Algorithms

J. H. WEGSTEIN, Editor

ALGORMITH 214
q-BESSEL FUNCTIONS $I_n(t)$

J. M. S. Simões Pereira
Gulbenkian Scientific Computing Center, Lisbon, Portugal

```
procedure qBessel (t, q, n, j, s); integer n, j; real t, q, s;
  array s;
```

comment This procedure computes values of any q-Bessel function $I_n(t)$ for n integer (positive, negative or zero) by the use of the well-known expansion

$$I_n(t) = \sum_{k=0}^{\infty} \frac{q^{\frac{1}{2}k(k-1)+\frac{1}{2}(n+k)(n+k-1)} t^{n+2k}}{(q)_k (q)_{n+k}}$$

where $|q| < 1$, $(q)_n = (1-q)(1-q^2)\cdots(1-q^n)$, $(q)_0 = 1$ and $1/(q)_{-n} = 0$ $(n=1, 2, 3, \cdots)$. (See L. Carlitz, The product of q-Bessel functions, *Port. Math. 21* (1962), 5–9.) Moreover, j denotes the number of terms (at least 2) retained in the summation, and $s[i]$ stands for the sum of the first $i+1$ terms of the expansion. This procedure has been translated into Fortran for the IBM 1620 and run successfully;

```
begin integer k, m, p; real c, u; m := abs(n); c := 1; if
  n = 0 then go to A;
for p := 1 step 1 until m do c := c × (1−q↑p); if n < 0 then
  go to B;
A: u := q ↑ (n×(n−1)/2) × (t↑n)/c; s[0] := u;
for k = 1 step 1 until j do
begin u := u × q ↑ (n+2×k−2) × (t↑2)/((1−q↑k)×(1−q↑(n+k)));
  s[k] := s[k−1] + u end;
B: u := q↑((m−1)×m/2) × t ↑ (n+2×m)/c; s[m] := u;
for k := m + 1 step 1 until j do
begin u := u × q ↑ (n+2×k−2) × (t↑2)/((1−q↑k)(1−q↑(n+k)));
  s[k] := s[k−1] + u end
end
```


ALGORITHM 215
SHANKS

Henry C. Thacher, Jr.*
Argonne National Laboratory, Argonne, Ill.

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

```
procedure Shanks (nmin, nmax, kmax, S);
value nmin, nmax, kmax;
integer nmin, nmax, kmax;
array S;
```

comment This procedure replaces the elements $S[nmin]$ through $S[nmax-2\times kmax]$ of the array S by the $e[kmax]$ transform of the sequence S. The elements $S[nmax-2\times kmax+1]$ through $S[nmax-1]$ are destroyed. The $e[k]$ transforms were discovered by D. Shanks (*J. Math. Phys. 34* (1955), 1–42). $e[1]$ is equivalent to the *(delta)* ↑ 2 transformation. The $e[k]$ transforms are particularly valuable in estimating B in sequences which may be written in the form $S[n] = B + \sum a[i] \times q[i] \uparrow n$ $(i=1, 2, \cdots, k)$.

The transformation is carried out by the epsilon algorithm (Wynn, P., *M.T.A.C 10* (1956), 91–96). Algol procedures for applying the algorithm to series of complex terms are given by Wynn (*BIT 2* (1962), 232–255).

The body of this procedure has been tested using the Dartmouth Self-Contained Algol Processor for the LGP-30 computer. It gave the following results on the sequence for the smaller zero of the Laguerre polynomial, $L[2](x)$:

n	$S[n]$	$e[1](S[n])$	$e[2](S[n])$	$e[1]^2(S[n])$
0	0.0000000	0.5714285	0.5857432	0.5857616
1	0.5000000	0.5851059	0.5857854	0.5857859
2	0.5625000	0.5857318	0.5857861	0.5857861
3	0.5791016	0.5857816		
4	0.5838396	0.5857859		
5	0.5852172			
6	0.5856198		*True Value*	0.5857864375

These results are in satisfactory agreement with those given by by Wynn (1956);

```
begin integer j, k, limj, limk, two kmax;
  real T0, T1;
  two kmax := kmax + kmax;
  limj := nmax;
for j := nmin step 1 until limj do
  begin T0 := 0;
  lim := j − nmin;
  if limk > two kmax then limk := two kmax⟩  limk := limk − 1;
  for k := 0 step 1 until limk do
    begin T1 := S [j−k] − S [j−k−1];
      if T1 ≠ 0 then T1 := T0 + 1/T1 else
      if S [j−k] = ₁₀99 then T1 := T0 else
        T1 := ₁₀99;
      comment ₁₀99 may be replaced by the largest number
        representable in the computer;
      T0 := S [j−k−1];
      S [j−k−1] := T1
    end for k
  end for j
end Shanks
```

ALGORITHM 216
SMOOTH

RICHARD GEORGE*
Argonne National Laboratory, Argonne, Ill.

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

```
procedure SMOOTH (Data) which is a list of length: (n);
  integer n; real array Data;
  begin
    comment This procedure accomplishes fourth-order smooth-
      ing of a list using the method given by Lanczos, Applied
      Analysis (Prentice-Hall, 1956). This algorithm requires only
      one additional list for temporary storage;
    real Factor, Top; integer Max I, I, J; array Delta [1 : n];
    Factor := 3.0/35.0;
    Max I := n − 1;
    for I := 1 step 1 until Max I do
      Delta [I] := Data [I+1] − Data [I];
    for J := 1 step 1 until 3 do
      begin
        Top := Delta [1];
        Max I := Max I − 1;
        for I := 1 step 1 until Max I do
          Delta [I] := Delta [I+1] − Delta [I]
      end;
    Max I := n − 2;
    for I := 3 step 1 until Max I do
      Data [I] := Data [I] − Delta [I−2] × Factor;
    Data [1] := Data [1] + Top/5.0 + Delta [1] × Factor;
    Data [2] := Data [2] − Top × 0.4 − Delta [1]/7.0;
    Data [n] := Data [n] − Delta [n−3]/5.0 + Delta [n−4] × Factor;
    Data [n−1] := Data [n−1] + Delta [n−3] × 0.4 − Delta
      [n−4]/7.0
  end;
```

CERTIFICATION OF ALGORITHM 8
EULER SUMMATION [P. Naur et al. *Comm. ACM* 3, May 1960]

HENRY C. THACHER, JR.*
Argonne National Laboratory, Argonne, Ill.

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

The body of *euler* was tested on the LGP-30 computer using the Dartmouth SCALP translator. No errors were detected.

The program gave excellent results when used to derive the coefficients for the expansion of $ln\ (1+x)$ in shifted Chebyshev polynomials from the first ten terms of the power series. For $n = 0, 1, 2, 3, 4$, the coefficient of x^i in the power series was multiplied by the coefficient of $T_n{}^*(x)$ in the expression of x^i in terms of the $T_n{}^*(x)$. The product, for $i = 1, 2, \cdots, 10$ was used as $fct(i)$ in the program. Results for $n = 0$ were as follows:

i	$fct(i)$	ds	sum
1	+0.50000000	—	—
2	−0.18750000	+0.07812500	+0.3281250
3	+0.10416667	+0.05729166	+0.3854167
4	−0.068359375	−0.005940758	+0.3794759
5	+0.049218750	−0.001928713	+0.3775471
6	−0.037597656	−0.001357019	+0.3761900
7	+0.029924665	+0.0001742393	+0.3763642
8	−0.024547577	+0.0000571311	+0.3764212
9	+0.020607842	+0.0006395427	+0.3764607
10	−0.017619705	−0.0000055069	+0.3764551
True Value[1]			+0.3764528129.....

[1] Clenshaw, C. W., *Chebyshev Series for Mathematical Functions*. National Physical Laboratory Math Tables, Vol. 5, London, H.M.S.O. (1962).

Errors less than 0.2×10^{-5} were also found for $n = 1, 2, 3, 4, 5, 6, 7, 8$ and 9.

This technique appears to be a useful supplement to direct telescoping (Algorithms 37 and 38) and to the methods recommended by Clenshaw[1], for slowly convergent power series.

REMARK ON ALGORITHM 77
INTERPOLATION, DIFFERENTIATION, AND INTEGRATION [P. E. Hennion, *Comm. ACM 5*, Feb. 1962]

P. E. HENNION
Giannini Controls Corp., Berwyn, Penn.

It was brought to my attention through the CERTIFICATION OF ALGORITHM 77 AVINT (V. E. Whittier, *Comm. ACM*, June, 1962) that restrictions on the upper and lower limits of integration existed, i.e., (1) $x1o \geqq xa(1)$, (2) $xup \geqq xa(nop)$. To remove these restrictions the following two changes should be made.

1. Replace the two lines starting at line $L12$: and ending after the statement $ib := 2$; with the following code:

```
L12: sum := 0;  sy1 := x1o; ib := 2; ju1 := nop;
     for ia := 1 step 1 until nop do begin
     if xa [ia] ≧ x1o then go to L17;  ib := ib + 1; end;
L17: for ia := 1 step 1 until nop do begin
     if xup ≧ xa [ju1] then go to L18; ju1 := ju1 − 1;  end;
L18: ju1 := ju1 − 1;
```

2. Change line $L13$: to read

```
L13: if jm ≠ ib then go to L14;
```

GOSDEN et al.—*cont'd from page 661*

tions. Currently there appears to be no rival to COBOL—only variations seem to be considered; broadly speaking, Czechoslovakia tends towards ALGOL and Poland towards FORTRAN for a scientific language. In particular, we found that the Poles were well acquainted with most of the latest developments in the West and made frequent short and some extensive visits to computing centers in Europe and America.

Acknowledgments. We wish to thank the national standardization bodies of Czechoslovakia, Poland and America for the arrangements that were made for our visit, and the research institutes for the preliminary preparation and valuable discussions held with them.

REFERENCES

1. BROMBERG, H. Standardization of programming languages. *Datamation 9*, 8 (Aug. 1963), 41–50.
2. GORN, S., ED. Structures of standards - processing organizations in the computer area. *Comm. ACM 6*, 6 (June 1963), 294–305.
3. WARE, W. H., ED. Soviet computer technology—1959. *Comm. ACM 3*, 3 (Mar. 1960), 131–166.
4. BLACHMAN, N. M. The state of digital computer technology in Europe. *Comm. ACM 4*, 6 (June 1961), 256–265.
5. SVOBODA, A., AND VALACH, M. Decimal arithmetic unit. (In English), *Stroje Na Zpracova'ni' Informaci'*, Sbomi'k VIII, 1962.
6. LUKASZEWICZ, L. SAKO—an automatic coding system. *Annual Review in Automatic Programming*, Vol. 2, p. 161, 1961.
7. OPLER, A. Automatic program translation. *Datamation 9*, 5 (May 1963), 45–48.