Monte Carlo Method in Option Pricing

Group 13 SID: 500671702

September 9, 2022

1 Introduction

1.1 Background of Monte Carlo method

Monte Carlo method is referred as algorithms that use random numbers to solve problems which might be deterministic in principle. Monte Carlo Methods has become more practical as computational cost of generating random number becomes cheaper.

Monte Carlo method is frequently used in option pricing when there are numerous sources of uncertainty and random features. In the fields of pricing financial derivatives and estimating transaction risk, the dimension of variables may be thousands, and computational difficulty increases exponentially, which is called "Curse of Dimensionality". Monte Carlo method performs well since its computational complexity no longer depends on the number of dimensions.

1.2 Abstract

This report introduces different techniques in Monte Carlo method, including two methods of variance reduction (antithetic variables, importance sampling, control variates) and three methods of generating random paths (Milstein and Euler). Then Finite Difference method is introduced and compared with Monte Carlo method. Lastly, this report provides two numerical examples of European and Asian call option pricing. The first one focuses on comparing performances of applicable Monte Carlo techniques and Crank-Nicolson scheme in Finite Difference, and the second one focuses on generating random paths and variance reduction.

2 Basic Monte Carlo method

2.1 Basic idea and method

The basic idea of Monte Carlo method is to repeat the experiment many times or use a sufficiently long simulation run to obtain quantities of interest using Law of Large Numbers. It efficiently reduces computational difficulty and increases the accuracy of solutions.

Lemma 2.1

***Law of Large Numbers**: $X_1, X_2, ..., X_N$ are iid random variables with $E[X_i] = \mu$, then*

$$\frac{1}{n}\sum_{i=1}^{n} X_i \to \mu \text{ as } n \to \infty \text{ with probability of 1} \tag{2.1}$$

The way of implementing basic Monte Carlo method to compute $E[f(x)]$ is:

- Generate N *iid* copies of X, denote as $X_1, X_2, ..., X_N$

- $E[f(x)] \approx \frac{1}{N}\sum_{i=1}^{n} f(x_i)$

EXAMPLE 1. $\int_a^b f(x)\,\mathrm{d}x = (b-a)\int_a^b f(x)\frac{1}{b-a}\,\mathrm{d}x = (b-a)Ef(x) \approx \frac{b-a}{N}\sum_{i=1}^{n} f(x)$, *where* $X \sim U(a,b)$.

2.2 Accuracy and efficiency

Lemma 2.2

Central Limit Theorem: *$X_1, X_2, ..., X_N$ are iid random variables with $E[X_i] = \mu$, $Var[X_i] = \sigma^2 \leq \infty$, then*

$$\frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu}{\frac{\sigma}{\sqrt{n}}} \xrightarrow{d} N(0,1) \tag{2.2}$$

Combining with the empirical rule,

$$P(|\frac{1}{N}\sum_{i=1}^{N} X_i - \mu| \geq \frac{3\sigma}{\sqrt{N}}) = P(|\frac{\frac{1}{N}\sum_{i=1}^{N} X_i - \mu}{\frac{\sigma}{\sqrt{N}}}| \geq \frac{3\sigma}{\sqrt{N}}) \approx 2(1-\Phi(3)) = 0.0027, \tag{2.3}$$

where Φ is *c.d.f.* of $N(0,1)$.

Then the error and accuracy can be derived as

$$Error(N) := |\frac{1}{N}\sum_{i=1}^{N} X_i - \mu| \leq \frac{3\sigma}{\sqrt{N}} \text{ with probability close to 1} \tag{2.4}$$

$$\Rightarrow \text{Accuracy of MC} \sim O(\frac{1}{\sqrt{N}}) \tag{2.5}$$

3 Variance reduction

3.1 Basic idea and methods

Variance reduction is to search for alternative and more accurate estimators of a given quantity. It is typically needed in simulations of highly complex models with high computation cost per replication, and in simulations of rare events with too many replications in crude Monte Carlo.

Key idea 3.1

If we can find function g, such that

$$Eg(x) = Ef(x),\ Var\, g(x) \ll Var\, f(x) \tag{3.1}$$

Then $Ef(x) = Eg(x) \approx \frac{1}{N}\sum g(X_i), Error \sim \sqrt{\frac{Var(g(x))}{N}} \ll \sqrt{\frac{Var(f(x))}{N}}$

Classic variance reduction techniques include systematic sampling, importance sampling, control variates, and partial integration. Systematic sampling methods range from the simplest, antithetic variables, to the more sophisticated, stratified sampling. Importance sampling can be used in rare event sampling, and is the basis of reweighting and score function strategies for sensitivity analysis. Control variates is often used in pricing Asian options. Partial integration lowers variance by replacing integrals over some variables or over parts of space by their averages. Various variations and combinations of these methods have been developed, such as Latin hypercube sampling, weighted Monte Carlo, and moment matching. Here we will describe three of classic techniques.

3.2 Antithetic variables

Key idea 3.2

Assume $X \overset{d}{=} -X, cov(f(x), f(-x)) \leqslant 0$, *then*

$$Ef(x) = E\frac{f(x) + f(-x)}{2} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i) + f(-X_i)}{2} \tag{3.2}$$

Variance can be reduced by at least 50%.

Proof. Assume random variable X is symmetric, i.e. $X \overset{\mathrm{d}}{=} -X$, then

$$Ef(x) = Ef(-x) = E\frac{f(x) + f(-x)}{2} \tag{3.3}$$

$$\begin{aligned} Var(\frac{f(x) + f(-x)}{2}) &= \frac{1}{4}[Var(f(x)) + Var(f(-x)) + 2Cov(f(x), f(-x))] \\ &= \frac{1}{2}[Var(f(x)) + Cov(f(x), f(-x))] \end{aligned} \tag{3.4}$$

If $cov(f(x), f(-x)) \leqslant 0$, then variance is reduced by at least 50%. □

Lemma 3.3

If $f(x)$ *is monotone, then* $cov(f(x), f(-x)) \leqslant 0$.

Considering European call option in Black-Scholes model, $f(x) = S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}X} - K)^+$ is monotonically increasing, it makes sense to use ***Antithetic Variables***.

Advantage:

- It reduces the variance of the sample paths and improves the precision.
- It reduces number of simulations when achieving same accuracy and reduces computational cost.

Disadvantage:

- It only works well when $cov(f(x), f(-x)) \leqslant 0$.

3.3 Importance sampling

Consider European call option in Black-Scholes model. Let $S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}x} - K = 0$, we can get

$$x^* = \frac{ln\frac{K}{S_0} - (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} \tag{3.5}$$

If $x^* > 2$, then $p(f(x) > 0) = p(X > x^*) < 1 - \Phi(2) = 0.02$.

Thus for basic Monte Carlo, around 98% of simulations give 0 value, only 2% of simulations give new information. We need to find a method to achieve more non-zero values.

Key idea 3.4

Suppose $p(f(x) = 0) \approx 1$, *find* $g_\beta(y)$ *such that*

$$Ef(x) = Eg_\beta(y) \approx \frac{1}{N}\sum_{i=1}^{N} g_\beta(Y_i), \quad p(g_\beta(y) \neq 0) \text{ is large} \tag{3.6}$$

Back to European call option, $Ef(x) = E(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}X} - K)^+$.
Let $x = y - \beta$, $Y \sim N(0,1)$,

$$\begin{aligned} Ef(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-\frac{x^2}{2}} dx \overset{x=y-\beta}{=\!=\!=} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(y-\beta) e^{-\frac{(y-\beta)^2}{2}} dy \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(y-\beta) e^{\beta y - \frac{1}{2}\beta^2} e^{-\frac{y^2}{2}} dy \\ &= E\left[f(y-\beta) e^{\beta y - \frac{1}{2}\beta^2} \right] \\ &=: Eg_\beta(y),\ Y \sim N(0,1) \end{aligned} \tag{3.7}$$

Regarding to the choose of β,
On one hand,

$$g_\beta(Y) > 0 \Leftrightarrow f(Y-\beta) > 0 \Leftrightarrow Y - \beta > x^* \Leftrightarrow Y > x^* + \beta \tag{3.8}$$

we should choose β negative enough so that $p(g_\beta(y) > 0)$ can be large.
On the other hand,

$$Var(e^{\beta Y - \frac{1}{2}\beta^2}) = e^{\beta^2} - 1 \tag{3.9}$$

to improve the precision, $|\beta|$ should not be too large.
Typically, we take $\beta = -x^*$ so that around 50% of simulations will give non-zero values.

Advantage:

- It saves much time in computation and reduce computational cost.
- It increases the efficiency and precision by placing much of the probability mass in the rare event region. This is because when we want to obtain a rare event using crude Monte Carlo, we need a lot of samples and the variance is very high. Also, sometimes in practice we cannot have a very large sample size.

Disadvantage:

- It only works well when $p(f(x) = 0) \approx 1$.
- As system complexity increases, designing good biased distributions for importance sampling becomes more difficult.
- It can easily backfire when the sampling measure is not appropriately mimicking the target (e.g. it has too fast tail decay compared to the target, which will lead to wildly varying likelihood ratio weighting).

3.4 Control variates

Key idea 3.5

Suppose we know precisely $h^* = Eh(x)$, we can estimate $f(x)$ by

$$Ef(x) = Eg(x) \approx \frac{1}{N} \sum_{i=1}^{N} g(X_i), \quad g(x) = f(x) + \beta_{min}(h^* - h(x)) \tag{3.10}$$

$V(\beta) := Var\left[f(x) + \beta(h^* - h(x))\right] = Var(f(x)) - 2\beta Cov(f(x), h(x)) + \beta^2 Var(h(x))$,

Let $\beta_{min} = \frac{Cov(f(x),h(x))}{Var(h(x))}$, let ρ denote correlation of $f(x)$ and $h(x)$,

$$V(\beta)_{min} = V(\beta_{min}) = (1-\rho^2) Var(f(x)) = Var(g(x)) \tag{3.11}$$

If $|\rho| \approx 1$, then $Var(g(x)) \approx 0$, the variance is largely reduced.

Advantage:

- It saves a lot of computational cost, as we can estimate $h(x)$ with simulation number $M \ll N$.

Disadvantage:

- It only works well when $h(x)$ is designed well. A control variate must be cheap and effective enough for the increased effort to be compensated by precision gains.

4 Generating random paths

Some financial derivatives' payoff depends on terminal stock price S_T, so it is more appropriate to price their values by generating the trajectories of underlying stock prices. This section introduces how to generate paths $\{S_t, 0 \leq t \leq T\}$ rather than directly calculating S_T which is the terminal point of a path.

4.1 Basic idea

Key idea 4.1

*Use **Stochastic Differential Equation** to model the dynamics of the stock price with independent and normal distributed time increments:*

$$dY(t) \sim b(Y(t), t)dt + \sigma(Y(t), t)W(t)dt \tag{4.1}$$

The randomness is given by the white noise process $(W(t))_{t\geq 0}$.

In more rigorous cases in financial mathematics, ***Brownian motion*** $((B(t))_{t\geq 0})$ replaces the white noise process:

$$dY(t) \sim b(Y(t), t)dt + \sigma(Y(t), t)dB(t) \tag{4.2}$$

Here are the characterisations of the Brownian motion:

- $B(0) = 0$
- B has Gaussian increments: $B(t_2) - B(t_1) \sim N(0, (\sqrt{t_2 - t_1})^2)$
- B has independent increments: the future increments $B(t_4) - B(t_3)$ are independent of the past values $B(t_2) - B(t_1)$, if $t_1 \leq t_2 \leq t_3 \leq t_4$
- B has continuous paths: $t \to B(t)$. Although the path is continuous, $t \to B(t)$ is nowhere differentiable with a probability of 1.

The process is widely applied in pricing. For instance, in the Black-Scholes model, the stock price satisfies the SDE $dS(t) = rS(t)dt + \sigma S(t)dB(t)$.

4.2 Euler method

Key idea 4.2

*Applying **Gaussian increments** characteristic of Brownian motion, the equation (4.2) can be transformed into*

$$Y(t + \Delta t) \approx Y(t) + b(Y(t), t)\Delta t + \sigma(Y(t), t)\sqrt{\Delta t}\xi, \quad \xi \sim N(0, 1) \tag{4.3}$$

In this transformation, the randomness is given by ξ.

Then, generating a sample path of $(Y(t))_{0\leq t\leq T}$ using Euler method can be easily implemented by the steps:

- set time increment $\Delta t = \frac{T}{M}$ where $M \in \mathbb{Z}^+$

- generate M iid copies from the normal distribution with expectation of 0 and variance of 1, denoted as $\xi_i, i = 0, 1, ..., M-1$
- conduct the recursion: $Y^{m+1} := Y^m + b(Y^m, m\Delta t)\Delta t + \sigma(Y^m, m\Delta t)\sqrt{\Delta t}\xi_m$. The terminal point of each path $Y^m \approx Y(m\Delta t)$.

After the generation of paths, each terminal point represents the stock price from N simulations, which can be used to estimate the expectation of payoff.

4.3 Milstein method

Key idea 4.3

Considering the dynamics of $b(y,t)$ and $\sigma(y,t)$ (denoted as b and σ),

$$Y(t+\Delta t) \approx Y(t) + b(Y(t),t)\Delta t + \sigma(Y(t),t)\sqrt{\Delta t}\xi + \sigma(Y(t),t)\frac{\partial \sigma(Y(t),t)}{\partial y}\frac{\Delta t}{2}(\xi^2 - 1), \quad \xi \sim N(0,1) \tag{4.4}$$

*In the derived process of equation (4.4), we can use **Ito's formula**:*
Let $f(t,y) \in \mathbb{C}^{1,2}$, then

$$df(t,Y(t)) = \frac{\partial}{\partial t}f dt + \frac{\partial}{\partial y}f dY(t) + \frac{1}{2}\frac{\partial^2}{\partial y^2}f(dY(t))^2 = (\frac{\partial}{\partial t}f + b\frac{\partial}{\partial y}f + \frac{1}{2}\sigma^2\frac{\partial^2}{\partial y^2}f)dt + \sigma f_y dB(t) \tag{4.5}$$

To simplify, equation(4.4) can be denoted as $df(t,Y(t)) =: Lfdt + GfdBt$. Then the equation(4.3) can be transformed into

$$Y(t+\Delta t) = Y(t) + b(Y(t),t)\Delta t + \sigma(Y(t),t)(B(t+\Delta t) - B(t)) + I + II + III + IV \tag{4.6}$$

where $I := \int_t^{t+\Delta t}\int_t^s Lb(Y_u,u)\,duds \sim O((\Delta t)^2)$
$II := \int_t^{t+\Delta t}\int_t^s Gb(Y_u,u)\,dBuds \sim GbN(0,\frac{1}{3}(\Delta t)^3$
$III := \int_t^{t+\Delta t}\int_t^s L\sigma(Y_u,u)\,dudBs \sim L\sigma N(0,\frac{1}{3}(\Delta t)^3$
$IV := \int_t^{t+\Delta t}\int_t^s Gb(Y_u,u)\,dBudBs \overset{d}{=} G\sigma\frac{\Delta t}{2}(\xi^2-1)$, *where* $\xi \sim N(0,1)$
As IV dominates the other three, including this part into the equation(4.3) will bring a significant effect on reducing the error.

4.4 Efficiency

The error of Euler method consists of MC simulation error $\sim O(\frac{1}{\sqrt{N}})$ and dicretisation error $\sim O(\frac{1}{\sqrt{M}})$.

Milstein method has a lower discretisation error $\sim \frac{1}{M}$ than Euler method theoretically. However, these two methods do not differ too much when the option price is path-dependent and has a simple structure. In practice, there exists such examples where Milstein method performs better in path-dependent options.

5 Comparison with Finite Difference method

5.1 Basic idea of Finite Difference method

Let $\frac{dy(t)}{dt} = f(y(t),t))$ be a first-order ODE. That is to say, f is a function that returns the derivative, in other words, of a state given a time and state value. Let t be a numerical grid of the interval $[t_0, T]$ with spacing Δt.

We may approximate $\frac{dy(t)}{dt}$ by:

$$Forward\ difference: \frac{dy(t)}{dt} \approx \frac{y(t+\Delta t)-y(t)}{\Delta t} \Rightarrow y(t+\Delta t) = y(t) + f(y(t),t)\cdot \Delta t$$

$$Backward\ difference: \frac{dy(t)}{dt} \approx \frac{y(t)-y(t-\Delta t)}{\Delta t} \Rightarrow y(t) = y(t-\Delta t) + f(y(t),t)\cdot \Delta t$$

$$Central\ difference: \frac{dy(t)}{dt} \approx \frac{y(t+\Delta t)-y(t-\Delta t)}{2\Delta t} \Rightarrow y(t+\Delta t) = y(t)+\frac{\Delta t}{2}\big[f(y(t),t)+f(y(t+\Delta t),t+\Delta t)\big]$$

5.2 Application in Black-Scholes model

Consider a transformation from the Black-Scholes equation to a heat equation with $V(\tilde{t}, S)$.

$$x := ln\Big(\frac{S}{K}\Big), t = \frac{\sigma^2}{2}(T-\tilde{t}), q := \frac{2r}{\sigma^2}, u(t,x) := \frac{1}{K}V(T-\frac{2t}{\sigma^2}, Ke^x)\cdot e^{\frac{1}{2}(q-1)x+\frac{1}{4}(q+1)^2 t}$$

Then $u(t,x)$ satisfies the heat equation

$$u_t - u_{xx} = 0, t > 0, x \in \mathbb{R}$$

$$u(0,x) = (e^{\frac{q+1}{2}x} - e^{\frac{q-1}{2}x})^+$$

We can apply Explicit Euler Scheme to $u_t - u_{xx} = 0$ by discretisation. Define $\lambda := \frac{\Delta t}{(\Delta x)^2}$, we get

$$u(t+\Delta t, x) = \lambda u(t, x-\Delta x) + (1-2\lambda)u(t,x) + \lambda u(t, x+\Delta x)$$

We also need to set up the boundary value. From $u(t,x) := \frac{1}{K}V(T-\frac{2t}{\sigma^2}, Ke^x)e^{\frac{1}{2}(q-1)x+\frac{1}{4}(q+1)^2 t}$ and $x := ln\Big(\frac{S}{K}\Big)$, consider cases where S is vary small and S is quite large, we can get $u(t, x_{min}) = 0$, and $u(t, x_{max}) = \exp\big(\frac{q+1}{2}x_{max} + \frac{(q+1)^2}{4}t\big) - \exp\big(\frac{q-1}{2}x_{max} + \frac{(q-1)^2}{4}t\big)$. For numerical purpose, we may set $x_{min} = -3\sigma\sqrt{T}$ and $x_{max} = 3\sigma\sqrt{T}$.

Then the algorithm of Explicit Euler Scheme can be implemented. The essential idea is that the initial value $u(0,x)$ can be obtained from the heat equation, then $u(\Delta t, x)$ can be computed. Knowing $u(\Delta t, x)$, we can compute $u(2\Delta t, x)$. Repeating this procedure, we can arrive at $u(\frac{\sigma^2}{2}T, x)$. Then the price or prices can be obtained by $P(S_0) = V(0,S_0) = u(\frac{\sigma^2}{2}T, ln\frac{S_0}{K})K\cdot exp(-\frac{q-1}{2}ln(\frac{S_0}{K}) - \frac{(q+1)^2}{8}\sigma^2 T)$.

For Implicit Euler Scheme and Crank-Nicolson Scheme in Black-Scholes PDE, we can use the same idea of transformation and the same way to define λ. And for simplification, we use $u(m\Delta t, x_{min}+n\Delta t)$ to denote $u_{m,n}$ and treat them as a system of linear equations,

$$\vec{u_{m-1}} = \boldsymbol{A}\vec{u_m} + \vec{b_m} \Rightarrow \vec{u_m} = \boldsymbol{A}^{-1}(\vec{u_{m-1}} - \vec{b_m})\ \textit{for Implicit Euler Scheme}$$

$$\boldsymbol{A}\vec{u_{m+1}} - \vec{b_{m+1}} = \boldsymbol{B}\vec{u_m} + \vec{b_m}\ \textit{for Crank-Nicolson Scheme}$$

Now solving them is similar to solving $\boldsymbol{M}\vec{x} = \vec{b}$. Using the same idea of iteration that we implement in Explicit Euler Scheme, we can solve the system and then solve the Black-Scholes PDE by Implicit Euler Scheme and Crank-Nicolson Scheme.

Set $\Delta t \sim \frac{1}{M}$, $\Delta x \sim \frac{1}{N}$, $\lambda \sim \frac{\Delta t}{\Delta x^2}$, where M is the number of time steps, N is the number of grids for x, λ is related to stability. There are two error types in finite difference approach. One is round-off error which loses the precision because of computing rounding, the other one is truncation error which increases during the approximation procedure due to the "finite" number of time steps. We are interested in the latter and we can derive it by Taylor expansion. Comparison of these three schemes is showed in the table below.

Comparison:

Stability: Both Crank-Nicolson and Implicit Euler Scheme are always stable, because Trapezoidal Formula is also an implicit formula.

Error/Accuracy: For Crank-Nicolson Scheme, since we take the linear approximation of u_t and u_{xx} around t at $(t + \frac{\Delta t}{2})$, it's total error is $O(\Delta t^2) + O(\Delta x^2)$.

Number of computations is $O(MN)$ with number of time step M, and number of x's discretisation N, fixed in one dimension. Also, to optimise the computation, we need $\Delta t \sim \Delta x$ for Crank-Nicolson, which means $M \sim N$.

Number of Computations to achieve accuracy Δx^2 for Crank-Nicolson is $O(N^2)$, which is more efficient compared to Explicit and Implicit Euler Scheme.

Scheme	**Stability**	**Error/Accuracy**	**Number of Computations**	**Number of Computations to achieve accuracy Δx^2**
Explicit	*if* $\lambda \leq \frac{1}{2}$	$O(\Delta t) + O(\Delta x^2)$	$O(MN)$	$O(N^3)$
Implicit	*Always*	$O(\Delta t) + O(\Delta x^2)$	$O(MN)$	$O(N^3)$
Crank Nicolson	*Always*	$O(\Delta t^2) + O(\Delta x^2)$	$O(MN)$	$O(N^2)$

5.3 Monte Carlo V.S. Finite Difference

Since Crank-Nicolson performs best among three Finite Difference methods, Monte Carlo should compared with it. We consider the case where $Payoff = f(S_T^1, S_T^2, \cdots\cdots, S_T^d)$ in d-Dimension, and the goal is to achieve the accuracy $\sim O(\varepsilon)$. We set number of simulations N for Monte Carlo, and number of time steps $M \sim \frac{1}{\Delta t} \sim \frac{1}{\Delta x}$, number of grids for each time step $\sim \frac{1}{(\Delta x)^d}$ for Crank-Nicolson.

	Monte Carlo	**FD Crank-Nicolson**
Error	$\sim \frac{1}{\sqrt{N}}$	$\sim (\Delta x)^2$
Number of computations	$\sim O\Big(\frac{1}{\varepsilon^2}\Big)$	$\sim O\Big(\frac{1}{\varepsilon^{\frac{d+1}{2}}}\Big)$
Better in Dimension	$d > 3$	$d \leq 2$
Path dep. Option	Yes	No
Output(s)	$V(0, S_T^\star)$	$\big(V(t,S)\big)_{t,S}$

The advantage of Monte Carlo is, overall, it works better in sample space of terminal stock price, which the space is bigger than dimension 3. Also, in terms of the usage for pricing different types of option, Monte Carlo can also calculate the path dependent option, which is generally the exotic option. While Finite Difference methods need a well defined PDE for pricing the exotic options. However, the disadvantage of Monte Carlo is that we only obtain option price for a particular initial stock price, while finite difference method works better if we are interested in different initial stock prices.

6 Numerical Example

6.1 Asian Call Option Pricing Example

Consider an Asian Call Option with $r = 0.03$, $\sigma = 0.3$, $S_0 = 100$, $K = 100$, $T = 5$, $N = 10^5$, $N_t = T * 252$, $dt = \frac{T}{N_t} = \frac{1}{252}$, where N_t is number of trading days until maturity.

Firstly, we use ***Euler method*** and ***Milstein method*** of generating random paths to estimate the basic option price. We set dt as the time increment. After generating N_t *iid* copies of $N(0,1)$, the terminal stock price of each path can be computed by a recursion process.

Then we use ***Control Variates*** to improve our estimation.

Regarding to control variates, we use geometric Asian Call option price as $h(x)$ to estimate the arithmetic Asian Call option price. We set the simulation number about $h(x)$ be 10^3, simulation number for control variates method be 10^4. Then we can compute β_{min} and h^*.

A discrete arithmetic Asian call option has the payoff: $(\frac{1}{N+1}\sum_{i=0}^{N} S_{\frac{T_i}{N}} - K)^+$

A discrete geometric Asian call option has the payoff: $([\prod_{i=0}^{N} S_{\frac{T_i}{N}}]^{\frac{1}{N+1}} - K)^+$

In the Black-Scholes model, the price of the geometric Asian call option is given by

$$e^{-rT}(S_0 e^{\rho T} N(d_1) - K N(d_2)),$$

$$\text{where} \qquad \rho := \frac{1}{2}(r - \frac{1}{2}\sigma^2 + \hat{\sigma}^2), \qquad \hat{\sigma} := \sigma\sqrt{\frac{2N+1}{6(N+1)}}$$

$$d_1 := \frac{1}{\sqrt{T}\hat{\sigma}}(ln(\frac{S_0}{K}) + (\rho + \frac{1}{2}\hat{\sigma}^2)T), \qquad d_2 := \frac{1}{\sqrt{T}\hat{\sigma}}(ln(\frac{S_0}{K}) + (\rho - \frac{1}{2}\hat{\sigma}^2)T)$$

Also, we try to use ***Antithetic Variables*** to do variance reduction.

The table below shows our results of output price and standard error. We can see that the variance is effectively reduced, and Control Variates works better than Antithetic Variables.

Method	Estimated price	Standard Error
Euler Method	17.3958	0.1124
Milstein Method	17.5205	0.1126
Antithetic variables	17.4139	0.0650
Control Variates	17.4991	0.0331

6.2 European Call Option Pricing Example

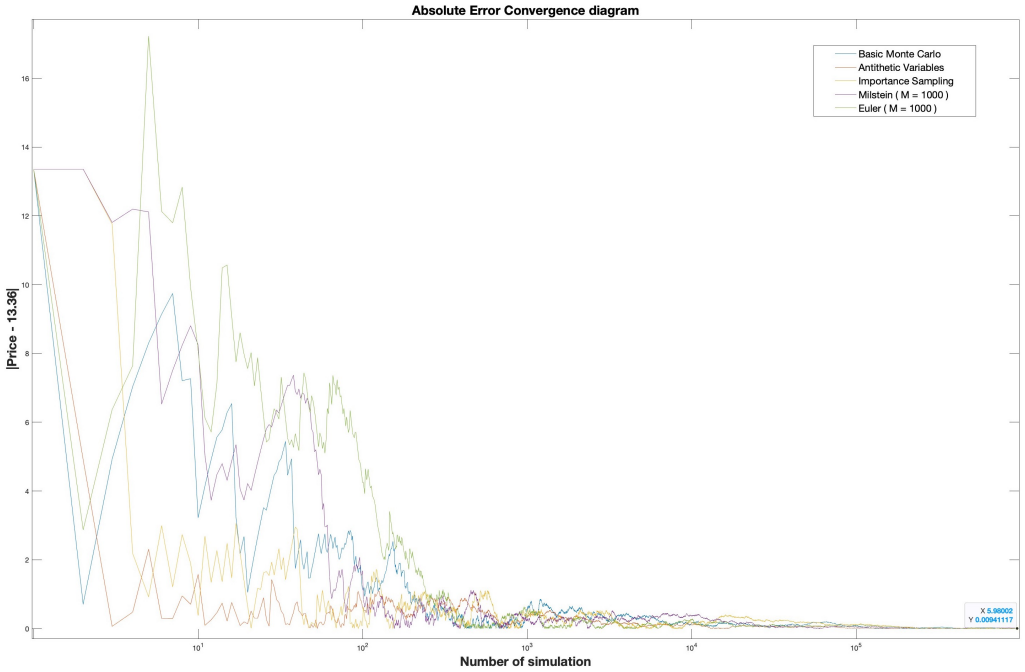
Consider the European call option example in tutorial 2: $S_0 = 100$, $r = 0.07$, $\sigma = 0.25$, $K = 100$, $T = 1$ (the option price by analytic computation is about 13.36).

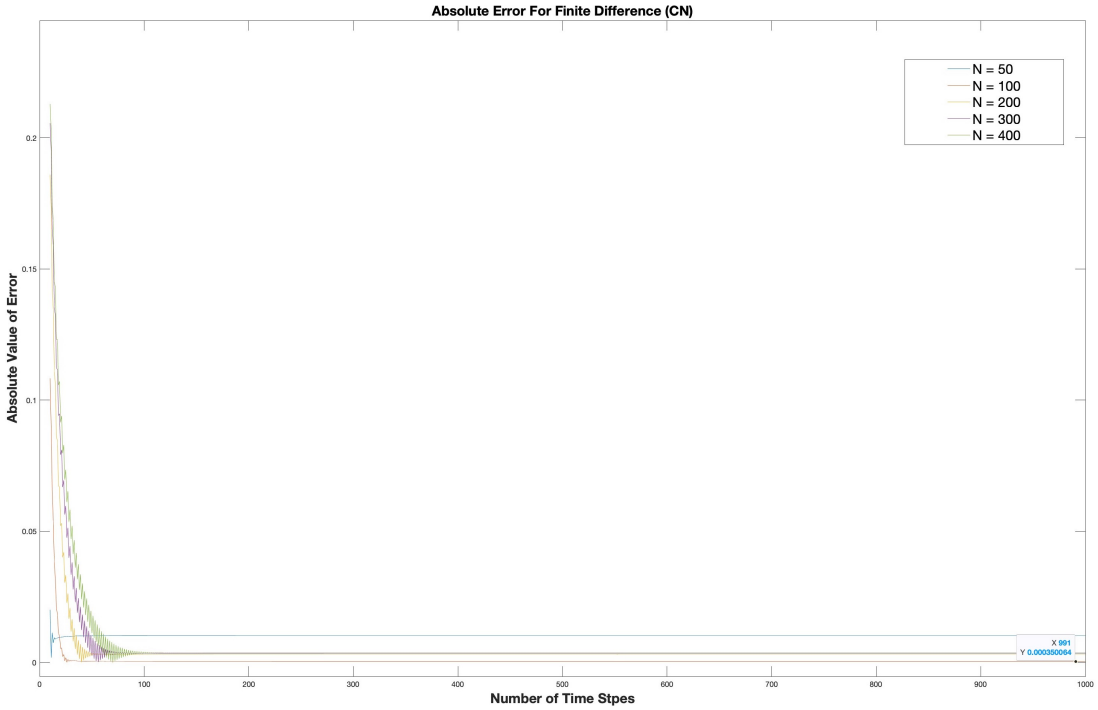
Regarding to this pricing problem, we solve it with Monte Carlo methods, including basic Monte Carlo, Antithetic Variable and Importance Sampling in variance reduction, Milstein and Euler in random path methods. Then we compare their accuracy and number of computations, as well as the accuracy with Crank Nicolson scheme in Finite Difference method.

Below is the absolute error convergence diagram for Monte Carlo methods. We set y-axis to be the absolute value of the difference between our computing outputs and 13.36 so that it clearly illustrates the comparison between the performance of methods. In the first 10 simulations, Antithetic Variables reduces the errors within 2 very fast. To achieve the same accuracy, Milstein and Euler need about 100 simulations of random paths. After 10^3 simulations, errors of most methods converge well to 0, but Euler method needs 10^5 times. What's more, Antithetic Variables works better than Importance Sampling, because our example doesn't satisfy the assumption of Importance Sampling.

Regarding to Finite Difference method, since Crank-Nicolson performs best, we use it for illustration. From the graph, Crank-Nicolson with 100 grids for each time step works best for this problem. For all Ns in the graph, the errors get stable when the total number of time steps is larger than 100.

Overall, from the two charts, finite difference method is more accurate and converges faster than Monte Carlo method when pricing European option.


Absolute Error Convergence diagram
Basic Monte Carlo
Antithetic Variables
Importance Sampling
Milstein (M = 1000)
Euler (M = 1000)
|Price - 13.36|
Number of simulation
X 5.98002
Y 0.00941117


Absolute Error For Finite Difference (CN)
N = 50
N = 100
N = 200
N = 300
N = 400
Absolute Value of Error
Number of Time Stpes
X 991
Y 0.000350064