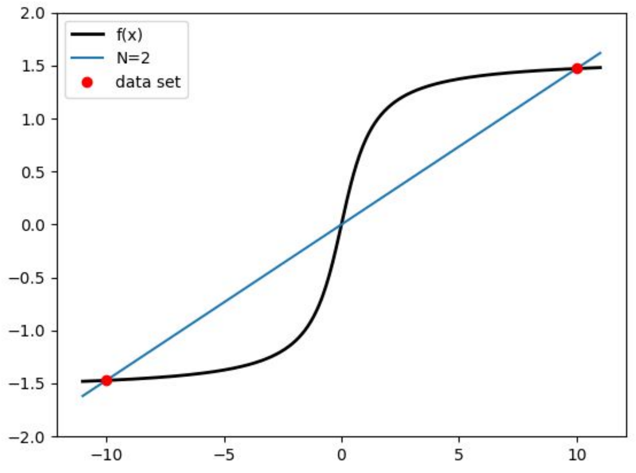
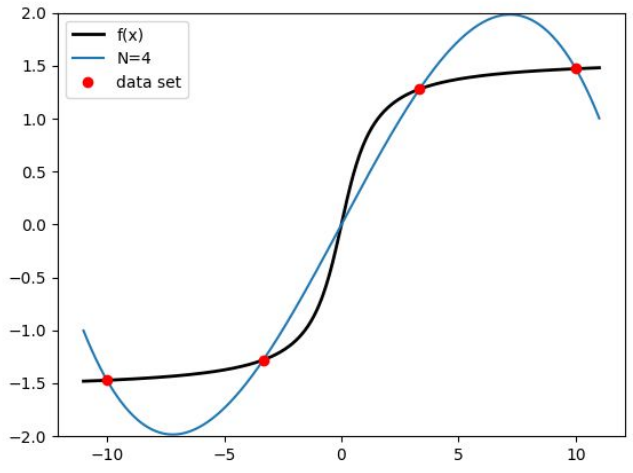
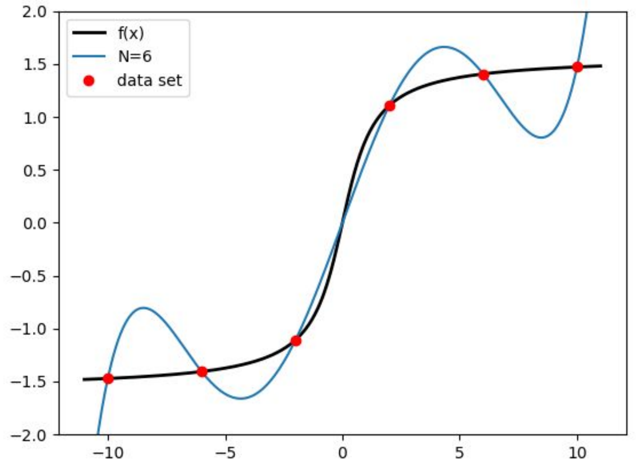
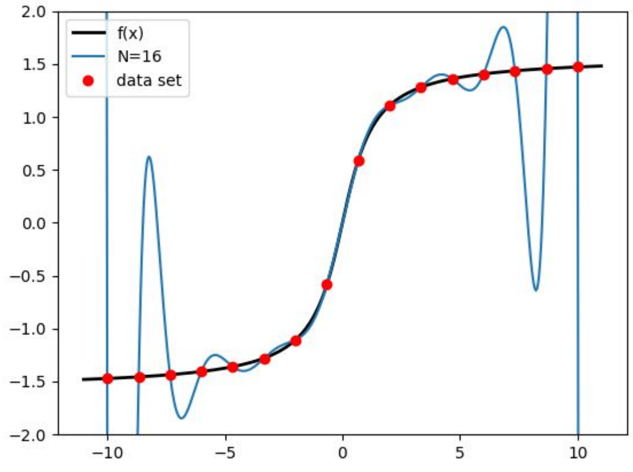
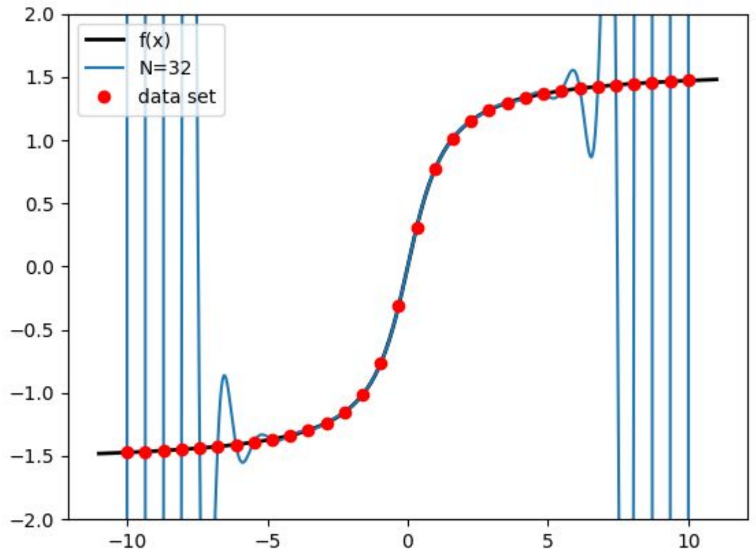
Below are the plots for $N \in \{2, 4, 6, 16, 32\}$ where N is the number of interpolating points for a polynomial that is used to approximate $arctan(x)\ x \in [-10, 10]$





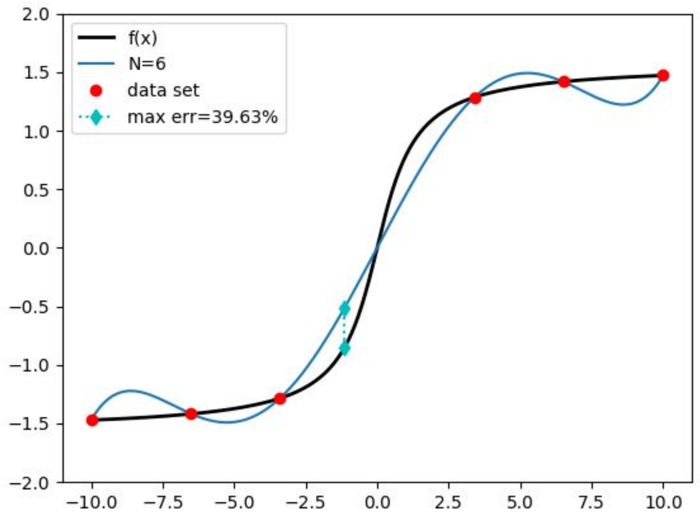




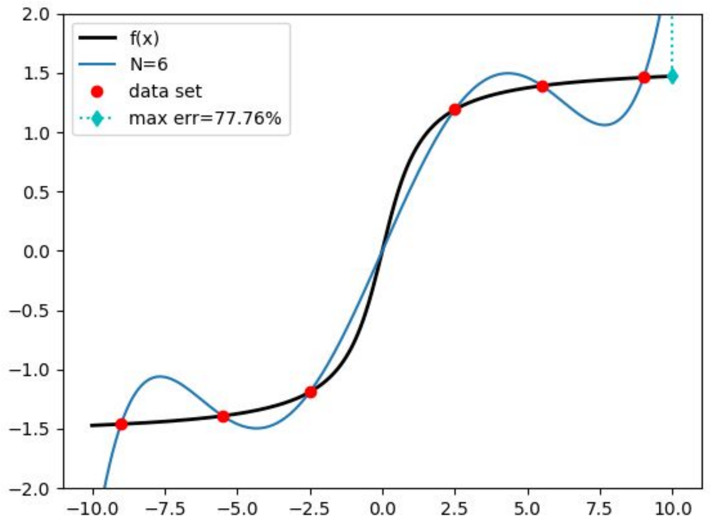


Immediately I noticed that increasing the number of points did not lead to better results. The polynomials are so osculatory that they would be horrendous to use for any sort of interpolation.

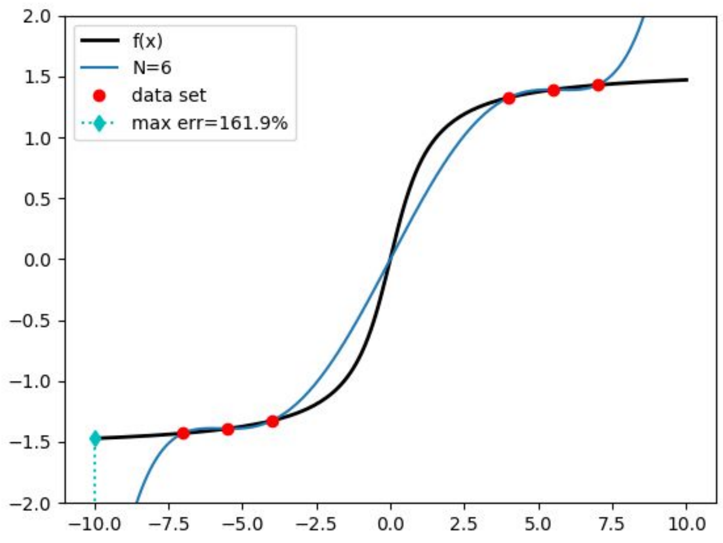
Then, we had to find an optimal distribution of points for $N = 6$**.**
Below is the best plot I could produce



The distribution was $x_1, x_2 \ldots x_6 \;=\; -10, -6.5, -3.4, 3.4, 6.5, 10$
I found that this was a daunting task. I tried moving the points as a group closer to the center, and the result was worse

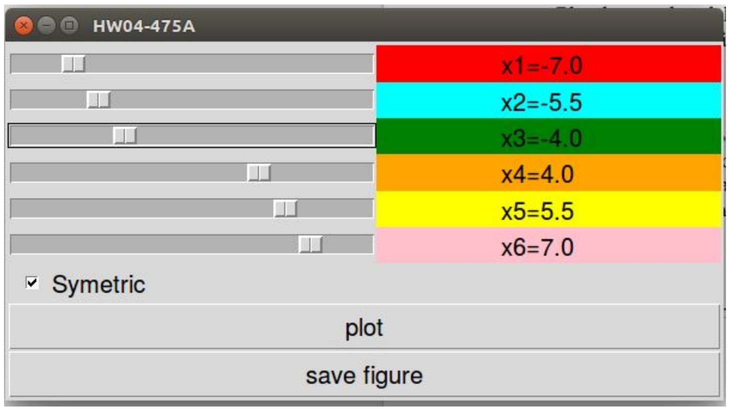


I tried to bunch the points up hoping that the polynomial would smooth out towards the ends, and be steep in the middle. This did not fair so well either, because beyond the points towards the extremes of the interval, the polynomial diverged quickly



I began to notice that the ratio of $\frac{\delta_{output}}{\delta_{input}}$ was likely very, VERY large.

Because I was changing the input and getting output which I could not feasibly predict, I developed a small GUI program to develop the plots for me, where I could pick the locations of $x_1 x_2 \ldots x_6$ with a slider bar, and click a button to plot, this allowed me to play around with the process of finding an optimal distribution, which actually aided me in finding the best distribution.


HW04-475A

x1=-7.0
x2=-5.5
x3=-4.0
x4=4.0
x5=5.5
x6=7.0

[✓] Symetric

plot

save figure

This GUI program is included in the submission, but it has several dependencies:

1) numpy
2) matplotlib
3) appJar

If you wish to use the python program, make sure you install these dependencies using python3’s package manager ‘pip’