Algorithms

H. J. WEGSTEIN, Editor

ALGORITHM 77
INTERPOLATION, DIFFERENTIATION, AND INTEGRATION

PAUL E. HENNION
Grumman Aircraft Engineering Corporation, Bethpage, L. I., New York

```
real procedure AVINT (nop, jt, xarg, xlo, xup, xa, ya);
               value nop, jt, xarg, xlo, xup; real xarg, xlo, xup;
               integer nop, jt; real array xa, ya;
```

comment This procedure will perform interpolation, differentiation, or integration operating upon functions of one variable which over part or all of the interval of interest are adequately described by a di-parabolic fit.

The routine was originally programmed as an open subroutine for the IBM 704 in FORTRAN II and occupied 323 memory locations. It is based upon a Lagrange interpolation scheme specialized for averaged second order parabolas. The technique finds the slope of a function numerically defined at points 1, 2, 3 and 4 by fitting a parabola through the points 1, 2, 3, and another parabola through the points 2, 3, and 4. The slope then, at point 2, is the average analytical derivative of the two parabolas, i.e. the coefficients of the parabola through points 1, 2 and 3 $(a_1x_2^2+b_1x_2+c_1)$ and the coefficients of the parabola through points 2, 3, and 4 $(a_2x_2^2+b_2x_2+c_2)$ are determined by applying Lagrange's equations as shown below. The arithmetic mean of these coefficients $a = (a_1+a_2)/2$, $b = (b_1+b_2)/2$, $c = (c_1+c_2)/2$ are used to supply the slope in the interval from 2 to 3, namely $(2ax + b)$.

The interpolation is calculated in similar fashion, except the final formula is that a parabola $(ax^2 + bx + c)$.

The integration is performed likewise by a curve fitting process, e.g. the integral between any two points say 2 and 3 is the average integral of the two parabolas between the independent coordinate limits for points 2 and 3. The averaging process is done for each interval along the abscissa as the results obtained are accumulated to evaluate the definite integral.

Applying Lagrange's equations, the coefficients a, b, and c may be found by defining: $T_j = y_j/\prod_{i=1,\, i\neq j}^{n} (X_j - X_i)$ where $y = f(x)$, $n = 3$, $j = 1, 2, \cdots, n$, then $a = \sum_{i=1}^{n} T_i$, $b = \sum_{i=1}^{n} T_i \sum_{j=1,\, j\neq i}^{n} X_j$, $c = \sum_{i=1}^{n} T_i \prod_{j=1,\, j\neq i}^{n} X_j$;

```
begin real    ca, cb, cc, a, b, c, syl, syu, term1, term2, term3, da,
                dif, sum;
integer       jm, js, jul, ia, ib;
   start:     switch alpha := L1, L1, L12;  switch beta := L9,
                L5, L6;
              switch gamma := L10, L11;  switch delta := L8,
                L8, L13;
comment       For interpolation, differentiation or integration set
                jt = 1, 2, or 3 respectively;
              go to alpha [jt];
   L1:        if xarg ≥ xa [nop] then go to L2;
              if xarg ≥ xa [nop−1] then go to L2;
              if xarg ≤ xa [1] then go to L3;
              if xarg ≤ xa [2] then go to L3;  go to L4;
   L2:        jm := nop−1;  js := 1;  go to term;
   L3:        jm := 2;  js := 1;  go to term;
comment       Locate argument;
   L4:        for ia := 2 step 1 until nop do begin
              if xa [ia] > xarg then go to L7;  jm := ia end;
comment       Before loop is complete xarg ≤ xa [ia];
   L5:        ca := a;  cb := b;  cc := c;  js := 3;  jm :=
                jm+1;  go to term;
   L6:        a := (ca+a)/2;  b := (cb+b)/2;  c := (cc+c)/2;
                go to L9;
   L7:        js := 2;  go to term;
   L8:        go to beta [js];
   L9:        go to gamma [jt];
comment       Interpolation, jt = 1;
   L10:       da := a × xarg ↑ 2 + b × xarg + c;  go to exit1;
comment       Differentiation, jt = 2;
   L11:       dif := 2 × xarg + b;  go to exit2;
comment       Integration, jt = 3;
   L12:       sum := 0;  syl := xlo;  jul := nop − 1;
                ib := 2;
   L16:       for jm := ib step 1 until jul do begin;
comment       Lagrange formulae;
              term1 := ya [jm − 1]/((xa [jm − 1] − xa[jm]) ×
                (xa[jm − 1] − xa[jm + 1]));
              term2 := ya [jm]/((xa [jm] − xa [jm − 1]) ×
                (xa[jm] − xa [jm + 1]));
              term3 := ya [jm + 1]/((xa [jm + 1] − xa [jm − 1]) ×
                (xa [jm + 1] − xa [jm]));
              a := term1 + term2 + term3;
              b := −(xa [jm] + xa [jm + 1]) × term1 − (xa
                [jm − 1] + xa [jm + 1]) × term2 − (xa [jm − 1] +
                xa [jm]) × term3;
              c := xa [jm] × xa [jm + 1] × term1 + xa [jm − 1] ×
                xa [jm + 1] × term2 + xa [jm − 1] × xa [jm] ×
                term3; go to delta [jt];
   L13:       if jm ≠ 2 then go to L14;
              ca := a;  cb := b;  cc := c;  go to L15;
   L14:       ca := (a + ca)/2;  cb := (b + cb)/2;  cc :=
                (c + cc)/2;
   L15:       syu := xa [jm];
              sum := sum + ca × (syu ↑ 3 − syl ↑ 3)/3 + cb ×
                (syu ↑ 2 − syl ↑ 2)/2 + cc × (syu − syl);
              ca := a;  cb := b;  cc := c;  syl := syu end;
comment       End of loop on [jm] index;
              sum := sum + ca × (xup ↑ 3-syl ↑ 3)/3 + cb ×
                (xup ↑ 2-syl ↑ 2)/2 + cc × (xup − syl);  go
                to exit3;
   term:      ib := jm;  jul := ib;  go to L16;
comment       The results for interpolation, differentiation, and
                integration are da, dif, and sum respectively;
   exit1:     AVINT := da;  go to exit;
   exit2:     AVINT := dif;  go to exit;
   exit3:     AVINT := sum;
   exit:      end
```

ALGORITHM 78
RATIONAL ROOTS OF POLYNOMIALS WITH INTEGER COEFFICIENTS

C. Perry
University of California at San Diego, La Jolla, California

```
comment This ALGOL procedure, named ratfact, for finding
   rational roots of polynomials with integer coefficients is a
   pedagogical example illustrating the use of the for statement
   described in section 4.6.3. Also, an extension suggested by
   J. Peck of the well-known polynomial evaluation by nesting,
   i.e. Horner's method, is used. The polynomial f(x) = a0 + a1x+
   ··· +anx^n with integer coefficients and with a0an≠o has a
   lowest term rational root p/q if and only if a0q^n + a1q^(n-1)p +
   ··· +a(n-1)q p^(n-1) + anp^n = 0, also q must be a factor of an and
   p a factor of a0 . Procedure RATFACT outputs the nonzero
   rational roots p/q by execution of the procedure whose formal
   name is print. The output procedure uses the string whose formal
   name is format for control of the output format;
procedure ratfact (a, n, print, format);
  integer array a[0:n]; integer n; procedure print; string
    format;
  begin integer i, p, q, r, t, f, g;
  p loop: for p := 1 step 1 until abs (a[0]) do
    begin comment if p is not a factor of a [0] or q is not a factor
      of a[n] then skip to the end of the loop for advance in the
      respective for list;
      if a[0] ≠ (a[0]÷p)×p then go to 1
      else q loop: for q := 1 step 1 until abs (a[n]) do
        begin if a[n] ≠ (a[n]÷ q)×q then go to 2
          else
          begin comment root test and print;
            comment start polynomial evaluation;
              f := g := a[0]; t := p;
              for i := 1 step 1 until n do
              begin r := a[i]×t;
                    f := f×q+r;
                    g := -g×q+r;
                    t := t×p;
              end polynomial evaluation;
              comment computing r saves one subscript
                evaluation;
            if f=0 then print (format, p, q);
            if g=0 then print (format,-p, q);
            comment print is the formal name of the procedure
              to be used to output the variables in the format
              specified by the string whose formal name is format;
          end root test and print;
      2: end q loop;
1: end p loop;
  end ratfact, without overflow test.
```

ALGORITHM 79
DIFFERENCE EXPRESSION COEFFICIENTS

Thomas P. Giammo
Space Technology Laboratories, Inc., Los Angeles, California

```
procedure dicol (k, n, xp, xtab, coef);
value k, n; integer k, n; real xp;
array xtab, coef;
comment dicol produces the coefficients for the n ordinates
   (corresponding to the abscissae, xtab) in the n-point finite
   difference expression for the k-th derivative evaluated at xp.
   The method used is to determine the analytic expression for
   the k-th derivative of each coefficient in the n-point Lagrangian
   interpolation formula and evaluate it at xp. Note that k=0
   will produce the Lagrangian interpolation coefficients them-
   selves;
begin integer array xuse [1 : n-1]; real factk, sum, denom,
   part;
integer i, terms, j, m, high;
factk := 1.0; for i := 2 step 1 until k do factk := i×factk;
terms := n-k-1; if terms<0 then go to Z;
for j := 1 step 1 until n do
loop: begin sum := 0; denom := 1.0; part := 1.0;
        for i := 1 step 1 until n do
        if i ≠ j then denom := denom× (xtab [j] - xtab [i]);
        if terms = 0 then go to Y;
        m := 1; high := 1;
    A:  if (high = j)∨(xtab [high] = xp) then
        A1: begin high := high + 1; go to A end A1;
        if high > n then A2: begin m := m-1; if m>0
          then
        A3: begin high := xuse [m]+1; go to A end A3;
        go to X end A2;
        xuse [m] := high; m := m+1;
        if m≦terms then begin high := high + 1; go to
          A end;
        for i := 1 step 1 until terms do
          part := part× (xp - xtab [xuse [i]]);
        sum := sum + part; m := terms; part := 1.0;
        high := xuse [terms] + 1; go to A;
    Y:  sum := 1.0;
    X:  coef [j] := sum × factk/denom end loop;
      go to EXIT;
    Z:  for i := 1 step 1 until n do coef [i] := 0;
    EXIT: end dicol
```