Algorithms

G. E. FORSYTHE, Editor

ALGORITHM 221
GAMMA FUNCTION

WALTER GAUTSCHI (Recd 10 Aug. 63)
Oak Ridge National Laboratory,* Oak Ridge, Tenn.
* Now at Purdue University, Lafayette, Ind.

```
real procedure gamma (z);  value z;  real z;
comment  This is an auxiliary procedure which evaluates Γ(z)
  for 0 < z ≦ 3 to 10 significant digits. It is based on a polynomial
  approximation given in H. Werner and R. Collinge, Math.
  Comput. 15 (1961), 195–197. This procedure must be replaced
  by a more accurate one if more than 10 significant digits are de-
  sired in Algorithm 222 below. Approximations to the gamma
  function, accurate up to 18 significant digits, may be found in
  the paper quoted above;
begin
  integer k;  real p, t;  array A[0:10];
  A[0] := 1.0;  A[1] := .4227843370;  A[2] := .4118402518;
  A[3] := .0815782188;  A[4] := .0742379076;
  A[5] := − .0002109075;  A[6] := .0109736958;
  A[7] := − .0024667480; A[8] := .0015397681;
  A[9] := −.0003442342;  A[10] := .0000677106;
  t := if z ≦ 1 then z else if z ≦ 2 then z−1 else z−2;
    p := A[10];
  for k := 9 step −1 until 0 do p := t × p + A[k];
  gamma := if z ≦ 1 then p/(z × (z +1)) else if z ≦ 2 then
    p/z else p
end gamma
```

ALGORITHM 222
INCOMPLETE BETA FUNCTION RATIOS

WALTER GAUTSCHI (Recd 10 Aug. 63)
Oak Ridge National Laboratory,* Oak Ridge, Tenn.
* Now at Purdue University, Lafayette Ind.

```
comment  Let B_x(p, q) = ∫_0^x t^(p−1) (1 − t)^(q−1) dt (p > 0, q > 0,
  0 ≦ x ≦ 1) denote the incomplete beta function. The objective
  of this algorithm is to evaluate a sequence of ratios I_x(p, q) =
  B_x(p, q)/B_1(p, q), as one of the parameters p, q varies in steps of
  unity while the other remains fixed. The procedure incomplete beta
  q fixed evaluates I_x(p + n, q) for n = 0, 1, ··· , nmax, assum-
  ing 0 < p ≦ 1, q > 0, whereas the procedure incomplete
  beta p fixed evaluates I_x(p, q + n) for n = 0, 1, ··· , nmax, as-
  suming 0 < q ≦ 1, p > 0. The number d of significant digits
  desired can be specified, but is only guaranteed when x ≦ 1/2.
  When x > 1/2, the complements 1 − I_x will be accurate to d sig-
  nificant digits. In the region 0 < p ≦ 1, 0 < q ≦ 2, I_x(p, q) is
  calculated from a power series expansion. The sequences f(n)
  = I_x(p + n, q) and g(n) = I_x(p, q + n), including initial values,
  are generated recursively by means of the recurrence relations
  f(n + 1) − (1 + (n + p + q − 1) x/(n + p)) f(n) + ((n + p + q
  − 1) x/(n + p)) f(n − 1) = 0, g(n + 1) − (1 + (n+p+q − 1)·
  (1 − x)/(n + q)) g(n) + ((n + p + q − 1) (1 − x)/(n + q))
  ·g(n − 1) = 0. Since the former is mildly unstable, a variant
  of the backward recurrence algorithm of J. C. P. Miller is ap-
  plied to it. A global real procedure gamma (z) must be avail-
  able (see Algorithm 221);
real procedure Isubx p and q small (x, p, q, d);
  value x, p, q, d;
  integer d;  real x, p, q;
comment  This procedure evaluates I_x(p, q) to d significant
  digits when 0 < p ≦ 1 and 0 < q ≦ 2. It first calculates
  B_x(p, q) by a series expansion in powers of x, and then divides
  the result by B_1(p, q) = Γ(p)Γ(q)/Γ(p + q), using the real
  procedure gamma;
begin integer k;  real epsilon, u, v, s;
  epsilon := .5 × 10↑(−d);
  u := x↑p;  s := u/p;  k := 0;
L0:  u := (k−q+1) × (k+p) × x × u/(k+1);
  v := u/(k+p+1);  s := s + v;  k := k + 1;
  if abs(v)/s > epsilon then go to L0;
  Isubx p and q small := s × gamma(p+q)/(gamma(p) ×
    gamma(q))
end Isubx p and q small;
procedure forward (x, p, q, I0, I1, nmax, I);
  value x, p, q, I0, I1, nmax;
  integer nmax;  real x, p, q, I0, I1;  array I;
comment  Given I0 = I_x(p, q), I1 = I_x(p, q+1), this procedure
  generates I_x(p, q+n) for n = 0, 1, 2,··· , nmax, and stores the
  results in the array I;
begin integer n;
  I[0] := I0;  if nmax > 0 then I[1] := I1;
  for n := 1 step 1 until nmax − 1 do
    I[n+1] := (1+(n+p+q−1) × (1−x)/(n+q)) × I[n]
      − (n+p+q−1) × (1−x) × I[n−1]/(n+q)
end forward;
procedure backward (x, p, q, I0, nmax, d, I);
  value x, p, q, I0, nmax, d;
  integer nmax, d;  real x, p, q, I0;  array I;
comment  Given I0 = I_x(p, q), this procedure generates
  I_x(p+n, q) for n = 0, 1, 2, ··· , nmax to d significant digits,
  using a variant of J. C. P. Miller's backward recurrence al-
  gorithm. The results are stored in the array I;
begin
  integer n, nu, m; real epsilon, r; array Iapprox,
  Rr [0:nmax];
  I[0] := I0;  if nmax > 0 then
  begin
    epsilon := .5 × 10↑(−d);
    for n := 1 step 1 until nmax do Iapprox[n] := 0;
    nu := 2 × nmax + 5;
L1:  n := nu;  r := 0;
L2:  r := (n+p+q−1) × x/(n+p+(n+p+q−1) × x
      − (n+p) × r);
    if n ≦ nmax then Rr[n−1] := r;  n := n − 1;
      if n ≧ 1 then go to L2;
    for n := 0 step 1 until nmax − 1 do
      I[n+1] := Rr[n] × I[n];
    for n := 1 step 1 until nmax do
      if abs ((I[n] − Iapprox[n])/I[n]) > epsilon then
      begin
        for m := 1 step 1 until nmax do Iapprox[m] := I[m];
        nu := nu + 5;  go to L1
      end
  end
end backward;
```

```
procedure Isubx qfixed(x, p, q, nmax, d, I);   value x, p, q, nmax, d;
  integer nmax, d;   real x, p, q;   array I;
comment This procedure generates I_x(p+n,q), 0 < p ≦ 1, for
  n=0, 1, ··· , nmax to d significant digits, using the procedure
  backward. In order to calculate the initial value I0=I_x(p,q), it
  first reduces q modulo 1 to q_0 , where 0 < q_0 ≦ 1, then obtains
  I_x(p,q_0) and I_x(p,q_0+1) by the real procedure Isubx p and q small,
  and finally uses these as initial values for the procedure for-
  ward, which connects with I_x(p,q) by the recurrence in q;
begin integer m, mmax;   real s, q0, Iq0, Iq1;
  m := entier(q);   s := q − m;
  q0 := if s > 0 then s else s + 1;
  mmax := if s > 0 then m else m − 1;
  Iq0 := Isubx p and q small(x, p, q0, d);
  if mmax > 0 then Iq1 := Isubx p and q small(x, p, q0+1, d);
  begin array Iq[0:mmax];
    forward (x, p, q0, Iq0, Iq1, mmax, Iq);
    backward(x, p, q, Iq[mmax], nmax, d, I)
  end
end Isubx q fixed;
procedure Isubx pfixed(x, p, q, nmax, d, I);   value x, p, q, nmax, d;
  integer nmax, d;   real x, p, q;   array I;
comment This procedure generates I_x(p, q+n), 0 < q ≦ 1, for
  n=0, 1, ··· , nmax to d significant digits, using the procedure
  forward. The initial values I0=I_x(p,q), I1=I_x(p,q+1) are ob-
  tained by twice applying the procedure backward. The initial
  values for the latter are provided by the real procedure Isubx p
  and q small;
begin integer m, mmax;   real s, p0, I0, I1, Iq0, Iq1;
  m := entier(p);   s := p − m;
  p0 := if s > 0 then s else s + 1;
  mmax := if s > 0 then m else m − 1;
  I0 := Isubx p and q small(x, p0, q, d);
  I1 := Isubx p and q small(x, p0, q+1, d);
  begin array Ip[0:mmax];
    backward(x, p0, q, I0, mmax, d, Ip); Iq0 := Ip[mmax];
    backward(x, p0, q+1, I1, mmax, d, Ip); Iq1 := Ip[mmax]
  end;
  forward(x, p, q, Iq0, Iq1, nmax, I)
end Isubx p fixed;
procedure incomplete beta q fixed(x, p, q, nmax, d, I);
  value x, p, q, nmax, d;
  integer nmax, d;   real x, p, q;   array I;
comment This procedure obtains the final results I_x(p+n,q),
  0 < p ≦ 1, n=0, 1, ··· , nmax, directly from the procedure
  Isubx q fixed, if x ≦ 1/2, or via the relation I_x(p+n,q) =
  1 − I_{1−x}(q,p+n) and the procedure Isubx p fixed, if x > 1/2. The
  indicated substitution in the case x > 1/2 is made to ensure fast
  convergence of both the power series used in the real procedure
  Isubx p and q small, and the backward recurrence algorithm used
  in the procedure backward. If the parameters x, p, q, nmax are
  not in the intended range, control is transferred to a nonlocal
  label called alarm;
begin integer n;
  if x < 0∨x > 1∨p ≦ 0∨ p > 1∨q ≦ 0∨ nmax < 0 then go to
  alarm;
  if x=0∨x=1 then for n := 0 step 1 until nmax do I[n] := x else
  begin
    if x ≦ .5 then Isubx q fixed(x, p, q, nmax, d, I) else
    begin
      Isubx p fixed(1−x, q, p, nmax, d, I);
    for n := 0 step 1 until nmax do I[n] := 1 − I[n]
    end
  end
end incomplete beta q fixed;
procedure incomplete beta p fixed(x, p, q, nmax, d, I);
  value x, p, q, nmax, d;   integer nmax, d;   real x, p, q;   array I;
comment This procedure, the exact analogue to the procedure
  incomplete beta q fixed, generates the final results I_x(p,q+n),
  0 < q ≦ 1, n=0, 1, ··· , nmax. For the setup of the procedure,
  see the comment in incomplete beta q fixed;
begin integer n;
  if x < 0 ∨ x > 1 ∨ q ≦ 0 ∨ q > 1 ∨p ≦ 0∨ nmax < 0 then go to
  alarm;
  if x=0∨x=1 then for n := 0 step 1 until nmax do I[n] := x else
  begin
    if x ≦ .5 then Isubx p fixed(x, p, q, nmax, d, I) else
    begin
      Isubx q fixed(1−x, q, p, nmax, d, I);
      for n := 0 step 1 until nmax do I[n] := 1 − I[n]
    end
  end
end incomplete beta p fixed
```

Reference: Walter Gautschi, Recursive computation of special functions. U. of Michigan, Eng. Summer Conf., Numerical Analysis, 1963.

REMARK ON REVISION OF ALGORITHM 41 EVALUATION OF DETERMINANT [Josef G. Solomon, *Comm. ACM 4* (Apr. 1961), 176; Bruce H. Freed, *Comm. ACM 6* (Sept. 1963), 520]

Leo J. Rotenberg (Recd 7 Oct. 63)
Box 2400, 362 Memorial Dr., Cambridge, Mass.

While desk-checking the program an error was found. For example, the algorithm as published would have calculated the value zero as the determinant of the matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

The error lies in the search for a nonzero element in the rth column of the matrix b.

Editor's Note. Apparently the best general determinant evaluators in this section are imbedded in the linear equation solvers Algorithm 43 [*Comm. ACM 4* (Apr. 1961), 176, 182; and *6* (Aug. 1963), 445] and Algorithm 135 [*Comm. ACM 5* (Nov. 1962), 553, 557]. They search each column for the largest pivot in absolute value. Algorithm 41 searches only for a nonzero pivot in each column, and will therefore fail for the matrix

$$\begin{bmatrix} 2^{-t} & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

if $t \gg s$, for a machine with s-bit floating point.

It is hoped that soon a good determinant evaluator will be published to take the place of Algorithm 41.—G. E. F.

CERTIFICATION OF ALGORITHM 122 TRIDIAGONAL MATRIX [Gerard F. Dietzel, *Comm. ACM 5* (Sept. 1962), 482]

Peter Naur (Recd 27 Sept. 63)
Regnecentralen, Copenhagen, Denmark

TRIDIAG needed the following corrections:

1. Insert k among the local integers to read:
   integer i, j, $j1$, $j2$, $j3$, $j4$, $n1$, k;
2. At the end of line 5 of the procedure body, insert the colon to read $U[j, i] := 0$;
3. Change the round parenthesis to a square bracket following **for** $k := j3 \cdots$ to read $temp := A[j1, k]$;

With these corrections the algorithm worked satisfactorily with the Gier Algol system. As a test it was tried with the following matrix:

$$HBH\ TESTMATRIX[j, i] = HBH\ TESTMATRIX[i, j] = n + 1 - j \qquad (j \geqq i)$$

(cf. the Certification of Alg. 85, *Comm. ACM 6* (Aug. 1963), 447). As a check the resulting matrix was rotated back again, using the resulting U-matrix, and the largest deviation of any element from the original was found.

For comparison the figures obtained by using the algorithms given by Wilkinson in *Numerische Mauhematik 4* (1962), 354–376, may be used. Wilkinson's algorithms use Householder's method of obtaining the tridiagonal form. It should be noted that the deviations given in the table below for Householder's method refer to the final result of obtaining the eigenvalues and vectors, and not only the tridiagonal form, and thus include error contributions from a rather longer chain of calculations than the ones given for *TRIDIAG*. The times, however, only refer to the tridiagonalisation process in both cases.

	$n=5$	$n=10$	$n=15$
Largest deviation			
TRIDIAG,	$1.4_{10}-7$	$7.0_{10}-7$	$2.4_{10}-6$
householder tridiagonalisation		$1.4_{10}-7$	$1.3_{10}-6$
Time of execution, in GIER ALGOL, seconds			
TRIDIAG	2	7	34
householder tridiagonalisation	1	4	10

These figures clearly demonstrate the superiority of the Householder process. Since, in addition, the Householder method in the form given by Wilkinson uses much less storage for variables, Algorithm 122 cannot be recommended.

Revised Algorithms Policy

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. For the convenience of the printer, contributors are requested to double-space material and underline all words that are to appear in boldface type.

An algorithm must be written in ALGOL [see *Communications of the ACM*, January 1963], and normally consists of a commented procedure declaration. Each procedure must be accompanied by a complete driver program in ALGOL which generates test data, calls the procedure, and outputs test answers. Moreover, the author's previously obtained test answers should be given in comments in the driver program. The output statements can take a form like

write ('$x=$', x, '$A=$', **for** i := 1 **step** 1 **until** n **do** $A[i]$);

The driver program may be published with the algorithm, as it should be of assistance to any user.

Insofar as possible, all contributions will be refereed both by human beings and by an ALGOL compiler. It is intended that each algorithm published be a substantial addition to knowledge; thus it should be well-organized, clearly commented and syntactically correct. Because ALGOL compilers are often incomplete, authors should indicate in comments any unusual features of ALGOL that are used, like recursive procedure calls. Authors should give great attention to the correctness of their programs, since referees cannot be expected to debug them. Although each algorithm has been tested by its contributor, no liability is assumed by the contributor, the editor, or the Association for Computing Machinery in connection therewith.

Galley proofs will be sent to the authors for corrections; obviously proofreading is of paramount importance.

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.

REMARK ON ALGORITHM 123

ERF(x) [Martin Crawford and Robert Techo, *Comm. ACM 5* (Sept. 1962), 483; *6* (June 1963), 316; *6* (Oct. 1963), 618]

STEPHEN P. BARTON AND JOHN F. WAGNER (Recd 2 Dec. 63)
General Telephone and Electronics Laboratories, Bayside, New York

This algorithm may err when the Taylor series expands about a root of the nth-order Hermite polynomial; one such error has already been noted [Remark on Algorithm 123, D. Ibbetson, *Comm. ACM 6* (Oct. 1963), 618]. The difficulty springs from the Taylor-series truncation criterion, which assumes that the magnitude of successive terms in the Taylor series decreases. This is not always so, as may be seen by relating

$$\Phi^{(n)}(x) \equiv \frac{2}{\sqrt{\pi}} \frac{d^{n-1}}{dx^{n-1}} (e^{-x^2}), \qquad (n \geqq 1)$$

to the Hermite polynomial $H_n(x)$, which can be defined as

$$H_n(x) \equiv (-1)^n e^{x^2} \frac{d^n}{dx^n} (e^{-x^2}).$$

Therefore

$$\Phi^{(n)}(x) = \frac{2}{\sqrt{\pi}} (-1)^{n-1} e^{-x^2} H_{n-1}(x).$$

As a result, $\Phi^{(n)}(x)$ vanishes when x is a root of $H_{n-1}(x)$ and the Taylor series may be terminated prematurely.

The algorithm was translated into FORTRAN II and run on a Scientific Data Systems 910 computer (39-bit mantissa) with the following changes:

(1) The argument was decremented by 0.25 rather than 0.5.

(2) The truncation criterion for *abs* (T) was 10^{-12} rather than 10^{-10}.

Errors, detected for $x = 1/\sqrt{2}$ and $x = 2.652$, were traced to the above described premature truncation of the relevant Taylor series. These arguments correspond to the roots of $H_2(x)$ and $H_7(x)$.

The program was therefore modified to sum a fixed number of terms, with special attention to the difficulties that might arise when expanding about roots of $H_n(x)$. In particular, in Algorithm 123, line 9, the coefficient, $A^n/n!$, of the nth term in the Taylor expansion, is obtained via the intermediate step of dividing the $(n-1)$-term, T, by the $(n-1)$-derivative, V. The possibility of dividing by $V = 0$ when the Taylor expansion takes place about roots of $H_{n-2}(x)$ was avoided by modifying the program to compute coefficients directly from the recursion relation,

$$A^n/n! = [A^{n-1}/(n-1)!][A/n].$$

In selecting the number of terms to be included in each Taylor series, consideration should also be given to the size of the standard decrement (specified as 0.5 in line 3 of Algorithm 123), for it is the combination of these two parameters which largely determines the accuracy and running time. A brief survey suggested that at least 10-digit accuracy could be obtained if a decrement of 0.4 were employed with 16 terms in each Taylor series; this resulted in an average running time of about 3.5 seconds per computation for arguments in the range $0 \leqq x \leqq 5.0$.

REFERENCE: H. MARGENAU and G. M. MURPHY, *The Mathematics of Physics and Chemistry*, pp. 119, 122. D. van Nostrand, 1943.

(*Algorithms are continued on page 148.*)

ALGORITHMS NOT IN CACM HAVE BEEN INCLUDED, WHEN KNOWN TO US.

LIST OF MODIFIED SHARE CLASSIFICATION

A1	REAL ARITHMETIC, NUMBER THEORY
A2	COMPLEX ARITHMETIC
B1	TRIG AND INVERSE TRIG FUNCTIONS
B2	HYPERBOLIC FUNCTIONS
B3	EXPONENTIAL AND LOGARITHMIC FUNCTIONS
B4	ROOTS AND POWERS
C1	OPERATIONS ON POLYNOMIALS AND POWER SERIES
C2	ZEROS OF POLYNOMIALS
C5	ZEROS OF ONE OR MORE TRANSCENDENTAL EQUATIONS
C6	SUMMATION OF SERIES, CONVERGENCE ACCELERATION
D1	QUADRATURE
D2	ORDINARY DIFFERENTIAL EQUATIONS
D3	PARTIAL DIFFERENTIAL EQUATIONS
D4	DIFFERENTIATION
E1	INTERPOLATION
E2	CURVE AND SURFACE FITTING
E3	SMOOTHING
E4	MINIMIZING OR MAXIMIZING A FUNCTION
F1	MATRIX OPERATIONS, INCLUDING INVERSION
F2	EIGENVALUES AND EIGENVECTORS OF MATRICES
F3	DETERMINANTS
F4	SIMULTANEOUS LINEAR EQUATIONS
F5	ORTHOGONALIZATION
G1	SIMPLE CALCULATIONS ON STATISTICAL DATA
G2	CORRELATION AND REGRESSION ANALYSIS
G5	RANDOM NUMBER GENERATORS
G6	PERMUTATIONS AND COMBINATIONS
G7	SUBSET GENERATORS AND CLASSIFICATIONS
H	OPERATIONS RESEARCH, GRAPH STRUCTURES
I5	INPUT - COMPOSITE
J6	PLOTTING
K2	RELOCATION
M1	SORTING
M2	DATA CONVERSION AND SCALING
O2	SIMULATION OF COMPUTING STRUCTURE
S	APPROXIMATION OF SPECIAL FUNCTIONS...
S	FUNCTIONS ARE CLASSIFIED S01 TO S22, FOLLOWING
S	FLETCHER-MILLER-ROSENHEAD, INDEX OF MATH. TABLES
Z	ALL OTHERS

A1		REAL ARITHMETIC, NUMBER THEORY	
A1	7	EUCLIDEAN ALGORITHM	4-60(240)
A1	35	SIEVE OF ERATOSTHENES	3-61(151),4-62(209),
A1	35	8-62(438)	
A1	61	RANGE ARITHMETIC	7-61(319)
A1	68	AUGMENTATION	8-61(339),11-61(498)
A1	72	COMPOSITIONS	11-61(498),8-62(439)
A1	93	GENERALIZED ARITHMETIC	6-62(344),10-62(514)
A1	95	PARTITIONS	6-62(344)
A1	99	JACOBI SYMBOL	6-62(345),11-62(557)
A1	114	PARTITIONS	8-62(434)
A1	139	DIOPHANTINE EQUATION	11-62(556)
A2			
A2		COMPLEX ARITHMETIC	
A2	116	COMPLEX DIVIDE	8-62(435)
A2	186	COMPLEX ARITHMETIC	7-63(386)
A2		COMPLEX ARITHMETIC	BIT 1962(233)
B1			
B1		TRIG AND INVERSE TRIG FUNCTIONS	
B1	206	ARCCOSSIN	9-63(519)
B1		ARCSIN(Z)	BIT 1962(236)
B1		ARCCOS(Z)	BIT 1962(236)
B1		ARCTAN(Z)	BIT 1962(236)
B1		SIN FCN.BY CHEBYSHEV EXPANSION	NUM.MATH.V4(411)
B1		COS FCN.BY CHEBYSHEV EXPANSION	NUM.MATH.V4(411)
B1		TAN FCN.BY CHEBYSHEV EXPANSION	NUM.MATH.V4(412)
B1		ARCSIN BY CHEBYSHEV EXPANSION	NUM.MATH.V4(412)
B1		ARCTAN BY CHEBYSHEV EXPANSION	NUM.MATH.V4(412)
B2			
B2		HYPERBOLIC FUNCTIONS	
B2		SINH(X)	BIT 1962(235)
B2		COSH(X)	BIT 1962(235)
B3			
B3		EXPONENTIAL AND LOGARITHMIC FUNCTIONS	
B3	46	EXP(Z),Z COMPLEX	4-61(178),6-62(347)
B3	48	LOG(Z),Z COMPLEX	4-61(179),6-62(347),
B3	48	7-62(391)	
B3		EXP FCN.BY CHEBYSHEV EXPANSION	NUM.MATH.V4(410)
B3		LOG FCN.BY CHEBYSHEV EXPANSION	NUM.MATH.V4(411)
B4			
B4		ROOTS AND POWERS	
B4	53	ROOTS OF COMPLEX NUMBERS	4-61(180),7-61(322)
B4	106	POWERS OF COMPLEX NUMBER	7-62(388),11-62(557)
B4	190	POWERS OF COMPLEX NUMBERS	7-63(388)
C1			
C1		OPERATIONS ON POLYNOMIALS AND POWER SERIES	
C1	29	POLYNOMIAL SHIFTER	11-60(604)
C1	131	DIVIDE POWER SERIES	11-62(551)
C1	134	EXPONENTIATE POWER SERIES	11-62(553),7-63(390)
C1	158	EXPONENTIATE POWER SERIES	3-63(104),7-63(390),
C1	158	9-63(522)	
C1	193	REVERT POWER SERIES	7-63(388),12-63(745)
C2			
C2		ZEROS OF POLYNOMIALS	
C2	3	BAIRSTOW	2-60(74),6-60(354),
C2	3	2-61(105),3-61(153),4-61(181)	
C2	30	BAIRSTOW-NEWTON	12-60(643),5-61(238),
C2	30	1-62(50)	
C2	59	RESULTANT METHOD	5-61(236)
C2	75	RATIONAL ROOTS-INTEGER COEFF.	1-62(48),7-62(392),
C2	75	8-62(439)	
C2	78	RATIONAL ROOTS-INTEGER COEFF.	2-62(97),3-62(168),
C2	78	8-62(440)	
C2	105	NEWTON-MAEHLY	7-62(387),7-63(389)
C2	174	BOUNDS ON ZEROS	6-63(311)
C2		ZEROS IN THE RIGHT HALF PLANE	ZH.VYCH.MAT.MAT.FIZ.-
C2		1963(364)	
C5			
C5		ZEROS OF ONE OR MORE TRANSCENDENTAL EQUATIONS	
C5	2	REGULA FALSI	2-60(74),6-60(354),
C5	2	8-60(475),3-61(153)	
C5	4	BISECTION	3-60(174),3-61(153)
C5	15	REGULA FALSI	8-60(475),11-60(602),
C5	15	3-61(153)	
C5	25	REAL ZEROS	11-60(602),3-61(153),
C5	25	3-61(154)	
C5	26	REGULA FALSI	11-60(603),3-61(153)
C5	196	MULLERS METHOD	8-63(442)
C5		ZEROS BY INTERP.OR BISECTION	BIT 1963(205)
C6			
C6		SUMMATION OF SERIES, CONVERGENCE ACCELERATION	
C6	8	EULER SUM	5-60(311),11-63(663)
C6	128	FOURIER SERIES SUMMATION	10-62(513)
C6	157	FOURIER SERIES SUMMATION	3-63(103),9-63(521),
C6	157	10-63(618)	
C6	215	EPSILON ALGORITHM	11-63(662)
C6		EPSILON ALGORITHM	BIT 1962(240)
C6		FIND LIMIT OF SEQUENCE	BIT 1961(64)
C6		SUM FOURIER SERIES	COMP.J.V6(248)
D1			
D1		QUADRATURE	
D1	1	QUADRATURE	2-60(74)
D1	32	MULTIPLE INTEGRAL	2-61(106),2-63(69)
D1	60	ROMBERG METHOD	6-61(255),3-62(168),
D1	60	5-62(281)	
D1	84	SIMPSONS RULE	4-62(208),7-62(392),
D1	84	8-62(440),11-62(557)	
D1	98	COMPLEX LINE INTEGRAL	6-62(345)
D1	103	SIMPSONS RULE	6-62(347)
D1	125	GAUSSIAN COEFFICIENTS	10-62(510)
D1	145	ADAPTIVE SIMPSON	12-62(604),4-63(167)
D1	146	MULTIPLE INTEGRAL	12-62(604)
D1	182	ADAPTIVE SIMPSON	6-63(315)
D1	198	ADAPTIVE,MULTIPLE INTEGRAL	8-63(443)
D1		ADAPTIVE SIMPSONS RULE	BIT 1961(290)
D1		MONTE CARLO QUADRATURE	COMP.J.V6(281)
D2			
D2		ORDINARY DIFFERENTIAL EQUATIONS	
D2	9	RUNGE-KUTTA	5-60(312)
D2	194	ZEROS OF O.D.E. SYSTEM	8-63(441)
D2	218	KUTTA-MERSON	12-63(737)
D3			
D3		PARTIAL DIFFERENTIAL EQUATIONS	
D3		CONFORMAL MAP-ELLIPSE TO CIRCLE	BIT 1962(243)
D3		PDE SOLNS.BY INTEGRAL OPERATORS	STANFORD UNIV.-
D3		APPL.MATH.STAT.REP.NONR 225(37)	NO.24
D3		KERNEL FCN.IN BNDY.VALUE PROBS.	NUM.MATH.V3(209)
D4			
D4		DIFFERENTIATION	
D4	79	DIFFERENCE EXPRESSION COEFF.	2-62(97),3-63(104)
E1			
E1		INTERPOLATION	
E1	18	RATIONAL INTERP.-CONT.FRACT.	9-60(508),8-62(437)
E1	70	AITKEN INTERPOLATION	11-61(497),7-62(392)
E1	77	INTERPOLATION,DIFFN.,INTEGRN.	2-62(96),6-62(348),
E1	77	8-63(446),11-63(663)	
E1	167	CONFLUENT DIVIDED DIFFERENCES	4-63(164),9-63(523)
E1	168	INTERPOLATION-DIVIDED DIFFCES.	4-63(165),9-63(523)
E1	169	INTERPOLATION-DIVIDED DIFFCES.	4-63(165),9-63(523)
E1	187	DIFFCES.AND DERIVS.-RECURSIVE	7-63(387)
E1	210	LAGRANGE INTERPOLATION	10-63(616)
E1	211	HERMITE INTERPOLATION	10-63(617),10-63(619)
E2			
E2		CURVE AND SURFACE FITTING	
E2	28	LEAST SQUARES BY ORTHOG. POLYN.	11-60(604),12-61(544)
E2	37	ECONOMIZATION	3-61(151),8-62(438),
E2	37	8-63(445)	
E2	38	ECONOMIZATION	3-61(151),8-63(445)
E2	74	LEAST SQUARES WITH CONSTRAINTS	1-62(47),6-63(316)
E2	91	CHEBYSHEV FIT	5-62(281),4-63(167)
E2	164	SURFACE FIT	4-63(162),8-63(450)
E2	176	SURFACE FIT	6-63(313)
E2	177	LEAST SQUARES WITH CONSTRAINTS	6-63(313),7-63(390)
E2		CONTINUED FRACTION EXPANSION	BIT 1962(245)
E3			
E3		SMOOTHING	

E3	188	SMOOTHING	7-63(387)
E3	189	SMOOTHING	7-63(387)
E3	216	SMOOTHING	11-63(663)
E4			
E4		MINIMIZING OR MAXIMIZING A FUNCTION	
E4	129	MINIMIZE FUNCT. OF N VARIABLES	11-62(550) 9-63(521)
E4	178	MINIMIZE FUNCT. OF N VARIABLES	6-63(313)
E4	203	MINIMIZE FUNCT.OF N VARIABLES	9-63(517)
E4	204	MINIMIZE FUNCT.OF N VARIABLES	9-63(519)
E4	205	MINIMIZE FUNCT.OF N VARIABLES	9-63(519)
F1			
F1		MATRIX OPERATIONS, INCLUDING INVERSION	
F1	42	INVERSION	4-61(176),11-61(498),
F1	42	1-63(38),8-63(445)	
F1	50	INVERSE OF HILBERT MATRIX	4-61(179),1-62(50),
F1	50	1-63(38)	
F1	51	INVERSE OF PERTURBED MATRIX	4-61(180),7-62(391)
F1	52	INVERSE OF TEST MATRIX	4-61(180),8-61(339),
F1	52	11-61(498),8-62(438),1-63(39),8-63(446)	
F1	58	INVERSION-GAUSSIAN ELIMINATION	5-61(236),6-62(347),
F1	58	8-62(438),12-62(606)	
F1	66	INVERSION-SORT METHOD	7-61(322),1-62(50),
F1	66	6-62(348)	
F1	67	CRAM MATRIX	7-61(322),6-62(348)
F1	120	INVERSION-GAUSSIAN ELIMINATION	8-62(437),1-63(40),
F1	120	8-63(445)	
F1	140	INVERSION	11-62(556),8-63(448)
F1	150	INVERSE OF SYMMETRIC MATRIX	2-63(67),7-63(390)
F1	166	MONTE CARLO INVERSE	4-63(164),9-63(523)
F1	197	MATRIX DIVISION	8-63(443)
F2			
F2		EIGENVALUES AND EIGENVECTORS OF MATRICES	
F2	85	JACOBI METHOD	4-62(208),8-62(440),
F2	85	8-63(447)	
F2	104	REDUCTION-BAND TO TRIDIAGONAL	7-62(387)
F2	122	GIVENS TRIDIAGONAL REDUCTION	9-62(482)
F2	183	REDUCTION-BAND TO TRIDIAGONAL	6-63(315)
F2		HOUSEHOLDERS METHOD	NUM.MATH.V4(354)
F2		EIGENVALUES OF TRIDIAG.MATRIX	NUM.MATH.V4(354)
F2		EIGENVECTORS OF TRIDIAG.MATRIX	NUM.MATH.V4(354)
F2		LR TRANSFORMATION METHOD	NUM.MATH.V5(273)
F2		EIGENVALUES-LAGUERRES METHOD	STANFORD UNIV.-
F2		APPL.MATH.STAT.REP.NONR 225(37)NO.21	
F2		HOUSEHOLDERS METHOD	STANFORD UNIV.-
F2		APPL.MATH.STAT.REP.NONR 225(37)NO.18	
F2		EIGENVALUES BY QR-ALGORITHM	COMP.J.V4(344)
F2		TRIDIAGONAL SIMIL.BY ELIM.	COMP.J.V4(175)
F3			
F3		DETERMINANTS	
F3	41	DETERMINANT EVALUATION	4-61(176),9-63(520)
F3	159	DETERMINANT EVALUATION	3-63(104),12-63(739)
F3	170	DETERMINANT-POLYNOMIAL ELEMENTS	4-63(165),8-63(450)
F4			
F4		SIMULTANEOUS LINEAR EQUATIONS	
F4	16	CROUT WITH PIVOTING	9-60(507),10-60(540),
F4	16	3-61(154)	
F4	17	SOLVE TRIDIAGONAL MATRIX	9-60(508)
F4	24	SOLVE TRIDIAGONAL MATRIX	11-60(602)
F4	43	CROUT WITH PIVOTING	4-61(176),4-61(182),
F4	43	8-63(445)	
F4	92	SIMULT.EQNS.-ITERATIVE SOLN.	5-62(286)
F4	107	GAUSSIAN ELIMINATION	7-62(388),1-63(39),
F4	107	8-63(445)	
F4	126	GAUSSIAN ELIMINATION	10-62(511)
F4	135	CROUT WITH EQUILIBRATION	11-62(553),11-62(557)
F4	195	BAND SOLVE	8-63(441)
F4	220	GAUSS-SEIDEL	12-63(739)
F4		GAUSSIAN ELIMINATION	BIT 1962(256)
F4		LINEAR SYSTEM WITH BAND MATRIX	BIT 1963(207)
F5			
F5		ORTHOGONALIZATION	
F5	127	ORTHONORMALIZATION	10-62(511)
G1			
G1		SIMPLE CALCULATIONS ON STATISTICAL DATA	
G1	208	DISCRETE CONVOLUTION	10-63(615)
G1	212	DETERMINE DISTRIB.FCN.FROM DATA	10-63(617)
G2			
G2		CORRELATION AND REGRESSION ANALYSIS	
G2	39	CORRELATION COEFFICIENTS	3-61(152)
G2	142	TRIANGULAR REGRESSION	12-62(603)
G5			
G5		RANDOM NUMBER GENERATORS	
G5	121	RANDOM NORMAL	9-62(482)
G5	133	RANDOM FLAT	11-62(553),12-62(606)
G5	133	3-63(105),4-63(167)	
G5	200	RANDOM NORMAL	8-63(444)
G5		RANDOM SAMPLES,VARIOUS DISTRIB.	COMP.J.V6(279)
G6			
G6		PERMUTATIONS AND COMBINATIONS	
G6	71	PERMUTATIONS	11-61(497),4-62(209),
G6	71	8-62(439)	
G6	86	PERMUTATIONS	4-62(208),8-62(440)
G6	87	PERMUTATIONS	4-62(209),8-62(440),
G6	87	10-62(514)	
G6	94	COMBINATIONS	6-62(344),11-62(557),
G6	94	12-62(606)	
G6	102	PERMUTATIONS	6-62(346)
G6	115	PERMUTATIONS	8-62(434),10-62(514),
G6	115	12-62(606)	
G6	130	PERMUTATIONS	11-62(551)
G6	152	COMBINATIONS	2-63(68),7-63(385)
G6	154	COMBINATIONS	3-63(103),8-63(449)
G6	155	COMBINATIONS	3-63(103),8-63(449)
G6	156	COMBINATIONS	3-63(103),8-63(450)
G6	160	COMBINATIONS	4-63(161),8-63(450),
G6	160	10-63(618)	
G6	161	COMBINATIONS	4-63(161),8-63(450),
G6	161	10-63(619)	
G6	202	PERMUTATIONS	9-63(517)
G7			
G7		SUBSET GENERATORS AND CLASSIFICATIONS	
G7	81	SUBSEQUENCES	3-62(166)
G7	82	SUBSEQUENCES	3-62(167)
G7	83	CLASSIFICATIONS	3-62(167)
H			
H		OPERATIONS RESEARCH, GRAPH STRUCTURES	
H	27	ASSIGNMENT PROBLEM	11-60(603),10-63(618),
H	27	12-63(739)	
H	40	CRITICAL PATH SCHEDULING	3-61(152),9-61(392),
H	40	10-62(513)	
H	69	CHAIN TRACING	9-61(392)
H	96	ANCESTOR	6-62(344),3-63(104)
H	97	SHORTEST PATH	6-62(345)
H	119	PERT NETWORK	8-62(436)
H	141	FIND PATH	11-62(556)
H	153	INTEGER PROGRAMMING	2-63(68),8-63(449)
H	217	MIN.ACCESS COST CURVE	12-63(737)
I5			
I5		INPUT - COMPOSITE	
I5		OPTICAL SCANNING OF NUMBERS	ZH.VYCH.MAT.MAT.FIZ.-
I5		1962(236)	
J6			
J6		PLOTTING	
J6	162	XY PLOTTER	4-63(161),8-63(450)
K2			
K2		RELOCATION	
K2	173	TRANSFER ARRAY VALUES	6-63(311),10-63(619)
M1			
M1		SORTING	
M1	23	SORT	11-60(601),5-61(238)
M1	63	SORT	7-61(321),8-62(439),
M1	63	8-63(446)	
M1	64	SORT	7-61(321),8-62(439),
M1	64	8-63(446)	
M1	65	SORT	7-61(321),8-62(439),
M1	65	8-63(446)	
M1	76	SORT	1-62(48),6-62(348)
M1	113	TREESORT	8-62(434)
M1	143	TREESORT	12-62(604)
M1	144	TREESORT	12-62(604)
M1	151	LOCATE IN A LIST	2-63(68)
M1	175	SHUTTLE SORT	6-63(312),10-63(619),
M1	175	12-63(739)	
M1	201	SHELL SORT	8-63(445)
M1	207	STRING SORT	10-63(615)
M2			
M2		DATA CONVERSION AND SCALING	
M2		DATA PROCESSING-VECTORCARDIOGRM	CACM 2-62(121)
O2			
O2		SIMULATION OF COMPUTING STRUCTURE	
O2	100	PROCESSING OF CHAIN-LINKED LIST	6-62(346)
O2	101	PROCESSING OF CHAIN-LINKED LIST	6-62(346)
O2	137	NESTED FOR STATEMENT	11-62(555)
O2	138	NESTED FOR STATEMENT	11-62(555)
S			
S		APPROXIMATION OF SPECIAL FUNCTIONS...	
S		FUNCTIONS ARE CLASSIFIED S01 TO S22, FOLLOWING	
S		FLETCHER-MILLER-ROSENHEAD, INDEX OF MATH. TABLES	
S03	19	BINOMIAL COEFFICIENTS	10-60(540),6-62(347),
S03	19	8-62(438)	
S03	33	FACTORIAL N	2-61(106)
S07		POLAR TRANSF. BY CHEBYSHEV EXP.	NUM.MATH.V4(413)
S13	14	COMPLEX EXPONENTIAL INTEGRAL	7-60(406)
S13	20	REAL EXPONENTIAL INTEGRAL	10-60(540),2-61(105),
S13	20	4-61(182)	
S13	108	EXPONENTIAL INTEGRAL	7-62(388),7-62(393)
S13	109	EXPONENTIAL INTEGRAL	7-62(388),7-62(393)
S13		EXPONENTIAL INTEGRAL EXPANSION	CHIFFRES-V6(187)
S13		EI(X) BY CHEBYSHEV EXPANSION	NUM.MATH.V4(413)
S14	31	GAMMA FUNCTION	2-61(105),12-62(605)
S14	34	GAMMA FUNCTION	2-61(106),7-62(391)
S14	54	GAMMA FUNCTION	4-61(180)
S14	80	GAMMA FUNCTION	3-62(166)
S14	147	DERIVATIVE OF GAMMA FUNCTION	12-62(605),4-63(168)
S14	179	BETA RATIO	6-63(314)
S14		GAMMA FUNCTION	BIT 1962(238)
S14		GAMMA FCN. BY CHEBYSHEV EXP.	NUM.MATH.V4(413)
S15	11	HERMITE POLYNOMIAL	6-60(353)
S15	123	ERROR FUNCTION	9-62(483),6-63(316),
S15	123	10-63(618)	
S15	180	ERROR FUNCTION	6-63(314)
S15	181	ERROR FUNCTION	6-63(315)
S15	185	ERROR FUNCTION	7-63(386)
S15	209	ERROR FUNCTION	10-63(616)
S15		ERF(X) BY CHEBYSHEV EXPANSION	NUM.MATH.V4(414)
S16	13	LEGENDRE POLYNOMIAL	6-60(353),2-61(105),

Key—1st column: A1, A2, etc. is the key to the Modified Share Classification listing given at the beginning of the Index; 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.

Algorithms—Continued from 145

CERTIFICATION OF ALGORITHM 150
SYMINV2 [H. Rutishauser, *Comm. ACM 6* (Feb. 1963), 67]

PETER NAUR (Recd 27 Sept. 63)
Regnecentralen, Copenhagen, Denmark

Since the translator refuses to run programs with more than one occurrence of the same identifier in a formal parameter list, the second *a* was taken out when this procedure was run with the GIER ALGOL system [cf. also the discussion in *Comm. ACM 6* (July 1963), 390]. Otherwise it ran smoothly. For testing the accuracy, segments of the Hilbert matrix were inverted and the results multiplied by the original segment and compared with the unit matrix. The largest deviation in any element was found to be:

Order	*Max. deviation from elements of the unit matrix*	*Order*	*Max. deviation from elements of the unit matrix*
2	$-1.49_{10}-8$	6	$-7.32_{10}-3$
3	$-2.38_{10}-7$	7	$-3.59_{10}-1$
4	$-1.53_{10}-5$	8	$-2.95_{10}\ 1$
5	$-3.36_{10}-4$	9	$-1.25_{10}\ 1$

These figures may be compared directly with the ones related to Algorithms 120, *INVERSION II*, and *gjr* [*Comm. ACM 6* (Aug. 1963), 445]. A comparison shows that all three algorithms yield about the same accuracy, with *syminv*2 being the best in most cases, however. This is not too surprising since the knowledge that the matrix is symmetric ought to simplify the calculation considerably.

The lengths of the three procedures after translation are as follows:

	Number of GIER words
*syminv*2	216
INVERSION II	279
gjr	302

Execution times for *syminv*2 in GIER ALGOL are:

Order	*Time (sec)*
5	1
10	3.5
15	10.5
20	23

This is about half the time of execution of *INVERSION II* or *gjr*.

CERTIFICATION OF ALGORITHM 197
MATRIX DIVISION [M. Wells, *Comm. ACM 6* (Aug. 1963), 443]

M. WELLS (Recd 18 Nov. 63)
University of Leeds, Leeds, England

The procedure was tested on a Ferranti Pegasus, using the ALGOL compiler developed by the de Havilland Aircraft Company at Hatfield. The line after the one labelled 'start of program' should read

for $i := 1$ **step** 1 **until** m **do**

(the first 1 was omitted).

The statement labelled *back substituton* is incorrect, and should read

back substitution: **for** $j := 1$ **step** 1 **until** n **do**
begin **for** $i := 1$ **step** 1 **until** m **do**
$c[i,j] := (c[i,j] - dot\ (b[k,i],\ c[k,j],\ k,i-1))/b[i,\ i];$
for $i := m$ **step** -1 **until** 1 **do**
$c[i,j] := (c[i,j] - dot\ (b[i,m+1-k],\ \ c[m+1-k,j], k,m-i))/b[i,i]$
end *of double back substitution*

With these changes the program was operated successfully on a number of small test problems. The procedure is only applicable to symmetric positive definite matrices, and no systematic attempt has yet been made to assess the accuracy of the results.

The word 'symmetric' should be inserted before 'positive definite' in the comment.

It is interesting to note that the original, incorrect version of the procedure will divide one symmetric matrix by another, and so can be used for matrix inversion.

CERTIFICATION OF ALGORITHM 209
GAUSS [D. Ibbetson, *Comm. ACM 6* (Oct. 1963), 616]

(Pvt.) G. W. GLADFELTER (Recd 4 Nov. 63)
RA17667701, 1st Inf. Battle Group U.S. Military Academy (9822), West Point, N.Y.

The algorithm was translated into FORTRAN for the GE 225 and used to publish a table of the error function. No errors were found in the algorithm and the table produced agreed with the published tables at hand (6 significant figures).