Zack Traczyk
ztraczyk@ucsc.edu
Professor Long
03/30/21

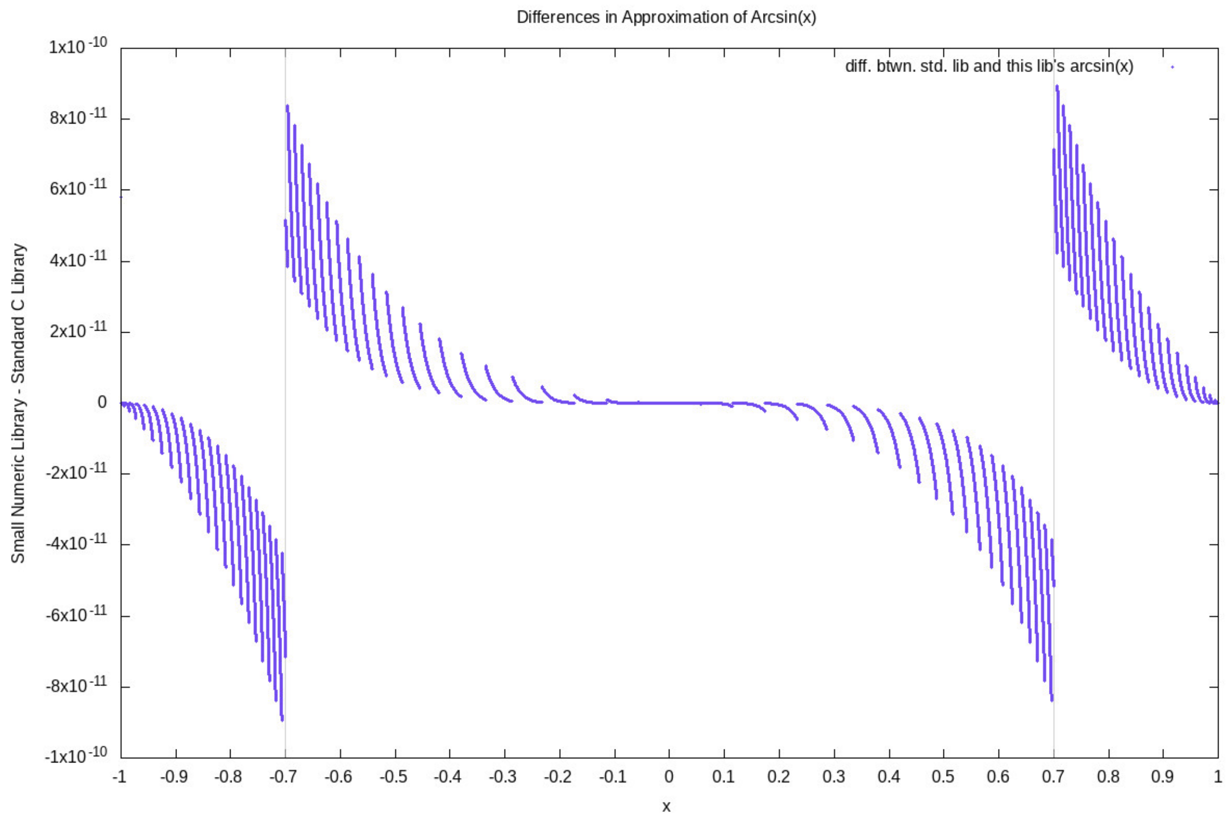
CSE13S Spring 2021

Assignment 2 - A Small Numerical Library

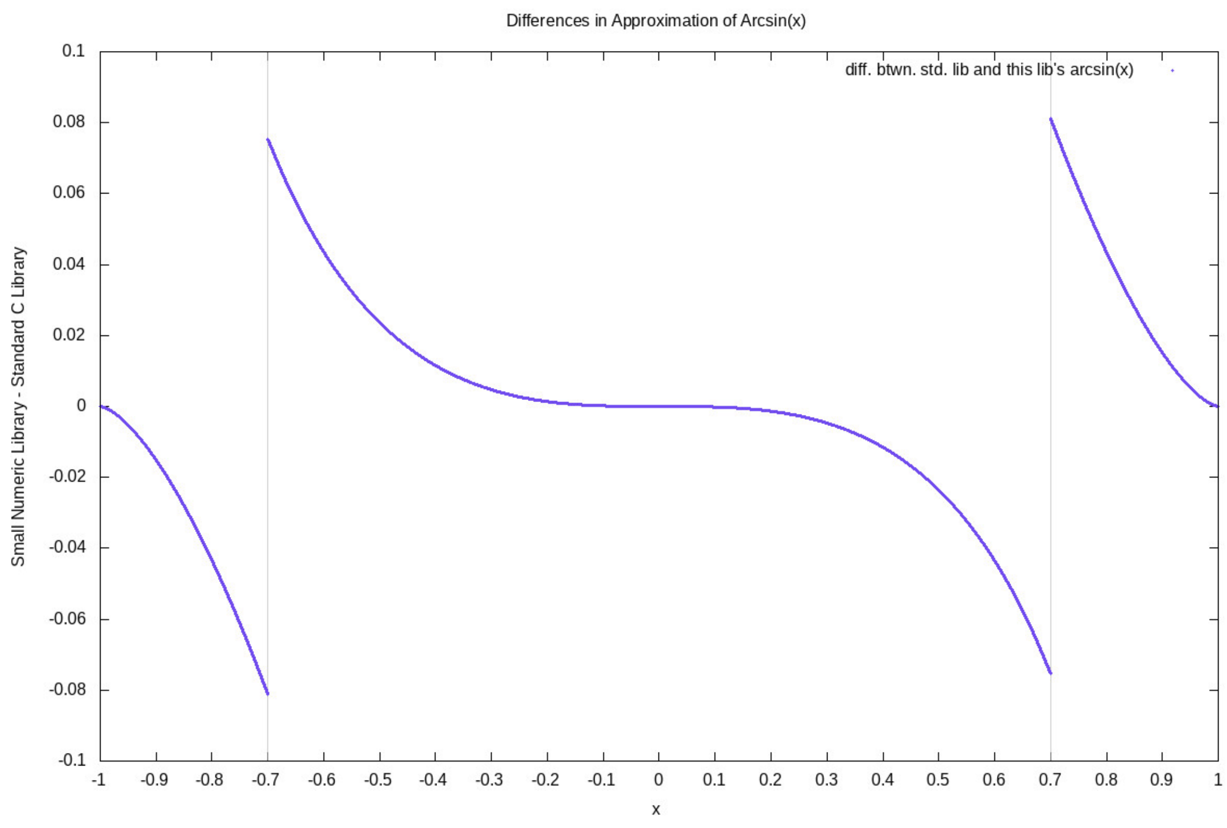
WRITEUP DOCUMENT

When analyzing my data there were quite a few discrepancies between my implementation and the standard C math library. These variances can be explained by looking at this library's implementations for arcsin, and subsequently arccos and arctan, as well as the implementation for log.

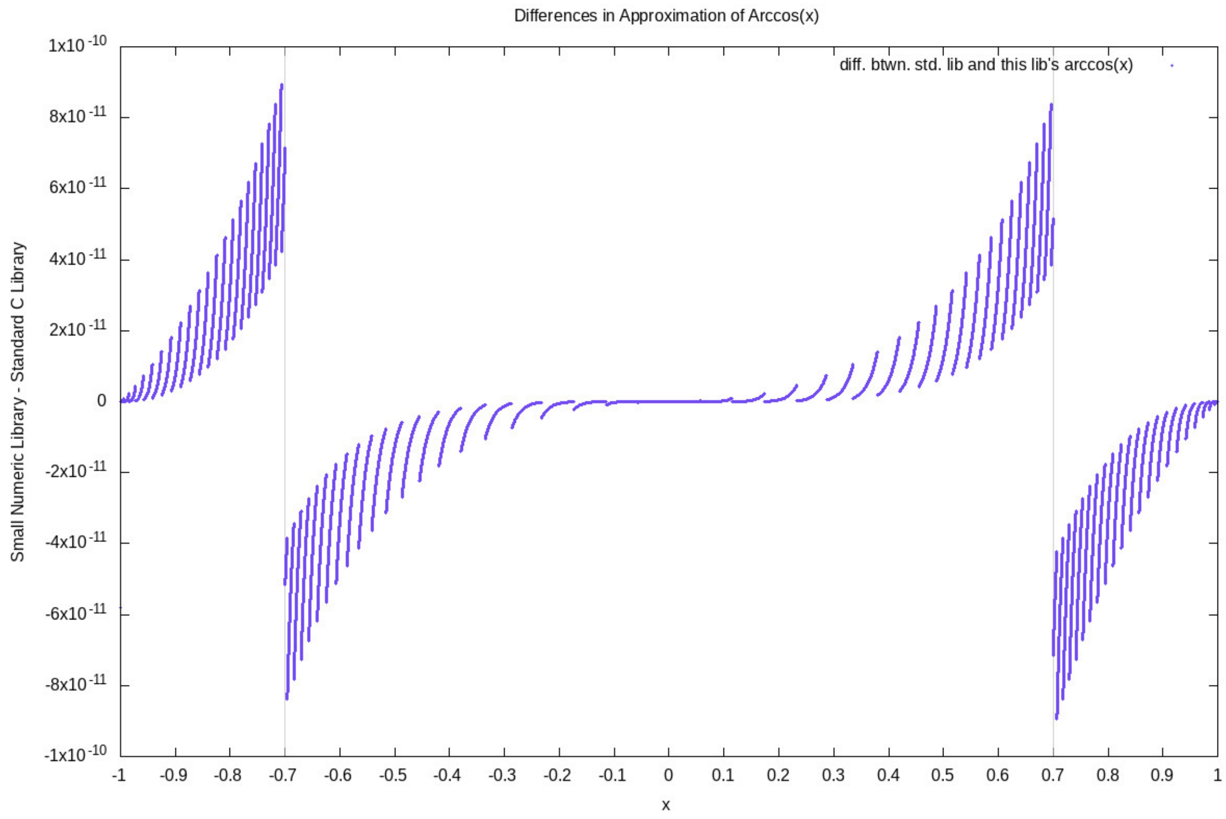
As discussed in DESIGN.pdf, arcsin is implemented by using the maclaurin series for arcsin when | *x* | <= *0.7* and the maclaurin series for arccos at $\sqrt{1-x^2}$ when | x | > 0.7. This was to reduce an inherent property of taylor series: the approximation becomes less accurate the farther from the center. The result can be seen in the graph below where the difference between the implemented numerical library and the Standard c library is graphed with intervals of 0.00001.



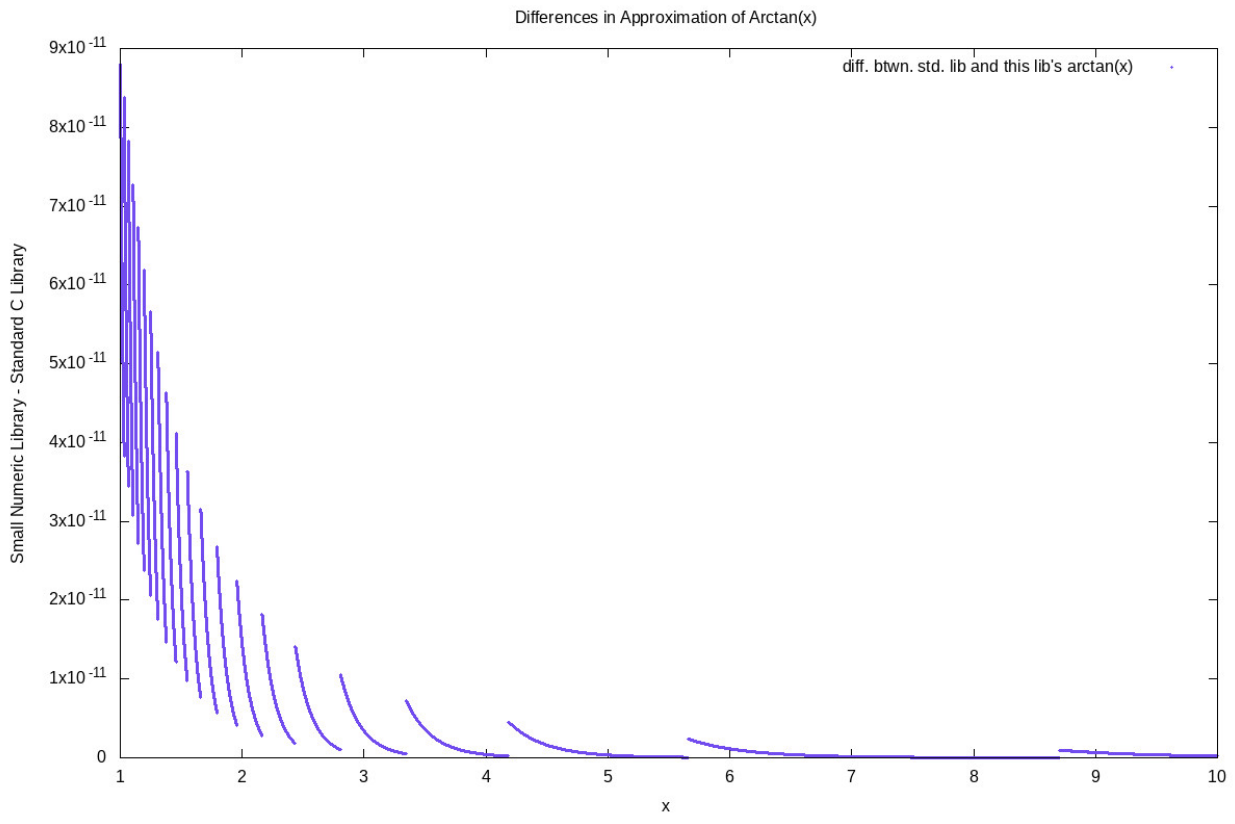
At x = 0.7 and x = -=0.7 light gray lines are drawn to show where the function switches from arcsin (in the middle) to the translated arccos function (on the outsides). Intuitively, the graph slopes away from a difference of 0, in other words losing accuracy, as the x values stray further from the centers of the maclaurin series. Another interesting property of the graph is the repeated nonlinear “waves”. These values are probably occurring because of inaccuracies in floating point representation. Since computers cannot actually represent fractions that are not composed of base 2 numbers (i.e. 1/3 or 1/10), I assume that it was these numbers that got less and less accurate as they were summed. Looking at the mclauren series, each term is increasing by x^2. Therefore, as x gets farther from 0, the magnitude of the terms increases exponentially, taking more iterations before the final term is less than epsilon. This explains why the error increases in a smooth slope, since more terms are being used, the inaccuracy is compounded on itself. To test this hypothesis I tried calculating arcsin with a set number of terms. My logic was that if given the same amount of terms, there would not be as dramatic of a sweeping motion. This is because the loss of accuracy would not affect the number of terms being calculated. There are 100,000 terms and that is it. The error does not get to compound on itself as much. As seen below, this hypothesis worked, proving that the waves were due to floating point inaccuracies compounded by increasing iterations.



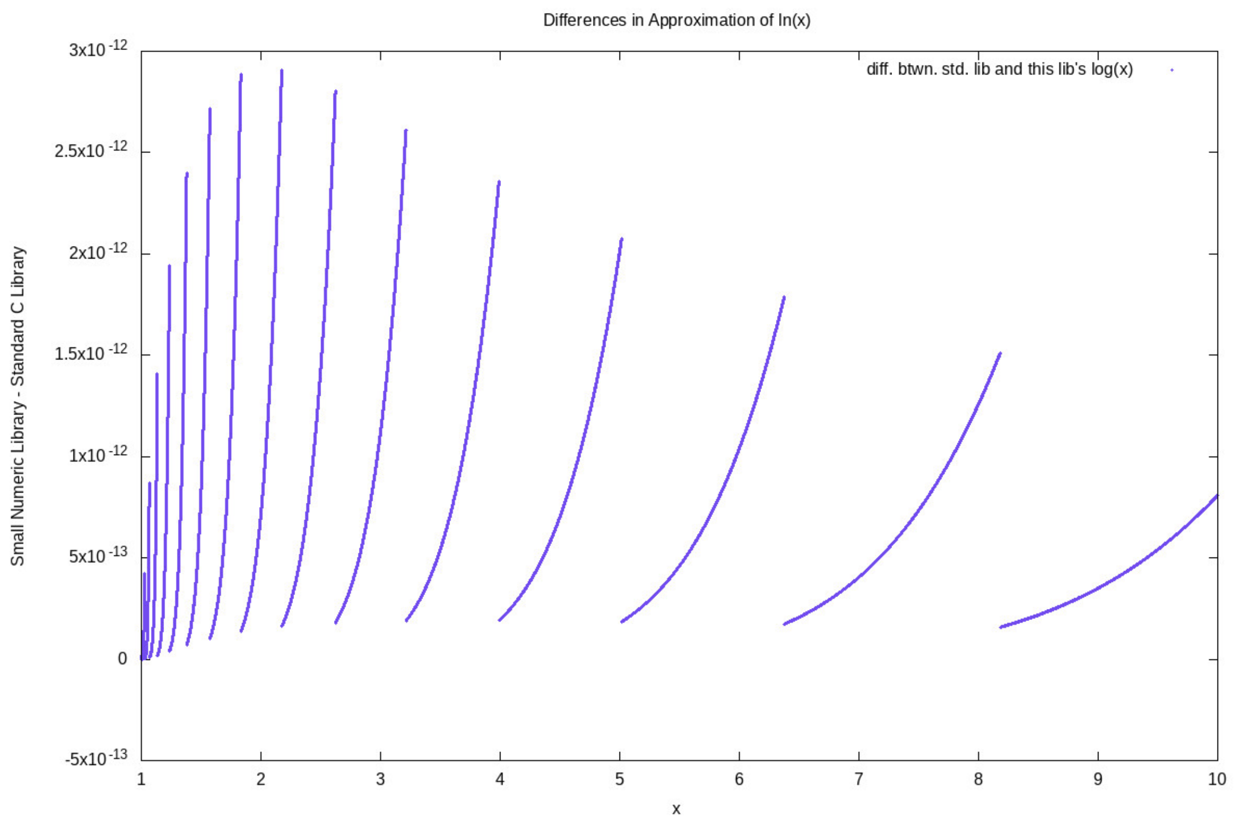
I implemented arccos using π/2 - arcsin; therefore, the differences in precision can be explained with the same reasoning as sin. They are implemented identically. The graph clearly shows this as it is identical to arcsin if it were flipped over the x-axis.



Like arccos, arctan is also using arcsin. However, in this case, x is being translate to $\sqrt{1+x^2}$. This results in a downward slope since the maclaurin series is centered further along the x axis to the right.



Unlike the trigonometric functions above, log, really ln, is calculated using newton's method. Terms are calculated until the difference between the previous and current term is less than epsilon. This means that the difference between library's is never greater than epsilon since the degree of accuracy is beyond that. The sweeping in the log graph seems to be caused by floating point inaccuracies as described in the trigonometric functions. As terms are represented with less accuracy, more terms are then required to shrink to epsilon and the error compounds until a base 2 friendly number is reached.


Differences in Approximation of ln(x)
diff. btwn. std. lib and this lib's log(x)
Small Numeric Library - Standard C Library
x
3x10^-12
2.5x10^-12
2x10^-12
1.5x10^-12
1x10^-12
5x10^-13
0
-5x10^-13
1
2
3
4
5
6
7
8
9
10