1969

Smoothing and Estimation Derivatives of Equispaced Data

J. J. Casaletto

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:

69-035

SMOOTHING AND ESTIMATING DERIVATIVES OF EQUISPACED DATA

by

J. J. Casaletto and J. R. Rice

May, 1969

CSD TR 35

ABSTRACT

This paper discusses various aspects of the smoothing and estimation of derivatives of equispaced data. The background for least squares polynomial smoothing is summarized. The various alternatives for programs in a computing center's library are discussed and a particular alternative is selected as most suitable. An algorithm named SMOOTH is given (in Fortran) which implements this alternative. SMOOTH estimates the smoothed value of the data or its first or second derivative based on specified polynomial degree and number of points to enter the smoothing. The paper concludes with a discussion of methods suitable to compute large arrays of smoothing weights. There are 3 appendices which contain, respectively, explicit formulas associated with Gram polynomials, explicit formulas for the smoothing weights and tables of initial segments of arrays for computing large tables of smoothing weights.

SMOOTHING AND ESTIMATING DERIVATIVES OF EQUISPACED DATA

1. MATHEMATICAL BACKGROUND

An algorithm for estimating the values and derivatives of equispaced, tabulated functions is presented. Let $\{(x_j, y_j)\}$ $j = -n, -n+1, \ldots, n$ be the data with $h = x_{j+1} - x_j$. The algorithm is based on least squares polynomial approximation and computes smoothing weight coefficients to apply to the data.

We consider the GRAM (or Tchebycheff) polynomials $\{P_{m,n}(x)\}$ of degree m. These polynomials are orthogonal

$$\sum_{j=-n}^{n} P_{m,n}(j)\, P_{k,n}(j) = 0 \qquad m \neq k$$

and are given explicitly by

$$P_{m,n}(j) \sum_{k=0}^{m} \frac{(-1)^{k+m}(m+k)!\,(n+j)!\,(2n-k)!}{(k!)^2(m-k)!\,(n+j-k)!\,(2n)!}$$

Let $Q_{m,n}(j)$ be the best least squares approximation of y_j by a polynomial of degree m on the 2n+1 points $i = 0, \pm 1, \ldots, \pm n$. We have

$$Q_{m,n}(j) = \sum_{k=o}^{m} a_k P_{k,n}(j)$$

where $a_k = [\sum_{j=-n}^{n} y_j P_{k,n}(j)]/S_{k,n}$ and $S_{k,n} = \sum_{j=-n}^{n} [P_{k,n}(j)]^2$.

Therefore, we have the smoothed estimate of the data

$$y_j \sim Q_{m,n}(j) = \sum_{k=o}^{m} \left[\frac{\sum_{i=-n}^{n} y_i P_{k,n}(i)}{S_{k,n}} \right] P_{k,n}(j)$$

and, in particular, we have the smoothed value for y_o, given by

$$y_o \sim Q_{m,n}(0) = \sum_{k=o}^{m} \left[\frac{\sum_{i=-n}^{n} y_i P_{k,n}(i)}{S_{k,n}} \right] P_{k,n}(0)$$

$$= \sum_{i=-n}^{n} \left[\sum_{k=o}^{m} \frac{P_{k,n}(i)\ P_{k,n}(0)}{S_{k,n}} \right] y_i \quad .$$

The smoothing weights are denoted by

$$A^m_{n,i} = \sum_{k=o}^{m} \frac{P_{k,n}(i)\ P_{k,n}(0)}{S_{k,n}} \quad .$$

Smoothed estimates for the first and second derivatives are found by differentiating $Q_{m,n}(j)$ and evaluating it for $j = 0$. Therefore,

$$\frac{dy_o}{dx} = y'_o \sim \frac{1}{h}\, Q'_{m,n}(0) = \frac{1}{h} \sum_{i=-n}^{n} \sum_{k=o}^{m} \frac{P_{k,n}(i)\ P'_{k,n}(0)}{S_{k,n}}\ y_i \quad \text{and}$$

$$\frac{d^2y_o}{dx^2} = y''_o \sim \frac{1}{h^2}\, Q''_{m,n}(0) = \frac{1}{h^2} \sum_{i=-n}^{n} \sum_{k=o}^{m} \frac{P_{k,n}(i)P''_{k,n}(0)}{S_{k,n}}\ y_i$$

The weights here are denotes, respectively, by $B^m_{n,i}$ amd $C^m_{n,i}$. See Appendix II for $A^m_{n,i}$, $B^m_{n,i}$, and $C^m_{n,i}$ for $m = 1, 3, 5$.

2. ALTERNATIVES FOR APPLICATIONS AND COMPUTING WEIGHTS

A computing center's library should contain a routine which will, when given the arrays X and Y, compute the smoothed values for y_o, y'_o, and y''_o. The major consideration for such a routine is to obtain the weights $A^m_{n,j}$, $B^m_{n,j}$, and $C^m_{n,j}$.

One possible method is to calculate, for any m and n, $P_{m,n}(j)$ for any j and to evaluate its derivatives at zero from explicit formulae. This method has the advantages that any requested weights can be calculated and the routine is relatively short. However, the computation time would be rather large.

At the other end of the spectrum, one can limit m,n and the maximum order of the derivative; and calculate and store the necessary weights on some auxiliary storage device. The advantage here is, of course, speed. However, it is disadvantageous to restrict m,n and the order of the derivative, although these might not be, in practice, great restrictions. The major disadvantage is the number of constants which must be stored. If the maximum m is 5, maximum n is 50 (i.e. 101 points), and only y_o, y'_o and y''_o are allowed, then, taking into account the facts that $A^{2k+1}_{n,j} = A^{2k}_{n,j}$, $B^{2k+2}_{n,j} = B^{2k+1}_{n,j}$, $C^{2k+3}_{n,j} = C^{2k+2}_{n,j}$ for k = 0, 1, ..., and the symmetry and anti-symmetry of these constants about j = 0, it requires some 10,579 constants. This can be reduced further by observing that $A^{1}_{n,j}$ is independent of j and that $B^{1}_{n,o} = B^{3}_{n,o} = B^{5}_{n,o} = 0$. This gives 9,154 constants and involves some additional logic. This is a large block of storage for such a routine, but not overwhelmingly large if one wants the highest possible speed.

These methods represent two extremes. The first has a large capability and small storage requirements, but is slow. The second has a limited, though practical, capability and large storage requirements, but is fast. We present an algorithm which makes a compromise on the time-storage-capability relationship; that is, using some explicit formulae, setting a practical restriction on m and computing smoothed values only for y_o,y'_o and y''_o.

Note in Appendix II, that the numerators of the weights are polynomials in n and j, while the denominators are polynomials in n; therefore, the denominators are calculated separately. The numerators are denoted by $W_{n,j}$. The procedure is to generate integers $W_{n,j}$, form the sum $\sum W_{n,j}\, y_j$, and divide by the integral denominator and the appropriate power of h. Note, further, that n is fixed for each entry into the routine and thus the $W_{n,j}$ are polynomials in j, $j = 0, \pm 1, \pm 2, \ldots, \pm n$. This suggests the method of differences to evaluate $W_{n,j}$. This method is more efficient for evaluating polynomials at a large number of equispaced points; whereas nested evaluation is more efficient for a smaller number of evaluations.

To make this quantitative, let $P_n(j)$ be a polynomial of degree n, to be evaluated at k equispaced points. Let the unit of work be an addition, and assume one multiplication is equivalent to μ additions (μ is thus a machine characteristic). Then nested evaluation requires $nk(\mu+1)$ additions. Evaluation by differences requires (n+1) starting values, the construction of a difference table, and final evaluations. If the (n+1) starting values are obtained from nested evaluation, then for $k > n+1$, $n(n+1)(\mu+1/2)+nk$ additions are necessary. The difference table method is more efficient when $(k-n-1)$ is greater than $(n+1)/2\mu$.

The difference table method can be improved be representing $P_n(j)$ in a point-value form; i.e. by the vector $(P_n(0), P_n(1), \ldots, P_n(n))$. Using this approach, $nk-n(n+1)/2$ additions are necessary. Hence, this method is always more efficient than the nested form when the values of $P_n(j)$ are required, at least, at each $j = 0, 1, \ldots, n$.

One may use another method of representing the polynomial $P_n(j)$, the "forward difference diagonal" form with step h; i.e. by the vector

$$(\Delta_h^n P_n(0), \Delta_h^{n-1} P_n(0), \ldots, \Delta_h P_n(0), P_n(0)),$$

where $\Delta_h^i P_n(x)$ is the i-th forward difference of $P_n(x)$ with step-size h. This representation eliminates the construction of the difference table, $P_n(0)$ is given and successive values of $P_n(j)$ are obtainec by a "rippled" (using intermediate sums as summands) addition process.

Similarly, one may use the backward difference diagonal representation with step h; i.e.

$$(\nabla_h^n P_n(0), \nabla_h^{n-1} P_n(0), \ldots, \nabla_h P_n(0), P_n(0))$$

where $\nabla_h^i P_n(x)$ is the i-th backward difference of $P_n(x)$ with step-size h. With this form each successive $P_n(j)$ is obtained with n additions. In the calculation of the next value, the diagonal is updated so that it is the backward difference diagonal for the next x. Thus, the vector defining the polynomial is always changing except for the first element which, of course, is the constant difference. The advantages of this form are storage, only a one-dimensional array of n+1 elements is needed, and ease of coding.

Consider the example $P_4(x) = x^4+2x^3+5x^2+6x+1$, the backward difference diagonal at 0 with step 1 is $B = (24, -24, 12, 2, 1)$. Now $P_4(0) = B(5) = 1$, then the simple string (a simple DO loop in Fortran),

$$B(1)+B(2) \rightarrow B(2)+B(3) \rightarrow B(3)+B(4) \rightarrow B(4)+B(5) \rightarrow B(5),$$

updates the diagonal and the vector B is now the backward difference diagonal of $P_4(x)$ at 1. B is now (24, 0, 12, 14, 15), thus $P_4(1)=B(5) = 15$.

3. REMARKS ON THE ALGORITHM "SMOOTH"

The input parameter list, Y, X, INIT, NDERV, NDEG, NPTS, LENGTH, for the function subprogram SMOOTH is described in the initial comment cards. We use the usual convention that the zero-th derivative is the function value. The limits and checks on the arguments are also described in the comment cards.

Now, if NDERV = 0 and NDEG = 0 or 1, then $A^{0}_{n,j} = A^{1}_{n,j} = 1/(2n+1)$; hence, $W_{n,j} = 1$. If NDERV = 1 and NDEG = 1 or 2, then $B^{1}_{n,j} = B^{2}_{n,j} =$ $= 3j/(n(n+1)(2n+1))$; hence $W_{n,j} = 3j$. And if NDERV = 1 with NDEG = 0, or, if NDERV = 2 with NDEG = 0 or 1, then SMOOTH is set to zero.

In the remaining cases, the $W_{n,j}$ are polynomials in j, and the coefficients are polynomials in n (we call these the n-polys of j). Because of symmetry the $W_{n,j}$ are only generated for j = 0, 1, ..., n and the appropriate sign attached for negative j. The parameters NDERV and NDEG point to an implicit triangular array of weights, whose columns are indexed by n (corresponding to NPTS) and whose rows are indexed by j (corresponding to the (INIT+j)-th ordinate). Therefore, NPTS specifies the particular column of this array; i.e. $W_{NPTS,j}$ for j = 0, 1,...,NPTS. Smooth places no restriction on NPTS.

Let m be the greatest integer in (k+1)/2 where k is the degree of $W_{n,j}$ as a polynomial in j. The procedure in SMOOTH is to evaluate $W_{n,j}$ for j = 0,1,...,m, storing them as $B(k-m+j+1) = W_{n,j}$, then reflecting the appropriate values into B(1),..., B(k-m). Thus the B-vector is now a point-value form of $W_{n,j}$. After using these m weights, the B-vector is manipulated so as to become the backward difference diagonal at j = m with step 1. Now SMOOTH continually updates the B-vector, using the $W_{n,j}$ = B(k+1) in the sum $\sum_j W_{n,j}$(Y(INIT+j)+SIGN*Y(INIT-j)), where SIGN = ± 1 as appropriate.

In order to evaluate the initial $W_{n,j}$ we use nested evaluation. The n-polys are evaluated separately and combined with the powers of j. Note that all the n-polys are also polynomials in n(n+1).

We note that for large n, the $W_{n,j}$ and the denominators are very large integers. Thus the use of integer arithmetic is limited by machine word length. SMOOTH, as presented, uses floating point, single precision arithmetic. If exact values for these coefficients are desired, we can scale down the $W_{n,j}$'s and their denominators by canceling a common factor, and/or using double precision arithmetic. The following common factors of the numerators and denominators exist:

$A^3_{n,j}$: 3	$A^5_{n,j}$: $2^2 \cdot 3 \cdot 5 = 60$
$B^3_{n,j}$: $2 \cdot 3 \cdot 5 = 30$	$B^5_{n,j}$: $2^4 \cdot 3^3 \cdot 5 \cdot 7 = 15120$
$C^3_{n,j}$: $2 \cdot 3 \cdot 5 = 30$	$C^5_{n,j}$: $2^2 \cdot 3^3 \cdot 5 \cdot 7 = 3780$

4. THE COMPUTATION OF TABLES OF SMOOTING WEIGHTS

SMOOTH is designed to calculate and use one specific set of weights. To obtain a routine to produce tables of these weights the starting values, $W_{n,o}, \ldots, W_{n,m}$ (m is the greatest integer in (k+1)/2 where k is the degree of $W_{n,j}$ as a polynomial in j), for each column are most efficiently generated by difference methods. For each pair (NDERV, NDEG), NDEG=3,5, one would use the point-value forms ($W_{N,j}$, $W_{N+1,\ j}$, ..., $W_{M,j}$) for j = 0, 1, ..., m, where:

(1) N = $\frac{\text{NDEG}+1}{2}$, a minimum NPTS for NDEG,

(2) $M = L+N$ where L is the degree of $W_{n,j}$ as a polynomial in n, and

(3) n as above

and the point-value form for the denominator, in lieu of using any explicit formulae. This gives the initial segments of the first (L+1) columns. Each column can be completed as in SMOOTH. To obtain the initial segments for additional columns, the m+1 vectors, above, should be manipulated so that they are backward difference diagonals at n=M, and then updated for each column.

Two other methods for table generation are obtained by replacing the point-value forms, for the $W_{n,j}$ noted above, by either the forward or the backward difference diagonals at j = o. The use of these diagonals facilitates program coding. Appendix III contains all the point-value, forward difference diagonal, and backward difference diagonal vectors for the cases allowed in SMOOTH. Note that the common factors have been canceled.

Finally, we note that SMOOTH may be used to compute tables of smoothing weights with its range of allowable arguments. One inserts write statements at the appropriate points (indicated by comment cards) and runs SMOOTH through the range of desired values of NDERV, NDEG and NPTS. This approach is much less efficient in computation time (and restricts the range of NDERV and NDEG), but it requires a trivial modification of SMOOTH.

```
      FUNCTION SMOOTH(Y,X,INIT,NDERV,NDEG,NPTS,LENGTH)
C   SMOOTH OPERATES ON AN EQUISPACED DATA FUNCTION (X,Y) TO PRODUCE THE
C   SMOOTHED VALUE OF THE FUNCTION, OR ITS 1ST OR 2ND DERIVATIVES, AT A
C   SPECIFIED POINT.  THE USER MUST SPECIFY THE DEGREE AND THE NUMBER OF
C   POINTS, AS DESCRIBED IN THE PARAMETER DESCRIPTION BELOW, TO BE USED
C   IN SMOOTHING.  SMOOTHING IS DONE BY APPLYING WEIGHTS TO THE NEIGHBORING
C   ORDINATES.  THE WEIGHTS ARE DETERMINED BY THE LEAST SQUARES
C   APPROXIMATION USING GRAM POLYNOMIALS.

C   PARAMETERS
C     Y     *** THE ARRAY OF ORDINATES.
C     X     *** THE ARRAY OF ABSCISSAS.  USED ONLY TO COMPUTE THE STEP-SIZE H.
C               TO PASS H DIRECTLY, SET LENGTH=-LENGTH AND PUT H IN 2ND ARGUMENT.
C     INIT*** THE SMOOTHED VALUE IS DESIRED AT X(INIT).
C     NDERV** THE ORDER OF THE DERIVATIVE WITH THE USUAL CONVENTION
C               CONCERNING 0.  THE LIMITS ARE NDERV= 0, 1, 2.
C     NDEG*** THE DEGREE OF THE APPROXIMATING POLYNOMIAL.
C               THE LIMITS ARE 0 TO 5
C     NPTS*** THE NUMBER OF POINTS TO BE USED IS 2*NPTS+1
C                    I.E. NPTS POINTS ON BOTH SIDES OF X(INIT)
C     LENGTH* THE NUMBER OF ELEMENTS IN THE ARRAY X.
C               USED ONLY TO CHECK IF REQUEST IS LEGITIMATE
C               LENGTH=0 ELIMINATES THIS CHECK

      DIMENSION X(1),Y(1),B(6),MESSGE(5)
      REAL N2

C    STEP-SIZE CALCULATION
      H=-1
      IF(LENGTH.LE.0) H=X(1)
      IF(LENGTH.EQ.0) LENGTH=INIT+NPTS
      LENGTH=IABS(LENGTH)
      IF(H.LT.0.) H=X(INIT+1)-X(INIT)

C    PARAMETER CHECKS
      DO 1 I=1,5
    1 MESSGE(I)=0
      IF(NDERV.LT.0.OR.NDERV.GT.2) MESSGE(1)=1
      IF(NDEG.LT.0.OR.NDEG.GT.5) MESSGE(2)=1
      IF(NDEG.GT.NPTS+NPTS) MESSGE(3)=1
      IF(INIT.LE.0) MESSGE(4)=1
      IF(INIT.LE.NPTS.OR.INIT+NPTS.GT.LENGTH) MESSGE(5)=1
      IF(MESSGE(1)+MESSGE(2)+MESSGE(3)+MESSGE(4)+MESSGE(5).EQ.0) GO TO 6
      WRITE(6,2)
    2 FORMAT(//46H PARAMETER ERROR IN CALL TO SMOOTHING ROUTINE   )
      IF(MESSGE(1).EQ.1) WRITE(6,3) NDERV
    3 FORMAT(45H ROUTINE NOT EQUIPPED FOR DERIVATIVE OF ORDERI2)
      IF(MESSGE(2).EQ.1) WRITE(6,4) NDEG
    4 FORMAT(45H ROUTINE NOT EQUIPPED FOR SMOOTHING BY DEGREEI2)
      IF(MESSGE(3).EQ.1) WRITE(6,5) NDEG,NPTS
    5 FORMAT(30H NO UNIQUE SOLUTION FOR DEGREEI2,13H BASED ON 2*(I3,10H
    51)+1 POINTS)
```

```
      IF(MESSGE(4).EQ.1) WRITE(6,6)
    6 FORMAT(49H DERIVATIVE REQUESTED AT NEGATIVELY INDEXED POINT)
      IF(MESSGE(5).EQ.1) WRITE(6,7) LENGTH
    7 FORMAT(15H ARRAY SIZE OF I3,50H INDICATES NOT ENOUGH POINTS TO ACC
     71OMODATE REQUEST/107H NOTE THAT PARAMETER 6 INDICATES THAT TWICE TH
     72AT NUMBER PLUS ONE POINTS ARE TO BE USED IN THE APPROXIMATION )
      RETURN

    8 SMOOTH=0
      IF(NDEG-NDERV.LT.0) RETURN
      L=1
      KEY=2
      SIGN=1
      N2=(NPTS+1)*NPTS
      TDN=NPTS+NPTS+1
      T2=TDN*TDN-16.
      IF(NDERV-1) 9,15,13

C   UNDERIVED SMOOTHING

    9 IF(NDEG.GT.1) GO TO 11
C   DEGREE 0 OR 1
      SMOOTH=Y(INIT)
      DO 10 K=1,NPTS
   10 SMOOTH=SMOOTH+Y(INIT+K)+Y(INIT-K)
      SMOOTH=SMOOTH/TDN
      RETURN

   11 DENOM=(T2+12.)*TDN
      IF(NDEG.GT.3) GO TO 12
C   DEGREE 2 OR 3
      B(2)=9.*T2-3.
      B(3)=B(2)-15.
      GO TO 22
C   DEGREE 4 OR 5
   12 B(3)=(225.*N2-750.)*N2+180.
      B(4)=B(3)-1050.*N2+2520.
      B(5)=B(3)-4200.*N2+21420.
      DENOM=4.*T2*DENOM
      GO TO 20


C   2ND DERIVATIVE SMOOTHING

   13 DENOM=(T2+12.)*N2*TDN*H*H
      IF(NDEG.GT.3) GO TO 14
C   DEGREE 2 OR 3
      B(2)=-30.*N2
      B(3)=B(2)+90.
      GO TO 22
C   DEGREE 4 OR 5
   14 DENOM=2.*T2*(N2-2.)*DENOM
      B(3)=((-1050.*N2+3675.)*N2-3150.)*N2
```

```
      B(4)=B(3)+(8820.*N2-27090.)*N2 16900.
      B(5)=B(3)+(35280.*N2-221760.)*N2+170100.
      GO TO 20

C   1ST DERIVATIVE SMOOTHING

   15 DENOM=N2*TDN*H
      SIGN=-1
      IF(NDEG.GT.2) GO TO 17
C   DEGREE 1 OR 2
      DO 16 K=1,NPTS
   16 SMOOTH=SMOOTH+FLOAT(K)*(Y(INIT+K)-Y(INIT-K))
      SMOOTH=3.*SMOOTH/DENOM
      RETURN

   17 DENOM=DENOM*(T2+12.)*(N2-2.)
      IF(NDEG.GT.4) GO TO 18
C   DEGREE 3 OR 4
      B(2)=0
      B(3)=(75.*N2-180.)*N2+60.
      B(4)=(150.*N2-990.)*N2+330.
      GO TO 21
C   DEGREE 5
   18 DENOM=4.*T2*(N2-6.)*DENOM
      B(3)=0
C   B(1),B(2), AND B(6) ARE USED HERE AS TEMP STORAGE
      B(1)=(4.*N2-39.)*N2+90.
      B(2)=(15.*N2-50.)*N2+12.
      Z=-B(1)*(420.*N2-140.)
      B(1)=B(1)*((300.*N2-300.)*N2+100.)+11*B(2)*B(2)
      B(6)=-770.*N2+1155.
      DO 19 J=4,6
C H AS STEP-SIZE ALREADY INCLUDED IN DENOM.  USED HERE AS TEMP STORAGE.
      H=J-3
   19 B(J)=(((693.*H*H+B(6))*B(2)+Z)*H*H+B(1))*H

C   SET B-VECTOR TO P-V FORM AND GENERATE PARTIAL SUM
      L=2
   20 KEY=3
   21 L=L+1
   22 B(KEY-1)=SIGN*B(KEY+1)
      IF(KEY.EQ.3) B(1)=SIGN*B(5)
      SMOOTH=B(KEY)*Y(INIT)
   23 DO 24 K=1,L
C   TO OBTAIN THE FIRST PART OF W(N,J) ARRAY, PRINT B(KEY+K) HERE.
   24 SMOOTH=SMOOTH+B(KEY+K)*(Y(INIT+K)+SIGN*Y(INIT-K))
      IF(L.EQ.NPTS) GO TO 28

C   TRANSFORM B-VECTOR INTO BACKWARD DIFFERENCE DIAGONAL
      M=KEY+L
      M1=M-1
```

```
      DO 25 I=1,M1
      K=M-I
      DO 25 J=1,K
   25 B(J)=B(J+1)-B(J)

C  GENERATE AND SPILL NEW WEIGHTS INTO SUM
      M1=L+1
      DO 27 K=M1,NPTS
      DO 26 J=2,M
   26 B(J)=B(J)+B(J-1)
C  TO OBTAIN THE REST OF THE W(K,J) ARRAY, PRINT B(M) HERE.
   27 SMOOTH=SMOOTH+B(M)*(Y(INIT+K)+SIGN*Y(INIT-K))
C  DENOM INCLUDES STEP-SIZE ADJUSTMENT
   28 SMOOTH=SMOOTH/DENOM
      RETURN
      END
```

APPENDIX I: GRAM POLYNOMIALS, DERIVATIVES AND SPECIAL VALUES

A. THE GRAM POLYNOMIALS

$$P_{o,n}(j) = 1$$

$$P_{1,n}(j) = \frac{j}{n}$$

$$P_{2,n}(j) = \frac{3j^2-n(n+1)}{n(2n-1)}$$

$$P_{3,n}(j) = \frac{5j^3-(3n^2+3n-1)j}{n(n-1)(2n-1)}$$

$$P_{4,n}(j) = \frac{35j^4-5(6n^2+6n-5)j^2 + 3n(n^2-1)(n+2)}{2n(n-1)(2n-1)(2n-3)}$$

$$P_{5,n}(j) = \frac{63j^5-35(2n^2+2n-3)j^3+(15n^4+30n^3-35n^2-50n+12)j}{2n(n-1)(n-2)(2n-1)(2n-3)}$$

B. THE DERIVED POLYNOMIALS

$$P'_{o,n}(j) = 0$$

$$P'_{1,n}(j) = \frac{1}{n}$$

$$P'_{2,n}(j) = \frac{6j}{n(2n-1)}$$

$$P'_{3,n}(j) = \frac{15j^2-(3n^2+3n-1)}{n(2n-1)(n-1)}$$

$$P'_{4,n}(j) = \frac{10j(14j^2-(6n^2+6n-5))}{2n(n-1)(2n-1)(2n-3)}$$

$$P'_{5,n}(j) = \frac{315j^4-105(2n^2+2n-3)j^2+(15n^4+30n^3-35n^2-50n+12)}{2n(n-1)(n-2)(2n-1)(2n-3)}$$

C. THE SECOND DERIVATIVES

$$P''_{o,n}(j) = 0$$

$$P''_{1,n}(j) = 0$$

$$P''_{2,n}(j) = \frac{6}{n(2n-1)}$$

$$P''_{3,n}(j) = \frac{30j}{n(n-1)(2n-1)}$$

$$P''_{4,n}(j) = \frac{420j^2-10(6n^2+6n-5)}{2n(n-1)(2n-1)(2n-3)}$$

$$P''_{5,n}(j) = \frac{1260j^3-210(2n^2+2n-3)j}{2n(n-1)(n-2)(2n-1)(2n-3)}$$

D. SPECIAL VALUES

$P_{k,n}(0) = 0$, for k odd

$P_{o,n}(0) = 1$

$$P_{2,n}(0) = -\frac{n+1}{2n-1}$$

$$P_{4,n}(0) = \frac{3(n+1)(n+2)}{2(2n-1)(2n-3)}$$

$P'_{k,n}(0) = 0$, for k even

$$P'_{1,n}(0) = \frac{1}{n}$$

$$P'_{3,n}(0) = -\frac{3n^2+3n-1}{n(n-1)(2n-1)}$$

$$P'_{5,n}(0) = \frac{15n^4+30n^3-35n^2-50n+12}{2n(n-1)(n-2)(2n-1)(2n-3)}$$

$P''_{k,n}(0) = 0$, for k=0 and k odd

$$P''_{2,n}(0) = \frac{6}{n(2n-1)}$$

$$P''_{4,n}(0) = \frac{5(6n^2+6n-5)}{n(n-1)(2n-1)(2n-3)}$$

E. THE SUM OF THE SQUARES

$$S_{o,n} = 2n+1$$

$$S_{1,n} = \frac{(n+1)(2n+1)}{3n}$$

$$S_{2,n} = \frac{(n+1)(2n+1)(2n+3)}{5n(2n-1)}$$

$$S_{3,n} = \frac{(n+1)(n+2)(2n+1)(2n+3)}{7n(n-1)(2n-1)}$$

$$S_{4,n} = \frac{(n+1)(n+2)(2n+1)(2n+3)(2n+5)}{9n(n-1)(2n-1)(2n-3)}$$

$$S_{5,n} = \frac{(n+1)(n+2)(n+3)(2n+1)(2n+3)(2n+5)}{11n(n-1)(n-2)(2n-1)(2n-3)}$$

APPENDIX II: FORMULAE FOR THE WEIGHTS

NOTATION:

$$A^m_{n,j} = \sum_{k=o}^{m} \frac{P_{k,n}(j)P_{k,n}(0)}{S_{k,n}}$$

denotes the j-th weight for m-th degree smoothing based on 2n+1 points

$$B^m_{n,j} = \sum_{k=o}^{m} \frac{P_{k,n}(j)P'_{k,n}(0)}{S_{k,n}}$$

denotes the j-th weight for the 1st derivative by m-th degree smoothing based on 2n+1 points.

$$C^m_{n,j} = \sum_{k=o}^{m} \frac{P_{k,n}(j)P''_{k,n}(0)}{S_{k,n}}$$

denotes the j-th weight for the 2nd derivative by m-th degree smoothing based on 2n+1 points.

$$A^1_{n,j} = \frac{1}{2n+1}$$

$$A^3_{n,j} = \frac{3[(3n^2+3n-1)-5j^2]}{(2n-1)(2n+1)(2n+3)}$$

$$A^5_{n,j} = \frac{15[63j^4-35(2n^2+2n-3)j^2 + (15n^4+30n^3-35n^2-50n+12)]}{4(2n-3)(2n-1)(2n+1)(2n+3)(2n+5)}$$

$$B^1_{n,j} = \frac{3j}{n(n+1)(2n+1)}$$

$$B^3_{n,j} = \frac{25[3n^4+6n^3-3n+1]j-35[3n^2+3n-1]j^3}{n(n-1)(n+1)(n+2)(2n-1)(2n+1)(2n+3)}$$

$$B^5_{n,j} = \frac{\begin{gathered}\{693[15n^4+30n^3-35n^2-50n+12]j^5-35[4(3n^2+3n-1)(2n^2+11n+15)(2n^2-7n+6) \\ +11(15n^4+30n^3-35n^2-50n+12)(2n^2+2n-3)]j^3 \\ +[12(2n^2+11n+15)(2n^2+7n+6)(2n^2-7n+6)(2n^2-3n+1)+28(3n^2+3n-1)^2(2n^2-7n+6)(2n^2+11n+15) \\ +11(15n^4+30n^3-35n^2-50n+12)^2]j\}\end{gathered}}{4n(n-2)(n-1)(n+1)(n+2)(n+3)(2n-3)(2n-1)(2n+1)(2n+3)(2n+5)}$$

$$C^1_{n,j} = 0$$

$$C^3_{n,j} = \frac{30[3j^2 - n(n+1)]}{n(n+1)(2n-1)(2n+1)(2n+3)}$$

$$C^5_{n,j} = \frac{-15[105(6n^2+6n-5)j^4 - 3(196n^4+392n^3-196n^2-392n+245)j^2 + (70n^6+210n^5-35n^4 -420n^3-35n^2+210n)]}{2n(n-1)(n+1)(n+2)(2n-3)(2n-1)(2n+1)(2n+3)(2n+5)}$$

NESTED FORM FOR NUMERATORS

$A^1_{n,j}$: 1

$A^3_{n,j}$: $(9\cdot n+9)\cdot n - 3 - 15\cdot j\cdot j$

$A^5_{n,j}$: $[945\cdot j\cdot j-((1050n+1050)n-1575)]\cdot j\cdot j+(((15n+30)n-35)n-50)n+12$

$B^1_{n,j}$: $3\cdot j$

$B^3_{n,j}$: $[((-105n-105)n+35)\cdot j\cdot j+((75n+150)\cdot n\cdot n-75)n+25]\cdot j$

$B^5_{n,j}$: $\{[[(((10395n+20790)n-24255)n-34650)n+8316]\cdot j\cdot j+(((((-13230n-39690)n+33075)n+132300)n-37485)n-110250)n+26460]\cdot j\cdot j$

$+((((((3675n+14700)n-7350)n-73500)n-13965)n+111720)n+26460)n-44100)n+10584\ \}\cdot j$

$C^1_{n,j}$: 0

$C^3_{n,j}$: $90\cdot j\cdot j-(30n+30)n$

$C^5_{n,j}$: $[((-9450n-9450)n+7875)\cdot j\cdot j+(((8820n+17640)n-8820)n-17640)n+11025]\cdot j\cdot j+$

$(((((-1050n-3150)n+525)n-6300)n+525)n-3150)n$

SPECIAL FORMS FOR NUMERATORS, $W_{n,o}, \ldots, W_{n,m}$

LET $\delta = (n+1)n$

$A^{1}_{n,j}$: 1

$A^{3}_{n,o}$: $9\delta-3$

$A^{3}_{n,1}$: $A^{3}_{n,o}-15$ (i.e. (numerator of $A^{3}_{n,o}$)-15)

$A^{5}_{n,o}$: $(225\delta-750)\delta + 180$

$A^{5}_{n,1}$: $A^{5}_{n,o}-1050\delta+2520$

$A^{5}_{n,2}$: $A^{5}_{n,1}-4200\delta+21420$

$B^{1}_{n,j}$: $3j$

$B^{3}_{n,o}$: 0

$B^{3}_{n,1}$: $(75\delta-180)\delta+60$

$B^{3}_{n,o}$: $(150\delta-990)\delta+330$

$B^{5}_{n,o}$: 0

SET: $Z_1 = (4\delta-39)\delta+90$ $\qquad Z_2 = (15\delta-50)\delta+12$

$W = -770\delta+1155$ $\qquad Z = -Z_1($[illegible]$20\delta-140)$ $\qquad Z_1 = Z_1((300\delta-300)\delta+100)+11Z_2Z_2$

THEN FOR j = 1, 2, 3

$B^{5}_{n,j}$: $(((693jj+W)Z_2+Z)jj+Z_1)j$

$C^1_{n,j}$: 0

$C^3_{n,o}$: -30δ

$C^3_{n,1}$: $C^3_{n,o}+90$

$C^5_{n,o}$: $((-1050\delta+3675)\delta-3150)\delta$

$C^5_{n,1}$: $C^5_{n,o}+(8820\delta\ -27090)\delta+18900$

$C^5_{n,2}$: $C^5_{n,o}+\ (35280\delta-221760)\delta+170100$

APPENDIX III. Initial Segments of Special Arrays for Computing Weights by Differences

Values are given in the following tables so that complete tables of weights may be made by differences without any direct evaluation. Also given are the factors which will appear in both numerator and denominator of the formulae. These factors have been cancelled in these tables.

TABLE I: 3RD DEGREE SMOOTHING

j \ n	2	3	4	5
0	17	35	59	
1	12	30	54	
$S_{3,n}$ = denom	35	105	231	429

$A^3_{n,j}$

factors 3

even in j, degree 2

TABLE II: 5hTH DEGREE SMOOTHIN

j \ n	3	4	5	6	7	8
0	393	1253	3003	6093	11063	
1	225	945	2520	5400	10125	
2	-90	210	1260	3510	7500	
$S_{5,n}$ = denom	693	3003	9009	21879	46189	88179

$A^5_{n,j}$

factors $2^2 \cdot 3 \cdot 5$

even in j, degree 4

TABLE III: 1ST DERIVATIVE BY 3RD DEGREE SMOOTHING

j \ n	2	3	4	5	6	7	8	9
0	0	0	0	0	0			
1	56	290	882	2072	4160			
2	-7	335	1351	3521	7445			
$S_{3,n}$ = denom	84	1260	8316	36036	120120	334152	813960	1790712

$B^3_{n,j}$

factors 2·3·5

odd in j, degree 3

TABLE IV: 1ST DERIVATIVE BY 5TH DEGREE SMOOTHING

j \ n	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1485	20153	129528	564840	1926375	5531295	13972728	31954728	67481505			
2	-297	15883	158508	823338	3079725	9351573	24509058	575[illegible]8308	123735843			
3	33	-9667	52458	603198	2930411	10157763	28816158	71227658	158937093			
$S_{3,n}$ = denom	1980	60060	720720	5250960	27713400	116396280	411863760	1274816400	3543239700	9013504500	21289039200	47206130400

$B^5_{n,j}$

factors $2^4 \cdot 3^3 \cdot 5 \cdot 7$

odd in j, degree 5

TABLE V: 2ND DERIVATIVE BY 3RD DEGREE SMOOTHING

j \ n	2	3	4	5	6	7
0	-6	-12	-20			
1	-3	-9	-17			
$S_{3,n}$ = denom	21	126	462	1,287	3,003	6,188

$C^3_{n,j}$

factors $2 \cdot 3 \cdot 5$

even in j, degree 2

TABLE VI: 2ND DERIVATIVE BY 5TH DEGREE SMOOTHING

j \ n	3	4	5	6	7	8	9	10	11	12
0	-350	-1850	-6650	-18900	-45780	-98700	-194700			
1	-95	-1055	-4760	-15080	-38859	-87115	-176440			
2	335	755	35	-4855	-19751	-54495	-124335			
$S_{5,n}$ = denom	660	8,580	60,060	291,720	1,108,536	3,527,160	9,806,280	24,515,700	56,241,900	120,180,060

$C^5_{n,j}$

factors $2^2 \cdot 3^3 \cdot 5 \cdot 7$

even in j, degree 4

Tables of Forward Differences Diagonals with Respect to n.
NOTE: These numbers are used right to left in the program description.

Table 1a: 3rd Degree Smoothing

j	AT n=2		
0	17	18	6
1	12	18	6

Table 2a: 5th Degree Smoothing

j	AT n=3				
0	393	860	890	450	90
1	225	720	855	450	90
2	-90	300	750	450	90

Table 3a: 1st Derivative by 3rd Degree Smoothing

j	AT n=2				
0	0	0	0	0	0
1	56	234	358	240	60
2	-7	342	674	480	120

Table 4a: 1st Derivative by 5th Degree Smoothing

j	AT n=3								
0	0	0	0	0	0	0	0	0	0
1	1485	18668	90707	235230	365056	351820	207270	68600	9800
2	-297	16180	126445	395760	673592	680960	410760	137200	19600
3	33	-9700	71825	416790	871068	964740	606690	205800	29400

Table 5a: 2nd Derivative by 3rd Degree Smoothing

j	AT n=2		
0	-6	-6	-2
1	-3	-6	-2

Table 6a: 2nd Derivative by 5th Degree Smoothing

j	AT n=3						
0	-350	-1500	-3300	-4150	-3030	-1200	-200
1	-95	-960	-2745	-3870	-2974	-1200	-200
2	335	420	-1140	-3030	-2806	-1200	-200

Tables of Backward Difference Diagonals with Respect to n.

NOTE: These numbers are used right to left in the program description

Table 1b: 3rd Degree Smoothing

j	AT n = 2		
0	17	12	6
1	12	12	6

Table 2b: 5th Degree Smoothing

j	AT n = 3				
0	393	330	260	180	90
1	225	225	225	180	90
2	-90	-90	120	180	90

Table 3b: 1st Derivative by 3rd Degree Smoothing

j	AT n = 2				
0	0	0	0	0	0
1	56	56	58	60	60
2	-7	28	74	120	120

Table 4b: 1st Derivative by 5th Degree Smoothing

j	AT n = 3								
0	0	0	0	0	0	0	0	0	0
1	1485	1485	1485	1482	1476	1470	1470	0	9800
2	-297	-297	-339	-456	-648	-840	-840	0	19600
3	33	-1353	-3061	-5274	-7992	-10710	-10710	0	29400

Table 5b: 2nd Derivative by 3rd Degree Smoothing

j	AT n = 2		
0	-6	-4	-2
1	-3	-4	-2

Table 6b: 2nd Derivative by 5th Degree Smoothing

j	AT n = 3						
0	-350	-320	-290	-260	-230	-200	-200
1	-95	-111	-127	-148	-174	-200	-200
2	335	336	302	188	-6	-200	-200

Complete Initial Difference Tables For The Weights, Displaying The Point-Value, Forward Diagonal, and Backward Diagonal Forms At Minimum n.

$A^3_{n,j}$ j = 0

```
                      17   35   59   point-value
                    12   18   24
         backward  6          6   forward
```

j = 1

```
                      12   30   54
                    12   18   24
                   6          6
```

$A^5_{n,j}$ j = 0

```
          393  1253  3003  6093 11063
       330  860  1750  3090  4970
    260       890  1340  1880
  180              450   540
 90                   90
```

j = 1

```
           225  945  2520  5400 10125
       225  720  1575  2880  4725
    225       855  1305  1845
  180              450   540
 90                   90
```

j = 2

```
          -90  210  1260  3510  7500
       -90 300  1050  2250  3990
    120       750  1200  1740
  180              450   540
 90                   90
```

$B^3_{n,j}$ j = 1

```
             56  290   882  2072  4160
          56 234  592  1190  2088
       58      358   598   898
    60           240     300
  60                 60
```

j = 2

```
             -7  335  1351  3521  7445
          28 342  1016  2170  3924
       74      674  1154  1754
    120          480     600
  120                 120
```

$\beta^{5}_{n,j}$

```
j=1          1485  20153 129528 564840 1926375 5531295 13972728 31954728 67481505
        1485 18668 109375 435312 1361535  3604920 8441433 17982000  35526777
       1485       90707 325937 926223 2243385 4836513  9540567 17544777
      1482         235230 600286 1317162  2593128 4704054 8004210
     1476                365056 716876 1275966 2110926  3300156
    1470                   351820  559090   834960 1189230
   1470                          207270  275870  354270
    0                                   68600   78400
  9800                                      9800

j=2        -297  15883 158508  823338 3079725 9351573 24509058 57508308 123735843
       -297 16180 142625  664830 2256387  6271848 15157485 32999250 66227535
      -339       126445 522205 1591557 4015461 8885637 17841765 33228285
     -456         395760 1069352 2423904  4870176  8956128 15386520
    -648                673592 1354552 2446272 4085952  6430392
   -840                    680960 1091720  1639680  2344440
  -840                            410760  547960  704760
   0                                  137200   156800
19600                                     19600

j=3          33  -9667  52458  603198 2930411 10157763 28816158 71227658 158937093
      -1353  -9700 62125  550740 2327213  7227352 18658395 42411500  87709435
     -3061        71825 488615 1776473 4900139 11431043 23753105 45297935
    -5274          416790 1287858 3123666  6530904 12322062 21544830
   -7992                 871068 1835808 3407238  5791158  9222768
  -10710                    964740 1571430  2383920  3431610
 -10710                            606690  812490  1047690
    0                                  205800   235200
29400                                      29400
```

$C^{3}_{n,j}$ $j=0$

-6 -12 -20
-4 -6 -8
-2 -2

$j=1$

-3 -9 -17
-4 -6 -8
-2 -2

$C^{5}_{n,j}$ $j=0$ -350 -1850 -6650 -18900 -45780 -98700 -194700
-320 -1500 -4800 -12250 -26880 -52920 -96000
-290 -3300 -7450 -14630 -26040 -43080
-260 -4150 -7180 -11410 -17040
-230 -3030 -4230 -5630
-200 -1200 -1400
-200 -200

$j=1$ -95 -1055 -4760 -15080 -38859 -87115 -176440
-111 -960 -3705 -10320 -23779 -48256 -89325
-127 -2745 -6615 -13459 -24477 -41069
-148 -3870 -6844 -11018 -16592
-174 -2974 -4174 -5574
-200 -1200 -1400
-200 -200

$j=2$ 335 755 35 -4855 -19751 -54495 -124335
336 420 -720 -4890 -14896 -34744 -69840
302 -1140 -4170 -10006 -19848 -35096
188 -3030 -5836 -9842 -15248
-6 -2806 -4006 -5406
-200 -1200 -1400
-200 -200