A Review of Multilevel Monte Carlo Methods

Rohin Jain

Supervisor:
Thomas McWalter

A dissertation submitted to the Faculty of Commerce, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Philosophy.

December 19, 2019

MPhil in Mathematical Finance,
University of Cape Town.

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy in the University of the Cape Town. It has not been submitted before for any degree or examination in any other University.

December 19, 2019

Abstract

The Monte Carlo method (MC) is a common numerical technique used to approximate an expectation that does not have an analytical solution. For certain problems, MC can be inefficient. Many techniques exist to improve the efficiency of MC methods. The Multilevel Monte Carlo (ML) technique developed Giles (2008) is one such method. It relies on approximating the payoff at different levels of accuracy and using a telescoping sum of these approximations to compute the ML estimator. This dissertation summarises the ML technique and its implementation. To start with, the framework is applied to a European call option. Results show that the efficiency of the method is up to 13 times faster than crude MC. Then an American put option is priced within the ML framework using two pricing methods. The Least Squares Monte Carlo method (LSM) estimates an optimal exercise strategy at finitely many instances, and consequently a lower bound price for the option. The dual method finds an optimal martingale, and consequently an upper bound for the price. Although the pricing results are quite cloto the corresponding crude MC method, the efficiency produces mixed results. The LSM method performs poorly within an ML framework, while the dual approach is enhanced.

Acknowledgements

I would like to thank God for the opportunity and strength to study this degree. His unfailing strength has helped me produce this dissertation.

A big thank you must go to my supervisor, Thomas McWalter, who spent countless hours helping me and providing me guidance on how to approach the problems I faced.

Lastly, I would like to thank my friends and family who believed in me, even when I did not believe in myself. Your support was invaluable to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computing expectations is a key component of many Mathematical Finance problems. In pricing financial instruments, this expectation comprises a discounted payoff function conditioned on the information available today. Some expectations have a closed-form, analytical solution. However, most times, there may not be such a 'neat' solution.

In these instances, numerical techniques can estimate these expectations. Common numerical techniques include Monte Carlo (MC) methods, finite difference approaches and Fourier transformations. MC methods have an advantage in that the order of the error term is independent of the dimension of the problem. This makes MC methods very popular for solving higher-dimensional problems.

Unfortunately, MC methods have poorer accuracy relative to other methods when dealing with lower-dimensional problems. A simple way to improve accuracy is to increase the number of samples used. However, this increases the computational burden. As a result, many methods have been developed to ease this problem. Antithetic sampling, stratified sampling, importance sampling and the use of control variates are all means to improve the efficiency of MC estimates. In recent times, Multilevel Monte Carlo (ML) methods have also become a popular technique.

Multilevel Monte Carlo was first introduced by Heinrich (2001) where it was used as an approximation for high-dimensional parametric integrals. Giles (2008) tailored the method to approximate expectations in Mathematical Finance. Since its introduction, the applications of ML methods have dramatically increased. Belomestny *et al.* (2013) uses the ML technique to price American style derivatives. We can also compute the Greeks of an option using the ML framework (Burgos and Giles, 2012). Jump-diffusion SDEs can be simulated using ML techniques as shown in Xia and Giles (2012). We can even combine ML methods with quasi-Monte Carlo techniques to further enhance the efficiency (Giles and Waterhouse, 2009).

A key idea in ML is to approximate the payoff function at different levels of

accuracy. The less accurate levels are known as the coarse levels, while the more accurate levels are known as the fine levels. Simulating from the fine levels leads to better answers, but is more time-consuming. By constructing the ML estimator carefully, this method can ensure that the bulk of the simulations are from the coarse level while maintaining the accuracy of the finest level. This results in the ML estimator being more efficient than the corresponding MC estimator for the same level of accuracy.

It is not always possible to construct the ML estimator in this manner. Luckily, the complexity theorem describes situations where the ML estimator will be more efficient than the MC estimator. Unfortunately, proving the complexity theorem in non-trivial problems is tricky. In these scenarios, convergence plots are used to determine if the complexity theorem is satisfied.

This dissertation summarises the ML method and applies it to various pricing techniques. Chapter 2 outlines the framework used for MC and ML methods. It describes the ML approach and includes an error analysis for both methods. It also details the ML Algorithm. Chapter 3 applies the ML framework to a European call option and solidifies the theory from the previous chapter. Chapter 4 summarises the American pricing problem and outlines two methods of pricing an American put option. The ML framework is then applied to both these methods and the results are discussed. Finally, conclusions are presented in Chapter 5.

Chapter 2

Framework for Multilevel Monte Carlo

Monte Carlo methods can approximate an expectation of a discounted payoff function. The payoff, P, is usually a function of the underlying asset process. Assume a probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$, where $\mathcal{F}$ satisfies the usual conditions. The asset process, $\{X_t\}_{0\leq t\leq T}$, is commonly modelled as a solution to the following stochastic differential equation (SDE):

$$\mathrm{d}X_t = a(t, X_t)\mathrm{d}t + b(t, X_t)\mathrm{d}W_t \qquad X_0 = x\,, \tag{2.1}$$

where $a(t, X_t)$ and $b(t, X_t)$ satisfy some integrability conditions.

Under the equivalent risk-neutral measure, $\mathbb{Q}$, the time-0 price of a financial instrument with a payoff, P, is

$$e^{-rT}\mathbb{E}_{\mathbb{Q}}\Big[P\big(X(\omega)\big)\Big], \tag{2.2}$$

where r is the constant short rate and $X(\omega)$ is a sample path of the asset process, $\{X_t\}_{0\leq t\leq T}$. From hereon, assume that all expectations are under the risk-neutral measure.

Sections 2.1 and 2.2 outline how to estimate the expectation in (2.2). This is followed by an error analysis which contains the theorem that identifies the instances where ML is more efficient than MC. Finally, Section 2.4 outlines the ML algorithm and numerical implementation.

2.1 Crude Monte Carlo Estimation

Monte Carlo estimates rely on the Strong Law of Large Numbers (SLLN) which states:

Theorem 2.1. *If $Y_1, ..., Y_n$ are independent and identically distributed (iid) random variables with $\mathbb{E}[Y_i] = \mu$, then $\hat{Y}_n := \frac{1}{n}\sum_{i=1}^{n} Y_i$ converges to the expected value. That is:*

$$\hat{Y}_n \xrightarrow{\text{a.s.}} \mu \quad as \quad n \to \infty \,.$$

To compute the MC estimate, we must simulate n independent sample paths of $\{X_t\}_{0\le t\le T}$ and calculate the corresponding payoff values which will also be independent. Choosing the arithmetic mean as the estimator for MC and applying the SLLN yields

$$\hat{Y} := \frac{1}{n}\sum_{i=1}^{n} P\big(X(\omega^{(i)})\big) \xrightarrow{\text{a.s.}} \mathbb{E}\Big[P\big(X(\omega)\big)\Big] \quad \text{as} \quad n \to \infty.$$

Therefore, the MC estimate converges to the true value.

Sometimes it may not be possible to directly simulate $P(X(\omega))$. In those instances, an approximation must be used. A common approximation is an Euler-Maruyama estimate with a certain number of time steps. As the number of time steps increases, the path becomes finer and the accuracy of the approximation improves. Let P_L denote an approximation for the payoff with M^L number of steps. The convergence of the MC estimator still holds as we increase L. That is,

$$\hat{Y} := \frac{1}{n}\sum_{i=1}^{n} P\big(X_L(\omega^{(i)})\big) \xrightarrow{n\to\infty} \mathbb{E}\Big[P\big(X_L(\omega)\big)\Big] \xrightarrow{L\to\infty} \mathbb{E}\Big[P\big(X(\omega)\big)\Big].$$

To simplify notation, let $\mathbb{E}[P_L]$ denote $\mathbb{E}\Big[P\big(X_L(\omega)\big)\Big]$ and P_L^i denote $P\big(X_L(\omega^{(i)})\big)$.

2.2 Multilevel Monte Carlo Estimation

In the Multilevel Monte Carlo (ML) approach, Giles (2008) suggests a new estimator that can approximate $\mathbb{E}[P_L]$. The ML approach relies on the following telescoping sum,

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}]. \tag{2.3}$$

The idea is to estimate each of the expectations on the RHS independently and efficiently. Each expectation will be estimated by an MC estimate. That is, each $\mathbb{E}[P_l - P_{l-1}]$ will be estimated by

$$Y_l = \frac{1}{N_l}\sum_{i=1}^{N_l} \left[P_l^i - P_{l-1}^i\right]$$

and $\mathbb{E}[P_0]$ estimated by

$$Y_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} P_0^i.$$

The combined estimator, Y_{ML}, is written as the sum of the estimators,

$$Y_{\text{ML}} = \frac{1}{N_0}\sum_{i=1}^{N_0} P_0^{(i)} + \frac{1}{N_1}\sum_{i=1}^{N_1}\left[P_1^{(i)} - P_0^{(i)}\right] + \ldots + \frac{1}{N_L}\sum_{i=1}^{N_L}\left[P_L^{(i)} - P_{L-1}^{(i)}\right].$$

Each estimator, Y_l, is independent since independent samples are used. Moreover, the number of samples, N_l, used for each Y_l can be different.

The key idea in ML is to construct P_l and P_{l-1} in such a way that there is a large positive correlation between P_l and P_{l-1}. In other words, minimising $\mathbb{V}\text{ar}[P_l - P_{l-1}]$. As a result, fewer samples are needed to estimate $\mathbb{E}[P_l - P_{l-1}]$ to the same level of accuracy. This is important as generating fine paths (i.e., paths with many time steps) are computationally intense.

There are many ways to induce this correlation. A simple way to do this is to re-use the Brownian increments from P_l to construct P_{l-1}. Do this by first creating M^l Brownian increments needed for P_l. Then divide these increments into groups of M. Within each group, sum up the increments to form a Brownian increment for P_{l-1}. These M^{l-1}-many increments will be used to create P_{l-1}. Further details regarding this method can be found in Section 4.4.

Equation (2.3) provides a means to interpret the ML estimator. The level-L approximation of the price, $\mathbb{E}[P_L]$, can be decomposed into a level-0 price, $\mathbb{E}[P_0]$, plus a sum of corrections in price. Each $\mathbb{E}[P_l - P_{l-1}]$ represents the correction in price from level $l-1$ to l. So, if P_l converges quickly to P, then for large l, $P_l - P_{l-1}$ will be close to zero. In those cases, L need not be large to ensure an accurate approximation of P.

2.3 Mean Square Error Analysis

The error in an estimator, $\hat{Y}$, can be analysed using the mean square error,

$$\text{MSE} \equiv \mathbb{E}\left[(\hat{Y} - \mathbb{E}[Y])^2\right], \tag{2.4}$$

where $\mathbb{E}[Y]$ is the expectation that is being estimated, i.e., $\mathbb{E}[P]$. This expression can be manipulated to illustrate the composition of the error due to the variance and

bias in the estimator. Adding and subtracting $\mathbb{E}[\hat{Y}]$ results in,

$$\begin{aligned}\text{MSE} &= \mathbb{E}\left[\left(\left(\hat{Y}-\mathbb{E}[\hat{Y}]\right)+\left(\mathbb{E}[\hat{Y}]-\mathbb{E}[Y]\right)\right)^2\right] \\ &= \mathbb{E}\left[\left(\hat{Y}-\mathbb{E}[\hat{Y}]\right)^2\right]+2\left(\mathbb{E}[\hat{Y}]-\mathbb{E}[Y]\right)\mathbb{E}\left(\hat{Y}-\mathbb{E}[\hat{Y}]\right)+\left(\mathbb{E}[\hat{Y}]-\mathbb{E}[Y]\right)^2 \\ &= \underbrace{\mathbb{E}\left[\left(\hat{Y}-\mathbb{E}[\hat{Y}]\right)^2\right]}_{\text{Variance of Estimator}}+\Bigg(\underbrace{\mathbb{E}[\hat{Y}]-\mathbb{E}[Y]}_{\text{Bias Error}}\Bigg)^2, \end{aligned} \tag{2.5}$$

where the second equality follows due to $\hat{Y}$ being the only random variable in the expression.

The bias of an estimator is partly due to the discretisation error. As the path becomes finer, this bias becomes smaller. Both ML and MC estimators have the same expected value, i.e., $\mathbb{E}[\hat{Y}_{\text{MC}}] = \mathbb{E}[\hat{Y}_{\text{ML}}] = \mathbb{E}[P_L]$. Consequently, each estimator will have the same bias error, $(\mathbb{E}[\hat{Y}] - \mathbb{E}[Y])^2$. So it can be concluded that any differences in the accuracy of estimators arise due to differences in the variances of the estimators.

The bias error will depend on the discretisation scheme. It can be shown that for both Euler-Maruyama and Milstein schemes, the weak order of convergence is one. That is,

$$\left|\mathbb{E}[\hat{Y}]-\mathbb{E}[Y]\right| = \mathcal{O}(h_L), \tag{2.6}$$

where h_L is the size of the timestep on the finest level. As a result, the second term of (2.5) will be $\mathcal{O}(h_L^2)$. If the aim is to keep the $\text{MSE} \leq \epsilon^2$, then we require $h_L = \mathcal{O}(\epsilon)$.

2.3.1 Error Analysis for Crude Monte Carlo

The variance of a MC estimator is expressed as,

$$\mathbb{E}\left[\left(\hat{Y}-\mathbb{E}[\hat{Y}]\right)^2\right] = \mathbb{V}\text{ar}\left[\frac{1}{N}\sum_{i=1}^{N} P_L^i\right] = \frac{1}{N}\mathbb{V}\text{ar}[P_L] = \mathcal{O}\left(\frac{1}{N}\right),$$

where the second equality follows due to P_L^i being independent and identically distributed. If the aim is to keep the $\text{MSE} \leq \epsilon^2$, then we require $N = \mathcal{O}(\epsilon^{-2})$.

Computational complexity, $\mathcal{C}$, can be defined as the product of the number of sample paths and the number of time steps per sample path. Then, the computational complexity is

$$\mathcal{C} = N\frac{T}{h_L} = \mathcal{O}(\epsilon^{-2})\,\mathcal{O}(\epsilon^{-1}) = \mathcal{O}(\epsilon^{-3}).$$

2.3.2 Error Analysis for Multilevel Monte Carlo

In certain circumstances, ML can maintain the MSE accuracy, while reducing the computational complexity. This relies heavily on the complexity theorem, presented below. The proof of this theorem can be found in Giles (2008).

Theorem 2.2. *Let P denote a functional of a solution of the stochastic differential equation, (2.1), for a given Brownian path W_t, and let P_l denote the corresponding approximation using a numerical discretisation with timestep $h_l = \frac{T}{M^l}$.*

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

i. $|\mathbb{E}[P_l - P]| \leq c_1 h_l^{\alpha}$

ii. $\mathbb{E}[\hat{Y}_l] = \begin{cases} \mathbb{E}[P_0] & \text{if} \quad l = 0 \\ \mathbb{E}[P_l - P_{l-1}] & \text{if} \quad l > 0 \end{cases}$

iii. $\mathbb{V}ar[Y_l] \leq c_2 N_l^{-1} h_l^{\beta}$

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 of L and N_l, for which the multilevel estimator,

$$\hat{Y}_{\mathrm{ML}} = \sum_{l=0}^{L} \hat{Y}_l,$$

has a mean square error with bound

$$\mathrm{MSE} \equiv \mathbb{E}\left[\left(\hat{Y}_{\mathrm{ML}} - \mathbb{E}[P]\right)^2\right] < \epsilon^2$$

and computational complexity, C, with bound

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

Condition *(i)* refers to the weak convergence of the estimator. In both the Euler-Maruyama and Milstein schemes, $\alpha = 1 > \frac{1}{2}$. The second condition, *(ii)*, is trivially satisfied by the way the estimator is constructed. Condition *(iv)* is also satisfied since the computational complexity is simply the product of the number of sample

paths, N_l, and the number of time steps, T/h_l. Condition *(iii)* is the trickiest requirement that needs to be satisfied. Moreover, the value of β determines the bound on the computational complexity. Regardless, as long as $\beta > 0$, these bounds are less than the MC complexity bound, $\mathcal{O}(\epsilon^{-3})$.

The complexity theorem guarantees the existence of a better estimator. However, it does not provide any guidance as to the construction of this estimator. Important questions like how to correlate P_l and P_{l-1} while maintaining conditions *(i)* and *(iii)* are not answered. Additionally, the number of samples per level, N_l, and the finest level of approximation, L, need to be determined.

2.4 Multilevel Algorithm

The Multilevel Algorithm sets out a procedure to estimate the optimal number of samples per level, N_l, and the finest level of approximation, L. This algorithm does not guarantee the correct N_l and L will be found, but it does a fairly good job in most cases. The number of samples is determined by minimising the MSE for a fixed level of cost, C. Although the MSE consists of the variance and the bias of the estimator, the variance is the only component that depends on N_l. The variance of the estimator is given by

$$\mathbb{V}\mathrm{ar}[\hat{Y}_{\mathrm{ML}}] = \sum_{l=0}^{L} \frac{V_l}{N_l},$$

where $V_l := \mathbb{V}\mathrm{ar}[P_l - P_{l-1}]$, while the computational cost, $\mathcal{C}$, is calculated as

$$\mathcal{C} = \sum_{l=0}^{L} \frac{N_l}{h_l}.$$

Treating N_l as a continuous variable and performing a Lagrangian optimisation results in

$$\begin{aligned} \mathcal{L} &= \sum_{l=0}^{L} \frac{V_l}{N_l} + \lambda\left(\sum_{l=0}^{L} \frac{N_l}{h_l} - C\right) \\ \frac{\partial \mathcal{L}}{\partial N_l} &= -\frac{V_l}{N_l^2} + \frac{\lambda}{h_l} = 0 \\ \implies N_l &= \lambda^{-1/2}\sqrt{V_l h_l}. \end{aligned} \tag{2.7}$$

Now plug (2.7) into $\mathbb{V}\text{ar}[\hat{Y}_{\text{ML}}] \leq \epsilon^2/2$ and make λ the subject,

$$\mathbb{V}\text{ar}[\hat{Y}_{\text{ML}}] = \sum_{l=0}^{L} \frac{\sqrt{\lambda}}{\sqrt{V_l h_l}} V_l \leq \frac{\epsilon^2}{2}$$

$$\implies \lambda^{-1/2} \geq 2\epsilon^{-2} \sum_{l=0}^{L} \sqrt{\frac{V_l}{h_l}}. \tag{2.8}$$

To determine the optimal sample size, plug (2.8) in (2.7) and apply the ceiling function to obtain

$$N_l = \left\lceil 2\epsilon^{-2} \sqrt{V_l h_l} \sum_{l=0}^{L} \sqrt{\frac{V_l}{h_l}} \right\rceil. \tag{2.9}$$

As expected, the number of samples per level depends on the variance of that level. As the variance increases, more samples are required. Additionally, as the accuracy increases (i.e., ϵ decreases), the number of samples per level increases.

Although L can be determined by forcing the bias of the approximation to be proportional to $\epsilon/\sqrt{2}$, Giles and Szpruch (2018) show that this L is not optimal. Consequently, the ML Algorithm proposes to use (2.9) in an iterative procedure. In this algorithm, more levels are added as long as the process has not converged. The iterative algorithm is summarised on the next page.

ML Algorithm

1. Start with $L = 0$.

2. Estimate V_L with an initial $N_L = 10^4$ samples using

$$V_L = \mathbb{E}\left[(P_L - P_{L-1})^2\right] - \left(\mathbb{E}[P_L - P_{L-1}]\right)^2$$
$$\approx \frac{1}{N_L}\sum_{i=1}^{N_L}(P_L^i - P_{L-1}^i)^2 - \left(\frac{1}{N_L}\sum_{i=1}^{N_L}(P_L^i - P_{L-1}^i)\right)^2,$$

   where $P_{L-1}^i = 0$ if $L = 0$.

3. Store V_L into variance vector, $\mathcal{V} = [V_0, ..., V_L]$.

4. Store N_L into sample size vector, $\mathcal{N} = [N_0, ..., N_L]$.

5. Define the optimal $\mathcal{N}^* = [N_0^*, N_1^*, ..., N_L^*]$ using (2.9).

6. For all $l \in \{0, ..., L\}$ such that $N_l < N_l^*$

   (a) Generate $N_l - N_l^*$ additional samples.

   (b) Compute the new variance, V_l^*, given by

$$V_l^* = \frac{1}{N_l^*}\sum_{i=1}^{N_l^*}(P_l^i - P_{l-1}^i)^2 - \left(\frac{1}{N_l^*}\sum_{i=1}^{N_l^*}(P_l^i - P_{l-1}^i)\right)^2,$$

   where N_l^* contains the original samples from N_l plus additional samples generated in Step 6a.

   (c) Update sample size vector $N_l = N_l^*$.

   (d) Update the variance vector $V_l = V_l^*$.

7. For all $l \in \{0, ..., L\}$, compute Y_l, given as

$$Y_l = \frac{1}{N_l}\sum_{i=1}^{N_l}\left[P_l^i - P_{l-1}^i\right]$$

   where $P_{L-1}^i = 0$ if $L = 0$.

8. If $L \geq 2$, stop the procedure if

$$\max\left(M^{-1}|Y_{L-1}|, |Y_L|\right) < \frac{1}{\sqrt{2}}(M-1)\epsilon. \tag{2.10}$$

9. If $L < 2$ or not converged, set $L := L + 1$ and go to Step 2.

Chapter 3

European Call Option

In this chapter, the ML algorithm is applied to a simple European call option. This chapter solidifies some of the theory from the previous chapter through an example. The first part of the chapter is to show that the complexity theorem is satisfied. This will be confirmed by the convergence plots. Finally, the prices and execution times are compared.

3.1 Applying the Complexity Theorem

As mentioned in Section 2.3.2, all but condition (iii) of the complexity theorem are satisfied. To prove (iii), two important facts should be recalled. Firstly, a function is Lipschitz continuous if there exists a constant C such that the following inequality holds:

$$|f(x) - f(y)| \leq C|x - y|. \tag{3.1}$$

Secondly, an Euler-Maruyama discretisation has a strong order convergence of 0.5. That is,

$$\begin{aligned} \Big(\mathbb{E}\big[|X - X_l|^p\big]\Big)^{1/p} &= \mathcal{O}\Big(h_l^{1/2}\Big), \\ \mathbb{E}\big[|X - X_l|^2\big] &= \mathcal{O}\Big(h_l\Big), \end{aligned} \tag{3.2}$$

in particular.

The aim is to show that $\mathbb{V}\mathrm{ar}[Y_l]$ is bounded. Recall that Y_l is the level l estimator given by

$$Y_l = \frac{1}{N_l}\sum_{i=1}^{N_l}\Big[P(X_l^{(i)}) - P(X_{l-1}^{(i)})\Big] = \frac{1}{N_l}\sum_{i=1}^{N_l}(P_l^{(i)} - P_{l-1}^{(i)}),$$

where the second equality is just the shorthand notation. If one can show that $V_l := \mathbb{V}\mathrm{ar}[P_l - P_{l-1}] = \mathcal{O}(h_l)$, then $\mathbb{V}\mathrm{ar}[Y_l] = \mathcal{O}(h_l/N_l)$ and the proof is complete

with $\beta = 1$. Proceed as follows,

$$\begin{aligned}
\mathbb{V}\text{ar}[P_l - P] &\leq \mathbb{E}\big[(P_l - P)^2\big] \\
&\leq C^2\mathbb{E}\big[|X_l - X|^2\big] \qquad (3.3)\\
&= \mathcal{O}(h_l), \qquad (3.4)
\end{aligned}$$

where (3.3) is a result of Lipschitz continuity, (3.1), and (3.4) is a result of strong convergence of paths, (3.2). To conclude the proof,

$$\begin{aligned}
V_l &= \mathbb{V}\text{ar}[P_l - P_{l-1}] \\
&= \mathbb{V}\text{ar}\big[(P_l - P) - (P_{l-1} - P)\big] \\
&\leq 2\big(\mathbb{V}\text{ar}[P_l - P] + \mathbb{V}\text{ar}[P_{l-1} - P]\big) \qquad (3.5)\\
&= \mathcal{O}(h_l), \qquad (3.6)
\end{aligned}$$

where (3.5) is due to $\mathbb{C}\text{ov}[P_l - P, P_{l-1} - P] \geq 0$ and (3.6) follows from (3.4).

3.2 Results

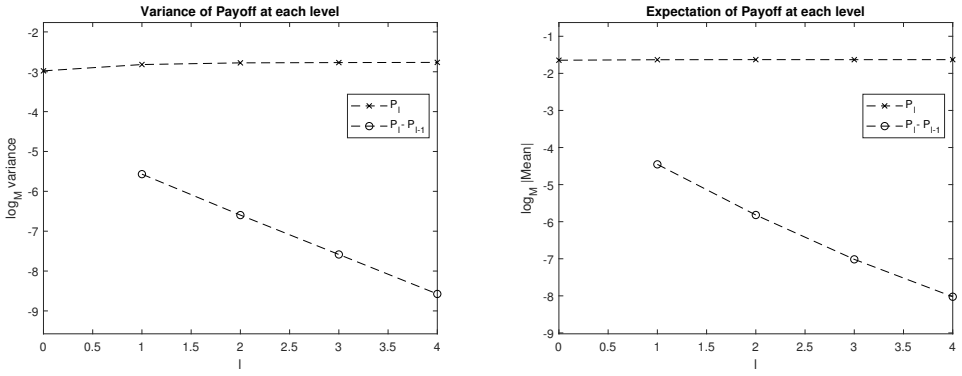
It may not always be easy to prove the complexity theorem mathematically. In those cases, convergence plots are used to determine if the conditions of the complexity theorem are satisfied. Figures 3.1a and 3.1b are a visual representation of conditions (iii) and (i) respectively.

The results in this section are based on the same parameters used by Giles (2008). The values are: $S_0 = 1, K = 1, r = 0.05, T = 1$ and $\sigma = 0.2$. The ML parameters are $M = 4$ and $N = 2 \times 10^6$, where N is the number of samples used to produce the plots.

Figure 3.1a plots $V_l := \mathbb{V}\text{ar}[P_l - P_{l-1}]$ and $\mathbb{V}\text{ar}[P_l]$ for various levels. It is plotted on a logarithmic scale so that the slope can be interpreted in a meaningful way. The gradient of $\log_M(V_l)$ is approximately -1, which suggests that it can be approximated by a straight line given by

$$\begin{aligned}
&\log_M(V_l) \approx -l + c \\
&\implies V_l = M^{-l+c} \propto M^{-l} \propto h_l \\
&\implies V_l = \mathcal{O}(h_l),
\end{aligned}$$

which is the expected result from Section 3.1. This means that $\mathbb{V}\text{ar}[P_l - P_{l-1}]$ decreases as the level increases. Consequently, fewer samples are required to estimate $\mathbb{E}[P_l - P_{l-1}]$ with the same level of accuracy. This is important since it is expensive to produce samples on the finer levels.



(a) Variance on each level. **(b)** Mean Corrections.

Fig. 3.1: Convergence plots for a European call option.

Figure 3.1b shows the plots of $\mathbb{E}[P_l - P_{l-1}]$ for various levels. It is plotted on the logarithmic scale to show a similar convergence relationship regarding $\mathbb{E}[P_l - P_{l-1}]$. That is, $|\mathbb{E}[P_l - P_{l-1}]| = \mathcal{O}(h_l)$. This is not surprising, as it results from the weak convergence of Euler-Maruyama paths. The telescoping identity,

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

provides a meaningful interpretation as well. The level-L price approximation, $\mathbb{E}[P_L]$, can be decomposed into a level-0 price, $\mathbb{E}[P_0]$, plus a sum of corrections in price. Each $\mathbb{E}[P_l - P_{l-1}]$ represents the correction in price from level $l-1$ to l. Figure 3.1b shows that $\mathbb{E}[P_l - P_{l-1}]$ is very small on the last levels which means that the price is close to the level L approximation.

Figure 3.2a shows the optimal number of levels, L, and the number of samples for each level, N_l, for a given level of accuracy, ϵ. As the accuracy increases (i.e., ϵ decreases), so the total number of levels used increases. Moreover, sample sizes (N_l) decrease on higher levels due to lower V_l.

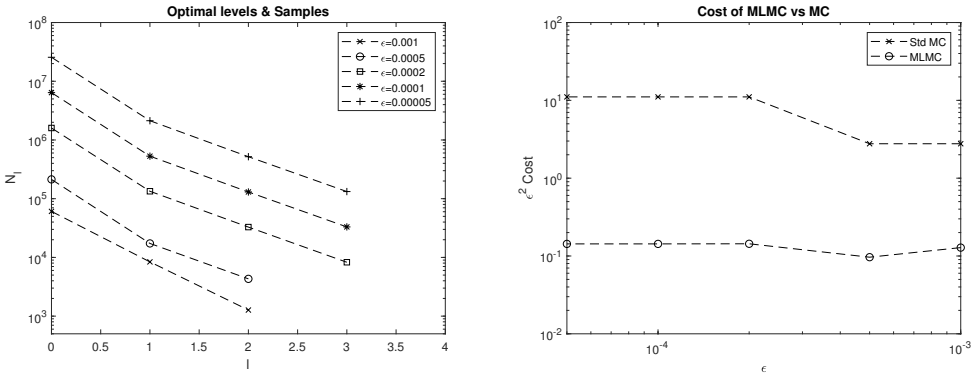
Figure 3.2b shows the cost for both the ML and MC methods. The plot is scaled by ϵ^2 so that differences in the MC and ML costs can be observed easily. In the MC case, the cost is $\mathcal{O}(\epsilon^{-3})$. So the scaled cost is

$$\epsilon^2 \mathcal{O}(\epsilon^{-3}) = \epsilon^{-1}.$$

As ϵ increases, cost decreases. This explains the drop in graph. By the complexity theorem, the cost in the ML case is $\mathcal{O}(\epsilon^{-2}(\log \epsilon)^2)$. This is the cost given by the complexity theorem when $\beta = 1$. So the scaled cost is

$$\epsilon^2 \mathcal{O}\big(\epsilon^{-2}(\log \epsilon)^2\big) = \mathcal{O}\big((\log \epsilon)^2\big).$$

This is the reason why the ML graph is relatively flat.



(a) Samples per level. **(b)** Computational Cost.

Fig. 3.2: Cost and samples plot for a European call option.

To conclude this section, the prices and execution times of ML and MC are compared. In the ML framework, the number of samples on each level is given by (2.9). This ensures the $\mathrm{MSE} \leq \epsilon^2$. For fairness of comparison, MC should also maintain the same level of error. Equation (2.5) shows that the MSE can be decomposed into the variance and bias of the estimator. The bias error is the same for both ML and MC and, as a result, the MC variance needs to be controlled. That is,

$$\frac{\mathbb{V}\mathrm{ar}[P_L]}{N_L} \leq \frac{\epsilon^2}{2}$$
$$\implies N_L \geq 2\epsilon^{-2}\mathbb{V}\mathrm{ar}[P_L].$$

Since N_L needs to be an integer, the ceiling function is applied to obtain

$$N_L = \left\lceil 2\epsilon^{-2}\mathbb{V}\mathrm{ar}[P_L] \right\rceil.$$

Although the terminal value of the stock process can be generated analytically, it was decided to use the Euler-Maruyama method instead. This is done for two reasons. Firstly, it shows that the ML method is robust under this discretisation bias. Secondly, Monte Carlo methods are commonly used for problems which do not have an analytical solution. As a result, it makes sense to compare ML and MC methods as if the analytical solution does not exist.

The code is run 10 times and the average price is presented in Table 3.1. The code was executed in MATLAB R2018a on a 2.7GHz CPU with 8GB RAM. The average was used since these estimators are random variables and, as a result, the estimates would vary. The use of averages reduces the effect of random noise on

the estimates. As expected both methods produce answers close to the analytical solution.

Tab. 3.1: European call price (analytical price: 0.10451).

Epsilon	**0.001**	**0.0005**	**0.0002**	**0.0001**	**0.00005**
ML	0.10438	0.10449	0.10453	0.10447	0.10448
MC	0.10468	0.10433	0.10452	0.10450	0.10447

Table 3.2 presents the ratio of MC and ML time based on an average of 10 runs. A value greater than 1 shows that ML is more efficient than the MC method. Clearly, for all levels of accuracy shown, the ML is more efficient. Moreover, this efficiency is improved for higher levels of accuracy.

Tab. 3.2: Ratio of MC over ML run time.

Epsilon	**0.001**	**0.0005**	**0.0002**	**0.0001**	**0.00005**
Ratio	1.20823	2.59685	11.80358	13.43637	13.64791

Chapter 4

Pricing American Options with a Multilevel Approach

American options, in general, can be tricky to price. The complexity of the option lies in the ability to exercise the option any time before the expiration. In most cases, there are no closed-form analytical solutions to these problems. As a result, numerical techniques are used to estimate the price of the option. There have been many numerical methods developed to price these options. The finite difference (FD) approach is one such method. It produces very accurate results. The disadvantage of FD is that the implementation becomes complicated and time-consuming when the dimension of the problem increases. This is known as the curse of dimensionality. This is where Monte Carlo methods prevail. Although this dissertation only considers one-dimensional problems, ML methods can be applied more widely.

In this chapter, we explore the American pricing problem and summarise two methods for computing the price of an American put option. Thereafter, the multilevel approach is applied to both these methods. The chapter is concluded with a comparison of the price and execution speeds of these methods.

4.1 American Pricing Problem

Traditionally, the American pricing problem has been synonymous with finding an optimal exercise strategy. That is, the price of an American option is given by

$$V_0 = \sup_{\tau \in \mathcal{T}} \mathbb{E}[e^{-r\tau} \tilde{h}_\tau(S_\tau)], \tag{4.1}$$

where $\mathcal{T}$ represents the continuous set of exercise times and $\tilde{h}_\tau(S_\tau)$ the undiscounted payoff of an American option if exercised at time τ.

One way of simplifying the problem is to consider a finite set of exercise times, $\tilde{\mathcal{T}} = \{t_1, t_2, ..., t_M = T\}$, rather than the continuous set of exercise times, $\mathcal{T}$. An additional assumption is that one cannot exercise at inception (i.e., at time $t_0 = 0$).

This price corresponds to a Bermudan option with M exercise opportunities. A Bermudan option is a good approximation for an American option if M is large enough.

Once the problem has been discretised, a dynamic programming formulation is used to compute the price of the Bermudan option. This formulation works backwards in time from maturity to determine the value of the option at each time step. At maturity, the price of the option is simply the exercise value. At any time before maturity, the value of the option is computed as the maximum of the exercise value and the continuation value. In other words,

$$\begin{aligned} V_M(s) &= \tilde{h}_M(s) \\ V_i(s) &= \max\left\{\tilde{h}_i(s), \mathbb{E}\left[e^{-r\Delta t}V_{i+1}(S_{i+1})|S_i = s\right]\right\}, \quad i = 0, ..., M-1, \end{aligned} \tag{4.2}$$

where $\Delta t = T/M$ and V_i represents the time-t_i value of the option. The conditional expectation in (4.2) is the continuation value. It is the time-t_i value of the option if it is held to time t_{i+1}, given the current stock price.

4.2 Least Squares Monte Carlo

Various techniques have been developed to estimate the conditional expectation in (4.2). One of the more famous methods is known as Least Squares Monte Carlo (LSM) which was developed by Carriere (1996), Tsitsiklis and Van Roy (1999) and popularised by Longstaff and Schwartz (2001). In this method, the conditional expectation is approximated by a linear combination of basis functions. That is,

$$\mathbb{E}[e^{-r\Delta t}V_{i+1}(S_{i+1})|S_i = x] = f(\boldsymbol{\beta}_i, x) := \sum_{j=0}^{\mathcal{J}} \beta_j^{(i)}\psi_j(x),$$

where $\mathcal{J}+1$ is the number of basis functions used, $\psi_j(x)$ the weighted j^{th} basis function and $\beta_j^{(i)}$ the coefficient of the j^{th} basis function. The weighting factor, $e^{-x/2}$, is not necessary, however, it produces better numerical results when a matrix scaling problem occurs. The Laguerre polynomials with a weighting of $e^{-x/2}$ were chosen as the basis functions for this chapter. The unweighted polynomials, $\phi_j(x)$, are given by,

$$\begin{aligned} \phi_0(x) &= 1 \\ \phi_1(x) &= 1 - x \\ \phi_k(x) &= \frac{1}{k}(2k - 1 - x)\phi_{k-1}(x) - \frac{k-1}{k}\phi_{k-2}(x), \quad k \geq 2. \end{aligned}$$

The $\hat{\boldsymbol{\beta}}_i$ vector is estimated by minimising the mean square error. The estimated continuation value can be written as $\hat{\boldsymbol{\beta}}_i^{\intercal}\boldsymbol{\psi}(S_i)$ where $\hat{\boldsymbol{\beta}}_i = \left[\beta_0^{(i)}, ..., \beta_{\mathcal{J}}^{(i)}\right]^{\intercal}$ and $\boldsymbol{\psi}(S_i) = [\psi_0(S_i), ..., \psi_{\mathcal{J}}(S_i)]^{\intercal}$. Now, the mean square error can be written as

$$\mathbb{E}\left[\left(\boldsymbol{\psi}(S_i)^{\intercal}\boldsymbol{\beta}_i - \mathbb{E}[e^{-r\Delta t}V_{i+1}(S_{i+1})|S_i]\right)^2\right].$$

Taking the derivative with respect to $\boldsymbol{\beta}_i$ and setting it equal to zero results in,

$$\mathbb{E}\left[\boldsymbol{\psi}(S_i)\left(\boldsymbol{\psi}(S_i)^{\intercal}\boldsymbol{\beta}_i - \mathbb{E}[e^{-r\Delta t}V_{i+1}(S_{i+1})|S_i]\right)\right] = 0. \tag{4.3}$$

Rearranging (4.3) leads to

$$\begin{aligned}\mathbb{E}[\boldsymbol{\psi}(S_i)\boldsymbol{\psi}(S_i)^{\intercal}]\boldsymbol{\beta}_i &= \mathbb{E}\Big[\boldsymbol{\psi}(S_i)\mathbb{E}\big[e^{-r\Delta t}V_{i+1}(S_{i+1})|S_i\big]\Big] \\ &= \mathbb{E}\big[\boldsymbol{\psi}(S_i)e^{-r\Delta t}V_{i+1}(S_{i+1})\big], \end{aligned} \tag{4.4}$$

where (4.4) is obtained by using the Tower property. Finally, making $\boldsymbol{\beta}_i$ the subject results in

$$\boldsymbol{\beta}_i = \mathbb{E}\big[\boldsymbol{\psi}(S_i)\boldsymbol{\psi}(S_i)^{\intercal}\big]^{-1}\mathbb{E}\big[\boldsymbol{\psi}(S_i)e^{-r\Delta t}V_{i+1}(S_{i+1})\big]. \tag{4.5}$$

Writing (4.5) in matrix notation and generalising to the case of N samples used in the regression, leads to

$$\boldsymbol{\beta}_i = (FF^{\intercal})^{-1}FY, \tag{4.6}$$

where

$$F = \begin{bmatrix} \psi_0(S_i^{(1)}) & \psi_0(S_i^{(2)}) & ... & \psi_0(S_i^{(N)}) \\ \psi_1(S_i^{(1)}) & \psi_1(S_i^{(2)}) & ... & \psi_1(S_i^{(N)}) \\ \vdots & \vdots & & \vdots \\ \psi_{\mathcal{J}}(S_i^{(1)}) & \psi_{\mathcal{J}}(S_i^{(2)}) & ... & \psi_{\mathcal{J}}(S_i^{(N)}) \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} V_i\big(S_i^{(1)}\big) \\ \vdots \\ V_i\big(S_i^{(N)}\big) \end{bmatrix}.$$

Now one can estimate (4.2) and consequently the price of the option. For further details regarding implementation, the reader is referred to Longstaff and Schwartz (2001).

LSM defines an exercise strategy since at every time step one can decide whether to exercise or not. That is, exercise when $\tilde{h}_i(s) > \mathbb{E}[e^{-r\Delta t}\tilde{V}_{i+1}(S_{i+1})|S_i = s]$. Define this strategy as $\tilde{\tau}$. Then by (4.1),

$$V_0(S_0) = \sup_{\tau\in\mathcal{T}} \mathbb{E}[e^{-r\tau}\tilde{h}_\tau(S_\tau)] \geq \sup_{\tilde{\tau}\in\tilde{\mathcal{T}}} \mathbb{E}[e^{-r\tilde{\tau}}\tilde{h}_{\tilde{\tau}}(S_{\tilde{\tau}})] = V_0^{\text{LSM}}(S_0).$$

The price is low-biased for two reasons. Firstly, only a finite number of exercise opportunities are considered. Secondly, the conditional expectation in (4.2) is approximated by a linear combination of Laguerre polynomials. Although these polynomials may provide a good estimate for this conditional expectation, it is unlikely to be the exact value of the expectation. Consequently, an imperfect exercise strategy may be obtained, leading to a low-biased price.

4.3 Dual Pricing Method

In the LSM framework, an optimal exercise strategy was proposed. In contrast, the dual approach aims to find an 'optimal martingale'. Haugh and Kogan (2004) and Rogers (2002) worked independently on this approach and provided different estimators. Their work is related to the results of Davis and Karatzas (1994). In this section, we explore the main theorem found in Rogers (2002).

Let h_t be the discounted time-t exercise value. Then the discounted time-t price of an American option is be given by

$$V_t^* = \sup_{t \leq \tau \leq T} \mathbb{E}[h_\tau | \mathcal{F}_t].$$

Assuming a few technical conditions are satisfied, Lamberton (2009) shows that V_t^* is the Snell envelope process of $\{h_t\}_{0 \leq t \leq T}$. Therefore, V_t^* is a supermartingale and its Doob-Meyer decomposition is given by

$$V_t^* = V_0^* + M_t^* + A_t^*, \tag{4.7}$$

where M_t^* represents the Doob martingale and A_t^* represents the previsible decreasing process with $M_0^* = 0$ and $A_0^* = 0$.

Let H_0^1 be the space of martingales, $\{M_t\}_{0 \leq t \leq T}$, such that $\sup_{0 \leq t \leq T} |M_t| \in \mathcal{L}^1$ and $M_0 = 0$ where $\mathcal{L}^1$ is the space of integrable random variables. Then an upper bound for the option price is constructed as

$$\begin{aligned} V_0^* &= \sup_{0 \leq \tau \leq T} \mathbb{E}[h_\tau] \\ &= \sup_{0 \leq \tau \leq T} \mathbb{E}[h_\tau - M_\tau] && \text{(Optional Sampling Theorem)} \\ &\leq \mathbb{E}\left[\sup_{0 \leq t \leq T} (h_t - M_t) \right] && \text{(Jensen's Inequality).} \end{aligned}$$

Taking an infimum over all martingales in H_0^1 results in

$$V_0^* \leq \inf_{M \in H_0^1} \mathbb{E}\left[\sup_{0 \leq t \leq T} (h_t - M_t) \right]. \tag{4.8}$$

We now prove that (4.8) holds with equality. It can be shown that $M_t^* \in H_0^1$. Moreover, M_t^* is the martingale which minimises the expectation in (4.8). First note that

$$V_t^* = \sup_{t \le \tau \le T} \mathbb{E}[h_\tau | \mathcal{F}_t] \ge \mathbb{E}[h_t | \mathcal{F}_t] = h_t. \tag{4.9}$$

Now, write (4.9) in terms of its Doob decomposition. That is,

$$h_t \le V_t^* = V_0^* + M_t^* + A_t^*.$$

To conclude,

$$\begin{aligned} \inf_{M \in H_0^1} \mathbb{E}\left[\sup_{0 \le t \le T} (h_t - M_t) \right] &\le \mathbb{E}\left[\sup_{0 \le t \le T} (h_t - M_t^*) \right] \qquad (4.10) \\ &\le \mathbb{E}\left[\sup_{0 \le t \le T} (V_t^* - M_t^*) \right] \\ &= \mathbb{E}\left[\sup_{0 \le t \le T} (V_0^* + M_t^* + A_t^* - M_t^*) \right] \\ &= \mathbb{E}\left[\sup_{0 \le t \le T} (V_0^* + A_t^*) \right] \\ &= V_0^*, \qquad (4.11) \end{aligned}$$

where (4.10) follows since $M_t^* \in H_0^1$ and (4.11) results, since A_t^* is a decreasing process. Therefore, the price of an American option can be written as

$$V_0^* = \inf_{M \in H_0^1} \mathbb{E}\left[\sup_{0 \le t \le T} (h_t - M_t) \right].$$

Unless the exact Doob martingale is used, the price will be high biased.

Unfortunately, determining the optimal martingale is no easier than finding the optimal exercise time. The Doob martingale can be estimated directly using a recursive formula. Haugh and Kogan (2004) proceed this way, but this method is computationally intensive as sub-simulations are required. Rogers (2002) describes finding this martingale as more of an art than a science. By trying different martingales, a good proxy for the Doob martingale can be found. A simple, yet effective, proxy suggested by Rogers (2002) is the discounted price of the corresponding European put option with the proxy starting at zero. That is,

$$M_t = e^{-rt}\mathrm{BS}(S_t, t) - \mathrm{BS}(S_0, 0), \tag{4.12}$$

where $\mathrm{BS}(S_t, t)$ represents the Black-Scholes put formula with S_t being the stock price at time t. The second term in (4.12) is needed to ensure that the martingale starts at zero.

Since this suggested proxy is not actually the Doob martingale, Rogers (2002) advises a simple optimisation procedure to improve the price of the estimate. On a smaller set of sample paths, find the optimum λ by minimising

$$\mathbb{E}\left[\sup_{0\leq t\leq T}(h_t - \lambda M_t^*)\right]. \tag{4.13}$$

Once λ is estimated, compute V_0 by using a new larger set of sample paths and evaluating (4.13).

4.4 Applying the ML Framework

A key decision required when using the ML approach is how to approximate the payoff of the option at varying levels of accuracy. A simple way to vary the accuracy is to adjust the number of time steps in the sample path. More time steps leads to higher levels of accuracy. Let the level-l approximation be based on a sample path with M^l time steps. In other words, let it have M^l number of exercise opportunities.

Another important decision is how to ensure positive correlation between P_l and P_{l-1} which results in fewer samples being needed to estimate $\mathbb{E}[P_l - P_{l-1}]$ accurately.

In LSM, an M^l-sized vector of Brownian increments for the finer path, P_l, is first created. Then a subset of this vector is formed by grouping Brownian increments in the following way: Sum the first M increments to form the first coarse Brownian increment. Then sum the second M increments to form the second coarse Brownian increment. This process is repeated until a new vector of M^{l-1} Brownian increments is created. This will be used to construct the coarse path, P_{l-1}.

For clarity, let $[\Delta W_1^f, ..., \Delta W_{M^l}^f]$ represent the M^l-sized vector of Brownian increments for the fine path. Then, the Brownian increments for the coarse path can be constructed as

$$\begin{aligned}
\Delta W_1^c &= \Delta W_1^f + \Delta W_2^f + ... + \Delta W_M^f \\
\Delta W_2^c &= \Delta W_{M+1}^f + \Delta W_{M+2}^f + ... + \Delta W_{2M}^f \\
&\vdots \\
\Delta W_{M^{l-1}}^c &= \Delta W_{M^l-M}^f + \Delta W_{M^l-(M-1)}^f + ... + \Delta W_{M^l}^f.
\end{aligned}$$

In the dual approach, the correlation is introduced in the following way. Consider a single sample path that has M^l time steps. Then for each time step compute $Z_t^f := h_t - \lambda M_t$. So we have $[Z_1^f, ..., Z_{M^l}^f]$ for the fine path. In order to compute the

coarse path, every M^{th} point is selected. That is,

$$\begin{aligned} Z_1^c &= Z_M^f \\ Z_2^c &= Z_{2M}^f \\ &\vdots \\ Z_{M^{l-1}}^c &= Z_{M^l}^f. \end{aligned}$$

Now to compute the payoff, the maximum of each vector is taken. That is,

$$\begin{aligned} P_l &= \max\left(Z_1^f, ..., Z_{M^l}^f\right) \\ P_{l-1} &= \max\left(Z_1^c, ..., Z_{M^{l-1}}^c\right) \end{aligned}$$

For both LSM and the dual approach, the ML Algorithm was slightly adapted. Instead of using the ML Algorithm to determine L, it was decided to fix $L = 5$. This means that the ML procedure would always allow for M^L exercise opportunities. If the algorithm was allowed to determine L, then there could be cases where $L < 5$. In those instances, the number of exercise opportunities is less than M^L. Consequently, the price could be low-biased as an option with fewer exercise opportunities is less than or equal to the price of an option with more exercise opportunities.

4.5 Results

Results of applying the ML technique to pricing an American option are presented in this section. The prices and times are an average of 10 executions of the code. The put option parameters are: $\sigma = 0.2, r = 0.05, K = 20$ and $T = 1$ with the results presented for varying initial stock price, S_0. For comparison purposes, the results for crude LSM and dual approaches are presented. Since we are only considering a finite number of exercise opportunities, the finite difference (FD) method is also low-biased. However, if a sufficiently large number of time steps are considered, this price will be close to the unbiased price. A FD method with 10000 time steps and 960 spatial steps is used as a benchmark. The FD method was implemented with a modified SOR algorithm, which had a weighting of $\omega = 1.55$ and tolerance of 1e^{-8}.

The LSM is based on 50000 antithetic samples with 1024 time steps and the first five Laguerre polynomials, weighted by $e^{-x/2}$. The dual method used 10000 antithetic sample paths to determine the optimum λ. A further 50000 independent antithetic samples were used to determine the price. The convergence plots in Figures 4.1 and 4.2 used 10000 paths. The ML mesh parameter was set to $M = 4$. ML results are produced for varying ϵ as in ML Algorithm.

As mentioned in earlier sections, the LSM and dual prices are low- and high-biased, respectively. The pricing results for selected S_0 are presented in Table 4.1. LSM prices form a tight lower bound. These prices, however, become slightly high-biased for OTM values. This is because some matrices in (4.6) are close to singular when performing the matrix inversion, which results in poor estimates of β and the price. The dual prices are always high-biased (except for ITM), but these bounds are not as tight as the LSM bounds.

In general, the ML prices are very close to corresponding crude MC prices. Sometimes, they are slightly better than the corresponding MC estimate. The ML framework maintains the low- and high-biased nature of LSM and dual methods, respectively. In some instances, there is a small improvement in the estimator, and the ML prices form a tighter bound on the true price. Except for the ML LSM ATM case, there seems to be no significant improvement in the accuracy by changing ϵ in the ML Algorithm. This is because the ML Algorithm is heuristic and does not guarantee an improvement in the accuracy.

Each $\mathbb{E}[P_l - P_{l-1}]$ can be thought of as the correction in price from the approximation level $l-1$ to l. Figures 4.1 and 4.2 show that $\mathbb{E}[P_l - P_{l-1}]$ is decreasing and that on the finer levels, these values are rather small. As a result, increasing L should not have a significant effect on the estimated price. In other words, limiting the number of levels to 5 will not result in the ML price being vastly different from the crude price.

Tab. 4.1: American put prices.

	LSM	**ML: LSM**			**FD**	**Dual**	**ML: Dual**		
		0.005	**0.002**	**0.001**			**0.005**	**0.002**	**0.001**
ITM ($S_0 = 15$)	4.9991	4.9999	4.9990	4.9990	5.0000	4.9990	4.9944	4.9930	4.9932
ATM ($S_0 = 20$)	1.2117	1.1983	1.2116	1.2150	1.2177	1.2348	1.2326	1.2329	1.2327
OTM ($S_0 = 25$)	0.1831	0.1814	0.1815	0.1814	0.1808	0.1827	0.1827	0.1823	0.1822

As the ML prices are fairly accurate, it makes sense to compare the execution times, which are presented in Table 4.2. As expected, FD times are the slowest. This is due to the high resolution grid used for the reasons already mentioned. For the ML cases, as ϵ decreases, the execution time increases. This is because $\epsilon \propto 1/N_l$ as seen in (2.9). In all LSM cases, the time decreases if the option is more OTM. This is expected since F in (4.6) only takes sample paths that have a positive exercise value. OTM options will only have a few sample paths that have a positive exercise value. Consequently, fewer paths are considered and the inversion procedure in (4.6) is quicker.

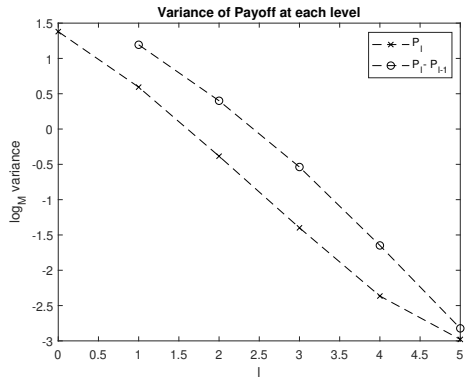
Unfortunately, except for the OTM ($\epsilon = 0.005$) case, the ML LSM times are

worse relative to the crude LSM. This is because $\mathbb{V}\mathrm{ar}[P_l - P_{l-1}]$ starts off relatively high and does not decrease fast enough. For example, in Figure 4.1e, the average gradient is approximately only -0.25. The ML method could be enhanced if the correlation between P_l and P_{l-1} is increased.

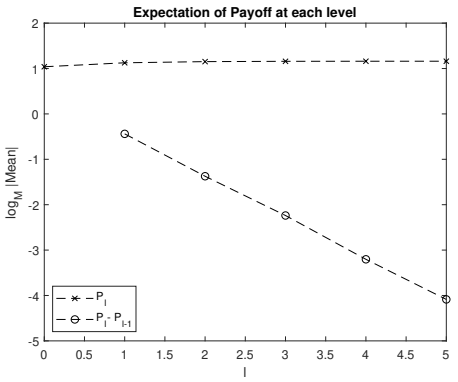
On the other hand, ML dual times are less than the crude times. In contrast to the ML LSM method, $\mathbb{V}\mathrm{ar}[P_l - P_{l-1}]$ starts off low and decreases quickly as l increases. For example, in Figure 4.2a, the variance value starts at -2 and the average gradient is -1.5. This means that even on the first level, not many samples are required.

Tab. 4.2: American put run times (seconds).

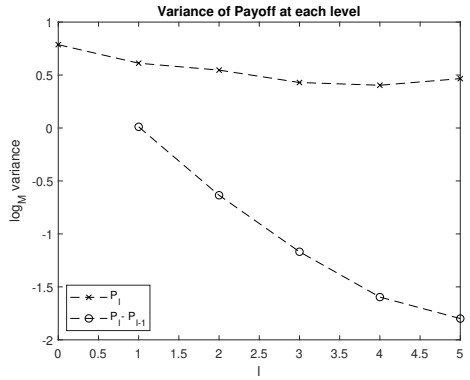
	LSM	**ML: LSM**			**FD**	**Dual**	**ML: Dual**		
		0.005	**0.002**	**0.001**			**0.005**	**0.002**	**0.001**
ITM ($S_0 = 15$)	21.1323	38.4956	234.2211	840.3514	1108.1548	13.7279	6.1410	6.7648	9.2599
ATM ($S_0 = 20$)	11.5520	18.0093	133.1596	547.6129	1081.1995	13.6459	6.2594	7.4644	9.0490
OTM ($S_0 = 25$)	6.0519	4.1325	9.0316	39.9099	1090.0790	15.2328	6.7818	6.7765	7.7134



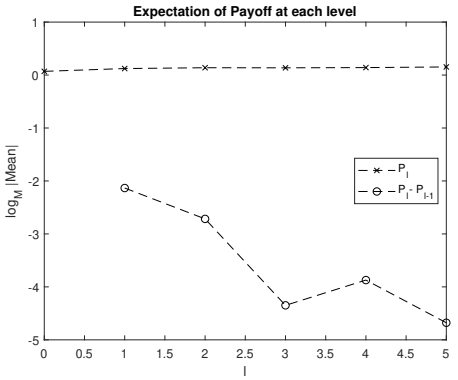
(a) Variance: ITM.



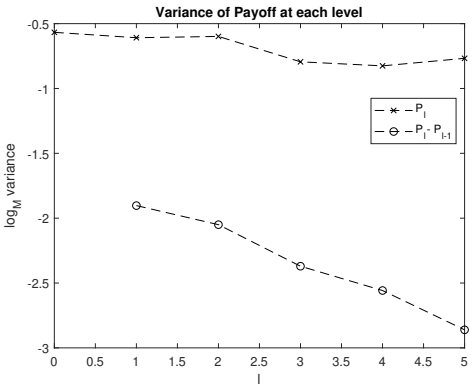
(b) Mean Corrections: ITM.



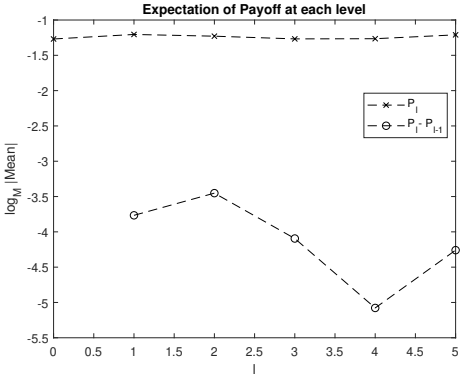
(c) Variance: ATM.



(d) Mean Corrections: ATM.

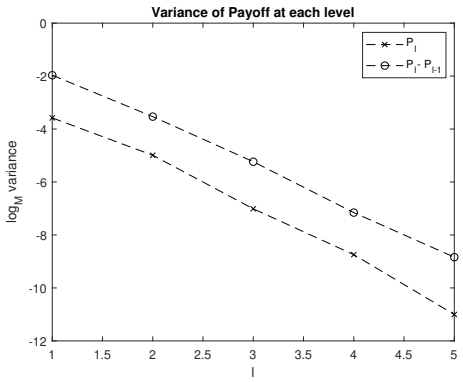


(e) Variance: OTM.

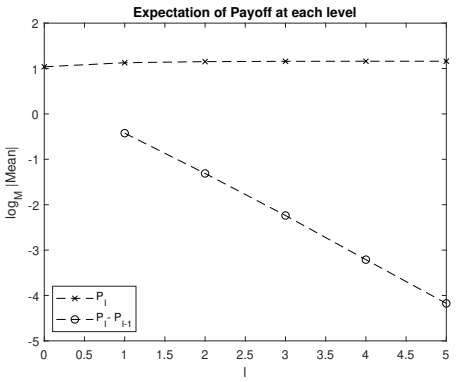


(f) Mean Corrections: OTM.

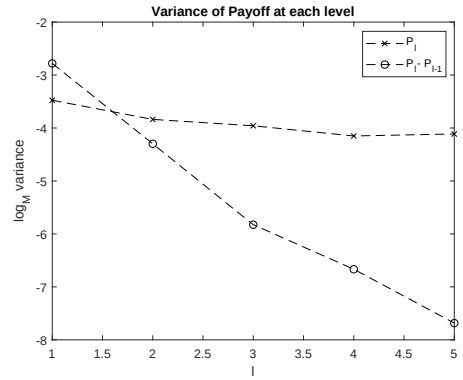
Fig. 4.1: LSM approach: mean and variance of payoff functions.



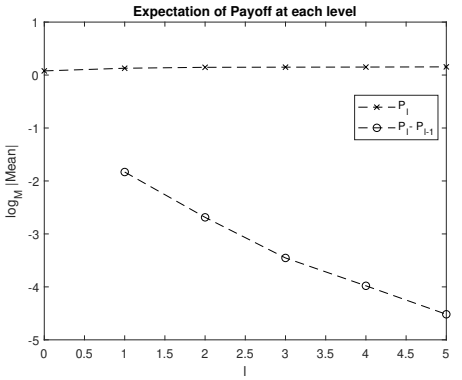
(a) Variance: ITM.



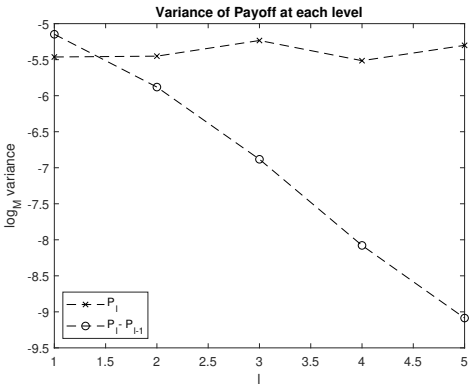
(b) Mean Corrections: ITM.



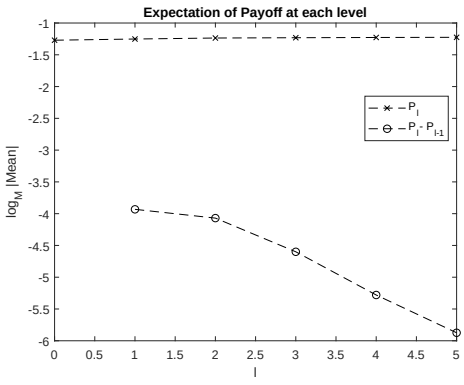
(c) Variance: ATM.



(d) Mean Corrections: ATM.



(e) Variance: OTM.



(f) Mean Corrections: OTM.

Fig. 4.2: Dual approach: mean and variance of payoff functions.

Chapter 5

Conclusions

ML methods rely on approximating a payoff at different levels of accuracy. Although there may be many ways to do this, a common approach used by Giles (2008) is to vary the number of time steps in the sample path. That is, the level-l approximation (P_l) uses a sample path with M^l time steps. The ML estimator comprises the estimate at the coarsest level plus a sum of corrections from the finer levels. The coarsest level estimate (P_0) and the correction terms ($P_l - P_{l-1}$) are estimated by regular MC methods.

A key decision in ML is how to ensure a positive correlation between P_l and P_{l-1}. The greater the correlation, the lower the variance of the correction term. This means that fewer samples are required to compute the estimate. Although there are many ways to introduce this correlation, the most common way is to reuse the Brownian increments from the finer path to construct the coarser path.

ML methods are not guaranteed to be more efficient than MC methods in every situation. If the ML estimator satisfies the conditions in the complexity theorem, then the ML method is more efficient. One of the trickier conditions to prove is that $\mathbb{V}\mathrm{ar}[P_l - P_{l-1}] = \mathcal{O}(h_l^\beta)$. In most cases, proving this is challenging. In these instances, Giles (2008) uses convergence plots to determine the behaviour of $\mathbb{V}\mathrm{ar}[P_l - P_{l-1}]$ and decide whether the condition is satisfied.

The complexity theorem does not provide any guidance on choosing the finest level of approximation, L, or on how many samples to use on each level, N_l. The heuristic ML Algorithm developed by Giles (2008) provides reasonably good estimates for these values.

In the case of the European call option, the complexity theorem is satisfied and convergence plots confirm this. Therefore, the ML estimator is more efficient than the classic MC estimator. For finer levels of accuracy, there is up to a 13 fold improvement in the execution speeds.

Two methods of pricing an American put option were considered. The LSM method estimates an exercise boundary and provides a low-biased estimate for the

price. The dual approach, in contrast, is a high-biased estimate that aims to find an 'optimal martingale' rather than an optimal exercise time. The ML framework was applied to each of these methods.

For both the LSM and dual cases, the ML prices were consistent with the corresponding crude prices. Moreover, the ML framework maintains the low- or high-biased nature of the estimator. In certain circumstances, this bias is smaller and the ML estimate forms a tighter bound on the true price. Adjusting the accuracy level in the ML Algorithm did not provide any significant improvement in the estimate.

The ML framework improved the speed in the dual approach, but not for the LSM approach. This is due to the behaviour of $\mathbb{V}\mathrm{ar}[P_l - P_{l-1}]$ over the levels. In the dual case, this variance starts at a small value and decreases quickly as l increases. In contrast, the LSM variance values started at larger values and decreased quite slowly over the levels. This could be caused by the correlation in the LSM approach not being as strong as the correlation in the dual approach. This demonstrates that the success of the ML method relies heavily on the ability to introduce this strong correlation.

Bibliography

Belomestny, D., Schoenmakers, J. and Dickmann, F. (2013). Multilevel dual approach for pricing American style derivatives, *Finance and Stochastics* **17**(4): 717–742.

Burgos, S. and Giles, M. B. (2012). Computing Greeks using Multilevel path simulation, *in* L. Plaskota and H. Woźniakowski (eds), *Monte Carlo and Quasi-Monte Carlo Methods 2010*, Vol. 23 of *Springer Proceedings in Mathematics Statistics*, Springer, pp. 281–296.

Carriere, J. F. (1996). Valuation of the early-exercise price for options using simulations and non-parametric regression, *Insurance: Mathematics and Economics* **19**(1): 19–30.

Davis, M. H. and Karatzas, I. (1994). A deterministic approach to optimal stopping, *in* F. P. Kelly (ed.), *Probability, Statistics and Optimisation: A Tribute to Peter Whittle*, Vol. 104 of *Wiley Series in Probability and Mathematical statistics*, Wiley, pp. 455–466.

Giles, M. B. (2008). Multilevel Monte Carlo path simulation, *Operations Research* **56**(3): 607–617.

Giles, M. B. and Szpruch, L. (2018). Multilevel Monte Carlo methods for applications in Finance, *in* M. Dempster, J. Kanniainen, J. Keane and E. Vynckier (eds), *High-Performance Computing in Finance*, Chapman and Hall/CRC Financial Mathematics Series, Chapman and Hall/CRC, pp. 197–247.

Giles, M. B. and Waterhouse, B. J. (2009). Multilevel Quasi-Monte Carlo path simulation, *in* H. Albrecher, W. J. Runggaldier and W. Schachermayer (eds), *Advanced Financial Modelling*, Vol. 8 of *Radon Series on Computational and Applied Mathematics*, Walter de Gruyter, pp. 165–181.

Haugh, M. B. and Kogan, L. (2004). Pricing American options: A duality approach, *Operations Research* **52**(2): 258–270.

Heinrich, S. (2001). Multilevel Monte Carlo methods, *in* S. Margenov, J. Wasniewski and P. Y. Yalamov (eds), *International Conference on Large-Scale Scientific Computing*, Vol. 2179 of *Lecture Notes in Computer Science*, Springer, pp. 58–67.

Lamberton, D. (2009). Optimal stopping and American options, *Ljubljana Summer School on Financial Mathematics* p. 134.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: A simple least-squares approach, *The review of financial studies* **14**(1): 113–147.

Rogers, L. C. (2002). Monte Carlo valuation of American options, *Mathematical Finance* **12**(3): 271–286.

Tsitsiklis, J. N. and Van Roy, B. (1999). Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives, *IEEE Transactions on Automatic Control* **44**(10): 1840–1851.

Xia, Y. and Giles, M. B. (2012). Multilevel path simulation for jump-diffusion SDEs, *in* L. Plaskota and H. Woźniakowski (eds), *Monte Carlo and Quasi-Monte Carlo Methods 2010*, Vol. 23 of *Springer Proceedings in Mathematics Statistics*, Springer, pp. 695–708.