Least Squares Cubic Spline Approximation I — Fixed Knots

Carl de Boor and John R. Rice

April 1968

Department of Computer Sciences
Purdue University

CSD TR 20

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 I — Fixed Knots

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

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 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.

2 Mathematical Background

2.1 Definition of splines

Let $\pi : a = \xi_0 < \xi_1 < \cdots < \xi_{k+1} = b$ 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^n_π the linear space of all such functions. Define

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

[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.

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

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

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.

2.2 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 $\{c_{ij} | i = 0, \ldots, k; j = 0, \ldots, n\}$, where

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

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

$$\begin{aligned} c_{ij}(\Delta\xi_i)^j = \sum_{s=0}^{r-1} \gamma_{j-r,s} \Big[c_{i+1,s}(\Delta\xi_i)^s - \sum_{t=s}^{r-1} \binom{t}{s} c_{it} (\Delta\xi_i)^t \Big], \\ j = r, \ldots, n; \quad i = 0, \ldots, k, \end{aligned} \tag{2.4}$$

where

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

This gives

Repr. II. The set $\{\xi_0, \ldots, \xi_{k+1}\}$ and the set $\{c_{ij} | i = 0, \ldots, k+1;\ j = 0, \ldots, r-1\}$, 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.

Repr. III. The set $\{\xi_0, \ldots, \xi_{k+1}\}$ and the set $\{c_{ij} | (j = 0$ and $i = 0, \ldots, k+1)$ or $(j = 1, \ldots, r-1$, 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 $k \times (r-1)$ 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:

$$\begin{aligned} &\sum_{s=0}^{r-1} \gamma_{js} [c_{i-1,s}(-\Delta\xi_{i-1})^{s-r-j} + \sum_{t=s}^{r-1} \binom{t}{s} c_{it} \{(\Delta\xi_i)^{t-r-j} - (\Delta\xi_{i-1})^{t-r-j}\} \\ &\qquad -c_{i+1,s}(\Delta\xi_i)^{s-r-j}] = 0, \quad i = 1, \ldots, k; \quad j = 0, \ldots, r-2. \end{aligned} \tag{2.6}$$

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 π as the unique element $s \in S_\pi^n$ satisfying

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

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

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

It should be pointed out [5], [9] that the set $\{S(A, \Xi, x_i) | i = 1, \ldots, N\}$ *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

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

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

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

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

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.

2.3 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

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

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

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

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 (G.-S.) 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 successfully solving a least-squares approximation problem $m-1$ times. In formulae,

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

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

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

The reader should refer to [7] and [4] for some experimental results, and to [1] for a rigorous comparative analysis a 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.,

$$\begin{aligned} \langle f,g\rangle &= \sum_{i=1}^{N}[f(x_{i-1})g(x_{i-1})+f(x_i)g(x_i)]W_i, \\ \text{with } \ W_i &= (w(x_{i-1})+w(x_i))(x_i-x_{i-1})/4, \end{aligned} \tag{2.13}$$

where $X = \{x_i | i = 1, \dots, N\}$ 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 $\pi : \xi_0 < \xi_1 < \cdots < \xi_{k+1}$ 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 calculation.

3 The Algorithm

3.1 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 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 `MODE` = 0, produces the L.-S. approximation to the given u using a specification 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 `KNOT` knots in Ξ (`MODE` = 1,2). `MODE` = 3 permits the efficient evaluation of the least-squares (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.

3.2 Input

The input to `FXDKNT` consists of:

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

(ii) `LX` abscissae 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

$$\texttt{XX(L-1)} < \texttt{X(L)} = \texttt{X(L+1)} = \cdots = \texttt{X(M)} < \texttt{X(M+1)}$$

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 `MODE` = 0,1,2) the sets of (additional) knots `ADDXI(i)`, $i = 1, \ldots,$ `JADD`:

If `MODE` = 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 knot”.

(iv) (in `MODE` = 1,2) the integer `KNOT`.

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 `CHANGE`:

In `MODE` $= 3$, `CHANGE` gives the current value of the one knot being varied. If `MODE` $\neq 3$, the *integer* `IPRINT=IFIX(CHANGE)` is expected to be between 0 and 3, specifying various output options.

3.3 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 `XIL`(i), $i = 1, \ldots,$`KNOT`. The entries of `XIL` are increasing with i, `XIL` contains the boundary knots.

(iii) (`MODE` $\neq 3$) the values `UERROR(L)` of $u - u^*$ at `XX(L)`, `L` $= 1, \ldots,$`LX`, u^* being the best approximation to u by cubic splines on the current set.

(iv) (`MODE` $\neq 3$ and `CHANGE` $= 1$) Repr. II, I, IV of u^* in `VORDL`, `COEFL`, 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 `CHANGE` ≥ 1 and `MODE` $\neq 3$.

3.4 The algorithm `NUBAS`

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

$$\pi\colon \texttt{XIL(1)} < \cdots < \texttt{XIL(KNOT)}$$

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

$$\hat{\pi}\colon \texttt{XIL(1)} < \cdots < \texttt{XIL(INSERT-1)} < \texttt{XKNOT} < \texttt{XIL(INSERT)} < \cdots < \texttt{XIL(KNOT)}\ .$$

This problem is solved in NUBAS.

Thus, initially one has present for each Ψ_i, Repr. II in $\texttt{VORD}(i,\cdot,\cdot)$, Repr. I in $\texttt{XI}(\cdot)$, $\texttt{COEF}(\cdot,\cdot)$, and Repr. IV in $\texttt{FCT}(\cdot,i)$; further one has $u-u^*$ in UERROR, and $\langle u,\Psi_i\rangle$ in $\texttt{BC}(i)$.

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

Next, with $\texttt{ILAST} = \texttt{KNOT} + 2$, an element $\phi_{\texttt{ILAST}}$ 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 MODE.

If $\texttt{MODE} = 1$, then with $\texttt{ILM1} = \texttt{ILAST} - 1$,

$$f(x) = \begin{cases} \Psi_{\texttt{ILM1}}(x) & , \quad x \leq \ \texttt{XKNOT}, \\ -\Psi_{\texttt{ILM1}}(x) & , \quad x > \ \texttt{XKNOT}, \end{cases}$$

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

If the reinforcing mode $\texttt{MODE} = 2$ is used,

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

is chosen provided that such a function was in fact constructed during an earlier call to FXDKNT. Otherwise, MODE is set to 1, and the algorithm proceeds in that mode.

Repr. III for $\phi_{\texttt{ILAST}}$ is computed from f and stored in VORDL 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 COEFL, and, from it, Repr. IV, storing it in FCTL.

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

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

the inner product $\langle \phi^{(i)}_{\texttt{ILAST}}, \Psi_i\rangle$ being computed in subroutine DOT using Repr. IV of the functions involved.

Hence, after the calculation

$$\mathtt{VORDL} \leftarrow \mathtt{VORDL} - \sum_{i=1}^{\mathtt{ILM1}} \mathtt{TEMP}(i) * \mathtt{VORD}(i),$$

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

Another call to EVAL derives from this Repr.s I and IV. Finally, Repr. I, II, IV of the $\Psi_{\mathtt{ILAST}}$ are stored via

$$\mathtt{C} \leftarrow \sqrt{\langle \mathtt{FCTL}, \mathtt{FCTL} \rangle}$$

$$\mathtt{COEF} \leftarrow \mathtt{COEFL/C}$$

$$\mathtt{VORD(ILAST)} \leftarrow \mathtt{VORDL/C}$$

$$\mathtt{FCT(ILAST)} \leftarrow \mathtt{FCTL/C}$$

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

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

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

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

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

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

If $\mathtt{MODE} = 3$, 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 $\phi_{\mathtt{ILAST}}$ is then defined as in $\mathtt{MODE} = 2$.

3.5 The algorithm FXDKNT

FXDKNT uses NUBAS in the following way.

MODE = **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), ADDI(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 MODE = 1,2 call. MODE is set to 1. If JADD $- 2 > 0$, the program proceeds, after

$$\texttt{JADD} \leftarrow \texttt{JADD} - 2, \quad \texttt{ADDXI}(i) \leftarrow \texttt{ADDXI}(i+2), \quad i = 1, \ldots, \texttt{JADD},$$

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

$$\texttt{FXDKNT} \leftarrow \langle \texttt{UERROR}, \texttt{UERROR} \rangle / (\texttt{XX(LX)} - \texttt{XX(1)})$$

and FXDKNT is terminated.

MODE = **1,2**. If KNOT $\geq$ KNOTSV, KNOT is set equal to KNOTSV, 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), $i = 1, \ldots,$JADD.

If KNOT < KNOTSV, this action is preceded by the following: The (KNOTSV–KNOT) 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 KNOT knots.

In either case, the program returns the square of the L.-S. error, FXDKNT, of the current best 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), CHANGE is taken as the value of an additional knot. The most recent value of FXDKNT was earlier saved in ERBUT1, and a call to NUBAS in MODE = 2 with XKNOT $\leftarrow$ CHANGE produces, as described earlier, an increased knot set, an additional $\Psi_{\texttt{ILAST}}$, and BC(ILAST) $\leftarrow \langle \texttt{UERROR}, \Psi_{\texttt{ILAST}} \rangle$.

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

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

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

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

If the previous call to FXDKNT was in MODE $= 3$ (MODE3=.TRUE.), CHANGE 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, as described earlier, a changed $\Psi_{\texttt{ILAST}}$.

4 Variables in this Program

Global with calling program:

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

Global in `FXDKNT`

`BC(30)`	`TREND(100)`
`FCT(100,30)`	`TRPZWT(100)`
`ILAST`	`VORD(30,28,2)`
`INSIRT(30)`	`XKNOT`
`IORDER(28)`	

Local in `FXDKNT`

`IPRINT` = `IFIX(CHANGE)`	`KNOTSV`
`CUBERR(100)`	`MODE3`
`ERBUT1`	`PRINT(100)`
`ERRL1`	`WEIGHT(100)`
`ERRL2`	`XSCALE`
`ERRL99`	

Local in `NUBAS`

`C`	`ILM1` = `ILAST`−1
`COEF(381,4)`	`INSERT`
`ICLAST`	`XI(381)`

5 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

x	$u(x)$	$u^*(x)$	$(u - u^*) \times 10^2$
595	.644	.624	2.03
605	.622	.636	–1.37
615	.638	.643	– .47
625	.649	.646	.29
635	.652	.647	.52
645	.639	.646	– .71
655	.646	.645	.08
665	.657	.645	1.17
675	.652	.647	.46
685	.655	.652	.26
695	.664	.659	.45
705	.663	.667	– .44
715	.663	.675	–1.21
725	.668	.681	–1.33
735	.676	.685	– .89
745	.676	.685	– .87
755	.686	.679	.66
765	.679	.669	1.00
775	.678	.658	2.05
785	.683	.650	3.31
795	.694	.651	4.29
805	.699	.666	3.26
815	.710	.701	.93
825	.730	.759	–2.90
835	.763	.846	–8.34
845	.812	.965	–15.28
855	.907	1.103	–19.64
865	1.044	1.248	–20.44
875	1.336	1.386	– 5.00
885	1.881	1.502	37.89
895	2.169	1.583	58.60
905	2.075	1.615	46.03
915	1.598	1.583	1.46
925	1.211	1.481	–27.01
935	.916	1.323	–40.67
945	.746	1.129	–38.33
955	.672	.922	–24.98
965	.627	.721	– 9.41
975	.615	.548	6.70
985	.607	.424	18.34
995	.606	.369	23.73
1005	.609	.395	21.37
1015	.603	.480	12.33
1025	.601	.589	1.17
1035	.603	.691	– 8.84
1045	.601	.753	–15.24
1055	.611	.743	–13.16
1065	.601	.626	– 2.54
1075	.608	.372	23.58

The (rounded) values of the least-squares approximation u^* to u and the error are given along side 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 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. I for u^*, and the L_1, L_2, and L_∞ norm of the error, is as follows:

Knots	Coefficients	Knots	Coefficients
595	.623718	835	.846403
	$.147983 \times 10^{-2}$		$.103636 \times 10^{-1}$
	$-.303437 \times 10^{-4}$		$.170647 \times 10^{-3}$
	$.194334 \times 10^{-6}$		$-.231291 \times 10^{-5}$
675	.647403	915	$.158343 \times 10^{1}$
	$.356044 \times 10^{-3}$		$-.674063 \times 10^{-2}$
	$.162946 \times 10^{-4}$		$-.384450 \times 10^{-3}$
	$-.196743 \times 10^{-6}$		$.348626 \times 10^{-5}$
755	.679440	995	.368658
	$-.814283 \times 10^{-3}$		$-.131654 \times 10^{-2}$
	$-.309237 \times 10^{-4}$		$.452251 \times 10^{-3}$
	$.839879 \times 10^{-6}$		$-.544051 \times 10^{-5}$
835		1075	

Average error = .108380, Least-Squares error = .177236, Maximum error = .586038

6 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. Technical Report CSD-TR 21, Computer Sciences, Purdue University, April 1968.

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.