

Out[1]: ([Show / Hide code](#))

Bézier Surfaces

CSE 4303 / CSE 5365 Computer Graphics

2020 Fall Semester, Version 1.1, 2020 December 04

Expressing an object in terms of triangles whose corners are vertices with specific positions is simple. Since individual triangles are *planar* though, wherever two triangles share vertices there's a distinct edge between them which can be painfully obvious in the render. One way to solve this problem is to make the object out of smaller triangles. This helps but even though the triangles are smaller, they are still planar and so there's still a *discontinuity* where any two meet (unless the two triangles happen to be *co-planar*). Still, maybe this discontinuity won't be a problem since any render ultimately results in the creation of individual pixels and if the triangles are small enough, the discontinuities might be unnoticeable.

Once we are lead to the notion of just making smaller triangles, we stumble across several issues,

- Expressing objects in terms of smaller and smaller triangles means making lots and lots of teeny, tiny triangles and if this has to be done manually, it will be incredibly *tedious*.
- Since different uses of an object might require different render *resolutions*, we might have to have a different version of the object for each particular usage scenario, each specifically handcrafted for that use. This would be a considerable repetition of work and keeping all of those versions in sync as changes are made would be tedious and error-prone.
- The triangle representation is easy to process when rendering, but it doesn't correspond to how designers or artists think about objects when they are creating them. Triangles are not *intuitive*.

Given all this, what we want is an intuitive way for designers and artists to create a shape *once* that can then be made as fine-grain as necessary in an automatic and efficient way.

Bézier curves are a way to capture the overall shape of an object while postponing the actual level of detail until the shape is rendered. Bézier surfaces can be made as smooth as desired, consistent with how much resolution one needs and how much computational effort one wishes to expend.

Before we dive into a discussion of Bézier however, let's quickly review parametric curves.

(Simple) Parametric Curves

A parametric curve is an ordered set of equations each defining a function in terms of some number of independent variables. The value of the parametric curve at any point is the ordered collection of the values of each of the functions at that point. For example, an n -dimensional parametric curve of one independent variable t has the form

$$\mathbf{F}(t) = (f_0(t), f_1(t), \dots, f_{n-1}(t))$$

To avoid needless repetition, we will assume $t \in [0, 1]$ and $f_i(t) \in \mathbb{R}$ for all of our parametric curve definitions unless otherwise specified.

Since we are primarily interested in two-dimensional and three-dimensional objects, the majority of our parametric curves will be defined as,

Two dimensional

$$\mathbf{F}(t) = (x(t), y(t))$$

or

Three dimensional

$$\mathbf{F}(t) = (x(t), y(t), z(t))$$

As with traditional coordinates, there are *homogeneous* forms of the parametric curve representation. These are defined as,

Two dimensional homogeneous

$$\mathbf{H}(t) = (x(t), y(t), w(t))$$

or

Three dimensional homogeneous

$$\mathbf{H}(t) = (x(t), y(t), z(t), w(t))$$

Example

The classic example used to demonstrate a parametric curve is that of the unit circle centered at the origin. First, the more traditional representation,

$$x^2 + y^2 = 1$$


This representation is certainly correct, but not as useful as it might be when one wants to plot an *approximate* circle with an *arbitrary* resolution. The parametric version of this unit circle is $\mathbf{C}(t) = (x(t), y(t))$ as we are considering a two-dimensional representation. Using basic trigonometry,

we can easily fill in the definitions for the x and y functions,

$$\mathbf{C}(t) = (\cos(2\pi t), \sin(2\pi t))$$

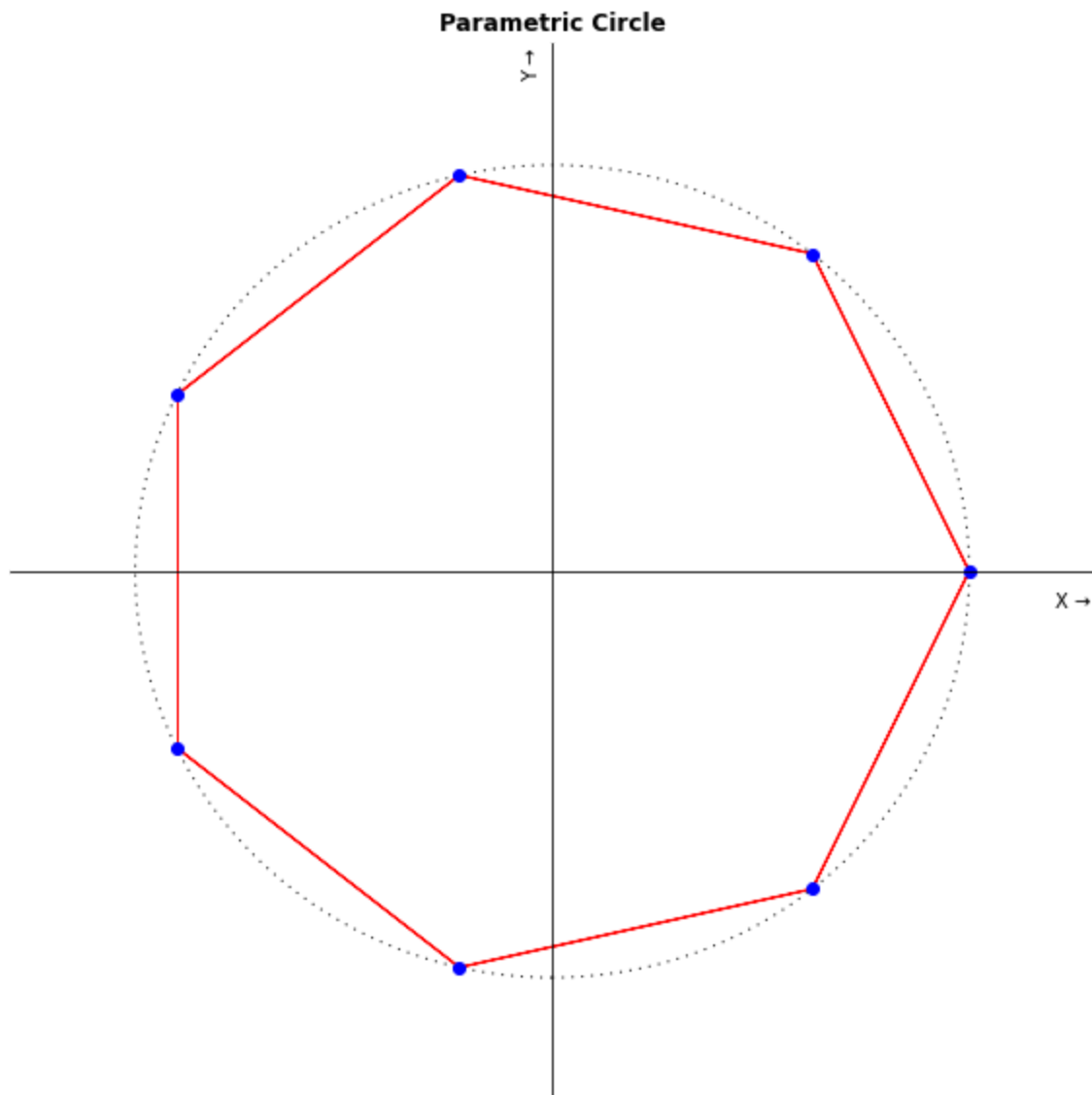
To describe the complete circle, we need the arguments to the \cos and \sin functions to run from 0 to 2π . Since t runs from 0 to 1 we multiply it by 2π to get the desired span. Now we can plot the unit circle to any resolution we desire simply by evaluating $\mathbf{C}(t)$ for a suitable number of values of t between 0 and 1.

In the following example, the ideal unit circle is drawn as a dotted line. The parametric version is a collection of points connected by line segments. The more points and line segments (that is, the greater the *resolution* of the parametric version), the closer the parametric version approaches the ideal unit circle. Eventually, when the resolution is high enough, the parametric circle becomes indistinguishable from the ideal unit circle. Exactly how high "high enough" has to be depends on the requirements of the exact usage scenario.

Resolution 

7

☒ Show Points



There's a lot more to parametric functions, of course, but we can get through quite a bit of Bézier curves

and surfaces with just this level of description.

Linear Bézier Curves

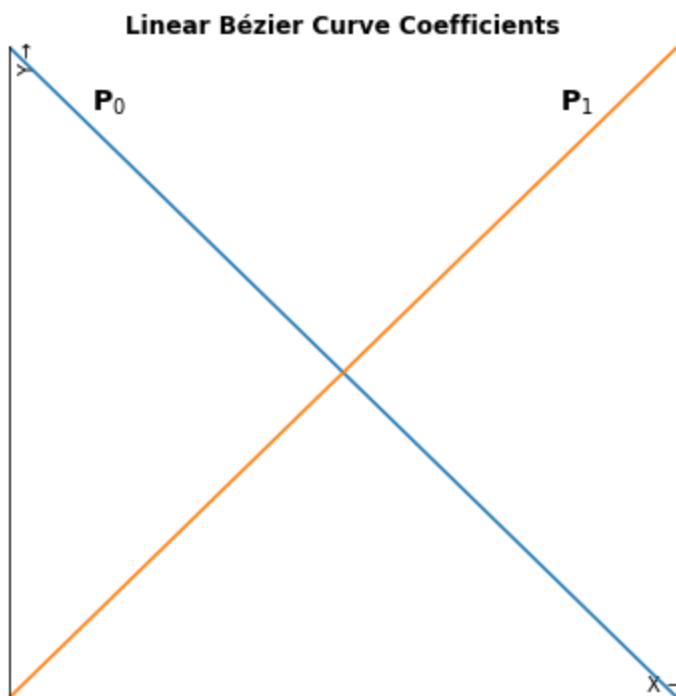
Bézier curves are classified by how many *control points* they have. The simplest Bézier curve is *linear* and has only two control points, P_0 and P_1 . It's not a very interesting "curve", as it is just a line between the two points. However, we start with it because the general method for higher-order curves is built on this foundation.

The parametric definition of a linear Bézier curve is,

$$\mathbf{B}^1(t) = (1 - t) \cdot \mathbf{P}_0 + t \cdot \mathbf{P}_1$$

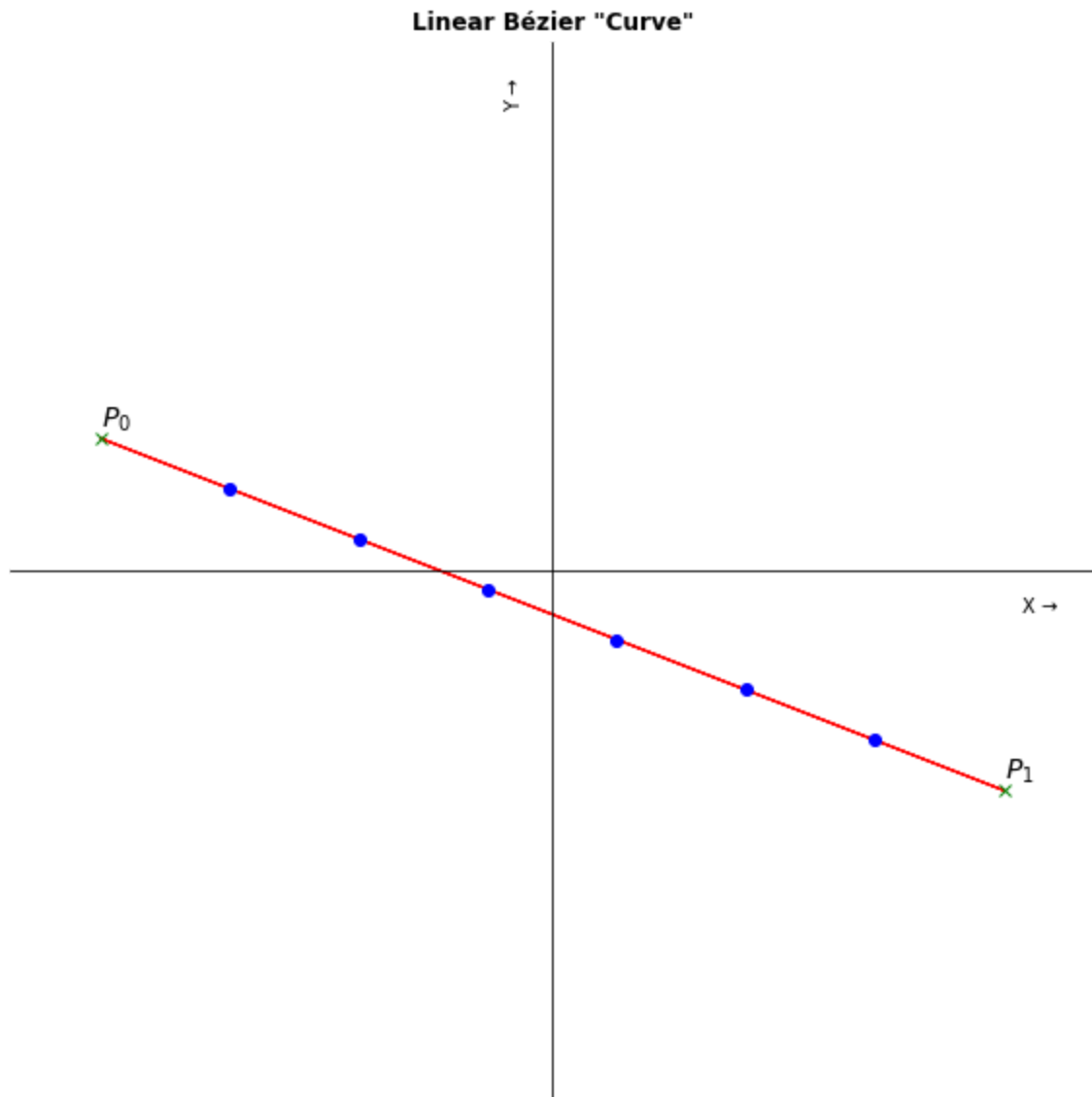
This form should be familiar. $\mathbf{B}^1(t)$ is simply a *linear combination* of P_0 and P_1 where the coefficients are $1 - t$ and t . Intuitively, the meaning of this definition should be clear: as t goes from 0 to 1, the value of $\mathbf{B}^1(t)$ goes from \mathbf{P}_0 to \mathbf{P}_1 , blending the values of the two points for any value of t in-between.

The coefficients of \mathbf{P}_0 and \mathbf{P}_1 as t goes from 0.0 to 1.0 are shown in the following graph,



Here's an example,

p_{0x}	<input type="text" value="-5.0"/>	p_{1x}	<input type="text" value="5.0"/>
p_{0y}	<input type="text" value="1.5"/>	p_{1y}	<input type="text" value="-2.5"/>
Resolution	<input type="text" value="8"/>	<input checked="" type="checkbox"/> Show Points	



$$\mathbf{B}^1(t) = (1 - t) \cdot (-5.0, 1.5) + t \cdot (5.0, -2.5)$$

If we take the definition of \mathbf{B}^1 and expand it completely for a given set of values for \mathbf{P}_0 and \mathbf{P}_1 , we can get a clearer understanding of the ultimate shape of the curve.

Out[6]: As a particular example, given $\mathbf{P}_0 = (-5.0, 1.5)$ and $\mathbf{P}_1 = (5.0, -2.5)$, we can substitute those values in to the definition of $\mathbf{B}^1(t)$ and obtain the $x(t)$ and $y(t)$ functions.

$$\begin{aligned}\mathbf{B}^1(t) &= (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \\ &= (1-t) \cdot (-5.0, 1.5) + t \cdot (5.0, -2.5)\end{aligned}$$

$$\begin{aligned}x(t) &= (-5.0)(1-t) + (5.0)t \\ &= 10.0t - 5.0\end{aligned}$$

$$\begin{aligned}y(t) &= (1.5)(1-t) + (-2.5)t \\ &= 1.5 - 4.0t\end{aligned}$$

If we rearrange the x component of the curve to solve for t in terms of x , we find,

$$\begin{aligned}x(t) &= 10.0t - 5.0 \\ t(x) &= 0.1x + 0.5\end{aligned}$$

Substituting that representation of t into the y component of the curve, we obtain the explicit version of the curve,

$$\begin{aligned}y(x) &= y(t(x)) \\ &= 1.5 - 4.0t \quad \text{with} \quad t(x) = 0.1x + 0.5 \\ &= -0.4x - 0.5\end{aligned}$$

Clearly, this $\mathbf{B}^1(t)$ is a simply a straight line.

OK, so it works. We get a line between \mathbf{P}_0 and \mathbf{P}_1 with as many intermediate points as we desire. Since we already knew how to draw a line between two points, this doesn't seem all that interesting but it will be when we consider the next-higher-order curve, the *Quadratic Bézier curve*.

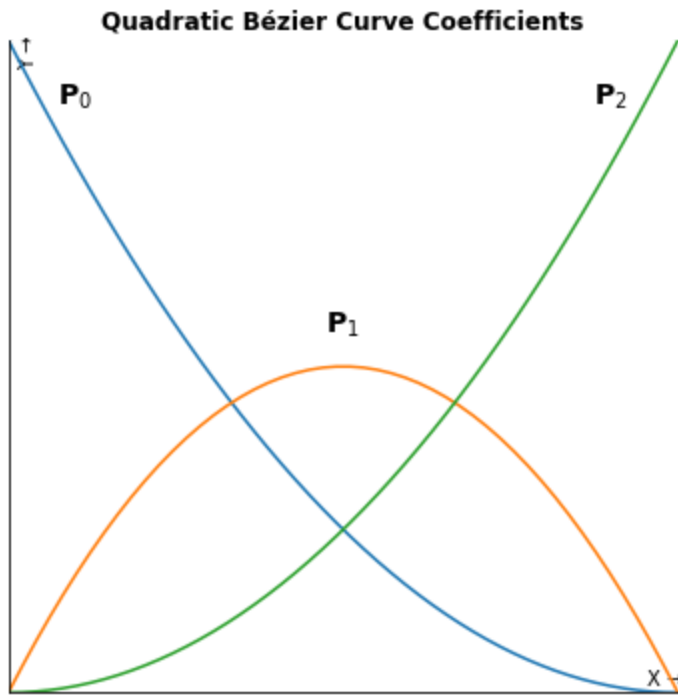
Quadratic Bézier Curves

The next-higher order of Bézier curve is *Quadratic*. Instead of two control points, the quadratic form has *three*, \mathbf{P}_0 , \mathbf{P}_1 , and \mathbf{P}_2 . The definition of this order curve is built from the linear curve in a straightforward way.

Just as the linear curve is a linear combination of two *points*, the quadratic curve is a linear combination of the two *linear curves* formed by the three control points. The first linear curve $\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1}^1$ is from \mathbf{P}_0 to \mathbf{P}_1 and the second linear curve $\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2}^1$ is from \mathbf{P}_1 to \mathbf{P}_2 . The derivation goes as follows,

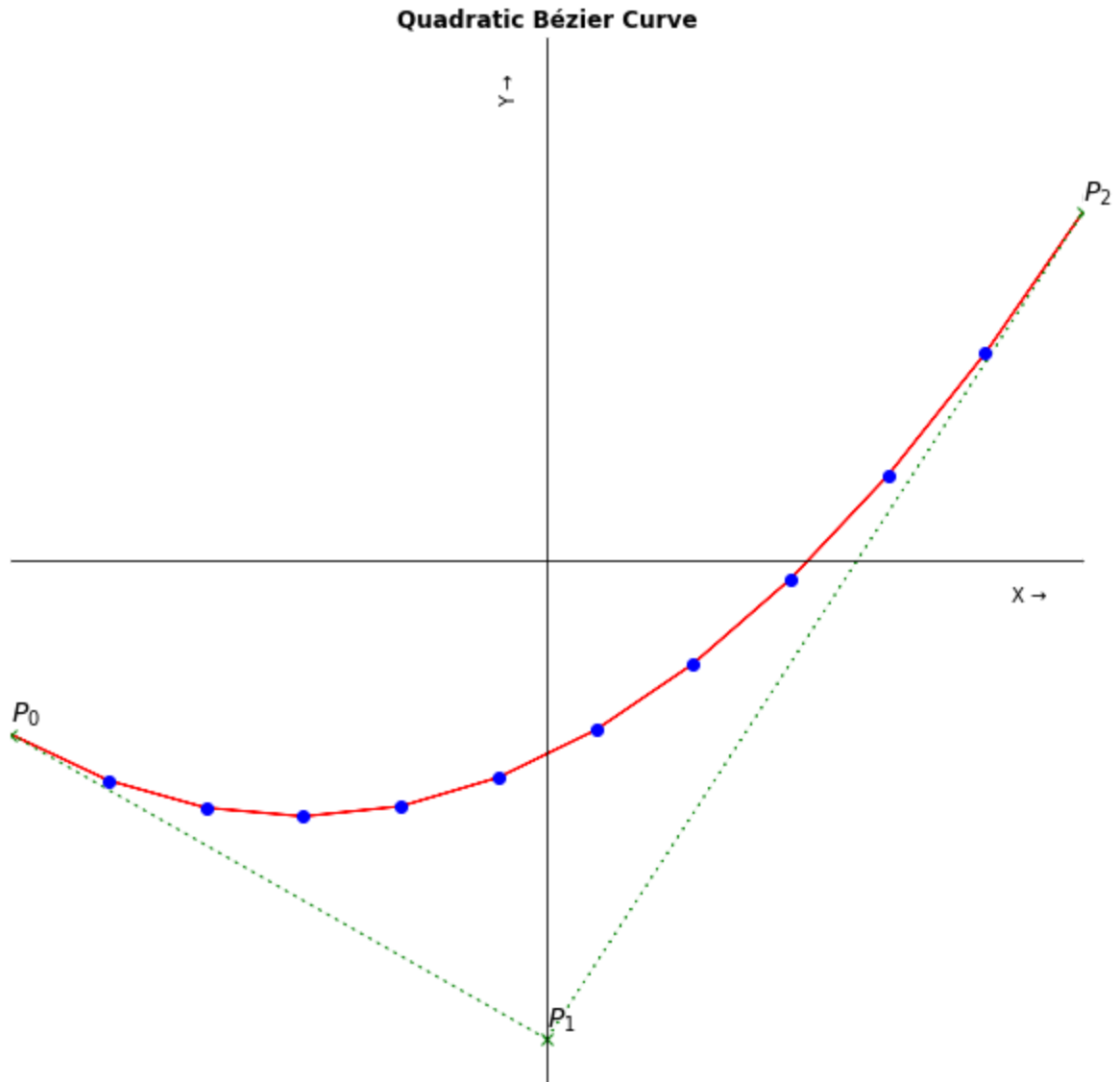
$$\begin{aligned}\mathbf{B}^2(t) &= (1-t)\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1}^1 + t\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2}^1 \\ &= (1-t)[(1-t)\mathbf{P}_0 + t\mathbf{P}_1] + t[(1-t)\mathbf{P}_1 + t\mathbf{P}_2] \\ &= (1-t)^2\mathbf{P}_0 + t(1-t)\mathbf{P}_1 + t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2 \\ &= (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2\end{aligned}$$

$\mathbf{B}^2(t)$ is a linear combination of \mathbf{P}_0 , \mathbf{P}_1 , and \mathbf{P}_2 where the coefficients are $(1 - t)^2$, $2t(1 - t)$, and t^2 . These coefficients are shown in the following graph for t from 0.0 to 1.0,



Here's an example,

p_{0x}	<input type="text" value="-6.0"/>	p_{2x}	<input type="text" value="6.0"/>
p_{0y}	<input type="text" value="-2.0"/>	p_{2y}	<input type="text" value="4.0"/>
Resolution	<input type="text" value="12"/>	<input checked="" type="checkbox"/> Show Points	
p_{1x}	<input type="text" value="0.0"/>	p_{1y}	<input type="text" value="-5.5"/>



$$\mathbf{B}^2(t) = (1-t)^2 \cdot (-6.0, -2.0) + 2t(1-t) \cdot (0.0, -5.5) + t^2 \cdot (6.0, 4.0)$$

If we take the definition of \mathbf{B}^2 and expand it completely for a given set of values for \mathbf{P}_0 , \mathbf{P}_1 , and \mathbf{P}_2 , we can get a clearer understanding of the ultimate shape of the curve.

Out[9]: As a particular example, given $\mathbf{P}_0 = (-2, 4)$, $\mathbf{P}_1 = (0, -4)$, and $\mathbf{P}_2 = (2, 4)$, we can substitute those values in to the definition of $\mathbf{B}^2(t)$ and obtain the $x(t)$ and $y(t)$ functions.

$$\begin{aligned}\mathbf{B}^2(t) &= (1-t)^2 \mathbf{P}_0 + 2t(1-t) \mathbf{P}_1 + t^2 \mathbf{P}_2 \\ &= (1-t)^2 \cdot (-2, 4) + 2t(1-t) \cdot (0, -4) + t^2 \cdot (2, 4)\end{aligned}$$

$$\begin{aligned}x(t) &= (-2)(1-t)^2 + (0)2t(1-t) + (2)t^2 \\ &= 4t - 2\end{aligned}$$

$$\begin{aligned}y(t) &= (4)(1-t)^2 + (-4)2t(1-t) + (4)t^2 \\ &= 16t^2 - 16t + 4\end{aligned}$$

If we rearrange the x component of the curve to solve for t in terms of x , we find,

$$\begin{aligned}x(t) &= 4t - 2 \\ t(x) &= \frac{x+2}{4}\end{aligned}$$

Substituting that representation of t into the y component of the curve, we obtain the explicit version of the curve,

$$\begin{aligned}y(x) &= y(t(x)) \\ &= 16t^2 - 16t + 4 \quad \text{with} \quad t(x) = \frac{x+2}{4} \\ &= (x+2)^2 - 4(x+2) + 4 \\ &= x^2\end{aligned}$$

Clearly, this $\mathbf{B}^2(t)$ is a parabola.

Examining the derivation of $\mathbf{B}^2(t)$ closely, we see that the only ways it would turn out *not* to be a parabola would be if,

1. At least two of the three control points are distinct and the distinct points are colinear. In this case, $\mathbf{B}^2(t)$ is a *line*.
2. The three control points are all the same. In this case, $\mathbf{B}^2(t)$ is a single *point*.

An interesting characteristic to notice about the quadratic Bézier curve is that the second control point \mathbf{P}_1 is usually *not* on the curve between control points \mathbf{P}_0 and \mathbf{P}_2 . This is true for all higher-order Bézier curves as well. Aside from the first (\mathbf{P}_0) and last (\mathbf{P}_n) control points, the control points will be on the generated curve only if the distinct control points are colinear. These intermediate control points are for the control of the shape of the curve only.

The dotted lines from \mathbf{P}_0 to \mathbf{P}_1 and from \mathbf{P}_1 to \mathbf{P}_2 show how the generated curve fits; it's tangent to these lines at \mathbf{P}_0 and \mathbf{P}_2 . The entire quadratic Bézier curve fits inside the triangle formed by the three control points.

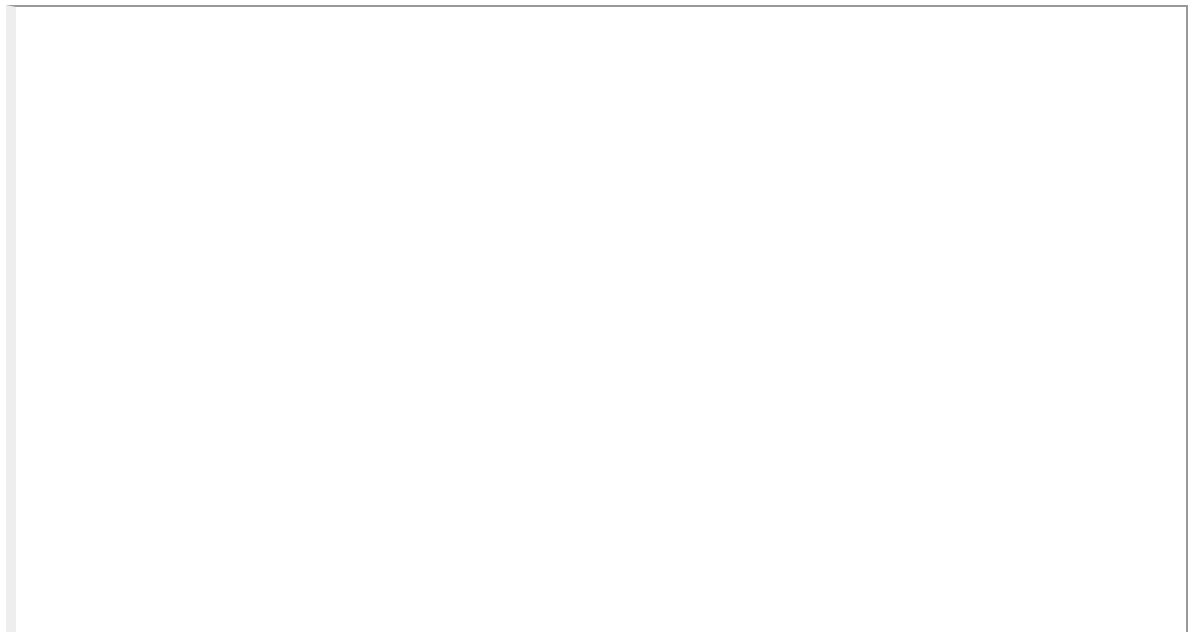
Quadratic Bézier curves are certainly more useful than their linear predecessors, but parabolas can take one only so far. We can generate even better curves if we go to the next higher order.

Cubic Bézier Curves

The next-higher order of Bézier curve is *Cubic*. Instead of three control points, the cubic form has *four*, \mathbf{P}_0 , \mathbf{P}_1 , \mathbf{P}_2 , and \mathbf{P}_3 . The cubic Bézier curve is built from the quadratic curve in the same way that the quadratic curve was built from the linear curve.

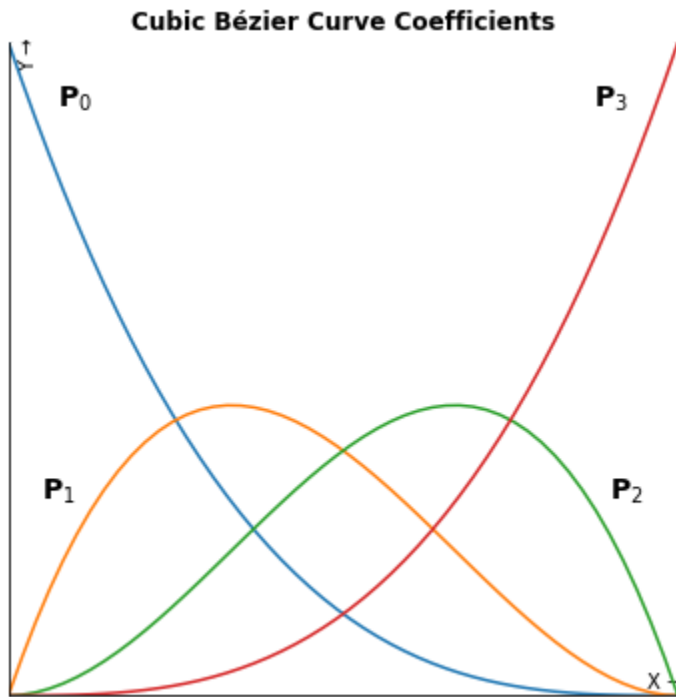
The cubic curve $\mathbf{B}^3(t)$ is a linear combination of the two quadratic curves formed by the four control points. The first quadratic curve $\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2}^2$ includes control points \mathbf{P}_0 , \mathbf{P}_1 , and \mathbf{P}_2 and the second quadratic curve $\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3}^2$ includes the control points \mathbf{P}_1 , \mathbf{P}_2 , and \mathbf{P}_3 .

Starting with that definition and going through a bunch of algebraic manipulation and simplification, we can derive the following,



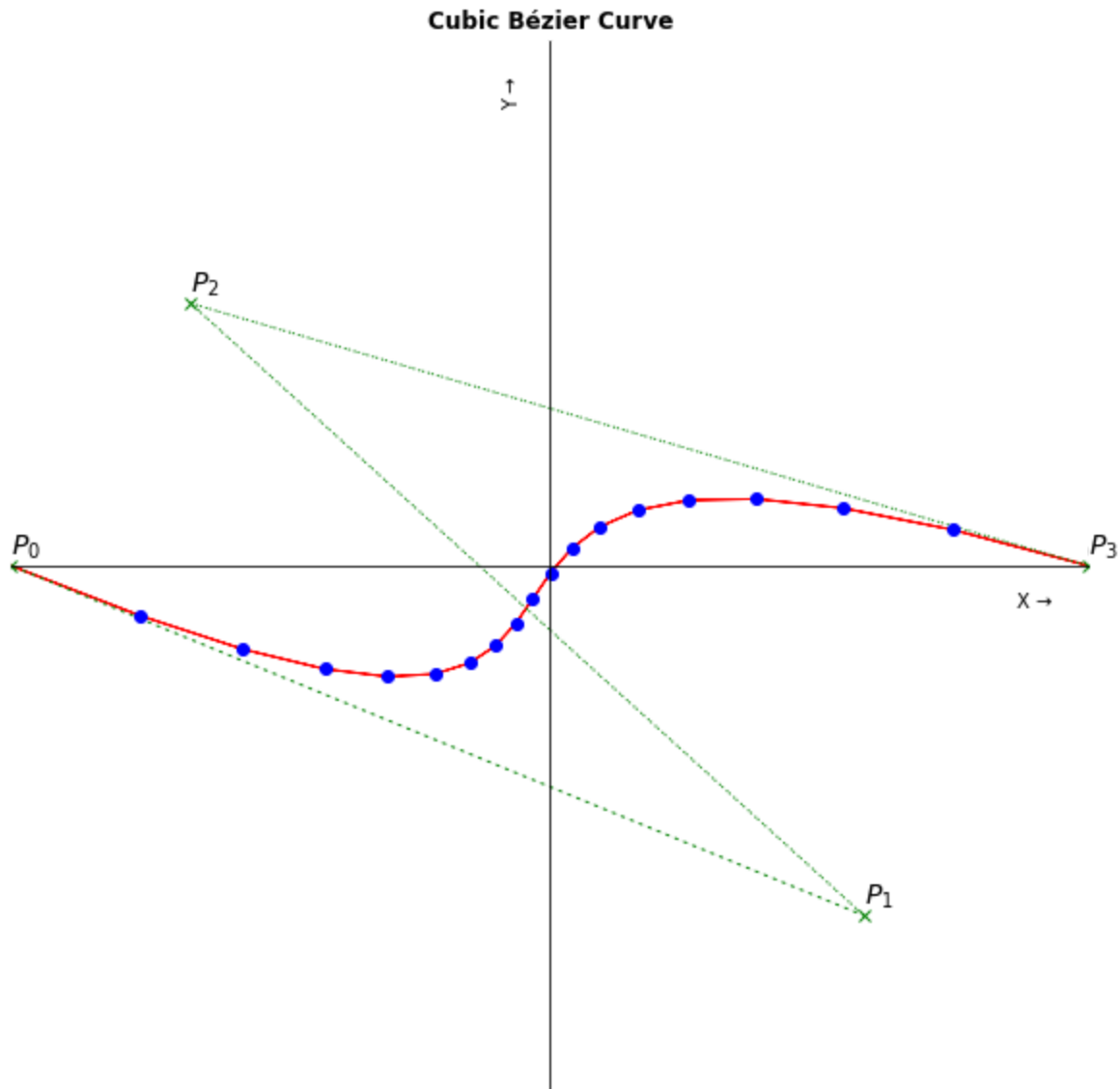
$$\begin{aligned}
\mathbf{B}^3(t) &= (1-t)\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2}^2 + t\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3}^2 \\
&= (1-t)[(1-t)\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1}^1 + t\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2}^1] + t[(1-t)\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2}^1 + t\mathbf{B}_{\mathbf{P}_2\mathbf{P}_3}^1] \\
&= (1-t)[(1-t)\langle(1-t)\mathbf{P}_0 + t\mathbf{P}_1\rangle + t\langle(1-t)\mathbf{P}_1 + t\mathbf{P}_2\rangle] \\
&\quad + t[(1-t)\langle(1-t)\mathbf{P}_1 + t\mathbf{P}_2\rangle + t\langle(1-t)\mathbf{P}_2 + t\mathbf{P}_3\rangle] \\
&= (1-t)^2\langle(1-t)\mathbf{P}_0 + t\mathbf{P}_1\rangle + t(1-t)\langle(1-t)\mathbf{P}_1 + t\mathbf{P}_2\rangle \\
&\quad + t(1-t)\langle(1-t)\mathbf{P}_1 + t\mathbf{P}_2\rangle + t^2\langle(1-t)\mathbf{P}_2 + t\mathbf{P}_3\rangle \\
&= (1-t)^3\mathbf{P}_0 + t(1-t)^2\mathbf{P}_1 + t(1-t)^2\mathbf{P}_1 + t^2(1-t)\mathbf{P}_2 \\
&\quad + t(1-t)^2\mathbf{P}_1 + t^2(1-t)\mathbf{P}_2 + t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3 \\
&= (1-t)^3\mathbf{P}_0 + t(1-t)^2\mathbf{P}_1 + t(1-t)^2\mathbf{P}_1 + t(1-t)^2\mathbf{P}_1 \\
&\quad + t^2(1-t)\mathbf{P}_2 + t^2(1-t)\mathbf{P}_2 + t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3 \\
&= (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3
\end{aligned}$$

$\mathbf{B}^3(t)$ is a linear combination of \mathbf{P}_0 , \mathbf{P}_1 , \mathbf{P}_2 , and \mathbf{P}_3 where the coefficients are $(1-t)^3$, $3t(1-t)^2$, $3t^2(1-t)$, and t^3 . These coefficients are shown in the following graph for t from 0.0 to 1.0,



Here's an example,

p_{0x}	<input type="text" value="-6.0"/>	p_{3x}	<input type="text" value="6.0"/>
p_{0y}	<input type="text" value="0.0"/>	p_{3y}	<input type="text" value="0.0"/>
Resolution	<input type="text" value="19"/>	<input checked="" type="checkbox"/> Show Points	
p_{1x}	<input type="text" value="3.5"/>	p_{2x}	<input type="text" value="-4.0"/>
p_{1y}	<input type="text" value="-4.0"/>	p_{2y}	<input type="text" value="3.0"/>



$$\mathbf{B}^3(t) = (1-t)^3 \cdot (-6.0, 0.0) + 3(1-t)^2t \cdot (3.5, -4.0) + 3(1-t)t^2 \cdot (-4.0, 3.0) + t^3 \cdot (6.0, 0.0)$$

With the two intermediate control points \mathbf{P}_1 and \mathbf{P}_2 , we can form much more interesting shapes than with linear or even quadratic Bézier curves. Similar to the quadratic case, the dotted lines from \mathbf{P}_0 to \mathbf{P}_1 , from \mathbf{P}_1 to \mathbf{P}_2 , and from \mathbf{P}_2 to \mathbf{P}_3 show how the generated curve fits; it's tangent to these lines at \mathbf{P}_0 and \mathbf{P}_3 . The entire cubic Bézier curve fits in the quadrilateral formed by the four control points.

Also as with the quadratic case, we can derive the explicit representation of the curve for a given set of \mathbf{P}_0 , \mathbf{P}_1 , \mathbf{P}_2 , and \mathbf{P}_3 values and determine its characteristics.

Out[12]: As a particular example, given $\mathbf{P}_0 = (-2, 0)$, $\mathbf{P}_1 = (-1, 6)$, $\mathbf{P}_2 = (0, -3)$, and $\mathbf{P}_3 = (1, 0)$, we can substitute those values into the definition of $\mathbf{B}^3(t)$ and obtain the expressions for $y(t)$ and $x(t)$.

$$\begin{aligned}\mathbf{B}^3(t) &= (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3 \\ &= (1-t)^3 \cdot (-2, 0) + 3t(1-t)^2 \cdot (-1, 6) + 3t^2(1-t) \cdot (0, -3) + t^3 \cdot (1, 0) \\ x(t) &= (-2)(1-t)^3 + (-1)3t(1-t)^2 + (0)3t^2(1-t) + (1)t^3 \\ &= 3t - 2 \\ y(t) &= (0)(1-t)^3 + (6)3t(1-t)^2 + (-3)3t^2(1-t) + (0)t^3 \\ &= 27t^3 - 45t^2 + 18t\end{aligned}$$

If we rearrange the x component of the curve to solve for t in terms of x , we find,

$$\begin{aligned}x(t) &= 3t - 2 \\ t(x) &= \frac{x + 2}{3}\end{aligned}$$

Substituting that representation of t into the y component of the curve, we obtain the explicit version of the curve,

$$\begin{aligned}y(x) &= y(t(x)) \\ &= 27t^3 - 45t^2 + 18t \quad \text{with} \quad t(x) = \frac{x + 2}{3} \\ &= (x + 2)^3 - 5(x + 2)^2 + 6(x + 2) \\ &= x^3 + x^2 - 2x\end{aligned}$$

Clearly, this $\mathbf{B}^3(t)$ is a cubic.

Examining the derivation of $\mathbf{B}^3(t)$ closely, we see that the only ways it would turn out *not* to be a cubic would be if,

1. At least three of the four control points are distinct and the distinct points result in a *parabola*.
2. At least two of the four control points are distinct and the distinct points are colinear. In this case, $\mathbf{B}^3(t)$ is a *line*.
3. The four control points are all the same. In this case, $\mathbf{B}^3(t)$ is a single *point*.

Notice how a $\mathbf{B}^3(t)$ can be used to generate a *cubic*, *parabola*, *line*, or *point*. A $\mathbf{B}^2(t)$ can be used to generate a *parabola*, *line*, or *point*. A $\mathbf{B}^1(t)$ can be used to generate a *line* or a *point*. With the proper selection of control points, a $\mathbf{B}^n(t)$ may be used to generate any curve $\mathbf{B}^m(t)$ where $m \leq n$.

Degree Lifting

Instead of looking backwards from a higher degree to a lower degree, we can look forwards from a lower degree to a higher degree. In other words, it's not hard to show that a Bézier curve of degree n can be expressed as a Bézier curve of degree $n + 1$ *without changing its shape*. This is known as *degree lifting* (or *degree elevation*). This process is useful in that we retain the original shape of the curve but afterwards have an additional control point. (An increased number of control points allows for more subtle manipulation of the shape of the curve.)

As we move from degree n to $n + 1$ we need one additional control point. That is, a Bézier curve of degree n has $n + 1$ control points, while a curve of degree $n + 1$ has $n + 2$ control points. We will label the first set as \mathbf{P}_0 through \mathbf{P}_n and the second set as \mathbf{Q}_0 through \mathbf{Q}_{n+1} .

Since we don't want the shape to change at all, the beginning and ending control points do not change. Therefore, $\mathbf{Q}_0 = \mathbf{P}_0$ and $\mathbf{Q}_{n+1} = \mathbf{P}_n$. The remaining n control points \mathbf{Q}_1 through \mathbf{Q}_n are on the lines connecting the adjacent \mathbf{P}_i . These \mathbf{Q}_i may be computed as a linear combination of the "neighboring" $\mathbf{P}_{i-1}, \mathbf{P}_i$ as,

$$\mathbf{Q}_i = \frac{i}{n+1} \mathbf{P}_{i-1} + \left(1 - \frac{i}{n+1}\right) \mathbf{P}_i, \quad 1 \leq i \leq n$$

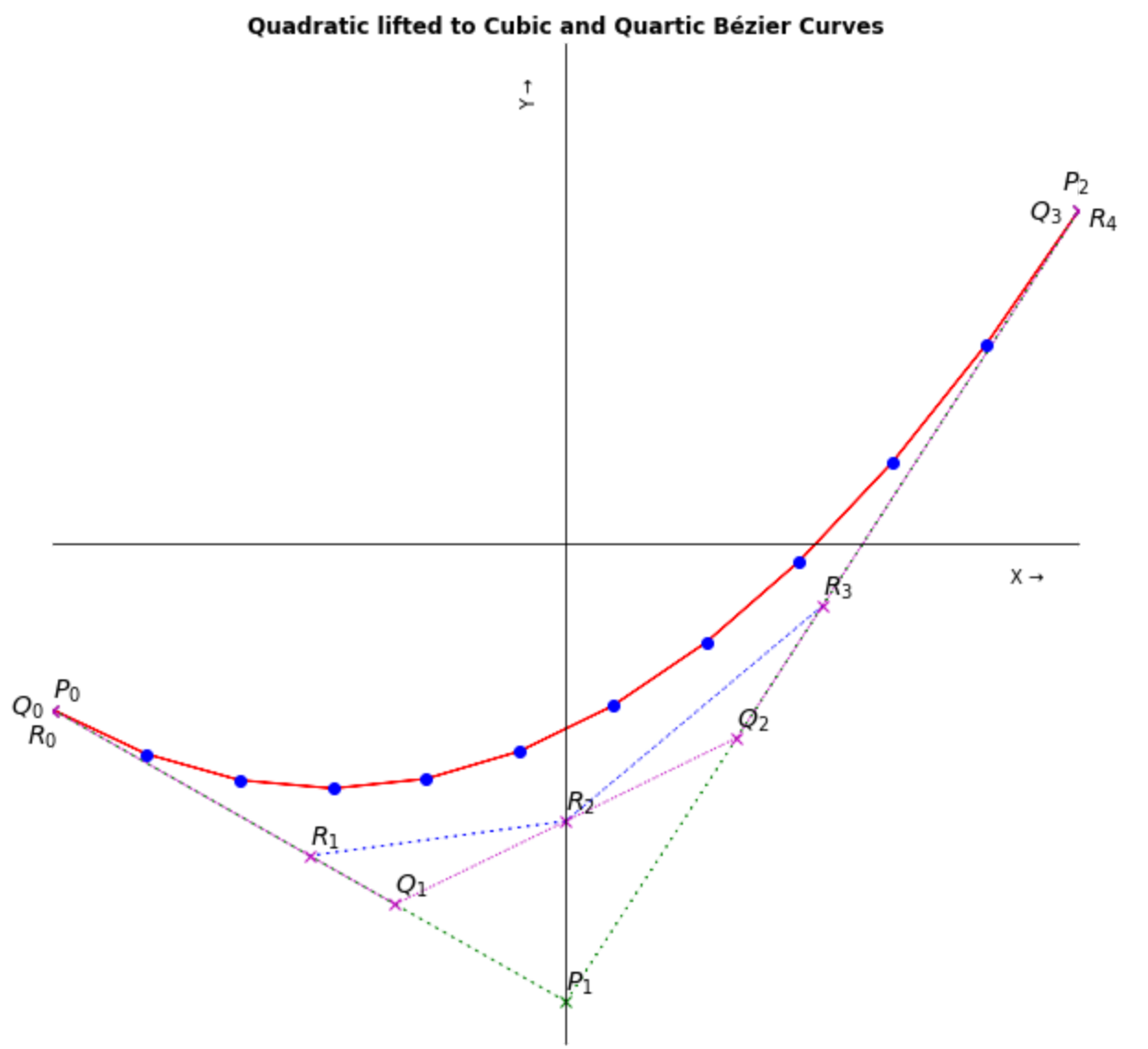
The following example shows a Quadratic Bézier curve lifted one degree to Cubic while retaining its shape.

Resolution
12
☒ Show Points
☒ Show Quartic

p_{0x}
 -6.0
 p_{2x}
 6.0

p_{0y}
 -2.0
 p_{2y}
 4.0

p_{1x}
 0.0
 p_{1y}
 -5.5



$$\begin{aligned}
\mathbf{B}^2(t) &= (1-t)^2 \cdot (-6.0, -2.0) + 2t(1-t) \cdot (0.0, -5.5) + t^2 \cdot (6.0, 4.0) \\
\mathbf{B}^3(t) &= (1-t)^3 \cdot (-6.0, -2.0) + 3(1-t)^2t \cdot (-2.0, -4.3) + 3(1-t)t^2 \cdot (2.0, -2.3) + t^3 \cdot (6.0, \\
\mathbf{B}^4(t) &= (1-t)^4 \cdot (-6.0, -2.0) + 4(1-t)^3t \cdot (-3.0, -3.8) + 6(1-t)^2t^2 \cdot (0.0, -3.3) + 4(1-t) \\
&\quad + t^4 \cdot (6.0, 4.0)
\end{aligned}$$

The four **Q** control points are "closer" to the generated curve than the three control points **P** because we are using a cubic representation for a parabolic curve. When we degree-lift again, from cubic **Q** to quartic **R**, the control points once again move closer to the generated curve. As the number of control points increases, the lines between them will approach the curve closer and closer.

Bézier Surfaces

[This section will make more sense if one reads the appendix *Generating Bézier Curve Representations Directly* first.]

As soon as one has a firm grip on Bézier curves, a natural next step is to extend the concept of a curve to that of making a *surface*. For a surface we have to do a double summation across *two* independent variables, u and v . The general definition has this form,

$$\mathbf{B}^{n,m}(u, v) = \sum_{i=0}^n \sum_{j=0}^m \binom{n}{i} u^i (1-u)^{n-i} \binom{m}{j} v^j (1-v)^{m-j} \mathbf{P}_{i,j}$$

where the $\mathbf{P}_{i,j}$ are the control points. As with t in the curve case, $u, v \in [0, 1]$.

Though Bézier surfaces can have any number of control points in the u and v directions, we are most concerned with one kind of Bézier surface, that of a *Bicubic*. A Bicubic Bézier surface uses sixteen (16) control points arranged in 4×4 grid and is defined as follows,

$$\mathbf{B}^{3,3}(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 \binom{3}{i} u^i (1-u)^{3-i} \binom{3}{j} v^j (1-v)^{3-j} \mathbf{P}_{i,j}$$

If we evaluate that equation, we obtain,

Out[14]:

$$\begin{array}{ll}
(1-u)^3(1-v)^3 \mathbf{P}_{0,0} & 3v(1-u)^3(1-v)^2 \mathbf{P}_{0,1} \\
3v^2(1-u)^3(1-v) \mathbf{P}_{0,2} & v^3(1-u)^3 \mathbf{P}_{0,3} \\
\\
3u(1-u)^2(1-v)^3 \mathbf{P}_{1,0} & 9uv(1-u)^2(1-v)^2 \mathbf{P}_{1,1} \\
9uv^2(1-u)^2(1-v) \mathbf{P}_{1,2} & 3uv^3(1-u)^2 \mathbf{P}_{1,3} \\
\\
3u^2(1-u)(1-v)^3 \mathbf{P}_{2,0} & 9u^2v(1-u)(1-v)^2 \mathbf{P}_{2,1} \\
9u^2v^2(1-u)(1-v) \mathbf{P}_{2,2} & 3u^2v^3(1-u) \mathbf{P}_{2,3} \\
\\
u^3(1-v)^3 \mathbf{P}_{3,0} & 3u^3v(1-v)^2 \mathbf{P}_{3,1} \\
3u^3v^2(1-v) \mathbf{P}_{3,2} & u^3v^3 \mathbf{P}_{3,3}
\end{array}$$

as the coefficients for the 16 control points. We add all 16 of the scaled \mathbf{P}_{ij} to get the point corresponding to each u, v pair. The Bézier surface itself may now be calculated by using a pair of nested loops iterating across the desired values for u and v , computing the coefficients corresponding to each u, v pair, and then forming the appropriate linear combination of $\mathbf{P}_{0,0}$ through $\mathbf{P}_{3,3}$ to get each point of the surface.

Though not required, we usually divide the u and v ranges into the same number of subdivisions, called the *resolution* of the Bézier surface. For example if we resolved the surface with 5 distinct points, we would have,

$$u, v \in [0 \quad 0.25 \quad 0.50 \quad 0.75 \quad 1.0]$$

Notice that the difference between successive u and v elements is $\frac{1}{n-1}$ where n is the resolution. The first point is always 0 and the last point is always 1 and therefore there are $n - 2$ internal points and $n - 1$ spaces in-between the two end points.

Implementation hint: If you're using Python, an easy way to get the proper spacing of u , and v (and t for that matter) points is to use numpy's `linspace` function thusly `numpy.linspace(0.0, 1.0, n)` where n is the total number of points.

In C, just remember that for resolution = r , there are r points in all to compute and the value of the i th point is $\frac{i}{r-1} \quad i \in [0, r - 1]$.

Bézier Surface Examples

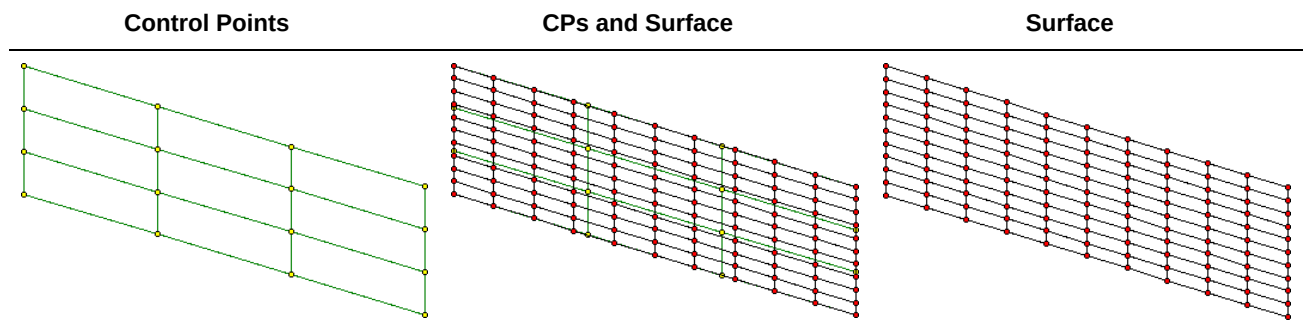
Equi-spaced, Coplanar \mathbf{P}_{ij} Example

Point	Coordinates	Point	Coordinates	Point	Coordinates	Point	Coordinates
P00	(1.0, 4.0, 0.0)	P10	(1.0, 3.0, 0.0)	P20	(1.0, 2.0, 0.0)	P30	(1.0, 1.0, 0.0)
P01	(2.0, 4.0, 0.0)	P11	(2.0, 3.0, 0.0)	P21	(2.0, 2.0, 0.0)	P31	(2.0, 1.0, 0.0)

<i>Point</i>	<i>Coordinates</i>	<i>Point</i>	<i>Coordinates</i>	<i>Point</i>	<i>Coordinates</i>	<i>Point</i>	<i>Coordinates</i>
P02	(3.0, 4.0, 0.0)	P12	(3.0, 3.0, 0.0)	P22	(3.0, 2.0, 0.0)	P32	(3.0, 1.0, 0.0)
P03	(4.0, 4.0, 0.0)	P13	(4.0, 3.0, 0.0)	P23	(4.0, 2.0, 0.0)	P33	(4.0, 1.0, 0.0)

The above control points are all in the XY plane (all of the z coordinates are zero). Further, they are equi-spaced in both the x and y dimensions, thereby forming a 4×4 square of squares. The result is a Bézier surface that is itself a plane. Since the corners of a Bézier surface always coincide with the four control points \mathbf{P}_{00} , \mathbf{P}_{03} , \mathbf{P}_{30} , and \mathbf{P}_{33} , the generated planar Bézier surface coincides with the planar region formed by the control points. (Note that the *generated points* do not necessarily align with the *control points*. Aside from the four corners, alignment between the generated points and the control points happens only when the spacing of the u , v values happens to coincide with the spacing of the control points.)

The following images show the control point grid, the combined control point grid and Bézier surface, and the Bézier surface alone for these control points. The resolution is 11 in both the u and v dimensions, so the surface comprises $10 \times 10 (= 100)$ quads. The view orientation is $\phi = 70^\circ$, $\theta = 20^\circ$, $\psi = 0^\circ$.



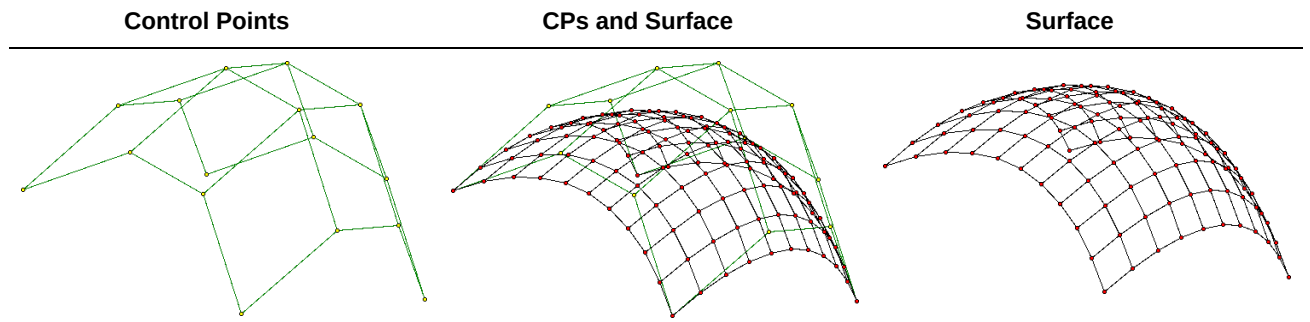
Simple Hull Example

<i>Point</i>	<i>Coordinates</i>	<i>Point</i>	<i>Coordinates</i>	<i>Point</i>	<i>Coordinates</i>	<i>Point</i>	<i>Coordinates</i>
P00	(1.0, 4.0, 0.0)	P10	(1.0, 3.0, 1.0)	P20	(1.0, 2.0, 1.0)	P30	(1.0, 1.0, 0.0)
P01	(2.0, 4.0, 1.0)	P11	(2.0, 3.0, 2.0)	P21	(2.0, 2.0, 2.0)	P31	(2.0, 1.0, 1.0)
P02	(3.0, 4.0, 1.0)	P12	(3.0, 3.0, 2.0)	P22	(3.0, 2.0, 2.0)	P32	(3.0, 1.0, 1.0)
P03	(4.0, 4.0, 0.0)	P13	(4.0, 3.0, 1.0)	P23	(4.0, 2.0, 1.0)	P33	(4.0, 1.0, 0.0)

While all of these control points are equi-spaced in both the x and y dimensions as in the previous example, they are no longer confined to the XY plane (their z coordinates are not restricted to the value zero). The result is a Bézier surface that somewhat resembles a simple hull (though upside down). As usual, the corners of this Bézier surface coincide with the four control points \mathbf{P}_{00} , \mathbf{P}_{03} , \mathbf{P}_{30} , and \mathbf{P}_{33} . All of the other generated points of the Bézier surface are pulled up towards the other control points. As can be seen in the central picture below, the entire Bézier surface is enclosed in the volume formed by the control points.

The following images show the control point grid, the combined control point grid and Bézier surface, and the Bézier surface alone for these control points. The resolution is 11 in both the u and v dimensions, so the surface comprises $10 \times 10 (= 100)$ quads. The view orientation is $\phi = 70^\circ$,

$$\theta = 20^\circ, \psi = 0^\circ.$$



Appendix — Generating Bézier Curve Representations Directly

We defined the Cubic Bézier curve in terms of the Quadratic curve and the Quadratic curve in terms of the Linear curve. One might rightly wonder if there's a way to represent these definitions directly instead of recursively. The answer is yes!

The definitions we have for the three kinds of Bézier curves we have studied so far are,

$$\begin{aligned} \text{Linear} &: (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \\ \text{Quadratic} &: (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2 \\ \text{Cubic} &: (1-t)^3\mathbf{P}_0 + 3(1-t)^2t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3 \end{aligned}$$

There are two patterns here: the numeric coefficients of each term and the powers of $(1-t)$ and t . The numeric coefficients may be seen more clearly if written in a table thus,

<i>Linear</i>	1	1		
<i>Quadratic</i>	1	2	1	
<i>Cubic</i>	1	3	3	1

This is instantly recognizable as a portion of *Pascal's Triangle* (notice how the sides are 1 and the interior values are the sum of the values directly above). The value of each entry in Pascal's triangle is $\binom{n}{m} \equiv \frac{n!}{m!(n-m)!}$. In our case, n is the line (1, 2, 3) and m is the entry in that line (starting at 0). Thus, the first line has entries $\binom{1}{0} = 1$ and $\binom{1}{1} = 1$. The second line has entries $\binom{2}{0} = 1$, $\binom{2}{1} = 2$, and $\binom{2}{2} = 1$. Finally, the third line is $\binom{3}{0} = 1$, $\binom{3}{1} = 3$, $\binom{3}{2} = 3$, and $\binom{3}{3} = 1$.

The pattern of the powers of $(1-t)$ and t are even easier to derive. Looking at the cubic case, we can express the left-to-right products of $(1-t)$ and t more uniformly as,

$$(1-t)^3t^0 \quad (1-t)^2t^1 \quad (1-t)^1t^2 \quad (1-t)^0t^3$$

Written that way, it's clear that the successive products of $(1-t)$ and t are $(1-t)^{n-i}t^i$ where n is the

order of the curve and i is the left-to-right index $\in [0, n]$.

Combining our derivation of the numeric coefficients and the powers of $(1 - t)$ and t , we can now write the definition of a Bézier curve of order n as,

$$\mathbf{B}^n(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$

Pretty neat, eh? We can now determine the value of any term in any order of Bézier curve directly.

Verification

Let's see how this definition works out for the three orders of Bézier curves that we determined by hand. We consider the linear case $\mathbf{B}^1(t)$ first.

$$\begin{aligned} \mathbf{B}^1(t) &= \sum_{i=0}^1 \binom{1}{i} t^i (1-t)^{1-i} \mathbf{P}_i \\ &= \binom{1}{0} t^0 (1-t)^{1-0} \mathbf{P}_0 + \binom{1}{1} t^1 (1-t)^{1-1} \mathbf{P}_1 \\ &= 1 \cdot 1 \cdot (1-t)^1 \cdot \mathbf{P}_0 + 1 \cdot t \cdot (1-t)^0 \cdot \mathbf{P}_1 \\ &= (1-t) \cdot \mathbf{P}_0 + t \cdot \mathbf{P}_1 \end{aligned}$$

Well, that agrees with what we defined above. What about the quadratic case, $\mathbf{B}^2(t)$?

$$\begin{aligned} \mathbf{B}^2(t) &= \sum_{i=0}^2 \binom{2}{i} t^i (1-t)^{2-i} \mathbf{P}_i \\ &= \binom{2}{0} t^0 (1-t)^{2-0} \mathbf{P}_0 + \binom{2}{1} t^1 (1-t)^{2-1} \mathbf{P}_1 + \binom{2}{2} t^2 (1-t)^{2-2} \mathbf{P}_2 \\ &= 1 \cdot 1 \cdot (1-t)^2 \cdot \mathbf{P}_0 + 2 \cdot t \cdot (1-t)^1 \cdot \mathbf{P}_1 + 1 \cdot t^2 \cdot (1-t)^0 \cdot \mathbf{P}_2 \\ &= (1-t)^2 \cdot \mathbf{P}_0 + 2 \cdot t \cdot (1-t) \cdot \mathbf{P}_1 + t^2 \cdot \mathbf{P}_2 \end{aligned}$$

So far so good. The quadratic case agrees as well. Now for the cubic case, $\mathbf{B}^3(t)$.

$$\begin{aligned} \mathbf{B}^3(t) &= \sum_{i=0}^3 \binom{3}{i} t^i (1-t)^{3-i} \mathbf{P}_i \\ &= \binom{3}{0} t^0 (1-t)^{3-0} \mathbf{P}_0 + \binom{3}{1} t^1 (1-t)^{3-1} \mathbf{P}_1 + \binom{3}{2} t^2 (1-t)^{3-2} \mathbf{P}_2 + \binom{3}{3} t^3 (1-t)^{3-3} \mathbf{P}_3 \\ &= 1 \cdot 1 \cdot (1-t)^3 \cdot \mathbf{P}_0 + 3 \cdot t \cdot (1-t)^2 \cdot \mathbf{P}_1 + 3 \cdot t^2 \cdot (1-t)^1 \cdot \mathbf{P}_2 + 1 \cdot t^3 \cdot (1-t)^0 \cdot \mathbf{P}_3 \\ &= (1-t)^3 \cdot \mathbf{P}_0 + 3 \cdot t \cdot (1-t)^2 \cdot \mathbf{P}_1 + 3 \cdot t^2 \cdot (1-t) \cdot \mathbf{P}_2 + t^3 \cdot \mathbf{P}_3 \end{aligned}$$

Our three orders have each corresponded to what we derived by hand above. *Et voila, ça marche!*

Extending the Direct Representation to Surfaces

[Do I need to put the detailed derivation in here?]

$$\mathbf{B}^{n,m}(u, v) = \sum_{i=0}^n \sum_{j=0}^m \binom{n}{i} u^i (1-u)^{n-i} \binom{m}{j} v^j (1-v)^{m-j} \mathbf{P}_{i,j}$$

Appendix — Matrix Representation

The Bézier curve representations we gave above were in analytic expression form, but they can also be expressed in a rather tidy matrix form. Let's take a look at the analytic expression representation of the cubic form,

$$\begin{aligned} \mathbf{B}^3(t) &= \sum_{i=0}^3 \binom{3}{i} t^i (1-t)^{3-i} \mathbf{P}_i \\ &= (1-t)^3 \mathbf{P}_0 + 3t(1-t)^2 \mathbf{P}_1 + 3t^2(1-t) \mathbf{P}_2 + t^3 \mathbf{P}_3 \end{aligned}$$

Using the definition of matrix multiplication, we can immediately rewrite this as the multiplication of a row vector and a column vector as follows,

$$\begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

Looking good so far, but that row vector is kind of complicated. It would be much nicer (and easier to extend to higher orders) if we could use $\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix}$ on the left and instead write something like,

$$\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

In this representation, we've inserted a matrix \mathbf{M} in-between our original row vector and column vector. If we make this matrix 4×4 , the matrix multiplication row / column size constraints will be satisfied. We want the result of the multiplication of $\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix}$ by \mathbf{M} to yield

$\begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix}$. All we have to do is find the values of the elements of \mathbf{M} that give us this result. This might seem as if it would be tricky to figure out, but it turns out to be pretty simple.

Let's start by expanding the elements in our desired t row vector,

Out[15]:

$$\begin{aligned}(1-t)^3 &\rightarrow -t^3 + 3t^2 - 3t + 1 \\ t(1-t)^2 &\rightarrow t^3 - 2t^2 + t \\ t^2(1-t) &\rightarrow -t^3 + t^2 \\ t^3 &\rightarrow t^3\end{aligned}$$

Having done that, we see that we need certain coefficients of $t^0 (= 1)$, t , t^2 , and t^3 for each of the elements. Some of those coefficients are 0. For example, in the last element, all of the coefficients are 0 except that for t^3 , which is 1. On the other hand, for the first element all four of the coefficients are non-zero, being -1 for t^3 , 3 for t^2 , -3 for t^1 , and finally 1 for t^0 .

We can determine what the values of all these coefficients are and how to place them in the matrix \mathbf{M} if we now show all of \mathbf{M} 's elements and perform the original multiplication. Doing so we find,

Out[16]:

$$\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$= \begin{bmatrix} m_{00} + m_{10}t + m_{20}t^2 + m_{30}t^3 \\ m_{01} + m_{11}t + m_{21}t^2 + m_{31}t^3 \\ m_{02} + m_{12}t + m_{22}t^2 + m_{32}t^3 \\ m_{03} + m_{13}t + m_{23}t^2 + m_{33}t^3 \end{bmatrix}^T$$

If we now equate corresponding elements, we have,

Out[17]:

$$\begin{aligned}m_{00} + m_{10}t + m_{20}t^2 + m_{30}t^3 &= -t^3 + 3t^2 - 3t + 1 \\ m_{01} + m_{11}t + m_{21}t^2 + m_{31}t^3 &= t^3 - 2t^2 + t \\ m_{02} + m_{12}t + m_{22}t^2 + m_{32}t^3 &= -t^3 + t^2 \\ m_{03} + m_{13}t + m_{23}t^2 + m_{33}t^3 &= t^3\end{aligned}$$

Out[18]: From which we trivially determine,

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Take careful note of the subscripts for the individual m_{ij} when extracting the coefficients and placing them in the matrix. The i digit is the *row* and the j digit is the *column* of the matrix. Rows and columns are numbered starting at 0.

So we can now write,

$$\mathbf{B}^3(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

which certainly is quite a tidy form.

Determining the m_{ij} Directly

While the final form for $\mathbf{B}^3(t)$ is tidy, we did have to go through shenanigans to get the values of m_{ij} . It's natural to wonder if there is a direct way to compute those values just as we found for the $b_{i,n}$ values. The answer to this question is yes, but deriving the m_{ij} requires a diversion into the definitions of a vector space and a basis thereof.

[*Diversion into the definitions of a vector space and a basis thereof goes here.*]

From those definitions, we can now show that any n th order Bernstein polynomial can be written in terms of a basis formed from $1, t, t^2, t^3, \dots, t^n$. The m_{ij} values are the coefficients of the basis elements for the corresponding Bernstein polynomial.

Specifically, for a Bézier curve of order n , the matrix \mathbf{M} will be order $(n + 1) \times (n + 1)$ and each entry will be,

$$m_{ij} = \begin{cases} 0 & \text{for } j > i \\ \binom{n}{n-j} & \text{for } j = i \\ (-1)^{i-j} \binom{n}{n-j} \binom{n-j}{n-i} & \text{for } j < i \end{cases}$$

Using this definition of m_{ij} , we can immediately write the matrix forms of the linear, quadratic, quartic, and quintic Bézier curves as follows,

Out[19]:

Linear

$$\mathbf{B}^1(t) = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix}$$

Out[20]:

Quadratic

$$\mathbf{B}^2(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix}$$

Out[21]:

Quartic

$$\mathbf{B}^4(t) = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 12 & -12 & 4 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix}$$

Out[22]:

Quintic

$$\mathbf{B}^5(t) = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 & t^5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \\ \mathbf{P}_5 \end{bmatrix}$$

Pretty neat, huh?

Extending the Matrix Representation to Surfaces

The matrix representation concept may be extended to Bézier surfaces.

Why Bother?

One may wonder why we bother with expressing the Bézier curves in this matrix form given the analytical expression is perfectly serviceable. This simplest reason is that *perfectly serviceable* is not the same as *perfect*.

1. [*Ease of extension to higher orders.*]
2. [*Computational efficiency in certain cases.*]

Appendix — Historical Notes



Pierre Bézier

What are now known as Bézier curves and surfaces were popularized in the early 1960s by Pierre Bézier while working as an engineer at Renault. The curves were a means to improve the design of automobile bodies. These kinds of curves and surfaces were in use earlier at Citroën where they had been developed by Paul de Casteljau. De Casteljau was prohibited from publishing his work by Citroën company policy.

Both Bézier's and de Casteljau's work were based on the mathematical background developed by Бернштейн (Bernstein) and they created their work independently of each other.

Paul de Casteljau

While Bézier's name is generally attached to the curves, de Casteljau is remembered for having developed a robust and numerically stable method for calculating the values of the points on the curves and surfaces (in 1959). He received the 2012 Bézier Award from the Solid Modeling Association (SMA). Bézier himself died in 1999 but in 1993 he did go on record as lamenting the fact that de Casteljau was "deprived of part of the well deserved fame that his discoveries and inventions should have earned him".





Сергей Натанович Бернштейн

The coefficient $\binom{n}{i} t^i (1 - t)^{n-i}$ of the control point \mathbf{P}_i is known as the *Bernstein Polynomial*, after Сергей Натанович Бернштейн (romanized as *Sergei Natanovich Bernstein*), who devised it as part of a proof of the Stone-Weierstrass Approximation Theorem in about 1912.