AD-A170 465

(12)

AD

AMSMC/SA/MR-8

METHODS FOR CALCULATING THE PROBABILITY DISTRIBUTION OF SUMS OF INDEPENDENT RANDOM VARIABLES

GEORGE J. SCHLENKER

JULY 1986

Distribution Statement

Approved for public release; distribution unlimited.

DTIC
ELECTE
1986
B

DTIC FILE COPY

U.S. ARMY ARMAMENT, MUNITIONS AND CHEMICAL COMMAND
SYSTEMS ANALYSIS OFFICE
ROCK ISLAND, ILLINOIS 61299-6000

86 8 1 014

DISPOSITION

Destroy this report when no longer needed. Do not return it to the originator.

DISCLAIMER

The findings in this report are not to be construed as an official position of either the Department of the Army or of the US Army Armament, Munitions and Chemical Command.

AMSMC/SA/MR-8

METHODS FOR CALCULATING THE PROBABILITY DISTRIBUTION OF SUMS OF INDEPENDENT RANDOM VARIABLES

GEORGE J. SCHLENKER

July 1986

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AMSMC/SA/MR-8	2. GOVT ACCESSION NO. AD-A170 465	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Methods for Calculating the Probability Distribution of Sums of Independent Random Variables		5. TYPE OF REPORT & PERIOD COVERED Report - Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) George J. Schlenker		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Armament, Munitions and Chemical Command Systems Analysis Office Rock Island, IL 61299-6000		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE July 1986 13. NUMBER OF PAGES 74
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Additions to/from the DISTRIBUTION LIST are invited and should be forwarded to the following address: Commander, U.S. Army Armament, Munitions and Chemical Command, ATTN: AMSMC-SAS, Rock Island, IL 61299-6000
AUTOVON 793-5041/6370

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Operations Research	Numerical Methods
Statistical Analysis	Numerical Convolutions
Distribution Theory	Integral Transforms
Network Theory	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report surveys numerical methods for obtaining the probability distribution of a sum of statistically independent random variables. Study objectives are to investigate the relative accuracy and computational effort for each of the following methods: (a) evaluation of closed-form solutions for particular cases, (b) discrete numerical convolution of probability densities, (c) Normal probability approximation to the distribution of a sum, (d) numerical inversion of the Laplace transform of the convolution, (e) Erlang approximation for

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Block 20 (cont'd):

convolutions of a two-parameter Weibull distribution, (f) convolution of probability densities using the FFT algorithm for calculating finite Fourier transforms, and (g) Monte-Carlo simulation.

Methods are sketched for deriving analytic expressions for the distribution of the sum of RVs of certain distributions. Each numerical method is described and illustrated using RVs from several distributional forms, such as uniform, exponential, gamma, and Weibull, as well as mixture models. In terms of run time and accuracy, some methods are particularly suited to certain distributional forms. If problem applications are quite special and if the time for program coding (as well as running) is a consideration, Monte-Carlo simulation may be the preferred method. All computer source programs are listed in annexes.

QUALITY INSPECTED
QUALITY INSPECTED
QUALITY INSPECTED
DTIC
ELECTE
AUG 4 1986
S D
B
Dist
A-1

EXECUTIVE SUMMARY

This report describes the results of a study of several numerical methods for calculating points on the distribution of a sum of statistically independent random variables. The report is directed to practitioners of statistical and numerical methods. Immediate motivation for the study arose in connection with the random time to accomplish a collection of tasks. However, application to a variety of problems is anticipated because of the generality of the methods.

Study objectives are to investigate the relative accuracy and computational effort, viz, run time, for each of the following methods:

(a) Evaluation of closed-form solutions for particular cases.

(b) Discrete numerical convolution of probability densities.

(c) Normal probability approximation to the distribution.

(d) Numerical inversion of the Laplace transform of the convolution. (Bellman's method).

(e) Erlang approximation for convolutions of a two-parameter Weibull distribution. (Johnson's method).

(f) Convolution of probability densities using the FFT algorithm for calculating finite Fourier transforms.

(g) Monte-Carlo simulation.

Normal approximation for sums of independent random variables (RVs) is made in several areas, including quality control and analytic network theory. Because of the frequently uncritical assumption of Normality, the error of this approximation is a particular focus here.

Methods are sketched for deriving analytic expressions for the distribution of the sum of RVs of certain distributions. Each numerical method is described and illustrated using RVs from several distributional forms, such as uniform, exponential, gamma, and Weibull, as well as mixture models. In terms of run time and accuracy, some methods are particularly suited to certain distributional forms. If problem applications are quite special and if the time for program coding (as well as running) is a consideration, Monte-Carlo simulation may be the preferred method. All computer source programs are listed in annexes.

CONTENTS

LIST OF TABLES

AMSMC-SAS July 1985

MEMORANDUM REPORT

SUBJECT: Methods for Calculating the Probability Distribution of Sums of Independent Random Variables

1. References

a. Bellman, R.E., Kalaba, R.E., and Shiffman, B. A Numerical Inversion of the Laplace Transform, RM-3513-ARPA, The RAND Corp, Santa Monica, CA, April 1963.

b. Schlenker, G. Numerical Methods in Renewal Theory, OR-68-2, (AD 828276), USA WECOM, Rock Is, IL, Feb 1968.

c. Schlenker, G. Reliability and Maintainability of the M48A1 Tank, OR-63-2(AD439501) USA WECOM, Rock Is, IL, Apr 1963.

2. Background

Problems associated with the distribution of sums of independent random variables (RVs) occur in various analyses. Examples are: (a) estimating the total cost of an end item or a project, given component cost estimates; (b) estimating time to complete a series of sequential tasks, (This problem can be generalized to estimating the completion time for a series of networks.); (c) estimating dimensional variability in an assembly of serially arranged parts; and (d) estimating statistical confidence limits on the mean of a random variable (RV). Changing problem context may obscure the mathematical identity of these familiar problems. Often, the probability distribution of the sum is assumed to be Normal, since the central limit theorem guarantees Normality as the number of RVs in the sum becomes infinitely large. However, if either tail of the distribution of the sum of a small number of independent RVs is to be estimated with accuracy, it is prudent to be cautious in immediately assuming Normality. This report addresses the issue of accuracy of a Normal approximation and other issues associated with different methods of calculating the cumulative distribution function (c.d.f.) of a sum of n random variables (RVs), when the RVs have a variety of distributional forms.

3. Study Objectives

Specific objectives of the study reported here are: (a) identify the error of approximation for the c.d.f. of an n-component sum as n increases; This error is examined for cases in which all of the components have the same distribution and for cases in which the form of the distribution is the same but in which the parameters are unique. (b) obtain closed-form expressions for the c.d.f. of the sum for special cases, which may be used to check various numerical

methods; (c) obtain measures of computational effort for several numerical methods for comparative purposes; and (d) suggest which methods are suitable for treating particular cases.

4. Discussion

The sum of two RVs has a distribution which is the mathematical convolution of the component distributions. Thus, the sum of n RVs has a distribution which is the n-fold [1] convolution of the component distributions. The problem of calculating the distribution of the sum is, then, equivalent to obtaining an n-fold convolution. This fact can be exploited in calculating the c.d.f. of the sum numerically and in obtaining a formula for the c.d.f. of the sum. Consider the case of two continuous RVs, defined on 0 to infinity. Call the variables x and y and call the sum z. The c.d.f. of any RV will be denoted by F(.), with a subscript referring to the variable of interest. Similarly, the notation for the probability density function (p.d.f.) of interest will be f(.), with a specifying subscript. Thus, the c.d.f. and p.d.f. of the RV x are, respectively,

$F_x(x)$ and $f_x(x)$. For this case, the convolution theorem yields:

$$F_z(z) = \text{Integral}(0,z): F_y(z - t)\, f_x(t)\, dt \,. \qquad (1)$$

Similarly, from (1), the p.d.f. of z is written as

$$f_z(z) = \text{Integral}(0,z): f_y(z - t)\, f_x(t)\, dt. \qquad (2)$$

For specific distributional forms the indicated integration may be simple to carry out. If so, a closed form expression for the desired convolution is obtained. If not, one can use discrete numerical convolution. The numerical equation is obtained from equation (1) by discretizing the domains of the functions at, say, m identical points: t(i), for 1 le i le m. Then, the diferential form $f_x(t)\, dt$ is replaced by a probability difference and the integral becomes a sum, as follows:

$$F_z(z(k)) = \text{Sum } i=1,k: F_y(z(k)-t(i))\,(F_x(t(i)-F_x(t(i-1)))). \qquad (3)$$

The accuracy of the numerical method improves by increasing the number (m) of discrete points, assuming that the range t(m) - t(1) adequately covers the domain of the c.d.f. of z in the sense that the upper-tail probability beyond z(m) is negligible--say, 1/100,000. If numerical convolutions are to be performed recursively, it is necessary to anticipate the domain of the highest-order convolution when choosing z(m). Because of the need for a high density of discrete points, the size of m generally becomes quite large for four or more convolutions. Clearly, this situation produces a computational burden which increases rapidly with the order (n) of convolution. For n greater than, e.g. 4, other numerical methods may be preferred on the basis of efficiency.

---

[1] Order of convolution is defined here as the number of distributions being convolved. This is equal to the # of RVs in a sum.

5. Integral Transforms

In dealing with convolutions it is helpful to use a theorem [1] from the theory of integral transforms--either Laplace or Fourier. That theorem states that the transform of a convolution of two function is the product (in the complex plane) of the transforms of the functions. Following a UK convention, I denote the Laplace transform of a function with the function symbol having an asterisk superscript. For example, the Laplace transform of f(x) is f*(s), with complex argument s. Thus, the p.d.f. of z in (2) can be characterized by the transform:

$$f^*_z(s) = f^*_y(s)\, f^*_x(s) . \qquad (4)$$

If the RV z is added to another RV w yielding the sum v, one can immediately write the transform of the p.d.f. of v as

$$f^*_v(s) = f^*_y(s)\, f^*_x(s)\, f^*_w(s) , \qquad (5)$$

instead of convolving $f_y(y)$ with $f_x(x)$, and the result with $f_w(w)$.

Of course, it is necessary to be able to invert the transform to achieve the desired result. More will be said about this later. For many probability distributions of interest, the Laplace transform can be written in simple form. Examples are the uniform distribution on (0,a), which has the Laplace transform of the p.d.f.:

$$f^*(s) = (1 - \exp(-as))/a/s, \qquad (6)$$

and the exponential distribution with rate parameter r, whose p.d.f. transform is

$$f^*(s) = r/(s + r). \qquad (7)$$

If each of the n random variables in the sum has the same distribution, the transform of the p.d.f. of the sum is just the nth power of the transformed p.d.f. For the sum of n uniform (0,1) deviates, yielding the RV t,

$$f^*_t(s) = (1 - \exp(-s))^n / s^n . \qquad (8)$$

The Laplace transform of the c.d.f. of t is obtained from the transformed p.d.f. simply by dividing by s, since the c.d.f. is just the integral of the p.d.f. To facilitate inversion, the expression for the n th power of 1 - exp(-s) is be expanded as a sum of binomial terms:

$$\text{Sum over } i\ (0,n):\ C(n,i)\,(-1)^i \exp(-is),$$

where C(n,i) is the # of combinations of n objects taken i at a time.

---

[1] An exposition of the theorem is found, e.g., in Jenkins, G.M. and Watts, D.G. Spectral Analysis, Holden-Day, c. 1968.

The transformed c.d.f. of t can be readily inverted analytically:

$$F_t(t) = \text{Sum over } i\ (0,n):\ (-1)^i\, C(n,i)\, u(t-i)\, (t-i)^n/n!\ , \qquad (9)$$

where u(t-x) is the unit step function at x. This expression is quickly and accurately evaluated, even for large values of n (n > 10). Calculation was performed by the routine NFOLD.U on the Prime 9955 minicomputer with a run time limited by the print buffer, i.e., in a fraction of a second. This program is listed in Annex A.

6. If the RV of interest (t) is the sum of n identical exponential RVs, the Laplace transform of the p.d.f. of t is, from (7),

$$f_t^*(s) = r^n/(s + r)^n\ . \qquad (10)$$

This expression also has a simple inverse:

$$f_t(t) = r^n\, t^{n-1}\, \exp(-rt)/(n-1)! \qquad (11)$$

This is recognized as a gamma p.d.f. with shape parameter n and rate parameter r. This result illustrates the familiar theorem that the sum of n identical exponential RVs has an Erlang distribution, i.e., a gamma distribution with integer shape parameter. It follows immediately from (10) that the sum of N identical gamma distributions, having shape parameter n, is also a gamma distribution with shape parameter Nn, since the Laplace transform of its p.d.f.

$$f_t^*(s) = r^{Nn}/(s + r)^{Nn}\ , \qquad (12)$$

has the same form as the transform of the gamma p.d.f. in (10). Altho the Laplace transforms in these examples have simple inverses, transforms are still useful in calculating convolutions of probability distributions when this condition does not exist. The reasons for this assertion are: (a) that the transforms of the distributions being convolved are often simple functions of s, (b) that the product of such transforms are easy to evaluate, and (c) that numerical methods exist for calculating the inverse Laplace transform. One such method was developed by Richard Bellman (Ref [1a]). I have found this method useful in several applications, such as in solving integral equations (Ref [1b]) as well as for obtaining the distribution of sums of RVs. Further discussion of Bellman's method is deferred to a later point.

7. Measures of Accuracy

Several measures can be used in describing the accuracy of numerical methods for approximating the c.d.f. of a sum of RVs. Two are used here: (a) the maximum absolute error over a finite set on the domain of the c.d.f., and (b) the square root of the mean squared error or RMS error, evaluated over the same set of points. For most methods in this study, I have used 20, equally-spaced points on the domain of the c.d.f., such that tail probabilities are less than 0.01 beyond the range of points used. An exception to this selection of points

is made when using Bellman's method. In that case 16, log-spaced points are used to span the range of the sum. As an illustration of these measures, consider the Normal approximation to the N-fold convolution of a standard (0,1) uniform distribution. Using the exact result, given in (9), the measures of error are calculated for several choices of N. These are shown in Table 1.

TABLE 1
ERROR IN THE NORMAL APPROXIMATION OF CONVOLUTIONS OF N STANDARD UNIFORM PROBABILITY DISTRIBUTIONS

N	Max Abs Error	RMS Error
2	0.0164	0.0096
3	0.0097	0.0054
4	0.0074	0.0038
5	0.0057	0.0029
10	0.0028	0.0013

In many applications, the error associated with a very large (small) value of the c.d.f. is more appropriate than either of the above error measures. For 5 convolutions of a uniform distribution, the error of a Normal approximation is about 0.1% for values of the c.d.f. > 0.95. By nearly any measure, 4 or 5 convolutions of a given uniform distribution is well approximated by a Normal distribution whose mean and variance are N times the uniform mean and variance. However, not all distributions of sums of uniform RVs are this well approximated by a Normal c.d.f. The case of sums of different uniform RVs is considered below. One may ask if Monte-Carlo simulation is competitive in terms of accuracy--if not in terms of run time--with a Normal approximation. For the case considered above, 20 thousand Monte-Carlo replications produces a typical RMS error of 0.002 to 0.003. This is about the same accuracy as the Normal approximation for N = 5. The run time for 20,000 replications on the Prime 9955 is approximately a linear function of the number of RVs in the sum. For this case, approximate run time T, in seconds, is given by

$$T = 8(N - 2) + 30. \qquad (13)$$

For simple cases such as this, Monte-Carlo is quite expensive in terms of run time. However, Monte Carlo becomes more attractive when the problem becomes mathematically intractable.

8. Sums of Non-identical Uniform RVs

The n-fold convolution of the standard uniform distribution was obtained in closed form (9) by inversion of the Laplace transform, given in (8). This result can be generalized by permitting each of the n uniform distributions to have a different range, but with common threshold parameter. Thus, the kth member of the set is defined on, say, 0 to a(k). The Laplace transform of the p.d.f. of the sum (t) is, then,

$$f^*_t(s) = \text{Product over } k=1 \text{ to } n: \ (1 - \exp(-a(k)s)/(a(k)s). \qquad (14)$$

The inverse transform is somewhat complicated to obtain, and is not derived here. The exact c.d.f. for the sum of n different uniform RVs is simply presented, with the following definitions, as

$$F_t(t) = 1/a/n![t^n + \text{Sum over } k\ (1,n):\ (-1)^k \text{ Sum over } j\ (1,C(n,k)):$$

$$u(t - S_{kj}(n))(t - S_{kj}(n))^n\ ], \qquad (15a)$$

where

$$a = \text{Product over } k=1 \text{ to } n:\ a(k)$$

and where

$S_{kj}(n)$ is the j th sum of the k tuple of n values of a(i), taken

k at a time. For example,

$$S_{1j}(n) = a(j), \qquad 1 \le j \le n,$$

$$S_{21}(n) = a(1)+a(2) \quad \text{and} \quad S_{2C(n,2)} = a(n-1)+a(n). \qquad (15b)$$

The implementing computer program, given in Annex B, calculates the error of a Normal approximation for a larger class of sums of uniform RVs. Consider the following special case in which the range of the k th uniform RV in the sum of n is k. Normal errors in the c.d.f. of the sum are given in Table 2. Compare with Table 1. Note that the errors are about twice those in Table 1. However, even these errors are relatively small (1% or less) for N = 5.

TABLE 2
ERROR IN THE NORMAL APPROXIMATION OF CONVOLUTIONS OF N DIFFERENT UNIFORM PROBABILITY DISTRIBUTIONS *

N	Max Abs Error	RMS Error
2	0.0321	0.0188
3	0.0179	0.0104
4	0.0131	0.0071
5	0.0102	0.0054
10	0.0049	0.0024

* Range of the k th uniform RV is taken to be k.

9. Sums of Identical Exponential RVs

For a somewhat different picture, consider the case of a sum of N exponential RVs from the same c.d.f. The error of a Normal approximation is shown in Table 3 as a function of N.

TABLE 3
ERROR IN THE NORMAL APPROXIMATION OF N CONVOLUTIONS OF
AN EXPONENTIAL PROBABILITY DISTRIBUTION

N	Max Abs Error	RMS Error
2	0.0945	0.0436
3	0.0769	0.0365
4	0.0648	0.0316
5	0.0596	0.0278
10	0.0416	0.0183
15	0.0340	0.0142

Tne errors shown in Table 3 are about one order of magnitude greater than those in Table 1, indicating that sums of exponential RVs approach Normality much more gradually than sums of uniform RVs. For a sum of 15 exponentials, the Normal error is about 1% or less for values of the c.d.f. > 0.98. Clearly, this example indicates a need for caution in applying the Normal assumption.

10. Sums of Different Exponentials

The p.d.f. of the sum of exponential RVs from the same distribution was shown (11) to have the Erlang form. If a set of n exponential RVs from distributions with unique mean values are summed, tne form of the c.d.f. is somewhat complicated. However, an analytic model exists for this, more general case. The computer program which is used for evaluating this distribution is found in Annex C. If the rate parameter, r(k), of the distribution of an arbitrary kth RV is unique, the c.d.f. of the sum of n RVs is given by

$$F_t(t) = r \sum_{i=1}^{n} A(i)(1 - \exp(-r(i)t))/r(i), \quad (16)$$

where $r = \prod_{k=1}^{n} r(k)$,

and where the vector A(*) is the solution of a certain matrix equation: M A = B. Elements of the B vector are all zero except the nth (last). A typical element of M, m(i,j), involves the sum of all (i-1) tuple products of r(k), with k not = to j. Thus, e.g.,

$$m(3,j) = \sum_{k \neq j} \sum_{l > k,\ l \neq j} r(k)r(l).$$

The 1st row of M has elements = 1. Other rows are like tne one above. Equation (16) can be used to calculate the Normal c.d.f. error for a special case. In a set of n exponential RVs, let the range of the mean values be fixed at 2. Let the kth RV have the mean value 1 + (k-1)/(n-1). Tne Normal errors for the c.d.f. of tne sum of these RVs are shown in Table 4. Comparison with the results of Table 3 indicates that greater errors of Normal approximation occur when the RVs in the sum have different mean values. For this example the error is about 10% greater than for the n-fold convolution of the same exponential distribution.

TABLE 4
ERROR IN THE NORMAL APPROXIMATION OF CONVOLUTIONS OF
N DIFFERENT EXPONENTIAL PROBABILITY DISTRIBUTIONS*

N	Max Abs Error	RMS Error
2	0.1036	0.0477
3	0.0823	0.0393
4	0.0693	0.0338
5	0.0634	0.0297
10	0.0436	0.0195

* For the kth RV in a set of n, let the mean value be (k-1)/(n-1).

11. An Exponential Mixture Model

In forming a sum of independent RVs, one may think of each RV as the duration of a particular activity in a serial network of n activities. The distribution of the sum is, then, the distribution of completion time for the network. A variation of this model is one in which the nodes, separating activities, permit two exit paths, each of which has a given probability of being taken. If either activity can occur prior to the next network node, the form of the probability distribution for the transit time to next node is a mixture of the distributions for the alternate activity times, with weights equal to the probability the activity is taken. This model is a particular instance of a semi-Markov process, a type of stochastic process frequently observed in industrial operations. An interesting special case of a two-component mixture model is one in which the components (alternate activities) are exponentially distributed. The form of the c.d.f. for this inter-node duration is

$$F(t) = a(1 - \exp(-r1\ t)) + (1-a)(1 - \exp(-r2\ t)), \qquad (17)$$

where a is the weight associated with the first component, and with rate parameters r1 and r2 for the 1st and 2nd component distributions, respectively. The p.d.f for this mixture model is

$$f(t) = a\ r1 \exp(-r1\ t) + (1-a)\ r2 \exp(-r2\ t). \qquad (18)$$

The sum of n such "activities" will have a distribution denoted by $g_n(t)$, for the p.d.f., and by $G_n(t)$, for the c.d.f. of time t. Using the convolution theorem of Laplace transforms, the transform of $g_n(t)$ can be written as

$$g_n^*(s) = [a\ r1/(s + r1) + (1-a)\ r2/(s + r2)]^n . \qquad (19)$$

To facilitate obtaining an inverse, this expression is expanded in a power series of terms in

$$1/(s + r1)^i\ /(s + r2)^{n-i} .$$

The mixed products in this series must, then, be expressed in a continued fraction expansion. This result can be inverted term by term. For example, for n = 2, the Laplace transform after the indicated operations is

$$g^*_2(s) = (a\ r1)^2/(s+r1)^2 + 2a(1-a)r1r2/(r1-r2)/(s+r2) + ((1-a)r2)^2/(s+r2)^2 + 2a(1-a)r1r2/(r2-r1)/(s+r1). \quad (20)$$

The inverse transformation is obtained by inspection.

$$g_2(t) = (a\ r1)^2\ t\ \exp(-r1\ t) + ((1-a)r2)^2\ t\ \exp(-r2\ t) + 2a(1-a)r1r2/(r1-r2)(\exp(-r2\ t) - \exp(-r1\ t)) . \quad (21)$$

Integrating $g_2(t)$ produces the c.d.f.:

$$G_2(t) = 1 - a^2\ (1+r1\ t)\ \exp(-r1\ t) - (1-a)^2\ (1+r2\ t)\ \exp(-r2\ t) - 2a(1-a)/(r1-r2)(r1\ \exp(-r2\ t) - r2\ \exp(-r1\ t)). \quad (22)$$

Closed-form expressions for $G_n(t)$ for larger values of n are found in the implementing computer program in Annex D. These expressions are used to calculate the Normal approximation error for the c.d.f. Results are shown in Table 5 for a numerical example in which the parameter a = 0.8, and the mean values of the first and second components are in the ratio of 0.05 to 1.0. Rate parameters r1 and r2 are adjusted to always yield a mean value of the sum equal to unity. The last practice assures that the same points are evaluated in the domain of the c.d.f. regardless of the value of n. Also note that the computer program (LP.INV) uses 16 log-transformed points at which the c.d.f. error is evaluated--not the usual 20.

TABLE 5
ERROR IN THE NORMAL APPROXIMATION OF N CONVOLUTIONS OF A TWO-COMPONENT EXPONENTIAL MIXTURE PROB DISTRIBUTION

N	Max Abs Error	RMS Error (16 points)
2	0.286	0.198
3	0.242	0.166
4	0.213	0.142
5	0.188	0.124
10	0.117	0.074
15	0.092	0.052

The maximum absolute errors in Table 5 are nearly 3 times the corresponding errors in Table 3, which referred to convolutions of a single exponential component. Further, the RMS errors in Table 5 are about 4.0 to 4.5 times the corresponding errors in Table 3. These observations indicate that the sum of n RVs from an exponential mixture distribution may converge VERY SLOWLY, with increasing n, toward Normality. In fact, the approximation errors in the c.d.f. of the sum may be much greater than comparable errors in sums of exponential RVs, (which are even quite large). A frequently used rule of thumb for deciding what is a marginally large sample size in many statistical applications is that $n > 30$ is "large". However, for the 30-fold convolution of the exponential mixture c.d.f., one finds that the approximating Normal c.d.f. has a max absolute error of nearly 0.057 and an RMS error of about 0.026. When dealing with sums of RVs from a semi-Markov process, considerable inaccuracy can be encountered in taking a Normal approximation. This is the lesson of this particular example.

12. Monte-Carlo Simulation

As is shown above, closed-form expressions can be obtained for convolutions of an exponential mixture distribution by using Laplace transform methods. However, the complexity of inverting $G^*(s)$ grows rapidly with n. In this case in particular, alternatives to evaluating formulas are sought for calculating points of $G^{n}(t)$ for large values of n. As suggested above in paragraph 7, p.5, Monte-Carlo is a useful and quite general technique. For example, in the case of the exponential mixture model, generation of one RV from the mixture distribution involves: (a) drawing one uniform (0,1) deviate, U; (b) drawing a RV from an exponential distribution with rate parameter r1, if $U < a$; or (c) otherwise, drawing a RV from an exponential distribution having rate parameter r2. The sum of n such random variables is, of course, the RV of interest. Run time for generating an estimate of $G^{n}(t)$ by simulation is actually found to be somewhat less than that indicated by equation (13) for sums of uniform RVs, due to a different choice of points in the domain of the c.d.f. at which the distribution is evaluated. The particular numerical example, introduced in paragraph 11, is used to compare a Monte-Carlo estimate with a theoretical estimate and with a Normal approximation of $G^{n}(t)$. Results for two values of n are displayed in Table 6. The theoretical estimate is obtained by formula evaluation for $n = 3$, and is obtained by Bellman's numerical inversion method (to be discussed), for $n = 15$. For this problem the max absolute error in Bellman's method is quite small--typically < 0.001--making this a good theoretical estimate. Exponential rate parameters are scaled so that the mean of the sum is unity for both values of n. The effect of the time scaling makes the variance of the sum inversely proportional to n. Thus, the standard deviation of the sum is 1.4158 for $n=3$, and is 0.6332 for $n=15$, in this example. Note that the Normal approximation is quite poor at low quantiles, even for n as large as 15. Also note that the Monte-Carlo estimate is quite good for 20,000 replications. The max abs error is nearly 0.006, and the RMS error is about 0.003 for one random number stream.

TABLE 6
SEVERAL APPROXIMATIONS OF THE C.D.F. OF THE SUM OF N RV'S
FROM A TWO-COMPONENT EXPONENTIAL MIXTURE PROB DISTRIBUTION

	N = 3			N = 15		
Sum Value	Theory Eval'n	Monte Carlo	Norm Approx	Theory Eval'n	Monte Carlo	Norm Approx
0.0053	0.0000	0.0000	0.2412	0.0000	0.0000	0.0581
0.0281	0.0044	0.0043	0.2462	0.0000	0.0000	0.0624
0.0695	0.0432	0.0438	0.2555	0.0002	0.0000	0.0708
0.1304	0.1577	0.1580	0.2696	0.0027	0.0030	0.0848
0.2120	0.3256	0.3292	0.2889	0.0371	0.0336	0.1067
0.3161	0.4740	0.4796	0.3145	0.1070	0.1036	0.1400
0.4450	0.5668	0.5696	0.3475	0.1956	0.1938	0.1904
0.6024	0.6218	0.6259	0.3894	0.3109	0.3088	0.2650
0.7930	0.6646	0.6676	0.4419	0.4473	0.4478	0.3718
1.0239	0.7073	0.7123	0.5067	0.5948	0.5929	0.5150
1.3057	0.7519	0.7542	0.5855	0.7360	0.7330	0.6854
1.6552	0.7981	0.7996	0.6782	0.8545	0.8571	0.8496
2.1013	0.8451	0.8473	0.7817	0.9370	0.9381	0.9590
2.7003	0.8918	0.8949	0.8851	0.9817	0.9811	0.9964
3.5859	0.9367	0.9394	0.9661	0.9974	0.9974	1.0000
5.2401	0.9771	0.9764	0.9986	1.0000	1.0000	1.0000

RMS errors in the Monte-Carlo c.d.f. estimate seem to vary inversely as the square root of the sample size (S), over the range from 5 to 20 thousand replications, and do not vary statistically with the order (N) of the convolution. Typical RMS errors for this example over this range in S vary from 0.002 to 0.004. An approximation for Monte-Carlo run time (sec) on the Prime 9955 is

$$T = S\,N/4, \tag{23}$$

where S is given in thousands of replications, and with 2 le N le 20. Run time--as opposed to c.p.u. time--is dependent on the number of other users sharing the couputer and upon the nature of their jobs. The value of T given here is representative of the active part of a work day.

13. Bellman's Metnod

A numerical method is given in Ref 1a by Bellman for inverting Laplace transforms. Derivation of the method proceeds from the definition of the Laplace transform of an analytic function F(t):

$$F^*(s) = \text{Integral}(0,\inf):\ \exp(-st)\,F(t)\,dt. \tag{24}$$

The variable of integration is changed to x via the transformation

$$t(x) = \ln\,(2/(x + 1)). \tag{25}$$

Then, the real variable x is defined on (-1,1). Now, the transform variable, s, is replaced with a discrete real variable c k, with c constant and k integer, 1 le k le m. That is, the transform is evaluated at discrete, evenly spaced points on the real line. The variable of integration is also discretized at m points, x(j), 1 le j le m. Thus, the integration operation is replaced by a summation. Gaussian quadrature is chosen as the mean of evaluating the integral. The x(j) are chosen as the points of the independent variable in an mth order quadrature. Notationally, let

$$g(j) = F(t(x(j))), \qquad j = 1, 2, \ldots, m. \quad (26)$$

The weight function [1] for gaussian quadrature is denoted by w(j), for 1 le j le m. With this notation, equation (24) becomes

$$F^*(ck) = \text{Sum over } j \ (1,m): \ 0.5\, w(j)((x(j)+1)/2)^{ck-1}\, g(j), \quad (27)$$

for k = 1, 2, ..., m. This equation is seen to be a matrix equation, which can be written compactly as

$$\underline{F}^* = \underline{\underline{A}}\ \underline{g}, \quad (28a)$$

where $\underline{F}^*$ and $\underline{g}$ are m-component column vectors and where a typical element a_{kj} of the A matrix is

$$a_{kj} = 0.5\, w(j)\, ((x(j)+1)/2)^{ck-1}. \quad (28b)$$

Equation (28) is solved for $\underline{g}$. Then, m points of F(t) are obtained from (26), with associated values of the independent variable, t, obtained from (25). For the best accuracy for the c.d.f. on several sample problems using Bellman's method, it is found that the value of the constant c should be unity and that the problem scale parameters should be adjusted so that the mean value of the sum (t) is approximately unity. (If necessary, rescaling t can be done following the solution of (28), in order to preserve original units of the independent variable.) It is found that the matrix A becomes progressively closer to being singular as m increases. For double-precision arithmetic on the Prime 9955, it is found that truncation error limits the maximum value of m to about 16. However, these 16 points of F(t) are calculated quite rapidly and accurately. For example, the RMS error for 3 convolutions of an exponential is about 0.000004, and the RMS error for 3 convolutions of the above exponential mixture is about 0.00027. Thus, when the Laplace transform of a distribution of interest is easily and accurately calculated, Bellman's method is the method of choice.

---

[1] The weights, w(j), and the points, x(j), for mth order gaussian quadrature are listed in Handbook of Mathematical Functions, AMS 55 (1966), on page 916, for values of m from 2 to 96.

14. Convolutions of a Two-Parameter Weibull Distribution

Analytic methods and/or Bellman's inverse Laplace transform are not well suited to obtain convolutions of certain types of probability distributions. Examples are: (a) a distribution with a threshold parameter and an upper truncation limit, and (b) a distribution whose Laplace transform is difficult to express or to evaluate accurately. One probability distribution of practical [1] interest which suffers from the last difficulty is the two-parameter Weibull function, whose c.d.f. is given as

$$F(x) = 1 - \exp(-(x/a)^{b}) , \qquad 0 \le x < \inf, \qquad (29)$$

with scale parameter a and shape parameter b.
Altho the transform can be expressed as an error function of s, the result is difficult to evaluate with accuracy sufficient for inversion via Bellman's method. Further, closed-form expressions for the n-fold convolution of (29) become quite complicated for n large. The closed-form expression for n = 2, taken from Ref 1c, for the special case in which the shape parameter, b, = 2, is

$$F_t(t) = 1 - \exp(-z^{2}) - \mathrm{sqrt}(\pi/2)\, z\, (N(z) - N(-z)), \qquad (30a)$$

where N(z) is the standard Normal integral with argument z, and with

$$z = t/a . \qquad (30b)$$

A general formula which approximates the n-fold convolution of a two-parameter Weibull distribution was derived by Leonard Johnson [2]. The LJ approximation is an Erlang distribution in the argument u, where

$$u = (pt/a)^{b} . \qquad (31)$$

The parameter p is chosen so that the mean of the approximating distribution matches its counterpart in the convolution distribution.

$$p = \mathrm{gamma}(n + 1/b)/\mathrm{gamma}(1 + 1/b)/n! , \qquad (32)$$

with complete gamma function gamma(argument).

---

[1] The two-parameter Weibull distribution has proved to be a good model for the life distribution of components or systems subject to fatigue failure. For this reason it is used extensively in the automotive industry. See Ref 1c.

[2] Johnson, L. GMR Reliability Manual, GMR-302,
General Motors Research Labs, 1960.

Thus, the LJ approximation for the n-fold convolution c.d.f. is

$$F_t(t) = 1 - \exp(-u) \sum_{i=0}^{n-1} u^i / i! \quad . \qquad (33)$$

From equations (31,32,33) it is seen that the LJ approximation is exact for n = 1. To see how the error of approximation grows with the order of convolution, consider the following numerical example. Let the scale parameter, a, = 6 and the shape parameter, b, = 2. To compare the LJ approximation with other methods, this problem is also solved using these other methods: (a) discrete numerical convolution, using 1028 points on a domain that comprises the 0th to the 99.97th percentiles, (b) Normal approximation, and (c) Monte-Carlo simulation with a sample size of 20,000 replications. The RMS error, over 20 equi-spaced points, for each of these methods is shown in Table 7. The error for the discrete-numerical (DN) convolution is shown for n = 2, since an analytic expression exists as a check, in this instance. Since this error is relatively quite small, the DN solution is used to evaluate the c.d.f. errors for other values of n. As expected, the Normal approximation decreases with n. By contrast, the LJ approx error increases with n. For n greater than or equal to 7, the Normal approximation has a smaller RMS error than the LJ approximation, and hence is preferred to LJ there. RMS errors of the Monte-Carlo (MC) method are relatively independent of convolution order. The values given here are the average produced by two random number streams.

TABLE 7
RMS ERRORS IN THE C.D.F. OF THE SUM OF N IDENTICAL
TWO-PARAMETER WEIBULL RV'S PRODUCED BY SEVERAL METHODS

Convol'n Order (N)	Method of Calculation DN	LJ	NA	MC [1]
1	0.0000	0.0000	0.0194	0.0013
2	0.0004	0.0028	0.0145	0.0024
3	0.0 [2]	0.0036	0.0104	0.0022
4	0.0	0.0041	0.0081	0.0026
5	0.0	0.0045	0.0067	0.0026
6	0.0	0.0049	0.0060	0.0028
7	0.0	0.0054	0.0054	0.0036

[1] Average value of the error over two random number streams.
[2] Value of the discrete numerical error is not evaluated for n > 2, but is considered relatively small versus other errors.

15. Run Time Comparisons

Whereas, the discrete numerical method is quite accurate, and is reasonably fast for n = 2, run time for this method increases as a power function of n - 2, with a power of about 1.25. Thus, for a constant density of 1024 points on the domain of the convolution c.d.f., an approximate run time (sec) is given by

$$T = 40\,(n-2)^{1.25} \quad . \tag{34}$$

For n = 2 only 20 points of the numerical convolution are evaluated. For this reason run time is a fraction of a second for n = 2, whereas for larger values of n, the maximum number of points (1024) are calculated for each of the convolutions except the last (nth). It is emphasized that run time for any method strongly depends upon the background activity of the (time-shared) computer. Equation (34) gives nearly the maximum time experienced. Minimum run times are approximately half of maximum. The form of equation (34) suggests that computational overhead, e.g. paging, increases faster than n does. When n is 3, the Monte-Carlo run time for 20,000 replications is about the same as the run time for the DN method. However, for higher-order convolutions, MC is faster. For example, for n = 4, DN requires 50% more time to execute than MC. For n = 5, DN requires 75% more time to run than MC, i.e., the ratio of run times is about 1.75. For n = 8, this ratio is 2.6. Thus, if one is satisfied with an RMS error less than 0.3%, Monte-Carlo would be the preferred of these two methods, for n > 3. Considering the errors of the LJ and NA methods, these are not very attractive unless execution time is a major consideration. If minimum run time is a primary consideration for this type of problem, a hybrid method might be used in which DN is used for n < 4, LJ used for 4 le n < 7, and NA used for n ge 7. The computer source program (INT.TEST) used in making the comparisons in Table 7 is found in Annex E.

16. Fourier Transform Method

As noted above (p. 3, pgf. 5), the product of an integral transform of each of two functions corresponds to the transform of the convolution of the functions. This theorem has already been exploited in connection with the Laplace transform. This paragraph is concerned with an application of this theorem using the Fourier transform. An important and practical Fourier transform method uses an algorithm for calculating the finite Fourier transform (or its inverse) due to Cooley and Tukey, and called the fast Fourier transform or FFT [1]. The speedy execution of the FFT makes practical the following method. Two density functions are each evaluated at a particular number of equi-spaced points on their domains. These data vectors are input to the FFT, which yields the complex-valued transforms. These transforms are multiplied (observing the rules of complex arithmetic) to obtain the transform of the convolution density. Finally, the inverse FFT is performed on this function to yield the required density. In the computer program for performing these operations, found in Annex E, a function f(x) is represented in complex form by a set of n points in which there are n/2 real components and n/2 imaginary components. Note that n must be an integer power of 2 for this purpose. These real and complex components are stored in adjacent storage locations in the n-element vector. Of course, the densities being convolved have only real components, so that all imaginary components of f(x) are assigned 0 value. Since the transform occurs in place, the transform of f(x),

---

[1] Bloomfield, P. Fourier Analysis of Time Series: An Introduction, John Wiley, New York, NY, c. 1976.

denoted as f*(w), is also stored in the n-element vector with real and imaginary components of the transform also located in adjacent positions. In general, the imaginary components of f*(w) are non-zero. The n/2 real frequency components of f*(w) are denoted by f*(w(k)) with k odd, and the n/2 imaginary frequency components are located in elements of the vector f*(k) with k even, (k = 1, 2, ..., n). In this formulation the transform and its inverse are duals related by equations (35) and (36):

$$f^*(w(k)) = \text{Sum over } j\ (1,n):\ \exp(-iw(k)(j-1))\ f(x(j))/n, \quad (35)$$

where w(k) is the kth complex frequency with

$$w(k) = 2\ pi\ (k-1)/n\ , \qquad k = 1, 2, \ldots, n,$$

and where i is the pure imaginary, sqrt(-1).

Then,

$$f(x(j)) = \text{Sum over } k\ (1,n):\ \exp(iw(k)(j-1))\ f^*(w(k)). \quad (36)$$

Because of the dual nature of f(x) and f*(w), the same routine that produces a transform can obtain an inverse transform merely by specifing which type of operation is wanted via "sign" = -1 for the Fourier transform, and by sign = 1 for an inverse transform. The computer code for this algorithm is found in Annex F.

17. A series of numerical tests were performed for accuracy and run time using the Fourier transform method. These are compared with Monte-Carlo tests using the same test functions. Probability densities used as test functions have the standardized Erlang and standardized Weibull forms. In both instances the scale parameter is unity, and the function is characterized by just a shape parameter. In the first numerical example with n Erlang densities being convolved, n-1 of these have been assigned a shape parameter of 2 and one is assigned shape parameter 3. RMS errors are shown in Table 8, for selected values of n, for both the Fourier transform (FFT) method and for a Monte-Carlo simulation with 20,000 replications. The RMS error is obtained over 16 equi-spaced points on the domain. The number of points (equivalently, real Fourier frequencies) used to represent the densities is also a parameter in these tests.

TABLE 8
RMS ERRORS IN THE C.D.F. OF THE SUM OF N RV'S FROM
TWO ERLANG DISTRIBUTIONS VIA FFT AND MONTE-CARLO METHODS

Convol'n Order (N)	FFT with # real frequencies 1024	2048	4096	Monte Carlo (20k reps)
2	0.0008	0.0004	0.0002	0.0022
3	0.0016	0.0008	0.0004	0.0019
4	0.0023	0.0012	0.0006	0.0007
5	0.0031	0.0016	0.0008	0.0010
10	0.0072	0.0036	0.0018	0.0008
20	0.0136	0.0092	0.0041	0.0011

Using the FFT method with 2048 real frequencies, the run time for any value of n varies from about 9 to 18 seconds. Run time for the FFT method seems to be dominated by the time to obtain the Fourier transforms and to obtain the inverse. Because relatively little time is spent in multiplying transforms, run time is essentially independent of the convolution order for the values shown. By contrast, it is seen that Monte-Carlo run time (T) increases nearly linearly with N:

$$T = 8(N - 2) + 12. \qquad (37)$$

(This approximation is quite similar to that given in equation (13) for evaluating the c.d.f. of the sum of N uniform random variables at 20 points via Monte-Carlo.) Run time for the FFT method does increase in a proportional manner with # of real Fourier frequencies. For example, when 1024 real frequencies are used run time is about 5 seconds. This time increases to 9 seconds for 2048 real frequencies and to about 20 seconds for 4096 real frequencies. Thus, in terms of run time, calculating the c.d.f. of the sum of 3 Erlang RVs is nearly the same using either Monte-Carlo, with 20,000 replications, or the FFT method, using 4096 real frequencies. It is noted that for a high-order convolution integral--say, > 10--a very large number of Fourier frequencies are required to make the accuracy of the FFT method competitive with Monte-Carlo. This point is illustrated by the results in Table 8. It is also demonstrated by another numerical example. Consider the case in which all the distributions being convolved are standardized exponential. The RMS errors for the FFT and Monte-Carlo methods for this case are shown in Table 9. Note that these results are substantially the same as those in Table 8.

TABLE 9
RMS ERRORS IN THE C.D.F. OF THE SUM OF N IDENTICAL EXPONENTIAL RV'S USING FFT AND MONTE-CARLO METHODS

Convol'n Order (N)	Numerical Method *	
	Fourier Transform	Monte Carlo
2	0.00025	0.0020
3	0.00045	0.0018
4	0.00064	0.0019
5	0.00082	0.0022
10	0.00178	0.0010

* The FFT method implemented here has 4096 real Fourier frequencies (8192 element array). Simulation sample is 20,000 replications. Monte-Carlo results shown are averages for two random streams.

18. A third numerical example was used to test the accuracy of the Fourier transform method. In this case a standardized Weibull density with shape parameter = 2 is convolved n times to yield the p.d.f. for the sum of n such Weibull RVs. For the particular case in which n is 2, the numerical error in the c.d.f. is found by comparing the exact result from equation (30) with the FFT approximation. The RMS error for this case is 0.00089, about three times that for the previous two examples. Thus, the numerical error of the Fourier transform method is rather sensitive to the form of

the distributions being convolved. Simpson's rule is used to calculate mean and SD from a numerical c.d.f. for the sum of two Weibull RVs from the distribution with shape parameter = 2. For the case of 4096 real Fourier frequencies, the error in the FFT mean is 0.186%. By contrast, the error in the mean value using the discrete numerical (DN) convolution with 1024 points is 0.09%. A comparable relationship exists in the RMS error in the c.d.f. for the DN versus the FFT method. RMS error in the convolution c.d.f. for this example is 0.00038, for DN, versus 0.00089 for FFT. By either measure of error, the DN method, applied on a set of 1024 points, incurrs less than half the error of the FFT method, applied on a set of 4096 points. If accuracy of results were tne sole criterion, discrete numerical convolution would certainly be preferred to the FFT method. However, for high-order convolutions, DN is computationally expensive relative to FFT. For example, 5 convolutions of a Weibull distribution using DN with this degree of discretization takes about 160 sec. (equation (34)). In a comparable run environment, FFT with 4096 real Fourier frequencies requires about 50 sec for the same problem. Thus, the FFT method executes this problem in one third the time required by the DN method, given the specified density of points. It is noted that the maximum number of real frequencies (4096) used with FFT in the above examples is the maximum permitted on our Prime computer. The computer system limit on the number of double-precision words allocated to a vector is less than 16,384. If the number of real frequencies were doubled, to 8192, the dynamic storage required for both real and imaginary frequency components would be 16,384.

19. Summary and Conclusions

This report has surveyed several methods for calculating probability distributions of sums of independent random variables. Formulas for tne c.d.f. of the sum have been derived for several cases. These cases include n random variables from: (a) a standard uniform distribution, (b) uniquely different uniform distributions, (c) an Erlang distribution, including the exponential as a special case, (d) different exponential distributions, (e) two-component exponential mixtures, and (f) a Weibull(2) distribution (two RVs only). The closed-form solutions were used to evaluate the accuracy of various numerical methods, including approximations.

20. The sum of n RVs from some distributions have a c.d.f. which rapidly approaches Normality with increasing n. Examples of this sort are the uniform distribution and distributions which appear Normal, such as gamma with large shape parameter. However, other distributional forms exhibit relatively slow convergence. These include exponential and exponential mixture distributions. Tne last is particularly slow in converging toward Normality. For this case, the sum of 15 RVs has a c.d.f. whose Normal approximation has a max absolute error of more than 0.08, which is intolerably large for most purposes. Generally, the c.d.f. errors of a Normal approximation are larger, for a given n, if the scale (or rate-) parameters of the component distributions exhibit a large range than if all distributions are identical.

21. It is difficult to make unqualified statements concerning the superiority of any one of the numerical methods. This situation is due in part to the diversity of user requirements for speed and accuracy and, in part, to the fact that some methods are parti-

cularly suited to just some classes of distributions. For example, Bellman's method requires that the Laplace transform of the probability density be easily and accurately calculated. The class of gamma distributions and of mixtures of gamma distributions are, therefore, well suited to this method. Because of superior run time and accuracy [1], Bellman's method is the method of choice for distributions in the gamma class, when certain conditions are met. These are: (a) the user must be satisfied with a sixteen-point characterization of the p.d.f. and of the c.d.f., and (b) available machine arithmetic can operate on a floating-point word with 48 bit mantissa and 7 bit (or more) exponent. (These requirements are met on the Prime 9955 minicomputer with double-precision arithmetic.) The last feature is necessary to avoid truncation error, which is critical to Bellman's method. In those instances where Bellman's method is inapplicable, discrete numerical convolution of the component distributions offers the greatest potential for accuracy, at a cost of run time. Where run time is an important consideration as well as accuracy, use of the Fourier transform method with the FFT algorithm is attractive, providing the order of convolution does not exceed about ten. (This statement presumes that the max vector dimension < 16,384.) Another advantage of the Fourier transform method is that it is quite flexible with regard to distributional forms that can be handled. Of course, the Normal approximation is preferred in those instances where the form of the component distributions assures rapid convergence toward Normality. For distributions on a bounded domain, such as the uniform, relatively small RMS errors in the c.d.f. by Normal approximation are incurred when the order of convolution is 5 or more. In cases where accuracy is not too stringent--say, an RMS error of 0.002--Monte-Carlo simulation [2] is the most flexible and resonably efficient method studied. A somewhat surprising finding is that Monte-Carlo is preferred, in many cases, to discrete numerical convolution when the tolerable RMS error is about 0.2% and when the number (n) of random variables in the sum is three or more. Monte-Carlo run time increases linearly with n, but the rate of increase is not as great as that for discrete numerical convolution. In comparing Monte-Carlo with FFT, it is noted that the RMS error for Monte-Carlo does not increase with n, as the FFT error does. When limited by computer storage to 4096 real frequency components, the FFT method becomes less accurate than Monte-Carlo for n greater than about ten. Also, time to code a given application for a Monte-Carlo simulation is generally the least of any method.

[1] Numerical error in the c.d.f. of convolutions of the gamma family have errors via Bellman's method of the order of 10**-5.

[2] A Monte-Carlo sample of 20,000 replications was used for nearly all numerical tests. This sample size is a practical value in view of these facts: (a) run time is proportional to sample size, and (b) RMS error is inversely proportional to square root of the sample. Halving the RMS error would increase run time by a factor of 4. For an RMS error in the c.d.f. of much less than 0.2%, the required Monte-Carlo run time would make this method non-competitive with others.

DISTRIBUTION

Copies		
1	HQDA WASH DC 20310	DAMO-ZA
1	HQDA WASH DC 20310	DALO-SMZ
	COMMANDER USAMC 5001 EISENHOWER AVE. ALEXANDRIA, VA 22333-0001	
1	ATTN:	AMCRE-IP
1		AMCPA-S
1		AMCDP
	DIRECTOR, US AMSAA ABERDEEN PG, MD 21005-5066	
1	ATTN:	AMXSY-DL
1		AMXSY-R
1		AMXSY-MP
	COMMANDER USA COMMUNICATIONS AND ELECTRONICS COMMAND FT MONMOUTH, NJ 07703-5304	
1	ATTN:	AMSEL-PL-SA
	COMMANDER CECOM (R&D) FT MONMOUTH, NJ 07703-5304	
1	ATTN:	AMSEL-SAD
	COMMANDER USAMICOM REDSTONE ARSENAL, AL 35809-5060	
1	ATTN:	AMSMI-DS
	COMMANDER USATACOM WARREN, MI 48090	
1	ATTN:	AMSTA-V
	OFFICE OF PROJECT MGR CANNON ARTY WPNS,DOVER NJ 07801-5001	
1	ATTN:	AMCPM-CAWS
	COMMANDER, US ARMY LOGISTICS CENTER FORT LEE, VA 23801	
1	ATTN:	ATCL-S

1 COMMANDER
DEFENSE LOGISTICS STUDIES
INFORMATION EXCHANGE
FORT LEE, VA 23801

COMMANDER
USA LOGISTICS EVAL AGENCY
NEW CUMBERLAND army DEPOT
NEW CUMBERLAND, PA 17070
1 ATTN: DAL-LEM

COMMANDER
US MRSA
LEXINGTON, KY 40511-5101
1 ATTN: AMXMD-ER

DIRECTOR, US ARMY
INVENTORY RESEARCH OFFICE
ROOM 800, CUSTOM HOUSE
2 ND & CHESNUT STREETS
PHILADELPHIA, PA 19106
1 ATTN: AMXMC-IRO

COMMANDER USATECOM
ABERDEEN PG, MD 21005-5055
1 ATTN: AMSTE-SY

12 DEFENCE TECHNICAL INFORMATION CENTER
CAMERON STATION
ALEXANDRIA, VA 22314

COMMANDER US ARDEC (D)
DOVER, NJ 07801-5001
1 ATTN: SMCAR- -LC (D)
1 -SC (D)
1 -SE (D)
1 -RAA (D)
1 -MSI (D)

COMMANDER US AMCCOM (R)
ROCK IS, IL 61299-6000
1 ATTN: AMSMC- -AS (R)
1 -IR (R)
1 -QA (R)
1 -MA (R)
1 -OP (R)
7 -SA (R)
1 -IMP-L (R)

DIRECTOR, AMCCOM
AMMO CENTER
SAVANNA, IL 61074
1 ATTN: SARAC-DO

COMMANDER
WATERVLIET ARSENAL
WATERVLIET, NY 12189-5000
1 ATTN: AMXMC-LCB-TL

COMMANDER
CHEMICAL R AND D CENTER
ABERDEEN PROVING GROUND
(EDGEWOOD AREA), MD 21010-5423
1 ATTN: AMSMC-CLJ-IA (A)

DIRECTOR US AMETA
ROCK IS, IL 61299-6000
1 ATTN: AMXOM-QA

DIRECTOR
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93940
1 ATTN: DEPT OF OPERATIONS ANAL.

1 DIRECTOR
ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD
ARLINGTON, VA 22209

DIRECTOR
USA TRASANA
WHITE SANDS MISSILE RANGE
WHITE SANDS, NM 88002-5502
1 ATTN: ATAA-SL

COMMANDER
USA COMBINED ARMS COMBAT
DEVELOPMENT ACTIVITY
FT LEAVENWORTH, KS 66207
1 ATTN: ATZL-CAM-M

COMPUTER SOURCE PROGRAMS

Source programs listed in Annexes A-F are written in SIMSCRIPT 2.5 for the PRIME minicomputer. However, the source code does not employ features peculiar to this computer. Each Annex contains a MAIN or executive program and several routines and functions. At the beginning of each program listing are found a functional description and an I/O list. All utility functions and routines are included among these listings. Inputs to MAIN programs are read interactively, with prompting messages displyed at the terminal. No external files are used. Since output is lengthy, it is necessary to set up a COMO file to display all of it and to obtain a permanent copy. The functions of the MAIN programs are summarized here:

RUN.NFOLD.U, in Annex A, obtains the probability density and cumulative probability distribution of the n-fold convolution of a standard uniform distribution. Approximations to the p.d.f. and c.d.f. of convolution based on a Normal probability function are calculated and printed for comparison.

RUN.NFOLD.GU, in Annex B, calculates and prints the p.d.f. and c.d.f. of the sum of a set of n uniform random variables drawn from distributions having a common threshold parameter but with different domains. Normal probability approximations to the p.d.f. and c.d.f. are calculated and printed for comparison with exact results. The maximum absolute error and the RMS error are calculated for the Normal approximation to the c.d.f. Optionally, a Monte-Carlo simulation can be performed and error statistics calculated and printed.

RUN.NFOLD.E, in Annex C, obtains the p.d.f. and c.d.f. of the sum of n exponential random variables. Two options are available: (a) all exponential random variables are from the same distribution, and (b) each exponential RV is from a uniquely different distribution. The exact c.d.f. is compared with a Normal approximation on a finite point set. Max abs and RMS errors are calculated and printed.

LP.INV, in Annex D, obtains the p.d.f. and the c.d.f. of the sum of n random variables from Erlang distributions and exponential mixture distributions. A numerical method based upon the Laplace transform is used to obtain approximate results. This method involves calculating the inverse transform via Bellman's method. A closed-form solution to the problem is used to calculate the error in Bellman's method and the errors of a Normal approximation and of a Monte-Carlo estimate of the c.d.f.

INT.TEST, in Annex E, tests a variety of methods for obtaining convolution integrals of a two-parameter Weibull distribution. The methods being compared are: (a) evaluation of an analytic expression, (b) Leonard Johnson's approximation based on the Erlang distribution, (c) discrete numerical convolution, and (d) Monte-Carlo simulation. The max absolute error and the RMS error, over a finite set of points, are calculated and printed for each numerical approximation.

TEST.CONVOLV, in Annex F, obtains convolutions of either standardized Erlang or Weibull distributions using a numerical method based on the finite Fourier transform. Comparisons with exact results and, optionally, with Monte-Carlo estimates are also given.

ANNEX A

SIMSCRIPT SOURCE PROGRAM: RUN.NFOLD.U

```
 1  PREAMBLE ''RUN.NFOLD.U
 2  NORMALLY MODE IS REAL
 3  DEFINE SNORM AS A REAL FUNCTION GIVEN 1 ARGUMENT
 4  DEFINE ERRFX AS A REAL FUNCTION GIVEN 1 ARGUMENT
 5  END ''PREAMBLE

 1  MAIN ''RUN.NFOLD.U
 2  ''
 3  ''Driver program to obtain the probability density and cum probability
 4  ''of the N-fold convolution of a standard, uniform dist.  PDFs and CDFs
 5  ''of the Normal dist having same mean and SD is printed for comparison.
 6  ''
 7      DEFINE I,N AS INTEGER VARIABLES
 8      DEFINE ANSWER AS A TEXT VARIABLE
 9  'LO'SKIP 1 LINE
10      PRINT 4 LINES THUS
This program calculates and prints the p.d.f. and c.d.f of the N-fold
convolution of a standard, (0-1) uniform probability distribution.  User
inputs are the integer N and the upper probability limit (PMAX) to terminate.

15      PRINT 1 LINE THUS
 INPUT THE VALUE OF N.
17      READ N
18      PRINT 1 LINE THUS
 INPUT THE (MAX) VALUE OF THE CDF TO TERMINATE CALCULATIONS.
20      READ PMAX
21      LET AVG=N/2.0
22      LET VAR=N/12.0
23      LET COND=1.0/SQRT.F(2.0*PI.C*VAR) ''FOR NORMAL DENSITY COEF
24      LET STDV=SQRT.F(VAR)
25      LET LIM=AVG + 3.0*STDV
26      LET LIM=MIN.F(REAL.F(N), LIM)
27      LET DELT=LIM/20.0
28      LET LINES.V=9999
29      SKIP 2 LINES
30      PRINT 6 LINES WITH N
31      THUS
PROBABILITY DISTRIBUTION OF A **-FOLD CONVOLUTION OF A STD UNIFORM DISTRIBUTION


-------------------------------------------------------------------------
Indep     N-fold Convolution        Normal Prob Distrib      Difference
Variable  p.d.f.     c.d.f.         p.d.f.     c.d.f.        c.d.f.
-------------------------------------------------------------------------
38      LET MAE=0.0
39      LET RMS=0.0
40      FOR I=1 TO 20 DO
41          LET T=I*DELT
42          CALL NFOLD.U (N,T) YIELDING PDF,CDF
43          LET ARG=(T-AVG)/STDV
44          LET NPDF=COND*EXP.F(-0.5*ARG**2)
45          LET NCDF=SNORM(ARG)
46          LET DIFF=CDF-NCDF
```

```
   47          LET MAE=MAX.F(MAE,ABS.F(DIFF))
   48          ADD DIFF**2 TO RMS
   49          PRINT 1 LINE WITH T, PDF, CDF, NPDF, NCDF, DIFF
   50          THUS
**.***    *.******   *.******       *.******   *.******      *.******
   52          IF CDF GE PMAX
   53              GO TO L1
   54          OTHERWISE
   55      LOOP ''OVER I
   56  'L1'PRINT 2 LINES THUS
```

```
   59      LET RMS=SQRT.F(0.05*RMS)
   60      PRINT 2 LINES WITH MAE,RMS
   61      THUS
Max abs error in Normal approximation of c.d.f. _ *.******
RMS error in Normal approximation of c.d.f. _____ *.******
   64      PRINT 1 LINE THUS
    DO YOU HAVE OTHER VALUES OF N?  (YES OR NO).
   66      READ ANSWER
   67      IF SUBSTR.F(ANSWER,1,1) = "Y"
   68          GO TO L0
   69      OTHERWISE
   70      STOP
   71  END ''MAIN

    1  ROUTINE NFOLD.U GIVEN N, T YIELDING PDF, CDF
    2  ''
    3  ''Routine calculates the probability density function (PDF) and the cum-
    4  ''ulative distribution function (CDF) of the N-fold convolution of a
    5  ''standard uniform (0,1) probability dist.  Real-valued argument is T.
    6  ''
    7  ''With the following notation for the CDF  argument t:  F(n,t), with
    8  ''the combination of n things taken i at a time denoted as C(n,i),
    9  ''and with the unit step function at x denoted by u(t-x),
   10  ''
   11  ''                                    i                       n
   12  ''     F(n,t) = Sum (i=0 to n): (-1)  C(n,i) u(t-i) (t-i) /n!
   13  ''
   14      DEFINE I,N AS INTEGER VARIABLES
   15      IF T LE 0.0
   16          LET PDF=0.0
   17          LET CDF=0.0
   18          RETURN
   19      OTHERWISE
   20      IF T GE REAL.F(N)
   21          LET PDF=0.0
   22          LET CDF=1.0
   23          RETURN
   24      OTHERWISE
   25      LET COMBIN=N
   26      LET FACT=1.0
   27      FOR I=2 TO N, LET FACT=FACT*I ''FOR N FACTORIAL
   28      LET PDF=T**(N-1)
   29      LET CDF=PDF*T
   30      LET SIGN= -1.0
   31      FOR I=1 TO N DO
```

```
32          LET TI=I
33          IF T LE TI
34              GO TO L1
35          OTHERWISE
36          LET TERM=SIGN*COMBIN*(T-TI)**(N-1)
37          ADD TERM TO PDF
38          ADD TERM*(T-TI) TO CDF
39          LET SIGN= -SIGN
40          LET COMBIN=COMBIN*(N-I)/(I+1)
41      LOOP ''OVER I
42  'L1'LET PDF=PDF*N/FACT
43      LET CDF=CDF/FACT
44      RETURN
45  END ''NFOLD.U
```

```
 1  FUNCTION SNORM(Z)
 2  ''
 3  ''ROUTINE CALCULATES THE STANDARD NORMAL PROBABILITY INTEGRAL.
 4  ''REF:  APPROXIMATION OBTAINED FROM AMS 55, ABRAMOWITZ AND STEGUN.
 5  ''
 6      IF ABS.F(Z) > 7.0
 7          GO TO L2
 8      OTHERWISE
 9      LET P=0.5+SIGN.F(Z)*0.5*ERRFX(ABS.F(Z)/SQRT.F(2.0))
10      RETURN WITH P
11  'L2'LET P=0.5+SIGN.F(Z)*0.5
12      RETURN WITH P
13  END ''OF SNORM
```

```
 1  FUNCTION ERRFX(X)
 2  ''
 3  ''ROUTINE CALCULATES THE ERROR FUNCTION.  THIS FUNCTION IS CALLED BY
 4  ''SNORM(Z).
 5  ''REFERENCE:  AMS 55, 'HANDBOOK OF MATHEMATICAL FUNCTIONS', NAT. BUREAU
 6  ''OF STANDARDS, NOV. 1970, (P. 299).
 7  ''
 8      LET S=SIGN.F(X)
 9      LET X=ABS.F(X)
10      IF X<0.00000000001
11          RETURN WITH 0.0
12      OTHERWISE
13      IF X>10.0
14          RETURN WITH S
15      OTHERWISE
16      LET T=1.0/(1.0+0.3275911*X)
17      LET SUM=1.06140543*T
18      LET SUM=(SUM-1.45315203)*T
19      LET SUM=(SUM+1.42141374)*T
20      LET SUM=(SUM-0.284496736)*T
21      LET SUM=(SUM+0.254829592)*T
22      RETURN WITH S*(1.0-SUM*EXP.F(-X*X))
23  END ''OF FUNCTION ERRFX
```

ANNEX B

SIMSCRIPT SOURCE PROGRAM: RUN.NFOLD.GU

```
 1  PREAMBLE ''RUN.NFOLD.GU
 2  NORMALLY MODE IS REAL
 3  DEFINE SNORM AS A REAL FUNCTION GIVEN 1 ARGUMENT
 4  DEFINE ERRFX AS A REAL FUNCTION GIVEN 1 ARGUMENT
 5  DEFINE ICOMBIN AS AN INTEGER FUNCTION GIVEN 2 ARGUMENTS
 6  END ''PREAMBLE

 1  MAIN ''RUN.NFOLD.GU
 2  ''
 3  ''Driver program to obtain the probability density and cum prob of the
 4  ''N-fold convolution of a set of uniform distributions having a common
 5  ''lower domain limit (CL) and having different upper domain limits.
 6  ''PDFs & CDFs of Normal dist having same avg and s.d. are also printed.
 7  ''
 8      DEFINE FLAGM,I,J,K,M,N,NCELLS,NREPS,SEED AS INTEGER VARIABLES
 9      DEFINE ANSWER AS A TEXT VARIABLE
10      DEFINE NCV,HISTV AS INTEGER, 1-DIMENSIONAL ARRAYS
11      DEFINE AV,XV,CDFV AS REAL, 1-DIMENSIONAL ARRAYS
12      DEFINE SM AS A REAL, 2-DIMENSIONAL ARRAY
13      LET LINES.V=9999
14      LET RT12=SQRT.F(12.0)
15      LET NCELLS=10
16      RESERVE CDFV(*) AS NCELLS
17  'LO'SKIP 1 LINE
18      PRINT 7 LINES THUS
```

This program calculates and prints the p.d.f. and c.d.f of the N-fold convolution of a set of N uniform probability distributions, each of which is defined on its own, possibly, unique interval--CL to upper limit. Inputs are integer N, upper c.d.f. value to terminate calculation (PMAX), common lower argument value (CL), and N upper limits of the uniform ranges. Max value of N permitted by the program is 20. Optionally, a Monte-Carlo histogram can be obtained.

```
26      SKIP 2 LINES
27      PRINT 1 LINE THUS
 INPUT THE VALUE OF N.
29      READ N
30      LET N=MIN.F(N,20)
31      RESERVE AV(*) AS N
32      RESERVE NCV(*) AS N
33      LET NCV(1)=N
34      LET NCV(N)=1
35      FOR K=2 TO N-1, LET NCV(K)=ICOMBIN(N,K)
36  ''
37  ''RESERVE MATRIX OF N-TUPLE SUMS OF AV(*).
38  ''
39      RESERVE SM(*,*) AS N BY *
40      FOR I=1 TO N, RESERVE SM(I,*) AS NCV(I)
41      PRINT 1 LINE THUS
 INPUT THE (MAX) VALUE OF THE CONVOLUTION CDF TO TERMINATE CALCULATIONS.
43      READ PMAX
44      PRINT 1 LINE THUS
 INPUT THE COMMON VALUE OF THE ARGUMENT LOWER LIMIT (OR THRESHOLD).
46      READ CL
```

```
      LET AVG=0.0
      LET VAR=0.0
      LET ACON=1.0
      FOR I=1 TO N DO
          PRINT 1 LINE WITH I
          THUS
 INPUT THE UPPER LIMIT OF THE ARGUMENT RANGE FOR UNIFORM VARIABLE # **.
          READ AU
          IF AU LE CL
              PRINT 1 LINE WITH AU,CL
              THUS
INPUT ERROR.  UPPER ARG LIM  ..... IS LESS THAN LOWER  .....
              STOP
          OTHERWISE
          LET AV(I)=AU-CL
          LET ACON=ACON*AV(I)
          ADD 0.5*AV(I) TO AVG
          ADD AV(I)**2/12.0 TO VAR
      LOOP ''OVER (I) UNIFORM COMPONENTS
      LET COND=1.0/SQRT.F(2.0*PI.C*VAR) ''FOR NORMAL DENSITY COEF
      LET STDV=SQRT.F(VAR)
      LET LIM=AVG + 3.0*STDV
      LET LIM=MIN.F(LIM, 2.0*AVG)
      LET DELT=LIM/20.0
      PRINT 1 LINE THUS
 DO YOU WANT A MONTE-CARLO SIMULATION?  (YES OR NO).
      READ ANSWER
      IF SUBSTR.F(ANSWER,1,1) = "Y"
          LET FLAGM=1
          PRINT 1 LINE THUS
 INPUT THE INDEX (1 TO 9) OF THE RANDOM NUMBER SEED.
          READ SEED
          PRINT 1 LINE THUS
 INPUT THE NUMBER OF REPLICATIONS WANTED.
          READ NREPS
          PRINT 1 LINE WITH NREPS
          THUS
A Monte-Carlo simulation of ***** replications has begun.
          LET NCELLS=10
          LET DELX=2.0*DELT
          RESERVE XV(*) AS NCELLS
          RESERVE HISTV(*) AS NCELLS
          FOR K=1 TO NCELLS, LET HISTV(K)=0
          LET AVGX=0.0
          LET VARX=0.0
          FOR K=1 TO NCELLS, LET XV(K)=N*CL+K*DELX
 ''
 ''SIMULATE FOR NREPS REPLICATIONS.
 ''
          FOR I=1 TO NREPS DO
              LET SUM=0.0
              FOR J=1 TO N DO
                  ADD UNIFORM.F(CL,CL+AV(J),SEED) TO SUM
              LOOP ''OVER J
              ADD SUM TO AVGX
              ADD SUM**2 TO VARX
 ''
```

```
  104  ''DETERMINE CELL OF HISTOGRAM AND ADD TO CELL COUNT.
  105  ''
  106              FOR K=1 TO NCELLS DO
  107                  IF SUM LE XV(K)
  108                      ADD 1 TO HISTV(K)
  109                      GO TO K2
  110                  OTHERWISE
  111              LOOP ''OVER K
  112  'K2'    LOOP ''OVER (I) REPLICATIONS
  113          LET AVGX=AVGX/NREPS
  114          LET VARX=VARX/NREPS-AVGX**2
  115          PRINT 1 LINE THUS
Monte-Carlo simulation has been completed.
  117      OTHERWISE
  118          LET FLAGM=0
  119      ALWAYS
  120  ''
  121  ''FILL RAGGED TABLE SM WITH N-TUPLE SUMS OF THE ELEMENTS OF AV.
  122  ''
  123      CALL STUPLES (N, AV(*), SM(*,*))
  124      SKIP 2 LINES
  125      PRINT 7 LINES WITH N
  126      THUS
                                                                          (1)
PROB DISTRIBUTION OF A **-FOLD CONVOLUTION OF A SET OF UNIFORM FUNCTIONS


Indep     N-fold Convolution        Normal Prob Distrib      Difference
Variable  p.d.f.     c.d.f.         p.d.f.     c.d.f.        c.d.f.

  134      LET RMS.DIFF=0.0
  135      LET MAE.DIFF=0.0
  136      LET K=0 ''TO COUNT PAIRS
  137      FOR I=1 TO 20 DO
  138          LET T=I*DELT
  139          CALL NFOLD.GU (N, ACON, SM(*,*), T) YIELDING PDF, CDF
  140          IF MOD.F(I,2)=0
  141              ADD 1 TO K
  142              LET CDFV(K)=CDF
  143          ALWAYS
  144          LET ARG=(T-AVG)/STDV
  145          LET NPDF=COND*EXP.F(-0.5*ARG**2)
  146          LET NCDF=SNORM(ARG)
  147          LET DIFF=NCDF-CDF
  148          ADD DIFF**2 TO RMS.DIFF
  149          LET MAE.DIFF=MAX.F(MAE.DIFF,ABS.F(DIFF))
  150          PRINT 1 LINE WITH T+N*CL, PDF, CDF, NPDF, NCDF, DIFF
  151          THUS
****.**** *.******   *.******       *.******   *.******      *.******
  153          IF CDF GE PMAX
  154              GO TO L1
  155          OTHERWISE
  156      LOOP ''OVER I
  157  'L1'PRINT 2 LINES THUS

  160      LET RMS.DIFF=SQRT.F(0.05*RMS.DIFF)
```

```
  161       PRINT 2 LINES WITH MAE.DIFF,RMS.DIFF
  162       THUS
Max abs error  between c.d.f. and the Normal c.d.f. approx   *.******
RMS difference between c.d.f. and the Normal c.d.f. approx   *.******
  165       PRINT 3 LINES WITH AVG+N*CL,STDV
  166       THUS
    Mean value of the convolution is *****.***** with std dev *****.*****.
    (1)  Mean and Std Dev of each of the Uniform distributions:
         Component       Mean Value     Std Deviation
  170           FOR I=1 TO N DO
  171               PRINT 1 LINE WITH I, CL+0.5*AV(I), AV(I)/RT12
  172               THUS
             **          *****.*****    *****.*****
  174           LOOP ''OVER (I) UNIFORM DISTRIBUTIONS
  175           SKIP 2 LINES
  176           IF FLAGM NE 1
  177               GO TO K3
  178           OTHERWISE
  179           PRINT 7 LINES WITH N,NREPS
  180           THUS
MONTE-CARLO SAMPLE DISTRIBUTION OF THE SUM OF ** UNIFORM RANDOM VARIABLES

NUMBER OF REPLICATIONS:  ******

-------------------------------------------------------------------------
Indep      Histo       Sample       Sample       Diff Versus
Variable   Frequency   p.d.f.       c.d.f.       analytic c.d.f.
-------------------------------------------------------------------------
  188           LET XCDF=0.0
  189           LET RMS.DIFF=0.0
  190           LET M=0
  191           FOR K=1 TO NCELLS, ADD HISTV(K) TO M
  192           FOR K=1 TO NCELLS DO
  193               LET XPDF=HISTV(K)/M
  194               LET XCDF=XCDF+XPDF
  195               LET DIFF=XCDF-CDFV(K)
  196               ADD DIFF**2 TO RMS.DIFF
  197               PRINT 1 LINE WITH XV(K),HISTV(K),XPDF,XCDF,DIFF
  198               THUS
****.****  ******      *.*****      *.*****       *.******
  200           LOOP ''OVER (K) HISTO CELLS
  201           PRINT 2 LINES THUS
-------------------------------------------------------------------------

  204           LET RMS.DIFF=SQRT.F(RMS.DIFF/REAL.F(NCELLS))
  205           PRINT 1 LINE WITH RMS.DIFF
  206           THUS
RMS difference: sample c.d.f. - analytic c.d.f.   *.******
  208           LET SDX=SQRT.F(VARX)
  209           LET SEX=SDX/SQRT.F(REAL.F(NREPS))
  210           PRINT 3 LINES WITH AVGX,SDX,AVGX-1.96*SEX,AVGX+1.96*SEX
  211           THUS
Sample Average Value  *****.*****    Sample Standard Deviation *****.*****
95 percent confidence interval in mean:  *****.*****, *****.*****

  215  'K3'RELEASE NCV(*)
  216       RELEASE AV(*)
  217       RELEASE SM(*,*)
```

```
218      PRINT 1 LINE THUS
  DO YOU HAVE OTHER PROBLEMS OF THIS KIND?  (YES OR NO).
220      READ ANSWER
221      IF SUBSTR.F(ANSWER,1,1) = "Y"
222          GO TO L0
223      OTHERWISE
224      STOP
225  END ''MAIN

  1  ROUTINE NFOLD.GU GIVEN N, ACON, SM, T YIELDING PDF, CDF
  2  ''
  3  ''Routine calculates the prob density function (PDF) and the cum-
  4  ''ulative dist function (CDF) of the N-fold convolution of a set of
  5  ''N  unique uniform distributions.  The range of the i th distribution
  6  ''is (0, a(i)).  The product, over N, of the a(i) is the argument ACON.
  7  ''All n-tuple sums of elements a(i) are entered in the ragged table SM,
  8  ''where the k th row and j th column element is the j th k-tuple sum.
  9  ''E.g., the first row of SM contains a(j).  Real-valued argument is T.
 10  ''Num of combinations of N objects taken K at a time is DIM.F(SM(K,*)).
 11  ''With the following notation for the CDF, with argument t:  F(n,t),
 12  ''with the j th k-tuple sum for the n th convolution denoted by
 13  ''
 14  ''     S  (n) ,
 15  ''      kj
 16  ''
 17  ''and with the unit step function at x denoted by u(t-x),
 18  ''
 19  ''                                n
 20  ''     F(n,t) = (1/ACON/n!)(t    + Sum over k=1 to n and j=1 to C(n,k):
 21  ''
 22  ''                k                                 n
 23  ''           (-1) u(t - S  (n))(t - S  (n)) ),
 24  ''                      jk           jk
 25  ''
 26  ''where C(n,k) is the # combinations of n things taken k at a time.
 27  ''
 28       DEFINE I,J,K,N AS INTEGER VARIABLES
 29       DEFINE SM AS A REAL, 2-DIMENSIONAL ARRAY
 30       IF T LE 0.0
 31           LET PDF=0.0
 32           LET CDF=0.0
 33           RETURN
 34       OTHERWISE
 35       IF T GE SM(N,1)
 36           LET PDF=0.0
 37           LET CDF=1.0
 38           RETURN
 39       OTHERWISE
 40       LET FACT=1.0
 41       FOR I=2 TO N, LET FACT=FACT*I ''FOR N FACTORIAL
 42       LET PDF=T**(N-1)
 43       LET CDF=PDF*T
 44       LET SIGN= 1.0
 45       FOR K=1 TO N DO
 46           LET SIGN= -SIGN
 47           FOR J=1 TO DIM.F(SM(K,*)) DO
 48               IF T > SM(K,J)
```

```
49                  LET TARG=T-SM(K,J)
50                  LET INCR=SIGN*TARG**N
51                  ADD INCR/TARG TO PDF
52                  ADD INCR TO CDF
53              ALWAYS
54          LOOP ''OVER (J) COLUMNS
55      LOOP ''OVER (K) ROWS
56      LET PDF=PDF*N/FACT/ACON
57      LET CDF=CDF/FACT/ACON
58      RETURN
59  END ''NFOLD.GU

 1  FUNCTION ICOMBIN (N, K)
 2  ''
 3  ''INTEGER-VALUED # OF COMBINATIONS OF N OBJECTS TAKEN K AT A TIME.
 4  ''
 5      DEFINE C,I,K,N AS INTEGER VARIABLES
 6      IF K = 0
 7          RETURN WITH  1
 8      OTHERWISE
 9      LET C=1
10      FOR I=1 TO K DO
11          LET C=C*(N-I+1)/I
12      LOOP ''OVER I
13      RETURN WITH C
14  END ''FUNCTION ICOMBIN

 1  ROUTINE STUPLES (N, AV, SM)
 2  ''
 3  ''Routine fills the elements of a ragged table, SM, having N rows.
 4  ''The k,j element of this table consists of the j th k-tuple sum of the
 5  ''elements of the vector AV.  Routine is called by NFOLD.GU.
 6  ''
 7      DEFINE I,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,
 8      I17,I18,I19,J,K,N AS INTEGER VARIABLES
 9      DEFINE JV AS AN INTEGER, 1-DIMENSIONAL ARRAY
10      DEFINE AV AS A REAL, 1-DIMENSIONAL ARRAY
11      DEFINE SM AS A REAL, 2-DIMENSIONAL ARRAY
12      IF N > 20
13          PRINT 1 LINE WITH N
14          THUS
NUMBER OF VARIABLES (= **) EXCEEDS THE CAPACITY OF 20 IN ROUTINE STUPLES.
16          STOP
17      OTHERWISE
18      RESERVE JV(*) AS N ''LOCALLY
19      LET SM(N,1)=0.0
20      FOR J=1 TO N, ADD AV(J) TO SM(N,1)
21      FOR I1=1 TO N DO
22        LET S1=AV(I1)
23        LET SM(1,I1)=S1
24        IF N < 3
25          GO TO L1
26        OTHERWISE ''gen 2 tuples
27        FOR I2=I1+1 TO N DO
28          ADD 1 TO JV(2)
29          LET S2=S1+AV(I2)
30          LET SM(2,JV(2))=S2
```

```
31          IF N < 4
32            GO TO L2
33          OTHERWISE ''gen 3 tuples
34          FOR I3=I2+1 TO N DO
35            ADD 1 TO JV(3)
36            LET S3=S2+AV(I3)
37            LET SM(3,JV(3))=S3
38            IF N < 5
39              GO TO L3
40            OTHERWISE ''gen 4 tuples
41            FOR I4=I3+1 TO N DO
42              ADD 1 TO JV(4)
43              LET S4=S3+AV(I4)
44              LET SM(4,JV(4))=S4
45              IF N < 6
46                GO TO L4
47              OTHERWISE ''gen 5 tuples
48              FOR I5=I4+1 TO N DO
49                ADD 1 TO JV(5)
50                LET S5=S4+AV(I5)
51                LET SM(5,JV(5))=S5
52                IF N < 7
53                  GO TO L5
54                OTHERWISE ''gen 6 tuples
55                FOR I6=I5+1 TO N DO
56                  ADD 1 TO JV(6)
57                  LET S6=S5+AV(I6)
58                  LET SM(6,JV(6))=S6
59                  IF N < 8
60                    GO TO L6
61                  OTHERWISE ''gen 7 tuples
62                  FOR I7=I6+1 TO N DO
63                    ADD 1 TO JV(7)
64                    LET S7=S6+AV(I7)
65                    LET SM(7,JV(7))=S7
66                    IF N < 9
67                      GO TO L7
68                    OTHERWISE ''gen 8 tuples
69                    FOR I8=I7+1 TO N DO
70                      ADD 1 TO JV(8)
71                      LET S8=S7+AV(I8)
72                      LET SM(8,JV(8))=S8
73                      IF N < 10
74                        GO TO L8
75                      OTHERWISE ''gen 9 tuples
76                      FOR I9=I8+1 TO N DO
77                        ADD 1 TO JV(9)
78                        LET S9=S8+AV(I9)
79                        LET SM(9,JV(9))=S9
80                        IF N < 11
81                          GO TO L9
82                        OTHERWISE ''gen 10 tuples
83                        FOR I10=I9+1 TO N DO
84                          ADD 1 TO JV(10)
85                          LET S10=S9+AV(I10)
86                          LET SM(10,JV(10))=S10
87                          IF N < 12
```

```
88                          GO TO L10
89                        OTHERWISE ''gen 11 tuples
90                        FOR I11=I10+1 TO N DO
91                          ADD 1 TO JV(11)
92                          LET S11=S10+AV(I11)
93                          LET SM(11,JV(11))=S11
94                          IF N < 13
95                            GO TO L11
96                          OTHERWISE ''gen 12 tuples
97                          FOR I12=I11+1 TO N DO
98                            ADD 1 TO JV(12)
99                            LET S12=S11+AV(I12)
100                           LET SM(12,JV(12))=S12
101                           IF N < 14
102                             GO TO L12
103                           OTHERWISE ''gen 13 tuples
104                           FOR I13=I12+1 TO N DO
105                             ADD 1 TO JV(13)
106                             LET S13=S12+AV(I13)
107                             LET SM(13,JV(13))=S13
108                             IF N < 15
109                               GO TO L13
110                             OTHERWISE ''gen 14 tuples
111                             FOR I14=I13+1 TO N DO
112                               ADD 1 TO JV(14)
113                               LET S14=S13+AV(I14)
114                               LET SM(14,JV(14))=S14
115                               IF N < 16
116                                 GO TO L14
117                               OTHERWISE ''gen 15 tuples
118                               FOR I15=I14+1 TO N DO
119                                 ADD 1 TO JV(15)
120                                 LET S15=S14+AV(I15)
121                                 LET SM(15,JV(15))=S15
122                                 IF N < 17
123                                   GO TO L15
124                                 OTHERWISE ''gen 16 tuples
125                                 FOR I16=I15+1 TO N DO
126                                   ADD 1 TO JV(16)
127                                   LET S16=S15+AV(I16)
128                                   LET SM(16,JV(16))=S16
129                                   IF N < 18
130                                     GO TO L16
131                                   OTHERWISE ''gen 17 tuples
132                                   FOR I17=I16+1 TO N DO
133                                     ADD 1 TO JV(17)
134                                     LET S17=S16+AV(I17)
135                                     LET SM(17,JV(17))=S17
136                                     IF N < 19
137                                       GO TO L17
138                                     OTHERWISE ''gen 18 tuples
139                                     FOR I18=I17+1 TO N DO
140                                       ADD 1 TO JV(18)
141                                       LET S18=S17+AV(I18)
142                                       LET SM(18,JV(18))=S18
143                                       IF N < 20
144                                         GO TO L18
```

```
145                                          OTHERWISE ''gen 19 tuples
146                                          FOR I19=I18+1 TO N DO
147                                            ADD 1 TO JV(19)
148                                            LET S19=S18+AV(I19)
149                                            LET SM(19,JV(19))=S19
150  'L19'                                   LOOP ''OVER I19
151  'L18'                                 LOOP ''OVER I18
152  'L17'                               LOOP ''OVER I17
153  'L16'                             LOOP ''OVER I16
154  'L15'                           LOOP ''OVER I15
155  'L14'                         LOOP ''OVER I14
156  'L13'                       LOOP ''OVER I13
157  'L12'                     LOOP ''OVER I12
158  'L11'                   LOOP ''OVER I11
159  'L10'                 LOOP ''OVER I10
160  'L9'                LOOP ''OVER I9
161  'L8'              LOOP ''OVER I8
162  'L7'            LOOP ''OVER I7
163  'L6'          LOOP ''OVER I6
164  'L5'        LOOP ''OVER I5
165  'L4'      LOOP ''OVER I4
166  'L3'    LOOP ''OVER I3
167  'L2'  LOOP ''OVER I2
168  'L1'LOOP ''OVER I1
169      RELEASE JV(*)
170      RETURN
171  END ''STUPLES
```

```
  1  FUNCTION SNORM(Z)
  2  ''
  3  ''ROUTINE CALCULATES THE STANDARD NORMAL PROBABILITY INTEGRAL.
  4  ''REF:  APPROXIMATION OBTAINED FROM AMS 55, ABRAMOWITZ AND STEGUN.
  5  ''
  6      IF ABS.F(Z) > 7.0
  7          GO TO L2
  8      OTHERWISE
  9      LET P=0.5+SIGN.F(Z)*0.5*ERRFX(ABS.F(Z)/SQRT.F(2.0))
 10      RETURN WITH P
 11  'L2'LET P=0.5+SIGN.F(Z)*0.5
 12      RETURN WITH P
 13  END ''OF SNORM
```

ANNEX C

SIMSCRIPT SOURCE PROGRAM: RUN.NFOLD.E

```
 1  PREAMBLE ''RUN.NFOLD.E
 2  NORMALLY MODE IS REAL
 3  DEFINE SNORM AS A REAL FUNCTION GIVEN 1 ARGUMENT
 4  DEFINE ERRFX AS A REAL FUNCTION GIVEN 1 ARGUMENT
 5  END ''PREAMBLE

 1  MAIN ''RUN.NFOLD.E
 2  ''
 3  ''Driver program to obtain the probability density and cum prob of the
 4  ''N-fold convolution of a set of N exponential dist's.  Two options
 5  ''are available:  (a) all exponential dist's in the set to be convolved
 6  ''are identical,  (b) all of the exponential dist's are unique.  PDFs
 7  ''and CDFs of a Normal dist with the same mean and variance is printed
 8  ''for comparison with the convolution.
 9  ''
10      DEFINE I,J,K,L,M,N,NCELLS AS INTEGER VARIABLES
11      DEFINE ANSWER AS A TEXT VARIABLE
12      DEFINE INDEX AS AN INTEGER, 1-DIMENSIONAL ARRAY
13      DEFINE AV,LAV AS REAL, 1-DIMENSIONAL ARRAYS
14      DEFINE MAT AS A REAL, 2-DIMENSIONAL ARRAY
15      LET NCELLS=20
16  'LO'SKIP 1 LINE
17      PRINT 7 LINES THUS
```

This program calculates and prints the p.d.f. and c.d.f of the N-fold convolution of a set of N exponential distributions. Two progam options exist for these distributions: (a) all distributions have the same mean value, and (b) all distributions have unique (or different) means. User inputs are the number (N) of distributions to be convolved, the upper c.d.f. limit to terminate calculations, and the mean values of these exponential distributions.

```
25      PRINT 1 LINE THUS
 INPUT THE VALUE OF N.
27      READ N
28      PRINT 1 LINE THUS
 INPUT THE (MAX) VALUE OF THE CDF TO TERMINATE CALCULATIONS.
30      READ PMAX
31      RESERVE AV(*),LAV(*) AS N
32      RESERVE INDEX(*) AS N-1
33      RESERVE MAT(*,*) AS N BY N
34      PRINT 1 LINE THUS
 DO ALL EXPONENTIAL DISTRIBUTIONS HAVE THE SAME PARAMETER?  (Y OR N).
36      READ ANSWER
37      IF SUBSTR.F(ANSWER,1,1) = "Y"
38          PRINT 1 LINE THUS
 INPUT THE COMMON MEAN VALUE.
40          READ AVG
41          FOR I=1 TO N, LET LAV(I)=1.0/AVG
42          LET VAR=N*AVG**2
43          LET AVG=N*AVG
44      OTHERWISE
45          LET AVG=0.0
46          LET VAR=0.0
47          LET LAMBDA=1.0
```

```
48          IF N > 15
49              PRINT 1 LINE THUS
 MAX # OF CONVOLUTIONS IS LIMITED TO 15 FOR THIS OPTION.  CHOOSE THE 15.
51              RELEASE AV(*),LAV(*),INDEX(*)
52              RELEASE MAT(*,*)
53              LET N=15
54              RESERVE AV(*),LAV(*),INDEX(*) AS N
55              RESERVE MAT(*,*) AS N BY N
56          ALWAYS
57          FOR I=1 TO N DO
58              PRINT 1 LINE WITH I
59              THUS
 INPUT THE MEAN VALUE OF THE ** TH EXPONENTIAL DISTRIBUTION.
61              READ THETA
62              ADD THETA TO AVG
63              ADD THETA**2 TO VAR
64              LET LAV(I)=1.0/THETA
65              LET LAMBDA=LAMBDA*LAV(I)
66          LOOP ''OVER (I) EXPONENTIAL DISTRIBUTIONS
67  ''
68  ''OBTAIN THE COEFFICIENTS AV(*) FOR THE CONVOLUTION DENSITY.
69  ''
70          IF N = 2
71              LET AV(1)=LAMBDA/(LAV(2)-LAV(1))
72              LET AV(2)= -AV(1)
73              GO TO L2
74          OTHERWISE
75          FOR J=1 TO N, LET MAT(1,J)=1.0
76          FOR I=2 TO N-1, FOR J=1 TO N, LET MAT(I,J)=0.0
77          FOR J=1 TO N DO
78              LET PNJ=1.0
79              FOR K=1 TO N DO
80                  IF K NE J
81                      LET PNJ=PNJ*LAV(K)
82                  ALWAYS
83              LOOP ''OVER K
84              LET MAT(N,J)=PNJ
85          LOOP ''OVER (J) COLUMNS
86          CALL NTUPLES (N, LAV(*), MAT(*,*)) ''FILL EL'MTS OF MAT(*,*)
87  ''
88  ''OBTAIN THE INVERSE OF MAT(*,*).
89  ''
90          CALL MAT.INVERSE (N, MAT(*,*))
91          FOR I=1 TO N, LET AV(I)=LAMBDA*MAT(I,N)
92      ALWAYS
93  'L2'LET COND=1.0/SQRT.F(2.0*PI.C*VAR)
94      LET STDV=SQRT.F(VAR)
95      LET LIM=AVG + 4.0*STDV
96      LET DELT=LIM/NCELLS
97      LET LINES.V=9999
98      SKIP 2 LINES
99      PRINT 7 LINES WITH N
100     THUS
```

(1)

PROB DISTRIBUTION OF THE CONVOLUTION OF A SET OF ** EXPONENTIAL DISTRIBUTIONS

```
Indep      N-fold Convolution          Normal Prob Distrib       Difference
Variable   p.d.f.      c.d.f.          p.d.f.      c.d.f.        c.d.f.
-------------------------------------------------------------------------
  108       LET MAE=0.0
  109       LET RMS=0.0
  110       FOR I=1 TO NCELLS DO
  111           LET T=I*DELT
  112           CALL NFOLD.E (N, AV(*), LAV(*), T) YIELDING PDF,CDF
  113           LET ARG=(T-AVG)/STDV
  114           LET NPDF=COND*EXP.F(-0.5*ARG**2)
  115           LET NCDF=SNORM(ARG)
  116           LET DIFF=CDF-NCDF
  117           LET MAE=MAX.F(MAE,ABS.F(DIFF))
  118           ADD DIFF**2 TO RMS
  119           PRINT 1 LINE WITH T, PDF, CDF, NPDF, NCDF, DIFF
  120           THUS
**.***     *.******   *.******       *.******   *.******      *.******
  122           IF CDF GE PMAX
  123               GO TO L1
  124           OTHERWISE
  125       LOOP ''OVER I
  126   'L1'PRINT 2 LINES THUS
-------------------------------------------------------------------------

  129       LET RMS=SQRT.F(RMS/REAL.F(NCELLS))
  130       PRINT 2 LINES WITH MAE,RMS
  131       THUS
Max abs error in a Normal approximation of sum c.d.f. _ *.******
RMS error of a Normal approximation of the sum c.d.f. _ *.******
  134       PRINT 3 LINES WITH AVG, STDV
  135       THUS
(1) Mean Value of the Convolution  .............  Std Dev  .............
    Mean Values of each of the exponential distributions
    ---------------------------------------------------
  139       FOR I=1 TO N DO
  140           PRINT 1 LINE WITH I, 1.0/LAV(I)
  141           THUS
    Number  **  Mean  .............
  143       LOOP ''OVER I
  144       SKIP 2 LINES
  145       RELEASE LAV(*)
  146       RELEASE AV(*)
  147       RELEASE MAT(*,*)
  148       PRINT 1 LINE THUS
    DO YOU HAVE SIMILAR PROBLEMS TO SOLVE?  (YES OR NO).
  150       READ ANSWER
  151       IF SUBSTR.F(ANSWER,1,1) = "Y"
  152           GO TO L0
  153       OTHERWISE
  154       STOP
  155   END ''MAIN
```

```
ROUTINE NFOLD.E GIVEN N, AV, LAV, T YIELDING PDF, CDF
''
''Routine calculates the probability density function (PDF) and the cum
''distribution function (CDF) of the N-fold convolution of a set of
''exponential dist's.  Two options are provided:  (a) each of the N
''dist's has the same mean, and (b) each dist has a unique mean.
''If the mean values of the exponential dist's are all unique, the
''convolution can be expressed as a weighted sum of the exponential
''dist's.  These weights are passed to the routine in the vector AV(*).
''Rate parameters of the N expon dists are given in the vector LAV(*).
''The real-valued argument of the convolution is T.
''
''If each of the dist's has the same rate parameter, a, (option (a)),
''the p.d.f. and c.d.f. of the n-fold convolution are given,
''respectively, by these Erlang functions:
''
''                      n-1
''    f(n,t) = a (at)   exp(-at)/(n-1)!                    t > 0.
''
''                                              i
''    F(n,t) = 1 - exp(-at) Sum(i=0 to n-1): z / i!
''
    DEFINE I,N AS INTEGER VARIABLES
    DEFINE AV,LAV AS REAL, 1-DIMENSIONAL ARRAYS
    IF T LE 0.0
        LET PDF=0.0
        LET CDF=0.0
        RETURN
    OTHERWISE
    IF LAV(1)=LAV(2) ''ALL RATE PARMS ASSUMED EQUAL
        LET Z=LAV(1)*T
        LET EXPZ=EXP.F(-Z)
        LET FACT=1.0
        LET ZI=1.0
        LET SUM=1.0
        FOR I=1 TO N-1 DO
            LET FACT=FACT*I
            LET ZI=Z*ZI
            ADD ZI/FACT TO SUM
        LOOP ''OVER I
        LET PDF=LAV(1)*ZI/FACT*EXPZ
        LET CDF=1.0-EXPZ*SUM
        RETURN
    OTHERWISE
    LET PDF=0.0
    LET CDF=0.0
    FOR I=1 TO N DO
        LET EXPZ=EXP.F(-LAV(I)*T)
        ADD AV(I)*EXPZ TO PDF
        ADD AV(I)/LAV(I)*(1.0-EXPZ) TO CDF
    LOOP ''OVER I
    RETURN
END ''NFOLD.E
```

```
 1  ROUTINE FOR MAT.INVERSE (N, AM)
 2  ''
 3  ''ROUTINE TO OBTAIN THE INVERSE OF THE N BY N MATRIX AM VIA THE
 4  ''COMPACT FORM OF THE GAUSS-JORDAN METHOD.  INV IS RETURNED IN AM.
 5  ''
 6      DEFINE I, J, K, N AS INTEGER VARIABLES
 7      DEFINE AM AS A REAL, 2-DIMENSIONAL ARRAY
 8      FOR I=1 TO N DO
 9          LET P=AM(I,I)
10          IF P=0.0
11              PRINT 2 LINES WITH I THUS
 ERROR IN ROUTINE MAT.INVERSE.  THE ** TH DIAGONAL ELEMENT IS ZERO.
 THE MATRIX CANNOT BE INVERTED.
14              STOP
15          OTHERWISE
16          LET AM(I,I)=1.0
17          FOR J=1 TO N DO
18              LET AM(I,J)=AM(I,J)/P
19          LOOP ''OVER J
20          FOR J=1 TO N DO ''THE SECOND J-LOOP
21              IF J=I
22                  GO TO EOJ ''END OF J-LOOP
23              OTHERWISE
24              LET P=AM(J,I)
25              LET AM(J,I)=0.0
26              FOR K=1 TO N DO
27                  SUBTRACT P*AM(I,K) FROM AM(J,K)
28              LOOP ''OVER K
29  'EOJ'   LOOP ''OVER J
30      LOOP ''OVER I
31      RETURN
32  END ''ROUTINE MAT.INVERSE

 1  ROUTINE NTUPLES (N, LAV, MAT)
 2  ''
 3  ''Routine fills N-2 row elements in the MAT(*,*) which are contributed
 4  ''by n-tuples associated with variable LAV(*).  Routine is called by
 5  ''RUN.NFOLD.E.
 6      DEFINE I,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,J,K,N AS INTEGER
        VARIABLES
 7      DEFINE INDEX AS AN INTEGER, 1-DIMENSIONAL ARRAY
 8      DEFINE LAV AS A REAL, 1-DIMENSIONAL ARRAY
 9      DEFINE MAT AS A REAL, 2-DIMENSIONAL ARRAY
10      IF N > 15
11          PRINT 1 LINE WITH N
12          THUS
INPUT ERROR TO ROUTINE NTUPLES.  NUMBER OF CONVOLUTIONS, **, IS EXCESSIVE.
14          STOP
15      OTHERWISE
16      RESERVE INDEX(*) AS N-1 ''LOCALLY
17      FOR J=1 TO N DO
18      LET K=0
19      FOR I=1 TO N DO
20          IF I NE J
21              ADD 1 TO K
22              LET INDEX(K)=I
23          ALWAYS
```

```
24      LOOP ''OVER (I) PARAMETERS
25      IF N < 3
26          RELEASE INDEX(*)
27          RETURN
28      OTHERWISE
29      FOR I1=1 TO N-1 DO
30          LET LA1=LAV(INDEX(I1))
31          ADD LA1 TO MAT(2,J)
32          IF N < 4
33              GO TO L1
34          OTHERWISE ''gen 2 tuples
35          FOR I2=I1+1 TO N-1 DO
36              LET LA2=LA1*LAV(INDEX(I2))
37              ADD LA2 TO MAT(3,J)
38              IF N < 5
39                  GO TO L2
40              OTHERWISE ''gen 3 tuples
41              FOR I3=I2+1 TO N-1 DO
42                  LET LA3=LA2*LAV(INDEX(I3))
43                  ADD LA3 TO MAT(4,J)
44                  IF N < 6
45                      GO TO L3
46                  OTHERWISE ''gen 4 tuples
47                  FOR I4=I3+1 TO N-1 DO
48                      LET LA4=LA3*LAV(INDEX(I4))
49                      ADD LA4 TO MAT(5,J)
50                      IF N < 7
51                          GO TO L4
52                      OTHERWISE ''gen 5 tuples
53                      FOR I5=I4+1 TO N-1 DO
54                          LET LA5=LA4*LAV(INDEX(I5))
55                          ADD LA5 TO MAT(6,J)
56                          IF N < 8
57                              GO TO L5
58                          OTHERWISE ''gen 6 tuples
59                          FOR I6=I5+1 TO N-1 DO
60                              LET LA6=LA5*LAV(INDEX(I6))
61                              ADD LA6 TO MAT(7,J)
62                              IF N < 9
63                                  GO TO L6
64                              OTHERWISE ''gen 7 tuples
65                              FOR I7=I6+1 TO N-1 DO
66                                  LET LA7=LA6*LAV(INDEX(I7))
67                                  ADD LA7 TO MAT(8,J)
68                                  IF N < 10
69                                      GO TO L7
70                                  OTHERWISE ''gen 8 tuples
71                                  FOR I8=I7+1 TO N-1 DO
72                                      LET LA8=LA7*LAV(INDEX(I8))
73                                      ADD LA8 TO MAT(9,J)
74                                      IF N < 11
75                                          GO TO L8
76                                      OTHERWISE ''gen 9 tuples
77                                      FOR I9=I8+1 TO N-1 DO
78                                          LET LA9=LA8*LAV(INDEX(I9))
79                                          ADD LA9 TO MAT(10,J)
```

```
                                                    IF N < 12
                                                        GO TO L9
                                                    OTHERWISE ''gen 10 tuples
                                                    FOR I10=I9+1 TO N-1 DO
                                                        LET LA10=LA9*LAV(INDEX(I10))
                                                        ADD LA10 TO MAT(11,J)
                                                        IF N < 13
                                                            GO TO L10
                                                        OTHERWISE ''gen 11 tuples
                                                        FOR I11=I10+1 TO N-1 DO
                                                        LET LA11=LA10*LAV(INDEX(I11))
                                                        ADD LA11 TO MAT(12,J)
                                                        IF N < 14
                                                            GO TO L11
                                                        OTHERWISE ''gen 12 tuples
                                                        FOR I12=I11+1 TO N-1 DO
                                                           LET LA12=LA11*LAV(INDEX(I12))
                                                           ADD LA12 TO MAT(13,J)
                                                        LOOP ''OVER I12
                                                        LOOP ''OVER I11
                                                    LOOP ''OVER I10
                                                LOOP ''OVER I9
                                            LOOP ''OVER I8
                                        LOOP ''OVER I7
                                    LOOP ''OVER I6
                                LOOP ''OVER I5
                            LOOP ''OVER I4
                        LOOP ''OVER I3
                    LOOP ''OVER I2
'L1'LOOP ''OVER I1
    LOOP ''OVER (J) COLUMNS
    RELEASE INDEX(*)
END ''ROUTINE NTUPLES
```

ANNEX D

SIMSCRIPT SOURCE PROGRAM: LP.INV

```
 1  PREAMBLE ''LP.INV
 2  NORMALLY MODE IS REAL
 3  DEFINE A1,A2 AS REAL VARIABLES
 4  DEFINE SNORM AS A REAL FUNCTION GIVEN 1 ARGUMENT
 5  DEFINE ERRFX AS A REAL FUNCTION GIVEN 1 ARGUMENT
 6  DEFINE LTRNS.FUN AS A REAL FUNCTION GIVEN 3 ARGUMENTS
 7  END ''PREAMBLE

 1  MAIN ''LP.INV
 2  ''
 3  ''Obtains the inverse Laplace transform at a set of discrete points via
 4  ''Bellman's method.  This method approximates an integral by gaussian
 5  ''quadrature.  The quadrature formula leads to a matrix eq'n whose
 6  ''sol'n defines the inv in terms of the transform evaluated at M points.
 7  ''An example of this method is provided on pp 14 thru 18 of "Numerical
 8  ''Methods in Renewal Theory," (AD 828276), Feb 1968.
 9  ''
10      DEFINE FLAGE,FLAGM,FLAGMX,I,J,K,L,M,N,NREPS,SEED AS INTEGER
        VARIABLES
11      DEFINE ANSWER,FILNAM,TITLE,DIST.NAME AS TEXT VARIABLES
12      DEFINE IPVT,HISTV AS INTEGER, 1-DIMENSIONAL ARRAYS
13      DEFINE DET,TV AS REAL, 1-DIMENSIONAL ARRAYS
14      RESERVE DET(*) AS 2
15      DEFINE DSTARV,DV,GSTARV,GV,LAV,WV,XV,CDFV AS REAL, 1-DIMENSIONAL
        ARRAYS
16      DEFINE AM AS A REAL, 2-DIMENSIONAL ARRAY
17      LET M=16 ''TERMS IN THE GAUSSIAN QUADRATURE
18      RESERVE CDFV(*) AS M
19      LET FILNAM = "GAUSS.Q16.DATA"
20  ''  LET DIST.NAME = "Gamma(3)"
21      LET DIST.NAME = "Expon Mix"
22      LET FLAGMX=1
23      LET K=1
24      RESERVE TV(*),HISTV(*) AS M
25      LET CON.AVG=K   ''CONSTANT RELATING AVG TO RATE PARM
26      RESERVE WV(*), XV(*), IPVT(*) AS M
27      RESERVE DSTARV(*),DV(*),GSTARV(*),GV(*) AS M
28      RESERVE AM(*,*) AS M BY M
29  ''
30  ''READ THE QUADRATURE POINTS AND WEIGHTS FROM THE FILE:  FILNAM.
31  ''
32      LET EOF.V=1
33      LET LINES.V=9999
34      OPEN UNIT 4 FOR INPUT,
35      OLD,
36      FILE NAME IS FILNAM
37      RECORD SIZE IS 120
38      USE UNIT 4 FOR INPUT
39      READ TITLE USING UNIT 4
40      PRINT 2 LINES WITH FILNAM,TITLE
41      THUS
Data file  ***************  is read for ****************************************
```

```
      FOR I=1 TO M DO
          READ XV(I),WV(I) USING UNIT 4
      LOOP ''OVER (I) QUADRATURE POINTS
      CLOSE UNIT 4
      USE UNIT 5 FOR INPUT
      FOR I=1 TO M, LET TV(I)=LOG.E.F(2.0/(XV(M-I+1)+1.0))
      PRINT 7 LINES WITH DIST.NAME
      THUS
This program calculates and prints the c.d.f. of an N-fold convolution of
a set of N ********** distributions with different mean values.  This c.d.f.
is obtained by numerically inverting the Laplace transform of this function at
a discrete set of points on the real line.  Required inputs are:  N and the
mean values of each of these distributions.  Mean values of these dist-
ributions should be scaled so that the sum of the means is near 1.
Optionally, a comparative Monte-Carlo simulation may be performed.
 'LO'SKIP 2 LINES
      PRINT 1 LINE THUS
 INPUT THE VALUE OF N.
      READ N
      RESERVE LAV(*) AS N
      LET AVG=0.0
      LET VAR=0.0
      LET FLAGE=0
 ''   IF FLAGMX NE 1
 ''       GO TO L3
 ''   OTHERWISE
      IF N LE 3
          LET FLAGE=1
      ALWAYS
      PRINT 1 LINE THUS
 INPUT THE PROPORTION OF THE 1ST EXPONENTIAL COMPONENT.
      READ A1
      IF A1=1.0
          LET FLAGMX=0
          GO TO L3
      OTHERWISE
      PRINT 1 LINE THUS
 INPUT THE MEAN VALUE OF THE 1ST EXPONENTIAL COMPONENT.
      READ THETA
      LET LA1=1.0/THETA
      PRINT 1 LINE THUS
 INPUT THE MEAN VALUE OF THE 2ND EXPONENTIAL COMPONENT.
      READ THETA
      LET LA2=1.0/THETA
      LET A2=1.0-A1
      LET LAV(1)=LA1
      LET LAV(2)=LA2
      LET AVG1=A1/LA1+A2/LA2
      LET VAR1=2.0*(A1/LA1**2+A2/LA2**2) - AVG1**2
      LET AVG=N*AVG1
      LET VAR=N*VAR1
      GO TO L4
'L3'LET FLAGE=0
      FOR I=1 TO N DO
          PRINT 1 LINE WITH I
          THUS
 INPUT THE MEAN VALUE OF THE ** TH RANDOM VARIABLE.
```

```
            READ THETA
            ADD THETA TO AVG
            LET LAV(I)=CON.AVG/THETA
            ADD CON.AVG*(CON.AVG+1.0)/LAV(I)**2 - THETA**2 TO VAR
            IF I > 1
                IF LAV(I)=LAV(I-1)
                    ADD 1 TO FLAGE
                ALWAYS
            ALWAYS
        LOOP ''OVER (I) COMPONENTS
        LET LA1=LAV(1)
        IF FLAGE = N-1
            LET FLAGE=1
        OTHERWISE
            LET FLAGE=0 ''NOT ALL DISTS ARE THE SAME
        ALWAYS
    'L4'LET COND=1.0/SQRT.F(2.0*PI.C*VAR)
        LET STDV=SQRT.F(VAR)
    ''  PRINT 1 LINE THUS
    ''  INPUT THE SCALE FACTOR (1 GE GAMMA LE 1.2) IN LAPLACE TRANSFORM.
    ''  READ GAMMA
        LET GAMMA=1.0
        PRINT 1 LINE THUS
      DO YOU WANT TO PERFORM A MONTE-CARLO SIMULATION?  (YES OR NO).
        READ ANSWER
        IF SUBSTR.F(ANSWER,1,1) = "Y"
            LET FLAGM=1
            PRINT 1 LINE THUS
      INPUT THE INDEX (1 THRU 9) OF THE RANDOM # SEED.
            READ SEED
            PRINT 1 LINE THUS
      INPUT THE NUMBER OF REPLICATIONS WANTED.
            READ NREPS
            PRINT 1 LINE WITH NREPS
            THUS
A Monte-Carlo simulation of ***** replications has begun.
            FOR L=1 TO M, LET HISTV(L)=0
            LET AVGT=0.0
            LET VART=0.0
    ''
    ''SIMULATE FOR NREPS REPLICATIONS.
    ''
            FOR I=1 TO NREPS DO
                LET SUM=0.0
                FOR J=1 TO N DO
    ''              ADD ERLANG.F(CON.AVG/LAV(J),K,SEED) TO SUM
                    IF UNIFORM.F(0.0,1.0,SEED) LE A1
                        ADD EXPONENTIAL.F(1.0/LA1,SEED) TO SUM
                    OTHERWISE
                        ADD EXPONENTIAL.F(1.0/LA2,SEED) TO SUM
                    ALWAYS
                LOOP ''OVER J
                ADD SUM TO AVGT
                ADD SUM**2 TO VART
                FOR L=1 TO M DO
                    IF SUM LE TV(L)
                        ADD 1 TO HISTV(L)
```

```
158                     GO TO K2
159                 OTHERWISE
160                     LOOP ''OVER (L) CELLS
161  'K2'       LOOP ''OVER (I) REPLICATIONS
162             LET AVGT=AVGT/NREPS
163             LET VART=VART/NREPS-AVGT**2
164             PRINT 1 LINE THUS
Monte-Carlo simulation has been completed.
166         LET CDF.MC=0.0
167         FOR L=1 TO M DO
168             LET PDF=HISTV(L)/NREPS
169             ADD PDF TO CDF.MC
170             LET CDFV(L)=CDF.MC
171         LOOP ''OVER (L) CELLS
172         OTHERWISE
173             LET FLAGM=0
174         ALWAYS
175  ''
176  ''GET T'FORMS OF PDF AND  CDF AT M POINTS.   PLACE IN DSTARV & GSTARV.
177  ''
178         FOR I=1 TO M DO
179             LET S=GAMMA*I
180             LET GSTARV(I)=LTRNS.FUN (N, LAV(*), S)
181             LET DSTARV(I)=GSTARV(I)*S
182         LOOP ''OVER (I) POINTS ON THE REAL LINE IN THE S-PLANE
183  ''
184  ''FILL THE ELEMENTS OF AM(*,*).
185  ''
186         FOR I=1 TO M DO
187             FOR J=1 TO M DO
188                 LET E=GAMMA*I - 1.0
189                 LET AM(I,J)=0.5*WV(J)*(0.5*(XV(J)+1.0))**E
190             LOOP ''OVER (J) COLUMNS
191         LOOP ''OVER (I) ROWS
192  ''
193  ''SOLVE THE EQUATION:  AM * GV = GSTARV.
194  ''
195         LET J=0
196         CALL SGEFA (AM(*,*), IPVT(*), J)
197         IF J NE 0
198             PRINT 1 LINE WITH J
199             THUS
TROUBLE FRACTORING THE MATRIX AM IN PROGRAM LP.INV.  J =     *.
201             STOP
202         OTHERWISE
203         CALL SGEDI (AM(*,*), IPVT(*), DET(*), 11)
204         IF DET(2) < -82
205             SKIP 2 LINES
206             PRINT 2 LINES WITH DET(1), DET(2)
207             THUS
DETERMINANT OF MATRIX AM = *.**** X 10 EXPON (****), WHICH IS ALMOST
SINGULAR.  ACCURACY OF INV(AM) IS QUESTIONABLE.
210         ALWAYS
211  ''     CALL MAT.INVERSE (M, AM(*,*))
212         CALL MAT.VEC.MPY (AM(*,*), GSTARV(*), M) YIELDING GV(*)
213         CALL MAT.VEC.MPY (AM(*,*), DSTARV(*), M) YIELDING DV(*)
214  ''
```

```
  215  ''PRINT OUTPUT C.D.F.
  216  ''
  217       SKIP 2 LINES
  218       PRINT 7 LINES WITH N,DIST.NAME
  219       THUS
                                                                        (1)
PROB DISTRIBUTION OF THE CONVOLUTION OF A SET OF ** ********** DIST'S


Indep     N-fold Convolution     Norm Prob Distrib      Difference
Variable  p.d.f.     c.d.f.      p.d.f.     c.d.f.      c.d.f.

  227       LET MAE.DIFF=0.0
  228       LET RMS.DIFF=0.0
  229       LET MAE.MC=0.0
  230       LET RMS.MC=0.0
  231       LET MAE.NI=0.0
  232       LET RMS.NI=0.0
  233       FOR I BACK FROM M TO 1 DO
  234            SKIP 1 LINE
  235            LET T=TV(M-I+1)
  236            LET PDF=DV(I)
  237            LET CDF=GV(I)
  238            LET NPDF=COND*EXP.F(-0.5*((T-AVG)/STDV)**2)
  239            LET NCDF=SNORM((T-AVG)/STDV)
  240            LET DIFF=CDF-NCDF
  241            LET MAE.DIFF=MAX.F(MAE.DIFF,ABS.F(DIFF))
  242            ADD DIFF**2 TO RMS.DIFF
  243            PRINT 1 LINE WITH T,PDF,CDF,NPDF,NCDF,DIFF
  244            THUS
***.****  *.******   *.******   *.******   *.******   *.******
  246            IF FLAGE=1 ''ALL RATE PARMS ARE THE SAME
  247  ''             CALL NFOLD.U GIVEN N, 0.5*LAV(1)*T YIELDING EPDF,ECDF
  248                IF A1 =1.0
  249                    CALL ERLANG (K*N, LA1, T) YIELDING EPDF,ECDF
  250                OTHERWISE
  251                    CALL NFOLD.MIXE (N, A1, LA1, LA2, T) YIELDING EPDF,ECDF
  252                ALWAYS
  253                LET RESID=CDF-ECDF
  254                LET MAE.NI=MAX.F(MAE.NI,ABS.F(RESID))
  255                ADD RESID**2 TO RMS.NI
  256  'L1'          PRINT 1 LINE WITH EPDF,ECDF,CDF-ECDF
  257                THUS
Exact fun *.******   *.******   Dif rel to exact cdf  *.******
  259            OTHERWISE
  260                LET ECDF=CDF
  261            ALWAYS
  262            LET DIFF.MC=CDFV(M-I+1) - ECDF
  263            LET MAE.MC=MAX.F(MAE.MC,ABS.F(DIFF.MC))
  264            ADD DIFF.MC**2 TO RMS.MC
  265       LOOP ''OVER (I) CDF POINTS
  266       PRINT 2 LINES THUS

  269       LET RMS.DIFF=SQRT.F(RMS.DIFF/REAL.F(M))
  270       LET RMS.NI=SQRT.F(RMS.NI/REAL.F(M))
  271       LET RMS.MC=SQRT.F(RMS.MC/REAL.F(M))
```

```
  272       PRINT 2 LINES WITH M,MAE.DIFF,M,RMS.DIFF
  273       THUS
Max abs error in Normal approx (over **) to c.d.f.     *.******
RMS difference in c.d.f. and Normal approx (over **)   *.******
  276       PRINT 2 LINES WITH MAE.NI,RMS.NI
  277       THUS
Max abs error in num inverse est of the c.d.f.         *.******
RMS error in the num inverse est of the c.d.f.         *.******
  280       IF FLAGM=1
  281           PRINT 2 LINES WITH MAE.MC,RMS.MC
  282           THUS
Max abs error in Monte-Carlo estimate of the c.d.f.    *.******
RMS error of the Monte-Carlo estimate of the c.d.f.    *.******
  285       ALWAYS
  286       PRINT 3 LINES WITH AVG,STDV,DIST.NAME
  287       THUS
     Mean of Convolution Distribution  ****.****   Std Dev  ****.****
(1)  Mean of each of the ********** distributions:
     ------------------------------------------
  291       FOR I=1 TO N DO
  292           IF FLAGMX=1
  293               LET AVGXI=A1/LAV(1)+A2/LAV(2)
  294           OTHERWISE
  295               LET AVGXI=CON.AVG/LAV(I)
  296           ALWAYS
  297           PRINT 1 LINE WITH I,AVGXI
  298           THUS
     Number  **  Mean  ****.****
  300       LOOP ''OVER (I) COMPONENT COMPONENTS
  301       SKIP 2 LINES
  302       IF FLAGM NE 1
  303           GO TO L2
  304       OTHERWISE
  305       PRINT 7 LINES WITH N,DIST.NAME,NREPS
  306       THUS
SAMPLE PROB DIST OF THE SUM OF A SET OF ** ********** RANDOM VARIABLES

Monte-Carlo Sample  *****

-------------------------------------------------------------------------
Indep       Histo       Sample       Sample
Variable    Frequency   p.d.f.       c.d.f.
-------------------------------------------------------------------------
  314       LET XCDF=0.0
  315       FOR I=1 TO M DO
  316           LET XPDF=HISTV(I)/NREPS
  317           ADD XPDF TO XCDF
  318           PRINT 1 LINE WITH TV(I),HISTV(I),XPDF,XCDF
  319           THUS
****.****   *****      *.*****      *.*****
  321       LOOP ''OVER (I) HISTO CELLS
  322       PRINT 2 LINES THUS
-------------------------------------------------------------------------

  325       LET SDT=SQRT.F(VART)
  326       LET SET=SDT/SQRT.F(REAL.F(NREPS))
  327       PRINT 3 LINES WITH AVGT,SDT,AVGT-1.96*SET,AVGT+1.96*SET
  328       THUS
```

```
Sample Average Value  ****.****  Sample Standard Deviation  ****.****
95 percent confidence interval in mean:  ****.****, ****.****

  332  'L2'PRINT 1 LINE THUS
    DO YOU HAVE SIMILAR PROBLEMS TO SOLVE?  (YES OR NO).
  334      READ ANSWER
  335      IF SUBSTR.F(ANSWER,1,1) = "Y"
  336          RELEASE LAV(*)
  337          GO TO L0
  338      OTHERWISE
  339      STOP
  340  END ''LP.INV

    1  ROUTINE FOR MAT.VEC.MPY (AM, BV, NELMTS) YIELDING CV
    2  ''
    3  ''ROUTINE TO MULTIPLY THE SQUARE MATRIX AM , OF NELMTS BY NELMTS,
    4  ''BY THE VECTOR BV (NELMTS BY 1), YIELDING THE VECTOR CV (NELMTS BY 1).
    5  ''
    6      DEFINE I, J, K, NELMTS AS INTEGER VARIABLES
    7      DEFINE BV, CV AS REAL, 1-DIMENSIONAL ARRAYS
    8      DEFINE AM AS A REAL, 2-DIMENSIONAL ARRAY
    9      RESERVE CV(*) AS NELMTS
   10      FOR I=1 TO NELMTS DO
   11          LET CV(I)=0.0
   12          FOR K=1 TO NELMTS DO
   13              ADD AM(I,K)*BV(K) TO CV(I)
   14          LOOP ''OVER K
   15      LOOP ''OVER I
   16      RETURN
   17      END ''ROUTINE MAT.VEC.MPY

    1  FUNCTION LTRNS.FUN (N, LAV, S)
    2  ''
    3  ''Obtains Laplace transform of a convolution of N prob dist functions
    4  ''having rate parameters LAV(*).  Complex argument (S) is evaluated
    5  ''only on the real line.  The inv t'form of this function is the c.d.f.
    6  ''This function must be particularized for the desired form of the prob
    7  ''functions.  The form used here is indicated by the comment statements.
    8  ''
    9      DEFINE I,N AS INTEGER VARIABLES
   10      DEFINE LAV AS A REAL, 1-DIMENSIONAL ARRAY
   11      LET F=1.0/S
   12  ''
   13  ''CODE FOR UNIFORM WITH MEAN = 1/LAV(i).
   14  ''
   15  ''  FOR I=1 TO N, LET F=F/S*(1.0-EXP.F(-2.0*S/LAV(I)))
   16  ''
   17  ''CODE FOR EXPONENTIAL.
   18  ''
   19  ''  FOR I=1 TO N, LET F=F*LAV(I)/(S + LAV(I))
   20  ''
   21  ''CODE FOR GAMMA(2).
   22  ''
   23  ''  FOR I=1 TO N, LET F=F*LAV(I)**2/(S + LAV(I))**2
   24  ''
   25  ''CODE FOR GAMMA(3).
   26  ''
```

```
''  FOR I=1 TO N, LET F=F*LAV(I)**3/(S + LAV(I))**3
''
''CODE FOR EXPONENTIAL MIX.
''
    FOR I=1 TO N DO
        IF A1 = 1.0
            LET F=F*LAV(1)/(S+LAV(1))
        OTHERWISE
            LET F=F*(A1*LAV(1)/(S+LAV(1))+A2*LAV(2)/(S+LAV(2)))
        ALWAYS
    LOOP ''OVER I
    RETURN WITH F
END ''FUNCTION LTRNS.FUN

ROUTINE SGEFA ( A, IPVT, INFO)
''
''FACTORS THE MATRIX A(*,*) INTO UPPER (U) AND STRICTLY LOWER (L)
''TRIANGULAR MATRICES SUCH THAT A(*,*) = U(*,*)L(*,*).  ROUTINE
''IS INTENDED FOR USE WITH OTHER ROUTINES OF THE LINEAR OPERATIONS
''PACKAGE--LINPACK.  THIS VERSION IS A CONVERSION OF THE FORTRAN
''ROUTINE WRITTEN BY CLEVE MOLER, U. OF N.M. AND ARGONNE NAT LAB.
''
''ARGUMENTS:
''NAME       MODE       ENTRY VALUE                RETURN VALUE
''
''A          REAL(N, N) SQUARE MATRIX.     UPPER TRIANGULAR MATRIX AND
''                                         MULTIPLIERS WHICH WERE USED TO
''                                         TO OBTAIN IT.  ARE STORED IN L.
''N          INTEGER ORDER OF THE MATRIX A.  DIMENSION OF A(*,*).
''IPVT       INTEGER(N).                    VECTOR OF PIVOT INDICES.
''INFO       INTEGER INDICATOR.             = 0 FOR NORMAL VALUE.
''                                          = K IF U(K,K) EQ 0.0.  THIS
''                                          INDICATES THAT SGESL OR SGEDI
''                                          WILL DIVIDE BY 0 IF CALLED.
''
    DEFINE I,INFO,J,K,KP1,L,N,NM1 AS INTEGER VARIABLES
    DEFINE IPVT AS AN INTEGER, 1-DIMENSIONAL ARRAY
    DEFINE A AS A REAL, 2-DIMENSIONAL ARRAY
''
''GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING.
''
    LET N=DIM.F(IPVT(*))
    LET INFO=0
    LET NM1=N-1
    IF NM1 < 1
        GO TO L7
    OTHERWISE
    FOR K=1 TO NM1 DO
        LET KP1=K+1
''
''FIND L = PIVOT INDEX IN THIS COLUMN.
''
        LET SMAX=ABS.F(A(K,K))
        LET L=K
        FOR I=K+1 TO N DO
            IF ABS.F(A(I,K)) > SMAX
                LET L=I
```

```
44                  LET SMAX=ABS.F(A(I,K))
45              ALWAYS
46          LOOP ''FOR MAX ELEMENT
47          LET IPVT(K)=L
48  ''
49  ''ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED.
50  ''
51          IF A(L,K) = 0.0
52              GO TO L4
53          OTHERWISE
54  ''
55  ''INTERCHANGE IF NECESSARY.
56  ''
57          IF L = K
58              GO TO L1
59          OTHERWISE
60          LET T=A(L,K)
61          LET A(L,K)=A(K,K)
62          LET A(K,K)=T
63  'L1'    LET T=-1.0/A(K,K)
64          FOR I=K+1 TO N, LET A(I,K)=T*A(I,K)
65  ''
66  ''ROW ELIMINATION WITH COLUMN INDEXING.
67  ''
68          FOR J=KP1 TO N DO
69              LET T=A(L,J)
70              IF L=K
71                  GO TO L2
72              OTHERWISE
73              LET A(L,J)=A(K,J)
74              LET A(K,J)=T
75  'L2'        FOR I=K+1 TO N, LET A(I,J)=T*A(I,K)+A(I,J)
76          LOOP ''OVER (J) COLUMNS
77          GO TO L5
78  'L4'    LET INFO=K
79  'L5'LOOP ''OVER K
80  'L7'LET IPVT(N)=N
81      IF A(N,N)=0.0
82          LET INFO=N
83      ALWAYS
84      RETURN
85  END ''SGEFA
```

```
 1  ROUTINE SGEDI (A, IPVT, DET, JOB)
 2  ''
 3  ''
 4  ''SGEDI COMPUTES THE DETERMINANT AND INVERSE OF A MATRIX USING THE
 5  ''RESULTS PRODUCED BY SGEFA.
 6  ''
 7  ''ARGUMENTS:
 8  ''A(*,*)  _ THE REAL FACTORED MATRIX FROM SGEFA ON INPUT.  ON OUTPUT THE
 9  ''          ARRAY CONTAINS THE MATRIX INV, IF REQUESTED. ELSE, UNCHANGED.
10  ''IPVT(*)  THE INTEGER PIVOT VECTOR FROM SGEFA.
11  ''JOB ____ AN INTEGER SWITCH.
12  ''          = 11 FOR BOTH DETERMINANT AND INVERSE.
13  ''          = 01 FOR INVERSE ONLY.
14  ''          = 10 FOR DETERMINANT ONLY.
```

```
''DET(*) _ CONTAINS THE DETERMINANT OF THE MATRIX, IF REQUESTED.  ELSE,
''         IS NOT REFERENCED.  DETERMINANT = DET(1)*10.0**DET(2), WITH
''         DET(1) BETWEEN 0 AND 10, AND WITH DET(2) A FLOATED INTEGER.
''
''NOTE:  A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS A
''ZERO ON THE DIAGONAL AND THE INVERSE IS REQUESTED.
''
    DEFINE I,J,JOB,K,KB,KP1,L,N,NM1 AS INTEGER VARIABLES
    DEFINE IPVT AS AN INTEGER, 1-DIMENSIONAL ARRAY
    DEFINE DET, WORK AS REAL, 1-DIMENSIONAL ARRAYS
    DEFINE A AS A REAL, 2-DIMENSIONAL ARRAY
    LET N=DIM.F(IPVT(*))
    RESERVE WORK(*) AS N ''LOCALLY
''
''CALCULATE THE DETERMINANT IF REQUESTED.
''
    IF DIV.F(JOB,10) = 0
        GO TO L6
    OTHERWISE
    LET DET(1)=1.0
    LET DET(2)=0.0
    LET TEN=10.0
    FOR I=1 TO N DO
        IF IPVT(I) NE I
            LET DET(1) = -DET(1)
        ALWAYS
        LET DET(1)=A(I,I)*DET(1)
        IF DET(1)=0.0
            GO TO L6
        OTHERWISE
'L1'    IF ABS.F(DET(1)) GE 1.0
            GO TO L2
        OTHERWISE
        LET DET(1)=TEN*DET(1)
        SUBTRACT 1.0 FROM DET(2)
        GO TO L1
'L2'    IF ABS.F(DET(1)) < TEN
            GO TO L4
        OTHERWISE
        LET DET(1)=DET(1)/TEN
        ADD 1.0 TO DET(2)
        GO TO L2
'L4'LOOP ''OVER I
''
''GET INVERSE OF UPPER TRIANGULAR MATRIX U(*,*).
''
'L6'IF MOD.F(JOB,10)=0
        RELEASE WORK(*)
        RETURN
    OTHERWISE
    FOR K=1 TO N DO
        LET A(K,K)=1.0/A(K,K)
        LET T=-A(K,K)
        FOR I=1 TO K-1, LET A(I,K)=T*A(I,K)
        LET KP1=K+1
        IF N < KP1
            GO TO L9
```

```
        OTHERWISE
        FOR J=KP1 TO N DO
            LET T=A(K,J)
            LET A(K,J)=0.0
            FOR I=1 TO K, LET A(I,J)=T*A(I,K)+A(I,J)
        LOOP ''OVER J
'L9'LOOP ''OVER K
''
''FORM INVERSE(U)*INVERSE(L)
''
    LET NM1=N-1
    IF NM1 < 1
        RELEASE WORK(*)
        RETURN
    OTHERWISE
    FOR KB=1 TO NM1 DO
        LET K=N-KB
        LET KP1=K+1
        FOR I=KP1 TO N DO
            LET WORK(I)=A(I,K)
            LET A(I,K)=0.0
        LOOP ''OVER I
        FOR J=KP1 TO N DO
            LET T=WORK(J)
            FOR I=1 TO N, LET A(I,K)=T*A(I,J)+A(I,K)
        LOOP ''OVER J
        LET L=IPVT(K)
        IF L NE K ''SWAP ELEMENTS OF VECTORS K AND L
            FOR I=1 TO N DO
                LET T=A(I,K)
                LET A(I,K)=A(I,L)
                LET A(I,L)=T
            LOOP ''OVER I TO SWAP
        ALWAYS
    LOOP ''OVER KB
    RELEASE WORK(*)
    RETURN
END ''SGEDI
```

```
ROUTINE ERLANG GIVEN N, R, T YIELDING PDF, CDF
''
''Calculates the probability density function (PDF) and cum distribu-
''tion function (CDF) for an Erlang function with integer shape para-
''meter N, with rate parameter R, and with real argument T.
''
    DEFINE I,N AS INTEGER VARIABLES
    LET Z=R*T
    LET EXPZ=EXP.F(-Z)
    LET FACT=1.0
    LET ZI=1.0
    LET SUM=1.0
    FOR I=1 TO N-1 DO
        LET FACT=FACT*I
        LET ZI=ZI*Z
        ADD ZI/FACT TO SUM
    LOOP ''OVER I
    LET PDF=R*ZI/FACT*EXPZ
```

```
19      LET CDF=1.0-EXPZ*SUM
20      RETURN
21  END ''ROUTINE ERLANG

 1  ROUTINE NFOLD.MIXE (N,A1,LA1,LA2,T) YIELDING PDF,CDF
 2  ''
 3  ''Routine produces the probability density function (PDF) and cum-
 4  ''ulative distribution function (CDF) for an N-fold convolution of a
 5  ''two-component exponential mixture function having first proportion of
 6  ''A1 and with rate parameters LA1 and LA2.  Real-valued argument is T.
 7  ''
 8      DEFINE I,J,K,N AS INTEGER VARIABLES
 9      LET A2=1.0-A1
10      LET E1=EXP.F(-LA1*T)
11      LET E2=EXP.F(-LA2*T)
12      IF N=1
13          LET PDF=A1*LA1*E1+A2*LA2*E2
14          LET CDF=A1*(1.0-E1)+A2*(1.0-E2)
15          RETURN
16      OTHERWISE
17      IF N=2
18          LET XCOEF=2.0*A1*A2*LA1*LA2/(LA1-LA2)
19          LET PDF=(A1*LA1)**2*T*E1+(A2*LA2)**2*T*E2+XCOEF*(E2-E1)
20          LET CDF=1.0-A1**2*E1*(1.0+LA1*T)-A2**2*E2*(1.0+LA2*T)
21                  -XCOEF*(E2/LA2-E1/LA1)
22          RETURN
23      OTHERWISE
24      IF N=3
25          LET ARG1=LA1*T
26          LET ARG2=LA2*T
27          LET F13=1.0-E1*(1.0+ARG1+0.5*ARG1**2)
28          LET F23=1.0-E2*(1.0+ARG2+0.5*ARG2**2)
29  ''      LET F12=1.0-E1*(1.0+ARG1)
30  ''      LET F22=1.0-E2*(1.0+ARG2)
31  ''      LET F11=1.0-E1
32  ''      LET F21=1.0-E2
33          LET A=1.0/LA1/(LA1-LA2)
34          LET B=1.0/LA1/LA2 - LA1/LA2/(LA2-LA1)**2
35          LET APB=A+B
36          LET C=1.0/(LA2-LA1)**2
37          LET AP=1.0/LA2/(LA2-LA1)
38          LET BP=1.0/LA1/LA2 - LA2/LA1/(LA2-LA1)**2
39          LET APP=AP+BP
40          LET PDF=0.5*A1**3*LA1*ARG1**2*E1+0.5*A2**3*LA2*ARG2**2*E2+3.0*A1
41          **2*A2*(APB*E1*LA1**2*LA2-A*LA1**3*ARG2*E1+C*LA1**2*LA2*E2)+3.0
42          *A1*A2**2*(APP*E2*LA1*LA2**2-AP*LA2**3*ARG1*E2+C*LA1*LA2**2*E1)
43          LET CDF=A1**3*F13+A2**3*F23+3.0*A1**2*A2*(1.0-E2*LA1**2/(LA1-
44          LA2)**2-(ARG1*LA2/(LA2-LA1)+(LA2-2.0*LA1)*LA2/(LA2-LA1)**2)*E1)
45          +3.0*A1*A2**2*(1.0-E1*LA2**2/(LA2-LA1)**2
46          -(ARG2*LA1/(LA1-LA2)+(LA1-2.0*LA2)*LA1/(LA1-LA2)**2)*E2)
47          RETURN
48      OTHERWISE
49      PRINT 1 LINE WITH N
50      THUS
 INPUT ERROR TO ROUTINE NFOLD.MIXE.  N =  **
52      STOP
53  END ''NFOLD.MIXE
```

ANNEX E

SIMSCRIPT SOURCE PROGRAM: INT.TEST

```
 1  PREAMBLE ''INT.TEST
 2  NORMALLY MODE IS REAL
 3  DEFINE SNORM AS A REAL FUNCTION GIVEN 1 ARGUMENT
 4  DEFINE ERRFX AS A REAL FUNCTION GIVEN 1 ARGUMENT
 5  DEFINE FACTORIAL AS A REAL FUNCTION GIVEN 1 ARGUMENT
 6  DEFINE COMPLETE.GAMMA AS A REAL FUNCTION GIVEN 1 ARGUMENT
 7  DEFINE NUM.CNVL AS A REAL FUNCTION GIVEN 3 ARGUMENTS
 8  END ''PREAMBLE

 1  MAIN ''INT.TEST
 2  ''
 3  ''Program tests a variety of methods for obtaining convolution integrals
 4  ''of a two-parameter Weibull distribution.  Program compares Leonard
 5  ''Johnson's approx for the 2nd order failure distribution with an exact
 6  ''expression when time to fail is a Weibull RV.  Ref:  Reliability and
 7  ''Maintainability of the M48A1 Tank, p.26 ff.
 8      DEFINE ANSWER AS A TEXT VARIABLE
 9      DEFINE FLAGM,I,J,K,KORD,M,N,NCELLS,NREPS,SEED AS INTEGER
10      VARIABLES
11      DEFINE HISTV AS AN INTEGER, 1-DIMENSIONAL ARRAY
12      DEFINE TV,FYV,FZV,DELFXV AS REAL, 1-DIMENSIONAL ARRAYS
13      LET N=1024 ''ELEMENTS IN FYV(*)
14      RESERVE FYV(*),FZV(*),DELFXV(*) AS N
15      LET LINES.V=9999
16      LET NCELLS=20
17      RESERVE HISTV(*),TV(*) AS NCELLS
18      PRINT 5 LINES THUS
Program calculates the convolution integral of N (N le 8) identical Weibull
distributions via several methods.  This convolution distribution is the c.d.f.
of the sum of N, identical Weibull random variables.  Methods include:
(a) evaluation of an analytic expression, (b) Leonard Johnson's (L-J) approxi-
mation, (c) finite numerical convolution, and (d) Monte-Carlo simulation.
24  'LO'SKIP 2 LINES
25      PRINT 1 LINE THUS
INPUT THE SCALE PARAMETER OF THE WEIBULL DISTRIBUTION.
27      READ ETA
28      LET C=2.0/ETA**2
29      PRINT 1 LINE THUS
INPUT THE WEIBULL SHAPE PARAMETER.
31      READ SHAPE
32  'L1'PRINT 1 LINE THUS
INPUT THE NUMBER (LE 8) OF CONVOLUTIONS OF THIS DISTRIBUTION WANTED.
34      READ KORD
35      IF KORD > 8
36          GO TO L1
37      OTHERWISE
38      LET ORDER=KORD
39  ''
40  ''FILL THE ARRAYS OF DISCRETE VALUES OF THE C.D.F.
41  ''
42      LET FX0=0.0
43      LET ERR=0.00001
```

```
    44      LET TMAX=ETA*(-LOG.E.F(ERR))**(1.0/SHAPE)
    45      LET AVG1=ETA*COMPLETE.GAMMA(1.0+1.0/SHAPE)
    46      LET VAR1=ETA**2*COMPLETE.GAMMA(1.0+2.0/SHAPE) - AVG1**2
    47      LET STDV1=SQRT.F(VAR1)
    48      LET TMAX=MAX.F(TMAX,ORDER*AVG1+3.5*SQRT.F(ORDER*VAR1))
    49      LET DELZ=TMAX/N
    50      FOR I=1 TO N DO
    51          LET X=I*DELZ
    52          LET FX=1.0 - EXP.F(-(X/ETA)**SHAPE)
    53          LET FYV(I)=FX
    54          LET DELFXV(I)=FX-FX0
    55          LET FX0=FX
    56      LOOP ''OVER (I) DISCRETE POINTS OF THE CDF
    57      IF KORD > 2
    58          PRINT 1 LINE WITH N
    59          THUS
Starting numerical convolution with **** points.
    61          FOR K=1 TO KORD-2 DO
    62              FOR I=1 TO N, LET FZV(I)=NUM.CNVL (I, FYV(*), DELFXV(*))
    63              FOR I=1 TO N, LET FYV(I)=FZV(I)
    64          LOOP ''OVER (K) CONVOL ORDERS
    65          PRINT 1 LINE THUS
Numerical convolution completed.
    67      ALWAYS
    68      LET PSI=COMPLETE.GAMMA(ORDER+1.0/COMPLETE.GAMMA(1.0+1.0/SHAPE)/
    69      SHAPE)/FACTORIAL(KORD)
    70      LET DELT=TMAX/NCELLS
    71      FOR K=1 TO NCELLS, LET TV(K)=K*DELT
    72      PRINT 1 LINE THUS
DO YOU WANT A MONTE-CARLO ESTIMATE OF THE CONVOLUTION C.D.F.?  (Y OR N).
    74      READ ANSWER
    75      IF SUBSTR.F(ANSWER,1,1) = "Y"
    76          LET FLAGM=1
    77          PRINT 1 LINE THUS
INPUT THE INDEX OF THE RANDOM # SEED.
    79          READ SEED
    80          PRINT 1 LINE THUS
INPUT THE NUMBER OF REPLICATIONS WANTED.
    82          READ NREPS
    83          PRINT 1 LINE WITH NREPS
    84          THUS
A Monte-Carlo simulation of ***** replications has begun.
    86          FOR K=1 TO NCELLS, LET HISTV(K)=0
    87          LET AVGT=0.0
    88          LET VART=0.0
    89  ''
    90  ''SIMULATE FOR NREPS REPLICATIONS.
    91  ''
    92          FOR I=1 TO NREPS DO
    93              LET SUM=0.0
    94              FOR J=1 TO KORD DO
    95                  ADD WEIBULL.F(SHAPE,ETA,SEED) TO SUM
    96              LOOP ''OVER J
    97              ADD SUM TO AVGT
    98              ADD SUM**2 TO VART
    99              FOR K=1 TO NCELLS DO
   100                  IF SUM LE TV(K)
```

```
                    ADD 1 TO HISTV(K)
                    GO TO K2
                    OTHERWISE
                LOOP ''OVER (K) CELLS
'K2'        LOOP ''OVER (I) REPLICATIONS
            LET AVGT=AVGT/NREPS
            LET VART=VART/NREPS-AVGT**2
            PRINT 1 LINE THUS
Monte-Carlo simulation has been completed.
        OTHERWISE
            LET FLAGM=0
        ALWAYS
        SKIP 2 LINES
        PRINT 1 LINE WITH KORD,SHAPE,ETA
        THUS
CONVOLUTION C.D.F. OF ORDER ** OF A WEIBULL DIST: SHAPE **.* AND SCALE ***.***
        PRINT 5 LINES THUS


-------------------------------------------------------------------------------
Indep     Exact     L-J Aprx    Numerical   Normal    Histo     Sample
Variable  c.d.f.(1) c.d.f.      c.d.f.(2)   Aprx      Freq      c.d.f.
-------------------------------------------------------------------------------
        LET CDFX=0.0
        LET MAELJ=0.0 ''FOR MAX ABS ERROR IN C.D.F. FOR L-J APPROX
        LET MAEDN=0.0 ''FOR MAX ABS ERROR IN C.D.F FOR DISCRETE NUM APPROX
        LET MAEMC=0.0 ''FOR MAX ABS ERROR IN C.D.F. FOR MONTE CARLO
        LET MAENA=0.0 ''FOR MAX ABS ERROR IN C.D.F. OR NORMAL APPROX
        LET RMSLJ=0.0
        LET RMSDN=0.0
        LET RMSMC=0.0
        LET RMSNA=0.0
        LET AVG=ORDER*AVG1
        LET VAR=ORDER*VAR1
        LET STDV=SQRT.F(VAR)
        FOR K=1 TO NCELLS DO
            LET T=TV(K)
            LET M=(T+0.4999*DELZ)/DELZ
            LET X=(PSI*T/ETA)**SHAPE
            LET SUM=1.0
            LET FACT=1.0
            LET XI=1.0
            FOR I=1 TO KORD-1 DO
                LET FACT=FACT*I
                LET XI=XI*X
                ADD XI/FACT TO SUM
            LOOP ''OVER I
            LET QK=1.0 - EXP.F(-X)*SUM
            IF KORD=1
                LET FZ=FYV(M)
            OTHERWISE
                LET FZ=NUM.CNVL (M, FYV(*), DELFXV(*))
            ALWAYS
            IF SHAPE=2.0 AND KORD=2
                LET ARG=T*SQRT.F(C/2.0)
                LET INTG=EXP.F(0.5*ARG**2)*(SNORM(ARG)-SNORM(-ARG))*ARG*
                SQRT.F(PI.C/2.0)
                LET Q2=1.0 - EXP.F(-ARG**2)*(1.0+INTG)
```

```
  158          OTHERWISE
  159              LET Q2=FZ
  160          ALWAYS
  161          LET MAELJ=MAX.F(MAELJ,ABS.F(Q2-QK))
  162          LET MAEDN=MAX.F(MAEDN,ABS.F(Q2-FZ))
  163          LET FN=SNORM((T-AVG)/STDV)
  164          LET NERR=FN-Q2
  165          LET MAENA=MAX.F(MAENA,ABS.F(NERR))
  166          ADD NERR**2 TO RMSNA
  167          ADD (Q2-QK)**2 TO RMSLJ
  168          ADD (Q2-FZ)**2 TO RMSDN
  169          IF FLAGM=1
  170              LET PDFX=HISTV(K)/NREPS
  171              ADD PDFX TO CDFX
  172              LET MAEMC=MAX.F(MAEMC,ABS.F(Q2-CDFX))
  173              ADD (Q2-CDFX)**2 TO RMSMC
  174              PRINT 1 LINE WITH T,Q2,QK,FZ,FN,HISTV(K),CDFX
  175              THUS
***.***   *.******   *.******   *.******   *.******  ******   *.******
  177          OTHERWISE
  178              PRINT 1 LINE WITH T,Q2,QK,FZ,FN
  179              THUS
***.***   *.******   *.******   *.******   *.******
  181          ALWAYS
  182      LOOP ''OVER (K) VALUES OF TIME
  183      LET RMSLJ=SQRT.F(RMSLJ/REAL.F(NCELLS))
  184      LET RMSDN=SQRT.F(RMSDN/REAL.F(NCELLS))
  185      LET RMSMC=SQRT.F(RMSMC/REAL.F(NCELLS))
  186      LET RMSNA=SQRT.F(RMSNA/REAL.F(NCELLS))
  187      PRINT 4 LINES WITH N
  188      THUS

-------------------------------------------------------------------------
(1)  The discrete numerical approx is treated as exact if either the
     Weibull shape parameter is not 2 or the number of convolutions is not 2.
(2)  Number of discrete points in numerical convolution  ****
  193      PRINT 4 LINES WITH MAELJ,RMSLJ,MAEDN,RMSDN,MAENA,RMSNA,MAEMC,RMSMC
  194      THUS
Max abs error and RMS error in c.d.f. of L-J approximation  *.******  *.******
Max abs error and RMS error in c.d.f. of discrete numerical *.******  *.******
Max abs error and RMS error in c.d.f. of Normal approx      *.******  *.******
Max abs error and RMS error in c.d.f. of Monte-Carlo sim    *.******  *.******
  199      PRINT 2 LINES WITH AVG,STDV
  200      THUS
Mean of the convolution distribution   ****.****   Std Dev  ****.****

  203      IF FLAGM=1
  204          LET SET=SQRT.F(VART/REAL.F(NREPS))
  205          PRINT 3 LINES WITH AVGT,SQRT.F(VART),AVGT-1.96*SET,AVGT+1.96*SET
  206          THUS
Sample average of sum of Weibull RVs   ****.****   Std Dev  ****.****
95 percent confidence interval in mean ****.****, ****.****

  210      ALWAYS
  211      PRINT 1 LINE THUS
DO YOU WANT TO CONTINUE?  (YES OR NO).
  213      READ ANSWER
```

```
214       IF SUBSTR.F(ANSWER,1,1) = "Y"
215           GO TO L0
216       OTHERWISE
217       STOP
218   END ''INT.TEST

  1   FUNCTION FACTORIAL(N)
  2   ''
  3   ''CALCULATES THE FACTORIAL OF INTEGER N.
  4   ''
  5   DEFINE I AND N AS INTEGER VARIABLES
  6       IF N LE 0
  7           RETURN WITH 1.0
  8       OTHERWISE
  9       LET F=1.0
 10       FOR I=1 TO N, LET F=F*I
 11       RETURN WITH F
 12   END ''FACTORIAL

  1   FUNCTION COMPLETE.GAMMA(XX)
  2   ''
  3   ''CALCULATES THE COMPLETE GAMMA FUNCTION WITH SINGLE REAL ARGUMENT XX.
  4   ''METHOD: THE RECURSION RELATION AND POLYNOMIAL APPROXIMATION IS TAKEN
  5   ''FROM:  C. HASTINGS, JR, 'APPROXIMATIONS FOR DIGITAL COMPUTERS,'
  6   ''PRINCETON UNIV. PRESS, 1955.
  7   ''
  8           IF XX > 57.0
  9               GO TO L130
 10           OTHERWISE
 11   'L6'    LET X = XX
 12           LET ERR = 0.000001
 13           LET GAMMA = 1.0
 14           IF X LE 2.0
 15               GO TO L50
 16           OTHERWISE
 17               GO TO L15
 18   'L10'   IF X LE 2.0
 19               GO TO L110
 20           OTHERWISE
 21   'L15'   SUBTRACT 1.0 FROM X
 22           LET GAMMA = GAMMA * X
 23           GO TO L10
 24   'L50'   IF X = 1.0
 25               GO TO L120
 26           OTHERWISE
 27           IF X > 1.0
 28               GO TO L110
 29           OTHERWISE
 30   'L60'   IF X > ERR
 31               GO TO L80
 32           OTHERWISE
 33           LET Y = REAL.F(TRUNC.F(X))-X
 34           IF ABS.F(Y) LE ERR
```

```
    35              GO TO L130
    36          OTHERWISE
    37          IF Y+ERR GE 1.0
    38              GO TO L130
    39          OTHERWISE
    40  'L70'   IF X > 1.0
    41              GO TO L110
    42          OTHERWISE
    43  'L80'   LET GAMMA = GAMMA / X
    44          ADD 1.0 TO X
    45          GO TO L70
    46  'L110'  LET Y = X - 1.0
    47          LET GY = 1.0+Y*(-0.57710166+Y*(0.98585399+Y*(-0.87642182+Y*
    48          (0.83282120+Y*(-0.56847290+Y*(0.25482049+Y*(-0.05149930))))))
    49          LET GAMMA = GAMMA * GY
    50  'L120'  RETURN WITH GAMMA
    51  'L130'  PRINT 1 LINE WITH XX THUS
ERROR IN COMPLETE.GAMMA.  ARGUMENT = ***.*****
    53          STOP
    54  END ''COMPLETE.GAMMA

     1  FUNCTION NUM.CNVL (N, FYV, DELFXV)
     2  ''
     3  ''Function calculates a value of the c.d.f. of the sum of two random
     4  ''variables-- x and y--whose c.d.f.'s are evaluated at a discrete #
     5  ''of points on their domains.  This distribution of the sum is the con-
     6  ''volution of the dist's of x and y.  The convolution distribution is
     7  ''evaluated for the N th discrete argument.  The set of c.d.f. values
     8  ''of y are given by the vector FYV, and the first backward differences
     9  ''in the c.d.f. of x, defined on the same finite domain, are given in
    10  ''DELFXV.
    11      DEFINE I,N AS INTEGER VARIABLES
    12      DEFINE FYV,DELFXV AS REAL, 1-DIMENSIONAL ARRAYS
    13      LET GN=0.0
    14      FOR I=1 TO N-1, ADD FYV(N-I)*DELFXV(I) TO GN
    15      RETURN WITH GN
    16  END ''FUNCTION NUM.CNVL
```

ANNEX F

SIMSCRIPT SOURCE PROGRAM: TEST.CONVOLV

```
 2  PREAMBLE ''TEST.CONVOLV
 3  NORMALLY MODE IS REAL
 4  DEFINE SNORM AS A REAL FUNCTION GIVEN 1 ARGUMENT
 5  DEFINE ERRFX AS A REAL FUNCTION GIVEN 1 ARGUMENT
 6  DEFINE COMPLETE.GAMMA AS A REAL FUNCTION GIVEN 1 ARGUMENT
 7  DEFINE W2WFUN AS A REAL FUNCTION GIVEN 1 ARGUMENT
 8  DEFINE EFUN AS A REAL FUNCTION GIVEN 2 ARGUMENTS
 9  DEFINE WFUN AS A REAL FUNCTION GIVEN 2 ARGUMENTS
10  END ''PREAMBLE

 1  MAIN ''TEST.CONVOLV
 2  ''
 3  ''Program to run routine CONVOLV.  This program generates 2 p.d.f.'s
 4  ''defined on a discrete point set, from prob dist's to be numerically
 5  ''convolved via Fourier transformation, multiplication of transforms,
 6  ''and inversion.  The # of real (as opposed to imaginary) points in
 7  ''the transform and its inverse must be a power of 2 in order to use
 8  ''the Cooley-Tukey FFT algorithm.  Comparisons with exact results and,
 9  ''optionally, Monte-Carlo results are also given.
10  ''
11  DEFINE FLAGE,FLAGM,FLAGW,I,J,K,L,M,MINCR,N,NFOLD,NCELLS,NREPS,SEED AS
12  INTEGER VARIABLES
13  DEFINE ANSWER,FUN.NAME AS TEXT VARIABLES
14  DEFINE HISTV AS AN INTEGER, 1-DIMENSIONAL ARRAY
15  DEFINE TV,XV,YV,PDFV,CDFV AS REAL, 1-DIMENSIONAL ARRAYS
16      PRINT 11 LINES THUS
 This program calculates the probability distribution of the sum of a
set of random variables of a particular type, such as Erlang or Weibull.
This is equivalent to obtaining the N-fold convolution of the probability
functions of the set of N.  For a given type of random variable, two sets
of parameters are permitted.  Distributions having the 1st parameter set are
convolved N-1 times with the distribution having the 2nd parameter set.
Where available, exact results are calculated and displayed.  A numerical
method for obtaining convolution integrals based on the Fourier transform
is used in all cases to obtain an approximation of the convolution p.d.f.
and c.d.f.  Optionally, Monte-Carlo simulation is used for sample estimates.

28      PRINT 3 LINES
29      THUS
The current program version treats convolutions of an Erlang or a Weibull
distribution in standardized form, i.e., characterized by a shape parameter.

33      PRINT 1 LINE THUS
 IF THE ERLANG FORM IS WANTED, INPUT AN  E; OTHERWISE, INPUT A W.
35      READ ANSWER
36      IF SUBSTR.F(ANSWER,1,1) = "E"
37          LET FLAGW=0
38          LET FLAGE=1 ''TRIGGER FORMAT FOR EXACT RESULTS
39          LET FUN.NAME= "Erlang"
40      OTHERWISE
41          LET FLAGW=1
```

```
42          LET FUN.NAME= "Weibull"
43      ALWAYS
44      PRINT 1 LINE THUS
 INPUT THE NUMBER OF CONVOLUTIONS WANTED.
46      READ NFOLD
47  ''  LET N=4096
48      LET N=8192
49      LET M=DIV.F(N,2) ''NUMBER OF REAL POINTS IN THE SERIES
50      RESERVE XV(*), YV(*) AS N
51      LET MINCR=256 ''SKIP INTERVAL FOR PRINTING
52      LET NCELLS=DIV.F(M,MINCR)
53      LET NCELLS=MAX.F(16,NCELLS)
54      LET MINCR=DIV.F(M,NCELLS)
55      RESERVE HISTV(*) AS NCELLS
56      RESERVE TV(*) AS NCELLS ''FOR INDEPENT VAR IN A HISTOGRAM
57      RESERVE PDFV(*),CDFV(*) AS NCELLS
58      LET LINES.V=9999
59      LET ETA1=1.0
60      LET ETA2=1.0
61      IF FLAGW=1
62          GO TO L7
63      OTHERWISE
64  'LO'PRINT 1 LINE WITH FUN.NAME
65      THUS
 INPUT THE INTEGER SCALE PARAM OF THE 1ST STD ******* DISTRIBUTION.
67      READ K
68      IF K < 1
69          PRINT 1 LINE THUS
 Try again using a positive integer.
71          GO TO L0
72      OTHERWISE
73  ''
74  ''CALCULATE MEAN AND VARIANCE OF 1ST DIST.
75  ''
76      LET AVG1=K
77      LET VAR1=K
78  'L1'PRINT 1 LINE WITH FUN.NAME
79      THUS
 INPUT THE INTEGER SCALE PARAM OF THE 2ND STD ******* DISTRIBUTION.
81      READ L
82      IF L < 1
83          PRINT 1 LINE THUS
 Try again using a positive integer.
85          GO TO L1
86      OTHERWISE
87  ''
88  ''CALCULATE MEAN AND VAR OF 2ND DIST AND OF CONVOLUTION DIST.
89  ''
90      LET AVG2=L
91      LET VAR2=L
92      LET AVG=AVG1*(NFOLD-1)+AVG2
93      LET VAR=VAR1*(NFOLD-1)+VAR2
94      LET STDV=SQRT.F(VAR)
95      LET STDV1=SQRT.F(VAR1)
96      LET STDV2=SQRT.F(VAR2)
97      SKIP 2 LINES
98      PRINT 7 LINES WITH FUN.NAME,K,L,M
```

```
   99      THUS
EXACT CONVOLUTION OF TWO ******* DENSITIES WITH SHAPE PARAMS ** AND **
Number of real points in the Fourier transform  ****

-------------------------------------------------------------------------------
Indep      1st       1st       2nd       2nd       Conv      Conv      Normal
Variable   p.d.f.    c.d.f.    p.d.f.    c.d.f.    p.d.f.    c.d.f.    c.d.f.
-------------------------------------------------------------------------------
  107      LET RANGE=AVG+3.7*SQRT.F(VAR)
  108  'L4'LET DARG=RANGE/M
  109  ''
  110  ''CHECK RANGE AND MODIFY, AS NECESSARY.
  111  ''
  112      IF EFUN((NFOLD-1)*K+L,RANGE) < 0.9999
  113          ADD DARG TO RANGE
  114          GO TO L4
  115      OTHERWISE
  116      GO TO L8
  117  ''
  118  ''GET INPUTS FOR WEIBULL DISTRIBUTION.
  119  ''
  120  'L7'PRINT 1 LINE WITH FUN.NAME
  121      THUS
    INPUT THE SHAPE PARAMETER OF THE 1ST ******* DISTRIBUTION.
  123      READ SHAPE1
  124      PRINT 1 LINE WITH FUN.NAME
  125      THUS
    INPUT THE SHAPE PARAMETER OF THE 2ND ******* DISTRIBUTION.
  127      READ SHAPE2
  128      LET ERR=0.00001
  129      LET T1MAX=ETA1*(-LOG.E.F(ERR))**(1.0/SHAPE1)
  130      LET T2MAX=ETA2*(-LOG.E.F(ERR))**(1.0/SHAPE2)
  131      LET AVG1=ETA1*COMPLETE.GAMMA(1.0+1.0/SHAPE1)
  132      LET AVG2=ETA2*COMPLETE.GAMMA(1.0+1.0/SHAPE2)
  133      LET VAR1=ETA1**2*COMPLETE.GAMMA(1.0+2.0/SHAPE1) - AVG1**2
  134      LET STDV1=SQRT.F(VAR1)
  135      LET VAR2=ETA2**2*COMPLETE.GAMMA(1.0+2.0/SHAPE2) - AVG2**2
  136      LET STDV2=SQRT.F(VAR2)
  137      LET AVG=(NFOLD-1)*AVG1+AVG2
  138      LET VAR=(NFOLD-1)*VAR1+VAR2
  139      LET STDV=SQRT.F(VAR)
  140      LET RANGE=MAX.F(T1MAX,T2MAX)
  141      LET RANGE=MAX.F(RANGE,AVG+3.7*SQRT.F(VAR))
  142  ''
  143  ''PRINT HEADINGS FOR INPUT DISTRIBUTIONS.
  144  ''
  145      SKIP 2 LINES
  146      PRINT 3 LINES WITH FUN.NAME,SHAPE1,SHAPE2,M
  147      THUS
EXACT CONVOLUTION OF TWO ******* DENSITIES WITH SHAPE PARAMS **.* AND **.*

Number of real points in the Fourier transform  ****
  151      IF SHAPE1=2.0 AND SHAPE2=2.0
  152          IF ETA1 NE ETA2 OR NFOLD NE 2
  153              GO TO L9
  154          OTHERWISE
  155          LET FLAGE=1
  156          PRINT 4 LINES THUS
```

```
Indep      1st        1st        2nd        2nd        Conv       Conv       Normal
Variable   p.d.f.     c.d.f.     p.d.f.     c.d.f.     p.d.f.     c.d.f.     c.d.f.

  161           GO TO L8
  162       OTHERWISE
  163  'L9'LET FLAGE=0
  164       PRINT 4 LINES THUS

Indep       1st         1st         2nd         2nd
Variable    p.d.f.      c.d.f.      p.d.f.      c.d.f.

  169  'L8'LET ARG0=0.0 ''FIXED
  170       LET RANGE=RANGE-ARG0
  171       LET DARG=RANGE/M
  172       LET AVGTD=0.0 ''THEORETICAL, DISCRETIZED
  173       LET VARTD=0.0
  174       LET XSUM=1.0
  175       LET XXSUM=0.0
  176       LET F10=0.0
  177       LET F20=0.0
  178       LET F30=0.0
  179       LET J=0 ''TO COUNT CELLS
  180       LET MAENA=0.0
  181       LET RMSNA=0.0
  182  ''
  183  ''GET TEST FUNCTIONS.
  184  ''
  185       FOR I=1 TO M DO
  186           LET ARG=I*DARG+ARG0
  187           IF MOD.F(I,2)=0
  188               LET COEF=2.0
  189           OTHERWISE
  190               LET COEF=4.0
  191           ALWAYS
  192           IF FLAGW=0
  193               LET F1=EFUN(K,ARG)
  194               LET F2=EFUN(L,ARG)
  195               LET F3=EFUN((NFOLD-1)*K+L,ARG)
  196           OTHERWISE
  197               LET F1=WFUN(SHAPE1,ARG)
  198               LET F2=WFUN(SHAPE2,ARG)
  199               IF FLAGE=1
  200                   LET F3=W2WFUN(ARG)
  201               ALWAYS
  202           ALWAYS
  203           LET XV(2*I-1)=F1-F10
  204           LET YV(2*I-1)=F2-F20
  205           LET CDENS=F3-F30
  206           ADD CDENS*ARG TO AVGTD
  207           ADD CDENS*ARG**2 TO VARTD
  208           LET F10=F1
  209           LET F20=F2
  210           LET F30=F3
  211           LET UPPER=1.0-F3
  212           ADD COEF*UPPER TO XSUM
  213           ADD COEF*ARG*UPPER TO XXSUM
```

```
  214  ''
  215  ''FILL IMAGINARY COMPONENTS WITH ZEROS.
  216  ''
  217          LET XV(2*I)=0.0
  218          LET YV(2*I)=0.0
  219          IF MOD.F(I,MINCR)=0
  220              ADD 1 TO J
  221              LET TV(J)=ARG
  222              IF FLAGE=1
  223                  LET PDFV(J)=CDENS
  224                  LET CDFV(J)=F3
  225                  LET FN=SNORM((ARG-AVG)/STDV)
  226                  LET NERR=FN-F3
  227                  LET MAENA=MAX.F(MAENA,ABS.F(NERR))
  228                  ADD NERR**2 TO RMSNA
  229                  PRINT 1 LINE WITH ARG,XV(2*I-1),F1,YV(2*I-1),F2,CDENS,
  230                  F3,FN THUS
***.***  *.******  *.******  *.******  *.******  *.******  *.******  *.******
  232              OTHERWISE
  233                  PRINT 1 LINE WITH ARG,XV(2*I-1),F1,YV(2*I-1),F2
  234                  THUS
 **.****    *.******   *.******   *.******   *.******
  236              ALWAYS
  237          ALWAYS
  238      LOOP ''OVER (I) DATA POINTS
  239      PRINT 2 LINES THUS
```

```
  242      PRINT 4 LINES WITH FUN.NAME,AVG1,STDV1,FUN.NAME,AVG2,STDV2,AVG,STDV
  243      THUS
Mean and standard deviation of the 1st ******* dist'n:     ***.****  ***.****
Mean and standard deviation of the 2nd ******* dist'n:     ***.****  ***.****
Theoretical mean and SD of the convolution distribution:   ***.****  ***.****

  248      IF FLAGE=1
  249          LET VARTD=VARTD-AVGTD**2
  250          LET XSUM=XSUM*DARG/3.0
  251          LET XXSUM=2.0*DARG/3.0*XXSUM - XSUM**2
  252          PRINT 3 LINES WITH AVGTD,SQRT.F(VARTD),XSUM,SQRT.F(XXSUM)
  253          THUS
Avg and SD of theoretical, discretized convolution dist:   ***.****  ***.****
Alternate (2nd order) calculation of average and std dev:  ***.****  ***.****

  257          LET RMSNA=SQRT.F(RMSNA/REAL.F(NCELLS))
  258          PRINT 3 LINES WITH MAENA,RMSNA
  259          THUS
Max abs error and RMS error in c.d.f. of Normal approx:    *.******  *.******


  263      ALWAYS
  264  ''
  265  ''TAKE NUMERICAL CONVOLUTION.
  266  ''
  267      PRINT 2 LINES WITH NFOLD,M
  268      THUS
Starting ** convolutions using FFT with **** real points.
```

```
  271       CALL CONVOLV (NFOLD, XV(*), YV(*))
  272       SKIP 1 LINE
  273       LET SUM= 0.0
  274       FOR I=1 TO M, ADD YV(2*I-1) TO SUM
  275       PRINT 1 LINE WITH 2.0*SUM/N
  276       THUS
Cumulative of numerical convolution density is  .............
  278       SKIP 2 LINES
  279       PRINT 6 LINES WITH NFOLD,FUN.NAME
  280       THUS
EXACT VERSUS NUMERICAL CONVOLUTION OF ** ******* PROB DISTRIBUTIONS


-------------------------------------------------------------------------------
Indep       Theory       Theory       Numer        Numer        Diff
Variable    p.d.f.       c.d.f.       p.d.f.       c.d.f.       c.d.f.
-------------------------------------------------------------------------------
  287       LET CDF.FT=0.0
  288       LET J=0
  289       LET AVGFT=0.0
  290       LET VARFT=0.0
  291       LET MAEFT=0.0
  292       LET RMSFT=0.0
  293       LET XSUM=1.0
  294       LET XXSUM=0.0
  295       FOR I=1 TO M DO
  296           LET ARG=I*DARG+ARG0
  297           IF MOD.F(I,2)=0
  298               LET COEF=2.0
  299           OTHERWISE
  300               LET COEF=4.0
  301           ALWAYS
  302           LET PDF.FT=YV(2*I-1)*2.0/N
  303           ADD ARG*PDF.FT TO AVGFT
  304           ADD ARG**2*PDF.FT TO VARFT
  305           ADD PDF.FT TO CDF.FT
  306           LET UPPER=1.0-CDF.FT
  307           ADD COEF*UPPER TO XSUM
  308           ADD COEF*ARG*UPPER TO XXSUM
  309           IF MOD.F(I,MINCR)=0
  310               ADD 1 TO J
  311               IF FLAGE=1
  312                   LET PDF=PDFV(J)
  313                   LET CDF=CDFV(J)
  314               OTHERWISE
  315                   LET PDF=PDF.FT
  316                   LET CDF=CDF.FT
  317                   LET PDFV(J)=PDF
  318                   LET CDFV(J)=CDF
  319               ALWAYS
  320               LET DIFF=CDF.FT-CDF
  321               LET MAEFT=MAX.F(MAEFT,ABS.F(DIFF))
  322               ADD DIFF**2 TO RMSFT
  323               PRINT 1 LINE WITH ARG,PDF,CDF,PDF.FT,CDF.FT,DIFF
  324               THUS
**.****    *.******    *.******    *.******    *.******    *.******
  326           ALWAYS
  327       LOOP ''OVER (I) POINTS
```

```
  328       PRINT 2 LINES THUS
-----------------------------------------------------------------------

  331       LET VARFT=VARFT-AVGFT**2
  332       LET XSUM=XSUM*DARG/3.0
  333       LET XXSUM=2.0*DARG/3.0*XXSUM - XSUM**2
  334       LET RMSFT=SQRT.F(RMSFT/REAL.F(NCELLS))
  335       PRINT 5 LINES WITH NFOLD,FUN.NAME,AVGFT,SQRT.F(VARFT),XSUM,
  336       SQRT.F(XXSUM),MAEFT,RMSFT
  337       THUS
Mean value and standard deviation of the sum of **  ******* RVs via FFT:
Calculated mean  ***.****   Std Dev  ***.****
Alternate  mean  ***.****   Std Dev  ***.****
Max abs error and RMS error in convol c.d.f. via FFT  *.******  *.******

  343       PRINT 1 LINE THUS
    DO YOU WANT TO PERFORM A MONTE-CARLO SIMULATION?  (YES OR NO).
  345       READ ANSWER
  346       IF SUBSTR.F(ANSWER,1,1) NE "Y"
  347           GO TO L5
  348       OTHERWISE
  349   ''  LET FLAGM=1
  350       PRINT 1 LINE THUS
    INPUT THE INDEX (1 THRU 9) OF THE RANDOM # SEED.
  352       READ SEED
  353       PRINT 1 LINE THUS
    INPUT THE NUMBER OF REPLICATIONS WANTED.
  355       READ NREPS
  356       PRINT 1 LINE WITH NREPS
  357       THUS
A Monte-Carlo simulation of ***** replications has begun.
  359       LET AVGT=0.0
  360       LET VART=0.0
  361       FOR I=1 TO NCELLS, LET HISTV(I)=0
  362   ''
  363   ''SIMULATE FOR NREPS REPLICATIONS.
  364   ''
  365       FOR I=1 TO NREPS DO
  366           LET SUM=0.0
  367           FOR J=1 TO NFOLD-1 DO
  368               IF FLAGW=0
  369                   ADD ERLANG.F(AVG1,K,SEED) TO SUM
  370               OTHERWISE
  371                   ADD WEIBULL.F(SHAPE1,ETA1,SEED) TO SUM
  372               ALWAYS
  373           LOOP ''OVER (J) RV'S
  374           IF FLAGW=0
  375               ADD ERLANG.F(AVG2,L,SEED) TO SUM
  376           OTHERWISE
  377               ADD WEIBULL.F(SHAPE2,ETA2,SEED) TO SUM
  378           ALWAYS
  379           ADD SUM TO AVGT
  380           ADD SUM**2 TO VART
  381           FOR J=1 TO NCELLS DO
  382               IF SUM LE TV(J)
  383                   ADD 1 TO HISTV(J)
  384                   GO TO K2
```

```
  385              OTHERWISE
  386          LOOP ''OVER (J) CELLS
  387  'K2'LOOP ''OVER (I) REPLICATIONS
  388      LET AVGT=AVGT/NREPS
  389      LET VART=VART/NREPS - AVGT**2
  390      LET SET=SQRT.F(VART/REAL.F(NREPS))
  391      PRINT 3 LINES WITH NREPS,AVGT,SQRT.F(VART),AVGT-1.96*SET,AVGT+1.96*SET
  392      THUS
Sample mean from **** reps of Monte-Carlo sim  ***.****  Std Dev  ***.****
95% statistical confidence interval in mean:   ***.****, ***.****

  396      SKIP 2 LINES
  397      PRINT 6 LINES WITH NFOLD, FUN.NAME
  398      THUS
SAMPLE PROB DIST OF THE SUM OF A SET OF ** ******* RANDOM VARIABLES


-------------------------------------------------------------------------
Indep       Histo        Sample       Sample       Theory       Differ
Variable    Frequency    p.d.f.       c.d.f.       c.d.f.       c.d.f.
-------------------------------------------------------------------------
  405      LET XCDF=0.0
  406      LET MAEMC=0.0
  407      LET RMSMC=0.0
  408      FOR J=1 TO NCELLS DO
  409          LET XPDF=HISTV(J)/NREPS
  410          ADD XPDF TO XCDF
  411          LET CDF=CDFV(J)
  412          LET DIFF=XCDF-CDF
  413          LET MAEMC=MAX.F(MAEMC,ABS.F(DIFF))
  414          ADD DIFF**2 TO RMSMC
  415          PRINT 1 LINE WITH TV(J),HISTV(J),XPDF,XCDF,CDF,DIFF
  416          THUS
 ***.****    ******      *.******      *.******      *.******      *.******
  418      LOOP ''OVER (J) HISTO CELLS
  419      PRINT 2 LINES THUS
-------------------------------------------------------------------------

  422      LET RMSMC=SQRT.F(RMSMC/REAL.F(NCELLS))
  423      PRINT 2 LINES WITH MAEMC,RMSMC
  424      THUS
Max abs error and RMS error in convol c.d.f. via Monte-Carlo  *.******  *.******

  427  'L5'STOP
  428  END ''TEST.CONVOLV

    1  FUNCTION EFUN (K, X)
    2  ''
    3  ''Test cum prob function used in TEST.CONVOLV.  Function shown belo
    4  ''is a Erlang distribution with (integer) shape parameter K and stand-
    5  ''ardized argument X.
    6  ''
    7      DEFINE I,K AS INTEGER VARIABLES
    8      LET EX=EXP.F(-X)
    9      IF K= 1
   10          RETURN WITH 1.0-EX
   11      OTHERWISE
   12      LET FACT=1.0
```

```
13      LET XI=1.0
14      LET SUM=1.0
15      FOR I=1 TO K-1 DO
16          LET FACT=FACT*I
17          LET XI=XI*X
18          ADD XI/FACT TO SUM
19      LOOP ''OVER I
20      RETURN WITH 1.0-EX*SUM
21  END ''FUNCTION EFUN

 1  FUNCTION WFUN (SHAPE, ARG)
 2  ''
 3  ''Function calculates the cumulative probability function for a
 4  ''Weibull distribution having shape parameter SHAPE and standardized
 5  ''independent variable ARG.
 6  ''
 7      RETURN WITH 1.0 - EXP.F(-ARG**SHAPE)
 8  END ''FUNCTION WFUN

 1  FUNCTION W2WFUN (ARG)
 2  ''
 3  ''Calculates the convolution c.d.f. with argument ARG of 2 standardized
 4  ''Weibull probability distributions, each having shape parameter 2.
 5  ''
 6      LET INTG=EXP.F(0.5*ARG**2)*(SNORM(ARG)-SNORM(-ARG))*ARG*
 7      SQRT.F(PI.C/2.0)
 8      RETURN WITH 1.0 - EXP.F(-ARG**2)*(1.0+INTG)
 9  END ''FUNCTION W2WFUN

 1  ROUTINE CONVOLV (NFOLD, XV, YV)
 2  ''
 3  ''Routine for calculating the result of a sequence of convolutions on
 4  ''2 probability density functions (p.d.f.'s).  The 1st density (XV) is
 5  ''convolved with the 2nd (YV), and the result is recursively convolved
 6  ''NFOLD-1 times with the 1st p.d.f.  The program returns the NFOLD-con-
 7  ''convoluted p.d.f. (in complex form) in the vector YV.  Method:  The
 8  ''program obtains the Fourier transform (FT) of the X-series in XV(*),
 9  ''and the Y-series in YV(*).  Then, a complex product is calculated and
10  ''placed in YV(*).  NFOLD-1 additional complex products are taken
11  ''between YV(*) and XV(*).  This final product is inverted in place
12  ''in YV(*).
13  ''   NAME       TYPE                ENTRY VALUE          RETURN VALUE
14  ''    XV     REAL ARRAY       COMPLEX X-DENSITY      FT OF X-DENSITY
15  ''    YV     REAL ARRAY       COMPLEX Y-DENSITY      CONVOLUTION DENSITY
16  ''NOTE:  THE DIMENSION OF ARRAYS MUST BE AN INTEGER POWER OF 2.
17  ''
18      DEFINE I,IMAX,K,N,NFOLD,NP2 AS INTEGER VARIABLES
19      DEFINE XV, YV AS REAL, 1-DIMENSIONAL ARRAYS
20      LET N=DIM.F(XV(*))
21      LET IMAX=N/2
22  ''
23  ''CHECK VALUE OF N.
24  ''
25      LET NP2=1
26  'P0'LET NP2=NP2*2
27      IF NP2<N
28          GO TO P0
```

```
    OTHERWISE
    IF NP2>N
        PRINT 1 LINE WITH N THUS
IMPROPER VALUE OF INPUT-ARRAY DIMENSION (= *****) IN ROUTINE CONVOLV.
        STOP
    OTHERWISE ''GO ON
''
''OBTAIN THE FOURIER TRANSFORMS OF XV AND YV.
''
    CALL FOUR.TRANS(-1,XV(*))
    CALL FOUR.TRANS(-1,YV(*))
''
''PLACE PRODUCT IN YV(*).
''
    FOR K=1 TO NFOLD-1 DO
        FOR I=1 TO IMAX DO
            LET TEMPR=YV(2*I-1)
            LET TEMPI=YV(2*I)
            LET YV(2*I-1)=XV(2*I-1)*TEMPR-XV(2*I)*TEMPI
            LET YV(2*I)=XV(2*I-1)*TEMPI+XV(2*I)*TEMPR
        LOOP ''OVER (I) FOURIER FREQUENCIES
    LOOP ''OVER (K) CONVOLUTION ORDER
''
''GET INVERSE TRANSFORM.
''
    CALL FOUR.TRANS(1 YV(*))
    RETURN
END ''CONVOLV

ROUTINE FOUR.TRANS (ISIGN, DATA)
''
''Routine to calculate the Fourier transform (or inverse transform)
''of a sampled data trace, which is passed in the input vector DATA(*).
''The algorithm used is the Cooley-Tukey fast Fourier transform (FFT),
''implemented by Norman Brenner of the MIT Lincoln Lab.  The technique
''requires that the # of real data points (N) be EXACTLY 2**K, K > 0.
''If ISIGN = -1, the routine yields the transform.  If ISIGN = 1, the
''inverse transform is produced. Program output, in either case, is
''tne one-dimensional array DATA(*).  When giving the transform with
''N/2 complex frequencies, requiring N elements, the real and imaginary
''components are stored in adjacent storage positions.  If a ISIGN = -1
''transform is followed by a +1 transform, the original trace appears
''scaled by a factor of N.
''Transform amplitudes are defined by:
''FT(K)=SUM OVER J : EXP(-2*PI*IMAG*(J-1)*(K-1)/N)*DATA(J),
''1 LE K LE N.
''Input series in DATA must be in complex form with DIM.F(DATA(*))=2*N.
''
DEFINE I, ISIGN, NDIM, AND N AS INTEGER VARIABLES
DEFINE IP0,IP1,IP2,IP3,I1,I2A,I2B,I3,AND I3REV AS INTEGER VARIABLES
DEFINE  DATA AS A REAL, 1-DIMENSIONAL ARRAY
LET NDIM=DIM.F(DATA(*))
LET N=NDIM/2
    LET IP0=2
    LET IP3=IP0*N
    LET I3REV=1
FOR I3=1 TO IP3 BY IP0 DO ''TO P50
```

```
29      IF I3<I3REV
30          LET TEMPR=DATA(I3)
31          LET TEMPI=DATA(I3+1)
32          LET DATA(I3)=DATA(I3REV)
33          LET DATA(I3+1)=DATA(I3REV+1)
34          LET DATA(I3REV)=TEMPR
35          LET DATA(I3REV+1)=TEMPI
36      ALWAYS
37      LET IP1=IP3/2
38  'P3'IF I3REV>IP1
39          SUBTRACT IP1 FROM I3REV
40          LET IP1=IP1/2
41          IF IP1 GE IP0
42              GO TO P3
43          OTHERWISE
44      ALWAYS
45      ADD IP1 TO I3REV
46  LOOP ''OVER I3 (P50)
47      LET IP1=IP0
48  'P6'IF IP1 GE IP3
49          RETURN
50      OTHERWISE
51      LET IP2=IP1*2
52      LET THETA=2.0*PI.C/REAL.F(ISIGN*IP2/IP0)
53      LET SINTH=SIN.F(THETA/2.0)
54      LET WSTPR=-2.0*SINTH**2
55      LET WSTPI=SIN.F(THETA)
56      LET WR=1.0
57      LET WI=0.0
58  FOR I1=1 TO IP1 BY IP0 DO
59      FOR I3=I1 TO IP3 BY IP2 DO
60          LET I2A=I3
61          LET I2B=I2A+IP1
62          LET TEMPR=WR*DATA(I2B)-WI*DATA(I2B+1)
63          LET TEMPI=WR*DATA(I2B+1)+WI*DATA(I2B)
64          LET DATA(I2B)=DATA(I2A)-TEMPR
65          LET DATA(I2B+1)=DATA(I2A+1)-TEMPI
66          ADD TEMPR TO DATA(I2A)
67          ADD TEMPI TO DATA(I2A+1)
68      LOOP ''OVER I3
69      LET TEMPR=WR
70      LET WR=WR*WSTPR-WI*WSTPI+WR
71      LET WI=WI*WSTPR+TEMPR*WSTPI+WI
72  LOOP ''OVER I1
73      LET IP1=IP2
74      GO TO P6
75  END ''FOUR.TRANS
```