FCM II Programming Assignment 1

Jonathan Engle

February 6th, 2024

1 Executive Summary

In this report we explore the computational powers and accuracy of polynomial interpolation. The three types of interpolation that we consider through this project are: Monomial basis, Lagrange basis, and Newtons basis. On these interpolation polynomials we need to consider two different types of meshes, the first one we will consider are the intuitive even/ uniform meshes on the selected interval $[a, b]$. We will then discuss the draw backs to these types of methods as depicted in Runges counter example ($f_2(x)$). To move around these types of obstacles we will also consider a different partition of our $[a, b]$ called the Chebyshev meshes. Graphs and other visual aids are provided to further emphasize the error analysis as well as draw backs and advantages.

2 Statement of the Problem

To implement polynomial interpolation techniques, we first need a set of test data or test functions to interpolate over. The two major functions we consider throughout this project are:

$$f_1(x) = (x-2)^9$$

Or

$$f_1(x) = x^9 - 18x^8 + 144x^7 - 672^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$$

$$f_2(x) = \frac{1}{1+x^2}$$

The motivation for selecting these functions is to analyze the accuracy of these methods as well as two different choices of meshes. With the known function, we are able to verify the theoretical results discussed in lecture as well as discuss the efficacy of each method with respect to a higher degree polynomial $f_1(x)$, and a version of Runge's counter example $f_2(x)$. These two functions will aid in the intuition behind why mathematicians such as Chebyshev were inspired to create different meshes on the interpolated interval $[a, b]$

A brief discussion of our meshes (reasoning to follow in section 3). Two meshes that we consider for these interpolation issues are a standard uniform mesh which is generalized as:

$$x_i = a + \frac{b-a}{n} i, 0 \leq i \leq n$$

When first learning about interpolation the uniform mesh is the most natural mesh that first comes to mind. But this then brings the question of: What happens if our function does not behave uniformly everywhere? How do we combat any issues on the outsides of our range? These are questions that Chebyshev answers in the Chebyshev mesh:

$$x_i = \cos\left(\frac{(2i+1)\pi}{2n+2}\right), 0 \leq i \leq n$$

The motivation of this mesh is to concentrate our interpolated $x_i's$ on the end of our interval, which as seen in practice contains many advantages.

Leja ordering: Given a sequence $x = \{x_i\}_{i=1}^{\infty}, x \in X$ in our case is the interpolated x values and X is the entire interpolated interval, we can re-order the sequence with respect to the magnitude of each of the points i.e:

$$\text{For } i = 1\text{: } x_0 = \max_{x \in X} |x|$$
$$\text{For } i = 2, 3, \ldots, n\text{: } x_k = \Pi_{k=1}^{j} |x_j - x_k|$$
$$\text{Or}$$
$$\text{For } i = 2, 3, \ldots, n\text{: } x_k = \max_{x \in X} \Pi_{k=1}^{j} |x - x_k|$$

Note that the generalized form starts at $i = 0$, for our purposes of implementing this Leja ordering method into Matlab we choose to start at $i = 1$ due to the indexing of the Programming language. The motivation for incorporating this Leja ordering method is to increase computational speed as well as distribution of the points. Furthermore to increase accuracy, Leja ordering moves to attack the issue of the weights ($\omega(x)$) as seen in Theorem 8.2. As seen by Bos et al (2005) the Leja ordering allows us to improve interpolation accuracy for different more complex functions such as simple differential equations. For this project we consider more common examples which allow us to gain intuition with regards to interpolation techniques. Furthermore, the motivation for choosing different ordering schemes is to deal with this particular issue of where to insert nodes with respect to a given function and to choose a correct $\omega(x)$ weight.

3 Description of the Algorithms and Implementation

This section moves to describe the theoretical aspects of algorithms and implementation that have been completed in this assignment.

3.1 Monomial Basis

The first interpolation basis we consider is the Monomial Basis. This Monomial basis is the most natural guess of interpolating polynomials when first dealt with an interpolation problem. The monomial basis is defined as:

$$p_n(x) = \sum_{k=0}^{n} a_k \phi_k(x)$$

Where $\phi_k(x) = x^k$ i.e $\phi_0(x) = 1, \phi_1(x) = x, \ldots \phi_k(x) = x^k$. With this structure of our basis now generates the question of how to solve for the weights ($a_k's$). To do this we rely on our knowledge from FCM I, and set up a system with $n + 1$ unknowns and $n + 1$ equations as follows:

$$\begin{cases} P_n(x) = a_0 + a_1 x + \cdots + a_n (x)^n \\ P_n(x_i) = y_i = a_0 + a_1 x_i + \cdots + a_n (x_i)^n \end{cases}$$
$$\begin{bmatrix} 1 & x_0^1 & x_0^2 & \ldots & x_0^n \\ 1 & x_1^1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}$$

This all too familiar form of $A\vec{a} = y$ where we wish to solve for a brings us to our techniques of computing A^{-1} from FCM 1. To do this we perform LU decomposition to solve for this ill conditioned Vandermonde matrix A. We also use complete pivoting to avoid any issues and make the code more accessible for the user. As we can see the Monomial basis is a trivial and natural choice for an interpolation basis but contains its drawbacks as it is difficult and computationally costly for solving $A\vec{a} = y$.

3.2 Lagrange Basis

The next basis we consider for the interpolation techniques is the Lagrange Basis. This basis takes the form of:

$$P_n(x) = \sum_{k=0}^{n} a_k \phi_k(x) = \sum_{k=0}^{n} a_k \left(\Pi_{j=0, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)} \right)$$

Note that in our implementation we move the index over by one due to the indexing of Matlab. A key observation and motivation for using this method is that If $k \neq i, \phi_k(x_i) = \sum_{k=0}^{n} a_k \left(\Pi_{j=0, j\neq k}^{n} \frac{(x_i - x_j)}{(x_k - x_j)} \right) = 0$ since $(x_j - x_j) = 0$. Thus our A matrix becomes:

$$A = \begin{bmatrix} \phi_0(x_0) & \dots & \phi_n(x_n) \\ \phi_0(x_1) & \dots & \phi_n(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_n) & \dots & \phi_n(x_n) \end{bmatrix} = I_{n \times n}$$

This observation makes computing our $\vec{a}$ extremely easy, bypassing the computation of A^{-1} yielding $a_k = y_k \forall k = 0 \dots n$. This is the main advantage to using the Lagrange basis for interpolation. However, computing $\phi_k(x)$ is complicated as well as computing further extensions of the function such as $Pn'(x)$ and $\int P_n(x)$.

3.3 Newton Basis

Finally, with these two basis in mind, we consider the Newton basis as our third testing basis. Newtons basis functions up to a degree n are defined as $\phi_k(x) = \Pi_{j=0}^{k-1}(x - x_j), k = 0, \dots, n$. As both of the previous methods we obtain a linear system. The linear system now is formed as:

$$A = \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \dots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \dots & \phi_n(x_n) \end{bmatrix}$$

Where everything above the diagonal entries of Matrix A are zeros leaving us with a lower triangular matrix. Where we then can utilize forward substitution to lessen the computational cost. The Newtons basis provides us with the best of both of the previous methods in that it is easier to work with extensions of $P_n(x)$ such as $P_n'(x)$ and $\int P_n(x)$. While keeping our matrix A relatively simple in that it is easy to compute A^{-1}. For the implementation we use the forward substitution method from our previous FCM I assignment. As you can see the Lagrange Basis is difficult and computationally costly to perform, to bypass this issue we perform Barycentric method for ease of computations and computational complexity.

4 Description of the Experimental Design and Results

This section moves to describe the experimental design, testing, and results of the following interpolation techniques as discussed above. We will break up this section with respect to the functions $f_1(x) = (x-2)^9$ and the famous Runges counter example of $f_2(x) = \frac{1}{1+x^2}$. We begin first by discussing that our errors can be bounded for any choice of n by the following theorem.

- **Theorem 8.2**: Let $x_0, x_1, \dots, x_n$ be $n+1$ distinct nodes and let x be a point belonging to the domain of a given function f. Assume that $f \in C^{n+1}(I_x)$, where I_x is the smallest interval containing the nodes $x_0, x_1, \dots, x_n$ and x. Then the interpolation error at the point x is given by

  $$E_n(x) = f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$$

  where $\xi \in I_x$ and ω_{n+1} is the nodal polynomial of degree $n+1$.

All of our errors for the following computations were bounded by this theorem for small(5), medium (51) and large (151) n. Figures are shown for small cases to emphasize details. We compute the error by taking $\|f(x) - P_n(x)\|_\infty$, note $\|f(x_i) - P_n(x_i)\|_\infty = 0$ since we are using those interpolant values of the given function.

4.1 $f_1(x) = (x-2)^9$

For the following function $f_1(x) = (x-2)^9$ we can first note that it is a high order odd polynomial with one distinct root. To interpolate this polynomial we first consider the Monomial Basis with equally meshed partitions on the interval $[4, 4]$ the Monomial basis constructs a good approximation for this function and approaches the correct coefficients as n gets larger. Errors are bound and also decrease as n gets large. Next, we consider the Lagrange and Newtons basis with similar results with better initial accuracy. Note that we do obtain issues at the end points of each of our interpolating polynomials with respect to our error. To combat this issue we can also test out the Chebyshev mesh which emphasizes our interpolating points towards the end of the range. This incorporation of the Chebyshev mesh decreases our error and creates a better fitting line for all of the methods. This particular function is nice in that there are no major issues that arise due to the fact that it is a polynomial.

4.2 $f_2(x) = \frac{1}{1+x^2}$

Now we dive into a deeper more interesting case called Runges counter example. This function looks somewhat of a bell shape (figures bellow) and is fascinating as it creates issues for our standard mesh size for interpolation. Immediately after running all of the alternative basis' we can see that issues arrise at the end points and we obtain large oscillations even as $n \to \infty$. This phenominon is caused by the $f^{(n+1)}(\xi)$ growing quickly to infinity as $n \to \infty$. More formally:

$$\max_{-1 \le \xi \le 1} f^{(n+1)}(\xi) \le (n+1)! 5^{n+1}$$

Furthermore, the Lebesque constant increases quickly for this function as well for this case. This leads us to the main motivation for selecting the Chebyshev polynomials as we can influence the error significantly by adjusting our weights to the end of our intervals.

5 Conclusion and Comparison of Methods

Overall this project sought to compare the three different Interpolation methods called Monomial basis, Lagrange Basis and Newtons basis. As seen in our discussion all of these methods contain their own respective strengths and weaknesses. When observing error it is crucial to check that our methods are accurate and align with the theoretical results of Theorem 8.2. As we have found issues can arise as found in the generalized Runges counter example. Interpolation techniques are fascinating in that it is how we discretize and visualize continuous functions with approximations. Understanding the motivations and draw backs to multiple methods further expands our intuition behind numerical methods to solve for real world problems.

6 Theorems, Tables, and Figures

6.1 Theorems, Properties, Citations

1. **Theorem 8.1**: Given $n+1$ distinct nodes $x_0, \ldots, x_n$ and $n+1$ corresponding values $y_0, \ldots, y_n$, there exists a unique polynomial $\Pi_n \in \mathbb{P}_n$ such that $\Pi_n(x_i) = y_i$ for $i = 0, \ldots, n$.

2. **Theorem 8.2**: Let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct nodes and let x be a point belonging to the domain of a given function f. Assume that $f \in C^{n+1}(I_x)$, where I_x is the smallest interval containing the nodes $x_0, x_1, \ldots, x_n$ and x. Then the interpolation error at the point x is given by

$$E_n(x) = f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

   where $\xi \in I_x$ and ω_{n+1} is the nodal polynomial of degree $n+1$.

3. Citation: Bos, L P, and M Caliari. Newton Interpolation at Regularised Leja Sequences, www.math.unipd.it/ marcov/pdf/lejac.pdf. Accessed 4 Feb. 2024.
   Link: `https://www.math.unipd.it/~marcov/pdf/lejac.pdf`

6.2 Figures

6.2.1 Monomial Basis

Figure 1: Monomial Basis with Chebychev mesh on $f_1(x), n = 4$

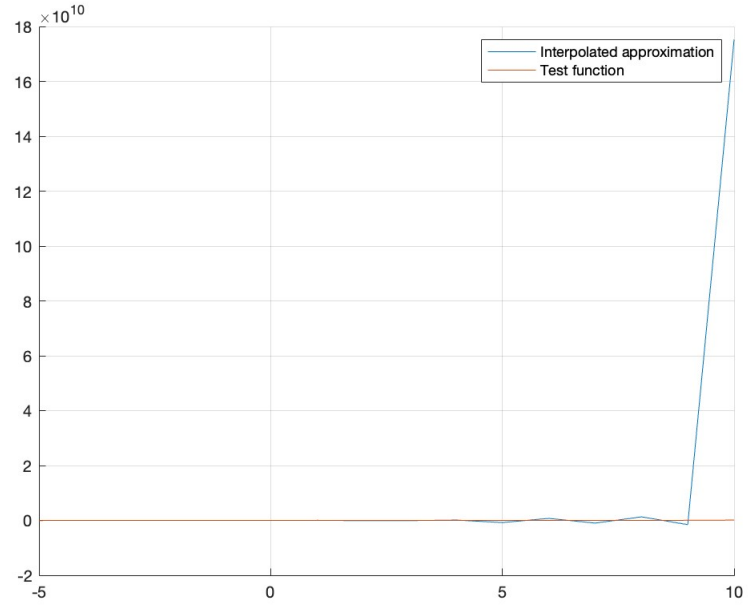


Figure 2: Monomial Basis with Normal mesh on $f_1(x), n = 4$

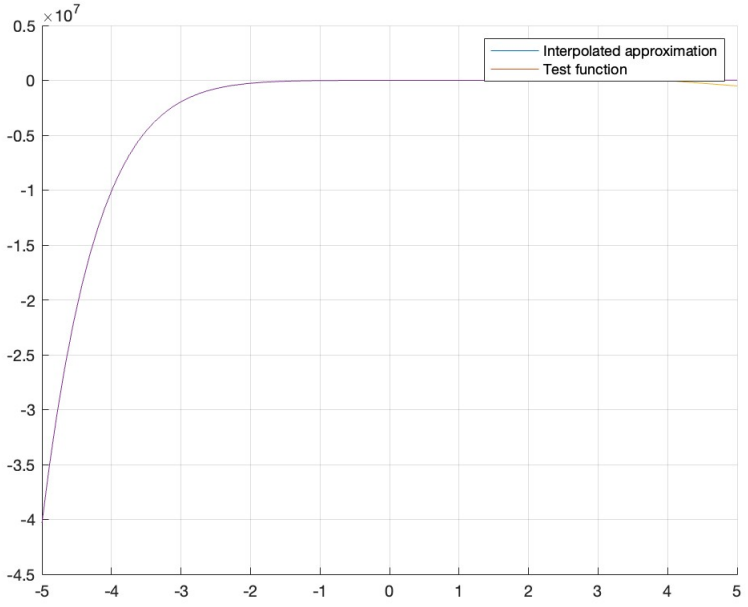


Figure 3: Monomial Basis with Chebychev mesh on $f_2(x), n = 4$

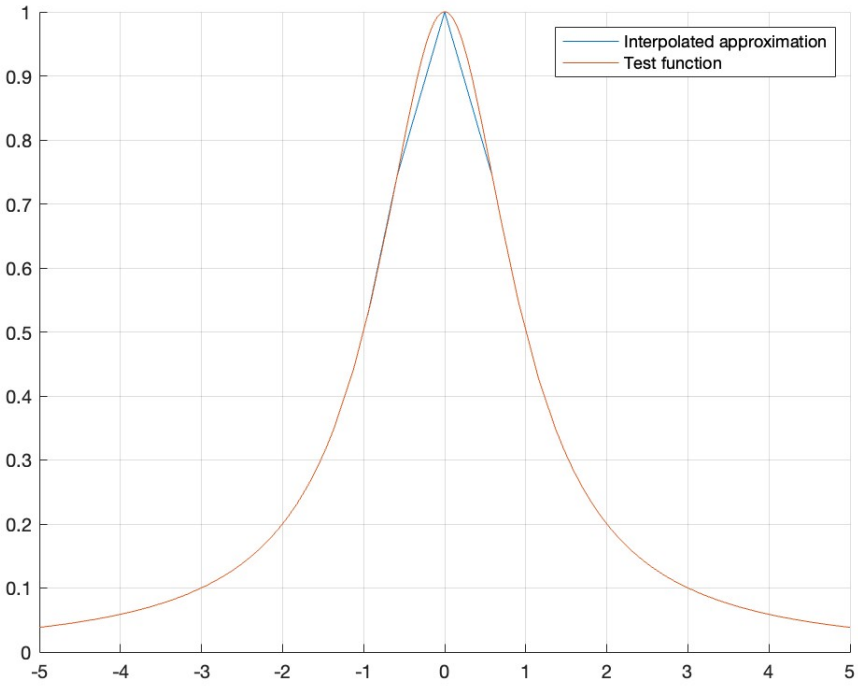
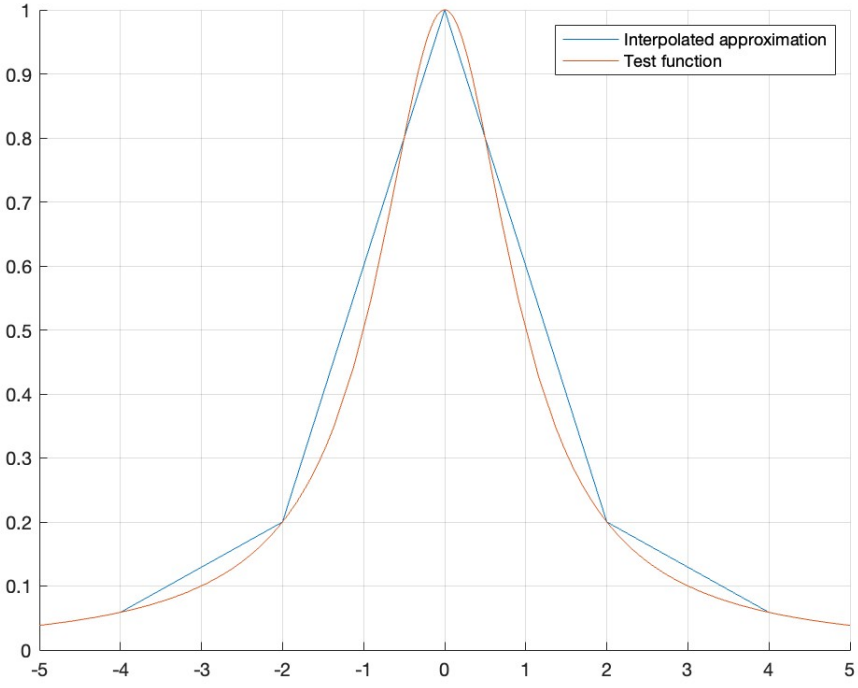


Figure 4: Monomial Basis with Normal mesh on $f_2(x), n = 4$



6.2.2 Lagrange Basis

Figure 1: Lagrange Basis with Chebychev mesh on $f_1(x), n = 4$

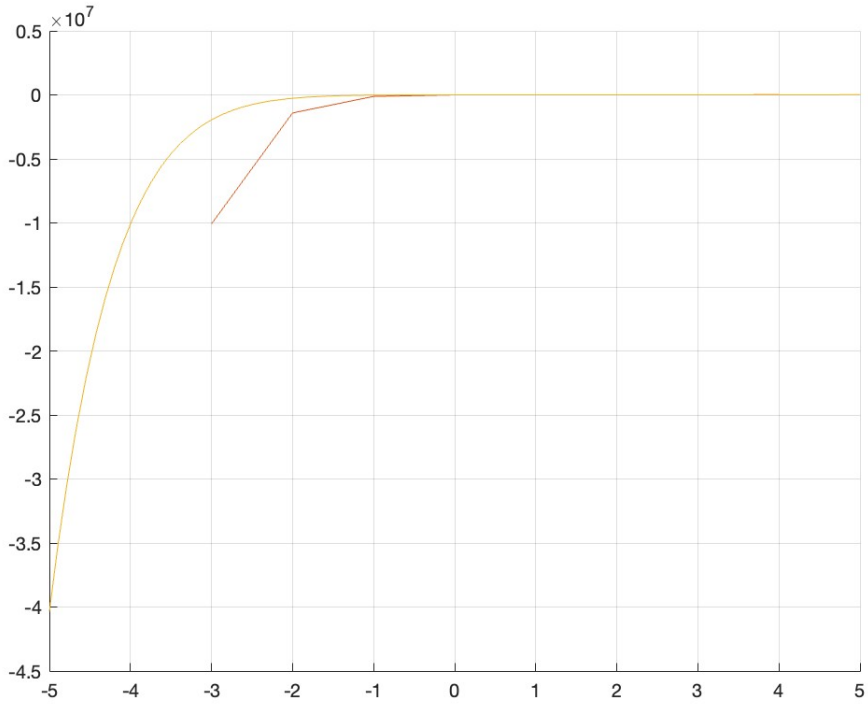


Figure 2: Lagrange Basis with Normal mesh on $f_2(x), n = 4$

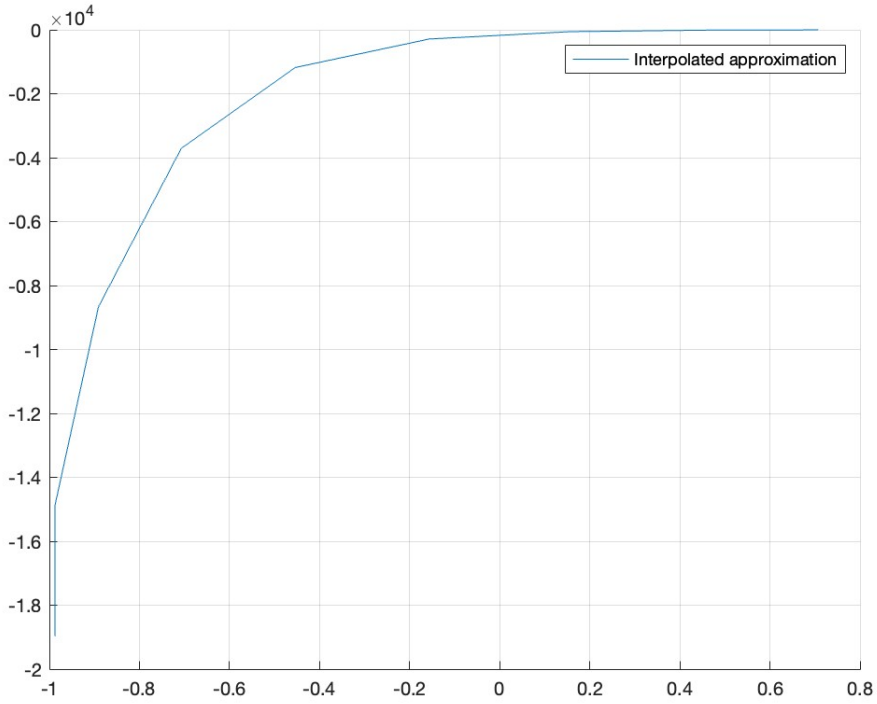


Figure 3: Lagrange Basis with Chebychev mesh on $f_2(x), n = 4$ Note: I believe I had an index issue here causing the shift

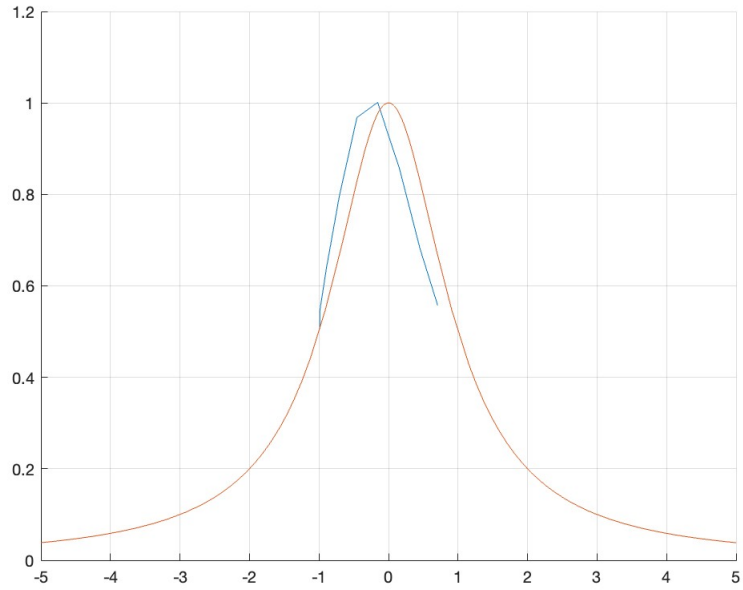
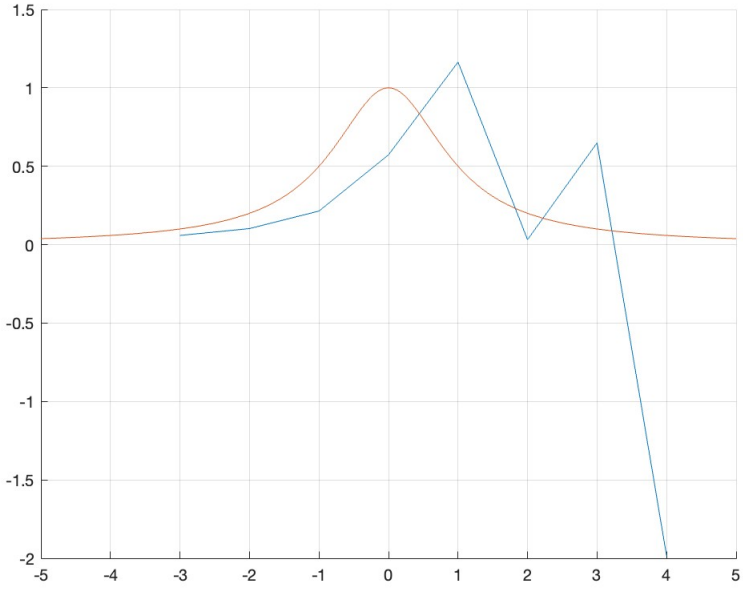


Figure 4: Lagrange Basis with Normal mesh on $f_2(x), n = 4$ Note: I believe I had an index issue here causing the shift



6.2.3 Newton Basis

Figure 1: Newton Basis with Chebychev mesh on $f_1(x), n = 5$

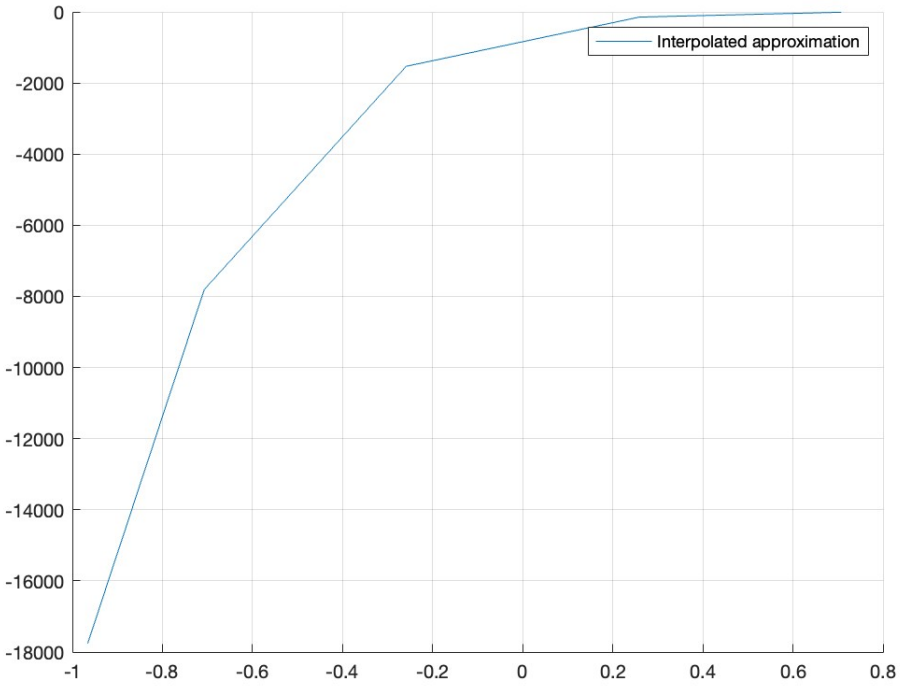


Figure 2: Newton Basis with Normal mesh on $f_1(x), n = 5$

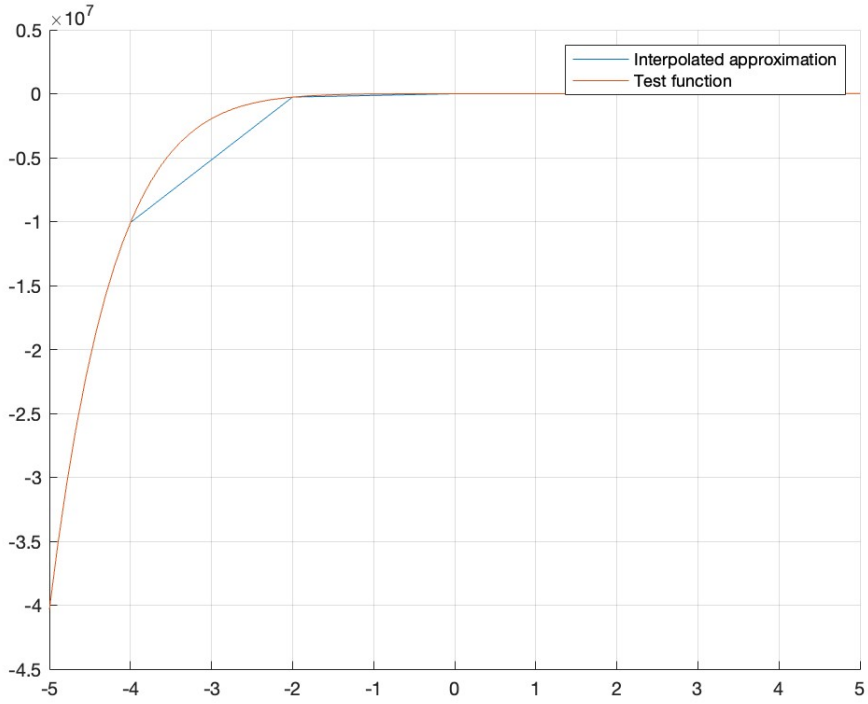


Figure 3: Newton Basis with Chebychev mesh on $f_2(x), n = 4$

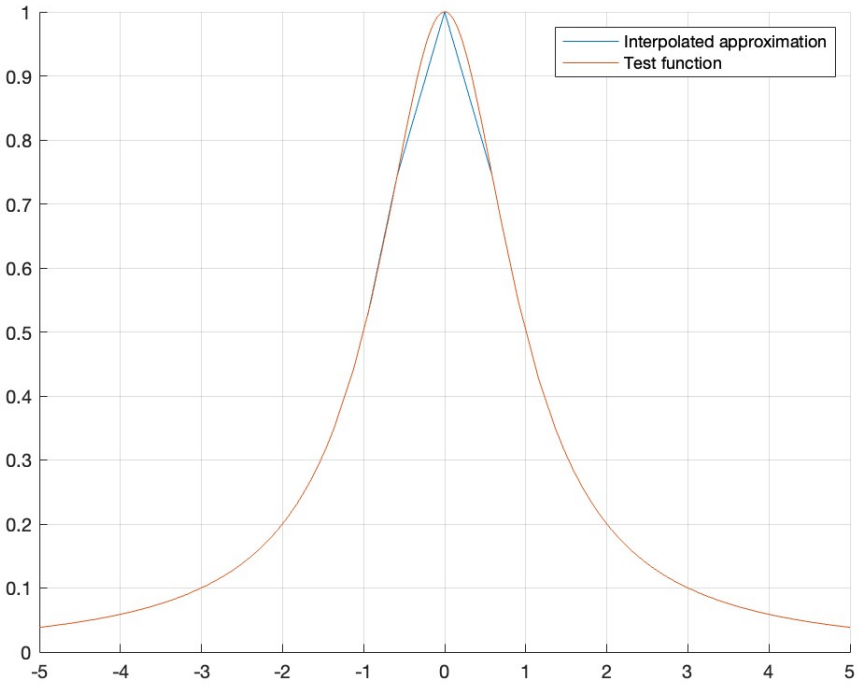


Figure 4: Newton Basis with Normal mesh on $f_2(x), n = 4$