PHY 480

Session 12

Random Number Generation:

1. In the constructor, I would put in statements that take in a seed, the number of random numbers, and the distribution. The seed and the number of random numbers would be private variables.

2. I plotted the uniformly distributed random numbers on in 2 dimensions where (x,y) = (first random number, second number).

3. The plots look appropriate (see histogram_uniform and histogram_gaussian). The uniform histogram looks generally flat and the Gaussian histogram has the general shape of a Gaussian distribution.

Random Walking:

1. We have $r_{rms} = \sqrt{\left\langle \Delta x^2 + \Delta y^2 \right\rangle} = 2\sqrt{\left\langle \Delta x^2 \right\rangle} = 2\sqrt{2/3}$

Since $\left\langle \Delta x^2 \right\rangle = \int_a^b x^2 dx/(b-a) = (a^2+ab+b^2)/3$ and $a = -b = \sqrt{2}$.

3. See random_walk_plot.

4. It scales as $R = r_{rms}\sqrt{N} = 2N\sqrt{2/3}$. Yes, I can reproduce the fact that $R \propto \sqrt{N}$. I got the pairs (N,R) = (10^4,475), (4x10^4,905),(16x10^4,1841) which confirms the scaling as multiplying N by 4 and 16 caused R to be multiplied by $2 = \sqrt{4}$ and $4 = \sqrt{16}$, respectively.

5. I would use at least $\sqrt{N}$ trials as that's what R is proportional too. I modified the code (see random_walk.cpp). Yes it agrees, see random_walk_length_plot. We expect to have a slope of 0.5 since $\ln R / \ln N = r_{rms}/2 +$ C. The slope calculated was 0.49.

Monte Carlo Integration: Uniform and Gaussian Sampling:

1. I would want to choose the limits to encompass several standard deviations of the Gaussian in order to integrate over a large percentage of the Gaussian. Probably 5 or 6 sigma.

2. I observe a linear fit with slope approximately equal to $-0.55 \approx -1/2$ on a log/log plot implying that error $\propto N^{-1/2}$ which agrees with the notes.

3. In agreement with the notes, the error continues to be proportionate to $N^{-1/2}$ regardless of dimension (For 3 dimensions I got a slope of -0.56 and for 10 dimensions I got a slope of -0.59, both of which are close to -1/2).

4. The integrand function should return $f(x)/w(x) = (x_1 + x_2 + \ldots)^2$ which the function integrand2 does. The results are better than for random sampling because with random sampling you might be evaluating the integral when it is very small a lot of times which is wasteful. Yes, $D = 100$ works. It only has a 10^{-4} which further illustrates how Monte Carlo integration scales the same regardless of dimension.

Monte Carlo Integration: GSL Routines:

2. I changed the integrand to e^{-x}. Plain gave an error of 627.3 sigma. Miser gave 602.8 sigma. Vegas warm-up gave 1378.4 sigma. Vegas final gave 2418.9 sigma.

C++ Class for a Random Walk:

1. See RandomWalk_test.dat

2. Some advantages are that you can hide all of the machinery that goes into generating the random numbers in the class definition. Additionally you can get x and y at each step by taking a step with attribute .step() and then calling .get_x( ) and .get_y( ). This saves you from having to continuously add the steps distances (x and y) up in the main function. One disadvantage is that you have less ability to vary the program in the main function since every thing is hidden. For example, if you want to change the probability distribution used to pick

how far one steps in the x and y directions you would have to go into the definition of the class to do so.

3. You could extend the class definition to incorporate more probability distributions to choose how far one steps in the x and y directions. You could also extend the class to include different methods of determining how large of a step to take and in what direction to take it such as picking the angle and then the distance that you travel at that angle (polar). You could even extend it to more that 2 dimensions by adding .get_z( ) and so on.