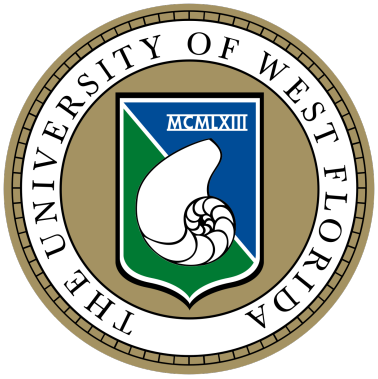
Methods of Polynomial Interpolation

An Analysis of Error and Computational Performance

Nicholas Sebasco

Advisor: Dr. Lei Hsin Kuo



Mathematics
University of West Florida
United States
August 8th 2021

Introduction

This research will compare and contrast methods of polynomial interpolation. The following functions will be approximated by degree n polynomials, and their absolute errors and computational speed will be analyzed.

$$f(x) = Sin(x), x \in [0, 2\pi] \tag{1}$$

$$g(x) = Sin(5x), x \in [0, 2\pi] \tag{2}$$

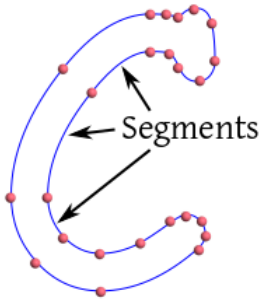
$$h(x) = \frac{1}{1+x^2}, x \in [-5, 5] \tag{3}$$

Each polynomial will be constructed for $n = 5$, 10, 15, 25, and 35 points. After construction, the polynomials will be plotted on $n_t = 1000$ test points. 6 curves for each n value were created for two sets of figures. The first set of figures are plots of the 6 approximations along with the actual function. The second set of figures are plots of the error of each of the 6 curves. The figures regarding the error distributions are defined as: $\varepsilon := \{|f(x_j) - p(x_j)|\}_{j=1}^{m}$, where $x_i = a + [b-a]\frac{i}{n_t}, i = 0, 1, ..., n_t$. The 6 curves are defined as:

1. Direct-Equal: The Direct method constructed on n equally spaced points.
2. Direct-Chebyshev: The Direct method constructed on n Chebyshev nodes.
3. Lagrange-Equal: A Lagrange polynomial constructed on n equally spaced points.
4. Lagrange-Chebyshev: A Lagrange polynomial constructed on n Chebyshev nodes.
5. Hermite-Equal: A Hermite polynomial constructed on n equally spaced points.
6. Hermite-Chebyshev: A Hermite polynomial constructed on n Chebyshev nodes.

One of the oldest and most applied problems in mathematics is the approximation of a function with a simpler function. Polynomials are ideal to approximate with because they have nice properties are relatively well understood [6]. Interpolating polynomials are fascinating because of their potential to solve computer graphics, signal processing, derivative approximation, cryptography, and many other types of real world problems.

Figure 1: Polynomial interpolation has interesting applications in typography. The letter C from the Comic Sans font.



Numerical Methods

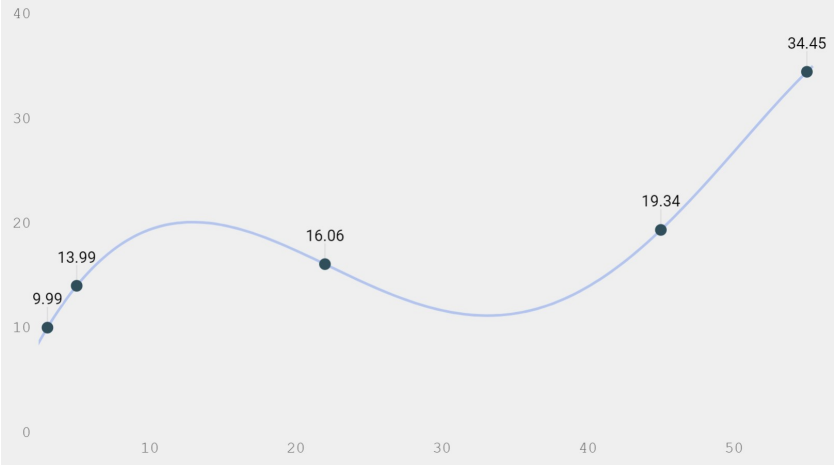


Figure 2: A degree 5 interpolating polynomial passing through a dataset.

Polynomial interpolation involves the construction of a smooth continuous polynomial from a discrete set of data points; it permits the approximation of a function. Given $a = x_0 < x_1 < x_2 < \cdots < x_n = b$, which is a partition of the closed interval $[a, b]$. Polynomial interpolation is the process of finding a nth degree polynomial $p(x)$ that agrees with $f(x)$ at the points of the partition. Polynomial interpolation is a subclass of polynomial approximation in which the interpolating polynomial must pass through all of the data points [1].

$$p\,(x) = f(x_i), \qquad \forall\ i = 0, 1, 2, \cdots, n. \tag{4}$$

The Weierstrass Approximation theorem confirms that, if given a continuous function on a closed interval, $f(x) \in C[a, b]$, then for each $\epsilon > 0$, there exists a polynomial $p(x)$ with the property that:

$$|f(x) - p(x)| < \epsilon, \qquad \forall\ x \in [a, b] \tag{5}$$

Thus, polynomials can be used to approximate functions arbitrarily well. Polynomials have many desirable properties, including: that they are easy to evaluate, differentiate, and integrate.

0.1 The Direct Method

One of the most straightforward approaches to polynomial interpolation involves the use of the Vandermonde matrix, from henceforth the method will be referred to as the Direct method. Consider some nth degree polynomial [1]:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{6}$$

To construct the interpolating polynomial via the Direct method, the coefficients must be determined by solving the following linear system:

$$\begin{aligned} f_0 &= a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n \\ f_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n \\ &\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ f_n &= a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n \end{aligned} \tag{7}$$

Now consider the matrix of monomials, V, the column vector of coefficients, a, and the column vector, f:

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \tag{8}$$

$$a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \quad f = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \tag{9}$$

The matrix V is called the Vandermonde matrix and solving the following system will yield the desired coefficient vector which describes the interpolating polynomial:

$$Va = f \tag{10}$$

There are several methods from linear algebra for solving this system including: Eigenvalue Decomposition, QR Decomposition, Singular Value Decomposition (SVD), Gaussian Elimination, and solving the system with the inverse matrix. Each method employs a novel strategy which may be more or less computationally efficient in terms of solving a given system. Having to solve a system of equations can be done in $O(n^2)$ arithmetic operations best case; however, the general case is $O(n^3)$ operations. These systems can be ill-conditioned, tiny changes in the $x's$ can generate large changes in the solution.

0.2 Lagrange Interpolation

Consider some dataset of size n: $\{x_i, y_i\}_i^n = 0$ where $n \geq 1$, there exists polynomials $L_k \in P(x)$, $k = 0, 1, 2, \cdots, n$. The polynomial [1]:

$$p_n(x) = \sum_{k=0}^{n} L_k(x) \cdot y_k \tag{11}$$

Satisfies the interpolation conditions:

$$p_n(x_i) = y_i, i = 0, 1, 2, \cdots, n. \tag{12}$$

And L_k is defined as:

$$L_k(x) = \prod_{0<i<n;\ i\neq k} \frac{x - x_i}{x_k - x_i} \tag{13}$$

0.3 Hermite Interpolation

Hermite interpolation extends the idea of Lagrange interpolation by requiring the derivative values at the interpolation points. For the datasets: $\{y_i\}_{i=0}^n$ and $\{z_i\}_{i=0}^n$ corresponding to the dataset $\{x_i\}_{i=0}^n$ with $n \geq 0$, we need to find a polynomial $p_{2n+1} \in P_{2n+1}$ [2].

$$p_{2n+1}(x_i) = y_i, p'_{2n+1}(x_i) = z_i, \qquad i = 0, 1, 2, \cdots, n. \tag{14}$$

To satisfy the constraints imposed by (14) two new quantities are defined:

$$H_k(x) = [L_k(x)]^2 \cdot (1 - 2L'_k(x_k)(x - x_k)), \tag{15}$$

$$K_k(x) = [L_k(x)]^2 \cdot (x - x_k) \tag{16}$$

The Hermite polynomial takes its final form with:

$$p_{2n+1}(x) = \sum_{k=0}^{n} [H_k(x) \cdot y_k + K_k(x) \cdot z_k] \tag{17}$$

0.4 Chebyshev Nodes

The Chebyshev nodes are a set of real numbers representing the roots of the Chebyshev polynomials of the first kind. Chebyshev polynomials of the first kind can be obtained from the recurrence relation [3]:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2x \cdot T_n(x) - T_{n-1}(x) \end{aligned} \tag{18}$$

For some natural number i, the nodes in the interval $(-1, 1)$ are:

$$x_j = cos(\frac{2j-1}{2i}\pi), \qquad j = 1, 2, \cdots, n. \tag{19}$$

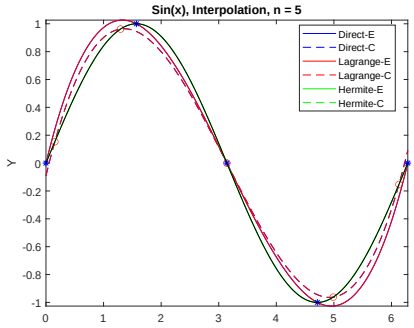
This formula can be generalized to any interval $[a, b]$ via an affine transformation [5]:

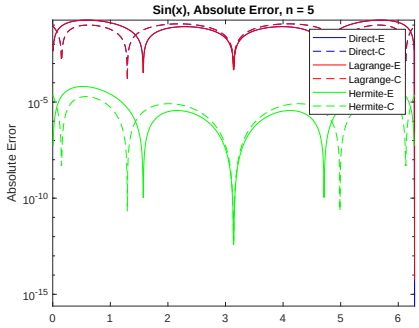
$$x_j = cos(\frac{2j-1}{2i}\pi) \cdot \frac{(b-a)}{2} + \frac{(b+a)}{2}, \qquad j = 1, 2, \cdots, n. \tag{20}$$

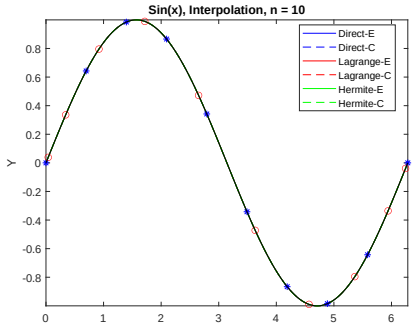
These nodes can be used as opposed to n equally spaced points to create an interpolating polynomial which mitigates Runge's phenomenon. Runge's phenomenon is the tendency for high degree interpolating polynomials to start violently oscillating near the endpoints; consequently, generating a large amount of error.

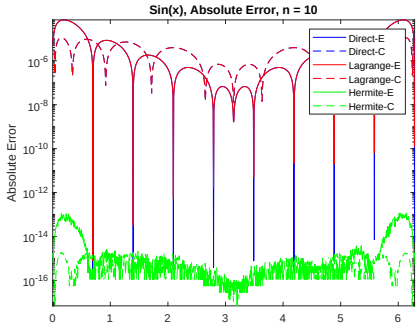
Numerical Experiments

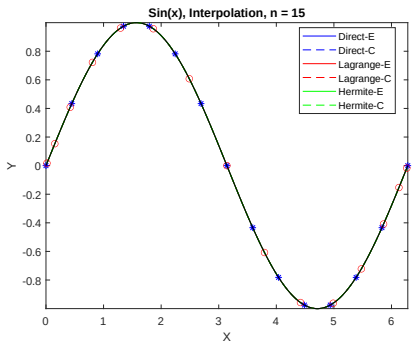
0.5 $f(x) = Sin(x)$

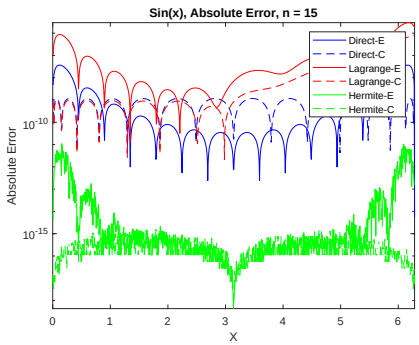












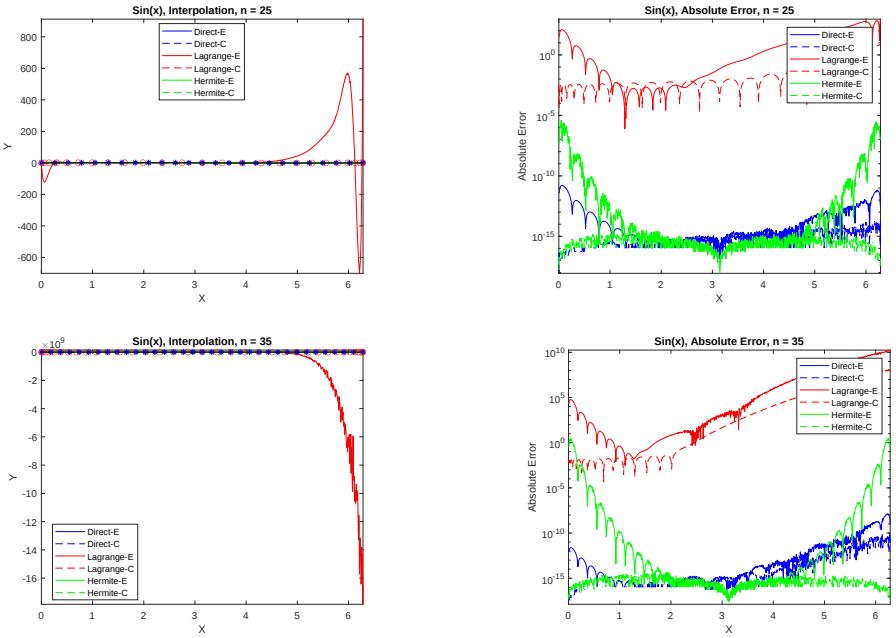
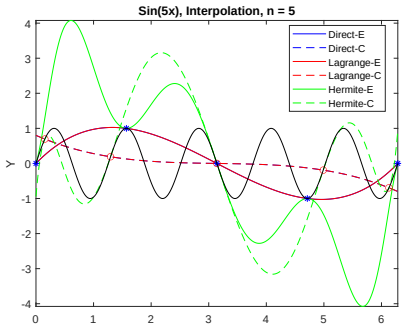
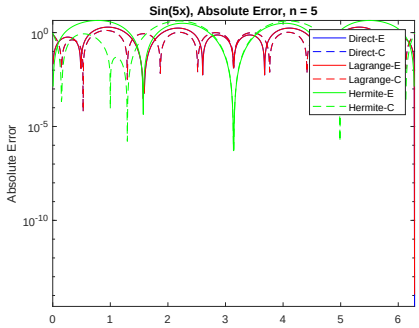
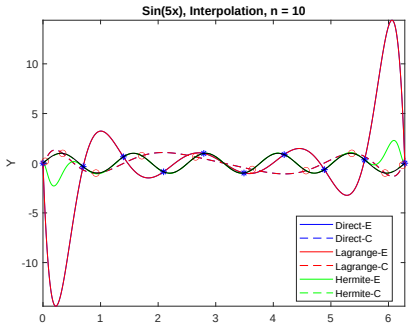


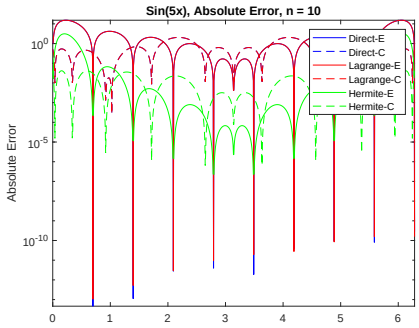
Figure 3: Approximation & error analysis of $Sin(x)$.

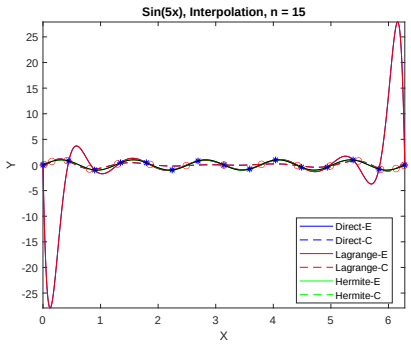
0.6 $f(x) = Sin(5x)$

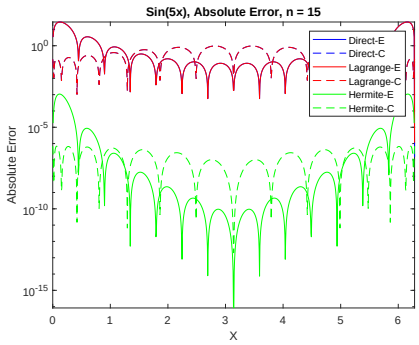

Sin(5x), Interpolation, n = 5
Y
Direct-E
Direct-C
Lagrange-E
Lagrange-C
Hermite-E
Hermite-C


Sin(5x), Absolute Error, n = 5
Absolute Error
Direct-E
Direct-C
Lagrange-E
Lagrange-C
Hermite-E
Hermite-C


Sin(5x), Interpolation, n = 10
Y
Direct-E
Direct-C
Lagrange-E
Lagrange-C
Hermite-E
Hermite-C


Sin(5x), Absolute Error, n = 10
Absolute Error
Direct-E
Direct-C
Lagrange-E
Lagrange-C
Hermite-E
Hermite-C


Sin(5x), Interpolation, n = 15
Y
X
Direct-E
Direct-C
Lagrange-E
Lagrange-C
Hermite-E
Hermite-C


Sin(5x), Absolute Error, n = 15
Absolute Error
X
Direct-E
Direct-C
Lagrange-E
Lagrange-C
Hermite-E
Hermite-C

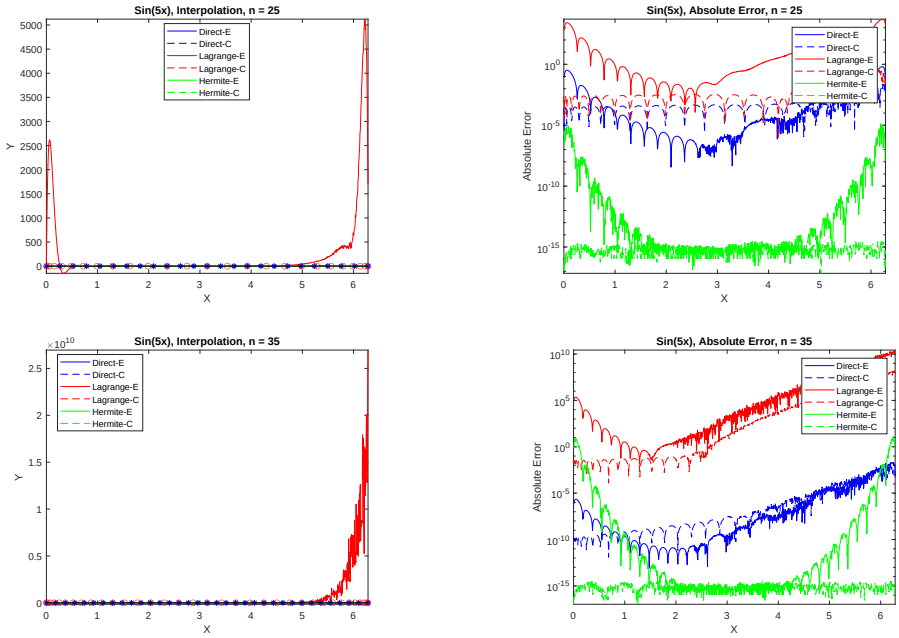
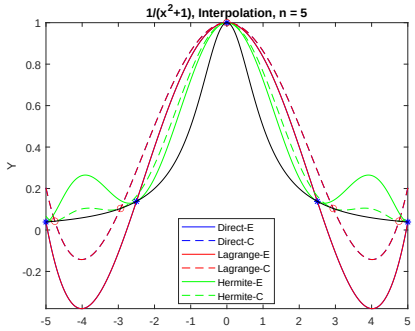
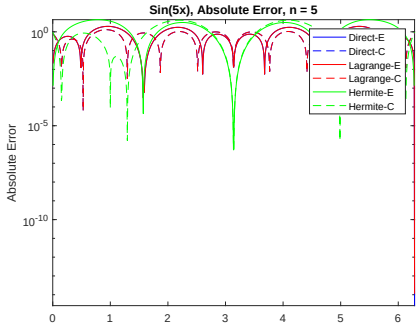
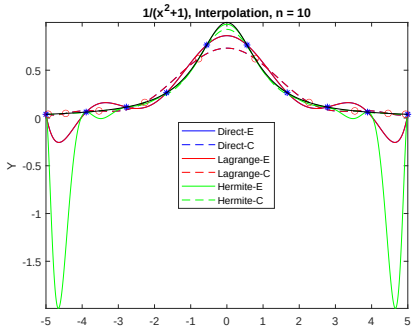


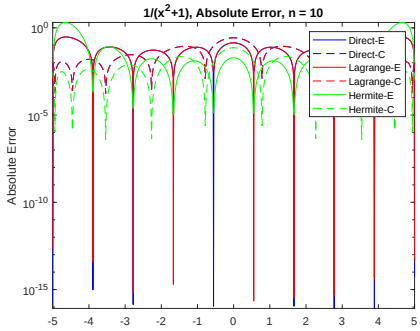
Figure 4: Approximation & error analysis of $Sin(5x)$.

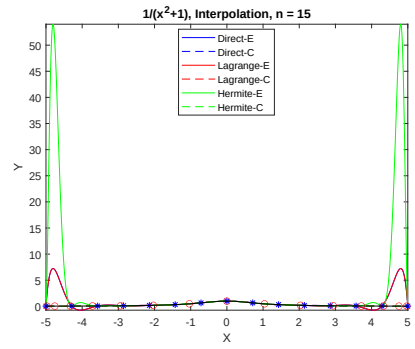
0.7 $f(x) = \dfrac{1}{1+x^2}$

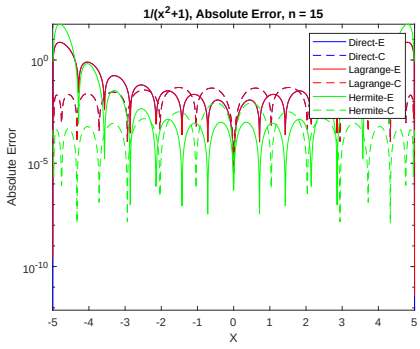












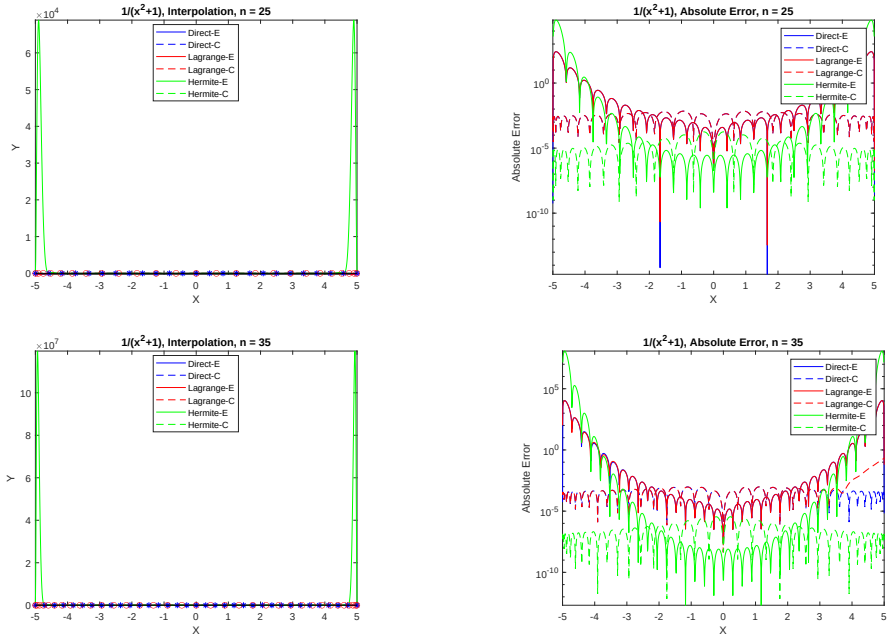


Figure 5: Approximation & error analysis of $\dfrac{1}{1+x^2}$.

0.8 Maximum Error

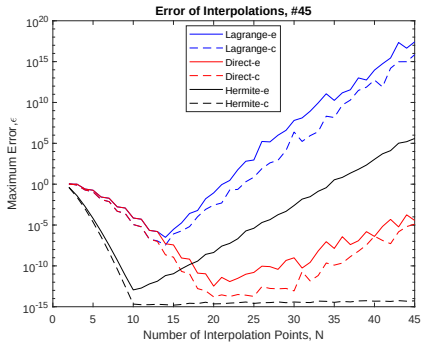


Figure 6: Maximum error of $Sin(x)$.

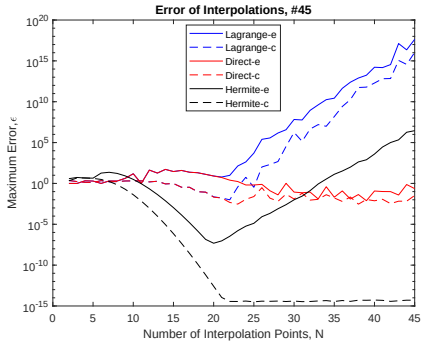


Figure 7: Maximum error of $Sin(5x)$.

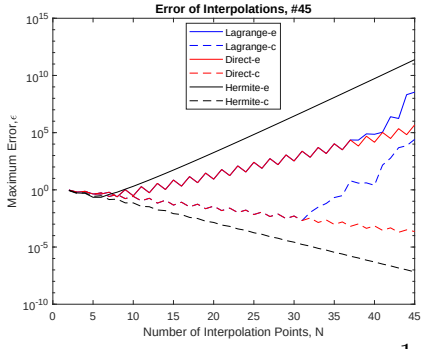


Figure 8: Maximum error of $\dfrac{1}{1+x^2}$.

0.9 CPU Time

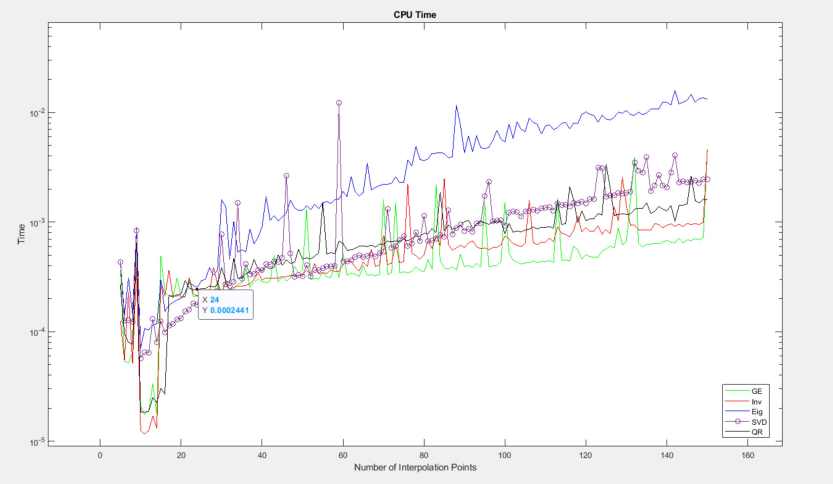


Figure 9: Timing various methods of solving equation (10) for the Direct method.

Discussion

The error in the approximations of $Sin(x)$ for $n = 5$ can be visually seen, the curves deviate from each other by at most 10^{-1}. The Hermite polynomials produce the lowest error with a max absolute error of approximately 10^{-4}. The transition from $n = 10$ to $n = 15$ indicates a dramatic decrease in absolute error. The Lagrange interpolating polynomials show the highest error at this point, followed by the direct method interpolating polynomials, and finally the Hermite polynomials maintain the lowest error. Interestingly the Hermite polynomials appear to have reached the limit in error reduction, 10^{-16}. This phenomena is due to the 16 place decimal precision of floating point numbers in MATLAB. Another interesting phenomena that can be observed is that the error curves generated by the n equally spaced points begin forming a "U" shape, that is, the absolute error is much higher on the end points relative to the center of the curve. The curves generated by the Chebyshev nodes appear to be immune to this phenomenon. From $n = 25$ to $n = 35$ the error behavior for various interpolation methods is quite different. The Lagrange error (for both the Chebyshev nodes and n equally spaced points) blows up and our sinusoidal wave is no longer visually perceptible in the interpolation graphs. The error of the direct method interpolating polynomials reaches an all time low at $n = 25$. At $n = 35$, the error toward the right end point, 2π, starts to creep back up. By extrapolation, for progressively larger n values I predict that this error will continue to get higher. The parabolic "U shape" becomes incredibly prominent for the Hermite polynomial constructed from n equally spaced points. For the Hermite polynomial constructed from Chebyshev nodes the error appears to have remained constant at the lowest value from $n = 15$ to $n = 35$.

$Sin(5x)$ represents a sinusoidal wave with a higher frequency and shorter period relative to $Sin(x)$. At $n = 5$ the approximations are notably worse. Interestingly, the Lagrange polynomial from n equally spaced points depicts the same approximation as it did for $Sin(x)$. This implies that we need more data points or a better heuristic for choosing data points. The Hermite polynomials also depict a lot of error. For the Hermite polynomial from n equally spaced points the curve looks like a sine wave where each of the subsequent peak/ trough pairs is lowered by some linearly decreasing offset. For $n = 5$, none of the approximations are without a substantial amount of visual error and thus it is clear that more data points are needed. From $n = 10$ to $n = 15$ the Lagrange polynomials and direct method polynomials share nearly the same amount of error. This error is still suboptimal, especially at the endpoints, which depict 2 large peaks for the polynomials created from n equally spaced points. The higher degree of the polynomial appears to be amplifying the end point error. The polynomials created from the Chebyshev nodes enjoy a much lower absolute error near the endpoints; however, they express a very small increase in error relative to the polynomials created from n equally spaced points toward the center of the curves. From $n = 25$ to $n = 35$ a lot of interesting stuff happens. The Lagrange Chebyshev error reaches an all time low at $n = 25$, then it blows up out of control at $n = 35$. The error for the Lagrange interpolating polynomial from n equally spaced points is so large at the endpoints that the peak at the endpoints becomes the only discernible feature on the interpolating polynomial graphs. The direct method polynomial error gets substantially lower and reaches an all time low at $n = 35$. The Direct Chebyshev error looks better (lower) at the left end point; however, toward the center of the graph the Direct method polynomial from n equally spaced points produces a lower error. The Hermite polynomial from n equally spaced

points expresses an all time minimum error toward the center of the curve; however, the end point error is starting to get very prominent. The Hermite from Chebyshev nodes continues to be the best approximation with the lowest overall error.

$\frac{1}{1+x^2}$ represents the first non sine function; although, visually it does look similar to $Sin(5x)$ on the interval $[1, 2.15]$. The most striking feature from these curves is how the Hermite polynomials constructed from n equally spaced points absolute error at the end points blows up out of control. By $n = 35$ the error at the endpoints is 10^7 and the spikes at the end points are the only discernible features on the interpolating polynomials graph. The Lagrange and Direct absolute error for n equally spaced points are also bad, and actually worse than the Hermite polynomial towards the center of the curves; however, the error is nowhere near as bad at the end points. The Hermite Chebyshev polynomial produces the lowest overall absolute error again; however, it only reaches about 10^{-10} this time. The absolute error for the Direct Chebyshev and Lagrange Chebyshev polynomials appears to be getting incrementally better and reaches an all time low at $n = 35$ of about 10^{-5}.

Conclusion

The clear winner in terms of minimization of the absolute error was the Hermite polynomial constructed from the Chebyshev nodes. The only cases where the absolute error was not significantly lower than the other interpolating polynomials was for small values of n. Another interesting observation that can be made from this experiment is that although the Chebyshev node polynomials enjoy significantly less end point error, the n equally spaced points polynomials express slightly lower error towards the center of the curves. This begs the question of whether some piecewise combination of the 2 functions should be used in order to enjoy the absolute lowest overall error. In other words if you are near the end points, use the Chebyshev nodes; otherwise, use the n equally spaced point polynomial. The spikes in end point error and the parabolic "U" shape observed in the error graphs is known as Runge's phenomenon, which was discussed in the background section. Runge's phenomenon shows that higher degree polynomials do not always improve the overall accuracy of polynomial interpolation. The difference in the error plots for $n = 5$, $Sin(x)$ and $Sin(5x)$ was interesting. With 5 nodes, no interpolation succeeded in reducing visually perceptible error for Sin(5x); whereas, the Sin(x) interpolation yielded significantly less error. This shows that the interpolation accuracy may be sensitive to small changes to the function being approximated. The plot of the cpu times of the algorithms used to solve the linear system for the Direct method indicates that Eigenvalue Decomposition is approximately 1 - 2 orders of magnitude slower than the the fastest method, Guassian Elimination. The maximum error plots show that for the $Sin(x)$ and $Sin(5x)$ the lagrange polynomials produce the most error. For $\frac{1}{1+x^2}$, the Hermite polynomial constructed from n equally spaced points generates tremendous error and even does worse than the Lagrange polynomials. The Hermite-Chebyshev polynomial consistetly produces the lowest maximum error. Further experimentation could be done with a numeric data type with more precision than MATLAB's *double* data type. Furthermore, I could try different n values, functions to approximate, methods of polynomial interpolation, and building the piecewise polynomials I

described above.

References

[1] Dr. Lei Hsin Kuo , "Lecture 3 - Interpolation & Polynomial Approximation," *The University of West Florida Department of Mathematics and Statistics*, pp. 10-25, June 2021.

[2] Dr. Lei Hsin Kuo , "Lecture 4 - Least Square Approximation & Hermite Interpolation," *The University of West Florida Department of Mathematics and Statistics*, pp. 01-25, July 2021.

[3] Dr. Lei Hsin Kuo , "Lecture 5 - Best Approximation in Infinity Norm & Chebyshev Polynomials," *The University of West Florida Department of Mathematics and Statistics*, pp. 24-39, June 2021.

[4] Berrut, J. and Trefethen, L., 2004. Barycentric Lagrange Interpolation. SIAM Review, 46(3), pp.501-517.

[5] Stewart, Gilbert W. (1996), Afternotes on Numerical Analysis, SIAM, ISBN 978-0-89871-362-6

[6] Epperson, J. (2021). An Introduction to Numerical Methods and Analysis 3e. Wiley.