The Computer Simulation of Monté Carlo Methods and Random Phenomena

The Computer Simulation of Monté Carlo Methods and Random Phenomena

By

Abdo Abou Jaoudé

Cambridge Scholars Publishing

The Computer Simulation of Monté Carlo Methods and Random Phenomena

By Abdo Abou Jaoudé

This book first published 2019

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-5275-2327-6
ISBN (13): 978-1-5275-2327-2

To my father Miled and my mother Rebecca, my brother Maroun and my sister Zeina and all my family… who were my moral support during all my life and without whom this work would not have been accomplished.

TABLE OF CONTENTS

Preface xi

Chapter I 1
Introduction

Chapter II 5
Probability and Random Variables: Some Important Definitions

Chapter III 12
Random Number Generators
- I. The Generators 12
  - 1. Congruential Generators
  - 2. Shift-register Generators
  - 3. Lagged-Fibonacci Generators
  - 4. Combination Generators
- II. The Programs 15
  - 1. The First Program
  - 2. The Second Program
  - 3. The Third Program
- III. Conclusion 31

Chapter IV 32
The Computation of *Pi* and *e*: The Base of Natural Logarithms
- I. Transcendental Numbers 32
- II. The Monté Carlo Integration Method 33
- III. The Algorithms to Compute *Pi* 34
  - 1. The First Algorithm
  - 2. The Second Algorithm
  - 3. The Third Algorithm
  - 4. The Fourth Algorithm
  - 5. The Fifth Algorithm: Archimedes Method
  - 6. The Sixth Algorithm
  - 7. The Seventh Algorithm
  - 8. The Eighth Algorithm: Buffon's Needle
  - 9. The Ninth Algorithm: Buffon's Needle

10. The Tenth Algorithm: Buffon's Needle
11. The Eleventh Algorithm: The Formula of Gregorie and Leibniz
12. The Twelfth Algorithm: The Formula of Wallis
13. The Thirteenth Algorithm: The Formula of Viète
14. The Fourteenth Algorithm: The Formula of Euler
15. The Fifteenth Algorithm: The Formula of Ramanujan
IV. The Algorithms to Compute the Base of Natural Logarithms: The Sixteenth Program 63
V. Conclusion 66

Chapter V 67
The Simulation of Random Phenomena
I. Introduction 67
II. The Algorithms 68
1. The Neutron Shielding Problem
2. The Loaded Die Problem
3. The Kzovck Family – The Dying Family Name Problem
4. The Flywheel Problem
5. The Coin Problem
6. The Delta Problem
7. Rolling One Die Problem
8. The Distance Problem
9. The Flips Problem
10. The Obtuse Angle Problem
11. The Random Section Problem
12. The Random Walk Problem
13. The Second Random Walk Problem: The Drunkard
14. The Sum of Two Dice Problem
15. The Chord Problem
16. The Birthday Problem
17. The Two Dice Problem
18. The Unit Cube Problem
III. Conclusion 115

Chapter VI 117
The Estimation of Areas and Volumes by Monté Carlo Techniques
I. Numerical Integration 117
II. The Estimation of Areas and Volumes 119
1. The First Algorithm
2. The Second Algorithm

3. The Third Algorithm
4. The Fourth Algorithm
5. The Fifth Algorithm
6. The Sixth Algorithm
7. The Seventh Algorithm
8. The Eighth Algorithm
9. The Ninth Algorithm
10. The Tenth Algorithm
11. The Eleventh Algorithm
12. The Twelfth Algorithm
13. The Thirteenth Algorithm
14. The Fourteenth Algorithm
III. The Estimation of Triple Integrals and More 168
1. The First Algorithm
2. The Second Algorithm
3. The Third Algorithm
4. The Fourth Algorithm
IV. Conclusion 178

Chapter VII 179
Applying the Monté Carlo Technique to some well-known Statistical Distributions
I. The Discrete Probability Distributions 179
1. The Bernoulli Distribution
2. The Binomial Distribution
3. The Multinomial Distribution
4. The Hypergeometric Distribution
5. The Multivariate Hypergeometric Distribution
6. The Negative Binomial Distribution
7. The Geometric Distribution
8. The Binomial and the Normal Distributions
9. The Binomial and the Poisson Distributions
10. The Discrete Gamma and Exponential Distributions
11. The Discrete Chi-Squared Distribution
12. The Discrete Weibull Distribution
II. The Continuous Probability Distributions 222
1. The Normal Distribution
2. The Standard Normal Distribution
3. The Binomial and the Normal Distributions
4. The Continuous Gamma and Exponential Distributions
5. The Continuous Chi-Squared Distribution

6. The Lognormal Distribution
7. The Continuous Weibull Distribution
8. The Beta Distribution
9. The Erlang Distribution
10. The t-distribution
11. The F-distribution
III. Conclusion 255

Chapter VIII 256
The Development of Kolmogorov's Set of Axioms: An Introduction to the Complex Probability Paradigm
I. Introduction 256
II. The Original Set of Axioms 257
III. The Imaginary Part 258
IV. Analysis and Interpretation 260
1. What Does the Imaginary Part Denote?
2. The Degree of our Knowledge
3. What Does the Last Relation Mean?
4. The Chaotic Factor
5. The Magnitude of the Chaotic Factor
6. An Example
7. A Summary
V. The Development of Two Probability Distributions 273
1. The Binomial Distribution
2. The Gaussian or Normal Distribution
VI. The Algorithms 278
1. The Bernoulli Algorithm
2. The Binomial Algorithm
VII. Conclusion 282

Chapter IX 285
The Conclusion of the Book

Bibliography and References 288

PREFACE

This book is titled "*The Computer Simulation of Monté Carlo Methods and Random Phenomena*". It includes algorithms that illustrate the famous Monté Carlo Methods and the computer simulation of random phenomena in the following areas: random number generation, simulation of Buffon's needle, computation of *Pi* and *e* (the base of logarithms), simulation of a random walk, integration (simple and multiple), areas and volumes, statistical distributions, algorithm of the neutron shielding…Why algorithms and why Monté Carlo Methods? My background is a Ph.D. in Computer Science and a Ph.D. in Applied Mathematics, so I combined my knowledge in computers with Applied Mathematics and I wrote this book. What is important here is both the algorithms (that are my creation) and the theorems to already established mathematical problems. In fact the theorems are the work of eminent mathematicians like the famous French *Georges-Louis Leclerc or Comte de Buffon* (1707-1788), the famous Greek mathematician *Archimedes* (287-212 BC), the Swiss *Jacques Bernoulli* (1654-1705), the Indian *Srinivasa Ramanujan* (1887-1920), the German mathematician *Carl Friedrich Gauss* (1777-1855) who was considered the prince of mathematicians, the Swiss mathematician *Leonhard Euler* (1707-1783), *Stanislaw M. Ulam* (1909-1984) who applied the whimsical name of Monté Carlo Methods to this way of imitating reality by a computer. We mention also, *Von Neumann*, *Bayer* and *Diaconis*, *Chaitin*, *Evans*, *Flehinger*, *Greenbaum*, *Hammersley* and *Handscomb*, *Hansen*, *Marsaglia*, and *Niederreiter*.

We must pay tribute to those magnificent giants of science who contributed to enrich our knowledge of mathematics and computer science and increased our understanding of all-natural phenomena.

Moreover, the book develops methods for simulating complicated processes or phenomena. If the computer can be made to imitate an experiment or process, then by repeating the computer simulation with different data, we can draw statistical conclusions. In such an approach, the conclusions may lack a high degree of mathematical precision but still be sufficiently accurate to enable us to understand the process being simulated.

Eighty-two algorithms were written for this purpose. The work was done using Microsoft Visual C++ due to its excellent well-definedness, modularity, portability, and efficiency.

An additional section was added and it is: The Development of *Kolmogorov*'s Set of Axioms: An Introduction to the Complex Probability Paradigm. The theorem is my creation also and was the subject of 12 international journals papers since 2010 till 2018. I wrote two algorithms to illustrate it.

The book purpose is hence to write algorithms to simulate Monté Carlo Methods and Random Phenomena. It was executed on a workstation computer to acquire a suitable speed and efficiency needed for these numerical and computer science methods.

To conclude, due to its universality, mathematics is the most positive and certain branch of science. Surely, the pleasure of working and doing mathematics and computer science is everlasting. I hope that the reader will benefit from both and share the pleasure of examining the present manuscript. Finally, the combination of both mathematics and computer science leads to "magical" and amazing results and algorithms, and the following work is an illustration of this approach.

Abdo Abou Jaoudé, Ph.D.
Notre Dame University-Louaizé, Lebanon
December 28th, 2018.

CHAPTER I

INTRODUCTION

"Chance is the pseudonym of God when He did not want to sign"
Anatole France

In applied mathematics, the name *Monté Carlo* is given to the method of solving problems by means of experiments with random numbers. This name, after the casino at Monaco, was first applied around 1944 to the method of solving deterministic problems by reformulating them in terms of a problem with random elements which could then be solved by large-scale sampling. But, by extension, the term has come to mean any simulation that uses random numbers.

The development and proliferation of computers has led to the widespread use of Monté Carlo methods in virtually all branches of science, ranging from nuclear physics (where computer-aided Monté Carlo was first applied) to astrophysics, biology, engineering, medicine, operations research, and the social sciences.

The Monté Carlo Method of solving problems by using random numbers in a computer – either by direct simulation of physical or statistical problems or by reformulating deterministic problems in terms of one incorporating randomness – has become one of the most important tools of applied mathematics and computer science. A significant proportion of articles in technical journals in such fields as physics, chemistry, and statistics contain articles reporting results of Monté Carlo simulations or suggestions on how they might be applied. Some journals are devoted almost entirely to Monté Carlo problems in their fields. Studies in the formation of the universe or of stars and their planetary systems use Monté Carlo techniques. Studies in genetics, the biochemistry of DNA, and the random configuration and knotting of biological molecules are studied by Monté Carlo methods. In number theory, Monté Carlo methods play an important role in determining primality or factoring of very large integers far beyond the range of deterministic methods. Several important new statistical techniques such as “bootstrapping” and “jackknifing” are based on Monté Carlo methods.

Hence, the role of Monté Carlo methods and simulation in all of the sciences has increased in importance during the past several years. These methods play a central role in the rapidly developing subdisciplines of the computational physical sciences, the computational life sciences, and the other computational sciences. Therefore, the growing power of computers and the evolving simulation methodology have led to the recognition of computation as a third approach for advancing the natural sciences, together with theory and traditional experimentation. At the kernel of Monté Carlo simulation is random number generation.

Generation of random numbers is also at the heart of many standard statistical methods. The random sampling required in most analysis is usually done by computers. The computations required in Bayesian analysis have become viable because of Monté Carlo methods. This has led to much wider applications of Bayesian statistics, which, in turn, has led to the development of new Monté Carlo methods to refinement of existing procedures for random number generation.

Various methods for the generation of random numbers have been used. Sometimes, processes that are considered random are used, but for Monté Carlo methods, which depend on millions of random numbers, a physical process as a source of random numbers is generally cumbersome. Instead of “random” numbers, most applications use “pseudorandom” numbers, which are deterministic but “look” like they were generated randomly. Chapter III discusses methods for the generation of sequences of pseudorandom numbers that simulate uniform distributions over a given interval. These are the basic sequences from which are derived pseudorandom numbers from other distributions, pseudorandom samples, and pseudo stochastic processes.

Moreover, the Latin prefix “*pseudo*” means in English false, so when it is added to the word random, it describes clearly enough the process of generating random numbers. Why? This makes us return to the definition and the meaning of the word *random* that the philosophers have meditated upon. They said: is there any deterministic mathematical equation that could describe the inherent randomness existent in nature or is there none?…It is a philosophical debate that has never ended till our times. But for us, as computer scientists and mathematicians, we agree that we can write deterministic mathematical equations that can yield random numbers *nearly similar* to that existent in nature. Hence, we called the generators of those numbers: “*pseudorandom generators of random numbers*”.

Concerning the structure of the book, it is the following:

Chapter I, is an introduction to the manuscript stating the basic ideas that we will develop throughout the whole manuscript.

Chapter II, is an introductory chapter to define mathematically the concepts of probability, random variables, and some very important definitions that will be used in the following chapters.

Chapter III, deals with some famous random number generators that will be implemented in different algorithms in this book.

Chapter IV, illustrates the use of Monté Carlo methods to compute *Pi* and *e*: the base of natural logarithms. Eighteen algorithms were used for this purpose including the Buffon’s needle algorithm.

Chapter V, talks about the simulation of random phenomena which are: the neutron shielding problem, the loaded die problem, the Kzovck’s family name that will die out after a certain number of generations and its correspondent probability, the flywheel problem, the coin program, the discriminant delta problem, the rolling die problem, the average distance between two points in a circle problem, the matching coin flips problem, the three random points on the edges of a square problem, the random section constant, the random walk problem, the drunkard problem, the sum of two dice problem, the chord problem, the birthday problem, the two dice problem, the unit cube problem.

Chapter VI, is dedicated to the computation of areas and volumes using the Monté Carlo technique. Simple (= areas), double (= volumes), triple and multidimensional integrals are hence solved by this method. This is surely interesting enough since some integrals are not solved analytically due to the difficulty of accomplishing this task, but they are computed numerically due to the straightforwardness and simplicity of the method.

Chapter VII, applies the technique to some well-known statistical distributions which are: Bernoulli distribution, Binomial, Beta, discrete and continuous Chi-squared distributions, F-distribution, discrete and continuous Gamma distributions, Exponential, Erlang, Geometric, Hypergeometric, Multivariate Hypergeometric, Lognormal, Multinomial, Negative Binomial distributions, Binomial versus the Normal distribution, the Standard Normal distribution, the Normal distribution, the Binomial versus the Poisson distribution, t-distribution, discrete and continuous Weibull distributions.

Chapter VIII, is a development of Kolmogorov's axioms which are at the foundations of probability theory and hence it opens the door to a deterministic expression of probabilistic events. It is an interesting chapter indeed that should be read and that I preferred to include in this manuscript since the latter deals with random phenomena and probability theory. Two algorithms illustrate the original idea, but surely a whole dissertation can be written on this chapter alone, since determinism versus nondeterminism is a very deep debate among mathematicians and among physicists like between Albert Einstein and Niels Bohr…

Chapter IX, finally, is a conclusion of the book. In the last chapter, we conclude this interesting and exciting topic with few pages in which we try to summarize the previous chapters developed in the manuscript.

CHAPTER II

PROBABILITY AND RANDOM VARIABLES: SOME IMPORTANT DEFINITIONS

"Dicing with the Deity: If God played dice, He'd win…"
Ian Stewart

It is important before using extensively the concepts of probability and random variables in the whole book that we define them first. In fact, what follows is a list of definitions that we will use in the subsequent chapters:

Definition 1: A random experiment is an experiment in which:

a) All outcomes of the experiment are known in advance.
b) Any performance of the experiment results in an outcome that is not known in advance.
c) The experiment can be repeated under identical conditions.

Definition 2: The set of all possible outcomes of a statistical or random experiment is called the sample space and is represented by the symbol *S*.

Definition 3: An event is a subset of a sample space.

Definition 4: The complement of an event *A* with respect to *S* is the subset of all elements of *S* that are not in *A*. We denote the complement of *A* by the symbol *A'* or $\overline{A}$.

Definition 5: The intersection of two events *A* and *B*, denoted by the symbol $A \cap B$ is the event containing all elements that are common to *A* and *B*.

Definition 6: Two events *A* and *B* are mutually exclusive, or disjoint, if $A \cap B = \varnothing$, that is if *A* and *B* have no elements in common.

Definition 7: The union of the two events *A* and *B*, denoted by the symbol $A \cup B$, is the event containing all the elements that belong to *A* or *B* or both.

Definition 8: A permutation is an arrangement of all or part of a set of objects. The number of permutations of *n* distinct objects is *n*!. The number of permutations of *n* distinct objects taken *r* at a time is:

$$_nP_r = \frac{n!}{(n-r)!}$$

Definition 9: The number of distinct permutations of *n* things of which n_1 are of one kind, n_2 of a second kind,... n_k of a k^{th} kind is:

$$\frac{n!}{n_1!n_2!...n_k!}$$

Definition 10: The number of combinations of *n* distinct objects taken *r* at a time is:

$$\binom{n}{r} = {}_nC_r = \frac{n!}{r!(n-r)!}$$

Definition 11: The number of arrangements of a set of *n* objects into *r* cells with n_1 elements in the first cell, n_2 elements in the second, and so forth, is:

$$\binom{n}{n_1, n_2, \cdots, n_r} = \frac{n!}{n_1!n_2!\cdots n_r!}, \quad \text{where } n_1 + n_2 + \cdots + n_r = n$$

Definition 12: The probability of an event *A* is the sum of the weights of all sample points in *A*. Therefore:

a) $0 \le P(A) \le 1$

b) $P(\varnothing) = 0$

c) And $P(S) = 1$

Furthermore, if $A_1, A_2, A_3, \cdots$ is a sequence of mutually exclusive events then:

$$P(A_1 \cup A_2 \cup A_3 \cdots) = P(A_1) + P(A_2) + P(A_3) + \cdots$$

Definition 13: If an experimenter can result in any one of *N* different equally likely outcomes and if exactly *n* of these outcomes corresponding to event *A*, then the probability of event *A* is:

$$P(A) = \frac{n}{N}$$

Definition 14: If *A* and *B* are any two events, then:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Definition 15: If *A* and *B* are mutually exclusive, then:

$$P(A \cup B) = P(A) + P(B)$$

Definition 16: If $A_1, A_2, \cdots A_n$ are mutually exclusive, then:

$$P(A_1 \cup A_2 \cup \cdots \cup A_n) = P(A_1) + P(A_2) + \cdots + P(A_n)$$

Definition 17: If $A_1, A_2, \cdots A_n$ is a partition of a sample space *S*, then:

$$P(A_1 \cup A_2 \cup \cdots \cup A_n) = P(A_1) + P(A_2) + \cdots + P(A_n) = P(S) = 1$$

Definition 18: For three events *A*, *B*, and *C*,

$$P(A \cup B \cup C) =$$
$$P(A) + P(B) + P(C) - P(A \cap B) - P(A \cap C) - P(B \cap C) + P(A \cap B \cap C)$$

Definition 19: If *A* and *A*' are complementary events, then:

$$P(A) + P(A') = 1$$

Definition 20: The conditional probability of *B*, given *A*, is denoted by:

$$P(B / A) = \frac{P(A \cap B)}{P(A)}, \quad \text{if } P(A) > 0$$

Definition 21: Two events *A* and *B* are independent if and only if:

$$P(B/A) = P(B) \quad \text{or} \quad P(A/B) = P(A)$$

Otherwise, *A* and *B* are dependent.

Definition 22: If in an experiment the events *A* and *B* can both occur, then:

$$P(A \cap B) = P(A).P(B/A) = P(B).P(A/B)$$

Definition 23: Two events *A* and *B* are independent if and only if:

$$P(A \cap B) = P(A).P(B)$$

Definition 24: If, in an experiment, the events $A_1, A_2, \cdots A_k$ can occur, then

$$P(A_1 \cap A_2 \cap A_3 \cap \ldots \cap A_k) = \\ P(A_1).P(A_2/A_1).P(A_3/A_1 \cap A_2)\ldots P(A_k/A_1 \cap A_2 \cap \ldots \cap A_{k-1})$$

If the events $A_1, A_2, \cdots A_k$ are independent, then:

$$P(A_1 \cap A_2 \cap A_3 \cap \ldots \cap A_k) = P(A_1).P(A_2).P(A_3)\ldots P(A_k)$$

Definition 25: If the events $B_1, B_2, \ldots B_k$ constitute a partition of the sample space *S* such that $P(B_i) \neq 0$ for *i* = 1, 2, …, *k* then for an event *A* of *S*,

$$P(A) = \sum_{i=1}^{k} P(B_i \cap A) = \sum_{i=1}^{k} P(B_i).P(A/B_i)$$

Definition 26: (Bayes' Rule) If the events $B_1, B_2, \ldots B_k$ constitute a partition of the sample space *S*, where $P(B_i) \neq 0$ for *i* = 1, 2, …, *k* , then for any event *A* in *S* such that $P(A) \neq 0$, then :

$$P(B_r / A) = \frac{P(B_r \cap A)}{\sum_{i=1}^{k} P(B_i \cap A)} = \frac{P(B_r).P(A / B_r)}{\sum_{i=1}^{k} P(B_i).P(A / B_i)}, \text{ for } r = 1, 2, \ldots, k$$

Definition 27: A random variable is a function that associates a real number with each element in the sample space.

Definition 28: If a sample space contains a finite number of possibilities or an unending sequence with as many elements as there are whole numbers, it is called a *discrete sample space.*

Definition 29: If a sample space contains an infinite number of possibilities equal to the number of points on a line segment, it is called a *continuous sample space.*

Definition 30: The set of ordered pairs $(x, f(x))$ is a *probability function*, *probability mass function*, or *probability distribution* of the discrete random variable *X* if, for each possible outcome *x*,

$$\begin{cases} 1 - f(x) \geq 0 \\ 2 - \sum_{x} f(x) = 1 \\ 3 - P(X = x) = f(x) \end{cases}$$

Definition 31: The function $f(x)$ is a *probability density function* for the continuous random variable *X*, defined over the set of real numbers *R*, if:

$$\begin{cases} 1 - f(x) \geq 0, \text{ for all } x \in R \\ 2 - \int_{-\infty}^{\infty} f(x).dx = 1 \\ 3 - P(a < X < b) = \int_{a}^{b} f(x).dx \end{cases}$$

Definition 32: Let *X* be a random variable with probability distribution $f(x)$. The *mean* or *expected value* of *X* is:

$$\begin{cases} \mu = E(X) = \sum_x x.f(x) \text{ , if } X \text{ is discrete,} \\ \text{and } \mu = E(X) = \int_{-\infty}^{\infty} x.f(x).dx \text{ , if } X \text{ is continuous.} \end{cases}$$

Definition 33: Let X be a random variable with probability distribution $f(x)$ and mean μ. The *variance* of X is:

$$\begin{cases} V(X) = \sigma^2 = E[(X-\mu)^2] = \sum_x (x-\mu)^2.f(x) \text{, if } X \text{ is discrete,} \\ \text{and} \\ V(X) = \sigma^2 = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2.f(x).dx \text{ , if } X \text{ is continuous.} \end{cases}$$

The positive square root of the variance is σ and is called the *standard deviation* of X.

Definition 34: The variance of a random variable X is:

$$V(X) = \sigma^2 = E(X^2) - [E(X)]^2 = E(X^2) - \mu^2$$

Definition 35: If a and b are constant, and X and Y are two independent random variables then:

$$\begin{cases} E(aX+b) = a.E(X)+b \\ E(X \pm Y) = E(X) \pm E(Y) \\ E(X.Y) = E(X).E(Y) \end{cases}$$

and

$$\begin{cases} V(aX+b) = a^2.V(X) \\ V(aX+bY) = a^2.V(X) + b^2.V(Y) \end{cases}$$

Definition 36: If the random variable X assumes the values $x_1, x_2, \dots, x_k$, with equal probabilities, then the *discrete uniform distribution* is given by:

$$f(x;k) = \frac{1}{k} \quad \text{where} \quad x = x_1, x_2, \ldots, x_k$$

Definition 37: The density function of the *continuous uniform random variable* X on the interval $[A, B]$ is:

$$f(x;A,B) = \begin{cases} \dfrac{1}{B-A} & A \le x \le B \\ 0 & \text{elsewhere.} \end{cases}$$

CHAPTER III

RANDOM NUMBER GENERATORS

"You believe in the God who plays dice, and I in complete law and order"
Albert Einstein, Letter to Max Born

I. The Generators

Virtually all random number generators are based on the theory that may be described as follows: we have a finite set X and a function $f : X \rightarrow X$ that takes elements of X into other elements of X. Given an initial (*seed*) value, $x \in X$, (x might be a single computer word or an array of computer words), the generated sequence is:

$$\boxed{x, f(x), f^2(x), f^3(x), \ldots}$$

where $f^2(x)$ means $f(f(x))$, $f^3(x)$ means $f(f^2(x)) = f(f(f(x)))$ and so on. The three most common classes of random number generators are:

$$\begin{cases} \textbf{(1)} \ \textit{\textbf{Congruential.}} \\ \textbf{(2)} \ \textit{\textbf{Shift-register.}} \\ \textbf{(3)} \ \textit{\textbf{Lagged-Fibonacci.}} \end{cases}$$

An important property of any generator is its *period*, or how many numbers it produces before it enters a repeating pattern.

1- For ***Congruential Generators***, the finite set X is the set of reduced residues of some modulus m and $f(x) = (ax + b) \bmod m$. Thus, with an initial element $x_0 \in X$, the generated sequence is:

$$\boxed{x_0, x_1, x_2, \ldots \text{ with } x_{n+1} = (ax_n + b) \bmod m}$$

A wide variety of choices for *a*, *b*, and *m* have been described in the literature like by *Marsaglia* (1972) or *Knuth* (1997) for methods of finding periods and establishing the structure of congruential sequences. With periods around 2^{32}, congruential generators have been used successfully in Monté Carlo simulations for the past 40 years. Most of the random number generators provided by computer systems or software packages are congruential generators. But random points in higher dimensions with coordinates produced by congruential generators show a crystal-line regularity that makes them unsuitable for certain applications and that taken, with their relatively short periods, has led to the gradual adoption of longer-periods generators for serious Monté Carlo studies.

2- For ***Shift-Register Generators***, the finite set *X* is the set of $1\times k$ binary vectors $x=(b_1,b_2,\ldots,b_k)$ and the function f is a linear transformation, $f(x)=xT$, with T a $k\times k$ binary matrix and all arithmetic modulo 2. With an initial binary vector *x* the sequence is:

$$\boxed{x, xT, xT^2, xT^3, \ldots}$$

with the matrix *T* chosen so that the period is long and multiplication by *T* is reasonably fast in computer implementations. Shift-register generators are sometimes called *Tausworthe generators.* The use of shift-register generators is declining. Those based on standard length computer word with *k* = 32, do not perform as well on tests of randomness as do congruential generators, and their periods are, like those of congruential generators, too short. Their main use is in forming part of a combination generator. The use of shift-register generators with extremely long binary vectors (*k* = 607, 1279, or even 9689 bits) still has some attraction, for the periods are extremely long (2^k-1), and special hardware (called shift registers and hence the name of the general method), is easily constructed for their implementation. They are mainly used in special-purpose machines for Monté Carlo studies in physics.

3- For ***Lagged-Fibonacci Generators***, the finite set *X* is the set of $1\times r$ vectors $x=(x_1,x_2,\ldots,x_r)$ *with elements* x_i in some finite set *S* on which there is a binary operation $\circ$. The function f is defined by:

$$f(x_1,x_2,\ldots,x_r)=(x_2,x_3,x_4,\ldots,x_r,x_1\circ x_{r+1-s})$$

where r and s $(1 \le s \le r)$ are the two lag parameters. Informally, a lagged-Fibonacci sequence is described by means of a set of r seed values followed by the rule for generating succeeding values:

$$\boxed{x_1, x_2, \ldots, x_r, x_{r+1}, \ldots \quad \text{with} \quad x_n = x_{n-r} \circ x_{n-s}}$$

but to define and establish formally the period and structure of such sequences they must be viewed as iterates $x, f(x), f^2(x), \ldots$ on the set X of $1 \times r$ vectors with elements in the set S on which the binary operation $\circ$ is defined.

Various choices for S and $\circ$ lead to interesting sequences – for example, when S is the set of reduced residues of some modulus m and $\circ$ is addition or subtraction mod m; S is the set of reduced residues relatively prime to m and $\circ$ is multiplication; S is the set of $1 \times k$ binary vectors and $\circ$ is addition of binary vectors (exclusive-or); S is the set of floating-point computer numbers $0 \le x < 1$ having 24-bit fractions and $x \circ y = \{$ if $x > y$ then $x - y$ else $x - y + 1$ $\}$. Such generators are often designated $F(r, s, \circ)$ generators.

While examples of generators of each of the three standard methods described above are widely used and – for most purposes – work quite well, new methods are always being developed. All standard generators (with the exception of lagged-Fibonacci using multiplication) fail one or more stringent tests of randomness such as those described in *Marsaglia* (1985, 1995), and many of them have periods too short for the huge samples that current computer speeds make possible.

4- *Combination Generators*: Experience has shown, and there is theory to support it, that combining two different kinds of generators, by perhaps subtraction or multiplication, produces a combination generator that has a much longer period and performs better, or no worse than, either component in tests of randomness. The McGill Random Number Generator Super-Duper, one of the most commonly used generators of the past 30 years, combines a congruential generator with a shift-register generator. Its period is about 2^{62}.

II. The Programs

1- In the First Program on random number generators we test the uniformity of the following generators:

1- In the first function, *the built in C++ random number generator* which is *rand*() and which has a seed value determined by *srand*(*time*(0)) and a period = 32767.

2- In the second function, *the extended congruential generator* which is:

$$\boxed{x_n = \left(1999x_{n-1} + 4444x_{n-2}\right) \bmod 2^{31} - 1}$$

and which has a seed = two 31-bit integers and an approximate period = 4.6×10^{18}.

3- In the third function, *the congruential generator* which is:

$$\boxed{x_n = \left(69069x_{n-1}\right) \bmod 2^{32}}$$

and which has a seed = one 32-bit odd integer and an approximate period = 4.3×10^{9}.

4- In the fourth function, *the congruential generator* which is:

$$\boxed{x_n = \left(69069x_{n-1} + 1\right) \bmod 2^{32}}$$

and which has a seed = one 32-bit integer and an approximate period = 4.3×10^{9}.

5- In the fifth function, *the congruential generator* which is:

$$\boxed{x_n = \left(16807x_{n-1}\right) \bmod 2^{31} - 1}$$

and which has a seed = one 31-bit integer $\neq 0$ and an approximate period = 2.1×10^{9}.

6- In the sixth function, *the combination generator* which is:

$$\boxed{x_n = \left(x_{n-1} \times x_{n-2}\right) \bmod 2^{26}}$$

and which has a seed = two 32-bit odd x's and a period = $2^{26} = 67,108,864$.

To test the program, consider the following input set:

$$\{\ (lb = 0, ub = 20000),\ (lb = 100, ub = 10000),\ (lb = 0, ub = 32767),\ (lb = 0, ub = 2000000),\ (lb = 0, ub = 300000000),\ (lb = 0, ub = 30000000000)\ \},$$

where lb = lower bound and ub = upper bound of the simulation interval.

The first program is the following:

```
// this program tests the uniformity of the C++ built in random
// function and of other random numbers generators

#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <iomanip>

using namespace std;

void uniform1(long double,long double);
void uniform2(long double,long double);
void uniform3(long double,long double);
void uniform4(long double,long double);
void uniform5(long double,long double);
void uniform6(long double,long double);
long double gen2(long double,long double);
long double gen3(long double);
long double gen4(long double);
long double gen5(long double);
long double gen6(long double,long double);

int main()
{
        long double lb,ub;

        cout << fixed << setprecision(7);

        cout << "                RANDOM NUMBER GENERATORS"
              << endl;
        cout << "              --------------------------------------------------"
              << endl;
        cout << endl;

        cout << "Input the lower bound of the interval = ";
        cin   >> lb;
        cout << "Input the upper bound of the interval = ";
        cin   >> ub;
        cout << endl;

        uniform1(lb,ub);
        uniform2(lb,ub);
        uniform3(lb,ub);
        uniform4(lb,ub);
        uniform5(lb,ub);
        uniform6(lb,ub);

        return 0;
}

long double gen2(long double xn1,long double xn2)
{
        long double xn;

        xn = (long double) fmod(((1999*xn1)+(4444*xn2)),pow(2,31)-1);

        return xn;
}

long double gen3(long double xn1)
{
        long double xn;

        xn = (long double) fmod(69069*xn1,pow(2,32));
        return xn;
}

long double gen4(long double xn1)
{
        long double xn;

        xn = (long double) fmod((69069*xn1)+1,pow(2,32));

        return xn;
}

long double gen5(long double xn1)
{
        long double xn;

        xn = (long double) fmod(16807*xn1,pow(2,31)-1);

        return xn;
}

long double gen6(long double xn1,long double xn2)
{
        long double xn;

        xn = (long double) fmod((xn1*xn2),pow(2,26));

        return xn;
}

void uniform1(long double slb,long double sub)
{
        long int i,j,limit1,limit2,output;
        long double EP,SP,counter;

        srand( time(0) );

        limit1 = 60000;
        limit2 = 32767;

        counter = 0;
```