

MEM6804 Modeling and Simulation for Logistics & Supply Chain

物流与供应链建模与仿真

Theory Analysis

Lecture 4: Random Variate Generation

SHEN Haihui 沈海辉

Sino-US Global Logistics Institute
Shanghai Jiao Tong University

🏠 shenhaihui.github.io/teaching/mem6804f
✉ shenhaihui@sjtu.edu.cn

Spring 2021 (full-time)



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

董浩云航运与物流研究院
CY TUNG Institute of Maritime and Logistics
中美物流研究院 (工程系统管理研究院)
Sino-US Global Logistics Institute (Institute of Industrial & System Engineering)



- 1 Introduction
- 2 Random Number Generation
 - ▶ Definition
 - ▶ Pseudo-Random Numbers
 - ▶ Linear Congruential Generator
 - ▶ More Sophisticated RNGs
 - ▶ Tests for Random Numbers
- 3 Random Variate Generation
 - ▶ Inverse-Transform Technique
 - ▶ Acceptance-Rejection Technique
 - ▶ Other Ad-Hoc Methods
 - ▶ Generating Poisson Process



1 Introduction

2 Random Number Generation

- ▶ Definition
- ▶ Pseudo-Random Numbers
- ▶ Linear Congruential Generator
- ▶ More Sophisticated RNGs
- ▶ Tests for Random Numbers

3 Random Variate Generation

- ▶ Inverse-Transform Technique
- ▶ Acceptance-Rejection Technique
- ▶ Other Ad-Hoc Methods
- ▶ Generating Poisson Process



- Random **variable** is a variable whose values are random and depend on a probability distribution.
 - E.g., normal, exponential, Poisson, etc.
- Random **variate** is a *particular* outcome (i.e. observed sample, realization) of a random variable.
 - E.g., 5 random variates (outcomes) from a $\mathcal{N}(0, 1)$ random variable: 0.5377, 1.8339, -2.2588 , 0.8622, 0.3188.
- When simulating a system, we often need to generate random variates (e.g., interarrival time, service time) from all kinds of distributions (e.g., exponential distribution, arbitrary empirical distribution).

- In practice:
 - Most simulation softwares have build-in functions to generate random variates from common distributions.
 - Most programming languages have implemented the common routines of random variate generation in the libraries.
- It is nevertheless worthwhile to understand how random variate generation occurs.
 - In case when build-in functions or libraries are unavailable.
 - To better understand the randomness in stochastic simulation.
 - Be alert to some inadequate random variate generator.
- To produce a sequence of random variates from a given distribution (of a random variable):
 - ① Start with random variates from $\text{Unif}(0, 1)$ (called **random numbers**).
 - ② All random variates with given distribution are “transformed” from **random numbers**.

1 Introduction

2 Random Number Generation

- ▶ Definition
- ▶ Pseudo-Random Numbers
- ▶ Linear Congruential Generator
- ▶ More Sophisticated RNGs
- ▶ Tests for Random Numbers

3 Random Variate Generation

- ▶ Inverse-Transform Technique
- ▶ Acceptance-Rejection Technique
- ▶ Other Ad-Hoc Methods
- ▶ Generating Poisson Process



- **Random numbers** are a sequence of **independent** random observations from **uniform** distribution on $[0, 1]$.
 - If $U \sim \text{Unif}(0, 1)$, then $\mathbb{E}[U] = \frac{1}{2}$, $\text{Var}(U) = \frac{1}{12}$, and its pdf is

$$f(u) = \begin{cases} 1, & 0 \leq u \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

- 10 random numbers generated in MATLAB: 0.8147, 0.9058, 0.1270, 0.9134, 0.6324, 0.0975, 0.2785, 0.5469, 0.9575, 0.9649.
- Statistical Properties
 - Uniformity: Each value on $[0, 1]$ has equal likelihood.
 - Independence: Implies no correlation between successive numbers.

- Uniformity

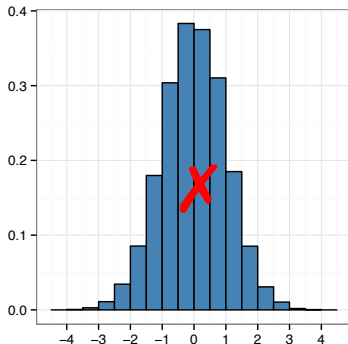
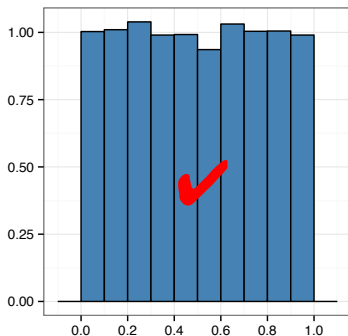


Figure: Empirical pdf (i.e., Scaled Histogram): Uniformity vs Nonuniformity (from [ZHANG Xiaowei](#))

- Independence

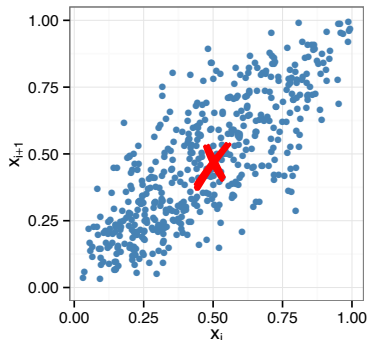
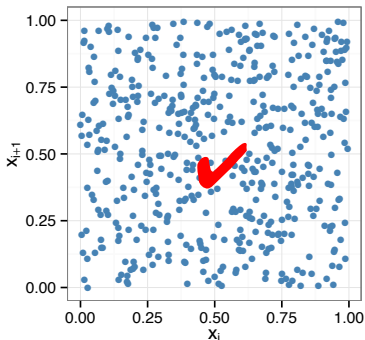


Figure: Scatter Plot: Uncorrelated vs Correlated (from [ZHANG Xiaowei](#))

- A computer can NOT generate true randomness! It can only give us **pseudo-random** (伪随机) numbers.
- “Pseudo” means *false*
 - Generating random numbers by a known method removes true randomness.
 - The set of pseudo-random numbers can be repeated.
- Goal: To produce a sequence of numbers in $[0, 1]$ that imitates the ideal properties of random numbers.
 - Statistical properties are the most important.
 - True randomness is not the first priority.

- Properties of a good random number generator (RNG):
 - ① Pass statistical tests.
 - ② Solid theoretical support.
 - ③ Fast.
 - ④ Sufficiently long cycle (period).
 - ⑤ Portable to different computers.
 - ⑥ Replicable.
- Techniques for RNG:
 - Linear Congruential Generator (LCG)
 - Combined LCG
 - Multiple Recursive Generator (MRG)

- Linear Congruential Generator (LCG, 线性同余发生器) is a simple and early development of RNG.

- ① Produce a sequence of integers x_1, x_2, \dots between 0 and $m - 1$ by

$$x_{i+1} = (ax_i + c) \bmod m, \quad i = 0, 1, 2, \dots$$

- The initial value x_0 is called the *seed* (种子), a is *multiplier* (乘子), c is *increment* (增量), and m is *modulus* (模数).

- ② Transform x_i 's to values between 0 and 1 by

$$u_i = \frac{x_i}{m}, \quad i = 0, 1, 2, \dots$$

- Possible values of u_i : $\{0, \frac{1}{m}, \dots, \frac{m-1}{m}\}$. (May not cover all!)
- The selection of the values for a , c , m , and x_0 drastically affects the statistical properties and the cycle length.

- Example: Use LCG with $x_0 = 27$, $a = 17$, $c = 43$, and $m = 100$.

$$x_0 = 27$$

$$x_1 = (17 \times 27 + 43) \bmod 100 = 502 \bmod 100 = 2$$

$$u_1 = 2/100 = 0.02$$

$$x_2 = (17 \times 2 + 43) \bmod 100 = 77 \bmod 100 = 77$$

$$u_2 = 77/100 = 0.77$$

$$x_3 = (17 \times 77 + 43) \bmod 100 = 1352 \bmod 100 = 52$$

$$u_3 = 52/100 = 0.52$$

$$x_4 = (17 \times 52 + 43) \bmod 100 = 927 \bmod 100 = 27$$

$$u_4 = 27/100 = 0.27$$

The cycle length is only 4!

- Try <https://xiaoweiz.shinyapps.io/randNumGen> for different parameters.



- An actual use of LCG ([Lewis et al. 1969](#)): $a = 7^5$, $c = 0$, $m = 2^{31} - 1 = 2,147,483,647$ (a prime number).
 - It adopts $u_i = \frac{x_i}{m+1}$.
 - It passes many of the standard statistical tests.
 - Cycle length $\approx 2^{31} - 2 \approx 2 \times 10^9$ (well over 2 billion).
- Note: By letting modulus m be a power of 2 (or close), the modulo operation can be conducted efficiently, since most digital computers use a binary representation of numbers.
- As computing power has increased, LCG is not adequate nowadays; more sophisticated RNGs are used in practice.

- Combined LCG: Combine $J (\geq 2)$ LCG (with $c = 0$).
- For 32-bit computers, **L'Ecuyer (1988)** suggests combining $J = 2$ generators with $a_1 = 40,014$, $m_1 = 2,147,483,563$, $a_2 = 40,692$, and $m_2 = 2,147,483,399$.
 - ① Select seed $x_{1,0}$ in the range $[1, m_1 - 1]$ for the first generator, and seed $x_{2,0}$ in the range $[1, m_2 - 1]$ for the second. Set $j = 0$.
 - ② Calculate

$$x_{1,j+1} = a_1 x_{1,j} \bmod m_1,$$

$$x_{2,j+1} = a_2 x_{2,j} \bmod m_2.$$
 - ③ Let $x_{j+1} = (x_{1,j+1} - x_{2,j+1}) \bmod (m_1 - 1)$.
(*Remark:* mod uses floored division, i.e., $y \bmod m = y - m \lfloor \frac{y}{m} \rfloor$.)
 - ④ Return

$$u_{j+1} = \begin{cases} \frac{x_{j+1}}{m_1}, & \text{if } x_{j+1} > 0, \\ \frac{m_1 - 1}{m_1}, & \text{if } x_{j+1} = 0. \end{cases}$$
 - ⑤ Set $j = j + 1$ and go to Step 2.

It has cycle length $(m_1 - 1)(m_2 - 1)/2 \approx 2 \times 10^{18}$.



- Multiple Recursive Generator (MRG): Extends LCG by using a higher-order recursion:

$$x_i = (a_1x_{i-1} + a_2x_{i-2} + \cdots + a_kx_{i-K}) \bmod m.$$

- A specific instance that has been widely implemented is MRG32k3a[†] (L'Ecuyer 1999), which is a *combined MRG* with $J = 2$ and $K = 3$.
 - It has cycle length $\approx 3 \times 10^{57}$, which is enormous.
 - If you could generate one billion (10^9) pseudo-random numbers per second, then it would take longer than the age of the universe to exhaust the period of MRG32k3a!

[†]MRG32k3a or its adaptation is one of the RNGs used in MATLAB, R, SAS, Arena, etc.



- Tests based on generated sequences of numbers.
 - *Frequency Test* for uniformity (discussed in next lecture)
 - Kolmogorov–Smirnov test (柯尔莫哥洛夫–斯米尔诺夫检验)
 - chi-square test (χ^2 test, 卡方检验)
 - *Autocorrelation Test* for independence.
- There are also some *theoretical tests* without actually generating any numbers, e.g., spectral test (谱检验).
- Fortunately, the well-known RNGs which are widely used in simulation softwares and languages have been extensively tested and validated.
- Be careful when the RNG at hand is not explicitly known or documented!
 - Even RNGs that have been used for years in popular commercial softwares (e.g., Excel, Visual Basic), have been found to be inadequate ([L'Ecuyer 2001](#)).

- 1 Introduction
- 2 Random Number Generation
 - ▶ Definition
 - ▶ Pseudo-Random Numbers
 - ▶ Linear Congruential Generator
 - ▶ More Sophisticated RNGs
 - ▶ Tests for Random Numbers
- 3 Random Variate Generation
 - ▶ Inverse-Transform Technique
 - ▶ Acceptance-Rejection Technique
 - ▶ Other Ad-Hoc Methods
 - ▶ Generating Poisson Process



- Assumption: RNG is available, i.e. we have a sequence of random numbers (i.e., $\text{Unif}(0, 1)$ random variates).
- Goal: Produce random variates from a given probability distribution (e.g. exponential, Poisson, etc.).
- Widely-used techniques[†]
 - Inverse-transform technique (generic)
 - Acceptance-rejection technique (generic)
 - Other ad-hoc methods for some specific distributions

[†] Methods introduced in this lecture are exact; there are also approximation methods such as MCMC.

- Let $F(x)$ be the CDF of X , i.e., $F(x) = \mathbb{P}(X \leq x)$.

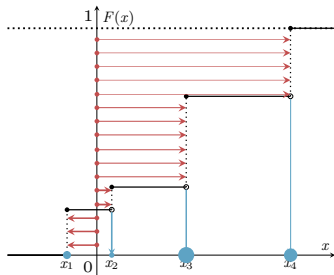
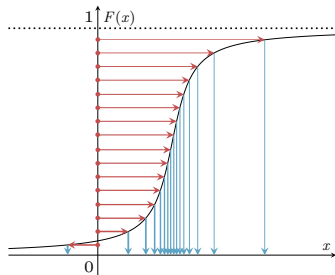


Figure: Continuous Random Variable

Figure: Discrete Random Variable

- Procedures
 - 1 Generate (as needed) random numbers (on vertical axis).
 - 2 Map inversely to points on horizontal axis, which are the desired random variates from $F(x)$.

- The formal definition of inverse function is

$$F^{-1}(y) := \min\{x : F(x) \geq y\}, \quad 0 < y < 1.$$

- If $U \sim \text{Unif}(0, 1)$, then $F^{-1}(U)$ has the same distribution as X , i.e.,

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x).$$

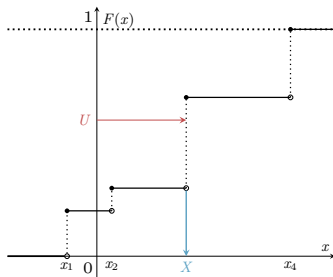
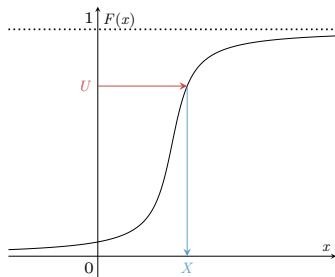


Figure: Continuous Random Variable

Figure: Discrete Random Variable

- The inverse-transform technique is useful when the CDF is so simple that its inverse function can be *analytically solved* or *easily computed*.
- It can be used to sample from various continuous distributions
 - uniform
 - exponential
 - triangular
 - Weibull
 - Cauchy
 - Pareto
- It can be used to sample from all (in principle) discrete distributions, e.g.,
 - discrete uniform
 - geometric
 - arbitrary empirical distribution

- Goal: Generate random variates from $X \sim \text{Unif}(a, b)$.
- Intuition: Since $X = a + (b - a)U$, we just need to:
 - ① Generate random number u_i ;
 - ② Output $x_i = a + (b - a)u_i$ as the required random variates.
- For $X \sim \text{Unif}(a, b)$, the pdf and CDF are

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b, \\ 0, & \text{otherwise,} \end{cases} \quad F(x) = \begin{cases} 0, & x < a, \\ \frac{x-a}{b-a}, & a \leq x \leq b, \\ 1, & b < x. \end{cases}$$

- Solve the inverse function of $F(x)$,

$$F^{-1}(y) = a + (b - a)y, \quad 0 < y < 1.$$

- So, $F^{-1}(U) = a + (b - a)U$ has the same distribution as X .



- Goal: Generate random variates from $X \sim \text{Exp}(\lambda)$.
- For $X \sim \text{Exp}(\lambda)$, the pdf and CDF are

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0, \end{cases} \quad F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

- Solve the inverse function of $F(x)$,

$$F^{-1}(y) = -\frac{1}{\lambda} \ln(1 - y), \quad 0 < y < 1.$$

- So, $F^{-1}(U) = -\frac{1}{\lambda} \ln(1 - U)$ has the same distribution as X .
- *Remark:* $1 - U \sim \text{Unif}(0, 1) \implies -\frac{1}{\lambda} \ln(U)$ is sufficient.
- Numerical test for $\text{Exp}(1)$ in **Excel**.
 - ① Generate 200 random numbers.
 - ② Obtain 200 random variates via the inverse function.

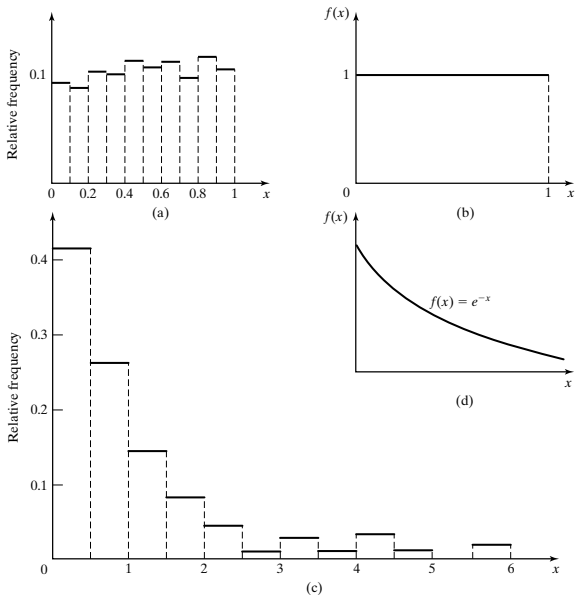


Figure:

(a) Empirical histogram of 200 generated uniform random numbers;
 (b) Theoretical density of $\text{Unif}(0, 1)$;

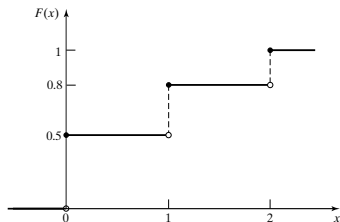
(c) Empirical histogram of 200 generated variates from $\text{Exp}(1)$;
 (d) Theoretical density of $\text{Exp}(1)$.

(from [Banks et al. \(2010\)](#))

- Consider a discrete random variable X taking values 0, 1, 2 with probability 0.5, 0.3 and 0.2.
- The pmf and CDF are

$$p(x) = \begin{cases} 0.5, & x = 0, \\ 0.3, & x = 1, \\ 0.2, & x = 2, \end{cases} \quad F(x) = \begin{cases} 0, & x < 0, \\ 0.5, & 0 \leq x < 1, \\ 0.8, & 1 \leq x < 2, \\ 1, & 2 \leq x. \end{cases}$$

- Solve the inverse function. (Recall $F^{-1}(y) := \min\{x : F(x) \geq y\}$.)



$$F^{-1}(y) = \begin{cases} 0, & 0 < y \leq 0.5, \\ 1, & 0.5 < y \leq 0.8, \\ 2, & 0.8 < y < 1. \end{cases}$$

Try it in Excel.



- Why do we need another method when the inverse-transform technique is already generic?
 - The CDF of a desired distribution may not have an analytical form.
 - The inverse CDF may not exist in closed form and may be challenging to evaluate, e.g., beta, gamma, normal, etc.
 - Although you can solve the inverse transform via numerical methods anyway, the efficiency may be low.
- Acceptance-rejection technique is also useful for generating a *non-stationary Poisson process* (more details later).

- Goal: Generate random variates from $X \sim \text{Unif}(1/4, 1)$ using acceptance-rejection technique.
 - ① Generate a random number u (from $U \sim \text{Unif}(0, 1)$).
 - ② If $u \geq 1/4$, **accept** u , output u as the desired random variate; if $u < 1/4$, **reject** u , and return to Step 1.
 - ③ If another $\text{Unif}(1/4, 1)$ random variate is needed, repeat the procedure from Step 1; stop otherwise.
- Important Observation 1: To produce one random variate using A-R technique, one may need to generate multiple random numbers.
 - Whereas there exists a one-to-one mapping for the inverse-transform method.

- Important Observation 2: The accepted values of U are **conditioned** values.
 - U itself does not have the desired distribution.
 - U conditioned on the event $\{U \geq 1/4\}$ does!
- For $1/4 \leq x \leq 1$,

$$\mathbb{P}\{U \leq x | U \geq 1/4\} = \frac{\mathbb{P}\{U \leq x \text{ and } U \geq 1/4\}}{\mathbb{P}\{U \geq 1/4\}} = \frac{x - 1/4}{3/4},$$

which is exactly the desired CDF of $X \sim \text{Unif}(1/4, 1)$.

- Suppose we want to generate random variates from X , whose density $f(x)$ has support $[a, b]$ and is upper bounded by M .

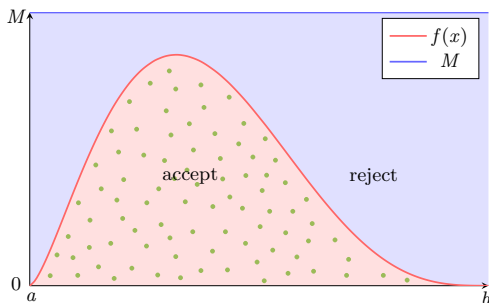


Figure: Bounded Support (original image from [ZHANG Xiaowei](#))

- Generate random variate pairs $(y_1, z_1), (y_2, z_2), \dots$, from $\text{Uniform}\{(y, z) : a \leq y \leq b, 0 \leq z \leq M\}$.
 - y_i from $Y \sim \text{Unif}(a, b)$, z_i from $Z \sim \text{Unif}(0, M)$
- Accept the pair if $z_i < f(y_i)$ and output y_i as random variate from X with density $f(x)$.

- Y conditioned on the event $\{Z < f(Y)\}$ has the same distribution as X , i.e., having density $f(x)$.
 - $(Y, Z) \sim \text{Uniform}\{(y, z) : a \leq y \leq b, 0 \leq z \leq M\}$.

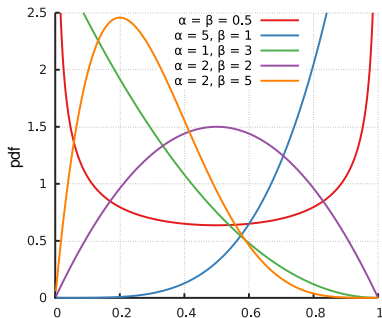
Proof.

$$\begin{aligned}
 \mathbb{P}\{Y \leq x | Z < f(Y)\} &= \frac{\mathbb{P}\{Y \leq x, Z < f(Y)\}}{\mathbb{P}\{Z < f(Y)\}} \\
 &= \frac{\int_a^x \int_0^{f(y)} f_{Y,Z}(y, z) dz dy}{\int_a^b \int_0^{f(y)} f_{Y,Z}(y, z) dz dy} \quad \text{Note: } f_{Y,Z}(y, z) = \frac{1}{(b-a)M} \\
 &= \frac{\int_a^x \int_0^{f(y)} \frac{1}{(b-a)M} dz dy}{\int_a^b \int_0^{f(y)} \frac{1}{(b-a)M} dz dy} = \frac{\int_a^x \int_0^{f(y)} dz dy}{\int_a^b \int_0^{f(y)} dz dy} \\
 &= \frac{\int_a^x f(y) dy}{\int_a^b f(y) dy} = \frac{\mathbb{P}\{X \leq x\}}{1} = \mathbb{P}\{X \leq x\}. \quad \blacksquare
 \end{aligned}$$

- The acceptance rate is $\mathbb{P}\{Z < f(Y)\} = \frac{1}{(b-a)M}$.



- Goal: Generate random variates from $\text{Beta}(\alpha, \beta)$, where the density is $f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$, $x \in [0, 1]$.



- If $\alpha > 1$ and $\beta > 1$, then $f(x)$ is maximized at $x = \frac{\alpha-1}{\alpha+\beta-2}$ and the maximum is $M = \frac{(\alpha-1)^{\alpha-1}(\beta-1)^{\beta-1}}{(\alpha+\beta-2)^{\alpha+\beta-2}B(\alpha, \beta)}$.
- The acceptance rate is $\frac{1}{(b-a)M} = \frac{1}{(1-0)M} = \frac{1}{M}$.

- Generate random variates from X , whose density $f(x)$ is upper bounded by $Mg(x)$, where $g(x)$ is *instrumental* density.

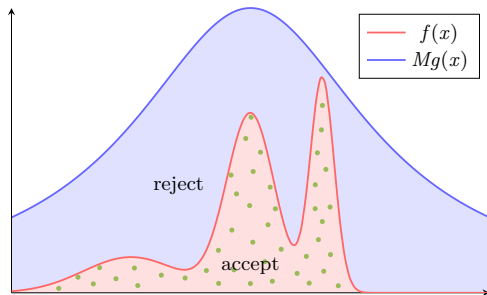


Figure: Unbounded Support (original image from [ZHANG Xiaowei](#))

- Generate random variate pairs $(y_1, z_1), (y_2, z_2), \dots$, from $\text{Uniform}\{(y, z) : y \in \text{support of } g(\cdot), 0 \leq z \leq Mg(y)\}$.
 - y_i from $Y \sim g(\cdot)$, z_i from $Z \sim \text{Unif}(0, Mg(y_i))$ (**why?**)
- Accept the pair if $z_i < f(y_i)$ and output y_i as random variate from X with density $f(x)$.

- Y conditioned on the event $\{Z < f(Y)\}$ has the same distribution as X , i.e., having density $f(x)$.
 - Let Θ denote $\{(y, z) : y \in \text{support of } g(\cdot), 0 \leq z \leq Mg(y)\}$.
 - $(Y, Z) \sim \text{Uniform } \Theta$.

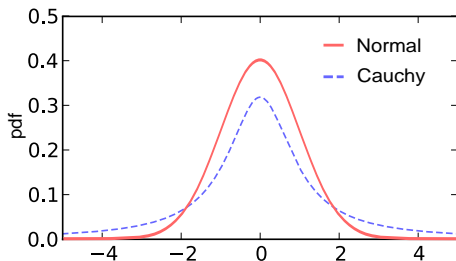
Proof.

$$\begin{aligned}
 \mathbb{P}\{Y \leq x | Z < f(Y)\} &= \frac{\mathbb{P}\{Y \leq x, Z < f(Y)\}}{\mathbb{P}\{Z < f(Y)\}} \\
 &= \frac{\int_{-\infty}^x \int_0^{f(y)} f_{Y,Z}(y, z) dz dy}{\int_{-\infty}^{\infty} \int_0^{f(y)} f_{Y,Z}(y, z) dz dy} \quad \text{Note: } f_{Y,Z}(y, z) = \frac{1}{\Theta \text{ area}} \\
 &= \frac{\int_{-\infty}^x \int_0^{f(y)} \frac{1}{\Theta \text{ area}} dz dy}{\int_{-\infty}^{\infty} \int_0^{f(y)} \frac{1}{\Theta \text{ area}} dz dy} = \frac{\int_{-\infty}^x \int_0^{f(y)} dz dy}{\int_{-\infty}^{\infty} \int_0^{f(y)} dz dy} \\
 &= \frac{\int_{-\infty}^x f(y) dy}{\int_{-\infty}^{\infty} f(y) dy} = \frac{\mathbb{P}\{X \leq x\}}{1} = \mathbb{P}\{X \leq x\}. \quad \blacksquare
 \end{aligned}$$

- The acceptance rate is

$$\mathbb{P}\{Z < f(Y)\} = \frac{1}{\Theta \text{ area}} = \frac{1}{\int_{-\infty}^{\infty} Mg(y) dy} = \frac{1}{M \int_{-\infty}^{\infty} g(y) dy} = \frac{1}{M} \cdot \text{Shanghai Jiao Tong University}$$

- Goal: Generate random variates from $\mathcal{N}(0, 1)$, where the density is $f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$, $x \in (-\infty, \infty)$.
- Use Cauchy(0, 1) density as instrumental density, which is $g(x) = \frac{1}{\pi(1+x^2)}$, $x \in (-\infty, \infty)$.



- It is easy to see that $\frac{f(x)}{g(x)} = \sqrt{\frac{\pi}{2}}(1+x^2)e^{-\frac{x^2}{2}}$ is maximized at $x = \pm 1$ and the maximum is $\sqrt{\frac{2\pi}{e}}$, which is the required M .
- The acceptance rate is $\frac{1}{M} = \sqrt{\frac{e}{2\pi}} \approx 0.6577$.

- Box–Muller method for $\mathcal{N}(0, 1)$ random variates:
 - ① Generate u_1 and u_2 independently from $\text{Unif}(0, 1)$.
 - ② Let $z_1 = \sqrt{-2 \ln u_1} \cos(2\pi u_2)$ and $z_2 = \sqrt{-2 \ln u_1} \sin(2\pi u_2)$.
- z_1 and z_2 are random variates from $\mathcal{N}(0, 1)$ (independent).
- Intuition:
 - For two independent $\mathcal{N}(0, 1)$ RVs Z_1 and Z_2 ,

$$Z_1^2, Z_2^2 \sim \chi_1^2, \quad Z_1^2 + Z_2^2 \sim \chi_2^2.$$
 - $X \sim \text{Exp}(1/2) \iff X \sim \chi_2^2.$
 - $-2 \ln u_1$ is a random variate from $\text{Exp}(1/2)$ (and thus χ_2^2).
 - The angle is distributed uniformly around the circle.
- Rigorous proof.

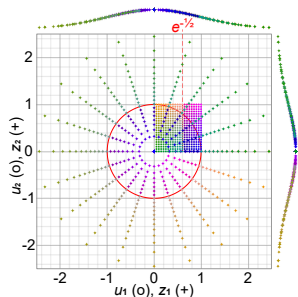


Figure: Box–Muller Method Visualisation
(image by Cmglee / CC BY 3.0)

[Interactive Graph](#)

- **Poisson process** with rate λ : Interarrival time distribution is exponential with rate λ (or mean $1/\lambda$), and

$$N(t+h) - N(t) \sim \text{Poisson}(\lambda h). \quad (\text{same as } N(h))$$

- To generate Poisson process with rate λ , one only need to generate iid $\text{Exp}(\lambda)$ random variates.
 - s_i , the arrival time of the i th arrival, satisfies

$$s_i = s_{i-1} - (1/\lambda) \ln(u_i), \quad i = 1, 2, \dots$$

- **Nonhomogeneous Poisson process** with rate (intensity) function $\lambda(t)$:

$$N(t+h) - N(t) \sim \text{Poisson}(m(t+h) - m(t)),$$

where $m(t) = \int_0^t \lambda(s) ds$.



- To generate nonhomogeneous Poisson process with rate function $\lambda(t)$, one can use the acceptance-rejection method (which is also called *thinning* in this context).
- Idea behind thinning:
 - Generate a *stationary* Poisson arrival process at the fastest rate $\lambda^* = \max_t \lambda(t)$.
 - But “accept” only a portion of arrivals, thinning out just enough to get the desired time-varying rate.
- Algorithm:
 - ① Set $t = 0$ and $i = 1$.
 - ② Generate x from $\text{Exp}(\lambda^*)$, and let $t \leftarrow t + x$ (this is the arrival time of the *stationary* Poisson process with rate λ^*).
 - ③ Generate random number u (from $\text{Unif}(0, 1)$).
If $u \leq \lambda(t)/\lambda^*$, then $s_i = t$ and $i \leftarrow i + 1$.
 - ④ Go to Step 2.

