Outline of topics for the 10 AM lecture

The topics are:

1. Essence of Simulation
2. Coin Flipping game
3. Common Types of Applications
4. Generation of Random Numbers
5. Generation of Random Observations from a Probability Distribution

1 Introduction

There are various times when computing the solution to a problem simply is not a feasible course of action:

- Analytical results are (often) not tractable
- Function values/computation takes too much time to evaluate
- Uncertainty in parameters leads to lots of potential outcomes

To prepare for simulating complex systems, a detailed simulation model needs to be formulated to describe the operation of the system and how it is to be simulated. A simulation model has several building blocks:

- The state of the system (e.g. the number of customers in a queueing system)
- The possible states of the system that can occur
• The possible events (e.g. arrivals and service completions in a queueing system) that would change the state of the system
• A simulation clock will record the passage of (simulated) time
• A method for randomly generating the events of various kinds
• A formula for identifying state transitions that are generated by the various kinds of events

1.1 Discrete-Event vs Continuous Simulation

Discrete event simulation is one where changes in the state of the system occur instantaneously at random points in time as a result of the occurrence of discrete events. For example, in a queueing system where the state of the system is the number of customers in the system. Continuous simulation is one where changes in state of the system occur continuously over time. For example, if we are interested in an airplane in flight, then state is defined as the current position of the airplane which changes continuously over time. Continuous simulation typically require differential equations.

2 Coin Flipping Game

Let us consider a new game with the following rules.

• Each play of the game involves repeatedly flipping an unbiased coin until the difference between the number of heads tossed and the number of tails is 3.
• If someone decides to play the game, one is required to pay $1 for each flip of the coin. No one is allowed to quit once he/she has started playing.
• The player receives $8 at the end of the game.

Therefore, the player makes a profit from the game if the number of flips required is fewer than 8, but he/she loses if more than 8 flips are required. Some example game outcomes are shown in Table 1. How would one can decide whether to play this game? One possible solution would decide on the basis of simulation. For example, one may spent one hour repeatedly flipping a coin and recording their results. This method imitates the actual play without actually winning or losing money.

Another solution would be to simulate the game on a computer. One problem, however, is a computer can not flip coins! It accomplishes the same task by generating a sequence of random observations from a uniform distribution between 0 and 1. One easy way is to generate these uniform random numbers is to use the RAND function in Excel. To simulate the flipping of a coin, the computer can use the following correspondence.

• 0.0000 to 0.4999 corresponds to head
Table 1: These 3 possible game outcomes show how the player can either win or lose money. A coin toss of heads is denoted by a ‘H’ and tails is denoted by a ‘T’.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Flips</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>HHH</td>
<td>3</td>
<td>+$5</td>
</tr>
<tr>
<td>THTTT</td>
<td>5</td>
<td>+$3</td>
</tr>
<tr>
<td>THHTHTHTTTT</td>
<td>11 flips</td>
<td>-$3</td>
</tr>
</tbody>
</table>

- 0.5000 to 0.9999 corresponds to tail

**Monte Carlo Simulation**: A type of simulation where a simulation is run a large number of times and the average is taken to determine the outcome. For example, running coin flipping game 100 times and then averaging the result to see whether there is a better chance of winning or losing is an example of Monte Carlo simulation.

One can also study the $M/M/1$ queueing theory model (Poisson input, exponential service time, and single server) using simulation.

### 3 Common Applications of Simulation

Simulation is exceptionally versatile technique. It can be used in various application such as:

- Design and operation of queueing systems
- Managing inventory systems
- Estimating probability of completing a project by a given deadline
- Design and operation of manufacturing systems
- Design and operation of distributed systems
- Financial risk analysis
- Health care analysis

### 4 Generation of Random Numbers

We already know that implementing a simulation model requires random numbers in order to obtain random observations from probability distributions. A **random number generator** is an algorithm that produces sequences of numbers that follow a specified probability distribution and possess the appearance of randomness. Take note that no computer can create a truly random number.
4.1 Characteristics of Random Numbers

- From any uniform random variable we can generate any other distribution of random variable.

A uniform random number is a random observation from a (continuous) uniform distribution over some interval \([a,b]\). The probability density function of this uniform distribution is

\[
f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}
\]

For a given random integer number in the range 0 to \(\bar{n}\), dividing this number by \(\bar{n}\) yields a uniform random number (if \(\bar{n}\) is small, this approximation should be improved by adding \(\frac{1}{2}\) to the random integer number and then dividing by \(\bar{n} + 1\) instead).

4.2 Congruential Methods for Random Number Generation

The main objective here is to generate a sequence of random numbers. We can represent the sequence as \(\{x_n\}_{n=1}^{\infty}\). The mixed congruential method generates a sequence of random numbers over the range from 0 to \(m - 1\). The method always calculates the next random number from the last one obtained, given an initial random number \(x_0\), called the seed. In particular, it calculates \((n + 1)\)th random number \(x_{n+1}\) from the \(n\)th random number \(x_n\) by using recurrence relation

\[
x_{n+1} = (ax_n + c) \pmod{m},
\]  

where \(a, c,\) and \(m\) are positive integers and \(a, c < m\). The possible values of \(x_{n+1}\) are \(0, 1, \ldots, m - 1\).

4.3 Test for Randomness

There are some properties we need to check to ensure the generated sequence has random properties.

- Low frequency of occurrence
- Little to no correlation between numbers
- \(a\) and \(m\) being large prime numbers generally works
5 Generating Nonuniform Random Variables

5.1 Inverse Transformation Method

The inverse transformation method can sometimes be used to generate random observations. Letting $X$ be the random variable involved, we denote the cumulative distribution function by

$$F(x) = P\{X \leq x\}.$$ 

Generating each observation then requires the following two steps:

1. Generate a uniform random number, $y$, between 0 and 1.
2. Set $F(x) = y$ and solve for $x = F^{-1}(y)$, which is the desired random observation from the probability distribution. Figure 1 shows the graphical method.

This works when $F^{-1}$ is easy to calculate. For example, Excel uses $\text{NORMINV(RAND(), } \mu, \sigma)$ to generate a random observation from a normal distribution with mean $\mu$ and standard deviation $\sigma$.

![Graphical method of inverse transformation method](image)

Figure 1: The inverse transformation method can be used to generate a random number from a probability distribution $F(x)$.

5.2 Acceptance Rejection Method

For many continuous distributions, it is not feasible to apply the inverse transformation method because $x = F^{-1}(r)$ can not be computed. The acceptance-rejection method is an-
other possible method. Consider, the *triangular distribution* having probability distribution

\[
f(x) = \begin{cases} 
  x & 0 \leq x \leq 1 \\
  2 - x & 1 \leq x \leq 2 \\
  0 & \text{otherwise}
\end{cases}
\]

The acceptance-rejection method uses the following two steps to generate a random observation.

1. Generate a uniform random number \( r_1 \) between 0 and 1, and set \( x = 2r_1 \) (so that range of possible values of \( x \) is 0 to 2)

2. Accept \( x \) with

\[
\text{Probability} = \begin{cases} 
  x & 0 \leq x \leq 1 \\
  2 - x & 1 \leq x \leq 2
\end{cases}
\]

to be the desired random observation (since this probability equals \( f(x) \)). Otherwise reject \( x \) and repeat the two steps. In general, to generate the event of accepting (or rejecting) \( x \) according to this probability, the method implements step 2 as follows:

Generate a uniform random number \( r_2 \) between 0 and 1.

\[
\begin{align*}
\text{Accept } x & \text{ if } r_2 \leq f(x) \\
\text{Reject } x & \text{ if } r_2 > f(x)
\end{align*}
\]

If \( x \) is rejected, repeat the two steps. Due to \( x = 2r_1 \) being accepted with a probability = \( f(x) \), the probability distribution of accepted values has \( f(x) \) as its density function, so accepted values are valid random observations from \( f(x) \). If the largest value of \( f(x) \) for any \( x \), say \( L \), is not equal to 1, then \( r_2 \) should be multiplied by \( L \) in step 2.

Note, all figures taken from [1].

**References**