Design of mathlib

Moore Macauley
University of California, Santa Cruz

October 7, 2022

1 Introduction

In assignment 2, I was tasked with recreating the functions sin(), cos(), arcsin(), arccos(), arctan(), and log() from the c math library, with a maximum error of 10^{-10}. While the functions I created are good, they are not as accurate as the library's implementation. For each of the functions, there are several possible reasons as to why, which I will endeavor to explain now, using a graph showing the difference between the result of my functions minus the library's implementation. As I know that the libray's function is more accurate than mine, I will consider making my function more accurate synonymous to making it closer to the library function.

2 Sine and Cosine

As can be seen from Figure 1, while my implementation is good, with error well under the minimum of 10^{-10}, it is still quite different from the library implementation. Looking at this graph a clear pattern emerges, as the input gets further away from 0 the difference increases. The larger difference as inputs get further from 0 makes sense given that to implement sin and cos, I used a Taylor Sequence centered at 0. We know that Taylor Sequences get less accurate as they get further from their centers, so it must follow that my Taylor Sequence centered around 0 gets worse the further I am from it. The library implementation could be more accurate simply because it accepts a smaller error. As the number of terms in a Taylor Sequence gets larger, the error decreases, with an infinite number of terms being the exact value. If the library continues adding terms until the error is lower than the minimum error I used, this could explain the difference. Similarly, the library could be using a different center depending on the value. Sin and cos are functions that repeat, and have multiple values where the exact values are known. While for sin (0,0) is certainly the easiest, and what I used, $(\pi, 0)$ or $(\pi/2, 1)$ could also definitely work, with cos also having a number of values that could also be used. Based on that, the math library could be recentering itself depending on which known input and output is closest, greatly improving its implementation.

3 Arcsin and Arccos

Figure 2 contains the differences between my arcsin and the library's. One thing is immediately noticeable, the fact that it has a pretty big outlier at x=-1. The explanation for this outlier, however, can be explained via figure 3, which is a graph of the arcsin function.

As can be seen from that graph, the slope near x=-1 gets very, very large, which means that any small deviations will have a very large impact. As my sin and cos functions were used to calculate arcsin,

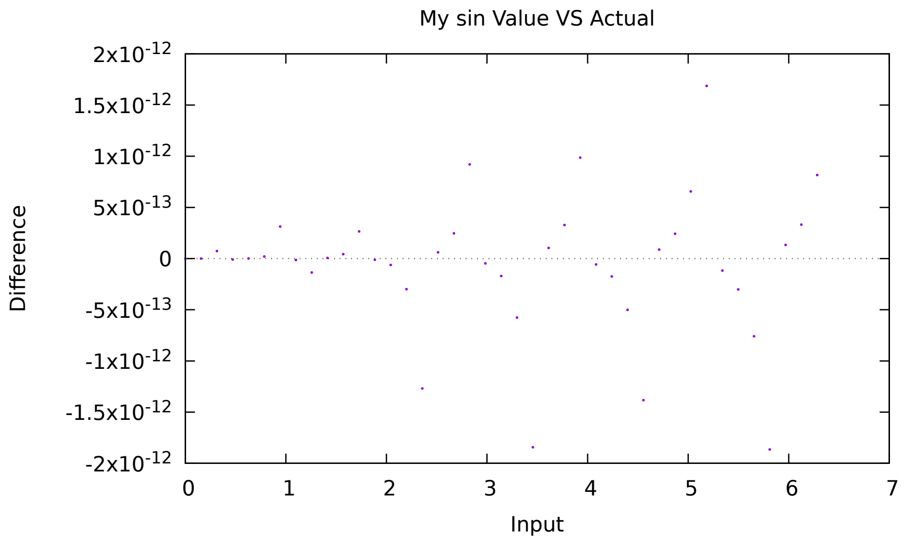

Figure 1: These are the sin differences. As the graph for cosine looks almost identical, it has been left out.

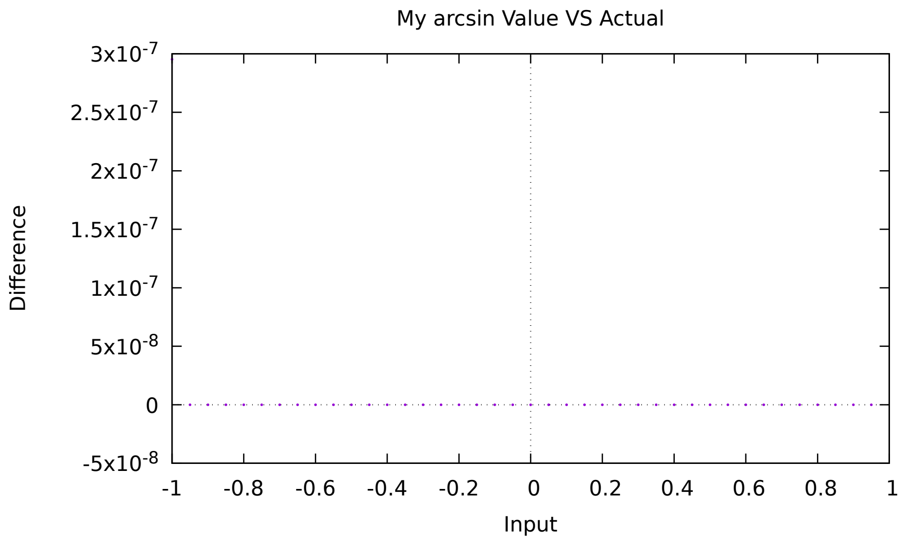

Figure 2: This is arcsin differences. The graph for arccos has a negative outlier, but aside from that it looks almost identical.

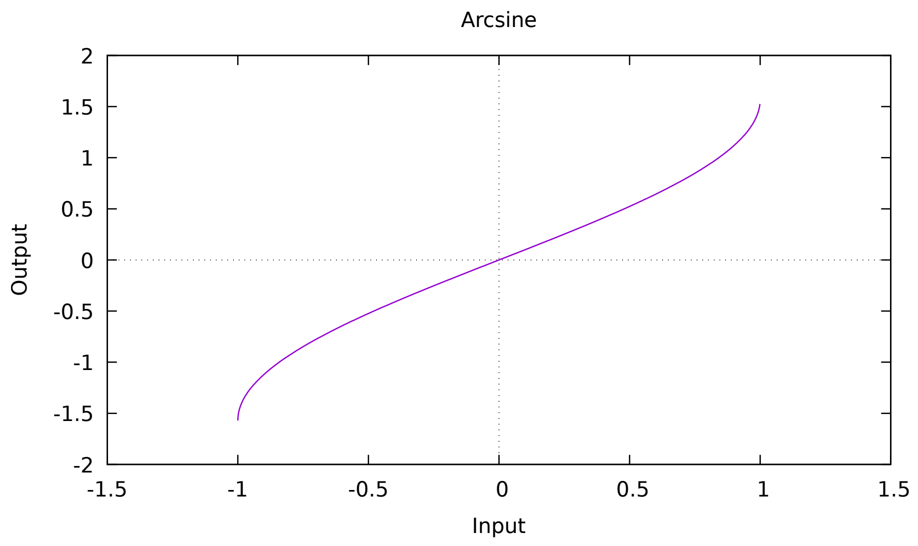

Figure 3: This is a graph of arcsin. Once again, arccos looks very similar.

the very small difference in those values could compound into very large errors. The library's function could be more accurate because its sin and cos are more accurate, reducing the importance of these compounding errors. Unlike with sin and cos, a Talor Series is not used for arcsin, so I can't change the center to make it more accurate. Additionally, a different minimum error would not help either, as due to the compounding errors, the error is already above what I set my minimum error to. As this is the case, decreasing the minimum error would make me more precisely wrong, which doesn't exactly help.

As for the rest of the data, excluding the outlier, visible in figure 4, once again we can decrease the minimum error to make the code take longer, but also be more accurate, and we could increase the accuracy of sin and cos with the methods described above, to reduce compounding errors.

For arccos, the implementation of it is based entirely on the implementation arcsin. As such, making my arcsin implementation closer to the libray's also making my arccos implementation closer, and there's not much else I can do.

4 Natural Logarithmic

Unfortunately, log is implemented differently than sin or cos, and does not use a Taylor series, so there can be no moving of centers. It does, however, depend on a professor provided exponential function, so the simplest way to improve the accuracy of my log function would be to improve this exponential function to reduce any errors carried over from it. Unlike the arcsin outlier, all of the data points are still under my minimum error, so decreasing the minimum error and allowing the function to run more times would also serve to increase the accuracy. These, however, are really all that can be done to improve it.

5 Arctangent

Looking at the graph of differences for arctan in figure 5, the most striking thing is its complete lack of any sort of pattern. Where all of the other graphs had some shape to them, this one has absolutely none.

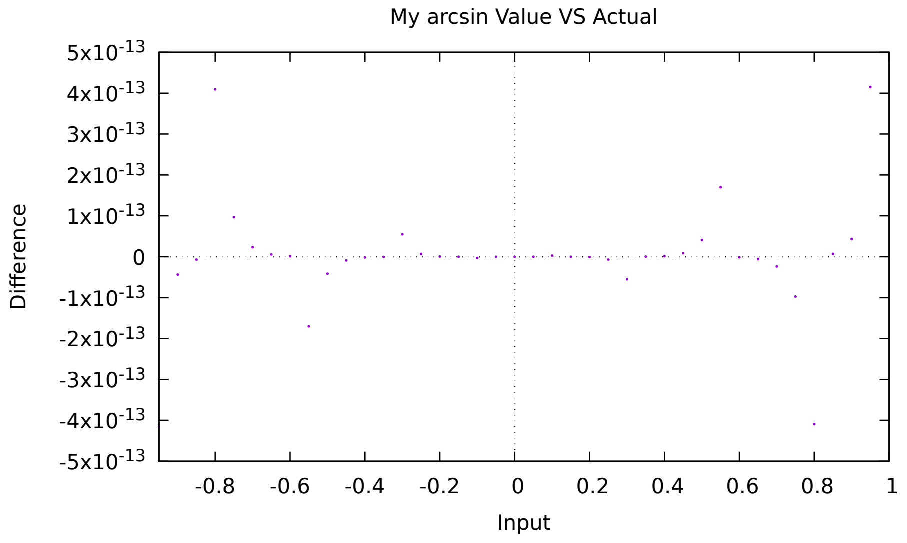

Figure 4: This is a graph of arcsin with the outlier removed. Once again, arccos looks very similar.

The error for it is also much higher on average than any other function (ignoring the arcsin and arccos outlier). Clearly, something is different about arctan. It is based entirely on three functions, arcsin, arccos, and the professor provided sqrt function. Outside of the outlier at -1, arcsin has a much lower error than arctan, and so does arccos, so while improving their accuracy would help, they are not the primary cause. However, looking at figure 5, which shows the difference between the library's implementation and the professor provided function, it can be seen that the error in arctan and sqrt are close in value and in pattern. Based on this, we can infer that sqrt is the main cause of the difference between the library's implementation and mine, so the best way to improve the accuracy of arctan would be to improve the accuracy of the sqrt function. (It should be noted that the inaccuracy of sqrt was pointed out by Mrbengly#5343 on discord (real name unknown to me, they are a student in the class). All conclusions made because of this information were mine alone)

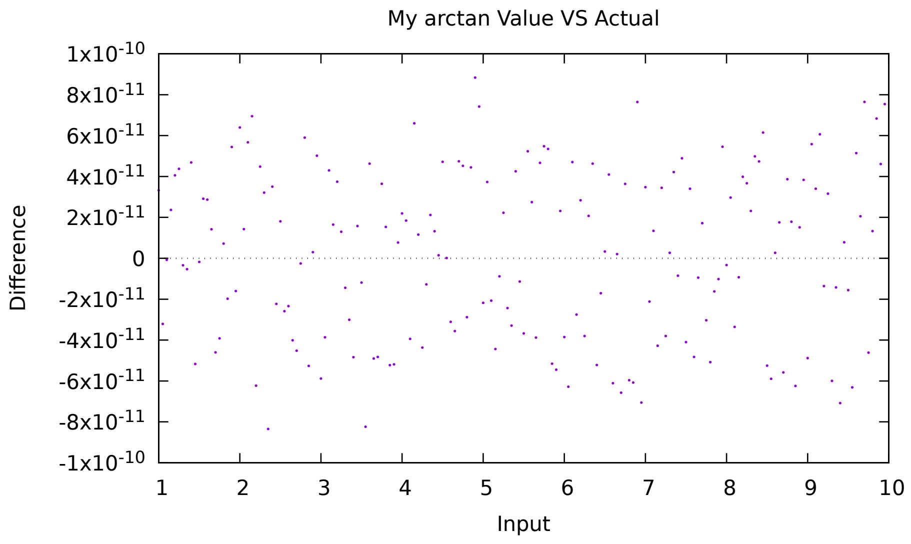

Figure 5: Very random differences from arctan.

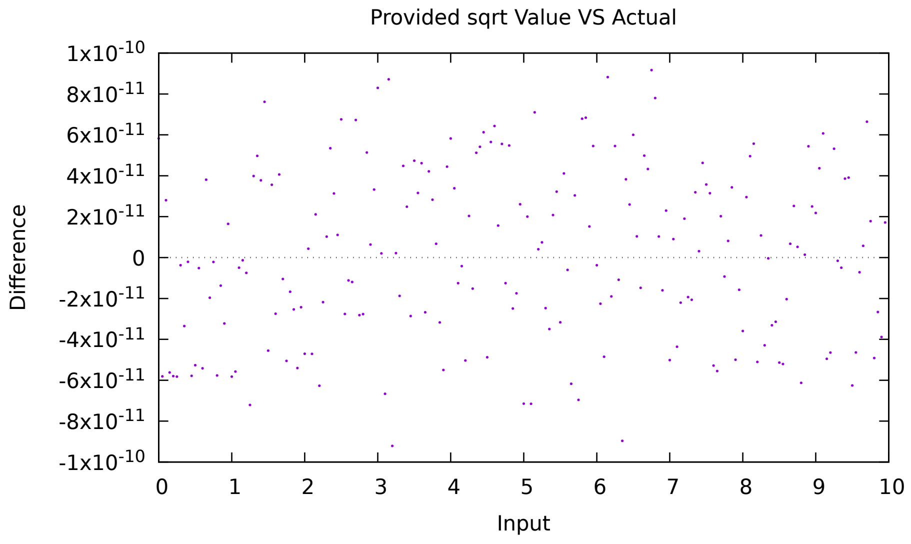

Figure 6: Looks awfully similar in shape to figure 4.