ALGORITHM 44
BESSEL FUNCTIONS COMPUTED RECURSIVELY

MARIA E. WOJCICKI
RCA Digital Computation and Simulation Group, Moorestown, New Jersey

```
procedure Bessfr(N, FX, LX, Z) Result: (J, Y);
    value LX, FX, N;
    real FX, LX, Z; real array J, Y; integer N;
comment Bessel Functions of the first and second kind, JP(X)
  and YP(X), integral order P, are computed by recursion for
  values of X, FX ≦ X ≦ LX, in steps of Z. The functions are
  computed for values of P, 0 ≦ P ≦ N. M[SUB], the initial
  value of P being chosen according to formulae in Erdelyi's
  Asymptotic Expansions. The computed values of JP(X) and
  YP(X) are stored as column vectors for constant argument in
  matrices J, Y of dimension (N+1) by entier ((LX − FX)/Z + 1);
begin real PI, X, GAMMA, PAR, LAMDA, SUM, SUM1;
    integer P, SUB, MAXSUB;
            PI := 3.14159265;
            GAMMA := .57721566;
            PAR := 63.0 − 1.5 × ℓn (2 × PI);
            MAXSUB := entier ((LX − FX)/Z);
begin real array JHAT [0:N, 0:MAXSUB];
    integer array M[0:MAXSUB];
          SUB := 0;
        for X := FX step Z until LX do
begin if (X > 0) ∧ (X < 10) then M [SUB] := 2 × entier (X) + 9
  else
begin real ALOG;
      ALOG := (PAR − 1.5 × ℓn (X))/X;
      M [SUB] := entier (X × (exp (ALOG) + exp
        (−ALOG))/2) end;
      if N > M [SUB] then
begin for P := M [SUB] + 1 step 1 until N do
      J [P, SUB] := 0 end;
      JHAT [M [SUB], SUB] := 10 ↑ (−9);
comment Having set the uppermost ĴP(X) to a very small
  number we are now going to compute all the ĴP(X) down to
  P = 0;
    for P := M [SUB] step −1 until 1 do
    JHAT [P−1, SUB] := 2 × P/X × JHAT [P, SUB] −JHAT
      [P+1, SUB];
    SUM := SUM1 := 0;
    for P := 2 step 2 until (M [SUB] ÷ 2) do
    SUM := SUM + JHAT [P, SUB];
    LAMDA := JHAT [0, SUB] + 2 × SUM;
    for P := 0 step 1 until N do
    J [P, SUB] := JHAT [P, SUB] /LAMDA;
comment JP(X) have been computed by use of ĴP(X);
    for P := 2 step 2 until (M [SUB] ÷ 2) do
    SUM1 := SUM1 + (−1) × (−1) ↑ P ÷ J [2 × P, SUB]
      /2/P;
    Y [0, SUB] := 2/PI × (J [0, SUB] × (GAMMA + ℓn(X/2))
      + 4 × SUM1);
    for P := 0 step 1 until (M[SUB]−1) do
    Y [P+1, SUB] := (−2/PI/P + J [P+1, SUB] × Y [P,
      SUB])/J [P, SUB];
    SUB := SUB + 1 end end end
```