000 values).

Thus a MLCG with a selected seed will outperform a random sequence or a MLCG with a random choice of seed, 99.8% of the time. This includes RANDU. (Of cause if the random number stream was quality checked and only the sequences producing highly representative samples were saved onto disk, theses recorded sequences would obviously match those from the MLCG with selected seeds.)

Since there are a number of MLCGs that pass the "spectral" test it would seem prudent to restrict the selection of the MLCG to be used from these passing the "spectral" test. Thus the tables of seeds giving highly representative samples given in Appendix 3 are for the five MLCGs that pass the "spectral" test. It was noticed during the research that adding an offset to a MLCG (in this case the one termed "Flying" see table 5.3) that had a full cycle without an offset, caused it no longer to have a full cycle.

The conclusion was that MLCG with a selected seed can provide highly representative samples and will outperform real random numbers chosen at random 99.8% of the time.

8.6 USE OF SMALLER MODULI WITH MLCGS

The situation that discrete-event simulation was requiring much smaller samples than other applications of random numbers, led to the investigation of MLCG with smaller moduli. These appeared to give better results than the larger moduli. However with the size of moduli examined (1021 to 524287) and with sample sizes of 1000 there was a large amount of overlap of the sequences. Tables of the seeds that gave highly representative and non-overlapping samples of 1000 sample values for two of these smaller moduli (262139 and 524287) are given in appendix 3.

For smaller samples the smaller moduli MLGC outperform large moduli MLCGs. The use of smaller moduli for small sample sizes is worthy of more research.

8.7 PERFORMANCE OF THE MERSENNE TWISTER

One of the latest developments in RNGs is the Mersenne Twister. This has a massive cycle such that it may be considered that for any practical consideration that any sequence will not repeat. When samples created by this RNG were tested they were found to be statistically the same as those created by the real random numbers. The concern that with a large cycle they would be inferior to the seven MLCGs previously tested were unfounded. Again the seed used by the MLCG to generate the 624 seeds required by the Mersenne Twister was very important. The seeds that gave the highest quality samples with the Mersenne Twister are given in Appendix 3.

The Mersenne Twister is a viable alternative to the MLCGs but has no great advantage.

8.8 PERFORMANCE OF DESCRIPTIVE SAMPLING

Descriptive Sampling was found to produce more satisfactory samples than MLCGs and real random numbers. Descriptive Sampling produce greater than three times the number of "highly representative" samples. The quality of its samples was superior to the MLCGs, with 90% of its samples being equal or superior to the top 10% of those produced by the MLCGs.

The effect of multiple shuffles did not seem to have any predictable effect either an improvement or deterioration in the quality of the sample. In this thesis only a single shuffle was made. It was found that the choice of MLCG to perform the shuffle was not important but the seed selection was very important. The seeds giving the high quality samples using Descriptive Sampling are given in Appendix 3.

Descriptive Sampling gave more and better "highly representative" samples than MLCGs, the Mersenne Twister, or real random numbers.

8.9 SPEED OF PROCESSING

When considering processing speed, the MLCGs were found to be fastest and the Mersenne Twister was the slowest, but all in practical terms were acceptable in the concept of discrete-event simulation.

MLCG are fast but the Mersenne Twister and Descriptive Sampling are acceptable for Discrete-event Simulation

8.10 EFFECTS OF INACCURATE FORECASTS OF SAMPLE SIZES

Since the selection of seed for the MLCGs and for the Mersenne Twister was based on the sample size and Descriptive Sampling performance was considered to be dependent on the correct estimation of sample size, the effect of underestimating the sample size was considered. Errors of forecasting the sample size of up to 100% (the required sample size was 2000 rather than the estimate of 1000) were considered and Descriptive Sampling was found to be less affected.

Descriptive Sampling's performance in producing "Highly Representative" samples is less affected by underestimating sample sizes than MLCGs and DT.

8.11 REQUIREMENT FOR WARM UP

It is reasonable to accept that if the production rate of our target system the transfer line was being measured that the measurements should only commence after parts are being produced. Thus a basic system fill is accepted. However it is normally accepted that for long-term measurement of production rate the measurements should be delayed until a "steady state" is achieved. This research showed that using a section of the real life situation a queue, that the system could not be placed in the steady state condition and no length of run would leave the queue in such a state. The research showed that it was detrimental to delay commencing the measurements beyond the basic system fill. The mathematical model showed that Blomqvist's result (more accurate results with no delay in measurement) held for absolute error as well as for squared difference. (At least for the example analysed.) Indeed when investigating the effects on accuracy measurements of a number of short runs against a long run, a warm up period was used and the initial queue size created for a range of machine utilisations was the empty queue!

Except for a system fill no warm up is required and the loss of measurements is detrimental to the simulation study due to the loss of accuracy.

8.12 REMOVING THE BIAS DUE TO THE INITIAL CONDITIONS

The research confirmed that the measured average queue size was affected by the initial size of the queue, but that the effect was not removed by delaying measurement. The only way was to have a long simulation run. If "forgetting time" is defined as the time the system has to run in order for the statistical behaviour to be independent of the initial state, the simulation run time required to simulate a queue, in order to remove from the measurement of average queue size the effect of the initial queue size, was twice the "forgetting time".

Bias in measurements due to starting conditions can be removed by lengthy runs; it cannot be removed by discarding early readings.

8.13 THE NUMBER OF REPLICATES

It is frequently stated that accuracy is improved by using a number of replicates rather than a lengthy run. This research demonstrated that this was not true for the queue. Since queues are frequently found in discrete-event models and it is indeed possible to state a case that all discrete-event models contain queues, this result may well be general.

In the research a run of 2000 items was found to be more accurate than eight runs of 250. This supports the result that the bias due to the initial conditions requires a lengthy run.

Greater accuracy is obtained from one long run than a number of shorter runs.

8.14 DETERMINING THE RUN LENGTH

A scheme was developed to enable a run time to be estimated that would enable a representative number of events to occur during the simulation. The method described assumes the engineer is able to identify an unusually event and the minimum number of occurrences required for the simulation run to be acceptable as representative.

It was demonstrated that Descriptive Sampling was able to ensure that a representative number would occur with a shorter run time than would be achieved from a RNG that was mimicking true random numbers or indeed true random numbers. This was true even if the RNG was used with a selected seed that assured a highly representative sample.

Descriptive Sampling can ensure that a certain number of representative events occur in a shorter run than any method of introducing randomness that mimics real random numbers. It can also guarantee the design rates (e.g. breakdown rates) are accurately reflected in the simulation run,

8.15 REPLIES TO THE ORIGINAL SPECIFIC QUESTIONS

In section 3.13 it was stated that to obtain the information necessary to make the decisions on what methods should be chosen certain specific questions were asked. As discussed in the previous parts of this section, the results to these questions and studies were:

1) For the specific questions on the introduction of suitable randomness:

- A quality check was applied successfully to obtain samples that were "highly representative".
- MLCG, MT and DS did give quality samples and thus were able to be used but the seed selection was very important.
- MLCGs were able to be used but DS outperformed them.
- As stated above the seed selection was seen as vitally important, at least with samples of 1000.
- It was found that DS was less sensitive to any under-forecasting of the required sample size.

2) The answers to the specific question on the initial conditions were:

- Only a warm up period to obtain system fill was necessary.
- Measurements should commence immediately after the system fill.

3) On the question of should there be one long run or a number of short runs the analysis led to the conclusion:

- Greater accuracy was obtained from one single long run.

4) The need to have an initial run that was adequate to prevent false conclusions being drawn was answered by:

- A procedure was developed that would ensure a representative number of extreme events would occur. It was found in this respect, DS was able to obtain such a run with least resources.

8.16 REPLIES TO THE "KEY" ISSUES

By answering these questions the "key" issues raised in section 1.2 have been successfully addressed, although the results reject much of the "received wisdom" currently available in the discrete-event domain, such as the relative inconsequentiality of seed values, the requirement to reach a steady-state before commencing measurements, the need for an extended "warm up" to prevent distortions from the start up position, that multiple short runs are better than a single equivalent long run, the importance of "proper" random number generation that mimic closely the behaviour of real randomness.

The main requirement from specialised simulation software is that they should include Descriptive Sampling as a facility.

If they are to continue to offer a MLCG as the standard then it should be made possible to input a seed value rather than just a stream number with no control on the actual seed used.

Ideally they should include as part of the system the ability to select seeds based on the quality control procedures described in this thesis. Such software is demonstrated by the prototype discussed later in section 10.

One problem met during this research was the lack of knowledge on what sample sizes were actually being used. It would be helpful if the sample sizes actually used were reported by the packages.

9. IMPLICATIONS FOR THE DESIRED METHODS

The decisions on the methods to be used were:

- Randomness is to be provided by Descriptive Sampling
- Any of the MLCGs can provide the "pseudorandom" number generator. The Fishman and Moore generator is a suitable choice.
- Initial Runs should be single long runs with run length calculated using the method in section 7.
- Size of samples should be decided from the length of run calculations
- Seed selection should be based on the sample sizes.
- Seed selection should use the quality control, described in the thesis, to test that the samples created are "highly representative".
- Only a warm up sufficient to provide system fill.
- There need to be incorporate feedback to improve the operation of the methods. Information on the number of sample values actually used would assist in determining the sample required in further runs and thus select better seeds.

If it is desired to use specialised simulation software that cannot support Descriptive Sampling, any of the five MLCGs analysis will be suitable second choice. The Fishman and Moore generator performed well in creating quality samples. As with Descriptive Sampling the seeds used should be those that create samples of the desired distribution that pass the quality check.

If the specialised simulation software prevents any other its own MLCG, then the stream would need to be tested to see if they meet the quality requirement.

Descriptions of computer assistance in calculating the sample size and determining the seed for Descriptive Sampling are described in the next section.

10. COMPUTER SUPPORT

10.1 THE PROGRAM

To provide support to the selected methods, an interactive computer program was developed. It uses the programs that were created to perform the quality control. In order that the user interface could be improved, the programs were however rewritten in Visual Basic 6. At present only the Negative Exponential distribution is supported but the menu structure is in place for more distributions to be included. The program operates under Windows.

The computer system:

- Calculates the number of samples required in the simulation for the target machine (the machine with significant but infrequent events e.g. machine breakdowns),
- Determines the requested number of seeds such that each one of the seeds will guarantee a "highly representative" sample given the number of samples values required.

10.2 CALCULATING THE NUMBER OF SAMPLES

On running the program "PROTOT", the user is presented with a screen as shown on Diagram 10.1 with all the fields blanked. The user of the system is prompted to select from a drop-down menu of distributions the required distribution. (Although the system provides a comprehensive list only the Negative Exponential is programmed in the prototype)

The next stage will depend on the distribution chosen. In the case of the Negative Exponential, only the mean of the distribution is requested. The

system then places the curser in the "What is the tail?" field allowing the user to specify the "tail". There is a check that the "value of the tail" is "reasonable".

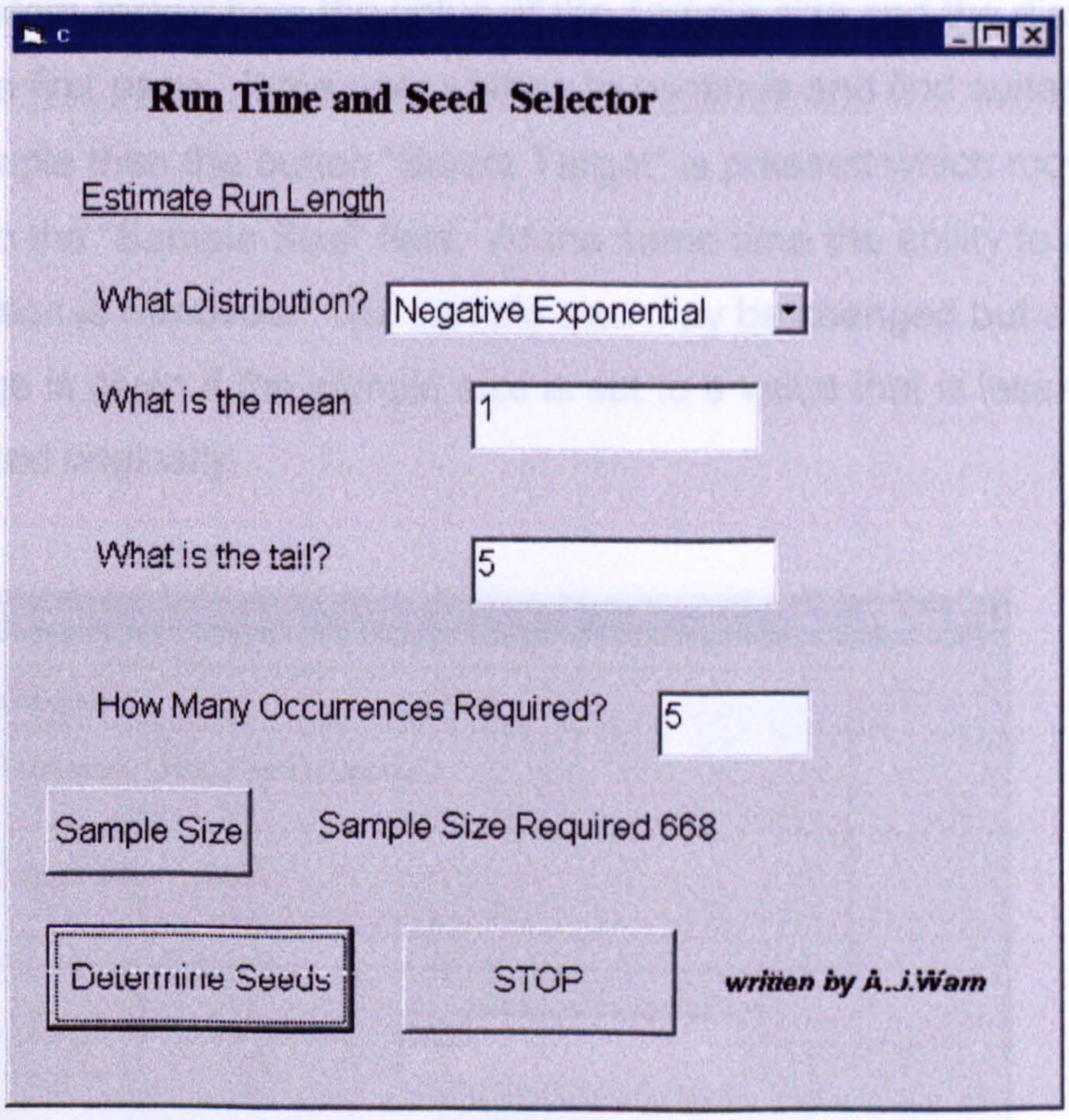


Diagram 10.1: The First Screen of the Prototype after the Sample Size has been Calculated

The number of occurrences required in the tail is next requested. Then pushing the button "Sample Size" causes the calculation of the sample size and this is displayed as shown in diagram 10.1.

Pushing the button "Determine Seeds" moves the system on to the second screen. (Diagram 10.2)

10.3 DETERMINING THE SEEDS

The system remembers the value of the sample size and the distribution chosen from the first page. If the user wishes to continue and find suitable seeds for this sample then the button "Select Target" is pressed which moves the sample size into the "Sample Size" field. At the same time the ability to input a distribution is removed. The sample size may be changed but a warning message is given if the sample size is set to a value that is less than that calculated originally.

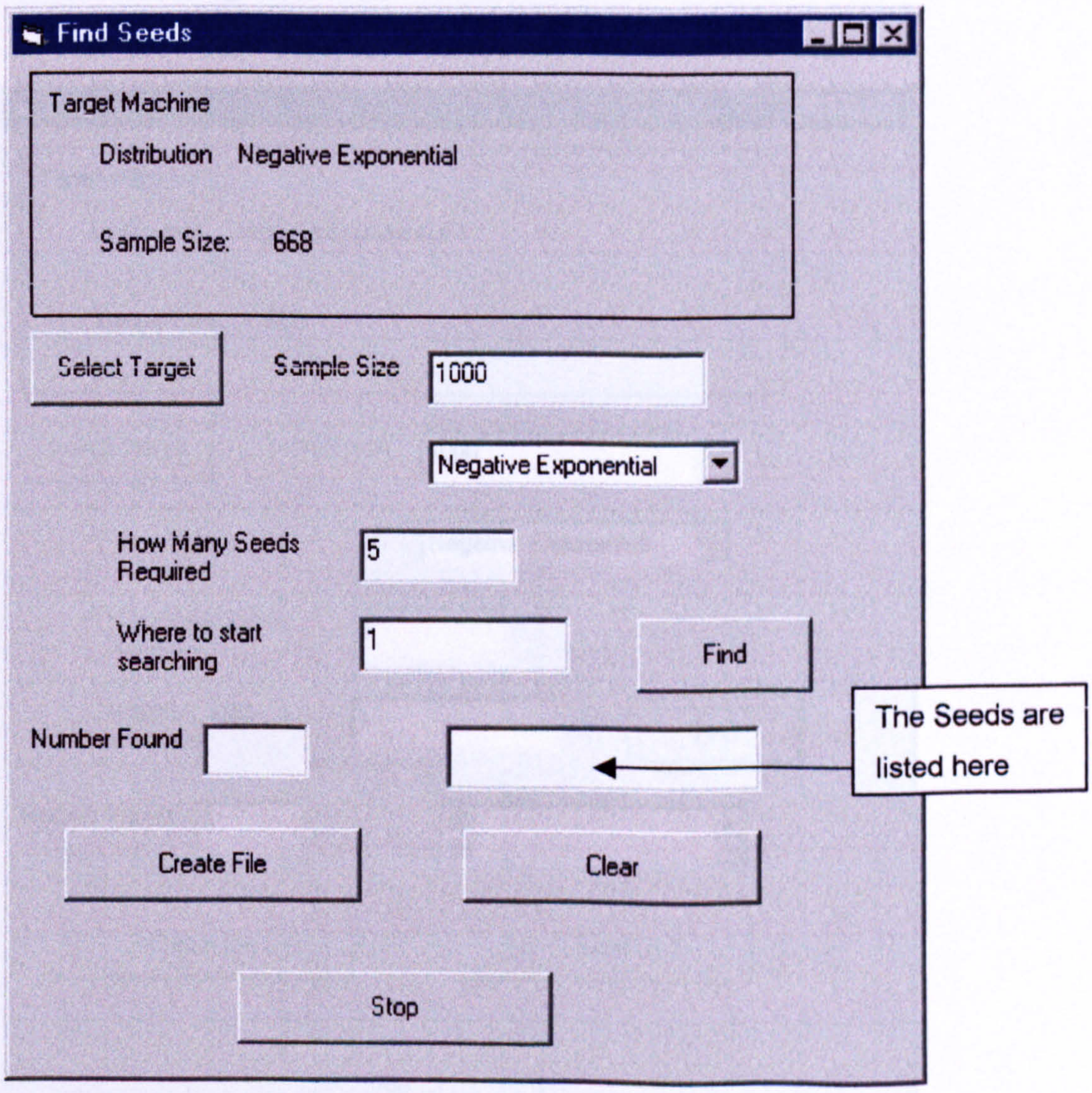


Diagram 10.2: The Second Screen Ready to search for the Seeds

The user may input a sample size and a distribution as is shown in diagram 10.2. The number of seeds required is then entered. The system will search for the seeds starting from a given number. This enables different seeds to be used in a simulation for the same sample size and distribution. This number is entered in the field "Where to start searching".

At this stage the "Find" button is pressed and the system finds the required seeds. The diagram 10.2 is at the stage of where the "Find" button is to be pressed. The number of seeds found, shown in the "Number Found" box, indicates the progress of the search. The seeds found are displayed in the field indicated in diagram 10.2.

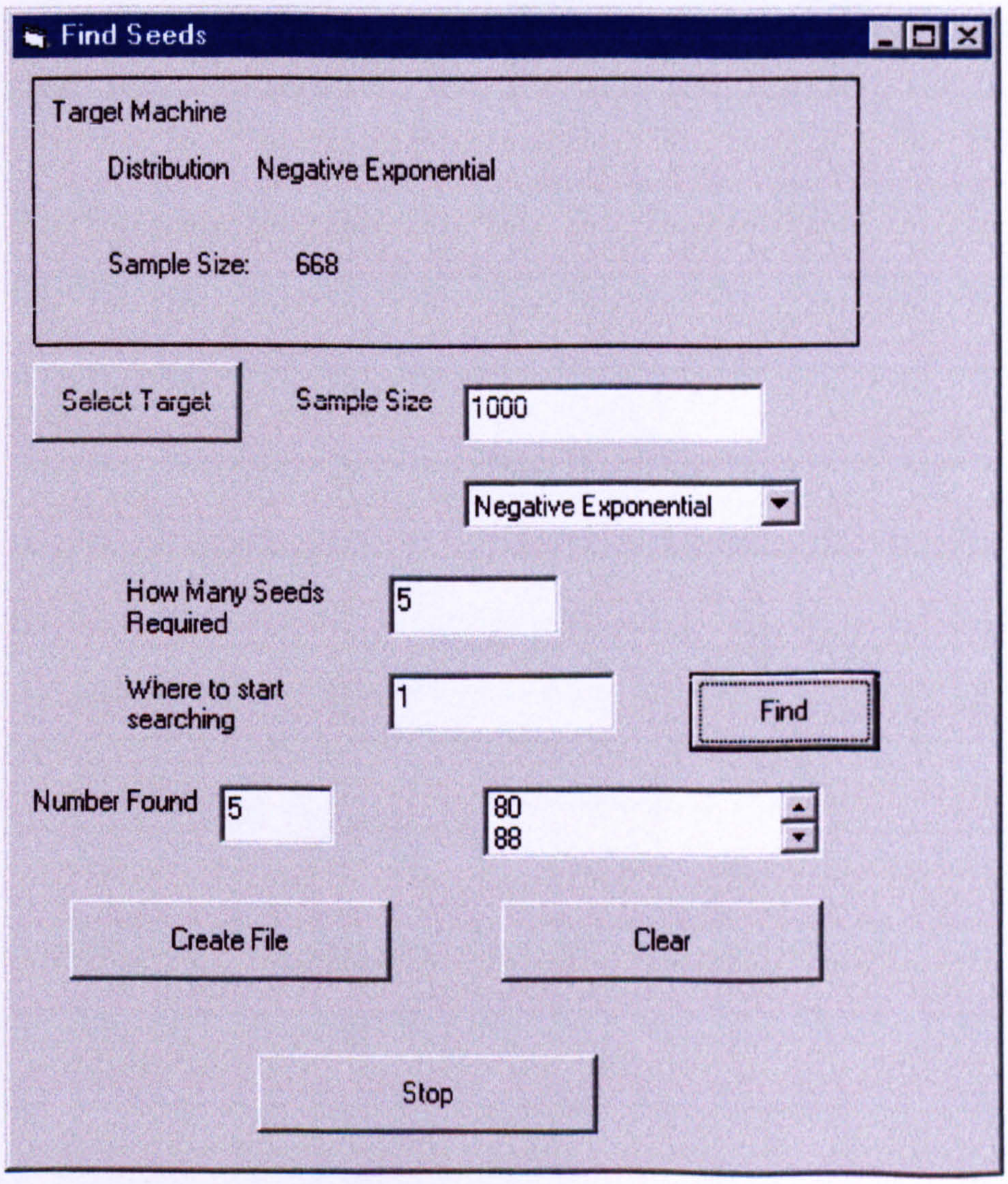


Diagram 10.3 Screen after the Seeds have been found

After the "Find" button is pressed the screen appears as in diagram 10.3. Pushing the button "Create File" causes a file of random deviates, one for each seed to be created (in the CSV format).

The "Clear" button clears the fields ready for the program to determine the seeds for the next variable.

11.0 FUTURE RESEARCH REQUIRED

The next step required is to extend the research and program development to other statistical distributions.

There is a need to determine the best approach for very small or very large sample sizes.

There is also a need to determine how best to develop the methods of creating samples with Descriptive Sampling that minimise the quality loss due to inaccurate forecasts of the number of samples required. Indeed there may be better ways to create the sequence than using a program like SHUFFLE.

There is a need for development of methods of determining the samples sizes, given the sample size of the "target" machine, required for all the other distributions in the simulation. This may have to be in the form of an expert system.

There is a need for the results of the simulation to be analysed to determine if the required accuracy has actually been obtained, or, if a comparison of two designs of a proposed facility is being made, to determine whether any results obtained so far are sufficient for any measured difference in their performance to be statistically significant.

It is necessary to consider other forms of production line and processes in order to determine if they have features that could lead to alternative results. A possibility may be in a system where there are lengthy feedback loops and it may be required to extend the warm up period until some items have been processed by the feedback loop.

APPENDICES

CONTENTS

APPENDIX 1: THE PROGRAMS USED IN THE STUDY

List of Programs

LNTIME

This program was to measure the time to convert to a negative exponential deviate. PTIME is the routine used in the whole study as a timing routine.

```
Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line
  1    $Pagesize:50
  2    c testing the time to perform a conversion to neg. exponential
  3           integer*4 n,i,k
  4           call Ptime(time1)
  5           do 100    k=1,100
  6           do 100    n=1,10000
  7           do 100     i=1,10000
  8             v=0.7
  9             expval=-dlog(1D0-v)
 10     100   continue
 11           call Ptime(time2)
 12           dur=time2-time1
 13            write(*,200)dur
 14     200    format(f20.2)
 15          stop
 16          end
```

PTIME

```
           subroutine Ptime(time)
  c        Prints time and returns time in seconds
           call Gettime(ihr,imin,isec,i100th)
           f100th=i100th
           time=(ihr*60+imin)*60+isec+f100th/100
           fsec=isec+f100th/100
           write(*,100)ihr,imin,fsec
 100       Format('  Time is ',i3,' Hr',i3,' Mins',F7.3,' Secs')
           return
           end
```

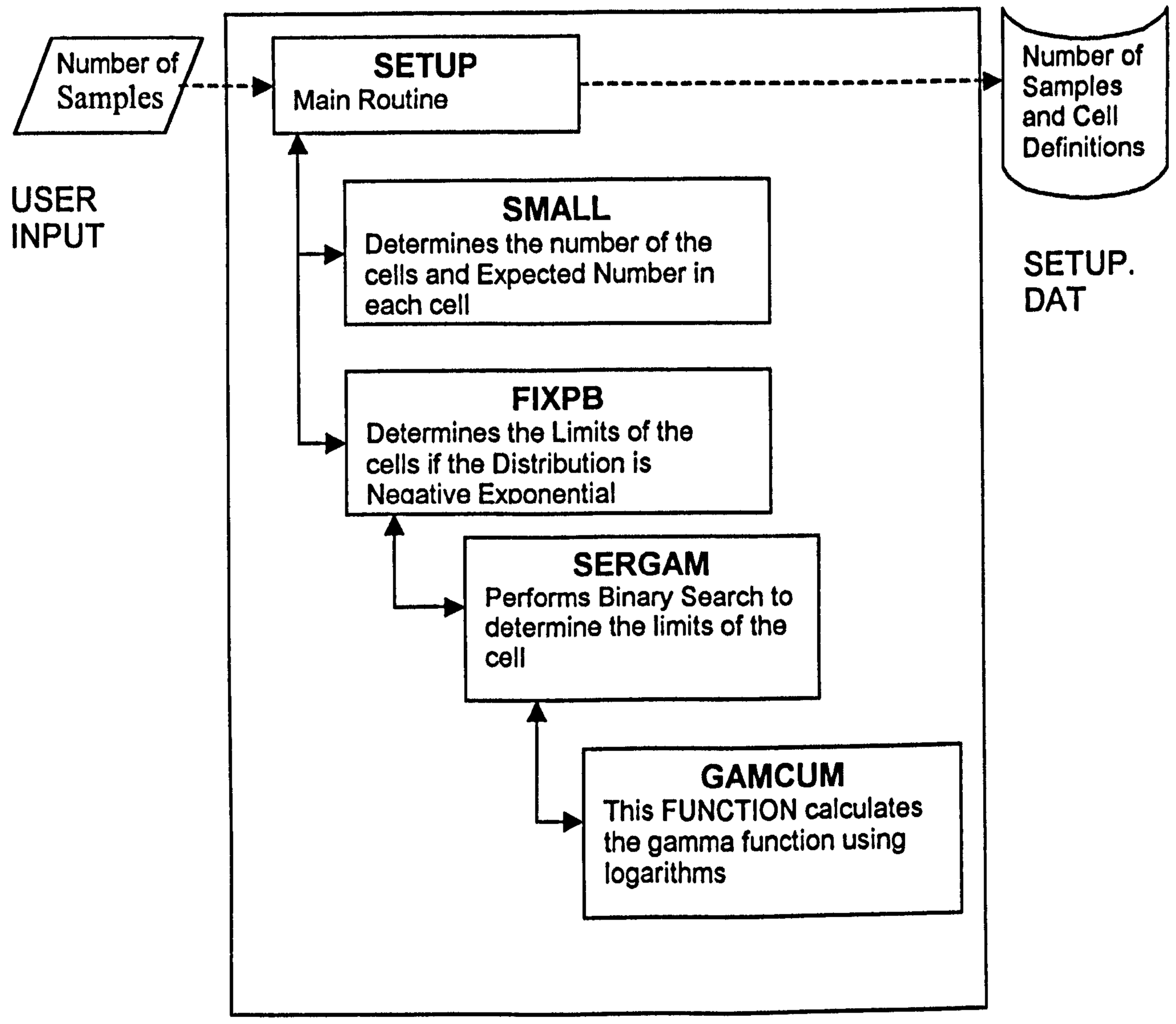


Diagram A1.1: Structure of Modules used in SETUP

SETUP

The SETUP program sets up the control file to enable the Quality Measure to be calculated. It uses FIXPB that is programmed for the Negative Exponential Distribution and would need to be replaced for other distributions.

```
PAGE  1
SETUP
Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line
     1  $Pagesize:45
     2  c Program  SETUP
     3  c This program creates the arrays required for the CHI Square
     4  c          test and creates a file
     5  c
     6  c The array A is dimensioned 5000
     7  c May need to be re-dimensioned dimA is set to 5000
     8  c If A is re-dimensioned change value of dimA
     9  c
    10  c This program uses subroutine SMALL and  FIXPB
    11  c
    12        Integer K(3),L(3)
    13        Real*8  A(5000)
    14        Integer*4 max, ntrials,minent,dimA
    15        Character*1 Drive
    16  c See above
    17        dimA=5000
    18  c Set up default values
    19          Minent =5
    20          Ntrials=1000
    21          max=10
    22        write(*,5)
    23   5    format('   If 1000 is not the number of samples'/
    24       +'   Enter value or just ENTER if 1000 is acceptable')
    25        read(*,6)i
    26   6    format(i10)
    27        if(i.ne.0) ntrials=i
    28  c
    29        write(*,10) Ntrials
    30   10   Format(///'  This program will now create the files'/
    31       + '  necessary for performing the calculation.'/
    32       + '  The structure of the files is dependent'/
    33       + '  only on the smallest required entry in a category'/
    34       + '  for the Chi Square Test.'/
    35       + '  The calculation is based on a sample of ',i5)
    36     write(*,11)
    37   11   Format(/'  The default value for the minimum number'
```

```
  SETUP
PAGE  2
Microsoft FORTRAN Optimizing Compiler Version 4.01
 Line#  Source Line
   38       + '  of entries is 5'/'  A larger value may be used'/
   39       + '  Enter the larger value '/
   40       + '  or just ENTER to accept default value')
   41         read(*,20)i
   42   20    format(i10)
   43     if(i.gt.0)minent=i
   44        if(minent.GE.5) go to 90
   45   c If Min Entry too small (i.e. less than 5)
   46    30    Continue
   47         do 80 n=1,10
   48           Write(*,50)
   49    50      Format('  Input value must be greater than 5')
   50           read(*,20)minent
   51           if(minent.lt.5) go to 80
   52           go to 90
   53    80    Continue
   54         Stop ' Too many tries!'
   55    90    continue
   56         if(dimA.lt.(mintrials/minent))
   57        + Stop ' In Setup:  Array A too Small'
   58     Write(*,100)
   59    100  format('  10 is the default maximum number'/
   60       +'  of Consecutive Observations.'/
   61       +'  Enter other value if required '/
   62       +'  Just ENTER to accept 10.')
   63        read(*,20)i
   64        if(i.gt.0) max=i
   65        write(*,150)
   66    150   Format('  What is the drive for the files?')
   67          read(*,200) drive
   68    200    Format(A1)
   69   c Open file
   70          open(10,file=Drive//':\setup.dat')
   71   c Write header
   72          write(10,300)ntrials,minent,max
   73    300    Format(3i10)
   74         do 500 Nevents = 1,max
   75         do 310 nc=1,dimA
   76    310   A(nc)=0
   77          nobs=Ntrials/nevents
   78   c Given the Number of Trial, the number if events in each
   79   c observation and the minimum number of observations in each
   80   c category for the Chi square test
```

```
PAGE   3
SETUP
Microsoft FORTRAN Optimizing Compiler Version 4.01
 Line#  Source Line
   81  c Calculate the categories so as to have the smallest number in each
   82        call SMALL(Nobs,minent,K,L,ncat)
   83        write(*,320)
   84   320  Format(' Cells size defined ')
   85  c Determine the array A with the boundary times from
   86  c a Gamma distribution with integer number of events.
   87        Call FIXPB(A,K,L,Nevents,ncat)
   88        write(*,330)
   89   330  Format(' Cell Boundaries Defined')
   90  c
   91  c Write out table
   92         write(10,400)nevents,(K(J),J=1,3),(L(I),I=1,3),ncat
   93   400     Format(8i10)
   94         write(10,450)(A(j),j=1,ncat)
   95   450    format(250f12.6)
   96   500  continue
   97        stop         '    File setup.dat has been created'
   98        end

No errors detected
```

SETUP needs to run if the number of samples, minimum number in a cell, the maximum number of sample values to summed to produce the test statistic need to be changed. As previously stated if the distribution to be tested is to change then FIXPB needs to be changed.

SMALL

```
Microsoft FORTRAN Optimizing Compiler
Line#  Source
 1  c
 2        Subroutine SMALL(Nobs,minent,K,L,ncat)
 3  c This subroutine determine the largest number of categories
 4  c it also determines the number in each category
 5  c
 6  c Each category holds at least Minent
 7  C Nobs is the number of samples
 8  C Ncat is the number of cells (or categories)
 9  c K is the output array of expected counts
10  c L is the number of cells, bottom, middle and top
11  c Written by A.J.Warn 2002
12  c
13        Integer*4 Nobs,minent,K(3),L(3)
14  c
15        NB=Nobs/Minent
16        If (NB.LT.2)
17       +  Stop ' Too Few Observations in Subroutine SMALL'
18  c Calculate K the maximum number of categories each with a
19  c minimum of minent expected values
20  c
21        MS=nobs/NB
22        MX=mod(nobs,MS)
23        ML=(NB-MX)/2
24        MU=NB-MX-ML
25              K(1)=MU
26        L(1)=MS
27        K(2)=MX
28        L(2)=MS+1
29        K(3)=ML
30        L(3)=MS
31        ncat=K(1)+K(2)+K(3)
32        Return
33        End
```

FIXPB

```
Microsoft FORTRAN Optimzing Compiler Version 4.01
Line#  Source Line
   1
   2        Subroutine Fixpb(A,K,L,Nevents,ncat)
   3  c This subroutine takes the definitions of the categories and
   4  c determines the upper limit of time of each category
   5  c The probability distribution is a Gamma with positive values
   6  c
   7  c Written A.J Warn 2001
   8  c
   9        Real*8 A(1),Totobs,Cobs,prob
  10              Integer*4 KC(3),K(1),L(1)
  11  c
  12  c     Calculate the Number of Observations
  13        Totobs=0
  14  C     Also change array K to cum values
  15        Kcum=0
  16        do 10 kcount=1,3
  17        Totobs=Totobs+K(kcount)*L(kcount)
  18        Kcum=Kcum+K(kcount)
  19        KC(kcount)=Kcum
  20   10   Continue
  21        if(KC(3).ne.ncat)
  22       +Stop 'Error detected in Fixfb category count'
  23  c     Determine the first probability
  24  c
  25     Cobs=0
  26        i=1
  27        do 400 nt=1,ncat-1
  28  c Note there is one less boundary between categories than
  29  c than there is categories
  30  c
  31       if(nt.GT.KC(1)) go to 100
  32            Cobs=Cobs+L(1)
  33            prob=Cobs/Totobs
  34            go to 200
  35   100  continue
  36        if(nt.gt.KC(2)) go to 110
  37          Cobs=Cobs+L(2)
  38          prob=Cobs/Totobs
  39          go to 200
  40   110  Continue
  41        if(nt.gt.KC(3)) Stop ' Logic Error in Fixpb'
  42          Cobs=Cobs+L(3)
```

```
Page 2 Fixpb
Microsoft FORTRAN Optimizing Compiler Version 4.01
 Line#  Source Line
   43 prob=Cobs/Totobs
   44   200  Continue
   45  c Determine time for this cum probability
   46  c with a Gamma distribution of mean 1 and
   47  c Nevents (a positive integer)
   48  c The result is put in A(nt)
   49  c 50    Check if Negative Exponential (nevents = 1)
   51  c
   52        if(nevents.gt.1) go to 300
   53  c Cumulative Distribution F(x)=1-exp(-x) where x is
   54  c standardised time.
   55  c       Thus x=-(ln(1-F(x))
   56           A(nt)=-dlog(1.0D0-prob)
   57           go to 400
   58   300  continue
   59       call sergam(prob,Nevents,A(nt),1000,1.0D-12)
   61   400  Continue
   62       Return
   63       End
```

SERGAM

```
Line          Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line
 1  c
 2        Subroutine Sergam(Target,Nevent,Answer,numit,acc)
 3  c
 4  c Written by Alan J Warn 2001
 5  c
 6  c This routine searches Cumulative Gamma Distribution to Find
 7  c Target probability and Resultant Time given as Answer
 8  c
 9  c       LIMITATIONS
10  c
11  c        This assumes positive integer number of events.
12  c    Target must be between 1 and 0
13  c
14  c   An acc accuracy can be specified
15  c         it can be positive or   negative
16  c
17  c      If accuracy (acc) is positive
18  c            a value greater than the target is acceptable
19  c      if within the limit but not below
20  c
21  c      If accuracy (acc) is negative a value less than the
13  c      target is acceptable if within the limit but not above.
14  c
15  c      numit is the maximum number of binary searches that will
16  c      be attempted.
17  c
```

```
Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line
 18         Real*8 Target,acc,Answer,top,bot,result,gamcum
 19         Integer*4 Nevent,numit
 20         if(target.lt.0.or.Target.gt.1)
 21        +     STOP 'Target Not Valid in SERGAM'
 22    20   Continue
 23         top=1
 24         do 100 n=1,29
 25         result=gamcum(Nevent,top)
 26         if(result.gt.target) go to 200
 27         top=top*2
 28   100    continue
 29         Stop '   Out of Range in  Sergam'
 30   200    continue
 31         bot=0
 32         do 400 n=1,numit
 33         Answer=(top+bot)/2
 34         result=gamcum(Nevent,Answer)
 35         if(result.gt.target) top=Answer
 36         if(result.lt.target) bot=Answer
 37         if(result.eq.target) return
 38         if(acc.lt.0) go to 300
 39         if((result.ge.target).and.(result.le.(target+acc)))
 40        +   Return
 41         go to 400
 42   300    if((result.le.target).and.(result.ge.(target+acc)))
 43        +   Return
 44   400    Continue
 45         Stop ' Did not get accuracy in number of iterations'
 46         End
```

GAMCUM

```
Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line
  1  c
  2  c This is a function that returns a GAMCUM
  3  c value for the cumulative probability of NEvents
  4  c Written by A J Warn 2001
  5        Real*8 Function GAMCUM(NEvents,Time)
  6        Real*8 Alpha,pdf,cum,value,total,tot,bot,lvalue,time
  7        Integer*4 NEvents
  8        Alpha=NEvents
  9        if(NEvents.le.0)
 10       +   Stop 'Error in GAMCUM Number of Events not positive'
 11  c
 12  c Check if the Negative Exponential if it is not go to 10
 13  c
 14        if(NEvents.gt.1) go to 10
 15           GAMCUM=1.0D0-DEXP(-Time)
 16           Return
 17  c
 18   10   Continue
 19        lvalue=-Time
 20        bot=0.0D0
 30        tot=0.0D0
 40        do 40 n=1,Nevent-1
 50        bot=bot+dlog(n)
 60        value=lvalue+dfloat(n)*(dlog(Time))-bot
 70        tot=tot+dexp(value)
 80   40    continue
 90        GAMCUM=1-dexp(-Time)-tot
100        Return
 101                End
```

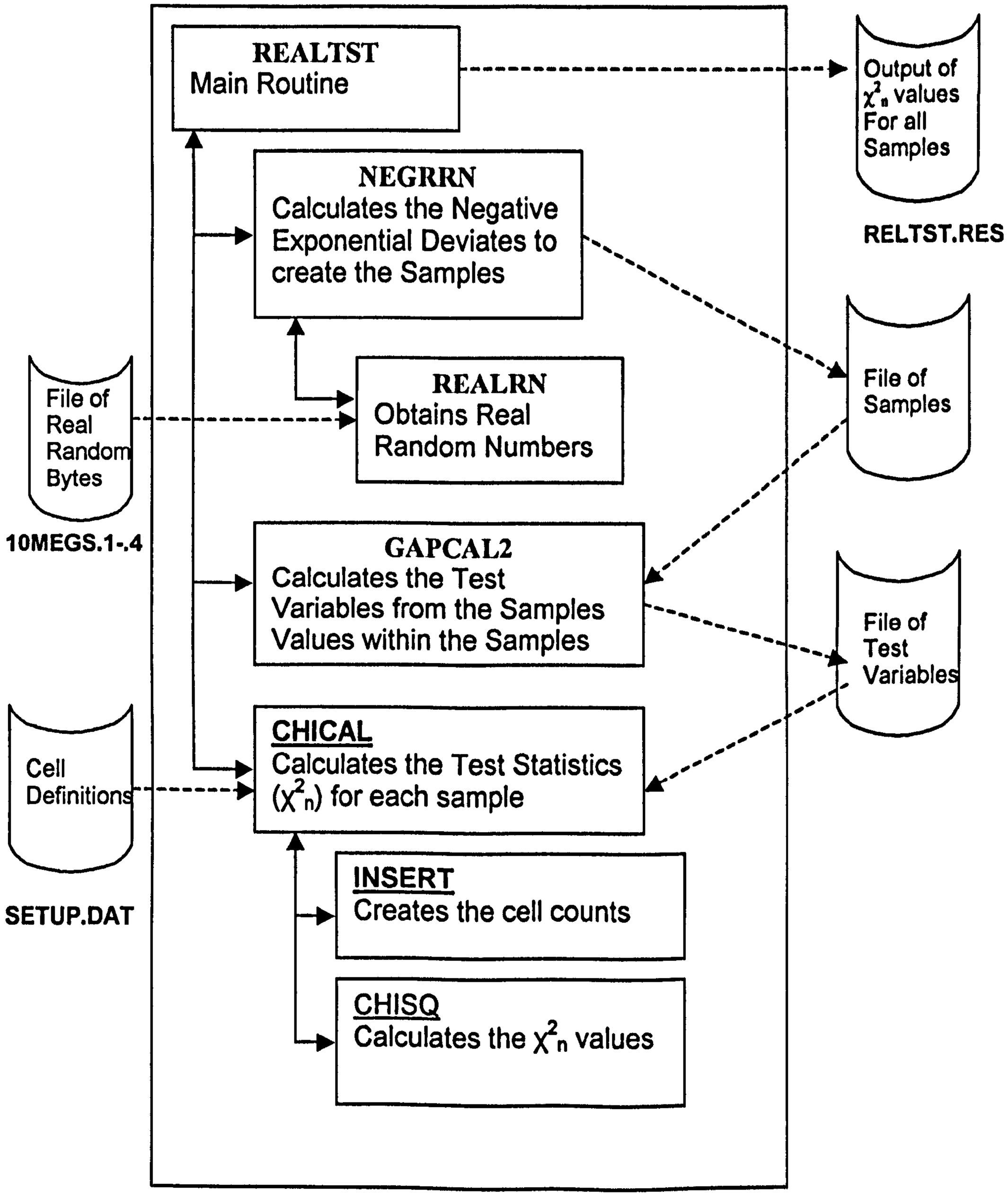


Diagram A1.2: Structure of Modules used in REALTST

REALTST

```
 $Pagesize:50
C REALTST Test
c This program is to evaluate Real Random Number
c capability to produce satisfactory samples
c from a Negative Exponential distribution
c
c Written by A,J.Warn 2002
c
c The Random Numbers are from random.org
c They are produced from atmospheric noise
c They were created by Mads Haahr
c They were downloaded from http://www.random.org
c
c It produces a file            lcgtest.res
c
c the first record is the description of the method used
c       the second is the number of cells used to calculate
c the chi squared value      then follows pairs of records
c    the set number
c    10 chi square values for sequences of 1 - 10
c    (default is 10 var. max)
c    if more than 20 it will write more than one record
c
      Real*8  A(250),AO(250,20),CHI(20)
      Real*8 seed
      integer*4 is, ntrial,max,ncata(20),iseed,iseed2,iseeds
       Character*1 Drive
       Character*50 iname
       Character*30 name
c Default Values
c  Max number in a sequence
       max=10
c Number of Neg Exp samples
       Ntrial=1000
c  A block size of records written on the work file
       Nblock =10
c First Select media for files
      write(*,10)
 10   Format(' What is the drive for the work files?')
       read(*,20) drive
 20    Format(A1)
c
c    set set numbering
      iseed=1
```

```
      write(*,30)
30    format (' How many sets?')
      read(*,40)iseed2
40    format(i10)
      iseeds=1
      open(11,file=drive//':realtst.res')
      call ptime(time1)
      write(11,50)
50    format('Real Random Numbers')
      do 300 is=iseed,iseed2,iseeds
      seed=is
c Now create the random neg exp deviates with mean 1 on a file
      Call NegRRN(seed,Ntrial,Nblock,Drive)
c Create the observations by combining the random neg exp deviates
      call gapc2(Drive,max)
c Calculate the sum of the percentage squared differences
c         (Chi square statistic)
c
      call chical(Drive,max,CHI,ncata)
c  if first record after header write number of catagories
c  Degrees of Freedom
      if (is.eq.iseed) write(11,60) (ncata(I),I=1,max)
60    format(20i5)
c write seed
      write(11,70)is
c write sum of percentage squared differences
70    format(i15)
      write(11,100)    (CHI(I),I=1,max)
100   Format(20f10.2)
300   Continue
c Now tell how long it took!
      call ptime(time2)
      time=time2-time1
      if(time.lt.0)time=time+(24*60*60)
      ihr=int(time/3600)
      imins=int((time-ihr*3600)/60)
      fsec=time-ihr*3600-imins*60
      write(*,400)ihr,imins,fsec
400   Format(' Time taken',I3,' Hrs',I3,' Mins',f7.3,' Secs')
      Stop
      End
```

negRRN

```
   Subroutine negRRN(s,Ntrial,Nblock,Drive)
C   This subroutine produces a file of Ntrial neg exp deviates
c    File is on drive Drive
c
c    They are blocked in blocks of Nblock
      integer*4 ,ntrial,nprod,nrand
      Real*8 s,output(1000),v
      Character*1 Drive,shove,rfile
      Character*8 MEG(4)
      Character*11 fname
      Integer frdata,fwork,once
      fwork=7
      frdata=15
c in June 2002 4 10 Meg files were used from RANDOM.ORG
c These were 10megs.1 10megs.2 10megs.3 10megs.4
     MEG(1)='10megs.1'
     MEG(2)='10megs.2'
     MEG(3)='10megs.3'
     MEG(4)='10megs.4'
     NFtop=4
c
c    if first time open first real random number file
      if(s.GT.1) go to 40
         NRfile=1
         nrand=0
         print '(a60)',
    +' Give the drive with the file containing files of random bits'
         read(*,30)rfile
```

```
 30    Format(a1)
       f name=rfile//':\'//MEG(NRfile)
       open(frdata,file=fname,form='binary')
       print '(a32)',' First Random Digit File Opened '
c Set indicator of new file
       once=0
       nerr=10
 40   Continue
c       Open work file
       open(fwork,FILE=Drive//':\cneg2.dat')
       rewind fwork
c
c Write header
        write(fwork,100)ntrial,nblock,s
 100  Format('Header',2i6,f20.0)
c
       nprod=0
 200  continue
      do 1000 nr=1,nblock
c  Determine a random number
  210    Continue
         call realrn(frdata,v,nerr)
c if nerr set to 0 no error
         if(nerr.eq.0)go to 250
c end of file was detected determine if last pass
         if(once.ge.3) go to    220
c set up for next pass
           rewind frdata
           print '(a14)',' Rewound file '
           nerr=0
c off set the numbers by one byte
           read(frdata,end=3200)shove
           once=once+1
           go to 210
  220   continue
c Close old file and open next not all used
          close(frdata)
          print  '(a25)',' Random Digit File Closed '
          NRfile=NRfile+1
          if(NRfile.gt.NFtop) go to 3000
            fname=rfile//':\'//MEG(NRfile)
            open(frdata,file=fname,form='binary')
            print '(a31)',' Next Random Digit File Opened '
            once=0
            nerr=10
          go to 210
 250    nprod=nprod+1
```

```
            nrand=nrand+1
c Calculate a value drawn from a Negative Exponential Distribution
            expval=-dlog(1D0-v)
            output(nr)=expval
            if(nprod.GE.ntrial) go to 2000
 1000             Continue
c                 A full block available to be written
            write(fwork,1001)nblock,(output(i),i=1,nblock)
 1001    Format(I3,100E20.12)
            go to 200
c Back to processing
 2000  continue
c write outlast block even if an incomplete block
            write(fwork,1001)nr,(output(i),i=1,nr)
            rewind fwork
            return
c           ***************************************************************************
c end of file error routines
 3000   Write(*,3100) nrand,NRfile,once+1,frdata,fname,s,nprod,nerr
 3100   Format(' In NegRRN after',i10,' Random Numbers the',i3,' file'
        +' read',i2,' times.'/' file number ',i3/' file name ',a12/
        +' value of set number ',f20.0/
        +' number of random numbers for this set ',i10
        +' error flag was ',i4)
          Stop ' End of Real Random Numbers before end of processing'
3200 continue
            write(*,3100) nrand,NRfile,once+1,frdata,fname,s,nprod,nerr
            stop' Abnormal end Check name of random digits file.'
                  End
```

REALRN

```
c This routine reads a file created by Mads Haahr
c It is able to be downloaded from
c    http://www.random.org
c It contains random bits generated from atmospheric noise.
c this subroutine returns a number between 0 and 1.
c Amended to read I4 June 2002
      Subroutine realrn(nfile,value,nerr)
c
c The routine assumes a file is on nfile
c Value is the returned value between 0-1
c   nerr is an error code:
c   nerr=0 no error
c   nerr=1 end of file value to be used
c
c        on input nerr=10 means a new file has started
c
     Real*8 value
     integer nfile
     integer*4 input
       if(nerr.eq.10) nread=0
       read(nfile,end=15)input
       nread=nread+1
c the most negative value is hex 80000000 this is seen as -2147483648
c the most positive value is hex 7FFFFFFF this is seen as  2147483647
c since we do not wish to have zero or one
c
       value=(input+2.147483649D09)/(4.294967297D09)
        nerr=0
       return
 15     nerr=1
       return
      end
```

GAPC2

```
      subroutine gapc2(Drive,max)
c
      integer*4 ntrial,nblock,nread,nrecin,irec,nevent,outcnt
      integer*4 max
      real*8 input(200),rem(10),vout(200),iseed
      Character*1 Drive
c  open input file
      open(7,file=Drive//':\cneg2.dat')
       read(7,20)ntrial,nblock
c
c     nread is the number of fields read from file
c     nrecin is the number of field being processed from current record
       nread=0
       nrecin=0
c
c create output file and header
       open(8,file=Drive//':\gapcal.dat')
        write(8,10) ntrial,nblock
 10     Format('output2',2i6)
        do 2000  nevent=1,max
            write(8,11) nevent,ntrial
 11        Format(2i6)
c
c here will be a loop for setting offset  (not used at present)
            offset=0
c rewind inout and read header
            rewind 7
            read(7,20)ntrial,nblock
 20         format(7X,2I6)
c
c Read the first data record
            read(7,30) nrecs,(input(i),i=1,nrecs)
 30         format(I3,100F20.12)
            nread=nrecs
c
c  save for later values up to offset value
           if(offset.eq.0) go to 200
              noff=0
              go to 50
c need to read another record
 40    read(7,30)nrecs,(input(i),i=1,nrecs)
       nread=nread+nrecs
 50    continue
```

```
        do 100 n=1,nrecs
             rem(n)=input(n)
           noff=noff+1
           if(noff.ge.offset) go to 200
 100     continue
         go to 40
c
c Commencing the processing
c starting with offset field

 200  nrecin=offset+1
        if(nrecin.LE.nrecs) go to 250
c   Need to read another record
          read(7,30)nrecs,(input(i),i=1,nrecs)
          nread=nread+nrecs
          nrecin=1
 250  Continue
        nrec=0
c nrec is number of field combined
c
c outcnt is the field in output record being currently processed
        outcnt=1
c
 300   Continue
c See if number of events has reached num. required
          if(nrec.ge.nevent) go to 400
c If not add field to total
           vout(outcnt)=vout(outcnt)+input(nrecin)
           nrec=nrec+1
           if(nrecin.ge.nrecs)go to 700
             nrecin=nrecin+1
          go to 300
 400    nrec=0
          if(outcnt.ge.nblock) go to 500
            outcnt=outcnt+1
          go to 300
c see if record should be written
 450    if(outcnt.LE.nblock) go to 700
c If it does
 500    continue
          irec=outcnt
         write(8,600)nevent,irec,(vout(i),i=1,irec)
         do 550 i=1,nblock
              vout(i)=0
 550  continue
 600  format(2i6,20f20.12)
         outcnt=1
          go to 300
 700  Continue
```

```
          if(outcnt.ge.nblock.and.nrec.ge.nevent) go to 900
c Check if end of file has been reached if so go to end of file
c processing
 800    if(nread.GE.ntrial) go to 1000
c       Read New Record
        read(7,30)nrecs,(input(i),i=1,nrecs)
        nread=nread+nrecs
        nrecin=1
        GO TO 300
 900    continue
        irec=outcnt
        write(8,600)nevent,irec,(vout(i),i=1,irec)
        do 950 i=1,nblock
          vout(i)=0
 950   continue
         outcnt=1
         nrec=0
          go to 800
c     Write out remaining record
 1000   continue
c is nrec is less than nevent then outcnt must be reduced by 1
         if (nrec.lt.nevent)outcnt=outcnt-1
         if(outcnt.gt.0) write(8,600) nevent,outcnt,(vout(i),i=1,outcnt)
         do 1010 i=1,nblock
            vout(i)=0
 1010  Continue
         outcnt=0
 2000 continue
         rewind 8
         Return
         End
```

CHICAL

```
$Pagesize:50
      subroutine chical(Drive,max,CHI,ncatA)
c
      integer*4 ntrial,nblock,nread,nrecin,irec,nevent,outcnt
      integer*4 max,nbase,KA(20,3),LA(20,3),ncatA(20),B(250,20)
      integer*4 K(3),L(3)
      real*8 vout(250),ATAB(250,20),a(250),CHI(20),value
      Character*1 Drive
c
c
c
      open(8,file=Drive//':\gapcal.dat')
      open(10,file=Drive//':\setup.dat')
c Read Headers
      Read(8,10) ntrial,nblock
 10   Format(7X,2i6)
       read(10,20)nbase,minent
 20   Format(2I10)
c
c
      Do 400 ncom=1,max
          read(10,40)nevents,(K(J),J=1,3),(L(I),I=1,3),ncat
 40       Format(8i10)
          if(ncom.ne.nevents) stop ' Incorrect Set-up File'
           do 100 j=1,3
              KA(nevents,J)=k(J)
              LA(nevents,J)=L(J)
 100      Continue
          ncata(nevents)=ncat
          READ(10,200)(A(j),j=1,250)
 200      format(250f12.6)
           do 300 j=1,250
               ATAB(J,nevents)=A(j)
 300       continue
 400   continue
        rewind (10)
c
```

```
        do 1000  nevent=1,max
          read(8,450) nw,ntrial
 450      format(2i6)
          If(nw.ne.nevent)
     +       STOP ' gap cal Exp File out of sequence'
c  Calculate the number of full blocks
           itop=ntrial/(nevent*nblock)
c  If number of readings remain
c  is enough there is another record
          irem=ntrial-itop*(nevent*nblock)
          if(irem.GE.nevent)itop=itop+1
          do 700 nrec=1,itop
               Read(8,500)nev,irec,(vout(i),i=1,irec)
 500           format(2i6,20f20.12)
             do 600 i=1,irec
               value=vout(i)
               Call Insert(Atab,B,ncata,value,nevent)
 600         continue
 700     continue
 1000   Continue
        open(6,file=drive//':\counts.dat')
        do 1010 n=1,20
          write(6,1020) (B(i,n),i=1,250)
 1010  continue
 1020  format(250i10)
         Call chisq(B,KA,LA,CHI,max,nbase,ntrial)
        do 2000 i =1,250
        do 2000 j=1,20
            b(i,j)=0
2000 continue
        rewind(8)
        rewind(10)
        rewind(6)
          Return
          End
```

INSERT

```
      Subroutine Insert(ATab,B,ncell,value,nvalues)
      Integer*4 B(250,20),ncell(20)
      Real*8 ATab(250,20),value,CHI(20)
      ntop=ncell(nvalues)-1
       do 500 i=1,ntop
        If(Value.Gt.ATab(i,nvalues))go to 500
        B(i,nvalues)=B(i,nvalues)+1
        Return
500    Continue
       B(ncell(nvalues),nvalues)=B(ncell(nvalues),nvalues)+1
      Return
      End
```

CHISQ

```
      Subroutine chisq(B,K,L,CHI,max,nbase,ntrial)
c this uses B and K,L to calculate Chi square for each nevents
      Integer*4 B(250,20),K(20,3),L(20,3)
      Integer*4 max,nbase,ntrial
      Real*8 CHI(20),cum,exp
      do 500 nevent=1,max
        cum=0
        ncat=0
        do 400 i=1,3
          if(K(nevent,i).lt.1) go to 400
          do 300 j=1,K(nevent,i)
            ncat=ncat+1
            if(ncat.GT.250) go to 600
            exp=dfloat(L(nevent,i))*dfloat(ntrial)/dfloat(nbase)
            Cum=cum+((b(ncat,nevent)-exp)**2)/exp
 300      Continue
 400   continue
      CHI(nevent)=cum
500   Continue
      Return
600   write(*,601)(K(nevent,i),i=1,3),nevent
601   Format(' K is ',3i6, ' number of events ',i6)
      write(*,602) ncat
602   Format(' ncat is ',i6)
      stop ' *** Out of range in chisq '
      End
```

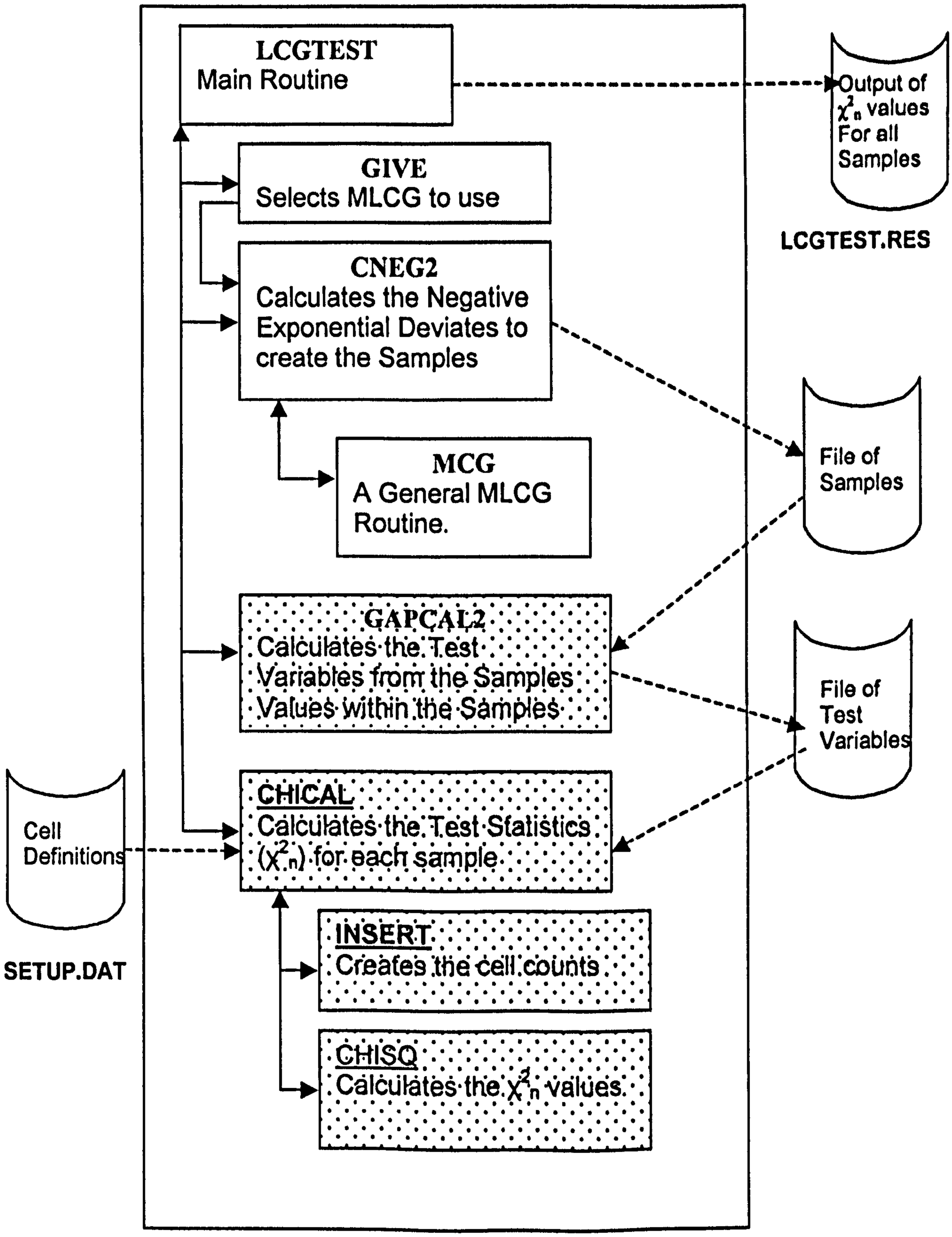


Diagram A1.3: Structure of Modules used in LCGTEST

LCGTEST

```
$Pagesize:50
C LCGTest
c This program is to evaluate LCGs/seeds capability to produce
c satisfactory samples from a Negative Exponential     distribution
c
c Written by A,J.Warn 2001
c
c       The LCG may be selected from a list or input own definition
c
c The seed range may given also a step thus allowing only
c odd seeds
c
c It produces a file            lcgtest.res
c
c the first record is the description of the LCG used
c       the second is the number of cells used to calculate
c the chi squared value
c then follows pairs of records
c    the seed
c    10 chi square values for sequences of 1 - 10
c       (default is 10 var. max)
c if more than 20 it will write more than one record
c
c The LCG is written using double precision to allow
c for a modulus greater than 2**31-1
c
c It is not aimed at speed!
c
c
      Real*8  A(250),AO(250,20),CHI(20)
      Real*8 Mult, Modulus, Offset, seed
      integer*4 is, ntrial,max,ncata(20),iseed,iseed2,iseds
       Character*1 Drive
       Character*50 iname
       Character*30 name
c Default Values
c  Max number in a sequence
       max=10
c                Number of Neg Exp samples
      Ntrial=1000
c  A block size of records written on the work file
      Nblock =10
```

```
c First Select media for files
      write(*,10)
 10   Format(' What is the drive for files?')
      read(*,20) drive
 20    Format(A1)
c Now Provide the information on the generator
      call Give(name,Mult,Modulus,Offset)
c Confirm selection
       write(*,40)name,Modulus,Mult,Offset
 40    Format('  Name of Generator ',a30/
    +' Modulus    ',f20.0/
    +' Multiplier ',f20.0/
    +' Offset     ',f20.0)
c
c                                    set seed              range
                                     write(*,50)
 50    format(' What is the initial seed?')
       read(*,51)iseed
 51    Format(i15)
       write(*,52)
 52    format (' What is the final seed?')
       read(*,51)iseed2
       write(*,53)
 53   format (' What is the step?')
       read(*,51) iseeds
       open(11,file=drive//':lcgtest.res')
       call ptime(time1)
       write(11,55)name,ntrial,max,iseed,iseed2,iseeds
 55   format(a30,5i10)
      do 300 is=iseed,iseed2,iseeds
       seed=is
c Now create the random neg exp deviates with mean 1 on a file
      Call Cneg2(Mult,Modulus,Offset,seed,Ntrial,Nblock,Drive)
c Create the observations by combining
c the random neg exp deviates
      call gapc2(Drive,max)
c Calculate the sum of the percentage squared differences
c       (Chi square statistic)
c
       call chical(Drive,max,CHI,ncata)
c  if first record after header write number of catagories
c  Degrees of Freedom
       if (is.eq.iseed) write(11,60) (ncata(I),I=1,max)
 60     format(20i5)
c write seed
      write(11,70)is
c write sum of percentage squared differences
 70    format(i15)
```

```
      write(11,100)  (CHI(I),I=1,max)
 100   Format(20f10.2)
 300   Continue
c Now tell how long it took!
       call ptime(time2)
      time=time2-time1
      if(time.lt.0)time=time+(24*60*60)
      ihr=int(time/3600)
      imins=int((time-ihr*3600)/60)
      fsec=time-ihr*3600-imins*60
      write(*,400)ihr,imins,fsec
 400   Format(' Time taken',I3,' Hrs',I3,' Mins',f7.3,' Secs')
      Stop
      End
```

GIVE

```
$pagesize:50
c
      Subroutine Give(Descr,Mult,Modulus,offset)
      Character*30 Descr,D(20)
      Character*50 idescr
      Character*1 y
      Real*8 Mult,Modulus,offset,Am(20),ab(20),ao(20)
c
      D(1)=' F&M Line 18 Knuth(1997)'
      Am(1)=62089911
      Ab(1)=2**31-1
      D(2)='Flying Line 17 Knuth(1997)'
      Am(2)=314159269
      Ab(2)=2**31-1
      D(3)='Lewis+ Line 19 Knuth(1997)'
      Am(3)=16807
      Ab(3)=2**31-1
      D(4)='Uniran'
      Am(4)=630360016
      Ab(4)=2**31-1
      D(5)='Randu'
      Am(5)=65539
      Ab(5)=2147483648.
      D(6)='L"Ecuyer Line 21 Knuth(1997)'
      Am(6)=40692
      Ab(6)=2**31-249
      D(7)='L. C. Killingbeck'
      Am(7)=2650845021.0D0
      Ab(7)=4294967296.0D0
      Number=7
c
      Write(*,10)
 10    Format('  Now set the Random Number Generator'/
     +'  This Program uses only MLCGs'/
     +'  Here is a list of pre programmed values for some'/
     +'  standard Generators.'//)
      write(*,20)
 20   Format(' Ref #',' Description  ')
      do 50 n=1,number
      write(*,30)n,D(n)
 30   Format(i4,2x,a30)
 50   continue
```

```
      do 100 n=1,10
         write(*,55)
55       Format('  Type the ref # to select one, or 0 for Own')
         read(*,60)nr
60       Format(i6)
         if (nr.gt.0) go to 200
         if (nr.gt.number) go to 100
          go to 300
100   continue
        Stop '  Incorrect Ref # being enteredl'
200    Descr=D(nr)
         Mult=AM(nr)
         Modulus=Ab(nr)
         Offset=AO(nr)
         go to 500
300    Continue
         write(*,305)
305    Format('  What description do you wish to give the MCG?'/
     + ' 30 characters maximum.')
         read(*,306)iDescr
306    format(a50)
         Descr=iDescr
         if(Descr.NE.iDescr) write(*,307) descr
307    format('  Name given too long will truncate to:'/'  ',a30)
         Write(*,310)
310    Format('  If Modulus to be used is 2^31-1, enter y'/
     + '  if not just Enter')
         Read(*,320)y
320    Format(a1)
         modulus=2**31-1
          if(y.eq.'y'.or.y.eq.'Y')go to 400
          write(*,330)
330    Format(
     +'   Enter value of Modulus'/'  finish with a decimal point')
          read(*,340)modulus
340    Format(f20.0)
400    Continue
        write(*,410)
```

```
      do 450 n=1,10
410   Format(
     +' Enter value of Multiplier'/' finish with a decimal point'/
     +' Value must be less than the Modulus')
        read(*,340)mult
        if(mult.LT.modulus) go to 460
450    Continue
      Stop ' Entering a Multiplier Bigger than the Modulus'
460   Continue
      write(*,470)
470   format(
     +' Enter value for offset if any if none enter 0.'/
     +' Finish with a decimal point')
      read(*,340)offset
500    Continue
       return
       end
```

MCG

```
        subroutine MCG(s,a,b,c)
C This subroutine uses Real*8 variables for a MLCG
C
C This subroutine takes in a seed if the initial time called
C or the previous Random Number (not normalised to 0-1)
C and gives a Random Number not normalised.
C
C s is the seed or Previous Random Number as input and the next Random
C Number as output.
C b is the modulus
C a is the multiplier
C is the fixed value or offset.
C
      Real*8 a,b,s,c
      s=dmod((a*s+c),b)
      return
      end
```

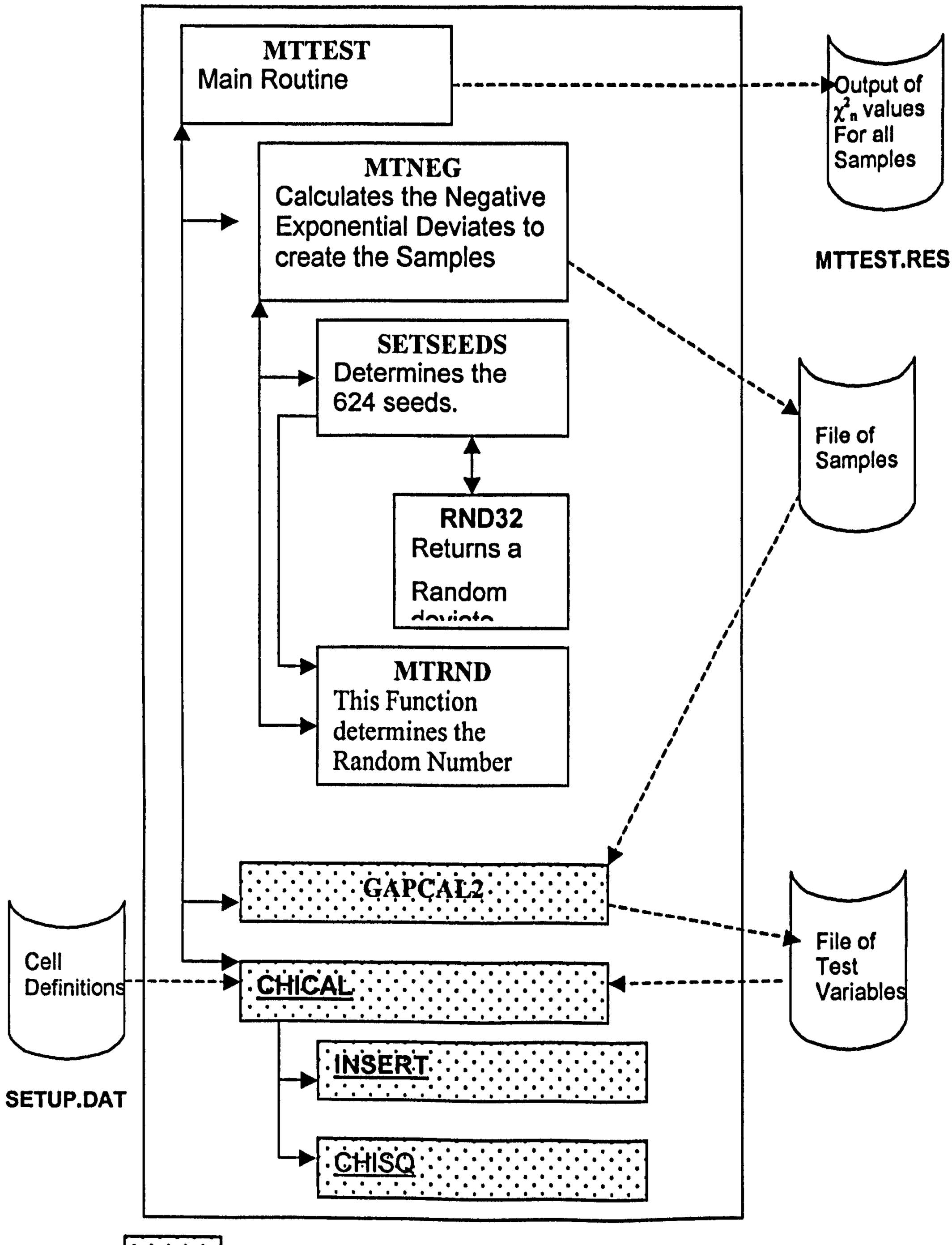


Diagram A1.4: Structure of Modules used in MTTEST

MTTEST

```
C MTTEST
c This program is to evaluate MT capability to produce
c satisfactory samples from a Negative Exponential     distribution
c
c Written by A,J.Warn 2002
c        Amended to read a file of seeds
c
c The seed range is not for the MT but to prime it
c
c It produces a file    MTtest.res
c
c the first record is the description of the MT used
c        the second is the number of cells used to calculate
c the chi squared value
c then follows pairs of records
c    the seed
c    10 chi square values for sequences of 1 - 10
c        (default is 10 var. max)
c if more than 20 it will write more than one record
c
c
      Real*8  A(250),AO(250,20),CHI(20)
      Real*8 Mult, Modulus, Offset
      integer*4 seed,aseed(1000),dseed
      integer*4 is, ntrial,max,ncata(20),iseed,iseed2,iseds
       Character*1 Drive
       Character*50 iname
       Character*30 fseed,name
c set file unit
            dseed=4
c  A block size of records written on the work file
        Nblock =10
c First Select media for files
        write(*,10)
 10   Format(' What is the drive for files?')
        read(*,20) drive
 20    Format(A1)
        open(10,file=Drive//':\setup.dat')
c read header
        read(10,30)ntrial,minent,max
 30      Format(3i10)
        write(*,35)ntrial,minent,max
 35     Format(' Data read from SETUP'/
     + ' Number of Sample Values',i10/
     + ' Min Entry in a cell    ',i10/
     + ' Maximum Number of values combined in test',i4)
        rewind 10
        close(10)
```

```
c New
      ifseed=0
c Is the seed values to be read from a csv file
      write(*,42)
 42   Format(' Are the seed values to be read from a file y or n')
      read(*,20)yes
      if((yes.ne.'y').and.(yes.ne.'Y')) go to 49
 43    write(*,44)
 44    Format(' How many seeds to be read from file')
      read(*,51)iseed2
      write(*,45)
45    Format(' Give full path of file with seed values'/
     + ' Including extension e.g. CSV')
      read(*,55)fseed
      ifseed=1
      open(dseed,file=fseed)
      read(dseed,51,end=46)(aseed(nums),nums=1,iseed2)
      go to 47
46    write(*,51) aseed(1)
      stop ' Too few seeds Program Stopped'
47    continue
      iseed=1
      iseeds=1
      go to 54
c
 49   continue
c        set seed range
         write(*,50)
 50   format(' What is the initial seed?')
      read(*,51)iseed
 51   Format(i15)
      write(*,52)
 52   format (' What is the final seed?')
      read(*,51)iseed2
      write(*,53)
 53   format (' What is the step?')
      read(*,51) iseeds
 54   continue
      open(11,file=drive//':MTtest.res')
      call ptime(time1)
      name=' Mersenne Twister'
      write(11,55)name,ntrial,max,iseed,iseed2,iseeds
 55   format(a30,5i10)
```

```
      do 300 is=iseed,iseed2,iseeds
         seed=is
         if(ifseed.gt.0)seed=aseed(is)
c Now create the random neg exp deviates
c with mean 1 on a file
         Call mtneg(seed,Ntrial,Nblock,Drive)
c Create the observations by combining
c the random neg exp deviates
         call gapc2(Drive,max)
c Calculate the sum of the percentage squared differences
c        (Chi square statistic)
c
       call chical(Drive,max,CHI,ncata)
c  if first record after header write number of catagories
c  Degrees of Freedom
         if (is.eq.iseed) write(11,60) (ncata(I),I=1,max)
 60    format(20i5)
c write seed
         write(11,70)seed
c write sum of percentage squared differences
 70    format(i15)
         write(11,100)  (CHI(I),I=1,max)
 100   Format(20f10.2)
 300   Continue
c Now tell how long it took!
          call ptime(time2)
         time=time2-time1
         if(time.lt.0)time=time+(24*60*60)
         ihr=int(time/3600)
         imins=int((time-ihr*3600)/60)
         fsec=time-ihr*3600-imins*60
         write(*,400)ihr,imins,fsec
400    Format(' Time taken',I3,' Hrs',I3,' Mins',f7.3,' Secs')
       Stop
      End
```

MTNEG

```
c
      Subroutine MTneg(seed,Ntrial,Nblock,Drive)
C This subroutine produces a file of Ntrial neg exp deviates
c with an initial seed of IS
c File is on drive Drive
c
c They are blocked in blocks of Nblock
      integer*4 ,ntrial
      Real*8 output(1000),v
       Integer*4 mt(0:623),seed,is
       Real*8 mtrnd
      Character*1 Drive
c
c create file
       is=seed
       call setseed(mt,is)
       open(7,FILE=Drive//':/cneg2.dat')
       rewind 7
c
        st=s
c
c Write header
       write(7,100)ntrial,nblock,s
 100   Format('Header',2i6,f20.0)
c
       nprod=0
 200   continue
    do 1000 nr=1,nblock
c  Determine a random number
          v=mtrnd(mt)
          nprod=nprod+1
c Calculate a value drawn from a Negative Exponential Distribution
          expval=-dlog(1D0-v)
          output(nr)=expval
          if(nprod.GE.ntrial) go to 2000
 1000   Continue
c
       write(7,1001)nblock,(output(i),i=1,nblock)
 1001  Format(I3,100E20.12)
       go to 200
 2000 continue
c write outlast block even if an incomplete block
       write(7,1001)nr,(output(i),i=1,nr)
       rewind 7
      Return
      End
```

SETSEED

```
      Subroutine setseed(array,seed)
      real*8 va(7),rnd32
      integer*4 array(0:623),a,c,seed
         a=69069
         c=0
      do 10 n=0,623
         v=rnd32(seed,a,c)
        array(n)=seed
 10   continue
      return
      end
```

RND32

```
c ***************************************************************************
c ********************************rnd32**************************************
c ***************************************************************************
c
c  This Function which is REAL*8 Returns a random variate between 1 and 0
c  It is a MCG with a Modulus of 2^32
c  It requires a multiplier a and if required an offset c
c  Both these values should be Integer*4
c  If there is no offset it should be set to zero
c                An initial Seed is required that should be an Integer*4
c  The seed will be changed after each call of the Function
c  The seed should not be used as its Bit pattern is used not its
c  value.  If printed a seed may appear as negative.
c
c by Takano H
      real*8 Function rnd32(seed,a,c)
      integer*4 seed,a,c
      real*8 bit
      bit=1D0
      if(c.NE.0)bit=0d0
      seed=iand(a*seed+c,-1)
      if(seed.lt.0) rnd32=(dfloat(seed)+2D0**32)/(2d0**32-bit)
      if(seed.gt.0) rnd32=(dfloat(seed))/(2d0**32-bit)
      if(seed.eq.0) rnd32=0
      return
      end
```

MTRND

```
c                                This Function mtrnd
c
c    This routine creates random Numbers based on paper
c    written by Matsumoto M and Nishimura T (1998),
c    also a FORTRAN version of program by Takano H, (1999)
c       amended by A J Warn
c
c    Array mt() has to be filled with seeds          before the first call
c    of this routine    (624 values) Integer*4 variables
c
      Real*8 Function mtrnd(mt)
      integer*4 Umask,Lmask,Tmaskb,Tmackc
      dimension mt(0:623)
      integer*4 mag01(0:1)
      mag01(0)=0
      mag01(1)=-1727483681
      Lmask= 2147483647
      Tmaskb=-1658038656
      Tmaskc=-272236544
      Umask=-2147483647
      Umask=Umask-1
c Microsoft Fortran does not allow Umask to be set directly to -2147483648
c
      do 1000 kk=0,226
         it=ior(iand(mt(kk),Umask),iand(mt(kk+1),Lmask))
         mt(kk)=ieor(ieor(mt(kk+397),ishft(it,-1)),mag01(iand(it,1)))
 1000   continue
        do 1100 kk=227,622
         it=ior(iand(mt(kk),Umask),iand(mt(kk+1),Lmask))
         mt(kk)=ieor(ieor(mt(kk-227),ishft(it,-1)),mag01(iand(it,1)))
 1100   Continue
        it=ior(iand(mt(623),Umask),iand(mt(0),Lmask))
        mt(623)=ieor(ieor(mt(396),ishft(it,-1)),mag01(iand(it,1)))
c
      mti=0
      it=mt(mti)
      mti=mti+1
      it=ieor(it,ishft(it,-11))
      it=ieor(it,iand(ishft(it,7),Tmaskb))
      it=ieor(it,iand(ishft(it,15),Tmaskc))
      it=ieor(it,ishft(it,-18))
c
      if(it.lt.0)mtrnd=(dfloat(it)+2.0D0**32)/(2.0D0**32-1.0D0)
      if(it.gt.0)mtrnd=dfloat(it)/(2.0D0**32-1.0D0)
      if(it.eq.0)mtrnd=0.0D0
c
      return
      end
```

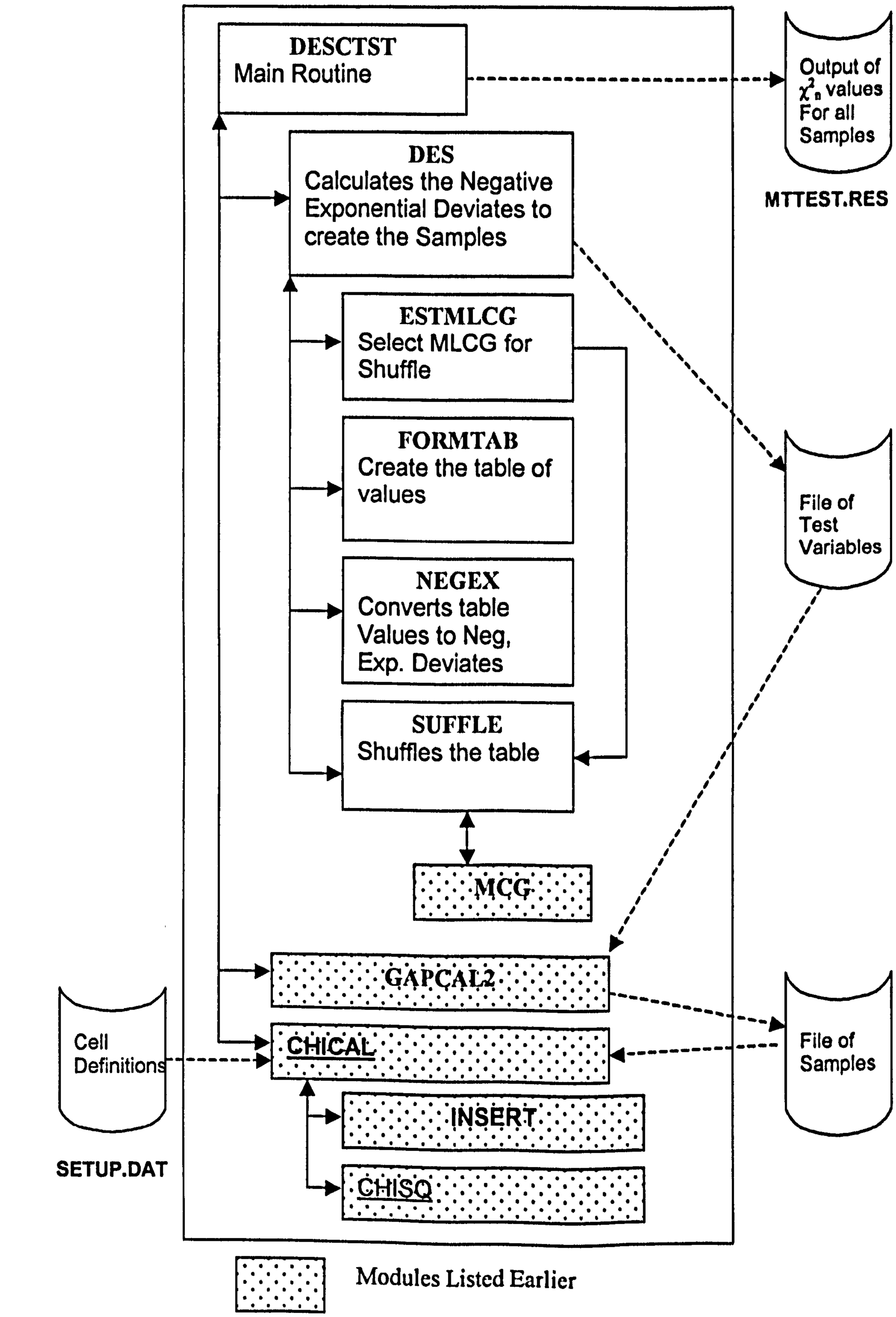


Diagram A1.5: Structure of Modules used in DESCTST

DESCTST

```
$Pagesize:50
C DESCTST Test
c This program is to evaluate Descriptive Sampling
c capability to produce       satisfactory samples
c from a Negative Exponential      distribution
c
c Written by A,J.Warn 2002
c
c
c It produces a file              des.res
c
c the first record is the description of the method used
c        the second is the number of cells used to calculate
c the chi squared value      then follows pairs of records
c    the set number
c    10 chi square values for sequences of 1 - 10
c    (default is 10 var. max)
c    if more than 20 it will write more than one record
c
       Real*8  A(250),AO(250,20),CHI(20)
       Real*8 seed,sn
       integer*4 is, ntrial,max,ncata(20),iseed,iseed2,iseeds
       integer*4 Nset
       Character*1 Drive
       Character*30 name
        name='Descriptive Sampling'
c  A block size of records written on the work file
        Nblock =10
c First Select media for files
       write(*,10)
 10    Format(' What is the drive for the work files?')
       read(*,20) drive
 20    Format(A1)
       open(10,file=Drive//':\setup.dat')
c Write header
       read(10,30)ntrial,minent,max
 30     Format(3i10)
       write(*,30)ntrial,minent,max
          rewind 10
          close(10)
```

```
c
c Here we set the number of values sorted
      write(*,35)
 35   Format(' How many values should be sorted?')
      read(*,36)Nset
 36   Format(i10)
c                                          set seeds
       write(*,40)
 40   format(' What is the initial seed?')
      read(*,50)iseed
 50   Format(i15)
      write(*,60)
 60   format (' What is the final seed?')
      read(*,50)iseed2
      write(*,70)
 70   format (' What is the step?')
      read(*,50) iseeds
      open(11,file=drive//':\r\des.res')
      call ptime(time1)
      do 300 is=iseed,iseed2,iseeds
          seed=is
c Now create the random neg exp deviates with mean 1 on a file
c so same RNG used for all seeds
           if(is.eq.iseed)iflag=0
           if(is.gt.iseed)iflag=1
           Call DES(Nset,Ntrial,Nblock,Drive,seed,iflag)
c Create the observations by combining the random neg exp deviates
           call gapc2(Drive,max)
c Calculate the sum of the percentage squared differences
c        (Chi square statistic)
c
        call chical(Drive,max,CHI,ncata)
c  if first record after header write number of catagories
c  Degrees of Freedom
           if(is.eq.iseed) write(11,75)name,ntrial,max,iseed,iseed2,iseeds
 75    format(a30,5i10)
          if (is.eq.iseed) write(11,80) (ncata(I),I=1,max)
 80    format(20i5)
c write seed
          write(11,90)is
c write sum of percentage squared differences
 90       format(i15)
          write(11,100)        (CHI(I),I=1,max)
 100      Format(20f10.2)
 300   Continue
```

```
c Now tell how long it took!
      call ptime(time2)
      time=time2-time1
      if(time.lt.0)time=time+(24*60*60)
      ihr=int(time/3600)
       imins=int((time-ihr*3600)/60)
       fsec=time-ihr*3600-imins*60
      write(*,400)ihr,imins,fsec
 400  Format(' Time taken',I3,' Hrs',I3,' Mins',f7.3,' Secs')
         Stop
         End
```

DES

```
 $pagesize:50
c
      Subroutine DES(Nset,Ntrial,Nblock,Drive,seed,iflag)
      Character*1 Drive
       real*8 array(10000),output(1000),seed
       integer*4 Ntrial,nprod,Nset,nsample
c Open File File Name is used by an older program
      open(7,File=Drive//':/cneg2.dat')
      rewind 7
      write(7,100) ntrial,nblock,seed
 100  Format('Header',2i6,f20.0)
c
      nprod=0
 200  continue
      call DESNEXP(array,Nset,seed,iflag)
      iflag=1
      nsample=0
 300  continue
      do 1000 nr=1,nblock
            nprod=nprod+1
             nsample=nsample+1
            output(nr)=array(nsample)
            if(nprod.ge.ntrial) go to 2000
            if(nsample.LT.Nset) go to 1000
           call DESNEXP(array,Nset,seed,iflag)
            nsample=0
 1000  Continue
      write(7,1001) nblock,(output(i),i=1,nblock)
 1001 Format(i3,100e20.12)
      go to 300
 2000 Continue
      write(7,1001) nr,(output(i),i=1,nr)
      rewind 7
      return
      end
```

DESNEX

```
      Subroutine DESNEXP(array,noreq,seed,iflag)
      Real*8 array(1),a,b,c,seed
      integer*4 noreq
      character*30 Descr
c This program computes a fixed number of negative exponential
c distribution using the method of adaptive sampling.
c
c Set RN if first time
        if(iflag.NE.0) go to 10
          call     EstMLCG(Descr,a,b,c)
 10     continue
c   Form Table
          call FormTab(array,noreq)
          call NegExp(array,noreq)
c   shuffle table
       call Shuffle(array,noreq,seed,a,b,c)
       return
       end
```

FORMTAB

```
Subroutine FormTab(array,noreq)
      Real*8 array(1),step,hstep
      integer*4 noreq,n
c
c First an array is created
      step=1D0/dfloat(noreq)
      hstep=step/2D0
      do 10 n=1,noreq
        array(n)= n*step-hstep
 10   continue
      return
      end
```

NEGEXP

```
      subroutine NegExp(array,noreq)
      Real*8 array(1),value
      Integer*4 noreq,n
      do 10 n=1,noreq
         value=array(n)
         array(n)=-dlog(1D0-value)
10    continue
      return
      end
```

ESTMLCG

```
      Subroutine EstMLCG(Descr,Mult,Modulus,offset)
      Character*30 Descr,D(7)
      Real*8 Mult,Modulus,offset,Am(7),ab(7)
c
      D(1)=' F&M Line 18 Knuth(1997)'
      Am(1)=62089911
      Ab(1)=2**31-1
      D(2)='Flying Line 17 Knuth(1997)'
      Am(2)=314159269
      Ab(2)=2**31-1
      D(3)='Lewis+ Line 19 Knuth(1997)'
      Am(3)=16807
      Ab(3)=2**31-1
      D(4)='L"Ecuyer Line 21 Knuth(1997)'
      Am(4)=40692
      Ab(4)=2**31-249
      D(5)='L. C. Killingbeck'
      Am(5)=2650845021.0D0
      Ab(5)=4294967296.0D0
      Number=5
c
      Write(*,10)
 10    Format('  Now select the Random Number Generator'/
     +'  To be used in the Shuffle'//)
       write(*,20)
20    Format(' Ref #',' Description  ')
      do 50 n=1,number
         write(*,30)n,D(n)
30       Format(i4,2x,a30)
50    continue
     do 100 n=1,10
         write(*,55)
55       Format('  Type the ref # to select ')
        read(*,60)nr
60     format(i6)
        if (nr.Lt.0) go to 100
        if (nr.gt.number) go to 100
        go to 200
100   continue
        Stop '  Incorrect Ref # being entered!'
200   Descr=D(nr)
      Mult=AM(nr)
      Modulus=Ab(nr)
      Offset=0
     Return
     End
```

SHUFFLE

```
c
      Subroutine Shuffle(array,noreq,seed,a,b,c)
c
c This subroutine shuffles array    whose size is noreq
c The randomness is provided by a MLCG
c MCG is a double precision version       of an MLCG
c the Modulus is b
c the Multiplier is a
c the offset or additive constant is c
c a seed is required whose value changes
c
      real*8 array(1),seed,a,b,c,temp
      integer*4 noreq,n
      do 100 n=1,noreq-1
               call MCG(seed,a,b,c)
               num=int((noreq-n)*seed/b)+n
               temp=array(n)
              array(n)=array(num)
              array(num)=temp
 100  Continue
      return
      end
```

SELECT

```
c This program selects the best sets of samples
c First it filters
c then it sorts
c
c It requires a file with the cut-off values
c It also requires a set of weights
          Character*1 Drive
          Character*30 data,name,cut,output
      Integer*4 ntrials,minent,nrec,nogood,ncell
      Integer*4 count,nread,ftop500,fcut,finput,n500
      Real*8 CHITOT,temp
c The program is using a default value of 10
c as the number of values and weights
c if this is to be change nval needs to be altered.
c if it is to be greater than 20
c the dimensions of the arrays need to be changed.
c
      real*8 range(20),chi(20),w(20)
      integer*4 ncata(20),num(20),aok(20)
c
c Define variables
c The following variables are defined assuming 500 best are being
c determined
      Real*8 qual(500),tseed(500)
      Integer Index(500)
c t is subject to both 20 and 500
          Real*8 t(500,20)
c
c
c Set the default values which would need to be changed with dimensions
          n500=500
          nval=10
c Set file numbers
          ftop500=7
          finput=9
          fcut=10
          w(1)=1
          w(2)=2
          w(3)=3.086
          w(4)=4.154
          w(5)=5.413
          w(6)=6.353
          w(7)=7.714
          w(8)=8.308
          w(9)=9.818
          w(10)=10.8
     write(*,10)
```

```
c       Request the name of the file with the cut off values
 10   Format(' Give the file name with the cut off values'/
     + ' The full path is required including the extension.')
          read(*,11) cut
c Open the file and read the cut off values
c There is a record per value
 11   Format(a30)
      open(fcut,file=cut)
      read(fcut,12,end=13)(range(i),i=1,nval)
 12   format(f10.4)
      go to 20
 13     Stop 'Select: The Cut off value file has too few values'
 20   Continue
c Request the name of the output file
      write(*,30)
 30   Format(' Give the file name for the output file'/
     +  'The full path is required including the extension.')
          read(*,11) output
c Request the name of the file with
c the data for the RNG to be analysed
         print '(a46)',
     +' What file contains the data for the RNG to be'
         print '(a41)',
     +' analysed? Add the extension (e.g.  .dat)     '
         read(*,11) data
c
c Open the output and input files
c Output file
      open(ftop500,file=output)
c Next the file containing the sums of percentage differences
      open(finput,file=data)
c Read the heading of the data
      read(finput,40)name,ntrial,max
 40    format(a30,2i10)
      read(finput,50) (ncata(l),l=1,max)
 50    format(20i5)
 60    continue
       read(finput,100,end=350)is
 100      format(i15)
       read(finput,110,end=300)(CHI(l),l=1,max)
          nread=nread+1
          nok=0
 110    Format(20f10.2)
```

```
      do 120 n=1,max
         if(CHI(n).gt.range(n)) go to 60
c  Calculate total of square values
         CHITOT=0
        do 120 i=1,max
           CHITOT=CHITOT+CHI(i)*w(i)
 120    continue
c  Add to table
       nrec=nrec+1
       tseed(nrec)=is
       index(nrec)=nrec
       qual(nrec)=CHITOT
       do 130 i=1,max
        t(nrec,i)=CHI(i)
 130    Continue
       if(nrec.lt.n500) go to 60
       if(nrec.eq.n500) go to 400
 300    Continue
        Stop ' Select Input File out of sequence'
 350    endmet=1
       write(*,360)nrec
 360    Format(' Select: End of input file reached'/
    + ' Number of sets selected was ',i4)
 400    Continue
c Sort the records (100)
c A simple sort as sequence likely to be random
        do 510 n=1,nrec-1
         do 500 m=n+1,nrec
           if (qual(n).lt.qual(m)) go to 500
           temp=qual(n)
           qual(n)=qual(m)
           qual(m)=temp
           itemp=tseed(n)
           tseed(n)=tseed(m)
           tseed(m)=itemp
           itemp=index(n)
           index(n)=index(m)
           index(m)=itemp
 500    Continue
 510   Continue
c
c Now the main loop          if more records
        if(endmet.gt.0) go to 2000
c   Read the next record
 600    Continue
```

```
        read(finput,100,end=2000)is
        read(finput,110,end=1000) (CHI(I),I=1,max)
c   Calculate total of square values
        CHITOT=0
        do 610 i=1,max
        CHITOT=CHITOT+CHI(i)*w(i)
 610     continue
c   if worse than the bottom record get another record
        if(CHITOT.gt.qual(nrec)) go to 600
c
c   if better
c The last record will be lost from the best nrec
c Its data is held at index position index(nrec)
         notei=index(nrec)
c replace data with new record
        do 620 k=1,max
        t(notei,k)=chi(k)
 620     Continue
c Next find where new record slots in
c   Move records down the table until slot for new record found
        do 700 i=nrec-1,1,-1
c Check if i+1 is the slot by examining the ith record
        if(CHITOT.gt.qual(i)) go to 800
c not the i+1th record move ith down (ith already made vacant)
            qual(i+1)=qual(i)
            tseed(i+1)=tseed(i)
            index(i+1)=index(i)
 700     continue
c if not found must be the top
        go to 900
 800     Continue
c replace ith+1 record (already made vacant)
            qual(i+1)=CHITOT
            tseed(i+1)=is
c   set index to where data was stored earlier
            index(i+1)=notei
c    Record inserted go to next record
     go to 600
 900     Continue
c New record must go to the top
            qual(1)=CHITOT
           tseed(1)=is
c set index
            index(1)=notei
c go back and read the next record
        go to 600
```

```
c
c Abnormal End of File
 1000    continue
         write(*,1010)is
 1010    format(' select: Record Missing for seed', F20.0/
     +' will continue processing')
         go to 600
c Normal End
 2000    continue
c At end of file write out best records
 2010    Format(a30,3f15.0,2i15)
          do 2050 k=1,nrec
            write(ftop500,2020)tseed(k),qual(k),(t(index(k),kk),kk=1,max)
 2020     format(f15.0,f15.2,20f10.2)
 2050    Continue
        Stop '  Run is complete'
        End
```

MARKOV

```
Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line
1         real*8 total,var,average,sq
2         real*8 util,proc,arrive
3         real*8 a(0:1000),b(0:1000),p(0:1000)
4         real*8 avea(10000),vara(10000)
5              real*8 sqa(10000),cuma(10000)
6  c On-line printer changed for file 2002
7  c A J Warn
8  c
9  C open output file
10          open(10,file='c:\markov.prn')
11  C  set processing time and inter-arrival
12   1        write(*,2)
13   2     Format(' What machine utilisation is to be used?'/
14      +          ' Must be less than 1')
15                read(*,3)util
16   3     format(f6.4)
17          if(util.gt.1) go to 1
18          theory=util*(1D0-5D-1*util)/(1D0-util)
19   4     write(*,5)
20   5     format(' What is the initial Queue Size'/
21      +          ' It must not be greater than 100')
22          read(*,6)nstart
23   6     format(i10)
24          if(nstart.gt.100) go to 4
25             ntop=nstart+1
26          Proc=100*util
27          arrive=100
28  C Set up Probability Matrix
29             p(0)=dexp(-util)
30           total=p(0)
31          do 10 n=1,999
32                p(n)=p(n-1)*(util)/n
33   10           total=total+p(n)
34          p(1000)=1-total
```

```
 Microsoft FORTRAN Optimizing Compiler Version 4.01
 Line#  Source Line
35  C set the initial queue
36            do 20 i=0,1000
37   20         a(i)=0
38          if(nstart.LE.1) a(0)=1
39          if(nstart.gt.1) a(nstart-1)=1
40         do 500    item=1,10000
41  c Calculate the probabilities just before next item
42                    total=0
43          do 110 n=0,999
44                     b(n)=0
45                     do 100 i=0,n
46                          b(n)=b(n)+a(n-i)*p(i)
47  100                 continue
48                    total=total+b(n)
49                    ntop=n
50  c
51                    if(total.GE.0.999999.and.b(n).le.1D-15) go to 200
52  110       continue
53  200       ntop=ntop+1
54                b(ntop)=1-total
55  C Adjust for the item just processed
56  C note if the queue was empty the next arrival
57  c will go direct to be served and start the cycle.
58  c Since we are determining the distribution of the queue
59  c size at the end of processing this is dead time and may
60  c be ignored. There is no distortion on the patterns of
61  c arrivals as they are exponential and there is no memory
62          a(0)=b(0)+b(1)
63          total=a(0)
64         do 300 n2=1,ntop-1
65              a(n2)=b(n2+1)
66   300        total=total+a(n2)
67  C a(ntop) should be zero
68           a(ntop)=1-total
69   500  continue
```

```
Microsoft FORTRAN Optimizing Compiler Version 4.01
Line#  Source Line

70   520  format(i8,4G15.6E3)
71                average=0
72                Do 600 n=1,ntop
73   600          average=average+n*b(n)
74                var=0
75                sq=0
76                do 800 n=0,ntop
77               var=var+b(n)*(n-average)**2
78                    sq=sq+b(n)*(n-theory)**2
79   800      continue
80                avea(item)=average
81                vara(item)=var
82                sqa(item)=sq
83   1500  continue
84          cuma(1)=avea(1)-theory
85         do 1510 i = 2,1000
86          cuma(i)=cuma(i-1)+avea(i)-theory
87   1510     continue
88         do 1520 i=1,10000
89          cuma(i)=cuma(i)/i + theory
90   1520     continue
91            write(10,2000)       util,nstart
93       1 ' of',f6.4,' Queue initialised at',i4/
94       2 ' part no  Average',8x,' Variance',5x,
95       3 'Stand.Error',3x,'Cum.Average')
96         do 3000 i=1,10000
97           write(10,520)i,avea(i),vara(i),sqa(i),cuma(i)
98   3000  Continue
99           stop
100           end
```

MISSING

```
Microsoft FORTRAN Optimizing Compiler Version 4.01
 Line#  Source Line
    1       integer*4 is,n,itop
    2         itop=2**31-1
    3         write(*,1)
    4   1      format(' Program Miss Started')
    5         do 20 n=1,itop
    6             is=n
    7         call rand(is,r)
    8         if(n.ne.is) go to 10
    9         write(*,1000)is
   10   1000   Format(i20)
   11          Stop ' Missing number'
   12   10     continue
   13         if(mod(n,10000).eq.0) write(*,2000)n
   14   2000   FORMAT(' Reached ',i20)
   15   20     continue
   16         stop 'No loop of 1'
   17         end
   18  c
   19         subroutine rand(is,r)
   20  c
   21  C This subroutine returns a random number r between 1 and 0
   22  c is is a seed which is used on the first time the subroutine
   23  c is called
   24  c it is returned with a value used in subsequent calls
   25  c          IS is Integer*4
   26             integer*4 is,ia,ib
   27          real*8 a,b,s
   28          a=314159269
   29          b=2**31-1
   30          s=is
   31          s=dmod((s*a)+1D0,b)
   32          is=s
   33          r=s/b
   34          if(r.eq.0) r=.0000000001
   35          return
   36          end

No errors detected
```

TAILMLCG

```
c  FOR NEGATIVE EXPONENTIAL DISTRIBUTIONS
c
c This program will give the exact number of samples values in the tail
c for a certain size of sample it is to aid the setting of minimum
c duration of simulation runs
c
c A J Warn  2001
c
      integer*4 size,count(10),is,tot(10)
      real*8 seed,v,expval,a,b,c
      character*30 name
      Call Give(name,a,b,c)
      Write(*,10)
 10   Format(' How big is the sample?')
      read(*,20)size
      write(*,30)
 20   Format(i10)
 30   Format(' What seed is to be tested?')
      read(*,20)is
     seed=is
      do 100 n=1,size
         call MCG(seed,a,b,c)
         v=seed/b
c Calculate a value drawn from a Negative Exponential Distribution
         expval=-dlog(1D0-v)
         I=expval
         if(i.lt.1) go to 100
         if(i.gt.10)i=10
           count(I)=count(I)+1
 100  Continue
      write(*,110)name,size,is
 110  Format(' Number of Values in Tail'//
     + ' This is for a Negative Exponential Distibution'/
     + ' For a MLGC ',a30/
     + ' Sample Size ',i10/' and seed',i12/
     + '       Tail Number of Entries'/)
         tot(10)=count(10)
        do 115 n=9,1,-1
          tot(n)=tot(n+1)+count(n)
 115   continue
 120  do 140 n=1,10
         write(*,130)n,tot(n)
 130     format(i10,i17)
 140  continue
      stop ' Completed. Run again for different values.'
       end
```

BAD100

```
$Pagesize:50
C This program selects the worst 100 seeds
c The output is a file bad100.res
c
c Written by A J Warn May 2002
c
c The file containing the Chi Squared Test is a Comma Separated
c file.  Using Microsoft FORTRAN Compiler the comma terminates the
c field and the next field starts following the comma.
c This is not in the ANSI standard
c
c Define variables
      Character*1 Drive
      Character*30 data,name
      Integer*4 ntrials,minent,nrec,nogood,ncell
      Integer n100,maxmax
      Real*8 w(20)
      Real*8     Sseed,eseed,sstep,CHITOT,temp
c The following variables are defined assuming 100 worst are being
c determined
      Real*8 qual(100),tseed(100)
      Integer Index(100),fbot100,finput,fchisq
c
c The following variables are dimensioned to the largest value
c of max 20, if a larger values is to be used the dimension would
c need to be increased.      Note t is subject to both conditions.
c Also maxmax will need to be changed.
      Real CHI(20)
      Real*8 t(100,20)
      Integer ncata(20)
c
c Set the default values which would need to be changed with dimensions
       n100=100
       maxmax=20
c Set file numbers
         fbot=7
         finput=9
         w(1)=1
         w(2)=2
         w(3)=3.086
         w(4)=4.154
         w(5)=5.413
         w(6)=6.353
         w(7)=7.714
         w(8)=8.308
         w(9)=9.818
         w(10)=10.8
```

```
c First Request drive containing the files
c All the files are assumed to be on the same drive
      print '(a41)',' What Drive contains the Chisquare values'
      print '(a28)',' and will hold output files?'
      read(*,10) Drive
 10    Format(A1)
c Request the name of the file with the data for the RNG to be analysed
      print '(a46)',' What file contains the data for the RNG to be'
      print '(a41)',' analysed? Add the extension (e.g.  .dat)     '
      read(*,20) data
 20   Format(a30)
c
c Open the files
c
c the file containing the sums of percentage differences
       open(finput,file=data)
c Then the File for the results
       open(fbot100,file=Drive//':\bad100.res')
c
c         Read the heading of the data
        read(finput,30)name,ntrial,max,isseed,ieseed,isstep
        sseed=isseed
        eseed=ieseed
        sstep=isstep
 30    format(a30,5i10)

         write(*,40)name,ntrial,max,sseed,eseed,sstep
 40    Format(///' Data to be analysed:'/
     +' Name of RNG ',a30/
     +' Number of Samples being tested',i10/
     +' Maximum of Combined Samples being analysed',i4/
     +' Seed Range:',/
     +'    starting    ',f20.0,/
     +'    finishing   ',f20.0,/
     +'    in steps    ',f20.0,/)
c Test that number of combined readings is in the range of program
        if(max.gt.maxmax)
     +  Stop ' bad100: maxmax value too small also check dimensions'
c Transfer information to the outputs
       write(fbot100,30)name,ntrial,max,isseed,ieseed,isstep
c Read the number of cells
       read(finput,50) (ncata(I),I=1,max)
 50    format(20i5)
c Read the first n100          acceptable record
c initialise counts
       nrec=0
       nogood=0
 90    Continue
```

```
            read(finput,100,end=200)is
 100      format(i15)
         read(finput,110,end=300) (CHI(I),I=1,max)
 110      Format(20f10.2)
          nrec=nrec+1
c   Calculate total of square values
         CHITOT=0
         do 120 i=1,max
         CHITOT=CHITOT+CHI(i)*w(i)
 120      continue
c   Add to table
         tseed(nrec)=is
         index(nrec)=nrec
         qual(nrec)=CHITOT
         do 130 i=1,max
          t(nrec,i)=CHI(i)
 130      Continue
            if(nrec.lt.n100) go to 90
         if(nrec.eq.n100) go to 400
 200      Continue
c if end of file and fewer than n100 stop with message
         write(*,340)data,is,nrec
 340      Format(' bad100: File ',A30/
     +' End of file too few seeds saved will terminate'
     +  /' Last seed ',i15
     +  /' Number saved ',I4)
            Stop
 300      Continue
          write(*,350) data,is
 350       Format(' bad100: File ',A30/
     +'Data out of sequence at seed',i10,/
     +'End of file too few seeds saved will terminate')
            Stop
 400      Continue
c         Sort the records (100)
c A simple sort as sequence likely to be random
```

```
      do 510 n=1,n100-1
        do 500 m=n+1,n100
          if (qual(n).lt.qual(m)) go to 500
          temp=qual(n)
          qual(m)=temp
          itemp=tseed(n)
          tseed(n)=tseed(m)
          tseed(m)=itemp
          itemp=index(n)
          index(n)=index(m)
          index(m)=itemp
 500      Continue
 510     Continue
c
c Now the main loop
c   Read the next record
 600     Continue
         read(finput,100,end=2000)is
         read(finput,110,end=1000) (CHI(I),I=1,max)
         CHITOT=0
         do 610 i=1,max
           CHITOT=CHITOT+CHI(i)*w(i)
 610     continue
c   if worse than the bottom record get another record
         if(CHITOT.lt.qual(n100)) go to 600
c
c   if better
c The last record will be lost from the worst n100
c Its data is held at index position index(n100)
          notei=index(n100)
c replace data with new record
         do 620 k=1,max
             t(notei,k)=chi(k)
 620  Continue
c Next find where new record slots in
c   Move records down the table until slot for new record found
                                                          do 700 i=n100-
1,1,-1
c Check if i+1 is the slot by examining the ith record
         if(CHITOT.lt.qual(i)) go to 800
c not the i+1th record move ith down (ith already made vacant)
            qual(i+1)=qual(i)
                  tseed(i+1)=tseed(i)
                        index(i+1)=index(i)
 700     continue
```

```
c if not found must be the top
        go to 900
 800     Continue
c replace ith+1 record (already made vacant)
            qual(i+1)=CHITOT
            tseed(i+1)=is
c   set index to where data was stored earlier
            index(i+1)=notei
c    Record inserted go to next record
     go to 600
 900     Continue
c New record must go to the top

          qual(1)=CHITOT
           tseed(1)=is
c set index
           index(1)=notei
c go back and read the next record
        go to 600
c
c Abnormal End of File
 1000    continue
        write(*,1010)is
 1010    format(' bad100: Record Missing for seed', F20.0/
    +' will continue processing')
c        Normal End
 2000    continue
c At end of file write out worst n100
 2010    Format(a30,3f15.0,2i15)
        do 2050 k=1,n100
         write(fbot100,2020)tseed(k),qual(k),(t(index(k),kk),kk=1,max)
 2020                                        format(f15.0,f15.2,20f10.2)
 2050    Continue
        Stop
        End
```

READRAN2

```
$PAGESIZE:50
c This is to read file from Rand.org
c to enable Chi-Square Test
c must be run for each file and for number of bytes ignored
c Amended to Input integer*4 June 2002
     Character*1 Drive,Alpha,fileNo
         character*30 name
     Real*8 outp,min,max
```

```
      integer*4 input
      integer*4 count(100),nread,n
          print '(a44)',' What number of the random digit file 1-4 ? '
      read(*,5)infile
5                                    format(i1)
      if(infile.gt.4) stop ' Try again 1-4'
       fileNo=Char(48+infile)
      name='D:\10megs.'//fileNo
      print '(a21)',' How many shoves 0-3 '
      read(*,5)nsh
      if(nsh.gt.3) stop ' try again 0-3 '
          Open(8,file=name,form='binary')
       Open(9,file='C:\counts.res')
       if(nsh.eq.0) go to 7
       do 6 n=1,nsh
                    read(8) Alpha
6      continue
7       nread=0
         do 10 n=1,10000000
            read(8,end=15)input
         if(input.lt.min)min=input
         if(max.lt.input)max=input
         nread=nread+1
c the most negative value is hex 80000000
c     this is seen as -2147483648
c the most positive value is hex 7FFFFFFF
c     this is seen as  2147483647
c since we do not wish to have zero or one
c
          outp=(input+2.147483649D09)/(4.294967297D09)
          index=1+int(outp*100)
          count(index)=count(index)+1
 10    Continue
       print '(a25)',' End of file not reached '
 15    Continue
        write(9,50)(count(i),i=1,100),nread
 50     format(101i12)
       write(*,40)nread
40    format('  Number of records read ',i14)
100   format(i12,f10.6)
      write(*,150)min,max
150   format(2f20.1)
       close(8)
       stop 'Normal ending'
       end
```

APPENDIX 2: THE RANDOM NUMBER GENERATOR RANDU

The RNG RANDU was included by IBM in its Scientific Subroutines for the IBM360 in the 1960s. (IBM 1970)

The use of the spectral test indicates as Knuth (1998) states "the generator fails most three-dimensional criteria for randomness and should never had been used." Even if one questions the need to meet the three-dimensional criteria, it is easy to find a generator that does and that has no disadvantage to RANDU.

The definition of RANDU is

$$R_{n+1} = (65539R_n) \bmod 2^{31}$$

The seed R_0 should be odd.

It can easily be shown by induction that if R_0 is odd then since 65539 is odd and 2^{31} is even, R_n must be odd for all positive n.

If one considers all possible seeds less than 2^{31} then a number of cycles of different sizes are obtained. To simplify description each cycle may be defined by its smallest member. If this is done table App2.1, which follows, is obtained.

From the table it may be seen that there are a total of 60 cycles. It may be seen there are two equal size cycles of odd numbers. All odd numbers (less than 2^{31}) are included in the two cycles. One of these two cycles has 1 as its smallest member the other has 5 as its smallest member.

There are 58 cycles of even numbers. The smallest consisting of one value, 1,073,741,824. There are many small cycles of even numbers. It is clear that the advice not to have an even seed is well founded.

TABLE APP2.1: FULL SET OF CYCLE LENGTHS OF RANDU WITH DIFFERENT SEEDS

Lowest Seed	Cycle Length	Lowest Seed	Cycle Length
1	536,870,912	5	536,870,912
2	268,435,456	10	268,435,456
4	134,217,728	20	134,217,728
8	67,108,864	40	67,108,864
16	33,554,432	80	33,554,432
32	16,777,216	160	16,777,216
64	8,388,608	320	8,388,608
128	4,194,304	640	4,194,304
256	2,097,152	1,280	2,097,152
512	1,048,576	2,560	1,048,576
1,024	524,288	5,120	524,288
2,048	262,144	10,240	262,144
4,096	131,072	20,480	131,072
8,192	65,536	40,960	65,536
16,384	32,768	81,920	32,768
32,768	16,384	163,840	16,384
65,536	8,192	327,680	8,192
131,072	4,096	655,360	4,096
262,144	2,048	1,310,720	2,048
524,288	1,024	2,621,440	1,024
1,048,576	512	5,242,880	512
2,097,152	256	10,485,760	256
4,194,304	128	20,971,520	128
8,388,608	64	41,943,040	64
16,777,216	32	83,886,080	32
33,554,432	16	167,772,160	16
67,108,864	8	335,544,320	8
134,217,728	4	671,088,640	4
268,435,456	2	1,342,177,280	2
536,870,912	2		
1,073,741,824	1		

APPENDIX 3: TABLES OF SELECTED SEEDS FOR CREATING SAMPLES OF 1000 SAMPLE VALUES

TABLE APP3.1: FISHMAN AND MOORE MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	52260	1430
2	56242	1431
3	6050	1463
4	52358	1470
5	21204	1482
6	1536	1497
7	15474	1497
8	23926	1506
9	27836	1512
10	37846	1536
11	12261	1542
12	14211	1547
13	35459	1549
14	52375	1555
15	58334	1557
16	11365	1557
17	5601	1562
18	29330	1562
19	9866	1569
20	2111	1576
21	39333	1579
22	46803	1586
23	54871	1589
24	32279	1589
25	59749	1593
26	43880	1593
27	42242	1599
28	51386	1601
29	34475	1601
30	1139	1604
31	14679	1606
32	28651	1608
33	22003	1609
34	34863	1610
35	6721	1612
36	34307	1612
37	44237	1614
38	3121	1615
39	21704	1619
40	23291	1620
41	49765	1621
42	24024	1621
43	30200	1623
44	7872	1623
45	25901	1624
46	45097	1625
47	18596	1626
48	20455	1628
49	57479	1636
50	4696	1636
51	37291	1637
52	37127	1637
53	47618	1640
54	20720	1642
55	37313	1642
56	28746	1649
57	38953	1651
58	55042	1653
59	45748	1657
60	12431	1659
61	42173	1661
62	36478	1664
63	32398	1664
64	49990	1664
65	3605	1665
66	37464	1666
67	33882	1666
68	46720	1666
69	36051	1668
70	44302	1671
71	45665	1674
72	9973	1676
73	13517	1680
74	2682	1680
75	46390	1680
76	31041	1681
77	49617	1681
78	47466	1682
79	39766	1683
80	34969	1684
81	59789	1686
82	224	1688
83	13068	1688
84	36982	1693
85	22659	1693
86	57611	1695
87	40603	1696
88	58072	1696
89	41497	1696
90	31037	1698
91	9840	1702
92	18938	1706
93	50458	1707
94	54047	1707
95	33586	1712
96	52636	1712
97	14149	1714
98	8978	1715
99	43872	1719
100	1562	1723
101	24222	1724
102	36002	1725
103	24081	1727
104	36106	1737
105	49812	1741
106	42939	1741
107	45590	1742
108	42647	1743
109	32390	1749
110	38060	1751
111	52341	1754
112	22651	1755
113	59159	1758
114	54710	1759
115	29773	1759
116	20978	1764
117	24473	1774
118	41514	1776
119	32967	1777
120	3914	1783
121	2695	1796
122	48807	1803
123	115	1829
124	15682	1834

TABLE APP3.2: L'ECUYER MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	33180	1448
2	7498	1484
3	22541	1504
4	31503	1508
5	18282	1514
6	1409	1515
7	52987	1517
8	24127	1518
9	21998	1533
10	10599	1534
11	34895	1544
12	55526	1552
13	31422	1553
14	29856	1554
15	22300	1566
16	16358	1566
17	53337	1567
18	32678	1569
19	29063	1573
20	37822	1575
21	17801	1578
22	13469	1580
23	6736	1584
24	37291	1585
25	4238	1585
26	10762	1585
27	56228	1590
28	47682	1593
29	59304	1595
30	10587	1597
31	36537	1597
32	25374	1598
33	6052	1603
34	40903	1605
35	24784	1607
36	20960	1610
37	37665	1615
38	33451	1616
39	32662	1616
40	18661	1617
41	21484	1620
42	16395	1624
43	55229	1625
44	53949	1629
45	40441	1630
46	18590	1631
47	37294	1632
48	47351	1635
49	13050	1636
50	40507	1639
51	35629	1640
52	40743	1641
53	1604	1646
54	5216	1648
55	55188	1651
56	28996	1653
57	39390	1657
58	59401	1659
59	32451	1659
60	37604	1660
61	22766	1661
62	10977	1662
63	9118	1662
64	37946	1667
65	17720	1669
66	57263	1669
67	29703	1671
68	34465	1672
69	23093	1672
70	57556	1673
71	14157	1674
72	3316	1680
73	43587	1680
74	23402	1682
75	52520	1684
76	22734	1687
77	15803	1688
78	22645	1689
79	18786	1690
80	56258	1695
81	22821	1697
82	38831	1700
83	28668	1701
84	17914	1701
85	53495	1704
86	17028	1707
87	16983	1707
88	22572	1712
89	21045	1714
90	44949	1714
91	28588	1715
92	10877	1716
93	43811	1720
94	41790	1721
95	6018	1721
96	26041	1725
97	24023	1726
98	5436	1729
99	40015	1731
100	3015	1731
101	6073	1735
102	10290	1738
103	39318	1744
104	59914	1753
105	12228	1754
106	25269	1754
107	27098	1762
108	29736	1766
109	40591	1773
110	32446	1779
111	46590	1780
112	49091	1783
113	55485	1800
114	45696	1815
115	34947	1822

TABLE APP3.3: LEWIS ET AL. MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	36670	1430
2	53224	1449
3	43532	1482
4	54490	1527
5	54338	1535
6	25450	1537
7	5695	1537
8	40212	1552
9	2738	1561
10	40048	1565
11	49786	1569
12	10443	1573
13	18943	1573
14	44132	1576
15	59880	1580
16	45376	1585
17	15502	1585
18	33657	1588
19	6214	1590
20	23851	1591
21	56752	1595
22	14079	1596
23	13531	1598
24	59393	1598
25	38123	1601
26	2201	1603
27	34526	1603
28	31259	1606
29	9779	1607
30	21950	1609
31	2377	1611
32	51664	1614
33	33015	1614
34	54331	1614
35	47097	1619
36	52166	1619
37	53419	1623
38	46940	1624
39	10195	1631
40	52405	1633
41	58714	1634
42	28780	1634
43	9213	1637
44	57691	1637
45	46332	1638
46	5448	1641
47	19515	1641
48	424	1642
49	55440	1642
50	30905	1643
51	59110	1649
52	18255	1650
53	22942	1651
54	19644	1655
55	11853	1656
56	35206	1659
57	41313	1661
58	21744	1661
59	26394	1668
60	55828	1670
61	59164	1670
62	35299	1672
63	16855	1673
64	5711	1683
65	33272	1684
66	41748	1687
67	42389	1687
68	43948	1692
69	34257	1693
70	31986	1697
71	378	1709
72	14346	1710
73	1725	1711
74	43779	1712
75	35897	1712
76	3671	1712
77	18462	1716
78	21644	1716
79	7922	1716
80	55004	1718
81	44341	1719
82	47977	1723
83	6528	1724
84	29925	1724
85	34580	1724
86	53505	1726
87	57276	1726
88	45178	1727
89	21964	1728
90	44664	1729
91	16312	1729
92	53439	1733
93	16206	1733
94	26870	1734
95	2547	1739
96	37484	1745
97	41061	1745
98	10830	1749
99	28456	1750
100	51488	1752
101	52584	1757
102	23708	1764
103	9211	1768
104	43774	1778
105	48704	1800
106	26557	1807
107	27107	1817

TABLE APP3.4: "FLYING" MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	38381	1442
2	8528	1479
3	57073	1499
4	30023	1544
5	34518	1548
6	24869	1548
7	25712	1548
8	23182	1551
9	40753	1560
10	26932	1561
11	30298	1565
12	38291	1581
13	54377	1584
14	39512	1584
15	1640	1585
16	41815	1588
17	48322	1589
18	28943	1591
19	58023	1591
20	14724	1597
21	53725	1597
22	3901	1598
23	14421	1600
24	25873	1602
25	46766	1611
26	5110	1611
27	5220	1612
28	27421	1612
29	24506	1614
30	1955	1614
31	39824	1614
32	55130	1616
33	38792	1616
34	39665	1619
35	28397	1619
36	25613	1621
37	17839	1622
38	32780	1622
39	37509	1626
40	44179	1626
41	54929	1631
42	5545	1636
43	15351	1637
44	9630	1638
45	30683	1640
46	51140	1641
47	56463	1645
48	203	1646
49	23358	1650
50	24019	1650
51	156	1651
52	57455	1652
53	16806	1653
54	8889	1655
55	14350	1656
56	25010	1656
57	40528	1657
58	36118	1659
59	21266	1659
60	12593	1661
61	13558	1662
62	13763	1664
63	3806	1665
64	40202	1668
65	27419	1670
66	39759	1672
67	28831	1672
68	31568	1674
69	39085	1677
70	9969	1680
71	28104	1680
72	11238	1681
73	31918	1684
74	6265	1684
75	31216	1685
76	10219	1686
77	51461	1688
78	44480	1690
79	39680	1690
80	38868	1693
81	39280	1693
82	30247	1695
83	2254	1699
84	53201	1701
85	39398	1701
86	24089	1702
87	51148	1703
88	22315	1704
89	46075	1705
90	58637	1708
91	24912	1708
92	41205	1709
93	12119	1712
94	31013	1713
95	57452	1715
96	10422	1716
97	55102	1719
98	27540	1720
99	16260	1722
100	26112	1724
101	46819	1728
102	3696	1731
103	52215	1734
104	54093	1734
105	55538	1737
106	608	1746
107	47343	1747
108	35372	1757
109	25920	1764
110	24540	1781
111	51252	1790
112	29026	1792
113	1120	1825
114	55655	1832

TABLE APP3.5: KILLINGBECK MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	36120	1447
2	37336	1470
3	49665	1476
4	42759	1501
5	27780	1503
6	15172	1520
7	24608	1525
8	41083	1536
9	15708	1539
10	54929	1548
11	47710	1552
12	46714	1553
13	50954	1555
14	3702	1564
15	39130	1565
16	23305	1571
17	36516	1572
18	39757	1577
19	12633	1578
20	41909	1578
21	52300	1583
22	6382	1596
23	57000	1600
24	2133	1605
25	50358	1606
26	27017	1617
27	56705	1619
28	23266	1622
29	31678	1623
30	37296	1627
31	43342	1627
32	58949	1628
33	11232	1629
34	44302	1631
35	56740	1631
36	51066	1631
37	42320	1633
38	11877	1634
39	15675	1635
40	48169	1637
41	29306	1640
42	43870	1642
43	50535	1644
44	15387	1644
45	48819	1645
46	56192	1648
47	25536	1651
48	56100	1654
49	9883	1654
50	12410	1655
51	12113	1656
52	2283	1658
53	10187	1658
54	1715	1659
55	40448	1660
56	5817	1660
57	42339	1661
58	43129	1662
59	35944	1663
60	57891	1664
61	26227	1665
62	17022	1666
63	28428	1667
64	11009	1667
65	48380	1669
66	31528	1672
67	17041	1675
68	29116	1675
69	36394	1677
70	20634	1678
71	48214	1679
72	43764	1681
73	9816	1683
74	40021	1684
75	33506	1685
76	19346	1686
77	42684	1688
78	20300	1689
79	18845	1691
80	55217	1695
81	50523	1695
82	965	1696
83	48204	1699
84	18499	1700
85	16031	1700
86	11465	1700
87	49098	1701
88	51044	1704
89	41093	1706
90	44806	1707
91	13805	1707
92	49792	1709
93	36908	1710
94	22077	1727
95	59445	1729
96	34359	1731
97	6960	1732
98	20664	1733
99	19053	1735
100	16565	1735
101	13110	1742
102	50476	1745
103	3978	1748
104	30107	1749
105	57214	1753
106	59788	1764
107	8578	1765
108	34172	1777
109	21773	1812
110	54058	1820

TABLE APP3.6: MLCG WITH A MODULUS OF 262139 (WHOLE CYCLE)

Modulus: 262139 Multiplier: 92717

Ranking	Seed	Quality Measure
1	119066	1376.3
2	170849	1459.7
3	56892	1463.3
4	147654	1467.4
5	42402	1471.6
6	84468	1484.1
7	217753	1484.5
8	106371	1486.3
9	209964	1494.9
10	258626	1495.3
11	143803	1500.3
12	138240	1501.8
13	40669	1502.3
14	148251	1503.9
15	147949	1516.1
16	216969	1516.3
17	86266	1520.0
18	154668	1527.3
19	29005	1530.7
20	137358	1535.2
21	17328	1537.0
22	44587	1544.7
23	125678	1549.9
24	53004	1551.3
25	103857	1559.0
26	99094	1566.2
27	107654	1566.6
28	169105	1570.0
29	250622	1570.2
30	55171	1572.7
31	133489	1574.9
32	2051	1578.0
33	249427	1578.4
34	234950	1581.6
35	163070	1582.4
36	180914	1582.4
37	100641	1589.6
38	128940	1591.1
39	223632	1593.3
40	80374	1605.6
41	68508	1606.7
42	178963	1609.9
43	183382	1610.4
44	43499	1613.5
45	30903	1615.4
46	23226	1615.4
47	187765	1615.8
48	67299	1617.2
49	79783	1618.9
50	96778	1622.5
51	55974	1623.7
52	14103	1623.8
53	189877	1625.7
54	24434	1626.3
55	221042	1628.5
56	61520	1630.3
57	148817	1631.4
58	983	1631.5
59	227010	1633.0
60	243615	1634.2
61	222804	1637.6
62	73874	1637.9
63	150285	1641.0
64	205441	1642.1
65	124398	1644.4
66	3088	1645.2
67	260083	1646.9
68	44540	1647.5
69	43221	1649.8
70	74523	1655.4
71	197469	1657.7
72	104515	1658.0
73	21890	1658.4
74	78301	1658.6
75	10863	1662.6
76	45500	1666.2
77	226048	1666.9
78	102309	1677.8
79	220199	1684.6
80	238418	1690.4
81	247872	1695.2
82	52191	1696.5
83	167173	1697.4
84	12979	1698.9
85	37406	1706.2
86	262016	1710.9
87	39381	1712.6
88	114414	1713.8
89	184068	1730.1
90	67409	1738.3
91	111470	1741.4
92	38820	1750.2
93	106940	1753.7
94	159011	1755.1
95	244914	1773.1
96	215741	1791.7

TABLE APP3.7: MLCG WITH A MODULUS OF 524287 (WHOLE CYCLE)

Modulus: 524287 Multiplier: 283741

Ranking	Seed	Quality Measure
1	99311	1436.5
2	345709	1440.0
3	217571	1453.3
4	136932	1456.4
5	167278	1462.6
6	106693	1469.4
7	419174	1473.1
8	401173	1477.1
9	22131	1480.2
10	238092	1480.8
11	230751	1481.1
12	278551	1485.2
13	475842	1486.3
14	276519	1486.9
15	175135	1490.8
16	9776	1490.8
17	499915	1492.7
18	286079	1493.6
19	364519	1496.6
20	332820	1498.1
21	473595	1499.5
22	475035	1499.7
23	160583	1500.9
24	192867	1501.3
25	456713	1502.2
26	282544	1503.4
27	280699	1505.3
28	76202	1507.1
29	35452	1508.2
30	66947	1511.3
31	461624	1512.1
32	353077	1512.8
33	462276	1513.8
34	356439	1515.4
35	234321	1515.6
36	521672	1520.9
37	288080	1521.0
38	289890	1522.3
39	471852	1522.6
40	154036	1524.4

Ranking	Seed	Quality Measure
41	292668	1524.6
42	435287	1524.9
43	517588	1526.3
44	196103	1530.3
45	196517	1532.9
46	436807	1533.3
47	116885	1534.5
48	264928	1535.7
49	300164	1536.0
50	279758	1537.8
51	99304	1538.1
52	103985	1538.3
53	385192	1540.0
54	250761	1546.2
55	484202	1547.1
56	235383	1547.5
57	220816	1548.4
58	462051	1549.4
59	196988	1550.5
60	356425	1550.7
61	297530	1551.1
62	110717	1551.5
63	386100	1552.6
64	398340	1552.7
65	69135	1556.0
66	225128	1558.0
67	454916	1558.6
68	426163	1560.6
69	501041	1561.9
70	217525	1562.5
71	126688	1562.7
72	412319	1563.5
73	504437	1564.6
74	378899	1567.9
75	170943	1571.0
76	28427	1572.5
77	435932	1574.2
78	328998	1577.6
79	40958	1578.4
80	351138	1578.4

Ranking	Seed	Quality Measure
81	170240	1580.3
82	280628	1582.2
83	357692	1582.2
84	501490	1582.7
85	336038	1583.2
86	459829	1584.2
87	193327	1585.6
88	510474	1585.9
89	519302	1586.2
90	371550	1588.5
91	75552	1588.6
92	144809	1589.0
93	385309	1589.4
94	250100	1591.9
95	259860	1594.1
96	426797	1599.6
97	488231	1599.7
98	286995	1599.8
99	174474	1600.8
100	514065	1600.8
101	33972	1601.3
102	293836	1602.4
103	98228	1602.8
104	247217	1603.2
105	379676	1604.5
106	487023	1607.2
107	195830	1607.7
108	411554	1608.4
109	382854	1610.2
110	101051	1612.3
111	331585	1614.5
112	140070	1614.7
113	265893	1616.4
114	518481	1619.2
115	42433	1619.3
116	20840	1620.1
117	231703	1620.2
118	56963	1622.7
119	58865	1623.8
120	457841	1623.9

Table APP3.7 (Continued): MLCG with a Modulus of 524287 (whole cycle)

Modulus: 524287 Multiplier: 283741

Ranking	Seed	Quality Measure
121	47498	1628.0
122	190090	1631.2
123	378416	1633.4
124	65565	1633.5
125	89346	1636.1
126	465538	1637.7
127	435125	1637.9
128	55726	1638.3
129	482327	1639.9
130	222227	1640.4
131	366651	1640.8
132	45834	1642.0
133	179504	1643.7
134	435076	1645.7
135	25247	1646.1
136	86890	1646.2
137	329191	1647.5
138	347771	1648.1
139	384855	1648.7
140	424722	1649.2
141	393803	1650.6
142	255452	1651.0
143	437397	1653.1
144	39633	1653.6
145	3003	1656.8
146	91433	1658.5
147	6369	1658.6
148	204144	1659.6
149	171262	1661.8
150	350127	1665.9

Ranking	Seed	Quality Measure
151	247408	1669.6
152	472285	1672.5
153	500530	1672.5
154	257712	1674.4
155	313015	1676.0
156	261072	1679.8
157	291695	1685.7
158	308204	1690.8
159	409192	1692.2
160	191983	1693.9
161	169782	1694.9
162	44091	1695.1
163	97276	1705.7
164	16030	1707.3
165	70473	1709.8
166	236114	1714.3
167	62047	1716.4
168	103989	1717.6
169	498632	1721.3
170	460243	1725.3
171	36026	1725.5
172	506562	1726.1
173	140365	1729.1
174	478239	1730.9
175	283129	1737.0
176	421938	1744.0
177	16036	1745.8
178	162905	1746.6
179	96755	1752.7
180	471525	1756.5

Ranking	Seed	Quality Measure
181	259136	1761.9
182	447325	1762.7
183	322957	1765.0
184	134887	1774.6
185	302185	1804.1
186	114415	1811.5
187	97844	1822.1

TABLE APP3.8: DESCRIPTIVE SAMPLING: FISHMAN AND MOORE MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	46433	1231
2	11964	1232
3	5084	1241
4	1351	1253
5	7473	1258
6	22130	1260
7	31035	1269
8	19786	1270
9	32497	1277
10	53515	1277
11	5439	1278
12	45106	1280
13	50740	1283
14	35261	1286
15	48983	1290
16	54944	1292
17	38394	1294
18	3047	1307
19	18602	1309
20	49342	1311
21	54926	1314
22	26262	1316
23	15956	1322
24	25751	1322
25	3133	1324
26	57254	1326
27	26807	1327
28	45159	1329
29	48591	1330
30	28353	1331
31	7275	1331
32	1123	1334
33	36412	1336
34	49581	1337
35	22024	1339
36	39021	1339
37	24500	1339
38	43528	1340
39	17081	1342
40	20243	1342

Rank	Seed	Quality Measure
41	9693	1343
42	18350	1345
43	9350	1345
44	28814	1346
45	17621	1349
46	38051	1349
47	21702	1350
48	34917	1350
49	7736	1350
50	54706	1350
51	565	1352
52	26735	1352
53	35459	1352
54	13805	1354
55	59108	1356
56	20285	1357
57	23508	1357
58	37631	1357
59	52757	1357
60	8215	1360
61	6890	1361
62	30049	1362
63	10652	1364
64	15641	1364
65	2068	1364
66	47834	1364
67	42627	1365
68	10993	1366
69	88	1368
70	8282	1368
71	10680	1369
72	46910	1369
73	25929	1370
74	44876	1371
75	45525	1371
76	11130	1373
77	37325	1373
78	11303	1375
79	23047	1376
80	44937	1376

Rank	Seed	Quality Measure
81	53920	1378
82	54811	1380
83	33263	1382
84	13625	1383
85	10425	1384
86	24098	1385
87	2190	1385
88	20176	1385
89	41160	1386
90	32714	1387
91	30213	1387
92	30526	1389
93	30644	1389
94	27057	1390
95	51811	1390
96	10879	1390
97	26004	1390
98	43045	1390
99	35679	1390
100	57948	1391
101	53557	1391
102	25461	1392
103	39999	1392
104	19496	1394
105	44044	1396
106	40869	1397
107	58884	1399
108	44491	1399
109	29737	1402
110	1081	1402
111	4809	1402
112	1818	1404
113	21500	1405
114	10111	1406
115	10288	1407
116	57765	1408
117	28840	1408
118	24574	1409
119	3851	1409
120	12726	1409

TABLE APP3.9: DESCRIPTIVE SAMPLING: L'ECUYER MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	43503	1221
2	39710	1246
3	3745	1249
4	52580	1250
5	19743	1250
6	14228	1254
7	25227	1270
8	50040	1272
9	4142	1272
10	8282	1273
11	10732	1276
12	41658	1282
13	30017	1283
14	31816	1288
15	35213	1301
16	18514	1306
17	23033	1308
18	36088	1310
19	23563	1311
20	14243	1312
21	42069	1315
22	41047	1316
23	30192	1318
24	5531	1318
25	5889	1319
26	50811	1319
27	42662	1321
28	50866	1323
29	5807	1323
30	8486	1324
31	19426	1325
32	54996	1326
33	55831	1328
34	41737	1329
35	24401	1330
36	28047	1330
37	3124	1331
38	10578	1336
39	7179	1337
40	26691	1337

Rank	Seed	Quality Measure
41	21214	1337
42	19712	1337
43	5293	1341
44	9615	1342
45	31649	1342
46	56023	1343
47	42361	1343
48	45334	1343
49	27961	1345
50	21437	1345
51	5201	1347
52	12485	1348
53	58632	1349
54	9284	1350
55	45829	1351
56	48998	1351
57	7378	1354
58	43292	1354
59	23350	1354
60	29137	1354
61	8039	1355
62	1335	1356
63	55527	1357
64	11627	1358
65	6498	1358
66	6885	1359
67	35709	1361
68	31687	1362
69	47652	1362
70	39310	1362
71	33674	1363
72	7420	1364
73	53395	1369
74	58328	1370
75	31486	1371
76	5755	1372
77	54181	1373
78	20517	1373
79	42356	1373
80	56907	1374

Rank	Seed	Quality Measure
81	16949	1374
82	1373	1374
83	33204	1374
84	35370	1375
85	47840	1377
86	43810	1378
87	6411	1378
88	10815	1380
89	57571	1382
90	51600	1382
91	1639	1383
92	48513	1384
93	1313	1386
94	42248	1387
95	51502	1387
96	19522	1387
97	22865	1388
98	35537	1388
99	7744	1388
100	22650	1389
101	43394	1391
102	26503	1391
103	5868	1392
104	39393	1392
105	26268	1393
106	33158	1393
107	58554	1394
108	27471	1394
109	3217	1395
110	39509	1395
111	27333	1396
112	52273	1397
113	5617	1397
114	49484	1398
115	19867	1398
116	35127	1398
117	25473	1399
118	644	1399
119	45238	1399
120	20851	1400

TABLE APP3.10: DESCRIPTIVE SAMPLING: KILLINGBECK MLCG (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	23599	1172
2	17971	1239
3	4690	1247
4	8396	1261
5	3601	1263
6	31493	1264
7	44010	1285
8	22646	1289
9	59824	1293
10	50199	1296
11	15690	1296
12	50586	1296
13	46160	1309
14	46905	1310
15	43253	1311
16	44746	1314
17	39179	1320
18	20912	1321
19	1266	1321
20	40610	1325
21	32261	1327
22	16756	1327
23	41522	1327
24	27209	1330
25	15037	1337
26	11834	1339
27	6202	1339
28	38640	1342
29	7952	1344
30	22384	1345
31	26194	1345
32	24153	1348
33	36046	1350
34	39892	1350
35	40663	1351
36	20558	1351
37	686	1351
38	56232	1353
39	3311	1353
40	47341	1354
41	21214	1337
42	19712	1337
43	5293	1341
44	9615	1342
45	31649	1342
46	56023	1343
47	42361	1343
48	45334	1343
49	27961	1345
50	21437	1345
51	5201	1347
52	12485	1348
53	58632	1349
54	9284	1350
55	45829	1351
56	48998	1351
57	7378	1354
58	43292	1354
59	23350	1354
60	29137	1354
61	8039	1355
62	1335	1356
63	55527	1357
64	11627	1358
65	6498	1358
66	6885	1359
67	35709	1361
68	31687	1362
69	47652	1362
70	39310	1362
71	33674	1363
72	7420	1364
73	53395	1369
74	58328	1370
75	31486	1371
76	5755	1372
77	54181	1373
78	20517	1373
79	42356	1373
80	56907	1374
81	35333	1356
82	25325	1357
83	13228	1357
84	48973	1358
85	38971	1358
86	46965	1358
87	29798	1360
88	8245	1360
89	5294	1365
90	57089	1365
91	10790	1365
92	54778	1368
93	35383	1369
94	56848	1371
95	48047	1374
96	15980	1374
97	56794	1374
98	37749	1375
99	49517	1376
100	25518	1376
101	10312	1377
102	24318	1377
103	225	1377
104	20065	1379
105	28112	1380
106	46912	1380
107	3673	1381
108	50680	1382
109	58821	1383
110	37118	1383
111	56378	1384
112	24562	1384
113	57356	1385
114	18237	1385
115	40915	1386
116	280	1388
117	47878	1393
118	4483	1393
119	37240	1393
120	39247	1395

TABLE APP3.11: DESCRIPTIVE SAMPLING: "FLYING" MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	33314	1119
2	1387	1155
3	12794	1237
4	19752	1265
5	14745	1277
6	36255	1279
7	33674	1288
8	44870	1291
9	30254	1296
10	4365	1298
11	52212	1301
12	1952	1304
13	2558	1309
14	14628	1309
15	8655	1311
16	42155	1311
17	24876	1315
18	31126	1317
19	25761	1317
20	30813	1320
21	34990	1321
22	31472	1322
23	43366	1323
24	25074	1325
25	9334	1326
26	22559	1329
27	58461	1330
28	8856	1334
29	58050	1337
30	15843	1337
31	26224	1337
32	20655	1337
33	18153	1339
34	59893	1340
35	51488	1340
36	12225	1340
37	21119	1349
38	13118	1350
39	24366	1350
40	5254	1352
41	5103	1353
42	39886	1355
43	23851	1356
44	29490	1358
45	44378	1359
46	56315	1360
47	4349	1360
48	26330	1360
49	40010	1361
50	20041	1362
51	34077	1362
52	20121	1363
53	4657	1364
54	52747	1366
55	51028	1367
56	629	1368
57	13471	1371
58	50236	1372
59	58193	1372
60	50176	1373
61	26020	1373
62	31508	1373
63	34863	1373
64	27261	1374
65	41684	1374
66	33554	1374
67	30953	1374
68	53808	1375
69	12232	1375
70	15816	1375
71	4324	1377
72	43818	1377
73	15029	1377
74	23318	1378
75	24137	1380
76	49152	1380
77	14961	1381
78	17278	1381
79	52761	1381
80	42286	1382
81	59190	1382
82	1321	1382
83	42188	1382
84	8594	1383
85	31050	1383
86	48514	1383
87	31129	1384
88	10228	1384
89	4766	1384
90	36554	1384
91	44426	1387
92	10656	1388
93	2416	1388
94	14898	1388
95	18394	1389
96	42227	1390
97	58039	1390
98	21645	1391
99	14227	1391
100	48537	1391
101	26047	1391
102	42090	1392
103	42706	1393
104	31594	1394
105	6092	1395
106	58851	1396
107	9416	1397
108	17950	1398
109	46088	1398
110	11700	1398
111	27152	1401
112	28332	1402
113	50223	1402
114	13888	1402
115	26962	1404
116	38552	1406
117	6905	1406
118	34189	1407
119	23273	1408
120	39585	1409

TABLE APP3.12: DESCRIPTIVE SAMPLING: LEWIS MLGC (SEED VALUES 1-60000)

Rank	Seed	Quality Measure
1	17716	1172
2	50567	1223
3	37635	1258
4	38236	1265
5	48660	1272
6	14877	1284
7	22992	1297
8	21708	1300
9	51255	1307
10	31632	1307
11	45702	1307
12	2884	1308
13	32295	1312
14	15449	1316
15	21263	1323
16	30602	1326
17	51185	1328
18	18690	1329
19	18278	1329
20	46710	1329
21	33302	1329
22	12236	1331
23	47057	1332
24	30459	1333
25	59279	1334
26	50989	1335
27	22196	1339
28	47136	1340
29	54756	1341
30	12644	1344
31	26789	1345
32	45206	1348
33	53508	1349
34	14747	1349
35	12006	1351
36	6015	1352
37	1298	1353
38	57975	1353
39	54776	1354
40	42999	1354

Rank	Seed	Quality Measure
41	15118	1354
42	40209	1354
43	29709	1355
44	20587	1356
45	35634	1358
46	31757	1359
47	5832	1361
48	34238	1361
49	34057	1364
50	24126	1365
51	59039	1366
52	45244	1367
53	59007	1367
54	53814	1370
55	26620	1370
56	29529	1370
57	1782	1371
58	53553	1372
59	43514	1372
60	31054	1372
61	36268	1374
62	55715	1375
63	42375	1375
64	1694	1377
65	871	1377
66	42302	1378
67	49439	1379
68	46293	1380
69	11945	1380
70	982	1381
71	40049	1382
72	51688	1382
73	3734	1383
74	29142	1383
75	37154	1383
76	36081	1385
77	36211	1385
78	56620	1386
79	7172	1388
80	10959	1389

Rank	Seed	Quality Measure
81	27754	1389
82	48366	1389
83	16115	1390
84	50331	1390
85	9655	1391
86	2379	1391
87	41191	1392
88	10151	1392
89	43679	1392
90	10327	1393
91	35473	1393
92	10568	1394
93	34010	1394
94	15098	1395
95	28652	1395
96	43665	1396
97	9141	1397
98	23639	1398
99	3985	1398
100	32881	1401
101	45732	1402
102	52612	1402
103	23019	1402
104	14252	1403
105	14724	1403
106	15670	1404
107	50868	1404
108	14116	1405
109	37171	1406
110	23411	1406
111	56442	1407
112	2540	1413
113	3598	1413
114	45602	1414
115	49344	1415
116	17303	1415
117	6531	1416
118	36389	1416
119	41023	1416
120	7889	1417

APPENDIX 4: DETERMINING THE MISSING SEED IN FLYING

A count of seeds for the MLCG referred to as Flying, when an offset or additive constant of 1 is used, indicated that there was one missing seed. The program MISSING (see Appendix 1) detected that there was a missing seed. The missing seed is 1395119659. The seed has a cycle of 1; that is the seed generates itself.

APPENDIX 5: STATISTICAL TERMINOLOGY USED

In this thesis the following convention of terminology has been used.

Sample

The sample is the set of sample values that are to be investigated.

Sample Values

The sample value is the value actually measured. In the study of how to provide a representative sample of 1000 random negative exponential deviates for a simulation run, our original sample is the sample of 1000 negative exponential deviates created from the random source being evaluated. A sample value is an individual negative exponential deviate.

A simple example to illustrate the use of the terminology would be if the investigation were into the relationship between weight and height, the sample values would be the pairs of measurements of height and weight of the individual included in the study. The sample would be the whole set of sample values. It is assumed not everyone has been measured and only a relatively small number have been measured.

Test Variable

The test value is the value of the variable created from the sample value or values that relates to the property being studied. In our evaluation of random numbers it was the sample values themselves and sets of test values created by summing a number of sequential values. There were in fact in this case ten sets of test variables created from the one set of samples.

In the example of weight and height the test variable chosen could well be the correlation coefficient (R). In this case the set of samples values will be reduced to a single variable.

Test Statistic

In order to examine the test variable statistically it may well be necessary to perform another transformation, this time transforming the test variable into the test statistic. In our case there was a conversion into χ^2:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where

The summation is over a set of ranges or cells that will contain all the values of the test variable.

E is the expect number of times the test variable will fall into a range or cell

O is the observed number of times the test variable falls into a range or cell.

An approximation of the distribution of χ^2 is known.

In the case of the correlation coefficient *r*, the following conversion is often made to *t*,

$$t = \frac{r\sqrt{N-2}}{\sqrt{1-r^2}}$$

Where N is the number of samples

This test statistic is distributed as Student's t with N-2 degrees of freedom.

APPENDIX 6: OBTAINING A SOURCE OF RANDOM NUMBERS

AP6.1 The First Attempt

To obtain a set of real random numbers, four files of 10 Megabyte of random bytes were downloaded from the web site, Random.org (Haahr 1999). These were named on the web site as, "10Megs.1", "10Megs.2", "10.Megs.3" and "10.Megs.4". The files are streams of random bits created as previously described, from an unused radio frequency.

The first attempt to read the files was using a FORTRAN program "READRAN" version 1.

FORTRAN allows three Integer sizes, with 1, 2 and 4 bytes used respectively, to hold the number. In an attempt to generate as many random numbers from the stream of random bits, the first attempt was to read using an integer defined by only two bytes. In this version of the program, two consecutive bytes were read to create each random number.

To enable more random numbers to be created the file could be read again after first pass by ignoring the first byte in the file. Thus is called a "nudge" in the program. Thus without a "nudge", the first integer was bytes 1 and 2, the second 3 and 4 and so on. If a nudge is invoked, the first integer is formed from bytes 2 and 3, the second 4 and 5 and so on. By using a nudge the number of integers able to be read from the files was 41,943,036.

The values taken by the integer variable created by reading from the file {note that the compiler does not normally allow the value –32768 (or -2^{15}) to be available for use} were from –32768 to 32767 (or 2^{15}-1).

The random variable was calculated by adding 32769 and then dividing by 65537. This gave a floating-point variable lying between 0 and 1.

To test the method of extraction and the conversion, the random variables produced from each pass were counted into 100 equally spaced cells between 0 and 1. To test the even spread of numbers, a Chi-Square test was applied. The results are shown in Table AP6.1.

Each pass of the files gave values that were acceptable, but the worst was acceptable only at 5.01%. However when the data was considered as one total set it was only acceptable at 0.95%. This is shown in the small table where the χ^2 values are calculated at the foot of Table AP6.1. This was considered unacceptable.

AP6.2 The Second Attempt

In an attempt to resolve the problem, the number of meaningful decimal places of the random number was increased so they could be more accurately assigned to one of the 100 cells. To achieve this the integer size was increased to 4 bytes.

The computer program was amended to read four bytes from the file for each integer. This meant only half the number of integers could be read by one pass of the file. However the file may be "nudged" three times, thus enabling the file to be read a total of four times rather than twice.

On the first pass, bytes 1,2,3 and 4 made the first integer, 5,6,7 and 8 the second and so on. For the second pass after a "nudge", bytes 2,3,4 and 5 made the first integer, and 6,7,8 and 9 made the second. The third run was made after two "nudges ", bytes 3,4,5 and 6 made the first integer and 7,8,9 and 10 made the second and the fourth pass was made after three "nudges", with 4,5,6 and 7 being the first integer and 8,9,10 and 11 being the second.

In this second version, the values taken by the integer variable by reading from the file {note that the compiler does not allow the value –2147483648 (or -2^{31}) to be available for normal use} were from –2147483648 to 2147483647 (or 2^{31}-1).

The random variable was calculated by adding 2.147483649×10^9 and then dividing by: 4.294967297×10^9. The calculation was performed in double precision floating point. This gave a double precision variable lying between 0 and 1.

The number of random variables able to be created from the four files dropped from 41,943,036 to 41,943,028, a very small decrease of 8.

The method of extraction and the conversion was again tested by the random variables produced from each pass being counted into 100 equally spaced cells between 0 and 1. A Chi-Square test was applied.

The results are shown in Table AP6.2.

The results are acceptable all the individual sets are acceptable at the level of 7.5% and the whole is acceptable at 11%. This method was used to produce the real random number used in the testing and calibration of the test and evaluation of real random numbers as source of randomness.

Program "READRAN Version 2" is given in Appendix 1.

Table APP6.1: Results of Test Using READRAN Version 1

(Note each row represents one of the 100 cells).

Counts									
10megs.1		10megs.2		10megs.3		10megs 4		Total	
Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Counts	Chi Sq
52361	52386	52964	52188	52555	52593	52351	52874	420272	1.6889
52965	52209	52142	52733	52665	52765	52620	52485	420584	3.1731
51921	52116	52426	52560	52188	52326	52515	52280	418332	2.8763
52093	52174	51920	52495	52254	52779	52764	52857	419336	0.0212
52518	52391	52320	52580	52514	52782	52278	52216	419599	0.0678
52570	52702	52731	52535	52534	52465	52633	52058	420228	1.5169
52606	52332	52268	52298	52312	52379	52275	52509	418979	0.4857
52300	52541	52268	52607	52559	52013	52525	52307	419120	0.2297
52197	52407	52554	52332	52366	52729	52215	52094	418894	0.6859
52595	52452	52512	52329	52208	52279	52508	52121	419004	0.4334
52503	52111	52351	52321	52287	52803	52420	52415	419211	0.1147
52603	52358	52524	52292	52280	52135	52139	52381	418712	1.2303
52625	52193	52972	52316	52740	52747	52156	52696	420445	2.4545
53076	52395	52654	52937	52370	52310	52025	52754	420521	2.8360
52370	52759	52264	52316	52590	52252	52325	52061	418937	0.5803
52515	52706	52380	52600	52424	52378	52291	52634	419928	0.5904
52456	52596	52816	52840	52202	52295	52263	52300	419768	0.2718
52429	52377	52658	52337	52246	52525	52342	52608	419522	0.0200
52294	52487	52145	52914	52477	52814	52176	52242	419549	0.0336
52381	52435	52168	52309	52468	52735	52771	52150	419417	0.0004
52192	52207	52289	52264	52373	52077	52417	52081	417900	5.5838
52258	52579	52591	52033	52619	52361	52432	52425	419298	0.0418
52413	52646	52166	52710	52026	52553	52451	52205	419170	0.1616
52415	52338	52311	52397	52999	52594	52370	52347	419771	0.2767
52708	52487	52673	52295	52763	52410	52390	52468	420194	1.3903
52321	52159	52653	52676	52241	52286	52737	52653	419726	0.2084
51969	52201	52618	52496	52214	52265	52662	52332	418757	1.0810
52616	52735	52370	52795	52311	52588	52143	52482	420040	0.8861
52566	52769	52148	52351	52231	52027	52338	52360	418790	0.9777
52670	52191	52523	52795	52793	52347	52298	52296	419913	0.5554
52434	52442	52541	52471	52335	52299	52533	52599	419654	0.1192
52024	52113	52740	52318	52390	52179	52502	52250	418516	1.9933
52479	52785	52381	52331	52901	51988	52903	52416	420184	1.3542
52193	52491	52690	51883	52050	52460	52371	51844	417982	5.0014
52239	52505	52064	52459	52240	52481	52653	52462	419103	0.2555
52351	52504	52410	52594	52413	52648	52211	52616	419747	0.2390
52153	52017	52329	52517	52226	52318	52518	52415	418493	2.0949
52219	52347	52364	52702	52809	52823	52773	52459	420496	2.7075
52204	52537	52449	52544	52419	52436	52621	52015	419225	0.1005

Table APP6.1: Results of Test Using READRAN Version 1 (Continued)

(Note each row represents one of the 100 cells).

Counts									
10megs.1		10megs.2		10megs.3		10megs 4		Total	
Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Counts	Chi Sq
52151	52640	51996	52780	52647	52558	52639	52324	419735	0.2213
52488	52650	52664	52570	52595	52494	52385	52548	420394	2.2140
52773	52285	52425	52383	52743	52204	52502	52469	419784	0.2982
52305	52414	52649	52345	52218	52727	52581	52199	419438	0.0001
52463	52509	52178	52670	53018	52470	52506	53067	420881	5.0172
52416	52146	52322	52486	52418	52316	52594	52605	419303	0.0387
52347	52879	52605	52046	52566	52618	52796	52115	419972	0.6995
52520	52565	52711	52362	52164	52607	52594	52894	420417	2.3209
52152	52332	52704	52362	52104	52624	52498	52069	418845	0.8169
52220	52923	52771	52364	52333	52661	52837	52865	420974	5.6811
52380	52339	52757	52919	52721	52576	52500	52616	420808	4.5249
52858	52665	52655	52209	51898	52505	52875	52648	420313	1.8574
52614	52305	52397	52145	52299	52311	52951	52748	419770	0.2750
52378	52138	52168	52706	52261	52163	52711	52731	419256	0.0725
52349	52238	52619	52714	52266	52370	52517	52288	419361	0.0115
52241	52829	52784	52267	52612	52499	52450	52624	420306	1.8281
52651	52518	52713	52463	52697	52641	52599	52358	420640	3.4886
52805	52438	52975	52706	52278	52913	52361	52425	420901	5.1565
52467	52149	52323	52367	51873	52487	52636	52521	418823	0.8795
52955	52331	52001	52407	52282	52281	52310	52267	418834	0.8479
52753	52033	52915	52423	52206	52120	52480	52691	419621	0.0866
52304	52461	52547	51935	52267	52611	52705	52701	419531	0.0241
52460	52406	51993	52169	52633	52562	52355	52166	418744	1.1232
52603	52600	52600	52584	52831	52414	52678	52392	420702	3.8554
52344	52454	52531	52286	52480	51958	52477	52328	418858	0.7810
52629	52222	52622	51971	52461	52430	51958	52627	418920	0.6210
52664	52374	52217	52701	52404	52734	52359	52736	420189	1.3722
52188	52658	52144	52337	52610	52531	52294	52019	418781	1.0053
52872	52952	52713	52128	52569	52683	52330	52345	420592	3.2172
52281	52233	52171	52544	52147	52245	52072	52724	418417	2.4483
52204	52540	52491	51973	52205	52204	52145	52038	417800	6.3373
52522	52766	52643	52499	52506	52164	52136	52382	419618	0.0839
52308	52659	52097	52830	52800	52754	52009	52287	419744	0.2345
52122	52284	52551	52166	52691	52864	52004	52418	419100	0.2602
52319	52159	52623	52355	52459	52532	52296	52617	419360	0.0118
51967	52565	52205	52937	52599	52170	52034	52682	419159	0.1756

Table APP6.1: Results of Test Using READRAN Version 1 (Continued)

(Note each row represents one of the 100 cells).

Counts									
10megs.1		10megs.2		10megs.3		10megs.4		Total	
Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Counts	Chi Sq
52428	52860	52535	52318	52246	52739	52743	52691	420560	3.0424
52482	52466	52310	52503	52177	52217	52417	52333	418905	0.6580
52851	52336	52347	52306	52349	52351	52496	52515	419551	0.0347
52714	52219	52308	53020	52316	52583	52281	52598	420039	0.8832
52212	52551	52485	52099	52614	52991	52320	52603	419875	0.4714
52335	52746	52156	52350	52118	52176	52431	52228	418540	1.8900
52633	52274	52772	52246	52809	52533	52132	52317	419716	0.1945
52837	52141	52486	52352	52325	52366	52435	52442	419384	0.0051
52296	52117	52395	52368	52592	52187	52413	52489	418857	0.7838
52651	52171	52271	52633	52471	52205	52237	52681	419320	0.0290
52144	52741	52287	52204	52399	52479	52245	52799	419298	0.0418
52311	52238	52636	52389	52147	52995	52313	52306	419335	0.0217
52755	52401	52080	52696	52470	51997	52404	52501	419304	0.0381
52346	52245	52496	52199	52242	51986	52506	52537	418557	1.8186
52206	52508	52212	52192	52573	52280	51936	52268	418175	3.7573
51999	52706	51977	52466	52373	52102	52069	52452	418144	3.9452
52376	52554	52601	52853	52535	52072	52643	52303	419937	0.6120
52197	51933	52318	52263	52343	52220	52432	52406	418112	4.1439
52573	52393	52226	52329	52650	52233	52273	52041	418718	1.2099
52257	52186	51829	51936	52621	52245	52725	52941	418740	1.1363
52042	52279	52342	52349	52389	52089	52458	52262	418210	3.5507
52909	52553	51965	52144	52235	52266	52269	51847	418188	3.6799
52674	52701	52121	52391	52512	52013	52261	52570	419243	0.0837
52300	52391	52274	52769	52484	52620	52785	52301	419924	0.5810
52477	52330	52624	51925	52437	52689	52567	52147	419196	0.1310

Calculation of χ^2

	10megs.1		10megs 2		10megs 3		10megs 4		
	Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Pass1	Pass2	Total
Sample Size	5242880	5242879	5242880	5242879	5242880	5242879	5242880	5242879	41943036
Chi- Sq. Value	112.351	95.180	123.210	120.434	97.705	118.851	95.693	114.502	134.9697
Prob. of Chi-Sq. Value	16.95%	59.00%	5.01%	7.05%	51.79%	8.49%	57.54%	13.66%	0.95%

Table APP6.2: Results of Test Using READRAN Version 2

Counts																	
10megs.1				10megs.2				10megs.3				10megs.4				Total	
Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Counts	Chi-Sq.
26120	26264	26269	26159	26451	26210	26541	26001	26230	26311	26360	26320	26106	26454	26270	26454	420520	2.831
26586	26142	26416	26084	26162	26356	26009	26404	26278	26369	26405	26432	26361	26371	26288	26134	420797	4.453
25936	25900	25931	26168	26099	26419	26272	26094	25900	26176	26247	26080	26233	26233	26230	26001	417919	5.445
25984	26103	26135	26095	25939	26283	26015	26250	26297	26643	25976	26163	26304	26487	26489	26393	419556	0.038
26081	26193	26462	26231	26211	26270	26130	26330	26061	26577	26480	26242	26211	26059	26107	26190	419835	0.391
26320	26575	26201	26078	26126	26171	26560	26315	26294	26178	26196	26228	26294	26142	26283	25871	419832	0.385
26169	26114	26456	26248	25970	26043	26335	26281	26165	26053	26161	26365	26280	26137	26017	26397	419191	0.137
26010	26086	26328	26491	26241	26207	26041	26430	26421	26189	26186	25836	26498	26147	26060	26207	419378	0.007
26131	26218	26016	26123	26255	26139	26257	26141	26104	26275	26212	26411	25912	26163	26253	25859	418469	2.203
26161	26244	26459	26237	26355	26113	26185	26228	26256	26394	25982	25920	26398	26266	26146	25882	419226	0.099
26238	26142	26211	25919	26478	26138	25822	26149	26274	26416	25952	26333	26187	26178	26176	26185	418798	0.953
26070	26246	26568	26149	26144	26138	26415	26186	26087	26094	26220	26062	26185	26371	25987	26050	418972	0.501
26180	26101	26485	26108	26480	26189	26507	26144	26442	26294	26329	26500	26103	26478	26080	26226	420646	3.524
26630	26328	26383	26024	26385	26501	26224	26396	26219	26039	26102	26208	26040	26269	25930	26446	420124	1.147
26257	26443	26146	26347	26133	26052	26155	26301	26131	26232	26490	26055	26208	26051	26139	26032	419172	0.159
26306	26381	26240	26359	25998	26271	26415	26346	26232	26124	26219	26274	26132	26186	26189	26490	420162	1.277
26098	26155	26299	26379	26356	26344	26403	26450	26145	26142	26004	26102	26092	26054	26125	26185	419333	0.023
26121	26064	26341	26338	26330	26127	26354	26237	26453	26279	25816	26268	26046	26232	26315	26409	419730	0.214
25914	26079	26324	26363	26128	26354	25976	26511	26028	26358	26410	26414	26108	25979	26029	26209	419184	0.145
26042	26222	26369	26246	26104	26070	26091	26268	26444	26295	26056	26471	26471	26033	26330	26151	419663	0.129
26091	26127	26133	26110	26290	26062	26035	26240	26039	26076	26348	26025	26064	26301	26367	25819	418127	4.050
26074	26111	26130	26412	26123	25959	26409	26010	26385	26157	26191	26157	26292	26092	26096	26266	418864	0.765
26344	26105	26106	26573	26049	26321	26151	26418	26049	26352	26008	26226	26327	26046	26167	26188	419430	0.000
26392	26137	26046	26229	25909	26206	26414	26222	26564	26477	26458	26151	26328	26326	26055	26049	419963	0.677
26449	26263	26206	26171	26344	26012	26290	26231	26449	26150	26264	26205	26001	26125	26342	26294	419796	0.319

Table APP6.2: Results of Test Using READRAN Version 2 (Continued)

Counts																	
10megs.1				10megs.2				10megs.3				10megs.4				Total	
Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Counts	Chi-Sq.
25947	26234	26401	25959	26355	26220	26330	26485	26084	26007	26197	26302	26394	26337	26365	26349	419966	0.684
25958	26122	26051	26108	26342	26240	26308	26293	26102	26182	26124	26116	26366	26344	26336	26011	419003	0.435
26145	26323	26411	26356	26201	26294	26115	26439	26076	25995	26191	26539	26119	26140	25967	26295	419606	0.074
26342	26396	26244	26406	26141	26227	26039	26162	26128	26029	26140	26026	26313	26223	26055	26153	419024	0.394
26417	26034	26207	26102	26179	26388	26290	26341	26285	25964	26453	26340	26082	26334	26168	25918	419502	0.012
26339	26173	26122	26299	26332	26224	26239	26287	26439	26133	25918	26193	26331	26627	26230	26003	419889	0.502
26144	26147	25928	25987	26277	26175	26493	26164	26095	26096	26343	26109	26244	26188	26291	26088	418769	1.043
26264	26337	26142	26403	26226	26196	26100	26092	26438	25913	26392	26023	26236	26163	26604	26205	419734	0.220
26250	26332	25975	26189	26322	25728	26402	26179	26013	26151	26075	26333	26040	26129	26367	25752	418237	3.395
25882	26288	26391	26240	25993	26226	26091	26263	26026	26165	26229	26348	26459	26201	26221	26273	419296	0.043
26152	26369	26147	26088	26063	26263	26296	26280	26101	26404	26269	26196	25990	26267	26172	26309	419366	0.010
26132	25962	26048	26077	26228	26147	26125	26399	26259	26447	25996	25893	26326	26426	26226	26017	418708	1.244
26240	26112	25917	26191	26022	26121	26295	26531	26283	26296	26471	26483	26394	26240	26321	26170	420087	1.028
26225	26139	26020	26425	26274	26271	26205	26301	26105	26227	26347	26236	26268	25819	26385	26219	419466	0.003
26200	26383	25981	26287	26110	26423	25909	26379	26214	26529	26459	26061	26286	26054	26381	26291	419947	0.637
26077	26358	26362	26233	26306	26208	26311	26312	26387	26218	26157	26229	26124	26161	26206	26347	419996	0.763
26288	26171	26514	26153	26356	26224	26101	26198	26513	26050	26261	26183	26125	26460	26403	26031	420031	0.860
26217	26267	26106	26177	26383	26463	26284	25900	26183	26426	26061	26326	26322	25832	26297	26397	419641	0.106
26114	26275	26305	26178	26116	26142	26012	26475	26541	26184	26418	26227	26330	26672	26120	26350	420459	2.523
26519	26123	25923	26051	26459	26198	25900	26321	26081	26098	26367	26249	26292	26316	26336	26321	419554	0.036
26135	26358	26168	26474	26342	26040	26204	25950	26188	26344	26336	26228	26418	26076	26324	25982	419567	0.045
26220	26255	26323	26332	26389	26292	26362	26107	26005	26191	26184	26442	26319	26418	26306	26506	420651	3.553
26229	26244	25958	26119	26332	26380	26395	26001	26072	26423	26052	26229	26107	25932	26420	26158	419051	0.343
26118	26309	26046	26557	26443	26054	26281	26274	26264	26179	26036	26438	26447	26278	26337	26531	420592	3.218
26411	26222	25997	26154	26290	26274	26494	26668	26242	26204	26500	26396	26207	26243	26318	26414	421034	6.132

Table APP6.2: Results of Test Using READRAN Version 2 (Continued)

Counts																	
10megs.1				10megs.2				10megs.3				10megs.4				Total	
Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Counts	Chi-Sq.
26286	26242	26598	26456	26383	26082	26298	26160	25821	26253	26111	26292	26448	26156	26451	26514	420551	2.995
26279	26121	26284	26131	26063	26052	26293	26039	26124	26294	26121	25955	26367	26350	26542	26354	419369	0.009
26329	25897	26076	26266	26079	26385	26117	26347	25969	26199	26321	25995	26284	26316	26451	26445	419476	0.005
26145	26302	26236	25961	26251	26507	26388	26241	26109	26161	26191	26232	26294	26141	26251	26187	419597	0.066
26099	26459	26093	26322	26430	26159	26308	26050	26197	26245	26354	26206	26193	26327	26207	26238	419887	0.497
26378	26323	26299	26220	26267	26357	26463	26136	26345	26262	26388	26409	26332	26191	26294	26191	420855	4.839
26491	26026	26260	26372	26549	26264	26389	26394	25987	26446	26235	26406	26277	25836	26041	26544	420517	2.816
26213	26180	26292	25989	26331	26155	26015	26242	25852	26087	26053	26440	26167	26194	26484	26350	419044	0.356
26631	26187	26339	26175	25993	26283	26035	26155	25931	26101	26378	26206	26213	26149	26129	26137	419042	0.359
26303	26277	26412	25707	26583	26169	26285	26191	26045	26275	26107	25785	26178	26314	26261	26342	419234	0.092
26109	26280	26222	26208	26416	26049	26158	25922	26105	26363	26195	26280	26343	26410	26384	26312	419756	0.253
26428	26361	26054	26077	26015	26048	26006	26142	26433	26196	26232	26394	26050	26013	26335	26169	418953	0.543
26240	26421	26323	26128	26261	26466	26293	26061	26359	26048	26415	26312	26183	26067	26452	26291	420320	1.887
25970	26219	26397	26260	26168	26043	26385	26283	26339	26031	26169	25963	26309	26207	26187	26148	419078	0.296
26380	26100	26192	26075	26125	26080	26451	25838	26046	25890	26365	26503	25972	26372	25940	26208	418537	1.902
26116	26152	26579	26247	26011	26579	26231	26157	26207	26471	26230	26278	26227	26390	26156	26370	420401	2.247
26022	26517	26197	26166	26112	26122	26056	26238	26285	26324	26348	26235	26062	26089	26264	25955	418992	0.458
26446	26497	26381	26402	26326	26308	26335	25769	26250	26263	26269	26365	26253	26207	26024	26086	420181	1.344
26181	26272	26125	25998	26156	26364	26051	26211	25805	26081	26365	26197	26190	26412	25911	26345	418664	1.400
26119	26037	26103	26529	26389	25883	26131	26118	26231	26077	26013	26154	26167	26134	26006	25928	418019	4.749
26398	26488	26085	26219	26093	26229	26483	26225	26471	25975	25980	26143	25946	26112	26139	26219	419205	0.121
25995	26365	26337	26330	26096	26481	26044	26370	26406	26480	26420	26304	25889	26114	26148	26207	419986	0.736
26256	26257	25819	25977	26217	26252	26289	25857	26339	26284	26304	26543	25970	26329	25992	26040	418725	1.186
26422	26171	25931	26017	26165	26243	26481	26142	26311	26245	26172	26285	26108	26186	26208	26457	419544	0.031
25990	26023	25998	26570	26228	26386	26006	26588	26311	26236	26319	25977	26108	26481	25955	26229	419405	0.002

Table APP6.2: Results of Test Using READRAN Version 2 (Continued)

Counts																	
10megs.1				10megs.2				10megs.3				10megs.4				Total	
Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Counts	Chi-Sq.
26347	26390	26027	26413	26179	26175	26300	26087	25953	26398	26241	26279	26408	26561	26287	26064	420109	1.098
26320	26385	26192	26111	26541	26272	25805	26264	26331	26238	25874	26022	26346	26290	26091	26088	419170	0.162
26306	26341	26577	26025	26005	26337	26371	25999	25966	26198	26417	26174	26153	26328	26380	26217	419794	0.315
26512	26167	26159	26019	26143	26423	26114	26543	26223	26170	26041	26360	26092	26250	26120	26290	419626	0.091
26208	26173	26025	26392	26210	26113	26301	26012	26332	26571	26301	26448	26281	26420	26078	26219	420084	1.019
26165	26521	26196	26256	26038	26175	26146	26209	25920	26006	26237	26211	26335	26249	26130	26000	418794	0.965
26504	26189	26085	26031	26493	26210	26229	25997	26505	26198	26252	26278	25954	25859	26121	26411	419316	0.031
26515	26063	26341	26109	26126	26115	26391	26250	26304	26154	26048	26242	26293	26181	26180	26295	419607	0.074
26073	25916	26169	26158	26041	26132	26301	26200	26261	26028	26286	26106	26174	26319	26189	26108	418461	2.240
26424	25965	26262	26226	26257	26400	26039	26253	26426	26206	26069	26031	26189	26453	26069	26256	419525	0.021
26080	26371	26096	26398	26241	26065	26081	26163	26277	26177	26147	26327	26042	26306	26235	26530	419536	0.027
26153	26207	26110	25982	26231	26071	26352	26277	25941	26466	26153	26482	26265	26053	25987	26213	418943	0.566
26301	26135	26468	26301	26233	26479	25877	26244	26074	26005	26424	26020	26102	26370	26345	26146	419524	0.021
26178	26003	26209	26262	26255	26100	26266	26118	26112	26131	26162	25882	26141	26218	26390	26348	418775	1.024
26031	26179	26118	26298	25979	26267	26176	25871	26351	26061	26164	26167	26118	26200	25777	26017	417774	6.540
25809	26385	26222	26327	25963	26315	26055	26179	26222	26125	26185	26012	25977	26200	26109	26284	418369	2.685
26112	26130	26214	26366	26196	26163	26348	26647	26472	25996	26010	26034	26280	26031	26320	26230	419549	0.034
25832	25840	26402	26128	26344	26165	26004	26126	26165	26084	26215	26147	26320	26033	26132	26377	418314	2.971
26090	26151	26501	26277	26039	26265	26221	26086	26409	26279	26262	25992	26200	25847	26109	26230	418958	0.532
26221	26176	25990	25955	26091	26066	25695	25832	26412	26160	26166	26031	26281	26412	26383	26486	418357	2.746
25971	25921	26097	26395	26374	26224	25992	26149	26348	26081	26063	26041	26161	26036	26334	26246	418433	2.371
26564	26358	26369	26215	25814	26058	26170	26115	26114	26212	26154	26076	26140	25903	26150	25979	418391	2.575
26282	26327	26342	26338	25881	26138	26188	26207	26298	26072	26160	25886	26398	26303	25815	26214	418849	0.806
26116	26276	26221	26135	26179	26366	26119	26426	26387	26199	26125	26457	26326	26130	26483	26191	420136	1.187
26431	26020	26068	26334	26339	26026	26320	25925	26288	26432	26180	26282	26214	25930	26387	26257	419433	0.000

Table APP6.2: Results of Test Using READRAN Version 2 (Continued)

	10megs.1				10megs.2				10megs.3				10megs.4			
	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4	Pass1	Pass2	Pass3	Pass4
Sample Size	2621440	2621439	2621439	2621439	2621440	2621439	2621439	2621439	2621440	2621439	2621439	2621439	2621440	2621439	2621439	2621439
Chi-Sq. Value	119.815	85.0455	113.938	98.26266	97.936	79.1702	118.416	119.802	114.197	91.85801	87.0598	111.046	71.1415	113.886	98.4484	112.576
Prob. Of Chi Sqr.	7.59%	84.00%	14.47%	50.20%	51.13%	92.89%	8.93%	7.60%	14.10%	68.18%	79.88%	19.20%	98.44%	14.55%	49.67%	16.58%

Total	
Sample Size	41943028
Chi Sqr. Value	116.462
Probability of Chi sq Value	11.1%

APPENDIX 7 TABLES OF REJECTION RATES

Number of Sequential Samples	Number of Cells k	Critical CHI-Square Value	Number of Samples Rejected	Measured Rejection Rate	Calculated Range of Measured Rejection Rate		Expected Values	
							Number of Samples Rejected	Rate of Rejection
1	200	232.91	2026	5.07%	4.73%	5.40%	2000	5.00%
2	100	123.23	1956	4.89%	4.56%	5.22%	2000	5.00%
3	66	84.82	2025	5.06%	4.72%	5.40%	2000	5.00%
4	50	66.34	2144	5.36%	5.01%	5.71%	2000	5.00%
5	40	54.57	2017	5.04%	4.71%	5.38%	2000	5.00%
6	33	46.19	2074	5.19%	4.84%	5.53%	2000	5.00%
7	28	40.11	1972	4.93%	4.60%	5.26%	2000	5.00%
8	25	36.42	1888	4.72%	4.39%	5.05%	2000	5.00%
9	22	32.67	2015	5.04%	4.70%	5.37%	2000	5.00%
10	20	30.14	1944	4.86%	4.53%	5.19%	2000	5.00%

Table APP7.1: Measured Rate of Rejection with 5% "Confidence" Level

Number of Sequential Samples	Number of Cells k	Critical CHI-Square Value	Number of Samples Rejected	Measured Rejection Rate	Calculated Range of Measured Rejection Rate		Expected	
							Number of Samples Rejected	Rate of Rejection
1	200	248.33	422	1.06%	0.90%	1.21%	400	1.00%
2	100	134.64	421	1.05%	0.90%	1.21%	400	1.00%
3	66	94.42	448	1.12%	0.96%	1.28%	400	1.00%
4	50	74.92	419	1.05%	0.89%	1.20%	400	1.00%
5	40	62.43	404	1.01%	0.86%	1.16%	400	1.00%
6	33	53.49	453	1.13%	0.97%	1.30%	400	1.00%
7	28	46.96	422	1.06%	0.90%	1.21%	400	1.00%
8	25	42.98	427	1.07%	0.91%	1.23%	400	1.00%
9	22	38.93	417	1.04%	0.89%	1.20%	400	1.00%
10	20	36.19	402	1.01%	0.85%	1.16%	400	1.00%

Table APP7.2: Measured Rate of Rejection with 1% "Confidence" Level

Number of Sequential Samples	Number of Cells k	Critical CHI-Square Value	Number of Samples Rejected	Measured Rejection Rate	Calculated Range of Measured Rejection Rate		Expected Values	
							Number of Samples Rejected	Rate of Rejection
1	200	254.13	215	0.54%	0.42%	0.65%	200	0.50%
2	100	138.99	218	0.55%	0.43%	0.66%	200	0.50%
3	66	98.10	222	0.56%	0.44%	0.67%	200	0.50%
4	50	78.23	221	0.55%	0.44%	0.67%	200	0.50%
5	40	65.48	221	0.55%	0.44%	0.67%	200	0.50%
6	33	56.33	247	0.62%	0.50%	0.74%	200	0.50%
7	28	49.65	228	0.57%	0.45%	0.69%	200	0.50%
8	25	45.56	255	0.64%	0.51%	0.76%	200	0.50%
9	22	41.40	221	0.55%	0.44%	0.67%	200	0.50%
10	20	38.58	207	0.52%	0.41%	0.63%	200	0.50%

Table APP7.3: Measured Rate of Rejection with 0.5% "Confidence" Level

Number of Sequential Samples	Number of Cells k	Critical CHI-Square Value	Number of Samples Rejected	Measured Rejection Rate	Calculated Range of Measured Rejection Rate		Expected Values	
							Number of Samples Rejected	Rate of Rejection
1	200	266.39	44	0.11%	0.06%	0.16%	40	0.10%
2	100	148.23	45	0.11%	0.06%	0.16%	40	0.10%
3	66	105.99	47	0.12%	0.06%	0.17%	40	0.10%
4	50	85.35	47	0.12%	0.06%	0.17%	40	0.10%
5	40	72.06	43	0.11%	0.06%	0.16%	40	0.10%
6	33	62.49	52	0.13%	0.07%	0.19%	40	0.10%
7	28	55.48	50	0.13%	0.07%	0.18%	40	0.10%
8	25	51.18	52	0.13%	0.07%	0.19%	40	0.10%
9	22	46.80	39	0.10%	0.05%	0.15%	40	0.10%
10	20	43.82	43	0.11%	0.06%	0.16%	40	0.10%

Table APP7.4: Measured Rate of Rejection with 0.1% "Confidence" Level

APPENDIX 8: TEST TO FAIL MANY RANDOM NUMBER GENERATORS

The development of tests that RNGs will fail is not difficult. A simple example that all generators of the form :

$$X_n = F[X_{n-1}, X_{n-2}, \ldots\ldots, X_{n-k}]$$

will fail can be created based on the fact that such generators always create the same value of X_n from the vector of $\mathbf{X_i}$ where i varies from n-k to n-1.

If it is considered that an attempt is to be made to plot the process in two dimensions, it is convenient to establish a transformation of the vector $\mathbf{X_i}$ into a single number. The method in the test is to select at random a vector $\mathbf{X_i}$ with i taken as 1 to k. This is given the number 1. This will create a value X_{k+1} this creates another vector $\mathbf{X_i}$ with i as 2 to k+1, this is given the value 2. Any duplicate vectors created are not renumbered but the next sequence number is given to the next unassigned vector. If no more vectors are being created to assign numbers but all vectors have not been assigned numbers, a vector not yet assigned is selected and the process continues with that vector. This continues until all vectors have a transformation number. Obviously this cannot be done in practice as there will never be enough time except for when k is 1, but it can form a mind experiment.

This can also be done (as a mind experiment) with a sequence of true random numbers.

The test is now applied by selecting another $\mathbf{X_i}$ vector and generating a sequence of vectors as in the normal process of producing random numbers. The vectors are transformed to the number given them in the calibration. A plot is formed of the nth number against the n+1 number. The real random numbers will create a random pattern. Any RNG of the above form will produce a set of points lying on a straight line. If the random number generator requires at certain intervals a number of random numbers to be skipped the points will lie on two lines. The different patterns created by the RNG and the real random numbers can easily be seen to be significantly different.

LIST OF REFERENCES

Ahrens J.H. and Dieter U., (1972), Computer Methods for Sampling from the Exponential and Normal Distributions, Commun. Assoc. Comput. Mach., 15, (1972) Pages 873-882

Arnold S F, (1990), Mathematical Statistics, Prentice Hall

Bennett D. J. (1998), Randomness, Harvard University Press, Massachusetts

Blum L, Blum M and Shub M, (1986) A Simple Unpredictable Pseudo-Random Number Generator, SIAM Journal of Computing, 15, 2 (1986) pages 364-383.

Blomqvist N., (1970), On the Transient Behavior of the G/G/1 Waiting-times, Skandinavisk Aktuarietidskrift, Pages 118-129

Brenner M. E., (1963), Selective Sampling-a technique for Reducing Sample Size in Simulation of Decision-Making Problems, The Journal of Industrial Engineering, Part 14 (Nov-Dec), Pages 291-295

Cheng R. C. H. (1976), A Note on the Effect of Initial Conditions on a Simulation Run, Operational Research Quarterly, Volume 27, Part 2ii, Pages 467 – 470

Cochran, W G, (1952), The χ^2 test of Goodness of Fit, Ann. Math. Statist., 23, Pages 315-345

Conway R. W., (1963), Some Tactical Problems of Digital Simulation, Management Science, Vol. 10 No 1, Pages 47-61

Couture R. and L'Ecuyer P., (1994), On the Lattice Structure of Certain Linear Congruent Sequences Related to AWC/SWB Generators, Mathematics of Computation, Vol 62 (No 206 April), Pages 799-808

Cox D. R. and Smith W. L., (1961), Queues, Chapman and Hall

Daley, D. J., (1968), Monte Carlo Estimation of the Mean Queue Size in a Stationary GI/M/1 Queue, Journal Operations Research Society of America, Vol. 16, No. 5, Pages 1002-1005

Devroye L, (1986), Non-Uniform Random Variate Generation, Springer-Verlag

Dudewicz E. J. and Ralley T. G., (1981), The Handbook of Random Number Generation and Testing with TESTRAND Computer Code, American Sciences Press, Ohio

Durstenfeld R., (1964), Algorithm 235 Random Permutation, Communications of the ACM, volume 7, Number 7, July 1964

Feller W., (1968), An Introduction to Probability Theory and Its Applications, New York, John Wiley

Ferrenberg A. M., Landau D.P., and Wong Y. J., (1992), Monte Carlo Simulations: Hidden Errors from "Good" Random Number Generators, Physical Review Letters, Vol. 69 (No 23 Dec), Pages 3382-3384

Fisher R A Sir, (1960), The Design of Experiments, Oliver and Boyd

Fishman G. S., (1978), Principles of Discrete Event Simulation, John Wiley & Sons

Fishman G. S. and Moore L. R., (1986), An Exhausting Analysis of Multiplicative Congruential Random Number Generators with Modulus 2^31-1, SIAM J Sci. Stat. Comput., Vol. 7 (No. 1), Pages 24-45

Gafarian A. V., Ancker C. J., (1966), Mean Value Estimation from Digital Computer Simulation, Operations Research, Volume 14, Pages 25-44

Gafarian A. V., Ancker C. J., Morisaku, (1978), Evaluation of Commonly Used Rules for Detecting "Steady-State" in Computer Simulation, Naval Research Logistic Quarterly, vol. 25, Pages 511-529

Golomb S. W. (1967), Shift Register Sequences, Holden-Day, Sans Francisco

Grove D. M. and Davis T. P., (1992), Engineering Quality & Experimental Design, Longman Group

Haahr M,, (1999), http://www.random.org.

Haahr M., (2002), Unpublished E Mail to A J Warn

Hacking I, (1965), Logic of Scientific Inference, Cambridge University Press

Harpell J. L., Lane M. S. and Mansour A. H., (1989), Operations Research in Practice: A Longitudinal Study, Interfaces, Vol. 19 (No 3), Pages 65-74

Hollocks B. W., (1966), The Effect of Starting Value on the FORTRAN PRN Generator, Workstudy and OR Department Internal Report, Samuel Fox and Co. Ltd, England

Hollocks B. W., (1995), Experimental Practice and the Implications for Simulation Software, In: Proceedings of the European Simulation Multi-Conference, edited by F. Breitnecker and I Husinsky, pages 153-158, ISCS.

IBM (1970), System/360 Scientific Subroutine Package Version III Programmer's Manual, Program Number 360A-CM-03X, IBM Corporation, New York.

Kao C. and Tang H., (1997), Several Extensively Tested Multiple Recursive Random Number Generators, Computers and Mathematics with Applications, Vol. 36 (No 6), Pages 129-136

Kelton W. D., (1985), Transient Exponential-Erlang Queues and Steady-State Simulation, Communications of ACM, Vol. 28 (No 7), Pages 741-749

Kelton W. D. and Law A. M., (1983), A New Approach for Dealing with the Startup Problem in Discrete Simulation, Naval Research Logistic Quarterly, Vol. 30, Pages 641-658

Kelton W. D. and Law A. M., (1985), The Transient Behavior of the M/M/s Queue, with Implications for Steady-State Simulation, Operations Research, Vol. 33, Pages 378-396

Kleijnen J. P. C., (1974), Statistical Techniques in Simulation Part 1, Marcel Dekker, New York

Kleijnen J. P. C. and van Groenendaal W., (1992), Simulation: A Statistical Perspective, John Wiley

Knuth, (1997), The Art of Computer Programming, Volume 2 Seminumerical Algorithms, Third Edition, Addison-Wesley

Knuth (1999), Changes to Volume 2: Seminumerical Algorithms, http://www-ca-faculty.stanford.edu/~knuth.

Kouikoglou V and Phillis Y A, (2001), Hybrid Simulation Models of Production Networks, 2001, Kluwer Academic

Ladbook J, (1998) Modelling Breakdowns: An Inquest, Unpublished MPhil.Thesis, University of Birmingham, England

Landau D. P., (1987), Monte Carlo Studies of Critical and Multicritical Phenomena, In: Applications of the Monte Carlo Method in Statistical Physics, Binder K. (Editor), Springer-Verlag, pages 93-123

Lanner Group (1998), WITNESS Version 9 User Manual, Lanner Group, (1999), Redditch, United Kingdom.

Law A M and Kelton W D, Simulation Modeling and Analysis, 3rd Edition, 2000, Mc Graw Hill

L'Ecuyer P., (1988), Efficient and Portable Combined Random Number Generators, Communications of the ACM, Vol. 31 (No 6) June, Pages 742-749,774

L'Ecuyer P., (1994), Uniform Random Number Generation, Annals of Operational Research, Vol. 53, Pages 77 to 120

L'Ecuyer P, (1999), Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure, Mathematics of Computation, 68, 225 (1999), pages 249-260, also available from http://www.iro.umontreal.ca/~lecuyer/papers.html.

L'Ecuyer P, (2001), Software for Uniform Random Number Generation: Distinguishing the Good and the Bad, Proceedings of the 2001 Winter Simulation Conference, IEEE Press, Dec 2001, pages 318-330, also available from http://www.iro.umontreal.ca/~lecuyer/papers.html.

L'Ecuyer P and Lemieux C, (1999), Quasi-Monte Carlo via Linear Shift-Register Sequences, Proceedings of the 1999 Winter Simulation Conference, IEEE Press, Dec 1999, pages 632-639, also available from http://www.iro.umontreal.ca/~lecuyer/papers.html.

Lehmer D. H., (1951), Mathematical Methods in Large-scale Computing Units, Annals of the Computing Laboratory of Harvard University 26, Harvard University

Lewis P A W, Goodman A S and Miller J M (1969), A Pseudo-random Number Generator for the System/360, IBM Systems Journal, Volume 8, 1969, Pages 136-147

Marsaglia G., (1968), Random Numbers Fall Mainly in the Planes, Proceedings of the National Academy of Science, Vol. 61, Pages 25-28

Marsaglia G., (1985), A Current View of Random Number Generators, In: Billard L. (editor) Computer Science and Statistics: The Interface, Elsevier Science, Pages 3-10

Marsaglia G (1993), Monkey Test for Random Number Generators, Computers & Mathematics with Applications, 9, 1993, Pages 1-10, also available form http://stat.fsu.edu/~geo/

Marsaglia G (2002), http://stat.fsu.edu/~geo/, web site maintained by B Narasimhan

Marsaglia G. and Zaman A, 1991), A New Class of Random Number Generators, The Annals of Applied Probability, Vol. 1 (No. 3) 1991, Pages 462-480

Matsumoto M and Nishimura T (1998), Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, Transactions on Modeling and Computer Simulation, 8(1), (1998) Pages 3-30 also available from http://www.math.keio.ac.jp/~matumoto/emt.html.

McCullough B.D. and Wilson B (1999), On the accuracy of statistical procedures in Microsoft Excel 97, Computational Statistics and Data Analysis, Vol 31, Pages 27-37

Mersenne Twister Home Page (2002), http://www.math.keio.ac.jp/~matumoto/emt.html.

Microsoft (1994), User Guide Microsoft Excel, version 5.0, Microsoft Corporation, 1994.

Murray J.R. and Kelton W. D, (1988), The Transient Behavior of the M/M/2 Queue and Steady-State Simulation, Computers and Operations Research, Vol 15, Pages 357-367

Neave H.R., (1989), Statistical Tables, Unwin Hyman, London

Niederreiter H., (1978), Quasi-Monte Carlo Methods and Pseudo-Random Numbers, Bulletin of the American Mathematical Society, Vol. 84 (No 6), Pages 957-1037

Niederreiter H., (1992), Random Number Generation and Quasi-Monte Carlo Methods, Philadelphia Society for Industrial and Applied Mathematics

Pearson M. A., (2000), The Incorporation of Target Performance Measures and Constrained Optimisation in the Newsboy Problem, Journal of the Operational Research Society, Volume 51, Pages 744 to 754

Pidd M., (1992), Computer Simulation in Management Science, John Wiley

Ripley B. D., (1987), Stochastic Simulation, John Wiley

Robson D. E., (1970), Tests on the ICL Random Number Generator, OR Department Internal Paper, British Steel Corporation Rotherham Works, England

Robinson S., (1995), An Heuristic Technique for Selecting the Run-Length of Non-Terminating Steady-State Simulations, Simulation, Vol. 65, No 3, September, Pages 170-179.

Robinson S., (2003), Unpublished E Mail to A J Warn

Saliby E, (1980) A Reappraisal of Some Simulation Fundamentals. Ph D Thesis, 1980, University of Lancaster

Saliby E, (1990a), Understanding the Variability of Simulation Results: An Empirical Study, Journal of the Operational Research Society, Vol. 41, No. 4, (1990), Pages 319-327.

Saliby E, (1990b), Descriptive Sampling: A Better Approach to Monte Carlo Simulation, Journal of the Operational Research Society, Vol. 41, No. 12, (1990), Pages 1133-1142.

Siegel S, (1956), Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill

Simulation Study Group, (1991), Simulation in UK Manufacturing Industry, Horrocks R. Editor, DTI Study Report, The Management Consulting Group, Coventry U K

Tajima A., Ninomiya S., and Tezuka S., (1998), Analysis of the anomaly of ran1() generator in Monte Carlo pricing of financial derivatives, Journal of the Operations Research Society of Japan, No 3 Sept, Pages 387-397

Takano H, (1999), FORTRAN Translation of MT19937, http://www.ath.keio.ac.jp/~matumoto/mt19937.f.

Tausworthe, (1965), Random Numbers Generated by Linear Recurrence Module Two, Maths Comp., No 19, Pages 201-209

Tezuka S., L'Ecuyer P. and Couture R., (1993), On the Lattice Structure of the Add-With-Carry and Subtract-With Borrow Random Number Generators, ACM Trans. On Modeling and Computer Simulation, Vol. 3 (No 4) Oct., Pages 315-331

Tocher K. D., (1960), The Art of Simulation, The English Universities Press

Wadsack K. and Tobias A., (1994) Simulating good simulation, OR Insight, Vol. 7, Issue 1, Pages 28-31.

Walker J, (1998), ENT: A Pseudorandom Number Sequence Test Program, http://www.fourmilab.ch/random/.

Warn A J, (1972), Planning Model for Material Handling in a Parts and Accessories Depot, In: Operational Research in Industrial Systems, edited by Brennan J., The English University Press

Welch P. D., (1983), The Statistical Analysis of Simulation Results, In: Lavenberg S. S. (Editor), Computer Performance Handbook, Academic Press, New York, Pages 267-329

Whitt W., (1991), The Efficiency of One Long Run Versus Independent Replicates in Steady-State Simulation, Management Science, Vol. 37 (No 6) June, Pages 645-666

Wilson J. R. and Pritsker A. B., (1978), Evaluation of startup policies in simulation experiments, Simulation, Vol. 31, Pages 79-89

Woodward J, (1961), Industrial Sociology Lecture at Imperial College London University, unpublished.

Yule G.U. (1938), A Test of Tippet's Random Sampling Numbers, Journal of the Royal Statistical Society, Volume 101, Pages 167-172