CS412: Lecture #14

Mridul Aanjaneya

March 5, 2015

Cubic Hermite Splines

Let us assume a number of x-locations $x_1 < x_2 < \ldots < x_n$ and let us make the hypothesis that we know *both* f and f' at every location x_i. We denote these values by $y_i = f(x_i)$ and $y_i' = f'(x_i)$, for $i = 1, 2, \ldots, n$. As with other methods based on piecewise polynomials, we construct the interpolant as

$$s(x) = \begin{cases} s_1(x), x \in I_1 \\ s_2(x), x \in I_2 \\ \vdots \\ s_{n-1}(x), x \in I_{n-1} \end{cases}$$

where $I_k = [x_k, x_{k+1}]$. In this case, each individual $s_k(x)$ is constructed to match both the function values y_k, y_{k+1} as well as the derivatives y_k', y_{k+1}' at the endpoints of I_k. In detail:

$$s_k(x_k) = y_k, \quad s_k(x_{k+1}) = y_{k+1}, \quad s_k'(x_k) = y_k', \quad s_k'(x_{k+1}) = y_{k+1}' \tag{1}$$

Since $s_k(x) = a_3^{(k)}x^3 + a_2^{(k)}x^2 + a_1^{(k)}x + a_0^{(k)}$ has four unknown coefficients, equation (1) can uniquely define the appropriate values of $a_3^{(k)}$, $a_2^{(k)}$, $a_1^{(k)}$, $a_0^{(k)}$.

Note that equation (1) guarantees that $s(x)$ is *continuous* with continuous derivatives (e.g., a C^1 function). However, we do not strictly enforce that the 2*nd derivative* should be continuous, and in fact it generally will *not* be.

The most straightforward method for determining the coefficients of $s_k(x) = a_3^{(k)}x^3 + a_2^{(k)}x^2 + a_1^{(k)}x + a_0^{(k)}$ mimics the Vandermonde approach for polynomial interpolation:

$$\begin{aligned}
s_k(x_k) = y_k \quad &\Rightarrow \quad a_3^{(k)}x_k^3 + a_2^{(k)}x_k^2 + a_1^{(k)}x_k + a_0^{(k)} = y_k \\
s_k(x_{k+1}) = y_{k+1} \quad &\Rightarrow \quad a_3^{(k)}x_{k+1}^3 + a_2^{(k)}x_{k+1}^2 + a_1^{(k)}x_{k+1} + a_0^{(k)} = y_{k+1} \\
s_k'(x_k) = y_k' \quad &\Rightarrow \quad 3a_3^{(k)}x_k^2 + 2a_2^{(k)}x_k + a_1^{(k)} = y_k' \\
s_k'(x_{k+1}) = y_{k+1}' \quad &\Rightarrow \quad 3a_3^{(k)}x_{k+1}^2 + 2a_2^{(k)}x_{k+1} + a_1^{(k)} = y_{k+1}'
\end{aligned}$$

$$\Rightarrow \begin{bmatrix} x_k^3 & x_k^2 & x_k & 1 \\ x_{k+1}^3 & x_{k+1}^2 & x_{k+1} & 1 \\ 3x_k^2 & 2x_k & 1 & 0 \\ 3x_{k+1}^2 & 2x_{k+1} & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3^{(k)} \\ a_2^{(k)} \\ a_1^{(k)} \\ a_0^{(k)} \end{bmatrix} = \begin{bmatrix} y_k \\ y_{k+1} \\ y'_k \\ y'_{k+1} \end{bmatrix}$$

The second method attempts to mimic the Lagrange interpolation approach, where we wrote

$$\mathcal{P}_{n-1}(x) = y_0 l_0(x) + y_1 l_1(x) + \ldots + y_n l_n(x)$$

where

$$l_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

What if we could do something similar here? Can we write

$$s_k(x) = y_k q_{00}(x) + y_{k+1} q_{01}(x) + y'_k q_{10}(x) + y'_{k+1} q_{11}(x)$$

Yes, if we have: Note that all q_{ij}'s are *cubic* polynomials.

$$\begin{array}{c|c|c|c} q_{00}(x_k) = 1 & q_{10}(x_k) = 0 & q_{01}(x_k) = 0 & q_{11}(x_k) = 0 \\ q_{00}(x_{k+1}) = 0 & q_{10}(x_{k+1}) = 0 & q_{01}(x_{k+1}) = 1 & q_{11}(x_{k+1}) = 0 \\ q'_{00}(x_k) = 0 & q'_{10}(x_k) = 1 & q'_{01}(x_k) = 0 & q'_{11}(x_k) = 0 \\ q'_{00}(x_{k+1}) = 0 & q'_{10}(x_{k+1}) = 0 & q'_{01}(x_{k+1}) = 0 & q'_{11}(x_{k+1}) = 1 \end{array}$$

In the special case where $x_k = 0, x_{k+1} = 1$, these functions are symbolized with $h_{ij}(x)$ and called the *canonical Hermite basis functions*. Thus, in that case,

$$s_k(x) = y_k h_{00}(x) + y_{k+1} h_{01}(x) + y'_k h_{10}(x) + y'_{k+1} h_{11}(x)$$

In this case, we can either solve a 4×4 system for the coefficients of each $h_{ij}(x)$, or construct it using simple algebraic arguments, e.g.,

$$\begin{aligned} h_{11}(0) = h'_{11}(0) = 0 \quad &\Rightarrow \quad x^2 \text{ is a factor of } h_{11}(x) \\ h_{11}(1) = 0 \quad &\Rightarrow \quad x - 1 \text{ is a factor of } h_{11}(x) \end{aligned}$$

i.e., $h_{11}(x) = Cx^2(x-1) = C(x^3 - x^2) \Rightarrow h'_{11}(x) = C(3x^2 - 2x)$. Given that $h'_{11}(1) = 1 = C(3-2) = C \Rightarrow h_{11}(x) = x^3 - x^2$. The four basis polynomials are similarly derived to be:

$$\begin{aligned} h_{00}(x) &= 2x^3 - 3x^2 + 1 \\ h_{10}(x) &= x^3 - 2x^2 + x \\ h_{01}(x) &= -2x^3 + 3x^2 \\ h_{11}(x) &= x^3 - x^2 \end{aligned}$$

In the more general case where $I_k = [x_k, x_{k+1}]$ (instead of $[0,1]$), we can obtain the basis polynomials using a change of variable $t = (x - x_k)/(x_{k+1} - x_k)$ as follows:

$$s_k(x) = y_k \underbrace{h_{00}(t)}_{q_{00}(x)} + y_{k+1} \underbrace{h_{01}(t)}_{q_{01}(x)} + y'_k \underbrace{(x_{k+1} - x_k)h_{10}(t)}_{q_{10}(x)} + y'_{k+1} \underbrace{(x_{k+1} - x_k)h_{11}(t)}_{q_{11}(x)}$$

The last, and quite common, approach for generating the Hermite spline is using tools similar to Newton interpolation. Remember, when interpolating through (x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3), we obtain

$$\begin{aligned} \mathcal{P}_3(x) &= f[x_0] \cdot 1 + f[x_0, x_1] \cdot (x - x_0) + f[x_0, x_1, x_2] \cdot (x - x_0)(x - x_1) \\ &+ f[x_0, x_1, x_2, x_3] \cdot (x - x_0)(x - x_1)(x - x_2) \end{aligned}$$

The idea is as follows: perform Newton interpolation through the points $(x_k^\star, y_k^\star)$, (x_k, y_k), (x_{k+1}, y_{k+1}), $(x_{k+1}^\star, y_{k+1}^\star)$, where $x_k^\star = x_k - \varepsilon, x_{k+1}^\star = x_{k+1} + \varepsilon$.

We will compute this interpolant using the Newton method, and ultimately set $\varepsilon \to 0$ such that $x_k^\star$ converges onto x_k, and $x_{k+1}^\star$ converges onto x_{k+1}, respectively. Thus,

$$\begin{aligned} s_k(x) &= f[x_k^\star] + f[x_k^\star, x_k](x - x_k^\star) + f[x_k^\star, x_k, x_{k+1}](x - x_k^\star)(x - x_k) \\ &+ f[x_k^\star, x_k, x_{k+1}, x_{k+1}^\star](x - x_k^\star)(x - x_k)(x - x_{k+1}) \end{aligned}$$

Taking the limit as $\varepsilon \to 0$

$$\begin{aligned} s_k(x) &= \left(\lim_{x_k^\star \to x_k} f[x_k^\star] \right) \\ &+ \left(\lim_{x_k^\star \to x_k} f[x_k^\star, x_k] \right) (x - x_k) \\ &+ \left(\lim_{x_k^\star \to x_k} f[x_k^\star, x_k, x_{k+1}] \right) (x - x_k)^2 \\ &+ \left(\lim_{\substack{x_k^\star \to x_k \\ x_{k+1}^\star \to x_{k+1}}} f[x_k^\star, x_k, x_{k+1}, x_{k+1}^\star] \right) (x - x_k)^2 (x - x_{k+1}) \end{aligned}$$

We use the shorthand notation $f[x_k, x_k] = \lim_{x_k^\star \to x_k} f[x_k^\star, x_k]$ and construct the finite difference table as usual.

$x_k^\star$	$f[x_k^\star]$			
x_k	$f[x_k]$	$f[x_k^\star, x_k]$		
x_{k+1}	$f[x_{k+1}]$	$f[x_k, x_{k+1}]$	$f[x_k^\star, x_k, x_{k+1}]$	
$x_{k+1}^\star$	$f[x_{k+1}^\star]$	$f[x_{k+1}, x_{k+1}^\star]$	$f[x_k, x_{k+1}, x_{k+1}^\star]$	$f[x_k^\star, x_k, x_{k+1}, x_{k+1}^\star]$

When $\varepsilon \to 0$, the quantities in this table that involve $x_k^\star$ or $x_{k+1}^\star$ may need to be expressed through limits, e.g.,

$$x_k^\star \to x_k, \quad x_{k+1}^\star \to x_{k+1}, \quad f[x_k^\star] = y_k^\star \to y_k, \quad f[x_{k+1}^\star] = y_{k+1}^\star \to y_{k+1}$$

$$\begin{aligned} f[x_k^\star, x_k] &= \frac{f[x_k] - f[x_k^\star]}{x_k - x_k^\star} \quad \xrightarrow{x_k^\star \to x_k} \quad f'(x_k) = y_k' \\ f[x_{k+1}, x_{k+1}^\star] &= \frac{f[x_{k+1}^\star] - f[x_{k+1}]}{x_{k+1}^\star - x_{k+1}} \quad \xrightarrow{x_{k+1}^\star \to x_{k+1}} \quad f'(x_{k+1}) = y_{k+1}' \end{aligned}$$

Thus, the table gets filled as follows:

x_k	y_k			
x_k	y_k	y_k'		
x_{k+1}	y_{k+1}	$f[x_k, x_{k+1}]$	$f[x_k^\star, x_k, x_{k+1}]$	
x_{k+1}	y_{k+1}	y_{k+1}'	$f[x_k, x_{k+1}, x_{k+1}^\star]$	$f[x_k^\star, x_k, x_{k+1}, x_{k+1}^\star]$

The remaining divided differences are computed normally using the recursive definition. Often times, we skip the "stars" on x_k's and use the simpler notation $f[x_k, x_k]$, $f[x_k, x_k, x_{k+1}, x_{k+1}]$, etc.