ALGORITHM 21
BESSEL FUNCTION FOR A SET OF INTEGER ORDERS

W. Börsch-Supan
National Bureau of Standards, Washington 25, D. C.

```
procedure BESSELSETINT (x, n, ε, J) ; value x, n, ε ;
    real x, ε ; integer n ; real array J ;
comment: This procedure computes the values of the Bessel
    functions Jp(x) for real argument x and the set of all integer
    orders from 0 up to n and stores these values into the array J,
    whose subscript bounds should include the integers from 0 up
    to n.  n must be nonnegative.
      The computation is done by applying the recursion formula
    backward from p = k down to p = 0 as described in MTAC 11
    (1957), 255-257.  k is chosen to yield errors less than 10^-5
    approximately after the first application of the recursion. The
    recursion is repeated with a larger k until the difference be-
    tween the results of the two last recursions doesn't exceed the
    given bound ε > 0. The steps in increasing k are chosen in
    such a way that the errors decrease at least by a factor of
    approximately 10^-5. There is no protection against overflow. ;
begin real dist, rec0, rec1, rec2, sum, max, err ;
    integer k, p ; Boolean s ; real array Jbar[0:n] ;
    if x = 0 then
    begin J[0] := 1 ; for p := 1 step 1 until n do J[p] := 0 ;
      go to Exit
    end ;
    dist := if abs(x) ≧ 8 then 5 × abs(x) ↑ (1/3) else 10 ;
    k := entier ((if abs(x) ≧ n then abs(x) else n) + dist) + 1 ;
    s := false ;
Rec: rec0 := 0 ; rec1 := 1 ; sum := 0 ;
      for p := k step -1 until 1 do
      begin J[if p > n + 1 then n else p - 1] := rec2 :=
                2 × p/x × rec1 - rec0 ;
            if p = 1 then sum := sum + rec2
            else if p ÷ 2 × 2 ≠ p then sum :=
              sum + 2 × rec2 ;
            rec0 := rec1 ; rec1 := rec2
      end recursion ;
Norm: for p := 0 step 1 until n do J[p] := J[p]/sum ;
      if s then
      begin max := 0 ;
            for p := 0 step 1 until n do
            begin err := abs (J[p] - Jbar[p]) ;
                  if err > max then max := err
            end maximum error ;
            if max ≦ ε then go to Exit
      end then
      else s := true ;
      for  p := 0 step 1 until n do Jbar[p] := J[p] ;
      k := entier (k + dist) ;
      go to Rec ;
Exit: end BESSELSETINT
```

CERTIFICATION OF ALGORITHM 21 [S17]
BESSEL FUNCTION FOR A SET OF INTEGER ORDERS

[W. Börsch-Supan, *Comm. ACM 3* (Nov. 1960), 600]

J. Stafford (Recd. 16 Nov. 1964)
Westland Aircraft Ltd., Saunders-Roe Division, East Cowes, Isle of Wight, Eng.

If this procedure is used with a combination of a moderately small argument and a moderately large order, the recursive evaluation of *rec*2 in the last line of the first column can easily lead to overflow. This occurred, for instance, in trying to evaluate $J_{10}(0.01)$.

The following alterations correct this:

(i) Declare a **real** variable *z* and an **integer** variable *m*;

(ii) After line *rec* insert:

```
z := MAX/4 × abs (x/k);
comment MAX is a large positive number approaching in
  size the largest number which can be represented. The nu-
  merical value of MAX/4 is written into the procedure;
```

(iii) At the end of the first column insert:

```
if abs(rec2) > z then
begin
  rec1 := rec1/z;  rec2 := rec2/z;  sum := sum/z;
  for m := n step -1 until p - 1 do J[m] := J[m]/z
end;
```

With these alterations the procedure was run on a National-Elliott 803, for $x = -1, 0, 0.01, 1, 10$ and $n = 0, 1, 2, 10, 20$. The results agreed exactly with published seven-place tables.

[See also Algorithm 236, Bessel Functions of the First Kind (*Comm. ACM 7* (Aug. 1964), 479) which is not restricted to integer values. Although it is a much more complicated program, Algorithm 236 is slightly faster than Algorithm 21 as corrected, at least in some cases.—Ed.]