Algorithm AS 251: Multivariate Normal Probability Integrals with Product Correlation Structure

Author(s): Charles W. Dunnett

Source: *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 38, No. 3 (1989), pp. 564-579
Published by: Wiley for the Royal Statistical Society
Stable URL: https://www.jstor.org/stable/2347754
Accessed: 15-07-2019 13:43 UTC

```
      DETMUL = 0.0
      DO 410 K = 1, KSET
         DETMUL = DETMUL + (DET(K) - DETALL) * (NG(K) - 1) / TWO
  410 CONTINUE
C
C        Calculate RHO and W2
C
      M = NT - KSET
      R1 = 0.0
      R2 = 0.0
      DO 420 K = 1, KSET
         R1 = R1 + ONE / (NG(K) - 1)
         R2 = R2 + ONE / (NG(K) - 1) ** 2
  420 CONTINUE
      RHO = ONE - (R1 - ONE / M) * (2 * (NP ** 2) + 3 * NP - 1) /
     *      (F6 * (NP + 1) * (KSET - 1))
      W2 = NP * (NP + 1) * ((NP - 1) * (NP + 2) * (R2 - ONE / M ** 2) -
     *     F6 * (KSET - 1) * (1 - RHO) ** 2) / (F48 * RHO ** 2)
C
      MDF = (KSET - 1) * MP
      CHI = -TWO * RHO * DETMUL
      RETURN
      END
```

Algorithm AS 251

Multivariate Normal Probability Integrals with Product Correlation Structure

By Charles W. Dunnett†

McMaster University, Hamilton, Canada

[Received November 1984. Final revision October 1988]

Keywords: Multinormal; Multivariate normal; Multivariate Student distribution; Product correlation structure

Language

Fortran 77

Description and Purpose

Let $X_1, X_2, \ldots, X_N$ be multinormal with zero means, unit variances and correlation structure defined by

$$\rho_{IJ} = b_I b_J \qquad (I \neq J; -1 < b_I < 1). \tag{1}$$

The algorithm provides an approximation for

$$\text{PROB} = \Pr[(X_1, \ldots, X_N) \in \mathbb{R}] \tag{2}$$

with specified error bound, where $\mathbb{R}$ is a subset of N-dimensional space of the form

$$\mathbb{R} = \{(X_1, \ldots, X_N)\colon B(I) \leqslant X_I \leqslant A(I), \text{ for } I = 1, \ldots, N\}, \tag{3}$$

where $B(I) < A(I)$ are finite or infinite real numbers.

†*Address for correspondence*: Department of Mathematics and Statistics, McMaster University, 1280 Main Street West, Hamilton, Ontario, Canada, L8S 4K1.

If the mean μ_I of the Ith co-ordinate is not zero or its standard deviation σ_I is not unity, then μ_I must be subtracted from $A(I)$ and $B(I)$ and the results divided by σ_I before using the algorithm. Thus, our set-up is identical with that considered by Schervish (1984) except for the special correlation structure that is assumed here. The advantage of the present algorithm over Schervish's, when the product correlation structure specified by equation (1) holds, is that computing times are reduced considerably. The computing times for Schervish's algorithm increase rapidly with N, making his algorithm impractical to use for dimensions much higher than 5 or 6. With the present algorithm, in contrast, no practical limitation is imposed by the value of N.

Although the use of the present algorithm is restricted to problems where the correlation structure (1) holds, this restriction is satisfied in many practical problems. For example, the case where all the correlation coefficients are equal to a common value $\rho \geqslant 0$ occurs frequently and may be handled by defining $b_I = \sqrt{\rho}$. The correlation structure given by $b_I = 1/\sqrt{(1 + n_0/n_I)}$ arises in multiple comparisons between N treatments and a control, where n_0 and n_I are the sample sizes in the control and Ith treatment groups (Dunnett, 1955). The same correlation structure arises in the method of Hsu (1984) for obtaining simultaneous upper confidence limits on the differences between $N + 1$ treatment means and the unknown best mean.

Suppose that V is distributed independently of $X_1, X_2, \ldots, X_N$ as a χ^2 random variable with ν degrees of freedom and define $T_I = (X_I + \delta_I)/\sqrt{(V/\nu)}$ for $I = 1, \ldots, N$ where the δ_I are constants. Then $T_1, T_2, \ldots, T_N$ have a joint non-central multivariate Student distribution with correlation matrix $\{\rho_{IJ}\}$, degrees of freedom ν and non-centrality vector $(\delta_1, \ldots, \delta_N)$. In certain applications, the value of

$$\text{PROB} = \Pr[B(I) \leqslant T_I \leqslant A(I);\, I = 1, \ldots, N] \tag{4}$$

may be required instead of equation (2). This can be determined by expressing it as

$$\text{PROB} = \int_0^{\infty} G_N\{yB(I) - \delta_I, yA(I) - \delta_I;\, I = 1, \ldots, N\} q_\nu(y)\,\mathrm{d}y \tag{5}$$

where $G_N\{B(I);\ A(I);\ I = 1, \ldots, N\}$ denotes the corresponding multivariate normal probability and

$$q_\nu(y) = 2(\nu/2)^{\nu/2} y^{\nu-1} \exp(-\nu y^2/2)/\Gamma(\nu/2)$$

is the density function of $\sqrt{(V/\nu)}$. This integral can be evaluated using an appropriate numerical integration routine, such as the International Mathematical and Statistical Libraries' (1987a) QDAGI, employing either the present algorithm or that of Schervish to evaluate the function in the integrand depending on whether or not the product correlation structure holds. Anyone desiring a subroutine of 60 lines developed by the author for computing multivariate Student probability integrals given by equation (4) using QDAGI may obtain a listing of it from the author.

Numerical Method

The random variables $X_1, X_2, \ldots, X_N$ can be expressed in terms of $N + 1$ independent standard normal variates $Y_1, \ldots, Y_N, Z$ by setting $X_I = \sqrt{(1 - b_I^2)}\, Y_I + b_I Z$; $I = 1, 2, \ldots, N$. This enables the probability (2) to be written as a single integral; see Dunnett and Sobel (1955) or Curnow and Dunnett (1962). The resulting integral may be written

$$\text{PROB} = \frac{1}{\sqrt{\pi}} \int_0^\infty \left[\prod_{I=1}^{N} \left\{ \Phi\left(\frac{A(I) - \sqrt{2} b_I z}{\sqrt{(1 - b_I^2)}} \right) - \Phi\left(\frac{B(I) - \sqrt{2} b_I z}{\sqrt{(1 - b_I^2)}} \right) \right\} + \prod_{I=1}^{N} \left\{ \Phi\left(\frac{A(I) + \sqrt{2} b_I z}{\sqrt{(1 - b_I^2)}} \right) - \Phi\left(\frac{B(I) + \sqrt{2} b_I z}{\sqrt{(1 - b_I^2)}} \right) \right\} \right] \exp(-z^2)\, \mathrm{d}z, \tag{6}$$

where $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standardized univariate normal distribution. MVNPRD uses Simpson's rule to compute an approximation to equation (6), in such a way that a prescribed accuracy EPS is achieved.

To approximate the integral of a function $f(z)$ over an interval $[a, b]$ using Simpson's rule, the value of the function is computed at the two end points and at the midpoint of the interval. The approximate value of the integral is given by

$$\left\{ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right\} \frac{b-a}{6} \tag{7}$$

and a bound on its error is

$$f_4(a, b)(b - a)^5/2880, \tag{8}$$

where $f_4(a, b)$ is a bound on the absolute value of the fourth derivative of $f(z)$ over the interval: for example, see Shampine and Allen (1973).

MVNPRD computes these for the function in the integrand of equation (6) over intervals of initial width HINC, which are subdivided into smaller intervals if necessary so that the sum of their error bounds does not exceed a determined amount EL for each interval. The value of EL is chosen for each interval so that the sum of the error bounds over all the intervals of width HINC up to a maximum value z_u of z, added to a bound on the error arising from neglecting values z greater than z_u, is less than EPS.

The prescribed accuracy bound EPS is apportioned into an amount 0.1EPS to bound the error from neglecting the upper tail and the remaining amount 0.9EPS which is divided according to the number of intervals of width HINC that there are between zero and z_u. The number of these intervals determines the initial value for EL but, to decrease the amount of computing necessary, the value of EL is increased for each interval after the first by the amount of the excess of the value EL assigned to the previous interval over the error bound actually achieved.

Other values were tried for the constant 0.1 used to apportion EPS but there was little effect on the timing and accuracy of the computations over a range of values around the value chosen.

The values of the derivatives of the two product expressions in the integrand of equation (6) are denoted by FOU1 and GOU1 respectively in the program. Over each interval or subinterval, the bound on the absolute value of the fourth derivative of the function is approximated by fitting a quadratic function to the fourth derivative values at the midpoint and two end points and by taking the largest absolute value over the interval. This results in the computed bounds being approximate rather than mathematically guaranteed, but numerical computations indicate that the approximate bounds are usually very near to the correct bounds based on the actual maximum of the fourth derivative over the intervals (see the example displayed in Table 3, later).

The width HINC of the initial Simpson's rule intervals may be chosen by the user, or the value 0.0 entered which will invoke a default value. The default value chosen is 0.24 which was found to be a satisfactory compromise between a very small value which would entail a large number of computations and a large value for which the quadratic approximation for the maximum absolute value of the fourth derivative might be in question. If any of the values of b_I are near ± 1, the user is advised to enter a smaller value for HINC than its default value.

There is an option to omit the computation of the fourth derivatives. The reason for this is that a large part of the total computing time is devoted to this and it can become excessive for values of N greater than 8–10. The user forgoes strict control over the error when this option is exercised. However, a bound is determined by computing for each interval or subinterval the value obtained by halving the width of the interval and applying Simpson's rule to each half: the absolute value of the difference between the result for the whole interval and the sum of the results for the two halves is used as an intuitive bound.

The case where some $b_I = 0$, which means that these variables have zero correlation with the other variables, is computed by factoring out the contributions of these variables to PROB and computing them as univariate normal integrals. Thus, MVNPRD uses numerical integration only for the variables that have non-zero values of b_I.

Structure

The structure was chosen to be as close as possible to that used by Schervish (1984) for his algorithm MULNOR and is as follows.

SUBROUTINE MVNPRD(A, B, BPD, EPS, N, INF, IERC, HINC, PROB, BOUND, IFAULT)

Formal parameters

A	Real array (N)	input:	the upper limits of integration
B	Real array (N)	input:	the lower limits of integration
BPD	Real array (N)	input:	the values of b_I defining the correlation structure
EPS	Real	input:	the desired accuracy
N	Integer	input:	the number of dimensions (the maximum value is 50; to increase it, change the values of NN in the two parameter statements in MVNPRD and PFUNC)
INF	Integer array (N)	input:	INF(I) = 0 if the Ith range of integration is $(B(I), \infty)$; INF(I) = 1 if the Ith range of integration is $(-\infty, A(I))$; INF(I) = 2 if the Ith range of integration is $(B(I), A(I))$
IERC	Integer	input:	method of error control: IERC = 1 if strict error control based on the fourth derivative is to be used; IERC = 0 if intuitive error control based on halving the intervals is to be used
HINC	Real	input:	interval width for Simpson's rule (enter

			HINC = 0.0 to invoke the default value 0.24)
PROB	Real	output:	the approximation to the N-variate probability
BOUND	Real	output:	the bound on the actual error of the approximation: if IERC = 1, it is an actual bound based on the fourth derivative; if IERC = 0, it is an intuitive bound based on halving each interval or subinterval
IFAULT	Integer	output:	a fault indicator: $=1$ if $N < 1$ or $N > 50$; $=2$ if any BPD(I) $\geqslant 1$ or BPD(I) $\leqslant -1$; $=3$ if any INF(I) $\neq 0$, 1 or 2; $=4$ if any INF(I) $= 2$ and $A(I) \leqslant B(I)$; $=5$ if the number of terms computed exceeds the upper limit, set at 400 if IERC = 1 and 800 if IERC = 0; $=6$ if a fault occurs in normal subroutines; $=7$ if subintervals are too narrow or too many; $=8$ if BOUND exceeds EPS; $=0$ otherwise

Auxiliary Algorithms

Function PFUNC calculates the value of the function in the integrand of equation (6) and the value of its fourth derivative, and is provided as part of the algorithm. Also provided is a subroutine ASSIGN which calculates the derivatives of the univariate normal CDFs shown in equation (6) and a function WMAX which determines the largest absolute value of a quadratic function fitted to three points. Additional functions needed, which must be provided by the user, are algorithms to compute the lower tail area of the standard normal distribution and to provide the value X such that the lower tail area of the standard normal is P. The functions ALNORM of Hill (1973) and PPND7 of Wichura (1988) are used here, but these may be replaced by S15ABF and G01CEF from the Numerical Algorithms Group's (1983) subroutine library or by ANORDF and ANORIN from the International Mathematical and Statistical Libraries' (1987b) subroutine library, if one of these libraries is available to the user.

Accuracy and Timing

Table 1 shows comparative accuracies and timings of the two versions of MVNPRD, i.e. with and without strict error control, with those obtained using Schervish's MULNOR and with known or published values. A VAX 8600 computer system at McMaster University using single-precision arithmetic was used in the timing comparisons. The program has also been run on a CDC Cyber 170 computer at McMaster University and on a Prime 750 computer at University College of

TABLE 1

Comparison of MVNPRD with MULNOR and with known or published results for cases of equal correlation ρ and identical ranges of integration

	End points				*Known or*	*MVNPRD, with IERC = 0*			*MVNPRD, with IERC = 1*				*MULNOR*	
N	*B(I)*	*A(I)*	*ρ*	*EPS*	*published result*†	*PROB*	*BOUND*	*TIME*‡	*PROB*	*BOUND*	*TIME*‡	*PROB*	*BOUND*	*TIME*‡
3	−5.0	5.0	0.5	10^{-4}	1.0	0.999 952	0.000 058	0.01	0.999 952	0.000 063	0.01	0.999 974	0.000 093	0.33
3	0.0	∞	0.5	10^{-4}	0.25	0.250 016	0.000 047	0.01	0.250 022	0.000 048	0.01	0.249 998	0.000 082	0.02
3	0.0	∞	$\frac{1}{3}$	10^{-4}	0.206 130	0.206 164	0.000 060	0.01	0.206 164	0.000 065	0.01	0.206 133	0.000 086	0.02
3	−2.0	2.0	0.9	10^{-4}	Unknown	0.923 402	0.000 031	0.01	0.923 402	0.000 040	0.01	0.923 401	0.000 052	0.44
3	−∞	2.0	0.9	10^{-4}	0.961 700 68	0.961 701	0.000 043	0.01	0.961 719	0.000 044	0.01	0.961 696	0.000 090	0.06
3	−∞	2.0	0.5	10^{-4}	0.942 533 45	0.942 493	0.000 086	0.01	0.942 492	0.000 089	0.01	0.942 526	0.000 091	0.04
4	0.0	∞	0.5	10^{-4}	0.2	0.200 022	0.000 059	0.01	0.200 027	0.000 061	0.02	0.199 998	0.000 080	0.83
4	−∞	2.0	0.5	10^{-4}	0.928 450 60	0.928 436	0.000 040	0.01	0.928 436	0.000 044	0.02	0.928 446	0.000 090	2.37
6	−∞	2.0	0.5	10^{-5}	0.904 421 00	0.904 421	0.000 005	0.01	0.904 421	0.000 005	0.13	§	§	§
9	−∞	2.0	0.5	10^{-5}	0.875 305 98	0.875 306	0.000 003	0.02	0.875 308	0.000 005	0.92	§	§	§
14	−∞	2.0	0.5	10^{-5}	0.837 872 27	0.837 872	0.000 005	0.02	0.837 874	0.000 005	8.13	§	§	§
24	−∞	2.0	0.5	10^{-5}	0.784 872 59	0.784 872	0.000 003	0.03	§	§	§	§	§	§

†The result is known to be $1/(N+1)$ for end points $(0, \infty)$ and $\rho = 0.5$; the result for $N = 3$, $\rho = \frac{1}{3}$ and end points $(0, \infty)$ is given in Moran (1956); other numerical results shown are from tables by Milton (1963).

‡Times shown are in seconds of central processor unit time on a VAX 8600 computer using single-precision arithmetic.

§Not computed owing to the excessive computing time required.

Swansea, as well as on an IBM PC microcomputer using the Microsoft Fortran compiler.

All numerical results shown in Table 1 are for equal correlations and identical limits of integration for each variable. Slightly longer computing times are required by MVNPRD for unequal correlations or different limits of integration since then the product terms in the integrand of equation (6) will need to be recomputed for each value of I.

Table 2 shows the accuracy of MVNPRD for an unequal correlation case with limits of integration $(0, \infty)$ for $N = 3$, where an exact expression is available (see Moran (1956)) for the probability integral, for a range of values for EPS.

Table 3 shows a particular example, giving the individual subintervals and the error bounds computed for each. The subintervals where the fourth derivative is not correctly given by the quadratic approximation fitted to the three values computed by

TABLE 2

Values of PROB and BOUND obtained with MVNPRD for an unequal correlation case†

EPS	*MVNPRD, with IERC = 0*			*MVNPRD, with IERC = 1*		
	PROB	*BOUND*	*True error*‡	*PROB*	*BOUND*	*True error*‡
10^{-4}	0.223 702 60	0.48×10^{-4}	0.42×10^{-4}	0.223 702 65	0.54×10^{-4}	0.42×10^{-4}
10^{-5}	0.223 668 88	0.92×10^{-5}	0.81×10^{-5}	0.223 661 48	0.45×10^{-5}	0.07×10^{-5}
10^{-6}	0.223 661 00	0.67×10^{-6}	0.19×10^{-6}	0.223 660 93	0.75×10^{-6}	0.12×10^{-6}
10^{-7}	0.223 660 84	0.76×10^{-7}	0.29×10^{-7}	0.223 660 81	0.81×10^{-7}	0.06×10^{-7}

† $N = 3$, $\rho_{12} = 0.5$, $\rho_{13} = 0.4$, $\rho_{23} = 0.3$ (or $b_1 = \sqrt{6/3}$, $b_2 = \sqrt{6/4}$, $b_3 = \sqrt{6/5}$) and end points $(0.0, \infty)$.
‡ Exact result 0.223 660 81, computed from $\frac{1}{2} - (\cos^{-1}\rho_{12} + \cos^{-1}\rho_{13} + \cos^{-1}\rho_{23})/4\pi$ (Moran, 1956).

TABLE 3

Detailed results for a particular example ($N = 3$, $\rho = 0.9$, end points $(-2.0, 2.0)$ and EPS = 0.0001)

Interval	*Contribution to PROB*	*True error*	*Computed bound*	*Correct bound, if different*†
0.00–0.24	0.265 703 51	0.000 003 45	0.000 003 74	–
0.24–0.48	0.237 049 92	0.000 001 57	0.000 002 73	–
0.48–0.72	0.188 608 69	−0.000 002 56	0.000 005 76	–
0.72–0.96	0.132 528 16	−0.000 005 00	0.000 007 24	0.000 006 90
0.96–1.08	0.004 612 31	0.000 000 54	0.000 000 80	–
1.08–1.20	0.029 890 98	0.000 000 72	0.000 000 83	0.000 000 82
1.20–1.32	0.016 346 61	−0.000 000 61	0.000 001 31	–
1.32–1.44	0.006 581 63	−0.000 001 37	0.000 001 41	0.000 001 45
1.44–1.56	0.001 761 66	−0.000 000 05	0.000 000 84	–
1.56–1.68	0.000 287 38	0.000 000 59	0.000 000 61	0.000 000 62
1.68–1.92	0.000 031 25	0.000 003 47	0.000 014 19	–
1.92–∞	0.000 000 00	−0.000 000 03	0.000 000 27	–
	0.923 402 10	0.000 000 73	0.000 039 75	0.000 039 42

† Correct bound calculated from the actual maximum instead of the quadratic approximation to the maximum of the fourth derivative.

the algorithm are indicated, along with the correct values for the bounds based on the actual maximum of the fourth derivative.

Acknowledgements

This algorithm had its origins in a program developed many years ago by the author when he was employed by Lederle Laboratories in Pearl River, USA. The author is greatly indebted to that organization for their support of his work.

The author is also indebted to McMaster University and to University College of Swansea for generously providing time on their computers, which initially were a CDC Cyber 170 at McMaster and a Prime 750 at Swansea, later replaced by VAX computers in each location, and assistance during the development of this program. Special thanks are due to K. A. Redish whose suggestions greatly improved the efficiency of the algorithm. Financial support was provided by a research grant from the Natural Sciences and Engineering Research Council of Canada. The author is grateful to a referee and the editors for suggestions made on earlier versions of the algorithm.

References

Curnow, R. N. and Dunnett, C. W. (1962) The numerical evaluation of certain multivariate normal integrals. *Ann. Math. Statist.*, **33**, 571–579.

Dunnett, C. W. (1955) A multiple comparisons procedure for comparing several treatments with a control. *J. Amer. Statist. Ass.*, **50**, 1096–1121.

Dunnett, C. W. and Sobel, M. (1955) Approximations to the probability integral and certain percentage points of a multivariate analogue of Student's *t*-distribution. *Biometrika*, **42**, 258–260.

Hill, I. D. (1973) Algorithm AS 66: The normal integral. *Appl. Statist.*, **22**, 424–427.

Hsu, J. C. (1984) Ranking and selection and multiple comparisons with the best. In *Design of Experiments: Ranking and Selection* (eds T. J. Santner and A. C. Tamhane), pp. 23–33. New York: Dekker.

International Mathematical and Statistical Libraries (1987a) *MATH/Library, Fortran Subroutines for Mathematical Applications*, vol. 1, ch. 4. Houston: International Mathematical and Statistical Libraries.

——(1987b) *STAT/Library, Fortran Subroutines for Statistical Analysis*, vol. 1, ch. 17. Houston: International Mathematical and Statistical Libraries.

Milton, R. C. (1963) Tables of the equally correlated multivariate normal probability integral. *Technical Report 27*. University of Minnesota, Minneapolis.

Moran, P. A. P. (1956) The numerical evaluation of a class of integrals. *Proc. Camb. Philos. Soc.*, **52**, 230–233.

Numerical Algorithms Group (1983) *NAG Fortran Library Manual, Mark 10*. Oxford: Numerical Algorithms Group.

Schervish, M. J. (1984) Algorithm AS 195: Multivariate normal probabilities with error bound. *Appl. Statist.*, **33**, 81–94; correction, **34** (1985), 103–104.

Shampine, L. F. and Allen, R. C. (1973) *Numerical Computing: an Introduction*, p. 66. Philadelphia: Saunders.

Wichura, M. J. (1988) Algorithm AS 241: The percentage points of the normal distribution. *Appl. Statist.*, **37**, 477–484.

```
      SUBROUTINE MVNPRD(A, B, BPD, EPS, N, INF, IERC, HINC, PROB, BOUND,
     *                  IFAULT)
C
C        ALGORITHM AS 251.1  APPL.STATIST. (1989), VOL.38, NO.3
C
C        FOR A MULTIVARIATE NORMAL VECTOR WITH CORRELATION STRUCTURE
C        DEFINED BY RHO(I,J) = BPD(I) * BPD(J), COMPUTES THE PROBABILITY
C        THAT THE VECTOR FALLS IN A RECTANGLE IN N-SPACE WITH ERROR
C        LESS THAN EPS.
C
```

```
      INTEGER NN
      PARAMETER (NN=50)
      REAL A( * ), B( * ), BPD( * ), EPS, HINC, PROB, BOUND
      INTEGER INF( * ),  N, IERC, IFAULT
C
      REAL ESTT(22), FV(5), FD(5), F1T(22), F2T(22), F3T(22), G1T(22),
     *     G3T(22), PSUM(22), H(NN), HL(NN), BB(NN) ,ZERO, HALF, ONE,
     *     TWO, FOUR, SIX, PT1, PT24, ONEP5, X2880, SMALL, DXMIN, SQRT2,
     *     ERRL, BI, START, Z, ADDN, EPS2, EPS1, ZU, Z2, Z3, Z4, Z5, ZZ,
     *     ERFAC, EL, EL1, PART0, PART2, PART3, FUNC0, FUNC2, FUNCN, WT,
     *     CONTRB, DLG, DX, DA, ESTL, ESTR, SUM, EXCESS, ERROR, PROB1,
     *     SAFE
      INTEGER INFT(NN), LDIR(22), I, NTM, NMAX, LVL, NR, NDIM
C
      REAL ALNORM, PPND7
      EXTERNAL ALNORM, PPND7
      DATA ZERO, HALF, ONE, TWO, FOUR, SIX / 0.0, 0.5, 1.0, 2.0, 4.0,
     *     6.0 /
      DATA PT1, PT24, ONEP5, X2880 / 0.1, 0.24, 1.5, 2880.0 /
      DATA SMALL, DXMIN, SQRT2 / 1.0E-10, 0.0000001, 1.41421356237310 /
C
C        CHECK FOR INPUT VALUES OUT OF RANGE.
C
      PROB = ZERO
      BOUND = ZERO
      IFAULT = 1
      IF (N .LT. 1 .OR. N .GT. NN) RETURN
      DO 10 I = 1, N
         BI = ABS(BPD(I))
         IFAULT = 2
         IF (BI .GE. ONE) RETURN
         IFAULT = 3
         IF (INF(I) .LT. 0 .OR. INF(I) .GT. 2) RETURN
         IFAULT = 4
         IF (INF(I) .EQ. 2 .AND. A(I) .LE. B(I)) RETURN
   10 CONTINUE
      IFAULT = 0
      PROB = ONE
C
C        CHECK WHETHER ANY BPD(I) = 0.
C
      NDIM = 0
      DO 20 I = 1, N
         IF (BPD(I) .NE. ZERO) THEN
            NDIM = NDIM + 1
            H(NDIM) = A(I)
            HL(NDIM) = B(I)
            BB(NDIM) = BPD(I)
            INFT(NDIM) = INF(I)
         ELSE
C
C        IF ANY BPD(I) = 0, THE CONTRIBUTION TO PROB FOR THAT
C        VARIABLE IS COMPUTED FROM A UNIVARIATE NORMAL.
C
            IF (INF(I) .LT. 1) THEN
               PROB = PROB * (ONE - ALNORM(B(I), .FALSE.))
            ELSE IF (INF(I) .EQ. 1) THEN
               PROB = PROB * ALNORM(A(I), .FALSE.)
            ELSE
               PROB = PROB * (ALNORM(A(I), .FALSE.) -
                      ALNORM(B(I), .FALSE.))
            END IF
            IF (PROB .LE. SMALL) PROB = ZERO
         END IF
   20 CONTINUE
      IF (NDIM .EQ. 0 .OR. PROB .EQ. ZERO) RETURN
C
C        IF NOT ALL BPD(I) = 0, PROB IS COMPUTED BY SIMPSON'S RULE.
```

```
C        BUT FIRST, INITIALIZE THE VARIABLES.
C
      Z = ZERO
      IF (HINC .LE. ZERO) HINC = PT24
      ADDN = -ONE
      DO 30 I = 1, NDIM
         IF (INFT(I) .EQ. 2 .OR. (INFT(I) .NE. INFT(1) .AND.
     *       BB(I) * BB(1) .GT. ZERO) .OR.
     *       (INFT(I) .EQ. INFT(1) .AND. BB(I) * BB(1) .LT.
     *       ZERO)) ADDN = ZERO
   30 CONTINUE
C
C        THE VALUE OF ADDN IS TO BE ADDED TO THE PRODUCT EXPRESSIONS IN
C        THE INTEGRAND TO INSURE THAT THE LIMITING VALUE IS ZERO.
C
      PROB1 = ZERO
      NTM = 0
      NMAX = 400
      IF (IERC .EQ. 0) NMAX = NMAX * 2
      CALL PFUNC(Z, H, HL, BB, NDIM, INFT, ADDN, SAFE, FUNC0, NTM, IERC,
     *           PART0)
      EPS2 = EPS * PT1 * HALF
C
C        SET UPPER BOUND ON Z AND APPORTION EPS.
C
      ZU = -PPND7(EPS2, IFAULT) / SQRT2
      IF (IFAULT .NE. 0) THEN
         IFAULT = 6
         RETURN
      END IF
      NR = IFIX(ZU / HINC) + 1
      ERFAC = ONE
      IF (IERC .NE. 0) ERFAC = X2880 / HINC ** 5
      EL = (EPS - EPS2) / FLOAT(NR) * ERFAC
      EL1 = EL
C
C        START COMPUTATIONS FOR THE INTERVAL (Z, Z + HINC).
C
   40 ERROR = ZERO
      LVL = 0
      FV(1) = PART0
      FD(1) = SAFE
      START = Z
      DA = HINC
      Z3 = START + HALF * DA
      CALL PFUNC(Z3, H, HL, BB, NDIM, INFT, ADDN, FD(3), FUNCN, NTM,
     *           IERC, FV(3))
      Z5 = START + DA
      CALL PFUNC(Z5, H, HL, BB, NDIM, INFT, ADDN, FD(5), FUNC2, NTM,
     *           IERC, FV(5))
      PART2 = FV(5)
      SAFE = FD(5)
      WT = DA / SIX
      CONTRB = WT * (FV(1) + FOUR * FV(3) + FV(5))
      DLG = ZERO
      IF (IERC .NE. 0) THEN
         CALL WMAX(FD(1), FD(3), FD(5), DLG)
         IF (DLG .LE. EL) GO TO 90
         DX = DA
         GO TO 60
      END IF
      LVL = 1
      LDIR(LVL) = 2
      PSUM(LVL) = ZERO
C
C        BISECT INTERVAL.  IF IERC = 1, COMPUTE ESTIMATE ON LEFT
C        HALF.  IF IERC = 0, ON BOTH HALVES.
C
```

```
   50 DX = HALF * DA
      WT = DX / SIX
      Z2 = START + HALF * DX
      CALL PFUNC(Z2, H, HL, BB, NDIM, INFT, ADDN, FD(2), FUNCN, NTM,
     *           IERC, FV(2))
      ESTL = WT * (FV(1) + FOUR * FV(2) + FV(3))
      IF (IERC .EQ. 0) THEN
         Z4 = START + ONEP5 * DX
         CALL PFUNC(Z4, H, HL, BB, NDIM, INFT, ADDN, FD(4), FUNCN, NTM,
     *              IERC, FV(4))
         ESTR = WT * (FV(3) + FOUR * FV(4) + FV(5))
         SUM = ESTL + ESTR
         DLG = ABS(CONTRB - SUM)
         EPS1 = EL / TWO ** (LVL - 1)
         ERRL = DLG
      ELSE
         FV(3) = FV(2)
         FD(3) = FD(2)
         CALL WMAX(FD(1), FD(3), FD(5), DLG)
         ERRL = DLG / TWO ** (5 * LVL)
         SUM = ESTL
         EPS1 = EL * (TWO ** LVL) ** 4
      END IF
C
C        STOP SUBDIVIDING INTERVAL WHEN ACCURACY IS SUFFICIENT,
C        OR IF INTERVAL TOO NARROW OR SUBDIVIDED TOO OFTEN.
C
      IF (DLG .LE. EPS1 .OR. DLG .LT. SMALL) GO TO 70
      IF (IFAULT .EQ. 0 .AND. NTM .GE. NMAX) IFAULT = 5
      IF (ABS(DX) .LE. DXMIN .OR. LVL .GT. 21) IFAULT = 7
      IF (IFAULT .NE. 0) GO TO 70
C
C        RAISE LEVEL.  STORE INFORMATION FOR RIGHT HALF AND APPLY
C        SIMPSON'S RULE TO LEFT HALF.
C
   60 LVL = LVL + 1
      LDIR(LVL) = 1
      F1T(LVL) = FV(3)
      F3T(LVL) = FV(5)
      DA = DX
      FV(5) = FV(3)
      IF (IERC .EQ. 0) THEN
         F2T(LVL) = FV(4)
         ESTT(LVL) = ESTR
         CONTRB = ESTL
         FV(3) = FV(2)
      ELSE
         G1T(LVL) = FD(3)
         G3T(LVL) = FD(5)
         FD(5) = FD(3)
      END IF
      GO TO 50
C
C        ACCEPT APPROXIMATE VALUE FOR INTERVAL.
C        RESTORE SAVED INFORMATION TO PROCESS
C        RIGHT HALF INTERVAL.
C
   70 ERROR = ERROR + ERRL
   80 IF (LDIR(LVL) .EQ. 1) THEN
         PSUM(LVL) = SUM
         LDIR(LVL) = 2
         IF (IERC .EQ. 0) DX = DX * TWO
         START = START + DX
         DA = HINC / TWO ** (LVL - 1)
         FV(1) = F1T(LVL)
         IF (IERC .EQ. 0) THEN
            FV(3) = F2T(LVL)
            CONTRB = ESTT(LVL)
         ELSE
```

```
            FV(3) = F3T(LVL)
            FD(1) = G1T(LVL)
            FD(5) = G3T(LVL)
         END IF
         FV(5) = F3T(LVL)
         GO TO 50
      END IF
      SUM = SUM + PSUM(LVL)
      LVL = LVL - 1
      IF (LVL .GT. 0) GO TO 80
      CONTRB = SUM
      LVL = 1
      DLG = ERROR
   90 PROB1 = PROB1 + CONTRB
      BOUND = BOUND + DLG
      EXCESS = EL - DLG
      EL = EL1
      IF (EXCESS .GT. ZERO) EL = EL1 + EXCESS
      IF ((FUNC0 .GT. ZERO .AND. FUNC2 .LE. FUNC0) .OR.
     *    (FUNC0 .LT. ZERO .AND. FUNC2 .GE. FUNC0)) THEN
         ZZ = -SQRT2 * Z5
         PART3 = ABS(FUNC2) * ALNORM(ZZ, .FALSE.) + BOUND / ERFAC
         IF (PART3 .LE. EPS .OR. NTM .GE. NMAX .OR.
     *       Z5 .GE. ZU) GO TO 100
      END IF
      Z = Z5
      PART0 = PART2
      FUNC0 = FUNC2
      GO TO 40
  100 PROB = (PROB1 - ADDN * HALF) * PROB
      BOUND = PART3
      IF (NTM .GE. NMAX .AND. IFAULT .EQ. 0) IFAULT = 5
      IF (BOUND .GT. EPS .AND. IFAULT .EQ. 0) IFAULT = 8
      RETURN
      END
      SUBROUTINE PFUNC(Z, A, B, BPD, N, INF, ADDN, DERIV, FUNCN, NTM,
     *                 IERC, RESULT)
C
C        ALGORITHM AS 251.2  APPL.STATIST. (1989), VOL.38, NO.3
C
C        COMPUTE FUNCTION IN INTEGRAND AND ITS 4TH DERIVATIVE.
C
      INTEGER NN
      PARAMETER (NN=50)
      REAL A( * ), B( * ), BPD( * ), Z, ADDN, DERIV, FUNCN, RESULT
      INTEGER INF( * ), N, NTM, IERC
C
      REAL FOU(NN), FOU1(4, NN), TMP(4), GOU(NN), GOU1(4, NN), FF(4),
     *    GF(4), TERM(4), GERM(4), ZERO, ONE, TWO, THREE, FOUR, SIX,
     *    EIGHT, TWELVE, SIXTN, SMALL, U, U1, U2, BI, HI, HLI, BP,
     *    RSLT1, RSLT2, DEN, SQRT2, SQRTPI, PHI, PHI1, PHI2, PHI3, PHI4,
     *    FRM, GRM
      INTEGER INFI, I, J, K, M, L, IK
C
      REAL ALNORM
      EXTERNAL ALNORM
      DATA ZERO, ONE, TWO, THREE, FOUR, SIX, EIGHT, TWELVE, SIXTN,
     *     SMALL / 0.0, 1.0, 2.0, 3.0, 4.0, 6.0, 8.0, 12.0, 16.0,
     *     0.1E-12 /
      DATA SQRT2, SQRTPI / 1.41421356237310, 1.77245385090552 /
C
      DERIV = ZERO
      NTM = NTM + 1
      RSLT1 = ONE
      RSLT2 = ONE
      BI = ONE
      HI = A(1) + ONE
      HLI = B(1) + ONE
```

```
      INFI = -1
      DO 60 I = 1, N
         IF (BPD(I) .EQ. BI .AND. A(I) .EQ. HI .AND. B(I) .EQ. HLI .AND.
     *       INF(I) .EQ. INFI) THEN
            FOU(I) = FOU(I - 1)
            GOU(I) = GOU(I - 1)
            DO 10 IK = 1, 4
               FOU1(IK, I) = FOU1(IK, I - 1)
               GOU1(IK, I) = GOU1(IK, I - 1)
   10       CONTINUE
         ELSE
            BI = BPD(I)
            HI = A(I)
            HLI = B(I)
            INFI = INF(I)
            IF (BI .EQ. ZERO) THEN
               IF (INFI .LT. 1) THEN
                  FOU(I) = ONE - ALNORM(HLI, .FALSE.)
               ELSE IF (INFI .EQ. 1) THEN
                  FOU(I) = ALNORM(HI, .FALSE.)
               ELSE
                  FOU(I) = ALNORM(HI, .FALSE.) - ALNORM(HLI, .FALSE.)
               END IF
               GOU(I) = FOU(I)
               DO 20 IK = 1, 4
                  FOU1(IK, I) = ZERO
                  GOU1(IK, I) = ZERO
   20          CONTINUE
            ELSE
               DEN = SQRT(ONE - BI * BI)
               BP = BI * SQRT2 / DEN
               IF (INFI .LT. 1) THEN
                  U = -HLI / DEN + Z * BP
                  FOU(I) = ALNORM(U, .FALSE.)
                  CALL ASSIGN(U, BP, FOU1(1, I))
                  BP = -BP
                  U = -HLI / DEN + Z * BP
                  GOU(I) = ALNORM(U, .FALSE.)
                  CALL ASSIGN(U, BP, GOU1(1, I))
               ELSE IF (INFI .EQ. 1) THEN
                  U = HI / DEN + Z * BP
                  GOU(I) = ALNORM(U, .FALSE.)
                  CALL ASSIGN(U, BP, GOU1(1, I))
                  BP = -BP
                  U = HI / DEN + Z * BP
                  FOU(I) = ALNORM(U, .FALSE.)
                  CALL ASSIGN(U, BP, FOU1(1, I))
               ELSE
                  U2 = -HLI / DEN + Z * BP
                  CALL ASSIGN(U2, BP, FOU1(1, I))
                  BP = -BP
                  U1 = HI / DEN + Z * BP
                  CALL ASSIGN(U1, BP, TMP(1))
                  FOU(I) = ALNORM(U1, .FALSE.) + ALNORM(U2, .FALSE.) -
     *                     ONE
                  DO 30 IK = 1, 4
                     FOU1(IK, I) = FOU1(IK, I) + TMP(IK)
   30             CONTINUE
                  IF (-HLI .EQ. HI) THEN
                     GOU(I) = FOU(I)
                     DO 40 IK = 1, 4
                        GOU1(IK, I) = FOU1(IK, I)
   40                CONTINUE
                  ELSE
                     U2 = -HLI / DEN + Z * BP
                     CALL ASSIGN(U2, BP, GOU1(1, I))
                     BP = -BP
                     U1 = HI / DEN + Z * BP
```

```
                 GOU(I) = ALNORM(U1, .FALSE.) +
     *                    ALNORM(U2, .FALSE.) - ONE
                 CALL ASSIGN(U1, BP, TMP(1))
                 DO 50 IK = 1, 4
                    GOU1(IK, I) = GOU1(IK, I) + TMP(IK)
   50            CONTINUE
              END IF
           END IF
        END IF
     END IF
     RSLT1 = RSLT1 * FOU(I)
     RSLT2 = RSLT2 * GOU(I)
     IF (RSLT1 .LE. SMALL) RSLT1 = ZERO
     IF (RSLT2 .LE. SMALL) RSLT2 = ZERO
   60 CONTINUE
     FUNCN = RSLT1 + RSLT2 + ADDN
     RESULT = FUNCN * EXP(-Z * Z) / SQRTPI
C
C        IF 4TH DERIVATIVE IS NOT WANTED, STOP HERE.
C        OTHERWISE, PROCEED TO COMPUTE 4TH DERIVATIVE.
C
      IF (IERC .EQ. 0) RETURN
      DO 70 IK = 1, 4
         FF(IK) = ZERO
         GF(IK) = ZERO
   70 CONTINUE
      DO 100 I = 1, N
         FRM = ONE
         GRM = ONE
         DO 80 J = 1, N
            IF (J .EQ. 1) GO TO 80
            FRM = FRM * FOU(J)
            GRM = GRM * GOU(J)
            IF (FRM .LE. SMALL) FRM = ZERO
            IF (GRM .LE. SMALL) GRM = ZERO
   80    CONTINUE
         DO 90 IK = 1, 4
            FF(IK) = FF(IK) + FRM * FOU1(IK, I)
            GF(IK) = GF(IK) + GRM * GOU1(IK, I)
   90    CONTINUE
  100 CONTINUE
      IF (N .LE. 2) GO TO 230
      DO 130 I = 1, N
         DO 120 J = I + 1, N
            TERM(2) = FOU1(1, I) * FOU1(1, J)
            GERM(2) = GOU1(1, I) * GOU1(1, J)
            TERM(3) = FOU1(2, I) * FOU1(1, J)
            GERM(3) = GOU1(2, I) * GOU1(1, J)
            TERM(4) = FOU1(3, I) * FOU1(1, J)
            GERM(4) = GOU1(3, I) * GOU1(1, J)
            TERM(1) = FOU1(2, I) * FOU1(2, J)
            GERM(1) = GOU1(2, I) * GOU1(2, J)
            DO 110 K = 1, N
               IF (K .EQ. I .OR. K .EQ. J) GO TO 110
               CALL TOOSML(1, TERM, FOU(K))
               CALL TOOSML(1, GERM, GOU(K))
  110       CONTINUE
            FF(2) = FF(2) + TWO * TERM(2)
            FF(3) = FF(3) + TWO * TERM(3) * THREE
            FF(4) = FF(4) + TWO * (TERM(4) * FOUR + TERM(1) * THREE)
            GF(2) = GF(2) + TWO * GERM(2)
            GF(3) = GF(3) + TWO * GERM(3) * THREE
            GF(4) = GF(4) + TWO * (GERM(4) * FOUR + GERM(1) * THREE)
  120    CONTINUE
  130 CONTINUE
      DO 170 I = 1, N
         DO 160 J = I + 1, N
            DO 150 K = J + 1, N
```

```
              TERM(3) = FOU1(1, I) * FOU1(1, J) * FOU1(1, K)
              TERM(4) = FOU1(2, I) * FOU1(1, J) * FOU1(1, K)
              GERM(3) = GOU1(1, I) * GOU1(1, J) * GOU1(1, K)
              GERM(4) = GOU1(2, I) * GOU1(1, J) * GOU1(1, K)
              IF (N .GT. 3) THEN
                 DO 140 M = 1, N
                    IF (M .EQ. I .OR. M .EQ. J .OR.
     *                  M .EQ. K) GO TO 140
                    CALL TOOSML(3, TERM, FOU(M))
                    CALL TOOSML(3, GERM, GOU(M))
  140            CONTINUE
              END IF
              FF(3) = FF(3) + SIX * TERM(3)
              FF(4) = FF(4) + SIX * TERM(4) * SIX
              GF(3) = GF(3) + SIX * GERM(3)
              GF(4) = GF(4) + SIX * GERM(4) * SIX
  150      CONTINUE
  160   CONTINUE
  170 CONTINUE
      IF (N .LE. 3) GO TO 230
      DO 220 I = 1, N
         DO 210 J = I + 1, N
            DO 200 K = J + 1, N
               DO 190 M = K + 1, N
                  TERM(4) = FOU1(1, I) * FOU1(1, J) * FOU1(1, K) *
     *                      FOU1(1, M)
                  GERM(4) = GOU1(1, I) * GOU1(1, J) * GOU1(1, K) *
     *                      GOU1(1, M)
                  IF (N .GT. 4) THEN
                     DO 180 L = 1, N
                        IF (L .EQ. I .OR. L .EQ. J .OR. L .EQ. K .OR.
     *                      L .EQ. M) GO TO 180
                        CALL TOOSML(4, TERM, FOU(L))
                        CALL TOOSML(4, GERM, GOU(L))
  180                CONTINUE
                  END IF
                  FF(4) = FF(4) + FOUR * SIX * TERM(4)
                  GF(4) = GF(4) + FOUR * SIX * GERM(4)
  190         CONTINUE
  200      CONTINUE
  210   CONTINUE
  220 CONTINUE
C
  230 CONTINUE
      PHI = EXP(-Z * Z) / SQRTPI
      PHI1 = -TWO * Z * PHI
      PHI2 = (FOUR * Z ** 2 - TWO) * PHI
      PHI3 = (-EIGHT * Z ** 3 + TWELVE * Z) * PHI
      PHI4 = (SIXTN * Z ** 2 * (Z ** 2 - THREE) + TWELVE) * PHI
      DERIV = PHI * (FF(4) + GF(4)) + FOUR * PHI1 * (FF(3) + GF(3)) +
     *        SIX * PHI2 * (FF(2) + GF(2)) +
     *        FOUR * PHI3 * (FF(1) + GF(1)) + PHI4 * FUNCN
      RETURN
      END
      SUBROUTINE ASSIGN(U, BP, FF)
C
C        ALGORITHM AS 251.3  APPL.STATIST. (1989), VOL.38, NO.3
C
C        COMPUTE DERIVATIVES OF NORMAL CDF'S.
C
      REAL FF(4), U, BP
      REAL U2, HALF, ONE, THREE, SQ2PI, T1, T2, T3, ZERO, UMAX, SMALL
      INTEGER I
C
      DATA HALF, ONE, THREE, SQ2PI / 0.5, 1.0, 3.0, 2.50662827463100 /
      DATA ZERO, UMAX, SMALL / 0.0, 8.0, 0.1E-07 /
      IF (ABS(U) .GT. UMAX) THEN
         DO 10 I = 1, 4
```

```
            FF(I) = ZERO
   10    CONTINUE
      ELSE
         U2 = U * U
         T1 = BP * EXP(-HALF * U2) / SQ2PI
         T2 = BP * T1
         T3 = BP * T2
         FF(1) = T1
         FF(2) = -U * T2
         FF(3) = (U2 - ONE) * T3
         FF(4) = (THREE - U2) * U * BP * T3
         DO 20 I = 1, 4
            IF (ABS(FF(I)) .LT. SMALL) FF(I) = ZERO
   20    CONTINUE
      END IF
      RETURN
      END
      SUBROUTINE WMAX(W1, W2, W3, DLG)
C
C        ALGORITHM AS 251.4  APPL.STATIST. (1989), VOL.38, NO.3
C
C        LARGEST ABSOLUTE VALUE OF QUADRATIC FUNCTION FITTED
C        TO THREE POINTS.
C
      REAL W1, W2, W3, DLG
      REAL QUAD, QLIM, QMIN, ONE, TWO, B2C
      DATA ONE, TWO, QMIN / 1.0, 2.0, 0.00001 /
C
      DLG = MAX(ABS(W1), ABS(W3))
      QUAD = W1 - W2 * TWO + W3
      QLIM = MAX(ABS(W1 - W3) / TWO, QMIN)
      IF (ABS(QUAD) .LE. QLIM) RETURN
      B2C = (W1 - W3) / QUAD / TWO
      IF (ABS(B2C) .GE. ONE) RETURN
      DLG = MAX(DLG, ABS(W2 - B2C * QUAD * B2C / TWO))
      RETURN
      END
      SUBROUTINE TOOSML(N, FF, F)
C
C        ALGORITHM AS 251.5  APPL.STATIST. (1989), VOL.38, NO.3
C
C        MULTIPLY FF(I) BY F FOR I = N TO 4.  SET TO ZERO IF TOO SMALL.
C
      REAL FF(4), F
      INTEGER N
      REAL ZERO, SMALL
      INTEGER I
      DATA ZERO, SMALL / 0.0, 0.1E-12 /
C
      DO 10 I = N, 4
         FF(I) = FF(I) * F
         IF (ABS(FF(I)) .LE. SMALL) FF(I) = ZERO
   10 CONTINUE
      RETURN
      END
```