LAB # 07

02/25/25

---

1 Introduction

This lab is centered around exploring and building an understanding of simple interpolation techniques. From class, we saw that the polynomial going through the points $(x_j, f(x_j))_{j=0}^n$ where $f(x) \in C^{(n+1)}[a,b]$ and $x_j \in [a,b]$ is unique. As said in class, there are many ways of approximating and constructing this polynomial which we will see in this lab. In doing so, we will identify the more stable method as well as discuss the error that arises in each method. Later, we will explore the interpolation error which is given by the polynomial $\Psi(x) = (x - x_0)\dots(x - x_n)$. We will be exploring this error and investigating node placements and how that effects the error.

2 Pre-Lab

As shown in class, the most naive way of constructing an interpolation polynomial $f(x) \in C^{(n+1)}[a,b]$ with data $(x_j, f(x_j))_{j=0}^n$ where $x_j \in [a,b]$ is to write the polynomial in terms of monomials and solve for the coefficients. That is we would write a polynomial of the following form in 1 and solve for each coefficient so that it matches our provided data.

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{1}$$

To solve for these coefficients so that the corresponding polynomial passes through our given points $(x_j, f(x_j))$, we can solve the following equation in 2.

$$\mathbb{V}\tilde{\mathbf{a}} = \tilde{\mathbf{b}} \tag{2}$$

In 2, $\mathbb{V}$ is the Vandermonde matrix which can be found below. The vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{b}}$ are vectors of the coefficients and the monomials respectively as shown below.

$$\mathbb{V} = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}, \tilde{\mathbf{a}} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \text{ and } \tilde{\mathbf{b}} = \big[fixthenotationhere\big]$$

In the following section, we will first develop codes to produce polynomial interpolations using *Monomial Expansion*, *Lagrange Polynomials*, and *Newton-Divided Differences*. After constructing these codes, we will explore their errors. Next, we will build methods that improve these interpolation approximations and we will again explore their errors the same way.

3 Exploring Interpolation

Between class and lab, we have now learned the following techniques for constructing and evaluating interpolation polynomials.

1. Monomial Expansion
2. Lagrange Polynomials

3. Newton-Dividend Differences

For the remaining sections, we will focus on implementing these techniques to interpolate the function found below in 3 on the interval $[-1, 1]$.

$$f(x) = \frac{1}{1 + (10x)^2} \tag{3}$$

The interpolation nodes that will be used are given by $x_j = -1(j-1)h$ where $h = \frac{2}{N-1}$ for $j = 0, \ldots, N$.

3.1 Exercises: Different evaluation techniques

3.1.1

For this part, we were tasked with developing codes that can be used for constructing and evaluating the polynomials at the 1000 different points on the interval $[-1, 1]$ using the three methods outlined above. The codes produced were built using the *lab_7.py* as a framework. The code produced can be found in the GitHub repository under **Lab_07** directory.

3.1.2

For this section, we were to use a range of nodes. In each of the plots below, there are approximations and their absolute errors when using $N = 2, 3, \ldots, 10$ nodes. The first plot uses *Monomial Expansion*, the second using *Lagrange Polynomials*, and the third using **Newton-Dividend Differences**.

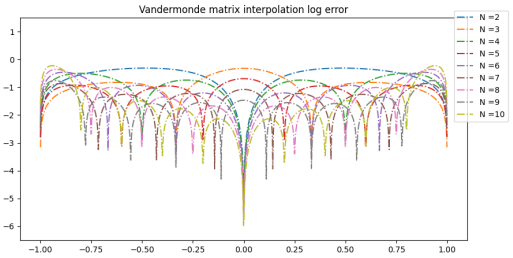

Figure 1: Log of the absolute error of interpolation approximation using Monomial Expansion.

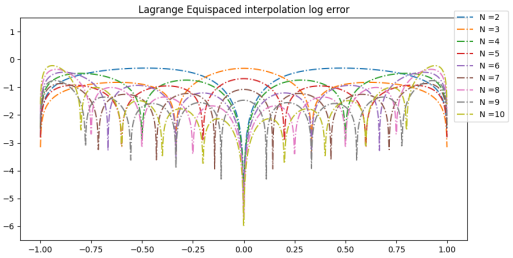

Figure 2: Log of the absolute error of interpolation approximation using Lagrange Polynomials.

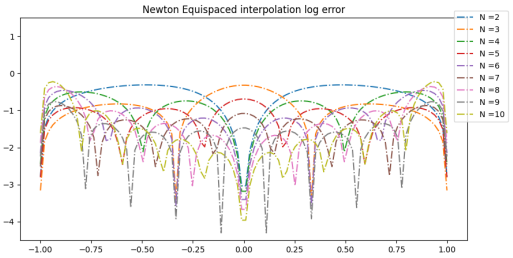

Figure 3: Log of the absolute error of interpolation approximation using Newton-Dividend Differences.

As shown in figures 4, 5, and 6 the methods behave quite similarly for lower number of nodes. The interpolations using Monomial Expansion and Lagrange Polynomials are almost identical with a maximum absolute error around $\log_{10}(10^{-6})$ while the Newton-Dividend Differences method was slightly better with a maximum absolute error around $\log_{10}(10^{-4.5})$.

3.1.3

The following plots below include approximations and their absolute errors when using $N = 11, 12, \ldots, 19$ nodes. The first plot uses *Monomial Expansion*, the second using *Lagrange Polynomials*, and the third using **Newton-Dividend Differences**. As shown in figures 4, 5, and 6 the methods behave

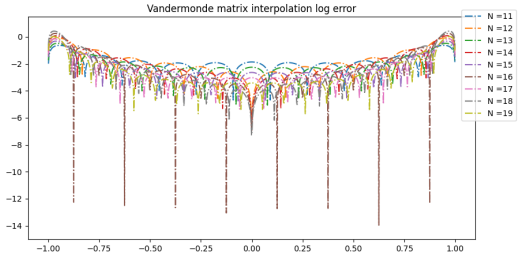

Figure 4: Log of the absolute error of interpolation approximation using Monomial Expansion.

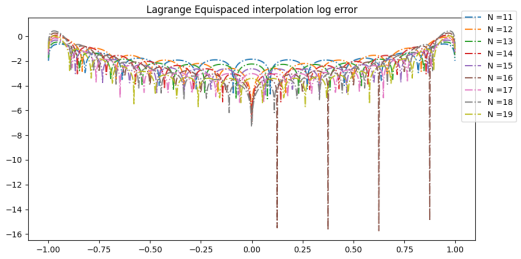

Figure 5: Log of the absolute error of interpolation approximation using Lagrange Polynomials.

quite differently for higher number of nodes. The interpolations using Monomial Expansion and Lagrange Polynomials are similar in their maximum absolute errors which are around $\log_{10}(10^{-12})$, and $\log_{10}(10^{-14})$ respectively while the Newton-Dividend Differences method was much better with a maximum absolute error still around $\log_{10}(10^{-4.5})$.

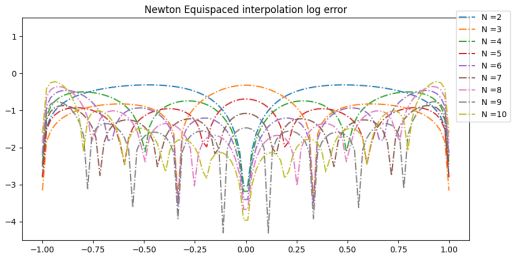

Figure 6: Log of the absolute error of interpolation approximation using Newton-Dividend Differences.

3.2 Improving the approximation

Due to time constraints and errors in lab, I was not able to reach this section.

4 Deliverables

All code, both *.py* and *.tex*, produced for this lab have been pushed to the GitHub repository under **Lab 07**. Additionally, this PDF rendering has been submitted through Canvas.