Algorithms

G. E. FORSYTHE, Editor

ALGORITHM 234
POISSON-CHARLIER POLYNOMIALS [S23]

J. M. S. SIMÕES PEREIRA (Recd. 6 Jan. 1964)
Gulbenkian Scientific Computing Center, Lisboa, Portugal

```
real procedure PCpolynomial (x, n, a);
  integer n; real x, a;
comment PCpolynomial computes values of the Poisson-
  Charlier polynomial p_n(x) defined by L. Carlitz, Characteriza-
  tion of certain sequences of orthogonal polynomials, Portugaliae
  Mathematica 20 (1961), 43-46:

      p_n(x) = a^{n/2}(n!)^{-1/2} Σ_{r=0}^{n} (-1)^{n-r} C(n,r) r! a^{-r} C(x,r).

  In this algorithm u stands for the successive terms of the summa-
  tion, s stands for the sum of these terms and all other symbols
  possess evident meanings. Clearly each term of the summation
  is obtained from the preceding one by the indicated multipli-
  cation;
begin
  integer j; real u, s, c;
  u := (-1) ↑ n;
  s := u;
  c := 1;
  for j := 1 step 1 until n do c := c × j;
  for j := 0 step 1 until n - 1 do
    begin u := - u × (n - j) × (x - j)/(a × (j + 1)); s :=
      s + u end;
  PCpolynomial := sqrt(a ↑ n/c) × s
end PCpolynomial
```

ALGORITHM 235
RANDOM PERMUTATION [G6]

RICHARD DURSTENFELD (Recd. 2 Jan. 64)
General Atomic, San Diego 12, Calif.

```
procedure SHUFFLE (a, n, random);
  value n; integer n; real procedure random; integer
    array a;
begin
  comment SHUFFLE applies a random permutation to the
    sequence a[i] where i = 1, 2, ... , n. The procedure random is
    supposed to supply a random element from a large population
    of real numbers uniformly distributed over the open unit
    interval 0 < r < 1. The array a is declared to be integer but
    actually it suffices for its type to agree with that of the vari-
    able b (in the procedure body);
  integer i, j; real b;
  for i := n step - 1 until 2 do
    begin j := entier (i × random + 1);
          b := a[i]; a[i] := a[j]; a[j] := b
    end loop i
end SHUFFLE
```

Note. Numbers in brackets following Algorithm titles indicate the subject category for the algorithm, based on the Modified SHARE Classification listing given in the March, 1964 issue of the *Communications of the ACM*.

REMARK ON ALGORITHM 60 [D1]
ROMBERG INTEGRATION [F. L. Bauer, *Comm. ACM 4* (June 1961) 255; *5* (Mar. 1962), 168; *5* (May 1962), 281]

HENRY C. THACHER, JR.* (Recd. 20 Feb. 1964 and 23 Mar. 1964)
Argonne National Laboratory, Argonne, Ill.

* Work supported by the U. S. Atomic Energy Commission.

The Romberg integration algorithm has been used with great success by many groups [1, 2], and appears to be among the most generally reliable quadrature methods available. It is, therefore, worth pointing out that it is not entirely foolproof, and that a significant class of integrands exists for which the extrapolated values are poorer estimates of the integral than the corresponding trapezoidal sums.

The validity of the Romberg procedure depends upon the possibility of expanding the error of the trapezoidal rule in powers of h^2, where h is the stepsize. One expansion of this type is the Euler-Maclaurin sum formula. An alternative expression may be obtained from the Fourier series expansion. The coefficients of h^{2r} in the Euler Maclaurin formula are proportional to the difference of the values of the $(2r+1)$-th derivative at the two ends of the range. Thus, any integral for which the odd derivatives of the integrand either vanish or are equal at the limits will not be improved by Romberg extrapolation. Among the common examples of such integrals are integrals of periodic functions over a period and integrals for which the derivatives vanish at both limits. An example of the last type is the integral approximation to the modified Hankel function [3], $e^x K_p(x) = \int_0^L e^{x(1-\cosh t)}\cosh(pt)dt$, where L is taken so large that the contribution of the integral from L to ∞ may be neglected. Several other examples are given under the heading "Exceptional cases" by Bauer, Rutishauser and Stiefele [7]. This paper is among the most extensive discussions of the Romberg method in English.

The algorithm also fails when the expansion of the error term contains other powers of h along with the even ones. Rutishauser [4] discusses estimating integrals of the form $\int_0^a f(x)\,dx = \int_0^a (\varphi(x)/\sqrt{x})\,dx$. If such integrals are estimated by the trapezoidal rule, assigning the value 0 to $f(0)$, the error may be expressed in the form $\sum c_k h^{2k} + \sqrt{h}\sum d_k h^k$. Although the standard Romberg extrapolation fails when applied to this sequence of estimates, Rutishauser presents a modified procedure which is effective.

The extrapolation is also invalid when the integrand is discontinuous, although this exception is trivial from the computational standpoint.

It has also been pointed out [5, 6] that the Romberg procedure may amplify round-off errors. The losses, while significant, do not appear prohibitive for most applications.

REFERENCES:

1. THACHER, H. C., JR. Certification of algorithm 60. *Comm. ACM 5* (Mar. 1962), 168.
2. BUCHNER, K. H. Certification of algorithm 60. *Comm. ACM 5* (May, 1962), 281.
3. FETTIS, H. E. Algorithm 163, modified Hankel function. *Comm. ACM 6* (Apr. 1963), 161–2; *6* (Sep. 1963), 522.

4. RUTISHAUSER, H. Ausdehnung des Rombergschen Prinzips. *Numer. Math. 5* (1963), 48–54.
5. MCKEEMAN, W. M. Personal communication, Sept. 1963.
6. ENGELI, M. Personal communication, Jan. 1964.
7. BAUER, F. L., RUTISHAUSER, H., AND STIEFELE, E. New aspects in numerical quadrature. Proc. Symp. Appl. Math 15, 1963, 199–218.

CERTIFICATION OF ALGORITHM 128 [C6] SUMMATION OF FOURIER SERIES [M. Wells, *Comm. ACM 5* (Oct. 1962), 513]

HENRY C. THACHER, JR.* (Recd. 18 Mar. 1964)
Argonne National Lab., Argonne, Ill.

* Work supported by the U.S. Atomic Energy Commission

The body of *Fourier* was transcribed for the Dartmouth SCALP translator for the LGP-30 computer. After uniformizing the spelling of *zeros* (lines 5 and 9 in the procedure body), the program compiled and ran without difficulty.

In the procedure statement for *Fourier*, the actual parameter corresponding to X should be an expression depending on the actual parameter corresponding to r.

The SCALP program was tested for the finite series:

$$A = \sum_{r=0}^{n-1} \cos rw = \frac{\sin ((n-1)w/2)}{\sin (w/2)} \cos (nw/2) + 1$$

$$B = \sum_{r=0}^{n-1} \sin rw = \frac{\sin ((n-1)w/2)}{\sin (w/2)} \sin (nw/2)$$

for $w = 0.1, 0.2, 0.5$ and 1.0, and for $n = 1(1)51$. Although the algorithm appears to be numerically correct, the results showed evidence of serious numerical instability, particularly for small values of w. For $w = 0.1$, and $n = 51$, the error in A was .00109, and in B, $-.00231$. Since the largest A for $n < 51$ is 10.5, and the largest B about 20, the best result obtainable with the 7+ significant digit arithmetic of the SCALP system is about .00001. For comparison, a program summing the same series using a forward recurrence based on the addition formulas for the sine and cosine gave errors of .00012 and $-.00018$. It was, however, only about half as fast.

REMARK ON ALGORITHM 135 [F4] CROUT WITH EQUILIBRATION AND ITERATION [W. M. McKeeman, *Comm. ACM 5* (Nov. 1962), 555–557, 559]

WILLIAM MARSHALL MCKEEMAN (Recd. 1 Apr. 1964)
Computation Center, Stanford University, Stanford, Calif.

The following corrections to the published algorithm are recommended:

1. Two lines above the bottom line of procedure *SOLVE* one must change

$$y[k] := t \quad \text{to} \quad y[k] := t/A[k,k]$$

2. In procedure *EQUILIBRATE*, all occurrences of the subscript k must be changed to j.

3. The statement *cnr* := 1.0 should be added at the start of the body of procedure *LINEARSYSTEM*, so that *cnr* will have a value the first time it is used.

4. Line 19 from the end of *LINEARSYSTEM* should be changed from

if *normdy* = 0 **then begin** *cnr* := 1.0; **go to** *enditer* **end**;

to read

if *normdy* = 0 **then go to** *enditer*;

This correction makes sure that *cnr* retains a reasonable value in case *normdy* should be 0 for some column.

5. The symbol "-" must be removed from the parameter delimiters in the declarations of procedures *LINEARSYSTEM*, *RESIDUALS* and *SOLVE*.

6. Four lines above the bottom line of procedure *LINEARSYSTEM*, delete the first occurrence of *X*[*i*,*k*] :=

7. In the third line of the heading of procedure *IP*2, the parameter delimiter

) extra therm:(

should be changed to

) extra term:(

CERTIFICATION OF ALGORITHM 170 [F3] REDUCTION OF A MATRIX CONTAINING POLYNOMIAL ELEMENTS [P. E. Hennion, *Comm. ACM 6* (April 1963), 165; *6* (Aug. 1963), 450]

KAREN B. PRIEBE (Recd. 18 Dec. 1963 and 18 Feb. 1964)
Woodward Governor Co., Rockford, Ill.

Algorithm 170 was translated into FAST for the NCR 315 and gave satisfactory results with the following corrections:

1. **real procedure** ... **integer** *NCOL*, *N*; should be replaced by

procedure *POLYMATRIX* (*A*, *NCOL*, *N*, *COE*, *NP*1);
value *NCOL*, *N*; **real array** *A*, *COE*;
integer *NCOL*, *N*, *NP*1;

2. At the end of the first comment add:
The global integer procedure *MAX* is assumed and furnishes the maximum of two integers.

3. **integer** *i*, *j*, *k*, ... *COE*[1:*M*];
should be replaced by
integer *i*, *j*, *k*, *j*1, *j*2, *j*3, *j*4, *j*5, *j*6, *j*7, *j*8, *j*9, *j*10, *j*11, *M*;
array *C*1, *C*2[1:*N*×*NCOL*+1];

4. Immediately after *start*: the statement

*NP*1 := *N* + 1;

should be added, and the third line after *start*: i.e.,

for *k* := 1 **step** 1 **until** *M* **do begin**

should be replaced by

for *k* := 1 **step** 1 **until** *NP*1 **do begin**

5. The third line after *L*10: i.e.,

for *k* := 1 **step** 1 **until** *M* **do** ...

should be replaced by

for *k* := 1 **step** 1 **until** *j*7 **do** ...

The last two changes simply shorten both of the indicated FOR statements.

[EDITOR'S NOTE. In addition to the above corrections, we have two comments on the Remark on Algorithm 170 by Hennion, *loc. cit.*, p. 450:

First, the semicolon at the end of the first line after *L*0 MUST be removed.

Second, correction (4) is irrelevant.

The referee confirms that a transcription into Burroughs Extended ALGOL of the program as corrected by Mrs. Priebe runs on the B5000.—G.E.F.]