

Piecewise Cubic Example

Consider interpolating the points

i	0	1	2	3	4
x_i	0.00	0.10	0.25	0.50	1.00
h_i	0.10	0.15	0.25	0.50	–
y_i	1.0000	0.2857	0.1379	0.0741	0.0385

Table 1: Data for the cubic spline.

where $h_i = x_{i+1} - x_i$ and the y_i were generated from Runge's function

$$y = \frac{1}{1 + 25x^2} \quad (1)$$

We will interpolate this data with a cubic spline using the x_i as the breakpoints. We will use the following spline as our basis

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3,$$

for $i = 0, \dots, r-1$ where $r = 4$ in this case. So there are four cubic interpolants, and 16 coefficients to determine. We need the derivatives later, so they are:

$$\begin{aligned} s'_i(x) &= b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2, \\ s''_i(x) &= 2c_i + 6d_i(x - x_i). \end{aligned}$$

Consider first the interpolation conditions on the left side of the interval: $s_i(x_i) = y_i$, $i = 0, \dots, r-1$. Noticing that $(x - x_i)|_{x=x_i} = (x_i - x_i) = 0$, we get the set of equations:

$$\begin{aligned} s_0(x_0) &= a_0 = y_0, \\ s_1(x_1) &= a_1 = y_1, \\ s_2(x_2) &= a_2 = y_2, \\ s_3(x_3) &= a_3 = y_3. \end{aligned}$$

Now consider the interpolation conditions on the right side of the interval: $s_i(x_{i+1}) = y_{i+1}$, $i = 0, \dots, r-1$. Noticing that $(x - x_i)|_{x=x_{i+1}} = (x_{i+1} - x_i) = h_i$, we get the set of equations:

$$\begin{aligned} s_0(x_1) &= a_0 + b_0h_0 + c_0h_0^2 + d_0h_0^3 = y_1, \\ s_1(x_2) &= a_1 + b_1h_1 + c_1h_1^2 + d_1h_1^3 = y_2, \\ s_2(x_3) &= a_2 + b_2h_2 + c_2h_2^2 + d_2h_2^3 = y_3, \\ s_3(x_4) &= a_3 + b_3h_3 + c_3h_3^2 + d_3h_3^3 = y_4. \end{aligned}$$

So now we have 8 equations for the 16 coefficients. In order to get the other equations, we impose continuity conditions. First, the first derivatives: $s'_i(x_{i+1}) = s'_{i+1}(x_{i+1})$, $i = 0, \dots, r - 2$.

$$\begin{aligned} s'_0(x_1) &= b_0 + 2c_0h_0 + 3d_0h_0^2 = b_1 = s'_1(x_1), \\ s'_1(x_2) &= b_1 + 2c_1h_1 + 3d_1h_1^2 = b_2 = s'_2(x_2), \\ s'_2(x_3) &= b_2 + 2c_2h_2 + 3d_2h_2^2 = b_3 = s'_3(x_3), \end{aligned}$$

Notice that we only get three equations out of this condition, since we cannot impose continuity at the outside knots x_0 and x_4 .

Now the second derivatives: $s''_i(x_{i+1}) = s''_{i+1}(x_{i+1})$, $i = 0, \dots, r - 2$.

$$\begin{aligned} s''_0(x_1) &= 2c_0 + 6d_0h_0 = 2c_1 = s''_1(x_1), \\ s''_1(x_2) &= 2c_1 + 6d_1h_1 = 2c_2 = s''_2(x_2), \\ s''_2(x_3) &= 2c_2 + 6d_2h_2 = 2c_3 = s''_3(x_3), \end{aligned}$$

So now we have a total of 14 equations for 16 unknowns. Let's write them out in matrix form (where all the 0 entries are replaced by a dot to make the nonzero entries easier to see):

$$\begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & h_0 & \cdot & \cdot & \cdot & h_0^2 & \cdot & \cdot & h_0^3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & h_1 & \cdot & \cdot & \cdot & h_1^2 & \cdot & \cdot & h_1^3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & h_2 & \cdot & \cdot & \cdot & h_2^2 & \cdot & \cdot & h_2^3 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & h_3 & \cdot & \cdot & \cdot & h_3^2 & \cdot & \cdot & h_3^3 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot & \cdot & 2h_0 & \cdot & \cdot & \cdot & 3h_0^2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot & \cdot & 2h_1 & \cdot & \cdot & \cdot & 3h_1^2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot & \cdot & 2h_2 & \cdot & \cdot & \cdot & 3h_2^2 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & -2 & \cdot & \cdot & 6h_0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & -2 & \cdot & \cdot & 6h_1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & -2 & \cdot & \cdot & 6h_2 & \cdot \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_1 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2)$$

$A_1 \qquad w = u_1$

The rows correspond to the equations listed previously in the same order. Make sure that you understand how this system is formed from the previous equations. Also notice that A_1 is rectangular 14×16 —count it yourself, don't just look at its shape on the page, since the width of the columns is not the same as the height of the rows. Therefore, an infinite number of coefficient vectors w will solve this system. We need two more equations before w (and hence our spline) will be unique. There are several options, but note that each option provided below generates the necessary two rows and hence the options are mutually exclusive.

One option is a “natural spline” by setting the derivatives at the first and last data point to be zero:

$$\begin{aligned} s'_0(x_0) &= b_0 = 0, \\ s'_3(x_4) &= b_3 + 2c_3h_3 + 3d_3h_3^2 = 0 \end{aligned} \quad (3)$$

As mentioned in class, such a choice *may* degrade the order of accuracy of the approximation from $\mathcal{O}(h^4)$ to $\mathcal{O}(h^2)$ near the ends of the spline. Since this example has only four segments, “near” the ends of the spline is almost the entire spline, so it probably is not a good idea to use a “natural” spline. However, as we shall see below, maybe it isn’t too bad. You should be able to easily translate (3) into two rows to add to (2) and make a square system.

A second option is the “not a knot” condition, that the third derivatives match at the first interior knots on each end. In this case:

$$\begin{aligned} s_0'''(x_1) &= 6d_0 = 6d_1 = s_1'''(x_1) \\ s_2'''(x_3) &= 6d_2 = 6d_3 = s_3'''(x_3) \end{aligned} \tag{4}$$

While this choice does not degrade the order of accuracy of the approximation, it does in effect mean that we are using only two separate interpolants, rather than four. Again, you should easily be able to translate (4) into two rows to add to (2).

A third option is to match the derivatives at the endpoints. Using (1) we find that $y'_0 = 0$ and $y'_4 = -0.0030$:

$$\begin{aligned} s'_0(x_0) &= b_0 = y'_0, \\ s'_3(x_4) &= b_3 + 2c_3h_3 + 3d_3h_3^2 = y'_4 \end{aligned} \tag{5}$$

This option would probably be the most accurate, since it uses the most information about the underlying function. However, it needs additional information beyond that provided by table 1. Again, from (5) we can get two rows to add to (2).

Finally, we could be more creative. Let us assume that we know that the underlying function generating the data is differentiable and symmetric, but we do not know its actual form. Therefore, we do not know its general derivative and cannot use (5). But a symmetric differentiable function must have derivative zero at the origin (why?). So we can use the exact derivative on the left end of the interval, and then not a knot at the other end where we have no additional information:

$$\begin{aligned} s'_0(x_0) &= b_0 = 0, \\ s_2'''(x_3) &= 6d_2 = 6d_3 = s_3'''(x_3) \end{aligned} \tag{6}$$

You could consider these some weird new “mixed” conditions, but the point is that we have used the knowledge that we have about the problem—the data is symmetric and differentiable—where we can, and we used the most accurate approximation elsewhere.

To finish the example, we take the two additional rows from (6) and put them on the bottom of (2):

$$\begin{array}{c}
 \left[\begin{array}{cccccccccccccccc}
 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
 . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . \\
 . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\
 . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\
 1 & . & . & . & h_0 & . & . & . & h_0^2 & . & . & . & h_0^3 & . & . \\
 . & 1 & . & . & . & h_1 & . & . & . & h_1^2 & . & . & . & h_1^3 & . \\
 . & . & 1 & . & . & . & h_2 & . & . & . & h_2^2 & . & . & . & h_2^3 \\
 . & . & . & 1 & . & . & . & h_3 & . & . & . & h_3^2 & . & . & . & h_3^3 \\
 . & . & . & . & 1 & -1 & . & . & 2h_0 & . & . & . & 3h_0^2 & . & . \\
 . & . & . & . & . & 1 & -1 & . & . & 2h_1 & . & . & . & 3h_1^2 & . \\
 . & . & . & . & . & . & 1 & -1 & . & . & 2h_2 & . & . & . & 3h_2^2 \\
 . & . & . & . & . & . & . & . & 2 & -2 & . & . & 6h_0 & . & . \\
 . & . & . & . & . & . & . & . & . & 2 & -2 & . & . & 6h_1 & . \\
 . & . & . & . & . & . & . & . & . & . & 2 & -2 & . & . & 6h_2 \\
 . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . \\
 . & . & . & . & . & . & . & . & . & . & . & . & . & 6 & -6
 \end{array} \right]
 \begin{array}{c}
 \left[\begin{array}{c}
 a_0 \\
 a_1 \\
 a_2 \\
 a_3 \\
 b_0 \\
 b_1 \\
 b_2 \\
 b_3 \\
 c_0 \\
 c_1 \\
 c_2 \\
 c_3 \\
 d_0 \\
 d_1 \\
 d_2 \\
 d_3
 \end{array} \right]
 =
 \begin{array}{c}
 \left[\begin{array}{c}
 y_0 \\
 y_1 \\
 y_2 \\
 y_3 \\
 y_1 \\
 y_2 \\
 y_3 \\
 y_4 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0
 \end{array} \right]
 \end{array}
 \quad (7)
 \end{array}
 \end{array}$$

Notice that the vector of coefficients w did not change. Now we have a square matrix. The pattern of nonzero entries in the matrix is not particularly pretty (for example, not tridiagonal), so it may not be immediately obvious how to solve the system efficiently. It is also small enough (16×16) that MATLAB can solve it essentially instantly treating the matrix as dense.¹

The file `piecewiseCubic.m` implements (7) in MATLAB, and the results are shown in figure 1. Notice that the results are very good to the left where there are more data points and we had an exact condition with which to build one of the two missing equations. The results are much poorer to the right, where there are fewer data points and the not a knot condition essentially requires a single cubic spline from $x = 0.2$ to $x = 1.0$.

¹These days desktop computers are fast enough that you only need to use sparse matrices for systems with hundreds of rows. But if you were trying to fit hundreds of data points, it is worth knowing that because there are only a constant number of nonzero entries per row—a cubic spline problem could never have more than eight, since no row will involve more than two splines, and each spline has four coefficients—there are techniques to solve this system in $\mathcal{O}(n)$ time, where n is the number of rows. But you can take CPSC 402 if you are interested in those details.

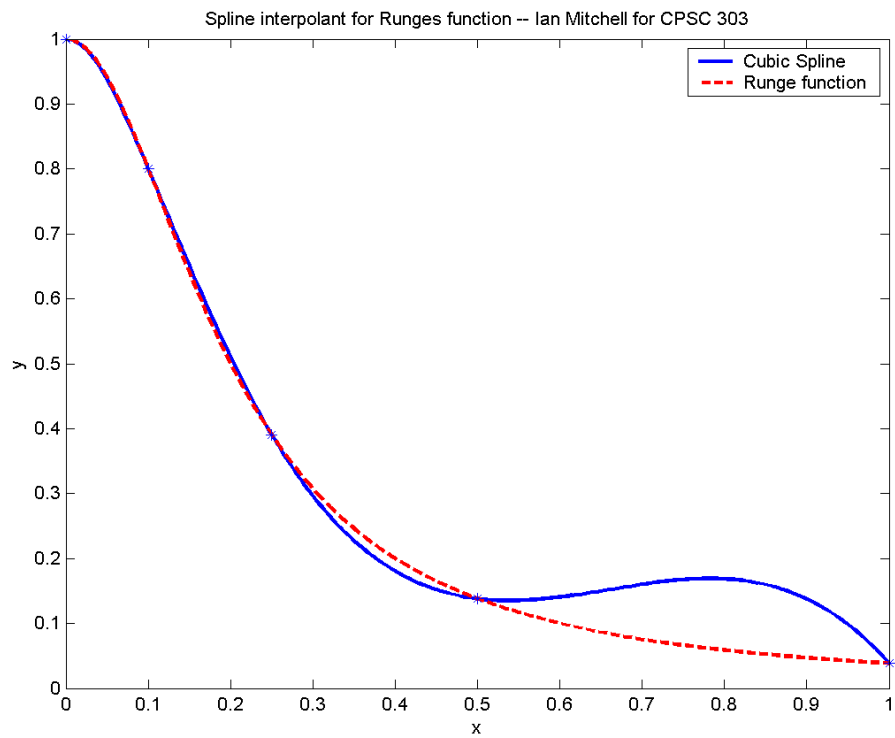


Figure 1: Cubic spline approximation of Runge's function.