Lecture 3

Chapter 2. Computer Simulations and Monte Carlo Methods

Figure 1: Monte Carlo Casino

1. Monte Carlo Methods and Random Number Generators

1. Monte Carlo Methods and Random Number Generators

- **Monte Carlo (MC) methods** are methods of approximation (estimation) based on computer simulations involving random numbers;
- the name *does* come from the famous Monte Carlo casino in Monaco (probability distributions involved in gambling are often complicated, but they can be estimated via simulations);
- the main purpose of simulations is estimating quantities whose direct computation is complicated, expensive, time consuming, dangerous, or plainly impossible (think space shuttle launch, spread of a virus or disease, performance of a medical device or procedure, etc.);
- MC methods are mostly used for computation of probabilities, expected values and other distribution characteristics, but they can also be used to estimate lengths, areas, volumes, integrals, irrational numbers (like $\pi, e, \sqrt{2}$), etc.

- recall that the probability can be defined as a long-run proportion (relative frequency);
- with the help of random number generators, computers can actually simulate a "long run";
- the longer the run is simulated, the more accurate the results obtained.

Some examples include:

- **Forecasting**. We already know from Statistics that given a distribution model, it is often very difficult to make reasonably *remote predictions*. Often a one-day development depends on the results obtained during *all* the previous days. Then prediction for tomorrow may be straightforward, whereas computation of a one-month forecast is already problematic.

  On the other hand, simulation of such a process can be easily performed day by day (or even minute by minute). Based on present results, we simulate the next day and then the next and so on. For a time n, we estimate X_n from the (already known) variables $X_1, X_2, \ldots, X_{n-1}$. Controlling the length of this do-loop, we can obtain forecasts for the next days, weeks, months or years. Such simulations can be used to predict weather, profit, stock prices, costs, etc. Simulation of future failures reflects reliability of devices and systems. Simulation of future stock and commodity prices plays a crucial role in finance, as it allows evaluations of options and other financial deals.

- **Signal transmission (percolation)**. Consider a network of nodes (a graph), some nodes connected, others not. A signal is sent from a certain node. Once a node k receives a signal, it sends it along each of its output lines with some probability p_k. After a certain period of time, one wishes to estimate the proportion of nodes that received a signal, the probability for a certain node to receive it, etc.

  That would mean generating Bernoulli variables with parameters p_k. Line k transmits if the corresponding generated variable $X_k = 1$. In the end, we simply count the number of nodes that got the signal, or verify whether the given node received it.

  This general *percolation* model describes the way many phenomena may spread in real life. The role of a signal may be played by a computer virus spreading from one computer to another, or by rumors spreading among people, or by fire spreading through a forest, or by a disease spreading between residents.

- **Markov chain Monte Carlo**. This is a modern technique of generating random variables from rather complex, often intractable distributions, as long as *conditional distributions* have a reasonably simple form. In semiconductor industry, for example, the joint distribution of good and defective chips on a produced wafer (which is a thin slice of semiconductor) has a rather complicated correlation structure. As a result, it can only be written explicitly for rather simplified artificial models. On the other hand, the quality of each chip is predictable based on the quality of the surrounding, neighboring chips. Given its neighborhood, conditional probability for a chip to fail can be written, and thus, its quality can be simulated by generating a corresponding Bernoulli random variable with $X_i = 1$ indicating a failure.

  According to the Markov chain Monte Carlo (MCMC) methodology, a long sequence of random variables is generated from conditional distributions. A wisely designed MCMC will then produce random variables that have the desired *unconditional* distribution, no matter how complex it is.

- **Queuing systems (server facilities)**. A *queuing system* is described by a number of random variables. It involves spontaneous arrivals of jobs, their random waiting time, assignment to servers, their random service and departure time; some jobs may exit prematurely, others may not enter the system if it appears full or busy, also, intensity of the incoming traffic and the number of servers on duty may change during the day.

  One wants to be able to evaluate a queuing system's vital performance characteristics, such as a job's average waiting time, average length of a queue, the proportion of customers who had to wait, the proportion of "unsatisfied customers" (that exit prematurely or cannot enter), the proportion of jobs spending more than a certain time in the system, the expected usage of each server, the average number of available (idle) servers at the time when a job arrives, and so on.

- so, different types of phenomena can be computer-simulated; however, *one* simulation is not enough for estimating probabilities and expectations;
- once we understand how to program the given phenomenon *once*, we can embed it in a do-loop and repeat similar simulations a large number of times, generating a long run;
- since the simulated variables are random, we will generally obtain a number of different realizations, from which we calculate probabilities and expectations as long-run frequencies and averages.

Random Number Generators

- implementation of MC methods reduces to generation of random variables from given distributions; all mathematical and statistical software packages (Matlab, Maple, Mathematica, SAS, R, Splus, SPSS, Minitab, Excel, etc.) have built-in procedures for generating random variables from the most common (discrete or continuous) distributions;
- as it turns out, the main job is that of generating random numbers from a Uniform distribution, in fact, from a Standard Uniform distribution; these can then be used to generate random numbers from any given distribution.

- generating *truly random* Uniformly distributed numbers is not an easy task and is an ongoing research area in Modern Statistics and Stochastic Analysis;
- how do we know that the numbers obtained are "truly random" and do not have any undesired patterns? For example, quality random number generation is so important in coding and password creation that people design special tests to verify the "randomness" of generated numbers;
- more often, pseudo-random (deterministic) numbers are generated, i.e. a long list of numbers; the user specifies a random number *seed* that points to the location from which the list will be read; often each seed is generated within the system, to improve the quality of random numbers.

2. Discrete Methods

2. Discrete Methods

These are methods for generating some *simple discrete variables*, most being specific to certain distributions, by using their interpretation and relationship with other variables, rather than their definition (pdf or cdf).

From here on, we denote by $U \in U(0,1)$ a Standard Uniform variable. Let us recall the pdf and cdf (equation (5.10) in Chapter 1, Lecture 2):

$$f_U(x) = \begin{cases} 1, & x \in [0,1] \\ 0, & x \notin [0,1] \end{cases}, \; F_U(x) = P(U \leq x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 < x \leq 1 \\ 1, & x \geq 1\,. \end{cases} \quad (2.1)$$

Bernoulli Distribution $Bern(p), p \in (0,1)$

Recall that a Bernoulli distribution models the occurrence (or nonoccurrence) of an event (success), with a given probability p.

Let U be a Standard Uniform variable. That means its value is in $(0,1)$, just like the value of p. Then define

$$X = \begin{cases} 1, & U < p \\ 0, & U \geq p \end{cases}, \tag{2.2}$$

i.e. define "success" as $(U < p)$ (and, obviously, "failure" as $(U \geq p)$).
Let us check that this is indeed a Bernoulli variable with parameter p. We have

$$P(X = 1) = P(U < p) = F_U(p) = p,$$

by (2.1), since $p \in (0,1)$. So X has the desired distribution.

Algorithm 2.1.

Read $p \in (0,1)$
$U = rand;$
$X = (U < p);$

- now we can use this simple way of simulating success/failure with a given probability, for all the variables that are defined in terms of that.

Binomial Distribution $B(n,p), n \in \mathbb{N}, p \in (0,1)$

Recall (Remark 5.2 in Chapter 1, Lecture 2) that a Binomial $B(n,p)$ variable is the sum of n independent $Bern(p)$ variables.

Algorithm 2.2.

Read $n \in \mathbb{N}, p \in (0,1)$
$U = rand(n,1);$
$X = sum(U < p);$

Geometric Distribution $Geo(p), p \in (0,1)$

A Geometric variable counts the number of failures that occurred before the 1^{st} success. We can simulate that.

Algorithm 2.3.

Read $p \in (0,1)$
$X = 0$; % initial number of failures
while $rand \geq p$ % while there are failures
  $X = X + 1$; % count number of failures
end % stop at the first success

Remark 2.4.

Obviously, the same algorithm can be used to simulate a $Y \in SGeo(p)$ variable, as well. Y is the number of *trials* needed to get the 1^{st} success, so simply let $Y = X + 1$ in Algorithm 2.3.

Negative Binomial (Pascal) Distribution $NB(n,p), n \in \mathbb{N}, p \in (0,1)$

Again, we use the same Remark 5.2 (in Chapter 1, Lecture 2), which states that a $NB(n,p)$ variable is the sum of n independent $Geo(p)$ random variables.

Algorithm 2.5.

Read $n \in \mathbb{N}, p \in (0,1)$
$X = zeros(1,n)$;
for $i = 1 : n$ % generate n independent $Geo(p)$ variables
  while $rand \geq p$ % while there are failures
    $X(i) = X(i) + 1$; % count number of failures
  end % stop at the first success
end
$Y = sum(X)$; % the sum is a $NBin(n,p)$ variable

Arbitrary Discrete Distribution

Let X be an arbitrary discrete random variable with pdf

$$X \begin{pmatrix} x_0 & x_1 & \cdots \\ p_0 & p_1 & \cdots \end{pmatrix}, \quad \sum p_i = 1. \tag{2.3}$$

- we generalize the idea that was used to generate a $Bern(p)$ variable;
- for that, we basically divided the interval $[0, 1]$ into the disjoint subintervals $[0, p)$ and $[p, p + (1 - p)]$;
- we can do the same for any number of subintervals, as seen in Figure 2.

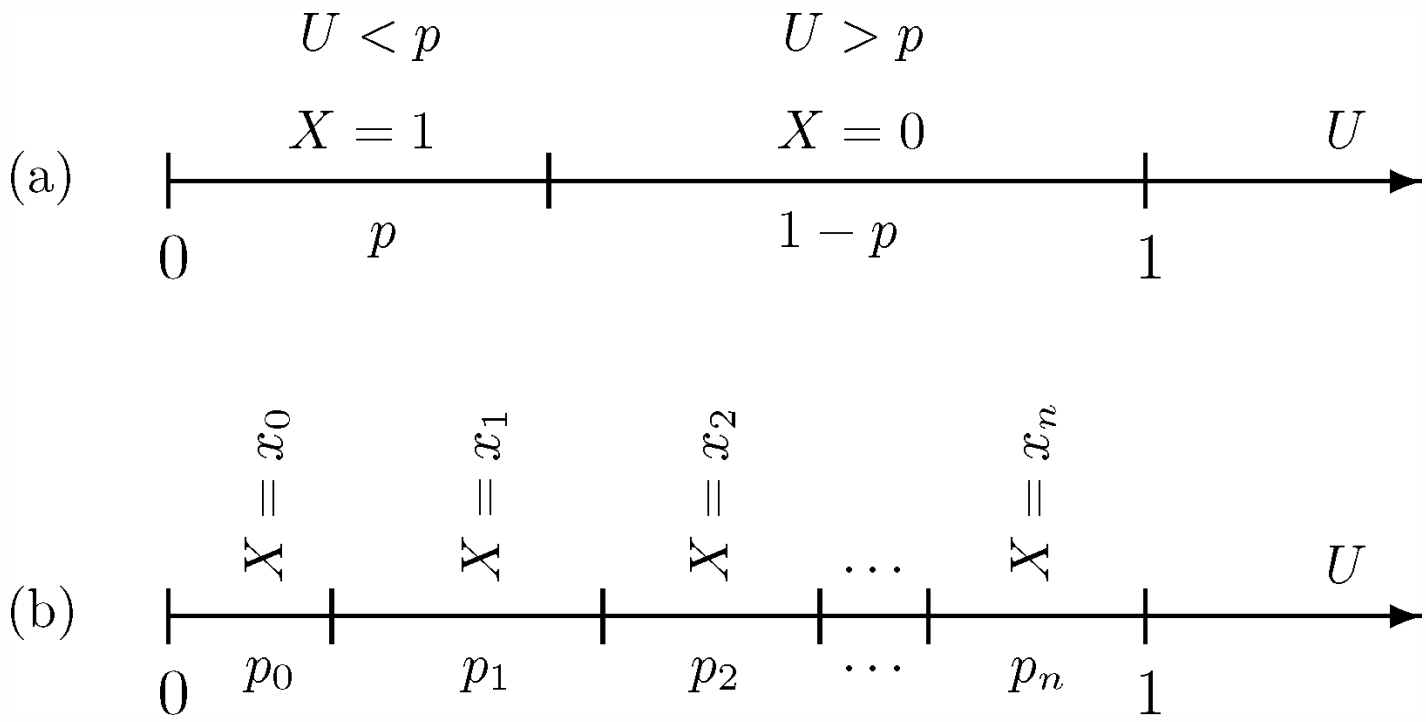

Figure 2: Generating arbitrary discrete distributions

Algorithm 2.6.

- Read $x_i, p_i, \ i = 1, 2, \ldots$
- Divide the interval $[0, 1]$ into subintervals

$$\begin{aligned} A_0 &= [0, p_0) \\ A_1 &= [p_0, p_0 + p_1) \\ A_2 &= [p_0 + p_1, p_0 + p_1 + p_2) \\ &\ldots \\ A_i &= [p_0 + \cdots + p_{i-1}, p_0 + \cdots + p_i) \\ &\ldots \end{aligned}$$

  Then length$(A_i) = p_i$, for a finite or countably infinite number of intervals.
- Get $U \in U(0, 1)$.
- If $U \in A_i$, then let $X = x_i$.

Then, we have

$$\begin{aligned} P(X = x_i) &= P(U \in A_i) \\ &= P\Big(p_0 + \cdots + p_{i-1} \leq U < p_0 + \cdots + p_{i-1} + p_i\Big) \\ &= F_U(p_0 + \cdots + p_{i-1} + p_i) - F_U(p_0 + \cdots + p_{i-1}) \\ &= (p_0 + \cdots + p_{i-1} + p_i) - (p_0 + \cdots + p_{i-1}) \\ &= p_i, \end{aligned}$$

so X has indeed the desired pdf (2.3).

Example 2.7.

Let us use Algorithm 2.6 to generate a variable with pdf

$$X \begin{pmatrix} 0 & 1 & 2 & \dots \\ p_0 & p_1 & p_2 & \dots \end{pmatrix}, \quad \sum_{i \in \mathbb{N}} p_i = 1.$$

Recall that for a discrete random variable, the cdf is computed as

$$F(x) = P(X \leq x) = \sum_{x_i \leq x} p_i,$$

i.e., in this case,

$$F(k) = \sum_{i \leq k} p_i = p_0 + p_1 + \cdots + p_k, \quad k = 0, 1, \dots,$$

so to check if $U \in A_k$, we check that $F(k-1) \leq U < F(k)$.

Algorithm

Read $p_0, p_1, \ldots$

Get $U \in U(0,1)$

$i = 0$; % initial value of X

$F = p_0$; % initial value of cdf $F(0)$

while $U \geq F$ % check if $U \in A_i$

$i = i + 1$;

$F = F + p_i$; % new value of cdf, $F(i+1)$

end % the while loop ends when $U < F(i)$

$X = i$;

We can use this to generate a Poiss(λ) variable, with pdf

$$X \begin{pmatrix} i \\ \frac{\lambda^i}{i!} e^{-\lambda} \end{pmatrix}_{i=0,1,\ldots} = \begin{pmatrix} 0 & 1 & 2 & \ldots \\ e^{-\lambda} & \lambda e^{-\lambda} & \frac{\lambda^2}{2} e^{-\lambda} & \ldots \end{pmatrix}, \quad \lambda > 0.$$

Algorithm 2.8.

Read $\lambda > 0$
$U = rand;$
$i = 0;$
$F = exp(-\lambda);$
while $U >= F$
  $i = i + 1;$
  $F = F + exp(-\lambda) * \lambda\hat{}i/factorial(i);$
end
$X = i;$

3. Inverse Transform Method

This is a method used when we want to generate a random variable whose cdf F does not have a very complicated form. It is based on the following result:

Theorem 3.1.

Let X be a continuous random variable with cdf $F : \mathbb{R} \to \mathbb{R}$. Then $U = F(X) \in U(0, 1)$.

Proof.

So, F is the cdf of X, i.e. $F(x) = P(X \leq x)$, for all $x \in \mathbb{R}$. We will show that U has the $U(0, 1)$ pdf, by starting with its cdf and then taking its derivative.

First off, let us notice that, being a cdf (i.e. a probability), $F(x) \in [0, 1]$, for all $x \in \mathbb{R}$ and, thus, all the values of U are in $[0, 1]$.

Proof.

Secondly, X being a continuous random variable, there exists an interval $[a, b] \subseteq \mathbb{R}$ such that :

- $F : [a, b] \to [0, 1]$ is strictly increasing (therefore injective),
- $F(x) = 0, \forall x < a$ and
- $F(x) = 1, \forall x > b$.

Hence, its inverse $F^{-1} : [0, 1] \to [a, b]$ exists.

Now, let us consider the cdf, F_U.
Let $x \in \mathbb{R}$.
If $x < 0$, then $F_U(x) = P(U \leq x) = P(\text{impossible event}) = 0$.
Hence, $f_U(x) = F'_U(x) = 0$.
If $x > 1$, then $F_U(x) = P(U \leq x) = P(\text{sure event}) = 1$ and thus,
$f_U(x) = F'_U(x) = 0$.
For $x \in [0, 1]$, we have

$$F_U(x) = P(U \leq x) = P(F(X) \leq x) = P\left(X \leq F^{-1}(x)\right) = F\left(F^{-1}(x)\right) = x.$$

Then the pdf is $f_U(x) = F'_U(x) = 1$ and, hence, $U \in U(0, 1)$.

As a consequence, to generate a continuous random variable with given cdf F, we generate a variable $U \in U(0,1)$ and let

$$X = F^{-1}(U). \tag{3.1}$$

Indeed, then the cdf of X is

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

for all $x \in \mathbb{R}$, the last assertion following from the fact that $F(x) \in [0,1]$. Thus X has the desired cdf F.

Example 3.2.

Use the ITM to generate a random variable X with pdf

$$f(x) = \frac{1}{2}(x+1),\ x \in [-1,1]. \tag{3.2}$$

Then use the value $U = 0.16$ to generate a value for X.

Solution. First, we find the cdf $F(x) = \int_{-\infty}^{x} f(t)dt$.

If $x < -1$, obviously $F(x) = 0$ (the integrand is 0).
If $x \in [-1,1]$, we have

$$\begin{aligned} F(x) &= \frac{1}{2}\int_{-1}^{x}(t+1)dt = \frac{1}{2}\left(\frac{1}{2}t^2 + t\right)\Big|_{-1}^{x} \\ &= \frac{1}{2}\left(\frac{1}{2}x^2 + x + \frac{1}{2}\right) = \frac{1}{4}(x+1)^2. \end{aligned}$$

If $x > 1$, then $F(x) = \frac{1}{2}\int_{-1}^{1}(t+1)dt = \int_{\mathbb{R}} f(t)dt = 1.$

So,

$$F(x) = \begin{cases} 0, & x < -1 \\ \frac{1}{4}(x+1)^2, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases}.$$

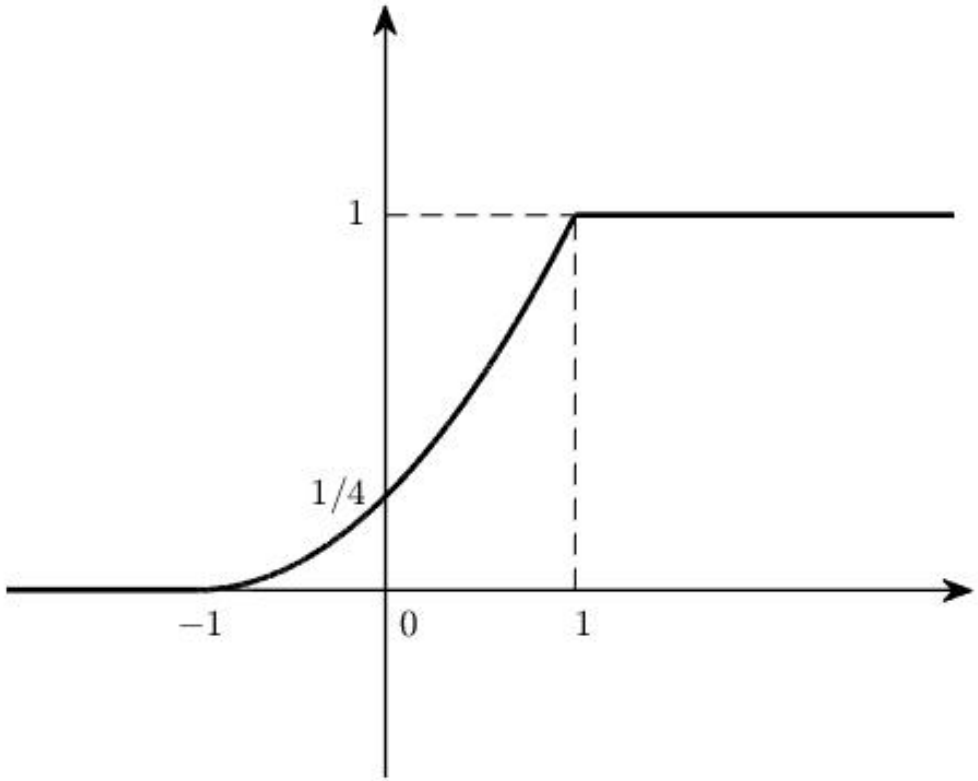

Figure 3: Function F in Example 3.2

The graph of the cdf F is shown in Figure 3. One can see that $F : [-1, 1] \to [0, 1]$ is one-to-one and surjective, so the inverse $F^{-1} : [0, 1] \to [-1, 1]$ exists.

We find it by solving $F(x) = y$ for x, i.e. $x = F^{-1}(y)$.

We have

$$\begin{aligned} \frac{1}{4}(x+1)^2 &= y, \\ (x+1)^2 &= 4y, \\ x+1 &= \sqrt{4y}, \\ x &= 2\sqrt{y}-1, \end{aligned}$$

so, $F^{-1}(y) = 2\sqrt{y} - 1$, for $y \in [0,1]$.
Then we generate X from U by

$$X = F^{-1}(U) = 2\sqrt{U} - 1.$$

For the value $U = 0.16$, we get $X = 2 \cdot 0.4 - 1 = -0.2$.

■

Example 3.3.

Use the ITM to generate $X \in Exp(\lambda), \lambda > 0$.

Solution. For $X \in Exp(\lambda)$, the pdf and cdf are given by

$$f(x) = \lambda e^{-\lambda x}, x \geq 0 \text{ and } F(x) = 1 - e^{-\lambda x}, x \geq 0,$$

respectively (see equation (5.14), Chapter 1, Lecture 2).
So, we find the inverse of the cdf

$$\begin{aligned} F(X) &= U, \\ 1 - e^{-\lambda X} &= U, \\ e^{-\lambda X} &= 1 - U, \\ -\lambda X &= \ln(1 - U), \\ X_1 &= -\frac{1}{\lambda} \ln(1 - U). \end{aligned} \tag{3.3}$$

Now, algebraically, we cannot simplify this expression, but we can notice the following:

$$U \in U(0,1) \iff 1 - U \in U(0,1).$$

Indeed, we see that for $x \in [0,1]$,

$$\begin{aligned} F_{1-U}(x) &= P(1-U \leq x) = P(U \geq 1-x) = 1 - P(U < 1-x) \\ &= 1 - F_U(1-x) \overset{(2.1)}{=} 1 - (1-x) = x = F_U(x). \end{aligned}$$

Then, by taking the derivative, $f_{1-U}(x) = f_U(x)$, so we can replace U by $1-U$ in (3.3) and get

$$X_2 = -\frac{1}{\lambda} \ln(U). \tag{3.4}$$

Notice that since $U, 1-U \in (0,1)$, we have that both $\ln(U), \ln(1-U) < 0$ and, thus, $X_1, X_2 > 0$, as they should be.

■

Discrete Inverse Transform Method

We can see that the previous algorithm seems to have one major fault, namely, that it is not applicable to discrete random variables, since in this case, the cdf F is *neither injective, nor surjective* and, thus, not invertible.

This problem can be overcome, by adjusting the algorithm the following way. In equation (3.1), we will take

$$X \quad = \quad F^{-1}(U) \quad = \quad \min\{x \mid F(x) \geq U\}. \tag{3.5}$$

This is known as the *generalized inverse* function.

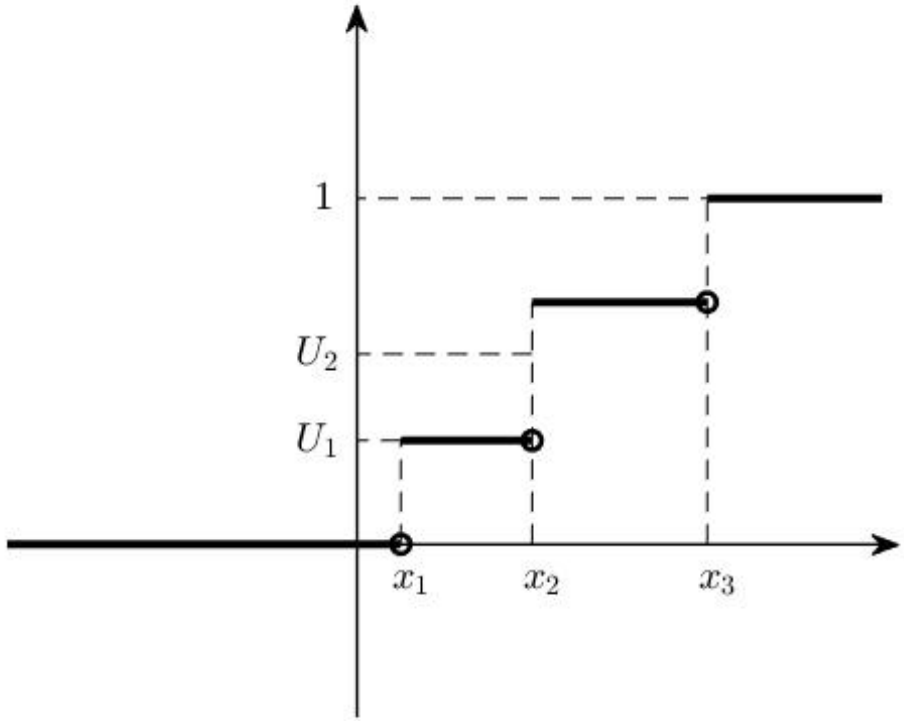

Figure 4: Generalized inverse

So, in Figure 4, we have

$$X_1 = F^{-1}(U_1) = \min\{x \mid F(x) \geq U_1\} = x_1, \text{ here } F(x_1) = U_1,$$
$$X_2 = F^{-1}(U_2) = \min\{x \mid F(x) \geq U_2\} = x_2, \text{ here } F(x_2) > U_2.$$

Example 3.4.
Let us revisit Geometric and Shifted Geometric variables and generate them using the DITM.

Solution. To keep computations simple, we generate a $SGeo(p)$ variable first and then adjust it accordingly to get a $Geo(p)$ variable.
For $X \in SGeo(p), p \in (0,1)$, recall the cdf (Chapter 1, Lecture 2):

$$F(x) = 1 - q^x = 1 - (1-p)^x, \; x \in \mathbb{N}.$$

We use (3.5) to find X:

$$\begin{aligned} 1 - (1-p)^x &\geq U, \\ (1-p)^x &\leq 1 - U, \\ x \ln(1-p) &\leq \ln(1-U), \\ x &\geq \frac{\ln(1-U)}{\ln(1-p)}, \text{ since } \ln(1-p) < 0. \end{aligned}$$

The smallest integer value satisfying this is the *ceiling function* value. Also, as before, $1-U$ can be replaced by U. Thus, a variable $X \in SGeo(p)$ is generated by

$$X = \left\lceil \frac{\ln(U)}{\ln(1-p)} \right\rceil. \tag{3.6}$$

For a $X \in Geo(p)$ variable, with cdf $F(x) = 1-(1-p)^{x+1}$, the same computations lead to

$$X = \left\lceil \frac{\ln(U)}{\ln(1-p)} - 1 \right\rceil. \tag{3.7}$$

■

Remark 3.5.

Notice how similar formula (3.6)

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

for simulating a $SGeo(p)$ variable is to formula (3.4)

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

which gives the simulation of an $Exp(\lambda)$ variable.

If $\lambda = -\ln(1-p)$, then the generated $SGeo(p)$ variable is just the ceiling of the $Exp(\lambda)$ one.

In other words, the ceiling of an Exponential variable has Shifted Geometric distribution. This just shows, again, the strong analogy between the two distributions.