Multilevel Monte Carlo Simulation Using Terminal Stratification

Author: Yuquan Li

Supervisor: Dr. Daniel J. Duffy

Student Registration Number: 1326297

Programme: MSc Mathematical Finance 2013-2014

Dissertation submitted in partial fulfilment of the degree of

MSc Mathematical Finance

September 2014

To my parents.

To those whom I love and those who love me.

Acknowledgements

I'm indebted to my supervisor Daniel J. Duffy for his invaluable guidance and consistent encouragement during my study. Without his deep insight and patience, this work couldn't have been done.

It is my honor to have Colin Rowat to be my director and give me the chance to join such a rigorous master programme. I am grateful to Nick Webber for helpful reply on my questions about programming.

Fellow students and faculties at department of Economics and school of Mathematics, including David Leppinen, Alex Bespalov, Daniel Loghin, Emma Steadman, Liu Yu, Tong Li, Shengxu Huang, Ning Xie, thank you for making my time at Birmingham memorable.

Abstract

This thesis improves the multilevel Monte Carlo simulation introduced in Giles [2008a] for option pricing. We use stratified sampling on the initial level and thus obtain a further variance reduction. Then we programme the whole procedure of option pricing using this algorithm by C++.

Chapter 1 introduces the mathematical background needed for understanding multilevel Monte Carlo methods. Chapter 2 introduces the multilevel Monte Carlo methods and stratified sampling. And then develops the detailed algorithm for option pricing. Chapter 3 discusses the design of our programme including the discussion of structure, design patterns, and implementation of some algorithms. Chapter 4 shows the numerical results using our algorithm.

Key words: option pricing, computational finance, Monte Carlo, multilevel, variance reduction, stratified sampling, C++ programming

Contents

List of Figures

List of Tables

List of Algorithms and Code

Chapter 1

Introduction

1.1 Review of Prior Work

Monte Carlo methods are widely used in pricing derivatives. Due to the demand of efficiency of the simulation, various improvement methods have been developed by researchers over the world.

Recently, Giles [2008a] has developed the multilevel Monte Carlo (MLMC) path simulation, which is inspired by the multigrid ideas for the iterative solution of linear systems of equations. According to his research, to make the mean-square-error $\mathcal{O}(\epsilon^2)$, the computational complexity of MLMC is $\mathcal{O}(\epsilon^2(\log\epsilon)^2)$ rather than the $\mathcal{O}(\epsilon^{-3})$ in plain Monte Carlo simulation. MLMC is simple to implement and can be combined with other variance reduction techniques.

Based on the work of Giles [2008a], there has been a series of breakthroughs in the relevant fields. Giles [2008b] makes significant improvement by using Milstein discretisation. In the article of Giles and Waterhouse [2009], quasi-Monte Carlo method–a variance reduction techniques is combined with MLMC. It uses rank-1 lattice rule quasi-Monte Carlo integration into MLMC path simulation and obtains a lower computational. Recently, Hoel et al. [2012] try an adaptive algorithm using non-uniform time stepping in MLMC. Their result shows that for a stopped diffusion

problem, the computational complexity can be reduced to $\mathcal{O}((\epsilon^{-1}\log(\epsilon))^2)$.

1.2 Thesis Outline

This thesis concerns multilevel Monte Carlo for option pricing combining with another variance reduction technique–stratified sampling. There has been papers trying to combining MLMC with variance reduction techniques, yet no article has discussed the effect of using stratified sampling in MLMC. To be specific, we consider the terminal stratification to stratify the final values of prices of the underlying asset.

Chapter 2 introduces the mathematical background needed for further discussion of multilevel Monte Carlo simulation. Chapter 3 discusses multilevel Monte Carlo methods and develops the analysis of it. Then we introduces terminal stratification and the implementation in MLMC. Chapter 4 focuses on the programme design and lists some algorithms and code. Chapter 5 shows the numerical results and illustrates the comparisons between stratified MLMC and plain MLMC.

Chapter 2

Mathematical Background

2.1 Introduction

Before introducing the multilevel Monte Carlo method (MLMC) developed by Giles [2008a], we need to have some reviews of option pricing using Monte Carlo methods, which is the foundation of MLMC.

Section 2.1 reviews the concepts of option pricing. It begins with a general description of the geometric Brownian motion, and then introduces the properties of some typical path-dependent options.

Section 2.2 discusses about Monte Carlo methods in the perspective of financial engineering. We begin with the general introduction of Monte Carlo methods, and then develops the method for sample path simulation. Then the framework for measuring the efficiency of Monte Carlo estimators is discussed. This section ends up with the discussion of variance reduction techniques which is the key in MLMC simulation.

2.2 Option Pricing

2.2.1 Geometric Brownian Motion

Geometric Brownian motion is the most fundamental stochastic process in modelling a financial asset. It is assumed that the log returns of the underlying assets prices follow the random walk with drift, or namely, the *Brownian Mothion*. A *geometric Brownian motion* (GBM) can be seen as an exponentiated *Brownian motion* (BM). The utmost advantage of GBM over BM is that it will not generate negative values during the simulations, which is undesirable when modelling the price of a stock or any other asset. Let $S(t)$ denote the stock price at time t, $S(t)$ is said to follow a geometric Brownian motion if it satisfies the following stochastic differential equation (SDE),

$$\frac{\mathrm{d}S(t)}{S(t)} = \mu dt + \sigma \mathrm{d}W(t), \qquad 0 < t < T \tag{2.1}$$

where μ is the drift parameter; σ is the volatility parameter; and $W(t)$ is a *standard* one-dimensional Wiener process, also known as Brownian motion. Both the drift and volatility parameters are assumed to be constants.

By a *standard* one-dimensional Wiener process, we mean a stochastic process $\{W(t), 0 \leq t \leq T\}$ characterized by the following properties:

(i) $W(0) = 0$;

(ii) The mapping $t \mapsto W(t)$ is *almost surely* everywhere continuous;

(iii) The increments $\{W(t_1) - W(t_0), W(t_2) - W(t_1), \ldots, W(t_n) - W(t_{n-1})\}$ are independent for any i and any $0 \leq t_0 < t_1 < \ldots < t_n \leq T$;

(iv) The increments $W(t) - W(s) \sim \mathcal{N}(0, t-s)$, where $\mathcal{N}(\mu, \sigma^2)$ denotes a normal distribution with expectation μ and variance σ^2.

A consequence of (iii) and (iv) is that the increments $W(t_i) - W(t_{i-1})$ are i.i.d normal random variables.

2.2.2 Path-Depdent Options

Path-dependent options are frequently encountered in the pricing of derivatives. The payoffs of these options depend on the price paths $\{S(t_0), S(t_1), \ldots, S(t_n)\}$ of the underlying assets. Pricing such options by simulation will usually result in some discretisation bias.

Asian Options

Asian options are one of the basic forms of exotic options. Unlike vanilla options, the payoff of an Asian option is determined by the average of underlying price over time. Let $\overline{S}$ denote the average of underlying S over the time period $\{0 \leq t \leq T\}$, T the exercise time, and K the constant strike price. The payoffs of Asian calls and Asian puts are

$$P_{call} = e^{-rT}\{\overline{S} - K\}^+ \tag{2.2}$$

$$P_{put} = e^{-rT}\{K - \overline{S}\}^+ \tag{2.3}$$

respectively, where

$$\{\overline{S} - K\}^+ = \mathsf{max}\{0, \overline{S} - K\}$$

The average $\overline{S}$ can be obtained in various ways shown as follows:

- Arithmetic average: discrete monitoring

$$\overline{S} = \frac{1}{n}\sum_{i=0}^{n} S(t_i), \qquad 0 = t_0 < t_1 < \ldots < t_n = T \tag{2.4}$$

- Arithmetic average: continuous monitoring

$$\overline{S} = \frac{1}{T}\int_0^T S(t)dt \tag{2.5}$$

- Geometric average: discrete monitoring

$$\overline{S} = \Big(\prod_{i=0}^{n} S(t_i)\Big)^{\frac{1}{n}}, \qquad 0 = t_0 < t_1 < \ldots < t_n = T \tag{2.6}$$

- Geometric average: continuous monitoring

$$\overline{S} = \exp\Big(\frac{1}{T}\int_0^T \ln(S(t))dt\Big) \tag{2.7}$$

Lookback Options

A Lookback option is an exotic derivative whose value depends on the maximum or minimum prices of underlying asset over the life of the option. Let S_{min} and S_{max} be the minimum and maximum value of the underlying asset over the period $\{0 \leq t \leq T\}$ respectively, where T is the expiry date. A lookback put option with floating strike, for example, has the payoff

$$\begin{aligned} P_{float\_put} =& \{S_{max} - S(T)\}^+ \\ =& S_{max} - S(T). \end{aligned} \tag{2.8}$$

The strike price of such a option (2.8) is not fixed as that of a European vanilla option. There is also a fixed strike version of lookback put option, whose payoff is

$$P_{fix\_call} = \{K - S_{min}\}^+,$$

where K is the strike price in usual sense.

2.3 Monte Carlo Methods

Monte Carlo methods are a family of computational algorithms based on repeated random sampling to yield the numeric results. In the simplest application (Figure 2.1), π is approximated by counting the number of the uniformly scattering points and calculating the ratio between the number of points lying inside or outside the circle inscribing in a square area. The law of large number ensures that the result will converge to the real value as the number of scattering points increase. Monte Carlo simulations are frequently used in pricing various options, particularly if for the complex derivatives whose values do not have a closed-form expression or if additional randomness of underlying assets such as stochastic volatility (Heston [1993]), stochastic interest rates (Vasicek, Hull and White [1990], Cox et al. [1985]) are included in the model.

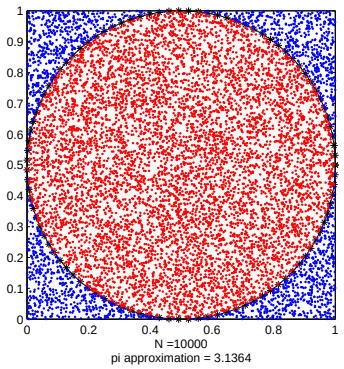


Figure 2.1: Approximation π

2.3.1 Generation Sample Paths

Now we move to the discussion of the simulation of geometric Brownian motion. Suppose $S(t)$ follows ordinary Brownian motion defined in (2.1). We apply Itô's lemma to $V(t) = \log(S(t))$ to give,

$$
\begin{aligned}
dV(t) &= \Big(\frac{\partial V(t)}{\partial t} + \frac{1}{2}\sigma^2 S^2(t)\frac{\partial^2 V(t)}{\partial t^2}\Big)dt + \frac{\partial V(t)}{\partial t}dS(t) \\
&= \frac{1}{2}\sigma^2 S^2(t)\Big(-\frac{1}{S^2(t)}\Big)dt + \frac{1}{S(t)}dS(t) \\
&= -\frac{1}{2}\sigma^2 dt + \frac{1}{S(t)}\Big(\mu S(t)dt + \sigma S(t)dW(t)\Big) \\
&= (\mu - \frac{1}{2}\sigma^2)dt + \sigma dW(t)
\end{aligned}
\tag{2.9}
$$

and hence we have,

$$
d\log(S(t)) = (\mu - \frac{1}{2}\sigma^2)dt + \sigma dW(t) \tag{2.10}
$$

It is straightforward to simulate $W(t_i)$ from the increments since the they are independent and identical normally distributed. Let $Z_1, Z_2, \ldots, Z_n$ denotes the i.i.d standard normal random variables. For any $0 \leq t_0 < t_1 < \ldots < t_n \leq T$, the increments satisfy the following equation:

$$
\mathrm{d}W(t_{i+1}) = W(t_{i+1}) - W(t_i) = \sqrt{t_{i+1} - t_i}Z_i, \qquad i = 0, 1, \ldots, n \tag{2.11}
$$

Set $t_0 = 0$, so that $W(t_0) = 0$. We can generate $W(t_i)$ by the following recursion:

$$
W(t_{i+1}) = W(t_i) + \sqrt{t_{i+1} - t_i}Z_i, \qquad i = 0, 1, \ldots, n \tag{2.12}
$$

Let T be the expiry date for a given option and N the step numbers. Then $\delta t = t_{i+1} - t_i = \frac{T}{N}$. By applying (2.11) to (2.13), for any $0 = t_0 < t_1 < \ldots < t_n = T$, we can simulating the values of $S(t_i)$ by the follow recursive procedure:

$$
\begin{aligned}
S(t_{i+1}) =& S(t_i)\exp\big((\mu - \frac{1}{2}\sigma^2)\delta t + \sigma\sqrt{\delta t}Z_i\big), \\
& i = 0, 1, \ldots, n,
\end{aligned}
\tag{2.13}
$$

where $Z_1, Z_2, \ldots, Z_n$ is the i.i.d standard normal random variables. Usually, $S(0)$ is assumed to be known since we can almost surely get any current stock price. Then the above procedure illustrates the way to generate a GBM path of the underlying asset prices of an option. Figure 2.2 shows the GBM paths generated under different step numbers N, where $S(0) = 100, \mu = 0.05, \sigma = 0.1, T = 1$. Note that this method is *exact* and produces no discretization error. But this is exceptional and most models in complex derivatives pricing can only be simulated numerically.

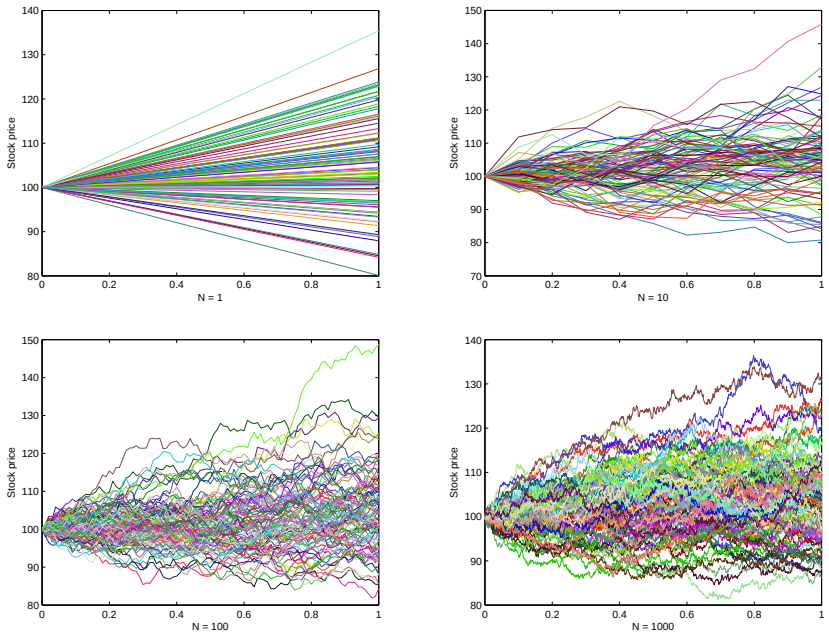


Figure 2.2: GBM path generating

2.3.2 A Path-Dependent Example

In discussing the general Monte Carlo methods, it is useful to have a typical example to refer. Consider a derivative of an underlying asset whose prices follow a geometric Brownian motion defined in (2.1). Under a *risk-neutral measure*, we must ensure the discounted asset prices are martingale which can be achieved through choice of drift (Shreve [2010]). To be simplified, μ is replaced by *risk-free interest rate* r and then (2.1) becomes:

$$\frac{\mathrm{d}S(t)}{S(t)} = r\mathrm{d}t + \sigma\mathrm{d}W(t) \tag{2.14}$$

Hence (2.13) becomes:

$$\begin{aligned} S(t_{i+1}) =& S(t_i)\exp\big((r - \frac{1}{2}\sigma^2)\delta t + \sigma\sqrt{\delta t}Z_i\big), \\ & i = 0, 1, \ldots, n, \end{aligned} \tag{2.15}$$

Let f denotes the payoff function of an option, then the price of which can be given by:

$$P = \mathbb{E}[e^{-rT}f(S(t))], \tag{2.16}$$

where $\mathbb{E}[]$ denotes the expectation and e^{-rT} is the discount factor. The simplest way to calculate the expected discounted payoff is to generate sample paths of $\{S(t_1), S(t_2), \ldots, S(t_N)\}$ and then do the computations along the paths. Take arithmetic average Asian call option described in (2.2) and (2.4) for example, we simulate M paths and each path has step numbers of N. We use Z_{ij} to denote the standard normal random variable in ith path and jth row. We assume that $\{Z_{ij}\}$ are i.i.d. The

simplified steps of pricing options are illustrated in the following algorithm.

---

set $\delta t = \frac{T}{N}$

for i=1,...,M

    for j=1,...,N

        generate Z_{ij}

        set $S_i(t_j) = S_i(t_{j-1}) \exp\big((r - \frac{1}{2}\sigma^2)\delta t + \sigma\sqrt{\delta t} Z_{ij}\big)$

    end

    set $\overline{S}_i = \frac{1}{N}\sum_{j=1}^{N} S_i(t_j)$

    set $P_i = e^{-eT}\{\overline{S}_i - K\}^+$

end

set $\hat{P}_M = \frac{1}{M}\sum_{i=1}^{M} P_i$

Listing 2.1: Algorithm for implified option pricing

2.3.3 Efficiency of Monte Carlo Estimators

The goal of multilevel Monte Carlo is to improve the estimators; before the discussion of that, we first need to explain how to measure among different estimators. The considerations of the efficiency of an estimator can mainly be divided into three parts: variance, computational cost, and bias.

Variance

Consider the estimator $\hat{P}_M = \frac{1}{M}\sum_{i=1}^{M} P_i$ in algorithm 2.1. In this case, $\mathbb{E}[\hat{P}_M] = P$, where P is the quantity being estimated, thus $\hat{P}_M$ is an unbiased estimator. The accuracy of our simulation can be estimated as:

$$
\begin{aligned}
error =& |\hat{P}_M - P| \\
=& \left| \frac{\sigma_P}{\sqrt{M}} \frac{\hat{P}_M - P}{\sigma_P/\sqrt{M}} \right|
\end{aligned}
\tag{2.17}
$$

According to the *Central Limit Theorem*, the distribution of the standardized estimator $\frac{\hat{P}_M - P}{\sigma_P/\sqrt{M}}$ will asymptotically converge to standard normal distribution, as the replication numbers M increases. This statement is often shown as follows:

$$
\frac{\hat{P}_M - P}{\sigma_P/\sqrt{M}} \to \mathcal{N}(0,1), \quad as\ M \to \infty
$$

Hence we have:

$$
\hat{P}_M - P \to \mathcal{N}(0, \frac{\sigma_P^2}{M}), \quad as\ M \to \infty \tag{2.18}
$$

Equation (2.18) indicates that the error of simulation approximately has a distribution of $\mathcal{N}(0, \sigma_P^2/M)$, meaning that the variance of error $\mathbb{V}[\hat{P}_M - P] \approx \sigma_P^2/M$. We can see that the variance of error is affected by both number of replications M and the variance of the estimators σ_P^2. Therefore, other conditions being equal, one should prefer the estimator with lowest variance in comparison with the alternative estimators of the same quantity.

Computational Cost

For two given unbiased estimators of the same quantity, how should we choose if one have lower variance and higher computational cost? A general framework of analysing the efficiency of estimators is developed by Glynn and Whitt [1992].

Suppose for simplicity, two estimators for desired quantity is unbiased with variance σ_1^2/m and σ_2^2/m respectively. The computational cost for each estimator is mc_1

and mc_2 respectively, when m function of evaluation are used. We assume that the computational cost is measured in time. In comparison of two unbiased estimators of the same quantity, one should prefer the one with smaller product between the variance and the computational cost. In other words, a smaller value of $\sigma_i^2 c_i$ indicates a higher efficiency of an estimator for a given quantity.

2.3.4 Variance Reduction Techniques

The computational demands for Monte Carlo methods have motivated a large number of researchers to develop new techniques to increase the efficiency of simulation estimators. The main methods of variance reduction include: antithetic variates, control variates, stratified sampling (Glasserman [2004]), importance sampling, quasi-Monte Carlo methods (L'Ecuyer [2004]), etc. We will discuss about the method of control variates - the key of MLMC, and importance sampling.

Control Variates

The method of control variates is among the most widely used and effective ones for variance reduction. To describe this method, let $\{P_1, P_2, \ldots, P_M\}$ be the outputs of the ith path of simulation, which in our example is the discounted payoff of each replication. We assume that P_i are i.i.d. The most common unbiased estimator for estimating the $\mathbb{E}[P]$ is the sample mean

$$\overline{P} = \frac{1}{M} \sum_{i=1}^{M} P_i, \tag{2.19}$$

which will converge to $\mathbb{E}[P]$ *almost surely* as $M \to \infty$. Suppose we have another output X_i on each path, which is well correlated with Yi and has known expectation $\mathbb{E}[X]$. We now construct another estimator - the control variate estimator for estimating $\mathbb{E}[P]$:

$$\begin{aligned}\overline{P}(\lambda) =& \overline{P} - \lambda\big(\overline{X} - \mathbb{E}[X]\big) \\ =& \frac{1}{M}\sum_{i=1}^{M}\big(Y_i - \lambda(X_i - \mathbb{E}[X])\big),\end{aligned} \tag{2.20}$$

where λ is constant. The control variate estimator in (2.20) is unbiased for estimating $\mathbb{E}[P]$ because

$$\mathbb{E}[\overline{P}(\lambda)] = \mathbb{E}[\overline{P} - \lambda\big(\overline{X} - \mathbb{E}[X]\big)] = \mathbb{E}[\overline{P}] = \mathbb{E}[P].$$

The variance of $P(\lambda)$

$$\begin{aligned}\sigma^2(\lambda) = \mathbb{V}[P(\lambda)] =& \mathbb{V}[P - \lambda(X - \mathbb{E}[X])] \\ =& \sigma_P^2 + \lambda^2\sigma_X^2 - 2\lambda\rho_{PX}\sigma_P\sigma_X,\end{aligned} \tag{2.21}$$

where $\sigma_P^2 = \mathbb{V}[P]$, $\sigma_X^2 = \mathbb{V}[X]$, and ρ_{PX} is the correlation coefficient between P and X. Note that the variance of control variate estimator in (2.20) and common estimator in (2.19) are

$$\begin{aligned}\mathbb{V}[\overline{P}(\lambda)] =& \frac{\sigma^2(\lambda)}{M} \\ \mathbb{V}[\overline{P}] =& \frac{\sigma_P^2}{M}\end{aligned} \tag{2.22}$$

respectively. Therefore, from (2.21), under the condition that

$$\lambda^2\sigma_X^2 - 2\lambda\rho_{PX}\sigma_P\sigma_X < 0,$$

or namely,

$$\lambda^2\sigma_X^2 < 2\lambda\rho_{PX}\sigma_P\sigma_X,$$

we have

$$\mathbb{V}[\overline{P}(\lambda)] < \mathbb{V}[\overline{P}].$$

The optimal value of λ^* to minimize the variance of control variate estimator described in (2.21) is given by

$$\lambda^* = \rho_{PX}\frac{\sigma_P}{\sigma_X} = \frac{\mathrm{Cov}[P,X]}{\mathbb{V}[X]}, \tag{2.23}$$

where $\mathrm{Cov}[]$ denotes the covariance. Substituting (2.23) into (2.21) and simplifying, we find that

$$\frac{\mathbb{V}[\overline{P}(\lambda)]}{\mathbb{V}[\overline{P}]} = 1 - \rho_{PX}^2. \tag{2.24}$$

This indicates that the effect of variance reduction is strongly determined by the correlation between P and X, regardless of the sign of ρ_{PX}.

Example

Take a simple example here to better illustrate control variates in details.

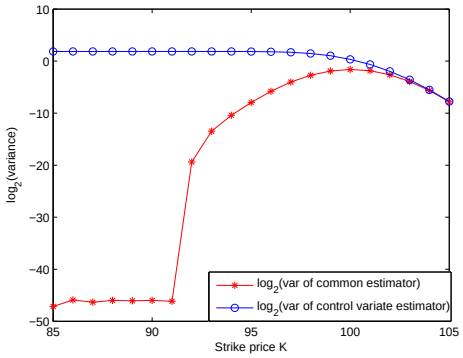


Figure 2.3: log(variance) of estimators under different strike price

Consider a *European Vanilla call option* on underlying asset $\{S(t), 0 \leq t \leq T\}$ with discounted payoff $e^{-rT}\{S(T) - K\}^+$. In a risk-neutral measure, the discounted asset price $e^{-rT}S(t)$ is a martingale and $\mathbb{E}[e^{-rT}S(T)] = S(0)$, thus providing an ap-

propriate control variate. We can therefore construct a control variate estimator

$$\frac{1}{M}\sum_{i=1}^{M}\Big(P_i - \lambda[S_i(T) - e^{-rT}S(0)]\Big), \tag{2.25}$$

where M is the number of paths and P_i the discounted payoff in each replication. In our case, the correlation between P and $S(T)$ depends on the strike price K. For an European call option, the lower the strike price is, the better variance reduction effect can be provided. This is illustrated in Figure 2.3 for the case that $S(t)$ follows GBM with parameters $r = 0.05, \sigma = 0.2, S(0) = 100, T = 1$.

Chapter 3

Multilevel Monte Carlo Simulation in Option Pricing

3.1 Introduction

The computational complexity of multilevel Monte Carlo method has been analysed by Giles [2008a]. The results show that for simple European options using Euler discretisation with GBM SDEs (2.1) and a Lipschitz payoff, the computational complexity can be reduced to $\mathcal{O}(\epsilon^2(\log\epsilon)^2)$ to make the mean-square-error $\mathcal{O}(\epsilon^2)$. For other complex options, such path-dependent ones, numerical results also show significant efficiency enhancement.

Section 3.1 reviews the Multilevel Monte Carlo methods and introduces the complexity theorem.

Section 3.2 reviews stratified sampling and develops the implementation of this variance reduction method in MLMC

3.2 Multilevel Monte Carlo Methods

3.2.1 Control variates and simple two-level MLMC

As mentioned in section 2.3.4, control variate is one of the classic approach of variance reduction techniques. Let $\overline{P}$ be the sample mean of discounted payoff, then the unbiased estimator for $\mathbb{E}(\overline{P})$ is

$$\mathbb{E}[\overline{P}(\lambda)] = \frac{1}{M}\sum_{i=1}^{M}\big(Y_i - \lambda(X_i - \mathbb{E}[X])\big), \tag{3.1}$$

as shown in (2.20).

Now let us consider the simplest case of MLMC, the two-level version MLMC. The construction of this is similar to control variate. Suppose we want to estimate $\mathbb{E}[P_1]$, where P_1 denote the discounted payoff of level 1 with timesteps $h_1 = N^{-1}T$. And we have the similar definition with P_0 whose timesteps is $h_0 = N^{-0}T = T$. The we can construct the unbiased estimator for $\mathbb{E}[P_1]$ as follows:

$$\begin{aligned}\mathbb{E}[P_1] =& \mathbb{E}[P_0] + \mathbb{E}[P_1 - P_0] \\ =& \frac{1}{M_0}\sum_{i=1}^{M_0} P_0^{(i)} + \frac{1}{M_1}\sum_{i=1}^{M_1}\left(P_1^{(i)} - P_0^{(i)}\right)\end{aligned} \tag{3.2}$$

The main difference between control variate estimator in (3.1) and two-level Monte Carlo estimator in (3.2) is that the value of $\mathbb{E}[P_0]$ is unknown (need to be estimated), and we choose $\lambda = 1$ in this case. The key is that P_1 and P_0 are calculated from two sample paths of $S(t)$ with different timesteps but the same Brownian path. To be more specific, the steps of construction two-level MC estimator is illustrated as follows:

level = 0,

$$\begin{aligned}
&\text{path } 1, && S(0), S_1(t_1), S_1(t_2), \ldots, S_1(T) \Rightarrow P_0^{(1)} \\
&\ldots && \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \Rightarrow \ldots \\
&\text{path } i, && S(0), S_i(t_1), S_i(t_2), \ldots, S_i(T) \Rightarrow P_0^{(i)} \\
&\ldots && \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \Rightarrow \ldots \\
&\text{path } M_0, && \underbrace{S(0), S_M(t_1), S_M(t_2), \ldots, S_M(T)}_{\text{number of periods}=N^0} \Rightarrow P_0^{(M)}
\end{aligned}$$

level = 1,

$$\left.\begin{aligned}
&\underbrace{S(0), S_1(t_1), S_1(t_2), \ldots, S_1(T)}_{\text{number of periods}=N^0} \Rightarrow P_0^{(1)} \\
&\underbrace{S^{'}(0), S_1^{'}(t_1), S_1^{'}(t_2), \ldots, S_1^{'}(T)}_{\text{number of periods}=N^1} \Rightarrow P_1^{(1)}
\end{aligned}\right\} \Rightarrow P_1^{(1)} - P_0^{(1)}$$

pair path 1,

$$\ldots \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \Rightarrow \ldots\ldots\ldots$$

$$\left.\begin{aligned}
&\underbrace{S(0), S_{M_1}(t_1), S_{M_1}(t_2), \ldots, S_{M_1}(T)}_{\text{number of periods}=N^0} \Rightarrow P_0^{(M)} \\
&\underbrace{S^{'}(0), S_{M_1}^{'}(t_1), S_{M_1}^{'}(t_2), \ldots, S_{M_1}^{'}(T)}_{\text{number of periods}=N^1} \Rightarrow P_1^{(M)}
\end{aligned}\right\} \Rightarrow P_1^{(M_1)} - P_0^{(M_1)}.$$

pair path M_1,

With the above procedure, we can then calculate $\mathbb{E}[P_1]$ by using (3.2).

3.2.2 Multilevel Monte Carlo

With the introduction of two-level Monte Carlo methods, it is then natural to generalise it into full multilevel version. Consider path simulations with different timesteps $h_l = N^{-l}T$ on each level l, where $l = 0, 1, \ldots, L$, and N is the refinement factor for each level which is usually chose to be 2. Let $P_l^{(i)}$ denote the discounted payoff one level l and path i. Thus we have

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}],$$

and therefore we can construct the unbiased estimator for $\mathbb{E}[P_L]$ as follows:

$$\frac{1}{M_0}\sum_{i=1}^{M_0} P_0^{(i)} + \sum_{l=1}^{L}\left\{\frac{1}{M_l}\sum_{i=1}^{M_l}\left(P_l^{(i)} - P_{l-1}^{(i)}\right)\right\}. \tag{3.3}$$

Again, $P_l^{(i)} - P_{l-1}^{(i)}$ comes from the identical Brownian path. Note that $\mathbb{V}[P_l - P_{l-1}]$ decreases with the level l, thus we could choose M_l adaptively to minimise the overall variance. This is summarized as the following theorem by Giles [2008a]:

Theorem 3.2.1. *Let P denote a functional of stochastic differential equation* $\mathrm{d}S(t) = a(S(t),t)\mathrm{d}t + b(S(t),t)\mathrm{d}W(t), 0 < t < T$ *for a given Brownian path* $W(t)$*, and let* $\hat{P}_l$ *denote the corresponding approximation using a numerical discretisation with time step* $h_l = N^{-l}T$*.*

If there exist independent estimators $\hat{Y}_l$ *based on* M_l *Monte Carlo samples, and positive constants* $\alpha \geq \frac{1}{2}, \beta, c_1, c_2, c_3$ *such that*

(i) $\mathbb{E}[\hat{P}_l - P] \leq c_1 h_l^{\alpha}$*,*

(ii) $$\mathbb{E}[\hat{Y}_l] = \begin{cases} \mathbb{E}[\hat{P}_0], & l = 0, \\ \mathbb{E}[\hat{P}_l - P_{l-1}], & l > 0, \end{cases}$$

(iii) $\mathbb{V}[\hat{Y}_l] \leq c_2 N_l^{-1} h_l^{\beta}$*,*

(iv) C_l*, the computational complexity of* $\hat{Y}_l$*, is bounded by* $C_l \leq c_3 N_l h_l^{-1}$

then there exists a positive constant c_4 *such that for any* $\epsilon < e^{-1}$*, there are values* L *and* N_l *for which the multilevel estimator*

$$\hat{Y} = \sum_{l=0}^{L} \hat{Y}_l$$

has a mean-square-error (MSE) with bound

$$MSE = \mathbb{E}[(\hat{Y} - \mathbb{E}[P])^2] < \epsilon^2$$

with a computational complexity C with bound

$$C \leq \begin{cases} c_4\epsilon^{-2}, & \beta > 1, \\ c_4\epsilon^{-2}(\log\epsilon)^2, & \beta = 1, \\ c_4\epsilon^{-2-(1-\beta)/\alpha}, & 0 < \beta < 1 \end{cases}$$

Proof. See Giles [2008a]. □

Giles [2008a] also shows that the optimal path number for each level M_l is

$$M_l = \lceil 2\epsilon^{-2}\sqrt{\mathbb{V}_l h_l}\Big(\sum_{l=0}^{L}\sqrt{\mathbb{V}_l/h_l}\Big)\rceil, \tag{3.4}$$

while in the computation, $\mathbb{V}_l$ in (3.4) is substituted by $\mathcal{C}_l$ which is the computational cost at each level. For a Euler discretisation with Lipschitz payoff, as $l \to \infty$, we have

$$\mathbb{E}[P - \hat{P}_l] \approx c_1 h_l,$$

and therefore,

$$\mathbb{E}[P_l - P_{l-1}] \approx (N-1)c_1 h_l \approx (N-1)\mathbb{E}[P - \hat{P}_l]$$

where P denote the real value and $\hat{P}_l$ is the numerical approximation for P on each level. We can use this to add a constrain of the remaining bias approximately, and thus obtain a desired bias less than $\epsilon/\sqrt{2}$. To reach this bias, we increase the upper bound of level L until

$$\max\{N^{-1}|\hat{Y}_{L-1}|, |\hat{Y}_L|\} < \frac{\epsilon}{\sqrt{2}}(N-1). \tag{3.5}$$

The procedure of option pricing using MLMC can be summarized as the following algorithm:

```
set L = 2
```

```
set ϵ = desired accuracy
set M_l = 10^4
for l = 0:L
    estimate V_l with M_l simulations
end
estimate optimal M_l' on each level using (3.4)
if M_l' > M_l
    set M_l = M_l'
    go to line 4
else
    check convergence using (3.5)
    if not converged
        set L = L + 1
        go to line 4
    else
        finish
```

Listing 3.1: Algorithm for MLMC option pricing

3.3 Using Stratified Sampling in MLMC

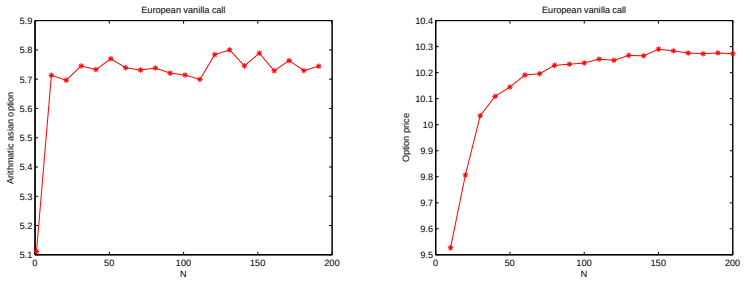
3.3.1 Analysis of Plain MLMC

Note that in the coarsest path P_0 of the multilevel paths, the timestep $h_0 = N^0 T = T$. Although the simulation P_0 is computationally cheap, the big timestep will greatly affect the output results, especially for the path-dependent options. Moreover, recall the MLMC estimator (3.2), the coarsest path P_0 contribute most of the variance because $\mathbb{V}[P_l - P_{l-1}]$ is small and will decrease as level l increases.

Example

Consider an arithmetic Asian option shown in (2.4), where $S(t)$ follows GBM with

parameters $r = 0.05, \sigma = 0.2, S(0) = 100, T = 1$. Figure 3.1a illustrate the fact that



(a) Asian option (b) European call option

Figure 3.1: Price of option under different timestep

timestep N greatly affect the results, the price is lower for approximately 20% when $N = 50$ contrast to $N = 200$.

Now consider a simple *European Vanilla call option* with discounted payoff $e^{-rT}\{S(T) - K\}^+$. Figure 3.1b shows the similar result as the case of Asian option, although the impact of N is slighter contrast to Asian option. Figure 3.2 shows the variance of P_l and $P_l - P_{l-1}$ under different level. It proves our analysis that $\mathbb{V}[P_0]$ is the main source of the total variance of MLMC estimator.

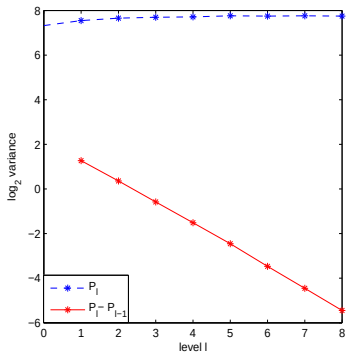


Figure 3.2: Variance of P_l and $P_l - P_{l-1}$

3.3.2 Stratified Sampling

To cope with the problems mentioned in section 3.3.1, we here introduce another variance reduction technique—stratified sampling. The main idea of stratified sampling is to constrain the fraction of samples to be drawn from pre-specified strata of the observations.

To be specific, suppose that we want to estimate $\mathbb{E}[Y]$ where Y is a random variable. Let X be the second random variable, which is valued in $\Re^d$, and we call this *stratification variable*. Divide $\Re^d$ in to k non-overlapping intervals $\Delta_1, \Delta_2, \ldots, \Delta_k$ such that $p_i := \mathbb{P}(X \in \Delta_i)$ and $\mathbb{P}(X \in \cup_i \Delta_i) = 1$. Then the generalised representation of stratified sampling estimator is described as

$$\mathbb{E}[Y] = \sum_{i=1}^{k} \mathbb{P}(X \in \Delta_i)\mathbb{E}[Y|X \in \Delta_i] = \sum_{i=1}^{k} p_i \mathbb{E}[Y|X \in \Delta_i] \tag{3.6}$$

Suppose that the total sample size is M, let m_i be the number of samples to be drawn from stratum Δ_i, such that $m_1 + m_2 + \ldots + m_k = M$. Y_{ij} denotes the independent draws from conditional distribution of Y given that $Y \in \Delta_i$ on strata $i = 1, 2, \ldots, k$, where $j = 1, 2, \ldots, m_k$.Then the unbiased estimator $\hat{Y}$ for $\mathbb{E}[Y]$ shown in (3.6) can be written as

$$\hat{Y} = \sum_{i=1}^{k} p_i \frac{1}{m_i} \sum_{j=1}^{m_i} Y_{ij}. \tag{3.7}$$

Substitute $q_i = m_i/m$ into (3.7), where q_i denote the fraction of observations drawn from strata i, and we have

$$\hat{Y} = \frac{1}{m} \sum_{i=1}^{k} \frac{p_i}{q_i} \sum_{j=1}^{m_i} Y_{ij}. \tag{3.8}$$

To show the effect of the variance reduction brought by stratified sampling, we consider a simple case where $m_i = Mp_i$–the proportional allocation of m_i. Thus the

variance of the estimator shown in 3.7

$$\begin{aligned}\mathbb{V}[\hat{Y}] =& \mathbb{V}[\sum_{i=1}^{k} p_i \frac{\sigma_i^2}{m_i}] \\ =& \sum_{i=1}^{k} p_i^2 \frac{\sigma_i^2}{m_i} \\ =& \frac{1}{M} \sum_{i=1}^{k=1} p_i \sigma_i^2,\end{aligned}$$

where $\sigma_i^2 = \mathbb{V}[Y|X \in \Delta_i]$. Recall the conditional variance formula which states

$$\mathbb{V}[Y] = \mathbb{E}[\mathbb{V}[Y|X]] + \mathbb{V}[\mathbb{E}[Y|X]]. \tag{3.9}$$

Each term in the right-hand-side of (3.9) is non-negative, which implies

$$\begin{aligned}\mathbb{V}[Y] \geq& \mathbb{E}[\mathbb{V}[Y|X]] \\ =& \frac{1}{M} \sum_{i=1}^{k} \mathbb{V}[Y|X \in \Delta_i]. \\ =& \frac{1}{M} \sum_{i=1}^{k=1} p_i \sigma_i^2 = \mathbb{V}[\hat{Y}]\end{aligned}$$

Therefore, the stratified sampling estimator has a lower variance. By minimizing the variance of the estimator by choosing optimal allocation, we can get a better result contrast to the simple case mentioned above.

3.4 Terminal Stratification in MLMC

Due to the fact that coarsest level contributes the most variance on total, we can implement stratified sampling on level $l = 0$, and hence to further reduce the overall variance of MLMC estimator. We here introduce the stratification strategy that is called terminal stratification. Usually, The final price of an underlying asset is critical in option pricing because in most case the payoff is relevant to the prices of

the underlying assets at expiration. Hence we can stratified the terminal values to eliminate the variability related to them. For an asset described by geometric Brownian motion (2.1), to stratify the final values of the asset is equivalent to stratify the terminal value of the Wiener process.

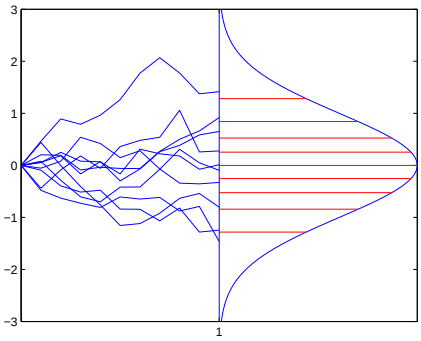


Figure 3.3: Terminal stratification of 10 Wiener paths

Suppose the Wiener path we generated is $W(t_1), W(t_2), \ldots, W(t_n)$ and our aim is to stratify the terminal value $W(t_n)$, i.e. $W(T)$. We divide $\Re^d$ into k equiprobable strata and the proportional allocation. Set

$$V_i = \frac{i-1}{k} + \frac{U_i}{K}, \quad i = 1, 2, \ldots, k$$

where $U_1, U_2, \ldots, U_k$ are the independent random variables drawn from standard uniform distribution $U(0,1)$. Then $\Phi^{-1}(V_1), \Phi^{-1}(V_2), \ldots, \Phi^{-1}(V_k)$ form the stratified samples from $\mathcal{N}(0,1)$. Similarly, we can form the stratification of distribution of $W_i(T)$ by $\sqrt{T}\Phi^{-1}(V_1), \sqrt{T}\Phi^{-1}(V_2), \ldots, \sqrt{T}\Phi^{-1}(V_k)$. We now get the terminal values of the Wiener paths $W_i(T) = \sqrt{T}\Phi^{-1}(V_i)$ as well as the initial value $W(0) = 0$, we can then use Brownian bridge construction to fill in the remaining Wiener paths. Generating k Wiener paths stratified along $W(T)$ can be accomplished by the following algorithm:

```
for i = 1,2,...,k
    set W_i(0) = 0
```

```
    generate  U_i ~ U(0,1)
    set  V_i = (i − 1 + U_i)/k
    set  W_i(T) = √T Φ^{-1}(V_i)
    for  j = 1,2,...,N-1
        generate  Z_j ~ N(0,1)
        set  W_i(t_j) = ((t_N − t_j)/(t_N − t_{j-1})) W_i(t_j − 1) + ((t_j − t_{j-1})/(t_N − t_{j-1})) W(T) + √(((t_N − t_j)(t_j − t_{j-1}))/(t_N − t_{j-1})) Z_j
    end
end
```

Listing 3.2: Algorithm for stratifing Wiener paths

Combining Listing 3.2, 3.1 and 2.1, it is then natural to derive the algorithm of using terminal stratification in multilevel Monte Carlo simulation. Figure 3.3 shows 10 Wiener paths generated using terminal stratification. The 10 strata are equiprobable under the distribution of $W(T)$.

Note that when using terminal stratification, the larger the total strata number k, the lower the variance produced. In practice, we usually set $k = M$ to get the best variance reduction effect possible.

Chapter 4

Programme Design

4.1 Introduction

We implement the multilevel Monte Carlo simulation using terminal stratification algorithms mentioned in previous chapters in C++. This chapter mainly discuss the structure, the design pattern, and the implementation of main algorithms of our programme.

The structure of our programme is shown in Figure 4.1. `IO_Manager` is registered in `IOFactoryRegistration` and is created by `IOFactory`. Similarly, `ConfigurationManager` is registered in `ConfigurationFactoryRegistration` and is created in `ConfigurationFactory`.

The other factory mentioned in Figure 4.1 are actually the same template factory, we change the name in the figure just for better illustrating the structure of our programme.

`IO_Manager`

The `IO_Manager` is responsible for the IO of our application. It is set up with `InputManager` and `OutputManager` in `Application_Wrapper`. It is used by `Factory` to supply object identifier to identify `char`s. It is also used by application

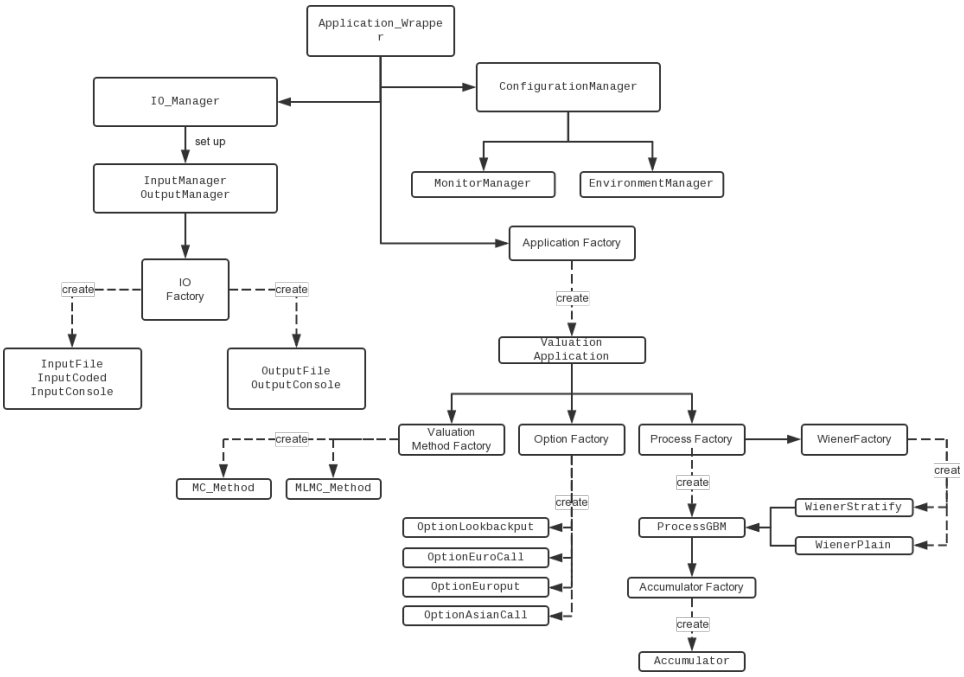


Figure 4.1: Programme structure

objects to supply the parameter values.

`ConfigurationManager`

The `ConfigurationManager` is a singleton, which is accessible from anywhere in the programme. It is set up with `MonitorManager` and `EnvironmentManager` in `Application_Wrapper`. It is used in:

- `IOFactory`, to specify IO types
- IO objects, to supply the IO parameters
- Other objects that need `MonitorManager`

`ProcessGBM`

`ProcessGBM` is where we generate our path. We need to call the Wiener factory to decide which Wiener path method to use.

4.2 Design Patterns

4.2.1 Template Factory Pattern

We use the template factory pattern based on Alexandrescu [2001].

Object Creation with `Factory`

Take an example in `ValuationApplication`, we create the `OptionBase` object from the `Factory` as follows:

```
opt_ = Factory<OptionBase>::Instance().CreateObject();
```

The template factory here is parameterized by the base class, thus different classes can use the identical `Factory`. The advantage is that we do not need to make any modification to the `Factory` and the registration procedure if new base classes are introduced.

The code of template class definition of `Factory` is shown in Appendix List 1.

Object Registration with `FactoryRegistration`

Registration is simple to implement as well, take `OptionEuroCall` for example:

```
namespace
{
    FactoryRegistration<OptionBase, OptionEuroCall>
   RegisterOptionEuroCall;
}
```

Classes are registered in their anonymous namespace. The code of template class definition of `FactoryRegistration` is shown in Appendix List 2.

4.2.2 Singleton Pattern

The `Factory<OptionBase>` shown in section 4.2.1 is a singleton and is accessed by its `Instance()` method. Singleton ensures that only one instance of a class ex-

ists. It is useful for base class such as `OptionBase` where only one object is need to coordinate actions. The constructors, default constructors, and copy constructors in this case are private so that the objects can not be instantiated in any other way. Moreover, singleton can be made available globally in a programme which is effective on a project scope.

4.3 Main Algorithms

4.3.1 Terminal Stratification

The terminal stratification is realized using the following code:

```
void WienerStratify::FillUpPath(VecB & path, const long M)
{
    if (j_ == M) j_ = 0;

    long lb = path.LBound();
    long ub = path.UBound();
    double dt = T_ / (ub - lb);

    path[lb] = z0_;

    double u = rv::my_ran2();
    double v = (j_ + u)/M;          //stratified at terminal
    path[ub] = rv::cndev(v);
    ++j_;

    long i = long(std::floor(0.5*(ub - lb + 1)));
    Get_Brownian_bridge(path, dt, lb, i, ub);
}
```

where `path.LBound()` and `path.UBound()` returns the lower bound and the upper bound indexing of `path` respectively. `z0_` here is set to be 0, and so that $W(0) = 0$. `rv::cndev(v)` return the value of $\Phi^{-1}(v)$.

The code is straight forward to understand, it implement the algorithm mentioned in 3.4. The function `Get_Brownian_bridge()` is a private function, which calculate the value of elements in Wiener path between $W(0)$ and $W(T)$ by the Brownian bridge construction. The code of `Get_Brownian_bridge()` is listed in Appendix List 3

4.3.2 Path Generation

The path generation function in our programme is coded in the process class because for different stochastic process the path generation method varies. For a geometric Brownian motion, the code is as follows:

```
void ProcessGBM::FillUpPath(std::vector<double> & Spath, const long M)
    const
{

    long N = Spath.size() - 1;
    double dt = T_ / N;
    Spath[0] = S_0_;

    VecB Wpath(0, N);
    wie_->FillUpPath(Wpath, M);

    for(long i = 1;  i <= N; ++i)        // for each time step
    {
        Spath[i] = Euler(Spath[i-1], dt, Wpath[i] - Wpath[i-1]);
    }
```

```
}
```

where `VecB` is an array class that supports arbitrary upper and lower indexing bounds, `Euler` is the discretisation method to be called. We also provide Milstein scheme and exact method for GBM. This is the path generation for plain Monte Carlo methods. For multilevel Monte Carlo methods, there are some difference. One of the main difference is the input parameters. In MLMC, we need to generate two paths simultaneously with the same Wiener path. The level l and the refinement factor should be the input parameters too. The code for path generation in MLMC is listed in Appendix 4.

Chapter 5

Numerical Results

5.1 Introduction

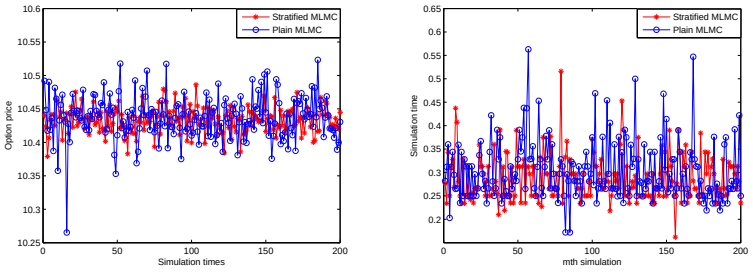
In this chapter, numerical results for European call option, arithmetic Asian option and lookback put option are presented. All the options are priced under geometric Brownian motion model. All the simulations in this chapter use the same parameter values $S(0) = 100, r = 0.05, \sigma = 0.2, T = 1, K = 100$, except for the section that simulate under different strike price K.

5.2 European Options

5.2.1 Simulations under Different ϵ

Here we provide some numerical results of the effect of using stratified sampling in MLMC. We first consider simple European vanilla call option with the same payoff mentioned in section 2.3.4 with parameters $r = 0.05, \sigma = 0.2, S(0) = 100, K = 100, T = 1$.

We run the programme for 200 times and the environment during running are kept same, and set the initial path number of MLMC $M_0 = 10000$. Figure 5.1a



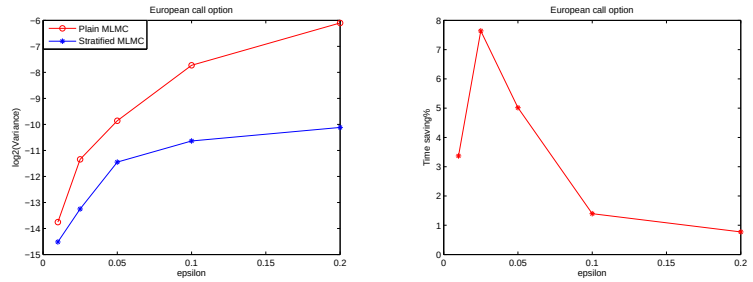
(a) Option prices (b) Time taken

Figure 5.1: Stratified MLMC vs MLMC for European option

shows the fact that the prices of options enjoys a lower variance when using stratified sampling. The accuracy for this test is $\epsilon = 0.05$;

The variance of option price when using stratified sampling is 3.5824×10^{-4}, and that using plain sampling is 1.0778×10^{-3}. The ratio between the two variance is approximately 3.

The simulation time is reduced as well, as we can see from Figure 5.1b. The average time taken for stratified sampling MLMC is $0.2875s$, and that for plain MLMC is $0.3027s$. We obtain a efficiency enhancement of approximately 5%. This is due to the fact that $\mathbb{V}_0$ is reduced and thus the optimal path number $M_l^{'}$ is reduced as well. Thus less computational effort is needed for the simulation to reach the desired accuracy.



(a) Variance (b) Computational cost reduction

Figure 5.2: Stratified MLMC vs MLMC under different ϵ for European option

We also do the test under $\epsilon = \{0.2, 0.1, 0.05, 0.024, 0.01\}$. Figure 5.2a and 5.2b

shows that the effect of using terminal stratification in MLMC varies with ϵ. For higher ϵ, stratified MLMC has a higher variance reduction. The reason is that higher ϵ means lower accuracy, and thus less computational efforts are spent on reducing the variance of plain MLMC. Moreover, the effect of using stratified sampling at $l = 0$ is not affected by the value ϵ, or namely, it will provide similar results under different ϵ. Therefore, when ϵ is getting smaller, more computational efforts are spent on increasing the accuracy. And hence the difference between the variance of plain MLMC and stratified MLMC becomes smaller. Despite that, stratified MLMC still provide half variance contrast to plain MLMC when $\epsilon = 0.01$. Table 5.1 illustrates the detailed variance when using plain and stratified MLMC under different values of ϵ. The variance ratios in the table is the ratio between plain MLMC and stratified MLMC and we round them to the nearest integers.

	Plain MLMC		Stratified MLMC		Variance
ϵ	Variance	Time	Variance	Time	Ratio
0.2	0.014562	0.1398s	0.0009	0.1388s	16
0.1	0.003908	0.1588s	0.000629	0.1566s	6
0.05	0.001078	0.3027s	0.000358	0.2875s	3
0.025	0.000385	0.9769s	0.000103	0.9022s	4
0.01	0.000072	6.2761s	0.000043	6.0647s	2

Table 5.1: Stratified MLMC vs MLMC under different ϵ for European option

5.2.2 Simulations under Different Strike Prices

We also find the that the variance ratio is affected by the strike price K. Figure 5.3 shows that given $\epsilon = 0.1$ and all the other parameters remains the same, the variance ratio decreases as the strike price increases. In other word, stratified sampling using in MLMC have a better effect if the European vanilla option is deep in-the-money.

Table 5.2 illustrates the variance ratio when using plain MLMC and stratified

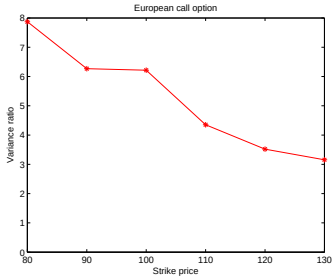


Figure 5.3: Variance ratio under different K for European option

MLMC.

K	Variance of Plain MLMC	Variance of Stratified MLMC	Variance Ratio
80	0.003657	0.000465	8
90	0.003866	0.000617	6
100	0.003908	0.000629	6
110	0.003982	0.000915	4
120	0.003418	0.000971	4
130	0.002593	0.000822	3

Table 5.2: Stratified MLMC vs MLMC under different K for European option

5.2.3 Simulations under Different Initial Path Numbers

Recall section 3.4 that we can obtain better variance reduction result if we set more strata. As we mentioned in 3.2 that we need to set a initial path number M_0 when implementing multilevel Monte Carlo methods. Therefore, we can set a larger value of M_0 to obtain a further variance reduction.

Given that $\epsilon = 0.1, K = 100$ and the remaining parameters to be the same. The results shown in Figure 5.4 proves our analysis. The variance ratio is almost doubled when $M_0 = 20000$ contrast to $M_0 = 1000$. We also find that the variance ration remains the same when M_0 is increased from 20000 to 30000. This is due to the fact that the variance reduction brought by stratified sampling will converge as the path number M increases. To be specific, as the strata k increases, the

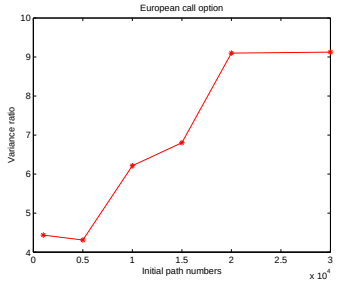


Figure 5.4: Variance ratio under different M_0 for European option

equiprobable subintervals becomes smaller, so as the variability of underlying price reaching each strata.

Results in Table 5.3 shows that when $M_0 = 20000$, the variance using terminal stratification is close in value compared to the case when $M_0 = 10000, \epsilon = 0.05$. But the average running time in this case is only $0.1942s$, much less than $0.2875s$ shown in Table 5.1. Hence we have found another way to improve the efficiency of MLMC using terminal stratification, which is to increase the initial path number M_0.

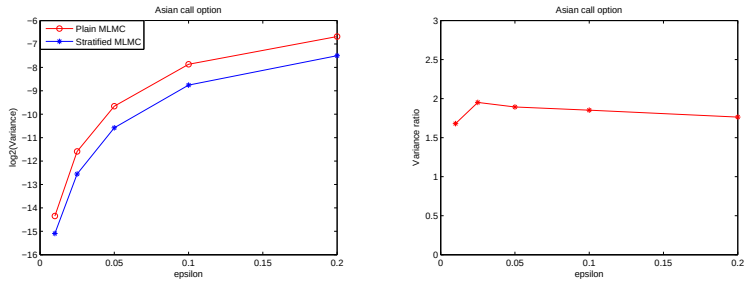
	Plain MLMC	Sratified MLMC		Variance
M_0	Variance	Variance	Time	Ratio
1000	0.004779	0.000968	0.1835s	5
5000	0.004321	0.000907	0.1561s	5
10000	0.003708	0.000629	0.1566s	6
15000	0.003982	0.000915	0.1877s	7
20000	0.003693	0.000389	0.1942s	9
30000	0.003426	0.000384	0.2286s	9

Table 5.3: Stratified MLMC vs MLMC under different M_0 for European option

5.3 Arithmetic Asian Options

5.3.1 Simulations under Different ϵ

Now consider an arithmetic Asian call option with payoff (2.2). All other parameters are set to be same as section 5.2.



(a) Variance (b) Variance ratio

Figure 5.5: Variance under different ϵ for Asian option

Figure 5.5a shows that stratified MLMC still has an advantage over the plain MLMC under different values of ϵ, but the variance reduction brought by terminal stratification is not as effective as the case of European option. It is straightforward to note the fact that the payoff of Asian option is not completely determined by the terminal underlying asset prices. Thus the impact of using terminal stratification is reduced. Figure 5.5b shows that the variance ratios dose change much under various values of ϵ. Table 5.1 illustrates the detailed data.

ϵ	Variance of Plain MLMC	Variance of Stratified MLMC	Variance Ratio
0.2	0.009743	0.005526	2
0.1	0.004281	0.002312	2
0.05	0.001238	0.000654	2
0.025	0.000324	0.000166	2
0.01	0.000048	0.000028	2

Table 5.4: Stratified MLMC vs MLMC under different K for Asian option

5.3.2 Simulations under Different Strike Prices

We now consider the simulations under different strike prices K. Again, we set $\epsilon = 0.1$ and keep other parameters to be the same. Figure 5.6 further proves our previous statement that terminal stratification has less influence on variance reduc-

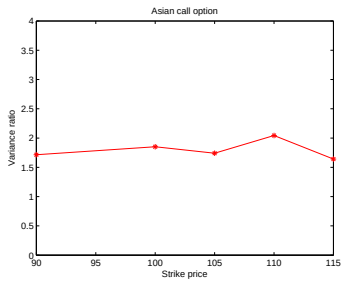


Figure 5.6: Variance ratio under different K for Asian option

tion compared with European option. The variance ratio are almost equivalent when K ranges from 90 to 115. Despite that, Table 5.5 shows that stratified sampling still lead to a variance reduction of 50%.

ϵ	Variance of Plain MLMC	Variance of Stratified MLMC	Variance Ratio
90	0.004287	0.002500	2
100	0.004281	0.002312	2
105	0.002029	0.003534	2
110	0.003091	0.001513	2
115	0.002385	0.001453	2

Table 5.5: Stratified MLMC vs MLMC under different K for Asian option

5.3.3 Simulations under Different Initial Path Numbers

In this case, we choose $\epsilon = 0.1$ and set $K = 100$. Consider the simulations under different initial path numbers.

Figure 5.7 shows that the variance ratio under lower M_0 is higher. We can see in Table 5.6 that the variance of stratified MLMC is of the same order of magnitude under different M_0. While the variance of plain MLMC keeps going down as the initial path number increased.

This probably explain the reason for the decreasing variance ratio as M_0 rises. The initial path number dose not affect the variance when using terminal stratification

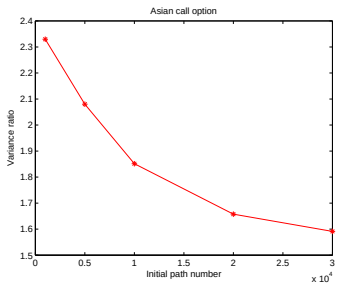


Figure 5.7: Variance ratio under different M_0 for Asian option

M_0	Variance of Plain MLMC	Variance of Stratified MLMC	Variance Ratio
1000	0.005078	0.002180	2
5000	0.004157	0.001999	2
10000	0.004281	0.002312	2
20000	0.003601	0.002173	2
30000	0.003543	0.002227	2

Table 5.6: Stratified MLMC vs MLMC under different M_0 for Asian option

for arithmetic Asian option.

5.4 Lookback options

Now consider a lookback put option with floating strike defined in (2.8).

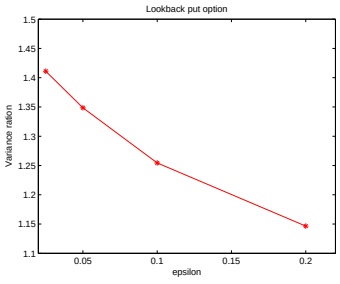


Figure 5.8: Variance ratio under different ϵ for Lookback option

Figure 5.8 and Table 5.7 show that the variance reduction ratio is not so significant under different ϵ. But stratified MLMC still generate less variance in simulation

compared with plain MLMC. Moreover, the variance ratio rises as smaller ϵ is set.

ϵ	Variance of Plain MLMC	Variance of Stratified MLMC	Variance Ratio
0.2	0.021895	0.019096	1
0.1	0.005675	0.004524	1
0.05	0.001491	0.001105	1
0.025	0.000352	0.000249	1

Table 5.7: Stratified MLMC vs MLMC under different K for Lookback option

Chapter 6

Conclusions

6.1 Summary of Results

This work concerns the application of multilevel Monte Carlo (MLMC) methods in option pricing problems. We consider the combination of MLMC and terminal stratification.

The numerical results in Chapter 5 show that terminal stratification will lead to a significant variance reduction for European options. We also shows that stratified MLMC is more effective when the option is deep in-the-money. As shown in section 5.2.3, one can obtain further efficiency enhancement by adding the initial path number M_0 when using stratified MLMC. For path-dependent options, the influence of using terminal stratification is not that significant because the payoff of such options are not totally decided by the terminal underlying asset prices. Results shows that terminal stratification can still have a variance reduction of 50% for arithmetic Asian options. While for lookback options, stratified MLMC has less advantage on plain MLMC compared with European options and Asian options.

We programme the whole procedure of option pricing using stratified MLMC, plain MLMC and plain Monte Carlo methods in C++. The programme uses template factory design pattern and enjoys a high maintainability and expandability.

6.2 Future Work

We name some potential directions for future work:

It is possible to investigate other stratified sampling strategies for different options. For example, it may be more effective to stratified the average rather than the terminal values when pricing Asian options.

Another direction may be the theoretical analysis on the algorithms mentioned in this thesis.

Appendix

```
1  #include <string>
2  #include <map>
3  template<typename Base_class> class Factory
4  {
5      public:
6          // ... Singleton stuff omitted
7          typedef Base_class * (*DerivedClassCreator)();
8          typedef std::pair<std::string, DerivedClassCreator> CreatePair;
9          typedef std::map<std::string, DerivedClassCreator> CreateMap;
10         typedef std::pair<std::string, std::string> LookUpPair;
11         typedef std::map<std::string, std::string> LookUpMap;
12         void RegisterObject
13             (const std::string \&, const std::string \&,
     DerivedClassCreator);
14         Base\_class * CreateObject();
15     private:
16         // ... Singleton stuff omitted
17         CreateMap CreateObjectsMap\_;
18         LookUpMap IDLookUpMap\_;
19 };
20 #include "Factory.cpp"
```

Listing 1: Template definition of `Factory`

```
#include <string>
#include <iostream>
#include "Factory.h"
#include "CreateableBase.h"

template<typename Base_class, typename Derived_class>
FactoryRegistration<Base_class, Derived_class>::FactoryRegistration()
{
    Factory<Base_class>::Instance().RegisterObject
    (
        ClassID<Derived_class>(),
        ClassName<Derived_class>(),
        FactoryRegistration<Base_class, Derived_class>::Create
    );
    }

template<typename Base_class, typename Derived_class>
Base_class * FactoryRegistration<Base_class, Derived_class>::Create()
{
    return new Derived_class;
}
```

Listing 2: Template definition of `FactoryRegistration`

```
WienerStratify::Get\_Brownian\_bridge(VecB & Wpath, double dt, long i,
    long j, long N)
{
    if (i == j || j == N) return;

    double c_1 = (N - j) / double(N - i);
    double c_2 = (j - i) / double(N - i);
    double c_3 = std::sqrt(dt * (N - j) * (j - i) / double(N - i));

    double z = rv::cndev(rv::my_ran2());

    Wpath[j] = c\_1 * Wpath[i] + c\_2 * Wpath[N] + c\_3 * z;

    if (j > i + 1) Get_Brownian_bridge(Wpath, dt, i, long(std::floor(0.5
    + 0.5*(i + j))), j);
    if (j < N - 1) Get_Brownian_bridge(Wpath, dt, j, long(std::floor(0.5
    + 0.5*(j + N))), N);
}
```

Listing 3: Function `Get_Brownian_bridge()`

```
void ProcessGBM::FillUpPath(std::vector<double> & path_fine, std::vector<
    double> & path_coarse, const double Ref_factor, const long l, const
    long M) const
{

    long N_coarse = path_coarse.size() - 1;
    long N_fine   = path_fine.size() - 1;

    double dt_coarse = T_ / N_coarse;
    double dt_fine   = T_ / N_fine;

    double rt_fine = std::sqrt(dt_fine);

    path_fine[0]   = S_0_;
    path_coarse[0] = S_0_;

    if (l == 0)
    {
        VecB Wpath(0, 1);
        wie_->FillUpPath(Wpath, M);

        double dt_fine = T_;

        double dW_fine = Wpath[1] - Wpath[0];      // Wiener increment
    for fine path

        path_fine[1] = Euler(path_fine[0], dt_fine, dW_fine);

        path_coarse[0] = 0;
    }

    else
    {
```

```
31          for (long i = 01; i <= N_coarse; ++i)  // for each time step
32          {
33              double dW_coarse = 0;
              // Wiener increment for coarse path
34
35              for (long j = 01; j <= Ref_factor; ++j)
36              {
37                  double w = rv::GetNormalVariate();
38
39                  double dW_fine = w*rt_fine;
              // Wiener increment for fine path
40
41                  dW_coarse += dW_fine;
42
43                  long col = (i - 1)*Ref_factor + j;
              // the column of path to fill
44
45                  path_fine[col] = Euler(path_fine[col - 1], dt_fine,
    dW_fine);
46              }
47
48              path_coarse[i] = Euler(path_coarse[i - 1], dt_coarse,
    dW_coarse);
49          }
50      }
51  }
```

Listing 4: Function `FillUpPath()` for MLMC

References

A. Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley, 2001. 30

J.C. Cox, J.E. Ingersoll, and S.A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53:385–407, 1985. 7

M.B. Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3): 607–617, 2008a. iii, 1, 3, 17, 20, 21

M.B. Giles. Improved multilevel monte carlo convergence using the milstein scheme. *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 343–358, 2008b. 1

M.B. Giles and B.J. Waterhouse. Multilevel quasi-monte carlo path simulation. *Radon Series on Computational and Applied Mathematics*, 8:1–18, 2009. 1

P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004. 13

P.W. Glynn and W. Whitt. The asymptotic efficiency of simulation estimators. *Operations Research*, 40(3):505–520, 1992. 12

S.L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2): 327–343, 1993. 7

H. Hoel, E.V. Schwerin, A. Szepessy, and R. Tempone. Adaptive multilevel monte

carlo simulation. *Numerical Analysis of Multiscale Computations*, 82:217–234, 2012. 1

J.C. Hull and A. White. Pricing interest-rate derivative securities. *The Review of Financial Studies*, 3(4):573–592, 1990. 7

P. L'Ecuyer. Quasi-monte carlo methods in finance. *Winter Simulation Conference*, pages 1645–1655, 2004. 13

S.E. Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer, 2010. 10

O. Vasicek. An equilibrium characterization of the term structure. 7