Week 9: Wednesday, Mar 30

Problem du jour

Suppose $p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$. Write a linear system for the coefficients a_j such that $p(0) = p_0$, $p'(0) = q_0$, $p(1) = p_1$, $p'(1) = q_1$.

Answer: If we simply write the interpolation conditions in matrix form, we have

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} p_0 \\ q_0 \\ p_1 \\ q_1 \end{bmatrix}.$$

If we were a little clever, we might notice that the first two equations simply yield $a_0 = p_0$ and $a_1 = q_1$, so we can reduce to the 2-by-2 system

$$\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} p_1 - p_0 - q_0 \\ q_1 - q_0 \end{bmatrix}$$

Logistics

- HW 5 is due via CMS at 11:59 next Monday
- Prelim 2 is 4/7: least squares; nonlinear equations and optimization; interpolation, numerical differentiation, and integration

Summary of last time

We spent most of the last lecture discussing three forms of polynomial interpolation. In each case, we were given function values $\{y_i\}_{i=0}^d$ at points $\{x_i\}_{i-1}^d$, and we wanted to construct a degree d polynomial such that $p(x_i) = y_i$. We do this in general by writing

$$p(x) = \sum_{j=0}^{d} c_j \phi_j(x),$$

where the functions $\phi_j(x)$ form a basis for the space of polynomials of degree at most d. Then we use the interpolation conditions to determine the

coefficients c_j via a linear system

$$Ac = y,$$

where $A_{ij} = \phi_j(x_i)$. In the last lecture, we considered three choices of basis functions $\phi_j(x)$:

1. Power basis:
$$\phi_j(x) = x^j.$$
2. Lagrange basis:
$$\phi_j(x) = \frac{\prod_{i \neq j}(x - x_i)}{\prod_{i \neq j}(x_j - x_i)}.$$
3. Newton basis:
$$\phi_j(x) = \prod_{i<j}(x - x_i).$$

The power basis yields an ill-conditioned system matrix (the Vandermonde matrix). The Lagrange basis leads to a trivial linear system, but it takes $O(d)$ time to evaluate each Lagrange polynomial and so $O(d^2)$ time to evluate the interpolant. The Newton basis is a nice compromise: the coefficients can be computed in $O(d^2)$ time as the solution to an upper triangular system or through a divided difference recurrence, and the polynomial itself can be evaluated in $O(d)$ time using an algorithm like Horner's rule.

Divided differences and derivatives

The coefficients in the Newton form of the interpolant are *divided differences*. For a given function f known at sample points $\{x_i\}_{i=1}^n$, we can evaluate divided differences recursively:

$$\begin{aligned} f[x_i] &= f(x_i), \\ f[x_i, x_{i+1}, \ldots, x_j] &= \frac{f[x_i, x_{i+1}, \ldots, x_{j-1}] - f[x_{i+1}, \ldots, x_j]}{x_i - x_j}. \end{aligned}$$

This recurrence is numerically preferable to finding the coefficients of the Newton interpolant by back substitution.

You might recognize the first divided difference $f[x_1, x_2]$ as a derivative approximation. In fact, if f is a differentiable function, then the mean value theorem tells us that $f[x_1, x_2] = f'(\xi)$ for some ξ between x_1 and x_2. Thus if f is a continuously differentiable function, it makes sense to define

$$f[x_i, x_i] \equiv f'(x_i).$$

This gives us a natural way to solve *Hermite* interpolation problems in which we specify both function values and derivatives at specified points.

More generally, it turns out that if $f \in C^{m-1}$, then

$$f[x_1, x_2, \ldots, x_m] = \frac{f^{(m-1)}(\xi)}{(m-1)!}, \quad \text{some } \xi \in (\min\{x_i\}, \max\{x_i\})$$

Therefore, in the limiting case as we let all the x_j approach some common point x_0, the Newton form of the interpolant degenerates into a Taylor approximation.

Error in polynomial approximation

The relation between divided differences and derivatives is incredibly useful in reasoning about how well polynomial interpolants approximate an underlying function. Suppose we approximate $f \in C^n$ by a polynomial p of degree $n-1$ that interpolates f at points $\{x_i\}_{i=1}^n$. At any point x, we can write $f(x) = p^*(x)$, where $p^*(x)$ is the degree n polynomial interpolating f at $\{x_i\}_{i=1}^n \cup \{x\}$. This may seem somewhat silly, but it gives us the error representation

$$\begin{aligned} f(x) - p(x) &= p^*(x) - p(x) \\ &= f[x_1, \ldots, x_n, x] \prod_{i=1}^{n} (x - x_i) \\ &= \frac{f^{(n)}(\theta)}{n!} \prod_{i=1}^{n} (x - x_i). \end{aligned}$$

If x lies within h of all the values x_i and $|f^{(n)}| \leq M_n$ on the interval bounded by the points in question, then we have

$$|f(x) - p(x)| \leq \frac{M_n h^n}{n!}.$$

This bound suggests that high-order polynomial interpolation of a smooth function over a bounded interval can provide very accurate approximations to the function values, with two catches. First, if the h^n term may not be small (especially in *extrapolation*, where x lies outside the convex hull of the data points). Second, M_n may grow quickly as a function of n. Note that these two effects are not independent; for example, we can scale the nodal coordinates to make h smaller, but then M_n gets commensurately bigger. The standard example of these effects, due to Runge, is the function

$$\phi(t) = \frac{1}{1+25t^2}.$$

Polynomial approximations to $\phi(t)$ by interpolation on a uniform mesh on $[-1,1]$ oscillate wildly toward the end points of the interval, and it is not true in this case that ever higher-degree interpolating polynomials provide ever-better function approximations. This is a general problem, known as the *Runge phenomena*.

Piecewise polynomial approximations

Polynomials are convenient for interpolation for a few reasons: we know how to manipulate them symbolically, we can evaluate them quickly, and there is a theorem of analysis (the Weierstrass approximation theorem) that says that any continuous function on some interval $[a,b]$ can be uniformly approximated by polynomials. In practice, though, high-degree polynomial interpolation does not always provide fantastic function approximation. An alternative approach that retains the advantages of working with polynomials is to work with *piecewise* polynomial functions.

Perhaps the simplest example is piecewise linear interpolation; if function values $f(x_j)$ are given at points $x_1 < x_2 < x_3 < \ldots < x_n$, then we write the approximating function $\hat{f}(x)$ as

$$\hat{f}(x) = \frac{f(x_j)(x-x_j) + f(x_{j+1})(x_{j+1}-x)}{x_{j+1}-x_j}, \quad x \in [x_j, x_{j+1}].$$

Alternately, we can write

$$\hat{f}(x) = \sum_{j=1}^{n} \phi_j(x) f(x_j)$$

where $\phi_j(x)$ is a "hat function":

$$\phi_j(x) = \begin{cases} (x - x_{j+1})/(x_j - x_{j+1}), & x \in [x_j, x_{j+1}], \\ (x - x_{j-1})/(x_j - x_{j-1}), & x \in [x_{j-1}, x_j], \\ 0 & \text{otherwise.} \end{cases}$$

This last you may recognize as similar in spirit to using a basis of Lagrange polynomials for polynomial interpolation.

Piecewise linear interpolation has several virtues: if f is positive or monotone, then any piecewise linear interpolant inherits these properties. But if f is reasonably smooth and the data points are widely spaced, it may make sense to use higher-order polynomials. For example, we might decide to use a *cubic spline* $\hat{f}(x)$ characterized by the properties:

- Interpolation: $\hat{f}(x_i) = f(x_i)$
- Twice differentiability: $\hat{f}'$ and $\hat{f}''$ are continuous at $\{x_2, \ldots, x_{n-1}\}$

The interpolation and differentiability constraints give us $4n - 2$ constraints on the $4n$-dimensional space of piecewise polynomial functions that are defined by general cubics on each interval $[x_j, x_{j+1}]$. In order to uniquely determine the spline, we need some additional constraint; common choices are

- Specified values of f' at x_1 and x_n
- A natural spline: $f''(x_1) = f''(x_n) = 0$
- Not-a-knot conditions: f''' is continuous at x_2 and x_{n-1}
- Periodicity: $f'(x_1) = f'(x_n), f''(x_1) = f''(x_n)$

In addition to spline conditions, one can choose piecewise cubic polynomials that satisfy Hermite interpolation conditions (sometimes referred to by the acronym PCHIP or Piecewise Cubic Hermite Interpolating Polynomials). That is, the function values and derivatives are specified at each nodal point. If we don't actually have derivative values prescribed at the nodal points, then we can assign these values to satisfy additional constraints. We gain this flexibility at the cost of some differentiability; piecewise cubic Hermite interpolants are in general not twice continuously differentiable.

As in the case of polynomial interpolation, there are several different bases for the space of piecewise cubic functions. Any choice of locally supported

basis functions (basis functions that are only nonzero on only a fixed number of intervals $[x_j, x_{j+1}]$) leads to a banded linear system which can be solved in $O(n)$ time to find either cubic splines or piecewise Hermite cubic interpolants. One common choice of basis is the B-spline basis, which you can find described in the book.