Lecture `splines`: Introduction to Cubic Splines

October 24, 2022

Summary: Motivation for and construction of piecewise cubic splines, with emphasis on the associated linear systems. Mostly taken from C. F. Van Loan's *Introduction to Scientific Computing*.

References: T. Sauer's *Numerical Analysis*, Section 3.4, first edition, pages 170-181.

Introduction

Splines[1] are piecewise defined curves which give a good fit to a set of data points. By "good fit" here, we mean visually pleasing, as typically with splines accuracy is not the main concern. By way of contrast, for high–order interpolating polynomials, one is concerned with accuracy (pointwise or global as discussed in class). *Cubic splines* are twice–continuously differentiable curves which are defined piecewise with cubic polynomials.

Piecewise Cubic Hermite Interpolation

Before turning to splines, we consider the following problem: given a collection of data $D_N = \{(x_i, y_i, s_i)\}$ which represents samples of a function $f(x)$ and its derivatives $f'(x)$ [that is, $f(x_i) = y_i$ and $f'(x_i) = s_i$], find a series of $N-1$ cubic functions $C_i(x)$ such that

$$C(x) = \begin{cases} C_1(x) & \text{if } x \in [x_1, x_2) \\ C_2(x) & \text{if } x \in [x_2, x_3) \\ & \vdots \\ C_{N-1}(x) & \text{if } x \in [x_{N-1}, x_N] \end{cases} \tag{1}$$

satisfying, for every $i = 1, 2, \cdots N-1$,

$$\begin{aligned} C_i(x_i) &= y_i, \quad C_i(x_{i+1}) = y_{i+1} \\ C_i'(x_i) &= s_i, \quad C_i'(x_{i+1}) = s_{i+1}. \end{aligned} \tag{2}$$

For each $C_i(x)$ we have four equations (the set above) and four unknowns (a cubic polynomial is determined by four expansion coefficients in whatever basis you choose, for example monomials: $c_1 + c_2 x + c_3 x^2 + c_4 x^3$). So we expect to uniquely determine the $C_i(x)$ for all

[1] According to A. Greenbaum and T. P. Chartier (see page 201 of *Numerical Methods*, Princeton University Press, 2012), *splines* were originally the thin wooden strips passed through key interpolation points in order to construct templates for airplanes during World War II.

$i = 1, \cdots, N-1$. Note that each $C_i(x)$ interpolates both function y_i, y_{i+1} and derivative s_i, s_{i+1} values; whence $C_i(x)$ is a Hermite interpolant. *Hermite interpolation* generalizes ordinary interpolation to include derivative values.

Let us consider just one $C_i(x)$ and its associated interval $[x_i, x_{i+1}]$. To make the notation nicer, consider instead the interval $[x_L, x_R]$ with the data (x_L, y_L, s_L) and (x_R, y_R, s_R). Also define the first-order Newton divided difference $y[x_L, x_R] = (y_R - y_L)/(x_R - x_L)$. Then our cubic interpolant takes the form

$$\begin{aligned}\mathcal{C}(x) &= y_L + y[x_L, x_R](x - x_L) \\ &+ \frac{y[x_L, x_R] - s_L}{x_R - x_L}(x - x_L)(x - x_R) \\ &+ \frac{s_R + s_L - 2y[x_L, x_R]}{(x_R - x_L)^2}(x - x_L)^2(x - x_R). \end{aligned} \tag{3}$$

$\mathcal{C}(x)$ is tailored to satisfy $\mathcal{C}(x_L) = y_L$, $\mathcal{C}(x_R) = y_R$, $\mathcal{C}'(x_L) = s_L$, and $\mathcal{C}'(x_R) = s_R$, which you can check explicitly by hand. Translating this expression to the general interval $[x_i, x_{i+1}]$ and using $\Delta x_i = x_{i+1} - x_i$, we get

$$\begin{aligned}C_i(x) &= y_i + y[x_i, x_{i+1}](x - x_i) \\ &+ \frac{y[x_i, x_{i+1}] - s_i}{\Delta x_i}(x - x_i)(x - x_{i+1}) \\ &+ \frac{s_{i+1} + s_i - 2y[x_i, x_{i+1}]}{\Delta x_i^2}(x - x_i)^2(x - x_{i+1}), \end{aligned} \tag{4}$$

which satisfies the conditions (2).

Splines

Once the data $\{(x_i, y_i, s_i)\}_{i=1}^N$ is given, the piecewise cubic interpolant $C(x)$ is completely determined. However, specifying the derivative data $\{s_i\}_{i=1}^N$ at the nodes x_i, or *knots* as they are often called when considering splines, is awkward. Indeed, mostly we just have data $\{(x_i, y_i)\}_{i=1}^N$, without the $\{s_i\}_{i=1}^N$ given at all. Therefore, we now shift our emphasis and *solve for the derivative data*. The solution $\{s_i\}_{i=1}^N$ along with the specified data $\{(x_i, y_i)\}_{i=1}^N$ then determine each $C_i(x)$ and so $C(x)$. How do we solve for the $\{s_i\}_{i=1}^N$?

We are searching for N values s_i, and so need N equations to define these N unknowns. We first enforce continuity of the second derivative explicitly at each of the $N-2$ points $x_2, x_3, \cdots, x_{N-1}$. It is these $N-2$ conditions which make our piecewise defined $C(x)$ a *spline*. They constitute $N-2$ linear equations (we still need two more!), and when enforced they ensure that, as a whole, the function $C(x)$ is twice continuously differentiable. Using Eq. (4), we express the equation $C_i''(x_{i+1}) = C_{i+1}''(x_{i+1})$ as follows:

$$\Delta x_{i+1} s_i + 2(\Delta x_i + \Delta x_{i+1}) s_{i+1} + \Delta x_i s_{i+2} = 3\left(\Delta x_{i+1} y[x_i, x_{i+1}] + \Delta x_i y[x_{i+1}, x_{i+2}]\right), \tag{5}$$

but leave the details to an appendix at the end. We stress that the unknowns in this equation are the s_i: we know the Δx_i, and the divided differences $y[\bullet]$ may be calculated from the given data $\{(x_i, y_i)\}_{i=1}^N$. Eq. (5) is enforced for all $i = 1, 2, \cdots N-2$.

Let us count equations and unknowns: in order to form the piecewise cubic Hermite interpolant, we need N values for the s_i. We have $N-2$ linear equations in the s_i. So we are still missing 2 equations. Before considering these two equations, we explore the type of system that we have come up with so far. We consider the $N-2$ equations for the s_i as being equations $2, \cdots, N-1$ in an N–by–N system. The unknowns are the s_i, and the system matrix now reads

$$A = \begin{pmatrix} ? & ? & ? & & \cdots & \cdots & ? \\ \Delta x_2 & 2(\Delta x_1 + \Delta x_2) & \Delta x_1 & & & & \\ 0 & \Delta x_3 & 2(\Delta x_2 + \Delta x_3) & \Delta x_2 & & & \\ & & & & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & & & \\ & & & & \Delta x_{N-1} & 2(\Delta x_{N-2} + \Delta x_{N-1}) & \Delta x_{N-2} \\ ? & \cdots & \cdots & & ? & ? & ? \end{pmatrix}. \tag{6}$$

Entries which are not shown are zero, so for rows 2 through $N-2$ the matrix is tridiagonal. Again, our unknowns are

$$\boldsymbol{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_{N-1} \\ s_N \end{pmatrix},$$

and the entire system reads

$$A\boldsymbol{s} = \begin{pmatrix} ? \\ 3\left(\Delta x_1 y[x_2, x_3] + \Delta x_2 y[x_1, x_2]\right) \\ 3\left(\Delta x_2 y[x_3, x_4] + \Delta x_3 y[x_2, x_3]\right) \\ \vdots \\ 3\left(\Delta x_{N-2} y[x_{N-1}, x_N] + \Delta x_{N-1} y[x_{N-2}, x_{N-1}]\right) \\ ? \end{pmatrix} \tag{7}$$

We have questions marks placed in the first and last rows of the matrix A and the right-hand side vector because these rows correspond to precisely the two unknown equations we have not yet specified. It is these last two degrees of freedom which specify what flavor of spline we have. Below are some of the most common techniques for completing the system.

• **Complete** or **clamped spline**: Here we simply explicitly enforce the two endpoint slope degrees of freedom:

$$s_1 = \alpha, \qquad s_N = \beta,$$

with α and β prescribed values. These two equations fill the first and last rows of (7).

• **Natural spline**: We form this spline by enforcing the second derivatives values of the cubics $C_1(x)$ and $C_{N-1}(x)$ at the endpoints. In other words, we enforce

$$C_1''(x_1) = \gamma, \qquad C_{N-1}''(x_N) = \delta$$

for prescribed values γ and δ. With (3), we can explicitly calculate $C_1''(x_1)$ and $C_{N-1}''(x_N)$ to enforce these conditions. The end result is the pair of equations

$$\begin{aligned} 2s_1 + s_2 &= 3y[x_1, x_2] - \tfrac{1}{2}\Delta x_1 \gamma \\ s_{N-1} + 2s_N &= 3y[x_{N-1}, x_N] + \tfrac{1}{2}\Delta x_{N-1}\delta. \end{aligned}$$

These two equations then serve as the first and last rows for the system (7).

• **Not-a-knot spline**: In the absence of other "boundary conditions", we may simply impose continuity of the *third* derivative between C_1 and C_2 and between C_{N-2} and C_{N-1}. That is,

$$C_1'''(x_2) = C_2'''(x_2), \qquad C_{N-2}'''(x_{N-1}) = C_{N-1}'''(x_{N-1}).$$

These conditions translate into the following set of equations:

$$\begin{aligned} \Delta x_2^2 s_1 + (\Delta x_2^2 - \Delta x_1^2)s_2 - \Delta x_1^2 s_3 &= 2\Delta x_2^2 y[x_1, x_2] - 2\Delta x_1^2 y[x_2, x_3] \\ \Delta x_{n-1}^2 s_{n-2} + (\Delta x_{n-1}^2 - \Delta x_{n-2}^2)s_{n-1} - \Delta x_{n-2}^2 s_n &= 2\Delta x_{n-1}^2 y[x_{n-2}, x_{n-1}] - 2\Delta x_{n-2}^2 y[x_{n-1}, x_n]. \end{aligned}$$

When these equations are used to close the system (7), the result is a linear system which is not quite tridiagonal. Why are these conditions called **not-a-knot**? At the interface x_{i+i} where $C_i(x)$ and $C_{i+1}(x)$ are matched, we have required continuity in the zeroth, first, and second derivatives. This matching does not, however, mean that $C_i(x)$ and $C_{i+1}(x)$ are the same cubic polynomial; there can be mismatch in the third derivative. Indeed, suppose $C_i(x) = x^3$ and $C_{i+1}(x) = 2x^3$, which match at $x_{i+1} = 0$ through the second derivative only. They are different cubic functions, and their *third* derivatives are not equal at $x_{i+1} = 0$. Therefore, in general such an x_{i+1} is called a **knot**, which indicates a "rough point" where we pass from one cubic polynomial $C_i(x)$ to different cubic $C_{i+1}(x)$. However, the first **not-a-knot** condition implies that $C_1(x)$ is the *same* cubic as $C_2(x)$. Indeed, it turns out that matching two cubic polynomials through their *third* derivatives at a point ensures that they are actually the same cubic polynomial. So in this case, the point x_2 is not a "rough" point, i.e. it is not a knot. The same comments apply to x_{N-1}.

Appendix: derviation of (5)

Straightforward differentiation of the formula (4) for $C_i(x)$ yields

$$\frac{1}{2}C_i''(x) = \frac{y[x_i, x_{i+1}] - s_i}{\Delta x_i} + \frac{s_{i+1} + s_i - 2y[x_i, x_{i+1}]}{\Delta x_i^2}(3x - 2x_i - x_{i+1}),$$

and sending $i \to i+1$ in this formula gives

$$\frac{1}{2}C_{i+1}''(x) = \frac{y[x_{i+1}, x_{i+2}] - s_{i+1}}{\Delta x_{i+1}} + \frac{s_{i+2} + s_{i+1} - 2y[x_{i+1}, x_{i+2}]}{\Delta x_{i+1}^2}(3x - 2x_{i+1} - x_{i+2}).$$

With the last two formulas,

$$\frac{1}{2}C_i''(x_{i+1}) = \frac{y[x_i, x_{i+1}] - s_i}{\Delta x_i} + \frac{s_{i+1} + s_i - 2y[x_i, x_{i+1}]}{\Delta x_i^2}(2\Delta x_i)$$
$$\frac{1}{2}C_{i+1}''(x_{i+1}) = \frac{y[x_{i+1}, x_{i+2}] - s_{i+1}}{\Delta x_{i+1}} + \frac{s_{i+2} + s_{i+1} - 2y[x_{i+1}, x_{i+2}]}{\Delta x_{i+1}^2}(-\Delta x_{i+1}).$$

We then enforce $\frac{1}{2}C_i''(x_{i+1}) = \frac{1}{2}C_{i+1}''(x_{i+1})$ for $i = 1, \cdots N-2$, with the $\frac{1}{2}$ factors here only for convenience. This yields the following equation

$$\frac{y[x_i, x_{i+1}] - s_i}{\Delta x_i} + 2\frac{s_{i+1} + s_i - 2y[x_i, x_{i+1}]}{\Delta x_i} = \frac{y[x_{i+1}, x_{i+2}] - s_{i+1}}{\Delta x_{i+1}} - \frac{s_{i+2} + s_{i+1} - 2y[x_{i+1}, x_{i+2}]}{\Delta x_{i+1}},$$

or upon combination of like terms

$$\frac{s_i + 2s_{i+1} - 3y[x_i, x_{i+1}]}{\Delta x_i} = \frac{-s_{i+2} - 2s_{i+1} + 3y[x_{i+1}, x_{i+2}]}{\Delta x_{i+1}}.$$

Multiplication of both sides by $\Delta x_i \Delta x_{i+1}$ then yields

$$\Delta x_{i+1}\left(s_i + 2s_{i+1} - 3y[x_i, x_{i+1}]\right) = \Delta x_i\left(-s_{i+2} - 2s_{i+1} + 3y[x_{i+1}, x_{i+2}]\right).$$

Moving all s_i terms to the left side and all divided differences to the right, we arrive at (5).