A PORTABLE PACKAGE OF SPECTRAL TEST FOR PSEUDO-RANDOM NUMBER GENERATORS

By

CHUNG-PEI CHU

Bachelor of Education

Taiwan Normal University

Taiwan, R. O. C.

1976

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fullfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1986

A PORTABLE PACKAGE OF SPECTRAL TEST FOR PSEUDO-RANDOM NUMBER GENERATORS

Thesis Approved:

J B Chandler
Thesis Adviser

D A Thoreson

M W Grace

Norman N. Durham
Dean of the Graduate College

PREFACE

This thesis writes and tests a portable software package of spectral test for pseudo-random number generators. Because calculations in spectral test require multiple precision integer operations. This software package include multiple precision integer arithmetic operations. This package has been tested from WATIV and FORTRAN G compilers in IBM 360/370, and from FORTRAN V complier in IBM 3081. This package can be run in any computer with FORTRAN compiler.

I would like to express sincere gratitude to my major adviser, Dr. J. P. Chandler, for his help on this thesis and for his assistance throughout my studies at Oklahoma State University. I am also thankful to my committee members, Dr. S. A. Thoreson and Dr. M. J. Folk, for their contributions and advice, and Dr. D. W. Grace for substituting during my oral examination.

My husband, Shen-Then, my son, Brian, deserve my deepest appreciation for their constant support, encouragement, and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Monte Carlo calculations are used in many areas of both applied mathematics and statistics. Such calculations depend on having available sequences of numbers which appear to be drawn at random from a particular probability distribution. Such numbers which are generated in a deterministic way are referred to as pseudo-random numbers. They only appear to be random.

The important studies of pseudo-random numbers are the following fields: the determination of the periods of the iterative processes used and the selection of those with periods of adequate length; the application of statistical tests to the output sequences of such generators. By far the most popular deterministic way to generate random number sequences is the linear congruential method, because it is possible to prove theoretically that the sequence has certain desirable random properties and that no degeneracy will occur. Two common types of statistical tests for randomness of random number generators are empirical tests and theoretical tests. Empirical tests are done by computer manipulation of groups of numbers from the sequence and evaluation of certain statistics. Theoretical tests are

done by establishing characteristics of the sequence using number theory based on the recurrence rule used to form the sequence.

An important test for the randomness of pseudo-random generators is the spectral test, formulated in 1965 by Coveyou and Macpherson [4]. The spectral test embodies aspects of both the empirical and the theoretical tests. It considers quantities averaged over the full period, and it requires a computer program to determine the results. Knuth [9] formulated this test into an algorithm in 1969. He considered this test especially important because, in addition to being passed by all good generators of linear congruential method, it is actually failed by all such generators known to be bad. But in this thesis we found an exception, one generator known to be bad pass this portable spectral test package.

The major goal of this thesis is to study the spectral test and to implement Knuth's spectral test algorithm for the evaluation of linear congruential pseudo-random number generators.

Chapter II presents a brief description of linear congruential pseudo-random number generators and establishes the rules used to choose the moduli and multipliers so that the sequence of maximum period can be obtained. Chapter III introduces the spectral test for the randomness of computer-generated sequences. The primary concern about the spectral test in this thesis will be placed on discussing

the theory behind the test, showing examples of the test, and reviewing Knuth's spectral test algorithm. Chapter IV describes the computer implementation of a portable package for the spectral test which is based on Knuth's algorithm. The final chapter will present the results from the computer tests and conclusions about the spectral test.

CHAPTER II

RANDOM NUMBER GENERATORS - LINEAR CONGRUENTIAL METHODS

In this chapter we will consider methods for generating sequences of random fractions, i.e., random real numbers U_n, uniformly distributed between zero and one. Actually it is generating integers X_n between zero and some number m; the fraction

$$U_n = X_n/m \tag{2.1}$$

will then lie between zero and one.

Of the countless methods which have been invented to generate pseudo-uniform sequences, almost all are subsumed under the linear congruential method. The main objective of this chapter will be to study the number theoretic properties of linear congruential methods and summarize the results in two theorems.

The Linear Congruential Scheme

The computing scheme which defines the procedures is as follows.

X_0,	the starting value;	non-negative integer.
a,	the multiplier;	Positive integer.
c,	the increment;	non-negative integer.

m, the modulus; $m > x$, $m > a$, $m > c$.

Define a sequence $< X_n >$ of non-negative integers, each less than m, by means of the congruence relation

$$X_{n+1} = (aX_n + c) \pmod{m}, \quad n \text{ a non-negative integer.} \quad (2.2)$$

This is called a linear congruential sequence. Finally, to obtain numbers in the interval [0,1), form the sequence (2.1). For example, the sequence obtained when $X_0 = 3$, A = 2, c = 1, m = 10 is

$$X_1 = (2 * 3 + 1) \pmod{10} = 7$$
$$X_2 = (2 * 7 + 1) \pmod{10} = 5$$
$$X_3 = (2 * 5 + 1) \pmod{10} = 1$$
$$X_4 = (2 * 1 + 1) \pmod{10} = 3$$

.

.

.

3,7,5,1,3,7,5,1,.... and U_n will be 3/10, 7/10, 5/10, 1/10, ...

It is clear that the congruential sequences always "gets into a loop", because it can contain at most m different numbers, each number in a particular sequence being determined solely by its predecessor. The repeating cycle is called the period; the sequence in the above example has a period of length 4. A useful sequence will of course have a relatively long period. The principles of choosing X_0, a, c, and m appropriately will be investigated carefully in following sections.

Best Choice of Modulus

Since the period of a congruential sequence can never exceed m, the value of m should be rather large. Another factor which influences the choice of m is speed of generation; the value of m should make the computation of $(aX_n + c) \pmod m$ quite fast. In practice it is desirable to choose m to be a power of 2 on a binary machine, or a power of 10 on a decimal machine. We are then able to avoid the division which is implicit in the congruence, and also the division to form X_n/m. In this thesis we will follow Knuth's [9] terminology "word size" is 2 for a binary computer which has 32 bits word. For example, let m equal the computer's word size. Compute the result of aX_n in register 2 , then result of $aX_n \pmod m$ is the lower half bits of the product. For example, if the computer contains 4 bit words, then the word size is 2**4. Suppose a=3, X_0=14; then a*X_0=42. The bit representation of 42 is 111010, the result of $aX_0 \pmod m$ is 10. The bit representation of 10 is 1010. It is the lower 4 bits of aX_n.

Knuth [9] suggested considering using m=word size $\pm$ 1, because when m=word size, the right-hand digits of X_n are much less random than the left-hand digits.

If p is a divisor of m, and if

$$Y_n = X_n \pmod p = X_n - q_i p$$

$$X_n = Y_n + q_i p$$

then $$X_{n+1} = (aX_n + c) \pmod m$$

$$= (aY_n + aq_1 p + c) \pmod{q_2 p}$$

$$= aY_n + aq_1 p + c - q_3 p$$

$$= (aY_n + c) - (q_3 - aq_1)p$$

$$= (aY_n + c) - q_4 p$$

$$Y_{n+1} = X_{n+1} \pmod{p}$$

$$= ((aY_n + c) - q_4 p) \pmod{p}$$

$$= aY_n + c - q_4 p - q_5 p$$

$$= aY_n + c - (q_4 + q_5)p$$

$$= (aY_n + c) \pmod{p} \qquad (2.3)$$

q_k are non-negative integer.

The eq(2.3) shows that the low-order bits of X_n form a congruential sequence. For example, X_0=13, a=1, c=3, m=2**4 , the bit representations of X_n are in table I. The low order two bits of X_n are the result of $Y_n = X_n \pmod{2**2}$. The result of eq(2.3) is that the low-order two bits of X_n form a congruential sequence which has a period of length 4 or less. Similarly, the low-order three bits are periodic with a period of length at most 2**3; and the least significant bit of X_n is either constant (see example in Table II) or it strictly alternates between zero and one.

This situation does not occur when m=word size$\pm$1; in this case, the low-order bits of X_n will behave just as randomly as the high-order bits do. For example, suppose X_0 =13, a=3, c=0, m=2**4-1, the bits representation of X_n in Table III.

In most applications, the low-order bits are insignificant, and the choice m=word size is quite

TABLE I

BIT REPRESENTATION OF X_n WHEN a=1, c=3, m=2**4
LEAST SIGNIFICANT BIT OF X_n ALTERNATES
BETWEEN ZERO AND ONE

n	Value of X_n	Bit Representation
0	13	1101
1	0	0000
2	3	0011
3	6	0110
4	9	1001
5	12	1100
6	15	1111
7	2	0010
8	5	0101
9	8	1000
10	11	1011
11	14	1110
12	1	0001
13	4	0100
14	7	0111
15	10	1010
16	13	1101

TABLE II

BIT REPRESENTATION OF X_n WHEN m=2**4, a=3, c=0
THE LEAST SIGNIFICANT BIT IS CONSTANT

n	Value Of X_n	Bit Representation
0	13	1101
1	7	0111
2	5	0101
3	15	1111
4	13	1101

TABLE III

BIT REPRESENTATION OF X_n WHEN m=2**4-1, a=3, c=0
LOW-ORDER BITS WOULD BE SATISFACTORILY RANDOM

n	Value Of X_n	Bit Representation
0	13	1101
1	9	1001
2	12	1100
3	6	0110
4	3	0011
5	9	1001

satisfactory. Other choices are possible and will be found in references [6,9].

Best Choice Of Multiplier

A long period is essential for any sequence which is to be used as a source of random numbers. But a long period is only one desirable criterion for the randomness of the sequence. It is quite possible to have a very long period in a completely nonrandom sequence. For example, when a = c = 1, the sequence is simply $X_{n+1} = (X_n + 1) \pmod m$ and this obviously has a period of length m, but it is not random.

In this section will discuss how to choose the multiplier a so as to give the period of maximum length. The terms multiplicative congruential method and mixed congruential method are used to denote linear congruential methods with c=0 and c≠0 respectively.

Mixed Congruential Methods

Theorem A. The mixed congruential sequence has a period of length m if and only if all following conditions are true.

1. c is relatively prime to m;
2. a = 1 (mod p), if p is a prime factor of m;
3. a = 1 (mod 4), if 4 is a factor of m.

Thus with m a power of 2, as is natural in a binary machine, we need only have c odd and a=1 mod 4. With m a power of 10, we need only have c not divisible by 2 or 5 and

a=1 (mod 20). When $m=p^e$, and p is odd,

$$a = 1 + kp^{\alpha}, \tag{2.4}$$

then the sequence has maximum length m. For example, m=3**3, c=2, p=3, choose a = 1 + 3 * 4 = 13 . The sequence in Table IV has a period of length m. When

$$m = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_n^{e_n},$$

select

$$a = 1 + kp_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \ldots p_n^{\alpha_n}. \tag{2.5}$$

For example, if m=2*3**2=18, c=5, X_0=5, k=5, select $a = 1+5*2^3*3 = 121$. The sequence in Table V has a period of length m.

Multiplicative Congruential Methods

Although multiplicative congruential methods are slightly faster than mixed congruential methods, the maximum period length of m can not be achieved. In fact, this is quite obvious, since the sequence now satisfies the relation

$$X_{n+1} = aX_n \pmod{m}, \tag{2.6}$$

and the value X_n=0 should never appear lest the sequence degenerate to zero. In general, if p is any divisor of m and if X is a multiple of p, we will have X_{n+1}, X_{n+2}, ..., all multiples of p. So when c=0, we will want X_n to be relatively prime to m for all n, and this limits the length of the period. We are no longer able to choose a so that the sequence has full period m. We can, however, choose a so that the period is acceptably large.

TABLE IV

EXAMPLE FOR m=p ,p IS ODD AND a = 1 + kp^{α}
SEQUENCE HAS MAXIMUM LENGTH m

n	Value Of X_n
0	1
1	15
2	8
3	25
4	3
5	14
6	22
7	18
8	20
9	19
10	6
11	26
12	16
13	21
14	5
15	13
16	9
17	11
18	10
19	24
20	17
21	7
22	12
23	23
24	4
25	0
26	2
27	1

TABLE V

EXAMPLE FOR $m = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_n^{e_n}$, SELECT $a = 1 + k p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_n^{\alpha_n}$, SEQUENCE HAS MAXIMUM LENGTH m.

n	Value Of X_n
0	5
1	16
2	15
3	20
4	13
5	12
6	17
7	10
8	9
9	14
10	7
11	6
12	11
13	4
14	3
15	8
16	1
17	0
18	5

The basic theorem for the multiplicative congruential methods is more complicated and more difficult than for the mixed congruential methods. Before we describe theorem B, we need some basic number theory.

Basic Number Theory For Theorem B [8]

1. (Euler) Let a be relatively prime to m and $\phi(m)$ be the number of positive integers less than and relatively prime to m. Then

$$a^{\phi(m)} = 1 \pmod{m}.$$

2. Let m have the factorization into distinct prime powers given by

$$m = 2^{e_1} p_1^{e_2} p_2^{e_3} \ldots .$$

The minimal universal exponent $\lambda(m)$, called the indicator of m, is the least common multiple

$$\lambda(m) = \text{LCM}\,[\lambda(2^{e_1}), \lambda(p_1^{e_2}), \ldots,]$$

of the indicators of the prime power factors of m [2].

3. The indicators of any power of an odd prime are equal to its totient

$\theta(m)$, defined by

$$\lambda(p^e) = \theta(p^e) = (p-1)(p^{e-1}) \quad \text{if } p > 2.$$

Furthermore,

$$\lambda(2) = 1, \quad \lambda(4) = 2, \quad \lambda(2^e) = 2 \quad \text{if } e > 3.$$

In order to give the basic result, it is necessary to define the concept of a primitive root. For a given integer m, a relatively prime integer a is said to be a primitive root of m if it belongs to the exponent $\theta(m)$ [8]. That is

to say, $\phi(m)$ is the smallest power to which a may be raised before unity is obtained. Note that $\phi(15) = 8$ while $\lambda(15) = 4$. Hence, there can exist no primitive root for m = 15.

The main properties of multiplicative congruential methods can be summarized in the following theorem.

Theorem B [2,6]. The sequence defined by taking c = 0 in the congruence relation (2.1) has maximal period, provided that

1. X_0 is relatively prime to m;

and

2. a is a primitive root of p^e, if p^e is a factor of m, with p odd and e as large as possible, or with p = 2 and e = 1 or 2;

and

3. $\phi(2^e) = 2^{e-2}$, if 2^e is a factor of m, with e > 2.

Moreover, for any m, there exist values of a satisfying these conditions, and, finally, the maximal period is the lowest common multiple of the periods, $(p-1)p^{e-1}$ or 2^{e-2}, with respect to the prime power factors.

Note that we can obtain a period of length m-1 if m is prime; this is just one less than the maximum length. Knuth [9] and Hull [6] list some rules to find primitive roots of m.

Rules to Find Primitive Roots

a is a primitive root modulo p if and only if

1. $p^e = 2$, a is odd; or $p^e = 4$, a mod 4 = 3; or $p^e = 8$, a

mod 8 = 3, 5, 7; or p = 2, e > 4, a mod 8 = 3 or 5;

or

2. p is odd, e = 1, a = 0 (mod p), and $a^{(p-1)/q} \not\equiv 1 \pmod{p}$ for any prime divisor q of (p - 1);

or

3. p is odd, e > 1, a satisfies condition 2, and $a^{p-1} \not\equiv 1 \pmod{p^2}$.

In the common case $m = 2^e$, with $e \geq 4$, the conditions above simplify to the single requirement that a = 3 or 5 (modulo 8). The second most common case is when $m = 10^e$. Knuth gave the following rules to find a primitive root of m.

If $m = 10^e$, $e \geq 5$, c = 0, and X_0 is not a multiple of 2 or 5, the period of the linear congruential sequence is $5 * 10^{e-2}$ if and only if a mod 200 equals one of the following 32 values:

3, 11, 13, 19, 21, 27, 29, 37, 53, 59, 61, 67, 69, 77, 83, 91, 109, 117, 123, 131, 133, 139, 141, 147, 163, 171,173, 179, 181, 187, 189, 197.

Potency

High potency is another criterion for a good multiplier. According to Knuth, "The potency of a linear congruential sequence with maximum period is defined to be the least integer s such that $(a-1)^s \equiv 0 \pmod{m}$. For multiplier with potencies less than 5 there is a high degree of dependency between X_n, X_{n+1}, and X_{n+2}. So a good

multiplier has potency higher than 5.

Other Methods

Linear congruential sequences are not the only sources of random numbers that have been proposed for computer use. Many other methods are developed from the linear congruential methods.

1. Quadratic Congruential Method [9].

$$X_{n+1} = (dX_n^2 + aX_n + c) \pmod{m} \qquad (2.7)$$

The sequence defined by (2.6) has a period of the maximum length m; the restrictions are not much more severe than in the linear method.

2. Additive number generator [6,9].

Some investigations have been made of another class of generators, called "additive". An example of such a generator is obtained if one begins with two integers X_0 and X_1 and then defines the rest of the sequence by

$$X_{n+1} = (X_n + X_{n-1}) \pmod{m}. \qquad (2.8)$$

Results with this simple Fibonacci sequence have not been very satisfactory.

3. Random-bit-generator [13].

Let $a=\{a_k\}$ be the sequence of 0's and 1's generated by the linear recursion relation

$$a_k = c_1 a_{k-1} + c_2 a_{k-2} + \ldots + c_n a_{k-n} \pmod{2} \qquad (2.9)$$

for any given set of integers c_i $(i=1,2,\ldots,n)$, each having the value 0 or 1. We require $c_n=1$, and the sequence has degree n. Define a set of numbers of the form

$$Y_K = 0.a_{qK+r-1}\ a_{qK+r-2}\ \ldots a_{qK+r-L} \qquad \text{base } 2 \qquad (2.10)$$

where r is a randomly chosen integer, $0 \leq r \leq 2^7 - 1$ and $L \leq n$. That is, Y_k is the binary expansion of a number whose binary representation is L consecutive digits in a; successive Y_K are spaced q digits apart.

Results with the Random-bit sequence have been pretty satisfactory. Theoretical results illustrating the randomness of this sequence are given in [13].

4. A combination of two congruential generators [10].

This method uses two different generators of the linear congruential type and has one shuffle the sequence produced by the other. Let $\{U_n\}$ and $\{V_n\}$ be the sequences generated by two unrelated congruential generators. We use a table of K locations which was filled with the numbers $U_1, U_2, \ldots, U_k$. Then to generate X_r, the rth random number to be used, we used the first 7 bits of V_r as an index to get X_r from the table. The location of X_r in the table is then filled with the next number from the sequence $\{U_n\}$.

The results of this method can be found in [10]. Knuth [9] thought that this method can be highly recommended. The sequence generated by this method will satisfy virtually anyone's requirements for randomness in a computer-generated sequence.

Shuffling a random number sequence is a good idea. But, until now, no one has been able to find a theory of shuffling. The methods available for shuffling can be found in [9].

CHAPTER III

THE SPECTRAL TEST FOR PSEUDO-RANDOM NUMBER GENERATOR

In this chapter, the spectral test which was formulated by Coveyou & MacPherson [4] will be the topic of study. The spectral test was developed for the evaluation of multiplicative congruential pseudo-random number generators; ie. those of the form

$$X_{n+1} = aX_n \pmod{m}.$$

The spectral test is significant because it not only allows all good random number generators to pass, but it actually fails all linear congruential sequences known to be bad in some other sense. Therefore it is the most powerful test known. The theory behind the test and the algorithm of the test will be presented.

Theory Behind The Spectral Test

The "finite Fourier transform" of a function defined on a finite set is the mathematical motivation on which the spectral test is based. The transform analysis techniques should be understood before the theory behind the spectral test is studied.

Transform Analysis Techniques

In general, people use transform analysis techniques to simplfy problems. For example, the logarithm is a transform which we have all used. Fig. 1 shows a flow diagram that demonstrates the general relationship between conventional and transform analysis procedures. The example problem is to determine the quotient Y = X/Z. Assume that extremely good accuracy is desired and a computer is not available. Conventional analysis implies that Y must be determined by long-hand division. If we must perform the computation of Y repeatedly, then conventional analysis (long-hand division) represents a time consuming process.

The right-hand side of Fig. 1 illustrates the basic steps of transform analysis. As shown, the first step is to convert or transform the problem statement. For the example problem, we choose the logarithm to transform division to a subtraction operation.

Because of this simplification, transform analysis then requires only a table look-up of log(X) and log(Z), and a subtraction operation to determine log(Y). From Fig 1, we next find the inverse transform (anti-logarithm) of log(Y) by table look-up and complete the problem solution. It is clear that by using transform analysis techniques we have reduced the complexity of the example problem.

In general, transforms often result in simplified problem solving analysis. One such transform analysis technique is the Fourier transform. This transform has been

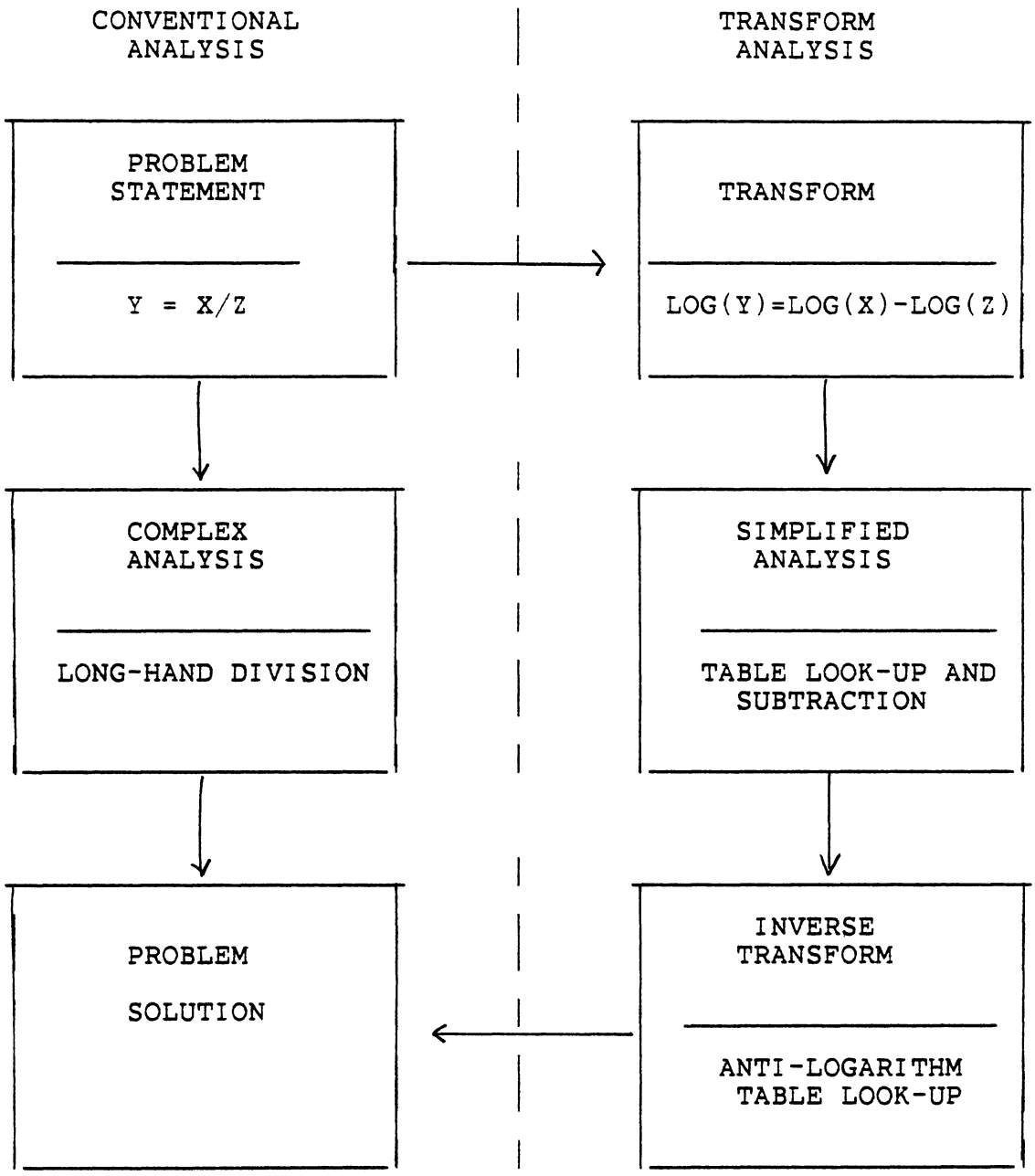

Figure 1. Flow diagram relationship of conventional and transform analysis.

found to be especially useful for problem simplification in many fields of scientific endeavor [1].

Finite Fourier Transform

Given that $F(t_1,t_2,\ldots,t_n)$ is any complex-valued function defined for all combinations of integers t , where $0<t_k<m$ for $1<k<n$, define the Fourier transform of F by the following rule.

$$f(s_1,s_2,\ldots,s_n)=$$

$$\sum_{0\le t_1,\cdots,t_n<m} \exp(-2\pi i(s_1 t_1+\ldots+s_n t_n))\ F(t_1,\ldots t_n) \quad (3.1)$$

The name "transform" is justified here since the original function $F(t_1,\ldots,t_n)$ can be reconstructed from its transform $f(s_1,\ldots,s_n)$ as follows:

$$F(t_1,\ldots,t_n)=$$

$$1/(m**2) \sum_{0\le s_1,\cdots,s_n<m} \exp(2\pi i(s_1 t_1+\ldots+s_n t_n))\ f(s_1,\ldots,s_n) \quad (3.2)$$

The value $(1/(m**2))f(s_1,\ldots,s_n)$ represents the amplitude of an n-dimensional complex plane wave with frequencies $s_1/m,\ldots,s_n/m$, if $F(t_1,\ldots,t_n)$ is written as a superposition of such waves.

As a consequence of relations (3.1) and (3.2), it is possible in theory to determine any property of F from its transform f and conversely.

Applying The Fourier Transform To Random Number Generation

The spectral test applies the Fourier transform concept to random number generation. Suppose that $X_0,X_1,\ldots$ is an infinite sequence of integers with $0<X_K<m$, and let n be a

fixed positive integer.

Define

$$F(t_1,\ldots,t_n)=$$

$$\lim_{N\to\infty} 1/N \sum_{0\le k<n} \delta X_k t_1 \delta X_{k+1} t_2 \ldots \delta X_{k+n-1} t_n, \qquad (3.3)$$

That is, $F(t_1,\ldots,t_n)$ is the limiting density of the number of appearances of the n-tuple $(t_1,\ldots,t_n)$ as n consecutive elements of the sequence $X_0, X_1, \ldots$. Since all sequences $X_0, X_1, \ldots$ are periodic, the assumption is made that the limit in (3.3) exists, and we set N is equal to the period length. In a truly random sequence for the uniform distribution, each possible n-tuple should appear equally often, so $F(t_1,\ldots,t_n)$ should be 1/(m**n) for all $t_1,\ldots,t_n$.

The Fourier transform of (3.3) has the simple form

$$f(s_1,s_2,\ldots,s_n) =$$

$$\lim_{N\to\infty} 1/N \sum_{0\le k<n} \exp\left(-2\pi i \left(s_1 X_k + s_2 X_{k+1} + \ldots + s_n X_{k+n-1}\right)/m\right) \qquad (3.4)$$

In a truly random sequence, this should be the transform of the constant function 1/(m**n); so in a random sequence we should have

$$f(s_1,\ldots,s_n) = \begin{cases} 1, & \text{if } s_1=s_2=\ldots=s_n=0 \pmod m \\ 0, & \text{otherwise} \end{cases} \qquad (3.5)$$

The theoretical test, which finds the average of some function depending only on n consecutive values of the sequence, can be completely determined from the values of $F(t_1,\ldots,t_n)$. Similarly, any theoretical test can be determined from the transformed function $f(s_1,\ldots,s_n)$ in (3.4), since this function carries the same information as

$F(t_1,\ldots,t_n)$ does.

For a linear congruential sequence, defined by a, m, c and X_0, having the maximum period length,

$$f(s_1,\ldots,s_n)= \exp(-2\pi ic((s(a)-s(1))/(a-1))/m)\delta(s(a)/m), \qquad (3.6)$$

where

$$s(a)=s_1+s_2a+s_3(a**2)+\ldots+s_n(a**(n-1))$$

and

$\delta(X)$ is 1 if X is an integer, 0 otherwise.

Equation (3.2) allows us to interpret $f(s_1,\ldots,s_n)/(m**n)$ physically as the amplitude of the n-dimensional complex wave

$$w(t_1,\ldots,t_n)=\exp(2\pi i(((s_1t_1)/m)+\ldots+((s_nt_n)/m))$$

By convention this wave may be assigned a "wave number" v corresponding to its "frequency", where

$$v = \sqrt{(s_1**2+\ldots+s_n**2)} \quad \text{when } |s_k|\leq(m/2) \text{ for } 1<k<n. \qquad (3.7)$$

and the wavelength is

$$L = 1/v$$

According to (3.5), no waves except the constant wave (frequency zero) should appear if the sequence X_0, X_1,... is truly random. An n-tuple consists of n successive members of the sequence and is regarded as a set of co-ordinates of a point in n-space. Marsaglia [11] has shown that if successive n-tuples of uniform (0,1) deviates produced using a multiplicative congruential generator are regarded in this manner, then a fixed number of parallel hyperplanes contains

all points. The distance between these hyper-planes is the wavelength, L , and the latter can therefore be regarded as a measure of the overall "accuracy" of the n-tuples, in the sense that any two n-tuples must be separated by at least this distance. This value therefore gives a measure of pseudo-randomness. It is clear that such a wave is harmful if the wavelength is long, harmless if it is short. Hence, the larger the value of v_n, the better. So if v_n is the smallest nonzero value of the wave number (3.7) for which $f(s_1,\ldots,s_n)\neq 0$ in a linear congruential sequence with maximum period, then the sequence X_0/m, X_1/m, X_2/m,... represents a sequence of random numbers uniformly distributed between 0 and 1, having "accuracy" or "truncation error" $1/v_n$, with respect to the independence of n consecutive values of the sequence averaged over the entire period.

We see that $f(s_1,\ldots,s_n)=0$ except when

$$s_1+s_2 a+s_3 a^{**}2+\ldots+s_n a^{**}(n-1)=0 \quad \text{(modulo m)} \tag{3.8}$$

and in this case $|f(s_1,\ldots,s_n)|=1$. Therefore for linear congruential sequences of maximum period, the smallest nonzero wave number is given by

$$v_n = \min \sqrt{(s_1^{**}2)+(s_2^{**}2)+\ldots+(s_n^{**}2)} \tag{3.9}$$

where the minimum is taken over all n-tuples of integers $(s_1,\ldots,s_n)\neq(0,\ldots,0)$ satisfying (3.8).

Examples Of The Spectral Test

Examples can make the calculation of the spectral test clear. In example 1 the values of a and m are small, so that we can calculate the minimum nonzero values of s easily. Example 2 was given by Knuth [9]; the values of a and m are large, and the values of s are the results from the computer.

Example 1: $a=7$, $m=11$, $c=0$, $X_0=0$

n=2, the minimum nonzero value of $s_1^2 + s_2^2$

for which $s_1 + 7s_2 = 0 \pmod{11}$

occurs for $s_1 = 1$, $s_2 = 3$,

hence $v_2 = \sqrt{10}$

n=3, the minimum nonzero value of $s_1^2 + s_2^2 + s_3^2$

for which $s_1 + 7s_2 + 49s_3 = 0 \pmod{11}$

occurs for $s_1 = -1$, $s_2 = 1$, $s_3 = 1$,

hence $v_3 = \sqrt{3}$

n=4, the minimum nonzero value of $s_1^2 + s_2^2 + s_3^2 + s_4^2$

for which $s_1 + 7s_2 + 49s_3 + 343s_4 = 0 \pmod{11}$

occurs for $s_1 = -1$, $s_2 = 1$, $s_3 = 1$, $s_4 = 0$,

hence $v_4 = \sqrt{3}$

n=5, the minimum nonzero value of $s_1^2 + s_2^2 + s_3^2 + s_4^2 + s_5^2$

for which $s_1 + 7s_2 + 49s_3 + 343s_4 + 2401s_5 = 0 \pmod{11}$

occurs for $s_1 = -1$, $s_2 = 1$, $s_3 = 1$, $s_4 = 0$, $s_5 = 0$

hence $v_5 = \sqrt{3}$

Knuth [9] gave the following example which shows further the meaning of the value of v_n.

Example 2: $X_0 = 0$, $a = 3141592621$, $c = 1$, $m = 10^{**}10$

n=2, the minimum nonzero value of $s_1^2 + s_2^2$

for which $s_1 + 3141592621 s_2 = 0 \pmod{10**10}$

occurs for $s_1 = 67654$, $s_2 = 226$

hence $v_2 = 67654.4$

This means if we want the sequence

$$U_0, U_1, U_2, \ldots = X_0/m, X_1/m, X_2/m, \ldots$$

to represent pseudo-random real numbers between 0 and 1 with adjacent pairs (U_k, U_{k+1}) essentially independent, we have an accuracy of about 1/67654 when the whole period is considered; i.e., the most significant 16 bits in binary notation may be considered random in this sense. Similarly, the minimum nonzero value of $s_1^2 + s_2^2 + s_3^2$ for which

$$s_1 + 3141592621 s_2 + 3141592621^2 s_3 = 0 \bmod 10**10$$

occurs for $s_1 = 227$, $s_2 = 983$, $s_3 = 130$;

hence $v_3 = \sqrt{1034718} = 1017$

Thus when the independence of consecutive triples (U_k, U_{k+1}, U_{k+2}) is considered, we have only about 10 bits of accuracy. When n=4,

the minimum nonzero value of $s_1^2 + s_2^2 + s_3^2 + s_4^2$ for which

$$s_1 + 3141592621 s_2 + 3141592621^2 s_3 + 3141592621^3 s_4 = 0 \bmod 10**10$$

occurs for $s_1 = 52$, $s_2 = -203$, $s_3 = -54$, $s_4 = 125$,

so, $v_4 = \sqrt{62454} = 249.9$

We are now reduced to eight-bit accuracy with respect to independence of successive quadruples.

The values of v_n for n greater than 5 are less important than those for n=1, 2, 3, 4, since complete independence of quintuples is perhaps asking for too much

randomness.

The value v_n therefore gives an indication as to whether the modulus m is sufficiently large for the desired general accuracy of the pseudo-random numbers. Hence, the larger the value of v_n, the better. However, Knuth noted that it is difficult to quantify this value in each case and proposed a "standardizing" transformation. As a measure of randomness, the volume of the ellipsoid in n-space defined by $(mX_1 - aX_2 - a^2X_3 - \ldots - a^{n-1}X_n)^{**}2 + X_2^2 + \ldots + X_n^2 < v_n^2$ serves as an indication of the probability that the integer points $(X_1, X_2, \ldots, X_n)$ are in the ellipsoid. The volume is defined by

$$C_n = (v_n \pi^{\frac{1}{2}})^{**}n \,/[(n/2)!m], \qquad (3.10)$$

where

$$(n/2)! = (n/2)((n/2)-1)\ldots(1/2)\sqrt{\pi} \quad \text{for odd n.}$$

which can be treated in a common fashion for all combinations of a and m. It can be deduced from (3.10) that the larger the value of C_n, the more preferable the generator. If C_2, C_3, C_4 and C_5 are all ≥ 0.1, then a passes the test, and if they are all ≥ 1, it passes the test "with flying colors", to quote Knuth. Merely passing the spectral test does not guarantee sufficient randomness for high-resolution Monte Carlo studies when m is too small.

The detailed description of the theory behind the spectral test can be found in [1,4,5,9].

A Computational Method

The problem of finding v_n is equivalent to finding the minimum value of $(mX_1 - aX_2 - \ldots - a^{n-1}X_n)^{**}2 + X^2 + \ldots + X_n^2$ or to minimizing $X^T QX$ where

$$Q = \begin{bmatrix} m^2 & -mX_2 & -mX_3 & \ldots & -mX_n \\ -mX_2 & 1+X_2^2 & X_2X_3 & \ldots & X_2X_n \\ -mX_3 & X_2X_3 & 1+X_3^2 & \ldots & X_3X_n \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ -mX_n & X_2X_n & X_3X_n & \ldots & 1+X_n^2 \end{bmatrix}$$

If X is a nonzero integer vector for which $X^T QX$ is minimized, and if q is the value of $Y^T QY$ for some nonzero integer vector Y, then $X_k^2 < R_{kk}q$ where $R=Q^{**}-1$.

If U is any integer matrix with determinant ± 1, the minimization problem defined by $X^T Q'X$ where $Q'=(U^{**}-1)^T QU^{**}(-1)$ and $R'=URU^T$ is equivalent to the problem of minimizing $X^T QX$. The matrices

$$U = \begin{bmatrix} 1 & & & & & & \\ & \cdot & & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & 1 & & \\ c_1 & \ldots & c_{k-1} & 1 & c_{k+1} & \ldots & c_n \\ & & & & & 1 & \\ & & & & & & \cdot \\ & & & & & & \cdot \end{bmatrix},$$

$$\begin{bmatrix} & & & \cdot & \\ & & & & 1 \end{bmatrix}$$

$$U^{-1} = \begin{bmatrix} 1 & & & & & & \\ & \cdot & & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & 1 & & \\ -c_1 & \cdots & -c_{k-1} & 1 & -c_{k+1} & \cdots & -c_n \\ & & & & 1 & & \\ & & & & & \cdot & \\ & & & & & & \cdot \\ & & & & & & & \cdot \\ & & & & & & & & 1 \end{bmatrix},$$

where k is some fixed number, $1<k<n$,and c_j = the nearest integer to Q_{jk}/Q_{kk} are used to reduce the original Q and R matrices. When $\prod_{k=1}^{n}(2\sqrt{R_{kk}\,q}+1)<1000$, an exhaustive search is used to find the minimum value of $X^T QX$. This is v_n^2.

Algorithm For the Spectral Test

Knuth [9] formulates the spectral test as a sequence of steps. The procedure in Fig. 2 assumes a, m, and n are given as input, and the quantity $q=v_n^2$ is output. Exact integer operations are implied throughout the computation, and this generally means multiple precision calculations are necessary. In practice, the algorithm in Fig. 2 is surprisingly efficient. However, Knuth pointed out that no

proof has yet been given which establishes that his algorithm terminates, although there are no known cases in which the algorithm has failed to do so. He also noted that this spectral test algorithm only for linear congruential sequences of maximum period. For example, when $m=2^e > 8$, and a mod 8=5 or a mod 8=3, the right way to define v_n in both cases are not clear, since $f(s_1, s_2, \ldots, s_n)$ has different intensities in different bands of the spectrum. For extensions of the test to other linear congruential sequences, see knuth [9] pages 93, 99 and 481. A portable package for the spectral test will be developed in chapter IV.

S1.[Initialize]

1. Set X(1)=1, and X(k+1)=aX(k) mod m for 1<k<n.
   If any X(k)>(m/2) then X(k)=X(k)-m.
2. Form matrices Q and R (see Fig. 3 , R=m Q instead of Q , so that all computations starting from S2 are done with integers.)
   Set Q(1,1)=m**2, R(1,1)= $\sum_{1\le j\le n} X(j)**2$.
   For 1<j<n , set
   Q(1,j)=Q(j,1)=-mX(j),
   R(1,j)=R(j,1)=mX(j),
   Q(j,j)=1+X(j)**2,
   R(j,j)=m**2;
   and for 1<j<k<n, set
   Q(j,k)=Q(k,j)=X(j)X(k),
   R(j,k)=R(k,j)=0.
3. Now set k=n and q=m**2

S2.[Find minimum Q_{jj}]

For 1≤j≤n
if Q(j,j)<q, then set q=Q(j,j).

S3.[Ready for exhaustive search?]

For 1<j<n set
c(j)= $\lfloor \sqrt{qR(j,j)}/m \rfloor$
if $\prod_{1\le j\le n} (2c(j)+1)<1000$, go to S6.

S4.[Transform]

1. For 1<j<n, j=k, set
   c(j)= $\lfloor Q(j,k)/Q(k,k)+(1/2) \rfloor$

Figure 2. Knuth's Spectral Test Algorithm

2. For 1<j<n, j=k, c(j)=0, then

TRANS(Q,j,k,-c(j)) and TRANS(R,k,j,c(j)).

The TRANS(P,i,j,t) operation is defined as follows:

For 1<r<n set P(i,r)=P(i,r)+tP(j,r)

For 1<r<n set P(r,i)=p(r,i)+tP(r,j)

S5.[Modify k]

k=k-1 if k=0 then

k=n go to S2.

S6.[Prepare for search]

1. k=n

2. for 1<j<n set X(j)=0

S7.[Advance X(k).]

X(k)=X(k)+1.

If X(k)>c(k) go to S9

S8.[Advance k]

k=k+1.

If k<n then

X(k)=-c(k)

go to S8

else

$q' = \sum_{1 \le j \le n} \sum_{1 \le j \le n} X(i)X(j)Q(i,j)$

if q'<q set q=q'

S9.[Decrease k]

Set k=k-1.

If k>1 then go to S7

STOP

Figure 2. Knuth's Spectral Test Algorithm (cont)

$$R = \begin{bmatrix} \Sigma X_j^2 & mX_2 & mX_3 & \dots & mX_n \\ mX_2 & m^2 & 0 & \dots & 0 \\ mX_3 & 0 & m^2 & \dots & 0 \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ mX_n & 0 & 0 & \dots & m^2 \end{bmatrix}$$

Figure 3. Knuth's R Matrix in The Spectral Test

CHAPTER IV

COMPUTER IMPLEMENTATION AND RESULTS

The main purpose of this thesis is to implement Knuth's spectral test algorithm. As the previous chapter described that all test data are large, so multiple precision calculations are necessary. The computer implementation of Knuth's spectral test algorithm includes two parts. The first part is to build up a portable integer multiple precision package to perform the arithmetic operations. Then write a spectral test package that calls the integer multiple precision arithmetic operations. Both packages were written in a portable subset of the 1966 A.N.S.I. Standard FORTRAN and A.N.S.I. FORTRAN 77 languages.

Integer Multiple Precision Operations Package

The purpose of the integer multiple precision arithmetic operation package is to allow a Fortran program to work with integer numbers of arbitrary precision, and to work in any computer. A multiple precision integer means the integer can occupy several computer words. This package is presently dimensioned to handle numbers that occupy up to 50 computer words. If a larger number is desired, the user

should do the following things:

1. The new value of NDGT should replace the value 50 of NDGT in subroutine PREPR.

2. The new value of NDGT should replace the dimension 50 of all arrays in the DIMENSION statement in all subroutine.

The other important parameter of the integer multiple precision operation package is the base of the word. Because decimal arithmetic operations are most popular, this package was designed to handle any word base 10**a, where a is a positive integer. The user has to input the base value. But base**2 has to be less than the maximum integer that can be stored in computer word; otherwise overflow would occur in the multiplication subroutine. In the test program we select base equal to 100, because we ran the program in an IBM 1130. In the IBM 1130, each word is 16 bits, so the largest fixed point number can be stored in one word is 32767. The smallest fixed point number that can be stored in one word is -32768. If the base is 100, then the largest value that can be stored in one word is 99. The results of multiplication of two words will not exceed 10000. Overflow will not happen in this case.

Storage Format of MPI

The subroutines were written to be called from a Fortran main program and to work with integer numbers in MPI form. All integer numbers used as MPI form must initially be defined by using the subroutines INPDT or KINPT. The

format of the MPI is as follows: the first (lowest-subscripted) word of the array contains the lowest order digits of the number in base NBASE, the second word contains the next lowest digits of the number in base NBASE , and so on. The reason we store the lower order digits first because addition, subtraction, and multiplication all start from lower order digits. MPI(NDGT) is the last word in MPI. In this test package NDGT is set to 50. But user can change this value easily. MPI(NDGT) contains the sign of the number. MPI(NLTH) is the word before MPI(NDGT), and it contains the number of words occupied by the integer number. When the integer numbers are greater or equal to zero, MPI(NDGT)=1, otherwise MPI(NDGT)=-1. The format of MPI can be found in Fig. 4.

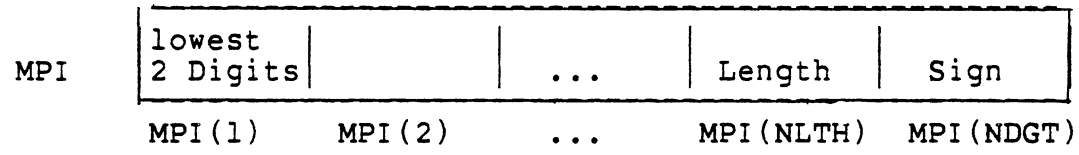

Figure 4. Storage Format of MPI

For example, for a FORTRAN integer -1034567, its MPI form will like in Fig. 5. The integer number occupied 4 words, so MPI(NLTH)=4. The number is less than zero, so

MPI(NDGT)=-1. The maximum FORTRAN integer that can be stored in MPI form in this test program contains 96 digits.

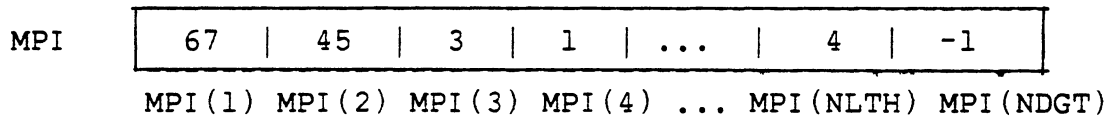

Figure 5. MPI Form of Integer -1034567

Input Data

Subroutine INPDT will read in a FORTRAN integer and convert it to MPI form. Each input line will contain one number in free format. Illegal characters in input line will be detected, and the program will print an error message and terminate.

Output Data

Subroutine OUTPT will transform a MPI number to a FORTRAN integer. Each output number will contain sign, length, and value. For example, if the input data has MPI form in Fig. 5, its output will present in Fig. 6.

Sign	Length	Value
-1	4	1034567

Figure 6. Output Format of -1034567

Use Of Multiple Precision Integer Package

The main program calling the integer multiple precision arithmetic package should contain one COMMON statement. The COMMON statement has the following form.

```
COMMON   /MULT/NDGT,NBASE,NLTH,NTEST,LP
```

Before calling the subroutines in the package, the main program should call subroutine PREPR to give the initial values to common variables and read in the value for the base. The calling statement is

```
Call PREPR(NULL)
```

The integer multiple precision arithmetic operations package contains 14 subroutines. A list of the subroutines and the description of the subroutines can be found in Table VI. For more detailed descriptions of the subroutines refer to the following pages.

PREPR. Calling statement : Call PREPR(I)
Where I is an indicator. I is returned equal to zero, if base value is legal. I equals to 1, if base value is not legal. The base should be a positive number, and base**2 should be less than the computer word size. The subroutine gives values to all common variables which appear in the

TABLE VI

SUBROUTINES AND DESCRIPTION OF INTEGER MULTIPLE PRECISION ARITHMETIC OPERATIONS PACKAGE

Name	Function
PREPR	Give Initial Values To Common Variables.
INPDT	Read In A FORTRAN Integer And Convert It To Multiple Precision Integer.
KINPT	Convert A Given FORTRAN Integer To a Multiple Precision Integer.
OUTPT	Print a Multiple Precision Integer As Sign, Length, And Value.
MADD	Addition.
MSUB	Subtraction.
MMPY	Multiplication.
MDIV	Division.
MSDV	Multiple Precision Integer Divided By a Integer Which is Less Than Nbase.
MCMP	Compare Two Multiple Precision Integer.
MSQRT	Find Nearest Square Root Of Multiple Precision Integer.
MFLT	Translate Floating Point Number To Multiple Precision Integer.

COMMON statement.

INPDT. Calling statement : Call INPDT(I)
I is a multiple precision integer. It is an output parameter. The subroutine will read in a FORTRAN integer and convert it to a multiple precision integer and store the result in I. If overflow occurs then higher-order digits of the result will be truncated.

KINPT. Calling statement : Call KINPT(I,J,K)
Where I and K are given Fortran integer, and J is a multiple precision integer. The value in I is converted to a multiple precision integer and is stored in J. K is an indicator; if K=0 then I is a new number; if K>0 then I is multiple precision integer and is stored in J.

OUTPT. Calling statement : Call OUTPT(I)
I is a multiple precision integer. The value in I will be printed as sign, length, and value.

MADD/MSUB. Calling statement : Call MADD(I,J,K)
or Call MSUB(I,J,K)
I, J, and K are multiple precision integers. For MADD, the sum I+J is placed in K; for MSUB, the difference I-J is placed in K. I, J,and/or K may refer to the same storage locations if desired. An overflow results whenever the result exceeds the words of K. In such a case as many as possible of the rightmost digits of the result are placed in K, together with the correct sign, and a warning message is

printed. The higher-order digits are lost.

MMPY. Calling statement : Call MMPY(I,J,K)
I, J, and K are multiple precision integers. I, J and/or K may denote the same storage location. The number in I is multiplied by the number in J, and the product is stored in K. The sign assigned to K is determined by the rules of algebra. If K has insufficient capacity to contain the product, an error message is printed and the excessive higher-order digits are lost.

MSDV. Calling statement : Call MSDV(I,J,K,L)
I and K are multiple precision integers; J and L are integers less than NBASE. The number in I is divided by the number in J, which for proper results should be less than NBASE. The quotient is placed in K and the remainder in L. The sign of the quotient is determined by the rules of algebra, while the remainder is assigned the same sign as the dividend, except that a zero quotient or remainder is always assigned a positive sign. If division by zero is attempted, an error message is printed and K and L are left in an undetermined condition.

MCMP. Calling statement : Call MCMP(I,J,K)
I and J are multiple precision integers. K is an integer. The numbers in I and J are algebraically compared. If I is greater than J, K is set to 1; if I=J, K is set to 0; if I is less than J, K is set to -1.

MSQRT. Calling statement : Call MSQRT(I,J)
I and J are multiple precision integers. The approximate square root of I is stored in J. If the square root of I is a real number which contains integer part and fraction part, we only stored the integer part of the square root in J. I and J can have the same storage location.

MFLT. Calling statement : Call MFLT(X,I)
X is a real number; I is a multiple precision integer. The approximate integer value of X is converted to a multiple precision integer stored in I. That means we only convert the integer part of X to a multiple precision integer and store it to I. If overflow occurs then an error message will be printed and higher-order digits of result will be truncated.

Spectral Test Subroutine Package

This package contains a calling program, subroutine SPEC , and subroutine TRANS. The calling program controls the spectral analysis. It calls SPEC which finds v and c. TRANS transforms the matrices Q and R. The package was written to be used with the integer multiple precision subroutine package for all computers which have a FORTRAN compiler.

Calling Program

This calling program must contain one INTEGER and COMMON statements as follows:

```
INTEGER  A(50), M(50)
COMMON   /IMULT/NDGT,NBASE,NLTH,NTEST,LP
```

The main functions of this program are following:

1. Input the number of tuples to be test, and input the number of groups to be tested.
2. Open the integer multiple precision subroutines package.
3. Input the multiplier and modulus to be tested.
4. Print the heading, and the values of A and M.
5. Call SPEC. The way to input A and M was described in the previous section.

Algorithm of Calling program:

```
S1:  Input ITUPLE and NTEST
      READ (IN,*) ITUPLE,NTEST
S2:  Open the integer multiple precision subroutine package
      CALL PREPR(NULL)
S3:  Input the value of A and M, output A and M, and
     call SPEC
      Do this step NTEST times
         CALL INPDT(A)
         CALL INPDT(M)
         CALL OUTPT(A)
         CALL OUTPT(M)
         CALL SPEC(A,M,ITUPLE)
```

Spectral Test Subroutine

Calling statement: Call SPEC(A,M,ITUPLE)

This subroutine implements Knuth's algorithm to calculate V_n and C_n of the generator. The value of n was decided by ITUPLE which was passed from the calling program. As presently dimensioned, the procedure can test the statistical independence of 2-tuples through 5-tuples. If the testing of more than 6-tuples is desired, the dimension 6 or higher value should replace the dimension 5 in the single, double, and triple dimension arrays in SPEC, TRANS , MEQU, MTEQU of spectral test subroutine package. The detailed description of SPEC can be found in the previous chapter (p.32), or in reference [5,7,9]. The program listing can be found in Appendix A.

Test and Verification

The portable spectral test package was tested with the 18 modulus-multiplier sets which were given by Knuth [9], and 19 modulus-multiplier sets which were given by Hwang [7]. All test sets were tested from 2-tuple through 5-tuple. The correct values of C_n for the first 18 sets were given by Knuth. The values of C_2, C_3, C_4 calculated by the spectral test package have the same values as Knuth's. The other 19 test sets were selected from Hwang's report. But only 18 test sets have the values of C_2, C_3, C_4 from this package agreed with those values which given by Hwang's to the needed accuracy. One test set has value of C_2 from this package different from the value of C_2 in Hwang's report. Hwang's program only runs on an IBM 1130 and no machine is

now available on which to resolve the disagreement. All values of C_n of test data can be found in Table VII and Table IX. Thirty-seven pairs of multiplier and modulus were used to test the spectral test package. Satisfactory results were produced from the WATFIV and FORTRAN G1 compilers on IBM 360/370, and from the 66 and 77 Standard FORTRAN V compiler on IBM 3081.

Result

Generators From Knuth

The first 18 test data sets which were given by Knuth contain bad and good multipliers. Line 1 through line 3 of TABLE VII show that these generators have very poor values of C_2, C_3, C_4, and C_5. They failed the spectral test because line 1 has a too small multiplier; multipliers in line 2 and line 3 have the simple form (2**k)+1; both are congruent to 1 mod 8, and have low potency.

From line 4 through line 11, the generators have a maximum length period. The multipliers are congruent to 5 mod 8 for m=2**35; they also are congruent to 21 mod 200 for m=10**10. They all have potency higher than 9. They all passed the spectral test with value C_2, C_3, C_4,and C_5 higher than .1, except that on line 5 the value of C_3 is 0.06. But the same multiplier for m=2**35 in line 9 has C_3 equals to 1.7.

Line 12-16 show multipliers that all are congruent to 5 modulo 8 and have potency higher than 15. The special thing

about these is the multipliers with only four 1's in their binary representation. All generators in this group failed the spectral test.

Line 17 and line 18 show generators that have been used extensively since they were suggested by O. Taussky [12] in the early 1950's. Both generators passed the spectral test and have values of C_2, C_3, C_4, and C_5 greater than 1.

Generators From Hwang

In this test group, from line 1 to line 8 were generators which were suggested by Hwang. Only the generator in line 3 failed the spectral test. The multiplier 87381 has binary representation 10101010101010101. The value of C_2 in line 6 is 2.8695. It is different from Hwang's 1.29. Line 9 is a generator known to be a bad generator [3], but this generator passed the spectral test by Hwang, passed Chandler's [3] Poisson generator test, and passed this spectral test too. Lines 12, 14, and 15 are Van Gelder's [14] generators, but two of them (line 14 and 15) failed the spectral test.

TABLE VII

LIST OF RESULTS OF FIRST TEST GROUP OF
THE PORTABLE SPECTRAL TEST PACKAGE

Line	A	M	C_2	C_3	C_4	C_5
1	23	10^8+1	0.000017	0.000511	0.013862	0.340398
2	2 +1	2**35	0.000002	0.000262	0.039777	4.657990
3	2 +1	2**35	3.141590	.000000002	.000000002	.000000005
4	3141592653	2**35	0.274043	0.126704	0.111172	0.006403
5	3141592221	10**10	1.348961	0.060825	4.686328	0.346005
6	3141592421	10**10	2.684311	0.343210	0.539159	0.593907
7	3141592621	10**10	1.437942	0.440881	1.924817	0.069969
8	3141592821	10**10	0.158409	2.926201	0.166419	0.092522
9	3141592221	2**35	1.237976	1.700704	1.115594	2.793342
10	3141592621	2**35	3.022285	0.168019	1.258907	0.329183
11	2718281821	2**35	2.591037	1.164212	1.749656	4.245553
12	8392709	2**35	0.015316	2.785573	0.066241	5.605964
13	8396805	2**35	0.015382	1.502379	0.066241	0.027679
14	8404997	2**35	1.120608	1.673829	0.066241	3.134601
15	4202501	2**35	0.749877	0.301854	0.066241	0.819472
16	16777216	2**35	0.000764	2.948927	0.066241	5.531194
17	12207031	2**35	3.032068	0.609932	1.846061	2.991829
18	30517578125	2**35	2.018722	4.015088	4.032269	0.399966

TABLE VIII

LIST OF RESULTS OF KNUTH"S SPECTRAL TEST

Line	A	M	C_2	C_3	C_4
1	23	10^8 +1	0.000017	0.00051	0.014
2	2 +1	2**35	0.000002	0.00026	0.040
3	2 +1	2**35	3.14	.000000002	.000000003
4	3141592653	2**35	0.27	0.13	0.11
5	3141592221	10**10	1.35	0.06	4.69
6	3141592421	10**10	2.68	0.34	0.54
7	3141592621	10**10	1.44	0.44	1.92
8	3141592821	10**10	0.16	2.93	0.17
9	3141592221	2**35	1.24	1.70	1.12
10	3141592621	2**35	3.02	0.17	1.26
11	2718281821	2**35	2.59	1.16	1.75
12	8392709	2**35	0.015	2.78	0.066
13	8396805	2**35	0.015	1.48	0.066
14	8404997	2**35	1.12	1.66	0.066
15	4202501	2**35	0.75	0.30	0.066
16	16777216	2**35	0.0008	2.92	0.066
17	12207031	2**35	3.03	0.61	1.85
18	30517578125	2**35	2.02	4.02	4.0311

TABLE IX

LIST OF RESULTS OF SECOND TEST GROUP OF
THE PORTABLE SPECTRAL TEST

line	A	M	C_2	C_3	C_4	C_5
1	45563	2**31	3.036996	3.764995	3.916921	0.876102
2	42845	2**31	2.685468	1.831955	4.617005	0.979160
3	87381	2**31	0.981747	0.000003	0.000031	0.000355
4	42844	2**31	2.685342	2.484599	3.804648	3.313031
5	909	2**15	0.309864	0.377954	5.210087	6.125197
6	173	2**15	2.869500	1.542116	1.274661	1.429904
7	213	2**15	2.384571	1.310845	0.656005	0.287358
8	205	2**15	2.552542	2.500594	2.390893	0.453290
9	4709	8388608	2.510170	0.580333	0.588273	0.883261
10	44653	2**31	2.916896	2.992396	2.063463	1.628019
11	231525	2**31	3.059750	0.819579	4.753134	0.372642
12	282629	2**31	1.459670	0.439157	0.955374	0.174666
13	253125	2**31	1.323086	0.575003	2.826895	0.054096
14	34821	2**31	1.773790	0.012394	0.196788	2.052188
15	65541	2**31	3.140440	0.000038	0.001211	0.034810
16	333	2**15	2.642280	0.657601	2.390893	2.062906
17	421	2**15	1.053077	0.445892	4.352281	1.625540
18	209	2**15	2.557336	1.335919	3.036657	1.429904
19	197	2**15	3.059522	0.356153	1.219843	2.062906

TABLE X

LIST OF RESULTS OF HWANG'S SPECTRAL TEST

line	A	M	C_2	C_3	C_4
1	45563	2**31	3.04	3.76	3.92
2	42845	2**31	2.69	1.83	4.62
3	87381	2**31	0.98	0.0000025	0.000031
4	42844	2**31	2.69	2.48	3.80
5	909	2**15	0.31	0.38	5.21
6	173	2**15	1.29	1.54	1.2
7	213	2**15	2.38	1.31	0.66
8	205	2**15	2.55	2.50	2.39
9	4709	8388608	2.51	0.58	0.59
10	44653	2**31	2.92	2.99	2.06
11	231525	2**31	3.05	0.82	4.75
12	282629	2**31	1.46	0.44	0.96
13	253125	2**31	1.32	0.58	2.83
14	34821	2**31	1.77	0.012	0.197
15	65541	2**31	3.14	0.000038	0.0012
16	333	2**15	2.64	0.66	2.39
17	421	2**15	1.05	0.45	4.35
18	209	2**15	2.56	1.34	3.04
19	197	2**15	>1	0.36	1.22

CHAPTER V

SUMMARY AND CONCLUSIONS

The three tasks which have been accomplished in this thesis are:

1. Learn how to choose good linear congruential pseudo random number generators.
2. Write and test a portable software package to perform integer arithmetic with arbitrary precision.
3. Write and test a portable spectral test package to rate the choice of a multiplier, a, in a linear congruential sequence of maximum period.

The linear congruential sequence can be generated in the following form:

$$X_{n+1} = (aX_n + c) \pmod{m} \qquad (5.1)$$

A good generator should have the following properties.

1. The number X_0 may be chosen arbitrarily.
2. The number m should be large. It may conveniently be taken as the computer's word size, since this makes the computation of $(aX+c) \pmod{m}$ quite efficient.
3. If m is a power of 2, pick a so that $a = 5 \pmod{8}$; if m is a power of 10, choose a so that $a = 21 \pmod{200}$. This choice of a together with the choice of c given below ensures that the random number generator has maximum length

and high potency. The digits in the binary or decimal representation of a should not have a simple, regular pattern.

4. The constant c should be an odd number, when m is a power of 2; and also not be a multiple of 5, when m is a power of 10.

The multiplicative congruential sequence of deviates is characterized by the multiplier a and the modulus m, and the spectral test evaluates the spatial distribution of n-tuples of this sequence, based on these two variables for various values of n. An n-tuple consists of n succesive members of the sequence and is regarded as a set of coordinates of a point in n-space. If successive n-tuples of uniform (0,1) deviates produced using a multiplicative congruential generator are regarded in this manner, then a fixed number of parallel hyperplanes contains all points. The distance between these hyperplanes is the inverse of a quantity known as the "wave number", V_n, and the latter can therefore be regarded as a measure of the overall "accuracy' of the n-tuples, in the sense that any two n-tuples must be separated by at least this distance. This value therefore gives an indication as to whether the modulus, m, is sufficiently large for the desired accuracy of the pseudo-random numbers. Hence, the larger the value of V_n, the better. Knuth proposed a "standardizing" transformation of V_n, namely

$$C_n = ((V_n \sqrt{\pi})^{**}n)/((n/2)!m), \qquad (5.2)$$

which can be treated in a common fashion for all

combinations of a and m. It can be deduced from Eq. 5.2 that the larger the value of C , the more preferable the generator. Quantitatively, it can be said that if C_2, C_3, C_4, and C_5 are all ≥ 0.1 then a passes the test.

To build up a portable spectral test package, first build up a portable integer multiple precision arithmetic operation package to support the spectral test subroutine. The test results of the portable spectral test subroutine are shown in Table VII and Table IX. The results of two test groups agreed with Knuth's and Hwang's to the needed accuracy, with one exception noted. The package has been tested from the WATFIV and G1 compilers on IBM 360/370, and from the 66 abd 77 Standard FORTRAN V compiler on IBM 3081. The package can be run in any computer with a FORTRAN compiler.

A SELECTED BIBLIOGRAPHY

[1] Brigham, E. O. The Fast Fourier Transform. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1974.

[2] Certaine, J. "On Sequences of Pseudo-random Numbers of Maximal Length." JACM 5 (March, 1958), 353-356.

[3] Chandler, J. P. Unpub. Communication, Oklahoma State University, 1971.

[4] Coveyou, R. R. and Macpherson, R. D. "Fourier Analysis of Uniform Random Number Generators." JACM 14 (1967), 100-119

[5] Golder, E. R. "The Spectral Test for the Evaluation of Congruential Pseudo-random Generators;" Algorithm AS 98. Appl. Statist. 25 (1976), 173-192.

[6] Hull, T. E. and Dobell, A. R. "Random number Generaors." SIAM Review 4 (1962), 230-254.

[7] Hwang, J. C. "The Spectral Test for the IBM/1130." Unpub. Report, Oklahoma State University, 1971.

[8] Jones, B. W. The Theory of Numbers. Holt, Rinehart and Winston. New York, Chicago (1966).

[9] Knuth, D. E. The Art of Computer Programming, Vol. 2: Seminumerical Algorithms. 2nd ed. Reading, MA: Addison-Wesley Publishing Co., 1973.

[10] Maclaren, M. D. and Marsaglia, G. "Uniform Random Number Generators." JACM 12 (1965), 83-89.

[11] Marsaglia, G. "Random Numbers Fall Mainly in the Planes." Proc. Nat. Acad. Sci. U.S.A. 61 (1968), 25-28.

[12] Taussky, O. and Todd, J. "Generation of Pseudo Random Numbers." Symposium on Monte Carlo Methods. Ed. Herbert A. Meyer. March (1954), 15-28.

[13] Tausworthe, R. C. "Random Numbers Generated by Linear Recurrence Modulo Two" Math. Comp. 19 (1965), 201-209.

[14] Van Gelder, A. "Some New Results in Pseudo-Random Number Generation" JACM 14(1967), 785-792.

APPENDIX A

LISTING OF COMPUTER PROGRAM

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                            * 0000000
C     STEST IS THE CALLING PROCEDURE OF SPEC ROUTINE                         * 0000000
C                                                                            * 0000000
C     PURPOSES ·                                                             * 0000000
C              1. INPUT BASE VALUE FOR MPI PACKAGE                           *
C              2. INPUT THE MULTIPLIER AND MODULUS OF MULTIPLICATIVE         * 0000000
C                   CONGRUENTIAL GENERATOR                                   * 0000000
C              3. DECIDE # OF TUPLES TO BE CALCULATED                        * 0000000
C              4  DECIDE # OF GROUPS OF TEST DATA                            * 0000000
C              5. CALL SPEC ROUTINE TO EXECUTE SPECTRAL TEST                 * 0000000
C                                                                            * 0000000
C                                                                            * 0000000
C    VARIABLES :                                                             * 0000000
C              1. NBASE  . BASE VALUE FOR MPI                                *
C              2. ITUPLE · # OF TUPLES OF EACH MULTIPLICATIVE                * 0000000
C                          CONGRUENTIAL GENERATOR                            * 0000000
C              3. NTEST  · # OF GROUPS OF TEST DATA                          * 0000000
C              4. A      : THE MULTIPLIER OF MULTIPLICATIVE                  * 0000000
C                          CONGRUENTIAL GENERATOR                            * 0000000
C              5. M      : MODULUS OF MULTIPLICATIVE CONGRUENTIAL            * 0000000
C                          GENERATOR OF MAXIMUM PERIOD                       * 0000000
C                                                                            * 0000000
C                                                                            * 0000000
C    INPUT     :                                                             * 0000000
C              1. NBASE FREE FORMAT                                          * 0000000
C              2  ITUPLE, NTEST FREE FORMAT                                  *
C              3. A    ANY FORTRAN INTEGER WITH FREE FORMAT                  * 0000000
C              3. M    INPUT FORMAT AS ABOVE                                 * 0000000
C                                                                            * 0000000
C                                                                            * 0000000
C    OUTPUT    :                                                             * 0000000
C              1  A· USE SUBROUTINE OUTPT TO PRINT A FORTRAN                 * 0000000
C                    NUMBER CONTAINS SIGN, LENGTH, VALUE                     * 0000000
C              2  M: USE SAME FORMAT AS ABOVE                                * 0000000
C              3. V· VALUE OF V(N), A REAL NUMBER                            *
C              4. C· VALUE OF C(N), A REAL NUMBER                            *
C                                                                            * 0000000
C                                                                            * 0000000
C    SUBROUTINES CALLED :                                                    * 0000000
C              1. INPDT · INPUT A AND M                                      * 0000000
C              2  OUTPT · OUTPUT A AND M                                     * 0000000
C              3. SPEC  : SPECTRAL TEST                                      * 0000000
C                                                                            * 0000000
C                                                                            * 0000000
C                                                                            * 0000000
C    ADVISOR    · DR  J  P  CHANDLER                                         * 0000000
C    PROGRAMMER : CHU, CHUNG-PEI               DATE : SUMMER 1986            * 0000000
C    LOCATION   · OKLAHOMA STATE UNIVERSITY                                  * 0000000
C                                                                            * 0000000
C                                                                            * 0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                              0000000

         INTEGER A(50),M(50)                                                   0000000
         DATA  PI/3.14159/,IFLAG1,IFLAG2/1,2/                                  0000000
         COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                                0000000
C                                                                              0000000
C                                                                              0000000
C   INPUT ITUPLE AND NTEST                                                     0000000
C   PRINT HEADING                                                              0000000
C   READ IN A AND M , ECHO CHECK INPUT DATA                                    0000000
C   CALL SPEC ROUTINE                                                          0000000
C                                                                              0000000
C                                                                              0000000
          IN=5
          LP=6
          NDGT=50
          READ(IN,*)NBASE
          READ(IN,*)ITUPLE,NTEST                                               0000000
          CALL PREPR(NULL)                                                     0000000
          DO 50 JK=1,NTEST                                                     0000000
             WRITE(LP,10)                                                      0000000
   10        FORMAT(1H1)                                                       0000000
             CALL INPDT(A)                                                     0000000
             CALL INPDT(M)                                                     0000000
             WRITE(LP,20)                                                      0000000
   20        FORMAT(///,5X,'VALUE OF A',/2X,'SIGN',2X,'LENGTH',                0000000
     1       2X,'VALUE')                                                       0000000
             CALL OUTPT(A)                                                     0000000
             WRITE(LP,30)                                                      0000000
   30        FORMAT(///,5X,'VALUE OF M',/2X,'SIGN',2X,'LENGTH',2X,             0000000
     1       'VALUE')                                                          0000000
             CALL OUTPT(M)                                                     0000000
             WRITE(LP,40)                                                      0000000
   40        FORMAT(///,5X,'VALUE OF V**2',/2X,'SIGN',2X,'LENGTH',2X,          0000000
     1       'VALUE',15X,'SUBSCRIPT',3X,'VALUE OF V',5X,'VALUE OF C')          0000000
             CALL SPEC(A,M,ITUPLE)                                             0000000
   50     CONTINUE                                                             0000000
         WRITE(LP,100)                                                         0000000
  100    FORMAT(1H1)                                                           0000000
  200    STOP                                                                  0000000
         END                                                                   0000000
```

```
      SUBROUTINE SPEC(A,M,ITUPLE)                                       0000000
C*****************************************************************      0000000
C                                                                *      0000000
C                                                                *      0000000
C                    SPECTRAL TEST                               *      0000000
C                                                                *      0000000
C   PURPOSE                                                      *      0000000
C        1  CALCULATES THE MINIMUM OF THE QUANTITY               *      0000000
C           TEMP=(M*X(1) -A*X(2)-      -A**(N-1)*X(N))**2       *      0000000
C           V(N)**2=TEMP+X(1)**2 +      +X(N)**2                *      0000000
C        2  C(N)=((PI**(N/2))*(V(N)**N))/((N/2)!*M)              *      0000000
C                                                                *      0000000
C           IN A LINEAR CONGRUENTIAL SEQUENCE WITH MAXIMUM       *      0000000
C        PERIOD DEFINED BY A MULTIPLIER 'A' AND MODULUS 'M'      *      0000000
C        . THE SEQUENCE X(1)/M,X(2)/M,   ,REPRESENTS A           *      0000000
C        SEQUENCE OF RANDOM NUMBERS UNIFORMLY DISTRIBUTED        *      0000000
C        ON (0,1) WITH A TURE ERROR 1/V(N). WITH RESPECT         *      0000000
C        TO THE INDEPENDENCE OF N CONSECUTIVE VALUES             *      0000000
C        SEQUENCE AVERAGED OVER THE ENTIRE PERIOD                *      0000000
C                                                                *      0000000
C        THIS PROCEDURE WAS IMPLEMENTED FOLLOWING KNUTH'S        *      0000000
C        SEMINUMERICAL ALGORITHMS(THE ART OF COMPUTER            *      0000000
C        PROGRAMMING,1969,PP 93-95)                              *      0000000
C                                                                *      0000000
C                                                                *      0000000
C   PARAMETERS                                                   *      0000000
C        1. A  : THE MULTIPLIER OF MULTIPLICATIVE CONGRUEN-      *      0000000
C                TIAL GENERATOR.                                 *      0000000
C        2. M    MODULUS OF MULTIPLICATIVE CONGRUENTIAL          *      0000000
C                GENERATOR OF MAXIMUM PERIOD                     *      0000000
C        3  ITUPLE   MAXIMUM # OF TUPLES OF THE SEQUENCE         *      0000000
C                TO BE TEST.                                     *      0000000
C                                                                *      0000000
C                                                                *      0000000
C   VARIABLES                                                    *      0000000
C        1  X   X(I+1)=A*X(I), MODE M                            *      0000000
C        2  Q  - INITIALLY CONTAINS THE COEFFICIENTS OF          *      0000000
C                THE SYSTEM OF EQUATIONS GENERATED WHILE         *      0000000
C                MINIMIXING THE QUANTITY DESCRIBED ABOVE         *      0000000
C        3  R    M**2*Q**(-1)                                    *      0000000
C        4. PROD   THE EXPECTED LENGTH OF SEARCH TO FIND         *      0000000
C                  THE ABSOLUTE MINIMUM OF THE QUANTITY          *      0000000
C                  DESCRIBED ABOVE                               *      0000000
C        5  Q3   V**2                                            *      0000000
C        6  TEMP,TEMP1,TEMP2   TEMPORARY STORAGE                 *      0000000
C                                                                *      0000000
C                                                                *      0000000
C   SUBPROGRAM CALLED                                            *      0000000
C        1. KINPT                  2  MEQU                      *      0000000
C        3  MADD                   4  MTEQU                     *      0000000
C        5  MSUB                   6  MSQRT                     *      0000000
C        7  MMPY                   8  MCMP                      *      0000000
C        9  MDIV                   10 OUTPT                     *      0000000
C       11  TRANS                                                *      0000000
C                                                                *      0000000
C                                                                *      0000000
C   LIMITATION                                                   *      0000000
C        THIS PROCEDURE MUST BE SUPPORTED BY INTEGER             *      0000000
C        MULTIPLE PRECESION ARITHMETIC OPERATIONS                *      0000000
C                                                                *      0000000
C                                                                *      0000000
C                                                                *      0000000
C   ADVISOR     - DR  J  P  CHANDLER                             *      0000000
C   PROGRAMMER    CHU, CHUNG-PEI                                 *      0000000
C   DATE        - SUMMER 1986                                    *      0000000
C   LOCATION      OKLAHOMA STATE UNIVERSITY                      *      0000000
C                                                                *      0000000
C                                                                *      0000000
C*****************************************************************      0000000
 2        INTEGER  A(50) M(50),Q3(50),Q(5,5,50),R(5,5,50),               0000000
     1    X(5,50),C1(5,50),TEMP(50),TEMP1(50),TEMP2(50),DC(50),Q2(50),   0000000
     2    ONE(50),TWO(50),ZERO(50),SUM(50),THOUND(50),PROD(50),NQ(50),    0000000
     3    NR(50),EX(5,50),C2(5,50),TEMP3(50)                             0000000
 3        REAL*8 V(5),C(5),DTEMP,DM                                      0000000
 4        DATA IFLAG1,IFLAG2/1,2/,PI/3 14159/                            0000000
 5        COMMON     /IMULT/NDGT,NBASE,NLTH,LP,IN                        0000000
CCCCC                                                                   0000000
C    STEP1 INITIALIZATION                                               0000000
CCCCC                                                                   0000000
 6        CALL KINPT(0,ZERO,0)                                           0000000
 7        CALL KINPT(1,ONE,0)                                            0000000
 8        CALL KINPT(2,TWO,0)                                            0000000
 9        CALL KINPT(1000,THOUND,0)                                      0000000
10        CALL MDIV(M,TWO,TEMP1,TEMP2)                                   0000000
11        CALL MEQU(X,IFLAG1,ONE,IFLAG1)                                 0000000
C                                                                       0000000
C   SET X MATRIX                                                        0000000
C                                                                       0000000
12        NPT=ITUPLE-1                                                  0000000
13        DO 10 K=1,NPT                                                 0000000
14           CALL MEQU(X,K,TEMP,IFLAG2)                                 0000000
15           CALL MMPY(A,TEMP,TEMP2)                                    0000000
16           CALL MDIV(TEMP2,M,NQ,TEMP)                                 0000000
17           IK=K+1                                                     0000000
18           CALL MCMP(TEMP,TEMP1,IR)                                   0000000
19           IF(IR)9 9,8                                                0000000
20    8      CALL MSUB(TEMP,M,TEMP)                                     0000000
21    9      CALL MEQU(X,IK,TEMP,IFLAG1)                                0000000
22    10  CONTINUE                                                      0000000
C                                                                       0000000
C    SET MATRIX Q,R                                                     0000000
C                                                                       0000000
C                                                                       0000000
C          Q(1,1,J)=M**2 , SUM=SUMATION OF X(K)**2                      0000000
C                                                                       0000000
23        DO 400 N=2,ITUPLE                                             0000000
24           CALL MMPY(M,M,TEMP)                                        0000000
25           CALL MTEQU(Q,IFLAG1,IFLAG1,TEMP,IFLAG1)                    0000000
```

```
26              CALL KINPT(0,SUM,0)                                          0000000
27              DO 20 K=1,N                                                  0000000
28                 CALL MEQU(X,K,TEMP,IFLAG2)                                0000000
29                 CALL MMPY(TEMP,TEMP,TEMP1)                                0000000
30                 CALL MADD(SUM,TEMP1,SUM)                                  0000000
31     20       CONTINUE                                                     0000000
32              CALL MTEQU(R,IFLAG1,IFLAG1,SUM,IFLAG1)                       0000000
       C                                                                    0000000
       C     R(1,K)=R(K,1)=M*X(K) , Q(1,K)=Q(K,1)=-M*X(K),Q(K,K)=1+X(K)**2  0000000
       C     R(K,K)=M**2                                                    0000000
       C                                                                    0000000
33              DO 40 K=2,N                                                  0000000
34                 CALL MEQU(X,K,TEMP,IFLAG2)                                0000000
35                 CALL MMPY(M,TEMP,TEMP1)                                   0000000
36                 CALL MTEQU(R,IFLAG1,K,TEMP1,IFLAG1)                       0000000
37                 CALL MTEQU(R,K,IFLAG1,TEMP1,IFLAG1)                       0000000
38                 TEMP1(NDGT)=-TEMP1(NDGT)                                  0000000
39                 CALL MTEQU(Q,IFLAG1,K,TEMP1,IFLAG1)                       0000000
40                 CALL MTEQU(Q,K,IFLAG1,TEMP1,IFLAG1)                       0000000
41                 CALL MMPY(TEMP,TEMP,TEMP)                                 0000000
42                 CALL MADD(ONE,TEMP,TEMP)                                  0000000
43                 CALL MTEQU(Q,K,K,TEMP,IFLAG1)                             0000000
44                 CALL MMPY(M,M,TEMP)                                       0000000
45                 CALL MTEQU(R,K,K,TEMP,IFLAG1)                             0000000
46                 IF(K EQ 2) GO TO 40                                       0000000
47                 KK=K-1                                                    0000000
       C                                                                    0000000
       C     Q(J,K)=Q(K,J)=X(J)*X(K) , R(J,K)=R(K,J)=0                      0000000
       C                                                                    0000000
48                 DO 30 J=2,KK                                              0000000
49                    CALL MEQU(X,J,TEMP1,IFLAG2)                            0000000
50                    CALL MEQU(X,K,TEMP2,IFLAG2)                            0000000
51                    CALL MMPY(TEMP1,TEMP2,TEMP)                            0000000
52                    CALL MTEQU(Q,J,K,TEMP,IFLAG1)                          0000000
53                    CALL MTEQU(Q,K,J,TEMP,IFLAG1)                          0000000
54                    CALL MTEQU(R,J,K,ZERO,IFLAG1)                          0000000
55                    CALL MTEQU(R,K,J,ZERO,IFLAG1)                          0000000
56     30          CONTINUE                                                  0000000
57     40       CONTINUE                                                     0000000
58              K=N                                                          0000000
59              CALL MMPY(M,M,Q3)                                            0000000
       CCCCCC                                                               0000000
       C     STEP2 FIND MINIMUM Q(J,J)                                      0000000
       CCCCCC                                                               0000000
60     50       DO 60 J=1,N                                                  0000000
61                 CALL MTEQU(Q,J,J,TEMP,IFLAG2)                             0000000
62                 CALL MCMP(TEMP,Q3,IR)                                     0000000
63                 IF(IR)55,60,60                                            0000000
64     55          CALL MADD(ZERO,TEMP,Q3)                                   0000000
65     60       CONTINUE                                                     0000000
       CCCCC                                                                0000000
       C     STEP3 READY FOR EXHAUSTIVE SEARCH ?                            0000000
       CCCCC                                                                0000000
66              CALL KINPT(1,PROD,0)                                         0000000
67              DO 70 J=1,N                                                  0000000
68                 CALL MTEQU(R,J,J,TEMP,IFLAG2)                             0000000

69                 CALL MMPY(TEMP,Q3,TEMP1)                                  0000000
70                 CALL MSQRT(TEMP1,TEMP2)                                   0000000
71                 CALL MDIV(TEMP2,M,TEMP,NR)                                0000000
72                 CALL MEQU(C1,J,TEMP,IFLAG1)                               0000000
73                 CALL MMPY(TEMP,TWO,TEMP)                                  0000000
74                 CALL MADD(TEMP,ONE,TEMP)                                  0000000
75                 CALL MMPY(PROD,TEMP,PROD)                                 0000000
76                 CALL MCMP(PROD,THOUND,IR)                                 0000000
77                 IF(IR)70,70,80                                            0000000
78     70       CONTINUE                                                     0000000
79              GO TO 140                                                    0000000
       CCCCCC                                                               0000000
       C     STEP4 TRANSFORM                                                0000000
       CCCCCC                                                               0000000
80     80       CALL MTEQU(Q,K,K,TEMP,IFLAG2)                               0000000
       C                                                                    0000000
       C  FIND NEAREST C(J)=Q(J,K)/Q(K,K)                                   0000000
       C                                                                    0000000
81              DO 120 J=1,N                                                 0000000
82                 IF (J EQ K)GO TO 120                                      0000000
83                 CALL MTEQU(Q,J,K,TEMP1,IFLAG2)                            0000000
84                 CALL MDIV(TEMP1,TEMP,TEMP2,NR)                            0000000
85                 CALL MDIV(TEMP,TWO,TEMP3,NQ)                              0000000
86                 IF(NR(NDGT))90,110,100                                    0000000
87     90          NR(NDGT)=1                                                0000000
88                 CALL MCMP(NR,TEMP3,IR)                                    0000000
89                 IF(IR)110,110,95                                          0000000
90     95          CALL MSUB(TEMP2,ONE,TEMP2)                                0000000
91                 GO TO 110                                                 0000000
92     100         CALL MCMP(NR,TEMP3,IR)                                    0000000
93                 IF(IR)110,110,105                                         0000000
94     105         CALL MADD(TEMP2,ONE,TEMP2)                                0000000
95     110         CALL MEQU(C2,J,TEMP2,IFLAG1)                              0000000
96     120      CONTINUE                                                     0000000
       C                                                                    0000000
       C       R(K,J)=R(K,J)+C1(J)*R(J,K) ,Q(J,K)=Q(J,K)+C1(J)*Q(K,J)       0000000
       C                                                                    0000000
97              DO 130 J=1,N                                                 0000000
98                 IF(J EQ K)GO TO 130                                       0000000
99                 CALL MEQU(C2,J,TEMP,IFLAG2)                               0000000
100                CALL MCMP(TEMP,ZERO,IR)                                   0000000
101                IF(IR)125,130,125                                         0000000
102    125         TEMP(NDGT)=-TEMP(NDGT)                                    0000000
103                CALL TRANS(Q,J,K,TEMP,N)                                  0000000
104                TEMP(NDGT)=-TEMP(NDGT)                                    0000000
105                CALL TRANS(R,K,J,TEMP,N)                                  0000000
106    130       CONTINUE                                                    0000000
       CCCC                                                                 0000000
       C     STEP5 MODIFICATION OF K                                        0000000
       CCCC                                                                 0000000
107             K=K-1                                                        0000000
108             IF(K EQ 0)K=N                                                0000000
110             GO TO 50                                                     0000000
       CCCC                                                                 0000000
       C     STEP6 PREPARATION FOR EXHAUSTIVE SEARCH                        0000000
       CCCC                                                                 0000000
```

```
111     140       K=N                                                    0000000
112               DO 150 J=1,N                                           0000000
113     150          CALL MEQU(EX,J,ZERO,IFLAG1)                         0000000
        CCCC                                                           0000000
        C    STEP7 ADVANCE X(K)                                        0000000
        CCCC                                                           0000000
114     160       CALL MEQU(EX,K,TEMP,IFLAG2)                          0000000
115               CALL MADD(TEMP,ONE,TEMP)                             0000000
116               CALL MEQU(EX,K,TEMP,IFLAG1)                          0000000
117               CALL MEQU(C1,K,TEMP1,IFLAG2)                         0000000
118               CALL MCMP(TEMP,TEMP1,IR)                             0000000
119               IF(IR)170,170,210                                    0000000
        CCCC                                                           0000000
        C    STEP8 ADVANCE K                                           0000000
        CCCC                                                           0000000
120     170       K=K+1                                                0000000
121               IF(K GT N)GO TO 190                                  0000000
122               CALL MEQU(C1,K,TEMP,IFLAG2)                          0000000
123               TEMP(NDGT)=-TEMP(NDGT)                               0000000
124               CALL MEQU(EX,K,TEMP,IFLAG1)                          0000000
125               GO TO 170                                            0000000
        CCCC                                                           0000000
        C    STEP9                                                     0000000
        CCCC                                                           0000000
126     190       CALL KINPT(Q,Q2,Q)                                   0000000
127               DO 200 I=1,N                                         0000000
128                  DO 200 J=1,N                                      0000000
129                     CALL MEQU(EX,I,TEMP,IFLAG2)                    0000000
130                     CALL MEQU(EX,J,TEMP1,IFLAG2)                   0000000
131                     CALL MMPY(TEMP,TEMP1,TEMP)                     0000000
132                     CALL MTEQU(Q,I,J,TEMP1,IFLAG2)                 0000000
133                     CALL MMPY(TEMP,TEMP1,TEMP)                     0000000
134                     CALL MADD(Q2,TEMP,Q2)                          0000000
135     200       CONTINUE                                             0000000
136               CALL MCMP(Q2,Q3,IR)                                  0000000
137               IF(IR)195,210,210                                    0000000
138     195       CALL MADD(Q2,ZERO,Q3)                                0000000
        C                                                              0000000
        C                                                              0000000
139     210     K=K-1                                                  0000000
140             IF(K GT 0)GO TO 160                                    0000000
        CCCC                                                           0000000
        C    STEP 10 CALCULATE V AND C                                 0000000
        C                                                              0000000
141                  LTHQ=Q3(NLTH)                                     0000000
142                  CALL OUTPT(Q3)                                    0000000
143                  DM=0.                                             0000000
144                  DTEMP=Q3(LTHQ)                                    0000000
145                  IL=LTHQ-1                                         0000000
146                  IF(IL LE 0) GO TO 310                             0000000
147                  DO 300 K=1,IL                                     0000000
148                     KK=IL+1-K                                      0000000
149                     DTEMP=DTEMP*NBASE+Q3(KK)                       0000000
150     300          CONTINUE                                          0000000
151     310          DTEMP=DTEMP*Q3(NDGT)                              0000000
152                  LTHQ=M(NLTH)                                      0000000

153                  IL=LTHQ-1                                         0000000
154                  DM=M(LTHQ)                                        0000000
155                  IF(IL LE 0) GO TO 330                             0000000
156                  DO 320 K=1,IL                                     0000000
157                     KK=IL+1-K                                      0000000
158                     DM=DM*NBASE+M(KK)                              0000000
159     320          CONTINUE                                          0000000
160     330          DM=DM*M(NDGT)                                     0000000
161                  V(N)=DSQRT(DTEMP)                                 0000000
162                  GO TO(340,340,350,360,370),N                      0000000
163     340          C(2)=(PI*V(2)*V(2))/DM                            0000000
164                  GO TO 380                                         0000000
165     350          C(N)=(4*PI*V(N)**N)/(3*DM)                        0000000
166                  GO TO 380                                         0000000
167     360          C(N)=(PI*PI*V(N)**N)/(2*DM)                       0000000
168                  GO TO 380                                         0000000
169     370          C(N)=(8*PI*PI*V(N)**N)/(15*DM)                    0000000
170     380          WRITE(LP,390)N,V(N),C(N)                          0000000
171     390          FORMAT(39X,I2,7X,F9 2,5X,F11 9)                   0000000
172     400       CONTINUE                                             0000000
173            RETURN                                                  0000000
174            END                                                     0000000
```

```
      SUBROUTINE TRANS(A,IRSUB,ICSUB,TEMP,N)                          0000000
C * *TRANS* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * C000000
C                                                                     * 0000000
C    PURPOSE      TRANSFORMATION                                      * 0000000
C         A(IRSUB,J)=A(IRSUB,J)+TEMP*A(ICSUB,J)                       * 0000000
C         A(J,IRSUB)=A(J,IRSUB)+TEMP*A(J,IRSUB)                       * 0000000
C                                                                     * 0000000
C    PARAMETERS                                                       * 0000000
C         A       THREE DIMENSIONAL VECTOR                            * 0000000
C         IRSUB   ROW SUBSCRIPTION OF A                               * 0000000
C         ICSUB   COL SUBSCRIPTION OF A                               * 0000000
C         TEMP    ONE DIMENSIONAL ARRAY                               * 0000000
C         N       NUMBER OF LOOP                                      * 0000000
C                                                                     * 0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                     0000000
      INTEGER  A(5,5,50),TEMP(50),TEMP2(50),TEMP3(50)                 0000000
      DATA     IFLAG1,IFLAG2/1,2/                                     0000000
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                          0000000
        DO 10 J=1,N                                                   0000000
           CALL MTEQU(A,ICSUB,J,TEMP2,IFLAG2)                         0000000
           CALL MMPY(TEMP2,TEMP,TEMP2)                                0000000
           CALL MTEQU(A,IRSUB,J,TEMP3,IFLAG2)                         0000000
           CALL MADD(TEMP3,TEMP2,TEMP3)                               0000000
           CALL MTEQU(A,IRSUB,J,TEMP3,IFLAG1)                         0000000
10    CONTINUE                                                        0000000
C                                                                     0000000
        DO 20 J=1,N                                                   0000000
           CALL MTEQU(A,J,ICSUB,TEMP2,IFLAG2)                         0000000
           CALL MMPY(TEMP,TEMP2,TEMP2)                                0000000
           CALL MTEQU(A,J,IRSUB,TEMP3,IFLAG2)                         0000000
           CALL MADD(TEMP3,TEMP2,TEMP3)                               0000000
           CALL MTEQU(A,J,IRSUB,TEMP3,IFLAG1)                         0000000
20    CONTINUE                                                        0000000
      RETURN                                                          0000000
      END                                                             0000000
```

```
      SUBROUTINE MEQU(IX,ISUB,IY,IFLAG)                               0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                     * 0000000
C    PURPOSE     IFLAG=1 THEN SET IX(ISUB,J)=IY(J)                    * 0000000
C                IFLAG=2 THEN SET IY(J)=IX(ISUB,J)                    * 0000000
C                                                                     * 0000000
C    PARAMETERS                                                       * 0000000
C       IX    TWO DIMENSIONAL ARRAY                                   * 0000000
C       ISUB   ROW SUBSCRIPTION OF IX                                 * 0000000
C       IY     ONE DIMENSIONAL ARRAY                                  * 0000000
C       IFLAG   FLAG INDICATES THE DIRECTION OF EQUAL                 * 0000000
C                                                                     * 0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                     0000000
      DIMENSION IX(5,50),IY(50)                                       0000000
      COMMON /IMULT/NDGT,NBASE,NLTH,LP,IN                             0000000
      IF(IFLAG EQ 2)GO TO 20                                          0000000
      DO 10 J=1,NDGT                                                  0000000
10       IX(ISUB,J)=IY(J)                                             0000000
      RETURN                                                          0000000
20    DO 30 J=1,NDGT                                                  0000000
30       IY(J)=IX(ISUB,J)                                             0000000
      RETURN                                                          0000000
      END                                                             0000000
```

```
      SUBROUTINE MTEQU(IX,IROW,ICOL,IY,IFLAG)                          0000000
C * *MTEQU* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                      0000000
C    PURPOSE       IFLAG=1 THEN SET IX(IROW,ICOL,J)=IY(J)              0000000
C                  IFLAG=2 THEN SET IY(J)=IX(IROW,ICOL,J)              0000000
C                                                                      0000000
C    PARAMETERS                                                        0000000
C                  IY       THREE DIMENSIONAL VECTOR                 0000000
C                  IROW     ROW SUBSCRIPTION OF IX                   0000000
C                  ICOL     COL SUBSCRIPTION OF IX                   0000000
C                  IY       ONE DIMENSIONAL ARRAY                    0000000
C                  IFLAG    FLAG INDICATES THE DIRECTION OF          0000000
C                           EQUAL                                    0000000
C                                                                      0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                      0000000
      DIMENSION IX(5,5,50),IY(50)                                      0000000
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                           0000000
C                                                                      0000000
      IF(IFLAG EQ 2) GO TO 20                                          0000000
      DO 10 J=1,NDGT                                                   0000000
   10    IX(IROW,ICOL,J)=IY(J)                                         0000000
      RETURN                                                           0000000
   20 DO 30 J=1,NDGT                                                   0000000
   30    IY(J)=IX(IROW,ICOL,J)                                         0000000
      RETURN                                                           0000000
      END                                                              0000000
```

```
C                                                                       0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
C                                                                 *     0000000
C       INTEGER MULTIPLE PRECISION ARITHMETIC                     *     0000000
C                OPERATION PROJECT                                *     0000000
C                                                                 *     0000000
C                                                                 *     0000000
C                                                                 *     0000000
C     PURPOSE       IMPLEMENT MULTIPLE PRECISION                  *     0000000
C                   INTEGER ARITHMETIC OPERATION                  *     0000000
C                   FOR IBM OS/360-370                            *     0000000
C     VARIABLES                                                   *     0000000
C         NDGT      # OF PRECISIONS OF EACH INTEGER               *     0000000
C         NBASE     BASE OF EACH PRECISION                        *     0000000
C         NLTH      POSITION TO STORE THE ACTUAL                  *     0000000
C                    LENGTH OF INTEGER                            *     0000000
C                                                                 *     0000000
C                                                                 *     0000000
C     SUBPROGRAM                                                  *     0000000
C         PREPR     GIVE THE VALUE TO COMMON VARIABLE             *     0000000
C         INPDT     READ IN A FORTRAN INTEGER AND                 *     0000000
C                   CONVERT TO MULTIPLE PRECISION                 *     0000000
C                   INTEGER                                       *     0000000
C         KINPT     CONVERT FORTRAN INTEGER TO MULTIPLE           *     0000000
C                   PRECISIONS BASE NBASE                         *     0000000
C         OUTPT     PRINT MULTIPLE PRECISION INTEGER              *     0000000
C                   AS SIGN AND VALUE TWO PARTS                   *     0000000
C         MADD      MULTIPLE PRECISION INTEGER                    *     0000000
C                   ADDITION                                      *     0000000
C         MSUB      MULTIPLE PRECISION INTEGER                    *     0000000
C                   SUBTRACTION                                   *     0000000
C         MMPY      MULTIPLE PRECISION INTEGER                    *     0000000
C                   MULTIPLICATION                                *     0000000
C         MDIV      MULTIPLE PRECISION INTEGER                    *     0000000
C                   DIVISION                                      *     0000000
C         MEQL      MULTIPLE PRECISION INTEGER                    *     0000000
C                   EQUAL OPERATION                               *     0000000
C         MSQRT     MULTIPLE PRECISION INTEGER                    *     0000000
C                   SQRT ROOT OPERATION                           *     0000000
C         MCMP      MULTIPLE PRECISION INTEGER                    *     0000000
C                   COMPARE OPERATION                             *     0000000
C                                                                 *     0000000
C                                                                 *     0000000
C     ADVISOR       DR  J  P  CHANDLER                            *     0000000
C     AUTHOR        CHU, CHUNG-PEI                                *     0000000
C     DATE        : SUMMER 1986                                   *     0000000
C     LOCATION    : OKLAHOMA STATE UNIVERSITY                     *     0000000
C                                                                 *     0000000
C                                                                 *     0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
C                                                                       0000000

1         SUBROUTINE PREPR(NFLG)                                        0000000

      C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
      C                                                           *     0000000
      C     PURPOSE       GIVE INITIAL VALUE OF COMMON          *     0000000
      C                   VARIABLES                             *     0000000
      C                                                         *     0000000
      C                                                         *     0000000
      C     PARAMETERS                                          *     0000000
      C        NFLG       IF NFLG=0 NBASE HAS LEGEAL VALUE      *     0000000
      C                   IF NFLG=1 NBASE HAS ILLEGEAL VALUE    *     0000000
      C                                                         *     0000000
      C * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
      C                                                                 0000000
2         COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                        0000000
      C                                                                 0000000
      C                                                                 0000000
3         NLTH=NDGT-1                                                   0000000
4         NFLG=0                                                        0000000
5         IF (NBASE GT 0)GO TO 30                                       0000000
6         NFLG=1                                                        0000000
7         WRITE(LP,20)                                                  0000000
8     20  FORMAT(//20X,'****  NBASE LESS OR EQUAL TO 0 ****')           0000000
9         GO TO 100                                                     0000000
10    30  NSIZE=(NBASE-1)*(NBASE-1)+2*(NBASE-1)                         0000000
11        NSIZE=NSIZE-2*(NBASE-1)                                       0000000
12        NOPBS=SQRT(FLOAT(NSIZE))+1                                    0000000
13        IF(NOPBS EQ NBASE) GO TO 100                                  0000000
14        NFLG=1                                                        0000000
15        WRITE(LP,40)                                                  0000000
16    40  FORMAT(//20X,'**** NBASE GREATER THAN SQUARE ROOT OF WORD',   0000000
          1' SIZE MINUS 1 ****')                                        0000000
17    100 RETURN                                                        0000000
18        END                                                           0000000
```

```
      SUBROUTINE INPDT(JX)                                              0000000
C * INPDT * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
C                                                                       *
C   PURPOSE                                                             *
C           READ IN FORTRAN NUMBER AND CONVERT IT INTO                  *
C           MULTIPLE PRECISION INTEGFR                                  *
C           INPUT DATA READ IN AS CHARACTER                             *
C           ONE FORTRAN NUMBER IN ONE INPUT LINE                        *
C           USE FREE FORMAT                                             *
C                                                                       *
C   VARIABLE                                                            *
C           JX IS AN MULTIPLE PRECISION INTEGER                         *
C                                                                       *
C                                                                       *
C   SUBROUTINE CALLED                                                   *
C           ICOD    CONVERT CHARACTER INPUT DATA TO                     *
C                   FORTRAN NUMBER                                      *
C           KINPT   CONVERT FORTRAN NUMBER TO MULTIPLE                  *
C                   PRECISION INTEGER                                   *
C                                                                       *
C                                                                       *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
C                                                                       0000000
      CHARACTER*1 CARD(80),DIGIT(10),BLANK,PLUS,MINUS                   0000000
      INTEGER BIGN,ENP,SIGN,COL                                         0000000
      DIMENSION JX(1),JINBR(50)                                         0000000
      DATA DIGIT/'0','1','2','3','4','5','6','7','8','9'/              0000000
     1,PLUS/'+'/,MINUS/'-'/,BLANK/' '/,COL/80/                          0000000
      COMMON /IMULT/NDGT,NBASE,NLTH,LP,IN                               0000000
CCC                                                                     0000000
C     INPUT 80 CHARACTERS                                               0000000
CCC                                                                     0000000
      READ(IN,1)(CARD(J),J=1,COL)                                       0000000
    1 FORMAT(80A1)                                                      0000000
      SIGN=1                                                            0000000
C                                                                       0000000
C   CHECK ILLEGEAL CHARACTER                                            0000000
C                                                                       0000000
      DO 10 J=1,COL                                                     0000000
         IF(CARD(J) EQ MINUS) GO TO 30                                  0000000
         IF(CARD(J) EQ PLUS) GO TO 35                                   0000000
         IF(CARD(J).NE BLANK) GO TO 50                                  0000000
   10 CONTINUE                                                          0000000
   15 WRITE(LP,20)                                                      0000000
   20 FORMAT(//,5X,'***  INPUT ERROR OCCUR ***')                        0000000
      RETURN                                                            0000000
C                                                                       0000000
C                                                                       0000000
C                                                                       0000000
   30 SIGN=-1                                                           0000000
   35 BIGN=J+1                                                          0000000
   40 J=J+1                                                             0000000

      IF (J GT COL)GO TO 80                                             0000000
   50 ERCOD=0                                                           0000000
      DO 60 JJ=1,10                                                     0000000
   60    IF(CARD(J) EQ DIGIT(JJ))ERCOD=-1                               0000000
      IF(ERCOD EQ 0)GO TO 75                                            0000000
      IF(J GT 1)GO TO 70                                                0000000
      BIGN=1                                                            0000000
      GO TO 40                                                          0000000
   70 K=J-1                                                             0000000
      IF(CARD(K) EQ BLANK)BIGN=J                                        0000000
      GO TO 40                                                          0000000
   75 IF(CARD(J) NE BLANK) GO TO 15                                     0000000
C                                                                       0000000
C                                                                       0000000
   80 N=J-BIGN                                                          0000000
      IF(N LE.0)GO TO 15                                                0000000
      ENP=J-1                                                           0000000
      K=N/4                                                             0000000
      KK=MOD(N,4)                                                       0000000
      IF(KK NE 0)K=K+1                                                  0000000
C                                                                       0000000
C     FETCH DATA                                                        0000000
C                                                                       0000000
      DO 90 JJ=1,K                                                      0000000
         J=K+1-JJ                                                       0000000
         CALL ICOD(CARD,BIGN,ENP,JINBR(J))                              0000000
   90 CONTINUE                                                          0000000
      JINBR(1)=JINBR(1)*SIGN                                            0000000
CCC                                                                     0000000
C     CONVERT DATA TO MULTIPLE PRECISION                                0000000
CCC                                                                     0000000
C                                                                       0000000
      DO 95 J=1,50                                                      0000000
   95    JX(J)=0                                                        0000000
C                                                                       0000000
C                                                                       0000000
      KINDK=K                                                           0000000
      DO 100 J=1,K                                                      0000000
          KJ=K+1-J                                                      0000000
          CALL KINPT(JINBR(KJ),JX,KINDK)                                0000000
          KINDK=KINDK-1                                                 0000000
  100 CONTINUE                                                          0000000
C                                                                       0000000
C                                                                       0000000
      RETURN                                                            0000000
      END                                                               0000000
```

```
      SUBROUTINE ICOD(CARD,STP,ENP,JINBR)                              0000000
C * ICOD * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  0000000
C                                                                      *0000000
C                                                                      *0000000
C    CONVERT CHARACTER VARIABLE TO FORTRAN NUMBER                      *0000000
C                                                                      *0000000
C    CARD   CHARACTER VARIABLE CONTINE 80 CHARACTERS                   *0000000
C     STP   STARTING POSITION OF CARD TO BE CONVERTED                  *0000000
C     ENP   ENDING POSITION OF CARD TO BE CONVERTED                    *0000000
C    JINBR  FORTRAN NUMBER                                             *0000000
C                                                                      *0000000
C                                                                      *0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  0000000
C                                                                      0000000
      CHARACTER*1 CARD(80),DIGIT(10)                                   0000000
      INTEGER STP ENP,JINBR                                            0000000
      DATA DIGIT/'0','1','2','3' '4','5','6','7','8','9'/             0000000
      COMMON /IMULT/NDGT,NBASE,NLTH,LP,IN                              0000000
C                                                                      0000000
C                                                                      0000000
      KENP=ENP-3                                                       0000000
      JINBR=0                                                          0000000
      IF (KENP LE STP)KENP=STP                                         0000000
      DO 20 J=KENP,ENP                                                 0000000
         DO 10 K=1,10                                                  0000000
   10       IF(CARD(J) EQ DIGIT(K))JINBR=JINBR*10+K-1                  0000000
   20 CONTINUE                                                         0000000
      ENP=ENP-4                                                        0000000
      RETURN                                                           0000000
      END                                                              0000000
```

```
      SUBROUTINE  KINPT(INPTX,JX,KINDK)                                0000000
C * KINPT * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  0000000
C                                                                      *0000000
C    CONVERT INTEGER TO MULTIPLE PRECISION                             *0000000
C    BASE NBASE                                                        *0000000
C                                                                      *0000000
C    PARAMETERS .                                                      *0000000
C        INPTX    INPUT FORTRAN INTEGER                               *0000000
C           JX    OUTPUT NBASE MULTIPLE PRECISIONS                    *0000000
C                 INTEGER  JX(1) CONTAINS THE LOWER                   *0000000
C                 ORDER DIGITS OF INPTX                               *0000000
C                 JX(NDGT) CONTAINS THE SIGN OF                       *0000000
C                 INPTX   IF INPTX>=0 THEN                            *0000000
C                 JX(NDGT)=1, OTHERWISE JX(NDGT)=-1                   *0000000
C                 JX(NDGT-1) CONTAINS THE TURE                        *0000000
C                 LENGTH OF JX                                        *0000000
C        KINDK    KINDK = 0  NEW NUMBER                               *0000000
C                 KINDK > 0  CONTINUEING NUMBER                       *0000000
C                                                                      *0000000
C                                                                      *0000000
C                                                                      *0000000
C    VARIABLES                                                         *0000000
C        JW       ABSOLUTE VALUE OF INPTX                             *0000000
C        LL       IF LL < = 0 FINISH CONVERTING                       *0000000
C                 OTHERWISE STAY IN CONVERT LOOP                      *0000000
C                                                                      *0000000
C                                                                      *0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  0000000
C                                                                      0000000
      DIMENSION JX(1)                                                  0000000
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                           0000000
C                                                                      0000000
      IF(KINDK)5,5,8                                                   0000000
    5 JX(NLTH)=0                                                       0000000
    8 JW=IABS(INPTX)                                                   0000000
      K=NDGT-2                                                         0000000
      L=JX(NLTH)+1                                                     0000000
C                                                                      0000000
C   DECIDE SIGN FOR MULTIPLE PRECISION INTEGER                         0000000
C                                                                      0000000
      IF(INPTX)10,20,20                                                0000000
   10 JX(NDGT)=-1                                                      0000000
      GO TO 22                                                         0000000
   20 JX(NDGT)=1                                                       0000000
   22 IF(KINDK LE 1) GO TO 40                                          0000000
C                                                                      0000000
C   IF JW LESS THAN NBASE                                              0000000
C                                                                      0000000
C                                                                      0000000
      IF(JW-NBASE)25,40,40                                             0000000
   25 JX(L)=JW                                                         0000000
      IF(KINDK)35,35,30                                                0000000

   30 L=L+1                                                            0000000
      JX(NLTH)=JX(NLTH)+1                                              0000000
      JX(L)=0                                                          0000000
   35 JX(NLTH)=JX(NLTH)+1                                              0000000
      GO TO 70                                                         0000000
C   CONVERT TO BASE  NBASE MULTIPLE PRECISION INTEGER                  0000000
C                                                                      0000000
   40 DO 60 J=L,K                                                      0000000
         JX(J)=MOD(JW,NBASE)                                           0000000
         JW=JW/NBASE                                                   0000000
         JX(NLTH)=JX(NLTH)+1                                           0000000
         IF(JW)60,70,60                                                0000000
   60 CONTINUE                                                         0000000
C                                                                      0000000
   70 L=JX(NLTH)+1                                                     0000000
      DO 80 J=L,K                                                      0000000
   80    JX(J)=0                                                       0000000
      RETURN                                                           0000000
      END                                                              0000000
```

```
      SUBROUTINE  MMPY(KA,KB,KC)                                        0000000
C * * * * * * * * * * * * * * * * * * * * * * * *                       0000000
C                                                   *                   0000000
C     KC = KA * KB                                  *                   0000000
C                                                   *                   0000000
C * * * * * * * * * * * * * * * * * * * * * * * *                       0000000
C                                                                       0000000
      DIMENSION KA(1),KB(1),KC(1),KD(50)                                0000000
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                            0000000
C                                                                       0000000
         DO 10 J=1,NDGT                                                 0000000
10       KD(J)=0                                                        0000000
         LTHA=KA(NLTH)                                                  0000000
         LTHB=KB(NLTH)                                                  0000000
C                                                                       0000000
C    CHECK OVERFLOW                                                     0000000
C                                                                       0000000
         KFLOW=NLTH-LTHA-LTHB                                           0000000
         IF(KFLOW)20,20,30                                              0000000
20       WRITE(LP,25)                                                   0000000
25       FORMAT(//5X,'*****HIGH ORDER DIGITS OF RESULT PASS TO',        0000000
     1      'RECEVING FIELD MAY BE TRUNCATED*****')                     0000000
C                                                                       0000000
C    CHECK IF ANY OPERAND IS ZERO                                       0000000
C                                                                       0000000
30       IF(LTHA-1+KA(1))40,50,40                                       0000000
40       IF(LTHB-1+KB(1))60,50,60                                       0000000
50       KD(NLTH)=1                                                     0000000
         KD(NDGT)=1                                                     0000000
         KD(1)=0                                                        0000000
         GO TO 110                                                      0000000
C                                                                       0000000
C     PERFORM MULTIPLICATION                                            0000000
C                                                                       0000000
60       DO 80 J=1,LTHA                                                 0000000
            KK=0                                                        0000000
            DO 70 K=1,LTHB                                              0000000
               L=J+K-1                                                  0000000
               KD(L)=KA(J)*KB(K)+KD(L)+KK                               0000000
               KK=KD(L)/NBASE                                           0000000
70             KD(L)=MOD(KD(L),NBASE)                                   0000000
80          KD(L+1)=KD(L+1)+KK                                          0000000
C                                                                       0000000
C     SET LENGTH FOR RESULT                                             0000000
C                                                                       0000000
            L=LTHA+LTHB                                                 0000000
            KD(NLTH)=L                                                  0000000
            IF(KD(L))100,90,100                                         0000000
90          KD(NLTH)=L-1                                                0000000
C                                                                       0000000
C     SET SIGN FOR RESULT                                               0000000
C                                                                       0000000

100      KD(NDGT)=KA(NDGT)*KB(NDGT)                                     0000000
110      DO 150 J=1,NDGT                                                0000000
150         KC(J)=KD(J)                                                 0000000
         RETURN                                                         0000000
         END                                                            0000000
```

```
      SUBROUTINE  MADD(KA,KB,KC)                                        0000000
C * MADD  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
C                                                                       *
C                                                                       *
C    PURPOSE    · CALCULATE KC=KA+KB                                    *
C    VARIABLES                                                          *
C        NSAME    INDICATE THE SIGN OF KA,KB                           *
C                 NSAME=1 KA,KB HAS SAME SIGN                          *
C                 NSAME=0 KA,KB HAS OPPOSITE SIGN                      *
C        NSIGN · INDICATES THE SIGN OF RESULT                          *
C                 NSIGN=0 RESULT HAS SAME SIGN AS KA                   *
C                 NSIGN=1 RESULT HAS SAME SIGN AS KB                   *
C        MXLTH · LONGER LENGTH OF KA,KB                                 *
C                                                                       *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      DIMENSION   KA(1),KB(1),KC(1),KD(50)
      COMMON     /IMULT/NDGT,NBASE,NLTH,LP,IN
C
C  INITIALIZE
C
        DO 10 J=1,NDGT
10      KD(J)=0
        NSAME=1
        LTHA=KA(NLTH)
        LTHB=KB(NLTH)
        MXLTH=MAXO(KA(NLTH),KB(NLTH))
        N=MXLTH+1
C
C     DECIDE WHETHER KA,KB HAS SAME SIGN
C
        IF(KA(NDGT) * KB(NDGT))100,50,50
C
C     KA,KB HAS SAME SIGN
C
50      DO 60 J=1,MXLTH
60      KD(J)=KA(J)+KB(J)
        NSIGN=0
        GO TO 200
C
C      KA,KB HAS OPPOSITE SIGN
C
100     NSAME=0
        IF(KA(NLTH)-KB(NLTH))130,110,150
C
C   KA,KB HAS SAME LENGTH COMPARE THEM
C
110     DO 120 J=1,MXLTH
          K=MXLTH+1-J
          IF(KB(K)-KA(K))150,115,130
115     KD(K)=0
120     CONTINUE

        KD(NDGT)=1
        KD(NLTH)=1
        GO TO 270
C
C   KB GREATER THAN KA
C
130     NSIGN=1
        DO 140 J=1,MXLTH
140     KD(J)=NBASE-1-KA(J)+KB(J)
        GO TO 200
C
C   KA GREATER THAN KB
C
150     NSIGN=0
        DO 160 J=1,MXLTH
160     KD(J)=NBASE-1-KB(J)+KA(J)
        GO TO 200
C
C     NORMALIZE
C
200     DO 210 J=1,MXLTH
          KD(J+1)=KD(J+1)+KD(J)/NBASE
210     KD(J)=MOD(KD(J),NBASE)
        IF(NSAME)240,220,240
C
C    KA,KB HAS OPPOSITE SIGN ,ELIMINATE LEADING 1
C
220     K=MXLTH+1
        KD(K)=KD(K)-1
        KD(1)=KD(1)+1
C
C    IF KD(1) OVERFLOW RENORMALIZE KC
C
        IF(KD(1)-NBASE)240,230,230
230     DO 235 J=1,N
          KD(J+1)=KD(J+1)+KD(J)/NBASE
235     KD(J)=MOD(KD(J),NBASE)
C
C    DECIDE THE LENGTH FOR RESULT
C
240     DO 245 J=1,N
          KK=N+1-J
          IF(KD(KK))250,245,250
245     CONTINUE
250     KD(NLTH)=KK
C
C     DECIDE THE SIGN OF RESULT
C
        IF(KK.LT NLTH) GO TO 258
        WRITE(LP,255)
255     FORMAT(//3X,'OVERFLOW OCCUR ON ADDITION')
258     IF(NSIGN)259,259,260
259     KD(NDGT)=KA(NDGT)
        GO TO 270
260     KD(NDGT)=KB(NDGT)
C
```

```
      C    PRINT DATA                                                    0000000
      C                                                                  0000000
57      270   DO 280  J=1,NDGT                                           0000000
58      280      KC(J)=KD(J)                                             0000000
      C                                                                  0000000
59          RETURN                                                       0000000
60          END                                                          0000000
```

```
 1          SUBROUTINE MSUB(KA,KB,KC)                                   0000000
      C * MSUB  * * * * * * * * * * * * * * * * * * * * *               0000000
      C                                                   *              0000000
      C                                                   *              0000000
      C      CALCULATE KC=KA-KB                           *              0000000
      C                                                   *              0000000
      C                                                   *              0000000
      C * * * * * * * * * * * * * * * * * * * * * * * * * *              0000000
      C                                                                  0000000
 2          DIMENSION KA(1),KB(1),KC(1),KD(50)                          0000000
 3          COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                      0000000
      C                                                                  0000000
      C     PRINT DATA                                                   0000000
      C                                                                  0000000
 4          KD(NDGT)=-KB(NDGT)                                          0000000
 5          DO 10 J=1,NLTH                                              0000000
 6       10    KD(J)=KB(J)                                              0000000
 7          CALL MADD(KA,KD,KC)                                         0000000
      C                                                                  0000000
 8          RETURN                                                      0000000
 9          END                                                         0000000
```

```
 1          SUBROUTINE OUTPT(JA)                                        0000000
      C *OUTPT* * * * * * * * * * * * * * * * * *                       0000000
      C                                           *                    0000000
      C  PURPOSE:                                 *                    0000000
      C   PRINT JA AS SIGN, LENGTH AND VALUE      *                    0000000
      C   LENGTH MEANS # OF MULTIPLE PRECISION    *                    0000000
      C   WORDS BE OCCUPIED BY JA.                *                    0000000
      C  LIMITATION                               *                    0000000
      C   MAXIMUM NUMBER OF DIGITS OF AN OUTPUT   *                    0000000
      C   INTEGER IS 96 DIGITS                    *                    0000000
      C                                           *                    0000000
      C                                           *                    0000000
      C * * * * * * * * * * * * * * * * * * * * * *                     0000000
      C                                                                  0000000
 2          CHARACTER*1 NOUT(100),DIGIT(10),KOUT(100)                   0000000
 3          DIMENSION  JA(1),JB(50)                                     0000000
 4          DATA DIGIT/'0','1','2','3','4','5','6','7','8','9'/        0000000
 5          COMMON     /IMULT/NDGT,NBASE,NLTH,LP,IN                     0000000
      C                                                                  0000000
      C   INITIALIZATION                                                0000000
      C                                                                  0000000
 6           DO 3  J=1,NDGT                                             0000000
 7       3      JB(J)=JA(J)                                             0000000
 8           DO 4 J=1,100                                               0000000
 9       4      NOUT(J)=DIGIT(1)                                        0000000
      C                                                                  0000000
      C  CALCULATED # OF DIGIT PER WORD                                  0000000
      C                                                                  0000000
10          NSUBT=NBASE                                                 0000000
11          NDGTW=0                                                     0000000
12       5  NSUBT=NSUBT/10                                             0000000
13          NDGTW=NDGTW+1                                               0000000
14          IF (NSUBT GE 10) GO TO 5                                    0000000
15          JINDX=0                                                     0000000
16          JWEND=JB(NLTH)                                              0000000
      C                                                                  0000000
      C  TRANSFER MPI TO CHARACTERS                                      0000000
      C                                                                  0000000
17          DO 30 KA=1,JWEND                                            0000000
18           DO 30 KB=1,NDGTW                                           0000000
19             JINDX=JINDX+1                                            0000000
20             IRMD=MOD(JB(KA),10)                                      0000000
21             JB(KA)=JB(KA)/10                                         0000000
22             IF(IRMD GT 0)GO TO 15                                    0000000
23             IF(KA NE JWEND) GO TO 10                                 0000000
24             IF(JB(KA) NE 0) GO TO 10                                 0000000
25             IF(JINDX GT 1)JINDX=JINDX-1                              0000000
27             IF(JINDX EQ 1)NOUT(1)=DIGIT(1)                           0000000
29             GO TO 35                                                 0000000
30       10    NOUT(JINDX)=DIGIT(1)                                     0000000
31             GO TO 30                                                 0000000
32       15    DO 20 KC=1,10                                            0000000
33                IF(IRMD EQ KC)NOUT(JINDX)=DIGIT(KC+1)                 0000000
35       20    CONTINUE                                                 0000000
36       30   CONTINUE                                                  0000000
      C                                                                  0000000
      C     PREPARE TO PRINT OUTPUT                                      0000000
      C                                                                  0000000
37       35  DO 40 J=1,JINDX                                           0000000
38             K=JINDX+1-J                                              0000000
39             KOUT(J)=NOUT(K)                                          0000000
40       40  CONTINUE                                                  0000000
41          WRITE(LP,50)JB(NDGT),JB(NLTH),(KOUT(J),J=1,JINDX)          0000000
42       50  FORMAT(/3X,I2,4X,I2,5X,96A1)                              0000000
43          RETURN                                                     0000000
44          END                                                        0000000
```

```
      SUBROUTINE MDIV(NA,NB,NQ,NR)                                      0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
C                                                                       *
C     NA=NB*NQ + NR                                                     *
C                                                                       *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
      DIMENSION  NA(1),NB(1),NQ(1),NR(1),NAA(50),NBB(50),               0000000
     1           NQQ(50),NRR(50),NBF(50),ND(50)                         0000000
      COMMON     /IMULT/NDGT,NBASE,NLTH,LP,IN                           0000000
         IF(NB(NLTH)-1+NB(1))30,30,40                                   0000000
   30    WRITE(LP,35)                                                   0000000
   35    FORMAT(//20X '*****DIVISOR IS ZERO*****')                      0000000
         GO TO 450                                                      0000000
   40    DO 50 J=1,NDGT                                                 0000000
            NAA(J)=NA(J)                                                0000000
            NBB(J)=NB(J)                                                0000000
            NQQ(J)=0                                                    0000000
            NRR(J)=0                                                    0000000
            NBF(J)=0                                                    0000000
   50       ND(J)=0                                                     0000000
C                                                                       0000000
C   COMPARE NA AND NB                                                   0000000
C                                                                       0000000
            JBSAV=NBB(NDGT)                                             0000000
            JASAV=NAA(NDGT)                                             0000000
            NAA(NDGT)=1                                                 0000000
            NBB(NDGT)=1                                                 0000000
            IFF=IMIF(NAA,NBB)                                           0000000
            IF(IFF-1)60,90,70                                           0000000
C                                                                       0000000
C   NA EQ NB RETURN QUOTIENT/REMAINDER OF I/O                           0000000
C                                                                       0000000
   60       LVAL=JBSAV*JASAV                                            0000000
            CALL KINPT(LVAL,NQ,0)                                       0000000
            CALL KINPT(0,NR,0)                                          0000000
            GO TO 450                                                   0000000
C                                                                       0000000
C   NA.LT NB  RETURN NQ=0 AND NR=NA                                     0000000
C                                                                       0000000
   70       CALL KINPT(0,NQ,0)                                          0000000
            NAA(NDGT)=JASAV                                             0000000
            DO 80 J=1,NDGT                                              0000000
   80          NR(J)=NAA(J)                                             0000000
            GO TO 450                                                   0000000
C                                                                       0000000
C   CHECK FOR DIVISION BY AN INTEGER  LT  NBASE CALL MSDV IF TRUE       0000000
C                                                                       0000000
   90       IF(NB(NLTH)-1)100,100,110                                   0000000
  100       KVAL=NB(NDGT)*NB(1)                                         0000000
            CALL MSDV(NA,KVAL,NQ,NREM)                                  0000000
            CALL KINPT(NREM,NR,0)                                       0000000
            GO TO 450                                                   0000000

  110       NPB=NB(NLTH)                                                0000000
            NPA=NA(NLTH)+1                                              0000000
            MLOOP=NPA-NPB                                               0000000
            LXTNT=NPB+1                                                 0000000
            NORM=NBASE/(NBB(NPB)+1)                                     0000000
            IF(NORM-1)130,130,120                                       0000000
C                                                                       0000000
C   NORMALIZE                                                           0000000
C                                                                       0000000
  120       CALL KINPT(NORM,NBF,0)                                      0000000
            CALL MMPY(NAA,NBF,NAA)                                      0000000
            CALL MMPY(NBB,NBF,NBB)                                      0000000
C                                                                       0000000
C   SET TRIAL DIVISOR                                                   0000000
C                                                                       0000000
  130       KPART=NAA(NPA)*NBASE+NAA(NPA-1)                             0000000
            KADBK=NBB(NPB)*NBASE                                        0000000
            IF(NAA(NPA)-NBB(NPB))150,140,150                            0000000
  140       NTRAL=NBASE-1                                               0000000
            GO TO 160                                                   0000000
  150       NTRAL=KPART/NBB(NPB)                                        0000000
            IF(NTRAL)160,330,160                                        0000000
  160       KTS1=NBB(NPB-1)*NTRAL                                       0000000
            KTS2=(KPART-NTRAL*NBB(NPB))*NBASE+NAA(NPA-2)                0000000
            KTEST=KTS1-KTS2                                             0000000
C                                                                       0000000
C   TEST TRIAL DIVISOR SELECTION                                        0000000
C                                                                       0000000
  170       IF(KTEST)190,190,180                                        0000000
  180       NTRAL=NTRAL-1                                               0000000
            GO TO 160                                                   0000000
  190       JAX=LXTNT                                                   0000000
C                                                                       0000000
C   INSERT APPROPRIATE PART OF DIVIDEND INTO ND                         0000000
C                                                                       0000000
            DO 200 J=1,LXTNT                                            0000000
               L=NPA+1-J                                                0000000
               K=LXTNT+1-J                                              0000000
  200          ND(K)=NAA(L)                                             0000000
  210       IF(ND(JAX))240,220,240                                      0000000
  220       JAX=JAX-1                                                   0000000
            IF(JAX)210,230,210                                          0000000
  230       JAX=1                                                       0000000
C                                                                       0000000
C   SET LENGTH OF INSERTED WORD                                         0000000
C                                                                       0000000
  240       ND(NLTH)=JAX                                                0000000
            ND(NDGT)=1                                                  0000000
C                                                                       0000000
C   TAKE TRIAL DIVISOR TIMES DIVISOR AND SUBTRACT FROM ND               0000000
C                                                                       0000000
            KK=0                                                        0000000
            JASET=NBB(NLTH)                                             0000000
            DO 250 J=1,JASET                                            0000000
               NSUM=NTRAL*NBB(J)+KK                                     0000000
               KK=NSUM/NBASE                                            0000000
```

```
 73     250          NBF(J)=MOD(NSUM,NBASE)                                    0000000
 74                  NBF(JASET+1)=KK                                           0000000
 75                  IF(KK)260,260,270                                         0000000
 76     260          NBF(NLTH)=JASET                                           0000000
 77                  GO TO 280                                                 0000000
 78     270          NBF(NLTH)=JASET+1                                         0000000
 79     280          NBF(NDGT)=1                                               0000000
 80                  CALL MSUB(ND,NBF,ND)                                      0000000
 81                  IF(ND(NDGT))290,310,310                                   0000000
      C                                                                        0000000
      C   ADD BACK NECESSARY                                                  0000000
      C                                                                        0000000
 82     290          DO 300 J=2,NPB                                            0000000
 83     300             ND(J)=(NBASE-1)-ND(J)                                  0000000
 84                  ND(1)=NBASE-ND(1)                                         0000000
 85                  ND(NLTH)=LXTNT                                            0000000
 86                  ND(NDGT)=1                                                0000000
 87                  NTRAL=NTRAL-1                                             0000000
 88                  CALL MADD(NBB,ND,ND)                                      0000000
 89                  ND(LXTNT)=0                                               0000000
      C                                                                        0000000
      C   RESET DIVIDEND                                                      0000000
      C                                                                        0000000
 90     310          DO 320 J=1,LXTNT                                          0000000
 91                     L=NPA+1-J                                              0000000
 92                     K=LXTNT+1-J                                            0000000
 93     320             NAA(L)=ND(K)                                           0000000
 94     330          NQQ(MLOOP)=NTRAL                                          0000000
 95                  MLOOP=MLOOP-1                                             0000000
 96                  NPA=NPA-1                                                 0000000
      C                                                                        0000000
      C   UPWARD FLOW POSSIBLE                                                0000000
      C                                                                        0000000
 97                  IF (MLOOP)340,340,130                                     0000000
      C                                                                        0000000
      C   CALCULATE THE REMAINDER                                             0000000
      C                                                                        0000000
 98     340          IF(NAA(NPB))350,350,370                                   0000000
 99     350          NPB=NPB-1                                                 0000000
100                  IF(NPB)360,360,340                                        0000000
101     360          NAA(NLTH)=1                                               0000000
102                  GO TO 380                                                 0000000
103     370          NAA(NLTH)=NPB                                             0000000
104     380          CALL MSDV(NAA,NORM,NRR,NRREM)                             0000000
105                  IF(NRREM)390,400,390                                      0000000
106     390          WRITE(LP,385)                                             0000000
107     385          FORMAT(//20X,'****A PROGRAMMING ERROR OCCURRED****')      0000000
108                  GO TO 450                                                 0000000
      C                                                                        0000000
      C   SET QUOTIENT LENGTH                                                 0000000
      C                                                                        0000000
109     400          JTEST=NA(NLTH)-NB(NLTH)+1                                 0000000
110                  IF(NQQ(JTEST))420,410,420                                 0000000
111     410          NQQ(NLTH)=JTEST-1                                         0000000
112                  GO TO 430                                                 0000000
113     420          NQQ(NLTH)=JTEST                                           0000000

114     430          DO 435 J=1,NDGT                                           0000000
115     435             NBB(J)=NB(J)                                           0000000
116                  DO 440 J=1,NLTH                                           0000000
117                     NQ(J)=NQQ(J)                                           0000000
118     440             NR(J)=NRR(J)                                           0000000
      C                                                                        0000000
      C   SET SIGN OF QUOTIENT AND REMAINDER                                  0000000
      C                                                                        0000000
119                     NQ(NDGT)=JASAV*JBSAV                                   0000000
120                     NR(NDGT)=JASAV                                         0000000
      C                                                                        0000000
      C                                                                        0000000
121     450 RETURN                                                            0000000
122         END                                                               0000000
```

```
      SUBROUTINE MSDV(NA,NDIV,NQ,NREM)                                  0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                       * 0000000
C       DA=NDIV*NQ + NREM                                               * 0000000
C       NDIV IS A FORTRAN INTEGER LESS                                  * 0000000
C       THAN NBASE                                                      * 0000000
C                                                                       * 0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                       0000000
      DIMENSION  NA(1),NQ(1) NWA(50)                                    0000000
      COMMON     /IMULT/NDGT,NBASE,NLTH,LP,IN                           0000000
           IF(NDIV)40,30,40                                             0000000
   30      WRITE(LP,35)                                                 0000000
   35      FORMAT(//20X,'****DIVISOR IS ZERO****')                      0000000
           GO TO 170                                                    0000000
   40      NSAVE=IABS(NDIV)                                             0000000
           DO 50 J=1,NDGT                                               0000000
   50         NWA(J)=0                                                  0000000
           KLOOP=NA(NLTH)                                               0000000
           NR=0                                                         0000000
C                                                                       0000000
C   COMPUTE QUOTIENT AND REMAINDER                                      0000000
C                                                                       0000000
   60      NQUO=NR*NBASE+NA(KLOOP)                                      0000000
           NWA(KLOOP)=NQUO/NSAVE                                        0000000
           NR=MOD(NQUO,NSAVE)                                           0000000
           KLOOP=KLOOP-1                                                0000000
           IF(KLOOP)70,70,60                                            0000000
   70      NREM=NR                                                      0000000
C                                                                       0000000
C   SET SIGN                                                            0000000
C                                                                       0000000
           IF(NA(NDGT)*NDIV)80,90,90                                    0000000
   80      NWA(NDGT)=-1                                                 0000000
           GO TO 100                                                    0000000
   90      NWA(NDGT)=1                                                  0000000
   100     NPA=NA(NLTH)                                                 0000000
           NREM=NA(NDGT)*NREM                                           0000000
C                                                                       0000000
C   SET LENGTH POINTER                                                  0000000
C                                                                       0000000
           IF(NWA(NPA))140,110,140                                      0000000
   110     IF(NPA-1)130,130,120                                         0000000
   120     NWA(NLTH)=NPA-1                                              0000000
           GO TO 150                                                    0000000
   130     NWA(NLTH)=1                                                  0000000
           GO TO 150                                                    0000000
   140     NWA(NLTH)=NPA                                                0000000
   150     DO 160 J=1,NDGT                                              0000000
   160        NQ(J)=NWA(J)                                              0000000
C                                                                       0000000
C                                                                       0000000

   170 RETURN                                                           0000000
       END                                                              0000000
```

```
      FUNCTION IMIF(NA,NB)                                              0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                       * 0000000
C     COMPARE TWO POSITIVE NUMBERS                                      * 0000000
C       IMIF=0      IF NA=NB                                            * 0000000
C       IMIF=1      IF NA>NB                                            * 0000000
C       IMIF=2      IF NA<NB                                            * 0000000
C                                                                       * 0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 0000000
C                                                                       0000000
      DIMENSION NA(1),NB(1)                                             0000000
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                            0000000
           JFLAG=0                                                      0000000
           IF(NA(NLTH)*NB(NLTH))30,30,40                                0000000
   30      WRITE(LP,35)                                                 0000000
   35      FORMAT(//20X,'*****CHECK INPUT DATA*****')                   0000000
   40      LTS=NA(NDGT)-1+(NB(NDGT)+1)/2                                0000000
           IF(LTS)50,60,80                                              0000000
   50      IF(LTS+1)70,90,90                                            0000000
   60      GO TO 140                                                    0000000
   70      JFLAG=-1                                                     0000000
   80      JA=NA(NLTH)                                                  0000000
           JB=NB(NLTH)                                                  0000000
           IF(JA-JB)90,110,140                                          0000000
   90      IMIF=2+JFLAG                                                 0000000
           GO TO 160                                                    0000000
   100     JA=JA-1                                                      0000000
   110     IF(JA)150,150,120                                            0000000
   120     IF(NA(JA)-NB(JA))130,100,140                                 0000000
   130     GO TO 90                                                     0000000
   140     IMIF=1-JFLAG                                                 0000000
           GO TO 160                                                    0000000
   150     IMIF=0                                                       0000000
   160     JRET=IMIF                                                    0000000
   180     RETURN                                                       0000000
           END                                                          0000000
```

```
      SUBROUTINE MFLT(XREAL,KA)                                          0000000
C * *MFLT* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    0000000
C                                                                        *  0000000
C     PURPOSE       TRANSLATE FLOATING POINT NUMBER TO                   *  0000000
C                   MULTIPLE PRECISION INTEGER                           *  0000000
C                                                                        *  0000000
C     PARAMETERS                                                         *  0000000
C                XREAL     THE FLOATING POINT TO BE TRANSLATED           *  0000000
C                          MULTIPLE PRECISION INTEGER                    *  0000000
C                KA        THE RESULT AFTER TRANSLATED  XREAL            *  0000000
C                                                                        *  0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  0000000
C                                                                           0000000
      DIMENSION KA(1)                                                    0000000
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN                             0000000
C                                                                        0000000
C     INITIALIZATION                                                     0000000
C                                                                        0000000
      DO 30 J=1,NDGT                                                     0000000
   30    KA(J)=0                                                         0000000
      RBASE=NBASE                                                        0000000
      SAVE=XREAL                                                         0000000
      L=0                                                                0000000
C                                                                        0000000
C     COUNT HOW MANY WORDS SHOULD BE USED                                0000000
C                                                                        0000000
   40 SAVE=SAVE/RBASE                                                    0000000
      L=L+1                                                              0000000
      IF(SAVE-1.0)50,40,40                                               0000000
C                                                                        0000000
C     STORE THE VALUE IN KA ,FROM LOW ORDER DIGIT TO HIGH ORDER DIGIT    0000000
C                                                                        0000000
   50 DO 60 J=1,L                                                        0000000
         K=L+1-J                                                         0000000
         XNORM=SAVE*RBASE                                                0000000
         KA(K)=XNORM                                                     0000000
         YNORM=KA(K)                                                     0000000
   60    SAVE=XNORM-YNORM                                                0000000
C                                                                        0000000
C     SET LENGTH, SIGN FIELDS OF KA                                      0000000
C                                                                        0000000
      KA(NLTH)=L                                                         0000000
      IF(XREAL)70,80,80                                                  0000000
   70 KA(NDGT)=-1                                                        0000000
      GO TO 90                                                           0000000
   80 KA(NDGT)=1                                                         0000000
   90 RETURN                                                             0000000
      END                                                                0000000
```

```
      SUBROUTINE MCMP(IX,IY,IR)                                          0000000
C * *MCMP * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    0000000
C                                                                        *  0000000
C     PURPOSE    - COMPARE IX AND IY , RETURN VALUE TO IR                *  0000000
C                  IR=-1  IF IX < IY                                     *  0000000
C                  IR=0   IF IX = IY                                     *  0000000
C                  IR=1   IF IX > IY                                     *  0000000
C                                                                        *  0000000
C     PARAMETERS                                                         *  0000000
C                IX     A MULTIPLE PRECISION INTEGER                     *  0000000
C                IY     A MULTIPLE PRECISION INTEGER                     *  0000000
C                IR     A FORTRAIN INTEGER                               *  0000000
C                                                                        *  0000000
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  0000000
      DIMENSION  IX(1),IY(1),IC(50)                                      0000000
      COMMON     /IMULT/NDGT,NBASE,NLTH,LP,IN                            0000000
C                                                                        0000000
      CALL MSUB(IX,IY,IC)                                                0000000
      IF(IC(NDGT) LT 0) GO TO 30                                         0000000
      IF(IC(NLTH) GT 1) GO TO 10                                         0000000
      IF(IC(1) EQ 0) GO TO 20                                            0000000
   10 IR=1                                                               0000000
      GO TO 40                                                           0000000
   20 IR=0                                                               0000000
      GO TO 40                                                           0000000
   30 IR=-1                                                              0000000
   40 RETURN                                                             0000000
      END                                                                0000000
```

```
      SUBROUTINE MSQRT(INPT,IOUT)                                       0000000
C * *MSQRT* * * * * * * * * * * * * * * * * * * * * * * * * * * * *     0000000
C                                                                       *
C       PURPOSE       IOTPT=SQRT(INPT)                                  *
C                                                                       *
C       PARAMETERS -                                                    *
C             BOTH INPT AND IOTPT ARE MULTIPLE PRECISION                *
C             INTEGER                                                   *
C                                                                       *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      INTEGER    INPT(1),IOUT(1),IOTPT(50),LIN(50),KA(50),KB(50),
     1    KC(50),KD(50),KE(50),TWO(50),ZERO(50)
      COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN
C
C    INITIALIZATION
C
      DO 30 J=1,NDGT
         IOTPT(J)=0
   30    KA(J)=INPT(J)
      CALL KINPT(0,ZERO,0)
      LTHA=INPT(NLTH)
      IF(LTHA-2)40,40,80
   40   IF(LTHA-1)60,60,50
C
C    INPT HAS TWO WORDS LONG
C
   50   IOTPT(1)=SQRT(FLOAT(KA(2)*NBASE+KA(1)))
        GO TO 70
C
C    INPT HAS ONLY ONE WORD
C
   60   IOTPT(1)=SQRT(FLOAT(KA(1)))
C
C    SET SIGN AND LENGTH FIELDS
C
   70   IOTPT(NDGT)=1
        IOTPT(NLTH)=1
        CALL MADD(IOTPT,ZERO,IOUT)
        GO TO 200
CCCC
CC    WHEN LENGTH OF INPT GREATER THAN 2
CCCC
C     COUNT THE DIGITS OF KA
C
   80   NDIG=(KA(NLTH)-1)*2
        IF(KA(1))100,90,100
   90   NDEC=1
        GO TO 110
  100   NDEC=ALOG10(FLOAT(KA(1)))+1

  110   NDIG=NDIG+NDEC
C
C     APPROXIMATION TO SQRT(KA)
C
        IF(NDIG-2*(NDIG/2))130,120,130
  120   NEXP=(NDIG-4)/2
        GO TO 140
  130   NEXP=(NDIG-3)/2
  140   RAP=KA(LTHA)*NBASE+KA(LTHA-1)
        RAP=SQRT(RAP)*10.**NEXP
        CALL MFLT(RAP,KB)
        CALL KINPT(2,TWO,0)
C
C     NEWTON ITERATION
C
  150   CALL MMPY(KB,KB,KC)
        CALL MSUB(KA,KC,KD)
        CALL MMPY(TWO,KB,KE)
        CALL MDIV(KD,KE,KD,KE)
        IF(KD(NLTH)-1+KD(1))160,170,160
  160   CALL MADD(KB,KD,KB)
        GO TO 150
  170   CALL MCMP(KC,KA,IR)
        IF(IR)190,190,180
  180   CALL KINPT(-1,KC,0)
        CALL MSUB(KB,KC,IOUT)
        GO TO 200
C
C   COPY RESULT TO IOTPT
C
  190   CALL MADD(KB,ZERO,IOUT)
  200   RETURN
        END
```

```
1         SUBROUTINE MEQL(KA,KB)
      C*************************************************************
      C                                                            *
      C                                                            *
      C         KB=KA                                              *
      C                                                            *
      C*************************************************************
2             DIMENSION KA(1),KB(1)
3             COMMON    /IMULT/NDGT,NBASE,NLTH,LP,IN
      C
4             DO 20 J=1,NDGT
5                KB(J)=KA(J)
6        20   CONTINUE
7             RETURN
8             END
```

VITA

CHUNG-PEI CHU

Candidate for the Degree of

Master of Science

Thesis: A PORTABLE PACKAGE OF THE SPECTRAL TEST FOR PSEUDO-RANDOM NUMBER GENERATORS

Major Field: Computing and Information Science

Biographical:

Personal Data: Born in Taipei, Taiwan, Republic of China, November 26, 1952, the daughter of Yen Kang and Yung Chen Chu. Married to Shen Then Chang on May 11, 1980.

Education: Graduated from Gi Zen High School, Taipei, Taiwan, R.O.C., in June, 1971; received Bachelor of Science Degree in Education from National Taiwan Normal University in June, 1976; completed requirements for the Master of Science Degree at Oklahoma State University in July 1986.

Professional Experience: Mathematics Teacher, Hu Kung Senior High School, Taipei, Taiwan, August, 1976, to June, 1978; Mathematics Tutor, Department of Mathematics, University of Wisconsin in Whitewater , September, 1978, to May, 1979; Programmer, Business and Economic Research Center, Oklahoma State University, Feburary, 1981, to December, 1982.