PHYSICS 3800 Assignment 3

For this assignment, all the "number crunching" should be done in C or Fortran. However, plotting and data analysis should be done in Python within a Jupyter notebook. For example, for Q1 below, you will need to compare your result with theoretical predictions that involve binomial coefficients, and so you could use the SciPy library for that. For Q2, MC evaluations of the integral are stored in files that are then read in for averaging and plotting in the Jupyter notebook.

Your submission should be a pdf produced from your Jupyter notebook, uploaded to Brightspace. Include your C/Fortran source code in a markdown cell. If you are not able to produce a pdf of the notebook, just upload the notebook itself to Brightspace.

1. (10 pts.) Write your own Generalized Feedback Shift Register random number generator with $p = 521$ and $q = 168$ to generate numbers between 0 and 1. You may use any RNG to generate the initial sequence of integers.

   Use your RNG to generate 100000 realizations of a random walk 20 steps long. A step of length $\Delta x = 1$ is taken to the left or right with equal probability, and the random walker starts at $x = 0$. Plot a histogram of the end locations x_f and compare it with the analytically expected result. A sample graph is shown in Fig. 1.

   Having, say, $x_f = 4$ arises from choosing to take $N_R = 12$ steps to the right and $N_L = 8$ steps to the left. The probability of this happening is

$$ {}_{20}C_{12} \left(\frac{1}{2}\right)^{12} \left(\frac{1}{2}\right)^{8} = \frac{20!}{12!\,8!} \frac{1}{2^{20}}. $$

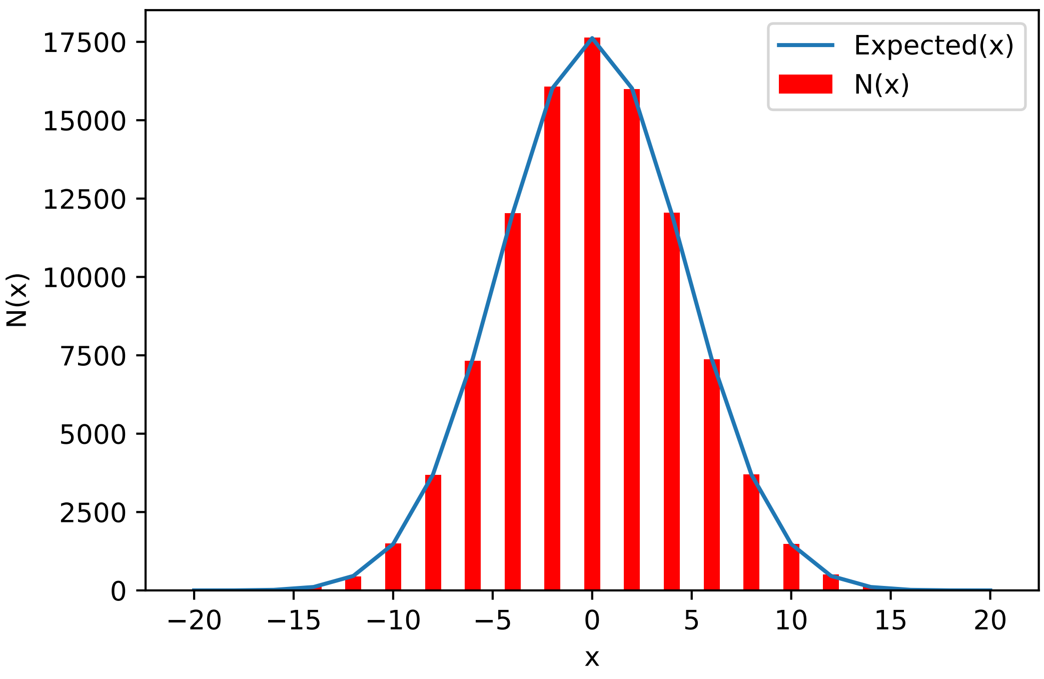

Figure 1: Comparison of simulated (red bars) and expected (line) distribution of final position x for a one-dimensional unbiased random walk of 20 discrete steps. Number of realizations is 100000. Note that with an even number of steps, x is even.

2. (10 pts.) Some physical problems require integration in many dimensions. For example, properties of a small atom like magnesium with 12 electrons requires integration over all three dimensions for all 12 particles giving a 36 dimensional integral. If we use 64 points for each integration, this requires $64^{36} \simeq 10^{65}$ evaluations. Even with a fast computer that can evaluate the integrand one million times per second, this would take 10^{50} seconds, which is significantly longer than the age of the universe (about 10^{17} seconds). The results of your task show how such problems can be done using MC techniques in the time it takes to eat lunch.

   Write a MC integration program to estimate the value of the following 10D integral:

$$I_2 = \int_0^1 dx_1 \int_0^1 dx_2 \cdots \int_0^1 dx_{10}(x_1 + x_2 + \cdots + x_{10})^2$$

   where the exact result is $155/6 = 25.8\dot{3}$

   Be sure to write your code so that there is a new random number seed used each time it is run (the program may request it from standard input, for example). Do not use a built-in random number generator. Here, we explore the impact of both re-running the code and of varying the number of MC evaluations of the integrand, or steps (MCS). Report three estimates for the integral and their absolute error ϵ for 1000 MCS. Make a plot of ϵ for three runs, and their average, as a function of MCS, for a maximum of 10^7 MCS. Overlay on this graph the expected behaviour, $\epsilon \sim n^{-1/2}$. Your graph should look something like Fig. 2.

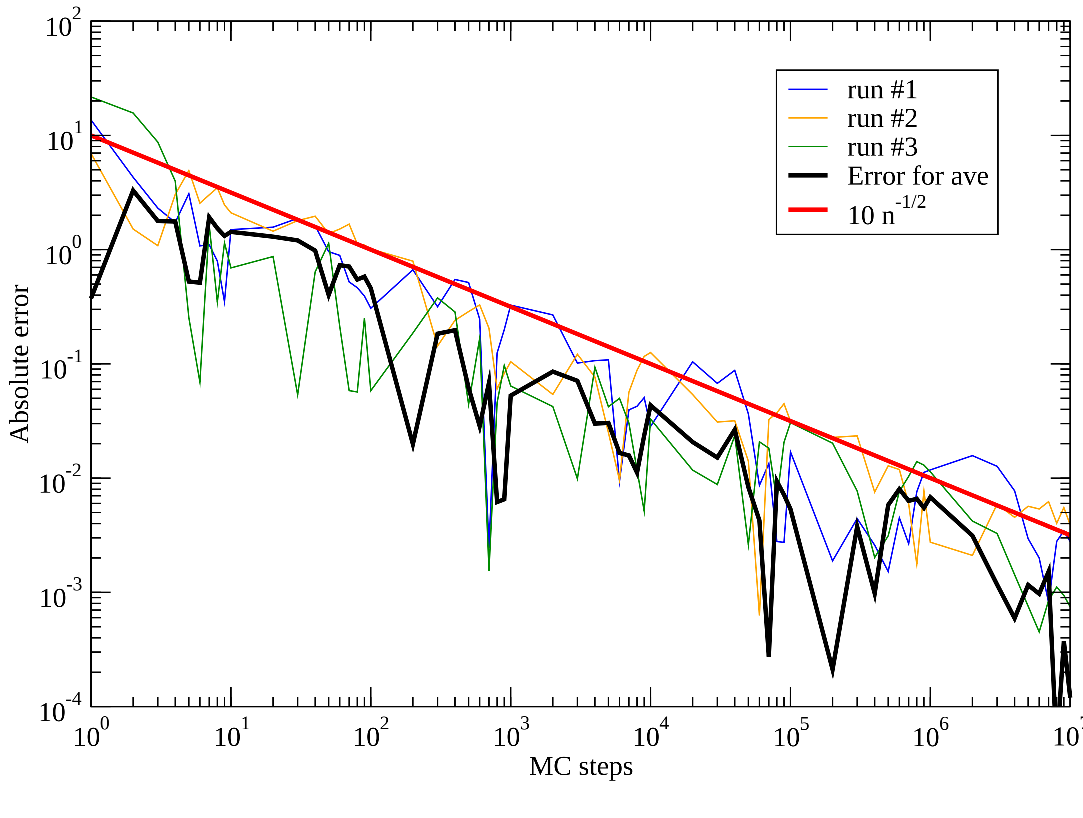

Figure 2: Absolute error for MC integration as a function of MC steps for three independent runs and their average. Shown is the expected scaling behaviour for the error.