CENG 222 - Chapter 5: Computer Simulations and Monte Carlo Methods

Burak Metehan Tunçel - June 2022

1 Simulation of Random Variables

1.1 Random Number Generators

Obtaining a good random variable is not a simple task. In the fields sensitive to good random numbers, their generation is typically related to an accurate measurement of some physical variable, for example, the computer time or noise.

More often than not, a *pseudo-random number generator* is utilized. This is nothing but a very *long list of numbers*. A user specifies a *random number seed* that points to the location from which this list will be read. It should be noted that if the same computer code is executed with the same seed, then the same random numbers get generated, leading to identical results. Often each seed is generated within the system, which of course improves the quality of random numbers.

Instead of a computer, a **table of random numbers** is often used for small-size studies.

One way or another, a random number generator or a table of random numbers delivers to us Uniform random variables $U_1,\ U_2,\ \ldots \in (0,\ 1)$. The next three subsections show how to transform them into a random variable X with the given desired distribution $F(x)$.

NOTATION

$U;\ U_1,\ U_2,\ \ldots =$ generated Uniform(0, 1) random variables

1.2 Discrete Methods

At this point, we have obtained one or several independent Uniform(0,1) random variables by means of a random number generator or a table of random numbers. Variables from certain simple distributions can be immediately generated from this.

Example 1 (Bernoulli)

First, simulate a Bernoulli trial with probability of success p. For a Standard Uniform variable U, define

$$X = \begin{cases} 1 & \text{if } U < p \\ 0 & \text{if } U \geq p \end{cases}$$

We call it "a success" if $X = 1$ and "a failure" if $X = 0$. Using the Uniform distribution of U, we find that

$$\boldsymbol{P}\{\text{ success }\} = \boldsymbol{P}\{U < p\} = p$$

Thus, we have generated a Bernoulli trial, and X has Bernoulli distribution with the desired probability p, see Figure 1a.
A MATLAB code for this scheme is

```
U = rand;
X = (U < p)
```

Example 2 (Binomial)

Once we know how to generate Bernoulli variables, we can obtain a Binomial variable as a sum of n independent Bernoulli. For this purpose, we start with n Uniform random numbers, for example:

```
n = 20; p = 0.68;
U = rand(n, 1); % generates an n × 1
                % vector of uniform
                % random numbers
X = sum(U < p)
```

Example 3 (Geometric)

A while-loop of Bernoulli trials will generate a Geometric random variable. We run the loop of trials until the first success occurs. Variable X counts the number of failures:

```
X = 1; % Need at least one trial
while rand > p; % Continue while there
                % are failures
  X = X+1;
end; % Stop at the first success
X
```

Example 4 (Negative Binomial)

Just as in Example 2, once we know how to generate a Geometric variable, we can generate a number of them and obtain a Negative Binomial(k, p) variable as a sum of k independent Geometric(p) variables.

```
p = 0.16; X = 0; i = 0;
while i < k;
  G = 1;
  while rand > p;
    G = G+1;
  end;
  X = X+G;
end;
```

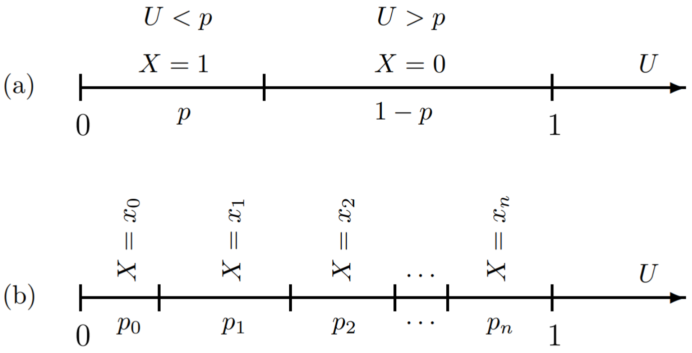

Figure 1

1.2.1 Arbitrary Discrete Distribution

Example 1 can be extended to any arbitrary discrete distribution. In this Example, knowing that the random number generator returns a number between 0 and 1, we divided the interval $[0,\ 1]$ into two parts, p and $(1-p)$ in length. Then we determined the value of X according to the part where the generated value of U fell, as in Figure 1a.

Now consider an arbitrary discrete random variable X that takes values x_0, x_1, ... with probabilities p_0, p_1, ...,

$$p_i = \boldsymbol{P}\left\{X = x_i\right\}, \qquad \sum_{i=1} p_i = 1$$

The scheme similar to Example 1 can be applied as follows.

Algorithm 5.1 (Generating Discrete Variables)

1. Divide the interval $[0,\ 1]$ into subintervals as shown in Figure 1b,

   $$\begin{aligned} A_0 &= [0,\ p_0) \\ A_1 &= [p_0,\ p_0 + p_1) \\ A_1 &= [p_0 + p_1,\ p_0 + p_1 + p_2) \\ &\quad \text{etc.} \end{aligned}$$

   Subinterval A_i will have length p_i; there may be a finite or infinite number of them, according to possible values of X.

2. Obtain a Standard Uniform random variable from a random number generator or a table of random numbers.

3. If U belongs to A_i, let $X = x_i$.

From the Uniform distribution, it follows again that

$$\boldsymbol{P}\left\{X = x_i\right\} = \boldsymbol{P}\left\{U \in A_i\right\} = p_i$$

Hence, the generated variable X has the desired distribution.

Notice that contrary to Examples 2 and 4, this algorithm is economic as it requires only one Uniform random number for each generated variable.

Values x_i can be written in any order, but they have to correspond to their probabilities p_i.

Example 5 (Poisson)

Let us use Algorithm 5.1 to generate a Poisson variable with parameter $\lambda = 5$. Recall that a Poisson variable takes values $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, ... with probabilities

$$p_i = \boldsymbol{P}\left\{X = x_i\right\} = e^{-\lambda}\frac{\lambda^i}{i!} \text{ for } i = 0, 1, 2, ...$$

Following the algorithm, we generate a Uniform random number U and find the set A_i containing U, so that

$$p_0 + \ldots + p_{i-1} \leq U < p_0 + \ldots + p_{i-1} + p_i$$

or, in terms of the cumulative distribution function,

$$F(i-1) \leq U < F(i)$$

This can be done in MATLAB by means of a while-loop:

```
lambda = 5; % Parameter
U = rand; % Generated Uniform variable
i = 0; % Initial value
F = exp(-lambda); % Initial value, F(0)
while (U >= F); % The loop ends when U < F(i)
  F = F + exp(-lambda) * lambda^i/gamma(i+1);
  i = i + 1;
end;
X=i
```

1.3 Inverse Transform Method

As Algorithm 5.1 applies to all discrete distributions, we now turn attention to the generation of continuous random variables. The method will be based on the following simple yet surprising fact.

Theorem 1

Let X be a continuous random variable with cdf $F_X(x)$. Define a random variable $U = F_X(X)$. The distribution of U is Uniform(0,1).

Proof. First, we notice that $0 \leq F(x) \leq 1$ for all x, therefore, values of U lie in $[0,\ 1]$. Second, for any $u \in [0,\ 1]$, find the cdf of U,

$$\begin{aligned} F_U(u) &= \boldsymbol{P}\left\{U \leq u\right\} \\ &= \boldsymbol{P}\left\{F_X(X) \leq u\right\} \\ &= \boldsymbol{P}\left\{X \leq F_X^{-1}(u)\right\} \\ &= F_X(F_X^{-1}(u)) \\ &= u \end{aligned}$$

In the case if F_X is not invertible, let $F_X^{-1}(u)$ denote the smallest x such that $F_X(x) = u$.

We see that U has cdf $F_U(u) = u$ and density $f_U(u) = F_U'(u) = 1$ for $0 \leq u \leq 1$. This is the Uniform(0, 1) density, hence, U has Uniform(0, 1) distribution. □

Regardless of the initial distribution of X, it becomes Uniform(0,1), once X is substituted into its own cumulative distribution function!

1.3.1 Arbitrary Continuous Distribution

In order to generate variable X with the given continuous cdf F, let us revert the formula $U = F(X)$. Then X can be obtained from a generated Standard Uniform variable U as $X = F^{-1}(U)$.

Algorithm 5.2 (Generating Continues Variables)

1. Obtain a Standard Uniform random variable from a random number generator.
2. Compute $X = F^{-1}(U)$. In other words, solve the equation $F(X) = U$ for X.

Example 6 (Exponential)

How shall we generate an Exponential variable with parameter λ? According to Algorithm 5.2, we start by generating a Uniform random variable U. Then we recall that the Exponential cdf is $F(x) = 1 - e^{-\lambda x}$ and solve the equation

$$1 - e^{-\lambda x} = U$$

The result is

$$X = -\frac{1}{\lambda}\ln(1-U) \tag{1}$$

Can this formula be simplified? By any rule of algebra, the answer is “no”. On the other hand, $(1-U)$ has the same distribution as U, Standard Uniform. Therefore, we can replace U by $(1-U)$, and variable

$$X_1 = -\frac{1}{\lambda}\ln(U) \tag{2}$$

although different from X, will also have the desired Exponential(λ) distribution.

Example 7 (Gamma)

Gamma cdf has a complicated integral form, not mentioning F^{-1}, needed for Algorithm 5.2. There are numerical methods of solving the equation $F(X) = U$, but alternatively, much simpler, for any integer α, a Gamma variable can be generated as a sum of α independent Exponential variables:

```
    X = sum( -1/lambda * log(rand(alpha,1)) )
```

1.3.2 Discrete Distributions Revisited

Algorithm 5.2 is not directly applicable to discrete distributions because the inverse function F^{-1} does not exist in the discrete case. In other words, the key equation $F(X) = U$ may have either *infinitely many roots* or *no roots* at all.

Moreover, a discrete variable X has a finite or countable range of possible values, and so does $F(X)$. The probability that U coincidentally equals one of these values is 0. We conclude that the equation $F(X) = U$ has *no roots* with probability 1.

Let us modify the scheme in order to accommodate discrete variables. Instead of solving $F(x) = U$ exactly, which is impossible, we solve it approximately by finding x, the smallest possible value of X such that $F(x) > U$.

The resulting algorithm is described below.

Algorithm 5.3 (Generating Discrete Variables, Revisited)

1. Obtain a Standard Uniform random variable from a random number generator or a table of random numbers.
2. Compute $X = \min\{x \in S \text{ such that } F(x) > U\}$, where S is a set of possible values of X.

Algorithms 5.1 and 5.3 are equivalent if values x_i are arranged in their increasing order in Algorithm 5.1.

Example 8 (Geometric revisited)

Applying Algorithm 5.3 to Geometric cdf

$$F(x) = 1 - (1-p)^x$$

we need to find the smallest integer x satisfying the inequality (solve it for x)

$$x > \frac{\ln(1-U)}{\ln(1-p)}$$

v The smallest such integer is the ceiling

$$X = \left\lceil \frac{\ln(1-U)}{\ln(1-p)} \right\rceil \tag{3}$$

1.3.3 Exponential-Geometric Relation

The formula (5.3) appears very similar to (5.1). With $\lambda = -\ln(1-p)$, our generated Geometric variable is just the ceiling of an Exponential variable! In other words, the ceiling of an Exponential variable has Geometric distribution.

This is not a coincidence.

- Exponential is the continuous analogue of geometric
- Both have the memoryless property.