1964

The effect of pseudo-random number bias on the simulation of a data collection system

Richard J. White
Lehigh University

THE EFFECT OF PSEUDO-RANDOM NUMBER BIAS ON THE SIMULATION OF A DATA COLLECTION SYSTEM

by

Richard J. White

A THESIS

Presented to the Graduate Faculty

of Lehigh University

in Candidacy for the Degree of

Master of Science

Lehigh University
1964

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

8 May 1964
(date)

W. A. Smith Jr.
Professor in Charge

C. F. Bonilla
Head of the Department

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to the numerous people who made this study possible. The following people deserve special thanks for their contributions of time and effort:

Professor William Smith and Miss Bonnie Small for their general guidance and advice on the approach to the problem.

Professor Sutton Monro and Mr. Kenneth Stephens for their advice in designing the experiment and analyzing the results.

Professor Wallace Richardson and Mr. Leo Willner for their guidance as committee members.

Mr. Kenneth Larson for his assistance in the number theory aspects of the problem.

TABLE OF CONTENTS

APPENDICES

ABSTRACT

The question of the effect of biased pseudo-random numbers on simulation results arises in a majority of industrial simulation applications.

A partial answer to this question has been obtained in this study by the introduction of statistically acceptable and unacceptable random number generators into the simulation of a data collection system. Classical tests for randomness were used in the study to select several methods of random-number generation with varying degrees of statistical bias. These methods of generation were then used to simulate the data collection system. All parameters of the simulations remained fixed except the random number generator. The results of the simulations were then statistically analyzed.

Practitioners in the area of systems simulation, including the author, seem to feel that simulation results are significantly effected by random number bias. Surprisingly, the effect was found to be statistically insignificant in this particular application.

In conclusion, the results were proposed as only a partial answer to the question of the effect of pseudo-random number bias. The results indicated that further research is needed in this area. Recommendations for further study were made on the basis of the results.

1. INTRODUCTION

1.1 Objectives of the Investigation

The question that continually arises in a majority of simulation applications is, "What effect does pseudo-random number bias have on simulation results?" The author feels that this question is of considerable importance for the following reasons:

1. Theoretical work in the field of pseudo-random number generation is ordinarily published only in the advanced level mathematical journals. Consequently, the practicing industrial engineer or systems engineer is often unaware of the latest improved methods of generation.

2. Most text books and published articles covering simulation and the Monte Carlo method only briefly mention random number generation. The method mentioned is usually the "Mid-Square Method" which is one of the first methods proposed for random number generation. This method has subsequently been shown to have undesirable properties [7]. If a method with acceptable properties is mentioned, in most cases, nothing is said about the choice of starting values. However, the starting values are the factors that determine the randomness of the sequence.

3. Industrial conditions often subject Monte Carlo applications to pressing schedules and lack of development funds. When these conditions exist, the

method of random number generation is often haphazardly chosen and not sufficiently understood.

Because of these reasons, it is highly probable that poor methods of generation and incorrect starting values are being used in many applications of the Monte Carlo method.

The purpose of this study is the investigation of the effect of biased pseudo-random numbers on simulation results. The problem was studied by introducing both statistically acceptable and unacceptable random number generators into the simulation of a data collection system.

Classical tests for randomness were used as a criteria for judging the acceptability of several methods of pseudo-random number generation. From the results of the tests for randomness, six different pseudo-random sequences were selected on the basis of their statistical properties. By varying the starting random number, five independent simulations were then run using each of the six pseudo-random sequences. All of the parameters of the simulations were fixed except the six pseudo-random sequences. The simulation results were then statistically analyzed to determine the effect of biased pseudo-random numbers.

1.2 History of Pseudo-Random Number Generation

During the past few years, simulation has become a powerful technique in several of the fields of science and engineering. As advances are made in computer technology, larger and more complex simulations are made possible. High speed digital computers are presently being used to simulate such diversified applications as war games, job shop operation, traffic and waiting line systems, weapons systems, economic models, and business games. The basic underlying principle of simulation is known as the Monte Carlo method. Although the term Monte Carlo method is usually associated with simulation, it applies to any calculation requiring the use of random numbers. Statistical sampling and the solution of definite integrals are examples of calculations which also make use of the Monte Carlo method.

The use of the Monte Carlo method to solve complex problems on a digital computer has resulted in the need for extensive quantities of random numbers. It is not uncommon for a large simulation to require more than 100,000 random numbers. There are, in general, three ways of obtaining random numbers for computer use. First, tapes or card decks of random numbers could be produced and the numbers stored in computer memory or read from an input device as they are needed. This method is impractical in certain applications for several reasons. The limited availability of input devices, the relatively low speed of unbuffered input devices, and the large quantity of numbers required are all factors that can result in high costs. Mechanical or electronic means of generating random numbers, such as radioactive decay or electronic noise, have been considered as a second source of

random numbers. The cost of additional equipment and the fact that the sequences cannot be reproduced are both factors that make this method impractical for most applications.

The third and most practical method of obtaining random numbers for most computer applications is through a programmed subroutine using an arithmetic process. This method has none of the disadvantages that cause the previously mentioned methods to be impractical. One of the most desirable features of an arithmetic process is that the random sequence can be completely reproduced. This feature is invaluable for checking program logic and accuracy of results. However, this feature is also considered to be a disadvantage if the theoretical aspects of randomness are considered. Because the sequence of numbers can be determined in advance, and completely reproduced, the sequence cannot be said to be truly random. For this reason, random numbers generated within the computer have been given the name "pseudo-random." Although a sequence of numbers generated by an arithmetic process cannot be truly random, it has been shown that certain arithmetic processes generate numbers that appear to be random when the classical statistical tests for randomness are applied. For this reason, computer generated pseudo-random numbers are almost universally accepted for computer applications.

Considerable theoretical work has been done in the field of pseudo-random number generation. Reference [10] contains a bibliography of 143 articles pertaining to this subject. Number theory and statistical testing have been applied and as a result, arithmetic processes with improved properties of randomness have evolved. It has

also been shown that even the latest and supposedly better methods of pseudo-random number generation have poor statistical properties in some instances [1, 11].

2. METHODS OF GENERATING PSEUDO-RANDOM NUMBERS

The initial work in the field of pseudo-random number generation was done in the late 1940's and early 1950's. This work was carried out jointly by a number of authors and is summarized in [24]. The methods that received the most attention during the first years of investigation were the Mid-Square Method and the use of a Fibonacci series. The Fibonacci series approach was dropped soon after its introduction because unacceptable statistical and cyclic properties were shown to exist [24]. The Mid-Square Method was used in most applications until the multiplicative congruential method was introduced.

2.1 The Mid-Square Method

The Mid-Square Method has the undesirable property that the sequence does not return to the starting value to repeat. As a result, the period of the sequence is only found by trial for a particular starting value. It has been shown in [7] that the sequence can degenerate to zeros or to cycles of short periods. Because of its initial acceptance, the Mid-Square Method is included in most of the textbooks on operations research and systems engineering. For this reason, the method was chosen for use in this study.

The sequence is generated by the following procedure:

Choose an arbitrary n-digit starting number X_o.

Generate X_1 as the central n digits of the square of X_o. Repeat the process producing X_{i+1} from X_i in the same manner.

Example:

$X_o = 4917$

$X_o^2 = 24176889$

$X_1 = 1768$

$X_1^2 = 03125824$

$X_2 = 1258$

etc.

2.2 The Multiplicative Congruential Method

The multiplicative congruential method was first introduced by Lehmer [3] in 1949. However, Lehmer proposed only a special case of the method by restricting the constant multiplier to a specific value. The method was later generalized and defined as follows:

Let: a - Predetermined constant multiplier

X_o - Starting random number

M - Modulus

Then the sequence of numbers $X_1, X_2, \ldots\ldots X_i\ldots$ is defined by the recurrsion relationship

$$X_{i+1} \equiv aX_i \pmod{M}$$

This expression, with the congruence sign ($\equiv$) means that aX_i is divided by M and X_{i+1} is the remainder. The random sequence is then formed by $X_o/M, X_1/M, \ldots\ldots, X_i/M\ldots\ldots$.

If M is chosen as 10^L for a decimal computer or 2^L for a binary computer, both divisions can be eliminated.

Example:

$a = 2403$

$X_o = 7623$

$M = 10^4$

$aX_o = (2403)\ (7623) = 18318069$

$\frac{aX_o}{M} = 1831.8069$

$X_1 = 8069$ (Remainder after division by 10^4)

$\frac{X_i}{M} = .8069$

$aX_1 = (2403)\ (8069) = 19389807$

$$\frac{aX_1}{M} = 1938.9807$$

$$X_2 = 9807$$

$$\frac{X_2}{M} = .9807$$

etc.

As the example shows, the two divisions only become a case of the programmer assuming the decimal point placement.

Once M is chosen, number theory can dictate a choice of "a" and "X_o" which will result in the maximum number of terms before the sequence repeats itself. The number of terms before the sequence repeats itself is known as the period (P) of the sequence. The International Business Machines Corporation [12] has published a reference manual which covers the number theory determination of "a" and "X_o." This manual is also an excellent reference for the method of programming the multiplicative congruential method. Equations 2.11 through 2.13, given below, were taken from [12, p. 10] and apply to a decimal computer.

$$P = 5 \times 10^{L/2} \qquad (2.11)$$

$a = 200\,t \pm r$ where t is an integer and r is any of the values 3, 11, 13, 19, 21, 27, 29, 37, 53, 61, 67, 69, 77, 83, 91. (2.12)

(A value close to $10^{L/2}$ is a good choice.)

X_o = Any random number not divisible by 2 or 5. (2.13)

Equation 2.11 shows that the period is somewhat less than the maximum obtainable. The maximum period can be seen to be 10^L because

there are 10^L possible unique numbers in the sequence. However, the period can be made arbitrarily large by choosing a suitable L if the computer being used has a variable word length. The multiplicative congruential method is sometimes referred to as the power residue method.

This method has been shown to be statistically stable for a majority of the possible "a" values [5, 12, 24]. It is referred to in a number of the more recent publications in the operations research and systems engineering field. However, as previously stated, in most cases nothing is said about the choice of "a" values. The choice of a small "a" value can result in sequences that have unacceptable statistical properties. As a result, this method was chosen as the second method of generation for use in this study.

2.3 The Mixed Congruential Method

The third method chosen for use in this study is known as the mixed congruential method. The method was first introduced by Rotenburg [21] and Conveyou [4]. It has received considerable attention and gained some popularity due to several advantages that will be explained later. The sequence of generated numbers is defined as follows:

Let:

a - Predetermined constant multiplier

X_o - Starting random number

M - Modulus

c - Additive constant

Then the sequence $X_1, X_2, \ldots\ldots, X_i, \ldots\ldots$ is defined by the recurrsion relationship

$$X_{i+1} \equiv aX_i + c \pmod{M}$$

As was the case with the multiplicative congruential method, the random sequence is then formed by $X_o/M, X_1/M, \ldots\ldots, X_i/M \ldots\ldots$.

Also, M is chosen as 10^L for a decimal computer or 2^L for a binary computer.

Example: $a = 2401$

$X_o = 8624$

$M = 10^4$

$c = 1403$

$aX_o = (2401)\ (8624) = 20706224$

$aX_o + c = 20707627$

$$\frac{aX_o + c}{M} = 2070.7627$$

X_1 = 7627 (Remainder after division by 10^4)

$\frac{X_1}{M}$ = .7627

aX_1 = (2401) (7627) = 18312427

aX_1+c = 18313830

$\frac{aX_1+c}{M}$ = 1831.3820

X_2 = 3830

$\frac{X_2}{M}$ = .3830

etc.

Again, once M is chosen, number theory can dictate a choice of "a," "X_o," and "c" which will result in a maximum period. Equations 2.31 through 2.34, given below, were taken from [1, p. 132] and apply to a decimal computer.

$P = 10^L$ (2.31)

X_o = Any random number. (2.32)

c = Any random number not divisible by 2 or 5. (2.33)

a = 1 (mod 20) or a - 1 must be divisible by 20. (2.34)

One of the advantages of this method results from the fact that the maximum period (10^L) can be obtained using the above restrictions. Secondly, multipliers of the form $a = 10^L + 1$ can be used which results in considerable savings in computer time. By using multipliers of this form, the product can be formed using shift and add instructions in place of the multiply instruction. However, this

method has been shown to be statistically unstable for certain "a" values [1, 11]. It has also been shown in [1, 11] that the constant multiplier (a) is the main factor in determining the statistical properties of the random sequence for both the mixed and multiplicative congruential methods. The starting random number (X_o) only determines the beginning point of the particular sequence being generated.

2.4 A Modification of Lehmer's Method

As was previously mentioned, Lehmer first introduced the multiplicative congruential method with a specific "a" value. During the initial phase of this study, the methods of generating random numbers at Western Electric Company's Engineering Research Center were investigated. It was found that a modification of Lehmer's method had been used for some applications. Since the modification resulted in poor statistical properties, this method was chosen as the fourth method for use in this study. Lehmer's method is defined in [12, p.5] as follows:

> Multiply an 8 digit number by 23. Remove the two left-hand digits from the 10 digit product and subtract them from the 8 right-hand digits

Because;

$$X_{i+1} = 23\ X_i - K\ (10^8 + 1)$$

The method can be defined by the congruence relationship:

$$X_{i+1} \equiv 23\ X_i \ (\text{Mod}\ 10^8 + 1)$$

Because FORTRAN with FORMAT for the IBM 1620 imposes the restriction of an 8 digit product, Lehmer's method was modified by using a 6 digit starting random number, thus forming an 8 digit product. The modified method is defined as follows:

$$X_{i+1} \equiv 23\ X_i \ (\text{Mod}\ 10^6 + 1)$$

The random sequence is then formed by X_o/M, X_1/M,....., X_i/M....

Example:

$X_o = 3897$

$aX_o = (23)\ (3897) = 089631$

$X_1 = 9631 - 8 = 9623$

$$aX_1 = (23)\ (9623) = 221329$$

$$X_2 = 1329 - 22 = 1307$$

etc.

The choice of X_o is arbitrary for this method with the exception that starting values of zero and one must be excluded [12].

3. STATISTICAL TESTS FOR RANDOMNESS

There have been many tests for randomness proposed in the literature of statistics over the past thirty years. Since extensive random number testing requires considerable calculation, the computer time for these tests becomes excessive when a large number of tests are performed. As a result, it was decided to test the three properties of random sequences which would be most likely to affect waiting line simulation results. It is felt by the author that these properties are goodness of fit to the unit interval, independence of successive numbers, and runs up and down. Goodness of fit and independence tests are the only tests applied in a majority of the published articles on random number testing. A test for runs was added because it was felt that unexpected runs in a sequence could have a significant effect on waiting line simulations. Three classical tests representative of these properties were programmed for the IBM 1620. The details covering the operation and logic of the test program are covered in the next chapter. The following paragraphs summarize the statistical tests included in the program.

3.1 Test for Goodness of Fit to the Unit Interval

Although a precise definition of a random number is controversial, it is generally agreed in the field of statistical testing that a random number should have the following basic properties:

1. Each number in the set of all possible numbers has an equal probability of occurrence.
2. The probability of occurrence of any number is independent of the occurrence of any of the other numbers in the set.

If generated random numbers are considered as decimals, the first property listed above requires that the random numbers appear to be drawn from a uniform distribution with:

$$f(x) = 1 \text{ for } 0 < x < 1$$

The classical chi-square test for goodness of fit was chosen to test this property. The number of class intervals used for the chi-square test was arbitrarily chosen in most of the previous articles concerned with testing pseudo-random numbers [1, 11, 19].

In this study, the choice of the "best" class interval for the goodness of fit test was based on the work of Mann and Wald [17]. The following formula is proven in [17]:

Let: K_n = the "best" number of equal class intervals.

n = sample size.

α = the size of critical region.

c - defined by $\frac{1}{2\pi}\int_{c}^{\infty} e^{-t^2/2}\,dt = \alpha$

Then: $K_n = 4\sqrt[5]{\frac{2(n-1)^2}{c^2}}$

If $\alpha = 0.05$ and $n = 1000$, then $c = 1.645$ and $K_n \cong 99$. Therefore, the number of class intervals was chosen as 100.

The choice of the number of class intervals allows the goodness of fit test to be further defined.

Let: n = the quantity of random numbers being tested.

α = confidence level.

s = number of cells in chi-square summation.

t = number of population parameters estimated from sample.

$\theta = s - t - 1$ (Degrees of Freedom)

For this test, $t = 0$, $s = 100$, $\theta = 100 - 1 = 99$.

The following table illustrates the observed and expected frequencies used for calculating the test statistic.

INTERVAL	OBSERVED FREQUENCY	EXPECTED FREQUENCY (e_i)
0.00-0.01	f_1	0.01n
0.01-0.02	f_2	0.01n
.	.	.
.	.	.
.	.	.
0.99-1.00	f_{100}	$\frac{0.01n}{n}$

$$\chi^2_G = \sum_{i=1}^{100} \frac{(f_i - e_i)^2}{e_i} = \frac{1}{0.01n} \sum_{i=1}^{n} (f_i - 0.01n)^2$$

An arbitrary starting value is then selected and χ^2_G is calculated for 100 consecutive blocks in the sequence. If the sequence is random, χ^2_G is distributed χ^2_{99}. The χ^2_{99} distribution is then divided into deciles and the frequency (F_i) of χ^2_G values in each decile is calculated.

Let: E_i = expected frequency in each decile.

$$\chi^2_A = \sum_{i=1}^{10} \frac{(F_i - E_i)^2}{E_i} = \frac{1}{10} \sum_{i=1}^{10} (F_i - 10)^2$$

After calculating χ^2_A, the sequence is considered as unacceptable if

$$\chi^2_A \geq \chi^2_{\alpha,9}.$$

3.2 Serial Correlation or Independence Test

The serial correlation or independence test was chosen to test the basic property that the probability of the occurrence of any number is independent of the occurrence of any other number in the sequence. The test is performed by first forming a 10 X 10 contingency matrix. Each axis consists of the unit interval divided into ten equal class intervals. For each successive random number in the sequence, an entry is made in the contingency matrix.

The matrix entry rule can be stated as follows: Find the i^{th} row subinterval containing X_i, then find the j^{th} column subinterval containing X_{i+1}. In the cell representing row i, column j, tally a one.

A chi-square statistic is also used in this test. Using the values defined in section 3.1; $t = 0$, $s = 100$, $\theta = 100 - 1 = 99$.

Let: e_{ij} = the expected number in any cell.

$$e_{ij} = \frac{1}{100} \times n = 0.01n$$

f_{ij} = observed frequency in cell ij.

$$\chi^2_S = \sum_{j=1}^{10} \sum_{i=1}^{10} \frac{(f_{ij} - e_{ij})^2}{e_{ij}} = \frac{1}{0.01n} \sum_{j=1}^{10} \sum_{i=1}^{10} (f_{ij} - 0.01n)^2$$

One hundred χ^2_S values are then calculated in the same manner as in the goodness of fit test. Good has shown in [8] that χ^2_S is not distributed as an exact χ^2_{99} distribution. However, it is felt by the author that the approximation is sufficiently close for practical purposes. The frequency (F_i) of χ^2_S values in each decile is then

calculated.

Let: E_i = observed frequency in each decile.

$$\chi^2_B = \sum_{i=1}^{10} \frac{(F_i - E_i)^2}{E_i} = \frac{1}{10} \sum_{i=1}^{10} (F_i - 10)^2$$

After calculating χ^2_B, the sequence is considered as unacceptable if $\chi^2_B \geq \chi^2_{\alpha,9}$.

3.3 Run Test

The run test is also based on the two basic properties of a random number. If the generated numbers appear to have equal probability of occurence and are essentially independent, the actual frequency of runs up and down should approximate the theoretical frequency of runs for a truly random sequence.

The method of computing runs consists of forming an n - 1 binary sequence in the computer memory as the numbers are generated. The i^{th} term of the sequence is zero if $X_i < X_{i+1}$ and is one if $X_i > X_{i+1}$. After the sequence of zeros and ones is formed, the actual frequencies (f_K) of runs of all lengths are counted. A series of K zeros bracketed by ones form a run of zeros of length K. The same definition applies to a run of ones.

The expected number of runs (e_K) of length K is determined from probability theory. The following two expressions are taken from [12, p. 8].

$$e_K = \frac{2\left[(K^2+3K+1)\ n-(K^3+3K-K-4)\right]}{(K+3)!} \qquad (3.31)$$

$$\text{Total expected no. of runs} = \frac{2n\ -\ 1}{3} \qquad (3.32)$$

Example: n = the quantity of random numbers being tested.

Length of Run (K)	Expected Frequency (e_K) From 3.31	e_K for n = 1000
1	$\frac{5n+1}{12}$	417
2	$\frac{11n-14}{60}$	183
3	$\frac{19n-47}{360}$	53
4	$\frac{29n-104}{2520}$	12
5	$\frac{41n-191}{20,160}$	2
6	$\frac{55n-314}{181,440}$	0
		667

Group

Total expected No. of runs = $\frac{2n-1}{3} = 667$

For this test; $t = 0$, $s = 4$, (for $N \geq 2500$), $\theta = 4 - 1 = 3$

$$\chi^2_R = \sum_{k=1}^{4} \frac{(f_K - e_K)^2}{e_K}$$

One hundred χ^2_R values are then calculated in the same manner as in the goodness of fit test. If the sequence is random, χ^2_R is distributed χ^2_3. The χ^2_3 distribution is then divided into ten deciles and the frequency (F_i) of χ^2_R values in each decile is calculated.

Let: E_i = expected frequency in each decile

$$\chi^2_C = \sum_{i=1}^{10} \frac{(F_i - E_i)^2}{E_i} = \frac{1}{10} \sum_{i=1}^{10} (F_i - 10)^2$$

After calculating χ^2_C, the sequence is considered as unacceptable if $\chi^2_C \geq \chi^2_{\alpha, 9}$.

4. THE RANDOM NUMBER GENERATOR TEST PROGRAM

The logic of the random number generator test program is illustrated by Figure 1 of Appendix A. The program was written in the Symbolic Programming System Language for the IBM 1620. The program was then assembled using the Symbolic Programming System Processor Program (1620/1710 two pass paper tape version). A copy of the symbolic source program and the generated machine language program is included in a supplementary program appendix. A copy of the supplementary program appendix is on file in the Industrial Engineering Office, Lehigh University.

The program has a number of limitations which are listed below:

n = no. of random numbers generated (sample size).

1. The sample size (n) must be a multiple of 100 or

   $n = 100K \qquad K = 5, 6, 7, \ldots\ldots\ldots, 1000$

2. Also, n must fall in the following range:

   $$500 \leq n \leq 100{,}000$$

3. Runs of length four or greater are grouped into one cell for the chi-square test. This is done so that the expected number of runs of length four or greater is always greater than five. As a result, the calculated chi-square value is only approximate for $n > 2500$.

One of the principal advantages of the test program is that it can be used to test any method of random number generation. This feature is accomplished by requiring that the generation subroutine be an input rather than an integral part of the program. The generation

subroutine must be written in machine language and then punched on paper tape. Preparation of the machine language generation subroutine requires general knowledge of IBM 1620 machine language. Instructions for preparing this tape and other input tapes are included in the instructions for program operation shown in Appendix B.

The program begins by reading the machine language generation subroutine and the parameter tape. The parameter tape contains the starting values for each sequence being tested. The program then reserves a table of storage locations corresponding to the 100 class intervals for the goodness of fit test. The goodness of fit test is the first of the four tests included in the program. The test is performed by generating a random number and then comparing the number to a point on the unit interval. The point on the unit interval is initialized as 1000 and continually decremented by 10 until the class interval of the generated number is found. A one is then added to the proper storage location in the table previously mentioned. This cycle is repeated until the quantity of random numbers tested is equal to the sample size. At this time, χ^2_G is calculated and printed. A printout of the frequency table is optional through a program switch.

The independence or serial correlation test is the next test performed by the program. The independence test is similar in logic to the goodness of fit test. A contingency table of storage locations is first reserved in memory. Section 3.2 explains this contingency table. The test is performed by generating two successive random numbers. Each number is compared to a point on the unit interval. The points are both initialized as 1000 and decremented

by 10 until the respective class intervals of the two successive numbers are found. A one is then added to the storage location in the contingency table that is common to both class intervals. This cycle is repeated until the quantity of random numbers tested is equal to the sample size and the χ_S^2 is calculated and printed. Again, a printout of the contingency table is optional.

Figure 1 shows that the modified poker test is the next test performed by the program. During the initial phase of this study, it was found that this test is of little value. It was found that all generators, including those with generally poor statistical properties, passed this test. As a result, this test was not used for statistical testing of the random number generators considered in this study.

The last test performed by the program is the run test. As was the case with the other tests, the program begins by reserving a table for the observed frequency of runs of all lengths up to twenty. The program then generates two successive random numbers and compares them. If the first number is less than the second, a specified counter for upward runs is incremented. If the first number is greater than the second, a specified counter for downward runs is incremented. If the numbers are equal, the run is continued in the direction indicated by the last comparison. This cycle is repeated by generating a third number and comparing it to the second number in the same manner. Thus, each number in the generated sequence is compared to the previous number in the sequence. This procedure is essentially the same as the procedure stated in Section 3.3, which describes the run test. The n-1 binary sequence is formed but is not stored. The count

of runs of zeros and ones is made along with the comparison thus saving memory space. This cycle is repeated until the quantity of numbers tested is equal to the sample size and then χ^2_R is calculated and printed. The printing of the run frequency table is also optional with this test.

5. THE WAITING LINE MODEL

5.1 Definition of the Problem

The Western Electric Engineering Research Center at Princeton, New Jersey is engaged in the development of data collection systems for use within the Western Electric Company. The author was given the assignment of investigating the queueing problems involved with these systems. It was found that the systems often involved queueing problems which could not be solved by theoretical queueing models. After considerable study, it became evident that simulation was the best method of solution.

The following paragraphs summarize the reasons for using simulation as the method of solution. In general, the data collection systems make use of modified telephone handsets which are connected to a centrally located paper tape punch. The production personnel using the data collection systems often dial messages of variable digit length. This becomes necessary when the information to be entered is variable in nature. Inspection data is an example of variable data. One unit of production may have only one defect which would require a message a few digits in length. Another unit may have fifteen defects of different types which would require a message thirty or forty digits in length, depending on the defect code and message format. The service time for any particular user consists of the time required to seize the system, dial in the message, and release the system. Since the message lengths may be variable in length, the service time distribution for a particular class of user can only be determined empirically in many instances. The only

exception to this occurs when the message format consists of a constant number of digits. In this case, the service time distribution can be considered to be constant.

Also, the queueing problems become complex because of the many types of personnel using the system. Operators, inspectors, clerks, and supervisors may all be using a particular data collection system. Each particular type of user may have a different format or message length to enter and may request service at a different arrival rate. Although in some cases it might be possible to analyze the problem theoretically, the analysis would be complex and time consuming.

It was decided that a simulator should be general enough to be used for any type of data collection equipment. A simulator was designed and programmed for the IBM 1620. The general logic of the simulation program is covered in Chapter 6.

The initial application of the simulator was chosen to be the Key Equipment Shop of Western Electric's Kearny, New Jersey, Works Location. Since this system had been formulated and investigated in detail, it was also chosen as the application to be used in this study.

5.2 Description of the System

A data collection system had been designed and installed in the Key Equipment Shop. The data collected was to be used for studies in the area of improved utilization of manufacturing information. Initially, six wiring operators were given data collection equipment. Simulation was used to study the effect on the waiting time characteristics of the system as additional operators were given data collection equipment.

The Key Equipment Shop consists of a conveyor carrying units to be wired. Wiring operators are stationed at regular intervals around the conveyor. An operator removes an unwired unit from the conveyor and performs the necessary wiring operations. When the wiring operation is finished, the operator places the wired unit on the conveyor and removes another unwired unit. The wired units are then channeled to an extension of the conveyor for inspection.

The data collection equipment consists of a number of data entry sets connected to a central control unit. The central control unit consists of logic circuits which control the system and operate a paper tape punch. The system is designed so that any entry set demanding the control equipment will receive a busy indication if another set is using the equipment.

The entry set is a modified telephone handset which has the receiver removed. The handset is the type that has both a standard dial and a card reader for automatic dialing. The user has the option of hand dialing or inserting a prepunched plastic card. One entry set is shared by two wiring operators, while each inspector has his own

entry set. Several control buttons are provided on each set. There is a button for each operator to use when requesting service and a common button for releasing the system. A fourth button is used for error correction. A clock is included in the central control unit, which automatically punches the time on the paper tape at five-minute intervals.

An operator enters data into the system after finishing the wiring operation on each unit. At this time, the operator removes a prepunched plastic card from the unit, seizes the system, inserts the card in the entry set, and releases the system after the card is automatically read. The digits on the card identify the type of unit wired. The operator's set number is automatically punched by the control unit. Special cards are entered to report exceptions, such as hospital visits, etc. An inspector enters data into the system after inspecting each wired unit. The units are also designated by a tag marked with the wiring operator's number. The inspector enters this number which is followed by a series of digits denoting the type and quantity of defects found. The digits entered denoting quality are determined from a fixed defect code. Although the defect code is usually memorized by the inspector, it is visually available for easy reference. The defect code consists of a single digit denoting the type of defect followed by another single digit denoting the quantity of the defect. The digits zero through nine are assigned specific types of defects. If the quantity of a particular defect exceeds nine, the same message is redialed to report the number of defects in excess of nine.

At the termination of each shift, the paper tape is removed from the punch and fed into an IBM 1620 computer. The computer summarizes quality and computes efficiency using the clock entries as a time standard. Quality and efficiency reports are obtained as output from the computer and are used for improved management of the shop.

5.3 The Arrival and Service Time Distributions

The Key Equipment Shop has two distinct types of personnel using the data collection system. In addition, the clock requests service at a regular interval. As will be explained in Chapter 6, the simulation program allows arrivals from four sources. Any number of the sources may be used or not depending on the application. Also, the arrival distributions must be specified in units of "time between arrivals." The following table summarizes the types of requests for service and their respective distributions. The remaining paragraphs of this section explain in detail the formulation of these distributions.

Arrival Source	Type of Request For Service	Arrival Distribution (Time Between Arrivals)	Service Time Distribution
1	Clock	Constant (300 sec.)	Constant (2 sec.)
2	This arrival source was not used in this application.		
3	Operators	Exponential (Mean = 23.0 sec.)	Constant (6 sec.)
4	Inspectors	Exponential (Mean = 31.36 sec.)	Empirical (Table 9 - Appendix E)

The first step in preparing the data for the simulation consisted of determining the arrival distributions for each specific type of arrival. For the Key Equipment Shop, the clock was designated as source one. The arrival distribution for the clock was specified as

a constant arrival rate of 5 minutes or 300 seconds between arrivals. The second arrival source was not used in this application.

The third arrival source was specified as the wiring operators. It was hypothesized that the wiring operators would follow a Poisson arrival distribution and, therefore, the time between arrivals would be exponentially distributed. It was felt that this assumption was valid because operators work completely independent of each other. The assumption was checked by sampling the time between requests for the system in the shop. Two hundred samples were taken and a Chi-Square test for goodness of fit to an exponential distribution was run. Fourteen class intervals were used for this test. The class intervals were chosen to be as small as possible while still maintaining an expected frequency in any class of five or greater. The calculated χ^2 value was found to be 6.23, which is not significant when compared to $\chi^2_{.05,\ 12} = 21.03$. Thus the hypothesis of an exponential arrival distribution was accepted. The number of operators using the system was then arbitrarily set at 80. On the average, operators enter into the system 15 times/shift. Using these two figures, the mean time between arrivals for the exponential distribution was calculated and a cumulative arrival distribution was formed. These calculations are shown in Appendix E.

The fourth arrival source was specified as the inspectors. Since the inspectors also work independently, an exponential arrival distribution was assumed and tested in the same manner as explained above. The calculated χ^2 value was found to be 8.14, which is not significant. The inspector arrival distribution was formed in the same manner as the

operator distribution. The difference in the two distributions being that the inspectors enter only 11 times/shift on the average. The reduction of 4 entries/shift from the operator figure is due to special exception entries made by the operators. The calculations for the mean time between arrivals is also shown in Appendix E.

The second step in preparing the data consisted of determining the service time distributions for each specified type of arrival.

The clock service time distribution was specified as a constant value of two seconds. The entry sets require six seconds for reading the plastic card entered by the operators. This figure includes seizure and sign off time. Therefore, the service time distribution for the operators was specified as a constant value of six seconds.

The frequencies of the various digit lengths of inspector entries were found by searching six weeks of past data tapes for all inspector entries. This was done on the IBM 1620 and resulted in a printout of the frequencies of the entry lengths for approximately 36,000 entries. A time study was then conducted in the shop to determine the average dialing time for an inspector. It was observed that inspectors average one second for each digit dialed and four seconds for seizure and sign off. Using these figures, the inspector service time distribution was formed. The values for this distribution are tabulated in Table 15 of Appendix E.

The simulation model imposed only one limitation on the distributions mentioned above. The distributions had to be specified in even seconds in order to maintain consistent time units throughout the simulation.

6. THE SIMULATION PROGRAM

The overall logic of the simulation program is illustrated by Figure 2 of Appendix A. The program was written in the Symbolic Programming System Language for the IBM 1620. The program was then assembled using the Symbolic Programming System Processor Program (1620/1710 two pass paper tape version). A copy of the symbolic source program and the generated machine language program is included in a supplementary program appendix. A copy of the supplementary program appendix is on file in the Industrial Engineering Office, Lehigh University.

6.1 The General Methodology

The program allows arrivals from four sources designated as ONE, TWO, THREE, and FOUR. Storage areas are reserved for four separate empirical or theoretical arrival distributions and their respective service time distributions. All arrival distributions must be in the units, "time between arrivals." Any time unit may be used, depending on the application, as long as all distributions are specified using the same time unit. In this study, all distributions were specified in seconds. Any number of the four arrival sources may be used depending on the application. One, two, or three service stations or channels may be used in a particular simulation depending on the application. This study uses only one of the three available service stations. If more than one service station is used, the queue discipline consists of all service stations serving a single waiting line. Each service station has an associated storage area designated BUSY1, BUSY2, and BUSY3. When an arrival is serviced by one of the

channels, the service time of the arrival is stored in one of these three locations depending on which channel is free. The units of BUSY1, BUSY2, and BUSY3 then become the same as the time units specified for the distributions.

The general logic of the program is sometimes referred to as the "master clock method." A storage area designated CLOCK is continually incremented by one time unit. After each time CLOCK is incremented, each source is checked for the occurrence of an arrival. If an arrival has occurred, the service time of the arrival and a new arrival time are found by Monte Carlo sampling. The arrival is then either assigned a service station or put in the waiting line. All pertinent data, such as waiting time and length of waiting line, is recorded and CLOCK is incremented again. Each time CLOCK is incremented, BUSY1, BUSY2, and BUSY3 are decremented by one time unit. Thus, a negative "busy time" indicates that the service station is empty.

6.2 Subroutine Descriptions

A majority of the blocks shown in Figure 2 consist of an entire subroutine. These subroutines will be explained in detail in the following paragraphs. Whenever possible, the subroutines will be explained in the same order as they occur in the flow diagram.

6.21 Subroutine for Determining Arrival Times

The subroutine for determining arrival times is entered with a value of one, two, three or four depending on the source for which the arrival time is desired. A random number is generated and the arrival distribution for the specified source is sampled, thus obtaining an arrival time. The arrival time is then added to storage areas ATIME1, ATIME2, ATIME3, or ATIME4 depending on the specified source. These storage areas then contain the time of the next arrival from each source.

6.22 Subroutine for Checking for Arrivals

This subroutine compares the simulation time (CLOCK) to the arrival times for each source (ATIME1, ATIME2, ATIME3, and ATIME4). If the arrival time of a particular source is equal to the simulation time, the digit one, two, three, or four is stored in the proper source designation (ONE, TWO, THREE, or FOUR). An arrival indicator is then turned on by storing the digit one in the location designated CHECK. The main program can then interrogate CHECK to determine if there has been an arrival during the present time unit. If CHECK indicates that there has been an arrival, the main program can then determine which source has arrived by interrogating storage locations ONE, TWO, THREE, and FOUR.

6.23 Subroutine for Determining Service Times

The subroutine for determining service times is basically the same as the arrival time subroutine. The subroutine is entered with a value of one, two, three, or four depending on which source requires a service time. The service time is then stored in STIME1, STIME2, STIME3, or STIME4, depending on the specified source.

6.24 Subroutine for Checking for the Presence of a Queue

Storage area is reserved for forming a waiting line in the computer memory. A one, two, three, or four is placed in the line depending on the source of the particular arrival. The service time and the waiting time are both stored along with the source designation for each person in line. This line of storage locations is moved and adjusted in the same manner as the actual waiting line would fluctuate. This subroutine searches the waiting line storage locations until the end of the line is found. If a queue is present, the address of the end of the waiting line is stored. If there is no queue, the indicator QCHECK is set to a specific number. QCHECK can then be interrogated by the main program for the presence of a queue.

6.25 Subroutine for Establishing Random Arrival Order

This subroutine is entered only when there is more than one arrival during any given time unit. For example, in certain simulations, it could be possible to have the simultaneous occurrence of arrivals from all four sources. When more than one arrival occurs, the order in which the arrivals are placed in the waiting line is determined randomly. This is done by assigning each source one-fourth of all the possible random numbers. A random number is generated and

the corresponding source is designated as first. This process continues until an arrival order has been established.

6.26 Subroutine for Placing Arrivals in the Waiting Line

After one or more arrivals have occurred and an arrival order has been determined, the arrivals are placed in the waiting line by this subroutine. The designation of the first arrival is stored at the end of the waiting line. This is done by using the address of the end of the waiting line which was previously stored in location QCHECK. The service time of the arrival is also stored, along with the source designation. This process is continued until all the arrivals have been placed in the waiting lines in the order previously established by the random arrival order subroutine.

6.27 Subroutine for Checking for Free Channels

This subroutine compares BUSY1, BUSY2, and BUSY3 to zero. The first service station that is found to have a negative storage location is then known to be free. The address of its storage location is then stored. The main program can then store the service time of the first person in the waiting line in the storage location for the free channel. After the service time is stored in the free channel, the waiting time for the first person in line is accumulated and the waiting line is shortened by one person.

6.28 Random Number Generator Subroutine

The random number generation subroutine consists of only reserved storage space when the object program is read into memory. Because there were four different methods of generating random numbers used in this study, the generation subroutine is read in as an input. This

eliminated the necessity of writing and assembling four different simulation programs. The generation subroutine must be written in machine language and entered as an input. Instructions for preparing the generation subroutine and other input tapes are included in the instructions for program operation shown in Appendix B.

7. DESCRIPTION OF THE EXPERIMENT

7.1 The Random Number Generator Tests

The initial phase of the experimental design consisted of the selection of the pseudo-random sequences to be used in the simulation of the data collection equipment. Chapter 3 shows that $\chi^2_{\alpha,9}$ is the critical chi-square value for the goodness of fit, independence, and run tests. A level of significance of five percent was chosen for the statistical tests of randomness. Thus, the critical value for all three tests of randomness is $\chi^2_{0.05,9}=16.92$. This level of significance was used throughout the experiment. The value of n, the quantity of random numbers tested in each block, was chosen as 1000. The restriction that the cell frequency for the chi-square test should be greater than five was the only consideration used in choosing this value. The question of the effect of sample size was not considered in this study.

Since sequences with varying degrees of nonrandomness were desired, classes of starting values were tested that have been previously shown to have poor statistical properties. Even with apriori knowledge of these classes of starting values, approximately 70 hours of IBM 1620 time was required to isolate sequences with suitable properties. Table 1 of Appendix C summarizes the statistical testing of the four methods of random number generation.

It is general knowledge in the field of random number generation that a small multiplier (a) will result in poor statistical properties. Consequently, small (a) values were chosen for the statistical tests of the multiplicative congruential method. The value of a = 100077

was chosen as the statistically acceptable multiplier for the multiplicative method. This multiplier had been tested by the author previous to this study. Table 1 of Appendix C shows the results of the statistical tests for a = 13, 21, and 100077.

The work of Allard, Dobell, and Hull [1] was extremely helpful in isolating both acceptable and unacceptable sequences using the mixed congruential method. Their results indicated that most poor multipliers are congruent to 1 (Mod 500). Thus, the values of a = 2501, 4001, and 5001 were chosen as possible poor multipliers. Their results also single out a = 101 as an acceptable multiplier. Table 1 shows the results of the statistical tests for a = 2501, 4001, 5001, and 101.

The testing of the mid-square method proceeded on a more haphazard basis. It has been shown in [7] that the mid-square method cannot be analyzed theoretically. The choice of a starting random number is arbitrary and the possibility of a degenerate sequence exists with any starting number. Three starting numbers were arbitrarily chosen and statistically tested. Table 1 shows the results of these tests. The first two sequences tested degenerated to zero while the third sequence remained stable for one hundred thousand numbers. However, it is possible that the stable sequence cycled a number of times while the one hundred thousand numbers were being generated. Cycling in the mid-square method is impossible to determine without using unreasonable amounts of computer time.

The starting value for the modification of Lehmer's method is determined by the method itself and thus only one test was necessary.

The results of this test are also shown in Table 1. It was found that the sequence cycles at n = 9000 for any starting value. In spite of the cycling, the sequence was still used in the experiment. The sequence was not eliminated because the opportunity for the study of the effect of known cycling was thought to be important.

7.2 The Simulation Phase of the Experiment

The six pseudo-random sequences marked with an asterisk in Table 1 were chosen for the simulation runs. The simulation phase of the experiment was then designed using these six sequences as a basis. The following results were chosen for statistical analysis:

1. Average waiting time.
2. Average length of waiting line.
3. Percentage of arrivals finding the system busy.
4. Maximum waiting time for any arrival.
5. Maximum length of waiting line.

These results were chosen because it was felt that they were representative of the results desired in most waiting line simulations.

The data collection system parameters remained fixed throughout the experiment. The values of these parameters were previously covered in Chapter 5. The sample size for one simulation was then fixed at 10,000 arrivals and 20,000 waiting lines observed. The simulation program outputs the results when each of these two sample sizes are reached. This results in equal sample sizes for each independent replication of a particular simulation. The particular sample sizes resulted in approximately two hours computer time for each simulation. The values of 10,000 and 20,000 were chosen because estimates of the standard deviation of the results were available from previous simulations at these sample sizes.

It was initially planned to use multiple "t" tests to compare the results of the simulations. All of the results obtained with

statistically poor sequences were to be compared with the results using the mixed method with a = 101. Thus, the simulation answers obtained with a = 101 were to be used as a standard. However, it was decided that any standard would only be hypothetical. The selection of a standard necessitates the assumption that the use of the standard results in correct or true simulation answers.

As a result, it was decided to use analysis of variance for simulation results that could be assumed to be normally distributed. Individual comparisons could still be made using Scheffe's S-method of judging contrasts [22]. Through the use of the S-method, comparisons between simulation results can be made without setting up a hypothetical standard.

Once the length of the simulation was fixed and the method of analysis chosen, the number of independent simulations or replications using a particular sequence was determined. Table 2 of Appendix C illustrates the experimental design for the multiplicative congruential method. The sample size calculations were made on an individual comparison basis and are shown in Appendix D. Individual comparisons were used because the S-method has less power than the overall analysis of variance. Therefore, a sample size chosen on the basis of individual comparisons would be more than sufficient for the analysis of variance test for a given Type Two Error. In making the sample size calculations, the maximum sample variance of all the results was estimated. This estimate was then used to calculate the difference in means (θ) that is detectable for an increasing number of replications (r). The computer time necessary for five replications was then calculated.

The computer time was found to be approximately the maximum time that was economically feasible. The corresponding value of θ was then examined and was found to be sufficient for this study. As a result, the number of replications was chosen as five.

Analysis of variance and Scheffe's S-method are valid for the average waiting time and the average length of waiting line because the Central Limit Theorem assures a close approximation to normality at the large number of waiting times and waiting lines observed in each simulation. Analysis of variance can also be used for a binomial variable (P) if the transformation 2 arcsine $\sqrt{P}$ is used to obtain approximate normality. The arcsine transformation was used so that analysis of variance could also be applied to the percentage of arrivals finding the system busy.

A preliminary test for homogeneous variances was planned for the results that were assumed to be normally distributed. The classical test for homogeneous variances is Bartlett's Test. It was decided that this test would be used, although the standard χ^2 approximation to the M statistic would not be used. When the n_i are small, this approximation can introduce considerable error [2]. Since this test is critical to the entire analysis, Table 5.10 in [2] was used for the maximum values of the percentage points of M.

A number of authors in the field of statistical testing object to a preliminary test for homogeneous variances. The objection is usually made because Bartlett's Test could show unequal variances when actually the variances are homogeneous. In this case, the analysis of variance test would not be run when it actually applies. However,

Snedecor proposes a partial solution to this problem in [23]. An approximate analysis of variance can still be applied when there is an indication that the variances are unequal. This test is based on weighted mean squares. Admittedly, the level of significance will change somewhat due to the preliminary test. However, the risk taken is somewhat lessened by using the approximate analysis of variance method if the variances are shown to be unequal. It was felt that the additional information obtained far offsets the increased risk.

Since nothing could be assumed about the distribution of the maximum waiting time and maximum length of waiting line, a nonparametric test was necessary. The Kruskal-Wallis One Criteria Variance Analysis Test [14] was chosen because of its similarity to analysis of variance. It is essentially a nonparametric one-way analysis of variance because it applies when it is desired to detect a difference in means for K independent samples. No attempt was made to calculate the sample size on the basis of this test because little is known about the power of the test.

8. ANALYSIS OF THE EXPERIMENT

The simulation phase of the experiment consisted of five independent simulations using each of the six pseudo-random sequences chosen for the study (Table 1 of Appendix C). The simulations were run and the desired results were obtained without any serious difficulties. Since each independent simulation required approximately two hours, the total computer time for the experiment was approximately sixty hours. The simulation results that were obtained using each of the biased pseudo-random sequences are shown in Tables 2 thru 7 of Appendix C.

The average waiting time results for each sequence are shown in Table 8. Bartlett's Test and analysis of variance were applied to the average waiting time results giving the values M = 7.36 and F = 1.46 respectively. The average length of waiting line results are shown in Table 9. These results were similarly analyzed giving the values M = 5.19 and F = 1.22. The results for the percentage of arrivals finding the system busy are shown in Tables 10 and 11. The same two tests were applied to these results giving values of M = 10.86 and F = 1.29.

The maximum waiting time results for each sequence are shown in Table 12. The nonparametric variance analysis test was applied to the maximum waiting time results giving the value H = 5.05. The maximum length of waiting line results are shown in Table 13. The nonparametric test was also applied to these results giving the value H = 4.80. Examples of each method of statistical analysis are included in Appendix D.

If all of the test statistic values given above are compared with

their critical values at a 5% level of significance, it can be seen that they are insignificant in all cases. This shows that the effect of the biased pseudo-random sequences on the simulation results was statistically insignificant in all cases. A summary of the values of the test statistics is included in Appendix D.

Table 14 of Appendix C shows a summary of the means and variances of the simulation results. Examination of Table 14 gives some insight into areas for further study. By examining the column corresponding to the modification of Lehmer's method, it can be seen that each type of result has its smallest mean in this column. Although the experiment does not show this to be statistically significant, the cycling of the generator could have caused this effect.

Also, for each type of result, the largest sample variance is found in the column corresponding to the mixed method with unacceptable statistical properties. This generator differs from the other generators in that its goodness of fit chi-square value (χ^2_A) is extremely high. Again, although it is not statistically significant, poor goodness of fit could have caused the high sample variances. It should be emphasized that these two points are only conjectures and would require further testing before anything conclusive can be stated.

A possible explanation of insignificant effects could be connected with the simulation subroutine that randomizes the arrival order. This subroutine could be thought of as a "scrambling routine." Each time the subroutine is used, an unpredictable quantity of random numbers is used. If any pattern is present causing a random number to be used for the same purpose at regular points in the sequence, this pattern is

eliminated by the subroutine. If a pattern exists, poor serial correlation properties would probably have their greatest effect. It is possible that the "scrambling routine" eliminated effects of serial correlation. Again, further testing would be necessary to make any definite conclusions about this point.

After examining the summary of means and variances for possible effects, the results of the experimental runs (Tables 2 thru 7) were analyzed for the consistency of results within a particular replication. Each replication was examined to determine if the results are of approximately the same relative magnitude. This was done by ranking the results in each column and comparing the ranks for each replication. If the results for the multiplicative method (Table 2) and the mixed method (Table 4) are examined, it can be seen that the results are surprisingly consistent. It is important to notice that these results were obtained with the only pseudo-random sequences that have acceptable chi-square values for all three statistical tests. If the results for the mid-square method (Table 6) and the modification of Lehmer's method (Table 7) are examined, it can be seen that the ranks of the results are extremely inconsistent for certain replications. Replications 3 and 5 in Table 6 and replications 2 and 4 in Table 7 have inconsistent ranks between the columns representing average results and maximum results. This analysis indicates that acceptable pseudo-random sequences produced the most consistent results for the particular application studied.

9. CONCLUSIONS

The purpose of this study was the investigation of the effect of biased pseudo-random numbers on simulation results. Throughout the study it was felt by the author that a significant effect would be found and that the results would indicate the magnitude of the effect.

The effect was studied by introducing random number generators with varying degrees of statistical bias into the simulation of a data collection system. The results of the simulations were then statistically analyzed to determine the effect of the biased random number generators. The author was surprised to find that the effect of pseudo-random number bias was statistically insignificant in this particular application.

10. RECOMMENDATIONS

The general opinion that pseudo-random number bias has a significant effect on simulation results seems to prevail in the field of systems simulation and random number generation. As was previously mentioned, [10] contains a bibliography of 143 articles pertaining to the methods and theory of pseudo-random number generation. These articles represent a substantial expenditure for research in this field. The results of this study indicate that a portion of any future research effort in this field should be spent in the area of the effect of biased pseudo-random numbers.

Investigators should consider the following points in future studies:

1. This study indicates that different types of simulation models may have entirely different sensitivities to random number bias. Several authors [1, 10,11, 19] have mentioned this point prior to this study.
2. Cycling within a pseudo-random sequence may have a significant effect on simulation results.
3. Poor goodness of fit properties may have a significant effect on the variance of simulation results. An attempt could be made to isolate pseudo-random sequences which are statistically unacceptable only because of goodness of fit. These sequences could then be introduced into a simulation model and the results statistically tested using the Model II analysis of variance method.

4. Simulations that require only a small quantity of random numbers may be significantly effected by pseudo-random number bias.

5. If a model is selected that can be solved analytically, a standard for comparing results would then be available.

6. Finally, this study indicates that a subroutine to discard random numbers might prove valuable in eliminating serial correlation effects within a simulation.

APPENDIX A

Program Flow Diagrams

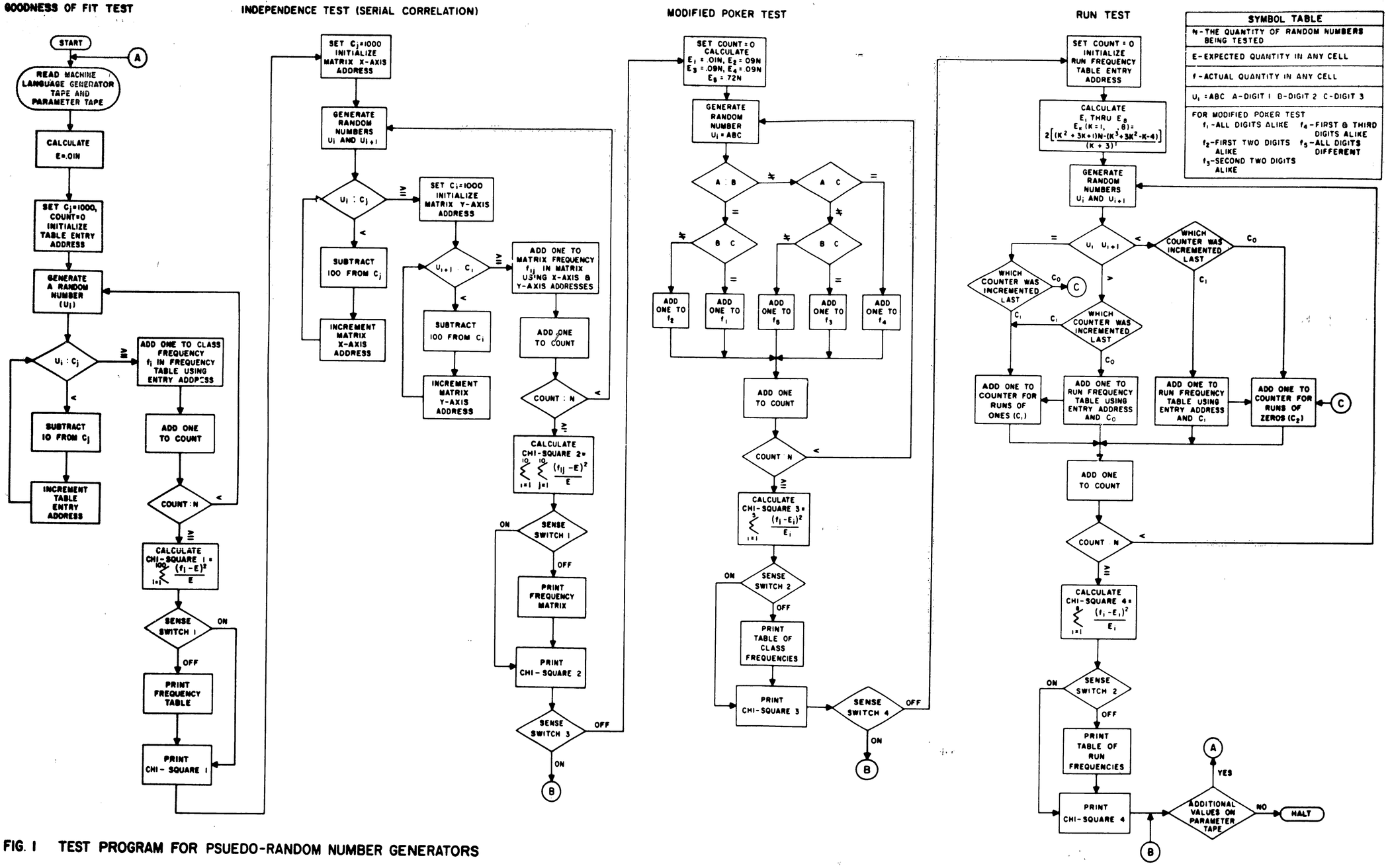


FIG. 1 TEST PROGRAM FOR PSUEDO-RANDOM NUMBER GENERATORS

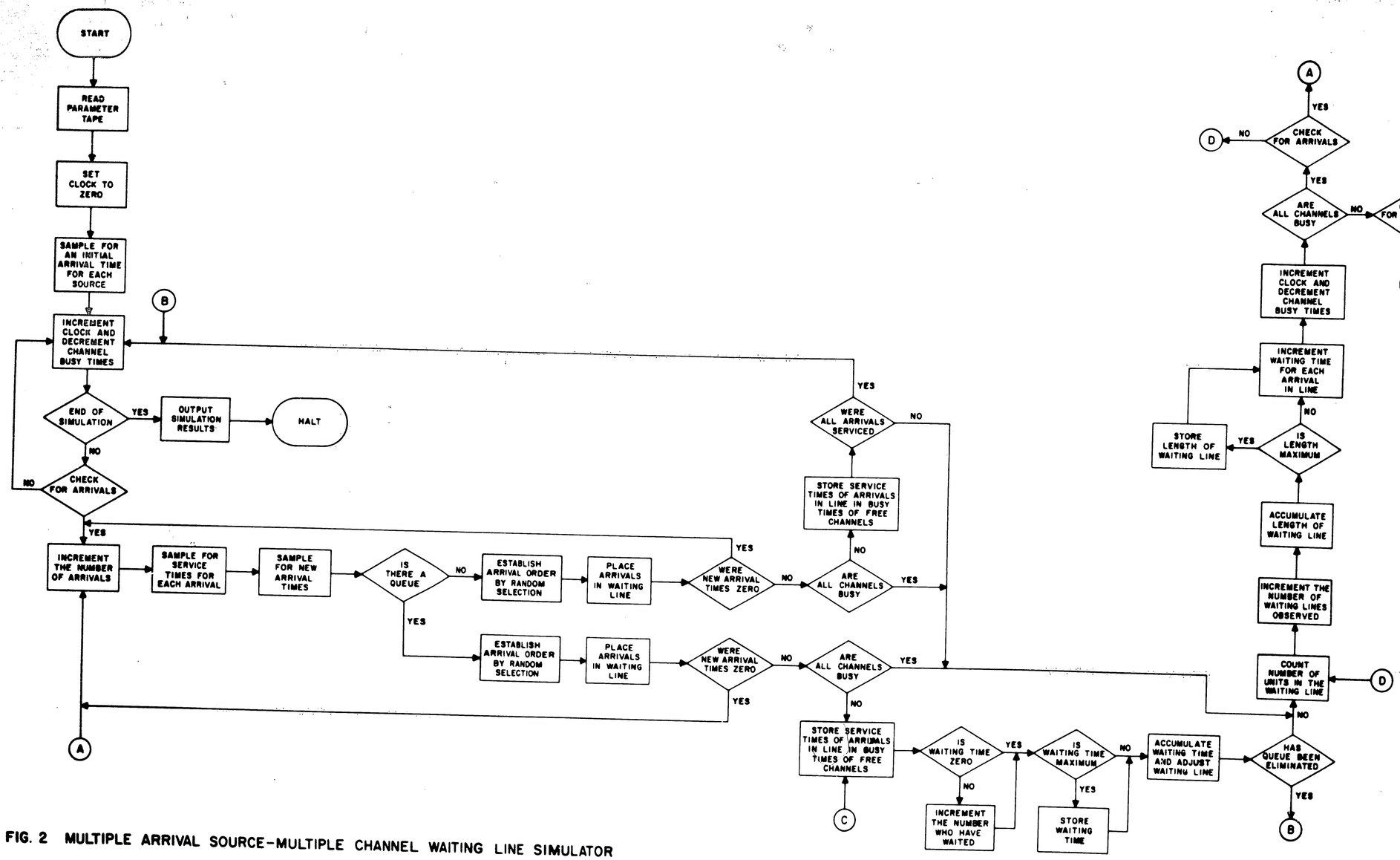


FIG. 2 MULTIPLE ARRIVAL SOURCE-MULTIPLE CHANNEL WAITING LINE SIMULATOR

APPENDIX B

Instructions for Program Operation

RANDOM NUMBER GENERATOR TEST PROGRAM

Program Operating Instructions

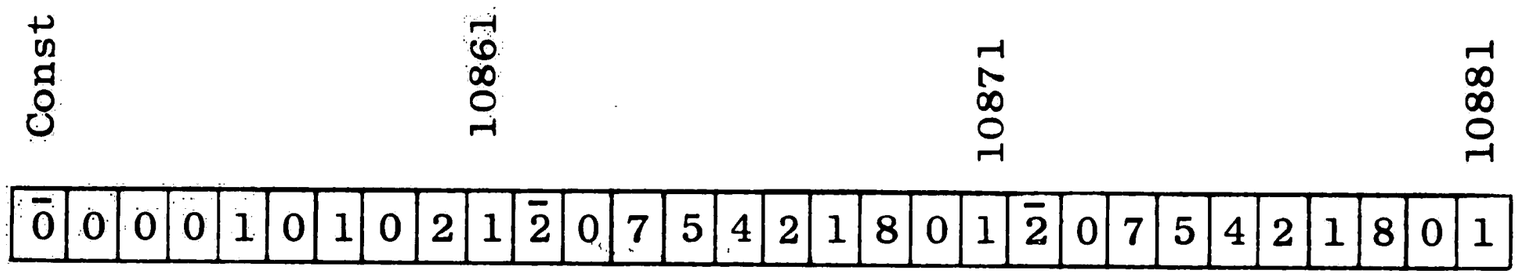
The general logic of the random number generator test program is covered in Chapter 4 and illustrated by Figure 1. The generality of the program is made possible by requiring that the generation subroutine be an input rather than an integral part of the program. Operation of the program requires that the generation subroutine be written in machine language and punched on paper tape. The subroutine is then read into location RAND + 1 during the initializing portion of the program. There are 1000 storage locations reserved at location RAND + 1 and successively higher locations for the subroutine.

A second machine language initializing subroutine is also required when more than one block of random numbers is tested from one sequence. This subroutine stores the last number in the block so that the sequence can be continued when the program branches back to start the series of tests over again. Memory location ISEED + 1 and the 499 successively higher memory locations are reserved for this subroutine. A third machine language initializing subroutine is required because each test requires the complete regeneration of the block of numbers being tested. The program branches to this subroutine before each test and the subroutine reinitializes the sequence to its starting number. Memory location SEED + 1 and the 499 successively higher memory locations are reserved for this subroutine. The second and third subroutines mentioned above are punched on the same input tape as the machine language generation subroutine.

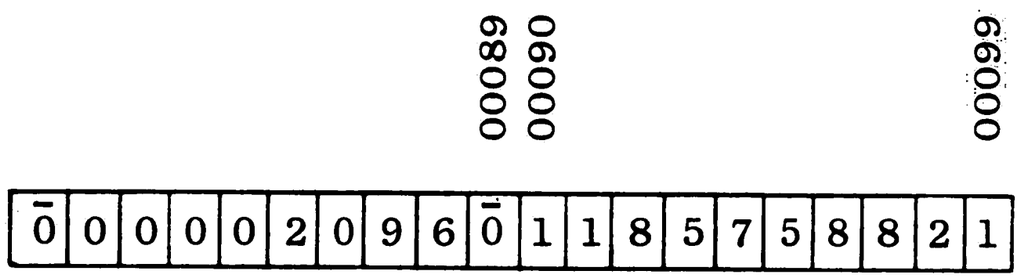
The preparation of the generation and initializing subroutine tape requires general knowledge of IBM 1620 machine language. This tape is referred to in the program as the generator tape. An example showing the necessary machine language instructions for the multiplicative congruential method will better explain the preparation of the generation and initializing subroutines.

Example: L = 10; a = 101021; X_o = 2075421801

The following starting values are read into location CONST from the parameter tape. (The preparation of the parameter tape will be explained later.)



The following memory locations represent the product area for the 1620:



The required machine language instructions are as follows:

(1) 23 10861 10881 Multiply the field at location 10881 by the field at location 10861 and store the product in location 00099.

(2) 32 00090 00000 Set a flag at location 00090.

(3)	26	10881	00099		Store the field at location 00099 in the field at location 10881.
(4)	33	00090	00000		Clear the flag at location 00090.
(5)	15	00089	00000		Store the digit $\bar{0}$ in location 00089.
(6)	26	01818	00092		Store the field at location 00092 in the field at location 01818.
(7)	42	00000	00000	E.O.L.	Branch back to the next sequential instruction.
(8)	26	10871	10881		Store the field at location 10881 in the field at location 10871.
(9)	42	00000	00000	E.O.L.	Branch back to the next sequential instruction.
(10)	26	10881	10871		Store the field at location 10871 in the field at location 10881.
(11)	42	00000	00000	E.O.L.	Branch back to the next sequential instruction.

Lines (1) through (7) make up the machine language generation subroutine. During the initializing stage of the program, the constant multiplier (a) and the starting random number (X_o) are read into location CONST from the parameter tape. A duplicate of X_o is included for initializing purposes. When a random number is required, the main program branches to instruction (1). Instruction (1) begins by multiplying (a) by (X_o), forming a twenty digit product. The resulting product is shown above. Instructions (2) and (3) store the right hand ten digits of the product as the new X_1. The program is designed to

test three digit random numbers in the form $\bar{0}$XXX. The zero must be added as the first digit for ease of comparison to four digit numbers. Instructions (4), (5), and (6) form and store the required four digit number in memory location STORE (01818). Instruction (7) branches back to the next sequential instruction in the main program.

After the series of four tests are performed, a subroutine is needed so that the next number in the sequence can be stored and used as the starting value for a new series of tests. Instructions (8) and (9) make up this subroutine. Instruction (8) stores the next number in the sequence in the location for the duplicate value of X_o. Instruction (9) branches back to the main program.

Instruction (10) and (11) make up the initializing subroutine required before each test. Instruction (10) stores the duplicate value of X_o in the storage location for X_i. Thus, the sequence being tested is started at the beginning for each test. Instruction (11) branches back to the main program. Instructions (1) through (11) are punched along with the E.O.L. characters on paper tape to make up the generator tape.

A second input tape is required and is referred to in the program as the parameter tape. This tape contains the following information:

1. A fixed length three digit number representing the number of blocks being tested in the particular sequence. The number appears once for each sequence being tested and is the first of the numbers representing a particular sequence.

2. The second number representing a particular sequence is the quantity of numbers being tested in each block. This number is of variable length and can be from three to six digits long.

3. The third number representing a particular sequence — contains the initializing fields for the random number generator subroutine. i.e., (a) and (X_o) for the multiplicative congruential method. This number is of variable length and can be any length up to 100 digits. The length and structure of the number is determined by the machine language generation subroutine.

An example showing the parameter tape for testing two sequences generated by the multiplicative congruential method will better illustrate the preparation of this tape. Fifty blocks of one thousand numbers each will be tested for each sequence.

Example: Sequence 1 - a = 101021, X_o = 2075421801

Sequence 2 - a = 110011, X_o = 2075421801

The parameter tape is prepared as follows:

(1) 02 E.O.L.	Two sequences being tested.
(2) 050 E.O.L.	Fifty blocks in sequence 1.
(3) 1000 E.O.L.	One thousand numbers per block.

(4) $\bar{0}0001010212\bar{0}7542180\bar{1}2075421801$ E.O.L. Initializing fields for sequence 1.

(5) 050 E.O.L.

(6) 1000 E.O.L.

(7) $\bar{0}0001100112\bar{0}7542180\bar{1}2075421801$ E.O.L.

Lines (1) through (7), along with the end of line characters, are punched on paper tape to make up the parameter tape. Lines (2) through (4) represent sequence 1 and lines (5) through (7) represent sequence 2. Line (1) represents the number of sequences being tested and can be any number up to a maximum of 99 sequences. The storage location CONST and the 99 successively higher memory locations are reserved for the initializing fields.

After the generator and parameter tapes are prepared, the program is run using the following steps:

1. Determine the program options desired from the following table:

SWITCH	ON	OFF
1	Tabular Printouts of Tests 1 and 2 are suppressed.	Complete Tabular Results of Tests 1 and 2 are typed out.
2	Tabular Printouts of Tests 3 and 4 are suppressed.	Complete Tabular Results of Tests 3 and 4 are typed out.
3	Test 3 is completely eliminated.	Test 3 is performed.
4	Test 4 is completely eliminated.	Test 4 is performed.

2. If tabular printouts of any of the tests are desired, set the margins 85 spaces apart. Next, set nine tab

stops every eight spaces starting from the left-hand margin. If chi-square values only are being typed out, no tab settings are necessary. Also, set the typewriter for single spacing.

3. Clear memory and read in the object program.

4. After the program has been read in, press START and the message LOAD GENERATOR TAPE will be typed out. Load the generator tape and press START. The tape will be read and the message LOAD PARAMETER TAPE will be typed out. Load the parameter tape and then position the paper so that the next line to be typed will be the first line on a page. Press START and the program will make the tests and type out each set of results exactly on one page. However, the paper settings are only necessary when complete tabular printouts are required.

MULTIPLE ARRIVAL SOURCE-MULTIPLE CHANNEL

WAITING LINE SIMULATOR

Program Operating Instructions

The general logic of the simulation program is covered in Chapter 6 and illustrated by Figure 2. The program was generalized for this study because a different generation subroutine was needed for each of the four methods of random number generation being considered. This was done by requiring that the generation subroutine be an input rather than an integral part of the program. Operation of the program requires that the generation subroutine be written in machine language and punched on paper tape. The subroutine is then read into RAND + 1 during the initializing portion of the program. There are 120 storage locations reserved at location RAND + 1 and successively higher memory locations for the subroutine.

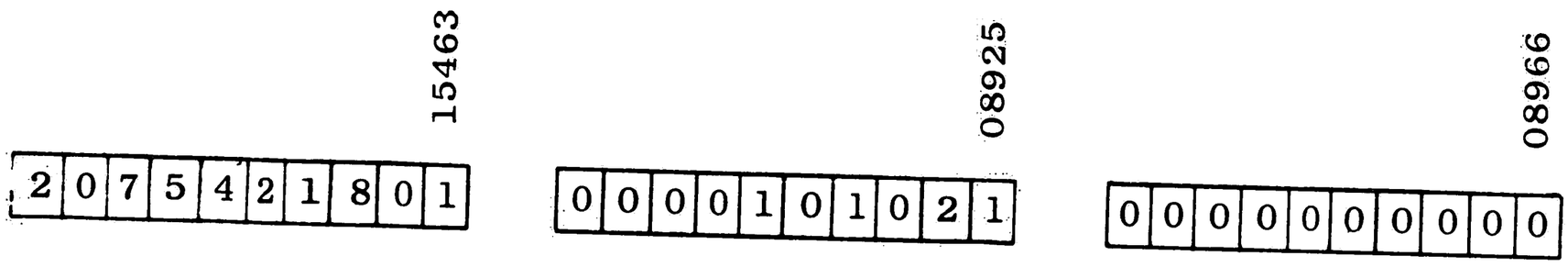
During the initializing portion of the program, three ten digit numbers for use in the random number generation subroutine are read from the parameter tape. The preparation of the parameter tape will be explained in detail later in this section. The first of the three numbers is the starting random number (X_o). This number is read into location NO - 9. The number is then stored in location RAND (15463) thus initializing the generation subroutine. The second number read is the constant multiplier (a). This number is read into location CONST - 9. The third number read is the additive constant (c). This number is read into location ADDIT - 9. The field locations of these three numbers are then NO, CONST, and ADDIT. Their respective

numerical field locations are 08914, 08925, and 08966. If any of the three numbers are not used for a particular generation method, they must still be punched on the parameter tape as zero. The location STORE (08867) is reserved for the random number after it is generated. The generation subroutine is required to place each generated number in location STORE. Location SAVE (08992) is reserved for branch back purposes after the generation subroutine is completed. Before branching to the generation subroutine, the main program stores the address of the next sequential instruction in SAVE.

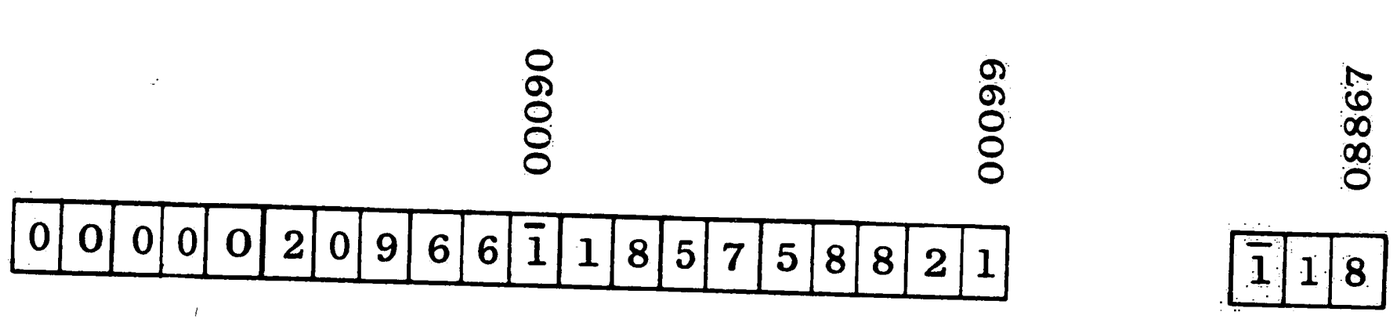
The preparation of the generation subroutine requires general knowledge of IBM 1620 machine language. The following example shows the preparation of the generation subroutine for the multiplicative congruential method.

Example: $L = 10$; $a = 101021$; $X_o = 2075421801$

The starting values are read into their respective locations from the parameter tape.



The following memory locations represent the product area for the 1620 and location STORE.



The required machine language instructions are as follows:

(1)	23	08925	15463	Multiply the field at location 08925 by the field at location 15463 and store the product in location 00099.
(2)	32	00090	00000	Set a flag at location 00090.
(3)	26	15463	00099	Store the field at location 00099 in the field at location 15463.
(4)	26	08867	00092	Store the field at location 00092 in the field at location 08867.
(5)	49	08992	00000 E.O.L.	Branch back to the next sequential instruction.

Lines (1) through (5) make up the machine language generation subroutine. When the main program requires a random number, it branches to instruction (1). Instruction (1) multiplies (a) by (X_o) forming a twenty digit product. Instructions (2) and (3) store the right hand ten digits of the product as the new X_1. Instruction (4) stores the three digit random number in location STORE (08867). Instruction (5) branches back to the next sequential instruction in the main program.

A second input tape is required and is referred to in the program as the parameter tape. This tape contains the following information:

1. A fixed length two digit number representing the number of simulations being run. This number appears only once as the first number on the tape. All other numbers on the parameter tape appear once for each simulation being run.

2. A fixed length two digit number representing the simulation number. When more than one simulation is being run, any particular simulation with a simulation number other than 01 does not require arrival and service time distributions. In other words, simulation numbers not equal to 01 eliminate the necessity for repeating the same distributions when a number of replications of the same simulation are being run.

3. A fixed length six digit number representing the simulation sample size in arrivals. When the number of arrivals equals this number, a table of simulation results is printed.

4. A fixed length six digit number representing the simulation sample size in number of waiting lines observed. A table of simulation results is printed when the simulation reaches this sample size.

5. A fixed length eight digit number representing the simulation sample size in time units. A table of simulation results is printed when the simulation reaches this sample size.

6. A fixed length ten digit number representing the starting random number (X_o).

7. A fixed length ten digit number representing the constant multiplier (a).

8. A fixed length ten digit number representing the additive constant (c).

9. A variable length number representing the arrival distribution for source one. This number is made up of a series of

two alternating fields. The first field in the alternating sequence is a five digit field equaling the time between arrivals. The second field is a three digit field equaling the cumulative probability that corresponds to the time between arrivals. This will be illustrated by an example later in this section. The number can be of any length up to a maximum of 1600 digits. All arrival distributions are of this form and can be any length up to 1600 digits.

10. Arrival distribution for source two.

11. Arrival distribution for source three.

12. Arrival distribution for source four.

13. A variable length number representing the service time distribution for source one. This number is made up of a series of two alternating fields. The first field in the alternating sequence is a three digit field equaling the service time. The second field is another three digit field equaling the cumulative probability corresponding to the service time. This will also be illustrated by an example later in this section. The number can be of any length up to a maximum of 360 digits. All service time distributions are of this form but have different maximum lengths.

14. Service time distribution for source two. The number can be of any length up to 240 digits.

15. Service time distribution for source three. The number can be of any length up to a maximum of 120 digits.

16. Service time distribution for source four. The number can be of any length up to a maximum of 60 digits.

17. A fixed length nine digit number which is read into BUSY1. If the number (000000000) is used, channel one is available for use in the simulation. If the number (999999999) is used, BUSY1 will never go negative and thus channel one is blocked from the simulation. The same logic applies to channel two and three.

18. A fixed length nine digit number which is read into BUSY2.

19. A fixed length nine digit number which is read into BUSY3.

In addition to the choice of the number of service stations or channels to be used in a simulation, any of the arrival sources may be blocked or eliminated from a simulation. This is accomplished by using the following distributions for the source being eliminated:

Arrival distribution; enter 999990009999999 E.O.L.

Service time distribution; enter 000000000999 E.O.L.

An example of the parameter tape for a hypothetical simulation will help to clarify the preparation of this tape.

Example: Suppose it is desired to run two independent samples of a system with two arrival sources and two service stations. Also, suppose that the multiplicative congruential method of random number generation is being used and the following values have been formulated.

Results are desired when:

Number of arrivals = 1000

Number of waiting lines observed = 2000

Simulation time = 10,000

Source one arrival distribution:

Time Between Arrivals	Cumulative Probability
100	.250
200	.555
300	.999

Source one service time distribution:

Service Time	Cumulative Probability
20	.100
21	.675
22	.999

Source two arrival distribution:

Time Between Arrivals	Cumulative Probability
500 (constant)	.999

Source two service time distribution:

Service Time	Cumulative Probability
30 (constant)	.999

Random number generator starting values:

X_o = 2075421801 (first simulation)

X_o = 6703429087 (second simulation)

a = 101021

The parameter tape is prepared as follows:

(1) 02 E.O.L. Two independent simulations being run.

(2) 01 E.O.L. Simulation number one.

(3) 001000 E.O.L.	Output the results when the number of arrivals equals 1000.
(4) 002000 E.O.L.	Output the results when the number of waiting lines observed equals 2000.
(5) 00010000 E.O.L.	Output the results when the simulation time equals 10,000.
(6) 2075421801 E.O.L.	Starting random number.
(7) 0000101021 E.O.L.	Constant multiplier.
(8) 0000000000 E.O.L.	The additive constant is not used.
(9) 0010000000100250 0020055500300999 E.O.L.	Source one arrival distribution.
(10) 0050000000500999 E.O.L.	Source two arrival distribution.
(11) 9999900099999999 E.O.L.	Source three is blocked.
(12) 9999900099999999 E.O.L.	Source four is blocked.
(13) 020000020100021675022999 E.O.L.	Source one service time distribution.
(14) 030000030999 E.O.L.	Source two service time distribution.
(15) 000000000999 E.O.L.	Source three is blocked.
(16) 000000000999 E.O.L.	Source four is blocked.
(17) 000000000 E.O.L.	Channel one is in use.

(18) 000000000 E.O.L.

(19) 999999999 E.O.L.

(20) 02 E.O.L.

(21) 001000 E.O.L.

(22) 002000 E.O.L.

(23) 00010000 E.O.L.

(24) 6703429087 E.O.L.

(25) 0000101021 E.O.L.

(26) 0000000000 E.O.L.

(27) 000000000 E.O.L.

(28) 000000000 E.O.L.

(29) 999999999 E.O.L.

Channel two is in use.

Channel three is blocked.

Simulation number two.

Lines (21) through (29) contain the data for simulation two.

Lines (1) through (29) along with the end of line (E.O.L.) characters are punched on paper tape to make up the parameter tape. Lines (2) through (19) represent the first simulation, while lines (20) through (29) represent the second simulation.

After the generator and parameter tapes are prepared, the program is run using the following steps:

1. Clear memory and read in the object program.
2. After the program has been loaded, press START and the message LOAD PARAMETER, POSITION PAPER, PRESS START will be typed out. Since the inclusion of the random number generator as an input was a last minute program change, the generator tape must be loaded before the parameter tape.

3. Load the generator tape and press START. The tape will be read and the READER NO FEED indicator will appear. Load the parameter tape and position the paper six lines from the top of the page. The typewriter should be set for single spacing. Press START and the simulations will be run and the results will be typed out exactly on one page.

APPENDIX C

Experimental Results

	Method of Generation	Starting Values	Goodness of Fit Test χ^2_A	Serial Correlation Test χ^2_B	Run Test χ^2_C
	Multiplicative Congruential	a = 13	8.60	1000.00	424.40
**	Multiplicative Congruential	a = 21	8.20	287.60	111.60
**	Multiplicative Congruential	a = 100077	3.60	2.60	9.20
	Mixed Congruential	a = 2501 c = 1	87.40	47.80	7.20
	Mixed Congruential	a = 4001 c = 1	69.40	34.00	23.40
**	Mixed Congruential	a = 5001 c = 1	123.60	145.20	15.10
**	Mixed Congruential	a = 101 c = 7777	15.40	8.40	11.80
**	Mid-square	X_o = 9653042877	17.20	6.40	23.60
	Mid-square	X_o = 2075421801	The sequence degenerated to a sequence of zeros.		
	Mid-square	X_o = 2468013579	The sequence degenerated to a sequence of zeros.		
**	A Modification of Lehmer's Method	a = 23	The sequence was found to cycle at n = 9000.		

X_o = 2075421801 (for each test)

** Denotes starting values and methods chosen for the simulation runs.

$\chi^2_{0.05,9}$ = 16.92 (critical value)

TABLE 1. Results of Statistical Tests on the Methods of Random Number Generation

Replication	Average Waiting Time	Average Length Of Waiting Line	Percentage Of Arrivals Finding The System Busy	Maximum Waiting Time	Maximum Length Of Waiting Line
1	2.868	1.381	0.4495	42	7
2	2.575	1.342	0.4280	35	6
3	2.704	1.347	0.4405	33	5
4	2.725	1.353	0.4420	41	7
5	2.819	1.382	0.4464	38	7
$\bar{X}$	2.738	1.361	0.4412	37.80	6.20
S^2	0.0129	0.0004	0.00007	14.70	0.88

TABLE 2. Simulation Results - Multiplicative Congruential Method (a = 100077)

Replication	Average Waiting Time	Average Length Of Waiting Line	Percentage Of Arrivals Finding The System Busy	Maximum Waiting Time	Maximum Length Of Waiting Line
1	2.796	1.379	0.4403	36	6
2	2.541	1.312	0.4331	32	6
3	2.642	1.338	0.4347	31	5
4	2.637	1.351	0.4370	38	7
5	2.687	1.342	0.4448	40	6
$\bar{X}$	2.661	1.344	0.4380	35.40	6.00
S^2	0.0086	0.0006	0.00002	14.80	0.50

TABLE 3. Simulation Results - Multiplicative Congruential Method (a = 21)

Replication	Average Waiting Time	Average Length Of Waiting Line	Percentage Of Arrivals Finding The System Busy	Maximum Waiting Time	Maximum Length Of Waiting Line
1	2.712	1.355	0.4376	32	6
2	2.920	1.400	0.4500	46	8
3	2.760	1.361	0.4430	43	7
4	2.584	1.330	0.4371	33	5
5	2.748	1.365	0.4420	33	6
$\bar{X}$	2.745	1.362	0.4419	37.40	6.40
S^2	0.0143	0.0010	0.00003	43.30	1.30

TABLE 4. Simulation Results - Mixed Congruential Method (a = 101)

Replication	Average Waiting Time	Average Length Of Waiting Line	Percentage Of Arrivals Finding The System Busy	Maximum Waiting Time	Maximum Length Of Waiting Line
1	2.564	1.308	0.4345	30	5
2	2.655	1.331	0.4391	32	6
3	2.977	1.422	0.4519	57	8
4	2.946	1.449	0.4423	52	7
5	2.542	1.278	0.4442	34	5
$\bar{X}$	2.737	1.358	0.4424	41.00	6.20
S^2	0.0390	0.0044	0.00004	157.00	1.79

TABLE 5. Simulation Results - Mixed Congruential Method (a = 5001)

Replication	Average Waiting Time	Average Length Of Waiting Line	Percentage Of Arrivals Finding The System Busy	Maximum Waiting Time	Maximum Length Of Waiting Line
1	2.711	1.358	0.4464	44	7
2	2.809	1.381	0.4476	32	6
3	2.720	1.360	0.4391	47	6
4	2.698	1.353	0.4395	35	6
5	2.654	1.326	0.4463	47	7
$\bar{X}$	2.718	1.356	0.4438	41.00	6.40
S^2	0.0031	0.0004	0.00002	49.50	0.30

TABLE 6. Simulation Results - Mid-Square Method

Replication	Average Waiting Time	Average Length Of Waiting Line	Percentage Of Arrivals Finding The System Busy	Maximum Waiting Time	Maximum Length Of Waiting Line
1	2.634	1.329	0.4499	29	6
2	2.379	1.294	0.4063	40	6
3	2.445	1.252	0.4328	36	5
4	2.831	1.380	0.4410	31	5
5	2.469	1.295	0.4296	25	5
$\bar{X}$	2.552	1.310	0.4319	32.20	5.40
S^2	0.0331	0.0023	0.0003	34.70	0.30

TABLE 7. Simulation Results - A Modification of Lehmer's Method

Generation Method	Mult. Congruential	Mult. Congruential	Mixed Congruential	Mixed Congruential	Mid-Square	A Mod. Of Lehmer's Meth.
Starting Values	a = 100077	a = 21	a = 101 c = 7777	a = 5001 c = 1	X_o=9653042877	a = 23
χ^2_A	3.60	8.20	15.40	123.60	17.20	CYCLES
χ^2_B	2.60	287.60	8.40	145.20	6.40	
χ^2_C	9.20	111.60	11.80	15.10	23.60	
Average Waiting Time For Five Independent Simulations	2.868	2.796	2.712	2.564	2.711	2.634
	2.575	2.541	2.920	2.655	2.809	2.379
	2.704	2.642	2.760	2.977	2.720	2.445
	2.725	2.637	2.584	2.946	2.698	2.831
	2.819	2.687	2.748	2.542	2.654	2.469
$\bar{X}_j$	2.738	2.661	2.745	2.737	2.718	2.552
S^2_j	0.0129	0.0086	0.0143	0.0390	0.0031	0.0331

TABLE 8. Simulation Results-Average Waiting Time

Generation Method	Mult. Congruential	Mult. Congruential	Mixed Congruential	Mixed Congruential	Mid-Square	A Mod. Of Lehmer's Meth.
Starting Values	a = 100077	a = 21	a = 101 c = 7777	a = 5001 c = 1	X_o=9653042877	a = 23
χ^2_A	3.60	8.20	15.40	123.60	17.20	CYCLES
χ^2_B	2.60	287.60	8.40	145.20	6.40	
χ^2_C	9.20	111.60	11.80	15.10	23.60	
Average Length of Waiting Line for Five Independent Simulations	1.381	1.379	1.355	1.308	1.358	1.329
	1.342	1.312	1.400	1.331	1.381	1.294
	1.347	1.338	1.361	1.422	1.360	1.252
	1.353	1.351	1.330	1.449	1.353	1.380
	1.382	1.342	1.365	1.278	1.326	1.295
$\bar{X}_j$	1.361	1.344	1.362	1.358	1.356	1.310
S^2_j	0.0004	0.0006	0.0010	0.0055	0.0004	0.0023

TABLE 9. Simulation Results - Average Length Of Waiting Line

Generation Method	Mult. Congruential	Mult. Congruential	Mixed Congruential	Mixed Congruential	Mid-Square	A Mod. Of Lehmer's Meth.
Starting Values	a = 100077	a = 21	a = 101 c = 7777	a = 5001 c = 1	X_o=9653042877	a = 23
χ^2_A	3.60	8.20	15.40	123.60	17.20	CYCLES
χ^2_B	2.60	287.60	8.40	145.20	6.40	
χ^2_C	9.20	111.60	11.80	15.10	23.60	
Percentage of Arrivals Finding The System Busy For Five Independent Simulations	0.4495	0.4403	0.4376	0.4345	0.4464	0.4499
	0.4280	0.4331	0.4500	0.4391	0.4476	0.4063
	0.4405	0.4347	0.4430	0.4519	0.4391	0.4328
	0.4420	0.4370	0.4371	0.4423	0.4395	0.4410
	0.4464	0.4448	0.4420	0.4442	0.4463	0.4296

TABLE 10. Simulation Results-Percentage Of Arrivals Finding The System Busy

Generation Method	Mult. Congruential	Mult. Congruential	Mixed Congruential	Mixed Congruential	Mid-Square	A Mod. Of Lehmer's Meth.
Starting Values	a = 100077	a = 21	a = 101 c = 7777	a = 5001 c = 1	X_o=9653042877	a = 23
χ^2_A	3.60	8.20	15.40	123.60	17.20	CYCLES
χ^2_B	2.60	287.60	8.40	145.20	6.40	
χ^2_C	9.20	111.60	11.80	15.10	23.60	
$2 \arcsin\sqrt{P}$	1.469	1.451	1.446	1.439	1.463	1.470
	1.426	1.437	1.471	1.449	1.466	1.382
	1.451	1.439	1.457	1.474	1.449	1.436
	1.455	1.445	1.445	1.455	1.450	1.453
	1.463	1.460	1.455	1.459	1.463	1.430
$\bar{X}_j$	1.453	1.446	1.455	1.455	1.458	1.434
S^2_j	0.0003	0.0001	0.0001	0.0002	0.0001	0.0011

TABLE 11. 2 Arcsin $\sqrt{P}$ Transformation Of The Percentage Of Arrivals Finding The System Busy

Generation Method	Mult. Congruential	Mult. Congruential	Mixed Congruential	Mixed Congruential	Mid-Square	A Mod. Of Lehmer's Meth.
Starting Values	a = 100077	a = 21	a = 101 c = 7777	a = 5001 c = 1	X_o=9653042877	a = 23
x_A^2	3.60	8.20	15.40	123.60	17.20	CYCLES
x_B^2	2.60	287.60	8.40	145.20	6.40	
x_C^2	9.20	111.60	11.80	15.10	23.60	
Maximum Waiting Time For Five Independent Simulations	42	36	32	30	44	29
	35	32	46	32	32	40
	33	31	43	57	47	36
	41	38	33	52	35	31
	38	40	33	34	47	25
$\bar{X}_j$	37.80	35.40	37.40	41.00	41.00	32.20
S_j^2	14.70	14.80	43.30	157.00	49.50	34.70

TABLE 12. Simulation Results-Maximum Waiting Time

Generation Method	Mult. Congruential	Mult. Congruential	Mixed Congruential	Mixed Congruential	Mid-Square	A Mod. Of Lehmer's Meth.
Starting Values	a = 100077	a = 21	a = 101 c = 7777	a = 5001 c = 1	X_o=9653042877	a = 23
χ^2_A	3.60	8.20	15.40	123.60	17.20	CYCLES
χ^2_B	2.60	287.60	8.40	145.20	6.40	
χ^2_C	9.20	111.60	11.80	15.10	23.60	
Maximum Length of Waiting Line For Five Independent Simulations	7	6	6	5	7	6
	6	6	8	6	6	6
	5	5	7	8	6	5
	7	7	5	7	6	5
	6	6	6	5	7	5
$\bar{X}_j$	6.20	6.00	6.40	6.20	6.40	5.40
S^2_j	0.88	0.50	1.30	1.79	0.30	0.30

TABLE 13. Simulation Results-Maximum Length Of Waiting Line

Method of Generation		Mult. Congruent.	Mult. Congruent.	Mixed Congruent.	Mixed Congruent.	Mid-Square	A Mod. of Lehmer's Meth.
Average Waiting Time	$\bar{X}_j$	2.738	2.661	2.745	2.737	2.718	2.552
	S_j^2	0.0129	0.0086	0.0143	0.0390	0.0031	0.0331
Average Length of Waiting Line	$\bar{X}_j$	1.361	1.344	1.362	1.358	1.356	1.310
	S_j^2	0.0004	0.0006	0.0010	0.0055	0.0004	0.0023
Percentage of Arrivals Finding The System Busy (P)	$\bar{X}_j$	0.4412	0.4380	0.4419	0.4424	0.4438	0.4319
	S_j^2	0.00007	0.00002	0.00003	0.00004	0.00002	0.0003
2 Arcsine $\sqrt{P}$ Transformation	$\bar{X}_j$	1.453	1.446	1.455	1.455	1.458	1.434
	S_j^2	0.0003	0.0001	0.0001	0.0002	0.0001	0.0011
Maximum Waiting Time	$\bar{X}_j$	37.80	35.40	37.40	41.00	41.00	32.20
	S_j^2	14.70	14.80	43.30	157.00	49.50	34.70
Maximum Length of Waiting Line	$\bar{X}_j$	6.20	6.00	6.40	6.20	6.40	5.40
	S_j^2	0.88	0.50	1.30	1.79	0.30	0.30

Table 14. Summary of Simulation Results

APPENDIX D

Analysis of Results

Sample Size Calculations

The calculations are based on individual comparisons using Scheffe's method.

Given: Estimate of the maximum $S^2 = 0.01$.

$\alpha = 0.05$; $\beta = 0.10$.

n = number of degrees of freedom used in calculating S^2.

r = number of replications in each column.

$K = 6$ (number of columns).

Let: $\theta = \mu_1 - \mu_2$ (actual difference between means)

$H = X_1 - X_2$

$$V(H) = \frac{\sigma^2}{r} + \frac{\sigma^2}{r} = \frac{2\sigma^2}{r}$$

$$\hat{V}(H) = \frac{2S^2}{r}$$

$$G = \frac{H - \theta}{\sqrt{\hat{V}(H)}} = \frac{X_1 - X_2 - \theta}{\sqrt{\frac{2S^2}{r}}}$$

$G \sim S$ where $S^2 = (K-1) F_{K-1, K(r-1)}$

From the definition of α:

$$S_{0.95} = \frac{H - 0}{\sqrt{\frac{2S^2}{r}}} \qquad (1)$$

From the definition of β:

$$-S_{0.10} = \frac{H - \theta}{\sqrt{\frac{2S^2}{r}}} \qquad (2)$$

Solving (1) and (2) simultaneously:

$$-S_{0.10}\sqrt{\frac{2S^2}{r}} = S_{0.95}\sqrt{\frac{2S^2}{r}} - \theta$$

$$\theta = (S_{0.95} + S_{0.10})\sqrt{\frac{2S^2}{r}} = (S_{0.95} + S_{0.10})\sqrt{\frac{2(.01)}{r}}$$

$$\theta = \frac{0.1414\ (S_{0.95} + S_{0.10})}{\sqrt{r}} \qquad (3)$$

The solution of (3) for several values of r is given below:

r	K(r-1)	$F_{0.05,5,K(r-1)}$	$F_{0.90,5,K(r-1)}$	$S_{0.95}$	$S_{0.10}$	θ
3	12	3.11	0.306	3.94	-1.24	0.220
4	18	2.77	0.310	3.72	-1.25	0.175
5	24	2.62	0.314	3.62	-1.26	0.149

Example of Calculations for Test of Homogeneity of Variance

The example shows the application of Bartlett's test to the average waiting time results.

Ho: $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \sigma_4^2 = \sigma_5^2 = \sigma_6^2$

Let: K = number of samples or columns.

n_j = degrees of freedom used in calculating S_j^2.

$$N = \sum_{j=1}^{6} n_j = 24$$

Sample (j)	S_j^2	Log S_j^2
1	0.0129	-1.8894
2	0.0086	-2.0655
3	0.0143	-1.8447
4	0.0390	-1.4089
5	0.0031	-2.5086
6	0.0331	-1.4802
	$\sum_{j=1}^{6} S_j^2 = 0.1110$	$\sum_{j=1}^{6}$ Log $S_j^2 = -11.1973$

$$M = 2.3026 \left[N \text{ Log } \left(\sum_{j=1}^{6} n_j S_j^2/N\right) - \sum_{j=1}^{6} n_j \text{ Log } S_j^2\right]$$

$$= 2.3026 \left[24 \text{ Log } (4 \text{ X } 0.1110/24) - 4 \text{ X } (-11.1973)\right]$$

$$= 7.37$$

$$C_1 = \sum_{j=1}^{6} \frac{1}{n_j} - \frac{1}{N} = \sum_{j=1}^{6} \frac{1}{4} - \frac{1}{24} = 1.46$$

$$M_{\alpha,K,C_1} = M_{0.05,6,1.46} = 12.10 \text{ (critical value)}$$

Therefore, because $7.37 < 12.10$, the null hypothesis is accepted.

Example of Analysis of Variance Calculations

The example shows the application of one way analysis of variance to the average waiting time results.

Let: X_{ij} = observed simulation result.

θ_j = effect due to the random number generator.

e_{ij} = experimental error.

Model I: $X_{ij} = \mu_j + \theta_j + e_{ij}$

Ho: $\theta_j = 0$

Given: r = 5 (replications in each column)

K = 6 (six means being compared)

Source of Estimate	Sum of Squares	Degrees of Freedom	Mean Square	Estimates
Between Columns	$5 \sum_{j=1}^{6} (\bar{X}_j - \bar{\bar{X}})^2 = SS_1$	5	$\frac{SS_1}{5} = S_1^2$	$\sigma^2 + n\sigma_\theta^2$
Within Columns	$\sum_{i=1}^{5} \sum_{j=1}^{6} (X_{ij} - \bar{X}_j)^2 = SS_2$	24	$\frac{SS_2}{24} = S_2^2$	σ^2
Total	$\sum_{i=1}^{5} \sum_{j=1}^{6} (X_{ij} - \bar{\bar{X}})^2$	29		

$$F = \frac{S_1^2}{S_2^2}$$

$$F_{0.05,K-1,K(r-1)} = F_{0.05,5,24} = 2.62 \quad \text{(Critical value)}$$

Using the value $\bar{\bar{X}} = 2.692$ and the values from Table 8;

$$S_1^2 = \frac{5\sum_{j=1}^{6} (\bar{X}_j - \bar{\bar{X}})^2}{5} = \frac{5(0.0020+0.0010+0.0028+0.0020+0.0007+0.0196)}{5}$$

$$= 0.0281$$

$$S_2^2 = \frac{\sum_{i=1}^{5} \cdot \sum_{j=1}^{6} (X_{ij} - \bar{X}_j)^2}{24} = \frac{(0.0515+0.0343+0.0571+0.1758+.0128+0.1325)}{24}$$

$$= 0.0193$$

$$F = \frac{0.0281}{0.0193} = 1.46$$

Therefore, because $1.46 < 2.62$, the null hypothesis is accepted.

Example of Calculations for Nonparametric One-Criteria Variance Analysis

The example shows the application of the Kruskal-Wallis one-criteria variance analysis test to the maximum length of waiting line results.

Ho: The six independent samples are from the same population.

The following table lists the ranks of the values in Table 13;

Mult. Method	Mult. Method	Mixed Method	Mixed Method	Mid-Square Method	Modified Lehmer's Method
25	15	15	4.5	25	15
15	15	29.5	15	15	15
4.5	4.5	25	29.5	15	4.5
25	25	4.5	25	15	4.5
15	15	15	4.5	25	4.5
84.5	74.5	89.0	78.5	95.0	43.5

Let: K = 6 (number of columns)

R_j = Summation of the ranks of column j.

n_j = number of observations in column j.

$$N = \sum_{j=1}^{6} n_j = 30$$

t = the number of tied observations in a tied group of scores.

$T = t^3 - t$

$$H = \frac{\frac{12}{N(N+1)} \sum_{j=1}^{6} \frac{R_j}{n_j} - 3(N+1)}{1 - \frac{\sum T}{N^3 - N}}$$

$$H \sim \chi^2_{K-1} \sim \chi^2_5$$

$$\chi^2_{0.05,5} = 11.07 \text{ (critical value)}$$

Using the values in the table of ranks:

t: 2 7 8 13

T: 6 336 504 2184

$$1 - \frac{\sum T}{N^3 - N} = 1 - \frac{3030}{26970} = 0.888$$

$$H = \frac{12(37691.00) - 3(30+1)}{5X30\ (30+1)\ (0.888)}$$

$$= 4.80$$

Therefore, because $4.80 < 11.07$, the null hypothesis is accepted.

Summary of Statistical Tests on the Simulation Results

Critical values at 5% level of significance:

Test for Homogeneity of Variance	M = 12.10
Analysis of Variance	F = 2.62
Kruskal-Wallis Nonparametric Test	H = 11.07

Average Waiting Time:

M = 7.36

F = 1.46

Average Length of Waiting Line:

M = 5.19

F = 1.22

Percentage of Arrivals Finding the System Busy:

M = 10.86

F = 1.29

Maximum Waiting Time:

H = 5.05

Maximum Length of Waiting Line:

H = 4.80

APPENDIX E

Simulation Data

Calculation of Distribution Means

Mean of Exponential Arrival Distribution for Operators:

Given: Operators average 11 units wired per shift.

Operators average 4 special entries per shift.

Eighty operators in the shop.

A shift is 460 minutes long.

(11 entries/shift + 4 entries/shift)/oper. x 80 oper. =

1200 entries/shift.

$$\frac{460 \text{ min./shift} \times 60 \text{ sec./min.}}{1200 \text{ entries/shift}} = 23.0 \text{ sec./entry}$$

Mean of Exponential Arrival Distribution for Inspectors:

(11 entries/shift)/oper. x 80 oper. = 880 entries/shift

$$\frac{460 \text{ min./shift} \times 60 \text{ sec./min.}}{880 \text{ entries/shift}} = 31.36 \text{ sec./entry}$$

Length of Entry	Seconds to Enter	Frequency	Freq./n	Cumulative Probability
0[1]	4	797	.022	.022
1	5	0		
2[2]	6	24,362	.664	.686
3	7	0		
4	8	6,507	.177	.863
5	9	0		
6	10	2,846	.078	.941
7	11	0		
8	12	1,557	.042	.983
9	13	0		
10	14	468	.013	.996
11	15	0		
12	16	93	.003	.999
13	17	0		
14	18	26	.001	1.000
15	19	0		
16	20	6	.000	
17	21	0		
18	22	0	.000	
		36,662	1.000	

Notes: 1 - Sign off only (error entries)

2 - Good units

TABLE 15. Inspector Service Time Distribution

BIBLIOGRAPHY

1. Allard, J. L., Dobell, A. R., and Hull, T. E., "Mixed Congruential Random Number Generators for Decimal Machines," Journal of the Association for Computing Machinery, Vol. 10, No. 2 (1963), pp. 131-141.

2. Bennet, C. A, and Franklin, N. L., Statistical Analysis in Chemistry and the Chemical Industry, John Wiley and Sons, Inc., New York, 1954, pp. 196-198.

3. Conway, R. W., Johnson, B. M., and Maxwell, W. L., "Some Problems of Digital Systems Simulation," Journal of the Institute of Management Science, Vol. 6, No. 1 (1960), pp. 92-110.

4. Coveyou, R. R., "Serial Correlation in the Generation of Pseudo-Random Numbers," Journal of the Association for Computing Machinery, Vol. 7, No. 1 (1960), pp. 72-74.

5. Edmonds, A. R., "The Generation of Pseudo-Random Numbers on Electronic Digital Computers," The Computer Journal, Vol. 2, No. 4 (1960), pp. 181-185.

6. Ehrenfeld, S. and Ben-Tuvia, S., "The Efficiency of Statistical Simulation Procedures," Technometrics, Vol. 4, No. 2 (1962), pp. 257-275.

7. Forsythe, G. E., "Generation and Testing of Random Digits at the National Bureau of Standards, Los Angeles, National Bureau of Standards Applied Math Series No. 12, 1951, pp. 34-35.

8. Good, I. J., "The Serial Test for Sampling Numbers and Other Tests for Randomness," Proc. Camb. Phil. Soc., Vol. 49, Part 2 (1953), pp. 276-284.

9. Harling, J., "Simulation Techniques in Operations Research - A Review," Operations Research, Vol. 6, No. 3 (1958), pp. 307-319.

10. Hull, T. E. and Dobell, A. R., "Random Number Generators," SIAM Review, Vol. 4, No. 3 (1962), pp. 230-251.

11. Hull, T. E. and Dobell, A. R., "Mixed Congruential Random Number Generators for Binary Machines," Journal of the Association for Computing Machinery, Vol. 11, No. 1 (1964), pp. 31-40.

12. International Business Machine Corporation, "Random Number Generation and Testing," Reference Manual C20-8011, New York, 1959.

13. Kahn, H. and Marshall, A. W., "Methods of Reducing Sample Size in Monte Carlo Computations," Operations Research, Vol. 1, No. 5 (1953), pp. 263-278.

14. Kruskal, W. H. and Wallis, W. A., "Use of Ranks in One-Criteria Variance Analysis," Journal of the American Statistical Association, Vol. 47, No. 260 (1952), pp. 584-621.

15. Lehmer, D. H., "Mathematical Methods in Large Scale Computing Units," Proceeding of a Second Symposium (1949) on Large-Scale Digital Calculating Machinery," The Annals of the Computation Laboratory of Harvard University, Vol. 26 (1951), pp. 141-146.

16. Malcolm, D. G., "Systems Simulation - A Fundamental Tool for Industrial Engineering," Journal of Industrial Engineering, Vol. 9, No. 3 (1958), pp. 177-187.

17. Mann, H. B. and Wald, A., "On the Choice of the Number of Class Intervals in the Application of the Chi-Square Test," The Annals of Mathematical Statistics," Vol. 13 (1942), pp. 306-317.

18. Moss, J. H., "Commentary on Harling's Simulation Techniques in Operations Research," Operations Research, Vol. 6, No. 4 (1958), pp. 591-593.

19. Peach, P., "Bias in Pseudo-Random Numbers," Journal of the American Statistical Association, Vol. 56, No. 295 (1961), pp. 610-618.

20. Progress in Operations Research, Edited by R. L. Ackoff, John Wiley and Sons, Inc., New York, 1961, pp. 363-419.

21. Rotenburg, A., "A New Pseudo-Random Number Generator," Journal of the Association for Computing Machinery, Vol. 7, No. 1 (1960), pp. 75-77.

22. Scheffe, H., The Analysis of Variance, John Wiley and Sons, Inc., New York, 1959, pp. 66-72.

23. Snedecor, G. W., Statistical Methods, Iowa State University Press, Ames, Iowa, 1956, pp. 285-289.

24. Taussky, O. and Todd, J., "The Generation and Testing of Pseudo-Random Numbers," Symposium on Monte Carlo Methods, John Wiley and Sons, Inc., New York, 1954, pp. 15-27.

25. Van Slyke, R. M., "Monte Carlo Methods and The Pert Problem," Operations Research, Vol. 11, No. 5 (1963), pp. 839-860.

VITA

PERSONAL HISTORY

Name:	Richard James White
Date of Birth:	September 17, 1938
Place of Birth:	Arcade, New York
Parents:	George C. and Mary E. White
Wife:	Carol Ann White
Children:	Elizabeth Ann White

EDUCATIONAL BACKGROUND

Arcade Central School 1952-56

Clarkson College of Technology
Bachelor of Mechanical Engineering 1956-60

New York University
Industrial Engineering Department Attended 1961-62

Lehigh University
Candidate for Master of Science Degree
in Industrial Engineering 1962-64

HONORS

Clarkson College Trustee Scholarship and
Bell Aircraft Scholarship

PROFESSIONAL EXPERIENCE

Western Electric Co., Inc.
New York, New York
Planning Engineer - Installation Methods
June 1960 to July 1962

Western Electric Co., Inc.
Princeton, New Jersey
Candidate in Lehigh Master's Degree Program
August 1962 to June 1964

PROFESSIONAL ORGANIZATIONS

Associate Member, American Society of Mechanical Engineers