PHYS/ENGP 3170/6170
Computational Physics and Engineering
Spring 2021

Assignment #4: Random Numbers and Monte Carlo
Due: Wednesday, Mar 17 11:59 pm

Readings: LPB 4.1-4.4; 5.21-5.22; 5.14-5.15; 5.17

(1) The code rand1.py, included with this assignment, generates N+1 random numbers in the interval [0,1] using the classic "drand48" linear congruent method, and saves pairs of consecutive numbers (r_i,r_{i+1}). To reduce the data to be stored and plotted, a pair is only saved if both r_i and r_{i+1} fall into the interval [0,0.01]. Do the pairs visually seem to be randomly distributed? What happens if you increase N (you will likely need to zoom further in to avoid plotting too many points)?

(2) Now replace the drand48() method with RANDU(), defined as $r_{i+1} = 65539\ r_i \bmod 2^{31}$. Again inspect the output for visual randomness, and don't forget to see what happens when you increase N.

(3) Modify the code to compute the average of $Q=(r_i)^2(r_{i+1})^2$. Use all N pairs, not just those in the interval [0,0.01]. Do you get the same result with the drand48() and RANDU() generators? How does the result compare with the analytic answer you *should* obtain if the random numbers are independent and uniformly distributed between 0 and 1?

[Required for 6170 students only, extra credit for 3170 students] Examine how the difference between the computed average of $(r_i)^2(r_{i+1})^2$ and the analytic answer varies with N. How *should* the difference fall off with N if the sequence is uncorrelated?

(4) The code Walk.py, included with this assignment, implements a two-dimensional random walk, in which the x-component and y-component of each step are uniformly distributed in the interval [-1, 1]. The distance from the origin after N steps is averaged over 100 trials, for all N from 1 to 10000. Plot the average distance from the origin as a function of N, and compare with the analytical prediction, Eq. (4.20) in the textbook. Note that you're plotting the *average* distance, while formula (4.20) gives the *rms* distance; the two are proportional but not identical.

[Required for 6170 students only, extra credit for 3170 students] Change the code so that instead of uniformly distributed steps, you always take a step of 1.0 up, down, left, or right, each with probability 25%. Compare with the previous results.

(5) The code, int_10d.py, included with this assignment, uses Monte Carlo integration to evaluate the 10-dimensional integral of Eq. (5.89). The approximation to the integral is printed out as a function of the number N of random points used, where N is every power of 2 between 2^1 and 2^{16}. Check that the answer converges to the analytic expression,

$I=155/6$, and examine whether the decrease of the error with increasing N is consistent with your expectation. To confirm the behavior of the error with N, you will need to modify the code to go to larger values of N.