Exercise #4 - Monte Carlo Integration

Computational Physics Lab

Instructor: Prof. Sean Dobbs

February 9, 2020

*Due by 11 PM, February 19th**

Submission

For each problem, you will write one or more python programs. These programs should follow the Python coding and formatting conventions outlined for the course. Many problems will have additional questions to answer, or will ask for a record of the terminal showing the output when the program is run with various arguements. These answers can be given either in block comments in the prologue of the corresponding assignment, or in clearly labeled text files. The terminal output should contain some information on the username and machine you are running on (this will often be in the command prompt).

You will be evaluated based on the files contained in your remote GitHub repository at the due date. You should and commit all the files you created to your local repository on a regular basis. You should do this by adding any new or modified files using "git add", and then finalizing changes using "git commit". Remember that "git status" will give you information on which files in your repository. have been changed. Remember to add short, but useful comments when performing a commit. When you are finished with the exercise, push the current status of your local repository to the remote repository on GitHub using the command "git push -u". You are encouraged to push your local files to the remote repository periodically before you are done, and certainly well before the deadline, if possible.

1. Rolling Dice

a) Write a program that generates and prints out two random numbers between 1 and 6, to simulate the rolling of two dice.

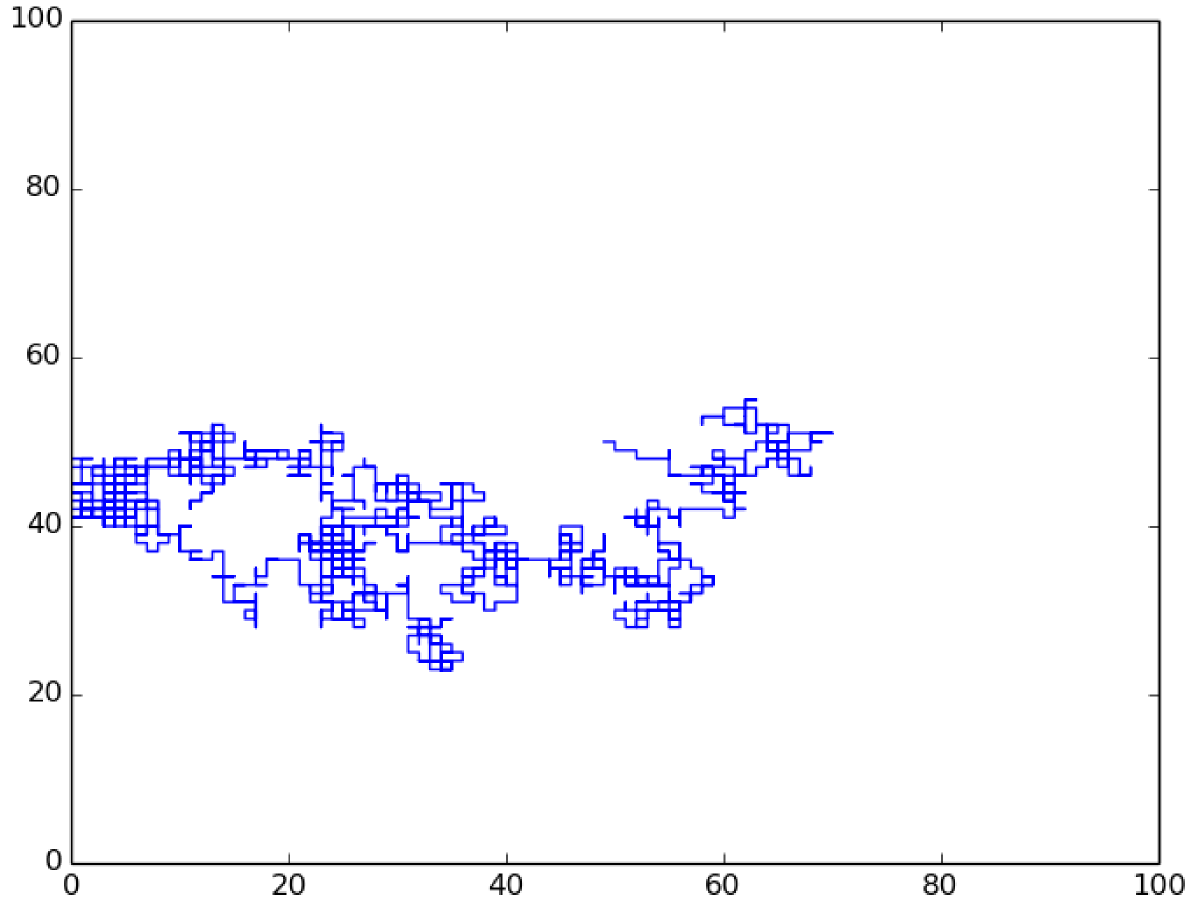
b) Modify your program to simulate the rolling of two dice a million times and count the number of times you get a double six. Divide the result by a million to get the fraction of times you get a double six. You should get a value close to, though probably not exactly equal to, 1/36. Your program should print the thrown dice values up to and including the first double six obtained. Have the program print out the number of double sixes, the number of dice throws, the double six fraction, and the number average number of throws per double six combination.

For full credit turn in your program, plus your results from part (b).

2. Brownian motion and random walks

Brownian motion is the motion of a particle, such as a smoke or dust particle, in a gas, as it is buffeted by random collisions with gas molecules. Make a simple computer simulation of such a particle in two dimensions as follows. The particle is confined to a square grid or lattice $L \times L$ squares on a side, so that its position can be represented by two integers $i, j = 0 \ldots L - 1$. It starts in the middle of the grid. On each step of the simulation, choose a random direction---up, down, left, or right---and move the particle one step in that direction. This process is called a random walk. The particle is not allowed to move outside the limits of the lattice---if it tries to do so, choose a new random direction to move in.

a) Write a program to perform 100 steps of this process on a lattice with $L = 101$ where the particle starts in the middle of the lattice, which will save an image of the path of the particle. (We choose an odd length for the side of the square so that there is one lattice site exactly in the center.) Run the program 5 times with N = 2000 steps to generate 5 images of 5 different random walks. The image should look like the following:



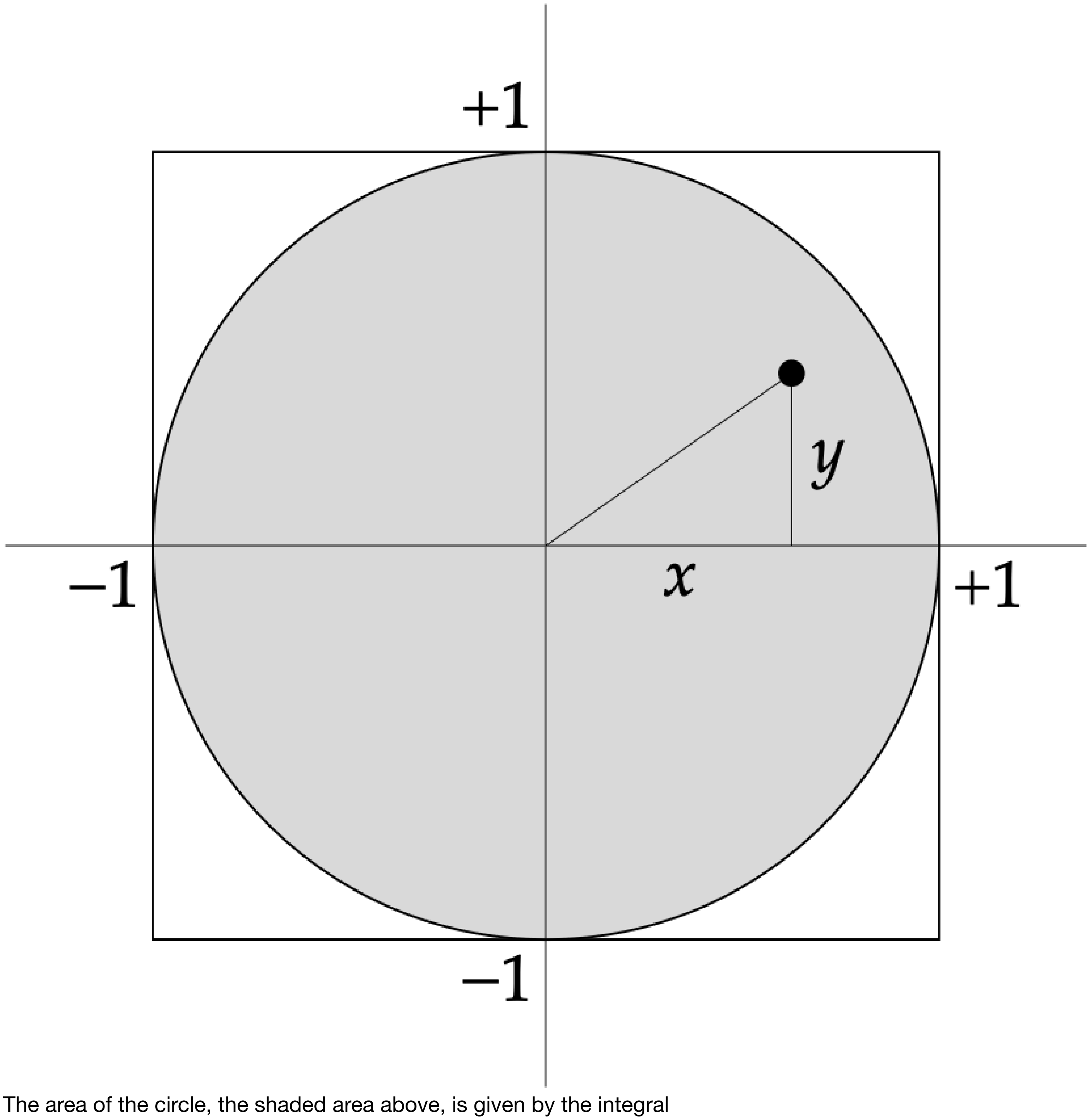
(b) We can use these ideas to solve many problems involving diffusion or the behavior of other particles in a dense medium. Let us consider the movement of photons in the sun. These photons frequently collide and scatter off of the charged particles that comprise the plasma in the sun. Note that the average distance between collisions, or the "mean free path", is $\ell = 1$ cm. First, modify your program from part (a) to have a lattice of L = 10001, and to run for N = 100000 steps. Then, calculate the distance D from the starting point to the final position of the particle, assuming the spacing between each lattice position is 1 cm. Now, run this simulation for 1000 particles, and calculate the average of the distance squared $\langle D^2 \rangle$ for these particles (Note that we expect that average of D to be $\langle D \rangle = 0$). The "root-mean-square" distance traveled is related to the number of steps by $D_{\mathrm{rms}} = \sqrt{N}\ell$. We can then determine the relationship between the number of steps and D (this is more useful for systems with more complicated behavior!).

Let's calculate a couple more interesting quantities. From our definition of D_{rms}, how many steps on average will the photon need to escape the Sun, i.e. for $D_{\mathrm{rms}} = 696,340$ km (the radius of the sun), assuming $\ell = 1$ cm. How many years will this take, assuming that the photon is moving at the speed of light between scatters?

For full credit turn in your programs from part (a) and (b), figures of position from part (a), and answers from part (b).

3. Volume of a hypersphere

This exercise asks you to estimate the volume of a sphere of unit radius in ten dimensions using a Monte Carlo method. Consider the equivalent problem in two dimensions, the area of a circle of unit radius: \medskip



The area of the circle, the shaded area above, is given by the integral

$$I = \iint_{-1}^{+1} f(x, y)\, dx\, dy,$$

where $f(x, y) = 1$ everywhere inside the circle and zero everywhere outside.

In other words, $f(x, y) = 1$ if $x^2 + y^2 \leq 1$, and 0 otherwise.

So if we didn't already know the area of the circle, we could calculate it by Monte Carlo integration. We would generate a set of N random points (x, y), where both x and y are in the range from -1 to 1. Then the two-dimensional version of Eq. (10.33) for this calculation would be

$$I \simeq \frac{4}{N} \sum_{i=1}^{N} f(x_i, y_i).$$

(a) Generalize this method to the ten-dimensional case and write a program to perform a Monte Carlo calculation of the volume of a sphere of unit radius in ten dimensions.

If we had to do a ten-dimensional integral the traditional way, it would take a very long time. Even with only 100 points along each axis (which wouldn't give a very accurate result) we'd still have $100^{10} = 10^{20}$ points to sample, which is impossible on any computer. But using the Monte Carlo method we can get a pretty good result with a million points or so.

(b) Modify your program to calculate the hyper volume as a function of the dimensions from n = 0 to n = 12. Print the results to the screen and generate a graph of the hyper volume vs dimension.

(c) Modify your program to calculate the error on the integral. Take the case of calculating the volume of a hypersphere of dimension n = 10. Calculate the error on this volume integral from N = 20,000 to N = 1,000,000 with N changing in steps of 20,000 (this should be 50 different values). Print the results to the screen and generate a graph of error vs N.

For full credit turn in a your final program, results from the various calculations in part (a), and the graphs and results from parts (b) and (c).