Derick Pan
dpan7

WRITEUP.pdf
Asgn2
A small Numerical Library

arcsin(x) Difference:

x	arcSin	Library	Difference
-1.0000	-1.56993607	-1.57079633	0.0008602531
-0.9000	-1.11976951	-1.11976951	0.0000000033
-0.8000	-0.92729522	-0.92729522	0.0000000012
-0.7000	-0.77539750	-0.77539750	0.0000000005
-0.6000	-0.64350111	-0.64350111	0.0000000003
-0.5000	-0.52359878	-0.52359878	0.0000000002
-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	-0.0000000002
0.6000	0.64350111	0.64350111	-0.0000000003
0.7000	0.77539750	0.77539750	-0.0000000005
0.8000	0.92729522	0.92729522	-0.0000000012
0.9000	1.11976951	1.11976951	-0.0000000033
1.0000	1.56993607	1.57079631	-0.0008602320

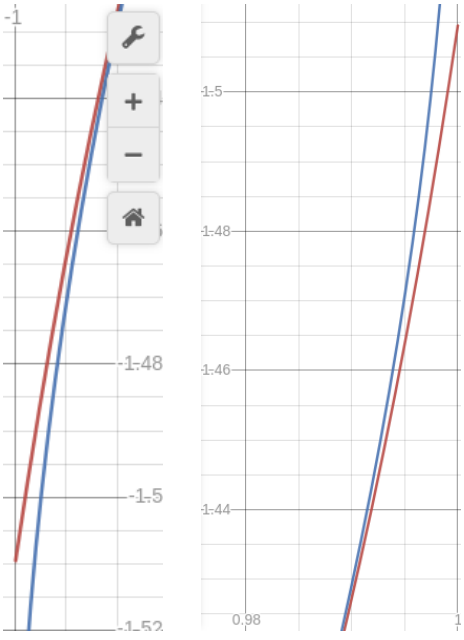
My arcsin function differs from the Library's arcSin function in two very noticeable ways.

1. The values closer to the center "0", follow a greater degree of accuracy, upwards but not limited to 10 decimal places of accuracy.
2. My implementation of the Maclaurin series for arcsin(x)

My implementation of arcsin(x) calculates values between -.4 to .4 to an accuracy of 10 decimal places, but values getting closer to the range lose more and more accuracy. The reason for this inaccuracy being the required amount of iterations needed to calculate the smallest and largest numbers in the range. For example, calculating arcsin(-1) using the Maclaurin series with an accuracy of 10 decimal places requires more than 1,000 iterations. The Maclaurin series used for arcsin is as follows:

$$\arcsin(x) = \sum_{k=0}^{\infty} \frac{(2k)!}{2^{2k}(k!)^2} \frac{x^{2k+1}}{2k+1}, \quad |x| \le 1.$$

After every iteration my implementation calculates the factorial for two times the iteration number, and the factorial for the iteration number squared. As an idea, during the 100th iteration, the factorial of the iteration # is 326 digits long.
If only 82 iterations were done for my code's implementation of the maclaurin series, my code answers would differ by several decimal points of precision.
On the graphs to the right, the red line represents the answers the answers to my arcSin(x) with only 82 iterations, and the blue line represents the already established arcsin(x). The two graphs represent the two tail ends of the equations.



Overall, for my implementation of arcsin(x) to achieve a greater rate of accuracy, I require a significant amount of iterations. As shown above, the original maclaurin series for arcsin(x) could compute to infinity, and mine's doesn't, so as a proof of concept, my implementation works.

Derick Pan
dpan7

arcCos(x) Difference:

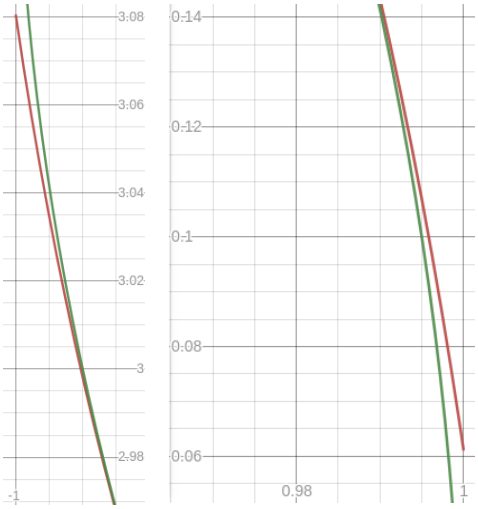
x	arcCos	Library	Difference
-1.0000	3.14073240	3.14159265	-0.0008602531
-0.9000	2.69056584	2.69056584	-0.0000000033
-0.8000	2.49809154	2.49809154	-0.0000000012
-0.7000	2.34619382	2.34619382	-0.0000000005
-0.6000	2.21429744	2.21429744	-0.0000000003
-0.5000	2.09439510	2.09439510	-0.0000000002
-0.4000	1.98231317	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	0.0000000002
0.6000	0.92729522	0.92729522	0.0000000003
0.7000	0.79539883	0.79539883	0.0000000005
0.8000	0.64350111	0.64350111	0.0000000012
0.9000	0.45102682	0.45102681	0.0000000033
1.0000	0.00086025	0.00000002	0.0008602320

My arcCos function differs from the Library's arcCos function in two very noticeable ways.

3. The values closer to the center "0", follow a greater degree of accuracy, upwards but not limited to 10 decimal places of accuracy.
4. My implementation of the Maclaurin series for arcCos(x).

My implementation of arcCos abuses the arcSin function. The only difference between the Maclaurin series for arcCos and arcSin is that: (π/2 -arcSin = arcCos).
So instead of creating another series with the only difference being π/2, my arcCos function calls arcSin and subtracts π/2 from the answer.

On the graphs to the right, the red line represents the answers the answers to my arcCos(x) by subtracting π/2 from my arcSin(x), and the blue line represents the already established arccos(x). The two graphs represent the two tail ends of the equations.



Similar to arcSin(x), in order to achieve a greater degree of accuracy, my maclaurin series needs to do more iterations than currently done.

Derick Pan
dpan7

arcTan(x) Difference:

x	arcTan	Library	Difference
1.0000	0.78539816	0.78539816	-0.0000000008
1.1000	0.83298127	0.83298127	-0.0000000010
1.2000	0.87605805	0.87605805	-0.0000000013
1.3000	0.91510070	0.91510070	-0.0000000011
1.4000	0.95054684	0.95054684	-0.0000000016
1.5000	0.98279372	0.98279372	-0.0000000017
1.6000	1.01219701	1.01219701	-0.0000000019
1.7000	1.03907226	1.03907226	-0.0000000023
1.8000	1.06369782	1.06369782	-0.0000000029
1.9000	1.08631839	1.08631840	-0.0000000028
2.0000	1.10714871	1.10714872	-0.0000000029
9.0000	1.46013904	1.46013911	-0.0000000700
9.1000	1.46134531	1.46134538	-0.0000000713
9.2000	1.46252566	1.46252574	-0.0000000728
9.3000	1.46368093	1.46368100	-0.0000000744
9.4000	1.46481189	1.46481197	-0.0000000762
9.5000	1.46591931	1.46591939	-0.0000000780
9.6000	1.46700391	1.46700399	-0.0000000800
9.7000	1.46806638	1.46806646	-0.0000000811
9.8000	1.46910739	1.46910748	-0.0000000834
9.9000	1.47012759	1.47012767	-0.0000000847
10.0000	1.47112759	1.47112767	-0.0000000862

The values of x go to 10 with each x being +0.1 from the previous. Values not shown are to save space.

My arcTan function differs from the Library's arcTan function.

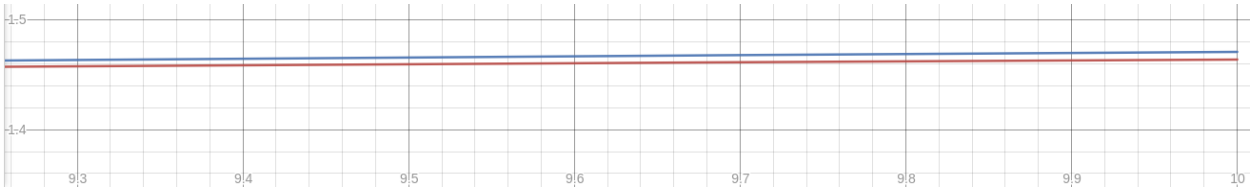
1. I only achieve an accuracy of 7-8 decimal places.
5. My implementation of the Maclaurin series for arcTan(x).

The possible implementations I considered for arcTan(x) is as follows:

$$\arctan(x)=\sum_{k=0}^{\infty}\frac{2^{2k}(k!)^2}{(2k+1)!}\frac{x^{2k+1}}{(1+x^2)^{k+1}}. \qquad \arctan(x)=\arcsin\left(\frac{x}{\sqrt{x^2+1}}\right)=\arccos\left(\frac{1}{\sqrt{x^2+1}}\right),\quad x>0.$$

I chose to compute arcTan(x) by computing arcSin($\frac{x}{\sqrt{x^2+1}}$). The reason I choose this method rather than the Maclaurin series is because I already implemented a Maclaurin series for arcSin, and I could use arcSin to derive the answer for arcTan.
The differences between my answers and the Library answers are due to the way I solve the square root for ($\frac{x}{\sqrt{x^2+1}}$). To find square roots, I used Newton's method with a precision of 10 decimal places. So that lessens my degree of accuracy when computing answers. Also,
The x values closer to 0 have a slightly higher degree of accuracy than the values closer to 10 because of the lack of iterations and the computing power necessary to compute iterations higher than 100. (values closer to 0's factorial aren't as large as 10.)

The graph shown below shows the right tail end of my arcTan(x) implementation and the widely used arcTan(x) function.



Derick Pan
dpan7

Log(x) Difference:

x	Log	Library	Difference
1.0000	0.00000000	0.00000000	0.000000000000000
1.1000	0.09531018	0.09531018	0.000000000000155
1.2000	0.18232156	0.18232156	0.000000000000521
1.3000	0.26236426	0.26236426	0.000000000012822
1.4000	0.33647224	0.33647224	0.000000000003776
1.5000	0.40546511	0.40546511	0.000000000022906
1.6000	0.47000363	0.47000363	0.000000000004028
1.7000	0.53062825	0.53062825	0.000000000014477
1.8000	0.58778666	0.58778666	0.000000000002066
1.9000	0.64185389	0.64185389	0.000000000005650
2.0000	0.69314718	0.69314718	0.000000000013559

The values of x go to 10 with each x being +0.1 from the previous. Values not shown are to save space.

My Log() function differs from the Library's Log() unction.

2. I only achieve an accuracy of 10 decimal places.
6. My implementation of Log()

My implementation of log computes ln(x) since e^x is the inverse of ln(x). I solve $ln(e^y) = x$. This gives me the equation:

$$x_{k+1} = x_k + \frac{y - e^{x_k}}{e^{x_k}}.$$

This formula is similar to how I solved square roots for arcTan(x), by using Newton's Method. To use this equation, I need to solve powers with base "e", and that was done using the exp() function code provided by Professor Long. This function's degree of accuracy is only up to the points of epsilon, which is 10^{-10} in this case. And, similar to the previous functions, my Log() function only stops when I deem the points of accuracy to be insignificant, which is 10 decimal points. Therefore, this function's accuracy is only up to 10 decimal places.

Overall, my Log() is very accurate with a degree of 10 digits of accuracy. Lowering the epsilon by a few more decimal points would raise the accuracy of this function.