Interpolation by polynomials and splines

Michael S. Floater

October 16, 2012

Abstract

These notes provide an introduction to the interpolation of points and sometimes derivatives by polynomials and splines.

1 Lagrange interpolation

We sometimes want to *interpolate* a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$ with a smooth curve. One way to do this is to fit a polynomial. If the points are sampled from some parametric curve $\mathbf{f}$ in sequence, the *parameter values* $t_0, t_1, \ldots, t_n$, with $t_i < t_{i+1}$, such that $\mathbf{x}_i = \mathbf{f}(t_i)$ may be available. Otherwise we have to choose suitable parameter values. In either case, once the t_i are determined, we look for the polynomial $\mathbf{p}$ of degree $\leq n$ that satisfies the *interpolation conditions*

$$\mathbf{p}(t_i) = \mathbf{x}_i, \qquad i = 0, 1, \ldots, n.$$

In fact $\mathbf{p}$ is uniquely determined by these conditions. If we express $\mathbf{p}$ in the form

$$\mathbf{p}(t) = \sum_{j=0}^{n} \mathbf{a}_j t^j, \tag{1}$$

we see that there are $n+1$ unknowns $\mathbf{a}_j$ and the interpolation conditions provide $n+1$ equations. These equations can be written as the linear system

$$\begin{bmatrix} 1 & t_0 & t_0^2 & \cdots & t_0^n \\ 1 & t_1 & t_1^2 & \cdots & t_1^n \\ \vdots & & & & \vdots \\ 1 & t_n & t_n^2 & \cdots & t_n^n \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}.$$

The matrix on the left is known as the *Vandermonde* matrix and it can be shown that its determinant is

$$\prod_{0 \le i < j \le n} (t_j - t_i),$$

which is clearly positive and therefore non-zero. Thus the matrix is non-singular and there is a unique interpolant $\mathbf{p}$.

In a modelling context, we might want to find the interpolating polynomial in some Bézier representation, such as

$$\mathbf{p}(t) = \sum_{j=0}^{n} \mathbf{c}_j B_i^n(t). \tag{2}$$

We could find the $\mathbf{c}_j$ by first finding the $\mathbf{a}_j$ and then converting the monomial form (1) to the Bézier form (2), or solve directly for the $\mathbf{c}_j$, which means solving the linear system

$$\begin{bmatrix} B_0^n(t_0) & B_1^n(t_0) & B_2^n(t_0) & \cdots & B_n^n(t_0) \\ B_0^n(t_1) & B_1^n(t_1) & B_2^n(t_1) & \cdots & B_n^n(t_1) \\ \vdots & & & & \vdots \\ B_0^n(t_n) & B_1^n(t_n) & B_2^n(t_n) & \cdots & B_n^n(t_n) \end{bmatrix} \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_n \end{bmatrix} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}.$$

If it is not important how we represent $\mathbf{p}$ we can avoid solving a linear system by representing $\mathbf{p}$ in the *Lagrange* basis. The i-th Lagrange function is the polynomial

$$L_i(t) = \prod_{\substack{j=0 \\ j \ne i}}^{n} \frac{t - t_j}{t_i - t_j}, \qquad i = 0, 1, \ldots, n,$$

which evidently has the property that $L_i(t_i) = 1$ and $L_i(t_k) = 0$ if $k \ne i$. Hence the interpolating polynomial is simply

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{x}_i L_i(t).$$

2 Hermite interpolation

It is often desirable to find an interpolating curve that also matches derivative data. This is known as *Hermite interpolation*. An important special case is to interpolate $\mathbf{f}$ and all its derivatives up to some order k at two points. If the corresponding parameter values are a and b, this means finding $\mathbf{p}$ such that

$$\mathbf{p}^{(i)}(a) = \mathbf{f}^{(i)}(a) \quad \text{and} \quad \mathbf{p}^{(i)}(b) = \mathbf{f}^{(i)}(b), \qquad i = 0, 1, \ldots, k.$$

These conditions determine the polynomial $\mathbf{p}$ uniquely if its degree is at most $n = 2k + 1$. The Bézier representation,

$$\mathbf{p}(t) = \sum_{j=0}^{n} \mathbf{c}_j B_j^n(u), \tag{3}$$

where $u = (t - a)/(b - a)$, is in this case quite convenient and easy to find.

In Chapter 2 we saw that

$$\mathbf{p}^{(i)}(a) = \frac{n!}{(n-i)!} \frac{\Delta^i \mathbf{c}_0}{(b-a)^i} \quad \text{and} \quad \mathbf{p}^{(i)}(b) = \frac{n!}{(n-i)!} \frac{\Delta^i \mathbf{c}_{n-i}}{(b-a)^i}. \tag{4}$$

Therefore, we need to find the coefficients $\mathbf{c}_0, \ldots, \mathbf{c}_n$ such that

$$\Delta^i \mathbf{c}_0 = \mathbf{b}_i, \quad \Delta^i \mathbf{c}_{n-i} = \mathbf{b}_{n-i}, \qquad i = 0, 1, \ldots, k,$$

where

$$\mathbf{b}_i := \frac{(n-i)!}{n!}(b-a)^i \mathbf{f}^{(i)}(a),$$
$$\mathbf{b}_{n-i} := \frac{(n-i)!}{n!}(b-a)^i \mathbf{f}^{(i)}(b).$$

One can show that the solutions are

$$\mathbf{c}_i = \sum_{j=0}^{i} \binom{i}{j} \mathbf{b}_j, \qquad \mathbf{c}_{n-i} = \sum_{j=0}^{i} (-1)^j \binom{i}{j} \mathbf{b}_{n-j}.$$

For example, in the cubic case, with $k = 1$ and $n = 3$,

$$\begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_0 + \mathbf{b}_1 \\ \mathbf{b}_3 - \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{f}(a) \\ \mathbf{f}(a) + (b-a)\mathbf{f}'(a)/3 \\ \mathbf{f}(b) - (b-a)\mathbf{f}'(b)/3 \\ \mathbf{f}(b) \end{bmatrix}. \tag{5}$$

3 Piecewise cubic Hermite interpolation

An alternative to using polynomials for interpolation is to use piecewise polynomials, which are better suited when the number of interpolation conditions is high. For example, we can interpolate points $\mathbf{x}_i = \mathbf{f}(t_i)$ and first derivatives $\mathbf{m}_i = \mathbf{f}'(t_i)$, where $t_0 < t_1 < \cdots < t_n$, with a piecewise cubic curve $\mathbf{s}$ with parameter domain $[t_0, t_n]$ and C^1 continuity by fitting a cubic to each consecutive pair of data. One way of representing the individual cubics is in Bernstein form. Thus, for $t \in [t_i, t_{i+1}]$, we let $\mathbf{s}(t) = \mathbf{s}_i(t)$, where

$$\mathbf{s}_i(t) = \sum_{j=0}^{3} \mathbf{c}_j B_j^3(u), \tag{6}$$

and, using (5),

$$\begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_i + h_i \mathbf{m}_i/3 \\ \mathbf{x}_{i+1} - h_i \mathbf{m}_{i+1}/3 \\ \mathbf{x}_{i+1} \end{bmatrix}, \tag{7}$$

with $h_i = t_{i+1} - t_i$ and $u = (t - t_i)/h_i$.

If the slopes $\mathbf{m}_i$ are not known, it is usual to estimate them from the points $\mathbf{x}_j$ nearby. For example, a common choice is to set

$$\mathbf{m}_i = \frac{\mathbf{x}_{i+1} - \mathbf{x}_{i-1}}{t_{i+1} - t_{i-1}}, \qquad i = 1, \ldots, n-1.$$

This is a good approximation to $\mathbf{f}'(t_i)$ when the t_i are uniformly spaced. At the endpoints, a one-sided approximation is necessary, and we could, for example, set

$$\mathbf{m}_0 = \frac{\mathbf{x}_1 - \mathbf{x}_0}{t_1 - t_0}, \qquad \mathbf{m}_n = \frac{\mathbf{x}_n - \mathbf{x}_{n-1}}{t_n - t_{n-1}}.$$

4 Cubic spline interpolation

An alternative way of choosing the interior slopes $\mathbf{m}_1, \ldots, \mathbf{m}_{n-1}$ is to determine them in such a way that the piecewise cubic curve $\mathbf{s}$ has C^2 continuity at the corresponding breakpoints. i.e.,

$$\mathbf{s}_i''(t_i) = \mathbf{s}_{i-1}''(t_i), \qquad i = 1, \ldots, n-1. \tag{8}$$

It turns out that this system of $n-1$ equations in the unknowns $\mathbf{m}_1, \ldots, \mathbf{m}_{n-1}$ has a unique solution. To see this, observe that applying the derivative formulas to the Bézier curve $\mathbf{s}_i$ in (6) gives

$$\mathbf{s}_i''(t_i) = \frac{6}{h_i^2}\Delta^2\mathbf{c}_0, \qquad \mathbf{s}_i''(t_{i+1}) = \frac{6}{h_i^2}\Delta^2\mathbf{c}_1,$$

with the $\mathbf{c}_j$ given by (7). Thus equation (8) can be expressed as

$$h_i\mathbf{m}_{i-1} + 2(h_{i-1} + h_i)\mathbf{m}_i + h_{i-1}\mathbf{m}_{i+1} = \mathbf{b}_i,$$

where

$$\mathbf{b}_i = 3\left(\frac{h_i}{h_{i-1}}\Delta\mathbf{x}_{i-1} + \frac{h_{i-1}}{h_i}\Delta\mathbf{x}_i\right).$$

This gives us the linear system of equations

$$\begin{bmatrix} \beta_1 & \gamma_1 & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{n-2} & \beta_{n-2} & \gamma_{n-2} \\ & & & \alpha_{n-1} & \beta_{n-1} \end{bmatrix} \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_{n-2} \\ \mathbf{m}_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 - h_1\mathbf{m}_0 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{n-2} \\ \mathbf{b}_{n-1} - h_{n-2}\mathbf{m}_n \end{bmatrix},$$

where $\alpha_i = h_i$, $\beta_i = 2(h_{i-1} + h_i)$, and $\gamma_i = h_{i-1}$. This is a tridiagonal system that is strictly diagonally dominant, and therefore has a unique solution.

An alternative method of finding the C^2 cubic spline interpolant $\mathbf{s}$ is to represent it in terms of cubic B-splines. The number of B-splines we need is 4 for the first interval $[t_0, t_1]$, plus 1 for each subsequent interval $[t_i, t_{i+1}]$, giving a total of $n + 3$. We find that $\mathbf{s}$ can be expressed as

$$\mathbf{s}(t) = \sum_{i=1}^{n+3} \mathbf{c}_i N_i^3(t),$$

with respect to the knot vector

$$\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{n+7}) = (t_0, t_0, t_0, t_0, t_1, \ldots, t_{n-1}, t_n, t_n, t_n, t_n).$$

Then the conditions

$$\mathbf{s}(t_i) = \mathbf{x}_i, \qquad i = 0, 1, \ldots, n,$$

and

$$\mathbf{s}'(t_0) = \mathbf{m}_0, \quad \mathbf{s}'(t_n) = \mathbf{m}_n,$$

give $n + 3$ equations in the $n + 3$ unknowns $\mathbf{c}_1, \ldots, \mathbf{c}_{n+3}$. The equations are again linear and form a tridiagonal system which has a unique solution.