Scientific computation/Monte Carlo method
Applied Stochastic Processes (FIN 514)

Jaehyuk Choi

2017-18, Module 1 (Fall)
September 25, 2017

Scientific Computation

- Most of science (including finance) problems can not be solved analytically. Even analytic/algebraic solutions still requires to be computed!
- We need problem solving methods tailored for computation.
- Scientific computation is a branch of (applied)mathematics/computer science to research efficient computation method.
- Scientific computation is also an important part of financial engineering/quantitative finance.

Example: numerical root finding

- Abel (1802-1829) proved that there is no general *algebraic* solution for the roots of a quintic equation ($a_5x^5 + \cdots + a_1x + a_0 = 0$)
- However, we can find the root numerically:
  - Bisection

$$x_{n+1} = \frac{a + x_n}{2} \quad \text{or} \quad \frac{x_n + b}{2}$$

  - Newton's method

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

  - Newton-Raphson method

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

  - Square root: $x = \sqrt{y}$ [Demo]

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{y}{x_n}\right)$$

Tips and cautions

- In general, carefully translate math formulas to computer implementation
- Avoid repeated evaluation. Special functions, $\exp, \sin, \cos, \log$, are several times expensive than that of multiplication.
- Even polynomials (Horner's rule):

$$a_0 + a_1 x + \cdots + a_n x^n \quad \text{v.s.} \quad a_0 + x(a_1 + x(\cdots(a_{n-1} + x\ a_n)))$$

- In computer, small difference (machine epsilon) is rounded to 0, which is unexpected and dangerous. For example,

$$f(x) = \sqrt{x+1} - \sqrt{x}$$

we get $f(10^{16}) = 0$. So better use the equivalent same formula, [Python Demo]

$$f(x) = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

Monte-Carlo (MC) Method

- Even using computer, many problems can not be solved with deterministic (or convergent) method. So we need MC.
- In finance, the focus is on the price of derivative, which is the expected value of stochastic payout

$$E(f(x)) \quad \text{or} \quad E(f(x_1, \cdots, x_D))$$

  where $x, x_1, \cdots, x_n$ are (possibly correlated) random numbers.
- MC is favored in a large dimensions (D) as the cost of the deterministic method grows exponentially ($\sim e^D$) whereas the cost of MC glows linearly ($\sim D$).
- Example: MC computation of π [Python Demo]

Generation of Uniform Random Number(RN)

The uniform random number between 0 and 1 is the mother of all RNs. By nature, computer can not generate true RNs. In fact true RNs are not even desirable as we want to control and reproduce the result. So we generate pseudo-RNs in the following way.

$$X_{n+1} = AX_n + B \mod C, \quad U_n = X_n/C$$

X_0 is the **seed** of the RN generation. Using a fixed seed, you get the same result. Two popular classic choices are

- $A = 65,539 = 2^{16} + 3, \quad C = 2^{31}, \quad (B = 0)$
- $A = 16,807 = 7^5, \quad C = 2^{31} - 1, \quad (B = 0)$

Discrete RN

Bivariate RNs, e.g., random walk, which takes 1 and -1 with probabilities p and $1-p$ resp. can be generated as

$$X_k = \begin{cases} 1 & \text{if } U_k < p \\ -1 & \text{if } U_k > p \end{cases},$$

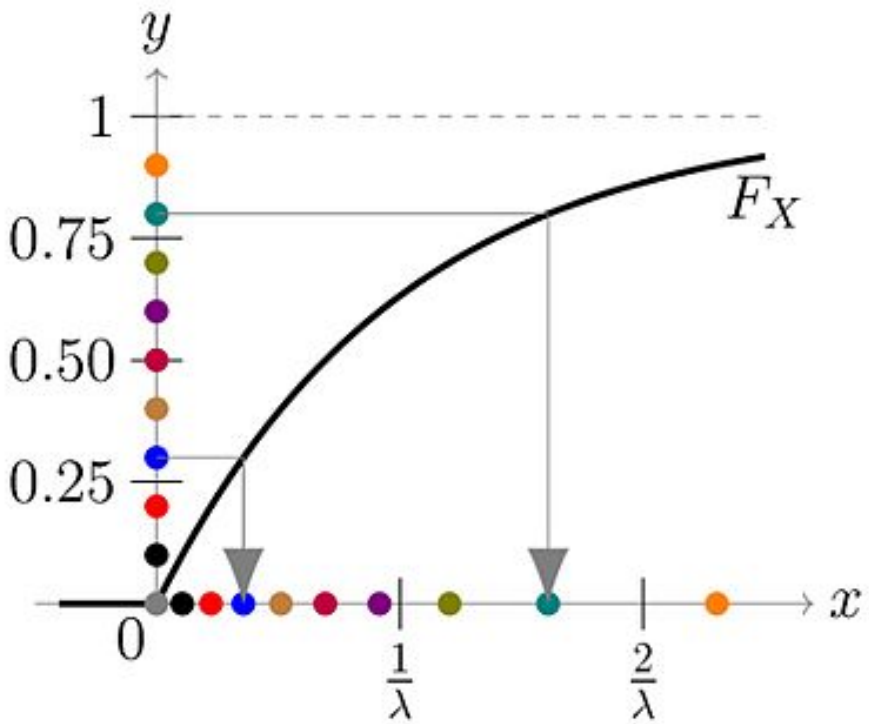
where U_k is uniform RNs. Multiple values can be similarly generated.

RN from a general distribution

If we know the cumulative density function (CDF) of a distribution $P(x)$, the inverse CDF gives random numbers following the distribution,

$$X = P^{-1}(U)$$

However, analytic form of inverse CDF is rare. Numerical root-finding is expensive.



Gaussian Distribution

- Evaluation related to Gaussian distribution is critical important.
- Evaluation of normal CDF, $N(z) = \int n(z)dz$

$$1 - N(x) = 1 - (a_1 t + a_2 t^2 + a_3 t^3)e^{-x^2/2}, \quad t = 1/(1 + px/\sqrt{2})$$

where $p = 0.47047$, $a_1 = 0.3480242$, $a_2 = 0.0958798$, $a_3 = 0.7478556$.

Gaussian RN (Box-Muller Method)

- Evaluation of the normal inverse CDF, N^{-1} is expensive and has been a topic of research.
- However, there is a brilliant trick of generating Gaussian RN:
  For two-dimensional Gaussian random numbers (Z_1, Z_2)

$$P\{z_1^2 + z_2^2 < R^2\} = \int_{z_1^2+z_2^2<R^2} \frac{1}{\sqrt{2\pi}^2} e^{-\frac{1}{2}(z_1^2+z_2^2)} dz_1 dz_2$$
$$= \frac{2\pi}{2\pi} \int_{r=0}^{R} r e^{-r^2/2} dR = 1 - e^{-R^2/2}$$

The CDF of the variable, $Y = R^2 = Z_1^2 + Z_2^2$, is $P(y) = 1 - e^{-y/2}$ and we know the inverse CDF:

$$P^{-1}(u) = -2\log(1-u).$$

Gaussian RN (Box-Muller Method) cont.

- This means the inverse function,

$$Y = -2\log(1 - U_1) \quad \text{for a uniform RV } U_1$$

  is the correct sampling of Y and

$$R = \sqrt{Y} = \sqrt{-2\log U_1} \quad (\text{symmetry: } U_1 \leftrightarrow 1 - U_1)$$

  is the correct sampling of R.
- Now, Z_1 and Z_2 can be split as two components from an random angle $2\pi U_2$:

$$Z_1 = \sqrt{-2\log U_1}\,\cos(2\pi U_2), \quad Z_2 = \sqrt{-2\log U_1}\,\sin(2\pi U_2)$$

  so we get two Gaussian RNs.

Gaussian RN (Marsaglia-Polar Method)

- The improvement of Box-Muller method
- $V_{1,2} = 2U_{1,2} - 1$ so that $V_{1,2}$ is uniform RVs between -1 and 1.
- Take if $0 < W = V_1^2 + V_2^2 < 1$ and reject otherwise so that (V_1, V_2) is uniform random point on the unit circle.
- W has a uniform distribution on $[0, 1]$, so can replace U_1,

$$P\{W < x\} = \pi(\sqrt{x})^2/\pi = x$$

- Using the trigonometric properties,

$$(\frac{V_1}{\sqrt{W}}, \frac{V_2}{\sqrt{W}}) = \Big(\cos(2\pi U_2), \sin(2\pi U_2)\Big)$$

Gaussian RN (Marsaglia-Polar Method) cont.

- Finally we get

$$Z_1 = \sqrt{-2\log W}\, V_1/\sqrt{W} = V_1\sqrt{-2(\log W)/W}$$
$$Z_2 = \sqrt{-2\log W}\, V_2/\sqrt{W} = V_2\sqrt{-2(\log W)/W}$$

- Even after *wasting* 21% of (V_1, V_2) pairs, it is still more efficient by avoiding expensive $\sin$ and $\cos$ evaluations, so Marsaglia-Polar method is the current standard.

Reducing MC variance

- Anti-thetic method; (π by MC)
  - By symmetry, U is a uniform random number, so is $1 - U$
  - By symmetry, Z is a Gaussian random number, so is $-Z$
- Importance sampling method: ($\mathrm{P}(Z > 30)$ by MC)
  - Original probability = Modified probability $\times$ Correction
- Low-disccrepancy sequence
  - Qusai-random numbers evenly distributed over $(0, 1)$.
- Control Variate: for $g(x)$ similar to $f(x)$ and known $E(g(X))$,

$$E_{CV}(f(X)) = E_{MC}(f(X)) + \Big(E(g(X)) - E_{MC}(g(X))\Big)$$

[Python Demo]

Generation of correlated Gaussian RNs ($N = 2$)

- We want to generate two standard Gaussian RNs, W_1 and W_2, correlated by ρ,

$$\frac{E(W_1 W_2)}{E(W_1)E(W_2)} = E(W_1 W_2) = \rho.$$

- We can achieve that by separating correlate and de-correlated parts:

$$W_1 = Z_1,\ W_2 = \rho Z_1 + \sqrt{1-\rho^2}\, Z_2,$$

  where Z_1 and Z_2 are independent and standard RNs.
- You can easily prove that (see StoFin mid-term exam):

$$E(W_2^2) = 1 \quad \text{and} \quad E(W_1 W_2) = \rho$$

- You'll see this trick in spread option and stochastic differential equations for stochastic volatility models (Heston and SABR model)

Covariance/correlation matrix

For N-dimensional samples, $X = (X_1, \cdots, X_N)^T$, with mean 0 and stdev σ_k, the covariance matrix $\boldsymbol{\Sigma}$ is given as

$$\boldsymbol{\Sigma}_{ij} = \mathrm{Cov}(X_i, X_j) = E\left((X_i - \bar{X}_i)(X_i - \bar{X}_j)\right) \quad (\boldsymbol{\Sigma}_{ij} = \sigma_i^2).$$

The correlation matrix R is similarly defined as

$$R_{ij} = \mathrm{Corr}(X_i, X_j) = \frac{\boldsymbol{\Sigma}_{ij}}{\sigma_i \sigma_j} \quad (R_{ii} = 1)$$

- If $\{X_i\}$ are independent standard normals, $\Sigma = I$
- Symmetric: $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$ $(\Sigma_{ij} = \Sigma_{ij})$
- Positive-definite: $\boldsymbol{a}^T \Sigma \boldsymbol{a} \geq 0$ for any (row) vector $\boldsymbol{a}$:

$$\mathrm{Var}(\boldsymbol{a}X) = \mathrm{Var}(a_1 X_1 + \cdots + a_N X_N) = \boldsymbol{a}\boldsymbol{\Sigma}\boldsymbol{a}^T \geq 0$$

- For a linear combination, $\boldsymbol{L}X$, $\mathrm{cov}(\boldsymbol{L}X) = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T$.

Generation of correlated Gaussian RNs (N in general)

- $X = (X_i)^T$ with covariance matrix $\mathbf{\Sigma}$: we want to find such that $X = \boldsymbol{L}Z$ where $Z = (Z_i)^T$ is independent standard normals.
- Then,

$$\mathbf{\Sigma} = \mathrm{Var}(\boldsymbol{L}Z) = \boldsymbol{L}\boldsymbol{I}\boldsymbol{L}^T = \boldsymbol{L}\boldsymbol{L}^T$$

  This is the definition of the square-root matrix of $\mathbf{\Sigma}$: $\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^T$.
- The square-matrix $\boldsymbol{L} = \sqrt{\mathbf{\Sigma}}$ is not unique. Cholesky decomposition (lower triangular matrix) is a numerically stable and fast solution. [Python Demo]
- The 2 asset case ($\rho = \rho_{12}$):

$$\boldsymbol{L} = \begin{pmatrix} \sigma_1 & 0 \\ \sigma_2\rho & \sigma_2\sqrt{1-\rho^2} \end{pmatrix}, \quad \mathbf{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

MC of Brownian Motion(s)

- Standard BM: $W_T \sim \sqrt{T} Z_1$ where Z_1 is a standard normal RV.
- Arithematic Brownian motion (Bachelier model)
  - $$dF_t = \sigma_n dW_t \quad \Rightarrow \quad F_T \sim F_0 + \sigma_n \sqrt{T} Z_1$$
  - From $t = S$ to $t = T$: $F_T \sim F_S + \sigma_n \sqrt{T - S} Z_1$
  - For N assets: $\boldsymbol{F}_T = \boldsymbol{F}_0 + \boldsymbol{L}\sqrt{T}\, Z$, where $\boldsymbol{L}\boldsymbol{L}^T = T\boldsymbol{\Sigma}$
- Geometric Brownian motion (BSM model)
  - $$\frac{dF_t}{F_t} = \sigma\, dW_t \quad \Rightarrow \quad F_T \sim F_0 \exp\left(-\frac{1}{2}\sigma^2 T + \sigma\sqrt{T} Z_1\right)$$
  - From $t = S$ to $t = T$: $F_T \sim F_S \exp\left(-\frac{1}{2}\sigma^2 (T - S) + \sigma\sqrt{(T - S)} Z_1\right)$
  - For N assets: $F_k(T) = F_k(0) \exp\left(-\frac{1}{2} T \Sigma_{kk} + \boldsymbol{L}_{k*}\sqrt{T}\, Z\right)$, where $\boldsymbol{L}\boldsymbol{L}^T = T\boldsymbol{\Sigma}$ and $\boldsymbol{L}_{k*}$ is the k-th row vector of $\boldsymbol{L}$.
- [Python Demo]