Numerical Methods Homework 9

Adam Kit - 3707437

15 June 2020

1 Cubic Spline Interpolation

We aim to interpolate the data table given in problem 1 by constructing a cubic spline. A cubic spline fits a set of data points with $n-1$ cubic polynomials S_i for $i = 0, \ldots, n-1$, as defined in 1

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{1}$$

Where the cubic polynomial S_i is on the domain $[x_i, x_{i+1}]$ except for when $i = n-1$, then the domain is $[x_i, x_n]$. Note the $3(n-1)$ unknowns we must solve for in 1. The polynomials will have the following properties.

1. $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$ for $i = 0, 1, \ldots, n-1$
2. $S'_{i-1}(x_i) = S'_i(x_i)$ for $i = 1, \ldots, n-1$
3. $S''_{i-1}(x_i) = S''_i(x_i)$ for $i = 1, \ldots, n-1$

Property 1 says that the interpolation passes through all the data points, and 2 and 3 force continuity at the end points of each interval. We can then impose the free boundry conditions of

$$S''_0(x_0) = S''_{n-1}(x_{n-1}) = 0$$

To find our coefficeints, we start with c_i:

$$\begin{bmatrix} 1 & 0 & 0 & & & \\ \delta_0 & 2(\delta_0 + \delta_1) & \delta_1 & \ddots & & \\ 0 & \delta_1 & 2(\delta_1 + \delta_2) & \delta_2 & & \\ 0 & 0 & \ddots & \ddots & \ddots & \\ & & & \delta_{n-2} & 2(\delta_{n-2} + \delta_{n-1}) & \delta_{n-1} \\ & & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 3(\Delta_1/\delta_1 - \Delta_0/\delta_0) \\ \vdots \\ 3(\Delta_{n-1}/\delta_{n-1} - \Delta_{n-2}/\delta_{n-2}) \end{bmatrix}$$

where $\delta_i = x_{i+1} - x_i$ and $\Delta_i = y_{i+1} - y_i$. b and d can be solved from

$$d_i = \frac{c_{i+1} - c_i}{3\delta_i}$$

$$b_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1})$$

For our examples, we have then 3 polynomials, and the coefficients for c can easily be solved since $\delta_i = 1$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ -27 \\ 0 \end{bmatrix}$$

This can be solved using the jacobi iteration method from last week, and we find $c_0 = c_3 = 0$, $c_1 = 3.4$, $c_2 = -7.6$, meaning $b_0 = 2.8666$, $b_1 = 6.2666$, $b_2 = 2.0666$ and $d_0 = 1.1333$, $d_1 = -3.666$, $d_2 = 2.5333$ and our polynomials are:

1. $S_0 = 1 + 2.866(x-1) + 1.133(x-1)^3$ on $[1, 2]$
2. $S_1 = 2 + 6.266(x-2) + 3.4(x-2)^2 - 3.66(x-2)^3$ on $[2, 3]$
3. $S_2 = 3 + 2.066(x-3) - 7.6(x-3)^2 + 2.533(x-3)^3$ on $[3, 4]$

So to get our value we can plug $x = 1.5$ into S_0 to find $y(1.5) = 2.29$.

2 Quadratic Splines

So we will have the polynomials of the quadratic spline as:

1. $p_1(x) = a_1 x^2 + b_1 x + c_1$ for $[-1, 0]$
2. $p_2(x) = a_2 x^2 + b_2 x + c_2$ for $[0, 1]$

At the continuity point $x = 0$, both functions must equal 1, i.e., $p_1(0) = p_2(0) = 1$, which will give us $c_1 = c_2 = 1$. Then taking the derivative of the first one and using the given condition $p_1'(-1) = 0$, we have

$$p_1'(x) = 2a_1 x + b_1 \Rightarrow p_1'(-1) = -2a_1 + b_1 = 0 \Rightarrow a_1 = b_1/2$$

. So we can find now b_1:

$$p_1(x) = b_1/2x^2 + b_1 x + 1 \Rightarrow p_1(-1) = b_1/2(-1)^2 + b_1(-1) + 1 = \frac{b_1}{2} - b_1 + 1 = 0 \Rightarrow b_1 = 2$$

so our first function is found to be:

$$p_1(x) = x^2 + 2x + 1$$

We also have the boundrary conditions of $p_1'(0) = p_2'(0)$, so

$$p_1'(x) = 2x + 2 \Rightarrow p_1'(0) = 2$$

$$p_2'(x) = 2a_2 x + b_2 \Rightarrow p_2'(0) = b_2 \rightarrow b_2 = 2$$

Finally we can insert the point $(1, 3)$ into p_2 to find a_2:

$$p_2(x) = a_2 x^2 + 2x + 1 \Rightarrow p_2(1) = a_2(1) + 2(1) + 1 = a_2 + 3 = 3 \rightarrow a_2 = 0$$

Our final quadratic spline is given below and is graphed in Figure 1:

1. $p_1(x) = x^2 + 2x + 1$ for $[-1, 0]$
2. $p_2(x) = 2x + 1$ for $[0, 1]$

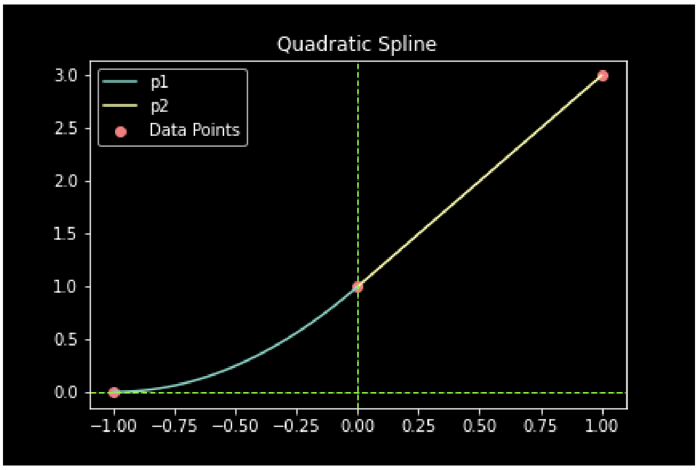

Figure 1: Quadratic Spine interpolation for problem 2

3 Value of Cosine from Sine Table

Given the table in problem 3, we can use lagrange interpolation to approximate $sin(x)$. Once approximated, we can use trig rules to deterime the value of $cos(x)$, namely:

$$1 = cos^2(x) + sin^2(x) \Rightarrow \pm\sqrt{1 - sin^2(x)} = cos(x)$$

The lagrange interpolation can be done using the functions found in Figure 2, which gives us $sin(1.74) = 0.9856$, from which we find

$$cos(1.74) = -\sqrt{1 - 0.9856^2} = 0.169$$

.

4 Find Minimum

We use the same interpolating scheme as in 2, and find the polynomial, take the derivative, set it equal to 0 and solve. Programatically this is achieved by finding the max value of the approximation and finding the x that represents that value. The maximum is found at the point $(5.69, 0.263)$ and a graphic can be seen in Figure 3.

```python
import numpy as np

def lagrangecoeff(x, y, n):
    indexed = x[n]
    denonimator = np.array([value for value in x if value != indexe
    denonimator = indexed - denonimator
    coef = y[n]
    for value in denonimator:
        coef /= value

    return coef
def polynomial(domain, x_values, y_values):
    N = len(x_values)
    sum = np.zeros_like(domain)
    lcoeffs = [lagrangecoeff(x_values, y_values, _) for _ in range(
    for i in range(N):
        cache_sum = np.ones_like(domain)*(lcoeffs[i])
        for j in range(N):
            if j == i:
                continue
            cache_sum *= domain - x_values[j]
        sum += cache_sum
    return sum
```

Figure 2: Python code for Lagrange Interpolation

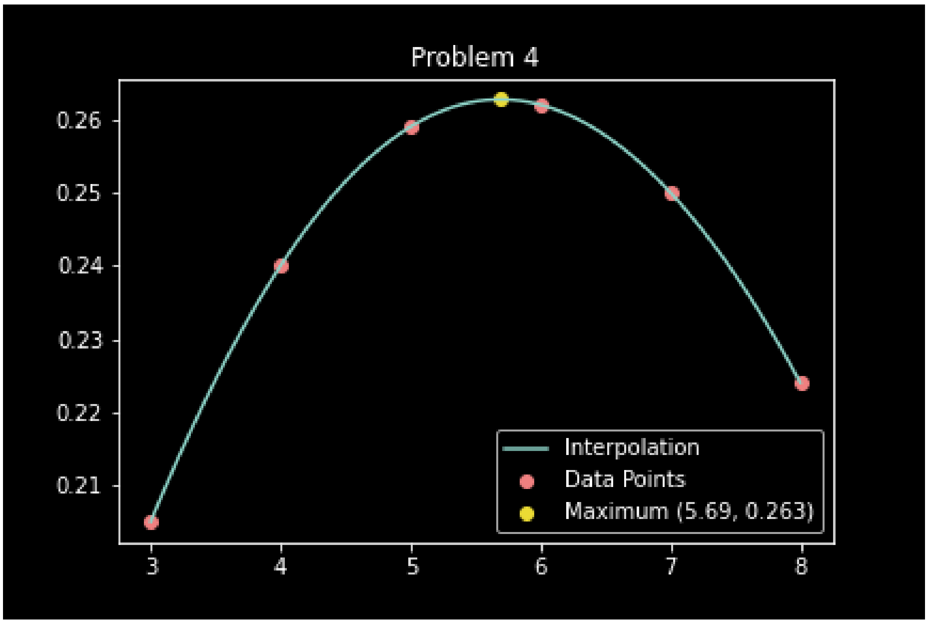

Figure 3: Lagrange Interpolation