Financial Mathematics 32000

Lecture 5

Roger Lee

2025 April 21

UNIT 3: Monte Carlo

Generating random variables

Monte Carlo estimate

Let Y be a discounted payoff. Example: $Y = e^{-\int_0^T r_t \mathrm{d}t}(S_T - K)^+$.

Want to calculate the time-0 price $C = \mathbb{E}Y$

Generate $Y_1, Y_2, \ldots$ independently and identically distributed as Y.

(*How?*)

Then the random variable

$$\hat{C}_M := \frac{Y_1 + Y_2 + \cdots + Y_M}{M}$$

is the Monte Carlo estimate of C. Note that

$$\mathbb{E}\hat{C}_M = C$$

is the expectation of the price, equal to the true price?

By the strong law of large numbers, with probability 1 we have

$$\hat{C}_M \to C \text{ as } M \to \infty.$$

as we do more simulations, the simulated discounted payoff will converge to the price

we use MC because we don't know the expected payouf and the var of the payout

How fast does convergence occur?

Let $\sigma^2 := \mathrm{Var}(Y)$. (Here σ does not denote volatility.) Then

$$\mathrm{Var}(\hat{C}_M) = \frac{1}{M^2}\mathrm{Var}(Y_1 + Y_2 + \cdots + Y_M) = \frac{1}{M^2}(M\sigma^2) = \frac{\sigma^2}{M}$$

By the Central Limit Theorem, we have convergence in distribution

$$\frac{\hat{C}_M - \mathbb{E}\hat{C}_M}{\sqrt{\mathrm{Var}\ \hat{C}_M}} \xrightarrow{d} N(0,1) \qquad \text{hence} \qquad \frac{\hat{C}_M - C}{\sigma/\sqrt{M}} \xrightarrow{d} N(0,1)$$

as $M \to \infty$. Conclusion still holds using sample stdev in place of σ:

$$\frac{\hat{C}_M - C}{\hat{\sigma}_M/\sqrt{M}} \xrightarrow{d} N(0,1)$$

because $\hat{\sigma}_M^2 \to \sigma^2$ where

$$\hat{\sigma}_M^2 := \frac{1}{M-1}\sum_{m=1}^{M}(Y_m - \hat{C}_M)^2.$$

is the "sample variance" and $\hat{\sigma}_M$ is the "sample standard deviation."

Confidence intervals

For 95% confidence interval, p =0.05, Cm + 1.96*std_error

So an asymptotic (for large M) confidence interval of $100(1-p)\%$ is

$$\left(\hat{C}_M - \mathcal{N}^{-1}(1-\frac{p}{2})\frac{\hat{\sigma}_M}{\sqrt{M}},\ \hat{C}_M + \mathcal{N}^{-1}(1-\frac{p}{2})\frac{\hat{\sigma}_M}{\sqrt{M}}\right)$$

1.96 1.96

where $\mathcal{N}$ is the standard Normal cdf.

- $\mathcal{N}^{-1}(1-\frac{p}{2})$ tells us: a radius of *how many standard deviations* of a normal distribution contains $100(1-p)\%$ of the probability? 1.96
- The $\hat{\sigma}_M/\sqrt{M}$ is called the *standard error*. how many $ is one stdev

  It gives us the estimated standard deviation of $\hat{C}_M$.
- Example: Let $p = 0.05$. Then $\mathcal{N}^{-1}(1-\frac{p}{2}) \approx 1.96$ and

  $$\left(\hat{C}_M - 1.96\frac{\hat{\sigma}_M}{\sqrt{M}},\ \hat{C}_M + 1.96\frac{\hat{\sigma}_M}{\sqrt{M}}\right)$$

  is an asymptotic 95% confidence interval for C.

Confidence intervals

- If $\hat{\sigma} = 20$ and we run $M = 10000$ simulations,

  then a 95% confidence interval has radius

$$1.96 \times \frac{20}{\sqrt{10000}} = 0.40$$

radius of 40 cents

  To reduce this to 0.04, we need to take $M = 1$ million.

radius of 4 cents

- So we will want to use *variance reduction* techniques, which reformulate the problem to keep σ small, or which carefully introduce dependence in the simulations to keep Var $\hat{C}_M$ small.

UNIT 3: Monte Carlo

Generating random variables

If we want the 23rd percentile, take the F^(-1) of 0.23 and we will get a number X that is smaller than 0
If we want the 88th percentile, take the F^(-1) of 0.88 and we will get a number X that is bigger than 0

Look at notebook

The inverse CDF method

- Assume the existence of a pseudo-random number generator whose output can be treated as if it is IID uniform on $(0, 1)$.
  Python: `numpy.random.Generator` has method `random()`
- To generate a random variable X having a CDF F, generate $U \sim \text{Uniform}(0, 1)$, then apply F^{-1}, the inverse CDF, to produce

$$X := F^{-1}(U)$$

As desired, $\mathbb{P}(X \leq x) = \mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x)$.
(If F not invertible, letting $F^{-1}(u) := \min\{x : F(x) \geq u\}$ works.)
Intuition: randomly choose the *percentile* between 0% and 100%, uniformly. The inverse CDF finds the corresponding value of X.

Generating normal random variables

- Python: `numpy.random.Generator` has method `normal()`

But what if you need to build your own?

- Could do $\mathcal{N}^{-1}(U)$, if an implementation of the inverse of the normal CDF $\mathcal{N}$ is available. Excel: `NORMSINV(RAND())`
- Box-Muller method: If (X, Y) are independent Normal$(0, 1)$, then $R := X^2 + Y^2$ has CDF $\mathbb{P}(R \leq r) = 1 - e^{-r/2}$. Given R, the point (X, Y) is uniformly distributed on the circle of radius $\sqrt{R}$. So generate *pairs* of independent normals by drawing U_1 and U_2 IID from a Uniform$(0, 1)$ distribution, and taking

SKIP BOX MULLER

$$R := -2\log(U_1)$$

$$(X, Y) := (\sqrt{R}\cos(2\pi U_2), \sqrt{R}\sin(2\pi U_2))$$

Simulating random variables in Python

default Random Number Generator

```
[1]  import numpy
     rng = numpy.random.default_rng(seed=0)
```

By starting w/ the same seed everytime, we will generate the same random numbers

```
[2]  rng.random(size=5)

     array([0.63696169, 0.26978671, 0.04097352, 0.01652764, 0.81327024])
```

```
[3]  rng.random(size=5)
```

if we put only random, it will be between 0 and 1

```
     array([0.91275558, 0.60663578, 0.72949656, 0.54362499, 0.93507242])
```

```
[4]  rng.normal(size=5)
```

Normal dist, no bounded domain

```
     array([-0.62327446,  0.04132598, -2.32503077, -0.21879166, -1.24591095])
```

```
[5]  rng.normal(size=(2,5))

     array([[-0.73226735, -0.54425898, -0.31630016,  0.41163054,  1.04251337],
            [-0.12853466,  1.36646347, -0.66519467,  0.35151007,  0.90347018]])
```

In some cases, no need to simulate entire path

If Y is a known function of random variables with distributions that you can directly simulate, then it is easy to generate $Y_1, Y_2, \ldots$

Example: Consider a call paying $(S_T - K)^+$ where S is GBM.

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma S_t \mathrm{d}W_t$$

Then

$$Y = e^{-rT}(S_T - K)^+ = e^{-rT}(S_0 e^{(r-\sigma^2/2)T+\sigma W_T} - K)^+$$

where $W_T \sim N(0,T)$. So let

the m-th simulation being run

$$Y_m := e^{-rT}(S_T - K)^+ = e^{-rT}(S_0 e^{(r-\sigma^2/2)T+\sigma\sqrt{T}Z^{(m)}} - K)^+$$

where the $Z^{(m)}$ are IID standard normal: $N(0,1)$.

But sometimes need to simulate entire path

- But what about more complicated dynamics, such as

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma(S_t, t)S_t\mathrm{d}W_t$$

here, sigma is not constant, but depends on the underlying

- Or what if σ or r follows a process driven by a second BM.
- What about more complicated contracts, such as an Asian option or a barrier option.

We may need to simulate the whole path.

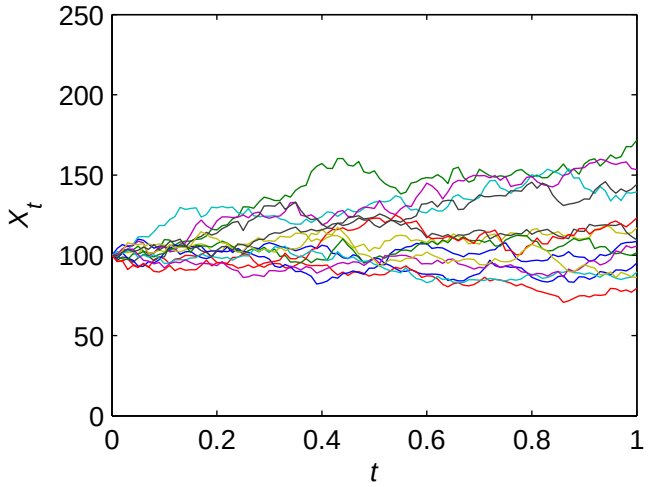
Geometric Brownian motion: $\mathrm{d}X_t = \mu X_t \mathrm{d}t + \sigma X_t \mathrm{d}W_t$

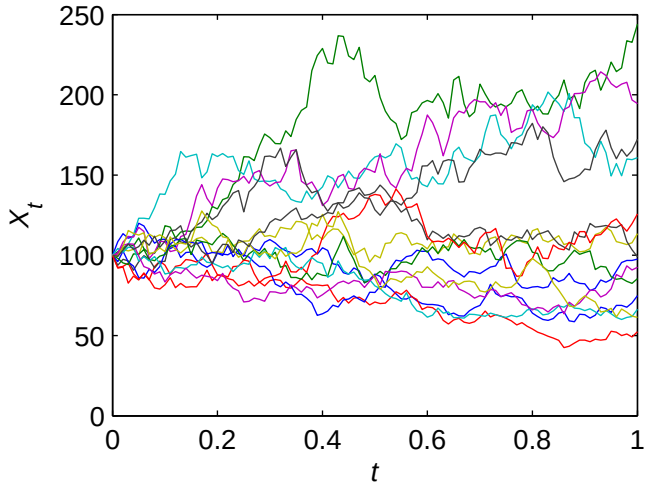
notebook: dlog(Xt) = …

Let $X_0 = 100$. Trajectories for $\mu = -0.15, +0.15$ and $\sigma = 0.20, 0.40$:

miu = 0.15

sigma = 0.2



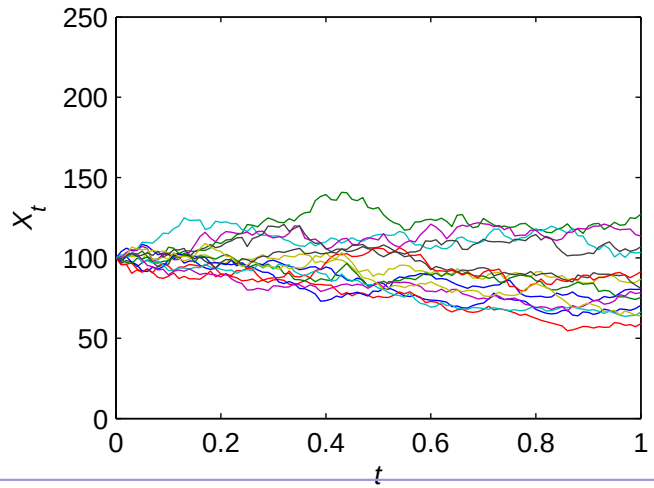


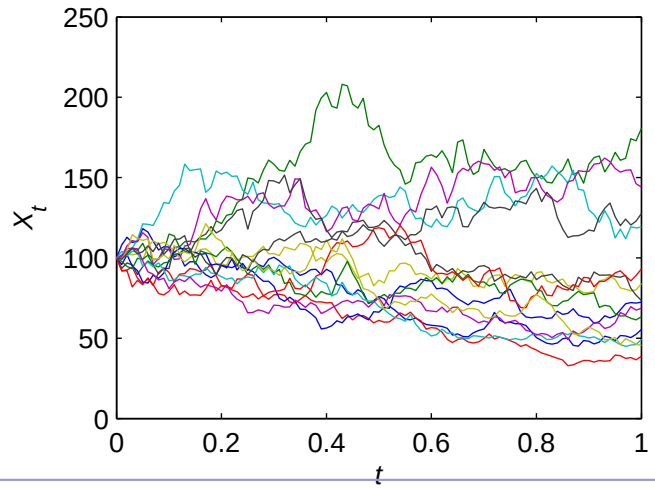
miu = 0.15

sigma = 0.4

miu = -0.15

sigma = 0.2





miu = -0.15

sigma = 0.4

Euler method to simulate path of a state variable

Suppose X satisfies

In one segment (ex. region I), we are fixing a and b
But the moment we go to the next time step, a new segment, region II, we have a new a and b
In the same region, between tn and tn+1, we have a fixed a and b
Look at notebook

$$\mathrm{d}X_t = a(X_t, t)\mathrm{d}t + b(X_t, t)\mathrm{d}W_t$$

Divide the time interval $[0, T]$ into N parts: $\Delta t = T/N$, $t_n = n\Delta t$.

Define the mth simulated path by initializing $X_0^{(m)} = X_0$,

and given X_{t_n}, obtain $X_{t_{n+1}}$ by

dWt

$$X_{t_{n+1}}^{(m)} = X_{t_n}^{(m)} + a(X_{t_n}^{(m)}, t_n)\Delta t + b(X_{t_n}^{(m)}, t_n)Z_n^{(m)}\sqrt{\Delta t}$$

where $Z_n^{(m)}$ are IID standard normal. Evaluate the discounted payoff for the mth path. Take the average across all paths $m = 1, \ldots, M$.

This extends directly to multidimensional state vectors X and multidimensional standard Brownian motion W.

Euler method convergence

With some assumptions on a and b, the Euler method has *weak* order of convergence 1 for general f, meaning SKIP

$$|\mathbb{E}f(X_T) - \mathbb{E}f(X_T^{(m)})| = O(\Delta t)$$

(Estimating $\mathbb{E}f(X_T^{(m)})$ produces additional error, not included here.)

Error analysis: Let $a_t = a(X_t)$ and $b_t = b(X_t)$. First subinterval:

$$\begin{aligned} X_{t_1} &= X_0 + \int_0^{t_1} a_t \mathrm{d}t + \int_0^{t_1} b_t \mathrm{d}W_t \\ &= X_0 + \int_0^{t_1} \left(a_0 + \int_0^t a\frac{\partial a}{\partial x} + \frac{1}{2}b^2\frac{\partial^2 a}{\partial x^2}\mathrm{d}s + \int_0^t b\frac{\partial a}{\partial x}\mathrm{d}W_s \right)\mathrm{d}t \\ &\quad + \int_0^{t_1} \left(b_0 + \int_0^t a\frac{\partial b}{\partial x} + \frac{1}{2}b^2\frac{\partial^2 b}{\partial x^2}\mathrm{d}s + \int_0^t b\frac{\partial b}{\partial x}\mathrm{d}W_s \right)\mathrm{d}W_t \end{aligned}$$

The Euler scheme keeps three terms to generate $X_{t_1}^{(m)}$

Crude analysis of discretization error of Euler scheme

SKIP

- Intuition for weak error: $|\mathbb{E}(X_T - X_T^{(m)})|$ is $O(\Delta t)$, because there are N time steps, and $\mathbb{E}$ of error is $O(\Delta t)^2$ at each step (the only nonzero-$\mathbb{E}$ error term in previous equation is the $\mathrm{d}s\mathrm{d}t$ term).
- Euler scheme has strong order of convergence 1/2, meaning that

$$\mathbb{E}|X_T - X_T^{(m)}| = O(\Delta t)^{1/2} \qquad \text{as } \Delta t \to 0$$

Intuition for strong error: Biggest ignored term is $\mathrm{d}W_s\mathrm{d}W_t$.

Variance of error at one time step $= O(\mathrm{Var}(\Delta W)^2) = O(\Delta t)^2$.

Variance of total error $X_T - X_T^{(m)}$ after N time steps $= O(\Delta t)$.

Standard deviation of $X_T - X_T^{(m)}$ after N time steps $= O(\Delta t)^{1/2}$.

This suggests that strong order of convergence is 1/2.

Milstein scheme

SKIP

Milstein scheme: Don't ignore the term

$$\int_0^{t_1} \int_0^t b(X_s) \frac{\partial b}{\partial x}(X_s) \mathrm{d}W_s \mathrm{d}W_t.$$

Approximate it as

$$b(X_0) \frac{\partial b}{\partial x}(X_0) \int_0^{t_1} \int_0^t \mathrm{d}W_s \mathrm{d}W_t = b(X_0) \frac{\partial b}{\partial x}(X_0) \int_0^{t_1} W_t \mathrm{d}W_t$$
$$= \frac{1}{2} b(X_0) \frac{\partial b}{\partial x}(X_0)(W_{t_1}^2 - t_1)$$

So Milstein is

$$X_{t_{n+1}}^{(m)} = X_{t_n}^{(m)} + a(X_{t_n}^{(m)})\Delta t + b(X_{t_n}^{(m)}) Z_n^{(m)} \sqrt{\Delta t} + \frac{1}{2} b \frac{\partial b}{\partial x}(X_{t_n}^{(m)})([Z_n^{(m)}]^2 - 1)\Delta t$$

Milstein has strong order of convergence 1, and weak order 1.

Weak convergence (important in option pricing): same order as Euler.

Covariance and correlation matrices

Recall that the covariance matrix of a zero-mean vector Z is $\mathbb{E}(ZZ^\top)$.

Let M be a real symmetric matrix. The following are equivalent:

- M is a covariance matrix of some vector.
- M is *positive semi-definite*, which means that $x^\top Mx \geq 0$ for all real vectors x.
- The eigenvalues of M are all nonnegative.
- The principal minors of M are all nonnegative. (Principal minors = the determinants of the matrices formed by crossing out any rows and corresponding columns of M).
- M has a *Cholesky decomposition* $LL^\top = M$ for some real lower-triangular matrix L with diagonal entries ≥ 0.

Covariance and correlation matrices

Moreover, the following are equivalent:

- M is a correlation matrix
- M is a covariance matrix and its diagonal elements are all 1.

Moreover, if M is a 3×3 matrix, the following are equivalent

- M is a correlation matrix.
- M is symmetric, its entries $\in [-1, 1]$, its diagonal entries $= 1$, and $\det M \geq 0$.

Corr(x,y) = Cov(x/stdev(x), y/stdev(y))

Interview question

Suppose $\text{Corr}(X, Y) = \text{Corr}(X, Z) = \text{Corr}(Y, Z) = \rho$.

What are the possible values of ρ?

Notebook

Generating correlated Brownian motions

To get a D-dimensional vector $\bar{W}$ of BM with correlation matrix H, find a matrix $L \in \mathbb{R}^{D\times D}$ such that $LL^\top = H$, and let $\bar{W} = LW$, where W is standard BM in $\mathbb{R}^D$.

So the process $\mathrm{d}X_t = a(X_t, t)\mathrm{d}t + b(X_t, t)\mathrm{d}\bar{W}_t$ becomes

$$\mathrm{d}X_t = a(X_t, t)\mathrm{d}t + b(X_t, t)L\ \mathrm{d}W_t$$

Simulation by Euler method:

$$X^{(m)}_{t_{n+1}} = X^{(m)}_{t_n} + a(X^{(m)}_{t_n}, t_n)\Delta t + b(X^{(m)}_{t_n}, t_n)L\ Z^{(m)}_n\sqrt{\Delta t}$$

where $Z^{(m)}_n$ are IID standard normal in $\mathbb{R}^D$. How to find L?

- `numpy.random.Generator.multivariate_normal` generates LZ
- `numpy.linalg.cholesky` returns L given H

Generating correlated Brownian motions

Example: if we want $\bar{W} = \binom{W^{[1]}}{W^{[2]}}$ with $\text{corr}(\Delta W^{[1]}, \Delta W^{[2]}) = \rho$, then

$$H = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

and Cholesky finds $LL^\top = H$ where

$$L = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix}$$

If Cholesky routine unavailable, can solve for L by traversing [the upper or lower triangular part of] H entry-by-entry. Each entry gives rise to an equation involving elements of L and only one unknown.

Generating correlated Brownian motions

Sometimes it is not necessary to simulate the entire path.

- Suppose $X_T - X_0$ is known to be multivariate normal with mean μT and covariance matrix HT.
- Suppose the option payoff depends only on X_T.

Then no need to divide $[0, T]$ into N steps. No need for Euler.

- Just generate

$$\underset{\text{Dx1}}{X_T^{(m)}} := \underset{\text{Dx1}}{X_0} + \underset{\text{Dx1}}{\mu T} + \underset{\text{DxD}}{L\sqrt{T}}\underset{\text{Dx1}}{Z^{(m)}}$$

where $LL^\top = H$ and the $Z^{(m)}$ are IID standard normal in $\mathbb{R}^D$.