Cubic Hermite splines

In the previous lecture, we introduced a different approach to piecewise-cubic polynomial interpolation. In particular, given n x-values (in ascending order)

$$x_1 < x_2 < \cdots < x_{n-1} < x_n$$

and n associated y-values (sampled from a function $f(x)$)

$$y_1, y_2, \ldots, y_{n-1}, y_n \quad , \text{ where } y_k = f(x_k)$$

<u>and</u> assume we also know the derivative $f'(x)$ at the same locations, denoted by:

$$y_1', y_2', \ldots, y_{n-1}', y_n' \quad , \text{ where } y_k' = f'(x_k)$$

As with other methods based on piecewise polynomials, we constuct the interpolant as

$$s(x) = \begin{cases} s_1(x) & x \in I_1 \\ s_2(x) & x \in I_2 \\ \vdots & \vdots \\ s_{n-1}(x) & x \in I_{n-1} \end{cases} \qquad \text{wher } I_k = [x_k, x_{k+1}].$$

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:

$$\left.\begin{aligned} s_k(x_k) &= y_k \\ s_k(x_{k+1}) &= y_{k+1} \\ s_k'(x_k) &= y_k' \\ s_k'(x_{k+1}) &= y_{k+1}' \end{aligned}\right\} \quad (*)$$

Since $s_k(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ has 4 unknown coefficients, the 4 equations $(*)$ could uniquely define the appropriate values of $a_0 \ldots a_3$ (as they do!).

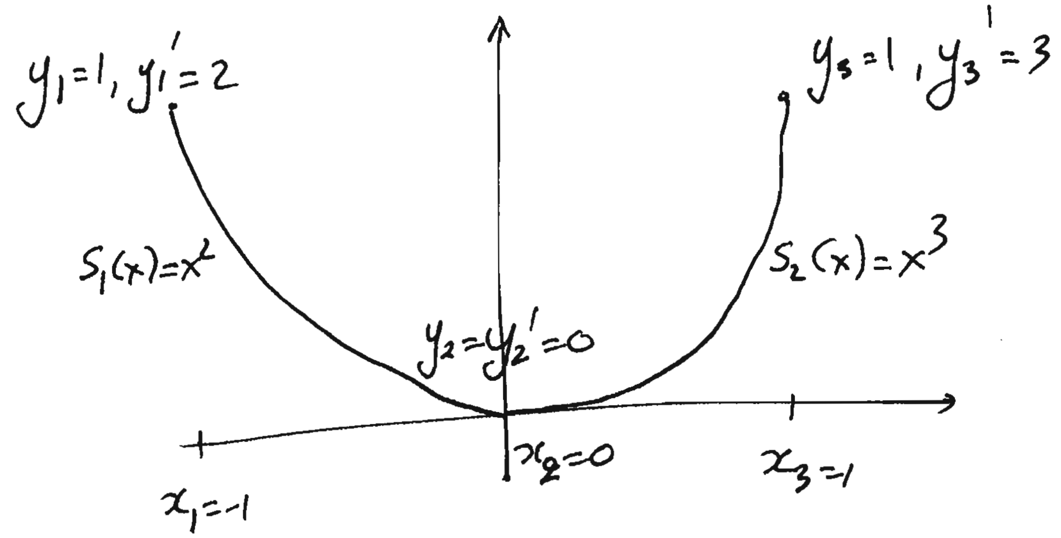
Note that, since we have

$$s_k(x_{k+1}) = y_{k+1} = s_{k+1}(x_{k+1})$$

and $$s_k'(x_{k+1}) = y_{k+1}' = s_{k+1}'(x_{k+1})$$

The resulting interpolant $s(x)$ is <u>continuous</u> with continuous derivatives (e.g. a C^1 function).

However, we do not strictly enforce that the 2nd derivative should be continuous, and in fact it generally will not be:



In this case $S_1''(0) = 2$

while $S_2''(0) = 0$

The most straightforward method for determining the coefficients of $S_k(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ mimics the Vandermonde approach for polynomial interpolation:

$$\left.\begin{array}{lll}
S_k(x_k) = y_k & \Rightarrow & a_3 x_k^3 + a_2 x_k^2 + a_1 x_k + a_0 = y_k \\
S_k(x_{k+1}) = y_{k+1} & \Rightarrow & a_3 x_{k+1}^3 + a_2 x_{k+1}^2 + a_1 x_{k+1} + a_0 = y_{k+1} \\
S_k'(x_k) = y_k' & \Rightarrow & a_3 \cdot 3x_k^2 + a_2 \cdot 2x_k + a_1 = y_k' \\
S_k'(x_{k+1}) = y_{k+1}' & \Rightarrow & a_3 \cdot 3x_{k+1}^2 + a_2 \cdot 2x_{k+1} + a_1 = y_{k+1}'
\end{array}\right\} \Rightarrow$$

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

The 2nd method attempts to mimic the Lagrange interpolation approach, where we wrote

$$p(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x)$$

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

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:

$q_{00}(x_k) = 1$	$q_{10}(x_k) = 0$	$q_{01}(x_k) = 0$	$q_{11}(x_k) = 0$
$q_{00}(x_{k+1}) = 0$	$q_{10}(x_{k+1}) = 0$	$q_{01}(x_{k+1}) = 1$	$q_{11}(x_{k+1}) = 0$
$q_{00}'(x_k) = 0$	$q_{10}'(x_k) = 1$	$q_{01}'(x_k) = 0$	$q_{11}'(x_k) = 0$
$q_{00}'(x_{k+1}) = 0$	$q_{10}'(x_{k+1}) = 0$	$q_{01}'(x_{k+1}) = 0$	$q_{11}'(x_{k+1}) = 1$

(All q_{ij}'s are cubic polynomials!)

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

$$s(x) = y_0 h_{00}(x) + y_1 h_{01}(x) + y_0' h_{10}(x) + y_1' h_{11}(x)$$

In this case we can either solve a 4x4 system for the coefficients of each $h_{ij}(x)$, or construct it using simple algebraic arguments, e.g.

$$h_{11}(0) = h_{11}'(0) = 0 \Rightarrow x^2 \text{ is a factor of } h_{11}(x)$$

$$h_{11}(1) = 0 \Rightarrow x-1 \text{ is a factor of } h_{11}(x)$$

i.e. $h_{11}(x) = C\,x^2(x-1) = C\,(x^3 - x^2)$

$$h_{11}'(x) = C\,(3x^2 - 2x)$$

$$1 = h_{11}'(1) = C\cdot(3-2) = C \Rightarrow \boxed{C = 1}$$

Thus $h_{11}(x) = x^3 - x^2$

The 4 basis polynomials are similarly derived to be:

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

You need not memorize these...

3/1/11 L

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 $t = \dfrac{x - x_k}{x_{k+1} - x_k}$ 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 $(x_0, y_0), \ldots, (x_3, y_3)$ we obtain:

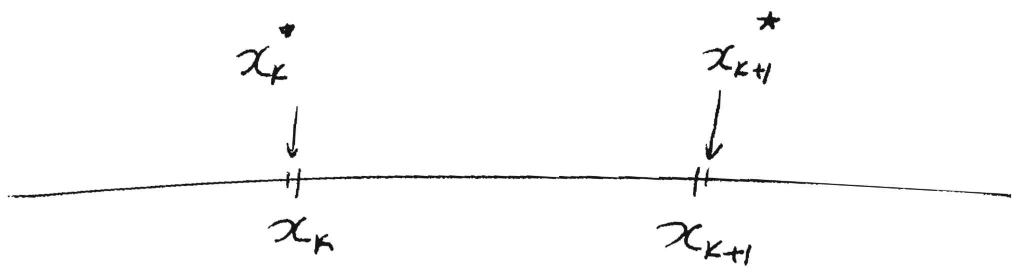
$$\begin{aligned} p(x) = {} & f[x_0] \cdot 1 \\ & + f[x_0, x_1] \cdot (x - x_0) \\ & + f[x_0, x_1, x_2] \cdot (x - x_0)(x - x_1) \\ & + f[x_0, x_1, x_2, x_3] \, (x - x_0)(x - x_1)(x - x_2) \end{aligned}$$

The idea is as follows:

Perform newton interpolation through the points

$$(x_k^*, y_k^*) \;,\; (x_k, y_k) \;,\; (x_{k+1}, y_{k+1}) \;,\; (x_{k+1}^*, y_{k+1}^*)$$

where
$$x_k^* = x_k - \varepsilon$$
$$x_{k+1}^* = x_{k+1} + \varepsilon$$



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

Thus:

$$\begin{aligned} S_k(x) = \; & f[x_k^*] \\ & + f[x_k^*, x_k]\,(x - x_k^*) \\ & + f[x_k^*, x_k, x_{k+1}]\,(x - x_k^*)(x - x_k) \\ & + f[x_k^*, x_k, x_{k+1}, x_{k+1}^*]\,(x - x_k^*)(x - x_k)(x - x_{k+1}). \end{aligned}$$

Taking the limit $\varepsilon \to 0$

$$
\begin{aligned}
S_K = & \left(\lim_{x_K^* \to x_K} f[x_K^*]\right) \cdot \\
& + \left(\lim_{x_K^* \to x_K} f[x_K^*, x_K]\right)(x - x_K) \\
& + \left(\lim_{x_K^* \to x_K} f[x_K^*, x_K, x_{K+1}]\right)(x - x_K)^2 \\
& + \left(\lim_{\substack{x_K^* \to x_K \\ x_{K+1}^* \to x_{K+1}}} f[x_K^*, x_K, x_{K+1}, x_{K+1}^*]\right)(x - x_K)^2 (x - x_{K+1})
\end{aligned}
$$

We use the shorthand notation

$$
f[x_K, x_K] := \lim_{x_K^* \to x_K} f[x_K^*, x_K]
$$

and constuct the finite difference table as usual.

? x_k^*	$f[x_k^*]$?			
x_k	$f[x_k]$	$f[x_k^*, x_k]$?		
x_{k+1}	$f[x_{k+1}]$	$f[x_k, x_{k+1}]$	$f[x_k^*, x_k, x_{k+1}]$	
? x_{k+1}^*	$f[x_{k+1}^*]$?	$f[x_{k+1}, x_{k+1}^*]$?	$f[x_k, x_{k+1}, x_{k+1}^*]$	$f[x_k^*, x_k, x_{k+1}, x_{k+1}^*]$

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

$$x_k^* \longrightarrow x_k$$

$$x_{k+1}^* \longrightarrow x_{k+1}$$

$$f[x_k^*] = y_k^* \longrightarrow y_k$$

$$f[x_{k+1}^*] = y_{k+1}^* \longrightarrow y_{k+1}$$

$$f[x_k^*, x_k] = \frac{f[x_k] - f[x_k^*]}{x_k - x_k^*} \xrightarrow[x_k^* \to x_k]{} f'(x_k) = y_k' \quad !$$

$$f[x_{k+1}, x_{k+1}^*] = \frac{f[x_{k+1}^*] - f[x_{k+1}]}{x_{k+1}^* - x_{k+1}} \xrightarrow[x_{k+1}^* \to x_{k+1}]{} f'(x_{k+1}) = y_{k+1}' .$$

Thus, the table gets filled as follows

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

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.