IE 525 - Numerical Methods in Finance

Monte Carlo simulation - Generating random variates

Liming Feng

Dept. of Industrial & Enterprise Systems Engineering
University of Illinois at Urbana-Champaign

- A **random variate** is a numeric outcome of a random variable or from a certain distribution
- Generating random variates from the uniform distribution $U[0,1]$
- **Inverse transform** method
- **Acceptance-rejection** method
- Generating normal random variates

Modulo operation

- Linear congruential generators that are used to generate uniform random variates are based on the modulo operation
- **Modulo operation** finds the remainder after division

$$7 \bmod 5 = 2, \quad 10 \bmod 5 = 0, \quad 4 \bmod 5 = 4$$

"$2 = 7 \bmod 5$" reads "2 is **congruent** to 7 modulo 5"

- 5 is called the **modulus**; $0, 1, 2, 3, 4$ are all possible outcomes of $n \bmod 5$, $\forall$ integer n

Linear congruential generators

- Want to generate uniform random variates between 0 and 1
- **Linear congruential** generator:

$$x_{i+1} = (ax_i + c) \mod m$$

$$u_{i+1} = x_{i+1}/m$$

- a : the multiplier, m : the modulus; $a, m > 0, c \geq 0$ are all integers
- You specify the initial value x_0 (called the **seed**). The above will generate integers x_i's that are between 0 and $m-1$, and u_i's that are between 0 and 1

Example

- Let $a = 6, m = 11, c = 0, x_0 = 1$

  $$x_{i+1} = 6x_i \mod 11$$

  generates $1, 6, 3, 7, 9, 10, 5, 8, 4, 2$ and then repeats. In this example, the generator has **full period**
- Let $a = 3, m = 11, c = 0, x_0 = 1$

  $$x_{i+1} = 3x_i \mod 11$$

  generates $1, 3, 9, 5, 4$ and then repeats
- Would like to achieve full period for a large m

Conditions for full period

- Often take $c = 0$ for better speed
- When $c = 0$, m is prime, full period is achieved for any $x_0 \neq 0$ if
  - $a^{m-1} - 1$ is a multiple of m
  - $a^j - 1$ is not a multiple of m for $j = 1, \cdots, m-2$
- E.g., $m = 2^{31} - 1 = 2,147,483,647$ and $a = 39373$ leads to a full period generator
- Will repeat after about 2.1 billion replicates

MRG and combined generators

- **Multiple recursive generator**

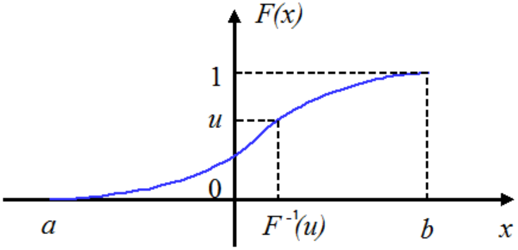
$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k}) \bmod m$$

- One may combine linear congruential generators or multiple recursive generators to extend the period
- The **L'Ecuyer algorithm** on p.52 (Fig 2.3) is a combined generator with a period of around 2^{185}, great uniformity, and fast implementation

- Variates generated as above only mimic randomness. The sequence is completely **deterministic** and even exhibits some regular patterns (lattice structure)
- Tests have been done for uniformity and independence
- Can easily produce an identical sequence by using the same seed. This is useful when one wants to repeat the simulation using the same variates

Inverse transform method

- Suppose the cdf of X is F. **Inverse transform method**: generating X from $F^{-1}(U)$, where $U \sim U[0,1]$
- Consider a continuous r.v. with strictly increasing cdf F on $[a,b]$ with $F(a)=0, F(b)=1$ ($a=-\infty, b=+\infty$ possible)
- Computing $x=F^{-1}(u)$ for $u \in [0,1] \Leftrightarrow$ finding $x \in [a,b]$ such that $F(x)=u$



Strictly increasing/continuous cdf

- $F^{-1}(u)$ is uniquely determined, strictly increasing, with

$$F(F^{-1}(u)) = u, \forall u \in [0,1]$$

$$F^{-1}(F(x)) = x, \forall x \in [a,b]$$

- Simulating X is equivalent to simulating $F^{-1}(U)$ since **X and $F^{-1}(U)$ have the same distribution**. Note that $U \leq F(x)$ is equivalent to $F^{-1}(U) \leq x$

$$\begin{aligned}
\mathbb{P}(X \leq x) &= F(x) \\
&= \mathbb{P}(U \leq F(x)) \\
&= \mathbb{P}(F^{-1}(U) \leq x)
\end{aligned}$$

Simulating exponential r.v.'s

- To simulate an **exponential** r.v. $X \sim Exp(\lambda)$,

$$F(x) = 1 - e^{-\lambda x}, x \geq 0 \Rightarrow F^{-1}(u) = -\frac{1}{\lambda}\ln(1-u), 0 \leq u \leq 1$$

- For $U \sim U[0,1]$, X can be generated from $-\frac{1}{\lambda}\ln(1-U)$
- Since $1-U$ is still uniform on $[0,1]$, we can generate X simply from $-\frac{1}{\lambda}\ln(U)$

- Simulate X from **Laplace** distribution with density $g(x) = \frac{1}{2}e^{-|x|}, x \in \mathbb{R}$ and cdf

$$G(x) = \begin{cases} \frac{1}{2}e^{x}, & x \leq 0 \\ 1 - \frac{1}{2}e^{-x}, & x > 0 \end{cases}$$

- Let Y be exponential with $\lambda = 1$, and U be $U[0,1]$. Then X can be generated from $-Y\mathbf{1}_{\{0 \leq U \leq 0.5\}} + Y\mathbf{1}_{\{0.5 < U \leq 1\}}$
  1. generate U_1, U_3 from $U[0,1]$
  2. $Y = -\ln(U_1)$
  3. if $U_3 \leq 0.5$ return $-Y$; else return Y

Why

- For any $x \leq 0$

$$\begin{aligned} & \mathbb{P}(-Y\mathbf{1}_{\{0\leq U\leq 0.5\}} + Y\mathbf{1}_{\{0.5<U\leq 1\}} \leq x) \\ = \; & \mathbb{P}(0 \leq U \leq 0.5, Y \geq -x) \\ = \; & \frac{1}{2}e^{x} \end{aligned}$$

For any $x > 0$

$$\begin{aligned} & \mathbb{P}(-Y\mathbf{1}_{\{0\leq U\leq 0.5\}} + Y\mathbf{1}_{\{0.5<U\leq 1\}} \leq x) \\ = \; & 1 - \mathbb{P}(-Y\mathbf{1}_{\{0\leq U\leq 0.5\}} + Y\mathbf{1}_{\{0.5<U\leq 1\}} > x) \\ = \; & 1 - \mathbb{P}(0.5 < U \leq 1, Y > x) \\ = \; & 1 - \frac{1}{2}e^{-x} \end{aligned}$$

No explicit form for F^{-1}

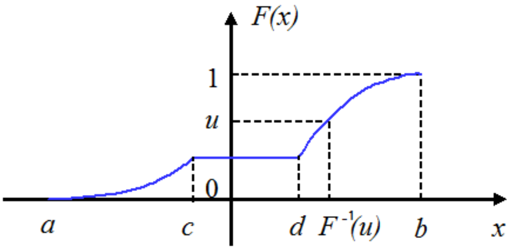
- If F^{-1} is not known explicitly, use the **Newton-Raphson** method
- To compute $x = F^{-1}(u)$, find the root of the equation $F(x) - u = 0$: start with an initial value x_0

$$x_{n+1} = x_n - \frac{F(x_n) - u}{F'(x_n)}$$

  $F'(x_n) = f(x_n)$, where f is the pdf of the distribution
- Repeat until $|F(x_{n+1}) - u| < \epsilon$ for some error tolerance level ϵ

Flat cdf

- Consider a cdf F that's flat on $[c, d]$



- The r.v. has zero probability of taking a value on $[c, d]$
- What is $F^{-1}(u)$ at $u = F(c)$?

Generalized inverse

- Let F be a cdf (and hence nondecreasing, right continuous) with $F(a-) = 0, F(b) = 1, 0 < F(x) < 1, \forall x \in (a, b)$ ($a = -\infty, b = +\infty$ possible)
- Define **generalized inverse** $F^{-1}(u)$ as follows:

$$F^{-1}(u) = \inf\{x \in [a, b] : F(x) \geq u\}$$

- Then F^{-1} is uniquely determined, non-decreasing with

$$F^{-1}(F(x)) \leq x, \forall x \in [a, b]$$

$$F(F^{-1}(u)) \geq u, \forall u \in [0, 1]$$

$F^{-1}(F(x)) \leq x$

- $F^{-1}(F(x)) \leq x, \forall x \in [a, b]$

$$F^{-1}(F(x)) = \inf\{y \in [a, b] : F(y) \geq F(x)\} \leq x$$

  since $x \in \{y \in [a, b] : F(y) \geq F(x)\}$

- In the example where F is flat on $[c, d]$,

$$c = F^{-1}(F(d)) < d$$

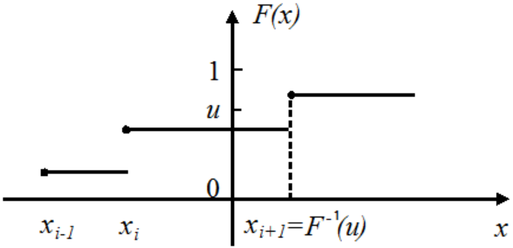
- $F(F^{-1}(u)) \geq u, \forall u \in [0,1]$
  since $F^{-1}(u) = \inf\{x \in [a,b] : F(x) \geq u\}$, there exists a decreasing sequence $\{x_n\} \in \{x \in [a,b] : F(x) \geq u\}$ such that $x_n \downarrow F^{-1}(u)$. By right continuity of F:

$$F(F^{-1}(u)) = \lim_{n \to +\infty} F(x_n) \geq u$$

  since each $F(x_n) \geq u$

Discrete distribution

- Consider a **discrete** distribution



- For any $F(x_i) < u \leq F(x_{i+1})$, $F^{-1}(u) = x_{i+1}$
- When $F(x_i) < u < F(x_{i+1})$, $F(x_{i+1}) = F(F^{-1}(u)) > u$

Inverse transform method in general

- $U \leq F(x)$ is equivalent to $F^{-1}(U) \leq x$ and hence X can still be simulated from $F^{-1}(U)$ since:

$$U \leq F(x) \Rightarrow F^{-1}(U) \leq F^{-1}(F(x)) \leq x$$

$$F^{-1}(U) \leq x \Rightarrow U \leq F(F^{-1}(U)) \leq F(x)$$

Simulating discrete r.v.

- Suppose X takes values x_1, x_2, x_3

$$X = \begin{cases} x_1, & \text{with probability } F(x_1) \\ x_2, & \text{with probability } F(x_2) - F(x_1) \\ x_3, & \text{with probability } F(x_3) - F(x_2) \end{cases}$$

- Let U be uniform on $[0, 1]$. Generate X according to

$$X = \begin{cases} x_1, & 0 \leq U \leq F(x_1) \\ x_2, & F(x_1) < U \leq F(x_2) \\ x_3, & F(x_2) < U \leq F(x_3) = 1 \end{cases}$$

Simulating from conditional distribution

- Let F be the cdf of X, $a < b, F(a) < F(b)$
- Let U be uniform on $[0,1]$. To generate X **conditional** on $a < X \leq b$, let $V = F(a) + (F(b) - F(a))U$ and return $F^{-1}(V)$

$$
\begin{aligned}
\mathbb{P}(F^{-1}(V) \leq x) &= \mathbb{P}(V \leq F(x)) \\
&= \mathbb{P}(F(a) + (F(b) - F(a))U \leq F(x)) \\
&= \frac{F(x) - F(a)}{F(b) - F(a)} \\
&= \mathbb{P}(X \leq x | a < X \leq b), \forall a < x \leq b
\end{aligned}
$$

Acceptance-rejection

- Want to generate a variate from a distribution with pdf $f(x)$
- Suppose you can find a distribution with pdf $g(x)$ that is easy to generate and

$$f(x) \leq cg(x)$$

  for some $c \geq 1$
- To generate from distribution $f(x)$
  1. generate X from distribution g
  2. generate U from $U[0, 1]$
  3. if $U \leq \frac{f(X)}{cg(X)}$, return X, otherwise, go to step 1

Why does it work

- Call the random variable generated using the above algorithm Y. Then Y has distribution $f(x)$:

$$\begin{aligned}
\mathbb{P}(Y \leq y) &= \mathbb{P}(X \leq y | U \leq \frac{f(X)}{cg(X)}) \\
&= \frac{\mathbb{P}(X \leq y, U \leq \frac{f(X)}{cg(X)})}{\mathbb{P}(U \leq \frac{f(X)}{cg(X)})} \\
&= \int_{-\infty}^{y} f(x)dx
\end{aligned}$$

Acceptance rate

- Using iterated conditioning,

$$\begin{aligned}\mathbb{P}\left(X \le y, U \le \frac{f(X)}{cg(X)}\right) &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{1}_{\{X \le y, U \le \frac{f(X)}{cg(X)}\}} | X\right]\right] \\ &= \int_{-\infty}^{\infty} \mathbf{1}_{x \le y} \frac{f(x)}{cg(x)} g(x) dx \\ &= \frac{1}{c} \int_{-\infty}^{y} f(x) dx\end{aligned}$$

$$\mathbb{P}\left(U \le \frac{f(X)}{cg(X)}\right) = \frac{1}{c} \int_{-\infty}^{\infty} f(x) dx = \frac{1}{c}$$

- **Acceptance rate** is $1/c$. Smaller c preferred
- What's the expected number of iterations needed to generate one variate using acceptance-rejection?

Acceptance-rejection for normal distribution

- Consider a standard normal distribution with pdf $f(x)$. Let $g(x)$ be the density of a Laplace distribution

$$\frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}} e^{|x| - \frac{1}{2}x^2} = \sqrt{\frac{2}{\pi}} e^{\frac{1}{2} - \frac{1}{2}(|x|-1)^2)} \le \sqrt{\frac{2e}{\pi}} = c \approx 1.3155$$

- To generate a standard normal random variate
  1. generate U_1, U_2 from $U[0,1]$
  2. $X = -\ln(U_1)$
  3. if $U_2 > e^{-\frac{1}{2}(X-1)^2}$, go to step 1
  4. else, generate U_3 from $U[0,1]$
  5. if $U_3 \le 0.5$, return $-X$; else return X

Acceptance-rejection vs inverse transform

- Acceptance-rejection is applicable when pdf is known but cdf is not
- Acceptance-rejection generalizes to multivariate distributions, while inverse transform doesn't
- To generate one variate, acceptance-rejection may need many uniform variates. This affects the performance of methods such as quasi-Monte carlo

Generating normal random variates

- For any normal r.v. X with mean μ and standard deviation σ

$$Z = \frac{X - \mu}{\sigma} \sim N(0, 1), \quad X = \mu + \sigma Z$$

- It suffices to generate standard normal random variates
- A multivariate normal r.v. can be generated from i.i.d. $N(0, 1)$ r.v.'s

Box-Muller

- **Box-Muller**: Suppose $Z_1, Z_2 \sim N(0,1)$ are independent. Let

$$Z_1 = r\cos(\theta), \quad Z_2 = r\sin(\theta)$$

  Then $r^2 \sim Exp(1/2)$ and $\theta \sim U[0, 2\pi]$, r and θ are independent.

  Box Muller algorithm:

  1. *Simulate two independent uniform r.v.'s:* $U_1, U_2 \sim U[0,1]$
  2. $r = \sqrt{-2\ln(U_1)}, \quad \theta = 2\pi U_2$
  3. $Z_1 = r\cos(\theta), \quad Z_2 = r\sin(\theta)$

- Two copies of $U[0,1]$ generate two copies of $N(0,1)$

Inverse transform method

- Inverse transform method for simulating a standard normal r.v. with cdf $\Phi(x)$ and pdf $\phi(x)$: need to **compute $\Phi^{-1}(u)$** for $u \in (0,1)$
- The **Beasley-Springer-Moro algorithm** on p.68 (Fig 2.13) is used to approximate $\Phi^{-1}(u)$, with maximum absolute error of 3×10^{-9} for $\Phi(-7) \leq u \leq \Phi(7)$
- Newton-Raphson can be used to further improve accuracy

$$x_{n+1} = x_n - \frac{\Phi(x_n) - u}{\phi(x_n)}$$

- Need an accurate algorithm to evaluate $\Phi(x)$

Evaluating $\Phi(x)$

- The **Marsaglia-Zaman-Marsaglia algorithm** on p.70 (Fig 2.15) is used to evaluate $\Phi(x)$, with a maximum relative error of 10^{-12}
- Inverse transform method for simulating from $N(0,1)$ (assuming one Newton step)
  1. Generate U from $U[0,1]$
  2. Compute $x_0 \approx \Phi^{-1}(U)$ using the Beasley-Springer-Moro algorithm
  3. Compute $\Phi(x_0)$ using the Marsaglia-Zaman-Marsaglia algorithm
  4. Return $x_1 = x_0 - \frac{\Phi(x_0)-u}{\phi(x_0)}$
- If more Newton steps used, repeat 3 and 4 above

Cholesky decomposition for multivariate normal

- If $Z \sim N(0, I_d)$, then

$$X = \mu + AZ \sim N(\mu, AA^\top) \text{ for any } d \times d \text{ matrix } A$$

- For $X \sim N(\mu, \Sigma)$, find A such that $AA^\top = \Sigma$
- **Cholesky decomposition** (when Σ is positive definite)

$$\Sigma = \begin{pmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & \vdots & \ddots & \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix} \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ & A_{22} & \cdots & A_{n2} \\ & & \ddots & \vdots \\ & & & A_{nn} \end{pmatrix}$$

Bivariate normal

- Bivariate normal r.v.'s

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

- Cholesky decomposition

$$AA^\top = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} A_{11} & A_{21} \\ 0 & A_{22} \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

Equations to solve:

$$A_{11}^2 = \sigma_1^2, \quad A_{11}A_{21} = \rho\sigma_1\sigma_2, \quad A_{21}^2 + A_{22}^2 = \sigma_2^2$$

Simulating bivariate normal

- Assuming positive diagonal entries for A, the solution is

$$A = \begin{pmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sqrt{1-\rho^2}\,\sigma_2 \end{pmatrix}$$

- *Simulate a bivariate normal r.v.* $X = (X_1, X_2)^\top$
  1. *Simulate two independent standard normal r.v.'s* $Z_1, Z_2 \sim N(0,1)$
  2. $X_1 = \mu_1 + \sigma_1 Z_1$
  3. $X_2 = \mu_2 + \rho\sigma_2 Z_1 + \sqrt{1-\rho^2}\,\sigma_2 Z_2$

Cholesky decomposition in general

- For positive definite Σ, entries of A are obtained by solving

$$\Sigma_{ij} = \sum_{k=1}^{j} A_{ik} A_{jk}, \quad j \leq i$$

The solutions are

$$A_{ij} = (\Sigma_{ij} - \sum_{k=1}^{j-1} A_{ik} A_{jk}) / A_{jj}, \quad j < i$$

$$A_{ii} = \sqrt{\Sigma_{ii} - \sum_{k=1}^{i-1} A_{ik}^2}$$

- See p.73 (Fig 2.16) for the implementation of the Cholesky decomposition