5101-275
Flat-Plate Solar
Array Project

DOE/JPL-1012-116
Distribution Category UC-63b

The RANDOM Computer Program

A Linear Congruential Random Number Generator

R.F. Miles, Jr.

(NASA-CR-176664) THE RANDOM COMPUTER PROGRAM: A LINEAR CONGRUENTIAL RANDOM NUMBER GENERATOR (Jet Propulsion Lab.) 56 p HC A04/MF A01 CSCL 09B
N86-22152
Unclas
G3/61 05866

February 15, 1986

Prepared for
U.S. Department of Energy
Through an Agreement with
National Aeronautics and Space Administration
by
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 85-97

5101-275
Flat-Plate Solar
Array Project

DOE/JPL-1012-116
Distribution Category UC-63b

The RANDOM Computer Program

A Linear Congruential Random Number Generator

R.F. Miles, Jr.

February 15, 1986

Prepared for
U.S. Department of Energy
Through an Agreement with
National Aeronautics and Space Administration
by
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 85-97

Prepared by the Jet Propulsion Laboratory, California Institute of Technology, for the U.S. Department of Energy through an agreement with the National Aeronautics and Space Administration.

The JPL Flat-Plate Solar Array Project is sponsored by the U.S. Department of Energy and is part of the National Photovoltaics Program to initiate a major effort toward the development of cost-competitive solar arrays.

This document reports on work done under NASA Task RE-152, Amendment 419, DOE/NASA IAA No. DE-AI01-85CE89008.

ABSTRACT

The RANDOM Computer Program is a FORTRAN program for generating random number sequences and testing linear congruential random number generators (LCGs). This document discusses the linear congruential form of a random number generator, and describes how to select the parameters of an LCG for a microcomputer. This document describes the following:

(1) The RANDOM Computer Program.

(2) RANDOM.MOD, the computer code needed to implement an LCG in a FORTRAN program.

(3) The RANCYCLE and the ARITH Computer Programs that provide computational assistance in the selection of parameters for an LCG.

The RANDOM, RANCYCLE, and ARITH Computer Programs are written in Microsoft FORTRAN for the IBM PC microcomputer and its compatibles. With only minor modifications, the RANDOM Computer Program and its LCG can be run on most microcomputers or mainframe computers.

CONTENTS

SECTION 1

INTRODUCTION

The RANDOM Computer Program is a FORTRAN program for generating random number sequences and testing linear congruential random number generators (LCGs). This document discusses the linear congruential form of a random number generator, and describes how to select the parameters of an LCG for a microcomputer. This document describes the following:

(1) The RANDOM Computer Program.

(2) RANDOM.MOD, the computer code needed to implement an LCG in a FORTRAN program.

(3) The RANCYCLE and the ARITH Computer Programs that provide computational assistance in the selection of parameters for an LCG.

The RANDOM, RANCYCLE, and ARITH Computer Programs are written in Microsoft FORTRAN for the IBM PC microcomputer and its compatibles. With only minor modifications, the RANDOM Computer Program and its LCG can be run on most microcomputers or mainframe computers.

The following topics are discussed in the various sections:

(1) Section 2: The mathematical form of the LCG and the selection of its parameters.

(2) Section 3: The implementation of an LCG using an IBM PC microcomputer, or a compatible microcomputer, equipped with the Intel 8088 microprocessor and the Intel 8087 numeric data processor. This discussion is relevant to the entire family of 8086/8088/80186/80286 microprocessors.

(3) Section 4: The statistical tests of the LCG that are part of the RANDOM Computer Program.

(4) Section 5: The RANDOM Computer Program.

(5) Section 6: Instructions for the user of the RANDOM Computer Program.

(6) Section 7: The RANCYCLE Computer Program for testing an LCG for a full-cycle sequence.

(7) Section 8: The ARITH Computer Program for testing a FORTRAN compiler and the associated microcomputer configuration for the correct arithmetic needed for the LCG.

(8) Section 9: The microcomputer diskette that contains the RANDOM Computer Program.

(9) Section 10: The References.

The following topics are discussed in the various appendixes:

(1) Appendix A: The Microsoft FORTRAN code for the RANDOM Computer Program.

(2) Appendix B: RANDOM.MOD includes the Microsoft FORTRAN code module for the LCG, the required initialization code, and the metacommands required for any program using the LCG.

(3) Appendix C: An example of a RANDOM Computer Program run.

(4) Appendix D: The Microsoft FORTRAN code for the RANCYCLE Computer Program.

(5) Appendix E: An example of a RANCYCLE Computer Program run.

(6) Appendix F: The Microsoft FORTRAN code for the ARITH Computer Program.

(7) Appendix G: An example of an ARITH Computer Program run.

SECTION 2

DISCUSSION OF THE LINEAR CONGRUENTIAL RANDOM NUMBER GENERATOR (LCG)

This discussion of the mathematical form and the selection of the parameters of the Linear Congruential Random Number Generator (LCG) is derived primarily from Knuth (Reference 1), to which the reader is referred for more detail. The LCG is one of the most discussed random number generators in the literature, not only because of its speed and excellent statistical properties, but also because a significant amount of theory has been developed for it. There also is available a microcomputer implementation of a multiplicative random number generator (References 2 and 3).

The purpose of the LCG is to generate a sequence of numbers that have random properties. Define the **n** th number of the sequence as X_n. The property that shall be used to define "random" is that the sequence $\{X_k,\ldots,X_{n-1}\}$ $(k < n)$ shall contain no information for predicting X_n, other than estimates of the range of X_n, which is known a priori. Since the primary motivation for generating the random numbers shall be simulation and the use of cumulative distribution functions for representing random variables (References 4 through 6), it is desirable to generate random number sequences $\{F_1,\ldots,F_n\}$ uniformly distributed over the range $[0,1)$. So while these sequences of fractional numbers could be described more precisely as "random fraction sequences uniformly distributed over the range $[0,1)$," both the X_n and the F_n sequences shall be referred to as "random number sequences."

The LCG is based on a modulo operator. A modulo operator differs from ordinary division in that the result of the modulo operation is only the remainder of the division, while the quotient is discarded. Thus, modulo operations, performed on the sequence of positive integers $\{0,1,2,3,4,5,\ldots\}$ for modulus 3, generates the sequence $\{0,1,2,0,1,2,\ldots\}$. Note that only the numbers $\{0,\ldots,\mathbf{m}-1\}$ can be generated for modulus **m**.

The formula for the LCG is:

$$X_{n+1} = (\mathbf{a}X_n + \mathbf{c}) \bmod \mathbf{m},$$

where **a** is called the multiplier, **c** the increment, and **m** the modulus. The rest of this Section discusses how to select values for these three parameters.

Given X_n, the formula is a function that uniquely determines X_{n+1}. Thus, the LCG clearly does not generate random number sequences, because with only the knowledge of X_n, we can calculate X_{n+1}. For example, $X_{n+1} = 3$ with $X_n = 2$ for the LCG:

$$X_{n+1} = (2X_n + 3) \bmod 4.$$

The requirement for generating truly random number sequences, therefore, must be relaxed somewhat. The requirement can be restated as "sequences of numbers that display the same statistical properties as random number sequences." In the literature, such sequences are called "pseudo-random."

Because we are interested in numbers generated over the interval [0,1), the numbers generated by the LCG are divided by the modulus **m**:

$$F_n = X_n/\mathbf{m}.$$

F_n now is generated over the interval [0,1), with the smallest fraction generated being 0.0 and the largest fraction generated $(\mathbf{m}-1)/\mathbf{m}$.

The use of the LCG, however, guarantees neither desirable statistical properties nor a uniform distribution. For example, with the random number seed, $X_0 = 0$, (the arbitrary number input to start the sequence) the LCG:

$$X_{n+1} = (4X_n + 4) \bmod 8,$$

generates the sequence {4,4,4,4,...}. Thus, if the desirable statistical properties and a uniform distribution are to be obtained, some constraints or functional relationships must be established among the parameters of the LCG. These constraints and functional relationships can be determined either empirically by trial and error, or by the more desirable application of theory. In practice, modulo theory is used to restrict the possible parameter values of the LCG, and empirical results are used to improve on the LCG's statistical properties.

Knuth (Reference 1) has proven the theorem that an LCG will be a full-cycle generator (every value between 0 and **m**-1 will be generated before the sequence repeats) if and only if:

(1) The increment **c** is relatively prime to **m** (**c** and **m** have no prime factors in common).

(2) **a**-1 is a multiple of **p** , for every prime **p** dividing **m** .

(3) **a**-1 is a multiple of 4, if **m** is a multiple of 4.

Full-cycle LCGs are desirable because they not only generate random number sequences of maximum length for a given modulus, but they also permit any number to be used as the random number seed without concern for being trapped in a short cycle.

Only LCGs will be considered for which $\mathbf{m} = 2^{\mathbf{q}}$, where **q** is an integer greater than 2. Thus Conditions (1) and (2) of the above Knuth Theorem can be satisfied by making **a** and **c** odd. Knuth recommends satisfying Condition (3) by requiring that:

$$\mathbf{a} \bmod 8 = 5.$$

In spite of these constraints, a full-cycle LCG does not guarantee desirable statistical properties. Thus, because the LCG:

$$X_{n+1} = (5X_n + 5) \bmod 4,$$

satisfies the three separate criteria of the Knuth Theorem, it is a full-cycle generator. It generates the repeating sequence {0,1,2,3,0,...}, however, which certainly does not look like a random number sequence (even though it is as probable as any other sequence).

Knuth specifies additional criteria, based on both theory and empirical results, that are most likely to result in a LCG with desirable statistical properties:

(4) The modulus $\mathbf{m}$ should be large, preferably as large as is practical.

(5) The multiplier $\mathbf{a}$ should be larger than $\mathbf{m}^{1/2}$, preferably larger than $\mathbf{m}/100$, but smaller than $\mathbf{m} - \mathbf{m}^{1/2}$.

(6) The multiplier $\mathbf{a}$ should not have a regular pattern, e.g., $\mathbf{a} = 121212$ would not be desirable.

(7) Set $\mathbf{c}/\mathbf{m} \approx 0.211,324,865,405,187, \ldots$

(8) Use only the most significant digits of F_n.

These shall be the first eight criteria used to construct the LCG. Based on specific hardware and software implementation, one further criterion, however, must now be developed. The additional criterion is discussed in Section 3.

SECTION 3

IMPLEMENTATION OF THE LCG

The single criterion presented in this section is specifically applicable to the Microsoft implementation of FORTRAN (References 7 and 8). It is run on a microcomputer using the Intel 8088 microprocessor, the Intel 8087 numeric data processor (References 9 through 17), and the IBM Disk Operating System (References 18 and 19). Although the Intel 8087 numeric data processor is not mandatory, its use, for only a small increment in microcomputer cost, allows a significant gain in speed to carry out computationally intensive simulations. The use of other computers and software will have a similar, but not necessarily the same, implementation criterion for construction of a satisfactory LCG.

Calculations must be exact, for the theory of the LCG to be valid. Even an error of 1 in the units digit will render the theory invalid, and will result in sequences that are not full-cycle. The Microsoft FORTRAN Manual states that the double-precision, real-number data type corresponds to the IEEE format, which has a precision of approximately 15.9 digits (Reference 7). This corresponds to the 52 bits of the significand and the sign bit of the 8087 numeric data processor for the long-real data type ($\pm 2^{52} = \pm 4.5\mathrm{x}10^{15}$). This places an absolute upper limit on the product of **a** times X_n for the LCG. One must be alert, nevertheless, to round-off errors as this limit is approached. These considerations lead to the following Criterion 9 for the Microsoft FORTRAN Compiler:

(9) The product of **a** times **m** shall be less than $2^{52} - \mathbf{c}$. This is equal to $4.5\mathrm{x}10^{15} - \mathbf{c}$.

All the criteria now are in hand for a set of rules to construct the LCG for the Microsoft FORTRAN Compiler. Set $\mathbf{m} > 2^{20}, \mathbf{m} \leq 2^{29}$, and $\mathbf{m}/100 < \mathbf{a} < (\mathbf{m} - \mathbf{m}^{1/2})$ such that $\mathbf{a}\,\mathbf{m} < (2^{52} - \mathbf{c})$ to satisfy Criteria 4, 5, and 9. The multiplier **a** should satisfy:

$$\mathbf{a} \bmod 8 = 5,$$

to satisfy Criteria 2 and 3, and should not have a regular pattern to satisfy Criterion 6. The increment **c** should be odd to satisfy Criterion 1 and should satisfy the **c**/**m** ratio of Criterion 7. Only the most significant digits of F_n should be considered to satisfy Criterion 8. These rules cover the nine criteria.

For compilers other than the Microsoft FORTRAN Compiler, use the ARITH Computer Program to establish the precision limits of the arithmetic. This will determine the equivalent of Criterion 9. The ARITH Computer Program is discussed in Section 8 and listed in Appendix D.

The following parameter values satisfy the LCG construction rules for the Microsoft FORTRAN Compiler:

$$\mathbf{a} = 671{,}093$$
$$\mathbf{c} = 7{,}090{,}885$$

$$\mathbf{m} = 33{,}554{,}432 = 2^{25}.$$

These parameters pass the statistical tests described in Section 4 and have been used in the SIMRAND I Computer Program (References 20 through 23).

A 43% increase in speed of the random number generator can be obtained, if the LCG code is written without the use of the FORTRAN internal modulo function. Because of their size, the random numbers and the LCG parameters will have to be of the double-precision floating-point data type. Assuming RANDOM has previously been assigned a value X_n, the Microsoft FORTRAN code for the LCG to generate the next number X_{n+1} in the sequence is :

```
RANX   = RANA*RANDOM + RANC
RANDIV = RANX/RANM
RANT   = DINT(RANDIV)
RANSUB = RANT*RANM
RANDOM = RANX - RANSUB
```

where, by the previous notation,

RANA = **a**
RANC = **c**
RANM = **m**

and

RANDOM = X_n (initial equation)
RANDOM = X_{n+1} (final equation).

SECTION 4

STATISTICAL TESTS FOR THE LCG

This section discusses the following five statistical tests that are performed in the RANDOM Computer Program:

(1) A Cycling Test.

(2) A Chi-Square Test for uniform distribution.

(3) A Kolmogorov-Smirnov Test for uniform distribution.

(4) A Median Runs Test.

(5) A Serial Test for correlation between the most significant digit of F_n and the most significant digit of F_{n+1}.

Although Knuth (Reference 1) considers these tests to be inadequate for rigorous investigations of LCGs, they have value because they are fast, easily understood, and should suffice for the validation of LCGs for most simulation work. LCGs for rigorous investigation should be written in the assembly language of the specific computer, and should use considerably larger modulus numbers than those proposed here.

At a minimum, these statistical tests will provide some assurance of uniform distributions, and also provide two tests for reasonable statistical properties. All LCGs probably will fail some statistical tests. Simulations usually are done with only a subset of the full-cycle random number sequence. Because subsets are less robust when subjected to a statistical test than is the full-cycle sequence, a subset of an excellent LCG may exhibit undesirable statistical properties. Thus, the best approach to obtain confidence in simulation results is to replicate the simulations with different random number seeds and different LCG parameters.

The Cycling Test checks for cycling of the LCG. If LCG-construction rules are followed, the LCG will generate the full-cycle sequence, which will have a non-repeating number of elements equal to the modulus. Given that the theoretical conditions for a full-cycle sequence are met, the reason for this procedure is to test the hardware and software for arithmetical errors. With errors, the LCG most probably will generate a non-repeating sequence less than the full cycle, and may not repeat the random number seed. The Cycling Test compares all numbers generated by the LCG with the random number seed. If the LCG generates a number equal to the random number seed, the program aborts with a count of the number of elements in the sequence that were generated. The full cycle can be tested by requesting the program to generate more numbers than the modulus. If the LCG operates correctly, the program should run through the full cycle, abort, and display a count equal to the modulus. The RANCYCLE Computer Program described in Section 7 specifically was written to carry out this full-cycle test efficiently. The Cycling Test is valid for any LCG for which the multiplier **a** and the modulus **m** are relatively prime.

The Chi-Square Test is the standard one-sample test with 100 categories (Reference 24), in which each category has an interval of 0.01 over the range [0.00,1.00). The null hypothesis is that the distribution is uniform. The uniform distribution hypothesis can be rejected at a significance level of 0.10 for a Chi-Square value = 117, or at a significance level of 0.01 for a Chi-Square value = 135. With the LCG parameters given earlier and a random number seed of 1, a sequence of 10,000 random numbers generated by the LCG yielded a Chi-Square value of 78.7. This is not large enough to reject the uniform distribution hypothesis at the 0.10 level of significance.

The Kolmogorov-Smirnov Test, which uses the same data as the Chi-Square Test, serves as a second test for a uniform distribution (Reference 24). The Kolmogorov-Smirnov Test measures the maximum absolute difference between the observed cumulative distribution function and that for a uniform distribution. The uniform distribution hypothesis can be rejected at a significance level of 0.10 for a Kolmogorov-Smirnov value = $1.22/\mathbf{n}^{1/2}$, or at a significance level of 0.01 for a Kolmogorov-Smirnov value = $1.63/\mathbf{n}^{1/2}$, where $\mathbf{n}$ is the number of random numbers generated. With the LCG parameters given earlier, and a random number seed of 1, a sequence of 10,000 random numbers generated by the LCG yielded a Kolmogorov-Smirnov value of 0.0088. This is not large enough to reject the uniform distribution hypothesis at the 0.10 level of significance (0.0122 for a sample size of 10,000).

The Median Runs Test (Reference 25) is based on a statistic which is a function of the number of runs in a sequence. Elements of a tested sequence either lie below, at, or above the median. A "run" is a maximal subsequence of elements of like kind. Thus, the subsequence which defines a run either contains only elements which lie below the median or contains only elements which lie at or above the median. Because the test is made for the null hypothesis of a uniformly distributed random number sequence, the median is 0.5. This test will identify sequences that have either too many or too few runs. For large samples from a uniformly distributed random number sequence, the number of runs is approximately normally distributed with a mean of half the sample size. The RANDOM Computer Program gives a Median-Runs statistic $\mathbf{z}$ for the sequence, which can be compared against significance levels for the normal distribution. The statistic $\mathbf{z}$ is distributed approximately as the standard normal distribution (with mean = 0.0 and standard deviation = 1.0). The runs-distribution hypothesis can be rejected at a significance level of 0.10 for $|\mathbf{z}| = 1.64$, or at a significance level of 0.01 for $|\mathbf{z}| = 2.58$. With the LCG parameters given earlier and a random number seed of 1, a sequence of 10,000 random numbers generated by the LCG yielded 5,065 runs, and a $\mathbf{z}$ value of +1.30. This is not large enough to reject the runs-distribution hypothesis at the 0.10 level of significance.

The Serial Test measures the correlation between the first significant digit of F_n and the first significant digit of F_{n+1}. The Chi-Square Test is used with a 10x10 array of 100 categories, with the rows corresponding to F_n and the columns F_{n+1}. If there is no correlation between the first significant digits of F_n and F_{n+1} (the null hypothesis), then the (F_n,F_{n+1}) pairs of digits should be uniformly distributed over the array for a uniformly distributed random number sequence. The same significance levels are appropriate as for the Chi-Square Test described above. The sample size for the Serial Test will only be half that for the previous tests. This is because the pairs (F_n, F_{n+1}) and (F_{n+1}, F_{n+2}) are correlated, so it is necessary to use only every other (F_n, F_{n+1}) pair that is generated by the LCG. With the LCG parameters given earlier, and a random number seed of 1, a sequence of 10,000 random numbers generated by the LCG yielded a Serial-Test Chi-Square value of 78.96. This is not large enough to reject the no-correlation hypothesis at the 0.10 level of significance.

SECTION 5

THE RANDOM COMPUTER PROGRAM

As stated above, the RANDOM Computer Program has been written specifically for the Microsoft implementation of FORTRAN. It runs on a microcomputer using the Intel 8088 microprocessor, the Intel 8087 numeric data processor, and the IBM Disk Operating System (Version 2.10). The Program comprises 16 modules. Table 5-1 gives the type, location, and a brief description of each of the Program functions and variables. Module 1 initializes the Program while Module 2 contains all the keyboard input. Module 4, containing Module 5 through Module 9, is a DO loop for generating the LCG random number sequence and the data for the five tests. Module 5 contains the LCG and calculates the F_n sequence from the X_n sequence. Module 9 displays the results for one pass through the DO loop, if that option is exercised. Module 10 through Module 14 calculate and display the statistics resulting from the tests. Module 15 contains the "GENERATOR CYCLED AT COUNT = ..." message that is displayed if Module 5 detects cycling. Module 16 stops the Program. Appendix A lists the FORTRAN code for Version 1.00x2 of the RANDOM Computer Program. The compiled source-code files (object-code files) can be linked into an executable file using either the Microsoft Linker provided with the Microsoft FORTRAN Compiler, or the Phoenix Software PLINK86 Linker (Reference 26), but not the IBM DOS 2.10 Linker. Appendix B gives, in RANDOM.MOD, the Microsoft FORTRAN code module for the LCG, the required initialization code, and the metacommands required for any program using the LCG. Appendix C gives an example of the user input and the RANDOM Computer Program output. The example of Appendix C generates the results used in the description of the statistical tests of Section 4.

~~PAGE 4-4 INTENTIONALLY BLANK~~

Table 5-1. RANDOM Function and Variable Type, Location, and Definition

Name	Type	Module Location	Definition
ABS	FUNCTION	12	Absolute value.
AFAULT	CHAR*1	1,2	Use default LCG parameters if .TRUE.
AINT	FUNCTION	3,6,8	Truncate to REAL.
AREAD	CHAR*1	1,2,9	Include intermediate screen output if .TRUE.
CHISQR	REAL	11,14	Summation for Chi-Square Test.
DELTA	REAL	11,14	Category difference in Chi-Square Test.
DEV	REAL	12	Absolute deviation in Kolmogorov-Smirnov Test.
DINT	FUNCTION	5	Truncate REAL*8 to REAL*8.
FKOL	REAL	12	Maximum absolute deviation in Kolmogorov-Smirnov Test.
FLOAT	FUNCTION	11,12,13,14	Convert INTEGER to REAL*4.
FMRUNS	REAL	13	Expected number of runs in Median Runs Test.
FRUNHI	REAL	13	FRUNHI = FLOAT(IRUNHI)
FRUNLO	REAL	13	FRUNLO = FLOAT(IRUNLO)
FRUNS	REAL	13	FRUNS = FLOAT(IRUNS)
I	INTEGER*4	1,11,12	Indexing variable for the IRNHIS(I) array.
IHIS	INTEGER*4	11,12	IHIS = IRNHIS(I).
IISER	INTEGER*4	14	IISER = ISER(ISER1,ISER2).
IRNBIN	INTEGER*4	6	Category identifier for Chi-Square Test.
IRNHIS	INTEGER*4	1,6,10,11,12	IRNHIS(I) is the histogram array for the Chi-Square and Kolmogorov-Smirnov Tests.
IRUNHI	INTEGER*4	1,7,9,13	Number of elements in the sequence equal to or greater than 0.5.

Table 5-1. (Cont'd)

Name	Type	Module Location	Definition
IRUNLO	INTEGER*4	1,7,9,13	Number of elements in the sequence less than 0.5.
IRUNS	INTEGER*4	1,7,9,13	Number of runs in the sequence for the Median Runs Test.
ISER	INTEGER*4	1,8,9,14	ISER(ISER1,ISER2) is the 10x10 array for the Serial Test.
ISER1	INTEGER*4	1,3,4,8,9,14	First element in the pair-wise correlation for the Serial Test.
ISER2	INTEGER*4	1,4,8,9,14	Second element in the pair-wise correlation for the Serial Test.
ISERW	INTEGER*4	14	Row label in writing ISER(ISER1,ISER2).
KOUNT	INTEGER*4	4,5,7,9	Indexing variable for the DO loop for generating the random number sequence.
KOUNT1	INTEGER*4	5,15	KOUNT1 = KOUNT. Used for transferring KOUNT outside the DO loop if cycling occurs.
LIN	LOGICAL*4	1,8	If LIN = .TRUE., the random number pair is used in the Serial Test. LIN alternates in value for each pass through the DO loop.
LRUNHI	LOGICAL*4	1,7	If LRUNHI = .TRUE., the run is at or above 0.5.
MOD	FUNCTION	9	Modulo Function (remainder arithmetic) used to display every 1,000th pass through the DO loop to the screen.
NKOUNT	INTEGER*4	2,4,11,12	User input for the length of the random number sequence.
RANA	REAL*8	1,2,5	Multiplier **a** for the LCG.
RANC	REAL*8	1,2,5	Additive constant **c** for the LCG.
RANDIV	REAL*8	1,5	Intermediate calculation in the LCG.
RANDOM	REAL*8	1,2,4,5,9,15	Element of the X_n random number sequence.

Table 5-1. (Cont'd)

Name	Type	Module Location	Definition
RANM	REAL*8	1,2,3,5	Modulus **m** for the LCG.
RANSUB	REAL*8	1,5	Intermediate calculation in the LCG.
RANT	REAL*8	1,5	Intermediate calculation in the LCG.
RANX	REAL*8	1,5	Intermediate calculation in the LCG.
RNFRAC	REAL	3,5,6,7,8,9	Element of the F_n random number sequence. $0.0 \leq$ RNFRAC $<$ 1.0.
RNSEED	REAL*8	1,2,3,4,15	User input random number seed for the LCG.
SNGL	FUNCTION	3,5	Convert REAL*8 to REAL*4.
SQRT	FUNCTION	13	Square root.
SRUNS	REAL	13	Standard deviation for the Median Runs Test. SRUNS = SQRT(VRUNS).
VRUNS	REAL	13	Variance calculated for the Median Runs Test.
ZRUNS	REAL	13	z-scale value for the Median Runs Test.

SECTION 6

USER INSTRUCTIONS FOR THE RANDOM COMPUTER PROGRAM

The RANDOM.FOR source code is compiled into the executable file RANDOM.EXE. The user executes the RANDOM Computer Program by typing "RANDOM". The Program displays the default LCG parameters as given in Section 3. The user may select the default LCG parameters, or enter new parameter values. The Program then asks for the number of random numbers (length of the random number sequence) to be generated. This number should be of the order of 1,000 or more for all the statistical tests to be valid. The Program then asks whether an intermediate screen output is to be displayed. Because it slows down the Program significantly, this option should be used only when the user wishes to examine every random number that is generated. If the intermediate screen output is not selected, then only every 10,000th random number will be displayed. The Program then asks for the random number seed X_0. The Program then enters into a DO loop that generates the random number sequence and accumulates the data for the statistical tests. If the random number seed is generated, the Program will abort with the message, "GENERATOR CYCLED AT COUNT = ..." If the Program proceeds normally, after the requested sequence of random numbers has been generated, the Program exits the DO loop and performs and displays the results of the statistical tests. The Program output also displays the last random number generated. It is this random number that should be used as the random number seed for the next run when a simulation is being replicated. Assuming that the total number of random numbers generated does not exceed the modulus of the LCG, this ensures that the same sequence of random numbers will not be used repetitively in the replication. The Program then terminates by returning to the operating system prompt.

SECTION 7

THE RANCYCLE COMPUTER PROGRAM

The RANCYCLE Computer Program is identical to the RANDOM Computer Program, except that the statistical tests have been eliminated to gain speed of operation. This program should be used to test an LCG for a full-cycle sequence. Used with an IBM PC-XT microcomputer, with a clock frequency of 4.7 Megahertz, the RANCYCLE Computer Program will generate four million random numbers per hour. Appendix D gives the FORTRAN code for Version 1.00x1 of the RANCYCLE Computer Program. An example of the user input and the RANCYCLE Computer Program output is presented in Appendix E.

SECTION 8

THE ARITH COMPUTER PROGRAM

The ARITH Computer Program tests the four FORTRAN double-precision arithmetical operations (+,-,*,/) and the FORTRAN double-precision modulo function (DMOD) and truncation function (DINT). The user can run the ARITH Computer Program to estimate the largest parameter values that can be entered into the LCG. The ARITH Program is menu driven. The user may select one of the following seven options:

```
****** THE ARITH MENU ******

+   :   Addition
-   :   Subtraction
*   :   Multiplication
/   :   Division
M   :   Modulo      (DMOD)
T   :   Truncation  (DINT)
Q   :   Quit
```

After an option is selected, the Program asks for one or two numbers. The Program displays the calculation, and the user examines the unit's digit for the correct value. Errors of one or two units in the unit's digit can be expected to occur as the limits of the precision of the hardware and software configuration are reached.

Appendix F gives the FORTRAN code for Version 1.00x1 of the ARITH Computer Program. An example of the user input and the RANCYCLE Computer Program output is presented in Appendix G.

PAGE 7-2 INTENTIONALLY BLANK

SECTION 9

THE COSMIC DISKETTE

The RANDOM Computer Program is available on microcomputer diskette from COSMIC (Computer Software Management & Information Center), NASA's clearinghouse where software is transferred from government agencies to industrial or other users (Reference 27). The RANDOM Computer Program files are contained as auxiliary files on the diskette for the SIMRAND I Computer Program (Reference 23). The microcomputer diskette is an industry-standard 5-1/4 inch, double-sided, double-density, soft-sector diskette, with 40 tracks and 9 sectors per track. It can be read with the Microsoft MS-DOS (Version 2.0 or later) operating system (References 18 and 19).

Along with the files for the SIMRAND I Computer Program, the microcomputer diskette contains all the files associated with the RANDOM Computer Program: RANDOM.FOR, RANDOM.MOD, RANDOM.EXE, RANCYCLE.FOR, RANCYCLE.EXE, ARITH.FOR, and ARITH.EXE. Because of the way the executable files (.EXE) have been compiled, the files require the operation of the Intel 8087 numeric coprocessor. All of these files are read-only protected. The RANDOM Computer Program can be run by typing "RANDOM" at the operating system prompt. The two executable auxiliary files, RANCYCLE.EXE and ARITH.EXE, can be run by typing "RANCYCLE" or "ARITH" at the operating system prompt.

SECTION 10

REFERENCES

1. Knuth, D. E., The Art of Computer Programming, Volume II: Seminumerical Algorithms, Addison-Wesley, Reading, Massachusetts, 1969.

2. Lewis, P. A., Orav, E. J., and Uribe, L., Introductory Simulation and Statistics Package, Wadsworth Advanced Books & Software, Monterey, California, 1984.

3. Lewis, P. A., Orav, E. J., and Uribe, L., Advanced Simulation and Statistics Package, Wadsworth Advanced Books & Software, Monterey, California, 1985.

4. Fishman, G. S., Principles of Discrete Event Simulation, John Wiley, New York, 1978.

5. Rubinstein, R. Y., Simulation and the Monte Carlo Method, John Wiley, New York, 1981.

6. Yakowitz, S. J., Computational Probability and Simulation, Addison-Wesley, Reading, Massachusetts, 1977.

7. Microsoft FORTRAN Compiler for the MS-DOS Operating System: User's Guide, Version 3.30, Microsoft Corporation, Bellevue, Washington, 1985.

8. Microsoft FORTRAN: Reference Manual, Microsoft Corporation, Bellevue, Washington, 1985.

9. The iAPX 86/88, 186/188 User's Manual: Programmer's Reference, Intel Corporation, Santa Clara, California, 1985.

10. Bradley, D. J., Assembly Language Programming for the IBM Personal Computer, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

11. Lafore, R., Assembly Language Primer for the IBM PC, New American Library, New York, 1984.

12. Morgan, C. L., and Waite, M., 8086/8088 16-Bit Microprocessor Primer, BYTE/McGraw-Hill, Peterborough, New Hampshire, 1982.

13. Morse, S. P., The 8086 Primer: An Introduction to Its Architecture, System Design, and Programming, Hayden Book Company, Rochelle Park, New Jersey, 1980.

14. Rector, R., and Alexy, G., The 8086 Book, OSBORNE/McGraw-Hill, Berkeley, California, 1980.

15. Scanlon, L. J., IBM PC Assembly Language: A Guide for Programmers, Robert J. Brady Co., Bowie, Maryland, 1983.

16. Startz, R., 8087 Applications and Programming for the IBM PC and Other PCs, Robert J. Brady Co., Bowie, Maryland, 1983.

17. Willen, D. C., and Krantz, J. I., 8088 Assembler Language Programming: The IBM PC, Howard W. Sams & Co., Indianapolis, Indiana, 1983.

18. Microsoft Corp., Disk Operating System, Version 2.10 (First Edition), IBM No. 1502343, International Business Machines, Boca Raton, Florida, September 1983.

19. Microsoft Corp., Disk Operating System: Technical Reference, Version 2.10 (First Edition), IBM No. 1502346, International Business Machines, Boca Raton, Florida, September 1983.

20. Miles, R. F., Jr., Introduction to SIMRAND: SIMulation of Research ANd Development Projects, JPL Publication 82-20, Jet Propulsion Laboratory, Pasadena, California, March 1, 1982.

21. Miles, R. F., Jr., "The SIMRAND Methodology: Simulation of Research and Development Projects," Large Scale Systems, Vol. 7, pp. 59-67, 1984.

22. Miles, R. F., Jr., The SIMRAND Methodology: Theory and Application for the Simulation of Research and Development Projects, JPL Publication 85-98, Jet Propulsion Laboratory, Pasadena, California, February 15, 1986.

23. Miles, R. F., Jr., The SIMRAND I Computer Program: Simulation of Research and Development Projects, JPL Publication 85-96, Jet Propulsion Laboratory, Pasadena, California, February 15, 1986.

24. Lindgren, B. W., Statistical Theory, MacMillan Publishing Co., New York, 1976.

25. Siegal, S., Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill, New York, 1956.

26. PLINK86: Linkage Editor for Intel 8086/8088, Phoenix Software Associates Ltd., North Easton, Maine, Version 1.47, 1985.

27. COSMIC, Suite 112, Barrow Hall, The University of Georgia, Athens, Georgia 30602, Phone: (404) 542-3265.

APPENDIX A

THE RANDOM COMPUTER PROGRAM

APPENDIX A

The RANDOM Computer Program

```
************************************************************************
*                            RANDOM.FOR                                *
*                                                                      *
*                                                                      *
*  PROGRAMMER:                                                         *
*     RALPH F. MILES, JR.                                              *
*     SYSTEMS DIVISION                                                 *
*     JET PROPULSION LABORATORY                                        *
*     PASADENA, CA 91109                                               *
*                                                                      *
*  VERSION: 1.00X2                                                     *
*  DATE:    04/24/85                                                   *
*                                                                      *
*----------------------------------------------------------------------*
*  THE PROGRAM "RANDOM.FOR" GENERATES, DISPLAYS, AND TESTS RANDOM      *
*  NUMBER SEQUENCES COMPATIBLE WITH MICROSOFT FORTRAN-77 AND THE       *
*  8086/8088 MICROPROCESSOR WITH THE 8087 NUMERIC COPROCESSOR.         *
*                                                                      *
*----------------------------------------------------------------------*
*                         CONFIGURATION CHANGES                        *
*                                                                      *
*  VER.      DATE                       CHANGES                        *
*                                                                      *
*  1.00X1  01/19/85  * ORIGINAL.                                       *
*  1.00X2  04/24/85  * MODULE  1: DELETE KOUNT1. CYCLE CHECK IN        *
*                                 MODULE 5. DELETE LWRITE. ADD LIN.    *
*                    * MODULE  2: WRITE DEFAULT PARAMETERS WITH F11.1. *
*                    * MODULE  4: DELETE KOUNT1.                       *
*                    * MODULE  5: KOUNT1 = KOUNT IF CYCLE.             *
*                    * MODULE  8: IF STATEMENT FOR LIN.                *
*                    * MODULE  9: USE MOD FN TO TEST FOR WRITE.        *
*                                 DELETE IF (LWRITE) ... ENDIF.        *
*                    * MODULE 13: MINOR FORMAT CHANGE.                 *
*                    * MODULE 14: FLOAT(NCOUNT) -> FLOAT(NCOUNT)/2.    *
*                    * PROGRAM  : ICORR? -> ISER                       *
*                                                                      *
************************************************************************


$TITLE:'RANDOM.LST'
$NODEBUG
$NOFLOATCALLS
$STORAGE:4

************************************************************************

      PROGRAM RANDOM

************************************************************************


***** INITIALIZE PROGRAM.  {MODULE 1}

      CHARACTER*1       AFAULT,AREAD
```

PRECEDING PAGE BLANK NOT FILMED

```
      DOUBLE PRECISION RANA,RANC,RANM,RNSEED,RANX,RANDIV,RANT,RANSUB,
     *                 RANDOM
      LOGICAL          LRUNHI,LIN

      DIMENSION IRNHIS(100),ISER(10,10)

*     INITIALIZE HISTOGRAM FOR CHI SQUARE AND K-S TESTS.
      DO 100 I=1,100
         IRNHIS(I) = 0.0
100   CONTINUE

*     PARAMETERS FOR MEDIAN RUNS TEST.
      IRUNS  = 0
      IRUNHI = 0
      IRUNLO = 0

*     INITIALIZE LIN AND HISTOGRAM ARRAY FOR SERIAL TEST.
      LIN = .TRUE.
      DO 110 ISER1 = 1,10
         DO 110 ISER2 = 1,10
            ISER(ISER1,ISER2) = 0
110   CONTINUE

*     DEFAULT LINEAR CONGRUENTIAL GENERATOR PARAMETERS.
      RANA   =   671093.0
      RANC   =  7090885.0
      RANM   = 33554432.0

***  {END MODULE 1}


***** KEYBOARD INPUT.  {MODULE 2}

      WRITE  (*,120)
120   FORMAT (1X,'GENERATOR IS: (RANA*RANDOM + RANC) MOD RANM.')

      WRITE  (*,130) RANA,RANC,RANM
130   FORMAT (1X,'DEFAULT PARAMETERS FOR GENERATOR ARE:'/
     *        5X,     'RANA = ',F11.1/
     *        5X,     'RANC = ',F11.1/
     *        5X,     'RANM = ',F11.1//
     *        1X,'USE DEFAULT PARAMETERS FOR GENERATOR (Y/N): '\)
      READ   (*,140) AFAULT
140   FORMAT (A1)

      IF (AFAULT .NE. 'Y') THEN

         WRITE  (*,150)
150      FORMAT (1X,5X,'ENTER RANA: '\)
         READ   (*,160) RANA
160      FORMAT (BN,F10.0)

         WRITE  (*,170)
170      FORMAT (1X,5X,'ENTER RANC: '\)
```

```
        READ   (*,180) RANC
180     FORMAT (BN,F10.0)

        WRITE  (*,190)
190     FORMAT (1X,5X,'ENTER RANM: '\)
        READ   (*,200) RANM
200     FORMAT (BN,F10.0)

      ENDIF

      WRITE  (*,210)
210   FORMAT (/1X,'TOTAL NUMBER OF RANDOM NUMBERS TO GENERATE: '\)
      READ   (*,220) NKOUNT
220   FORMAT (BN,I10)

      WRITE  (*,230)
230   FORMAT (/1X,'INCLUDE INTERMEDIATE SCREEN OUTPUT (Y/N): '\)
      READ   (*,240) AREAD
240   FORMAT (A1)

      WRITE (*,250)
250   FORMAT (/1X,'ENTER RANDOM NUMBER SEED: '\)
      READ (*,260) RNSEED
260   FORMAT (BN,F10.0)

***  {END MODULE 2}


***** INITIAL DATUM FOR SERIAL TEST ARRAY.  {MODULE 3}

      RNFRAC = SNGL(RNSEED/RANM)

      ISER1  = AINT(RNFRAC*10) + 1

***  {END MODULE 3}


***** GENERATE RANDOM NUMBERS AND TEST DATA.  {MODULE 4}

      RANDOM = RNSEED

      DO 280 KOUNT = 1,NKOUNT


*****   GENERATE ONE RANDOM NUMBER.  {MODULE 5}

*       FOR ACCURACY, DO MODULO ARITHMETIC W/O MODULO FUNCTION.
        RANX   = RANA*RANDOM + RANC
        RANDIV = RANX/RANM
        RANT   = DINT(RANDIV)
        RANSUB = RANT*RANM
        RANDOM = RANX - RANSUB

*       TEST FOR CYCLING OF THE RANDOM NUMBER GENERATOR.
```

```
        IF (RANDOM .EQ. RNSEED) THEN
           KOUNT1 = KOUNT
           GO TO 410
        ENDIF

        RNFRAC = SNGL(RANDOM/RANM)

        IF (RNFRAC .GE. 1.0) RNFRAC = 0.9999

***     {END MODULE 5}


*****   DATA FOR CHI SQUARE TEST.  {MODULE 6}

        IRNBIN = AINT(100*RNFRAC) + 1

        IRNHIS(IRNBIN) = IRNHIS(IRNBIN) + 1

***     {END MODULE 6}


*****   DATA FOR MEDIAN RUNS TEST.  {MODULE 7}

        IF (RNFRAC .GE. 0.5) THEN
           IRUNHI = IRUNHI + 1
        ELSE
           IRUNLO = IRUNLO + 1
        ENDIF

        IF (KOUNT .EQ. 1) THEN

           IF (RNFRAC .GE. 0.5) THEN
              LRUNHI = .TRUE.
              IRUNS  = 1
           ELSE
              LRUNHI = .FALSE.
              IRUNS  = 1
           ENDIF

        ELSE

           IF (LRUNHI) THEN

              IF (RNFRAC .LT. 0.5) THEN
                 IRUNS  = IRUNS + 1
                 LRUNHI = .FALSE.
              ENDIF

           ELSE

              IF (RNFRAC .GE. 0.5) THEN
                 IRUNS  = IRUNS + 1
                 LRUNHI = .TRUE.
              ENDIF
```

```
          ENDIF

        ENDIF

***     {END MODULE 7}


*****   DATA FOR SERIAL TEST.  {MODULE 8}

*       INCLUDE ONLY EVERY OTHER PAIR FOR RANDOMNESS.

        ISER2 = AINT(RNFRAC*10) + 1

        IF (LIN) THEN

          ISER(ISER1,ISER2) = ISER(ISER1,ISER2) + 1
          LIN = .FALSE.

        ELSE

          LIN = .TRUE.

        ENDIF

***     {END MODULE 8}


*****   WRITE VARIABLES FOR ONE LOOP.  {MODULE 9}

        IF ((AREAD .EQ. 'Y') .OR. (MOD(KOUNT,10000) .EQ. 0)) THEN

          WRITE  (*,270) KOUNT,RANDOM,RNFRAC,IRUNS,IRUNHI,IRUNLO,
     *                   ISER1,ISER2,ISER(ISER1,ISER2)
270       FORMAT (1X,'COUNT:',I8,6X,'RANDOM NUMBER: ',F10.0,6X,
     *              'RANDOM FRACTION: ',F6.4/
     *            1X,'MEDIAN RUNS TEST:    ',
     *              'IRUNS: ',I8,6X,'IRUNHI:',I8,6X,'IRUNLO:',I8/
     *            1X,'SERIAL TEST:         ',
     *              'ISER1: ',I8,6X,'ISER2: ',I8,6X,'ISER:  ',I8/)

        ENDIF

***     {END MODULE 9}


*       PREPARE FOR NEXT SERIAL DATA.
        ISER1 = ISER2

280   CONTINUE

***   END (KOUNT) DO LOOP.  {END MODULE 4}
```

```
***** WRITE RUN HISTOGRAM.  {MODULE 10}

      WRITE  (*,290)
290   FORMAT (/1X,'RUN HISTOGRAM FOR FRACTIONAL (0.0 - 1.0) ',
     *        'RANDOM NUMBERS')

      WRITE  (*,300) (IRNHIS(I),I=1,100)
300   FORMAT (1X,10I7)

***   {END MODULE 10}


***** CHI SQUARE TEST.  {MODULE 11}

      CHISQR = 0.0

      DO 310 I=1,100

         IHIS   = IRNHIS(I)
         DELTA  = FLOAT(IHIS) - FLOAT(NKOUNT)/100

         CHISQR = DELTA*DELTA/(FLOAT(NKOUNT)/100) + CHISQR

310   CONTINUE

      WRITE  (*,320) CHISQR
320   FORMAT (/1X,'CHI SQUARE:',F8.4)

***   {END MODULE 11}


***** KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST.  {MODULE 12}

      IHIS = 0
      FKOL = 0.0

      DO 330 I=1,100

         IHIS = IHIS + IRNHIS(I)
         DEV  = ABS(FLOAT(IHIS)/NKOUNT - FLOAT(I)/100)

         IF (FKOL .LT. DEV) FKOL = DEV

330   CONTINUE

      WRITE  (*,340) FKOL
340   FORMAT (/1X,'KOLMOGOROV-SMIRNOV MAXIMUM DEVIATION:',F8.4)

***   {END MODULE 12}


***** MEDIAN RUNS TEST.  {MODULE 13}

      FRUNS  = FLOAT(IRUNS)
```

```
      FRUNHI = FLOAT(IRUNHI)
      FRUNLO = FLOAT(IRUNLO)

      FMRUNS = (2*FRUNHI*FRUNLO)/(FRUNHI + FRUNLO) + 1

      VRUNS  = ((2*FRUNHI*FRUNLO)*(2*FRUNHI*FRUNLO - FRUNHI - FRUNLO))/
     *         (((FRUNHI + FRUNLO)**2)*(FRUNHI + FRUNLO -1))

      SRUNS = SQRT(VRUNS)

      ZRUNS = (FRUNS - FMRUNS)/SRUNS

      WRITE  (*,350) FRUNS,FMRUNS,SRUNS,ZRUNS
350   FORMAT (/1X,'MEDIAN RUNS TEST:'/
     *         1X,'FRUNS:',F12.2,8X,'FMRUNS:',F12.2,8X,'SRUNS:',F12.2/
     *         1X,'Z-SCORE FOR THE RUNS:',F14.4)

***   {END MODULE 13}


***** SERIAL (CHI SQUARE) TEST.  {MODULE 14}

*     USE FLOAT(NCOUNT)/2 AS THE NUMBER OF PAIR OBSERVATIONS.

      CHISQR = 0.0

      DO 360 ISER1 = 1,10
         DO 360 ISER2 = 1,10

            IISER = ISER(ISER1,ISER2)
            DELTA  = FLOAT(IISER) - (FLOAT(NKOUNT)/2)/100

            CHISQR = DELTA*DELTA/((FLOAT(NKOUNT)/2)/100) + CHISQR

360   CONTINUE

      WRITE  (*,370) CHISQR
370   FORMAT (/1X,'SERIAL TEST VALUE (CHI SQUARE):',F10.4)

      WRITE  (*,380)
380   FORMAT (1X,'SERIAL TEST HISTOGRAM:')

      DO 400 ISER1 = 1,10
         ISERW = ISER1 - 1
         WRITE  (*,390) ISERW,(ISER(ISER1,ISER2),ISER2=1,10)
390      FORMAT (1X,I1,':',10I7)
400   CONTINUE

***   {END MODULE 14}


***** GENERATOR CYCLE MESSAGE.  {MODULE 15}

410   CONTINUE
```

```
      IF (RANDOM .EQ. RNSEED) THEN
         WRITE  (*,420) KOUNT1
420      FORMAT (//1X,'GENERATOR CYCLED AT COUNT = ',I10//)
      ENDIF

***   {END MODULE 15}


***** STOP PROGRAM.  {MODULE 16}

      STOP

      END

***   {END MODULE 16}

****************************** RANDOM.FOR ******************************
```

APPENDIX B

THE MICROSOFT FORTRAN CODE FOR THE LCG

APPENDIX B

The Microsoft FORTRAN Code for the LCG

```
************************************************************************
*                          RANDOM.MOD                                  *
*                                                                      *
*                                                                      *
*  PROGRAMMER:                                                         *
*     RALPH F. MILES, JR.                                              *
*     SYSTEMS DIVISION                                                 *
*     JET PROPULSION LABORATORY                                        *
*     PASADENA, CA 91109                                               *
*                                                                      *
*  VERSION: 1.0X2                                                      *
*  DATE:    04/24/85                                                   *
*                                                                      *
*----------------------------------------------------------------------*
*  THIS CODE IS EXTRACTED FROM RANDOM.FOR, AND CONTAINS ONLY THOSE     *
*  LINES OF CODE FROM DESIGNATED MODULES THAT MUST BE INCORPORATED     *
*  (OR SOME EQUIVALENT LINES OF CODE MUST BE INCORPORATED) INTO A      *
*  MICROSOFT FORTRAN-77 PROGRAM TO USE THE RANDOM NUMBER GENERATOR OF  *
*  RANDOM.FOR.  THE VERSION NUMBER AND DATE ARE KEPT CONSISTENT WITH   *
*  CHANGES TO RANDOM.FOR EVEN IF NO CHANGES ARE MADE TO THE EXTRACTED  *
*  CODE LINES.                                                         *
*                                                                      *
************************************************************************


$TITLE:'RANDOM.LST'
$NODEBUG
$NOFLOATCALLS
$STORAGE:4


***** INITIALIZE PROGRAM.  {MODULE 1}

      DOUBLE PRECISION RANA,RANC,RANM,RNSEED,RANX,RANDIV,RANT,RANSUB,
     *                 RANDOM

*     DEFAULT LINEAR CONGRUENTIAL GENERATOR PARAMETERS.
      RANA   =   671093.0
      RANC   =  7090885.0
      RANM   = 33554432.0

***  {END MODULE 1}


***** KEYBOARD INPUT.  {MODULE 2}

      WRITE  (*,210)
210   FORMAT (/1X,'TOTAL NUMBER OF RANDOM NUMBERS TO GENERATE: '\)
      READ   (*,220) NKOUNT
220   FORMAT (BN,I10)

      WRITE  (*,250)
250   FORMAT (/1X,'ENTER RANDOM NUMBER SEED: '\)
      READ   (*,260) RNSEED
```


PRECEDING PAGE BLANK NOT FILMED

```
260    FORMAT (BN,F10.0)

***   {END MODULE 2}


***** GENERATE RANDOM NUMBERS.  {MODULE 4}

      RANDOM = RNSEED

      DO 280 KOUNT = 1,NKOUNT


*****    GENERATE ONE RANDOM NUMBER.  {MODULE 5}

*        FOR ACCURACY, DO MODULO ARITHMETIC W/O MODULO FUNCTION.
         RANX   = RANA*RANDOM + RANC
         RANDIV = RANX/RANM
         RANT   = DINT(RANDIV)
         RANSUB = RANT*RANM
         RANDOM = RANX - RANSUB

         RNFRAC = SNGL(RANDOM/RANM)

         IF (RNFRAC .GE. 1.0) RNFRAC = 0.9999

***      {END MODULE 5}

280   CONTINUE

***   END (KOUNT) DO LOOP.  {END MODULE 4}


***** STOP PROGRAM.  {MODULE 16}

      STOP

      END

***   {END MODULE 16}

****************************** RANDOM.MOD ********************************
```

APPENDIX C

EXAMPLE OF A RANDOM COMPUTER PROGRAM RUN

APPENDIX C
Example of a RANDOM Computer Program Run

```
A:> RANDOM

GENERATOR IS: (RANA*RANDOM + RANC) MOD RANM.
DEFAULT PARAMETERS FOR GENERATOR ARE:
    RANA =     671093.0
    RANC =    7090885.0
    RANM =   33554432.0

USE DEFAULT PARAMETERS FOR GENERATOR (Y/N): Y
TOTAL NUMBER OF RANDOM NUMBERS TO GENERATE: 10000
INCLUDE INTERMEDIATE SCREEN OUTPUT (Y/N): N
ENTER RANDOM NUMBER SEED: 1

COUNT:     10000      RANDOM NUMBER:   14745073.      RANDOM FRACTION:   .4394
MEDIAN RUNS TEST:     IRUNS:        5065     IRUNHI:     4929      IRUNLO:     5071
SERIAL TEST:          ISER1:           4     ISER2:         5      ISER:         61


RUN HISTOGRAM FOR FRACTIONAL (0.0 - 1.0) RANDOM NUMBERS
      88     100     105     112     108      84      89     101     103      97
     109      88     103     107     108      95     103     104      87     102
     102     105     106      93     102     113      97     112      79     122
     113      96     110     107      94     116     100      98     109      96
      96     102     103     104     103     103     103     101     110      83
     101      98      93      95     112     107      93      93      92     105
     103      94      71      93     106      93     100     103     101     104
     108      92      97      97      98      94     119     102      95     111
      84     108     104      90      95      89      94     106      80     104
     100      93     100      86     108     102     107     104      90     115

CHI SQUARE: 78.7200

KOLMOGOROV-SMIRNOV MAXIMUM DEVIATION:    .0088

MEDIAN RUNS TEST:
FRUNS:      5065.00        FMRUNS:      4999.99         SRUNS:         49.99
Z-SCORE FOR THE RUNS:            1.3005

SERIAL TEST VALUE (CHI SQUARE):     78.9600
SERIAL TEST HISTOGRAM:
0:      45      58      60      57      39      49      47      51      54      47
1:      46      53      42      56      60      49      38      52      48      46
2:      48      54      45      44      51      58      56      56      47      47
3:      47      52      60      50      61      44      41      55      46      59
4:      55      54      62      41      46      50      38      44      51      54
5:      45      57      53      55      58      50      48      57      59      50
6:      47      60      57      55      51      38      44      56      45      49
7:      43      52      46      49      52      37      57      48      45      55
8:      54      39      45      55      46      43      45      53      44      44
9:      51      37      55      62      48      39      52      57      47      53

Stop - Program terminated.
```

APPENDIX D

THE RANCYCLE COMPUTER PROGRAM

APPENDIX D

The RANCYCLE Computer Program

```
************************************************************************
*                           RANCYCLE.FOR                               *
*                                                                      *
*                                                                      *
*  PROGRAMMER:                                                         *
*     RALPH F. MILES, JR.                                              *
*     SYSTEMS DIVISION                                                 *
*     JET PROPULSION LABORATORY                                        *
*     PASADENA, CA 91109                                               *
*                                                                      *
*  VERSION: 1.00X1                                                     *
*  DATE:    02/22/85                                                   *
*                                                                      *
*----------------------------------------------------------------------*
*  THE PROGRAM "RANCYCLE.FOR" TESTS THE PARAMETERS OF THE LINEAR       *
*  CONGRUENTIAL GENERATOR OF RANDOM NUMBERS, THE FORTRAN IMPLEMENTA-   *
*  TION, AND THE MICROCOMPUTER HARDWARE FOR THE CYCLE LENGTH.  A       *
*  GOOD GENERATOR SHOULD GENERATE A MAXIMUM CYCLE LENGTH.              *
*                                                                      *
*----------------------------------------------------------------------*
*                       CONFIGURATION CHANGES                          *
*                                                                      *
*  VER.      DATE                   CHANGES                            *
*                                                                      *
*  1.00X1  02/22/85  * ORIGINAL.                                       *
*                                                                      *
************************************************************************


$TITLE:'RANCYCLE.LST'
$NODEBUG
$NOFLOATCALLS
$STORAGE:4

************************************************************************

      PROGRAM RANCYC

************************************************************************


***** INITIALIZE PROGRAM.  {MODULE 1}

      CHARACTER*1       AFAULT,AREAD
      DOUBLE PRECISION RANA,RANC,RANM,RNSEED,RANX,RANDIV,RANT,RANSUB,
     *                 RANDOM

*     DEFAULT LINEAR CONGRUENTIAL GENERATOR PARAMETERS.
      RANA    =    671093.0
      RANC    =   7090885.0
      RANM    = 33554432.0

***   {END MODULE 1}
```

PRECEDING PAGE BLANK NOT FILMED

```
***** KEYBOARD INPUT.  {MODULE 2}

      WRITE  (*,120)
120   FORMAT (1X,'GENERATOR IS: (RANA*RANDOM + RANC) MOD RANM.')

      WRITE  (*,130) RANA,RANC,RANM
130   FORMAT (1X,'DEFAULT PARAMETERS FOR GENERATOR ARE:'/
     *        5X,     'RANA = ',F11.1/
     *        5X,     'RANC = ',F11.1/
     *        5X,     'RANM = ',F11.1//
     *        1X,'USE DEFAULT PARAMETERS FOR GENERATOR (Y/N): '\)
      READ   (*,140) AFAULT
140   FORMAT (A1)

      IF (AFAULT .NE. 'Y') THEN

         WRITE  (*,150)
150      FORMAT (1X,5X,'ENTER RANA: '\)
         READ   (*,160) RANA
160      FORMAT (BN,F10.0)

         WRITE  (*,170)
170      FORMAT (1X,5X,'ENTER RANC: '\)
         READ   (*,180) RANC
180      FORMAT (BN,F10.0)

         WRITE  (*,190)
190      FORMAT (1X,5X,'ENTER RANM: '\)
         READ   (*,200) RANM
200      FORMAT (BN,F10.0)

      ENDIF

      WRITE  (*,210)
210   FORMAT (/1X,'TOTAL NUMBER OF RANDOM NUMBERS TO GENERATE: '\)
      READ   (*,220) NKOUNT
220   FORMAT (BN,I10)

      WRITE  (*,230)
230   FORMAT (/1X,'INCLUDE INTERMEDIATE SCREEN OUTPUT (Y/N): '\)
      READ   (*,240) AREAD
240   FORMAT (A1)

      WRITE (*,250)
250   FORMAT (/1X,'ENTER RANDOM NUMBER SEED: '\)
      READ (*,260) RNSEED
260   FORMAT (BN,F10.0)

***  {END MODULE 2}


***** INITIAL DATUM.  {MODULE 3}
```

```
      RNFRAC = SNGL(RNSEED/RANM)

***  {END MODULE 3}


***** GENERATE RANDOM NUMBERS AND TEST DATA.  {MODULE 4}

      RANDOM = RNSEED

      DO 280 KOUNT = 1,NKOUNT


*****    GENERATE ONE RANDOM NUMBER.  {END MODULE 5}

*        FOR ACCURACY, DO MODULO ARITHMETIC W/O MODULO FUNCTION.

         RANX   = RANA*RANDOM + RANC
         RANDIV = RANX/RANM
         RANT   = DINT(RANDIV)
         RANSUB = RANT*RANM
         RANDOM = RANX - RANSUB

*        TEST FOR CYCLING OF THE RANDOM NUMBER GENERATOR.
         IF (RANDOM .EQ. RNSEED) THEN
            KOUNT1 = KOUNT
            GO TO 410
         ENDIF

         RNFRAC = SNGL(RANDOM/RANM)

***      {END MODULE 5}


*****    WRITE VARIABLES FOR ONE LOOP.  {MODULE 9}

         IF ((AREAD .EQ. 'Y') .OR. (MOD(KOUNT,10000) .EQ. 0)) THEN

            WRITE  (*,270) KOUNT,RANDOM,RNFRAC
270         FORMAT (1X,'COUNT:',I8,6X,'RANDOM NUMBER: ',F10.0,6X,
     *                  'RANDOM FRACTION: ',F6.4/)

         ENDIF

***      {END MODULE 9}

280   CONTINUE

***   END (KOUNT) DO LOOP.  {END MODULE 4}


***** GENERATOR CYCLE MESSAGE.  {MODULE 15}

410   CONTINUE
```

```
      IF (RANDOM .EQ. RNSEED) THEN
         WRITE  (*,420) KOUNT1
420      FORMAT (//1X,'GENERATOR CYCLED AT COUNT = ',I10//)
      ENDIF

***   {MODULE 15}


***** STOP PROGRAM.  {MODULE 16}

      STOP

      END

***   {END MODULE 16}

****************************** RANCYCLE.FOR ******************************
```

APPENDIX E

EXAMPLE OF A RANCYCLE COMPUTER PROGRAM RUN

APPENDIX E
Example of a RANCYCLE Computer Program Run

```
A:> RANCYCLE

GENERATOR IS: (RANA*RANDOM + RANC) MOD RANM.
DEFAULT PARAMETERS FOR GENERATOR ARE:
    RANA =    671093.0
    RANC =   7090885.0
    RANM =  33554432.0

USE DEFAULT PARAMETERS FOR GENERATOR (Y/N): Y
TOTAL NUMBER OF RANDOM NUMBERS TO GENERATE: 100000
INCLUDE INTERMEDIATE SCREEN OUTPUT (Y/N): N
ENTER RANDOM NUMBER SEED: 1

COUNT:    10000     RANDOM NUMBER:  14745073.     RANDOM FRACTION:  .4394

COUNT:    20000     RANDOM NUMBER:  18354145.     RANDOM FRACTION:  .5470

COUNT:    30000     RANDOM NUMBER:  11285969.     RANDOM FRACTION:  .3363

COUNT:    40000     RANDOM NUMBER:  14970817.     RANDOM FRACTION:  .4462

COUNT:    50000     RANDOM NUMBER:   4701617.     RANDOM FRACTION:  .1401

COUNT:    60000     RANDOM NUMBER:  10297249.     RANDOM FRACTION:  .3069

COUNT:    70000     RANDOM NUMBER:  15439249.     RANDOM FRACTION:  .4601

COUNT:    80000     RANDOM NUMBER:  24780673.     RANDOM FRACTION:  .7385

COUNT:    90000     RANDOM NUMBER:  30391665.     RANDOM FRACTION:  .9057

COUNT:   100000     RANDOM NUMBER:  11759457.     RANDOM FRACTION:  .3505

Stop - Program terminated.
```

APPENDIX F

THE ARITH COMPUTER PROGRAM

APPENDIX F

The ARITH Computer Program

```
************************************************************************
*                              ARITH.FOR                               *
*                                                                      *
*                                                                      *
*  PROGRAMMER:                                                         *
*     RALPH F. MILES, JR.                                              *
*     SYSTEMS DIVISION                                                 *
*     JET PROPULSION LABORATORY                                        *
*     PASADENA, CA 91109                                               *
*                                                                      *
*  VERSION: 1.00X1                                                     *
*  DATE:    06/13/85                                                   *
*                                                                      *
*----------------------------------------------------------------------*
*  THE PROGRAM "ARITH.FOR" TESTS THE FOUR FORTRAN DOUBLE-PRECISION     *
*  ARITHMETICAL OPERATIONS (+,-,*,/) AND THE FORTRAN DOUBLE-PRECISION  *
*  FUNCTIONS MODULO (DMOD) AND TRUNCATION (DINT).                      *
*                                                                      *
*----------------------------------------------------------------------*
*                         CONFIGURATION CHANGES                        *
*                                                                      *
*   VER.    DATE                    CHANGES                            *
*                                                                      *
*  1.0X01  06/13/85  * ORIGINAL.                                       *
*                                                                      *
************************************************************************


$TITLE:'RANDOM.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:4

************************************************************************

      PROGRAM ARITH

************************************************************************


***** INITIALIZE PROGRAM.  {MODULE 1}

      CHARACTER*1      MENU
      DOUBLE PRECISION ARITH1,ARITH2,ARITH3

***   {END MODULE 1}


***** MENU DISPLAY.  {MODULE 2}

100   CONTINUE

      WRITE  (*,110)
110   FORMAT (1X,/////////////////////////
```

```
     *        24X,'****** THE ARITH MENU ****** '/
     *        /
     *        24X,'+     :    Addition               '/
     *        24X,'-     :    Subtraction            '/
     *        24X,'*     :    Multiplication         '/
     *        24X,'/     :    Division               '/
     *        24X,'M     :    Modulo      (DMOD)     '/
     *        24X,'T     :    Truncation (DINT)      '/
     *        24X,'Q     :    Quit                   '/
     *        /
     *        24X,'--------------------------- '/
     *        //)

      WRITE (*,'(20X,A\)') 'Enter a Menu Character & <RETURN>:  '
      READ  (*,'(BN,A1)') MENU

***   {END MODULE 2}


***** QUIT THE PROGRAM.  {MODULE 3}

      IF (MENU .EQ. 'Q') THEN

         WRITE  (*,120)
120      FORMAT (1X,/////////////////////////)

         GOTO 999

      ENDIF

***   {END MODULE 3}


***** PERFORM ADDITION.  {MODULE 4}

      IF (MENU .EQ. '+') THEN

         WRITE  (*,130)
130      FORMAT (1X,/////////////////////////
     *              30X,'****** ADDITION ******  ')

         WRITE  (*,140)
140      FORMAT (/////1X,5X,'Enter the first number for addition:  '\)
         READ   (*,150) ARITH1
150      FORMAT (BN,F20.0)

         WRITE  (*,160)
160      FORMAT (   //1X,5X,'Enter the second number for addition: '\)
         READ   (*,170) ARITH2
170      FORMAT (BN,F20.0)

         ARITH3 = ARITH1 + ARITH2

         WRITE  (*,180) ARITH3
```

```
180       FORMAT (   //1X,5X,'The addition of the two numbers is:   ',
     *            F20.0)

          WRITE (*,'(//1X,A\)') 'Enter <RETURN> to continue: '
          READ  (*,'(BN,A1)') MENU

      ENDIF

***   {END MODULE 4}


***** PERFORM SUBTRACTION.  {MODULE 5}

      IF (MENU .EQ. '-') THEN

          WRITE  (*,190)
190       FORMAT (1X,////////////////////////
     *            30X,'****** SUBTRACTION ****** ')

          WRITE  (*,200)
200       FORMAT (/////1X,5X,
     *            'Enter the first number for subtraction:  '\)
          READ   (*,210) ARITH1
210       FORMAT (BN,F20.0)

          WRITE  (*,220)
220       FORMAT (//1X,5X,'Enter the second number for subtraction: '\)
          READ   (*,230) ARITH2
230       FORMAT (BN,F20.0)

          ARITH3 = ARITH1 - ARITH2

          WRITE  (*,240) ARITH3
240       FORMAT (//1X,5X,'The subtraction of the two numbers is:   ',
     *            F20.0)

          WRITE (*,'(//1X,A\)') 'Enter <RETURN> to continue: '
          READ  (*,'(BN,A1)') MENU

      ENDIF

***   {END MODULE 5}


***** PERFORM MULTIPLICATION.  {MODULE 6}

      IF (MENU .EQ. '*') THEN

          WRITE  (*,250)
250       FORMAT (1X,////////////////////////
     *            30X,'****** MULTIPLICATION ****** ')

          WRITE  (*,260)
260       FORMAT (/////1X,5X,
```

```
     *           'Enter the first number for multiplication:  '\)
        READ   (*,270) ARITH1
270     FORMAT (BN,F20.0)

        WRITE  (*,280)
280     FORMAT (   //1X,5X,
     *           'Enter the second number for multiplication: '\)
        READ   (*,290) ARITH2
290     FORMAT (BN,F20.0)

        ARITH3 = ARITH1 * ARITH2

        WRITE  (*,300) ARITH3
300     FORMAT (   //1X,5X,
     *           'The multiplication of the two numbers is: ',F20.0)

        WRITE (*,'(//1X,A\)') 'Enter <RETURN> to continue: '
        READ  (*,'(BN,A1)') MENU

      ENDIF

***   {END MODULE 6}


***** PERFORM DIVISION.  {MODULE 7}

      IF (MENU .EQ. '/') THEN

        WRITE  (*,310)
310     FORMAT (1X,////////////////////////
     *           30X,'****** DIVISION ****** ')

        WRITE  (*,320)
320     FORMAT (/////1X,5X,'Enter the first number for division:  '\)
        READ   (*,330) ARITH1
330     FORMAT (BN,F20.0)

        WRITE  (*,340)
340     FORMAT (   //1X,5X,'Enter the second number for division: '\)
        READ   (*,350) ARITH2
350     FORMAT (BN,F20.0)

        ARITH3 = ARITH1 / ARITH2

        WRITE  (*,360) ARITH3
360     FORMAT (   //1X,5X,'The division of the two numbers is: ',
     *           F30.10)

        WRITE (*,'(//1X,A\)') 'Enter <RETURN> to continue: '
        READ  (*,'(BN,A1)') MENU

      ENDIF

***   {END MODULE 7}
```

```
***** PERFORM MODULO FUNCTION.  {MODULE 8}

      IF (MENU .EQ. 'M') THEN

         WRITE  (*,370)
370      FORMAT (1X,/////////////////////////
     *           30X,'****** MODULO ****** ')

         WRITE  (*,380)
380      FORMAT (/////1X,5X,
     *           'Enter the argument of the modulo function: '\)
         READ   (*,390) ARITH1
390      FORMAT (BN,F20.0)

         WRITE  (*,400)
400      FORMAT (   //1X,5X,
     *           'Enter the modulus of the modulo function:  '\)
         READ   (*,410) ARITH2
410      FORMAT (BN,F20.0)

         ARITH3 = DMOD(ARITH1,ARITH2)

         WRITE  (*,420) ARITH3
420      FORMAT (   //1X,5X,'The remainder is: ',F20.0)

         WRITE (*,'(//1X,A\)') 'Enter <RETURN> to continue: '
         READ  (*,'(BN,A1)') MENU

      ENDIF

***   {END MODULE 8}


***** PERFORM TRUNCATION.  {MODULE 9}

      IF (MENU .EQ. 'T') THEN

         WRITE  (*,430)
430      FORMAT (1X,/////////////////////////
     *           30X,'****** TRUNCATION ****** ')

         WRITE  (*,440)
440      FORMAT (/////1X,5X,
     *           'Enter the decimal number for truncation: '\)
         READ   (*,450) ARITH1
450      FORMAT (BN,F30.10)

         ARITH3 = DINT(ARITH1)

         WRITE  (*,460) ARITH3
460      FORMAT (   //1X,5X,
     *           'The truncation of the decimal number is: ',F30.10)
```

```
         WRITE (*,'(//1X,A\)') 'Enter <RETURN> to continue: '
         READ  (*,'(BN,A1)') MENU

      ENDIF

***   {END MODULE 9}


***** GO TO MENU.  {MODULE 10}

      GOTO 100

***   {END MODULE 10}


***** STOP PROGRAM.  {MODULE 11}

999   CONTINUE

      STOP

      END

***   {END MODULE 11}

****************************** ARITH.FOR ******************************
```

APPENDIX G

EXAMPLE OF AN ARITH COMPUTER PROGRAM RUN

APPENDIX G

Example of an ARITH Computer Program Run

```
A: > ARITH


                    ****** THE ARITH MENU ******

                    +    :    Addition
                    -    :    Subtraction
                    *    :    Multiplication
                    /    :    Division
                    M    :    Modulo     (DMOD)
                    T    :    Truncation (DINT)
                    Q    :    Quit

                    -----------------------------

                Enter a Menu Character & <RETURN>:  +


                         ****** ADDITION ******

     Enter the first number for addition:  1234

     Enter the second number for addition: 4321

     The addition of the two numbers is:              5555.

Enter <RETURN> to continue:


                    ****** THE ARITH MENU ******

                    +    :    Addition
                    -    :    Subtraction
                    *    :    Multiplication
                    /    :    Division
                    M    :    Modulo     (DMOD)
                    T    :    Truncation (DINT)
                    Q    :    Quit

                    -----------------------------

                Enter a Menu Character & <RETURN>:  Q


Stop - Program terminated.
```

PRECEDING PAGE BLANK NOT FILMED

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 85-97	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle The RANDOM Omputer Program	5. Report Date February 15, 1986 6. Performing Organization Code
7. Author(s) R.F. Miles, Jr.	8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109	10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546	JPL Publication 14. Sponsoring Agency Code

15. Supplementary Notes Sponsored by the U.S. Department of Energy through Interagency Agreement DE-AI01-85CE89008 with NASA; also identified as DOE/JPL 1012-116 and as JPL Project No. 5101-275 (RTOP or Customer Code 776-52-61).

16. Abstract

The RANDOM Computer Program is a FORTRAN program for generating random number sequences and testing linear congruential random number generators (LCGs). This document discusses the linear congruential form of a random number generator, and describes how to select the parameters of an LCG for a microcomputer. This document describes the following:

(1) The RANDOM Computer Program.

(2) RANDOM.MOD, the computer code needed to implement an LCG in a FORTRAN program.

(3) The RANCYCLE and the ARITH Computer Programs that provide computational assistance in the selection of parameters for an LCG.

The RANDOM, RANCYCLE, and ARITH Computer Programs are written in Microsoft FORTRAN for the IBM PC microcomputer and its compatibles. With only minor modifications, the RANDOM Computer Program and its LCG can be run on most microcomputers or mainframe computers.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Computer Programming and Software Systems Analysis	Unclassified-unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	72	

JPL 0184 R 9/8