CQF - Exam Two

LECLAIR

Contents

1 Introduction

The Monte Carlo Method is a mathematical technique, which is used to estimate the possible outcomes of an uncertain event.[1]
We will price Binary option using this methodology. We saw that in the Black-Scholes world, the fair value of an option is the present value of the expected payoff of this option at expiry under a risk-neutral random walk for the underlying asset. [2]

After outlining the numerical procedure we will explore how the option price varies as its parameter changes. We will then apply the antithetic variate technique to improve the efficiency of our Monte Carlo Procedure. Finally, we will compare the simulation of the path using both a normal and a lognormal random walk (under $\mathbb{Q}$).

1.1 Assumptions

Let S be the underlying asset of our option, r the risk free rate, σ the volatility of S. We will assume that r and σ are constant and that S is a non-dividend-paying asset. The Risk-Neutral ($\mathbb{Q}$) random walk for S is:

$$dS = Srdt + S\sigma dW_t \tag{1}$$

Using the Euler-Maruyama scheme to discretize the Risk-Neutral random walk, equation (1) become:

$$S(t + \Delta t) = S(t)(1 + r\ \Delta t + \sigma\epsilon\sqrt{\Delta t}) \tag{2}$$

where: ϵ is a random sample from a standard Normal distribution.
In practice is is usually more accurate to simulate "$\ln S$" instead of "S"[1]. Applying Itô's lemma to $d \ln S$ we get:

[1] Definition from: IBM website.

$$d\ln S = (r - \frac{1}{2}\sigma^2)dt + \sigma dW_t \tag{3}$$

Applying Euler-Maruyama scheme to our lognormal random walk, equation (3) become:

$$\ln S(t+\Delta t) - \ln(S) = (r - \frac{1}{2}\sigma^2)\Delta t + \sigma\epsilon\sqrt{\Delta t} \tag{4}$$

$$\Leftrightarrow S(t+\Delta t) = S(t)e^{(r-\frac{1}{2}\sigma^2)\Delta t + \sigma\epsilon\sqrt{\Delta t}} \tag{5}$$

Please note, dt does not need to be small since the expression is exact. Because we assume that r and σ are constant, we can write equation (5) as:

$$S(T) = S(0)e^{(r-\frac{1}{2}\sigma^2)T + \sigma\epsilon\sqrt{T}} \tag{6}$$

Our task is to price binary options. If we consider the pricing of an European option whose payoff only depends on the final asset value, we can use (6) to simulate the final asset price in one single step. Else, if the option is path-dependent (e.g. American option), we will need to use smaller time increments [2]. In our task, we will focus on *Cash-Or-Nothing European call options*.

1.2 Binary Option

A binary (call) option B(K,T) pays 1 if the stock price S_T at expiration T is greater than the strike price K and zero otherwise. [4]

When calculating the value of a binary option we are only interested in its payoff. Let $\mathcal{H}$ be the Heaviside function which is defined as:

$$\mathcal{H}(x) := \begin{cases} 1, & \text{if } x > 0. \\ 0, & \text{otherwise.} \end{cases}$$

Using the above notation, the payoff of a Binary call and a binary put is defined as:

- Binary call: $\mathcal{H}(S-K)$
- Binary put: $\mathcal{H}(K-S)$

1.3 Black-Schole-Merton equation

Using the Black-Schole-Merton equation, we know that the analytical solution for a Cash-or-nothing call and put options are:

Analytical Formula for Binary Call and Put Cash-or-Nothing Formula [3]

Cash-or-nothing call:
- $C = e^{-r(T-t)}N(d_2)$ [a]

Put-or-nothing call:
- $P = e^{-r(T-t)}N(-d_2) = e^{-r(T-t)}(1 - N(d_2))$

Put-Call relationship:
- $C + P = e^{-r(T-t)}$

[a] $N(d_2)$ is the probability that the option is In-The-Money (ITM) at expiry (i.e. $\Pr(S_T > K)$).

1.4 Notation

- $S_T = S(T)$ is the price of the underlying asset at expiry (Time = T)
- $S_t = S(t)$ is the price of the underlying asset at time t.
- S_0 is the initial price of the underlying asset (i.e. at time t=0).
- $\tau = T - t$ is the Time to Maturity of the option.
- method dS refers to equation: (2)
- method $d\ln S$ refers to equation (5)
- Standard Error: $\epsilon_N = \frac{\hat{\sigma}}{\sqrt{N}}$
- Relative Error: R.E. = $\frac{\text{Standard Error}}{\text{Estimate}}$

Unless explicitly stated, I will be using the below parameters by default.

Stock	Strike	τ	r	σ	Dividend
100	100	1	0.05	0.2	0

Table 1: Default parameters

2 Outline of the numerical procedure used [2]

1. Simulate the risk-neutral random walk starting at today's value of the asset S_0 over the required time horizon. This gives one realization of the underlying price path.
   Using the Euler-Maruyama scheme, the stochastic differential equation on (1) becomes
   $\boxed{S_{t+1} = S_t(1 + r\ \Delta t + \sigma\epsilon\sqrt{\Delta t})}$ where:

   - ϵ is a random sample from a standard Normal distribution.
   - $t = 0, 1, 2, \ldots, T-1$

   Using this recursive equation I generate one sample path starting at today's value of the asset S_0 over the time horizon T.

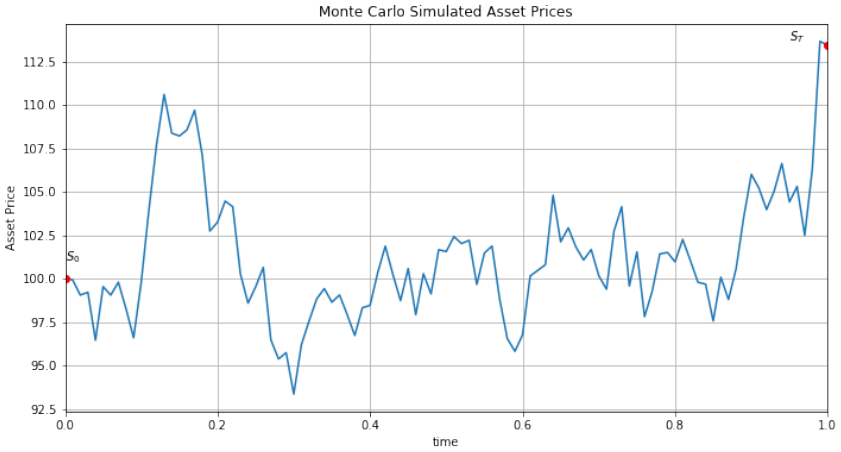

Figure 1: Step 1: One Sample path of S

2. For this sample path, calculate the option payoff. For a european binary call option the payoff is: $\mathcal{H}(S - K)$.

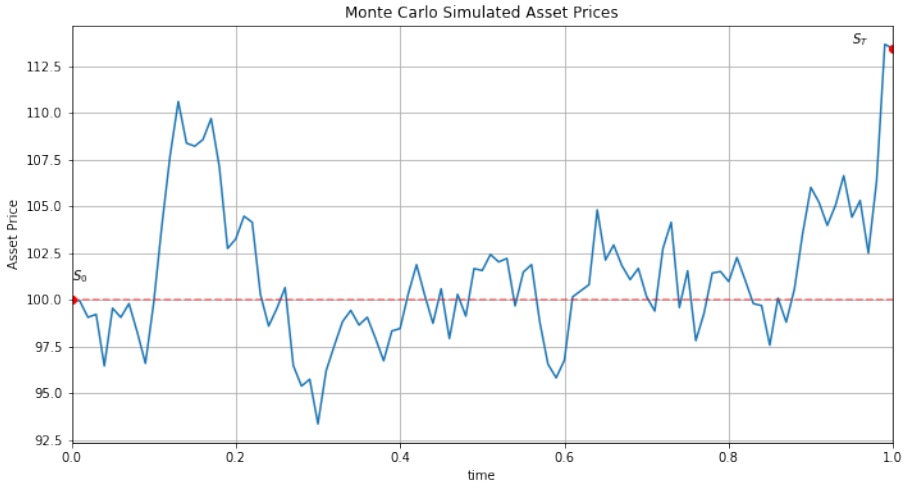

Figure 2: Step 2: Calculating the Payoff

Graphically, we can see that $S_T >$ Strike, therefore: $\boxed{\text{payoff} = 1}$.

3. Repeat step 1 and step 2 many times over the same time horizon.

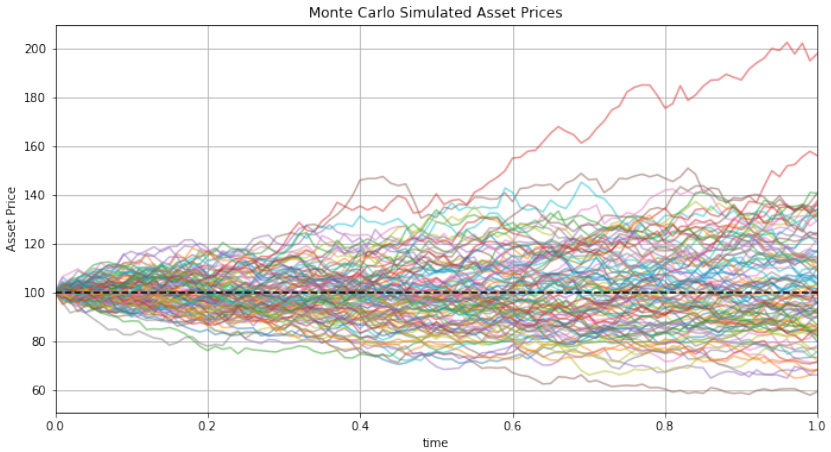

Figure 3: Step 3: 100 first simulations of sample path S. Black doted line is the Option Strike

4. Calculate the average payoff over all sample paths.

5. Take the present value of the average calculated in step 4. This is the estimated value of the option price.

3 Results

Table 2 summarise the theoretical price, estimated price and errors of the following European options:

- Cash-or-nothing Binary Call
- Cash-or-nothing Binary Put
- Asset-or-nothing Binary Call
- Asset-or-nothing Binary Put

	Cash - Call	Cash - Put	Asset - Call	Asset - Put
BSM Price	0.532325	0.418905	63.683070	36.316930
Estimated Price	0.530938	0.416610	63.234330	36.413050
Standard Error	0.001494	0.001492	0.182326	0.130923
Relative Error	0.002814	0.003582	0.002883	0.003596
Absolute Error	0.001387	0.002295	0.448731	0.096111
Runtime	0.008089	0.006687	0.006539	0.006972

Table 2: Result for 1 time step - 100,000 simulations - using default parameters

As stated in section 1.1, we will be considering European Cash-or-Nothing Call options for this assignment. The payoff of such options are *not* path-dependent and only depends on the final asset price. Therefore, we can generate the underlying asset path using the lognormal random walk and one single time step.

3.1 Change in BSM parameters

Amending *one* of the standard BSM parameters[2] while keeping the others constant in our Monte Carlo Procedure, the option price changes as per graph below.

[2]Black-Scholes-Merton Parameters: S_0, Strike, Volarility, Time to Maturity, Risk-Free Rate and Dividend Yield

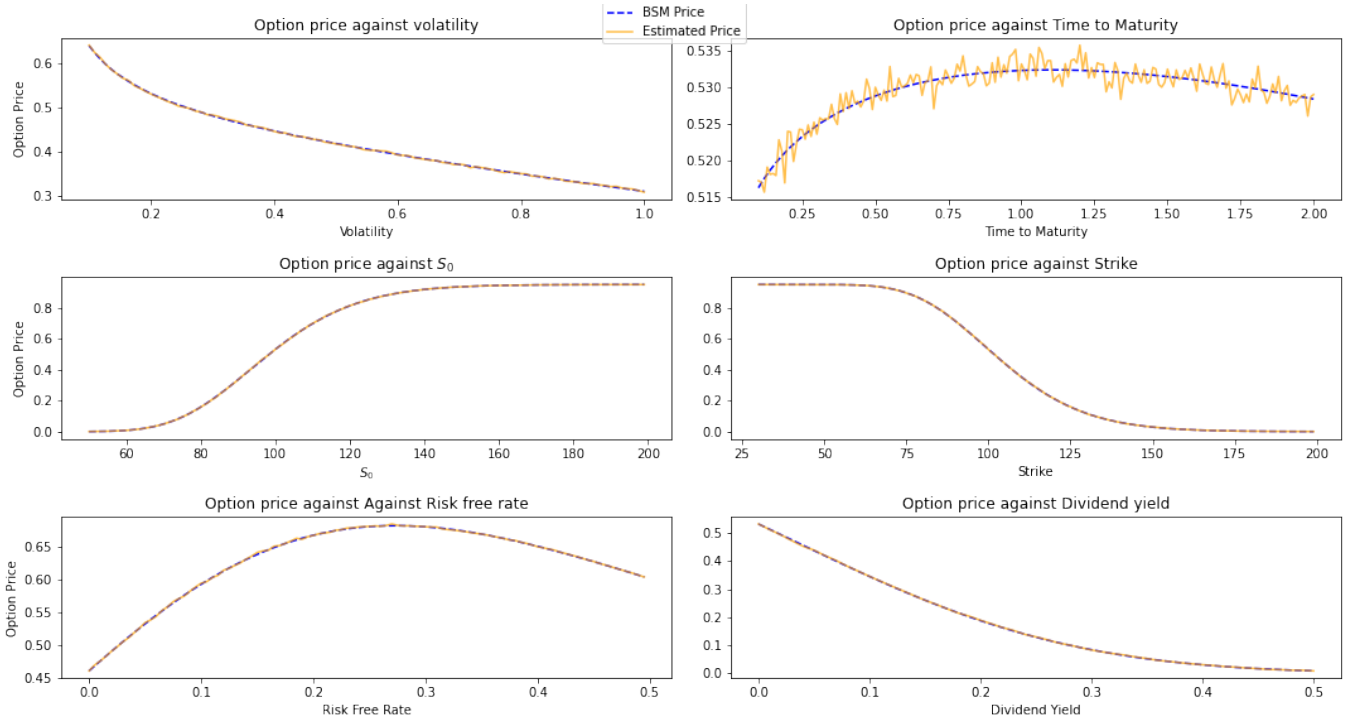

Figure 4: Summary: Change in Option Price

The first thing we notice is that the estimated option price is very accurate compared [3] to the theoretical Black-Scholes-Merton Model.

The following observation can be made:

- As the *volatility* increases option price decreases.
- As the *time to maturity* increases, option price first increases then slowly decreases.
- As the S_0 increases, option price increases.
- As the *Strike* increases, the option price decreases.
- As the *risk-free rate* increases, the option price first increases, then decreases after reaching a maxium at around (0.285,0.683588).
- As the *dividend yield* increases, the option price decreases.[4].

The graphs in Figure 5 shows how the standard error varies as we change on the BSM parameter. The first thing we noticed is that while the *option price against* S_0 graphs has a similar shape of a normal cumulative distribution function (CDF) and the *option price against strike* graph has a similar shape of a (1-CDF) graph, both *Standard Error against* S_0 and *Standard Error against strike* graphs behave like a normal probability density function (PDF).
Standard Error decreases as *time to maturity*, *risk-free rate* or *dividend yield* increases. Finally, as *volatility* increases, the Standard error increases until reaching a maximum at around (0.318,0.001504) then decreases.

[3] The top right Graph seems less accurate compared to the other, but that's only because of the range on the y-axis is smaller compare to the other graphs. Therefore each variation is more visible.

[4] In section 2.1. we assumed the underlying asset was paying no dividend. If we were to assume that the stock was paying a *constant* dividend yield "q", then equation (1) becomes $dS = (r - q)Sdt + \sigma S_t dW_t$. [14]

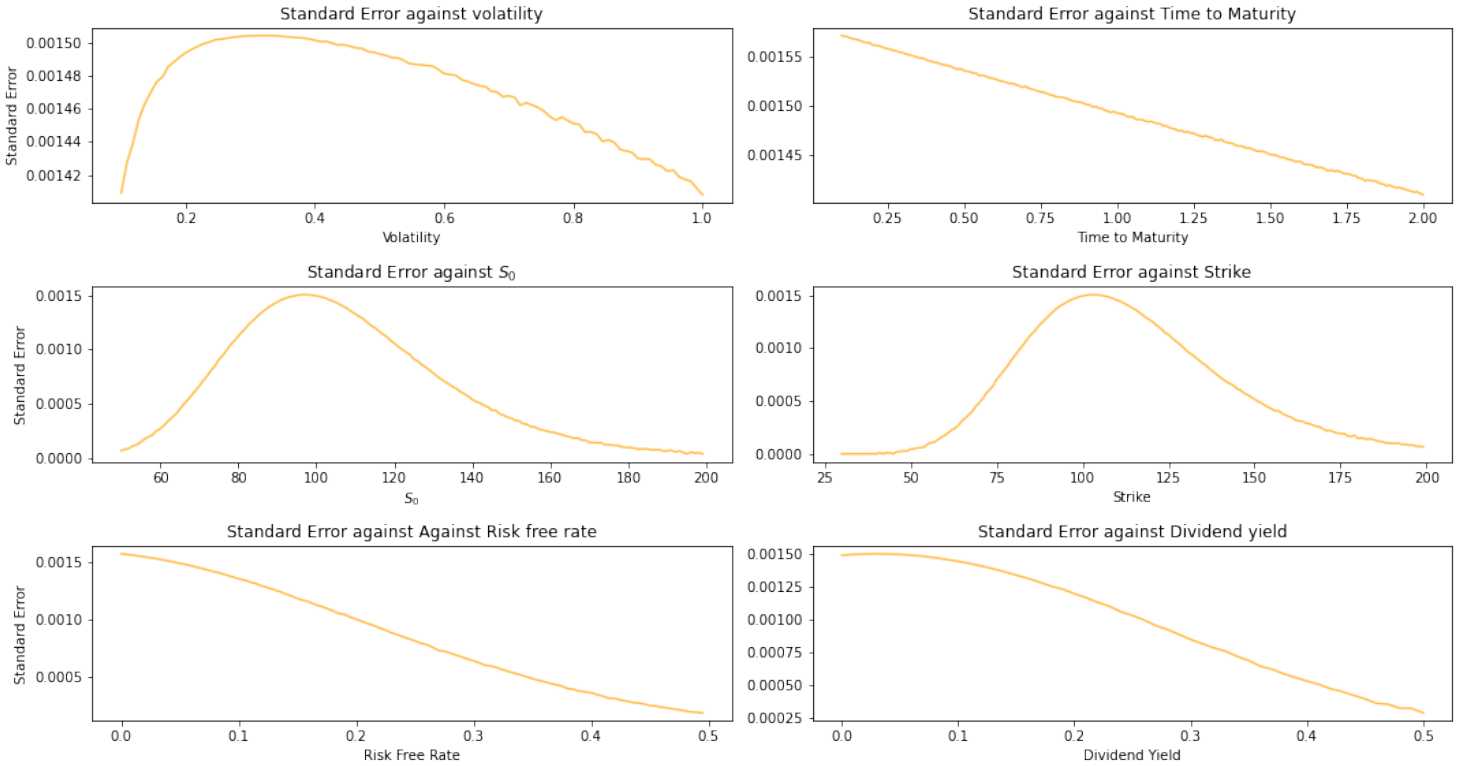

Figure 5: Summary: Change in Standard Error

Figure (6) shows that :

- Relative Error increases as: *volatility*, *Strike* and *dividend yield* increases.
- Relative Error decreases as: *time to maturity*, S_0, *risk-free rate* increases.

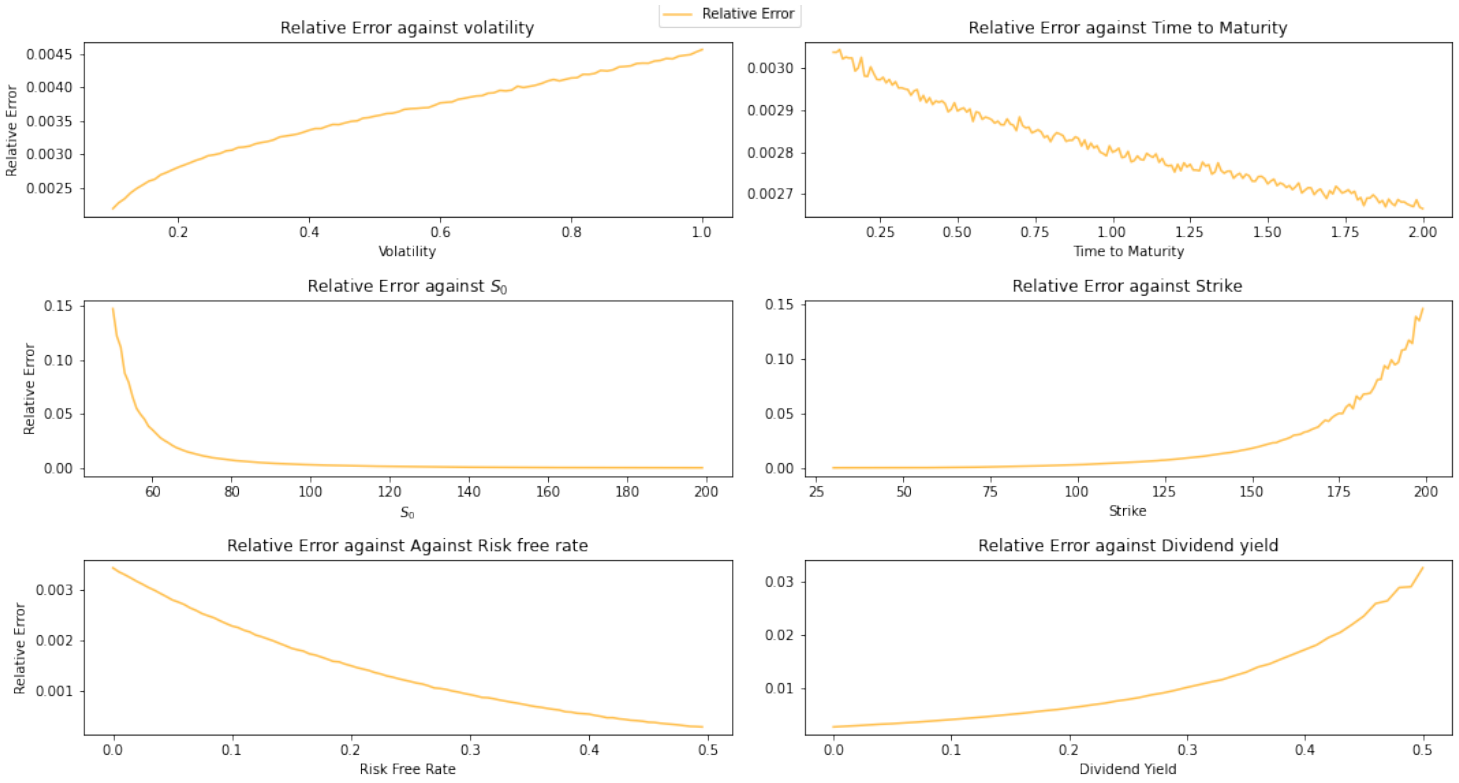

Figure 6: Summary: Change in Relative Error

Looking at the graph (Option Price against Number of Simulation), we can see the Estimated Price converges to theoretical price as the number of simulation increases.

3.2 Changing the number of simulations

The table below shows that as the number of simulations increases, the Standard Error and Relative Error decreases. More precisely, as the number of simulation quadruple, the Standard Error and Relative Error is reduced by a factor of a half. Graphically we can see on Figure 7 that Standard Error has an order of $O(N^{-\frac{1}{2}})$

Simulation	10,000			40,000			160,000		
Strike	90	100	110	90	100	110	90	100	110
BSM Price	0.714121	0.532325	0.353861	0.714121	0.532325	0.353861	0.714121	0.532325	0.353861
Estimated Price	0.708381	0.529454	0.352811	0.714183	0.533378	0.353453	0.715842	0.532469	0.355712
Standard Error	0.004148	0.004726	0.004595	0.002057	0.002360	0.002298	0.001026	0.001181	0.001151
Relative Error	0.005855	0.008926	0.013024	0.002881	0.004426	0.006502	0.001434	0.002217	0.003235
Absolute Error	0.005740	0.002871	0.001050	0.000062	0.001053	0.000408	0.001721	0.000144	0.001851
Runtime	0.001835	0.000879	0.000721	0.002441	0.002425	0.002293	0.010214	0.010754	0.010257

Table 3: Changing the number of simulations for different strike

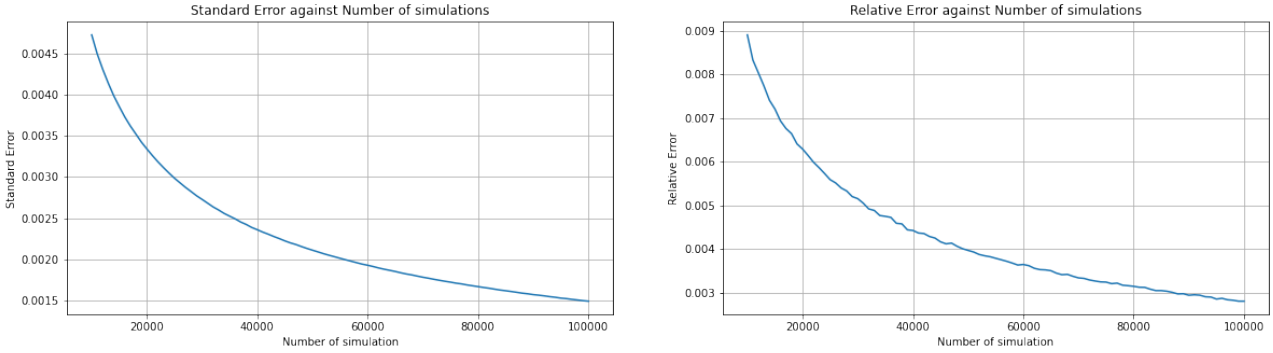

Figure 7: Errors against Number of Simulations

Furthermore, we can see that as the number of simulations increases, the estimated price converges to the theoretical price. We can see that as the number of simulation increases, the runtime increases too. Graphically, we can see that the relation between the number of simulation and the runtime is linear.

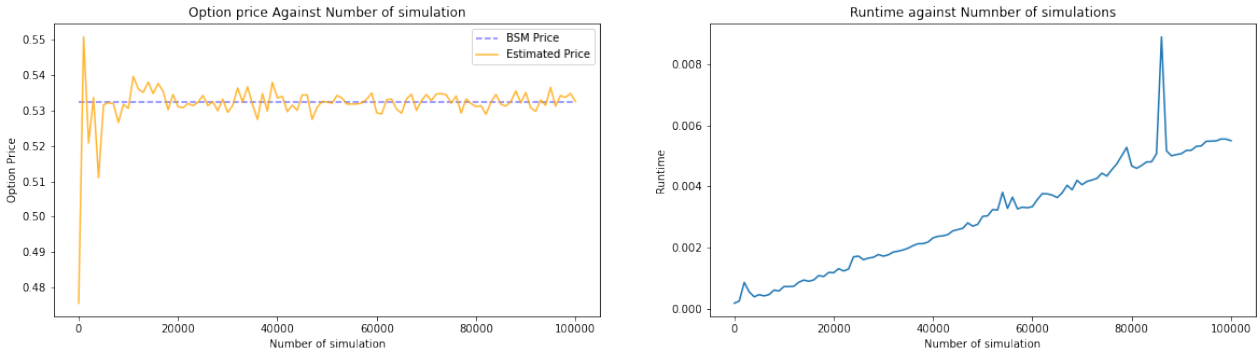

Figure 8: Option Price and Runtime against Number of Simulations

3.3 Variance Technique Reduction: Antithetic

The standard error with the Monte Carlo method is defined as $\frac{\sigma}{\sqrt{n}}$ (where n is the number of simulation). There are two ways to reduce the standard error. One way to reduce the error is to increase the number of simulations (mathematically $\lim_{n\to\infty} \frac{\sigma}{\sqrt{n}} \to 0$). The error is of order $O(n^{-\frac{1}{2}})$. In other world, to reduce the error by a factor of $\frac{1}{2}$ we would need to run 4 times more simulations which is quite inefficient.
Another way to reduce the error would be to reduce the variance σ. The antithetic method is a variance reduction technique which can be easily implemented.

Antithetic variate technique produces two estimate for an option value using the same set of random number from a standard Normal distribution. [5] [2]

To do so, we generate one sample path (same as before) $S_{1,t}$. Then, using the same set of random number we generate the antithetic path $\hat{S}_{1,t}$ by replacing our set of random number by their opposite (i.e. replace ϵ by $-\epsilon$). We then simulate many more sample path, calculate the option payoff then calculate its present value. The estimate of our option value is the average of both option estimate. (i.e. standard option and the option calculated using the antithetic path).

This method works because the standard normal distribution is symetrical about its y-axis. Therefore, if ϵ is a random sample from a standard normal distribution, then $-\epsilon$ is also a random sample from a standard normal distribution.
Recall that the variance of two random variables X and Y is: $Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y)$. In our case, $Cov(S, \hat{S}) < 0 \Rightarrow Var(X+Y) < Var(X) + Var(Y)$. Hence, the antithetic variates method is a *variance reduction* technique.

	Standard	Antithetic
Simulation	10,000	10,000
BSM Price	0.532325	0.532325
Estimated Price	0.528788	0.536161
Standard Error	0.004727	0.003336
Relative Error	0.008938	0.006222
Absolute Error	0.003536	0.003836
Runtime	0.005946	0.009781

Table 4: Monte Carlo method: Comparing Standard with Antithetic Variate method

The figure (9) shows that the Antithetic Variate method:

- is more accurate
- has a lower standard error and relative error compared to the standard Monte Carlo method.
- is more efficient

[5] These two estimate are not independent.

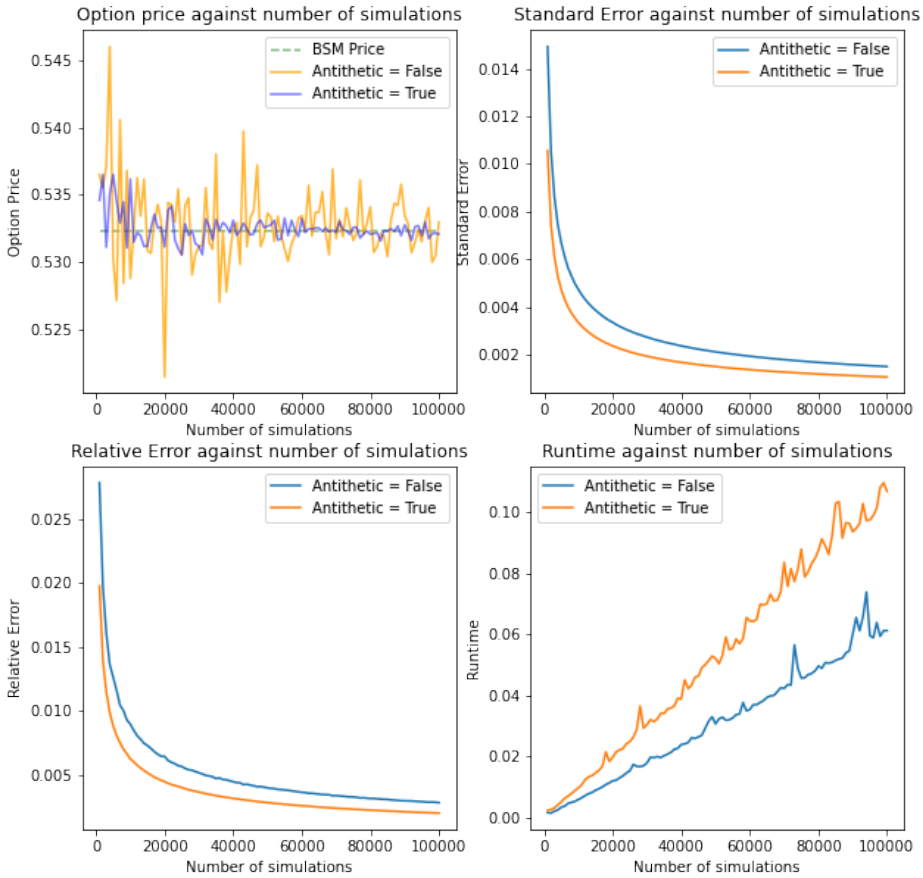

Figure 9: Antithetic variate

4 Any interesting observations and problems encountered.

4.1 Comparing Normal and Lognormal process

In this section, I will show why it is better to use a single step lognormal random walk when simulating Path-*independent* European option.
Unless stated otherwise, the below parameter will be used.

	Stock	Strik	τ	r	σ	Timesteps	Simulations
Value	100	100	1	0.05	0.2	100	100,000

Table 5: Default Parameters

The graphs below show that the standard error and relative error behave similarly for both a normal random walk and lognormal random walk. The only interesting point about these two graphs is that the Standard Error and Relative Error decreases as the number of simulations increases (as shown in Section 3.2).

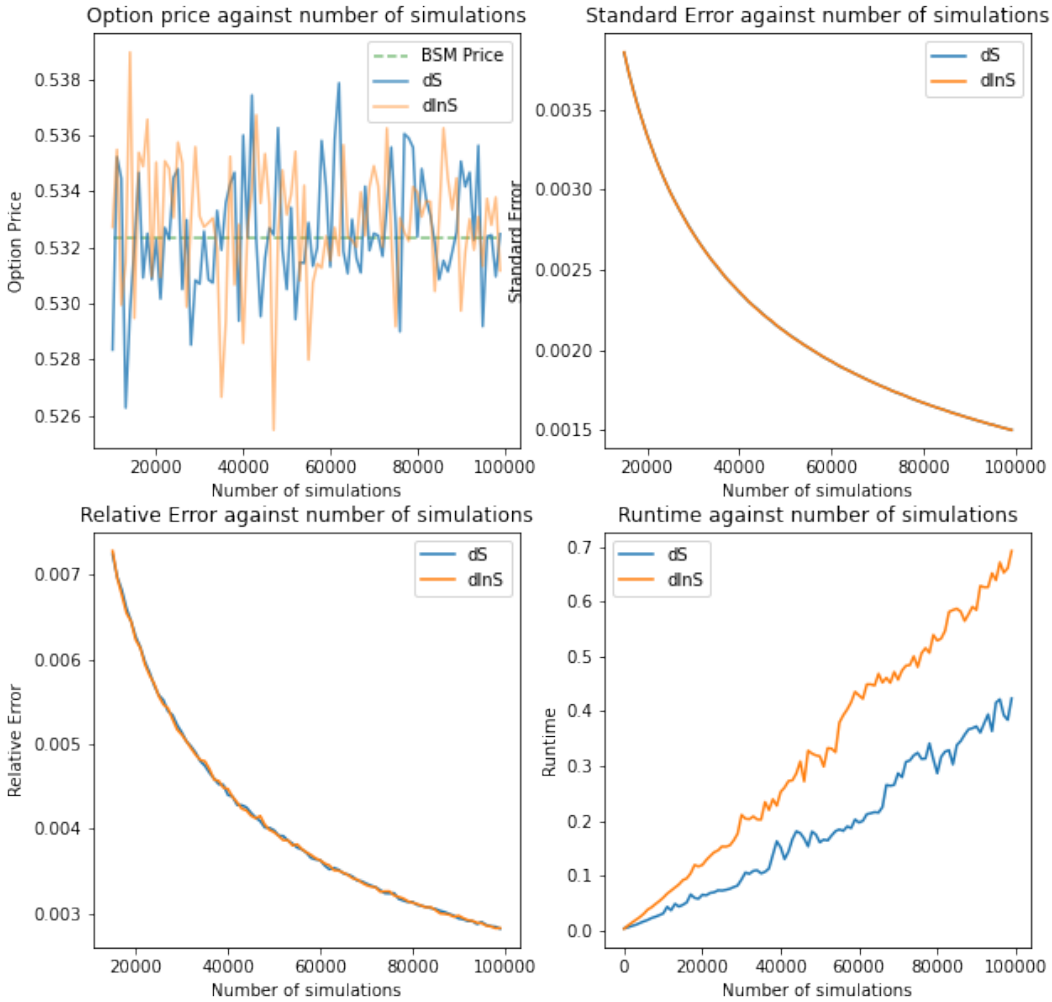

Figure 10: Comparing dS with d lnS - 100 timestep

4.1.1 Multi timestep: $\Delta t = 1/100$

The first graph shows how the Monte Carlo Method price converges toward the theoretical price. As the Number of simulation increases, the amptitude decreases. Both method (dS and dlnS) seems to converge at a similar rate.

The second graph shows that the first the method dS is more efficient than dlnS.

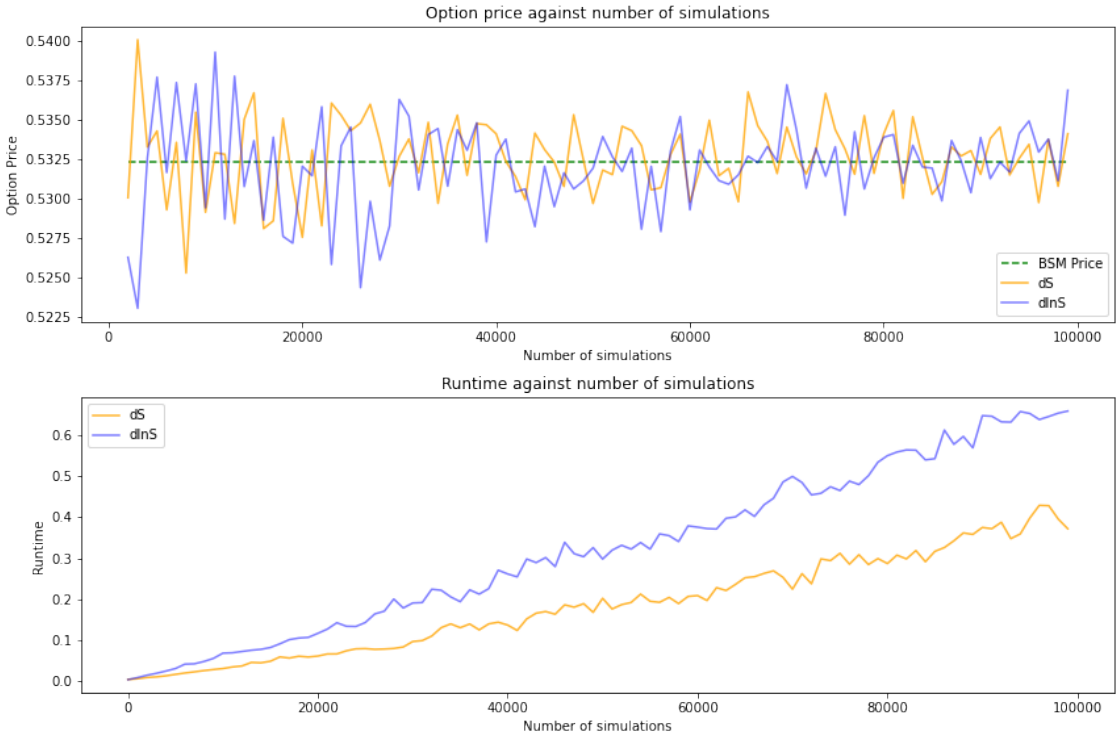

Figure 11: Image/dS Vs dlnS - 100 timestep - Price and Runtime

With the two facts mentioned above, the method dS seems to be more advantageous. In the next sub-section we will highlight the advantage of dlnS method.

4.1.2 One giant leap - 1 timestep $\Delta t = T$

When using a single time step $d \ln S$ is more accurate than the method dS. In fact, the method dS seems to consistently over-estimate the theoretical option price whereas method $d \ln S$ converges pretty quickly.

The second graph shows that the method dS is more efficient, but because it is inaccurate for one time step, the method $d \ln S$ should be preferred.

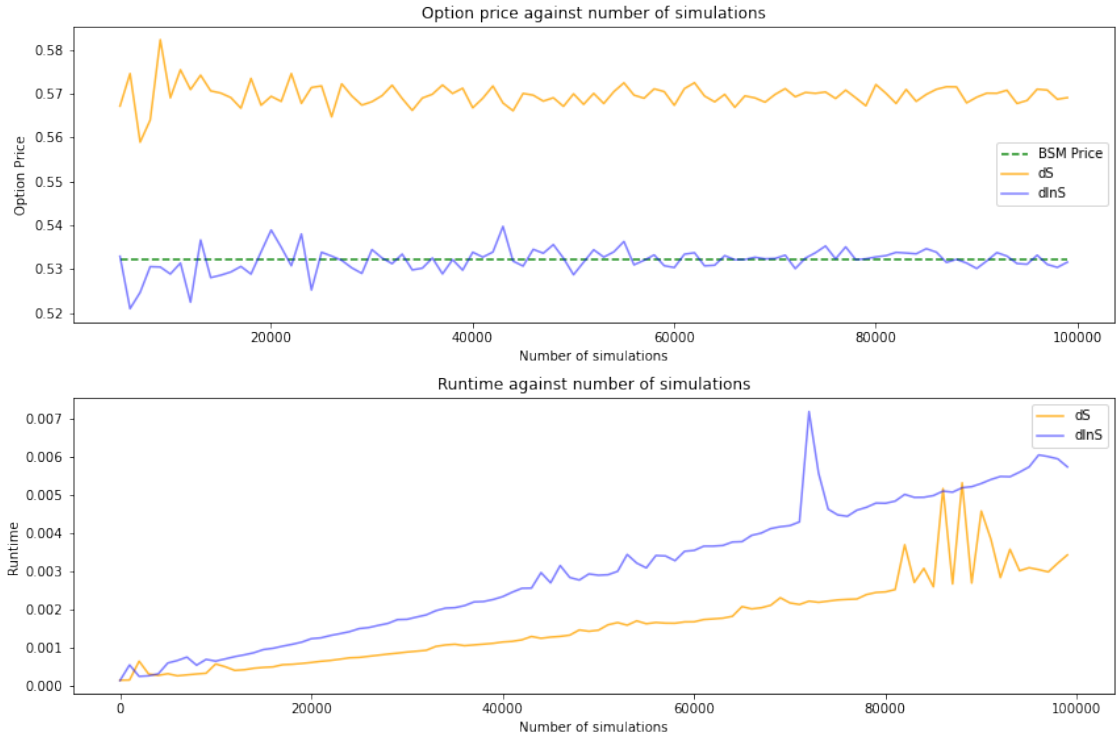

Figure 12: Comparing dS with d lnS - One giant leap

4.1.3 One single step Lognormal Random Walk Versus Multi timestep Normal Random Walk

Method dS uses 100 timestep ($\Delta t = \frac{1}{100} = 0.01$)
Method dlnS uses 1 timestep ($\Delta t = 1$)

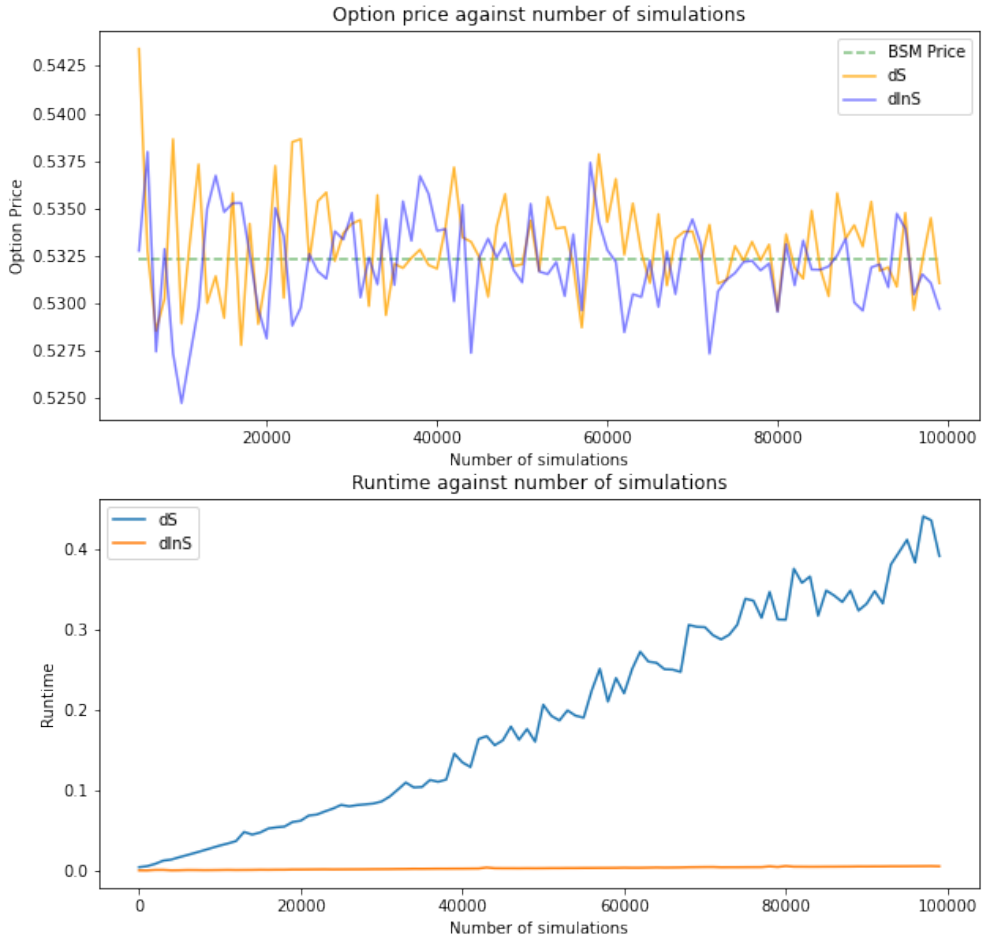

Figure 13: Comparing dS with d lnS - One giant leap

One single step $d \ln S$ method seems to be at least as accurate (if not more accurate) than the multistep dS method. However, using a single step is definitely more efficient than using multistep procedure. That's why using a single step lognormal random walk to simulate our underlying path should be prefered for European option.

5 Conclusion

The purpose of our assignment was to use the Monte Carlo scheme to price binary options. From the results section, we saw that as we changes each parameter, the Monte Carlo model did not deviate much from the theoretical Black-Scholes-Merton formula . We further saw that to reduce the standard error of our Monte Carlo Method, we could increase the number of simulations but that come at the cost of a less efficient procedure. Another way to improve the efficiency and accuracy of our model was to use antithetic variate technique.

Appendices

A Random Number

To generate random number, I used Numpy default RNG (random number generator) which uses the PCG-64 method (permuted congruential generator - 64-bit). Even though Mersenne Twister method have a larger period compare to the the PCG-64 (i.e. $2^{19937}-1$ vs 2^{128}), Melissa E. O'Neill showed in her paper[8] that the PCG family have better statistical property. [6]

B Box-Muller method

The Box-Muller method generates a pair of independent standard Normal random variables from a pair of independent uniformely random variables.

Box-Muller method[10]

Let U_1, U_2 be independent random variables from the same rectangular density function on the interval (0,1). Consider the random variables:

$$X_1 = r\cos(\theta)$$
$$X_2 = r\sin(\theta)$$
$$\text{where: } r = \sqrt{-2\ln(U1)} \quad \text{and} \quad \theta = 2\pi U_2$$

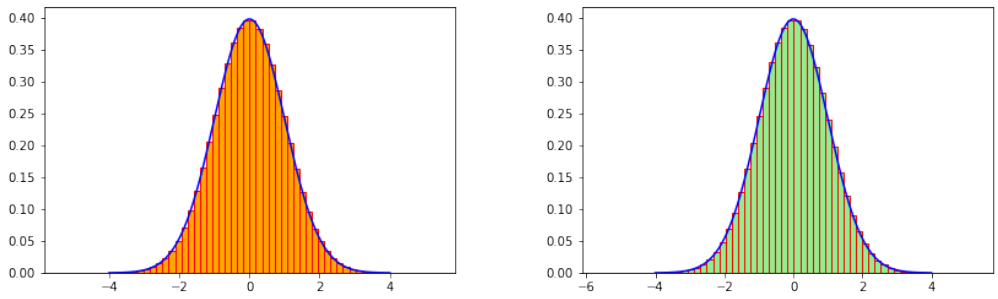

Figure 14: approximation to the Standard Normal distribution using the Box-Muller method on 2×10^6 standard uniformly distributed points.

After generating one million Normally-distributed random number using the Box-Muller method, I can confirm correlation between X_1 and X_2 is -7.322726434320136e-06. Therefore we can assume that there is no correlation between X_1 and X_2 and hence we can use this to simulate two paths simultanously during our Monte Carlo Procedure.

References

[1] Hull, John C. *Options, Futures, and Other Derivatives. Ninth Edition GLOBAL EDITION* PEARSON. pp491-496 ISBN-13: 978-1-292-21289-0

[2] Wilmott, Paul, *Paul Wilmott On Quantitative Finance* - Volume Three,pp1263-1279 John Wiley & Sons, Ltd ISBN-13: 978-0-470-01870-5

[3] Wilmott, Paul, *Paul Wilmott On Quantitative Finance* - Volume One,pp119-120 John Wiley & Sons, Ltd ISBN-13: 978-0-470-01870-5

[4] Gatheral, Jim *The volatility surface, A Practitioner's Guide*, pp104. John Wiley & Sons, Ltd ISBN-13: 978-0-4-471-79251-2

[5] West, Graeme, *Wilmott Magazine, p 70-76, May 2005* http://janroman.dhis.org/finance/Numerical%20Methods/cumulative%20normal.pdf

[6] Wang, Hui, *Monte Carlo Simulation with Applications to Finance* Chapman & Hall/CRC FINANCIAL MATHEMATICS SERIES ISBN-13: 978-1-4665-6690-3

[6] P. L'Ecuyer and R. Simard published a software library "TestU01" for empirical testing of random number generators. [9]

[7] Hart, John F.; et al. (1968). *Computer Approximations.* New York, NY: John Wiley & Sons, Inc. ISBN 978-0-88275-642-4.

[8] O'Neill, Melissa E.(September, 2014) *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation* https://www.cs.hmc.edu/tr/hmc-cs-2014-0905.pdf

[9] Pierre L'Ecuyer and Richard Simard. 2007. *TestU01: A C library for empirical testing of random number generators.* ACM Transactions on Mathematical Software. Article 22 (August 2007), 40 pages. DOI:https://doi.org/10.1145/1268776.1268777

[10] Box, G. E. P.; Muller, Mervin E. (1958). *A Note on the Generation of Random Normal Deviates.* *The Annals of Mathematical Statistics. Vol. 29, No.2 (Jun., 1958) pp.610–611.*

[11] M. Matsumoto and T. Nishimura, *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator*, ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30 (1998) DOI:10.1145272991.272995

[12] Alzubaidi, Hasan*Efficient Monte Carlo Algorithm Using Antithetic Variate and Brownian Bridge Techniques for Pricing the Barrier Options with Rebate Payments* Journal of Mathematics and Statistics, 2016, 12(1):1.11 DOI: 10.3844/jmssp.2016.1.11

[13] Peter Jäckel. *Monte Carlo Methods in Finance WILEY FINANCE Series* ISBN: 978-0-471-49741-7 (2002)

[14] Shreve, Steven E. *Stochastic Calculus for Finance II - Continuous-Time Models* Springer. Chapter 5, p.234-236, ISBN 0-387-40101-6