Homework 2: Random Numbers, Monte Carlo, and Multidimensional Integration

Ivan F. Vaelerio
Department of Physics, Astronomy, and Applied Physics,
Rensselaer Polytechnic Institute
valeri@rpi.edu

02/25/2011

0.1 Assignment

1. Use your computer's default pseudo-RNG to generate a sequence $\{x_i\}_{i=1,N}$ of random numbers in the range between 0.0 and 1.0 (choose an appropriate value for N, such as 10,000 or more).

   (a) Plot x_i as a function of sequence number i. Comment on the plot. Does it *look* random and uniform?

   (b) Plot x_i as a function of x_{i+k} for a small k of your choice. What can you learn from this plot?

   (c) Compute the k^{th} moment of your distribution. Compare to the theoretical value and comment.

   (d) Calculate the auto-correlation between x_i and x_{i+k} for a few values of k. Comment and discuss.

   (e) Prepare a plot with *bins* to assess the uniformity of the sequence. Use a reasonable number of bins (10, for instance). What does this plot tell you about the sequence?

   (f) Using the information in (a)-(e), comment on the quality of your sequence of random numbers.

   (g) Repeat with a sequence of numbers obtained from random.org.

2. Choose one of the two following problems:

   - **Use random numbers to calculate the integral in five dimensions:**

   $$\int_0^1 dx_1 \int_0^1 dx_2 \int_0^1 dx_3 \int_0^1 dx_4 \int_0^1 dx_5 (x_1 + x_2 + x_3 + x_4 + x_5)^3 \quad (1)$$

   (a) Compare your result with the analytical result.

   (b) Repeat the calculation for a varying number of random numbers.

   (c) Study how the error behaves as you change the number of points. Plot this error and find if you can detect a $\sim 1/\sqrt{N}$ behavior of the error, where N is the number of points.

   (d) Compare the order of magnitude of your error with (numerically obtained):

   $$\sigma^2 = < f^2 > - < f >^2 \quad (2)$$

   where

   $$f = (x_1 + x_2 + x_3 + x_4 + x_5)^3 \quad (3)$$

   - **Random walk in 3D**

   (a) Repeat the 2D work presented in class for an equivalent 3D random walk.

(b) At each step, consider a constant step size (i.e. of length 1.0, thereby defining the unit of distance). This means you could use spherical coordinates with constant $r = 1$, while randomly picking values for azimuthal angle $\theta \in [0, 2\pi]$ and zenith angle $\phi \in [0, \pi]$. Feel free to use Cartesian coordinates if you prefer; but in that case make sure you normalize the displacement vector.

(c) Vary the total number N of steps and plot $\sqrt{< R^2 >}$ as a function of $\sqrt{N}$. $< R^2 >$ could be computed by averaging over, say, $n_{average} = 16$ different random walks (choose a different value for $n_{average}$ if you want). Comment on the general behavior of the $\sqrt{< R^2 >}$ versus $\sqrt{N}$ plot.

(d) Plot a few trajectories and discuss their shape.

Make sure your report is self-contained with sufficient details and clear plots. Feel free to add listings of your code (or use pseudo-code). Collaborative work is allowed but each student will turn in an individual report.

Chapter 1

Pseudo-Random numbers

1.1 Algorithmic Considerations

We have various ways of testing the qualities produced by algorithms built for producing random numbers in computers. There are various algoirthms designed for testng whether these random numbers have any correlation with each other. One such way is by graphing the random data. This is done by having the points plotted as a function of index. Another way of determining if points produced by the computer's random number generator are mostly random is seeing if there is a correlation between a random number in a list and it another random number in a different index. If there is a correlation than we should be able to find some type of pattern. Computing the Kth moment can also provide us with some information as to how random the numbers are. This can be done by using:

$$\frac{1}{N}\sum_{i=1}^{N} x_i^k \approx \frac{1}{k+1} \tag{1.1}$$

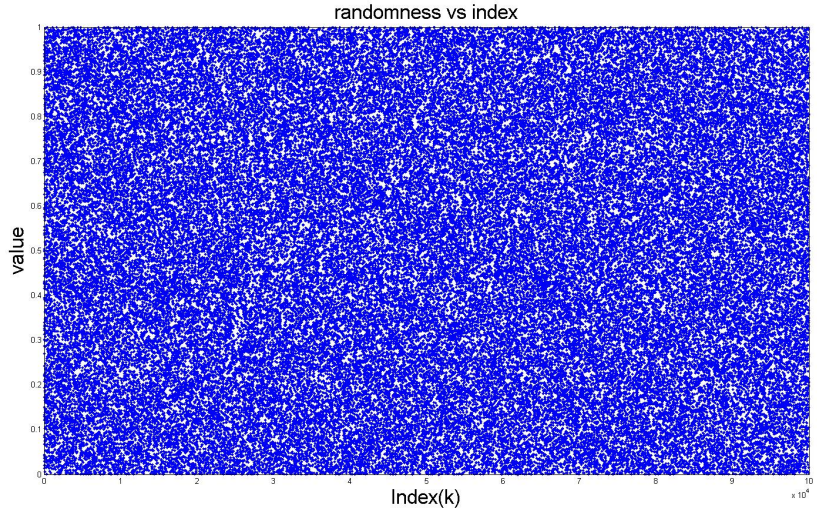
where N is the number of random points being used. We can also calculate the autocorrelation as follows:

$$C(k) = \frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k} \approx \frac{1}{4} \tag{1.2}$$

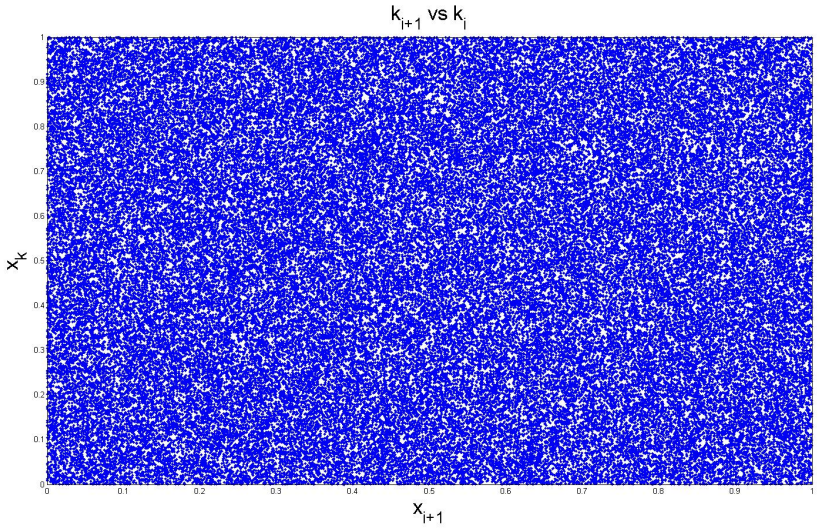
If we apply these mathematical definitions to the random numbers, and these numbers are mostly random, then we should obtain results very close to the predicted ones above. But how does this compare with real random numbers? Applying these tests to a real set of random numbers allows us to objectively see if our pseudo-random numbers obtained from computers are close enough to real random sets for our purposes.

1.2 Implementation and Results

The code that was implemented to makes these tests involved creating the numbers on an array and then calling theses stored values as needed. Plots that involved the correlation of the random numbers and the historgrams were done on MatLab. Below are various plots of the data obtained from the computer's pseudo-random number generator:



There doesn't seem to be any correlation between the random number produced and it's index. The following graph shows a plot of a random number and it's neighbor:

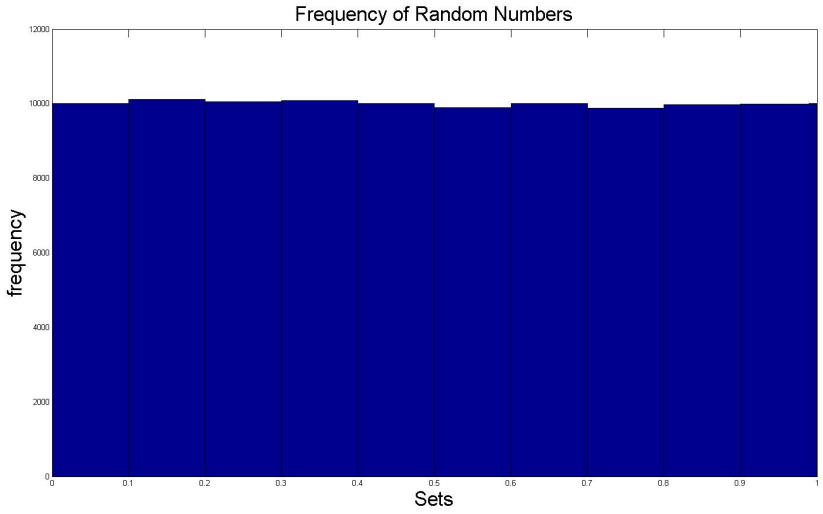


Similar to our first plot, there seems to be no discernible pattern relating the

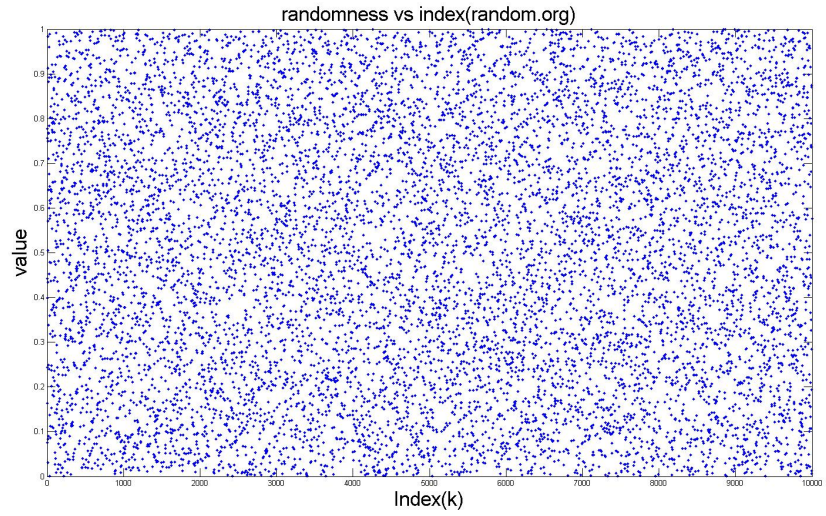
random numbers with their neighbors. There were approximately 100,000 numbers used for these graphs.
The Kth moment was calculated by implementing the summation disccused in the previous section. The calculated value was 0.199649. This number is very close to the analytical value 0.2, and it seems to demonstrate the computer's ability to produce pseudo-random numbers.
The correlation was calculated for values of k from 0 to 6. It seems that there is no clear improvement between the k values of 1 to 6. On the other hand when using value of 0 for k, the value jumps from 0.25 to 0.33. Various trials of this calculation yield very similar results. Below is a table showing the results:

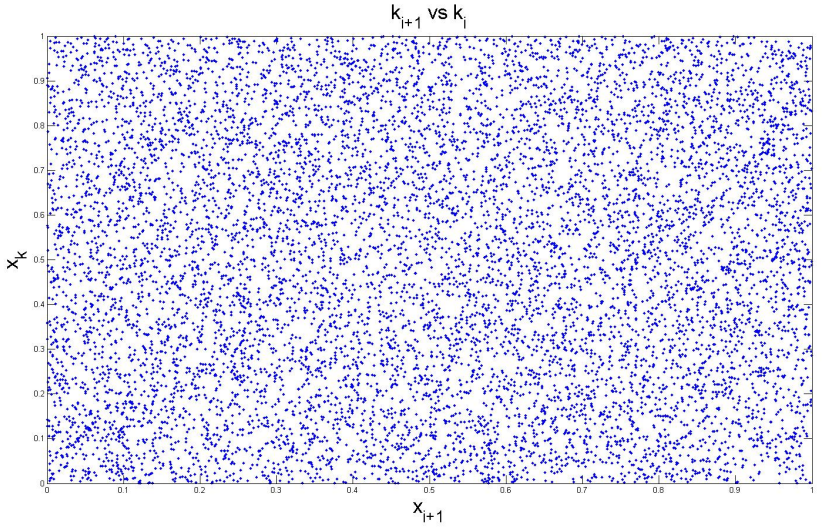
K	Calculated autocorrelation
0	0.333613
1	0.250565
2	0.250166
3	0.25041
4	0.249865
5	0.250541

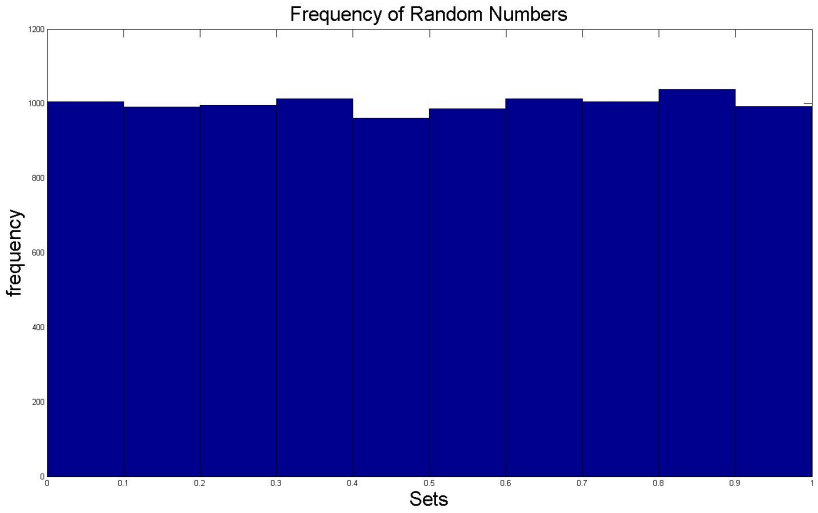
The following plot shows the frequency with which sets of random numbers appear on the built array:



The bins seem to have very close frequencies, again demonstrating the randomness of the numbers being generated. The followin graphs represent the uniformity and randomness from a set of numbers obtained from a website called Random.org. These graphs, which used about 10,000 points, seem to show similar randomness as the ones produced by the website:


randomness vs index(random.org)
value
Index(k)


k_{i+1} vs k_i
x_k
x_{i+1}



1.3 Discussion

It is worth noting the similarities between the graphs obtained from the random number generator from the computer and the array of numbers from Random.org. This adds to our confidence of our own computers to generate neccessary random numbers. It is also important to keep in mind that these numbers, despite their proximity to real random characteristics, still fall short of the tittle of true random numbers. Nevertheless they seem to suffice for many applications.

Chapter 2

Monte-Carlo

2.1 Algorithmic Considerations

Random numbers are not only useful for testing theorems, but they are also very important for computational reasons. Take for example the integral given for this problem:

$$\int_0^1 dx_1 \int_0^1 dx_2 \int_0^1 dx_3 \int_0^1 dx_4 \int_0^1 dx_5 (x_1 + x_2 + x_3 + x_4 + x_5)^3 \tag{2.1}$$

Using Monte-Carlo method, we are able to solve this integral without requiring complicated long summations or spending long hours(or years according to professor Meunier). Monte-Rarlo method requires us to sample the function using random numbers. For our integral we know that Monte-Carlo method is as follows:

$$\int_a^b f(x)dx \simeq (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) = (b-a) < f > \tag{2.2}$$

Above is the implementation of the Mean Value Theorem. By using random numbers in x_i, we are able to obtain a good enough value for the integral in question.

2.2 Implementation

The following pseudo-code was implemented to calculate the problem's integral:

x = 0;
while $i < size$ **do**
  pick random numbers for all the variables;
  x = x + $(x_1 + x_2 + x_3 + x_4 + x_5)^3$;
end
value of integral = y/size;

Algorithm 1: Finding the value of the integral using Monte-Carlo method.

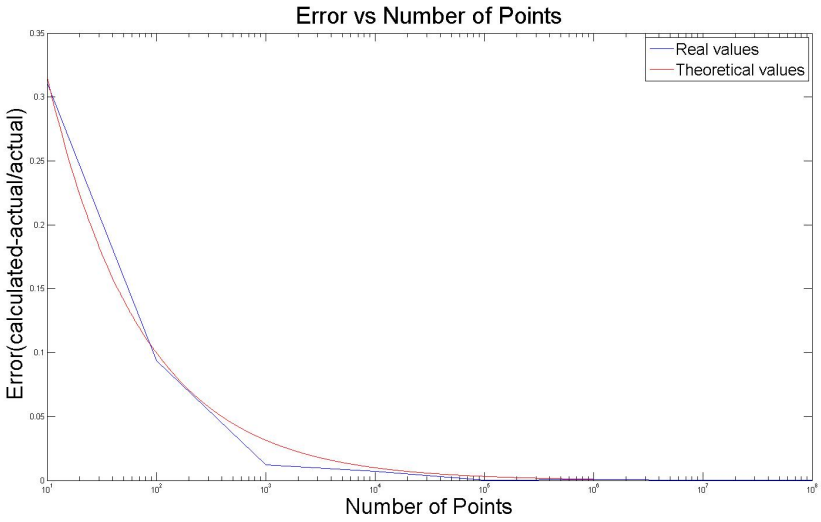
Where x_1, x_2, x_3, x_4, and x_5 are all chosen using a random number generator from the computer. This algorithm is also used when calculating the deviation of the obtained values. The value obtained is then compared using known online programs(such as Wolfram Alpha) to compute these values at higher accuracy.

2.3 Results

The integral was first calculated using 10^6 random numbers for each variable. The value obtained is 18.7402. The value obtained from Wolfram Alpha is 18.75. Below is a table of the values calculated for various numbers of random numbers per variable:

Number of random numbers per variable	Value of the Integrand
10^1	12.9113
10^2	16.9966
10^3	18.5173
10^4	18.6129
10^5	18.7475
10^6	18.7402
10^7	18.7498
10^8	18.7497

Beyond 10^8 random numbers the computer takes considerably more time.
The following is a graphical representation of the behaviour of the data points compared to an expected function. It is predicted that the error decreases with rate $1/\sqrt{N}$:



From the previous graph we can see that the obtainedl values for the error follow the analytical representation very closely. σ, which is the calculated error, is found to be 13.5239.

2.4 Discussion

It seems that the pseudo-random numbers produced by computers are "random" enough for our own purposes. It's ability to find the value of an otherwise impossible to solve integral to at least one decimal place is amazing. Of course errors must be taken into considerations. One interesting result is the fact that it does not matter whether we use the same numbers to compute the mean value of the function, or if we use completely different random numbers for $< f^2 >$ and $< f >^2$, the result is still very much the same. Indeed if the pseudo-random numbers produced by the computers had notable correlation, the values would matter if we were to use different random numbers. Such results gives us confidence with these random number generators.

2.5 References

Wolfram Alpha(online calculator)