

Algorithms

J. G. HERRIOT, Editor

ALGORITHM 272 PROCEDURE FOR THE NORMAL DISTRIBUTION FUNCTIONS* [S15]

M. D. MACLAREN

(Recd. 28 July 1964, 17 Nov. 1964 and 26 July 1965)
Argonne National Laboratory, Argonne, Ill., and Boeing
Scientific Research Laboratories, Seattle, Wash.

* Work performed in part under the auspices of the US Atomic
Energy Commission.

real procedure *phi*(*a*, *k*); **value** *a*, *k*; **real** *a*; **integer** *k*;
comment Before use, this procedure must be called once with
k = 3 to initialize **own** variables. Thereafter for *k* = 1 the
procedure gives

$$\Phi(a) = \frac{1}{(2\pi)^{1/2}} \int_{-\infty}^a \exp(-t^2/2) dt,$$

and for *k* = 2 it gives

$$\begin{aligned} \Phi^*(a) &= 2(\Phi(|a|) - .5) \\ &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^{|a|} \exp(-t^2/2) dt; \end{aligned}$$

begin own integer *N*;

own real *B*, *EPS*, *EPS2*, *EPS3*, *ONE*, *DELTA*, *DELTA2*, *PI2*;

comment $\Phi^*(a)$ is computed by Taylor's series expansion in the
interval $0 \leq a \leq B$, and by asymptotic series in the interval
 $B < a$. The Taylor's series expansion is made about one of the
points 0, B/N , $2B/N$, \dots , B , and the coefficients in the series
are computed using the recursion formula for Hermite poly-
nomials. The number of terms to take in the series is deter-
mined by an error estimate based on a majorizing series. This
procedure, which is essentially the familiar one of interpolat-
ing in a stored table of values, gives a fast program and can be
used effectively for many functions. In this case another sig-
nificant increase in speed could be obtained by also storing a
table of values of the first derivative of Φ^* . The **own** variables
B, *EPS* and *N* might be called program parameters. By suit-
ably choosing their values the programmer may make the
procedure as accurate as desired and may increase the speed
of the procedure at the cost of extra storage space. This is the
advantage of this procedure over others previously published
in this journal (see [1-4]).

The values of these program parameters are determined
when the procedure is coded, not when it is called. They are
set by means of an initializing call with *k* = 3. The other **own**
variables are computed from *B*, *EPS* and *N* when the initializ-
ing call is made. If FORTRAN IV were used, all the **own** vari-
ables could be set by use of a DATA statement. An alternative
to either method is to replace all occurrences of the parameters
by the appropriate constants.

The choice of the parameter *N* depends mainly on speed
versus storage considerations. The larger *N* is, the faster the
procedure will be and the more storage will be needed. Note,
however, that *N* must be chosen large enough so that
 $B^2(1/(2N) + 1/(4N^2)) \leq 1$, for otherwise the method of estimat-
ing the error in the Taylor's series may fail. The choice of *B*
may also affect the speed, because for smaller values of *a* the

asymptotic series for $\Phi^*(a)$ will take longer than the Taylor's
series. The choice of *B* depends, however, mainly on the error
desired. Neglecting roundoff, the maximum error in the com-
puted value of $\Phi^*(a)$ will be *EPS* if $a \leq B$ or $\max(EPS, \delta(a)/2)$
if $B < a$, where $\delta(a)$ is the absolute value of the smallest term
in the asymptotic series for $\Phi^*(a)$. Some values of $\delta(a)$ are:
 $\delta(4) = 3.0 \times 10^{-8}$, $\delta(5) = 3.0 \times 10^{-12}$, $\delta(5.5) = 1.4 \times 10^{-14}$, and
 $\delta(6) = 4.4 \times 10^{-17}$. If *N* is large enough, roundoff will be no
problem. (The referee has pointed out that the computation
for $B < a$ could be made by continued fractions, as in Algo-
rithm 180. The advantage of this would be that the continued
fraction expansion converges for all $a > 0$, but roundoff errors
may be significant for smaller values of *a*.)

With the program parameters having the values given
below, the procedure was compiled as a FORTRAN II subroutine
on the IBM 1620, using eight-digit floating point arithmetic,
and tested for many values of *a*. The error never exceeded
 2×10^{-8} . The program was also compiled with *B* = 6.0, *EPS* =
 2×10^{-15} and *N* = 60, using 15 digit arithmetic. Spot checks
turned up no errors greater than 2×10^{-15} ;

own real array *C*[0:16];

comment The array *C* must give the value of $\Phi^*(a)$ at the
point of expansion, i.e., *C*[*m*] must equal $\Phi^*(mB/N)$. Tables of
 $\Phi^*(a)$ to fifteen decimal places are published by the National
Bureau of Standards [5]. The upper bound for the array must
equal the value of the program parameter *N*;

real *f*, *f1*, *f2*, *x*, *y*, *z*, *t*, *t2*, *xt*;

integer *m*;

real procedure *max*(*x*, *y*); **value** *x*, *y*; **real** *x*, *y*;

begin *max* := **if** *x* ≤ *y* **then** *y* **else** *x*;

end *max*;

if *k* = 3 **then**

begin comment initialize **own** variables;

EPS := .00000002; *B* := 4.0; *N* := 16; *C*[0] := 0.0;

C[1] := .19741265;

C[2] := .38292492; *C*[3] := .5467530;

C[4] := .68268949;

C[5] := .78870045; *C*[6] := .86638560;

C[7] := .91988169;

C[8] := .95449974; *C*[9] := .97555105;

C[10] := .98758067;

C[11] := .99404047; *C*[12] := .99730020;

C[13] := .99884595;

C[14] := .99953474; *C*[15] := .99982317;

C[16] := .99993666;

ONE := .99999999;

comment *ONE* is the largest number less than 1 which may
be represented in the machine. This prevents loss of ac-
curacy in some implementations of floating point sub-
traction;

PI2 := .797884560802865;

comment *PI2* = $(2/\pi)^{1/2}$;

DELTA := *B*/*N*;

DELTA2 := .5 × *DELTA*;

EPS3 := 2.0 × *EPS*;

t2 := *max*(*B* × *DELTA*, *sqrt*(2.0) × *DELTA2*);

t := *DELTA2* × (*B* + *DELTA2*);

x := (*t* + *sqrt*(*t*)) × *exp*(.5 × *t*);

y := *t2* × (1.0 + *t2*) × *exp*(.5 × *t2* ↑ 2);

if *t2* ≤ 1 ∧ *y* ≤ *x* **then** *EPS2* := *EPS*/*y* **else** *EPS2* := *EPS*/*x*;

phi := 0

end initialization

else

begin comment compute $\Phi(a)$;

y := *abs*(*a*);

if *y* > *B* **then**

begin comment computation by asymptotic series;

x := *y* ↑ 2; *f* := *PI2* × *exp*(-.5 × *x*)/*y*;

x := 1.0/*x*; *z* := *f*; *f1* := -*f* × *x*;

```

for  $m := 3, m + 2$  while  $\text{abs}(f1) < \text{abs}(f)$  do
begin  $z := z + f1$ ;  $f := f1$ ;  $f1 := -f1 \times m \times x$ ;
  if  $\text{abs}(f) \leq EPS3$  then go to L1
end;
L1:  $z := ONE - z + .5 \times f$ 
end asymptotic computation
else
begin comment Taylor's series computation;
   $m := \text{entier}(y/DELTA)$ ;
   $x := m \times DELTA$ ;  $t := y - x$ ;
  if  $DELTA2 < t$  then
  begin  $m := m + 1$ ;  $x := x + DELTA$ ;  $t := y - x$  end;
   $xt := x \times t$ ;  $f2 := t \uparrow 2$ ;
   $f1 := t \times PF2 \times \exp(-.5 \times x \uparrow 2)$ ;
   $f2 := -.5 \times xt \times f1$ ;
   $z := C[m] + f1 + f2$ ;

```

```

for  $m := 3, m + 1$  while  $(m-1) \times EPS2 < \max(\text{abs}(f1), \text{abs}(f2))$  do
begin
   $f := (-x) \times f2 - f2 \times (m-2) \times f1 / (m-1) \times f/m$ ;
   $z := z + f$ ;  $f1 := f2$ ;  $f2 := f$ ;
end
end Taylor's series computation;
if  $k = 1$  then
begin
   $z := \text{if } 0 \leq a \text{ then } .5 + .5 \times z \text{ else } .5 - .5 \times z$ 
end;
 $\phi := z$ 
end computation
end  $\phi$ 

```

REFERENCES:

1. CRAWFORD, M., AND TCHO, R. Algorithm 123, Real error function, *ERF(x)*, *Comm. ACM* 5 (Sept. 1962), 482.
2. THACHER, H. C., JR. Algorithm 180, Error function—large X , *Comm. ACM* 6 (June 1963), 314.
3. IBBETSON, D. Algorithm 209, Gauss, *Comm. ACM* 6 (Oct. 1963), 616.
4. CYVIN, S. J. Algorithm 226, Normal distribution function, *Comm. ACM* 7 (May 1964), 295.
5. NATIONAL BUREAU OF STANDARDS. *Tables of Normal Probability Functions*, Applied Math. Series, No. 23, US Government Printing Off., Washington, D.C., 1953.

Revised Algorithms Policy • May, 1964

A contribution to the Algorithms department must be in the form of an algorithm, a certification, or a remark. Contributions should be sent in duplicate to the editor, typewritten double-spaced in capital and lower-case letters. Authors should carefully follow the style of this department, with especial attention to indentation and completeness of references. Material to appear in **boldface** type should be underlined in black. Blue underlining may be used to indicate *italic* type, but this is usually best left to the Editor.

An algorithm must be written in the ALGOL 60 Reference Language [*Comm. ACM* 6 (Jan. 1963), 1-17], and normally consists of a commented procedure declaration. Each algorithm must be accompanied by a complete driver program in ALGOL 60 which generates test data, calls the procedure, and outputs test answers. Moreover, selected previously obtained test answers should be given in comments in either the driver program or the algorithm. The driver program may be published with the algorithm if it would be of major assistance to a user.

Input and output should be achieved by procedure statements, using one of the following five procedures (whose body is not specified in ALGOL): [see "Report on Input-Output Procedures for ALGOL 60," *Comm. ACM* 7 (Oct. 1964), 623-629].

```

procedure inreal (channel, destination); value channel; integer channel;
real destination; comment the number read from channel channel is assigned to the variable destination; . . . ;
procedure outreal (channel, source); value channel, source; integer channel;
real source; comment the value of expression source is output to channel channel; . . . ;
procedure ininteger (channel, destination);
value channel; integer channel, destination; . . . ;
procedure outinteger (channel, source);
value channel, source; integer channel, source; . . . ;
procedure outstring (channel, string); value channel; integer channel;
string string; . . . ;

```

If only one channel is used by the program, it should be designated by 1. Examples:

```

  outstring (1, 'x = '); outreal (1, x);
  for  $i := 1$  step 1 until  $n$  do outreal (1,  $A[i]$ );
  ininteger (1, digit [17]);

```

It is intended that each published algorithm be a well-organized, clearly commented, syntactically correct, and a substantial contribution to the ALGOL literature. All contributions will be refereed both by human beings and by an ALGOL compiler. Authors should give great attention to the correctness of their programs, since referees cannot be expected to debug them. Because ALGOL compilers are often incomplete, authors are encouraged to indicate in comments whether their algorithms are written in a recognized subset of ALGOL 60 [see "Report on SUBSET ALGOL 60 (IFIP)," *Comm. ACM* 7 (Oct. 1964), 626-627].

Certifications and remarks should add new information to that already published. Readers are especially encouraged to test and certify previously uncertified algorithms. Rewritten versions of previously published algorithms will be refereed as new contributions, and should not be imbedded in certifications or remarks.

Galley proofs will be sent to the authors; obviously rapid and careful proofreading is of paramount importance.

Although each algorithm has been tested by its author, no liability is assumed by the contributor, the editor, or the Association for Computing Machinery in connection therewith.

The reproduction of algorithms appearing in this department is explicitly permitted without any charge. When reproduction is for publication purposes, reference must be made to the algorithm author and to the *Communications* issue bearing the algorithm.—G.E.F.

CERTIFICATION OF ALGORITHM 124 [S17]

HANKEL FUNCTION [Luis J. Schaeffer, *Comm. ACM* 5 (Sept. 1962), 483]

GEORGE A. REILLY (Recd. 5 Oct. 1964 and 4 Nov. 1964)
Westinghouse Research Laboratories, Pittsburgh, Pa.

This procedure, after modification, was run on the B-5000 using B-5000 ALGOL. Values obtained checked with US National Bureau of Standards *Handbook of Mathematical Functions*, Applied Mathematics Series 55, US Government Printing Office, Washington, D.C. 1964.

For $N = 0, 1$ and 2, accuracy was to 10 decimals for $X < 8.0$. It deteriorated to 6 decimals for $8 < X < 17.5$. For $3 \leq N \leq 9$ accuracy was to the 5 decimals of the tables.

Some changes proved necessary to make the algorithm run. Since the algorithm is short and the changes are involved, the algorithm is restated here. Note that a test for a zero argument X is included in the body of the procedure since $H[2]$ ought to be minus infinity when $X = 0$.

procedure *HANKEL* (N, X, H); **value** N, X ; **integer** N ;

real X ; **array** H ;

begin **real** K, P, R, A, S, T, D, L ; **integer** Q ;

if $X = 0$ **then**

begin **comment** In this case $H[2]$ is minus infinity. M denotes the largest number which can be represented in the machine.

The numerical value of M is to be written into the procedure:

```

   $H[2] := -M$ ;
   $H[1] := \text{if } N = 0 \text{ then } 1 \text{ else } 0$ ;
  go to exit

```

end;

$A := R := 1$; $H[1] := H[2] := S := 0$;

if $N = 0$ **then** **begin** $R := 1$; $S := D := 0$ **end**

else

begin **for** $Q := 1$ **step** 1 **until** N **do**

begin $R := R \times Q$; $S := S + 1/Q$ **end**; $D := R/N$

end;

$R := 1/R$; $K := X \times X/4$; $P := K \uparrow N$; $T := \ln(K) + 1.1544313298631$;

INDEX BY SUBJECT TO ALGORITHMS, 1965

<u>REAL ARITHMETIC, NUMBER THEORY</u>				<u>SIMPLE CALCULATIONS ON STATISTICAL DATA</u>			
A1	139	DIOPHANTINE EQUATION	11-62(556), 3-65(170)	G1	TAIL AREA PROB. FOR 2X2 TABLE	COMP. BULL. V9(56)	
A1	262	RESTRICTED PARTITIONS OF N	8-65(493)				
A1	*263	PARTITION GENERATOR	8-65(493)				
A1	*264	MAP OF PARTITIONS INTO INTEGERS	8-65(493)				
<u>TRIG AND INVERSE TRIG FUNCTIONS</u>				<u>RANDOM NUMBER GENERATORS</u>			
B1	206	ARCCOSSIN	9-63(519), 2-65(104)	G5	121 RANDOM NORMAL	9-62(482), 9-65(556)	
B1		SIN FCN. BY CHEBYSHEV EXPANSION	NUM. MATH. V4(411), V7(194)	G5	200 RANDOM NORMAL	8-63(444), 9-65(556)	
B1		COS FCN. BY CHEBYSHEV EXPANSION	NUM. MATH. V4(411), V7(195)	G5	266 PSEUDO-RANDOM NUMBERS	10-65(605)	
B1		TAN FCN. BY CHEBYSHEV EXPANSION	NUM. MATH. V4(412), V7(195)	G5	267 RANDOM NORMAL DEVIATES	10-65(606)	
<u>EXPONENTIAL AND LOGARITHMIC FUNCTIONS</u>				<u>PERMUTATIONS AND COMBINATIONS</u>			
B3	243	LOGARITHM OF COMPLEX NUMBER	11-64(660), 5-65(279)	G6	202 PERMUTATIONS	9-63(517), 9-65(556)	
				G6	235 RANDOM PERMUTATION	7-64(420), 7-65(445)	
				G6	250 INVERSE PERMUTATION	2-65(104), 11-65(670)	
<u>ZEROS OF POLYNOMIALS</u>				<u>OPERATIONS RESEARCH, GRAPH STRUCTURES</u>			
C2	256	MODIFIED GRAEFFE METHOD	6-65(379)	H	119 PERT NETWORK	8-62(436), 5-65(330)	
C2		LEHMERS METHOD	BIT 1964(255)	H	248 NETFLOW	2-65(103)	
<u>SUMMATION OF SERIES, CONVERGENCE ACCELERATION</u>				H	258 TRANSFORM	6-65(381), 7-65(445)	
C6	255	FOURIER COEFFICIENTS	5-65(279)	H	*263 INTEGER PROGRAMMING-GOMORY1	10-65(601)	
				H	MINIMAL SPANNING TREE	COMP. BULL. V8(67),	
				H	V8(109), V8(147), V9(18)	BIT 1964(194)	
				H	SIMPLEX METHOD		
<u>QUADRATURE</u>				<u>INPUT - COMPOSITE</u>			
D1	145	ADAPTIVE SIMPSON	12-62(604), 4-63(167),	I5	249 OUTREAL V	2-65(104)	
D1	145	3-65(171)					
D1	257	HAVIE INTEGRATOR	6-65(381)	L2	265 FIND PRECEDENCE FUNCTIONS	10-65(604)	
D1		ROMBERG METHOD	COLL. ANAL. NUM. (1961)				
D1		ROMBERG METHOD	BIT 1964(58)				
<u>PARTIAL DIFFERENTIAL EQUATIONS</u>				<u>SORTING</u>			
D3		LINEAR ELLIPTIC BNDY. VAL. PROB.	AUTOMATISIERTE	M1	245 TREESORT 3	12-64(701), 7-65(445)	
D3		BEHANDLUNG ELLIPTISCHER RANDWERTPROBLEME (PUB. 1962)		M1	271 QUICKERSORT	11-65(669)	
D3		BNDY. VALUE PROBS. - INTEGRAL OPRS	NUM. MATH. V7(56)	M1	SORTING OF INTEGERS	COMP. BULL. V9(63)	
<u>INTERPOLATION</u>				<u>SIMULATION OF COMPUTING STRUCTURE</u>			
E1	*264	INTERPOLATION IN A TABLE	10-65(602)	O2	EVALUATION OF FCNAL EXPRESSION	BIT 1965(137)	
<u>MINIMIZING OR MAXIMIZING A FUNCTION</u>				<u>SYMBOL MANIPULATION</u>			
E4	203	MINIMIZE FUNCT. OF N VARIABLES	9-63(517), 10-64(585),	R2	268 ALGOL 60 REF. LANG. EDITOR	11-65(667)	
E4	203	3-65(171)					
E4	205	MINIMIZE FUNCT. OF N VARIABLES	9-63(519), 3-65(171)				
E4	251	FUNCTION MINIMIZATION	3-65(169)				
E4		FIBONACCI SEARCH	COMP. BULL. V8(147)				
<u>MATRIX OPERATIONS, INCLUDING INVERSION</u>				<u>APPROXIMATION OF SPECIAL FUNCTIONS...</u>			
F1	231	INVERSION-GAUSS. ELIM.-COMP. PIV.	6-64(347), 4-65(220)	S	FUNCTIONS ARE CLASSIFIED SO1 TO S22, FOLLOWING		
				S	FLETCHER-MILLER-ROSENHEAD, INDEX OF MATH. TABLES		
<u>EIGENVALUES AND EIGENVECTORS OF MATRICES</u>				S15	272 NORMAL DISTRIBUTION FUNCTION	12-65(789)	
F2	253	SYMMETRIC QR-EIGENVALUES	4-65(217)	S16	259 LEGENDRE FUNCTION	8-65(488)	
F2	254	SYMMETRIC QR-EIGENVALUES, EIGENVECTORS	4-65(218)	S17	21 BESSEL FUNCTION	11-60(600), 4-65(219)	
F2	270	EIGENVECTORS BY GAUSSIAN ELIM.	11-65(668)	S17	124 HANKEL FUNCTION	9-62(483), 12-65(790)	
F2		SYMMETRIC-8 SECTION, INV. ITN.	BIT 1964(124)	S17	236 BESSEL FCNS OF FIRST KIND	8-64(479), 2-65(105)	
<u>DETERMINANTS</u>				S21	ELLIPTIC INTEGRALS-KINDS 1, 2, 3	NUM. MATH. V7(85),	
F3	269	DETERMINANT BY GAUSSIAN ELIM.	11-65(668)	S21	V7(353)		
				S21	JACOBIAN ELLIPTIC FUNCTIONS	NUM. MATH. V7(89)	
				S22	RIEMANN ZETA FUNCTION	BIT 1965(141)	
				S23	234 POISSON-CHARLIER POLYNOMIALS	7-64(420), 2-65(105)	
<u>SIMULTANEOUS LINEAR EQUATIONS</u>				<u>ALL OTHERS</u>			
F4	135	CROUT WITH EQUILIBRATION	11-62(553), 11-62(557),	Z	246 GRAYCODE	12-64(701), 6-65(382)	
F4	135	7-64(421), 2-65(104)		Z	252 VECTOR COUPLING COEFFICIENTS	4-65(217)	
F4		LEAST SQUARES SOLUTION	NUM. MATH. V7(271)	Z	260 6-J SYMBOLS	8-65(492)	
F4		GAUSSIAN ELIMINATION	BIT 1965(64)	Z	261 9-J SYMBOLS	8-65(492)	
F4		ELIM. WITH WEIGHTED ROW COMB.	NUM. MATH. V7(341)	Z	GRADER PROGRAM	CACM 5-65(277)	
				Z	CALCULATION OF EASTER	COMP. BULL. V9(18)	

Key—1st column: A1, B1, B3, etc. is the key to the underlined Modified Share Classification heading each group of algorithms; 2d column: number of the algorithm in CACM; 3d column: title of algorithm; 4th column: month, year and page (in parens) in CACM, or reference elsewhere. This index supplements the previously published indexes: Index by Subject to Algorithms, 1960-1963 [CACM 7 (Mar. 1964), 146-149], and 1964 [CACM 7 (Dec. 1964), 703].

comment The last constant is $2 \times \gamma$, Euler's constant;
for $Q := 0, Q + 1$ while $Q \leq N \vee L \neq H$ [2] do
begin $L := H[2]$; $H[1] := H[1] + A \times R$;
 $H[2] := H[2] + A \times (R \times (T - S) - (\text{if } q < N \text{ then } D/P \text{ else } 0))$;
 $A := A \times K/(Q+1)$; $R := -R/(Q+N+1)$;
 $S := S + 1/(Q+1) + 1/(Q+N+1)$;

if $Q + 1 < N$ then $D := D/(N - Q - 1)$;
end;
 $P := (X/2) \uparrow N$; $H[1] := H[1] \times P$; $H[2] := 0.318309886184$
 $\times H[2] \times P$;
comment The multiplicative constant is $1/Pi$;
exit;
end HANKEL