184
10/15/50

Ln. 1842

ORNL/CSD-59

UCC-ND

MASTER

**NUCLEAR
DIVISION**

**A FORTRAN Subroutine
for Computing the Optimal
Estimate of $f(x)$**

P. W. Gaffney

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL/CSD-59
Dist. Category UC-32

P. W. Gaffney

Date Published - October 1980

Union Carbide Corporation, Nuclear Division
operating the
Oak Ridge Gaseous Diffusion Plant . Paducah Gaseous Diffusion Plant
Oak Ridge Y-12 Plant . Oak Ridge National Laboratory
under Contract No. W-7405-eng-26
for the
Department of Energy

TABLE OF CONTENTS

ACKNOWLEDGMENTS	v
ABSTRACT	1
1. INTRODUCTION	3
2. METHOD OF COMPUTATION	5
3. SAMPLE PROGRAM AND OUTPUT	11
4. DISCUSSION	17
5. QUALITY ASSURANCE AND SOFTWARE STANDARD	39
6. LOGICAL FLOWCHART	41
7. FORTRAN LISTING OF SUBROUTINE RANGE	43
REFERENCES	67

ACKNOWLEDGMENTS

I wish to thank Dr. J. K. Reid for diligently testing the FORTRAN subroutines in this package, using the WATFIV compiler on the IBM computer at Harwell. His results and comments have led to improvements in the code. I am also indebted to Dr. R. C. Ward and Dr. I. S. Duff for making suggestions that have improved the presentation of the paper. Finally, I wish to thank Teresa Craig for her excellent typing of the manuscript.

Support for this work was provided by the Computing Applications Department, Computer Sciences Division, and the Fusion Energy Division.

A FORTRAN SUBROUTINE FOR COMPUTING THE OPTIMAL ESTIMATE OF $f(x)$

P. W. Gaffney

ABSTRACT

We present a FORTRAN subroutine called RANGE that is designed to compute the optimal estimate of a function f given values of the function at n distinct points $x_1 < x_2 < \dots < x_n$ and given a bound on one of the derivatives of f . We denote this estimate by $\hat{f}$. It is optimal in the sense that the error $|f - \hat{f}|$ has the smallest possible error bound.

1. INTRODUCTION

In this paper we present a FORTRAN subroutine to compute the solution of the following problem.

Given values of an unknown function f at n distinct points: $x_1 < x_2 < \dots < x_{n-1} < x_n$, and given a finite bound on the k^{th} derivative of f :

$$\|f^{(k)}\|_{\infty} \equiv \text{Max. } |f^{(k)}(x)| < L < \infty, \quad 1 \leq k \leq n,$$

$$x_1 \leq x \leq x_n$$

determine the range of possible values of $f(\alpha)$, (and hence the optimal estimate of $f(\alpha)$), where α is any point in the interval $x_1 \leq x \leq x_n$.

Gaffney⁵ and Gaffney and Powell⁶ solved this problem. They proved that the CLOSEST POSSIBLE bounds on $f(\alpha)$ are given by the interval

$$\text{Min. } [u(\alpha), \ell(\alpha)] \leq f(\alpha) \leq \text{Max. } [u(\alpha), \ell(\alpha)] \quad (1.1)$$

where the quantities $u(\alpha)$ and $\ell(\alpha)$ are the values at $x = \alpha$ of two perfect splines of degree k which pass through the given function values. A method for computing the range (1.1) is described in a companion paper by Gaffney.⁷ Therefore, the purpose of this paper is to present a FORTRAN subroutine for computing the values $u(\alpha)$, $\ell(\alpha)$, and the estimate of $f(\alpha)$ whose error has the smallest possible bound, that is, the quantity

$$\Omega(\alpha, L) = \frac{(u(\alpha) + \ell(\alpha))}{2} \quad (1.2)$$

The name of this subroutine is RANGE.

It is important that prospective users of RANGE are aware of the amount of computation involved in computing the numbers $u(\alpha)$ and $l(\alpha)$. Therefore, in Section 2 we give a brief description of the method used by RANGE. In Section 3 we present a sample program for a situation where RANGE may be used. In Section 4 we discuss some aspects of practical approximation that we believe may be useful to prospective users of RANGE. We recommend that these users should read this section, in particular the conclusions at the end of the section, before incorporating subroutine RANGE in a FORTRAN program. In Section 5 we describe the standards that subroutine RANGE adheres to, and in Section 6 we present a flowchart which describes the way in which RANGE should be called. Finally, in Section 7 we present the FORTRAN listing of subroutine RANGE.

Subroutine RANGE was begun while the author was working at A.E.R.E. Harwell, under an Extramural Research Contract with Oxford University. The subroutine and this paper were completed at Oak Ridge.

2. METHOD OF COMPUTATION

In this section we give a brief description of the algorithm used by RANGE. To do this we first recall, from reference 7, the solution of the optimal estimation problem and we review the properties of the functions u and l that provide the bounds (1.1).

Given values $f_1, \dots, f_n$ of a function f at the points

$$x_1 < x_2 < \dots < x_{n-1} < x_n$$

and given that the inequality

$$\text{Max. } |f^{(k)}(x)| < L < \infty, \quad 1 \leq k \leq n$$

$$x_1 \leq x \leq x_n$$

is satisfied, where the value of L is greater than the least value of

$$\text{Max. } |f^{(k)}(x)|$$

$$x_1 \leq x \leq x_n$$

that is consistent with the function values $f_1, \dots, f_n$, then the closest possible bounds on $f(x)$, for any x in the range $x_1 \leq x \leq x_n$, are given by the inequalities

$$\text{Min. } [u(x), l(x)] \leq f(x) \leq \text{Max. } [u(x), l(x)]. \quad (2.1)$$

The functions u and l in expression (2.1) are perfect splines of

degree k , they each have $n-k$ knots, and they each satisfy the interpolation conditions

$$u(x_i) = l(x_i) = f(x_i), \quad i = 1, \dots, n. \quad (2.2)$$

Furthermore, the k^{th} derivative of u satisfies the equation

$$u^{(k)}(x) = \begin{cases} L & x_1 < x < \eta_1 \\ (-1)^i L & \eta_i < x < \eta_{i+1}, \quad i=1, \dots, n-k-1 \\ (-1)^{n-k} L & \eta_{n-k} < x < x_n \end{cases} \quad (2.3)$$

and the k^{th} derivative of l satisfies the equation

$$l^{(k)}(x) = \begin{cases} -L & x_1 < x < \xi_1 \\ (-1)^{i+1} L & \xi_i < x < \xi_{i+1}, \quad i=1, \dots, n-k-1 \\ (-1)^{n-k+1} L & \xi_{n-k} < x < x_n \end{cases} \quad (2.4)$$

We calculate the knots $\{\eta_i\}$ and $\{\xi_i\}$ by solving the systems of equations

$$\sum_{j=0}^{n-k} (-1)^j \int_{\eta_j}^{\eta_{j+1}} M_{k,1}(x) dx - L^{-1}(k-1)! f(x_1, \dots, x_{1+k}) = 0, \quad i=1, \dots, n-k \quad (2.5)$$

and

$$\sum_{j=0}^{n-k} (-1)^{j+1} \int_{\xi_j}^{\xi_{j+1}} M_{k,1}(x) dx - L^{-1}(k-1)! f(x_1, \dots, x_{1+k}) = 0, \quad i=1, \dots, n-k \quad (2.6)$$

where $\eta_0 = \xi_0 = x_1$, $\eta_{n-k+1} = \xi_{n-k+1} = x_n$, $M_{k,i}(x)$ is a B-spline of degree $k-1$ with knots $x_1, \dots, x_{i+k}$, and $f(x_1, \dots, x_{i+k})$ is the k^{th} divided difference of f based on the points $x_1, \dots, x_{i+k}$.

When $k = 1$, equations (2.5) and (2.6) are linear. In this case it is straightforward to show that the knots $\{\eta_i\}$ and $\{\xi_i\}$ have the values

$$\eta_i = \frac{(-1)^{i+1}}{2L} (f(x_{i+1}) - f(x_i)) + \frac{(x_i + x_{i+1})}{2}, \quad i=1, \dots, n-1 \quad (2.7)$$

and

$$\xi_i = \frac{(-1)^i}{2L} (f(x_{i+1}) - f(x_i)) + \frac{(x_i + x_{i+1})}{2}, \quad i=1, \dots, n-1. \quad (2.8)$$

When the value of k is greater than 1, equations (2.5) and (2.6) are *non-linear*. Therefore, we solve them using a continuation method together with Newton iteration. A description of this technique is given by Gaffney.⁷

In order to compute the bounds (2.1), at a given value of x , say $x = \alpha$, we use the formulas (see Gaffney)⁷

$$UP \equiv \text{Max. } [u(\alpha), l(\alpha)] = \begin{cases} P_{k-1}(\alpha) + \frac{\pi(\alpha)}{(k-1)!} \text{ CUP, when } \pi(\alpha) > 0 \\ P_{k-1}(\alpha) + \frac{\pi(\alpha)}{(k-1)!} \text{ CLOW, otherwise} \end{cases} \quad (2.9)$$

and

$$\text{LOW} \equiv \text{Min. } [u(\alpha), l(\alpha)] = \begin{cases} P_{k-1}(\alpha) + \frac{\pi(\alpha)}{(k-1)!} \text{CLOW}, & \text{when } \pi(\alpha) > 0 \\ P_{k-1}(\alpha) + \frac{\pi(\alpha)}{(k-1)!} \text{CUP}, & \text{otherwise} \end{cases} \quad (2.10)$$

where

$$P_{k-1}(\alpha) = \sum_{i=1}^k f(\bar{x}_i) \prod_{\substack{j=1 \\ j \neq i}}^k \frac{\alpha - \bar{x}_j}{\bar{x}_i - \bar{x}_j}, \quad (2.11)$$

$$\pi(\alpha) = (\alpha - \bar{x}_1)(\alpha - \bar{x}_2) \dots (\alpha - \bar{x}_{k-1})(\alpha - \bar{x}_k), \quad (2.12)$$

$$\text{CLOW} = \text{Min.} \left[\int_{x_1}^{x_n} M_{\alpha}(x) u^{(k)}(x) dx, \int_{x_1}^{x_n} M_{\alpha}(x) l^{(k)}(x) dx \right], \quad (2.13)$$

$$\text{CUP} = \text{Max.} \left[\int_{x_1}^{x_n} M_{\alpha}(x) u^{(k)}(x) dx, \int_{x_1}^{x_n} M_{\alpha}(x) l^{(k)}(x) dx \right]. \quad (2.14)$$

The quantities $\bar{x}_1, \dots, \bar{x}_k$ are k of the data points which are closest to α , and $M_{\alpha}(x)$ is a B-spline of degree $k-1$ with the $k+1$ knots $\{\bar{x}_1, \dots, \bar{x}_k, \alpha\}$ arranged in ascending order.

Finally, the optimal estimate of $f(\alpha)$ is computed from the expression

$$\hat{f}(\alpha, L) = \frac{(\text{UP} + \text{LOW})}{2}. \quad (2.15)$$

Note that the smallest value of the error

$$|f(\alpha) - \Omega(\alpha, L)|$$

is zero, and that the maximum value which it can attain is the quantity

$$\frac{|UP - LOW|}{2} \quad (2.16)$$

The main calculation involved in computing the bounds (2.1) is the solution, when $k > 2$, of the non-linear equations (2.5) and (2.6) for the knots of u and l . Once this has been accomplished, the remaining calculations, namely (2.9) through (2.15), proceed rapidly. Moreover, since the knots of u and l do not depend on the point of interpolation, the computation of $u(\alpha)$, and $l(\alpha)$ for a sequence of values of α is fast.

In Section 6 we present a flowchart which describes the calculation outlined above and also provides a recommended sequence of computation.

3. SAMPLE PROGRAM AND OUTPUT

In this section we give an example of a situation where subroutine RANGE may be used.

We suppose that we are given the data of Table 1, and the bound

$$\text{Max. } |f^{(3)}(x)| < 8000.0, \quad (3.1)$$

$$-5.0 < x < 5.0$$

and that we wish to approximate the unknown function f by a function which passes through all of the values $f(x_i)$.

Table 1: Sample Data		
i	x_i	$f(x_i)$
1	-5.0	0.301599
2	-3.0	0.304435
3	-1.2	0.327397
4	-1.0	0.339216
5	-0.6	0.405263
6	-0.4	0.522222
7	-0.2	0.966667
8	0.0	2.300000
9	0.2	0.966667
10	0.4	0.522222
11	0.8	0.360606
12	1.0	0.339216
13	1.4	0.320202
14	3.2	0.303899
15	4.4	0.302064
16	5.0	0.301599

In order to obtain an approximation, it is sensible to use a formula that takes account of all of the given information, namely the data of Table 1 and the bound (3.1). Therefore, it is appropriate to use subroutine RANGE to compute the optimal estimate $\Omega(x, 8000)$ of $f(x)$.

The FORTRAN code for computing the optimal estimate $\Omega(x, 8000)$ for a sequence of values of x might be as shown in Figure 1. The results of passing a smooth curve through the values, $\Omega(xt_i, 8000)$, $l(xt_i, 9000)$, and $u(xt_i, 8000)$, computed by this code are shown in Figure 2. Specifically, Figure 2(b) shows the range of possible values of $f(x)$. That is, every function which passes through the function values given in Table 1 and which also satisfies the inequality (3.1), lies between the two curves shown in 2(b). Thus, the optimal estimate of $f(x)$ is simply the average of these two curves. It is shown in Figure 2(a).

An estimate of the performance of subroutine RANGE, on a typical problem, may be obtained by examining the CPU-time taken for the above example. To obtain an unbiased estimate of this time, the statements labeled MAN 470 through MAN 540 in Figure 1 were executed 1000 times and the average CPU-time taken by RANGE was calculated. This experiment, which was performed on both a DEC-10 computer and a CRAY-1 computer, was repeated on a number of different occasions. The resulting average CPU-time is shown in Table 2.

Table 2: Average CPU-Time in Seconds for Sample Program		
CODE	DEC-10	CRAY-1
RANGE	0.34	.038
TB07A/TG03A	0.11	.012

For the purposes of comparison, the table also shows the average CPU-time taken by the codes TB07A/TG03A⁸ which implement the optimal interpolation method described in reference 10. The reason why RANGE is approximately three times slower than this method is because RANGE computes the values of three functions, namely, ℓ , u , and Ω . This is in contrast to the optimal interpolation method which computes only one function.

REAL X(16), F(16), WK(200), ETA(13), PSI(13)	MAN 10
REAL XT(101), L(101), U(101), OMEGA(101)	MAN 20
INTEGER IL(16)	MAN 30
C	MAN 40
C ASSEMBLE THE DATA FROM TABLE 1.	MAN 50
C	MAN 60
DATA X /-5.,-3.,-1.2,-1.,-.6,-.4,-.2,0.,.2,.4,.8,1.,1.4,	MAN 70
• 3.2,4.4,5.0/	MAN 80
DATA F /-.301599,-.304435,-.327397,-.339216,-.405263,-.522222,	MAN 90
• .966667,2.3,-.966667,-.522222,-.360606,-.339216,-.320202,	MAN 100
• .303899,-.302064,-.301599/	MAN 110
C	MAN 120
C SET THE NUMBER OF DATA POINTS	MAN 130
C	MAN 140
N = 16	MAN 150
C	MAN 160
C SET THE VALUE OF K	MAN 170
C	MAN 180
K = 3	MAN 190
C	MAN 200
C SET THE LENGTH OF THE WORKSPACE ARRAY WK.	MAN 210
C NOTE THAT LWK MUST BE AT LEAST THE VALUE	MAN 220
C $5*N-2*K+1+(N-K)*MIN(K,N-K)$	MAN 230
C	MAN 240
LWK = 200	MAN 250
C	MAN 260
C SET THE VALUE OF THE BOUND ON THE KTH. DERIVATIVE	MAN 270
C OF F(X).	MAN 280
C	MAN 290
AL = 8000.0	MAN 300
C	MAN 310
C SET THE LENGTH OF THE ARRAYS ETA AND PSI.	MAN 320
C NOTE THAT LEP MUST BE AT LEAST N-K.	MAN 330
C	MAN 340
LEP = 13	MAN 350
C	MAN 360
C COMPUTE THE OPTIMAL ESTIMATE OF F AT 101	MAN 370
C EQUALLY SPACED VALUES OF X IN THE INTERVAL	MAN 380
C -5.0 .LE. X .LE. 5.0.	MAN 390
C	MAN 400
C NOTE THAT IN THE CALL TO RANGE THE VALUE OF	MAN 410
C THE VARIABLE IAG IS SET TO THE VALUE OF THE	MAN 420
C DO LOOP VARIABLE I. IN THIS WAY THE SUBSEQUENT	MAN 430
C COMPUTATION OF THE OPTIMAL ESTIMATE FOR I.GE.2	MAN 440
C IS MUCH FASTER.	MAN 450
C	MAN 460
DO 10 I=1,101	MAN 470
XT(I) = X(I) + 0.1*FLOAT(I-1)	MAN 480
CALL RANGE(I, N, X, F, K, WK, LWK, AL, XT(I), IL,	MAN 490
• LEP, ETA, PSI, L(I), U(I), OMEGA(I), IFAIL)	MAN 500
IF (IFAIL.EQ.0) GO TO 10	MAN 510
WRITE (6,99999) IFAIL	MAN 520
GO TO 20	MAN 530
10 CONTINUE	MAN 540
20 STOP	MAN 550
99999 FORMAT (3X, 8HIFAIL = , I4)	MAN 560
END	MAN 570

Figure 1: Sample Program

Figure 2(a) The optimal estimate $\Omega(x, 8000)$ of f , (b) the range of possible values for the data of Table 1.

۱۵

[illegible]

4. DISCUSSION

In this section we wish to show the types of approximation that may be obtained by different choices of the parameters L and k . The reason for doing this is that it is unusual for users to know, in advance, a bound on one of the derivatives of the function being approximated. Thus, it is important that users are aware of the effect on the approximation of an incorrect choice of L and/or k . In order to show these effects, we consider the example used in the sample program of Section 3. That is, we are given the data of Table 1 and we wish to obtain the optimal estimate of f . However, we now assume that a bound on one of the derivatives of f is not readily available and proceed to show how to obtain a lower bound on the k^{th} derivative of f , for k in the range $1 \leq k \leq n$. To do this, we require the following important result which was first given by Curry and Schoenberg in 1947 (see reference 1).

The k^{th} divided difference, $k \geq 1$, of any function $f(x)$, whose $(k-1)$ st derivative is continuous and whose k^{th} derivative may be piecewise continuous, can be written as

$$f(x_i, \dots, x_{i+k}) = \frac{1}{(k-1)!} \int_{x_i}^{x_{i+k}} M_{k,i}(x) f^{(k)}(x) dx \quad (4.1)$$

where $M_{k,i}(x)$ is a B-spline of degree $k-1$ with knots at the points

$$x_i < x_{i+1} < \dots < x_{i+k-1} < x_{i+k}.$$

We note that if $f(x) = x^k$, then (4.1) gives the value

$$\int_{x_i}^{x_{i+k}} H_{k,i}(x) dx = \frac{1}{k} \quad (4.2)$$

Therefore, it follows from (4.1), (4.2) and the fact that $H_{k,i}(x) > 0$, that the bound

$$k! |f(x_i, \dots, x_{i+k})| < \text{Max. } |f^{(k)}(x)| \quad (4.3)$$

$$x_i < x < x_{i+k}$$

holds throughout the range of values of i . Consequently, the value of the bound L must satisfy the inequality

$$k! \text{Max. } |f(x_i, \dots, x_{i+k})| < \|f^{(k)}\|_{\infty} < L. \quad (4.4)$$

$$i$$

In practice, if the user chooses a value of L that does not satisfy (4.4), then subroutine RANGE prints a message to this effect and gives the value of the left inequality of (4.4). In this way, the user can choose a more sensible value of L . As an alternative to this procedure an estimate of the least value of L , that is consistent with the function values, may be obtained directly from the divided differences $f(x_i, \dots, x_{i+k})$, $i = 1, \dots, n-k$, which may be computed, for example by the subroutine shown in Figure 3.

SUBROUTINE DIVDIF(N, X, F, K, DIFF)	DIV 10
C	DIV 20
C PURPOSE	DIV 30
C	DIV 40
C	DIV 50
C THIS SUBROUTINE COMPUTES THE KTH. DIVIDED DIFFERENCE OF THE	DIV 60
C FUNCTION VALUES F(X(1)),...,F(X(N)). THE RESULT IS STORED IN	DIV 70
C DIFF(I) I=1,...,N-K.	DIV 80
C	DIV 90
C	DIV 100
REAL X(N), F(N), DIFF(N)	DIV 110
DO 10 I=1,N	DIV 120
DIFF(I) = F(I)	DIV 130
10 CONTINUE	DIV 140
DO 30 J=1,K	DIV 150
C J DENOTES THE JTH DIVIDED DIFFERENCE OF F	DIV 160
MMJ = N - J	DIV 170
DO 20 I=1,MMJ	DIV 180
C DIFF(I) IS THE DIVIDED DIFFERENCE F(X(I)),...,X(I-J-1))	DIV 190
S1 = DIFF(I+1) - DIFF(I)	DIV 200
IPJ = I + J	DIV 210
S1 = S1/(X(IPJ)-X(I))	DIV 220
DIFF(I) = S1	DIV 230
20 CONTINUE	DIV 240
30 CONTINUE	DIV 250
RETURN	DIV 260
END	DIV 270

Figure 3: A Sample Subroutine to Compute Divided Differences

For the data of Table 1 the divided differences through order 5 are shown in Table 3.

Table 3: The divided differences, through order 5,
of the data of Table 1.

i	x_i	$f(x_i)$	$f(x_i, x_{i+1})$	$f(x_i, \dots, x_{i+2})$	$f(x_i, \dots, x_{i+3})$	$f(x_i, \dots, x_{i+4})$	$f(x_i, \dots, x_{i+5})$
1	-5.0	0.301599					
2	-3.0	0.304435	0.001418				
3	-1.2	0.327397	0.012757	0.002984			
4	-1.0	0.339216	0.059092	0.023168	0.005046		
5	-0.6	0.405263	0.165119	0.176711	0.063976	0.013393	0.046374
6	-0.4	0.522222	0.584795	0.699461	0.643438	0.226716	1.200886
7	-0.2	0.966667	2.222222	4.093567	4.242633	3.589195	3.220066
8	0.0	2.300000	6.666667	11.111111	11.695907	7.453274	-95.554792
9	0.2	0.966667	-6.666667	-33.333334	-74.074076	-107.212480	292.397670
10	0.4	0.522222	-2.222222	11.111111	74.074076	185.185190	-224.466900
11	0.8	0.360606	-0.404040	3.030303	-10.101011	-84.175086	75.922626
12	1.0	0.339216	-0.106952	0.495148	-3.168944	6.932066	-3.300984
13	1.4	0.320202	-0.047534	0.099030	-0.396118	2.310689	-0.727117
14	3.2	0.303899	-0.009057	0.017489	-0.033975	0.129337	-0.030281
15	4.4	0.302064	-0.001529	0.002510	-0.004406	0.008214	-0.001728
16	5.0	0.301599	-0.000775	0.000419	-0.000581	0.000956	

From these differences, we compute the values shown in Table 4 for the lower bound on L for $k = 1, \dots, 5$. The table also gives an approximate bound on the least value of L that is consistent with the function values of Table 1. This approximate bound is obtained, in an iterative way, by computing the smallest value of L for which equations (2.5) and (2.6) have a solution.

Table 4: The lower bound on L for some values of k		
k	$k! \text{ Max. } f(x_i, \dots, x_{i+k}) $ $1 \leq i \leq 16 - k$	L must be greater than:
1	6.666667	6.666667
2	66.666668	70.0
3	444.444456	714.0
4	4444.444560	7600.0
5	35087.720400	79000.0

The last column of Table 4 shows that the bound obtained from inequality (4.4) is, in this case, a gross underestimate for the least value of L . We have found that the value obtained from the left inequality of (4.4) is generally a poor estimate of the value of L that should be used to obtain a good approximation to f . Instead, it only provides a first approximation from which a more sensible value of L can be determined. The question now arises of how to obtain a more sensible value of L in the absence of any further information about f . Unfortunately, there is no pleasing answer to this question as the value of L has to be obtained by trial and error. However, we will

show, by an example, that it is far preferable to choose a large value of L than to choose too small a value.

To show the effect of choosing too small or too large a value for L , we consider the cases when L is allowed to take values at the extreme ends of the range

Least value consistent with the data. $< L < \infty$, (4.5)
--

and k has the value 3.

The effect of choosing too small a value for L

Figure 4 shows the optimal estimate of the data of Table 1 when $L = 714.8739$, which is a value very close to the least value of L . In this case the resulting approximation is sometimes called the "BEST" interpolant (see reference 2). Figure 4 shows that this interpolant is a very poor approximation to the data of Table 1. For example, compare it with the good approximation shown in Figure 2(a).

ORNL-DWG 80-2152 FED

Figure 4 The optimal estimate $\Omega(x, 714.8739)$ for the data of Table 1 when $k = 3$.

The reason for this poor approximation can be seen in Figure 5, where we have shown the range of possible values of f when $L = 714.8739$. Thus, for instance, the figure shows that the oscillations in $\Omega(x, 714.8739)$, between the first three and the last four data points, are due to the large differences in the magnitudes of the functions l and u in these regions. For example, compare Figure 5 with Figure 2(b). These large differences are the result of imposing the unrealistic constraint that the third derivative of f be uniformly "small" throughout the interval $x_1 < x < x_{16}$, or equivalently that the unknown function f is a quadratic polynomial!

ORNL-DWG 80-2150 FED

Figure 5(a) shows the range of possible values for the data of Table I when $L = 714.8739$, and (b) shows the irregular behavior in the region indicated in (a).

Now, since all functions which interpolate f at the values in Table 1 and which satisfy the bound

$$\begin{aligned} \text{Max. } |f^{(3)}(x)| &< 714.8739, \\ x_1 &\leq x \leq x_{16} \end{aligned} \tag{4.6}$$

lie between the two curves shown in Figure 5(a), it follows that the so-called BEST interpolant also lies between these two curves. Therefore, in this case, the BEST interpolant is in fact a very poor interpolant.

In general, we do not recommend using a value of L close to the least value of $\|f^{(k)}\|_\infty$ that is consistent with the given function values. Rather, we recommend choosing a large value of L .

The effect of choosing a large value of L

When the value of L is large, compared to the value at the left end of the interval (4.5), the optimal estimate usually provides a good piecewise polynomial approximation to the data. In fact, as L tends to infinity Gaffney and Powell⁶ proved that the optimal estimate converges to the unique spline function, $\bar{\Omega}$, of degree $k - 1$ which passes through the given function values and which has the $n - k$ knots that are the solution of the equations

$$\sum_{j=0}^{n-k} (-1)^j \int_{n_j}^{n_{j+1}} M_{k,i}(x) dx = 0, \quad i=1, \dots, n-k. \quad (4.7)$$

(Compare with equations (2.5) - (2.6).)

We note that this spline function, which does not depend on the value L, is called the optimal interpolation formula. We recommend the method described by Gaffney⁴ for computing $\bar{\Omega}$. For the data of Table 1, the optimal interpolant when $k = 3$ is shown in Figure 6.

ORNL-DWG 80-2153 FED

Figure 6 The optimal interpolant, $\bar{Q}$, for the data of Table 1 when $k = 3$.

The figure shows that $\bar{\Omega}$ is not too different from the approximation shown in Figure 2. However, in the limited number of test examples that we have run, we have found that the optimal estimate $\Omega(x, L)$, for a sensible value of L , usually provides a more accurate approximation than the optimal interpolation formula $\bar{\Omega}(x)$. To see that this is true in the present example, we first note that the data of Table 1 is obtained from the function

$$f(x) = 0.3 + (0.5 + 25x^2)^{-1}. \quad (4.8)$$

Thus, we can compute the error functions

$$E_1(x) = |f(x) - \bar{\Omega}(x)| \quad (4.9)$$

and

$$E_2(x, L) = |f(x) - \Omega(x, L)|, \quad (4.10)$$

at any value of x . We computed these functions at 501 equally spaced values of x in the range $[-5, 5]$. The maximum value attained by $E_1(x)$ is 0.17192172. Furthermore, the maximum value attained by $E_2(x, L)$, for a sequence of values of L , is shown in Figure 7.

ORNL-DWG 80-2451

FED

Figure 7 The maximum error in $\Omega(x, L)$.

The figure shows that the error $E_2(x, L)$ increases when the value of L approaches its lower limit. Moreover, although it is not apparent from Figure 7, the inequality

$$\text{Max. } E_2(x, L) < 0.17192172 \quad (4.11)$$

is valid when L is greater than or equal to 8000, and the minimum value of the maximum of E_2 occurs when L is equal to 11000.

Recommendation
<p style="text-align: center;">To obtain an accurate approximation $\Omega(x, L)$, we recommend choosing L very much larger than the least value of L that is consistent with the given data.</p> <p style="text-align: center;">To ensure that the result is acceptable, plot the approximation $\Omega(x, L)$.</p>

The value of k

We now consider the problem of choosing a sensible value for k . In general, the value of k should be very much smaller than the number of data points n . This ensures that the resulting approximation $\Omega(x, L)$ is composed of a large number of polynomial pieces. Thus, in this case, we would expect to achieve all the benefits of piecewise polynomial approximation. A value of k less than 8 should usually be sufficient. In fact, $k = 2, 3$, or 4 will suffice

for most practical problems. Whatever value is chosen for k , it is important that the user examines the approximation $\Omega(x, L)$, preferably in graphical form. In this way any unexpected behavior will be discovered immediately.

The effect of choosing too large a value of k can be seen in Figure 8(a) and (b). This figure shows the functions $\Omega(x, L)$ for the data of Table 1 when $k = 4$ and 5. The large oscillations are due entirely to the fact that these values of k are too large for the data of Table 1. The corresponding function when $k = 3$ is almost identical to the one shown in Figure 6.

ORNL-DWG 80-2155 FED

Figure 8 (a) and (b) are classic examples of choosing too large a value for the degree of the interpolation formula.

Recommendation

<p>The value of k should satisfy the inequality</p>
--

$k \ll n,$

<p>and only on sound numerical grounds should k be chosen larger than 8.</p>

<p>To ensure that the result is acceptable, plot the resulting approximation $\Omega(x, L)$.</p>

Conclusion

In this section we have shown the types of approximation that may be obtained by different choices of the parameters L and k . In general it is sensible to use RANGE when function values and a bound L on the k th derivative of f are given.

If only function values are provided, then it is possible to obtain, by trial and error, a bound on one of the derivatives of f . However, in this case extreme care should be exercised in the choice of this bound and in the choice of k . In practice we have found that a large value of L and a small value of k are usually sufficient to provide an acceptable approximation to f . In this context "large" is measured relative to the least value of $\text{Max}_{x_1 \leq x \leq x_n} |f^{(k)}(x)|$ that is consistent with the given function values. Since this quantity is in general unknown, a number of iterations may be required to obtain a

sensible value for L . Therefore, when a bound on one of the derivatives of f is not readily available, we do not recommend using RANGE. Instead we recommend using the optimal interpolation formula $\bar{O}^4$. In this case the only parameter that has to be chosen is k , and it is sensible to choose a value of k that is much smaller than the number n of function values.

The question of which interpolation formula to use in general will be answered in more detail in a forthcoming paper by Gaffney.⁹

5. QUALITY ASSURANCE AND SOFTWARE STANDARD

Subroutine RANGE was written to conform to the FORTRAN IV ANSI standard 1966, and it has been verified using the Bell Telephone Laboratories FORTRAN Verifier, PFORT.¹¹

The subroutine has been extensively tested on a wide variety of test problems, and it has been analyzed for errors using DAVE.¹⁰

To make the subroutine easier to read, it has been reformatted using POLISH.³

6. LOGICAL FLOWCHART

In this section we present a flowchart which describes the way in which subroutine RANGE should be called. Prospective users are advised to consult the flowchart before incorporating subroutine RANGE into a FORTRAN program. In particular we note that, if the subroutine is to be called repeatedly for a sequence of values of α , then the variable IAC should be reset to a value greater than 1 after the first call to RANGE. In this way, the knots $\{\eta_i\}$ and $\{\xi_i\}$ are computed only once and the remaining calculation is fast.

2. THE FORTRAN LISTING OF SUBROUTINE RANGE

```

SUBROUTINE RANGE(IAG, N, X, F, K, WK, LMK, AL, ALPHA, IL,
  LEP, ETA, PSI, LOW, UP, OMEGA, IFAIL)
C .....
C          PURPOSE
C .....
C      GIVEN VALUES OF A FUNCTION F(X) AT N DISTINCT POINTS
C      X(1),LT.X(2),.....,LT.X(N) AND GIVEN A FINITE BOUND,AL,
C      ON THE KTH. DERIVATIVE OF F(X), 1.LE.K.LE.N,
C      THIS SUBROUTINE COMPUTES THE CLOSEST POSSIBLE BOUNDS
C      ON F(ALPHA), WHERE ALPHA IS A SPECIFIED VALUE OF X.
C      THE SUBROUTINE ALSO PROVIDES THE ESTIMATE, OMEGA, OF
C      F(ALPHA) WHOSE ERROR HAS THE SMALLEST POSSIBLE BOUND.
C .....
C .....
C      **** I N P U T ****
C
C      IAG      IS AN INTEGER VARIABLE WHICH MUST BE SET TO THE VALUE +1
C               AT THE FIRST CALL OF THE SUBROUTINE. THE SUBROUTINE MAY
C               BE RE-ENTERED WITH A DIFFERENT VALUE OF ALPHA. IN THIS
C               CASE IF THE VALUE OF IAG IS GREATER THAN 1, AND THE
C               REMAINING PARAMETERS ARE UNALTERED, THEN EXECUTION IS
C               MUCH FASTER. NOTE THAT THE CODE DOES NOT CHECK THAT THE
C               REMAINING PARAMETERS ARE UNALTERED.
C               THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.
C
C      N        IS AN INTEGER VARIABLE WHICH MUST BE SET TO THE NUMBER
C               OF DATA POINTS X(I), I=1,.....,N. RESTRICTION: N.GE.2
C               THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.
C
C      X        IS A REAL ARRAY OF LENGTH AT LEAST N WHICH MUST BE
C               SET TO THE VALUES OF THE DATA POINTS X(I), I=1,.....,N.
C               RESTRICTION: THE DATA POINTS MUST BE DISTINCT AND
C               THEY MUST BE IN ASCENDING ORDER.
C               THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.
C
C      F        IS A REAL ARRAY OF LENGTH AT LEAST N WHICH MUST BE
C               SET TO THE FUNCTION VALUES F(X(1)),.....,F(X(N)).
C               THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.
C
C      K        IS AN INTEGER VARIABLE WHICH MUST BE SET TO THE ORDER
C               OF THE DERIVATIVE OF F(X) FOR WHICH A FINITE BOUND IS
C               GIVEN. THE VALUE OF K SHOULD BE VERY MUCH SMALLER THAN
C               THE VALUE OF N. IN FACT WE RECOMMEND THAT ONLY IN
C               EXCEPTIONAL CIRCUMSTANCES AND THEN ONLY ON SOUND
C               NUMERICAL GROUNDS, SHOULD THE VALUE OF K BE GREATER
C               THAN 8. RESTRICTION: 1.LE.K.LE.N
C               THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.
C
C      WK       IS A REAL ARRAY OF LENGTH AT LEAST:

```

```

RAN 10
RAN 20
RAN 30
RAN 40
RAN 50
RAN 60
RAN 70
RAN 80
RAN 90
RAN 100
RAN 110
RAN 120
RAN 130
RAN 140
RAN 150
RAN 160
RAN 170
RAN 180
RAN 190
RAN 200
RAN 210
RAN 220
RAN 230
RAN 240
RAN 250
RAN 260
RAN 270
RAN 280
RAN 290
RAN 300
RAN 310
RAN 320
RAN 330
RAN 340
RAN 350
RAN 360
RAN 370
RAN 380
RAN 390
RAN 400
RAN 410
RAN 420
RAN 430
RAN 440
RAN 450
RAN 460
RAN 470
RAN 480
RAN 490
RAN 500
RAN 510
RAN 520
RAN 530

```

C		$5^*N-2^*K+1+(N-K)^*MIN(K,N-K)$	RAN	540
C			RAN	550
C		WHICH IS USED AS WORKSPACE.	RAN	560
C			RAN	570
C	LWK	IS AN INTEGER VARIABLE WHICH MUST BE SET TO THE	RAN	580
C		LENGTH OF WK.	RAN	590
C		RESTRICTION: $LWK.GE.5^*N-2^*K+1+(N-K)^*MIN(K,N-K)$.	RAN	600
C		THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.	RAN	610
C			RAN	620
C	AL	IS A REAL VARIABLE WHICH MUST BE SET TO THE VALUE	RAN	630
C		L, OF THE FINITE BOUND ON THE KTH. DERIVATIVE OF	RAN	640
C		F(X).	RAN	650
C		RESTRICTION: L MUST BE GREATER THAN THE LEAST VALUE	RAN	660
C		OF THE MAXIMUM ABSOLUTE VALUE OF THE KTH. DERIVATIVE	RAN	670
C		OF F(X) THAT IS CONSISTENT WITH THE GIVEN FUNCTION	RAN	680
C		VALUES F(X(1)),...,F(X(N)). IN PARTICULAR L MUST	RAN	690
C		SATISFY THE INEQUALITIES	RAN	700
C			RAN	710
C		L .GT. 0	RAN	720
C		AND	RAN	730
C		L .GE. FACTORIAL(K)*ABS(F(X(I)),...,X(I+K))	RAN	740
C		I=1,...,N-K,	RAN	750
C			RAN	760
C		WHERE F(X(I)),...,X(I+K)) DENOTES THE KTH. DIVIDED	RAN	770
C		DIFFERENCE OF F(X) BASED ON THE POINTS X(I),...,X(I+K).	RAN	780
C		THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.	RAN	790
C			RAN	800
C	ALPHA	IS A REAL VARIABLE WHICH MUST BE SET TO THE VALUE	RAN	810
C		OF THE ARGUMENT X AT WHICH THE RANGE OF POSSIBLE	RAN	820
C		VALUES OF F(X) IS COMPUTED.	RAN	830
C		RESTRICTION: $X(1).LE.ALPHA$	RAN	840
C		THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.	RAN	850
C			RAN	860
C	IL	IS AN INTEGER ARRAY OF LENGTH AT LEAST N. IT IS USED	RAN	870
C		AS WORKSPACE.	RAN	880
C			RAN	890
C	LEP	IS AN INTEGER VARIABLE WHICH MUST BE SET TO THE LESSER	RAN	900
C		LENGTH OF ARRAYS ETA AND PSI. RESTRICTION: $LEP.GE.N-K$.	RAN	910
C		THIS ARGUMENT IS NOT ALTERED BY THE SUBROUTINE.	RAN	920
C			RAN	930
C			RAN	940
C	**** O U T P U T ****		RAN	950
C			RAN	960
C	ETA	IS A REAL ARRAY OF LENGTH AT LEAST N-K. ON EXIT	RAN	970
C		FROM THE SUBROUTINE ETA CONTAINS THE KNOTS	RAN	980
C		OF THE PERFECT SPLINE U(X).	RAN	990
C			RAN	1000
C	PSI	IS A REAL ARRAY OF LENGTH AT LEAST N-K. ON EXIT	RAN	1010
C		FROM THE SUBROUTINE PSI CONTAINS THE KNOTS OF THE	RAN	1020
C		PERFECT SPLINE L(X).	RAN	1030
C			RAN	1040
C	LOW	IS A REAL VARIABLE. ON EXIT FROM THE SUBROUTINE	RAN	1050
C		LOW IS SET TO THE GREATEST LOWER BOUND OF F(ALPHA).	RAN	1060
C			RAN	1070
C	UP	IS A REAL VARIABLE. ON EXIT FROM THE SUBROUTINE	RAN	1080

C	UP IS SET TO THE LEAST UPPER BOUND OF F(ALPHA).	RAN 1090
C		RAN 1100
C	OMEGA IS A REAL VARIABLE. ON EXIT FROM THE SUBROUTINE	RAN 1110
C	OMEGA IS SET TO THE OPTIMAL ESTIMATE OF F(ALPHA).	RAN 1120
C	THE SMALLEST VALUE OF THE ERROR OF THIS ESTIMATE	RAN 1130
C	OF F(ALPHA) IS ZERO, AND THE MAXIMUM VALUE WHICH	RAN 1140
C	IT CAN ATTAIN IS THE QUANTITY: $0.5 \cdot \text{ABS}(UP-LOW)$.	RAN 1150
C		RAN 1160
C	IFAIL IS AN ERROR RETURN FLAG. ON EXIT FROM THE SUBROUTINE	RAN 1170
C	IT HAS ONE OF THE FOLLOWING VALUES:	RAN 1180
C		RAN 1190
C	0 SUCCESSFUL ENTRY	RAN 1200
C	1 N .LT. 2	RAN 1210
C	2 K .LT. 1 OR K .GT. N	RAN 1220
C	3 X(I) .GE. X(I+1) FOR SOME I	RAN 1230
C	4 L .LT. FACTORIAL(K)*ABS(F(X(1),...,X(I+K))	RAN 1240
C	FOR SOME I	RAN 1250
C	5 MORE THAN IMAX ITERATIONS NEEDED TO CALCULATE	RAN 1260
C	THE KNOTS ETA AND/OR PSI.	RAN 1270
C	6 THE METHOD USED TO CALCULATE THE KNOTS ETA,	RAN 1280
C	AND PSI, FAILED. THIS IS USUALLY BECAUSE L	RAN 1290
C	IS TOO SMALL OR THE VALUE OF K IS TOO LARGE.	RAN 1300
C	7 L .LE. 0	RAN 1310
C	8 THE ARRAY WK IS TOO SMALL	RAN 1320
C	9 THE ARRAYS ETA AND PSI ARE TOO SMALL	RAN 1330
C	10 ALPHA .LT. X(1)	RAN 1340
C		RAN 1350
C		RAN 1360
C	**** ADDITIONAL ROUTINES ****	RAN 1370
C		RAN 1380
C	THE FOLLOWING ADDITIONAL ROUTINES ARE SUPPLIED WITH RANGE:	RAN 1390
C		RAN 1400
C	SUBROUTINE KNOTS	RAN 1410
C	SUBROUTINE RESID	RAN 1420
C	SUBROUTINE JAC	RAN 1430
C		RAN 1440
C	SINCE THESE ROUTINES ARE CALLED FROM WITHIN RANGE THE USER	RAN 1450
C	SHOULD ENSURE THAT THERE ARE NO POTENTIAL PROBLEMS DUE TO	RAN 1460
C	NAME CONFLICTS.	RAN 1470
C		RAN 1480
C		RAN 1490
C	**** QUALITY ASSURANCE AND SOFTWARE STANDARD ****	RAN 1500
C		RAN 1510
C	THE SUBROUTINES THAT COMPRISE THIS PACKAGE	RAN 1520
C	HAVE BEEN WRITTEN TO CONFORM TO THE FORTRAN IV	RAN 1530
C	ANSI STANDARD 1966, AND THEY HAVE BEEN VERIFIED	RAN 1540
C	USING THE BELL TELEPHONE LABORATORIES FORTRAN	RAN 1550
C	VERIFIER: PPORT.	RAN 1560
C	THE SUBROUTINES HAVE BEEN EXTENSIVELY TESTED ON	RAN 1570
C	A VARIETY OF TEST PROBLEMS, AND THEY HAVE BEEN	RAN 1580
C	ANALYSED FOR ERRORS USING THE DAVE SYSTEM FROM	RAN 1590
C	THE UNIVERSITY OF COLORADO.	RAN 1600
C	TO MAKE THE CODE EASY TO READ THE SUBROUTINES	RAN 1610
C	HAVE BEEN REFORMATTED USING POLISH.	RAN 1620
C		RAN 1630

```

C
C
C **** P.W.GAFFNEY SEPTEMBER 9. 1980 ****
C
C .....
C
C   REAL X(N), F(N), WK(LWK), ETA(LEP), PSI(LEP), LOW
C   REAL LAMBDA
C   INTEGER IL(N), HALF
C
C .....
C   N.B. IN THE FOLLOWING DATA STATEMENT
C   IMAX IS A LIMIT IMPOSED ON THE NUMBER OF NEWTON
C   ITERATIONS REQUIRED TO COMPUTE THE KNOTS ETA AND
C   PSI.
C   DEFAULT: IMAX=1000
C   MOUT IS THE LOGICAL UNIT NUMBER FOR THE OUTPUT UNIT
C   WHERE ERROR MESSAGES ARE PRINTED.
C   DEFAULT: MOUT = 6
C   TO SUPPRESS PRINTING SET MOUT .LE. 0
C
C   DATA IMAX, MOUT /1000,6/
C .....
C
C   DATA ZERO, POINT5, ONE /0.0,0.5,1.0/
C   IFAIL = 0
C   NMK = N - K
C   NK = MIN0(K,NMK)
C   LAJ = NMK*NK
C   KM1 = K - 1
C   KP1 = K + 1
C   KP2 = K + 2
C   NP1 = N + 1
C   NP2 = N + 2
C .....
C   CHECK THAT THE INPUT PARAMETERS ARE SENSIBLE
C .....
C   IF (ALPHA.LT.X(1)) GO TO 390
C   IF (N.LT.2) GO TO 320
C   IF (LWK.LT.5*N-2*K+1+LAJ) GO TO 370
C   IF (LEP.LT.NMK) GO TO 380
C   IF (K.LT.1 .OR. K.GT.N) GO TO 330
C   IF (AL.LE.ZERO) GO TO 360
C   NM1 = N - 1
C   DO 10 I=1,NM1
C       IF (X(I).LT.X(I+1)) GO TO 10
C       GO TO 340
C   10 CONTINUE
C
C   COMPUTE FACTORIAL K-1
C   FACT = ONE
C   IF (KM1.LT.2) GO TO 30
C   DO 20 I=2,KM1
C       FACT = FACT*FLOAT(I)
C   20 CONTINUE

```

```

RAN 1640
RAN 1650
RAN 1660
RAN 1670
RAN 1680
RAN 1690
RAN 1700
RAN 1710
RAN 1720
RAN 1730
RAN 1740
RAN 1750
RAN 1760
RAN 1770
RAN 1780
RAN 1790
RAN 1800
RAN 1810
RAN 1820
RAN 1830
RAN 1840
RAN 1850
RAN 1860
RAN 1870
RAN 1880
RAN 1890
RAN 1900
RAN 1910
RAN 1920
RAN 1930
RAN 1940
RAN 1950
RAN 1960
RAN 1970
RAN 1980
RAN 1990
RAN 2000
RAN 2010
RAN 2020
RAN 2030
RAN 2040
RAN 2050
RAN 2060
RAN 2070
RAN 2080
RAN 2090
RAN 2100
RAN 2110
RAN 2120
RAN 2130
RAN 2140
RAN 2150
RAN 2160
RAN 2170
RAN 2180

```

C END OF COMPUTING FACTORIAL K-1.	RAN 2190
C	RAN 2200
30 IF (IAG.GT.1) GO TO 80	RAN 2210
IF (K.EQ.N) GO TO 80	RAN 2220
C	RAN 2230
C COMPUTE THE KTH. DIVIDED DIFFERENCES F(X(I),...,X(I+K))	RAN 2240
C AND STORE THEM IN WK(K+1+I), I=1,...,N-K	RAN 2250
C	RAN 2260
DO 40 I=1,N	RAN 2270
KI = KP1 + I	RAN 2280
WK(KI) = F(I)	RAN 2290
40 CONTINUE	RAN 2300
DO 60 J=1,K	RAN 2310
C J DENOTES THE JTH DIVIDED DIFFERENCE OF F.	RAN 2320
NMJ = N - J	RAN 2330
DO 50 I=1,NMJ	RAN 2340
C WK(KP1+I) IS THE DIVIDED DIFFERENCE F(X(I),...,X(I+J-1)).	RAN 2350
KI = KP1 + I	RAN 2360
S1 = WK(KI+1) - WK(KI)	RAN 2370
IPJ = I + J	RAN 2380
S1 = S1/(X(IPJ)-X(I))	RAN 2390
WK(KI) = S1	RAN 2400
50 CONTINUE	RAN 2410
60 CONTINUE	RAN 2420
C	RAN 2430
C END OF COMPUTING THE DIVIDED DIFFERENCES.	RAN 2440
C	RAN 2450
C COMPUTE THE COMPONENTS OF CSTA, NAMELY:	RAN 2460
C FACTORIAL(K-1)*F(X(I),...,X(I+K))/AL, I=1,...,N-K,	RAN 2470
C IN WK(K+1+I), I=1,...,NMK, AND CHECK THAT THEY SATISFY	RAN 2480
C INEQUALITIES: ABS(CSTA(I)).LE.1/K, I=1,...,NMK	RAN 2490
C	RAN 2500
WRM = ZERO	RAN 2510
DO 70 I=1,NMK	RAN 2520
KI = KP1 + I	RAN 2530
WK(KI) = FACT*WK(KI)/AL	RAN 2540
WRM = AMAX1(ABS(WK(KI)),WRM)	RAN 2550
70 CONTINUE	RAN 2560
C	RAN 2570
C END OF COMPUTING THE COMPONENTS OF CSTA.	RAN 2580
C	RAN 2590
C CHECK THAT THE VALUE OF AL IS LARGE ENOUGH	RAN 2600
C	RAN 2610
IF (WRM.GT.ONE/FLOAT(K)) GO TO 350	RAN 2620
C.....	RAN 2630
C END OF CHECKING THE INPUT PARAMETERS.	RAN 2640
C.....	RAN 2650
C	RAN 2660
C.....	RAN 2670
C COMPUTE MACHINE EPSILON	RAN 2680
C.....	RAN 2690
80 EPS = ONE	RAN 2700
90 EPS = POINT5*EPS	RAN 2710
EPSP1 = EPS + ONE	RAN 2720
IF (EPSP1.GT.ONE) GO TO 90	RAN 2730


```

      EPS = EPS + EPS
C.....
C      END OF COMPUTING MACHINE EPSILON.
C.....
C
C.....
C      COMPUTE JINT SUCH THAT X(JINT).LE.ALPHA.LT.X(JINT+1)
C      NOTE THAT THE FOLLOWING CODE MAKES USE OF THE FACT THAT
C      IN FORTRAN THE RESULT OF INTEGER DIVISION IS ROUNDED
C      DOWN TO THE NEAREST INTEGER.
C.....
      ILEFT = 1
      IRITE = NP1
100 MIDDLE = (ILEFT+IRITE)/2
      IF (ALPHA.LT.X(MIDDLE)) IRITE = MIDDLE
      IF (ALPHA.GE.X(MIDDLE)) ILEFT = MIDDLE
      IF (IRITE-ILEFT+1) GO TO 100
      JINT = ILEFT
C.....
C      END OF COMPUTING JINT.
C.....
C
      IF (R.GT.1) GO TO 130
C
C.....
C      SET THE SOLUTION FOR K=1 AND RETURN
C.....
C
C IF RE-ENTERED,(WITH ALPHA AND IAG THE ONLY CHANGED
C PARAMETERS), THEN DO NOT RE-COMPUTE THE KNOTS
C
      IF (IAG.GT.1) GO TO 120
      DO 110 I=1,NMK
        IP1 = I + 1
        XP = X(I) + POINT5*(X(IP1)-X(I))
        FP = F(IP1) - F(I)
        ETA(I) = XP + POINT5*(FLOAT((-1)**IP1)*FP)/AL
        PSI(I) = XP + POINT5*(FLOAT((-1)**I)*FP)/AL
110 CONTINUE
120 I = JINT + 1
      J = I
      IF (ALPHA.LT.ETA(JINT)) I = JINT
      IF (ALPHA.LT.PSI(JINT)) J = JINT
      TEMP1 = FLOAT((-1)**(I-1))*AL*ALPHA-X(I) + F(I)
      TEMP2 = FLOAT((-1)**(J))*AL*(ALPHA-X(J)) + F(J)
      LOW = AMIN1(TEMP1,TEMP2)
      UP = AMAX1(TEMP1,TEMP2)
      OMEGA = POINT5*(LOW+UP)
C.....
C      RETURN TO CALLING PROGRAM
C.....
C
      GO TO 420
C
C.....

```

```

RAN 2740
RAN 2750
RAN 2760
RAN 2770
RAN 2780
RAN 2790
RAN 2800
RAN 2810
RAN 2820
RAN 2830
RAN 2840
RAN 2850
RAN 2860
RAN 2870
RAN 2880
RAN 2890
RAN 2900
RAN 2910
RAN 2920
RAN 2930
RAN 2940
RAN 2950
RAN 2960
RAN 2970
RAN 2980
RAN 2990
RAN 3000
RAN 3010
RAN 3020
RAN 3030
RAN 3040
RAN 3050
RAN 3060
RAN 3070
RAN 3080
RAN 3090
RAN 3100
RAN 3110
RAN 3120
RAN 3130
RAN 3140
RAN 3150
RAN 3160
RAN 3170
RAN 3180
RAN 3190
RAN 3200
RAN 3210
RAN 3220
RAN 3230
RAN 3240
RAN 3250
RAN 3260
RAN 3270
RAN 3280

```

```

C  COMPUTE THE CLOSEST POSSIBLE BOUNDS ON F(ALPHA) FOR K.E.2 *
C*****
C
C  IF RE-ENTERED,(WITH ALPHA AND IAG THE ONLY CHANGED
C  PARAMETERS), THEN DO NOT RE-COMPUTE THE KNOTS
C
C    130 IF (IAG.GT.1) GO TO 150
C
C      IF (K.EQ.N) GO TO 150
C
C*****
C    COMPUTE THE KNOTS OF THE PERFECT SPLINES *
C*****
      LNK = 4*NMK + (2*K)
      INCP1 = LNK - LAJ + 1
C
C  FIRST COMPUTE THE KNOTS ETA(1),...,ETA(N-K)
      CALL KNOTS(N, X, K, WK(KP2), WK(INCP1), NK, WK(NP2), LNK,
      * IL, NMK, ETA, IMAX, IFAIL, EPS)
C
C  THEN(IF IFAIL=0) COMPUTE THE KNOTS PSI(1),...,PSI(N-K)
      IF (IFAIL.EQ.5) GO TO 400
      IF (IFAIL.EQ.6) GO TO 410
      DO 140 I=1,NMK
        KI = KP1 + I
        WK(KI) = -WK(KI)
      140 CONTINUE
      CALL KNOTS(N, X, K, WK(KP2), WK(INCP1), NK, WK(NP2), LNK,
      * IL, NMK, PSI, IMAX, IFAIL, EPS)
      IF (IFAIL.EQ.5) GO TO 400
      IF (IFAIL.EQ.6) GO TO 410
C*****
C    END OF COMPUTING THE KNOTS. *
C*****
C
C*****
C    SET THE KNOTS OF THE AUGMENTED B-SPLINE M(ALPHA(X),BY *
C    CHOOSING K DATA POINTS CLOSEST TO ALPHA AND ARRANGING *
C    THE RESULTING K+1 POINTS IN ASCENDING ORDER. *
C    STORE THE KNOTS IN WK(1),...,WK(K+1), AND STORE *
C    THE SUBSCRIPT OF THE DATA POINT WHICH FORMS THE FIRST *
C    KNOT OF M(ALPHA(X) IN THE VARIABLE IBAR. *
C*****
      150 HALFK = K/2
      IF (JINT.LE.HALFK) GO TO 180
      IF (JINT.GE.NMK+HALFK) GO TO 210
      J1 = JINT - HALFK
      DO 160 I=1,HALFK
        II = J1 + I
        WK(I) = X(II)
      160 CONTINUE
      WK(HALFK+1) = ALPHA
      J2 = HALFK + 2
      DO 170 I=J2,KP1
        II = J1 - 1 + I

```

```

RAN 3290
RAN 3300
RAN 3310
RAN 3320
RAN 3330
RAN 3340
RAN 3350
RAN 3360
RAN 3370
RAN 3380
RAN 3390
RAN 3400
RAN 3410
RAN 3420
RAN 3430
RAN 3440
RAN 3450
RAN 3460
RAN 3470
RAN 3480
RAN 3490
RAN 3500
RAN 3510
RAN 3520
RAN 3530
RAN 3540
RAN 3550
RAN 3560
RAN 3570
RAN 3580
RAN 3590
RAN 3600
RAN 3610
RAN 3620
RAN 3630
RAN 3640
RAN 3650
RAN 3660
RAN 3670
RAN 3680
RAN 3690
RAN 3700
RAN 3710
RAN 3720
RAN 3730
RAN 3740
RAN 3750
RAN 3760
RAN 3770
RAN 3780
RAN 3790
RAN 3800
RAN 3810
RAN 3820
RAN 3830

```

WK(I) = X(II)	RAN 3840
170 CONTINUE	RAN 3850
IBAR = J1 + 1	RAN 3860
GO TO 240	RAN 3870
180 DO 190 I=1,JINT	RAN 3880
WK(I) = X(I)	RAN 3890
190 CONTINUE	RAN 3900
WK(JINT+1) = ALPHA	RAN 3910
J2 = JINT + 2	RAN 3920
DO 200 I=J2,KP1	RAN 3930
WK(I) = X(I-1)	RAN 3940
200 CONTINUE	RAN 3950
IBAR = 1	RAN 3960
GO TO 240	RAN 3970
210 IEND = JINT - NMK	RAN 3980
DO 220 I=1,IEND	RAN 3990
II = NMK + I	RAN 4000
WK(I) = X(II)	RAN 4010
220 CONTINUE	RAN 4020
WK(IEND+1) = ALPHA	RAN 4030
J2 = IEND + 2	RAN 4040
IBAR = NMK + 1	RAN 4050
IF (J2.GT.KP1) GO TO 240	RAN 4060
DO 230 I=J2,KP1	RAN 4070
II = NMK - 1 + I	RAN 4080
WK(I) = X(II)	RAN 4090
230 CONTINUE	RAN 4100
C*****	
C END OF SETTING THE KNOTS OF MALPHA(X).	RAN 4110
C*****	RAN 4120
C	RAN 4130
C*****	RAN 4140
C COMPUTE THE QUANTITIES CUP AND CLOW	RAN 4150
C*****	RAN 4160
C	RAN 4170
C SET CUP AND CLOW WHEN K=5	RAN 4180
240 IF (K.LT.N) GO TO 250	RAN 4190
CUP = AL/FLOAT(K)	RAN 4200
CLOW = -CUP	RAN 4210
GO TO 260	RAN 4220
C	RAN 4230
C CALCULATE CUP AND CLOW WHEN K.LT.N	RAN 4240
C FIRST COMPUTE THE INTEGRAL OF MALPHA*(THE KTH.DERIV. OF U/AL)	RAN 4250
250 I2 = NP2 + 5	RAN 4260
CALL RESID(-1, 1, 1, KP1, WK(1), NMK, ETA, WK(KP2), K,	RAN 4270
* WK(NP2), WK(I2), NP2, WK, NP1, IL(KP1), SS, ICONT,	RAN 4280
* LAMBDA,	RAN 4290
TEMP = AL*WK(NP2)	RAN 4300
C	RAN 4310
C THEN COMPUTE THE INTEGRAL OF MALPHA*(THE KTH.DERIV. OF L/AL)	RAN 4320
CALL RESID(-1, 1, 1, KP1, WK(1), NMK, PSI, WK(KP2), K,	RAN 4330
* WK(NP2), WK(I2), NP2, WK, NP1, IL(KP1), SS, ICONT,	RAN 4340
* LAMBDA)	RAN 4350
CUP = AMAX1(TEMP,-AL*WK(NP2))	RAN 4360
CLOW = AMIN1(TEMP,-AL*WK(NP2))	RAN 4370
	RAN 4380

C*****	RAN 4390
C END OF COMPUTING CUP AND CLOW.	RAN 4400
C*****	RAN 4410
C	RAN 4420
C*****	RAN 4430
C FIRST CALCULATE THE COEFFICIENTS C(1),...,C(K) OF THE	RAN 4440
C POLYNOMIAL:	RAN 4450
C C(1)+C(2)(X-X(IBAR))+C(3)(X-X(IBAR))(X-X(IBAR+1))+...	RAN 4460
C ...+C(K)(X-X(IBAR))(X-X(IBAR+1))...(X-X(IBAR+K-1))	RAN 4470
C OF DEGREE K-1, WHICH PASSES THROUGH THE K FUNCTION	RAN 4480
C VALUES F(X(IBAR)),...,F(X(IBAR+K-1)), AND STORE THE	RAN 4490
C COEFFICIENTS IN WK(N+2),...,WK(N+K+1). THEN EVALUATE	RAN 4500
C THE POLYNOMIAL, AT X=ALPHA, BY NESTED MULTIPLICATION	RAN 4510
C AND STORE THE RESULT IN THE VARIABLE SUM.	RAN 4520
C*****	RAN 4530
C	RAN 4540
260 C1 = F(IBAR)	RAN 4550
WK(NP2) = C1	RAN 4560
WK(NP2+1) = (F(IBAR+1)-C1)/(X(IBAR+1)-X(IBAR))	RAN 4570
IF (K.EQ.2) GO TO 290	RAN 4580
DO 280 I=3,K	RAN 4590
PROD = ONE	RAN 4600
SUM = ZERO	RAN 4610
LI = IBAR + I - 1	RAN 4620
XLI = X(LI)	RAN 4630
IM1 = I - 1	RAN 4640
DO 270 J=2,IM1	RAN 4650
ILJ = IBAR + J - 2	RAN 4660
PROD = PROD*(XLI-X(ILJ))	RAN 4670
J1 = NP2 + J - 1	RAN 4680
SUM = SUM + WK(J1)*PROD	RAN 4690
270 CONTINUE	RAN 4700
I1 = NP2 + I - 1	RAN 4710
ILI = LI - 1	RAN 4720
WK(I1) = (F(LI)-C1-SUM)/(PROD*(XLI-X(ILI)))	RAN 4730
280 CONTINUE	RAN 4740
C	RAN 4750
290 NPKP1 = N + K + 1	RAN 4760
SUM = WK(NPKP1)	RAN 4770
DO 300 I=1,KM1	RAN 4780
ILK = IBAR + K - I - 1	RAN 4790
NP = NPKP1 - I	RAN 4800
SUM = (ALPHA-X(ILK))*SUM + WK(NP)	RAN 4810
300 CONTINUE	RAN 4820
C*****	RAN 4830
C END OF EVALUATING THE INTERPOLATING POLYNOMIAL.	RAN 4840
C*****	RAN 4850
C	RAN 4860
C*****	RAN 4870
COMPUTE THE RANGE OF POSSIBLE VALUES OF F(ALPHA)	RAN 4880
C AND COMPUTE THE OPTIMAL ESTIMATE OMEGA	RAN 4890
C*****	RAN 4900
C	RAN 4910
PROD = ONE	RAN 4920
DO 310 I=1,K	RAN 4930

```

      LI = IBAR + I - 1
      PROD = PROD*(ALPHA-X(LI))
310 CONTINUE
      TEMP1 = SUM + PROD*CLOW/FACT
      TEMP2 = SUM + PROD*CUP/FACT
      LOW = AMIN1(TEMP1,TEMP2)
      UP = AMAX1(TEMP1,TEMP2)
      OMEGA = POINT5*(LOW+UP)
C*****
C      END OF COMPUTING THE CLOSEST POSSIBLE BOUNDS ON F(ALPHA)*
C*****
      GO TO 420
C*****
C      DIAGNOSTIC PRINTING
C*****
320 IFAIL = 1
      IF (NOUT.GT.0) WRITE (NOUT,99999)
      GO TO 420
330 IFAIL = 2
      IF (NOUT.GT.0) WRITE (NOUT,99998)
      GO TO 420
340 IFAIL = 3
      IF (NOUT.GT.0) WRITE (NOUT,99997)
      GO TO 420
350 IFAIL = 4
      WKM = WKM*FLOAT(K)*AL
      IF (NOUT.GT.0) WRITE (NOUT,99996) WKM
      GO TO 420
360 IFAIL = 5
      IF (NOUT.GT.0) WRITE (NOUT,99994)
      GO TO 420
370 IFAIL = 8
      IF (NOUT.GT.0) WRITE (NOUT,99992)
      GO TO 420
380 IFAIL = 9
      IF (NOUT.GT.0) WRITE (NOUT,99991)
      GO TO 420
390 IFAIL = 10
      IF (NOUT.GT.0) WRITE (NOUT,99990)
      GO TO 420
400 IF (NOUT.GT.0) WRITE (NOUT,99995)
      GO TO 420
410 IF (NOUT.GT.0) WRITE (NOUT,99993)
C      END OF PRINTING.
C*****
420 RETURN
C*****
99999 FORMAT (1X, 18HMESSAGE FROM RANGE, 3X, 6HN.LT.2)
99998 FORMAT (1X, 18HMESSAGE FROM RANGE, 3X, 15HK IS NOT IN THE,
* 22H INTERVAL 1.LE.K.LE.N)
99997 FORMAT (1X, 18HMESSAGE FROM RANGE, 3X, 15HTHE DATA POINTS,
* 26H ARE NOT IN ASCENDING ORDER.)
99996 FORMAT (1X, 18HMESSAGE FROM RANGE, 3X, 15HTHE VALUE OF AL,
* 17H MUST BE AT LEAST, 1PE14.6)
99995 FORMAT (1X, 18HMESSAGE FROM RANGE, 3X, 15HMETHOD FAILED T,

```

```

RAN 4940
RAN 4950
RAN 4960
RAN 4970
RAN 4980
RAN 4990
RAN 5000
RAN 5010
RAN 5020
RAN 5030
RAN 5040
RAN 5050
RAN 5060
RAN 5070
RAN 5080
RAN 5090
RAN 5100
RAN 5110
RAN 5120
RAN 5130
RAN 5140
RAN 5150
RAN 5160
RAN 5170
RAN 5180
RAN 5190
RAN 5200
RAN 5210
RAN 5220
RAN 5230
RAN 5240
RAN 5250
RAN 5260
RAN 5270
RAN 5280
RAN 5290
RAN 5300
RAN 5310
RAN 5320
RAN 5330
RAN 5340
RAN 5350
RAN 5360
RAN 5370
RAN 5380
RAN 5390
RAN 5400
RAN 5410
RAN 5420
RAN 5430
RAN 5440
RAN 5450
RAN 5460
RAN 5470
RAN 5480

```

<ul style="list-style-type: none"> • 3080 CONVERGE IN IMAI ITERATIONS.) 	RAN 5490
99994 FORMAT (1X, 18MESSAGE FROM RANGE, 3X, 15THE VALUE OF AL,	RAN 5500
<ul style="list-style-type: none"> • 13H IS .LE. ZERO) 	RAN 5510
99993 FORMAT (1X, 18MESSAGE FROM RANGE, 3X, 15THE CONTINUATIO.	RAN 5520
<ul style="list-style-type: none"> • 20HN METHOD HAS FAILED.//1X, 20THIS IS USUALLY BECA. 	RAN 5530
<ul style="list-style-type: none"> • 22HUSE AL IS TOO SMALL OR//1X, 18THE VALUE OF K IS . 	RAN 5540
<ul style="list-style-type: none"> • 10HTOO LARGE.) 	RAN 5550
99992 FORMAT (1X, 18MESSAGE FROM RANGE, 3X, 15H THE ARRAY WK I.	RAN 5560
<ul style="list-style-type: none"> • 11ES TOO SMALL) 	RAN 5570
99991 FORMAT (1X, 18MESSAGE FROM RANGE, 3X, 15H THE ARRAYS ETA.	RAN 5580
<ul style="list-style-type: none"> • 22H AND PSI ARE TOO SMALL) 	RAN 5590
99990 FORMAT (1X, 18MESSAGE FROM RANGE, 3X, 13HALPHA.LT.X(1))	RAN 5600
END	RAN 5610

```

      SUBROUTINE KNOTS(N, X, K, CSTA, AJ, NK, WK, ISW, L, NMK,
      *   ETA, IMAX, IFAIL, EPS)
C *****
C *
C *   PURPOSE
C *
C *   GIVEN THE DATA POINTS:  X(1).LT.X(2).....LT.X(N)
C *   AND THE VECTOR: (CSTA(1),....CSTA(N-K)), WHERE 1.LE.K.LE.N-1
C *   THIS SUBROUTINE COMPUTES THE KNOTS OF THE PERFECT SPLINE U
C *
C *****
C
      REAL X(N), CSTA(NMK), AJ(NMK,NK), WK(ISW), ETA(NMK),
      *   LASTSS, LAMBDA
      INTEGER L(NMK)
      DATA ZERO, POINT5, ONE, TWO, FIVE, TEN
      *   /0.0,0.5,1.0,2.0,5.0,100.0/
      IFAIL = 0
      LAMBDA = ONE
      N2 = 2*NMK
      N3 = 3*NMK
      N1P1 = NMK + 1
      N3P1 = N3 + 1
      I1 = 4*NMK + 1
      I2 = I1 + K
      EPS1 = TEN*FLOAT(NMK*NMK)*EPS*EPS
C *****
C   SET THE INITIAL APPROXIMATION
C *****
      KP1 = K + 1
      DO 20 I=1,NMK
          SUM = ZERO
          DO 10 J=1,KP1
              IPJM1 = I + J - 1
              SUM = SUM + X(IPJM1)
          10 CONTINUE
          ETA(I) = SUM/FLOAT(KP1)
      20 CONTINUE
C   END OF INITIALISATION.
C *****
C
      THE VALUES OF THE INTEGER VARIABLES: ICONT,IUP,IARG AND IA
C
      ICONT HAS THE VALUE -1 WHEN A CONTINUATION STEP IS IN
      PROGRESS. OTHERWISE IT HAS THE VALUE +1.
C
      IUP   IS SET TO +1 WHEN ETA IS UPDATED AND -1 OTHERWISE.
C
      IARG  IS USED IN RESID. THE RESIDUALS ARE EVALUATED AT ETA
      IF IARG=+1 AND AT ETA+(NEWTON CORRECTION) IF IARG=-1
C
      IA    IS DEFINED IN RESID. IN THIS SUBROUTINE
      IT IS SET TO THE VALUE +1.
C
C
C

```

```

KNO 17
KNO 20
KNO 30
KNO 40
KNO 50
KNO 60
KNO 70
KNO 80
KNO 90
KNO 100
KNO 110
KNO 120
KNO 130
KNO 140
KNO 150
KNO 160
KNO 170
KNO 180
KNO 190
KNO 200
KNO 210
KNO 220
KNO 230
KNO 240
KNO 250
KNO 260
KNO 270
KNO 280
KNO 290
KNO 300
KNO 310
KNO 320
KNO 330
KNO 340
KNO 350
KNO 360
KNO 370
KNO 380
KNO 390
KNO 400
KNO 410
KNO 420
KNO 430
KNO 440
KNO 450
KNO 460
KNO 470
KNO 480
KNO 490
KNO 500
KNO 510
KNO 520
KNO 530
KNO 540
KNO 550

```

ICONT = 1	KNO	560
IUP = -1	KNC	570
IA = 1	KNO	580
C	KNO	590
C*****	KNO	600
C OUTER LOOP FOR CONTINUATION METHOD	KNC	610
C*****	KNO	620
DO 230 JCONT=1,IMAX	KNO	630
IARG = 1	KNO	640
C	KNO	650
C*****	KNO	660
C BEGIN NEWTON ITERATION	KNO	670
C*****	KNO	680
DO 160 ITER=JCONT,IMAX	KNO	690
C	KNO	700
C COMPUTE THE RESIDUALS IN WK(NMK+1),...,WK(2*NMK), AND	KNO	710
C COMPUTE THE SUM OF SQUARES SS	KNO	720
CALL RESID(IA, IARG, NMK, N, X, NMK, ETA, CSTA,	KNO	730
K, WK(1), WK(12), M3, WK, NMK, L, SS,	KNO	740
ICONT, LAMBDA)	KNO	750
IF (ITER.EQ.JCONT) GO TO 50	KNO	760
C	KNO	770
C IF SS HAS DECREASED THEN UPDATE THE APPROXIMATION	KNO	780
IF (SS.LT.LASTSS) GO TO 30	KNO	790
C	KNO	800
C IF SS HAS INCREASED DUE TO POUNDING ERROR FINISH ITERATION	KNO	810
IF (SS.LE.LASTSS+FIVE*EPS*EPS) GO TO 140	KNO	820
C	KNO	830
C ELSE SET LAMBDA AND BEGIN CONTINUATION STEP	KNO	840
LAMBDA = POINT5*LAMBDA	KNO	850
IF (LAMBDA.LE.EPS) GO TO 250	KNO	860
GO TO 170	KNO	870
C	KNO	880
C UPDATE THE APPROXIMATION	KNO	890
30 IUP = 1	KNO	900
DO 40 I=1,NMK	KNO	910
ETA(I) = TA(I) + WK(I)	KNO	920
40 CONTINUE	KNO	930
C END OF UPDATE.	KNO	940
C	KNO	950
C IF SS IS LESS THAN OR EQUAL TO EPS1 FINISH ITERATION	KNO	960
IF (SS.LE.EPS1) GO TO 140	KNO	970
C	KNO	980
C STORE THE SUM OF SQUARES	KNO	990
50 LASTSS = SS	KNO	1000
C	KNO	1010
C COMPUTE THE NEWTON CORRECTION IN WK(1),...,WK(NMK)	KNO	1020
CALL JAC(N, X, NMK, ETA, L, K, WK(1), NK, AJ,	KNO	1030
WK(1), WK(N1P1))	KNO	1040
IARG = -1	KNO	1050
C	KNO	1060
C	KNO	1070
C COMPUTE THE SQUARE OF THE L2-NORM OF THE CORRECTION AND OF ETA	KNO	1080
C AND THEN TEST FOR CONVERGENCE	KNO	1090
S1 = ZERO	KNO	1100

	S2 = ZERO	KNO 1110
	DO 60 I=1,NMK	KNO 1120
	S1 = S1 + WK(I)*WK(I)	KNO 1130
	S2 = S2 + ETA(I)*ETA(I)	KNO 1140
60	CONTINUE	KNO 1150
	IF (S1.LE.EPS*EPS*S2) GO TO 140	KNO 1160
C		KNO 1170
C	TEST THAT THE COMPONENTS OF ETA+(NEWTON CORRECTION) ARE	KNO 1180
C	DISTINCT AND IN ASCENDING ORDER	KNO 1190
	IJ = -1	KNO 1200
	NMKH1 = NMK - 1	KNO 1210
	IREND = NMKH1	KNO 1220
	DO 70 I=1,NMKH1	KNO 1230
	E1 = ETA(I) + WK(I)	KNO 1240
	E2 = ETA(I+1) + WK(I+1)	KNO 1250
	IF (E1.LT.E2) GO TO 70	KNO 1260
	GO TO 90	KNO 1270
70	CONTINUE	KNO 1280
	IJ = 1	KNO 1290
	IREND = NMK	KNO 1300
C		KNO 1310
C	TEST THAT THE SCHOENBERG-WHITNEY CONDITIONS ARE SATISFIED	KNO 1320
	DO 80 I=1,NMK	KNO 1330
	IPK = I + K	KNO 1340
	TEMP = ETA(I) + WK(I)	KNO 1350
	IF (TEMP.LE.X(I) .OR. TEMP.GE.X(IPK)) GO	KNO 1360
	TO 90	KNO 1370
80	CONTINUE	KNO 1380
C	THE SCHOENBERG-WHITNEY CONDITIONS ARE SATISFIED.	KNO 1390
C		KNO 1400
	GO TO 160	KNO 1410
C		KNO 1420
C	REDUCE LAMBDA	KNO 1430
90	LAMBDA = POINT5*LAMBDA	KNO 1440
	IF (LAMBDA.LE.EPS) GO TO 250	KNO 1450
C		KNO 1460
C	IF DOING A CONTINUATION STEP THEN CONTINUE	KNO 1470
	IF (ICONT.EQ.(-1)) GO TO 200	KNO 1480
C		KNO 1490
C	ELSE COMPUTE AN LAMBDA SUCH THAT IF IJ=-1 THE COMPONENTS OF	KNO 1500
C	ETA +LAMBDA*(NEWTON CORRECTION) ARE DISTINCT AND IN	KNO 1510
C	ASCENDING ORDER,OR IF IJ=1 THE COMPONENTS SATISFY THE	KNO 1520
C	SCHOENBERG-WHITNEY CONDITIONS	KNO 1530
	ITEND = 1 + IABS(INT(ALOG10(EPS)/ALOG10(TWO)))	KNO 1540
	DO 100 IR=1,ITEND	KNO 1550
	IF (IJ.EQ.(-1)) GO TO 100	KNO 1560
	IRPK = IR + K	KNO 1580
	TEMP = ETA(IR) + LAMBDA*WK(IR)	KNO 1590
	IF (TEMP.LE.X(IR) .OR.	KNO 1600
	TEMP.GE.X(IRPK)) GO TO 120	KNO 1610
	GO TO 110	KNO 1620
100	E1 = ETA(IR) + LAMBDA*WK(IR)	KNO 1630
	E2 = ETA(IR+1) + LAMBDA*WK(IR+1)	KNO 1640
	IF (E1.GE.E2) GO TO 120	KNO 1650

110	CONTINUE	KNO 1660
	GO TO 170	KNO 1670
120	LAMBDA = POINT5*LAMBDA	KNO 1680
130	CONTINUE	KNO 1690
C	AT THIS STAGE LAMBDA IS .LE. EPS SO WE FINISH	KNO 1700
	GO TO 250	KNO 1710
C		KNO 1720
C	FINISH ITERATION...	KNO 1730
C	IF DOING A NEWTON STEP THEN RETURN	KNO 1740
140	IF (ICONT.EQ.1) GO TO 260	KNO 1750
C		KNO 1760
C	ELSE IF ETA HAS NOT BEEN UPDATED THEN SET ERROR FLAG	KNO 1770
	IF (IUP.EQ.(-1)) GO TO 250	KNO 1780
C		KNO 1790
C	OTHERWISE FINISH CONTINUATION AND START NEWTON STEP	KNO 1800
	LAMBDA = ONE + LAMBDA	KNO 1810
	ICONT = 1	KNO 1820
C		KNO 1830
C	STORE THE VECTOR ETA	KNO 1840
	DO 150 I=1,NMK	KNO 1850
	N3PI = N3 + I	KNO 1860
	WK(N3PI) = ETA(I)	KNO 1870
150	CONTINUE	KNO 1880
C	END OF STORING ETA.	KNO 1890
	GO TO 230	KNO 1900
160	CONTINUE	KNO 1910
C		KNO 1920
C	AT THIS STAGE NEWTONS METHOD HAS FAILED TO CONVERGE	KNO 1930
C	AFTER IMAX ITERATIONS.	KNO 1940
	GO TO 240	KNO 1950
C		KNO 1960
C	*****	KNO 1970
C	END OF NEWTON ITERATION.	KNO 1980
C	*****	KNO 1990
C		KNO 2000
C	*****	KNO 2010
C	CONTINUATION STEP	KNO 2020
C	*****	KNO 2030
C		KNO 2040
C	IF DOING A CONTINUATION STEP THEN CONTINUE	KNO 2050
170	IF (ICONT.EQ.(-1)) GO TO 200	KNO 2060
C		KNO 2070
C	ELSE BEGIN CONTINUATION...	KNO 2080
C	IF THIS IS THE FIRST CONTINUATION STEP THEN STORE THE	KNO 2090
C	INITIAL APPROXIMATION IN WK(3*NMK+1),...,WK(4*NMK)	KNO 2100
	IF (JCONT.GT.1) GO TO 190	KNO 2110
	DO 180 I=1,NMK	KNO 2120
	N3PI = N3 + I	KNO 2130
	WK(N3PI) = ETA(I)	KNO 2140
180	CONTINUE	KNO 2150
C		KNO 2160
C	STORE THE RESIDUALS, AT THE INITIAL APPROXIMATION, IN	KNO 2170
C	WK(2*NMK+1),...,WK(3*NMK)	KNO 2180
190	CALL RESID(IA, 1, NMK, N, X, NMK, WK(N3PI), CSTA, K,	KNO 2190
	WK(I1), WK(I2), N3, WK, N2, L, SS, ICONT,	KNO 2200

```

      LAMBDA)
C
C      SET ICONT = -1
C      ICONT = -1
C
C      IF ETA HAS NOT BEEN UPDATED THEN BEGIN ANOTHER ITERATION
200      IF (IUP.EQ.(-1)) GO TO 230
C
C      ELSE, IF NECESSARY, RESET THE INITIAL APPROXIMATION
C      IF (JCONT.EQ.1) GO TO 220
C      DO 210 I=1,NMK
C          N3PI = N3 + I
C          ETA(I) = WK(N3PI)
210      CONTINUE
220      IUP = -1
230      CONTINUE
C
C      AT THIS STAGE THE CONTINUATION METHOD HAS FAILED TO
C      CONVERGE AFTER IMAX ITERA IS.
C      GO TO 240
C
C      *****
C      END OF CONTINUATION METHOD.
C      *****
C
C      *****
C      SET THE ERROR RETURN FLAG: IFAIL
C      *****
240      IFAIL = 5
C      GO TO 260
250      IFAIL = 6
C      END OF SETTING.
C      *****
260      RETURN
C      *****
      END

```

```

KNO 2210
KNO 2220
KNO 2230
KNO 2240
KNO 2250
KNO 2260
KNO 2270
KNO 2280
KNO 2290
KNO 2300
KNO 2310
KNO 2320
KNO 2330
KNO 2340
KNO 2350
KNO 2360
KNO 2370
KNO 2380
KNO 2390
KNO 2400
KNO 2410
KNO 2420
KNO 2430
KNO 2440
KNO 2450
KNO 2460
KNO 2470
KNO 2480
KNO 2490
KNO 2500
KNO 2510
KNO 2520
KNO 2530
KNO 2540
KNO 2550
KNO 2560

```

```

SUBROUTINE RESID(IA, IARG, NR, N, X, NMK, ETA, CSTA, K,
  V, VINT, IW, WK, INC, L, SS, ICONT, LAMBDA)
  .....
  C
  C      PURPOSE
  C      .....
  C
  C      THE PURPOSE OF THIS SUBROUTINE IS TWO-FOLD DEPENDING ON THE
  C      VALUE OF IA.
  C
  C      IF IA = +1 THEN THE SUBROUTINE COMPUTES:
  C      A) THE INTEGERS L(J) WHERE  $X(L(J)) \leq \eta(J) < X(L(J)+1)$ 
  C          $J=1, \dots, N-K$ . IT IS ASSUMED THAT THE KNOTS  $\eta(J)$ ,
  C          $J=1, \dots, N-K$  SATISFY THE SCHOENBERG-WHITNEY CONDITIONS.
  C
  C      B) THE RESIDUALS AT  $\eta$  WHEN IARG = +1
  C         AND THE RESIDUALS AT  $\eta + (\text{NEWTON CORRECTION})$  WHEN IARG = -1
  C
  C      C) THE SUM OF THE SQUARES OF THE RESIDUALS
  C         WHERE THE RESIDUALS ARE:
  C          $F = \text{LAMBDA} * \text{CSTA}$  WHEN  $\text{LAMBDA} = \text{ONE}$  AND  $\text{ICONT} = +1$ 
  C         AND
  C          $F = \text{LAMBDA} * \text{CSTA} - (\text{ONE} - \text{LAMBDA}) * \text{CHAT}$  WHEN  $\text{ICONT} = -1$ 
  C         THE COMPONENTS OF CHAT ARE THE RESIDUALS AT THE INITIAL
  C         APPROXIMATION.
  C
  C      IN THIS CASE THE ARGUMENT NR IS THE NUMBER OF RESIDUALS.
  C      THEREFORE IT SHOULD HAVE THE VALUE N-K.
  C
  C      IF IA = -1 THEN THE SUBROUTINE COMPUTES:
  C
  C      THE INTEGRAL OF THE QUANTITY  $\text{MALPHA} * \text{PHIBAR}$  WHERE MALPHA
  C      IS THE AUGMENTED B-SPLINE AND PHIBAR IS A BANG-BANG
  C      FUNCTION WHICH HAS BREAKPOINTS AT  $\eta(1), \dots, \eta(N-K)$ ,
  C      AND WHICH HAS THE VALUE +1 ON THE INTERVAL
  C       $X(1) \leq X \leq \eta(1)$ .
  C
  C      IN THIS CASE ONLY ONE INTEGRAL IS CALCULATED. THEREFORE
  C      NR SHOULD HAVE THE VALUE +1.
  C
  C      .....
  C      REAL X(N), ETA(NMK), CSTA(NMK), V(K), VINT(K), WK(IW),
  C      LAMBDA
  C      INTEGER L(NMK), P, Q, QP1
  C      DATA ZERO, ONE, TWO /0.0, 1.0, 2.0/
  C      SS = ZERO
  C      .....
  C      IF IA = +1 THEN COMPUTE THE INTEGERS L(J),  $J=1, \dots, N-K$ 
  C      .....
  C      IF (IA.EQ.(-1)) GO TO 40
  C      DO 30 J=1, NMK
  C          IF (IARG.EQ.1)  $E1 = \eta(J)$ 
  C          IF (IARG.EQ.(-1))  $E1 = \eta(J) + WK(J)$ 
  C           $JPKM1 = J + K - 1$ 
  C          DO 10 JT= JPKM1
  C              IF ( $E1 < X(JT+1)$ ) GO TO 20

```

```

10      CONTINUE                                RES 560
20      L(J) = JT                                RES 570
30      CONTINUE                                RES 580
C      END OF COMPUTING L(J).                    RES 590
C      .....                                    RES 600
C      .....                                    RES 610
C      .....                                    RES 620
C      BEGIN TO COMPUTE THE NR RESIDUALS          RES 630
C      .....                                    RES 640
40      DO 350 I=1,NR                            RES 650
          IPK = I + K                            RES 660
          INCPI = INC + I                        RES 670
          KPI = K + 1                            RES 680
          IF (IA.EQ.1) GO TO 150                 RES 690
C      .....                                    RES 700
C      DETERMINE WHICH BREAKPOINTS ARE IN THE SUPPORT OF MALPHA
C      .....                                    RES 710
C      .....                                    RES 720
          DO 50 Q=1,NMK                          RES 730
              IF (X(1).LE.ETA(Q)) GO TO 60       RES 740
50      CONTINUE                                RES 750
C      .....                                    RES 760
C      IF THIS COMMENT STATEMENT IS REACHED DURING EXECUTION THEN
C      THERE ARE NO BREAKPOINTS IN THE SUPPORT OF THE B-SPLINE MALPHA.
C      THEREFORE, THE INTEGRAL OF MALPHA * PHIBAR IS THE VALUE:
C      (-1)**IEXP*(1/K) WHERE                    RES 770
C      IEXP = NMK                                RES 780
C      GO TO 70                                  RES 790
C      .....                                    RES 800
C      .....                                    RES 810
C      .....                                    RES 820
60      IF (ETA(Q).LT.X(KPI)) GO TO 80           RES 830
C      .....                                    RES 840
C      .....                                    RES 850
C      LIKEWISE IF THIS STATEMENT IS REACHED THEN THERE ARE NO
C      BREAKPOINTS IN THE SUPPORT OF MALPHA AND THEREFORE THE INTEGRAL
C      OF MALPHA * PHIBAR IS THE VALUE:          RES 860
C      (-1)**IEXP*(1/K) WHERE                    RES 870
C      IEXP = Q - 1                              RES 880
C      .....                                    RES 890
70      WK(INCPI) = FLOAT((-1)**(IEXP))/FLOAT(K) RES 900
          RETURN                                  RES 910
C      .....                                    RES 920
C      .....                                    RES 930
C      THE SUPPORT OF THE B-SPLINE CONTAINS THE BREAKPOINT ETA(Q).
C      WE NOW DETERMINE THE REMAINING BREAKPOINTS IN THE SUPPORT.
C      .....                                    RES 940
C      .....                                    RES 950
C      .....                                    RES 960
80      IF (Q.LT.NMK) GO TO 90                   RES 970
          P = NMK                                RES 980
          GO TO 120                              RES 990
90      QP1 = Q + 1                              RES 1000
          DO 100 J=QP1,NMK                      RES 1010
              IF (X(KPI).LE.ETA(J)) GO TO 110   RES 1020
100     CONTINUE                                RES 1030
          J = NMK + 1                            RES 1040
110     P = J - 1                              RES 1050
C      .....                                    RES 1060
C      THE SUPPORT OF MALPHA CONTAINS THE BREAKPOINTS ETA(Q),...,ETA(P).
C      .....                                    RES 1070
C      .....                                    RES 1080
C      NOW COMPUTE THE KNOT INTERVALS WHERE THE DATA POINTS LIE
C      .....                                    RES 1090
120     DO 140 J=Q,P                            RES 1100

```

	E1 = ETA(J)	RES 1110
	IL = 1	RES 1120
	IR = K + 2	RES 1130
130	MIDDLE = (IL+IR)/2	RES 1140
	IF (E1.LT.X(MIDDLE)) IR = MIDDLE	RES 1150
	IF (E1.GE.X(MIDDLE)) IL = MIDDLE	RES 1160
	IF (IR.GT.IL+1) GO TO 130	RES 1170
	L(J) = IL	RES 1180
140	CONTINUE	RES 1190
C	END OF COMPUTING THE KNOT INTERVALS.	RES 1200
	GO TO 200	RES 1210
C		RES 1220
C	IF IA=+1 THEN COMPUTE THE INTEGERS Q AND P	RES 1230
C		RES 1240
150	J1 = MAX0(1,I-K+1)	RES 1250
	J2 = MIN0(I+1,NMK)	RES 1260
C	COMPUTE THE INTEGER Q SUCH THAT L(Q-1).LT.I.LE.L(Q)	RES 1270
	DO 160 Q=J1,NMK	RES 1280
	IF (I.LE.L(Q)) GO TO 170	RES 1290
160	CONTINUE	RES 1300
C	END OF COMPUTING Q.	RES 1310
C		RES 1320
C	COMPUTE THE INTEGER P SUCH THAT L(P).LT.I+K.LE.L(P+1)	RES 1330
170	DO 180 J=J2,NMK	RES 1340
	IF (IPK.LE.L(J)) GO TO 190	RES 1350
180	CONTINUE	RES 1360
	J = NMK + 1	RES 1370
190	P = J - 1	RES 1380
C	END OF COMPUTING P.	RES 1390
C	*****	RES 1400
C	COMPUTE THE INTEGRALS AND STORE THE SUMS IN S1 AND S2	RES 1410
C	*****	RES 1420
200	DO 300 ISUM=1,2	RES 1430
	SUM = ZERO	RES 1440
	IF (ISUM.EQ.1) GO TO 210	RES 1450
	Q = Q + 1	RES 1460
	IF (Q.GT.P) GO TO 290	RES 1470
210	DO 280 J=Q,P,2	RES 1480
	IF (IARG.EQ.1) E1 = ETA(J)	RES 1490
	IF (IARG.EQ.(-1)) E1 = ETA(J) + WK(J)	RES 1500
	DO 220 IR=1,K	RES 1510
	V(IR) = ZERO	RES 1520
	VINT(IR) = ZERO	RES 1530
220	CONTINUE	RES 1540
	JINT = L(J)	RES 1550
	JJ1 = IPK - JINT	RES 1560
	E10 = E1 - X(JINT)	RES 1570
	E20 = X(JINT+1) - E1	RES 1580
	V(1) = ONE/(X(JINT+1)-X(JINT))	RES 1590
	VINT(1) = E10*V(1)	RES 1600
	NK = MIN0(K,JJ1)	RES 1610
	IF (NK.EQ.1) GO TO 240	RES 1620
	DO 230 IJ=2,NK	RES 1630
	JINTPJ = JINT + IJ	RES 1640
	V(IJ) = (E10/(X(JINTPJ)-X(JINT)))	RES 1650

```

      V(IJ-1)
      VINT(IJ) = E10*V(IJ)
230  CONTINUE
240  IF (JINT.EQ.1) GO TO 270
      MJ = JINT - I
      DO 260 IJ=1,MJ
          JINTMJ = JINT - IJ
          E30 = E1 - X(JINTMJ)
          V(1) = (E20/(X(JINT+1)-X(JINTMJ))) *
              V(1)
          VINT(1) = VINT(1) + E30*V(1)
          NKJ = MIN0(K-IJ,JJ1)
          IF (NKJ.LE.1) GO TO 260
          DO 250 LI=2,NKJ
              JINTPL = JINT + LI
              A1 = E30*V(LI-1) +
                  (X(JINTPL)-E1)*V(LI)
              A2 = X(JINTPL) - X(JINTMJ)
              V(LI) = A1/A2
              VINT(LI) = VINT(LI) + E30*V(LI)
250  CONTINUE
260  CONTINUE
270  TERM = VINT(JJ1)/FLCAT(K)
      SUM = SUM + TERM
280  CONTINUE
      IF (ISUM.EQ.1) S1 = SUM
290  IF (ISUM.EQ.2) S2 = SUM
300  CONTINUE
C      END OF COMPUTING THE INTEGRALS.
C      *****
C      *****
C      STORE THE RESIDUALS IN WK(INC+I), I=1,...,NR AND
C      COMPUTE THE SUM OF SQUARES IN SS
C      *****
      SUM = FLOAT((-1)**(Q-2))
      SUM = SUM*(S1-S2)
      IF (IA.EQ.1) GO TO 310
      S1 = TWO*SUM + (FLOAT((-1)**P)/FLOAT(K))
      GO TO 320
310  S1 = TWO*SUM + (FLOAT((-1)**P)/FLOAT(K)) -
      LAMBDA*CSTA(I)
      N2PI = 2*NMK + I
      IF (ICONT.EQ.(-1)) S1 = S1 - (ONE-LAMBDA)*WK(N2PI)
      SS = SS + S1*S1
320  INCP1 = INC + I
      WK(INCP1) = S1
330  CONTINUE
C      *****
C      END OF COMPUTING THE RESIDUALS.
C      *****
      RETURN
      END
RES 1660
RES 1670
RES 1680
RES 1690
RES 1700
RES 1710
RES 1720
RES 1730
RES 1740
RES 1750
RES 1760
RES 1770
RES 1780
RES 1790
RES 1800
RES 1810
RES 1820
RES 1830
RES 1840
RES 1850
RES 1860
RES 1870
RES 1880
RES 1890
RES 1900
RES 1910
RES 1920
RES 1930
RES 1940
RES 1950
RES 1960
RES 1970
RES 1980
RES 1990
RES 2000
RES 2010
RES 2020
RES 2030
RES 2040
RES 2050
RES 2060
RES 2070
RES 2080
RES 2090
RES 2100
RES 2110
RES 2120
RES 2130
RES 2140
RES 2150
RES 2160
RES 2170
RES 2180

```

```

C      SUBROUTINE JAC(N, X, NPK, ETA, L, K, V, NK, AJ, Y, B)      JAC 10
C      .....                                                    JAC 20
C      PURPOSE                                                    JAC 30
C      .....                                                    JAC 40
C      GIVEN THE DATA POINTS: X(1)<X(2)<....<X(N-1)<X(N)      JAC 50
C      AND THE KNOTS: ETA(1),ETA(2),....,ETA(N-K-1),ETA(N-K)    JAC 60
C      THIS SUBROUTINE COMPUTES:-                                  JAC 70
C      1) THE ELEMENTS:                                           JAC 80
C          J(P,Q)=(-1)**(Q-1)*MHP(ETA(Q))  1.LE.P,Q.LE.N-K      JAC 90
C          OF THE JACOBIAN MATRIX J AND STORES THE TRANSPOSE OF  JAC 100
C          J, IN COMPACT FORM, IN THE ARRAY AJ.                  JAC 110
C          N.B. THE ARRAY AJ HAS N-K ROWS AND NK=MIN(K,N-K)      JAC 120
C          COLUMNS.                                              JAC 130
C          THE FUNCTION MHP(X) IS A SPLINE OF DEGREE K-1        JAC 140
C          WITH KNOTS AT THE POINTS X(P),X(P+1),....,X(P+K).    JAC 150
C          AND IS NON-ZERO ONLY OVER THE RANGE X(P)<X<X(P+K).    JAC 160
C      AND                                                         JAC 170
C      2) PERFORMS GAUSSIAN ELIMINATION ON J AND THEN USES      JAC 180
C      BACK SUBSTITUTION TO SOLVE THE LINEAR EQUATIONS:         JAC 190
C          J.Y = -B                                              JAC 200
C          WHERE Y AND B ARE GIVEN COLUMN VECTORS OF LENGTH N-K. JAC 210
C      .....                                                    JAC 220
C      REAL X(N), ETA(NPK), V(K), AJ(NPK,NK), Y(NPK), B(NPK),    JAC 230
C      * MULT                                                    JAC 240
C      INTEGER L(NPK)                                             JAC 250
C      DATA ZERO, ONE, TWO /0.0,1.0,2.0/                       JAC 260
C      .....                                                    JAC 270
C      COMPUTE THE TRANSPOSE OF THE JACOBIAN MATRIX              JAC 280
C      .....                                                    JAC 290
C      DO 80 J=1,NPK                                             JAC 300
C          DO 10 I=1,K                                           JAC 310
C              V(I) = ZERO                                       JAC 320
C      10 CONTINUE                                              JAC 330
C          JINT = L(J)                                           JAC 340
C          XVALUE = ETA(J)                                       JAC 350
C          JJ1 = N - JINT                                         JAC 360
C          E1 = XVALUE - X(JINT)                                  JAC 370
C          E2 = X(JINT+1) - XVALUE                               JAC 380
C          V(1) = ONE/(X(JINT+1)-X(JINT))                        JAC 390
C          NKM = MINO(K,JJ1)                                      JAC 400
C          IF (NKM.EQ.1) GO TO 30                                  JAC 410
C          DO 20 IJ=2,NKM                                         JAC 420
C              JINTPJ = JINT + IJ                                 JAC 430
C              V(IJ) = (E1/(X(JINTPJ)-X(JINT)))*V(IJ-1)         JAC 440
C      20 CONTINUE                                              JAC 450
C      30 IF (JINT.EQ.1) GO TO 60                                  JAC 460
C          MJ = MINO(JINT,K) - 1                                  JAC 470
C          DO 50 IJ=1,MJ                                          JAC 480
C              JINTMJ = JINT - IJ                                 JAC 490
C              E3 = XVALUE - X(JINTMJ)                           JAC 500
C              V(1) = (E2/(X(JINT+1)-X(JINTMJ)))*V(1)          JAC 510
C              NKJ = MINO(K-IJ,JJ1)                              JAC 520
C              .....                                            JAC 530
C              .....                                            JAC 540
C              .....                                            JAC 550

```



```

      IF (NMJ.LE.1) GO TO 50
      DO 40 LI=2,NMJ
        JINTPL = JINT + LI
        V(LI) = (E3*V(LI-1)+(X(JINTPL)-XVALUE)*
          V(LI))/(X(JINTPL)-X(JINTMJ))
40      CONTINUE
50      CONTINUE
60      I1 = MAXO(0,K-JINT)
      KM = MINO(JINT,NK)
      DO 70 I=1,KM
        IPI1 = I + I1
        AJ(J,I) = TWO*((-ONE)**(J-1))*V(IPI1)
70      CONTINUE
80 CONTINUE
C      THE TRANSPOSE OF THE JACOBIAN MATRIX IS STORED IN COMPACT
C      FORM IN THE ARRAY AJ.
C      .....
C      .....
C      BEGIN GAUSSIAN ELIMINATION STEP
C      .....
      IF (NMK.EQ.1) GO TO 130
      NMKM1 = NMK - 1
      DO 120 JCOL=1,NMKM1
        JCOLP1 = JCOL + 1
        LJC = L(JCOL)
        IF (LJC.LT.JCOLP1) GO TO 120
        JINC = MINO(0,K-LJC)
        I1 = JCOL + JINC
        PIVOT = AJ(JCOL,I1)
        DO 110 I=JCOLP1,NMK
          IF (LJC.LT.I) GO TO 120
          I1 = I + JINC
          MULT = AJ(JCOL,I1)/PIVOT
          DO 90 J=JCOLP1,NMK
            LJ = L(J)
            IF (LJ.GE.JCOL+K) GO TO 100
            IINC = MINO(0,K-LJ)
            I1 = I + IINC
            I2 = JCOL + IINC
            AJ(J,I1) = AJ(J,I1) - MULT*AJ(J,I2)
90          CONTINUE
100         B(I) = B(I) - MULT*B(JCOL)
110        CONTINUE
120 CONTINUE
C      END OF ELIMINATION.
C      .....
C      .....
C      BEGIN BACK SUBSTITUTION
C      .....
130 DO 160 IT=1,NMK
      J = NMK - IT + 1
      LJ = L(J)
      JINC = MINO(0,K-LJ)

```

JAC 560
 JAC 570
 JAC 580
 JAC 590
 JAC 600
 JAC 610
 JAC 620
 JAC 630
 JAC 640
 JAC 650
 JAC 660
 JAC 670
 JAC 680
 JAC 690
 JAC 700
 JAC 710
 JAC 720
 JAC 730
 JAC 740
 JAC 750
 JAC 760
 JAC 770
 JAC 780
 JAC 790
 JAC 800
 JAC 810
 JAC 820
 JAC 830
 JAC 840
 JAC 850
 JAC 860
 JAC 870
 JAC 880
 JAC 890
 JAC 900
 JAC 910
 JAC 920
 JAC 930
 JAC 940
 JAC 950
 JAC 960
 JAC 970
 JAC 980
 JAC 990
 JAC 1000
 JAC 1010
 JAC 1020
 JAC 1030
 JAC 1040
 JAC 1050
 JAC 1060
 JAC 1070
 JAC 1080
 JAC 1090
 JAC 1100

I1 = J + JINC	JAC 1110
PIVOT = AJ(J,I1)	JAC 1120
SUM = ZERO	JAC 1130
IF (IT.EQ.1) GO TO 150	JAC 1140
SUM = ZERO	JAC 1150
DO 140 I=1,MMKM1	JAC 1160
JPI = J + I	JAC 1170
IF (JPI.GT.MMK) GO TO 150	JAC 1180
LJPI = L(JPI)	JAC 1190
IF (LJPI.GE.J+K) GO TO 150	JAC 1200
IINC = MINO(0,K-LJPI)	JAC 1210
I1 = J + IINC	JAC 1220
SUM = SUM + AJ(JPI,I1)*Y(JPI)	JAC 1230
140 CONTINUE	JAC 1240
150 Y(J) = (-B(J)-SUM)/PIVOT	JAC 1250
160 CONTINUE	JAC 1260
C END OF BACK SUBSTITUTION.	JAC 1270
C *****	JAC 1280
RETURN	JAC 1290
END	JAC 1300

REFERENCES

1. H. B. Curry, I. J. Schoenberg, "On Pólya Frequency Functions, IV." J. Analyse Math. 17, pp. 71-107. Also: Bull. Amer. Math. Soc., Abstract 380t, 53, pp. 1114 (1947).
2. G. de Boor, "A Practical Guide to Splines," Applied Mathematical Sciences, Series Number 27, Springer-Verlag Publications, pp. 222-227 (1978).
3. J. Dorrenbacher, D. Paddock, D. Wisneski, and L. D. Fosdick, "POLISH, A FORTRAN Program to Edit FORTRAN Programs," University of Colorado, Department of Computer Science Technical Report #CU-CS-050-76 (Revised), August 1979.
4. P. W. Gaffney, (1978), "To Compute the Optimal Interpolation Formula," Mathematics of Computation, Vol. 32, No. 143, July 1978.
5. P. W. Gaffney, "Optimal Interpolation," Oxford University, D. Phil. Thesis (1977).
6. P. W. Gaffney and M. J. D. Powell, "Optimal Interpolation," Proceedings of the Conference on Numerical Analysis Held at University of Dundee; Edited by G. A. Watson, Springer-Verlag, Number 506 in Lecture Notes in Mathematics Series (1975).
7. P. W. Gaffney, "The Range of Possible Values of $f(x)$," J. Institute Maths. Applics., Vol. 21, pp. 211-226 (1978).
8. P. W. Gaffney, "FORTRAN Subroutines for Computing the Optimal Interpolation Formula," A.E.R.E. Report No. R8781 (1977).
9. P. W. Gaffney, "Spline Interpolation - Optimal Knot or Not?," (to appear).
10. L. J. Osterweil and L. D. Fosdick, "DAVE - A Validation Error Detection and Documentation System for FORTRAN Programs," Software Practice and Experience 6, pp. 473-486 (1976).
11. B. G. Ryder, "The PFORT Verifier," Software Practice and Experience 4, pp. 359-377 (1974).