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.

$$\left.\begin{aligned}
S_k(x_k) = y_k &\Rightarrow a_3x_k^3 + a_2x_k^2 + a_1x_k + a_0 = y_k \\
S_k(x_{k+1}) = y_{k+1} &\Rightarrow a_3x_{k+1}^3 + a_2x_{k+1}^2 + a_1x_{k+1} + a_0 = y_{k+1} \\
S_k'(x_k) = y_k' &\Rightarrow a_3 \cdot 3x_k^2 + a_2 \cdot 2x_k + a_1 = y_k' \\
S_k'(x_{k+1}) = y_{k+1}' &\Rightarrow a_3 \cdot 3x_{k+1}^2 + a_2 \cdot 2x_{k+1} + a_1 = y_{k+1}'
\end{aligned}\right\} \Rightarrow$$

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

The 2nd method attempts to mimic the Lagrange interpolation approach, in which we wrote:

$$P(x) = y_0l_0(x) + y_1l_1(x) + \cdots + y_nl_n(x)$$

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

Could we apply the same principle here? Consider writing $S_k(x)$ as

$$S_k(x) = y_kq_{00}(x) + y_{k+1}q_{01}(x) + y_k'q_{10}(x) + y_{k+1}'q_{11}(x)$$

This polynomial will satisfy all stated conditions, if the polynomials q_{ij} are constructed such that:

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

(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 *canonical* Hermite basis functions. Thus, in that case:

$$S(x) = y_0h_{00}(x) + y_1h_{01}(x) + y_0'h_{10}(x) + y_1'h_{11}(x)$$

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.

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

$$\text{i.e. } h_{11}(x) = Cx^2(x-1) = C(x^3 - x^2)$$
$$h'_{11}(x) = C(3x^2 - 2x)$$
$$1 = h'_{11}(x) = C(3-2) = C \Rightarrow C = 1$$
$$\text{Thus, } h_{11}(x) = x^3 - x^2$$

The 4 basis polynomials are similarly derived to be:

No need to memorize these for exams.

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

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 $t = \frac{x - x_k}{x_{k+1} - x_k}$ 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:

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

The idea is as follows:

Perform Newton interpolation through the points

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

$$\begin{aligned} \text{where } x_k^* \quad &= \quad x_k - \varepsilon \\ x_{k+1}^* \quad &= \quad x_{k+1} + \varepsilon \end{aligned}$$

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:

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

Taking the limit $\varepsilon \to 0$:

$$\begin{aligned} S_k(x) &= (\lim_{x_k^* \to x_k} f[x_k^*]) \\ &\quad + (\lim_{x_k^* \to x_k} f[x_k^*, x_k])(x - x_k) \\ &\quad + (\lim_{x_k^* \to x_k} f[x_k^*, x_k, x_{k+1}])(x - x_k)^2 \\ &\quad + (\lim_{x_k^* \to x_k, x_{k+1}^* \to x_{k+1}} f[x_k^*, x_k, x_{k+1}, x_{k+1}^*])(x - x_k)^2(x - x_{k+1}) \end{aligned}$$

We use the shorthand notation:

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

and construct 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.

$$\begin{aligned} x_k^* &\to x_k \\ x_{k+1}^* &\to x_{k+1} \\ f[x_k^*] = y_k^* &\to y_k \\ f[x_{k+1}^*] = y_{k+1}^* &\to y_{k+1} \end{aligned}$$

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

$$f[x_{k+1}, x_{k+1}^*] = \frac{f[x_{k+1}^*] - f[x_{k+1}]}{x_{k+1}^* - x_{k+1}} \xrightarrow[x_{k+1}^* \to x_{k+1}]{} f'(x_{k+1}) = y'_{k+1} \; !$$

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 reamaining 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.