First Passage Time of a two-dimensional random walk

Objectives: The purpose of this assignment is to have you use control structures such as "if" statements and "for" and "while" loops, in the context of simulating a random walk in two dimensions. You will have the opportunity to plot and display your results for more than one trial of the simulation, allowing for some basic statistical analysis.

Logistics All homework assignments will consist of writing programs, one for each part of a problem, and providing a printout of the problem along with the output obtained when the program is run. These can be saved as pdfs or in html format and uploaded under the Assignment on Moodle, one combined file per homework assignment. Please write your name and the date at the top.

The different parts of the problem set will be graded as indicated on the assignment sheet. This includes the following (for 5 points): every program you turn in should start with a **comment or docstring** that lists **what the problem is**. Each part of your program should also have short comments describing how the program works. One way to turn in homework is to do it all in a **Jupyter notebook** – that is a convenient way of organizing and integrating code with explanations and equations, which need to be written in LaTeX. The notebook can then be saved as an html file.

Introduction The motion of a diffusing particle is often described as a random walk, since at each point in time the direction in which the particle moves comes from random collisions with its surrounding atoms and molecules. For simplicity, a random walk is often modelled on a lattice, where the particle is confined to move along the points of the grid. In this problem set, you will look at a random walk on a finite, two-dimensional square grid of points. The walker will begin at a point, and move according to a set of rules described below. One of the four boundaries will be absorbing, which means the walk ends when the particle reaches that boundary. This is an example of a "first passage" process, when something occurs for the first time. There are many examples of such processes, such as the firing of a neuron when the action potential threshold is crossed, or the Gambler's Ruin, which occurs when a gambler runs out of money. The obvious question to ask is how long, on average, does it take for the first passage process to occur, or in our case, for the random walk to end.

Simulating a random walk in two dimensions

Physicists often model motion using a lattice or grid to simplify the calculation. For this problem, you will use a two-dimensional grid that is 100 steps in "x" and 100 steps in "y"; the x axis goes from -50 to +50 and the y axis also goes from -50 to +50. Three of the walls of the system (at $x = -50$, $y = -50$ and $y = +50$) are ordinary, but the fourth wall (at $x = +50$) is absorbing.

On this grid, we'll start a particle at the origin, $(x, y) = (0,0)$, and then simulate it diffusing around, randomly, until it hits the absorbing wall, at which point the simulation ends. At each time step, the particle can move one step on the grid and your simulation will update the particle's x and y positions accordingly. Here are the rules for the random walk:

At each time step:

1. If the particle is not at an edge, it can take a step of one grid spacing in any of four directions: $\pm x$, $\pm y$; these should all be equally probable.

2. If the particle is at one of the ordinary walls, it cannot pass out of the box. It should have a 50% probability of taking a step perpendicularly away from the wall, and a 25% chance of stepping in either of the directions along the wall.
3. If the particle is at a corner, it has a 50% chance of going in either of the two directions along the walls.
4. If the particle touches the absorbing wall, the random walk ends.

Part 1. Simulate a single random walk until absorption.

A. (25 pts) Write a program that executes one 'walk' and calculates the number of steps the particle takes before being absorbed. Since each step takes one unit of time, you are also finding the time to absorption, also called the *first passage time*. You will be assessed on whether you have correctly implemented the walking rules above. It might be helpful to use the following function: np.random.randint(low,high) which picks a random integer between the integer value 'low' and 'high' (including low but excluding high)..

B. (15 pts) Make a plot of the trajectory taken by the particle. Use the matplotlib library.
   a. Label the x and y axes appropriately, and give the plot a title
   b. Indicate the starting position with a black asterisk and the end position with a red asterisk.

C. (5 pts) Print the time (i.e. number of steps) it took for the particle to be absorbed.

Part 2. Simulate many random walks and get some statistics.

If you run your code a few times, you will find that the time-to-absorption can vary enormously. We'd like to learn more about the **distribution** of absorption times.

D. (25 pts) Modify your code so that it measures the time-to-absorption for N different particles each starting at (0,0). Each of these walks can be thought of as an independent trial. (For this part, you will want to comment out or remove the part that plots the trajectory). Start with N = 100 trials, and move on to repeat this for N = 1000. Store the times in a list or an array.

E. (20 pts) Plot a histogram of the 1000 "first passage time" results found by your code. Look up how to generate histograms. Make sure the axes make sense and are labeled! For example, the x axis should be "First passage time". The Y axis should be "Number of occurrences".

F. (5 pts) Another way of interpreting your histogram is that it is proportional to the probability of the particle "surviving" to time *t* before being absorbed, $P(t)$. Looking at your histogram, what do you think the functional form of $P(t)$ might be? (You do **not** need to do a curve fit or other analysis here!) Your answer to this part should either be a clearly-labeled comment at the end of your Part 2 program, or as part of your Jupyter notebook.

Remember to turn in your complete code for both parts, with output and plots generated in each, as a single pdf or html file.

Due date: 11:59 pm on Friday September 28, 2018