Cubic Spline Curves and Surfaces
AUI Course
© Denbigh Starkey

Major points of these notes

1. Overview of spline curves and surfaces 2
2. What I won’t be covering 4
3. Hermite curves 5
4. Bezier curves 8
5. Convex hull property of Bezier splines 11
6. Converting between Hermite and Bezier systems 13
7. B-Spline curves 15
8. Continuity between consecutive splines 20
9. Comparing Hermite, Bezier, and B-Spline 22
10. Knot vectors and B-Splines 23
11. Spline surfaces 25
1. Overview of Spline Curves and Surfaces

Splines were first used for drawing curved lines long before they were used in computer-based design systems. They were flexible pieces of metal whose shape could be adjusted through the judicious use of small weights. Now they refer to parametric polynomial systems of curves, and parametric bipolynomial (two parameters) systems of surfaces.

Initially I’ll just consider curves, and will ignore surfaces until the end of these notes, when they will be shown to be equivalent to spline curves. Whereas a spline curve will be defined over a set of control points, a spline surface can be developed as a spline defined over a set of spline curves instead of points.

Spline curves can be partitioned into two types of curves: approximation splines and interpolation splines. If we have a set of control points that we want to define the curve, an approximation spline won’t, in general, pass accurately through the points, but will come close to them. An interpolation curve will, however, pass precisely through them.

There are uses for both kinds of curves. For example, if our control points are derived experimentally, then they are likely to include small errors, and an approximation spline is likely to be best. If, however, the points are mathematically precise, then an interpolation curve is likely to be best. Approximation splines also have the advantage that they are typically continuous to one more degree that interpolation splines, which means that they will appear somewhat smoother.

I’ll first describe two equivalent systems (Hermite and Bezier) for interpolation splines, and will then describe B-splines, which are approximation splines. Although quadratic splines are also very common, I’ll mainly describe cubic splines in these notes.

Some notation will be consistent between all of the methods for drawing curves. The most important is that the parametric equations will have the form:

\[
\begin{align*}
    x(u) &= a_xu^3 + b_xu^2 + c_xu + d_x \\
    y(u) &= a_yu^3 + b_yu^2 + c_yu + d_y \\
    z(u) &= a_zu^3 + b_zu^2 + c_zu + d_z
\end{align*}
\]

where \(0 \leq u \leq 1\).
Looking just at the $x$ term, we can rewrite this as

$$x(u) = U \ C_x$$

where

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$C_x = \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

Spline systems (e.g., Hermitian or Bezier) will compute $C_x$ as the product of two matrices, named the system matrix, which does not depend on the control values, and the geometry matrix, which contains the control values. So this equation will usually become:

$$x(u) = U \ M_{\text{splinetype}} \ G_{\text{splinetype},x}$$
$$y(u) = U \ M_{\text{splinetype}} \ G_{\text{splinetype},y}$$
$$z(u) = U \ M_{\text{splinetype}} \ G_{\text{splinetype},z}$$

For simplicity this will often be written over all three as

$$P(u) = U \ M_{\text{splinetype}} \ G_{\text{splinetype}}$$

Before seeing how this works with the Hermitian matrices, I’ll need one more bit of notation, related to continuity. We’ll see three different levels of continuity, $C^{(0)}$, $C^{(1)}$, and $C^{(2)}$. A curve is $C^{(0)}$ if it is continuous. I.e., all of its pieces are joined together, but not necessarily smoothly. $C^{(1)}$ means that the first derivatives are continuous, and so the curve appears to be smooth without any abrupt bends. $C^{(2)}$ means that the second derivatives are also continuous. So it will appear even smoother since the rate of change in the slope doesn’t suddenly change. Hermite and Bezier will be $C^{(1)}$, and B-splines will be $C^{(2)}$. However Hermite and Bezier interpolate, while B-splines approximate.
2. What I won’t be covering

I’ll be skipping some of the underlying mathematics in this first set of notes, although I have included the derivations of the Hermite and Bezier formulae. The biggest gap will be that I won’t cover the Cox-deBoor recursive definitions for B-splines, and hence also the underlying meaning of knots when defining B-splines. These equations, which were developed independently by Cox and deBoor, define the blending functions which, when summed, give any B-spline curve. In the next three sets of notes we’ll try to understand these recursive definitions.

The hope is that this first set of notes will provide a relatively gentle introduction to this topic, then following notes on the Cox-deBoor equations, NURBS curves, and NURBS Surfaces will get into the underlying mathematics.
3. Hermite Interpolation Splines

Hermite splines are specified by a starting and ending point, and a starting and ending curve direction vector. Hearn and Baker use \( P_0 \) and \( P_1 \) for the end points, and \( DP_0 \) and \( DP_1 \) for the direction vectors. For reasons that will become obvious when we get to Bezier splines, I’ll use \( P_0 \) and \( P_3 \) for the end points, and so \( DP_0 \) and \( DP_3 \) for the vectors at each end.

The Hermitian system is defined by

\[
P(u) = U \, M_H \, G_H
\]

where

\[
U = [u^3 \, u^2 \, u \, 1], \text{ as usual,}
\]

\[
M_H = \begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix} \text{ is the Hermitian matrix, and}
\]

\[
G_H = \begin{bmatrix}
P_0 \\
P_3 \\
DP_0 \\
DP_3
\end{bmatrix} \text{ is the Hermitian geometry vector.}
\]

To derive this, \( U = [u^3 \, u^2 \, u \, 1] \), and so \( U' = [3u^2 \, 2u \, 1 \, 0] \). Setting \( P(0) = P_0 \), \( P(1) = P_1 \), \( P'(0) = DP_0 \), and \( P'(1) = DP_3 \), gives the equations:

\[
P(0) = [0 \, 0 \, 0 \, 1] \, M_H \, G_H = P_0
\]

\[
P(1) = [1 \, 1 \, 1 \, 1] \, M_H \, G_H = P_3
\]

\[
P'(0) = [0 \, 0 \, 1 \, 0] \, M_H \, G_H = DP_0
\]

\[
P'(1) = [3 \, 2 \, 1 \, 0] \, M_H \, G_H = DP_3
\]

Stacking these up gives:
Canceling out the geometry vector gives

\[
M_H G_H = G_H
\]

As an example, say that we want to define a curve segment from the point \( P_0 = (0, 1) \) to the point \( P_3 = (2, 0) \), with a starting vector \( D P_0 = (0, 3) \), and an ending vector \( D P_3 = (3, 3) \), as shown below. I’ve added in a rough sketch of the curve that we’d like to get.

\[
M_H = \begin{bmatrix}
0 & 0 & 1 \\
1 & 1 & 1 \\
0 & 1 & 0 \\
3 & 2 & 1 \\
\end{bmatrix}^{-1} = \begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
\end{bmatrix}
\]

The geometry vector for \( x \) and \( y \) are:
\[
G_{H,x} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 3 \end{bmatrix} \quad \text{and} \quad G_{H,y} = \begin{bmatrix} 1 \\ 0 \\ 3 \\ 3 \end{bmatrix}.
\]

So we can use the formula to compute \(x(u)\) and \(y(u)\) with the equations:

\[
x(u) = \begin{bmatrix} u^3 & u^2 & u \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 3 \end{bmatrix}
\]

\[
y(u) = \begin{bmatrix} u^3 & u^2 & u \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 3 \\ 3 \end{bmatrix}.
\]

Multiplying out the matrices on the right, this gives

\[
x(u) = -u^3 + 3u^2 \\
y(u) = 8u^3 - 12u^2 + 3u + 1
\]

As \(u\) ranges from 0 to 1, this gives a curve segment that satisfies our requirements for starting and ending vectors. E.g., \(x(0) = 0, y(0) = 1\), and so \(P(0) = (0, 1)\), which is the correct starting point \(P_0\). Similarly, \(P(1) = (2, 0)\), which was our ending point \(P_3\). We also need to show that the line has an infinite slope at \(u = 0\), and a slope of 1 at \(u = 1\).

\[
\frac{dy}{dx} = \frac{\frac{dy}{du}}{\frac{dx}{du}} = \frac{24u^2 - 24u + 3}{-3u^2 + 6u}.
\]
Evaluating this at \( u = 0 \) and \( u = 1 \) we get:

\[
\left. \frac{dy}{dx} \right|_{u=0} = \frac{3}{0} = \infty, \quad \text{and} \quad \left. \frac{dy}{dx} \right|_{u=1} = \frac{3}{3} = 1.
\]
4. Bezier Interpolation Splines

The Bezier system is a modification of the Hermitian system, which instead of using \( P_0, P_3, DP_0, \) and \( DP_3, \) uses four control points \( P_0, P_1, P_2, \) and \( P_3, \) where \( P_0 \) and \( P_3 \) are the endpoints, as before,\(^1\) and \( P_1 \) and \( P_2 \) are placed as shown in the example below:

The two additional points have been drawn so that \( DP_0 = 3(P_1 - P_0) \) and \( DP_1 = 3(P_3 - P_2). \) I.e., \( P_1 \) will always be one third of the distance along \( DP_0, \) and \( P_2 \) will be on the other side of \( P_3 \) from the direction of \( DP_3, \) where the distance from \( P_2 \) to \( P_3 \) is one third of the length of the vector \( DP_3. \)

The Bezier Geometry vector is

\[
G_B = \begin{bmatrix}
P_0 \\
P_1 \\
P_2 \\
P_3 \\
\end{bmatrix}.
\]

\(^1\) Which is why I named the same points \( P_0 \) and \( P_3 \) in Hermite, to avoid confusion now
I can’t resist using some math to derive the Bezier formula. Since \( DP_0 = 3 \) \((P_1 - P_0)\) and \( DP_1 = 3(P_3 - P_2)\), we can convert the Hermitian geometry vector to the Bezier geometry vector through:

\[
\begin{bmatrix}
P_0 \\
P_3 \\
DP_0 \\
DP_3
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
-3 & 3 & 0 & 0 \\
0 & 0 & -3 & 3
\end{bmatrix}
\begin{bmatrix}
P_0 \\
P_1 \\
P_2 \\
P_3
\end{bmatrix}.
\]

Substituting this into the Hermitian equation we get:

\[
P(u) = [u^3 \ u^2 \ u \ 1]
\begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
-3 & 3 & 0 & 0 \\
0 & 0 & -3 & 3
\end{bmatrix}
\begin{bmatrix}
P_0 \\
P_1 \\
P_2 \\
P_3
\end{bmatrix}
\]

and then multiplying the two 4\(\times\)4 matrices gives the Bezier formula:

\[
P(u) = U \ M_B \ G_B
\]

where

\[
U = [u^3 \ u^2 \ u \ 1], \text{ as usual,}
\]

\[
M_B =
\begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 3 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\]
is the Bezier matrix, and
$G_B = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$ is the Bezier geometry vector.

Trying this on the figure shown above we have (combining the $x$ and $y$ formulae):

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 2 \\ 1 & -1 \\ 2 & 0 \end{bmatrix}$$

and so

$$x(u) = -x^3 + 3x^2$$
$$y(u) = 8x^3 - 12x^2 + 3x + 1$$

which is, as expected, the same curve that we got for the equivalent Hermitian system.
5. Convex Hull Property of Bezier Curves

Looking at our Bezier example, again, I’ve drawn some extra lines into the figure, and have eliminated the Hermitian arrows:

The lines, which enclose the four control points as tightly as possible, are called the convex hull. A very useful property of Bezier curves is that the generated spline must lie completely within the convex hull. So if you plot the curve that I generated, you’ll find that it starts at \( P_0 \), heads up towards \( P_1 \), makes a backwards S-shape, and finishes at \( P_3 \) coming in from the direction from \( P_2 \), without ever leaving the hull, as I’ve sketched. To see why the convex hull works, look again at the Bezier equation:

\[
P(u) = \begin{bmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}.
\]

Previously we expanded this as terms in powers of \( u \). Instead we can look at the multipliers of the control points by multiplying the first two matrices, which gives:

\[
P(u) = (-u^3 + 3u^2 - 3u + 1)P_0 + (3u^3 - 6u^2 + 3u)P_1 + (-3u^3 + 3u^2)P_2 +
\]
Defining:

\[ b_0(u) = (-u^3 + 3u^2 - 3u + 1) = (1 - u)^3 \]
\[ b_1(u) = (3u^3 - 6u^2 + 3u) = 3u(1 - u)^2 \]
\[ b_2(u) = (-3u^3 + 3u^2) = 3u^2(1 - u) \]
\[ b_3(u) = u^3 \]

These four functions are called the blending functions. They give the relative influence of each of the geometry elements as \( u \) ranges from 0 to 1. E.g., at \( u = 0 \), \( b_0 \) is 1, and the rest have no influence. At \( u = \frac{1}{2} \), \( b_0 \) and \( b_3 \) are both .125, and \( b_1 \) and \( b_2 \) are both .375.

For Bezier, the blending functions also give us the convex hull property.

First, \( P(u) = \sum_{i=0}^{3} b_i(u) P_i \), by definition. Also, \( \sum_{i=0}^{3} b_i(u) = 1 \), for any \( u \), since all of the \( u \) terms in the sum cancel out (e.g., the \( u^3 \) terms are \(-u^3, 3u^3, -3u^3, \) and \( u^3 \)), and each \( b_i \) satisfies \( 0 \leq b_i(u) \) when \( 0 \leq u \leq 1 \). Since the four non-negative values sum to 1, each of them cannot exceed 1, and so \( 0 \leq b_i(u) \leq 1 \) when \( 0 \leq u \leq 1 \). This means that every point on the curve is a weighted average of the four control points where each of the weights is positive and not greater than 1. This is like putting weights on the four control points and balancing the polygon on a point. The balance point must be inside the polygon. Similarly the curve must, at every point, lie within the convex hull.
6. Converting between Hermite and Bezier Systems

When converting between these two systems, use the geometry vector translation formulae, $DP_0 = 3(P_1 - P_0)$, $DP_3 = 3(P_3 - P_2)$. This is easiest to show with a couple of examples.

In the first example we have a Hermite system that we want to convert to Bezier.

This is an unusual curve since $P_0$ and $P_3$ are at the same point, but there are no restrictions on any of the geometry vector values, and so this is legal. In the figure below I’ve added in the Bezier points $P_1$ and $P_2$, and have also drawn in the convex hull. The curve will begin at $P_0$, head towards $P_1$, then curve back, finishing at the starting point coming in from the direction of $P_2$. It must, of course, lie completely within the convex hull, which in this case is a triangle.
In the other direction the figure below shows a Bezier system to be converted to Hermite:

The convex hull (shown in the figure) is a triangle again, but this time because one point is inside the hull formed by the other three. The Hermitian equivalent is shown below:

Once again the curve must stay within the convex hull shown in the first figure.
7. Cubic B-Spline Approximation Splines

B-splines are the most common approach to splines. Unlike Hermite and Bezier, they are approximation splines, and so they are based on finding approximation points for each control point, and then smoothly connecting those points to form the curve.

Each spline segment will depend on four consecutive control points, with the segment beginning and ending at approximations of the two middle points. So if, for example, we have \( n + 1 \) control points \( P_0, \ldots, P_n \), then there will be \( n - 1 \) smoothly connected segments that approximate \( P_1 \rightarrow P_2, P_2 \rightarrow P_3, \ldots \), and \( P_{n-1} \rightarrow P_n \). We’ll say that the \( i \)th segment is the one that approximates \( P_i \rightarrow P_{i+1} \), with control points \( P_{i-1}, P_i, P_{i+1}, P_{i+2} \), and so these four points will be the values in the \( i \)th geometry vector, \( G_i \).

I’ll do the derivation of the B-spline matrix in a backwards direction. First I’ll give the formula, and then show that it gives the results that we need in terms of \( C^{(0)}, C^{(1)}, \) and \( C^{(2)} \) continuity.

\[
P^i(u) = U M S G^i
\]

where

\[
U = [u^3 \ u^2 \ u \ 1] \text{ as usual,}
\]

\[
M_S = \frac{1}{6}
\begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 0 & 3 & 0 \\
1 & 4 & 1 & 0
\end{bmatrix}
\]

\[
G^i = \begin{bmatrix}
P_{i-1} \\
P_i \\
P_{i+1} \\
P_{i+2}
\end{bmatrix}
\]

This needs to be shown to meet the continuity requirements with the next patch, \( P^{i+1}(u) \), whose geometry vector will be
\[
G_{s}^{i+1} = \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}.
\]

For \(C^{(0)}\), \(C^{(1)}\), and \(C^{(2)}\) continuity we need \(P'(1) = P^{i+1}(0)\), \(P''(1) = P^{i+1'}(0)\), and \(P'''(1) = P^{i+1''}(0)\).

\[
P^{i}(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_{i} \\ P_{i+1} \\ P_{i+2} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_{i} \\ P_{i+1} \\ P_{i+2} \end{bmatrix}
= \frac{1}{6} (P_i + 4P_{i+1} + P_{i+2})
\]

\[
P^{i+1}(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_{i} \\ P_{i+1} \\ P_{i+2} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_{i} \\ P_{i+1} \\ P_{i+2} \end{bmatrix}
= \frac{1}{6} (P_i + 4P_{i+1} + P_{i+2})
\]

\[
P^{i'}(1) = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_{i} \\ P_{i+1} \\ P_{i+2} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 & -3 & 0 & 3 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_{i} \\ P_{i+1} \\ P_{i+2} \end{bmatrix}
= \frac{1}{6} (-3P_i + 3P_{i+2})
\]
\[ P^{i+1,0}(0) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{6} & -1 & 3 & -3 \ 3 & -6 & 3 & 0 \ -3 & 0 & 3 & 0 \ 1 & 4 & 1 & 0 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix} \]

\[ = \frac{1}{6} (-3P_i + 3P_{i+2}) \]

\[ P^{i,1}(1) = \begin{bmatrix} 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{6} & -1 & 3 & -3 \ 3 & -6 & 3 & 0 \ -3 & 0 & 3 & 0 \ 1 & 4 & 1 & 0 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 & 6 & -12 & 6 \ 6P_i - 12P_{i+1} + 6P_{i+2} \end{bmatrix} \]

\[ P^{i+1,1}(0) = \begin{bmatrix} 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{6} & -1 & 3 & -3 \ 3 & -6 & 3 & 0 \ -3 & 0 & 3 & 0 \ 1 & 4 & 1 & 0 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 6 & -12 & 6 & 0 \ 6P_i - 12P_{i+1} + 6P_{i+2} \end{bmatrix} \]

Computing the approximation points on the B-spline curve: Let \( Q_i \) be the point that approximates \( P_i \). So \( Q_i = P^0(0) = \frac{1}{6} (P_{i-1} + 4P_i + P_{i+1}) \). Say one has the 11 control points shown below. We could use this formula and compute, for example, \( Q_6 = \frac{1}{6} ((6, 12) + 4 (12, 6) + (9, 3)) = \frac{1}{6} (63, 39) = (10.5, 6.5) \), but this is both error-prone and tedious for 11 points. The way that I prefer to use is to assume that \( P_{i-1} = P_i + \alpha \) and \( P_{i+1} = P_i + \beta \), where \( \alpha \) and \( \beta \) are the distance, in point space, that the surrounding points are from the point that is being approximated. Now we can rewrite our formula for \( Q_i \) as:
\[ Q_i = P_i + \frac{1}{6} (\alpha + \beta) \]

E.g., if we are looking at \( P_6 \), to get to \( P_5 \) we add (-6, 6) as an \((x, y)\) difference, and to get to \( P_7 \) we add (-3, -3). The sum of these two is (-9, 3), and so \( Q_6 \), relative to \( P_6 \) is \( \frac{a}{6} \) in the \( x \) direction and \( \frac{3}{6} \) in the \( y \) direction. I show this, plus all of the other \( Q_i \) values and a sketch of the curve, on the next page.
Try a few others. E.g., $P_3$ is as much to the left of $P_4$ as $P_5$ is to the right, so their differences cancel out and $Q_4$ is located at $P_4$. To get $Q_9$, it is $\frac{1}{6}((0, -3) + (-3, 0))$ from $P_9$, which is $(-\frac{1}{2}, -\frac{1}{2})$ from $P_9$, as shown. Note that $P_0$ and $P_{10}$ influence the beginning and ending curve segments, but the curve starts at $Q_1$ and ends at $Q_9$.

**B-spline curve equations:** Say that we want the equation of the curve that approximates $P_3 \rightarrow P_6$ in the diagram above, and that the origin is in the lower left. The geometry vector will contain the points $P_4, P_5, P_6,$ and $P_7$, and so our equation will be

$$P^5(u) = [u^3 \ u^2 \ u \ 1] \cdot \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 6 \\ 12 \\ 6 \end{bmatrix}$$

$$x(u) = \frac{1}{6}(-12u^3 + 9u^2 + 27u + 39) = (-2u^3 + 1.5u^2 + 4.5u + 6.5)$$
\[ y(u) = \frac{1}{6} (9u^3 - 18u^2 - 18u + 66) = (1.5u^3 - 3u^2 - 3u + 11) \]

A quick validation check shows that \( P(0) = (6.5, 11) \) and \( P(1) = (10.5, 6.5) \), which are the values that we had for \( Q_5 \) and \( Q_6 \) in the diagram above.

**Convex hull property of cubic B-splines:** The blending functions for B-splines are:

\[
\begin{align*}
b_0(u) &= \frac{1}{6} (-u^3 + 3u^2 - 3u + 1) = \frac{1}{6} (1 - u)^3 \\
b_1(u) &= \frac{1}{6} (3u^3 - 6u^2 + 4) = \frac{1}{6} (3u(1 - u)^2 + 3(1 - u) + 1) \\
b_2(u) &= \frac{1}{6} (-3u^3 + 3u^2 + 3u + 1) = \frac{1}{6} (3u^2(1 - u) + 3u + 1) \\
b_3(u) &= \frac{1}{6} u^3.
\end{align*}
\]

For \( 0 \leq u \leq 1 \), each \( b_i \) clearly satisfies \( 0 \leq b_i(u) \), \( \sum_{i=0}^{3} b_i(u) = 1 \) since all of the \( u \) terms cancel out, and so \( 0 \leq b_i(u) \leq 1 \) for \( 0 \leq i \leq 1 \). Therefore the convex hull property holds. I.e., the curve segment that approximates \( P_i \to P_{i+1} \) lies within the convex hull defined by \( P_{i-1}, P_i, P_{i+1}, \) and \( P_{i+2} \).

If we look at the influence of the points, at \( u = 0 \), the four blending function values are \( b_0 = \frac{1}{6}, b_1 = \frac{4}{6}, b_2 = \frac{1}{6}, \) and \( b_3 = 0 \), which is the \( \frac{1}{6} (1, 4, 1) \) pattern which should by now be very familiar. If we try \( u = \frac{1}{2} \), we get \( b_0 = b_3 = \frac{1}{48}, \) and \( b_1 = b_2 = \frac{23}{48} \), and so the curve here is dominated by the middle two points.
8. Continuity between adjacent splines

**Hermite continuity:** If we have two adjacent Hermite splines, we will usually want them to be connected with at least $C^{(1)}$ continuity, which will mean that they are smoothly connected. Assume that the two geometry vectors are

\[
\begin{bmatrix}
P_{10} \\
P_{13} \\
DP_{10} \\
DP_{13}
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
P_{20} \\
P_{23} \\
DP_{20} \\
DP_{23}
\end{bmatrix}
\]

For $C^{(0)}$ continuity we need that the end point of the first is the same as the start point of the second, and so $P_{13} = P_{20}$. To also get $C^{(1)}$ we need the end vector of the first to be in the same direction as the start vector of the second, and so $kDP_{13} = DP_{20}$, for some $k > 0$. Graphically, we get the picture below. I’ve tried to clarify things by making the first poorly sketched segment, and its control values, all red, and the second segment and control values blue.

As discussed above, for the required continuity, $P_{13}$ and $P_{20}$ are the same point, and $DP_{13}$ and $DP_{20}$ are in the same direction (one is a positive multiple of the other).
**Bezier continuity:** This is similar to Hermite except, of course, that we are using the Bezier geometry vector. Assume that the two vectors are:

\[
\begin{bmatrix}
P_{10} \\
P_{11} \\
P_{12} \\
P_{13}
\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}
P_{20} \\
P_{21} \\
P_{22} \\
P_{23}
\end{bmatrix}
\]

For continuity we need \( P_{13} = P_{20} \) (end points match) and also that the points \( P_{12}, P_{13}/P_{20} \), and \( P_{21} \) lie in a straight line with \( P_{13}/P_{20} \) between the other two. This is shown in the example below:

**B-Spline continuity:** This is automatic by the setup. If we define one B-spline with the control points \( P_{i-1}, P_i, P_{i+1}, \) and \( P_{i+2} \), and the next one with \( P_i \) \( P_{i+1}, P_{i+2}, \) and \( P_{i+3} \), then the two curve segments will automatically be connected with \( C^{(2)} \) continuity.
9. Comparing Hermite, Bezier, and B-Spline

Of the three, Hermite is the least used, because the user interface for Bezier is much more convenient since the convex hull property helps the user predict their curve much better. The major reason for even mentioning Hermite in these notes is that the mathematical development of Bezier is much easier going through Hermite.

If you want an interpolating spline, then Bezier is the most common choice. However it is possible to use knot vector control to force B-spline to go accurately through control points, and so sometimes this is the approach taken to interpolation.

B-spline has the most flexibility. In these notes I’ve concentrated on cubic uniform, non-rational, B-splines. In their most powerful form (Non-Uniform Rational B-Splines, or NURBS) they provide considerable power. For example, the three curve systems that I’ve described here cannot be used to accurately generate a circle, but NURBS can do this.
10. Knot Vectors and B-Splines

In these notes I’ve concentrated on simple cubic B-splines. Most graphics packages provide much more control over the B-spline functions, and we’ll discuss this further in the later set of notes. In this section I’ll give a cook book approach to using these functions.

First, some confusion. If we are using B-spline polynomials with degree \( n \) (e.g., the cubic B-splines had degree 3), then the order of the spline is usually referred to in the literature as \( 1 + n \). This gets confusing because the degree of a polynomial is also often called the order. However to be consistent with the B-spline literature I’ll name the order of the spline to be \( 1 + \) the polynomial degree of the blending functions that define the curve. So our cubic B-splines have order 4. To add further confusion, the order is traditionally given the name \( d \). The order of the curve gives the number of curve segments that will be influenced by any control point. E.g., in our cubic B-splines, where \( d = 4 \), the point \( P_5 \) influenced the curve segments \( P^3(u) \), \( P^4(u) \), \( P^5(u) \), and \( P^6(u) \), since it was in their geometry vectors.

A knot vector is a vector of non-decreasing values whose length is \( d + n \), where \( n \) is the number of control points and \( d \) is the order of the spline. E.g., in our cubic B-spline example earlier which had 11 control points \( P_0 \ldots P_{10} \), we would need a knot vector with 15 values.

In that example, where we were using pure cubic B-splines, the curve completely missed the control points \( P_0 \) and \( P_{10} \). In general this isn’t what we want. What is usually required is that the curve accurately goes through the control points \( P_0 \) and \( P_{n-1} \) (assuming that there are \( n \) control points), and then smoothly approximates the intermediate points. To do this we assign an open uniform knot vector, which is the vector

\[
kv = (0, 0, \ldots, 0, 1, 2, \ldots, n-d, n-d+1, \ldots, n-d+1)
\]

where there are \( d \) zeros at the beginning, and \( d \) equal values at the end, and the intermediate values each increase by 1. This probably sounds confusing, so a couple of examples are needed. When we had a cubic \((d = 4)\) earlier with 11 control points, if we wanted the curve to start at \( P_0 \) and end at \( P_{10} \) then we would use the knot vector \((0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 8, 8)\), which has length 15, multiplicity of 4 at each end, and increases by 1 in the middle.
In a later set of notes we’ll see that a cubic B-spline with an open uniform knot vector over four control points is equivalent to a cubic Bezier over the same points.

A more extreme example would be if we wanted to draw a quadratic spline with only three control points, starting at the first and ending at the third. The knot vector for this would be \((0, 0, 0, 1, 1, 1)\) since \((2 + 1) + 3 = 6\), and the multiplicity at each end must be \((2 + 1)\).

So when using a graphics package that requires that you specify the order of the spline curve and a knot vector, then a legal knot vector is any vector that (a) has length \(\text{order} + \# \text{ of control points}\), and (b) has non-decreasing values. E.g., for a linear \((d = 2)\) spline with five control points, any of the following are legal knot vectors:

\[
\begin{align*}
(0, 0, 1, 2, 3, 4, 4) \\
(0, 1, 2, 3, 4, 5, 5) \\
(7, 7, 7, 7, 7, 7) \\
(2, 3, 3, 4.5, 5, 6, 6)
\end{align*}
\]

but only the first is the open uniform knot vector which will force the curve to begin and end on the first and last control points, respectively.
11. Spline Surfaces

Spline surfaces, in any of the three flavors that I’ve looked at, can be considered as simple spline systems where the data are simple spline curves instead of points.

Using Bezier as an example, we have

\[ S(u, v) = [u^3 \ u^2 \ u \ 1] \ M_B \begin{bmatrix} \ P_0(v) \\ \ P_1(v) \\ \ P_2(v) \\ \ P_3(v) \end{bmatrix} \]

Where we are defining a surface with two parameters, \( u \) and \( v \), and the geometry vector now becomes four Bezier curves instead of four points.

Each \( P_i(v) \) will have the form

\[ P_i(v) = [v^3 \ v^2 \ v \ 1] \ M_B \begin{bmatrix} \ P_{i0} \\ \ P_{i1} \\ \ P_{i2} \\ \ P_{i3} \end{bmatrix} \]

Stacking four of these together gives:

\[
\begin{bmatrix} \ P_0(v) & P_1(v) & P_2(v) & P_3(v) \end{bmatrix} = [v^3 \ v^2 \ v \ 1] \ M_B \begin{bmatrix} \ P_{00} & P_{01} & P_{02} & P_{03} \\ \ P_{10} & P_{11} & P_{12} & P_{13} \\ \ P_{20} & P_{21} & P_{22} & P_{23} \\ \ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix}
\]
Taking transposes, and remembering that \((ABC)^T = C^TB^TA^T\), we can substitute for the column vector in the original equation and get:

\[
S(u, v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M_B \begin{bmatrix} \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M_B^T \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}.
\]

In the case of Bezier, this gives us 16 control points where the surface patch will have corner points at the four corner control values, and the four edges of the patch will be defined by the Bezier curves defined by the top and bottom rows and the first and last columns of the geometry matrix.

For any spline system the general bicubic patch form will be:

\[
S(u, v) = U M G M^T V^T
\]

Where \(M\) is the matrix for that spline system, \(G\) is a 4\times4 geometry matrix, \(U = [u^3 \ u^2 \ u \ 1]\) and \(V = [v^3 \ v^2 \ v \ 1]\).