COMPPHYS FINAL

knudst

May 2021

1 Background: The Problem

I've currently set out to solve the problem of calculating pi using an n-sphere. This problem presents many unique challenges, chief among them being the visualization and wrapping of one's head around the idea of n-dimensional space. It's easy to understand the idea of a circle, or a sphere, but it is quite literally impossible for us to visualize the idea of a 4-Dimensional, 5-Dimensional, or any other higher dimensional object using our 3-Dimensional frame of reference.

2 Background: The Solution (Monte Carlo)

The solution to this dilemma is to not think of the spheres as objects occupying space but as a set of mathematical boundaries that a point can or can not fall within. From here we enter the Monte Carlo algorithm. Monte Carlo is an algorithm that through the use of random generation can be used to determine the area of shapes. The idea is that there is a region in space where points can be generated, and a space within that region that the object we are trying to calculate the n-volume of occupies. Points are randomly generated in this entire region of space and a check is done to determine whether or not that point falls within the region of the shape whose n-volume we are trying to determine. Given the ratio of points that fall in this space to the total number of points generated, we can then make an estimate to the volume that shape occupies, assuming we know the volume of the total region of space

3 Calculating Volume of n-Dimensional Shapes

Monte Carlo algorithm relies on the ability to accurately calculate the volume of an n-dimensional object, but how do we do that? Well, we only need to know the volume of one of the two objects we are trying to calculate, the n-cube. The n-cube defines the boundaries of the region of space we are generating points in, and can be easily calculated as:

$$V_n = s^n$$

Where n is the number of dimensions and s is the side length of one edge of the cube.
The points can then be checked against the definition of the boundaries of an n-sphere. For a point to fall within the region of an n-sphere, the following must be true:

$$x_1^2 + x_2^2 + x_3^2 ... + x_n^2 <= R^2$$

Where x is the coordinate in n-space and R is the radius of the n-sphere Using the ratio of points we calculated through the Monte Carlo algorithm, the volume of an N-Sphere looks like this

$$V_n = s^n * Ratio$$

Now, the larger the number of generated points, the higher accuracy the program will have, which will help us when determining pi.
The generalized formula for determining the volume of an n-dimensional sphere is as follows:

$$V_n(R) = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} * R^n$$

Where Γ is Euler's Gamma function. Setting the center of the sphere at the origin(IE, all parameters are zero at the center of the sphere), we can determine that the radius of the sphere is $\frac{1}{2}$ of the length of one side of the n-cube, analogous to 2 or 3 dimensional space.
Using this, we can now determine the value of pi simply by solving for pi using the above equation.

4 Results:Dimensionality and Computational Time

As expected, there is a clear decrease in accuracy for a given number of points when there is an increase in the number of dimensions. This can be explained simply by looking at the previously referenced definition of a point that falls within the n-sphere:

$$x_1^2 + x_2^2 + x_3^2 ... + x_n^2 <= R^2$$

Notice how as more dimensions are added to the n-sphere, it becomes more and more likely that there will be enough to exceed the value of R^2, thus, as more dimensions are added to the n-sphere, it is less and less likely a randomly generated point will fall within its bounds. This presents a clear issue of computational time for higher dimensional shapes/data-sets when using the Monte Carlo algorithm. The only way to maintain accuracy with higher dimensional data in Monte Carlo is to add more points, however, depending on how many points you're adding, and how high the dimensionality of the data you're working with is, this can add incredibly long amounts of computational time.

When working with high dimensional data-sets, parallel computation can be a useful tool. Allowing multiple threads to calculate the runs could reduce this task to O(1) time depending on how powerful of a GPU you posses and on the previously stated dimensionality of and number of points in your data. For example, the calculation of pi using just a 10-Dimensional Sphere and 10000 data points takes roughly $\frac{1}{4}$ of a second to process, and this time only increases for even higher dimensional objects, which can take magnitudes more of points in order to output equally accurate results.

Take for instance the 13-D hyper-sphere, which takes a number of points in the magnitude of 10^5 in order to reliably produce even close to accurate results. These computations begin taking exponentially longer, for instance, a calculation of 300000 points takes five seconds to calculate and produces a result of 3.19, a far cry from the accuracy we want.

This also means, as demonstrated on the charts in the next page, that the time cost to calculate a number of points outpaces the percent error we see in our determination of the value of pi. As we strive for more and more accuracy in the computation, we inevitably, for purposes of practicality, utilize a graphics processing unit in our calculation of the data.

Both of these simulations were run with a 13-D N-Sphere. Time is in seconds.