Areeb Amjad
afamjad@ucsc.edu
4/11/21

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

This writeup will compare four functions that were implemented in C (arcSin(x), arCos(x), arcTan(x), and Log(x)) to the library functions in the <math.h> library (asin(x), acos(x), atan(x), and log(x)). Differences in function outputs will be analyzed, and an attempt will be made to reason about these differences.

Images, such as terminal outputs and graphs, may be used to further demonstrate an approximation's effectiveness/ineffectiveness.

arcSin(x) and asin(x)

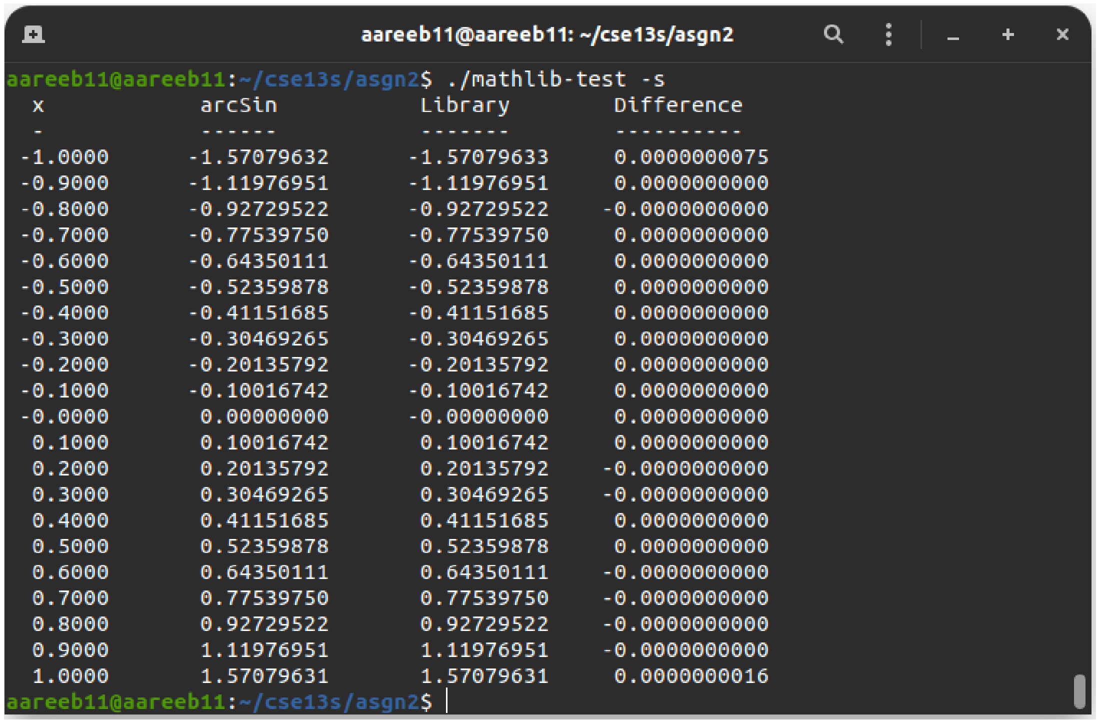
Initially, the arcSin function used the Taylor Series approximation. However, such an approximation is not nearly as accurate as compared to the Newton-Raphson method. More on why that is the case later.

The arcSin function uses the Newton iterate method to approximate $\sin^{-1}$. If we have a continuous and differentiable function and we want to calculate the root and we know it is near a point x_0, the Newtown-Raphson method says that the next best guess is defined by

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

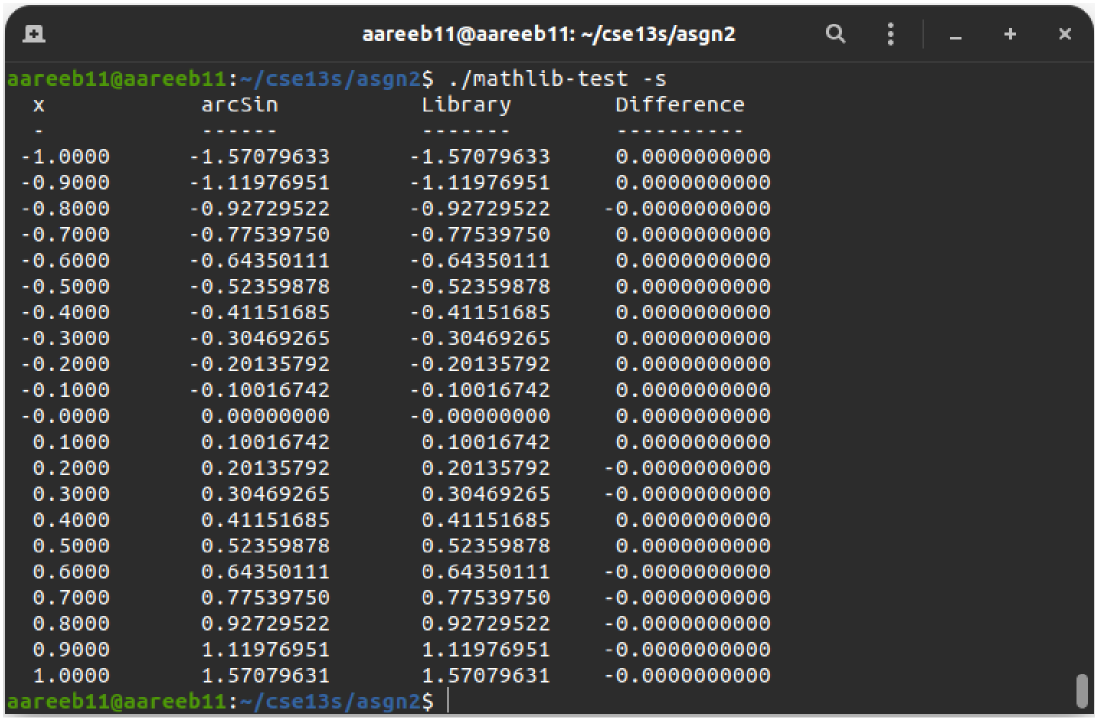
As we can see under "Difference" in Figure 1, this approximation overall is quite accurate. Notice, however, that there is a tiny difference at ±1. This slight inaccuracy is reduced in Figure 2 by using trigonometric identities specifically for -1 and 1:

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



```
aareeb11@aareeb11: ~/cse13s/asgn2
aareeb11@aareeb11:~/cse13s/asgn2$ ./mathlib-test -s
  x            arcSin           Library          Difference
  -            ------           -------          ----------
 -1.0000       -1.57079632      -1.57079633      0.0000000075
 -0.9000       -1.11976951      -1.11976951      0.0000000000
 -0.8000       -0.92729522      -0.92729522     -0.0000000000
 -0.7000       -0.77539750      -0.77539750      0.0000000000
 -0.6000       -0.64350111      -0.64350111      0.0000000000
 -0.5000       -0.52359878      -0.52359878      0.0000000000
 -0.4000       -0.41151685      -0.41151685      0.0000000000
 -0.3000       -0.30469265      -0.30469265      0.0000000000
 -0.2000       -0.20135792      -0.20135792      0.0000000000
 -0.1000       -0.10016742      -0.10016742      0.0000000000
 -0.0000        0.00000000      -0.00000000      0.0000000000
  0.1000        0.10016742       0.10016742      0.0000000000
  0.2000        0.20135792       0.20135792     -0.0000000000
  0.3000        0.30469265       0.30469265     -0.0000000000
  0.4000        0.41151685       0.41151685      0.0000000000
  0.5000        0.52359878       0.52359878      0.0000000000
  0.6000        0.64350111       0.64350111     -0.0000000000
  0.7000        0.77539750       0.77539750     -0.0000000000
  0.8000        0.92729522       0.92729522     -0.0000000000
  0.9000        1.11976951       1.11976951     -0.0000000000
  1.0000        1.57079631       1.57079631      0.0000000016
aareeb11@aareeb11:~/cse13s/asgn2$
```

Figure 1: Terminal output of arcSin and asin, before optimizing edge case accuracy.



```
aareeb11@aareeb11: ~/cse13s/asgn2
aareeb11@aareeb11:~/cse13s/asgn2$ ./mathlib-test -s
  x            arcSin           Library          Difference
  -            ------           -------          ----------
 -1.0000       -1.57079633      -1.57079633      0.0000000000
 -0.9000       -1.11976951      -1.11976951      0.0000000000
 -0.8000       -0.92729522      -0.92729522     -0.0000000000
 -0.7000       -0.77539750      -0.77539750      0.0000000000
 -0.6000       -0.64350111      -0.64350111      0.0000000000
 -0.5000       -0.52359878      -0.52359878      0.0000000000
 -0.4000       -0.41151685      -0.41151685      0.0000000000
 -0.3000       -0.30469265      -0.30469265      0.0000000000
 -0.2000       -0.20135792      -0.20135792      0.0000000000
 -0.1000       -0.10016742      -0.10016742      0.0000000000
 -0.0000        0.00000000      -0.00000000      0.0000000000
  0.1000        0.10016742       0.10016742      0.0000000000
  0.2000        0.20135792       0.20135792     -0.0000000000
  0.3000        0.30469265       0.30469265     -0.0000000000
  0.4000        0.41151685       0.41151685      0.0000000000
  0.5000        0.52359878       0.52359878      0.0000000000
  0.6000        0.64350111       0.64350111     -0.0000000000
  0.7000        0.77539750       0.77539750     -0.0000000000
  0.8000        0.92729522       0.92729522     -0.0000000000
  0.9000        1.11976951       1.11976951     -0.0000000000
  1.0000        1.57079631       1.57079631     -0.0000000000
aareeb11@aareeb11:~/cse13s/asgn2$
```

Figure 2: Terminal output of arcSin and asin, after optimizing edge case accuracy.

The reason why the Newton iterate/Newton-Raphson method may work so well is that each approximation gets closer and closer to the actual root value, and guessing continues until the current and previous approximations are close enough to each other. The initial guess is far off from the actual value, but with successive iterations, we are quickly able to move closer.

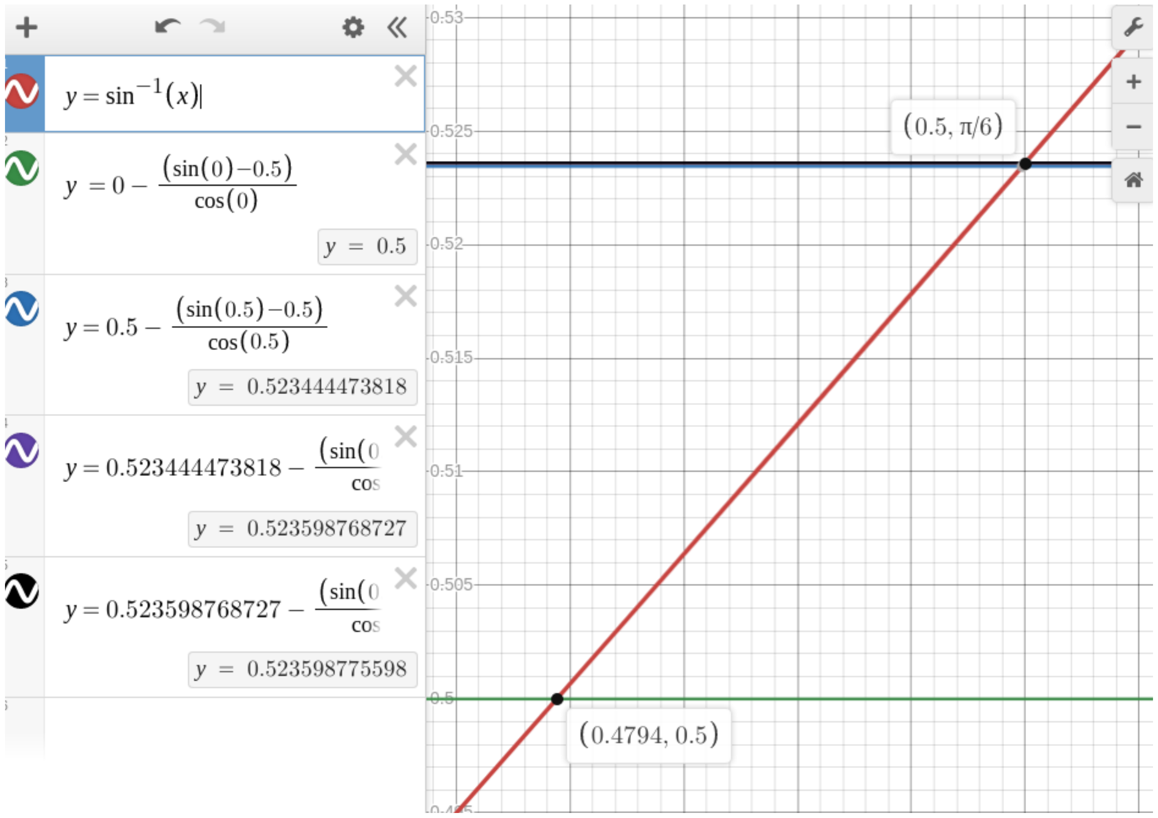
Say that we wanted to approximate $\sin^{-1}(0.5)$ using the Newton-Raphson method. Solving $y = \sin^{-1}(0.5)$ is the same as solving $\sin(y) - 0.5 = 0$ by applying sin on both sides. Now, plugging $\sin(y) - 0.5$ into Newton's method, where y is the guess: $y_{n+1} = y_n - (sin(y_n) - 0.5) / cos(y_n)$

Our initial guess will be $y_n = 0$. To find the next guess, we plug y_n and solve for the next guess: $y_{n+1} = 0 - (\sin(0) - 0.5) / \cos(0) = 0.5$.

We take this new value and plug it into the equation once again, to get ≈ 0.5234444738. Once more, and we get ≈ 0.5235987687. One final time and we get the value of $\sin^{-1}(0.5)$: 0.5235987756.

Now, if we did not know the value of $\sin^{-1}(0.5)$, we would stop when we see that the approximations are close enough. In this case, if we were to make one more guess using the value 0.5235987756, the result would be the same: 0.5235987756.

Figure 3 below displays the method graphically.



$y = \sin^{-1}(x)$

$y = 0 - \frac{(\sin(0) - 0.5)}{\cos(0)}$

$y = 0.5$

$y = 0.5 - \frac{(\sin(0.5) - 0.5)}{\cos(0.5)}$

$y = 0.523444473818$

$y = 0.523444473818 - \frac{(\sin(0}{\cos}$

$y = 0.523598768727$

$y = 0.523598768727 - \frac{(\sin(0}{\cos}$

$y = 0.523598775598$

Figure 3: Graphical representation of solving $\sin^{-1}(0.5)$ using Newton-Raphson's Method. The Taylor Series expansion, on the other hand, is not as accurate. The approximation may result in a comparatively imprecise output at x = ±1 because it converges very slowly for those values. In other words, the number of terms needed to approximate arcsin(±1) is far greater than the terms needed to approximate arcsin near 0. As we know, floating-point numbers are not real numbers and have finite precision, so they may not approximate a large number of terms very accurately.

Taking a look at Figure 4, we see that the slope of arcsin (i.e., its derivative) grows toward infinity. While convergence is possible at -1 and 1, we would need to calculate too many terms to make the approximation at these values with the Taylor Series useful.

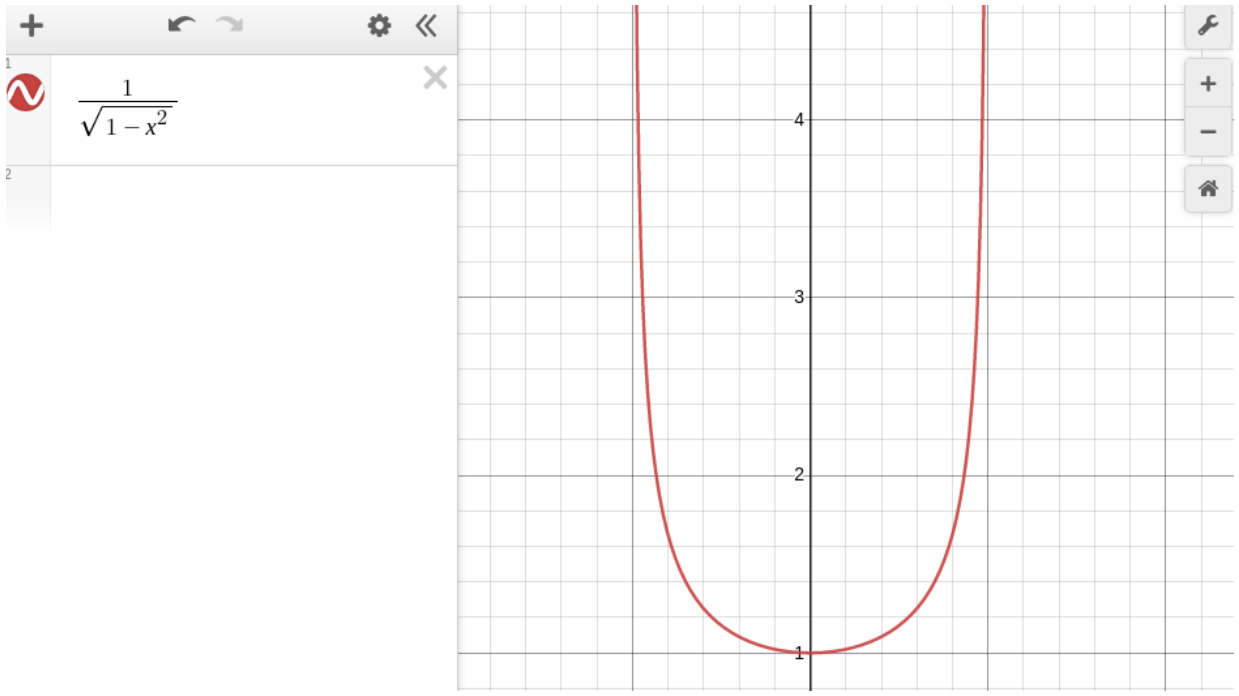


Figure 4: Plot of the derivative of $\sin^{-1}(x)$

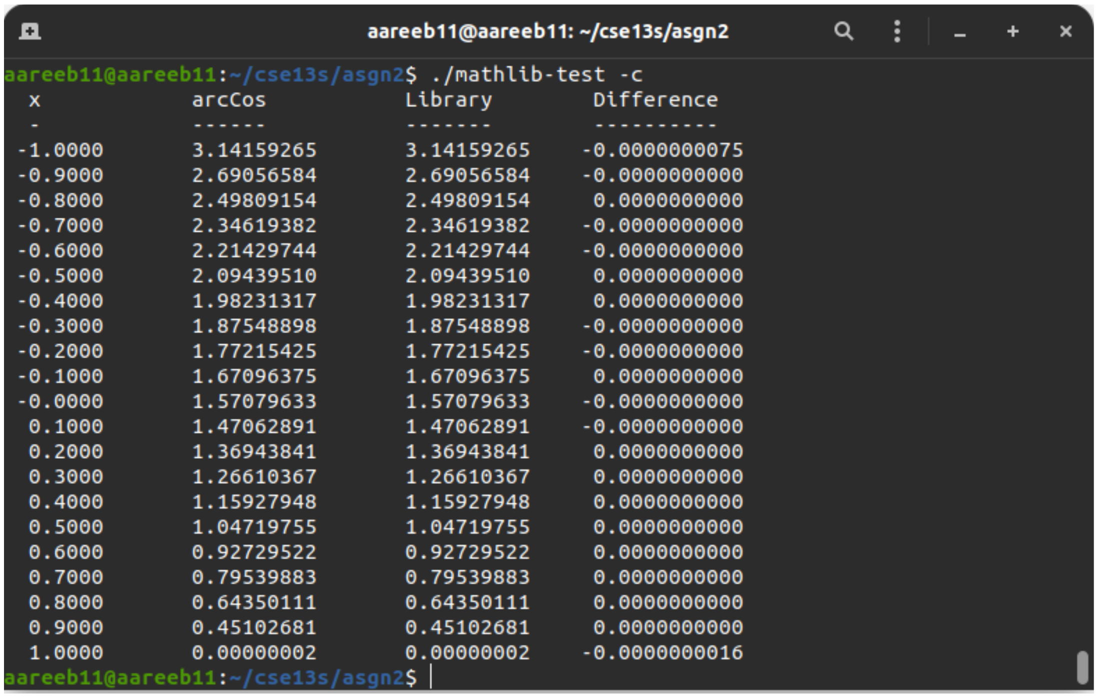
While the rate of convergence for Newton's method is faster than the Taylor Series, arcsin(±1) still converges slowly using Newton's method and produces slightly inaccurate results. This may be due to the reason mentioned above, but Newton's method only deals with the Taylor series's first-order term, while Taylor Series expansion uses many terms, and this is why the error for Newton's Method may be slight while in the Taylor Series it is much larger.

arcCos(x) and acos(x)

The arcCos function uses arcSin in its implementation; to be more precise, its regular implementation is $\pi / 2 - \texttt{arcSin(x)}$. As such, the results under “Difference” in Figures 1 and 5 are the same in magnitude; however, their signs are flipped. Given this, we can expect an identical behavior in the differences; i.e. just like arcsin, we notice that the difference between arcCos and acos is slight at ±1. Once again, trigonometric identities were used to reduce the inaccuracy at ±1:

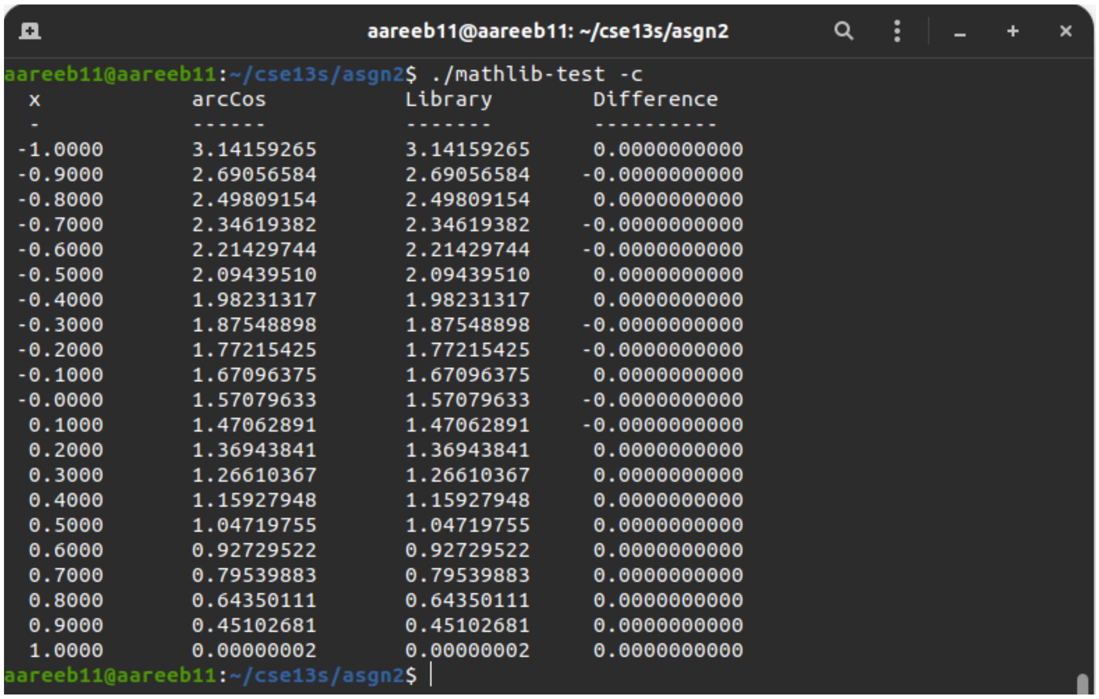
If x is 1, $cos^{-1}(x) = sin^{-1}(\sqrt{1 - x^2})$.

If x is -1, $cos^{-1}(x) = \pi - sin^{-1}(\sqrt{1 - x^2})$.



```
aareeb11@aareeb11:~/cse13s/asgn2$ ./mathlib-test -c
  x             arcCos          Library         Difference
  -             ------          -------         ----------
 -1.0000        3.14159265      3.14159265     -0.0000000075
 -0.9000        2.69056584      2.69056584     -0.0000000000
 -0.8000        2.49809154      2.49809154      0.0000000000
 -0.7000        2.34619382      2.34619382     -0.0000000000
 -0.6000        2.21429744      2.21429744     -0.0000000000
 -0.5000        2.09439510      2.09439510      0.0000000000
 -0.4000        1.98231317      1.98231317      0.0000000000
 -0.3000        1.87548898      1.87548898     -0.0000000000
 -0.2000        1.77215425      1.77215425     -0.0000000000
 -0.1000        1.67096375      1.67096375      0.0000000000
 -0.0000        1.57079633      1.57079633     -0.0000000000
  0.1000        1.47062891      1.47062891     -0.0000000000
  0.2000        1.36943841      1.36943841      0.0000000000
  0.3000        1.26610367      1.26610367      0.0000000000
  0.4000        1.15927948      1.15927948      0.0000000000
  0.5000        1.04719755      1.04719755      0.0000000000
  0.6000        0.92729522      0.92729522      0.0000000000
  0.7000        0.79539883      0.79539883      0.0000000000
  0.8000        0.64350111      0.64350111      0.0000000000
  0.9000        0.45102681      0.45102681      0.0000000000
  1.0000        0.00000002      0.00000002     -0.0000000016
aareeb11@aareeb11:~/cse13s/asgn2$
```

Figure 5: Terminal output of arcCos and acos, before optimizing edge case accuracy.



```
aareeb11@aareeb11:~/cse13s/asgn2$ ./mathlib-test -c
  x              arcCos           Library          Difference
  -              ------           -------          ----------
 -1.0000        3.14159265       3.14159265       0.0000000000
 -0.9000        2.69056584       2.69056584      -0.0000000000
 -0.8000        2.49809154       2.49809154       0.0000000000
 -0.7000        2.34619382       2.34619382      -0.0000000000
 -0.6000        2.21429744       2.21429744      -0.0000000000
 -0.5000        2.09439510       2.09439510       0.0000000000
 -0.4000        1.98231317       1.98231317       0.0000000000
 -0.3000        1.87548898       1.87548898      -0.0000000000
 -0.2000        1.77215425       1.77215425      -0.0000000000
 -0.1000        1.67096375       1.67096375       0.0000000000
 -0.0000        1.57079633       1.57079633      -0.0000000000
  0.1000        1.47062891       1.47062891      -0.0000000000
  0.2000        1.36943841       1.36943841       0.0000000000
  0.3000        1.26610367       1.26610367       0.0000000000
  0.4000        1.15927948       1.15927948       0.0000000000
  0.5000        1.04719755       1.04719755       0.0000000000
  0.6000        0.92729522       0.92729522       0.0000000000
  0.7000        0.79539883       0.79539883       0.0000000000
  0.8000        0.64350111       0.64350111       0.0000000000
  0.9000        0.45102681       0.45102681       0.0000000000
  1.0000        0.00000002       0.00000002       0.0000000000
aareeb11@aareeb11:~/cse13s/asgn2$
```

Figure 6: Terminal output of arcCos and acos, after optimizing edge case accuracy.

It looks like Newton's method approximates $\cos^{-1}(x)$ just as well as it does for $\sin^{-1}(x)$. And like arcsin, the reason why the Taylor series approximation may be resulting in a comparatively imprecise output at $x = \pm1$ is that the rate of convergence is very slow at $x = \pm1$ compared to the rate of convergence when $x = 0$ or when x is near 0.

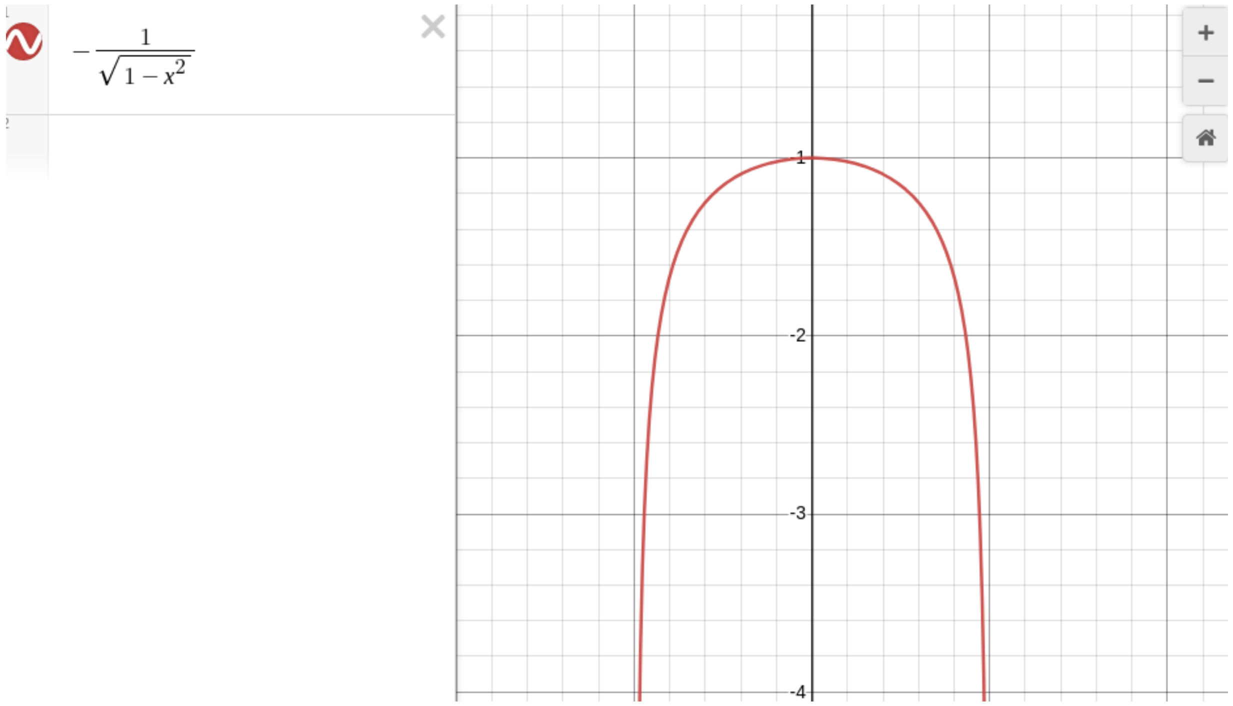
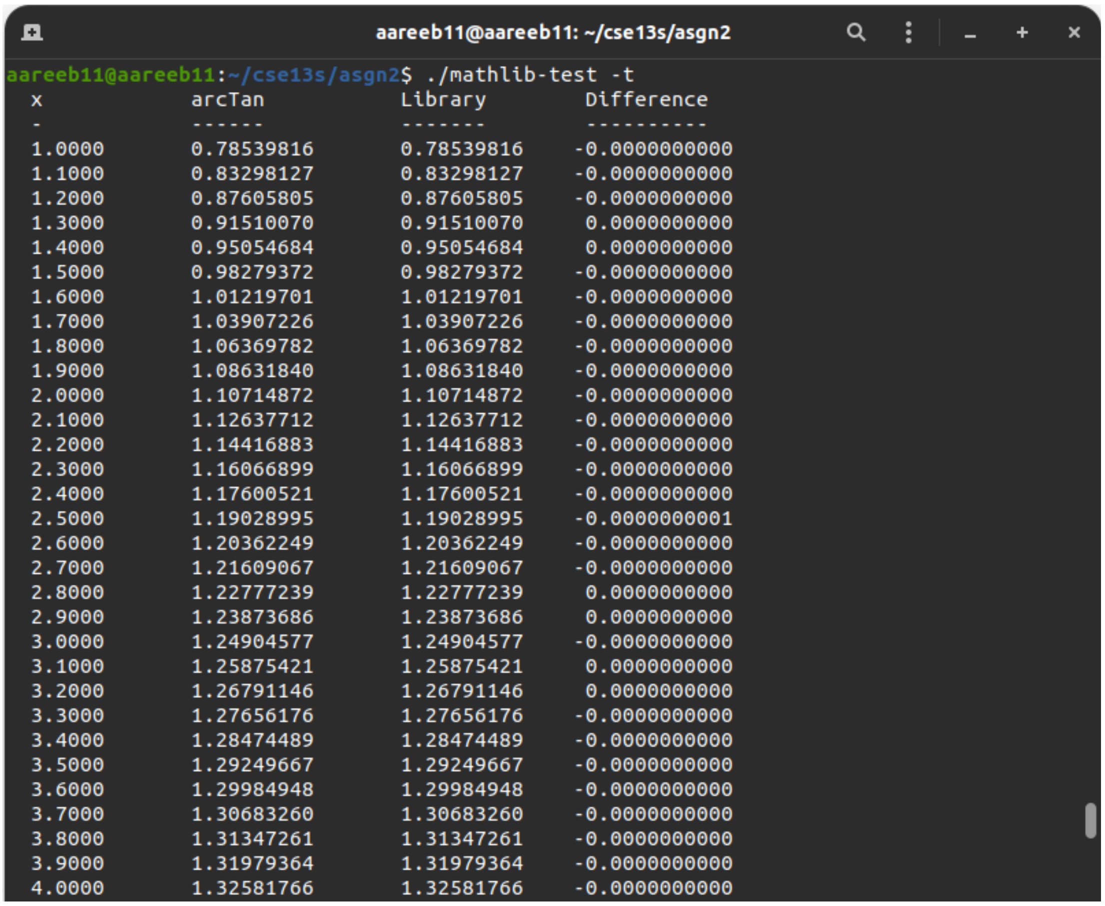


Figure 7: Graph of the derivative of $\cos^{-1}(x)$.

Looking at Figure 7, the derivative of $\cos^{-1}(x)$ at -1 and 1 grows towards negative infinity, and so the same explanation applies: we would need to calculate too many terms to make the approximation at these values accurate. Newton's method only deals with the Taylor series's first-order term in comparison.

arcTan(x) and atan(x)

arcTan's implementation also uses arcSin: `arctan = arcSin(x /` $\sqrt{x^2 + 1}$`)`. Thanks to Newton's method, the differences between functions are either $1 * 10^{-10}$ or 0. The small difference of $1 * 10^{-10}$ could be due to floating-point error in the process of computing `arcSin(x /` $\sqrt{x^2 + 1}$`)`.



```
aareeb11@aareeb11:~/cse13s/asgn2$ ./mathlib-test -t
  x             arcTan            Library           Difference
  -             ------            -------           ----------
  1.0000        0.78539816        0.78539816       -0.0000000000
  1.1000        0.83298127        0.83298127       -0.0000000000
  1.2000        0.87605805        0.87605805       -0.0000000000
  1.3000        0.91510070        0.91510070        0.0000000000
  1.4000        0.95054684        0.95054684        0.0000000000
  1.5000        0.98279372        0.98279372       -0.0000000000
  1.6000        1.01219701        1.01219701       -0.0000000000
  1.7000        1.03907226        1.03907226       -0.0000000000
  1.8000        1.06369782        1.06369782       -0.0000000000
  1.9000        1.08631840        1.08631840       -0.0000000000
  2.0000        1.10714872        1.10714872       -0.0000000000
  2.1000        1.12637712        1.12637712       -0.0000000000
  2.2000        1.14416883        1.14416883       -0.0000000000
  2.3000        1.16066899        1.16066899       -0.0000000000
  2.4000        1.17600521        1.17600521       -0.0000000000
  2.5000        1.19028995        1.19028995       -0.0000000001
  2.6000        1.20362249        1.20362249       -0.0000000000
  2.7000        1.21609067        1.21609067       -0.0000000000
  2.8000        1.22777239        1.22777239        0.0000000000
  2.9000        1.23873686        1.23873686        0.0000000000
  3.0000        1.24904577        1.24904577       -0.0000000000
  3.1000        1.25875421        1.25875421        0.0000000000
  3.2000        1.26791146        1.26791146        0.0000000000
  3.3000        1.27656176        1.27656176       -0.0000000000
  3.4000        1.28474489        1.28474489       -0.0000000000
  3.5000        1.29249667        1.29249667       -0.0000000000
  3.6000        1.29984948        1.29984948       -0.0000000000
  3.7000        1.30683260        1.30683260       -0.0000000000
  3.8000        1.31347261        1.31347261       -0.0000000000
  3.9000        1.31979364        1.31979364       -0.0000000000
  4.0000        1.32581766        1.32581766       -0.0000000000
```

Figure 8: Terminal output of arcTan and atan, cut off at x = 4.

As mentioned earlier, arcTan relies on arcSin for its implementation, and arcSin originally used the Taylor Series. This resulted in an increasing difference between arcTan and the library's atan (shown in Figure 9). It appears that starting from x = 1, arcTan starts to converge very slowly.

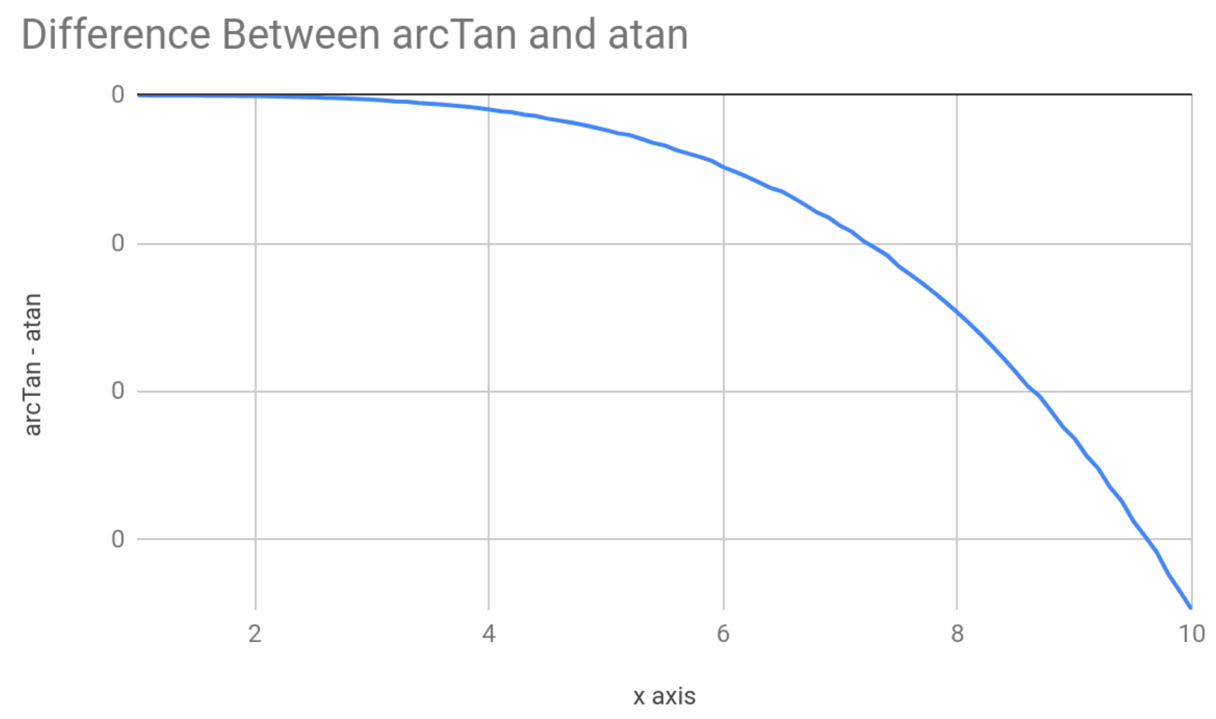


Figure 9: Plot of arcTan - atan against x, Taylor Series implementation

If we look at the Taylor Series for arctan and e^x:

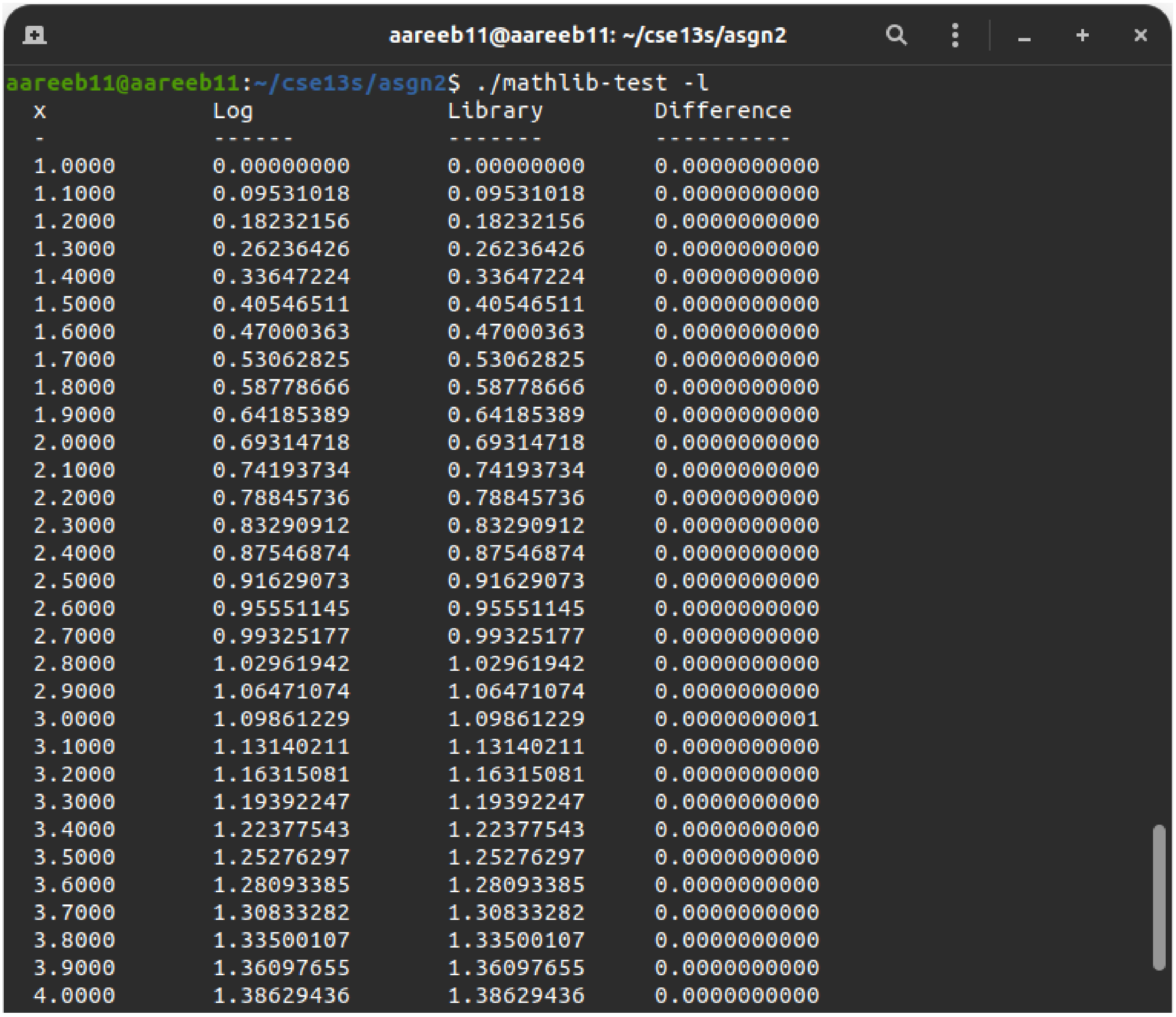
$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} x^{2k+1}$$

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

We notice that, unlike e^x, the denominators of terms are growing at a slower rate. Therefore, the terms of the arctan series decrease much more slowly, and the rate of convergence is very poor.

Log(x) and log(x)

The Log (natural log) implementation also uses the Newton-Raphson method to approximate. As we can see under “Difference” in Figure 10, the differences between Log and log are either 0 or 1 x 10^{-10}, just like arcTan. There is practically no difference between the functions. Figure 11 shows the same thing: when there are differences in the function, they are very small. These minor differences could again be due to floating-point error in calculating the exponential function.



```
aareeb11@aareeb11:~/cse13s/asgn2$ ./mathlib-test -l
  x            Log              Library          Difference
  -            ------           -------          ----------
  1.0000       0.00000000       0.00000000       0.0000000000
  1.1000       0.09531018       0.09531018       0.0000000000
  1.2000       0.18232156       0.18232156       0.0000000000
  1.3000       0.26236426       0.26236426       0.0000000000
  1.4000       0.33647224       0.33647224       0.0000000000
  1.5000       0.40546511       0.40546511       0.0000000000
  1.6000       0.47000363       0.47000363       0.0000000000
  1.7000       0.53062825       0.53062825       0.0000000000
  1.8000       0.58778666       0.58778666       0.0000000000
  1.9000       0.64185389       0.64185389       0.0000000000
  2.0000       0.69314718       0.69314718       0.0000000000
  2.1000       0.74193734       0.74193734       0.0000000000
  2.2000       0.78845736       0.78845736       0.0000000000
  2.3000       0.83290912       0.83290912       0.0000000000
  2.4000       0.87546874       0.87546874       0.0000000000
  2.5000       0.91629073       0.91629073       0.0000000000
  2.6000       0.95551145       0.95551145       0.0000000000
  2.7000       0.99325177       0.99325177       0.0000000000
  2.8000       1.02961942       1.02961942       0.0000000000
  2.9000       1.06471074       1.06471074       0.0000000000
  3.0000       1.09861229       1.09861229       0.0000000001
  3.1000       1.13140211       1.13140211       0.0000000000
  3.2000       1.16315081       1.16315081       0.0000000000
  3.3000       1.19392247       1.19392247       0.0000000000
  3.4000       1.22377543       1.22377543       0.0000000000
  3.5000       1.25276297       1.25276297       0.0000000000
  3.6000       1.28093385       1.28093385       0.0000000000
  3.7000       1.30833282       1.30833282       0.0000000000
  3.8000       1.33500107       1.33500107       0.0000000000
  3.9000       1.36097655       1.36097655       0.0000000000
  4.0000       1.38629436       1.38629436       0.0000000000
```

Figure 10: Terminal output of Log and log, cut off at x = 4.

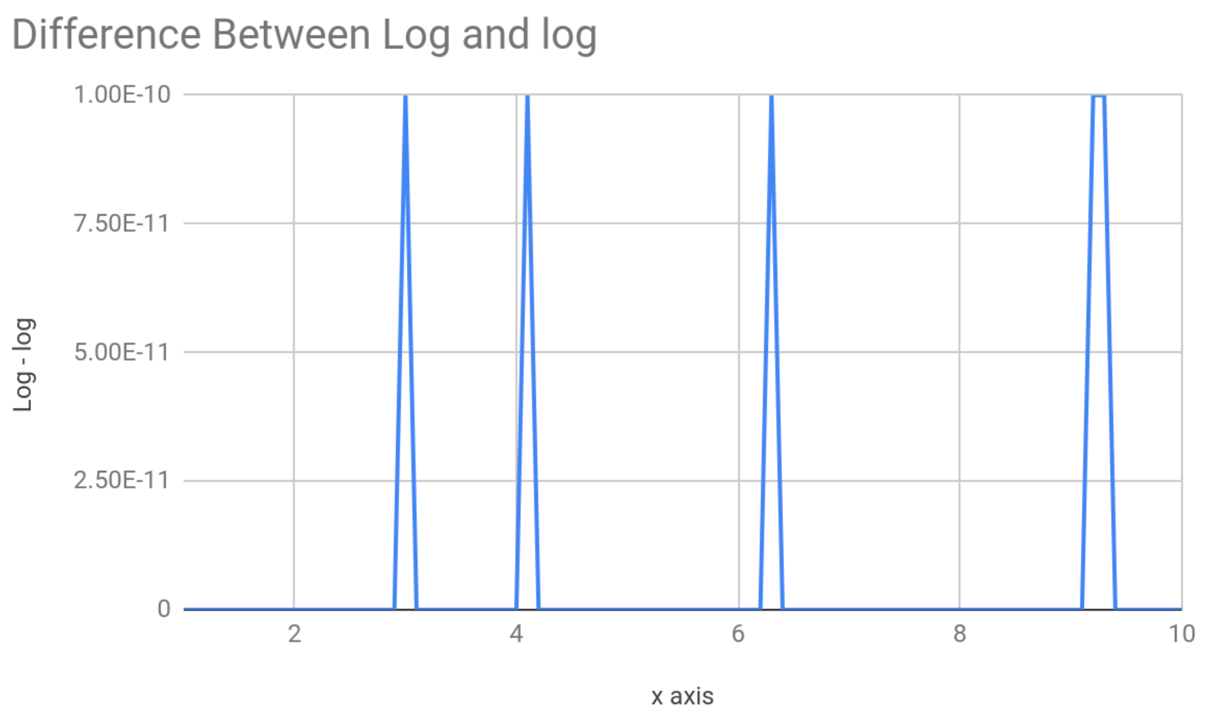
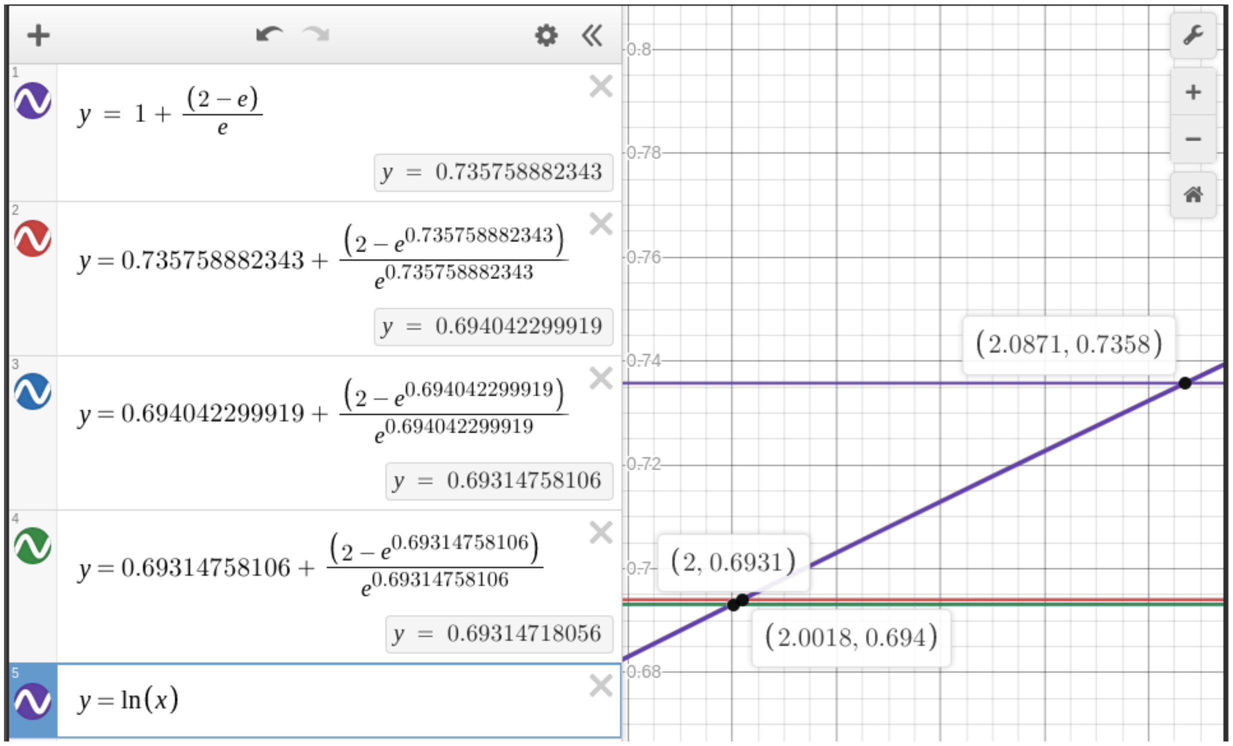


Figure 11: Plot of Log - log against x.

Figure 12 below displays Newton's method for ln(2) graphically.



$$y = 1 + \frac{(2-e)}{e}$$

$y = 0.735758882343$

$$y = 0.735758882343 + \frac{\left(2 - e^{0.735758882343}\right)}{e^{0.735758882343}}$$

$y = 0.694042299919$

$$y = 0.694042299919 + \frac{\left(2 - e^{0.694042299919}\right)}{e^{0.694042299919}}$$

$y = 0.69314758106$

$$y = 0.69314758106 + \frac{\left(2 - e^{0.69314758106}\right)}{e^{0.69314758106}}$$

$y = 0.69314718056$

$$y = \ln(x)$$

Figure 12: Graphical representation of solving ln(2) using Newton-Raphson's Method.

As we can see, the initial guess is 1, and the successive guess is rather far off from the actual value of ln(2). However, we get much closer to ln(2) with the guess after that, represented by the red line. We get even closer with the guess represented by the blue line (which is practically overlapping the green line), and we finally obtain the value of ln(2) with the green line. Newton-Raphson's method appears to be a very precise method for finding the roots of functions, especially non-polynomial ones.

Conclusion

I now understand that arcsin(x) could have been implemented using the Taylor Series centered about 0 or using the Newton-Raphson method. I also understand that the Newton-Raphson method generally converges much faster than the Taylor Series, which is why its preferred, but it has its limitations; we cannot use the method if our initial guess is at a point where the derivative is 0, or if there are no roots.