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, differen-
  tiation, or integration operating upon functions of one vari-
  able which over part or all of the interval of interest are ade-
  quately 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;  im :=
              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
```

CERTIFICATION OF ALGORITHM 77
AVINT (Paul E. Hennion, *Comm. ACM 5*, Feb., 1962)

VICTOR E. WHITTIER
Computations Res. Lab., The Dow Chemical Co., Midland, Mich.

AVINT was transliterated into BAC-220 (a dialect of ALGOL-58) and was tested on the Burroughs 220 computer. The following minor errors were found:

1. The first statement following label L11 should read:
   dif := 2 × a × xarg + b;
2. The semicolon (;) at the end of the line beginning with the label L16 should be deleted.
3. There appears to be a confusion between "1" (numeric) and "l" (alphabetic) following label L12. This portion of the program should read:
   L12: sum := 0; syl := xlo; jul := nop − 1; ib := 2;

After making the above corrections the procedure was tested for interpolation, differentiation, and integration using e^x, log X, and sin X in the range ($1.0 \leqq X \leqq 5.0$). Twenty-one values of each of these functions, evenly spaced with respect to X and accurate to at least 7 significant digits, were tabulated in the above range. Then the procedure was tested. The following table indicates approximately the accuracy obtained:

	Number of Significant Digits		
Function	*Interpolation*	*Differentiation*	*Integration*
e^x	≧4*	≧2	≧4
log X	≧4*	≧2	≧3
sin X	≧4*	≧2	≧4

* Except for interpolation between the first two points in the table.

The above results are quite reasonable in view of the relatively large increment in X. Tests using smaller increments in X and uneven spacing of X were also satisfactory.

It was also discovered that for integration the following restrictions must be observed:

1. xlo ≦ xa (1).
2. xup ≧ xa (nop).

REMARK ON ALGORITHM 77
INTERPOLATION, DIFFERENTIATION, AND INTEGRATION [P. E. Hennion, *Comm. ACM*, Feb., 1962]

P. E. HENNION
Giannini Controls Corp., Berwyn, Penn.

It was brought to my attention through the CERTIFICATION OF ALGORITHM 77 AVINT [V. E. Whittier, *Comm. ACM*, June, 1962] that restrictions on the upper and lower limits of integration existed, i.e., (1) $x10 \leqq xa$ (1), (2) $xup \geqq xa(nop)$. To remove these restrictions the following two changes should be made.

1. Before line *L*16: and after the statement *ib* := 2; place the following code:

```
      for ia := 1 step 1 until nop do begin
      if xa(ia) ≧ x10 then go to L17;  ib := ib + 1;  end;
L17:  ju 1 := nop + 1;  for ia := 1 step 1 until nop do begin
      ju 1 := ju 1 − 1;  if xa(ju 1) > xup end; ju 1 := ju 1 − 1;
```

2. Change line *L*13: to read:

```
L13:  if jm ≠ ib then go to L14;
```

REMARK ON ALGORITHM 77
INTERPOLATION, DIFFERENTIATION, AND INTEGRATION [P. E. Hennion, *Comm. ACM 5*, Feb. 1962]

P. E. HENNION
Giannini Controls Corp., Berwyn, Penn.

It was brought to my attention through the CERTIFICATION OF ALGORITHM 77 AVINT (V. E. Whittier, *Comm. ACM*, June, 1962) that restrictions on the upper and lower limits of integration existed, i.e., (1) $xlo \leqq xa(1)$, (2) $xup \geqq xa(nop)$. To remove these restrictions the following two changes should be made.

1. Replace the two lines starting at line *L*12: and ending after the statement *ib* := 2; with the following code:

```
L12:  sum := 0;  syl := xlo; ib := 2; jul := nop;
      for ia := 1 step 1 until nop do begin
      if xa [ia] ≧ xlo then go to L17;  ib := ib + 1; end;
L17:  for ia := 1 step 1 until nop do begin
      if xup ≧ xa [jul] then go to L18; jul := jul − 1;  end;
L18:  jul := jul − 1;
```

2. Change line *L*13: to read

```
L13:  if jm ≠ ib then go to L14;
```