LECTURE 9: *Introduction to Splines*

1.11 Splines

Spline fitting, our next topic in interpolation theory, is an essential tool for engineering design. As in the last lecture, we strive to interpolate data using low-degree polynomials between consecutive grid points. The piecewise linear functions of Section 1.10 were simple, but suffered from unsightly kinks at each interpolation point, reflecting a discontinuity in the first derivative. By increasing the degree of the polynomial used to model f on each subinterval, we can obtain smoother functions.

Long before numerical analysts got their hands on them, 'splines' were used in the woodworking, shipbuilding, and aircraft industries. In such work 'splines' refer to thin pieces of wood that are bent between physical constraints called *ducks* (apparently these were also called *dogs* and *rats* in some settings; modern versions are sometimes called *whales* because of their shape). The spline, a thin beam, bends gracefully between the ducks to give a graceful curve. For some discussion of this history, see the brief 'History of Splines' note by James Epperson in the 19 July 1998 NA Digest, linked from the class website. For a beautiful derivation of cubic splines from Euler's beam equation—that is, from the original physical situation, see Gilbert Strang's *Introduction to Applied Mathematics*, Wellesley Cambridge Press, 1986.

1.11.1 Cubic splines: first approach

Rather than setting $S'(x_j)$ to a particular value, suppose we simply require S' to be continuous throughout $[x_0, x_n]$. This added freedom allows us to impose a further condition: require S'' to be continuous on $[x_0, x_n]$, too. The polynomials we construct are called *cubic splines*. In spline parlance, the interpolation points $\{x_j\}_{j=0}^n$ are called *knots*.

These cubic spine requirements can be written as:

$$\begin{aligned}
s_j(x_{j-1}) &= f(x_{j-1}), && j = 1, \ldots, n;\\
s_j(x_j) &= f(x_j), && j = 1, \ldots, n;\\
s_j'(x_j) &= s_{j+1}'(x_j), && j = 1, \ldots, n-1;\\
s_j''(x_j) &= s_{j+1}''(x_j), && j = 1, \ldots, n-1.
\end{aligned}$$

Compare these requirements to those imposed by piecewise cubic Hermite interpolation. Add up all these new requirements:

$$n + n + (n-1) + (n-1) = 4n - 2 \text{ constraints}$$

and compare to the total free variables available:

$$(n \text{ cubic polynomials}) \times (4 \text{ variables per cubic}) = 4n \text{ variables.}$$

So far, we thus have an *underdetermined system*, and there will be infinitely many choices for the function $S(x)$ that satisfy the constraints.

There are several canonical ways to add two extra constraints that uniquely define S:

- *natural* splines require $S''(x_0) = S''(x_n) = 0$;
- *complete* splines specify values for $S'(x_0)$ and $S'(x_n)$;
- *not-a-knot* splines require S''' to be continuous at x_1 and x_{n-1}.

Since the third derivative of a cubic is a constant, the *not-a-knot* requirement forces $s_1 = s_2$ and $s_{n-1} = s_n$. Hence, while $S(x)$ interpolates the data at x_2 and x_{n-1}, the derivative continuity requirements are automatic at those knots; hence the name "not-a-knot".

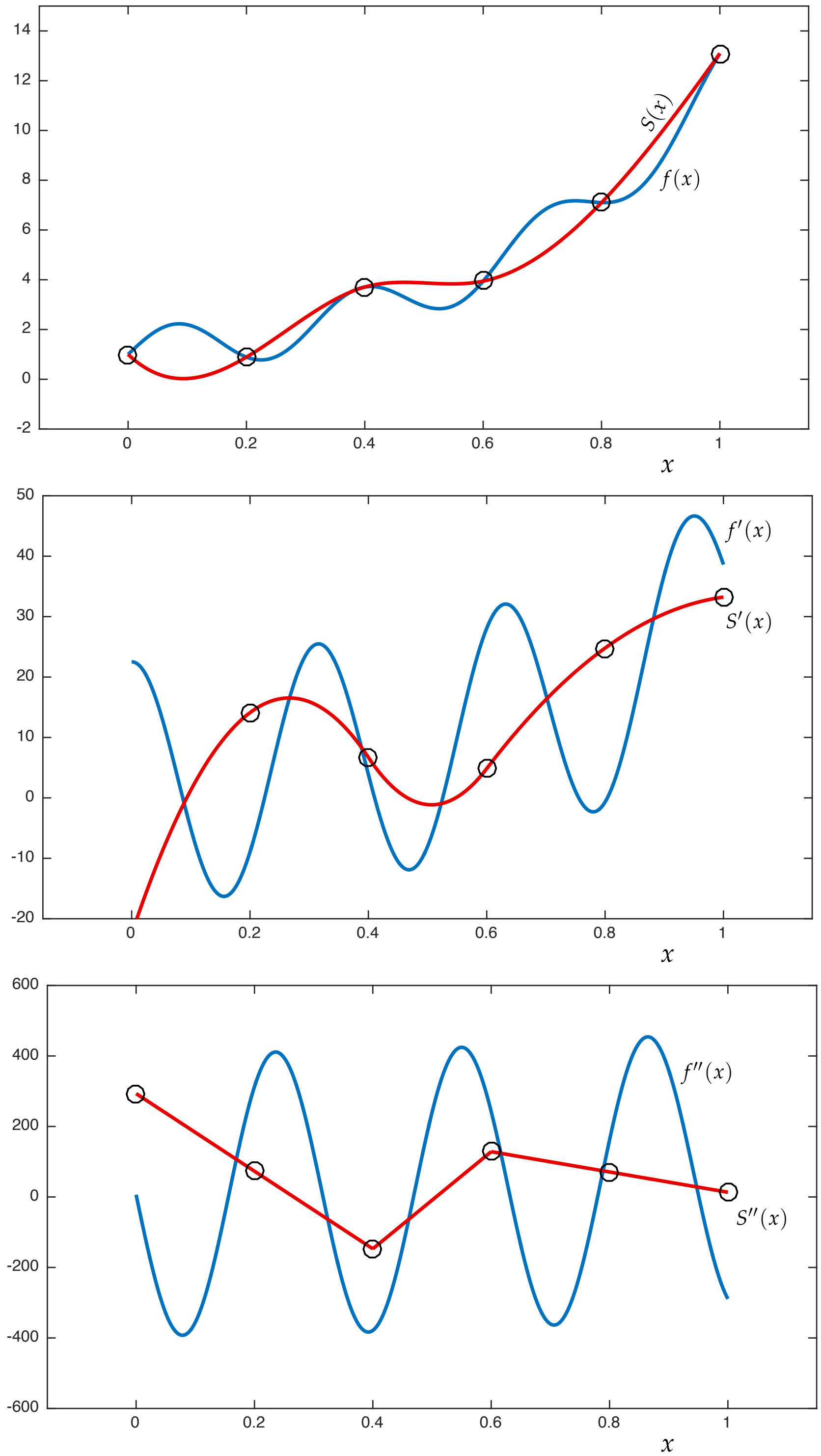

Figure 1.20: Not-a-knot cubic spline interpolant to $f(x) = \sin(20x) + e^{5x/2}$ at $n = 5$ uniformly spaced knots (top), along with its first (middle) and second (bottom) derivative. Note that S, S', and S'' are all continuous. Look closely at the plot of S'': clearly this function will have jump discontinuities at the interior nodes x_2 and x_3, but the *not-a-knot* condition forces S''' to be continuous at the knots x_1 and $x_4 = x_{n-1}$.

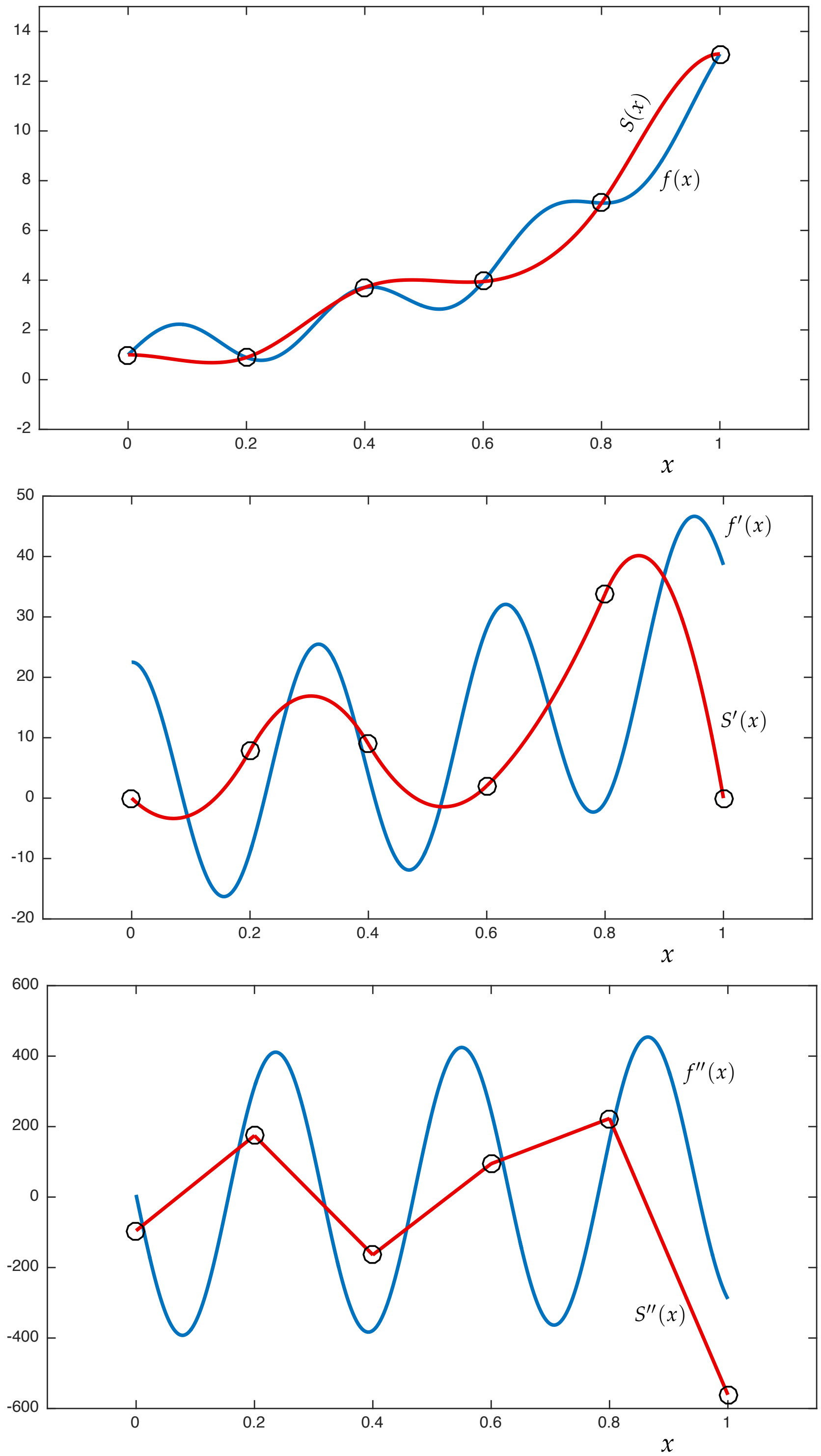

Figure 1.21: Complete cubic spline interpolant to $f(x) = \sin(20x) + e^{5x/2}$ at $n = 5$ uniformly spaced knots (top), along with its first (middle) and second (bottom) derivative. Note that S, S', and S'' are all continuous. For a *complete* cubic spline, one specifies the value of $S'(x_0)$ and $S'(x_n)$; in this case we have set $S'(x_0) = S'(x_n) = 0$, as you can confirm in the middle plot. In the bottom plot, see that $S'''(x)$ will have jump discontinuities at all the interior knots $x_1, \ldots, x_{n-1}$, in contrast to the not-a-knot spline shown in Figure 1.20.

Natural cubic splines are a popular choice for they can be shown, in a precise sense, to minimize curvature over all the other possible splines. They also model the physical origin of splines, where beams of wood extend straight (i.e., zero second derivative) beyond the first and final 'ducks.'

Continuing with the example from the last section, Figure 1.20 shows a not-a-knot spline, where S''' is continuous at x_1 and x_{n-1}. The cubic polynomials s_1 for $[x_0, x_1]$ and s_2 for $[x_1, x_2]$ must satisfy

$$\begin{aligned} s_1(x_1) &= s_2(x_1) \\ s_1'(x_1) &= s_2'(x_1) \\ s_1''(x_1) &= s_2''(x_1) \\ s_1'''(x_1) &= s_2'''(x_1) \end{aligned}$$

Two cubics that match these four conditions must be the same: $s_1(x) = s_2(x)$, and hence x_1 is 'not a knot.' (The same goes for x_{n-1}.) Notice this behavior in Figure 1.20. In contrast, Figure 1.21 shows the complete cubic spline, where $S'(x_0) = S'(x_n) = 0$.

However we assign the two additional conditions, we get a system of $4n$ equations (the various constraints) in $4n$ unknowns (the cubic polynomial coefficients). These equations can be set up as a system involving a banded coefficient matrix (zero everywhere except for a limited number of diagonals on either side of the main diagonal). We could derive this linear system by directly enforcing the continuity conditions on the cubic polynomial that we have just described. Instead, we will develop a more general approach that expresses the spline function $S(x)$ as the linear combination of special basis functions, which themselves are splines.

One can arrange Gaussian elimination to solve an $n \times n$ tridiagonal system in $\mathcal{O}(n)$ operations.

Try constructing this matrix!