Lab 1 Report
Ian Taulli
Computational Physics

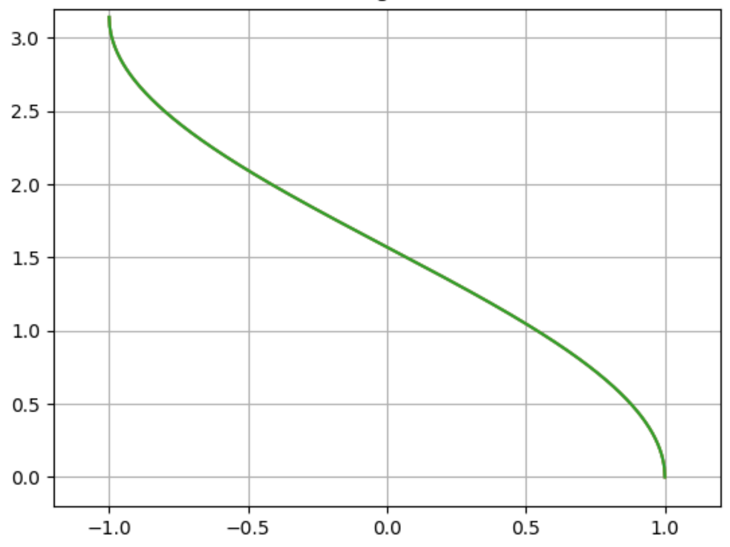
Part 3: Avoiding Excessive Error Amplification

Converting the formula:

$$\alpha = \arccos\left(\frac{\vec{a}\cdot\vec{b}}{|a||b|}\right) \tag{1}$$

directly into computer code means evaluating the Taylor series for arccos near the point of interest. If $y = f(x)$, then the error in y is proportional to $f'(x)\frac{x}{y}$. Examine the shape of the arccos function:

fig.1



Observe that near x = 1, the magnitude of the derivative approaches infinity. In the case, where the angle between the two vectors is small, the quantities $\vec{a}\cdot\vec{b}$ and $|a||b|$ are almost the same. Then $f'(x)$ is large, and $\frac{x}{y}$ is large as well. Therefore, for small angles, the error in the direct arccos calculation is large. A simple substitution for directly calculating arccos is a table look-up. Rather than calculating the Taylor series directly, one would round the argument to the available accuracy and match the value to a table. This circumvents having to deal with the error amplification locally, but another (suitable) method would be needed to calculate the table in the first place.

Part 4: Convergence Does Not Mean Correctness

The loop will stop when old_sum = sum + $\frac{1}{k}$, implying that $\frac{1}{k}$ must be within the machine precision of the program. The 32-bit decimals c++ uses have a precision of $5 \cdot 10^{-7}$, so the loop should stop around $k = \frac{1}{5 \cdot 10^{-7}} = 2 \cdot 10^6$. This is why the program appears to converge. When $k = 2097153$ the program can no longer differentiate old_sum from sum, so the for loop is broken and the final values of k and sum are printed. To avoid this condition stopping the sum the float variables are converted into doubles. In addition to turning the floats into doubles, it is useful to ask c++ to print after every million iterations. In this case, the last print occurs when $k = 4294000000$. Afterward k is returned as 0 and 1/0 is added to sum, causing sum to become inf. Notice that $2^{32} = 4294967296$. The iterable k is a 32-bit number, so before k can reach 4295000000 it hits it's upper limit and appears to be re-assigned k = 0. This is what causes the apparent bug in the harmonic series sum using doubles.