Splines

There are cases where polynomial interpolation is bad

• overshoot
• oscillations

Example

\[
\begin{align*}
0 & \quad 0.5 & \quad 1 & \quad 1.5 & \quad 2 & \quad 2.5 & \quad 3 & \quad 3.5 & \quad 4 \\
-6 & & -4 & & -2 & & 0 & & 2 & & 4 & & 6 \\
\end{align*}
\]

\[
(\ln |x| + 1)
\]

Interpolation at -4,-3,-2,-1,0,1,2,3,4

Idea behind splines

• use lower order polynomials to connect subsets of data points
• make connections between adjacent splines smooth
• because lower order polynomials, avoid oscillations and overshoots

We will cover

• linear splines
• quadratic splines
• cubic splines - most common

Autocad uses splines extensively

Linear splines

- connect each two points with straight line functions connecting each pair of points are

\[
\begin{align*}
f(x) & = f(x_i) + m_i (x - x_i) \\
f(x) & = f(x_i) + m_i (x - x_i) \\
\vdots & \\
f(x) & = f(x_n) + m_n (x - x_n)
\end{align*}
\]

m is slope between points

\[
m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}
\]

Linear splines are exactly the same as linear interpolation!

Example:

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>5</td>
<td>-0.67</td>
</tr>
<tr>
<td>8.2</td>
<td>4.2</td>
<td>-1.72</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.21</td>
</tr>
<tr>
<td>12.8</td>
<td>4.5</td>
<td></td>
</tr>
</tbody>
</table>

Linear splines
Problem with linear splines - not smooth at data points (or knots)
first derivative is not continuous
use higher order splines to get continuous derivatives - equate derivatives at neighboring splines

Quadratic splines - continuous first derivatives can have discontinuous second and higher derivatives
Derive second order polynomial between each pair of points \( f_i(x) = a_i x^2 + b_i x + c_i \)
For \( n+1 \) points \( (i=0,1,\ldots,n) \) - \( n \) intervals - \( 3n \) unknown parameters \((a's, b's, and c's)\)
Need 3n equations

Equations
• for interior knots (data points), equations on either side must equal knot at knot
  \[
  a_{i-1} x^2 + b_{i-1} x + c_{i-1} = f_i(x) = a_i x^2 + b_i x + c_i = a_{i+1} x^2 + b_{i+1} x + c_{i+1}
  \]
  \( n-1 \) interior knots
  \( 2n-2 \) interior equations

• the first and last functions must pass through the end points
  \[
  a_0 x^2 + b_0 x + c_0 = f_0(x) = f_1(x) = a_n x^2 + b_n x + c_n
  \]
  2 more equations: total is 2n so far

• first derivatives at interior nodes must be equal
  \[
  2a_i x_i + b_i = 2a_{i+1} x_i + b_{i+1}
  \]
  Another \( n-1 \) equations for a total of 3n-1 equations
  We need 3n equations.
  What is the last equation?

Last equation is arbitrary
We use
• assume second derivative is 0 at first point
  \( a_1 = 0 \)
Connect first two points by a straight line
Example:

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>8.2</td>
<td>4.2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>2</td>
</tr>
<tr>
<td>12.8</td>
<td>4.5</td>
<td>3</td>
</tr>
</tbody>
</table>

n=3

Set up interior equations (4 of them)

\[
\begin{align*}
(0.2)^2a_1 + 2.8b_1 + c_1 &= -4.2 \\
(0.2)^2a_1 + 2.8b_1 + c_1 &= -4.2 \\
(10)^2a_1 + 10b_1 + c_1 &= -1.1 \\
(10)^2a_1 + 10b_1 + c_1 &= -1.1
\end{align*}
\]

9 unknowns

Set up end-point equations (2 of them)

\[
\begin{align*}
(7)^2a_1 + 7b_1 + c_1 &= 5 \\
(12.8)^2a_1 + 12.8b_1 + c_1 &= 4.5
\end{align*}
\]

Set up first derivative equations at interior knots (2 of them)

\[
\begin{align*}
2a_1(8.2) + b_1 &= 2a_2(8.2) + b_2 \\
2a_2(10) + b_2 &= 2a_3(10) + b_3
\end{align*}
\]

Final arbitrary equation

\[a_1 = 0\]

The 9 equations are

\[
\begin{align*}
67.24a_1 + 8.2b_1 + c_1 &= 4.2 \\
67.24a_1 + 8.2b_1 + c_1 &= 4.2 \\
100a_1 + 10b_1 + c_1 &= 1.1 \\
100a_1 + 10b_1 + c_1 &= 1.1 \\
49a_1 + 7b_1 + c_1 &= 5 \\
163.84a_1 + 12.8a_1 + c_1 &= 4.5 \\
16.4a_1 + b_1 &= 16.4a_2 + b_2 \\
20a_2 + b_2 &= 20a_3 + b_3 \\
a_1 &= 0
\end{align*}
\]

Rearrange, and use \(a_1=0\) to eliminate

\[
\begin{align*}
8.2b_1 + c_1 &= 4.2 \\
67.24a_1 + 8.2b_2 + c_2 &= 4.2 \\
100a_1 + 10b_2 + c_3 &= 1.1 \\
100a_1 + 10b_3 + c_4 &= 1.1 \\
7b_1 + c_5 &= 5 \\
163.84a_1 + 12.8a_2 + c_6 &= 4.5 \\
16.4a_2 + b_2 - b_1 &= 0 \\
20a_2 + b_2 - 20a_3 - b_3 &= 0
\end{align*}
\]

Put in matrix form

\[
\begin{bmatrix}
8.2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 67.24 & 8.2 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 100 & 10 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 100 & 10 & 0 & 0 & 0 \\
7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 163.84 & 12.8 & 1 & 0 \\
-1 & 0 & 16.4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 1 & 0 & -20 & -1 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
a_1 \\
b_1 \\
a_2 \\
b_2 \\
c_1 \\
c_2 \\
c_3 \\
c_4 \\
c_5 \\
c_6
\end{bmatrix}
\begin{bmatrix}
4.2 \\
1.1 \\
5 \\
4.5 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]
Solve by any matrix method

Get

\[
\begin{align*}
    a_1 &= 0 \\
    b_1 &= -0.586 \\
    c_1 &= 8.950 \\
    e_1 &= 29.764 \\
    f_1(s) &= -\frac{2}{3}x^2 + \frac{29}{3}x - 29.764 \\
    f_1(s) &= -0.586x^2 + 8.950x - 29.764 \\
    f_1(s) &= 1.426x^2 - 31.293x + 171.452
\end{align*}
\]

Cubic splines avoid the straight line and the over-swing

\[f_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1\]

Can develop method like we did for quadratic - 4n unknowns - 4n equations

- interior knot equality
- end point fixed
- interior knot first derivative equality
- assume derivative value if needed

Can do a set of algebraic manipulations (which book shows) to reduce 4n equations to n-1 equations

\[
\begin{align*}
    f_1(s) &= \frac{f''(s)}{f'(s)} \left( x - x_i \right) - \frac{f''(s)}{f'(s)} \left( x - x_{i+1} \right) \\
    &= \frac{f(s) - f(x_{i+1})}{x_{i+1} - x_i} \left( x - x_i \right) - \frac{f(s) - f(x_i)}{x_{i+1} - x_i} \left( x - x_{i+1} \right) \\
    &= \frac{f(s)}{x_{i+1} - x_i} - \frac{f'(s)}{x_{i+1} - x_i} \left( x - x_i \right)
\end{align*}
\]

Unknowns - second derivatives

Second derivatives are evaluated using

\[
\begin{align*}
    (x_i - x_{i+1})f''(x_{i+1}) + 2(x_{i+1} - x_i)f''(x_i) + (x_i - x_{i+1})f''(x_i)
    &= \frac{6}{x_{i+1} - x_i} [f(x_{i+1}) - f(x_i)] + \frac{6}{x_i - x_{i+1}} [f(x_i) - f(x_{i+1})]
\end{align*}
\]

with end conditions

\[
\begin{align*}
    f''(x_1) &= 0 \\
    f''(x_n) &= 0
\end{align*}
\]

Becomes a tridiagonal system
Method outline

• Set up equations for second derivatives
• Get second derivatives and plug into spline equation for each section

Example:

\[ f(x) = \ln|\text{abs}(x) + 1| \]

\[ x = \{-4, -3, 3, 4\} \]

Now plug second derivative values into cubic spline equation:

\[ f(x) = \frac{f''(x_i)}{x_i - x_j} (x_i - x) + \frac{f''(x_j)}{x_i - x_j} (x - x_j) + \frac{f'(x_j)}{x_i - x_j} (x_j - x) + f(x_j) \]

This gets messy, so let’s go to Matlab

```
% Use spline command to fit a set of cubic splines to f(x) = ln(|abs(x) + 1|) given values of y at x = {-4, -3, 3, 4}. Use these cubic splines to generate interpolated values at xx = -4:0.1:4

if you want a challenge, compare with 8th order interpolating polynomial
```

Because cubic splines are used so widely, Matlab has built-in code for it

\[ >> \text{help spline} \]

```
SPLINE Cubic spline data interpolation.

PP = SPLINE(X,Y) provides the piecewise polynomial form of the cubic spline interpolant to the data values Y at the data sites X, for use with the evaluator PPVAL and the spline utility UNMKPP. X must be a vector. If Y is a vector, then Y is taken as the value to be matched at X. If Y is a matrix or ND array, Y(:,...,:,j) is taken as the value to be matched at X(j), hence the last dimension of Y must equal length(X) -- see below for an exception to this.

YY = SPLINE(X,Y,XX) is the same as YY = PPVAL(SPLINE(X,Y),XX), thus providing, in YY, the values of the interpolant at XX. For information regarding the size of YY see PPVAL.
```

In-class example to try:

Use spline command to fit a set of cubic splines to \( f(x) = \ln(|\text{abs}(x) + 1|) \) given values of \( y \) at \( x = \{-4, -3, 3, 4\} \). Use these cubic splines to generate interpolated values at \( xx = -4:0.1:4 \)

Hint:

\[ x = -4:4; \]

\[ y = \log|\text{abs}(x) + 1|; \]

\[ xx = -4:0.1:4; \]

If you want a challenge, compare with 8th order interpolating polynomial
Should get something like this