Math/Stat 2300 **Simulation Modeling**, Part I (March 30/April 1)

from text *A First Course in Mathematical Modeling*, Giordano, Fox, Horton, Weir, 2009.

Sometimes to obtain data it is not possible to conduct experiments or observe the behaviour. Conducting experiments can be prohibitively expensive. The system to conduct the experiment may not even exist.

In cases where the behaviour cannot be explained analytically or data collected directly, the modeler might simulate the behaviour indirectly in some manner and then test the various alternatives to estimate how each affects the behaviour. Then the data can be collected. This is called **simulation** and is examined in Chapter 5 of the text.

There are lots of types of simulation, but we will focus on **Monte Carlo Simulation**. **Monte Carlo Simulation** uses random numbers. We will talk about how to generate random numbers in section 5.2. We will mostly use Maple's random number generator function.

The phrase "A sequence of random numbers uniformly distributed in an interval m to n" means a set of numbers with no apparent pattern, where each number between m and n can appear with equal likelihood.

Example. If you toss a six-sided die 100 times and wrote down the numbers showing on the die each time, you would end up with a sequence of 100 random integers uniformly distributed between 1 and 6.

Probabilistic processes are processes with an element of chance involved. The opposite of probabilistic is **deterministic**.
Monte Carlo simulation is a probabilistic model.

Examples:
The area under a curve is **deterministic** (even if we can't find it precisely).
The time between arrivals of customers at an elevator on a particular day is **probabilistic**.

Observed behaviour can be deterministic or probabilistic. A model can be deterministic or probabilistic.

Some advantages of Monte Carlo Simulation:

1. relative ease with which it can sometimes be used to approximate very complex probabilistic systems
2. provides performance estimation over a wide range conditions rather than a very restricted range as often required by an analytic model
3. particular submodel can be changed easily (which has the potential of conducting a sensitivity analysis)
4. modeler has control over the level of detail in a simulation

Some disadvantages of Monte Carlo Simulation:

1. typically expensive to develop and operate : often requires many hours to construct; large amounts of computer time and memory to run.
2. probabilistic nature of simulation model limits the conclusions that can be drawn from a particular run unless a sensitivity analysis is conducted.

Simulating Deterministic Behaviour: Area under a curve (5.1)

Goal: Find an approximate value to the area under a nonnegative value.

Suppose $y = f(x)$ is some given continuous function satisfying $0 \leq f(x) \leq M$ over the closed interval $a \leq x \leq b$. Here M is an upper bound for $f(x)$.

We select a point $P(x, y)$ at random from within the rectangular region. We do this by generating two random numbers, x and y, satisfying $a \leq x \leq b$ and $0 \leq y \leq M$.

Once $P(x, y)$ is selected, we ask whether it lies within the region below the curve. Does y satisfy $0 \leq f(x) \leq M$?

Then count P by adding 1 to some counter. We count the total number of points generated and we count the number of points that lie below the curve.

Then we can calculate an approximate value for the area under the curve.

$$\frac{\text{area under the curve}}{\text{area of the rectangle}} \approx \frac{\text{number of points counted below curve}}{\text{total number of random points}}$$

Monte Carlo technique is probabilistic and typically requires a large number of trials before the deviation between the predicted and true values becomes small.

Note: the number of trials needed to ensure a predetermined level of confidence requires more knowledge of statistics. The general rule is that in order to double accuracy of the result, four times as many experiments are necessary.

Monte Carlo Area Algorithm

Input: Total number of n of random points to be generated in the simulation

Output: Approximate area under the specified curve $y = f(x)$ over the given interval $a \leq x \leq b$ where $0 \leq f(x) \leq M$.

Step 1: Initialize COUNTER=0

Step 2: for $i = 1$ to n

Step 2a: calculate random coordinates x_i and y_i that satisfy $a \leq x_i \leq b$ and $0 \leq y_i \leq M$

Step 2b: calculate $f(x_i)$ for the random x_i coordinate

Step 2c: if $y_i \leq f(x_i)$, then increment COUNTER by 1

Step 3: Calculate the area

$$A = M(b-a) \times \frac{COUNTER}{n}$$

Step 4: Output A.

Generating Random Numbers (5.2)

An important part of simulation is the generation of random numbers. So far we have used the Maple function `rand` to generate random numbers.

A computer does not really generate random numbers because computers employ deterministic processes. Computers generate a pseudorandom numbers. We will look two methods for generating pseudorandom numbers ourselves.

Middle-Square Method

Their middle-square method works as follows:

1. Start with a four-digit number x_0, called the **seed**.
2. Square it to obtain an eight-digit number (add a leading zero if necessary)
3. Take the middle four digits as the next random number.

Continuing , we obtain a sequence that appears random over the integers from 0 to 9999. These integers can be scaled to any interval a to b.

Example. Let's start with the seed $x_0 = 2041$.

n	x_n	square x_n
0	2041	04165681
1	1656	02742336
2	7423	55100929
3	1009	01018081
4	0180	00032400
5	0324	00104976
6	1049	01100401
7	1004	01008016
8	0080	00006400
9	0064	00004096
10	0040	00001600
11	0016	00000256
12	0002	00000004

One disadvantage: this method has a tendency to degenerate to zero.

Linear Congruence Method

This method is as follows:

- Choose three integers a, b and c
- Given an initial seed x_0, we generate a sequence using the rule

$$x_{n+1} = (ax_n + b) \bmod c$$

mod$c$ means to obtain the remainder after dividing the quantity $ax_n + b$ by c

Example. Let $a = 1$, $b = 7$ and $c = 10$. Then

$$x_{n+1} = (1 \cdot x_n + 7) \bmod 10$$

If $x_n = 115$, then

$$x_{n+1} = \text{remainder}\left(\frac{115+7}{10}\right) = \text{remainder}\left(\frac{122}{10}\right) = 2$$

One problem with any algorithm for generating pseudorandom numbers is **cycling**. Cycling is when out sequence of random numbers repeats itself.

This linear congruence method produces a sequence of integers between 0 and $c - 1$. Cycling is guaranteed with at most c random numbers in the sequence. To combat this, we choose c very large. Many computers use $c = 2^{31}$.

Simulating Probabilistic Behaviour (5.3)

Here, we will create simulation models for simple probabilistic behaviour. This type of simulation can be extended to more complex probabilistic behaviour.

Flipping a Fair Coin

The probability of obtaining heads or tails on a flip of a coin is $\frac{1}{2}$. What happens when we actually flip a coin? Will one out of every two slips be tails?

Probability is a long run average. Thus, in the long run, the ratio of the number of tails to the number of flips approaches 0.5.

Define $P(x)$ as follows where x is a random number between $[0, 1]$:

$$P(x) = \begin{cases} \text{heads} & 0 \leq x \leq \frac{1}{2} \\ \text{tails} & \frac{1}{2} < x \leq 1 \end{cases}$$

SO, $P(x)$ assigns either "heads" or "tails" to a number between $[0, 1]$.

Monte Carlo Fair Coin Algorithm

Input: Total number n of random flips of a fair coin to be generated in the simulation

Output: Probability of getting "tails" when we flip a fair coin.

Step 1: Initialize: COUNTER=0

Step 2: for $i = 1$ to n

Step 2a: generate a random number x_i between 0 and 1

Step 2b: if $0.5 < x \leq 1$, then COUNTER=COUNTER+1

Step 3: Calculate $P(tails) = COUNTER/n$

Step 4: Output Probability of tails, $P(tails)$

Rolling a Fair Die

The probability of the event of rolling a '1' is $\frac{1}{6}$. The probability of a particular number appearing on the die is

$$\frac{\text{number of occurences of the particular number}}{\text{total number of trials}}$$

Monte Carlo Roll of a Fair Die Algorithm

Input: Total number n of random rolls of a die to be generated in the simulation

Output: Probability of rolling a $[1, 2, 3, 4, 5, 6]$

Step 1: Initialize COUNTER1 through to COUNTER6 to zero

Step 2: for $i = 1$ to n

Step 2a: generate a random number x_i between 0 and 1

Step 2b: if x_i belongs to these intervals, then increment the appropriate counter

$$\begin{aligned}
0 \leq x_i \leq \frac{1}{6} \quad & COUNTER1 = COUNTER1 + 1 \\
\frac{1}{6} < x_i \leq \frac{2}{6} \quad & COUNTER2 = COUNTER2 + 1 \\
\frac{2}{6} < x_i \leq \frac{3}{6} \quad & COUNTER3 = COUNTER3 + 1 \\
\frac{3}{6} < x_i \leq \frac{4}{6} \quad & COUNTER4 = COUNTER4 + 1 \\
\frac{4}{6} < x_i \leq \frac{5}{6} \quad & COUNTER5 = COUNTER5 + 1 \\
\frac{5}{6} < x_i \leq 1 \quad & COUNTER6 = COUNTER6 + 1
\end{aligned}$$

Step 3: Calculate probability of each roll $j = \{1, 2, 3, 4, 5, 6\}$ by $COUNTER(j)/n$

Step 4: Output probabilities