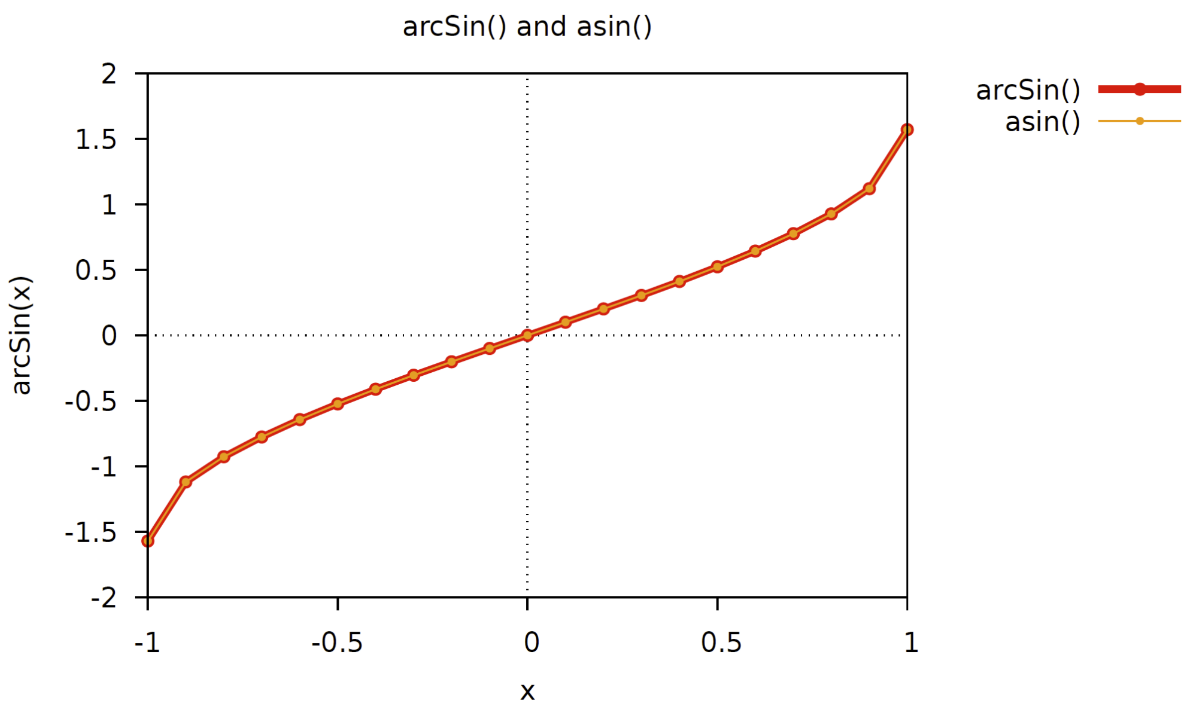
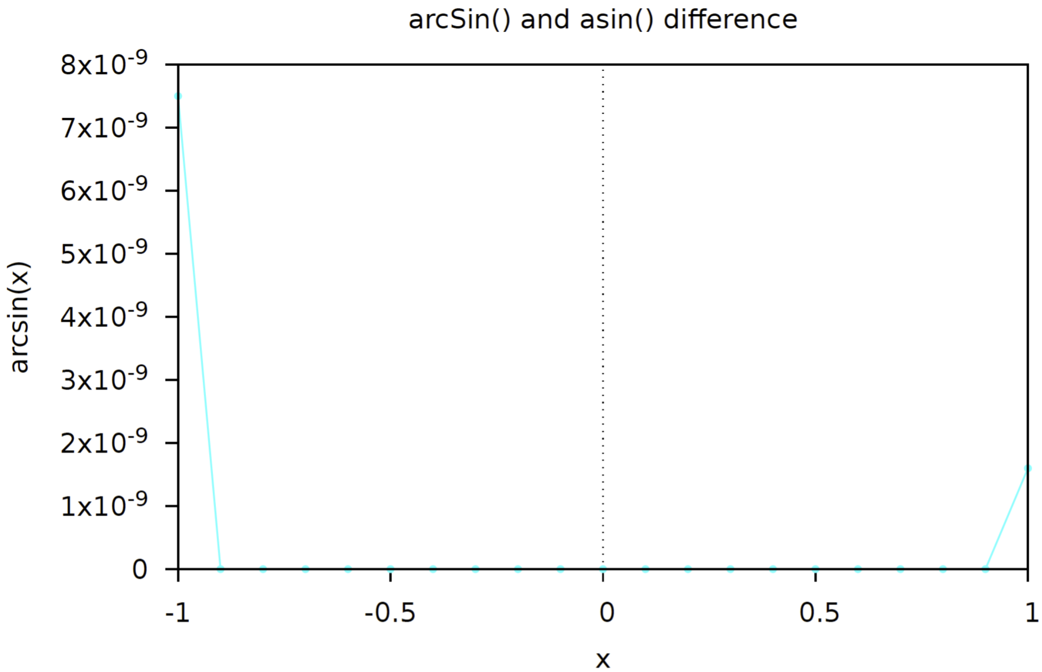
Natalie Valett, nvalett
CSE13S
Prof. Darrell Long
April 17, 2021

Asgn2 Writeup:

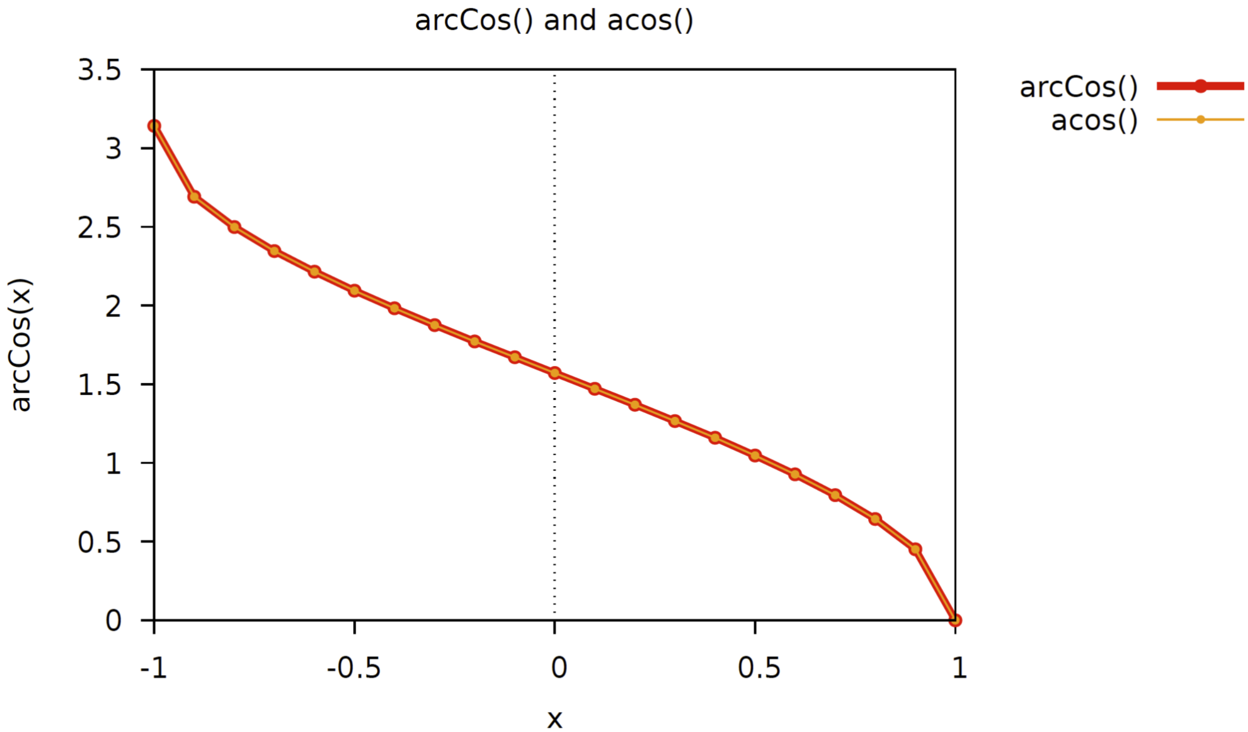
ARCSIN:

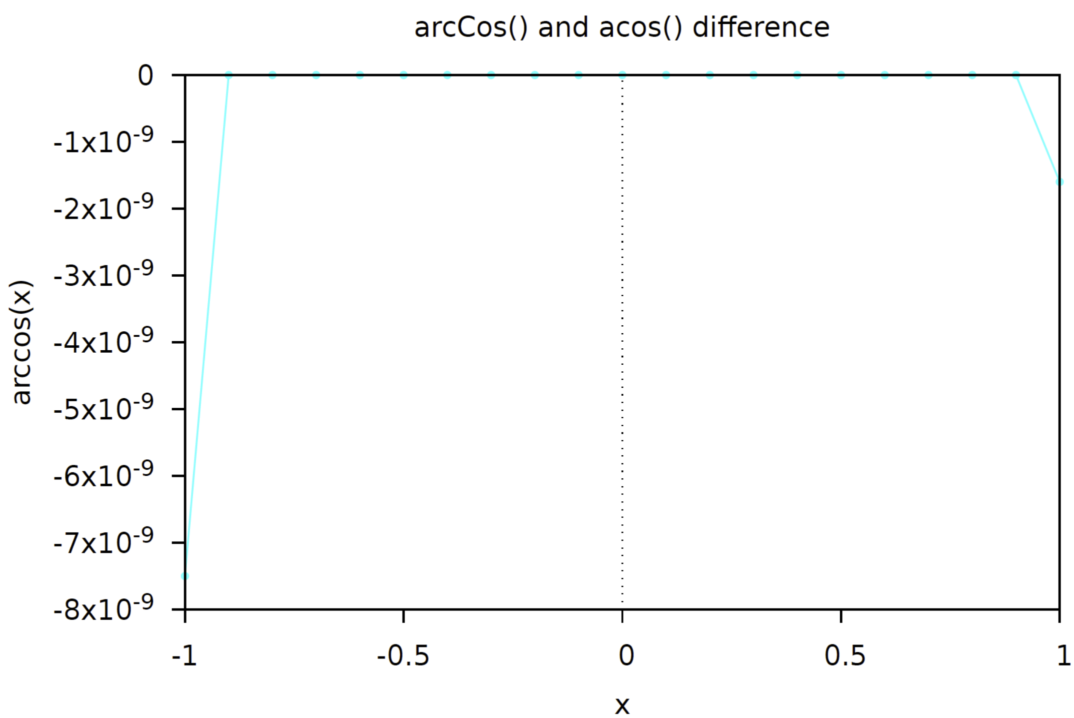




As can be seen in the first graph, my arcSin() function and the asin(0 function from the math.h library produce approximately the same result for all values of x from -1 to 1. Upon closer inspection, though, the second graph reveals that towards the endpoints of that range the 2 functions do produce slightly different results. According to the printed results from mathlib-test, at x = -1, the difference between arcSin(x) and asin(x) is 0.0000000075, with arcSin() returning a lower value than asin(). Less severely, at x = 1, the difference between the 2 functions is 0.0000000016. Besides these 2 divergences, all other values of x yield identical results between the 2 functions. This makes me think that the divergences are attributable to Newton's method perhaps losing accuracy as the value of x increases, or as it approaches the domain limits for the functions used in the calculation (sin and cos).

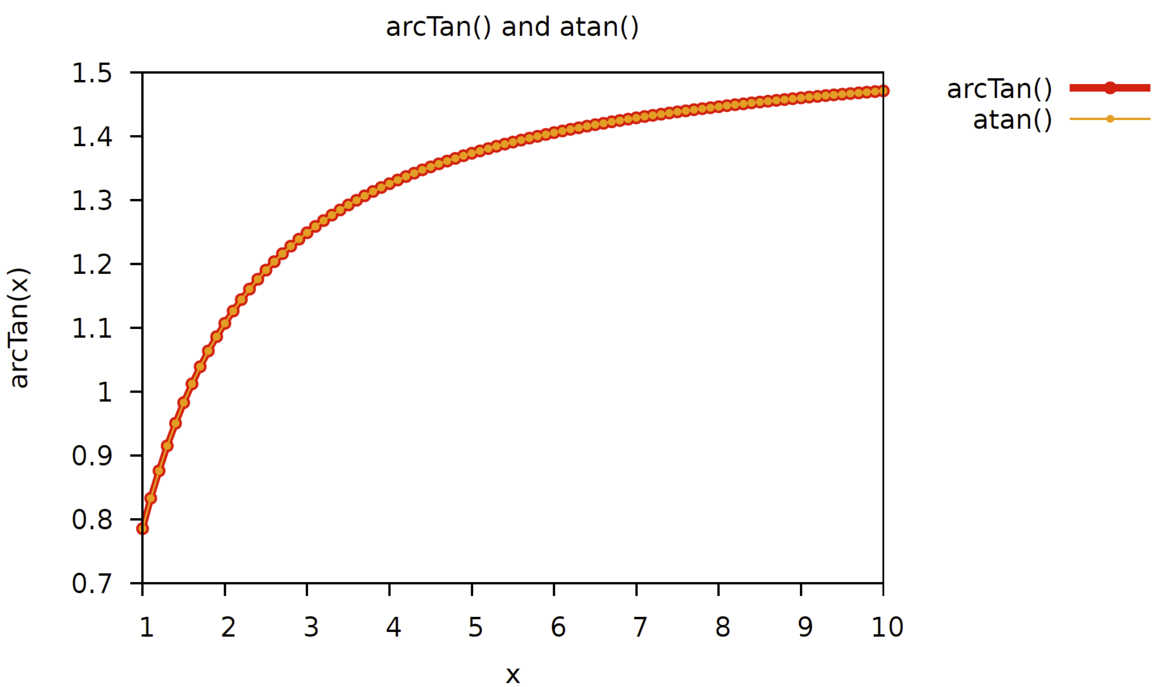
ARCCOS:

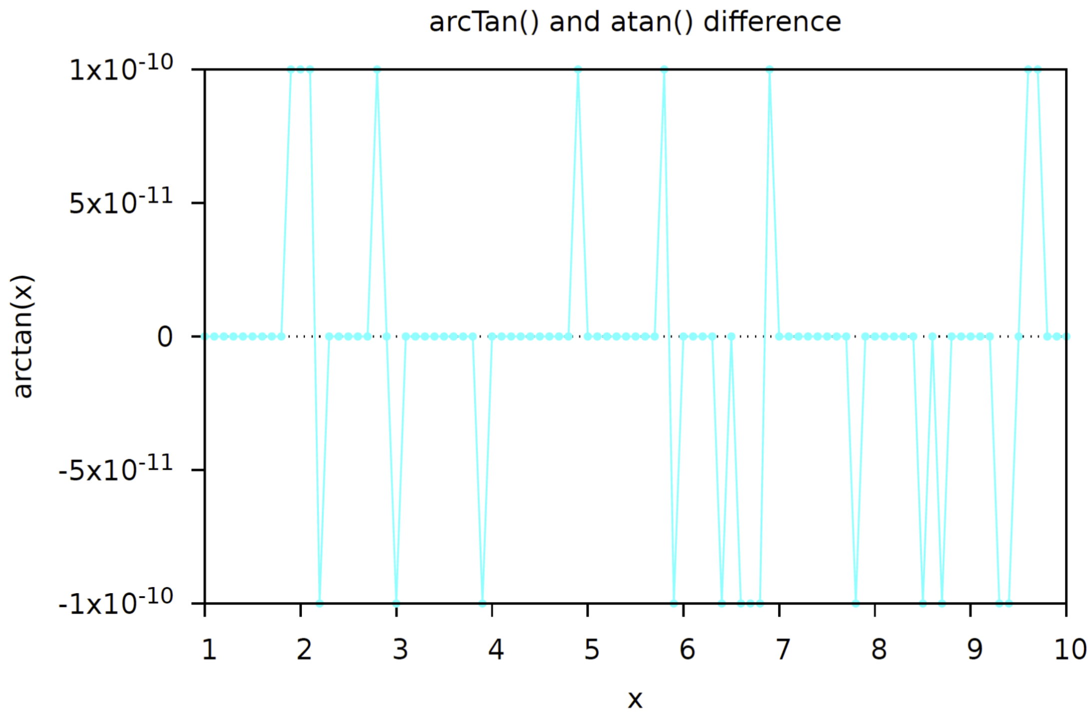




As with the arcSin function, arcCos() and acos() seem to produce the same result for most values of x, except for at the edge cases of -1 and 1. Because of arcCos's dependence on the arcSin() function, this similarity in divergence makes sense. arcCos() inherits error from arcSin() at the same values. To prove this, arcCos's calculated difference is of the same magnitude as arcSin's (0.0000000075 and 0.0000000016), but negative instead of positive because arcCos subtracts arcSin(x) from pi.

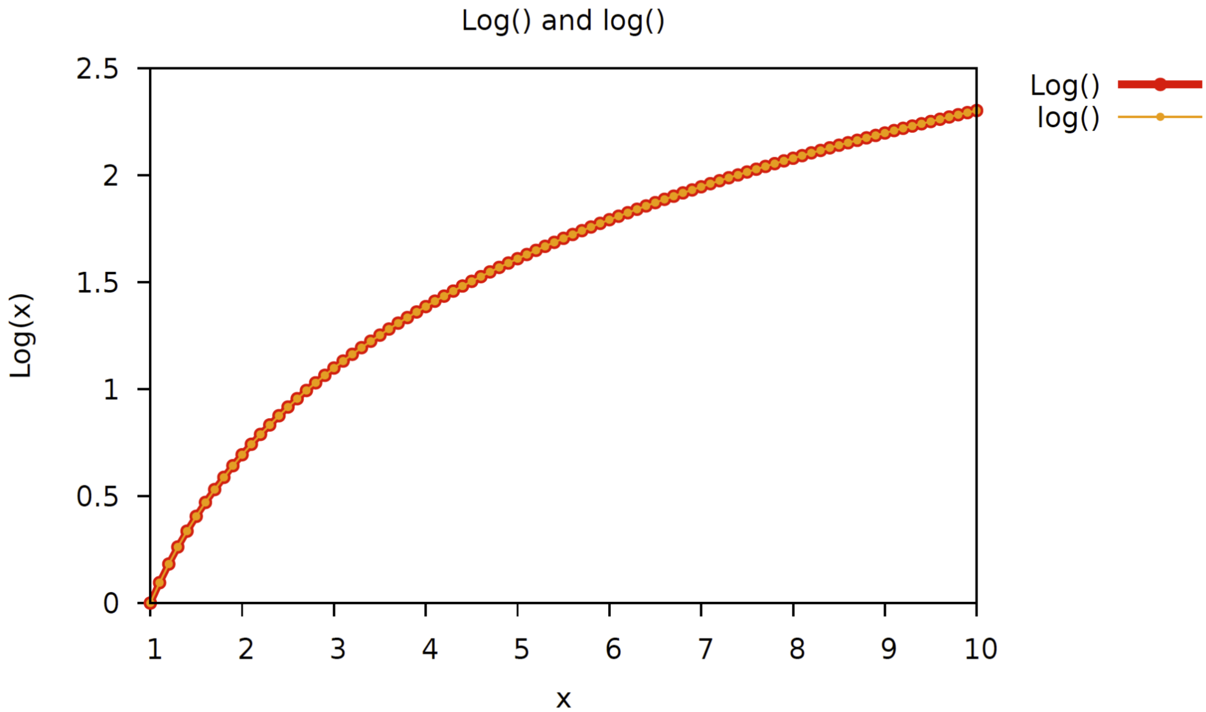
ARCTAN:

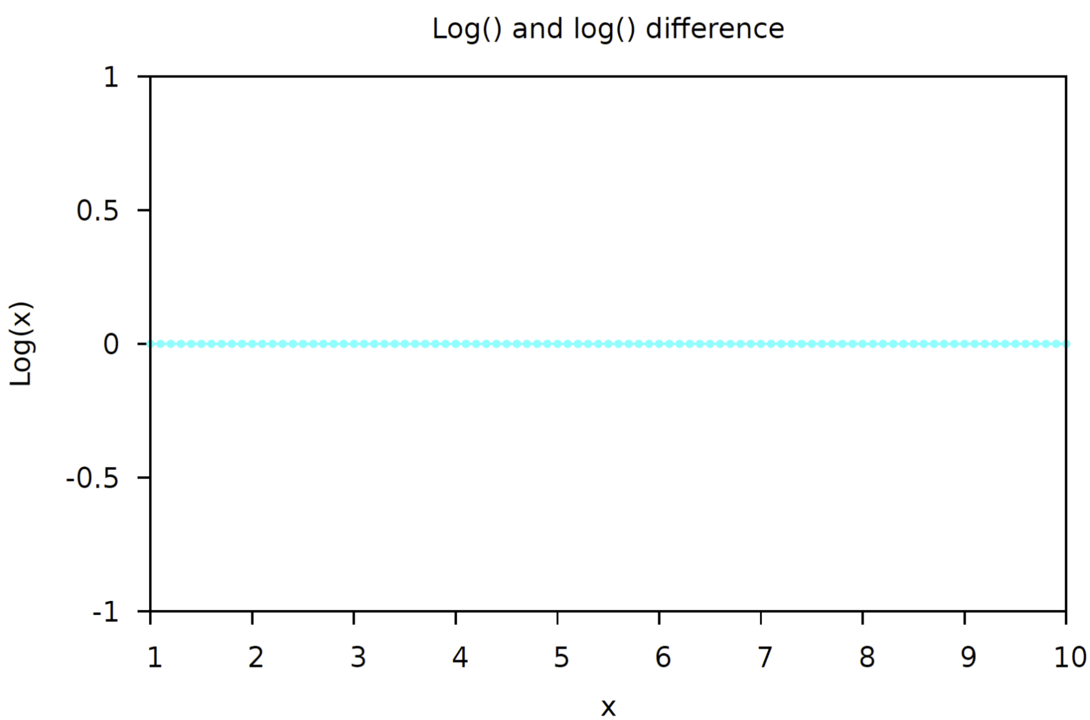




Although the second graph reveals that error is much more frequent in the arcTan() function than in arcCos and arcSin, it should be pointed out that these differences are of significantly less magnitude. Where there are divergences between the results of arcTan() and atan(), that difference is only of the magnitude 0.0000000001 (1×10^{-10}), which is only 1/16 of the error seen in the previous 2 functions. Because arcTan utilizes the arcSin() function, it's fair to assume that the error produced in arcTan() is due to the error generated in arcSin(). Unlike arcCos() however, arcTan()'s utilization of arcSin() does not simply pass x into it, but instead the value given to arcSin is $(x-e^{old}) / e^{old}$. This accounts for why the error generated is not constrained to only the same x-values where arcSin() generates error, but instead it's related to those x values where the value of $(x-e^{old}) / e^{old}$ (the x value passed into arcSin(x)) is close to or equal to -1 and 1.

LOG:





The Log() function I wrote produces no error from the log() function in the math.h library. This is the only function of the 4 that doesn't rely on arcSin(), so that helps to explain why this one produces no errors while the other 3 do. I also used Newton's method for this Log() function, so the computational method of this and arcSin() are the same.

However, based on my hypothesis that arcSin's error is attributable to Newton's method losing accuracy as the function domain is approached, Log's accuracy can be attributed to the range of x being tested. The domain of the natural log function is $[0, \infty)$, and we're only testing the function over domain $[1, 10]$. Because of its non-proximity to domain endpoints, perhaps the accuracy of the Log() function can be explained.