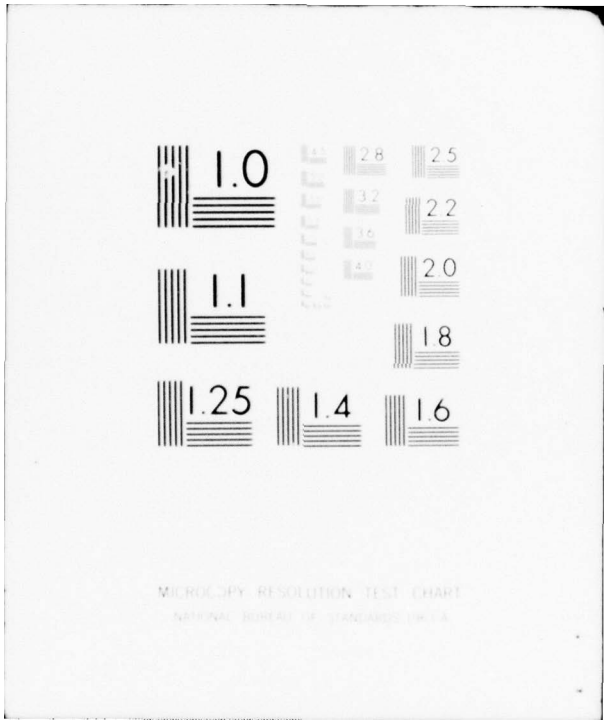
AD-A046 602 COLORADO STATE UNIV FORT COLLINS DEPT OF MATHEMATICS F/G 12/1
RESTRICTED RANGE ADAPTIVE CURVE FITTING.(U)
JUL 77 J A HULL, G D TAYLOR AFOSR-76-2878
UNCLASSIFIED AFOSR-TR-77-1258 NL

| OF |
AD
A046602

END
DATE
FILMED
12-77
DDC


1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR- 77 - 1258
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): RESTRICTED RANGE ADAPTIVE CURVE FITTING
5. TYPE OF REPORT & PERIOD COVERED: Interim rept.
6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): J. A. Hull and G. D. Taylor
8. CONTRACT OR GRANT NUMBER(s): AFOSR-76-2878

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Colorado State University
Department of Mathematics
Fort Collins, CO 80523

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A3

11. CONTROLLING OFFICE NAME AND ADDRESS:
AFOSR/NM
Bldg. 410
Bollins AFB, D.C. 20332

12. REPORT DATE: July 1977
13. NUMBER OF PAGES: 32

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): 35p.
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

DDC RECEIVED NOV 16 1977 F

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

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

407344

20. 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.

Send proofs to G. D. Taylor

[1]Research sponsored by the Air Force Office of Scientific Research, Air Force Systems, USAF, under grant no. 76-2878.

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 > 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 $\ell(x) \leq f(x) \leq u(x)$ 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 $p_i \in \Pi_{N-1} = \{q: q \text{ is a real algebraic polynomial of degree} \leq N-1\}$, p has SMTH continuous derivatives (on [a, b]) and for each $x \in X$, $\ell(x) \leq p(x) \leq u(x)$. By appropriately choosing $\ell(x)$ and u(x) then, the user obtains an approximation meeting a given preselected (set of) tolerance(s).

II. The Algorithm

The algorithm begins by choosing $\tilde{x}_1$ to be the largest point in X such that

1) $[a, \tilde{x}_1] \cap X$ contains at least N+1 points and
2) There is a best restricted range uniform approximation p_1 from Π_{N-1} to f (with respect to the constraining curves u(x) and $\ell(x)$) on $[a, \tilde{x}_1] \cap X$; that is, there exists $p_1 \in \Pi_{N-1}$ for which $\ell(x) \leq p_1(x) \leq u(x)$ holds for all $x \in [a, \tilde{x}_1] \cap X$ and $\|f - p_1\|_{[a,\tilde{x}_1]\cap X} = \inf\{\|f - q\|_{[a,\tilde{x}_1]\cap X}: q \in \Pi_{N-1}$ and $\ell(x) \leq q(x) \leq u(x)$ for all $x \in [a,\tilde{x}_1]\cap X\}$.

If $\tilde{x}_1 = b$, then since p_1 is a piecewise polynomial meeting our requirements, we successfully terminate the algorithm. If no such $\tilde{x}_1$ exists, the algorithm fails and an appropriate error message is generated. Otherwise, if SMTH = 0 (i.e. we only require the approximation to be continuous) we choose the right endpoint of the first subinterval, x_1, to be $\tilde{x}_1$. If SMTH > 0, in order to add to the stability of the algorithm we (in general) choose x_1 by "backing off" from $\tilde{x}_1$ in the following manner.

We first examine the error curve $f(x) - p_1(x)$ and find those points $\xi_1, \xi_2, \ldots, \xi_\ell$ in $(a, \tilde{x}_1] \cap X$ at which relative extrema occur. We will choose one of the ξ_ν'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 ξ 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 ξ would match that of f at ξ. This guarantees that when we smoothly join the next piece of the approximation to p_1 at ξ, this next piece will closely follow the direction of f at least near ξ. If we merely joined the second piece to the first at $\tilde{x}_1$, 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 $\tilde{x}_i$ to a smaller x_i contributes very significantly toward the stability of the algorithm. To continue, let $\tilde{f}'(\xi_\nu)$ be the derivative of the centered quadratic interpolation of f evaluated at ξ_ν. We then choose x_1 to be the largest ξ_ν such that $|\tilde{f}'(\xi_\nu) - p_1'(\xi_\nu)| < \text{EPS}$, where EPS is a tolerance which can be set by the user, or, if there does not exist such a ξ_ν, then we let x_1 be the largest ξ_ν at which $|f'(\xi_\nu) - p_1'(\xi_\nu)|$ 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 $[a, \tilde{x}_1] \cap X$, but only the largest N - SMTH - 1 of them.)

We continue by finding successive intervals $[x_1, x_2]$, $[x_2, x_3]$, ..., $[x_{m-1}, b]$ and corresponding polynomial approximations $p_2, p_3, \ldots, p_m \in \Pi_{N-1}$ to f so that $p_{\nu-1}^{(j)}(x_{\nu-1}) = p_\nu^{(j)}(x_{\nu-1})$ for $j = 0, 1, \ldots,$ SMTH and $\ell(x) \leq p_\nu(x) \leq u(x)$ for every $x \in [x_{\nu-1}, x_\nu] \cap X$ for $\nu = 2, \ldots, m$, $x_m = b$. This is accomplished as follows:

Suppose we have found the subintervals $[a, x_1]$, $[x_1, x_2]$, ..., $[x_{i-2}, x_{i-1}]$, 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-SMTH) points. We now determine an x_i and a p_i meeting the above requirements. We begin by choosing $\tilde{x}_i \in X$ to be the largest point in X which satisfies

1) $[x_{i-1}, \tilde{x}_i]$ contains at least max(2, N-SMTH) points and

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

If $\tilde{x}_i = b$, we set $x_i = \tilde{x}_i = b$, and the algorithm is successfully terminated. If no such $\tilde{x}_i$ exists, the algorithm fails and is terminated. Otherwise, we choose x_i completely analogous to our choice of x_1 above.

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

1) $[\hat{x}_{i-1}, b] \cap X$ contains at least max(2, N-SMTH) points and

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

Again, if we can find such an $\hat{x}_{i-1}$ then we change x_{i-1} to $\hat{x}_{i-1}$, set $x_i = b$, and successfully terminate the algorithm. If we cannot find such an $\hat{x}_{i-1}$, the algorithm is terminated and an appropriate error message is generated.

Remark. In our implementation of this algorithm, the $\tilde{x}_i$ are chosen as follows. At each step of this iterative procedure we will let $\tilde{a}$ be the current largest point in X such that requirements (1) and (2) are satisfied on $[x_{i-1}, \tilde{a}] \cap X$, and we will let $\tilde{b}$ be the current smallest point in X such that $\tilde{b} > \tilde{a}$ 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 $\tilde{x}_i = b$ and we are done. If the approximation fails to satisfy (2), we set $\tilde{b} = b$. Next, let $t = \min\{x \in X: [x_{i-1}, x] \cap X$ contains at least max(2, N-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 $\tilde{a} = t$.

In general, we proceed as follows. We let $t = \inf\{x \in X: (\tilde{b}-\tilde{a})/2 \leq x < \tilde{b}\}$. If this set is empty, we set $t = \sup\{x \in X: \tilde{a} \leq x \leq (\tilde{b} - \tilde{a})/2\}$. If $t = \tilde{a}$ then this procedure has converged and we set $\tilde{x}_i = t = \tilde{a}$. 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 $\tilde{a} = t$. If this approximation fails to satisfy (2) then we set $\tilde{b} = t$. We continue this process until $\tilde{b} - \tilde{a}$ is less than some user definable prescribed tolerance, at which point we accept $\tilde{a}$ as a good approximation to $\tilde{x}_i$ 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 $u(x) = f(x) + TOL$ and $\ell(x) = f(x) - TOL$ for $x \in X$, where 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 $\tilde{x}_i$ to x_i becomes more effective by having more densely packed points. Also, the subintervals $[x_{i-1}, \tilde{x}_i]$ may be chosen to be smaller (if needed in order to obtain a close enough approximation) while $[x_{i-1}, \tilde{x}_i] \cap X$ still contains at least max(2, N-SMTH) points. Only the original data points (indicated by "x") are shown in the following plots. In each case we chose N = 6, and SMTH = 2. The curves $\ell(x)$ and $u(x)$ 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.

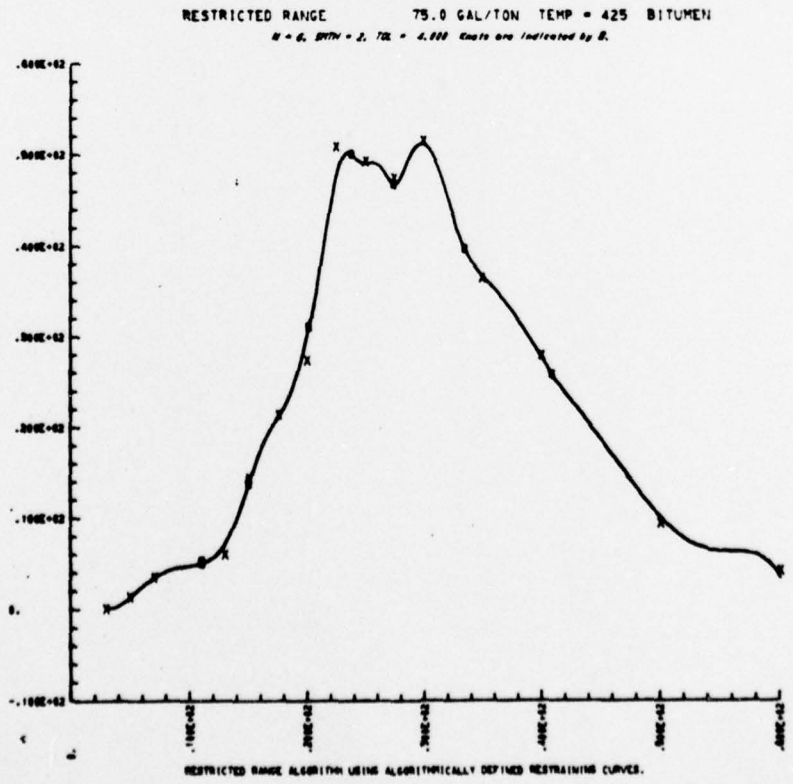


Figure 1

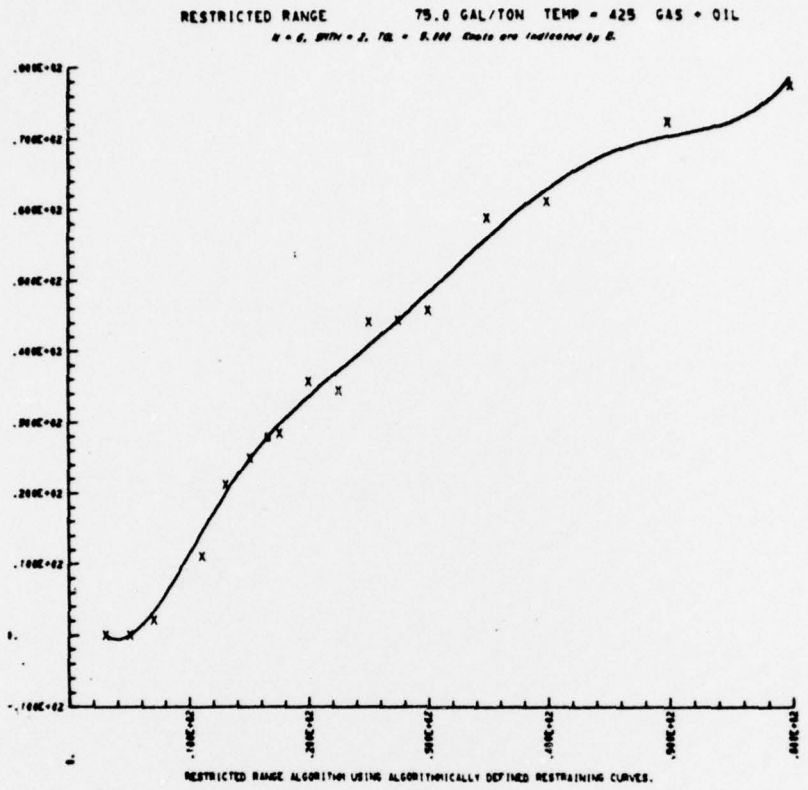


Figure 2

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.

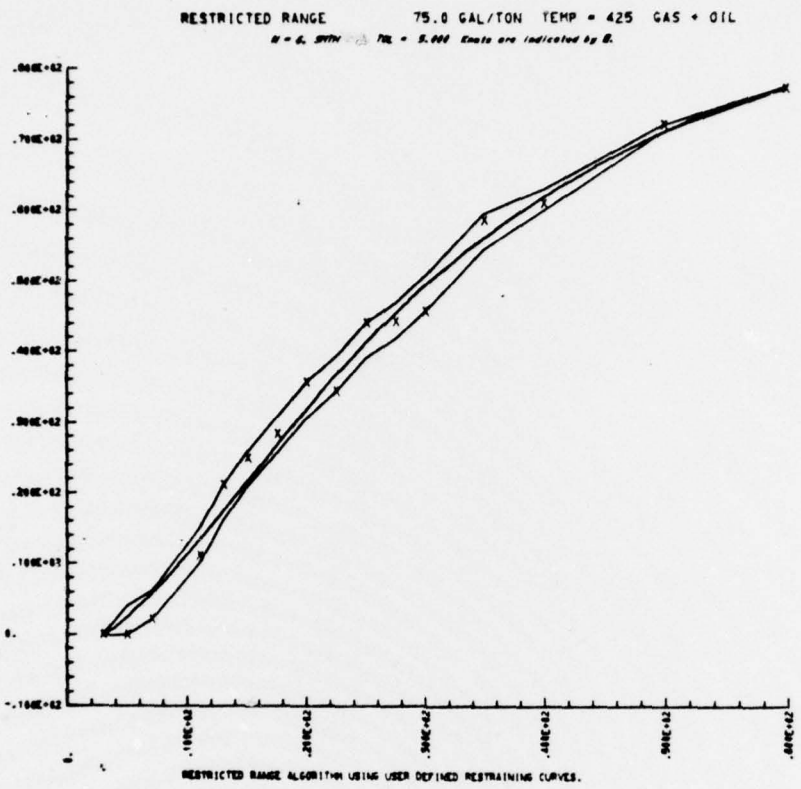


Figure 3

For clarity, we repeated the above plot but without the restraining curves.

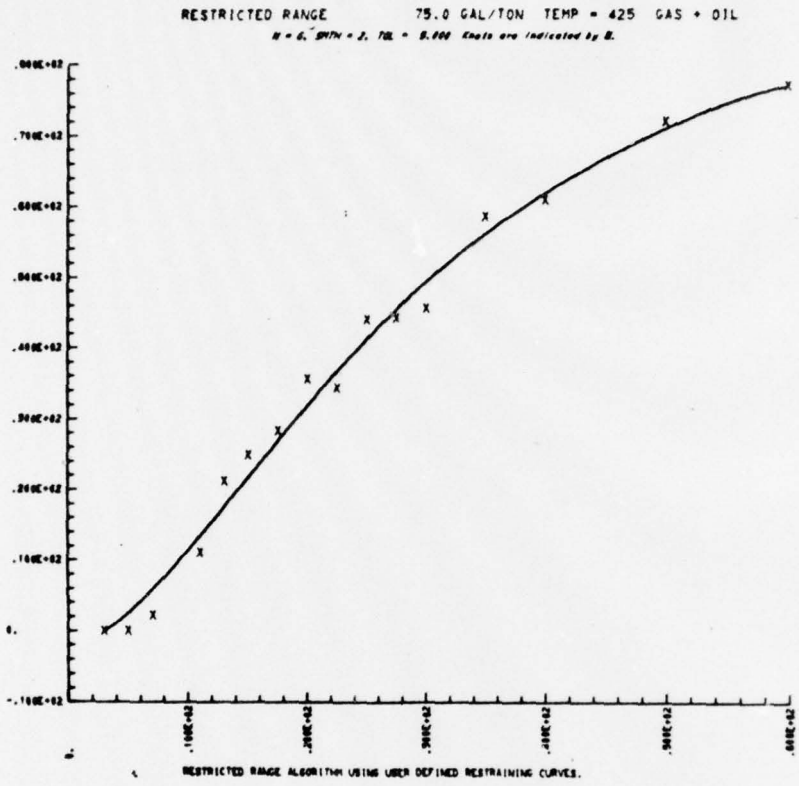


Figure 4

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

[1] M. Andrews, J.A. Hull and G. D. Taylor, Adaptive curve fittings for chemical processes, to appear.

[2] B.L. Chalmers, The Remez Excnage Algorithm for Approximation with Linear Restrictions, Trans. Amer. Math. Soc., 222(1976), 103-131.

[3] J.A. Hull and G.D. Taylor, Adaptive Curve Fitting, to appear.

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,TAPE5=INPUT,TAPE6=OUTPUT)                    DRIVER10
      LOGICAL ERROR                                                            DRIVER20
      COMMON XTABLE(500),FTABLE(500),DUMMY(1582),FU(500),FL(500)               DRIVER30
      INTEGER SMTH                                                             DRIVER40
      READ(5,50)IOPT                                                           DRIVER50
      READ (5,100) N,SMTH,TOL,SMLTOL                                           DRIVER60
      IF(IOPT.EQ.0)WRITE(6,200)                                                DRIVER70
      IF(IOPT.EQ.1)WRITE(6,300)                                                DRIVER80
      WRITE(6,400)N,SMTH,TOL                                                   DRIVER90
      MAXNUM=0                                                                 DRIVE100
   10 MAXNUM=MAXNUM+1                                                          DRIVE110
      READ (5,150) XTABLE(MAXNUM),FTABLE(MAXNUM),FU(MAXNUM),FL(MAXNUM)         DRIVE120
      IF (EOF(5).EQ.0.0) GO TO 10                                              DRIVE130
      MAXNUM=MAXNUM-1                                                          DRIVE140
      IF(IOPT.EQ.1)CALL SETRNG(MAXNUM,SMLTOL,TOL)                              DRIVE150
      CALL LINEAR (MAXNUM,300,MAXNUM,1.125)                                    DRIVE160
      CALL RRACF (N,SMTH,MAXNUM,ERROR)                                         DRIVE170
   50 FORMAT(I1)                                                               DRIVE180
  100 FORMAT(2I5,2F10.5)                                                       DRIVE190
  150 FORMAT (4F10.5)                                                          DRIVE200
  200 FORMAT(1H1,5X,60HRESTRICTED RANGE ALGORITHM--USER DEFINED RESTRAIN       DRIVE210
     $ING CURVES.)                                                             DRIVE220
  300 FORMAT(1H1,5X,71HRESTRICTED RANGE ALGORITHM--ALGORITHMICALLY DEFIN       DRIVE230
     $ED RESTRAINING CURVES.)                                                  DRIVE240
  400 FORMAT(1H0,5X,17HNUMBER OF COEFFS=,I2,1H,,I2,25H CONTINUOUS DERIVS       DRIVE250
     $, TOL= ,F7.5)                                                            DRIVE260
      END                                                                      DRIVE270


      SUBROUTINE LINEAR (OLDMAX,NAPROX,NTRUE,R)                                LINEAR10
C                                                                              LINEAR20
C  THIS SUBROUTINE FILLS IN BETWEEN THE ORIGINAL DATA POINTS WITH              LINEAR30
C  ABOUT NAPROX LINEARLY INTERPOLATED DATA POINTS.  THE SPACING USED           LINEAR40
C  DEPENDS UPON THE DATA.  FOR *SMOOTH* DATA, I.E., DATA SUCH THAT CON-        LINEAR50
C  SECUTIVE LINE SEGMENTS JOINING THE DATA POINTS DIFFER ONLY SLIGHTLY         LINEAR60
C  IN SLOPE, EQUISPACED POINTS ARE USED.  FOR MORE COMPLEX DATA, THE           LINEAR70
C  SPACING IS SUCH THAT MORE POINTS ARE CONCENTRATED IN AREAS WHERE            LINEAR80
C  THE FUNCTION BEING APPROXIMATED IS COMPLICATED AND FEWER WHERE IT           LINEAR90
C  IS SIMPLE.  THE MAXIMUM SPACING BETWEEN POINTS IS DELTA*R, WHERE            LINEA100
C  DELTA IS THE SPACING WHICH WOULD RESULT IF WE USED NAPROX EQUI-             LINEA110
C  SPACED POINTS, AND R IS A CONSTANT CHOSEN BY THE USER WHICH IS              LINEA120
C  GREATER THAN OR EQUAL TO 1.0.  (NOTE WHEN R IS SET TO 1.0, EQUI-            LINEA130
C  SPACING OF POINTS OCCURS REGARDLESS OF THE NATURE OF THE DATA.)             LINEA140
C  AS THE ORIGINAL DATA SET IS INCLUDED AS A SUBSET OF THE NEW DATA            LINEA150
C  SET, THE NUMBER OF POINTS IN THE NEW DATA SET IS BETWEEN NAPROX             LINEA160
C  AND NAPROX+OLDMAX.  THIS TRUE VALUE IS RETURNED TO THE USER IN              LINEA170
C  NTRUE.                                                                      LINEA180
C                                                                              LINEA190
      LOGICAL EQSPCD                                                           LINEA200
      REAL MINVAL,M                                                            LINEA210
      INTEGER OLDMAX,OLDM1                                                     LINEA220
      COMMON XTABLE(500),FTABLE(500),XSTR(316),FSTR(316),FUSTR(316),FLST       LINEA230
     1R(317),QSUB(317),FU(500),FL(500)                                         LINEA240
      Q(I)=((FSTR(I+1)-FSTR(I))/(XSTR(I+1)-XSTR(I))-(FSTR(I)-FSTR(I-1))/       LINEA250
     1(XSTR(I)-XSTR(I-1)))                                                     LINEA260
      AREA(I)=.5*(QSUB(I+1)+QSUB(I))*(XSTR(I+1)-XSTR(I))                       LINEA270
      DATA EPS,OMEGA/1.0E-8,1.0E-5/                                            LINEA280
      DO 10 I=1,OLDMAX                                                         LINEA290
         XSTR(I)=XTABLE(I)                                                     LINEA300
         FSTR(I)=FTABLE(I)                                                     LINEA310
         FUSTR(I)=FU(I)                                                        LINEA320
         FLSTR(I)=FL(I)                                                        LINEA330
```

```
10 CONTINUE                                                              LINEA340
   EQSPCD=.TRUE.                                                         LINEA350
   IF(R.EQ.1.0)GO TO 55                                                  LINEA360
   OLDM1=OLDMAX-1                                                        LINEA370
   SML=ABS(Q(2))                                                         LINEA380
   DO 20 I=2,OLDM1                                                       LINEA390
      TEMP=ABS(Q(I))                                                     LINEA400
      QSUB(I)=TEMP                                                       LINEA410
      IF (TEMP.LT.SML) SML=TEMP                                          LINEA420
20 CONTINUE                                                              LINEA430
   QSUB(1)=0.0                                                           LINEA440
   QSUB(OLDMAX)=0.0                                                      LINEA450
   DO 30 I=2,OLDM1                                                       LINEA460
30 QSUB(I)=QSUB(I)-SML                                                   LINEA470
   MINIVL=1                                                              LINEA480
   MINVAL=QSUB(2)                                                        LINEA490
   TOTAR=MINVAL*.5*(XSTR(2)-XSTR(1))                                     LINEA500
   DO 40 I=2,OLDM1                                                       LINEA510
      TEMP=QSUB(I)+QSUB(I+1)                                             LINEA520
      TOTAR=TOTAR+.5*TEMP*(XSTR(I+1)-XSTR(I))                            LINEA530
      IF (TEMP.GE.MINVAL) GO TO 40                                       LINEA540
      MINVAL=TEMP                                                        LINEA550
      MINIVL=I                                                           LINEA560
40 CONTINUE                                                              LINEA570
   DELTA=(XSTR(OLDMAX)-XSTR(1))/FLOAT(NAPROX-1)                          LINEA580
   DLTMAX=DELTA*R                                                        LINEA590
   IF(TOTAR.LT.OMEGA)GO TO 55                                            LINEA600
   EQSPCD=.FALSE.                                                        LINEA610
   M=ABS(QSUB(MINIVL+1)-QSUB(MINIVL))/(XSTR(MINIVL+1)-XSTR(MINIVL))      LINEA620
   H=(TOTAR/FLOAT(NAPROX-1)-.5*M*DLTMAX*DLTMAX)/(DLTMAX-DELTA)           LINEA630
   IF (H.LT.0.0) H=0.0                                                   LINEA640
   DO 50 I=1,OLDMAX                                                      LINEA650
      QSUB(I)=QSUB(I)+H                                                  LINEA660
50 CONTINUE                                                              LINEA670
   C=(TOTAR+H*(XSTR(OLDMAX)-XSTR(1)))/FLOAT(NAPROX-1)                    LINEA680
55 K=0                                                                   LINEA690
   I=0                                                                   LINEA700
   XX=XSTR(1)                                                            LINEA710
60 I=I+1                                                                 LINEA720
   IF (XX.LT.XSTR(K+1)) GO TO 70                                         LINEA730
   K=K+1                                                                 LINEA740
   XX=XSTR(K)                                                            LINEA750
   XTABLE(I)=XX                                                          LINEA760
   FTABLE(I)=FSTR(K)                                                     LINEA770
   FU(I)=FUSTR(K)                                                        LINEA780
   FL(I)=FLSTR(K)                                                        LINEA790
   IF (K.EQ.OLDMAX) GO TO 110                                            LINEA800
   DX=XSTR(K+1)-XSTR(K)                                                  LINEA810
   SLF=(FSTR(K+1)-FSTR(K))/DX                                            LINEA820
   SLFU=(FUSTR(K+1)-FUSTR(K))/DX                                         LINEA830
   SLFL=(FLSTR(K+1)-FLSTR(K))/DX                                         LINEA840
   M=(QSUB(K+1)-QSUB(K))/DX                                              LINEA850
   IF(.NOT.EQSPCD)GO TO 80                                               LINEA860
   XX=XX+DELTA                                                           LINEA870
   GO TO 60                                                              LINEA880
70 XTABLE(I)=XX                                                          LINEA890
   DIFF=XX-XSTR(K)                                                       LINEA900
   FTABLE(I)=FSTR(K)+DIFF*SLF                                            LINEA910
   FU(I)=FUSTR(K)+DIFF*SLFU                                              LINEA920
   FL(I)=FLSTR(K)+DIFF*SLFL                                              LINEA930
   IF(.NOT.EQSPCD)GO TO 80                                               LINEA940
   XX=XX+DELTA                                                           LINEA950
```

```
      GO TO 60                                                          LINEA960
   80 QOFXX=M*(XX-XSTR(K))+QSUB(K)                                      LINEA970
      IF (ABS(M).LT.EPS) GO TO 90                                       LINEA980
      TEMP=QOFXX**2+2.0*M*C                                             LINEA990
      IF (TEMP.LT.0.0) GO TO 100                                        LINE1000
      XX=XX+(-QOFXX+SQRT(TEMP))/M                                       LINE1010
      GO TO 60                                                          LINE1020
   90 XX=XX+C/QOFXX                                                     LINE1030
      GO TO 60                                                          LINE1040
  100 XX=XSTR(K+1)                                                      LINE1050
      GO TO 60                                                          LINE1060
  110 NTRUE=I                                                           LINE1070
      RETURN                                                            LINE1080
C                                                                       LINE1090
      END                                                               LINE1100

      SUBROUTINE SETRNG (MAXNUM,MINTOL,MAXTOL)                          SETRNG10
C                                                                       SETRNG20
C  THIS SUBROUTINE USES CONVEX COMBINATIONS OF MINTOL AND MAXTOL TO SET  SETRNG30
C  THE FU AND FL TOLERANCES AT EACH DATA POINT DEPENDING ON THE COMP-    SETRNG40
C  LEXITY OF THE FUNCTION BEING APPROXIMATED.  THAT IS, WHERE THE FUN-   SETRNG50
C  CTION IS SMOOTHEST, THE TOLERANCES WILL BE CLOSE TO (BUT AT LEAST AS  SETRNG60
C  BIG ON AT LEAST ONE SIDE) AS MINTOL, AND WHERE THE APPROXIMATION IS   SETRNG70
C  COMPLICATED THE APPROXIMATION WILL BE CLOSE TO (BUT NO BIGGER THAN)   SETRNG80
C  MAXTOL ON AT LEAST ONE SIDE.  FOR DATA WHICH IS #SMOOTH#, AS DE-      SETRNG90
C  SCRIBED IN SUBROUTINE LINEAR, THE TOLERANCES BECOME SOMEWHAT LESS     SETRN100
C  SIGNIFICANT, AND ARE SET TO MAXTOL AT ALL DATA POINTS.  THIS          SETRN110
C  ROUTINE FREES THE USER FROM HAVING TO CHOOSE AN INITIAL BAND OF       SETRN120
C  TOLERANCES--IT IS NOT NECESSARILY INTENDED TO BE A TOTALLY AUTO-      SETRN130
C  MATIC TOLERANCE BAND SELECTOR FOR ANY ARBITRARY FUNCTION.  EXPER-     SETRN140
C  IMENTING WITH USER SUPPLIED TOLERANCES MUST OFTEN RESULTS IN MORE     SETRN150
C  DESIRABLE FITS.                                                       SETRN160
C                                                                       SETRN170
      REAL MINTOL,MAXTOL                                                SETRN180
      COMMON XTABLE(500),FTABLE(1582),QSTR(500),FU(500),FL(500)         SETRN190
      DATA OMEGA/1.0E-5/                                                SETRN200
      Q(I)=((FTABLE(I+1)-FTABLE(I))/(XTABLE(I+1)-XTABLE(I))-(FTABLE(I)-F SETRN210
     1TABLE(I-1))/(XTABLE(I)-XTABLE(I-1)))*(XTABLE(I+1)-XTABLE(I-1))    SETRN220
      MAXM1=MAXNUM-1                                                    SETRN230
      QAVE=Q(2)                                                         SETRN240
      QSTR(2)=QAVE                                                      SETRN250
      BIG=ABS(QAVE)                                                     SETRN260
      SML=BIG                                                           SETRN270
      DO 10 I=3,MAXM1                                                   SETRN280
         TEMP=Q(I)                                                      SETRN290
         QSTR(I)=TEMP                                                   SETRN300
         QAVE=QAVE+TEMP                                                 SETRN310
         TEMP=ABS(TEMP)                                                 SETRN320
         IF (TEMP.GT.BIG) BIG=TEMP                                      SETRN330
         IF (TEMP.LT.SML) SML=TEMP                                      SETRN340
   10 CONTINUE                                                          SETRN350
      DIFF=BIG-SML                                                      SETRN360
      IF(DIFF.LT.OMEGA)GO TO 40                                         SETRN370
      QAVE=QAVE/FLOAT(MAXNUM-2)                                         SETRN380
      DO 30 I=2,MAXM1                                                   SETRN390
         TEMP=ABS(QSTR(I))                                              SETRN400
         TOLI=MINTOL*(BIG-TEMP)/DIFF+MAXTOL*(TEMP-SML)/DIFF             SETRN410
         TSUB=TOLI*(BIG-TEMP)/DIFF                                      SETRN420
         IF (QSTR(I).LT.0.0) GO TO 20                                   SETRN430
         FU(I)=TOLI                                                     SETRN440
         FL(I)=TSUB                                                     SETRN450
```

```
      GO TO 30                                                          SETRN460
20    FU(I)=TSUB                                                        SETRN470
      FL(I)=TOLI                                                        SETRN480
30 CONTINUE                                                             SETRN490
   FU(1)=FU(2)                                                          SETRN500
   FL(1)=FL(2)                                                          SETRN510
   FU(MAXNUM)=FU(MAXM1)                                                 SETRN520
   FL(MAXNUM)=FL(MAXM1)                                                 SETRN530
                                                                        SETRN540
                                                                        SETRN550
   RETURN                                                               SETRN560
40 DO 50 I=1,MAXNUM                                                     SETRN570
      FU(I)=MAXTOL                                                      SETRN580
      FL(I)=MAXTOL                                                      SETRN590
50 CONTINUE                                                             SETRN600
   RETURN                                                               SETRN610
C                                                                       SETRN620
   END                                                                  SETRN630

   SUBROUTINE RRACF (N,SMTH,MAXNUM,ERROR)                               RRACF 10
C                                                                       RRACF 20
C  THIS SUBROUTINE ADAPTIVELY COMPUTES A PIECEWISE POLYNOMIAL APPROXI-  RRACF 30
C  MATION OF DEGREE N-1 TO THE FUNCTION STORED IN THE ARRAYS XTABLE AND  RRACF 40
C  FTABLE WITH SMTH CONTINUOUS DERIVATIVES HAVING THE PROPERTY THAT FOR  RRACF 50
C  1 .LE. I .LE. MAXNUM (SEE BELOW) THE VALUE OF THE APPROXIMATION      RRACF 60
C  EVALUATED AT XTABLE(I) LIES BETWEEN (FTABLE(I) - FL(I)) AND          RRACF 70
C  (FTABLE(I) + FU(I)), WHERE THE ARRAYS FL AND FU CONTAIN THE DESIRED  RRACF 80
C  TOLERANCE REQUIRED OF THE APPROXIMATION BELOW THE CURVE AND ABOVE THE RRACF 90
C  CURVE (RESPECTIVELY) AT EACH OF THE (MAXNUM) POINTS BEING APPROXI-   RRACF100
C  MATED.                                                               RRACF110
C                                                                       RRACF120
C  THE PARAMETERS ARE AS FOLLOWS--                                      RRACF130
C                                                                       RRACF140
C     N - THE NUMBER OF COEFFICIENTS OF EACH POLYNOMIAL PIECE.  I.E.    RRACF150
C        ONE MORE THAN THE DEGREE OF THE PIECEWISE POLYNOMIAL APPROX.   RRACF160
C        AS THE ARRAYS ARE CURRENTLY DIMENSIONED, IT IS ASSUMED THAT    RRACF170
C        N IS NO BIGGER THAN 17.                                        RRACF180
C                                                                       RRACF190
C     SMTH - THE NUMBER OF CONTINUOUS DERIVATIVES DESIRED OF THE        RRACF200
C        APPROXIMATION.  SMTH MUST NOT BE GREATER THAN N-2.  IF ONLY    RRACF210
C        CONTINUITY OF THE APPROXIMATION IS REQUIRED, SET SMTH = 0.     RRACF220
C                                                                       RRACF230
C     MAXNUM - THE NUMBER OF POINTS ACTUALLY STORED IN THE ARRAYS       RRACF240
C        XTABLE, FTABLE, FU, AND FL.  (AS THESE ARRAYS ARE CURRENTLY    RRACF250
C        DIMENSIONED, MAXNUM MUST BE LESS THAN 500.  IF ONE WISHES TO   RRACF260
C        COMPUTE APPROXIMATIONS TO FUNCTIONS WITH MORE THAN 500         RRACF270
C        POINTS, ONE CAN EASILY MODIFY THIS PROGRAM TO CONTINUOUSLY     RRACF280
C        READ IN MORE POINTS AFTER ROOM IS MADE IN THESE STORAGE        RRACF290
C        ARRAYS BY HAVING COMPLETED SEVERAL SUBINTERVALS.)              RRACF300
C                                                                       RRACF310
C     ERROR - A LOGICAL VALUE SET TO .TRUE. IF AN ERROR OCCURS IN THE   RRACF320
C        PROGRAM (AN APPROPRIATE ERROR MESSAGE WILL ALSO BE PRINTED)    RRACF330
C        AND SET TO .FALSE. OTHERWISE.                                  RRACF340
C                                                                       RRACF350
C     BLANK COMMON PROVIDES THE REMAINDER OF THE INPUT.  IT IS ASSUMED  RRACF360
C     THAT THE FIRST 500 WORDS OF BLANK COMMON CONTAIN THE TABLE OF X   RRACF370
C     VALUES (XTABLE), THE NEXT 500 WORDS THE TABLE OF FUNCTION VALUES  RRACF380
C     (FTABLE), THE NEXT TWO WORDS ARE USED INTERNALLY THROUGHOUT THE   RRACF390
C     THE PROGRAM, AND THE NEXT 1080 WORDS IS AN ARRAY (CSTORE(18,60))  RRACF400
C     CONTAINING THE COMPUTED COEFFICIENTS AND THE KNOTS--I.E.          RRACF410
C     CSTORE(I,INT) IS THE COEFFICIENT OF THE I-1ST DEGREE TERM IN SUB-  RRACF420
```

```
C      INTERVAL NUMBER INT.  CSTORE(N+1,INT) IS THE LEFT END POINT OF          RRACF430
C      SUBINTERVAL INT.  THE NEXT ARRAY IS USED INTERNALLY TO STORE THE        RRACF440
C      ERRORS OF APPROXIMATION OCCURING AT EACH DATA POINT DURING THE          RRACF450
C      THE REMES ALGORITHM AND IT IS ALSO USED AS A SCRATCH STORAGE AR-        RRACF460
C      RAY THROUGHOUT THE PROGRAM.  THE LAST TWO ARRAYS IN BLANK COMMON        RRACF470
C      (FU AND FL) ARE THE ARRAYS IN WHICH THE USER SETS THE TOLERANCE         RRACF480
C      HE REQUIRES ABOVE AND BELOW THE CURVE AT EACH OF THE MAXNUM             RRACF490
C      POINTS AS DESCRIBED ABOVE.  FU AND FL ARE EACH DIMENSIONED 500          RRACF500
C      WORDS LONG.                                                             RRACF510
C                                                                              RRACF520
C  THE COEFFICIENTS AND SUBINTERVAL ENDPOINTS ARE PRINTED OUT AS THEY          RRACF530
C  ARE COMPUTED.  ALSO THE FUNCTION EVAL IS AVAILABLE TO THE USER TO           RRACF540
C  EVALUATE THE APPROXIMATION AT ANY POINT WITHIN THE ENTIRE INTERVAL.         RRACF550
C                                                                              RRACF560
      LOGICAL LAST,ERROR,DONE,ABORT                                            RRACF570
      INTEGER SMTH                                                             RRACF580
      DIMENSION C(18)                                                          RRACF590
      COMMON XTABLE(1000),LCTNLE,LCTNRE,DUM1(2580)                             RRACF600
      COMMON /SCALAR/ NPLUS0,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT          RRACF610
C                                                                              RRACF620
      NRSMTH=-1                                                                RRACF630
      ERROR=.FALSE.                                                            RRACF640
      DONE=.FALSE.                                                             RRACF650
      ABORT=.FALSE.                                                            RRACF660
      NPLUS0=N                                                                 RRACF670
      NPLUS1=N+1                                                               RRACF680
C                                                                              RRACF690
C  IN THE FIRST SUBINTERVAL THERE ARE NO INTERPOLATORY CONSTRAINTS--           RRACF700
C  CONSEQUENTLY, WE SET NLSMTH=-1.                                             RRACF710
C                                                                              RRACF720
      NLSMTH=-1                                                                RRACF730
      NX=NPLUS1-2-NLSMTH-NRSMTH                                                RRACF740
      NXM1=NX-1                                                                RRACF750
      LNGTH=NX+1                                                               RRACF760
C                                                                              RRACF770
C  WE INITIALLY TRY AS MUCH OF THE CURRENT REMAINING PORTION OF THE            RRACF780
C  WHOLE INTERVAL AS POSSIBLE AS AN INITIAL GUESS FOR EACH SUCCESSIVE          RRACF790
C  SUBINTERVAL.  LCTNLE IS THE LOCATION (IN THE ARRAY XTABLE) OF THE           RRACF800
C  LEFT END POINT OF THE CURRENT SUBINTERVAL, LCTNRE IS THE LOCATION OF        RRACF810
C  THE RIGHT END POINT.                                                        RRACF820
C                                                                              RRACF830
      LCTNLE=1                                                                 RRACF840
      LCTNRE=MAXNUM                                                            RRACF850
      DO 20 INTNUM=1,60                                                        RRACF860
         NINT=INTNUM                                                           RRACF870
C                                                                              RRACF880
C  SUBROUTINE COMPUT FINDS THE LARGEST SUBINTERVAL OF (XTABLE(LCTNLE),         RRACF890
C  XTABLE(LCTNRE)) WITH LEFT END POINT XTABLE(LCTNLE) SUCH THAT THE BEST       RRACF900
C  APPROXIMATION ON THIS SUBINTERVAL SATISFIES ALL THE CONSTRAINTS.  THE       RRACF910
C  RIGHT END POINT IS *BACKED OFF* TO THE LAST INTERIOR EXTREME POINT          RRACF920
C  OF F-P TO ADD TO THE STABILITY OF THE ALGORITHM.  THE LOCATION OF           RRACF930
C  THIS RIGHT END POINT IS STORED IN LCTNRE.  IF LCTNRE=MAXNUM (I.E.           RRACF940
C  WE ARE DONE), CONTROL IS PASSED TO LINE 40.  IF NO SUCH SUBINTERVAL         RRACF950
C  CAN BE FOUND, CONTROL IS PASSED TO LINE 50.  IF THERE ARE FEWER THAN        RRACF960
C  LNGTH POINTS FROM LCTNRE TO MAXNUM, LAST IS SET TO .TRUE. AND THE           RRACF970
C  SPECIAL CASE SUBROUTINE LSTINT IS CALLED.                                   RRACF980
C                                                                              RRACF990
         CALL COMPUT (C,LNGTH,MAXNUM,LAST,DONE,ABORT)                          RRAC1000
         IF (ABORT) GO TO 50                                                   RRAC1010
         IF (DONE) GO TO 40                                                    RRAC1020
         IF (INTNUM.GT.1) GO TO 10                                             RRAC1030
         NLSMTH=SMTH                                                           RRAC1040
```

```
      NX=NPLUS1-2-NLSMTH-NRSMTH                                          RRAC1050
      NXM1=NX-1                                                          RRAC1060
      LNGTH=NX+1                                                         RRAC1070
10    IF (LAST) GO TO 30                                                 RRAC1080
C                                                                        RRAC1090
C  SUBROUTINE STORE STORES THE COEFFICIENTS FOR THIS SUBINTERVAL IN THE   RRAC1100
C  ARRAY CSTORE.  IT ALSO PRINTS OUT THE COEFFICIENTS AND THE ERROR OF    RRAC1110
C  APPROXIMATION ON THIS SUBINTERVAL.  SUBROUTINE SETP(C,X,K) STORES THE  RRAC1120
C  VALUE OF THE POLYNOMIAL DETERMINED BY THE COEFFICIENTS IN THE ARRAY    RRAC1130
C  C AND ITS FIRST K DERIVATIVES AT THE POINT X IN THE ARRAY PPRIME.      RRAC1140
C                                                                        RRAC1150
      CALL STORE (C,LCTNLE,LCTNRE)                                       RRAC1160
      CALL SETP (C,XTABLE(LCTNRE),NLSMTH)                                RRAC1170
      LCTNLE=LCTNRE                                                      RRAC1180
      LCTNRE=MIN0(MAXNUM,MAXINT+LCTNLE-1)                                RRAC1190
20 CONTINUE                                                              RRAC1200
   WRITE (6,60) NINT                                                     RRAC1210
   ERROR=.TRUE.                                                          RRAC1220
   RETURN                                                                RRAC1230
30 CALL LSTINT (C,LNGTH,MAXNUM,ABORT)                                    RRAC1240
   IF (ABORT) GO TO 50                                                   RRAC1250
40 CALL STORE (C,LCTNLE,LCTNRE)                                          RRAC1260
   RETURN                                                                RRAC1270
50 WRITE (6,70)                                                          RRAC1280
   ERROR=.TRUE.                                                          RRAC1290
   RETURN                                                                RRAC1300
C                                                                        RRAC1310
60 FORMAT (1H0,39(2H* ),1H*,/,2H0*,12X,37HTHIS APPROXIMATION REQUIRES    RRAC1320
  1 MORE THAN,I3,13H SUBINTERVALS,12X,1H*,/,2H0*,28X,20H--PROGRAM ABO    RRAC1330
  2RTING--,20X,1H*,/,1H0,39(2H* ),1H*)                                   RRAC1340
70 FORMAT (1H0,39(2H* ),1H*,/,2H0*,15X,46HTHE ALGORITHM CANNOT MEET T    RRAC1350
  1HE DESIRED ACCURACY,16X,1H*,/,2H0*,28X,20H--PROGRAM ABORTING--,29X    RRAC1360
  2,1H*,/,1H0,39(2H* ),1H*)                                              RRAC1370
C                                                                        RRAC1380
      END                                                                RRAC1390

      SUBROUTINE COMPUT (C,LNGTH,MAXNUM,LAST,DONE,ABORT)                 COMPUT10
C                                                                        COMPUT20
C  THIS SUBROUTINE FINDS THE LARGEST SUBINTERVAL AND THE BEST APPROX-    COMPUT30
C  IMATION TO F ON THIS SUBINTERVAL SUCH THAT THE APPROXIMATION MEETS    COMPUT40
C  THE DESIRED ERROR TOLERANCE ON THE SUBINTERVAL.                       COMPUT50
C                                                                        COMPUT60
      INTEGER A,B                                                        COMPUT70
      LOGICAL LAST,OK,DONE,ABORT,TOOBIG                                  COMPUT80
      REAL C(18)                                                         COMPUT90
      COMMON XTABLE(1000),LCTNLE,LCTNRE,CSTORE(18,60),CDERIV(500)        COMPU100
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT         COMPU110
      COMMON /COMP/ LCTNX(18)                                            COMPU120
C                                                                        COMPU130
C  WE ASSUME THAT WE ARE CLOSE ENOUGH TO THE TRUE LARGEST SUBINTERVAL    COMPU140
C  RIGHT END POINT WHEN WE KNOW THAT OUR APPROXIMATION TO THE TRUE RIGHT  COMPU150
C  END POINT IS WITHIN ETA OF THE TRUE END POINT.                        COMPU160
C                                                                        COMPU170
      DATA ETA/.08/                                                      COMPU180
C                                                                        COMPU190
      LITTLE=LCTNLE+LNGTH-1                                              COMPU200
      A=0                                                                COMPU210
      LAST=.FALSE.                                                       COMPU220
10    NUMPTS=LCTNRE-LCTNLE+1                                             COMPU230
C                                                                        COMPU240
C  IF THERE DOES NOT EXIST A BEST RESTRICTED RANGE APPROXIMATION ON THE  COMPU250
```

```
C  CURRENT SUBINTERVAL, CONTROL IS PASSED TO LINE 30.                    COMPU260
C                                                                       COMPU270
      CALL REMES (C,LCTNX,TOOBIG,ABORT)                                 COMPU280
      IF (ABORT) RETURN                                                 COMPU290
      IF (TOOBIG) GO TO 30                                              COMPU300
      IF (LCTNRE.LT.MAXNUM) GO TO 20                                    COMPU310
      DONE=.TRUE.                                                       COMPU320
      RETURN                                                            COMPU330
C                                                                       COMPU340
C  A IS THE CURRENT LARGEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT     COMPU350
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL SATIFIES ALL CONSTRAINTS.  COMPU360
C                                                                       COMPU370
   20 A=LCTNRE                                                          COMPU380
      IF ((XTABLE(B)-XTABLE(A).GT.ETA).AND.(B-A.GT.1)) GO TO 40         COMPU390
      GO TO 50                                                          COMPU400
C                                                                       COMPU410
C  B IS THE CURRENT SMALLEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT    COMPU420
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL FAILS TO SATISFY THE CON- COMPU430
C  STRAINTS.                                                            COMPU440
C                                                                       COMPU450
   30 B=LCTNRE                                                          COMPU460
   40 NEWTRY=(A+B)/2+1                                                  COMPU470
      IF (NEWTRY.EQ.B) NEWTRY=NEWTRY-1                                  COMPU480
      IF (NEWTRY.LT.LITTLE) NEWTRY=LITTLE                               COMPU490
      IF (NEWTRY.EQ.LCTNRE) GO TO 50                                    COMPU500
      LCTNRE=NEWTRY                                                     COMPU510
      GO TO 10                                                          COMPU520
C                                                                       COMPU530
C  IF A IS STILL 0, THEN NO SUBINTERVAL WITH AT LEAST LNGTH POINTS      COMPU540
C  WILL WORK, SO THE ALGORITHM IS TERMINATED.                           COMPU550
C                                                                       COMPU560
   50 IF (A.NE.0) GO TO 60                                              COMPU570
      ABORT=.TRUE.                                                      COMPU580
      RETURN                                                            COMPU590
C                                                                       COMPU600
C  SINCE NEWTRY IS ALWAYS STRICTLY LESS THAN THE CURRENT B, IF NEWTRY=  COMPU610
C  LCTNRE, AND A IS NOT STILL 0, NEWTRY=A, WHICH IS A POINT WHICH SAT-  COMPU620
C  ISFIES ALL REQUIREMENTS.  WE NOW BACK THE RIGHT ENDPOINT OFF TO THE  COMPU630
C  BEST INTERIOR EXTREME POINT OF F-P TO ADD TO THE STABILITY OF THE AL- COMPU640
C  GORITHM.                                                             COMPU650
C                                                                       COMPU660
   60 DO 70 I=1,N                                                       COMPU670
         CDERIV(I)=C(I)                                                 COMPU680
         CSTORE(I,NINT)=C(I)                                            COMPU690
   70 CONTINUE                                                          COMPU700
      NDUMMY=N                                                          COMPU710
      CALL DERIV (CDERIV,NDUMMY)                                        COMPU720
      I=NX                                                              COMPU730
      LCTNRE=LCTNX(I)                                                   COMPU740
      NEWTRY=LCTNRE                                                     COMPU750
      SMLL=CMPR(CDERIV,NDUMMY,NEWTRY,LCTNLE-1,OK)                       COMPU760
      IF (OK) GO TO 100                                                 COMPU770
   80 I=I-1                                                             COMPU780
      IF (I.EQ.0) GO TO 100                                             COMPU790
      NEWTRY=LCTNX(I)                                                   COMPU800
      IF (NEWTRY.LT.LNGTH) GO TO 100                                    COMPU810
      TEMP=CMPR(CDERIV,NDUMMY,NEWTRY,LCTNLE-1,OK)                       COMPU820
      IF (.NOT.OK) GO TO 90                                             COMPU830
      LCTNRE=NEWTRY                                                     COMPU840
      GO TO 100                                                         COMPU850
   90 IF (TEMP.GE.SMLL) GO TO 80                                        COMPU860
      SMLL=TEMP                                                         COMPU870
```

```
      LCTNRE=NEWTRY                                                    COMPU880
      GO TO 80                                                         COMPU890
  100 LCTNRE=LCTNLE+LCTNRE-1                                           COMPU900
      IF (MAXNUM-LCTNRE+1.LT.LNGTH) LAST=.TRUE.                        COMPU910
      RETURN                                                           COMPU920
C                                                                      COMPU930
      END                                                              COMPU940

      SUBROUTINE LSTINT (C,LNGTH,MAXNUM,ABORT)                         LSTINT10
C                                                                      LSTINT20
C  THIS SUBROUTINE HANDLES THE SPECIAL CASE OF FINDING A SUBINTERVAL    LSTINT30
C  AND A BEST APPROXIMATION ON THAT SUBINTERVAL WHEN THERE ARE TOO      LSTINT40
C  FEW REMAINING POINTS FOR COMPUT TO WORK.                             LSTINT50
C                                                                      LSTINT60
      INTEGER OLDLE,OLDRE                                              LSTINT70
      LOGICAL TOOBIG,ABORT                                             LSTINT80
      REAL C(18)                                                       LSTINT90
      COMMON X(1000),LCTNLE,LCTNRE,CSTORE(18,60)                       LSTIN100
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT       LSTIN110
      COMMON /COMP/ LCTNX(18)                                          LSTIN120
C                                                                      LSTIN130
      DO 10 OLDLE=1,NPLUS1                                             LSTIN140
   10 CSTORE(OLDLE,NINT)=C(OLDLE)                                      LSTIN150
      OLDLE=LCTNLE                                                     LSTIN160
      OLDRE=LCTNRE                                                     LSTIN170
      LCTNRE=MAXNUM                                                    LSTIN180
      LCTNLE=MIN0(MAXNUM-LNGTH+1,(MAXNUM-OLDLE+1)/2)-1                 LSTIN190
   20 LCTNLE=LCTNLE+1                                                  LSTIN200
      IF (MAXNUM-LCTNLE+1.LT.LNGTH) GO TO 40                           LSTIN210
      CALL SETP (CSTORE(1,NINT),X(LCTNLE),NLSMTH)                      LSTIN220
      NUMPTS=LCTNRE-LCTNLE+1                                           LSTIN230
      CALL REMES (C,LCTNX,TOOBIG,ABORT)                                LSTIN240
      IF (ABORT) RETURN                                                LSTIN250
      IF (TOOBIG) GO TO 20                                             LSTIN260
      CALL STORE (CSTORE(1,NINT),OLDLE,LCTNLE)                         LSTIN270
      NINT=NINT+1                                                      LSTIN280
      DO 30 OLDLE=1,NPLUS1                                             LSTIN290
   30 CSTORE(OLDLE,NINT)=C(OLDLE)                                      LSTIN300
      RETURN                                                           LSTIN310
   40 ABORT=.TRUE.                                                     LSTIN320
      RETURN                                                           LSTIN330
C                                                                      LSTIN340
      END                                                              LSTIN350

      SUBROUTINE STORE (C,LCTNLE,LCTNRE)                               STORE 10
C                                                                      STORE 20
C  THIS SUBROUTINE OUTPUTS THE COEFFICIENTS AND ENDPOINTS OF THE       STORE 30
C  CURRENT APPROXIMATION AND SUBINTERVAL.  APPROPRIATE INFORMATION     STORE 40
C  IS STORED IN THE ARRAY CSTORE TO ALLOW THE ENTIRE PIECEWISE POLY-    STORE 50
C  NOMIAL APPROXIMATION TO BE EASILY EVALUATED AT ANY POINT BY THE      STORE 60
C  FUNCTION EVAL.                                                      STORE 70
C                                                                      STORE 80
      DIMENSION C(18)                                                  STORE 90
      COMMON XTABLE(500),FTABLE(502),CSTORE(18,60),DUM1(500)           STORE100
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT       STORE110
C                                                                      STORE120
      NUMPTS=LCTNRE-LCTNLE+1                                           STORE130
      WRITE (6,30) NINT,XTABLE(LCTNLE),XTABLE(LCTNRE),NUMPTS           STORE140
      WRITE (6,40) (I,C(I),I=1,N)                                      STORE150
      ERR=0.0                                                          STORE160
```

```
      DO 10 I=LCTNLE,LCTNRE                                          STORE170
         TEMP=ABS(FTABLE(I)-HORNER(C,XTABLE(I),N))                   STORE180
         IF (TEMP.GT.ERR) ERR=TEMP                                   STORE190
   10 CONTINUE                                                       STORE200
      WRITE (6,50) ERR                                               STORE210
      DO 20 I=1,N                                                    STORE220
   20 CSTORE(I,NINT)=C(I)                                            STORE230
      CSTORE(NPLUS1,NINT)=XTABLE(LCTNLE)                             STORE240
      RETURN                                                         STORE250
C                                                                    STORE260
   30 FORMAT (//,5X, 15HINTERVAL NUMBER,I4, 16H WHICH BEGINS AT,E23.16,/  STORE270
     1, 12H AND ENDS AT,E23.16,2X, 8HCONTAINS,I4, 8H POINTS.,/, 9H TH  STORE280
     2E COEF, 51HFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE,/)  STORE290
   40 FORMAT ((10X, 2HC(,I2, 3H) =,E24.16))                          STORE300
   50 FORMAT (/, 47H THE ERROR OF APPROXIMATION IN THIS INTERVAL IS,E24.  STORE310
     116, 1H.)                                                       STORE320
C                                                                    STORE330
      END                                                            STORE340


      SUBROUTINE DERIV (C,N)                                         DERIV 10
C                                                                    DERIV 20
C  THIS SUBROUTINE REPLACES THE COEFFICIENTS OF A POLYNOMIAL IN STAND-  DERIV 30
C  ARD FORM WITH THE COEFFICIENTS OF THIS POLYNOMIAL-S DERIVATIVE.    DERIV 40
C  THE NUMBER OF COEFFICIENTS, N, IS DECREMENTED.                    DERIV 50
C                                                                    DERIV 60
      DIMENSION C(N)                                                 DERIV 70
C                                                                    DERIV 80
      N=N-1                                                          DERIV 90
      DO 10 I=1,N                                                    DERIV100
   10 C(I)=FLOAT(I)*C(I+1)                                           DERIV110
      RETURN                                                         DERIV120
C                                                                    DERIV130
      END                                                            DERIV140


      SUBROUTINE SETP (C,X,SMTH)                                     SETP  10
C                                                                    SETP  20
C  THIS SUBROUTINE APPROPRIATELY STORES IN THE ARRAY PPRIME THE VAL-  SETP  30
C  UES WHICH MUST BE INTERPOLATED TO GIVE THE PIECEWISE POLYNOMIAL THE  SETP  40
C  DESIRED SMOOTHNESS.                                               SETP  50
C                                                                    SETP  60
      DIMENSION C(18)                                                SETP  70
      COMMON /COMP/ CSTORE(18)                                       SETP  80
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT     SETP  90
      COMMON /DDIF/ PPRIME(5),DUM1(36)                               SETP 100
      INTEGER SMTH                                                   SETP 110
      DO 10 I=1,N                                                    SETP 120
   10 CSTORE(I)=C(I)                                                 SETP 130
      NDUMMY=N                                                       SETP 140
      I=0                                                            SETP 150
   20 IF (I.GT.SMTH) RETURN                                          SETP 160
      IF (I.EQ.0) GO TO 30                                           SETP 170
      CALL DERIV (CSTORE,NDUMMY)                                     SETP 180
   30 PPRIME(I+1)=HORNER(CSTORE,X,NDUMMY)                            SETP 190
      I=I+1                                                          SETP 200
      GO TO 20                                                       SETP 210
C                                                                    SETP 220
      END                                                            SETP 230


      FUNCTION CMPR(C,N,NEWTRY,OK)                                   CMPR  10
```

```
C                                                                       CMPR  20
C  THIS SUBROUTINE COMPARES THE FIRST DERIVATIVE OF THE CURRENT PIECE OF CMPR  30
C  THE PIECEWISE POLYNOMIAL APPROXIMATION EVALUATED AT XTABLE(NEWTRY)   CMPR  40
C  WITH THE FIRST DERIVATIVE OF THE QUADRATIC INTERPOLATION OF F, CEN-   CMPR  50
C  TERED AROUND XTABLE(NEWTRY), EVALUATED AT XTABLE(NEWTRY).  IF THESE  CMPR  60
C  TWO DIFFER IN ABSOLUTE VALUE BY LESS THAN TOLER (EITHER ABSOLUTELY OR CMPR  70
C  RELATIVELY), WE SET OK TO .TRUE. AND WE ACCEPT XTABLE(NEWTRY) AS A   CMPR  80
C  REASONABLE SUBINTERVAL RIGHT ENDPOINT.  NOTE THAT THE USER MAY EASILY CMPR  90
C  CHANGE TOLER BY MEANS OF THE FOLLOWING DATA STATEMENT.               CMPR 100
C                                                                       CMPR 110
      LOGICAL OK                                                        CMPR 120
      COMMON XTABLE(500),FTABLE(500),DUM1(1582)                         CMPR 130
      DIMENSION C(18)                                                   CMPR 140
      DATA TOLER/.01/                                                   CMPR 150
C                                                                       CMPR 160
      OK=.FALSE.                                                        CMPR 170
      A=(FTABLE(NEWTRY)-FTABLE(NEWTRY-1))/(XTABLE(NEWTRY)-XTABLE(NEWTRY-CMPR 180
     11))                                                               CMPR 190
      B=(FTABLE(NEWTRY+1)-FTABLE(NEWTRY))/(XTABLE(NEWTRY+1)-XTABLE(NEWTR CMPR 200
     1Y))                                                               CMPR 210
      D=(B-A)/(XTABLE(NEWTRY+1)-XTABLE(NEWTRY-1))                       CMPR 220
      A=A+D*(XTABLE(NEWTRY)-XTABLE(NEWTRY-1))                           CMPR 230
      B=HORNER(C,XTABLE(NEWTRY),N)                                      CMPR 240
      CMPR=ABS(A-B)                                                     CMPR 250
      IF (CMPR.LE.TOLER) OK=.TRUE.                                      CMPR 260
      A=ABS(A)                                                          CMPR 270
      IF (A.LT..01) RETURN                                              CMPR 280
      IF (CMPR/A.LT.TOLER) OK=.TRUE.                                    CMPR 290
      RETURN                                                            CMPR 300
C                                                                       CMPR 310
      END                                                               CMPR 320

      SUBROUTINE REMES (C,LCTNX,TOOBIG,ABORT)                           REMES 10
C                                                                       REMES 20
C  THIS IS THE DRIVING PROGRAM FOR THE COMPUTATION OF THE BEST          REMES 30
C  RESTRICTED RANGE UNIFORM POLYNOMIAL APPROXIMATION TO F(X) (VALUES    REMES 40
C  ARE STORED IN XTABLE AND FTABLE) OF DEGREE LESS THAN OR EQUAL TO N-1 REMES 50
C  ON THE SUBINTERVAL (XTABLE(LCTNLE),XTABLE(LCTNRE)).  SEE THE PAPER BY REMES 60
C  B. CHALMERS FOR ADDITIONAL INFORMATION ON THIS ALGORITHM.            REMES 70
C                                                                       REMES 80
      DIMENSION XTPTS(18), C(18), SGNRXI(18), LCTNX(18), LCTNZ(18)      REMES 90
      COMMON XTABLE(500),FTABLE(500),LCTNLE,DUMMY(1081),ERROR(500),FU(50REMES100
     10),FL(500)                                                        REMES110
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMGRD,NINT        REMES120
      COMMON /DOIF/ DUM1(23),D(18)                                      REMES130
      LOGICAL STOP,TOOBIG,ABORT                                         REMES140
      INTEGER FRSTM1,START                                              REMES150
C                                                                       REMES160
C  EPS IS A MACHINE CONSTANT--SET EPS TO (APPROXIMATELY) THE SMALLEST   REMES170
C  VALUE SUCH THAT EPS + 1.0 IS GREATER THAN 1.0.                       REMES180
C                                                                       REMES190
      DATA ITERMX,EPS/30,1.0E-10/                                       REMES200
C                                                                       REMES210
C  FIRST WE INITIALIZE VARIOUS ARRAYS AND VALUES.                       REMES220
C                                                                       REMES230
      TOOBIG=.FALSE.                                                    REMES240
      FRSTM1=LCTNLE-1                                                   REMES250
      SGNRXI(1)=1.0                                                     REMES260
      DO 10 I=2,NX                                                      REMES270
   10 SGNRXI(I)=-SGNRXI(I-1)                                            REMES280
      NFPTS=NUMGRD-2                                                    REMES290
```

```
      START=2                                                          REMES300
      ERROR(1)=0.0                                                     REMES310
      IF (NLSMTH.GE.0) GO TO 20                                        REMES320
      NFPTS=NFPTS+1                                                    REMES330
      START=1                                                          REMES340
   20 IF (NRSMTH.LT.0) NFPTS=NFPTS+1                                   REMES350
      DELTA=FLOAT(NFPTS-1)/FLOAT(NX-1)                                 REMES360
      DO 30 I=1,NX                                                     REMES370
         LCTNX(I)=START+IFIX(FLOAT(I-1)*DELTA+.5)                      REMES380
         J=FRSTM1+LCTNX(I)                                             REMES390
         XTPTS(I)=XTABLE(J)                                            REMES400
         D(I)=FTABLE(J)                                                REMES410
   30 CONTINUE                                                         REMES420
C                                                                      REMES430
C  NOW WE BEGIN ITERATING.  CONVERGENCE OCCURS WHEN TWO CONSECUTIVE    REMES440
C  REFERENCE SETS (DETERMINED BY SOLVE) ARE THE SAME.                  REMES450
C                                                                      REMES460
      DO 70 ITER=1,ITERMX                                              REMES470
         CALL DIVDIF (C,LCTNX,SGNRXI,ABORT)                            REMES480
         IF (ABORT) RETURN                                             REMES490
         DO 40 I=START,NUMGRD                                          REMES500
   40    ERROR(I)=FTABLE(I+FRSTM1)-BPOLY(XTABLE(I+FRSTM1),C,N)         REMES510
         IF (ABS(C(NPLUS1)).LE.EPS) GO TO 60                           REMES520
         DO 50 I=1,NX                                                  REMES530
            IF (SGNRXI(I).NE.0.0) SGNRXI(I)=SIGN(1.0,ERROR(LCTNX(I)))  REMES540
            IF (SGNRXI(I).EQ.0.0) SGNRXI(I)=FLOAT(LCTNZ(I))            REMES550
            IF (I.EQ.1) GO TO 50                                       REMES560
            IF (SGNRXI(I)*SGNRXI(I-1).GE.0.0) GO TO 80                 REMES570
   50    CONTINUE                                                      REMES580
   60    CALL ZEROFD (LCTNX,LCTNZ,SGNRXI,ERROR)                        REMES590
         CALL SOLVE (XTPTS,LCTNX,LCTNZ,SGNRXI,ABS(C(NPLUS1)),STOP,TOOBIG REMES600
     1   )                                                             REMES610
         IF (.NOT.STOP) GO TO 70                                       REMES620
         IF (.NOT.TOOBIG) CALL TRANS (C,N)                             REMES630
         RETURN                                                        REMES640
   70 CONTINUE                                                         REMES650
C                                                                      REMES660
C  WE PRINT OUT ERROR MESSAGES IF SOMETHING GOES WRONG.                REMES670
C                                                                      REMES680
      WRITE (6,100) ITERMX                                             REMES690
      GO TO 90                                                         REMES700
   80 WRITE (6,110) ITER                                               REMES710
   90 ABORT=.TRUE.                                                     REMES720
      RETURN                                                           REMES730
C                                                                      REMES740
  100 FORMAT (1H0,39(2H* ),1H*,/,2H0*,11X,40HTHE REMES ALGORITHM HAS NOT REMES750
     1 CONVERGED IN,I3,12H ITERATIONS.,11X,1H*,/,2H0*,11X,36HPROGRAM ABO REMES760
     2RTED IN SUBROUTINE REMES.,30X,1H*,/,1H0,39(2H* ),1H*)            REMES770
  110 FORMAT (1H0,39(2H* ),1H*,/,2H0*,8X,12HIN ITERATION,I3,47H OF REMES REMES780
     1, NO ALTERNATION OF SIGN OCCURS AT THE,7X,1H*,/,2H0*,8X,57HEXTREME REMES790
     2 POINTS.  THE PROGRAM ABORTED IN SUBROUTINE REMES.,12X,1H*,/,1H0,3 REMES800
     39(2H* ),1H*)                                                     REMES810
C                                                                      REMES820
      END                                                              REMES830


      SUBROUTINE DIVDIF (C,LCTNX,SGNRXI,ABORT)                         DIVDIF10
C                                                                      DIVDIF20
C  THIS SUBROUTINE MAKES USE OF A DIVIDED DIFFERENCE SCHEME FOR SOLVING DIVDIF30
C  THE VANDERMONDE-LIKE LINEAR SYSTEM INHERENT IN THE REMES ALGORITHM. DIVDIF40
C  THE ADVANTAGES OF USING THIS SPECIAL PURPOSE LINEAR SYSTEM SOLVER   DIVDIF50
C  ARE--                                                               DIVDIF60
```

```
C                                                                       DIVDIF70
C     THIS ROUTINE REQUIRES ON THE ORDER OF N**2 OPERATIONS AS COMPARED DIVDIF80
C     TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER OF N**3 OPER- DIVDIF90
C     ATIONS.                                                           DIVDI100
C                                                                       DIVDI110
C     THIS ROUTINE REQUIRES ON THE ORDER OF N STORAGE LOCATIONS AS COM- DIVDI120
C     PARED TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER OF N**2. DIVDI130
C                                                                       DIVDI140
C  SEE THE FORTHCOMING PAPER BY J. A. HULL, S. F. MCCORMICK, AND G. D.  DIVDI150
C  TAYLOR FOR A COMPLETE DESCRIPTION OF THIS ALGORITHM.                 DIVDI160
C                                                                       DIVDI170
      COMMON XTABLE(500),FTABLE(500),LCTNLE,LCTNRE,CSTORE(18,60),ERROR(5DIVDI180
     100)                                                               DIVDI190
      COMMON /DDIF/ PPRIME(5),X(18),D(18)                               DIVDI200
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT        DIVDI210
      DIMENSION LCTNX(18), C(18), SGNRXI(18)                            DIVDI220
      INTEGER FRSTM1                                                    DIVDI230
      LOGICAL ABORT                                                     DIVDI240
C                                                                       DIVDI250
C  FIRST WE INITIALIZE SEVERAL VARIABLES.                               DIVDI260
C                                                                       DIVDI270
      FRSTM1=LCTNLE-1                                                   DIVDI280
      IF (NRSMTH.GE.0) SGNRXI(NPLUS1)=0.0                               DIVDI290
      IF (NRSMTH.LT.0) SGNRXI(NPLUS1)=SGNRXI(NX)                        DIVDI300
C                                                                       DIVDI310
C  SET UP THE VECTOR X AND THE FIRST ROW OF THE DIVIDED DIFFERENCE TAB- DIVDI320
C  LE, USING THE COEFFICIENT VECTOR C FOR TEMPORARY STORAGE.            DIVDI330
C                                                                       DIVDI340
      I=0                                                               DIVDI350
   10 IF (I.GT.NLSMTH) GO TO 20                                         DIVDI360
      I=I+1                                                             DIVDI370
      X(I)=XTABLE(LCTNLE)                                               DIVDI380
      C(I)=PPRIME(1)                                                    DIVDI390
      GO TO 10                                                          DIVDI400
   20 J=0                                                               DIVDI410
   30 IF (J.GE.NX) GO TO 40                                             DIVDI420
      I=I+1                                                             DIVDI430
      J=J+1                                                             DIVDI440
      X(I)=XTABLE(FRSTM1+LCTNX(J))                                      DIVDI450
      C(I)=D(J)                                                         DIVDI460
      GO TO 30                                                          DIVDI470
   40 IF (I.GE.NPLUS1) GO TO 50                                         DIVDI480
      I=I+1                                                             DIVDI490
      X(I)=XTABLE(LCTNRE)                                               DIVDI500
      C(I)=PPRIME(NLSMTH+2)                                             DIVDI510
      GO TO 40                                                          DIVDI520
   50 CONTINUE                                                          DIVDI530
C                                                                       DIVDI540
C  WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.                       DIVDI550
C                                                                       DIVDI560
      FAC=1.0                                                           DIVDI570
      DO 100 J=2,N                                                      DIVDI580
         JP1=J+1                                                        DIVDI590
         JM1=J-1                                                        DIVDI600
         FAC=FAC*FLOAT(JM1)                                             DIVDI610
         TEMP2=C(JM1)                                                   DIVDI620
         DO 90 I=J,N                                                    DIVDI630
            IF (X(I).NE.X(I-JM1)) GO TO 70                              DIVDI640
            IF (I.GT.NLSMTH+1) GO TO 60                                 DIVDI650
            IF (J.GT.NPLUS1-NX) GO TO 220                               DIVDI660
            TEMP1=PPRIME(J)/FAC                                         DIVDI670
            GO TO 80                                                    DIVDI680
```

```
   60        IF (NLSMTH+JP1.GT.NPLUS1-NX) GO TO 220                        DIVDI690
             TEMP1=PPRIME(NLSMTH+JP1)/FAC                                  DIVDI700
   70        TEMP1=(C(I)-C(I-1))/(X(I)-X(I-JM1))                           DIVDI710
   80        C(I-1)=TEMP2                                                  DIVDI720
             TEMP2=TEMP1                                                   DIVDI730
   90     CONTINUE                                                         DIVDI740
          C(N)=TEMP2                                                       DIVDI750
  100  CONTINUE                                                           DIVDI760
C                                                                          DIVDI770
C  L**(-1)F HAS NOW BEEN TEMPORARILY STORED IN THE COEFFICIENT ARRAY C.    DIVDI780
C  WE NOW COMPUTE L**(-1)C.  WE WILL COMPUTE THE DIVIDED DIFFERENCES       DIVDI790
C  IN THE TEMPORARY STORAGE ARRAY D.  FIRST WE SET UP THE FIRST COLUMN     DIVDI800
C  OF THE DIVIDED DIFFERENCE TABLE.                                        DIVDI810
C                                                                          DIVDI820
       I=0                                                                 DIVDI830
  110  IF (I.GT.NLSMTH) GO TO 120                                          DIVDI840
       I=I+1                                                               DIVDI850
       D(I)=0.0                                                            DIVDI860
       GO TO 110                                                           DIVDI870
  120  J=0                                                                 DIVDI880
  130  IF (J.GE.NX) GO TO 140                                              DIVDI890
       I=I+1                                                               DIVDI900
       J=J+1                                                               DIVDI910
       D(I)=SGNRXI(J)                                                      DIVDI920
       GO TO 130                                                           DIVDI930
  140  IF (I.GE.N) GO TO 150                                               DIVDI940
       I=I+1                                                               DIVDI950
       D(I)=0.0                                                            DIVDI960
       GO TO 140                                                           DIVDI970
  150  CONTINUE                                                            DIVDI980
C                                                                          DIVDI990
C  WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.                          DIVD1000
C                                                                          DIVD1010
       DO 190 J=2,N                                                        DIVD1020
          TEMP2=D(J-1)                                                     DIVD1030
          DO 180 I=J,N                                                     DIVD1040
             IF (X(I).NE.X(I-J+1)) GO TO 160                               DIVD1050
             TEMP1=0.0                                                     DIVD1060
             GO TO 170                                                     DIVD1070
  160        TEMP1=(D(I)-D(I-1))/(X(I)-X(I-J+1))                           DIVD1080
  170        D(I-1)=TEMP2                                                  DIVD1090
             TEMP2=TEMP1                                                   DIVD1100
  180     CONTINUE                                                         DIVD1110
          D(N)=TEMP2                                                       DIVD1120
  190  CONTINUE                                                            DIVD1130
C                                                                          DIVD1140
C  WE NOW COMPUTE M=(F(N+1)-W[TRANSPOSE]*B1))/(0.0-W[TRANSPOSE]*B2)        DIVD1150
C  =C(N+1)=(UNIFORM ERROR)                                                 DIVD1160
C  FIRST WE COMPUTE THE TWO DOT PRODUCTS SIMULTANEOUSLY.                   DIVD1170
C                                                                          DIVD1180
       W=1.0                                                               DIVD1190
       B1=0.0                                                              DIVD1200
       B2=0.0                                                              DIVD1210
       DO 200 I=1,N                                                        DIVD1220
          B1=B1+C(I)*W                                                     DIVD1230
          B2=B2+D(I)*W                                                     DIVD1240
          W=W*(X(NPLUS1)-X(I))                                             DIVD1250
  200  CONTINUE                                                            DIVD1260
C                                                                          DIVD1270
C  NOW WE COMPUTE M                                                        DIVD1280
C                                                                          DIVD1290
       C(NPLUS1)=(C(NPLUS1)-B1)/(SGNRXI(NPLUS1)-B2)                        DIVD1300
```

```
C                                                                       DIVD1310
C  FINALLY, WE COMPUTE THE COEFFICIENTS C(I).                           DIVD1320
C                                                                       DIVD1330
      DO 210 I=1,N                                                      DIVD1340
         C(I)=C(I)-C(NPLUS1)*D(I)                                       DIVD1350
  210 CONTINUE                                                          DIVD1360
      RETURN                                                            DIVD1370
  220 ABORT=.TRUE.                                                      DIVD1380
      WRITE (6,230)                                                     DIVD1390
      RETURN                                                            DIVD1400
C                                                                       DIVD1410
  230 FORMAT (1H0,39(2H* ),1H*,/,2H0*,11X,54HSUBROUTINE DIVDIF HAS FAILE DIVD1420
     1D--PROBABLY DUE TO AN INPUT,12X,1H*,/,2H0*,11X,58HERROR (TO DIVDIF DIVD1430
     2)--NOT ENOUGH ELEMENTS IN THE ARRAY PPRIME,8X,1H*,/,2H0*,11X,56HOR DIVD1440
     3 TWO IDENTICAL POINTS IN THE ARRAY X.  PROGRAM ABORTED,10X,1H*,/,2 DIVD1450
     4H0*,11X,21HIN SUBROUTINE DIVDIF.,45X,1H*,/,1H0,39(2H* ),1H*)      DIVD1460
C                                                                       DIVD1470
      END                                                               DIVD1480


      SUBROUTINE ZEROFD (LCTNX,LCTNZ,SGNRXI,ERROR)                      ZEROFD10
C                                                                       ZEROFD20
C  THIS SUBROUTINE LOCATES THE (APPROXIMATE) ZEROS BETWEEN CONSECUTIVE  ZEROFD30
C  POINTS OF THE CURRENT REFERENCE SET.  THE LOCATIONS OF THESE POINTS   ZEROFD40
C  IN THE ARRAY XTABLE RELATIVE TO THE BEGINNING OF THE CURRENT SUB-     ZEROFD50
C  INTERVAL ARE STORED IN LCTNZ.                                        ZEROFD60
C                                                                       ZEROFD70
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMGRD,NINT        ZEROFD80
      DIMENSION LCTNX(18), LCTNZ(18), SGNRXI(18), ERROR(300)            ZEROFD90
C                                                                       ZEROF100
      LCTNZ(1)=MINO(2,2+NLSMTH)                                         ZEROF110
      LCTNZ(NX+1)=MAX0(NUMGRD-1,NUMGRD-1-NRSMTH)                        ZEROF120
      DO 30 I=2,NX                                                      ZEROF130
         LTNIM1=LCTNX(I-1)                                              ZEROF140
         NUMPTS=LCTNX(I)-LTNIM1                                         ZEROF150
         DO 10 J=1,NUMPTS                                               ZEROF160
            NEWTRY=LTNIM1+J                                             ZEROF170
            IF (ERROR(LTNIM1)*ERROR(NEWTRY).LE.0.0) GO TO 20            ZEROF180
   10    CONTINUE                                                       ZEROF190
   20    LCTNZ(I)=NEWTRY                                                ZEROF200
   30 CONTINUE                                                          ZEROF210
      RETURN                                                            ZEROF220
C                                                                       ZEROF230
      END                                                               ZEROF240


      SUBROUTINE SOLVE (X,LCTNX,LCTNZ,SGNRXI,E,STOP,TOOBIG)            SOLVE 10
C                                                                       SOLVE 20
C  THIS SUBROUTINE PERFORMS THE MULTIPLE EXCHANGE OF THE REFERENCE      SOLVE 30
C  SET REQUIRED AT EACH ITERATION OF THE REMES ALGORITHM.               SOLVE 40
C                                                                       SOLVE 50
      COMMON XTABLE(500),FTABLE(500),LCTNLE,DUMMY(1081),ERROR(500),FU(50 SOLVE 60
     10),FL(500)                                                        SOLVE 70
      COMMON /DDIF/ DUM1(23),D(18)                                      SOLVE 80
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,DUM(4)                           SOLVE 90
      DIMENSION X(18), LCTNX(18), LCTNZ(18), SGNRXI(18)                 SOLVE100
      LOGICAL STOP,TOOBIG                                               SOLVE110
      INTEGER FRSTM1,RTEND                                              SOLVE120
C                                                                       SOLVE130
C  EPS IS A MACHINE CONSTANT--SET EPS TO ROUGHLY THE SMALLEST NUMBER    SOLVE140
C  SUCH THAT 1.0 + EPS .GT. 1.0.                                        SOLVE150
C                                                                       SOLVE160
```

```
      DATA EPS/1.0E-9/                                                  SOLVE170
C                                                                       SOLVE180
      STOP=.TRUE.                                                       SOLVE190
      FRSTM1=LCTNLE-1                                                   SOLVE200
C                                                                       SOLVE210
C  WE FIRST COMPUTE THE LOCATIONS OF THE NEW SET OF EXTREME POINTS,     SOLVE220
C  STORING THEM IN THE VECTOR LCTNX.  WE BEGIN BY CHOOSING AS THE ITH   SOLVE230
C  ELEMENT OF LCTNX THE LOCATION OF THE GRIDPOINT IN THE SUBINTERVAL    SOLVE240
C  BETWEEN THE ITH AND (I+1)ST ZERO WHICH RESULTS IN THE LARGEST ERROR  SOLVE250
C  OF THE SAME SIGN AS THE PREVIOUS ITH EXTREME POINT (THEREBY GUARANT- SOLVE260
C  EEING ALTERNATION).  AT THE SAME TIME WE SEARCH FOR THE GRIDPOINT    SOLVE270
C  WHICH RESULTS IN THE LARGEST (ABSOLUTE) ERROR, STORING ITS LOCATION  SOLVE280
C  (IN LNBGST) AND THE NUMBER OF THE SUBINTERVAL IN WHICH IT OCCURS (IN SOLVE290
C  INBGST).                                                             SOLVE300
C                                                                       SOLVE310
      BIGER=-1.0E30                                                     SOLVE320
      BIGEST=-1.0E30                                                    SOLVE330
      DO 70 INTNUM=1,NX                                                 SOLVE340
         BIG=-1.0E30                                                    SOLVE350
         LFTEND=LCTNZ(INTNUM)                                           SOLVE360
         RTEND=LCTNZ(INTNUM+1)                                          SOLVE370
         SGN=SGNRXI(INTNUM)                                             SOLVE380
         DO 60 NEWLOC=LFTEND,RTEND                                      SOLVE390
            SAME1=SGN*ERROR(NEWLOC)-E                                   SOLVE400
            OPP1=-SGN*ERROR(NEWLOC)-E                                   SOLVE410
            IF (SGN.LT.0.0) GO TO 10                                    SOLVE420
            SAME2=ERROR(NEWLOC)-FL(FRSTM1+NEWLOC)                       SOLVE430
            OPP2=-ERROR(NEWLOC)-FU(FRSTM1+NEWLOC)                       SOLVE440
            GO TO 20                                                    SOLVE450
   10       SAME2=-ERROR(NEWLOC)-FU(FRSTM1+NEWLOC)                      SOLVE460
            OPP2=ERROR(NEWLOC)-FL(FRSTM1+NEWLOC)                        SOLVE470
   20       IF (SAME1.LE.BIG) GO TO 30                                  SOLVE480
            BIG=SAME1                                                   SOLVE490
            LCTNX(INTNUM)=NEWLOC                                        SOLVE500
            LCTNZ(INTNUM)=0                                             SOLVE510
   30       IF (SAME2.LE.BIG) GO TO 40                                  SOLVE520
            BIG=SAME2                                                   SOLVE530
            LCTNX(INTNUM)=NEWLOC                                        SOLVE540
            LCTNZ(INTNUM)=IFIX(SGN)                                     SOLVE550
   40       IF (BIGER.LT.BIG) BIGER=BIG                                 SOLVE560
            IF (BIGEST.LT.BIG) BIGEST=BIG                               SOLVE570
            IF (OPP1.LE.BIGEST) GO TO 50                                SOLVE580
            BIGEST=OPP1                                                 SOLVE590
            INBGST=INTNUM                                               SOLVE600
            LNBGST=NEWLOC                                               SOLVE610
            KIND=0                                                      SOLVE620
   50       IF (OPP2.LE.BIGEST) GO TO 60                                SOLVE630
            BIGEST=OPP2                                                 SOLVE640
            INBGST=INTNUM                                               SOLVE650
            LNBGST=NEWLOC                                               SOLVE660
            KIND=-IFIX(SGN)                                             SOLVE670
   60    CONTINUE                                                       SOLVE680
   70 CONTINUE                                                          SOLVE690
      IF (BIGEST.LT.EPS) RETURN                                         SOLVE700
      IF (ABS(BIGER-BIGEST).LT.EPS) GO TO 120                           SOLVE710
C                                                                       SOLVE720
C  AT THIS POINT IT IS STILL NECESSARY TO INSERT THE LOCATION OF THE    SOLVE730
C  GRIDPOINT RESULTING IN THE LARGEST ERROR INTO LCTNX.  THERE ARE THREE SOLVE740
C  CASES, EACH OF WHICH IS HANDLED SEPARATELY--ALTERNATION IS PRESERVED SOLVE750
C  AT THE GRIDPOINTS.                                                   SOLVE760
C                                                                       SOLVE770
      YBIGST=XTABLE(FRSTM1+LNBGST)                                      SOLVE780
```

```
      YIBIG=XTABLE(LCTNX(INBGST)+FRSTM1)                                SOLVE790
      IF ((INBGST.EQ.1).AND.(YBIGST.LT.YIBIG)) GO TO 80                 SOLVE800
      IF ((INBGST.EQ.NX).AND.(YBIGST.GT.YIBIG)) GO TO 100               SOLVE810
      IF (YBIGST.LT.YIBIG) NEWINT=INBGST-1                              SOLVE820
      IF (YBIGST.GT.YIBIG) NEWINT=INBGST+1                              SOLVE830
      LCTNX(NEWINT)=LNBGST                                              SOLVE840
      LCTNZ(NEWINT)=KIND                                                SOLVE850
      GO TO 120                                                         SOLVE860
   80 DO 90 I=2,NX                                                      SOLVE870
         J=NX-I+2                                                       SOLVE880
         LCTNX(J)=LCTNX(J-1)                                            SOLVE890
         LCTNZ(J)=LCTNZ(J-1)                                            SOLVE900
         SGNRXI(J)=SGNRXI(J-1)                                          SOLVE910
   90 CONTINUE                                                          SOLVE920
      LCTNX(1)=LNBGST                                                   SOLVE930
      LCTNZ(1)=KIND                                                     SOLVE940
      IF (KIND.EQ.0) SGNRXI(1)=-SGNRXI(1)                               SOLVE950
      GO TO 120                                                         SOLVE960
  100 DO 110 J=1,NXM1                                                   SOLVE970
         LCTNX(J)=LCTNX(J+1)                                            SOLVE980
         LCTNZ(J)=LCTNZ(J+1)                                            SOLVE990
         SGNRXI(J)=SGNRXI(J+1)                                          SOLV1000
  110 CONTINUE                                                          SOLV1010
      LCTNX(NX)=LNBGST                                                  SOLV1020
      LCTNZ(NX)=KIND                                                    SOLV1030
      IF (KIND.EQ.0) SGNRXI(NX)=-SGNRXI(NX)                             SOLV1040
C                                                                       SOLV1050
C  NOW THAT LCTNX IS ACCEPTABLE, WE CHECK FOR CONVERGENCE OF THE ALGOR- SOLV1060
C  ITHM, STORING THE EXTREME POINTS IF THE CONVERGENCE CRITERION IS NOT  SOLV1070
C  MET.                                                                 SOLV1080
C                                                                       SOLV1090
  120 DO 130 I=1,NX                                                     SOLV1100
         IF (LCTNZ(I).EQ.0) GO TO 140                                   SOLV1110
  130 CONTINUE                                                          SOLV1120
      TOOBIG=.TRUE.                                                     SOLV1130
      RETURN                                                            SOLV1140
  140 DO 170 I=1,NX                                                     SOLV1150
         NEWLOC=FRSTM1+LCTNX(I)                                         SOLV1160
         IF (LCTNZ(I).NE.0) GO TO 150                                   SOLV1170
         D(I)=FTABLE(NEWLOC)                                            SOLV1180
         GO TO 160                                                      SOLV1190
  150    IF (LCTNZ(I).EQ.1) D(I)=FTABLE(NEWLOC)-FL(NEWLOC)              SOLV1200
         IF (LCTNZ(I).EQ.-1) D(I)=FTABLE(NEWLOC)+FU(NEWLOC)             SOLV1210
         SGNRXI(I)=0.0                                                  SOLV1220
  160    TEMP=XTABLE(NEWLOC)                                            SOLV1230
         IF (ABS(TEMP-X(I)).LE.EPS) GO TO 170                           SOLV1240
         X(I)=TEMP                                                      SOLV1250
         STOP=.FALSE.                                                   SOLV1260
  170 CONTINUE                                                          SOLV1270
      RETURN                                                            SOLV1280
C                                                                       SOLV1290
      END                                                               SOLV1300


      FUNCTION BPOLY(XX,C,N)                                            BPOLY 10
C                                                                       BPOLY 20
C  THIS FUNCTION IS USED TO EVALUATE (BY THE APPROPRIATE ADAPTATION OF  BPOLY 30
C  HORNERS METHOD) THE POLYNOMIAL                                       BPOLY 40
C                                                                       BPOLY 50
C     C(1) + C(2)*(XX-X(1)) + C(3)*(XX-X(1))*(XX-X(2)) + . . .          BPOLY 60
C       + C(N)*(XX-X(1))*(XX-X(2))* . . . *(XX-X(N-1))                  BPOLY 70
C                                                                       BPOLY 80
```

```
      DIMENSION C(18)                                                    BPOLY 90
      COMMON /DDIF/ DUMMY(5),X(18),DUM(18)                               BPOLY100
C                                                                        BPOLY110
      NM1=N-1                                                            BPOLY120
      BPOLY=C(N)                                                         BPOLY130
      DO 10 I=1,NM1                                                      BPOLY140
         J=N-I                                                           BPOLY150
         BPOLY=C(J)+(XX-X(J))*BPOLY                                      BPOLY160
   10 CONTINUE                                                           BPOLY170
      RETURN                                                             BPOLY180
C                                                                        BPOLY190
      END                                                                BPOLY200


      SUBROUTINE TRANS (C,N)                                             TRANS 10
C                                                                        TRANS 20
C  THIS SUBROUTINE TRANSFORMS A POLYNOMIAL WRITTEN IN THE FORM           TRANS 30
C                                                                        TRANS 40
C     C(1) + C(2)*(X-X(1)) + C(3)*(X-X(1))*(X-X(2)) + . . .             TRANS 50
C       + C(N)*(X-X(1))*(X-X(2))* . . . *(X-X(N-1))                      TRANS 60
C                                                                        TRANS 70
C  TO A POLYNOMIAL WRITTEN IN TERMS OF POWERS OF X.  THE X(I)-S ARE      TRANS 80
C  SUPPLIED BY SUBROUTINE DIVDIF.                                        TRANS 90
C                                                                        TRANS100
      DIMENSION C(18)                                                    TRANS110
      COMMON /DDIF/ DUMMY(5),X(18),DUM(18)                               TRANS120
C                                                                        TRANS130
      NM1=N-1                                                            TRANS140
      DO 20 J=1,NM1                                                      TRANS150
         K=N-J                                                           TRANS160
         DO 10 I=K,NM1                                                   TRANS170
            C(I)=C(I)-X(K)*C(I+1)                                        TRANS180
   10    CONTINUE                                                        TRANS190
   20 CONTINUE                                                           TRANS200
      RETURN                                                             TRANS210
C                                                                        TRANS220
      END                                                                TRANS230


      FUNCTION EVAL(X)                                                   EVAL  10
C                                                                        EVAL  20
C  THIS FUNCTION EVALUATES THE PIECEWISE POLYNOMIAL APPROXIMATION AT     EVAL  30
C  ANY POINT IN THE ENTIRE INTERVAL.                                     EVAL  40
C                                                                        EVAL  50
      COMMON DUMMY(1002),CSTORE(18,60),DUM(500)                          EVAL  60
      COMMON /SCALAR/ N,NPLUS1,DUM2(5),NINT                              EVAL  70
      IF (NINT.LT.2) GO TO 20                                            EVAL  80
      DO 10 I=2,NINT                                                     EVAL  90
         ISTORE=I-1                                                      EVAL 100
         IF (X.LE.CSTORE(NPLUS1,I)) GO TO 30                             EVAL 110
   10 CONTINUE                                                           EVAL 120
      ISTORE=NINT                                                        EVAL 130
      GO TO 30                                                           EVAL 140
   20 ISTORE=1                                                           EVAL 150
   30 EVAL=HORNER(CSTORE(1,ISTORE),X,N)                                  EVAL 160
      RETURN                                                             EVAL 170
C                                                                        EVAL 180
      END                                                                EVAL 190


      FUNCTION HORNER(C,X,N)                                             HORNER10
C                                                                        HORNER20
```

```
C  THIS FUNCTION EVALUATES A POLYNOMIAL IN STANDARD FORM BY HORNERS METHOD.
C
      DIMENSION C(N)
C
      HORNER=C(N)
      I=N
   10 IF (I.LT.2) RETURN
      HORNER=HORNER*X+C(I-1)
      I=I-1
      GO TO 10
C
      END
```

```
HORNER30
HORNER40
HORNER50
HORNER60
HORNER70
HORNER80
HORNER90
HORNE100
HORNE110
HORNE120
HORNE130
HORNE140
HORNE150
```

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

INPUT :

```
1                                    ( THIS DENOTES RESTRAINTS OPTION )
6    2    4.00     .175              ( N, SMTH, MAXTOL, MINTOL )
                                     ( XTABLE, FTABLE)
3.0          0.0
5.0          1.3
7.0          3.4
11.0         5.2
13.0         6.0
15.0        14.4
17.5        21.4
20.0        27.4
22.5        50.9
25.0        49.3
27.5        47.5
30.0        51.5
35.0        36.5
40.0        27.9
50.0         9.4
60.0         4.2
```

TO EMPLOY THE USER DEFINED RESTRAINING 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:

```
27.5      47.5      1.0       4.0
                    (U)       (L)
```

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) = .1497565266967138E+02
C(2) = -.1144565861352237E+02
C(3) = .3049103763268548E+01
C(4) = -.3346122135351450E+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) = .8953993229386797E+03
C(2) = -.3853784676856376E+03
C(3) = .6639335801960442E+02
C(4) = -.5682948501274041E+01
C(5) = .2409284901711972E+00
C(6) = -.4025042180238581E-02

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

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

C(1) = .2080819121698872E+05
C(2) = -.6119991900236899E+04
C(3) = .7147141380302528E+03
C(4) = -.4142494450662980E+02
C(5) = .1192134278058383E+01
C(6) = -.1362678361482650E-01

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

INTERVAL NUMBER 4 WHICH BEGINS AT .2000000000000000E+02
AND ENDS AT .2359270706355915E+02 CONTAINS 28 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

C(1) = -.2671258659287486E+06
C(2) = .6180344279850274E+05
C(3) = -.5704535104277922E+04
C(4) = .2625284927906778E+03
C(5) = -.6022618197926448E+01
C(6) = .5509132822059026E-01

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .3398666798696013E+01.

INTERVAL NUMBER 5 WHICH BEGINS AT .2359270706355915E+02
AND ENDS AT .2738518411304983E+02 CONTAINS 21 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

C(1) = -.1013104278693695E+07
C(2) = .2002989828887461E+06
C(3) = -.1582334382597386E+05
C(4) = .6243710376956733E+03
C(5) = -.1230580964377970E+02
C(6) = .9691424185808195E-01

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

INTERVAL NUMBER 6 WHICH BEGINS AT .2738518411304983E+02
AND ENDS AT .3343698511741786E+02 CONTAINS 37 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

C(1) = .2974023810520601E+06
C(2) = -.4737982327455143E+05
C(3) = .3008544262538955E+04
C(4) = -.9516814694657342E+02
C(5) = .1499809050599829E+01
C(6) = -.9421877586332617E-02

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

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

C(1) = .6567620303936116E+05
C(2) = -.8417321059978451E+04
C(3) = .4305390572534816E+03
C(4) = -.1097600576167463E+02
C(5) = .1394356985419112E+00
C(6) = -.7061442077258562E-03

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

C(1) = .1984768817575497E+05
C(2) = -.2082281676169310E+04
C(3) = .8723916835701220E+02
C(4) = -.1819359280069307E+01
C(5) = .1886185858999134E-01
C(6) = -.7772177593868418E-04

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