Models, Algorithms and Data (MAD): Introduction to Computing

Spring semester 2019

Prof. Dr. Jens Honore Walther
Dr. Julija Zavadlav
ETH Zentrum, CLT
CH-8092 Zürich

Set 4

Issued: 15.03.2019; Due: 24.03.2019

In this exercise, you will learn about fitting data using Lagrange interpolation and cubic splines. In the first problem, you will interpolate a function at equally spaced points with Lagrange interpolation. In doing so, you will observe Runge's phenomenon, namely interpolation divergence that occurs near the endpoints when using this technique. Motivated by this problem, you will learn about using cubic splines which offer us a more robust and oscillation free method for data interpolation. In the second problem, you will derive the matrix equation for computing interpolating splines with clamped boundary conditions, and in the final problem, you will investigate the effect of using different boundary conditions on the results of cubic spline interpolation. Before providing the problems, we provide a review of the theory behind Lagrange interpolation and cubic spline interpolation below:

1 Lagrange Interpolation Review

Given a set of $N+1$ data points:

$$\{(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)\}$$

the Lagrange interpolating polynomial function is given by:

$$L(x) = \sum_{i=0}^{N} y_i \cdot l_i(x)$$

with each $l_i(x)$ given by:

$$l_i(x) = \prod_{\substack{0 \le m \le N \\ m \ne i}} \left(\frac{x - x_m}{x_i - x_m} \right)$$

It should be clear from this formula that $l_i(x_j) = 1$ when $i = j$ and is exactly zero when $i \neq j$. Thus $L(x)$ interpolates the $N+1$ data points perfectly (i.e., $L(x_0) = y_0$, $L(x_1) = y_1$, and so on). This looks like a very attractive result! We have a simple closed form expression for an interpolating function that **perfectly** interpolates all of our data points. As you will discover in the exercise, due to the high order polynomial terms in the interpolating functions, the model is prone to oscillating near the endpoints (Runge's Phenomenon).

<u>**Note:**</u> This oscillatory problem can be addressed by intelligently choosing interpolation points at Chebyshev nodes, although we will not address this in this exercise.

2 Cubic Spline Review

The basic idea with splines is to generate polynomial segments for each interval (i.e., the space between any two data points). The known data points are also referred to as 'nodes'. Thus for N data points, you end up with $N-1$ distinct cubic polynomial segments in the $N-1$ intervals. This is shown in Fig. 1, where the 4 known data points enable us to construct 3 different cubic polynomials in the 3 intervals.

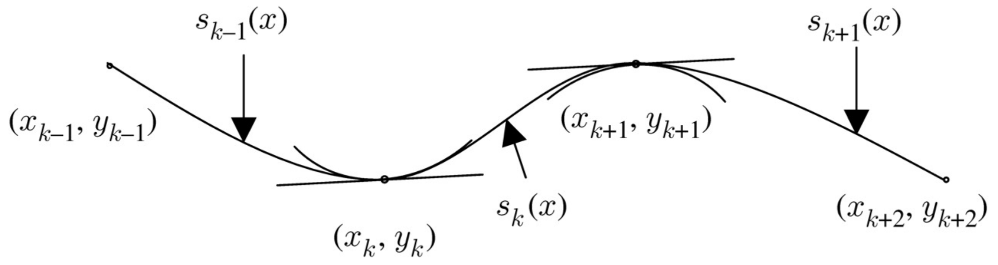

Figure 1: A schematic showing the concept behind cubic splines (source: http://rspa.royalsocietypublishing.org/content/464/2094/1483). The first and second derivatives of the two distinct cubic polynomials, $S_{k-1}(x)$ and $S_k(x)$, must be equal at (x_k, y_k), and those for $S_k(x)$ and $S_{k+1}(x)$ must be equal at (x_{k+1}, y_{k+1}). (*Remark*: Index 'k' shown in this figure is equivalent to index 'i' in Eq. 1 below.)

As done in the lecture notes, these polynomials can be computed by imposing the requirements that the polynomials to the left and right of each node have the same first and second derivative values at the shared node. The resulting polynomial segments take the form:

$$f(x) = f_i''\frac{(x_{i+1}-x)^3}{6\Delta_i} + f_{i+1}''\frac{(x-x_i)^3}{6\Delta_i} + \left(\frac{y_{i+1}-y_i}{\Delta_i} - (f_{i+1}''-f_i'')\frac{\Delta_i}{6}\right)(x-x_i) + \left(y_i - f_i''\frac{\Delta_i^2}{6}\right) \tag{1}$$

The only remaining task in determining the analytic form of the polynomial segments is to estimate the f'' values at all the nodes. This requires us to solve the tri-diagonal matrix formed by the equation:

$$\begin{aligned} \frac{\Delta_{i-1}}{6}f_{i-1}'' + \left(\frac{\Delta_{i-1}+\Delta_i}{3}\right)f_i'' + \frac{\Delta_i}{6}f_{i+1}'' &= \frac{y_{i+1}-y_i}{\Delta_i} - \frac{y_i-y_{i-1}}{\Delta_{i-1}} && (2)\\ \Rightarrow A_i f_{i-1}'' + B_i f_i'' + C_i f_{i+1}'' &= D_i && (3) \end{aligned}$$

Assuming that we are given only 4 node points (as shown in Fig. 1), the required system takes the following form:

$$\begin{bmatrix} B_2 & C_2 \\ A_3 & B_3 \end{bmatrix} \cdot \begin{bmatrix} f_2'' \\ f_3'' \end{bmatrix} = \begin{bmatrix} D_2 \\ D_3 \end{bmatrix} \tag{4}$$

As discussed in the lecture, some sort of user-intervention is required to compute f_1'' and f_4''. For natural cubic splines, these two values are set to be zero.

Note: In general, for a large number of data points, the system of equations shown in Eq. 3 requires the solution of a tri-diagonal matrix (all elements zero, except for the three diagonals).

Note: The A_i, B_i, C_i, and D_i values shown in Eq. 3 are the same at every node if and only if the nodes are equidistant (i.e., Δ_i is constant).

Question 1: Lagrange Interpolation and Runge's Phenomena

Consider the function:

$$f(x) = \frac{1}{\sqrt{1+x^2}}$$

evaluated at 11 evenly spaced points on the interval $[-5, 5]$. Generate the data points and plot the data points along with the true function. You should see something similar to Figure 2.

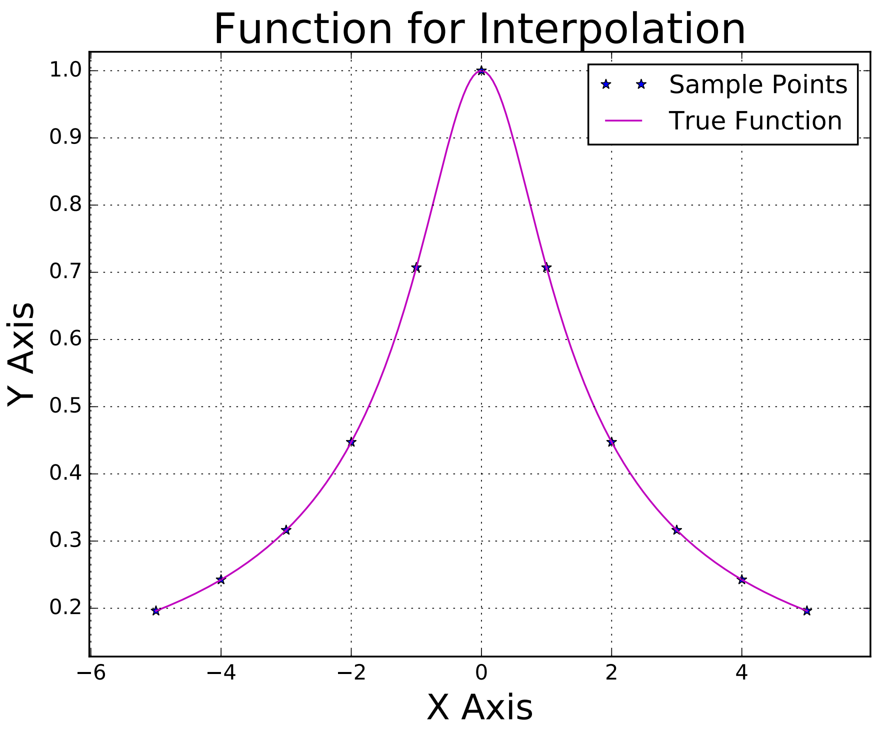

Figure 2: Visualization of Function and Interpolation Points

For this problem, you will compute the Lagrange interpolating polynomial using these 11 data points, and then plot the results of the interpolation against the true function. To plot the result of the Lagrange interpolation use large number of points, e.g., 200. How well does the model fit the data? Would you trust this fit for extrapolation or interpolation of these data points?

Hint: When addressing this problem, try to write helper functions that make your work easier. For example, try to write a general helper function that computes the individual Lagrange interpolants (i.e., the $l_i(x)$ functions from above), and then call this helper function multiple times to find the full Lagrange interpolating function, $L(x)$.

Question 2: Cubic splines with clamped end conditions

Re-derive the matrix system shown in Eq. 4 for interpolating cubic spline with clamped end points. 'Clamped' refers to restricting the first derivatives at the end points to be zero, i.e., $f'_1, f'_4 = 0$.

Hint: Natural cubic splines ($f''_1, f''_4 = 0$) may give rise to non-zero derivatives at the end points. Our goal now is to enforce the derivatives at the end points to be zero ($f'_1, f'_4 = 0$). To determine how this condition affects the second derivatives at $i = 1$ and $i = 4$, we can use the relation

$$\text{for } x_i \le x \le x_{i+1}\text{:}\;\; f'(x) = f''_{i+1}\left[\frac{(x-x_i)^2}{2\Delta_i} - \frac{\Delta_i}{6}\right] - f''_i\left[\frac{(x_{i+1}-x)^2}{2\Delta_i} - \frac{\Delta_i}{6}\right] + \frac{y_{i+1}-y_i}{\Delta_i}$$

and evaluate it at $x = x_1$ or $x = x_4$, respectively.

Question 3: The impact of using different boundary conditions

You are given x and y values at 4 nodes: $x = [0, 1, 2, 3]$, $y = [1, 0, 0.3, 0]$.

a) Write a code to interpolate this dataset using:

- Natural cubic splines
- Cubic splines with right end clamped (you can leave the left end to be 'free', i.e., use $f'' = 0$ at the left-most node).
- Cubic splines with both ends clamped

Generate 100 interpolation points in between the first and the last nodes. Evaluate your piecewise polynomials at these interpolation points and generate plots to compare the results.

Note: When computing splines with clamped end conditions, the matrix system will be larger than the one shown in Eq. 4. For instance, with the right end clamped, you will need to include f''_4 in the variables to be solved for, and with both ends clamped, you will need to include both f''_1 and f''_4.

b) We denote the cubic polynomial corresponding to the interval $[x_k, x_{k+1}], k = 1, ..., N-1$ as $S_k(x) = a_{k,0} + a_{k,1}(x - x_k) + a_{k,2}(x - x_k)^3 + a_{k,3}(x_{k+1} - x)^3$. Use the code to find the coefficients of the $S_k(x)$ for all segments and all three cases. Manually (i.e., on paper), compute the first and second derivatives of the $S_k(x)$ and check that both derivatives are continuous at the nodes.