Random Number Generation

Kaiza Amouh

January 9, 2021

Outline

1. Pseudo-Random Numbers on [0, 1]
   - Properties of pseudo-random numbers
   - Generation routines
2. Discrete Random Variables simulation
3. Simulation of Continuous Random Variables
   - Inverse distribution method
   - Rejection sampling
   - Simulation of Normal Distribution
   - Generalized Rejection Sampling
   - Bivariate Gaussian Distribution

Outline

1. Pseudo-Random Numbers on [0, 1]
   - Properties of pseudo-random numbers
   - Generation routines
2. Discrete Random Variables simulation
3. Simulation of Continuous Random Variables

Properties

A sequence of Random Numbers R_1, R_2, ..., must have two important statistical properties :

Uniformity

- Consider we generate N numbers within the interval $[0, 1]$
- Then devide that interval into n subintervals of equal length
- There should be an expected $\frac{N}{n}$ numbers landing in each subinterval.

Independence

The probability of observing a value in a particular interval is independent of the previous value drawn.

Density and Moments

A Random number R must be independently drawn from a uniform distribution with the following properties:

$$f_R(x) = \begin{cases} 1 & \textit{for } x \in [0,1] \\ 0 & \textit{otherwise} \end{cases}$$

$$\mathbb{E}(R) = \int_0^1 x\, dx = \frac{1}{2}\left[x^2\right]_0^1 = \frac{1}{2}$$

$$\mathbb{V}(R) = \int_0^1 x^2\, dx - (\mathbb{E}(R))^2 = \frac{1}{3}\left[x^3\right]_0^1 - \left(\frac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$

Pseudo-Random numbers

- Pseudo : because generating numbers using a known method removes the potential for true randomness
- If the method is known, the set of random numbers can be replicated !
- The goal is to produce a sequence of numbers in $[0, 1]$ that imitates the ideal properties of random numbers : uniform distribution and independence.

Problems with Pseudo-Random numbers

- Generated numbers might not be uniformly distributed
- Generated numbers might be discrete-valued instead of continuous-valued
- Mean and Variance of the generated numbers might be too low or too high
- There might be dependence (i.e. correlation)

Tests to assess the quality of generated numbers

- Frequency test : uses Kolmogorov-Smirnov or Chi-square test to compare the generated distribution to a Uniform one
- Autocorrelation test : tests the correlation between numbers and compares the sample correlation to the expected correlation, zero

Generation Routines Considerations

- The routine should be fast : individual computations are inexpensive, but a simulation may require many millions of random numbers
- Portable to different computers : ideally to different programming languages. This ensures the program produces same results
- Have sufficiently long cycle : the cycle length, or period, represents the length of random number sequence before previous numbers begin to repeat in an earlier order
- Replicable : given the starting point, it should be possible to generate the same set of random numbers, regardless of the system used for simulation
- Closely approximate the ideal statistical properties of uniformity and independence

Techniques for generating Random Numbers

Pseudo-Random Number Generators

- Linear Congruential Method (LCM)
  Most widely used technique for generating random numbers
- Combined Linear Congruential Generators (CLCG)
  Extension to yield longer period (or cycle)

Quasi-Random Number Generators

- Van der Corput and Halton sequence (Radical inverse functions)
- The Kakutani sequence (Ergodic transform with p-adic additions)
- The Faure sequence (Combination of radial inverse functions)
- Sobol and Niederreiter sequence (Generalization of Faure sequence)

Linear Congruential Method

The idea is to produce a sequence of integers, $X_1, X_2, \ldots$ between 0 and $(m-1)$ by following a recursive relationship:

$$X_{i+1} = (aX_i + c) \ mod \ m, \quad i = 0, 1, 2, \ldots$$

$$\nearrow \qquad \uparrow \qquad \nwarrow$$

$$(Multiplier) \quad (Increment) \quad (Modulus)$$

- X_0 is called the *seed*
- The values choosen for a, c, m, and X_0 drastically affects the statistical properties and cycle length
- If $c \neq 0$ then it is called *mixed congruential* method
- When $c = 0$ it is called *multiplicative congruential* method

Linear Congruential Method

The generated integers lay in the range $[0, (m-1)]$, and can be converted to floating Random numbers within $[0, 1]$

$$R_i = \frac{X_i}{m}, \quad i = 1, 2, ...$$

Example for $X_0 = 27$, $a = 17$, $c = 43$, and $m = 100$

$$X_1 = (17 * 27 + 43) \ mod \ 100 = 502 \quad mod \ 100 = 2, \quad R_1 = 0.02$$
$$X_2 = (17 * 2 + 43) \quad mod \ 100 = 77 \quad mod \ 100 = 77, \quad R_2 = 0.77$$
$$X_3 = (17 * 77 + 43) \ mod \ 100 = 1352 \ mod \ 100 = 52, \quad R_3 = 0.52$$

Linear Congruential Method

- Notice that R_i is discrete instead of continuous, as generated numbers only take values from the set $I = \left\{0, \frac{1}{m}, \frac{2}{m}, ..., \frac{m-1}{m}\right\}$
  This issue can be solved by choosing a very large integer for the modulus m
- One may also face a density problem when the generated numbers are clustered around few regions on $[0, 1]$
  A proper choice of parameters a, c, m, and X_0 should solve that issue
- Most digital computers use a binary representation of numbers. Thus, speed and efficiency are better when the modulus m is a power of 2

Combined Linear Congruential Generators

With increased computing power, the complexity of simulated systems is increasing, requiring longer period generators.
Approach : Combine two or more *multiplicative congruential generators in such a way to produce a generator with good statistical properties.*

Method suggested by L'Ecuyer

- *If $W_{i,1}$, $W_{i,2}$, ..., $W_{i,k}$ are any independent, discrete valued random variables (not necessarily identically distributed)*
- *If one of them, say $W_{i,1}$ is uniformly distributed on the integers from 0 to $(m_1 - 2)$, then:*

$$W_i = \left(\sum_{j=1}^{k} W_{i,j} \right) \ mod \ (m_1 - 1)$$

is uniformly distributed on the integers from 0 to $(m_1 - 2)$

Combined Linear Congruential Generators

Let $X_{i,1}$, $X_{i,2}$, ..., $X_{i,k}$ be the i^{th} output from k different multiplicative congruential generators.

- The j^{th} generator has prime modulus m_j and multiplier a_j, and period is $(m_j - 1)$
- Produced integers $X_{i,j}$ are approx $\sim \mathcal{U}([1, (m_j - 1)])$
- $W_{i,j} = X_{i,j} - 1$ is approx $\sim \mathcal{U}([0, (m_j - 2)])$

Suggested form

$$X_i = \left(\sum_{j=1}^{k} (-1)^{j-1} W_{i,j} \right) \mod (m_1 - 1) \quad \text{Hence, } R_i = \begin{cases} \frac{X_i}{m_1}, & X_i > 0 \\ \frac{m_1 - 1}{m_1}, & X_i = 0 \end{cases}$$

The maximum possible period for such a generator is

$$p = \frac{(m_1 - 1)(m_2 - 1)...(m_k - 1)}{2^{k-1}}$$

Example of Combined Linear Congruential Generators

- For a 32-bit computers, L'Ecuyer (1988) suggests combining $k = 2$ generators with $m_1 = 2'147'483'563$, $a_1 = 40'014$, $m_2 = 2'147'483'399$, and $a_2 = 40'692$.
- The combined generator's period is $\frac{(m_1-1)(m_2-1)}{2} \sim 2 * 10^{18}$
  The algorithm becomes :

Algorithm

- Step 1 : Select seeds
  $X_{1,0}$ in the range $[1, 2'147'483'562]$ for the 1^{st} generator
  $X_{2,0}$ in the range $[1, 2'147'483'398]$ for the 2^{nd} generator

Example of Combined Linear Congruential Generators

Algorithm

- Step 2 : For each individual generator,
  $X_{1,j+1} = 40014 * X_{1,j} \mod 2'147'483'563$
  $X_{2,j+1} = 40692 * X_{1,j} \mod 2'147'483'399$
- Step 3 : $X_{j+1} = (X_{1,j+1} - X_{2,j+1}) \mod 2'147'483'562$
- Step 4: Return

$$R_{j+1} = \begin{cases} \frac{X_{j+1}}{2'147'483'563}, & X_{j+1} > 0 \\ \frac{2'147'483'562}{2'147'483'563}, & X_{j+1} = 0 \end{cases}$$

- Step 5 : Set $j = j + 1$, go back to step 2.

Outline

1 Pseudo-Random Numbers on [0, 1]

2 Discrete Random Variables simulation

3 Simulation of Continuous Random Variables

Discrete Random Variables

- Discrete Random Variables take their values from a countable set
- They can be simulated from Uniformly distributed random numbers $U \sim \mathcal{U}_{[0,1]}$

Usual Discrete Random Variables

- Head or Tail
- Bernoulli's law
- Binomial distribution
- Discrete distribution on a finite Set
- Poisson distribution

Head or Tail

Let $X \sim \mathcal{B}\left(\frac{1}{2}\right)$ denote a *Head-or-Tail* Random Variable

- We generate $U \sim \mathcal{U}_{[0,1]}$ using one of the algorithms detailed in previous sections
- Then :

$$\begin{cases} U \leq \frac{1}{2} & \Longrightarrow \quad X = \text{"}Head\text{"} \\ U > \frac{1}{2} & \Longrightarrow \quad X = \text{"}Tail\text{"} \end{cases}$$

Formal expression of X

$$X = \mathbf{1}_{\{U \leq \frac{1}{2}\}}$$

Bernoulli distribution

- A Random Variable $X \sim \mathcal{B}(p)$ is said to follow a *Bernoulli* distribution with parameter p, when X is the result of an experiment with only two possible outcomes: Success and Failure
- $\mathbb{P}(X = "SUCCESS") = p$

Simulation

First, generate $U \sim \mathcal{U}_{[0,1]}$. Then :

$$X = \mathbf{1}_{\{U \leq p\}} \quad i.e. \quad \begin{cases} U \leq p \implies X = 1 \ (SUCCESS) \\ U > p \implies X = 0 \ (FAILURE) \end{cases}$$

Binomial distribution

- A Binomial Random Variable $X \sim \mathcal{B}(n, p)$ represents the sum of n independent variables $Y_i \sim \mathcal{B}(p)$

$$\{Y_1, ..., Y_n\} \text{ i.i.d., } Y_i \sim \mathcal{B}(p) \implies X = \sum_{i=1}^{n} Y_i \sim \mathcal{B}(n, p)$$

Simulation

We simulate n *Bernoulli* variables and sum them up

$$X = \sum_{i=1}^{n} \mathbf{1}_{\{U_i < p\}}$$

Discrete distribution on a finite Set

Let X be a random variable with values in $\{1, ..., K\}$. Let :

$$p_k = \mathbb{P}(X = k), \quad P_k = \sum_{j \leq k} p_j, \quad P_0 = 0, \quad P_1 = p_1 \text{ and } P_K = 1$$

Algorithm

We simulate $U \sim \mathcal{U}_{\{[0,1]\}}$ and set $X = k$ if $P_{k-1} \leq U < P_k$

$$X = \sum_{k=1}^{K} k \mathbf{1}_{\{P_{k-1} \leq U < P_k\}}$$

This method is equivalent to dividing the $[0, 1]$ interval into K sub-intervals of respective length p_k

Poisson distribution - 1^{st} algorithm

A Random Variable $X \sim \mathcal{P}(\lambda)$ is said to follow a *Poisson* distribution of parameter λ when:

$$p_k = \mathbb{P}(X = k) = e^{-\lambda}\frac{\lambda^k}{k!} \quad \textit{for } k \in \mathbb{N}$$

Let : $P_k = \sum_{j=0}^{k} p_j$. We can easily see that :

$$p_{k+1} = \frac{\lambda}{k+1}p_k \quad \textit{and} \quad P_{k+1} = P_k + \frac{\lambda}{k+1}p_k$$

Simulation algorithm

$$X = \sum_{k\geq 0} k\mathbf{1}_{\{P_{k-1}\leq U<P_k\}} \quad \textit{with the convention } P_{-1} = 0$$

Poisson distribution - 2^{nd} algorithm

- Another simulation method for *Poisson* distribution arises from its relationship with *Exponential* distribution
- If a random event occurs every Y_i time period, with $Y_i \sim \mathcal{E}(\lambda)$, Then the number of events that occur in a unit time period $X \sim \mathcal{P}(\lambda)$

Simulation algorithm

Let $\{Y_1, Y_2, ...\}$ $i.i.d.$, $Y_i \sim \mathcal{E}(\lambda)$, and define $Z_k := \sum_{i=1}^{k} Y_i$. X is simulated as:

$$X = \sum_{k \geq 0} k \mathbf{1}_{\{Z_k \leq 1 < Z_{k+1}\}}$$

- In other words, we simulate several $\mathcal{E}(\lambda)$ and sum them up as we go until the sum reaches 1. The necessary number of simulations to reach a sum of 1 is our *Poisson* Random Variable

Outline

1. Pseudo-Random Numbers on [0, 1]
2. Discrete Random Variables simulation
3. Simulation of Continuous Random Variables
   - Inverse distribution method
   - Rejection sampling
   - Simulation of Normal Distribution
   - Generalized Rejection Sampling
   - Bivariate Gaussian Distribution

Continuous Random Variables - Inverse distribution

Let X be a continuous Random Variable with a strictly increasing probability distribution function $F(x) = \mathbb{P}(X \leq x)$

Theorem

$$F(X) \sim \mathcal{U}_{[0,1]}$$

Proof:

Let $u = F(x) \Leftrightarrow x = F^{-1}(u)$

$$F(x) = \mathbb{P}(X \leq x) \implies F\left(F^{-1}(u)\right) = \mathbb{P}\left(X \leq F^{-1}(u)\right)$$
$$\iff u = \mathbb{P}(F(X) \leq u)$$

And we recognize the distribution function of a *Uniform* Random Variable

- Hence if we know the expression of F^{-1}, we just have to simulate $U \sim \mathcal{U}_{[0,1]}$ and generate $X = F^{-1}(U)$

Inverse distribution - Example of *Exponential distribution*

$$X \sim \mathcal{E}(\lambda) \Longleftrightarrow F(x) = 1 - e^{-\lambda x} \Longleftrightarrow F^{-1} = -\frac{1}{\lambda} ln(1-u)$$

Simulation algorithm

- First generate $U \sim \mathcal{U}_{[0,1]}$
- Then compute $X = -ln(1-U)/\lambda$
- Note that if $U \sim \mathcal{U}_{[0,1]}$ then $(1-U) \sim \mathcal{U}_{[0,1]}$ also. We can therefore deduce X as

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

Rejection Sampling method

The goal is to simulate a continuous Random Variable X with density function f and distribution function F.

Assumptions

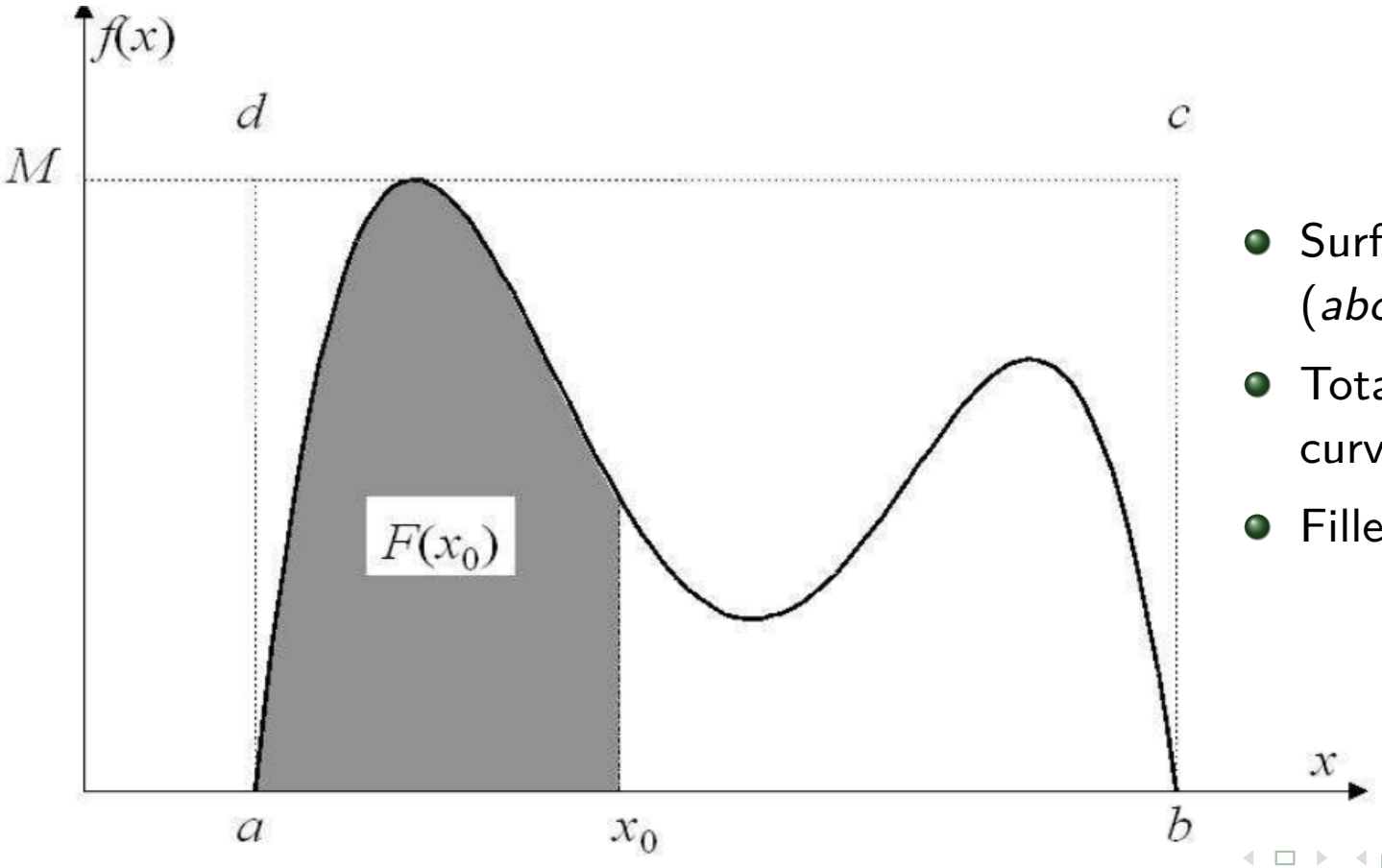
- The density f has a compact support, *i.e.* $f(x) = 0$ if $x \notin [a, b]$ for some a and b. Example:

$$f(x) = 6x(1-x)\mathbf{1}_{\{0 \leq x \leq 1\}}$$

- There exists an upper bound M of f:

$$\forall x \in [a, b] : f(x) \leq M$$

Rejection Sampling method



- Surface of the Rectangle $(abcd) = M(b - a)$
- Total surface under the curve $= 1$ (a density !)
- Filled surface $= F(x_0)$

Rejection Sampling method

- We first drop a point $A = (X, Y)$ uniformly distributed on the rectangle $(abcd)$
  - Generate $U_1 \sim \mathcal{U}_{[0,1]} \longrightarrow X = a + (b - a)U_1$
  - Generate $U_2 \sim \mathcal{U}_{[0,1]} \longrightarrow Y = MU_2$
- If $Y \leq f(X)$, we keep X. Otherwise, we drop another point
- We keep generating until the condition $Y \leq f(X)$ is met

Theorem

The Random Variable X obtained through the procedure described above does have a density function f and a distribution function F

Rejection Sampling method - Proof of the Theorem

$$\mathbb{P}(X \leq x_0) = \mathbb{P}(X \leq x_0 | \text{ we keep } A(X,Y)) = \mathbb{P}(X \leq x_0 | Y \leq f(X))$$

$$= \frac{\mathbb{P}(X \leq x_0, Y \leq f(X))}{\mathbb{P}(Y \leq f(X))}$$

We have

$$\begin{cases} \mathbb{P}(X \leq x_0, Y \leq f(X)) = \frac{\text{surface of the filled zone}}{\text{surface of the rectangle } (abcd)} = \frac{F(x_0)}{M(b-a)} \\ \mathbb{P}(Y \leq f(X)) = \frac{\text{surface under the curve}}{\text{surface of the rectangle } (abcd)} = \frac{1}{M(b-a)} \end{cases}$$

Therefore

$$\mathbb{P}(X \leq x_0) = \frac{F(x_0)}{M(b-a)} * \frac{M(b-a)}{1} = F(x_0)$$

Normal Distribution

- A Random Variable $X \sim \mathcal{N}(0,1)$ has the following distribution function:
$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$$
- The simulation methods described previously cannot be applied for Normal distribution
  - There is no simple formula for $F^{-1} \Rightarrow$ we can't use the inverse function method
  - The density function doesn't have a compact support $\Rightarrow$ we can't use the Rejection Sampling method
- The *Box-Muller* method can be used to generate a couple of independent Normal Random Numbers

The Box-Muller method

We want to simulate $X \sim \mathcal{N}(0,1)$ and $Y \sim \mathcal{N}(0,1)$, independent, with joint probability density function:

$$f_{X,Y}(x,y) = \frac{1}{2\pi} \exp\left(-\frac{x^2+y^2}{2}\right)$$

We switch to polar coordinates $x = \rho\cos\theta$ and $y = \rho\sin\theta$ and we get:

$$\begin{aligned} f_{X,Y}(x,y) &= \frac{1}{2\pi} \exp\left(-\frac{x^2+y^2}{2}\right) dxdy \\ &= \frac{1}{2\pi} \exp\left(-\frac{\rho^2}{2}\right) \rho d\rho d\theta \\ &= f_{R,\Theta}(\rho,\theta) d\rho d\theta \end{aligned}$$

The Box-Muller method

- The joint density function of Random Variables R and Θ can be split into two independent parts:
  - $\frac{1}{2\pi}$ = density of $\Theta \sim \mathcal{U}_{[0,2\pi]}$
  - $\rho \exp\left(-\frac{\rho^2}{2}\right)$ = density of R
- The distribution function of R can then be deduced as:

$$F_R(\rho) = \mathbb{P}(R \leq \rho) = \int_0^\rho \rho e^{-\frac{t^2}{2}} dt = 1 - e^{-\frac{t^2}{2}}$$

- We can easily recognize the distribution function of an $\mathcal{E}\left(\frac{1}{2}\right)$ Squared. Finally:

$$R^2 \sim \mathcal{E}\left(\frac{1}{2}\right) \quad \Theta \sim (U)_{[0,2\pi]}$$

The Box-Muller Algorithm

- We first simulate two Uniform Random Variables
  - $U_1 \sim \mathcal{U}_{[0,1]} \longrightarrow R = \sqrt{-2lnU_1}$
  - $U_2 \sim \mathcal{U}_{[0,1]} \longrightarrow \Theta = 2\pi U_2$
- Then we set $X = R\cos\Theta$ and $Y = R\sin\Theta$
- By construction, the two Random Variables X and Y are independent (their joint density function is the product of their respective densities)

Simulation of $Z \sim \mathcal{N}\left(\mu, \sigma^2\right)$

If $X \sim \mathcal{N}(0,1)$, it is well known that

$$Z := \mu + \sigma X \ \sim \ \mathcal{N}\left(\mu, \sigma^2\right)$$

We therefore generate $X \sim \mathcal{N}(0,1)$ and compute $Z = \mu + \sigma X$

Application of the Central Limit Theorem

- Another method to simulate a *Normal Distribution* can be derived from the Central Limit Theorem
- We know that for n Random Variables, independent and identically distributed, with Expectation μ, Variance σ^2, and Sum S:

$$\frac{S - n\mu}{\sigma\sqrt{n}} \xrightarrow{law} \mathcal{N}(0,1) \quad (\textit{convergence reached for } n \approx 10)$$

- We choose a Random Variable $U \sim \mathcal{U}_{[0,1]}$, with $\mathbb{E}(U) = \frac{1}{2}$ and $\mathbb{V}(U) = \frac{1}{12}$
- If we simulate 12 independent *Uniform* variables, we have

$$\mathbb{E}\left(\sum_{i=1}^{12} U_i\right) = \sum_{i=1}^{12} \mathbb{E}(U_i) = 6 \quad \textit{and} \quad \mathbb{V}\left(\sum_{i=1}^{12} U_i\right) = \sum_{i=1}^{12} \mathbb{V}(U_i) = 1$$

- A *Normal* Random Variable X can therefore be built as $X = \left(\sum_{i=1}^{12} U_i\right) - 6$

Extension of Rejection Sampling

- The *Rejection Sampling* algorithm described earlier can be extended to density functions with no compact support
- One can simulate $X \sim f$ using another density g, assumed to exist with the following property:

$$\exists a \in \mathbb{R}^+, \forall x, \quad f(x) \leq g(x)$$

- We denote C_f, C_g and $C_a g$ the respective graphs of $f(x)$, $g(x)$ and $ag(x)$

Extension of Rejection Sampling

Generalized Rejection Sampling

- We generate a point $A = (X, Y)$ such that

$$\begin{cases} X \sim G \quad (\textit{for instance } X = G^{-1}(U_1)\ ) \\ Y \sim \mathcal{U}_{[0,ag(X)]} \quad (\textit{for instance } Y = a * G(X) * U_2\ ) \end{cases}$$

- If $Y \leq f(X)$, we keep X. Otherwise, we simulate another point
- We keep doing this until the condition $Y \leq f(X)$ is fulfilled

Proof of Generalized Rejection

$$\mathbb{P}(X \leq x_0) = \frac{\mathbb{P}(X \leq x_0, Y \leq f(X))}{\mathbb{P}(Y \leq f(X))}$$

$$\mathbb{P}(X \leq x_0, Y \leq f(X)) = \int_{-\infty}^{x_0} g(x)\mathbb{P}(ag(x)U \leq f(x))\,dx$$

$$= \int_{-\infty}^{x_0} g(x)\mathbb{P}\left(U \leq \frac{f(x)}{ag(x)}\right)dx = \int_{-\infty}^{x_0} g(x)\frac{f(x)}{ag(x)}dx = \frac{F(x_0)}{a}$$

$$\mathbb{P}(Y \leq f(X)) = \int_{\mathbb{R}} g(x)\mathbb{P}(ag(x)U \leq f(x))\,dx = \frac{1}{a}$$

And we finally have:

$$\mathbb{P}(X \leq x_0) = F(x_0)$$

Application to the Normal Distribution

Let ϕ the density function of the *Normal* distribution. We have:

$$\forall x, \quad \phi(x) = \frac{\exp\left(-\frac{x^2}{2}\right)}{\sqrt{2\pi}} \leq \frac{\exp\left(\frac{1}{2} - |x|\right)}{\sqrt{2\pi}}$$

Because

$$\exp\left(-\frac{x^2}{2}\right) \leq \exp\left(\frac{1}{2} - |x|\right) \Leftrightarrow \frac{x^2}{2} - |x| + \frac{1}{2} \geq 0 \Leftrightarrow \frac{(|x| - 1)^2}{2}$$

which is always true. The r.h.s can be rewritten as

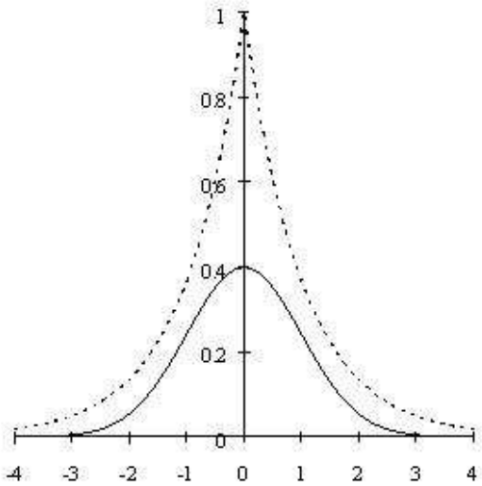
$$\frac{\exp\left(\frac{1}{2} - |x|\right)}{\sqrt{2\pi}} = \sqrt{\frac{2e}{\pi}} * \frac{1}{2} e^{-|x|} = \sqrt{\frac{2e}{\pi}} * g(x)$$

where $g(x)$ is the density of the *Double Exponential* distribution (an $\mathcal{E}(1)$ multiplied by a random sign generated through a *Head or Tail*

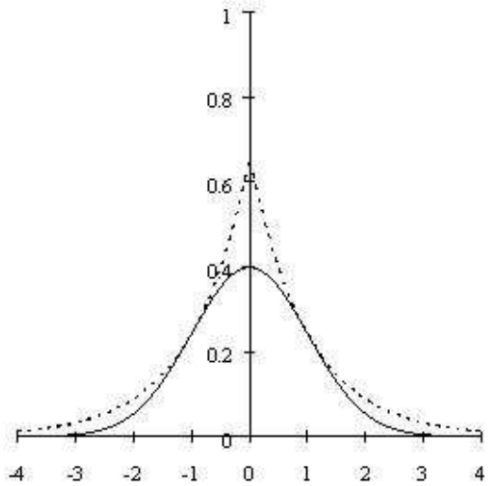
Effect of the choice of a

- The coarser the upper bound is, the more values we would need to draw in order to reach an acceptable value
- It is therefore recommended to choose a as small as possible

$a = 1$



$a = \sqrt{\frac{2e}{\pi}} < 1$



Bivariate Gaussian Distribution

The goal is to generate two Normal Distributions $X \sim \mathcal{N}(\mu_x, \sigma_x)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y)$ such that

$$\rho(X, Y) := \frac{Cov(X, Y)}{\sigma_x \sigma_y} = \rho$$

Algorithm

- First, generate $Z_1,\ Z_2 \sim \mathcal{N}(0, 1)$, independent, with joint density

$$f(z_1, z_2) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}\left(z_1^2 + z_2^2\right)\right]$$

- Then set:

$$\begin{cases} X &= \sigma_x Z_1 + \mu_x \\ Y &= \sigma_y \left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right) + \mu_y \end{cases}$$

Proof of the simulation routine

First, lets examine the marginal distributions of X and Y

$$\begin{aligned} X &= \sigma_x Z_1 + \mu_x \\ &= \sigma_x \mathcal{N}(0,1) + \mu_x \\ &= \mathcal{N}\left(\mu_x, \sigma_x^2\right) \end{aligned}$$

$$\begin{aligned} Y &= \sigma_y \left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right) + \mu_y \\ &= \sigma_y \left(\rho \mathcal{N}(0,1) + \sqrt{1-\rho^2}\mathcal{N}(0,1)\right) + \mu_y \\ &= \sigma_y \left(\mathcal{N}(0,\rho^2) + \mathcal{N}(0,1-\rho^2)\right) + \mu_y \\ &= \sigma_y \mathcal{N}(0,1) + \mu_y \\ &= \mathcal{N}\left(\mu_y, \sigma_y^2\right) \end{aligned}$$

Proof of the simulation routine

Second, we can check $Cov(X, Y)$ and $\rho(X, Y)$

$$\begin{aligned}
Cov(X, Y) &= \mathbb{E}\left[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))\right] \\
&= \mathbb{E}\left[(\sigma_x Z_1 + \mu_X - \mu_X)\left(\sigma_Y\left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right) + \mu_Y - \mu_Y\right)\right] \\
&= \mathbb{E}\left[(\sigma_x Z_1)\left(\sigma_Y\left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right)\right)\right] \\
&= \sigma_X \sigma_Y \mathbb{E}\left[\rho Z_1^2 + \sqrt{1-\rho^2} Z_1 Z_2\right] \\
&= \sigma_X \sigma_Y \rho \mathbb{E}\left[Z_1^2\right] \\
&= \sigma_X \sigma_Y \rho
\end{aligned}$$

$$\rho(X, Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \rho$$