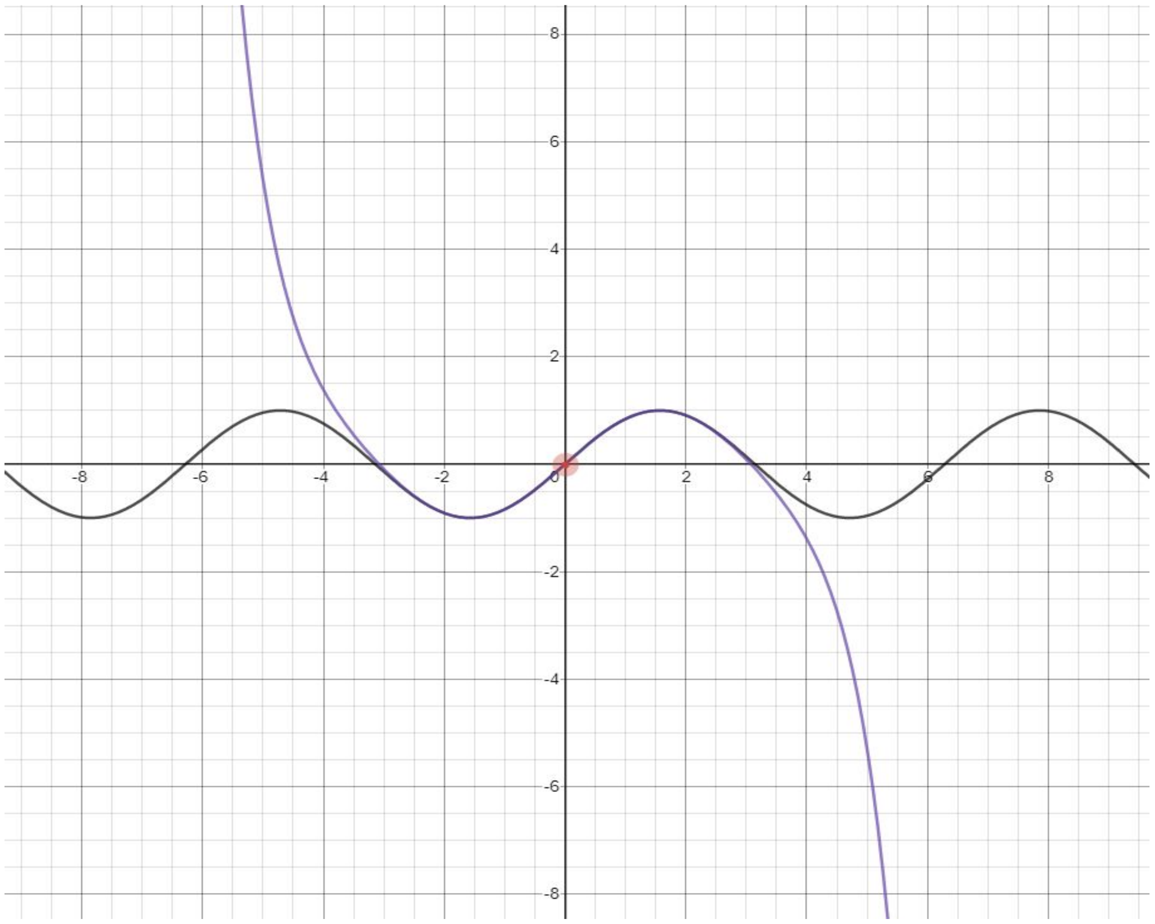
Assignment 2 - The differences of math.h vs homemade implementation

For my Lab I implemented the sin, cos, and tan functions using a Taylor polynomial to the 8th degree. A taylor approximation must be centered at a point, and for all three functions I chose to center mine at 0. This means that as we get further away from the point at which the function is centered , the accuracy of it decreases considerably. As an example, here is a graph showing the true sine wave and a taylor approximation of degree 8.



You can see that at about -pi and pi the approximation becomes almost worthless, however for any value between -pi and pi the approximation is decently accurate. The reason for the position of the inaccuracy relates directly to the order of the approximation. The higher the order the more accurate the graph. This is because for each order, we had a term that gives us a derivative that can describe more of the curve. Because of this, a taylor series of infinite order would be equivalent to the graph of Sin itself. This however would be inefficient to put it lightly and so for my purposes the taylor polynomial of the 8th degree works just fine. The math.h implementation uses essentially the same sort of method, however it has multiple algorithms and tables to help refine its approximations which gives it an almost spot on answer. All of this is true for Sine Cosine and Tangent.

For a quick explanation of why the Exp implementations give different results, The way that Exp is calculated is by using an epsilon to define how accurate an answer must be before it can be used. We had to use an accuracy of 0.000000001, which is extremely accurate, however the math.h function has a way of telling how accurate the epsilon should be to return a very close to exact result, which gives it better results and performance in most cases. Because of this the tolerances of the math.h function are much tighter, making it more accurate.