Alex Merino

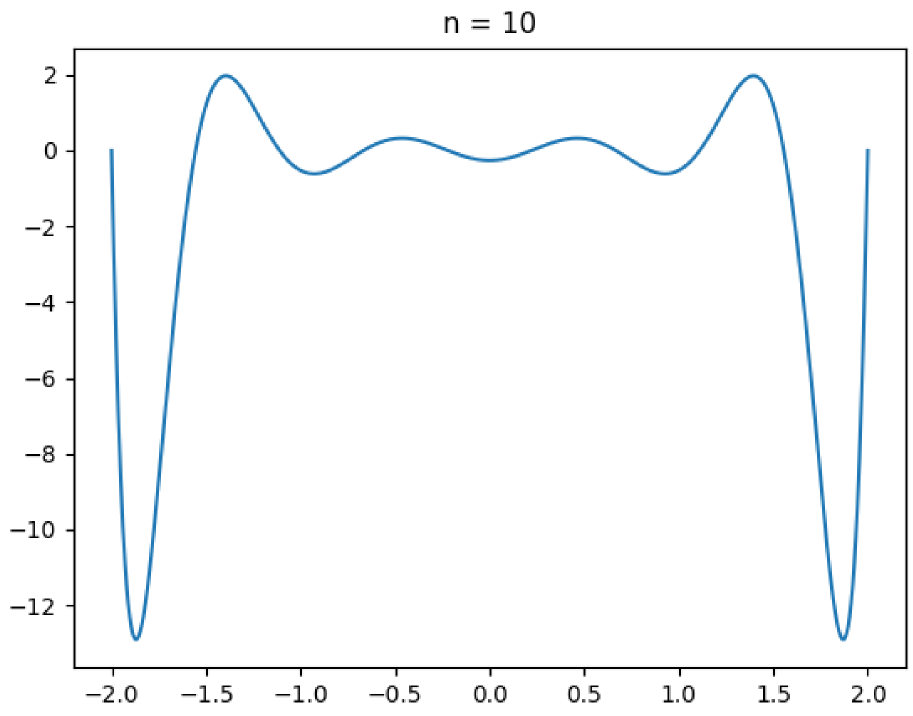
MTH 3312

Due: 11/6/2024

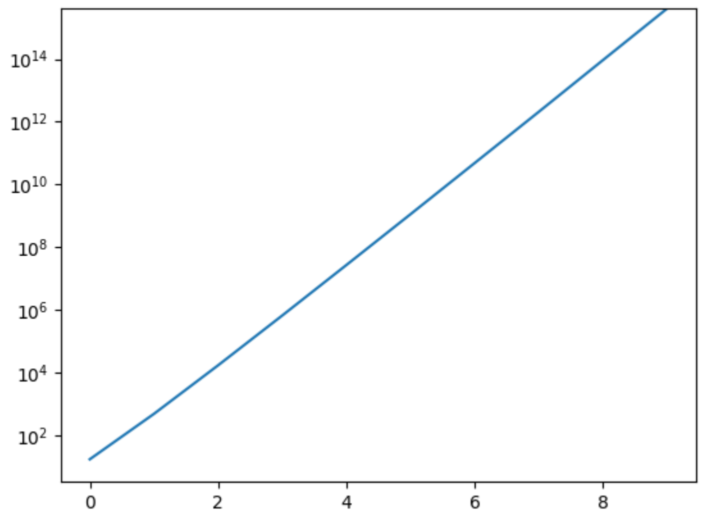
All code documented in the python files as well.

## Q1)

When  running, the *q1.py* file, the first 10 plots to show up are the functions $g_n = (x-x_1)(x-x_2)\dots(x-x_n)$, for the 10 different n values that were given. I did not want to fill up the report with these 10 different images, so when running the file, they will show. An example is when n = 10, shown below.

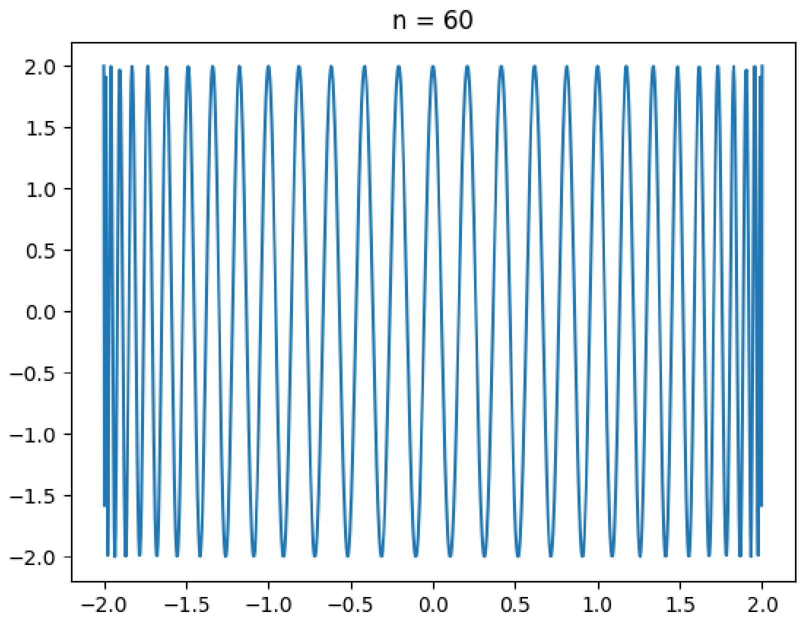


The 11th graph to show up should look as such:

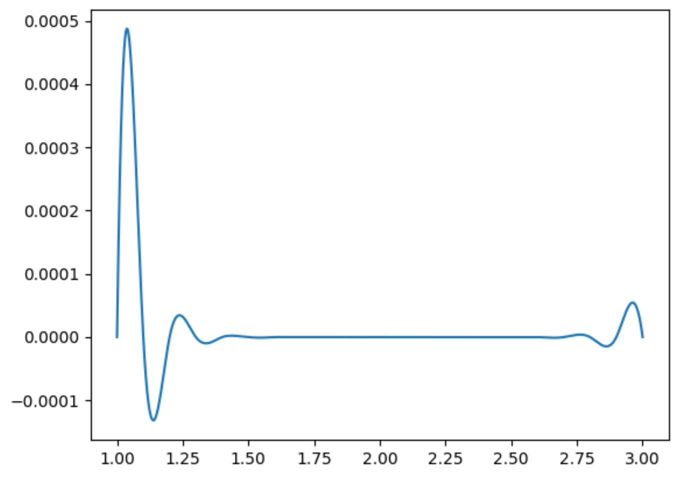


where we have the maximum values of each graph plotted on a logarithmic scale. The rate is ~4.

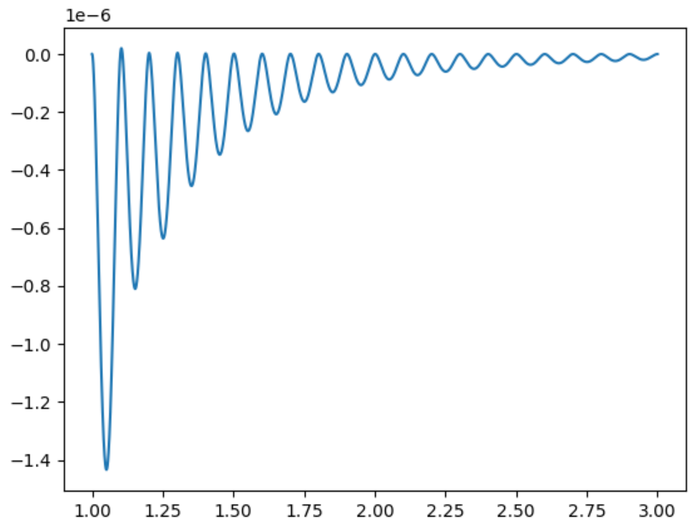
After the maximum graph shows up, the next plots to show up are the Chebyshev plots. As the number of nodes in the polynomial increases, the number of oscillations increases, and becomes very tight towards the endpoints. Below is when the number of nodes is 60.



## Q2)



Plotted above is the graph of the error between the natural cubic spline and the true values of ln(x). Near the beginning of the range the maximum difference is achieved with a value of $5*10^{-4}$. It oscillates for the values in the beginning of the range, but is very close to zero for most of the values. Near the end of the range, it starts to have a larger error.



Using the clamped splines with a value of 1 for the derivative on the left and a value of 1/3 for the derivative on the right, we get this graph which at first looks worse than the natural spline, but upon further analysis, it is actually 2 degrees better than the natural spline. The absolute value of the maximum error is $1.4*10^{-6}$, which is much better than the $5*10^{-4}$ from the previous graph. The magnitude of the error each oscillation decreases as the functions continue form 1 to 3. So using the clamped end conditions is better, other than the relatively large error from the end boundary on the right.