Least Squares Cubic Spline Approximation II — Variable Knots

Carl de Boor and John R. Rice

April 1968

Department of Computer Sciences
Purdue University

CSD TR 21

Versions of the spline programs of de Boor and Rice are available in the program library of IMSL as `ICSFKU` and `ICSVKU`. Contact

International Mathematical & Statistical Libraries
GNB Building
7500 Bellaire
Houston, Texas 77036

Retyped March 1994

Least Squares Cubic Spline Approximation II — Variable Knots

Carl de Boor[1] and John R. Rice[2]

1 Introduction

This paper presents an extension and application of the algorithm in [2]. We refer to this algorithm as `FIXEDKNOT`. `FIXEDKNOT` is for the computation of least-squares approximations on finite point sets by cubic polynomial splines with fixed knots. The algorithm presented here incorporates `FIXEDKNOT` and treats the knots as variables. The spline depends nonlinearly on the knots and thus we have a nonlinear least-squares approximation problem to solve.

`FIXEDKNOT` has several features specifically designed to facilitate its incorporation into an algorithm which treats the knots as variables. The algorithm presented here is only one of several approaches to varying the knots and some of these other approaches are discussed in general terms at the end of this paper.

An algorithm which varies the knots is desirable because it greatly increases the flexibility of the spline approximants. This flexibility can also be achieved by simply increasing the number of knots. This, of course, increases the complexity of the approximant obtained. There are situations where this increase is an efficient approach since it is difficult and time consuming to solve the nonlinear approximation problem. However, this increase in complexity is very undesirable in an application which involves smoothing or the representation of physical data or shapes. In such an application one must minimize the number of parameters involved in order to obtain the best results. Thus this algorithm is primarily useful for obtaining low to medium accuracy (2 to 5 significant digits) approximations of functions of a more or less arbitrary nature. It has, for example, been used to obtain 3 significant digit approximations to curves with 8 or 10 local extrema and which have a completely unsystematic nature.

Note that this algorithm should *not* be used for high accuracy approximations to mathematically defined functions (e.g., for computer system function subroutines). The degree of convergence for spline approximation is such that this is very unlikely to be efficient unless the polynomial degree is rather high. See [1] and [5] for further results and compare these with polynomial and rational approximations [3].

[1] This work was initiated at the General Motors Research Laboratories. The final stages were partially supported by NSF grant GP-7163. We wish to thank John Hoff for assistance in preparing preliminary versions of this algorithm.

[2] This author was also partially supported by NSF grant GP-4052.

2 Mathematical Background

We assume that the reader is familiar with `FIXEDKNOT` and we use the notation of that paper. We recall that a spline of degree n with k knots $\Xi = \{\xi_i | a = \xi_0 < \cdots < \xi_{k+1} = b\}$ may be defined by

$$S(A, \Xi, x) = \sum_{i=1}^{k} a_i (x - \xi_i)_+^n + \sum_{j=0}^{n} a_{k+j+1} x^j$$

where $A = (a_1, a_2, \ldots, a_{k+n+1})$.

We consider a function $f(x)$ defined on a finite set

$$X = \{x_i | a \leq x_i < x_{i+1} \leq b, \quad i = 1, 2, \ldots, m\}.$$

Given a value n for the degree and a number h of knots we have the

Approximation Problem. Determine the spline $S(A^*, \Xi^*, x)$ so that

$$[\int [f(x) - S(A, \Xi, x)]^2]^{\frac{1}{2}} \tag{2.1}$$

is minimized among all splines of degree n with k knots.

Since $f(x)$ is only defined on the finite set X, one must use a quadrature formula for the integral in this problem. We assume this is to be done (our algorithm uses the trapezoidal rule), but retain the integral sign for simpler notation.

There are three basic mathematical questions associated with this problem, namely those of the existence, uniqueness and characterization of $S(A^*, \Xi^*, x)$. We discuss these briefly.

The Existence Question. Simple examples show that this least-squares approximation problem does not always have a solution, e.g., take $f(x) = |x|$ on [–1,+1] and approximate by a cubic spline with three knots. One may generalize the concept of spline by allowing the knots to coalesce with the possibility of a resultant loss of smoothness where the knots coalesce. These are called *extended splines* and are presented in [6], see also [4]. In this broader set of approximating functions there always exists a least-squares approximation. In order to avoid technical difficulties, the algorithm presented in this paper does not allow the knots to coalesce.

The Uniqueness Question. It is known from general theoretical results [6], from specific theoretical results [C. de Boor, 1963, unpublished] and from examples that the solution of the least-squares nonlinear approximation problem need not be unique. Consider the approximation to x^3 on [–1,+1] by a broken line with one break. If the break occurs for $x = 0$, then by symmetry there is no break. But no least-squares nonlinear spline approximation can have an inactive knot (see the next section). Thus the best approximation does not have a knot at $x = 0$ and, again by symmetry, there are at least two best approximations. This line of reasoning can be applied in general.

Furthermore, there may be approximations which are local minima of (2.1), but which are not best approximations. The algorithm presented here attempts to obtain a *local* minimum of (2.1) and hence even if it converges there is no guarantee that a best approximation has been obtained.

Characterization. There are no known necessary and sufficient conditions for $S(A^*, \Xi^*, x)$ to be a best approximation. The algorithm here is based on the usual necessary conditions that one derives for a local minimum.

Strict Monotonicity of the Error. It is known [de Boor, 1963, unpublished] for any specific $f(x)$ and fixed degree n that if the error

$$E_k^2 = \int [f(x) - S(A^*, \Xi^*, x)]^2$$

of the best approximation with $k+1$ knots is not zero, then the error E_{k+1} of the best approximation with $k+1$ knots is strictly less than E_k, i.e., $E_{k+1} < E_k$. See [4] for details and extensions.

Inherent Limitations of the Algorithm. The problem which this algorithm attempts to solve cannot be solved by an algorithm. Thus this algorithm is limited. This theoretical limitation is manifested in several different ways. First, there is the problem of ascertaining when "convergence" has taken place. This is required on two different levels, namely, for the whole algorithm and for the adjustment of knots within this latter problem. The decision that "convergence" has taken place is made on the basis of certain *ad hoc* numerical tests which are not infallible.

These decisions are delicate in view of the need to achieve some efficiency. Thus these tests have been developed on the basis of experience with a certain class of problems. It is hoped that this class is representative of those met in general. However, these tests may be completely inadequate in new situations. If it is intended to use this algorithm extensively for a certain class of problems, it may well pay to experiment with adjustments in these tests in order to achieve better efficiency with minimum risk.

The initial guess for the knots chosen might be extremely poor and result in reaching a local minimum far from the best approximation. The simple scheme of equal spacing used here to obtain an initial guess might well be modified and improved for certain classes of approximation problems.

3 The Algorithm and Numerical Procedures

The basic idea of the algorithm is to vary the knots one by one so as to decrease the L_2-error. This is done systematically from right to left by two procedures, `SWEEP` and `OPT`. The procedure `SWEEP` controls the overall scheme and `OPT` does the variation of the individual knots. The basic scheme used in `OPT` is a discrete Newton's method.

There are two delicate points in an implementation of such a scheme. The first is a suitable choice of termination criteria for the various iterations in the algorithm. One desires to achieve the required accuracy without doing an excessive amount of wasteful computation.

The second point is to make the computation of the L_2-error as efficient as possible. It is unavoidable that this number be evaluated frequently and it is a nontrivial computation. Furthermore, it is easily seen that it is very inefficient to compute the L_2-error each time by a standard L_2-approximation procedure. Note that if only one knot is changed and if only one of the orthogonal functions involves this knot, then the L_2-approximation problem can be solved on the basis of previous information with an order of magnitude less computation than one can solve such problems in general. It is always arranged so this is the case and the procedure `FIXEDKNOT` has a number of features to allow this. More detailed study shows that it is also possible to make use of some previous information when changing from one knot to another. These points are discussed in more detail in [2].

Choice of the Initial Knots. There are two single alternatives. If `NOKNOT` is negative, then –`NOKNOT` knots are chosen equally spaced in the interval (`XX(1)`, `XX(LX)`). If `NOKNOT` is positive, then this number of knots is to be read as data. If the function is very unsystematic, it is often profitable to use an initial set of knots concentrated in the regions of rapid change in the function.

Optimization of the Knots – `SWEEP` *and* `OPT`. Given an initial set of knots, their optimization is guided by the procedure `SWEEP`. Each knot is, in turn, varied so as to minimize the L_2-error as a function of this knot. This is started with the last (i.e., the rightmost) interior knot and done sequentially to the left. A cycle refers to one complete pass from right to left. This process is repeated until a termination is encountered.

The variation of the `I`-th knot `XI(I)` is carried out in `OPT` using what may be termed the "discrete Newton's" method. Let $e(t)$ denote the L_2-error as a function of the position t of `XI(I)`. Given three points, `ALEFT`<A<`ARIGHT`, a new guess `ABEST` for the location of `XI(I)` is determined as the location of the minimum of the parabola $p(t)$ satisfying

$$p(\texttt{ALEFT}) = e(\texttt{ALEFT}),\ p(\texttt{A}) = e(\texttt{A}),\ p(\texttt{ARIGHT}) = e(\texttt{ARIGHT}).$$

The parabola must have a minimum in order for this to make sense. Also, as to avoid getting wild guesses through extrapolation, `ABEST` should be between `ARIGHT` and `ALEFT`. For this it is sufficient to have

$$e(\texttt{ARIGHT}), e(\texttt{ALEFT}) \geq e(\texttt{A}). \tag{3.1}$$

Thus the first part of OPT consists of a search algorithm for such a set of three points ARIGHT, ALEFT and A. The basic step size for this search is based on the value of CHANGE = average change in the knots in the preceding cycle. The initial value of CHANGE is .4 and it is measured relative to the length of the interval (XI(I-1), XI(I+1)).

Once such a set is found the parabolic interpolation commences. The newly found guess ABEST replaces one of ARIGHT, ALEFT or A in such a way that the inequalities (3.1) remain valid while making the new value of ARIGHT–ALEFT as small as possible.

Termination Criteria. There are two termination criteria for SWEEP. The first is a simple bound on the number of complete cycles or sweeps of varying all the knots, i.e.,

$$\text{No more than } \texttt{ITER} \text{ cycles throught } \texttt{SWEEP} \tag{3.2}$$

For normal use we recommend that one set ITER=4. In more difficult cases, especially when a larger number of knots is used, one might need to increase ITER. The second criterion is to terminate if

$$\left|\frac{\texttt{PREVER-ERROR}}{\texttt{ERROR}}\right| \leq .4 * \texttt{ACC} \tag{3.3}$$

where ACC = desired accuracy in L_2-error (*not* the L_2-error itself), ERROR = current value of the L_2-error and PREVER = value of the L_2-error at the start of the current cycle of knot variation. This criterion is based on the assumption that the algorithm is converging linearly (or faster) and the error is reduced at each cycle by a factor of .6 or less. If one notes that the algorithm is converging somewhat slower than this, one should replace the coefficient .4 by a somewhat smaller number.

Note that this is rarely worthwhile to compute an approximation which gives the best L_2-error with more than one or two significant digits. We recommend setting ACC = .1 for general applications.

There are four termination criterion for OPT. The first is a simple bound on the number of guesses at the best position of XI(I), i.e.,

$$\text{No more than } \texttt{INDLP} \text{ guesses for } \texttt{XI(I)}. \tag{3.4}$$

We recommend INDLP = 10, a bound which is large enough so that termination rarely occurs from this criterion.

The second criterion is a form of buffering to prevent the knots from coalescing. Set $\mathtt{H} = \mathtt{XI(I+1)} - \mathtt{XI(I-1)}$, then constrain XI(I) by

$$\mathtt{XI(I-1)} + .0625\,\mathtt{H} \leq \mathtt{XI(I)} \leq \mathtt{XI(I+1)} - .0625\,\mathtt{H} \tag{3.5}$$

This form of constraint allows a group of knots to become very closely spaced which is sometimes essential. However, it keeps them separated enough to (almost always) avoid failure due to numerical instabilities.

The third criterion is for the search of a triplet of points to initialize the parabolic interpolation phase. The search for such a triplet is terminated if (in the case of search to the right)

$$\frac{e(\mathtt{A}) - e(\mathtt{ARIGHT})}{\mathtt{ERROR}} \leq \frac{\mathtt{ACC}}{\mathtt{LXI}} \tag{3.6}$$

where LXI = number of interior knots and ERROR is the L_2-error at the end of the previous cycle. In case of search to the left we terminate if

$$\frac{e(\mathtt{A}) - e(\mathtt{ALEFT})}{\mathtt{ERROR}} \leq \frac{\mathtt{ACC}}{\mathtt{LXI}}. \tag{3.7}$$

These criteria are relatively stringent because we feel it is very desirable to be able to enter the parabolic interpolation phase for at least one time. Thus this criterion might not cause termination in OPT even when the decrease in the L_2-error is insignificant for the later stages of the algorithm in a reasonable number of cases.

The criterion can be visualized as based on the assumption that the search is converging linearly (or faster) with an error reduction of 1–1/LXI or smaller. However, the situation here is somewhat different than in SWEEP as we do not necessarily desire to expend effort for an accurate placement of XI(I). That is to say, in the initial stages of the algorithm the set of knots is far enough from optimum that it is wasteful to accurately optimize one of them with the others inaccurately located. It is unusual for this termination criterion to be active in the terminal phases of the algorithm.

The fourth criterion is for the termination of the parabolic interpolation process. We locate ABEST as noted above and compute the value EPRED of the parabola at its lowest point, i.e., EPRED = p(ABEST). The optimization is terminated if

$$\left|\frac{\mathtt{EPRED} - e(\mathtt{ABEST})}{\mathtt{ERROR}}\right| \leq 5 * \mathtt{ACC} \tag{3.8}$$

This criterion assumes convergence which is somewhat faster than linear. This is plausible since a discrete Newton method is used. The particular factor 5 was chosen on the basis of some experiments and reflects a balance between global efficiency and local accuracy as discussed in the preceding paragraph.

The most common cause for termination is that CHANGE become small. This implies that little movement of the knots takes place in OPT which in turn causes the criterion (3.3) in SWEEP to terminate the algorithm.

4 Variables in the Program

Global with FIXEDKNOT

ADDXI(26)	LX
COEFL(27,4)	MODE
FCTL(100)	U(100)
INTERV	UERROR(100)
JADD	VORDL(28,2)
KNOT	XIL(28)
LMAX	XX(100)
debug	

Other Variables

AA	ELEFT
AHIGH	ERIGHT
ALOW	ETRY
DEL	II
DELX	ITRR
DUMB	K
DXLEFT	LPCNT
DXRGHT	LXI1 = LXI+1
DYLEFT	LXI2 = LXI+2
DYRGHT	SGN
E	BD

Global in VARYKNOT

ACC	LXI
CHANGE	Q
ERROR	XI(28)

Other Important Variables

A	INFO(20)
ABEST	INTER
ALEFT	KVARY
ARIGHT	NOKNOT
EPRED	PREVER
EPSERR	K

5 Example

We consider a set of data which has three distinct features: (i) It is actual data (expressing a thermal property of titanium); (ii) It is difficult to approximate using classical techniques;

(iii) There is a significant amount of noise in the data.

Titanium Heat Data `XX(I)`, `U(I)` with approximation `U*(I)` and error `UERROR(I)`

XX	U	U*	UERROR×10^3	XX	U	U*	UERROR×10^3
595	.644	.619	25.35	845	.812	.796	15.86
605	.622	.629	–7.22	855	.907	.876	31.27
615	.638	.638	.24	865	1.044	1.051	–7.38
625	.649	.644	4.50	875	1.336	1.370	–34.31
635	.652	.650	2.35	885	1.881	1.838	42.64
645	.639	.653	–14.46	895	2.169	2.195	–25.98
655	.646	.656	–10.13	905	2.075	2.078	–3.19
665	.657	.658	–.89	915	1.598	1.582	15.62
675	.652	.659	–6.96	925	1.211	1.197	14.15
685	.655	.660	–4.57	935	.916	.931	–14.63
695	.664	.660	4.05	945	.746	.761	–15.36
705	.663	.660	2.69	955	.672	.667	5.31
715	.663	.661	2.13	965	.627	.624	2.71
725	.668	.662	6.12	975	.615	.612	3.20
735	.676	.664	12.47	985	.607	.608	–1.24
745	.676	.666	9.93	995	.606	.606	.15
755	.686	.670	16.29	1005	.609	.604	4.62
765	.679	.675	4.32	1015	.603	.604	–.65
775	.678	.681	–3.20	1025	.601	.603	–2.49
785	.683	.689	–6.49	1035	.603	.604	–.74
795	.694	.700	–5.78	1045	.601	.604	–3.22
805	.699	.712	–13.29	1055	.611	.605	6.24
815	.710	.727	–17.25	1065	.601	.605	–4.19
825	.730	.745	–14.87	1075	.608	.605	2.66
835	.763	.765	–2.39				

We present two approximations to this data. The first is computed with an initial set of 7 equally spaced knots in the interval (595, 1075). The second is computed with another initial set of knots. This is the approximation shown in the above table.

Initial Knots

Case 1:	595	675	755	835	915	995	1075
Case 2:	595	725	850	910	975	1040	1075

The point of these two cases is that the algorithm converges to two distinct local minima of the nonlinear least-squares approximation problem. Note that the data have a very pronounced peak near 900, and in Case 1 we have three interior knots to the left of this peak, while in Case 2 we have only two to the left of this peak.

The final approximations obtained are presented for both cases. The final knots are given along with the coefficients `C(I)`, $\mathtt{I} = 0, 1, 2, 3$ of the cubic polynomial pieces of the spline. These are the coefficients `COEFL(I,J)`, $\mathtt{J} = 1, 2, 3, 4$ defined in [2] for the interval $[\xi_{\mathtt{I}}, \xi_{\mathtt{I+1}}]$. The origin for each polynomial piece is the knot $\xi_{\mathtt{I}}$, immediately to the left.

Case 1			**Case 2**		
Least Square Error	=	03489	Least Square Error	=	.01305
Average Error	=	.02296	Average Error	=	.00933
Maximum Error	=	.11716	Maximum Error	=	.04264

KNOTS	Cubic Coefficients	KNOTS	Cubic Coefficients
595.	C(0) = .63371	595.	C(0) = .61865
	C(1) = $.16475^{-3}$		C(1) = $.11658^{-2}$
	C(2) = $.19591^{-5}$		C(2) = $-.11255^{-4}$
	C(3) = $-.81758^{-8}$		C(3) = $.37272^{-7}$
755.28	C(0) = .67678	835.32	C(0) = .76609
	C(1) = $.16269^{-3}$		C(1) = $.22139^{-2}$
	C(2) = $-.19723^{-5}$		C(2) = $.15616^{-4}$
	C(3) = $.17472^{-6}$		C(3) = $.78696^{-5}$
839.60	C(0) = .78122	876.56	C(0) = $.14362^{+1}$
	C(1) = $.35567^{-2}$		C(1) = $.43668^{-1}$
	C(2) = $.42224^{-4}$		C(2) = $.98940^{-3}$
	C(3) = $.92733^{-5}$		C(3) = $-.61055^{-4}$
877.06	C(0) = $.14612^{+1}$	902.46	C(0) = $.21703^{+1}$
	C(1) = $.45759^{-1}$		C(1) = $-.27909^{-1}$
	C(2) = $.10844^{-2}$		C(2) = $-.37536^{-2}$
	C(3) = $-.78416^{-4}$		C(3) = $.18772^{-3}$
896.20	C(0) = $.21844^{+1}$	910.47	C(0) = $.18022^{+1}$
	C(1) = $.10478^{-2}$		C(1) = $-.51906^{-1}$
	C(2) = $-.34197^{-2}$		C(2) = $.75881^{-3}$
	C(3) = $.91502^{-4}$		C(3) = $-.37241^{-5}$
910.22	C(0) = $.17793^{+1}$	977.85	C(0) = .61061
	C(1) = $-.40887^{-1}$		C(1) = $-.37235^{-3}$
	C(2) = $.42760^{-3}$		C(2) = $.60407^{-5}$
	C(3) = $-.13771^{-5}$		C(3) = $-.28471^{-7}$
1075.		1075.	

The algorithm required six cycles through `SWEEP` for Case 1. The L_2-error decreased as follows:

cycle	1	2	3	4	5	6
L_2-error	.09176	.05927	.03944	.03588	.03509	.03489

The algorithm required seven cycles through `SWEEP` for Case 2. The L_2-error decreased as follows:

cycle	1	2	3	4	5	6	7
L_2-error	.04595	.03848	.02761	.02177	.01432	.01321	.01305

These two cases required about 17 and 23 seconds, respectively, of execution time on a `IBM` 7094 for a `FORTRAN IV` version of this algorithm. They required about xxx and yyy seconds, respectively, of execution time on a `CDC` 6500 in Algol.

6 Other Nonlinear Algorithms Based on `FIXEDKNOT`

The procedure `FIXEDKNOT` is designed to be readily adaptable to form a basis for a variety of nonlinear spline approximation algorithms. We briefly outline four such algorithms. We have used one of these (the last one) extensively and another (the second one) in some experimentations.

6.1 Non-systematic knot optimization

We have observed that there is frequently a significant amount of wasted computation in problems involving a larger number of knots, say more than 5 or 6. It occurs that a few, perhaps most, of the knots become correctly placed, while the remaining ones (somewhat more delicate) requires several additional cycles to locate accurately. The systematic nature of the algorithm `VARYKNOT` requires one nevertheless to adjust the position of all knots in each cycle. It is clear to us that one can devise workable criteria for determining reasonably well which knots are more critical. One could use these criteria to optimize the knots in an unsystematic manner to increase the efficiency of the computation. We have not formalized such criteria and believe their use would significantly increase the logical complexity of the algorithm.

6.2 Systematic insertion of additional knots – L_∞ criterion

A plausible scheme is to start out with no knots at all, find the best linear cubic approximation, then insert a knot near (or at) the location of the maximum error. One then could compute a linear spline approximation with one knot, and insert a second knot near the location of the maximum error. This process is then repeated until the error is reduced to some desired level.

We have experimented with this scheme and it does in fact work. Special provisions must be made if the data contain wild points or pronounced peaks. The maximum error will then occur several times at one point. The new knots should be placed on alternating sides of this point and prevented from converging to this point. It usually happens that enough knots are placed in the neighborhood of a wild point so that the data are actually interpolated nearby. This is normally undesirable and this scheme is not recommended for such data.

This scheme is not as attractive as we had expected. In addition to the problem of wild points and peaks, it consistently leads to more knots than really required, sometimes excessively so. However, it usually requires less computation time than schemes (e.g., see 6.4) which optimize the locations of knots. Thus when this process was applied to the data of the example, it took 15 interior knots to produce an approximation of the same accuracy as had been obtained in Case 2 above with an optimal placing of 5 interior knots. Execution time, on the other hand, on an `IBM` 7094, was merely 4 seconds. We conclude that the location of the maximum error is not a completely reliable guide for the place to insert additional knots.

6.3 Systematic insertion of additional knots – L_2 criterion

Consider a process like 6.2 where we locate that interval between adjacent knots which has the most error in the L_2 sense. We suspect that it is better to insert additional knots into this interval than near the location of the maximum error. We have not tested this suspicion, however.

6.4 Systematic insertion of knots with optimization

We have used extensively an algorithm which systematically increases the number of knots and optimizes all knots after each insertion. This algorithm only requires the user to specify the desired accuracy of approximation and the algorithm determines the number as well as the location of the knots. In order to achieve efficiency, the convergence criteria during the algorithm must depend on how close one is to the requested accuracy. Once this matter is satisfactorily settled, we find that it requires only slightly longer to obtain suitable approximations with this scheme than it does with `VARYKNOT` starting with the correct number of knots roughly placed.

Note that the algorithm is essentially different from that of 6.2. Even though the initial guess at the new knot locations is made similarly, the optimization process eliminates the difficulties with wild points. In fact, it is highly recommended for data smoothing, the identification of wild points and other types of data analysis.

7 References

1. G. Birkhoff and C. de Boor, Error bounds for cubic spline interpolation, *J. Math. Mech.* **13** (1964), 827–835.

2. C. de Boor and J.R. Rice, Least squares cubic spline approximation I-Fixed knots. Technical Report CSD-TR 20, Computer Sciences, Purdue University (1968).

3. J.F. Hart et. al., *Computer Approximations*, John Wiley, New York (1986).

4. C.R. Hobby and J.R. Rice, Approximation from a curve of functions, *Arch. Rat. Mech.* **24** (1967), 91–106.

5. A. Meir and A. Sharma, Degree of approximation of spline interpolation, *J. Math. Mech.* **15** (1966), 759–767.

6. J.R. Rice, *The approximation of functions*, Vol II, Chapter 10, Addison Wesley (1969).

```
C     PROGRAM SPLINE(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C                   NONLINEAR SPLINE APPROXIMATION
C     PROGRAM WRITTEN BY CARL DE BOOR AND JOHN RICE
C                         PURDUE UNIVERSITY
C     SUPPORTED BY THE NATIONAL SCIENCE FOUNDATION   GP-4052,GP-7163
C
C     PLEASE REPORT ANY CASES OF INOPERATION TO THE AUTHORS.
C                               THANKS
C     ****            NUMERICAL ANALYSIS CONTROL                 ****
C          CONTROL PARAMETERS          FUNCTION
C            ITER                        NO. OF SWEEPS THRU OPT
C            BD (IN OPT)                 IMPROVEMENT NEEDED TO REPEAT
C            EPSERR( IN SWEEP)           IMPROVEMENT NEEDED TO REPEAT
C            DIST (IN OPT NEAR 30,80)    KEEPS KNOTS SEPARATED
C            INDLP                       NO. OF PASSES THRU OPT
C     THE FOLLOWING IS THE MAIN PROGRAM FOR VARYKNOT
C
      IMPLICIT NONE
      CHARACTER *4 INFO(20)
      LOGICAL debug
      real ACC,ADDXI(26),CHANGE,COEFL(27,4),DEL,DUMB,ERROR,
     1     FCTL(100),
     2     FXDKNT,
     3     U(100),UERROR(100),VORDL(28,2),Q,
     4     XI(28),XIL(28),XX(100)
      integer I,IABS,IERROR,INTERV,ITER,
     1     J,JADD,KNOT,L,LMAX,LX,LXI,LXI1,LXI2,MODE,NOKNOT
C
c     COMMON INPUT SERVES AS INPUT TO FXDKNT
C                SEE FXDKNT FOR DEFINITIONS OF VARIABLES
      COMMON/INPUT/LX,XX,U,JADD,ADDXI,MODE,debug
C     COMMON OUTPUT SERVES AS OUTPUT FROM FXDKNT
C                SEE FXDKNT FOR DEFINITIONS OF VARIABLES
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
C
C     COMMON OTHER SERVES AS COMMUNICATION BETWEEN OPT,SWEEP AND HERE
C         LXI = NUMBER OF INTERIOR KNOTS,  LXI1 = LXI+1, LXI2 = LXI+2
```

```
C         Q   = NUMERICAL CONTROL VARIABLE USED BETWEEN OPT AND SWEEP
C         CHANGE = DITTO
C         ERROR =  CURRENT VALUE OF THE L-2 ERROR - SQUARED
C         ACC   = DESIRED ACCURACY OF L-2 ERROR
C         XI(28)= ARRAY FOR KNOTS
      COMMON/ OTHER / LXI,LXI1,LXI2,Q ,CHANGE,ERROR ,ACC, XI
      debug = .false.
C
C     ACC = .1 AND ITER = 4 TO 8 SEEM TO BE GOOD VALUES FOR TYPICAL USES
      ACC = .1
      ITER = 8
      ITER = 2     ! temporary
C
C  ***INFO IS SIMPLY AN IDENTIFICATION OF THE DATA***
    1 READ(5,550, END = 40), (INFO(I), I=1,20)
  550 FORMAT(20A4)
      WRITE(6,551), (INFO(I), I=1,20)
  551 FORMAT( 20A4)
C
C          READ IN NO. OF POINTS=LX AND THE DATA XX AND U
C  *** IF NOKNOT.GE.2, THEN READ IN LXI2=NOKNOT KNOTS***
C  ***    OTHERWISE PROGRAM CHOOSES LXI2 =-NOKNOT EQUISPACED KNOTS   ***
      READ *,  NOKNOT, LX, (XX(I), U(I), I=1,LX)
      LXI2 = IABS(NOKNOT)
C  IF *NOKNOT* IS .GT. 0, READ IN NOKNOT KNOTS (INCL.BOUNDARY POINTS.)
      IF (NOKNOT .GT. 0) READ *, (XI(J),J=1,LXI2)
C
C       **CHECK ON GIVEN DATA
C     THESE CHECKS PREVENT USER FROM EXCEEDING BOUNDS ON STORAGE
C                          AND FROM PRESENTING UNORDERED XX ARRAY
      IERROR = 0
      IF (LX .GE. LXI2+2 .AND. LX .LE. 100) GO TO 3
      WRITE(6,662) LX
  662 FORMAT(' NO. OF DATA POINTS, LX = ',I4,
     1    ',NOT WITHIN THE BOUNDS ABS(NOKNOT)+2 AND 100')
      IERROR = 1
                                    GO TO 7
```

```
    3 IF (LXI2 .GE. 3 .AND. LXI2 .LE. 28)  GO TO 4
      WRITE(6,660) NOKNOT
  660 FORMAT('1KNOT CONTROL PARAMETER NOKNOT =',I3,
     1    ' NOT WITHIN BOUNDS.')
      IERROR = 1
    4 DO 6 L=2,LX
         IF (XX(L) .GT. XX(L-1))       GO TO 6
         WRITE(6,664) L,XX(L),U(L)
  664    FORMAT(' DATA POINT ',I4,2F14.8,' NOT IN ASCENDING ORDER.')
         IERROR = IERROR + 1
    6    CONTINUE
      IF (IERROR .LT. 1)               GO TO 14
    7 WRITE(6,666) IERROR
  666 FORMAT(' *** CORRECT INDICATED ',I3,
     1    ' INPUT ERROR(S) AND RESTART.')
      GO TO 1
C
C       **INITIALIZE
   14 IF (NOKNOT .GT. 0)               GO TO 30
C
C              WHEN NOKNOT IS NEG., INTRODUCE -NOKNOT EQUISPACED KNOTS
      XI(1) = XX(1)
      XI(LXI2) = XX(LX)
      DEL = (XX(LX) - XX(1))/FLOAT(LXI2-1)
      DO 26  J = 3,LXI2
         XI(J-1) = XI(J-2) + DEL
   26 continue
C
C                          SET UP INITIAL APPROXIMATION
   30 ADDXI(1) = XI(1)
      ADDXI(2) = XI(LXI2)
      LXI1 = LXI2-1
      LXI = LXI1-1
      MODE = 0
      JADD = LXI2
      DO 35 J = 3,LXI2
         ADDXI(J) = XI(J-1)
```

```
   35 continue
      ERROR = FXDKNT(0.)
C      ***NOTE.   MODE HAS BEEN  SET EQUAL TO 1
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,611) (L,XX(L),U(L),UERROR(L), L=1,LX)
  611 FORMAT(' GIVEN DATA AND ERROR IN FIRST APPROXIMATION'//
     1                      (I4,3F14.8))
      WRITE(6,612) NOKNOT,ITER
  612 FORMAT(' '/' NO. OF INITIAL KNOTS =',I3/
     1       ' ITER =',I3)
      WRITE(6,900) (XI(I), I=1,LXI2)
  900 FORMAT(' KNOTS PRIOR TO OPTIMIZATION'/(9F12.6))
C
C                            OPTIMIZE KNOTS
      CALL SWEEP(ITER)
C
      WRITE(6,640)
  640 FORMAT(49X,'***  FINAL OUTPUT  ***'///)
      MODE = 1
      JADD = 0
      DUMB = FXDKNT(1.)
                                      GO TO 1
   40                                 STOP
C
      END
C
C***********************************************************************
C
      SUBROUTINE SWEEP(ITRR)
C
C        KVARY+1 = INDEX OF KNOT       BEING VARIED
C        SUBROUTINE OPT(I) OPTIMIZES ITH INTERIOR KNOT
C
      IMPLICIT NONE
      LOGICAL debug
      real ACC,ADDXI(26),CHANGE,COEFL(27,4),DUMB,EPSERR,ERROR,
     1     FCTL(100),
```

```
     2     FXDKNT,PREVER,Q,
     3     U(100),UERROR(100),VORDL(28,2),
     4     XI(28),XIL(28),XX(100)
      integer I,INTERV,ITER,ITRR,
     1     JADD,K,KNOT,KVARY,LMAX,LX,LXI,LXI1,LXI2,MODE
      COMMON/INPUT/LX,XX,U,JADD,ADDXI,MODE,debug
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
      COMMON/ OTHER / LXI,LXI1,LXI2,Q ,CHANGE,ERROR ,ACC,     XI
C         AT ALL TIMES, ERROR CONTAINS (L2 ERROR)**2 OF CURRENT B.A.
C
      ITER = ITRR
C       **NEXT      CARDS SET NUMERICAL ANALYSIS CONTROLS
      EPSERR =    ACC/2.5
      CHANGE = .4*FLOAT(LXI)
C
   10 KVARY = LXI
      Q = CHANGE/FLOAT(LXI)
      if (.not. debug)        go to 11
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE (6,902) ITER,Q
  902 FORMAT (' ITER, Q  ',I5,E20.8)
   11 CHANGE = 0.
      PREVER = ERROR
      MODE = 2
      JADD = 0
      KNOT = KNOT - 1
      DUMB = FXDKNT(0.)
   20 CONTINUE
      if (.not. debug)        go to 21
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,900) KVARY
  900 FORMAT(' '///' VARYING',I4,'TH INTERIOR KNOT')
      WRITE(6,901) ERROR
  901 FORMAT(' SQ. OF L2-ERROR ',E16.6)
C
   21 CALL OPT(KVARY)
      KVARY = KVARY -1
```

```
      JADD = JADD + 1
      IF( JADD .LE. 1 )       GO TO 25
      K= JADD
      DO 22 I = 2,JADD
      K= K-1
   22 ADDXI(K+1) = ADDXI(K)
   25 ADDXI(1) = XI(KVARY + 2)
      KNOT = LXI1 - JADD
      MODE = 2
      DUMB = FXDKNT(0.)
      IF( KVARY .NE. 0 )      GO TO 20
C        THE LAST CALL TO FXDKNT PRODUCES THE B.A. USING ALL KNOTS,
C        SINCE THEN ADDXI CONTAINS ALL KNOTS
      ERROR = DUMB
C     *** THE FOLLOWING TWO CARDS PRODUCE PRINTED OUTPUT OF L1,L2,L-INF
C *   JADD = 0
C *   DUMM = FXDKNT(2.)
C
C       **IF CHANGE IN ERROR IS BIG ENOUGH MAKE ANOTHER SWEEP, ELSE QUIT
      IF (PREVER-ERROR .LE. EPSERR*PREVER)    GO TO 60
      ITER = ITER-1
C
C       **CHECK NUMBER OF PASSES THROUGH SWEEP
      IF(ITER.GT.0)           GO TO 10
   40 CONTINUE
C
C          IN FINAL VERSION GO TO 40, GO TO 60 ARE REPLACED BY RETURN
      if (.not. debug)        return
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,620)
  620 FORMAT (' *** NO. OF ALLOWABLE SWEEPS USED UP')
                              RETURN
   60 CONTINUE
      if (.not. debug)        return
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,610)
                              RETURN
```

```
  610 FORMAT(' *** SWEEP DISCONTINUED - INSUFFICIENT CHANGE IN ERROR')
      END
C
C*************************************************************************
C
      SUBROUTINE OPT(II)
C
C     I REFERS TO THE ITH INTERIOR KNOT
C     OPT FINDS THE OPTIMAL ITH KNOT BETWEEN THE I-1ST AND I+1ST KNOTS
C     THE REMAINING KNOTS ARE HELD FIXED.
C         INDLP = A BOUND ON THE NUMBER OF TRIES ALLOWED
C                 FOR IMPROVEMENT OF THE ITH KNOT
C         Q = MULTIPLICATION FACTOR WHICH SHOULD DECREASE AS A
C             FUNCTION OF THE NO. OF SWEEPS THRU SWEEP
C             Q IS ALTERED IN SWEEP
C
      IMPLICIT NONE
      LOGICAL debug
      real A,AA,ABEST,ACC,ADDXI(26),AHIGH,ALEFT,ALOW,ARIGHT,BD,
     1     CHANGE,COEFL(27,4),DEL,DIFF,DIST,DXLEFT,DXRGHT,DYLEFT,
     2     DYRGHT,E,EBEST,ELEFT,EPRED,ERIGHT,ERROR,ETRY,FCTL(100),
     3     FXDKNT,H,Q,SGN,U(100),UERROR(100),VORDL(28,2),
     4     XI(28),XIL(28),XX(100)
      integer I,II,INDLP,INTERV,
     1     JADD,KNOT,LMAX,LPCNT,LX,LXI,LXI1,LXI2,MODE
      COMMON/INPUT/LX,XX,U,JADD,ADDXI,MODE,debug
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
      COMMON/ OTHER / LXI,LXI1,LXI2,Q ,CHANGE,ERROR ,ACC,XI
C
      I = II
C       **NUMERICAL ANALYSIS PARAMETERS SET HERE
      INDLP=9
      BD = ACC*ERROR/FLOAT(LXI)
      DIST = .0625
      H = XI(I+2)-XI(I)
      ALOW = XI(I) + DIST*H
      AHIGH = XI(I+2) - DIST*H
```

```
      LPCNT= 0
      MODE = 3
C
C       **BEGIN SEARCH - FIND THREE VALUES FOR THE ITH KNOT
C         SUCH THAT L2-ERROR AT MIDDLE VALUE, A , IS LESS THAN
C         ERROR AT LEFT VALUE, ALEFT, AND AT RIGHT VALUE, ARIGHT
      A  = XI(I+1)
      E = FXDKNT(A)
      ALEFT = A + Q*(XI(I)-A)
      ELEFT = FXDKNT(ALEFT)
      if (.not. debug)                 go to 5
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      ARIGHT = 0.
      ERIGHT = 0.
      WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
    5 SGN = SIGN(1.,ELEFT-E)
      IF (SGN.GE.0.)                   GO TO 20
                                       GO TO 60
C
C       **SEARCHING FOR NEW KNOT TO THE RIGHT
   10 ALEFT = A
      ELEFT = E
      A = ARIGHT
      E = ERIGHT
   20 ARIGHT = A + Q*(XI(I+2)-A)
C
C       **BUFFER TO PREVENT COALESCING OF KNOTS
   30 IF (AHIGH.GE.ARIGHT)             GO TO 40
      AA = AHIGH
      if (.not. debug)                 go to 199
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,610) I
                                       GO TO 199
C
   40 ERIGHT = FXDKNT(ARIGHT)
      if (.not. debug)                 go to 41
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
```

```
      WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
   41 IF (E.LE.ERIGHT)                 GO TO 100
C
C       **CHECK TO STOP OPT
      IF(E -ERIGHT.LE.BD .OR. LPCNT .GT. INDLP )  GO TO 240
   50 LPCNT = LPCNT+1
      IF(SGN.GT.0.) GO TO 10
C
C       **SEARCHING FOR NEW KNOT TO THE LEFT
   60 ARIGHT = A
      ERIGHT = E
      A = ALEFT
      E = ELEFT
   70 ALEFT = A + Q*(XI(I)-A)
C
C
C       **BUFFER TO PREVENT COALESCING OF KNOTS
   80 IF (ALEFT.GE.ALOW)               GO TO 90
      AA = ALOW
      if (.not. debug)                 go to 199
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,620) I
                                       GO TO 199
C
   90 ELEFT = FXDKNT(ALEFT)
      if (.not. debug)                 go to 91
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
   91 IF (E.LE.ELEFT)                  GO TO 100
C
C       **CHECK TO STOP OPT
      IF(E - ELEFT.LE.BD .OR. LPCNT .GT. INDLP )  GO TO 230
                                       GO TO 50
C
C       **REQUIRED 3 VALUES HAVE BEEN FOUND
C         FOLLOWING CODE FINDS PT. AT WHICH MIN OF PARABOLA CURVE PASSIN
C         THRU THE ERROR VALUES AT THE PTS ALEFT, A, ARIGHT OCCURS
```

```
  100 DXLEFT = ALEFT - A
      DXRGHT = ARIGHT - A
      DYLEFT = (ELEFT-E)/DXLEFT
      DYRGHT = (ERIGHT-E)/DXRGHT
      DIFF = DYLEFT - DYRGHT
      IF (DIFF .EQ. 0.)                GO TO 200
      DEL = .5/DIFF*(DXRGHT*DYLEFT-DXLEFT*DYRGHT)
      EPRED = E+DEL*(DYRGHT+(DXRGHT-DEL)/(ARIGHT-ALEFT)*DIFF)
      ABEST = A + DEL
      EBEST = FXDKNT(ABEST)
      if (.not. debug)                 go to 109
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE (6,900) ELEFT,EBEST,ERIGHT,ALEFT,ABEST,ARIGHT
C
C       **DETERMINE WHETHER ABEST GIVES BEST APPRX AND MAKE APPROPRIATE
C         SWITCHING OF THE AI'S DEPENDING ON SIGN OF DEL
  109 IF (EBEST.LE.E)                  GO TO 130
      IF(DEL)110,200,120
  110 ALEFT = ABEST
      ELEFT = EBEST
      GO TO 170
  120 ARIGHT = ABEST
      ERIGHT = EBEST
      GO TO 170
  130 IF(DEL)140,200,150
  140 ARIGHT = A
      ERIGHT = E
      GO TO 160
  150 ALEFT = A
      ELEFT = E
  160 A = ABEST
      E = EBEST
C
C       **FOLLOWING       TESTS DETERMINE WHETHER OR NOT TO
C         REITERATE PARABOLA MINIMIZATION PHASE
  170 IF (ABS(EPRED-EBEST).LT.5.*BD)   GO TO 210
      IF(LPCNT.GT.INDLP)               GO TO 200
```

```
      LPCNT = LPCNT+1
                                      GO TO 100
C
  199 ETRY = FXDKNT(AA)
      IF (E.LT.ETRY)                  GO TO 200
      A = AA
      E = ETRY
  200 CHANGE = CHANGE + ABS(A -XI(I+1))/H
      XI(I+1) = A
      ERROR = E
      if (.not. debug)                RETURN
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
                                      RETURN
C
C           IN FINAL VERSION GO TO 210, IS REPLACED BY GO TO 200
  210 CONTINUE
      if (.not. debug)                go to 200
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,640) LPCNT
      GO TO 200
  230 A = ALEFT
      E = ELEFT
      if (.not. debug)                go to 200
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,640) LPCNT
      GO TO 200
  240 A = ARIGHT
      E = ERIGHT
      if (.not. debug)                go to 200
C     ***  THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT  ***
      WRITE(6,640) LPCNT
      GO TO 200
  610 FORMAT(' *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (',I2,
     1') MOVED TOO CLOSE TO RIGHT NEIGHBOR')
  620 FORMAT(' *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (',I2,
     1') MOVED TOO CLOSE TO LEFT NEIGHBOR')
```

```
640 FORMAT(' *** OPT DISCONTINUED AT',I4,
   1' - INSUFFICIENT CHANGE IN ERROR')
900 FORMAT(' PARABOLA - ERROR VALUES ',3E20.6/
   112X,'AI VALUES    ',3E20.6)
    END
```

```
C
C*******************************************************************
C*******************************************************************
C*******************************************************************
C
      FUNCTION FXDKNT (CHANGE)
C               THE FUNCTION RETURNS THE SQUARE OF THE L2-ERROR
      IMPLICIT NONE
      LOGICAL MODE3
C**  IT MAY BE NECESSARY ON SOME SYSTEMS TO MENTION ALL COMMON BLOCKS
C    LISTED HERE IN THE PROGRAM CALLING *FXDKNT*, TOO, TO INSURE THAT
C    THE INFO IN THESE BLOCKS DOES NOT DIE BETWEEN CALLS TO *FXDKNT*.
      real ADDXI(26),BC(30),CHANGE,COEFL(27,4),CUBERR(100),DIF,
     1     DOT,ERBUT1,ERRL1,ERRL2,ERRL99,FCT(100,30),FCTL(100),
     2     FXDKNT,SCALE,SERROR(100),T,TREND(100),
     3     U(100),UERROR(100),VORD(30,28,2),VORDL(28,2),W,
     4     WEIGHT(100),XIL(28),XKNOT,XSCALE,XX(100)
      integer I,IDUM,IE,IO,ILAST,ILM3,ILOC,INSERT,INSIRT(30),INTERV,
     1     IORDER(28),IPRINT,J,JADD,K,KNOT,KNOTSV,L,LMAX,LX,MODE
      DOUBLE PRECISION TRPZWT(100),SUM
      COMMON / WANDT / TREND,TRPZWT
      COMMON/ INPUT /LX,XX,U,JADD,ADDXI,MODE
C           U(L) = FCT TO BE APPR AT XX(L), L=1,LX.
C                XX(L) IS ASSUMED TO BE NONDECREASING WITH L
C           ADDXI(I) = I-TH KNOT TO BE INTRODUCED, I=1,JADD
C           MODE = 0,1,2,3 . SEE COMMENTS BELOW ( AND IN NUBAS)
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
C           UERROR(L) = ERROR OF B.L2 A. TO U, L=1,LX
C           KNOT = CURRENT NO. OF KNOTS (INCL BDRY KNOTS)
C           INTERV = KNOT - 1  = CURRENT NO. OF INTERVALS (POL.PIECES)
C           XIL(K),K=1,KNOT, CURRENT (ORDERED) SET OF KNOTS
C           THE MAXIMUM ERROR OCCURS AT XX(LMAX)
C        IF ARG=1, FCTL(L) CONTAINS THE CURRENT B.A TO U AT XX(L)
C           COEFL(I,.) CONTAINS THE POL.COEF. ON I-TH INTERVAL FOR B.A.
C           VORDL(I,.) CONTAINS VALUE AND DERIV. OF B.A. AT XIL(I)
      COMMON/ BASIS /FCT,VORD,BC,ILAST
C**  A CHANGE IN THE COLUMN LENGTH OF *FCT* FORCES CHANGE IN ST.NO.69
```

```
C    IN *NUBAS* .
C           FCT (L,M) = BASIS FCT M AT XX(L)
C         VORD(M,K,L) CONTAINS THE ORDS (L=1) AND SLOPES (L=2) OF FCT M
C           AT THE KNOT INTRODUCED AS K-TH. CORRELATION TO ORDERING OF
C           KNOTS BY SIZE IS DONE VIA IORDER, I.E., ORD AND SLOPE AT
C           XIL(K) ARE IN VORD(M,IORDER(K),.).
C           BC(I) = COORDINATE OF U (AND OF B.A. TO U) WRTO I-TH O.N.FCT
C           ILAST = CURRENT NO. OF BASIS FCTNS
      COMMON/ LASTB /IORDER,INSIRT,XKNOT
C           THE FCT ILAST (TO BE) INTRODUCED LAST HAS ADDITIONAL KNOT
C           XKNOT, THE KNOT JUST INTRO-
C           DUCED HAS INDEX INSERT IN XIL,INSERT IS SAVED IN INSIRT(ILAS
C           FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).
C     ***LOCAL VARIABLES
      COMMON /LOCAL/ XSCALE,KNOTSV,ERBUT1,CUBERR,WEIGHT,MODE3
C           XSCALE = XX(LX) - XX(1), USED TO NORMALIZE INNER PRODUCT
C                  = LENGTH OF THE INTERVAL OF INTEGRATION
C           KNOTSV = NO. OF KNOTS USED IN MOST RECENT CALL TO FXDKNT
C           ERBUT1 = SQ. OF L2-ERROR OF APPR USING ALL BUT THE ONE
C                    KNOT BEING VARIED ( USED IN MODE = 3)
C           CUBERR = UERROR OF B.A. BY CUBIC POL-S (NEEDED FOR MODE = 2)
C           MODE3 = TRUE OR FALSE DEP. ON WHETHER PREV. CALL WAS IN
C                   MODE=3 OR NOT
C           CHANGE  = THE NEW VALUE OF THE KNOT BEING VARIED IF MODE=3;
C                  IT IS USED TO CONTROL PRINTED OUTPUT OTHERWISE.
      IPRINT = IFIX(CHANGE)
      IF (MODE.GT.0)                  GO TO 29
C-----------
C        *** MODE=0* COMPUTE BASIS FCTNS 1 THROUGH 4 AND  B.A. TO U WRTO
C         THESE, THEN SET MODE = 1 AND PUT UERROR INTO U.
      XSCALE = XX(LX) - XX(1)
      DO 10 I=5,30
         INSIRT(I) = 0
   10 continue
      DO 11 L=1,LX
         UERROR(L) = U(L)
         TREND(L) = T(XX(L))
```

```
         WEIGHT(L) = W(XX(L))
   11 continue
      DO 12 L=3,LX
         TRPZWT(L-1) = (XX(L)-XX(L-2))/2.*WEIGHT(L-1)
   12 continue
      TRPZWT(1) = (XX(2)-XX(1))/2.*WEIGHT(1)
      TRPZWT(LX) = (XX(LX)-XX(LX-1))/2.*WEIGHT(LX)
C
      XIL(1) = ADDXI(1)
      XIL(2) = ADDXI(2)
      IORDER(1) = 1
      IORDER(2) = 2
      KNOT = 2
      INTERV = 1
      DO 19 I=1,4
         ILAST = I
         CALL NUBAS
         DO 19 L=1,LX
            UERROR(L) = UERROR(L) - BC(I)*FCT(L,I)
   19 continue
C
      MODE = 1
      DO 20 L = 1,LX
         CUBERR(L) = UERROR(L)
   20 continue
C        IF (JADD.LE.2), ONLY B.A. BY CUBICS IS COMPUTED
C         OTHERWISE, ADDXI(I), I.GT.2, CONTAINS ADDITIONAL KNOTS
      JADD = JADD - 2
      IF (JADD.LE.0)                    GO TO 60
      DO 21 I=1,JADD
         ADDXI(I) = ADDXI(I+2)
   21 continue
                                        GO TO 51
C-----------
   29                                   GO TO (40,40,30),MODE
C-----------
C        *** MODE=3 *** MERELY CHANGE THE LAST KNOT INTRODUCED TO
```

```
C                       CHANGE (THE INPUT ARGUMENT TO  FIXDKNT ) AND
C                       RECOMPUTE L2 ERROR.
C                       THIS MODE SHOULD BE USED FOR MINIMIZING THE
C                       L2-ERROR WRTO THE KNOT
C                       INTRODUCED LAST AS IT MINIMIZES THE COMP WORK
C                  IF MODE3 = TRUE (I.E., THE PRECEDING CALL TO FXDKNT
C                       WAS IN MODE=3),THE PROGR WILL ASSUME THAT CHANGE
C                       HAS THE SAME ORDER REL TO THE OTHER KNOTS AS THE
C                       PREV.INTRODUCED VALUE FOR KNOT. OTHERWISE
C                  IF MODE3 = FALSE, I.E., THE PRECEDING CALL WAS IN
C                       SOME OTHER MODE), A FCT IS ADDED WITH CHANGE AS
C                       THE ADDITIONAL KNOT.
C                       UERROR IS ASSUMED TO CONTAIN ERROR OF B.A. TO U
C                       ALL PREV FCTNS, AND ERBUT1 the SQ. OF ITS L2NORM
C                               **NOTE** IF THE NEXT CALL TO FXDKNT
C                       IS IN A MODE OTHER THAN 3, THE CHANGE PROPOSED
C                       NOW WILL BE MADE PERMANENT.
   30 XKNOT = CHANGE
      IF (MODE3)                        GO TO 35
      MODE3 = .TRUE.
      MODE = 2
      CALL NUBAS
      KNOTSV = KNOT
      MODE = 3
                                        GO TO 36
   35 CALL NUBAS
   36 FXDKNT = ABS(ERBUT1 - BC(ILAST)/XSCALE*BC(ILAST))
                                        RETURN
C-----------
C      ***MODE=1,2***  RETAIN THE FIRST KNOT KNOTS INTRODUCED EARLIER
C                       (HENCE THEIR CORRESP FCTNS) BUT REPLACE FURTHER
C                       FCTNS (IF ANY) BY FCTNS HAVING ADDITIONAL
C                       KNOTS ADDXI(I),I=1,JADD) HENCE
C                       IF KNOT.LT.KNOTSV(=NO.OF KNOTS USED IN PREV CALL
C                       THEN 40 THROUGH 49 RESTORES ARRAYS IORDER,XIL,
C                       UERROR TO THE STATE OF  ILAST = KNOT + 2 ,
C                       INVERTING THE ACTION OF DO 11 ... TO 14 IN NUBAS
```

```
   40 IF (KNOT.LT.KNOTSV)              GO TO 42
      KNOT = KNOTSV
      IF (.NOT.MODE3)                  GO TO 50
      DO 41 L=1,LX
         UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)
   41 continue
                                       GO TO 49
   42 DO 43 L=1,LX
         UERROR(L) = CUBERR(L)
   43 continue
      IF (KNOT.LE.2)                   GO TO 48
      IDUM = KNOT + 1
      DO 45 IO=IDUM,KNOTSV
         INSERT = INSIRT(ILAST)
         ILM3 = ILAST - 3
         DO 44 K=INSERT,ILM3
            IORDER(K) = IORDER(K+1)
            XIL(K) = XIL(K+1)
   44    continue
         ILAST = ILAST-1
   45 continue
      DO 47 I=5,ILAST
         DO 47 L=1,LX
            UERROR(L) = UERROR(L) - BC(I)*FCT(L,I)
   47 continue
                                       GO TO 49
   48 XIL(2) = XIL(ILAST-2)
      IORDER(2) = 2
      KNOT = 2
   49 IF (JADD.GT.0)                   GO TO 51
      ILAST = KNOT + 2
      INTERV = KNOT - 1
                                       GO TO 60
C
C      ***MODE=1,2***   ADD JADD BASIS FCTNS, I.E., FOR IO=1,JADD,
C                       CONSTRUCT FCT ILAST WITH ONE MORE KNOT, VIZ.
C                       XKNOT=ADDXI(IO), THAN THE PREVIOUS LAST FCT,
```

```
C                        ORTHONORMALIZE IT OVER ALL PREVIOUS FCTNS, THEN
C                        COMPUTE THE COORDINATE BC(ILAST) OF U WRTO IT,
C                        SUBTRACT OUT ITS COMPONENT FROM UERROR.
   50 IF (JADD.LE.0)                   GO TO 61
   51 DO 52 IO=1,JADD
         XKNOT = ADDXI(IO)
         CALL NUBAS
         DO 52 L=1,LX
            UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)
   52 continue
C
   60 FXDKNT= DOT(31,2)/XSCALE
      ERBUT1 = FXDKNT
      KNOTSV = KNOT
   61 MODE3 = .FALSE.
      IF (IPRINT.EQ.0)                 RETURN
C          VARIOUS PRINTING IS DONE DEP ON THE IPRINT = IFIX(CHANGE)
                                       GO TO (70,80,90),IPRINT
C
C         COMPUTE COEFFICIENTS OF B.A. AND PRINT
C     ****               BEST APPROXIMATION PRINTOUT            ****
C       FORMAT IS
C             KNOTS XI(J)         CUBIC COEFFICIENTS P(I,J) IN
C                                 INTERVAL (XI(J), XI(J+1))
C                         ERROR CURVE (SCALED)
C
C       THE FOLLOWING FORTRAN CODE FINDS VALUES AT X OF THE
C       APPROXIMATION FROM THIS OUTPUT----
C                  I=LXI
C                1 A=X-XI(I)
C                  IF(A .ge. 0.) goto 4
C                  I=I-1
C                  IF(I .gt. 0) goto 1
C                  I=1
C                4 V=P(1,I)+A*(P(2,I)+A*(P(3,I)+A*P(4,I)))
C
   70 WRITE(6,610)
```

```
      DO 72  I=1,KNOT
         ILOC = IORDER(I)
         DO 72  L=1,2
            SUM = 0.D0
            DO 71 J=1,ILAST
               SUM = SUM + BC(J)*VORD(J,ILOC,L)
   71       continue
            VORDL(I,L) = SUM
   72 continue
      CALL EVAL
      DO 73 I=1,INTERV
         WRITE(6,620) I,XIL(I)
         WRITE (6,630) (J,COEFL(I  ,J),J=1,4)
   73 continue
      WRITE (6,620) KNOT,XIL(KNOT)
  610 FORMAT(12X,'KNOTS',22X,'CUBIC COEFFICIENTS'//)
  620 FORMAT(5X, 'XI(',I2,') =', F12.6)
  630 FORMAT(37X,'C(',I1,') =',E16.6)
C
C        **COMPUTE L2, L1, MAX ERRORS AND PRINT
   80 ERRL2 = SQRT(FXDKNT)
      ERRL1 = 0.
      ERRL99= 0.
      DO 82 L=1,LX
         DIF = ABS(UERROR(L)*WEIGHT(L))
         IF(ERRL99.GT.DIF)            GO TO 81
         LMAX = L
         ERRL99 = DIF
   81    ERRL1 = ERRL1+ DIF
   82 CONTINUE
      ERRL1 = ERRL1/FLOAT(LX)
      WRITE(6,623) ERRL2, ERRL1, ERRL99,XX(LMAX)
C     *** THE FOLLOWING CARD IS TEMPORARY
      GO TO (90,96,96),IPRINT
C
C       **  SCALE ERROR CURVE AND PRINT
   90 IE = 0
```

```
      SCALE = 1.
      IF (ERRL99.GE.10.)                GO TO 92
      DO 91 IE=1,9
         SCALE = SCALE*10.
         IF (ERRL99*SCALE.GE.10.)      GO TO 92
   91 CONTINUE
   92 DO 93 L=1,LX
         SERROR(L) = UERROR(L)*SCALE
   93 continue
                                        GO TO (94,95,95),IPRINT
   94 WRITE (6,621) IE,(L,XX(L),FCTL(L),SERROR(L),L=1,LX)
                                        GO TO 96
   95 WRITE (6,622) IE,(L,XX(L),SERROR(L),L=1,LX)
   96                                   RETURN
  621 FORMAT(' '//15X,'APPROXIMATION AND SCALED ERROR CURVE'/8X,
     *'DATA POINT',7X,'APPROXIMATION',3X,'DEVIATION X 10E+',I1/
     *(1X,I4,F16.8,F16.8,F17.6))
  622 FORMAT(' '//'  ERROR CURVE'/ 8X, 'DATA POINT', 23X,
     1'DEVIATION X 10E+',I1/( 1X,I4,F16.8,16X,F17.6))
  623 FORMAT(' '///10X,'LEAST SQUARE ERROR =',E20.6/
     1             10X,'AVERAGE ERROR      =',E20.6/
     2             10X,'MAXIMUM ERROR      =',E20.6,' AT',F12.6///)
      END
C
C*************************************************************************
C
      SUBROUTINE INTERP
C
C            COMPUTE THE SLOPES VORDL(I,2), I=2,KNOT-1 AT INTERIOR
C            KNOTS OF CUBIC SPLINE FOR GIVEN VALUES VORDL(I,1),I=1,KNOT,
C            AT ALL  THE KNOTS AND GIVEN BOUNDARY DERIVATIVES
      IMPLICIT NONE
      real COEFL(27,4),D(28),DIAG(28),FCTL(100),G,UERROR(100),
     1     VORDL(28,2),XIL(28)
      integer INTERV,KNOT,LMAX,M,NJ
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
      DATA DIAG(1),D(1)/1.,0./
```

```
      DO 10 M=2,KNOT
         D(M) = XIL(M) - XIL(M-1)
         DIAG(M) = (VORDL(M,1)-VORDL(M-1,1))/D(M)
   10 continue
      DO 20 M=2,INTERV
         VORDL(M,2) = 3.*(D(M)*DIAG(M+1) + D(M+1)*DIAG(M))
         DIAG(M) = 2.*(D(M)+D(M+1))
   20 continue
      DO 30 M=2,INTERV
         G = -D(M+1)/DIAG(M-1)
         DIAG(M) = DIAG(M) + G*D(M-1)
         VORDL(M,2) = VORDL(M,2) + G*VORDL(M-1,2)
   30 continue
      NJ = KNOT
      DO 40 M=2,INTERV
         NJ = NJ - 1
         VORDL(NJ,2) = (VORDL(NJ,2) - D(NJ)*VORDL(NJ+1,2))/DIAG(NJ)
   40 continue
                                     RETURN
      END
C
C************************************************************************
C
      FUNCTION DOT (M,INDEX)
C         COMPUTE INNER PRODUCT OF FCT M WITH FCT ILAST (INDEX=1) OR
C     UERROR (INDEX=2)
      IMPLICIT NONE
      real ADDXI(26),BC(30),COEFL(27,4),DOT,
     1     FCT(100,30),FCTL(100),
     2     TREND(100),
     3     U(100),UERROR(100),VORD(30,28,2),VORDL(28,2),
     4     XIL(28),XX(100)
      integer ILAST,INDEX,INTERV,
     1     JADD,KNOT,L,LMAX,LX,M,MODE
      DOUBLE PRECISION TRPZWT(100),G(100), SUM
      COMMON / WANDT / TREND,TRPZWT
      COMMON/ INPUT /LX,XX,U,JADD,ADDXI,MODE
```

```
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
      COMMON/ BASIS /FCT,VORD,BC,ILAST
                                        GO TO (10,30),INDEX
   10 IF (M.EQ.ILAST)                   GO TO 20
      DO 11 L=1,LX
         G(L) = FCT(L,M)*FCTL(L)
   11 continue
                                        GO TO 80
   20 DO 21 L=1,LX
         G(L) = FCTL(L)*FCTL(L)
   21 continue
                                        GO TO 80
   30 IF (M.EQ.31)                      GO TO 40
      DO 31 L=1,LX
         G(L) = FCTL(L)*UERROR(L)
   31 continue
                                        GO TO 80
   40 DO 41 L=1,LX
         G(L) = UERROR(L)*UERROR(L)
   41 continue
C
C  EFFICIENTLY PROGRAMMED DOUBE PRECISION ACCUMULATION OF SCALAR
C  PRODUCTS IS CALLED FOR HERE.  AT PURDUE, WE USE
C         D  =  ARITH1(C,N,A,IA,B,IB)
C  WHICH RETURNS THE VALUE OF
C         D  =  C - SUM(A(1+J*IA) * B(1+J*IB), J=0,...,N-1)
C
   80 SUM = 0.D0
      DO 81, L=1,LX
         SUM = SUM + G(L)*TRPZWT(L)
   81 continue
      DOT = SUM
                                        RETURN
      END
C
C*************************************************************************
C
```

```
      SUBROUTINE EVAL
C        COMPUTE POL. COEFF COEFL(I,K) OF FCT ILAST FROM VORDL,
C        THEN COMPUTE FCTL(L) = (FCT ILAST)*TREND AT XX(L),L=1,LX
C
      IMPLICIT NONE
      real ADDXI(26),COEFL(27,4),DUM1,DUM2,DX,
     1     FCTL(100),
     2     TREND(100),TRPZWT(100),
     3     U(100),UERROR(100),VORDL(28,2),
     4     XIL(28),XX(100)
      integer I,INTERV,ISWTCH,
     1     J,JADD,KNOT,L,LMAX,LX,MODE
      COMMON / WANDT / TREND,TRPZWT
      COMMON/ INPUT /LX,XX,U,JADD,ADDXI,MODE
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
      DO 10 I=1,INTERV
         COEFL(I,1) = VORDL(I,1)
         COEFL(I,2) = VORDL(I,2)
         DX = XIL(I+1) - XIL(I)
         DUM1 = (VORDL(I+1,1)-VORDL(I,1))/DX
         DUM2 =  VORDL(I,2)+VORDL(I+1,2)-2.*DUM1
         COEFL(I,3) = (DUM1-DUM2-VORDL(I,2))/DX
         COEFL(I,4) = DUM2/DX/DX
   10 continue
C
      J = 1
      ISWTCH = 1
      DO 20 L=1,LX
                                    GO TO (11,13),ISWTCH
   11    IF (J.EQ.INTERV)           GO TO 12
         IF (XX(L).LT.XIL(J+1))     GO TO 13
         J = J + 1
                                    GO TO 11
   12    ISWTCH = 2
   13    DX = XX(L) - XIL(J)
         FCTL(L) = (COEFL(J,1)+DX*(COEFL(J,2)+DX*(COEFL(J,3)
     *                                +DX*COEFL(J,4))))*TREND(L)
```

```
   20 continue
                                             RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE NUBAS
      IMPLICIT NONE
      real ADDXI(26),BC(30),C,COEF(381,4),COEFL(27,4),
     1     DOT,DX,FCT(100,30),FCTL(100),
     2     ONEOVC,TEMP(30),
     3     U(100),UERROR(100),VORD(30,28,2),VORDL(28,2),
     4     XI(381),XIL(28),XKNOT,XX(100)
      integer I,IBOUND,ICLAST,ID,ILAST,ILM1,ILOC,INSERT,INSIRT(30),
     1     INTERV,IO,IORDER(28),JADD,K,KNOT,L,LMAX,LX,MODE
      COMMON/ INPUT /LX,XX,U,JADD,ADDXI,MODE
      COMMON/ OUTPUT /UERROR,FCTL,XIL,COEFL,VORDL,KNOT,LMAX,INTERV
      COMMON/ BASIS /FCT,VORD,BC,ILAST
      COMMON/ LASTB /IORDER,INSIRT,XKNOT
C          COEF(IC,.) CONTAINS THE POL COEFFICIENTS OF FCT M FOR INTER-
C          VAL TO THE RIGHT OF XI(IC), IC=ICM,ICM+M-3,
C          WITH ICM = M*(M-7)/2 + 10 (WITH OBVIOUS MODS FOR M.LE.4)
C          THE FCT ILAST (TO BE) INTRODUCED LAST, HAS ITS VALUES AT THE
C          THE POINTS XX(L) IN FCTL(L),           HAS FIRST INDEX ICLAS
C          IN COEF AND XI, HAS ADDITIONAL KNOT XKNOT, THE KNOT KNOTS
C          FOR IT ARE CONTAINED, IN INCREASING ORDER, IN XIL,ITS COR-
C          RESPONDING ORDS AND SLOPES ARE IN VORDL, THE KNOT JUST INTRO
C          DUCED HAS INDEX INSERT IN XIL,INSERT IS SAVED IN INSIRT(ILAS
C          FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).
      logical REPEAT
      REPEAT = .FALSE.
      IF (MODE.GT.0)                   GO TO 8
C--------***CONSTRUCT FCT ILAST FOR ILAST.LE.4
      XI(ILAST) = XIL(1)
      ICLAST = ILAST
      ILM1 = ILAST-1
      IF (ILAST.GT.2)                  GO TO 7
```

```
      IF (ILAST.EQ.2)                  GO TO 6
C        FIRST BASIS FCT IS A CONSTANT
      VORDL(1,1) = 1.
      VORDL(2,1) = 1.
      VORDL(1,2) = 0.
      VORDL(2,2) = 0.
                                       GO TO 67
C        SECOND BASIS FCT IS A STRAIGHT LINE
    6 VORDL(2,2) = VORDL(1,1)/(XIL(2) - XIL(1))*2.
      VORDL(1,2) =-VORDL(2,2)
C
    7 VORDL(2,1) = - VORDL(2,1)
      VORDL(2,2) = - VORDL(2,2)
                                       GO TO 59
C--------
    8                                  GO TO (10,10,140),MODE
C--------***SET UP CONSTANTS DEP.ON ILAST. INSERT NEW KNOT  INTO XIL
C           AND UPDATE VORD FOR FCT M,M=1,ILAST-1
   10 KNOT = KNOT + 1
      ILAST = KNOT + 2
      ICLAST = ILAST*(ILAST-7)/2 + 10
      ILM1 = ILAST-1
      INTERV = KNOT - 1
      DO 11 INSERT=2,INTERV
      IF (XKNOT.LT.XIL(INSERT))        GO TO 12
   11 CONTINUE
                                       GO TO 95
   12 IF (XKNOT.LE.XIL(INSERT-1))      GO TO 95
      IO = KNOT
      DO 13 L=INSERT,INTERV
      IO = IO - 1
      XIL(IO+1) = XIL(IO)
   13 IORDER(IO+1) = IORDER(IO)
      IORDER(INSERT) = KNOT
                                       go to 14
  140 insert = insirt(ilast)
   14 XIL(INSERT) = XKNOT
```

```
      DX = XKNOT - XIL(1)
      DO 15 I=1,4
      VORD(I,KNOT,1)=COEF(I,1)+DX*(COEF(I,2)+DX*(COEF(I,3)
     *                                    +DX*COEF(I,4)))
   15 VORD(I,KNOT,2)=COEF(I,2)+DX*(2.*COEF(I,3)+DX*3.*COEF(I,4))
      IF(ILM1.LT.5) GO TO 20
      ID = 4
      IBOUND = 4
      DO 19 I=5,ILM1
      ID = ID + I - 4
      IBOUND = IBOUND + I - 3
   17 IF (ID.EQ.IBOUND)                GO TO 18
      IF (XKNOT.LT.XI(ID+1))           GO TO 18
      ID = ID + 1
                                       GO TO 17
   18 DX = XKNOT - XI(ID)
      VORD(I,KNOT,1)=COEF(ID,1)+DX*(COEF(ID,2)+DX*(COEF(ID,3)
     *                                        +DX*COEF(ID,4)))
   19 VORD(I,KNOT,2)=COEF(ID,2)+DX*(COEF(ID,3)*2.+DX*3.*COEF(ID,4))
C--------
C--------DEFINE LAST BASIS FUNCTION
   20 CONTINUE
                                       GO TO (30,40,50),MODE
C        *** MODE=1 *** ADD ILAST-TH BASIS FUNCTION. CONSTRUCT FROM FCT
C                       ILAST-1 BY REFLECTING THE PART OF THE LATTER TO
C                       THE RIGHT OF XKNOT ACROSS THE X-AXIS, THEN INTER
C                       POLATING. THIS SHOULD INDUCE ONE MORE OSCILLATIO
C                       N IN FCT ILAST THAN IN FCT ILAST-1
C
   29 MODE = 1
   30 VORDL(1,2) = 0.
      DO 31 K=1,KNOT
   31    VORDL(K,1) = MAX(0.,XIL(K)-XKNOT)**3
      VORDL(KNOT,2) = 3.*(XIL(KNOT)-XKNOT)**2
                                       GO TO 55
C
C        *** MODE=2 *** REPLACE FCT ILAST BY INTERPOLATING IT AT THE
```

```
C                       CURRENT SET OF KNOTS. IF FCT ILAST HAS NOT BEEN
C                       PREVIOUSLY DEF (INSIRT(ILAST)=0)(SEE 9 ABOVE,
C                       ALSO MAIN AT 10)) SET MODE=1,PROCEED IN THAT MOD
C
   40 IF (INSIRT(ILAST).EQ.0)          GO TO 29
      VORDL(1,1)=VORD(ILAST,1,1)
      VORDL(1,2)=VORD(ILAST,1,2)
      ID = ICLAST
      IBOUND = ICLAST + ILAST - 4
      DO 43 K=2,INTERV
   41 IF (ID.EQ.IBOUND)                GO TO 42
      IF (XIL(K).LT.XI(ID+1))          GO TO 42
      ID = ID +1
                                       GO TO 41
   42 DX = XIL(K)- XI(ID)
   43 VORDL(K,1) = COEF(ID,1)+DX*(COEF(ID,2)+DX*(COEF(ID,3)
     *                                     +DX*COEF(ID,4)))
      VORDL(KNOT,1)=VORD(ILAST,2,1)
      VORDL(KNOT,2)=VORD(ILAST,2,2)
                                       GO TO 55
C
C        *** MODE=3 *** CHANGE FCT ILAST BY CHANGING JUST THE KNOT INTRO
C                       DUCED LAST
C
   50 ID = ICLAST + INSERT - 1
      DX = XKNOT - XI(ID)
      XI(ID) = XKNOT
      IF (DX.GE.0.)                    GO TO 51
      ID = ID - 1
      DX = XKNOT - XI(ID)
   51 VORDL(INSERT,1) = COEF(ID,1) +DX*(COEF(ID,2)+DX*(COEF(ID,3)
     *                                           +DX*COEF(ID,4)))
C
C        *** INTERPOLATE
   55 CALL INTERP
                                       GO TO (57,57,59),MODE
   57 ID = ICLAST - 1
```

```
      DO 56 IO=1,INTERV
      ID = ID + 1
   56 XI(ID) = XIL(IO)
      INSIRT(ILAST) = INSERT
C--------
C--------*** ORTHONORMALIZE FCT ILAST OVER PREVIOUS (ORTHONORMAL) SET
C            THEN COMPUTE THE COMPONENT BC(ILAST) OF UERROR WRTO IT
C            FINALLY,STORE THE VARIOUS REPRESENTATIONS OF FCT ILAST
C
   59 CALL EVAL
      TEMP(ILAST) = SQRT(DOT(ILAST,1))
      IF (REPEAT .AND. ABS(1.-TEMP(ILAST)) .GT. .5) GO TO 65
      DO 60 I=1,ILM1
      TEMP(I) = DOT(I,1)
      DO 69 L=1,LX
   69   FCTL(L) = FCTL(L) - TEMP(I)*FCT(L,I)
      DO 61 K=1,KNOT
      ILOC = IORDER(K)
      DO 61 L=1,2
   61   VORDL(K,L) = VORDL(K,L) - TEMP(I)*VORD(I,ILOC,L)
   60 CONTINUE
   67 CALL EVAL
      C = SQRT(DOT(ILAST,1))
      WRITE (6,667)ILAST,(TEMP(I),I=1,ILAST),C
  667 FORMAT(I3,7E11.3/(7X,7E11.3))
      IF(ILAST .GT. 1 .AND. REPEAT .AND. ABS(1.-C) .GT. .5) GO TO 65
      IF (C+TEMP(ILAST) .LE. TEMP(ILAST)) GO TO 65
      ONEOVC = 1./C
                                        GO TO 68
   65 ONEOVC = 0.
   68 BC(ILAST) = DOT(ILAST,2)*ONEOVC
      DO 62 K=1,KNOT
      ILOC = IORDER(K)
      DO 62 L=1,2
      VORDL(K,L) = VORDL(K,L)*ONEOVC
   62 VORD(ILAST,ILOC,L) = VORDL(K,L)
      IF (ONEOVC .EQ. 0. .OR. ILAST .EQ. 1
```

```
     *       .OR. C .GE. 1.E-2*TEMP(ILAST)) GO TO 152
      REPEAT = .TRUE.
      CALL INTERP
                                 GO TO 59
  152 CONTINUE
      ID = ICLAST - 1
      DO 63 IO=1,INTERV
      ID = ID + 1
      DO 63 L=1,4
   63 COEF(ID,L) = COEFL(IO,L)*ONEOVC
      DO 64 L=1,LX
   64 FCT(L,ILAST) = FCTL(L)*ONEOVC
C--------
                                 RETURN
C
C     ***  THIS OUTPUT INDICATES A FAILURE CONDITION ***
   95 WRITE (6,950) XKNOT,ILAST
  950 FORMAT (15H   *** NEW KNOT,E20.8,13H FOR FUNCTION,I3,50H OUT OF BO
     *UNDS OR COINCIDENT WITH A PREVIOUS KNOT./36H   *** EXECUTION CANNO
     *T BE CONTINUED)
                                 STOP
C
      END
C
C***********TREND AND WEIGHT FUNCTIONS*********************************
C
      FUNCTION T(Z)
      real T,Z
      T = 1.
      RETURN
      END
C
      FUNCTION W(Z)
      W = 1.
      RETURN
C
      END
```