Least Squares Cubic Spline Approximation I -
Fixed Knots

Carl de Boor and John R. Rice

April 1968

CSD TR 20
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 \textit{(polynomial) spline function of degree} \( n \) \textit{on} \( \pi \) is, by definition, any function \( s(x) \in C^{(n-2)}[a,b] \), which on each of the intervals \( (\xi_i, \xi_{i+1}) \), \( i = 0, \ldots, k \), reduces to a polynomial of degree \( \leq n \). The points \( \xi_i \) are called \textit{knots} (or, \textit{joints}). We denote by \( S^n_\pi \) the linear space of all such functions. Define

\[
(\xi-x)^n_+ = \begin{cases} 
(x-\xi)^n, & x \geq \xi, \\
0, & x < \xi.
\end{cases}
\]

Then it is easily shown that each \( s \in S^n_\pi \) is uniquely represented by two sets of parameters, \( \mathcal{C} = [\xi_1, \ldots, \xi_k] \) and \( A = [a_1, \ldots, a_{k+1}] \), where

\[
s(x) = S(\mathcal{C}, x) = \sum_{i=1}^k a_i (x-\xi_i)^n_+ + \sum_{j=0}^n a_{k+j+1} x^j.
\]

Apparently, the boundary \"knots\" \( \xi_0, \xi_{k+1} \), 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 \( s \in S^n_\pi \) 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:
Repr. I. The set \( \{ \xi_0, \ldots, \xi_k \} \) and the set of polynomial coefficients \( \{ c_{ij} \mid i = 0, \ldots, k; \ j = 0, \ldots, n \} \), where

\[
S(A, \xi_1, x) = \sum_{j=0}^{n} c_{ij} (x-\xi_1)^j, \text{ for } \xi_i \leq x \leq \xi_{i+1}, \ i = 0, \ldots, k.
\]

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

\[ r = \frac{(n+1)}{2}, \]

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

\[
c_{ij}(\Delta \xi_1)^j = \sum_{s=0}^{r-1} \gamma_{ij} [c_{i+1,s}(\Delta \xi_1)^s - \sum_{s=0}^{r-1} \xi_c (\Delta \xi_1)^t],
\]

where

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

This gives

Repr. II. The set \( \{ \xi_0, \ldots, \xi_k \} \) and the set \( \{ c_{ij} \mid i = 0, \ldots, k+1; \ j = 0, \ldots, r-1 \} \),

where

\[
c_{ij} = \frac{1}{j!} \left. \frac{d^j S(A, \xi_1, x)}{dx^j} \right|_{x = \xi_1}.
\]

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 $S(A, x)$ and its derivatives up to the $(r-1)$st. But since $S(A, x)$ is in $C^{(n-2)}[a, b]$, a small subset of the $c_{ij}$ is sufficient.

Repr. III. The set $\{c_{ij}| (j=0 \text{ and } i=0, \ldots, k+1) \text{ or } (j=1, \ldots, r-1, \text{ and } i=0, k+1)\}$.

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:

\[
\begin{align*}
R_i = \sum_{s=0}^{r-1} \gamma_{js}(c_{i-1, s}(\Delta \xi_{i-1}^{s-r-j} + \sum_{\tau=s}^{r-1} (\tau c_{i, \xi}(\Delta \xi_{i}^{\tau-r-j}(\Delta \xi_{i}^{\tau-r-j})}(i, j = 1, \ldots, r-2).
\end{align*}
\]

(2.6)

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

\[
\begin{align*}
s(\xi_i) &= f(\xi_i), \quad i=0, \ldots, k+1, \\
s(\xi_j) &= f(\xi_j), \quad i=0, k+1; \quad j=1, \ldots, r-1.
\end{align*}
\]

(2.7)

The algorithm under discussion employs each of these representations and the following
Repr. IV. The set \( \{ S(A, \omega, x_i) \mid i = 1, \ldots, N \} \), 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 \( \{ S(A, \omega, x_i) \mid i = 1, \ldots, N \} \) represents \( S(A, \omega, x) \) if and only if for some subset \( \hat{X} \) of \( X \) with
\[
\hat{x}_1 < \hat{x}_2 < \cdots < \hat{x}_{n+k+1}
\]
one has
\[
(2.8) \quad \hat{x}_i < \xi_i < \hat{x}_{i+n+1}, \quad i = 1, \ldots, k.
\]

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
\[
(2.9) \quad S(A, \omega, x) = \sum_{i=-n}^{k} b_i B_i(x),
\]
and
\[
b_i(x) = (\xi_{i+n+1} - \xi_i) g_i(\xi_{i+1}, \ldots, \xi_{i+n+1}; x), \quad i = -n, \ldots, k,
\]
\[
g_n(s; x) = (s-x)^n_+,
\]
with
\[
\xi_{-n} \leq \cdots \leq \xi_{-1} \leq a, \quad b \leq \xi_{k+2} \leq \cdots \leq \xi_{k+n+1}.
\]

Here, \( f(\xi_1, \ldots, \xi_{i+n+1}) \) denotes the \( (n+1) \)st divided difference of the function \( f(s) \) on the points \( \xi_1, \ldots, \xi_{i+n+1} \).
It is not difficult to see that

\[ B^1_i(x) \geq 0 \text{ with equality iff } x \in \left( \xi^*_1, \xi^*_{i+n+1} \right) \],

[.] \[ \sum_{i=-n}^{n} B^1_i(x) = 1, \text{ all } x \in \left[ \xi_0^*, \xi_{k+1}^* \right]. \]

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

\[ ||f|| = (\langle f, f \rangle)^{1/2}. \]

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 \( P^i u \) of \( u \), i.e., \( u^* = P^i u \) is determined by \( u^* \in S \), 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 \( \{Y_i\}_{i=1}^{m} \) of \( S \), i.e., a generating set for \( S \) which satisfies

\[ \langle Y_i, Y_j \rangle = \delta_{i,j}, \text{ for } i, j = 1, \ldots, m. \]

For then,

\[ P^i u = \sum_{i=1}^{m} \langle u, Y_i \rangle Y_i. \]

Given a basis \( \{\phi_i\}_{i=1}^{m} \) for \( S \), an orthonormal basis \( \{Y_i\} \) 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 $\hat{Y}_i$ is computed as the normalized error of the best approximation to $\phi_i$ 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,

$$\hat{Y}_i = \phi_i - \sum_{j=1}^{i-1} \langle \phi_i, \phi_j \rangle \phi_j, \quad i = 1, \ldots, m.$$  

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

$$\hat{Y}_i = \phi_i - \sum_{j=1}^{i-1} \langle \phi_i, Y_j \rangle Y_j, \quad Y_i = \hat{Y}_i / |\hat{Y}_i|,$$


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.,

$$\langle f, g \rangle = \sum_{i=1}^{N} \left[ f(x_{i-1}) g(x_{i-1}) + f(x_i) g(x_i) \right] w_i,$$

where $w_i = \frac{1}{2} (x_{i+1} - x_i)$.\(\text{\textcopyright}\)
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

\[
t(x)s(x), \quad s(x) \in S^3_n,
\]

where \( \pi : \xi_0 < \xi_1 < \ldots < \xi_{k+1} \) 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 \( \phi_i \) so as to have at least one more extremum than \( \psi_{i-1} \).

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 \( T \). The algorithm FXDKNT is, therefore, designed to permit the experimentation with various choices of \( T \) 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 \( \text{MDE} = 0 \), produces the \( \mathcal{L} - \mathcal{S} \) approximation to the given \( u \) using a specified knot set \( T \). 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 \( \text{KNOT} \) knots in \( T \) (\( \text{MDE} = 1,2 \)). \( \text{MDE} = 3 \) permits the efficient evaluation of the \( \mathcal{L} - \mathcal{S} \) error as a function of one additional knot to be inserted between two neighboring knots, thus making it possible to minimize the \( \mathcal{L} - \mathcal{S} \) error with respect to one knot with relatively little work.

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

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

(ii) \( \text{LX} \) abscissa and ordinates, \( XX(L), U(L), L=1, \ldots, \text{LX} \), 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

\[ XX(L-1) < X(L) = X(L+1) = \ldots = X(H) < X(H+1) \]
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 $M8DE = 0,1,2$) the set of (additional) knots $ADOXI(i), i=1,...,JADD$:

If $M8DE = 0$, then $ADOXI(1)$ and $ADOXI(2)$ 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 $ADOXI$ need not be ordered in any way. $JADD$ may be zero (or even negative) to signify "no additional knots".

(iv) (in $M8DE = 1,2$) the integer $KNST$.

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 $M9DE = 3$, giving the current value of the one knot being varied. If $M9DE \neq 3$, $ARG$ is taken to be an integer between 0 and 3, specifying various output options.

(c) The output. 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 $X_{IL}(i)$, $i = 1, \ldots, \text{KNOT}$.
The entries of $X_{IL}$ are increasing with $i$, $X_{IL}$ contains the boundary knots.

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

(iv) (MODE $\neq 3$ and ARG = 1) Repr. II, I, IV of $\hat{u}$ in \text{VORDL}, \text{CBEFL}, and \text{FCTL}, respectively; and the integer \text{LMAX}, indicating that $(u-\hat{u})w$ attains its maximum at $XX(\text{LMAX})$.

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

(d) The algorithm NU8AS. The heart of the FXDKNT algorithm is the repeated solution of the following problem:

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

$$\pi: X_{IL}(1) < \ldots < X_{IL}(\text{KNOT})$$

and the L.-S. approximation $\hat{u}$ 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

$$\hat{\pi}: X_{IL}(1) < \ldots < X_{IL}(<\text{INSERT}-1) < \text{XKNOT} < X_{IL}(\text{INSERT}) < \ldots < X_{IL}(\text{KNOT}).$$

This problem is solved in NU8AS.

Thus, initially, one has present, for each $y_i$, Repr. II in \text{VORD}(i, \ldots), Repr. I in $X_{IL}(\ldots)$, $\text{CBEF}(\ldots)$, and Repr. IV in \text{FCT}(\ldots)$; further one has $(u-\hat{u})$ in \text{UERROR}, and $\langle u, y_i \rangle$ in \text{BC}(i).
KNOT is increased by one, and the current knot set XIL is enlarged by the insertion of the additional knot XKN0T so that XIL contains the knots again in increasing order. Repr. II for the $\Psi_i$'s is updated to include $\Psi_i(XKN0T)$ and $\Psi_i(XKN0T)$, while the other two representations remain unchanged.

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

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

$$\begin{cases} 
\Psi_{ILM1}(x), & x \leq XKN0T, \\
-\Psi_{ILM1}(x), & x > XKN0T, 
\end{cases}$$

thus making it quite likely that $\phi_{ILAST}$ has one more local extremum than $\Psi_{ILM1}$.

If the reinforcing mode MODE = 2 is used,

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

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

Repr. III for $\phi_{ILAST}$ is computed from $f$ and stored in VERDL 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 CBEFL, and, from it, Repr. IV, storing it in FCTL.
The modified Gram-Schmidt process is then applied. Specifically, the components $\text{TEMP}(i) = \langle \phi_{\text{ILAST}}^i, \psi_i \rangle$ of $\phi_{\text{ILAST}}^i$ with respect to the orthonormal basis $\{\psi_i\}_{i=1, \ldots, \text{ILM1}}$ of $S$ are computed by

$$\text{TEMP}(i) = \langle \text{FCTL}, \text{FCT}(i) \rangle$$

$$\text{FCTL} = \text{FCTL} - \text{TEMP}(i) \times \text{FCT}(i)$$

the inner product $\langle \phi_{\text{ILAST}}^i, \psi_i \rangle$ being computed in subroutine DOT using Repr. IV of the functions involved.

Hence, after the calculation

$$\text{VORDL} = \text{VORDL} - \sum_{i=1}^\text{ILM1} \text{TEMP}(i) \times \text{VORD}(i),$$

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

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

$$C = \sqrt{\langle \text{FCTL}, \text{FCTL} \rangle}$$

$$C\text{EFL} = C\text{EFL}/C$$

$$\text{VORD}(\text{ILAST}) = \text{VORDL}/C$$

$$\text{FCT}(\text{ILAST}) = \text{FCTL}/C$$

Also, the component $\text{BC}(\text{ILAST})$ of $u$ with respect to $\psi_{\text{ILAST}}$ is computed as

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

Except in $\text{MODE} = 3$, a call to NUBAS is followed by

$$\text{UERROR} = \text{UERROR} - \text{BC}(\text{ILAST}) \times \text{FCT(\text{ILAST})},$$

so that $\text{UERROR}$ contains $u-u^\#$. 
For \( M9DE = 0 \) and \( M8DE = 3 \), there are minor modifications in NUBAS. In case \( M9DE = 0 \), one of the first four \( \psi_i \) is computed so that, in the above, 'with one additional knot' has to be replaced by 'of one degree higher'. Explicitly, for \( i=1,2,3,4 \), \( \alpha_i \) and hence \( \alpha_i \), is a polynomial of degree \( i-1 \).

If \( M9DE = 3 \), \( XKNOT \) 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 \( J_{LAST} \) is then defined as in \( M9DE = 2 \).

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

\( M9DE = 0 \). \( U \) 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 \( \psi_1, \ldots, \psi_4 \) 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 \( M9DE = 1,2 \) call. \( M9DE \) is set to 1. If \( JADD > 2 \), the program proceeds, after

\( JADD - JADD, \ ADDXI(i) - ADDXI(i+2), i=1,\ldots,JADD, \)

as for \( M9DE = 1 \). 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.
MODE = 1, 2: If \( KN9T \geq KN9TSV \), \( KN9T \) is set equal to \( KN9TSV \), and JADD successive calls to NUBAS produce the L.-S. approximation to \( u \) by cubic splines having the knots introduced earlier and additional knots \( ADDX(i), i=1, \ldots, JADD \).

If \( KN9T < KN9TSV \), this action is preceded by the following:

The \((KN9TSV-KN9T)\) 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 \( KN9T \) 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 \( MODE = 0 \).

MODE = 3. If the previous call to \( FXDKNT \) was in a mode other than 3 (\( MODE3=FALSE \)), \( ARG \) is taken as the value of an additional knot. The current value of \( FXDKNT \) is saved in \( ERBUT1 \), and a call to NUBAS in \( MODE = 2 \) with \( XXKN9T = ARG \) produces, as described earlier, an increased knot set, an additional \( \psi_{ILAST} \) and \( BC(ILAST) = <UERROR, \psi_{ILAST}> \).

But the component \( BC(ILAST) \times \psi_{ILAST} \) of \( u \) (or, \( UERROR \)), with respect to \( \psi_{ILAST} \) is not taken out of \( UERROR \). Rather, \( FXDKNT \) is computed as

\[
FXDKNT = ERBUT1 - (BC(ILAST)^2)/(XX(LX)-XX(1)),
\]

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

\[
||u-u^*||^2 = ||u||^2 - \sum_{i=1}^{ILAST} (BC(i))^2 = ERBUT1 - (BC(ILAST))^2.
\]

If the previous call to \( FXDKNT \) was in \( MODE = 3 \) (\( MODE3=TRUE \)), \( ARG \) 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 \( MODE = 3 \) produces,
4. Variables in this program

Global with calling program:

- AODXI (26)
- CBEFL (27,4)
- FCTL (100)
- INTERV
- JADD
- KN6T
- LMAX
- LMAX
- U(100)
- UERROR (100)
- VORDL (28,2)
- XIL (28)
- XX (100)

Global in FXKNT

- BC (30)
- FCT (100,30)
- ILAST
- INSRT (30)
- IERROR (28)

Local in FXKNT

- ARG = I
- PRINT = CHANGE
- CUBERR (100)
- ERBU1
- ERR1
- ERR2
- ERR99

Local in NUBAS

- C
- CBEF (381,4)
- ICLAST

ILMI = ILAST-1
INSERT
x{i(381)
6. **Example**: 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

<table>
<thead>
<tr>
<th>x</th>
<th>u(x)</th>
<th>u*(x)</th>
<th>(u-u*)x10^2</th>
<th>x</th>
<th>u(x)</th>
<th>u*(x)</th>
<th>(u-u*)x10^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>595</td>
<td>.644</td>
<td>.624</td>
<td>2.03</td>
<td>845</td>
<td>.812</td>
<td>.965</td>
<td>-15.28</td>
</tr>
<tr>
<td>605</td>
<td>.622</td>
<td>.636</td>
<td>-1.37</td>
<td>855</td>
<td>.907</td>
<td>1.103</td>
<td>-19.64</td>
</tr>
<tr>
<td>615</td>
<td>.638</td>
<td>.643</td>
<td>-0.47</td>
<td>865</td>
<td>1.064</td>
<td>1.248</td>
<td>-20.44</td>
</tr>
<tr>
<td>625</td>
<td>.649</td>
<td>.646</td>
<td>0.29</td>
<td>875</td>
<td>1.336</td>
<td>1.386</td>
<td>-5.00</td>
</tr>
<tr>
<td>635</td>
<td>.652</td>
<td>.647</td>
<td>0.52</td>
<td>885</td>
<td>1.881</td>
<td>1.502</td>
<td>37.89</td>
</tr>
<tr>
<td>645</td>
<td>.639</td>
<td>.646</td>
<td>-0.71</td>
<td>895</td>
<td>2.169</td>
<td>1.583</td>
<td>58.60</td>
</tr>
<tr>
<td>655</td>
<td>.646</td>
<td>.645</td>
<td>0.08</td>
<td>905</td>
<td>2.075</td>
<td>1.615</td>
<td>46.03</td>
</tr>
<tr>
<td>665</td>
<td>.657</td>
<td>.645</td>
<td>1.17</td>
<td>915</td>
<td>1.598</td>
<td>1.583</td>
<td>-1.46</td>
</tr>
<tr>
<td>675</td>
<td>.652</td>
<td>.647</td>
<td>0.46</td>
<td>925</td>
<td>1.211</td>
<td>1.481</td>
<td>-27.01</td>
</tr>
<tr>
<td>685</td>
<td>.655</td>
<td>.652</td>
<td>0.26</td>
<td>935</td>
<td>.916</td>
<td>1.323</td>
<td>-40.67</td>
</tr>
<tr>
<td>695</td>
<td>.664</td>
<td>.659</td>
<td>0.45</td>
<td>945</td>
<td>.746</td>
<td>1.129</td>
<td>-38.33</td>
</tr>
<tr>
<td>705</td>
<td>.663</td>
<td>.667</td>
<td>-0.44</td>
<td>955</td>
<td>.672</td>
<td>.922</td>
<td>-24.98</td>
</tr>
<tr>
<td>715</td>
<td>.663</td>
<td>.675</td>
<td>-1.21</td>
<td>965</td>
<td>.627</td>
<td>.721</td>
<td>-9.41</td>
</tr>
<tr>
<td>725</td>
<td>.668</td>
<td>.681</td>
<td>-1.33</td>
<td>975</td>
<td>.615</td>
<td>.548</td>
<td>6.70</td>
</tr>
<tr>
<td>735</td>
<td>.676</td>
<td>.685</td>
<td>-0.89</td>
<td>985</td>
<td>.607</td>
<td>.424</td>
<td>18.34</td>
</tr>
<tr>
<td>745</td>
<td>.676</td>
<td>.685</td>
<td>-0.87</td>
<td>995</td>
<td>.606</td>
<td>.369</td>
<td>23.73</td>
</tr>
<tr>
<td>755</td>
<td>.686</td>
<td>.679</td>
<td>0.66</td>
<td>1005</td>
<td>.609</td>
<td>.395</td>
<td>21.37</td>
</tr>
<tr>
<td>765</td>
<td>.679</td>
<td>.669</td>
<td>1.00</td>
<td>1015</td>
<td>.603</td>
<td>.480</td>
<td>12.33</td>
</tr>
<tr>
<td>775</td>
<td>.678</td>
<td>.658</td>
<td>2.05</td>
<td>1025</td>
<td>.601</td>
<td>.589</td>
<td>1.17</td>
</tr>
<tr>
<td>785</td>
<td>.683</td>
<td>.650</td>
<td>3.31</td>
<td>1035</td>
<td>.603</td>
<td>.691</td>
<td>-8.84</td>
</tr>
<tr>
<td>795</td>
<td>.694</td>
<td>.651</td>
<td>4.29</td>
<td>1045</td>
<td>.601</td>
<td>.753</td>
<td>-15.24</td>
</tr>
<tr>
<td>805</td>
<td>.699</td>
<td>.666</td>
<td>3.26</td>
<td>1055</td>
<td>.611</td>
<td>.743</td>
<td>-13.16</td>
</tr>
<tr>
<td>815</td>
<td>.710</td>
<td>.701</td>
<td>0.93</td>
<td>1065</td>
<td>.601</td>
<td>.626</td>
<td>-2.54</td>
</tr>
<tr>
<td>825</td>
<td>.730</td>
<td>.759</td>
<td>-2.90</td>
<td>1075</td>
<td>.608</td>
<td>.372</td>
<td>23.58</td>
</tr>
<tr>
<td>835</td>
<td>.763</td>
<td>.846</td>
<td>-8.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 \( \pi \) was chosen to be uniformly spaced, with 5 interior knots. Apparently, this is a poor choice for the location of the knots, as may be 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. 1 for \( u^* \), and the \( L_1 \), \( L_2 \), and \( L_\infty \) norm of the error, as follows:

<table>
<thead>
<tr>
<th>Knots</th>
<th>Coefficients</th>
<th>Knots</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>595</td>
<td>.623718</td>
<td>835</td>
<td>.846403</td>
</tr>
<tr>
<td></td>
<td>.147983x10^-2</td>
<td></td>
<td>.103636x10^-1</td>
</tr>
<tr>
<td></td>
<td>-.303437x10^-4</td>
<td></td>
<td>.170647x10^-3</td>
</tr>
<tr>
<td></td>
<td>.194334x10^-6</td>
<td></td>
<td>-.231291x10^-5</td>
</tr>
<tr>
<td>675</td>
<td>.647403</td>
<td>915</td>
<td>.158343x10^1</td>
</tr>
<tr>
<td></td>
<td>.356044x10^-3</td>
<td></td>
<td>-.674063x10^-2</td>
</tr>
<tr>
<td></td>
<td>.162946x10^-4</td>
<td></td>
<td>-.384450x10^-3</td>
</tr>
<tr>
<td></td>
<td>-.196743x10^-6</td>
<td></td>
<td>.348626x10^-5</td>
</tr>
<tr>
<td>755</td>
<td>.679440</td>
<td>995</td>
<td>.368658</td>
</tr>
<tr>
<td></td>
<td>-.814281x10^-3</td>
<td></td>
<td>-.131654x10^-2</td>
</tr>
<tr>
<td></td>
<td>-.309237x10^-4</td>
<td></td>
<td>.452251x10^-3</td>
</tr>
<tr>
<td></td>
<td>.839879x10^-6</td>
<td></td>
<td>-.544051x10^-5</td>
</tr>
</tbody>
</table>

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


FUNCTION FXDKIT (ARG)
DOUBLE PRECISION TRPZT, SUM
LOGICAL CODE3
DIMENSION HEIGHT(100), CUBERRR(100)
COMMON/ TPSST / TREND(100), TRPZ T(100), PRINT(200)
COMMON/ INPUT/LXL XX(100), U(100), JADD, JUXIX(26), MODE
U(J) = FCT TO BE APPR AT XX(J), J=1, LX
XX(J) IS ASSIGNED TO BE NONDECREASING WITH J
ALDX(J) = J-TH KNOT TO BE INTRODUCED, J=1, JADD
MODE = 0, 2, 3, .. SEE COMMENTS BELOW ( AND IN MUSTAS).
COMMON / OUTPUT / VORDL(30,2), FCFL(129), COEFL(27,4),
VORDL(28,2), KNOTLMAX, INTERV-
NDER(J) = ERROR OF J-L2 A, TO U, J=1, LX
KNOT = CURRENT NO. OF KNOTS ( INCL BDRY KNOTS)
INTERV = KNOT - 1 = CURRENT NO. OF INTERVALS (POL.PIECES)
XIL(K) = KTH KNOT, CURRENT (ORDRED) SET OF KNOTS
THE MAXIMUM ERROR OCCURS AT XX(LX)
IF (ARG=1), FCFL(J) CONTAINS THE CURRENT HAB TO U AT XX(J)
COEFL(J,1) CONTAINS THE POL.COEFS. OF I-TH INTERVAL FOR J, A.
VORDL(J,1) CONTAINS VALUE AND DERIV. OF J, A, AT XIL(J)
COMMON / BASIS / FCFL(30), VORDL(30,28-2), BC(30), ILAST
FCFL(J,1) = BASIS FCT M AT XX(J)
VORDL(J,1) CONTAINS THE ODDS (L=1) AND SLOPES (L=2) OF FCT K
AT THE KNOT INTRODUCED AS K-TH, CORRELATION TO ORDERING OF
KNOTS BY SIZE IS DON'T VIOL ORDER, I.E., ORDER AND SLOPE AT
XIL(K) ARE IN VORDL(KORDER(K)).
SC(J) = COORDINATE OF U (AND OF .A. TO U) INTRO I-TH G, FCT
ILAST = CURRENT NO. OF BASIS FCTS
COMMON / LAST: / ORDER(28), INSERT(32), XIL
THE FCT (LAST TO USE) INTRODUCED LAST HAS ADDITIONAL KNOT
XKNOT, THE KNOT JUST INTRO-
DUCED HAS INDEX INSERT IN XIL, INSERT IS SAVED IN INSERT (ILAST)
FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).

***LOCAL VARIABLES
XSCALE = XX(LX) - XX(1) , USED TO STABILIZE INNER PRODUCT
LENGTH = LENGTH OF THE INTERVAL OF INTEGRATION
KNOTSV = NO. OF KNOTS USED IN MOST RECENT CALL TO FXDKIT
ERROR = SO. OF L2 ERROR OF APPR USING ALL BUT THE ONE
KNOT BEING VARIED ( USED IN MODE = 3)
CUBERR = ERROR OF J, A, BY CURRICULUM (NEEDED FOR MODE = 2)
CODE3 = TRUE OR FALSE DEP. ON WHETHER BREV. CALL HAS IS
CODE=3 OR NOT
EQUIVALENCE (PRINT, CHANGE)
ARG IS EITHER FIXED POINT (-CODE=3) TO PICK PRINT-OUT OPTION
- OR IS FLOATING POINT (-CODE=3) TO GIVE ME' VALUE OF KNOT VARIAB

CHANGE = ARG
IF (MODE GT 0) GO TO 20

20 *** MODE=2 COMPUTE BASIS FCT 1 THROUGH 6 AND PUT TO U INTO THESE:
XSCALE = XX(LX) - XX(1)
DO 10 I=5,30
10 INSERT(I) = 0
DO 11 L=1,LX
UERROR(L) = U(L)
TREND(L) = T(XX(L))
11 WEIGHT(L) = 'r(XX(L))
Do 12 L=2,LX
12 TRPZ'"(L) = (XX(L)-XX(L-1))/4.*EIGHT(L-1)+EIGHT(L))
C
XIL(1) = ADDXI(1)
XIL(2) = ADDXI(2)
INDX(L) = 1
ORDER(2) = 2
KNOT = X
INTERV = 1
Do 19 K=1,4
19 UERROR(L) = UERROR(L)
GO TO 51
C
C MODE = 1
Do 20 L = 1,LX
20 UERROR(L) = UERROR(L)
IF (JADD-UFL-LF.2), ONLY B.A. BY CURVICS IS COMPUTED
C OTHERWISE, ADDXI(1), I.GT.2, CONTAINS ADDITIONAL KNOTS
JADD = JADD - 2
IF (JADD-LF.5)
DO 21 K=1,JADD
21 ADDXI(J) = ADDXI(I+1)
GO TO 51
C
C-------
C
29
C
C-------
C
MODE = 3 *** HERELY REPLACE THE LAST KNOT INTRODUCED BY
C CHANGE AND RECOPUTE L2 ERROR. CHANGE ENTERS
C VIA THE ARGUMENT JPRINT = CHANGE.
C THIS MODE SHOULD BE USED FOR
C MINIMIZING THE L2-ERROR WRT TO THE KNOT
C INTRODUCED LAST AS IT MINIMIZES THE COMP WORK
C IF MODE = TRUE (I.E., THE PRECEDING CALL TO FXDKNT
C WAS IN MODE=3), THE PROG 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 MODE=FALSE (THE PRECEDING CALL WAS IN SOME OTHER MODE
C a FCT IS ADDED WRT CHANGE AS THE ADD. KNOT.
C UERROR IS ASSUMED TO CONTAIN ERROR OF B.A. TO U WR1
C ALL PREV FCN'S. **NOTE** IF THE NEXT CALL TO FXDKNT
C IS IN A MODE OTHER THAN 3, THE CHANGE PROPOSED
C NOW WILL BE MADE PERMANENT.
C
30 XKNOT = CHANGE
IF (MODE)
MODE = TRUE
END = FXDKNT
MODE = 2
CALL NUJAS
G0 TO 35
KNOTSV = KNOT
MODE = 3
GO TO 36
35 CALL NUHAS
36 FXDXMT = EROUT1 - AC(ILAST)*XSCALE*AC(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 KNOTL=KNOTSV=NO. OF KNOTS USED IN PREV CAL.
C 40 THROUGH 49 RESTORES ARRAYS IORDER, XIL, UERROR TO THE STATE OR
C ILAST = KNOT + 2, INVERTING THE ACTION OF DO 11... TO 14 IN N
40 IF (KNOTL<KNOTSV) GO TO 42
KNOT = KNOTSV
IF (.NOT. CODE3) GO TO 50
DO 41 L=1,LX
41 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,I)
GO TO 49
42 DO 43 L=1,LX
43 UERROR(L) = CUBERR(L)
IF (KNOTL<=21) GO TO 48
IDUM = KNOT + 1
DO 45 IO=IDUM,KNOTSV
INSERT = INSRT(ILAST)
ILM3 = ILAST - 3
DO 44 K=1,ILM3
44 IORDER(K) = IORDER(K+1)
45 ILAST = ILAST-1
DO 47 L=1,LX
46 XIL(K) = XIL(K+1)
47 UERROR(L) = UERROR(L) - SC(I)*FCT(L,1)
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 ***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 U(C(ILAST)) OF U* INTO 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 NUHAS
GO TO 61
DO 52 L=1,LX
52 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)

KNOTSV = KNOT
IF (IPRINT,EO,0) RETURN

VARIABLES PRINTING IS DONE DEP ON THE ARG = IPRINT
GO TO (70,80,90,IPRINT)

COMPUTE COEFFICIENTS OF B*A AND PRINT

ERROR CURVE (SCALED)

THE FOLLOWING FORTRAN CODE FINDS VALUES AT X OF THE
APPROXIMATION FROM THIS OUTPUT----

I=LXI
1 A=X-XI(I)
   IF(A) 2,4,4
   J=I-1
2 IF(J) 3,3
   J=I+1
3 V=P(1,1)+A*P(2,1)+A*(P(3,1)+A*P(4,1))

70 WRITE(6,610)
1 DO 71 J=1,ILAST
   DO 71 K=J,1,ORDER(I)
   SU=SU1+BC(J)*VORD(J,ILCC,L)
71 VORDL(I,L)=SU*

CALL EVAL
72 WRITE(6,620) 1,XIL(I)
73 WRITE (6,630) (J,COEFL(I,J),J=1,4)
74 WRITE (6,640) (KNOT,XI(J),J=1,4)

ERROR CURVES VALUES AT X OF THE
APPROXIMATION FROM THIS OUTPUT----

I=LXI
1 A=X-XI(I)
   IF(A) 2,4,4
   J=I-1
2 IF(J) 3,3
   J=I+1
3 V=P(1,1)+A*P(2,1)+A*(P(3,1)+A*P(4,1))

80 ERRL2 = SORT(FXDST)
81 ERRL1 = ERPL1+DIF
82 CONTINUE
**ROUTINE INTFRP**

**COMPUTE THE SLOPES VORDL(I,2), I=2,KNOT-1 AT INTERIOR**

**KNOTS OF CUBIC SPLINE FOR GIVEN VALUES VORDL(I,1), I=1,KNOT, AT ALL THE KNOTS AND GIVEN BOUNDARY DERIVATIVES**

**DATA DIAG(1),S(1)/1,1/0,**

**DO 10 I=2,KNOT**

**D(I) = XIL(I) - XIL(I-1)**

**10 VORDL(I,2) = VORDL(I,1) - VORDL(I-1,1) / D(I)**

**DO 20 M=2,INTERV**

**VORDL(M,2) = 3*(S(M) - DIAG(M)) + D(M)*DIAG(M)**

**20**

**DO 40 I=2,INTERV**

**C = -D(M+1)/DIAG(M)**

**DIAG(M) = DIAG(M) + C*D(M)**

**30**

**VORDL(M,2) = VORDL(M,1) + C*VORDL(M-1,2)**

**= KNOT**
DO 46 INTERV
   MJ = MJ + 1
46 VORDL(MJ,2) = (VORDL(MJ,1) - D(MJ)*VORDL(MJ+1,2))/D(MJ)
   RETURN
END

C*****************************************************************************
C  FUNCTION DOT (INDEX)
C  COMPUTE INNER PRODUCT OF FCT M WITH FCT ILAST (INDEX=1) OR
DOUBLE PRECISION DDOT,C,TRPZWT
CO::ON / VORDL / TREND(100),TRPZWT(100),G(100)
CO::ON INPUT/LX,XX(100),U(100),JADDO,ADUX(26),ODE
CO::ON/ OUTPUT/UERROR(100,FCT(100),XIL(28),COEFL(27,4),
* VORDL(28,2),KNOT,LMAX,INTERV
CO::ON/ BASIS /FCT(100,30),VORDL(30,28,2),AC(30),ILAST
10 IF (INDEX .NE. ILAST) GO TO 20
   DO 11 L=1,LX
11 C(L) = FCT(L,M)*FCT(ILAST,L)
   GO TO 80
20 IF (INDEX .EQ. 31) GO TO 30
   DO 31 L=1,LX
31 C(L) = FCT(L,L)*UERROR(L)
   GO TO 80
30 IF (INDEX .EQ. 30) GO TO 40
   DO 40 L=1,LX
40 G(L) = UERROR(L)*UERROR(L)
   DO 51 L=1,LX
51 DDOT = DDOT + (G(L-1) + G(L))*TRPZWT(L)
   GO TO 80
41 DO 41 L=1,LX
41 G(L) = UERROR(L)*UERROR(L)
   DDOT = DDOT
   DO 51 L=2,LX
51 DDOT = DDOT + (G(L-1) + G(L))*TRPZWT(L)
   GO TO 80
80 C = DDOT
   RETURN
END

C*****************************************************************************
C  SUBROUTINE EVAL
C  COMPUTE POL. COEFF COEFL(I,X) OF FCT ILAST FROM VORDL,
C  THEN COMPUTE FCT(L,L) = (FCT ILAST)*TREND AT XX(L1,L=1,LX
DOUBLE PRECISION G,TRPZWT
CO::ON / VORDL / TREND(100),TRPZWT(100),G(100)
CO::ON INPUT/LX,XX(100),U(100),JADDO,AC(21,26),ODE
CO::ON OUTPUT/UERROR(100,FCT(100),XIL(28),COEFL(27,4),
* VORDL(28,2),KNOT,LMAX,INTERV
DO 11 I=1,INTERV
   COEFL(I,1) = VORDL(I,1)
   COEFL(I,2) = VORDL(I,2)
   MX = 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

\[ C \]
\[ J = 1 \]
\[ ISWITCH = 1 \]
\[ DO 20 L=1,LX \]
\[ GO TO (11,13), ISWITCH \]
\[ 11 IF \( JF_{INTF} \) GO TO 12 \]
\[ IF \( XX(L) < XIL(J+1) \) GO TO 13 \]
\[ J = J + 1 \]
\[ GO TO 11 \]
\[ 12 ISWITCH = 2 \]
\[ 13 DX = XX(L) - XIL(J) \]
\[ 20 FCFL(L) = (COEFL(J,1)+DX*(COEFL(J,2)+DX*(COEFL(J,3)
\[ + DX*COEFL(J,4))))*TREN(L) \]
\[ RETURN \]
\[ END \]

\[ C \]
\[ SUBROUTINE NUBAS \]
\[ DOUBLE PRECISION SUM \]
\[ COMMON/INPUT/LX,XX(100),U(100),JADD,ADDX(26),MODE \]
\[ COMMON/OUTPUT/UERROR(1001),FCFL(100),XIL(28),\]
\[ COEFL(27,4), VORDL(28,2),KNOT,LMAX,INTERV \]
\[ COMMON/BASIS/FCFL(100,30),VORD(30,28,2),BC(30),ILAST \]
\[ COMMON/LAST3/ORDER(28),INSERT(30),XKNOT \]
\[ COEFL(IC,*) CONTAINS THE POL COEFFICIENTS OF FC FL M FOR INTER \]
\[ \text{VAL TO THE RIGHT OF } XIL(IC), IC=1,IC+M-3, \]
\[ \text{WITH } IC = \#(M-1)/2 + 1 \text{ (WITH OBVIOUS MODS FOR } \#, \#E,4) \]
\[ \text{THE FCFL (TO BE) INTRODUCED LAST, HAS ITS VALUES AT THE } \]
\[ \text{POINTS XX(L) IN FCFL(L), HAS FIRST INDEX } IC LAST \]
\[ \text{IN COEFL AND } XIL \text{, HAS ADDITIONAL KNOT } XKNOT \text{, THE KNOT } XKNOT \]
\[ \text{FOR IT ARE CONTAINED, IN INCREASING ORDER, IN } XIL, XIL \text{ \text{COR }\]
\[ \text{RESPONDING ORDS AND SLOPES ARE IN VORDL, THE KNOT JUST INTRO-} \]
\[ \text{DUCED HAS INDEX INSERT IN } XIL INSERT IS SAVED IN INSERT(ILAST) \]
\[ \text{FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE } \text{MODE} = 2,3,4) \]
\[ \text{DIMENSION TEMP(30),XIL(381),COEFL(381,4) \]
\[ \text{IF } (\text{MODE}=GT,0) \text{ GO TO 8} \]
\[ C-------***\text{CONSTRUCT FCFL LAST FOR ILAST} \LE,4 \]
\[ XIL(ILAST) = XIL(1) \]
\[ ICLAST = ILAST \]
\[ ILM1 = ILAST-1 \]
\[ IF (ILAST=GT,2) \text{ GO TO 7} \]
\[ IF (ILAST=EQ,2) \text{ GO TO 6} \]
\[ C \]
\[ \text{FIRST BASIS FCFL IS A CONSTANT} \]
\[ VORDL(1,1) = 1. \]
\[ VORDL(2,1) = 1. \]
\[ VORDL(1,2) = 0. \]
\[ VORDL(2,2) = 0. \]
\[ \text{GO TO 67} \]
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 99
C--------
8  GO TO (10,10,14),KODE
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
   ILA1 = 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) = XKNOT
   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*(COEF(I,3) + DX*3.0*COEF(I,4))
   ID = 4
   IBOUND = 4
   DO 19 I=5,ILML1
       ID = IO + 1 - I
   19 IBOUND = IBOUND + I - 3
   17 IF (ID.EQ.1999999) GO TO 18
   IF (XKNOT.LT.XIL(ID+1)) GO TO 18
   ID = IO + 1
   GO TO 17
18 DX = XKNOT - XIL(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) + DX*3.0*COEF(ID,4))
C-------
C  DEFINE LAST BASIS FUNCTION
C  *** KODE=1 *** ADD ILAST-TH BASIS FUNCTION CONSTRUCT FROM FCT
C     ILAST-1 BY REFLECTING THE PART OF THE LATTER...0
C     THE RIGHT OF XKNOT ACROSS THE X-AXIS, THEN INTE
C     POLATING, THIS SHOULD INDUCE ONE SINE OSCILLATI
C     IN FCT ILAST THRU IN FCT ILAST-1

29 MODE = 1
30 VORDL(1,2) = VORD(ILM1,1,2)
   DO 31 K=1,INSERT
      ILOC = IORDER(K)
31 VORDL(K,1) = VORD(ILM1,1,ILOC,1)
   DO 32 K=INSERT,INTERV
      ILOC = IORDER(K+1)
32 VORDL(K+1,1) = -VORD(ILM1,1,ILOC,1)
   VORDL(KNRT,2) = -VORD(ILM1,2,2)
   GO TO 55

*** MODE=2 *** REPLACE FCT ILAST BY INTERPOLATING IT AT THE
   CURRENT SET OF KNOTS. IF FCT ILAST HAS NOT BEEN
   PREVIOUSLY DEF (INSRT(ILAST)=0) (SEE 9 ABOVE,
   ALSO MAIN AT 10) SET MODE=1, PROCEED IN THAT MODE
40 IF (INSRT(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
   IF (ID.EQ.IBOUND) GO TO 42
   IF (XIL(K) LT XI(ID+1)) GO TO 42
      ID = ID + 1
      GO TO 41
   DX = XI(K) - XI(ID)
43 VORDL(K,1) = COEF(ID,1) + DX*(COEF(ID,2) + DX*(COEF(ID,3)
      * + DX*COEF(ID,4))
   VORDL(KNRT,1) = VORD(ILAST,2,1)
   VORDL(KNRT,2) = VORD(ILAST,2,2)
   GO TO 55

*** MODE=3 *** CHANGE FCT ILAST BY CHANGING JUST THE KNOT INTRO
   DUCED LAST
50 ID = ICLAST + INSERT - 1
   DX = XKNOT - XI(ID)
   XI(ID) = XKNOT
   IF (DX .GE. 0.) GO TO 51
      ID = ID - 1
51 DX = XKNOT - XI(ID)
   VORDL(INSERT,1) = COEF(ID,1) + DX*(COEF(ID,2) + DX*(COEF(ID,3)
      * + DX*COEF(ID,4))

*** 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) = XI(IO)
   INSRT(ILAST) = INSERT
**ORTHONORMALIZE** FCT ILast OVER **PREVIOUS** (ORTHOMORAL) SET

THEN COMpute THE COMPONENT BC(ILAST) OF ERROR WRT TO IT

FINALLY, STORE THE VARIOUS REPRESENTATIONS OF FCT ILast

**CALL EVAL**

DO 69 I=1,ILS
  TFIP(I) = - DOT(I,1)
DO 69 L=1,LX
69 FCT(L) = FCT(L) + TFIP(I)*FCT(L,1)
DO 61 K=1,KN0T
  ILOC = IORDER(K)
DO 61 L=1,2
  SUM = 0.DO
DO 61 L=1,ILM1
61 SUM = SUM + TFIP(I)*VORD(L,ILOC,L)
61 VORD(L,ILOC) = VORD(L,ILOC,L)/SUM

**CALL EVAL**

C = SORT(DOT(ILAST,1))
  C(ILAST) = DOT(ILAST,2) / C
DO 62 K=1,KN0T
  ILOC = IORDER(K)
DO 62 L=1,2
  VORD(L,ILOC) = VORD(L,ILOC,L)/C
62 VORD(ILAST,ILOC,L) = VORD(L,ILOC,L)
ID = ILast - 1
DO 63 IO=1,INTERV
  ID = ID + 1
DO 63 L=1,4
63 COFL(ID,L) = COEFL(ID,L)/C
DO 64 L=1,LX
64 FCT(ILOC,I-LAST) = FCT(L)/C

**RETURN**

*** THIS OUTPUT INDICATES A FAILURE CONDITION ***

WRITE (6,950) XKTOT,ILAST
950 FORAY (15H *** NEW KNOTS, ETG. & 13K FOR FUNCTION, 13, 5TH OUT OF 10 * UNDS OR COINCIDENT WITH A PREVIOUS KNOT, 36H *** EXECUTION CANNOT BE CONTINUED)

STOP

END

**** AND WEIGHT FUNCTIONS***********************

**FUNCTION T(Z)**

T = 1.
RETURN

**FUNCTION V(Z)**

\[ \pi = 1 \]

RETURN