Exercises #8
due: Monday, April 19, 2004

Reading Assignment:

- For additional information on this week's material, please read:
  Giordano *Computational Physics* Chapter 7.1-7.5 (Random Systems) pp 157-174

- To prepare for next week, please read
  Rex and Thornton *Modern Physics*, Chapter 9 (Statistical Physics) pp 272–285
  Giordano *Computational Physics* Chapter 8.3 (The Monte Carlo Method) pp 213-215

1. Tests for random number generators

   (a) Test for sequential correlation of random numbers:
   Open the file *rantest2d.m* in the editor and familiarize yourself with its contents. The program is set up to test random numbers generated by Matlab and random numbers generated with a congruential method, a so-called "modulo" random number generator whose parameters you can adjust. As discussed in class, considering pairs of consecutively generated random numbers as x and y coordinates and plotting them in the x-y plane provides a test for correlations between random numbers.

   i. Before you run the program, identify the statements that set the state for the Matlab random number generator and that define the seed for the modulo generator. Run the program and compare the graphs for the Matlab generated numbers and the graph for the numbers generated by the congruential method. What are the plots showing? Note the parameters for a, m, c, and `seed` for the modulo generator. Why is it a problem that the parameters yield such a small number of points in the plain?

   ii. Change the parameters to $a = 1103515245$, $m = 2^{31}$, $c = 12345$ and note how the points are distributed in the plain.

   iii. Change the parameters to $a = 1277$, $m = 2^{17}$, $c = 0$ and note how the points are distributed in the plain.

   iv. Play with the parameters and see what happens if you change the value of the `seed` for the "bad" parameters.

   (b) Test for uniform distribution of random numbers

   i. Open the file *rantestbin.m* in the editor and familiarize yourself with its contents. Describe in a few sentences how the data are binned and prepared for comparison with the theoretical distribution. Type `help hist` to learn about Matlab's histogram function. Run the program and comment on the

deviations between theoretical and actual distribution. Run the program several times and note the values of the reduced χ^2. How do you interpret these values?
Hint: In order to get a better feeling for the relationship between the quality of the random numbers and the reduced χ^2 values, observe what happens if the distribution is definitely not uniform. Include a statement like `sample=0.8*sample` or `sample=sin(sample)` in your program before the sample is evaluated and note the resulting distribution and the χ^2 values.
Don't forget to remove this statement before you work on the next part.

ii. The program is set up to call the Matlab generator `rand` for random numbers in uniform distribution. Open the file *randmod.m* in the editor. This program generates random numbers with a congruential method as in (a) and with parameters that are set to give a poor result ($a = 65539$, $m = 32767$, $c = 17$, `seed`=17). Change the program *rantestbin.m* so that it calls *randmod* instead of *rand*. Run the program, note the value for χ^2, and comment on the deviations between theoretical and actual distribution. Since the seed fixes the sequence of random numbers, nothing will change if you run it again. In order to generate different sequences of random numbers, use the statement `seed=sum(100*clock)` in *randmod*. Run the program several times and note the values of the seed and the value of the reduced χ^2.

2. Random walker test for random number generators, is the cycle length long enough?

We found in class that the root mean square length r_{rms} of the random walk should depend on the number of steps N as $r_{\text{rms}} = \sqrt{N}$. In a simulation, this will not be the case if the random number generator cycles through, i.e. starts repeating its sequence of random numbers.

(a) Write a Matlab program that performs a random walk of length N in one dimension and calculates the square of the distance between the starting point and the end point. Run it several times for relatively short walks ($N < 10$) using the Matlab random number generator to convince yourself that your program works properly. For example, plot the position as a function of step number and make sure the displacement always has a magnitude of one.

(b) Extend your program to collect data for a number random walkers and determine the average and the standard deviation of the squared displacement.

(c) Extend your program to perform the simulation/evaluation in (b) for a number of different lengths N of the walk, between 10 and 5000, say, and plot the mean squared displacements of the walks. For the Matlab random number generator, you should find a straight line with slope one.

(d) Repeat (c) using the modulo random number generator with two different sets of parameters. Discuss your results.

3. Monte Carlo integration:

   (a) The file *MCintegration.m* contains a simple application of random numbers in uniform distribution. The area of a circle is estimated by randomly "dropping" points into a square circumscribing the circle and comparing the number of points that fall into the circle to the total number of points dropped. In this way an estimate for the value of π can be obtained. Run the program a few times and see what values of π are obtained. Double the total number of points N and run the program again a few times, does the value for π improve?

   (b) Describe a method to determine the error in π obtained by MC integration, implement it in your program, and obtain a value for π and its error.

Please don't forget to do the PreClass on the web before Wednesday, April 14.