SIAM J. NUMER. ANAL.
Vol. 14, No. 3, June 1977

PACKAGE FOR CALCULATING WITH B-SPLINES*

CARL DE BOOR†

Abstract. Eight FORTRAN subprograms for dealing computationally with piecewise polynomial functions (of one variable) are presented. The package is built around an algorithm for the stable evaluation of B-splines of arbitrary order and with knots of arbitrary multiplicity. Four examples illustrate how one might use these routines: interpolation by splines of general order k (and not necessarily at the knots); least squares approximation by splines to discrete data; the determination of the derivative of a spline with respect to a knot; and the approximate solution of an ordinary differential equation with rather general side conditions by collocation.

CONTENTS

1. Piecewise polynomial functions. Let $\boldsymbol{\xi} := (\xi_i)_1^{l+1}$ be a strictly increasing real sequence and let k be a positive integer. If $P_1, \cdots, P_l$ is any sequence of l polynomials, each of order k (or, degree $<k$), then we define a corresponding *piecewise polynomial* (or, *pp*) *function f of order k* by the prescription

$$f(t) := P_i(t) \quad \text{if } \xi_i < t < \xi_{i+1}; \quad i = 1, \cdots, l.$$

Whenever convenient, we think of such an f as defined on the whole real line $\mathbb{R}$ by extension of the first and the last piece, i.e.,

$$f(t) := \begin{cases} P_1(t), & \text{if } t < \xi_2, \\ P_l(t), & \text{if } \xi_l < t. \end{cases}$$

* Received by the editors November 24, 1972, and in final revised form March 25, 1976.

† Mathematics Research Center, University of Wisconsin, Madison, Wisconsin 53706. This work was supported in part by the U.S. Atomic Energy Commission Contract W-7405-ENG-36, and in part by NSF Grants GP-07163 and GP-11695 and by the United States Army under Contract DA-31-124-ARO-D-462 and DAAG29-75-C-0024.

At the (interior) *breakpoints* $\xi_2, \cdots, \xi_l$, f is as yet undefined. In a sense, f has *two* values at such a point, viz.

$$f(\xi_i^-) = P_{i-1}(\xi_i) \quad \text{and} \quad f(\xi_i^+) = P_i(\xi_i).$$

For definiteness and in order to obtain a (single valued) function, the programs below arbitrarily make f continuous from the right, i.e.,

$$f(\xi_i) := f(\xi_i^+), \quad \text{for } i = 2, \cdots, l.$$

We will denote the collection of all such pp functions of order k with breakpoint sequence $\boldsymbol{\xi} = (\xi_i)_1^{l+1}$ by

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

Note that $\mathbb{P}_{k,\boldsymbol{\xi}}$ is a linear space, of dimension kl since it is isomorphic to the direct product of l copies of

$$\mathbb{P}_k := \text{the linear space of all polynomials of order } k \text{ (degree} < k).$$

A pp function can be represented in a computer in a variety of ways. If such a function f and some of its derivatives are to be evaluated (for graphing purposes, say), then the following representation seems most convenient and efficient:

The *pp-representation for* $f \in \mathbb{P}_{k,\boldsymbol{\xi}}$ *consists of*

(i) *the integers* k *and* l, *giving order and number of its polynomial pieces, respectively*;

(ii) *the strictly increasing sequence* $\xi_1, \xi_2, \cdots, \xi_{l+1}$ *of its breakpoints*; *and*

(iii) *the matrix* $C = (c_{ji})_{j=1;i=1}^{k}{}^{l}$ *of its right derivatives at the breakpoints, i.e.*,

$$c_{ji} := D^{j-1}f(\xi_i^+), \qquad j = 1, \cdots, k; \quad i = 1, \cdots, l.$$

In terms of these numbers, the value of the jth derivative D^jf of f at a point t is found (in PPVALU) as

$$(D^jf)(t) = \sum_{r=j}^{k-1} c_{r+1,i}(t-\xi_i)^{r-j}/(r-j)!, \tag{1.1a}$$

where (with the earlier conventions) i is the integer such that

$$\begin{aligned} &\text{either:} && i = 1 && \text{and} && t < \xi_2 \\ &\text{or:} && 1 < i < l && \text{and} && \xi_i \leqq t < \xi_{i+1} \\ &\text{or:} && i = l && \text{and} && \xi_l \leqq t. \end{aligned} \tag{1.1b}$$

Note that ξ_1 and ξ_{l+1} are, strictly speaking, not breakpoints but that ξ_1 serves as the point of expansion for the first polynomial piece.

An alternative representation makes use of the *truncated power function*

$$(x)_+^r := (\max\{x, 0\})^r$$

and the linear functional

$$\text{jump}_\alpha f := f(\alpha^+) - f(\alpha^-).$$

Setting

$$\lambda_{ij}f := \text{jump}_{\xi_i}(D^jf), \quad \text{all } i, j,$$

one easily computes that

$$\lambda_{ij}(t-\xi_r)_+^s/s! = \delta_{ir}\delta_{js} = \begin{cases} 1, & \text{if } i=r \text{ and } j=s, \\ 0, & \text{otherwise,} \end{cases}$$

which shows the double sequence

$$(t-\xi_i)_+^j/j!, \qquad i=1,\cdots,l; \quad j=0,\cdots,k-1,$$

of kl functions in $\mathbb{P}_{k,\boldsymbol{\xi}}$ to be linearly independent, hence, as $\dim \mathbb{P}_{k,\boldsymbol{\xi}} = kl$, to be a basis for $\mathbb{P}_{k,\boldsymbol{\xi}}$. Consequently, every $f \in \mathbb{P}_{k,\boldsymbol{\xi}}$ admits of the irredundant representation given by the identity

$$(1.2) \qquad f(t) = \sum_{j<k} (D^j f)(\xi_1)(t-\xi_1)^j/j! + \sum_{i=2}^{l} \sum_{j<k} (\text{jump}_{\xi_i} D^j f)(t-\xi_i)_+^j/j!.$$

This representation seems particularly attractive when solving the following common computational problem involving pp functions: Construct the particular $f \in \mathbb{P}_{k,\boldsymbol{\xi}}$ which satisfies certain conditions (usually also satisfied by some function g which f is intended to approximate), and which has a certain number of continuous derivatives. These latter conditions can be written in the form

$$(1.3) \qquad \text{jump}_{\xi_i} D^{j-1} f = 0, \quad \text{for } j=1,\cdots,\nu_i, \quad i=2,\cdots,l,$$

for some vector $\boldsymbol{\nu} := (\nu_i)_2^l$ with nonnegative integer entries.

The subset of all $f \in \mathbb{P}_{k,\boldsymbol{\xi}}$ satisfying (1.3) for a given $\boldsymbol{\nu}$ is a linear subspace of $\mathbb{P}_{k,\boldsymbol{\xi}}$ which we will denote by

$$\mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}.$$

We infer at once from (1.2) and (1.3) that every $f \in \mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}$ admits of the irredundant representation given by the identity

$$(1.4) \qquad f(t) = \sum_{j<k} (D^j f)(\xi_1)(t-\xi_1)^j/j! + \sum_{i=2}^{l} \sum_{j=\nu_i}^{k-1} (\text{jump}_{\xi_i} D^j f)(t-\xi_i)_+^j/j!.$$

In comparing this representation to the pp-representation in (1.1), two complaints come to mind: (i) the value of f at a point t can involve considerably more than just k of the coefficients; and (ii) for a very nonuniform $\boldsymbol{\xi}$, some of the basis functions become nearly linearly dependent; hence relative changes of a certain size in the coefficients may correspond to relative changes of much larger or much smaller size in the function represented.

Both of these objections can be overcome (at least for moderate k) by forming *analytically* certain linear combinations of these functions $(t-\xi_i)_+^j$ to obtain a new basis whose elements each vanish outside a "small" interval, as follows. For each t, $g(s) := (t-s)_+^{k-1}$ is a polynomial of degree $<k$ on any interval not containing the point t. Hence, if we take the kth divided difference[1] in s at points $t_i, \cdots, t_{i+k}$ all in an interval not containing t, then we will get the value 0.

Defining M_i by

$$M_i(t) := g_k(t; t_i, \cdots, t_{i+k}), \quad \text{all } t,$$

[1] For an elementary discussion of divided differences see, e.g., S. Conte and C. de Boor, *Elementary Numerical Analysis*, 2nd ed., McGraw-Hill, New York, 1972.

as the kth divided difference in s at $t_i, \cdots, t_{i+k}$ (for each fixed t) of the function

$$g_k(t;s) := (t-s)_+^{k-1}$$

we obtain a function of t which vanishes outside the smallest interval containing the points $t_i, \cdots, t_{i+k}$. Further, $M_i(t)$ is a linear combination of $(t-s)_+^{k-1}$, and also of some of its derivatives in s in case of coincidence among the t_j's, all evaluated at $s = t_i, \cdots, t_{i+k}$.

Precisely, *if* $t_{i_1}, \cdots, t_{i_r}$ *are the distinct numbers among* $t_i, \cdots, t_{i+k}$ *with* t_{i_j} *appearing* d_j *times among them,* $j = 1, \cdots, r$, *then* M_i *is a linear combination of the functions*

$$(t-t_{i_j})_+^{k-s}, \qquad s = 1, \cdots, d_j; \quad j = 1, \cdots, r.$$

Hence, M_i *is then a pp function of order* k *having breakpoints only at* $t_{i_1}, \cdots, t_{i_r}$, *and satisfying*

$$\text{jump}_{t_{ij}}(D^s M_i) = 0, \quad \textit{for } s = 0, \cdots, k-1-d_j; \quad j = 1, \cdots, r.$$

2. B-splines. We will actually deal with a scaled version of the function M_i just introduced. Since

$$g_k(s;t) - (-1)^k g_k(t;s) \equiv (s-t)^{k-1},$$

and since the kth divided difference vanishes on $\mathbb{P}_k$, we have

$$\begin{aligned} N_{i,k}(t) &:= (t_{i+k} - t_i) g_k(t_i, \cdots, t_{i+k}; t) \\ &= (t_{i+k} - t_i)(-1)^k M_i(t). \end{aligned}$$

We call $N_{i,k}$ the (*normalized*) *B-spline of order* k *on the knot sequence* $t_i, \cdots, t_{i+k}$.

B-splines were introduced by Schoenberg in [10] for uniformly spaced knot sequences, and by Curry and Schoenberg in [7] for arbitrary knot sequences. Many of their properties are discussed in [8]. Additional material on which some of the subroutines below are based can be found in [1]. The following theorem is proved in [8]:

THEOREM. *For a given strictly increasing sequence* $\boldsymbol{\xi} = (\xi_i)_1^{l+1}$, *and given nonnegative integer sequence* $\boldsymbol{\nu} = (\nu_i)_2^l$, *with* $\nu_i \leqq k$, *all* i, *set*

$$n := k + \sum_{i=2}^{l} (k - \nu_i) = kl - \sum_{i=2}^{l} \nu_i = \dim \mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}$$

and let $\mathbf{t} := (t_i)_1^{n+k}$ *be any nondecreasing sequence so that*

(i) $t_1 \leqq t_2 \leqq \cdots \leqq t_k \leqq \xi_1$, $\xi_{l+1} \leqq t_{n+1} \leqq \cdots \leqq t_{n+k}$;

(ii) *for* $i = 2, \cdots, l$, *the number* ξ_i *occurs exactly* $k - \nu_i$ *times in* $\mathbf{t}$.

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

A simple proof goes as follows: From our earlier statement about $N_{i,k}$'s cousin M_i, we know that the sequence $(N_{i,k})_1^n$ (considered as functions on $[t_k, t_{n+1}]$) lies in $\mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}$. Since $n = \dim \mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}$, it therefore suffices to show that this sequence is linearly independent (considered as functions on $[t_k, t_{n+1}]$). But this follows from the following lemma proved in [4]:

LEMMA. *Let λ_i be the linear functional given by the rule*

$$\lambda_i f := \sum_{j<k} ((-D)^{k-1-j}\psi_i)(\tau_i)(D^j f)(\tau_i)$$

with

$$\psi_i(t) := (t_{i+1}-t)\cdots(t_{i+k-1}-t)/(k-1)!$$

and τ_i some point in (t_i, t_{i+k}). Then

$$\lambda_i N_{j,k} = \delta_{ij}, \quad \textit{all } i, j.$$

This theorem gives rise to the B-representation for a pp function.

The B-representation for $f \in \mathbb{P}_{k,\xi,\nu}$ consists of

(i) *the integers k and n, giving the order of f* (as a pp function) *and the number of linear parameters* (i.e., $n = kl - \sum_i \nu_i$, the dimension of $\mathbb{P}_{k,\xi,\nu}$), *respectively*;

(ii) *the vector $\mathbf{t} = (t_i)_1^{n+k}$ containing the knots* (possibly partially coincident and constructed from $\boldsymbol{\xi}$ and $\boldsymbol{\nu}$ as in the theorem) *in increasing order*; *and*

(iii) *the vector $\mathbf{a} := (a_i)_1^n$ of 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 $\mathbf{t}$.*

In terms of these numbers, the value of the jth derivative $D^j f$ of f at a point t is found (in BSPLEV or in BVALUE) as

$$(D^j f)(t) = \sum_{r=i-k+j+1}^{i} a_{r,j+1} N_{r,k-j}(t) \tag{2.1}$$

where (cf. [1])

$$a_{r,j+1} := \begin{cases} a_r, & j=0, \\ (k-j)\dfrac{a_{rj}-a_{r-1,j}}{t_{r+k-j}-t_r}, & j>0 \end{cases} \tag{2.2}$$

(as calculated in BSPLDR, or directly in BVALUE), provided that

$$\text{either:}\quad t_i \leqq t < t_{i+1} \quad \text{and} \quad k \leqq i < n$$
$$\text{or:}\quad t_i \leqq t \leqq t_{i+1} \quad \text{and} \quad i = n.$$

Otherwise, since k, n, $\mathbf{t}$ and $\mathbf{a}$ represent f only on $[t_k, t_{n+1}]$, $(D^j f)(t)$ is set to zero. This is in contrast to the evaluation of $D^j f$ from the pp-representation of f which represents f on all of $\mathbb{R}$.

The choice of the first and last k t_i's is quite arbitrary. The specific choice

$$t_1 = t_2 = \cdots = t_k = \xi_1 \quad \text{and} \quad \xi_{l+1} = t_{n+1} = \cdots = t_{n+k}$$

has the advantage that then the two matrices

$$(D^{j-1}N_{i,k}(\xi_1^+))_{i,j=1}^k \quad \text{and} \quad (D^{j-1}N_{n+1-i,k}(\xi_{l+1}^-))_{i,j=1}^k$$

are both invertible and triangular, which helps in imposing boundary conditions on f. Specifically, the homogeneous conditions

$$D^j f(\xi_1^+) = 0 \quad \text{for } j = 0, \cdots, \nu \quad (\text{or,} \quad D^j f(\xi_{l+1}^-) = 0 \quad \text{for } j = 0, \cdots, \nu)$$

become then simply

$$a_j = 0 \quad \text{for } j = 1, \cdots, \nu+1 \quad (\text{or,} \quad a_{n-j} = 0 \quad \text{for } j = 0, \cdots, \nu)$$

in terms of the B-spline coefficients for f. With this choice for the end knots, the construction of $\mathbf{t}$ from $\boldsymbol{\xi}$ and $\boldsymbol{\nu}$ can be visualized as in the following diagram:

breakpoints	ξ_1	ξ_2	ξ_3	$\cdots$	ξ_l	ξ_{l+1}
continuity conditions	$\nu_1 = 0$	ν_2	ν_3		ν_l	$\nu_{l+1} = 0$
knot multiplicity	$k - \nu_1 = k$	$k - \nu_2$	$k - \nu_3$	$\cdots$	$k - \nu_l$	$k - \nu_{l+1} = k$
knots	t_1 $\vdots$ t_k	t_{k+1} $\vdots$ $t_{2k-\nu_2}$	$t_{2k-\nu_2+1}$ $\vdots$		$\vdots$ t_n	t_{n+1} $\vdots$ t_{n+k}

If $f \in \mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}$ is also known to be periodic with period $\xi_{l+1} - \xi_1$, then a different choice of the end knots seems indicated. Suppose that it is known, more precisely, that

$$D^j f(\xi_1^+) = D^j f(\xi_{l+1}^-) \quad \text{for } j = 0, \cdots, \nu_1 - 1 \tag{2.3}$$

for some positive integer ν_1 less than k. Then one would choose ξ_1 and ξ_{l+1} to be knots of multiplicity $k - \nu_1$ only and would obtain the remaining end knots by a periodic extension of the interior knots. Precisely, one would set

$$t_{\nu_1+1} = \cdots = t_k = \xi_1$$

and then determine the remaining end knots by

$$t_i = t_{n-\nu_1+i} + (\xi_{l+1} - \xi_1), \quad \text{all } i.$$

The required periodicity for f, i.e., the conditions (2.3), are then simply enforced by requiring the appropriate periodicity for $\mathbf{a}$, i.e.,

$$a_i = a_{n-\nu_1+i}, \quad \text{for } i = 1, \cdots, \nu_1.$$

The values of $N_{i,j}(t)$ needed are generated (in BSPLVN and, implicitly, in BVALUE) *not* by forming the divided differences directly, but from a recurrence relation for B-splines [1]:

$$g_k(t_i, \cdots, t_{i+k}; t)$$
$$= \frac{t - t_i}{t_{i+k} - t_i} g_{k-1}(t_i, \cdots, t_{i+k-1}; t) + \frac{t_{i+k} - t}{t_{i+k} - t_i} g_{k-1}(t_{i+1}, \cdots, t_{i+k}; t).$$

An error analysis of this process has been given by Cox [6]. The above recurrence relation as well as various others of interest are most directly derived by applying Leibniz' formula

$$(fg)(t_i, \cdots, t_{i+k}) = \sum_{j=i}^{i+k} f(t_i, \cdots, t_j) g(t_j, \cdots, t_{i+k})$$

for the kth divided difference of a product to the particular product

$$g_r(s;t)=g_{r-1}(s;t)(s-t)$$

for appropriate r; e.g., $r=k$ gives the above recurrence relation.

3. Conversion from one representation to the other. Conversion from a B-repr. to the pp-repr. for f is easily accomplished (in BSPLPP): The $l+1$ distinct points among $t_k, \cdots, t_{n+1}$ are stored in order in $\xi_1, \cdots, \xi_{l+1}$, and, for $i=1, \cdots, l$, the numbers $D^{j-1}f(\xi_i^+), j=1, \cdots, k$, are computed (in BSPLEV) and stored in c_{ji}, $j=1, \cdots, k$.

The conversion from the pp-repr. to a B-repr. for f is more difficult because the pp-repr. contains no *explicit* information about the smoothness of f at breakpoints, i.e., about the *minimum* knot multiplicity necessary to represent the given function, nor could such information be derived numerically, i.e., in finite precision arithmetic, from the pp-repr. But if f is *known* to lie in $\mathbb{P}_{k,\boldsymbol{\xi},\boldsymbol{\nu}}$ for a certain $\boldsymbol{\nu}$, then the appropriate knot sequence $\mathbf{t}$ can be constructed from $\boldsymbol{\xi}$ and $\boldsymbol{\nu}$ as in the theorem in § 2, and, by the lemma in that section, the corresponding coefficient sequence $\mathbf{a}$ can be obtained as

$$a_i=\sum_{r<k}((-D)^{k-1-r}\psi_i)(\tau_i)(D^rf)(\tau_i)$$

with $\psi_i(t)=(t_{i+1}-t)\cdots(t_{i+k-1}-t)/(k-1)!$ and τ_i arbitrary except that $t_i<\tau_i<t_{i+k}$. If f is so representable, then τ_i can always be chosen to be one of the breakpoints ξ_j so that the required derivatives of f can be read off directly from the pp-repr.

4. Description of the specific subprograms. We give first a summary of the FORTRAN variables and their intended meaning, and a terse summary of the subprograms and their intended use.

The **B-representation** consists of:

T(1), $\cdots$, 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), $\cdots$, A(N), B-spline coefficients for 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 $\leqq 20$.

The **pp-representation** consists of:

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

C(1, 1), $\cdots$, C(K, LXI), sort of polynomial coefficients. Precisely, C(J, I) is (J−1)st derivative at XI(I)$^+$, J = 1, $\cdots$, K; used as coefficients for the polynomial piece to the right of XI(I), I = 1, $\cdots$, LXI (also to the left if I = 1).

LXI, number of polynomial pieces.

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

Other variables are defined in the subprogram summary to follow.

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), $\cdots$, ADIF(N, NDERIV).

Expects NDERIV in the interval [2, K].

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

calculates value of spline and its derivatives at X from B-repr. and returns them in SVALUE(1), $\cdots$, SVALUE(NDERIV). Can use A for ADIF if NDERIV = 1. Otherwise, must have ADIF filled beforehand by BSPLDR. Uses INTERV and BSPLVN.

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, BSPLEV.

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) ≦ X ≦ T(ILEFT + 1) (as would be expected) then VNIKX(I, J) is filled with the value of (J − 1)st derivative of N(LEFT − K + I, K) at X, I = 1, $\cdots$, K, J = 1, $\cdots$, NDERIV. Get derivative from the right or left if X = T(ILEFT) or T(ILEFT + 1), respectively. Expects NDERIV in [1, K]. Uses BSPLVN.

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

calculates the value of all possibly nonzero B-splines at X of order J = max {JHIGH, (J + 1) * (INDEX − 1)} on T. ILEFT in input, assumed so that T(ILEFT) < T(ILEFT + 1); get *division by zero* otherwise. If T(ILEFT) ≦ X ≦ T(ILEFT + 1) (as would be expected), then VNIKX(I) is filled with value of N(ILEFT − J + I, J) at X, I = 1, $\cdots$, J. Get limit from the right or left, if X = T(ILEFT) or T(ILEFT + 1). Can save time 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.

function BVALUE (T, A, N, K, X, IDERIV)

calculates value at X of IDERIV-th derivative of spline from B-repr. Expects nonnegative IDERIV. Uses INTERV.

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

computes largest ILEFT in [1, LXT] such that XT(ILEFT) ≦ X.

$$\text{If}\left\{\begin{array}{r} \mathrm{X}<\mathrm{XT}(1) \\ \mathrm{XT}(\mathrm{I}) \leqq \mathrm{X}<\mathrm{XT}(\mathrm{I}+1) \\ \mathrm{XT}(\mathrm{LXT}) \leqq \mathrm{X} \end{array}\right\}, \text{then ILEFT}=\left\{\begin{array}{c} 1 \\ \mathrm{I} \\ \mathrm{LXT} \end{array}\right\}, \text{MFLAG}=\left\{\begin{array}{r} -1 \\ 0 \\ 1 \end{array}\right\}.$$

function PPVALU (XI, C, LXI, K, X, IDERIV)

calculates value at X of IDERIV-th derivative of spline from pp-repr. Expects nonnegative IDERIV. Uses INTERV.

a. *Subroutine* BSPLDR (**t**, **a**, n, k, ADIF, NDERIV).

```
      SUBROUTINE BSPLDR ( T, A, N, K, ADIF, NDERIV )
CONSTRUCTS DIV.DIFF.TABLE FOR B-SPLINE COEFF. PREPARATORY TO DERIV.CALC.
C     DIMENSION T(N+K)
      DIMENSION T(1),A(N),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 .NE. 0.)  ADIF(I,ID) =
     *                   (ADIF(I,ID-1) - ADIF(I-1,ID-1))/DIFF*FKMID
   20       CONTINUE
                                                     RETURN
      END
```

With T, A, N, K containing a B-repr. for a certain f, the routine generates the numbers

$$\mathrm{ADIF}(i,j) := a_{ij}, \qquad i=j,\cdots,\mathrm{N}; \quad j=1,\cdots,\mathrm{NDERIV}$$

according to (2.2), which are needed in BSPLEV for the calculation of D^jf for all $j<\mathrm{NDERIV}$ according to (2.1). Entries a_{ij} whose computation would involve division by zero because of coincidences among the t_i's are never given a value by this routine; the same coincidence among the t_i's prevents their use later on in BSPLEV.

It is a waste of time (though not fatal) to have NDERIV < 2 or K < NDERIV, provided ADIF has length $\geqq \mathrm{N} * \max\{2, \mathrm{NDERIV}\}$.

b. *Subroutine* BSPLEV (**t**, ADIF, n, k, X, SVALUE, NDERIV).

```
      SUBROUTINE BSPLEV ( T, ADIF, N, K, X, SVALUE, NDERIV )
CALCULATES VALUE OF SPLINE AND ITS DERIVATIVES AT *X* FROM B-REPR.
C     DIMENSION T(N+K)
      DIMENSION T(1),ADIF(N,NDERIV),SVALUE(NDERIV), VNIKX(20)
      ID = MAX0(MIN0(NDERIV,K),1)
      DO 5 IDUMMY=1,ID
    5    SVALUE(IDUMMY) = 0.
      CALL INTERV(T,N+1,X,I,MFLAG)
      IF (X .LT. T(K))                          GO TO 99
      IF (MFLAG .EQ. 0)                         GO TO 20
      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-ID
      CALL BSPLVN(T,KP1MN,1,X,I,VNIKX)
   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,KP1MN,2,X,I,VNIKX)
                                                GO TO 21
C
   99                                           RETURN
      END
```

With f the function whose B-repr. is contained in T, the first column of ADIF, N, and K, the routine calculates $(D^{j-1}f)(X)$ according to (2.1) and stores it in SVALUE(j), $j=1, \cdots$, NDERIV. If NDERIV$=1$, A can be used in place of ADIF. If NDERIV$\in[2, K]$, then it is *assumed* the ADIF is the result of a prior

CALL BSPLDR (T, A, N, K, ADIF, NDERIV).

The routine uses INTERV to determine the appropriate integer I such that

$$K \leqq I \leqq N \quad \text{and} \quad T(I) \leqq X < T(I+1)$$
$$\text{or} \quad T(I) < X \leqq T(I+1) \quad \text{in case } T(I+1)=T(N+1).$$

If no such I exists, SVALUE(j) is set to zero, $j=1, \cdots$, NDERIV. The routine also uses BSPLVN.

c. *Subroutine* BSPLPP (**t**, **a**, n, k, SCRTCH, ξ, C, l).

```
      SUBROUTINE BSPLPP ( T, A, N, K, SCRTCH, XI, C, LXI )
CONVERTS B-SPLINE REPRESENTATION TO PIECEWISE POLYNOMIAL REPRESENTATION
C     DIMENSION T(N+K),XI(*+1),C(K,*)
C  HERE, * = THE FINAL VALUE OF THE OUTPUT PARAMETER LXI.
      DIMENSION T(1),A(N),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
                                                RETURN
      END
```

This subroutine converts the B-repr. contained in T, A, N, K into the pp-repr. as described in § 3, storing it in XI, C, LXI, K. The routine uses BSPLDR AND BSPLEV.

d. *Subroutine* BSPLVD (**t**, k, X, ILEFT, VNIKX, NDERIV).

```
      SUBROUTINE BSPLVD ( T, K, X, ILEFT, VNIKX, NDERIV )
CALCULATES VALUE AND DERIV.S OF ALL B-SPLINES WHICH DO NOT VANISH AT X
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
C     DIMENSION T(ILEFT+K)
      DIMENSION T(1),VNIKX(K,NDERIV), A(20,20)
      IDERIV = MAX0(MIN0(NDERIV,K),1)
      KP1 = K+1
      CALL BSPLVN(T,KP1-IDERIV,1,X,ILEFT,VNIKX)
      IF (IDERIV .EQ. 1)               GO TO 99
      MHIGH = IDERIV
      DO 15 M=2,MHIGH
         JP1MID = 1
         DO 11 J=IDERIV,K
            VNIKX(J,IDERIV) = VNIKX(JP1MID,1)
   11       JP1MID = JP1MID + 1
         IDERIV = IDERIV - 1
         CALL BSPLVN(T,KP1-IDERIV,2,X,ILEFT,VNIKX)
   15    CONTINUE
C
      JLOW = 1
      DO 20 I=1,K
         DO 19 J=JLOW,K
   19       A(I,J) = 0.
         JLOW = I
```

```
20      A(I,I) = 1.
      KMD = K
      DO 40 M=2,MHIGH
         KMD = KMD-1
         FKMD = FLOAT(KMD)
         I = ILEFT
         J = K
         DO 25 LDUMMY=1,KMD
            IPKMD = I + KMD
            FACTOR = FKMD/(T(IPKMD) - T(I))
            DO 24 L=1,J
24             A(L,J) = (A(L,J) - A(L,J-1))*FACTOR
            I = I - 1
25          J = J - 1
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
```

This subroutine is of help in the efficient construction of a system of equations to determine the B-repr. for a certain f from information about its value and its derivatives (see, e.g., Example 5 below). The routine generates the value at $t=\mathrm{X}$ of all $N_{ik}(t)$ and of their first NDERIV -1 derivatives which are not trivially zero at X. Specifically, the routine returns the numbers

$$\mathrm{VNIKX}(i,\mathrm{M}) \leftarrow D^{\mathrm{M}-1}N_{i+\mathrm{ILEFT}-k,k}(\mathrm{X}), \qquad i=1,\cdots,k; \quad \mathrm{M}=1,\cdots,\mathrm{NDERIV}.$$

VNIKX is taken to be a two-dimensional array with NDERIV columns of length K. It is *assumed* that ILEFT is such that

$$\mathrm{T(ILEFT)} < \mathrm{T(ILEFT}+1) \quad \text{and} \quad \mathrm{T(ILEFT)} \leqq \mathrm{X} \leqq \mathrm{T(ILEFT}+1),$$

with $\mathrm{K} \leqq \mathrm{ILEFT} \leqq \mathrm{N}$, if T has length N+K. The routine uses BSPLVN.

e. *Subroutine* BSPLVN (**t**, JHIGH, INDEX, X, ILEFT, VNIKX).

```
      SUBROUTINE BSPLVN ( T, JHIGH, INDEX, X, ILEFT, VNIKX )
C  CALCULATES THE VALUE OF ALL POSSIBLY NONZERO B-SPLINES AT *X* OF
C  ORDER MAX(JHIGH,(J+1)(INDEX-1)) ON *T*.
C     DIMENSION T(ILEFT+JHIGH)
      DIMENSION T(1),VNIKX(JHIGH), DELTAM(20),DELTAP(20)
      DATA J/1/
C  CONTENT OF J, DELTAM, DELTAP IS EXPECTED UNCHANGED BETWEEN CALLS.
                                             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
```

This is the "technical lemma" behind BSPLEV and BSPLVD. It incorporates the second algorithm of [1], for the stable evaluation of B-splines. *Assuming* that T(ILEFT) $\leqq$ X $\leqq$ T(ILEFT + 1), the routine returns, in the one-dimensional array VNIKX, the numbers

$$\text{VNIKX}(i) \leftarrow N_{\text{ILEFT}-j+i,j}(\text{X}), \qquad i=1,\cdots,j,$$

where the value of the integer j in this expression depends on JHIGH and INDEX:

if INDEX = 1, then j = JHIGH;

if INDEX = 2, then j = max {JHIGH, J + 1}, where J contains the previous call's j.

The second option is useful in the efficient evaluation of a spline *and* its derivatives, as in BSPLEV and BSPLVD, but *assumes* that T, X, ILEFT and VNIKX, and J, DELTAM and DELTAP have been left unchanged since the previous call.

It is *assumed* that ILEFT has been so chosen that

$$\text{T(ILEFT)} < \text{T(ILEFT}+1).$$

Division by zero will result if T(ILEFT) = T(ILEFT + 1). The routine uses only the numbers T(ILEFT + 1 − j), $\cdots$, T(ILEFT + j).

f. *Function* BVALUE (**t**, **a**, n, k, X, IDERIV).

```
      FUNCTION BVALUE ( T, A, N, K, X, IDERIV )
CALCULATES VALUE AT *X* OF *IDERIV*-TH DERIVATIVE OF SPLINE FROM B-REPR.
C     DIMENSION T(N+K)
      DIMENSION T(1),A(N), AJ(20),DP(20),DM(20)
      BVALUE = 0.
      KMIDER = K - IDERIV
      IF (KMIDER .LE. 0)                    GO TO 99
C
C  *** FIND *I* IN (K,N) SUCH 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)).
      KM1 = K-1
      CALL INTERV (T, N+1, X, I, MFLAG )
      IF (X .LT. T(K))                      GO TO 99
      IF (MFLAG .EQ. 0)                     GO TO 20
      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  *** DIFFERENCE THE COEFFICIENTS *IDERIV* TIMES
   20 IMK = I-K
      DO 21 J=1,K
         IMKPJ = IMK + J
   21    AJ(J) = A(IMKPJ)
      IF (IDERIV .EQ. 0)                    GO TO 30
```

```
   22 DO 23 J=1,IDERIV
         KMJ = K-J
         FKMJ = FLOAT(KMJ)
         DO 23 JJ=1,KMJ
            IHI = I + JJ
            IHMKMJ = IHI - KMJ
   23       AJ(JJ) = (AJ(JJ+1) - AJ(JJ))/(T(IHI) - T(IHMKMJ))*FKMJ
C
C  *** COMPUTE VALUE AT *X* IN (T(I),T(I+1)) OF IDERIV-TH DERIVATIVE,
C      GIVEN ITS RELEVANT B-SPLINE COEFF. IN AJ(1),...,AJ(K-IDERIV).
   30 IF (IDERIV .EQ. KM1)                    GO TO 39
      IP1 = I+1
      DO 32 J=1,KMIDER
         IPJ = I + J
         DP(J) = T(IPJ) - X
         IP1MJ = IP1 - J
   32    DM(J) = X - T(IP1MJ)
      IDERP1 = IDERIV+1
      DO 33 J=IDERP1,KM1
         KMJ = K-J
         ILO = KMJ
         DO 33 JJ=1,KMJ
            AJ(JJ) = (AJ(JJ+1)*DM(ILO) + AJ(JJ)*DP(JJ))/(DM(ILO)+DP(JJ))
   33       ILO = ILO - 1
   39 BVALUE = AJ(1)
C
   99                                         RETURN
      END
```

Using (2.1)–(2.2) and the first algorithm in [1], for the stable evaluation of a B-spline series, this function subprogram computes and returns the value of $D^{\mathrm{IDERIV}}f$ at X, where f is the pp function whose B-repr. is contained in T, A, N, K, provided

$$T(K) \leqq X \leqq T(N+1).$$

Otherwise, it returns 0 (cf. BSPLEV). The routine uses INTERV (exactly as in BSPLEV).

The routine could be changed easily to coincide in output with PPVALU, giving the value of the first or the last polynomial piece, as the case may be, in case $X \notin [T(K), T(N+1)]$. For, whether or not $X \in [T(I), T(I+1)]$, statements 20 ff produce the value of $D^{\mathrm{IDERIV}}P$ at X where P is the polynomial which agrees with f on $[T(I), T(I+1)]$, provided I is so chosen that $K \leqq I \leqq N$ and $T(I) < T(I+1)$.

g. *Subroutine* INTERV (XT, LXT, X, ILEFT, MFLAG).

```
      SUBROUTINE INTERV ( XT, LXT, X, ILEFT, MFLAG )
C 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
         IHI = LXT
C
   20 IF (X .GE. XT(IHI))              GO TO 40
      IF (X .GE. XT(ILO))              GO TO 100
C
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
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 MFLAG = 1
      ILEFT = LXT
                                                RETURN
      END
```

It is *assumed* that XT is a one-dimensional array of length LXT containing a nondecreasing sequence of real numbers. The routine returns integers ILEFT and MFLAG as follows:

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

The program is written so as to minimize the work in the common case that this call's X is close to the previous call's X. It uses, e.g., only three comparisons if the two X's lie in the same interval and only six comparisons if the two X's lie in adjacent intervals. This is accomplished by starting the search for ILEFT with the value of ILEFT that was returned at the previous call (and was saved in the local variable ILO). If $XT(ILO) \leqq X < XT(ILO+1)$ does not hold, then the program locates ILO and IHI such that

$$XT(ILO) \leqq X < XT(IHI)$$

and, once they are found, uses bisection (on the function $f(i) := (XT(i) - X)$ to find the correct value for ILEFT. The local variable ILO is initialized to the value one.

h. *Function* PPVALU (ξ, C, l, k, X, IDERIV).

```
      FUNCTION PPVALU ( XI, C, LXI, K, X, IDERIV )
CALCULATES VALUE AT *X* OF *IDERIV*-TH DERIVATIVE OF SPLINE FROM PP-REPR
      DIMENSION XI(LXI),C(K,LXI)
      PPVALU = 0.
      FLOATK = K - IDERIV
      IF (FLOATK .LE. 0.)                 GO TO 99
      CALL INTERV(XI,LXI,X,I,NDUMMY)
      DX = X - XI(I)
      J = K
    1    PPVALU = PPVALU/FLOATK*DX + C(J,I)
         J = J-1
         FLOATK = FLOATK - 1.
         IF (FLOATK .GT. 0.)               GO TO 1
   99                                      RETURN
      END
```

Using (1.1), this function returns the value of $D^{\text{IDERIV}}f(x)$, with f the pp function whose pp-repr. is contained in XI, C, LXI, K. The routine uses INTERV.

Remarks. (i) *Restrictive and dishonest* DIMENSION *statements.* Internal DIMENSION statements (in BSPLEV, BSPLVN, BVALUE and, more importantly, in BSPLVD) arbitrarily restrict the order to

$$k \leqq 20.$$

Further, in several instances, *dishonest* DIMENSION statements (although correct according to ANSI FORTRAN) were used because it was impossible to supply the exact dimensions of an array appearing as an argument from input parameters:

(α) The array T has always been declared to have length 1, since correct specification of its length N+K would have required an additional argument equal to $n+k$ in value; and

(β) in BSPLPP, the declarations XI(1), C(K, 1) are used since the correct specifications, XI(LXI+1), C(K, LXI), depend on the parameter LXI computed in the routine. The routines as now written will therefore fail to work in any system which overdoes subscript checking. In the latter situation, one would have to use some larger number, like "89" or "923" instead of "1", and then worry about the few systems which insist that the declared dimension of an array argument in a subroutine should not exceed its declared dimension in the calling program.

(ii) *Extreme indentation of jump statements*, as carried out in the subprograms above and in the examples below, has been objected to by some people who claim that such practice makes the FORTRAN text less readable. I have ignored this objection since I feel strongly that such display of all discontinuities in the program flow makes it easier to trace that flow and so to understand the program. I am less certain about the indentation of DO-loops.

(iii) *Loss of local variables between calls.* The routines BSPLVN and INTERV will not work as efficiently as intended in any system which enforces the ANSI FORTRAN rule that local variables in a subprogram become undefined after the execution of a RETURN. In such a system, the local variable ILO in INTERV would be re-initialized to the value 1 between calls, thereby defeating a time saving mechanism built into INTERV. As to BSPLVN, the local variable J would be re-initialized to the value 1 between calls, forcing a recomputation of certain quantities, —a waste of time which the INDEX = 2 option for this routine was designed to avoid.

Anyone cursed with such a system will have to do one of two things: Either include the variable ILO, and the variables J, DELTAM, DELTAP in the calling sequence of INTERV and of BSPLVN, respectively, or put these variables into a block COMMON to be shared by these routines with the calling routine.

(iv) *Make it a package*! The casual user is likely to use *explicitly* only BSPLVD followed by BVALUE or else by BSPLPP and then PPVALU. For such a user (and probably in general) it pays to combine the routines into one package with several entry points, to be called for by the loader as one item and with their references to each other internalized.

(v) *Right vs. left limits at breakpoints.* At present, the programs produce pp functions which, together with their derivatives, are continuous from the right. This is accomplished by choosing the interval indicator ILEFT for a given argument X so that

$$\mathrm{T(ILEFT)} \leqq \mathrm{X} < \mathrm{T(ILEFT+1)} \quad \text{or} \quad \mathrm{XI(ILEFT)} \leqq \mathrm{X} < \mathrm{XI(ILEFT+1)}.$$

Strictly speaking, this procedure gives the limit from the right as the value of the pp function or its derivatives whenever the argument X coincides with a break point or a knot. If the limit from the left at such a point is wanted then one has to choose ILEFT so that

$$\mathrm{T(ILEFT)} < \mathrm{X} \leqq \mathrm{T(ILEFT+1)} \quad \text{or} \quad \mathrm{XI(ILEFT)} < \mathrm{X} \leqq \mathrm{XI(ILEFT+1)}.$$

This is done now in BSPLEV and in BVALUE whenever X is the right end point of the basic interval [T(K), T(N + 1)], and in PPVALU when X > XI(LXI). Hence with T, A, N, K containing the B-repr. for f, the statement

$$\mathrm{V = BVALUE(T, A, I-1, K, T(I), J)}$$

produces the limit from the left of $f^{(\mathrm{J})}$ at T(I); i.e., then

$$\mathrm{V} = f^{(\mathrm{J})}(\mathrm{T(I)}^-).$$

Similarly, with XI, C, LXI, K, containing the pp-repr. for f, the statement

$$\mathrm{V = PPVALU(XI, C, I-1, K, XI(I), J)}$$

produces the limit from the left of $f^{(\mathrm{J})}$ at XI(I); i.e., then

$$\mathrm{V} = f^{(\mathrm{J})}(\mathrm{XI(I)}^-).$$

The routine BSPLEV can be used similarly to produce limits from the left at knots provided one is prepared to recompute ADIF each time prior to such a call. The routines BSPLVD and BSPLVN can be caused to produce limits from the left by proper choice of the input argument ILEFT.

(vi) *Splines vs. pp functions.* The routines were written for the handling of pp functions, not of splines, where by *spline* I mean any linear combination of a B-spline sequence *considered as a function on the whole real line* $\mathbb{R}$. This distinction becomes apparent in BSPLEV and in BVALUE which evaluate the spline correctly only on [T(K), T(N + 1)] and require, incidentally, that N ≧ K, hence that T have at least 2 * K entries.

It is certainly possible to change the routines (BSPLDR, BSPLEV, BSPLVD, BSPLVN and BVALUE) by the judicious use of MAX0 and MIN0 statements in the computation of subscripts and DO ranges to allow for the correct evaluation of a given B-spline series on all of $\mathbb{R}$, and therefore, incidentally, allow for T having as few as $K+1$ entries. But at such a point, one should consider further changes such as changing from N to K to N+K and K as the integer variables in the B-repr., etc.

Such changes are quite similar to those which would internalize periodicity in such a way that the user is only required to specify the knots in one period, all other knots being generated in the subprograms by periodicity whenever needed.

5. Examples. The subroutines are written with the idea that, in determining a pp function from certain linear (or nonlinear) information about it, one would attempt to calculate its B-repr. by solving an appropriate system of equations, and then convert to pp-repr. for later use of the calculated function. In this way one makes use of the good condition (relative to other possible bases for splines [1]), and of the small support of the elements, of the B-spline basis while determining the pp function, and later exploits its piecewise polynomial character for economical evaluation. Examples 2, 3 and 5 illustrate this idea.

Example 1: *Graphing some B-splines.* This first example shows which B-spline values BSPLVD (or BSPLVN) generates; it also encourages the reader to produce for himself some graphic material for § 2. Finally it exercises the subprograms INTERV, BSPLVN, and a bit of BSPLVD.

The program below generates the values at a sequence of points of the seven parabolic B-splines on a certain knot sequence T of 10 partially coincident knots. (See Table 1.)

```
C   EXAMPLE 1, FIRST PROGRAM
      DIMENSION T(10),VALUES(7)
      DATA T /3*0.,2*1.,3.,4.,3*6./
      DATA VALUES /7*0./,K,N /3,7/
      NPOINT = 25
      XL = T(K)
      DX = (T(N+1)-T(K))/FLOAT(NPOINT-1)
      PRINT 600,(I,I=1,7)
  600 FORMAT(4H1   X,8X,7(1HNI1,4HK(X),6X))
C
      DO 10 I=1,NPOINT
         X = XL + FLOAT(I-1)*DX
         CALL INTERV( T, N+K, X, ILEFT, MFLAG )
         IF (ILEFT .GT. N)  ILEFT = N
         ILFTMK = ILEFT - K
         CALL BSPLVD ( T, K, X, ILEFT, VALUES(ILFTMK+1), 1 )
COULD HAVE USED  CALL BSPLVN(T,K,1,X,ILEFT,VALUES(ILFTMK+1))
         PRINT 610, X, VALUES
  610    FORMAT(F7.3,7F12.7)
         DO 10 J=1,K
   10       VALUES(ILFTMK+J) = 0.
                                                STOP
      END
```

TABLE 1
Output from the first program of Example 1

X	N1K(X)	N2K(X)	N3K(X)	N4K(X)	N5K(X)	N6K(X)	N7K(X)
0.0	1.000000	0.0	0.0	0.0	0.0	0.0	0.0
0.250	0.562500	0.375000	0.062500	0.0	0.0	0.0	0.0
0.500	0.250000	0.500000	0.250000	0.0	0.0	0.0	0.0
0.750	0.062500	0.375000	0.562500	0.0	0.0	0.0	0.0
1.000	0.0	0.0	1.000000	0.0	0.0	0.0	0.0
1.250	0.0	0.0	0.765625	0.223958	0.010417	0.0	0.0
1.500	0.0	0.0	0.562500	0.395833	0.041667	0.0	0.0
1.750	0.0	0.0	0.390625	0.515625	0.093750	0.0	0.0
2.000	0.0	0.0	0.250000	0.583333	0.166667	0.0	0.0
2.250	0.0	0.0	0.140625	0.598958	0.260417	0.0	0.0
2.500	0.0	0.0	0.062500	0.562500	0.375000	0.0	0.0
2.750	0.0	0.0	0.015625	0.473958	0.510417	0.0	0.0
3.000	0.0	0.0	0.0	0.333333	0.666667	0.0	0.0
3.250	0.0	0.0	0.0	0.187500	0.791667	0.020833	0.0
3.500	0.0	0.0	0.0	0.083333	0.833333	0.083333	0.0
3.750	0.0	0.0	0.0	0.020833	0.791667	0.187500	0.0
4.000	0.0	0.0	0.0	0.0	0.666667	0.333333	0.0
4.250	0.0	0.0	0.0	0.0	0.510417	0.473958	0.015625
4.500	0.0	0.0	0.0	0.0	0.375000	0.562500	0.062500
4.750	0.0	0.0	0.0	0.0	0.260417	0.598958	0.140625
5.000	0.0	0.0	0.0	0.0	0.166667	0.583333	0.250000
5.250	0.0	0.0	0.0	0.0	0.093750	0.515625	0.390625
5.500	0.0	0.0	0.0	0.0	0.041667	0.395833	0.562500
5.750	0.0	0.0	0.0	0.0	0.010417	0.223958	0.765625
6.000	0.0	0.0	0.0	0.0	0.0	0.0	1.000000

A more expensive procedure for obtaining the same numbers would use conversion to pp-repr. for each of the 7 *B*-splines, and so would exercise BSPLDR, BSPLEV, BSPLPP and PPVALU, as well as BSPLVN and INTERV, as follows.

```
C   EXAMPLE 1A,  USES *PPVALU*
       DIMENSION T(10),A(7),XI(5),C(12),SCRTCH(21)
       DATA T /3*0., 2*1., 3., 4., 3*6. /
       DATA K,N /3,7 /
       NPOINT = 25
       XL = T(K)
       DX = (T(N+1)-T(K))/FLOAT(NPOINT-1)
C
       DO 5 I=1,7
     5    A(I) = 0.
       DO 20 J=1,7
          A(J) = 1.
          CALL BSPLPP( T, A, N, K, SCRTCH, XI, C, LXI )
          DO 10 I=1,NPOINT
             X = XL + FLOAT(I-1)*DX
             VALUE = PPVALU( XI, C, LXI, K, X, 0)
    10       PRINT 610, X, VALUE
   610       FORMAT(F7.3,F12.7)
    20    A(J) = 0.
                                              STOP
       END
```

Example 2: *Spline interpolation.* For given $\boldsymbol{\xi}=(\xi_i)_1^{l+1}$, the problem is the determination of an element $f\in\mathbb{P}_{k,\boldsymbol{\xi}}\cap C^{(k-2)}$ (i.e., a spline of order k with $l-1$ simple knots) which agrees with a given function g at the points $\tau_1<\tau_2<\cdots<\tau_n$ all in $[\xi_1, \xi_{l+1}]$, where

$$n=k+l-1.$$

For its solution, generate the knot sequence T by

$$\mathrm{T}(1)=\cdots=\mathrm{T}(k)=\xi_1, \qquad \mathrm{T}(n+1)=\cdots=\mathrm{T}(n+k)=\xi_{l+1},$$
$$\mathrm{T}(k+i)=\xi_{i+1}, \qquad i=1,\cdots,l-1,$$

and let $(N_{ik})_{i=1}^n$ be the corresponding sequence of B-splines of order k which, by the theorem in § 2, is a basis for $\mathbb{P}_{k,\boldsymbol{\xi}}\cap C^{(k-2)}$. Then, Schoenberg and Whitney [11] have shown that there exists, regardless of g, exactly one $f\in\mathbb{P}_{k,\boldsymbol{\xi}}\cap C^{(k-2)}$ which agrees with g at $\tau_1,\cdots,\tau_n$ if and only if

$$N_{ik}(\tau_i)\neq 0, \qquad i=1,\cdots,n. \tag{5.1}$$

This f can be written in exactly one way as

$$f=\sum_{i=1}^n a_iN_{ik}$$

for certain coefficients $a_1,\cdots,a_n$. These coefficients can, therefore, be found as the solution to the linear system

$$\sum_{j=1}^n N_{jk}(\tau_i)a_j=g(\tau_i), \qquad i=1,\cdots,n. \tag{5.2}$$

This linear system has a *banded* coefficient matrix. For,

$$N_{jk}(\tau_i)\neq 0 \quad \text{if and only if } \tau_i\in(t_j, t_{j+k}),$$

hence, if $N_{ik}(\tau_i)\neq 0$ and therefore $\tau_i\in(t_i, t_{i+k})$, then $N_{jk}(\tau_i)$ is nonzero for at most k j's, and each such j must be such that $(t_i, t_{i+k})\cap(t_j, t_{j+k})\neq\emptyset$, i.e., $|i-j|<k$. Hence, if (5.1) is satisfied, then the coefficient matrix $(N_{jk}(\tau_i))_{i,j}$ of our system (5.2) is, at worst, a band matrix with $k-1$ lower and $k-1$ upper diagonals. If the τ_i's are regularly spaced with respect to the t_i's, then the matrix can have as little as k bands.

One should take advantage of this band structure; after all, a major point in favor of the B-spline basis is the fact that its use often leads to systems with band matrices. For this reason, it is assumed in the program fragment below (and in later examples) that a band matrix solver is available (see, e.g., [12, pp. 71–92]). Correspondingly, we store the coefficient matrix of (5.2) *diagonal by diagonal* in the first $2k-1$ *columns* of an array Q, with the main diagonal going into column k; i.e.,

$$\mathrm{Q}(i, k+j)\leftarrow N_{i+j,k}(\tau_i), \qquad j=-k+1,\cdots,k-1. \tag{5.3}$$

Now, for each i, the

CALL BSPLVN($\mathbf{t}$, k, 1, τ_i, ILEFT, DUMMY) produces the k values

$$\text{DUMMY}(r) = N_{\text{ILEFT}-k+r,k}(\tau_i), \qquad r = 1, \cdots, k,$$

if (as we assume) ILEFT is chosen so that

$$t_{\text{ILEFT}} \leqq \tau_i \leqq t_{\text{ILEFT}+1} \quad \text{and} \quad t_{\text{ILEFT}} < t_{\text{ILEFT}+1}.$$

Hence, in order to fill Q properly, we zero out the first $2k-1$ entries in the ith row of Q and then enter, for $r = 1, \cdots, k$,

$$\text{DUMMY}(r) = N_{i+(\text{ILEFT}-i-k+r),k}(\tau_i)$$

into

$$\text{Q}(i, \text{ILEFT}-i+r) = \text{Q}(i, k+(\text{ILEFT}-i-k+r)).$$

Assuming that $\text{K}=k$, $\text{N}=n$, the knot sequence $\text{T}(i)=t_i$, $i=1,\cdots,n+k$, generated from the given ξ_j's, and $\text{TAU}(i)=\tau_i$, $i=1,\cdots,n$, are all already available, the following program fragment sets up the banded coefficient matrix of (5.2) in the array Q as described above and loads the right side into an array B,

$$\text{B}(i) \leftarrow g(\tau_i), \qquad\qquad i=1,\cdots,n.$$

Statement 99 is an error return signaling violation of (5.1).

```
C   FRAGMENT FOR EXAMPLE 2
       KM1 = K-1
       NP2MK = N+2-K
       KPKM1 = K+KM1
       DO 30 I=1,N
          DO 13 J=1,KPKM1
   13        Q(I,J) = 0.
          CALL INTERV( T(K), NP2MK, TAU(I), ILEFT, MFLAG )
          ILEFT = ILEFT + KM1
          IF (MFLAG)                       99,15,14
   14        IF (I .LT. N)                 GO TO 99
             ILEFT = N
   15     CALL BSPLVN ( T, K, 1, TAU(I), ILEFT, DUMMY )
COULD HAVE USED  BSPLVD(T,K,TAU(I),ILEFT,DUMMY,1)
          L = ILEFT - I
          DO 16 J=1,K
             L = L+1
   16        Q(I,L) = DUMMY(J)
          IF (Q(I,K) .EQ. 0.)              GO TO 99
   30     B(I) = G(TAU(I))
```

Suppose now that, having found the interpolating f (in the sense that we have its B-repr. $\mathbf{t}$, $\mathbf{a}$, n, k stored in T, A, N, K, respectively), we wish to calculate $f''(\text{T0})$ for some point $\text{T0} \in [\xi_1, \xi_{l+1}]$. If no other use of f is to be made, then it does not pay to convert to pp-repr. In any event, one might do one of the following in order to have the desired number in SV by the time statement 70 is reached:

```
(A)    to the point:

   70 SV = BVALUE ( T, A, N, K, T0, 2 )


(B)    using  BSPLEV  carefully:

       SV = 0.
       CALL INTERV ( T(K), NP2MK, T0, ILEFT, MFLAG )
       IF (MFLAG)                           70,40,39
```

```
39 IF (TO .GT. T(N+1))                GO TO 70
   ILEFT = ILEFT - 1
40 CALL BSPLDR ( T(ILEFT), A(ILEFT), K, K, SCRTCH, 3 )
   CALL BSPLEV ( T(ILEFT), SCRTCH, K, K, TO, DUMMY, 3 )
70 SV = DUMMY(3)
```

(C) **using BSPLEV simply but expensively:**

```
   CALL BSPLDR ( T, A, N, K, SCRTCH, 3 )
   CALL BSPLEV ( T, SCRTCH, N, K, TO, DUMMY , 3 )
70 SV = DUMMY(3)
```

(D) **whole hog:**

```
   CALL BSPLPP ( T, A, N, K, SCRTCH, XI, C, LXI )
70 SV = PPVALU ( XI, C, LXI, K, TO, 2 )
```

It is, of course, assumed above that SCRTCH, DUMMY, XI and C have all been dimensioned appropriately, typically as sufficiently large one-dimensional arrays. Note that neither the earlier program fragment nor (B) will work on a compiler which refuses to recognize an argument consisting of a *subscripted* array name as an array of proper length.

Example 3: *Least squares approximation.* There was no real reason (except, perhaps, a quest for simplicity) in the preceding example for the interior knots of the interpolating spline to have been simple. In the present example, we merely assume that the knot sequence $\mathbf{t}=(t_i)_1^{n+k}$ is given somehow as a nondecreasing sequence, with

$$t_i<t_{i+k},\quad \text{all } i,\quad \text{and}\quad t_k<t_{k+1},\quad t_n<t_{n+1}$$

to ensure that the corresponding sequence $(N_{ik})_1^n$ of B-splines of order k is linearly independent on $[t_k, t_{n+1}]$. $(N_{ik})_1^n$ is then a basis for

$$\mathbb{S}_{k,\mathbf{t}},$$

the linear space of all functions on $[t_k, t_{n+1}]$ which are linear combinations of $N_{1k}, \cdots, N_{nk}$, or, the linear space of *polynomial splines on* $[t_k, t_{n+1}]$ *of order k with interior knots* $t_{k+1}, \cdots, t_n$.

We then consider weighted least squares approximation from $\mathbb{S}_{k,\mathbf{t}}$ to a function g given in the sense that we know

$$\mathrm{G}(i) := g(x_i),\qquad i=1,\cdots,l,$$

for some strictly increasing sequence $(x_i)_1^l$ of points in $[t_k, t_{n+1}]$: We wish to find $f\in\mathbb{S}_{k,\mathbf{t}}$ so that

$$\|g-f\|_2\leqq\|g-h\|_2,\quad\text{for all } h\in\mathbb{S}_{k,\mathbf{t}}\tag{5.4}$$

where

$$\|h\|_2^2 := \sum_{i=1}^{l}[h(x_i)]^2w_i$$

for a given sequence $(w_i)_1^l$ of positive weights.

If (N_{ik}) is linearly independent on $\{x_1, \cdots, x_l\}$, i.e., if (5.1) is satisfied for some subsequence $(\tau_i)_1^n$ of (x_i), then there is exactly one $f \in \mathbb{S}_{k,\mathbf{t}}$ so that (5.4) holds. Its coefficient vector $\mathbf{a}$ (with respect to the B-spline basis for $\mathbb{S}_{k,\mathbf{t}}$) is the solution of the so-called *normal equations*

$$\sum_{j=1}^{n} \langle N_{ik}, N_{jk}\rangle a_j = \langle N_{ik}, g\rangle, \qquad i=1, \cdots, n, \tag{5.5}$$

with $\langle \cdot, \cdot \rangle$ the *inner product*

$$\langle g, h\rangle := \sum_{i=1}^{l} g(x_i)h(x_i)w_i. \tag{5.6}$$

The $n \times n$ coefficient matrix of (5.5) is a *band matrix*: Since $N_{rk}(t) \neq 0$ if and only if $t_r < t < t_{r+k}$, we have

$$N_{ik}(t)N_{jk}(t) \equiv 0, \quad \text{if } |i-j| \geqq k;$$

therefore

$$\langle N_{ik}, N_{jk}\rangle = 0, \quad \text{if } |i-j| \geqq k,$$

showing the coefficient matrix to be banded with $<k$ nonzero subdiagonals and $<k$ nonzero superdiagonals.

As in Example 2, we intend to take advantage of this special structure and assume again that a subprogram for solving banded systems is available (a program for the Cholesky decomposition of a positive definite band matrix would be ideal; see [12, pp. 50–56]). Correspondingly, we store the coefficient matrix in an array Q by diagonals, i.e.,

$$\mathrm{Q}(i, k+j) \leftarrow \langle N_{ik}, N_{i+j,k}\rangle, \qquad j=-k+1, \cdots, k-1; \quad i=1, \cdots, n.$$

The subroutine EQUTL2 below generates (for given X(L), G(L), and WEIGHT(L), L = 1, ⋯, LX, and given N, K, and T(1), ⋯, T(N+K)) the 2 * K − 1 columns of Q together with the right sides

$$\mathrm{B}(i) := \langle N_{ik}, g\rangle, \qquad i=1, \cdots, n.$$

Since B-spline values are most efficiently generated by finding simultaneously the value of *every* nonzero B-spline at one point, the outermost loop runs over the points X(1), ⋯, X(LX), and the inner loops over some of the rows and columns of Q; hence several entries of Q are built up at the same time.

Only the main and upper diagonals are initially computed, the lower diagonals are then set by symmetry.

```
      SUBROUTINE EQUTL2 ( T, N, K, Q, B )
C     DIMENSION T(N+K),Q(N,2*K-1)
      DIMENSION T(1),Q(N,1),B(N),VNIKX(20)
      COMMON /DATA/ LX,X(200),G(200),WEIGHT(200)
      KM1 = K-1
      KPKM1 = K+KM1
      DO 7 I=1,N
         B(I) = 0.
         DO 7 J=1,KPKM1
```

```
 7          Q(I,J) = 0.
      ILEFT = K
      IMK = 0
      DO 20 L=1,LX
10          IF (ILEFT .EQ. N)             GO TO 15
            IF (X(L) .LT. T(ILEFT+1))     GO TO 15
            ILEFT = ILEFT+1
            IMK = ILEFT-K
                                          GO TO 10
15       CALL BSPLVN(T,K,1,X(L),ILEFT,VNIKX)
         DO 20 JJ=1,K
            DW = VNIKX(JJ)*WEIGHT(L)
            I = IMK + JJ
            B(I) = DW*G(L) + B(I)
            J = K
            DO 20 M=JJ,K
               Q(I,J) = DW*VNIKX(M) + Q(I,J)
20             J = J + 1
      NM1 = N-1
      DO 30 I=1,NM1
         DO 30 J=1,KM1
30          Q(I+J,K-J) = Q(I,K+J)
                                          RETURN
      END
```

Example 4: *Differentiating B-splines with respect to knots.* Quite often, an approximation such as the one obtained in the preceding example can be improved greatly by repositioning the given interior knots $t_{k+1}, \cdots, t_n$. This leads to the problem of least squares approximation by splines with *variable* knots: To choose $\mathbf{a}=(a_i)_1^n$, and (simple) knots $t_{k+1}, \cdots, t_n$ in (t_k, t_{n+1}) so as to minimize

$$E(\mathbf{a}, \mathbf{t}) := \left\| g - \sum_{i=1}^{n} a_i N_{ik} \right\|_2^2$$

with

$$\|h\|_2^2 := \langle h, h \rangle, \quad \text{all } h;$$

i.e., $\|\cdot\|_2$ is the norm derived from some inner product, such as (5.6).

Some methods for minimizing E require knowledge of the first partial derivative of E with respect to each of the $2n-k$ variables. The partial derivative with respect to a_i is simply

$$(\partial/\partial a_i)E = -2\left\langle g - \sum_j a_j N_{jk}, N_{ik} \right\rangle;$$

whereas the partial with respect to t_i is

$$(\partial/\partial t_i)E = -2\left\langle g - \sum_j a_j N_{jk}, \sum_j a_j (\partial N_{jk}/\partial t_i) \right\rangle$$

and therefore requires the evaluation of

$$\partial N_{jk}/\partial t_i.$$

By the definition of N_{jk} (see § 2)

$$N_{jk}(t) = g_k(t_{j+1}, \cdots, t_{j+k}; t) - g_k(t_j, \cdots, t_{j+k-1}; t),$$

hence

$$(\partial/\partial t_i)N_{jk}(t) = d_{j+1} - d_j$$

with

$$d_j := (\partial/\partial t_i)g_k(t_j, \cdots, t_{j+k-1}; t).$$

If $j \notin [i-k+1, i]$, i.e., if $i \notin [j, j+k-1]$, then $g_k(t_j, \cdots, t_{j+k-1}; t)$ does not depend on t_i and so $d_j = 0$ in that case. Otherwise, for $j \in [i-k+1, i]$, from the general theory of divided differences,

$$\begin{aligned} d_j &= (\partial/\partial t_i)g_k(t_j, \cdots, t_{i-1}, t_i, t_{i+1}, \cdots, t_{j+k-1}; t) \\ &= g_k(t_j, \cdots, t_{i-1}, t_i, t_i, t_{i+1}, \cdots, t_{j+k-1}; t). \end{aligned}$$

Therefore, with $\hat{N}_{jk}$ the B-spline based on the knot sequence $\hat{\mathbf{t}} = (\hat{t}_j)$ with

$$\hat{t}_s := \begin{cases} t_s, & s \leqq i, \\ t_{s-1}, & s > i \end{cases}$$

(i.e., with the multiplicity of t_i increased by one), we have

$$(\partial/\partial t_i)N_{jk}(t) = d_{j+1} - d_j$$

where

$$d_j = \begin{cases} \hat{N}_{jk}(t)/(t_{j+k-1} - t_j), & i-k < j \leqq i, \\ 0, & \text{otherwise.} \end{cases}$$

With T(K)≦T(L)<T(L+1)≦T(N+1) and T(L)≦XX≦T(L+1), one might generate, for $i = k+1, \cdots, n$ (in sequence), the $k+1$ (not trivially zero) numbers

$$\mathrm{VD}(m) := (\partial/\partial t_i)N_{mk}(\mathrm{XX}), \qquad m = i-k, \cdots, i,$$

as in the following program fragment, in which these $k+1$ numbers are printed out, at statement 50, for each i, for want of some better idea of what to do with them here.

```
C     FRAGMENT FOR EXAMPLE 4.  DIFFERENTIATION WRTO A KNOT.
         DO 13 JJ=1,K
   13       THAT(JJ) = T(JJ)
         NPK = N + K
         DO 14 JJ=K,NPK
   14       THAT(JJ+1) = T(JJ)
         KM1 = K - 1
         KP1 = K + 1
         DO 100 I=KP1,N
            IMK = I-K
            THAT(I) = THAT(I+1)
            DO 19 JJ=IMK,I
   19          VD(JJ) = 0.
            LHAT = L
            IF (L .GE. I)     LHAT = L + 1
            JLOW = MAX0(LHAT,I) - KM1
            JHIGH = MIN0(LHAT,I)
            IF (JLOW .GT. JHIGH)          GO TO 50
            CALL BSPLVN ( THAT, K, 1, XX, LHAT, DUMMY )
            KMLHAT = K-LHAT
            DO 40 J=JLOW,JHIGH
               KM1PJ = KM1+J
               KMLPJ = KMLHAT+J
```

```
40        VD(J-1) = DUMMY(KMLPJ)/(T(KM1PJ)-T(J))
       JJ = I
       DO 41 J=1,K
          VD(JJ) = VD(JJ) - VD(JJ-1)
41        JJ = JJ - 1
50     PRINT 650, I, (VD(J),J=IMK,I)
650    FORMAT(I5/(3E20.10))
100 CONTINUE
```

Example 5: *Solving an ODE by collocation.* This final example concerns a program for experimentation with the collocation method for solving an ordinary differential equation (ODE) which uses, explicitly or implicitly, all of the subprograms of the package. The subroutine BSPLVD was written especially for an application such as this. An earlier version of this program (see [2]) was used to calculate the numerical example in [5].

The mathematical background for this program, as found in [5], is as follows: Given a real valued function $F = F(t; z_0, \cdots, z_{m-1})$ on $\mathbb{R}^{m+1}$, and continuous linear functionals $\beta_1, \cdots, \beta_m$ on $C^{(m-1)}[a, b]$ together with numbers $c_1, \cdots, c_m$, the problem is to find a function g on $[a, b]$ such that

$$(D^m g)(t) = F(t; g(t), \cdots, (D^{m-1}g)(t)) \quad \text{on } [a, b],$$
$$\beta_i g = c_i, \qquad i = 1, \cdots, m.$$

Assuming g to be such a function (and the only one in some neighborhood of g), we attempt to approximate it by pp functions, using collocation. Specifically, with $\boldsymbol{\xi} = (\xi_i)_1^{l+1}$ given (with $\xi_1 = a$, $\xi_{l+1} = b$, say), we look for $f \in \mathbb{P}_{k+m,\boldsymbol{\xi}} \cap C^{(m-1)}$ for which

$$\text{(5.7)} \qquad \begin{aligned} (D^m f)(\tau_i) &= F(\tau_i; f(\tau_i), \cdots, (D^{m-1}f)(\tau_i)), \qquad & i = 1, \cdots, kl, \\ \beta_i f &= c_i, & i = 1, \cdots, m. \end{aligned}$$

Here, we choose the *collocation points* $(\tau_i)_1^{kl}$ k to a subinterval and distributed the same in each interval; i.e., with

$$-1 \leqq \rho_1 < \rho_2 < \cdots < \rho_k \leqq 1$$

picked somehow, we set

$$\tau_{(i-1)k+r} := [\xi_{i+1} + \xi_i + \rho_r(\xi_{i+1} - \xi_i)]/2, \qquad r = 1, \cdots, k; \quad i = 1, \cdots, l.$$

The program below leaves the choice of $(\rho_i)_1^k$ to an unspecified subroutine COLPNT. See [5] for reasons why the roots of the kth Legendre polynomial (the Gauss–Legendre points) are particularly good candidates for $(\rho_i)_1^k$.

It is shown in [5] that (5.7) can be solved by Newton's method starting with a sufficiently close initial guess f_0 (provided F is sufficiently smooth and $\max_i (\xi_{i+1} - \xi_i)$ is sufficiently small); i.e., (5.7) has a solution $f = \lim_{r\to\infty} f_r$ with f_{r+1} the solution y of the *linear* problem

$$\text{(5.8)} \qquad \begin{aligned} (D^m y)(\tau_i) + \sum_{j<m} v_j(\tau_i)(D^j y)(\tau_i) &= h(\tau_i), \qquad i = 1, \cdots, kl, \\ \beta_i y = c_i, \qquad i &= 1, \cdots, m, \end{aligned}$$

where

$$v_j(t) := -(\partial F/\partial z_j)(t; f_r(t), \cdots, (D^{m-1}f_r)(t)), \qquad j = 0, \cdots, m-1,$$

and

$$h(t) := F(t; f_r(t), \cdots, (D^{m-1}f_r)(t)) + \sum_{j<m} v_j(t)(D^j f_r)(t).$$

Let $\mathbf{t} := (t_i)_1^{n+m+k}$ be the nondecreasing sequence (generated in the subroutine KNOTS below from $\boldsymbol{\xi}$, m and k) which contains each of ξ_1 and ξ_{l+1} $m+k$ times, and each of the interior breakpoints $\xi_2, \cdots, \xi_l$ k times. Then, $n = kl + m$, and

$$\mathbb{P}_{k+m,\boldsymbol{\xi}} \cap C^{(m-1)} = \mathbb{S}_{k+m,\mathbf{t}}$$

(cf. Example 3), considered as functions on $[\xi_1, \xi_{l+1}]$. The solution of (5.8) can therefore be written in the form

$$y = \sum_{j=1}^{n} a_j N_{j,k+m}$$

with $(N_{i,k+m})_1^n$ the B-splines of order $k+m$ on the knot sequence $\mathbf{t}$.

The linear system for the coefficient vector $\mathbf{a}$ of the solution y of (5.8) is banded with $k+m-1$ nonzero lower and $k+m-1$ nonzero upper diagonals, provided the linear functionals β_i are of the form

$$\beta_i y := \sum_{j<m} \alpha_{ij}(D^j y)(x_i)$$

for certain scalars α_{ij} and certain points x_i in $[\xi_1, \xi_{l+1}]$, and collocation equations and side condition equations are ordered according to the value of the independent variable they involve. Such ordering is enforced in EQUATE below, where the $2(k+m)-1$ diagonals of the coefficient matrix for (5.8) are generated (using BSPLVD in an essential way) and stored in the first $2(k+m)-1$ columns of an array Q. The linear system is then solved in a subroutine BANMAT left unspecified here; it is the band matrix solver whose availability was postulated in earlier examples.

The specific example considered here is taken from [9]:

$$\varepsilon g''(t) + [g(t)]^2 = 1 \quad \text{on } [0, 1]$$
$$g'(0) = g(1) = 0,$$

with $\varepsilon = .005$, as specified in the subroutines SOLU and SIDEC below. The linear problem (5.8) for the determination of $y = f_{r+1}$ from f_r becomes

$$\varepsilon y''(\tau_i) + v_0(\tau_i)y(\tau_i) = h(\tau_i), \qquad i = 1, \cdots, kl,$$
$$y'(0) = y(1) = 0,$$

with

$$v_0(t) := 2f_r(t), \qquad h(t) := [f_r(t)]^2 + 1.$$

After $f = \lim_r f_r$ has been obtained numerically, this approximation to the solution g is used to obtain (in a subroutine NEWNOT) a hopefully more suitable distribution of knots, based on considerations in [3] and [9].

```
C  EXAMPLE 5 , SOLUTION OF AN ODE BY COLLOCATION
      DIMENSION T(200),A(200),ASAVE(200)
      DIMENSION Q(4400),TEMPS(200),TEMPL(2000)
COULD EQUIVALENCE *Q* WITH *TEMPL* .
COMMON BLOCK /APPROX/ CONTAINS THE PP-REPRESENTATION OF THE CURRENT APPR
      COMMON /APPROX/ XI(100),C(2000),LXI,KPM
      COMMON /OTHER/ ITERMX,K,RHO(19)
C
C  *** SET PARAMETERS
C      (ALEFT,ARIGHT) IS THE INTERVAL OF APPROXIMATION
C      LXI IS THE NUMBER OF POLYNOMIAL PIECES
C      KPM IS THE ORDER OF THE POLYNOMIAL PIECES
C      XI(2), ... , XI(LXI) ARE THE INTERIOR BREAKPOINTS.
C
      DATA ALEFT,ARIGHT,LXI0,IDEGRE,NTIMES,REPEAT,RELERR
     . /   0.  , 1.   ,   4  ,   5  ,   3  , 3.   ,1.E-6/
      KPM = IDEGRE + 1
C  *** SET THE VARIOUS PARAMETERS CONCERNING THE PARTICULAR DIF.EQU.
C       INCLUDING A FIRST APPROX. IN CASE THE DE IS TO BE SOLVED BY
C       ITERATION   ( ITERMX .GT. 0 ) .
      CALL SOLU (1, TEMPS(1), TEMPS)
C  *** THE FOLLOWING FIVE STATEMENTS COULD BE REPLACED BY A READ IN
C      ORDER TO OBTAIN A SPECIFIC (NONUNIFORM) SPACING OF THE BREAKPNTS.
         DX = (ARIGHT - ALEFT)/FLOAT(LXI0)
         TEMPS(1) = ALEFT
         DO 4 J=2,LXI0
    4       TEMPS(J) = TEMPS(J-1) + DX
         TEMPS(LXI0+1) = ARIGHT
C  *** GENERATE, IN KNOTS, THE REQUIRED KNOTS T(1),...,T(N+K).
         CALL KNOTS ( 1, TEMPS, LXI0, KPM, T, N )
         NT = 1
   10       ITER = 0
            ERR = 1.
            AMAX = 0.
C  *** GENERATE THE BANDED COEFFICIENT MATRIX *Q* AND RIGHT SIDE *A*
C      FROM COLLOCATION EQUATIONS AND SIDE CONDITIONS. THEN SOLVE VIA
C      *BANMAT*, OBTAINING THE B-REPRESENTATION OF THE APPROX. IN
C      *T*, *A*, *N*, *KPM* .
            CALL EQUATE ( T, N, KPM, TEMPS, 0, A )
            CALL BANMAT(N,IDEGRE,IDEGRE,1,1,Q,N,A,N,DETERM,TEMPS)
   20          CONTINUE
               CALL BSPLPP(T,A,N,KPM,TEMPL,XI,C,LXI)
               IF (ERR .LE. RELERR*AMAX)GO TO 30
               ITER = ITER+1
               IF (ITER .GT. ITERMX)   GO TO 30



C  *** SAVE B-SPLINE COEFF. OF CURRENT APPROX. IN *ASAVE*, THEN GET NEW
C      APPROX. AND COMPARE WITH OLD. IF COEFF. ARE MORE THAN *RELERR*
C      APART (RELATIVELY) OR IF NO. OF ITERATIONS IS LESS THAN *ITERMX*
C      CONTINUE ITERATING.
               DO 25 I=1,N
   25             ASAVE(I) = A(I)
               CALL EQUATE ( T, N, KPM, TEMPS, Q, A )
               CALL BANMAT(N,IDEGRE,IDEGRE,1,1,Q,N,A,N,DETERM,TEMPS)
               ERR = 0.
               AMAX = 0.
               DO 26 I=1,N
                  AMAX = AMAX1(AMAX,ABS(A(I)))
   26             ERR = AMAX1(ERR,ABS(A(I)-ASAVE(I)))
                                             GO TO 20
C  *** ITERATION (IF ANY) COMPLETED. PRINT OUT APPROX. BASED ON CURRENT
C      BREAKPOINT SEQUENCE, THEN TRY TO IMPROVE THE SEQUENCE.
   30       PRINT 630,KPM,LXI,(XI(L),L=2,LXI)
  630       FORMAT(34H APPROXIMATION BY SPLINES OF ORDER,I2,4H ON ,I3,
     *             25H INTERVALS. BREAKPOINTS -/(5E20.10))
            IF (ITERMX .GT. 0) PRINT 637,ITER
```

```
  637       FORMAT(6H AFTER,I3,11H ITERATIONS)
            DO 38 I=1,LXI
               II = (I-1)*KPM
   38          PRINT 638, XI(I),(C(II+J),J=1,KPM)
  638       FORMAT(F9.3,E13.6,10E11.3)
C  AT THIS POINT, ONE MIGHT INSERT A CALL TO SOME SUBROUTINE *ERROR*
C  IN ORDER TO COMPARE APPROX. WITH THE EXACT ANSWER, ETC.
C  THE FOLLOWING CHECKS OUT ERROR BY EXERCISING BVALUE.
C  A CALL TO PPVALU WOULD BE MORE EFFICIENT.
            XX = ALEFT
            DX = (ARIGHT - ALEFT)/8.
            DO 85 LL=1,9
               CALL SOLU (3, XX, TEMPS)
               ERRORB = TEMPS(1) - BVALUE(T, A, N, KPM, XX, 0)
               PRINT 685, XX, ERRORB
  685          FORMAT(2E20.10)
   85          XX = XX + DX
   90       CONTINUE
            IF (NT. EQ. NTIMES) STOP
C  *** FROM THE PP-REPR. OF THE CURRENT APPROX., OBTAIN IN *NEWNOT*
C      A NEW (AND HOPEFULLY BETTER)  SEQUENCE OF BREAKPOINTS, THEIR
C      NUMBER BEING DETERMINED BY THE INITIAL NUMBER LXI0 ON ENTERING
C      THE LOOP STARTING AT 10, THE NUMBER *NT* OF TIMES THROUGH
C      THE LOOP, AND THE PARAMETER *REPEAT*.
            LXINEW = LXI0 + IFIX(FLOAT(NT)/REPEAT)
            CALL NEWNOT(XI,C,LXI,KPM,TEMPS,LXINEW,TEMPL)
            CALL KNOTS(2,TEMPS,LXINEW,KPM,T,N)
            NT = NT + 1
                                                  GO TO 10
      END


      SUBROUTINE KNOTS ( MODE, XI, LXI, KPM, T, N )
C  SETS UP THE KNOT ARRAY T FROM THE GIVEN BREAKPOINT ARRAY XI AND COLPT
C     DIMENSION XI(LXI+1),T(*+KPM)
C  HERE, * = FINAL VALUE OF THE OUTPUT PARAMETER *N*
      DIMENSION XI(1),T(1)
      COMMON /DIFEQU/ M,ISIDEC,XSIDEC(10)
      COMMON /OTHER/ ITERMX,K,RHO(19)
                                                  GO TO (9,10),MODE
    9 K = KPM - M
      CALL COLPNT(K,RHO)
C  *RHO* NOW CONTAINS THE COLL.POINTS FOR THE STAND.INTERV. (-1,1) .
   10 N = LXI*K + M
      JJ = N + KPM
      JJJ = LXI + 1
      DO 11 LL=1,KPM
         T(JJ) = XI(JJJ)
   11    JJ = JJ - 1
      DO 12 J=1,LXI
         JJJ = JJJ - 1
         DO 12 LL=1,K
            T(JJ) = XI(JJJ)
   12       JJ = JJ - 1
      DO 13 LL=1,KPM
   13    T(LL) = XI(1)
      PRINT 600, N
  600 FORMAT(I5,11H PARAMETERS)
                                                  RETURN
      END


      SUBROUTINE SIDEC ( VNIKX, KPM, N, Q, B )
C  SIDE CONDITIONS ARE SPECIFIED HERE
C  IF THE I-TH SIDECONDITION, THE ONE AT XX = XSIDEC(I), IS OF THE FORM
C           W(M)D**(M-1) + ... + W(1)D**0   =   RS
C  FOR CERTAIN CONSTANTS W(M),...,W(1),RS (DEPENDING ON I), THEN
C  DURING THE I-TH CALL TO SIDEC, THE FOLLOWING SHOULD BE EXECUTED -
C     DO * J=1,KPM
```

```
C         Q(1,J) = 0.
C         DO * L=1,M
C    *       Q(1,J) = W(L)*VNIKX(J,L) + Q(1,J)
C      B = RS $ ISIDEC = ISIDEC+1 $ RETURN
       COMMON /DIFEQU/ M,ISIDEC,XSIDEC(10)
       DIMENSION VNIKX(KPM,KPM),Q(N,KPM)
                                       GO TO (10,20,999),ISIDEC
   10 DO 11 J=1,KPM
   11    Q(1,J) = VNIKX(J,2)
       B = 0.
                                       GO TO 90
   20 DO 21 J=1,KPM
   21    Q(1,J) = VNIKX(J,1)
       B = 0.
   90 ISIDEC = ISIDEC + 1
  999                                  RETURN
       END


       SUBROUTINE EQUATE ( T, N, KPM, SCRTCH, Q, B )
C   SETS UP COLLOCATION EQUATIONS USING COLPNTS RHO AND INFO FROM *SOLU*
C      DIMENSION T(N+KPM),SCRTCH(KPM,M+1),Q(N,KPM*2-1)
       DIMENSION T(1),SCRTCH(KPM,1),Q(N,1),B(N), V(20)
       COMMON /DIFEQU/ M,ISIDEC,XSIDEC(10)
       COMMON /OTHER/ ITERMX,K,RHO(19)
       MP1 = M+1
       KPM2M1 = KPM*2 - 1
       DO 7 J=1,KPM2M1
          DO 7 I=1,N
    7        Q(I,J) = 0.
       ISIDEC = 1
       ID = 0
       DO 30 I=KPM,N,K
          XM = (T(I+1)+T(I))/2.
          DX = (T(I+1)-T(I))/2.
          DO 30 LL=1,K
             XX = XM + DX*RHO(LL)
                                       GO TO 20
   19           CALL BSPLVD(T,KPM,XSIDEC(ISIDEC),I,SCRTCH,M)
                CALL SIDEC(SCRTCH,KPM,N,Q(ID,I-ID+1),B(ID))
   20           ID = ID + 1
                IF (ISIDEC .GT. M)     GO TO 21
                IF (XSIDEC(ISIDEC) .LT. XX)
                                       GO TO 19
   21        CALL SOLU ( 2, XX, V )
             CALL BSPLVD(T,KPM,XX,I,SCRTCH,MP1)
             KK =    I -    ID
             DO 25 J=1,KPM
                KK = KK + 1
                DO 25 L=1,MP1
   25              Q(ID,KK) = V(L)*SCRTCH(J,L) + Q(ID,KK)
   30        B(ID) = V(MP1+1)
       I = N
   31    IF (ISIDEC .GT. M)            RETURN
         CALL BSPLVD(T,KPM,XSIDEC(ISIDEC),I,SCRTCH,M)
         ID = ID + 1
         CALL SIDEC(SCRTCH,KPM,N,Q(ID,I-ID+1),B(ID))
                                       GO TO 31
       END


       SUBROUTINE SOLU ( MODE, XX, V )
C   INFORMATION ABOUT THE EQUATION IS DISPENSED FROM HERE
       COMMON /APPROX/ XI(100),C(2000),LXI,KPM
       COMMON /DIFEQU/ M,ISIDEC,XSIDEC(10)
       COMMON /OTHER/ ITERMX,K,RHO(19)
       DIMENSION V(20)
                                       GO TO (10,20,30),MODE
C   INITIALIZE EVERYTHING
```

```
C   I.E. SET THE ORDER *M* OF THE DIF.EQU., THE NONDECREASING SEQUENCE
C   XSIDEC(I),I=1,...,M, OF POINTS AT WHICH SIDE COND,S ARE GIVEN AND
C   ANYTHING ELSE NECESSARY.
   10 M = 2
      XSIDEC(1) = 0.
      XSIDEC(2) = 1.
C   ***PRINT OUTPUT HEADING
      PRINT 499
  499 FORMAT(37H CARRIER,S NONLINEAR PERTURB. PROBLEM)
      EPS = .5E-2
      PRINT 610, EPS
  610 FORMAT(5H EPS ,E20.10)
      FACTOR = (SQRT(2.) + SQRT(3.))**2
      S2OVEP = SQRT(2./EPS)
C   *** INITIAL GUESS FOR NEWTON ITERATION. UN(X) = X*X - 1.
      LXI = 1
      XI(1) = 0.
      DO 16 I=1,KPM
   16    C(I) = 0.
      C(1) = -1.
      C(3) = 2.
      ITERMX = 10
                                            RETURN
C
C   PROVIDE VALUE OF LEFT SIDE COEFF.S AND RIGHT SIDE AT XX
C   SPECIFICALLY, AT XX THE DIF.EQU. READS
C        V(M+1)D**M + V(M)D**(M-1) + ... + V(1)D**0  = V(M+2)
C   IN TERMS OF THE QUANTITIES V(I),I=1,...,M+2, TO BE COMPUTED HERE.
   20 CONTINUE
      V(3) = EPS
      V(2) = 0.
      UN = PPVALU(XI,C,LXI,KPM,XX,0)
      V(1) = 2.*UN
      V(4) = UN**2 + 1.
                                            RETURN
C   PROVIDE VALUE OF SOLUTION AT XX.
   30 CONTINUE
      EP1 = EXP(S2OVEP*(1.-XX))*FACTOR
      EP2 = EXP(S2OVEP*(1.+XX))*FACTOR
      V(1) = 12./(1.+EP1)**2*EP1 + 12./(1.+EP2)**2*EP2 - 1.
                                            RETURN
      END


      SUBROUTINE NEWNOT ( XI, C, LXI, K, XINEW, LXINEW, SCRTCH )
C   RETURNS LXINEW+1 KNOTS IN XINEW WHICH ARE EQUIDISTRIBUTED ON
C   (A,B) WRTO A CERTAIN MONOTONE FTN G(X). I.E.,
C        XINEW(I) = A + G**(-1)((I-1)*STEP), I=1,...,LXINEW+1,
C   WHERE  STEP = G(B)/LXINEW.  HERE,
C        G(X) = INTEGRAL OF H(Y)**(1/K) FROM A TO X,
C   A = XI(1),  B = XI(LXI+1), AND H(Y) IS A STEP FTN ON XI WHICH IS
C   PROPORTIONAL TO  ABS(D**K(F(X))) WHERE F IS THE PP FTN OF ORDER K
C   IN XI, C, LXI.  SPECIFICALLY, ON EACH SUBINTERVAL OF XI, H(X) IS
C   A WEIGHTED SUM OF THE ABS.JUMPS OF D**(K-1)(F) AT THE TWO ENDPOINTS.
C      ALSO,
C        SCRTCH(I,J)  =  D**(J-1)(G(XI(I))), J=1,2, I=1,...,LXI.
C      DIMENSION XI(LXI+1),XINEW(LXINEW+1)
      DIMENSION XI(1),C(K,LXI),XINEW(1),SCRTCH(LXI,2)
      DATA IPRINT /0/
      XINEW(1) = XI(1)
      XINEW(LXINEW+1) = XI(LXI+1)
      IF (LXI .LE. 1)                       GO TO 90
      SCRTCH(1,1) = 0.
      DIFPRV = ABS(C(K,2) - C(K,1))/(XI(3)-XI(1))
      ONEOVK = 1./FLOAT(K)
      DO 10 I=2,LXI
         DIF = ABS(C(K,I) - C(K,I-1))/(XI(I+1) - XI(I-1))
         SCRTCH(I-1,2) = (DIF + DIFPRV)**ONEOVK
         SCRTCH(I,1) = SCRTCH(I-1,1) + SCRTCH(I-1,2)*(XI(I)-XI(I-1))
```

```
   10    DIFPRV = DIF
      SCRTCH(LXI,2) = (2.*DIFPRV)**ONEOVK
      STEP = (SCRTCH(LXI,1)+SCRTCH(LXI,2)*(XI(LXI+1)-XI(LXI)))
     *                 /FLOAT(LXINEW)
      IF (IPRINT .GT. 0)
     *   PRINT 600, STEP,(I,SCRTCH(I,1),SCRTCH(I,2),I=1,LXI)
  600 FORMAT(7H STEP =,E16.7/(I5,2E16.5))
      IF (STEP .LE. 0.)                 GO TO 90
      J = 1
      DO 30 I=2,LXINEW
         STEPI = FLOAT(I-1)*STEP
   21       IF (J .EQ. LXI)                 GO TO 27
            IF (STEPI .LE. SCRTCH(J+1,1))
     *                                      GO TO 27
            J = J + 1
                                            GO TO 21
   27    IF (SCRTCH(J,2) .EQ. 0.)           GO TO 29
            XINEW(I) = XI(J) + (STEPI - SCRTCH(J,1))/SCRTCH(J,2)
                                            GO TO 30
   29       XINEW(I) = (XI(J) + XI(J+1))/2.
   30    CONTINUE
                                            RETURN
C
   90 STEP = (XI(LXI+1) - XI(1))/FLOAT(LXINEW)
      DO 93 I=2,LXINEW
   93    XINEW(I) = XI(1) + FLOAT(I-1)*STEP
                                            RETURN
      END
```

```
CARRIER'S NONLINEAR PERTURB. PROBLEM
EPS        .5000000005-02
  18 PARAMETERS
APPROXIMATION BY SPLINES OF ORDER 6 ON   4 INTERVALS. BREAKPOINTS -
    .2500000000+00      .5000000000+00      .7500000000+00
AFTER  5 ITERATIONS
    .000  -.100000+01  -.298-06  -.477-05   .401-03  -.755-02   .957-01
    .250  -.100000+01   .693-05  -.421-03   .465-01  -.130+01   .193+02
    .500  -.999944+00   .112-02  -.629-01   .676+01  -.188+03   .280+04
    .750  -.991799+00   .163+00   .452+01  -.139+02   .215+04   .609+05
     .0000000000        -.2980232239-07
     .1250000000+00     -.1490116119-07
     .2500000000+00     -.4470348358-07
     .3750000000+00     -.3278255463-06
     .5000000000+00     -.1199543476-05
     .6250000000+00     -.3650784492-04
     .7500000000+00     -.4369020462-04
     .8750000000+00      .1047752798-02
     .1000000000+01     -.1490116119-07
  18 PARAMETERS
APPROXIMATION BY SPLINES OF ORDER 6 ON   4 INTERVALS. BREAKPOINTS -
    .4414182566+00      .6527622417+00      .8313461617+00
AFTER  2 ITERATIONS
    .000  -.100000+01  -.169-06  -.340-03   .137-01  -.226+00   .151+01
    .441  -.999983+00   .338-03  -.208-02   .936+00  -.273+02   .548+03
    .653  -.998831+00   .234-01   .260+00   .315+02  -.819+03   .242+05
    .831  -.958713+00   .820+00   .199+02  -.971+02   .235+05  -.155+06
     .0000000000         .0000000000
     .1250000000+00      .7450580597-07
     .2500000000+00     -.3501772881-06
     .3750000000+00      .4619359970-06
     .5000000000+00      .3427267075-06
     .6250000000+00      .1519918442-05
     .7500000000+00     -.3835558891-04
     .8750000000+00     -.1826137304-03
     .1000000000+01     -.1490116119-07
  18 PARAMETERS
APPROXIMATION BY SPLINES OF ORDER 6 ON   4 INTERVALS. BREAKPOINTS -
    .4450281076+00      .6788925678+00      .8464950994+00
```

```
AFTER  1 ITERATIONS
    .000  -.100000+01  -.837-07  -.377-03   .149-01  -.242+00   .160+01
    .445  -.999981+00   .365-03  -.109-01   .161+01  -.461+02   .769+03
    .679  -.998029+00   .394-01   .554+00   .424+02  -.989+03   .351+05
    .846  -.944247+00   .110+01   .253+02  -.250+02   .290+05  -.254+06
     .0000000000            -.5960464478-07
     .1250000000+00          .7450580597-07
     .2500000000+00         -.3874301910-06
     .3750000000+00          .5066394806-06
     .5000000000+00          .2481043339-05
     .6250000000+00          .5312263966-05
     .7500000000+00         -.3357976675-04
     .8750000000+00         -.3033354878-03
     .1000000000+01         -.1490116119-07
```

Acknowledgments. The package was put together at Los Alamos Scientific Laboratory in June of 1971 around a subroutine (BSPLVN) written about a year earlier at Purdue University, and appeared together with some explanatory material in a technical report [2]. The version presented here differs in certain detail: The pp-repr. was enlarged to include the right endpoint of the last interval, INTERV was rewritten completely to avoid use of an ENTRY point, the calling sequence of BSPLDR (and therefore of BSPLPP) was enlarged to make the output from BSPLDR explicit, BSPLVD was streamlined, the calling sequence of BSPLVN was reordered for the sake of consistency, and the function BVALUE was added. The explanatory material was completely rewritten in response to comments by Fred Dorr and Blair Swartz.

REFERENCES

[1] C. DE BOOR, *On calculating with B-splines*, J. Approximation Theory, 6 (1972), pp. 50–62.

[2] ———, *Subroutine package for calculating with B-splines*, Technical Report LA-4728-MS, Los Alamos Scientific Laboratory, Los Alamos, NM, 1971.

[3] ———, *Good approximation by splines with variable knots*, Spline Functions and Approximation Theory, A. Meir and A. Sharma, eds., ISNM vol. 21, Birkhäuser Verlag, Basel, 1973, pp. 329–358.

[4] C. DE BOOR AND G. FIX, *Spline approximation by quasi-interpolants*, J. Approximation Theory, 8 (1973), pp. 19–45.

[5] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, this Journal, 10 (1973), pp. 582–606.

[6] M. G. COX, *The numerical evaluation of B-splines*, J. Inst. Math. Appl., 10 (1972), pp. 134–149.

[7] H. B. CURRY AND I. J. SCHOENBERG, *On spline distributions and their limits: the Pólya distributions*, Abstract, Bull. Amer. Math. Soc., 53 (1947), p. 1114.

[8] ———, *On Pólya frequency functions IV; the fundamental spline functions and their limits*, J. Analyse Math., 17 (1966), pp. 71–107.

[9] D. S. DODSON, *Optimal order approximation by polynomial spline functions*, Ph.D. thesis, Purdue Univ., Lafayette, IN, Aug. 1972.

[10] I. J. SCHOENBERG, *Contributions to the problem of approximation of equidistant data by analytic functions*, Quart. Appl. Math., 4 (1946), pp. 45–99, 112–141.

[11] I. J. SCHOENBERG AND A. WHITNEY, *On Pólya frequency functions III*, Trans. Amer. Math. Soc., 74 (1953), pp. 246–259.

[12] J. H. WILKINSON AND C. REINSCH, *Linear Algebra*, vol. II, Handbook for Automatic Computation, Band 186, Grundlehren der Mathematischen Wissenschaften, Springer-Verlag, New York, 1971.