Purdue University

Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

1968

Least Squares Cubic Spline Approximation I - Fixed Knots

Carl de Boor

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

Report Number:
68-020

de Boor, Carl and Rice, John R., "Least Squares Cubic Spline Approximation I - Fixed Knots" (1968). *Department of Computer Science Technical Reports.* Paper 141.
https://docs.lib.purdue.edu/cstech/141

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

Least Squares Cubic Spline Approximation I -
Fixed Knots

Carl de Boor and John R. Rice

April 1968

CSD TR 20

Least Squares Cubic Spline Approximation I - Fixed Knots

Carl de Boor* and John R. Rice *+

1. Introduction. Spline functions, and, more generally, piecewise polynomial functions are the most successful approximating functions in use today. They combine ease of handling in a computer with great flexibility, and are therefore particularly suited for the approximation of experimental data or design curve measurements.

For a rather complete list of the recent literature on splines, the reader is referred to the bibliography of [8].

This paper presents an algorithm for the computation of the least-squares approximation to a given function u by cubic splines with a given fixed set of knots. But since the successful use of splines for purposes of "smoothly" approximating a given set of data depends strongly on the proper placement of the knots, the algorithm is written so as to facilitate experimentation with various knot sets in as economical a fashion as possible. In [2], use is made of this in a program which attempts to compute the least-squares-approximation to a given function u by cubic splines with a fixed number of knots.

As a consequence, the algorithm is somewhat more complex than seems warranted for the mere calculation of the L_2-approximation to u by a linear family of functions.

*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.

+This author was also partially supported by NSF grant GP-4052.

2. Mathematical background.

(a) Definition of splines. Let $\pi: a = \xi_0 < \xi_1 < \ldots < \xi_{k+1} = b$ be a partition of the interval $[a,b]$. A (polynomial) spline function of degree n on π is, by definition, any function $s(x) \in C^{(n-2)}[a,b]$, which on each of the intervals (ξ_i, ξ_{i+1}), $i = 0,\ldots,k$, reduces to a polynomial of degree $\leq n$. The points ξ_i are called knots (or, joints). We denote by S_π^n the linear space of all such functions. Define

$$(2.1) \qquad (x-\xi)_+^n = \begin{cases} (x-\xi)^n, & x \geq \xi, \\ 0, & x < \xi. \end{cases}$$

Then it is easily shown that each $s \in S_\pi^n$ is uniquely represented by two sets of parameters, $\Xi = \{\xi_1,\ldots,\xi_k\}$ and $A = \{a_1,\ldots,a_{k+n+1}\}$, where

$$(2.2) \qquad s(x) = S(A,\Xi,x) = \sum_{i=1}^{k} a_i (x-\xi_i)_+^n + \sum_{j=0}^{n} a_{k+j+1} x^j .$$

Apparently, the boundary "knots" ξ_0, ξ_{k+1}, play no role in this representation. In fact, the right-hand side of (2.2) is well-defined on the entire line. Hence, we may and will consider each $s \in S_\pi^k$ to be defined by (2.2) on the entire line. Nevertheless, we retain the boundary "knots" for use in other representations.

(b) Representation of splines. The representation (2.2) is useful for mathematical analysis, but is very ill-conditioned and cumbersome to evaluate. In computations, the following representations are to be preferred.

For purposes of evaluation, the following seems best:

<u>Repr. I</u>. The set $\{\xi_o,\dots,\xi_k\}$ and the set of polynomial coefficients $\{c_{ij} \mid i = 0,\dots,k;\ j = 0,\dots,n\}$, where

$$S(A,\Xi,x) = \sum_{j=0}^{n} c_{ij}(x-\xi_i)^j, \text{ for } \xi_i \le x \le \xi_{i+1},\ i = 0,\dots,k. \tag{2.3}$$

It is clear that this representation is highly redundant, requiring $(n+1)(k+1)$ linear parameters. In particular, if n is odd, and

$$r = (n+1)/2,$$

then c_{ij}, $j = r,\dots,n$, may be computed from $c_{ij}, c_{i+1,j}, j=0,\dots, r-1$, by

$$c_{ij}(\Delta\xi_i)^j = \sum_{s=0}^{r-1} \gamma_{j-r,s}\Big[c_{i+1,s}(\Delta\xi_i)^s - \sum_{t=s}^{r-1} \binom{t}{s} c_{is}(\Delta\xi_i)^t\Big],$$

$$j = r,\dots,n;\ i = 0,\dots,k, \tag{2.4}$$

where

$$\Delta\xi_m = \xi_{m+1} - \xi_m, \text{ and } \qquad \gamma_{ij} = (-1)^{i+j} \sum_{t=0}^{r-1} \binom{t}{i}\binom{r-1+t-j}{t-j}.$$

This gives

<u>Repr. II</u>. The set $\{\xi_o,\dots,\xi_{k+1}\}$ and the set $\{c_{ij} \mid i = 0,\dots,k+1;\ j = 0,\dots,r-1\}$,

where

$$c_{ij} = \frac{1}{j!} \left.\frac{d^j S(A,\Xi,x)}{dx^j}\right|_{x = \xi_i}. \tag{2.5}$$

This representation is redundant, too, requiring $(k+2)(n+1)/2$ linear parameters.

In reducing Repr. I to Repr. II, we only used the continuity of $S(A,\Xi,x)$ and its derivatives up to the $(r-1)$st. But since $S(A,\Xi,x)$ is in $C^{(n-2)}[a,b]$, a small subset of the c_{ij} is sufficient.

<u>Repr. III</u>. The set $\{\xi_0,\ldots,\xi_{k+1}\}$ and the set $\{c_{ij} \mid (j=0 \text{ and } i=0,\ldots,k+1)$ or $(j=1,\ldots,r-1, \text{ and } i=0,k+1)\}$.

To pass from Repr. III (and thence to other representations) is the spline interpolation problem. Its solution consists in solving a system of $k*(r-1)$ equations in the unknowns $c_{ij}, i=1,\ldots,k;\ j=1,\ldots,r-1$, whose coefficient matrix is block tridiagonal of block size $r-1$. The pertinent equations are:

$$\sum_{s=0}^{r-1} \gamma_{js}\Big[c_{i-1,s}(-\Delta\xi_{i-1})^{s-r-j} + \sum_{t=s}^{r-1} \binom{t}{s} c_{it}\{(\Delta\xi_i)^{t-r-j} - (\Delta\xi_{i-1})^{t-r-j}\}$$

$$(2.6) \qquad - c_{i+1,s}(\Delta\xi_i)^{s-r-j}\Big] = 0,$$

$$i = 1,\ldots,k;\ j = 0,\ldots,r-2.$$

It is clear that this representation requires $n+k+1$ linear parameters, hence is not redundant. In particular, it makes sense to define <u>the spline of degree</u> n <u>interpolating</u> $f \in C^{(r-1)}[a,b]$ <u>on</u> k as the unique element $s \in S_\pi^n$ satisfying

$$(2.7) \qquad \begin{aligned} s(\xi_i) &= f(\xi_i), \quad i=0,\ldots,k+1, \\ s^{(j)}(\xi_i) &= f^{(j)}(\xi_i), \quad i=0,k+1;\ j=1,\ldots,r-1. \end{aligned}$$

The algorithm under discussion employs each of these representations and the following

Repr. IV. The set $\{S(A,\Xi,x_i) \mid i=1,\ldots,N\}$, where $X = \{x_i \mid i=1,\ldots,N\}$ is a given (increasing) set of points (cf. below).

It should be pointed out [5; 9] that the set $\{S(A,\Xi,x_i) \mid i=1,\ldots,N\}$ represents $S(A,\Xi,x)$ if and only if for some subset $\hat{X}$ of X with $\hat{x}_1 < \hat{x}_2 < \ldots < \hat{x}_{n+k+1}$ one has

$$\hat{x}_i < \xi_i < \hat{x}_{i+n+1}, \quad i=1,\ldots,k. \tag{2.8}$$

For completeness, we mention a further non-redundant representation valid for arbitrary n, which makes use of the so called B-splines and brings out the "local" character of splines:

Repr. V. The set $\{\xi_{-n},\ldots,\xi_{k+n+1}\}$ and the set $\{b_{-n},\ldots,b_k\}$, where

$$S(A,\Xi,x) = \sum_{i=-n}^{k} b_i\, B_i(x), \tag{2.9}$$

and

$$B_i(x) = (\xi_{i+n+1} - \xi_i)g_n(\xi_i,\ldots,\xi_{i+n+1};x), \quad i=-n,\ldots,k,$$

$$g_n(s;x) = (s-x)_+^n,$$

with

$$\xi_{-n} \le \cdots \le \xi_{-1} \le a, \; b \le \xi_{k+2} \le \cdots \le \xi_{k+n+1}.$$

Here, $f(\xi_i,\ldots,\xi_{i+n+1})$ denotes the (n+1)st divided difference of the function $f(s)$ on the points $\xi_i,\ldots,\xi_{i+n+1}$.

It is not difficult to see that

$$B_i(x) \geq 0 \text{ with equality iff } x \notin (\xi_i, \xi_{i+n+1}),$$

$$\sum_{i=-n}^{k} B_i(x) = 1, \text{ all } x \in [\xi_0, \xi_{k+1}].$$

This representation is particularly useful for the study and computational handling of splines with repeated knots as the limit of splines with pairwise distinct knots defined above.

(c) Least-squares approximation. Let M be a linear space with inner product $\langle f,g\rangle$ and associated norm

$$||f|| = (\langle f,f\rangle)^{\frac{1}{2}}.$$

Let S be a finite-dimensional subspace of M. Given $u \in M$, the error

$$E(w) = ||u-w||$$

of approximating u by w is uniquely minimized over all $w \in S$ by the orthogonal projection $P_S u$ of u, i.e., $u^* = P_S u$ is determined by

$$u^* \in S, \text{ and, for all } w \in S, \ \langle u^*,w\rangle = \langle u,w\rangle.$$

u^* is most advantageously computed with the aid of an orthonormal basis $\{\Psi_i\}_{i=1}^m$ of S, i.e., a generating set for S which satisfies

$$\langle \Psi_i,\Psi_j\rangle = \delta_{ij}, \quad i,j=1,\ldots,m.$$

For then,

$$P_S u = \sum_{i=1}^{m} \langle u,\Psi_i\rangle\Psi_i \ . \tag{2.10}$$

Given a basis $\{\phi_i\}_1^m$ for S, an orthonormal basis $\{\Psi_i\}$ for S may be constructed from it by a variety of techniques (e.g., [3], [6]).

The best-known of these is the Gram-Schmidt-orthonormalization procedure, in which each Ψ_i is computed as the normalized error of the best approximation to ϕ_i by elements in the span of $\{\phi_j\}_{j=1}^{i-1}$, i.e. by successively solving a least-squares approximation problem $m-1$ times. In formulae,

$$
(2.11)\qquad \left.\begin{aligned} \hat{\Psi}_i &= \phi_i - \sum_{j=1}^{i-1} \langle\phi_i,\Psi_j\rangle\, \Psi_j\,, \\ \Psi_i &= \hat{\Psi}_i/\|\hat{\Psi}_i\|\,, \end{aligned}\right\} \qquad i = 1,\ldots,m.
$$

A slight reordering of the computations, resulting in the so-called modified Gram-Schmidt-process, has proven to be more stable in practice:

$$
(2.12)\qquad \left.\begin{aligned} \phi_i^{(1)} &= \phi_i \\ \phi_i^{(j+1)} &= \phi_i^{(j)} - \langle\phi_i^{(j)},\, \Psi_j\rangle\, \Psi_j\,,\quad j=1,\ldots,i-1 \\ \Psi_i &= \phi_i^{(i)}/\|\phi_i^{(i)}\| \end{aligned}\right\} \qquad i=1,\ldots,m.
$$

The reader should refer to [7] and [4] for some experimental results, and to [1] for a rigorous comparative analysis à la Wilkinson of the two computational processes.

The algorithm under discussion uses the trapezoidal sum approximation to

$$
\int_{x_1}^{x_N} f(x)\; g(x)\; w(x)dx
$$

as inner product, i.e.,

$$
(2.13)\qquad \langle f,g\rangle = \sum_{i=1}^{N} [f(x_{i-1})g(x_{i-1}) + f(x_i)g(x_i)]W_i,
$$

with $W_i = (w(x_{i-1}) + w(x_i))(x_i - x_{i-1})/4$.

where $X = \{x_i | i=1,\ldots,N\}$ is a given finite point set and $w(x)$ is a non-negative function, both to be supplied by the user. Hence M may be taken as the set of all real functions on X. The set S consists of all functions of the form

$$t(x)s(x), \qquad s(x) \in S_\pi^3,$$

where $\pi: \xi_o < \xi_1 < \ldots < \xi_{k+1}$ is a fixed knot set and $t(x)$ a trend function to be supplied by the user. We will ignore the presence of $t(x)$ in the subsequent discussion.

It has been our experience that a careful choice of the initial basis $\{\phi_i\}$ for S can greatly increase the reliability of the subsequent calculation of the L_2- approximation to u via the modified G.-S. process. A straightforward but costly approach would consist in reinforcement, i.e., in the repeated application of the modified G.-S. process until Repr. II or Repr. III of the basis elements becomes stationary. The algorithm under discussion permits this approach if desired (cf. below the case MODE = 2 in the algorithm NUBAS). Less costly would be the construction of a "nearly" orthogonal basis. Vague as this term is, the following process is based on this notion, and has proven quite successful: construct each ϕ_i so as to have at least one more extremum than Ψ_{i-1}.

It is also mandatory that computation of the inner products be made somewhat more accurately than the other computations. This may be accomplished by "double precision accumulation" of the products, or, as in this algorithm, complete double precision arithmetic in the inner product calculations.

3. The algorithm.

(a) General remarks. As stated earlier, the success of approximation by splines depends heavily on the correct choice of the knot set Ξ. The algorithm FXDKNT is, therefore, designed to permit the experimentation with various choices of Ξ in as economical a fashion as possible. This is done by using four modes of operation.

An initial call to FXDKNT, which must be in MØDE = 0, produces the L.-S. approximation to the given u using a specified knot set Ξ. Subsequent calls may be used to modify repeatedly the current knot set. Thus more knots may be added while retaining all or at least the first KNØT knots in Ξ (MØDE = 1,2). MØDE = 3 permits the efficient evaluation of the L.-S. error as a function of one additional knot to be inserted between two neighboring knots, thus making it possible to minimize the L.-S. error with respect to one knot with relatively little work.

(b) Input. The input to FXDKNT consists of:

(i) The integer MØDE which is assumed to be one of 0,1,2,3: A call with MØDE = 0 will change MØDE to 1; a call with MØDE = 2 may change MØDE to 1.

(ii) LX abscissa and ordinates, XX(L),U(L),L=1,...,LX, of the function $u(x)$ to be approximated.

The numbers XX(L) are assumed to be increasing with L, and should normally be strictly increasing. A quick look at the inner product (2.13) shows that repeated points

$$XX(L-1) < X(L) = X(L+1) = \ldots = X(M) < X(M+1)$$

are effectively ignored unless U(L) $\neq$ U(M) in which case u is treated as if it had a jump discontinuity at XX(L) of size U(M) - U(L).

(iii) (in MØDE = 0,1,2) the set of (additional) knots ADDXI(i), i=1,...,JADD:

If MØDE = 0, then ADDXI(1) and ADDXI(2) are taken as the left and right boundary knot, respectively. The only restriction on the remaining entries, if any, (or on the entries in any subsequent call) is that each should fall within this interval and not be coincident with any knot already in use (an error message will result in the contrary case). In particular, the entries of ADDXI need not be ordered in any way. JADD may be zero (or even negative) to signify "no additional knots".

(iv) (in MØDE = 1,2) the integer KNØT.

This number is part of the information returned by FXDKNT; but if it is decreased between two calls to FXDKNT by an amount M, the M knots introduced last in prior calls will be removed from the current knot set.

(v) The number ARG:

ARG is taken to be a real number in MØDE = 3, giving the current value of the one knot being varied. If MØDE $\neq$ 3, ARG is taken to be an integer between 0 and 3, specifying various output options.

(c) <u>The output</u>. The output of (information returned from) FXDKNT consists of:

(i) The number FXDKNT = $||u-u^*||^2/(XX(LX)-XX(1))$, giving the L.-S. error of the current best approximation to u;

(ii) The current knot set XIL(i), i = 1,...,KNØT.
The entries of XIL are increasing with i, XIL contains the boundary knots.

(iii) (MØDE $\neq$ 3) the values UERROR(L) of u-u* at XX(L), L = 1,...LX, u* being the b. a. to u by cubic splines on the current knot set.

(iv) (MØDE $\neq$ 3 and ARG = 1) Repr. II, I, IV of u* in VØRDL, CØEFL, and FCTL, respectively; and the integer LMAX, indicating that (u-u*)w attains its maximum at XX(LMAX).

(v) In addition, FXDKNT has some printed output in case ARG > 0, and MØDE $\neq$ 3.

(d) <u>The algorithm NUBAS</u>. The heart of the FXDKNT algorithm is the repeated solution of the following problem:

Given an orthonormal basis $\{\Psi_i\}$ for the linear space S of all cubic splines on

$$\pi: \text{XIL}(1) < \ldots < \text{XIL}(\text{KN}\Theta\text{T})$$

and the L.-S. approximation u* to u by elements in S, find the L.-S. approximation $\hat{u}^*$ to u by elements in $\hat{S}$, where $\hat{S} \supset S$ is the linear space of all cubic splines on

$$\hat{\pi}: \text{XIL}(1) < \ldots < \text{XIL}(\text{INSERT-1}) < \text{XKN}\Theta\text{T} < \text{XIL}(\text{INSERT}) < \ldots < \text{XIL}(\text{KN}\Theta\text{T}).$$

This problem is solved in NUBAS.

Thus, initially, one has present, for each Ψ_i, Repr. II in VØRD(i,.,.), Repr. I in XI(.), CØEF(.,.), and Repr. IV in FCT(.,i); further one has u-u* in UERROR, and $<u,\Psi_i>$ in BC(i).

KNØT is increased by one, and the current knot set XIL is enlarged by the insertion of the additional knot XKNØT so that XIL contains the knots again in increasing order. Repr. II for the Ψ_i's is updated to include Ψ_i(XKNØT) and Ψ_i'(XKNØT), while the other two representations remain unchanged.

Next, with ILAST = KNOT + 2, an element ϕ_{ILAST} of $\hat{S}$ but not in S is constructed as that element of $\hat{S}$ which interpolates a certain function f on the current knot set. The choice of f depends on MØDE.

If MØDE = 1, then, with ILM1 = ILAST-1,

$$f(x) = \begin{cases} \Psi_{\text{ILM1}}(X), & X \leq \text{XKNØT}, \\ -\Psi_{\text{ILM1}}(X), & X > \text{XKNØT}, \end{cases}$$

thus making it quite likely that ϕ_{ILAST} has one more local extremum then Ψ_{ILM1}.

If the reinforcing mode MØDE = 2 is used,

$$f(x) = \Psi_{\text{ILAST}}$$

is chosen provided that such a function was in fact constructed during an earlier call to FXDKNT. Otherwise, MØDE is set to one, and the algorithm proceeds in that mode.

Repr. III for ϕ_{ILAST} is computed from f and stored in VØRDL and is then augmented to Repr. II in the subroutine INTERP, using equations (2.6). Subroutine EVAL then supplies Repr. I using (2.4), storing it in CØEFL, and, from it, Repr. IV, storing it in FCTL.

The modified Gram-Schmidt-process is then applied. Specifically, the components TEMP(i) = $<\phi_{ILAST}, \Psi_i>$ of ϕ_{ILAST} with respect to the orthonormal basic $\{\Psi_i | i=1,\ldots,ILM1\}$ of S are computed by

$$\left.\begin{array}{l} \mathrm{TEMP}(i) \leftarrow \langle \mathrm{FCTL}, \mathrm{FCT}(i)\rangle \\ \mathrm{FCTL} \leftarrow \mathrm{FCTL} - \mathrm{TEMP}(i)*\mathrm{FCT}(i) \end{array}\right\} \quad i=1,\ldots,\mathrm{ILM1},$$

the inner product $<\phi^{(i)}_{ILAST}, \Psi_i>$ being computed in subroutine DØT using Repr. IV of the functions involved.

Hence, after the calculation

$$\mathrm{V\Theta RDL} \leftarrow \mathrm{V\Theta RDL} - \sum_{i=1}^{\mathrm{ILM1}} \mathrm{TEMP}(i)*\mathrm{V\Theta RD}(i),$$

VØRDL contains Repr. II of a cubic spline in $\hat{S}$ orthogonal to S.

Another call to EVAL derives from this Repr. I and IV. Finally, Repr. I, II, IV of the Ψ_{ILAST} is stored via

$$\begin{array}{l} \mathrm{C} \leftarrow \sqrt{\langle \mathrm{FCTL}, \mathrm{FCTL}\rangle} \\ \mathrm{C\Theta EF} \leftarrow \mathrm{C\Theta EFL/C} \\ \mathrm{V\Theta RD(ILAST)} \leftarrow \mathrm{V\Theta RDL/C} \\ \mathrm{FCT(ILAST)} \leftarrow \mathrm{FCTL/C} \end{array}$$

Also, the component BC(ILAST) of u with respect to Ψ_{ILAST} is computed as

$$\mathrm{BC(ILAST)} \leftarrow \langle \mathrm{UERROR}, \mathrm{FCTL}\rangle/\mathrm{C}.$$

Except in MØDE = 3, a call to NUBAS is followed by

$$\mathrm{UERROR} \leftarrow \mathrm{UERROR} - \mathrm{BC(ILAST)} * \mathrm{FCT(ILAST)},$$

so that UERROR contains $u-\hat{u}^*$.

For MØDE = 0 and MØDE = 3, there are minor modifications in NUBAS. In case MØDE = 0, one of the first four Ψ_i is computed so that, in the above, "with one additional knot" has to be replacad by "of one degree higher". Explicitly, for i=1,2,3,4, ϕ_i, and hence Ψ_i, is a polynomial of degree i-1.

If MØDE = 3, XKNOT is not taken as an additional knot but rather as a new value for the knot introduced last. Accordingly, the current knot set is changed (at that knot) but not increased, and ϕ_{ILAST} is then defined as in MØDE = 2.

(e) The algorithm FXDKNT. FXDKNT uses NUBAS in the following way.

MØDE = 0. U is put into UERROR, trend and weight are evaluated at the XX's, the quantities W_i (see (2.13)) are computed and stored in TRPZWT. The initial knot set is set up to consist of just the two boundary knots which are taken to be ADDXI(1), ADDXI(2). Four calls to NUBAS produce the orthonormal basis $\Psi_1,..,\Psi_4$ for the set of cubic polynomials as described above, their various representations and the L.-S. approximation to u by cubic polynomials. UERROR is saved in CUBERR for possible use later on in a MØDE = 1,2 call. MØDE is set to 1. If JADD-2 > 0, the program proceeds, after

JADD ← JADD-2, ADDXI(i) ← ADDXI(i+2), i=1,...,JADD,

as for MØDE = 1. Otherwise, the L.-S. error of the current L.-S. approximation to u is computed as

FXDKNT ← <UERROR,UERROR>/(XX(LX)-XX(1))

and FXDKNT is terminated.

<u>MODE = 1,2</u>: If KNØT $\geq$ KNØTSV, KNØT is set equal to KNØTSV, and JADD successive calls to NUBAS produce the L.-S. approximation to u by cubic splines having the knots introduced earlier and additional knots ADDXI(i), i=1,...,JADD.

If KNØT < KNØTSV, this action is preceded by the following: The (KNØTSV-KNØT) knots introduced last into the current knot set by a preceding call or calls are removed from it. The various arrays such as UERROR are restored to the stage where we had just computed the L.-S. approximation to u using just the first KNØT knots.

In either case, the program returns the square of the L.-S. error, FXDKNT, of the current b. approximation to u computed as in MØDE = 0.

<u>MØDE = 3</u>. If the previous call to FXDKNT was in a mode other than 3 (MØDE3=FALSE), ARG is taken as the value of an additional knot. The current value of FXDKNT is saved in ERBUT1, and a call to NUBAS in MØDE = 2 with XKNØT $\leftarrow$ ARG produces, as described earlier, an increased knot set, an additional Ψ_{ILAST}, and BC(ILAST) $\leftarrow \langle UERROR, \Psi_{ILAST} \rangle$.

But the component BC(ILAST)* Ψ_{ILAST} of u (or, UERROR), with respect to Ψ_{ILAST} is <u>not</u> taken out of UERROR. Rather, FXDKNT is computed as

FXDKNT $\leftarrow$ ERBUT1 - (BC(ILAST)**2)/(XX(LX)-XX(1)),

using the well known fact that if $u^* = \sum_{i=1}^{ILAST} BC(i)\Psi_i$, then

$$||u-u^*|| = ||u||^2 - \sum_{i=1}^{ILAST} (BC(i))^2$$

$$= ERBUT1 - (BC(ILAST))^2.$$

If the previous call to FXDKNT was in MØDE = 3 (MØDE3=TRUE), ARG is taken as a new value for the additional knot introduced in the first in a sequence of such calls. Hence, a call to NUBAS in MØDE = 3 produces,

4. Variables in this program

Global with calling program:

ADDXI (26)	LX
CØEFL(27,4)	MØDE
FCTL(100)	U(100)
INTERV	UERRØR(100)
JADD	VØRDL(28,2)
KNØT	XIL(28)
LMAX	XX(100)

Global in FXDKNT

BC(30)	TREND(100)
FCT(100,30)	TRPZWT(100)
ILAST	VØRD(30,28,2)
INSIRT(30)	XKNØT
IØRDER(28)	

Local in FXDKNT

ARG = IPRINT = CHANGE	KNØTSV
CUBERR(100)	MØDE3
ERBUT1	PRINT(100)
ERRL1	WEIGHT(100)
ERRL2	XSCALE
ERRL99	

Local in NUBAS

C	ILM1 = ILAST-1
CØEF(381,4)	INSERT
ICLAST	XI(381)

6. <u>Example</u>: The set of data used here has three distinct features: (i) It is actual data, expressing a thermal property of titanium; (ii) It is difficult to approximate by classical approximating functions; (iii) There is a significant amount of noise in the data.

TITANIUM HEAT DATA

x	u(x)	u*(x)	$(u-u^*)\times10^2$	x	u(x)	u*(x)	$(u-u^*)\times10^2$
595	.644	.624	2.03	845	.812	.965	-15.28
605	.622	.636	-1.37	855	.907	1.103	-19.64
615	.638	.643	- .47	865	1.044	1.248	-20.44
625	.649	.646	.29	875	1.336	1.386	- 5.00
635	.652	.647	.52	885	1.881	1.502	37.89
645	.639	.646	- .71	895	2.169	1.583	58.60
655	.646	.645	.08	905	2.075	1.615	46.03
665	.657	.645	1.17	915	1.598	1.583	1.46
675	.652	.647	.46	925	1.211	1.481	-27.01
685	.655	.652	.26	935	.916	1.323	-40.67
695	.664	.659	.45	945	.746	1.129	-38.33
705	.663	.667	- .44	955	.672	.922	-24.98
715	.663	.675	-1.21	965	.627	.721	- 9.41
725	.668	.681	-1.33	975	.615	.548	6.70
735	.676	.685	- .89	985	.607	.424	18.34
745	.676	.685	- .87	995	.606	.369	23.73
755	.686	.679	.66	1005	.609	.395	21.37
765	.679	.669	1.00	1015	.603	.480	12.33
775	.678	.658	2.05	1025	.601	.589	1.17
785	.683	.650	3.31	1035	.603	.691	- 8.84
795	.694	.651	4.29	1045	.601	.753	-15.24
805	.699	.666	3.26	1055	.611	.743	-13.16
815	.710	.701	.93	1065	.601	.626	- 2.54
825	.730	.759	-2.90	1075	.608	.372	23.58
835	.763	.846	-8.34				

The (rounded) values of the Least-squares approximation u^* to u and the error are given alongside the given data. For this approximation, the knot set π was chosen to be uniformly spaced, with 5 interior knots. Apparently, this is a poor choice for the location of the knots, as may seen by comparing u^* with the approximation to u listed in [2].

Other output, as produced by a run of a FORTRAN version of the algorithm on an IBM 7094, includes Repr. I for u^*, and the L_1, L_2, and L_∞ norm of the error, as follows:

Knots	Coefficients	Knots	Coefficients
595	.623718	835	.846403
	$.147983\times10^{-2}$		$.103636\times10^{-1}$
	$-.303437\times10^{-4}$		$.170647\times10^{-3}$
	$.194334\times10^{-6}$		$-.231291\times10^{-5}$
675	.647403	915	$.158343\times10^{1}$
	$.356044\times10^{-3}$		$-.674063\times10^{-2}$
	$.162946\times10^{-4}$		$-.384450\times10^{-3}$
	$-.196743\times10^{-6}$		$.348626\times10^{-5}$
755	.679440	995	.368658
	$-.814283\times10^{-3}$		$-.131654\times10^{-2}$
	$-.309237\times10^{-4}$		$.452251\times10^{-3}$
	$.839879\times10^{-6}$		$-.544051\times10^{-5}$
835		1075	

Average error = .108380, Least Square error = .177236, Maximum error = .586038.

References

[1] A. Björk, Solving linear least-squares problems by Gram-Schmidt orthogonalization, Bit 7 (1967) 1-21.

[2] C.de Boor and J.R. Rice, Least-squares cubic spline approximation II - Variable Knots.

[3] G. Golub, Numerical methods for solving linear least-squares problems, Numer. Math. 7 (1965) 206-216.

[4] T.L. Jordan, Experiments on error growth associated with some linear least-squares procedures, Los Alamos Scientific Laboratory Report LA-3717 (1967).

[5] S.J. Karlin and W.J. Studden, Tchebycheff systems: With applications in analysis and statistics, Interscience, 1966.

[6] M.O. Peach, Simplified technique for constructing orthonormal functions, Bull. Amer. Math. Soc. 50 (1944) 556-641.

[7] J.R. Rice, Experiments on Gram-Schmidt orthogonalization, Math. Comp. 20 (1966) 325-328.

[8] J.R. Rice, The approximation of functions, Vol. II, Chapter 10, Addison-Wesley, 1968.

[9] I.J. Schoenberg and A. Whitney, On Pólya frequency functions III, Transactions Amer. Math. Soc., 74 (1953) 246-259.

```
      FUNCTION FXDKNT (ARG)
C                   THE FUNCTION RETURNS THE SQUARE OF THE L2-ERROR
      DOUBLE PRECISION  TRPZNT,SUM
      LOGICAL MODE3
      DIMENSION WEIGHT(100),CUBERR(100)
      COMMON / NAMDT / TREND(100),TRPZ T(100), PRINT(200)
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),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(100),FCTL(100),XIL(28),COEFL(27,4),
     *                VORDL(28,2),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.AB TO U AT XX(L)
C           COEFL(I,.) CONTAINSTHE 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(100,30),VORD(30,28,2),BC(30),ILAST
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(.,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/ LASTS /IORDER(28),INSIRT(30),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(ILAST)
C           FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).
C     ***LOCAL VARIABLES
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
      EQUIVALENCE (IPRINT,CHANGE)
C           ARG IS EITHER FIXED POINT (MODE.NE.3) TO PICK PRINT-OUT OPTIO
C           OR IS FLOATING POINT (MODE=3) TO GIVE NEW VALUE OF KNOT VARIEL
      CHANGE = ARG
      IF (MODE.GT.0)                     GO TO 20
C----------
C        *** MODE=0* COMPUTE BASIS FCT 1 THROUGH 4 AND  ... TO ... WRTO THESE
C        THEN SET MODE = 1 AND PUT UERROR INTO U.
      XSCALE = XX(LX) - XX(1)
      DO 10 I=5,30
   10 INSIRT(I) = 0
```

```
      DO 11 L=1,LX
      UERROR(L) = U(L)
      TREND(L) = T(XX(L))
   11 WEIGHT(L) = W(XX(L))
      DO 12 L=2,LX
   12 TRPZWT(L) = (XX(L)-XX(L-1))/4.*(WEIGHT(L-1)+WEIGHT(L))
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
   19 UERROR(L) = UERROR(L) - BC(I)*FCT(L,I)
C
      MODE = 1
      DO 20 L = 1,LX
   20 CURERR(L) = UERROR(L)
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
   21 ADDXI(I) = ADDXI(I+2)
                                          GO TO 51
C----------
   29                                     GO TO (40,40,30),MODE
C----------
C        *** MODE=3 *** MERELY REPLACE THE LAST KNOT INTRODUCED BY
C                       CHANGE AND RECOMPUTE L2 ERROR.  CHANGE ENTERS
C                       VIA THE ARGUMENT JPRINT = CHANGE.
C                       THIS MODE SHOULD BE USED FOR
C                                MINIMIZING THE 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(THE PRECEDING CALL WAS IN SOME OTHER MODE
C                       , A FCT IS ADDED WITH CHANGE AS THE ADD. KNOT.
C                       UERRCR IS ASSUMED TO CONTAIN ERROR OF B.A. TO U WRT
C                       ALL PREV FCTNS. **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.
      ERBUT1 = FXDKNT
      MODE = 2
      CALL NUBAS
```

```
      KNOTSV = KNOT
      MODE = 3
                                             GO TO 36
   35 CALL NUBAS
   36 FXDKNT = 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 CAL..
C         40 THROUGH 49 RESTORES ARRAYS IORDER,XIL, UERROR TO THE STATE OF
C         ILAST = KNOT + 2 ,  INVERTING THE ACTION OF DO 11 ... TO 14 IN N
   40 IF (KNOT.LT.KNOTSV)                    GO TO 42
      KNOT = KNOTSV
      IF (.NOT.MODE3)                        GO TO 50
      DO 41 L=1,LX
   41 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)
                                             GO TO 49
   42 DO 43 L=1,LX
   43 UERROR(L) = CURERR(L)
      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)
   44 XIL(K) = XIL(K+1)
   45 ILAST = ILAST-1
      DO 47 I=5,ILAST
      DO 47 L=1,LX
   47 UERROR(L) = UERROR(L) - BC(I)*FCT(L,I)
                                             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
   52 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)
C
   60 FXDKNT= DOT(31,2)/XSCALE
      KNOTSV = KNOT
   61 MODE3 = .FALSE.
      IF (IPRINT.EQ.0)                 RETURN
C          VARIOUS PRINTING IS DONE DEP ON THE ARG = IPRINT
                                       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) 2,4,4
C                    2 I=I-1
C                      IF(I) 3,3,1
C                    3 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
   71 SUM=SUM + BC(J)*VORD(J,ILOC,L)
   72 VORDL(I,L) = SUM
      CALL EVAL
      DO 73 I=1,INTERV
      WRITE(6,620) I,XIL(I)
   73 WRITE (6,630) (J,COEFL(I   ,J),J=1,4)
      WRITE (6,620) KNOT,XIL(KNOT)
  610 FORMAT(42X,5HKNOTS,22X,18HCUBIC COEFFICIENTS//)
  620 FORMAT(35X, 3HXI(, I2, 3H) =, F12.6)
  630 FORMAT(67X,2HC(,I1,3H) =,E16.6)
C
C        **COMPUTE L2, L1, MAX ERRORS AND PRINT
   80 ERRL2 = SQRT(FXDKNT)
      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
   93 PRINT (L) = UERROR(L)*SCALE
                                          GO TO (94,95,95),IPRINT
   94 WRITE (6,621) IE,(L,XX(L),FCTL(L),PRINT(L),L=1,LX)
                                          GO TO 96
   95 WRITE (6,622) IE,(L,XX(L),PRINT(L),L=1,LX)
   96                                     RETURN
  621 FORMAT(1H //45X,36HAPPROXIMATION AND SCALED ERROR CURVE/38X,
     *10HDATA POINT,7X,13HAPPROXIMATION,3X,16HDEVIATION X 10E+,I1/
     *(31X,I4,F16.8,F16.8,F17.6))
  622 FORMAT(1H //58X, 11HERROR CURVE/38X, 10HDATA POINT, 23X,
     116HDEVIATION X 10E+,I1/(31X,I4,F16.8,16X,F17.6))
  623 FORMAT(1H ///40X20HLEAST SQUARE ERROR =,F20.6/
     1              40X20HAVERAGE ERROR      =,F20.6/
     2              40X20HMAXIMUM ERROR      =,F20.6,3H 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
      DIMENSION D(28), DIAG(28)
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                 VORDL(28,2),KNOT,LMAX,INTERV
      DATA DIAG(1),D(1)/1.,0./
      DO 10 M=2,KNOT
      D(M) = XIL(M) - XIL(M-1)
   10 DIAG(M) = (VORDL(M,1)-VORDL(M-1,1))/D(M)
      DO 20 M=2,INTERV
      VORDL(M,2) = 3.*(D(M)*DIAG(M+1) + D(M+1)*DIAG(M))
   20 DIAG(M) = 2.*(D(M)+D(M+1))
      DO 30 M=2,INTERV
      G = -D(M+1)/DIAG(M-1)
      DIAG(M) = DIAG(M) + G*D(M-1)
   30 VORDL(M,2) = VORDL(M,2) + G*VORDL(M-1,2)
      NJ = KNOT
```

```
      DO 40 M=2,INTERV
      MJ = MJ - 1
   40 VORDL(MJ,2) = (VORDL(MJ,2) - D(MJ)*VORDL(MJ+1,2))/DIAG(MJ)
                                                 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)
      DOUBLE PRECISION DDOT,G,TRPZWT
      COMMON / WANDT / TREND(100),TRPZWT(100),G(100)
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                VORDL(28,2),KNOT,LMAX,INTERV
      COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST
                                                 GO TO (10,30),INDEX
   10 IF (M.EQ.ILAST)                            GO TO 20
      DO 11 L=1,LX
   11 G(L) = FCT(L,M)*FCTL(L)
                                                 GO TO 80
   20 DO 21 L=1,LX
   21 G(L) = FCTL(L)*FCTL(L)
                                                 GO TO 80
   30 IF (M.EQ.31)                               GO TO 40
      DO 31 L=1,LX
   31 G(L) = FCTL(L)*UERROR(L)
                                                 GO TO 80
   40 DO 41 L=1,LX
   41 G(L) = UERROR(L)*UERROR(L)
   80 DDOT = 0.D0
      DO 81 L=2,LX
   81 DDOT = DDOT + (G(L-1) + G(L))*TRPZWT(L)
C
      DOT = DDOT
                                                 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
      DOUBLE PRECISION G,TRPZWT
      COMMON / WANDT / TREND(100),TRPZWT(100),G(100)
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                VORDL(28,2),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
   10 COEFL(I,4) = DUM2/DX/DX
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)
   20 FCTL(L) = (COEFL(J,1)+DX*(COEFL(J,2)+DX*(COEFL(J,3)
     *                                           +DX*COEFL(J,4))))*TREND(L)
                                              RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE NUBAS
      DOUBLE PRECISION SUM
      COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
      COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
     *                 VORDL(28,2),KNOT,LMAX,INTERV
      COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST
      COMMON/ LASTB /IORDER(28),INSIRT(30),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 ICLAST
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(ILAST
C           FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).
      DIMENSION TEMP(30),XI(381),COEF(381,4)
      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 (10,10,14),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
C
   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))
      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
                                                      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 INTE
C                          POLATING, THIS SHOULD INDUCE ONE MORE OSCILLATI
C                          N IN FCT ILAST THAN IN FCT ILAST-1
```

```
C
   29 MODE = 1
   30 VORDL(1,2) = VORD(ILM1,1,2)
      DO 31 K=1,INSERT
      ILOC = IORDER(K)
   31 VORDL(K,1) = VORD(ILM1,ILOC,1)
      DO 32 K=INSERT,INTERV
      ILOC = IORDER(K+1)
   32 VORDL(K+1,1) =-VORD(ILM1,ILOC,1)
      VORDL(KNOT,2) =-VORD(ILM1,2,2)
                                          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 MODE
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
      DO 69 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
      SUM = 0.D0
      DO 68 I=1,ILM1
   68 SUM = SUM + TEMP(I)*VORD(I,ILOC,L)
   61 VORDL(K,L) = VORDL(K,L) + SUM
   67 CALL EVAL
      C = SQRT(DOT(ILAST,1))
      BC(ILAST) = DOT(ILAST,2)  / C
      DO 62 K=1,KNOT
      ILOC = IORDER(K)
      DO 62 L=1,2
      VORDL(K,L) = VORDL(K,L)/C
   62 VORD(ILAST,ILOC,L) = VORDL(K,L)
      ID = ICLAST - 1
      DO 63 IO=1,INTERV
      ID = ID + 1
      DO 63 L=1,4
   63 COEF(ID,L) = COEFL(IO,L)/C
      DO 64 L=1,LX
   64 FCT(L,ILAST) = FCTL(L)/C
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)
      T = 1.
      RETURN
      END
C
      FUNCTION W(Z)
      W = 1.
      RETURN
      END
```