Least Squares Cubic Spline Approximation, II - Variable Knots

Carl de Boor

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

Report Number:
68-021
Least Squares Cubic Spline Approximation II — Variable Knots
Carl deBoor and John R. Rice
April 1968
Department of Computer Sciences
Purdue University
CSD TR 21

Versions of the spline programs of deBoor 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
2 Mathematical Background

We assume that the reader is familiar with FIXEDKNOT and we use the notation of that paper. We recall that a spline of degree \( n \) with \( k \) knots \( \Xi = \{\xi_0, \xi_1, \ldots, \xi_{k+1} = b\} \) may be defined by

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

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

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

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

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

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

\[
\left( \int [f(x) - S(A, \Xi, x)]^2 \right)^{1/2}
\]

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

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

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

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

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 It is known from general theoretical results [6], from specific theoretical results [C. deBoor, 1963, unpublished] and from examples that the solution of the least-squares nonlinear approximation problem need not be unique.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

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

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

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

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

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

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

These decisions are delicate in view of the need to achieve some efficiency. Thus these tests have been developed on the basis of experience with a certain class of problems. It is hoped that this class is representative of those met in general. However, these tests may be completely inadequate in new situations. If it is intended to use this algorithm extensively for a certain class of problems, it may well pay to experiment with adjustments in these tests in order to achieve better efficiency with minimum risk.
The initial guess for the knots chosen might be extremely poor and result in reaching a local minimum far from the best approximation. The simple scheme of equal spacing used here to obtain an initial guess might well be modified and improved for certain classes of approximation problems.

3 The Algorithm and Numerical Procedures

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

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

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

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

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

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

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

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

$$e(\text{ARIGHT}), e(\text{ALEFT}) \geq e(A).$$

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

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

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

$$\text{ITER} \leq 4.$$  \hspace{1cm} (3.2)

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

$$\left| \frac{\text{PREVER-ERROR}}{\text{ERROR}} \right| \leq 0.4 \times \text{ACC}$$  \hspace{1cm} (3.3)

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

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

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

(3.4) \[ \text{No more than INDLP guesses for } XI(I). \]

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

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

(3.5) \[ XI(I - 1) + .0625H \leq XI(I) \leq XI(I + 1) - .0625H \]

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

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

(3.6) \[ \frac{e(A) - e(\text{ARIGHT})}{\text{ERROR}} \leq \frac{\text{ACC}}{LXI} \]

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

(3.7) \[ \frac{e(A) - e(\text{ALEFT})}{\text{ERROR}} \leq \frac{\text{ACC}}{LXI}. \]

These criteria are relatively stringent because we feel it is very desirable to be able to enter the parabolic interpolation phase for at least one time. Thus this criterion might not cause termination in OPT even when the decrease in the $L_2$-error is insignificant for the later stages of the algorithm in a reasonable number of cases.
The criterion can be visualized as based on the assumption that the search is converging linearly (or faster) with an error reduction of 1-1/LX1 or smaller. However, the situation here is somewhat different than in SWEEP as we do not necessarily desire to expend effort for an accurate placement of XI(J). That is to say, in the initial stages of the algorithm the set of knots is far enough from optimum that it is wasteful to accurately optimize one of them with the others inaccurately located. It is unusual for this termination criterion to be active in the terminal phases of the algorithm.

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

$$\left| \frac{\text{EPRED} - p(\text{ABEST})}{\text{ERROR}} \right| \leq 5 \cdot \text{ACC}$$

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

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

### 4 Variables in the Program

**Global with FIXEDKNOT**

- `ADDXI(26)`
- `COEFL(27,4)`
- `FCTL(100)`
- `INTERV`
- `JADD`
- `KNOT`
- `LMAX`

**Global in VARYKNOT**

- `ACC`
- `CHANGE`
- `ERROR`

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LX</td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td></td>
</tr>
<tr>
<td>U(100)</td>
<td></td>
</tr>
<tr>
<td>UERROR(100)</td>
<td></td>
</tr>
<tr>
<td>VORDL(28,2)</td>
<td></td>
</tr>
<tr>
<td>XI(28)</td>
<td></td>
</tr>
<tr>
<td>XX(100)</td>
<td></td>
</tr>
</tbody>
</table>
Other Important Variables
A        INFO (16)
ABEST    INTER
ALEFT    KVARY
ARIGHT   NOKNOT
EPRED    PREVER
EPSERR   II

Other Variables
AA       ELEFT
AHIGH    ERIGHT
ALLOW    ETRY
DEL      II
DELX     ITRR
DUMB     K
DXLEFT   LPCNT
DXRIGHT  LXI1 = LXI+1
DYLEFT   LXI2 = LXI+2
DYRIGHT  SGN
E

5 Example

We consider a set of data which has three distinct features: (i) It is actual data (expressing a thermal property of titanium); (ii) It is difficult to approximate using classical techniques; (iii) There is a significant amount of noise in the data.
Titanium Heat Data XX(1), U(I) with approximation U*(I) and error UERROR(I)

<table>
<thead>
<tr>
<th>XX</th>
<th>U</th>
<th>U*</th>
<th>UERROR×10^3</th>
<th>XX</th>
<th>U</th>
<th>U*</th>
<th>UERROR×10^3</th>
</tr>
</thead>
<tbody>
<tr>
<td>595</td>
<td>.644</td>
<td>.619</td>
<td>25.35</td>
<td>845</td>
<td>.812</td>
<td>.796</td>
<td>15.86</td>
</tr>
<tr>
<td>605</td>
<td>.622</td>
<td>.629</td>
<td>-7.22</td>
<td>855</td>
<td>.907</td>
<td>.876</td>
<td>31.27</td>
</tr>
<tr>
<td>615</td>
<td>.638</td>
<td>.638</td>
<td>.24</td>
<td>865</td>
<td>1.044</td>
<td>1.051</td>
<td>-7.38</td>
</tr>
<tr>
<td>625</td>
<td>.649</td>
<td>.644</td>
<td>4.50</td>
<td>875</td>
<td>1.336</td>
<td>1.370</td>
<td>-34.31</td>
</tr>
<tr>
<td>635</td>
<td>.652</td>
<td>.650</td>
<td>2.35</td>
<td>885</td>
<td>1.881</td>
<td>1.838</td>
<td>42.64</td>
</tr>
<tr>
<td>645</td>
<td>.639</td>
<td>.653</td>
<td>-14.46</td>
<td>895</td>
<td>2.169</td>
<td>2.195</td>
<td>-25.98</td>
</tr>
<tr>
<td>655</td>
<td>.646</td>
<td>.656</td>
<td>-10.13</td>
<td>905</td>
<td>2.075</td>
<td>2.078</td>
<td>-3.19</td>
</tr>
<tr>
<td>665</td>
<td>.657</td>
<td>.658</td>
<td>-3.89</td>
<td>915</td>
<td>1.598</td>
<td>1.582</td>
<td>15.62</td>
</tr>
<tr>
<td>675</td>
<td>.652</td>
<td>.659</td>
<td>-6.96</td>
<td>925</td>
<td>1.211</td>
<td>1.197</td>
<td>14.15</td>
</tr>
<tr>
<td>685</td>
<td>.655</td>
<td>.660</td>
<td>-4.57</td>
<td>935</td>
<td>.916</td>
<td>.931</td>
<td>-14.63</td>
</tr>
<tr>
<td>695</td>
<td>.664</td>
<td>.660</td>
<td>4.05</td>
<td>945</td>
<td>.746</td>
<td>.761</td>
<td>-15.36</td>
</tr>
<tr>
<td>705</td>
<td>.663</td>
<td>.660</td>
<td>2.59</td>
<td>955</td>
<td>.672</td>
<td>.687</td>
<td>5.31</td>
</tr>
<tr>
<td>715</td>
<td>.663</td>
<td>.661</td>
<td>2.13</td>
<td>965</td>
<td>.627</td>
<td>.624</td>
<td>2.71</td>
</tr>
<tr>
<td>725</td>
<td>.668</td>
<td>.662</td>
<td>6.12</td>
<td>975</td>
<td>.615</td>
<td>.612</td>
<td>3.20</td>
</tr>
<tr>
<td>735</td>
<td>.676</td>
<td>.664</td>
<td>12.47</td>
<td>985</td>
<td>.607</td>
<td>.608</td>
<td>-1.24</td>
</tr>
<tr>
<td>745</td>
<td>.676</td>
<td>.666</td>
<td>9.33</td>
<td>995</td>
<td>.606</td>
<td>.606</td>
<td>-1.15</td>
</tr>
<tr>
<td>755</td>
<td>.686</td>
<td>.670</td>
<td>16.29</td>
<td>1005</td>
<td>.609</td>
<td>.604</td>
<td>4.62</td>
</tr>
<tr>
<td>765</td>
<td>.679</td>
<td>.675</td>
<td>4.32</td>
<td>1015</td>
<td>.603</td>
<td>.604</td>
<td>-0.65</td>
</tr>
<tr>
<td>775</td>
<td>.678</td>
<td>.681</td>
<td>-3.20</td>
<td>1025</td>
<td>.601</td>
<td>.603</td>
<td>-2.49</td>
</tr>
<tr>
<td>785</td>
<td>.683</td>
<td>.689</td>
<td>-6.49</td>
<td>1035</td>
<td>.603</td>
<td>.604</td>
<td>-7.4</td>
</tr>
<tr>
<td>795</td>
<td>.694</td>
<td>.700</td>
<td>-5.78</td>
<td>1045</td>
<td>.601</td>
<td>.604</td>
<td>-3.22</td>
</tr>
<tr>
<td>815</td>
<td>.710</td>
<td>.727</td>
<td>-17.25</td>
<td>1065</td>
<td>.601</td>
<td>.605</td>
<td>-4.19</td>
</tr>
<tr>
<td>825</td>
<td>.730</td>
<td>.745</td>
<td>-14.87</td>
<td>1075</td>
<td>.608</td>
<td>.605</td>
<td>2.66</td>
</tr>
<tr>
<td>835</td>
<td>.763</td>
<td>.765</td>
<td>-2.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least Square Error</td>
<td>.03489</td>
<td>.01305</td>
</tr>
<tr>
<td>Average Error</td>
<td>.02296</td>
<td>.00933</td>
</tr>
<tr>
<td>Maximum Error</td>
<td>.11716</td>
<td>.04264</td>
</tr>
<tr>
<td>Knots</td>
<td>Cubic Coefficients</td>
<td>Knots</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------</td>
<td>-------</td>
</tr>
<tr>
<td>595</td>
<td>C(0) = .63371</td>
<td>595</td>
</tr>
<tr>
<td></td>
<td>C(1) = .16475^-3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(2) = .19591^-3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(3) = .51758^-8</td>
<td></td>
</tr>
<tr>
<td>755.23</td>
<td>C(0) = .67678</td>
<td>835.32</td>
</tr>
<tr>
<td></td>
<td>C(1) = .16269^-3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(2) = -.19723^-3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(3) = .17472^-6</td>
<td></td>
</tr>
<tr>
<td>839.60</td>
<td>C(0) = .79122</td>
<td>876.56</td>
</tr>
<tr>
<td></td>
<td>C(1) = .35667^-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(2) = .42224^-4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(3) = .92733^-5</td>
<td></td>
</tr>
<tr>
<td>877.06</td>
<td>C(0) = .14612^-4</td>
<td>902.46</td>
</tr>
<tr>
<td></td>
<td>C(1) = .45759^-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(2) = .10844^-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(3) = -.78416^-4</td>
<td></td>
</tr>
<tr>
<td>896.20</td>
<td>C(0) = .21844^-1</td>
<td>910.47</td>
</tr>
<tr>
<td></td>
<td>C(1) = .10478^-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(2) = -.54197^-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(3) = .91502^-4</td>
<td></td>
</tr>
<tr>
<td>910.22</td>
<td>C(0) = .17793^-4</td>
<td>977.85</td>
</tr>
<tr>
<td></td>
<td>C(1) = -.60887^-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(2) = .42760^-3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(3) = -.13771^-5</td>
<td></td>
</tr>
<tr>
<td>1075.</td>
<td></td>
<td>1075.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-error</td>
<td>.09176</td>
<td>.05927</td>
<td>.03944</td>
<td>.03588</td>
<td>.03509</td>
<td>.03489</td>
</tr>
</tbody>
</table>

The algorithm required seven cycles through SWEEP for Case 2. The $L_2$-error decreased as follows:
These two cases required about 17 and 23 seconds, respectively, of execution time on a IBM 7094 for a FORTRAN IV version of this algorithm. They required about xxx and yyy seconds, respectively, of execution time on a CDC 6500 in Algol.

6 Other Nonlinear Algorithms Based on FIXEDKNOT

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

6.1 Non-systematic knot optimization

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

6.2 Systematic insertion of additional knots – $L_\infty$ criterion

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

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

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

6.3 Systematic insertion of additional knots – L2 criterion

Consider a process like 6.2 where we locate that interval between adjacent knots which has the most error in the L2 sense. We suspect that it is better to insert additional knots into this interval than near the location of the maximum error. We have not tested this suspicion, however.

6.4 Systematic insertion of knots with optimization

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

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


NONLINEAR SPLINE APPROXIMATION

PROGRAM WRITTEN BY CARL DE BOOR AND JOHN RICE

PURDUE UNIVERSITY

SUPPORTED BY THE NATIONAL SCIENCE FOUNDATION GP-4052,GP7163

PLEASE REPORT ANY CASES OF INOPERATION TO THE AUTHORS.

THANKS

**** NUMERICAL ANALYSIS CONTROL ****

CONTROL PARAMETERS     FUNCTION

ITER     NO. OF SWEEPS THRU OPT
BD (IN OPT)     IMPROVEMENT NEEDED TO REPEAT
EPSERR (IN SWEEP)     IMPROVEMENT NEEDED TO REPEAT
DIST (IN OPT NEAR 30,80)     KEEPS KNOTS SEPARATED
INDLP     NO. OF PASSES THRU OPT

THE FOLLOWING IS THE MAIN PROGRAM FOR VARYKNOT

DIMENSION INFO(16)

COMMON INPUT SERVES AS INPUT TO FXDKNT

SEE FXDKNT FOR DEFINITIONS OF VARIABLES
COMMON/ INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE

COMMON OUTPUT SERVES AS OUTPUT FROM FXDKNT

SEE FXDKNT FOR DEFINITIONS OF VARIABLES
COMMON/ OUTPUT /UERROR(100), FCTL(100),XIL(28),COEFL(27,4),

* VURDL(28,2),KNOT,LMAX,INTERV

COMMON OTHER SERVES AS COMMUNICATION BETWEEN OPT,SWEEP AND HERE

LXI = NUMBER OF INTERIOR KNOTS, LX1 = LXI+1, LX2 = LXI+2
Q     = NUMERICAL CONTROL VARIABLE USED BETWEEN OPT AND SWEEP
CHANGE     = DITTO
ERROR     = CURRENT VALUE OF THE L-2 ERROR - SQUARED
ACC     = DESIRED ACCURACY OF L-2 ERROR
XI(28)     = ARRAY FOR KNOTS

COMMON/ OTHER / LX1,LX2,LI2,Q, CHANGE,ERROR ,ACC, XI(28)

ACC = .1 AND ITER = 4 TO 8 SEEM TO BE GOOD VALUES FOR TYPICAL USES
ACC = .1

15
ITER = 8

C ***INFO IS SIMPLY AN IDENTIFICATION OF THE DATA***
1 READ(5,605) (INFO(I),I=1,16)
   605 FORMAT(16A5)
   WRITE(6,651) (INFO(I),I=1,16)
   651 FORMAT(1H1,20X,16A5 //)
C
C *** IF NOKNOT GE.1, THEN READ IN LXI2=NOKNOT KNOTS***
C *** OTHERWISE PROGRAM CHOOSES LXI2 = NOKNOT EQUISPACED KNOTS ***
C READ(6,610) NOKNOT, LX,
   (XX(I), U(I), I=1,LX)
C
C ***CHECK ON GIVEN DATA***
C THESE CHECKS PREVENT USER FROM EXCEEDING BOUNDS ON STORAGE
C AND FROM PRESENTING UNORDERED XX ARRAY
   IF(IABS(NOKNOT) .GE. 28 .OR.IABS(NOKNOT).LT. 3 ) GO TO 3
   IF( LX.LT.0 .OR. LX.GT. 100 ) GO TO 4
   WRITE(6,610) (I, XX(I), U(I), I=1,LX)
   WRITE(6,612) NOKNOT, ITER
   DO 2 L=2,LX
      IF(XX(L)-XX(L-1)) 6,6,2
      2 CONTINUE
      GO TO 14
   3 WRITE(6,660)
      GO TO 17
   4 WRITE(6,662)
      GO TO 17
   6 WRITE(6,664)
   7 WRITE(6,666)
      GO TO 1
C
C ***INITIALIZE***
14 IF( NOKNOT .LT. 0) GO TO 25
C
C *** READ IN LXI2 = NOKNOT KNOTS ***
LXI2 = NOKNOT
READ(5,601) (XI(J), J = 1,LXI2)
601 FORMAT(6F12.6)
GO TO 30

C C WHEN NOKNOT IS NEG., INTRODUCE -NOKNOT EQUISPACED KNOTS
25 LXI2 = -NOKNOT
   XI(1) = XX(1)
   XI(LXI2) = XX(LX)
   DEL = (XX(LX) - XX(1))/FLOAT(LXI2-1)
   DO 26 J = 3,LXI2
   26 XI(J-1) = XI(J-2) + DEL

C C SET UP INITIAL APPROXIMATION
30 ADDXI(1) = XI(1)
   ADDXI(2) = XI(LXI2)
   LXI1 = LXI2-1
   LXI = LXI1-1
   NODE = 0
   JADD = LXI2
   DO 35 J = 3,LXI2
   35 ADDXI(J) = XI(J-1)
   ERROR = FXDKWT(0)

C ***NOTE. NODE HAS BEEN SET EQUAL TO 1
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
   WRITE(6,900) (XI(I), I=1,LXI2)
900 FORMAT(28H KNOTS PRIOR TO OPTIMIZATION/(9F12.6))

C C OPTIMIZE KNOTS
   CALL SWEEP(ITER)

C WRITE(6,640)
640 FORMAT(49X,22H FINAL OUTPUT ***///)
   MODE = 1
   JADD = 0
   DUMB = FXDKWT(1)

C GO TO 1
C 610 FORMAT(214,       /(2F12.8))
611 FORMAT (11H GIVEN DATA/(I4,2F14.8))
612 FORMAT (1H /32H NO. OF INITIAL KNOTS =,I3/
                      7H ITER =,I3)
660 FORMAT(32H1KNOT CONTROL PARAMETER 'NOKNOT'/
                      1 19H NOT WITHIN BOUNDS 
                      1 28H NOT WITHIN BOUNDS 0 TO 100.)
662 FORMAT(24H1NO. OF DATA POINTS 'LX'/
                     1 20H IN ASCENDING ORDER.)
664 FORMAT(24H1DATA POINTS NOT READ IN/
                         1 20H IN ASCENDING ORDER.)
666 FORMAT(1H //43H CORRECT INDICATED INPUT ERROR AND RESTART.)
END
C
C******************************************************************************
C
C SUBROUTINE SWEEP(ITRR)
C
C KVARY+1 = INDEX OF KNOT BEING VARIED
C SUBROUTINE GPT(I) OPTIMIZES ITH INTERIOR KNOT
C
C COMMON /INPUT/LX,X(100),U(100),JADD,ADDX(26),MODE
C COMMON/ OUTPUT /ERROR(100),FCTRL(100),XIL(28),COEFL(27,4),
* VOROL(28,2),KNOT,LMAX,INTERV
C COMMON/ OTHER / LX1,LX1,LX1,Q,CHANGE,ERROR,ACC, XI(28)
C AT ALL TIMES, ERROR CONTAINS (L2 ERROR)**2 OF CURRENT B.A.
C
C ITER = ITRR
C **NEXT CARDS SET NUMERICAL ANALYSIS CONTROLS
C EPSERR = .ACC/2.5
C CHANGE = .4*FLOAT(LXI)
C
C 10 KVARY = LXI
C Q = CHANGE/FLOAT(LXI)
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE (6,902) ITER,Q
C902 FORMAT (6H ITER, Q IS,E20.8)
CHANGE = 0.
PREVER = ERROR
MODE = 2
JADD = 0
KNOT = KNOT - 1
DUMB = FXDKNT(0)
20 CONTINUE
*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE(6,900) KVARY
C*900 FORMAT(1H ///6H VARYING,I4,16TH INTERIOR KNOT)
C*** WRITE(6,901) ERROR
C*901 FORMAT(16H SQ. OF L2-ERROR ,E16.6)
C
CALL OPT(KVARY)
KVARY = KVARY -1
JADD = JADD + 1
IF (JADD .LE. 1) GO TO 22
K = JADD
DO 21 I = 2,JADD
K = K -1
21 ADDXI(K+1) = ADDXI(K)
22 ADDXI(1) = XI(KVARY + 2)
KNOT = LXI1 - JADD
MODE = 2
DUMB = FXDKNT(0)
IF (KVARY .NE. 0 ) GO TO 20

C THE LAST CALL TO FXDKNT PRODUCES THE B.A. USING ALL KNOTS
C SINCE THEN ADDXI CONTAINS ALL KNOTS
ERROR = DUMB
C*** THE FOLLOWING TWO CARDS PRODUCE PRINTED OUTPUT OF L1,L2,L-INF
C**  JADD = 0
C**  DUMB = FXDKNT(2)
C
C **IF CHANGE IN ERROR IS BIG ENOUGH MAKE ANOTHER SWEEP, ELSE QUIT
IF (ABS(PREVER-ERROR)/PREVER .LE. EPSERR) GO TO 60
ITER = ITER-1

19
C
C     **CHECK NUMBER OF PASSES THROUGH SWEEP
IF(ITER.EQ.0) GO TO 40
GO TO 10
40 CONTINUE
C
C     IN FINAL VERSION GO TO 40, GO TO 60 ARE REPLACED BY RETURN
C*** WRITE(6,620)
RETURN
60 CONTINUE
C*** WRITE(6,610)
RETURN
C*610 FORMAT(54H *** SWEEP DISCONTINUED - INSUFFICIENT CHANGE IN ERROR)
C*620 FORMAT (36H *** NO. OF ALLOWABLE SWEEPS USED UP)
END
C
C******************************************************************************
C
SUBROUTINE OPT(I)
C
I REFERS TO THE I TH INTERIOR KNOT
OPT FINDS THE OPTIMAL I TH KNOT BETWEEN THE I-1ST AND I+1ST KNOTS
THE REMAINING KNOTS ARE HELD FIXED.
INDLP = A BOUND ON THE NUMBER OF TRIES ALLOWED
       FOR IMPROVEMENT OF THE I TH KNOT
Q = MULTIPLICATION FACTOR WHICH SHOULD DECREASE AS A
   FUNCTION OF THE NO. OF SWEEPS THRU SWEEP
Q IS ALTERED IN SWEEP
C
COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
COMMON/ OUTPUT /ERROR(100),FCTL(100),XIL(28),COEFL(27,4),
       VORDL(28,2),KNOT,LMAX,INTERV
COMMON/ OTHER / LXI,LXI1,LXI2,Q,CHANGE,ERROR,ACC, XI(28)
C
I = II
20
**NUMERICAL ANALYSIS PARAMETERS SET HERE**

C

INDEP=9
RD = ACCERROR/FLOAT(LXI)
DIST = .0525
H = XI(I+2)-XI(I)
ALOW = XI(I) + DIST*H
AHIGH = XI(I+2) - DIST*H
LPCNT= 0
MODE = 3

C **BEGIN SEARCH - FIND THREE VALUES FOR THE ITH KNOT**

C SUCH THAT L2-ERROR AT MIDDLE VALUE, A , IS LESS THAN

C ERROR AT LEFT VALUE, ALEFT, AND AT RIGHT VALUE, ARIGHT

A = XI(I+1)
E = FXDXNT(A)
ALEFT = A + Q*(XI(I)-A)
ERIGHT = FXDXNT(A)

C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***

C**** ARIGHT = 0.
C**** ERIGHT = 0.
C**** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
SN = SIGN(1.,ELEFT-E)
IF (SN.GE.0) GO TO 20
GO TO 60

C

**SEARCHING FOR NEW KNOT TO THE RIGHT**

10 ALEFT = A
ELEFT = E
A = ARIGHT
E = ERIGHT

20 ARIGHT = A + Q*(XI(I+2)-A)

C

**BUFFER TO PREVENT COALESING OF KNOTS**

30 IF (AHIGH.GE.ARIGHT) GO TO 40
AA = AHIGH

C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***

C**** WRITE(6,610) I
GO TO 199

C 40 ERIGHT = FXDKNT(ARIGHT)
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C**** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
     IF (E.LE.ERIGHT) GO TO 100
C
C   **CHECK TO STOP OPT
     IF(E -ERIGHT.LE.RD .OR. LPCNT .GT. INDLP ) GO TO 240
     5 LPCNT = LPCNT+1
     IF(SGN.GT.0) GO TO 10
C
C   **SEARCHING FOR NEW KNOT TO THE LEFT
     60 ARIGHT = A
     ERIGHT = E
     A = ALEFT
     E = ELEFT
     70 ALEFT = A + Q*(XI(I)-A)
C
C
C   **BUFFER TO PREVENT COALESCING OF KNOTS
     80 IF (ALEFT.GE.ALOW) GO TO 90
     AA = ALOW
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C**** WRITE(6,620) I
     GO TO 199
C
C 90 ELEFT = FXDKWT(ALEFT)
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C**** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
     IF (E.LE.ELEFT) GO TO 100
C
C   **CHECK TO STOP OPT
     IF(E - ELEFT.LE.RD .OR. LPCNT .GT. INDLP ) GO TO 230
     GO TO 50
C
C   **REQUIRED 3 VALUES HAVE BEEN FOUND

22
C     FOLLOWING CODE FINDS PT. AT WHICH MIN OF PARABOLA CURVE PASSING
C     THRU THE ERROR VALUES AT THE PTS ALEFT, A, ARIGHT OCCURS
100  DLEFT = ALEFT - A
     DXRIGHT = ARIGHT - A
     DYLEFT = (ALEFT-E)/DLEFT
     DYRIGHT = (ARIGHT-E)/DXRIGHT
     DEL = .5/(DYLEFT-DYRIGHT)*(DXRIGHT*DYLEFT-DXLEFT*DYRIGHT)
     EPRED = F+DEL*(DYRIGHT+(DEL-DXRIGHT)/(ARIGHT-ALEFT)*(DYRIGHT-DLEFT))
     ABEST = A + DEL
     EBEST = FXDKNT(ABEST)
*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE (6,900) ELEFT,EBEST,ERIGHT,ALEFT,ABEST,ARIGHT
C
C     **DETERMINE WHETHER ABEST GIVES BEST APPRX AND MAKE APPROPRIATE
     SWITCHING OF THE AI'S DEPENDING ON SIGN OF DEL
     IF (EBEST.LE.E) GO TO 130
     IF(DEL)110,200,120
110  ALEFT = ABEST
     ELEFT = EBEST
     GO TO 170
120  ARIGHT = ABEST
     ERIGHT = EBEST
     GO TO 170
130  IF(DEL)140,200,150
140  ARIGHT = A
     ERIGHT = E
     GO TO 160
150  ALEFT = A
     ELEFT = E
160  A = ABEST
     E = EBEST
C
C     **FOLLOWING TESTS DETERMINE WHETHER OR NOT TO
C     REITERATE PARABOLA MINIMIZATION PHASE
170  IF (ABS(EPRED-EBEST).LT.5.*RD) GO TO 210
     IF(LPCNT.GT.INDLP) GO TO 200
     LPCNT = LPCNT+1

23
C
GO TO 100

190 ETRY = FXDKNT(AA)
     IF (E.LT.ETRY) GO TO 200
     A = AA
     E = ETRY
200 CHANGE = CHANGE + ABS(A - XI(I+1))/H
     XI(I+1) = A
     ERROR = E
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
RETURN
C
C IN FINAL VERSION GO TO 210, IS REPLACED BY GO TO 200
210 CONTINUE
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE(6,640) LPCNT
     GO TO 200
230 A = ALEFT
     E = ELEFT
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE(6,640) LPCNT
     GO TO 200
240 A = ARIGHT
     E = ERIGHT
C *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
C*** WRITE(6,640) LPCNT
     GO TO 200
C+610 FORMAT(46H *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (.I2,35H IS
C****MOVED TOO CLOSE TO RIGHT NEIGHBOR)
C+620 FORMAT(46H *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (.I2,34H IS
C****MOVED TOO CLOSE TO LEFT NEIGHBOR)
C+640 FORMAT(24H *** OPT DISCONTINUED AT,I4,31H - INSUFFICIENT CHANGE IN
C*** ERROR)
C+900 FORMAT(25H PARABOLA - ERROR VALUES ,3E20.6/12X,13HAI VALUES
C*** 1 3E20.6)
END
FUNCTION FXDKNT (ARG)

DOUBLE PRECISION TTRPZWT, SUM

LOGICAL MODE

DIMENSION WEIGHT(100), CURBERR(100)

COMMON / WANDT / TREND(100), TTRPZWT(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), KNRT, LMAX, INTERV

C UERROR(L) = ERROR OF BEST L2 APPROX 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 BAPPROX TO U AT XX(L)
C COEFL(I,.) CONTAINS THE POL. COEFF. 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

FCT (L,M) = BASIS FCT M AT XX(L)
VORD(M,K,L) CONTAINS THE ORDS (L=1) AND SLOPES (L=2) OF FCT M
AT THE KNOT INTRODUCED AS K-TH. CORRELATION TO ORDERING OF
KNOTS BY SIZE IS DONE VIA IORDER. I.E., ORD AND SLOPE AT
XIL(K) ARE IN VORD(M,IORDER(K),1).
BC(IJ) = COORDINATE OF U (AND OF B.A. TO U) WRTO I-JTH 0...M.FCT
ILAST = CURRENT NO. OF BASIS FCTNS

COMMON/ LASTS / IORDER(28), INSRT(30), XXKNOT
THE FCT ILAST (TO BE) INTRODUCED LAST HAS ADDITIONAL KNOT
XXKNOT, THE KNOT JUST INTRO-
DUCED HAS INDEX INSERT IN XIL, INSERT IS SAVED IN INSRT(ILAST)
FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).

***LOCAL VARIABLES

25
XSCALE = XX(LX) - XX(1), USED TO NORMALIZE INNER PRODUCT

LENGTH OF THE INTERVAL OF INTEGRATION

KNOTSV = NO. OF KNOTS USED IN MOST RECENT CALL TO FXDXNT

ERBUT1 = SQ. OF L2-ERROR OF APPROX USING ALL BUT THE ONE

KNOT BEING VARIED (USED IN MODE = 3)

CUBERB = UERROR OF B.A. BY CUBIC POL.-S (NEEDED FOR MODE = 2)

MODE3 = TRUE OR FALSE DEP. ON WHETHER PREV. CALL WAS IN

MODE=3 OR NOT

EQUIVALENCE (IPRINT,CHANGE)

ARG IS EITHER FIXED POINT (MODE.NE.3) TO PICK PRINT-OUT OPTION

OR IS FLOATING POINT (MODE=3) TO GIVE NEW VALUE OF KNOT VARIED

CALL CHANGE

IF (MODE.GT.0) GO TO 29

**** MODE=0* COMPUTE BASIS FCT 1 THROUGH 4 AND .A. TO U WRTO THESE

THEN SET MODE = 1 AND PUT UERROR INTO U.

XSCALE = XX(LX) - XX(1)

DO 10 I=5,30

10 INSRT(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))

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 CUBERR(L) = UERROR(L)
C
IF (JADD.LE.2), ONLY B.APPROX BY CUBICS IS COMPUTED
C
OTHERWISE, ADDXI(1), 1.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 WRT 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 ASSURE 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
UERROR IS ASSUMED TO CONTAIN ERROR OF B.A. TO U WRTO
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.
C
30 XKNOT = CHANGE
IF (MODE3)
MODE3 = .TRUE.
ERROR1 = FXDKNT
MODE = 2
CALL NUBAS
KNOTSV = KNOT
GO TO 35
MODE = 3
GO TO 36
35 CALL NUBAS
36 FXDKNT = ERBU1 - BC(ILAST)/XSCALE * BC(ILAST)
RETURN

C-----------------
C ***MODE=1,2*** RETAIN THE FIRST KNOT KNOTS INTRODUCED EARLIER
C (HENCE THEIR CORRESP FCTNS) BUT REPLACE FURTHER
C FCTNS (IF ANY) BY FCTNS HAVING ADDITIONAL
C KNOTS ADDXI(I),I=1,JADD, HENCE
C IF KNOT.LT.KNOTSV=-NO.OF KNOTS USED IN PREV CALL
C 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 NUBAS
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,LAST)
GO TO 49

42 DO 43 L=1,LX
43 UERROR(L) = CUBERR(L)
IF (KNOT.LE.2) GO TO 48
IDUM = KNOT + 1
DO 45 IO=IDUM,KNOTSV
INSERT = INSRT(ILAST
ILM3 = ILAST - 3
DO 44 K=INSERT,ILM3
IORDER(K) = IORDER(K+1)
44 XIL(X) = XIL(X+1)
45 ILAST = ILAST-1
DO 47 I=5,ILAST
DO 47 L=1,LX
47 UERROR(L) = UERROR(L) - BC(I) * FCT(I,L)
GO TO 49

48 XIL(2) = XIL(ILAST-2)
IORDER(2) = 2
KNOT = 2
49 IF (JADD.NE.0) GO TO 51

28
ILAST = KNOT + 2
INTERV = KNOT - 1

GO TO 60

C ***MODE=1,2***
C 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.
C
50 IF (JADD.LE.0)
51 DO 52 IO=1,JADD
52 XKNOT = ADDXI(IO)
      CALL NUBAS
      DO 52 LI=1,LX
      52 VERROR(L) = UERROR(L) - BC(ILAST)*FCT(L,ILAST)
C
60 FXDKN= DOT(31,2)/XSCALE
      KNOTSV = KNOT
61 MODE3 = .FALSE.
      RETURN
      IF (IPRINT.EQ.0)
      GO TO (70,80,90),IPRINT
C
C COMPUTE COEFFICIENTS OF BEST APPROX 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(1)
C IF(A) 2.4,4
C 2 I=I-1

29
C
  IF(I) 3,3,1
  3 I=1
C
70 WRITE(6,610)
    DO 72 I=1,KNOT
       ILOC = IORDR(I)
    DO 72 L=1,2
       SUM = 0.D0
    DO 71 J=1,ILAST
       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 CDEEFFICIENTS//)
620 FORMAT(35X,3HXI(F2.6)
630 FORMAT(67X,2HC(I1,3H)=,.16E6)
C
C **COMPUTE L2, L1, MAX ERRORS AND PRINT
80 ERRL2 = SQRT(FDXKNT)
  ERRL99= 0.
  DO 82 L=1,LX
     DIF = ABS(ERROR(L)*WEIGHT(L))
     IF(ERRL99.GT.DIF) GO TO 81
    LMAX = L
    ERRL99 = DIF
81 ERRL1 = ERRL1+ DIF
  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

30
90 IE = U
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) = UERRGR(L)*SCALE
GO TO (94,95,95).IPRINT
94 WRITE (6,621) IE,(1,XX(L),FCTL(L),PRINT(L),L=1,LX)
GO TO 96
95 WRITE (6,622) IE,(1,XX(L),PRINT(L),L=1,LX)
96 RETURN
621 FORMAT(1H //45X,36HAPPROXIMATION AND SCALED ERROR CURVE/36X,
*10HDATA POINT,7X,13HAPPROXIMATION,3X,16HDEVIATION X 1OE+,11/
 *(31X,I4,F16.8,F16.8,F17.6))
622 FORMAT(1H //56X, 11HERROR CURVE/38X, 10HDATA POINT, 23X,
116HDEVIATION X 1OE+,11/(31X,I4,F16.8,16X,F17.6))
623 FORMAT(1H //40X20HLEAST SQUARE ERROR =,.E20.6/
1 40X20HAVVERAGE ERROR =,.E20.6/
2 40X20HMAXIMUM ERROR =,.F20.6,3H AT,F12.6///;
END
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)
31
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
   NJ = NJ - 1
40   VORDL(NJ,2) = (VORDL(NJ,2) - D(NJ)*VORDL(NJ+1,2))/DIAG(NJ)
   RETURN

C*****************************************************************************
C
C FUNCTION DOT (M,INDEX)
C
C COMPUTE INNER PRODUCT OF FCT M WITH FCT ILAST (INDEX=1) OR
C UERROR (INDEX=2)
C
DOUBLE PRECISION DDOT,G,TRPZNT
COMMON / WANDT / TRPZNT(100),G(100)
COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),CSEFL(27,4),
  * VORDL(28,2),KNOT,LMAX,INTERV
COMMON/ BASIS /FCT(100,30),VRBD(30,28,2),SC(30),ILAST

10 IF (M.EQ.ILAST)   GO TO (10,30),INDEX
   DO 11 L=1,LX
10   G(L) = FCT(L,1)*FCTL(L)
11   GO TO 80

20   DO 21 L=1,LX
20   G(L) = FCT(L)*FCTL(L)
21   GO TO 80

30   IF (M.EQ.31)   GO TO 40
   DO 31 L=1,LX
30   G(L) = FCT(L)*UERROR(L)
31   GO TO 80

32
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

SUBROUTINE EVAL
C
COMPUTE POL. COEFF COEFL(I,K) OF PCT ILAST FROM VORDL,
C
THEN COMPUTE FCTL(L) = (PCT 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
ISWTC = 1
DO 20 L=1,X
GO TO (11,13),ISWTC
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 ISWITCH = 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)))) * TREN(L)
RETURN
END

C
C******************************************************************************
C
C SUBROUTINE MUBAS
DOUBLE PRECISION SUM
COMMON/INPUT/LX,XX(100),U(100),JADD,ADDDXI(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 /ORDER(28),INSIRT(30),KNOT
C COEFL(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 MODE=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 COEFL AND XI, HAS ADDITIONAL KNOT KNOT, 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),COEFL(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.

34
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 N,M=1,ILAST-1
C
10 KNUT = KNUT + 1
ILAST = KNUT + 2
ICLAST = ILAST*(ILAST-7)/2 + 10
ILM1 = ILAST-1
INTERV = KNUT - 1
DO 11 INSERT=2,INTERV
IF (XXNUT.LE.XIL(INSERT))  
GO TO 12
11 CONTINUE
12 IF (XXNUT.LE.XIL(INSERT-1))  
GO TO 95
13 ID = KNUT
DO 14 L=INSERT,INTERV
ID = ID - 1
XIL(ID+1) = XIL(ID)
14 IORDER(ID+1) = IORDER(ID)
IORDER(INSERT) = KNUT
C
15 XIL(INSERT) = XXNUT
DX = XXNUT - XIL(1)
DO 15 I=1,4
VORD(I,KNUT,1)=CDEF(I,1) + DX*(CDEF(I,2)+DX*(CDEF(I,3)
* + DX*CDEF(I,4)))
35
15 VORD(I,KNOT,2) = CDEF(I,2) + DX*(2.0*CDEF(I,3) + DX*3.0*CDEF(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 (XXN0T.IX(ID+1))
    GO TO 18
   ID = IX + 1
    GO TO 17
18 DX = XN0T - XX(I)
   VORD(I,KNOT,1) = CDEF(I,1) + DX*(CDEF(I,2) + DX*(CDEF(I,3)
      * DX*CDEF(I,4)))
   19 VORD(I,KNOT,2) = CDEF(I,2) + DX*(CDEF(I,3) + 2.0*DX*3.0*CDEF(I,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 XN0T ACROSS THE X-AXIS, THEN INTER
C   POLATING. THIS SHOULD INDUCE ONE MORE OSCILLATIO
C   N IN FCT ILAST THAN IN FCT 1-LAST-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 (INSERT(ILAST)=0)(SEE 9 ABOVE,
C

ALSO MAIN AT 10) SET MODE=1, PROCEED IN THAT MODE

C

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
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

*** MODE=3 *** CHANGE FCT ILAST BY CHANGING JUST THE KNOT INTRO
DUCE 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

*** INTERPOLATE
55 CALL INTERP                   GO TO (57,57,59),MODE
57 ID = ICLAST - 1
   DO 56 ID=1,INTERV
      ID = ID + 1
   END
56 XI(ID) = XIL(IO)
   INSRT(ILAST) = INSERT
C----------
C------------- ORTHONORMALIZE FCT ILAST OVER PREVIOUS (ORTHONORMAL) SET
C          THEN COMPUTE THE COMPONENT BC(ILAST) OF UERROR WRT3 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
69 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 CUEF(ID,L) = CUEFL(IO,L)/C
   DO 64 L=1,LX
64 FCT(L,ILAST) = FCTL(L)/C
C----------

RETURN

38
*** 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 CANNOT
*T BE CONTINUED)               STOP

C
END

C+++++++++++++++++++TREND AND WEIGHT FUNCTIONS+++++++++++++++++++++++
C
FUNCTION T(Z)
  T = 1.
  RETURN
END

FUNCTION W(Z)
  W = 1.
  RETURN
END

39