Chapter 2

Random Number Generation

1. Introduction

- Computers generate random number for everything from cryptography to video games and gambling.
- Random numbers are fundamental to MC simulation.
- There are two categories of random numbers
  - ✓ True random numbers
  - ✓ Pseudorandom numbers

1. Introduction

- True random numbers:

- The computer measures some type of physical phenomenon that takes place outside of the computer.
- For example, the computer could measure the radioactive decay of an atom. According to quantum theory, there's no way to know for sure when radioactive decay will occur, so this is essentially pure randomness.

1. Introduction

- Pseudorandom numbers:

- They are alternative to "true" random numbers. A computer could use a seed value and an algorithm to generate numbers that appear to be random, but that are in fact predictable.
- The random numbers that we use on computer are not random at all. There are generated by entirely deterministic algorithms. They should be called pseudo-random numbers.
- The pseudo-random numbers should be behave like random numbers. We will treat them as if they were genuinely random.

What to Use: Random or Pseudo-Random?

- If you're playing a video game, it doesn't really matter whether the events that occur in that game are cased by "true" random numbers or pseudorandom numbers.
- On the other hand, if you're using encryption, you don't want to use pseudorandom numbers that an attacker could guess.

Random vs Pseudo-Random vs Quasi-Random

- Truly Random
  - Exhibiting true randomness
- Pseudorandom
  - Appearance of randomness but having a specific repeatable pattern
- Quasi-random
  - Having a set of non-random numbers in a randomized order

2. Methods for Random Number Generation (RNG)

- A central part of every stochastic simulation algorithm is a generator of random numbers.
- How a computer can actually generate a random number？ Where does this randomness come from.
- If it’s just a piece of computer code, isn’t it possible the numbers could be predictable?
- Generating good uniform random variables is technically complex.

2. Methods for RNG

- We don't intend, at least in this version of the lesson, to talk about the topic of RNG in very much detail.
- However, because MC methods rely mostly on being able to generate random numbers (often with a given PDF), it is important to mention that having a good RNG is important to guarantee the quality of the output of MC methods.
- We will cover:
  - What you need to know as a user.
  - Background information to make you better informed.

2. Methods for RNG

General considerations in construction of RNG:

- ✓ Period --- there should be a sufficiently long period before repetition occurs;
- ✓ Reproducibility --- the method should be seedable, to allow an experiment to be reproduced;
- ✓ Efficiency --- quick to run, minimal storage;
- ✓ Portability --- if the method is implemented on a different system, it should produce the same results;
- ✓ Randomness --- statistical test should be satisfied;
- ✓ Uniformity --- statistical test should be satisfied;

……

2. Methods for RNG

- **The Mid-Square Method**: (von Neuman)

$$x_0 = 0.9872,\ x_0^2 = 0.97\ 5353\ 76,$$

$$x_1 = 0.5353,\ x_1^2 = 0.28\ 6546\ 09,$$

$$x_2 = 0.6546,\ \ldots\ldots$$

- It is one of the oldest attempt of RNG, very popular in the early days. Hard to state conditions for picking initial seed that will generate a "good" sequence.
- It is an unsatisfactory method (easily settle into patterns of very short period). Also, zeros, once they appear, are carried in subsequent numbers.

The Mid-Square Method

"Bad" sequences:

- $x_0 = 5197$

$5197^2 = 27008809 \rightarrow x_1 = 0088, \quad R_1 = 0.0088$
$0088^2 = 00007744 \rightarrow x_2 = 0077, \quad R_2 = 0.0077$
$0077^2 = 00005929 \rightarrow x_3 = 0059, \quad R_3 = 0.0059$

- $x_i = 6500$

$6500^2 = 42250000 \rightarrow x_{i+1} = 2500, \quad R_{i+1} = 0.2500$
$2500^2 = 06250000 \rightarrow x_{i+2} = 2500, \quad R_{i+2} = 0.2500$

The method is too limited to be actually really useful.

Linear Congruential Generator (LCG)

This is by far the most popular method.

The iterative formula is

$$X_{n+1} = (aX_n + c) \bmod m, \; n \geq 0,$$

where the four integers are, respectively

m, the modulus $(= 2^{32}, \text{say})$

a, the multiplier (choose carefully)

c, the increment (may be zero)

X_0, the starting value, or seed

- An LCG with c=0 and m prime is called Multiplicative LCG (MLCG).

Linear Congruential Generator (LCG)

Random number seed:

- Virtually all computer methods of random number generation start with an initial random number seed.
- This seed is used to generate the next random number and then is transformed into a new seed value.

Linear Congruential Generator (LCG)

$$X_{n+1} = (aX_n + c) \bmod m, \; n \geq 0,$$

Properties :

(a) $\mathbf{X_i} \in \{0,1,\ldots,\boldsymbol{m}-1\}$.

(*b*) The $\mathbf{X_i}$ are periodic with period $\leq$ M.

Uniform random real numbers between 0 and 1 are given by

$$U_n = X_n / m, n = 1, 2, \cdots$$

Period

- The linear congruential sequence like other sequences generated by the formula

$$X_{n+1} = f(X_n) \bmod m$$

always has a repeated cycle or period.

- Long period is desirable. A sequence of full period is one whose period is m.
- The choice of the parameters m, a, c is important.

Examples of LCG

$$X_{n+1} = (aX_n + c) \bmod m, \; n \geq 0,$$

- m=10, X0=a=c=7.

  This generates the sequence:

  7,6,9,0,7,6,9,0,……

  The period is 4.

- m=16, a=5, c=1, X0=0.

  This generates the sequence:

  0,1,6,15, 12, 13, 2, 11, 8, 9,14,7,4,5,10,3,0,

  The sequence is of full period, 16.

Tausworthe Generator

- **Feedback shift-register method (FSR) or Tausworthe generator** (using the binary nature of computers)

$$x_k = (c_p x_{k-p} + \cdots + c_1 x_{k-1}) \bmod 2,$$

$$p - \textbf{given},\ c_i = 0 \textbf{ or } 1,\ \textbf{initial values:} (x_{-p,}, \ldots, x_{-1}),$$

$$u_1 = 0.x_v \ldots x_{v+L-1},\ u_2 = 0.x_{2v} \ldots x_{2v+L-1},\ \ldots$$

$$u_k = \sum_{j=1}^{L} x_{kv+j-1} 2^{-j} = 0.x_{kv} \cdots x_{kv+L-1}$$

where v is the step size

and L is the word length (32 or 64).

Fibonacci Generators

- These are generators where current value is sum (or difference, or XOR) of two preceding elements.
- Lagged Fibonacci generators use two numbers earlier in sequence

$$J_k = (J_{k-p} + J_{k-q}) \bmod (m)$$

$$U_k = \frac{J_k}{m}$$

p, q **are the lags.**

More Methods

- **Multiple recursive generator (MRG)**

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \cdots + a_k X_{n-k}) \bmod m, \; n \geq k,$$

$$U_n = X_n / m$$

$$\text{initial value} : X_{k-1}, \ldots, X_0.$$

- **Nonlinear congruential method**:

$$X_n = f(X_n) \bmod m$$

More Methods

- **Combined Linear Congruential Generators**

- Longer period generator is needed because of the increasing complexity of simulated systems.

- Approach: Combine two or more multiplicative congruential generators.

3. Methods for Testing a Generator

- **What are good random numbers?**

  The numbers should meet “all” aims.

- **Requirements:**

- A large period;
- Pass as many tests as possible;
- Solid theoretical support;
- Reproducibility, fast and efficient, cheap and easy…

Choosing a RNG is like choosing a new car (speed vs robustness and reliability): for some applications speed is preferred, while for others robustness and reliability are more important, whereas in coding and cryptography unpredictability is crucial.

3. Methods for Testing a Generator

- The quality of RNGs can be assessed in two ways.
- Investigate the theoretical properties of the RNG:
  - The period length of the generator
  - Various measures of uniformity
  - Independence.
- This type of testing is called theoretical, as it does not require the actual output of the generator but only its algorithmic structure and parameters.
- Powerful theoretical tests are only feasible if the generators have a sufficiently simple structure, such as those of linear congruential and multiple recursive methods and combined versions thereof.

3. Methods for Testing a Generator

- Statistical tests

- Apply a battery of statistical tests to the output of the generator, with the objective to detect deviations from uniformity and independence.
- Hence, any candidate generator should pass a wide range of statistical tests that examine uniformity and independence.

Theoretical vs Statistical Testing

- Theoretical analysis is more powerful, but is in general very hard or unavailable (period length, measures of uniformity and independence).
  It relies on the algorithmic structure and parameters (does not require actual output).

- Empirical testing is readily applicable in practice, and hence very useful. Two properties are usually tested: uniformity and independence.
  - Easy to run;
  - Can check the correctness of analysis and implementation;

Two Categories of Testing

- Testing for uniformity. The hypotheses are:

H_0: $R_i \sim U[0,1]$

H_1: $R_i \not\sim U[0,1]$

Failure to reject the null hypothesis, H_0, means that evidence of non-uniformity has not been detected.

- Testing for independence. The hypotheses are:

H_0: $R_i \sim$ independently distributed

H_1: $R_i \not\sim$ independently distributed

Failure to reject the null hypothesis, H_0, means that evidence of dependence has not been detected.

Some Empirical Tests

- Uniformity
  - Goodness of fit,
  - Chi-square test;
  - Kolmogorov-Smirnov test;

- Independence
  - Gap Test
  - Runs Test
  - Poker Test
  - Spectral Test
  - Autocorrelation Test

Chi-square Test

- Divide [0,1) into K disjoint intervals of length 1/K;
- The test statistic is

$$\chi_0^2 = \sum_{i=1}^{K} \frac{(N_j - E_j)^2}{E_j},$$

where N_j **– observed frequency of the** j**th interval,**

$E_j = n/K$ **– expectedfrequency** $(n = \sum_j N_j)$

- The statistic has a Chi-square distribution with k-1 degrees of freedom.

Chi-square Test

Null hypothesis for the Chi-square test:

H_0 : The samples are realizations of i.i.d. U(0,1) RV.

- Large values of the statistic χ_0^2 suggest nonuniformity, since the points are less uniformly distributed over [0,1).

- The null hypothesis is rejected at the $100\alpha\%$ significance level if

$$\chi_0^2 > \chi_{k-1,\alpha}^2, \text{ where } \alpha = P(\chi_0^2 > \chi_{k-1,\alpha}^2).$$

Kolmogorov-Smirnov Test

- The statistics of the K-S test is

$$\max | F_n(x) - x |;$$

In the present case, $F(x) = x$.

$$\text{where } F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{(-\infty, x]}(u_i)$$

- The asymptotic distribution of the statistic is complicated, but the quantiles have been tabulated.

- This test is still weak, but stronger than the Chi-square test.

Some Empirical Tests for Independence

Testing independence is a more complicated matter.

- ✓ Serial test;
- ✓ Autocorrelation test;
- ✓ Gap test;
- ✓ Coupon collector's test;
- ✓ Permutation test;
- ✓ Run test;

……

Since correlations often happen among the consecutive numbers in a sequence, these tests examine these aspects.

Serial Test

- Consider the pairs

$$(U_{2k}, U_{2k+1})$$

and use the Chi-square test for the K^2 kategories in the two-dimensional space.

- Triple, quadruples, etc. can be used as well.

- Weak test.

Autocorrelation Test

- The correlation coefficient between two random variables reflects their linear dependence.
- If they are independent, then their correlation coefficient is zero (but the inverse is not true).
- Define the autocorrelation coefficient of order j:

$$\rho(j) = \frac{\frac{1}{n-j}\sum_{i=1}^{n-j}(X_i - \overline{X})(X_{i+j} - \overline{X})}{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2}$$

$$u_j = \sqrt{n-j}\ \rho(j) \to N(0,1), \text{ if } \rho = 0$$

Application-Based Tests

- One of the most important tests is the test run of the RNG with a similar problem to the one that should be solved but from which the exact solution is already known.
- If the MC method performs well with this task, the RNG seems to be suitable for more complicated problem.

Why So Many Tests?

- Often a new RNG is required to pass a suite of tests.
- The proposer of a new RNG needs to "prove" that his numbers are sufficiently random.
- A RNG may pass a number of them, but can fail a particular test.
- Passing many tests may improve one's confidence in the RNG, but never guarantees that the RNG is foolproof for all kinds of simulations.
- Bad RNGs are those that fail simple tests, good ones fail only complicated tests that are hard to find.
- No RNG is "perfect".

4. Examples

- **RNG in Matlab (default RNG for many year):**

  Using parameters (in LCG)

  a=7^5=16807;

  c=0;

  m=2^31-1=2,147,483,647 (over 2 billion);

  period = m-1;

- Between 1995-2007, the default RNG in Matlab is called “substract-with borrow”, which has a period of 2^1492.

4. Examples

- After 2007, the default RNG in Matlab is Mwesene Twister, a Twisted linear **Feedback shift-register method (FSR) , with a period 2^19937-1.**
- rand

  rand(n) --- n X n random matrix;

  rand(m,n) --- m X n random matrix;

  rand('state', 0) --- Return rand to its default initial state

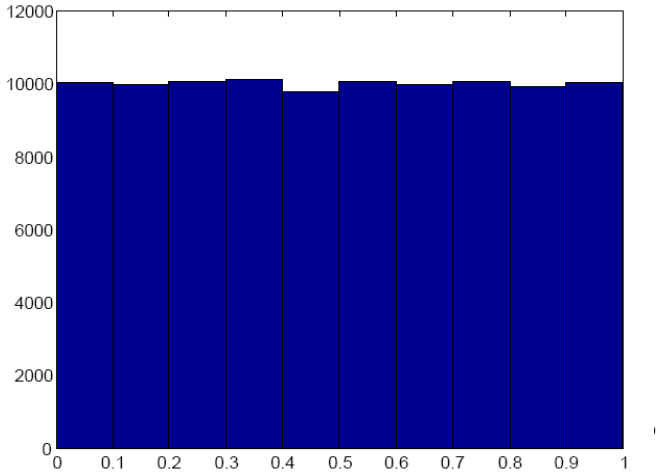
  rand('state', n);

  rand(('state', sum(100*clock)) ;

  rand('seed')

  X = rand(1,10^5);

  hist(X,10)



5. Final Remarks

- Generating “good” uniform random variables is technically complex.
- Always run your simulation with more than one RNG.
- For serious applications, do not trust the RNGs provided in popular commercial software such as Excel, Visual Basic, etc.
- LCG is not suitable for cryptographic purposes. From enough numbers in the sequence it is possible to predict how the sequence continues, due to the linear recursive relation. Non-linear generator can be used when cryptographic security is required.

5. Final Remarks

- **Low-discrepancy sequences as an alternative**
- Some computations making use of a RNG can be summarized as the computation of a total or average value, such as the computation of integrals by the MC method.
- For such problems, it may be possible to find a more accurate solution by the use of so-called low-discrepancy sequences, also called quasi-random numbers.
- Low-discrepancy sequences have a definite pattern that fills in gaps evenly; a truly random sequence may leave larger gaps.

The End of Chapter 2