William Santosa
wsantosa@ucsc.edu
24 January 2021

CSE13S Winter 2021
Assignment 2 : A Small Numerical Library
WriteUp Document

Description

This WriteUp document attempts to explain a few different things about my implemented functions and the <math.h> functions. Namely:

1) To analyze the differences in output of my program vs. output from <math.h>
2) Provide reasoning for differences in outputs
3) Use graphs to support my arguments

In addition to the above, I will also describe:

1) Problems I ran into
2) Solutions to the problems

Problem(s)

There weren't too many problems with implementing the prototype functions within mathlib.c as most of the code was either given to us in full or in pseudocode. As such, the main problem came when trying to represent the data and the outputs in a comprehensible way. When the floats are printed in the mathlib-test.c executable, the differences are too small to be represented as they don't have enough decimal points to be represented accurately (Figure 1 - 5). Plus, the steps were too large to get an accurate gauge of what is happening (Figure 6 / 7). Additionally, I also ran into a problem when trying to format the function outputs such that it was able to be read by a graphing utility.

Solution(s)

The first problem, making the differences easier to see, was an easy fix. I just had to change the precision of the floats so that it was longer. Plus, I had to change the step to be smaller, otherwise the graph would have too little data points for it to be accurate. The next problem, representing the data using graphs, took a lot longer than I expected it to. I had to figure out what to use to graph the data and I decided gnuplot would be easier to use than google spreadsheets. As such, I had to get the data into a readable format for gnuplot. After a few quick google searches and trial and error to figure out how to write text into a file, it worked (Figure 13). I just had to use fprintf to write into a file (Yes I know it says log, I just wanted a message to send so that I could tell if it works up until that point). After that, plotting was pretty simple. I just had to learn gnuplot's syntax and export it as a png to post on this pdf file.

Differences & Reasoning in Program vs. <math.h>

Sin/Cos/Tan

The difference graph for Sin, Cos, and Tan (Figure 8, 9, 10) kept increasing in terms of value as x went further away from 0, indicating that the accuracy of my implemented function is decreasing. This is most likely due to the fact that my epsilon (Defined as 1e-14) is larger than <math.h> library's epsilon and subsequently has less iterations for the taylor series, which, as mentioned earlier, would have less accuracy relative to the <math.h> Sin, Cos, and Tan functions. The graph itself has both positive and negative differences because my implementation switched between adding and subtracting like the taylor series does. As a result of the decreased accuracy, the difference isn't always positive or always negative, rather, it swaps between the two from point to point.

Exp

The difference graph for Exp (Figure 11) has negligible values up until x = 4, where the difference begins to become more noticeable. It increases in value as x increases, which is exactly the same the previous graphs (Sin/Cos/Tan). The smaller accuracy relative to the <math.h> function can also be attributed to decreased iterations of the taylor series, which, as previously stated, results in less accuracy.

Log

The difference graph for Log (Figure 12) is drastically different from the other difference graphs mentioned. It doesn't follow the same convention where it increases as it goes farther away from 0. Rather, it decreases in value as it x goes farther away from 0. However, there are instances where the value of x increases and then decreases right away, at around x = 1.5, 2.2, 3.3, 5.0, and 7.5. I noticed that the x values of these fluctuations occur at 1.5^(k), where k is a positive integer value. That would mean the next fluctuation would occur at around 11.4 if the domain were increased beyond 10.0.

Conclusion/Takeaways

In conclusion, I have learned that it is very hard to approximate complex functions accurately and that perfectly representing these real numbers is impossible. These complex values always result in an approximation and it is much, much easier to analyze differences between values if they are graphed. Plus, I have also learned that there are many different ways to implement these functions, with varying degrees of accuracy and success.

Images (Graphs & Screenshots)

Figure 1.

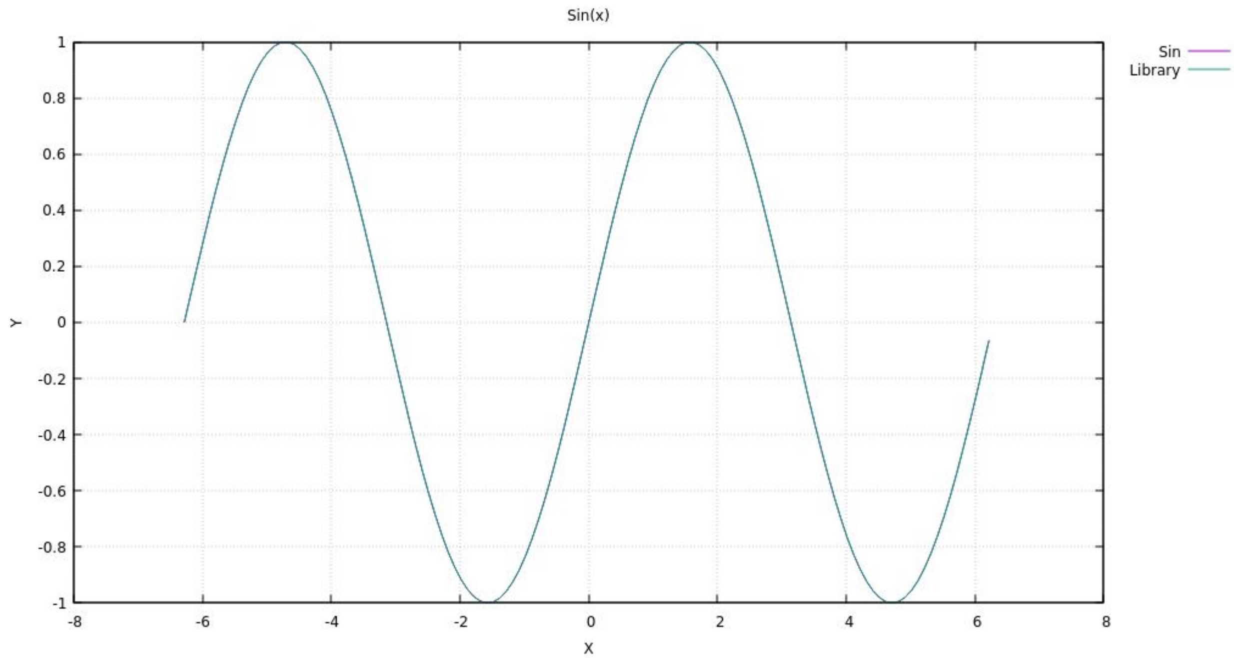

Figure 2.

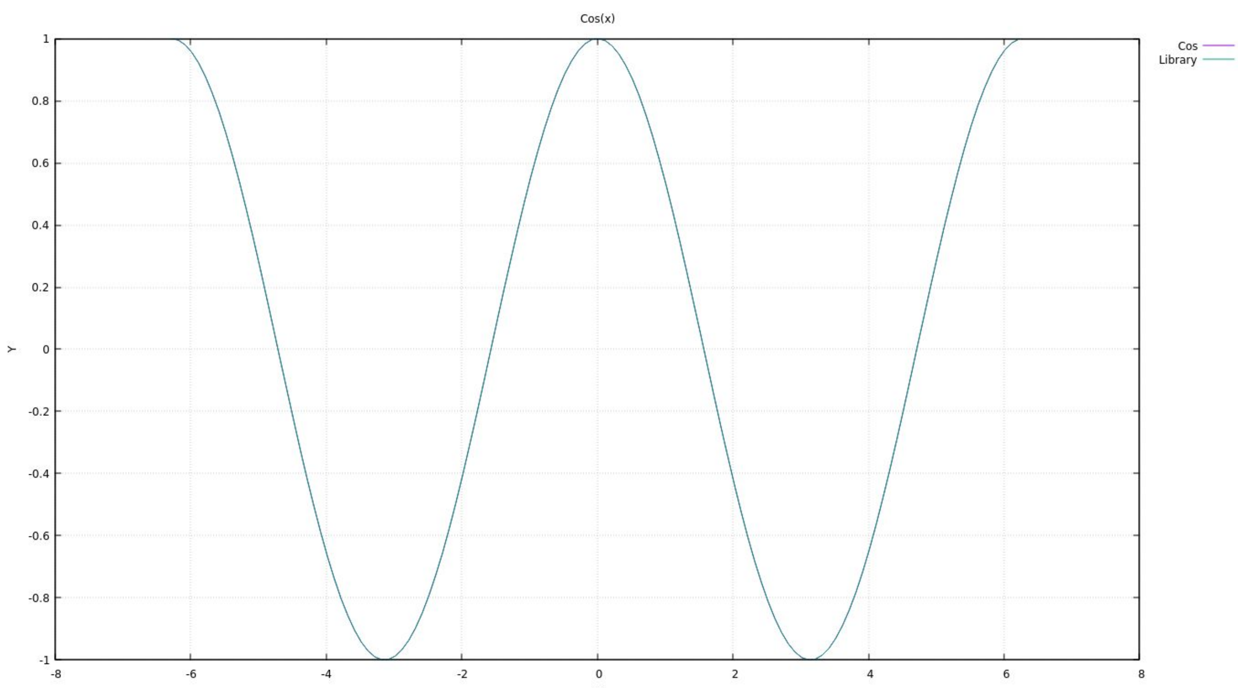

Figure 3.

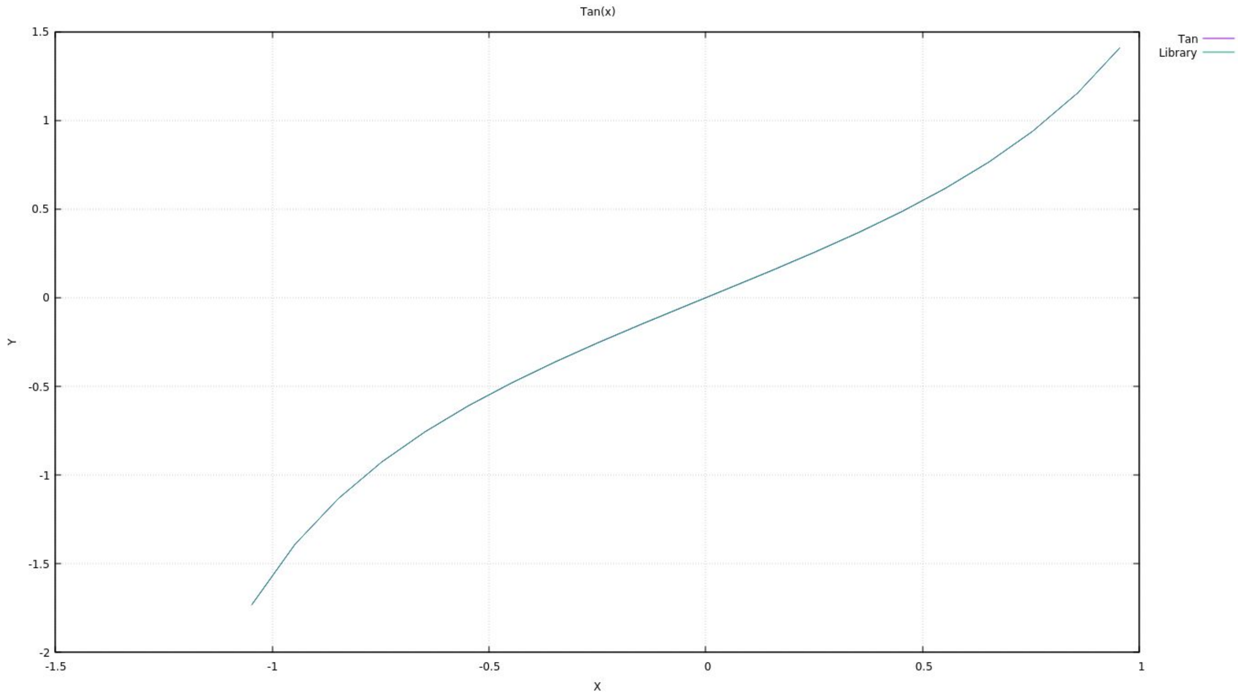

Figure 4.

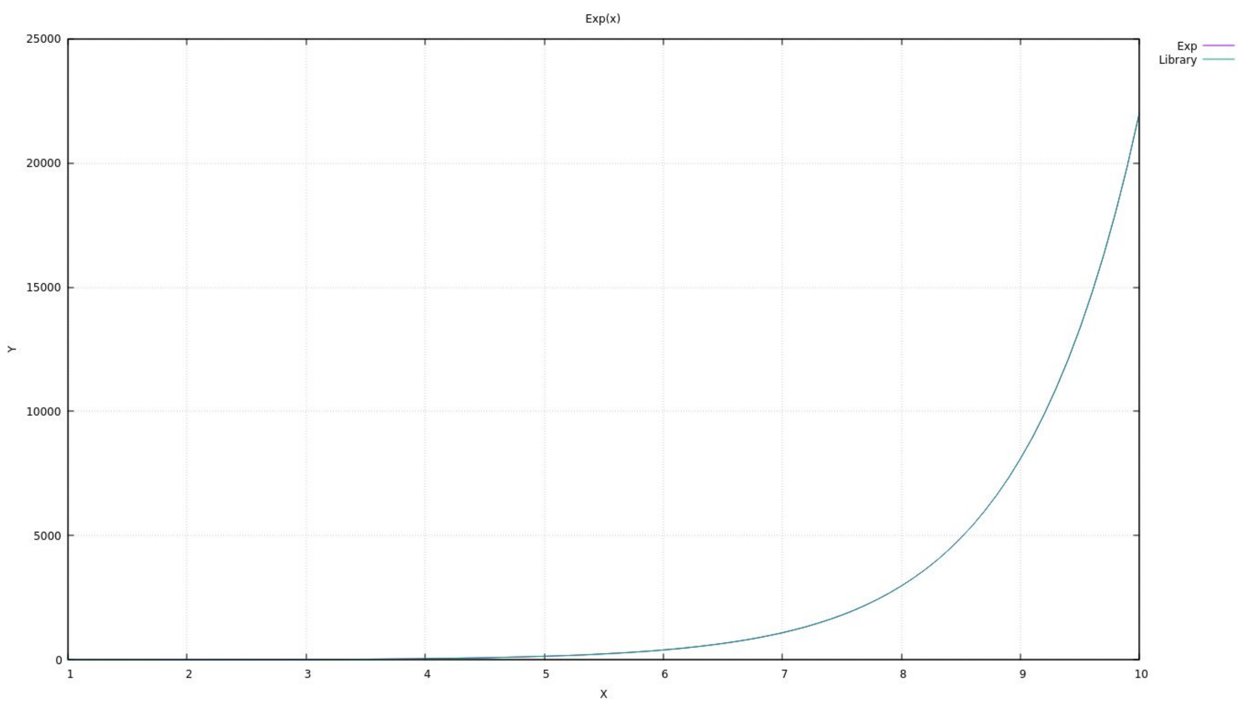

Figure 5.

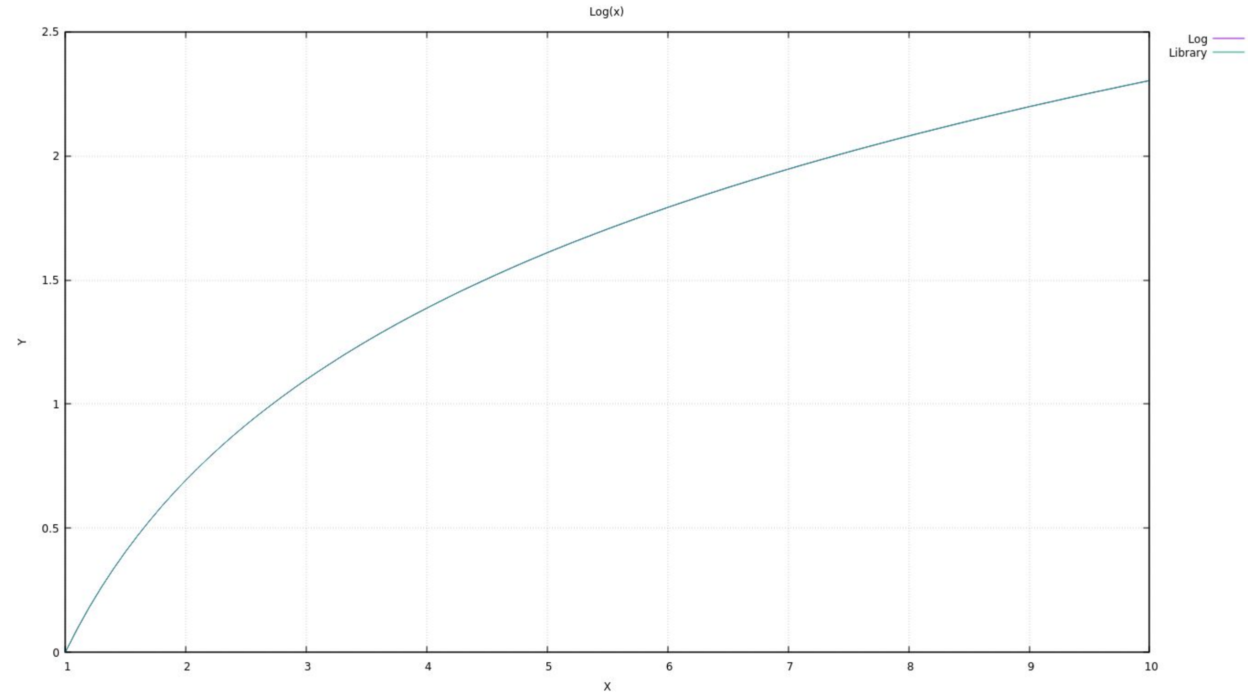

Figure 6.

x	Log	Library	Difference
-	---	-------	----------
6.6000	1.88706964903237972564	1.88706964903237972564	0.00000000000000000000
6.6100	1.88858365386359494664	1.88858365386359472460	0.00000000000000022204
6.6200	1.89009536994891691464	1.89009536994891669259	0.00000000000000022204
6.6300	1.89160480419777132788	1.89160480419777088379	0.00000000000000044409
6.6400	1.89311196348834243075	1.89311196348834220871	0.00000000000000022204
6.6500	1.89461685466776263986	1.89461685466776263986	0.00000000000000000000
6.6600	1.89611948455229795130	1.89611948455229750721	0.00000000000000044409
6.6700	1.89761985992753201558	1.89761985992753201558	0.00000000000000000000
6.6800	1.89911798754855420945	1.89911798754855398741	0.00000000000000022204
6.6900	1.90061387414013660546	1.90061387414013682751	-0.00000000000000022204
6.7000	1.90210752639692004529	1.90210752639692004529	0.00000000000000000000
6.7100	1.90359895098358999910	1.90359895098358999910	0.00000000000000000000
6.7200	1.90508815453505775395	1.90508815453505775395	0.00000000000000000000
6.7300	1.90657514365663582900	1.90657514365663605105	-0.00000000000000022204
6.7400	1.90805992492421516715	1.90805992492421516715	0.00000000000000000000
6.7500	1.90954250488443788569	1.90954250488443788569	0.00000000000000000000
6.7600	1.91102289005487224749	1.91102289005487224749	0.00000000000000000000
6.7700	1.91250108692418319123	1.91250108692418296918	0.00000000000000022204
6.7800	1.91397710195230374985	1.91397710195230352781	0.00000000000000022204
6.7900	1.91545094157060402651	1.91545094157060402651	0.00000000000000000000
6.8000	1.91692261218206039253	1.91692261218206039253	0.00000000000000000000
6.8100	1.91839212016142046657	1.91839212016142024453	0.00000000000000022204
6.8200	1.91985947185537031423	1.91985947185537009219	0.00000000000000022204
6.8300	1.92132467358269809488	1.92132467358269787283	0.00000000000000022204
6.8400	1.92278773163445837469	1.92278773163445837469	0.00000000000000000000
6.8500	1.92424865227413310897	1.92424865227413310897	0.00000000000000000000
6.8600	1.92570744173779284658	1.92570744173779306863	-0.00000000000000022204
6.8700	1.92716410623425726811	1.92716410623425704607	0.00000000000000022204
6.8800	1.92861865194525172740	1.92861865194525128331	0.00000000000000044409
6.8900	1.93007108502556623542	1.93007108502556623542	0.00000000000000000000
6.9000	1.93152141160321266788	1.93152141160321266788	0.00000000000000000000
6.9100	1.93296963777957775399	1.93296963777957753194	0.00000000000000022204
6.9200	1.93441576962957717534	1.93441576962957717534	0.00000000000000000000
6.9300	1.93585981320181099719	1.93585981320181077514	0.00000000000000022204
6.9400	1.93730177451871177219	1.93730177451871199423	-0.00000000000000022204
6.9500	1.93874165957669997162	1.93874165957669974958	0.00000000000000022204
6.9600	1.94017947434632742620	1.94017947434632720416	0.00000000000000022204

Figure 7.

x	Exp	Library	Difference
-	---	------	----------
6.6000	735.09518924197254818864	735.09518924197266187548	-0.00000000000011368684
6.6100	742.48301871662249595829	742.48301871662238227145	0.00000000000011368684
6.6200	749.94509711188186429354	749.94509711188186429354	0.00000000000000000000
6.6300	757.48217064180846591626	757.48217064180869328993	-0.00000000000022737368
6.6400	765.09499302003678167239	765.09499302003700904606	-0.00000000000022737368
6.6500	772.78432553514903702307	772.78432553514858227572	0.00000000000045474735
6.6600	780.55093712680240969348	780.55093712680286444083	-0.00000000000045474735
6.6700	788.39560446263055837335	788.39560446263101312070	-0.00000000000045474735
6.6800	796.31911201590366999881	796.31911201590389737248	-0.00000000000022737368
6.6900	804.32225214398033585894	804.32225214397988111159	0.00000000000045474735
6.7000	812.40582516754068365117	812.40582516754113839852	-0.00000000000045474735
6.7100	820.57063945062623133708	820.57063945062611765024	0.00000000000011368684
6.7200	828.81751148146679497586	828.81751148146736341005	-0.00000000000056843419
6.7300	837.14726595413981158345	837.14726595414026633080	-0.00000000000045474735
6.7400	845.56073585103399636864	845.56073585103376899497	0.00000000000022737368
6.7500	854.05876252614848453959	854.05876252614848453959	0.00000000000000000000
6.7600	862.64219578923416520411	862.64219578923393783043	0.00000000000022737368
6.7700	871.31189399076890822471	871.31189399076913559838	-0.00000000000022737368
6.7800	880.06872410779863002972	880.06872410779908477707	-0.00000000000045474735
6.7900	888.91356183063282969670	888.91356183063294338353	-0.00000000000011368684
6.8000	897.84729165041369469691	897.84729165041358101007	0.00000000000011368684
6.8100	906.87080694756707544002	906.87080694756775756105	-0.00000000000068212103
6.8200	915.98501008114510568703	915.98501008114499200019	0.00000000000011368684
6.8300	925.19081247905376130802	925.19081247905364762119	0.00000000000011368684
6.8400	934.48913472920503409114	934.48913472920514777798	-0.00000000000011368684
6.8500	943.88090667157257485087	943.88090667157302959822	-0.00000000000045474735
6.8600	953.36706749117843173735	953.36706749117820436368	0.00000000000022737368
6.8700	962.94856581200735945458	962.94856581200770051510	-0.00000000000034106051
6.8800	972.62635979187814427860	972.62635979187814427860	0.00000000000000000000
6.8900	982.40141721825239073951	982.40141721825250442635	-0.00000000000011368684
6.9000	992.27471560501919611852	992.27471560501919611852	0.00000000000000000000
6.9100	1002.24724229024445776304	1002.24724229024479882355	-0.00000000000034106051
6.9200	1012.31999453490823270840	1012.31999453490811902157	0.00000000000011368684
6.9300	1022.49397962262776218267	1022.49397962262787586951	-0.00000000000011368684
6.9400	1032.77021496039083103824	1032.77021496039105841191	-0.00000000000022737368
6.9500	1043.14972818029468726309	1043.14972818029491463676	-0.00000000000022737368
6.9600	1053.63355724231109888933	1053.63355724231109888933	0.00000000000000000000

Figure 8.

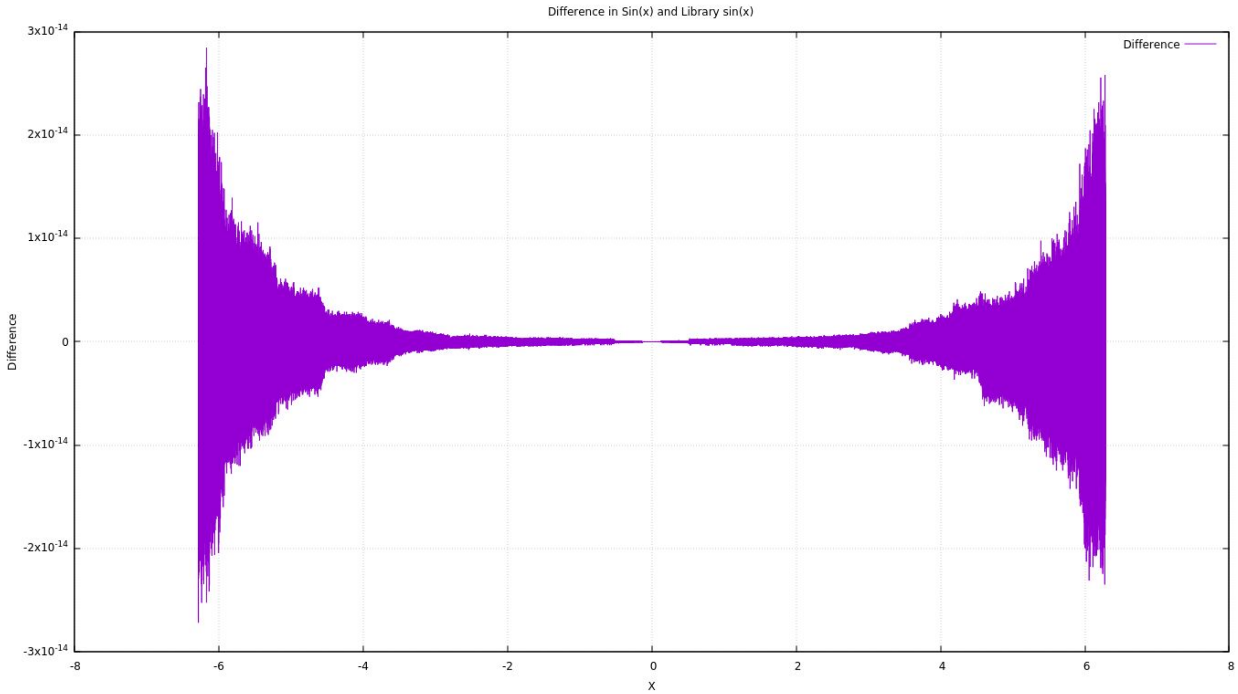

Figure 9.

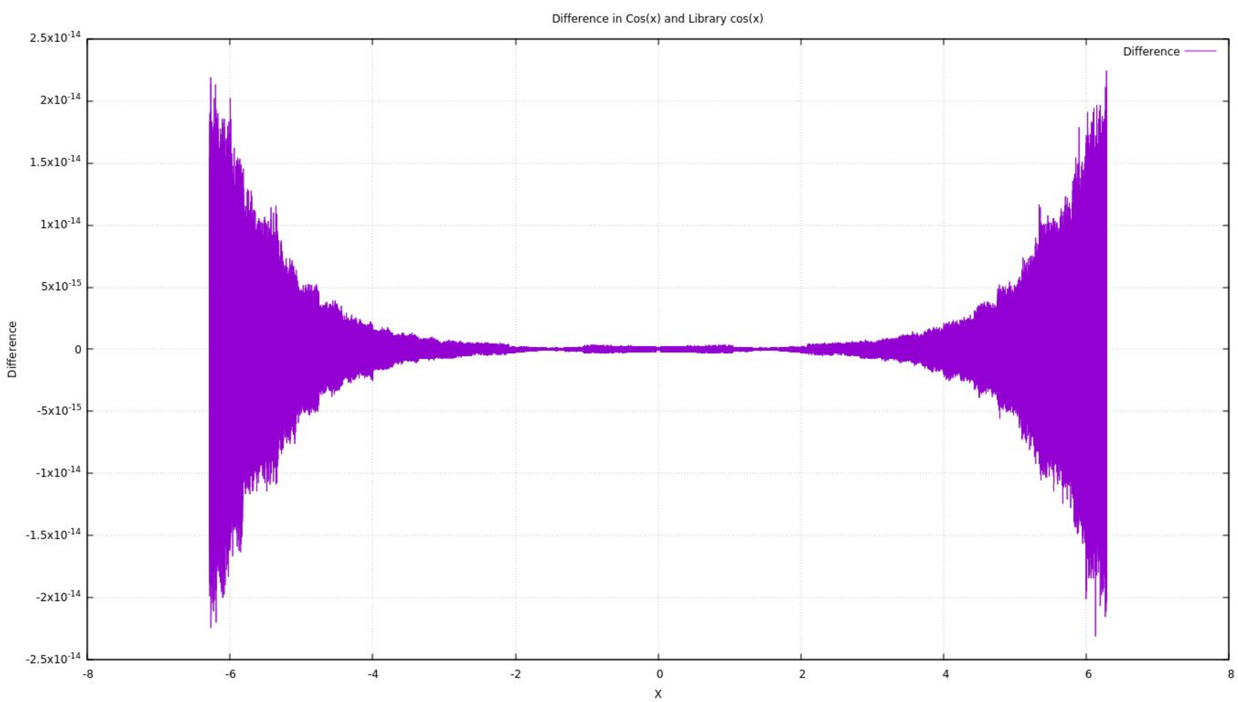

Figure 10.

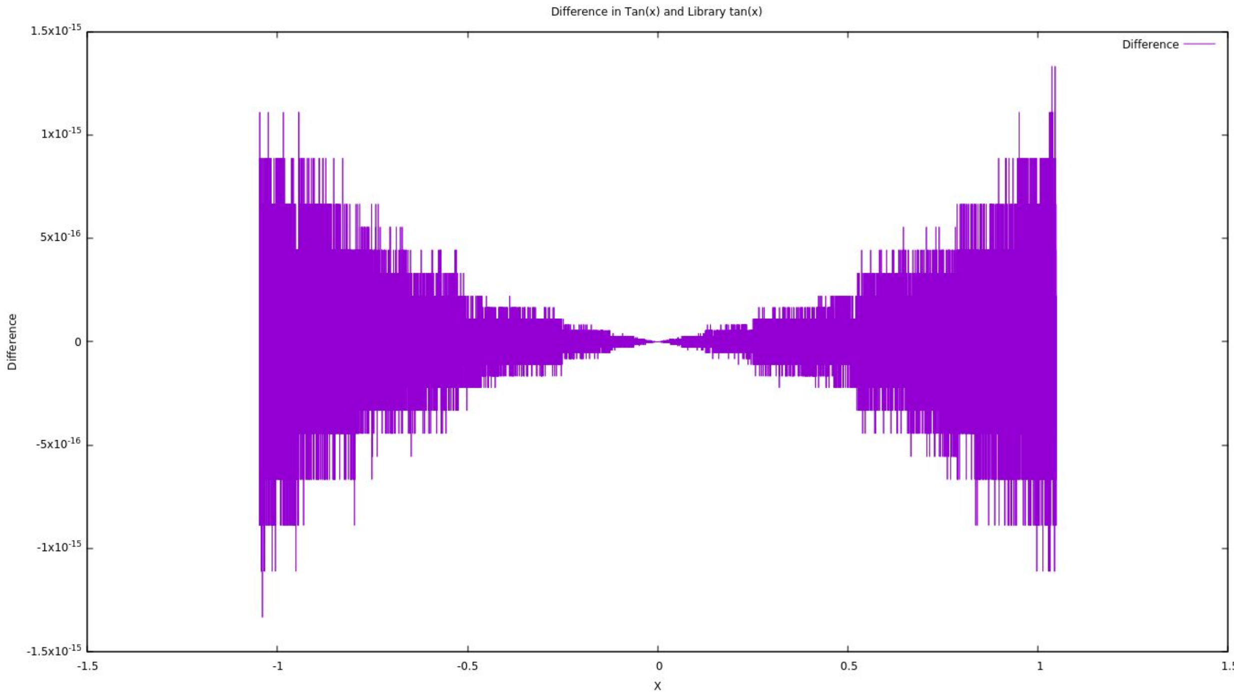

Figure 11.

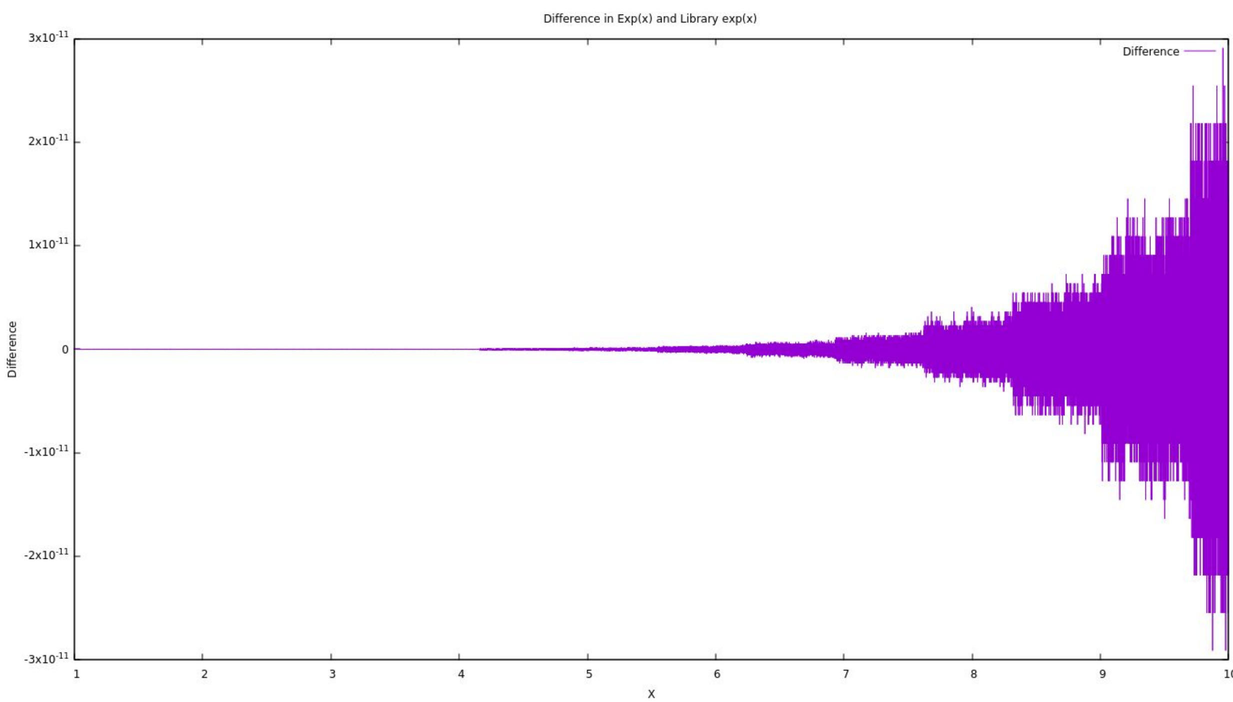

Figure 12.

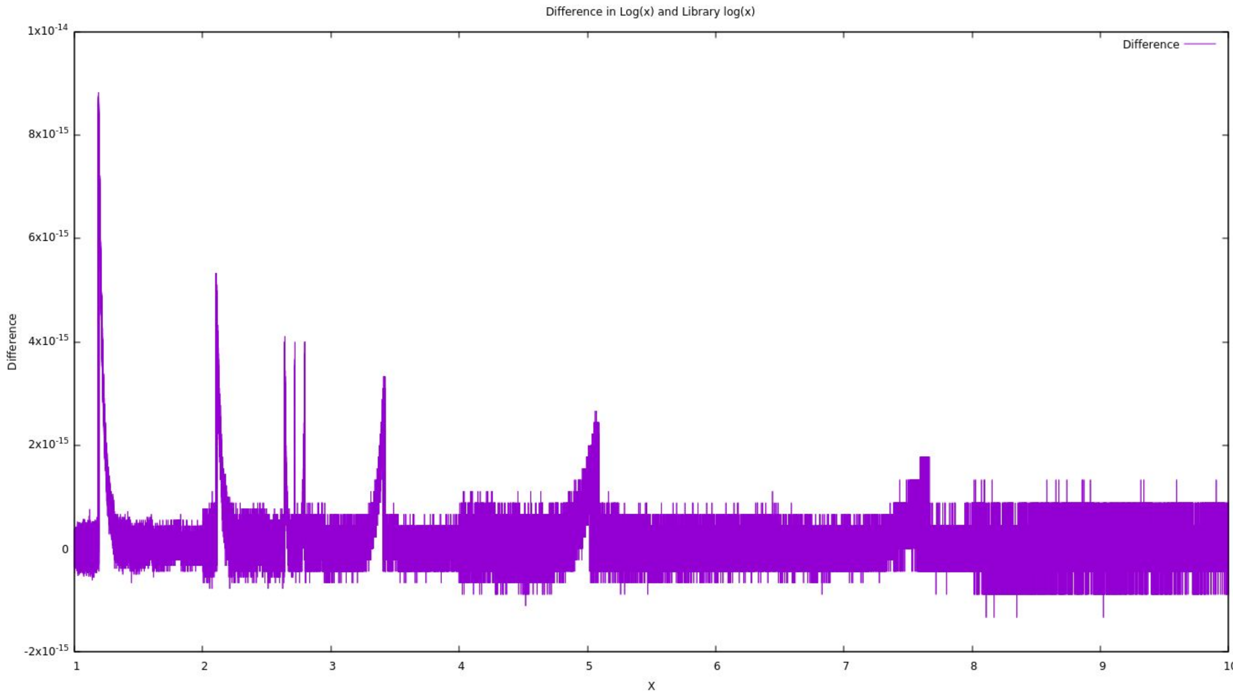

Figure 13.

```c
if (do_log) {
    printf("Calculating log, writing to .dat file.");
    double n = 0.0;
    FILE *fp;
    fp = fopen("exp.dat", "w");
    while (n < 10) {
        fprintf(fp,"%7.10lf %16.20lf\n", n, (Exp(n) - exp(n)));
        n += 0.00001;
    }
    fclose(fp); // Closing File
}
return 0;
```

References

1. https://www.javatpoint.com/fprintf-fscanf-in-c
2. http://www.cplusplus.com/reference/cstdio/fprintf/
3. http://www.gnuplot.info/
4. https://www.youtube.com/watch?v=9QUtcfyBFhE
5. https://www.youtube.com/watch?v=F_XcgIxdExE&ab_channel=MathAndPhysics