LAB # 08

03/04/2025

---

1 Introduction

This lab revolves around splines which are piecewise approximations. The focus today will be on building and developing our code for creating linear and cubic splines.

2 Pre-Lab

Before lab, we are to write a subroutine that constructs and evaluates a line that goes through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ at a point α. The code produced used the following formula.

$$y\_eval = f_0 + \frac{(f(x_1) - f(x_0))}{x_1 - x_0}(\alpha - x_0)$$

The code that was produced can be found in the **Lab_08.py** file in the GitHub repository.

3 Building Splines

For this portion of the lab, we are focused on using the code produced in the Pre-Lab with the sample code provided in the following sections to build our codes for *linear* and *cubic* splines.

3.1 Constructing piecewise linear approximaitions

Using a combination of the code produced for the Pre-Lab, the code provided in the lab assignment, and the code provided on Canvas, I was able to complete the functions *line_eval()* and *eval_lin_spline()* functions where the former is the Pre-Lab helper function which is used in the latter. As said before, the code that was produced can be found in the **Lab_08.py** file in the GitHub repository.

3.2 Exercise 01

For this exercise, we are asked to consider the following function.

$$f(x) = \frac{1}{1 + (10x)^2}, \text{ over the interval } [-1, 1]$$

This function is the same that was investigated in Lab_07. Using Desmos, the following plot was produced as reference.

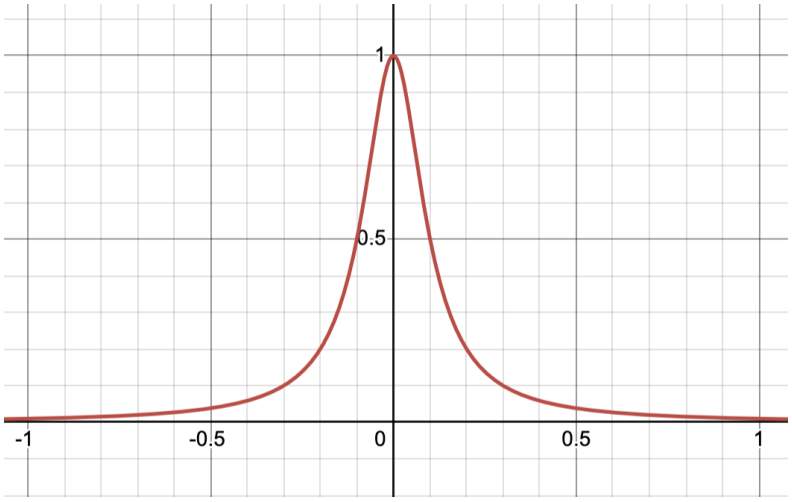

Figure 1: Plot of $f(x)$ over the interval $[-1, 1]$ using Desmos for reference.

Using the plot above as reference, I was then able to plot the output of the linear spline code with assurance that it was working properly. The following is a plot of the output of the *eval_lin_spline()* function that was developed earlier.

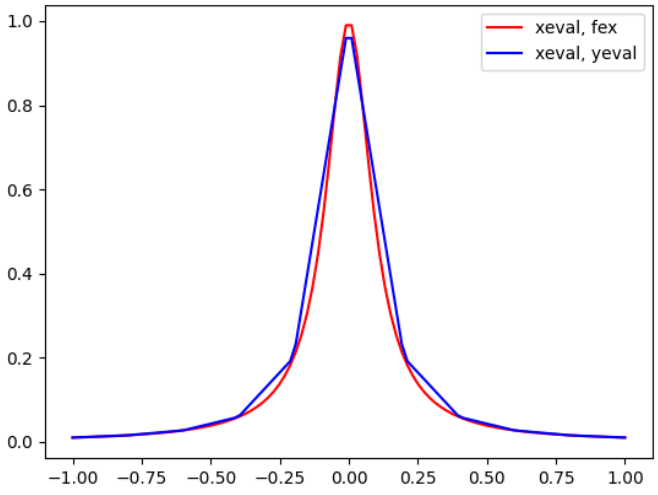

Figure 2: Plot of Linear Spline approximation.

As you can see, the output closely matches that of the plot produced using Desmos. That being said, we can also investigate the error more closely as shown in the plot below. As you can see

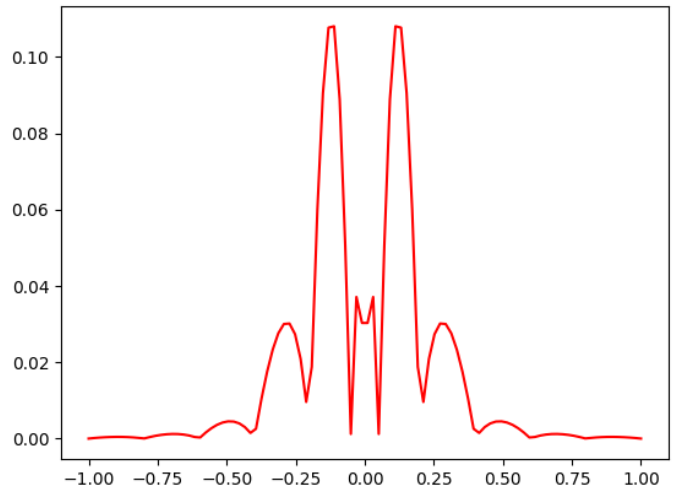

Figure 3: Error of Linear Spline approximation.

above, the maximum absolute error is roughly 10^{-11}. When compared to last week's results for the same function, we see that the linear interpolation performs far less accurately compared to the Monomial Expansion, Lagrange Polynomial, and Newton-Dividend Differences methods with lower number of nodes which all had maximum absolute errors that were less that 10^{-6}. The linear spline did, however, perform better than last week's method when larger numbers of nodes were used.

3.3 Constructing cubic splines

For this portion of the lab, we are focused on using the code produced in the Pre-Lab with the sample code provided in the following sections to build our code for *cubic* splines. Similar to before, a combination of the Pre-Lab code, the code in the lab assignment, and the code provided on Canvas was used. It is important to note that while progress was made for this section, I was not able to get a working set of helper functions for this section. The code that was produced can be found on the GitHub Repository.

4 Deliverables

All of the python code, latex code, and latex rendering can be found in the GitHub repository in the **Lab_08** folder.