LA-4728-MS
UC-32
ISSUED: August 1971

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

Subroutine Package for Calculating with B-Splines

by

Carl de Boor

Visiting Staff Member
Present address: Computer Science Department, Purdue University, Lafayette, Indiana 47907

SUBROUTINE PACKAGE FOR CALCULATING WITH B-SPLINES

by

Carl de Boor

ABSTRACT

Seven FORTRAN subprograms are presented for dealing with piecewise polynomial functions (of one variable) computationally. The package is built around an algorithm for the stable evaluation of B-splines of arbitrary order. Three examples illustrate what uses one might make of these routines: interpolation by splines of general order k (and not necessarily at the knots), the determination of the derivative of a spline with respect to a knot, and the approximate solution of an ordinary linear differential equation by collocation.

1. REPRESENTATIONS

In this set of subroutines, a piecewise polynomial function is represented either in terms of its local polynomial pieces (pp-repr.) or in terms of its coefficients with respect to the appropriate B-spline basis (B-repr.).

More precisely, the pp-representation for a piecewise polynomial function, s(t), consists of:

The integers K and LXI, giving the order (i.e., K - 1 is the degree) and the number of polynomial pieces, respectively;

The one-dimensional array XI(i), i = 1,...,LXI, giving the break points (in increasing order); and

The two-dimensional array C(j,i), j = 1,...,K; i = 1,..., LXI, with

$$C(i,j) = s^{(j-1)}(XI(i)),$$

the various derivatives of s at the various break points.

From this, $s^{(j)}(t)$ is found (in PPVALU) as

$$s^{(j)}(t) = \sum_{r=j}^{K-1} C(r+1,i)(t - XI(i))^{r-j}/(r-j)!$$

where i is such that

$$\begin{aligned} & i = 1 \text{ and } t < XI(2), \\ \text{or} \quad & 1 < i < LXI \text{ and } XI(i) \le t < XI(i+1), \\ \text{or} \quad & i = LXI \text{ and } XI(LXI) \le t. \end{aligned}$$

The B-representation for a piecewise polynomial function, s(t), consists of:

The integers K and N, giving the order (as a spline) and the number of linear parameters for s, respectively;

The one-dimensional array T(i), i = 1,..., N + K, containing the joints (possibly partially coincident) in increasing order; and

The one-dimensional array A(i), i = 1,..., N, containing the coefficients with respect to the B-spline basis on T.

Formally,

$$s = \sum_{i=1}^{N} A(i)\, N_{i,K} \quad , \qquad (1)$$

where

$$N_{i,r}(t) = g_r(T(i),\ldots,T(i+r);\, t)(T(i+r) - T(i))$$

is a so-called "normalized" B-spline. Here $g_r(t_i,\ldots,t_{i+r};t)$ is the r-th divided difference (in s for fixed t) of

$$g_r(s;t) = (s-t)_+^{r-1} = \begin{cases} (s-t)^{r-1} , & s > t \\ 0 \quad , & s \le t \quad . \end{cases}$$

From this, $s^{(j)}(t)$ is found (in BSPLEV) as

$$s^{(j)}(t) = \sum_{r=i-K-j+1}^{i} A(r,j+1) N_{r,K-j}(t) \quad , \qquad (2)$$

where

$$A(r,j+1) = \begin{cases} A(r) & , \; j = 0 \\ (K-j)\dfrac{A(r,j) - A(r-1,j)}{T(r+K-j) - T(r)} & , \; j > 0 \end{cases} \qquad (3)$$

(as calculated in BSPLDR), provided that

$$T(i) \le t < T(i+1) \text{ and } K \le i < N,$$

or

$$T(i) \le t \le T(i+1) \text{ and } i = N.$$

Otherwise, $s^{(j)}(t)$ is defined to be zero; i.e., s is taken to vanish identically outside the interval $[T(K), T(N+1)]$. This is done for programming convenience and agrees with the usual interpretation of (1) only if $T(1) = T(K)$ and $T(N+1) = T(N+K)$. It is also in contrast to the pp-repr. which defines s on the entire real line by extending the first and the last polynomial pieces.

Note that s and its derivatives are taken to be <u>continuous from the right</u>; i.e., if t equals a joint, then $s^{(j)}(t)$ is taken to be the number $s^{(j)}(t+)$ except when $t = T(N+1)$, the right end point, where $s^{(j)}(t) = s^{(j)}(t-)$. Similarly, the pp-repr. is interpreted as right-continuous at the breakpoints XI(i), i = 2,...,LXI. Those who prefer left-continuous functions will find it easy to modify BSPLEV (or PPVALU) accordingly right after the call to INTERV, so as to pick i such that

$$T(i) < t \le T(i+1)$$

$$(\text{or } XI(i) < t \le XI(i+1)\ ).$$

B-splines were introduced in [1], and many of their properties are discussed in [2]. Additional material on which some of these subroutines are based can be found in [3].

2. CONVERSION FROM ONE REPRESENTATION TO THE OTHER

B-repr. to pp-repr. is easily accomplished (in BSPLPP): The LXI distinct points among T(i), i = 1,...,N, are stored in XI(i), i = 1, ...,LXI, and for j = 1,...,K, the value of $s^{(j-1)}(XI(i))$ is computed (in BSPLEV) and stored in C(j,i), i = 1,...,LXI.

The conversion from pp-repr. to B-repr. is more difficult because the pp-repr. contains no <u>explicit</u> information about the defect at the break points, i.e., about the knot multiplicity at the break points necessary to represent the given function. If s(t) is known to be representable as a spline of order K with the (possibly partially coincident) knots T(i), i = 1,..., K + N, then the coefficients A(i) for the B-repr. can be calculated explicitly (see [4]) as

$$A(i) = \frac{1}{(K-1)!}\sum_{r<K} \psi_i^{(K-r-1)}(\tau)\,(-)^{K-r-1}\, s^{(r)}(\tau),$$

where τ is arbitrary, except that

$$T(i)+ \le \tau \le T(i+K)- \quad ,$$

and

$\psi_i(t) = (T(i+1) - t)(T(i+2)-t)\ldots(T(i+K-1) - t).$

If s is so representable, then τ can always be chosen to be one of the break points XI(j) so that the required derivatives of s can be read off directly from the pp-repr.

3. EXAMPLES

The subroutines are written with the idea that, in determining a piecewise polynomial 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 [3]) of the B-spline basis while determining the spline, and later exploits its piecewise polynomial character for economical evaluation.

As an example, consider the problem of determining a spline s of order k on $[\alpha,\beta]$ with simple knots $\xi_1 < \ldots < \xi_m$ in (α,β) which agrees with a given function f at the points $\tau_1<\tau_2<\ldots<\tau_N$ in $[\alpha,\beta]$, where $N = k+m$. Every spline of order k with simple knots at $\xi_1,\ldots,\xi_m$ can be written in exactly one way as

$$s = \sum_{i=1}^{N} A(i)\; N_{i,k} \quad ,$$

with $N_{i,k}$ the B-splines based on the knot set

$$T(1) = T(2) = \ldots = T(k) = \alpha \;;$$

$$T(k+i) = \xi_i, \; i = 1,\ldots,m;$$

$$T(N+1) = \ldots = T(N+k) = \beta \;.$$

Further, there exists, regardless of f, exactly one such spline that agrees with f at the points τ_i if, and only if, the τ_i's satisfy

$$N_{i,k}(\tau_i) \neq 0, \; i = 1,\ldots,N. \qquad (4)$$

Assuming that $K = k$; $M = m$; $XI(i) = \xi_i$, $i = 1,\ldots,M$; and $TAU(i) = \tau_i$, $i = 1,\ldots,N$ with $N = K+M$ to be input already, the following program fragment sets up the NxN matrix, C, and the right side, B, of the linear system

C*A = B,

whose solution vector, A, contains the coefficients A(i), i = 1,...,N of the interpolating spline. Statement 99 is an error return signalling violation of (4).

```
      IF (TAU(1) .GE. XI(1))       GO TO 99
      DO 10 I = 1,K
   10 T(I) = TAU(1)
      JPK = K
      DO 11 J = 1,M
      JPK = JPK + 1
   11 T(JPK) = XI(J)
      IF (TAU(N) .LE. XI(M))       GO TO 99
      DO 12 J = 1,K
      JPK = JPK + 1
   12 T(JPK) = TAU(N)
      KM1 = K - 1
      NP2MK = N + 2 - K
      DO 30 I = 1,N
      DO 13 J = 1,N
   13 C(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, TAU(I),ILEFT,K,1,DUMMY)
      L = ILEFT - K
      DO 16 J = 1,K
      L = L + 1
   16 C(I,L) = DUMMY(J)
      IF (C(I,I) .EQ. 0.)          GO TO 99
   30 B(I) = F(TAU(I))
```

In practice, one would make explicit use of the fact that C is a band matrix of band width 2k - 1 (or even smaller, if the τ_i's are regularly spaced with respect to the ξ_i's). This is done in the last example below. But, to complete the present example, suppose that we wish to calculate $s''(t_0)$, where s is the interpolating spline just computed and t_0 is a point between τ_1 and τ_N. If no other use of s is to be made, then it does not pay to convert to pp-repr. In any event, one might do one of the following. (At statement 50, SV contains the desired number.)

(a) Simple but costly:

```
      CALL BSPLDR(T, A, N, K, 3)
      CALL BSPLEV(T, A, N, K, TO, DUMMY, 3)
      SV = DUMMY(3)
   50 ......
```

(b) Cheaper but less simple:

```
      SV = 0.
      CALL INTERV(T(K), NP2MK, TO, ILEFT,MFLAG)
```

```
      IF (MFLAG)                  50,40,39
39    IF (TO .GT. T(N+1))         GO TO 50
      ILEFT = ILEFT - 1
40    CALL BSPLDR(T(ILEFT), A(ILEFT), K, K, 3)
      ILEFT = ILEFT + KM1
      CALL BSPLEV(T, A, N, K, TO, DUMMY, 3)
      SV = DUMMY(3)
50    ......
```

(c) Whole hog:

K,/

```
      CALL BSPLPP (T, A, N, K, XI, C, LXI)
      SV = PPVALU(XI, C, LXI, TO, 2)
50    ......
```

It is, of course, assumed above that DUMMY, XI, and C have all been dimensioned (C was used earlier), typically as sufficiently large one-dimensional arrays.

A completely different example arises in least-squares approximation by splines with variable knots. With T, A, α, β, k, and N as defined in the previous example, we wish to determine A(i), i = 1,..., N and $\alpha \le T(k+1) \le \ldots \le T(N) \le \beta$ so as to minimize

$$E(A,T) = \| f - \sum_{i=1}^{N} A(i)\, N_{i,k} \|_2^2 ,$$

with

$$\| g \|_2^2 = \langle g, g\rangle, \text{ all } g;$$

i.e., $\| \;\; \|_2$ is the norm derived from some inner product. Some methods for minimizing E require knowledge of the first partial derivatives of E with respect to each of the 2N - k variables. Write $a_i = A(i)$, $t_i = T(i)$, to simplify notation. Then, the partial derivatives with respect to the a_i's are

$$(\partial/\partial a_i)\, E = -2\langle f - \sum_j a_j N_{j,k} , N_{i,k}\rangle \quad ;$$

whereas those with respect to the t_i's are

$$(\partial/\partial t_i) E = -2\langle f - \sum_j a_j N_{j,k}, \sum_j a_j (\partial/\partial t_i) N_{j,k}\rangle \quad ,$$

and therefore require the evaluation of

$$(\partial/\partial t_i)\, N_{j,k} \quad .$$

Clearly, this last partial derivative is zero if $i < j$ or if $i > j+k$, because only the knots

$$t_j , \ldots , t_{j+k}$$

enter the formula for $N_{j,k}$:

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

$$= g_k(t_{j+1}, \ldots , t_{j+k}; t) - g_k(t_j, \ldots , t_{j+k-1}; t) \quad .$$

Here, $g_k(s\,;t) = (s - t)_+^{k-1}$. Now, from the general theory of divided differences,

$$(\partial/\partial t_i)\, f(t_0 , \ldots , t_{i-1} , t_i , t_{i+1} , \ldots , t_r)$$

$$= f(t_0 , \ldots , t_{i-1} , t_i , t_i , t_{i+1}, \ldots , t_r)$$

if f is sufficiently smooth. Hence,

$$(\partial/\partial t_i)\, N_{j,k}(t) = \begin{cases} 0 & , \; i < j \\ -g_k(t_j , t_j , t_{j+1} , \ldots , t_{j+k-1}; t) & , \; i = j \\ (t_{j+k} - t_j)\, g_k(t_j , \ldots , t_i , t_i , \ldots , t_{j+k}; t) & , \; j < i < j + k \\ g_k(t_{j+1} , \ldots , t_{j+k-1} , t_{j+k} , t_{j+k}; t) & , \; i = j + k \\ 0 & , \; j + k < i \quad . \end{cases}$$

Therefore, with $\hat{N}_{j,k}$ the B-spline based on the knot set $\hat{t}_j$ with

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

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

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

where

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

With T(K) $\le$ T(L) $<$ T(L+1) $\le$ T(N+1) and T(L) $\le$ XX $\le$ T(L+1), one might generate the k+1 (not trivially zero) numbers

$$\text{VD}(m) = (\partial/\partial t_i)\, N_{m,k}\,(\text{XX}),\ m = i-k, \ldots, i,$$

as at the top of the next column.

```
      IMK = I - K
      DO 19 JJ = IMK, I
   19 VD(JJ) = 0.
      IF(L.GE.I)                      L = L + 1
      IML = I - L
      JLOW = MAXO(1 - IML, 1) + IMK          = MAXO(L,I)-KML
      JHIGH = MINO(K - IML, K) + IMK         = MINO(L,I)
      IF (JLOW .GT. JHIGH)            GO TO 50
      DO 20 JJ = 1, I
      NPK = N + K
   20 THAT(JJ) = T(JJ)
      DO 21 JJ = I, NPK
   21 THAT(JJ + 1) = T(JJ)
      CALL BSPLVN (THAT,XX,L,K,1,DUMMY)
      KML = K - L
      KM1 = K - 1
      DO 40 J = JLOW, JHIGH
      KM1PJ = KM1 + J
      KMLPJ = KML + 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 ......
```

Finally, we include a program for experimentation with the collocation method for solving ODE's which uses, explicitly or implicitly, all of the subroutines in this package. Note, in particular, the use of BSPLVD in the generation of the matrix for the linear system to be solved (in SUBROUTINE EQUATE). This program was used to calculate the numerical example in [5].

```
      PROGRAM TESTBS(INPUT,OUTPUT)
      DIMENSION A(1000),T(100),XI(100),C(2200),TI(100)
C  THE FUNCTION TO BE APPROXIMATED IS
      SOLU(X) = 1./(1.+X)
C  TI(I),I=1,NP    COLLOCATION POINTS PER INTERVAL BETWEEN NEIGHBORING
C                  BREAKPOINTS FOR STANDARD INTERVAL (-1, 1).
C                  IT IS ASSUMED THAT TI(NP)=1, IN CASE TI(1) = -1.
C  K               ORDER OF SPLINE TO BE USED.
C  NTIMES          NO. OF DIFFERENT BREAKPOINT SETS TO BE TRIED.
C  T(1),ARIGHT     LEFT AND RIGHT BOUNDARY POINT OF INTERVAL CONSIDERED.
C                  THESE CARRY BOUND.COND. (IF ANY) SPECIFIED IN *EQUATE*
C  INTERV          USED TO SPECIFY BREAK POINTS.
C                  IF INTERV .GT. 0, INTERV-1 EQUISPACED INTERIOR BR.PTS.
C                  OTHERWISE, INTERV IS REDEFINED BY A READ, AND
C                  INTERV-1 INTERIOR BREAK POINTS ARE READ IN.
C  T(I),I=1,JOINTS BREAK POINTS INCL. BOUNDARY POINTS.
C  N               NUMBER OF EQU. TO BE SOLVED IN *BANMAT*.
    1 READ 500,NP,(TI(I),I=1,NP)
  500 FORMAT(I3/(5E15.3))
      PRINT 603,NP,(TI(I),I=1,NP)
  603 FORMAT(I3,20H COLLOCATION POINTS /(10F12.8))
      READ 501,K
  501 FORMAT(I3)
      KM1 = K-1
      READ 500,NTIMES,T(1),ARIGHT
      N = NP
      DO 60 IDUMMY = 1,NTIMES
      READ 501,INTERV
      IF (INTERV .GT. 0)          GO TO 9
      READ 500,INTERV,(T(I),I=2,INTERV)
                                  GO TO 19
```

```
    9  DX = (ARIGHT - T(1))/FLOAT(INTERV)
       DO 10 J=2,INTERV
   10  T(J) = T(J-1) + DX
   19  JOINTS = INTERV+1
       T(JOINTS) = ARIGHT
CONSTRUCT SET OF JOINTS T AND THE N COLLOCATION EQUATIONS C*X=A.  N, C,
C  AND A ARE SET IN *EQUATE*,  THE SOL.  X IS RETURNED BY *BANMAT* IN *A*.
       CALL EQUATE(T,JOINTS,K,TI,N,C,A)
       PRINT 599,N
  599  FORMAT(I5,12H PARAMETERS  )
       IF (N .LE. 0)                GO TO 60
       CALL BANMAT(N,KM1,KM1,1,1,C,N,A,N,DETERM,XI)
CONVERT TO PP-REPR.
       CALL BSPLPP(T,A,N,K,XI,C,LXI)
COMPUTE ERROR AT BREAK POINTS, LOCAL AND GLOBAL MAX. ERROR.
       CALL RESETI
       XI(LXI+1) = ARIGHT
       ERRMAX = 0.
       JLOW = 1 - K
       JHIGH = 0
       DO 40 L=1,LXI
       ERRLOC = 0.
       XX = XI(L)
       DX = (XI(L+1)-XI(L))/20.
       DO 39 M=1,21
       ERROR = ABS(SOLU(XX)-PPVALU(XI,C,LXI,K,XX,0))
       IF(M.EQ.21)    A(L)=ERROR
       IF (ERROR .GT. ERRLOC)    ERRLOC = ERROR
   39  XX = XX + DX
       IF (ERRLOC .GT. ERRMAX)    ERRMAX = ERRLOC
       JLOW = JLOW+K
       JHIGH = JHIGH+K
   40  PRINT 600,L,XI(L),ERRLOC,(C(J),J=JLOW,JHIGH)
  600  FORMAT(20XI5,F12.3,3X14HLOCAL ERROR =  E10.3/(20X10E10.3))
       PRINT 604,(A(IER),IER=1,LXI)
  604  FORMAT(17H ERROR AT KNOTS=  /(10X10E10.3))
       PRINT 602,ERRMAX
  602  FORMAT(17H MAXIMUM ERROR =  E10.3)
   60  N = NP
                                    GO TO 1
       END


       SUBROUTINE EQUATE(T,JOINTS,K,TI,N,C,A)
       DIMENSION T(1),TI(1),C(N,1),A(1),DUMMY(64),V(9)
       INTEGER TKM1
C    P0, P1, ..., PM ARE COEFFIENT FUNCTIONS OF M-TH ORDER DIFF. EQU.
C    HERE, M = IDEGRE-1.  F IS THE RIGHT SIDE FUNCTION.
C    BOUNDARY CONDITIONS ARE SPECIFIED IN THE PROGRAM AS ILLUSTRATED.
       P0(X) = -1./(1.+X)**2
       P1(X) = 1./(1.+X)
       P2(X) = 1.
       F(X) = 0.
       IDEGRE = 3
       KM1 = K-1
       TKM1 = K + KM1
       NP = N
       IF (NP .NE. K-IDEGRE+1)        GO TO 99
       LLOW = 1
       IF (TI(1) .EQ. -1.)    LLOW = 2
                                    GO TO (101,102),LLOW
  101  MULTIP = NP
       N = (JOINTS-1)*MULTIP + IDEGRE - 1
                                    GO TO 103
  102  MULTIP = NP - 1
       N = (JOINTS-1)*MULTIP + IDEGRE
  103  JOINT = JOINTS
       JOINTS = 2*K + MULTIP*(JOINTS-2)
       JJ = JOINTS
       DO 104 LL=1,K
       T(JJ) = T(JOINT)
  104  JJ = JJ - 1
```

```
      JJJ = JOINT
      DO 105 J=2,JOINT
      JJJ = JJJ - 1
      DO 105 LL=1,MULTIP
      T(JJ) = T(JJJ)
  105 JJ = JJ - 1
  106 T(JJ) = T(1)
      JJ = JJ - 1
      IF (JJ .GT. 1)                  GO TO 106
      PRINT 600,JOINTS,(T(M),M=1,JOINTS)
  600 FORMAT(I5/(10F12.5))
C
      DO 107 JJ=1,N
      DO 107 KK=1,TKM1
  107 C(JJ,KK) = 0.
      ID = 0
      I = K
      XX = T(K)
      CALL BSPLVD(T,K,XX,I,DUMMY,IDEGRE)
C  NEXT 6 CARDS SPECIFY FIRST BOUND.COND . (NOT NEC. TO HAVE ONE AT
C      BOTH ENDS.)
      ID = 1
      KK = I - ID
      DO 201 L=1,K
      KK = KK+1
  201 C(1,KK) = DUMMY(L)
      A(1) = 1.
C  THE FOLLOWING CARDS WOULD INTRODUCE A SECOND B.C. AT THIS END.
C  THE B.C. IS     2.*U - 3.*UPRIME = 25.
C     ID = ID + 1
C     KK = I - ID
C     JJ = K
C     DO 202 L=1,K
C     JJ = JJ + 1
C     KK = KK + 1
C 202 C(2,KK) = 2.*DUMMY(L) - 3.*DUMMY(JJ)
C     A(2) = 25.
C
C  SET UP EQUATIONS  ARISING FROM COLLOCATION.
      IBACK = 1
                                      GO TO (114,10),LLOW
  114 IBACK = 2
      DO 119 I=K,N,MULTIP
      XM = (T(I+1) + T(I))/2.
      DX = (T(I+1) - T(I))/2.
      DO 119 LL=LLOW,NP
      XX = XM + DX*TI(LL)
      CALL BSPLVD(T,K,XX,I,DUMMY,IDEGRE)
                                      GO TO 10
  118 CONTINUE
  119 CONTINUE
      I = N
      XX = T(JOINTS)
      CALL BSPLVD(T,K,XX,I,DUMMY,IDEGRE)
C  THE NEXT 6 CARDS SET UP B.C.  U = .5 AT RIGHT END POINT
C   ADDITIONAL COND. COULD BE SPECIFIED FOLLOWING THIS AND EARLIER MODEL
      ID = ID+1
      KK = I - ID
      DO 401 L=1,K
      KK = KK+1
  401 C(ID,KK) = DUMMY(L)
      A(ID) = .5
                                      GO TO 30
   10 V(1) = P0(XX)
      V(2) = P1(XX)
      V(3) = P2(XX)
      ID = ID+1
      KK = I - ID
      DO 15 J=1,K
      JJ = J
      KK = KK+1
```

```
      DO 15 L=1,IDEGRE
      C(ID,KK) = C(ID,KK) + V(L)*DUMMY(JJ)
15    JJ = JJ + K
      A(ID) = F(XX)
                                GO TO (114,118),IBACK
99    N = 0
30                              RETURN
      END
```

4. SPECIFIC DESCRIPTION OF THE ROUTINES

Internal DIMENSION statements (in BSPLEV, and, more importantly, in BSPLVD) arbitrarily restrict the order to $K \le 20$.

a. <u>SUBROUTINE BSPLDR (T, A, N, K, NDERIV)</u>. The array A is assumed to have as its first N entries the coefficients of some s(t) with respect to the B-spline basis on the knot set T(i), i = 1,...,N + K. Treating A as a two-dimensional array with columns length N, the routine generates the numbers A(i,j), i = j,...,N; j = 2,...,NDERIV, which are needed in BSPLEV for the calculation of $s^{(j)}(t)$ for j<NDERIV.

It is a waste of time (though not fatal) to have NDERIV > K or NDERIV < 2.

b. <u>SUBROUTINE BSPLEV (T, A, N, K, X, SVALUE, NDERIV)</u>. With s(t) the function whose B-repr. is contained in T, A, N, K, the routine calculates $s^{(j-1)}(X)$ and stores it in SVALUE(j), j = 1,...,NDERIV. If NDERIV > 1, then it is <u>assumed</u> that a

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

has been executed at least once before the call to BSPLEV.

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

$$K \le I \le N \text{ and } T(I) \le X < T(I+1)$$

$$(\text{or } T(I) < X \le T(I+1), \text{ if } T(I+1) = T(N+1)).$$

If no such I exists, SVALUE (j) is set to zero, j = 1,..., NDERIV.

The routine also uses BSPLVN.

c. <u>SUBROUTINE BSPLPP (T, A, N, K, XI, C, LXI)</u>. This routine converts the B-repr. contained in T,A, N, K into the pp-repr., storing it in XI, C, LXI,K. The routine uses BSPLDR and BSPLEV.

d. <u>SUBROUTINE BSPLVD (T,K,X, ILEFT, VNIKX,NDERIV)</u>. This subroutine is of help in the efficient construction of a system of equations to determine the B-repr. for s(t) from information about its value and its derivatives. The routine generates the value at t = X of all $N_{i,K}(t)$ and their first NDERIV-1 derivatives which are not trivially zero at X.

Specifically, the routine returns the numbers

$$\text{VNIKX}(i,M) = N^{(M-1)}_{i+\text{ILEFT}-K,K}(X) \quad ,$$

$$i = 1,\dots,K;\ M = 1,\dots,\text{NDERIV} \quad .$$

VNIKX is taken to be a two-dimensional array with column length K. It is <u>assumed</u> that ILEFT is such that both

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

and

$$T(\text{ILEFT}) \le X \le T(\text{ILEFT}+1) \quad .$$

The routine uses BSPLVN.

e. <u>SUBROUTINE BSPLVN(T,X,ILEFT,JHIGH,INDEX,VNIKX)</u>. This routine incorporates the second algorithm of [3] for the stable evaluation of B-splines. The routine returns, in the one-dimensional array VNIKX, the numbers

$$\text{VNIKX}(i) = N_{\text{ILEFT}-J+i,J}(X), \quad i = 1,\dots,J \quad , \qquad (5)$$

where the value of the integer J depends on JHIGH and INDEX:

if INDEX = 1, then J = JHIGH;

if INDEX = 2, and, on entering, J = m , then J = max (JHIGH, m + 1).

The second possibility is useful in the efficient evaluation of a spline and its derivatives (as in BSPLEV and BSPLVD). If INDEX = 2, then VNIKX is assumed to contain, on entering, the numbers described in (5) with J as it is on entering. Further, ILEFT is assumed to be such that both

$$T(ILEFT) < T(ILEFT+1)$$

and

$$T(ILEFT) \le X \le T(ILEFT+1) \quad .$$

~~not T(ILEFT)≠~~

will

Division by zero ~~may~~ result if this last assumption is not satisfied. T(ILEFT) = T(ILEFT+1)

~~Change to accept ILEFT, right ...~~ Cut out ENTRY RESETI

f. <u>SUBROUTINE INTERV(XI,LXI,X,ILEFT,MFLAG).</u> This subroutine assumes that XI is a one-dimensional array of length LXI containing a nondecreasing sequence of real numbers. It returns integers ILEFT and MFLAG as follows:

if		ILEFT	MFLAG
	X < XI(1)	1	-1
	XI(I) ≤ X and X < XI(I+1)	I	0
	XI(LXI) ≤ X	LXI	1

The program starts the search for ILEFT with the value of ILEFT that was returned at the previous call (and was saved in the local variable I) to minimize the work in the common case that this call's X is close to the previous call's X. Should this assumption not be valid, then the program locates I and IHIGH such that

$$XI(I) < X < XI(IHIGH)$$

and, once they are found, uses bisection (on the function f(i) = XI(i) - X) to find the correct value for ILEFT.

The local variable I is initialized to the value 1. ~~Once the routine is used in a program with a specific array XI, the statement~~

~~CALL RESETI~~

~~should be executed before using INTERV with a different array; this resets I to the value 1. RESETI is an ENTRY point to INTERV.~~

stet!

g. <u>FUNCTION PPVALU(XI,C,LXI,K,X,IDERIV).</u> This function returns the value of $s^{(IDERIV)}(X)$, where s(t) is the piecewise polynomial function whose pp-repr. is contained in XI, C, LXI, K.

The routine uses INTERV.

5. FORTRAN LISTING OF THE SUBROUTINES

FORTRAN decks are available upon request, as of the date of distribution of this report, from Group C-6 at Los Alamos or from the author at the Computer Science Department, Purdue University.

```
      SUBROUTINE BSPLDR(T,A,N,K,NDERIV)
C  TO DIFFERENCE B-SPLINE COEFFICIENTS PREPARATORY TO DERIV.CALC.
      DIMENSION T(1),A(N,1)          NDERIV/
      KMID = K
      DO 20 IDERIV=2,NDERIV
      KMID = KMID - 1
      FKMID = FLOAT(KMID)
      DO 20 I=IDERIV,N
      IPKMID = I + KMID
      DIFF = T(IPKMID) - T(I)
   20 IF (DIFF .GT. 0.)
     * A(I,IDERIV) = (A(I,IDERIV-1) - A(I-1,IDERIV-1))/DIFF*FKMID
                                    RETURN
      END
```

MAY BE END WITH A CONTINUE

NDERiV

```
      SUBROUTINE BSPLEV(T,A,N,K,X,SVALUE,NDERIV)
 CALCULATES VALUE OF SPLINE AND ITS DERIVATIVES AT *X* FROM B-REPR.
      DIMENSION T(1),A(N,1),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 .LE. K)                GO TO 99
      I = I-1
      IF (X .EQ. T(I))            GO TO 10
   20 KP1MN = K+1-NDERIV
      CALL BSPLVN(T,X,I,KP1MN,1,VNIKX)
      IDERIV = NDERIV
   21 LEFT = I - KP1MN
      DO 22 L=1,KP1MN
      LEFTPL = LEFT+L
   22 SVALUE(IDERIV) = SVALUE(IDERIV) + VNIKX(L)*A(LEFTPL,IDERIV)
      IF (IDERIV .LE. 1)          GO TO 99
      IDERIV = IDERIV - 1
      KP1MN = KP1MN + 1
      CALL BSPLVN(T,X,I,0,2,VNIKX)
                                   GO TO 21
   99                              RETURN
      END
```

K/

```
      SUBROUTINE BSPLPP(T,A,N,K,XI,C,LXI)
 CONVERTS B-SPLINE REPRESENTATION TO PIECEWISE POLYNOMIAL REPRESENTATION.
      DIMENSION T(1),A(N,1),XI(1),C(K,1)
      CALL BSPLDR(T,A,N,K,K)
      LXI = 0
      DO 50 ILEFT=K,N
      IF (T(ILEFT+1) .EQ. T(ILEFT))GO TO 50
      LXI = LXI + 1
      XI(LXI) = T(ILEFT)
      CALL BSPLEV(T,A,N,K,XI(LXI),C(1,LXI),K)
   50 CONTINUE
                                   RETURN
      END
```

```
      SUBROUTINE BSPLVD(T,K,X,ILEFT,VNIKX,NDERIV)
 CALCULATES VALUE AND DERIV.S OF ALL B-SPLINES WHICH DO NOT VANISH AT *X*.
      DIMENSION T(1),VNIKX(K,K),A(20,20),V(20)
      CALL BSPLVN(T,X,ILEFT,K+1-NDERIV,1,VNIKX(NDERIV,NDERIV))
      IF (NDERIV .LE. 1)          RETURN
      DO 10 I=1,K
      DO 9 J=1,K
    9 A(I,J) = 0.
   10 A(I,I) = 1.
      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,X,ILEFT,0,2,VNIKX(IDERIV,IDERIV))
   15 CONTINUE
C
      KMD = K
      DO 40 M=2,NDERIV
      KMD = KMD-1
      FKMD = FLOAT(KMD)
      I = ILEFT
      J = K
   19 JM1 = J-1
```

~~Consider returning with rows + columns reversed.~~

Better! Rewrite BAN TO HAVE C(i,j) = i th element in row j

OLD NTRY

```
      IPKMD = I + KMD
      DIFF = T(IPKMD) - T(I)
      IF (J .LE. 1)            GO TO 22
      IF (DIFF .EQ. 0.)        GO TO 21
      DO 20 L=1,J
   20 A(L,J) = (A(L,J) - A(L,J-1))/DIFF*FKMD
   21 J = JM1
      I = I - 1
                                 GO TO 19

   22 IF (DIFF .EQ. 0.)        GO TO 30
      A(1,1) = A(1,1)/DIFF*FKMD
C        40
   30 DO 35 I=1,K
      V(I) = 0.
      JLOW = MAX0(I,M)
      DO 35 J=JLOW,K
   35 V(I) = V(I) + A(I,J)*VNIKX(J,M)
      DO 40 I=1,K
   40 VNIKX(I,M) = V(I)
                                 RETURN
      END
```

NO NEED TO DIMENSION V

```
      SUBROUTINE BSPLVN(T,X,ILEFT,JHIGH,INDEX,VNIKX)
CALCULATES THE VALUE AT *X* OF ALL B-SPLINES OF ORDER *JHIGH*
C  ON *T* WHICH DO NOT TRIVIALLY VANISH AT *X* .
      DIMENSION T(1),VNIKX(1)
      DIMENSION DELTAM(20),DELTAP(20)
      DATA J/1/,(DELTAM(I),I=1,20),(DELTAP(I),I=1,20)/40*0./
                                 GO TO (10,20),INDEX
   10 J = 1
      VNIKX(1) = 1.
                                 GO TO 29
   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
   29 IF (J .LT. JHIGH)        GO TO 20
                                 RETURN
      END


      SUBROUTINE INTERV(XI,LXI,X,ILEFT,MFLAG)
COMPUTES LARGEST I (1 .LE. I .LT. LXI) SUCH THAT XI(I) .LE. X.
C  PROGRAM RETURNS THIS I IN *ILEFT* WITH  *MFLAG* = 0
C  IF X .LT. XI(1) ,  THEN *ILEFT* = 1, *MFLAG* = -1
C  IF X .GE. XI(LXI),  THEN *ILEFT* = LXI, *MFLAG* = 1
      DIMENSION XI(1)
      DATA I,IHIGH / 1,2/
      MFLAG = 0
    7 IF (X - XI(I))           20,10,10
    8 I = IHIGH
    9 IHIGH = I+1
   10 IF (I .LT. LXI)          GO TO 11
      MFLAG = 1
                                 GO TO 40
   11 IF (X - XI(IHIGH))       40, 8,12
   12 IHIGH = IHIGH + IHIGH - I
      IF (IHIGH - LXI)         13,15,14
   13 IF (X - XI(IHIGH))       30, 8,12
```

~~IF(I.LE.LXI) GOTO 7~~
I = LXI
IHIGH = I+1

~~MERGE~~ STET

```
14    IHIGH = LXI
15    IF (X - XI(LXI))           30, 8, 8
20    IF (I .EQ. 1)             GO TO 39
21    I = I + I - IHIGH
      IF (I-1)                   23,24,22
22    IF (X - XI(I))             21, 9,31
23    I = 1
24    IF (X - XI(1))             38, 9,31
30    I = (I+IHIGH)/2
                                  GO TO 32
31    IHIGH = (I+IHIGH)/2

32    MIDDLE = (I + IHIGH)/2
      IF (MIDDLE - I)             40,40,33
33    IF (X - XI(MIDDLE))         34,36,35
34    IHIGH = MIDDLE
                                  GO TO 32
35    I = MIDDLE
                                  GO TO 32
36    I = MIDDLE
                                  GO TO 9
38    IHIGH = I+1
39    MFLAG = -1
40    ILEFT = I
                                  RETURN
```

Stet

```
      ENTRY RESET
      I = 1
      IHIGH = 2
                                  RETURN
      END
```

```
      FUNCTION PPVALU(XI,C,LXI,K,X,IDERIV)
C     CALCULATES VALUE AT *X* OF *IDERIV*-TH DERIVATIVE OF SPLINE FROM PP-REPR.
      DIMENSION XI(1),C(K,1)
      CALL INTERV(XI,LXI,X,I,NDUMMY)
      DX = X - XI(I)
      PPVALU = 0.
      FLOATK = K - IDERIV
      J = K
                                   GO TO 2
 1    PPVALU = PPVALU/FLOATK*DX + C(J,I)
      J = J-1
      FLOATK = FLOATK - 1.
 2    IF (FLOATK .GT. 0.)         GO TO 1
                                   RETURN
      END
```

REFERENCES

1. H. B. Curry and I. J. Schoenberg, On spline distributions and their limits: the Polya distributions, Abstr., Bull. Amer. Math. Soc. 53 (1947), 1114.

2. H. B. Curry and I. J. Schoenberg, On Polya frequency functions IV: the fundamental spline functions and their limits, J. Analyse Math. 17 (1966), 71-107.

3. C. de Boor, On calculating with B-splines, J. Approximation Theory, to appear.

4. C. de Boor and G. Fix, Spline approximation by quasi-interpolants, J. Approximation Theory, to appear.

5. C. de Boor and B. Swartz, Collocation at Gaussian points, to appear.