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)
  \[ p(u) = au + b \]
  \[
  \begin{align*}
  x &= a_x u + b_x \\
  y &= a_y u + b_y \\
  z &= a_z u + b_z
  \end{align*}
  \]

- Parametric cubic curve (in E^3)
  \[ p(u) = au^3 + bu^2 + cu + d \]
  \[
  \begin{align*}
  x &= a_x u^3 + b_x u^2 + c_x u + d_x \\
  y &= a_y u^3 + b_y u^2 + c_y u + d_y \\
  z &= a_z u^3 + b_z u^2 + c_z u + d_z
  \end{align*}
  \]

- Basis (monomial or power)
  \[
  \begin{bmatrix}
  u & 1 \\
  u^3 & u^2 & u & 1
  \end{bmatrix}
  \]
Hermite Curves

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

\[ p(0) = d \]
\[ p(1) = a + b + c + d \]
\[ p''(0) = c \]
\[ p''(1) = 3a + 2b + c \]

- Solving for the coefficients:

\[ a = 2p(0) - 2p(1) + p''(0) + p''(1) \]
\[ b = -3p(0) + 3p(1) - 2p''(0) - p''(1) \]
\[ c = p''(0) \]
\[ d = p(0) \]
Hermite Curves - Hermite Basis

Substituting for the coefficients and collecting terms gives

\[ p(u) = (2u^3 - 3u^2 + 1)p(0) + (-2u^3 + 3u^2)p(1) + (u^3 - 2u^2 + u)p''(0) + (u^3 - u^2)p''(1) \]

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

\[ p(u) = H_1(u)p(0) + H_2(u)p(1) + H_3(u)p''(0) + H_4(u)p''(1) \]
Hermite Curves - Matrix Form

- Putting this in matrix form

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

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

\[
= U M_H
\]

- \( M_H \) 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 \( p(u) \)

\[
\mathbf{B} = \begin{bmatrix}
p(0) \\
p(1) \\
p''(0) \\
p''(1)
\end{bmatrix}
\]

\[
p(u) = U M_H \mathbf{B}
\]
Hermite and Algebraic Forms

- \( \mathbf{M}_H \) transforms geometric coefficients ("coordinates") from the Hermite basis to the algebraic coefficients of the monomial basis.

\[
\begin{align*}
\mathbf{A} &= \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \\
p(u) &= \mathbf{p} = \mathbf{UA} = \mathbf{UM}_H \mathbf{B} \\
\mathbf{A} &= \mathbf{M}_H \mathbf{B} \\
\mathbf{B} &= \mathbf{M}_H^{-1} \mathbf{A}
\end{align*}
\]

\[
\mathbf{M}_H^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}
\]
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 \( p_0, p_1, p_2, \) and \( p_3 \).
- 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:

\[
\begin{align*}
p(0) &= p_0 \\
p(1) &= p_3 \\
p^u(0) &= k_1(p_1 - p_0) \\
p^u(1) &= k_2(p_3 - p_2)
\end{align*}
\]
Substituting this into the Hermite curve expression and rearranging, we get

\[ p(u) = [(2 - k_1)u^3 + (2k_1 - 3)u^2 - k_1u + 1]p_0 + [k_1u^3 - 2k_1u^2 + k_1u]p_1 \\
+ [-k_2u^3 + k_2u^2]p_2 + [(k_2 - 2)u^3 + (3 - k_2)u^2]p_3 \]

In matrix form, this is

\[ p(u) = UM_B P \quad M_B = \begin{bmatrix} 2 - k_1 & k_1 & -k_2 & k_2 - 2 \\
2k_1 - 3 & -2k_1 & k_2 & 3 - k_2 \\
-k_1 & k_1 & 0 & 0 \\
1 & 0 & 0 & 0 \end{bmatrix} \quad P = \begin{bmatrix} p_0 \\
p_1 \\
p_2 \\
p_3 \end{bmatrix} \]
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 \( p_0 \) is at \( u = 0 \), \( p_1 \) at \( u = 1/3 \), \( p_2 \) at \( u = 2/3 \) and \( p_3 \) at \( u = 1 \). Then

\[
p''(0) = (p_1 - p_0)/(1/3 - 0) = 3(p_1 - p_0)
\]
\[
p''(1) = (p_3 - p_2)/(1 - 2/3) = 3(p_3 - p_2)
\]

and \( k_1 = k_2 = 3 \), giving a nice symmetric characteristic matrix:

\[
M_B = \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 3 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\]

So

\[
p(u) = (-u^3 + 3u^2 - 3u + 1)p_0 + (3u^3 - 6u^2 + 3u)p_1 + (-3u^3 + 3u^2)p_2 + u^3p_3
\]
General Bézier Curves

- This can be rewritten as

\[ p(u) = (1 - u)^3 p_0 + 3u(1 - u)^2 p_1 + 3u^2(1 - u)p_2 + u^3 p_3 = \sum_{i=0}^{3} \binom{3}{i} u^i (1 - u)^{3-i} p_i \]

- Note that the binomial expansion of

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

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

\[ p(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1 - u)^{n-i} p_i \]
General Bézier Curves

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

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

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

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

- Of particular interest to us (in addition to cubic curves):
  - Linear: $p(u) = (1 - u)p_0 + up_1$
  - Quadratic: $p(u) = (1 - u)^2p_0 + 2u(1 - u)p_1 + u^2p_2$
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

![Diagram of Bézier curve properties](image)
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 $A = M_B P$ once control points are known, then evaluate $p(u_i) = [u_i^3 u_i^2 u_i 1]A$, $i = 0,1,2,\ldots,n$ for $n$ fixed increments of $u$
- For better numerical stability, take e.g. a quadratic curve (for simplicity) and rewrite

$$p(u) = (1 - u)^2 p_0 + 2u(1 - u)p_1 + u^2 p_2$$

$$= (1 - u)[(1 - u)p_0 + up_1] + u[(1 - u)p_1 + up_2]$$

- 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

\[ p_{i,j} = (1-u)p_{i,j-1} + up_{i+1,j-1} \]

\[
\begin{align*}
    i &= 0,1,2,\ldots,n-j \\
    j &= 1,2,\ldots,n
\end{align*}
\]

where \( p_{i,0} \quad i = 0,1,2,\ldots,n \) are the control points for a degree \( n \) Bézier curve and \( p_{0,n} = p(u) \)

- 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.
Starting with the control points
and a given value of \( u \)
In this example, \( u \approx 0.25 \)
de Casteljau Algorithm

\[ q_0(u) = (1 - u)p_0 + up_1 \]
\[ q_1(u) = (1 - u)p_1 + up_2 \]
\[ q_2(u) = (1 - u)p_2 + up_3 \]
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

\[ p(u) = (1 - u)r_0(u) + ur_1(u) \]
de Casteljau algorithm
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 $p_i = p(u_i)$ where $u_i = \frac{i}{n}$, $i = 0, 1, \ldots, n$

- For an arbitrary cubic curve
  \[ p_i = a \left( \frac{i^3}{n^3} \right) + b \left( \frac{i^2}{n^2} \right) + c \left( \frac{i}{n} \right) + d \]

  - 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 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 \textbf{knots}, “interesting” values in parameter space
- For Bézier curves, parameter space is normally $[0, 1]$, and the knots are at 0 and 1.

$\begin{align*}
0 & \quad \text{Knot} \\
u & \quad \text{Parameter Space} \\
1 & \quad \text{Knot}
\end{align*}$

- Now build a \textbf{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 $[0,0,1,1]$

\textbf{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: $p_0 = p(0,0,0), p_1 = p(0,0,1), p_2 = p(0,1,1), p_3 = p(1,1,1)$
- Quadratic Bézier control points: $p_0 = p(0,0), p_1 = p(0,1), p_2 = p(1,1)$
Polar form rules

- Polar values are symmetric in their arguments, i.e. all permutations of a polar label are equivalent.
  \[ p(0,0,1) = p(0,1,0) = p(1,0,0), \text{ etc.} \]

- Given \( p(u_1, u_2, \ldots, u_{n-1}, a) \) and \( p(u_1, u_2, \ldots, u_{n-1}, b) \), for any value \( c \) we can compute

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

  That is, \( p(u_1, u_2, \ldots, u_{n-1}, c) \) is an affine combination of

  \( p(u_1, u_2, \ldots, u_{n-1}, a) \) and \( p(u_1, u_2, \ldots, u_{n-1}, b) \).

Examples:

\[
\begin{align*}
p(0, u_1) &= (1 - u)p(0, 0, 1) + up(0, 1, 1) \\
p(0, u) &= \frac{(4 - u)p(0, 2) + (u - 2)p(0, 4)}{2} \\
p(1, 2, 3, u) &= \frac{(u_2 - u)p(2, 1, 3, u_1) + (u - u_1)p(3, 2, 1, u_2)}{u_2 - u_1}
\end{align*}
\]
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

\[ p(0,0,0) \]
\[ p(0,0,u) \]
\[ p(0,0,1) \]
\[ p(u,u,u) \]
\[ p(u,1,1) \]
\[ p(0,1,1) \]
\[ p(1,1,1) \]
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:
  - $p(0,0,0)$, $p(0,0,1)$, $p(0,1,2)$, $p(1,2,2)$, $p(2,2,2)$

- 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

\[ p(0,0,0) \quad p(0,0,1) \quad p(1,2,2) \quad p(0,1,2) \quad p(2,2,2) \]

\[ u = 0.5 \]

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

\[
(p(0,0,0), p(0,0,1), p(0,0.5,0.5), p(0,0,0.5), p(0,0,0.5), p(0.5,1,2), p(0,0.5,1), p(0,0.5,1), p(0.5,0.5,1), p(0.5,0.5,1))
\]

\[
u = 0.5
\]

Knot vector = \([0,0,0,0.5,0.5,1,2,2,2]\)
Inserting a 3rd knot to get a point

\[ p(0,0,0) \quad p(0,0,1) \quad p(0,1,2) \quad p(1,2,2) \]

\[ u = 0.5 \]

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:

\[
p(u) = \sum_{i=0}^{m-n-1} N_{i,n}(u)p_i
\]

\[
N_{i,0}(u) = \begin{cases} 
1 & u_i \leq u < u_{i+1} \\
0 & \text{otherwise}
\end{cases}
\]

\[
N_{i,n}(u) = \frac{u - u_i}{u_{i+n} - u_i}N_{i,n-1}(u) + \frac{u_{i+n+1} - u}{u_{i+n+1} - u_{i+1}}N_{i+1,n-1}(u)
\]

where \( m \geq i + n + 1 \)
Uniform periodic basis

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

$$N(u) = \begin{cases} 
\frac{1}{6}u^3 & \text{if } u < 1 \\
-\frac{1}{2}(u-1)^3 + \frac{1}{2}(u-1)^2 + \frac{1}{2}(u-1) + \frac{1}{6} & \text{if } u < 2 \\
\frac{1}{2}(u-2)^3 - (u-2)^2 + \frac{2}{3} & \text{if } u < 3 \\
-\frac{1}{6}(u-3)^3 + \frac{1}{2}(u-3)^2 - \frac{1}{2}(u-3) + \frac{1}{6} & \text{otherwise}
\end{cases}$$
Basis over \([0,1]\)

- Pieces of single basis function associated with 4 overlapping copies for active control points

\[
N(u) = \begin{cases} 
\frac{1}{6}u^3 \\
-\frac{1}{2}u^3 + \frac{1}{2}u^2 + \frac{1}{2}u + \frac{1}{6} \\
\frac{1}{2}u^3 - u^2 + \frac{2}{3} \\
-\frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6} 
\end{cases}
\]

\[p(u) = N_0(u) \ p_3 + N_1(u) \ p_2 + N_2(u) \ p_1 + N_3(u)p_0\]
Uniform periodic B-Spline

\[ p(u) = \left( -\frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6} \right)p_0 + \right) + \left( \frac{1}{2}u^3 - u^2 + \frac{2}{3} \right)p_1 + \left( -\frac{1}{2}u^3 + \frac{1}{2}u^2 + \frac{1}{2}u + \frac{1}{6} \right)p_2 + \left( \frac{1}{6}u^3 \right)p_3 \]

\[ p(u) = \frac{1}{6} \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \]
Composite B-Spline
Uniform periodic B-Spline

\[ p(t,1) = \frac{3-t}{3} p(0,1) + \frac{1}{3} p(1,1) \]
\[ p(t,2) = \frac{3-t}{2} p(1,1) + \frac{1}{2} p(2,1) \]
\[ p(t,3) = \frac{3-t}{1} p(2,1) + \frac{1}{1} p(3,1) \]

\[ p(t,1,2) = \frac{3-t}{3} p(1,2) + \frac{1}{3} p(2,2) \]
\[ p(t,2,3) = \frac{3-t}{2} p(2,2) + \frac{1}{2} p(3,2) \]
\[ p(t,3,4) = \frac{3-t}{1} p(3,2) + \frac{1}{1} p(4,2) \]

\[ p(t,t,1) = \frac{3-t}{3} p(1,1,2) + \frac{1}{3} p(2,1,2) \]
\[ p(t,t,2) = \frac{3-t}{2} p(2,1,2) + \frac{1}{2} p(3,1,2) \]
\[ p(t,t,3) = \frac{3-t}{1} p(3,1,2) + \frac{1}{1} p(4,1,2) \]
Example:

General Case

$p(0,1,2)$  $p(1,2,3)$  $p(2,3,4)$  $p(3,4,5)$  $p(4,5,6)$  $p(5,6,7)$  $p(6,7,8)$  $p(7,8,9)$  $p(8,9,10)$  $p(9,10,11)$
Composite B-Spline

Example:

\[ p(0,1,2) \]
\[ p(1,2,3) \]
\[ p(2,3,4) \]
\[ p(3,4,5) \]
\[ p(4,5,6) \]
\[ p(5,6,7) \]
\[ p(6,7,8) \]
\[ p(7,8,9) \]
\[ p(8,9,10) \]
\[ p(9,10,11) \]
Composite B-Spline

Example:

\[ p(0,1,2) \]
\[ p(1,2,3) \]
\[ p(2,3,4) \]
\[ p(3,4,5) \]
\[ p(4,5,6) \]
\[ p(5,6,7) \]
\[ p(6,7,8) \]
\[ p(7,8,9) \]
\[ p(8,9,10) \]
\[ p(9,10,11) \]
Composite B-Spline

Example:

\[ p(0, 1, 2) \]
\[ p(1, 2, 3) \]
\[ p(2, 3, 4) \]
\[ p(3, 4, 5) \]
\[ p(4, 5, 6) \]
\[ p(5, 6, 7) \]
\[ p(6, 7, 8) \]
\[ p(7, 8, 9) \]
\[ p(8, 9, 10) \]
\[ p(9, 10, 11) \]