Bernstein Polynomials and Bézier Curves

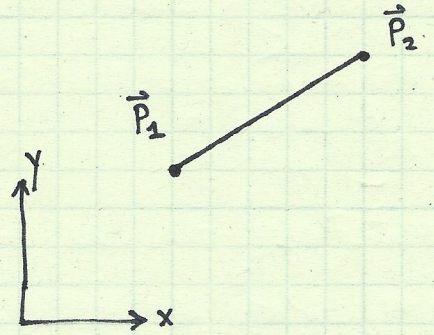
Consider the following question:

Q: If one has a set of $p+1$ control points $\vec{P}_i \in \mathbb{R}^d$, how might one define a parametric polynomial curve of degree p from these points?

To simplify our exposition, we set $d = 2$ in what follows.

The easy or straightforward case is $p = 1$, where we simply draw a straight line between the two control points.

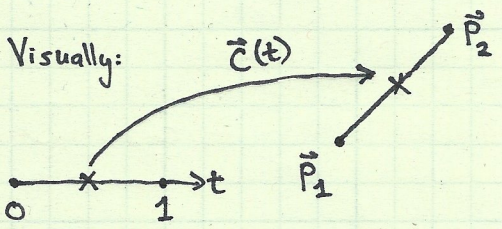
The Straightforward Case: $p = 1$



Parametrically, we have defined a linear curve $\vec{C}: [0,1] \to \mathbb{R}^2$ s.t.:

$$\vec{C}(t) = (1-t)\vec{P}_1 + t\vec{P}_2 \quad \forall t \in [0,1]$$

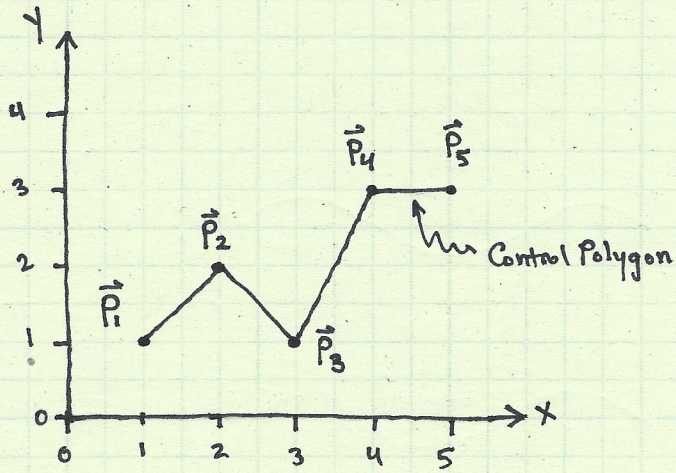
Visually:



The curve $\vec{C}(t)$ is the best fit with respect to the two control points, and it is the curve with the least variation (wiggles) between the points as well.

Now, suppose that $p > 1$. Then, it is not quite as obvious how to define the curve $\vec{C}(t)$. To fix some numbers, let us consider an example with $p = 4$.

A Not-So-Straightforward Example: $p = 4$



$\vec{P}_1 = (1,1)$
$\vec{P}_2 = (2,2)$
$\vec{P}_3 = (3,1)$
$\vec{P}_4 = (4,3)$
$\vec{P}_5 = (5,3)$

Above, we have also constructed a piecewise linear interpolation of the control points which is referred to as the control polygon. Visually, it represents the shape of the control points.

Perhaps the most-straightforward approach to defining a degree p polynomial curve using the $p+1$ control points is polynomial interpolation. This is precisely the basis of the classical finite element method.

Given a set of $p+1$ data points:

$$\vec{P}_i = (x_i, y_i) \qquad i = 1, \ldots, p+1$$

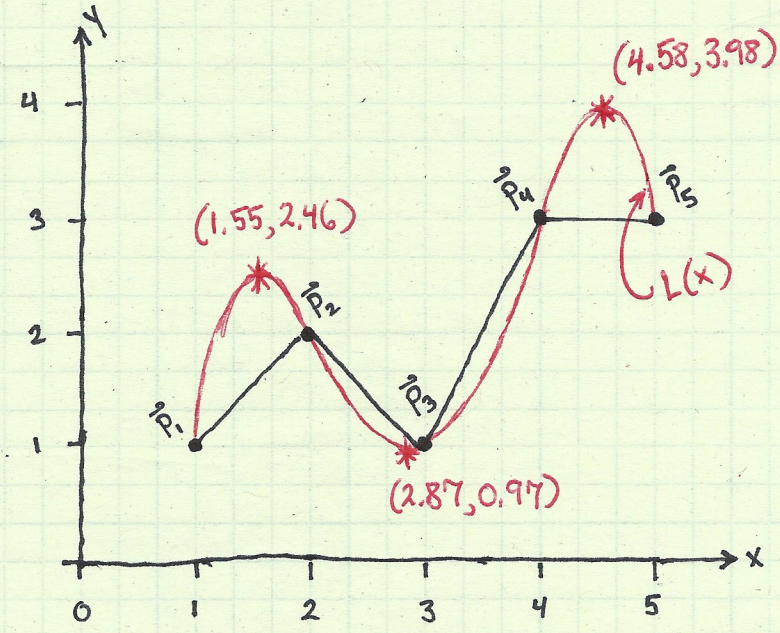
where no two x_i are the same, the interpolation polynomial in Lagrange form is:

$$L(x) = \sum_{i=1}^{p+1} y_i \, l_i(x)$$

where l_i is the i^{th} Lagrange basis polynomial of degree p:

$$l_i(x) = \prod_{\substack{1 \le m \le p+1 \\ m \neq i}} \frac{x - x_m}{x_i - x_m} = \frac{(x-x_1)}{(x_i-x_1)} \cdots \frac{(x-x_{i-1})}{(x_i-x_{i-1})} \frac{(x-x_{i+1})}{(x_i-x_{i+1})} \cdots \frac{(x-x_{p+1})}{(x_i-x_{p+1})}$$

For our earlier example, the interpolation polynomial looks as follows:



Looking above, we immediately see the advantage of the interpolation polynomial – it exactly fits the control points. However, from the perspective of fitting the shape of the control points and the control polygon, the interpolation polynomial suffers from many distinct disadvantages:

- Both the max and min values of $L(x)$ over the interval $[1,5]$ are larger and smaller, respectively, than the max and min of the data points. To be precise,

$$\max_{1 \le x \le 5} L(x) > \max_j y_j$$

$$\min_{1 \le x \le 5} L(x) < \min_j y_j$$

- The curve $L(x)$ exhibits more oscillations than the control polygon and consequently has more local extrema than the control polygon (five versus four).

- In fact, polynomial interpolation is susceptible to Runge's phenomenon, a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation of high degree over a set of equispaced points.

While polynomial interpolation may be ideally suited for data fitting, it is obviously not ideal for design. Designers are interested in not only manipulating point locations on a curve but more generally in manipulating the shape of the curve. In this sense, the control points must furnish natural shape handles that permit intuitive creation or modification of the curve to satisfy prescribed aesthetic or functional requirements.

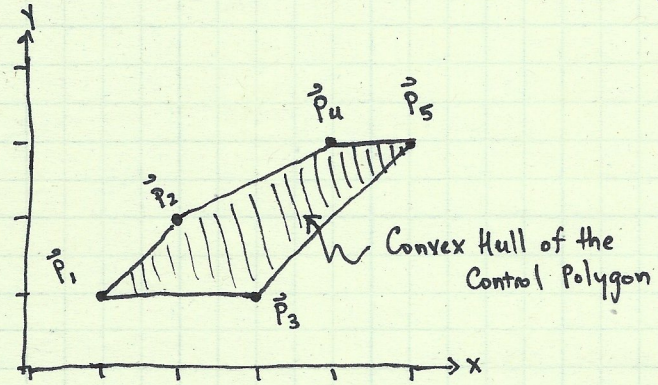
In what follows, we prescribe four main requirements that a curve should satisfy:

- The curve should begin at $\vec{P}_1$ and end at $\vec{P}_{p+1}$. This is the so-called endpoint interpolation property.

- The start (end) of the curve should be tangent to the first (last) section of the Bézier polygon. This is the so-called tangency property.

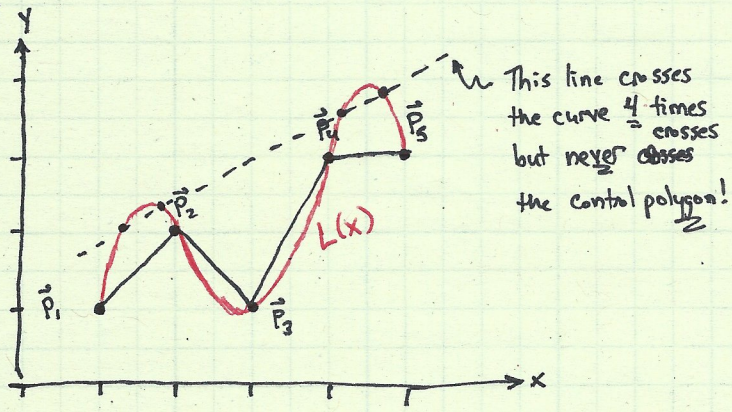
- The curve should lie within the convex hull of the control polygon, that is, the smallest convex set that contains the control points. This is the so-called convex hull property.

Convex Hull for Our Earlier Example:



- The curve should not oscillate more than the control polygon. To be more precise, if a line is drawn through the curve, the number of intersections with the curve is no more than the number of intersections with the control polygon. This is the so-called variation diminishing property.

The interpolation polynomial from our earlier example violates the variation diminishing property.



Bézier curves satisfy the aforementioned requirements and more. First introduced by Issac Jacob Schoenberg in 1930, they were popularized by French engineer Pierre Bézier in the 1960s when he used them to design automobile bodies at Renault.

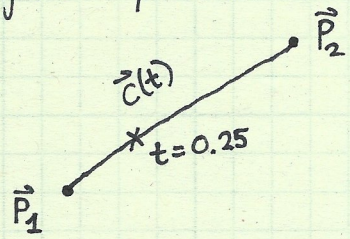
Bézier curves have a recursive, geometric construction, so it will be easiest to discuss their construction for $p=1$ before proceeding to higher polynomial degrees.

Bézier Curve, $p = 1$: In this setting, we have two control points:

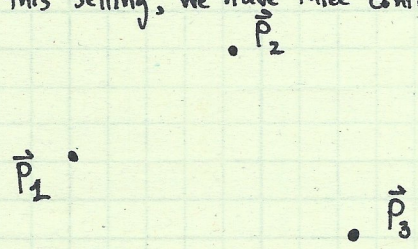
To define the curve, we first choose a parameter $t \in [0,1]$ and then interpolate between the control points:

$$\vec{C}(t) = (1-t)\,\vec{P}_1 + t\,\vec{P}_2$$

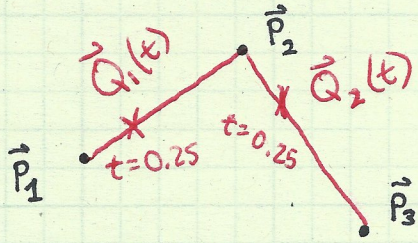
Sweeping over t yields the curve:



Bézier Curve, $p = 2$: In this setting, we have three control points.



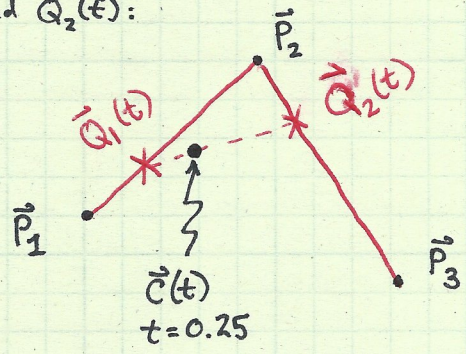
To define the curve, we again choose a parameter $t \in [0,1]$ and interpolate between adjacent control points:



$$\vec{Q}_1(t) = (1-t)\,\vec{P}_1 + t\,\vec{P}_2$$

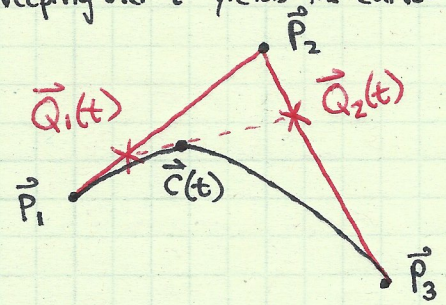
$$\vec{Q}_2(t) = (1-t)\,\vec{P}_2 + t\,\vec{P}_3$$

We finally define our curve by interpolating between $\vec{Q}_1(t)$ and $\vec{Q}_2(t)$:

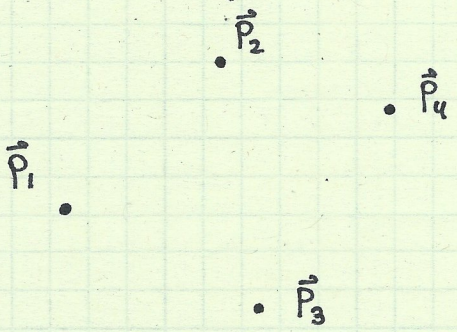


$$\vec{C}(t) = (1-t)\,\vec{Q}_1(t) + t\,\vec{Q}_2(t)$$

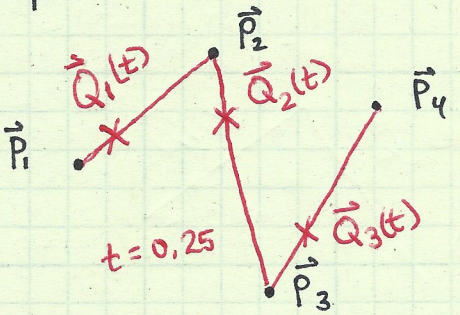
Sweeping over t yields the curve:



Bézier Curve, $p=3$: In this setting, we have four control points:

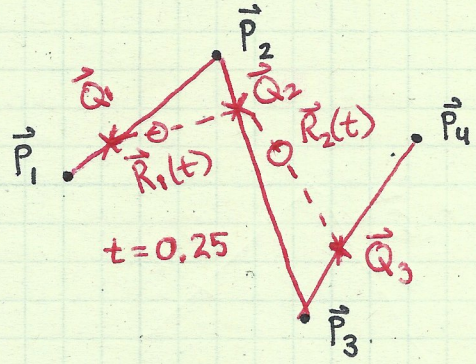


Again, choose $t \in [0,1]$ and interpolate between adjacent control points:



$$\vec{Q}_1(t) = (1-t)\,\vec{P}_1 + t\,\vec{P}_2$$
$$\vec{Q}_2(t) = (1-t)\,\vec{P}_2 + t\,\vec{P}_3$$
$$\vec{Q}_3(t) = (1-t)\,\vec{P}_3 + t\,\vec{P}_4$$

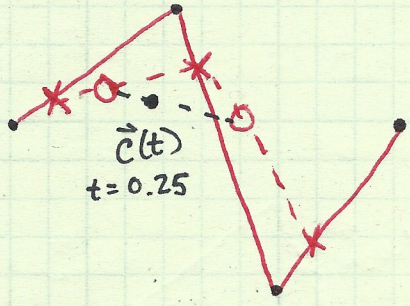
Now, we interpolate between adjacent $\vec{Q}_i(t)$:



$$\vec{R}_1(t) = (1-t)\vec{Q}_1(t) + t\,\vec{Q}_2(t)$$

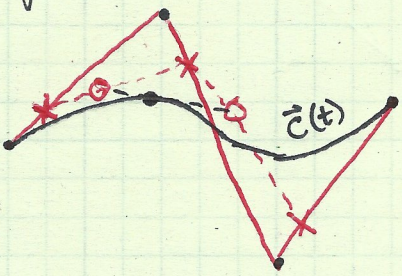
$$\vec{R}_2(t) = (1-t)\vec{Q}_2(t) + t\,\vec{Q}_3(t)$$

We finally define our curve by interpolating between $\vec{R}_1(t)$ and $\vec{R}_2(t)$:



$$\vec{C}(t) = (1-t)\vec{R}_1(t) + t\,\vec{R}_2(t)$$

Sweeping over t yields the curve:



Visually, we see that the geometric construction of a Bézier curve yields a curve that "follows" the shape of the control polygon. Animations of the geometric construction of a Bézier curve for degrees $p = 1, 2, 3, 4$ may be found at:

www.jasondavies.com/animated-bezier

Note that a Bézier curve is simply formed via convex combinations of the control points. Thus, by construction, a Bézier curve satisfies the convex hull property. A Bézier curve also satisfies the endpoint interpolation and tangency properties by construction.

The recursive definition of a Bézier curve was first introduced by Paul de Casteljau, a French mathematician, in 1959 and for this reason it is referred to as De Casteljau's algorithm. Mathematically, we have:

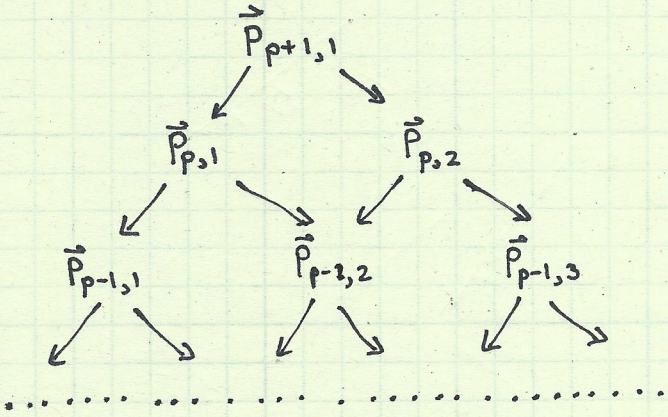
$$\vec{P}_{i,j}(t) = \begin{cases} \vec{P}_j & \text{if } i = 1 \text{ and } j = 1, \ldots, p+1 \\ (1-t)\,\vec{P}_{i-1,j} + t\,\vec{P}_{i-1,j+1} & \text{if } i = 2, \ldots, p+1 \text{ and } j = 1, \ldots, p+2-i \end{cases}$$

With this recursion relation, the control points correspond to level $i=1$ (i.e., $\vec{P}_{1,j} = \vec{P}_j$) and the point on the curve is the single point on level $i=p+1$. That is,

$$\vec{C}(t) = \vec{P}_{p+1,1}(t)$$

We can associate other levels with points $\vec{R}_j$ and $\vec{Q}_j$ from before. Notably, we have $\vec{P}_{2,j} = \vec{Q}_j$ and $\vec{P}_{3,j} = \vec{R}_j$.

We can write down De Casteljau's algorithm as a triangle scheme where we use directed arrows to denote dependencies:



which inspires the following pseudo code:

```
function de Casteljau (i, j, t)
begin
    if i = 1 then
        return P_j
    else
        return (1-t) * de Casteljau (i-1, j, t) + t * de Casteljau (i-1, j+1, t)
end
```

Obviously, the above algorithm is not efficient for large p as it has $O(p^2)$ complexity, but it is exceptionally stable as it expresses points on a curve solely by convex combinations and thus is not susceptible to catastrophic cancellation error.

To obtain, a better understanding of the underlying parametrization, let us return back to the geometric construction of a Bézier curve. Note that:

For $p=1$: $\vec{C}(t) = (1-t)\vec{P}_1 + t\vec{P}_2$ — Control Points / Coefficients; Polynomial Basis Functions

For $p=2$:
$$\vec{C}(t) = (1-t)\vec{Q}_1(t) + t\vec{Q}_2(t)$$
$$= (1-t)^2\vec{P}_1 + 2t(1-t)\vec{P}_2 + t^2\vec{P}_3$$

Polynomial Basis Functions

For $p = 3$:

$$\begin{aligned}\vec{C}(t) &= (1-t)\vec{R}_1(t) + t\vec{R}_2(t) \\ &= (1-t)^2\vec{Q}_1(t) + 2t(1-t)\vec{Q}_2(t) + t^2\vec{Q}_3(t) \\ &= (1-t)^3\vec{P}_1 + 3t(1-t)^2\vec{P}_2 + 3t^2(1-t)\vec{P}_3 + t^3\vec{P}_4\end{aligned}$$

Polynomial Basis Functions

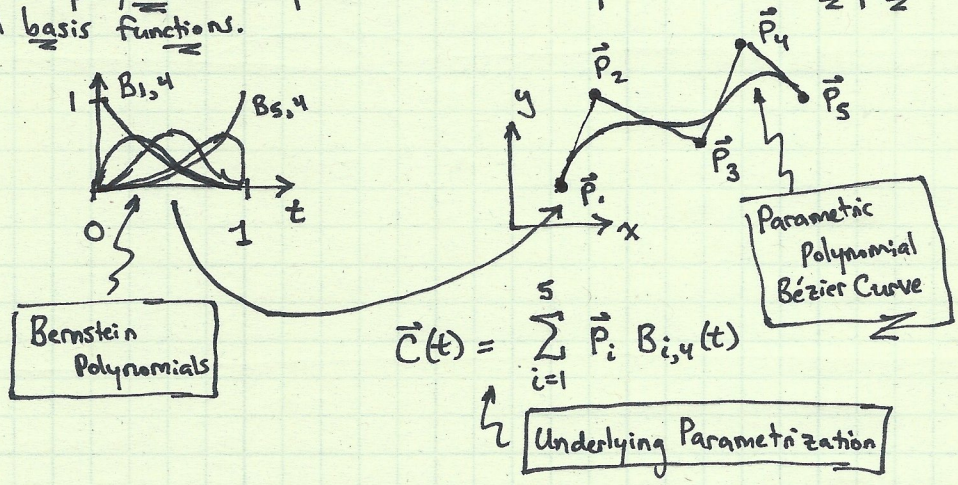
For general p, we have that:

$$\vec{C}(t) = \sum_{i=1}^{p+1} B_{i,p}(t)\,\vec{P}_i \qquad t \in [0,1] \qquad (*)$$

where $B_{i,p}(t)$ is the i^{th} Bernstein polynomial of degree p:

$$B_{i,p}(t) = \binom{p}{i-1} t^{i-1} (1-t)^{p+1-i}$$

Formula (*) exactly describes how the unit interval maps onto the Bézier curve via a parametric polynomial composed as a sum of products of control points and polynomial basis functions.



Consequently, we have a direct relationship between geometric objects (i.e., Bézier curves), parametric mappings, and basis functions (i.e., Bernstein polynomials). We inferred the underlying basis functions in the parametric mapping from the construction of the Bézier curve, but we could have just as easily defined the Bernstein polynomials first and then inferred the Bézier curve. The first approach is akin to design while the second is akin to analysis.

Bernstein polynomials are an interesting topic in their own right, especially in the mathematics community. In fact, Bernstein polynomials were first used by Sergei Bernstein in proving any continuous function can be uniformly approximated as closely as desired by a polynomial function.

Bernstein polynomials have many desirable properties. For example, they exhibit a recursive relationship just like Bézier curves:

$$B_{i,p}(t) = (1-t)\,B_{i,p-1}(t) + t\,B_{i-1,p-1}(t)$$

with: $B_{1,0}(t) \equiv 1$

$B_{i,p}(t) \equiv 0$ if $i<1$ or $i>p+1$

This relationship provides a stable means of evaluating Bernstein polynomials. In addition, Bernstein polynomials satisfy the following:

Positivity: $B_{i,p}(t) > 0$ for $t \in (0,1)$

Symmetry: $B_{i,p}(1-t) = B_{p+1-i,p}(t)$

Endpoint Interpolation: $B_{i,p}(0) = \delta_{i,1}$, $B_{i,p}(1) = \delta_{i,p+1}$

Derivative as Recursion: $\frac{d}{dt} B_{i,p}(t) = p\left(B_{i-1,p-1}(t) - B_{i,p-1}(t)\right)$

Constant Integral: $\int_0^1 B_{i,p}(t)\,dt = \frac{1}{p+1}$

The aforementioned properties may actually be used to show many of the properties of a Bézier curve. In addition, Bernstein polynomials comprise a partition of unity:

Partition of Unity: $\sum_{i=1}^{p+1} B_{i,p}(t) = 1$

One can actually immediately show that Bézier curves satisfy the convex hull property by immediately appealing to two properties of Bernstein polynomials: positivity and partition of unity. To see this, note that:

$$\vec{C}(t) = \sum_{i=1}^{p+1} \vec{P}_i\, B_{i,p}(t)$$

is simply a linear combination of all its control points with positive coefficients who sum to one; that is, $\vec{C}(t)$ is a convex combination of control points. Later, we will demand that spline basis functions also satisfy positivity and the partition of unity property to ensure spline curves, surfaces, and volumes also satisfy the convex hull property.

To conclude, let us briefly discuss the variation diminishing property once again. While it is visually obvious, it is actually quite delicate to prove. It turns out the proof is inherently related to the fact that one can elevate the polynomial degree of a Bézier curve without changing its shape or even parametrization. This is a direct result of the fact that Bernstein polynomials can be written as a linear combination of Bernstein polynomials of higher degree:

$$B_{i,p-1}(t) = \frac{p+1-i}{p} B_{i,p}(t) + \frac{i}{p} B_{i+1,p}(t)$$

Therefore, if we want to degree elevate a Bézier curve of degree p as follows:

$$\vec{C}(t) = \sum_{i=1}^{p+1} B_{i,p}(t)\,\vec{P}_i = \sum_{i=1}^{p+2} B_{i,p+1}(t)\,\vec{Q}_i$$

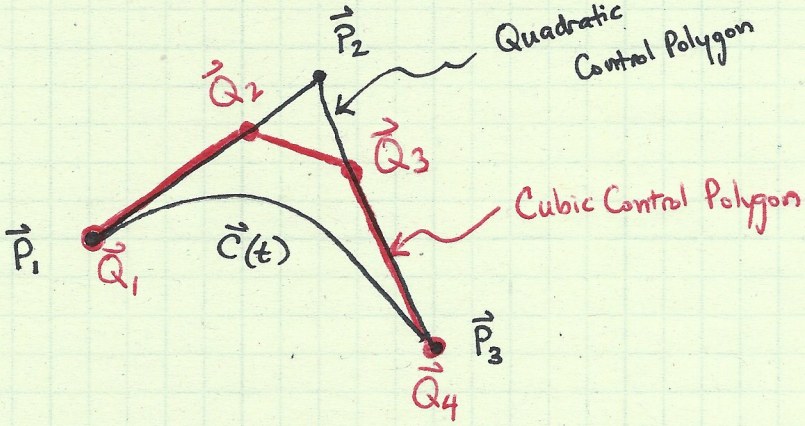
then we simply set:

Degree Elevation

$$\begin{cases} \vec{Q}_{i+1} = \frac{i}{p+1}\vec{P}_i + \left(1 - \frac{i}{p+1}\right)\vec{P}_{i+1} & i = 1, \dots, p \\ \vec{Q}_1 = \vec{P}_1\,, \quad \vec{Q}_{p+2} = \vec{P}_{p+2} \end{cases}$$

Note that each control point $\vec{Q}_i$ is a convex combination of control points $\vec{P}_i$. In fact, the elevated control polygon is simply a piecewise linear interpolation of the original control polygon.

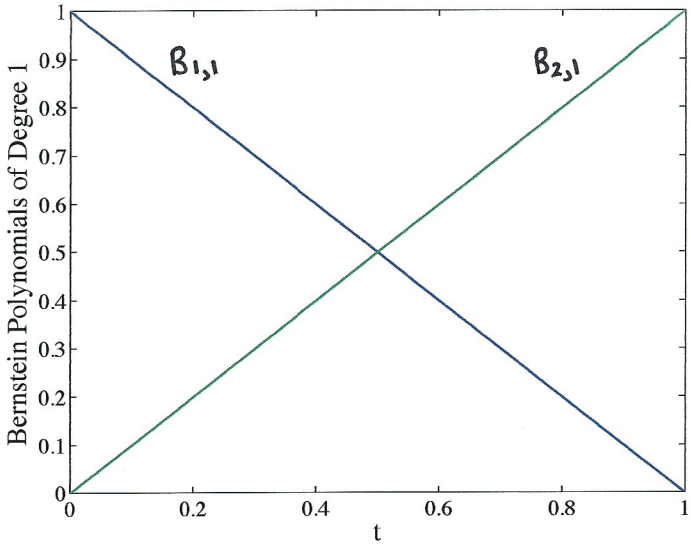
Below, we illustrate the control points $\vec{P}_i$ for a quadratic Bézier curve and the control points $\vec{Q}_i$ for the degree-elevated cubic curve



Notice that the cubic control polygon lies closer to the curve $\vec{C}(t)$ than the quadratic control polygon. In fact, under the limit of repeated degree elevation, the control polygons converge uniformly to the Bézier curve. This is a classical result due to the Weierstrass Approximation Theorem.

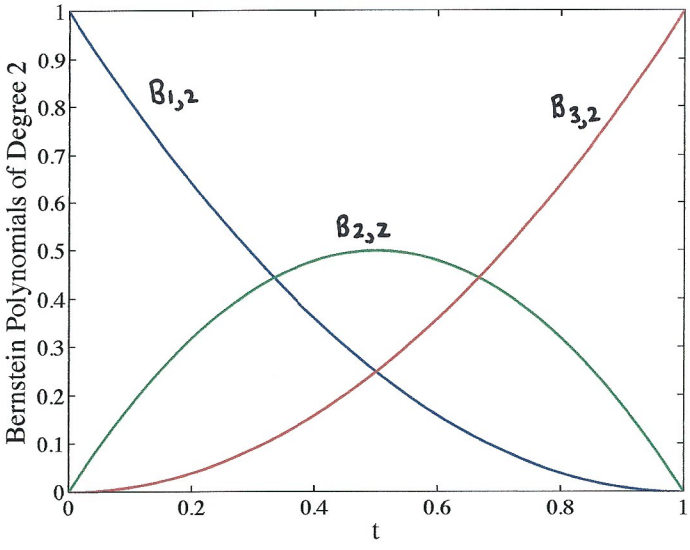
Notice also that the cubic control polygon also "wiggles" less than the quadratic control polygon by construction. In fact, it is an immediate result of piecewise linear interpolation that a line intersects the cubic control polygon no more times than it intersects the quadratic control polygon. This property also obviously holds under repeated degree elevation. As the control polygons obtained via degree elevation converge uniformly to the Bézier curve, the variation diminishing property for the quadratic Bézier curve is hence established.

Appendix 2: Plots of Bernstein Polynomials



$$B_{1,1}(t) = 1-t$$

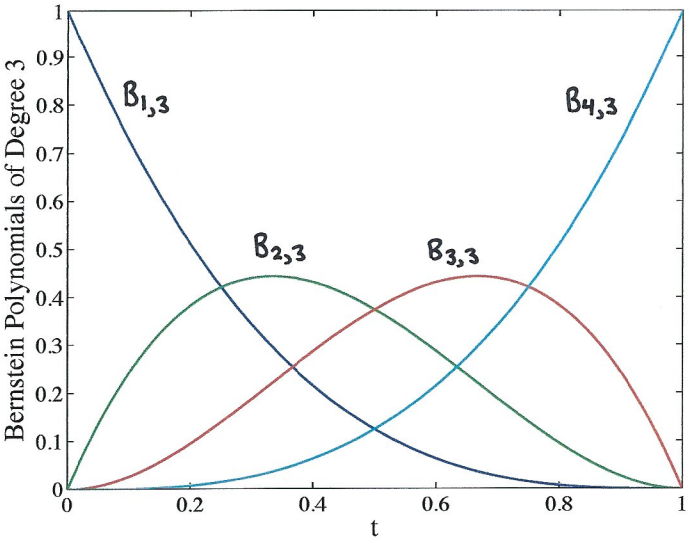
$$B_{2,1}(t) = t$$



$$B_{1,2}(t) = (1-t)^2$$

$$B_{2,2}(t) = 2t(1-t)$$

$$B_{3,2}(t) = t^2$$

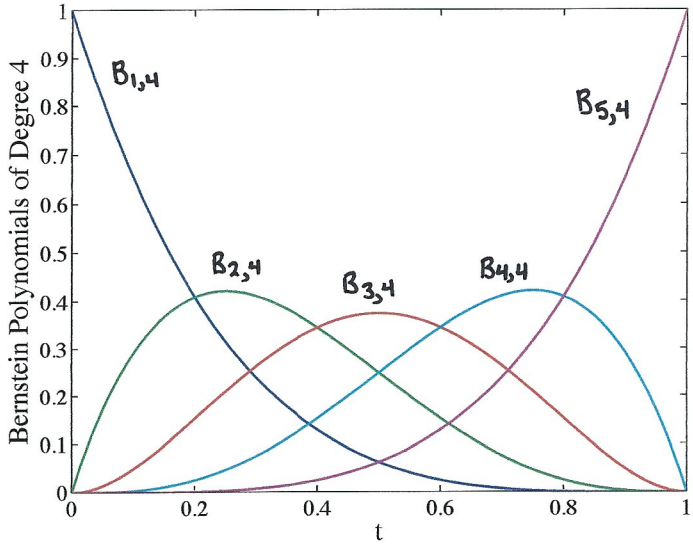


$$B_{1,3}(t) = (1-t)^3$$

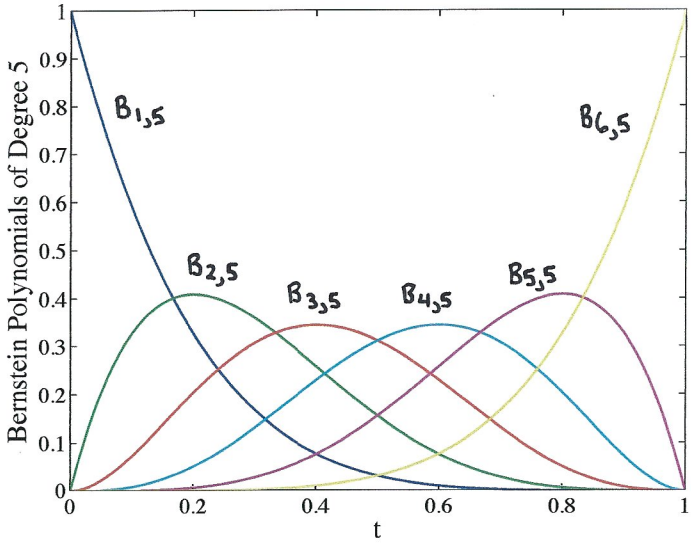
$$B_{2,3}(t) = 3t(1-t)^2$$

$$B_{3,3}(t) = 3t^2(1-t)$$

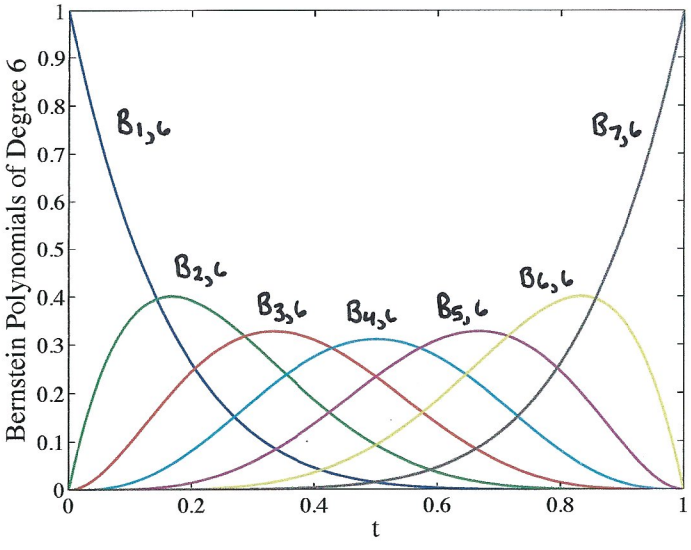
$$B_{4,3}(t) = t^3$$



$$B_{1,4}(t) = (1-t)^4$$
$$B_{2,4}(t) = 4t(1-t)^3$$
$$B_{3,4}(t) = 6t^2(1-t)^2$$
$$B_{4,4}(t) = 4t^3(1-t)$$
$$B_{5,4}(t) = t^4$$



$$B_{1,5}(t) = (1-t)^5$$
$$B_{2,5}(t) = 5t(1-t)^4$$
$$B_{3,5}(t) = 10t^2(1-t)^3$$
$$B_{4,5}(t) = 10t^3(1-t)^2$$
$$B_{5,5}(t) = 5t^4(1-t)$$
$$B_{6,5}(t) = t^5$$



$$B_{1,6}(t) = (1-t)^6$$
$$B_{2,6}(t) = 6t(1-t)^5$$
$$B_{3,6}(t) = 15t^2(1-t)^4$$
$$B_{4,6}(t) = 20t^3(1-t)^3$$
$$B_{5,6}(t) = 15t^4(1-t)^2$$
$$B_{6,6}(t) = 6t^5(1-t)$$
$$B_{7,6}(t) = t^6$$