

## II. Simulation

For additional reading on this topic, see Chapter 22 of the text, in particular, sections 22.1, 22.3, 22.4, 22.5, and 22.6.

### 1 Some examples

**Example 2.1.1:** (Coin Tossing Game) Someone offers you a chance to play a game that involves simply tossing a fair coin 10 times. You win if you get a string of at least 3 consecutive heads. You'll be charged \$1 to play the game, but if you win, you'll earn \$25. Should you play the game? (Assume you are risk-neutral.)

**Solution Approach:** You determine whether to play the game by determining whether playing or not playing results in a higher expected income (since you are risk neutral.) Clearly, your expected outcome from not playing is zero. So you want to determine whether your expected income from playing the game is positive. To do this, you would determine the probability  $p$  that in 10 tosses you get a string of at least 3 consecutive heads. To ensure a positive expected income from playing, it is required that  $p > 1/25$ . What is  $p$ ?

(1) Analytical Approach

Using your knowledge of probability, you could compute  $p$ . It might take a while!

What if you'd never taken a course in probability before? You might use the following experimental solution strategy instead.

(2) Experimental Approach:

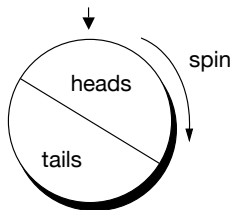
Before committing to playing the game, go toss a coin 10 times and observe whether you get at least 3 consecutive heads. Repeat this experiment a reasonably large number of times ... say 20. Compute the proportion of the 20 experiments that resulted in a success. Call this proportion  $\hat{p}$ . Your hope is that  $\hat{p}$  is a pretty good estimate of the real probability  $p$  that we want to know.

Naturally we would prefer having an exact expression for the probability  $p$  to having a statistical estimate, so the analytical approach is the correct strategy whenever it's possible. But there are many problems (including this example) where the analytical approach is difficult, if not impossible, to carry out. For these problems, we use the experimental approach. But obviously the experimental approach suggested above has its flaws. Tossing a coin 200 times and recording the results would be a time consuming and tedious task! Luckily, we can get a computer to do it for us. Suppose that the computer contained a "genie" that would spit out a sequence of i.i.d. uniform  $[0,1]$  random variables on demand. To "simulate" a single coin toss experiment on the computer, we would do the following

(1) Ask the genie for a uniform  $[0,1]$  random variable.

- (2) If the random variable is less than or equal to  $1/2$ , we label the coin tails; otherwise we call it heads.

Effectively, what we are doing is asking the genie to spin a roulette wheel. We have divided the roulette wheel in half, with one half labeled heads and the other half labeled tails. Having the pointer pointing to tails at the end of the spin corresponds to our uniform random variable being less than  $1/2$ .



Note that the probability of this experiment resulting in heads is exactly  $1/2$ . So we have accurately “simulated” the behavior of a fair coin. If we repeat these steps 200 times, we have effectively simulated 200 fair coin tosses.

**Example 2.1.2:** (Blackjack) Suppose you’ve devised a Blackjack strategy that you think you will help you win big in Vegas. Before going, you would like to test how effective your strategy is. Your hope is that your strategy would give you better than 50% chance of winning against the house. One obvious approach to determine your odds is to experiment with a deck of cards. Deal out many, many rounds of Blackjack . . . say, 100. If you win more than 50 of the hands, you feel encouraged. What’s wrong with this approach? In addition to being time consuming, there’s another problem. Even the best card-counters only have a minimal edge over the house (say 51% or 52% odds.) A mere 100 shuffles is not likely enough to give you an accurate assessment of the odds of winning to within a percent or two. You would need a much larger number of repetitions of the game. In this situation, you *really* need a computer.

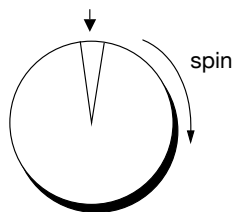
Let’s ask for help from our genie again. To simulate a random shuffle of the deck, start by labeling the 52 cards by numbers 1 through 52. To deal the first card,

- (1) Ask the genie for a uniform  $[0,1]$  random variable. Call it  $U$ .
- (2) The first card drawn is the one labeled  $\lceil 52U \rceil$ .

Note: For a number  $x$ ,  $\lceil x \rceil$  is the smallest integer greater than or equal to  $x$ .

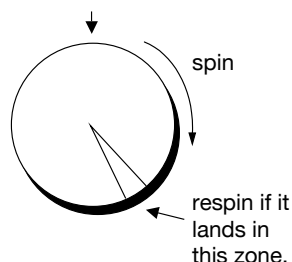
$$P(\lceil 52U \rceil = j) = P(j - 1 < 52U \leq j) = P\left(\frac{j-1}{52} < U \leq \frac{j}{52}\right) = \frac{1}{52}$$

Again, this experiment is just like having the genie spin a roulette wheel that is divided into 52 regions, where each region corresponds to a particular card in the deck. The probability of landing in any one particular one of those regions is  $1/52$ .



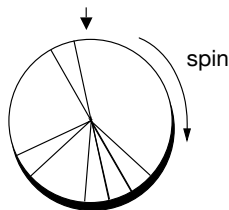
Thus, the above steps are simulating the random draw of a card from a deck of 52. Now to get the second card,

- (1) Ask the genie for a uniform  $[0,1]$  random variable  $U$ .
- (2) Let  $I = \lceil 52U \rceil$ . If the  $I^{\text{th}}$  card in the deck is the same as the first card drawn, then return to step 1. Otherwise, the second card is labeled  $I$ .



In general, to get the  $k^{\text{th}}$  card,

- (1) Ask the genie for a uniform  $[0,1]$  random variable  $U$ .
- (2) Let  $l = \lceil 52U \rceil$ . If the  $l^{\text{th}}$  card in the deck matches the label of any of the first  $k - 1$  cards drawn, then return to step 1. Otherwise, the  $k^{\text{th}}$  card is labeled  $l$ .



## 2 Random Number Generation

Since computers do not contain genies, we hope there's a more scientific approach to generating uniform random numbers. It turns out there is. We are going to describe a class of mathematical algorithms that are widely used for generating uniform  $[0,1]$  random numbers. Since the results of mathematical algorithms are reproducible (and thus predictable), such algorithms used to produce random numbers are called **pseudorandom number generators**. In what follows, when we refer to the output of such an algorithm as a random number, we really mean that it's a pseudo-random number. Often we are interested in simulating situations that call for distributions other than uniform. Luckily, there are ways to convert uniform random numbers into random numbers that follow other distributions; we'll discuss this in detail later.

**Generating Uniform Random Numbers: Linear Congruential Generators**

Suppose that  $x_0$  is a large nonnegative integer. Consider the sequence of numbers that would be generated from the following recursion:

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

Here,  $a$ ,  $b$ , and  $m$  are nonnegative integers. The mathematical operator “mod” stands for modulo. It means:

$y \bmod m$  = the remainder when we divide  $m$  into  $y$

To make this clearer, let’s look at some examples:

$$\begin{aligned} 3 \bmod 7 &= 3 \\ 12 \bmod 4 &= 0 \\ 9 \bmod 6 &= 3 \\ 27 \bmod 16 &= 11 \end{aligned}$$

Let’s look at the sequence of integers that are generated by our recursion

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

Let’s take  $x_0 = 7$ ,  $a = 5$ ,  $b = 3$ ,  $m = 16$ . Then

$$\begin{aligned} x_1 &= (5 \cdot 7 + 3) \bmod 16 = 38 \bmod 16 = 6 \\ x_2 &= (5 \cdot 6 + 3) \bmod 16 = 33 \bmod 16 = 1 \\ x_3 &= (5 \cdot 1 + 3) \bmod 16 = 8 \bmod 16 = 8 \\ x_4 &= (5 \cdot 8 + 3) \bmod 16 = 43 \bmod 16 = 11 \\ x_5 &= (5 \cdot 11 + 3) \bmod 16 = 58 \bmod 16 = 10 \end{aligned}$$

What’s happening is that an unpredictable<sup>1</sup> sequence of integers between 0 and 15 is produced. So if we divide each  $x_i$  by 16, we are getting a sequence of “uniform random numbers” between 0 and 1.

The algorithm

$$\begin{aligned} x_{n+1} &= (ax_n + b) \bmod m \\ U_{n+1} &= x_{n+1}/m \end{aligned}$$

for generating a sequence of numbers  $U_0, U_1, U_2, \dots$  is called a linear congruential generator (LCG). The integer  $a$  is called the multiplier of the LCG,  $b$  is called its increment, and  $m$  is called its modulus. The integer  $x_0$  is called its seed.

Observe that these generators exhibit periodic behavior. For after at most  $m + 1$  iterations of the recursion, some value in the sequence of  $x_i$ ’s must be repeated. From that point onwards, the

<sup>1</sup>Of course, if you know the recursion and the values of  $m$ ,  $a$  and  $b$ , the sequence is completely predictable! But otherwise, the sequence has the appearance of randomness.

generator will repeat the rest of the sequence again and again. The sequence will repeatedly cycle through the same numbers. This is undesirable . . . but unavoidable. To ameliorate the situation, we want to choose the parameters  $x_0$ ,  $a$ ,  $b$ , and  $m$  in such a way that the length of a period (or cycle) is very long, so that the repetition only begins after an enormous number of iterations.

As we mentioned already, the parameter  $m$  of an LCG is the longest possible period that the LCG can have before it begins to repeat itself. But merely choosing a large value for  $m$  is not enough to guarantee a long period for an LCG ( $m$  is only an upper bound on the period.) To guarantee that an LCG has a full period (i.e., of length  $m$ ) for every possible choice of seed  $x_0$ , we rely on some higher mathematics, namely number theory. It tells us that if the following three conditions on  $a$ ,  $b$  and  $m$  hold, then the LCG will have full period for every possible choice of seed  $x_0$ :

- (1) The only positive integer that exactly divides both  $m$  and  $b$  is 1.
- (2) If  $q$  is a prime number dividing  $m$ , then  $q$  divides  $a - 1$ .
- (3) If 4 divides  $m$ , then 4 divides  $a - 1$ .

**Example 2.2.1:** An LCG with  $a = 5$ ,  $b = 3$ , and  $m = 16$  has full period. Of course, in addition to conditions (1) through (3), we would also like it to have a very large choice of  $m$ .

Most computers' random number generators use LCGs in which  $b = 0$ . Such LCGs are called multiplicative generators. Multiplicative generators do not have full period, since they violate condition 1. However, they can have period  $m - 1$  for every seed  $x_0$  selected from the set  $\{1, \dots, m - 1\}$ . This is true if the multiplicative generator satisfies:

- (1)  $m$  is prime.
- (2) The smallest integer  $l$  for which  $a^l - 1$  is divisible by  $m$  is  $l = m - 1$ .

**Example 2.2.2:** An LCG with  $a = 2$ ,  $b = 0$ , and  $m = 5$  has period  $m - 1 = 4$ . If we are using a multiplicative generator, we would like it to have a very large choice of  $m$  in addition to satisfying (1) and (2).

Once parameters are selected for a random number generator, it must be subjected to a battery of statistical tests in which the appearance of randomness of the numbers produced is validated. One choice of  $a$  and  $m$  that perform well is:

$$\begin{array}{l} m = 2^{31} - 1 \\ a = 630,360,016 \end{array}$$

### **Non-Uniform Random Number Generation**

In our quest to build imitations of real systems via simulation, we will need a way to take uniform random numbers (as obtained from an LCG) and turn them into random numbers that are distributed according to another type of distribution. In the blackjack example, we devised a way to

generate random variables  $X$ , where  $X$  is the label of a card randomly drawn from the deck, from the distribution:

$$P(X = i) = 1/52 \quad \text{for } i = 1, \dots, 52$$

In this case,  $X$  is a discrete random variable. We will consider the problems of how to generate both continuous and discrete random variables by using the output of an LCG.

**A. Continuous Random Variables** Consider the following example:

**Example 2.2.3** (Exponential Interarrivals) Suppose that we want to simulate the queues in a grocery store. We want to mimic a Poisson arrival process with rate  $\lambda$ . Let  $A_n$  be the instant at which customer  $n$  arrives; then  $C_n = A_n - A_{n-1}$  represents the interarrival time between the  $(n-1)^{th}$  and  $n^{th}$  customers. We know that the random variables  $C_n$  are exponentially distributed with parameter  $\lambda$ . How do we generate the  $C_n$ 's?

Before we solve this specific problem, let's state the question in more general terms:

Given a cumulative distribution function (cdf)  $F(\cdot)$  and a sequence of i.i.d uniform random variables  $U_0, U_1, U_2, \dots$ , how do we generate a sequence of i.i.d. random variables  $X_0, X_1, X_2, \dots$ , with distribution  $F(\cdot)$ ?

### a. Inverse Transformation Method

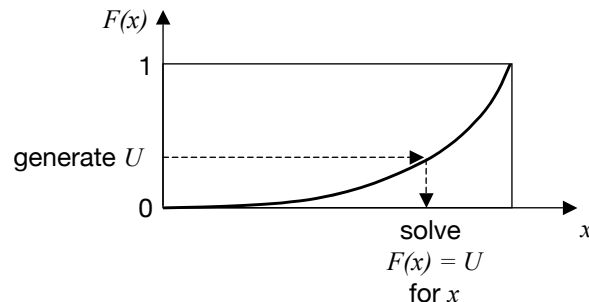
One approach to solving this problem is called the inverse transformation method. This works for random variables whose cumulative distribution function  $F(\cdot)$  can be obtained in closed form. Examples of such random variables include uniform, exponential, triangular and Weibull random variables. We shall see some examples later; first, we describe the approach:

#### INVERSE TRANSFORMATION METHOD

To generate a random number from the cumulative distribution function  $F(x)$ :

- (1) Generate a uniform  $[0, 1]$  random number  $U$ .
- (2) Set  $F(X) = U$  and solve for  $X$ . Return  $X$ .

Here's a graphical depiction of what's happening in the inverse transformation method:



Why does the inverse transformation method work? Suppose  $X$  is obtained using the inverse transformation method. Then

$$X = F^{-1}(U)$$

where  $U$  is a uniform $[0,1]$  random variable. To prove that  $x$  has cumulative distribution function given by  $F(\cdot)$ , observe:

$$\begin{aligned} P(X \leq a) &= P(F^{-1}(U) \leq a) \\ &= P(F(F^{-1}(U)) \leq F(a)) \\ &= P(U \leq F(a)) \\ &= F(a) \quad \text{since } U \sim \text{uniform } [0, 1] \text{ and } F(a) \in [0, 1] \end{aligned}$$

**Example 2.2.4:** (Generating exponential random variables) Back to our grocery store example. We want to generate random variables from an exponential distribution with parameter  $\lambda$ . The cumulative distribution function for an exponential with parameter  $\lambda$  is

$$F(x) = 1 - e^{-\lambda x} \quad \text{for } x > 0.$$

First we generate a uniform  $[0,1]$  random variable  $U$ . Then we set

$$F(X) = 1 - e^{-\lambda X} = U$$

Next, we solve for  $x$ :

$$\begin{aligned} 1 - e^{-\lambda X} &= U \\ e^{-\lambda X} &= 1 - U \\ -\lambda X &= \ln(1 - U) \\ X &= -\frac{\ln(1 - U)}{\lambda} \end{aligned}$$

Now  $X$  is a number sampled from the exponential distribution.

There is an even faster way to do this. To see how, observe that if  $U$  is a uniformly distributed random variable on  $[0,1]$ , then  $1 - U$  is also uniformly distributed on  $[0,1]$ . Therefore

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

has the same distribution as

$$X' = -\frac{\ln U}{\lambda}$$

Using the latter method instead of the former saves us one subtraction per generation. This seems like a small savings, but it yields a considerable improvement in efficiency when we are generating an enormous number of exponential random variables.

**Example 2.2.5** (Generating uniform  $[a, b]$  random variables) Suppose we'd like to generate random variables that are uniformly distributed on the interval  $[a, b]$ . The cdf for this distribution is

$$F(x) = \begin{cases} 0 & \text{for } x < a \\ \frac{x - a}{b - a} & \text{for } a \leq x \leq b \\ 1 & \text{for } x > b \end{cases}$$

To use the inverse transformation method to generate observations from a uniform  $[a, b]$  random variable, first we generate a uniform  $[0,1]$  random variable  $U$ . Then we set

$$\frac{X - a}{b - a} = U$$

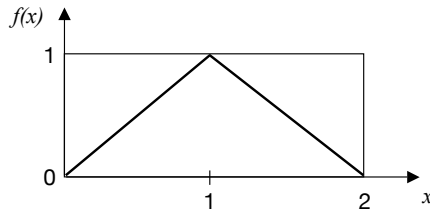
Next, we solve for  $X$ :

$$X = a + (b - a)U$$

Now  $X$  is a uniform  $[a, b]$  random number.

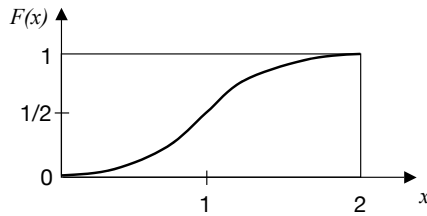
**Example 2.2.6** Suppose we'd like to generate a random variable with the following density function:

$$f(x) = \begin{cases} x & \text{for } 0 \leq x \leq 1 \\ 2 - x & \text{for } 1 \leq x \leq 2 \end{cases}$$

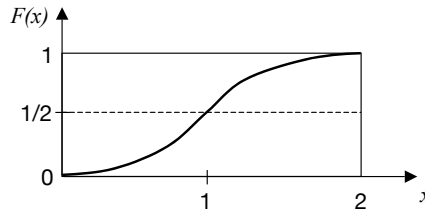


First compute the cumulative distribution function,  $F$ :

$$F(x) = \begin{cases} \frac{x^2}{2} & \text{for } 0 \leq x \leq 1 \\ 2x - \frac{x^2}{2} - 1 & \text{for } 1 \leq x \leq 2 \end{cases}$$



Now generate  $U \sim$  uniform  $[0, 1]$ .



*Case 1:*  $U \leq 1/2$ . Then let

$$\begin{aligned} \frac{X^2}{2} &= U \\ X &= \pm\sqrt{2U} \\ \Rightarrow X &= \sqrt{2U} \quad \text{since } X < 0 \text{ is not possible} \end{aligned}$$

Case 2:  $U \geq 1/2$ . Then let

$$\begin{aligned} 2X - \frac{X^2}{2} - 1 &= U \\ X^2 - 4X + 2 &= -2U \\ (X - 2)^2 &= 2 - 2U && \text{note } 0 \leq 2 - 2U \leq 1, \text{ since } 1/2 \leq U \leq 1 \\ X &= 2 \pm \sqrt{2 - 2U} \\ \Rightarrow X &= 2 - \sqrt{2 - 2U} && \text{since } X > 2 \text{ is not possible} \end{aligned}$$

### b. Acceptance/Rejection Method

Some types of random variables are difficult to generate using the inverse transformation method. Consider the following example:

**Example 2.2.7:** (When inversion is difficult) Consider a random variable with the density function

$$f(x) = \begin{cases} 6x(1-x) & \text{for } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

To use inversion, we would first compute the cdf  $F(x)$ :

$$F(x) = \int_0^x f(a) da = 6 \int_0^x a(1-a) da = 3x^2 - 2x^3 \quad \text{for } 0 \leq x \leq 1$$

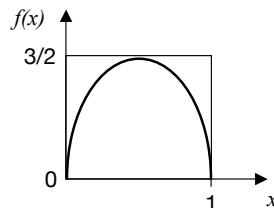
Then we generate a uniform  $[0,1]$  random variable, and would set

$$F(X) = 3X^2 - 2X^3 = U$$

Unfortunately it's difficult to solve this cubic equation in terms of  $X$ . So inversion is hard to use here. But luckily, there's an alternative.

Generate a point in the shaded rectangle at right by generating 2 uniform random variables  $U_1$  and  $U_2$  and taking the point with coordinates  $(U_1, 3U_2/2)$ :

Generate a point in the shaded rectangle at right by generating 2 uniform random variables  $U_1$  and  $U_2$  and taking the point with coordinates  $(U_1, 3U_2/2)$ :

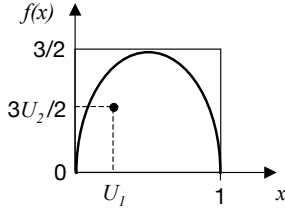


The acceptance/rejection method can be applied to generate random variables when the density function of the random variable is defined over a finite interval  $[a, b]$ . The general method is stated as follows:

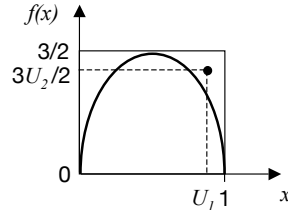
It may require several loops before a random number is successfully generated. Specifically, the expected number of loops required to generate one random variable  $X$  is  $M(b-a)$ .

To show that the A/R method works, we want to show that for any iteration,

$$P(X \leq \alpha | X \text{ accepted}) = F(\alpha) = \int_a^\alpha f(z) dz$$

**Case 1: Acceptance**

If the generated point  $(U_1, 3U_2/2)$  lies below the curve  $f(x)$ , return  $x = U_1$ .

**Case 2: Rejection**

If the generated point  $(U_1, 3U_2/2)$  lies above the curve  $f(x)$ , start over with a new pair  $U_1, U_2$ .

## ACCEPTANCE/REJECTION METHOD

To generate a random number from the density function  $f(x)$  defined on the interval  $a \leq x \leq b$ :

- (1) Let  $M = \max_{a \leq x \leq b} f(x)$
- (2) Generate two uniform  $[0, 1]$  random numbers,  $U_1$  and  $U_2$ .
- (3) If  $MU_2 \geq f(a + (b - a)U_1)$ , return to step 2. Otherwise, return

$$X = a + (b - a)U_1$$

**Proof:**

$$\begin{aligned}
 P(X \leq \alpha | X \text{ accepted}) &= \frac{P(X \leq \alpha, X \text{ accepted})}{P(X \text{ accepted})} \\
 &= M(b - a)P(X \leq \alpha, X \text{ accepted}) \\
 &= M(b - a)P(a + (b - a)U_1 \leq \alpha, MU_2 \leq f(a + (b - a)U_1)) \\
 &= M(b - a)P\left(U_1 \leq \frac{\alpha - a}{b - a}, U_2 \leq \frac{f(a + (b - a)U_1)}{M}\right) \\
 &= M(b - a) \int_0^1 P\left(U_1 \leq \frac{\alpha - a}{b - a}, U_2 \leq \frac{f(a + (b - a)U_1)}{M} \mid U_1 = u\right) du \\
 &= M(b - a) \int_0^{\frac{\alpha - a}{b - a}} P\left(U_2 \leq \frac{f(a + (b - a)u)}{M}\right) du \\
 &= M(b - a) \int_0^{\frac{\alpha - a}{b - a}} \frac{f(a + (b - a)u)}{M} du \\
 &= \int_a^\alpha f(z) dz \quad (\text{letting } z = a + (b - a)u, dz = (b - a)du) \\
 &= F(\alpha)
 \end{aligned}$$

**c. Generating Normal Random Variables**

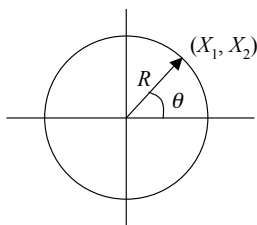
To generate a Normal random variable with mean 0 and variance 1, there is an alternative to using the Inverse Transformation or the Acceptance/Rejection methods. The method that we will

describe generates two independent Normal (0, 1) random variables, and it makes use of some properties of the Normal distribution.

Suppose that we have a random vector  $(X_1, X_2)$ , where  $X_1$  and  $X_2$  are iid random variables with distribution Normal (0,1). If we plot this point on the  $xy$ -plane, we can identify it with its polar coordinates

$$(X_1, X_2) = (R \cos \theta, R \sin \theta),$$

where  $R$  is the radius and  $\theta$  is the angle between the vector and the positive  $x$ -axis.



The idea behind the method is to generate the radius  $R$  and the angle  $\theta$ , and use them to obtain  $X_1$  and  $X_2$ .

First we should recall that if  $X_1, X_2$  are two independent Normal (0, 1) random variables, then

$$R^2 = X_1^2 + X_2^2$$

has distribution Chi Square with two degrees of freedom,  $\chi_2^2$ . We also need to note that a  $\chi_2^2$  random variable has the same distribution as two times an Exponential (1) random variable. That is, if  $Y$  has distribution  $\chi_2^2$ , and  $W$  has distribution Exponential (1), then

$$R^2 = X_1^2 + X_2^2 \stackrel{D}{=} Y \stackrel{D}{=} 2W.$$

We can easily compute  $W$  by inversion according to the formula

$$W = -\log U_1$$

where  $U_1$  is Uniform (0, 1).

It can be shown that the angle  $\theta$  has distribution Uniform (0,  $2\pi$ ), so we can generate  $\theta$  with the formula

$$\theta = 2\pi U_2$$

where  $U_2$  is Uniform (0, 1).

This gives us the following algorithm:

- (1) Generate two independent Uniform (0, 1) random variables (in our notation,  $U_1$  and  $U_2$ )
- (2) Compute the two Normal (0, 1) random variables according to

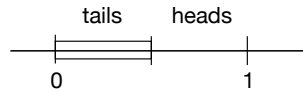
$$\begin{aligned} X_1 &= \sqrt{2(-\log U_1)} \cos(2\pi U_2) \\ X_2 &= \sqrt{2(-\log U_1)} \sin(2\pi U_2) \end{aligned}$$

**B. Discrete Random Variables**

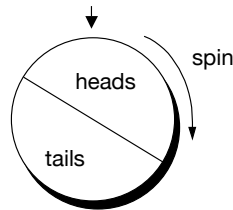
It is especially easy to generate discrete random variables. Recall the coin flipping example. In that example, we were effectively generating random variables  $X$  according to the Bernoulli ( $1/2$ ) distribution:

$$P(X = 0) = 1/2 \quad P(X = 1) = 1/2$$

where tails are labeled 0 and heads labeled 1. We generated a uniform  $[0,1]$  random variable. Then we said that if the resulting random variable was less than  $1/2$ , the coin would be labeled tails (i.e.,  $X = 0$ ); otherwise it would be heads ( $X = 1$ ).



Another way to look at it is that we divided a roulette wheel in half and labeled one half heads and the other tails. A spin of the wheel generates a random variable with the desired distribution.



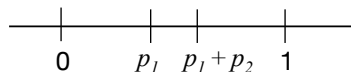
We can generalize this method considerably to more complicated types of discrete random variables. Suppose we want to generate a random variable  $X$  with the probability mass function

$$P(X = x_1) = p_1$$

$$P(X = x_2) = p_2$$

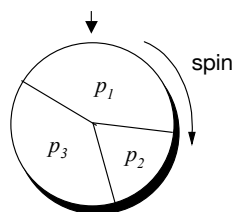
$$P(X = x_3) = p_3$$

To generate a random variable  $X$  according to this distribution, we subdivide the interval into 3 subintervals of length  $p_1$ ,  $p_2$ , and  $p_3$ :



Then (as usual) we generate a uniform  $[0,1]$  variable  $U$ .

$$\begin{aligned} \text{If } U \leq p_1, & \quad \text{output } X = x_1 \\ \text{If } p_1 < U \leq p_1 + p_2, & \quad \text{output } X = x_2 \\ \text{If } p_1 + p_2 < U \leq 1, & \quad \text{output } X = x_3 \end{aligned}$$



**Example 2.2.8:** (Generating binomial random variables) Suppose we are a widget manufacturing company. We ship truckloads containing 20 widgets to our customers. The probability that any given widget we ship out is defective is  $1/10$ . We'd like to generate a random variable  $X$  that represents the number of defective widgets in a truckload. The random variable  $X$  has a binomial distribution with parameters  $(20, 1/10)$ . To generalize the method described earlier, we subdivide the unit interval into 21 subintervals (one subinterval for each of the possible outcomes  $0, 1, 2, \dots, 20$ ). The length of the  $i^{\text{th}}$  subinterval is

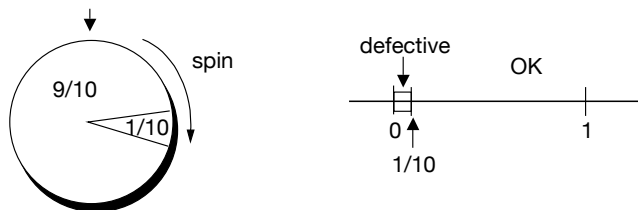
$$P(X = i) = \binom{20}{i} \left(\frac{1}{10}\right)^i \left(\frac{9}{10}\right)^{20-i}$$

We generate a uniform  $[0,1]$  variable  $U$ , and then search the 21 intervals to determine which one contains  $U$ ; the value of  $X$  is then determined by which interval  $U$  fell in.

There's another approach we could use for this problem. Recall that a binomial random variable  $X$  with parameters  $(n, p)$  is equal to the sum of  $n$  i.i.d Bernoulli random variables with parameter  $p$ :

$$X = Y_1 + Y_2 + Y_3 + \dots + Y_n \quad \text{with } Y_i \sim \text{Bernoulli}(p) \text{ for } i = 1, 2, \dots, n$$

So another way to generate  $X$  is just by generating  $n$  Bernoulli ( $p$ ) random variables and summing them. In particular, for our widget example, we would generate 20 Bernoulli ( $1/10$ ) random variables and add them up.



### 3 Overview of Simulation

Now that we know how to generate random variables from various distributions, we are equipped to design simulations. A simulation is a representation of a system as it evolves over time. It consists of a model, or replica, of the system that we are trying to study, and a series of experiments whose results provide an estimate of a certain parameter in which we are interested. For example, in our original coin tossing problem, our system was simply the coin tossing game and its associated rules. Our experiment was a set of 10 coin tosses; we repeated this experiment 20 times, for a total of 200 tosses. The parameter we were trying to estimate was the probability of getting three or more consecutive heads in those 10 tosses. In the blackjack example, our system was the game

of blackjack, our experiment was a single hand of blackjack, and the parameter we were trying to estimate was the probability of beating the house in blackjack. The power of simulation is that it is possible to build representations of extremely complex systems, and it can give insight into such systems when analytical and numerical methods do not suffice. Later, we will discuss the design of simulations. But before we do, we will provide a general framework for the parameter estimation that simulation is used for, and discuss how to go about getting satisfactory estimates for the parameters desired.

In general terms, a simulation is designed to solve the following problem:

Given a random variable  $X$  with (possibly unknown) distribution  $F(x)$ , estimate  $\mu = E(X)$ .

Observe that both our coin tossing and blackjack examples fit into the framework of estimating an expectation  $E(X)$ . In the coin tossing example, our random variable  $X$  is an indicator for success or failure in tossing at least 3 consecutive heads in a string of 10 tosses. So  $X = 1$  if we succeed and  $X = 0$  otherwise. In the blackjack problem, our random variable  $X$  is an indicator for success or failure in a hand of blackjack; it equals 1 when we win and it's zero otherwise. The expectation  $E(X)$  is the probability of success in either game. Notice that in these problems, we know the form of the distribution  $F(x)$ ; it is Bernoulli in both cases. We don't know the parameter of the distribution, however (that's what we're trying to estimate.) But in simulating complex systems, we may not even know the form of the distribution  $F(x)$ . The simulation approach to solving the problem of estimating  $\mu = E(X)$  is:

Estimate  $\mu = E(X)$  by running  $n$  independent and identical experiments, thereby obtaining  $n$  i.i.d. random variables  $X_0, X_1, \dots, X_n$  each having the distribution  $F(x)$ .

## 4 Quantifying the Quality of an Estimate

The question is: how large should  $n$  be? In other words, how many replications of the same experiment should be run in order to get a "satisfactory" estimate? Naturally, the more experiments we run, the better our estimate will be. But we would like a precise way to characterize what a "satisfactory" estimate is. And we would also like to know how many replications are required to achieve it. The Central Limit Theorem will be the key to answering this question. First observe that since our goal is to estimate  $\mu = E(X)$  by generating  $n$  i.i.d. random variables  $X_1, \dots, X_n$  having having the distribution  $F(x)$ , our estimate for  $\mu$  is simply the sample mean, which we'll call:

$$\bar{X}_n = \frac{1}{n}(X_1 + X_2 + \dots + X_n)$$

From the CLT, we know

$$\frac{X_1 + X_2 + \dots + X_n - n\mu}{\sigma\sqrt{n}} \stackrel{D}{\approx} N(0, 1)$$

or equivalently,

$$\bar{X}_n \stackrel{D}{\approx} \mu + \frac{\sigma}{\sqrt{n}}N(0, 1)$$

where  $\mu = E(X_i)$  and  $\sigma^2 = \text{Var}(X_i)$ .

The CLT approximation  $\bar{X}_n \stackrel{D}{\approx} \mu + \frac{\sigma}{\sqrt{n}}N(0, 1)$  is telling us three very important things, namely:

- (1) The error term  $\frac{\sigma}{\sqrt{n}}$  gets smaller at a rate of  $\frac{1}{\sqrt{n}}$ . This means that in order to improve the accuracy of the estimator by a factor of 10, the number  $n$  of replications goes up by a factor of 100.
- (2) The error term is normally distributed. This will be used to obtain “confidence intervals” for  $\mu$ . (Confidence intervals are the measure of how satisfactory an estimate is; we’ll discuss them in more detail later.)
- (3) The only way the specific distribution of the random variable  $X$  affects the accuracy of the estimate is through its standard deviation  $\sigma$ . Thus, the magnitude of  $\sigma$  is a measure of how difficult it is to estimate  $\mu$  via simulation.

Now let’s rewrite the CLT approximation and tackle our estimation problem.

$$\frac{\sqrt{n}}{\sigma}(\bar{X}_n - \mu) \stackrel{D}{\approx} N(0, 1)$$

This is telling us that

$$P\left(-z \leq \frac{\sqrt{n}}{\sigma}(\bar{X}_n - \mu) \leq z\right) \approx P(-z \leq N(0, 1) \leq z)$$

For any particular value of  $z$ , the quantity on the right hand side of the above expression is something we can obtain from a table for the standard normal distribution. (These tables contain values for the cdf  $\Phi(z) = P(N(0, 1) \leq z)$ ). We use this to compute the right hand side:

$$\begin{aligned} P(-z \leq N(0, 1) \leq z) &= P(N(0, 1) \leq z) - P(N(0, 1) \leq -z) \\ &= \Phi(z) - \Phi(-z) \\ &= \Phi(z) - (1 - \Phi(z)) \quad (\text{since by the symmetry of the standard normal} \\ &= 2\Phi(z) - 1 \quad \text{distribution, } \Phi(-a) = 1 - \Phi(a)) \end{aligned}$$

### Building Confidence Intervals

Suppose we choose the value of  $z$  from the normal tables to ensure that

$$P(-z \leq N(0, 1) \leq z) = 1 - \delta$$

for some number  $\delta$  in the interval  $[0,1]$ . This is equivalent to finding  $z$  that satisfies

$$2\Phi(z) - 1 = 1 - \delta$$

or equivalently

$$\Phi(z) = 1 - \frac{\delta}{2}$$

For example, if  $1 - \delta = 0.90$ , then we look up the value of  $z$  on the normal tables that satisfies

$$\Phi(z) = 1 - \frac{\delta}{2} = 1 - .05 = .95$$

and we obtain  $z = 1.65$ .

What are we doing by choosing a value  $1 - \delta$  such that

$$P\left(-z \leq \frac{\sqrt{n}}{\sigma}(\bar{X}_n - \mu) \leq z\right) \approx P(-z \leq N(0, 1) \leq z) = 1 - \delta?$$

To see what we're doing, first notice that the condition  $\frac{\sqrt{n}}{\sigma}(\bar{X}_n - \mu) \leq z$  is equivalent to the condition  $\bar{X}_n - \frac{\sigma z}{\sqrt{n}} \leq \mu$ . Moreover, the condition  $\frac{\sqrt{n}}{\sigma}(\bar{X}_n - \mu) \geq -z$  implies  $\mu \leq \bar{X}_n + \frac{\sigma z}{\sqrt{n}}$ . Using these facts, we see that setting the probability

$$P\left(-z \leq \frac{\sqrt{n}}{\sigma}(\bar{X}_n - \mu) \leq z\right) \approx 1 - \delta$$

is equivalent to setting

$$P\left(\bar{X}_n - \frac{\sigma z}{\sqrt{n}} \leq \mu \leq \bar{X}_n + \frac{\sigma z}{\sqrt{n}}\right) \approx 1 - \delta$$

In words, the above condition means that the probability that the parameter  $\mu$  (which is what we are trying to estimate!) lies in the interval

$$\left[\bar{X}_n - \frac{\sigma z}{\sqrt{n}}, \bar{X}_n + \frac{\sigma z}{\sqrt{n}}\right]$$

is approximately  $1 - \delta$ . This is known as a confidence interval. Thus, for a fixed number of experiments  $n$ , your sample mean  $\bar{X}_n$  falls within the above confidence interval with probability  $(1 - \delta)$ , where  $z$  is obtained by solving

$$\Phi(z) = 1 - \frac{\delta}{2}$$

Notice that this interval is not of much practical use. The reason is that requires knowledge of the standard deviation  $\sigma$  of the random variable whose mean we are trying to estimate. If we don't know the mean, we're not likely to know the standard deviation! So we have more work to do to get a usable confidence interval.

We need to eliminate the unknown parameter  $\sigma$  from the expression to make the confidence interval usable. Note that while we are running our  $n$  experiments to obtain the observations  $X_1, \dots, X_n$  of our random variable  $X$  (and thereby the sample mean  $\bar{X}_n$ ), we can also get an estimate for the variance  $\sigma^2$  of  $X$ :

$$v_n = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2 \approx \sigma^2 \quad \text{for large } n.$$

(There is a straightforward proof that  $v_n$  is an unbiased estimator for  $\sigma^2$ .) Therefore, we can use

$$\sqrt{v_n}$$

as an estimate for  $\sigma$  in our confidence interval:

$$\left[\bar{X}_n - \frac{\sigma z}{\sqrt{n}}, \bar{X}_n + \frac{\sigma z}{\sqrt{n}}\right] \approx \left[\bar{X}_n - \frac{\sqrt{v_n} z}{\sqrt{n}}, \bar{X}_n + \frac{\sqrt{v_n} z}{\sqrt{n}}\right]$$

What we have just shown is that

$$\left[ \bar{X}_n - z\sqrt{\frac{v_n}{n}}, \bar{X}_n + z\sqrt{\frac{v_n}{n}} \right]$$

is an approximate  $(1 - \delta)100\%$  confidence interval for  $\mu = E(X)$ .

In other words, we are  $(1 - \delta)100\%$  confident that the true mean  $\mu$  of the random variable  $X$  is within

$$z\sqrt{\frac{v_n}{n}} \quad \text{of our sample mean } \bar{X}_n$$

or equivalently,

$$P\left(\bar{X}_n - z\sqrt{\frac{v_n}{n}} \leq \mu \leq \bar{X}_n + z\sqrt{\frac{v_n}{n}}\right) \approx 1 - \delta$$

The quantity  $z\sqrt{\frac{v_n}{n}}$  is called the half-width of the confidence interval.

### Algorithm for Computing Confidence Intervals

1. Select  $n$ , the number of experiments to be run, and  $1 - \delta$ , the confidence level.
2. Run  $n$  independent replications of the experiment, obtaining the observations  $X_1, \dots, X_n$  of the random variable  $X$ .
3. Compute the sample mean:  $\bar{X}_n = \frac{1}{n}(X_1 + X_2 + \dots + X_n)$ ,  
and the sample variance:  $v_n = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2$
4. Select  $z$  such that  $\Phi(z) = 1 - \frac{\delta}{2}$ . Then an approximate  $(1 - \delta)100\%$  confidence interval for  $\mu = E(X)$  is:

$$\left[ \bar{X}_n - z\sqrt{\frac{v_n}{n}}, \bar{X}_n + z\sqrt{\frac{v_n}{n}} \right]$$

**Example 2.2.9:** (Estimating the cost of an inventory policy) Suppose we have come up with a policy for managing inventory in our company. To evaluate the effectiveness of the policy, we would like to calculate the expected cost that will be incurred in the next year as a result of using this policy. To use simulation here, we would need to run  $n$  independent replications (or experiments) each of which simulates the costs for the next year. Suppose we want an estimate with a 90% confidence interval (i.e.,  $1 - \delta = 0.9$ ). Suppose  $n = 6$ , and the results of our 6 experiments (i.e., the 6 observations for next year's cost, in millions of dollars) were 7, 11, 4, 9, 7 and 2. Then

$$\begin{aligned}\bar{X}_n &= \frac{1}{n}(X_1 + X_2 + \cdots + X_n) = \frac{1}{6}(7 + 11 + 4 + 9 + 7 + 2) = 6.7 \\ v_n &= \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2 = \frac{1}{5}((7 - 6.7)^2 + (11 - 6.7)^2 + (4 - 6.7)^2 \\ &\quad + (9 - 6.7)^2 + (7 - 6.7)^2 + (2 - 6.7)^2) \\ &= 9.6\end{aligned}$$

Since we want a 90% approximate confidence interval, we have  $1 - \delta = 0.9$ , or  $\delta = 0.1$ . So our next step is to determine the value of  $z$  for which

$$\Phi(z) = 1 - \frac{\delta}{2} = 0.95$$

The desired value of  $z$  is 1.65 (obtained from standard normal table.) So our 90% approximate confidence interval for the mean inventory cost over the next year (in millions of dollars) is

$$\left[ \bar{X}_n - z\sqrt{\frac{v_n}{n}}, \bar{X}_n + z\sqrt{\frac{v_n}{n}} \right] = \left[ 6.7 - 1.65\sqrt{\frac{9.6}{6}}, 6.7 + 1.65\sqrt{\frac{9.6}{6}} \right] = [4.6, 8.8]$$

This is a pretty wide confidence interval. Considering the magnitude of the expense we are talking about, we ought to increase the number of experiments. By how much? This is the original question we set out to answer. It turns out that the value of  $n$  depends on whether we want “absolute precision” or “relative precision”.

### Quantifying the Quality of Estimates: Notions of Precision

#### ABSOLUTE PRECISION

We want an estimate that is accurate to within a certain amount.

In the inventory cost example, saying that we want to estimate the mean cost to within \$500,000 would be an example of seeking absolute precision.

#### RELATIVE PRECISION

We want an estimate that is accurate to within a certain percentage.

Saying that we want to estimate the mean inventory cost to within 10% is an example of seeking relative precision. The method of determining the number  $n$  of experiments to run varies depending on which type of precision we want to achieve.

### Determining the Number of Experiments to Run

**Absolute Precision** Suppose we are seeking absolute precision. In particular, we want an estimate that is accurate to within an amount  $\epsilon$ . Then we should choose  $n$  to ensure that the half-width of our confidence interval is  $\epsilon$ :

$$\frac{z\sigma}{\sqrt{n}} \approx \epsilon$$

or equivalently:

$$n \approx \frac{z^2 \sigma^2}{\epsilon^2}$$

Now, this isn't very useful, because we don't know before we run experiments! To deal with this problem, we do a small number of "trial run" experiments, and use the sample variance obtained from those preliminary experiments in place of  $\sigma^2$  in the above estimate for the number  $n$  of additional experiments to run.

**Example 2.2.10:** (Estimating the cost of an inventory policy, continued) Suppose, for our inventory example, we want to estimate the mean to within \$500,000, and that the 6 experiments we already ran comprise our "trial runs". Here,  $\epsilon = 0.5$ , so

$$n \approx \frac{z^2 v_6}{\epsilon^2} = \frac{(1.65)^2 9.6}{(0.5)^2} = 104.544$$

We should run 104 additional experiments, and use our confidence interval methodology to produce a confidence interval based on the results of these 104 runs.

NOTE: It might be tempting to reuse the initial 6 runs as part of our 104 runs to produce our confidence interval. But reuse of the trial runs can undermine the quality of our final confidence interval; it's not recommended.

**Relative Precision** To get a relative precision interval (accurate to within a percentage  $\epsilon\%$ ) we should choose  $n$  to ensure that the half-width of our confidence interval is  $\epsilon\%$  of the mean:

$$\frac{z\sigma}{\sqrt{n}} \approx \frac{\epsilon}{100} |\mu|$$

or equivalently:

$$n \approx 10,000 \frac{z^2 \sigma^2}{\epsilon^2 \mu^2}$$

Again, we don't know  $\mu$  or  $\sigma^2$  so we need to perform a small number of "trial run" experiments to obtain a preliminary sample mean and sample variance. We use those preliminary estimates in place of  $\mu$  and  $\sigma^2$  in the above estimate for  $n$ .

**Example 2.2.11:** (Estimating the cost of an inventory policy, continued) Suppose that instead of absolute precision, we want to estimate the mean to within 10%, using the 6 experiments we already ran for our "trial runs". Then  $\epsilon = 10$ , and

$$n \approx 10,000 \frac{z^2 v_6}{\epsilon^2 \bar{X}_6^2} = 10,000 \frac{(1.65)^2 9.6}{(10)^2 (6.7)^2} = 58.22$$

We should run 58 additional experiments, and use our confidence interval methodology to produce a confidence interval for  $\mu$  based on the results of these 58 runs. Again, we should avoid the temptation to reuse the results of the initial runs in producing our confidence interval.

**Comparing Relative & Absolute Precision** Suppose that the purpose of our experiments is to estimate a probability  $p$  of a certain event. (In our dice coin tossing problem, for example, we were trying to estimate the probability  $p$  of a string of 3 or more consecutive heads in 10 tosses of a coin.) To generate our estimate, we run a sequence of experiments  $i = 1, 2, \dots, n$  and output  $X_i = 1$  if the event occurs in that experiment and  $X_i = 0$  otherwise. So each  $X_i$  has a Bernoulli( $p$ ) distribution. We know that for a Bernoulli random variable,

$$\mu = p \quad \text{and} \quad \sigma^2 = p(1 - p)$$

Next, we'll use this information to estimate the number  $n$  of runs necessary to estimate  $p$ .

Case 1: Absolute Precision  $\epsilon$

$$n \approx z^2 \frac{\sigma^2}{\epsilon^2} = z^2 \frac{p(1 - p)}{\epsilon^2}$$

Here, the number of runs required gets smaller as  $p \rightarrow 0$  or  $p \rightarrow 1$ .

Case 2: Relative Precision  $\epsilon\%$

$$n \approx 10,000 \frac{z^2 \sigma^2}{\epsilon^2 \mu^2} = 10,000 \frac{z^2(1 - p)}{\epsilon^2 p}$$

Here, the number of runs required gets larger as  $p \rightarrow 0$

So in this setting, the number of runs required depends on which kind of precision we're aiming for.