PH307: Introduction to Numerical Analysis
Tutorial Sheet 5

Note: This tutorial sheet deals with random number generators and the Monte-Carlo methods. The program corresponding to the last problem will be graded by your TA.

1. Do a literature survey on random number generators (RNGs) using Google.

2. Do a literature survey on Monte Carlo simulations, and, in particular, Metropolis algorithm, using Google.

3. As discussed in the lectures, some of the most popular, and easy to implement, RNGs are the linear congruential generators based upon the recursion relation

$$\begin{aligned} R_k &= \{aR_{k-1} + b\} \pmod{m} \\ x_k &= R_k/m \end{aligned}$$

Above, the right hand side of the first equation is the remainder one obtains upon dividing $aR_{k-1} + b$ by m. Here, R_k's are the generated random integers, a is called multiplier, b is called increment, m is the modulus, and R_0 is called the seed. For the best results, seed should be an odd number. It is the set of random numbers $x_k \in (0, 1)$, which are normally used in calculations. Of various choices available for the values of a, b, and m, the "linear portable generator" (also called "minimal standard generator") defined by $a = 7^5 = 16807$, $b = 0$, and $m = 2^{31} - 1 = 2147483647$ is the most popular. However, note that on a computer with 32-bit integer size, its implementation is not straightforward because most of the time operation aR_{k-1} will lead to an overflow. Thus there are two possibilities:

   (a) Store all the integers in 64-bit words (unsigned long in C++, and integer*8 in Fortran)

   (b) Use Schrage factorization to compute $aR \bmod m$ as

$$aR \bmod m = \begin{cases} a(R \bmod q) - r[R/q] & \text{if it is } \geq 0, \\ a(R \bmod q) - r[R/q] + m & \text{otherwise} \end{cases}$$

   where, for the choice of a and m above, $q = 127773$ and $r = 2836$. Here $[R/q]$ is the integer division of R by q.

Write a subroutine implementing the linear portable generator using either of the strategies above to avoid the integer overflow. Your routine should allow the user to use an arbitrary seed, and its output should be x_k. Note that you will have to ensure that your routine "remembers" the value of R_{k-1} everytime it is called.

4. Use the Monte Carlo approach to compute the value of π by computing the ratio of the area of one fourth of a unit circle to that of a unit square. Recall that this approach was discussed in detail in class. Implement the following algorithm in a computer program:

(a) Generate a large number of (N, which should be user specified) of pair of random numbers (x, y), with $0 \leq x \leq 1,\ 0 \leq y \leq 1$. Each pair will be called a point.

(b) Figure out how many of these points lie on or inside the unit circle centered at the origin. That is, how many of these points satisfy the inequality $x^2 + y^2 \leq 1$. Let this number be N_1.

(c) Rules of probability tell us that for $N \to \infty$, $N_1/N \to \pi/4$. Thus you should get an estimate for π by computing $4N_1/N$. Note that with increasing N, this ratio will approach the true value of π.

(d) To generate a random number in Fortran 90 you can either use the internal subroutine "random_number" as: "call random_number(x)", which will assign a random number to the real variable x. Alternatively, you can use the RNG that you have written above in problem 3.

(e) Plot the results you obtain as a function of N, where N is the total number of random numbers that you use in the calculation. Vary N from $N = 10$ to $N = 1000000$.

5. Compute the following integrals by the straightforward Monte Carlo method, i.e, by just averaging over the sampled random numbers, without applying a Metropolis-type technique

(a) $\int_0^{\pi/2} \sin x \, dx$

(b) $\int_0^1 x^2 \, dx$

(c) $\int_0^2 e^{-x} dx$

(d) $\int_0^1 \frac{dx}{1+x^2}$ (exact value $= \pi/4$)

(e) $\int_1^2 \frac{dx}{x}$ and compare with the exact value

(f) $\int_0^{\pi/2} \frac{dx}{\sin^2 x + \frac{1}{4}\cos^2 x}$ (correct answer π)

(g) $\int_0^1 \int_0^1 x^2 y^2 \, dx \, dy$

Vary the values of N as in the previous problem and again plot your results as a function of N for various integrals.

6. Compute the integral $I = \int_0^1 x^2 \, dx$ by the Metropolis method. For this, rewrite the integral as $I = \int_0^1 g(x) w(x) \, dx$, where the weight function is defined by $w(x) = (e^{x^2} - 1)$, and $g(x) = \frac{x^2}{(e^{x^2}-1)}$. Now, using the central-limit theorem, the integral can be written as $I = Z\langle g(x)\rangle_w$, where $Z = \int_0^1 ((e^{x^2} - 1) \, dx = 0.46265167$, and $\langle g(x)\rangle_w$ is the average of the function $g(x)$, with respect to the weight $w(x)$, for $x \in (0, 1)$. It is $\langle g(x)\rangle_w$ which you have to compute using the Metropolis algorithm. Start your calculations with a randomly chosen initial value of $x = x_0$ and then change its values in the steps $\Delta x_i = \Delta(2\eta_i - 1)$, where $\Delta = 0.4$ and $\eta_i \in (0, 1)$ is also a random number. Chose sampling points according to the Metropolis criteria. In order to reduce the correlation among the sampled points, accept only one in $M(= 15)$ points. Compute the integral

by ranging the total number of sampled points from $N = 10$ to $N = 100000$. Note that in order to get N sampling points, you will have to make $N \times M$ choices. User of your program should have the choice of using either the standard RNG which comes with the compiler, or the RNG written by you as per problem 3.