CALCULATION OF INTERPOLATING NATURAL SPLINE FUNCTIONS USING DE BOOR'S PACKAGE FOR CALCULATING WITH B-SPLINES

by

John G. Herriot

STAN-CS-76-569
OCTOBER 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Calculation of Interpolating Natural Spline Functions

Using de Boor's Package for Calculating with B-Splines

by

John G. Herriot

Computer Science Department
Stanford University
Stanford, California 94305

Abstract.

A FORTRAN subroutine is described for finding interpolating natural splines of odd degree for an arbitrary set of data points. The subroutine makes use of several of the subroutines in de Boor's package for calculating with B-splines. An ALGOL W translation of the interpolating natural spline subroutine and of the required subroutines of the de Boor package are also given. Timing tests and accuracy tests for the routines are described.

The work of this author was supported in part by the National Science Foundation under grant number MCS71-01996-A06.

Table of Contents

We now explain how a piecewise polynomial function can be expressed as a linear combination of B-splines. Let $\xi = (\xi_i)_i^{\ell+1}$ be a strictly increasing real sequence and let k be a positive integer. If $P_1,\ldots,P_\ell$ is any sequence of ℓ polynomials, each of order k (or, degree < k) then we define a corresponding piecewise polynomial f of order k by the prescription

$$f(t) = P_i(t) \quad \text{if } \xi_i < t < \xi_{i+1} ; \quad i = 1,2,\ldots,\ell .$$

We arbitrarily make f continuous from the right at the interior breakpoints, i.e.,

$$f(\xi_i) = f(\xi_i^+) \quad \text{for } i = 2,\ldots,\ell .$$

We denote the collection of all such piecewise polynomial. functions of order k with breakpoint sequence $\xi = (\xi_i)_{i=1}^{\ell+1}$ by

$$\mathbb{P}_{k,\xi} \quad .$$

Note that $\mathbb{P}_{k,\xi}$ is a linear space of dimension $k\ell$ since it is isomorphic to the direct product of ℓ copies of $\mathbb{P}_k$, the linear space of all polynomials of order k (degree < k) . A convenient way to represent a piecewise polynomial function $f \in \mathbb{P}_{k,\xi}$ is by

$$f(t) = \sum_{r=1}^{k} C_{r,i}(t - \xi_i)^{r-1} , \quad \xi_i \le t < \xi_{i+1} , \ i = 1,2,\ldots,\ell \qquad (2.7)$$

where $C_{r,i} = D^{r-1}f(\xi_i^+)/(r-1)!$, $r = 1,\ldots,k$; $i = 1,\ldots,\ell$. Then the j-th derivative of f at a point t is given by

$$D^jf(t) = \sum_{r=j+1}^{k} C_{r,i}(t - \xi_i)^{r-1-j}(r-1)!/(r-1-j)! \quad . \qquad (2.8)$$

We often wish to impose upon f the conditions that it have a certain number of continuous derivatives. We may write such conditions in the form

$$\text{jump}_{\xi_i} D^{j-1} f = 0 \ , \quad \text{for } j = 1,\ldots,\nu_i \ , \ i = 2, \ldots, \ell \tag{2.9}$$

for some vector $\nu = (\nu_i)_2^\ell$ with nonnegative integer entries. The subset of all $f \in \mathbb{P}_{k,\xi}$ satisfying (2.9) for a given ν is a linear subspace of $\mathbb{P}_{k,\xi}$ which we denote by $\mathbb{P}_{k,\underline{\xi},\underline{\nu}}$.

In order to obtain the B-spline representation of a piecewise polynomial function $f \in \mathbb{P}_{k,\xi,\nu}$ we need the following theorem which was proved by Curry and Schoenberg [5] and by de Boor [4].

<u>Theorem.</u> For a given strictly increasing $\xi = (\xi_i)_1^{\ell+1}$, and given nonnegative integer sequence $\nu = (\nu_i)_2^\ell$, with $\nu_i < k$, all i, set

$$n = k + \sum_{i=2}^{\ell} (k - \nu_i) = k\ell - \sum_{i=2}^{\ell} \nu_i = \dim \mathbb{P}_{k,\underline{\xi},\underline{\nu}} \tag{2.10}$$

and let $t = (t_i)_1^{n+k}$ be any non-decreasing sequence so that

(i) $t_1 \le t_2 \le \cdots \le t_k \le \xi_1$, $\xi_{\ell+1} \le t_{n+1} \le \cdots \le t_{n+k}$

(ii) for $i = 2,\ldots,\ell$, the number ξ_i occurs exactly $k-\nu_i$ times in t.

Then the sequence $N_{1,k},\ldots,N_{n,k}$ of B-splines of order k (or degree k-1) corresponding to the knot sequence $\underline{t}$ is a basis for $\mathbb{P}_{k,\underline{\xi},\underline{\nu}}$ considered as functions on $[t_k, t_{n+1}]$.

From this theorem we see that the B-spline representation for the piecewise polynomial function $f \in \mathbb{P}_{k,\underline{\xi},\underline{\nu}}$ is

$$f(t) = \sum_{r=i-k+1}^{i} a_r N_{r,k}(t) \ , \ \begin{cases} t_i \le t < t_{i+1} & \text{and } k < i < n \\ \text{or } \ t_i \le t \le t_{i+1} & \text{and } i = n \end{cases} \tag{2.11}$$

where $a = (a_i)_1^n$ are the coefficients of f with respect to the B-spline basis $(N_{i,k})_1^n$ for $\mathbb{P}_{k,\xi,\nu}$ on the knot sequence t . Then the j-th derivative of f at a point t is given by

$$D^j f(t) = \sum_{r=i-k+j+1}^{i} a_{r,j+1} N_{r,k-j}(t) \tag{2.12}$$

where

$$a_{r,j+1} = \begin{cases} a_r & , \quad j = 0 \\ (k-j)\dfrac{a_{rj} - a_{r-1,j}}{t_{r+k-j} - t_r} & , \quad j > 0 \end{cases} , \tag{2.13}$$

provided that either $t_i < t < t_{i+1}$ and $k \leq i < n$

or $t_i \leq t \leq t_{i+1}$ and $i = n$.

3. Representation of the Interpolating Spline.

Given a set of data points (x_i, y_i) , $i = N1, N1+1, \ldots, N2$ with $x_{N1} < x_{N2} < \ldots < x_{N2}$, we seek the interpolating natural spline function $S(x)$ of degree $2m-1$ with knots $x_{N1}, \ldots, x_{N2}$. For convenience in the FORTRAN implementation of the algorithm we shall assume throughout that $N1 = 1$. Then N2 is the number of data points. For our interpolating natural spline $S(x)$ we wish to make use of the B-spline representation given in equation (2.11) and the theorem on which it is based. We choose $k = 2m$, $\ell = N2-1$. Since $D^{j-1}S(x)$, $j = 1,2,\ldots,2m-1$, are continuous at all interior knots, we have $\nu_i = 2m-1$, all i , and we easily find that $n = N2 + 2m-2$. We choose

$$
\begin{aligned}
t_i &= x_1 , & i &= 1,2,\ldots,2m \\
t_{2m+i-1} &= x_i , & i &= 2,3,\ldots,N2-1 \qquad (3.1) \\
t_{N2+2m+i-2} &= x_{N2} , & i &= 1,2,\ldots,2m .
\end{aligned}
$$

The knot distribution is shown in Figure 1.

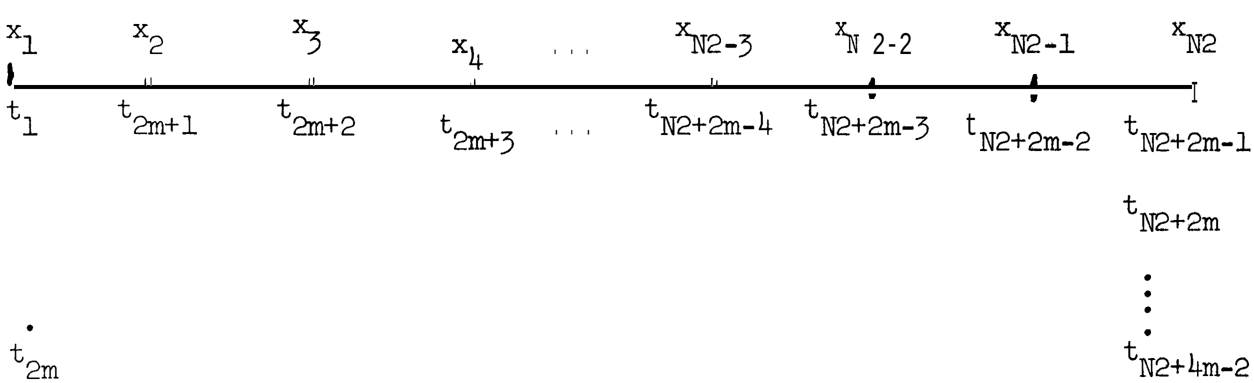

Figure1

From equation (2.11) we have the B-spline representation

$$S(t) = \sum_{r=i-2m+1}^{i} a_r N_{r,2m}(t) \begin{cases} t_i \leq t < t_{i+1} & \text{and } 2m < i < N2+2m-2 \\ \text{or} & \\ t_i \leq t \leq t_{i+1} & \text{and } i = N2+2m-2 \end{cases} \quad . \quad (3.2)$$

Now S(x) must satisfy the interpolating conditions

$$S(x_i) = y_i \quad , \qquad i = 1,2,\ldots,N2 \qquad (3.3)$$

and the natural spline end conditions

$$D^j S(x_1) = D^j S(x_{N2}) = 0 \ , \ j = m,m+1,\ldots,2m-2 \text{ if } m > 1 \ . \quad (3.4)$$

Substituting equation (3.2) into equations (3.3) and (3.4) we obtain the following set of equations for the determination of the a_r :

$$D^j S(x_1) = \sum_{r=1}^{2m} a_r D^j N_{r,2m}(x_1) = 0 \ , \ j = m,m+1,\ldots,2m-2 \qquad (3.5)$$

$$S(x_i) = \sum_{r=i}^{2m+i-1} a_r N_{r,2m}(x_i) = y_i \ , \ i = 1,2,\ldots,N2 \qquad (3.6)$$

$$D^j S(x_{N2}) = \sum_{r=N2-1}^{N2+2m-2} a_r D^j N_{r,2m}(x_{N2}) = 0 \quad , \quad j = m,m+1,\ldots,2m-2 \ . \qquad (3.7)$$

We now show that these equations lead to a (2m-1)-banded system of linear equations for the determination of the a_r . In Section 2 it was pointed out that $N_{r,2m}(t)$ is positive for $t_r < t < t_{r+2m}$ and zero otherwise. From equation (2.5) we conclude that at a knot t_j of multiplicity d_j , $D^s N_{r,2m}(t)$ is continuous for $s = 0,1,\ldots,2m-1-d_j$. In-particular, if $d_j = 2m-1$, then $N_{r,2m}(t)$ is continuous at t_j but none of its derivatives is continuous at t_j . If $d_j = 2m$, then even $N_{r,2m}(t)$ is discontinuous at t_j . For the coefficients in equations (3.6) we therefore conclude that

$$N_{1,2m}(x_1) \neq 0 \quad , \qquad N_{r,2m}(x_1) = 0 \ , \ r = 2,3,\ldots,2m$$

$$\left.\begin{array}{l} N_{2m+i-1,2m}(x_i) = 0 \\ \\ N_{r,2m}(x_i) \neq 0 \quad , \quad r = i,\ldots,2m+i-2 \end{array}\right\} \quad i = 2,3,\ldots,N2-1$$

$$N_{N2+2m-2,\,2m}(x_{N2}) \neq 0 \ , \quad N_{r,2m}(x_{N2}) = 0 \ , \ r = N2,\ldots,N2+2m-3 \quad .$$

For the coefficients in equation (3.5) we find that

$$\left.\begin{array}{llll} D^jN_{r,2m}(x_1) & \neq 0 & , & r = 1,2,\ldots,j+1 \\ \\ & = 0 & , & r = j+2,\ldots,2m \end{array}\right\} \quad j = m,m+1,\ldots,2m-2$$

and for the coefficients in (3.7)

$$\left.\begin{array}{llll} D^jN_{r,2m}(x_{N2}) & = 0 & , & r = N2-1,\ldots,N2+2m-3-j \\ \\ & \neq 0 & , & r = N2+2m-2-j,\ldots,N2+2m-2 \end{array}\right\} \quad j = m,\ldots,2m-2 \quad .$$

If we denote the non-zero coefficients of the system of equations given by (3.5), (3.6), (3.7) by x , then the coefficient matrix has the form:

		1	2	3	...	m	m+1	m+2	...	2m-2	2m-1	2m	2m+1	...	N2-2	N2-1	N2	N2+1	...	N2+m-3	N2+m-2	N2+m-1	...	N2+2m-4	N2+2m-3	N2+2m-2
(3.5)	m	x	x	x	...	x	x																			
	m+1	x	x	x	...	x	x	x																		
	⋮	⋮																								
	2m-3	x	x	x	...	x	x	x	...	x																
	2m-2	x	x	x	...	x	x	x	...	x	x															
(3.6)	1	x																								
	2		x	x	...	x	x	x	...	x	x	x														
	3			x	...	x	x	x	...	x	x	x	x													
	⋮																									
	N2-2														x	x	x	x	. . .	x	x	x	. . .	x		
	N2-1															x	x	x	.	x						
	N2																									x
(3.7)	2m-2																x	x	. . .	x	x	x	. . .	x	x	x
	2m-3																	x	.	x	x	x				x
	⋮																									
	m+1																			x	x	x	. . .	x	x	x
	m																				x	x	. . .	x	x	x

We have boxed the coefficients of a_1 and $a_{N2+2m\ 2}$ to emphasize the fact that these two coefficients can be calculated at once and eliminated from the remaining equations yielding a banded matrix with m-l subdiagonals and m-l superdiagonals. We use the first and last equations of (3.6) to obtain

$$a_1 = y_1/N_{1,2m}(x_1)$$

$$a_{N2+2m-2} = y_{N2}/N_{N2+2m-2,\ 2m}(x_{N2}) \ .$$

Then from (3.5) and (3.7) we have

$$\sum_{r=2}^{j+1} A_r \ D^j N_{r,2m}(x_1) = -a_1 D^j N_{1,2m}(x_1) \qquad j = m, m+1, \ldots, 2m-2 \tag{3.8}$$

and

$$\sum_{r=N2+2m-2-j}^{N2+2m-3} A_r \ D^j \ N_{r,\ 2m}(x_{N2}) = -a_{N2+2m-2} \ D^j \ N_{N2+2m-2}(x_{N2}) \qquad j = 2m-2, \ldots, m \ . \tag{3.9}$$

The remaining equations are given by (3.6) with the first and last equations omitted. Now we have a banded system; the unknowns are $a_2, a_3, \bullet$ **yaN2+2m-3 which we rename $z_1, z_2, \ldots, z_{N2+2m-4}$. We note that the diagonal elements of the system matrix are in order

$$D^m N_{2,2m}(x_1) \ , \ D^{m+1} N_{3,2m}(x_1) \ , \ldots, \ D^{2m-2} N_{m,2m}(x_1) \ ,$$

$$N_{m+1,2m}(x_2) \ , \ N_{m+2,2m}(x_3) \ , \ldots, \ N_{N2+m-2,\ 2m}(x_{N2-1}) \ ,$$

$$D^{2m-2} N_{N2+m-1,2m}(x_{N2}) \ , \ D^{b-3} N_{N2+m,\ 2m}(x_{N2}) \ , \ldots, \ D^m N_{N2+2m-3,2m}(x_{N2}) \ .$$

The subdiagonal and superdiagonal elements are also values and derivatives of B-splines evaluated at the knots.

All the elements of this matrix are calculated by using the subroutines of de Boor's B-spline package [4]. The elements of the matrix are stored in diagonal form for use of the band matrix solver subroutines BANDET and BANDSL which are essentially FORTRAN implementations of the corresponding ALGOL 60 procedures given by Martin and Wilkinson [10]. The diagonal elements are stored as $Q(i,m)$, the subdiagonal elements as $Q(i,j)$, $j = 1,2,\ldots,m-1$, the superdiagonal elements as $Q(i,j)$, $j = m+1,m+2,\ldots,2m-1$, where $i = 1,2,\ldots,N2+2m-4$.

The solution of this system of equations yields the coefficients a_r of the B-spline representation (3.2) for the interpolating natural spline $S(t)$. The values of $S(t)$ and its derivatives can be evaluated at any point by means of subroutines in the de Boor package. In particular we can obtain the piecewise polynomial representation (2.7) (or (1.1)) of $S(x)$ by evaluating the function and the derivatives at the breakpoints.

4. The FORTRAN Subroutine.

Before describing the FORTRAN subroutine NATSPP for the interpolating natural spline we first describe briefly those subroutines of the de Boor package [4] which are used in the subroutine NATSPP.

We begin with a summary of the FORTRAN variables and their intended use and a terse summary of the subprograms and their intended use,

The B-spline representation consists of

T(1),...,T(N+K) , the knot sequence, assumed nondecreasing; if t appears j times in this sequence, then the (K-j)-th derivative may jump at t.

A(1), ..., A(N) , B-spline coefficients for the function represented on (T(K), T(N+1)) .

N , the number of B-splines of order K for the given knot sequence.

K , order (= degree +1) of the B-splines; should be < 20 .

The piecewise-polynomial representation consists of

XI(1),...,XI(LXI+1) , the breakpoint sequence, assumed increasing.

C(1,1),...,C(K,LXI) , values of derivatives at breakpoints; precisely C(J,I) is (J-1)-st derivative at XI(I)+ , J =1,...,K . Note that the coefficients in (2.7) and (1.1) are these derivatives divided by (J-1): .

K , order (= degree +1) of polynomial pieces; should be $\leq$ 20 .

Other variables are defined in the subroutine summary which follows:

subroutine BSPLDR(T,A,N,K,ADIF,NDERIV)

Constructs divided difference table for B-spline coefficients preparatory to derivative calculation and stores it in ADIF(1,1),...,ADIF(N,NDERIV) . Expects NDERIV in the interval [2,K] . Used only in BSPLPP, prior to call of BSPLEV.

subroutine BSPLEV(T,ADIF,N,K,X,SVALUE,NDERIV)

Calculates value of spline and its derivatives at X from B-spline representation and returns them in SVALUE(1), . ..' SVALUE(NDERIV) . Can use A for ADIF if NDERIV = 1 . Otherwise must have ADIF filled beforehand by BSPLDR. Uses INTERV and BSPLVN. Used only in BSPLPP.

subroutine BSPLPP(T,A,N,K,SCRTCH,XI,C,LXI)

Converts B-spline representation to piecewise-polynomial representation. SCRTCH is temporary storage of size (N,K) . Uses BSPLDR and BSPLEV. Used in NATSPP, the subroutine for natural spline interpolation.

subroutine BSPLVN(T,JHIGH,INDEX,X,ILEFT,VNIKX)

Calculates value of all possibly'nonzero B-splines at X of order J = max{JHIGH,(J+1)*(INDEX-1)} on T . ILEFT is input, assumed so that T(ILEFT) < T(ILEFT+1) ; get division by zero otherwise. If T(ILEFT) $\leq$ X $\leq$ T(ILEFT+1) (as would be expected) then VNIKX(I) is filled with B-spline value N(ILEFT-J+I,J) at X , I = 1,...,J . Get limit from right or left, if X = T(ILEFT) or T(ILEFT+1) respectively. Can save time by using INDEX = 2 in case this call's desired order J is greater than the previous call's order (saved in J) provided T , X , ILEFT and VNIKX are unchanged between the calls. Otherwise, use INDEX = 1 . Used in BSPLEV, BSPLVD and NATSPP.

subroutine BSPLVD(T,K,X,ILEFT,VNIKX,NDERIV)

Calculates value and derivatives of order < NDERIV of all B-splines which do not vanish at X . ILEFT is input, assumed so that T(ILEFT) < T(ILEFT+1) ; get division by zero otherwise. If T(ILEFT) $\le$ X $\le$ T(ILEFT+1) (as would be expected) then VNIKX(I,J) is filled with value of (J-1)-st derivative of N(ILEFT-K+I,K) at X , I = 1,...,K , J = 1,...,NDERIV . Get derivative from right or left if X = T(ILEFT) or T(ILEFT+1) , respectively. Expects NDERIV in [1,K] . Uses BSPLVN. Used in NATSPP.

subroutine INTERV(XT,LXT,X,ILEFT,MFLAG)

Computes largest ILEFT in [1,LXT] such that XT(ILEFT) $\le$ X . It is assumed that XT is a one-dimensional array of length LXT containing a nondecreasing sequence of real numbers. The subroutine returns integers ILEFT and MFLAG as follows:

$$\text{if} \left\{ \begin{array}{r} X < XT(1) \\ XT(I) \le X < XT(I+1) \\ XT(LXT) \le X \phantom{< XT(I+1)} \end{array} \right\}, \ \text{then} \left\{ \begin{array}{ll} \text{ILEFT} & \text{MFLAG} \\ 1 & -1 \\ I & 0 \\ LXT & 1 \end{array} \right.$$

The value of ILEFT is saved in a local variable IL0 which under certain conditions is used to start the search for ILEFT in the next call. The local variable IL0 is initialized to the value one.

Note that only BSPLPP, BSPLVN and BSPLVD are called directly by the natural spline interpolation subroutine NATSPP. In addition to these subroutines of the de Boor package, NATSPP also calls subroutines BANDET and BANDSL for the solution of the linear system CX = B where C is an unsymmetric band matrix. These subroutines were taken from the library of the Stanford Center for Information Processing. They are translations of ALGOL 60 procedures given by Martin and Wilkinson [LO]. They are fully described in the complete listing of the FORTRAN subroutines in Appendix I.

Turning now to the subroutine NATSPP for the interpolating natural spline we note that it is a direct implementation of the method described in Section 3. First we give a summary of the FORTRAN variables and their intended use. The heading of the subroutine is

SUBROUTINE NATSPP(N2,N3,N4,M,M2,MM,X,Y,A,C,T,Q,TRL,INT,VNIKX) .

The input parameters are as follows:

N2 , the number of data points.

N4 , = N2+M2-4 .

M , 2*M-1 is the degree of the natural spline

admissible values range from 1 to N2 .

M2 , = 2*M , the order of the natural spline.

MM , = 2*M-1 , the degree of the natural spline.

X(1), ...,X(N2) , abscissas of the data points which must be strictly monotone increasing.

Y(1),...,Y(N2) , ordinates of the data points.

The output parameters are as follows:

N3 , = N2-1+MM , the number of B-splines in the B-spline representation (1.2).

A(1),...,A(N3) , the coefficients of the B-spline representation (1.2) of the natural spline.

C(1,1),...,C(M2,N2-1) , the coefficients of the piecewise polynomial representation (1.1) of the natural spline.

The remainder of the parameters are only for temporary storage. They are included in the declaration in order to make it possible to give them variable dimensions. They are:

T(1),...,T(N2+4*M-2) , the knot sequence.

Q(1,1), ...,Q(N4,M2) , elements of the band matrix of the equations for the calculation of the A(1) .

TRL(1,1),...,TRL(N4,M-1) , matrix for storing lower triangular matrix of the LU decomposition of the band matrix.

INT(1),...,INT(N4) , vector for recording row interchanges during decomposition of the band matrix.

VNIKX(1,1),...,VNIKX(M2,MM) , matrix for storing values and derivatives of B-splines as needed.

The subroutine NATSPP begins by computing the knot sequence T(1) from the abscissas of the data points. In order to get the coefficients of the first M-l rows of the band matrix which are given by (3.8) we call

BSPLVD(T,M2,X(1),M2,VNIKX,MM)

to obtain VNIKX(I,J) = $D^{J-1}N_{I,}$,(X(l)) , I = 1,...,M2 , J = 1,...,MM .
We use these to calculate A(1) and the coefficients of the first M-l rows and their right members. For the coefficients of the last M-l rows of the band matrix which are given by (3.9) we call

BSPLVD(T,M2,X(N2),N3,VNIKX,MM)

to obtain VNIKX(I,J) = $D^{J-1}N_{N2-2+I,M2}$(X(N2)) , I = 1,...,M2 , J =1,...,MM . We use these to calculate A(N2+2*M-2) and the coefficients of the last M-l rows and their right members. For the coefficients of the rest of the rows of the band matrix which are given by (3.6) (omitting first and last equations) we call

BSPLVN(T,M2,1,X(I),I+MM,VNIKX)

to obtain VNIKX(J,1) = $N_{I-1+J,M2}$(X(I)) , J = 1,...,M2 , I = 2,...,N2-1 .

The band matrix system is then solved using BANDET and BANDSL to obtain the coefficients A(1) . Finally we call

BSPLPP(T,A,N3,M2,Q,X,C,LXI)

to calculate the derivatives needed to produce the coefficients of the piecewise polynomial representation. Note that in BSPLPP, C(J,I) has the value $D^{J-1}S(X(I)^{+})$ whereas in NATSPP, C(J,I) has the value $D^{J-1}S(X(I)^{+})/(J-1)!$.

The complete listing of NATSPP with all embedded subroutines is given in Appendix I.

5. The ALGOL W Procedure.

Since we have available ALGOL W versions of the procedure NATSPLINE of Algorithm 472 of Herriot and Reinsch [8] and of Algorithm 480 of Lyche and Schumaker [9], it would be much easier to make comparison tests with the algorithm using the de Boor package [4] to calculate the interpolating natural spline if it were implemented in ALGOL W. The FORTRAN subroutine NATSPP was therefore translated into an ALGOL W procedure DEBNAT.

First the subroutines of the de Boor package used in NATSPP were translated into ALGOL W procedures with the same names and the same parameters. Special care was needed to deal with two unusual features of the FORTRAN package. In order to save the value of the local variable IL0 of INTERV and of the local variable J in BSPLVN from one call to the next, these variables were made global to all the procedures of the de Boor package (J was renamed JJ). For the same reason the arrays DELTAM and DELTAP used in BSPLVN were made global. These global quantities were initialized prior to any calls of the package procedures. The other unusual feature of the FORTRAN subroutines was use of VNIKX as a one-dimensional array in BSPLVN and as a two-dimensional array in BSPLVD. This was handled by making VNIKX a two-dimensional array in BSPLVD and introducing a corresponding one-dimensional array NVNIKX local to BSPLVD,

The ALGOL W procedures BANDET and BANSOL are completely similar to the corresponding FORTRAN subroutines. They are fully described in the complete listing of the ALGOL W procedures in Appendix II.

Because of the greater flexibility of ALGOL W in using dynamic array declarations, it was possible to reduce the number of formal parameters in the procedure DEBNAT compared to those in NATSPP. The heading of the procedure is

```
PROCEDURE DEBNAT(INTEGER VALUE N1,N2,M; REAL ARRAY X(*);
                 REAL ARRAY A(*,*); REAL ARRAY CFF(*));
```

The input parameters are as follows:

N1,N2 , subscript of first and last data point.

M , 2*M-1 is the degree of the natural spline
admissible values range from 1 to N2-N1+1 .

X(N1::N2) , contains the given abscissas X(1) which must be strictly monotone increasing.

A(N1::N2,0::2*M-1) , contains the given ordinates as zero-th column, i.e., A(I, 0) represents Y(I) .

The output parameters are as follows:

A(N1::N2,0::2*M-1) , the ciefficients of the piecewise polynomial representation (1.1) of the natural spline with $c_{j,i}$ = A(i,j-1) . (A(N2,0) is unchanged and no values are assigned to the last row of A .)

CFF(1::N2-N1+2*M-1) , the coefficients of the B-spline representation (1.2) of the natural spline.

The ALGOL W procedure DEBNAT is an exact translation of the FORTRAN subroutine NATSPP. The complete listing of DEBNAT with all embedded procedures is given in Appendix II.

Included among the tests described in Section 6 were tests to verify that NATSPP and DEBNAT produced the same results.

Another advantage of the ALGOL W procedure is that it is much easier to convert it to double precision arithmetic than it is to convert the FORTRAN subroutine NATSPP and the de Boor FORTRAN package to double precision arithmetic.

6. Tests.

Both the FORTRAN subroutine NATSPP and the ALGOL W procedure DEBNAT were tested extensively on the IBM 370/168 at the Stanford Center for Information Processing.

In order to verify that the routines were operating correctly for the evaluation of the polynomial coefficients of the spline $S(x)$, the values of $D^j S(x)/j!$, $j = 0,1,\ldots,2m-2$ were calculated at the right-hand endpoint of each subinterval $[x_i, x_{i+1})$ and compared with their values (the coefficients in equation (1.1)) at the left-hand endpoint of the next subinterval. For the first test we used the five data points $(-3,7)$, $(-1,11)$, $(0,26)$, $(3,56)$, $(4,29)$ with nonequidistant abscissas. Table I shows the results of a typical run using the FORTRAN subroutine NATSPP with $m = 2$ for these data points. The first line of each box gives the tabulated quantities at the given value of x which is the left-hand endpoint of the subinterval, and the second line of the box gives the tabulated quantities at the right-hand endpoint of the same subinterval. Similar results were obtained for $m = 1,3,4,5$ for the same data points. The close agreement of these quantities $D^j S(x)/j!$, $j = 0,1,\ldots,2m-2$ to the left and right of each breakpoint shows that the spline function and its derivatives satisfy the specified continuity conditions. This is a good indication of the correctness of the results. Note that in Table I $S''(-3)$ differs very slightly from its specified value of 0 and that $S(0)$ and $S(3)$ differ from their prescribed values in the least significant digit. Exactly the same results were obtained using the ALGOL W procedure DEBNAT. These results are very close to those obtained by using NATSPLINE which are given in Table I of Algorithm 472[8].

x	S(x)	S'(x)	S''(x)/2	S'''(x)/3!
-3.000000	7.000000 10.999999	-1.999995 9.999986	$-.4291534 \times 10^{-5}$ 5. [illegible]	1.000000 1.000000
-1.000000	11.000000 25. [illegible]	9.999997 18.99998	6.000000 2. [illegible]	-1.000001 -1.000001
0	25.99995 55. [illegible]	18.99998 [illegible]	2.999995 -14. [illegible]	-1.999998 -1.999998
3.000000	55.99998 29.00000	-16.99998 -32.00000	[illegible] $-.3242493 \times 10^{-4}$	4.999987 4.999987
4.000000	29.00000			

Table I. Cubic Natural Spline.
Five nonequidistant knots. Coefficients
calculated by NATSPP.

The same test was run for ten data points (x_i, y_i) with equidistant abscissas $x_i = i$ and ordinates given by

$$y_i = \begin{Bmatrix} 1 & , & i \text{ odd} \\ 0 & , & i \text{ even} \end{Bmatrix} \quad i = 1,2,\ldots,10 \ . \tag{4.1}$$

For $m = 1,\ldots,5$ the FORTRAN subroutine NATSPP and the ALGOL W procedure DEBNAT gave the same results. In all cases the specified continuity conditions at the breakpoints were satisfied.

The previous tests established that the FORTRAN subroutine NATSPP and the ALGOL W procedure DEBNAT produced identical results. Further tests for accuracy and timing were carried out using only the ALGOL W procedure DEBNAT. Corresponding results for the accuracy of NATSPP can be inferred from these tests.

As a check on the correctness of the piecewise polynomial coefficients, long precision versions of DEBNAT and NATSPLINE from Algorithm 472[8] were used to calculate the polynomial coefficients for the data points $(-3,7)$, $(-1,11)$, $(0,26)$, $(3,56)$, $(4,29)$ (data for Table I) for $m = 1,2,\ldots,5$. When rounded to short precision, the corresponding coefficients calculated by the two procedures were identical, (Except that for $D^j S(1)/j!$, $j = m,\ldots,2m-2$, NATSPLINE gave the specified values 0 and DEBNAT gave values of order 10^{-t} , $t > 9$.) The same comparison test was run for the set of N2 data points (x_i, y_i) with equidistant abscissas $x_i = i$ and ordinates given by

$$y_i = \begin{Bmatrix} 1 & , & i \text{ odd} \\ 0 & , & i \text{ even} \end{Bmatrix} \quad i = 1,\ldots,N2 \ . \tag{6.2}$$

Values of $N2 = 10,20,\ldots,50$ and $m = 1,2,\ldots,7$ were used.' Again when the long precision coefficients were rounded to short precision, the corresponding coefficients calculated by the two procedures were identical (with the same exceptions as above and occasional differences in the least significant digit for the case $m = 7$).

In order to study the effect of round-off error build-up, both long and short precision versions of DEBNAT were used to calculate the B-spline coefficients and the piecewise polynomial coefficients. From the previous comparison tests we see that we can regard the long precision coefficients to be correct and hence the differences between the long precision and short precision coefficients are the errors in the latter. This error test was first run for the data points of Table I for $m = 1,2,\ldots,5$. For $m = 1$ all short precision coefficients calculated by DEBNAT are exactly correct. For $m = 2$, the maximum errors in the B-spline coefficients and in the piecewise polynomial coefficients were of order approximately 10^{-5} . For $m = 3$ and 4 the maximum errors increased to about 10^{-3} . For $m = 5$ the errors exceeded one and the results were unacceptable. This may be due in part to the fact that for five points, $m = 5$ is an extreme case. Tests were also run for the example with equidistant knots and ordinates given by (6.2). Values of $N2 = 10,20,\ldots,50$ and $m = 1,2,\ldots,7$ were used. The results for $m = 1,2,3,4$ were similar to those for the data of Table I. For $m = 5$ the maximum errors were of order 10^{-2} . For $m = 6,7$ the maximum errors exceeded one.

-Long precision and short precision versions of NATSPLINE were used on the same data to find the errors in the piecewise polynomial coefficients calculated by short precision NATSPLINE. The results appeared to be somewhat better than for DEBNAT. For the data used in Table I, the maximum errors in the piecewise polynomial coefficients were 0 for $m = 1$ and of order 10^{-5} for $m = 2,\ldots,5$. For the example with equidistant knots and ordinates given by (6.2), the maximum errors in the piecewise polynomial coefficients were 0 for $m = 1$, of order 10^{-5} for $m = 2,3$, of order 10^{-4} for $m = 4$, of order 10^{-2} for $m = 5$, and of order 10^{-1} for $m = 6,7$.

We conclude that DEBNAT should not be used in short precision for $m > 5$ and NATSPLINE should not be used in short precision for m i 7 .

In addition to the tests for accuracy, timing tests were carried out for long precision and short precision versions of both DEBNAT and NATSPLINE on the IBM 370/168 computer at the Stanford Center for Information Processing. The tests were made using the example with equidistant knots and ordinates given by (6.2). Values of $N2 = 10,20,\ldots,100$ and $m = 1,2,\ldots,7$ were used. The time for both procedures was found to be approximately proportional to the number N2 of knots. For DEBNAT the time was found to be approximately proportional to $m^{1.7}$ for $m \geq 3$ while for NATSPLINE it was approximately proportional to m^2. The actual times were almost exactly the same for the short precision and long precision versions.

The time T in seconds for the execution of the procedure DEBNAT was found to be approximately

$$T = (N/60)(.0265m^{1.7}) , \quad m \geq 3 .$$

This formula seriously underestimates the time for $m = 1$ and 2 .

For NATSPLINE the time was found to be approximately

$$T = (N/60)(.015m^2) .$$

For $m < 5$ the times for NATSPLINE were somewhat less than those for DEBNAT, but for $m \geq 6$ the times were nearly the same.

Since we found that for the IBM 370/168 the times for short precision and long precision are nearly the same, we recommend the use of long precision for all calculations using these procedures. Converting the given ALGOL W procedures to long precision requires only replacement of all real identifiers by long real identifiers. The same recommendation would apply to any machine on which long precision is approximately as fast as short precision.

For reasons of accuracy we would also recommend the use of double precision for the FORTRAN subroutines. We have not attempted to convert the FORTRAN de Boor package to double precision.

References

1. Anselone, P. M. and Laurent, P. J., "A general method for the construction of interpolating and smoothing spline functions," Numer. Math. 12 (1968), 66-82.
2. de Boor, Carl, "On uniform approximation by splines," J. Approximation Theory 1 (1968), 219-235.
3. de Boor, Carl, "On calculating with B-splines," J. Approximation Theory 6 (1972), 50-62.
4. de Boor, Carl, "Package for calculating with B-splines," MRC Technical Summary Report #1333, Mathematics Research Center, The University of Wisconsin -Madison, Oct. 1973.
5. Curry, H. B. and Schoenberg, I. J., "On Pólya frequency functions. IV. The fundamental spline functions and their limits," J. Analyse Math. 17 (1966), 71-107.
6. Greville, T. N. E., "Spline functions, interpolation and numerical quadrature." In Mathematical Methods for Digital Computers, Vol. II. A. Ralston and H. S. Wilf (Eds.), Wiley, New York, 1967.
7. Greville, T. N. E., "Introduction to spline functions." In Theory and Applications of Spline Functions. T. N. E. Greville (Ed.), Academic Press, New York, 1969, pp. l-35. (Pub. No. 22 Mathematics Research Center, U.S. Army, U. of Wisconsin.)
8. Herriot, John G. and Reinsch, Christian H., "Algorithm 472. Procedures for natural spline interpolation," Comm. ACM 16 (1973), 763-768.
9. Lyche, Tom and Schumaker, Larry L., "Algorithm 480. Procedures for computing smoothing and interpolating natural splines," Comm. ACM 17 (1974), 463-467.
10. Martin, R. S., and Wilkinson, J. H., "Solution of symmetric and unsymmetric band equations and the calculation of eigenvectors of band matrices," Numer. Math. 9 (1967), 279-301. Also in J. H. Wilkinson and C. Reinsch, Linear Algebra, Handbook for Automatic Computation, vol. II, F. L. Bauer (chief Ed.), Band 186 in die Grundlehren der matematischen Wissenschaften, Springer-Verlag, New York, Heidelberg, Berlin. 1971, pp. 70-92.

APPENDIX I

```
      SUBROUTINE NATSPP(N2,N3,N4,M,M2,MM,X,Y,A,C,T,Q,TRL,INT,VNIKX)
C     NATSPP COMPUTES THE COEFFICIENTS OF BOTH THE PIECEWISE
C     POLYNOMIAL REPRESENTATION AND THE B-SPLINE REPRESENTATION OF A
C     NATURAL SPLINE S(X) OF DEGREE(2*M-1),INTERPOLATING THE
C     ORDINATES Y(I) AT POINTS X(I),I=1 THROUGH N2.
C          PIECEWISE POLYNOMIAL REPRESENTATION:
C     FOR XX IN(X(I),X(I+1)),I=1,...,N2-1,
C     S(XX)=C(1,I)+C(2,I)*T+...+C(2*M,I)*T**(2*M-1)
C     WITH T=XX-X(I).
C          B-SPLINE REPRESENTATION:
C     FOR XX IN (X(1),X(N2)),
C     S(XX)=A(1)*N(1,2*M,XX)+A(2)*N(2,2*M,XX)+...
C              +A(N2+2*M-2)*N(N2+2*M-2,2*M,XX)
C     WHERE N(J,2*M,XX) IS THE (NORMALIZED) B-SPLINE OF DEGREE
C     (2*M-1) ON THE KNOT SEQUENCE T(J),...,T(J+2*M).
C     INPUT:
C        N2       THE NUMBER OF DATA POINTS
C        N4       =N2+M2-4
C        M        2*M-1 IS THE DEGREE OF THE NATURAL SPLINE,
C                 ADMISSIBLE VALUES RANGE FROM 1 TO N2,
C                 RECOMMENDED VALUES ARE NOT GREATER THAN 5 (SAY)
C        M2       =2*M, THE ORDER OF THE NATURAL SPLINE
C        MM       =2*M-1, THE DEGREE OF THE NATURAL SPLINE
C        X(1),...,X(N2)  ABSCISSAS OF THE DATA POINTS WHICH
C                 MUST BE STRICTLY MONOTONE INCREASING
C        Y(1),...,Y(N2)  ORDINATES OF THE DATA POINTS
C     OUTPUT:
C        N3       =N2-1+MM, THE NUMBER OF B-SPLINES IN THE
C                 B-SPLINE REPRESENTATION OF THE NATURAL SPLINE
C        A(1),...A(N3)  THE COEFFICIENTS OF THE B-SPLINE
C                 REPRESENTATION OF THE NATURAL SPLINE
C        C(1,1),...C(M2,N2-1)  THE COEFFICIENTS OF THE PIECEWISE
C                 POLYNOMIAL REPRESENTATION OF THE NATURAL SPLINE
C     TEMPORARY STORAGE:
C        T(1),...,T(N2+4*M-2)  THE KNOT SEQUENCE
C        Q(1,1),...,Q(N4,M2)  ELEMENTS OF THE BAND MATRIX OF THE
C                 EQUATIONS FOR THE CALCULATION OF THE A(I)
C        TRL(1,1),...,TRL(N4,M-1)  MATRIX FOR STORING LOWER-TRIANGULAR
C                 MATRIX OF THE LU DECOMPOSITION OF THE BAND MATRIX
C        INT(1),...,INT(N4)  VECTOR FOR RECORDING ROW INTERCHANGES
C                 DURING DECOMPOSITION OF THE BAND MATRIX
C        VNIKX(1,1),...,VNIKX(M2,MM)  MATRIX FOR STORING VALUES AND
C                 DERIVATIVES OF B-SPLINES AS NEEDED
      DIMENSION X(1),Y(1),T(1),Q(N4,1),A(1),VNIKX(M2,MM),TRL(N4,1),
     1  INT(1),C(M2,1)
      DO 5 I=1,MM
    5    T(I)=X(1)
      N2M1=N2-1
      DO 6 I=1,N2M1
    6    T(I+MM)=X(I)
      N3=N2M1 + MM
      DO 7 I=1,M2
    7    T(I+N3)=X(N2)
C     GET COEFFICIENTS OF FIRST M-1 ROWS
      CALL BSPLVD(T,M2,X(1),M2,VNIKX,MM)
      A1=Y(1)/VNIKX(1,1)
      MM1=M-1
      IF(M.EQ.1) GO TO 70
      DO 40 I=1,MM1
         DO 41 J=1,MM
```

```
   41          Q(I,J)=0.
            MPI=M+I
            LL=M-I
            DO 42 L=2,MPI
               LL=LL+1
   42          Q(I,LL)=VNIKX(L,MPI)
   40       A(I)=-A1*VNIKX(1,MPI)
C     GET COEFFICIENTS OF NEXT N2-2 ROWS
   70 MM2=M-2
      DO 30 I=2,N2M1
         CALL BSPLVN(T,M2,1,X(I),I+MM,VNIKX)
         IM2=I+MM2
         DO 16 L=1,MM
   16       Q(IM2,L)=VNIKX(L,1)
   30    A(IM2)=Y(I)
C     GET COEFFICIENTS OF LAST M-1 ROWS
      CALL BSPLVD(T,M2,X(N2),N3,VNIKX,MM)
      ANP=Y(N2)/VNIKX(M2,1)
      IF(M.EQ.1) GO TO 80
      DO 50 I=1,MM1
         II=N2+M2-3-I
         DO 51 J=1,MM
   51       Q(II,J)=0.
         MPI=M+I
         MPIM1=MPI-1
         LL=M-I
         DO 52 L=1,MPIM1
            LL=LL+1
   52       Q(II,L)=VNIKX(LL,MPI)
   50       A(II)=-ANP*VNIKX(M2,MPI)
   80 CONTINUE
      CALL BANDET(TRL,INT,Q,N4,M-1,2*M-1)
      CALL BANDSL(TRL,INT,A,Q,N4,M-1,2*M-1)
      DO 60 I=1,N4
         J=N3-I
   60    A(J)=A(J-1)
      A(1)=A1
      A(N3)=ANP
      CALL BSPLPP(T,A,N3,M2,Q,X,C,LXI)
         DO 12 I=1,N2M1
            FAC=1.0
            DO 12 J=2,M2
               FJ=FLOAT(J-1)
               FAC=FAC*FJ
   12          C(J,I)=C(J,I)/FAC
      RETURN
      END
C     ************************ START OF BANDET ****************************
C                                  PRIOR TO 03/23/72
C     BANDET AND BANDSL ARE TWO SUBROUTINES WHICH SOLVE C*X = B
C     WHEN C IS AN UNSYMMETRIC BAND MATRIX (THEY WILL WORK WITH
C     SYMMETRIC BAND MATRICES BUT TAKE NO ADVANTAGE OF THEIR
C     STRUCTURE).
C
C     C HAS M1 SUBDIAGONALS AND M2 SUPERDIAGONALS.
C     THE MATRIX C IS TRANSFORMED TO A BY MAKING EACH DIAGONAL OF
C     C A COLUMN OF A.  THUS A IS NX(M1+M2+1) WHEN C IS NXN.
C     A TYPICAL A IS PICTURED BELOW.  C HERE IS 4X4, WITH
C     2 SUBDIAGONALS AND 1 SUPERDIAGONAL
C
C     0          0          C(1,1)          C(1,2)
```

```
C       0            C(2,1)          C(2,2)          C(2,3)
C  C(3,1)            C(3,2)          C(3,3)          C(3,4)
C  C(4,2)            C(4,3)          C(4,4)          0
C
C       THIS TRANSFORMATION IS THE FOLLOWING, ASSUMING THAT C(I,J)
C       IS A BAND ELEMENT IN C:
C       C(I,J) --> A(I,M1+1+(J-I))
C       ALL OTHER ELEMENTS OF A ARE 0
C
C       BANDET FINDS THE LU DECOMPOSITION OF A, STORING
C       THE LOWER TRIANGULAR MATRIX IN M AND NXM1 MATRIX,
C       AND OVERWRITING THE UPPER TRIANGULAR MATRIX INTO A.
C       BANDSL USES THIS DECOMPOSITION TO SOLVE A*X = B WHERE
C       THE RIGHT HAND SIDE IS INPUT IN THE VECTOR B, AND X IS
C       OUTPUT IN THE VECTOR B.
C
C       THESE ROUTINES WERE TRANSLATED FROM THOSE PRESENTED BY
C       J. WILKINSON IN NUMERISCHE MATHEMATIK VOL 9, P 283
C       TRANSLATOR: BARBARA RYDER, CSD SERRA HOUSE, X3124
C
        SUBROUTINE BANDET(M,INT,A,N,M1,M3)
        DIMENSION M(N,M1),INT(N),A(N,M3)
C
C       M-----AN NXM1 MATRIX FOR STORING LOWER TRIANGULAR
C              MATRIX OF LU DECOMPOSITION OF A
C       INT---AN NX1 VECTOR FOR RECORDING ROW INTERCHANGES
C              DURING DECOMPOSITION
C       A-----AN NX(M1+M2+1) MATRIX WHOSE COLUMNS ARE THE DIAGONALS
C              OF C, THE BAND MATRIX BEING DECOMPOSED
C              A(*,1) - A(*,M1)    ARE SUBDIAGONALS OF C
C              A(*,M1+1)           IS DIAGONAL OF C
C              A(*,M1+2) - A(*,M1+M2+1) ARE SUPERDIAGONALS OF C
C       N----NUMBER OF ROWS IN A
C       M1----NUMBER OF SUBDIAGONALS IN C
C       M3----TOTAL NUMBER OF DIAGONALS IN C, I.E. WIDTH OF BAND
C              M3 = M1 (# SUBDIAGS) + M2 (# SUPERDIAGS) +1
C
        REAL M
   25   L=M1
        DO 40 I=1,M1
              K2=M1+2-I
              DO 50 J=K2,M3
   50         A(I,J-L)=A(I,J)
              L=L-1
              K2=M3-L
              DO 60 J=K2,M3
   60         A(I,J)=0.0
   40   CONTINUE
   45   L=M1
        DO 70 K=1,N
        X=A(K,1)
        I=K
        K2=K+1
        IF(L.LT.N)L=L+1
   72   IF(L.LT.K2)GO TO 81
   79        DO 80 J=K2,L
             IF(ABS(A(J,1))-ABS(X))80,80,82
   82        X=A(J,1)
             I=J
   80        CONTINUE
   81   INT(K)=I
```

```
      IF(X) 73,75,73
   73 IF (I-K) 77,78,77
   77      DO 90 J=1,M3
           X=A(K,J)
           A(K,J)=A(I,J)
   90      A(I,J)=X
   78      IF(L.LT.K2) GO TO 70
   83      DO 100 J=K2,L
           M(K,J-K)=A(J,1)/A(K,1)
           X=M(K,J-K)
           DO 110 JJ=2,M3
  110      A(J,JJ-1)=A(J,JJ)-A(K,JJ)*X
  100       A(J,M3)=0.0
   70 CONTINUE
      RETURN
C
C     A SUBROUTINE CALLED ERROR MUST BE SUPPLIED TO HANDLE
C     THE ZERO PIVIT SITUATION;  A SAMPLE ROUTINE FOLLOWS
C
   75 CALL ERROR
      RETURN
      END
      SUBROUTINE ERROR
  200 FORMAT(//'  ZERO PIVOT')
      WRITE(6,200)
      RETURN
      END
C ************************** END OF BANDET ****************************
C     ************************ START OF BANDSL ***************************
C                                PRIOR TO 3/23/72
      SUBROUTINE BANDSL (M,INT,B,A,N,M1,M3)
      DIMENSION INT(N),A(N,M3),M(N,M1),B(N)
C
C     ALL PARAMETERS SAME AS IN BANDET EXCEPT FOR:
C     B-----RIGHTHAND SIDE OF LINEAR SYSTEM C*X = B
C            SOLUTION IS RETURNED IN B
C
      REAL M
      INTEGER W
      L=M1
      DO 10 K=1,N
      I=INT(K)
      IF (I-K) 11,12,11
   11 X=B(K)
      B(K)=B(I)
      B(I)=X
   12 K2=K+1
      IF (L.LT.N) L=L+1
   14 IF(L.LT.K2) GO TO 13
   15      DO 20 I=K2,L
           X=M(K,I-K)
   20      B(I)=B(I)-X*B(K)
   10 CONTINUE
      L=1
      DO 30 II=1,N
      I=N+1-II
      X=B(I)
      W=I-1
      IF(L-1) 32,33,32
   32      DO 40 K=2,L
   40      X=X-A(I,K)*B(K+W)
```

```
   33 B(I)=X/A(I,1)
      IF(L-M3)31,30,30
   31 L=L+1
   30 CONTINUE
      RETURN
      END
C ************************** END OF BANDSL ****************************
      SUBROUTINE BSPLDR(T,A,N,K,ADIF,NDERIV)
CONSTRUCTS DIV.DIFF.TABLE FOR B-SPLINE COEFF. PREPARATORY TO DERIV.CALC.
      DIMENSION T(1),A(1),ADIF(N,NDERIV)
      DO 10 I=1,N
   10    ADIF(I,1) = A(I)
      KMID = K
      DO 20 ID=2,NDERIV
         KMID = KMID - 1
         FKMID = FLOAT(KMID)
         DO 20 I=ID,N
            IPKMID = I + KMID
            DIFF = T(IPKMID) - T(I)
            IF(DIFF .EQ. 0.)                GO TO 20
            ADIF(I,ID) = (ADIF(I,ID-1) - ADIF(I-1,ID-1))/DIFF*FKMID
   20       CONTINUE
                                            RETURN
      END
      SUBROUTINE BSPLEV(T,ADIF,N,K,X,SVALUE,NDERIV)
CALCULATES VALUE OF SPLINE AND ITS DERIVATIVES AT *X* FROM B-REPR.
      DIMENSION T(1),ADIF(N,NDERIV),SVALUE(1)
      DIMENSION VNIKX(20)
      DO 5 IDUMMY=1,NDERIV
    5    SVALUE(IDUMMY) = 0.
      KM1 = K-1
      CALL INTERV(T(K),N+1-KM1,X,I,MFLAG)
      I = I+KM1
      IF (MFLAG)                            99,20,9
    9 IF(X .GT. T(I))                       GO TO 99
   10 IF (I .EQ. K)                         GO TO 99
      I = I - 1
      IF(X .EQ. T(I))                       GO TO 10
C
C *I* HAS BEEN FOUND IN (K,N) SO THAT T(I) .LE. X .LT. T(I+1)
C      (OR .LE. T(I+1), IF T(I) .LT. T(I+1) = T(N+1)).
   20 KP1MN = K+1-NDERIV
      CALL BSPLVN(T,KP1MN,1,X,I,VNIKX)
      ID = NDERIV
   21    LEFT = I - KP1MN
         DO 22 L=1,KP1MN
            LEFTPL = LEFT+L
   22       SVALUE(ID) = VNIKX(L)*ADIF(LEFTPL,ID) + SVALUE(ID)
         ID = ID - 1
         IF(ID .EQ. 0)                      GO TO 99
         KP1MN = KP1MN + 1
         CALL BSPLVN(T,0,2,X,I,VNIKX)
                                            GO TO 21
C
   99                                       RETURN
      END
      SUBROUTINE BSPLPP(T,A,N,K,SCRTCH,XI,C,LXI)
CONVERTS B-SPLINE REPRESENTATION TO PIECEWISE POLYNOMIAL REPRESENTATION
      DIMENSION T(1),A(1),SCRTCH(N,K),XI(1),C(K,1)
      CALL BSPLDR(T,A,N,K,SCRTCH,K)
      LXI = 0
```

```
      XI(1) = T(K)
      DO 50 ILEFT=K,N
         IF (T(ILEFT+1) .EQ. T(ILEFT))      GO TO 50
         LXI = LXI + 1
         XI(LXI+1) = T(ILEFT+1)
         CALL BSPLEV(T,SCRTCH,N,K,XI(LXI),C(1,LXI),K)
   50    CONTINUE
C
                                             RETURN
      END
      SUBROUTINE BSPLVD ( T, K, X, ILEFT, VNIKX, NDERIV  )
C CALCULATES VALUE AND DERIV.S OF ALL B-SPLINES WHICH DO NOT VANISH AT X
      DIMENSION T(1) ,VNIKX(K,NDERIV)
      DIMENSION A(20,20)
C
C FILL VNIKX(J,IDERIV), J=IDERIV, ... ,K  WITH NONZERO VALUES OF
C B-SPLINES OF ORDER K+1-IDERIV, IDERIV=NDERIV, ... ,1, BY REPEATED
C CALLS TO BSPLVN
      CALL BSPLVN(T,K+1-NDERIV,1,X,ILEFT,VNIKX(NDERIV,NDERIV))
      IF (NDERIV .LE. 1)                   GO TO 99
      IDERIV = NDERIV
      DO 15 I=2,NDERIV
         IDERVM = IDERIV-1
         DO 11 J=IDERIV,K
   11       VNIKX(J-1,IDERVM) = VNIKX(J,IDERIV)
         IDERIV=IDERVM
         CALL BSPLVN(T,0,2,X,ILEFT,VNIKX(IDERIV,IDERIV))
   15    CONTINUE
C
      DO 20 I=1,K
         DO 19 J=1,K
   19       A(I,J) = 0.
   20    A(I,I) = 1.
      KMD = K
      DO 40 M=2,NDERIV
         KMD = KMD-1
         FKMD = FLOAT(KMD)
         I = ILEFT
         J = K
   21       JM1 = J - 1
            IPKMD = I + KMD
            DIFF = T(IPKMD) - T(I)
            IF (JM1 .EQ. 0)                GO TO 26
            IF (DIFF .EQ. 0.)              GO TO 25
            DO 24 L=1,J
   24          A(L,J) = (A(L,J) - A(L,J-1))/DIFF*FKMD
   25       J = JM1
            I = I - 1
                                           GO TO 21
   26    IF (DIFF .EQ. 0.)                 GO TO 30
         A(1,1) = A(1,1)/DIFF*FKMD
C
   30    DO 40 I=1,K
            V = 0.
            JLOW = MAX0(I,M)
            DO 35 J=JLOW,K
   35          V = A(I,J)*VNIKX(J,M) + V
   40       VNIKX(I,M) = V
   99                                      RETURN
      END
      SUBROUTINE BSPLVN(T,JHIGH,INDEX,X,ILEFT,VNIKX)
```

```
CALCULATES THE VALUE OF ALL POSSIBLY NONZERO B-SPLINES AT *X* OF
C ORDER MAX(JHIGH,(J+1)(INDEX-1)) ON *T*.
      DIMENSION T(1),VNIKX(1)
      DIMENSION DELTAM(20),DELTAP(20)
      DATA J/1/,DELTAM,DELTAP/40*0./
                                        GO TO (10,20),INDEX
   10 J = 1
      VNIKX(1) = 1.
      IF (J .GE. JHIGH)                 GO TO 99
C
   20    IPJ = ILEFT+J
         DELTAP(J) = T(IPJ) - X
         IMJP1 = ILEFT-J+1
         DELTAM(J) = X - T(IMJP1)
         VMPREV = 0.
         JP1 = J+1
         DO 26 L=1,J
            JP1ML = JP1 - L
            VM = VNIKX(L)/(DELTAP(L) + DELTAM(JP1ML))
            VNIKX(L) = VM*DELTAP(L) + VMPREV
   26       VMPREV = VM*DELTAM(JP1ML)
         VNIKX(JP1) = VMPREV
         J = JP1
         IF (J .LT. JHIGH)              GO TO 20
C
   99                                   RETURN
      END
      SUBROUTINE INTERV ( XT, LXT, X, ILEFT, MFLAG )
COMPUTES LARGEST ILEFT IN (1,LXT) SUCH THAT XT(ILEFT) .LE. X
      DIMENSION XT(LXT)
      DATA ILO/1/
      IHI = ILO + 1
      IF (IHI .LT. LXT)                 GO TO 20
         IF (X .GE. XT(LXT))            GO TO 110
         IF (LXT .LE. 1)                GO TO 90
         ILO = LXT - 1
                                        GO TO 21
   20 IF (X .GE. XT(IHI))               GO TO 40
   21 IF (X .GE. XT(ILO))               GO TO 100
C**** NOW X .LT. XT(IHI) . FIND LOWER BOUND
   30 ISTEP = 1
   31 IHI = ILO
      ILO = IHI - ISTEP
      IF (ILO .LE. 1)                   GO TO 35
      IF (X .GE. XT(ILO))               GO TO 50
      ISTEP = ISTEP*2
                                        GO TO 31
   35 ILO = 1
      IF (X .LT. XT(1))                 GO TO 33
                                        GO TO 53
C**** NOW X .GE. XT(ILO) . FIND UPPER BOUND
   40 ISTEP = 1
   41 ILO = IHI
      IHI = ILO + ISTEP
      IF (IHI .GE. LXT)                 GO TO 45
      IF (X .LT. XT(IHI))               GO TO 50
      ISTEP = ISTEP*2
                                        GO TO 41
   45 IF (X .GE. XT(LXT))               GO TO 110
      IHI = LXT
C**** NOW XT(ILO) .LE. X .LT. XT(IHI) . NARROW THE INTERVAL
```

```
   50 MIDDLE = (ILO + IHI)/2
      IF (MIDDLE .EQ. ILO)                    GO TO 100
C     NOTE. IT IS ASSUMED THAT MIDDLE = ILO IN CASE IHI = ILO+1
      IF (X .LT. XT(MIDDLE))                  GO TO 53
         ILO = MIDDLE
                                              GO TO 50
   53    IHI = MIDDLE
                                              GO TO 50
C**** SET OUTPUT AND RETURN
   90 HFLAG = -1
      ILEFT = 1
                                              RETURN
  100 MFLAG = 0
      ILEFT = ILO
                                              RETURN
  110 HFLAG = 1
      ILEFT = LXT
                                              RETURN
      END
```

APPENDIX II

```
PROCEDURE DEBNAT(INTEGER VALUE N1,N2,M; REAL ARRAY X(*);
                    REAL ARRAY A(*,*); REAL ARRAY CFF(*));
COMMENT DEBNAT COMPUTES THE COEFFICIENTS OF BOTH THE PIECEWISE
   POLYNOMIAL REPRESENTATION AND THE B-SPLINE REPRESENTATION OF A
   NATURAL SPLINE S(X) OF DEGREE (2*M-1), INTERPOLATING THE
   ORDINATES Y(I) AT POINTS X(I), I=N1 THROUGH N2.
          PIECEWISE POLYNOMIAL REPRESENTATION:
   FOR XX IN (X(I),X(I+1)),I=N1,...,N2-1,
   S(XX)=A(I,0)+A(I,1)*T+...+A(I,2*M-1)*T**(2*M-1)
   WITH T=XX-X(I).
          B-SPLINE REPRESENTATION:
   FOR XX IN (X(N1),X(N2)),
   S(XX)=CFF(1)*N(1,2*M,XX)+CFF(2)*N(2,2*M,XX)+...
          +CFF(N2-N1+2*M-1)*N(N2-N1+2*M-1,2*M,XX)
   WHERE N(J,2*M,XX) IS THE (NORMALIZED) B-SPLINE OF DEGREE
   (2*M-1) ON THE KNOT SEQUENCE T(J),...,T(J+2*M).
   INPUT:
      N1,N2 SUBSCRIPT OF FIRST AND LAST DATA POINT
      M     2*M-1 IS THE DEGREE OF THE NATURAL SPLINE,
            ADMISSIBLE VALUES RANGE FROM 1 TO N2-N1+1,
            RECOMMENDED VALUES ARE NOT GREATER THAN 7 (SAY)
      X(N1::N2) CONTAINS THE GIVEN ABSCISSAS X(I) WHICH
            MUST BE STRICTLY MONOTONE INCREASING
      A(N1::N2,0::2*M-1) CONTAINS THE GIVEN ORDINATES AS ZERO-TH
            COLUMN, I.E. A(I,0) REPRESENTS Y(I),
   OUTPUT:
      A(N1::N2,0::2*M-1) THE COEFFICIENTS OF THE PIECEWISE POLYNOMIAL
            REPRESENTATION OF THE NATURAL SPLINE,(A(N2,0) IS
            UNCHANGED AND NO VALUES ARE ASSIGNED TO THE LAST
            ROW OF A)
      CFF(1::N2-N1+2*M-1) THE COEFFICIENTS OF THE B-SPLINE
            REPRESENTATION OF THE NATURAL SPLINE;
IF (M > 0) AND (M <= N2-N1+1) THEN
BEGIN
PROCEDURE BSPLDR(REAL ARRAY T,A(*); INTEGER VALUE N,K;
                   REAL ARRAY ADIF(*,*); INTEGER VALUE NDERIV);
COMMENT CONSTRUCTS DIV. DIFF. TABLE FOR B-SPLINE COEFF.
        PREPARATORY TO DERIV. CALC.. ARRAY DIMENSIONS ARE AS
        FOLLOWS:  T(1::N+K),  A(1::N),  ADIF(1::N,1::NDERIV).
        NDERIV SHOULD BE IN (2,K);
BEGIN
   INTEGER KMID;
   REAL DIFF;
   FOR I:=1 UNTIL N DO ADIF(I,1):=A(I);
   KMID:=K;
   FOR ID:=2 UNTIL NDERIV DO
   BEGIN
      KMID:=KMID-1;
      FOR I:=ID UNTIL N DO
      BEGIN
         DIFF:=T(I+KMID) - T(I);
         IF DIFF¬=0 THEN
            ADIF(I,ID):=(ADIF(I,ID-1) - ADIF(I-1,ID-1))/DIFF*KMID
      END
   END
END BSPLDR;
PROCEDURE BSPLEV(REAL ARRAY T(*); REAL ARRAY ADIF(*,*);
                   INTEGER VALUE N,K; REAL VALUE X;
                   REAL ARRAY SVALUE(*); INTEGER VALUE NDERIV);
COMMENT CALCULATES VALUE OF SPLINE AND ITS DERIVATIVES AT X FROM
```

```
            B-REPRESENTATION.    A&RAY DIMENSIONS ARE AS FOLLOWS:
            T(1::N+K), ADIF(1::N,1::NDERIV), SVALUE(1::NDERIV);
    BEGIN
        REAL ARRAY VNIKX(1::K);
        REAL ARRAY TT(1::N+1);
        INJEGER KM1,MFLAG,I,LEFT,ID,KP1MN;
        FOR I DUMMY:=1 UNTIL NDERIV DO SVALUE(IDUMMY):=0;
        KM1:=K-1;
        FOR 1 DUMMY:=1 UNTIL N+1 DO TT(IDUMMY):=T(IDUMMY+K-1);
        INTERV(TT,N+1-KM1,X,I,MFLAG);
        I:=I+KM1;
        IF MFLAG<0 THEN GO TO S99
        ELSE IF MFLAG=O THEN GO JO S20;
        IF X>T(I) THEN GO TO S99;
S10:    IF I=K THEN GO TO S99;
        I:=I-1;
        IF X=T(I) THEN GO JO S10;
        COMMENT I HAS BEEN FOUND AN(K,N) S O THAT T(I)<=X<=T(I+1)
                (OR <=T(I+1), IF T(I)<T(I+1)=T(N+1));
S20:    KP1MN:=K+1-NDERIV;
        BSPLVN(T,KP1MN,1,X,I,VNIKX);
        ID:=NDERIV;
S21:    LEFT:= I-KP1MN;
        FOR L:=1 UNTIL KP1MN DO
            SVALUE(ID):=VNIKX(L)*ADIF(LEFT+L,ID) + SVALUE(ID);
        ID:=ID-1;
        If ID=0 THEN GC TO 393;
        KP1MN:=KP1MN+1;
        BSPLVN(T,0,2,X,I,VNIKX);
        GO TO S21;
s99 :
    END BSPLEV;
    PROCEDURE BSPLPP(REAL ARRAY T,A(*);INTEGER VALUE N,K;
                        REAL ARRAY SCRTCH(*,*);REAL ARRAY XI(*);
                        REAL ARRAY C(*,*); INTEGER RESULT LXI);
    COMMENT CONVERTS B-SPLINE REPRESENTATIONTO PIECEWISE POLYNOMIAL
            REPRESENTATION.  ARRAYDIMENSIONS ARE AS FOLLOWS:
            T(1::N+K), A(1::N), SCRTCH(1::N,1::K), XI(1::LXI+1),
            C(1::K,1::LXI).  LXI=N-K-1IF NO REPAEATED KNOTS;
    BEGIN
        BSPLDR(T,A,N,K,SCRTCH,K);
        LXI:=0;
        XI(1):=T(K);
        FOR ILEFJ:=K UNTIL N DO
            IF T(ILEFT+1)¬=T(ILEFT) THEN
            BEGIN
                LXI:=LXI+1;
                XI(LXI+1):=T(ILEFT+1J;
                BSPLEV(T,SCRTCH,N,K,XI(LXI),C(*,LXI),K)
            END
    END BSPLPP;
    PROCEDURE BSPLVN(REAL ARRAY T(*);INTEGER VALUE JHIGH,INDEX;
                        REAL VALUE X;INTEGER VALUE ILEFJ;
                        REAL ARRAY VNIKX(*)J;
    COMMENT CALCULAJES THE VALUE OF ALL POSSIBLY NONZERO B-SPLINES AT X
            O f ORDER MAX(JHIGH,(JJ+1)(INDEX-1JJ UNT.  INDEX=1 OR 2.
            JJ, DELTAM, DELTAP ARE GLOBAL VARIABLES.
            BEFORE THE FIRST CALL ONE MUST SET JJ=1 AND ALL ELEMENT'S OF
            DELTAM AN0 DELTAP = 40.  ARRAYDIMENSIONS ARE AS FOLLOWS:
            T(1::N+K), VNIKX(1::K), DELTAM(1::K), DELTAP(1::K);
    BEGIN
```

```
            INTEGER JP1,JP1ML;
            REAL VMPREV,VM;
            IF INDEX=1 THEN
            BEGIN
                JJ:=1;
                VNIKX(1):=1;
                IF JJ>=JHIGH THEN GO TO S99
            END;
      S20:  DELTAP(JJ):=T(ILEFT+JJ)-X ;
            DELTAM(JJ):= X-T(ILEFT-JJ+1);
            VMPREV:=0;
            JP1:=JJ+1;
            FOR L:=1 UNTIL JJ DO
            BEGIN
                JP1ML:=JP1-L;
                VM:=VNIKX(L)/(DELTAP(L) + DELTAM(JP1ML));
                VNIKX(L):=VM*DELTAP(L) + VMPREV;
                VMPREV:=VM*DELTAM(JP1ML)
            END;
            VNIKX(JP1):=VMPREV;
            JJ:=JP1;
            IF JJ<JHIGH THEN GO TO S20;
      S99:
        END BSPLVN;
        PROCEDURE BSPLVD(REAL ARRAY T(*); INTEGER VALUE K;
                         REAL VALUE X ; INTEGER VALUE ILEFT;
                         REAL ARRAY VNIKX(*,*);INTEGER VALUE NDERIV);
        COMMENT CALCULATES VALUE AND DERIVS. OF ALL B-SPLINES WHICH DO
                NOT VANISH AT X.  ARRAY DIMENSIONS ARE AS FOLLOWS:
                T(1::N+K) , VNIKX(1::K,1::NDERIV);
        BEGIN
            INTEGER IDERIV,IDERVM,KMD,I,J,JM1,JLOW;
            REAL V,DIFF;
            REAL ARRAY NVNIKX(1::K);
            REAL ARRAY A(1::K,1::K);
            COMMENT FILL VNIKX(J,IDERIV),J=IDERIV,...,K WITH NONZERO
                    VALUES OF B-SPLINES OF ORDER K+1-IDERIV,
                    IDERIV=NDERIV,...,1 BY REPEATED CALLS TO BSPLVN;
            BSPLVN(T,K+1-NDERIV,1,X,ILEFT,NVNIKX);
            FOR IDUMMY:=NDERIV UNTIL K DO
                VNIKX(IDUMMY,NDERIV):=NVNIKX(IDUMMY-NDERIV+1);
            IF NDERIV<=1 THEN GO TO S99;
            IDERIV:=NDERIV;
            FOR I:=2 UNTIL NDERIV DO
            BEGIN
                IDERVM:=IDERIV-1;
                FOR J:=IDERIV UNTIL K DO
                    VNIKX(J-1,IDERVM):=VNIKX(J,IDERIV);
                IDERIV:=IDERVM;
                BSPLVN(T,0,2,X,ILEFT,NVNIKX);
                FOR IDUMMY:=IDERIV UNTIL K DO
                    VNIKX(IDUMMY,IDERIV):=NVNIKX(IDUMMY-IDERIV+1)
            END;
            FOR I:=1 UNTIL K DO
            BEGIN
                FOR J:=1 UNTIL K DO
                    A(I,J):=0;
                A(I,I):=1
            END;
            KMD:=K;
            FOR M:=2 UNTIL NDERIV DO
```

```
      CALCULATES THE VALUE OF ALL POSSIBLY NONZERO B-SPLINES AT *X* OF
C ORDER MAX(JHIGH,(J+1)(INDEX-1)) ON *T*.
      DIMENSION T(1),VNIKX(1)
      DIMENSION DELTAM(20),DELTAP(20)
      DATA J/1/,DELTAM,DELTAP/40*0./
                                          GO TO (10,20),INDEX
   10 J = 1
      VNIKX(1) = 1.
      IF (J .GE. JHIGH)                   GO TO 99
C
   20    IPJ = ILEFT+J
         DELTAP(J) = T(IPJ) - X
         IMJP1 = ILEFT-J+1
         DELTAM(J) = X - T(IMJP1)
         VMPREV = 0.
         JP1 = J+1
         DO 26 L=1,J
            JP1ML = JP1 - L
            VM = VNIKX(L)/(DELTAP(L)+DELTAM(JP1ML))
            VNIKX(L) = VM*DELTAP(L) + VMPREV
   26       VMPREV = VM*DELTAM(JP1ML)
         VNIKX(JP1) = VMPREV
         J = JP1
         IF (J.LT.JHIGH)                  GO TO 20
C
   99                                     RETURN
      END
      SUBROUTINE INTERV(XT,LXT,X,ILEFT,MFLAG)
COMPUTES LARGEST ILEFT IN (1,LXT) SUCH THAT XT(ILEFT).LE. X
      DIMENSION XT(LXT)
      DATA ILO /1/
      IHI = ILO + 1
      IF (IHI.LT.LXT)                     GO TO 20
         IF (X .GE. XT(LXT))              GO TO 110
         IF (LXT .LE. 1)                  GO TO 90
         ILO = LXT - 1
                                          GO TO 21
   20 IF (X .GE. XT(IHI))                 GO TO 40
   21 IF (X.GE.XT(ILO))                   GO TO 100
C**** NOW X .LT. XT(IHI) . FIND LOWER BOUND
   30 ISTEP = 1
   31 IHI = ILO
      ILO = IHI - ISTEP
      IF (ILO .LE. 1)                     GO TO 35
      IF (X .GE. XT(ILO))                 GO TO 50
      ISTEP = ISTEP*2
                                          GO TO 31
   35 ILO = 1
      IF (X .LT. XT(1))                   GO TO 90
                                          GO TO 50
C**** NOW X .GE. XT(ILO) . FIND UPPER BOUND
   40 ISTEP = 1
   41 ILO = IHI
      IHI = ILO + ISTEP
      IF (IHI .GE. LXT)                   GO TO 45
      IF (X .LT. XT(IHI))                 GO TO 50
      ISTEP = ISTEP*2
                                          GO TO 41
   45 IF (X .GE. XT(LXT))                 GO TO 110
      IHI = LXT
C**** NOW XT(ILO) .LE. X .LT. XT(IHI) . NARROW THE INTERVAL
```

```
   50 MIDDLE = (ILO + IHI)/2
      IF (MIDDLE .EQ. ILO)                GO TO 100
C     NOTE. IT IS ASSUMED THAT MIDDLE = ILO IN CASE IHI = ILO+1
      IF (X .LT. XT(MIDDLE))              GO TO 53
         ILO = MIDDLE
                                          GO TO 50
   53    IHI = MIDDLE
                                          GO TO 50
C**** SET OUTPUT AND RETURN
   90 MFLAG = -1
      ILEFT = 1
                                          RETURN
  100 MFLAG = 0
      ILEFT = ILO
                                          RETURN
  110 HFLAG = 1
      ILEFT = LXT
                                          RETURN
      END
```

APPENDIX II

```
PROCEDURE DEBNAT( INTEGER VALUE N1,N2,M; REAL ARRAY X(*);
                  REAL ARRAY A(*,*); REAL ARRAY CFF(*));
COMMENT DEBNAT COMPUTES THE COEFFICIENTS OF BOTH THE PIECEWISE
   POLYNOMIAL REPRESENTATION AND THE B-SPLINE REPRESENTATION OF A
   NATURAL SPLINE S(X) OF DEGREE (2*M-1), INTERPOLATING THE
   ORDINATES Y(I) AT POINTS X(I), I=N1 THROUGH N2.
        PIECEWISE POLYNOMIAL REPRESENTATION:
   FOR XX IN (X(I),X(I+1)),I=N1,...,N2-1,
   S(XX)=A(I,0)+A(I,1)*T+...+A(I,2*M-1)*T**(2*M-1)
   WITH T=XX-X(I).
        B-SPLINE REPRESENTATION:
   FOR XX IN (X(N1),X(N2)),
   S(XX)=CFF(1)*N(1,2*M,XX)+CFF(2)*N(2,2*M,XX)+...
        +CFF(N2-N1+2*M-1)*N(N2-N1+2*M-1,2*M,XX)
   WHERE N(J,2*M,XX) IS THE (NORMALIZED) B-SPLINE OF DEGREE
   (2*M-1) ON THE KNOT SEQUENCE T(J),...,T(J+2*M).
   INPUT:
      N1,N2  SUBSCRIPT OF FIRST AND LAST DATA POINT
      M      2*M-1 IS THE DEGREE OF THE NATURAL SPLINE,
             ADMISSIBLE VALUES RANGE FROM 1 TO N2-N1+1,
             RECOMMENDED VALUES ARE NOT GREATER THAN 7 (SAY)
      X(N1::N2) CONTAINS THE GIVEN ABSCISSAS X(I) WHICH
             MUST BE STRICTLY MONOTONE INCREASING
      A(N1::N2,0::2*M-1) CONTAINS THE GIVEN ORDINATES AS ZERO-TH
             COLUMN, I.E. A(I,0) REPRESENTS Y(I),
   OUTPUT:
      A(N1::N2,0::2*M-1) THE COEFFICIENTS OF THE PIECEWISE POLYNOMIAL
             REPRESENTATION OF THE NATURAL SPLINE,(A(N2,0) IS
             UNCHANGED AND NO VALUES ARE ASSIGNED TO THE LAST
             ROW OF A)
      CFF(1::N2-N1+2*M-1) THE COEFFICIENTS OF THE B-SPLINE
             REPRESENTATION OF THE NATURAL SPLINE;
IF (M > 0) AND (M <= N2-N1+1) THEN
BEGIN
PROCEDURE BSPLOR(REAL ARRAY T,A(*); INTEGER VALUE N,K;
                 REAL ARRAY ADIF(*,*); INTEGER VALUE NDERIV);
COMMENT CONSTRUCTS DIV.DIFF. TABLE FOR B-SPLINE COEFF.
        PREPARATORY TO DERIV.CALC..ARRAY DIMENSIONS ARE AS
        FOLLOWS: T(1::N+K), A(1::N), ADIF(1::N,1::NDERIV).
        NDERIV SHOULD BE IN(2,K);
BEGIN
   INTEGER KMID;
   REAL DIFF;
   FOR I:=1 UNTIL N DO ADIF(I,1):=A(I);
   KMID:=K;
   FOR ID:=2 UNTIL NDERIV DO
   BEGIN
      KMID:=KMID-1;
      FOR I:=ID UNTIL N DO
      BEGIN
         DIFF:=T(I+KMID)-T(I);
         IF DIFF¬=0 THEN
          ADIF(I,ID):=(ADIF(I,ID-1) - ADIF(I-1,ID-1))/DIFF*KMID
      END
   END
END BSPLOR;
PROCEDURE BSPLEV(REAL ARRAY T(*); REAL ARRAY ADIF(*,*);
                 INTEGER VALUE N,K; REAL VALUE X;
                 REAL ARRAY SVALUE(*); INTEGER VALUE NDERIV);
COMMENT CALCULATES VALUE OF SPLINE AND ITS DERIVATIVES AT X FROM
```

```
               B-REPRESENTATION.   ARRAY DIMENSIONS ARE AS FOLLOWS:
               T(1::N+K), ADIF(1::N,1::NDERIV), SVALUE(1::NDERIV);
      BEGIN
         REAL ARRAY VNIKX(1::K);
         REAL ARRAY TT(1::N+1);
         INTEGER KM1,MFLAG,I,LEFT,ID,KP1MN;
         FOR IDUMMY:=1 UNTIL NDERIV DO SVALUE(IDUMMY):=0;
         KM1:=K-1;
         FOR IDUMMY:=1 UNTIL N+1 DO TT(IDUMMY):=T(IDUMMY+K-1);
         INTERV(TT,N+1-KM1,X,I,MFLAG);
         I:=I+KM1;
         IF MFLAG<0 THEN GO TO S99
         ELSE IF  MFLAG=0 THEN GO TO S20;
         IF X>T(I) THEN GO TO S99;
   S10:  IF I=K THEN GO TO S99;
         I:=I-1;
         IF X=T(I) THEN GO TO S10;
         COMMENT I HAS BEEN FOUND AN (K,N) SO THAT T(I)<=X<=T(I+1)
               (OR <=T(I+1), IF T(I)<T(I+1)=T(N+1));
   S20:  KP1MN:=K+1-NDERIV;
         BSPLVN(T,KP1MN,1,X,I,VNIKX);
         ID:=NDERIV;
   S21:  LEFT:=I-KP1MN;
         FOR L:=1 UNTIL KP1MN DO
            SVALUE(ID):=VNIKX(L)*ADIF(LEFT+L,ID) + SVALUE(ID);
         ID:=ID-1;
         IF ID=0 THEN GO TO S99;
         KP1MN:=KP1MN+1;
         BSPLVN(T,0,2,X,I,VNIKX);
         GO TO S21;
   S99:
      END BSPLEV;
      PROCEDURE BSPLPP(REAL ARRAY T,A(*);INTEGER VALUE N,K;
                        REAL ARRAY SCRTCH(*,*); REAL ARRAY XI(*);
                        REAL ARRAY C(*,*); INTEGER RESULT LXI);
      COMMENT CONVERTS B-SPLINE REPRESENTATION TO PIECEWISE POLYNOMIAL
               REPRESENTATION.  ARRAY DIMENSIONS ARE AS FOLLOWS:
               T(1::N+K),  A(1::N), SCRTCH(1::N,1::K), XI(1::LXI+1),
               C(1::K,1::LXI).  LXI=N-K-1 IF NO REPEATED KNOTS;
      BEGIN
         BSPLDR(T,A,N,K,SCRTCH,K);
         LXI:=0;
         XI(1):=T(K);
         FOR ILEFT:=K UNTIL N DO
            IF T(ILEFT+1)¬=T(ILEFT) THEN
            BEGIN
               LXI:=LXI+1;
               XI(LXI+1):=T(ILEFT+1);
               BSPLEV(T,SCRTCH,N,K,XI(LXI),C(*,LXI),K)
            END
      END BSPLPP;
      PROCEDURE BSPLVN(REAL ARRAY T(*);INTEGER VALUE JHIGH,INDEX;
                        REAL VALUE X; INTEGER VALUE ILEFT;
                        REAL ARRAY VNIKX(*));
      COMMENT CALCULATES THE VALUE OF ALL POSSIBLY NONZERO B-SPLINES AT X
               OF ORDER MAX(JHIGH,(JJ+1)(INDEX-1)) ON T.   INDEX=1 OR 2.
               JJ, DELTAM, DELTAP ARE GLOBAL VARIABLES.
               BEFORE THE FIRST CALL ONE MUST SET JJ=1 AND ALL ELEMENTS OF
               DELTAM AND DELTAP = 40.  ARRAY DIMENSIONS ARE AS FOLLOWS:
               T(1::N+K),  VNIKX(1::K), DELTAM(1::K), DELTAP(1::K);
      BEGIN
```

```
        INTEGER JP1,JP1ML;
        REAL VMPREV,VM;
        IF INDEX=1 THEN
        BEGIN
           JJ:=1;
           VNIKX(1):=1;
           IF JJ>=JHIGH THEN GO TO S99
        END;
   S20: DELTAP(JJ):=T(ILEFT+JJ)-X;
        DELTAM(JJ):=X-T(ILEFT-JJ+1);
        VMPREV:=0;
        JP1:=JJ+1;
        FOR L:=1 UNTIL JJ DO
        BEGIN
           JP1ML:=JP1-L;
           VM:=VNIKX(L)/(DELTAP(L) + DELTAM(JP1ML));
           VNIKX(L):=VM*DELTAP(L) + VMPREV;
           VMPREV:=VM*DELTAM(JP1ML)
        END;
        VNIKX(JP1):=VMPREV;
        JJ:=JP1;
        IF JJ<JHIGH THEN GO TO S20;
   S99:
     END BSPLVN;
     PROCEDURE BSPLVD(REAL ARRAY T(*); INTEGER VALUE K;
                      REAL VALUE X; INTEGER VALUE ILEFT;
                      REAL ARRAY VNIKX(*,*);INTEGER VALUE NDERIV);
     COMMENT CALCULATES VALUE AND DERIVS. OF ALL B-SPLINES WHICH DO
             NOT VANISH AT X.  ARRAY DIMENSIONS ARE AS FOLLOWS:
             T(1::N+K), VNIKX(1::K,1::NDERIV);
     BEGIN
        INTEGER IDERIV,IDERVM,KMD,I,J,JM1,JLOW;
        REAL V,DIFF;
        REAL ARRAY NVNIKX(1::K);
        REAL ARRAY A(1::K,1::K);
        COMMENT FILL VNIKX(J,IDERIV),J=IDERIV,...,K WITH NONZERO
                VALUES OF B-SPLINES OF ORDER K+1-IDERIV,
                IDERIV=NDERIV,...,1 BY REPEATED CALLS TO BSPLVN;
        BSPLVN(T,K+1-NDERIV,1,X,ILEFT,NVNIKX);
        FOR IDUMMY:=NDERIV UNTIL K DO
           VNIKX(IDUMMY,NDERIV):=NVNIKX(IDUMMY-NDERIV+1);
        IF NDERIV<=1 THEN GO TO S99;
        IDERIV:=NDERIV;
        FOR I:=2 UNTIL NDERIV DO
        BEGIN
           IDERVM:=IDERIV-1;
           FOR J:=IDERIV UNTIL K DO
              VNIKX(J-1,IDERVM):=VNIKX(J,IDERIV);
           IDERIV:=IDERVM;
           BSPLVN(T,0,2,X,ILEFT,NVNIKX);
           FOR IDUMMY:=IDERIV UNTIL K DO
              VNIKX(IDUMMY,IDERIV):=NVNIKX(IDUMMY-IDERIV+1)
        END;
        FOR I:=1 UNTIL K DO
        BEGIN
           FOR J:=1 UNTIL K DO
              A(I,J):=0;
           A(I,I):=1
        END;
        KMD:=K;
        FOR M:=2 UNTIL NDERIV DO
```

```
            BEGIN
                KMD:=KMD-1;
                I:=ILEFT;
                J:=K;
    S21:        JM1:=J-1;
                DIFF:=T(I+KMD) - T(I);
                IF JM1=0 THEN GO TO S26;
                IF DIFF=0 THEN GO TO S25;
                FOR L:=1 UNTIL J DO
                    A(L,J):=(A(L,J)- A(L,J-1))/DIFF*KMD;
    S25:        J:=JM1;
                I:=I-1;
                              GO TO S21;
    S26:        IF DIFF=0 THEN GO TO S30;
                A(1,1):=A(1,1)/DIFF*KMD;
    S30:        FOR I:=1 UNTIL K DO
                BEGIN
                    V:=0;
                    JLOW:=IF I>=M THEN I ELSE M   ;
                    FOR J:=JLOW UNTIL K DO
                        V:=A(I,J)*VNIKX(J,M) + V;
                    VNIKX(I,M):=V
                END;
            END;
    S99:
        END BSPLVO;
        PROCEDURE INTERV(REAL ARRAY XT(*);INTEGER VALUE LXT; REAL VALUE X;
                         INTEGER RESULT ILEFT, MFLAG);
        COMMENT COMPUTES LARGEST ILEFT IN(1,LXT) SUCH THAT XT(ILEFT)<=X.
                XT IS OF SIZE XT(1::LXT).  ILO IS A GLOBAL VARIABLE.
                ILO MUST BE SET EQUAL TO 1 BEFORE THE FIRST CALL OF INTERV;
        BEGIN
            INTEGER IHI,ISTEP,MIDDLE;
            IHI:=ILO+1;
            IF IHI<LXT THEN GO TO S20;
            IF X>=XT(LXT) THEN GO TO S110;
            IF LXT<=1 THEN GO TO S90;
            ILO:=LXT-1;
                              GO TO S21;
    S20:    IF X>=XT(IHI) THEN GO TO S40;
    S21:    IF X>=XT(ILO) THEN GO TO S100;
            COMMENT NOW X<XT(IHI).  FIND LOWER BOUND;
    S30:    ISTEP:=1;
    S31:    IHI:=ILO;
            ILO:=IHI-ISTEP;
            IF ILO<=1 THEN GO TO S35;
            IF X>=XT(ILO) THEN GO TO S50;
            ISTEP:=ISTEP*2;
                              GO TO S31;
    S35:    ILO:=1;
            IF X<XT(1) THEN GO TO S90;
                              GO TO S50;
            COMMENT NOW X>=XT(ILO).  FIND UPPER BOUND;
    S40:    ISTEP:=1;
    S41:    ILO:=IHI;
            IHI:=ILO+ISTEP;
            IF IHI>=LXT THEN GO TO S45;
            IF X<XT(IHI) THEN GO TO S50;
            ISTEP:=ISTEP*2;
                              GO TO S41;
    S45:    IF X>=XT(LXT) THEN GO TO S110;
```

```
            IHI:=LXT;
            COMMENT NOW XT(ILO)<=X<XT(IHI).  NARROW THE INTERVAL;
      S50:  MIDDLE:=(ILO+IHI) DIV 2;
            IF MIDDLE=ILO THEN GO TO S100;
            COMMENT IT IS ASSUMED THAT MIDDLE=ILO IN CASE IHI=ILO+1;
            IF X<XT(MIDDLE) THEN GO TO S53;
            ILO:=MIDDLE;
                                  GO TO S50;
      S53:  IHI:=MIDDLE;
                                  GO TO S50;
            COMMENT SET OUTPUT AND EXIT;
      S90:  MFLAG:=-1;
            ILEFT:=1;
                                  GO TO SFIN;
      S100: MFLAG:=0;
            ILEFT:=ILO;
                                  GO TO SFIN;
      S110: MFLAG:=1;
            ILEFT:=LXT;
      SFIN:
         END INTERV;
         PROCEDURE BANDET(REAL ARRAY M(*,*);INTEGER ARRAY INT(*);
                           REAL ARRAY A(*,*); INTEGER VALUE N,M1,M2);
         COMMENT BANDET AND ITS COMPANION PROCEDURE BANSOL SOLVE THE SYSTEM
                 OF EQUATIONS A*X=B WHERE A IS A NONSYMMETRIC BAND MATRIX.
                 (THEY WILL WORK WITH SYMMETRIC BAND MATRICES BUT TAKE NO
                 ADVANTAGE OF THEIR STRUCTURE.)
                 THE BAND MATRIX A OF ORDER N WITH M1 SUB-DIAGONAL ELEMENTS
                 AND M2 SUPER-DIAGONAL ELEMENTS IN A TYPICAL ROW IS STORED AS
                 AN N BY (M1+M2+1) ARRAY, A(1::N,1::M1+M2+1), WITH THE
                 SUB-DIAGONAL ELEMENTS IN A(*,J),J=1,...,M1, THE DIAGONAL
                 ELEMENTS IN A(*,M1+1), AND THE SUPER-DIAGONAL ELEMENTS IN
                 A(*,J), J=M1+2,...,M1+M2+1.  THE MATRIX A IS FACTORIZED BY
                 BANDET INTO THE PRODUCT OF A LOWER-TRIANGULAR MATRIX AND AN
                 UPPER-TRIANGULAR MATRIX USING PARTIAL PIVOTING.  THE LOWER
                 TRIANGLE IS STORED AS AN N BY M1 ARRAY M(1::N,1::M1) AND THE
                 UPPER TRIANGLE IS OVERWRITTEN ON A.  DETAILS OF THE
                 INTERCHANGES ARE STORED IN THE ARRAY INT(1::N);
         BEGIN
            INTEGER I,L,M3;
            REAL X;
            M3:=M1+M2+1;
            L:=M1;
            FOR I:=1 UNTIL M1 DO
            BEGIN
               FOR J:=M1+2-I UNTIL M3 DO
                  A(I,J-L):=A(I,J);
               L:=L-1;
               FOR J:=M3-L UNTIL M3 DO
                  A(I,J):=0
            END I;
            L:=M1;
            FOR K:=1 UNTIL N DO
            BEGIN
               X:=A(K,1);
               I:=K;
               IF L<N THEN L:=L+1;
               FOR J:=K+1 UNTIL L DO
                  IF ABS(A(J,1))>ABS(X) THEN
                  BEGIN
                     X:=A(J,1);
```

```
                        I:=J
                    END J;
                INT(K):=I;
                IF I¬=K THEN
                    FOR J:=1 UNTIL M3 DO
                    BEGIN
                        X:=A(K,J);
                        A(K,J):=A(I,J);
                        A(I,J):=X
                    END J;
                FOR I:=K+1 UNTIL L DO
                BEGIN
                    X:=M(K,I-K):=A(I,1)/A(K,1);
                    FOR J:=2 UNTIL M3 DO
                        A(I,J-1):=A(I,J) - X*A(K,J);
                    A(I,M3):=0
                END I
            END K
        END BANDET;
        PROCEDURE BANSOL(REAL ARRAY M(*,*); INTEGER ARRAY INT(*);
                         REAL ARRAY B(*); REAL ARRAY A(*,*);
                         INTEGER VALUE N,M1,M2);
        COMMENT THE PARAMETERS M,INT,A,N,M1,M2 ARE THE SAME AS IN BANDET.
                BANSOL SOLVES THE SYSTEM DECOMPOSED BY BANDET WITH
                RIGHT-HAND SIDE B(1::N). THE SOLUTION IS RETURNED IN B;
        BEGIN
            INTEGER I,L,M3,W;
            REAL X;
            M3:=M1+M2+1;
            L:=M1;
            FOR K:=1 UNTIL N DO
            BEGIN
                I:=INT(K);
                IF I¬=K THEN
                BEGIN
                    X:=B(K);
                    B(K):=B(I);
                    B(I):=X
                END;
                IF L<N THEN L:=L+1;
                FOR I:=K+1 UNTIL L DO
                    B(I):=B(I) - M(K,I-K)*B(K)
            END K;
            L:=1;
            FOR I:=N STEP -1 UNTIL 1 DO
            BEGIN
                X:=B(I);
                W:=I-1;
                FOR K:=2 UNTIL L DO
                    X:=X- A(I,K)*B(K+W);
                B(I):=X/A(I,1);
                IF L<M3 THEN L:=L+1
            END I
        END BANSOL;
            INTEGER TN2,TN3,TN4,IM2,TMM,MPI,LL,MM2,II,IM2,LXI,JJ,ILO;
            REAL A1,ANP,FAC;
            REAL ARRAY DELTAM,DELTAP(1::2*M);
            REAL ARRAY T(1::N2-N1+4*M-1);
            REAL ARRAY TA(1::N2-N1+2*M-1);
            REAL ARRAY TX,TY(1::N2-N1+1);
            REAL ARRAY C(1::N2-N1+2*M-1,1::2*M);
```

```
REAL ARRAY TRL(1::N2-N1+2*M-3,1::M-1);
REAL ARRAY VNIKX(1::2*M,1::2*M-1);
REAL ARRAY C(1::2*M,1::N2-N1+1);
INTEGER ARRAY INT(1::N2-N1+2*M-3);
TN2:=N2-N1+1;
TM2:=2*M;
TN4:=TN2+TM2-4;
TMM:=TM2-1;
FOR I:=N1 UNTIL N2 DO
BEGIN
    TX(I-N1+1):=X(I);
    TY(I-N1+1):=A(I,0)
END I;
JJ:=ILO:=1;
FOR I:=1 UNTIL TM2 DO DELTAM(I):=DELTAP(I):=40;
FOR I:=1 UNTIL TMM DO T(I):=TX(1);
FOR I:=1 UNTIL TN2-1 DO T(I+TMM):=TX(I);
TN3:=TN2-1+TMM;
FOR I:=1 UNTIL TM2 DO T(I+TN3):=TX(TN2);
COMMENT GET COEFFICIENTS OF FIRST M-1 ROWS;
BSPLVD(T,TM2,TX(1),TM2,VNIKX,TMM);
A1:=TY(1)/VNIKX(1,1);
FOR I:=1 UNTIL M-1 DO
BEGIN
    FOR J:=1 UNTIL TMM DO Q(I,J):=0;
    MPI:=M+I;
    LL:=M-I;
    FOR L:=2 UNTIL MPI DO
    BEGIN
        LL:=LL+1;
        Q(I,LL):=VNIKX(L,MPI)
    END L;
    TA(I):=-A1*VNIKX(1,MPI)
END I;
COMMENT GET COEFFICIENTS OF NEXT TN2 ROWS;
MM2:=M-2;
FOR I:=2 UNTIL TN2-1 DO
BEGIN
    BSPLVN(T,TM2,1,TX(I),I+TMM,VNIKX(*,1));
    IM2:=I+MM2;
    FOR L:=1 UNTIL TMM DO Q(IM2,L):=VNIKX(L,1);
    TA(IM2):=TY(I)
END I:
COMMENT GET COEFFICIENTS OF LAST M-1ROWS;
BSPLVD(T,TM2,TX(TN2),TN3,VNIKX,TMM);
ANP:=TY(TN2)/VNIKX(TM2,1);
FOR I:=1 UNTIL M-1 DO
BEGIN
    II:=TN2+TM2-3-I;
    FOR J:=1 UNTIL TMM DO Q(II,J):=0;
    MPI:=M+I;
    LL:=M-I;
    FOR L:=1 UNTIL MPI-1 DO
    BEGIN
        LL:=LL+1;
        Q(II,L):=VNIKX(LL,MPI)
    END L;
    TA(II):=-ANP*VNIKX(TM2,MPI)
END I;
BANDET(TRL,INT,Q,TN4,M-1,M-1);
BANSOL(TRL,INT,TA,Q,TN4,M-1,M-1);
```

```
        FOR I:=1 UNTIL TN4 DO TA(TN3-I):=TA(TN3-I-1);
        TA(1):=A1;
        TA(TN3):=ANP;
        FOR I:=1 UNTIL TN3 DO CFF(I):=TA(I);
        BSPLPP(T,TA,TN3,TM2,Q,TX,C,LXI);
        FOR I:=A UNTIL TN2-1 DO
        BEGIN
           FAC:=1;
           FOR J:=2 UNTIL TM2 DO
           BEGIN
              FAC:=FAC*(J-1);
              C(J,I):=C(J,I)/FAC
           END J
        END I;
        FOR I:=1 UNTIL TN2-1 DO
           FOR J:=1 UNTIL TM2 DO
              A(N1+I-1,J-1):=C(J,I)
     END DEBNAT;
```