Project 7: Monte Carlo Methods

Due Dates

Week 10, Monday at 4:59

Your submission should be a single zip file containing all the .py files containing your code as well as a Single typed PDF document with your test results and answers to the problems/questions. Word documents will not be accepted. Your zip file should be in the form <lastname>_project_< N >.zip where you replace <lastname> with your last name and <N> with the project number. Upload your zip file via the link provided on the project webpage found at http://urminsky.ca.

In your PDF, please include your python code and any figures or graphs you were asked to plot.

No handwritten work will be accepted. Please include any mathematics in a typed format. You are welcome to use latex or word to do your write up.

Please write and hand in your own work. Work which is deemed too similar to another students work will be given an automatic zero. It is ok to work together, in fact, I would encourage you to work together. However, when too many similarities exists between two or more student's work, all work will be given automatic zeros.

Documentation: *Make sure that your code is well enough commented so that it can be easily understood by almost anyone, including yourself, if you look at it again at the end of the course (when you've forgotten how you coded it.)*

Project Briefing

This week you have two tasks. The first is to set up a very simple Monte Carlo integration to estimate the value of . The second is to construct a program to simulate a 1-dimensional random walk. You will then use this program to investigate how the mean and root-mean-square displacement vary with the number of steps.

You may and should use numpy's random number generators

Part 1

Design, write, and test a PYTHON program that performs a Monte Carlo integration to find the area of a circle of unit radius and hence determine the value of π. The program should be called MC_circle and should take the input argument N_points. What it should then do is use the built-in random number generator to generate N_points points (x, y) in the unit square ($0 < x < 1$ and $0 < y < 1$). It should then count the number of these points that are within a radius of one of the origin. Since the area of such a quarter circle is $\pi/4$, this should allow you to generate an estimate of π.

Use your program to estimate using 50, 500, 5000, and 50000 points. For each value of N_points, carry out enough trials (i.e. randomly generate new sets of points) to get an average value for π and standard deviation. Record these values in a table.

How does the standard deviation decrease as the number of points increases (i.e., what is the functional dependence)? Automate the proceedure above using PYTHON. write a script which call MC_circle many times and use any built in Python functions to compute the average and standard deviations.

Part 2

Design, write, and test a PYTHON program that performs a direct Monte Carlo simulation of a 1-d random walk. Assume that the step-size is 1 unit of length. (We dont really need to specify an absolute unit, since there are no other physical quantities directly involved. We can a posteriori adopt a unit suitable to our application. For example, if we wanted to simulate a drunk walking home from a bar, an appropriate choice would be 1 meter.)

The program should:

- Compute a user-specified number of random walks, all having the same number of steps;
- Calculate the mean displacement $< x >$ and the root mean square displacement, xrms $= \sqrt{< s2 >}$, for the set of walks;
- Print the mean and root mean square values of the displacement.

The program should be named rwalk1d. The following input parameters should be supplied to the program as input arguments:

N_walks the number of random walks to be computed

N_steps number of steps in each walk

P_neg probability that step is in negative direction

Program design issues

To make your code modular, have rwalk1d itself call a (sub)-function walk to compute each individual random walk. The sub-function walk should take N_steps and P_neg as arguments and it should return the cumulative displacement, x, for the walk. This subfunction should be called within a for loop and within this loop keep running totals of x and x2. When the loop has completed, the mean values of x and x2 may be computed. Give some thought to testing your program. Previously, you tested your programs (or should have) by comparing your output with exact theoretical solutions. Here you are modeling a random process, so what can you do to check that the program is working correctly?

Part 3

When you are satisfied that the program is working properly, set P_neg = 0.5 and run a series of experiments in which N_steps is increased from 1 to at least 1000. N_walks should be kept fixed at a suitably large value, at least 1000. Note that it is not necessary to increase N_steps by the same additive amount each time. Formulate a strategy for selecting values of N_steps to cover the specified range, noting that the range covers several orders of magnitude.

Suppose that at each step, the step length decreases by a certain amount, say 10%. That is, the second step has a length 90% of the first, the third step length is 90% of the second and so on. This might happen if the particle undergoing the random walk loses kinetic energy as a

result of collisions. Modify your program to create a new program to investigate this case. What happens to the mean and rms displacements?