Lab Problems

1. Let's study the bounding method for sampling probability distributions.

   (a) Starting by modifying the `vetowell` program from Lecture 3, write a Monte Carlo program called `bound` to sample the position distribution of the infinite square well ground state, $p(x) = 2\sin^2(\pi x)$, $x \in [0, 1]$. Use the bounding veto method, with

   $$q(x) = x(1 - x)\,, \tag{67}$$

   as the bounding function. You will need to invert the cumulative distribution corresponding to $q(x)$ to do this; use any method you like (analytic inversion, Newton's method to find roots, etc.). Have `bound` output a histogram and make the arguments of `bound` be the number of Monte Carlo events and the number of bins in the histogram.

   (b) Plot the output of `bound` with 100000 Monte Carlo events and 100 bins on $x \in [0, 1]$. Compare to the plot of $p(x)$.

2. Write a Monte Carlo program called `mcfunc` that runs a Monte Carlo in two random numbers x, y that are both uniformly distributed on $[0, 1]$ and computes the distribution that is a function g of the random numbers x and y. Make `mcfunc` take in the arguments `g`, the function of x and y; `Nev`, the number of Monte Carlo events to generate; and `Nbins`, the number of bins in the resulting histogram. You will also need to determine the smallest and largest value of $g(x, y)$ to make the histogram; assume that the smallest value is 0 and the largest value of $g(x, y)$ occurs when $x = y = 1$.

   Apply `mcfunc` to the functions in Problem 5 of the homework:

   $$g(x, y) = x + y \tag{68}$$
   $$g(x, y) = xy \tag{69}$$

   and compare the plots of the resulting distributions to the results you found in Problem 5. Use `Nev` = 1000000 and `Nbins` = 100.

3. In this lab problem, you will write your own Markov Chain Monte Carlo simulating the radioactive decay of atoms with half-life λ. We would like to determine the number of atoms that decay in a time T. Write a function `decay` that takes the half-life, the time interval T, and the number of samples `Nsamp` to simulate as inputs.

   The function `decay` should implement the Markov Chain algorithm defined in Lecture 2. That is, it finds the time that the first atom decays, and if that is less than T, it finds the time the second atom decays, etc. It then records the number of decays that occurred in time T for that sample and inputs that into a histogram. This repeats for all of the `Nsamp` number of samples, and `decay` should return the histogram. Have the histogram extend to the integer closest to 10 times the half-life.

For a half-life of $\lambda = 1$s and a time interval of $T = 5$s, plot the resulting histogram from `decay`. Use `Nsamp` = 100000. What is the mean and variance of the distribution? From the relationship of the mean and the variance, what can you conclude about the distribution?

4. In this problem, we will study a system called a *random walk*. A random walk is generated by starting at the origin and then taking a step of random size in the x and y dimensions, and then takes a random step from that point, etc., repeating numerous times. Random walks are models for Brownian motion, the erratic motion of particles suspended in fluid. (Einstein's Ph.D. thesis was on using Brownian motion to determine Avogadro's number.)

   (a) Write a program in Mathematica called `randomwalk` that generates points in a random walk. `randomwalk` takes an argument `Nstep` which is the number of steps in the random walk to take. Have `randomwalk` start at the position $(x, y) = (0, 0)$ and then take a step whose x and y coordinates are determined by choosing random numbers uniformly on $[-1, 1]$. Then, from the first step location, determine the next step location by displacing the current x and y coordinates by an amount determined by choosing random numbers on $[-1, 1]$, and continue, until you have generated `Nstep` steps. Have `randomwalk` return a table of the x and y coordinates of every step.

   (b) Using `randomwalk`, plot its output for a walk with 10000 steps. Run this a few times; every new time you run `randomwalk`, different random numbers are used, so you find a different random walk. Describe general features you see in the random walks.

   (c) On average over a large ensemble of random walks, what should the average x and y locations be after 1 step? Use your `randomwalk` program to determine the average distance of the first step from the origin; call this d_1. Find d_1 from 100000 random walks of a single step. How close is your result for d_1 from `randomwalk` to the exact result

   $$d_1 = \frac{\sqrt{2}}{3} + \frac{\log(3 + 2\sqrt{2})}{6} \, ? \tag{70}$$

   (Not for credit, but can you derive this result?)

   (d) Now, use `randomwalk` to calculate the distance from the end of a random walk of many steps to the origin. Calculate the distance from the origin of random walks with 2^n steps averaged over 1000 different walks, with $n = 0, 1, 2, \ldots, 10$. Make a plot of the number of steps versus the distance from the origin. How does the average distance from the origin scale with the number of steps? That is, if I take N random steps, approximately how far from the origin should I expect to be?