Ch 3.5: Splines

Friday, October 17, 2025 11:04 AM

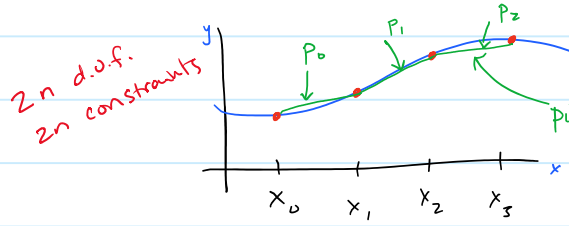
In our previous discussions of interpolation, we considered a single polynomial,

but if we have many points $\{x_0, x_1, \cdots, x_n\}$, (we'll ALWAYS assume these are distinct) then we need a high-degree polynomial which can be undesirable (e.g., Runge phenomenon if equispaced nodes)

A very popular alternative is to do a piecewise polynomial interpolation

Ex: piecewise linear

The mathematics are straightforward so we won't focus on



p_i: polynomials
s is piecewise polynomial

$$s = \begin{cases} p_1 & \text{if } x \in [x_0, x_1) \\ p_2 & x \in [x_1, x_2) \\ p_3 & x \in [x_2, x_3] \end{cases}$$

In Matlab/numpy, this is "interp1"

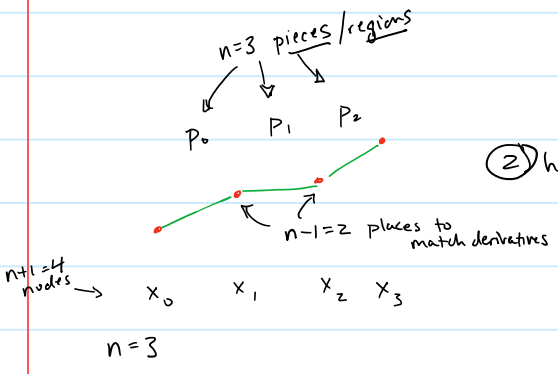
Downside of piecewise linear is that the approximation is not differentiable, ie., derivative on one piece doesn't match derivative on neighboring piece.

$$s(x) = \{ p_j(x) \text{ if } x \in [x_j, x_{j+1}) \}$$ i.e.,

① interpolation requirement $s(x_i) = f(x_i)$, $i = 0, 1, \cdots, n$

meaning

$$\begin{aligned} p_i(x_i) &= f(x_i) \\ p_i(x_{i+1}) &= f(x_{i+1}) \end{aligned} \quad i = 0, 1, \cdots, n-1 \Big\} \; 2n \text{ constraints}$$



(So for n piecewise linear polynomials, we have $2n$ parameters, which matches $2n$ constraints ✓)

② have slopes agree on interior nodes $x_1, \cdots, x_{n-1}$ (exclude x_0, x_n)

meaning

$$p_i'(x_{i+1}) = p_{i+1}'(x_{i+1}), \quad i = 0, 1, \cdots, n-2 \Big\} \; n-1 \text{ constraints}$$

Impossible w/ piecewise linear!

Going beyond linear interpolation,

what if we want to impose conditions ① and ② ?

that's $2n + (n-1) = 3n-1$ constraints.

Suppose we use a quadratic rather than linear polynomial on each region?

Then $3n$ parameters but only $3n-1$ constraints so it's not unique unless we specify one more constraint. A boundary condition is natural, but only have room for one (x_0 or x_n, not both) ... AWKWARD!

Friday, October 17, 2025 11:08 AM

Suppose we use a cubic polynomial? Then we have $4n$ parameters. We can add a lot of constraints. Let's add

(3) Have second derivatives agree on interior nodes

$$p_i''(x_{i+1}) = p_{i+1}''(x_{i+1}), \quad i = 0, 1, \ldots, n-2 \quad \Big\} \; n-1 \text{ constraints}$$

Then (1) + (2) + (3) is $2n + (n-1) + (n-1) = 4n-2$ constraints

So we have 2 extra degrees of freedom. This is a nice symmetric case. We can impose

(4a) "Natural" or "Free" Boundaries,

$$S''(x_0) = 0 \qquad \text{i.e., no curvature at end}$$
$$S''(x_n) = 0 \qquad \text{i.e., } p_0''(x_0) = 0, \; p_{n-1}''(x_n) = 0$$

all useful. we'll focus on 4' "Natural"

or

(4b) "clamped boundary" — of course only works if we can compute f'

$$S'(x_0) = f'(x_0) \qquad \text{i.e., } p_0'(x_0) = f'(x_0)$$
$$S'(x_n) = f'(x_n) \qquad p_{n-1}'(x_n) = f'(x_n)$$

or

(4c) "not-a-knot"

S''' continuous at x_1 and x_{n-1}

i.e., $p_0'''(x_1) = p_1'''(x_1)$

$p_{n-2}'''(x_{n-1}) = p_{n-1}'''(x_{n-1})$

"knot" is synonymous with "node", meaning a x_i point

or

(4d) "periodic" $\quad S'(x_0) = S'(x_n)$, $S''(x_0) = S''(x_n)$

⚠ Condition (2) says $S \in C^1([a,b])$, i.e., derivatives match each other

Not the same as piecewise Hermite interpolation, which would require derivatives of S to match those of f

Conditions (1) + (2) + (3) + ((4a) or (4c) or (4d)) do not require knowing f'

Friday, October 17, 2025 11:09 AM

SPLINES

A piecewise function satisfying conditions like ①, ② and ③ is called a spline.

(Note: B-Splines are a bit different, since don't interpolate, but have advantage that if you move a node, you don't have to recompute from scratch. Used in NURBS for CAD (computer-aided design) programs. These and Bézier curves (used in graphics, eg. fonts, vector graphics) are beyond the scope of this course)

vector vs. raster (∞ resolution) (JPEG)

We'll talk exclusively about CUBIC SPLINES ① + ② + ③ + ④

and in particular NATURAL CUBIC SPLINES ① + ② + ③ + ④a

⚠ interp1 (Matlab and Scipy) takes a "cubic" option but doesn't satisfy ③.

To get a cubic spline, scipy.interpolate.CubicSpline (call boundary conditions) (python)

or interp1 w/ "spline" option } "not-a-knot" (Matlab)

or spline

or csape / csapi (Matlab curve-fitting toolbox)

So... we have $4n$ degrees of freedom, $4n$ constraints

Basic technique: Fix a basis (representation) for the polynomials in each interval, setup a $4n \times 4n$ linear system, and solve in $O(n^3)$ time.

Tricks: pick a good representation for the polynomials (for stability) and tricks to solve system in $O(n)$ time

1) How to represent $p_j(x)$

ex. $p_j(x) = a_j + b_j x + c_j x^2 + d_j x^3$ $\quad j = 0, 1, \dots, n-1$

better: $p_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$

... so clearly $a_j = f(x_j)$

Friday, October 17, 2025 11:09 AM

2) Setup matrix system

$\Rightarrow \boxed{a_j = f(x_j)}$ ✓ (*)

Recall:

(1) 1a) $P_j(x_j) = f(x_j)$

1b) $P_j(x_{j+1}) = f(x_{j+1})$ $\quad j = 0, 1, \dots, n-1$

(2) $P_j'(x_{j+1}) = P_{j+1}'(x_{j+1})$ $\quad j = 0, 1, \dots, n-2$

(3) $P_j''(x_{j+1}) = P_{j+1}''(x_{j+1})$ $\quad j = 0, 1, \dots, n-2$

(4a) $P_0''(x_0) = 0$

$P_{n-1}''(x_n) = 0$

Note $P_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$, $P_j''(x) = 2c_j + 6d_j(x - x_j)$

So $P_j'(x_j) = b_j$ (**) and likewise $P_j''(x_j) = 2c_j$ (***)

Notation: $h_j = \Delta x_j$, i.e., $h_j = x_{j+1} - x_j$

Note 1b can now be written $P_j(x_{j+1}) = f(x_{j+1}) = P_{j+1}(x_{j+1})$

$j = 0, 1, \dots, n-1$ (A) $a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 = a_{j+1}$ $\quad j = 0, 1, \dots, n-1$

via (*) (w/ $a_n := f(x_n)$)

And (2), with (**), gives

$P_j'(x_{j+1}) = P_{j+1}'(x_{j+1})$

$j = 0, 1, \dots, n-1$ (B) $b_j + 2c_j h_j + 3d_j h_j^2 = b_{j+1}$ via (**) (w/ $b_n = s'(x_n)$)

And (3), with (***), gives

$P_j''(x_{j+1}) = P_{j+1}''(x_{j+1})$

$j = 0, 1, \dots, n-1$ $\quad 2c_j + 6d_j h_j = 2c_{j+1}$ via (***) (w/ $c_n = s''(x_n)/2$)

i.e., (C) $c_j + 3d_j h_j = c_{j+1}$

$\Rightarrow d_j = \dfrac{c_{j+1} - c_j}{3h_j}$

Friday, October 17, 2025 11:11 AM

Plug into

$j = 0, 1, \ldots, n-1$

(A) $a_{j+1} = a_j + b_j h_j + h_j^2/3 (2c_j + c_{j+1})$

and

(B) $b_{j+1} = b_j + h_j (c_j + c_{j+1})$

Recall a_j are known!

you don't need to be able to recreate this on an exam

So $b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$ $(j = 0, 1, 2, \ldots, n-1)$

and $b_{j+1} = \frac{1}{h_{j+1}}(a_{j+2} - a_{j+1}) - \frac{h_{j+1}}{3}(2c_{j+1} + c_{j+2})$ $(j = -1, 0, 1, \ldots, n-2)$

* both hold for $j = 0, \ldots, n-2$

So now (B) becomes (after grouping terms)

* $j = 0, 1, \ldots, n-2$ $\quad h_j c_j + 2(h_j + h_{j+1}) c_{j+1} + h_{j+1} c_{j+2} = \underbrace{\frac{3}{h_{j+1}}(a_{j+2} - a_{j+1}) - \frac{3}{h_j}(a_{j+1} - a_j)}_{\text{known!}}$

So... Solve these $(n-1)$ equations for c

then plug into earlier equations to find b and d.

Also need boundary condition (4a) $p_0''(x_0) = 0$ and $p_{n-1}''(x_n) = 0$

$p_i''(x) = 2c_i + 6d_i(x - x_i)$

So $p_0''(x_0) = 2c_0$, so $p_0''(x_0) = 0$ means $c_0 = 0$.

also, recall we defined $c_n = s''(x_n)/2$, so $s''(x_n) = 0 \Rightarrow c_n = 0$.

Thus:

$$
\begin{aligned}
c_0 &= 0 && \equiv y_0 \\
h_0 c_0 + 2(h_0 + h_1) c_1 + h_1 c_2 &= \tfrac{3}{h_1}(a_2 - a_1) - \tfrac{3}{h_0}(a_1 - a_0) && \equiv y_1 \\
h_1 c_1 + 2(h_1 + h_2) c_2 + h_2 c_3 &= \tfrac{3}{h_2}(a_3 - a_2) - \tfrac{3}{h_1}(a_2 - a_1) && \equiv y_2 \\
&\vdots \\
h_{n-2} c_{n-2} + 2(h_{n-2} + h_{n-1}) c_{n-1} + h_n c_n &= \tfrac{3}{h_{n-1}}(a_n - a_{n-1}) - \tfrac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) && \equiv y_{n-1} \\
c_n &= 0 && \equiv y_n
\end{aligned}
$$

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & \cdots & 0 \\
h_0 & 2(h_0+h_1) & h_1 & 0 & & \\
0 & h_1 & 2(h_1+h_2) & h_2 & & \\
\vdots & & \ddots & \ddots & \ddots & 0 \\
 & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\
0 & \cdots & & 0 & 0 & 1
\end{bmatrix}
\cdot
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}
=
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}
$$

This is a system of $(n+1)$ unknowns and $(n+1)$ equations

Friday, October 17, 2025 11:11 AM

This is a system of (n+1) unknowns and (n+1) equations

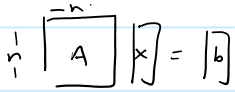
① Fact: it's invertible

(proof: it's "diagonally dominant", cf. ch 6.6, and by the Gershgorin Disk theorem, there are aka Gershgorin Circle no zero eigenvalues... a special case is proven in Thm. 6.21)

② It's a tridiagonal matrix, and can be solved efficiently

A fact* everyone should know

Specifically, note that solving a general system of n linear equations and n unknowns takes $O(n^3)$ flops



* it's not quite true due to fast Strassen based schemes, but it's true enough

But if you know A is diagonal, now it's uncoupled, so only $O(n)$ flops

If it's lower (or upper triangular), it's $O(n^2)$ flops

It turns out tridiagonal matrices can be solved in $O(n)$ flops, eg. "Thomas Algorithm" or "Crout Factorization" (Algo 6.7 in our book)

in Matlab/Scipy, using a sparse structure will already make it pretty fast to solve

Theory

Thm 3.11 There is a unique natural (4a) cubic spline interpolant

proof we just proved this when we showed the above tridiagonal matrix is invertible

Thm 3.12 If f is differentiable at x_0 and x_n, then there is a unique clamped (4b) cubic spline interpolant

proof similar

Friday, October 17, 2025 11:12 AM

Error Bounds

Thm 3.13 If $f \in C^4([a,b])$ with $\max_{x\in[a,b]} |f^{(4)}(x)| = M$ and

S is the clamped cubic spline interpolant on

$a = x_0 < x_1 < x_2 < \dots < x_n = b$ then

$(\forall x \in [a,b])$

$$|f(x) - S(x)| \le M \cdot \frac{5}{384} \cdot \max_{0\le j\le n-1} (x_{j+1} - x_j)^4.$$

(similar results hold for natural and not-a-knot cubic splines)

Interpretation of Thm 3.13

Consider uniformly spaced points

$x_0 = a, \quad x_1 = a+h, \quad x_2 = a+2h, \quad \dots, \quad x_n = a+nh = b$

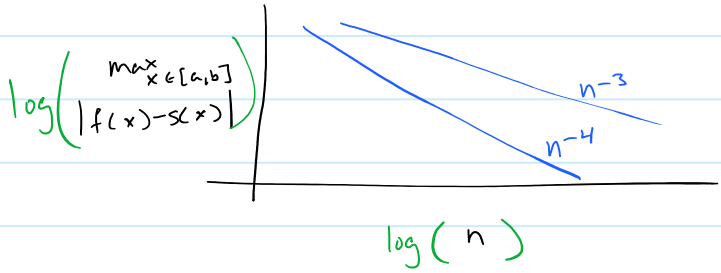
so $h = \frac{b-a}{n}$

then the theorem says the error is

$$|f(x) - S(x)| \le \text{constant} \cdot h^4$$

i.e. $\le$ constant $\cdot\, n^{-4}$

We say it is 4th order accurate



$y = n^{\alpha}$ is a straight-line on a log-log plot

top 10 fact you should know after this course

Summary

Cubic Splines are piece-wise cubic, C^2 functions that interpolate on the nodes, are 4th order accurate, and require $O(n)$ computation for $n+1$ nodes.