Homework 4

Due 2019 October 25

Report instructions

Please upload *one* PDF file to Gradescope (Entry code: ME62YG).

For this homework you will implement code (on PrairieLearn) for a pseudo-random number generator (RNG) and assess the quality of a stream of random numbers from a particular RNG. You will also study the technique of importance sampling. Random numbers and importance sampling are essential ingredients of Monte Carlo simulation techniques and in the next homework, you will modify your MD code to do Monte Carlo simulations.

Introduction

Uniform Random Number Generator

On PrairieLearn, you are going to write (and test) a RNG that uses the linear congruent generator (LCG) algorithm. An LCG generates a random number X_{n+1} by taking the current random number X_n and performing the following arithmetic,

$$X_{n+1} = (aX_n + c) \mod m,$$

where m is the upper limit of possible values of X_n, $a < m$, and $c < m$; $\mod$ denotes the modulus. This algorithm requires choosing a number X_0 to start the stream, called the seed.

The quality of the LCG algorithm is very sensitive to the choice of these parameters. For instance, your period can never be longer than m.

Gaussian Random Number Generator

If you have a stream of uniformly distributed random numbers, it is possible to apply a transformation to get a stream of random numbers that obeys some other distribution.

In simulations of physical systems at thermodynamic equilibrium, it is useful to have random numbers drawn from a Gaussian distribution. There are some standard methods to accomplish this, and you should use textbooks and the internet to learn about these algorithms.

After implementing a Gaussian RNG, you will verify that it works by plotting a histogram. The correct normalization of the histogram is not obvious. If you have N random numbers, and your histogram contains N_B bins over a range $\pm L$, you should multiply by the normalization factor $\frac{N_\mathrm{B}}{2LN}$.

Testing Random Number Generators

To test a RNG that claims to produce uniformly distributed random numbers on the interval [0,1), we can test the uniformity of the distribution by creating a histogram of the stream of random numbers. We create a series of evenly-spaced bins between [0,1) and we generate a stream of random numbers,

binning each value appropriately. If the RNG stream is indeed uniformly distributed, we should observe the same number of random numbers in each bin.

Of course, if we actually try to do this empirically, we will find that not every bin gets the same number of counts. Instead, there will be fluctuations that cause some bins to get more counts than others. On the other hand, a RNG stream that is not truly uniform will also lead to a histogram with unequal counts in each bin. We want to distinguish between non-uniformity that occurs because of statistical fluctuations (i.e., having only a finite sample size) and non-uniformity due to a broken random number generator. We use the following metric to distinguish between these two cases:

$$\chi^2 = \sum_i (N_i - n_i)^2 / n_i$$

where N_i is the number of counts in bin i and n_i is the expected number of counts in bin i (the total number of points divided by the number of bins $N_{\rm B}$ for our case of the uniform distribution).

If $n_i > 5$ is true for all i, the χ^2 statistic approaches the eponymous chi-squared distribution with $N_{\rm B} - 1$ degrees of freedom. The statement of the goodness-of-fit can be simplified to this: If this χ^2 value is greater than some critical value, then with some quantifiable confidence this indicates that the generator of this stream is not what it is claimed to be (i.e., a uniform distribution of $x \in [0, 1)$). We can try solving for these critical values numerically by integrating the probability distribution function for the chi-squared distribution, or for degrees of freedom up to 100 just use the tabulated values provided by NIST. For this homework, let us choose the 90% confidence level for the hypothesis that the generator is not broken (random). Critical values of χ^2 should be taken from the column labeled "0.10".

One can also bin numbers in more than 1 dimension. For example, for 2 dimensions one could take pairs of numbers and bin them onto a grid. This is useful because it will expose correlations between successive numbers in the series. This is also a serious problem, because we generally use a RNG stream with the assumption that successive numbers are independent of one another.

You will write functions that compute the χ^2 metric also for histograms in two and three dimensions. Note that the χ^2 test becomes increasingly stringent in higher dimensions, and RNGs judged suitable for research purposes are generally required to pass the χ^2 test in 10 dimensions. You may find yourself working with degrees of freedom greater than 100 (off the NIST table mentioned above), in which case you will have to calculate the values yourself by numerical integration or use the critical value calculator found at Chi Square Calculator.

For 90% confidence level, you should set the "probability" parameter to 0.9. (Note: Do not set it to *0.10* as you did above, where it represented the upper tail integral.).

Importance Sampling.

You will work to compute the integral $I = \int_{-\infty}^{\infty} dx \, \frac{\exp(-x^2/2)}{1+x^2}$ which cannot be done analytically.

First, you will estimate the value of the continuous integral by performing a summation of the function at a discrete set of grid points: $I = \int_{-\infty}^{\infty} dx \, f(x) \approx \frac{1}{\Omega} \sum_{i=1}^{N} f(x_i)$, where Ω is the appropriate normalization involving N and L.

Second, you will compute this integral using Monte Carlo with importance sampling. Given an integral $\int_{-\infty}^{\infty} f(x)dx$, previously we will have to choose a bound L, and discretize the x-axis with N evenly spaced bins. There are usually two problems with this approach. Firstly, a good L or N is not always known beforehand. Secondly, if the function $f(x)$ is known to have sharp peaks somewhere, naturally one would need more bins around these areas, but it would be expensive to increase the density of bins everywhere.

The idea of importance sampling is that if we want to evaluate $I = \int_{-\infty}^{\infty} dx\, f(x)$ we can rewrite the integral as $I = \int_{-\infty}^{\infty} dx\, p(x)\frac{f(x)}{p(x)} = \int_{-\infty}^{\infty} dx\, p(x)g(x)$ where $p(x)$ is a probability distribution (i.e., positive for all x and normalized such that $\int_{-\infty}^{\infty} dx\, p(x) = 1$) and $g(x) = \frac{f(x)}{p(x)}$ is the estimator.

Using statistics, one recognizes that

$$\int_{-\infty}^{\infty} \frac{f(x)}{p(x)} p(x)dx = E\left[\frac{f(x)}{p(x)}\right],$$

where $E\left[\frac{f(x)}{p(x)}\right]$ is the expectation value of $\frac{f(x)}{p(x)}$ under the probability distribution depicted by $p(x)$. To actually calculate the expectation value, one uses the following estimate

$$E\left[\frac{f(x)}{p(x)}\right] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)},$$

where N random numbers $\{x_i\}$ are sampled according to the probability distribution depicted by $p(x)$.

Essentially, one turns a integration problem into a sampling problem. It is called importance sampling, because the variance is minimum when one chooses a $p(x)$ such that $p(x) \propto f(x)$, in the sense that we sample more points where $f(x)$ is the largest in magnitude and sample less points where $f(x)$ is small in magnitude. In reality, this is not practical, the next best thing one can do is to match the shapes and the locations of the peaks of $p(x)$ and $f(x)$ as closely as possible.

For our integrand $f(x) = \frac{\exp\left(-\frac{x^2}{2}\right)}{1+x^2}$, we will choose $p(x) = K\exp\left(-\frac{x^2}{2\alpha}\right)$, where K is chosen so that $p(x)$ is a valid probability distribution (i.e., normalized correctly).

The utility of doing this is that we know how to sample a Gaussian distribution using a Gaussian random number generator.

Thus, for Monte Carlo integration with importance sampling, we will sample random configurations distributed according to the function $p(x)$, and we will evaluate the integral I by computing the expectation value of the estimator $g(x)$: $I = \langle g(x)\rangle_{p(x)}$.

Your report

1. Discuss and show how long it takes before LCG(16,3,1,2) produces repeating values! Are all the numbers between 0 and 15 represented?
2. To show that your Gaussian RNG produces the correct distribution, generate a stream of 10,000 numbers with mean 0 and standard deviation 1 and histogram the values. If normalized

correctly, your histogram should match the function $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$. Produce a plot of your histogram and compare with the Gaussian function.

3. Generate a stream of random numbers on the interval [0,1) using your LCG function and report the chi-squared values in 1, 2, and 3 dimensions. Perform the same tests on the random.random() module that comes with Python. For each stream, state explicitly if the RNG passes the χ^2 tests. Plot the numbers of your PrairieLearn problems 4.8, 4.9, and 4.10 binned in **2** dimensions (for 4.8, make the streamed values into pairs; for 4.10, just use two of the three members in the triplet). Use the first 1000 points of the supplied data sets for plotting.
4. For your numerical integration, choose finite limits of integration $\pm L$ that you believe are a good approximation to the infinite limits and justify why. Also find the correct expression for Ω, the normalization.
5. Report the result of the numerical integration with your choice of parameters when using the rectangle rule.
6. Determine an expression for K that normalizes $p(x)$.
7. Using the functional forms of $p(x)$ and $f(x)$ given above, write down an analytic expression for the estimator $g(x)$. Also write down an expression for the estimator of the variance of I computed with Monte Carlo integration sampling the distribution $p(x)$.
8. Compute the value of the integral with Monte Carlo integration using your estimator for the mean and your Gaussian distribution of random numbers. Also report the variance of your answer using the appropriate variance estimator. Notice, that for different alpha's you are using different importance functions and hence will get a different variance. Graph the variance versus alpha and submit this with your code.
9. You should optimize your importance sampling by finding the parameter α that minimizes the variance. Choose at least 8 different values of α between 0 and 2 and report your estimate of the integral and the variance for each. Submit a plot of the variance as a function of α. Based on this plot, report your optimal choice of α.
10. If the variance were infinite for certain values of α, we would not know it from the results above. As a first test, run your simulation again with 4 times as many steps. If your variance is well defined, then we know that the variance should be effectively independent of how long you run it. If this turns out not to be the case, then you should worry that you have infinite variance! Submit a graph that graphs alpha versus (the ratio your calculated variance with 4 times as many steps to the original calculated variance). Also indicate where you have infinite variance.
11. Calculate the variance analytically, expressing your answer as a function of α. State the range of values of alpha for which the variance is infinite.
12. Now that you have located values of alpha where the variance is infinite, we will look to see how this manifests itself in the numerical data. Make a trace plot of $g(x, \alpha)$ versus sample number for a value of α that has infinite variance and a value of α that has finite variance. Describe the qualitative difference between these two situations.