11. Parametric curves

Reading

Recommended:

- Bartels, Beatty, and Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*, 1987.
- Farin, *Curves and Surfaces for Computer Aided Geometric Design*, 1997.

Outline

- Basics:
  - Introduction to mathematical splines
  - Bézier curves
  - Continuity conditions (C^0, C^1, C^2, C^3)
- Splines:
  - C^2 interpolating splines
  - B-splines
  - Catmull-Rom splines

Introduction

Mathematical splines are motivated by the "loftsman's spline":

- Long, narrow strip of wood or plastic
- Used to fit curves through specified data points
- Shaped by lead weights called "ducks"
- Gives curves that are "smooth" or "fair"

Such splines have been used for designing:

- Automobiles
- Ship hulls
- Aircraft fuselages and wings

Mathematical splines

The mathematical splines we'll use are:

- Piecewise
- Parametric
- Polynomials

We've seen the term parametric before. Let's look at the other two terms. . . .

Parametric polynomial curves

A parametric "polynomial" curve is a parametric curve where each function $x(t)$, $y(t)$ is described by a polynomial:

$$
\begin{aligned}
x(t) &= \sum_{i=0}^{n} a_i \, t^i \\
y(t) &= \sum_{i=0}^{n} b_i \, t^i
\end{aligned}
$$

Polynomial curves have certain advantages:

- Easy to compute
- Infinitely differentiable

Piecewise parametric polynomial curves

A "piecewise" parametric polynomial curve uses different polynomial functions for different parts of the curve.

- **Advantage:** Provides flexibility.
- **Problem:** How do you guarantee smoothness (continuity) at the joints?

In the rest of the lecture, we'll look at:

1. Bézier curves — general class of polynomial curves
2. Splines — ways of putting these curves together

Bézier curves

- Developed simultaneously by Bézier (at Renault) and de Casteljau (at Citroen), circa 1960.
- The Bézier curve $Q(u)$ is defined by nested interpolation:

- V_i's are "control points"
- $\{V_0, \ldots, V_n\}$ is the "control polygon"

Bézier curves: Basic properties

Bézier curves enjoy some nice properties:

- Endpoint interpolation:

$$
\begin{aligned}
Q(0) &= V_0 \\
Q(1) &= V_n
\end{aligned}
$$

- Convex hull: The curve is contained in the "convex hull" of its control polygon
- Symmetry:

$$
\begin{aligned}
&Q(u) \text{ defined by } \{V_0, \ldots, V_n\} \\
&\qquad \equiv Q(1-u) \text{ defined by } \{V_n, \ldots, V_0\}
\end{aligned}
$$

Bézier curves: Explicit formulation

Let's give V_i a superscript V_i^j to indicate the level of nesting.

An explicit formulation for $Q(u)$ is given by the recurrence:

$$V_i^j \;=\; (1-u)\, V_i^{j-1} \;+\; u\, V_{i+1}^{j-1}$$

Explicit formulation, cont.

For $n = 2$, we have:

$$\begin{aligned} Q(u) &= V_0^2 \\ &= (1-u)\, V_0^1 \;+\; u\, V_1^1 \\ &= (1-u)\, \left[(1-u)\, V_0^0 \;+\; u\, V_1^0\right] \;+\; u\, \left[(1-u)\, V_1^0 \;+\; u\, V_2^0\right] \\ &= (1-u)^2\, V_0^0 \;+\; 2u\,(1-u)\, V_1^0 \;+\; u^2\, V_2^0 \end{aligned}$$

In general:

$$Q(u) \;=\; \sum_{i=0}^{n} V_i \underbrace{\binom{n}{i}\, u^i\,(1-u)^{n-i}}_{B_i^n(u)}$$

$B_i^n(u)$ is the i'th Bernstein polynomial of degree n.

Bézier curves: More properties

Here are some more properties of Bézier curves

$$Q(u) \;=\; \sum_{i=0}^{n} V_i \binom{n}{i}\, u^i\,(1-u)^{n-i}$$

- Degree: $Q(u)$ is a polynomial of degree n
- Control points: A Bézier curve of degree n requires $n+1$ control points. Why?

More properties, cont.

- Tangents:

$$\begin{aligned} Q'(0) &= n(V_1 - V_0) \\ Q'(1) &= n(V_n - V_{n-1}) \end{aligned}$$

- k'th derivatives: In general,
  - $Q^{(k)}(0)$ depends only on $V_0, \ldots, V_k$
  - $Q^{(k)}(1)$ depends only on $V_n, \ldots, V_{n-k}$
  - (At intermediate points $u \in (0,1)$, *all* control points are involved for every derivative.)

Cubic curves

For the rest of this discussion, we'll restrict ourselves to piecewise cubic curves.

- In CAGD, higher-order curves are often used
  - Gives more freedom in design
  - Can provide higher degree of continuity between pieces
- For graphics, piecewise cubic let's you do just about anything
  - Lowest degree for specifying points to interpolate and tangents
  - Lowest degree for specifying curve in space

All the ideas here generalize to higher-order curves.

Matrix form of Bézier curves

Bézier curves can also be described in matrix form:

$$
\begin{aligned}
Q(u) &= \sum_{i=0}^{3} V_i \binom{3}{i} u^i (1-u)^{3-i} \\
&= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u) V_2 + u^3 V_3 \\
&= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} \\
&= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M_{\text{Bézier}} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix}
\end{aligned}
$$

Display: Recursive subdivision

Q: Suppose you wanted to draw one of these Bézier curves — how would you do it?

A: Recursive subdivision:

Display, cont.

Here's pseudocode for the recursive subdivision display algorithm:

procedure *Display*($\{V_0, \ldots, V_n\}$):
  if $\{V_0, \ldots, V_n\}$ flat within ϵ **then**
    Output line segment $V_0 V_n$
  else
    Subdivide to produce $\{L_0, \ldots, L_n\}$ and $\{R_0, \ldots, R_n\}$
    Display($\{L_0, \ldots, L_n\}$)
    Display($\{R_0, \ldots, R_n\}$)
  end if
end procedure

Positional (C^0) continuity

To build up more complex curves, we can piece together different Bézier curves to make "splines."

Q: What condition ensures positional continuity?

Derivative (C^1) continuity

Q: What condition ensures derivative continuity?

Q: How might you build an interactive system to satisfy these constraints?

Curvature (C^2) continuity

Q: Suppose you want even higher degrees of continuity — e.g., not just slopes but curvatures — what additional geometric constraints are imposed?

C^3 continuity

Summary of continuity conditions

- C^0 straightforward, but generally not enough
- C^3 is too constrained (with cubics)

Creating continuous splines

We'll look at three ways to specify splines with C^1 and C^2 continuity:

1. C^2 interpolating splines
2. B-splines
3. Catmull-Rom splines

C^2 Interpolating splines

Problem: Describe an interactive system for specifying C^2 interpolating splines.

Solution:

1. Let user specify first four Bézier control points.
2. This constrains next ________ control points — draw these in.
3. User then picks ________ more.
4. Repeat steps 2–3.

The control points specified by the user, called "joints," are interpolated by the spline.

A more in-depth analysis

Recall that for each of x and y, we needed to specify ________ conditions for each cubic Bézier segment.

So if there are m segments, we'll need ________ constraints.

Q: How many of these constraints are determined by each joint?

In-depth analysis, cont.

At each interior joint j, we have:

1. Last curve ends at j
2. Next curve begins at j
3. Tangents of two curves at j are equal
4. Curvature of two curves at j are equal

The m segments give:

- ________ interior joints
- ________ conditions

The 2 end joints give 2 further constraints:

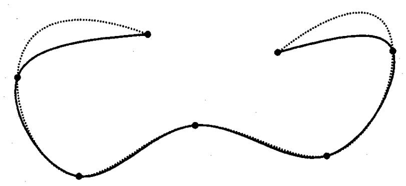
1. First curve begins at first joint
2. Last curve ends at last joint

Gives ________ constraints altogether.

End conditions

The analysis shows that specifying $m + 1$ joints for m segments leaves 2 extra degrees of freedom, which could be specified in a variety of ways:

- Our interactive system
  - Constraints specified as ________
- "Natural" cubic splines
  - Second derivatives at endpoints defined to be 0
- Maximal continuity
  - Require C^3 continuity between first and last pairs of curves



Natural splines vs. maximal continuity (Bartels et al., 3.4)

Global vs. local control

These C^2 interpolating splines yield only "global control" — moving any one control point may change the entire curve!

Global control is problematic:

- Makes splines difficult to design
- Makes incremental display inefficient

There's a fix, but nothing comes for free. Two choices:

- B-splines
  - Keep C^2 continuity
  - Give up interpolation
- Catmull-Rom splines
  - Keep interpolation
  - Give up C^2 continuity — provides C^1 only

B-splines

Previous construction (C^2 interpolating splines):

- Choose Bézier control points, constrained by the "A-frames."

New construction (B-splines):

- Choose points on A-frames
- Let these determine the Bézier control points

The B-splines I'll describe are known more precisely as "uniform B-splines."

B-spline construction

The control points in this construction are called "de Boor points."

B-spline properties

Here are some properties of B-splines:

- C^2 continuity
- Approximating
  - · Does not interpolate control points
- Locality
  - · Each segment determined by 4 control points
  - · Each control point determines 4 segments
- Convex hull
  - · Curve lies inside convex hull of control points

Algebraic construction of B-splines

$$
\begin{aligned}
V_1 &= \underline{\hspace{2cm}} B_1 + \underline{\hspace{2cm}} B_2 \\
V_2 &= \underline{\hspace{2cm}} B_1 + \underline{\hspace{2cm}} B_2 \\
V_0 &= \underline{\hspace{2cm}}\ [\underline{\hspace{2cm}} B_0 + \underline{\hspace{2cm}} B_1] \\
&\quad + \underline{\hspace{2cm}}\ [\underline{\hspace{2cm}} B_1 + \underline{\hspace{2cm}} B_2] \\
&= \underline{\hspace{2cm}} B_0 + \underline{\hspace{2cm}} B_1 + \underline{\hspace{2cm}} B_2 \\
V_3 &= \underline{\hspace{2cm}} B_1 + \underline{\hspace{2cm}} B_2 + \underline{\hspace{2cm}} B_3
\end{aligned}
$$

Algebraic construction of B-splines, cont.

Once again, this construction can be expressed in terms of a matrix:

$$
\begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}
$$

Drawing B-splines

Drawing B-splines is therefore quite simple:

procedure *Draw-B-Spline*($\{B_0, \ldots, B_n\}$):
  for $i \leftarrow 0$ **to** $n - 3$ **do**
    Convert $B_i, \ldots, B_{i+3}$ into a
      Bezier control polygon $V_0, \ldots, V_3$
    Display($\{V_0, \ldots, V_3\}$)
  end for
end procedure

Catmull-Rom splines

The Catmull-Rom splines

- Give up C^2 continuity
- Keep interpolation

For the derivation, let's go back to the interpolation algorithm. We had 4 conditions at each joint j:

1. Last curve ends at j
2. Next curve begins at j
3. Tangents of two curves at j are equal
4. Curvature of two curves at j are equal

If we. . .

- Eliminate condition 4
- Make condition 3 depend only on local control points

. . .then we can have local control!

Derivation of Catmull-Rom splines

Idea: (same as B-splines)

- Start with control points to interpolate
- Build a cubic Bézier curve between successive points

The endpoints of the cubic Bézier are obvious:

$$V_0 = B_1$$
$$V_3 = B_2$$

Q: What should we do for the other two points?

Derivation of Catmull-Rom, cont.

A: Catmull & Rom set the deriviative at the control points to *half* the the vector between neighboring control points:

Many other choices work — for example, using an arbitrary constant τ times this vector gives a "tension" control.

The Catmull-Rom splines also admit a matrix formulation:

$$\begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 & 6 & 0 & 0 \\ -1 & 6 & 1 & 0 \\ 0 & 1 & 6 & -1 \\ 0 & 0 & 6 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

Exercise: Derive this matrix.

Properties

Here are some properties of Catmull-Rom splines:

- C^1 continuity
- Interpolating
- Locality
- Convex hull property?