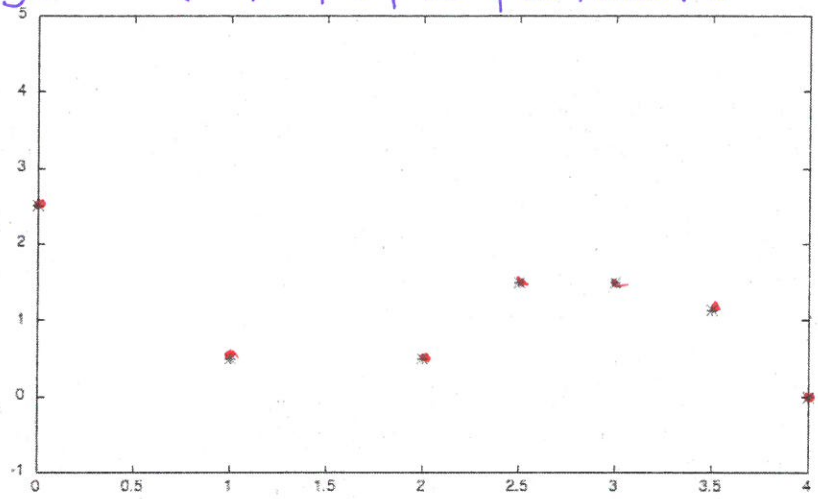
Day 13

Interpolation — given a set of points find a function that "fits" interpolates the data.

x_nodes	0	1	2	2.5	3	3.5	4
y_nodes	2.5	.5	.5	1.5	1.5	1.125	0

test1.jpg



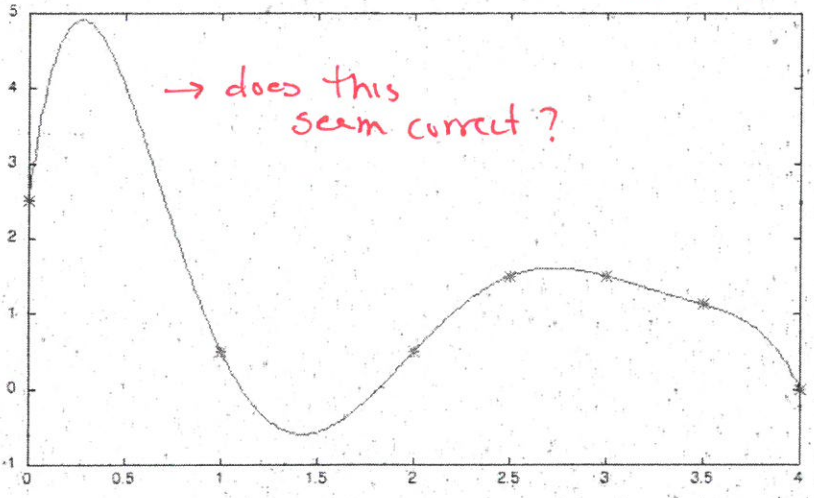
How would we naturally draw a curve?

7 data points.

Polynomial Interpolation...

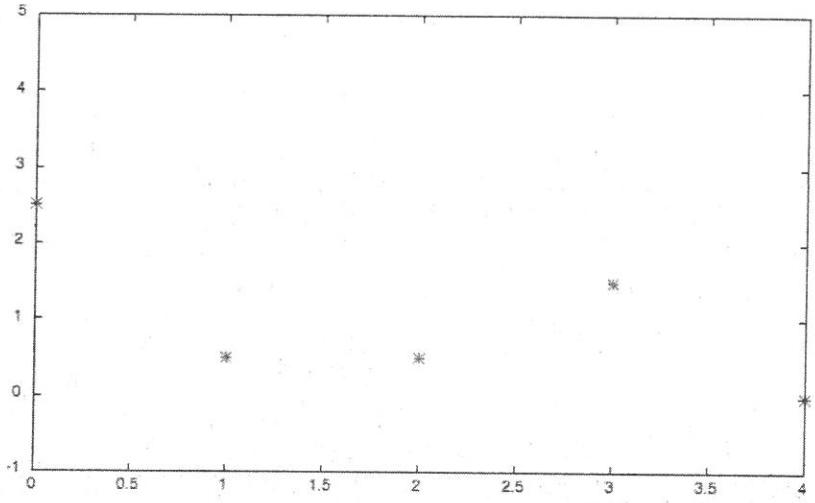
This is P_6...

test2.jpg



Remove some of the points.

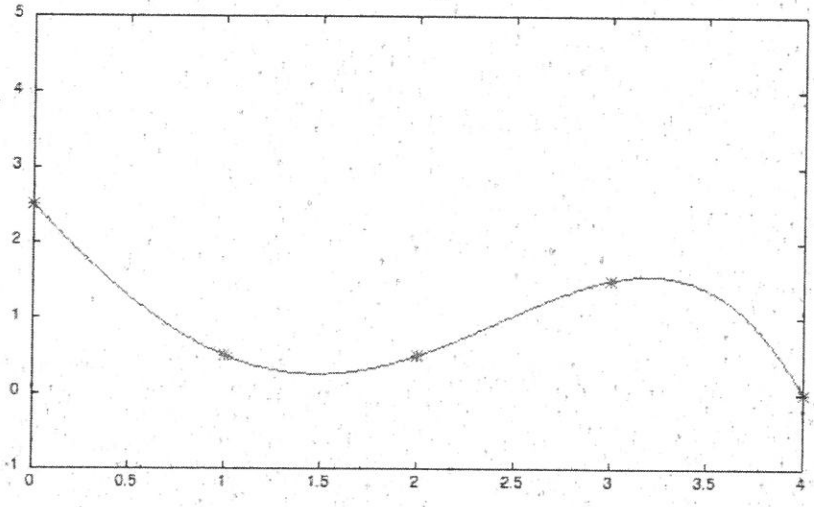
test1small.jpg



5 data points.

P_4 – does a better job of finding the natural curve.

test2small.jpg



Problems w/ Polynomial Interpolation...

- As n increases, we don't always get a better approximation.
- If we have 31 data points then we must use P_{30} even if P_4 would fit better... How do we remove data.
- Error has to do with spacing of nodes ... we don't always have control of this.

Today – Piecewise Interpolation.

First try... connect the dots with straight lines... use the data as endpoints.

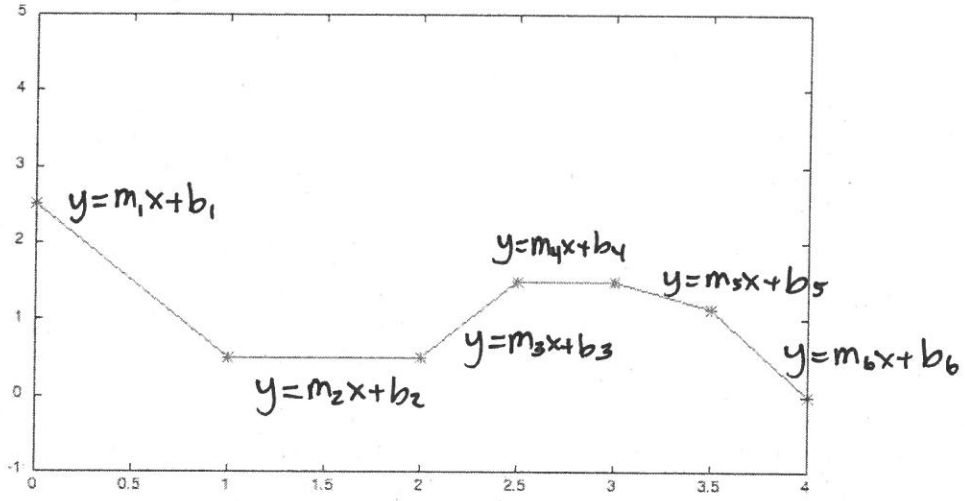
Piecewise Linear
- easy to do
- not a smooth curve, can't do calculus on edges.

Piecewise Quadratic. (NO GRAPH.)
- Divide Points into groups of 3 and fit quadratic.
- If n=even we will have at least one line segment
- Still discontinuous where the polynomials join.

DRAW THESE ON THE POINTS.

PIECEWISE LINEAR

test3.py

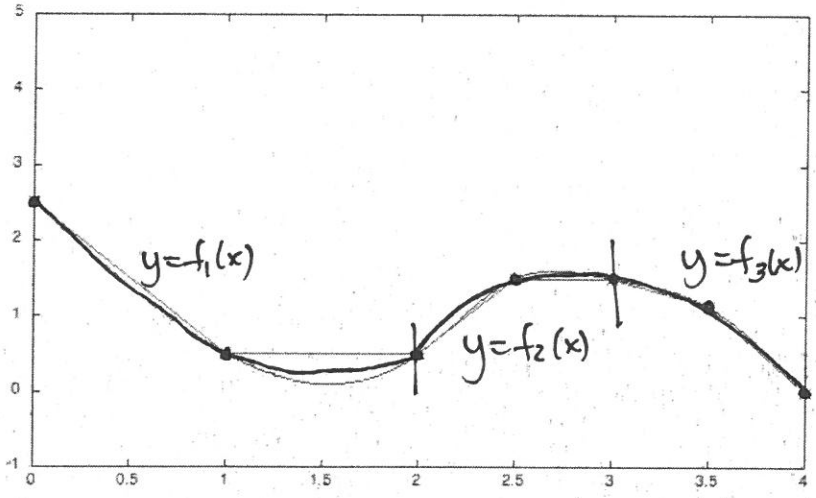


$$L(x) = \begin{cases} m_1x + b_1 & 0 \le x < 1 \\ m_2x + b_2 & 1 \le x < 2 \\ \vdots & \vdots \end{cases}$$

How could we make this better?

PIECEWISE QUADRATIC.

NOT ON A GRAPH.



GOAL – Find a smooth, non-oscillatory interpolation.

NATURAL CUBIC SPLINE INTERPOLATION.

Assume we have n-data points (x_i, y_i) $i = 1, 2, \ldots, n$
and $x_1 < x_2 < \ldots < x_n$ no repeated x-values...

let $a = x_1$ and $b = x_n$ giving a range $[a, b]$ for our x-values.

We seek a function $S(x)$ that interpolates our data

$$S(x_i) = y_i \quad i = 1, 2, \ldots, n$$

and require $S'(x)$ and $S''(x)$ are continuous.
This guarantees smoothness of $S(x)$.

finally require $\int_a^b [S''(x)]^2 dx$ be as small as possible.

↖ this stops extreme curvature.

There is a unique solution to this problem... we find that

NATURAL CUBIC SPLINE

1. $S(x)$ is a polynomial of degree ≤ 3 on each sub interval $[x_{j-1}, x_j]$
2. $S(x)$, $S'(x)$ and $S''(x)$ are continuous for $a \leq x \leq b$
3. $S''(x_1) = S''(x_n) = 0$

boundary condition at the ends of the domain.

How do we construct these things?

Start by considering the curvature at each data point.

$$M_i \equiv s''(x_i) \quad i = 1, 2, \ldots, n$$

if $s(x)$ is cubic then $s''(x)$ is linear and on each interval has the endpoints:

$$s''(x_{j-1}) = M_{j-1} \quad \text{and} \quad s''(x_j) = M_j$$

we can write down the equation for a straight line!!

$$s''(x) = \frac{(x_j - x)M_{j-1} + (x - x_{j-1})M_j}{x_j - x_{j-1}}$$

for $x_{j-1} \le x \le x_j$

Should look like $P_1(x)$...

Just a line between (x_{j-1}, M_{j-1}) (x_j, M_j)

Now we can integrate this twice!

- How many constants of integration? TWO
- Use the boundary conditions to solve for these constants.

$$s(x_{j-1}) = y_{j-1} \quad \text{and} \quad s(x_j) = y_j$$

$$s(x) = \frac{(x_j - x)^3 M_{j-1} + (x - x_{j-1})^3 M_j}{6(x_j - x_{j-1})} + \frac{(x_j - x)y_{j-1} + (x - x_{j-1})y_j}{(x_j - x_{j-1})} - \frac{1}{6}(x_j - x_{j-1})\left[(x_j - x)M_{j-1} + (x - x_{j-1})M_j\right]$$

This is the cubic spline FORMULA...

Here we know x_j and y_j, x are the points where we will graph our spline

BUT How do we solve for M_j?

remember this is the curvature!

We must require continuity at the joining points.

$s'(x)$ on $[x_{j-1}, x_j]$ must match $s'(x)$ on $[x_j, x_{j+1}]$

AT x_j

If we set $S'(x_j) = S'(x_j)$ across these jumps. We find.

$$\frac{x_j - x_{j-1}}{6} M_{j-1} + \frac{(x_{j+1} - x_{j-1})}{3} M_j + \frac{(x_{j+1} - x_j)}{6} M_{j+1}$$

$$= \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}} \qquad j = 2, 3, \ldots, n$$

this holds at all of the interior points!

On the boundary M_1 and M_n we require

$M_1 = M_n = 0$ NO curvature — NATURAL

we can pick other conditions for these boundaries.

SHOW SPLINE GRAPH

Ex Say we are given $(1,1)(2,1/2)(3,1/3)(4,1/4)$ and we want a cubic spline interpolation.

We have $n=4$ points and need to first solve for M_1, M_2, M_3, M_4.

$M_1 = 0$ and $M_4 = 0$ "Natural Boundaries"

Really we only solve for interior curvature points...

$j=2$ $$\frac{x_2 - x_1}{6} M_1 + \frac{x_3 - x_1}{3} M_2 + \frac{x_3 - x_2}{6} M_3 = \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_3 - x_1}$$

$$\frac{1}{6} M_1 + \frac{2}{3} M_2 + \frac{1}{6} M_3 = \frac{1}{3}$$

$j=3$ $$\frac{1}{6} M_2 + \frac{2}{3} M_3 + \frac{1}{6} M_4 = \frac{1}{2}$$

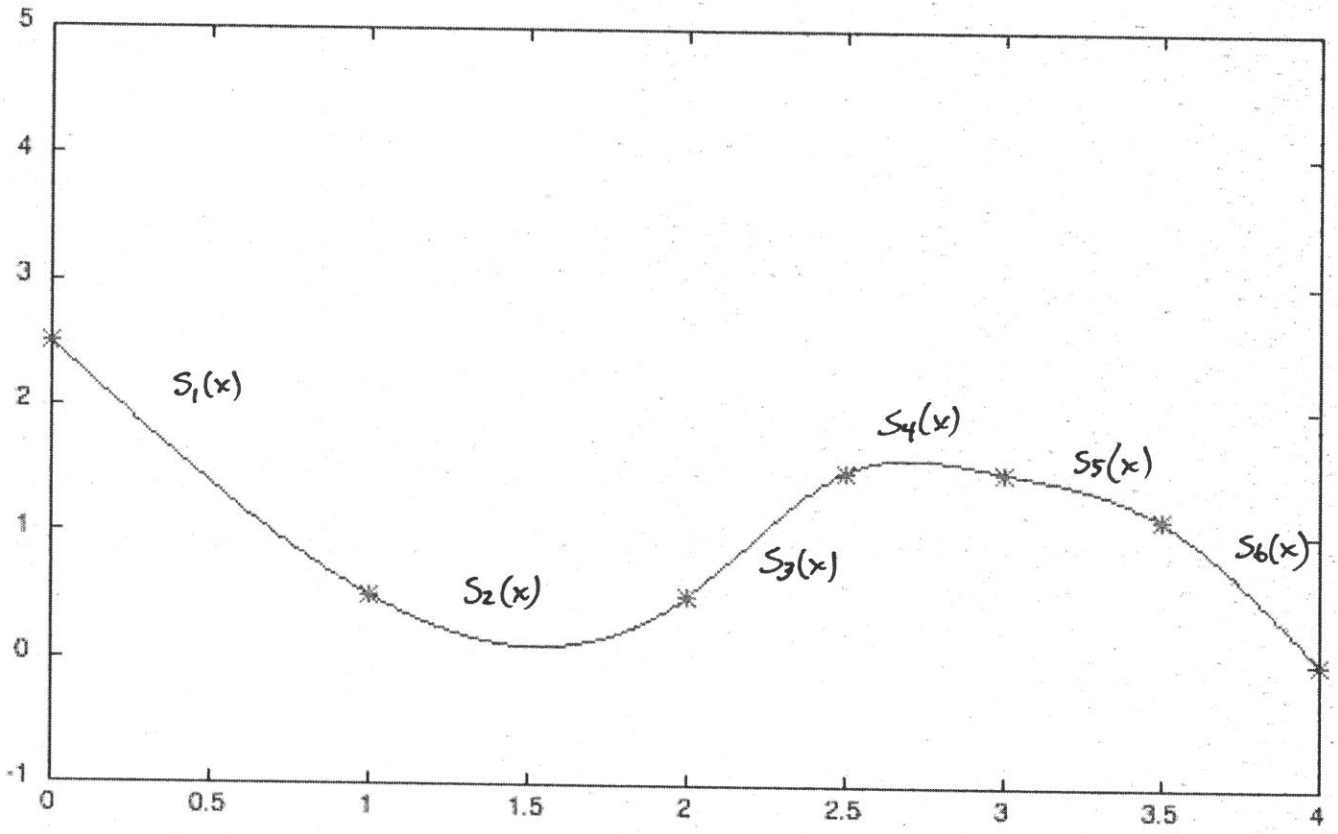
But $M_1 = M_4 = 0$ so $$\begin{bmatrix} 2/3 & 1/6 \\ 1/6 & 2/3 \end{bmatrix} \begin{bmatrix} M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/2 \end{bmatrix}$$

simple brute force — TAKE THE INVERSE.

$M_2 = 1/2$ and $M_3 = 0$

NATURAL CUBIC SPLINE.

test4.jpg.



Then we can construct the spline.

Note. Solving for M_j always results in a TRIDIAGONAL MATRIX.
If node spacing is even ... h ..

Always tridiagonal extra nice for even spacing.

$$A = \begin{bmatrix} 2h/3 & h/6 & 0 & 0 & \cdots \\ h/6 & 2h/3 & \ddots & & \\ 0 & \ddots & \ddots & & 0 \\ 0 & & \ddots & \ddots & \\ \vdots & & & \ddots & h/6 \\ & & & h/6 & 2h/3 \end{bmatrix}$$

when:

$j=2$ $\quad s(x) = \dfrac{(2-x)^3 M_1 + (x-1)^3 M_2}{6(2-1)} + \dfrac{(2-x)1 + (x-1)1/2}{(2-1)}$

$$- 1/6(2-1)\left[(2-x)M_1 + (x-1)M_2\right]$$

$$= \frac{1}{12}x^3 - \frac{1}{4}x^2 - \frac{1}{3}x + \frac{3}{2}$$

$j = 3$ $\quad s(x) = -\frac{1}{12}x^3 + \frac{3}{4}x^2 + \frac{7}{3}x + \frac{17}{6}$

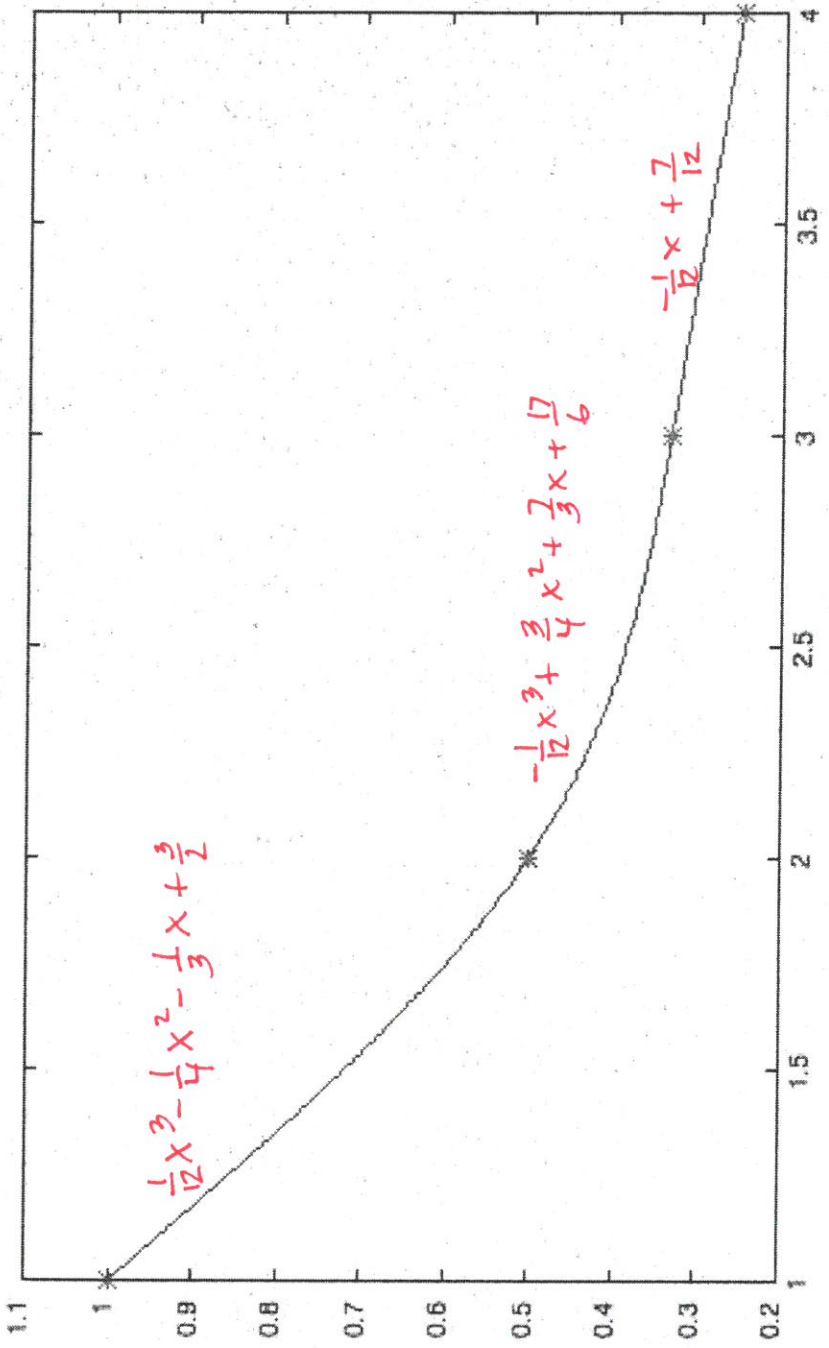
$j = 4$ $\quad s(x) = -\frac{1}{12}x + \frac{7}{12}.$

Put these together for the final answer

$$s(x) = \begin{cases} \frac{1}{12}x^3 - \frac{1}{4}x^2 - \frac{1}{3}x + \frac{3}{2} & 1 \le x < 2 \\ -\frac{1}{12}x^3 + \frac{3}{4}x^2 + \frac{7}{3}x + \frac{17}{6} & 2 \le x < 3 \\ -\frac{1}{12}x + \frac{7}{12} & 3 \le x \le 4 \end{cases}$$

Example spline.jpg.

Cubic Spline (1,1) (2, 1/2) (3, 1/3) (4 (1/4)



Matlab ...

- Input the nodes x_nodes, y_nodes n = # of node points.
- Input the eval points x_eval = a:.01:b
- compute the distances between points

$$h = x\_nodes(2:n) - x\_nodes(1:n-1)$$

- find the diagonal elements

$$D0 = 2(h(1:n-2) + h(2:n-1))./3$$
$$D1 = h(2:n-2)./6$$

- Construct A:

$$A = diag([D0], 0) + diag([D1], -1) + diag([D1], 1)$$

$$A = \begin{bmatrix} D0 & 0 & 0 & \cdots \\ 0 & D0 & 0 & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & D0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ D1 & 0 & & \vdots \\ 0 & D1 & \ddots & \vdots \\ & & \ddots & D1\ 0 \end{bmatrix} + \begin{bmatrix} 0 & D1 & 0 & \cdots \\ \vdots & 0 & D1 & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \end{bmatrix}$$

- Construct b

$$b = [y(3:n) - y(2:n)]./h(2:n-1) - [y(2:n-1) - y(1:n-2)]./h(1:n-2)$$

- Finally

$$M = b * inv(A)$$

- Set the boundary points

$$M = [0\ ;\ z\ ;\ 0];$$

- Find the coefficients/spline. Use a. For loop or implied loop.

BOUNDARY CONDITIONS.

For the natural cubic spline we assumed

$$M_1 = M_n = 0 \quad \underline{or} \quad S''(x_1) = S''(x_n) = 0$$

- no curvature at end points.
- idea comes from the bending beam problem in physics.

Sometimes we need or want other assumptions at the boundary.

EX in our example spline $f(x) = 1/x$
so we know $f''(x) = \frac{2}{x^3}$

we could have set $M_1 = 2$ and $M_4 = 1/32$

What we are doing here is choosing boundary conditions!

"NATURAL" $S''(x_1) = S''(x_n) = 0$ not the best ... can lead to large errors.

"CLAMPED" $S'(x_1) = m_1$ and $S'(x_n) = m_n$

where m_1 and m_n are given slopes at the endpoints.
better that "natural" but we need to know the slope at the endpoints.

"NOT-A-KNOT" $S(x_2) = y_2$ and $S(x_{n-1}) = y_{n-1}$

we don't use the points $(x_2, y_2)(x_{n-1}, y_{n-1})$ in the interpolation, rather we use them as boundary conditions on the left most and right most spline.

In each case we are still left with $n-2$ eqns for $n-2$ unknowns.