$$(1+y)\ln(1+y)$$

$$= y + \sum_{j=2}^{\infty} (-1)^j \left(\frac{1}{j-1} - \frac{1}{j} \right) y^j = \ln(1+x) \quad (10)$$

$$= \sum_{j=1}^{\infty} (-1)^{j-1} \frac{1}{j} x^j.$$

Thus,

$$a_1 = 1,$$

$$a_j = (-1)^j/j(j-1) \qquad (j \geq 2), \quad (11)$$

$$b_j = -(-1)^j/j \qquad (j \geq 1).$$

Applying the algorithm to these coefficients leads to the $c_{i,j}$ array of which the first few elements are as follows:

$i \backslash j$	1	2	3	4	5	6	7
1	1						
2	-1	1					
3	$\frac{3}{2}$	-2	1				
4	$-\frac{17}{6}$	4	-3	1			
5	$\frac{37}{6}$	$-\frac{26}{3}$	$\frac{15}{2}$	-4	1		
6	$-\frac{1759}{120}$	$\frac{81}{4}$	$-\frac{37}{2}$	12	-5	1	
7	$\frac{13279}{360}$	$-\frac{1003}{20}$	$\frac{187}{4}$	$-\frac{100}{3}$	$\frac{35}{2}$	-6	1

As might be expected from the fact that $Y(X)$ has a branch point at $X = e^{-1/e} = .6922 \cdots$, this power series has a small radius of convergence. The corresponding continued fraction, however, obtained by the *qd* algorithm, converges quite satisfactorily. Even for $X = 4$, the error of the sixth convergent is less than 2.4 percent, and the error of the 15th about .0015 percent.

Acknowledgments. It is a pleasure to acknowledge the helpful criticism and advice of Dr. Richard F. King, Argonne National Laboratory, in improving the clarity of the presentation and suggesting the inclusion of a specific example.

RECEIVED APRIL, 1965

REFERENCE

1. THACHER, H. C., JR. Algorithm 273, SERREV. *Comm. ACM 9* (Jan. 1965), 11.

Algorithms

J. G. HERRIOT, Editor

ALGORITHM 273
SERREV [C1]

HENRY C. THACHER, JR. (Recd. 2 Apr. 1965)
Argonne National Laboratory, Argonne, Illinois

(Work supported by the US Atomic Energy Commission.)

```
procedure SERREV (A, B, C, N);
  value N;  integer N;  array A, B, C;
comment This procedure produces in the array C the coefficients
  of the power series y^j = Σ_{i=j}^{N} C_ij x^i, where y is the solution of
        f(y) = Σ_{i=1}^{N} A_i y^i = g(x) = Σ_{i=1}^{N} B_i x^i
  and A_1 = 1. The arrays A and B are linear, with bounds 1 and
  M ≧ N. The array C is square, with bounds 1:M, 1:M. Ele-
  ments above the diagonal are not used. The derivation of the
  method is given in [1];
begin integer I, J, K, LIM;  real T;
  for I := 1 step 1 until N do
    begin for J := I-1 step -1 until 1 do
      begin T := 0;  LIM := I-J;
        for K := 1 step 1 until LIM do T := C[K,1] × C[I-K,J]
          + T;  C[I, J+1] := T
      end for J;
      T := B[I];
      for J := 2 step 1 until I do T := T-A[J] × C[I,J];
      C[I,1] := T
    end for I
end
```

REFERENCE:

1. THACHER, H. C., Jr. Solution of transcendental equations by series reversion. *Comm. ACM 9* (Jan. 1966), 10–11.

ALGORITHM 274
GENERATION OF HILBERT DERIVED TEST MATRIX [F1]

J. BOOTHROYD (Recd. 19 May 1965 and 27 Aug. 1965)
University of Tasmania, Hobart, Tas., Australia

```
procedure testmx(a,n);  value n;  integer n;  array a;
comment T. J. Dekker, "Evaluation of Determinants, Solution
  of Systems of Linear Equations and Matrix Inversion" [Rep.
  No. MR63, Mathematical Centre, Amsterdam] describes a test
  matrix M[1:n, 1:n] with the following properties:
    (a) elements M[i,j] are positive integers,
    (b) the inverse has elements (-1) ↑ (i+j) × M[i,j],
    (c) the degree of ill-condition increases rapidly with increas-
        ing n.
  Such matrices may be formed by M = FG^-1HG where F is a
  diagonal matrix diag(fi) with fi = factorial (n+i-1)/(factorial
  (i-1)↑2)/factorial(n-i), H is the order n segment of a Hilbert
  matrix and G is diagonal, diag(gi), with gi derived from the prime
  decomposition of fi by:
    fi = p1^m1 p2^m2 ··· pk^mk,      gi = p1^(m÷2) p2^(m÷2) ··· pk^(mk÷2).
```

This procedure forms matrices $a[1:n, 1:n]$ of this type and follows Dekker in principle but not in detail. Factorials are avoided by evaluating the fi with a recursion sequence

$$f[1] := n, \qquad f[i+1] := f[i] \times (n\uparrow 2 - i\uparrow 2) \div i\uparrow 2 \qquad (i=1, 2, \cdots, n-1),$$

permitting the exact computation of fi for much larger n than would otherwise be possible. In the evaluation of expressions of the form $(a\times b) \div c$, where the result is integral but c is not a factor of either a or b, numerator integer overflow is avoided by the simple device

$$\text{expression} := q \times b + (r\times b) \div c \quad \text{where} \quad a = q \times c + r.$$

Test matrices for $2 \leqq n \leqq 15$ have been computed on a machine with a 39-bit integer register. During tests of the procedure the specification of the array parameter was changed from **real** to **integer** and the results checked by matrix multiplication using an exact double precision integer inner-product routine. The unit matrix was obtained in all cases. As **real** arrays these matrices will find use only for values of n such that all integer elements have an exact floating point representation. For $10 \leqq n \leqq 15$ the values of the elements of largest modulus are:

n	$M[i,j]max$
10	1616615
11	49884120
12	108636528
13	490804314
14	1859890032
15	22096817600;

```
begin integer i, j, k, fi, gi, d, q, r;  Boolean even;
  integer array f, g[1:n];
  comment First we compute F = diag(fi);
  fi := f[1] := n;  j := n × n;
  for i := 1 step 1 until n−1 do
  begin d := i × i;  k := j−d;
    q := fi ÷ d;  r := fi−q × d;
    f[i+1] := fi := q × k + (r×k) ÷ d
  end;
  comment And now, using a modified prime factors algorithm
    to obtain G = diag(gi), we compute FG⁻¹, whose elements re-
    place those of F;
  for i := 1 step 1 until n do
  begin d := gi := 1;  q := fi := f[i];  j := 2;
newj:  even := false;
next:  if q ≧ j then
    begin q := fi ÷ j;
      if fi ≠ q × j then
      begin j := j+d;  d := 2;  go to newj end;
      if even then gi := gi × j;  even := ¬ even;
      fi := q;  go to next
    end;
    g[i] := gi;  f[i] := f[i] ÷ gi
  end;
  comment Finally, in one operation (FG⁻¹)HG where H is a
    nonexistent Hilbert matrix whose reciprocal elements,
    i+j−1, are computed as we go;
  for i := 1 step 1 until n do
  begin fi := f[i];
    for j := 1 step 1 until n do
    begin gi := g[j];  k := i+j−1;
      q := fi ÷ k;  r := fi − q × k;
      a[i, j] := q × gi + (r×gi) ÷ k
    end
  end
end testmx
```

CERTIFICATION OF ALGORITHM 56 [S21] COMPLETE ELLIPTIC INTEGRAL OF THE SECOND KIND

[J. R. Herndon, *Comm. ACM 4*, (Apr. 1961), 180]

GERHARD MEIDELL LARSSEN (Recd. 9 Aug. 1965)
Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen mit Rechengruppe der Luftfahrt, Technische Hochschule, Stuttgart, Germany

Algorithm 56 was run on a UNIVAC 1107 using the UNIVAC 1107 ALGOL 60 compiler (dated January 25, 1965). The single-precision floating-point arithmetic of this translator carries eight significant digits.

Two syntactical errors were removed from the algorithm:

1. The line

$$ELLIPTIC\ 2 := (((0.040905094 \times t +$$

was changed to

$$ELLIPTIC\ 2 := ((0.040905094 \times t +$$

2. The function *log* was changed to *ln*.

In addition, the statement

$$t := 1 - k \times k$$

was removed from the algorithm and the complementary parameter itself used as input to the procedure:

real procedure *ELLIPTIC* 2 (t); **value** t; **real** t;

to avoid cancellation error for values of k near 1. [While the use of t as input parameter is good computationally, the name of the procedure is then slightly misleading.—J.G.H.]

Several values of the complete elliptic integral of the second kind were computed for $1 \geq t > 0$. The maximum error was found to be about $7_{10}-7$, compared with A. M. Legendre, Tafeln der Elliptischen Normalintegrale, Stuttgart, 1931. For $t = 0$ an error exit from the *ln* routine takes place.