Using Available Curve-Shaped Information with a Non-Uniform B-Spline

The B-spline is a commonly used tool for solving problems in computer-aided design. A procedure for describing the curve shape and discontinuity conditions at input coordinate locations is outlined for non-uniform B-splines.

Martin Roche
Robotics Assembly Group
National Bureau of Standards
Gaithersburg, MD

Sheri Beusan
Grumman Aerospace Corporation
Bethpage, NY

Splines have been used to interpolate curve data for many years in various industries. The B-spline, with its defining polygon, has become extremely popular, because of the increased use of interactive graphics [1-4]. Instead of a strict interpolation procedure, many engineers using B-spline curves in shape design force the user of the software to begin by defining or sketching the so-called defining polygon. This polygon determines a B-spline curve. Once this curve is generated along with its defining polygon, the designer can interactively modify the defining polygon to obtain a locally modified B-spline curve of desired shape.

Non-Uniform B-Spline Input Interpolation Procedure

The procedure examined here begins not by inputting the defining polygon; rather it uses known or available curve shape or property information. The particular way this existing information should be input is very dependent upon the particular facility, hardware availability, and associated software.

This information should be handled a certain way once input to the computer. An overview of the procedure for an interpolation for a first approximation by a non-uniform B-spline curve (the parameterization is not from zero to one for each segment) is called for. Existing mathematical routines are mentioned and an effective way to organize the information to define curves using these splines is outlined, in the context of transferring the information needed to generate the desired curve shape.

The types of conditions desired at given input coordinate points are outlined, and mathematical concepts and required algorithms are discussed. Lastly, ways in which the input information should be processed so that the algorithms might be applied are described.

Imposing conditions at the input points. There is nothing sacred about the number of or nature of possible conditions. They can be easily changed depending on the particular application or flexibility one wishes the curve form to have. The option information will be transferred through a code procedure (an integer array will carry the information as to which option has been selected for each point). The use of a code to transmit curve property information has been suggested by others [3, 5], but details of an implementation have not been included.

For simplicity of discussion, we restrict our attention to the case of a B-spline curve of order 4 (degree 3) and of

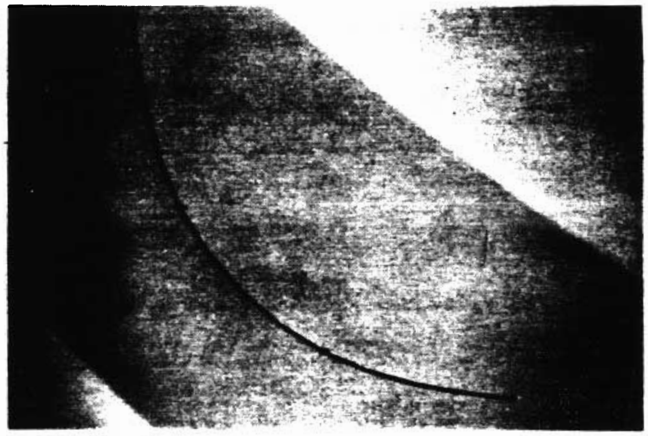

Fig. 1 A B-spline curve with all input coordinate points having continuous second derivatives.

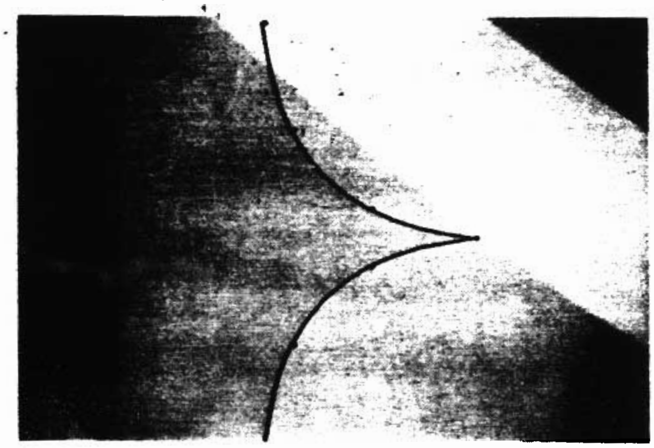

Fig. 2 A B-spline curve that has a tangent discontinuity at a point.

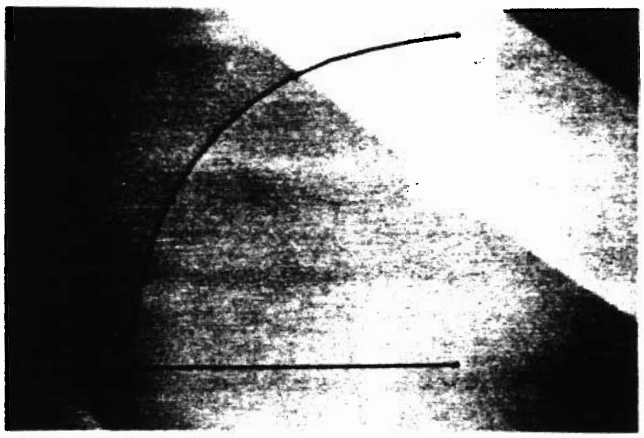

Fig. 3 A B-spline curve, with the fourth displayed point from the left being the start of a straight line segment. At this point there is a curvature discontinuity.

Fig. 4 A B-spline curve defined from the top right point to the bottom right point. At the fourth point a straight line starts; there is a tangent discontinuity at this point.

dimension 2. We are considering the construction of a parametric cubic spline in the $x-y$ plane. We first assume that we will be supplied with a sequence of coordinate points $[(x(1),y(1)), (x(2), y(2)), \ldots ,(x(m),y(m))]$ that we wish the B-spline curve to interpolate in the given order with the increasing parameter value.

At each of these points, a number of properties can be specified that will determine the character of the curve in the vicinity of the point. The number of available options for first and last points will be limited as compared to the other points. We will first list the options that are available for points other than the two end points:

(1) The given point in the string of input coordinate points has a continuous second derivative. This is the standard spline condition (Figure 1).
(2) The given point will locate a tangent discontinuity (Figure 2).
(3) The given point is the start of a straight line segment. At this point there will be a curvature discontinuity (Figure 3).
(4) The given point is the start of a straight line segment. At this point there will be a tangent discontinuity (Figure 4).
(5) The given point is the end point of a straight line segment. A curvature discontinuity exists at the point (Figure 5).
(6) The given point is the end point of a straight line segment and a tangent discontinuity exists at this point (Figure 6).
(7) The given point is the end point of a straight line segment and the start of another (Figure 7).

To accomplish the information transfer, we used an integer array, NCODE(I), $I=1, \ldots ,m$, and for each of the I-points (I not equal to 1 or m) we assigned one of the values 1 through 7. The default for each point was 1; if the user wished a different condition he/she could make the appropriate change.

At the first or last point of the sequence, two conditions are allowed. Either the second derivative is taken to be zero—a commonly used approach—or it is the start of a straight line segment. The zero second derivative is the default condition and the NCODE( ) value is 1. If the first point is the start of straight line segment, NCODE(1) is given the value 3, and if the last point is the end of a straight line segment, NCODE($n+1$) is given the value of 5.

Underlying mathematical problem. The underlying mathematics for B-spline interpolation of arbitrary order for non-uniform spacing of the knots has been well formulated in the literature [5,6]. How the knots, data points, and other interpolation information is related to the defining polygon is

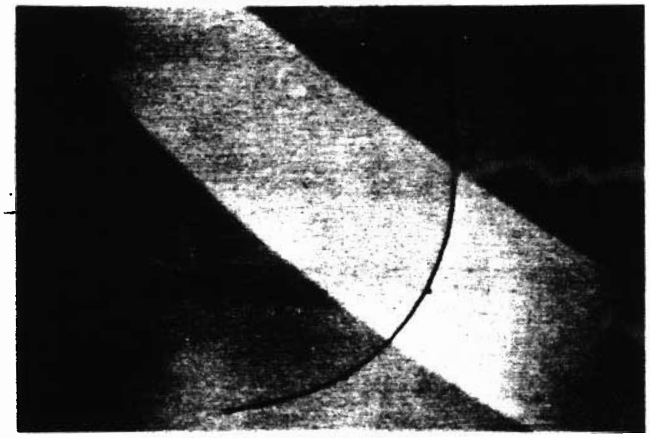

Fig. 5 A B-spline curve that starts with a straight line and blends into a curved segment. At the transition point there is a curvature discontinuity.

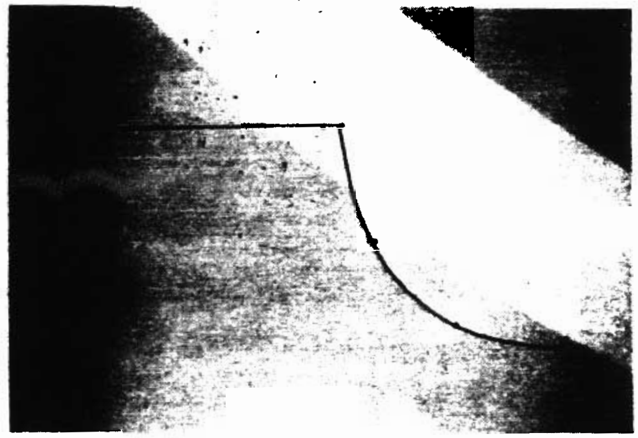

Fig. 6 A B-spline curve that starts with a straight line segment and ends with a general curved segment. At the transition point there is a tangent discontinuity.

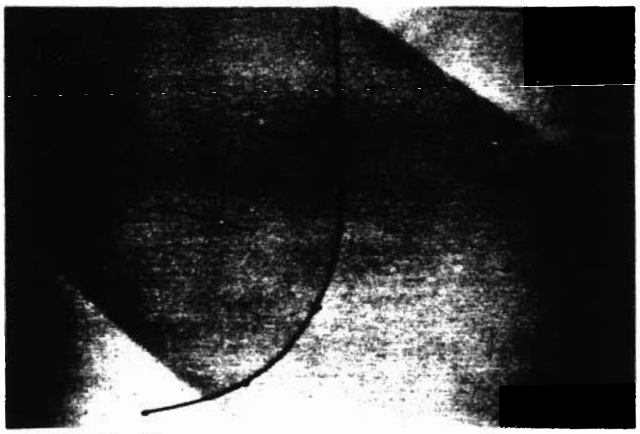

Fig. 7 A B-spline curve with two adjacent distinct line segments. The start of one line segment is the end of the other.

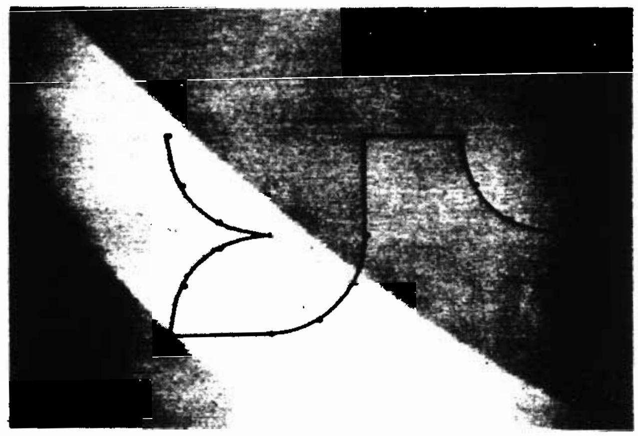

Fig. 8 A B-spline curve that is a composite of the seven curves of Figures 1-7.

Table I

First Input Coordinate Point (in two parts).

NCODE(1) = 1

e	$ND(e)$	$p(e)$	$g(e)=(gx(e),gy(e))$		j	$T(j)$
1	+3	0.0	0.0,	0.0	1	0.0
2	+1	0.0	$x(1)$,	$y(1)$	2	0.0
					3	0.0
					4	0.0

NCODE(1) = 3

e	$ND(e)$	$p(e)$	$g(e)=(gx(e),gy(e))$		j	$T(j)$
1	+1	0.0	$x(1)$,	$y(1)$,	1	0.0
2	+1	$P(1,1)$	$X(1,1)$	$Y(1,1)$	2	0.0
3	+1	$P(1,2)$	$X(1,2)$,	$Y(1,2)$	3	0.0
					4	0.0

Table II

The i-th Input Coordinate Point for $(1 < i < m)$. Up to this point assume, that L interpolation values and N knot values have been determined. This table is in seven parts, one for each of the possible NCODE(i) values.

e	$ND(e)$	$p(e)$	$g(e)=(gx(e),gy(e))$		j	$T(j)$
NCODE(i) = 1						
$L+1$	+1	$s(i)$	$x(i)$,	$y(i)$	$N+1$	$s(i)$
NCODE(i) = 2						
$L+1$	−3	$s(i)$	0.0,	0.0	$N+1$	$s(i)$
$L+2$	+1	$s(i)$	$x(i)$,	$y(i)$	$N+2$	$s(i)$
$L+3$	+3	$s(i)$	0.0,	0.0	$N+3$	$s(i)$
NCODE(i) = 3						
$L+1$	+1	$s(i)$	$x(i)$,	$y(i)$	$N+1$	$s(i)$
$L+2$	+1	$P(i,1)$	$X(i,1)$,	$Y(i,1)$	$N+2$	$s(i)$
$L+3$	+1	$P(i,2)$	$X(i,2)$,	$Y(i,2)$		
NCODE(i) = 4						
$L+1$	−3	$s(i)$	0.0,	0.0	$N+1$	$s(i)$
$L+2$	+1	$s(i)$	$x(i)$,	$y(i)$	$N+2$	$s(i)$
$L+3$	+1	$P(i,1)$	$X(i,1)$,	$Y(i,1)$	$N+3$	$s(i)$
$L+4$	+1	$P(i,2)$	$X(i,2)$,	$Y(i,2)$		
NCODE(i) = 5						
					$N+1$	$s(i)$
$L+1$	+1	$t(i)$	$x(i)$,	$y(i)$	$N+2$	$s(i)$
NCODE(i) = 6						
$L+1$	+1	$s(i)$	$x(i)$,	$y(i)$	$N+1$	$s(i)$
$L+2$	+3	$s(i)$	0.0,	0.0	$N+2$	$s(i)$
					$N+3$	$s(i)$
NCODE(i) = 7						
$L+1$	+1	$s(i)$	$x(i)$,	$y(i)$	$N+1$	$s(i)$
$L+2$	+1	$P(i,1)$	$X(i,1)$,	$Y(i,1)$	$N+2$	$s(i)$
$L+3$	+1	$P(i,2)$	$X(i,2)$,	$Y(i,2)$	$N+3$	$s(i)$

explained. It is shown that the defining polygon can be determined by solving a linear system of equations.

The knot sequence $T = T(i)_{i=1}^{n+k}$, contains real numbers such that $k<n$, $T(i)<T(i+1)$, and $T(i)<T(i+k)$ for all i, and has a corresponding sequence of B-splines of order k, $(B)_{i=1}^{n}$. The spline S_k is a vector

$$S_k^{(t)} = \sum_{j=l}^{n} b_j B_j(t),\ T(k)^+ \leq t \leq T(n+1)^-$$

where each $bj = (bx_j, by_j)$ is a point of the defining sequence for the defining polygon.

To solve for the defining points of the defining polygon n interpolation must be given. This information could be coordinate information or derivative information at a particular parameter value t. In fact, when $t=t_i$, for some i, it may be necessary to distinguish whether $t=T(i)^-$ or $t=T(i)^+$ because of derivative discontinuities (see Figure 2).

In our implementation, we followed Butterfield's suggestion [5] and incorporated this type of information with that of the derivatives in an array $ND(i)$. In particular, if $(P_i)_{i=1}^{n}$ are the parameter values where the spline has specified derivative values $g(i)=(gx(i),gy(i))$, $i=1,\ldots n$, then $ND(i)$ is a nonzero integer such that $(|ND(i)|-1)$ is order of the derivative at

$$t_i = \begin{cases} p_{i^-} & \text{if } ND(i)>0 \\ p_{i^-} & \text{if } ND(i)<0,\ i=1,\ldots,n \end{cases}$$

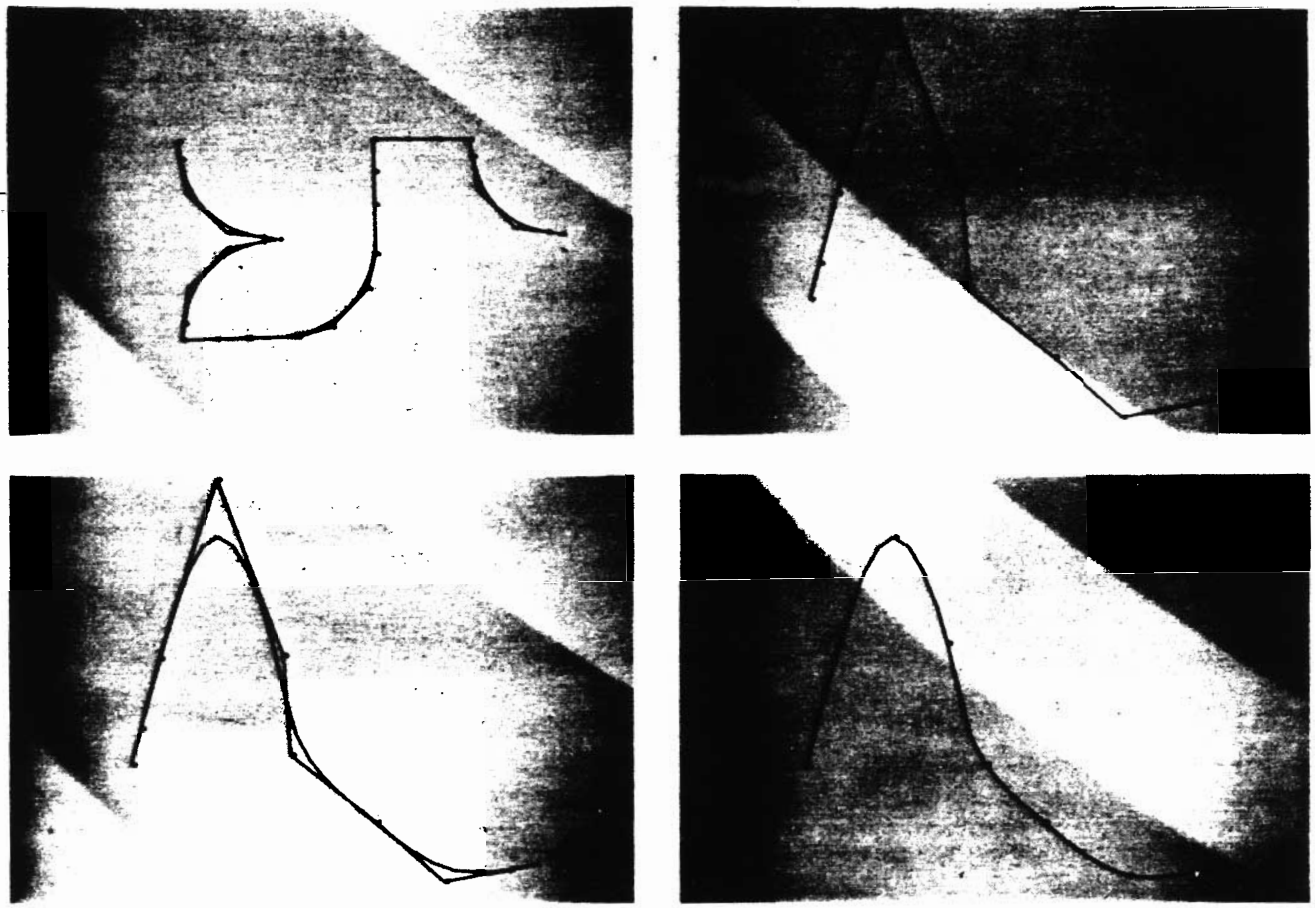

Fig. 9 The B-spline curve of Figure 8 with its defining polygon displayed. **(a)**A B-spline curve defining polygon; **(b)**the defining polygon of **(a)** along with the B-spline curve; **(c)**the B-spline curve generated by the above defining polygon.

Table III

Last Input Coordinate Point (in two parts).

NCODE(m+1) = 1

e	$ND(e)$	$p(e)$	$g(e)=(gx(e),gy(e))$		j	$T(j)$
$n-1$	-1	$s(m)$	$x(m)$,	$y(m)$	$n+1$	$s(m)$
n	-3	$s(m)$	0.0,	0.0	$n+2$	$s(m)$
					$n+3$	$s(m)$
					$n+4$	$s(m)$

NCODE(m+1) = 5

e	$ND(e)$	$p(e)$	$g(e)=(gx(e),gy(e))$		k	$T(j)$
n	-1	$s(m)$	$x(m)$,	$y(m)$	$n+1$	$s(m)$
					$n+2$	$s(m)$
					$n+3$	$s(m)$
					$n+4$	$s(m)$

Appendix

Listing of a Subroutine that Loads the Matrix A

Subroutine bemat (n, k, x, t, nd, A, nr, integs, nblock, io)

```
c
c
c      bemat—b spline matrix A construction
c            The construction of the Matrix A, stored in the condensed
c            form for the interpolation problem.
c
c
c
c
c      The equation defining the b-spline interpolation problems may be
c      written in matrix form as
c            A*b = f,
c      where A is n × n matrix, b is n × ns matrix and f is a
c      n × ns matrix (ns being the dimension of the space, i.e., 2-dim,
c      3-dim, 4-dim).
c
c      Routines called: bsplvd (which calls bsplvb)
c            NOTE:
c               a. Routine bsplvd can be found in reference 6 on pg. 288.
c               b. Routine bsplvb can be found in reference 6 on pg. 134.
c
c      Inputs: n, k, x, t, nd, io
c         n = no. of interpolation points
c
c         k = order of the b-spline
c
c         x(1), . . ., x(n+k) are the knots which the b-splines are defined in
c         terms of. For the b-curve the independent parameter x is such that
c         x(k)    .le x
c                 .le x (n+1)
c
c         t(1), . . ., t(n) are the values at which specified derivative values
c         are given.
c
c         nd(1), . . . nd(n) are non-zero integers giving the order of the
c         derivatives at the t(i) values. The order of the derivative at t(i)
c         is given by abs(nd(i)) − 1 and the value of x
c         at which interpolation is required by
c            pi = t(i) plus if nd(i)   .gt 0
c            pi = t(i) minus if nd(i)   .lt 0
c
c
c
c
c      Outputs: A, nr(  ), integs (i,j)
c         A: is a n × n matrix in condensed form.
c            It is stored as a one-dimensional array.
c
c         nr(  ): an array which contains the numbers of the last rows of
c         non-zero elements in the ith column of a, i.e., for column i the
c         number nr(i) is the last row which has a non-zero value for this
c         column.
c
c         integs(,): integer array description of the block structure of
c                    matrix A.
c         integs(1,i) = no. of rows of block i = nrow
c         integs(2,i) = no. of columns of block i = ncol
c         integs(3,i) = no. of elim. steps in block i = last
c            i = 1,2 . . .,nbloks
c
c         The linear system is of order
c            or = sum (integs(3,i), i=1, . . .,nbloks),
c         but the total no. of rows in the blocks is
c            nbrows = sum(integs(1,i); i=1, . . .,nbloks)
c
c               Note: If j = (i−1)*k then,
c               a(j+1), . . .,a(j+k) is the ith
c               row of the condensed form for the interpolation problem.
c
c
```

The restrictions on the p_i's are:

$$T(k)^+ \le p_i \le T(n+1)^-,$$

$$T(i)^+ \le p_i \le T(i+k)^-,$$

and if

$$T(j)^+ \le p_i \le T(j+1)^-, \text{ then } t_j^+ \le p_{i+1}$$

the defining coordinate points are determined by solving the system

$$\sum_{j=1}^{n} b_j B_j^{(l_i)}(t_i) = g(i) = (gx(i), gy(i)), \; i=1, \ldots, n,$$

where $l_i =$ $l_i = (|ND(i)|-1)$ and $B_j^{(m)}$ denotes the mth derivative of B_j. This can be written in matrix form as

$AB = G$,

where A is a $n \times n$ matrix, B is a n-row matrix of the defining coordinate values to be determined, and G contains the interpolation information. (We coded Butterfield's algorithm to determine A and then solved this system using routines supplied by deBoor [6].)

Using the input information. The input coordinate arrays $x(i)$, $y(i)$, $i=1, \ldots, m$ are supplied. The knot sequence will be provided by Tables I–III. To facilitate construction of these tables, the following notation is introduced:

$s(1) = 0.0$ and

$s(i) = s(i-1)+\mathrm{SQRT}((x(i)-x(i-1))**2+(y(i)-y(i-1))**2)$,

for $i=2, \ldots, m$. Also for the ith input coordinate point set:

$$P(i,1) = s(i)+(s(i+1)-s(i))/3$$
$$P(i,2) = s(i)+2(s(i+1)-s(i))/3$$
$$X(i,1) = x(i)+(x(i+1)-x(i))/3$$
$$X(i,2) = x(i)+2(x(i+1)-x(i))/3$$
$$Y(i,1) = y(i)+(y(i+1)-y(i))/3$$
$$Y(i,2) = y(i)+2(y(i+1)-y(i))/3$$

The NCODE values determine how many duplicate knot values will be added to the knot sequence for a particular input coordinate point (except for the first and last point) and parameter values and numerical function values for the function evaluation. Three tables have been prepared for the possible cases. One table is the first point, one table is for any point between the first and last point, and one table is for the last point.

```
c       dimension nd(1),t(1),x(1),nr(50),A(400),integs(3,50)
        dimension dn(16)
        real aaa(4,4)
c
c     initialize index for b-spline basis
c
        jo = k
        nblock = 0
        nrows = 0
c
c
c     set loop index i to parameter index (row no. of matrix)
c
c
        do 100 i=1,n
        nrows = nrows + 1
c
c     set bounds for knot interval search
c
c
        l = max(k,i,jo)
        iu = min(i+k-1,n)
        if(nd(i)   .lt.  0)go to 25
        if(nd(i)   .eq.  0)go to 200
        if(t(i)    .lt.  x(1))go to 200
        do 20 j=1,iu
        if(t(i)    .lt.  x(j+1))go to 35
20      continue
        go to 200
25      if(t(i)    .le.  x(1))go to 200
        do 30 j=1,iu
        if(t(i)    .le.  x(j+1)go to 35
30      continue
        go to 200
35      if(j   .eq.  jo)go to 50
c
c
c     set nr(i), the last row in which a non-zero element can exist in the
c     ith column of a
c
c     integs(i,j)values will be loaded.
c
c
        nblock = nblock + 1
        integs(1,nblock) = nrows - 1
        integs(2,nblock) = k
        integs(3,nblock) = j - jo
        nrows = 1
        j1 = j - 1
        do 40 is = jo,j1
40      nr(is-k+1) = i-1
        jo = j
c
c
c     For the determination of the derivative of the b-spline
c     bases n(j,k), see algorithm bsplvd (which calls bsplvb)
c
c
50      continue
        il = j
        do 51 ia=1,3
        if(x(il)   .ne.   x(il+1)   )go to 53
        il = il + 1
51      continue
53      ti = t(i)
        nderiv = iabs(nd(i))
        call bsplvd(x,k,ti,il,aaa,dn,nderiv)
c
c
c     load the a matrix with the appropriate nd values
c
c
        ns = k*(nderiv -1)
        nf = (i-1)*k
        do 52 js =1,k
52      a(nf + js) = dn(ns + js)
100   continue
c
c
        jk1 = jo -k +1
        do 110 i=jk1,n
110   nr(i) = n
        nblock = nblock+1
        integs(1,nblock) = nrows
        integs(2,nblock) = k
        integs(3,nblock) = k
        go to 999
200     write(io,201)
201     format('error exit')
999     continue
        return
        end
```

Determine the B-spline defining polygon. With the arrays of the last section, the B-spline defining polygon can be determined. This is a two-step procedure:

- Determine the matrix A of Section 3. This can be accomplished by an application of an algorithm [5]. (This algorithm has been coded in Appendix A.) The arrays $T(\ )$, $p(\ )$, $ND(\ )$ are used by this algorithm along with a call to subroutine BSPLVD, a coded version of which can be found in [6].

- With the matrix A and the arrays $gx(\ )$, $gy(\ )$ a specialized banded linear matrix solver is used to determine the solution of the linear system, i.e., the solution being the B-spline defining polygon points,

$$b_j = (bx_j, by_j),\ j = 1, \ldots, n.$$

A coded form of such a solver can be found in the appendix of [6].

With the B-spline defining polygon points determined, B-spline curve coordinate values can be compiled with a sequence of calls to routine BVALUE of [6]. This procedure was carried out to generate figures in the article.

Conclusion

The described procedure is easy to use and simplifies design since it employs one constructive definition. It allows us to define a general curve using one form of a vector equation. Curvature and tangent discontinuity conditions can be constructed. It also facilitates the addition of options to this general procedure. □

References

1 Faux, I. D., and Pratt, M. J., *Computational Geometry and Design and Manufacture*, Chichester, England: Ellis Horwood Ltd., 1979.

2 McKee, J. M., and Kazden, R. J., "G-Prime B-Spline Manipulation Package," *David W. Taylor Naval Ship Research and Development Center Report 77-0036*, Bethesda, MD.

3 Wu, S., Abel, J., and Greenberg, D., "An Interactive Computer Graphics Approach to Surface Representation," *Communication of the ACM*, Vol. 20, No. 10, Oct. 1977, pp. 703-712.

4 Rogers, D., and Adams, J., *Mathematical Elements for Computer Graphics*, New York: McGraw-Hill, 1976.

5 Butterfield, K. B., "The Development and Application of Algorithms Associated with Surface Representation," Ph.D. Thesis, Brunel University, Uxbridge, U.K., Jan. 1978.

6 deBoor, C., *A Practical Guide to Splines*, New York: Springer-Verlag, 1978.