

MATH3290 Mathematical Modeling

Tutorial 4

24th October 2018

Outline

1

Simulation Modeling

- Why do we need simulation?
- Generating Random Numbers

2

The generation of other random variables

- Discrete random variables
- Continuous random variables

Why do we need simulation?

Introduction

The main use of simulation is to approximate quantities that are difficult to compute exactly.¹

¹P. Olofsson. Probabilities: the little number that rule our lives. 2015.

Why do we need simulation?

Example: Social wealth distribution

Suppose that:

- 1 There are 100 people in a room.
- 2 Everyone has \$100 at the beginning.
- 3 For each person, he/she will *randomly* give another person (in the room, can be him/herself) \$1 per minute.

Several assumptions:

- 1 Assume that we can generate a set of random integers from 1 to 100 in `MATLAB`.
- 2 The distributed result for each person is independent.
- 3 It is allowed that one may have debt.

Question: What is the long-term distribution of the wealth?

Why do we need simulation?

Example: Social wealth distribution (Cont.)

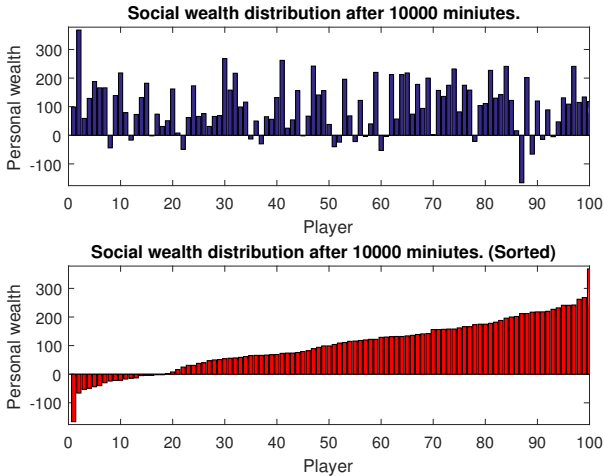


Figure: Social wealth distribution after 10000 minutes.

Why do we need simulation?

Example: Social wealth distribution (Cont.)

What can we learn from the result above?

- If one has more than \$100 finally, refer him/her to be a *rich* person. Then, the ratio of the rich = 50%.
- The first 20% richest people have 45.24% of total wealth.
- The first 10% richest people have 25.19% of total wealth.
- 18 people go bankrupt.

Why do we need simulation?

Some Extensions

- How does the wealth distribution change if one is not allowed to have debt?
That is, one may not give people money if they have no money but they can receive from others.
- Are the initial money and number of people crucial in this example?
- Is there an explanation of the result above from the point of view of sociology?
- Is there a theoretical formula for the social wealth distribution?

Pseudorandom v.s. purely random

Many programming languages have built-in *random number generators* for user such as `rand` or `randn` in MATLAB, to generate *pseudorandom number*, since in principle computer is a deterministic machine and it should not exhibit random behavior. Here is an interesting number:

0.814723686393179².

This is the first number produced by the MATLAB random number generator with its default settings.

²C. B. Moler. Numerical Computing with MATLAB. 2013.

A random sequence is a vague notion . . . in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests traditional with statisticians . . . (Lehmer, 1951)



Figure: D. H. Lehmer (1905-1991).

How does computer generate pseudorandom number

In general, two ways to generate *pseudorandom* numbers:

- Middle-Square method;
- Linear Congruence.

Middle-Square Method

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.

For example, take $x_0 = 2014$ and $x_0^2 = 04056196$. Generating 8 random numbers in this way yields

n	0	1	2	3	4	5	6	7
x_n	2014	0561	3147	9036	6492	1460	1316	7318

More digits can be used, say 6 digits and take $x_0 = 653217$, its square $x_0^2 = 426,692,449,089$ has 12 digits. Thus, take the middle 6 digits as the random number, namely, 692449.

Linear Congruence

Given an initial seed x_0 , generate a sequence by the rule

$$x_{n+1} = (a \times x_n + b) \mod (c),$$

where a is the multiplier, b is the increment and c is the modulus (Getting the remainder). For example, with $a = 1$, $b = 7$ and $c = 10$,

$$x_{n+1} = (1 \times x_n + 7) \mod (10)$$

means x_{n+1} is the integer remainder upon dividing $x_n + 7$ by 10. Thus if $x_0 = 7$, then we have

n	0	1	2	3	4	5	6	7	8	9	10
x_n	7	4	1	8	5	2	9	6	3	0	7

Linear Congruence (Cont.)

- The first 10 terms $\{x_1, \dots, x_{10}\}$ are a permutation of the integers from 0 to 9. It has a period equal to 10. That is, $x_n = x_{n+10}$ for any $n \in \mathbb{N}$.
- In general, for a large integer $c \in \mathbb{N}$ (say $c = 2^{31}$), the period of the above *pseudorandom* sequence is equal to c .
- If the *pseudorandom* sequence with values between 0 and $c - 1$ is scaled by dividing by c , the result is floating-point numbers uniformly distributed in the interval $[0, 1]$ discretely. For $c = 10$, our simple example begins with

n	0	1	2	3	4	5	6	7	8	9
x_n	7	4	1	8	5	2	9	6	3	0
$x_n/10$	0.7	0.4	0.1	0.8	0.5	0.2	0.9	0.6	0.3	0

- The smallest value is 0 and the largest one is 0.9 only.

Generation of discrete random variable

Given that $S = \{x_1, \dots, x_N\}$ is a finite set. Define $p_i := P(X = x_i)$ such that $\sum_{i=1}^N p_i = 1$. Next, we define the following function g

$$g(z) = \begin{cases} x_1, & 0 < z \leq p_1 \\ x_2, & p_1 < z \leq p_1 + p_2 \\ \vdots & \vdots \\ x_N, & p_1 + \dots + p_{N-1} < z \leq 1 \end{cases}$$

Assume that Z is a random variable uniformly distributed in $[0, 1]$. Then $g(Z)$ is our desired random variable. For example, to simulate the outcome from rolling a fair die, one can take $g(z) = \lceil 6 \times z \rceil$. When $0 < Z \leq \frac{1}{6}$, $g(Z)=1$ and when $\frac{1}{6} < Z \leq \frac{1}{3}$, $g(Z) = 2$ and so on.

Inverse transform sampling

Inverse transform sampling is a basic method generating sample numbers at random from any probability distribution given its cumulative distribution function.

Theorem

*Suppose that X is a random variable (r.v.) with **cumulative distribution function** (cdf) F . Then, $F(X)$ is a r.v. with uniform distribution in $[0, 1]$.*

Inverse transform sampling (Cont.)

Recall the definition of characteristic function of X :

$$\varphi_X(t) = \mathbb{E}(e^{itx}) = \int_{\mathbb{R}} e^{itx} dF(x).$$

Note that the distribution of a random variable is uniquely determined by its characteristic function. Next, we calculate $\varphi_{F(X)}$ as follows

$$\varphi_{F(X)}(t) = \int_{\mathbb{R}} e^{itF(x)} dF(x) = \int_0^1 e^{ity} dy \quad (y = F(x)).$$

On the other hand, if $U \sim \text{Unif}(0, 1)$, then

$$\varphi_U(t) = \int_{\mathbb{R}} e^{itx} dF_U(x) = \int_0^1 e^{itx} dx,$$

as the probability density function of U is non-zero only in $[0, 1]$.

Inverse transform sampling (Cont.)

Therefore, $F(X) \sim \text{Unif}(0, 1)$. If we can generate samples U 's that are of uniform distribution in $(0, 1)$ and F is given, then we may generate samples of X 's by $F^{-1}(U)$.

Example

Suppose that X is a random variable with the following probability density function f :

$$f(x) = \begin{cases} 3x^2 & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

The cdf of X is

$$F(x) = \int_{-\infty}^x f(t) dt = \begin{cases} 0 & x \leq 0 \\ x^3 & 0 < x < 1 \\ 1 & x \geq 1 \end{cases}$$

Then $F^{-1}(z) = z^{1/3}$ for any $z \in (0, 1)$.