MAT360 Section Summary: 3.4
Cubic Spline Interpolation

1. Summary

Cubic splines derive their name from a tool used by boat builders: it's a metal band, passed between fixed points to give a pleasing shape to a hull.

We want to pass a cubic through a bunch of knots so that a pleasing shape appears. We saw in the last section that we can pass cubics through a pair of points that "osculate" the true curve: that's Hermite interpolation.

We're going to use a different criterion in this section, which doesn't require knowledge of the derivatives at particular values, but which has not only slope continuity at the knots, but also second derivative continuity! It might seem odd that we can do this, as it seems that we don't have that much freedom – but, in fact, we can accomplish this feat.

There is a trade-off from Hermite splines, however, in that we can't control the actual values of the slope or second derivative: we just know that they're equivalent. That's the difference from Hermite, in which we actually specify the derivative values....

2. Definitions

- **Cubic spline interpolant**: Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \ldots < x_n = b$, a **cubic spline interpolant** S for f is a function that satisfies the following criteria:
  (a) $S(x)$ is a cubic polynomial denoted $S_i(x)$ on each subinterval $[x_i, x_{i+1}]$.
  (b) $S(x_i) = f(x_i)$ for all knots (S interpolates).
  (c) $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ (the function values match up at knots)
  (d) $S'_{i+1}(x_{i+1}) = S'_i(x_{i+1})$ (the derivative values match up at knots)
  (e) $S''_{i+1}(x_{i+1}) = S''_i(x_{i+1})$ (the second derivative values match up at knots)
  (f) One of the following holds:
    i. $S''(x_0) = S''(x_n) = 0$ (**free** or **natural** splines), or
    ii. $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (**clamped** splines).

  The cubics $S_i(x)$ are generally expressed in shifted form, as

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Example: #3a, p. 153

Example: #4a, p. 153

Example: #5a, p. 153

- **Hermite cubic splines**: splines made up of Hermite cubics joining subintervals.

3. Theorems/Formulas

Whereas for Hermite splines the individual cubics are determined locally, and can be successively generated along the way, the free or clamped cubic splines must be generated all at once: the conditions involved in their creation give rise to a system of linear equations, which must be simultaneously satisfied. We need some linear algebra!

Theorem 3.11: If f is defined at $a = x_0 < x_1 < \ldots < x_n = b$, then f has a unique natural spline interpolant on the nodes $x_0 < x_1 < \ldots < x_n$.

Theorem 3.12: If f is defined at $a = x_0 < x_1 < \ldots < x_n = b$ and differentiable at a and b, then f has a unique clamped spline interpolant on the nodes $x_0 < x_1 < \ldots < x_n$.

Theorem 3.13: Let $f \in C^4[a, b]$ with $|f^{(4)}(x)| \leq M$ on $[a, b]$. If S is the unique clamped cubic spline, then

$$|f(x) - S(x)| \leq \frac{5M}{384} \max_{0 \leq i \leq n-1} (x_{i+1} - x_i)^4$$

4. Properties/Tricks/Hints/Etc.

Linear splines are actually pretty popular, but though they are continuous, they aren't differentiable, which is a pretty serious problem.

Quadratic splines aren't very popular, because they don't have enough free parameters to be really useful: they give us fit, and then we can match derivatives at the nodes, but the problem is that once we've determined the starting slope at x_0, say, the slope at x_n is determined: we have no freedom to adjust that (all our free parameters are used up). So we're stuck with whatever we get, and that can be ugly.

Each cubic polynomial making up a spline has four free parameters, which it can use to satisfy four constraints: on each subinterval, we satisfy two constraints for interpolation, leaving us with two free parameters on each subinterval.

On the first subinterval, we satisfy a boundary derivative condition (either first or second derivative), and then we have one free parameter left. Once that parameter is fixed, the next subinterval cubic is stuck: it has to use its two free parameters to meet the first and second derivatives at its left endpoint; and so on, down the line, until we reach the last subinterval.

At the last subinterval, we have to satisfy the two constraints on the left endpoint, so it seems that we won't be able to satisfy the last derivative condition (either first or second derivative) on the right endpoint. But wait: we haven't actually established the choice of the free parameter on the first subinterval! As we play with that parameter, we can ultimately get the behavior we want on the far endpoint, and we stop. There is a unique choice of that free parameter that will give us the desired behavior, which is why we need to solve a system of equations to get the coefficients.

We determine the coefficients of each of the $S_i(x)$ cubic polynomials by matching their coefficients. Consider, for example, a trio of internal cubics S_1, S_2, and S_3.

$$S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3$$

$$S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3$$

$$S_3(x) = a_3 + b_3(x - x_3) + c_3(x - x_3)^2 + d_3(x - x_3)^3$$

Now when we evaluate at the left endpoint knot of these, we obtain

$$a_i = f(x_i)$$

because of the interpolation. The derivative at the left endpoint is also easy, being b_i. But this is unspecified!