ALGORITHM 61
PROCEDURES FOR RANGE ARITHMETIC

ALLAN GIBB*
University of Alberta, Calgary, Alberta, Canada

```
begin
  procedure RANGESUM (a, b, c, d, e, f);
    real a, b, c, d, e, f;
  comment The term "range number" was used by P. S. Dwyer,
  Linear Computations (Wiley, 1951). Machine procedures for
  range arithmetic were developed about 1958 by Ramon Moore,
  "Automatic Error Analysis in Digital Computation," LMSD
  Report 48421, 28 Jan. 1959, Lockheed Missiles and Space Divi-
  sion, Palo Alto, California, 59 pp. If a ≦ x ≦ b and c ≦ y ≦ d,
  then RANGESUM yields an interval [e, f] such that e ≦ (x + y)
  ≦ f. Because of machine operation (truncation or rounding) the
  machine sums a + c and b + d may not provide safe end-points
  of the output interval. Thus RANGESUM requires a non-local
  real procedure ADJUSTSUM which will compensate for the
  machine arithmetic. The body of ADJUSTSUM will be de-
  pendent upon the type of machine for which it is written and so
  is not given here. (An example, however, appears below.) It
  is assumed that ADJUSTSUM has as parameters real v and w,
  and integer i, and is accompanied by a non-local real procedure
  CORRECTION which gives an upper bound to the magnitude
  of the error involved in the machine representation of a number.
  The output ADJUSTSUM provides the left end-point of the
  output interval of RANGESUM when ADJUSTSUM is called
  with i = −1, and the right end-point when called with i = 1.
  The procedures RANGESUB, RANGEMPY, and RANGEDVD
  provide for the remaining fundamental operations in range
  arithmetic. RANGESQR gives an interval within which the
  square of a range number must lie. RNGSUMC, PNGSUBC,
  RNGMPYC and RNGDVDC provide for range arithmetic with
  complex range arguments, i.e. the real and imaginary parts
  are range numbers;
  begin
    e := ADJUSTSUM (a, c, −1);
    f := ADJUSTSUM (b, d, 1)
  end RANGESUM;
  procedure RANGESUB (a, b, c, d, e, f);
    real a, b, c, d, e, f;
  comment RANGESUM is a non-local procedure;
  begin
    RANGESUM (a, b, −d, −c, e, f)
  end RANGESUB;
  procedure RANGEMPY (a, b, c, d, e, f);
    real a, b, c, d, e, f;
  comment ADJUSTPROD, which appears at the end of this
  procedure, is analogous to ADJUSTSUM above and is a non-
  local real procedure. MAX and MIN find the maximum and
  minimum of a set of real numbers and are non-local;
  begin
    real v, w;
    if a < 0 ∧ c ≧ 0 then
1:  begin
        v := c; c := a; a := v; w := d; d := b; b := w
    end 1;
    if a ≧ 0 then
2:  begin
      if c ≧ 0 then
3:    begin
        e := a × c; f := b × d; go to 8
      end 3;
      e := b × c;
      if d ≧ 0 then
4:    begin
        f := b × d; go to 8
      end 4;
      f := a × d; go to 8
5:  end 2;
    if b > 0 then
6:  begin
      if d > 0 then
      begin
        e := MIN(a × d, b × c);
        f := MAX(a × c, b × d); go to 8
      end 6;
      e := b × c; f := a × c; go to 8
    end 5;
    f := a × c;
    if d ≦ 0 then
7:  begin
      e := b × d; go to 8
    end 7;
    e := a × d;
8:  e := ADJUSTPROD (e, −1);
    f := ADJUSTPROD (f, 1)
  end RANGEMPY;
  procedure RANGEDVD (a, b, c, d, e, f);
    real a, b, c, d, e, f;
  comment If the range divisor includes zero the program
  exists to a non-local label "zerodvsr". RANGEDVD assumes a
  non-local real procedure ADJUSTQUOT which is analogous
  (possibly identical) to ADJUSTPROD;
  begin
    if c ≦ 0 ∧ d ≧ 0 then go to zerodvsr;
    if c < 0 then
1:  begin
      if b > 0 then
2:    begin
        e := b/d; go to 3
      end 2;
      e := b/c;
3:    if a ≧ 0 then
4:    begin
        f := a/c; go to 8
      end 4;
      f := a/d; go to 8
    end 1;
    if a < 0 then
5:  begin
      e := a/c; go to 6
    end 5;
    e := a/d;
6:  if b > 0 then
7:  begin
      f := b/c; go to 8
    end 7;
    f := b/d;
```

```
8:  e := ADJUSTQUOT (e, −1);  f := ADJUSTQUOT (f,1)
end RANGEDVD;
procedure RANGESQR (a, b, e, f);
  real a, b, e, f;
comment ADJUSTPROD is a non-local procedure;
begin
      if a < 0 then
 1:     begin
          if b < 0 then
 2:       begin
            e := b × b;   f := a × a;   go to 3
          end 2;
          e := 0;  m := MAX (-a,b);  f := m × m;  go to 3
        end 1;
        e := a × a;  f := b × b;
 3:     ADJUSTPROD (e, −1);
        ADJUSTPROD (f, 1)
  end  RANGESQR;
  procedure  RNGSUMC (aL, aR, bL, bU, cL, cR, dL, dU, eL,
  eR, fL, fU);
    real aL, aR, bL, bU, cL, cR, dL, dU, eL, eR, fL, fU;
  comment  Rangesum is a non-local procedure;
  begin
    RANGESUM (aL, aR, cL, cR, eL, eR);
    RANGESUM (bL, bU, dL, dU, fL, fU)
  end  RNGSUMC;
  procedure  RNGSUBC (aL, aR, bL, bU, cL, cR, dL, dU, eL,
  eR, fL, fU);
    real aL, aR, bL, bU, cL, cR, dL, dU, eL, eR, fL, fU;
  comment  RNGSUMC is a non-local procedure;
  begin
    RNGSUMC (aL, aR, bL, bR, −cR, −cL, −dU, −dL, eL, eR,
    fL, fU)
  end  RNGSUBC;
  procedure  RNGMPYC (aL, aR, bL, bU, cL, cR, dL, dU, eL,
  eR, fL, fU);
    real aL, aR, bL, bU, cL, cR, dL, dU, eL, eR, fL, fU;
  comment  RANGEMPY, RANGESUB, and RANGESUM are
  non-local procedures;
  begin
    real L1, R1, L2, R2, L3, R3, L4, R4;
    RANGEMPY (aL, aR, cL, cR, L1, R1);
    RANGEMPY (bL, bU, dL, dU, L2, R2);
    RANGESUB (L1, R1, L2, R2, eL, eR);
    RANGEMPY (aL, aR, dL, dU, L3, R3);
    RANGEMPY (bL, bU, cL, cR, L4, R4);
    RANGESUM (L3, R3, L4, R4, fL, fU);
  end  RNGMPYC;
  procedure  RNGDVDC (aL, aR, bL, bU, cL, cR, dL, dU, eL,
  eR, fL, fU);
    real aL, aR, bL, bU, cL, cR, dL, dU, eL, eR, fL, fU;
  comment  RNGMPYC,  RANGESQR,  RANGESUM,  and
  RANGEDVD are non-local procedures;
  begin
    real L1, R1, L2, R2, L3, R3, L4, R4, L5, R5;
    RNGMPYC (aL, aR, bL, bU, cL, cR, −dU, −dL, L1, R1, L2,
    R2);
    RANGESQR (cL, cR, L3, R3);
    RANGESQR (dL, dU, L4, R4);
    RANGESUM (L3, R3, L4, R4, L5, R5);
    RANGEDVD (L1, R1, L5, R5, eL, eR);
    RANGEDVD (L2, R2, L5, R5, fL, fU)
  end  RNGDVDC
end
```

EXAMPLE

```
real procedure  CORRECTION (p);  real p;
comment  CORRECTION and the procedures below are in-
tended for use with single-precision normalized floating-point
arithmetic for machines in which the mantissa of a floating-point
number is expressible to s significant figures, base b. Limitations
of the machine or requirements of the user will limit the range of
p to b^m ≦ | p | < b^(n+1) for some integers m and n. Appropriate
integers must replace s, b, m and n below. Signal is a non-local
label. The procedures of the example would be included in the
same block as the range procedures above;
begin
    integer w;
    for w := m step 1 until n do
1:  begin
      if (b ↑ w ≦ abs (p)) ∧ (abs (p) < b ↑ (w + 1)) then
2:    begin
        CORRECTION := b ↑ (w+1−s);  go to exit
      end 2
    end 1;
    go to signal;
exit: end  CORRECTION;
real procedure  ADJUSTSUM (w, v, i);  integer i;
  real w, v;
comment  ADJUSTSUM exemplifies a possible procedure for use
with machines which, when operating in floating point addition,
simply shift out any lower order digits that may not be used. No
attempt is made here to examine the possibility that every digit
that is dropped is zero. CORRECTION is a non-local real pro-
cedure which gives an upper bound to the magnitude of the error
involved in the machine representation of a number;
begin
  real r, cw, cv, cr;
  r := w + v;
  if w = 0 ∨ v = 0 then go to 1;
  cw := CORRECTION (w);
  cv := CORRECTION (v);
  cr := CORRECTION (r);
  if cw = cv ∧ cr ≦ cw then go to 1;
  if sign (i × sign (w) × sign (v) × sign (r)) = −1 then go to 1;
  ADJUSTSUM := r + i × MAX (cw, cv, cr); go to exit;
1:  ADJUSTSUM := r;
exit: end  ADJUSTSUM;
real procedure  ADJUSTPROD (p, i);  real p;  integer i;
comment  ADJUSTPROD is for machines which truncate when
lower order digits are dropped. CORRECTION is a non-local real
procedure;
begin
    if p × i ≦ 0 then
1:  begin
      ADJUSTPROD := p;  go to out
    end 1;
    ADJUSTPROD := p + i × CORRECTION (p);
out:  end  ADJUSTPROD;
comment  Although ordinarily rounded arithmetic is preferable
to truncated (chopped) arithmetic, for these range procedures
truncated arithmetic leads to closer bounds than rounding does.
```

---

* These procedures were written and tested in the Burroughs 220 version of the ALGOL language in the summer of 1960 at Stanford University. The typing and editorial work were done under Office of Naval Research Contract Nonr-225(37). The author wishes to thank Professor George E. Forsythe for encouraging this work and for assistance with the syntax of ALGOL 60.