Cubic Spline Interpolation

1 Objective

Given a set of points \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \), we would like to find a piece-wise interpolating polynomial \( C(x) \) that is continuous in its second derivative. We will define \( C(x) \) as follows:

\[
C(x) = \begin{cases} 
q_1(x) & \text{if } x_1 \leq x < x_2 \\
q_2(x) & \text{if } x_2 \leq x < x_3 \\
& \vdots \\
q_{n-1}(x) & \text{if } x_{n-1} \leq x < x_n
\end{cases}
\]

where \( q_i \) is the \( i \)-th local polynomial, interpolating the interval \([x_i, x_{i+1})\).

Because \( C(x) \) is continuous up to and including its second derivative, the following conditions must hold for all \( i = 1, 2, \ldots, n - 2 \):

\[
\begin{align*}
q_i(x_{i+1}) &= q_{i+1}(x_{i+1}) = y_{i+1} \\
q'_i(x_{i+1}) &= q'_{i+1}(x_{i+1}) \\
q''_i(x_{i+1}) &= q''_{i+1}(x_{i+1})
\end{align*}
\]

The piece-wise interpolant \( C(x) \) as described above will provide us with a nice local fit for the data, while remaining smooth on the entire interval from \( x_1 \) to \( x_n \). However, we have yet to figure out a way to compute such \( C(x) \) for any given set of points \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \).

2 An Initial Idea: Cubic Hermite Interpolation

Suppose that the data points \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \) correspond to the values of some function \( f(x) \) evaluated at \( x_1, x_2, \ldots, x_n \). Suppose furthermore that \( f \) is continuous in its first derivative, and let \( s_1, s_2, \ldots, s_n \) be the values of \( f'(x) \) at \( x_1, x_2, \ldots, x_n \), respectively.

The idea behind Cubic Hermite interpolation is the following: given \( (x_i, y_i, s_i) \), we will fit the data with piece-wise cubic polynomials of form:

\[
q_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 + (x - x_{i+1}) \quad (i = 1, 2, \ldots, n - 1)
\]

so that for all \( i = 1, 2, \ldots, n - 2 \),

\[
\begin{align*}
q_i(x_{i+1}) &= q_{i+1}(x_{i+1}) = y_{i+1} \\
q'_i(x_{i+1}) &= q'_{i+1}(x_{i+1}) = s_{i+1}
\end{align*}
\]

From equations (4) and (5), it immediately follows that the Hermite interpolant is continuous in its first derivative, regardless of the values of \( y_i \)'s and \( x_i \)'s.
To determine the coefficients of each of the Hermite polynomials, let us first concentrate on finding $a$, $b$, $c$, and $d$ for just a single local Hermite cubic on the interval between two given points $x_L$ and $x_R$, given the function values $y_L$ and $y_R$, and derivative values $s_L$ and $s_R$. Since the Hermite cubic has the form:

$$q(x) = a + b(x - x_L) + c(x - x_L)^2 + d(x - x_L)^2(x - x_R)$$

its first derivative is

$$q(x) = b + 2c(x - x_L) + 2d(x - x_L)(x - x_R) + d(x - x_L)^2$$

Now, plugging in $x = x_L$ and $x = x_R$ into the above equations, we get

$$y_L = q(x_L) = a$$
$$s_L = q'(x_L) = b$$
$$y_R = q(x_R) = a + b\Delta x + c(\Delta x)^2$$
$$s_R = q'(x_R) = b + 2c\Delta x + d(\Delta x)^2$$

where $\Delta x = x_R - x_L$. Rearranging and performing some simple algebra, we compute the values of $a$, $b$, $c$, and $d$ as follows:

$$a = y_L$$
$$b = s_L$$
$$c = \frac{y_R - a - b\Delta x}{(\Delta x)^2} = \frac{\Delta x (\frac{y_R - y_L}{\Delta x} - s_L)}{(\Delta x)^2} = \frac{y' - s_L}{\Delta x}$$
$$d = \frac{s_R - b - 2c\Delta x}{(\Delta x)^2} = \frac{s_R - s_L - 2(\frac{y' - s_L}{\Delta x})\Delta x}{(\Delta x)^2} = \frac{s_R + s_L - 2y'}{(\Delta x)^2}$$

where $y' = \frac{y_R - y_L}{x_R - x_L}$. Thus, for any two adjacent points $x_L$ and $x_R$, we now have the procedure to compute the coefficients $a$, $b$, $c$ and $d$ of the local Hermite cubic. Therefore, by setting $x_L = x_i$ and $x_R = x_{i+1}$ for $i = 1, 2, \ldots, n$, the $i$th local Hermite cubic is given by:

$$q_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^2(x - x_{i+1})$$

with coefficients

$$a_i = y_i$$
$$b_i = s_i$$
$$c_i = \frac{y'_i - s_i}{\Delta x_i}$$
$$d_i = \frac{s_i + s_i - 2y'_i}{(\Delta x_i)^2}$$

where $\Delta x_i = x_{i+1} - x_i$ and $y'_i = \frac{y_{i+1} - y_i}{\Delta x_i}$. 

2
3 Towards the Cubic Spline

Although cubic Hermite interpolation is useful in itself, it is not the solution to our particular problem. There are two drawbacks to the Hermite cubic that prevent us from using it.

First, in order to perform Hermite interpolation, we must know the derivatives of the hypothetical “true” function. More often than not, however this data will not be available to us. For instance, in the case of protein structures, we know the $x$, $y$, and $z$ coordinates of each $C_{\alpha}$ atom, but it does not even make sense to talk about the derivatives of the coordinates at each amino acid position.

Secondly, as equations (4) and (5) demonstrate, the Hermite interpolant is guaranteed to be continuous only in the first derivative. However, we want our interpolant to be continuous in the second derivative on the entire open interval $(x_1, x_n)$.

Fortunately for us, when taken together, these two problems help us emerge with a solution. To arrive at this solution, we must first examine two key insights. The first is that, following equations (4) and (5), the Hermite interpolant is continuous in its first derivative regardless of the values of the $s_i$’s, since for all $i$, $q_i(x_{i+1}) = q_{i+1}(x_{i+1})$. Therefore, for any choice of $s_i$’s, setting the coefficients $a_i$, $b_i$, $c_i$, and $d_i$ as prescribed by the equations (6)-(9) ensures continuity in the first derivative. (You can verify this algebraically: however, the algebra is lengthy, so we skip it here.) Thus, if we treat $s_i$’s as unknowns, we have an unconstrained system of equations: any choice of $s_i$’s leads to a valid Hermite interpolant.

The second insight is that by requiring the interpolant to be continuous in the second derivative, we can create a constrained system of equations that ultimately allow us to uniquely determine the piece-wise cubic interpolant. By requiring continuity in the second derivative, we add the following constrains on our interpolant:

$$q_i''(x_{i+1}) = q_{i+1}''(x_{i+1}) \quad \text{For } i = 1, 2, \ldots, n - 2$$

These new constrains correspond naturally to a system of equations that, with some additional adjustments, can be solved to produce unique values of $\{s_1, s_2, \ldots, s_n\}$, and therefore, the unique sought-after interpolant for given values of $\{(x_i, y_i)\}$. This interpolant is called the cubic spline.

To see how we could obtain the cubic spline, let us first compute the second derivative of the local cubics $q_i(x)$ and $q_{i+1}(x)$.

$$q_i''(x) = \frac{y_i'' - s_i}{\Delta x_i} + \frac{s_i + s_{i+1} - 2y_i'}{(\Delta x_i)^2}(4(x - x_i) + 2(x - x_{i+1}))$$

$$q_{i+1}''(x) = \frac{y_{i+1}'' - s_{i+1}}{\Delta x_{i+1}} + \frac{s_{i+1} + s_{i+2} - 2y_{i+1}'}{(\Delta x_{i+1})^2}(4(x - x_{i+1}) + 2(x - x_{i+2}))$$

Evaluating $q_i''(x)$ at $x = x_{i+1}$, we get

$$q_i''(x_{i+1}) = \frac{y_i'' - s_i}{\Delta x_i} + \frac{s_i + s_{i+1} - 2y_i'}{(\Delta x_i)^2}(4\Delta x_i)$$
\[
q''_{i+1}(x) = \frac{2}{\Delta x_i} (2s_{i+1} + s_i - 3y'_{j})
\]

Now, computing \(q''_{i+1}(x)\) at \(x = x_{i+1}\), we get

\[
q''_{i+1}(x_{i+1}) = \frac{2y_{i+1} - s_{i+1}}{\Delta x_{i+1}} + \frac{s_{i+1} + s_{i+2} - 2y'_{i+1}}{(\Delta x_{i+1})^2}(-2\Delta x_{i+1})
\]

\[
= \frac{2}{\Delta x_{i+1}}(3y'_{i+1} - 2s_{i+1} - s_{i+2})
\]

Since \(q''_{i}(x_{i+1}) = q''_{i+1}(x_{i+1})\) for \(i = 1, 2, \ldots, n-2\),

\[
\frac{2}{\Delta x_i}(2s_{i+1} + s_i - 3y'_{j}) = \frac{2}{\Delta x_{i+1}}(3y'_{i+1} - 2s_{i+1} - s_{i+2})
\]

If we simplify the equation and re-arrange the terms so that all the unknowns \((s_i)'s\) are on the left-hand side, then for \(i = 1, 2, \ldots, n-2\), we get:

\[
\Delta x_{i+1}s_i + 2(\Delta x_i + \Delta x_{i+1})s_{i+1} + \Delta x_is_{i+2} = 3(\Delta x_{i+1}y_i + \Delta x_iy'_{i+1})
\]  \hspace{1cm} (11)

Thus, we have \(n-2\) linear equations of the form of equation (11). However, there are \(n\) unknowns \(\{s_1, s_2, \ldots, s_n\}\), so the system still possesses 2 degrees of freedom, and therefore does not have a unique solution. However, if we apply additional constraints to \(s_1\) and \(s_n\), we can obtain a fully qualified system of equations, and can solve for \(\{s_1, s_2, \ldots, s_n\}\) uniquely. Once the values of the \(s_i\)'s are found, we can plug them into equations (6)-(9), to obtain the coefficient values \(a_i, b_i, c_i,\) and \(d_i\) for each local cubic polynomial \(q_i(x)\).

There several ways we apply additional constrains to \(s_1\) and \(s_n\), and the spline interpolant we end up with depends on the method that we choose. We briefly discuss three such methods below. Verifying the algebra is left as an exercise.

**Complete Spline** This is the simplest method. We simply set \(s_1 = \mu_L\) and \(s_n = \mu_R\), where \(\mu_L\) and \(\mu_R\) are arbitrary given values.

**Natural Spline** Rather than prescribing values to \(s_1\) and \(s_n\) directly, we prescribe the values for the second derivative of the spline at the interval endpoints \(x_1\) and \(x_n\). That is, for given arbitrary values \(\mu_L\) and \(\mu_R\), let \(q''_1(x_1) = \mu_L\) and \(q''_{n-1}(x_n) = \mu_R\). This results in two new equations added to our system, making it uniquely solvable. If the values \(\mu_L = \mu_R = 0\), the spline is called the natural spline.

**Not-a-Knot Spline** We introduce two additional equations by enforcing not only second, but third derivative continuity at two points: \(x_2\) and \(x_n-1\). This makes our system fully qualified, and therefore uniquely solvable. This is the method used by the Matlab spline( ) function.

For a more rigorous treatment of spline interpolation, complete with detailed Matlab examples on setting up and solving the linear equations to compute the spline, see the third chapter of *Introduction to Scientific Computing*, by Charles Van Loan.