Homework 1

By Jonathan Pilling

Problem 1a.

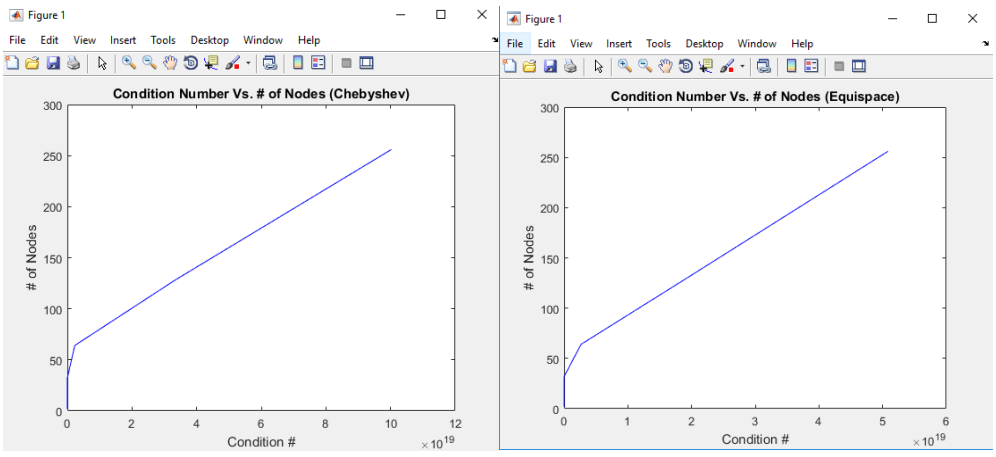
Building the Vandermonde matrix with Chebyshev extrema. First off, I changed the professors code to use the Runge Function. In the demo code, we were provided the x coordinates that were linearly spaced between [-1,1]. For every iteration of interpolation node, this would generate an equispace number of nodes to attempt and fit to the curve. For implementing Chebyshev extrema, I wrote a for loop for the number of nodes in the current iteration. This would assign x values to be their respective values from the Chebyshev Extrema equation.

```
for i=1:Nd(it) %Chebyshev Extrema equation
    x(i) = cos(((i-1)/(Nd(it)-1)) * pi);
end
```

1 is subtracted from our value of (i) so we start at 0, and iterate up to N-1.

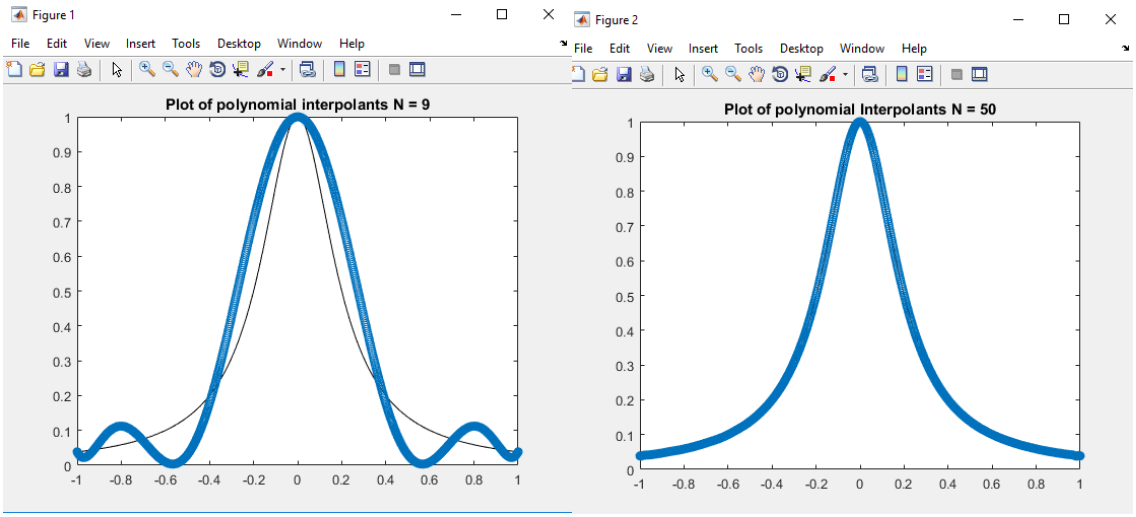
Problem 1b.

Generate a plot of condition number of the Vandermonde matrix vs. # of nodes. I wrote scripts for this problem, one for Chebyshev and one for equispace. I wanted to save x values that would be the condition number of the individual matrices that were created. So every iteration of the loop that goes over different number of interpolation nodes, I would save the condition of V using the Matlab cond() function. At the very end of the program, I plotted those values I saved vs. Nd array because this was the different number of nodes.

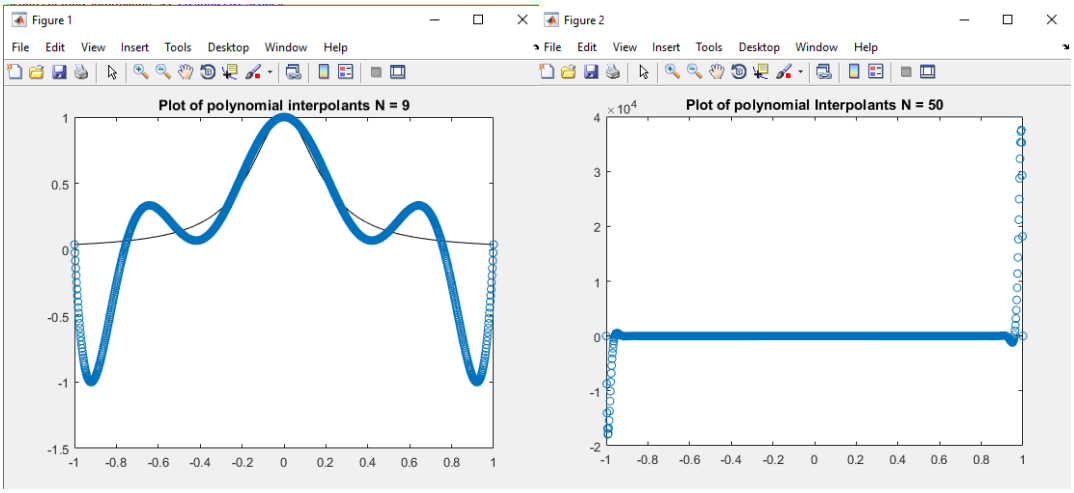


Problem 1c

This problem was pretty straightforward. Taking my existing code for part a, I set ND to be the value of 9 and then in a different section to be the value of 50. Instead of running the interpolation for a bunch of different Node values, you just needed to run it for a singular value of 9 and 50.



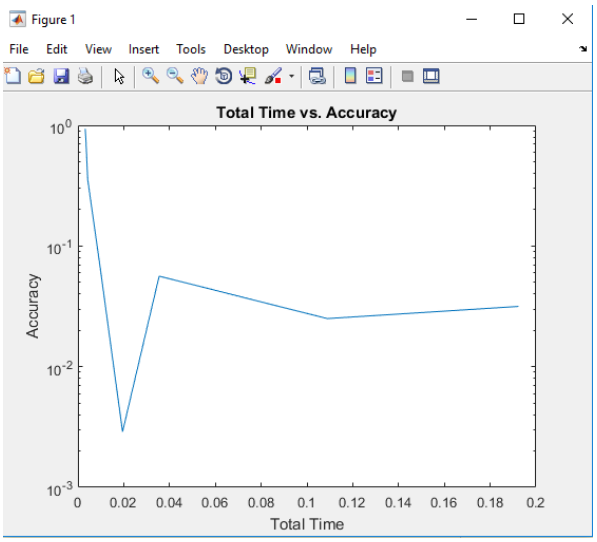
(Chebyshev)



(Equispace)

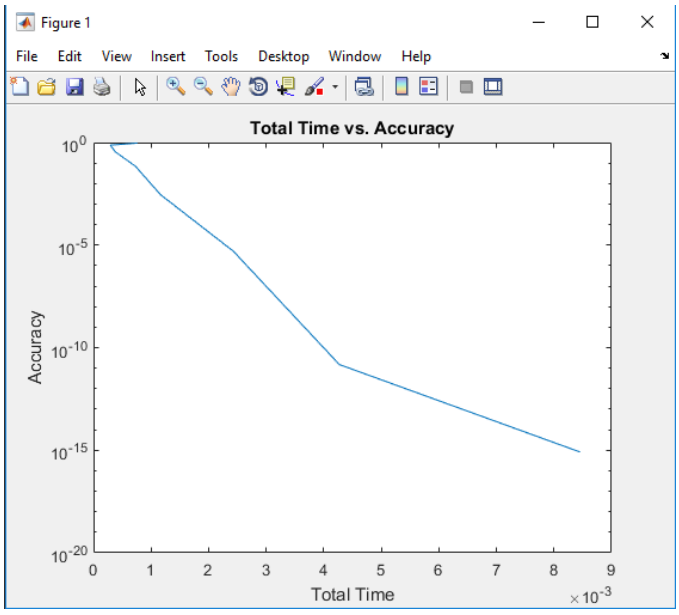
Problem 1d.

The current code evaluated this at 1000 points. I changed this value to 10,000. Placing the tic and toc functions inside the biggest for loop, it would capture the time it took to process a certain number of nodes. Every time I called toc, I would then save this value into an array that I would use later for plotting. At the end I plotted a semilog y plot with time values and error values e2.



Problem 1e.

Using the professors Barycheb program I was able to assign values to the ye_num matrix by plugging in the x values, the y values, and the 10,000 points. I feel like this one ran a lot better than the Vandermonde matrix. Since the Accuracy labeled is the error, the error going down is a good thing in my graph.



Problem 1d.

How does polyfit compare in terms of accuracy and computational cost to the barycentric approach? How does it compare to the Vandermonde approach?

Computational cost seems to be equivalent for the two approaches. The accuracy for this seems to be a lot worse than the other two attempts. You can see at the beginning the graph starts out pretty accurate, but as the number of nodes increases so does the time and measured value of error.