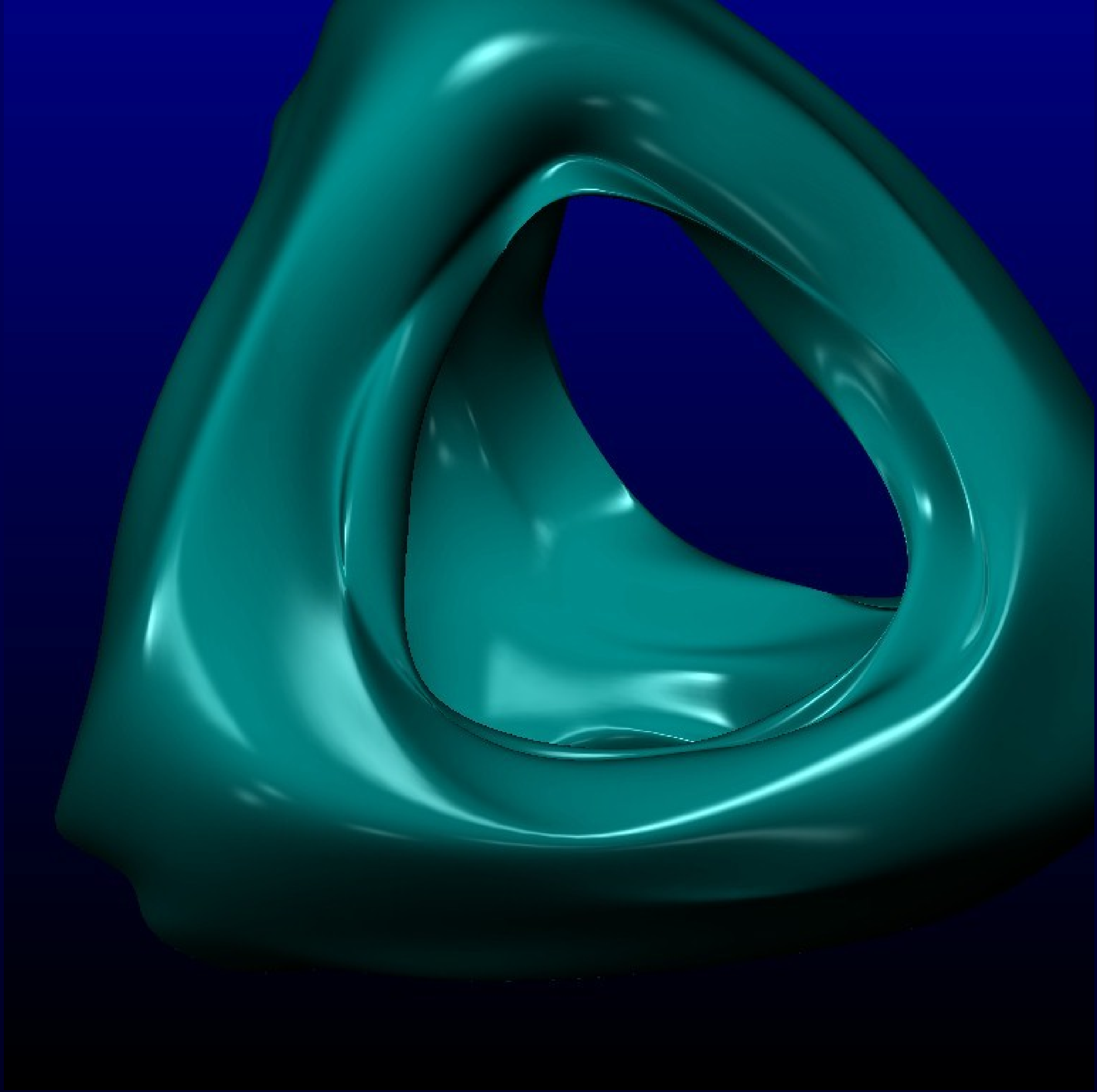
Interactive Graphics

Lecture 8: Introduction to Spline Curves



Splines

The word spline comes from the ship building trade where planks were originally shaped by bending them round pegs fixed in the ground.

Originally it was the pegs that were referred to as splines.

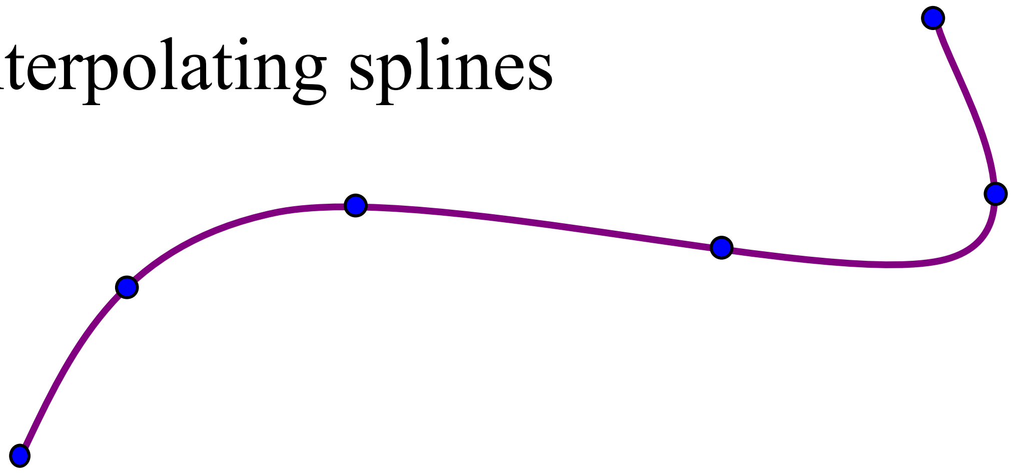
Now it is the smooth curve that is called a spline.

Interpolating Splines

Modern splines are smooth curves defined from a small set of points often called *knots*.

In one main class of splines, the curve must pass through each point of the set.

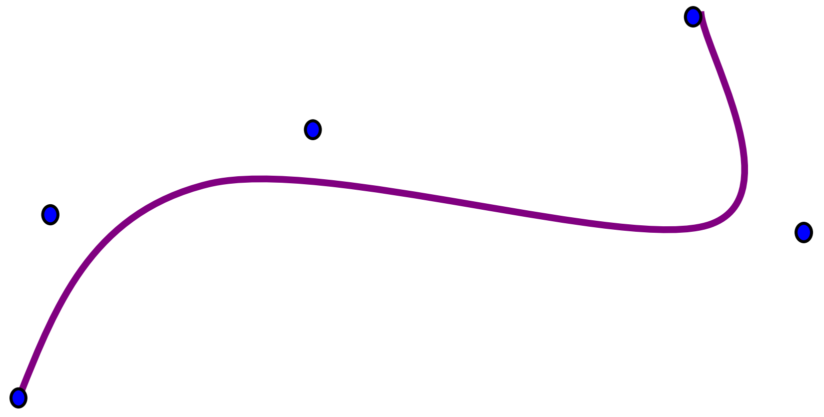
These are called interpolating splines



Approximating Splines

In other cases the curves do not pass through the points.

The points act as control points which the user can move to adjust the shape of the curve interactively



Non Parametric Spline

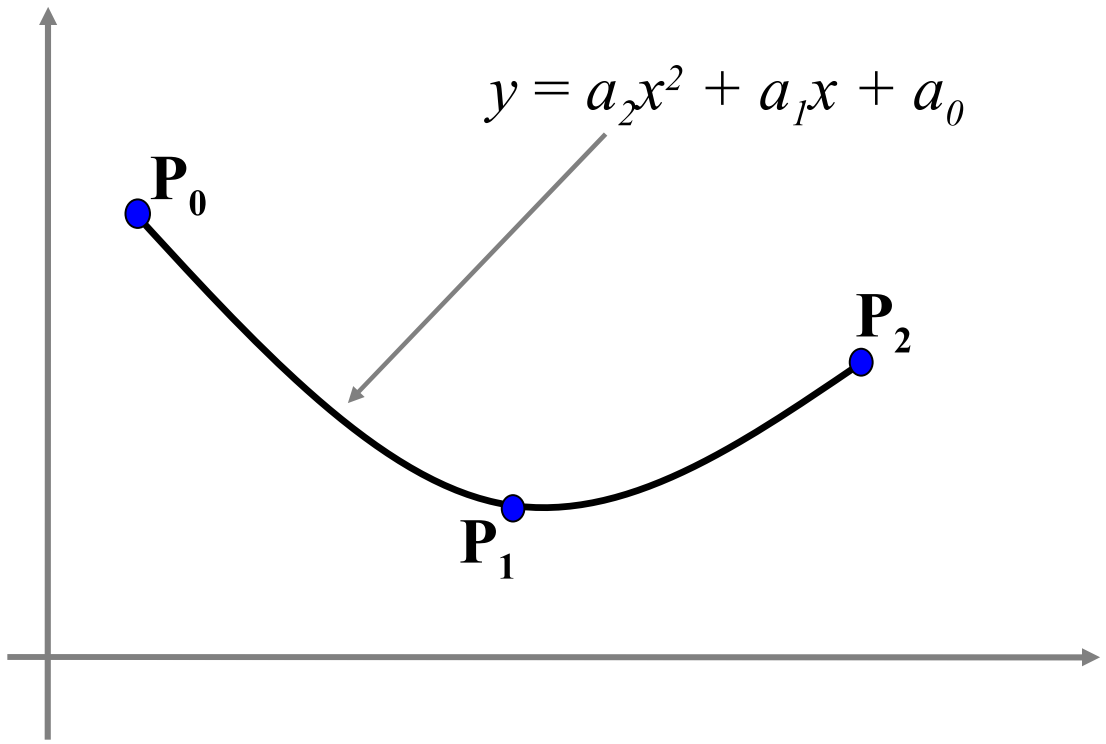
The simplest splines are just equations in x and y (for two dimensions)

The most common is the polynomial spline:

$y = a_2x^2 + a_1x + a_0$

given three points we can calculate a_2 a_1 and a_0

A non parametric (parabolic spline)



There is no control using non parametric splines.

Only one curve (a parabola) fits the data.

Control of non-Parametric Splines

There is no control using non parametric splines.

Only one curve (a parabola) fits the data.

Parametric Splines

If we write our spline in a vector form we get:

$$\mathbf{P} = \mathbf{a_2}\mu^2 + \mathbf{a_1}\,\mu + \mathbf{a_0}$$

which has a parameter μ

by convention, as μ ranges from 0 to 1 the point P traces out a curve.

Calculating simple parametric splines

We can now solve for the vector constants $\mathbf{a_0}$ $\mathbf{a_1}$ and $\mathbf{a_2}$ as follows.

Suppose we want the curve to start at point $\mathbf{P_0}$

$$\mathbf{P_0} = \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$$

we have $\mu=0$ at the start so

$$\mathbf{P_0} = \mathbf{a_0}$$

Calculating simple parametric splines

Suppose we want the spline to end at $\mathbf{P}_2$

we have that at the end $\mu = 1$

thus $\mathbf{P}_2 = \mathbf{a}_2\mu^2 + \mathbf{a}_1\mu + \mathbf{a}_0$

$= \mathbf{a}_2 + \mathbf{a}_1 + \mathbf{a}_0$

$= \mathbf{a}_2 + \mathbf{a}_1 + \mathbf{P}_0$

Calculating simple parametric splines

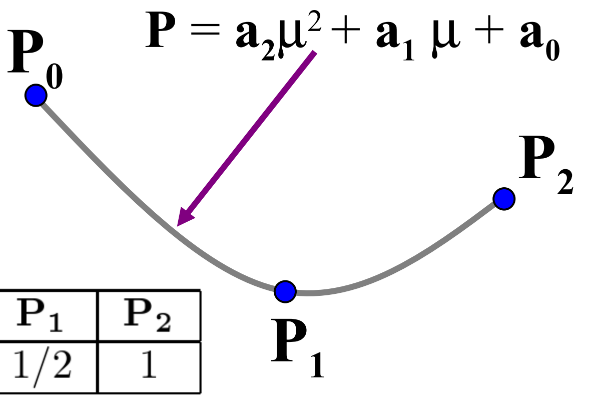
and in the middle (say $\mu = 1/2$) we want it to pass through $\mathbf{P_1}$

$$\mathbf{P_1} = \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$$
$$= \mathbf{a_2}/4 + \mathbf{a_1}/2 + \mathbf{P_0}$$

We have enough equations to solve for $\mathbf{a_1}$ and $\mathbf{a_2}$.

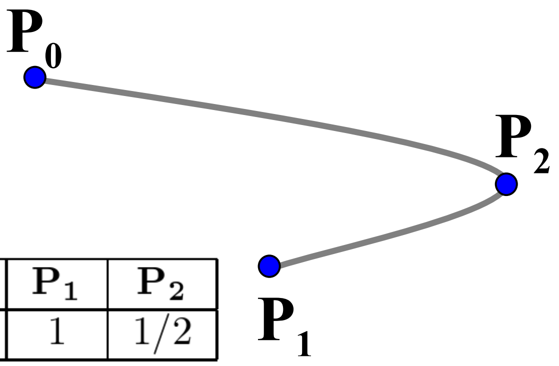
Notice that this formulation is the same in 2 and 3 dimensions.

Possibilities using parametric splines

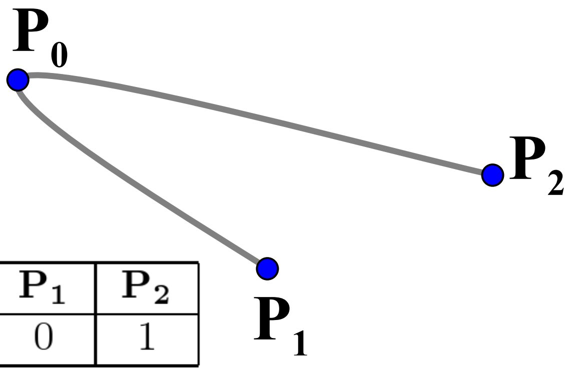


$\mathbf{P} = \mathbf{a}_2\mu^2 + \mathbf{a}_1 \mu + \mathbf{a}_0$
$\mathbf{P}_0$
$\mathbf{P}_1$
$\mathbf{P}_2$

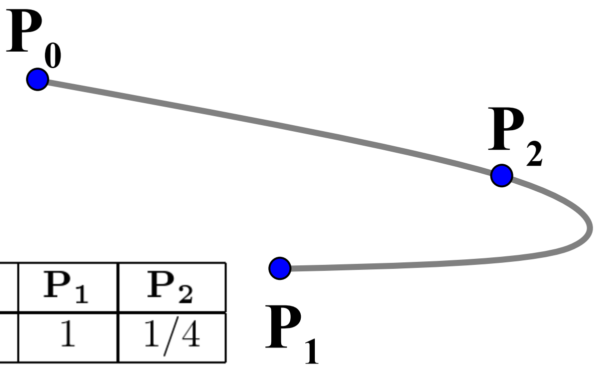
	$\mathbf{P}_0$	$\mathbf{P}_1$	$\mathbf{P}_2$
μ	0	1/2	1



	$\mathbf{P}_0$	$\mathbf{P}_1$	$\mathbf{P}_2$
μ	0	1	1/2



	$\mathbf{P}_0$	$\mathbf{P}_1$	$\mathbf{P}_2$
μ	1/2	0	1



	$\mathbf{P}_0$	$\mathbf{P}_1$	$\mathbf{P}_2$
μ	0	1	1/4

Higher order parametric splines

Parametric polynomial splines must have an order to match the number of knots.

3 knots - quadratic polynomial

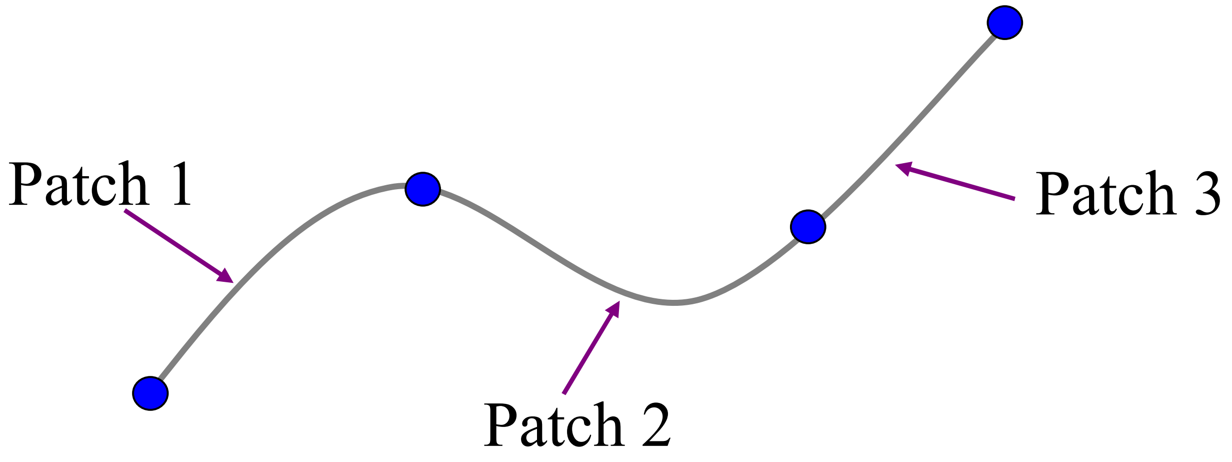
4 knots - cubic polynomial

etc.

Higher order polynomials are undesirable since they tend to oscillate

Spline Patches

To get round the problem, we can piece together a number of patches, each patch being a parametric spline.



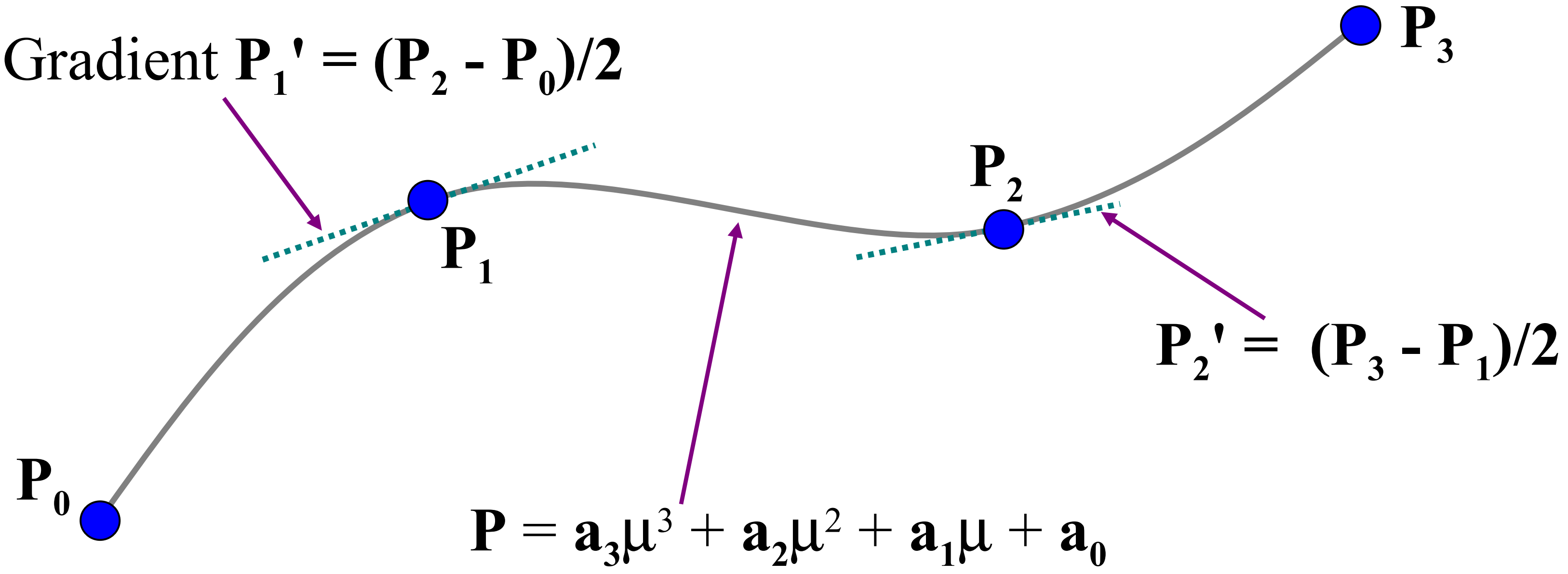
Cubic Spline Patches

The simplest, and most effective way to calculate parametric spline patches is to use a cubic polynomial.

$$\mathbf{P} = \mathbf{a_3}\mu^3 + \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$$

This allows us to join the patches together smoothly

Choosing the gradients



Gradient $\mathbf{P_1' = (P_2 - P_0)/2}$
$\mathbf{P_0}$
$\mathbf{P_1}$
$\mathbf{P_2}$
$\mathbf{P_3}$
$\mathbf{P_2' = (P_3 - P_1)/2}$
$\mathbf{P} = \mathbf{a_3}\mu^3 + \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$

Calculating a Cubic Spline Patch

$$\mathbf{P} = \mathbf{a}_3\mu^3 + \mathbf{a}_2\mu^2 + \mathbf{a}_1\mu + \mathbf{a}_0$$

for a patch joining points $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ we have $\mu=0$ at $\mathbf{P}_i$ and $\mu=1$ at $\mathbf{P}_{i+1}$

Substituting these values we get

$$\mathbf{P}_i = \mathbf{a}_0$$

$$\mathbf{P}_{i+1} = \mathbf{a}_3 + \mathbf{a}_2 + \mathbf{a}_1 + \mathbf{a}_0$$

Calculating a Cubic Spline Patch

differentiating $\mathbf{P} = \mathbf{a}_3\mu^3 + \mathbf{a}_2\mu^2 + \mathbf{a}_1\mu + \mathbf{a}_0$ we get

$\mathbf{P}' = 3\mathbf{a}_3\mu^2 + 2\mathbf{a}_2\mu + \mathbf{a}_1$

substituting for $\mu=0$ at $\mathbf{P}_i$ and $\mu=1$ at $\mathbf{P}_{i+1}$ we get

$\mathbf{P}'_i = \mathbf{a}_1$

$\mathbf{P}'_{i+1} = 3\mathbf{a}_3 + 2\mathbf{a}_2 + \mathbf{a}_1$

Calculating a Cubic Spline Patch

Putting these four equations into matrix form we get:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \begin{bmatrix} \mathbf{P_i} \\ \mathbf{P'_i} \\ \mathbf{P_{i+1}} \\ \mathbf{P'_{i+1}} \end{bmatrix}$$

Calculating a Cubic Spline Patch

Finally, inverting the matrix gives us the result we want. Notice that the matrix is the same for every patch

$$\begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & -3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P_i} \\ \mathbf{P'_i} \\ \mathbf{P_{i+1}} \\ \mathbf{P'_{i+1}} \end{bmatrix}$$

Bezier Curves

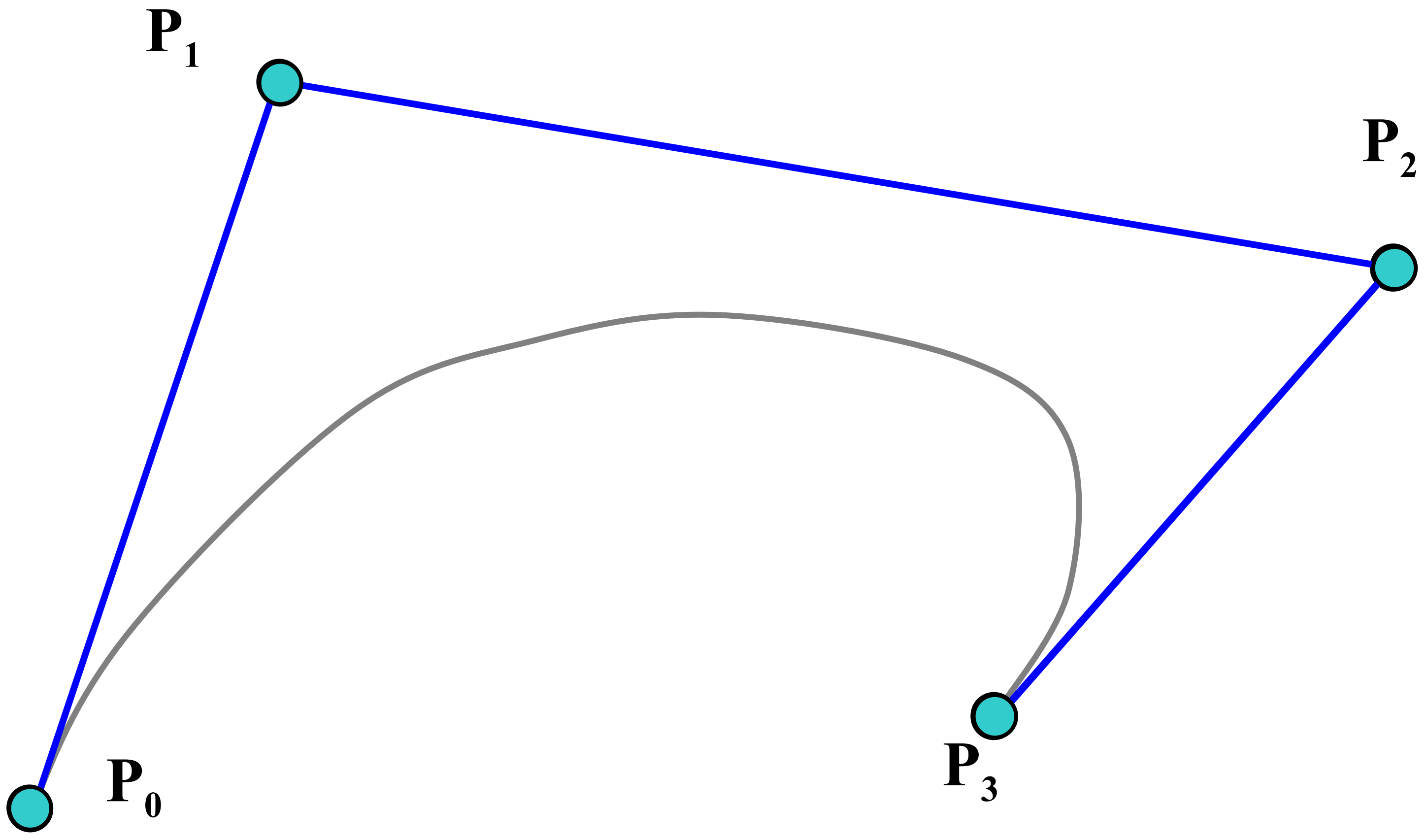
Bezier curves were developed as a method for CAD design. They give very predictable results for small sets of knots, and so are useful as spline patches.

The main characteristics of Bezier curves are

They interpolate the end points

The slope at an end is the same as the line joining the end point to its neighbour

A typical Bezier Curve



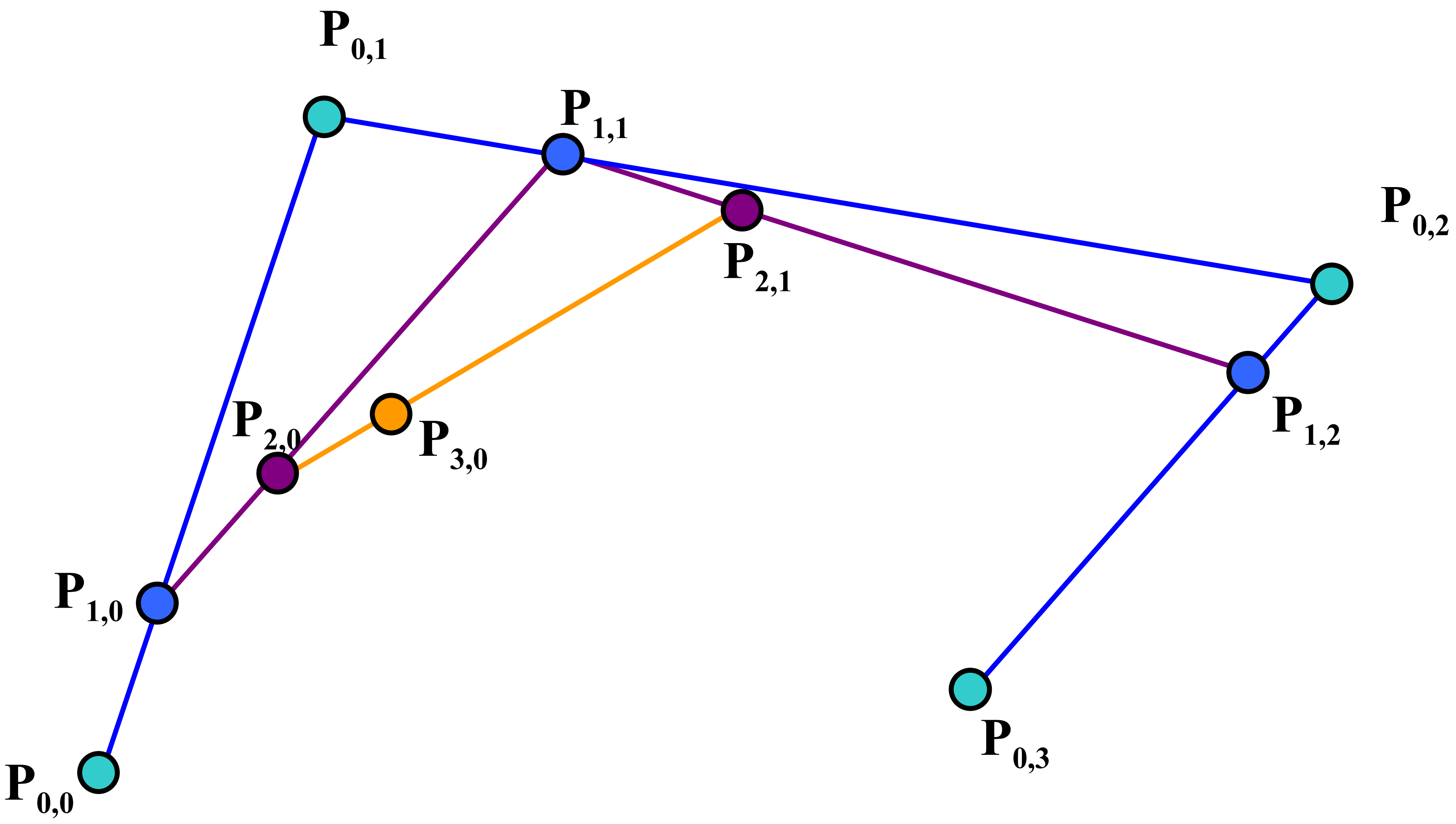
Casteljau's Algorithm

Bezier curves may be computed and visualised using a geometric construction due to Paul de Casteljau.

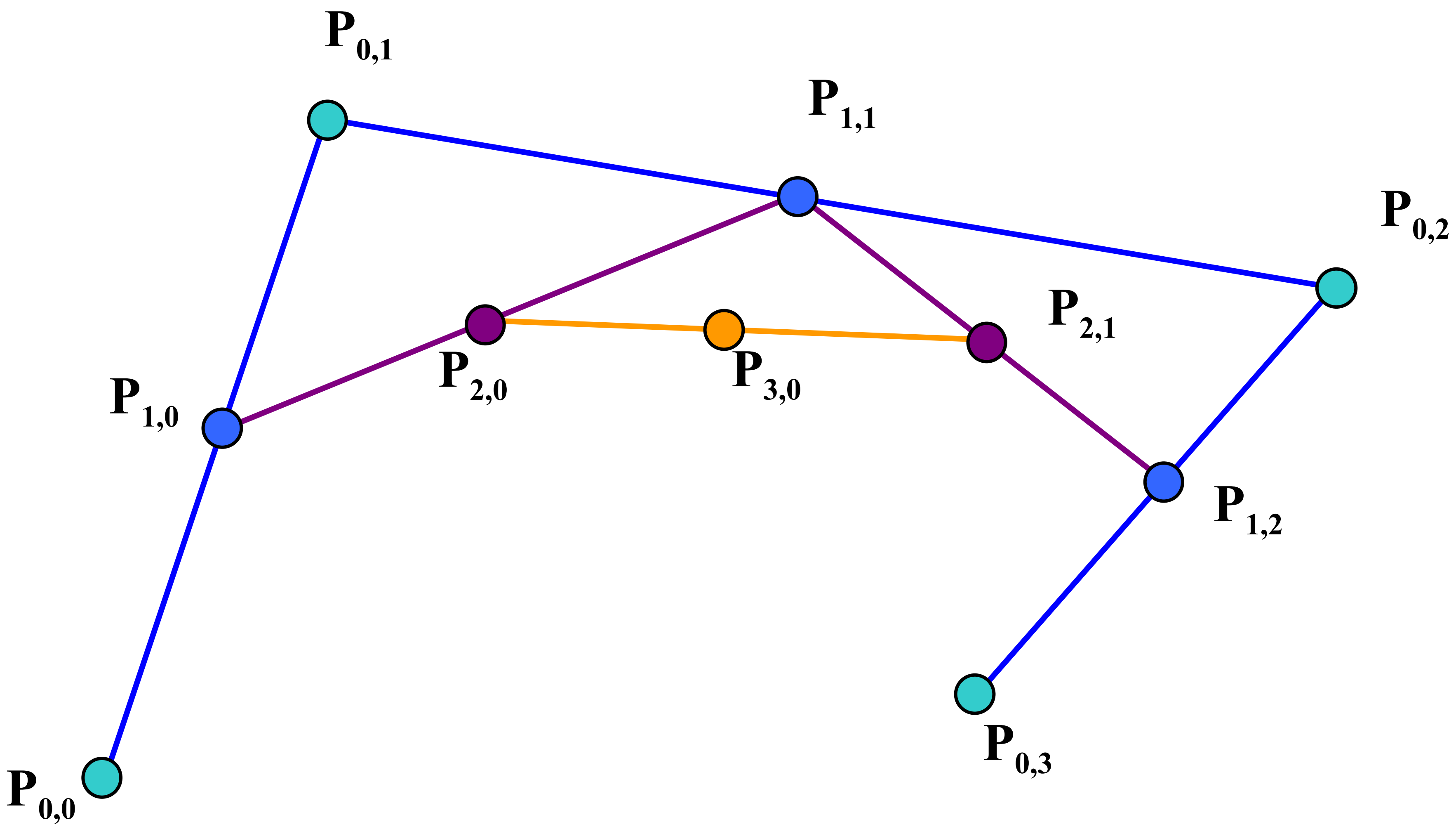
Like a cubic patch, we need a parameter μ which is to be 0 at the start of the curve, and 1 at the end.

A construction can be made for any value of μ

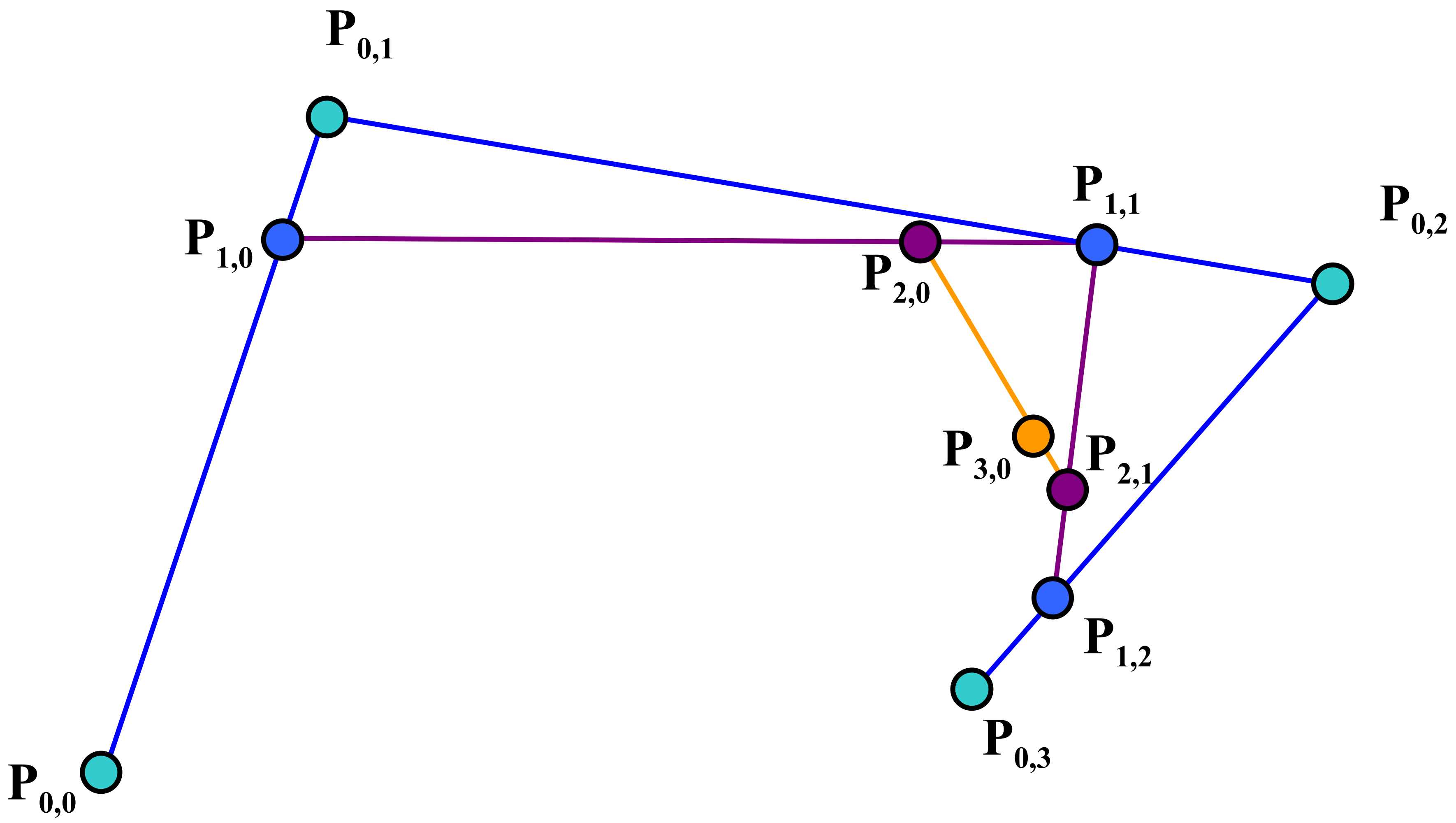
Casteljau's Construction $\mu = 0.25$



Casteljau's Construction $\mu = 0.5$



Casteljau's Construction $\mu = 0.75$



Bernstein Blending Function

Splines (including Bezier curves) can be formulated as a blend of the knots.

Consider the vector line equation

$$P = (1-\mu)\mathbf{P_0} + \mu\,\mathbf{P_1}$$

It is a linear ‘blend’ of two points, and could also be considered the 2 point Bezier curve!

Blending Equation

Any point on the spline is simply a blend of all the other points. For N+1 knots we have:

$$\mathbf{P}(\mu) = \sum_{i=0}^{N} \mathbf{P_i} W(N, i, \mu)$$

where W is the Bernstein blending function

$$W(N, I, \mu) = \binom{N}{i} \mu^i (i - \mu)^{N-i}$$

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

Expanded Bezier Equations

2 Point: $\mathbf{P}_0(1-\mu) + \mathbf{P}_1\mu$

3 Point: $\mathbf{P}_0(1-\mu)^2 + 2\mathbf{P}_1(1-\mu)\mu + \mathbf{P}_2\mu^2$

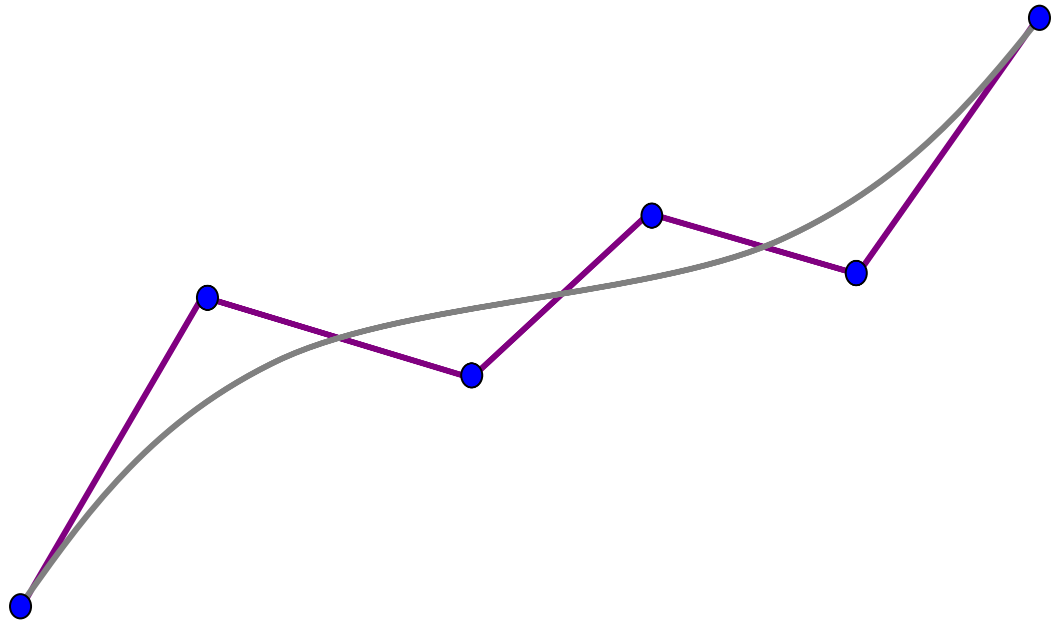
4 Point: $\mathbf{P}_0(1-\mu)^3 + 3\mathbf{P}_1(1-\mu)^2\mu + 3\mathbf{P}_2(1-\mu)\mu^2 + \mathbf{P}_3\mu^3$

etc

Bezier Curves lack local control

Since all the knots of the Bezier curve all appear in the blend they cannot be used for curves with fine detail.

However they are very effective as spline patches.



Four point Bezier Curves and Cubic Patches

Four point Bezier curves are equivalent to cubic patches going through the first and last knot (**P0** and **P3**)

It is possible to show their equivalence in two ways:

Expanding the iterative blending equation
Reversing the de Casteljau algorithm

Expanding the blending equation

For the case of four knots we can expand the Bernstein blending function to get a polynomial in μ:

$$\mathbf{P}(\mu) = \sum_{i=0}^{3} \mathbf{P_i} W(3, i, \mu)$$

$$\mathbf{P}(\mu) = \mathbf{P_0}(1-\mu)^3 + 3\mathbf{P_1}\mu(1-\mu)^2 + 3\mathbf{P_2}\mu^2(1-\mu) + \mathbf{P_3}\mu^3$$

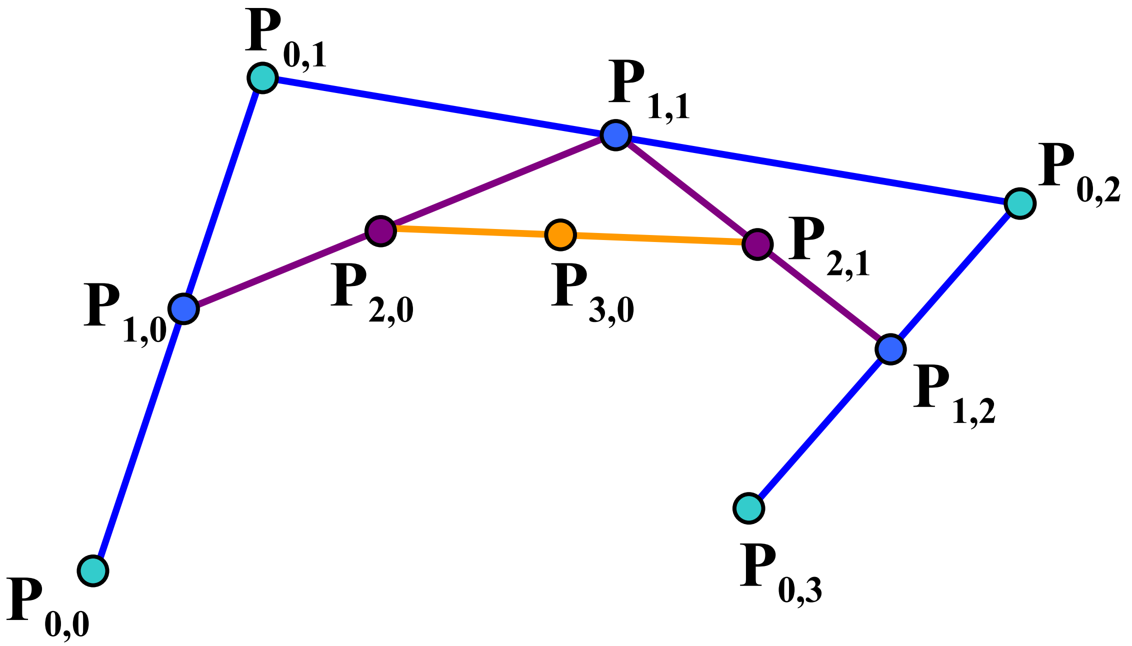
This can be multiplied out to give an equation of the form: $\mathbf{P}(\mu) = \mathbf{a_3}\mu^3 + \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$

where:

$$\begin{aligned}
\mathbf{a_3} &= \mathbf{P_3} - 3\mathbf{P_2} + 3\mathbf{P_1} - \mathbf{P_0} \\
\mathbf{a_2} &= 3\mathbf{P_2} - 6\mathbf{P_1} + 3\mathbf{P_0} \\
\mathbf{a_1} &= 3\mathbf{P_1} - 3\mathbf{P_0} \\
\mathbf{a_0} &= \mathbf{P_0}
\end{aligned}$$

Casteljau's algorithm gives the same result

We start from point $P_{3,0}$ and express it in terms of its construction line. Then the process is continued.



$$
\begin{aligned}
\mathbf{P}_{3,0} &= \mu\mathbf{P}_{2,1} + (1-\mu)\mathbf{P}_{2,0} \\
&= \mu[\mu\mathbf{P}_{1,2} + (1-\mu)\mathbf{P}_{1,1}] + (1-\mu)[\mu\mathbf{P}_{1,1} + (1-\mu)\mathbf{P}_{1,0}] \\
&= \mu^2\mathbf{P}_{1,2} + 2\mu(1-\mu)\mathbf{P}_{1,1} + (1-\mu)^2\mathbf{P}_{1,0} \\
&= \mu^2[\mu\mathbf{P}_{0,3} + (1-\mu)\mathbf{P}_{0,2}] + 2\mu(1-\mu)[\mu\mathbf{P}_{0,2} + (1-\mu)\mathbf{P}_{0,1}] \\
&\qquad +(1-\mu)^2[\mu\mathbf{P}_{0,1} + (1-\mu)\mathbf{P}_{0,0}]
\end{aligned}
$$

Continuing expanding

We can drop the first subscript (which indicates the recursion level) to get:

$$\begin{aligned}
\mathbf{P}(\mu) &= \mu^2[\mu\mathbf{P_3} + (1-\mu)\mathbf{P_2}] + 2\mu(1-\mu)[\mu\mathbf{P_2} + (1-\mu)\mathbf{P_1}] \\
&\quad +(1-\mu)^2[\mu\mathbf{P_1} + (1-\mu)\mathbf{P_0}] \\
&= \mu^3\mathbf{P_3} + 3\mu^2(1-\mu)\mathbf{P_3} + 3\mu(1-\mu)^2\mathbf{P_1} + (1-\mu)^3\mathbf{P_0}
\end{aligned}$$

which is the same as before

Control Points

We can summarise the four point Bezier Curve by saying that it has two points that are interpolated and two control points.

The curve starts at $\mathbf{P}_0$ and ends at $\mathbf{P}_3$ and its shape can be determined by moving control points ($\mathbf{P}_1$ and $\mathbf{P}_2$).

This could be done interactively using a mouse.

In summary

The simplest and most effective way to draw a smooth curve through a set of points is to use a cubic patch.

If no interaction is needed setting the gradients by the central difference $(\mathbf{P}_{i+1} - \mathbf{P}_{i-1})/2$ is effective.

If the user wants to interactively adjust the shape the four point Bezier formulation is ideal