Draft version February 11, 2026
Typeset using LaTeX **twocolumn** style in AASTeX7

Homework[1]: Interpolation

Haikun Xue[1]

[1] *University of Oklahoma, Department of Astronomy*

1. LINKS TO CODE

Here is a link to the code file https://github.com/HaikunXue/OU-ASTR-5900/blob/main/HW1/HW01.ipynb

2. QUESTION 1:

2.1. *Part a:*

As

$$\begin{cases} g_i(x) = a_i(x - x_i) + b_i \ (1) \\ g_i(x_i) = y_i \ (2) \\ g_i(x_{i+1}) = y_{i+1} \ (3) \end{cases}$$

Plug in (2) and (3) into (1):

$$\begin{cases} y_i = a_i(0) + b_i \\ y_{i+1} = a_i(x_{i+1} - x_i) + b_i \end{cases}$$

Simplify:

$$y_i - y_{i+1} = -a_i(x_{i+1} - x_i)$$
$$a_i = \frac{y_i - y_{i+1}}{x_{i+1} - x_i}$$

Q.E.D

2.2. *Part b:*

Deriving equation 3 and 4:
We have:

$$\begin{cases} g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \\ d_i = y_i \\ g_i'(x_{i+1}) = g_{x+1}'(x_{i+1}) \end{cases}$$

So

$$g_i'(x_{i+1}) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i$$

Then we can solve:

$$g_i'(x_{i+1}) = g_{i+1}'(x_{i+1})$$
$$3a_i(x_{i+1} - x_i)^2 + 2b_i(x_{i+1} - x_i) + c_i = 3a_{i+1} \times 0 + 2b_{i+1} \times 0 + c_{i+1}$$
$$3a_i(x_{i+1} - x_i)^2 + 2b_i(x_{i+1} - x_i) + c_i = c_{i+1}$$

as Jeffery defined in the video:
$h_i = x_{i+1} - x_i; \eta_i = y_{x+1} - y_i$
we have:

—

$$3a_ih_i^2 + 2b_ih_i + c_i = c_{i+1}$$

which is equation (3)!
Then we take second derivative of g(x)

$$g"_i(x_{i+1}) = g"_{i+1}(x_{i+1})$$
$$6a(x_{i+1} - x_i) + 2b_i = 6a_{i+1} \times 0 + 2b_{i+1}$$
$$6a(x_{i+1} - x_i) + 2b_i = 2b_{i+1}$$
$$3ah_i + b_i = b_{i+1}$$

which is equation (4)!

2.3. *Part c:*

we start from

$$\begin{cases} d_i = y_i \to 1) \\ a_ih_i^3 + b_ih_i^2 + c_ih_i = \eta_i \to 2) \\ 3a_ih_i + b_i = b_{i+1} \to 3) \\ 3a_ih_i^2 + 2b_ih_i + c_i = c_{i+1} \to 4) \end{cases}$$

We solve 3) for a_i :

$$3a_ih_i + b_i = b_{i+1}$$
$$a_i = \frac{b_{i+1} - b_i}{3h_1}$$

Then solve 2) for c_i :

$$a_ih_i^3 + b_ih_i^2 + c_ih_i = \eta_i$$
$$\frac{b_{i+1} - b_i}{3h_1}h_i^3 + b_ih_i^2 + c_ih_i = \eta_i$$
$$c_i = \frac{1}{h_i}(\eta_i - \frac{b_{i+1} - b_i}{3}h_i^2 - b_ih_i^2)$$
$$c_i = \frac{\eta_i}{h_i} - \frac{b_{i+1} - b_i}{3}h_i - b_ih_i$$

Finally, plug both a_i& c_i into 4):

$$3a_ih_i^2 + 2b_ih_i + c_i = c_{i+1}$$
$$3\frac{b_{i+1} - b_i}{3h_1}h_i^2 + 2b_ih_i + \frac{\eta_i}{h_i} - \frac{b_{i+1} - b_i}{3}h_i - b_ih_i =$$
$$\frac{\eta_{i+1}}{h_{i+1}} - \frac{b_{i+2} - b_{i+1}}{3}h_{i+1} - b_{i+1}h_{i+1}$$

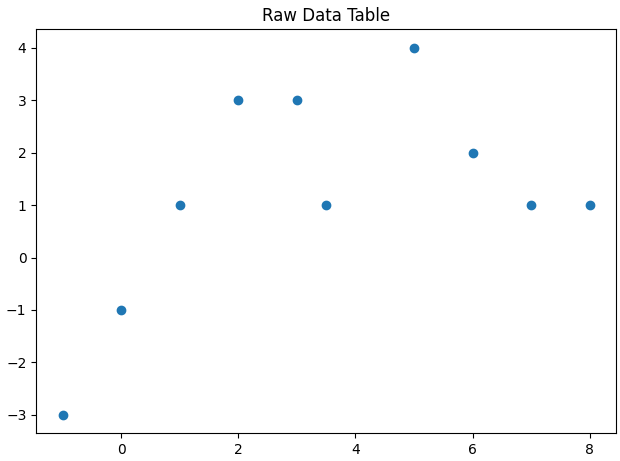

Figure 1. The dataset by itself

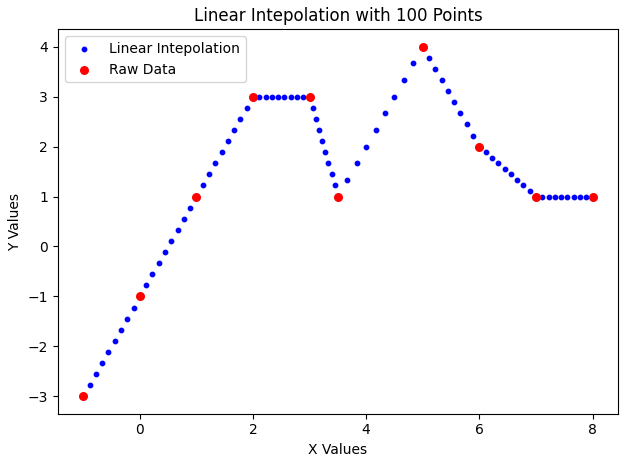

Figure 2. Linear Interpolation by Hand

Collect terms:

$$\begin{cases} \text{for } b_i : -h_i + 2h_i + \frac{1}{3}h_i - h_i = \frac{1}{3}h_i \\ \text{for } b_{i+1} : h_i - \frac{1}{3}h_i - \frac{1}{3}h_{i+1} + h_{i+1} = \frac{2}{3}(h_i - h_{i+1}) \\ \text{for } b_{i+2} : +\frac{1}{3}h_{i+1} \end{cases}$$

Then

$$\frac{1}{3}h_i b_i + \frac{2}{3}(h_i - h_{i+1})b_{i+1} + + \frac{1}{3}h_{i+1}b_{i+2} = \frac{\eta_{i+1}}{h_{i+1}} - \frac{\eta_i}{h_i}$$

which is the equation in the video!
QED.

3. QUESTION 2

Figure 1 shows the raw dataset.

3.1. *Part a:*

Figure 2 shows my linear interpolation by hand of the dataset. To achieve 10 times higher resolution, I sampled 10 data points between each original data points. In my code, I assume we are only dealing with points within the boundary of our original dataset, so no extrapolation.

3.2. *Part b:*

Figure 3 Shows the comparison between cubic interpolation function from Scipy.

3.3. *Part c:*

The spline sometimes predicts higher or lower values. This can be correct when we deal with physical situation such as trajectories or oscillators where there are local minimums and maximums between recorded data points. Where this is not acceptable is when the derivative is not continuous, such as the delta function or the step function. One scenarios this applies to is a sudden switch on/off in a circuit.

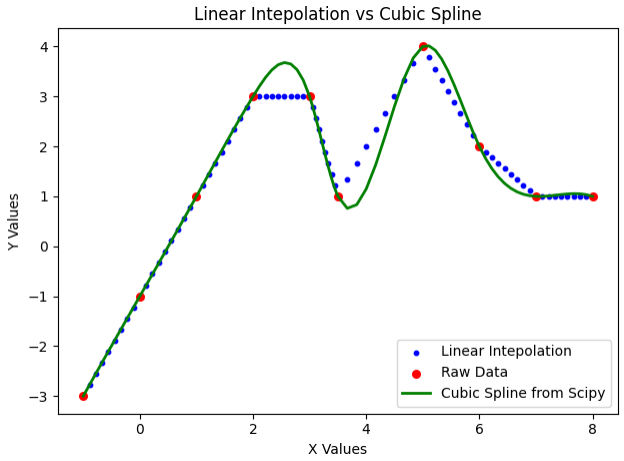

Figure 3. Linear Interpolation (by hand) vs Cubic Spline (from Scipy) as a continuous line

3.4. *Part d:*

Linear interpolation is faster to process and extended to more data points, but there could be a larger error bar if there is a clear trend/fit/physical nature to the scenario. Cubic Spline is slow, but create better interpolation to the data set, creating smoother transition at data points. There could be over fitting in cubic spline due to the cubic nature of the algorithm.

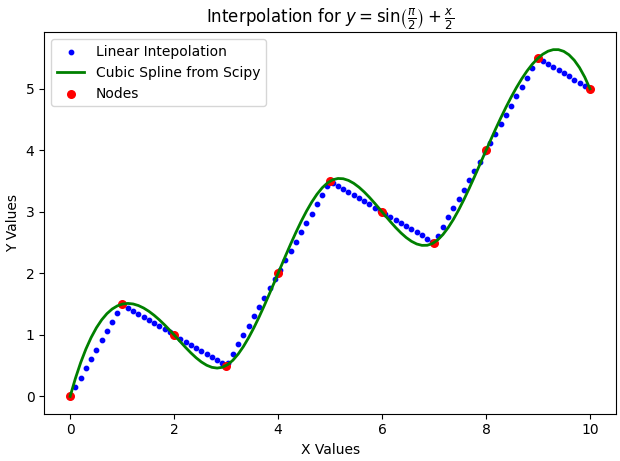

Figure 4. Linear Interpolation (by hand) vs Cubic Spline (from Scipy) for the function $y = \sin(\frac{\pi}{2}x) + \frac{x}{2}$

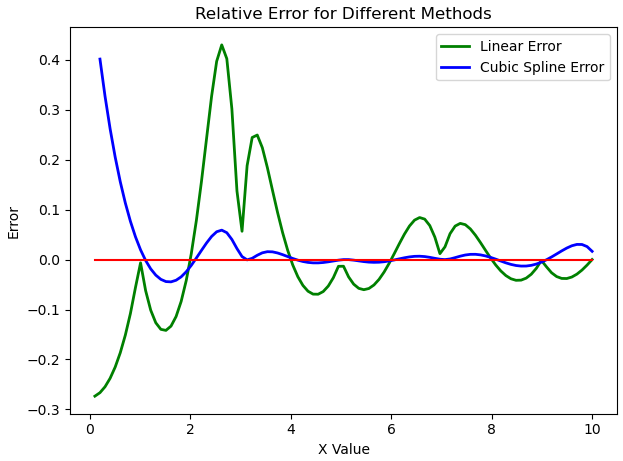

Figure 5. Relative error for our interpolation function, red line is at value y = 0.

4. QUESTION 3

4.1. *Part a:*

Figure 4 is my interpolation, both linear and cubic spline for the function:

$$y = \sin(\frac{\pi}{2}x) + \frac{x}{2}$$

4.2. *Part b:*

Figure 5 is the relative error between the interpolation functions to the actual function. The relative error is high at the beginning due to the small y values (approaching zero), and we see a consistent relatively low error from the Cubic spline method, but there are spikes from the linear method around x values of 2-4, 5, and around 7, where are places with second derivative changes in the original function (change of curve direction). So, for this function, the cubic spline method is more accurate.

5. QUESTION 4

In the case of the simple harmonic oscillator, the linear interpolation will not preserve conservation of mechanical energy, as for points between measured data points, the change in position will be linear, which lead to a linear change in potential energy. However, to preserved potential and kinetic energy, the change in energy will be a second order equation of time. Consider a simple spring, the equation will be

$$x(t) = A\cos(\omega t + \phi)$$

Then, the potential energy will be in the form:

$$U(t) = \frac{1}{2}k\,x(t)^2 = \frac{1}{2}kA^2\cos^2(\omega t + \phi)$$

Which will not be linear, so, the linear interpolation will not capture the conservation of mechanical energy well.

ACKNOWLEDGMENTS

Thanks to Professor Sean Matt for the template.