THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MATHEMATICS

INTEREST RATE MODEL AND OPTION PRICING

XINYU WANG
SPRING 2019

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Mathematics and Statistics
with honors in Area of Mathematics

Reviewed and approved* by the following:

Anna Mazzucato
Professor of Mathematics
Thesis Supervisor

Ludmil Zikatanov
Professor of Mathematics
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

In the 2008 Financial Crisis, one of the primary sources causes this disaster is Mortgage-Backed Security(MBS), which is one of the derivatives among thousands. People who are unfamiliar with finance would ask, what is derivatives? Derivative is a contract between two or more parties whose value is depended on an agreed-upon underlying financial products or set of assets. Financial institution developed derivatives aiming to creates tools for investors to hedge risks, but at the same time, hedging depends on accurately assuming risk. If financial institutions and investors do not know how to use it in a right way, it may lead to catastrophe. Therefore, how to price options become an essential problem.

There are two major underlying financial assets trading in the market, which are bonds and stocks. The bond market size is about $40.7 trillion and the stock market size is about $30 trillion in U.S., which means the stock options and bond options are major derivatives in the market to hedge risks. This paper focus exactly on these two options, which includes a view of the approach to the models and different methods to solve those models.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my advisor Professor Anna Mazzucato for everything: her help and guidance through my thesis work and graduate application; her encouragement for me to move further study on higher degree; her teaching enthusiastic on Math411. Having been the one to introduce me to the world of Financial Mathematics, I am grateful that I was able to share this experience with her. Her patience and kindness were extremely important in supporting me to complete this major task in my life.

I would also like to acknowledge my honors advisor Professor Sergei Tabachnikov and Professor Ludmil Zikatanov. Both of them are important people who have helped me in my success at Penn State and I thank them each for that.

Lastly, I would like to thank those people that have been with me through my thesis work and my college career. To my wonderful parents, and extended family which is far too extensive to list here, thank you for supporting me to study in such an amazing university, both financially and mentally, thank you for encouraging me to strive to do my best and never give up. I can not thank you enough for everything you have provided for me with over the years, but I hope this can be a small start to my large thank you. To my friends from hometown and ones who I have made over the years in Penn State, thank you for your help and support on my academic achievement. Without you, I would not be who I am today.

Again, thank you to all those mentioned above and any that may have been missed. I feel lucky to have such a support group as extensive as it is.

1 | Introduction to Option Pricing

In order to use mathematics in option pricing, it is necessary to start by specifying some background knowledge in financial mathematics.

1.1 What is option

Definition: An option is a contract which gives investors the right to buy or sell financial underlying in the future with previous determined price. Whether investor executes the contract is decided by the value of the underlying asset price such as stocks or bonds.

Option contract are classified with several characteristics including:

- possible execution times (a fixed data vs a time interval),
- the number of underlying assets,
- how the value of option depends on the asset prices (depending on the price at the execution time vs a path dependent value of the asset prices).

Brownian motion

Robert Brown first observed microscopic particles moving under buffeting of a gas, the mathematical model then was developed to describe their movements.

Definition of Brownian motion [1]:

- W_t is continuous, and W_0 = 0,
- the value of W_t is distributed, under P, as a normal random variable N(0,t),
- the increment W_{s+t} -W_s is distributed as a normal N(0,t), under P, and is independent of $\mathcal{F}$, the history of what the process did up to time s.

Brownian motion vs stock market

The behavior of the log of the market is very similar to Brownian motion, but we have misgivings about this model since Brownian motion wanders around zero, whereas the stock of a company normally grows at certain rate-there is an increasing trend if we look at historically prices for stocks and bond because of inflation. Therefore, the process Brownian motion with drift can reflect the log of this growth: $\log(S_t)$ = W_t+μt, for some constant μ.

Also, stock market only volatilize with certain degrees, it can not with too much noise, then the general form to reflect the movement in stock market: $\log(S_t)$ = σW_t+μt, for some constant μ.

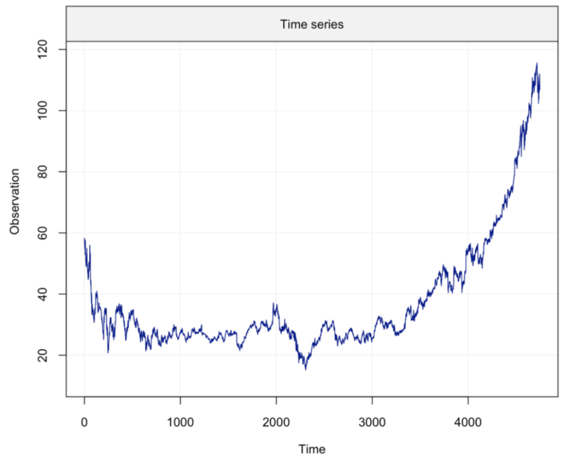

Figure 1.1: Microsoft stock price from 2000 to 2018. We can see from time 0 to 3000, the stock price centered around 30. But after time 3000, there is an obvious increasing trend, we need to use the drift to capture this feature.

1.2 Stochastic calculus

If we zoom in on Brownian motion plot, we could find that it is nowhere differentiable, which means we could not use traditional differential method. So here we introduce stochastic calculus.

Stochastic processes

A stochastic process X is a continuous process $(X_t : t \geqslant 0)$ such that X_t can be written as: [1]

$$X_t = X_0 + \int_0^t \sigma_s \mathrm{d}W_s + \int_0^t \mu_s d_s \ ,$$

where σ and μ are random $\mathcal{F}$-previsible processes. The differential form of this equation can be written:

$$\mathrm{d}X_t = \sigma_t \mathrm{d}W_t + \mu_t \mathrm{dt}.$$

In this special case, the σ_t and μ_t are some deterministic function which depends on time t. But σ and μ can be different types in many cases. For example, in Black Scholes Model, σ and μ are constant; in Interest Rate Model, σ and μ are depending on t for each T, which are $\sigma(t, T)$ and $\mu(t, T)$, where T is maturity.
However, if we consider more complex Stochastic Differential Equation (SDE), such as

$$\mathrm{d}X_t = X_t(\sigma \mathrm{d}W_t + \mu \mathrm{dt}).$$

We can not apply classic technique to solve for X_t, which brings us to Itô calculus.

Itô calculus

Consider a Taylor expansion of $f(W_t)$ for some smooth f in order to obtain $\frac{df(W_t)}{dt}$:

$$\mathrm{df}(W_t) = f'(W_t)\mathrm{d}W_t + 1/2 f''(W_t)(dW_t)^2 + 1/6 f'''(W_t)(dW_t)^3 + \ldots$$

In normal case, we assumed that $(dW_t)^2$ and higher terms to be zero. But in here, Brownian motion is different, we have $(dW_t)^2 = dt$. Therefore, we can not ignore $(dW_t)^2$ as $(dW_t)^2 \sim dt$. Finally Taylor gives us:

$$\mathrm{df}(W_t) = f'(W_t)\mathrm{d}W_t + 1/2 f''(W_t)\mathrm{dt}.$$

Itô formula

If X is a stochastic process, satisfying $\mathrm{d}X_t = \sigma_t \mathrm{d}W_t + \mu_t \mathrm{dt}$, and f is a deterministic twice continuously differentiable function, the Y_t = f(X_t) is also a stochastic process and is given by: [1]

$$\mathrm{d}Y_t = (\sigma_t f'(X_t))dW_t + (\mu_t f'(X_t) + 1/2 {\sigma_t}^2 f''(X_t))\mathrm{dt}.$$

to apply this formula, then we have Itô lemma:

$$dW_t = \frac{\partial f}{\partial W_t} dW_t + \left(\frac{\partial f}{\partial t} + 1/2 \frac{\partial^2 f}{\partial W_t^2}\right) dt.$$

SDEs from processes

It is an important application of Itô calculus to generate SDEs from functional expression for a process. Now, consider the exponential Brownian motion, we are interested in about the this exponential type since the stock market movement actually follows the exponential Brownian motion since the price can not go down below zero.

$$X_t = \exp(\sigma W_t + \mu \mathrm{t}).$$

In here, we took $Y_t = \sigma W_t + \mu$t, and f to be the exponential function f(x) = e^x, notice, the derivative of exponential function is itself, so we could apply Itô formula gives us: [1]

$$dW_t = \sigma f'(Y_t)dW_t + (\mu f'(Y_t) + 1/2\sigma^2 f''(Y_t))\mathrm{dt},$$

$$dW_t = X_t(\sigma dW_t + (\mu + 1/2\sigma^2)\mathrm{dt}).$$

We will use this formula repeatedly in later content, it is a very powerful tool to help us deal with SDEs.

1.3 Martingale representation theorem

A martingale measure is one which makes the expected future value is equal to its present value conditional on available information, see [2]

Martingales

A stochastic process M_t is a martingale with respect to a measure P if and only if:

- $E_P(|M_t|) < \infty$, for all t,
- $E_P(M_t|F_s) = M_s$, for all s $\leqslant$ t.

If X is a stochastic process driven by W_t with volatility σ_t, which satisfied the technical condition E[$(\int_0^T {\sigma_s}^2 ds)^{0.5}$] < ∞, then:

X is a martingale $\Longleftrightarrow$ X is driftless.

The martingale property allow to apply replicating strategy and construct Black Scholes Model in later chapter.

1.4 Construction strategies

Now we need to hook those mathematical tool into a financial model, in Black Scholes Model for example, we have a portfolio which consists of a stock and a riskless cash bond.

The portfolio (ϕ, φ)

A potfolio is a pair of processes ϕ_t and φ_t which represents respectively the number of units of stock and of the bond which we hold at t. The stock component of the portfolio ϕ should be $\mathcal{F}$ previsible: depending only on information up to time t but not t itself.

Self-financing property

A portfolio is self-financing if and only if the change in its value only depends on the change of the asset prices. It has something to do with SDE.
Now, we set up the stock price is S_t and bond price is B_t, the value of portfolio is V_t at time t, then the total value of this portfolio is given by V_t = $\phi_t S_t$ + φB_t. At next time increment, two things would happen: the old portfollio changes value because S_t and B_t changed price; and portfolio itself need to be adjusted to give a new portfolio as instructed by the trading strategy (ϕ, φ). If the cost of the adjustment is perfectly matched by the profits or losses made by the portfolio, then

no need extra money from outside. If we transfer this information into mathematics formula, we could get: [1]

$$(\phi_t, \varphi_t) \text{ is self-financing} \iff dV_t = \phi_t dS_t + \varphi_t dB_t.$$

Replicating strategy

Suppose we are in market of a stock and a riskless bond with volatility σ_t, and a claim X on events up to time T. A replicating strategy for X is a self-financing portfolio (ϕ, φ) such that V_T = $\phi_T S_T$ + $\varphi_T B_T$ = X, which means the claim value of some derivative which we need to pay off at time T is equal to the portfolio value itself, this important formula will provide us an ideal to develop Black-Scholes Model.

Three steps to replication, see [1]

- Find a measure Q under which S_t is a martingale,
- Form the process E_t = E_Q(X|$\mathcal{F}$),
- Find a previsible process ϕ_t, such that dE_t = $\phi_t dS_t$.

The tools described earlier are essential to do this, We will use Cameron-Martin-Girsano theorem for the first step and the martingale representation theorem for the third one. We will see how does those work in Black Scholes Model in next chapter.

2 | Black Scholes Model

The Black Scholes Model is a fundamental, but simplified model for option pricing. It is a good one to begin. The asset we considered is a stock which could be held without additional cost or benefit and was freely tradable at the price quoted. and here, we discussed is vanilla European option, also the simplest one, the maturity date is fixed. [3]

2.1 Black Scholes Model

We will assume the existence of a deterministic r and σ, where r is the riskless interest rate and σ is the stock volatility, such that the bond price B_t and stock price S_t follow:

$$B_t = \exp(\mathrm{rt}),$$

$$S_t = S_0 \exp(\sigma W_t + \mathrm{rt}).$$

we take the log of S_t and find its derivative:

$$log(S_t) = logS_0 + (\sigma W_t + \mathrm{rt}),$$

$$d(log(S_t)) = \sigma dW_t + \mathrm{rdt},$$

$$dS_t = \sigma S_t dW_t + \mathrm{r} S_t dt. \tag{2.1}$$

We know V_t is a function with respect to $logS_t$ and time t, then

$$V_t = f\ (logS_t, \mathrm{t}).$$

We treat $logS_t$ as X and find its derivative using Itô formula:

$$dV_t = \frac{\partial V}{\partial X} dX_t + \frac{\partial V}{\partial t} dt,$$

$$dV_t = \frac{\partial V}{\partial X}(_t) + \frac{\partial V}{\partial X} rdt + 1/2 \frac{\partial^2 V}{\partial X^2} \sigma^2 dt + \frac{\partial V}{\partial t} dt.$$

We need to change back to S, where X = logS

$$dV_t = s\frac{\partial V}{\partial S}(_t) + s\frac{\partial V}{\partial S}rdt + 1/2\frac{\partial^2 V}{\partial S^2}\sigma^2 s^2 dt + \frac{\partial V}{\partial t}dt. \tag{2.2}$$

Replicating strategy

- hold ϕ_t units of stock at time t, and
- hold φ_t = V_t - $\phi_t S_t$ units of the bond at time t.

This holding strategy will make this portfolio self-financing, thus we will have:

$$dV_t = \phi_t dS_t + \varphi_t dB_t. \tag{2.3}$$

We know $dB_t = rB_t dt$, since it is just normal derivative. Now we plug (2.1) into (2.3), we will get:

$$dV_t = \phi_t(\sigma \mathrm{s} dW_t + rsdt) + \varphi_t rB_t dt, \tag{2.4}$$

$$dV_t = \phi_t \sigma \mathrm{s} dW_t + (\varphi_t rB_t + \phi_t rs)dt. \tag{2.5}$$

Now, we could match dW_t term and dt term between equation (2.2) and (2.5), we will get following:

$$\phi_t = \frac{\partial V}{\partial S}, \tag{2.6}$$

$$\varphi_t rB_t + \phi_t rs = \mathrm{s}\frac{\partial V}{\partial S}r + 1/2\frac{\partial^2 V}{\partial S^2}\sigma^2 s^2 + \frac{\partial V}{\partial t}, \tag{2.7}$$

$$\varphi_t rB_t = 1/2\frac{\partial^2 V}{\partial S^2}\sigma^2 s^2 + \frac{\partial V}{\partial t}. \tag{2.8}$$

Remember, at the very beginning, we have the portfolio value formula, which is: $V_t = \phi_t S_t + \varphi_t B_t$. If we multiply both sides by interest rate r, we will get following:

$$\phi_t S_t r + \varphi_t B_t r - rV_t = 0. \tag{2.9}$$

Then we plug (2.6) and (2.8) inside (2.9), we will get:

$$\frac{\partial V}{\partial S}sr + 1/2\frac{\partial^2 V}{\partial S^2}\sigma^2 s^2 + \frac{\partial V}{\partial t} - Vr = 0. \tag{2.10}$$

This Partial Differential Equation (PDE) we get is Black Scholes Model. This equation can be transformed into heat equation through change of variable, so we will have exact solution for Black Schole Model. Next, we will talk about how to find its exact solution.

2.2 Exact solution

As we mentioned in the above content, we could transfer the (2.10) to a heat equation, which can be solved explicitly, then we change variables back to Black Scholes Model.

Step 1

We let x = $ln(S/E)$, τ = $T-t$, and a new function $Z(x,\tau)$ = $V(Ee^{e}, T-\tau)$. Then we derived a new equation.

$$\frac{\partial Z}{\partial \tau} - 1/2\frac{\partial^2 Z}{\partial x^2}\sigma^2 + \frac{\partial Z}{\partial x}(\frac{\sigma^2}{2} - r) + rZ = 0. \tag{2.11}$$

Step 2

New equation $u(x,\tau)$ = $e^{\alpha x+\beta\tau}Z(x,\tau)$, where the constants α,β are chosen so that the PDE of u is the heat equation. We get PDE for u is:

$$\frac{\partial u}{\partial \tau} - 1/2\frac{\partial^2 u}{\partial x^2}\sigma^2 + A\frac{\partial u}{\partial x}(\frac{\sigma^2}{2} - r) + Bu = 0, \tag{2.12}$$

where

$$A = \alpha\sigma^2 + \sigma^2/2 - r,\ B = (1+\alpha)r - \beta - (\alpha^2\sigma^2 + \alpha\sigma^2)/2.$$

We have to set $A = B = 0$, then we could get α, β:

$$\alpha = \frac{r}{\sigma^2} - \frac{1}{2},\ \beta = \frac{r}{2} + \frac{\sigma^2}{8} + \frac{r^2}{2\sigma^2}.$$

Then we will get heat equation for $u(x,\tau)$:

$$\frac{\partial u}{\partial \tau} - \frac{\sigma^2}{2}\frac{\partial^2 u}{\partial x^2} = 0. \tag{2.13}$$

Step 3

The solution $u(x,\tau)$ of the heat equation is given by the general formula:

$$u(x,\tau) = \frac{1}{\sqrt{2\sigma^2\pi\tau}}\int_{-\infty}^{\infty} e^{-\frac{(x-s)^2}{2\sigma^2\tau}}u(s,0)ds.$$

Notice, the initial condition $u(x,0)$ also changed to:

$$u(x,0) = e^{\alpha x}V(Ee^{x},T) = e^{\alpha x}, \text{ if } x > 0,$$

$$u(x,0) = 0, \text{ otherwise.}$$

The solution $u(x,\tau)$ is:

$$u(x,\tau) = \frac{1}{\sqrt{2\pi\sigma^2\tau}}\int_{0}^{\infty} e^{-\frac{(x-s)^2}{2\sigma^2\tau}}e^{\alpha s}ds. \tag{2.14}$$

Then, we evaluate the integral and perform backward substitutions $u(x,\tau) \longrightarrow Z(x,\tau) \longrightarrow V(s,T)$, we will get our exact solution for Black Scholes Model:

$$V(s,T) = s\Phi(\frac{log\frac{s}{k}+(r+1/2\sigma^2)T}{\sigma\sqrt{T}}) - ke^{-rT}\Phi(\frac{log\frac{s}{k}+(r-1/2\sigma^2)T}{\sigma\sqrt{T}}), \tag{2.15}$$

where

$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{\frac{-y^2}{2}}dy.$

People usually prefer to write it as:

$$V(s,T) = s\Phi(d_1) - ke^{-rT}\Phi(d_2), \tag{2.16}$$

where $d_1 = \frac{log\frac{s}{k}+(r+1/2\sigma^2)T}{\sigma\sqrt{T}}$, and $d_2 = \frac{log\frac{s}{k}+(r-1/2\sigma^2)T}{\sigma\sqrt{T}}$.

Plot from Python

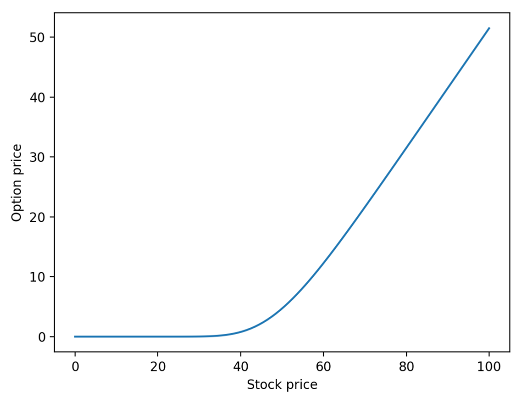

Figure 2.1: Stock price vs Option price by exact solution.We choose r = 0.03, $\sigma = 0.2$ and $T = 1$.

2.3 Monte Carlo method to compute European options price

Monte Carlo Option Price is a method often used in Mathematical finance to calculate the value of an option with multiple sources of uncertainties and random features, such as changing stock prices. After repeatedly simulating stock prices, it is possible to obtain estimates of the price of a European call option. A statistical simulation algorithm of this type is what we known as **"Monte Carlo method"**.

We choose Euler' formula to generate the path, which gives us following to calculate the stock price [4]

$$S_{n+1} = S_n + \mu S_n dt + \sigma S_n \varepsilon_{n+1}\sqrt{dt}. \tag{2.17}$$

where we have initial condition $S_0 = s, S_n$ is the predicted stock price at N steps, and $dt = \frac{T}{N}$, μ is the annual growth rate of the stock, and σ is the stock volatility. Each term in sequence ε takes on the value of -1 or 1 with same probability $\frac{1}{2}$, in other words, for each n = 1,2, . . .

$$\varepsilon_n = \begin{cases} 1 & \text{with probability } \frac{1}{2}, \\ -1 & \text{with probability } \frac{1}{2}. \end{cases} \tag{2.18}$$

From above steps, we only calculated one path, but we need to generate more paths, and calculate the average to be accurate. We use two dimensions Euler's formula to do this:

$$S_{n+1}^k = S_n^k + rS_n^k dt + \sigma S_n^k \varepsilon_n \sqrt{dt}. \tag{2.19}$$

(2.19) is identical to (2.17) for each k = 1,. . ., M, except the growth rate μ is replaced by interest rate r.

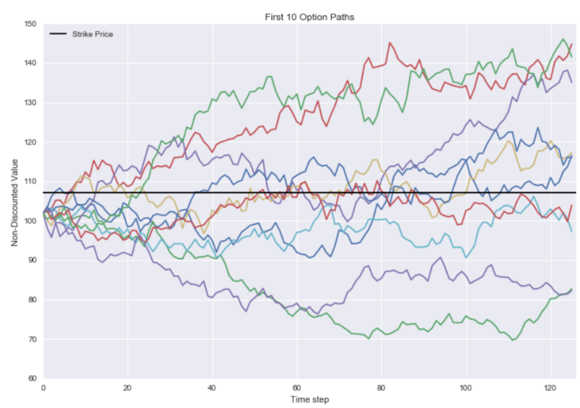

Figure 2.2: Stock price path generated by Monte Carlo, each color represents a path of stock price.

A European call option is a contract between two parties, a holder and a writer, whereby, for a premium paid to the writer, the holder can purchase the stock at a future data T (the expiration data) at a price K (the strike price) agreed upon in the contract. If the buyer elect to exercise the option on the expiration data, the writer is obligated to sell the underlying stock to the buyer at the price K, the strike price. Thus, the option has a payoff function:

$$f(S) = max(S - K, 0). \tag{2.20}$$

Option pricing requires that the average value of the payoffs equal to the compounded total return obtained by investing the option premium, we use V represents the price of option:

$$V = e^{-rT} \frac{1}{M} \sum_{k=1}^{M} f(S_N^k). \tag{2.21}$$

Example 1

Now, we use Python to implement. For this example, we choose the parameters as below: $r = 0.03, \sigma = 0.2, N = 200, K = 50, T = 1$, we adjust M.

Result from python

MCE(100)	**MCE(1000)**	**MCE(10000)**	**Eaxct solution**
$V(40) = 0.499$	$V(40) = 0.779$	$V(40) = 0.765$	$V(40) = 0.781$
$V(45) = 2.575$	$V(45) = 2.223$	$V(45) = 2.263$	$V(45) = 2.224$
$V(50) = 5.920$	$V(50) = 4.646$	$V(50) = 4.785$	$V(50) = 4.707$

Table 2.1: Comparison between MCE and exact solution for BSM. We try the different M from M=100, M=1000 to M=10000, see the change of accuracy.

Absolute error comparison

	MCE(100)	**MCE(1000)**	**MCE(10000)**
$S = 40$	$Err = 0.282$	$Err = 0.002$	$Err = 0.0016$
$S = 45$	$Err = 0.351$	$Err = 0.001$	$Err = 0.0390$
$S = 50$	$Err = 1.213$	$Err = 0.061$	$Err = 0.0780$

Table 2.2: We calculate the difference between MCE calculation and exact solution, then take the absolute value, we want to see how much digit of accuracy that MCE could achieve.

Relative error comparison

	MCE(100)	**MCE(1000)**	**MCE(10000)**
$S = 40$	$Err = 36.108\%$	$Err = 0.256\%$	$Err = 0.205\%$
$S = 45$	$Err = 15.782\%$	$Err = 0.045\%$	$Err = 1.756\%$
$S = 50$	$Err = 25.770\%$	$Err = 1.296\%$	$Err = 1.657\%$

Table 2.3: We use the formula: $Err = \frac{abs(MCE-exact)}{exact}$ to compute the relative error, to see how much percent is away the exact solution.

Plot from Python of MCE

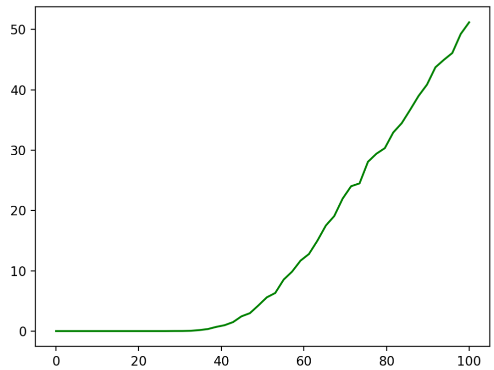

Figure 2.3: Stock option price plot by MCE. We choose $M = 1000, N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = [0, 100]$

Plot from Python for comparison

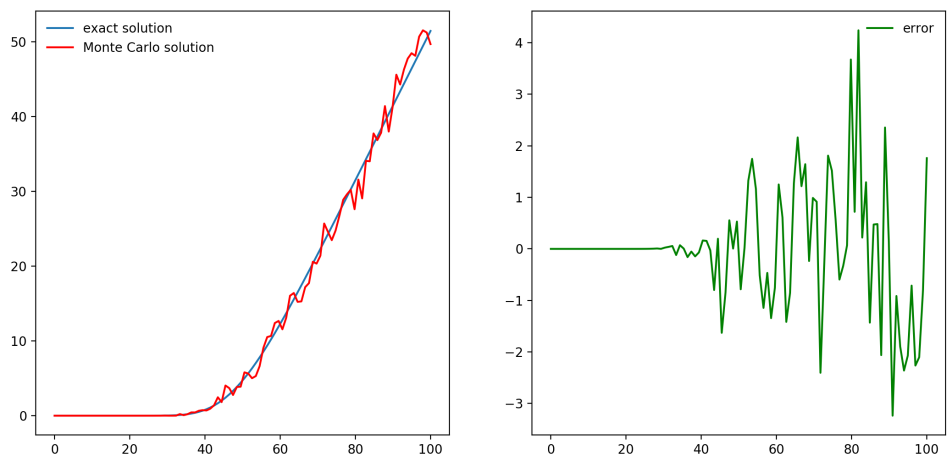

Figure 2.4: Plotted stock option price between MCE with M=100 and exact solution. Also right plot is the error function between them. We choose $M = 100, N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = [0, 100]$.

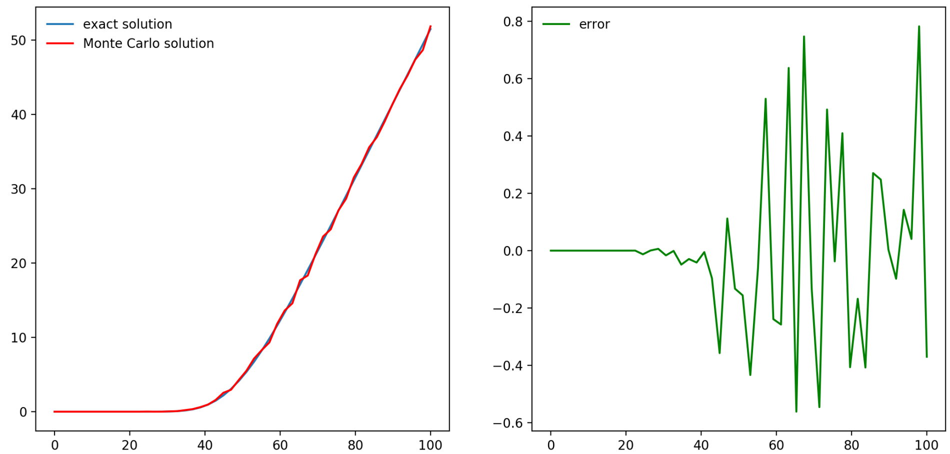

Figure 2.5: Plotted stock option price between MCE with M=1000 and exact solution. Also right plot is the error function between them. We choose $M = 1000, N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = [0, 100]$

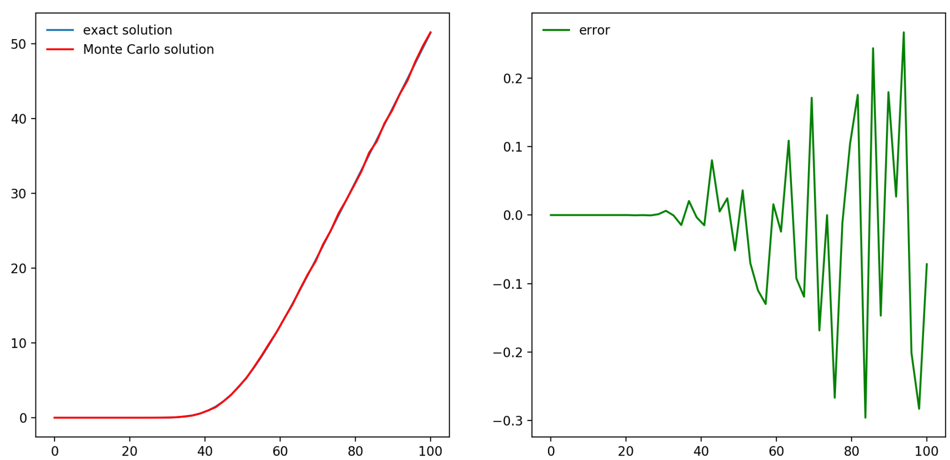

Figure 2.6: Plotted stock option price between MCE with M=10000 and exact solution. Also right plot is the error function between them. We choose $M = 10000, N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = [0, 100]$

Example 2

For this example, we choose the parameters as below: $r = 0.03, \sigma = 0.5, N = 200, K = 50, T = 1$. Except for σ, all the others keep the same, we want to see how σ could influence the option price.

Result from python

MCE(100)	MCE(1000)	MCE(10000)	Eaxct solution
$V(40) = 6.589$ $V(45) = 7.609$ $V(50) = 10.332$	$V(40) = 5.489$ $V(45) = 8.073$ $V(50) = 10.612$	$V(40) = 5.075$ $V(45) = 7.485$ $V(50) = 10.518$	$V(40) = 5.104$ $V(45) = 7.573$ $V(50) = 10.481$

Table 2.4: Comparison between MCE and exact solution for BSM. We try the different M from M=100, M=1000 to M=10000, see the change of accuracy.

Absolute error comparison

	MCE(100)	MCE(1000)	MCE(10000)
$S = 40$	$Err = 1.485$	$Err = 0.385$	$Err = 0.029$
$S = 45$	$Err = 0.036$	$Err = 0.5$	$Err = 0.088$
$S = 50$	$Err = 0.149$	$Err = 0.131$	$Err = 0.037$

Table 2.5: We calculate the difference between MCE calculation and exact solution, then take the absolute value, we want to see how much digit of accuracy that MCE could achieve.

Relative error comparison

	MCE(100)	MCE(1000)	MCE(10000)
$S = 40$	$Err = 29.10\%$	$Err = 7.54\%$	$Err = 0.57\%$
$S = 45$	$Err = 0.48\%$	$Err = 6.60\%$	$Err = 1.16\%$
$S = 50$	$Err = 1.42\%$	$Err = 1.25\%$	$Err = 0.35\%$

Table 2.6: We use the formula: $Err = \frac{abs(MCE - exact)}{exact}$ to compute the relative error, to see how much percent is away the exact solution.

Plot from Python of MCE

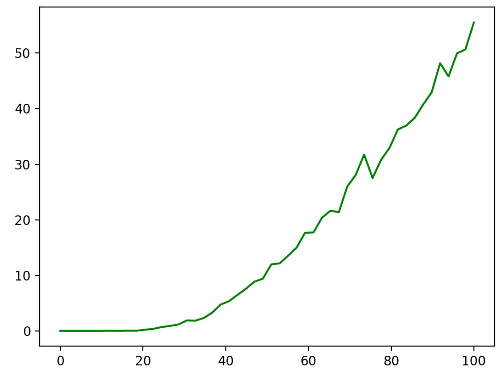

Figure 2.7: Stock option price plot by MCE. We choose $M = 1000, N = 200, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = [0, 100]$

Plot from Python for comparison

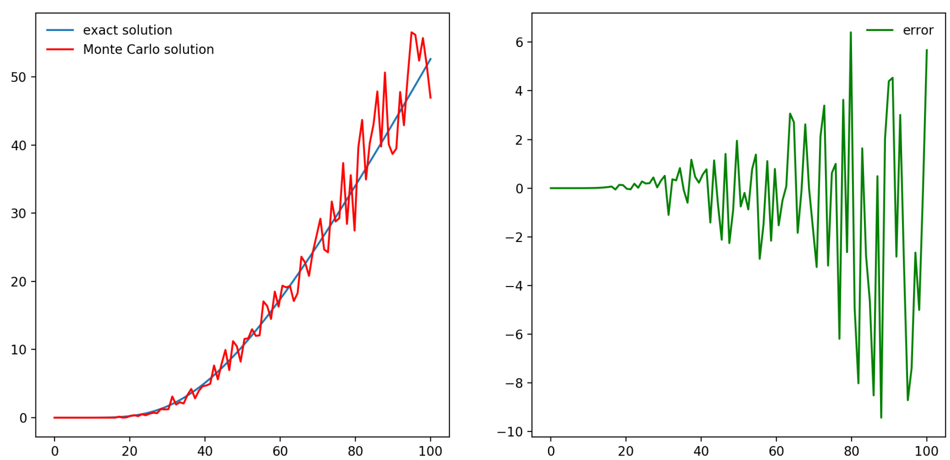

Figure 2.8: Plotted stock option price between MCE with M=100 and exact solution. Also right plot is the error function between them. We choose $M = 100, N = 200, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = [0, 100]$

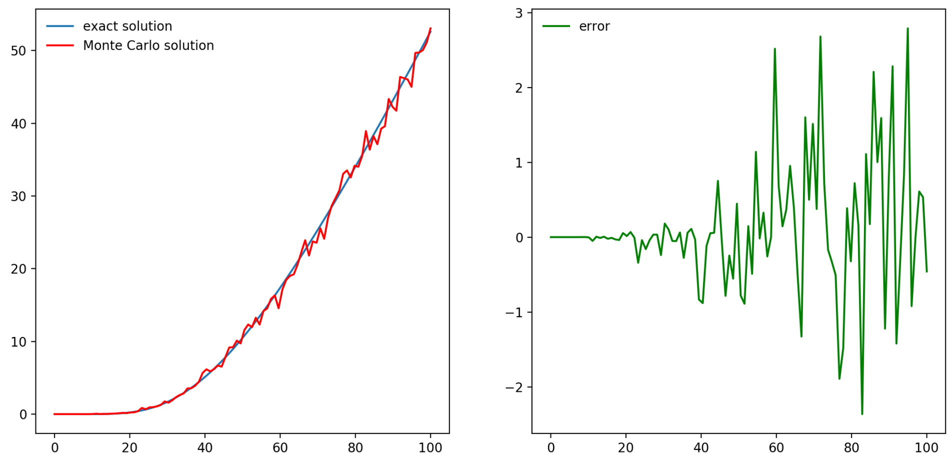

Figure 2.9: Plotted stock option price between MCE with M=1000 and exact solution. Also right plot is the error function between them. We choose $M = 1000, N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = [0, 100]$

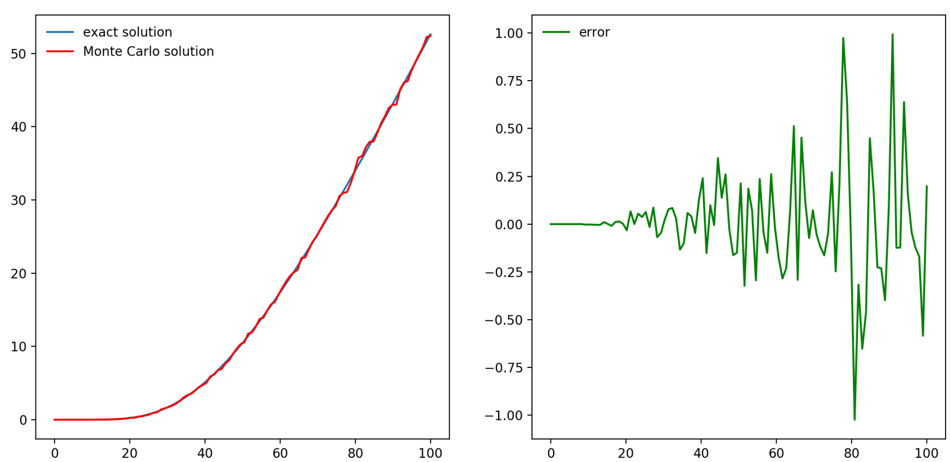

Figure 2.10: Plotted stock option price between MCE with M=10000 and exact solution. Also right plot is the error function between them. We choose $M = 10000, N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = [0, 100]$

Conclusion

Monte Carlo Simulation giving the option price is a sample average, according to the basic principle of statistics, its standard deviation is the standard deviation of the sample divided by the square root of the sample size. So the error reduces at the rate of 1 over the square root of the sample size, the accuracy of Monte Carlo Simulation is increasing as the size of sample increase (M increases).

Moreover, as we increase the volatility, the error seems also increase and the MC plot fluctuate more with higher volatility. Finally, the Monte Carlo Simulation is a simple method to implement, hence is popular used among investment banks.

2.4 Variance Reduction by Antithetic Variate

Antithetic variate is a simple and widely used method to increase the accuracy of Monte Carlo Estimation. The principle of this method is taking its antithetic path. For example, in our above simulation, we generated the path $\{\epsilon_1, \ldots, \epsilon_N\}$, and also to take $\{-\epsilon_1, \ldots, -\epsilon_N\}$, then we plug these two sequences into (2.17), we could get two corresponding payoffs $\left\{f(S_N^{k+}), f(S_N^{k-})\right\}$.

We choose the parameters as below: $r = 0.03, \sigma = 0.5, N = 200, K = 50, T = 1$. They are the same as example 2 in Section 2.3.

Result after applying antithetic variate

MCE(100)	**MCE(1000)**	**MCE(10000)**	**Eaxct solution**
$V(40) = 6.097$	$V(40) = 5.354$	$V(40) = 5.147$	$V(40) = 5.104$
$V(45) = 7.832$	$V(45) = 7.799$	$V(45) = 7.621$	$V(45) = 7.573$
$V(50) = 10.273$	$V(50) = 10.387$	$V(50) = 10.472$	$V(50) = 10.481$

Table 2.7: Comparison between MCE and exact solution for BSM after applying antithetic variate. We try the different M from M=100, M=1000 to M=10000, see the change of accuracy.

Absolute error comparison after applying Antithetic Variate

	MCE(100)	**MCE(1000)**	**MCE(10000)**
$S = 40$	$Err = 0.993$	$Err = 0.250$	$Err = 0.043$
$S = 45$	$Err = 0.259$	$Err = 0.226$	$Err = 0.048$
$S = 50$	$Err = 0.208$	$Err = 0.094$	$Err = 0.009$

Table 2.8: We calculate the difference between MCE calculation and exact solution, then take the absolute value, we want to see how much digit of accuracy that MCE could achieve.

Relative error comparison after applying Antithetic Variate

	MCE(100)	**MCE(1000)**	**MCE(10000)**
$S = 40$	$Err = 19.46\%$	$Err = 4.90\%$	$Err = 0.84\%$
$S = 45$	$Err = 3.42\%$	$Err = 2.98\%$	$Err = 0.63\%$
$S = 50$	$Err = 1.98\%$	$Err = 0.90\%$	$Err = 0.09\%$

Table 2.9: We use the formula: $Err = \frac{abs(MCE-exact)}{exact}$ to compute the relative error, to see how much percent is away the exact solution.

Conclusion

If we compare table 2.7, 2.8, 2.9 with 2.4, 2.5, 2.6, we could detect that the error becomes smaller after we applied the antithetic variate method

2.5 Explicit finite difference to compute European options price

Finite difference methods are aiming to obtain numerical solutions to partial differential equations. They constitute a very powerful and flexible technique and are capable of generating almost accurate numerical solutions for PDE. There are many finite difference methods, in this chapter, we will focus on explicit finite difference method, which is also called forward difference.

The idea underlying finite difference methods is to replace the partial derivatives by approximations based on Taylor series expansions near the point. For example, the partial derivatives $\frac{\partial u}{\partial \mu}$ can be written by the limiting difference [5]:

$$\frac{\partial u}{\partial \tau}(x, \tau) = \lim_{\delta\tau \to 0} \frac{u(x, \tau + \delta\tau) - u(x, \tau)}{\delta\tau}. \tag{2.22}$$

Instead of taking the limit $\delta\tau \to 0$, we regard $\delta\tau$ as nonzero but small, we obtain the approximation

$$\frac{\partial u}{\partial \tau}(x, \tau) \approx \frac{u(x,\tau+\delta\tau)-u(x,\tau)}{\delta\tau} + \mathcal{O}(\delta\tau). \tag{2.23}$$

We use (2.22) to represent each differential term in (2.10). But when we deal with (2.10), we need to use two-dimension Taylor expansion, normally, we use mesh points to represent this idea, we write

$$V_n^m = V(n\delta x, m\delta\tau). \tag{2.24}$$

Then we could come up with every term in (2.10) using this mesh points representation.

$$\frac{\partial V}{\partial t} = \frac{V_n^{m+1} - V_n^m}{\Delta t} + \mathcal{O}(\Delta t),$$

$$\frac{\partial V}{\partial s} = \frac{V_{n+1}^m - V_{n-1}^m}{2\Delta s} + \mathcal{O}(\Delta s)^2,$$

$$\frac{\partial^2 V}{\partial s^2} = \frac{V^m_{n+1} - 2V^m_n + V^m_{n-1}}{(\Delta s)^2} + \mathcal{O}(\Delta s)^2.$$

We ignore the error terms and plug into (2.10)

$$\frac{V^{m+1}_n - V^m_n}{\Delta t} - \frac{1}{2}\sigma^2 n^2 (\Delta s)^2 \frac{V^m_{n+1} - 2V^m_n + V^m_{n-1}}{(\Delta s)^2} - rn\Delta s \frac{V^m_{n+1} - V^m_{n-1}}{2\Delta s} + rV^m_n = 0. \tag{2.25}$$

Notice, we replace s with $n\Delta s$, after we do some simple algebra, we could get following

$$V^{m+1}_n = \frac{1}{2}(\sigma^2 n^2 \Delta t - rn\Delta t)V^m_{n-1} + (1 - \sigma^2 n^2 \Delta t - r\Delta t)V^m_n + \frac{1}{2}(\sigma^2 n^2 \Delta t + rn\Delta t)V^m_{n+1}. \tag{2.26}$$

We have initial conditions

$$V^0_n = max(S_n - K, 0), n = 0, 1, \ldots, N. \tag{2.27}$$

We also have boundary conditions

$$V^m_0 = 0, V^m_N = 0, m = 1, 2, \ldots, M. \tag{2.28}$$

The following condition of stability is needed to avoid oscillation. [6]

$$0 < \Delta t < \frac{1}{\sigma^2 N^2 + \frac{1}{2} r}. \tag{2.29}$$

Truncation error

Key issue is how accurate the numerical solution is. The usual way to measure is using the formula: (solution of finite different method - exact solution). Truncation error is the error made by truncating an infinite sum by a finite sum, the difference between these two results are truncation error. When we apply finite difference method to approximate solution for BSM, there is a place would have truncation error and would influence the overall accurate of the finite difference method.

When we compare the finite difference solution to the exact solution, we only plot the error from 0 to 2K instead of taking infinite interval. We can see that from the plot, at the very beginning, either the finite difference method or exact solution, they both centered at $y = 0$. The error between them can nearly be detected. However, at the very end of the plot, the difference between them are huge, which means the error is large and accurate is terrible. This phenomenon happened because we set the boundary conditions are all 0. Therefore, when we want to see the error, it is meaningless to count this part since we could simple avoid this huge jump by changing the boundary conditions. If we include this huge error into our consideration, then this finite different method is always inaccurate and inferior than other methods. Therefore, we truncate the error only around strike price in order to improve the accuracy of this finite different method.

Example 1

We choose $N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$

Plot from Python of explicit finite difference

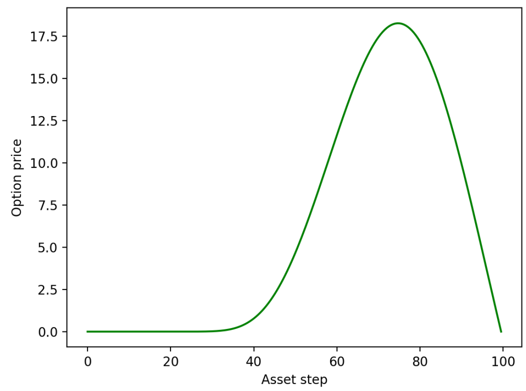

Figure 2.11: Stock option price plot by explicit finite difference with $\sigma = 0.2$. We set the boundary condition is all equal to 0, so at the final time step, the plot drops quickly to 0. We choose $N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$

Plot from Python for comparison

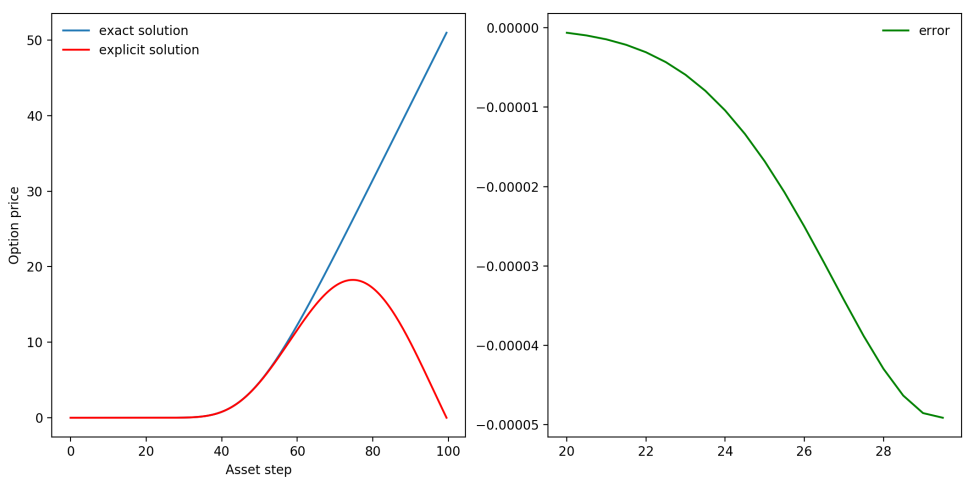

Figure 2.12: Plotted stock option price between explicit finite difference and exact solution. And right plot is the error between them. I only want to see the error of stock price from 40 to 60, which is centered around strike price. We could see the error is much smaller than MCE method. We choose $N = 200, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$

Example 2

We choose $N = 200, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$

Plot from Python of explicit finite difference

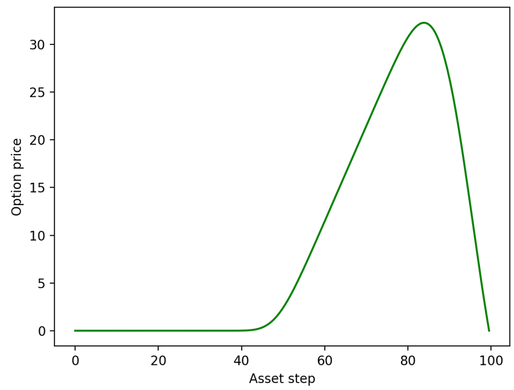

Figure 2.13: Stock option price plot by explicit finite difference with $\sigma = 0.5$. We set the boundary condition is all equal to 0, so at the final time step, the plot drops quickly to 0. We choose $N = 200, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$

Plot from Python for comparison

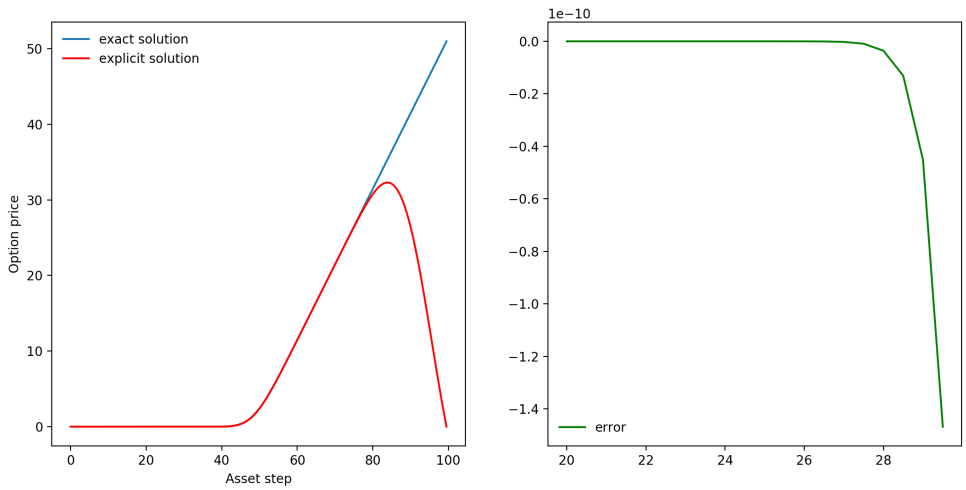

Figure 2.14: Plotted stock option price between explicit finite difference and exact solution. And right plot is the error between them. I only want to see the error of stock price from 40 to 60, which is centered around strike price. We could see the error is even smaller than last example. We choose $N = 200, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$

Conclusion

As we can see, when the stock price is centered around the strike price, the calculation for option price is almost equal to the exact solution, but when the stock price is far away from the strike price which is too high, the error became very large. this is the downside of this explicit method. Since we can not control the time space M, it is decided by the stability term, and also the boundary condition we set is 0 at N, it is decreasing extremely down to 0 at last time point.

2.6 Implicit finite difference to compute European options price

We already know that the methodology behind the finite difference is Taylor expansion, there is another way to write the limit for (2.21), which is:

$$\frac{\partial u}{\partial \tau}(x, \tau) = \lim_{\delta\tau \to 0} \frac{u(x, \tau) - u(x, \tau - \delta\tau)}{\delta\tau}. \tag{2.30}$$

We called this kind of method implicit finite difference, which is using backward to solve the equation. Instead of taking the limit $\delta\tau \to 0$, we regard $\delta\tau$ as nonzero but small, we obtain the approximation

$$\frac{\partial u}{\partial \tau}(x, \tau) \approx \frac{u(x,\tau) - u(x,\tau-\delta\tau)}{\delta\tau} + \mathcal{O}(\delta\tau). \tag{2.31}$$

We use (2.30) to represent each differential term in (2.10). But when we deal with (2.10), we need to use two-dimension Taylor expansion, normally, we use mesh points to represent this idea, we write

$$V_n^m = V(n\delta x, m\delta\tau). \tag{2.32}$$

Then we could come up with every term in (2.10) using this mesh points representation.

$$\frac{\partial V}{\partial t} = \frac{V_n^{m+1} - V_n^m}{\Delta t} + \mathcal{O}(\Delta t),$$

$$\frac{\partial V}{\partial s} = \frac{V_{n+1}^{m+1} - V_{n-1}^{m+1}}{2\Delta s} + \mathcal{O}(\Delta s)^2,$$

$$\frac{\partial^2 V}{\partial s^2} = \frac{V_{n+1}^{m+1} - 2V_n^{m+1} + V_{n-1}^{m+1}}{(\Delta s)^2} + \mathcal{O}(\Delta s)^2.$$

We ignore the error terms and plug into (2.10)

$$\frac{V_n^{m+1} - V_n^m}{\Delta t} - \frac{1}{2}\sigma^2 n^2 (\Delta s)^2 \frac{V_{n+1}^{m+1} - 2V_n^{m+1} + V_{n-1}^{m+1}}{(\Delta s)^2} - rn\Delta s \frac{V_{n+1}^{m+1} - V_{n-1}^{m+1}}{2\Delta s} + rV_n^{m+1} = 0. \tag{2.33}$$

Notice, we replace s with $n\Delta s$, after we do some simple algebra, we could get following

$$V_n^m = \frac{1}{2}(rn\Delta t - \sigma^2 n^2 \Delta t)V_{n-1}^{m+1} + (1 + \sigma^2 n^2 \Delta t + r\Delta t)V_n^{m+1} - \frac{1}{2}(\sigma^2 n^2 \Delta t + rn\Delta t)V_{n+1}^{m+1}. \tag{2.34}$$

Now, we let:

$$a_n = \frac{1}{2}(rn\Delta t - \sigma^2 n^2 \Delta t), \tag{2.35}$$

$$b_n = 1 + \sigma^2 n^2 \Delta t + r\Delta t, \tag{2.36}$$

$$c_n = -\frac{1}{2}(\sigma^2 n^2 \Delta t + rn\Delta t). \tag{2.37}$$

We call this method implicit finite difference because we can not solve it explicitly like last one, we need to use Matrix to solve for each time space, we then need to put the $a_n, b_n, and c_n$ into the Matrix as following: [6]

$$A = \begin{bmatrix} b_1 & c_2 & 0 & \dots \\ a_1 & b_2 & c_3 & \dots \\ \vdots & \ddots & \ddots & c_{N-1} \\ 0 & \dots & a_{N-2} & b_{N-1} \end{bmatrix}, \tag{2.38}$$

$$V^{m+1} = \begin{bmatrix} V_1^{m+1} \\ V_2^{m+1} \\ \vdots \\ V_{N-1}^{m+1} \end{bmatrix}, \tag{2.39}$$

$$b^m = \begin{bmatrix} V_1^m & -a_0 V_0^{m+1} \\ V_2^m & \\ \vdots & \\ V_{N-1}^m & -c_N V_N^{m+1} \end{bmatrix}. \tag{2.40}$$

We have initial conditions

$$V_n^0 = max(S_n - K, 0), n = 0, 1, \ldots, N. \tag{2.41}$$

We also have boundary conditions

$$V_0^{m+1} = 0, V_N^{m+1} = 0, m = 1, 2, \ldots, M. \tag{2.42}$$

Then we need to solve this linear process to get V^{m+1} for each time m.

$$AV^{m+1} = b^m. \tag{2.43}$$

In python, there is a simple command called 'solve' to deal with linear process each time, but this method is not efficient, you could use LU Decomposition Method or SOR Method. If you want to see details, see [5]. Notice, in this implicit finite difference method, we do not need any stability requirement, which means we are free to choose time size M and dt as well.

Example 1

We choose $N = 200, M = 500, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = n\Delta s, n = 0, 1, \ldots, N.$

Plot from Python of implicit finite difference

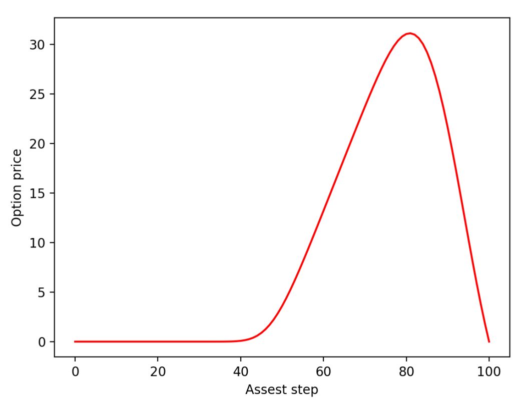

Figure 2.15: Stock option price plot by implicit finite difference with $\sigma = 0.2$. We set the boundary condition is all equal to 0, so at the final time step, the plot drops quickly to 0. We choose $N = 200, M = 500, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = n\Delta s, n = 0, 1, \ldots, N.$

Plot from Python for comparison

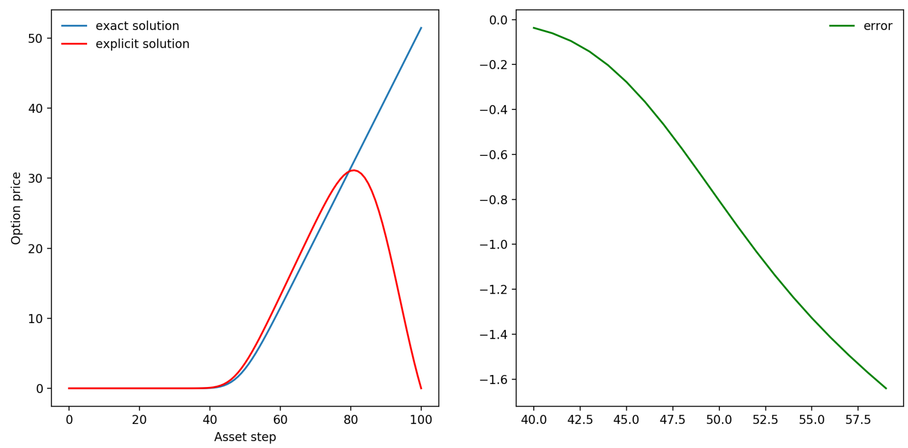

Figure 2.16: Plotted stock option price between implicit finite difference and exact solution. And right plot is the error between them. I only want to see the error of stock price from 40 to 60, which is centered around strike price. We could see the error is much smaller than MCE method. We choose $N = 200, M = 500, K = 50, r = 0.03, \sigma = 0.2, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$.

Example 2

We choose $N = 200, M = 500, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$.

Plot from Python of implicit finite difference

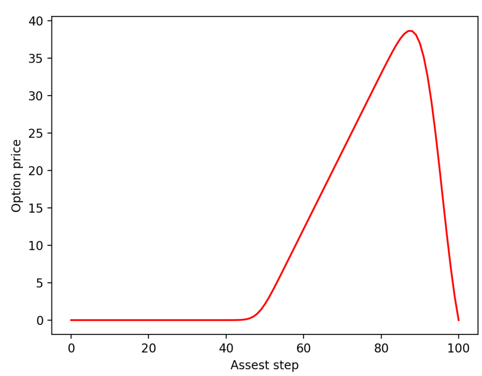

Figure 2.17: Stock option price plot by implicit finite difference with $\sigma = 0.5$. We set the boundary condition is all equal to 0, so at the final time step, the plot drops quickly to 0. We choose $N = 200, M = 500, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$.

Plot from Python for comparison

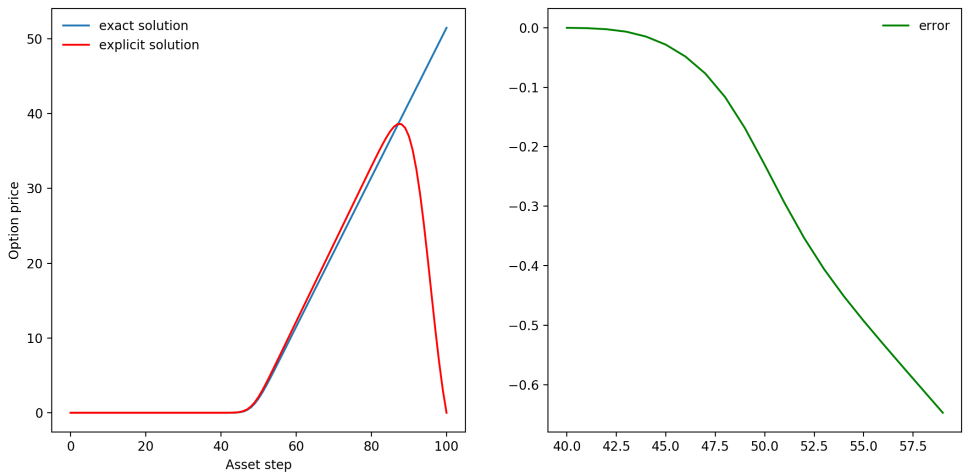

Figure 2.18: Compared stock option price plot between implicit finite difference and exact solution. And detect the error between them. I only want to see the error of stock price from 40 to 60, which is centered around strike price. We could see the error is even smaller than last example. We choose $N = 200, M = 500, K = 50, r = 0.03, \sigma = 0.5, T = 1, S = n\Delta s, n = 0, 1, \ldots, N$.

Conclusion

As we can see, when the stock price is centered around the strike price, the calculation for option price is much more accurate. However, overall this method is less accurate than the other two in some sense. But compared to explicit method, one of the advantages of implicit method is we could control the time size, there is no stability problem.

3 | Interest Rate Model

Time is money, a dollor today is worthy more than a dollar tomorrow since the interest rate. In Black-Scholes Model, we assume the interest rate is constant, but in reality, interest rates also fluctuates and we can also model this as a stochastic process. The uncertainty of the market opens up the possibility of derivatives on bonds, such as bond options, interest rate swaps and others. In this chapter, we will focus on bond options.

3.1 Rederived PDE for Interest Rate Model

We introduced a financial term named forward rate $f(0,T)$, which is people's prediction for future rate. We say it follows Brownian motion: [1]

$$f(t,T) = f(0,T)+\int_0^t \sigma(s,T)dW_s+\int_0^t \alpha(s,T)ds.$$

or in differential form

$$d_t f(t,T) = \sigma(t,T)dW_t + \alpha(t,T)dt. \tag{3.1}$$

where volatilities $\sigma(t,T)$ and the drifts $\alpha(t,T)$ is a deterministic function depends on current time t to maturity data T. (3.1) is a general form for interest rate model. In following chapter, we only focus on one particular and most well known interest rate model called **Vasicek Model, see [7]**

$$dr_t = a(b - r_t)dt + \sigma dW_t, \tag{3.2}$$

where a, b, σ are all positive constant.Then,

$$r_t = e^{-at}(r_0+\int_0^t abe^{au}du + \sigma \int_0^t e^{au}dW_u). \tag{3.3}$$

And by Itô's formula, we get:

$$r_t = e^{-at}(r_0 + b(e^{at} - 1) + \int_0^t e^{au}\sigma dW_u), \tag{3.4}$$

$$r_t = \mu_t + \sigma \int_0^t e^{a(u-t)}dW_u. \tag{3.5}$$

where μ_t is a deterministic function and $\int_0^t e^{au}dW_u$ is gaussian integral of a deterministic stochastic function with respect to Brownian motion, then [8]

$$E[r_t] = \mu_t. \tag{3.6}$$

By taking the expectation of (3.3) and (3.5), we also know that μ_t must satisify:

$$\mu_t = E[r_t] = r_0 + \int_0^t a(b - E[r_u])du,$$

$$d\mu_t = a(b - \mu_t). \tag{3.7}$$

which gives another way to calculate μ_t by taking $E[(r_t - \mu_t)^2] = \sigma^2 e^{-2at} E[\int_0^t e^{2au}du]$, we could use property of Brownian motion

$$var[r_t] = \sigma_t^2 \Longrightarrow r_t \sim N(\mu_t, \sigma^2 t). \tag{3.8}$$

Subtracting b from r_t, we set a driftless gaussian process, which must be a martingale if X(u) =r_t -b, then

$$E[X(u) = X(0)e^{-au}, \tag{3.9}$$

$$E[\int_0^t X(u)du] = \frac{X(0)}{a}(1 - e^{-at}). \tag{3.10}$$

Then we can calculate the variance:

$$var[\int_0^t X(u)du] = \frac{\sigma^2}{2a^3}(2at - 3t4e^{-at} - e^{-2at}). \tag{3.11}$$

Now, from (3.10)

$$E[-\int_0^t r_u du] = E[-\int_0^t (X(u) + b)du], \tag{3.12}$$

$\Longrightarrow$

$$E[\int_t^T r_u du] = E[\int_0^T r_u du - \int_0^t r_u du] = -(\frac{r_t - b}{a})(1 - e^{-a(T-t)} - b(T-t)). \tag{3.13}$$

And we can similarly calculate variance from (3.11)

$$var[\int_t^T r_u du] = \frac{\sigma^2}{2\sigma^3}(2a(T-t) - 3t4e^{-a(T-t)} - e^{-2a(T-t)}). \tag{3.14}$$

We can also express bond price with respect to r_t

$$B(t,T) = E[e^{-\int_t^T r_u du}],$$

where r_u is a function of r_t via filtration $\Longrightarrow B(t,T,r_t) = e^{A(t,T)r_t + D(t,T)}$, where $A(t,T) = \frac{-e^{-a(T-t)}}{a}$, $D(t,T) = (b - \frac{\sigma^2}{2a^2})[A(t,T) - (T-t)] - \frac{\sigma^2 A(t,T)^2}{4a}$.

$$\frac{\partial r_u(r_t)}{\partial r_t} = e^{-a(u-t)},$$

$\Longrightarrow$

$$\int_t^T \frac{\partial r_u(r_t)}{\partial r_t} du = \frac{1}{a}(1 - e^{-a(T-t)}).$$

$$\frac{\partial B}{\partial r_t}(t,T,r_t) = -A(t,T)B(t,T,r_T),$$

$\Longrightarrow$

$$B(t,T,r_t) = c(t,T)e^{-A(t,T)r_t}, \text{ where c is independent of } r_t, \tag{3.15}$$

$\Longrightarrow$

$$e^{-\int_0^t r_u(r_t)du}B(t,T,r_t) = E[e^{-\int_0^T r_u du}/\mathcal{F}]. \tag{3.16}$$

We use Taylor expand in t, then we could use Itô's formula on (3.16)

$$\begin{aligned} B(t,T,r_t)e^{-\int_0^t r_u(r_t)du} &= B(0,T,r_0) + \int_0^t -r_u e^{-\int_0^u r_v dv}B(u,T,r_u)du \\ &+ \int_0^t e^{-\int_0^u r_v dv}\frac{\partial}{\partial u}B(u,T,r_u)du \\ &+ \int_0^t e^{-\int_0^u r_v dv}\frac{\partial}{\partial r_u}B(u,T,r_u)(a(b-r_u)d_u + \sigma(dW_u)) \\ &+ \frac{1}{2}\sigma^2\int_0^t e^{-\int_0^u r_v dv}\frac{\partial^2}{\partial r_u^2}B(u,T,r_u)du. \end{aligned} \tag{3.17}$$

But (3.16) gives a martingale under the P measure, so all the d_u term must sum to zero, this gives us the PDE form of **Vasicek Model**, instead of using $B(t,T,r_t)$, we will use V to represent the value.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial r^2} + a(b-r)\frac{\partial V}{\partial r} - rV = 0. \tag{3.18}$$

3.2 Monte Carlo method to compute bond option

For this interest rate model, we can not solve it by hands directly,so we need to use approximation methods to deal with it. In this section, we will focus on the Monte Carlo method. Remember in section 2.3, we use Monte Carlo to simulate the stock price path, instead, in this section, we will use Monte Carlo to simulate the interest rate, since interest rate is stochastic process in this chapter.

As what we did before, we still use Euler's formula, which gives us following equation to predict interest rate [9]:

$$r_{n+1} = r_n + a(b-r_n)dt + \sigma r_n \varepsilon_{n+1}\sqrt{dt}. \tag{3.19}$$

where we have initial condition $r_0 = r, r_n$ is the predicted interest rate at N steps, and $dt = \frac{T}{N}$, and σ is the interest rate volatility. Each term in sequence ε takes on the value of -1 or 1 with same probability $\frac{1}{2}$, in other words, for each n = 1,2, . . .

$$\varepsilon_n = \begin{cases} 1 & \text{with probability } \frac{1}{2}, \\ -1 & \text{with probability } \frac{1}{2}. \end{cases} \tag{3.20}$$

From above steps, we only calculated one path, but we need to generate more paths, and calculate the average to be accurate. We use two dimensions Euler's formula to do this:

$$r_{n+1}^k = r_n^k + rS_n^k dt + \sigma r_n^k \varepsilon_n \sqrt{dt}. \tag{3.21}$$

Our goal is to find bond option price instead of interest rate, so we still need the formula to calculate the bond price and payoff function.

Bond pricing function with interest rate:

$$B(t) = e^{-\int_t^T r_u du}. \tag{3.22}$$

Notice, from this equation, we could know that all the bond price should be lower than 1, but I multiply 100 in later calculation, just easy for setting up the strike price, making strike price be integer.

Payoff function for option price:

$$f(B) = max(B(t) - K, 0). \tag{3.23}$$

Option pricing requires that the average value of the payoffs, we use V represents the price of option:

$$V = \frac{1}{M} \sum_{k=1}^{M} f(B_N^k). \tag{3.24}$$

Example 1

Now, we use Python to implement this idea. For this example, we choose the parameters as below: $a = 0.1, b = 0.04, \sigma = 0.2, N = 200, K = 97, T = 1$, we adjust M.

Plot from python

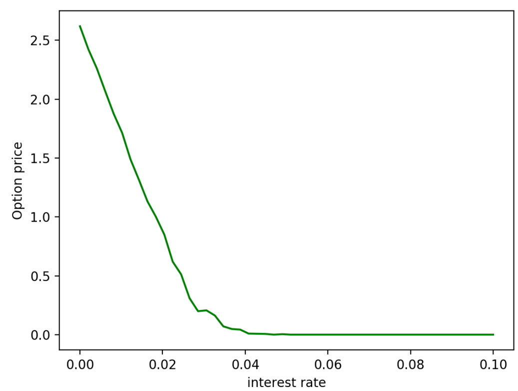

Figure 3.1: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 100, N = 200, K = 97, \sigma = 0.2, T = 1, r = [0.01, 0.1]$

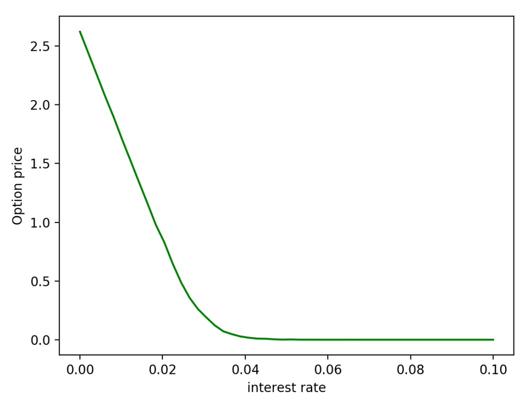

Figure 3.2: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 1000, N = 200, K = 97, \sigma = 0.2, T = 1, r = [0.01, 0.1]$

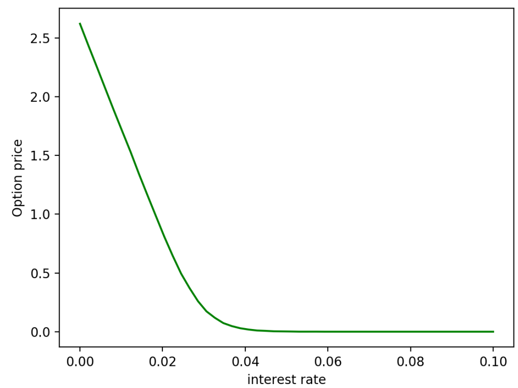

Figure 3.3: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 10000, N = 200, K = 97, \sigma = 0.2, T = 1, r = [0.01, 0.1]$

In this final plot, we will use M=1,000,000 and then use this approximation as our exact solution for interest rate model, we will use this plot to compare the result from explicit and implicit method on interest rate model for $\sigma = 0.2$.

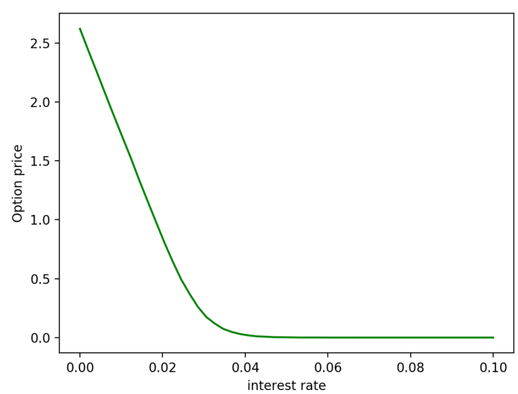

Figure 3.4: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 1,000,000, N = 200, K = 97, \sigma = 0.2, T = 1, r = [0.01, 0.1]$

Example 2

Now, we use Python to implement this idea. For this example, we choose the parameters as below: $a = 0.1, b = 0.04, \sigma = 0.5, N = 200, K = 96, T = 1$, we adjust M.

Plot from python

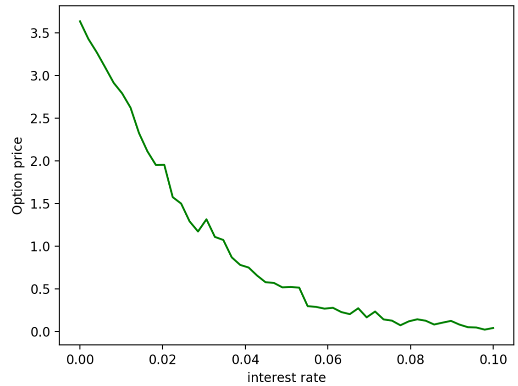

Figure 3.5: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 100, N = 200, K = 96, \sigma = 0.5, T = 1, r = [0.01, 0.1]$

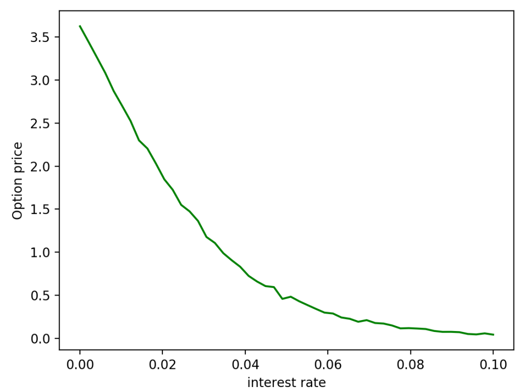

Figure 3.6: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 1000, N = 200, K = 96, \sigma = 0.5, T = 1, r = [0.01, 0.1]$

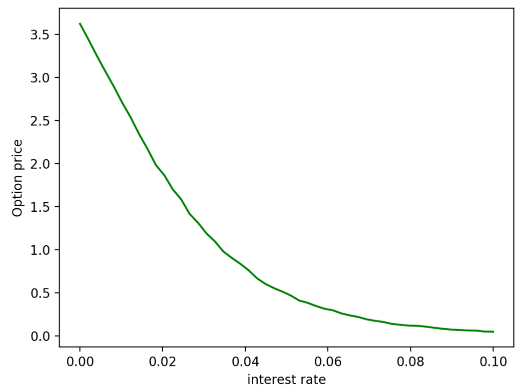

Figure 3.7: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 10000, N = 200, K = 96, \sigma = 0.5, T = 1, r = [0.01, 0.1]$

So same thing here, we will use M=1,000,000 as our exact solution for interest rate model for $\sigma = 0.5$.

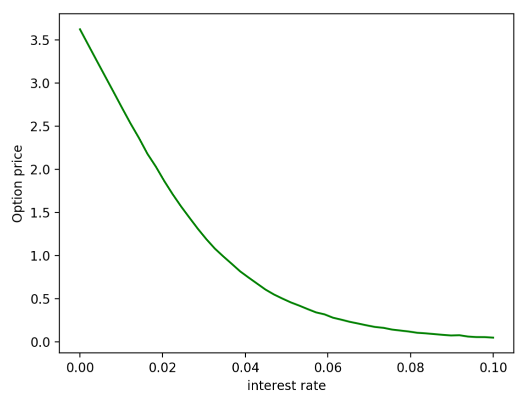

Figure 3.8: Bond option price plot by MCE. We choose $a = 0.1, b = 0.04, M = 1,000,000, N = 200, K = 96, \sigma = 0.5, T = 1, r = [0.01, 0.1]$

Conclusion

For interest rate model, we do not have exact solution, so we could assume large generated path as our exact solution. The same thing happened like we did for Black Scholes Model, as we increase the volatility, the MC plot fluctuate more with higher volatility. The plot for bond option price is the opposite way for stock option price since the interest rate and bond price have inverse relationship. The bond price keeps lower when interest rate is higher since people would like to put money in the bank for risk free interest rate instead of investing in bond.

3.3 Explicit method to compute bond option

We already familiar with the explicit method, we want to duplicate the same as we did for Black Scholes Model. We want to rewrite the (3.18) as below:

$$\frac{\partial V}{\partial \tau} + \frac{1}{2}\sigma^2 \frac{\partial^2 V}{\partial r^2} + a(b-r)\frac{\partial V}{\partial r} - rV = 0. \tag{3.25}$$

Then we let $\tau = T - t$, we would like to this change of variable is that we want to solve this PDE backward since we know all the information at maturity, but we want to know information at the very beginning [10]. Then (3.25) became:

$$-\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 V}{\partial r^2} + a(b-r)\frac{\partial V}{\partial r} - rV = 0. \tag{3.26}$$

Then we have:

$$\frac{\partial V}{\partial t} = \frac{V_n^{m+1} - V_n^m}{\Delta t} + \mathcal{O}(\Delta t),$$

$$\frac{\partial V}{\partial r} = \frac{V_{n+1}^m - V_{n-1}^m}{2\Delta r} + \mathcal{O}(\Delta r)^2,$$

$$\frac{\partial^2 V}{\partial r^2} = \frac{V_{n+1}^m - 2V_n^m + V_{n-1}^m}{(\Delta r)^2} + \mathcal{O}(\Delta r)^2.$$

We ignore the error terms and plug into (3.26)

$$-\frac{V_n^{m+1} - V_n^m}{\Delta t} + \frac{1}{2}\sigma^2 \frac{V_{n+1}^m - 2V_n^m + V_{n-1}^m}{(\Delta r)^2} + a(b-r_n)\frac{V_{n+1}^m - V_{n-1}^m}{2\Delta r} - r_n V_n^m = 0. \tag{3.27}$$

After we do some simple algebra, we could get following

$$V_n^{m+1} = \frac{1}{2}(\sigma^2 \frac{\Delta t}{\Delta r^2} - \frac{a(b-r_n)\Delta t}{2\Delta r})V_{n-1}^m + (1 - \frac{\sigma^2 \Delta t}{\Delta r^2} - r_n \Delta t)V_n^m + \frac{1}{2}(\sigma^2 \frac{\Delta t}{\Delta r^2} + \frac{a(b-r_n)\Delta t}{2\Delta r}). \tag{3.28}$$

We have initial conditions

$$V_n^0 = max(B(0) - K, 0), \tag{3.29}$$

where

$$B(0) = e^{-\int_0^T r_u du}. \tag{3.30}$$

We also have boundary conditions

$$V_0^m = 3, \tag{3.31}$$

$$V_N^m = 0, m = 1, 2, \dots, M. \tag{3.32}$$

The following condition of stability

$$\frac{1}{2(\frac{\sigma^2}{\Delta r^2} + r_{max})} < \Delta t < \frac{1}{\frac{\sigma^2}{\Delta r^2} + r_{max}}. \tag{3.33}$$

Therefore, we could simply let

$$\Delta t = \frac{1}{1.5(\frac{\sigma^2}{\Delta r^2} + r_(max))}. \tag{3.34}$$

Example 1

We choose $N = 200, K = 97, \sigma = 0.2, T = 1, a = 0.1, b = 0.04, r = [0, 0.08]$

Plot from Python of explicit finite difference

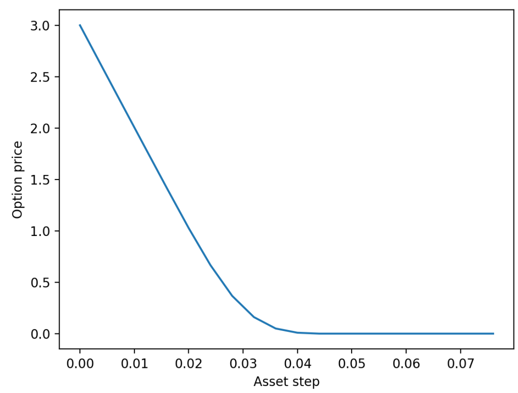

Figure 3.9: Bond option price plot by explicit finite difference with $\sigma = 0.2$. We set the boundary condition are equal to 3 for upper bound and 0 for lower bound. We choose $N = 200, K = 97, r = [0, 0.08]\sigma = 0.2, T = 1, a = 0.1, b = 0.04$

Plot from Python for comparison

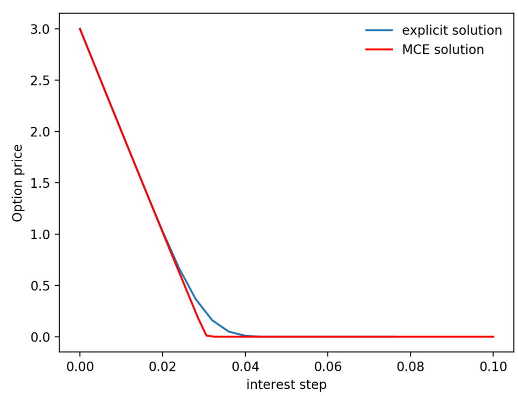

Figure 3.10: Plotted bond option price between explicit finite difference and MCE solution. We could see the error between them is small. We choose $N = 200, K = 97, r = [0, 0.08], \sigma = 0.2, T = 1, a = 0.1, b = 0.04$

Example 2

We choose $N = 200, K = 96, r = [0, 0.08], \sigma = 0.5, T = 1, a = 0.1, b = 0.04$

Plot from Python of explicit finite difference

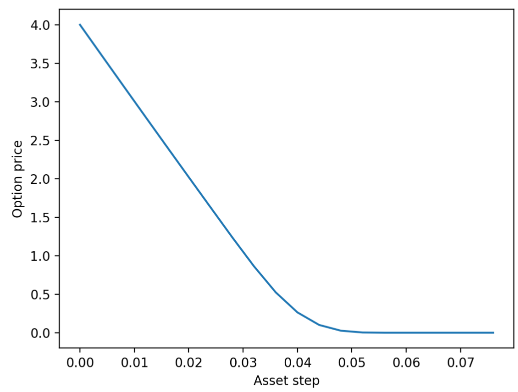

Figure 3.11: Bond option price plot by explicit finite difference with $\sigma = 0.5$. We set the boundary condition is equal to 4 for upper bound and 0 to lower bound. We choose $N = 200, K = 96, r = [0, 0.08], \sigma = 0.5, T = 1, a = 0.1, b = 0.04$

Plot from Python for comparison

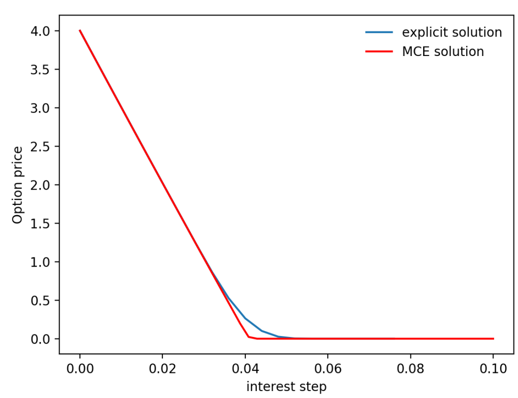

Figure 3.12: Plotted bond option price between explicit finite difference and MCE solution. We choose $N = 200, K = 96, r = [0, 0.08], \sigma = 0.5, T = 1, a = 0.1, b = 0.04$

Conclusion

As we increase the volatility of interest rate and lower the strike the price, we could see the option price increases. And also since the M is big enough is this situation, we could barely see the fluctuation from the comparison plot. I changed the boundary conditions,which is (3.31), in order to not have huge jump at the very beginning of the plot.

3.4 Implicit method to compute bond option

No matter explicit method or implicit method, we both need backward scheme to solve the PDE. Therefore, we finally would get the (3.26). Remember the approximation for implicit finite difference method:

$$\frac{\partial V}{\partial t} = \frac{V_n^{m+1} - V_n^m}{\Delta t} + \mathcal{O}(\Delta t),$$
$$\frac{\partial V}{\partial r} = \frac{V_{n+1}^{m+1} - V_{n-1}^{m+1}}{2\Delta r} + \mathcal{O}(\Delta r)^2,$$
$$\frac{\partial^2 V}{\partial r^2} = \frac{V_{n+1}^{m+1} - 2V_n^{m+1} + V_{n-1}^{m+1}}{(\Delta r)^2} + \mathcal{O}(\Delta r)^2.$$

We ignore the error terms and plug into (3.26)

$$-\frac{V_n^{m+1} - V_n^m}{\Delta t} + \frac{1}{2}\sigma^2 \frac{V_{n+1}^{m+1} - 2V_n^{m+1} + V_{n-1}^{m+1}}{(\Delta r)^2} + a(b - r_n)\frac{V_{n+1}^{m+1} - V_{n-1}^{m+1}}{2\Delta r} - r_n V_n^{m+1} = 0. \tag{3.35}$$

After we do some simple algebra, we could get following

$$V_n^m = \frac{1}{2}\left(\frac{a(b-r_n)\Delta t}{\Delta r} - \sigma^2 \frac{\Delta t}{\Delta r^2}\right)V_{n-1}^{m+1} + \left(1 + \sigma^2 \frac{\Delta t}{\Delta r^2} + r_n \Delta t\right)V_n^{m+1} - \frac{1}{2}\left(\frac{a(b-r_n)\Delta t}{\Delta r} + \sigma^2 \frac{\Delta t}{\Delta r^2}\right)V_{n+1}^{m+1}. \tag{3.36}$$

Now, we let:

$$a_n = \frac{1}{2}\left(\frac{a(b-r_n)\Delta t}{\Delta r} - \sigma^2 \frac{\Delta t}{\Delta r^2}\right), \tag{3.37}$$

$$b_n = 1 + \sigma^2 \frac{\Delta t}{\Delta r^2} + r_n \Delta t, \tag{3.38}$$

$$c_n = -\frac{1}{2}\left(\frac{a(b-r_n)\Delta t}{\Delta r} + \sigma^2 \frac{\Delta t}{\Delta r^2}\right). \tag{3.39}$$

Then we put all these into a linear system:

$$A = \begin{bmatrix} b_1 & c_2 & 0 & \dots \\ a_1 & b_2 & c_3 & \dots \\ \vdots & \ddots & \ddots & c_{N-1} \\ 0 & \dots & a_{N-2} & b_{N-1} \end{bmatrix}, \tag{3.40}$$

$$V^{m+1} = \begin{bmatrix} V_1^{m+1} \\ V_2^{m+1} \\ \vdots \\ V_{N-1}^{m+1} \end{bmatrix}, \tag{3.41}$$

$$b^m = \begin{bmatrix} V_1^m & -a_0 V_0^{m+1} \\ V_2^m & \\ \vdots & \\ V_{N-1}^m & -c_N V_N^{m+1} \end{bmatrix}. \tag{3.42}$$

We have initial conditions

$$V_n^0 = max(B(0) - K, 0), \tag{3.43}$$

where

$$B(0) = e^{-\int_0^T r_u du}. \tag{3.44}$$

We also have boundary conditions

$$V_0^m = 3, \tag{3.45}$$

$$V_N^m = 0, m = 1, 2, \ldots, M. \tag{3.46}$$

Then we need to solve this linear process to get V^{m+1} for each time m.

$$AV^{m+1} = b^m. \tag{3.47}$$

Example 1

We choose $N = 200, M = 2000, K = 97, \sigma = 0.2, T = 1, a = 0.1, b = 0.04, r = [0, 0.08]$

Plot from Python of implicit finite difference

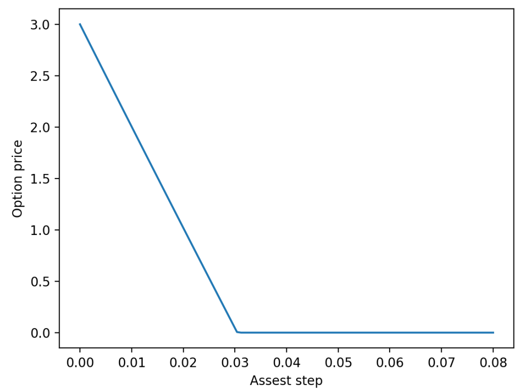

Figure 3.13: Bond option price plot by implicit finite difference with $\sigma = 0.2$. We set the boundary condition are equal to 3 for upper bound and 0 for lower bound. We choose $N = 200, M = 2000, K = 97, r = [0, 0.08]\sigma = 0.2, T = 1, a = 0.1, b = 0.04$

Plot from Python for comparison

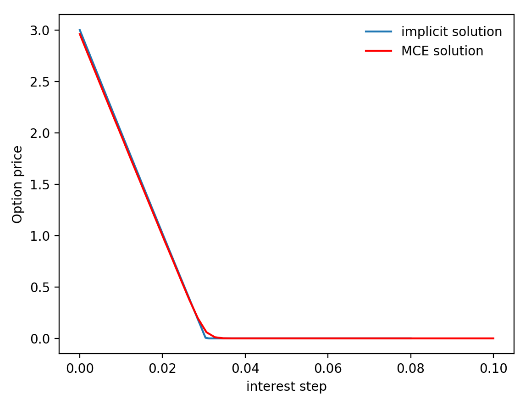

Figure 3.14: Plotted bond option price plot between implicit finite difference and MCE solution. We could see the error between them is small. We choose $N = 200, M = 2000, K = 97, r = [0, 0.08], \sigma = 0.2, T = 1, a = 0.1, b = 0.04$

Example 2

We choose $N = 200, M = 2000, K = 96, r = [0, 0.08], \sigma = 0.5, T = 1, a = 0.1, b = 0.04$

Plot from Python of implicit finite difference

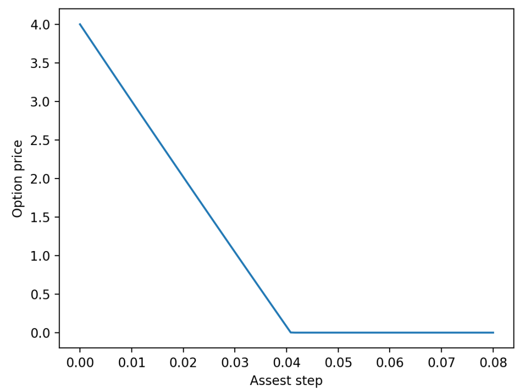

Figure 3.15: Bond option price plot by implicit finite difference with $\sigma = 0.5$. We set the boundary condition is equal to 4 for upper bound and 0 to lower bound. We choose $N = 200, M = 2000, K = 96, r = [0, 0.08], \sigma = 0.5, T = 1, a = 0.1, b = 0.04$

Plot from Python for comparison

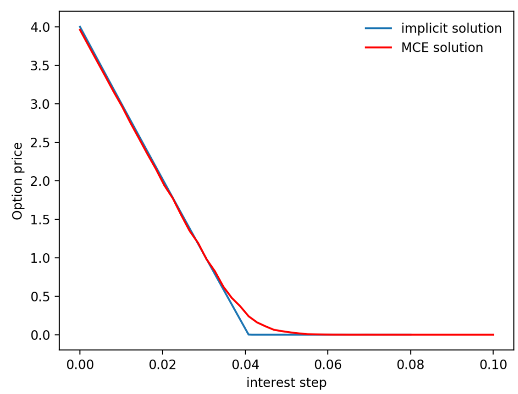

Figure 3.16: Plotted bond option price between implicit finite difference and MCE solution. We choose $N = 200, M = 2000, K = 96, r = [0, 0.08], \sigma = 0.5, T = 1, a = 0.1, b = 0.04$

Conclusion

We tried the same thing as we did for explicit finite difference method, we could find it follows the same rule. From above all three different methods, we could find the error between those is very little. And I also changed the boundary conditions, which is (3.45) for these two examples in order not to have huge jump in the plot.

4 | Conclusion and Future Work

This thesis is combination work with finance and mathematics, and also include how to use programming to help us solve partial differential equation. In Chapter 1, I introduce some background information on this interdisciplinary field, how to link mathematics with finance.

And in Chapter 2, I introduce the most famous and popular model in quantitative finance, which is Black Scholes Model for stock option pricing, including how to get the formula, how to find the exact solution, how to use different numerical methods to solve it. Normally, we could see that Monte Carlo method is to generate many different paths for stock price, this is random process relating to the fact $V = E[e^{-rt}(S_t - K)^+]$. However, we know this is not exactly correct since the market also influenced by other factors, such as regulations and pop news, it is not entirely random, but we ignore those factors in order to easily generate by mathematics. This is popular used by investment banking because they have powerful computers to generate much more paths to get much more accurate results than mine. For finite difference method, we could see this requires more mathematics calculations, but the result in some sense is better than Monte Carlo method's. But they have something in common, the larger of N and M, the more accurate result we could get. Moreover, the slightly difference between explicit method and implicit method is that the explicit method requires stability, this means we can not determine what is M, it is determined by N and σ; but implicit method does not have this problem.

In Chapter 3, I introduce the interest rate model for bond option pricing, particularly is Vasicek Model. Instead of predicting stock price paths, I want to generate the interest rate paths, and then use the formula to compute bond price, and the following steps are to duplicate the same process as I did in Chapter 2. Notice, we could not find exact solution on this interest rate model, so I did not compare the error between these methods and exact solution. However, I did compare between Monte Carlo and finite difference method, just have a basic idea of the error between them.

There are two major future work can be done based on this thesis. First, since limitation on my laptop, I only tried the relative small numbers for N and M, but we could also try large number of N and M, like millions or billions, to detect the error between them for Black Scholes Model. We could also use large enough numbers for Monte Carlo method as our exact solution for interest rate model and find the error between those different methods. Second, Vasicek Model is the just one of the interest rate models, it is also called mean reverting model. There are many other interest rate models such as Hull-White Model and Ho-Lee Model, we could use those models to generate interest rate paths and find corresponding bond price and option price as well.

Bibliography

[1] Martin Baxter and Andrew Rennie. *Financial Calculus*. Press Syndicate of The University of Cambridage, The Pitt Building, Trumpington Street,Cambridge, United Kingdom, 1996.

[2] M. Magdon-Ismail. The equivalent martingale measure: an introduction to pricing using expectations. *IEEE Transactions on Neural Networks*, 12(4):684–693, July 2001.

[3] A.S. Shinde and K.C. Takale. Study of black-scholes model and its applications. *Procedia Engineering*, 38:270 – 279, 2012. INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING.

[4] Helmut G. Katzgraber. Introduction to Monte Carlo Methods. *arXiv e-prints*, page arXiv:0905.1629, May 2009.

[5] Sam Howison Paul Wilmott and Jeff Dewynne. *The Mathematics of Financial Derivatives*. Press Syndicate of The University of Cambridage, The Pitt Building, Trumpington Street,Cambridge, United Kingdom, 1995.

[6] GINA DURA and ANA-MARIA MOSNEAGU. Numerical approximation of black-scholes equation. https://www.math.uaic.ro/ annalsmath/pdf-uri20anale/F1(2010)/Mosneagu.pdf.

[7] Rajna Gibson, Francois-Serge Lhabitant, and Denis Talay. Modeling the term structure of interest rates: A review of the literature. *Foundations and TrendsÂő in Finance*, 5(1âĂŞ2):1–156, 2010.

[8] Muhannad Al-Saadony, Paul Hewson, and Julian Stander. Cholesky decomposition for the vasicek interest rate model. *International Journal of Statistics and Probability*, 2:22, 08 2013.

[9] Banu Ozgurel. Monte carlo simulation for vasicek interest rate model parameters. *Journal of Selcuk University Natural and Applied Sciences*, ISDS 2014, 05 2014.

[10] Antoon Pelsser. *Efficient Methods for Valuing Interest Rate Derivatives*. Springer-Verlag London Berlin Heidelberg, Springer-VerlagLondon 2000, 1998.

Academic Vita

Xinyu Wang

Email: xinyu.wang14@gmail.com **Phone**: 814-880-5817

EDUCATION

Carnegie Mellon University ***Master of Science in Computational Finance*** Class 2021

Passed CFA Level I June 2018

The Pennsylvania State University | Schreyer Honors College University Park, PA

Eberly College of Science | Bachelor of Science in Mathematics and Bachelor of Science in Statistics Graduation: May 2019

Minor: Economic

RELEVANT EXPERIENCE

Research on Financial Mathematics **University Park, PA**

Supervisor: Prof Anna L Mazzucato *Oct 2017-Present*

- Learned three steps to replication in basic Black-Scholes model, utilized the same method to develop different interest rate models both in single-factor HJM and multi-factor HJM for bond pricing.
- Set up option pricing formula for interest rate model, then used Finite-difference Approximations and Monte Carlo methods to solve the equation and tried to determine which one is more accurately under different circumstances.

New Times Securities **Beijing, China**

Investment Research Department Intern *Jun 2017-Aug 2017*

- Analyzed and assessed real estate companies based on their asset collateral to identify qualified companies and provided them Loan Participation Mutual Fund around $80M totally to help them refinance
- Tracked stocks on watch list, assisted team members in analyzing the changes based on value-at-risk, price/earning ratio etc, and discussed weekly strategy
- Researched and assisted quantitative trading group to increase High Frequency Trading speed by using programming language kdp/q to replace Python

Research on Computational Finance **University Park, PA**

Designer and Conductor *Jan 2017-May 2017*

- Developed and completed models in Python to project option values according to previous stock price from Yahoo Finance
- Consolidated options, futures and other derivatives, constructed Black-Scholes-Merton model and Binomial Tree model to trace derivatives prices

Bank of China **Beijing, China**

Credit Department Intern *Jun 2016-Aug 2016*

- Researched approximately 50 corporate clients' credit histories in advance and attended the client monthly meeting
- Reviewed corporate clients' annual audit reports and recorded their consolidated financial statements internally
- Supported Manager in performing financial ratio analysis and writing corporate credit rating reports

ACTIVITIES AND LEADERSHIP

International Communication Internship Program **Hong Kong**

Intern in AXA *Jun 11th-18th, 2018*

- Recognized Asia-Pacific financial market and compared tariff policy between mainland of China and Hong Kong
- Participated in an investment trading competition, making decisions and orders according to given market information as a team

Schreyer Consulting Group **University Park, PA**

Career Development Chair *Jan 2017- May 2018*

- Engaged and practiced case interviews through case study workshops and was highly selected to McKinsey & Deloitte treks in Pittsburgh offices
- Served as a liaison between consulting firms and Schreyer Consulting Group to manage corporate events, expanding recruitment opportunities for honor students

Asia-Pacific Model United Nations Conference (AMUNC) **University of Hong Kong**

Director of Future EU Committee *Jun 19th-24th, 2017*

- Conducted and coached 14 fellow delegates in advanced during the debate in EU Energy Scramble
- Led extensive research on renewable energy, including developing alternative nuclear energy, exploring thorium based technology to help EU achieve 2030 Framework for Climate and Energy

Xi'an Jiao Tong Liverpool University Chinese Student Union **Suzhou**

Vice President *Jan 2016-Aug 2016*

- Assisted the president with changing department structure, separating public relations department and adding small fundraising group in each department leading to raise $10,000 over previous year
- Led students throughout the various activities especially on career development, developed an expanded system for interview preparation and alumni engagement, providing members increased exposure to networking chances