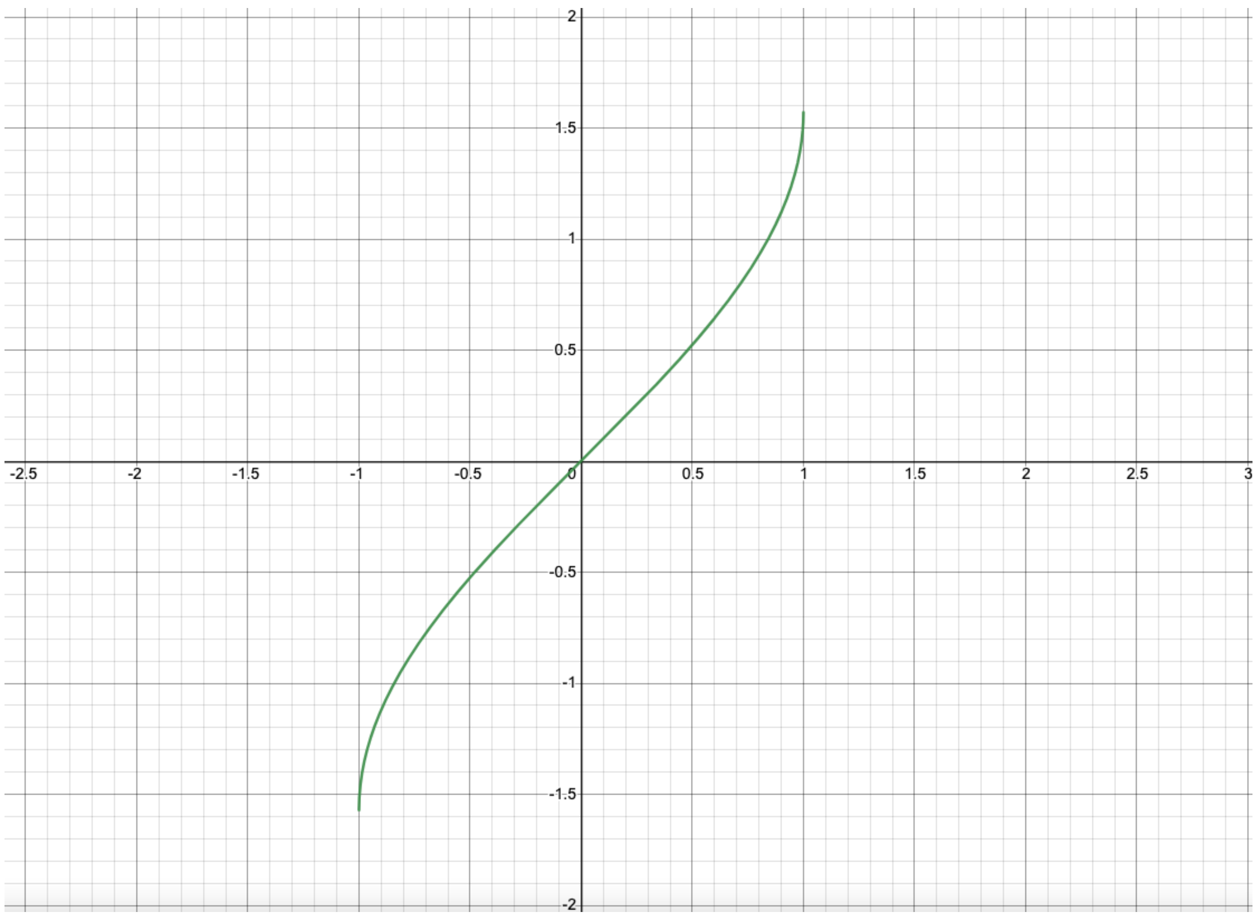
Rahul Vaidun
rvaidun@ucsc.edu
17 April 2021

CSE 13S Spring 2021
Assignment 2: A Small Numerical Library
Writeup Document

The program works by using Newton's method to calculate values of arcSin, and Log. To calculate the values of arcCos and arcTan the program uses trigonometric identities to calculate in terms of arccosine, this way only one function is being used.

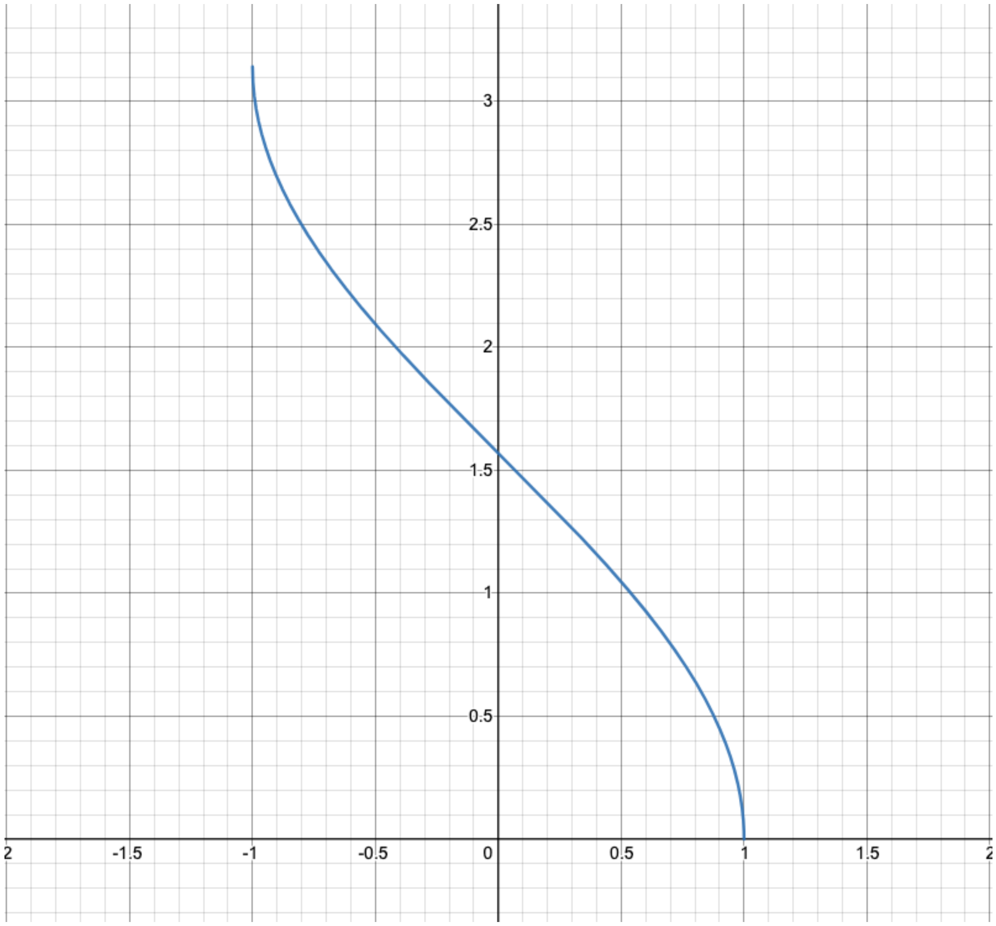
Calculating arcsin using Newton's method uses the formula f(x) = sin(x) - a where a is a constant, the value we are trying to find arcsin of. This method works well when the domain is from [-0.9,0.9]. However, as the value reaches the extremes the method rapidly loses accuracy. This is because the vertical asymptotes of the arcsine function are at -1 and 1 so as x gets closer to the asymptotes the change in x increases drastically, in turn losing accuracy for x.



The green line is the graph of arcsin(x). As we can see as x reaches 1 and -1 the slope of x increases, and the line becomes vertical. In order to increase accuracy for arcsine we can utilize trigonometric identities. When x is close to the extremes we can use the following identity

$$\sin^{-1}(x) = \cos^{-1}(\sqrt{1-x^2}),\ \ 0 \le x \le 1.$$

When we calculate arcsine in terms of cosine we are able to reduce the value of X. instead of having an x value that is close to the extremes we are able to reduce it by doing sqrt(1-x^2) and passing this value to the arccos function. By doing this we don't have a steep slope anymore.

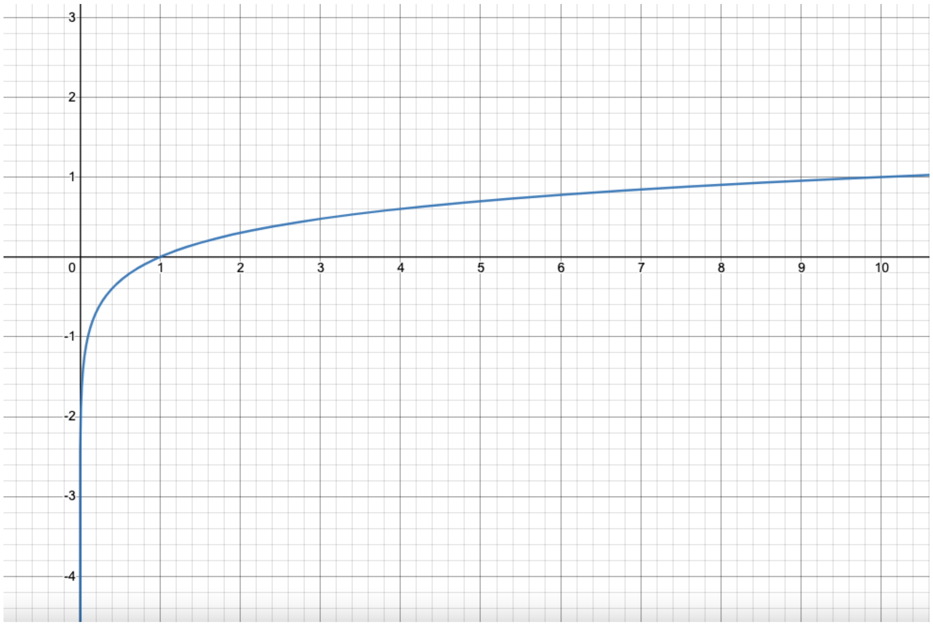


The blue line is the graph of arccos(x). As we can see the slope near the middle is much smaller than the slopes at the extremes.

Now that we have arcsin coded properly calculating the values of arccos, and arctan becomes much easier since we are using trigonometric identities.

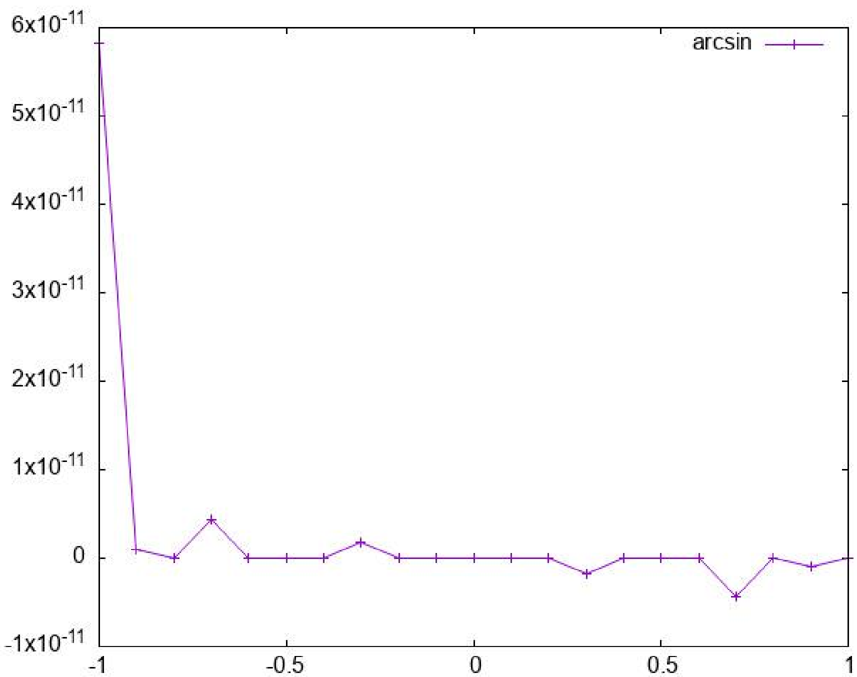
$$\arccos(x) = \frac{\pi}{2} - \arcsin(x). \qquad \arctan(x) = \arcsin\left(\frac{x}{\sqrt{x^2+1}}\right)$$

Log functions are calculated using Newton's method with the formula f(x) = e^x - a. Since we are only calculating the function between 1 and 10 Newton's method is accurate since the slope decreases as x increases.

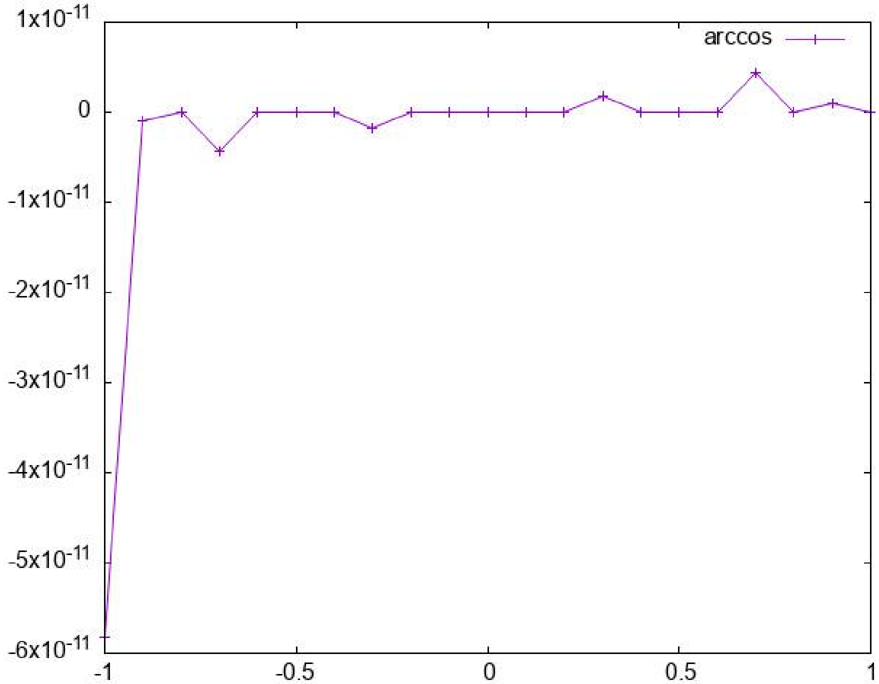


The graph above is log(x)

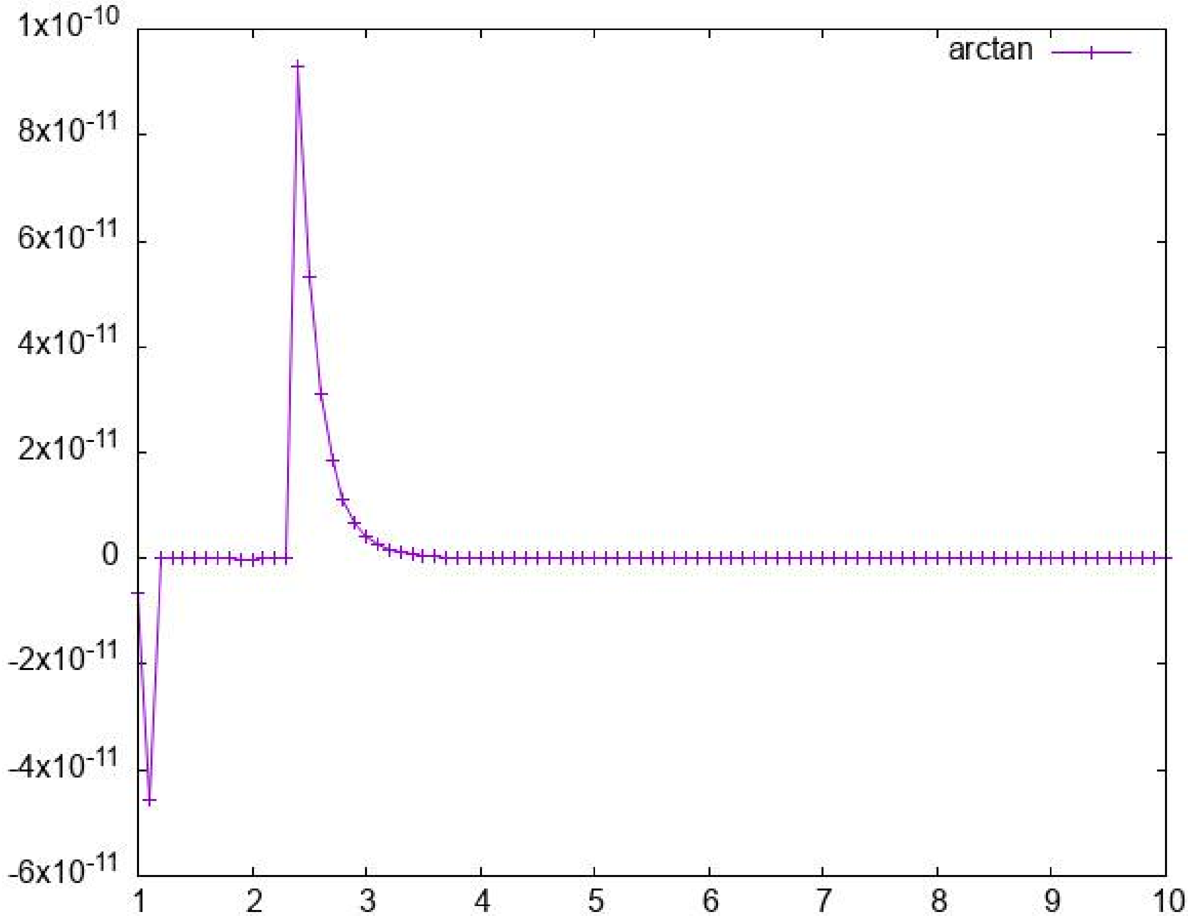
The difference between my implementation and the library implementation for the required functions are shown below. This decimal is accurate to 10 decimal places, however 15 decimals are shown in the graph



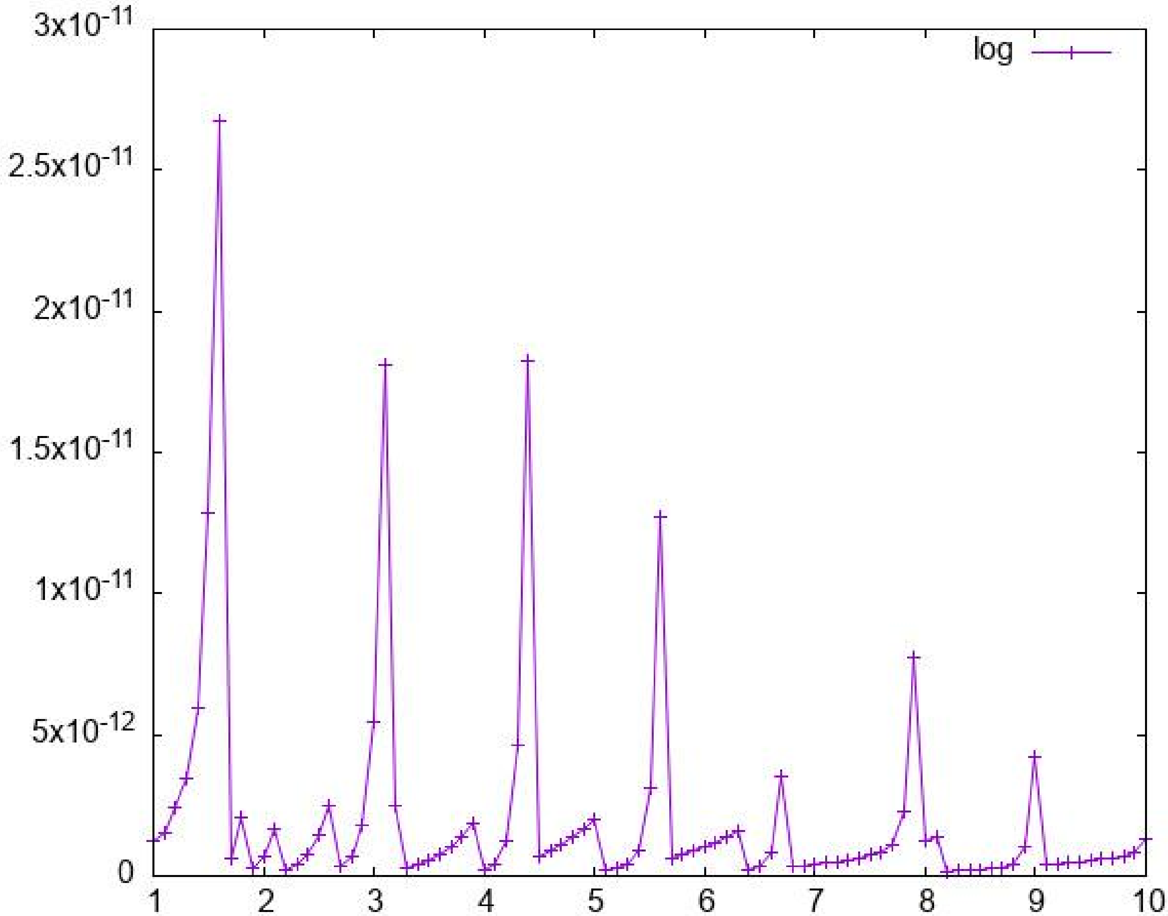
At x=-1 the difference spikes because x is being rounded up when the value is smaller than epsilon.



At x=-1 the difference spikes because x is being rounded up when the value is smaller than epsilon.



The graph spikes at x=2.1 and x=2.2 because x is being rounded up when the value is smaller than epsilon. In the beginning there is a sharp decline. I think this may be due to a luck factor of rapidly reaching a number smaller than epsilon. For arctan at x=1 the slope is steeper than when x gets bigger which may explain the inaccuracy. It is interesting to note that after x=3 there is nearly no difference between my implementation and the library



There are sudden spikes for log. This is because of the nature of floating point numbers. The actual answer probably cannot be represented by floating point exactly resulting in the spikes of the graph