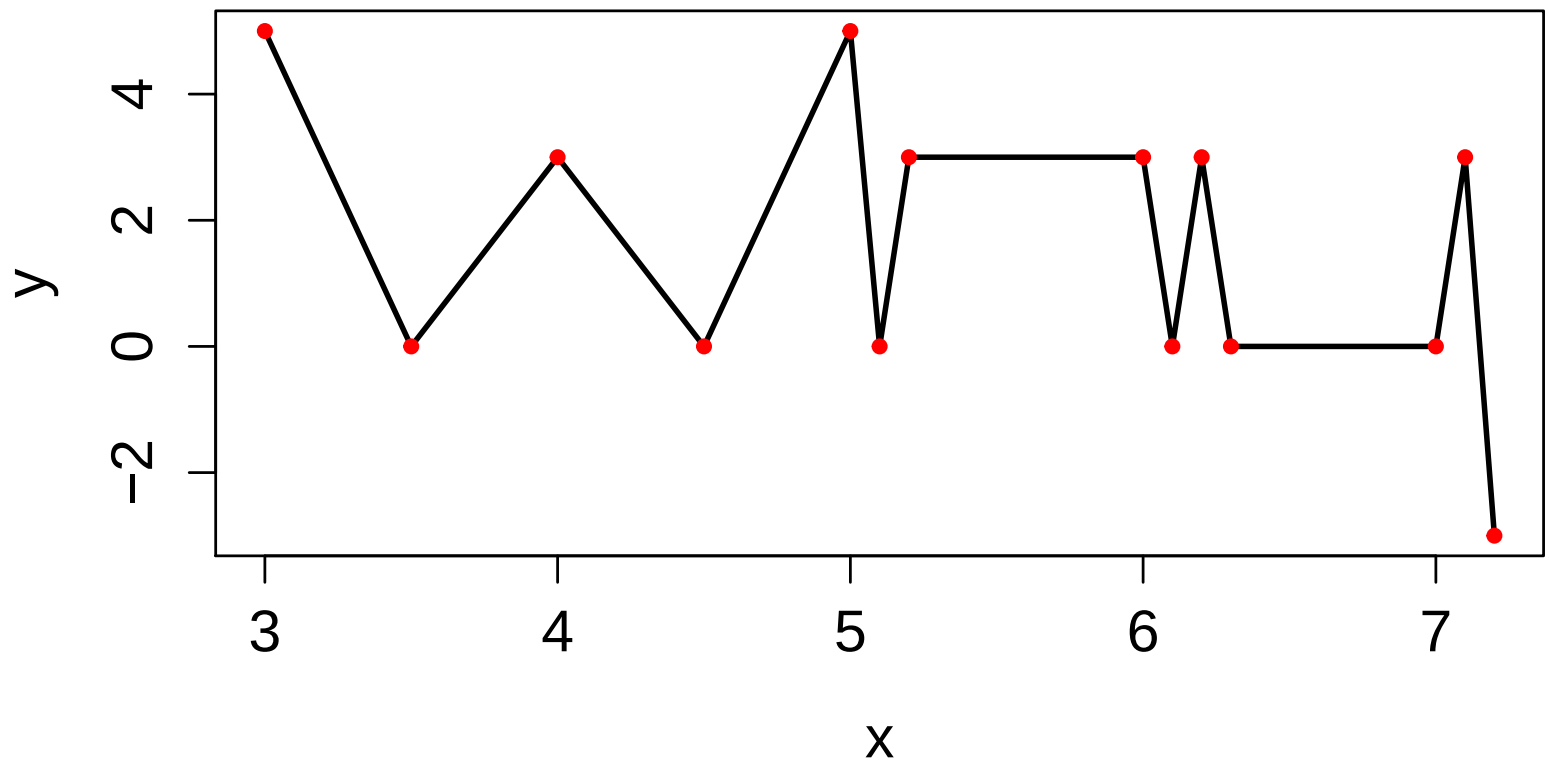
Computational Linear Algebra: Splines and Bézier Curves

David Shuman

March 10, 2022

Linear Splines

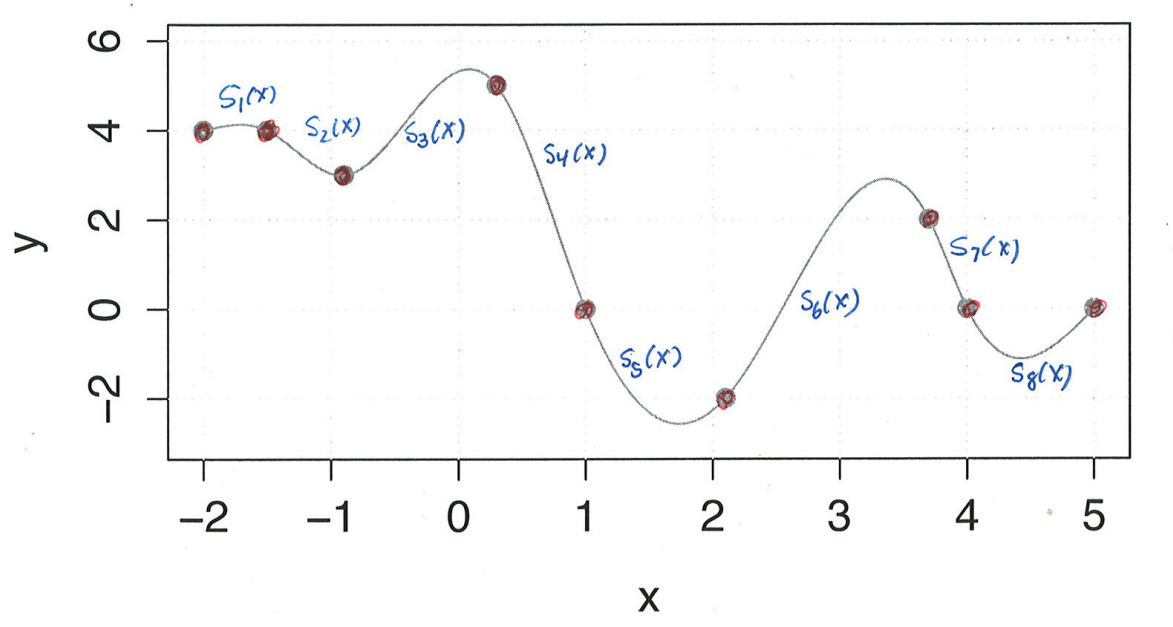
- "Connect the dots"
- What many numerical platforms like MATLAB and R use to plot line graphs of discretized functions
- plot(x,y,type="l",lwd=2)
- Not smooth!!



Cubic Splines

Fit a piecewise twice continuously differentiable function to the data by fitting a cubic (3rd degree polynomial) between each pair of points such that where they meet:

1. The y-values match
2. The first derivatives match
3. The second derivatives match

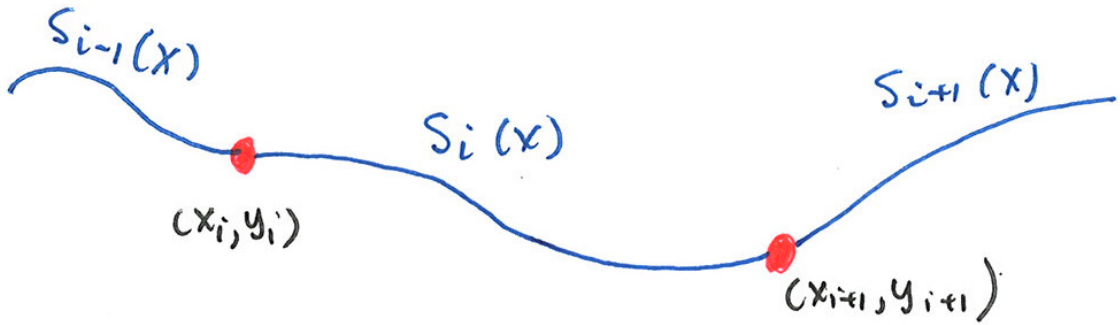


Note: n data points (knots) and $n-1$ spline functions

Cubic Splines

Define:

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$



Total unknowns: $3(n-1) = 3n - 3$
Want:

- $S_i(x_i) = y_i$ (done)
- $S_i(x_{i+1}) = y_{i+1}$ $\quad [n-1]$
- $S'_{i-1}(x_i) = S'_i(x_i)$ $\quad [n-2]$
- $S''_{i-1}(x_i) = S''_i(x_i)$ $\quad [n-2]$

Total equations: $(n-1) + 2(n-2) = 3n - 5$

Endpoint Conditions

Add two more equations, one on the first node and one on the last:

- Natural Spline. No concavity at the endpoints:

$$S_1''(x_1) = 0 \qquad\qquad S_{n-1}''(x_n) = 0$$

- Clamped. Specify the slope at the first and last endpoint

$$S_1'(x_1) = m_1 \qquad\qquad S_{n-1}'(x_n) = m_n$$

- FMM (not-a-knot)
  - $S_1 = S_2$ is a single cubic equation that fits through the first 3 points
  - x_2 is no longer a knot point
  - The same is true of $S_{n-2} = S_{n-1}$ and x_{n-1} is not a knot point
  - Also called the FMM-method named after its inventors, Forsythe, Malcolm, and Moler
- There are many others

Once the two endpoint conditions are specified, there is a unique set of spline functions through these points

Equations for Natural Cubic Splines

After taking some derivatives, making substitutions $\delta_i = x_{i+1} - x_i$ and $r_i = \frac{y_{i+1}-y_i}{\delta_i} - \frac{y_i-y_{i-1}}{\delta_{i-1}}$, and doing some algebraic manipulations, we get the system of equations (see p. 169 and 170)

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & 0 & 0 & 0 \\ 0 & \delta_2 & 2(\delta_2+\delta_3) & \delta_3 & 0 & 0 \\ 0 & 0 & \delta_3 & 2(\delta_3+\delta_4) & \delta_4 & 0 \\ 0 & 0 & 0 & \delta_4 & 2(\delta_4+\delta_5) & \delta_5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = 3 \begin{bmatrix} 0 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ 0 \end{bmatrix}$$

- Once you solve for the c_i's,

$$d_i = \frac{c_{i+1} - c_i}{3\delta_i} \text{ and } b_i = \frac{y_{i+1} - y_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1})$$

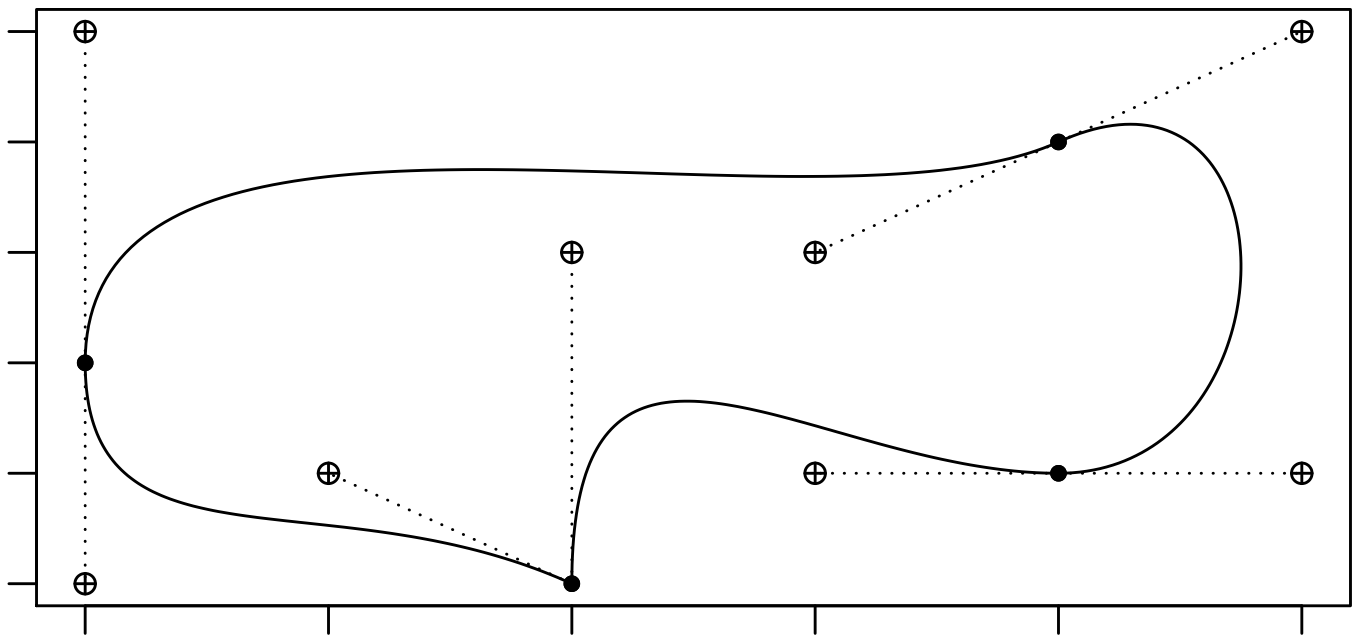
- Observe that the linear system is numerically nice:
  - tri-diagonal
  - strictly diagonally dominant
  - you can even play around with it to make it symmetric

Splines in R

- We are not going to program our own spline function, though — key point!— you could! It is not very hard and is well within your capabilities
- We will learn to use the R commands `spline` and `splinefun`
- Activity A14

Bézier Curves

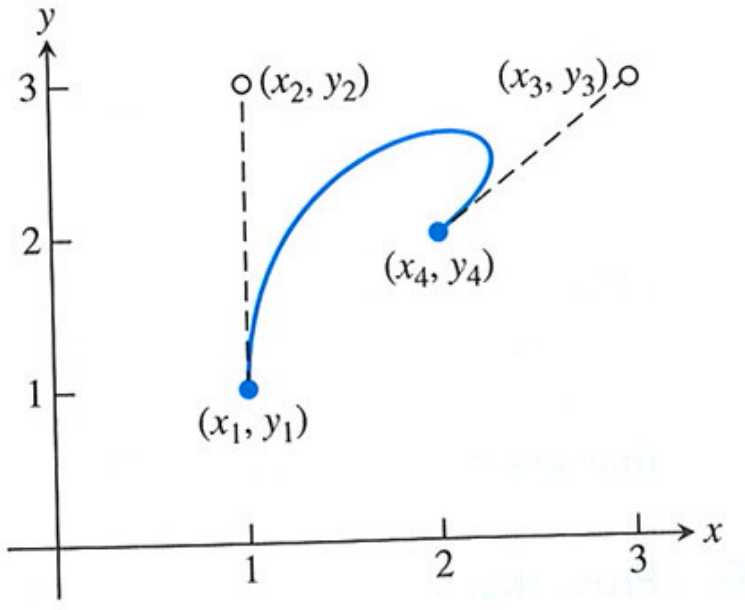
- Bézier curves are cubic splines in which the user controls the slope at each knot point
- To do this we must give up control over the smoothness of the first and second derivative at each point



- This plot has 4 Bézier splines
- Each spline has two endpoints • and two control points ⊕, which determine the tangent line direction at the endpoint

A Single Bézier Curve

Each Bézier curve is a cubic parametric equation



$$b(t) = (x(t), y(t))$$

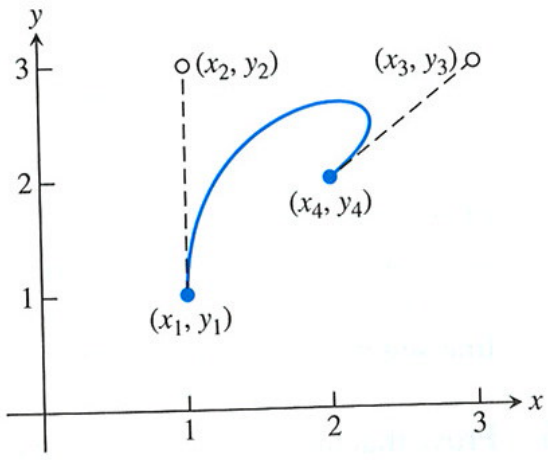
$$x(t) = x_1 + b_x t + c_x t^2 + d_x t^3$$
$$y(t) = y_1 + b_y t + c_y t^2 + d_y t^3$$

$$0 \leq t \leq 1$$

with the following properties:

$$x(0) = x_1 \qquad x(1) = x_4$$
$$y(0) = y_1 \qquad y(1) = y_4$$
$$x'(0) = 3(x_2 - x_1) \qquad x'(1) = 3(x_4 - x_3)$$
$$y'(0) = 3(y_2 - y_1) \qquad y'(1) = 3(y_4 - y_3)$$

A Single Bézier Curve



$$b(t) = (x(t), y(t))$$

$$x(t) = x_1 + b_x t + c_x t^2 + d_x t^3$$
$$y(t) = y_1 + b_y t + c_y t^2 + d_y t^3$$

$$0 \leq t \leq 1$$

- If you set

$$b_x = 3(x_2 - x_1) \qquad b_y = 3(y_2 - y_1)$$
$$c_x = 3(x_3 - x_2) - b_x \qquad c_y = 3(y_3 - y_2) - b_y$$
$$d_x = x_4 - x_1 - b_x - c_x \qquad d_y = y_4 - y_1 - b_y - c_y$$

  then the conditions on the previous slide are met

- Note: this is easy and fast to code
- Can extend to higher dimensions: spline example and Bézier example