SAPIENZA
UNIVERSITÀ DI ROMA

MASTER DEGREE IN CYBERSECURITY

Algorithms for Random Variates Generation

STATISTICS

Professor:
Tommaso Gastaldi

Student:
Federico Detomaso
1903906

Academic Year 2023/2024

Contents

1 Introduction

In simulation, pseudo random numbers serve as the foundation for generating samples from probability distribution models. We will now assume that the random number generator has been rigorously tested and that it produces sequences of $U_i \sim U(0,1)$ numbers. We now want to take the $U_i \sim U(0,1)$ and utilize them to generate from probability distributions.

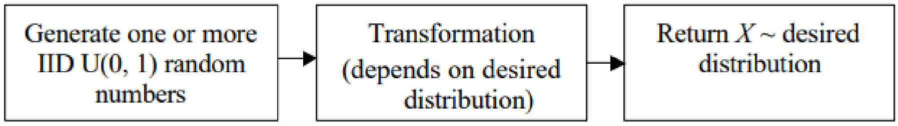


Figure 1: General form of Algorithms

The realized value from a probability distribution is called a **random variate**. Simulations use many different probability distributions as inputs. Thus, methods for generating random variates from distributions are required. Different distributions may require different algorithms due to the challenges of efficiently producing the random variables. Therefore, we need to know how to generate samples from probability distributions. In generating random variates the goal is to produce samples X_i from a distribution $F(x)$ given a source of random numbers, $U_i \sim U(0,1)$.

1.1 Requirements from a Algorithm

Exactness

- As far as possible use methods that results in random variates with exactly the desired distribution.
- Many approximate techniques are available, which should get second priority.
- One may argue that the fitted distributions are approximate anyways, so an approximate generation method should suffice. But still exact methods should be preferred.
- Because of huge computational resources, many exact and efficient algorithms exist.

Efficiency

- Efficiency of the algorithm in terms of storage space and execution time.
- Execution time has two components: set-up time and marginal execution time.
- Set-up time is the time required to do some initial computing to specify constants or tables that depend on the particular distribution and parameters.
- Marginal execution time is the incremental time required to generate each random variate.
- Since in a simulation experiment, we typically generate thousands of random variates, marginal execution time is far more than the set-up time.

Complexity

- Of the conceptual as well as implementational factors.
- One must ask whether the potential gain in efficiency that might be experienced by using a more complicated algorithm is worth the extra effort to understand and implement it.
- "Purpose" should be put in context: a more efficient but more complex algorithm might be appropriate for use in permanent software but not for a "one-time" simulation model.

Robustness

- When an algorithm is efficient for all parameter values.

2 Algorithms

2.1 Inverse Transform Method

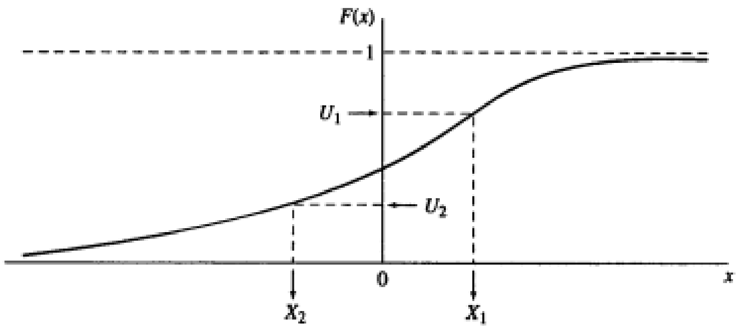
The inverse transform method is the preferred method for generating random variates provided that the inverse transform of the cumulative distribution function can be easily derived or computed numerically. The key advantage for the inverse transform method is that for every U_i use a corresponding X_i will be generated. That is, there is a one-to-one mapping between the pseudo-random number u_i and the generated variate x_i.

The inverse transform technique utilizes the inverse of the cumulative distribution function. First, generate a number, u_i, between 0 and 1 (along the U axis), then find the corresponding x_i coordinate by using $F^{-}1(.)$. For various values of u_i the x_i will be properly 'distributed' along the x-axis. The beauty of this method is that there is a one to one mapping between u_i and x_i. In other words, for each u_i there is a unique x_i because of the monotone property of the CDF.

2.1.1 Continuous Case

Suppose X is continuous with cumulative distribution function (CDF)

$F(x) = P(X \leq x)$ for all real numbers x x that is strictly increasing over all x.



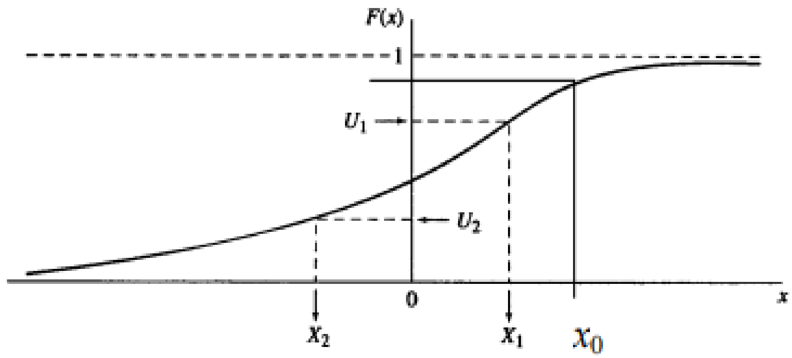
The inverse transformation algorithm is:

1. Generate $U \sim U(0,1)$ (random-number generator)
2. Return $X = F^{-1}(U)$.

Proof: (Assume F is strictly increasing for all x.)

For a fixed value x_0,

$$
\begin{aligned}
P(\text{returned } X \text{ is} \leq x_0) &= P(F^{-1}(U) \leq x_0) && \text{(def. of } X \text{ in algorithm)} \\
&= P(F(F^{-1}(U)) \leq F(x_0)) && (F \text{ is monotone } \uparrow) \\
&= P(U \leq F(x_0)) && \text{(def. of inverse function)} \\
&= P(0 \leq U \leq F(x_0)) && (U \geq 0 \text{ for sure}) \\
&= F(x_0) - 0 && (U \sim \mathrm{U}(0,1)) \\
&= F(x_0) && \text{(as desired)}
\end{aligned}
$$



Pick a fixed value x_0,
$X_1 \leq x_0$ if and only if $U_1 \leq F(x_0)$, so
$P(X_1 \leq x_0) = P(U_1 \leq F(x_0) = F(x_0)$, by definition of CDFs.

2.1.2 Discrete Case

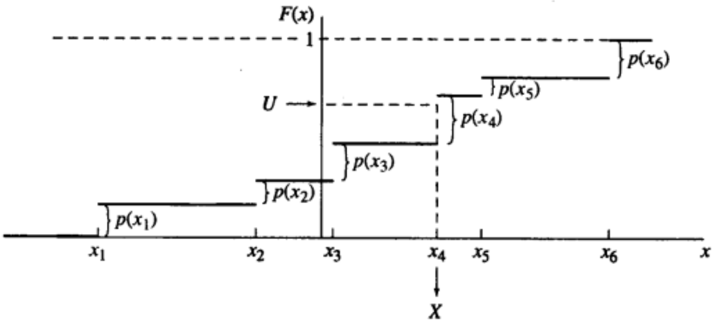
Suppose X is discrete with cumulative distribution function (CDF)

$$F(x) = P(X \leq x) \text{ for all real numbers } x.$$

and probability mass function

$$p(x_i) = P(X = x_i),$$

where $x_1, x_2, x_3, ...$ are the possible values X can take on



The inverse transformation algorithm is:

1. Generate $U \sim U(0,1)$ (random-number generator)

2. Find the smallest positive integer I such that $U \leq F(x_I)$

3. Return $X = x_I$.

Proof: From the above picture, $P(X = x_i) = p(x_i)$ in every case.

2.1.3 Examples

Discrete Distributions:

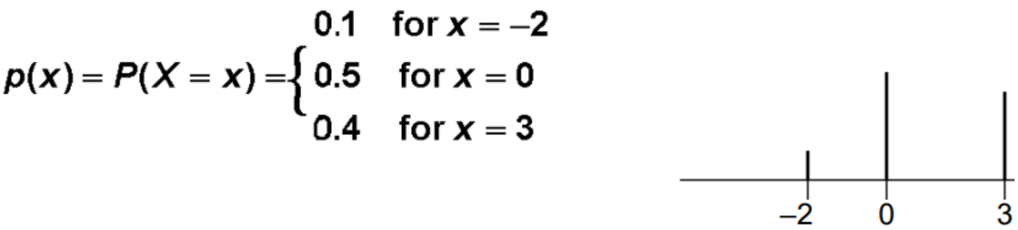


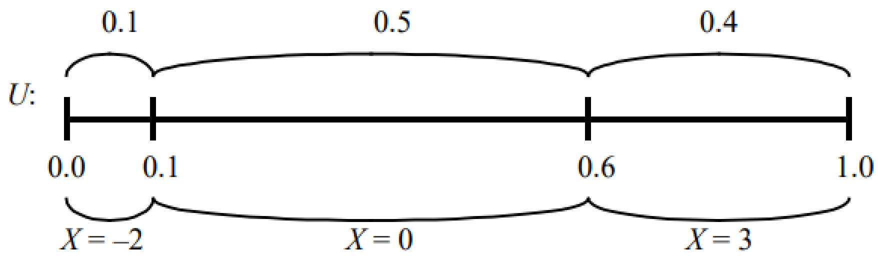
Figure 2: Probability Mass Function

Divide $[0,1]$ into subintervals of length $0.1, 0.5, 0.4$:

1. Generate $U \sim U(0,1)$;

2. See which subinterval it's in;

3. Return $X =$ corresponding value.



Continuous Distributions:

Weibull (α, β) distribution, parameters $\alpha > 0$, $\beta > 0$

Density Function is $f(x) = \begin{cases} \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^\alpha}, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$

CDF is $F(x) = \int_{-\infty}^{x} f(t)\,dt = \begin{cases} 1 - e^{-(x/\beta)^\alpha}, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$

Solve $U = F(X)$ for X:

$$\begin{aligned} U &= 1 - e^{-(X/\beta)^\alpha} \\ e^{-(X/\beta)^\alpha} &= 1 - U \\ -(X/\beta)^\alpha &= \ln(1-U) \\ X/\beta &= [-\ln(1-U)]^{1/\alpha} \\ X &= \beta[-\ln(1-U)]^{1/\alpha} \end{aligned}$$

Since $1 - U \sim U(0,1)$ as well, can replace $1 - U$ by U to get the final algorithm:

1. Generate $U \sim U(0,1)$
2. Return $X = \beta(-\ln(U))^{1/\alpha}$.

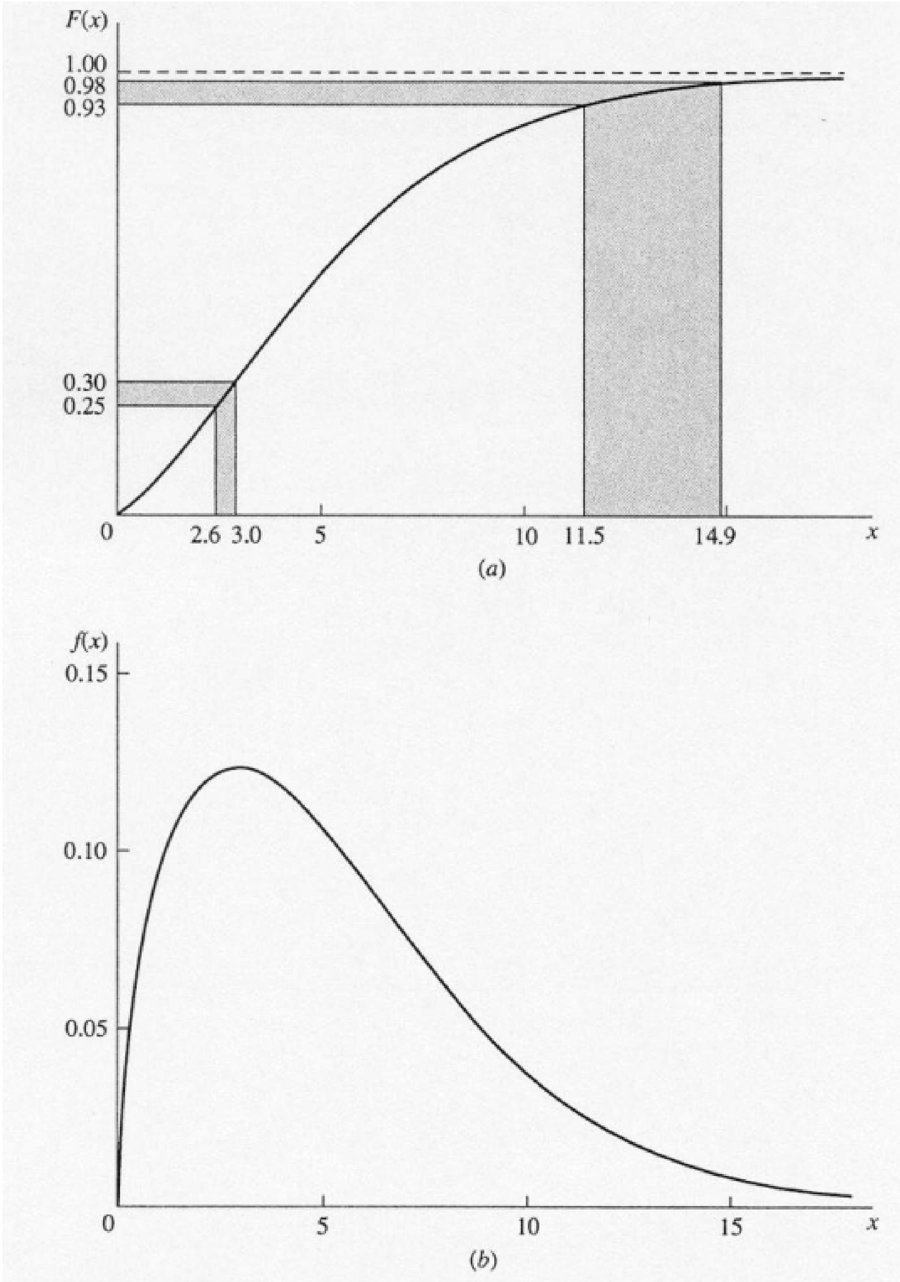


Figure 3: Example: Weibull ($\alpha = 1.5$, $\beta = 6$)

2.1.4 Problems with Inverse-Transform Approach

Performing the inversion of the Cumulative Distribution Function (CDF) may pose challenges, particularly due to the intricate nature of certain distributions. The process of numerically inverting the CDF may not always be the most expedient or straightforward approach, as it entails the utilization of numerical methods that can be computationally intensive. The feasibility and efficiency of this inversion process are contingent upon the specific characteristics of the underlying distribution, and alternative methodologies might be sought to achieve a more efficient and reliable solution.

2.1.5 Advantages of Inverse-Transform Approach

Facilitating variance-reduction techniques is pivotal in simulation studies of this method, exemplified by the investigation into the effects of expediting a bottleneck machine. This entails a comparative analysis between the existing machine and a faster alternative. To model the speedup, the service-time distribution for the accelerated machine is modified.

In the baseline run (Run 1) representing the current system, the inverse transform method is employed to generate service-time variates for the existing machine.

Subsequently, in Run 2, simulating the faster machine, the inverse transform method is again used for generating service-time variates. It is noteworthy that the very same uniform U(0,1) random numbers are utilized for both variates, leading to a positive correlation between service times.

The inverse transform method is particularly advantageous in intensifying this correlation compared to alternative variate-generation approaches.

This strengthened correlation is instrumental in reducing variability in the estimate of the effect of the faster machine. This reduction in variability is a primary reason why the inverse transform method is often considered the preferred choice. Another noteworthy technique is the use of common random numbers, which can significantly impact the quality of estimates or the computational effort required. Additionally, the generation of variates from truncated distributions plays a role in refining the precision of simulation results.

2.2 Composition Method

The composition method is an algorithm in which the desired probability distribution function (PDF) of a random variable can be expressed as a convex combination of several underlying distribution functions. A convex combination is a linear combination in which the coefficients are nonnegative and sum to one.

$F(x) = \sum_{j=1}^{\infty} p_j F(x_j)$,

where $p_j \geq 0$, $\sum_{j=1}^{\infty} p_j = 1$; and each F_j is a distribution function.

(Equivalently, can decompose density $f(x)$ or mass function $p(x)$ into convex combination of other density or mass functions)

In this context, the Composition Method exploits the mathematical concept of convexity to construct a new distribution function by combining existing ones in a convex way. By selecting appropriate weights for each distribution in the combination, the method allows for the creation of a composite distribution that reflects the desired characteristics of the target distribution.

The general composition algorithm is:

1. Generate a positive random integer J such that: $P(J=j)=p_j, j=1,2,...$

2. Return X with distribution function F_J.

Proof: For fixed x,

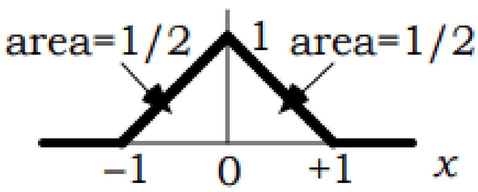
$$\begin{aligned}
\text{P(returned } X \le x) &= \sum_j P(X \le x \mid J=j) P(J=j) && \text{(condition on } J=j\text{)} \\
&= \sum_j P(X \le x \mid J=j) p_j && \text{(distribution of } J\text{)} \\
&= \sum_j F_j(x) p_j && \text{(given } J=j, X \sim F_j\text{)} \\
&= F(x) && \text{(decomposition of } F\text{)}
\end{aligned}$$

The trick is to find F_j's from which generation is easy and fast.

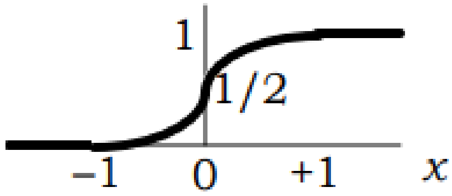
2.2.1 Examples

Symmetric triangular distribution on $[-1,1]$:

Density Function is $f(x)=\begin{cases} x+1, & \text{if } -1 \le x \le 0 \\ -x+1, & \text{if } 0 \le x \le +1 \\ 0, & \text{otherwise} \end{cases}$



$$\text{CDF is } F(x) = \begin{cases} 0, & \text{if } x < -1 \\ x^2/2 + x + 1/2, & \text{if } -1 \le x \le 0 \\ -x^2/2 + x + 1/2, & \text{if } 0 \le x \le +1 \\ 1, & \text{if } x > +1 \end{cases}$$



Inverse-transform:

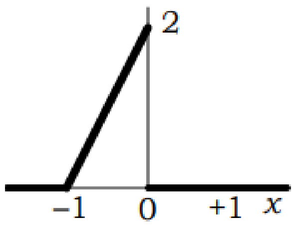
$$U = F(X) = \begin{Bmatrix} X^2/2 + X + 1/2 & \text{if } U < 1/2 \\ -X^2/2 + X + 1/2 & \text{if } U \ge 1/2 \end{Bmatrix}; \text{ solve for}$$

$$X = \begin{Bmatrix} \sqrt{2U} - 1 & \text{if } U < 1/2 \\ 1 - \sqrt{2(1-U)} & \text{if } U \ge 1/2 \end{Bmatrix}$$

Composition:

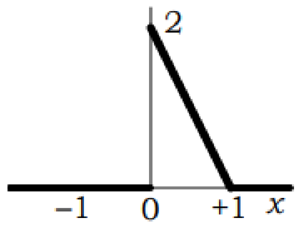
Define indicator function for the set A as $I_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$

$$\begin{aligned} f(x) &= (x+1)I_{[-1,0]}(x) + (-x+1)I_{[0,+1]}(x) \\ &= \underbrace{0.5}_{p_1}\underbrace{\{2(x+1)I_{[-1,0]}(x)\}}_{f_1(x)} + \underbrace{0.5}_{p_2}\underbrace{\{2(-x+1)I_{[0,+1]}(x)\}}_{f_2(x)} \end{aligned}$$



$F_1(x) = x^2 + 2x + 1$

$F_1^{-1}(U) = \sqrt{U} - 1$



$F_2(x) = -x^2 + 2x$

$F_2^{-1}(U) = 1 - \sqrt{1-U}$

Composition algorithm:

1. Generate $U_1, U_2 \sim U(0,1)$ independently
2. If $U_1 < 1/2$, return $X = \sqrt{U_2} - 1$; otherwise, return $X = 1 - \sqrt{1 - U_2}$

Method	U's	Compares	Adds	Multiplies	$\sqrt{\ }$'s
Inv. trnsfrm.	1	1	1.5	1	1
Composition	2	1	1.5	0	1

So composition needs one more U, one fewer multiply—faster if RNG (Random Number Generator) is fast.

2.2.2 Advantages of the Composition Method

- **Flexibility:** The Composition Method offers significant flexibility in modeling complex distributions by allowing the combination of multiple basic distributions to create a composite distribution.
- **Adaptability:** This approach works well when the desired distribution can be expressed as a convex combination of existing distributions, enabling more precise modeling.
- **Wide Applicability:** It can be used in a broad range of contexts, allowing the construction of tailored distributions for specific scenarios.

2.2.3 Disadvantages of the Composition Method

- **Computational Complexity:** Implementing the Composition Method can be computationally intensive, especially when dealing with the manipulation of complex distributions.
- **Difficulty in Weight Selection:** The proper choice of weights for the basic distributions can be critical and not always straightforward, requiring in-depth knowledge of the problem at hand.
- **Limitations in Representation:** Some distributions may be challenging to accurately represent through a combination of basic distributions, limiting the effectiveness of the Composition Method in certain contexts.

2.3 Convolution Method

Many random variables are related to each other through some functional relationship. One of the most common relationships is the convolution relationship. The distribution of the sum of two or more random variables is called the convolution. Let $Y_i \sim G(y)$ be independent and identically distributed random variables. Let $X = \sum_{i=1}^{n} Y_i$. Then the distribution of X is said to be the n-fold convolution of Y.
Some common random variables that are related through the convolution operation are:

- A binomial random variable is the sum of Bernoulli random variables.
- A negative binomial random variable is the sum of geometric random variables.
- An Erlang random variable is the sum of exponential random variables.
- A Normal random variable is the sum of other normal random variables.
- A chi-squared random variable is the sum of squared normal random variables.

The basic convolution algorithm simply generates $Y_i \sim G(y)$ and then sums the generated random variables.

1. Generate $Y_1, Y_2, ..., Y_n$ independently from their distribution
2. Return $X = Y_1 + Y_2 + ... + Y_n$.

Let's look at a couple of examples.
By definition, a negative binomial distribution represents one of the following two random variables:

- The number of failures in sequence of Bernoulli trials before the r^{th} success, has range $\{0, 1, 2, ...\}$
- The number of trials in a sequence of Bernoulli trials until the r^{th} success, it has range $\{r, r+1, r+2, ...\}$

The number of failures before the r^{th} success, has range $\{0, 1, 2, ...\}$. This is the sum of geometric random variables with range $\{0, 1, 2, ...\}$ with the same success probability.

If $Y \sim NB(r, p)$ with range $\{0, 1, 2, ...\}$, then

$$Y = \frac{1}{n} \sum_{i=1}^{r} X_i$$

when $X_i \sim Geometric(p)$ with range $\{0, 1, 2, ...\}$, and X_i can be generated via inverse transform with:

$$X_i = \lfloor \frac{\ln(1-U_i)}{\ln(1-p)} \rfloor$$

If we have a negative binomial distribution that represents the number of trials until the r^{th} success, it has range $\{r, r+1, r+2, ...\}$, in this text we call this a shifted negative binomial distribution.

A random variable from a “shifted” negative binomial distribution is the sum of shifted geometric random variables with range $\{0, 1, 2, 3, ...\}$. with same success probability. In this text, we refer to this geometric distribution as the shifted geometric distribution.

If $T \sim NB(r, p)$ with range $\{r, r+1, r+2, ...\}$, then

$$T = \frac{1}{n} \sum_{i=1}^{r} X_i$$

when $X_i \sim ShiftedGeometric(p)$ with range $\{0, 1, 2, 3, ...\}$, and X_i can be generated via inverse transform with:

$$X_i = 1 + \lfloor \frac{\ln(1-U_i)}{\ln(1-p)} \rfloor$$

Notice that the relationship between these random variables as follows:

Let Y be the number of failures in a sequence of Bernoulli trials before the r^{th} success. Let T be the number of trials in a sequence of Bernoulli trials until the r^{th} success,

Then, clearly, $T = Y + r$. Notice that we can generate Y, via convolution, as previously explained and just add r to get T.

$$Y = \frac{1}{n} \sum_{i=1}^{r} X_i$$

Where $X_i \sim Geometric(p)$ with range $\{0, 1, 2, ...\}$, and X_i can be generated via inverse transform with:

$$X_i = \lfloor \frac{\ln(1-U_i)}{\ln(1-p)} \rfloor$$

2.3.1 Contrast with Composition method

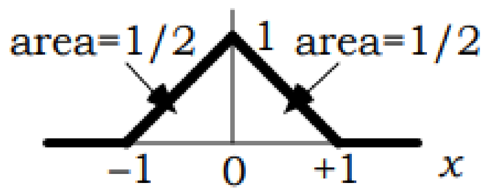
Composition involves representing the distribution function (or density or mass function) by expressing it as a (weighted) sum of other distribution functions (or densities or masses). This method allows for the creation of a composite distribution by combining the characteristics of multiple underlying distributions in a weighted manner.

In contrast, Convolution is a technique that expresses the random variable itself as the sum of other random variables. It focuses on the summation of independent random variables to derive the probability distribution of their sum.

2.3.2 Examples

Symmetric triangular distribution on $[-1, 1]$:

Density Function is $f(x) = \begin{cases} x+1, & \text{if } -1 \le x \le 0 \\ -x+1, & \text{if } 0 \le x \le +1 \\ 0, & \text{otherwise} \end{cases}$



By simple conditional probability:

If $U_1, U_2 \sim$ IID (Independent and Identically Distributed) $U(0, 1)$,
then $U_1 + U_2 \sim$ symmetric triangular on $[0, 2]$, so just shift left by 1:

$$\begin{aligned} X &= U_1 + U_2 - 1 \\ &= \underbrace{(U_1 - 0.5)}_{Y_1} + \underbrace{(U_2 - 0.5)}_{Y_2} \end{aligned}$$

Clearly, in terms of complexity the convolution method beats inverse trasform and composition.

2.3.3 Advantages of Convolution method

- **Mathematical Rigor:** The Convolution Method is grounded in mathematical rigor, providing a precise and well-defined approach to finding the distribution of the sum of independent random variables.

- **Versatility:** This technique is versatile and applicable to a wide range of scenarios involving the summation of random variables, making it a fundamental tool in probability theory and statistics.

- **Analytical Solutions:** In some cases, the Convolution Method allows for analytical solutions, providing explicit formulas for the probability distribution of the sum.

2.3.4 Disadvantages of Convolution method

- **Complexity for Non-Identically Distributed Variables:** When dealing with non-identically distributed random variables, the convolution of their distributions can become computationally complex and may not have a straightforward analytical solution.

- **Computational Burden:** In practice, the computation of convolutions, especially for complex distributions or a large number of variables, can be computationally burdensome.

- **Limited Applicability to Non-Independent Variables:** The Convolution Method assumes independence among the random variables, limiting its applicability in situations where variables are not truly independent.

2.4 Acceptance-Rejection Method

In the acceptance-rejection method, the probability density function (PDF) $f(x)$, from which it is desired to obtain a sample is replaced by a proxy PDF, $w(x)$, that can be sampled from more easily. The following illustrates how $w(x)$ is defined such that the selected samples from $w(x)$ can be used directly to represent random variates from $f(x)$. The PDF $w(x)$ is based on the development of a majorizing function for $f(x)$. A majorizing function, $g(x)$, for $f(x)$, is a function such that $g(x) \geq f(x)$ for $-\infty < x < +\infty$.

Usually, this method is used when inverse transform is not directly applicable or is inefficient (e.g. gamma, beta). Also it is applicable to continuous as well as discrete case.

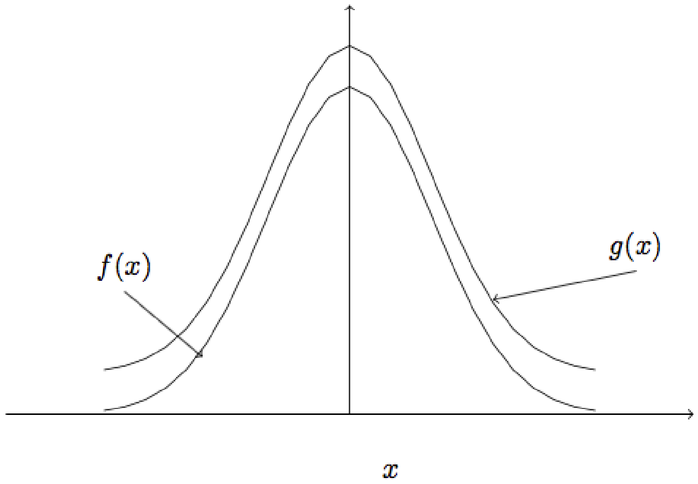


Figure 4: Concept of a Majorizing Function

Then $g(x) \geq 0$ for all x, but $\int_{-\infty}^{\infty} g(x)\, dx \geq \int_{-\infty}^{\infty} f(x)\, dx = 1$ (in general g(x) will not be a density function).

Set $c = \int_{-\infty}^{\infty} g(x)\, dx$

If $w(x)$ is defined as $w(x) = g(x)/c$ then $w(x)$ will be a probability density function.

The algorithm is:

1. Generate $W \sim w(x)$
2. Generate $U \sim U(0,1)$ (independent of W in Step 1)
3. If $U \leq f(W)/g(W)$, return $X = W$ and stop; else go back to Step 1 and try again.

The validity of the procedure is based on deriving the cumulative distribution function of W given that the $W = w$ was accepted, $P\{W \leq x \mid W = w$ is accepted $\}$.

The algorithm is inefficient if rejection occurs frequently.

Proof:

The key of proof is to get an X only conditional on acceptance in Step 3. So,

$$
\begin{aligned}
P(\text{generated } X \le x) &= P(Y \le x \mid \text{acceptance}) \\
&= \frac{P(\text{acceptance}, Y \le x)}{P(\text{acceptance})} \quad \text{(def. of cond'l. prob.)} \quad *
\end{aligned}
$$

For any y,

$$P(\text{acceptance} \mid Y = y) = P(U \le f(y)/t(y)) = f(y)/t(y)$$

since $U \sim \text{U}(0,1)$, Y is independent of U, and $t(y) > f(y)$. Thus,

$$
\begin{aligned}
P(\text{acceptance}, Y \le x) &= \int_{-\infty}^{\infty} P(\text{acceptance}, Y \le x \mid Y = y) r(y)\,dy \\
&= \underbrace{\int_{-\infty}^{x} P(\text{acceptance}, Y \le x \mid Y = y) r(y)\,dy}_{Y \le x \text{ on this range, guaranteeing } Y \le x \text{ in the probability}} + \\
&\quad \underbrace{\int_{x}^{\infty} P(\text{acceptance}, Y \le x \mid Y = y) r(y)\,dy}_{Y \ge x \text{ on this range, contradicting } Y \le x \text{ in the probability}} \\
&= \int_{-\infty}^{x} P(\text{acceptance}, Y \le x \mid Y = y) r(y)\,dy \\
&= \frac{1}{c} \int_{-\infty}^{x} \frac{f(y)}{t(y)} t(y)\,dy \quad \text{(def. of } r(y)\text{)} \\
&= F(x)/c \qquad **
\end{aligned}
$$

Next,

$$
\begin{aligned}
P(\text{acceptance}) &= \int_{-\infty}^{\infty} P(\text{acceptance} \mid Y = y) r(y)\,dy \\
&= \frac{1}{c} \int_{-\infty}^{\infty} \frac{f(y)}{t(y)} t(y)\,dy \\
&= \frac{1}{c} \int_{-\infty}^{\infty} f(y)\,dy \\
&= 1/c \qquad ***
\end{aligned}
$$

Since f is a density function and so integrates to 1. Putting *** and ** back into *,

$$
\begin{aligned}
P(\text{generated } X \le x) &= \frac{P(\text{acceptance}, Y \le x)}{P(\text{acceptance})} \\
&= \frac{F(x)/c}{1/c} \\
&= F(x),
\end{aligned}
$$

as desired.

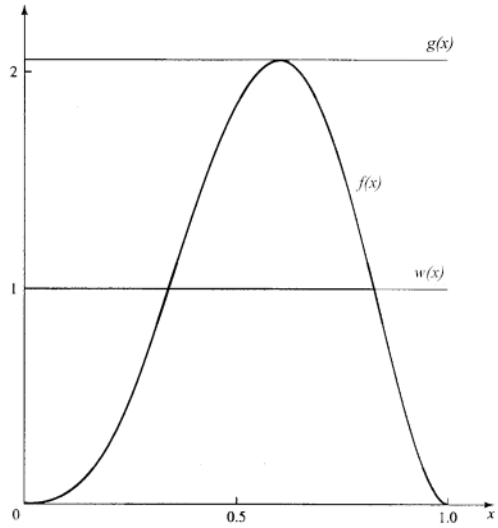
2.4.1 Examples

For a Beta(4,3) Distribution:

Density Function is $f(x) = \begin{cases} 60x^3(1-x)^2, & \text{if } 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$

Top of density is $f(0.6) = 2.0736$, so let $g(x) = 2.0736$ for $0 \le x \le 1$.

Thus, $c = 2.0736$, and W is the $U(0,1)$ density function.



Algorithm:

1. Generate $W \sim U(0,1)$
2. Generate $U \sim U(0,1)$ (independent of W)
3. If $U \le 60W^3(1-W)^2/2.0736$, return $X = W$ and stop; else go back to Step 1 and try again.

P(acceptance) in step 3 is $1/2.0736 = 0.48$.

2.4.2 Advantages of Acceptance-Rejection method

- **Generality:** The Acceptance-Rejection Method is a general technique applicable to a wide range of probability distributions, making it versatile for simulating random variables from various distributions.
- **Ease of Implementation:** It is relatively straightforward to implement, especially for unimodal and symmetric distributions, requiring only the knowledge of an envelope function.

- **No Derivative Information Required:** Unlike some other methods, the Acceptance-Rejection Method does not necessitate knowledge of the probability density function's derivative, simplifying its application in cases where such information is challenging to obtain.

2.4.3 Disadvantages of Acceptance-Rejection method

- **Efficiency Concerns:** The method may not be the most efficient for all distributions, especially when the envelope function is significantly larger than the target distribution in certain regions.

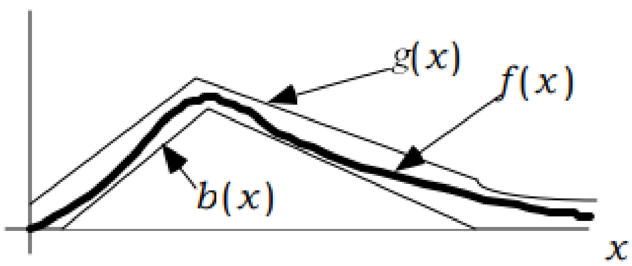
- **Envelope Function Selection:** Choosing an appropriate envelope function can be challenging, and in some cases, finding such a function may not be straightforward, affecting the efficiency of the method.

- **Limited Applicability for Multidimensional Distributions:** Extending the method to multidimensional distributions can be complex, and its efficiency may decrease in higher dimensions.

2.4.4 Alternative Improved Form

A potential bottleneck in the Acceptance-Rejection (A-R) method arises during the evaluation of the function $f(W)$ in the third step, especially if f is computationally complex. To address this potential inefficiency, a swift pre-test for acceptance can be incorporated just before entering the third step. If the pre-test is successful, it indicates that the test in the third step would also be successful. Consequently, the algorithm can terminate without executing the full test and evaluating $f(W)$.

One effective strategy for implementing this pre-test involves introducing a minorizing function $b(x)$ beneath $f(x)$. This minorizing function provides a lower bound that helps quickly assess whether the random variable W satisfies the acceptance criteria, streamlining the Acceptance-Rejection process and mitigating the computational burden associated with evaluating $f(W)$.



Given that $b(x) \leq f(x)$, the pre-test involves initially checking if $U \leq \frac{b(W)}{g(W)}$. If this condition is satisfied, the variable W is accepted immediately without the need for further testing (if not, the actual test in step 3 must be performed). A well-suited choice for $b(x)$ is one that is close to $f(x)$, ensuring that the pre-test and the step 3 test are in agreement most of the time. Additionally, an ideal $b(x)$ is characterized by being fast and easy to evaluate, enhancing the efficiency of both the pre-test and the subsequent steps in the process.

3 Conclusions

The field of algorithms for random variates generation plays a pivotal role in diverse applications, spanning statistics, computer science, finance, and more. Through a comprehensive exploration of various methods, we have witnessed the versatility and adaptability of these techniques in generating random variables that adhere to specific probability distributions.

Each method presents its own set of advantages and disadvantages. The Inverse Transform Method stands out for its simplicity, while the Composition and Convolution Methods offer powerful tools for combining and manipulating distributions. The Acceptance-Rejection Method provides a versatile approach, losing a bit in computational efficiency.

As technology advances and computational resources grow, researchers continue to refine existing methods and develop new algorithms for generating random variates. These advancements not only contribute to the theoretical foundations of probability and statistics but also find practical applications in fields where realistic simulations and modeling are crucial.

References

[1] Generating Random Variates. https://www.cyut.edu.tw/~hchorng/downdata/1st/SS8_Random%20Variate.pdf.

[2] Generation of Random Variables. https://www-eio.upc.es/~lmontero/lmm_tm/MESIO-SIM%20-%20%28US%29%20Generation%20of%20random%20variables.pdf.

[3] Generating Random Variates from Distributions. https://rossetti.github.io/RossettiArenaBook/app-rnrv-rvs.html#app:rnrv:rvs.