- HERMITE INTERPOLATION
  - match $f(x_i)$ & $f'(x_i)$

- <u>SPLINES</u>: § 3.4

A <u>spline function</u> is a function that consists of polynomial pieces joined together w/ certain smoothness conditions.

Points where the function changes character - KNOTS
$N-1$ knots, N spline functions

<u>LINEAR SPLINE</u>

$N+1$ points
$N-1$ knots
N polynomials

<u>2N</u> unknowns $\rightarrow$ 2 unknowns per line $\times$ n lines

Contin of S
$$\begin{cases} S_k(x_k) = y_k & \text{for } k=0,..,N-1 \quad \text{left } \cancel{\text{edge}} \text{ node of line} \\ S_k(x_{k+1}) = y_{k+1} & \text{for } k=0..,N-1 \quad \text{right node of line} \end{cases}$$

<u>2N eqns</u>.

<u>QUADRATIC SPLINES</u>:

<u>3N unknowns</u> — 3 unknowns per quadratic $\times$ n quadratics

Contin of S
$$\begin{cases} S_k(x_k)=y_k, & k=0,..,N-1 \quad N \text{ eq}^n\text{s} \\ S_k(x_{k+1})=y_{k+1}, & k=0,..,N-1 \quad N \text{ eq}^n\text{s} \end{cases}$$

Contin of S'
$$\left\{ S_k'(x_{k+1}) = S_{k+1}'(x_{k+1}), \quad k=0,..,N-2 \quad N-1 \text{ eq}^n\text{s} \right.$$
(interior knots)

Add the eq^n: $S'(x_0)=0$ for example

or $S'(x_n) = f'(x_n)$

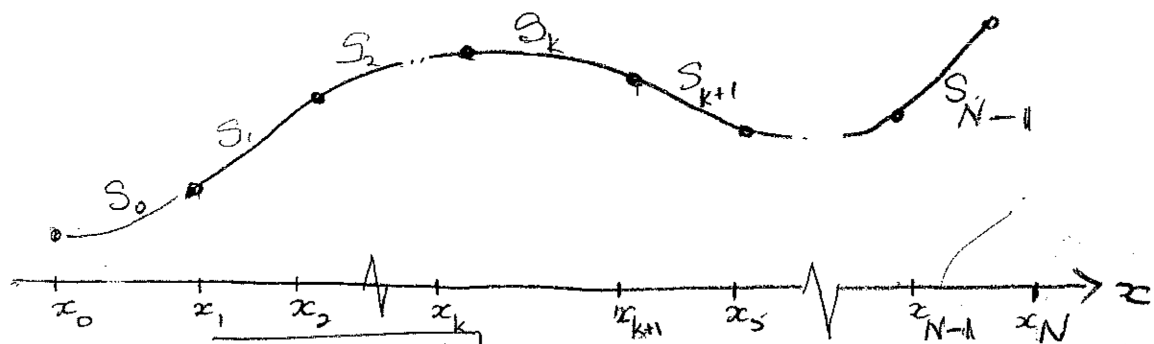
SPLINE

①

CUBIC SPLINE INTERPOLATION

Oscillatory nature of high-degree polynomials restricts their use.

Divide interval $[a, b]$ into subintervals and use approx. polynomial on each subinterval — PIECEWISE POLYNOMIAL APPROXIMATION



splines: $\boxed{4N \text{ unknowns}}$ — 4 unknowns/cubic $\times N$ cubics

CONTIN OF S

$S_k(x_k) = y_k$ For $k = 0, 1, 2, \ldots, N-1$ (N eq^{n}s)

$S_k(x_{k+1}) = y_{k+1}$ For $k = 0, 1, 2, \ldots, N-1$ (N eq^{n}s)

CONTIN OF S' $\quad S_k'(x_{k+1}) = S_{k+1}'(x_{k+1})$ For $k = 0, 1, 2, \ldots, N-2$ $(N-1)$ eq^{n}s

interior knots

CONTIN OF S'' $\quad S_k''(x_{k+1}) = S_{k+1}''(x_{k+1})$ For $k = 0, 1, 2, \ldots, N-2$ $(N-1)$ eq^{n}s

$4N-2$ constraints — Need 2 more

Add : $S''(x_0) = 0$; $S''(x_N) = 0$ FREE OR NATURAL BDRY

or

$S'(x_0) =$ given ; $S'(x_N) =$ given CLAMPED BDRY

FREE / NATURAL Forcing a flexible elastic rod through the data pts but letting the slope at the ends be free to equilibrate to the position that minimizes oscillatory behaviour of curve.

SPLINE

(2)

PTO

Why Cubic Splines?

- Discontinuities in the third derivative cannot be detected visually.
- Experience has shown that using splines of degree > 3 seldom yields any advantage.

Cubic Spline Smoothness

If S is the natural cubic spline function that interpolates $f \in \mathcal{C}^2$ at the knots, then

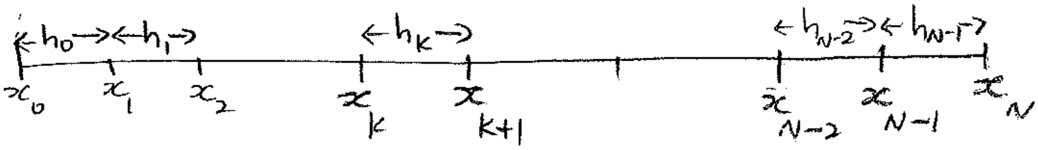
$$\int_a^b \left[S''(x)\right]^2 dx \leq \int_a^b \left[f''(x)\right]^2 dx$$

Avg Curvature of spline S is no larger than average curvature of any $\mathcal{C}^2$ function which agrees w/ S at the knots

Oscillation $\Rightarrow$ rapid change in first deriv $\Rightarrow$ large 2nd deriv.
Cubic splines have "controlled" 2nd derivs.

Look at polyshow.m
splinetx.m
pchip.m

- Changing data point has a global effect in that the entire curve is affected.



Given data (x_k, y_k) for $k=0,1,\cdots,N-1$

Find $S_k(x) = a_k + b_k(x-x_k) + c_k(x-x_k)^2 + d_k(x-x_k)^3$

which we use for $x_k \le x \le x_{k+1}$

for $k=0,1,\cdots,N-1$

<u>i.e.</u> Find the $4N$ unknowns a_k, b_k, c_k, d_k for $k=0,1,\cdots,N-1$

subject to the cubic spline conditions.

(i) $S_k(x_k) = y_k$ for $k=0,1,\cdots,N-1$

$\Longrightarrow$ $\boxed{a_k = y_k \text{ for } k=0,1,\cdots,N-1}$ Equation (i)

(ii) $S_k(x_{k+1}) = y_{k+1}$ for $k=0,1,\cdots,N-1$

$\Longrightarrow$ $\boxed{a_k + h_k b_k + h_k^2 c_k + h_k^3 d_k = y_{k+1} \text{ for } k=0,1,\cdots,N-1}$ Equation (ii)

(iii) $S_k'(x) = b_k + 2c_k(x-x_k) + 3d_k(x-x_k)^2$

So $S_k'(x_{k+1}) = S_{k+1}'(x_{k+1})$ for $k=0,1,\cdots,N-2$

$\Longrightarrow$ $\boxed{b_k + 2c_k h_k + 3d_k h_k^2 = b_{k+1} \text{ for } k=0,1,\cdots,N-2}$

Equation (iii)

(iv) $S_k''(x) = 2c_k + 6d_k(x-x_k)$

So $S_k''(x_{k+1}) = S_{k+1}''(x_{k+1})$ for $k=0,1,\cdots,N-2$

Equation (iv)

$\Longrightarrow$ $2c_k + 6d_k h_k = 2c_{k+1}$ $\Longrightarrow$ $\boxed{c_k + 3d_k h_k = c_{k+1}}$ for $k=0,\cdots,N-2$

and $k=N-1$

Define $c_N \equiv \frac{1}{2}S''(x_N)$, so since $S_{N-1}''(x_N) = S''(x_N)$

$\Longrightarrow 2c_{N-1} + 6d_{N-1}h_{N-1} = 2c_N$

So $\boxed{d_k = \dfrac{c_{k+1}-c_k}{3h_k} \quad \text{for } k=0,1,\dots,N-2,N-1.}$ eqn (iv')

where $c_N \equiv \frac{1}{2}S''(x_N)$

So if you know c_k's you can get your d_k's

Putting (iv') & (i) in (ii) gives

$$y_k + h_k b_k + h_k^2 c_k + h_k^2 \frac{c_{k+1}-c_k}{3} = y_{k+1} \quad \text{for } k=0,1,\dots,N-1$$

$$\Longrightarrow \boxed{b_k = \frac{y_{k+1}-y_k}{h_k} - \left(\frac{c_{k+1}+2c_k}{3}\right)h_k \quad \text{for } k=0,1,\dots,N-1}$$

(eqn ii')

So if you know your c_k's you can get your b_k's

Eqn (iii) is $b_k + 2c_k h_k + 3d_k h_k^2 = b_{k+1}$ for $k=0,1,\dots,N-2$

using (ii') using (iv') using (ii')

$$\left[\frac{y_{k+1}-y_k}{h_k} - \left(\frac{c_{k+1}+2c_k}{3}\right)h_k\right] + 2c_k h_k + \left[h_k(c_{k+1}-c_k)\right] = \frac{y_{k+2}-y_{k+1}}{h_{k+1}} - \left(\frac{c_{k+2}+2c_{k+1}}{3}\right)h_k$$

$$\Longrightarrow \frac{h_k}{3}c_k + \frac{2}{3}(h_k+h_{k+1})c_{k+1} + \frac{h_{k+1}}{3}c_{k+2} = \frac{y_{k+2}-y_{k+1}}{h_{k+1}} - \frac{y_{k+1}-y_k}{h_k}$$

for $k=0,1,\dots,N-2$

These are $N-1$ ~~linear~~ eqns involving $N+1$ unknowns $c_0, c_1, \dots, c_{N-1}, c_N$

Add 2 more conditions — eg Boundary Conditions — and problem may give unique solution

$$h_k c_k + 2(h_k + h_{k+1})c_{k+1} + h_{k+1}c_{k+2} = \frac{3}{h_{k+1}}(y_{k+2} - y_{k+1}) - \frac{3}{h_k}(y_{k+1} - y_k) \quad ✡$$

$$h_0 c_0 + 2(h_0 + h_1)c_1 + h_1 c_2 = \frac{3}{h_1}(y_2 - y_1)$$

for $k = 0, 1, \ldots, N-2$

$N-1$ eqns in $N+1$ unknowns $c_0, c_1, \ldots, c_N$

where $c_N = \frac{1}{2}S''(x_N)$.

(In general, $c_k = \frac{1}{2}S''(x_k)$)

Need 2 more conditions.

What will things look like in matrix-vector form.

Solve $A\vec{x} = \vec{b}$

$$\begin{array}{c} \\ N-1 \text{ ✡ eqn} \\ \\ \end{array}
\begin{bmatrix} * & & & & & \\ h_0 & 2(h_0+h_1) & h_1 & 0 & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} & \\ & & * & & * & * \end{bmatrix}
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_N \end{bmatrix} \quad N+1 \text{ eqns}$$

$$= 3\begin{bmatrix} * \\ \dfrac{y_2 - y_1}{h_1} - \dfrac{y_1 - y_0}{h_0} \\ \dfrac{y_3 - y_2}{h_2} - \dfrac{y_2 - y_1}{h_1} \\ \vdots \\ \dfrac{y_N - y_{N-1}}{h_{N-1}} - \dfrac{y_{N-1} - y_{N-2}}{h_{N-2}} \\ * \end{bmatrix}$$

The other 2 conditions will complete the $(N+1)\times(N+1)$ matrix-vector system.

```
function [v,sigma] = splinetx(x,y,u)
%SPLINETX  Textbook spline function.
%  v = splinetx(x,y,u) finds the piecewise cubic interpolatory
%  spline S(x), with S(x(j)) = y(j), and returns v(k) = S(u(k)).
%  [v,sigma] = splinetx(...) also returns sigma(k) = S''(x(k))/6.
%
%  Set up tridiagonal system.
%  h = offdiagonal, d = diagonal, b = right hand side.
%
   h = diff(x);
   n = length(x);
   k = (2:n-1)';
   d = zeros(n,1);
   d(k) = 2*(h(k-1) + h(k));
   b = zeros(n,1);
   b(k) = diff(diff(y)./h);
   A = diag(h,-1) + diag(d,0) + diag(h,1);
%
%  Not-a-knot end conditions.
%
   A(1,1:3) = [h(2) -(h(2)+h(1)) h(1)];
   A(n,n-2:n) = [h(n-1) -(h(n-2)+h(n-1)) h(n-2)];
%
%  Solve tridiagonal system.
%
   sigma = A\b;
%
%  Find interval indices.
%
   k = ones(size(u));
   for j = 2:n-1
      k(u > x(j)) = j;
   end
%
%  Evaluate spline.
%
   s = (u - x(k))./h(k);
   t = 1-s;
   sigma = reshape(sigma,size(y));
   v = s.*y(k+1) + t.*y(k) + ...
       ((s.^3-s).*sigma(k+1) + (t.^3-t).*sigma(k)).*h(k).^2;
```