Mathematical Modeling
Lia Vas

Simulation Modeling. Random Numbers

In many cases one of the following situations might occur:

- It is not possible to observe the behavior directly or to conduct experiments.
- A chance plays a part in the data.
- The system that we need to test does not exist yet. For example, it would be too expensive to create a system that we need to study.

In all the cases above, a model might be **simulated** and then test how a change in data would impact the behavior of the system. Such a model is refer to as **probabilistic** or **stochastic** (as opposed to **deterministic**).

One of the most frequently used methods of simulation is called **Monte Carlo simulation**. This method uses a large number of **random numbers** to generate a model. Using this method even a complex systems can be easily be described. However, sometimes it may require a lot of time and computer memory to be performed.

Any algorithm of simulating the behavior of a real system requires a random number generator. A computer does not really generate random numbers because computer employs a deterministic algorithm but a list of pseudo-random numbers which can be considered random. There are many algorithms for computing random numbers and there is not a single best among them.

Most programing languages have built-in random number generators (Excel, TI83+, Matlab all have it). In Matlab, the command **rand(1)** returns a random number between 0 and 1 assuming uniform distribution. We can build other random variables using **rand**. For example, to get a random number between a and b we can use $a+$**rand(1)**$(b-a)$. To get a 0 or 1 on a random way in Matlab, you can use **round(rand(1))**. The function **round(x)** returns 0 of $x \leq 1/2$ and returns 1 if $x > 1/2$.

The function **round** is built-in in Matlab so you can use it without entering the M-file round given below.

```
function y=round(x)
if x<=1/2
      y=0;
else y=1;
end
```

Another frequently used probability distribution is the normal distribution. To get values of the normal distribution with a mean of μ and standard deviation of σ^2, you can use

$$\sigma(-2\ln(\mathbf{rand(1)}))^{1/2}\cos(2\pi\ \mathbf{rand(1)}) + \mu.$$

An important fact to note is that a single run of the program might not be sufficient even if the number of events is very large. In many instances a single run of the program might just tell you if the program is written correctly and if it is behaving as it should. In practice, the objectives for simulation model very often may include the following:

1. Collecting statistics on the long-term behavior of the system.
2. Comparing alternative arrangements of the system. Investigating the effects of changing the parameters or the modeling assumptions.
3. Finding the optimal operating conditions for the system.
4. Monitoring how the initial conditions influence the running time. Sometimes certain choices of random numbers can create the 'artificial state' of the system which might halt the calculations before the system transitions into a 'busy state'.

Example 1. Let us suppose that we need to find the area under a curve $y = f(x)$ for $a \leq x \leq b$ assuming that $0 \leq y \leq M$. Instead of developing a deterministic algorithm involving determining the indefinite integral and then using the fundamental theorem of calculus to evaluate the definite integral, a different approach can be used.

1. Select a point (x, y) from the rectangle $a \leq x \leq b$, $0 \leq y \leq M$ randomly.
2. Calculate $f(x)$. If $y \leq f(x)$ the point is contributing to the area under the curve. If not, then it is above the curve and it should not be counted towards the area. Repeat this step for a large number of random points from the rectangle.
3. Find the area under the curve by noting that

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

It is easy to convert the steps above into a formal algorithm. The input is a function $f(x)$, and values of M, a, b and n. The output will be the area approximated using the Monte Carlo method.

```
function A=MC_area(f, M, a, b, n) (you should inline $f$ as a function of $x$ before executing the program)
c=0;        (set counter to 0 initially)
for i=1:n
   x=a+rand(1)*(b-a);        ($x$ is a random number between $a$ and $b$)
   y =rand(1)*M;        ($y$ is a random number between 0 and $M$)
   if y<=f(x)
      c=c+1;        (the counter is increased if point is on or below f(x), otherwise the counter is not increased)
   end        (end of if branch)
end        (end of for loop)
```

```
A=M*(b-a)*c/n;      (area = area of rectangle times ratio of the counter value and the total
number of points)
```

Example 2. Use Monte Carlo method to calculate the probability of three heads occurring when five fair coins are flipped (alternatively, the probability of getting three heads in five flips of one coin).

The input of the M-file calculating this probability is the number of tosses and the output is the probability of three heads occurring. Let 0 denote the tail and 1 denote the head.

```
function P=three_heads(n)
c=0;      (set counter to 0 initially)
for i=1:n
   sum5 = round(rand(1))+round(rand(1))+round(rand(1))+round(rand(1))+round(rand(1))
(calculates the result of 5 tosses, for example if the outcome is 2 heads and 3 tails, the value of
sum5 is 1+1+0+0+0=2)
   if sum5=3          c=c+1;      (sum5=3 corresponds to exactly 3 heads in five tosses.
This is a positive outcome and we increase counter by 1 in this case)
   end
end
P=c/n;
```

Example 3. Let us consider the probability model in which each event is not equally likely. Assume that we have a die that is loaded or braised according to the following distribution:

Roll value	Probability
1	0.1
2	0.1
3	0.2
4	0.3
5	0.2
6	0.1

The input is the total number of rolls and the output is the probability of each event.

```
function P=unfair(n)
P1=0; P2=0; P3=0; P4=0; P5=0; P6=0;      (set all 6 counters to 0 initially)
for i=1:n
   x=rand(1);      (x is a random number between 0 and 1)
   if x<=0.1      P1=P1+1;
      else if x<=0.2     P2=P2+1;      (0.2 is the sum of 0.1 and 0.1)
         else if x<=0.4     P3=P3+1;      (0.4 is the sum of 0.1, 0.1 and 0.2)
            else if x<=0.7     P4=P4+1;      (0.6 is the sum of 0.1, 0.1, 0.2 and 0.3)
               else if x<= 0.9     P5=P5+1;      (0.9 is the sum 0.1+0.1+0.2+0.3+0.2)
                  else P6=P6+1;
               end
            end
         end
```

```
        end
    end        (end five if branches)
end        (end for loop)
P1=P1/n; P2=P2/n; P3=P3/n; P4=P4/n; P5=P5/n; P6=P6/n;
P=[P1, P2, P3, P4, P5, P6];
```

It is easy to modify the program above if the die is fair (i.e. if the probability of each event is exactly 1/6).

Example 4. A train should leave a station at precisely 1 pm. However, it happens that the train is late some times. The train is on time in about 70% of cases, it is 5 minutes late in about 20% of cases and it is 10 minutes late in about 10% of cases. You are getting on the train at the next station. The train journey time is about 30 minutes with variations of 2 minutes. You are arriving to your station exactly at 1:30 in 40% of cases, at 1:28 in 30% of cases, at 1:32 in 20% of cases and at 1:34 in 10% of cases. Find the probability that you will catch the train.

Let t_1 stands for the departure time of the train, t_2 for the journey time and t_3 for your arrival time to the station. The favorable event is when $t_1 + t_2 > t_3$. For simplicity, let us consider time 1:00 pm as $t = 0$ and let us measure time in minutes.

An algorithm for simulating this situation is as follows

1. Input: number of events n
2. Output: the probability that you will catch the train = the number of favorable events divided by the total number of events n.
3. Initially set counter c of favorable events to 0.
4. t_1 is a discrete random variable that takes value 0 in 70% of cases, 5 in 20% of cases and 10 in 10% of cases. For t_2 we can assume the normal distribution with mean 30 and deviation 2. t_3 is a discrete random variable that takes value 28 in 30% of cases, 30 in 40% of cases, 32 in 20% of cases and 34 in 10% of cases. Thus,

$t1$	0	5	10
frequency	.7	.2	.1

$t3$	28	30	32	34
frequency	.3	.4	.2	.1

5. For $i = 1$ to n, repeat the following steps: calculate t_1, t_2 and t_3 and increase the counter by 1 if $t_1 + t_2 > t_3$.
6. Calculate the quotient of c and n.

This algorithm converts to the following M-file.

```
function P=train(n)
c=0;
for i=1:n
    x=rand(1)
    if x<.7     t1=0;
        else if x<.9     t1=5;     (.9 is .7+.2)
```

```
      else t1=10;      (in this case x > .9)
    end
  end
  y=rand(1)
  if y<.3    t3=28;
    else if y<.7     t3=30;     (.7 is .3+.4)
      else if y<.9     t3=32;     (.9 is .3+.4+.2)
        else    t3=34;
      end
    end
  end
  z=-2*log(rand(1))
  t2= 2*sqrt(z)*cos(2*pi*rand(1))+30;     (t2 has the normal distribution with mean
30 and standard deviation 2)
  if t1+t2>t3
    c=c+1;
  end
end
P=c/n;
```

Practice Problems.

1. Write a program in Matlab that calculates the volume under a surface $f(x, y)$ for $a \leq x \leq b$, $c \leq y \leq d$ if $0 \leq z \leq M$ using n random points. Test the validity of the program by comparing the outputs with the area obtained using double integrals.

2. Write an algorithm or program in Matlab that calculates an approximation of π by considering that the quotient between a quarter of the unit circle in the first quadrant and the unit square in the first quadrant is $\frac{\pi/4}{1} = \frac{\pi}{4}$.

3. An experiment consists of 5 trials of the same event. The experiment is considered successful just if all five trials yield a favorable outcome. Write down a simulation program that calculates the probability of success in the experiment if a trial is successful in 70% of cases.

4. Assume that you are looking at the outcomes of the two experiments. The first experiment is successful in 70% of cases and the second experiment can be run just if the outcome of the first experiment is favorable. If the second experiment is conducted, it is successful in 80% of cases. Write down a simulation program that calculates the probability of success in both experiments.

Solutions.

1. For example, you can test your problem on a plane $z = 1 - x - y$ over square $0 \leq x \leq 1$, $0 \leq y \leq 1$. Here $0 \leq z \leq 1$. You should get 1/6 as your answer.

2. Write an algorithm that picks a point from a unit square in the first quadrant. Set a counter initially to 0. If a point is below the circle, increase the counter by 1. If not, do not increase it. The quotient of the counter value c and the total number of points n used is equal to the quotient of the area of the circle $\pi/4$ and the area of the square 1. Thus, π is approximately equal to 4 times c/n.

3. Input: number of times n the whole experiment is conducted. Set counter c of successful experiments to 0 initially. Write down a subprocedure similar to **round** that returns 0 in 30% of cases and 1 in 70% of cases. Then, for $i = 1 : n$ repeat the following steps: 1) calculate the sum of 5 random numbers that are 0 in 30 and 1 in 70% of cases. 2) if this sum is 5, increase the counter by 1. Otherwise do not increase it. Finally, calculate the quotient of your counter c and n. The probability should turn up to be 16.8%.

4. The probability should be 56%.