Lecture 15

Splines and Bézier Curves

David Semeraro

University of Illinois at Urbana-Champaign

October 17, 2013

degree 1 spline

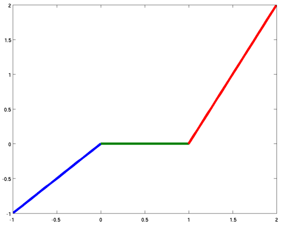
definition

A function $S(x)$ is a spline of degree 1 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is linear on each subinterval $[t_i, t_{i+1}]$.

Example

$$S(x) = \begin{cases} x & x \in [-1, 0] \\ 1 & x \in (0, 1) \\ 2x - 2 & x \in [1, 2] \end{cases}$$



degree 1 spline

Given data $t_0, \ldots, t_n$ and $y_0, \ldots, y_n$, how do we form a spline?
We need two things to hold in the interval $[a, b] = [t_0, t_n]$:

1. $S(t_i) = y_i$ for $i = 0, \ldots, n$
2. $S_i(x) = a_i x + b_i$ for $i = 0, \ldots, n$

Write $S_i(x)$ in point-slope form

$$\begin{aligned} S_i(x) &= y_i + m_i(x - t_i) \\ &= y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i}(x - t_i) \end{aligned}$$

Done.

degree 1 spline

```
input t,y vectors of data
input evaluation location x
find interval i with x ∈ [t_i, t_{i+1}]
S = y_i + (x-t_i) m_i
```

degree 1 spline

Interesting:

- input $n+1$ data points $t_0, \ldots, t_n, y_0, \ldots, y_n$
- in each interval we have $S_i(x) = a_i x + b_i$
- 2 unknowns per interval $[t_i, t_{i+1}]$
- or $2n$ total unknowns
- the $n+1$ pieces of input constraints $S(t_i) = y_i$ gives 2 constraints per interval
- or $2n$ total constraints

degree 2 splines

definition

A function $S(x)$ is a spline of degree 2 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. $S'(x)$ is continuous on $[a, b]$
4. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is quadratic on each subinterval $[t_i, t_{i+1}]$.

degree 2 splines

$$S(x) = \begin{cases} S_0(x) & x \in [t_0, t_1] \\ S_1(x) & x \in [t_1, t_2] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [t_{n-1}, t_n] \end{cases}$$

for each i we have

$$S_i(x) = a_i x^2 + b_i x + c_i$$

What are a_i, b_i, c_i?

degree 2 splines

- 3 unknowns in each interval
- $3n$ total unknowns
- $2n$ constraints for matching up the input data (2 per interval)
- $n-1$ interior points require continuity of the derivative: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$
- but this is just $n-1$ constraints
- total of $3n-1$ constraints
- extra constraint: $S'(x_0)=$given, for example.

degree 3 spline: cubic spline

definition

A function $S(x)$ is a spline of degree 3 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. $S'(x)$ is continuous on $[a, b]$
4. $S''(x)$ is continuous on $[a, b]$
5. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is cubic on each subinterval $[t_i, t_{i+1}]$.

degree 3 spline: cubic spline

In each interval $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by $S(t_i)$, $S(t_{i+1})$ specified (continuity of S)
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$
- $4n-2$ total constraints

This leaves 2 extra degrees of freedom. The cubic spline is not yet unique!

degree 3 spline: cubic spline

Some options:

- natural cubic spline: $S''(t_0) = S''(t_n) = 0$
- fixed-slope: $S'(t_0) = a$, $S'(t_n) = b$
- not-a-knot: $S'''(x)$ continuous at t_1 and t_{n-1}
- periodic: S' and S'' are periodic at the ends

natural cubic spline

How do we find $a_{0,i}, a_{1,i}, a_{2,i}, a_{3,i}$ for each i?

Consider knots $t_0, \ldots, t_n$. Follow our example with the following steps:

1. Assume we knew $S''(t_i)$ for each i
2. $S_i''(x)$ is linear, so construct it
3. Get $S_i(x)$ by integrating $S_i''(x)$ twice
4. Impose continuity
5. Differentiate $S_i(x)$ to impose continuity on $S'(x)$

natural cubic spline: Step 1

Assume we knew $S''(t_i)$ for each i

We know $S''(x)$ is continuous. So assume

$$z_i = S''(t_i)$$

(we don't actually know z_i, not yet at least)

natural cubic spline: Step 2

$S_i''(x)$ is linear, so construct it

Since $S_i''(x)$ is linear, and

$$S_i''(t_i) = z_i$$
$$S_i''(t_{i+1}) = z_{i+1}$$

we can write $S_i''(x)$ as

$$\begin{aligned} S_i''(x) &= z_i \frac{t_{i+1} - x}{t_{i+1} - t_i} + z_{i+1} \frac{x - t_i}{t_{i+1} - t_i} \\ &= \frac{z_i}{h_i}(t_{i+1} - x) + \frac{z_{i+1}}{h_i}(x - t_i) \end{aligned}$$

where $h_i = t_{i+1} - t_i$.

natural cubic spline: Step 3

Get $S_i(x)$ by integrating $S_i''(x)$ twice

Take

$$S_i''(x) = \frac{z_i}{h_i}(t_{i+1} - x) + \frac{z_{i+1}}{h_i}(x - t_i)$$

and integrate once:

$$S_i'(x) = -\frac{z_i}{2h_i}(t_{i+1} - x)^2 + \frac{z_{i+1}}{2h_i}(x - t_i)^2 + \hat{C}_i$$

twice:

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \hat{C}_i x + \hat{D}_i$$

adjust:

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i}(x - t_i)^3 + C_i(x - t_i) + D_i(t_{i+1} - x)$$

natural cubic spline: Step 4

Impose continuity

For each interval $[t_i, t_{i+1}]$, we require $S_i(t_i) = y_i$ and $S_i(t_{i+1}) = y_{i+1}$:

$$y_i = S_i(t_i) = \frac{z_i}{6h_i}(t_{i+1} - t_i)^3 + \frac{z_{i+1}}{6h_i}(t_i - t_i)^3 + C_i(t_i - t_i) + D_i(t_{i+1} - t_i)$$

$$= \frac{z_i}{6}h_i^2 + D_i h_i$$

$$D_i = \frac{y_i}{h_i} - \frac{h_i}{6}z_i$$

and

$$y_{i+1} = S_i(t_{i+1}) = \frac{z_i}{6h_i}(t_{i+1} - t_{i+1})^3 + \frac{z_{i+1}}{6h_i}(t_{i+1} - t_i)^3 + C_i(t_{i+1} - t_i) + D_i(t_{i+1} - t_{i+1})$$

$$= \frac{z_{i+1}}{6}(h_i)^2 + C_i h_i$$

$$C_i = \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}$$

natural cubic spline: Step 4

Impose continuity

So far we have

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1}-x)^3+\frac{z_{i+1}}{6h_i}(x-t_i)^3+\left(\frac{y_{i+1}}{h_i}-\frac{h_i}{6}z_{i+1}\right)(x-t_i)+\left(\frac{y_i}{h_i}-\frac{h_i}{6}z_i\right)(t_{i+1}-x)$$

natural cubic spline: Step 5

Differentiate $S_i(x)$ to impose continuity on $S'(x)$

$$S_i'(x) = -\frac{z_i}{2h_i}(t_{i+1} - x)^2 + \frac{z_{i+1}}{2h_i}(x - t_i)^2 + \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1} - \frac{y_i}{h_i} + \frac{h_i}{6}z_i$$

We need $S_i'(t_i) = S_{i-1}'(t_i)$:

$$S_i'(t_i) = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \underbrace{\frac{1}{h_i}(y_{i+1} - y_i)}_{b_i}$$

$$S_{i-1}'(t_i) = \frac{h_{i-1}}{6}z_{i-1} + \frac{h_{i-1}}{3}z_i + \underbrace{\frac{1}{h_{i-1}}(y_i - y_{i-1})}_{b_{i-1}}$$

Thus z_i is defined by

$$h_{i-1}z_{i-1} + 2(h_i + h_{i-1})z_i + h_i z_{i+1} = 6(b_i - b_{i-1})$$

natural cubic spline: Step 6

solve

z_i is defined by

$$h_{i-1}z_{i-1} + 2(h_i + h_{i-1})z_i + h_i z_{i+1} = 6(b_i - b_{i-1})$$

- This is $n-1$ equations, $n-1$ unknowns ($z_0 = z_n = 0$ already)
- an $(n-1)\times(n-1)$ tridiagonal system (add 2 for z_0 and z_n)

$$\begin{bmatrix} 1 & & & & & & & \\ h_0 & u_1 & h_1 & & & & & \\ & h_1 & u_2 & h_2 & & & & \\ & & h_2 & u_3 & h_3 & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & h_{n-3} & u_{n-2} & h_{n-2} & \\ & & & & & h_{n-2} & u_{n-1} & h_{n-1} \\ & & & & & & & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-2} \\ z_{n-1} \\ z_n \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-2} \\ v_{n-1} \\ 0 \end{bmatrix}$$

$$\begin{aligned} u_i &= 2(h_i + h_{i-1}) \\ v_i &= 6(b_i - b_{i-1}) \end{aligned}$$

example

Find the natural cubic spline for

x	-1	0	1
y	1	2	-1

1. Determine h_i, b_i, u_i, v_i

$$h = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ -3 \end{bmatrix} \quad u = \begin{bmatrix} 4 \end{bmatrix} \quad v = \begin{bmatrix} -24 \end{bmatrix}$$

2. Solve

$$\begin{bmatrix} 1 & & \\ 1 & 4 & 1 \\ & & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -24 \\ 0 \end{bmatrix}$$

3. Result:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -6 \\ 0 \end{bmatrix}$$

example

Find the natural cubic spline for

x	-1	0	1
y	1	2	-1

1. Plug z_i into

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1}-x)^3 + \frac{z_{i+1}}{6h_i}(x-t_i)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x-t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1}-x)$$

$$S(x) = \begin{cases} -(x+1)^3 + 3(x+1) - x & -1 \leqslant x < 0 \\ -(1-x)^3 - x + 3(1-x) & 0 \leqslant x < 1 \end{cases}$$

Algorithm: page 391-393 (NMC6), page 403-405 (NMC5)

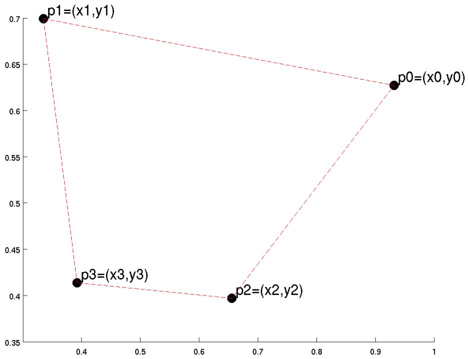
1. Compute for $i = 0, \ldots, n-1$

$$h_i = t_{i+1} - t_i \qquad b_i = \frac{1}{h_i}(y_{i+1} - y_i)$$

2. Set u, v:
3. tridiagonal solve to get z
4. substitute into the nested form for $S(x)$ equation 12, page 392 NMC6 (NMC5: equation 10 page 404)

Bézier Curves

- Different than splines
- Similar process
- Does not require interpolation, only that the curve stay within the *convex hull* off the control points
- Can move one point with only local effect



Parametric Form

A function $y = f(x)$ can be expressed in parametric form. The parametric form represents a relationship between x and y through a parameter t:

$$x = F_1(t) \qquad y = F_2(t)$$

Example

The equation for a circle can be written in parametric form as

$$\begin{aligned} x &= r\cos(\theta) \\ y &= r\sin(\theta) \end{aligned}$$

(x, y) is now expressed as $(x(t), y(t))$. We will use $0 \leqslant t \leqslant 1$.

Bézier Points

Consider a set of *control* points:

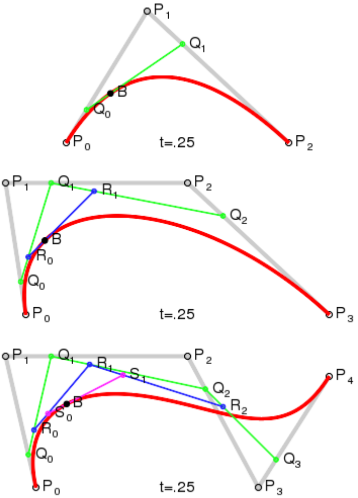
$$p_i = (x_i, y_i),\ \ i = 0, \ldots, n$$

These may be in any order.

So $p_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ or in parametric form the set of points is expressed as

$$P(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$$

Bézier Curves



points Q_0 and Q_1 vary linearly from $P_0 \to P_1$ and $P_1 \to P_2$

$Q's$ vary linearly, $R's$ vary quadratically

all within the hull of the control points

Bernstein Polynomial

The polynomials

$$q(t) = (1-t)^{n-i}t^i$$

have the nice property that for $0 < i < n$, $q(0) = q(1) = 0$.
If we scale them with

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

we have the *Bernstein* polynomials:

$$b_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

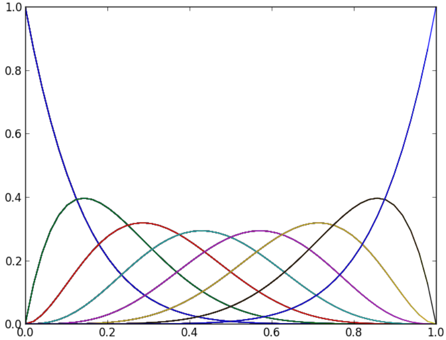
Among the interesting properties is that

$$\sum_{i=0}^{n} b_{i,n}(t) = (t + (1-t))^n = 1$$

(hint: binomial theorem)

Bernstein Polynomial

For $n = 7$ the Bernstein Polynomials look like this:



bernstein7.py

Bernstein Polynomial

The nth-degree Bézier Polynomial through the $n+1$ points is given by

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

where

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

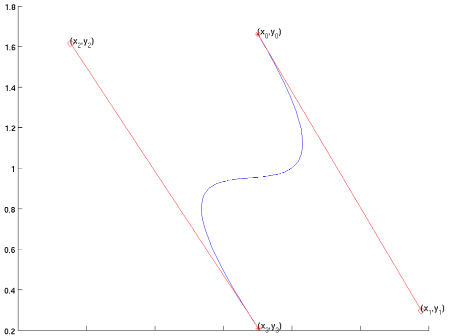
For $n = 3$ (cubic) we have

$$x(t) = (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t) t^2 x_2 + t^3 x_3$$
$$y(t) = (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t) t^2 y_2 + t^3 y_3$$

Cubic Bézier Curve

$$x(t) = (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t) t^2 x_2 + t^3 x_3$$
$$y(t) = (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t) t^2 y_2 + t^3 y_3$$

Notice that $(x(0), y(0)) = p_0$ and $(x(1), y(1)) = p_3$. So the Bézier curve interpolates the endpoints but not the interior points.



Bézier Curves

$$x(t) = (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t) t^2 x_2 + t^3 x_3$$
$$y(t) = (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t) t^2 y_2 + t^3 y_3$$

Notice:

1. $P(0) = p_0$ and $P(1) = p_3$
2. The slope of the curve at $t = 0$ is a secant:

$$\frac{dy}{dx} = \frac{dy}{dt}\frac{dt}{dx} = \frac{3(y_1 - y_0)}{3(x_1 - x_0)} = \frac{y_1 - y_0}{x_1 - x_0}$$

3. The slope of the curve at $t = 1$ is a secant between the last two control points.
4. The curve is contained in the convex hull of the control points

Bézier Curves

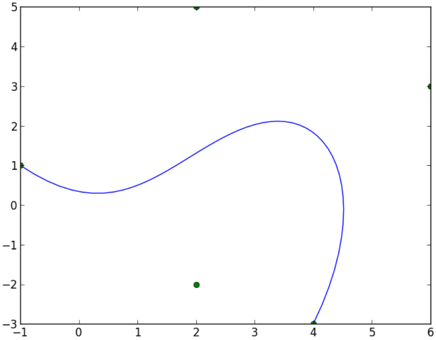
$$x(t) = (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t) t^2 x_2 + t^3 x_3$$
$$y(t) = (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t) t^2 y_2 + t^3 y_3$$

Easier construction given points $p_0, \ldots, p_3$:

$$P(t) = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

5 Point curve

The curve through 5 points looks like this:



Bezier.py

Vector Graphics, Fonts, Adobe

Vector Graphics include primitives like

- lines, polygons
- circles
- **Bézier curves**
- **Bézier splines** or Bezigons
- text (letters created from **Bézier** curves)

Flash Animation

- Use **Bézier curves** to construct animation path

Microsoft Paint, Gimp, etc

- Use **Bézier curves** to draw curves
- `http://msdn2.microsoft.com/en-us/library/ms534244.aspx`

Graphics

- Use **Bézier surfaces** to draw smooth objects

Bézier Surfaces

Take (n, m). That is, $(n+1, m+1)$ control points $p_{i,j}$ in 2d. Then let

$$\mathbf{P}(t,s) = \sum_{i=0}^{n} \sum_{j=0}^{m} \phi_{ni}(t)\phi_{mj}(s)\mathbf{p}_{ij}$$

Where, again, ϕ_{ni} are the Bernstein polynomials:

$$\phi_{ni}(t) = \binom{n}{i} (1-t)^{n-i} t^i$$

- again, all within the convex hull of control points
- `http://www.math.psu.edu/dlittle/java/parametricequations/beziersurfaces/index.html`