MATH 3351-1 **Numerical Analysis I** **Fall 2025**

Homework 5

Due Date: October 6

Grace Etzel

Complete the assignment in Google Colab and typeset your wite-up in LaTeX. Show all work and code, and provide clear justifications for your answers. Submissions that are messy or poorly organized may be returned with a grade of zero.

Link to Google Colab: Link

Problem 1.

(a) **Part 1: Interpolation Using Chebyshev and Equally Spaced Nodes:** Implement a Python program to interpolate the function

$$f(x) = \frac{1}{1 + 12x^2}$$

on the interval $[-1, 1]$ using Chebyshev interpolation nodes. You may use the provided lecture code to generate Chebyshev nodes and construct the interpolation polynomial via Newton's divided differences.

Then, repeat the interpolation using equally spaced nodes and Newton's divided differences. Plot and compare the resulting interpolants from both approaches. How does the behavior of interpolating polynomial differ between Chebyshev nodes and equally spaced nodes? What do you observe near the interval endpoints?

Solution. When we use Chebyshev interpolation, with $n = 4$, the function seems drastically different than the original function. However, this would change as we increase the number of nodes. Additionally, the equally spaced nodes suffers from a similar issues; however, equally nodes appears to be a vertical shift of the Chebyshev interpolation. More specifically, notice that the Chebyshev interpolation is closer to the orignial function point on the interval $[-1, -0.4] \cup [.4, 1]$, whereas equally spaced nodes is closer to the function on the interval $(-0.4, .4)$. This intuitively makes sense because Chebyshev interpolation has a bias towards the endpoints, which allows it to be closer to the original function points compared with equally spaced nodes.

(b) **Theoretical Justification via Nodal Products:** To illustrate why Chebyshev nodes yield lower interpolation errors, consider the following theoretical result:

Theorem 1. *Among all choices of nodes in* $[-1, 1]$*, the Chebyshev nodes minimize the maximum absolute value of the nodal product:*

$$\min_{x_1,\ldots,x_n} \max_{x\in[-1,1]} \left| \prod_{i=1}^{n} (x - x_i) \right| = \frac{1}{2^{n-1}}.$$

This is the mini-max problem of interpolation.

Using code, compute and compare the following quantity:

$$\max_{x\in[-1,1]} \left| \prod_{i=1}^{n} (x - x_i) \right|$$

for both Chebyshev nodes and equally spaced nodes. Generate a plot of the nodal product in each case. Does your numerical results confirm that Chebshev nodes produce a smaller maximum for the nodal product? Discuss how this observation supports the theory that Chebyshev nodes minimize the worst-case interpolation error.

Solution. First, I would like to validate that the errors we see from the nodal product perfectly matches the explanation and graph we had above. That is, the equally spaced nodes have a greater error from the original function along the ends compared with Chebyshev nodes. The reasoning behind why this happens is because Chebushev nodes attempts to minimize the maximum absolute value of the nodal product. We calculate this value in the code, which shows that the maximum value of the nodal product is smaller for Chebyshev's than for equally spaced nodes. This helps support or illustrate that Chebyshev nodes minimizes the maximum absolute value of the nodal product.

Problem 2.

(a) **Interpolation with 4 Nodes:** Interpolate the function

$$f(x) = \sin(x)$$

on the interval $[0, \pi/2]$ using 4 Chebyshev nodes. Construct the interpolation polynomial using Newton's divided differences, and plot the pointwise error between the interpolant and the original function over a dense grid.

Next, repeat the interpolation using 4 equally spaced nodes in $[0, \pi/2]$. Plot the corresponding error and compare it to the error from the Chebyshev-based interpolant. How do the errors differ in shape and magnitude? What do you observe near the endpoints of the interval? Which method gives better accuracy, and why?

Solution. When we compare the graph between Chebyshev interpolation and equally spaced nodes, we do cannot visually see the difference, since the error for both is relatively small. So, to look at the error comparison graph, notice that difference in the equally spaced nodes compared with the Chebyshev error at the end points is significantly greater than this difference seen in the middle of the graph (near the original function's inflection point). Therefore, the better estimate is that which minimizes the error on the ends. This is the characterization of the Chebyshev error, which we see int the error comparison graph.

(b) **Part 2: Interpolation with 10 Nodes:** Repeat the comparison using 10 nodes for both Chebyshev and equally spaced cases. Plot the errors and analyze how the increased number of nodes influences the quality of interpolation. Does increasing the number of nodes reduce the interpolation error for both methods? How does the error distribution change? Is one method more stable than the other?

Solution. In the original function graph, we can subtly see a better approximation of the original function for both equally spaced nodes and Chebyshev interpolation. In fact, both of these methods seem to almost completely match the original function. However, when we look at the error comparison, we start to see a clearer picture. Notice on the axis that the error is very small. Nevertheless, there is a large jump towards the ends for the equally spaced error, whereas in Chebyshev the jump in the ends is comparatively much smaller. Therefore, while they both seem to perfectly fit the model in the original graph, looking at the error shows that Chebyshev's nodes edges the equally spaced nodes on minimizing the error.

(c) **Determining the Degree Required for High Accuracy:** Determine the smallest degree n such that the Chebyshev interpolating polynomial $p_n(x)$ satisfies

$$\max_{x\in[0,\pi/2]} |f(x) - p_n(x)| < 10^{-10}.$$

Hint: Use the theoretical upper bound on interpolation error involving the nodal product and the $(n+1)$-st derivative of $f(x)$. Then, verify your result in code by incrementally increasing n until the error condition is met over a fine grid in $[0, \pi/2]$.

Solution. When we solve for the smallest degree in n such that the condition above holds. We find that $n = 9$. Look at the colab to find the code and explanation.