B-Spline Curves

We have seen how Bezier curves offer well-behaved control within a convex hull for any degree (number of control points) we choose. And from a computational point of view, it is a relatively simple matter to extend a Bezier curve by adding more control points because the computation of the new blending functions is straightforward.

However, we cannot easily control the curve locally. That is, any change to an individual control point will cause changes in the curve along its full length. In addition, we cannot create a local cusp in the curve, that is, we cannot create a sharp corner unless we create it at the beginning or end of a curve where it joins another curve. Finally, it is not possible to keep the degree of the Bezier curve fixed while adding additional points; any additional points will automatically increase the degree of the curve.

The so-called b-spline curve addresses each of these problems with the Bezier curve. It provides the most powerful and useful approach to curve design available today. However, the down side is that the b-spline curve is somewhat more complicated to compute compared to the Bezier curve, but as we shall see the two kinds of curves are closely related. In fact a b-spline curve degenerates into a Bezier curve when the order of the b-spline curve is exactly equal to the number of control points.

Mathematical Definition

For the Bezier curve defined over n+1 points the description in terms of a blending function is:

$$\vec{p}(u) = \sum_{i=0}^{n} \vec{p}_i \; J_{n,i}(u)$$

where $u \in [0,1]$ and:

$$J_{n,i}(u) = C_{n,i}\, u^i \,(1-u)^{n-i}$$

and $C_{n,i}$ is the binomial coefficient:

$$C_{n,i} = \binom{N}{i} = \frac{N!}{i!(N-i)!}$$

There are several key differences in the formulation of a b-spline curve. In the first place, since the curve can be defined over an arbitrarily large number of points without increasing the degree, we will replace the parametric variable, u, with a new variable, t, where t can range from $t_{\min} \le t \le t_{\max}$. We will associate values of $t = t_0, t_1, t_2, \ldots t_m$ with the m+1 control points that will define the curve. The t_i are called knots and the knots $[t_i]$ are called a knot vector or knot sequence. The only other restriction on t is that the values of t_i must be monotonically increasing. It simplifies the development to assume t takes on unit spacing, 0,1,2, etc and this is referred to as uniform knot spacing. The b-spline is defined for $t \in [t_0, t_m]$ as:

$$\vec{p}(t) = \sum_{i=0}^{n} \vec{p}_i \; N_{i,k}(t)$$

where $N_{i,k}(t)$ is the new blending function which now involves a new parameter, k, which will be defined below. $N_{i,k}(t)$ can be calculated from the "Cox-DeBoor" recursion relation:

$$N_{i,k}(t) = \frac{(t - t_i)\quad N_{i,k-1}(t)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - t)\quad N_{i+1,k-1}(t)}{t_{i+k} - t_{i+1}}$$

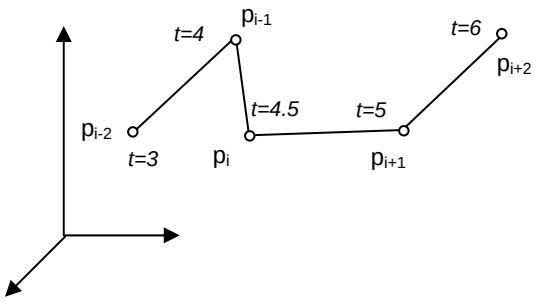
for k=2,3,…K and for all needed values of i. Note that this recursive definition for the b-spline blending function is very similar to what was developed earlier for the recursive definition of the Bezier blending function. The curve parameter, t, is not restricted to a unit range as u was before for the Bezier curve. The knots, t_i, are specific values of t that are associated with each of the control points, $\vec{p}_i$. As can be seen from the recursion equation above, the degree of the resulting polynomial (e.g., the degree of $N_{i,k}(t)$) is equal to k-1 so k is referred to as the "order" of the curve. An initial definition for $N_{i,k}(t)$ is needed to start the recursive calculation and this is:

$$N_{i,1}(t) = \begin{cases} 1 & t_i \le t \le t_{i+1} \\ 0 & elsewhere \end{cases}$$

The blending function, $N_{i,k}(t)$, is also referred to as the "basis" function, hence the name, b-spline curve (where b stands for basis). It defines a polynomial behavior over a range of t values.. The recursive definition of $N_{i,k}$ in terms of $N_{i,k-1}$ and so on until we get to $N_{i,1}$ is hard to do by hand, but it is a relatively simple matter for software implementation using modern programming languages.

Graphical Interpretation

Rather than the above mathematical definition, the b-spline curve is perhaps most clearly defined by describing its development first for a low order form and then extending this to higher order. For simplicity, we will begin with the description of a second order form that involves first degree (linear) basis functions. As a result, this order curve be a piecewise linear curve (e.g., a polyline). Consider the general problem of a piecewise linear interpolation of a set of points, $\vec{p}_0 \cdots \vec{p}_n$ as shown below (note that the values of t do not need to increase uniformly):



The interpolating linear curve can be written most easily in terms of a new variable, u, that varies from 0 to 1 over each segment so that we can write:

$$\vec{p}_{i-2}(u) = (1-u)\,\vec{p}_{i-2} + u\vec{p}_{i-1} \qquad \text{where} \quad u = \frac{t - t_{i-2}}{t_{i-1} - t_{i-2}}$$

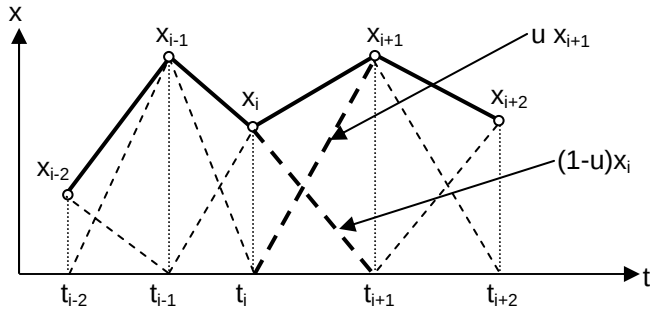
$$\vec{p}_{i-1}(u) = (1-u)\,\vec{p}_{i-1} + u\vec{p}_{i} \qquad \text{where} \quad u = \frac{t - t_{i-1}}{t_{i} - t_{i-1}}$$

$$\vec{p}_i(u) = (1-u)\vec{p}_i + u\vec{p}_{i+1} \qquad \text{where} \quad u = \frac{t-t_i}{t_{i+1}-t_i}$$

etc.

Here, we are using u as a simple dimensionless variable (0<u<1) for each segment while the values of t range from 0 to t_n or in this case from $t=t_{i-2}$ to $t=t_{i+2}$ for the portion of the curve shown in the figure above. Note also that the function, $\vec{p}_i(u)$ is indexed to the control point, $\vec{p}_i$, that defines the start of that particular segment.

It is somewhat easier to show just the curve in parametric space, e.g., the curve's x or y or z component versus t. For example, the x component of the curve (we can call it $\vec{p}_{ix}$ or rather just x for shorthand notation) would look like the following figure in parametric space:



where for the segments from $t=t_{i-1}$ to $t=t_{i+2}$ we can write:

$$x_{i-1}(u) = (1-u)x_{i-1} + ux_i = \frac{t_i - t}{t_i - t_{i-1}}x_{i-1} + \frac{t-t_{i-1}}{t_i - t_{i-1}}x_i \qquad \text{where} \quad u = \frac{t-t_{i-1}}{t_i - t_{i-1}}$$

$$x_i(u) = (1-u)x_i + ux_{i+1} = \frac{t_{i+1} - t}{t_{i+1} - t_i}x_i + \frac{t-t_i}{t_{i+1} - t_i}x_{i+1} \qquad \text{where} \quad u = \frac{t-t_i}{t_{i+1} - t_i}$$

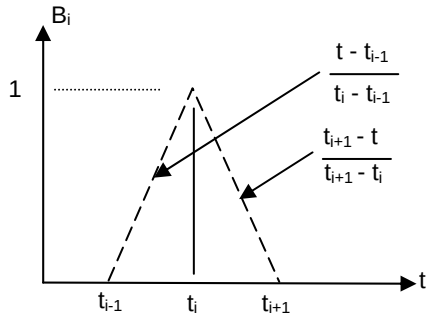
$$x_{i+1}(u) = (1-u)x_{i+1} + ux_{i+2} = \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}}x_{i+1} + \frac{t-t_{i+1}}{t_{i+2} - t_{i+1}}x_{i+2} \qquad \text{where} \quad u = \frac{t-t_{i+1}}{t_{i+2} - t_{i+1}}$$

Each straight line segment is thus the sum of descending and a rising dashed lines. But there is another more interesting way of looking at this segment of the curve (also called a bay). Rather than look at just the two (rising and falling) linear curves that make up the i-th segment, consider instead the two segments of the curve defined by one of the control points. For example, consider the i-th point, x_i, and combine the two segments that it defines as:

$$B_i(t) = \frac{t-t_{i-1}}{t_i - t_{i-1}} \qquad \textit{for} \qquad t_{i-1} \le t \le t_i$$

$$= \frac{t_{i+1} - t}{t_{i+1} - t_i} \qquad \textit{for} \qquad t_i \le t \le t_{i+1}$$

This defines a "hat" function as shown below which is a simple linear "basis" function for this curve:



We can then re-write the above expression for $x_i(u)$ or $x_i(t)$ in the new form:

$$x_i(t) = x_i B_i(t) + x_{i+1} B_{i+1}(t) \qquad for \qquad t_i \le t \le t_{i+1}$$

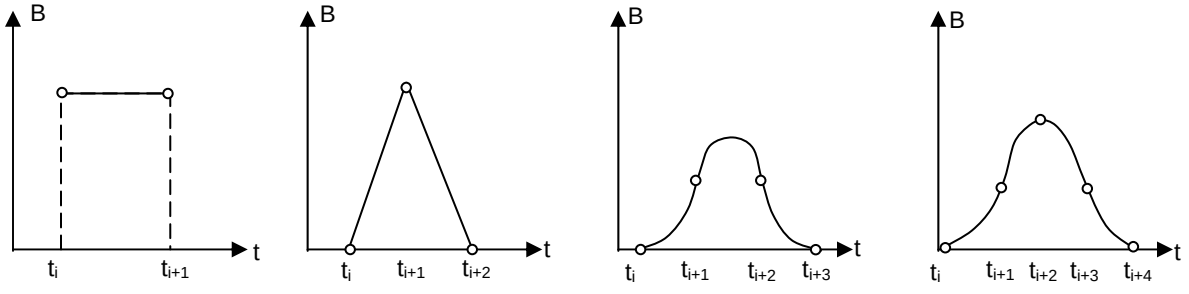
where the basis functions, $B_i(t)$, are defined above, and we must be careful in selecting the range of values of t to use for this segment (or bay). We can generalize this kind of representation for the full curve over all the control points as:

$$\vec{p}(t) = \sum_{i=0}^{n} \vec{p}_i(u) = \sum_{i=0}^{n} \vec{p}_i B_i(t)$$

where the $B_i(t)$ are called the basis functions (and are linear for this example).

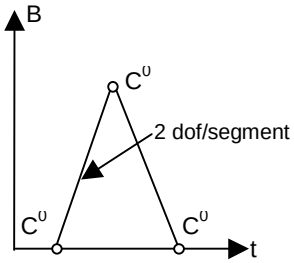
Higher Order Basis Functions

The above formulation will describe any piecewise linear curve (e.g., a polyline) but it will not define a smoother curve. However, we can readily generalize the process of defining the curve in terms of a basis function. In this case we must create a basis function that is higher than second order, e.g., third or fourth order or higher. For example, if we can create a third order (second degree or quadratic) basis function, then we can represent a curve as a superposition of quadratic basis functions so the resulting curve will be everywhere quadratic. Perhaps more revealing, we could also reduce the order of the basis function developed above from second to first order. The resulting first order or zero degree basis functions will have only a single parameter which is a constant. These points are much better illustrated in the following figure which shows the range of polynomial basis functions starting with first order or zero degree.

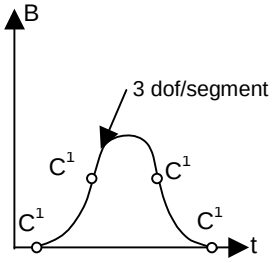


Let's consider the second order basis function shown above and that we developed earlier. It consists of two linear segments and each of these segments can be described by two constants (e.g., the slope and intercept or more conventionally, $x=c_0+c_1 t$). The two segments together require definition of 4 constants. However, we cannot specify all of them in an arbitrary fashion

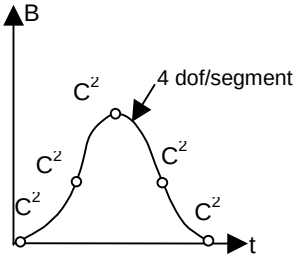
because the first curve must start with an amplitude, x=0, and the second curve must end with x=0. The other ends must meet at the common middle point. Together, these constitute 3 constraints for the 4 constants leaving one constant free to specify. We can specify this constant in order to achieve whatever overall amplitude we wish for the mid-point of the basis function. This way, the total degrees of freedom (constants) for the basis function is exactly matched by the total number of constraints. This is shown below graphically, and the process is extended for higher degree quadratic and cubic basis functions.



Constants (dof)=2*2=4
Constraints (C^0)=1*3=3
Net constants = 1



Constants (dof)=3*3=9
Constraints (C^1)=2*4=8
Net constants = 1

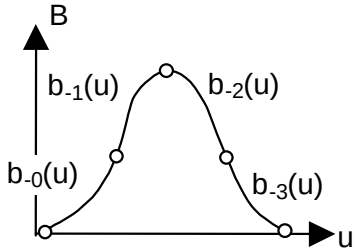


Constants (dof)=4*4=16
Constraints (C^1)=3*5=15
Net constants = 1

What we are really doing in the above illustration is requiring that each segment of the basis function be a polynomial of specified degree and then requiring that we have the highest possible degree of continuity between segments. For a linear function, we can have only positional continuity or what is called C^0 continuity. For a quadratic we can require positional and slope continuity or C^1 continuity. For a cubic we can require C^2 continuity or position, slope and curvature continuity. In each case, the next higher degree of continuity adds another constraint on the curve. As you can see, in each case the number of constants less the number of continuity constraints is exactly 1 which leaves us the freedom of fixing the overall amplitude of the basis function as needed.

Example: Cubic Basis Functions

As an example, we will illustrate the computation of the cubic or 4th order basis function. For this case we will have 4 segments defined over 5 control points with C^2 continuity at each point and the amplitude to be determined. We will use unit parameterization, $u \in [0,1]$. We will also use a simple notation to define the cubic polynomial for each segment of the basis function as follows. Note that we are using a negative subscript for the segment definition simply to avoid confusion with the appearance of the constants, b_i, on the right-hand side.



$$b_{-0}(u) = a_0 + b_0 u + c_0 u^2 + d_0 u^3$$
$$b_{-1}(u) = a_1 + b_1 u + c_1 u^2 + d_1 u^3$$
$$\vdots$$
$$b_{-i}(u) = a_i + b_i u + c_i u^2 + d_i u^3$$

This yields a total of 16 constants and as shown below, there are a total of 15 continuity constraints:

Position	Slope	Curvature
$0 = b_{-0}(0)$	$0 = b'_{-0}(0)$	$0 = b''_{-0}(0)$
$b_{-0}(1) = b_{-1}(0)$	$b'_{-0}(1) = b'_{-1}(0)$	$b''_{-0}(1) = b''_{-1}(0)$
$b_{-1}(1) = b_{-2}(0)$	$b'_{-1}(1) = b'_{-2}(0)$	$b''_{-1}(1) = b''_{-2}(0)$
$b_{-2}(1) = b_{-3}(0)$	$b'_{-2}(1) = b'_{-3}(0)$	$b''_{-2}(1) = b''_{-3}(0)$
$b_{-3}(1) = 0$	$b'_{-3}(1) = 0$	$b''_{-3}(1) = 0$

The 16th constraint is the specification of amplitude. It turns out to be most useful not to specify the amplitude at the mid-point of the basis function but rather to enforce the following constraint:

$$1 = b_{-0}(0) + b_{-1}(0) + b_{-2}(0) + b_{-3}(0)$$

In other words, we are specifying that the sum of the individual components of the basis function all sum up to unity at u=0 for each segment. As we will see later, it turns out that this also holds for any value of u and this is a particularly useful normalization that leads to the convex hull property discussed previously for the Bezier curve.

Using these results, we can readily solve for the 16 coefficients and write:

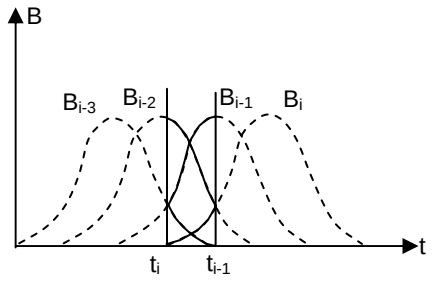
$$b_{-0}(u) = \tfrac{1}{6}u^3$$
$$b_{-1}(u) = \tfrac{1}{6}[1 + 3u + 3u^2 - 3u^3]$$
$$b_{-2}(u) = \tfrac{1}{6}[4 - 6u^2 + 3u^3]$$
$$b_{-3}(u) = \tfrac{1}{6}[1 - 3u + 3u^2 - u^3]$$

These can be recognized as a kind of blending function for the b-spline curve and we can use our previous matrix notation to represent them in an even more compact (and elegant) form:

$$\mathbf{b} = \begin{bmatrix} b_{-0}(u) & b_{-1}(u) & b_{-2}(u) & b_{-3}(u) \end{bmatrix} = \mathbf{u}\tilde{\mathbf{M}} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \tfrac{1}{6} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 3 & 0 & -3 \\ 0 & 1 & 4 & 1 \end{bmatrix}$$

where we can identify **u** as the familiar polynomial matrix and $\tilde{\mathbf{M}}$ as an interpolation matrix in much the same way as was developed previously for the parametric cubic polynomial and Bezier curves.

If we consider the i-th bay, the figure below illustrates the superposition of the 4 basis functions over this bay:



and the resulting curve over the bay can be written as:

$$\begin{aligned}\vec{p}_i(t) &= \vec{p}_{i-3}B_{i-3}(t) + \vec{p}_{i-2}B_{i-2}(t) + \vec{p}_{i-1}B_{i-1}(t) + \vec{p}_iB_i(t) \\ &= \sum_{r=-3}^{0} \vec{p}_{i+r}B_{i+r}(t) \\ &= \sum_{r=-3}^{0} \vec{p}_{i+r}b_r(u) = \sum_{r=0}^{3} \vec{p}_{i-r}b_{-r}(u)\end{aligned}$$

where we have accounted for the unusual negative index notation. As you can see, the rightmost part of the leftmost basis function is used, as is the leftmost part of the rightmost basis function. This "inversion" of the order of the basis functions is incorporated in our notation where the terms, $b_{-i}(u)$, are summed in reverse order. The same result can also be obtained simply by reversing the column order in the interpolating matrix, $\tilde{\mathbf{M}}$, above to yield:

$$\vec{p}_i(u) = \mathbf{u}\,\mathbf{M} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_i & \vec{p}_{i+1} & \vec{p}_{i+2} & \vec{p}_{i+3} \end{bmatrix}^T$$

Basis Function of Arbitrary Order

It can be shown that this type of basis function development can be generalized to any order (or degree) simply by applying the recursive Cox-DeBoor formula for $N_{i,k}(t)$ introduced at the outset. When we write this in matrix notation, the b-spline basis functions for a curve of order k (degree=k-1) can be written for unit parameterization as:

$$\mathbf{p}_i(u) = \mathbf{u}\,\mathbf{M}_S\,\mathbf{B}_S$$

where $\mathbf{M}_S$ is k by k dimension and $\mathbf{B}_S$ is k by 1 dimension. It also can be shown that the terms in $\mathbf{M}$ can be calculated as:

$$\left[M_S\right] = \left[M_{i+1,j+1}\right] = \left[\frac{1}{(k-1)!} C_{k-1,i} \sum_{m=j}^{k-1} (k-(m+1))^i (-1)^{m-j} C_{k,m-j}\right]$$

and

$$C_{i,j} = \binom{i}{j} = \frac{i!}{j!(i-j)!} = binomial\ \ coefficient$$

and where

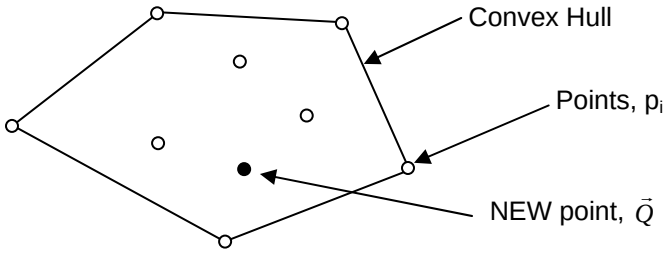
$$\left[B_S\right] = \begin{bmatrix} \vec{p}_i & \vec{p}_{i+1} & \vec{p}_{i+2} \cdots & \vec{p}_{i+k-1} \end{bmatrix}^T$$

Note that the dimension of $\mathbf{M}_S$ is k by k where k is the order of the basis functions, and $\mathbf{B}_S$ is a vector of k control points starting with the i-th point. These expressions form a convenient basis from which to prepare simple Matlab programs to illustrate the behavior of these remarkable curves for any specified order or degree. The only complicated portion is in the development of the interpolation matrix from the above formula but this requires a straightforward summation and needs to be done only once for the entire curve. The next section illustrates how to use these basis functions to define a curve.

Convex Hull Property

We have already noted the convex hull property of Bezier curves, but we showed this entirely from a geometric point of view using the deCasteljau formulation of the curve. The property is actually a result of the particular choice of basis function (Bernstein polynomial) and this is also true for the b-spline basis function as well. We will show this for the b-spline curves using a more formal mathematical argument.

A convex hull for points, $\vec{p}_i$, is the smallest convex polygon defined with points at the vertices that encloses ALL the points. This is illustrated in the figure below where it should be noted that not all points are used to define vertices of the enclosing convex polygon.



Mathematically, the convex hull is defined by requiring that any new point, $\vec{Q}$, within the convex hull can be written as:

$$\vec{Q} = \sum_i w_i \, \vec{p}_i$$

where it is also required that:

$$w_i \geq 0 \quad \text{and} \quad 1 = \sum_i w_i$$

We can show that this must define a convex hull by the following argument. For a convex hull a line connecting any two points in the hull cannot cross outside the hull. That is, the line must lie entirely within the hull. Assume two arbitrary points, $\vec{Q}_1$ *and* $\vec{Q}_2$ within the convex hull. Using the above expression for a new point, $\vec{Q}$, it follows that we can express these points as:

$$\vec{Q}_1 = \sum_i w_{1i} \, \vec{p}_i \quad \text{and} \quad \vec{Q}_2 = \sum_i w_{2i} \, \vec{p}_i$$

We can then define a straight line between them as:

$$\vec{Q}(\alpha) = \alpha \vec{Q}_1 + (1-\alpha)\vec{Q}_2$$

where α is the parametric coordinate and $\alpha \in [0,1]$. In other words, we can define any point, $\vec{Q}$, along this line as $\vec{Q} = \vec{Q}(\alpha)$ for a specific value of α. Using the expressions for $\vec{Q}_1$ and $\vec{Q}_2$ from above yields:

$$\vec{Q} = \vec{Q}(\alpha) = \alpha \sum_i w_{1i}\ \vec{p}_i + (1-\alpha)\sum_i w_{2i}\ \vec{p}_i$$
$$= \sum_i \left[\alpha\, w_{1i} + (1-\alpha) w_{2i}\right]\vec{p}_i$$
$$= \sum_i w_i\ \vec{p}_i$$

where w_i is defined as the bracketed expression above. Thus, $\vec{Q}(\alpha)$ satisfies the first requirement to be within the convex hull. From the definitions of the two arbitrary points, it also follows that $w_{1i} \geq 0$ and $w_{2i} \geq 0$, and the last condition:

$$\sum_i w_i = \alpha \sum_i w_{1i} + (1-\alpha)\sum_i w_{2i}$$
$$= \alpha \cdot 1 + (1-\alpha)\cdot 1 = 1$$

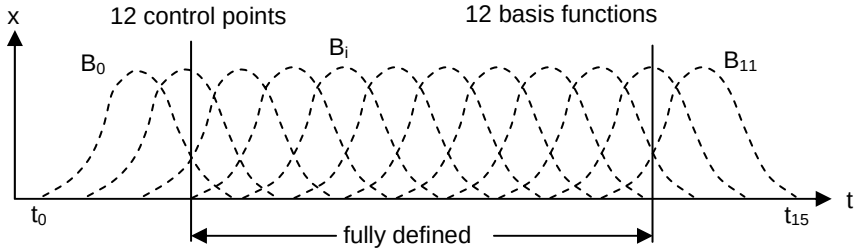
is also met. Thus the line, $\vec{Q}(\alpha)$, and any point, $\vec{Q}$, on it, must also lie within the convex hull! It should be clear from this argument that the principal requirement for a convex hull is that the weighting function used must everywhere sum up to unity. This can readily be verified for the b-spline basis functions by simply adding up the blending functions for any value of u to see that all of the terms involving the parameter, u, cancel out leaving unity as the final result. Another way to show this is to note that each of the rows of the interpolation matrix, $\tilde{\mathbf{M}}$ (or **M**) adds up to zero, except for the last row, which adds to unity..

Uniform Periodic B-Spline Curves

Now that we have defined suitable b-spline basis functions, we will use them to describe a curve as defined by specified control points and knot values for t. As noted in the initial development for a cubic basis function (k=4), the curve can be written as:

$$\vec{p}(t) = \sum_{i=0}^{m} \vec{p}_i(u) = \sum_{i=0}^{m} \vec{p}_i B_i(t) = \sum_{i=0}^{m} \vec{p}_i N_{i,4}(t)$$

where we can illustrate this graphically below:



For the case shown above over a range of t values from t_0 to t_{15} we can see the following:

Control points:	0…m (m+1)
Basis functions:	m+1 (using k+1=5 knots each)
Total knots:	m+1+(k+1)-1 = m+k+1 = m+5

However, close inspection of the above figure reveals that the curve is not fully defined at both ends because there are fewer basis functions defined over the first 3 bays and the last 3 bays. In fact the curve is "fully" defined only over the following region:

Defined knots:	(m+5)-3-3 = m-1
Defined bays:	(m-1)-1 = m-2
Parameter t:	$t_3 \le t \le t_{m+1}$

As a result of this behavior the uniform periodic b-spline curve does not start at the first control point nor does it end at the last control point. As we will see in the next section, this behavior can be overcome and the curves can be forced to pass through a given control point by repeating the control point one or more times in the $\mathbf{B}_S$ vector.

Remarks:

1. All basis segments are defined for $0 \le u \le 1$ and each function is an offset version of the previous.
2. The normalization condition, $1 = b_{-0}(0) + b_{-1}(0) + b_{-2}(0) + b_{-3}(0)$, plus the symmetry of B_k means that at each knot the basis functions sum to unity. By summing the full expressions for the $b_{-k}(u)$ directly, you can see that the basis functions ALWAYS sum to unity for any value u. As noted above, this results in the important convex hull property.
3. When the order (degree+1) of the basis function is exactly the same as the number of distinct control points, the b-spline degenerates into a conventional Bezier curve. We will discuss this further in the material below.
4. By generalizing on Remark #2 we can show that the convex hull property also applies to a full b-spline curve but it only applies for each successive group of k control points. This gives rise to an interesting graphical visualization of convex hull as discussed below.
5. Any control point will control the behavior of a single basis function but this function will affect only the behavior in a limited number of nearby bays (4 for a cubic). This is how the b-spline provides the desirable local control property.

Graphical Illustrations

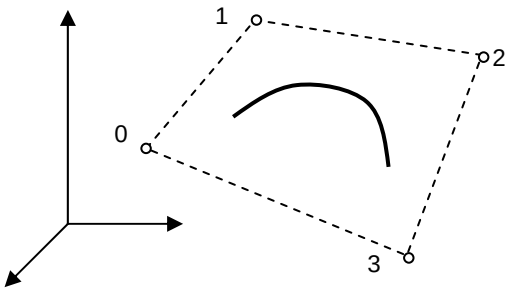
The power and capabilities of the b-spline curve can best be illustrated by examining how to actually create a curve and control it. We will consider a cubic b-spline because it usually offers the best combination of flexibility and computational simplicity (and it has been developed in our above treatment). For the cubic b-spline, the basis functions can be easily expressed in matrix form using the previous result:

$$\vec{p}_i(u) = \mathbf{u}\,\mathbf{M}_S\,\mathbf{B}_S = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_i & \vec{p}_{i+1} & \vec{p}_{i+2} & \vec{p}_{+3i} \end{bmatrix}^T$$

where the **B** matrix contains the relevant control points for the portion of the curve under consideration. Note also that the columns of the $\mathbf{M}_S$ matrix are obtained by reversing the order from the $\tilde{\mathbf{M}}$ matrix developed for the cubic basis functions in the previous sections.

Basic Curve Segment

For a single portion of the curve, we need consider only the 4 control points needed to define this segment (bay). As noted earlier, since the number of control points is exactly equal to the b-spline order, the resulting curve will behave generally like a Bezier curve. In particular it will exhibit the convex hull property and stay within the convex hull formed by the 4 control points. The figure below illustrates this.



One of the most striking results is that the curve does not actually start at the first point or stop at the last point like the Bezier curve does. This can be illustrated by evaluating the above expression for the cubic b-spline at u=0 or u=1 as follows (we will consider the u=0 end):

$$\vec{p}_0(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_0 & \vec{p}_1 & \vec{p}_2 & \vec{p}_3 \end{bmatrix}^T = \tfrac{1}{6}\vec{p}_0 + \tfrac{2}{3}\vec{p}_1 + \tfrac{1}{6}\vec{p}_2$$

Thus, the uniform periodic b-spline truly *approximates* the control points (rather than interpolating them as the conventional cubic spline curve does).. While this may appear to be a problem, it actually allows the curve to be "designed" more freely.

Segment with Repeated Control Point at Start

If it is necessary for the curve to start at a point closer to the first control point, then one can repeat the first control point in the $\mathbf{B}_S$ matrix as follows:

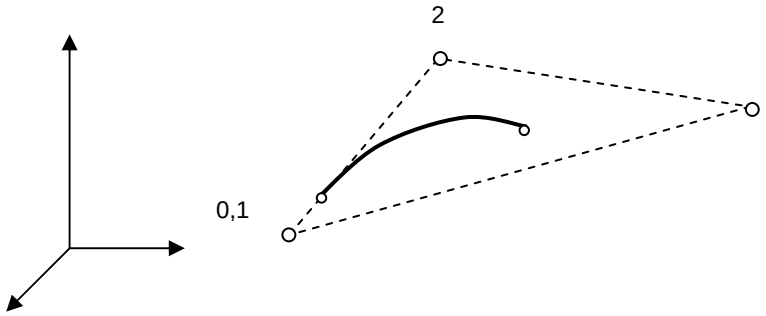
$$\vec{p}_0(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_0 & \vec{p}_0 & \vec{p}_2 & \vec{p}_3 \end{bmatrix}^T = \vec{p}_0 + \tfrac{1}{6}(\vec{p}_1 - \vec{p}_0)$$

where we have set $\vec{p}_1 = \vec{p}_0$. From this result it is clear that we have not quite achieved the desired result because the curve now starts at a point 1/6 th of the distance from $\vec{p}_0$ to $\vec{p}_1$ as shown in the figure below.

It can also be shown by taking the derivative of the curve that the starting tangent is:

$$\vec{p}_0'(0) = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}\tfrac{1}{6}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}\vec{p}_0 & \vec{p}_0 & \vec{p}_2 & \vec{p}_3\end{bmatrix}^T = \tfrac{1}{2}(\vec{p}_1 - \vec{p}_0)$$

which shows that the curve now starts out tangent to the chord from the first to the second points as shown in the above figure.



Segment with Control Point Repeated Twice at Start

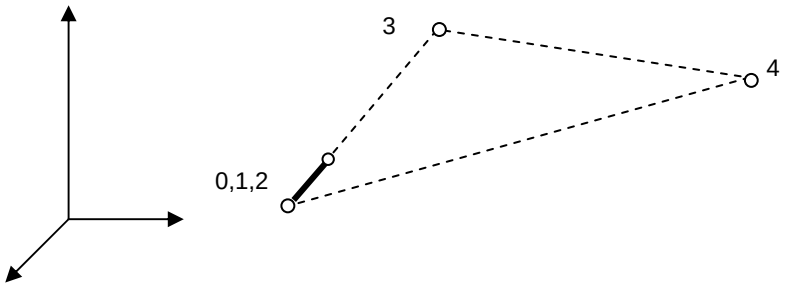
In order to get the curve to start at the first point we will have to repeat the first control point again:

$$\vec{p}_0(0) = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\tfrac{1}{6}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}\vec{p}_0 & \vec{p}_0 & \vec{p}_0 & \vec{p}_3\end{bmatrix}^T = \vec{p}_0$$

Now the curve starts at $\vec{p}_0$ as we want. The end of the curve is at:

$$\vec{p}_0(1) = \begin{bmatrix}1 & 1 & 0 & 1\end{bmatrix}\tfrac{1}{6}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}\vec{p}_0 & \vec{p}_0 & \vec{p}_0 & \vec{p}_3\end{bmatrix}^T = (1-\tfrac{1}{6})\vec{p}_0 + \tfrac{1}{6}\vec{p}_3$$

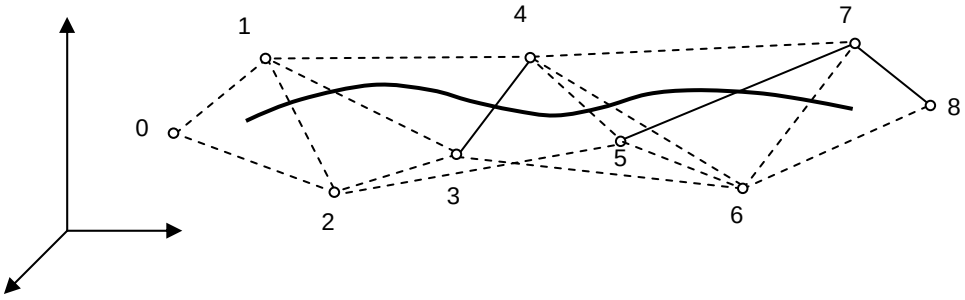
which is the point 1/6 of the way along the chord as shown in the figure below.



This result shows that while the curve will start out at the first point, it follows a straight line initially. If the next segment of the curve is defined by points 1,2,3 and a new point 4 as shown above, it will start at the end of the first straight segment and continue out into the convex hull formed by points 1…4 (as shown in the previous example for a single repeated starting control point). The result is that a b-spline curve starting with 3 coincident control points will start with a short straight segment (equal in length to 1/6 of the first chord). This may be objectionable but can be overcome by several means as will be discussed later.

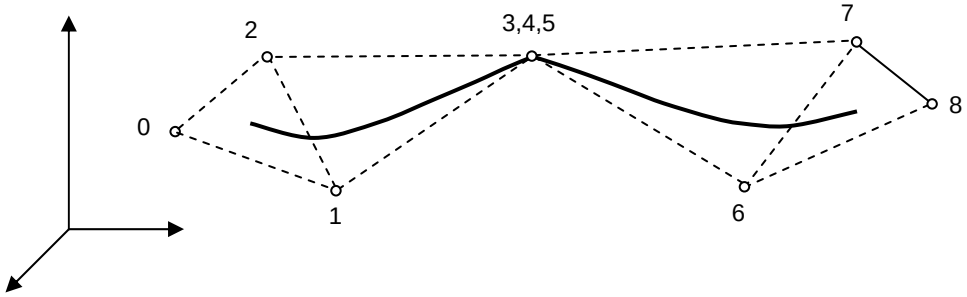
Curve with Multiple Control Points

To extend the curve to include more control points, it is only necessary to draw additional curves by taking in succession sets of 4 control points in the $\mathbf{B}_S$ matrix until the last point in the sequence is reached. This is illustrated graphically in the figure below.



Again, note that the curve will not start at the first point nor will it end at the last unless we repeat these control points 3 times in the $\mathbf{B}_S$ vector as we start and end the curve (but as noted this will produce a short straight segment at the start). More importantly, however, the figure above shows that the curve also obeys the convex hull property for each of its defining sections. That is, the curve must, in successive segments, always remain within the convex hulls formed by taking successive sets of 4 control points. These convex hulls are highlighted by dashed 4-sided polygons formed from successive sets of 4 control points in the above figure.

An interesting feature of the uniform periodic b-spline curve is the ability to form cusps and tangencies at intermediate points along the curve. This can be done simply by repeating a control point as needed. For example, to create a cusp at a particular control point, it is only necessary to repeat that control point 3 times. As the convex hulls illustrate below, this forces the curve through the control point and allows the possibility of a change in tangent.



Continuing with this reasoning, it should be obvious that we can also force the curve to define a straight line (or a straight portion within a curve) simply be forcing all of the control points to be colinear. Of course, reducing the degree to 1 will accomplish the same thing!

Phantom Control Points

Instead of simply repeating control points to force the curve to start at a given point, it is also possible to define new control points before the designated "starting" point. Normally, these control points are not shown in a graphical representation and so they are often referred to as "phantom" control points. To illustrate this approach, consider what happens if we create a phantom control point, $\vec{p}_{-1}$, which will appear just before the rest of the control points. The first segment of the b-spline will thus start at:

$$\vec{p}_0(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_{-1} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \end{bmatrix}^T$$

At this point, the location of $\vec{p}_{-1}$ has not been specified. We can require that the curve start at $\vec{p}_0$ by setting the above expression equal to $\vec{p}_0$ and solving for the $\vec{p}_{-1}$ that will accomplish this. This is left as an exercise. Additional phantom control points can be located to force the b-spline to behave in other ways; use your imagination to consider the possibilities…

Rational B-Spline Curves

The same reasoning that led us to consider rational Bezier curves can also be applied to b-spline curves as well. The easiest way to visualize this is to imagine that the b-spline curve is defined in 4-D homogeneous space using the four coordinates, $\left[\tilde{x}, \tilde{y}, \tilde{z}, h\right]$, where:

$$\tilde{x}(t) = \sum_{i=0}^{n} \tilde{p}_{ix} N_{i,k}(t), \quad \tilde{y}(t) = \sum_{i=0}^{n} \tilde{p}_{iy} N_{i,k}(t), \text{ and } \tilde{z}(t) = \sum_{i=0}^{n} \tilde{p}_{iz} N_{i,k}(t)$$

and

$$h(t) = \sum_{i=0}^{n} h_i N_{i,k}(t)$$

are the individual components. In this case, the $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ components of the control points, $\tilde{p}_i$, are used to define the curve components, along with the "weights", h_i, at each point. It is the additional flexibility provided by the h_i that make the rational curves so versatile. The actual Cartesian components, x, y, and z, are then given by the homogeneous projection back into 3-D space as:

$$x(t) = \frac{\sum_{i=0}^{n} \tilde{x}_i N_{i,k}(t)}{\sum_{i=0}^{n} h_i N_{i,k}(t)}, \quad y(t) = \frac{\sum_{i=0}^{n} \tilde{y}_i N_{i,k}(t)}{\sum_{i=0}^{n} h_i N_{i,k}(t)}, \quad z(t) = \frac{\sum_{i=0}^{n} \tilde{z}_i N_{i,k}(t)}{\sum_{i=0}^{n} h_i N_{i,k}(t)}$$

or in vector form:

$$\vec{p}(t) = \frac{\sum_{i=0}^{n} \tilde{p}_i \, N_{i,k}(t)}{\sum_{i=0}^{n} h_i \, N_{i,k}(t)} = \frac{\sum_{i=0}^{n} h_i \vec{p}_i \, N_{i,k}(t)}{\sum_{i=0}^{n} h_i \, N_{i,k}(t)}$$

When the basis functions are expressed as nonuniform or open curves as developed in the textbook, these rational curves are often referred to as nonuniform rational b-splines or NURBS, and such curves are widely used in CAD systems. We can also express the curves in terms of the uniform, periodic basis functions as developed in matrix form above. The result is:

$$\vec{p}(u) = \frac{\mathbf{u}\mathbf{M}_\mathbf{S}\mathbf{B}_\mathbf{S}}{\mathbf{u}\mathbf{M}_\mathbf{S}\mathbf{H}_\mathbf{S}}$$

where $\mathbf{M}_\mathbf{S}$ is the k by k interpolation matrix (k=curve order) for the b-spline and the geometric matrices, $\mathbf{B}_\mathbf{S}$ and $\mathbf{H}_\mathbf{S}$, contain the k control points governing a portion of the curve. For example, for a cubic basis function, k=4, and:

$$\mathbf{B}_\mathbf{S} = \begin{bmatrix} h_i\vec{p}_i & h_{i+1}\vec{p}_{i+1} & h_{i+2}\vec{p}_{i+2} & h_{i+3}\vec{p}_{i+3} \end{bmatrix}^T \text{ and } \mathbf{H}_\mathbf{S} = \begin{bmatrix} h_i & h_{i+1} & h_{i+2} & h_{i+3} \end{bmatrix}^T$$

Note that the $\mathbf{B}_\mathbf{S}$ matrix has vector components, each with the x, y and z components, while the $\mathbf{H}_\mathbf{S}$ matrix contains only the scalar weights, h_i. As for the Bezier curve, the challenge is to figure out how to adjust these weights to achieve a desired curve configuration. As for Bezier curves, it is possible to determine values of h_i that will allow quadratic b-splines to exactly represent standard conic sections, including the important case of a circle.

Summary Comments

These notes are provided to supplement the textbook which can at times become overwhelming in its mathematical detail. The treatment of b-splines in these notes is designed to emphasize the commonality with the previous curves we have discussed and to provide a unifying notation. However, the material presented here only touches on the vast literature concerning B-splines. There are a number of textbooks that cover much more, and the literature is fast becoming quite voluminous!

The following comments are provided to better clarify the limitations inherent in these notes and also to suggest extensions that might be explored.

1. We have considered only the uniform periodic b-spline curve. There are many other forms of b-splines that further illustrate its flexibility and expressiveness. For example, the curve can also be defined by the selection of the knot values (t_i) and their repetition in the Cox-DeBoor recursion formula for the basis functions. In these notes, we have considered only uniform knot values without repeating any values. If knot values are repeated at the ends of the curve as illustrated above, the result is called an "open" b-spline (or a uniform open b-spline if the knot increments are constant). Such curves can also be made to start and stop at the end points, and they offer some attractive properties.
2. Derivatives can also be computed for b-splines by taking derivatives of the basis functions. The results can also be recast in matrix form as we have done for other curves.

3. As we have seen, when control points are repeated at the ends for a periodic B-spline, the curve can be forced to pass through the end control point, but it turns out that it also includes a short straight segment. This may be undesirable in some, but it can be overcome by creating phantom control points..
4. Control points can be manipulated in order to force the b-spline curve to do a number of things such as creating the cusps we noted earlier. The curve can also be forced to pass through specific points if necessary. Another interesting situation is the possibility of adding additional control points without causing the curve to change its location. This is often needed to allow the curve to be given more "flexibility" without immediately changing its shape in an unexpected manner. (In other words, the CAD operator might want to add control points in preparation to making new changes in the curve, but not at the expense of disrupting the existing situation.)
5. Rational b-splines provide much greater expressive power and are generally preferred for CAD applications. A rational b-spline is formed by creating a conventional nonrational curve (the kinds we have created above) in 4-dimensional homogeneous coordinates and then projecting the curve back into 3D space. The resulting curves turn out to be more useful because they can be subjected to both affine and perspective transformations while the nonrational b-splines behave correctly only under affine transformations. In addition, rational b-splines can more easily be used to model circles and other conics which are commonly used in CAD. When combined with nonuniform parameterization to provide the greatest power and flexibility, these curves are often referred to as, nonuniform, rational b-splines or NURBS for short. Again, many textbooks provide more details on these curves.