COMPUTER PROJECT 1

Revision as of Monday, November 9th, 2020 at 06:32:28 AM

Due: Wednesday, November 18, 2020

The general theme of this computer project is random number generation on a computer. This is an important problem with a wide range of applications: Monte Carlo simulations, game-playing, cryptography, statistical sampling, cybersecurity, statistical modeling, evaluation of multiple integrals, and computations in statistical physics—to name a few. Specifically, the project concentrates on the problem of *uniform* (pseudo-) random numbers generation, with "uniform" understood as "*continuous* uniform" sampled from the interval $[0, 1)$. As was mentioned in class, a large family of statistical distributions (e.g., exponential, normal, etc.) can be obtained from the uniform distribution merely by means of a certain transformation of variables. Hence, having a *reliable* and *efficient* uniform random number generator (RNG) is at the foundation of the entire field of computer (or software) random number generation. As trivial as the subject may seem (what can possibly be easier than making up a nonsensical sequence of numbers?), random number generation is anything but trivial, and designing a "good" RNG is an art that requires a great deal of expertise in mathematics, statistics and computer engineering. The project focuses on two famous uniform RNGs: one "good" one, and one "not-so-good" one. The "not-so-good" one is the so-called Middle-Square RNG, and it is due to John von Neumann (1951), the famous scientist and one of the key figures behind the Manhattan project. The "good" one is due to Derrick H. Lehmer (1949), the famous number theorist and one of the founding fathers of computational number theory. Lehmer's RNG has become the basis for many of the RNGs in use today.

By and large, any software RNG that outputs a series $R_0, R_1, R_2, R_3, \ldots$ of independent numbers uniformly distributed on the interval $[0, 1)$ involves the following two steps. The first step is to obtain a stream of "unpredictable" integers $X_0, X_1, X_2, X_3, \ldots$ by iterating the recurrence $X_{k+1} = F(X_k)$, $k \geqslant 0$, where $F(x)$ is a given one-way function ("one-way" means the function is easy to evaluate for any appropriately specified argument, but is very difficult to work out the argument that yields a given output). It is assumed here that X_0 is a given number known as the "seed" number (or simply the "seed" for short). The integers $X_0, X_1, X_2, \ldots$ all come from a finite set $\{0, 1, 2, \ldots, m-1\}$ of $m \geqslant 1$ successive integers starting from zero. Note that unless the function $F(x)$ is carefully designed the integers X_k's need not actually span the entire set $\{0, 1, 2, \ldots, m-1\}$. If the integers X_k do span the entire set $\{0, 1, 2, \ldots, m-1\}$, then the RNG is said to have the full period; otherwise, the RNG is said to have only a partial period. The second step consists in scaling (or standardizing) each of the integers $X_0, X_1, X_2, X_3, \ldots$ into the unit interval $[0, 1)$ by dividing each X by m, i.e., computing $R_k = X_k/m$ for all $k \geqslant 0$. If the RNG is good, the R_k's will "appear" as though they are randomly and uniformly distributed between 0 and 1 (of course the "randomness" and "uniformity" are no more than an optical illusion, but this illusion may be sufficiently convincing for practical purposes, if $F(x)$ is designed properly).

Consider first the so-called Middle-Square RNG proposed by John von Neumann (1951). The idea of the Middle-Square Method is deceptively simple and is best explained via an actual example. Suppose we start off with a seed of $X_0 = 9\,876$ assuming that $m = 10\,000$, so the "unpredictable" integers $X_0, X_1, X_2, X_3, \ldots$ all come from the set $\{0, 1, 2, \ldots, 9\,999\}$. To get X_1, the seed is first squared (yielding $X_0^2 = 97\,535\,376$) and then the middle four digits of the result are used as X_1, i.e., $X_1 = 5\,353$. The next "unpredictable" integer, i.e., X_2, is obtained in the same fashion from X_1: the latter integer is first squared (yielding $X_1^2 = 28\,654\,609$) and then the middle four digits of the result are used as X_2, i.e., $X_2 = 6\,546$. The process continues until the stream $X_0, X_1, X_2, X_3, \ldots$ is as long as necessary. Once all the integers are generated, they are each standardized by division by $m = 10\,000$. For example, $R_0 = X_0/m = 9\,876/10\,000 = 0.9876$, $R_1 = X_1/m = 5\,353/10\,000 = 0.5353$, $R_2 = X_2/m = 6\,546/10\,000 = 0.6546$, and so on. Should it occur that say X_k^2 for some k is shorter than eight digits, it is padded with as many leading zeros as necessary to make it eight digits long. While it is easy to see that von Neumann's (1951) RNG is definitely based on the recurrence $X_{k+1} = F(X_k)$, one may find the function $F(x)$ used by the Middle-Square RNG to be somewhat problematic to express as a mathematical formula (it is simple to compute though). Now that you understand the

way the Middle-Square RNG operates, do the following:

(a) Implement the Middle-Square RNG in `R` as a separate function named `mid_square_rng`. More specifically, make the function take two input arguments: `N` followed by `X0`, where `N` denotes the desired length of the random sequence $R_0, R_1, R_2, \ldots$ to be generated, and `X0` stands for the seed. Your function must return the random sequence $R_0, R_1, R_2, \ldots, R_{N-1}$ assuming that $m = 10\,000$.

Submit your script.

(b) Set the seed to $X_0 = 1\,010$ and use your `mid_square_rng` to generate a series of $N = 20$ uniform $[0, 1]$ random numbers. Report the obtained series.

(c) Set the seed to $X_0 = 6\,100$ and use your `mid_square_rng` to generate a series of $N = 20$ uniform $[0, 1]$ random numbers. Report the obtained series.

(d) Set the seed to $X_0 = 3\,792$ and use your `mid_square_rng` to generate a series of $N = 20$ uniform $[0, 1]$ random numbers. Report the obtained series. Comment briefly on the obtained series.

(e) Explain briefly why if the initial value X_0 is selected from the set $\{0, 1, \ldots, 9\,999\}$ it is impossible to ever have X_k^2 more than eight digits long, for any $k \geqslant 1$.

(f) Based on the series you generated in part (b) explain briefly why the Middle-Square RNG is not a reliable source of random numbers.

(g) The series you generated in part (c) is an example of yet another problem that the Middle-Square RNG may exhibit when used as a source of random numbers. What is that problem?

Let us now turn attention to a better, more reliable RNG, viz. one proposed by Lehmer (1949), which has become probably the most popular RNG, and remains in heavy use even today. Lehmer's RNG relies on the recurrence $X_{k+1} = (aX_k + b) \pmod m$ where $m \geqslant 1$ is again a given integer referred as the *modulus*, $0 \leqslant a < m$ is also preset integer referred to as the *multiplier*, and $0 \leqslant b < m$ is referred to as the *increment*, and is set in advance as well. The "$\pmod m$" part guarantees that all X_k's will be confined to the set $\{0, 1, 2, \ldots, m-1\}$. The recurrence $X_{k+1} = (aX_k + b) \pmod m$ gives rise to a whole family of RNGs known as the *linear congruential* RNGs. Lehmer's original generation method had $b = 0$, although he mentioned $b > 0$ as a possibility. As was discussed in class, Lehmer's RNG is generally better than von Neumann's RNG, but not necessarily. It depends on the particular choice of X_0, a, b and m.

(a) Implement Lehmer's RNG in `R` as a separate function named `lehmer_rng`. More specifically, make the function take five input arguments: `N`, followed by `m`, followed by `a`, followed by `b`, followed `X0`, where `N` denotes the desired length of the random sequence $R_0, R_1, R_2, \ldots$ to be generated, `m` is the modulus, `a` is the multiplier, `b` is the increment, and `X0` stands for the seed. Your function must return the random sequence $R_0, R_1, R_2, \ldots, R_{N-1}$.

Submit your script.

(b) Set $X_0 = 1$, $m = 16$, and $b = 1$. Use your `lehmer_rng` function to generate a series of $N = 16$ uniform $[0, 1]$ random numbers for $a = 3, 4, 5, \ldots, 15$. Visualize the series using the `radarchart` function: make the contours represent the value of X_k and the radial numbers represent the serial number of X_k in the sequence. For what values of a is the period of the series the largest?

Submit your radarcharts along with your analysis.

(c) Set $X_0 = 1$, $m = 16$, and $a = 5$. Use your `lehmer_rng` function to generate a series of $N = 16$ uniform $[0, 1]$ random numbers for $b = 2, 3, 4, \ldots, 8$. Visualize the series using the `radarchart` function: make the contours represent the value of X_k and the radial numbers represent the serial number of X_k in the sequence. For what values of b is the period of the series the largest?

Submit your radarcharts along with your analysis.

(d) Set $X_0 = 6$, $m = 100$, $a = 21$, and $b = 1$. Use your `lehmer_rng` function to generate a series of $N = 20$ uniform $[0, 1]$ random numbers. Report the obtained series, and briefly comment on it.

(e) Set $X_0 = 1$, $m = 2^{11}$, $a = 1\,229$, and $b = 1$. Use your `lehmer_rng` function to generate a series of $N = 5\,000$ uniform $[0, 1]$ random numbers. Construct a histogram of the obtained series. Feel free to choose the bins as you please. What does the shape of the histogram suggest as to the statistical properties of the obtained random series? Submit the histogram.

(f) For the random series you generated in the previous part create a scatterplot showing R_{k+1} as a function of R_k. This kind of a plot is a standard technique used in statistics to understand the nature of inter-dependence or correlation (if any) between successive elements in the sample. What conclusion can you reach from the scatterplot about the series you generated in part (e)? Submit the scatterplot.

(g) Repeat parts (e) and (f) with $X_0 = 1$, $m = 244\,944$, $a = 1\,597$, and $b = 51\,749$. Does the new set parameters improve the quality of Lehmer's RNG? Briefly justify your answer.

(h) Repeat parts (e) and (f) for $X_0 = 1$, $m = 2^{31}$, $a = 2^{16} + 3$, and $b = 0$. For this choice of parameters Lehmer's RNG is known as RANDU, an RNG proposed by IBM in the 1960s. Would you say RANDU is a good RNG? Briefly justify your answer.

(i) For the series you generated in the previous part construct a 3D scatterplot showing R_{k+1} as a *bivariate* function of R_k and R_{k-1}. This kind of a plot offers a deeper insight into the correlation structure of the sampled data. What does the 3D scatterplot suggest as to the quality of RANDU? Briefly justify your answer.

References

Lehmer, D. H. (1949). Mathematical methods in large-scale computing units. In *Proceedings of the Second Symposium on Large-Scale Digital Calculating Machinery*, pages 141–146, Harvard University. Harvard University Press, Cambridge, MA.

von Neumann, J. (1951). *Various Techniques Used in Connection with Random Digits*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, pages 36–38. U.S. Government Printing Office, Washington, D.C.