Restricted Range Adaptive Curve Fitting

J. A. Hull, G. D. Taylor

Performing Organization Name and Address
Colorado State University
Department of Mathematics
Fort Collins, CO 80523

CONTROLLING OFFICE NAME AND ADDRESS
AFOSR/NN
Bldg. 410
Rollins AFB, D.C. 20332

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Curve fitting, data fitting, adaptive curve fitting with user imposed constraints.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
In this paper we present an algorithm for adaptively computing smooth piecewise polynomial approximations using restricted range uniform approximations. We also present several numerical examples, and offer suggestions for the effective use of this algorithm. We have found the algorithm to be effective for approximating a wide class of functions, either with or without significant levels of noise. Furthermore, since the user of this algorithm actually defines tolerance bands within which the approximation will lie, the algorithm allows...
the user a great deal of flexibility and control over the shape of the resulting approximations. A FORTRAN code of this algorithm is included in an appendix at the end of the paper.
RESTRICTED RANGE ADAPTIVE CURVE FITTING

J. A. Hull and G. D. Taylor

Applied Technology, 645 Almanor, Sunnyvale, California 94086
Department of Mathematics, Colorado State University, Ft. Collins, Colorado 80523

ABSTRACT

In this paper we present an algorithm for adaptively computing smooth piecewise polynomial approximations using restricted range uniform approximations. We also present several numerical examples and offer suggestions for the effective use of this algorithm. We have found the algorithm to be effective for approximating a wide class of functions, either with or without significant levels of noise. Furthermore, since the user of this algorithm actually defines tolerance bands within which the approximation will lie, the algorithm allows the user a great deal of flexibility and control over the shape of the resulting approximations.

I. Introduction

Let \( X \) be a finite set of real points and let \( f \) be a function defined on \( X \) or let data be given in tabular form. In the case of data given in tabular form, say \( \{(t_i, y_i)\}_{i=1}^M \), we shall set \( X = \{t_i\}_{i=1}^M \) and define \( f \) on \( X \) by \( f(t_i) = y_i \), \( i = 1, \ldots, M \). In what follows we shall use this functional notation. Let \( a = \min\{x: x \in X\} \) and \( b = \max\{x: x \in X\} \). For any function \( g \) defined on \( X \) define \( \|g\|_X = \max\{|g(x)|: x \in X\} \). Let SMTH and \( N \) be nonnegative integers supplied by the user, with \( N > \text{SMTH} \). Let \( u(x) \) and \( \ell(x) \) be functions defined on \( X \) supplied by the user such that for each \( x \in X \), we have \( \ell(x) < f(x) < u(x) \) and \( \ell(x) < u(x) \). In this setting our algorithm will calculate a piecewise polynomial approximation, \( p \), to \( f \), and a set of points \( \{x_i\}_{i=0}^k \subset X \) with \( a = x_0 < x_1 < \ldots < x_k = b \) such that \( p \) restricted to \( [x_i, x_{i+1}] \) is a polynomial \( p_i \in \Pi_{N-1} \) \( \{q: \) \( q \) is a real algebraic polynomial of degree \( \leq N-1 \} \), \( p \) has \( \text{SMTH} \) continuous derivatives (on \( [a, b] \)) and for each \( x \in X \), \( \ell(x) < p(x) < u(x) \). By appropriately choosing \( \ell(x) \) and \( u(x) \), the user obtains an approximation meeting a given preselected (set of) tolerance(s).

II. The Algorithm

The algorithm begins by choosing \( x_1 \) to be the largest point in \( X \) such that

1) \( [a, x_1] \cap X \) contains at least \( N+1 \) points and

2) There is a best restricted range uniform approximation \( p_1 \) from \( \Pi_{N-1} \) to \( f \) (with respect to the constraining curves \( u(x) \) and \( \ell(x) \)) on \( [a, x_1] \cap X \); that is, there exists \( p_1 \in \Pi_{N-1} \) for which

\[
\|f - p_1\|_{[a, x_1] \cap X} = \inf_{q \in \Pi_{N-1}} \|f - q\|_{[a, x_1] \cap X};
\]

\( \ell(x) < q(x) < u(x) \) for all \( x \in [a, x_1] \cap X \).
If $x_1 = b$, then since $p_1$ is a piecewise polynomial meeting our requirements, we successfully terminate the algorithm. If no such $x_1$ exists, the algorithm fails and an appropriate error message is generated. Otherwise, if $SMT = 0$ (i.e., we only require the approximation to be continuous) we choose the right endpoint of the first subinterval, $x_1$, to be $x_1$. If $SMT > 0$, in order to add to the stability of the algorithm we (in general) choose $x_1$ by "backing off" from $x_1$ in the following manner.

We first examine the error curve $f(x) - p_1(x)$ and find those points $\xi_1, \xi_2, \ldots, \xi_N$ in $(a, x_1] \cap X$ at which relative extrema occur. We will choose one of the $\xi_i$'s to be $x_1$. The motivation for choosing $x_1$ in this manner is that in the continuous setting, if $f$ is differentiable and $\xi$ is an interior relative extreme point of $f(x) - p_1(x)$, then $f'(\xi) - p_1'(\xi) = 0$ so that the derivative of $p_1$ at $\xi$ would match that of $f$ at $\xi$. This guarantees that when we smoothly join the next piece of the approximation to $p_1$ at $\xi$, this next piece will closely follow the direction of $f$ at least near $\xi$. If we merely joined the second piece to the first at $x_1$, no such guarantee can be made, and, in fact, severe oscillatory problems tend to set in. Our numerical experience indicates that the procedure of backing off from $x_1$ to a smaller $x_1$ contributes very significantly toward the stability of the algorithm. To continue, let $f'(\xi)$ be the derivative of the centered quadratic interpolation of $f$ evaluated at $\xi$. We then choose $x_1$ to be the largest $\xi$ such that $|f'(\xi) - p_1'(\xi)| < \text{EPS}$, where EPS is a tolerance which can be set by the user, or, if there does not exist such a $\xi$, then we let $x_1$ be the largest $\xi$ at which $|f'(\xi) - p_1'(\xi)|$ is a minimum. (In our implementation of the algorithm, we do not in general consider all of the relative extreme points of $f(x) - p_1(x)$ in $[a, x_1] \cap X$, but only the largest $N - SMT - 1$ of them.)
We continue by finding successive intervals \([x_1, x_2], [x_2, x_3], \ldots, [x_{m-1}, b]\) and corresponding polynomial approximations \(p_2, p_3, \ldots, p_m \in \Pi_{N-1}\) to \(f\) so that \(p_{v-1}^{(j)}(x_{v-1}) = p_v^{(j)}(x_{v-1})\) for \(j = 0, 1, \ldots, \text{SMTH}\) and \(i(x) \leq p_v(x) \leq u(x)\) for every \(x \in [x_{v-1}, x_v] \cap X\) for \(v = 2, \ldots, m, x_m = b\). This is accomplished as follows:

Suppose we have found the subintervals \([a, x_1], [x_1, x_2], \ldots, [x_{i-1}, x_i]\), and the corresponding approximations \(p_1, p_2, \ldots, p_{i-1}\). Assume further that \([x_{i-1}, b] \subset X\) contains at least \(\max(2, N-\text{SMTH})\) points. We now determine an \(x_i\) and a \(p_i\) meeting the above requirements. We begin by choosing \(x_i \in X\) to be the largest point in \(X\) which satisfies

1) \([x_{i-1}, x_i]\) contains at least \(\max(2, N-\text{SMTH})\) points and

2) There exists a best restricted range uniform approximation, \(p_i\), to \(f\) on \([x_{i-1}, x_i] \cap X\), subject to the constraint that \(p_i^{(j)}(x_{i-1}) = p_{i-1}^{(j)}(x_{i-1})\), \(j = 0, 1, \ldots, \text{SMTH}\).

If \(x_i = b\), we set \(x_i = x_i = b\), and the algorithm is successfully terminated. If no such \(x_i\) exists, the algorithm fails and is terminated. Otherwise, we choose \(x_i\) completely analogous to our choice of \(x_i\) above.

Finally, we consider the special case where \([x_{i-1}, b] \cap X\) contains fewer than \(\max(2, N-\text{SMTH})\) points. Specifically, we choose \(x_{i-1}\) to be a point in \(X\) closest to \((b - x_{i-2})/2\) which satisfies

1) \([x_{i-2}, b] \cap X\) contains at least \(\max(2, N-\text{SMTH})\) points and

2) There exists a best restricted range approximation, \(p_2\), to \(f\) from \(\Pi_{N-1}\) on \([x_{i-2}, b] \cap X\) subject to the constraint that \(p_1^{(j)}(x_{i-1}) = p_{i-1}^{(j)}(x_{i-1})\), \(j = 0, 1, \ldots, \text{SMTH}\).

Again, if we can find such an \(x_{i-1}\) then we change \(x_i\) to \(x_{i-1}\); set \(x_i = b\), and successfully terminate the algorithm. If we cannot find such an \(x_{i-1}\), the algorithm is terminated and an appropriate error message is generated.
Remark. In our implementation of this algorithm, the $\hat{x}_i$ are chosen as follows. At each step of this iterative procedure we will let $\hat{a}$ be the current largest point in $X$ such that requirements (1) and (2) are satisfied on $[x_{i-1}, \hat{a}] \cap X$, and we will let $\hat{b}$ be the current smallest point in $X$ such that $\hat{b} > \hat{a}$ and requirement (2) fails to be satisfied. We initialize this process by computing (or attempting to compute) the best restricted approximation on $[x_{i-1}, b] \cap X$ subject to the smoothness interpolatory constraints. If this approximation satisfies requirement (2), then we set $\hat{x}_i = b$ and we are done. If the approximation fails to satisfy (2), we set $\hat{b} = b$. Next, let $t = \min\{x \in X: [x_{i-1}, x] \cap X$ contains at least $\max(2, N_{\text{SMTH}})$ points$\}$. If the approximation on $[x_{i-1}, t] \cap X$ fails to satisfy (2) then the algorithm cannot meet the desired accuracy and fails. Otherwise, we set $\hat{a} = t$.

In general, we proceed as follows. We let $t = \inf\{x \in X: (\hat{b} - \hat{a})/2 < x < \hat{b}\}$. If this set is empty, we set $t = \sup\{x \in X: \hat{a} < x < (\hat{b} - \hat{a})/2\}$. If $t = \hat{a}$ then this procedure has converged and we set $\hat{x}_i = t = \hat{a}$. Otherwise, we compute (or attempt to compute) the best restricted range approximation with interpolatory constraints on $[x_{i-1}, t] \cap X$. If this approximation satisfies (2) then we set $\hat{a} = t$. If this approximation fails to satisfy (2) then we set $\hat{b} = t$. We continue this process until $\hat{b} - \hat{a}$ is less than some user definable prescribed tolerance, at which point we accept $\hat{a}$ as a good approximation to $\hat{x}_i$ and terminate this procedure. We compute the $\hat{x}_i$ in a manner analogous to the above.

We have implemented a Remes-like algorithm to compute best restricted range uniform approximations with interpolatory constraints. See [2] for a complete discussion of this problem.
III. Numerical Results

By setting \( u(x) = f(x) + \text{TOL} \) and \( l(x) = f(x) - \text{TOL} \) for \( x \in X \), where \( \text{TOL} \) is some positive, preselected tolerance, this algorithm simplifies to the best uniform approximation operator version of the algorithm given in [3]. For this reason we do not consider here any examples with restraining curves differing from the function being approximated by a constant. Indeed, the real value of this algorithm is that the tolerance we require our approximation to satisfy may vary from point to point. Consequently, where \( f \) is "nice" we can force our approximation to be close to \( f \), and where \( f \) is "bad" we can relax our requirements, thereby obtaining an approximation which more closely reflects the character of \( f \) than can be obtained by selecting a fixed tolerance throughout the domain of approximation. In the case of experimental data which contain considerable levels of noise, the user can force the approximation to lie on the "believable" side of the data; often, more useful approximations can be obtained with this algorithm than can be obtained with (for example) the discrete \( L^2 \) version of the algorithm given in [3].

This algorithm has been implemented as a FORTRAN program running on Colorado State University's CDC CYBER 172 and CDC 6400. In the appendix we give a listing of the algorithm. As examples, we now present approximations to experimental data involving the release of bitumen and gas and oil from oil shale heated to a constant temperature as a function of time. Because relatively few data points were available (14-20), we filled in the gaps between the data points by discretizing (200-500 points) the linear interpolation of the original data using an algorithm which added (somewhat) more points in regions where the function
being approximated was complicated (i.e., radically changing slopes between data points) and fewer points in regions where the function is smooth. The advantage of this unequal spacing over equally spaced points is that in regions where the function being approximated is complicated, the procedure of "backing off" from $x_i$ to $x_1$ becomes more effective by having more densely packed points. Also, the subintervals $[x_{i-1}, x_i]$ may be chosen to be smaller (if needed in order to obtain a close enough approximation) while $[x_{i-1}, x_i] \cap X$ still contains at least $\max(2, N - \text{SMTH})$ points. Only the original data points (indicated by "x") are shown in the following plots. In each case we chose $N = 6$, and $\text{SMTH} = 2$. The curves $\xi(x)$ and $\mu(x)$ were chosen by hand or by means of a simple algorithm at the original data points, and then they, too, were "filled in" using the same linear interpolation scheme as above. The TOL parameter listed on the plots is the largest tolerance allowed at any point throughout the approximation. This error is not in general reached.

Figure 1
Although the maximum tolerance in both of these examples is fairly large, the tolerances throughout most of these intervals of approximation were on the order of .15 to 1.5. That is, we required fairly close agreement with the function being approximated except at the "bad" points. As an example of what can be done by appropriately choosing the restraining curves, we chose a band containing the above data and computed restraining curves based on this band as the following plot shows. The curve in the center is the resulting approximation.
For clarity, we repeated the above plot but without the restraining curves.
For additional examples using this algorithm and a similar algorithm using best \( L^2 \) approximations, see [1].

By appropriately setting the restraining curves the user can, to a large extent, determine the shape of the approximation. The most effective way of determining such restraining curves seems to be trial and error. Ideally, these restraining curves would be determined in an interactive setting using a graphics terminal with a pen light as follows. First one would make a rough initial guess at what the restraining curves should be (using some simple algorithm or otherwise), then allow the
algorithm to compute the first piece of the approximation. One would then display the data, the current approximation and the current restraining curves and modify the restraining curves on the relevant subinterval as desired so that when this first piece is recomputed using the updated restraining curves, it behaves as desired. After the user is satisfied with the first piece, he would repeat the above strategy on each successive subinterval as they are determined by the algorithm.

REFERENCES


APPENDIX

Here we give a listing with driver, sample input and output which corresponds to Figure 1. The sample input is located after the listing of the code and prior to the sample output. This example uses the algorithmically defined restraining curves. In addition, instructions for user defined restraining curves are given in the sample input section.
PROGRAM DRIVER(INPUT,OUTPUT,TAPES=INPUT,TAPE=OUTPUT)
LOGICAL ERROR
COMMON XTAB(500),Ftab(500),DUMMY(1562),FU(500),FL(500)
INTEGER SMTH
REAL U(5),X(5),TOL,TMLTOL
READ(5)U(5),TOL
IF(IOPT.EQ.0)WRITE(0,200)
WRITE(6)U(5)
WRITE(4)U(5),TOL
MAX=MAX+1
READ(5)X(5),F(5),FML(MAX),FU(MAX),FL(MAX)
IF(IOPT.EQ.0)GO TO 10
MAX=MAX+1
CALL SETING(MAX+1),SMTH,TMLTOL
CALL LINEAR(MAX+1),SMTH,SMTH
CALL RPACF(MAX+1),SMTH,SMTH
50 FORMAT(11)
10 FORMAT(215,2F10.5)
150 FORMAT(4F10.5)
200 FORMAT(15H3,6G15.15H5RESERVED RANGE ALGORITHM--USER DEFINED RESTRAIN
250 FORMAT(15H3,6G15.15H5SING CURVES)
300 FORMAT(15H3,6G15.15H5RESTRICTED RANGE ALGORITHM--ALGORITHMICALLY DEFIN
350 FORMAT(15H3,6G15.15H5SE RESTRAINING CURVES)
400 FORMAT(15H3,6G15.15H5NUMBER OF COEFFS=121H12*25M CONTINUOUS DERIV
450 FORMAT(15H3,6G15.15H5
500 FORMAT(15H3,6G15.15H5END
SUBROUTINE LINEAR(OLDMAX,NAPROX,NTRUE,R)
C THIS SUBROUTINE FILLS IN BETWEEN THE ORIGINAL DATA POINTS
C ABOUT NAPROX LINEARLY INTERPOLATED DATA POINTS. THE SPACING USED
C DEPENDS UPON THE DATA, FOR #SMOOTH DATA, 1-E* DATA SUCH THAT CON-
C SECUTIVE LINE SEGMENTS JOINING THE DATA POINTS DIFFER ONLY SLIGHTLY
C IN SLOPE, EQUI-SPACED POINTS ARE USED. FOR MORE COMPLEX DATA, THE
C SPACING IS SUCH THAT MORE POINTS ARE CONCENTRATED IN AREAS WHERE
C THE FUNCTION BEING APPROXIMATED IS COMPLICATED AND FEWER WHERE IT
C IS SIMPLE. THE MAXIMUM SPACING BETWEEN POINTS IS DELTA*H, WHERE
C DELTA IS THE SPACING WHICH WOULD RESULT IF THE NAPROX EQU-
C SPACED POINTS, AND H IS A CONSTANT CHOSEN BY THE USER WHICH IS
C GREATER THAN OR EQUAL TO 1.0. (NOTE WHEN X IS SET TO 1.0, EQUI-
C SPACING OF POINTS OCCURS REGARDLESS OF THE NATURE OF THE DATA,
C AS THE ORIGINAL DATA SET IS INCLUDED AS A SUBSET OF THE NEW DATA
C SET THE NUMBER OF POINTS IN THE NEW DATA SET IS BETWEEN NAPROX
C AND NAPROX*OLDMAX. THE TRUE VALUE IS RETURNED TO THE USER IN
C NTRUE.
C
LOGICAL EOSPDCO
REAL MINVAL,H
INTEGER OLDMAX,OLDM,
COMMON XTAB(500),Ftab(500),XTAB(316),FSTR(316),FL(316),FUST(316),FLST
IR(317),SUB(317),FU(500),FL(500)
D(317),FSTR(317),FSTM(317),FST(317),FLST(316)
1(XSTR(1)-XTAB(I)),(FSTR(1)-FST(1)),(FSTR(I)-FST(1)),(FSTR(1)-FST(I-1)),
1(XSTR(I)-XTAB(1)),(FSTR(I)-FST(1)),(FSTR(I)-FST(1)),
AREA(I)=.5*(SUB(I)+SUB(I-1))
DATA EPS=OMEGA/1.0E=-8,1.0E=-5
DO 14 I=1,OLDMAX
XSTR(I)=XTAB(I)
FSTR(I)=FSTR(I)
FSTM(I)=FSTM(I)
14 CONTINUE

LINR10
LINR20
LINR30
LINR40
LINR50
LINR60
LINR70
LINR80
LINR90
LINR100
LINR110
LINR120
LINR130
LINR140
LINR150
LINR160
LINR170
LINR180
LINR190
LINR200
LINR210
LINR220
LINR230
LINR240
LINR250
LINR260
LINR270
LINR280
LINR290
LINR300
LINR310
LINR320
LINR330
10 CONTINUE
EQSPC0=TRUE
IF (IR.EQ.1) GO TO 55
OLDM1=OLDMAX-1
SML=ABS(U(2))
DO 20 I=2,OLDM1
    TEMP=ABS(U(I))
    QSUB(I)=TEMP
    IF (TEMP.LT.SML) SML=TEMP
20 CONTINUE
QSUB(I)=0.0
QSUB(OLDMAX)=0.0
DO 30 I=2,OLDM1
    QSUB(I)=QSUB(I)-SML
    MINVL=1
    MINVAL=QSUB(2)
    TOTAR=MINVAL*5*(XSTR(2)-XSTR(I))
        DO 40 J=I+1,OLDM1
            TEMP=QSUB(I)+QSUB(J)
            TOTAR=TOTAR+5*TEMP*ABS(XSTR(J)-XSTR(I))
        IF (TOTAR.GE.MINVL) GO TO 40
    MINVL=TOTAR
    MINVL=1
40 CONTINUE
DELTA=(XSTR(OLDMAX)-XSTR(I))/FLOAT(NAPROX-1)
OLTMX=DELTA
IF (TOTAR.LT.OLTMX) GO TO 55
EQSPCD=FALSE
M=ABS(QSUB(MINVL+1)-QSUB(MINVL))/(XSTR(MINVL+1)-XSTR(MINVL))
H=(TOTAR/FLOAT(NAPROX-1)-5*M*OLTMX*DELTA)/(DELTA*DELTA)
IF (H.LT.1.0) H=1.0
DO 50 I=1,OLDMAX
    QSUB(I)=QSUB(I)+H
50 CONTINUE
CK=(TOTAR+M*(XSTR(OLDMAX)-XSTR(I)))/FLOAT(NAPROX-1)
55 K=0
I=0
XX=XSTR(I)
60 I=I+1
IF (XX.LT.XSTR(K+1)) GO TO 70
K=K+1
XX=XSTR(K)
XTABLE(I)=XX
FTABLE(I)=FSTR(K)
FUN(I)=FUSTR(K)
FLN(I)=FLSTR(K)
IF (K.EQ.OLDMAX) GO TO 110
DX=XSTR(K+1)-XSTR(K)
SLF=(FSTR(K+1)-FSTR(K))/DX
SLF=(FUSTR(K+1)-FUSTR(K))/DX
SLF=(FLSTR(K+1)-FLSTR(K))/DX
M=(QSUB(K+1)-QSUB(K))/DX
IF (.NOT.EQSPCD) GO TO 80
XX=XX+DELTA
GO TO 60
70 XTABLE(I)=XX
DIFF=XX-XSTR(K)
FTABLE(I)=FSTR(K)+DIFF*SLF
FUN(I)=FUSTR(K)+DIFF*SLF
FLN(I)=FLSTR(K)+DIFF*SLF
IF (.NOT.EQSPCD) GO TO 80
XX=XX+DELTA
GO TO 60

END
GO TO 60
80 QOFX=I*M*(XX-XSTR(K))+QSUB(K)
   IF (ABS(M)+.LT.EPS) GO TO 90
   TEMP=QOFX*2*2.0*EPS
   IF (TEMP-.LT.0.0) GO TO 100
   XX=XX-(-QOFX*SQRT(TEMP))/M
   GO TO 60
   90 XX=XX/C/QOFXX
   GO TO 60
   100 XX=XSTR(K+1)
   GO TO 60
   NTRUE=I
   RETURN
C
END

SUBROUTINE SETRNG (MAXNUM+MINTOL+MAXTOL)
C
THIS SUBROUTINE USES CONVEX COMBINATIONS OF MINTOL AND MAXTOL TO SET
THE FU AND FL TOLERANCES AT EACH DATA POINT DEPENDING ON THE COM-
PLEXITY OF THE FUNCTION BEING APPROXIMATED. THAT IS, WHERE THE FUN-
CTION IS SMOOTHEST, THE TOLERANCES WILL BE CLOSE TO (BUT AT LEAST AS
BIG ON AT LEAST ONE SIDE) AS MINTOL+ AND WHERE THE APPROXIMATION IS
COUPLED THE APPROXIMATION WILL BE CLOSE TO (BUT NO BIGGER THAN)
MAXTOL ON AT LEAST ONE SIDE. FOR DATA WHICH IS NOT SMOOTH AS DES-
CRIBED IN SUBROUTINE LITTLE AT THE TOLERANCES BECOME SOMEWHAT LESS
SIGNIFICANT, AND ARE SET TO MAXTOL AT ALL DATA POINTS. THIS
SUBROUTINE FREES THE USER FROM HAVING TO CHOOSE AN INITIAL BAND OF
TOLERANCES--IT IS NOT NECESSARILY INTENDED TO BE A TOTALLY AUTO-
OMATIC TOLERANCE BAND SELECTION FOR ANY ARBITRARY FUNCTION--EXPER-
IMENTING WITH USER SUPPLIED TOLERANCES MUST OFTEN RESULT IN MORE
DESIRABLE FITS.
C
REAL MINTOL,MAXTOL
COMMON XTABLE(500),FTABLE(1582),GOSTR(500),FU(500),FL(500)
DATA OMEGA/1.0E=9,
Q1=1/(TABLE(1)+TABLE(1)-TABLE(1)+TABLE(1)+TABLE(1))
TABLE(1)=TABLE(1)-TABLE(1)
MAXM=MAXNUM=1
QAVE=I*0
QSTR(2)=QAVE
BIG=ABS(QAVE)
SML=3 * BIG
DO I=1,MAXM
   TEMP=Q(I)
   QSTR(I)=TEMP
   QAVE=QAVE+TEMP
   TEMP=MAX(TEMP)
   IF (TEMP-LT-BIG) BIG=TEMP
   IF (TEMP-LT-SML) SML=TEMP
   CONTINUE
   DIFF=BIG-SML
   IF (DIFF-LT-OMEGA) GO TO 40
   QAVE=QAVE/FLTAT(MAXNUM-2)
   DO 30 I=1,MAXM
      TEMP=QSTR(I)
      TOLI=MINTOL*(BIG-TEMP)/DIFF+MAXTOL*(TEMP-SML)/DIFF
      TSUB=10L1*(BIG-TEMP)/DIFF
      IF (QSTR(I)-LT-0.0) GO TO 20
      FU(I)=TOLI
      FL(I)=TSUB
      30 CONTINUE
      NTRUE=I
      RETURN
C
END
GO TO 30
20 FU(1) = SUB
FL(1) = TOL
30 CONTINUE
FU(1) = FU(2)
FL(1) = FL(2)
FU(MAXNUM) = FU(MAXN)
FL(MAXNUM) = FL(MAXN)
RETURN
40 DO 50 I = 1, MAXNUM
FU(I) = MAXTOL
FL(I) = MAXTOL
50 CONTINUE
RETURN
C END
SUBROUTINE RRACF(N,SMTH,MAXNUM,ERROR)
C
C THIS SUBROUTINE ADAPTIVELY COMPUTES A PIECEWISE POLYNOMIAL APPROX-
CIMATION OF DEGREE N-1 TO THE FUNCTION STORED IN THE ARRAYS XTABLE AND
C FTABLE WITH SMTH CONTINUOUS DERIVATIVES HAVING THE PROPERTY THAT
C 1+LE*1.0. MAXNUM (SEE BELOW) THE VALUE OF THE APPROXIMATION
C EVALUATED AT XTABLE(1) LIES BETWEEN (FTABLE(1) - FL(1)) AND
C (FTABLE(1) + FU(1)), WHERE THE ARRAYS FL AND FU CONTAIN THE DESIRED
C TOLERANCE REQUIRED OF THE APPROXIMATION BELOW THE CURVE AND ABOVE THE
C CURVE (RESPECTIVELY) AT EACH OF THE (MAXNUM) POINTS BEING APPROX-
CIMATED.
C
C THE PARAMETERS ARE AS FOLLOWS--
C
N = THE NUMBER OF COEFFICIENTS OF EACH POLYNOMIAL PIECE, I.E.
ONE MORE THAN THE DEGREE OF THE PIECEWISE POLYNOMIAL APPROX.
AS THE ARRAYS ARE CURRENTLY DIMENSIONED, IT IS ASSUMED THAT
N IS NO BIGGER THAN 17.

SMTH = THE NUMBER OF CONTINUOUS DERIVATIVES DESIRED OF THE
APPROXIMATION. SMTH MUST NOT BE GREATER THAN N-2. IF ONLY
CONTINUITY OF THE APPROXIMATION IS REQUIRED, SET SMTH = 0.

MAXNUM = THE NUMBER OF POINTS ACTUALLY STORED IN THE ARRAYS
XTABLE, FTABLE, FU, AND FL. (AS THESE ARRAYS ARE CURRENTLY
DIMENSIONED, MAXNUM MUST BE LESS THAN 500. IF ONE WISHES TO
COMPUTE APPROXIMATIONS TO FUNCTIONS WITH MORE THAN 500
POINTS, ONE CAN EASILY MODIFY THIS PROGRAM TO CONTINUOUSLY
READ IN MORE POINTS AFTER ROOM IS MADE IN THESE STORAGE
ARRAYS BY HAVING COMPLETED SEVERAL SUBINTERVALS.)

ERROR = A LOGICAL VALUE SET TO 'TRUE' IF AN ERROR OCCURS IN THE
PROGRAM (AN APPROPRIATE ERROR MESSAGE WILL ALSO BE PRINTED)
AND SET TO 'FALSE' OTHERWISE.

BLANK COMMON PROVIDES THE REMAINDER OF THE INPUT. IT IS ASSUMED
THAT THE FIRST 500 WORDS OF BLANK COMMON CONTAIN THE TABLE OF X
VALUES (XTABLE), THE NEXT 500 WORDS THE TABLE OF FUNCTION VALUES
(FTABLE), THE NEXT TWO WORDS ARE USED INTERNALLY THROUGHOUT THE
PROGRAM AND THE NEXT 1080 WORDS IS AN ARRAY (CSTORE(18x60))
CONTAINING THE COMPUTED COEFFICIENTS AND THE KNOTS--I.E.
CSTORE(IxINT) IS THE COEFFICIENT OF THE I-1ST DEGREE TERM IN SUB-
SUBROUTINE STORE STORES THE COEFFICIENTS FOR THIS SUBINTERVAL IN THE
ARRAY CSTORE. IT ALSO PRINTS OUT THE COEFFICIENTS AND THE ERROR OF
APPROXIMATION ON THIS SUBINTERVAL. SUBROUTINE SETP(xn) STORES THE
VALUE OF THE POLYNOMIAL DETERMINED BY THE COEFFICIENTS IN THE ARRAY
C AND ITS FIRST K DERIVATIVES AT THE POINT X IN THE ARRAY PRIME.

CALL STORE (C+LCNTLE+LCNTRE)
CALL SETP (C+XTABLE+LCNTRE)+NLSMT)
LCNTLE+LCNTRE
LCNTRE=MIN(MAXNUM+MAXINT+LCNTLE+1)

20 CONTINUE
WRITE (6+60) NINT
ERROR=TRUE
RETURN

30 CALL LSTINT (C+LENGTH+MAXNUM+ABORT)
IF (ABORT) GO TO 50

40 CALL STORE (C+LCNTLE+LCNTRE)
RETURN

50 WRITE (6+70)
ERROR=TRUE
RETURN

C

END

SUBROUTINE COMPUT (C+LENGTH+MAXNUM+LAST+DONE+ABORT)

THIS SUBROUTINE FINDS THE LARGEST SUBINTERVAL AND THE BEST APPROX-
IMATION TO F ON THIS SUBINTERVAL SUCH THAT THE APPROXIMATION MEETS
THE DESIRED ERROR TOLERANCE ON THE SUBINTERVAL.

INTEGER A,B
LOGICAL LAST+DONE+ABORT+TOOBIG
REAL C(18)
COMMON X+TABLE+LCNTLE+LCNTRE+STORE+N+1+60+CDERIV+500
COMMON SCALAR N+NPLUS1+X+NXM1+NLSMT+NRSMT+NUMPTS+NINT
COMMON COMP+LCNTLE(18)

WE ASSUME THAT WE ARE CLOSE ENOUGH TO THE TRUE LARGEST SUBINTERVAL
RIGHT END POINT WHEN WE KNOW THAT OUR APPROXIMATION TO THE TRUE RIGHT
END POINT IS WITHIN ETA OF THE TRUE END POINT.

DATA ETA+/08/

LITTLE=LCNTLE+LENGTH+1

&
LAST+FALSE

10 NUMPTS=LCNTRE+LCNTLE+1

C IF THERE DOES NOT EXIST A BEST RESTRICTED RANGE APPROXIMATION ON THE
C CURRENT SUBINTERVAL CONTROL IS PASSED TO LINE 30.
C
CALL REMES (C,CLTNX,TOOBIG,ABORT)
IF (ABORT) RETURN
IF (TOOBIG) GO TO 30
IF (CLTNRE-CLT.MAXNUM) GO TO 20
DONE=.TRUE.
RETURN
A IS THE CURRENT LARGEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
THE BEST APPROXIMATION ON THIS SUBINTERVAL SATISFIES ALL CONSTRAINTS.

20 A=CLTNRE
IF ((XTABLE(B)-XTABLE(A)-G.T.ETA) AND (B-A.GT.1)) GO TO 40
GO TO 50
B IS THE CURRENT SMALLEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
THE BEST APPROXIMATION ON THIS SUBINTERVAL FAILS TO SATISFY THE CON-

50 B=CLTNRE
40 NEWTRY=(A+B)/2+1
IF (NEWTRY.EQ.B) NEWTRY=NEWTRY-1
IF (NEWTRY.LT.LITTLE) NEWTRY=LITTLE
IF (NEWTRY.GE.LCTNRE) GO TO 50
LCTNRE=NEWTRY
GO TO 10
IF A IS STILL 0 THEN NO SUBINTERVAL WITH AT LEAST LENGTH POINTS
WILL WORK, SO THE ALGORITHM IS TERMINATED.

50 IF (A,NE,0) GO TO 60
ABORT=.TRUE.
RETURN
SINCE NEWTRY IS ALWAYS STRICTLY LESS THAN THE CURRENT B* IF NEWTRY=
LCTNRE, AND A IS NOT Still 0, NEWTRY<0, WHICH IS A POINT WHICH SAT-
ISFIES ALL REQUIREMENTS. WE NOW BACK THE RIGHT ENDPOINT OFF TO THE
BEST INTERIOR EXTREME POINT OF F-P TO ADD TO THE STABILITY OF THE AL-

GORITHM.

60 DO 70 I=1,N
CERIV(I)=C(I)
CSTORE(I,NINT)=C(I)
70 CONTINUE
NDUMMY=0
CALL DERIV (CERIV,NDUMMY)
I=NX
LCTNRE=CLTNX(I)
NEWTRY=LCTNRE
SMNL=CMHR(CERIV,NDUMMY,NEWTRY,LCTNLE-1,OK)
IF (OK) GO TO 100
80 =1-1
IF (1.EQ.0) GO TO 100
NEWTRY=LCTNRE(I)
IF (NEWTRY.LT.LNGTH) GO TO 100
TEMP=CMHP (CERIV,NDUMMY,NEWTRY,LCTNLE-1,OK)
IF (NOT.OK) GO TO 90
LCTNFER=NEWTRY
GO TO 100
90 IF (TEMP.GE.SMML) GO TO 80
SMML=TEMP

BEST AVAILABLE COPY
LCTNRE=NEWTRY
GO TO 80
100 LCTNRE=LCTNLE+LCTNRE=1
   IF (MAXNUM=LCTNRE+1.LT.LNGTH) LAST=.TRUE.
   RETURN.
C
END

SUBROUTINE LSTINT (C,LNGTH,MAXNUM,ABORT)
C
THIS SUBROUTINE HANDLES THE SPECIAL CASE OF FINDING A SUBINTERVAL
AND A BEST APPROXIMATION ON THAT SUBINTERVAL WHEN THERE ARE TOO
FEW REMAINING POINTS FOR COMPUT TO WORK.
C
INTEGER OLDEL,OLDRE
LOGICAL TOOBIG,ABORT
REAL C(18)
COMMON X(100),LCTNLE,LCTNRE,CSTORE(18,60)
COMMON /SCALAR/ N,NPLUSI,NX,NXH1,NLSMH,NRSMH,NUMPTS,NINT
COMMON /COMP/ LCTNX(18)
C
DO 12 OLDEL=1,NPLUS1
10 CSTORE(OLDEL+1,NINT)=C(OLDEL)
OLDEL=LCTNLE
LCTNRE=MAXNUM
LCTNLE=MIN1(MAXNUM-LNGTH+1,(MAXNUM-OLDEL+1)/2)-1
20 LCTNLE=LCTNLE+1
   IF (MAXNUM-LCTNLE+1.LT.LNGTH) GO TO 40
   CALL SETP (CSTORE(1),NINT)/X(LCTNLE),NLSMH)
   CALL REMES (C,LCTNX,TOOBIG,ABORT)
   IF (ABORT), RETURN.
   IF (TOOBIG) GO TO 20
   CALL STORE (CSTORE(1),NINT),OLDEL,LCTNLE)
   NINT=NINT+1
   DO 30 OLDEL=1,NPLUS1
30   CSTORE(OLDEL+1,NINT)=C(OLDEL)
   RETURN.
40   ABORT=.TRUE.
   RETURN.
C
END

SUBROUTINE STORE (C,LCTNLE,LCTNRE)
C
THIS SUBROUTINE OUTPUTS THE COEFFICIENTS AND ENDPOINTS OF THE
CURRENT APPROXIMATION AND SUBINTERVAL. APPROPRIATE INFORMATION
IS STORED IN THE ARRAY CSTORE TO ALLOW THE ENTIRE PIECEWISE POLYO-
NOMIAL APPROXIMATION TO BE EASILY EVALUATED AT ANY POINT BY THE
FUNCTION EVAL.
C
DIMENSION C(18)
COMMON XTABLE(500),FTABLE(500),CSTORE(18,60),DUM1(500)
COMMON /SCALAR/ N,NPLUSI,NX,NXH1,NLSMH,NRSMH,NUMPTS,NINT
C
NUMPTS=LCTNRE-LCTNLE+1
WHITE (6+30) NINT*XTABLE(LCTNLE)*XTABLE(LCTNRE),NUMPTS
WHITE (6+30) (1+C(I)),I=1,N
ERR=0.0

DO 10 I=LCTNLE+LCTNRE
   TEMP=ABS(PITABLE(I)-HORNER(C,XTABLE(I)+N))
   IF (TEMP .GT. ERR) ERR=TEMP
10 CONTINUE
WRITE (6,50) ERR
DO 20 I=1,N
20 CSTORE(I,NINT)=C(I)
   CSTORE(NPLUS1+NINT)*XTABLE(LCTNLE)
RETURN
C
30 FORMAT (/54,5HINTERVAL NUMBER:14.4,16H WHICH BEGINS AT:23:16/1
   1,12H AND ENDS AT:23,15+2X,8HCOUNTS:14+8H POINTS,+/,9H TH
   2E COEF. 5)INFLUENCES OF BEST APPROXIMATION IN THIS INTERVAL ARE+/1
40 FORMAT (10X,2HCOUNTS:14+8H POINTS,+/,9H TH
   2E COEF. 5)INFLUENCES OF BEST APPROXIMATION IN THIS INTERVAL ARE+/1
50 FORMAT (/54,47H THE ERROR OF APPROXIMATION IN THIS INTERVAL IS:+2
   116+1H+)
C
END

SUBROUTINE DERIV (C,N)
C
THIS SUBROUTINE REPLACES THE COEFFICIENTS OF A POLYNOMIAL IN STANDARD
AND FORM WITH THE COEFFICIENTS OF THIS POLYNOMIAL'S DERIVATIVE.
C
THE NUMBER OF COEFFICIENTS, N, IS DECREMENTED.
C
DIMENSION C(N)
C
N=N-1
DO 10 I=1,N
10 C(I)=FLOAT(I)*C(I+1)
RETURN
C
END

SUBROUTINE SETP (C,X,SMTH)
C
THIS SUBROUTINE APPROPRIATELY STORES IN THE ARRAY PPRIME THE VALUES WHICH MUST BE INTERPOLATED TO GIVE THE PIECEWISE POLYNOMIAL THE
DESIRED SMOOTHNESS.
C
DIMENSION C(NB)
COMMON /COMP/ CSTORE(18)
COMMON /SCALAR/ NPLUS1,NX,NXM,NLSMTH,NRSMTH,NHPTS,NINT
COMMON /DOIFA/ PPRIME(3),DUM(36)
INTEGER SMTH
DO 10 I=1,N
10 CSTORE(I)=C(I)
NDUMMY=2
1#0
IF (I.GT.SMTH) RETURN
IF (I.EQ.0) GO TO 30
CALL DERIV (CSTORE,NDUMMY)
30 PPRIME(I+1)=HORNER(CSTORE*X,NDUMMY)
I=I+1
GO TO 20
C
END

FUNCTION CMPR(C,N,NEWTRY,OK)
C
THIS SUBROUTINE COMPARES THE FIRST DERIVATIVE OF THE CURRENT PIECE OF THE PIECEWISE POLYNOMIAL APPROXIMATION EVALUATED AT XTABLE(NEWTRY) WITH THE FIRST DERIVATIVE OF THE QUADRATIC INTERPOLATION OF F, CENTERED AROUND XTABLE(NEWTRY), EVALUATED AT XTABLE(NEWTRY). IF THESE TWO DIFFER IN ABSOLUTE VALUE BY LESS THAN TOLER (EITHER ABSOLUTELY OR RELATIVELY) WE SET OK TO .TRUE., AND WE ACCEPT XTABLE(NEWTRY) AS A REASONABLE SUBINTERVAL RIGHT ENDPOINT. NOTE THAT THE USER MAY EASILY CHANGE TOLER BY MEANS OF THE FOLLOWING DATA STATEMENT.

LOGICAL OK
COMMON XTABLE(500),FTABLE(500),SUM1(1582)
DIMENSION C(18)
DATA TOLER/.01/

OK=.FALSE.
A=(FTABLE(NEWTRY)-FTABLE(NEWTRY-1))/XTABLE(NEWTRY-1)-XTABLE(NEWTRY-1))
B=(FTABLE(NEWTRY+1)-FTABLE(NEWTRY))/XTABLE(NEWTRY+1)-XTABLE(NEWTRY-1))

OK=(A-B)/XTABLE(NEWTRY-1)-XTABLE(NEWTRY-1))
A=ABS(A)
B=ABS(B)
CMPR=ABS(A-B)

IF(CMPR.LE.TOLER)OK=.TRUE.
A=ABS(A)
IF (A.LT.0.01) RETURN
IF (CMPR.A.LT.TOLER)OK=.TRUE.

END

SUBROUTINE REMES (CL,LCNTX,TOOBIG,ABORT)

THIS IS THE DRIVING PROGRAM FOR THE COMPUTATION OF THE BEST RESTRICTED RANGE UNIFORM POLYNOMIAL APPROXIMATION TO F(X) (VALUES ARE STORED IN XTABLE AND FTABLE) OF DEGREE LESS THAN OR EQUAL TO N-1 ON THE SUBINTERVALS XTABLE(LCST1), XTABLE(LCST2),. SEE THE PAPER BY CHALMERS FOR ADDITIONAL INFORMATION ON THIS ALGORITHM.

DIMENSION XPTS(18), C(18), SGNX1(18), LCNTX(18), LCST1(18)
COMMON XTABLE(500),FTABLE(500),SUM1(1582),EARR(1582),FU(50)

INTEGER SCLARN,NPLUS1,NX,NSWTH,NNSWTH,NMETH,NINT
COMMON /DO4F/DUM1(23),DIB18
LOGICAL STOR,TOOBIG,ABORT

EPS IS A MACHINE CONSTANT--SET EPS TO (APPROXIMATELY) THE SMALLEST VALUE SUCH THAT EPS * 1.0 IS GREATER THAN 1.0.

DATA ITERM,EPS/30,1.0E-10/

FIRST WE INITIALIZE VARIOUS ARRAYS AND VALUES.

TOOBIG=.FALSE.
FIRSTM=LCNTX-1
SGNX1(I)=1.0
DO 10 IU=1,NX
10 SGNXI(I)=SGNX1(I-1)

EPSM=NUMER/2

REME 10
REME 20
REME 30
REME 40
REME 50
REME 60
REME 70
REME 80
REME 90
REME 100
REME 110
REME 120
REME 130
REME 140
REME 150
REME 160
REME 170
REME 180
REME 190
REME 200
REME 210
REME 220
REME 230
REME 240
REME 250
REME 260
REME 270
REME 280
REME 290
REME 300
REME 310
REME 320
REME 330
REME 340
REME 350
REME 360
REME 370
REME 380
REME 390
REME 400
REME 410
REME 420
REME 430
REME 440
REME 450
REME 460
REME 470
REME 480
REME 490
REME 500
REME 510
REME 520
REME 530
REME 540
REME 550
REME 560
REME 570
REME 580
REME 590
REME 600
REME 610
REME 620
REME 630
REME 640
REME 650
REME 660
REME 670
REME 680
REME 690
REME 700
REME 710
REME 720
REME 730
REME 740
REME 750
REME 760
REME 770
REME 780
REME 790
START=2
ERROR(1)=.0
IF (NLSTM+GE.+0) GO TO 20
NFTPS=NFTPS+1

START=1
20 IF (NRSTM+LT.+0) NFTPS=NFTPS+1
DELTA=FLOAT(NFTPS-1)/FLOAT(NX-1)
DO 30 J=1,NX
LCNTX(I)=START+I*FLOAT(I-1)*DELTA+5
JNFTPS=M+LCNTX(I)
XTPTS(I)=XTABLE(J)
D(I)=FTABLE(J)
30 CONTINUE

NOW WE BEGIN ITERATING. CONVERGENCE OCCURS WHEN TWO CONSECUTIVE
REFERENCE SETS (DETERMINED BY SOLVE) ARE THE SAME.

DO TO ITER*,ITEMNX
CALL DIVDIF (C,LCTNX,SGNXI,ABORT)
IF (ABORT) RETURN
DO 40 I=START,NUMRD
40 ERROR(I)=FTABLE(I+FRSTM1)-BPOLY(XTABLE(I+FRSTM1)+C+N)
IF (ABS(C(NPLUS1))+LE.PS) GO TO 60
DO 50 J=1,NX
IF (SGNX(I,I),NE.,.0) SGNX(I,I)=SIGN(1.0+ERROR(LCNTX(I))))
IF (SGNX(I,I),EQ.,.0) SGNX(I,I)=FLOAT(LCNTX(I))
IF (L.JE.QD) GO TO 50
IF (SGNX(I,I)+SGNX(I-1),GE.,.0) GO TO 80
50 CONTINUE
60 CALL EROBOU (LCNTX,LCTNX,SGNXI,ERROR)
CALL SOLVE (XTPTS,LCTNX,LCTNX,SGNXI,ABS(C(NPLUS1))+STOP+TOOBIG)
1)
IF (NOTSTOP) GO TO 70
IF (NOTTOOBIG) CALL TRANS (C+N)
RETURN
70 CONTINUE

WE PRINT OUT ERROR MESSAGES IF SOMETHING GOES WRONG.

WRITE (6+100) ITEMNX
GO TO 90
80 WRITE (6+110) ITER
90 ABORT#,TRUE.
RETURN

100 FORMAT (1H0,39(2H+1),1H*/*,2H0*11K+40,THE REMES ALGORITHM HAs NOT
1 CONVERGED IN*132*ITERATIONS*11X/1M*12,2HC*11X*3MPROGRAM ABO
2ATED IN SUBROUTINE REMES*+30*1HC*/*,1H0,39(2H+1),1H*)
110 FORMAT (1H0,39(2H+1),1H*/*,2H0*8X*12MIN ITERATION*13+7M OF REMES
1* NO ALTERATION OF SIGN OCCURS AT THE TEN POINTS*5,7,8,9,10,12,N,14,
2 POINTS*, THE PROGRAM ABORTED IN SUBROUTINE REMES*+12A*1H*/*,1H0*3
39(2H+1),1H*)

END

SUBROUTINE DIVDIF (C,LCTNX,SGNXI,ABORT)

THIS SUBROUTINE MAKES USE OF A DIVIDED DIFFERENCE SCHEME FOR SOLVING
THE VANUERMONDE-LIKE LINEAR SYSTEM INHERENT IN THE REMES ALGORITHM.
THE ADVANTAGES OF USING THIS SPECIAL PURPOSE LINEAR SYSTEM SOLVER
ARE--
THIS ROUTINE REQUIRES ON THE ORDER OF N**2 OPERATIONS AS COMPARED
TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER OF N**3 OPER-
ATIONS.

THIS ROUTINE REQUIRES ON THE ORDER OF N STORAGE LOCATIONS AS COMP-
ARED TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER OF N**2.

SEE THE FORTHCOMING PAPER BY J. A. HULL, S. F. MCCORMICK, AND G. D.
TAYLOR FOR A COMPLETE DESCRIPTION OF THIS ALGORITHM.

COMMON XTABLE(500),FTABLE(500),LCTNLE,LCTNRE,CTORE(18,60),ERROR(1
100)
COMMON /DITF/ FPRIME(5),X(18),D(18)
COMMON /SCALAR/ N,NPLUS(1,N),NLSMTH,NRSMTH,NURMTS,NINT
DIMENSION LCTN(18),C(18),SGRXI(18)
INTEGER FRTM1
LOGICAL AQUIT

FIRST WE INITIALIZE SEVERAL VARIABLES.

FRTM1=LCTNLE-1
IF (N,NPLUS(1)) SGRXI(NPLUS1)=0.0
IF (N,NPLUS(NX)) SGRXI(NPLUS1)=SGRXI(NX)

SET UP THE VECTOR X AND THE FIRST ROW OF THE DIVIDED DIFFERENCE TAB-
LE, USING THE COEFFICIENT VECTOR C FOR TEMPORARY STORAGE.

I=0
10 IF (I.GT:NLSMTH) GO TO 20
I=I+1
X(I)=XTABLE(LCTNLE)
C(I)=FPRIME(I)
GO TO 10
20 J=0
30 IF (J.GE.NX) GO TO 40
I=I+1
J=J+1
X(I)=XTABLE(FRTM1+LCTN(1))
C(I)=D(J)
GO TO 30
40 IF (I.GE.NPLUS1) GO TO 50
I=I+1
X(I)=XTABLE(LCTNRE)
C(I)=FPRIME(NLSMTH+2)
GO TO 40
50 CONTINUE

WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.

FAC=1.0
DO 100 J=2,N
JP1=J-1
JM1=J+1
FAC=FAC*FLOAT(JM1)
TEMP=C(JM1)
DO 94 TPJ=J
IF (X(I).NE.X(I-JM1)) GO TO 70
IF (J.GT:NLSMTH) GO TO 60
IF (J.GT:NPLUS1-NX) GO TO 220
TEMP1=FPRIME(JM1)/FAC
GO TO 94
94 CONTINUE
100 CONTINUE

BEST AVAILABLE COPY
60 IF (NL=SMTH.*JPL0.T, NPLUS1=NX) GO TO 220
   TEMP1=PRIME(NL,SMTH.*JPL0)/FACTOR
70   TEMP1=C(I(I)-C(I(I))/(X(I)-X(I-J))
80   C(I-1)=TEMP2
90   CONTINUE
100 CONTINUE

C**IF HAS NOW BEEN TEMPORARILY STORED IN THE COEFFICIENT ARRAY C.
WE NOW COMPUTE L**(-1)C = WE WILL COMPUTE THE DIVIDED DIFFERENCES
C IN THE TEMPORARY STORAGE ARRAY D. FIRST WE SET UP THE FIRST COLUMN
C OF THE DIVIDED DIFFERENCE TABLE.

I=0
110 IF (J.GT.NL,SMTH.) GO TO 120
   I=I+1
   D(I)=0.0
   GO TO 110
120 J=0
130 IF (J.GE.NX) GO TO 140
   I=I-1
   J=J+1
   D(I)=SGNRXI(J)
   GO TO 130
140 IF (J.GE.NX) GO TO 150
   I=I-1
   D(I)=0.0
   GO TO 140
150 CONTINUE

C WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.

DO 190 J=2,N
   TEMP2=D(J-1)
   DO 180 I=J,N
      IF (X(I).NE.X(I-J)) GO TO 160
      TEMP1=0.0
      GO TO 170
160 TEMP1=(D(I)-D(I-1))/(X(I)-X(I-J))
170 D(I-1)=TEMP2
   TEMP2=TEMP1
180 CONTINUE
   D(N)=TEMP2
190 CONTINUE

C WE NOW COMPUTE W=(F(N+1)-WTRANSPOSE*B))/Q*Q-WTRANSPOSE*B2)
C =C(N+1)=1 UNIFORM ERROR)
C FIRST WE COMPUTE THE TWO DOT PRODUCTS SIMULTANEOUSLY.

W=1.0
A=0.0
B=0.0
DO 200 I=1,N
   A=AI+C(I)*W
   B=B2+D(I)*W
   W=W*(NPLUS1)-X(I))
200 CONTINUE

C NOW WE COMPUTE M

C(NPLUS)=C(NPLUS1)-B1/(SGNRXI(NPLUS1)-B2)
C FINALLY, WE COMPUTE THE COEFFICIENTS C(I).

DO 210 I=1,N
   C(I)=C(I)-C(NPLUS1)*D(I)

210 CONTINUE
RETURN

220 ABORT=TRUE,
WRITE (6,230)
RETURN

C 230 FORMAT (1H0,39(2H1),1H*,/2H0,*,11X,5X,SHUTDOWN DIVIDF HAS FAILED
10-PROBABLY DUE TO AN INPUT;12X,1H*,*/2H0,*,11X,5X,ERROR (TO DIVIDF
2-) NOT ENOUGH ELEMENTS IN THE ARRAY PRIME;8X,1H*,*/2H0,*,11X,5X,ERROR
3-TWO IDENTICAL POINTS IN THE ARRAY X. PROGRAM ABORTED;10X,1H*,*/2
4H0,*,11X,21MIN SUBROUTINE DIVIDF, 45X,1H*,*/1H0,39(2H0,1,H*)
C END

SUBROUTINE ZEROFD (LCTNX,LCTNZ,SGNRX1,ERROR)
C THIS SUBROUTINE LOCATES THE (APPROXIMATE) ZEROS BETWEEN CONSECUTIVE
C POINTS OF THE CURRENT REFERENCE SET, THE LOCATIONS OF THESE POINTS
C IN THE ARRAY X; TABLE RELATIVE TO THE BEGINNING OF THE CURRENT SUB-
C INTERVAL ARE STORED IN LCTNZ.
C COMMON /SCALAR/ N,NPLUS1,NX,NSMTH,NRSMTH,NUMRD,NUMIT
DIMENSION LCTNX(18),LCTNZ(18),SGNRX1(18),ERROR(180)
C
   LCTNX(NX*)=NUMRD-1 NUMRD=1-NRSMTH
   CLNZ(NX*)=NUMIT-1 NUMIT=1-NRMTH
   DO 30 J=2,NX
      LCTNX(J)=LCTNX(J-1)
      LCTNZ(J)=LCTNZ(J-1)
      NUMIT=NUMIT-1
   10 CONTINUE
   IF ( ERROR(NRTHM1)*ERROR(NRTHM)) LE 6.0 GO TO 20
   20 CONTINUE
   RETURN
C END

SUBROUTINE SOLVE (X,LCTNX,LCTNZ,SGNRX1,E,STOP,TOOBIG)
C THIS SUBROUTINE PERFORMS THE MULTIPLE EXCHANGE OF THE REFERENCE
C SET REQUARED AT EACH ITERATION OF THE REMES ALGORITHM.
C COMMON XI(500),F(500),LCTNL,E,DUMMY(1003),ERROR(500),FU(50
10),FL(500)
COMMON /DIVIF/ DUM(23),D(18)
COMMON /SCALAR/ N,NPLUS1,NX,NSMTH,NRMTH,NUMIT
DIMENSION X(16),LCTNX(18),LCTNZ(18),SGNRX1(18)
LOGICAL STOP,TOOBIG
INTEGER FSTHM1,KTN
C EPS IS A MACHINE CONSTANT--SET EPS TO ROUGHLY THE SMALLEST NUMBER
C SUCH THAT 1.0+EPS.GT. 1.0.
C
DATA EPS=1.0E-9
C
STOP,TRUE,
FRSTM=LCLTNL=1
C
WE FIRST COMPUTE THE LOCATIONS OF THE NEW SET OF EXTREME POINTS,
STORING THEM IN THE VECTOR LCLTX. WE BEGIN BY CHOOSING AS THE 1ST
ELEMENT OF LCLTX THE LOCATION OF THE GRIDPOINT IN THE SUBINTERVAL
BETWEEN THE 1ST AND (1+1)ST ZERO WHICH RESULTS IN THE LARGEST ERROR
OF THE SAME SIGN AS THE PREVIOUS 1ST EXTREME POINT (HEREBY GUARAN-
TEED ALTERNATION). AT THE SAME TIME WE SEARCH FOR THE GRIDPOINT
WHICH RESULTS IN THE LARGEST (ABSOLUTE) ERROR, STORING ITS LOCATION
(IN LNBGST) AND THE NUMBER OF THE SUBINTERVAL IN WHICH IT OCCURS (IN
INBGST).
C
BIGER=-1.0E30
BIGEST=-1.0E30
DO 70 INTNUM=1,NX
BIG=-1.0E30
LFTND=LCNTZ(INTNUM)
RTHN=LCNTX(INTNUM+1)
SN=SNRXX(INTNUM)
DO 60 NEWLOC=LFTND,NEWLOC=END
SAME=SN=ERROR(NEWLOC)=E
UPPER=SN=ERROR(NEWLOC)=E
IF (SN=L.T.0.0) GO TO 10
SAME2=ERROR(NEWLOC)=FL(NEWLOC)
UPPER2=ERROR(NEWLOC)=FL(NEWLOC)
GO TO 20
10 SAME=ERROR(NEWLOC)=FU(NEWLOC)
UPPER=ERROR(NEWLOC)=FU(NEWLOC)
GO TO 20
20 IF (SAME+LE+BIG) GO TO 30
BIG=SAME
LCNTX(INTNUM)=NEWLOC
LCNTZ(INTNUM)=0
30 IF (SAME+LE+BIG) GO TO 40
BIG=SAME
LCNTX(INTNUM)=NEWLOC
LCNTZ(INTNUM)=IFIX(SGN)
40 IF (BIGER-LT-BIG) BIGER=BIG
IF (BIGEST+LT-BIG) BIGEST=BIG
IF (UPPER+LT-BIGEST) GO TO 50
BIGEST=UPPER
INBGST=INTNUM
LNBGST=NEWLOC
KIND=0
50 IF (UPPER+LE-BIGEST) GO TO 60
BIGEST=UPPER
INBGST=INTNUM
LNBGST=NEWLOC
KIND=IFIX(SGN)
60 CONTINUE
70 CONTINUE
IF (BIGEST+LT+EPS) RETURN
IF (ABS(BIGER-BIGEST)+LT+EPS) GO TO 120
C
AT THIS POINT IT IS STILL NECESSARY TO INSERT THE LOCATION OF THE
GRIDPOINT RESULTING IN THE LARGEST ERROR INTO LCLTX. THERE ARE THREE
CASES, EACH OF WHICH IS HANDLED SEPARATELY--ALTERNATION IS PRESERVED
AT THE GRIDPOINTS.
C
BIGEST=XTABLE(FRSTM+LNBGST)
YBIG = TABLE(LCTNX(INBOST) = FASTM1)
IF ((INBOST .EQ. 0) .AND. (YBIGST .LT. YBIG)) GO TO 80
IF ((INBOST .EQ. NX) .AND. (YBIGST .GT. YBIG)) GO TO 100
IF (YBIGST .LT. YBIG) NEWINT = INBOST - 1
IF (YBIGST .GT. YBIG) NEWINT = INBOST + 1
LCTNX(NEWINT) = INBOST
LCTNZ(NEWINT) = KIND
GO TO 120
80 DO 90 I = 1, NX
       JNX = I - 2
       LCTNX(J) = LCTNX(J-1)
       LCTNZ(J) = LCTNZ(J-1)
       SGNRXI(J) = SGNRXI(J-1)
90 CONTINUE
LCTNX(1) = INBOST
LCTNZ(1) = KIND
IF (KIND .EQ. 0) SGNRXI(1) = -SGNRXI(1)
GO TO 120
100 DO 110 J = 1, NX
       LCTNX(J) = LCTNX(J-1)
       LCTNZ(J) = LCTNZ(J-1)
       SGNRXI(J) = SGNRXI(J-1)
110 CONTINUE
LCTNX(NX) = INBOST
LCTNZ(NX) = KIND
IF (KIND .EQ. 0) SGNRXI(NX) = -SGNRXI(NX)
C NOW THAT LCTNX IS ACCEPTABLE, WE CHECK FOR CONVERGENCE OF THE ALGORITHM.
C IF THE EXTREME POINTS IF THE CONVERGENCE CRITERION IS NOT
C MET-
C
120 DO 130 I = 1, NX
       IF (LCTNZ(I) .EQ. 0) GO TO 140
130 CONTINUE
       TOOBIG = .TRUE.
       RETURN
140 DO 170 I = 1, NX
       NEWLOC = FASTM1 .AND. LCTNX(I)
       IF (LCTNZ(I) .EQ. 0) GO TO 150
       D(I) = TABLE(NEWLOC)
       GO TO 160
150 IF (LCTNZ(I) .EQ. -1) D(I) = TABLE(NEWLOC) + FL(NEWLOC)
       IF (LCTNZ(I) .EQ. 1) D(I) = TABLE(NEWLOC) - FL(NEWLOC)
       SGNRXI(I) = 0.0
160 TEMP = TABLE(NEWLOC)
       IF (ABS(TEMP) .LE. EPS) GO TO 170
       A(I) = TEMP
       STOP = .FALSE.
170 CONTINUE
       RETURN
C END

FUNCTION BPOLY(XX,C,N)
C THIS FUNCTION IS USED TO EVALUATE (BY THE APPROPRIATE ADAPTATION OF
C HORNER'S METHOD) THE POLYNOMIAL
C
C(1) + C(2)*(XX-X(1)) + C(3)*(XX-X(1))*(XX-X(2)) + ...
* C(N)*(XX-X(1))*(XX-X(2))* ... *(XX-X(N-1))
DIMENSION C(16)
COMMON /DQDF/ DUMMY(5),X(18),DUM(18)
C
NM1=N-1
BPOLY=C(N)
DO 10 I=1,NM1
J=J-1
BPOLY=C(J)*(X-X(J))*BPOLY
10 CONTINUE
RETURN
C
END

SUBROUTINE TRANS(C,N)
THIS SUBROUTINE TRANSFORMS A POLYNOMIAL WRITTEN IN THE FORM
C(1) + C(2)*(X-X(1)) + C(3)*(X-X(1))*(X-X(2)) + ... 
+ C(N)*(X-X(1))*(X-X(2))*...*(X-X(N-1))
TO A POLYNOMIAL WRITTEN IN TERMS OF POWERS OF X. THE X(I)'S ARE
SUPPLIED BY SUBROUTINE DIVDIF.
C
DIMENSION C(16)
COMMON /DQDF/ DUMMY(5),X(18),DUM(18)
C
NM1=N-1
DO 20 J=1,NM1
K=J
DO 10 I=K,NM1
C[I]=C[I]-X(K)*C[I+1]
10 CONTINUE
20 CONTINUE
RETURN
C
END

FUNCTION EVAL(X)
THIS FUNCTION EVALUATES THE PIECEWISE POLYNOMIAL APPROXIMATION AT
ANY POINT IN THE ENTIRE INTERVAL.
C
COMMON DUMMY(1002),CSTORE(18,60)+DUM(500)
COMMON /SCALAR/ N,NPLUS1,DUM2(5)+NINT
IF (NINT.LT.2) GO TO 20
DO 10 I=2,NINT
ISTORE=I-1
IF (ISTORE.LT.NPLUS1) GO TO 30
10 CONTINUE
ISTORE=NINT
GO TO 30
20 ISTORE=1
30 EVAL=HORNER(CSTORE(1),ISTORE),X,N)
RETURN
C
END

FUNCTION HORNER(C,X,N)

BPOLY 90
BPOLY 100
BPOLY 110
BPOLY 120
BPOLY 130
BPOLY 140
BPOLY 150
BPOLY 160
BPOLY 170
BPOLY 180
BPOLY 190
BPOLY 200
TRANS 10
TRANS 20
TRANS 30
TRANS 40
TRANS 50
TRANS 60
TRANS 70
TRANS 80
TRANS 90
TRANS 100
TRANS 110
TRANS 120
TRANS 130
TRANS 140
TRANS 150
TRANS 160
TRANS 170
TRANS 180
TRANS 190
TRANS 200
TRANS 210
TRANS 220
TRANS 230
EVAL 10
EVAL 20
EVAL 30
EVAL 40
EVAL 50
EVAL 60
EVAL 70
EVAL 80
EVAL 90
EVAL 100
EVAL 110
EVAL 120
EVAL 130
EVAL 140
EVAL 150
EVAL 160
EVAL 170
EVAL 180
EVAL 190
HORNER 10
HORNER 20
C THIS FUNCTION EVALUATES A POLYNOMIAL IN STANDARD FORM BY HORNER'S
C METHOD.
C
DIMENSION C(N)

HORNER=C(N)
I=N
10 IF (I.LT.2) RETURN
HORNER=HORNER*X*C(I-1)
I=I-1
GO TO 10

END
RESTRICTED RANGE ADAPTIVE CURVE FITTING PROGRAM: SAMPLE RUN
(ALGORITHMICALLY DEFINED RESTRRAINING CURVES)

INPUT:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>4.00</th>
<th>.175</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>3.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>5.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.0</td>
<td>6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.0</td>
<td>14.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.5</td>
<td>21.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.0</td>
<td>27.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.5</td>
<td>50.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.0</td>
<td>49.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.5</td>
<td>47.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.0</td>
<td>51.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35.0</td>
<td>36.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.0</td>
<td>27.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50.0</td>
<td>9.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60.0</td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(This denotes restraints option)
N, SMTH, MAXTOL, MINTOL
(XTABLE, FTABLE)

To employ the user defined retraining curves option, the first data card should be 0, and the upper and lower tolerances to be allowed at each data point should be on the respective data cards.

Ex:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>27.5</td>
<td>47.5</td>
<td>1.0</td>
<td>4.0</td>
<td></td>
</tr>
</tbody>
</table>

OUTPUT:

INTERVAL NUMBER 1 WHICH BEGINS AT .3000000000000000E+01 AND ENDS AT .1100000000000000E+02 CONTAINS 41 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

C(1) = .149756526697138E+02
C(2) = -.11445686135223?E+02
C(3) = .3049203763268548E+01
C(4) = -.334612213531450E+00
C(5) = .1592800176610654E+01
C(6) = -.2534785932326346E-03

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .3251703402472117E+00.

INTERVAL NUMBER 2 WHICH BEGINS AT .1100000000000000E+02 AND ENDS AT .1500000000000000E+02 CONTAINS 24 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

C(1) = .895392229386797E+03
C(2) = -.385378467685676E+03
C(3) = .663933580196042E+02
C(4) = -.5682948501274041E+01
C(5) = .240928490171192E+00
C(6) = -.4025042180238581E-02

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1207492591612606E+01.

BEST AVAILABLE COPY
INTERVAL NUMBER 3 WHICH BEGINS AT .1500000000000000E+02
AND ENDS AT .2000000000000000E+02 CONTAINS 29 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

\[
\begin{align*}
C(1) &= 0.2080819121698872E+05 \\
C(2) &= -0.6119991900236899E+04 \\
C(3) &= 0.7147141380302528E+03 \\
C(4) &= -0.4142494450662980E+02 \\
C(5) &= 0.119213427805883E+01 \\
C(6) &= -0.1362678361482650E-01
\end{align*}
\]

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2829293208918557E+01.

INTERVAL NUMBER 4 WHICH BEGINS AT .2000000000000000E+02
AND ENDS AT .235927070635915E+02 CONTAINS 28 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

\[
\begin{align*}
C(1) &= -0.26712586592871.86E+06 \\
C(2) &= 0.6180344279850274E+05 \\
C(3) &= -0.570453104277922E+04 \\
C(4) &= 0.262528492790677E+03 \\
C(5) &= -0.602261819792644E+01 \\
C(6) &= 0.5509132822059026E-01
\end{align*}
\]

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .339666798696013E+01.

INTERVAL NUMBER 5 WHICH BEGINS AT .235927070635915E+02
AND ENDS AT .273851841130498E+02 CONTAINS 21 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

\[
\begin{align*}
C(1) &= -0.101310427869369,5E+07 \\
C(2) &= 0.2002989828887461E+06 \\
C(3) &= -0.1582334382597386E+05 \\
C(4) &= 0.624371037695673E+03 \\
C(5) &= -0.1230580964377970E+02 \\
C(6) &= 0.9691424185808195E-01
\end{align*}
\]

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .7623136251345386E+00.

INTERVAL NUMBER 6 WHICH BEGINS AT .273851841130498E+02
AND ENDS AT .3343698511741786E+02 CONTAINS 37 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

\[
\begin{align*}
C(1) &= 0.2974023810520601E+06 \\
C(2) &= -0.473798232745143E+05 \\
C(3) &= 0.300844262538955E+04 \\
C(4) &= -0.9516814694657342E+02 \\
C(5) &= 0.1499809050599829E+01 \\
C(6) &= -0.9421877586332617E-02
\end{align*}
\]

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1270142945357293E+01.

BEST AVAILABLE COPY
INTERVAL NUMBER 7 WHICH BEGINS AT .3343698511741786E+02 AND ENDS AT .4085087038553866E+02 CONTAINS 39 POINTS.

THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

\[
\begin{align*}
C(1) &= .6567620303936116E+05 \\
C(2) &= -.6417321059978451E+04 \\
C(3) &= .4305390572534816E+03 \\
C(4) &= -.1097600576167463E+02 \\
C(5) &= .139435685419112E+00 \\
C(6) &= -.7061442077258562E-03 \\
\end{align*}
\]

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1270142944891631E+01.

INTERVAL NUMBER 8 WHICH BEGINS AT .4085087038553866E+02 AND ENDS AT .6000000000000000E+02 CONTAINS 95 POINTS.

THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

\[
\begin{align*}
C(1) &= .1984768817575497E+05 \\
C(2) &= -.208228167616931E+04 \\
C(3) &= .872391683570122E+02 \\
C(4) &= -.1819359280069307E+01 \\
C(5) &= .186185858999134E-01 \\
C(6) &= -.777217759386484E-04 \\
\end{align*}
\]

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .7764477640884024E+00.