ALGORITHM 22
RICCATI-BESSEL FUNCTIONS OF FIRST AND SECOND KIND

H. Oser
National Bureau of Standards, Washington 25, D. C.

```
procedure RICCATIBESSEL (x, n, eps, S, C) ;
   value x, n, eps ;
   real x, eps; integer n; real array S, C ;
comment: RICCATIBESSEL computes Sk(x) = (πx/2)^½ Jk+½(x)
   and Ck(x) = -(πx/2)^½ Yk+½(x) for real x ≠ 0 and all integer
   values of k from 0 through n with a prescribed (absolute)
   accuracy eps. The computation is done by using the recursion
   relations of the cylinder functions. For abs(x) > n both Sk(x)
   and Ck(x) are computed by using the recursions for ascending
   orders. For n > abs(x) the functions Sk(x) are obtained by
   using the recursion in descending orders. (See STEGUN-
   ABRAMOWITZ, MTAC 11, 1957, 255-257). Reaching out two
   different intervals beyond the order n, the two vectors Sk^1(x)
   and Sk^2(x) are checked if the maximum component of their
   difference meets the tolerance eps. If this is not the case a
   maximum of 10 iterations is set up to achieve the required
   absolute accuracy. Initial values Skmax and Skmax-1 for the
   backward iteration are computed from the corresponding
   values Ckmax-1 and Ckmax. No check of accuracy is done in
   case n < abs(x). Both Ck(x) and Sk(x) are affected in this
   subcase by errors of the same order of magnitude as the sub-
   routines for sin (x) and cos (x) ;
 begin real r1, r2, r3, r4, r5, r6, step, acc, max, a, b, d1, d2 ;
    integer i, k, l, imax ;
    real array W[0:n] ;
    switch P := initial, improve ;
    acc: = ₁₀6 ;
    step: = ₁₀3 ;
    imax: = 10 ;
    comment: These constants may be chosen differently, but
       caution has to be taken because of overflow. acc sets an
       initial iteration to give roughly a 6-place accuracy.
       Subsequent iterations should improve the result to 3 more
       places each ;
    i := 1 ;
    if x = 0 then go to exit1 ;
    if n < abs(x) then
 case1: begin r1 := -sin(x) ; r2 := r4 := C[0] := cos(x) ;
             r5 := S[0] := sin(x) ;
             for k := 1 step 1 until n do
             begin C[k] := r3 := (2×k-1) × r2/x - r1 ;
                   S[k] := r6 := (2×k-1) × r5/x - r4 ;
                   r1 := r2 ; r2 := r3 ;
                   r4 := r5 ; r5 := r6
             end k ; go to finish
       end case1 ;
 case2: l := 1 ; r1 := -sin(x) ; r2 := C[0] := cos(x) ;
       for k := 1 step 1 until n do
       begin C[k] := r3 := (2×k-1) × r2/x - r1 ;
             r1 := r2 ;
             r2 := r3
       end ;
       a := n ;
 loop: for k := 1+n step 1 until if abs(x) ≦ 11
          then 12+a else 2×a+1 do
       begin r3 := (2×k-1) × r2/x - r1 ;
             if abs(r3/C[n]) > acc then go to S ;
             r1 := r2 ;
             r2 := r3 ;
             comment: This loop is most liable to cause
                      overflow ;
       end loop ;
       k := if abs(x) ≦ 11 then 12+a else 2×a + 1 ;
       r2 := r1 ;
 S:    r6 := x ↑ 2/(4×k ↑ 2×r2) ;
       r5 := 1/r3 ;
       go to P[l] ;
 initial: for k := k step -1 until 2 do
       begin W[if k>n+2 then n else k-2] := r4 :=
                (2×k-1) × r5/x - r6 ;
             r6 := r5 ;
             r5 := r4
       end ;
       d1 := r5/x - r6 ;
       d2 := if abs(W[0]) ≧
          abs(d1) then sin(x)/W[0] else cos(x)/d1 ;
       for k := 0 step 1 until n do
          W[k] := d2×W[k] ;
       acc := step × acc ;
       l := 2 ;
       a := a + step ↑ (1/3) ;
       r2 := C[n] ;
       r1 := C[n-1] ;
       go to loop ;
 improve: for k := k step -1 until 2 do
       begin S[if k > n+2 then n else k-2] := r4 :=
                (2×k-1) × r5/x-r6 ;
             r6 := r5 ;
             r5 := r4
       end k ;
       d1 := r5/x - r6 ;
       d2 := if abs(S[0]) ≧
          abs(d1) then sin(x)/S[0] else cos(x)/d1 ;
       max := 0 ;
       for k := 1 step 1 until n do
       begin S[k] := d2×S[k] ;
             b := abs(S[k] - W[k]) ;
             if b > max then max := b
       end ;
       if max < eps then go to finish ;
       for k := 0 step 1 until n do W[k] := S[k] ;
          acc := step × acc ;
          if i ≧ imax then go to exit2 ;
          i = i+1 ; a := a + step ↑ (1/3) ;
          r2 := C[n] ; r1 := C[n-1] ; go to loop ;
 exit1: go to finish ; comment: x = 0 ;
 exit2: go to finish ;
          comment: maximum number of iterations reached ;
 finish: end RICCATIBESSEL
```

CERTIFICATION OF ALGORITHM 22 [S17] RICATTI-BESSEL FUNCTIONS OF FIRST AND SECOND KIND [H. Oser, *Comm. ACM 3* (Nov. 1960), 600]

THOMAS BRAY (Recd. 9 Mar. 1970)
Boeing Scientific Research Laboratories, Seattle, WA 98124

KEY WORDS AND PHRASES: Ricatti-Bessel functions, Bessel functions of fractional order, spherical Bessel functions
CR CATEGORIES: 5.12

The procedure was translated into FORTRAN IV and run on an IBM 360/44 using double precision arithmetic (15 significant decimal digits). One error was discovered in the algorithm. The tenth line following the line with the label "*improve*" reads:

for k := 1 **step** 1 **until** n **do**

This line should read:

for k := 0 **step** 1 **until** n **do**

The results $S_k(x)/x$ and $-C_k(x)/x$ were computed using this correction and compared with Tables 10.1, 10.2 and 10.5 of [1]. The results agreed to the number of digits given in the tables for:

x	k
0.1	0(1)8
0.5	0(1)8
1.0	0(1)20
2.0	0(1)8
5.0	0(1)50
7.5	0(1)8
10.0	0(1)50
50.0	0(1)100
100.0	0(1)100

REFERENCES:
1. ABRAMOWITZ, M., AND STEGUN, I. A. *Handbook of Mathematical Functions*. Appl. Math. Ser. 55, Nat. Bur. Standards US Govt. Print. Off., Washington, D.C., 1964.