Unit 5

SIMULATION THEORY

Lesson 39

Learning objective:

- **To learn random number generation.**
- **Methods of simulation.**
- **Monte Carlo method of simulation**

You've already read basics of simulation now I will be taking up method of simulation, that is Random Number Generation

Random Number Generation

Random numbers or Pseudo-random numbers are often required for simulations performed on parallel computers. The requirements for parallel random number generators are more stringent than those for sequential random number generators. As well as passing the usual sequential tests on each processor, a parallel random number generator must give different, independent sequences on each processor. We consider the requirements for a good parallel random number generator, and discuss generators for the uniform and normal distributions. These generators can give very fast vector or parallel implementations.

Random Numbers and Simulation

In many fields of engineering and science, we use a computer to simulate natural phenomena rather than experiment with the real system. Examples of such computer experiments are simulation studies of physical processes like atomic collisions, simulation of queuing models in system engineering, sampling in applied statistics. Alternatively, we simulate a mathematical model, which cannot be treated by analytical methods. In all cases a simulation is a computer experiment to determine probabilities empirically. In these applications, random numbers are required to make things realistic.

Random number generation has also applications in cryptography, where the requirements on randomness may be even more stringent.

Hence, we need a good source of random numbers. Since the validity of a simulation will heavily depend on the quality of such a source, its choice or construction will be fundamental importance. Tests have shown that many so-called random functions supplied with programs and computers are far away from being random.

By generating random numbers, we understand producing a sequence of independent random numbers with a specified distribution. The fundamental problem is to generate random numbers with a uniform discrete distribution on {0,1,2,…,N} or more suitable on {0,1/N,2/N,…,1}, say. This is the distribution where each possible number is equally likely. For N large this distribution approximates the continuous uniform distribution U(0,1) on the unit interval. Other discrete and continuous distributions will be generated from transformations of the U(0,1) distribution.

At first, scientists who needed random numbers would generate them by performing random experiments like rolling dice or dealing out cards. Later tables of thousands of random digits created with special machines for mechanically generating random numbers or taken from large data sets as census reports were published.

With the introduction of computers, people began to search for efficient ways to obtain random numbers using arithmetic operations of a computer - an approach suggested by John von Neumann in the 1940's. Since the digital computer cannot generate random numbers, the idea is, for a given probability distribution, to develop an algorithm such that the numbers generated by this algorithm appear to be random with the specified distribution. Sequences generated in a deterministic way we call pseudo-random numbers. To simulate a discrete uniform distribution John von Neumann used the so-called middle square method, which is to take the square of the previous random number and to extract the middle digits.

Example: If we generate 4-digit numbers starting from 3567 we obtain 7234 as the next number since the square of 3567 equals 12**7234**89. Continuing in the same way the next number will be 3307.

Of course, the sequence of numbers generated by this algorithm is not random but it appears to be. However, as computations show the middle square method is a poor source of random numbers.

To summarize our discussion we need

- Precise mathematical formulations of the concept of randomness
- Detailed analysis of algorithms for generating pseudo-random numbers
- Empirical tests of random number generators

What is a random sequence?

A sequence of real numbers between zero and one generated by a computer is called "pseudo-random" sequence if it behaves like a sequence of random numbers. So far this statement is satisfactory for practical purposes but what one needs is a quantitative definition of random behaviour.

In practice we need a list of mathematical properties characterizing random sequences and tests to see whether a sequence of pseudo-random numbers yields satisfactory results or not. Loosely speaking, basic requirements on random sequences are that their elements are uniformly distributed and uncorrelated. The tests we can perform will be of theoretical and/or empirical nature.

Some definitions

D.H. Lehmer(1951) : "A random sequence is a vague notion embodying the idea of a sequence in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests, traditional with statisticians and depending somewhat on the uses to which the sequence is to be put."

J.N. Franklin (1962): " A sequence (U_0, U_1,…) (note: with U_i taking values in the unit interval [0,1]) is random if it has every property that is shared by all infinite sequences of independent samples of random variables from the uniform distribution."

Generating uniform random numbers

Deterministic generators yield numbers in a fixed sequence such that the forgoing *k* numbers determine the next number. Since the set of numbers used by a computer is finite, the sequence will become periodic after a certain number of iterations.

The general form of algorithms generating random numbers may be described by the following recursive procedure.

$X_n = f(X_{n-1}, X_{n-2}, \ldots, X_{n-k})$

with initial conditions $X_0, X_1, \ldots, X_{k-1}$. Here f is supposed to be a mapping from $\{0,1,\ldots,m-1\}^k$ into $\{0,1,\ldots,m-1\}$.

For most generators k=1 in which case the recursive relation simplifies to

$X_n = f(X_{n-1})$

with a single initial value X_0, the *seed* of the generator. Now f is a mapping from $\{0,1,\ldots,m-1\}$ into itself.

In most cases the goal is to simulate the continuous uniform distribution U(0,1). Therefore the integers X_n are rescaled to

$U_n = X_n/m$.

If m is large, the resulting granularity is negligible when simulating a continuous distribution.

A good generator should be of a long period and resulting subsequences of pseudo-random numbers should be uniform and uncorrelated. Finally, the algorithm should be efficient.

Remark: You should note that initializing the generator with the same seed X_0 would give the same sequence of random numbers. Usually one uses the clock time to initialize the generator.

Mathematicians have devised a variety of procedures to generate random numbers. With these procedures, random number generation can be done either manually or with the help of a computer. Also, several collections of random number tables are available. The most commonly used table contains uniformly distributed (or normally distributed) random numbers over the interval 0 to 1. To generate other types of random numbers which obey other distribution laws, we would require access to a computer.

The simplest method for obtaining random events is coin tossing. This method can be used to obtain an ideal random number generator. Here, we show that logistic map is able to simulate the coin tossing method. Also, we describe a numerical implementation of the ideal uniform random number generator. Comparing to usual congruential random number generators, which are periodic, the logistic generator is infinite, aperiodic and not correlated.

In modern science, random number generators have proven invaluable in simulating natural phenomena and in sampling data [1-2]. There are only a few methods for obtaining random numbers. For example, the simplest method is coin tossing, where the occurrence of heads or tails are random events. By virtue of the symmetry of the coin the events are equally probable. Hence they are called equally probable events. It is therefore considered that the probability of heads (tails) is equal to 1/2.

> **Coin tossing** : The **coin tossing** belongs to the category of mechanical methods which includes also: dices, cards, roulettes, urns with balls and other gambling equipments. The mechanical methods are not frequently used in science because of the low generation speed. The methods characterized by high generation speed are those, which are based on intrinsic random physical processes such as the electronic and radioactive noise. Because the sequence of numbers generated with mechanical and physical methods are not reproducible, these methods have a great disadvantage in numerical simulations.

Analytical methods: Methods which are implemented in computer algorithms, eliminates the disadvantages of the manual and physical methods. These methods are characterized by high speed, low correlation of the numbers and reproducibility. The major drawback of these methods is the periodicity of the generated sequences.

Middle square generators: The middle square method was proposed by J. von Neumann in the 1940's. Therefore these generators are also called von Neumann generators in the literature. The middle square method consists of taking the square of the previous random number and to extract the middle digits. This method gives rather poor results since generally sequences tend to get into a short periodic orbit.

Example: If we generate 4-digit numbers starting from 3567 we obtain 7234 as the next number since the square of 3567 equals 12723489. Continuing in the same way the next number will be 3307. The resulting sequence enters already after 46 iterations a periodic orbit:

3567, 7234, 3307, 9362, 6470, 8609, 1148, 3179, 1060, 1236, 5276, 8361, 9063, 1379, 9016, 2882, 3059, 3574, 7734, 8147, 3736, 9576, 6997, 9580, 7764, 2796, 8176, 8469, 7239, 4031, 2489, 1951, 8064, 280, 784, 6146, 7733, 7992, 8720, 384, 1474, 1726, 9790, 8441, 2504, 2700, 2900, 4100, 8100, 6100, 2100, 4100

Linear congruential generators: The linear congruential generator (LCG) was proposed D.H. Lehmer in 1948. The form of the generator is

$X_n = (aX_{n-1} + c) \bmod m$

The linear congruential generator depends on four parameters

parameter	name	range
m	the modulus	{1,2,...}
a	the multiplier	{0,1,...,m-1}
c	the increment	{0,1,...,m-1}
X_0	the seed	{0,1,...,m-1}

The operation mod m is called *reduction modulo m* and is a basic operation of *modular arithmetic*. Any integer x may be represented as

$x = \text{floor}(x/m) \cdot m + x \bmod m$

where the floor function floor(t) is the greatest integer less than or equal to t. This equation may be taken as definition of the reduction modulo m.

If $c = 0$ the generator is called multiplicative. For nonzero c the generator is called mixed.

Monte Carlo Method of Simulation

The Monte Carlo method owes its development to the two mathematicians, John Von Neumann and Stanislaw Ulam, during World War II. The principle behind this method of simulation is representative of the given system under analysis by a system described by some known probability distribution and then drawing random samples for probability distribution by means of random number. In case it is not possible to describe a system in terms of standard probability distribution such as normal, Poisson, exponential, gamma, etc., an empirical probability distribution can be constructed.

The deterministic method of simulation cannot always be applied to complex real life situations due to inherently high cost and time values required so as to obtain any meaningful results from the simulated model. Since there are a large number of interactions between numerous variables, the system becomes too complicated to offer an effective simulation approach. In such cases where it is not feasible to use an expectation approach for simulating systems, Monte Carlo method of simulation is used.

It can be usefully applied in cases where the system to be simulated has a large number of elements that exhibit chance (probability) in their behaviour. As already mentioned, the various types of probability distributions are used to represent the uncertainty of real-life situations in the model. Simulation is normally undertaken only with the help of a very high-speed data processing machine such as computer. The user of simulation technique must always bear in mind that the actual frequency or probability would approximate the theoretical value of probability only when the number of trials are very large i.e. when the simulation is repeated a large no. of times. This can easily be achieved with the help of a computer by generating random numbers.

A random number table is presented here for the quick reference of the students.

Random Number Table

52	**06**	**50**	**88**	**53**	**30**	**10**	**47**	**99**	**37**	**66**	**91**	**35**	**32**	**00**	**84**	**17**	**07**
37	**63**	**28**	**02**	**74**	**35**	**24**	**03**	**29**	**60**	**74**	**85**	**90**	**73**	**59**	**55**	**36**	**60**
82	**57**	**68**	**28**	**05**	**94**	**03**	**11**	**27**	**79**	**90**	**87**	**92**	**41**	**09**	**25**	**72**	**77**
69	**02**	**36**	**49**	**71**	**99**	**32**	**10**	**75**	**21**	**95**	**90**	**94**	**38**	**97**	**71**	**85**	**49**
98	**94**	**90**	**36**	**06**	**78**	**23**	**67**	**89**	**85**	**29**	**21**	**25**	**73**	**69**	**34**	**31**	**76**
96	**52**	**62**	**87**	**49**	**56**	**59**	**23**	**78**	**71**	**72**	**90**	**57**	**01**	**98**	**57**	**44**	**95**
33	**69**	**27**	**21**	**11**	**60**	**95**	**89**	**68**	**48**	**17**	**89**	**34**	**09**	**93**	**50**	**30**	**51**
50	**33**	**50**	**95**	**13**	**44**	**34**	**62**	**64**	**39**	**55**	**29**	**30**	**64**	**49**	**44**	**26**	**16**
88	**32**	**18**	**50**	**62**	**57**	**34**	**56**	**62**	**31**	**15**	**40**	**90**	**34**	**51**	**95**	**09**	**14**
90	**30**	**36**	**24**	**69**	**82**	**51**	**74**	**30**	**35**	**36**	**85**	**01**	**55**	**92**	**64**	**49**	**85**
50	**48**	**61**	**18**	**85**	**23**	**08**	**54**	**17**	**12**	**80**	**69**	**24**	**84**	**92**	**16**	**13**	**59**
27	**88**	**21**	**62**	**69**	**64**	**48**	**31**	**12**	**73**	**02**	**68**	**00**	**16**	**16**	**46**	**33**	**85**
45	**14**	**46**	**32**	**13**	**49**	**66**	**62**	**74**	**41**	**86**	**98**	**92**	**98**	**84**	**54**	**89**	**40**
81	**02**	**01**	**78**	**82**	**74**	**97**	**37**	**45**	**31**	**94**	**99**	**42**	**49**	**27**	**64**	**13**	**42**
66	**83**	**14**	**74**	**27**	**76**	**03**	**33**	**11**	**97**	**59**	**81**	**72**	**00**	**64**	**61**	**37**	**52**
74	**05**	**82**	**82**	**93**	**09**	**96**	**33**	**52**	**78**	**13**	**06**	**28**	**30**	**94**	**23**	**58**	**39**
30	**34**	**87**	**01**	**74**	**11**	**46**	**82**	**59**	**94**	**25**	**34**	**32**	**23**	**17**	**01**		**73**

Following are the steps involved in Monte-Carlo simulation:-

Step I.
Obtain the frequency or probability of all the important variables from the historical sources.

Step II.
Convert the respective probabilities of the various variables into cumulative problems.

Step III.
Generate random numbers for each such variable.

Step IV.
Based on the cumulative probability distribution table obtained in Step II, obtain the interval (i.e.; the range) of the assigned random numbers.

Step V.
Simulate a series of experiments or trails.
Remarks. Which random number to use?

The selection of specific random number is determined by establishing a systematic and thorough selection strategy before examining the list of digits given in the random number table.

In general, the practical life situations or systems are simulated by building, first a basic inherent model & subsequently relaxing some or all of the assumptions so as to obtain a more precise model representation. Thus model building for simulations is a stepwise process and the final model emerges only after a large number of successive refinements.

Application of Monte-Carlo Simulation: Monte-Carlo simulation can now easily be applied to an example of the bread-seller. Let us suppose that the demands per unit of the bread along with their respective probabilities are as follows:

Days No.	Demand (per unit)	Probability
1	20	0.10
2	21	0.15
3	22	0.25
4	23	0.20
5	24	0.10
6	25	0.05
7	26	0.15

We can easily use a sequence of 2-digit random numbers of generating the demand based on the above information. By assigning two digit random numbers to each of the possible outcomes or daily demand, we have:

(Per unit) Days No.	Demand	Probability	Cumulative Probability	Random Nos.
1	20	0.10	0.10	00 to 09
2	21	0.15	0.25	10 to 24
3	22	0.25	0.50	25 to 49
4	23	0.20	0.70	50 to 69
5	24	0.10	0.80	70 to 79
6	25	0.05	0.85	80 to 84
7	26	0.15	1.00	85 to 99

The first entry in the random number table is 00 to 09. It means that there are 10 random numbers (00 to 09). Since each of the ten numbers has an even chance of appearing. The probability of each

number = 1/10 or 0.10; a fact that is fully supported by the cumulative probability table.

Using the above procedure, by Monte Carlo method of simulation, demand for the required number of days can easily be determined using the random number table.

Now I'll take up few examples of random number to explain this & make its practical application clear.

Example 1

New Delhi Bakery House keeps stock of a popular brand of cake. Previous experience indicates the daily demand as given below:

Table 1

Daily demand	**Probability**
0	0.01
15	0.15
25	0.20
35	0.50
45	0.12
50	0.02

Consider the following sequence of random numbers:
R. No. : 21, 27, 47, 54, 60, 39, 43, 91, 25, 20.

Using this sequence, simulate the demand for the next 10 days. Find out the stock situation if the owner of the bakery house decides to make 30 cakes every day. Also estimate the daily average demand for the cakes on the basis of simulated data.

Solution :

Table 2

Daily Demand	Probability	Cumulative probability	Random Nos.
0	**0.01**	**0.01**	**00**
15	**0.15**	**0.16**	**01 to 15**
25	**0.20**	**0.36**	**16 to 35**
35	**0.50**	**0.86**	**36 to 85**
45	**0.12**	**0.96**	**86 to 95**
50	**0.02**	**1.00**	**96 to 99**

Table 3

Demand	Random Numbers	Next demand	If he makes 30 cakes in a day	
			Left out	Shortage
1	**21**	**25**	**5**	
2	**27**	**25**	**10**	
3	**47**	**35**	**5**	
4	**54**	**35**	**0**	
5	**60**	**35**		**5**
6	**39**	**35**		**10**
7	**43**	**35**		**15**
8	**91**	**45**		**30**
9	**25**	**25**		**25**
10	**20**	**25**		**20**
Total		**320**		**10**

Next demand is calculated on the basis of cumulative probability (e.g., random number 21 lies in the third item of cumulative probability, i.e., 0.36. Therefore, the next demand is 25.)

Similarly, we can calculate the next demand for others.

Total demand = 320

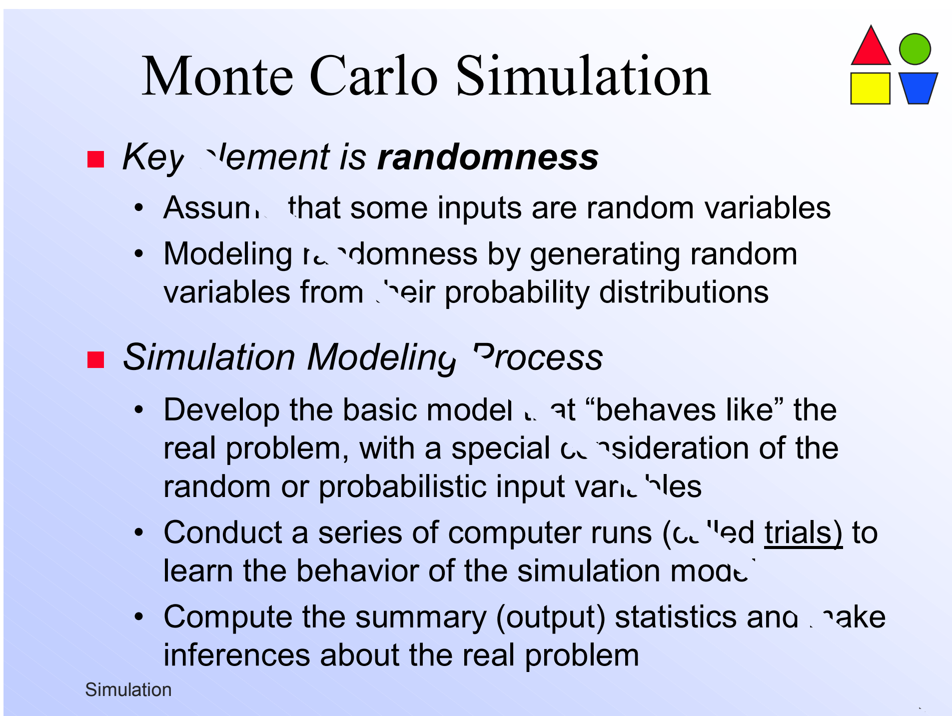
Average demand = Total demand / no. of days

The daily average demand for the cakes = 320 / 10 = 32 cakes.

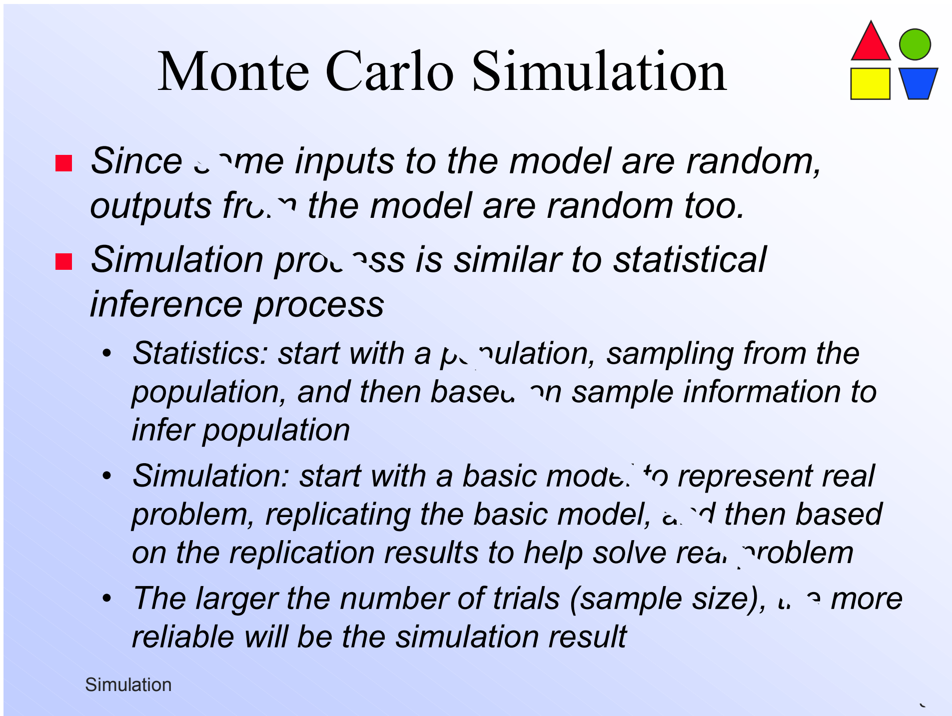
Summary

Hope you have understood the random number method of simulation. In next lesson we will study about the practical application of simulation.

Slide 1



Slide 2


Monte Carlo Simulation
Since some inputs to the model are random, outputs from the model are random too.
Simulation process is similar to statistical inference process
Statistics: start with a population, sampling from the population, and then based on sample information to infer population
Simulation: start with a basic model to represent real problem, replicating the basic model, and then based on the replication results to help solve real problem
The larger the number of trials (sample size), the more reliable will be the simulation result
Simulation

Slide 3

Example

Basic Model: Profit = f(demand)

Input: Demand → ***Relationship:*** function f → ***Output:*** Profit

How simulation works:

Step 1: basic model development: generate one possible random Demand and find the corresponding Profit

Step 2: basic model replication: generate many possible values of Demand and find corresponding Profits

Step 3: result summarization: calculate summarized statistics on the Profit such as average, min, max etc.