Parametric Curves

Parametric Representations

- 3 basic representation strategies:
  - Explicit: $y = mx + b$
  - Implicit: $ax + by + c = 0$
  - Parametric: $P = P_0 + t (P_1 - P_0)$
- Advantages of parametric forms
  - More degrees of freedom
  - Directly transformable
  - Dimension independent
  - No infinite slope problems
  - Separates dependent and independent variables
  - Inherently bounded
  - Easy to express in vector and matrix form
  - Common form for many curves and surfaces

Algebraic Representation

- All of these curves are just parametric algebraic polynomials expressed in different bases
- Parametric linear curve (in E^3)

$$\mathbf{p}(u) = \mathbf{a}u + \mathbf{b}$$

$$x = a_x u + b_x$$
$$y = a_y u + b_y$$
$$z = a_z u + b_z$$

- Parametric cubic curve (in E^3)

$$\mathbf{p}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}$$

$$x = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z = a_z u^3 + b_z u^2 + c_z u + d_z$$

- Basis (monomial or power)

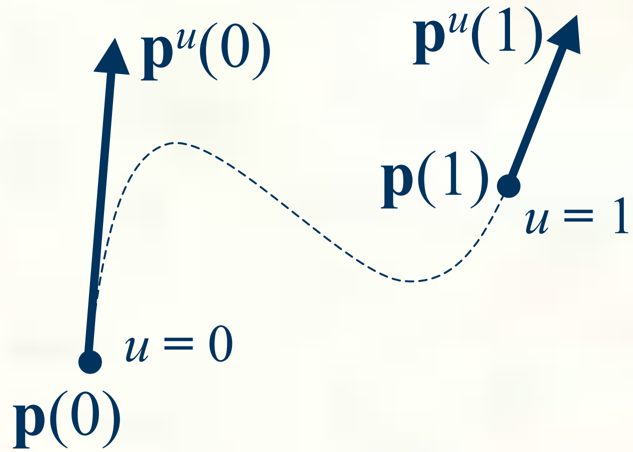
$$\begin{bmatrix} u & 1 \end{bmatrix}$$
$$\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

Hermite Curves

- 12 degrees of freedom (4 3-d vector constraints)
- Specify endpoints and tangent vectors at endpoints

$$\mathbf{p}(0) = \mathbf{d}$$
$$\mathbf{p}(1) = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$
$$\mathbf{p}^u(0) = \mathbf{c}$$
$$\mathbf{p}^u(1) = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$

$$\mathbf{p}^u(u) \equiv \frac{d\mathbf{p}}{du}(u)$$



- Solving for the coefficients:

$$\mathbf{a} = 2\mathbf{p}(0) - 2\mathbf{p}(1) + \mathbf{p}^u(0) + \mathbf{p}^u(1)$$
$$\mathbf{b} = -3\mathbf{p}(0) + 3\mathbf{p}(1) - 2\mathbf{p}^u(0) - \mathbf{p}^u(1)$$
$$\mathbf{c} = \mathbf{p}^u(0)$$
$$\mathbf{d} = \mathbf{p}(0)$$

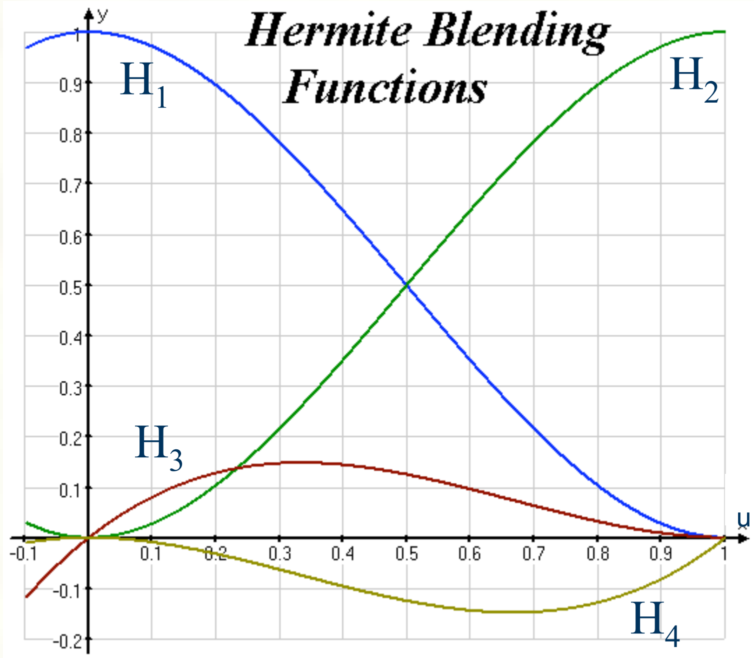
Hermite Curves - Hermite Basis

- Substituting for the coefficients and collecting terms gives

$$\mathbf{p}(u) = (2u^3 - 3u^2 + 1)\mathbf{p}(0) + (-2u^3 + 3u^2)\mathbf{p}(1) + (u^3 - 2u^2 + u)\mathbf{p}^u(0) + (u^3 - u^2)\mathbf{p}^u(1)$$

- Call

$$H_1(u) = (2u^3 - 3u^2 + 1)$$
$$H_2(u) = (-2u^3 + 3u^2)$$
$$H_3(u) = (u^3 - 2u^2 + u)$$
$$H_4(u) = (u^3 - u^2)$$



the Hermite **blending functions** or **basis functions**

- Then $\mathbf{p}(u) = H_1(u)\mathbf{p}(0) + H_2(u)\mathbf{p}(1) + H_3(u)\mathbf{p}^u(0) + H_4(u)\mathbf{p}^u(1)$

Hermite Curves - Matrix Form

- Putting this in matrix form

$$\mathbf{H} = \begin{bmatrix} H_1(u) & H_2(u) & H_3(u) & H_4(u) \end{bmatrix}$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$= \mathbf{UM}_{\mathrm{H}}$$

- $\mathbf{M}_{\mathrm{H}}$ is called the Hermite **characteristic matrix**
- Collecting the Hermite geometric coefficients into a geometry vector **B**, we have a matrix formulation for the Hermite curve $\mathbf{p}(u)$

$$\mathbf{B} = \begin{bmatrix} \mathbf{p}(0) \\ \mathbf{p}(1) \\ \mathbf{p}^u(0) \\ \mathbf{p}^u(1) \end{bmatrix}$$

$$\mathbf{p}(u) = \mathbf{UM}_{\mathrm{H}}\mathbf{B}$$

Hermite and Algebraic Forms

- $\mathbf{M}_\mathrm{H}$ transforms geometric coefficients ("coordinates") from the Hermite basis to the algebraic coefficients of the monomial basis

$$\mathbf{A} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

$$\mathbf{p}(u) = \mathbf{UA} = \mathbf{UM}_\mathrm{H}\mathbf{B}$$

$$\mathbf{A} = \mathbf{M}_\mathrm{H}\mathbf{B}$$

$$\mathbf{B} = \mathbf{M}_\mathrm{H}^{-1}\mathbf{A}$$

$$\mathbf{M}_\mathrm{H}^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}$$

Cubic Bézier Curves

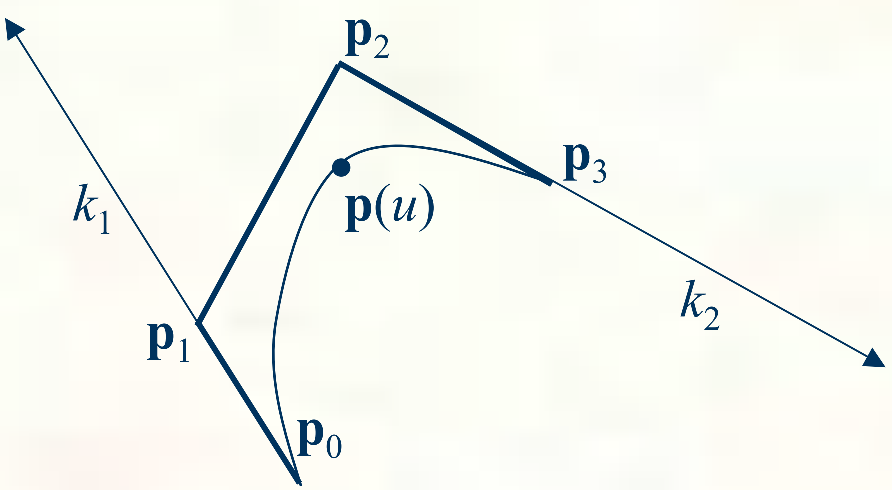
- Specifying tangent vectors at endpoints isn't always convenient for geometric modeling
- We may prefer making all the geometric coefficients points, let's call them **control points,** and label them $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$
- For cubic curves, we can proceed by letting the tangents at the endpoints for the Hermite curve be defined by a vector between a pair of control points, so that:

$$\mathbf{p}(0) = \mathbf{p}_0$$

$$\mathbf{p}(1) = \mathbf{p}_3$$

$$\mathbf{p}^u(0) = k_1(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{p}^u(1) = k_2(\mathbf{p}_3 - \mathbf{p}_2)$$



Cubic Bézier Curves

- Substituting this into the Hermite curve expression and rearranging, we get

$$\mathbf{p}(u) = \left[(2-k_1)u^3 + (2k_1-3)u^2 - k_1u + 1\right]\mathbf{p}_0 + \left[k_1u^3 - 2k_1u^2 + k_1u\right]\mathbf{p}_1$$
$$+ \left[-k_2u^3 + k_2u^2\right]\mathbf{p}_2 + \left[(k_2-2)u^3 + (3-k_2)u^2\right]\mathbf{p}_3$$

- In matrix form, this is

$$\mathbf{p}(u) = \mathbf{U}\mathbf{M}_{\mathrm{B}}\mathbf{P} \qquad \mathbf{M}_{\mathrm{B}} = \begin{bmatrix} 2-k_1 & k_1 & -k_2 & k_2-2 \\ 2k_1-3 & -2k_1 & k_2 & 3-k_2 \\ -k_1 & k_1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Cubic Bézier Curves

- What values should we choose for k_1 and k_2?
- If we let the control points be evenly spaced in parameter space, then $\mathbf{p}_0$ is at $u = 0$, $\mathbf{p}_1$ at $u = 1/3$, $\mathbf{p}_2$ at $u = 2/3$ and $\mathbf{p}_3$ at $u = 1$. Then

$$\mathbf{p}^u(0) = (\mathbf{p}_1 - \mathbf{p}_0)/(1/3 - 0) = 3(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{p}^u(1) = (\mathbf{p}_3 - \mathbf{p}_2)/(1 - 2/3) = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

  and $k_1 = k_2 = 3$, giving a nice symmetric characteristic matrix:

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

- So

$$\mathbf{p}(u) = \left(-u^3 + 3u^2 - 3u + 1\right)\mathbf{p}_0 + \left(3u^3 - 6u^2 + 3u\right)\mathbf{p}_1 + \left(-3u^3 + 3u^2\right)\mathbf{p}_2 + u^3\mathbf{p}_3$$

General Bézier Curves

- This can be rewritten as

$$\mathbf{p}(u) = (1-u)^3\mathbf{p}_0 + 3u(1-u)^2\mathbf{p}_1 + 3u^2(1-u)\mathbf{p}_2 + u^3\mathbf{p}_3 = \sum_{i=0}^{3}\binom{3}{i}u^i(1-u)^{3-i}\mathbf{p}_i$$

- Note that the binomial expansion of

$(u + (1 - u))^n$ is $\sum_{i=0}^{n}\binom{n}{i}u^i(1-u)^{n-i}$

- This suggests a general formula for Bézier curves of arbitrary degree

$$\mathbf{p}(u) = \sum_{i=0}^{n}\binom{n}{i}u^i(1-u)^{n-i}\mathbf{p}_i$$

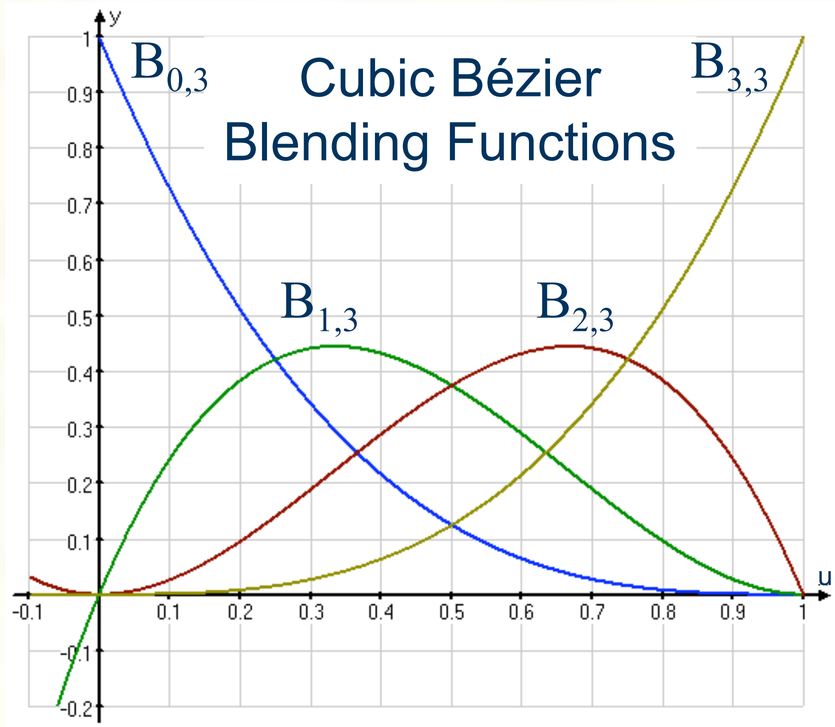
General Bézier Curves

- The binomial expansion gives the Bernstein basis (or Bézier blending functions) $B_{i,n}$ for arbitrary degree Bézier curves

$$\mathbf{p}(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} \mathbf{p}_i$$

$$B_{i,n}(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

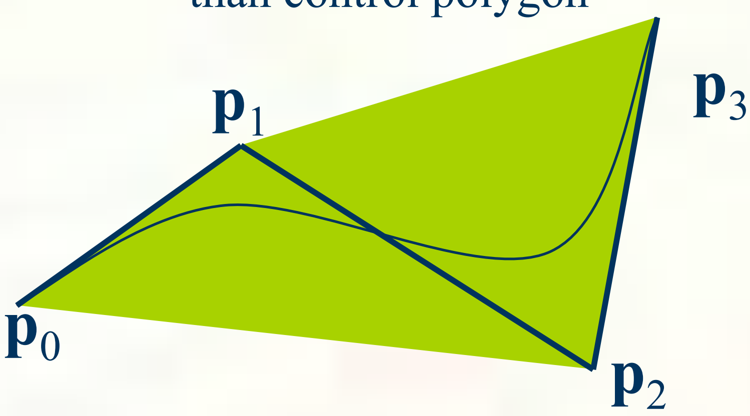
$$\mathbf{p}(u) = \sum_{i=0}^{n} B_{i,n}(u)\, \mathbf{p}_i$$

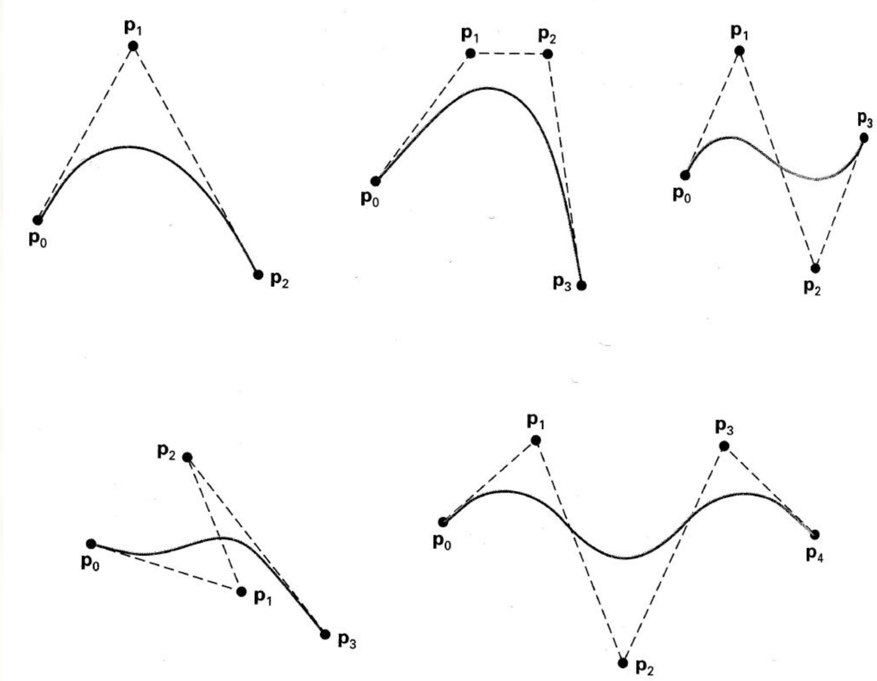


- Of particular interest to us (in addition to cubic curves):
  - Linear: $\mathbf{p}(u) = (1 - u)\mathbf{p}_0 + u\mathbf{p}_1$
  - Quadratic: $\mathbf{p}(u) = (1 - u)^2\mathbf{p}_0 + 2u(1 - u)\mathbf{p}_1 + u^2\mathbf{p}_2$

Bézier Curve Properties

- Interpolates end control points, not middle ones
- Stays inside **convex hull** of control points
  - Important for many algorithms
  - Because it's a convex combination of points, i.e. affine with positive weights
- Variation diminishing
  - Doesn't "wiggle" more than control polygon





Rendering Bézier Curves

- We can obtain a point on a Bézier curve by just evaluating the function for a given value of u
- Fastest way, precompute $\mathbf{A}=\mathbf{M}_B\mathbf{P}$ once control points are known, then evaluate $\mathbf{p}(u_i)=[u_i^3\ u_i^2\ u_i\ 1]\mathbf{A}$, $i = 0,1,2,\ldots,n$ for n fixed increments of u
- For better numerical stability, take e.g. a quadratic curve (for simplicity) and rewrite

$$\begin{aligned}\mathbf{p}(u) &= (1-u)^2\mathbf{p}_0 + 2u(1-u)\mathbf{p}_1 + u^2\mathbf{p}_2 \\ &= (1-u)[(1-u)\mathbf{p}_0 + u\mathbf{p}_1] + u[(1-u)\mathbf{p}_1 + u\mathbf{p}_2]\end{aligned}$$

- This is just a linear interpolation of two points, each of which was obtained by interpolating a pair of adjacent control points

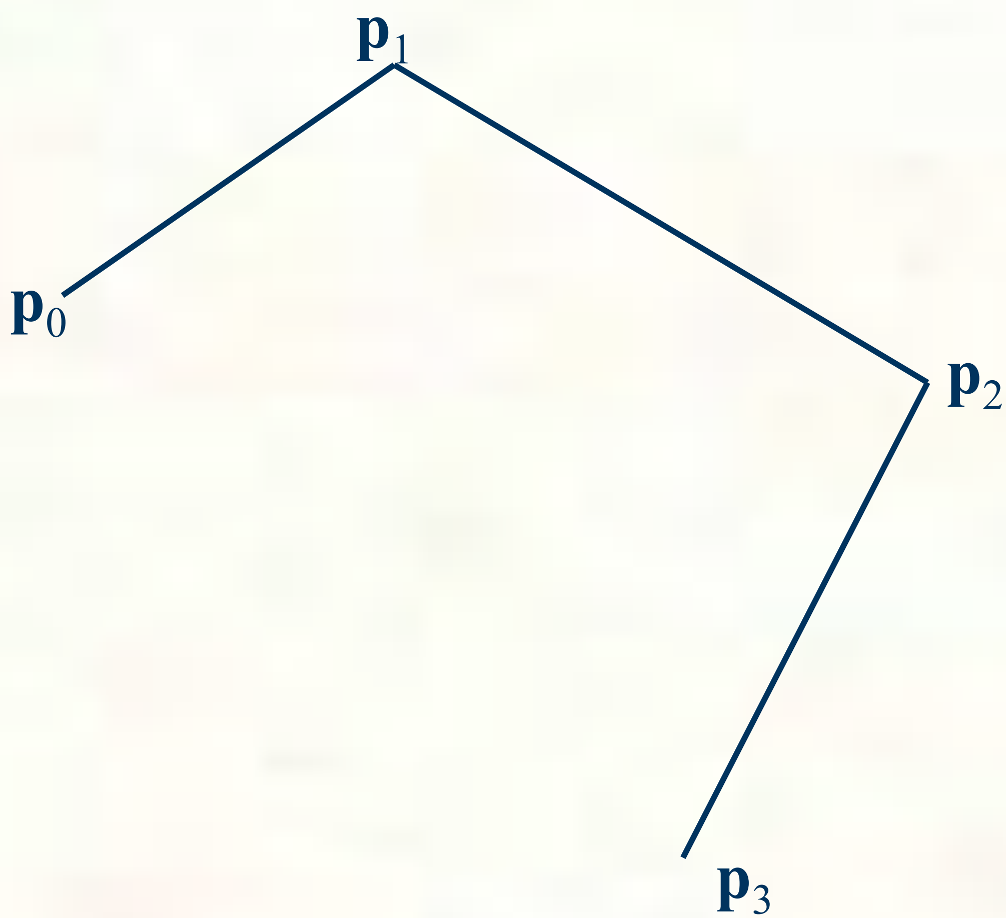
de Casteljau Algorithm

- This hierarchical linear interpolation works for general Bézier curves, as given by the following recurrence

$$\mathbf{p}_{i,j} = (1-u)\mathbf{p}_{i,j-1} + u\mathbf{p}_{i+1,j-1} \quad \begin{cases} i = 0,1,2,\ldots,n-j \\ j = 1,2,\ldots,n \end{cases}$$

  where $\mathbf{p}_{i,0} \quad i = 0,1,2,\ldots,n$ are the control points for a degree n Bézier curve and $\mathbf{p}_{0,n} = \mathbf{p}(u)$
- For efficiency this should not be implemented recursively.
- Useful for point evaluation in a recursive subdivision algorithm to render a curve since it generates the control points for the subdivided curves.

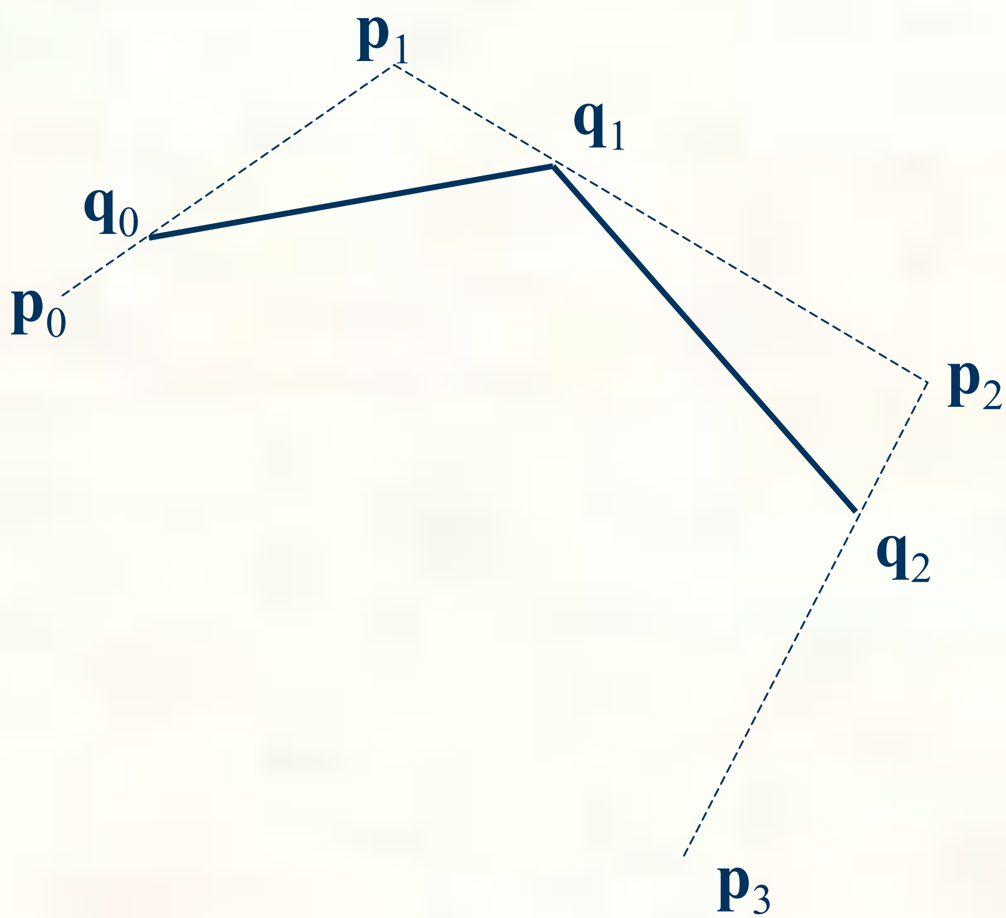
de Casteljau Algorithm



Starting with the control points and a given value of *u*

In this example, u≈0.25

de Casteljau Algorithm

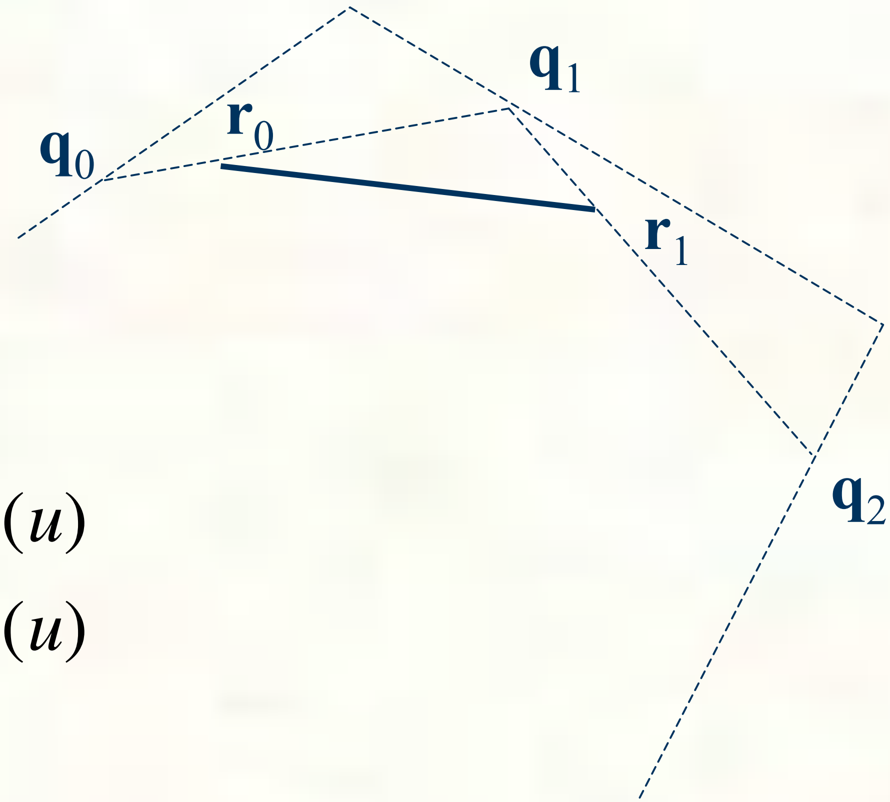


$$\mathbf{q}_0(u) = (1-u)\mathbf{p}_0 + u\mathbf{p}_1$$

$$\mathbf{q}_1(u) = (1-u)\mathbf{p}_1 + u\mathbf{p}_2$$

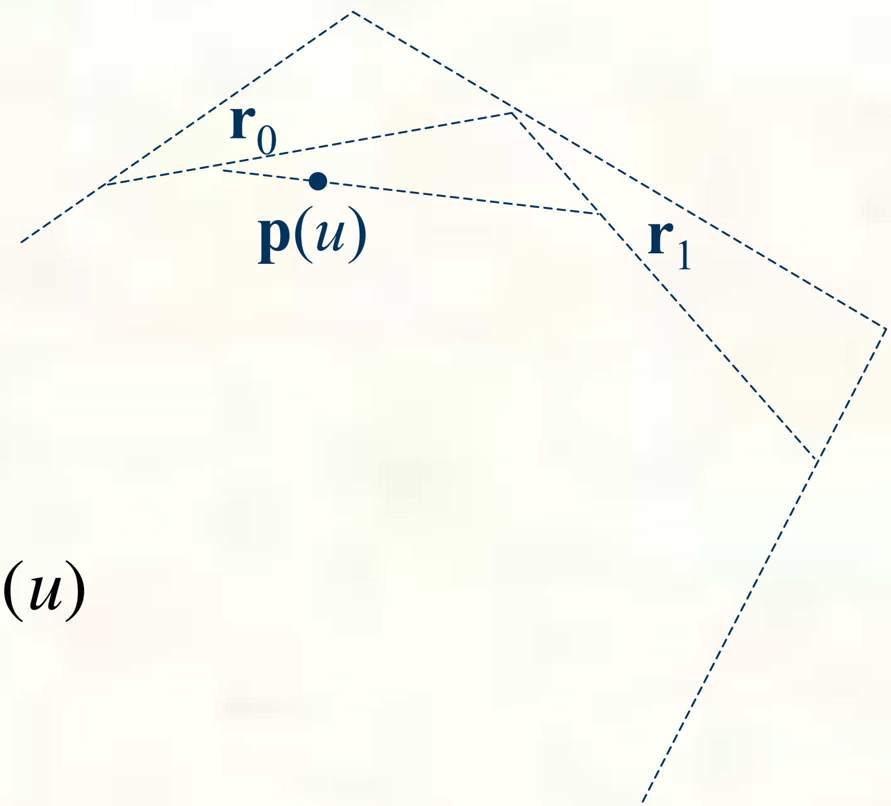
$$\mathbf{q}_2(u) = (1-u)\mathbf{p}_2 + u\mathbf{p}_3$$

de Casteljau Algorithm



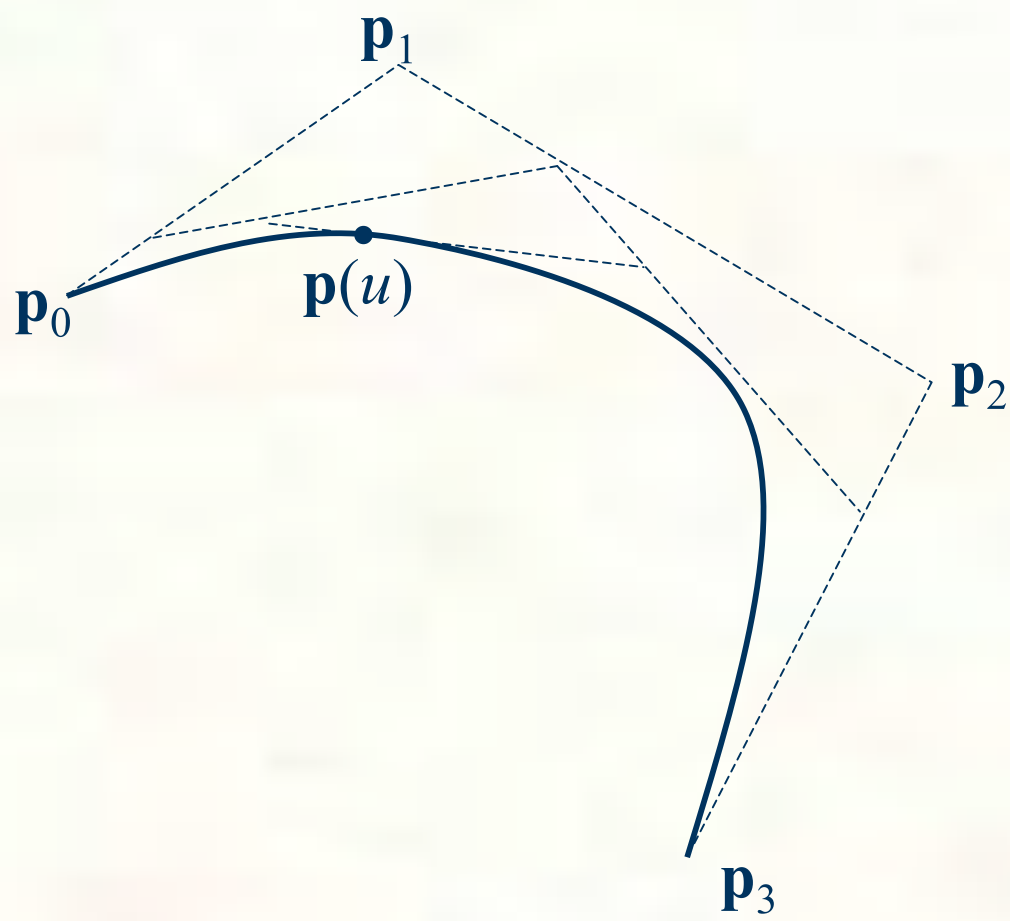
$$\mathbf{r}_0(u) = (1 - u)\mathbf{q}_0(u) + u\mathbf{q}_1(u)$$
$$\mathbf{r}_1(u) = (1 - u)\mathbf{q}_1(u) + u\mathbf{q}_2(u)$$

de Casteljau Algorithm



$$\mathbf{p}(u) = (1-u)\mathbf{r}_0(u) + u\mathbf{r}_1(u)$$

de Casteljau algorihm



Drawing Bézier Curves

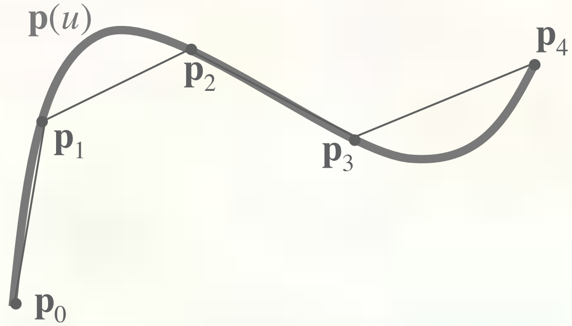
- How can you draw a curve?
  - Generally no low-level support for drawing curves
  - Can only draw line segments or individual pixels
- Approximate the curve as a series of line segments
  - Analogous to tessellation of a surface
  - Methods:
    - Sample uniformly
    - Sample adaptively
    - Recursive Subdivision

Uniform Sampling

- Approximate curve with n line segments
  - n chosen in advance
  - Evaluate $\mathbf{p}_i = \mathbf{p}(u_i)$ where $u_i = \frac{i}{n}$ $\quad i = 0,1,\ldots,n$
  - For an arbitrary cubic curve

$$\mathbf{p}_i = \mathbf{a}\left(i^3/n^3\right) + \mathbf{b}\left(i^2/n^2\right) + \mathbf{c}\left(i/n\right) + \mathbf{d}$$

  - Connect the points with lines



- Too few points?
  - Bad approximation
  - "Curve" is faceted
- Too many points?
  - Slow to draw too many line segments
  - Segments may draw on top of each other

Adaptive Sampling

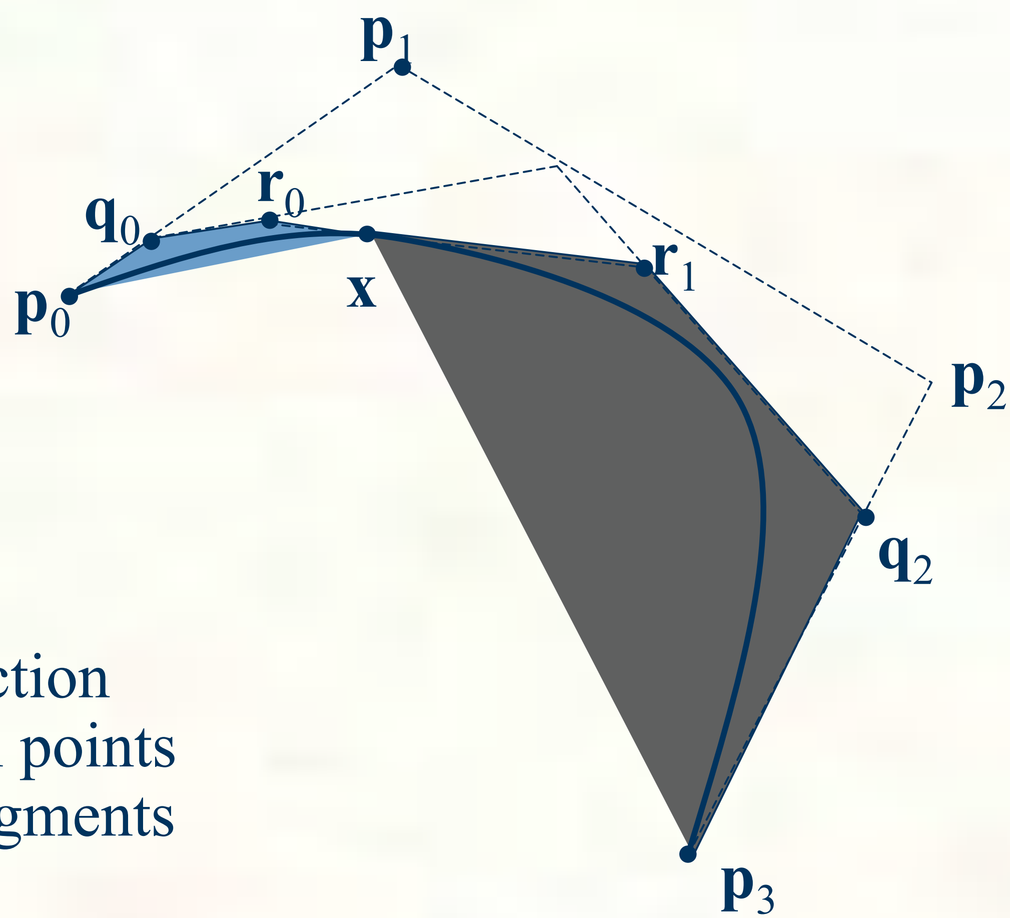
- Use only as many line segments as you need
  - Fewer segments needed where curve is mostly flat
  - More segments needed where curve bends
  - No need to track bends that are smaller than a pixel
- Various schemes for sampling, checking results, deciding whether to sample more

- Or, use knowledge of curve structure:
  - Adapt by recursive subdivision

Recursive Subdivision

- Any cubic curve segment can be expressed as a Bézier curve
- Any piece of a cubic curve is itself a cubic curve
- Therefore:
  - Any Bézier curve can be broken up into smaller Bézier curves
  - But how…?

de Casteljau subdivision



de Casteljau construction points are the control points of two Bézier sub-segments

Adaptive subdivision algorithm

- Use de Casteljau construction to split Bézier segment
- Examine each half:
  - If flat enough: draw line segment
  - Else: recurse

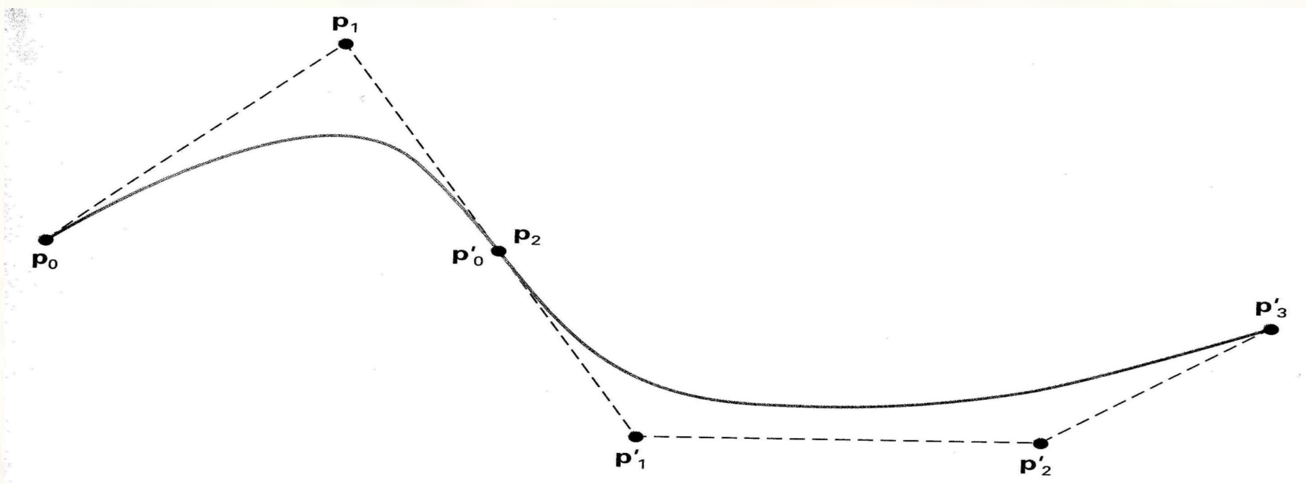
- To test if curve is flat enough
  - Only need to test if hull is flat enough
    - Curve is guaranteed to lie within the hull
  - e.g., test how far the handles are from a straight segment
    - If it's about a pixel, the hull is flat

Composite Curves

- Hermite and Bézier curves generalize line segments to higher degree polynomials. But what if we want more complicated curves than we can get with a single one of these? Then we need to build composite curves, like polylines but curved.
- Continuity conditions for composite curves
  - C^0 - The curve is continuous, i.e. the endpoints of consecutive curve segments coincide
  - C^1 - The tangent (derivative with respect to the **parameter**) is continuous, i.e. the tangents match at the common endpoint of consecutive curve segments
  - C^2 - The second parametric derivative is continuous, i.e. matches at common endpoints
  - G^0 - Same as C^0
  - G^1 - Derivatives wrt the coordinates are continuous. Weaker than C^1, the tangents should point in the same direction, but lengths can differ.
  - G^2 - Second derivatives wrt the coordinates are continuous
  - …

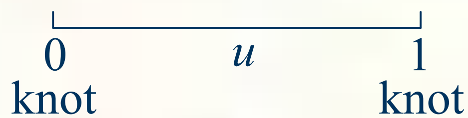
Composite Bézier Curves

- C^0, G^0 - Coincident end control points
- C^1 - $\mathbf{p}_3$ - $\mathbf{p}_2$ on first curve equals $\mathbf{p}_1$ - $\mathbf{p}_0$ on second
- G^1 - $\mathbf{p}_3$ - $\mathbf{p}_2$ on first curve proportional to $\mathbf{p}_1$ - $\mathbf{p}_0$ on second
- C^2, G^2 - More complex, use B-splines to automatically control continuity across curve segments



Polar form for Bézier Curves

- A much more useful point labeling scheme
- Start with **knots**, “interesting” values in parameter space
- For Bézier curves, parameter space is normally [0, 1], and the knots are at 0 and 1.



- Now build a **knot vector**, a non-decreasing sequence of knot values.
- For a degree n Bézier curve, the knot vector will have n 0’s followed by n 1’s [0,0,…,0,1,1,…,1]
  - Cubic Bézier knot vector [0,0,0,1,1,1]
  - Quadratic Bézier knot vector [0,0,1,1]
- **Polar labels** for consecutive control points are sequences of n knots from the vector, incrementing the starting point by 1 each time
  - Cubic Bézier control points: $\mathbf{p}_0 = \mathbf{p}(0,0,0)$, $\mathbf{p}_1 = \mathbf{p}(0,0,1)$, $\mathbf{p}_2 = \mathbf{p}(0,1,1)$, $\mathbf{p}_3 = p(1,1,1)$
  - Quadratic Bézier control points: $\mathbf{p}_0 = \mathbf{p}(0,0)$, $\mathbf{p}_1 = \mathbf{p}(0,1)$, $\mathbf{p}_2 = \mathbf{p}(1,1)$

Polar form rules

- Polar values are symmetric in their arguments, i.e. all permutations of a polar label are equivalent.

  $\mathbf{p}(0,0,1) = \mathbf{p}(0,1,0) = \mathbf{p}(1,0,0)$, etc.
- Given $\mathbf{p}(u_1, u_2,\ldots,u_{n\text{-}1}, a)$ and $\mathbf{p}(u_1, u_2,\ldots,u_{n\text{-}1}, b)$, for any value c we can compute

$$\mathbf{p}(u_1,u_2,\ldots,u_{n-1},c) = \frac{(b-c)\mathbf{p}(u_1,u_2,\ldots,u_{n-1},a) + (c-a)\mathbf{p}(u_1,u_2,\ldots,u_{n-1},b)}{b-a}$$

That is, $\mathbf{p}(u_1, u_2,\ldots,u_{n\text{-}1}, c)$ is an affine combination of $\mathbf{p}(u_1, u_2,\ldots,u_{n\text{-}1}, a)$ and $\mathbf{p}(u_1, u_2,\ldots,u_{n\text{-}1}, b)$.

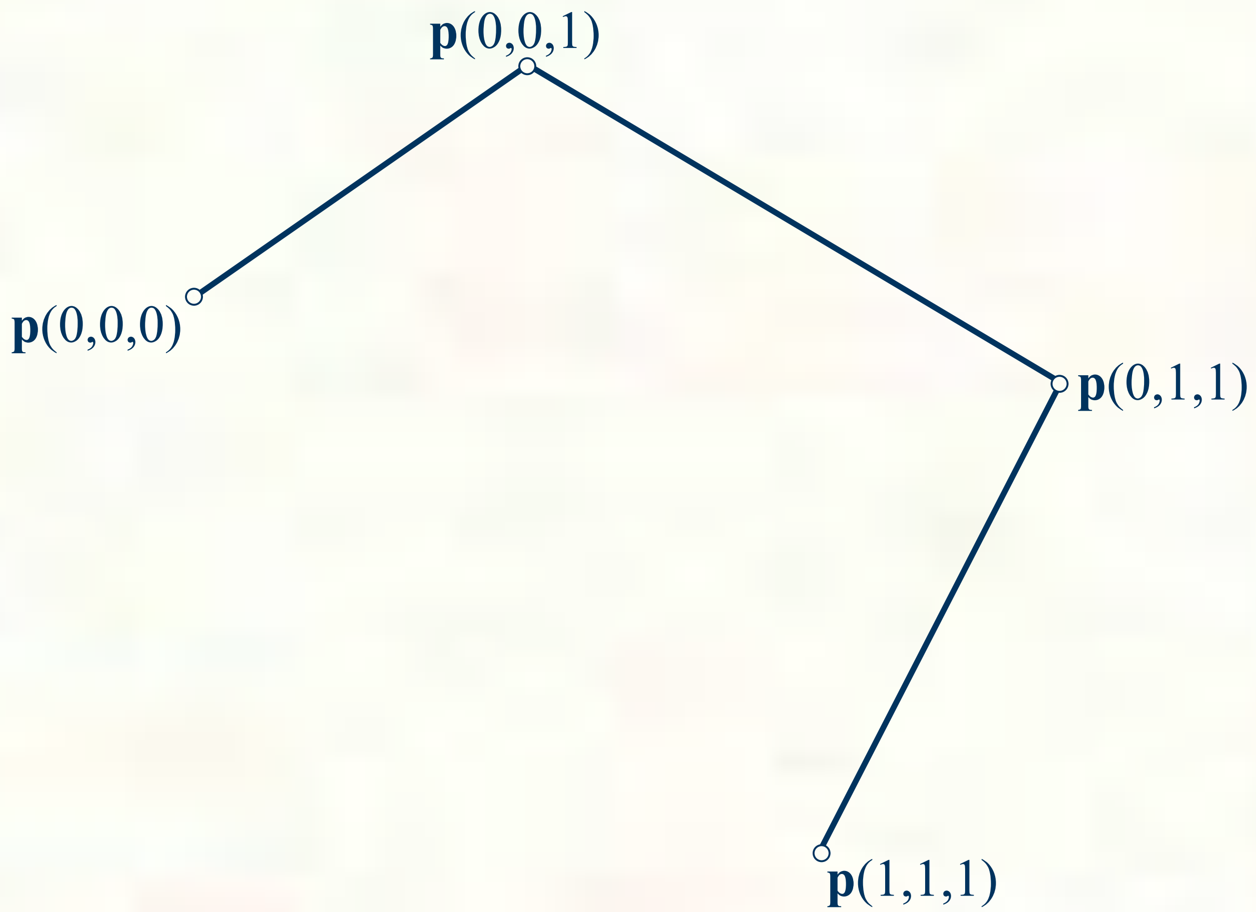
Examples:

$$\mathbf{p}(0,u,1) = (1-u)\mathbf{p}(0,0,1) + u\mathbf{p}(0,1,1)$$

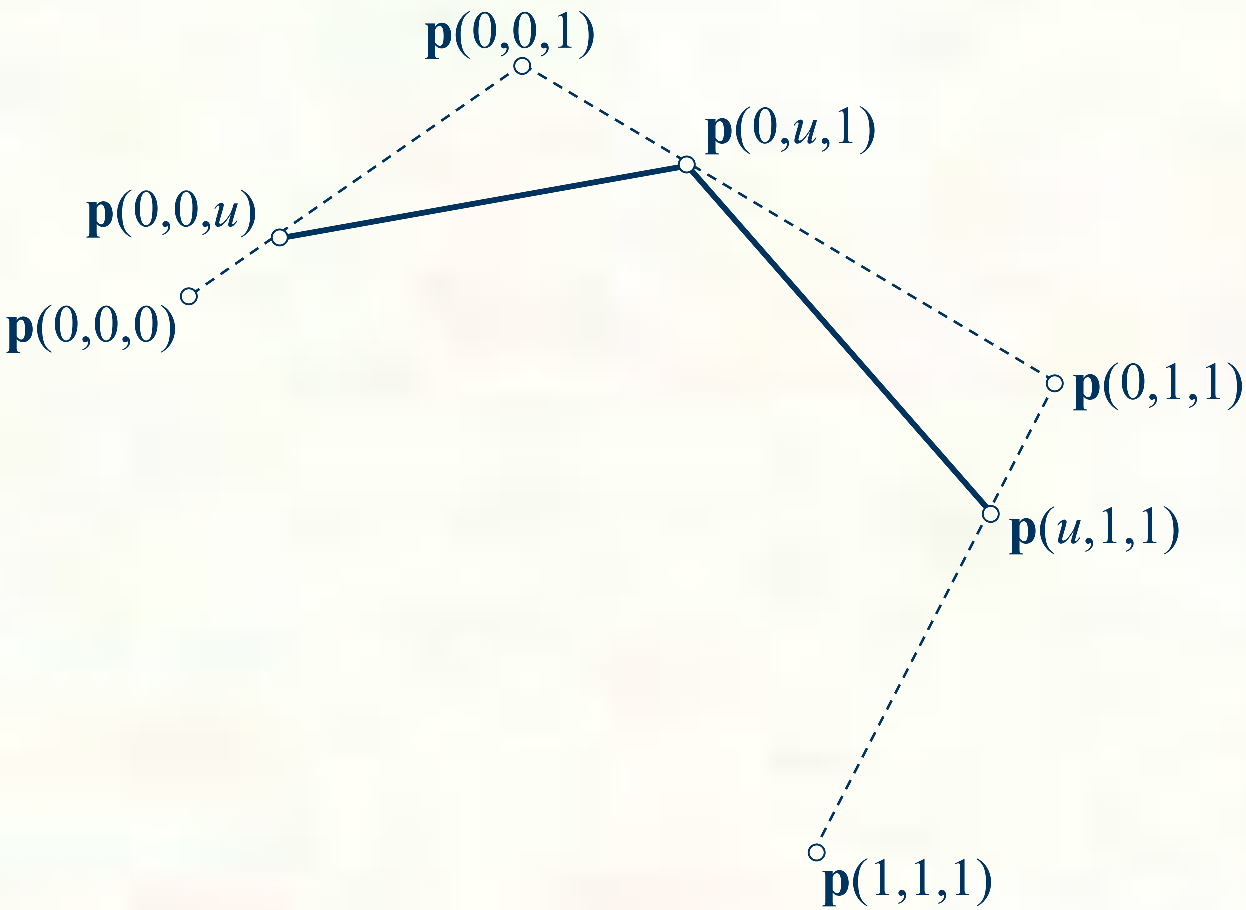
$$\mathbf{p}(0,u) = \frac{(4-u)\mathbf{p}(0,2) + (u-2)\mathbf{p}(0,4)}{2}$$

$$\mathbf{p}(1,2,3,u) = \frac{(u_2-u)\mathbf{p}(2,1,3,u_1) + (u-u_1)\mathbf{p}(3,2,1,u_2)}{u_2-u_1}$$

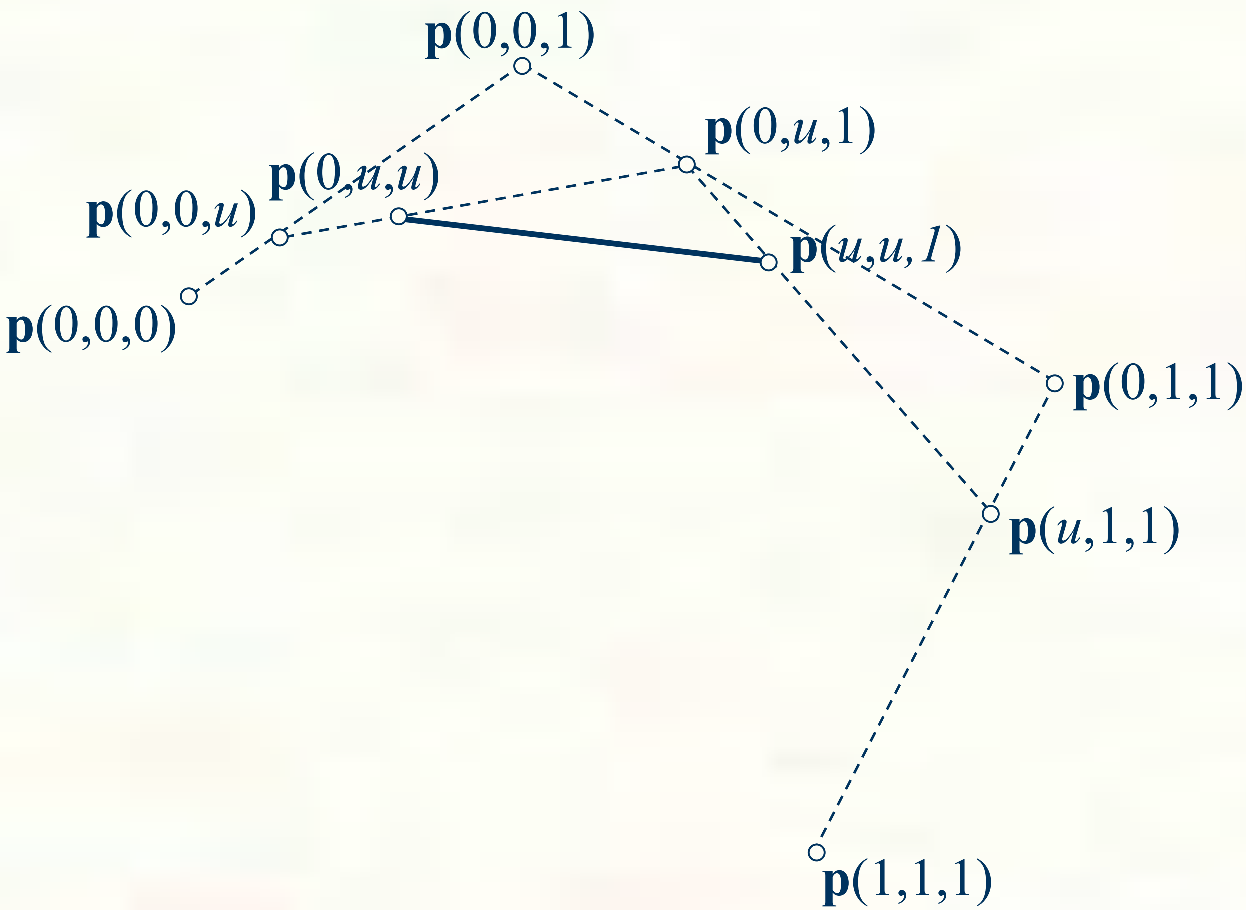
de Casteljau in polar form



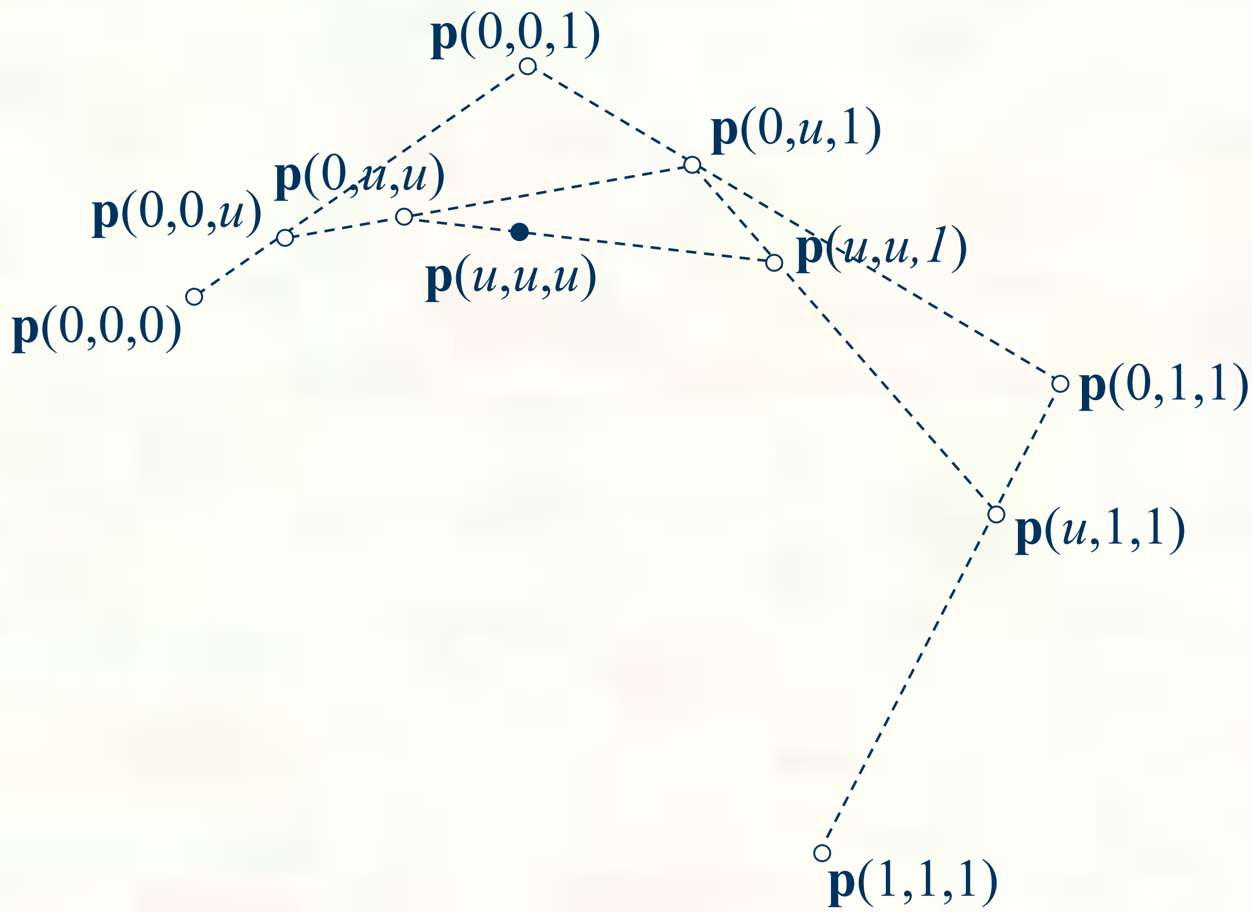
de Casteljau in polar form



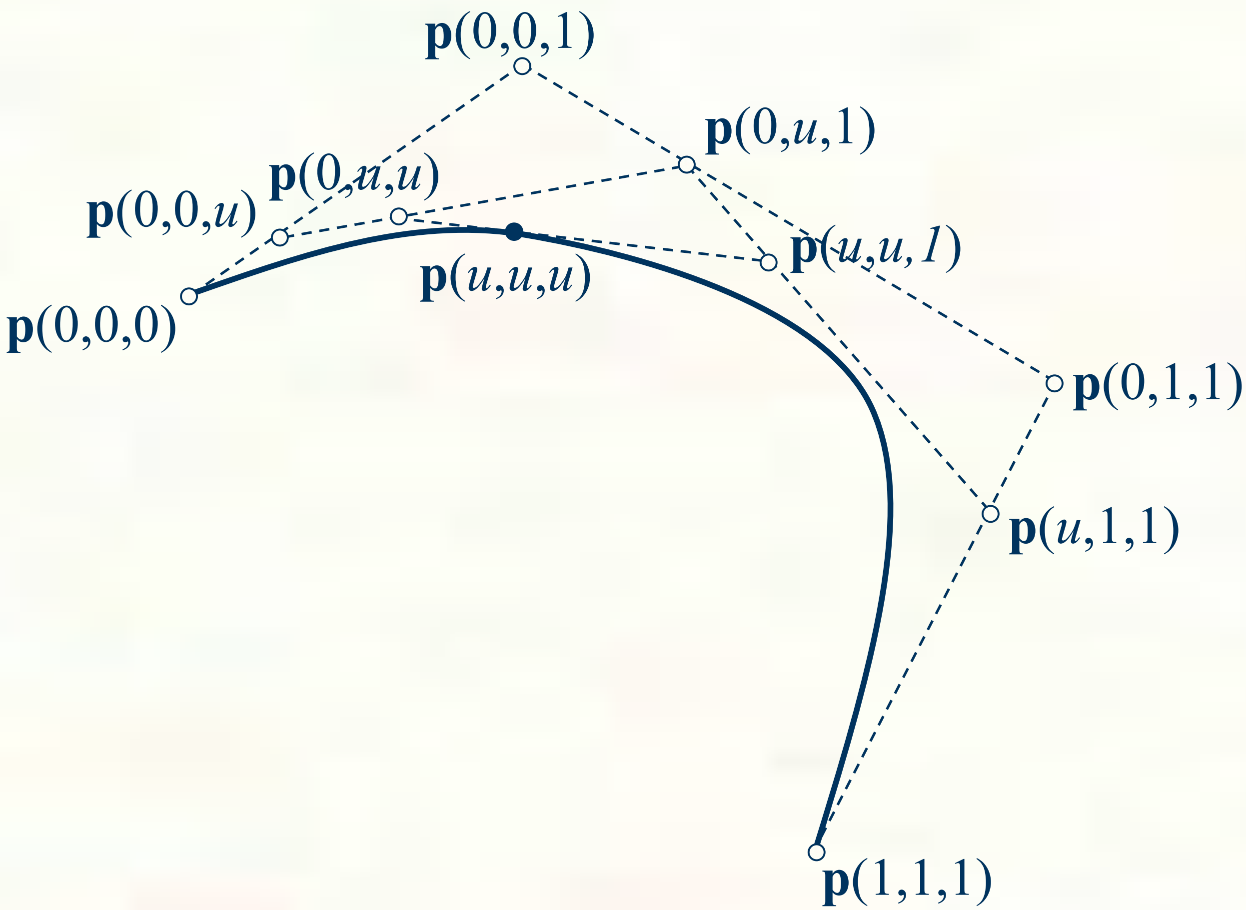
de Casteljau in polar form



de Casteljau in polar form

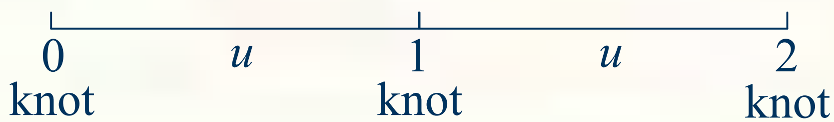


de Casteljau in polar form



Composite curves in polar form

- Suppose we want to glue two cubic Bézier curves together in a way that automatically guarantees C^2 continuity everywhere. We can do this easily in polar form.
- Start with parameter space for the pair of curves
  - 1st curve [0,1], 2nd curve (1,2]

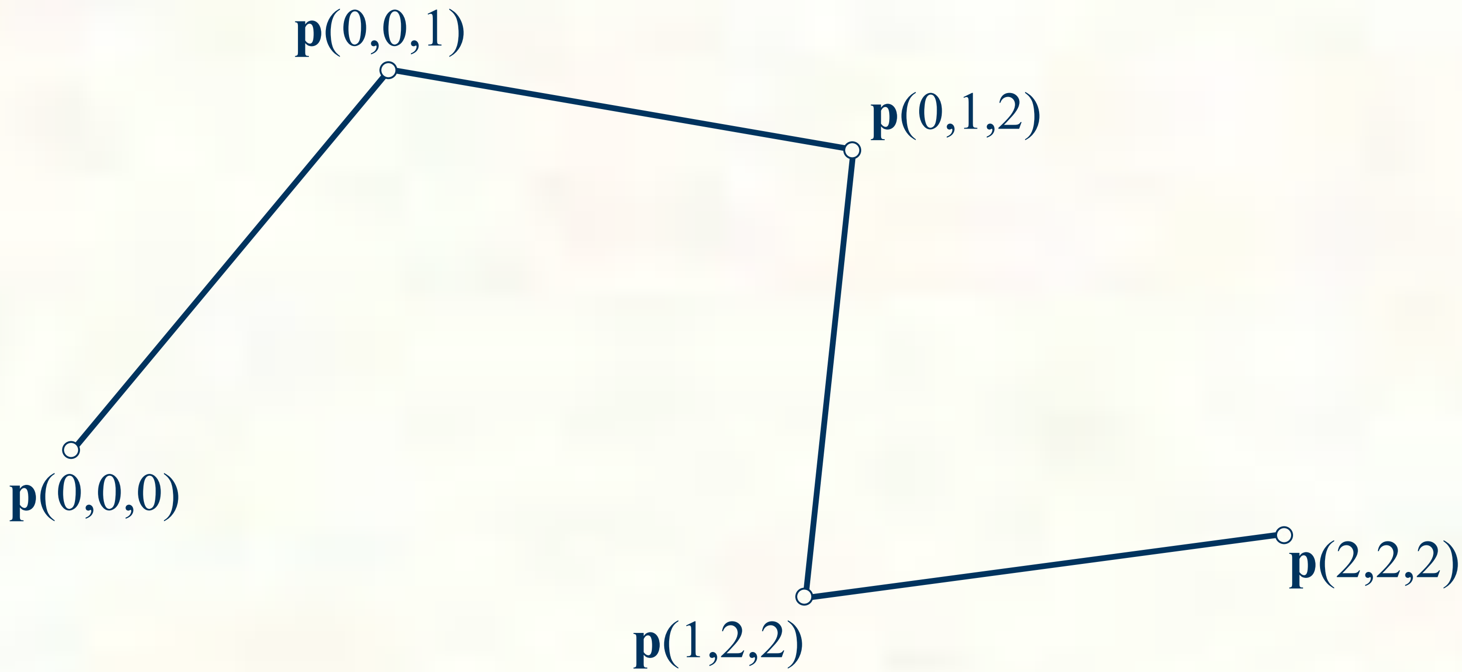


- Make a knot vector: [000,1,222]
- Number control points as before:

$$\mathbf{p}(0,0,0),\ \mathbf{p}(0,0,1),\ \mathbf{p}(0,1,2),\ \mathbf{p}(1,2,2),\ \mathbf{p}(2,2,2)$$

- Okay, 5 control points for the two curves, so 3 of them must be shared since each curve needs 4. That's what having only 1 copy of knot 1 achieves, and that's what gives us C^2 continuity at the join point at $u = 1$

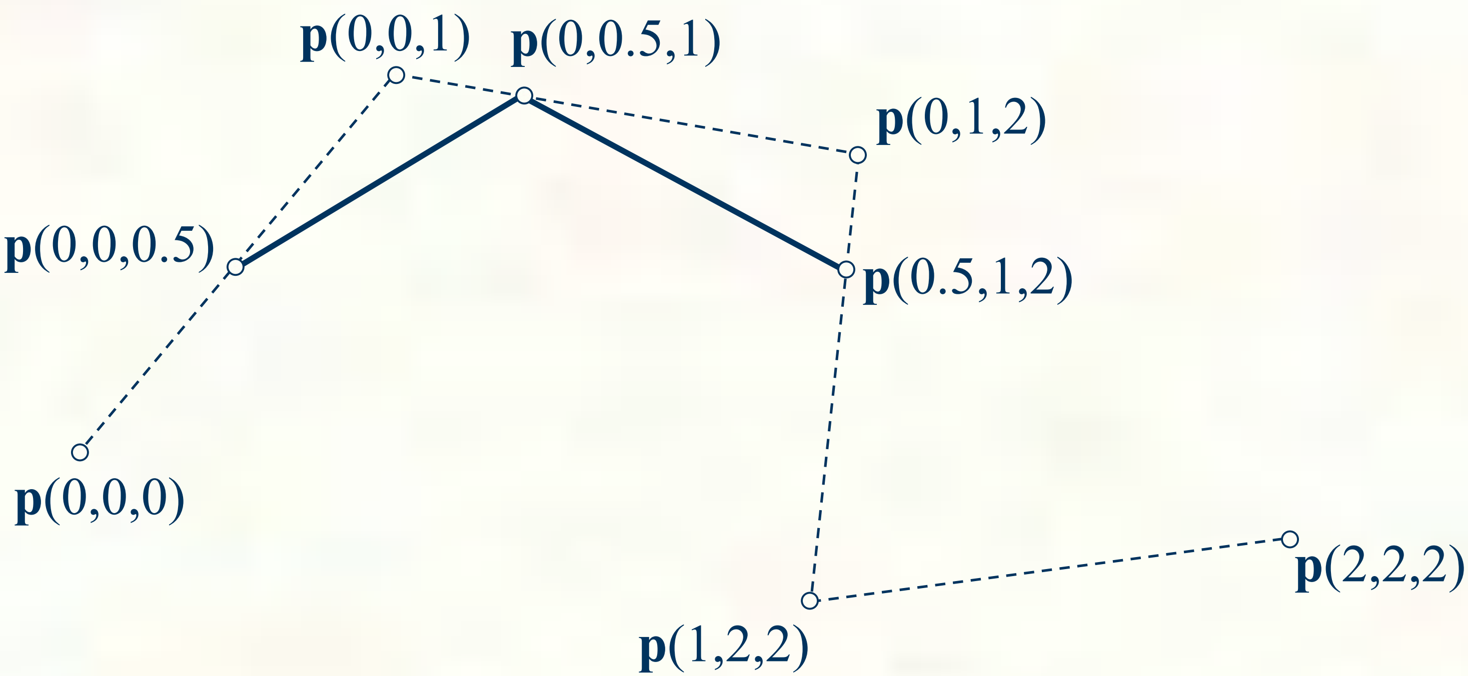
de Boor algorithm in polar form



$u = 0.5$

Knot vector = [0,0,0,1,2,2,2]

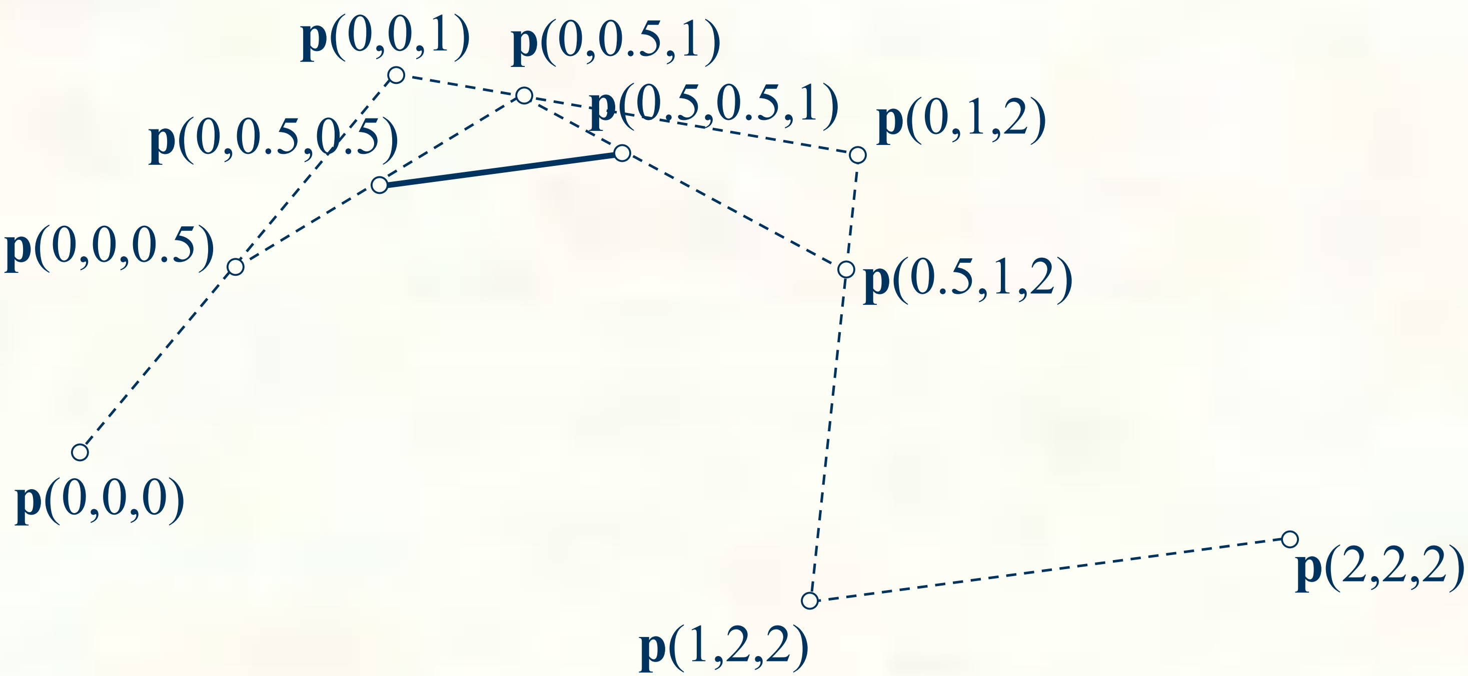
Inserting a knot



$u = 0.5$

Knot vector = [0,0,0,0.5,1,2,2,2]

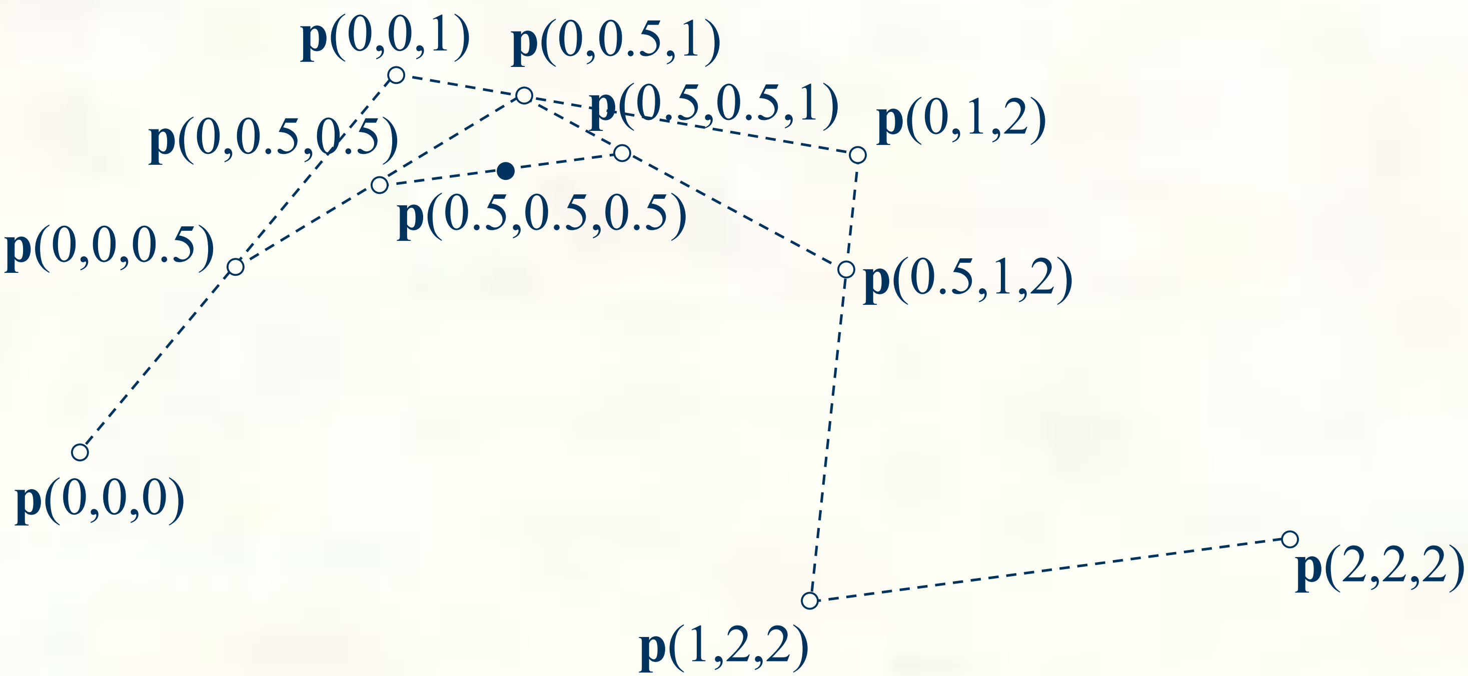
Inserting a 2nd knot



$u = 0.5$

Knot vector = [0,0,0,0.5,0.5,1,2,2,2]

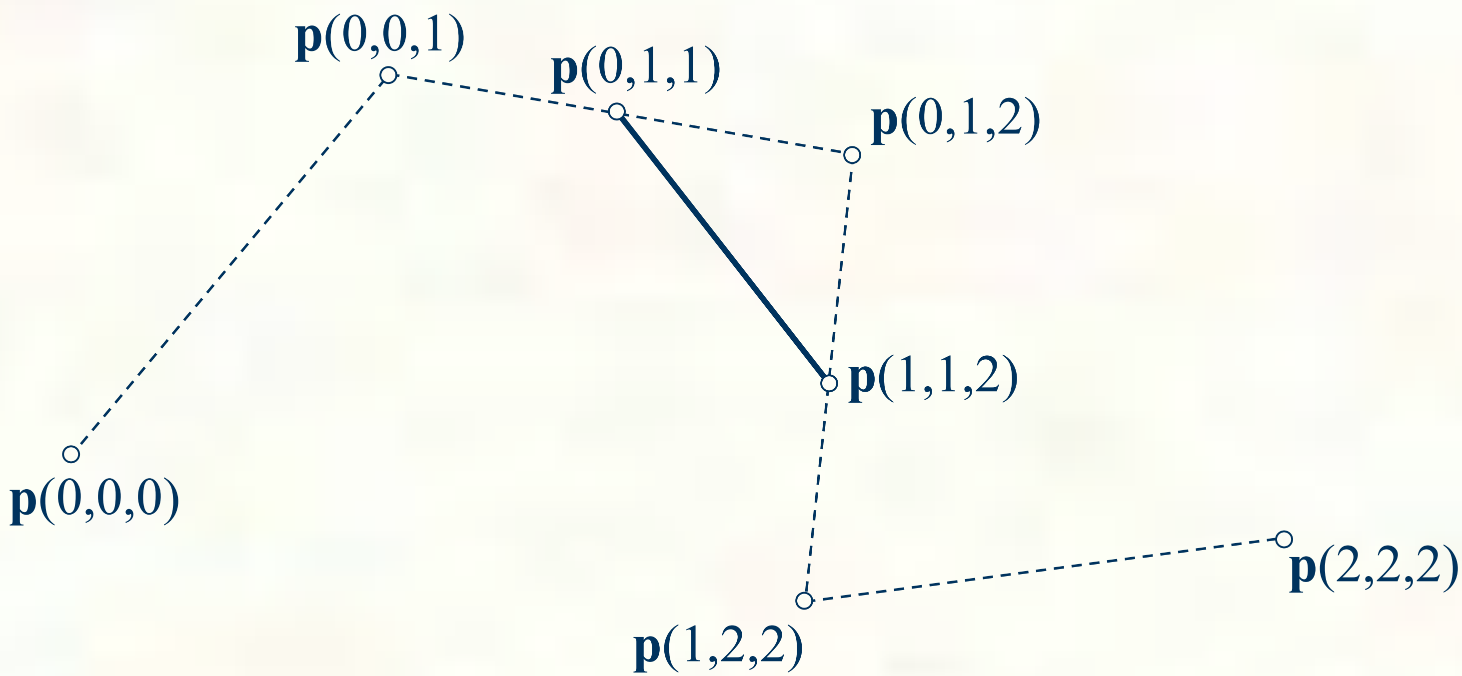
Inserting a 3rd knot to get a point



$u = 0.5$

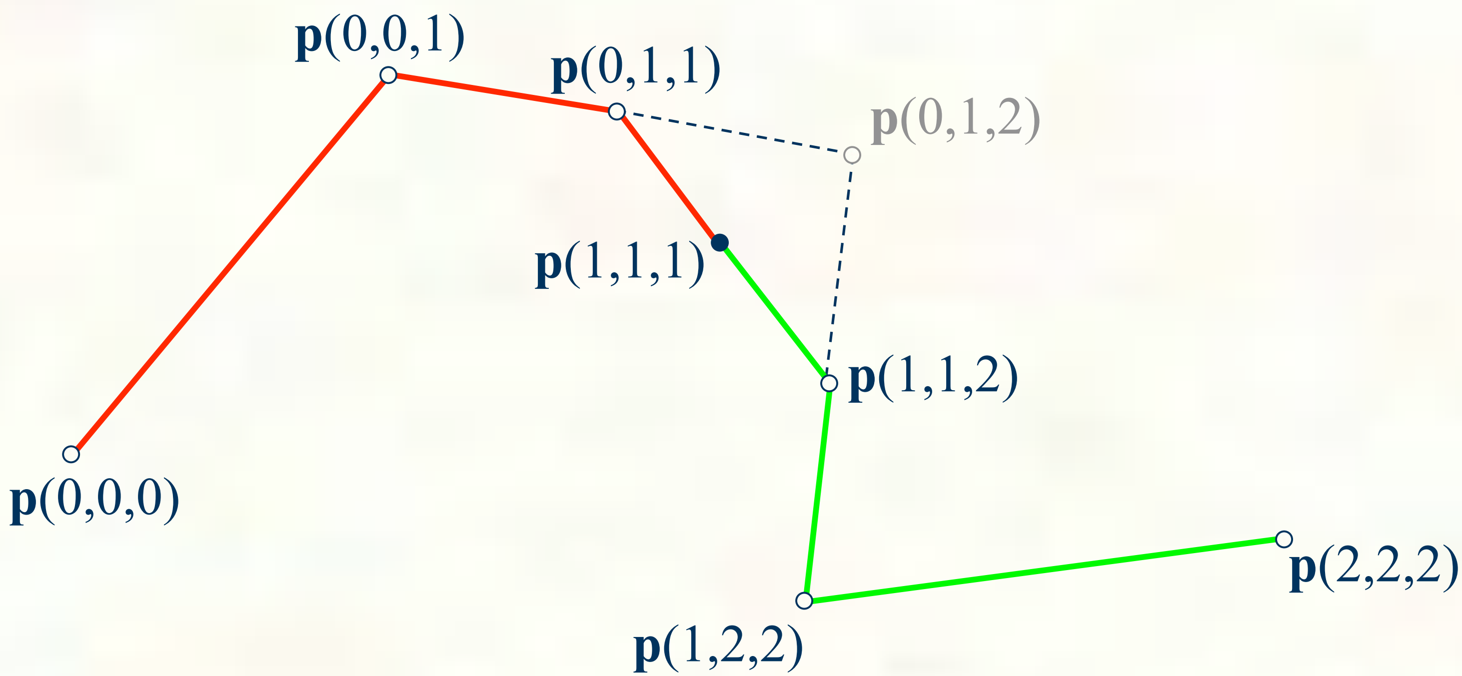
Knot vector = [0,0,0,0.5,0.5,0.5,1,2,2,2]

Recovering the Bézier curves



Knot vector = [0,0,0,1,1,2,2,2]

Recovering the Bézier curves



Knot vector = [0,0,0,1,1,1,2,2,2]

B-Splines

- B-splines are a generalization of Bézier curves that allows grouping them together with continuity across the joints
- The B in B-splines stands for basis, they are based on a very general class of spline basis functions
- Splines is a term referring to composite parametric curves with guaranteed continuity
- The general form is similar to that of Bézier curves

  Given $m + 1$ values u_i in parameter space (these are called **knots**), a degree n B-spline curve is given by:

$$\mathbf{p}(u) = \sum_{i=0}^{m-n-1} \mathrm{N}_{i,n}(u)\mathbf{p}_i$$

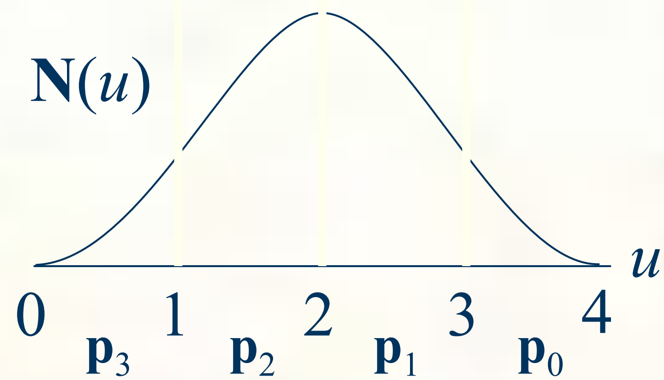
$$\mathrm{N}_{i,0}(u) = \begin{cases} 1 & u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{N}_{i,n}(u) = \frac{u - u_i}{u_{i+n} - u_i}\mathrm{N}_{i,n-1}(u) + \frac{u_{i+n+1} - u}{u_{i+n+1} - u_{i+1}}\mathrm{N}_{i+1,n-1}(u)$$

where $m \ge i + n + 1$

Uniform periodic basis

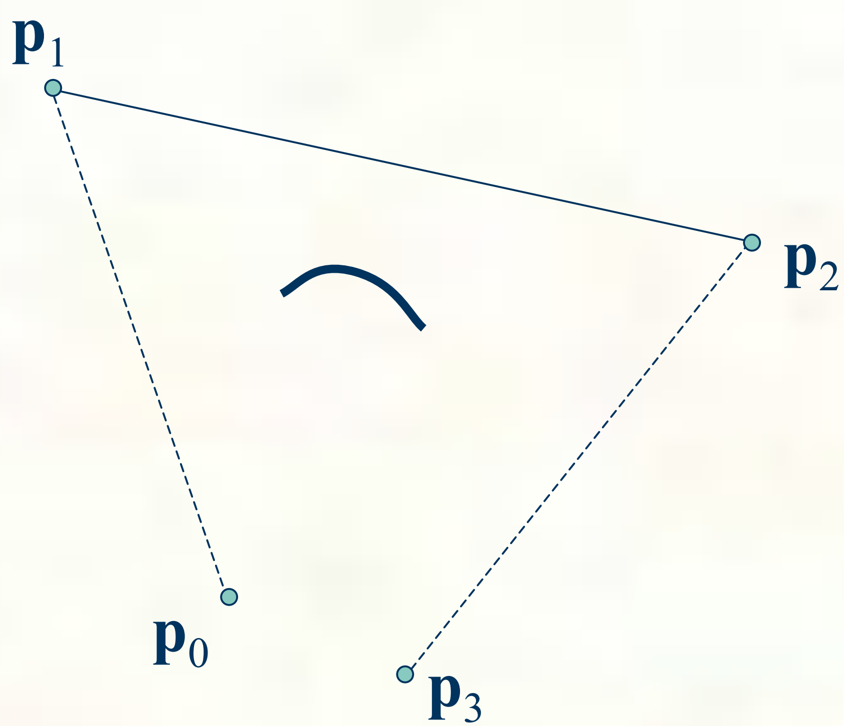
- Let $N(u)$ be a global basis function for our uniform cubic B-splines
- $N(u)$ is piecewise cubic



$$N(u) = \begin{cases} \frac{1}{6}u^3 & = \frac{1}{6}u^3 & \text{if } u < 1 \\ -\frac{1}{2}(u-1)^3 + \frac{1}{2}(u-1)^2 + \frac{1}{2}(u-1) + \frac{1}{6} & = -\frac{1}{2}u^3 + 2u^2 - 2u + \frac{2}{3} & \text{if } u < 2 \\ \frac{1}{2}(u-2)^3 - (u-2)^2 + \frac{2}{3} & = \frac{1}{2}u^3 - 4u^2 + 10u - \frac{22}{3} & \text{if } u < 3 \\ -\frac{1}{6}(u-3)^3 + \frac{1}{2}(u-3)^2 - \frac{1}{2}(u-3) + \frac{1}{6} & = -\frac{1}{6}u^3 + 2u^2 - 8u + \frac{32}{3} & \text{otherwise} \end{cases}$$

$$\mathbf{p}(u) = N(u)\,\mathbf{p}_3 + N(u+1)\,\mathbf{p}_2 + N(u+2)\,\mathbf{p}_1 + N(u+3)\,\mathbf{p}_0$$

Uniform periodic B-Spline



$$\begin{aligned}\mathbf{p}(u) = &(-1/6u^3 + 1/2u^2 - 1/2u + 1/6)\mathbf{p}_0 + \\ &(1/2u^3 - u^2 + 2/3)\mathbf{p}_1 + \\ &(-1/2u^3 + 1/2u^2 + 1/2u + 1/6)\mathbf{p}_2 + \\ &(1/6u^3)\mathbf{p}_3\end{aligned}$$