# Pricing American Options using Monte Carlo Method


Zhemin Wu
St Catherine's College
University of Oxford

A thesis submitted for the degree of
*Master of Science in Mathematical and Computational Finance*

June 21, 2012

# Acknowledgements

I am grateful to all of those who made this thesis possible. First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Lukas Szpruch, for his guidance and help with my thesis project. I would also like to thank Dr. Samuel Cohen and Dr. Lajos Gergely Gyurko for their valuable discussions about my thesis topics. Appreciation also goes to other departmental lecturers and administrators, and this work would not have been possible without their valuable time and work. The discussions with graduate student, Miss. Shu Zhou, is also highly appreciated.

# Abstract

This thesis reviewed a number of Monte Carlo based methods for pricing American options. The least-squares regression based Longstaff-Schwartz method (LSM) for approximating lower bounds of option values and the Duality approach through martingales for estimating the upper bounds of option values were implemented with simple examples of American put options. The effectiveness of these techniques and the dependencies on various simulation parameters were tested and discussed. A computing saving technique was suggested to reduce the computational complexity by constructing regression basis functions which are orthogonal to each other with respect to the natural distribution of the underlying asset price. The orthogonality was achieved by using Hermite polynomials. The technique was tested for both the LSM approach and the Duality approach. At the last, the Multilevel Mote Carlo (MLMC) technique was employed with pricing American options and the effects on variance reduction were discussed. A smoothing technique using artificial probability weighted payoff functions jointly with Brownian Bridge interpolations was proposed to improve the Multilevel Monte Carlo performances for pricing American options.

# Contents

# Chapter 1

# Introduction

American options are widely studied in mathematical finance. As simple examples, an American call/put option written on an underlying asset with strike $K$ and maturity $T$ gives the option holder the right to buy/sell the underlying asset at price of $K$ at any time up to the maturity $T$. Since the investors can choose an optimal time to exercise the option, the valuations of American type options are relatively more complex than pricing European options, which only can be exercised on maturity and are typically valued using the well-known Black Scholes model[1] under the arguments of no-arbitrage and perfect hedging by trading the underlying assets.

The value of an American option is the value achieved by exercising optimally. Pricing an American option is essentially an equivalence to a problem of solving an optimal stopping problem by defining the optimal exercise rule. The value of an American option is thus calculated by computing the expected discounted payoff under this rule. It is widely acknowledged that an analytical formula does not exist for the value of an American option. Thus as a result, the valuations of American options mostly rely on numerical simulations and the optimization procedure embedded in the problem makes this a challenge. Simple American type derivatives can be solved by the Finite Difference Method[2] (FDM) by stating the problem as a Black-Scholes type PDE with free boundaries associated with the early exercise feature. The penalty method[3] is one of the most popular techniques of pricing American options using FDM. The main drawback of Finite Difference Methods is the restriction to dimensions as the computational cost rapidly increases with the dimensions of the underlying (number of the underlying assets). In the recent years, Monte Carlo based methods[4] have been intensively developed to overcome the current existing problems with FDM based approaches. These Monte Carlo based methods can be roughly divided into two groups. The first group directly employs a recursive scheme for solving the optimal stopping problem by using backward dynamic programming principle. Different techniques are applied to approximate the nested conditional expectations. The least-squares regression method of Longstaff and Schwartz[5][6] is one of the most popular approaches in this group and has attracted the most interests recently. This method uses simple linear regressions through known basis functions of the current state of underlying asset price to approximate

conditional expectations and thus in turn to estimate a lower bound of the value of an American option following a sub-optimal exercise rule. This is what is called the primal approach. The second group comes from the Duality approach developed by Rogers[7] and independently by Haugh and Kogan[8]. The approach was implemented by Andersen and Broadie[9]. The basic idea of this dual approach is to represent the price of an American option through a minimization problem, leading to a high-biased approximation and thus providing upper bounds on prices. The lower bounds provided by the least-squares regression based primal approach and the upper bounds provided by the dual approach contain the true price. Recently, a very interesting extension was presented by Belomestny and Schoenmakers[10]. These authors combined Andersen and Broadie's algorithm[9] with Multilevel Monte Carlo Method (MLMC), developed by Mike Giles[11].

This thesis presents a review of above mentioned Monte Carlo based algorithms for pricing American options and provides implementations with simple examples of American put options. The purpose of the study is to test the effectiveness of these techniques and the dependencies on various simulation parameters. A potentially promising computing saving technique using orthogonal Hermite polynomials is suggested and tested. The Multilevel Mote Carlo method is employed with the algorithms and the effects on variance reduction are discussed. A smoothing technique using artificial probability weighted payoff functions jointly with Brownian Bridge interpolations is proposed to improve the Multilevel Monte Carlo performances for pricing American options. The thesis is organized in the following way.

Chapter 2 reviews the basic theoretical results of the valuation problems of American options. The basic problem formulations based on dynamic programming principle are derived.

Chapter 3 provides an introduction to the least-squares regression based Longstaff-Schwartz method (LSM) for approximating lower bounds of the prices of American options. The algorithms are tested using simple examples of an American put option written on single non-dividend paying stock. The dependencies on the number of Monte Carlo paths and the number of time steps (exercise opportunities) are discussed.

Chapter 4 reviews and implements the Duality approach for estimating the upper bounds of the prices of American options through a martingale technique using approximated value functions and nested Monte Carlo simulations. Various effects of the simulation parameters are discussed.

Chapter 5 suggests a computing saving technique for pricing American options. The technique aims to reduce the computational complexity by constructing regression basis functions which are orthogonal to each other with respect to the natural distribution of the underlying asset price. The orthogonality is achieved by using Hermite polynomials. The technique is tested for the both algorithms of the primal LSM approach and the Duality approach.

Chapter 6 employs the Multilevel Monte Carlo (MLMC) towards the mentioned methods and the effects on variance reduction for both the primal and the dual approaches are tested. A smoothing technique using artificial probability weighted payoff functions and Brownian Bridge interpolations is suggested to improve MLMC performances with the lower bounds estimations through LSM algorithm.

# Chapter 2

# American Options: the Problem Formulations

By definition, an American option is a derivative written on an underlying asset which can be exercised any time up to its maturity. The discretized version of American option, which is also called Bermudan option, is an American style derivative that can be exercised only on pre-specified dates up to maturity. The value of an American option is the maximum value which can be achieved by optimal exercising. This value can be estimated by defining an optimal exercising rule on the option and computing the expected discounted payoff of the option under this rule. This chapter presents the basic formulations of the valuation of American options.

## 2.1 American Options

In contrast to an European option, which can only be exercised at maturity $T$, an American option can be exercised at any time $t \in (0, T]$ although it is usually not exercisable at time 0. For the continuous time case, if we denote the payoff of an American option at any time $t \in (0, T]$ as

$$\tilde{h}(t) = \tilde{h}(S_t, t) \tag{2.1}$$

where $S_t$ is the price process of underlying asset, which is assumed to be a Markovian process, and given a class of admissible stopping times $\mathcal{T}_t$, which takes values in $[t, T]$ ($t \in [0, T]$), the American option pricing problem can be formulated as[4]

$$V_0 = \sup_{\tau \in \mathcal{T}_0} \mathbb{E}\left[e^{-\int_0^\tau r(u)\,\mathrm{d}u} \tilde{h}(S_\tau, \tau)\right] \tag{2.2}$$

where $V_0$ is the value of option at time $t = 0$ , $\{r(t), 0 \le t \le T\}$ is the instantaneous interest rate process and the expectation is taken under the equivalent martingale measure. With this formulation of price for American options, the general no-arbitrage principle holds[12]. We can also absorb the discount factor $e^{-\int_0^t r(u)\,\mathrm{d}u}$ into the payoff function $\tilde{h}(S_t, t)$, and rewrite (2.2) as

$$V_0 = \sup_{\tau \in \mathcal{T}_0} \mathbb{E}\left[h(S_\tau, \tau)\right] \tag{2.3}$$

where $h(S_t, t) = e^{-\int_0^t r(u)\,\mathrm{d}u}\tilde{h}(S_t, t)$ is the discounted payoff. Also, if the price process $S_t$ is Markovian, it is natural to assume the class of admissible stopping times $\mathcal{T}_t = \mathcal{T}(S_t)$ depends only on the current state of the price process, i.e., investors can make an exercise decision at time $t$ based on the price process state $S_t$.

As the simplest example, an American put option written on a single underlying asset $S_t$ with strike $K$ and maturity $T$ gives the option holder the right to sell the underlying asset at price $K$ at any time up to the maturity $T$. Assume the asset price $S_t$ follows a general geometric Brownian motion process given by

$$\mathrm{d}S_t = (r - q)S_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t \tag{2.4}$$

where $r$ and $q$ are interest rate and dividend rate respectively, which are assumed to be constants, $\sigma$ is the volatility and $W_t$ is a standard Brownian motion process. The value of American put option at $t = 0$ is then given by

$$V_0 = \sup_{\tau \in \mathcal{T}_0} \mathbb{E}\left[e^{-r\tau}\left(K - S_\tau\right)^+\right] \tag{2.5}$$

Figure 2.1 shows an example of an American put option written on a non-dividends paying stock with strike $K = 100$, interest rate $r = 5\%$, volatility $\sigma = 0.5$ and maturity $T = 5$ respectively. The value of American option is compared with that of the corresponding European option. The values of American option are calculated using Finite Difference Method (FDM) based penalty approach[3] and the values of European option are calculated using Black-Scholes formula[1].

Strictly speaking, an American option is continuously exercisable, i.e. it can be exercised at any time up to maturity $T$. However, this is not the case for numerical simulations. To numerically price an American option, we need to divide the entire time period $[0, T]$ into a discretized sequence $\{0 = t_0 < t_1 < t_2 < \cdots < t_M = T\}$ and assume the American option can only be exercised at this fixed set of dates, which is in fact the case with special name "Bermudan options". Actually, one may expect the value of the discretized version of American option is simply the approximation of the continuous one when the number of time segments $M \to \infty$. In this thesis, we will simply focus on the discretized case. Once we restrict our concern with time discretization, it's necessary to define a way to approximate the underlying Markovian asset price process. We denote $S_i = S(t = t_i), i \in (0, 1, 2, \cdots, M)$ as the state of asset price process at the $i$th exercise opportunity. The Euler-Maruyama scheme[13] then can be used to approximate the asset price process (2.4), which is given by

$$S_i = S_{i-1} + (r - q)S_{i-1}\Delta t + \sigma S_{i-1}\Delta W_i, \quad i = 1, 2, \cdots M \tag{2.6}$$

where $\Delta t = t_i - t_{i-1}$ is the size of time interval, which is assumed to be constant in this thesis, and $\Delta W_i = W_i - W_{i-1}$ is the independent Brownian increments, which follows a normal distribution $N(0, \sqrt{\Delta t})$. The discretized process $S_i$ given in this way is essentially a Markov chain.

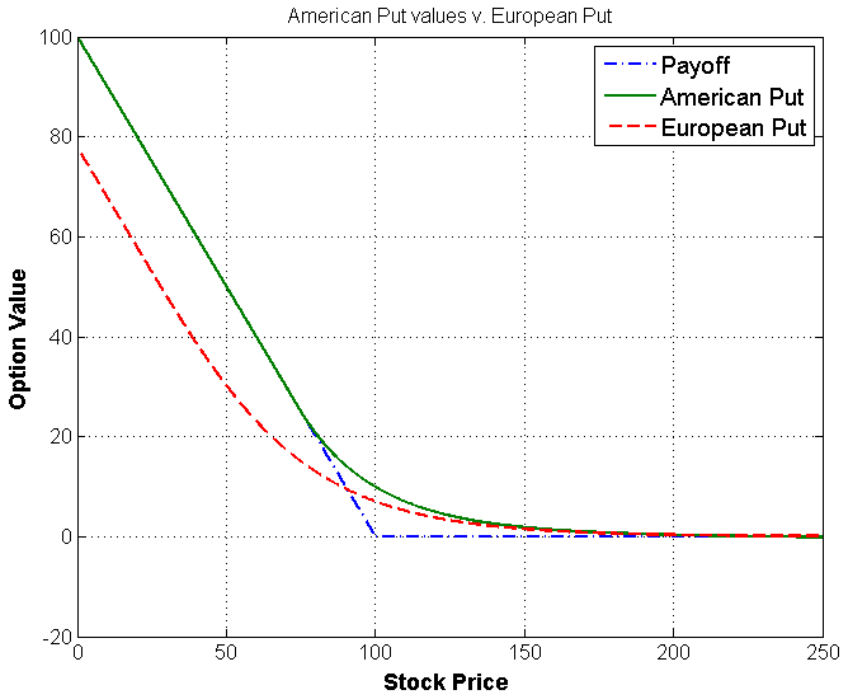


Figure 2.1: The value of American put option compared with the corresponding European option. The value of American option is calculated with Finite Difference Method (FDM). The value of European option is calculated with Black-Scholes formula. Parameters used are $K = 100$, $r = 5\%$, $q = 0\%$, $\sigma = 0.5$, $T = 5$.

## 2.2 Dynamic Programming Principle

Given the discretization setting in Section 2.1, the "dynamic programming principle" introduced by Richard Bellman[14] provides us an effective tool to price American options. Let's assume at time $t = t_i$, the American option under the question has not been excised before. Thus, according to (2.2), assuming the constant interest rate, we can write the value of an American option at $t = t_i$ as

$$\tilde{V}_i(s) = \sup_{\tau \in \mathcal{T}_i} \mathbb{E}\left[e^{-r(\tau - t_i)} \tilde{h}(S_\tau, \tau) \mid S_i = s\right] \tag{2.7}$$

which is essentially equivalent to a newly issued option at $t_i$ with the state of asset price starting from $S_i = s$. Now, if we work recursively from $t = t_M = T$, we are able to define a recursion formula for the price of American options. Firstly, let's consider the terminal time at option's maturity $T$. We know at maturity, the option holder will either choose to exercise the option or let the option expire worthless. Thus, at $t = t_M$, the value of an American option is simply the option's payoff, given by

$$\tilde{V}_M(s) = \tilde{h}(s, T), \quad S_M = s \tag{2.8}$$

where $\tilde{h}(x,t)$ is the payoff function. Now, consider the time $t = t_{M-1}$ with the state of asset price $S_{M-1} = s$, an investor will choose to exercise the option if and only if its current payoff is greater than the discounted expected value to be received if the investor choose not to exercise, thus according to (2.7), the value of option at $t_{M-1}$ is given by

$$\tilde{V}_{M-1}(s) = \max\left\{\tilde{h}(s, t_{M-1}), \mathbb{E}\left[D_{M-1,M}\tilde{V}_M(S_M) \mid S_{M-1} = s\right]\right\} \tag{2.9}$$

where $D_{M-1,M} = e^{-r(t_M - t_{M-1})}$ is the discount factor from $t_{M-1}$ to $t_M$. Thus, by dynamic programming principle, we can derive a recursive formulation for the value of American options as blow

$$\begin{cases} \tilde{V}_M(s) = \tilde{h}(s,T), \quad S_M = s \\ \tilde{V}_{i-1}(s) = \max\left\{\tilde{h}(s, t_{i-1}), \mathbb{E}\left[D_{i-1,i}\tilde{V}_i(S_i) \mid S_{i-1} = s\right]\right\}, i = 1 \cdots M \end{cases} \tag{2.10}$$

This formulation essentially provides the basic principle of optimally exercising an American option, i.e., the exercise desicsion is made at $t = t_{i-1}$ by comparing the immediate exercise payoff $\tilde{h}(s, t_{i-1})$ against the expected present value of continuing. Also, an American option is usually not exercisable at time 0, and this can be done by simply setting $\tilde{h}(s, 0) = 0$.

In (2.10), all the value functions $\tilde{V}_i(s)$ are given in time-$t_i$ currency. However, we are essentially interested in pricing an American option at time 0. So, it's natural to discount all the values to time 0, so that the discount factors $D_{i-1,i}$ can be discarded. Now let

$$h_i(s) = D_{0,i}\tilde{h}(s, t_i), \; V_i(s) = D_{0,i}\tilde{V}_i(s) \quad i = 0, \cdots, M$$

where $D_{0,i} = e^{-r(t_i - t_0)}$ is the discount factor from $t_0$ to $t_i$. Then, according to (2.10), we have

$$\begin{aligned} V_0(s) =& \tilde{V}_0(s), \\ V_M(s) =& h_M(s), \\ V_{i-1}(s) =& D_{0,i-1}\tilde{V}_{i-1}(s) \\ =& D_{0,i-1}\max\left\{\tilde{h}(s, t_{i-1}), \mathbb{E}\left[D_{i-1,i}\tilde{V}_i(S_i) \mid S_{i-1} = s\right]\right\} \\ =& \max\left\{h_{i-1}(s), \mathbb{E}\left[D_{0,i-1}D_{i-1,i}\tilde{V}_i(S_i) \mid S_{i-1} = s\right]\right\} \\ =& \max\left\{h_{i-1}(s), \mathbb{E}\left[V_i(S_i) \mid S_{i-1} = s\right]\right\} \end{aligned}$$

Thus, we can absorb the discount factors into the payoff and value functions, and (2.10) is then simplified as

$$\begin{cases} V_M(s) = h_M(s), \quad S_M = s \\ V_{i-1}(s) = \max\left\{h_{i-1}(s), \mathbb{E}\left[V_i(S_i) \mid S_{i-1} = s\right]\right\}, i = 1 \cdots M \end{cases} \tag{2.11}$$

## 2.3 Optimal Stopping Rule and Continuation Values

The dynamic programming recursions (2.11) give a way to choose optimal exercise strategy for the American option. Assume for any stopping time $\tau \in \{t_1, t_2, \cdots, t_M\}$, a sub-optimal value of American option is defined as

$$V_0^{\tau}(S_0) = \mathbb{E}\left[h_{\tau}(S_{\tau})\right] \tag{2.12}$$

The question here is to choose an optimal stopping time. We know that at any time $t_i$, a discounted value function $V_i(s)$ is assigned to each state of asset price process $S_i = s$ by the dynamic programming recursions (2.11) . Thus, it's natural to choose our optimal stopping time $\tau^*$ as the first time when the payoff exceeds the value function, $\tau^*$ is thus given by[4]

$$\tau^* = \min\left\{\tau_i \in \{t_1, \cdots, t_M\} : h_i(S_i) \geq V_i(S_i)\right\} \tag{2.13}$$

And on the other hand, at any time $t_i$, an investor will exercise the option when the state of asset price $S_i$ gives a payoff larger than the value function, given that the option has not been exercised before. And we call the set of asset price states $\{s : h_i(s) \geq V_i(s)\}$ the "exercise region" at time $t_i$ and the complement to this set as"continuation region". With these definitions, the optimal stopping rule (2.13) can be viewed as the first time the underlying asset price process $S_i$ enters an exercise region.

We can now define the "continuation value" of an American option as the value of holding rather than exercising the option. The "continuation value" is simply defined as[4]

$$\begin{cases} C_M(s) = 0 \\ C_i(s) = \mathbb{E}\left[V_{i+1}(S_{i+1}) \mid S_i = s\right], \quad i = 0, \cdots, M-1 \end{cases} \tag{2.14}$$

where $V_i(x)$ is defined by dynamic programming recursions (2.11). Thus, according to (2.11) we have

$$V_i(s) = \max\left\{h_i(s), C_i(s)\right\}, \ i = 1, \cdots, M \tag{2.15}$$

and the optimal stopping rule in (2.13) can be re-written as

$$\tau^* = \min\left\{\tau_i \in \{t_1, \cdots, t_M\} : h_i(S_i) \geq C_i(S_i)\right\} \tag{2.16}$$

and the value of the option is then determined by this rule as

$$V_0^{\tau^*}(S_0) = \mathbb{E}\left[h_{\tau^*}(S_{\tau^*})\right] \tag{2.17}$$

# Chapter 3

# Longstaff-Schwartz Method: a Lower Bounds Estimator using Least-Squares Regressions

As we have seen in Chapter 2, pricing American options is essentially reduced to a dynamic programming problem with the recursive formulation (2.11). The main difficulty lies in the estimations of the conditional expectations, i.e. the continuation values $C_i(s)$ given by

$$C_i(s) = \mathbb{E}\left[V_{i+1}(S_{i+1}) \mid S_i = s\right], \quad i = 0, \cdots, M-1 \tag{3.1}$$

The most popular approach to estimate the continuation values are based on regression methods, suggested by Longstaff and Schwartz[5]. The continuation value $C_i(s)$ in (3.1), that essentially is a conditional expectation, is approximated by a linear combination of known functions (basis functions) of the current state of asset price $s$. Regression method is then used to estimate the optimal coefficients for the approximation. In this thesis, the least-squares based regression is employed.

## 3.1 Approximating Continuation Values Using Least-Squares Regression

Our aim is to approximate the continuation values $C_i(s)$ by regression, i.e., we want to find an expression of the form

$$C_i(s) = \mathbb{E}\left[V_{i+1}(S_{i+1}) \mid S_i = s\right] = \sum_{j=1}^{J} \beta_{ij}\psi_j(s) = \beta_i^\top \psi(s) \tag{3.2}$$

where $\psi(s) = (\psi_1(s), \psi_2(s), \cdots, \psi_J(s))^\top$ is the vector of some basis functions and $\beta_i = (\beta_{i1}, \beta_{i2}, \cdots, \beta_{iJ})^\top$ is the vector of the regression coefficients, which depends on time $t_i$. If we use the least-squares rule, then the regression coefficients is determined by the following theorem.

**Theorem 3.1.** ***Least-squares based regression coefficients***
*If there exists a relationship between the continuation value $C_i(s)$ and current state of asset price $s$ in the form* (3.2)*, the least-squares regression will give the coefficients $\beta_i$ in the form*

$$\beta_i = \left(\mathbb{E}\left[\psi(S_i)\psi(S_i)^\top\right]\right)^{-1}\mathbb{E}\left[\psi(S_i)V_{i+1}(S_{i+1})\right] \equiv B_\psi^{-1}B_{\psi V} \tag{3.3}$$

*where $B_\psi = \mathbb{E}\left[\psi(S_i)\psi(S_i)^\top\right]$ is a $J\times J$ matrix and $B_{\psi V} = \mathbb{E}\left[\psi(S_i)V_{i+1}(S_{i+1})\right]$ is a vector of length $J$.*

*Proof.* The regression coefficients $\beta_i$ is determined by the least-squares optimization of

$$\mathbb{E}\left[\left(\psi\left(S_i\right)^\top\beta_i - \mathbb{E}\left[V_{i+1}\left(S_{i+1}\right)\mid S_i\right]\right)^2\right] \to \min$$

By taking the derivative with respect to $\beta_i$ and equating to 0, we get

$$\mathbb{E}\left[\psi\left(S_i\right)\left(\psi\left(S_i\right)^\top\beta_i - \mathbb{E}\left[V_{i+1}\left(S_{i+1}\right)\mid S_i\right]\right)\right] = 0$$

By rearranging, we get

$$\begin{aligned}\mathbb{E}\left[\psi\left(S_i\right)\psi\left(S_i\right)^\top\right]\beta_i =& \mathbb{E}\left[\psi\left(S_i\right)\mathbb{E}\left[V_{i+1}\left(S_{i+1}\right)\mid S_i\right]\right]\\ =& \mathbb{E}\left[\psi\left(S_i\right)V_{i+1}\left(S_{i+1}\right)\right]\end{aligned}$$

Solving for $\beta_i$, we finally get

$$\beta_i = \left(\mathbb{E}\left[\psi(S_i)\psi(S_i)^\top\right]\right)^{-1}\mathbb{E}\left[\psi(S_i)V_{i+1}(S_{i+1})\right] \tag{3.4}$$

□

The theoretical value of regression coefficients $\beta_i$ is given by (3.3), which can be estimated in practice by Monte Carlo simulation. Given N independent simulated paths of asset price $\left(S_1^{(n)}, S_2^{(n)}, \cdots, S_M^{(n)}\right), n = 1, \cdots, N$, which can be obtained by Euler-Maruyama scheme (2.6), and assuming that at $t_i$ the value functions $V_{i+1}\left(S_{i+1}^{(n)}\right)$ are known, the least-squares estimation of the regression coefficients $\beta_i$ is then given by

$$\hat{\beta}_i = \hat{B}_{\psi,i}^{-1}\hat{B}_{\psi V,i} \tag{3.5}$$

where

$$\hat{B}_{\psi,i} = \frac{1}{N}\sum_{n=1}^{N}\psi(S_i^{(n)})\psi(S_i^{(n)})^\top \tag{3.6}$$

$$\hat{B}_{\psi V,i} = \frac{1}{N}\sum_{n=1}^{N}\psi(S_i^{(n)})V_{i+1}(S_{i+1}^{(n)}) \tag{3.7}$$

and $S_i^{(n)}$ and $S_{i+1}^{(n)}$ correspond to the same Monte Carlo trajectory. Also, in practice, $V_{i+1}$ need to be replaced by the estimated values $\hat{V}_{i+1}$, given by

$$\hat{V}_{i+1}(S_{i+1}) = \max\left\{h_{i+1}(S_{i+1}), \hat{C}_{i+1}(S_{i+1})\right\} \tag{3.8}$$

Then the continuation value $C_i(S_i)$ can be estimated by

$$\hat{C}_i(S_i) = \hat{\beta}_i^\top \psi(S_i) \tag{3.9}$$

Now we can summarize the procedure for the regression coefficients estimations as in the following algorithm.

**Algorithm 3.2.** ***Backward induction for least-squares regression coefficients estimation***

1. *Simulate $N$ independent trajectories of the asset price process $\{S_1^{(n)}, S_2^{(n)}, \cdots, S_M^{(n)}\}$, $n = 1, \cdots, N$.*
2. *At terminal time $t_M$, set $\hat{V}_M(S_M^{(n)}) = h_M(S_M^{(n)})$.*
3. *Apply backward induction for $i = M-1, \cdots, 1$*
   (a) *estimate the regression coefficients by $\hat{\beta}_i = \hat{B}_{\psi,i}^{-1}\hat{B}_{\psi V,i}$, where $\hat{B}_{\psi,i}$ and $\hat{B}_{\psi V,i}$ are given by* (3.6) *and* (3.7) *respectively.*
   (b) *calculate the continuation values $\hat{C}_i(S_i^{(n)}) = \hat{\beta}_i^\top \psi(S_i^{(n)}), n = 1, \cdots, N$.*
   (c) *set $\hat{V}_i^{(n)} = \max\left\{h_i(S_i^{(n)}), \hat{C}_i(S_i^{(n)})\right\}, n = 1, \cdots, N$.*
4. *Store vectors $\beta_i$ ready to use.*

Once we have the estimations of the regression coefficients $\beta_i$, the value of the option can be estimated with the stopping rule defined in (2.16) using the 2nd set of independent Mote Carlo paths.

## 3.2 Low-biased Estimator Using Optimal Stopping Rule

The estimated regression coefficients $\hat{\beta}_i$ determines the approximations of the continuation values $\hat{C}_i(S_i)$ as in (3.9) at time step $t_i$, given the state of asset price $S_i$, which in turn define an optimal stopping strategy given by

$$\hat{\tau} = \min\left\{\tau_i \in \{t_1, \cdots, t_M\} : h_i(S_i) \geq \hat{C}_i(S_i)\right\} \tag{3.10}$$

Thus, by simulating a 2nd set of Monte Carlo paths and applying this stopping strategy, the value of the American option can be estimated as

$$\hat{V}_0(S_0) = \mathbb{E}\left[h_{\hat{\tau}}(S_{\hat{\tau}})\right] \tag{3.11}$$

As we can see, the approximated stopping rule given in (3.10) is inevitably sub-optimal, i.e., it departs from the true value given by (2.16). Thus, we must have

$$V_0(S_0) = \sup_{\tau \in \mathcal{T}_0} \mathbb{E}\left[h_\tau(S_\tau)\right] \geq \mathbb{E}\left[h_{\hat{\tau}}(S_{\hat{\tau}})\right] = \hat{V}_0(S_0) \tag{3.12}$$

which indicates that the estimator defined in (3.11) is a low-biased estimator which provides a lower bound of the true value. We now summarize the procedure in the following algorithm.

**Algorithm 3.3.** ***Low estimator using optimal stopping rule***

1. *Load regression coefficients* $\beta_i, i = 1, \cdots, M$
2. *Simulate* $N$ *independent paths of the asset price process* $\{S_1^{(n)}, S_2^{(n)}, \cdots, S_M^{(n)}\}$, $n = 1, \cdots, N$. *(This should be a 2nd set of paths independent from the 1st one used to estimate the regression coefficients* $\beta_i$ *gien in Algorithm 3.2)*
3. *Apply forward induction for* $i = 1, \cdots, M-1$, *and for each* $n = 1, \cdots, N$
   (a) *calculate continuation values* $\hat{C}_i(S_i^{(n)}) = \hat{\beta}_i^\top \psi(S_i^{(n)})$
   (b) *calculate the payoff functions* $h_i(S_i^{(n)})$
4. *For terminal time* $i = M$, *set* $C_M(S_M^{(n)}) = 0$ *and calculate* $h_M(S_M^{(n)})$
5. *Set* $\hat{V}_0^{(n)} = h_{i^*}(S_{i^*}^{(n)})$, $i^* = \min\left\{i \in \{1, \cdots, M\} : h_i(S_i^{(n)}) \geq \hat{C}_i(S_i^{(n)})\right\}$
6. *Calculate the estimated value of option as* $\hat{V}_0 = \frac{1}{N}\sum_{n=1}^{N} \hat{V}_0^{(n)}$

## 3.3 Numerical Examples

In this section, the Longstaff-Schwartz method is tested using a simple numerical example of an American put option written on a single non-dividend paying stock with the srike $K = 20$, interest rate $r = 5\%$, volatility $\sigma = 0.4$ and maturity $T = 1$ respectively. For the basis regression functions $\psi_i(x)$, we choose the weighted Laguerre polynomials suggested by Longstaff and Schwartz[5], which are defined as

$$\psi_j(x) = e^{-x/2} L_k(x), \quad k = j-1 \tag{3.13}$$

where $L_k(x)$ is the Laguerre polynomials defined as

$$\begin{aligned} L_k(x) =& \frac{e^x}{k!} \frac{\mathrm{d}^k}{\mathrm{d}x^k} \left(e^{-x} x^k\right) \\ =& \begin{cases} 1, & k = 0 \\ 1 - x, & k = 1 \\ \frac{1}{k}\left((2k-1-x)\, L_{k-1}(x)\right) - (k-1) L_{k-2}(x), & \text{for } k \geq 1 \end{cases} \end{aligned} \tag{3.14}$$

For the numerical testing, we fixed the number of basis functions $J = 5$. We initially set the number of Monte Carlo paths to be $N = 10000$ and fixed the number of time steps $M = 2^5 = 32$. Figure 3.1 shows the numerical results of Longstaff-Schwartz method (LSM) compared with those given by the penalty approach using Finite Difference Method (FDM)[3].

For further investigations, we change the number of independent Monte Carlo paths from 5e3 to 1e5. The results are shown in Table 3.1 and compared with those of the Finite Difference Method.

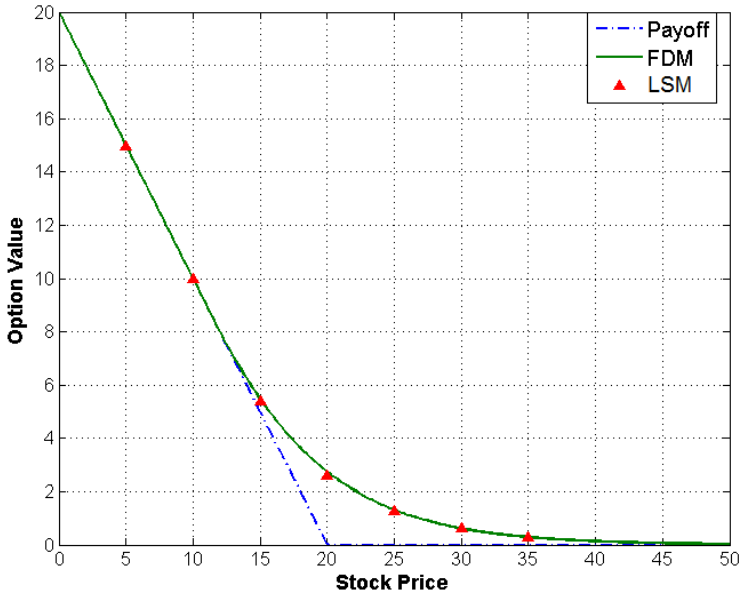


Figure 3.1: Values of American option estimated by Longstaff-Schwartz method compared with the Finite Difference Method (FDM). The parameters used are, $K = 20$, $r = 5\%$, $\sigma = 0.4$ and $T = 1$. Number of paths simulated $N = 10000$, number of time steps $M = 32$.

| | | LSM estimated values | | | Variances of LSM estimator | | |
|---|---|---|---|---|---|---|---|
| $S_0$ | FDM | N=5e3 | N=1e4 | N=1e5 | N=5e3 | N=1e4 | N=1e5 |
| 5 | 15.000 | 14.974 | 14.969 | 14.967 | 2.48E-05 | 1.30E-05 | 1.24E-06 |
| 10 | 10.000 | 9.964 | 9.981 | 9.971 | 9.72E-05 | 4.87E-05 | 5.02E-06 |
| 15 | 5.470 | 5.463 | 5.410 | 5.441 | 1.85E-03 | 9.41E-04 | 9.44E-05 |
| 20 | 2.733 | 2.745 | 2.665 | 2.713 | 2.18E-03 | 1.10E-03 | 1.09E-04 |
| 25 | 1.306 | 1.339 | 1.258 | 1.326 | 9.97E-04 | 5.50E-04 | 5.26E-05 |
| 30 | 0.613 | 0.633 | 0.638 | 0.619 | 5.69E-04 | 2.52E-04 | 2.66E-05 |
| 35 | 0.288 | 0.291 | 0.300 | 0.298 | 2.39E-04 | 1.31E-04 | 1.28E-05 |

Table 3.1: Results of Longstaff-Schwartz based Monte Carlo method using different number of simulated paths $N$, compared with Finite Different Method. The parameters used are, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$ and number of time steps $M = 32$.

As we can see in the table, all the three sets of results give the similar estimated values of the option, which have the similar accuracy compared with those given by FDM. However, fewer simulated paths will give much higher variances. Another noticeable fact is that the variances of the LSM estimators tend to be larger when the option is close to at-the-money. When the initial stock price $S_0 = 20$, the MC estimator gives the largest variances. It's worth mentioning that, some estimated values using Monte Carlo method are lower than those of the Finite Difference Method while others are higher, especially for the deep-out-the-money options. However, we shouldn't consider this as a contradiction with the

argument that the Longstaff-Schwartz method provides a low-biased estimator due to the fact that Monte Carlo method comes with two sources of errors. One is the bias from the method itself and another is from the variance of MC estimators. For deep-out-the-money options, the option values are relatively low, thus the same level variance will give a high biased value even with a low-biased estimator.

Next, to investigate the effects of number of exercise opportunities, we set the number of time steps $M$=8, 16, 32, 64 and 128 respectively and fix the number of Monte Carlo paths to be $N$=10000. The results are shown in Tabel 3.2. We notice that, when the time steps increases, the LSM estimated values are generally getting closer to the results given by FDM, which uses a very fine time stepping (500 steps). This is in agreements with the argument that the value of a discretized version of American option will be getting to an approximation of the value of continuous version American option when the number of time steps $M \to \infty$. The variances of the MC estimator are not noticeably affected for most cases. But for deep-in-the-money and deep-out-of-money options, higher number of time steps gives lower variances.

| | | LSM estimated values | | | | | Variances of LSM estimator | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | FDM | M=8 | M=16 | M=32 | M=64 | M=128 | M=8 | M=16 | M=32 | M=64 | M=128 |
| 5 | 15.000 | 14.876 | 14.937 | 14.969 | 14.983 | 14.992 | 4.96E-05 | 2.51E-05 | 1.30E-05 | 6.61E-06 | 3.18E-06 |
| 10 | 10.000 | 9.860 | 9.943 | 9.981 | 9.990 | 9.990 | 1.98E-04 | 9.87E-05 | 4.87E-05 | 2.50E-05 | 1.28E-05 |
| 15 | 5.470 | 5.418 | 5.441 | 5.410 | 5.421 | 5.449 | 9.40E-04 | 9.20E-04 | 9.41E-04 | 9.41E-04 | 9.99E-04 |
| 20 | 2.733 | 2.756 | 2.742 | 2.665 | 2.739 | 2.728 | 1.10E-03 | 1.04E-03 | 1.10E-03 | 1.08E-03 | 1.07E-03 |
| 25 | 1.306 | 1.400 | 1.348 | 1.258 | 1.227 | 1.289 | 5.73E-04 | 5.02E-04 | 5.50E-04 | 5.53E-04 | 5.90E-04 |
| 30 | 0.613 | 0.672 | 0.666 | 0.638 | 0.641 | 0.619 | 2.77E-04 | 2.93E-04 | 2.52E-04 | 2.87E-04 | 2.74E-04 |
| 35 | 0.288 | 0.331 | 0.311 | 0.300 | 0.302 | 0.281 | 1.51E-04 | 1.35E-04 | 1.31E-04 | 1.18E-04 | 1.19E-04 |

Table 3.2: Results of Longstaff-Schwartz based Monte Carlo method using different number of time steps $M$, compared with Finite Difference Method. The parameters used are $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$ and number of simulated MC parths $N = 10000$.

# Chapter 4

# The Duality through Martingales: an Upper Bounds Estimator

As we have seen in Chapter 3, the Longstaff-Schwartz based Monte Carlo method (LSM) provides a low-biased estimator, giving a lower bound of the true value of the option. We call this the "primal" problem. A new direction in Monte Carlo simulation of American options is the dual approach, developed by Rogers[7] and independently by Haugh and Kogan[8]. One of the most popular implementations of the method was presented by Andersen and Broadie[9]. The basic idea of this dual approach is to represent the price of an American option through a minimization problem. The duality minimizes over a class of supermartingales or martingales and leads to a high-biased approximation, providing upper bounds on prices. The lower bounds provided by LSM and the upper bounds provided by the dual approach contain the true price.

## 4.1 The Duality: a Minimization Problem over Martingales

As we have seen in Chapter 2, the discounted value functions $V_i(S_i)$ satisfies the recursive formulation given by (2.11), which is re-written below

$$\begin{cases} V_M(s) = h_M(s), \quad S_M = s \\ V_{i-1}(s) = \max\left\{h_{i-1}(s), \mathbb{E}\left[V_i(S_i) \mid S_{i-1}=s\right]\right\}, i = 1 \cdots M \end{cases} \tag{4.1}$$

Thus, we have the inequality:

$$V_i(S_i) \geq \mathbb{E}\left[V_i(S_{i+1}) \mid S_i\right], \ i = 0, \cdots, M-1 \tag{4.2}$$

which in fact indicates that the value function $V_i(S_i)$ is a supermartingale. And also, we have

$$V_i(S_i) \geq h_i(S_i), \ i = 0, \cdots, M \tag{4.3}$$

The value function process $V_i(S_i), \ i = 0, \cdots, M$ is in fact the minimal supermartingale dominating the discounted payoff functions $h_i(S_i)$ at all exercise times $t_i$, which is a well-known characterization of American option price [15]. This characterization was extended

by Haugh and Kogan[8], who proposed a minimization problem to formulate the pricing of American options.

In this thesis, we will employ the major results from Haugh and Kogan with specializations to martingales. Let $\mathscr{M} = \{\mathscr{M}_i, i = 0, \cdots, M\}$ be a martingale with $\mathscr{M}_0 = 0$, then the following theorem holds.

**Theorem 4.1.** ***Duality through martingales***
*For any martingales $\mathscr{M} = \{\mathscr{M}_i, i = 0, \cdots, M\}$ satisfying $\mathscr{M}_0 = 0$, the price of American option $V_0(S_0)$ satisfies the inequality*

$$V_0(S_0) \leq \inf_{\mathscr{M}} \mathbb{E}\left[\max_{i=1,\cdots,M} \{h_i(S_i) - \mathscr{M}_i\}\right] \tag{4.4}$$

*The equality holds with the optimal martingale $\mathscr{M}^*$ defined by*

$$\begin{cases} \mathscr{M}_0^* = 0, \\ \mathscr{M}_i^* = \sum_{k=1}^{i} \Delta_k, \ \ for \ \ \ i = 1, \cdots, M \end{cases} \tag{4.5}$$

*where $\Delta_i$ is given by*

$$\Delta_i = V_i(S_i) - \mathbb{E}\left[V_i(S_i) \mid S_{i-1}\right] \tag{4.6}$$

*Proof.* We first prove the inequality (4.4) holds. According to the optional sampling theorem of martingales, for any stopping time $\tau \in \{t_1, t_2, \cdots, t_M\}$ , $\mathbb{E}\left[\mathscr{M}_\tau\right] = \mathscr{M}_0 = 0$, and thus we have

$$\mathbb{E}\left[h_\tau(S_\tau)\right] = \mathbb{E}\left[h_\tau(S_\tau) - \mathscr{M}_\tau\right] \leq \mathbb{E}\left[\max_{i=1,\cdots,M} \{h_i(S_i) - \mathscr{M}_i\}\right]$$

By taking the infimum over the martingales $\mathscr{M}$, we get

$$\mathbb{E}\left[h_\tau(S_\tau)\right] \leq \inf_{\mathscr{M}} \mathbb{E}\left[\max_{i=1,\cdots,M} \{h_i(S_i) - \mathscr{M}_i\}\right] \tag{4.7}$$

The inequality (4.7) holds for any stopping times $\tau$, thus it also holds for the supremum over $\tau$. Thus we have proved the inequality

$$V_0(S_0) = \sup_{\tau} \mathbb{E}\left[h_\tau(S_\tau)\right] \leq \inf_{\mathscr{M}} \mathbb{E}\left[\max_{i=1,\cdots,M} \{h_i(S_i) - \mathscr{M}_i\}\right]$$

Next, we prove the equality in (4.4) holds for $\mathscr{M}^*$ defined by (4.5) and (4.6). We first show that $\mathscr{M}^*$ is actually a martingale. Given the definition of the $\Delta_i$, we have

$$\mathbb{E}\left[\Delta_i \mid S_{i-1}\right] = \mathbb{E}\left[V_i(S_i) - \mathbb{E}\left[V_i(S_i) \mid S_{i-1}\right] \mid S_{i-1}\right] = 0 \tag{4.8}$$

Thus, we have

$$\mathbb{E}\left[\mathscr{M}_i^* \mid S_{i-1}\right] = \mathbb{E}\left[\sum_{k=1}^{i} \Delta_k \mid S_{i-1}\right] = \sum_{k=1}^{i-1} \Delta_k = \mathscr{M}_{i-1}^*$$

which indicates that $\mathscr{M}^*$ satisfies the martingale property. Next, we use backward induction to prove that

$$\begin{aligned} V_i(S_i) =&\mathbb{E}[\max\{h_i(S_i), h_{i+1}(S_{i+1})-\Delta_{i+1}, h_{i+2}(S_{i+2})-\Delta_{i+2}-\Delta_{i+1}, \\ &\cdots h_M(S_M)-\Delta_M-\cdots-\Delta_{i+1}\} \mid S_i], \quad for\ i=1,\cdots,M \end{aligned} \tag{4.9}$$

This holds for the terminal time step $t_M$, since we have $V_M(S_M)=h_M(S_M)=\mathbb{E}[h_M(S_M)|S_M]$. Now, assume (4.9) holds for the time step $t_i$, then according to (4.1) and (4.6), we have

$$\begin{aligned} V_{i-1}(S_{i-1}) =& \max\left\{h_{i-1}(S_{i-1}), \mathbb{E}\left[V_i(S_i) \mid S_{i-1}\right]\right\} \\ =&\mathbb{E}\left[\max\left\{h_{i-1}(S_{i-1}), \mathbb{E}\left[V_i(S_i) \mid S_{i-1}\right]\right\} \mid S_{i-1}\right] \\ =&\mathbb{E}\left[\max\left\{h_{i-1}(S_{i-1}), V_i(S_i)-\Delta_i\right\} \mid S_{i-1}\right] \\ =&\mathbb{E}[\max\{h_{i-1}(S_{i-1}), h_i(S_i)-\Delta_i, h_{i+1}(S_{i+1})-\Delta_{i+1}-\Delta_i, \\ &\cdots h_M(S_M)-\Delta_M-\cdots-\Delta_i\} \mid S_{i-1}] \end{aligned}$$

which indicates (4.9) also holds for $t_{i-1}$. Finally, at $t=t_0$, recall that we have assumed the option is not exercisable immediately and $h_0(S_0)=0$, the option value at $t_0$ is thus given by

$$V_0(S_0) = \mathbb{E}\left[V_1(S_1) \mid S_0\right] = \mathbb{E}\left[V_1(S_1)-\Delta_1 \mid S_0\right]$$

and according to (4.9)

$$\begin{aligned} V_1(S_1) =&\mathbb{E}[\max\{h_1(S_1), h_2(S_2)-\Delta_2, h_3(S_3)-\Delta_3-\Delta_2, \\ &\cdots h_M(S_M)-\Delta_M-\cdots-\Delta_2\} \mid S_1] \end{aligned}$$

We then finally get

$$\begin{aligned} V_0(S_0) =& \quad \mathbb{E}\left[\mathbb{E}\left[\max_{i=1,\cdots,M}\left\{h_i(S_i)-\mathscr{M}_i^*\right\} \mid S_1\right] \mid S_0\right] \\ =& \qquad \mathbb{E}\left[\max_{i=1,\cdots,M}\left\{h_i(S_i)-\mathscr{M}_i^*\right\}\right] \end{aligned} \tag{4.10}$$

This verifies that equality in (4.4) holds for $\mathscr{M}^*$ defined by (4.5) and (4.6), which is indeed an optimal martingale. □

Equation (4.10) provides an estimator for pricing American options. If we can find a martingale $\hat{\mathscr{M}}$ that is close to the optimal martingale $\mathscr{M}^*$, then we can estimate the value of an American option by

$$\hat{V}_0(S_0) = \mathbb{E}\left[\max_{i=1,\cdots,M}\left\{h_i(S_i)-\hat{\mathscr{M}}_i\right\}\right] \tag{4.11}$$

which is indeed our duality estimator. Notice that the martingale $\hat{\mathscr{M}}$ is inevitably sub-optimal and thus (4.11) provides a high-biased estimator, providing an upper bound for the option price. This, conjunct with the lower bound given by LSM approach, gives a range containing the true price of the American option.

## 4.2 The Duality Estimation Using Martingales from Approximated Value Functions

As we have seen in the previous section, we can use (4.11) to estimate the duality by constructing a martingale $\hat{\mathscr{M}}$ that is close to the optimal martingale $\mathscr{M}^*$. This can be done by constructing the martingales based on the approximated value functions $\hat{V}_i(S_i)$. Given the definition of the optimal martingale $\mathscr{M}^*$ in (4.5) and (4.6), it's natural to define the approximation $\hat{\mathscr{M}}$ as

$$\begin{cases} \hat{\mathscr{M}}_0 = 0, \\ \hat{\mathscr{M}}_i = \sum_{k=1}^{i} \hat{\Delta}_k, \ \ for \ \ i = 1, \cdots, M \end{cases} \tag{4.12}$$

where $\hat{\Delta}_i$ is given by

$$\hat{\Delta}_i = \hat{V}_i(S_i) - \mathbb{E}\left[\hat{V}_i(S_i) \mid S_{i-1}\right] \tag{4.13}$$

It easy to verify that $\hat{\mathscr{M}}$ defined by (4.12) and (4.13) satisfies the general martingale property. Recall that $\hat{V}_i(S_i)$ is given by

$$\hat{V}_i(S_i) = \max\left\{h_i(S_i), \hat{C}_i(S_i)\right\} \tag{4.14}$$

and $\hat{C}_i(S_i)$ is the estimation of the continuation value, given by

$$\hat{C}_i(S_i) = \hat{\mathbb{E}}\left[V_{i+1}(S_{i+1})|S_i\right] = \hat{\beta}_i^\top \psi(S_i) \tag{4.15}$$

where $\hat{\beta}_i$ is the vector of LSM regression coefficients and $\psi(x)$ is the vector of our basis functions. One might think of replacing the conditional expectation $\mathbb{E}\left[\hat{V}_i(S_i)|S_{i-1}\right]$ in (4.13) with $\hat{C}_{i-1}(S_{i-1})$. However, this will not give a valid construction of martingales. The reason is that $\hat{C}_{i-1}(S_{i-1})$ is the estimation of the conditional expectation of true value functions, i.e. $\hat{C}_{i-1}(S_{i-1}) = \hat{\mathbb{E}}\left[V_i(S_i)|S_{i-1}\right]$, which is generally not equivalent to an estimation of $\mathbb{E}[\hat{V}_i(S_i)|S_{i-1}]$, which is the conditional expectation of approximated value functions. Thus, to construct a valid martingale, we need to use a nested simulation[4].

Assume we have simulated the main Monte Carlo paths $\{S_i^{(n)} : n = 1, \cdots, N\}$, at each step $S_{i-1}$, we simulate $m$ sub-successors $\{\tilde{S}_i^{(k)} : k = 1, \cdots, m\}$ and estimate the conditional expectation $\mathbb{E}[\hat{V}_i(S_i)|S_{i-1}]$ by

$$\hat{\mathbb{E}}[\hat{V}_i(S_i)|S_{i-1}] = \frac{1}{m}\sum_{k=1}^{m} \hat{V}_i(\tilde{S}_i^{(k)}) \tag{4.16}$$

where $\hat{V}_i(\tilde{S}_i^{(k)})$ is calculated in the same way as given in (4.14). Equation (4.16) gives a conditionally unbiased estimator of $\mathbb{E}[\hat{V}_i(S_i)|S_{i-1}]$ given $S_{i-1}$, thus providing a valid way to construct the martingales as in (4.12) and (4.13), which in turn gives a valid duality estimator as in (4.11). We summarize the procedure in the following algorithm.

**Algorithm 4.2.** ***Duality estimation using martingales based on approximated value functions***

1. *Load regression coefficients* $\beta_i, i = 1, \cdots, M$, *which are given by Algorithm 3.2.*
2. *Simulate* $N$ *independent paths of the asset price process* $\{S_1^{(n)}, S_2^{(n)}, \cdots, S_M^{(n)}\}$, $n = 1, \cdots, N$. *(This should be a 2nd set of paths independent from the 1st one used to estimate the regression coefficients* $\beta_i$ *gien in Algorithm 3.2)*
3. *Set the initial martingale* $\hat{\mathscr{M}}_0 = 0$
4. *Apply forward induction for* $i = 1, \cdots, M-1$, *and for each* $n = 1, \cdots, N$
   (a) *calculate continuation values* $\hat{C}_i(S_i^{(n)}) = \hat{\beta}_i^\top \psi(S_i^{(n)})$
   (b) *calculate the payoff functions* $h_i(S_i^{(n)})$
   (c) *calculate the approximated value functions* $\hat{V}_i(S_i^{(n)}) = \max\left\{h_i(S_i^{(n)}), \hat{C}_i(S_i^{(n)})\right\}$
   (d) *simulate* $m$ *independent sub Monte Carlo successors* $\{\tilde{S}_i^{(1)}, \tilde{S}_i^{(2)}, \cdots, \tilde{S}_i^{(m)}\}$ *from the previous time step* $S_{i-1}^{(n)}$
   (e) *calculate the estimation of martingale differential by* $\hat{\Delta}_i^{(n)} = \hat{V}_i(S_i^{(n)}) - \frac{1}{m}\sum_{k=1}^{m} \hat{V}_i(\tilde{S}_i^{(k)})$ *where* $\hat{V}_i(\tilde{S}_i^{(k)})$ *are calculated similarly using regression.*
   (f) *calculate the martingales* $\hat{\mathscr{M}}_i^{(n)} = \hat{\mathscr{M}}_{i-1}^{(n)} + \hat{\Delta}_i^{(n)}$
5. *For terminal time* $i = M$, *for each* $n = 1, \cdots, N$
   (a) *set* $C_M(S_M^{(n)}) = 0$ *and calculate* $h_M(S_M^{(n)})$
   (b) *set* $\hat{V}_M(S_M^{(n)}) = h_M(S_M^{(n)})$
   (c) *simulate* $m$ *independent sub Monte Carlo successors* $\{\tilde{S}_M^{(1)}, \tilde{S}_M^{(2)}, \cdots, \tilde{S}_M^{(m)}\}$ *from the previous time step* $S_{M-1}^{(n)}$
   (d) *calculate the estimation of martingale differential by* $\hat{\Delta}_M^{(n)} = \hat{V}_M(S_M^{(n)}) - \frac{1}{m}\sum_{k=1}^{m} \hat{V}_M(\tilde{S}_M^{(k)})$ *where* $\hat{V}_M(\tilde{S}_M^{(k)}) = h_M(\tilde{S}_M^{(k)})$.
   (e) *calculate the martingale* $\hat{\mathscr{M}}_M^{(n)} = \hat{\mathscr{M}}_{M-1}^{(n)} + \hat{\Delta}_M^{(n)}$
6. *Set* $\hat{V}_0^{(n)} = \max_{i=1,\cdots,M}\left(h_i(S_i^{(n)}) - \hat{\mathscr{M}}_i^{(n)}\right)$
7. *Calculate the estimated value of duality as* $\hat{V}_0 = \frac{1}{N}\sum_{n=1}^{N} \hat{V}_0^{(n)}$

## 4.3 Numerical Examples

We now test the algorithm 4.2 using the same example provided in Section 3.3. The American put option is written on a single non-dividend paying stock with the srike $K = 20$, interest rate $r = 5\%$, volatility $\sigma = 0.4$ and maturity $T = 1$ respectively. The weighted Laguerre polynomials defined in (3.13) and (3.14) are used as our regression basis functions. For the numerical testing, we fixed the number of basis functions $J=5$. Again, we fixed the number of time steps $M = 32$ and change the number of main Monte Carlo paths from 5e3 to 1e5. The number of simulated sub-successors for the construction of martingales are fixed with $m = 100$. The results are shown in Table 4.1. As we can see, the duality estimations are generally higher than the results either by the LSM or by the Finite Difference Method, which confirms the argument that the duality approach provides a high-biased estimator, thus giving a upper bound of the true price. When we change the number of main Monte Carlo paths, the fewer simulated paths again give higher variances and the highest variances tend to be with the at-the-money options. Another noticeable fact is that the variances of duality estimators are generally much smaller than those of the LSM, which provides the lower bounds.

| | | MC estimated values | | | | | |
|---|---|---|---|---|---|---|---|
| | | N=5e3 | | N=1e4 | | N=1e5 | |
| $S_0$ | FDM | LSM | DUAL | LSM | DUAL | LSM | DUAL |
| 5 | 15.000 | 14.974 | 14.983 | 14.969 | 14.983 | 14.967 | 14.983 |
| 10 | 10.000 | 9.964 | 10.061 | 9.981 | 10.062 | 9.971 | 10.062 |
| 15 | 5.469 | 5.463 | 5.617 | 5.410 | 5.618 | 5.441 | 5.610 |
| 20 | 2.726 | 2.745 | 2.836 | 2.665 | 2.841 | 2.713 | 2.831 |
| 25 | 1.306 | 1.339 | 1.352 | 1.258 | 1.343 | 1.326 | 1.359 |
| 30 | 0.613 | 0.633 | 0.648 | 0.638 | 0.642 | 0.619 | 0.648 |
| 35 | 0.288 | 0.291 | 0.313 | 0.300 | 0.307 | 0.298 | 0.310 |

| | Variances of MC estimators | | | | | |
|---|---|---|---|---|---|---|
| | N=5e3 | | N=1e4 | | N=1e5 | |
| S0 | LSM | DUAL | LSM | DUAL | LSM | DUAL |
| 5 | 2.48E-05 | 7.33E-08 | 1.30E-05 | 3.38E-08 | 1.24E-06 | 3.03E-09 |
| 10 | 9.72E-05 | 5.37E-06 | 4.87E-05 | 2.63E-06 | 5.02E-06 | 2.73E-07 |
| 15 | 1.85E-03 | 2.25E-05 | 9.41E-04 | 1.10E-05 | 9.44E-05 | 1.12E-06 |
| 20 | 2.18E-03 | 3.01E-05 | 1.10E-03 | 1.42E-05 | 1.09E-04 | 1.41E-06 |
| 25 | 9.97E-04 | 1.60E-05 | 5.50E-04 | 8.01E-06 | 5.26E-05 | 8.97E-07 |
| 30 | 5.69E-04 | 1.59E-05 | 2.52E-04 | 3.48E-06 | 2.66E-05 | 3.67E-07 |
| 35 | 2.39E-04 | 1.17E-05 | 1.31E-04 | 3.77E-06 | 1.28E-05 | 1.40E-07 |

Table 4.1: Results of duality approach using different number of simulated paths $N$, compared with Finite Different Method. The parameters used are, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, number of time steps $M = 32$ and number of sub-successors $m = 100$.

To investigate the effects of the number of sub-successors $m$ for the construction of martingales, we fixed the number of main Monte Carlo paths $N = 1e4$ and the number of time steps $M = 32$. We set the number of sub-successors as $m = 50, 100$, and 200 respectively. The results are shown in Table 4.2. As we can see, when the number of sub-successors increases, the estimated duality values decrease, giving smaller upper bounds and thus tighter ranges containing the true price, conjunct with the lower bounds given by LSM. This is reasonable because larger number of sub-successors gives better estimation of the conditional expectation $\mathbb{E}[\hat{V}_i(S_i)|S_{i-1}]$, which in turn gives higher accuracy in the estimations of the martingales $\hat{\mathscr{M}}$. Another interesting fact is that higher number of sub-successors also gives lower overall variances of the estimator, which again is the outcome of the better estimations of the conditional expectations.

| | MC estimated values | | | | |
|---|---|---|---|---|---|
| $S_0$ | FDM | LSM | DUAL/m=50 | DUAL/m=100 | DUAL/m=200 |
| 5 | 15.000 | 14.969 | 14.996 | 14.983 | 14.978 |
| 10 | 10.000 | 9.981 | 10.123 | 10.062 | 10.030 |
| 15 | 5.469 | 5.410 | 5.689 | 5.618 | 5.570 |
| 20 | 2.726 | 2.665 | 2.879 | 2.841 | 2.817 |
| 25 | 1.306 | 1.258 | 1.374 | 1.343 | 1.338 |
| 30 | 0.613 | 0.638 | 0.654 | 0.642 | 0.639 |
| 35 | 0.288 | 0.300 | 0.323 | 0.307 | 0.304 |

| | Variances of MC estimator | | | |
|---|---|---|---|---|
| $S_0$ | LSM | DUAL/m=50 | DUAL/m=100 | DUAL/m=200 |
| 5 | 1.30E-05 | 7.33E-08 | 3.38E-08 | 8.37E-09 |
| 10 | 4.87E-05 | 5.37E-06 | 2.63E-06 | 1.79E-06 |
| 15 | 9.41E-04 | 2.25E-05 | 1.10E-05 | 7.76E-06 |
| 20 | 1.10E-03 | 3.01E-05 | 1.42E-05 | 1.11E-05 |
| 25 | 5.50E-04 | 1.60E-05 | 8.01E-06 | 7.41E-06 |
| 30 | 2.52E-04 | 1.59E-05 | 3.48E-06 | 3.27E-05 |
| 35 | 1.31E-04 | 1.17E-05 | 3.77E-06 | 2.02E-06 |

Table 4.2: Results of duality approach using different number of sub-successors $m$ for the construction of martingales , compared with Finite Different Method. The parameters used are, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, number of main Monte Carlo paths $N = 1e4$, number of time steps $M = 32$.

Finally, we investigated the effects of number of time steps while fixing the number of main Monte Carlo paths $N = 1e4$ and the number of sub-successors $m = 100$. We set $M = 32, 64$ and 128 respectively. The results are shown in Table 4.3. As shown in the table, while the LSM estimates are getting closer to the results given by the Finite Difference Method, the estimates of duality have a trend of increasing, which means the ranges provided by the upper-lower bounds are getting wider. The reason behind this

phenomena is still unknown. Again, the variances of the MC estimators are not noticeably affected by the number of time steps for most cases.

| | MC estimated values | | | | | | |
|---|---|---|---|---|---|---|---|
| | | M=32 | | M=64 | | M=128 | |
| $S_0$ | FDM | LSM | DUAL | LSM | DUAL | LSM | DUAL |
| 5 | 15.000 | 14.969 | 14.983 | 14.983 | 14.997 | 14.992 | 15.006 |
| 10 | 10.000 | 9.981 | 10.062 | 9.990 | 10.083 | 9.990 | 10.095 |
| 15 | 5.470 | 5.410 | 5.618 | 5.421 | 5.653 | 5.449 | 5.671 |
| 20 | 2.733 | 2.665 | 2.841 | 2.739 | 2.882 | 2.728 | 2.900 |
| 25 | 1.306 | 1.258 | 1.343 | 1.227 | 1.387 | 1.289 | 1.401 |
| 30 | 0.613 | 0.638 | 0.642 | 0.641 | 0.663 | 0.619 | 0.683 |
| 35 | 0.288 | 0.300 | 0.307 | 0.302 | 0.317 | 0.281 | 0.339 |

| | Variances of MC estimator | | | | | |
|---|---|---|---|---|---|---|
| | M=32 | | M=64 | | M=128 | |
| $S_0$ | LSM | DUAL | LSM | DUAL | LSM | DUAL |
| 5 | 1.30E-05 | 3.38E-08 | 6.61E-06 | 4.02E-08 | 3.18E-06 | 4.49E-08 |
| 10 | 4.87E-05 | 2.63E-06 | 2.50E-05 | 3.06E-06 | 1.28E-05 | 3.25E-06 |
| 15 | 9.41E-04 | 1.10E-05 | 9.41E-04 | 1.18E-05 | 9.99E-04 | 1.15E-05 |
| 20 | 1.10E-03 | 1.42E-05 | 1.08E-03 | 1.40E-05 | 1.07E-03 | 1.51E-05 |
| 25 | 5.50E-04 | 8.01E-06 | 5.53E-04 | 8.05E-06 | 5.90E-04 | 8.24E-06 |
| 30 | 2.52E-04 | 3.48E-06 | 2.87E-04 | 3.89E-06 | 2.74E-04 | 9.28E-06 |
| 35 | 1.31E-04 | 3.77E-06 | 1.18E-04 | 1.95E-06 | 1.19E-04 | 2.33E-06 |

Table 4.3: Results of duality approach using different time steps $M$, compared with Finite Different Method. The parameters used are, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, number of main Monte Carlo paths $N = 1e4$, number of sub-successors $m = 100$.

# Chapter 5

# Orthogonality of Hermite Polynomials: A Computing Saving Technique

As we have seen in the previous chapters. For pricing American options, both the Longstaff-Schwartz method estimating lower bounds (Algorithm 3.3) and the duality method estimating upper bounds (Algorithm 4.2) rely on the regression based estimations of continuation values $C(S)$, which in turn depend on the quality of estimations of the regression coefficients $\beta$. Thus, for both methods, certain efforts are made towards the computing of regression coefficients, which is fulfilled by a pre-processing procedure (Algorithm 3.2). Recall that the the regression coefficients $\beta$ are estimated by backward induction using

$$\hat{\beta}_i = \hat{B}_{\psi,i}^{-1}\hat{B}_{\psi V,i}, \quad i = M-1, \cdots, 1 \tag{5.1}$$

where

$$\hat{B}_{\psi,i} = \hat{\mathbb{E}}\left[\psi(S_i)\psi(S_i)^\top\right] = \frac{1}{N}\sum_{n=1}^{N}\psi(S_i^{(n)})\psi(S_i^{(n)})^\top \tag{5.2}$$

$$\hat{B}_{\psi V,i} = \hat{\mathbb{E}}\left[\psi(S_i)\hat{V}_{i+1}(S_{i+1})\right] = \frac{1}{N}\sum_{n=1}^{N}\psi(S_i^{(n)})\hat{V}_{i+1}(S_{i+1}^{(n)}) \tag{5.3}$$

and $\hat{V}_{i+1}(S_{i+1}) = \max\left\{h_{i+1}(S_{i+1}), \hat{C}_{i+1}(S_{i+1})\right\}$ are estimated value functions and $\hat{C}_{i+1}(S_{i+1}) = \hat{\beta}_{i+1}^\top\psi(S_{i+1})$ are estimated continuation functions.

As we can see, for good estimations of $\beta$, a lot effort is made towards the estimations of $\hat{B}_{\psi,i}$ given by (5.2), which requires computing the sum of $N$ J×J matrices, where J is the number of basis functions. Usually, to achieve sufficient accuracy in $\beta$, the number of main Monte Carlo paths $N$ is required to be large. This will cause extremely slow computing with respect to Equation (5.2). However, if we can find a way to construct our basis functions $\psi(x)$ such that the matrix $B_{\psi,i} = \mathbb{E}\left[\psi(S_i)\psi(S_i)^\top\right]$ is a J×J identity matrix

$I$ and the estimation $\hat{B}_{\psi,i}$ is close to an identity matrix. Then we can simply discard $\hat{B}_{\psi,i}$ from (5.1) and construct a simplified estimator as

$$\hat{\beta}_i = \hat{B}_{\psi V,i} \tag{5.4}$$

which will give significant computing saving.

Notice that the $qr$ entry of the matrix $B_{\psi,i}$ is given by

$$(B_{\psi,i})_{qr} = \mathbb{E}\left[\left(\psi(S_i)\psi(S_i)^\top\right)_{qr}\right] = \mathbb{E}\left[\psi_q(S_i)\psi_r(S_i)\right], \quad q,r \in \{1,\cdots,J\} \tag{5.5}$$

Thus, if we can construct our basis functions with the orthogonality such that

$$\mathbb{E}\left[\psi_q(S_i)\psi_r(S_i)\right] = \int_{-\infty}^{+\infty} \psi_q(S_i)\psi_r(S_i)p(S_i)\,\mathrm{d}S_i = \delta_{qr} = \begin{cases} 1 & q = r \\ 0 & q \neq r \end{cases} \tag{5.6}$$

where $p(S_i)$ is the unconditional probability density function of asset price state $S_i$, the simplified estimator (5.4) will then be valid. "Hermite polynomials" are promising candidates for the constructions of our basis functions due to the natural orthogonality inherent.

## 5.1 Hermite Polynomials

The probabilists' "Hermite Polynomials" are a classical orthogonal polynomial sequence named after Charles Hermite[16] , defined by

$$H_k(x) = (-1)^k e^{x^2/2}\frac{\mathrm{d}^k}{\mathrm{d}x^k}e^{-x^2/2} = \begin{cases} 1, & k = 0 \\ x, & k = 1 \\ xH_{k-1}(x) - (k-1)H_{k-2}(x), & \text{for } k \geq 1 \end{cases} \tag{5.7}$$

The The most noticeable property of "Hermite Polynomials" is that they are orthogonal with respect to the weight function

$$w(x) = e^{-x^2/2} \tag{5.8}$$

And the following equation holds for all probabilists' "Hermite Polynomials"[17].

$$\int_{-\infty}^{+\infty} H_q(x)H_r(x)e^{-x^2/2}\,\mathrm{d}x = \sqrt{2\pi}q!\delta_{qr} \tag{5.9}$$

By rearranging, we get

$$\int_{-\infty}^{+\infty} \frac{H_q(x)}{\sqrt{q!}}\frac{H_r(x)}{\sqrt{r!}}\frac{e^{-x^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x = \delta_{qr} \tag{5.10}$$

Notice that $\phi(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$ is the probability density of a standard normal distribution and let

$$\widetilde{\psi}_k(x) = \frac{H_k(x)}{\sqrt{k!}} \tag{5.11}$$

we get

$$\mathbb{E}\left[\widetilde{\psi}_q(x)\widetilde{\psi}_r(x)\right] = \int_{-\infty}^{+\infty} \widetilde{\psi}_q(x)\widetilde{\psi}_r(x)\phi(x)\,\mathrm{d}x = \delta_{qr} \tag{5.12}$$

where the expectation is taking with respect to random a variable $x \sim N(0,1)$. Comparing (5.12) wihth (5.6), we find that $\widetilde{\psi}_k(x)$ is a good candidate for our basis functions. The only thing we need to do is to convert our asset price state $S_i$ to a random variable following a standard normal distribution.

## 5.2 Constructing Orthogonal Basis Functions through Hermite Polynomials

As we have seen in the previous section, if we can convert our state of the asset price $S_i$ into a standard normal random variable, we are able to use (5.11) to construct a set of basis functions which are orthogonal to each other. For this purpose, instead of using Euler-Maruyama scheme as given in (2.6), we change our method to simulate the Monte Carlo paths using the exact solutions to SDE (2.4), given by

$$S_i = S_{i-1}\exp\left((r-q-\frac{\sigma^2}{2})\Delta t + \sigma\Delta W_i\right), \quad i = 1,\cdots,M \tag{5.13}$$

Here, we assume the entire time horizon is equally divided. Under this method, the price state $S_i$ has an explicit relation with the initial value $S_0$, given by

$$S_i = S_0\exp\left((r-q-\frac{\sigma^2}{2})i\Delta t + \sigma\sum_{k=1}^{i}\Delta W_k\right) \tag{5.14}$$

Notice that the Brownian increments $\Delta W_k$ in (5.14) are independent to each other, $S_i$ thus has an explicit log-normal distribution and $\log(S_i)$ follows a normal distribution

$$\log(S_i) \sim N\left(\log(S_0)+(r-q-\frac{\sigma^2}{2})i\Delta t, \sigma^2 i\Delta t\right) \tag{5.15}$$

Now, we can easily convert $S_i$ into a standard normal random variable $Y_i$ by

$$Y_i = \frac{\log(S_i)-\log(S_0)-(r-q-\frac{\sigma^2}{2})i\Delta t}{\sigma\sqrt{i\Delta t}} \sim N(0,1) \tag{5.16}$$

Finally, we construct our basis functions through

$$\psi_j(S_i) = \widetilde{\psi}_k(Y_i) = \frac{H_k(Y_i)}{\sqrt{k!}}, \quad k=j-1,\ j=1,\cdots,J \tag{5.17}$$

where $H_k(x)$ is the "Hermite Polynomials" given by (5.7). By doing this, it's easy to verify that

$$\mathbb{E}\left[\psi_q(S_i)\psi_r(S_i)\right] = \mathbb{E}\left[\widetilde{\psi}_{q-1}(Y_i)\widetilde{\psi}_{r-1}(Y_i)\right] = \delta_{qr} \tag{5.18}$$

Thus, we essentially construct a set of basis functions who are orthogonal to each other with respect to the natural distribution of the underlying asset price sate $S_i$. To estimate the regression coefficients with these orthogonal basis functions, the simplified estimator (5.4) can be used, providing large amounts of computing saving.

## 5.3 Numerical Examples

To test the idea, we use the same example of an American put option written on a single non-dividend paying stock with the initial price $S_0 = 20$, the srike $K = 20$, interest rate $r = 5\%$, volatility $\sigma = 0.4$ and maturity $T = 1$ respectively. As mentioned earlier, we do not use regression on the terminal time step $t_M$ since the value functions are simply set as equal to the payoff functions at $t_M$. So, we start from the simplest case with 2 time steps and focus on the middle time step $t_1$. We tested the Algorithm 3.2 using the orthogonal basis functions given in (5.17). For comparison, we tested both the usual estimator given by (5.1) and the simplified estimator given by (5.4). We first computed the estimated matrix $\hat{B}_{\psi,1} = \hat{\mathbb{E}}\left[\psi(S_1)\psi(S_1)^\top\right]$ and set the number of Monte Carlo paths as $N$=1e4, 1e5 and 1e6 respectively. We also fixed the number of basis functions as J=5. The results are shown in Table 5.1 below. As we can see, by using the orthogonal basis functions, the estimated matrix $\hat{B}_{\psi,1} = \hat{\mathbb{E}}\left[\psi(S_1)\psi(S_1)^\top\right]$ is close to an identity matrix $I$, as expected. The quality of this similarity with identity matrix improves with larger number of Monte Carlo paths $N$.

| N=1e4 | | | | |
|---|---|---|---|---|
| 1 | 0.001659 | −0.01207 | 0.014806 | 0.006298 |
| 0.001659 | 0.982932 | 0.027992 | −0.00831 | 0.043112 |
| −0.01207 | 0.027992 | 0.981291 | 0.084783 | −0.02209 |
| 0.014806 | −0.00831 | 0.084783 | 0.962568 | 0.077781 |
| 0.006298 | 0.043112 | −0.02209 | 0.077781 | 0.896297 |
| **N=1e5** | | | | |
| 1 | −0.0008 | −0.00443 | 0.001724 | −0.00217 |
| −0.0008 | 0.993736 | 0.001858 | −0.01202 | 0.019144 |
| −0.00443 | 0.001858 | 0.982148 | 0.02813 | −0.02248 |
| 0.001724 | −0.01202 | 0.02813 | 0.966005 | 0.074931 |
| −0.00217 | 0.019144 | −0.02248 | 0.074931 | 0.936065 |
| **N=1e6** | | | | |
| 1 | 0.000946 | −0.00045 | 0.000104 | 0.000853 |
| 0.000946 | 0.999371 | 0.001517 | 0.000935 | 0.000187 |
| −0.00045 | 0.001517 | 1.00083 | 0.002049 | 0.006765 |
| 0.000104 | 0.000935 | 0.002049 | 1.007879 | 0.008535 |
| 0.000853 | 0.000187 | 0.006765 | 0.008535 | 1.023918 |

Table 5.1: The estimated matrix $\hat{B}_\psi = \hat{\mathbb{E}}\left[\psi(S)\psi(S)^\top\right]$ using orthogonal basis functions. The parameters used are $S_0 = 20$, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, number of time steps $M$=2 and number of basis functions $J$=5.

Table 5.2 gives the estimated regression coefficients $\hat{\beta}_1$ and the corresponding computing time using the simplified estimator (5.4) compared with those given by the standard estimator (5.1). Again we set the number of Monte Carlo paths as $N$=1e4, 1e5 and 1e6

respectively. As shown in the table, the coefficients given by the simplified method are reasonably close to those obtained from the standard approach, and better accuracy was achieved with larger number of Monte Carlo paths. Also, by comparing the computing time used, we found the simplified approach was much faster than the standard approach, as expected.

| | N=1e4 | | N=1e5 | | N=1e6 | |
|---|---|---|---|---|---|---|
| | Standard | Simple | Standard | Simple | Standard | Simple |
| $\hat{\beta}_1$ | $\begin{pmatrix} 2.618 \\ -2.137 \\ 0.678 \\ 0.149 \\ -0.162 \end{pmatrix}$ | $\begin{pmatrix} 2.607 \\ -2.085 \\ 0.590 \\ 0.245 \\ -0.224 \end{pmatrix}$ | $\begin{pmatrix} 2.621 \\ -2.109 \\ 0.662 \\ 0.126 \\ -0.137 \end{pmatrix}$ | $\begin{pmatrix} 2.620 \\ -2.101 \\ 0.641 \\ 0.160 \\ -0.180 \end{pmatrix}$ | $\begin{pmatrix} 2.630 \\ -2.106 \\ 0.647 \\ 0.138 \\ -0.132 \end{pmatrix}$ | $\begin{pmatrix} 2.628 \\ -2.101 \\ 0.643 \\ 0.138 \\ -0.127 \end{pmatrix}$ |
| Time(s) | 0.0376 | 0.0072 | 0.3176 | 0.0571 | 3.386 | 0.6241 |

Table 5.2: The estimated regression coefficients $\hat{\beta}$ and the corresponding computing time based on orthogonal basis functions using simplified estimator $\hat{\beta}_i = \hat{B}_{\psi V,i}$ compared with those given by the standard estimator $\hat{\beta}_i = \hat{B}_{\psi,i}^{-1}\hat{B}_{\psi V,i}$. The parameters used are $S_0 = 20$, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, number of time steps $M$=2 and number of basis functions $J$=5. Platform used: Lenovo T400 laptop, Intel Core 2 Duo Pro 2.8Ghz, 4GB RAM.

Next, we tested the idea together with the LSM approach (Algorithm 3.3) and the duality approach (Algorithm 4.2) to see how well it works with pricing American options. We used two different number of main Monte Carlo paths. Because the accuracy in regression coefficients $\beta$ is essential, we set $N$=1e6 for the pre-processing procedure for $\beta$ estimations (Algorithm 3.2). Then we reset $N$=1e4 for fast processing with LSM and duality simulations. The aproaches using orthogonal basis functions based on Hermite polynomials (both standard and simplified) were tested, compared with the results given by the weighted Laguerre polynomials defined as in (3.13). Other simulations settings used are the number of time steps $M$=32, the number of sub-successors for the duality estimations $m = 100$ and the number of basis functions $J = 5$. The results are presented in Table 5.3. As shown in the table, the results under different approaches are close to each other. While giving similar accuracies, the use of orthogonal basis functions generally gives smaller lower bounds from LSM and larger upper bounds from Duality, indicating wider lower-upper price ranges compared with those given by the basis functions using weighted Laguerre polynomials. Also, as expected, when using orthogonal basis functions, the simplified approach gives very close results compared with those obtained from the standard approach but with much faster speed.

To test the effects of time steps, we set the time steps as $M$=8, 16, 20, 24, 28, 32, 48 and 64 respectively and fixed other parameters as $S_0$=15, number of basis functions $J$=5. The number of Mote Carlo paths is set as $N$=1e4 for both the pre-processing of $\beta$ estimations and the LSM and Duality calculations. The results are shown in Table 5.4.

| | | LSM | | | DUAL | | |
|---|---|---|---|---|---|---|---|
| $S_0$ | FDM | Weighted Laguerre | Orthogonal | Orthogonal Simplified | Weighted Laguerre | Orthogonal | Orthogonal Simplified |
| 10 | 10.000 | 9.965 | 9.962 | 9.962 | 10.047 | 10.033 | 10.037 |
| 15 | 5.470 | 5.372 | 5.341 | 5.317 | 5.632 | 5.687 | 5.700 |
| 20 | 2.733 | 2.671 | 2.630 | 2.682 | 2.860 | 2.933 | 2.944 |
| 25 | 1.306 | 1.274 | 1.267 | 1.271 | 1.395 | 1.496 | 1.504 |
| Mean time (s) | | Weighted Laguerre 111.06 | | Orthogonal 101.14 | | Orthogonal Simplified 20.22 | |

Table 5.3: Estimated lower bounds given by LSM and upper bounds given by Duality approach using the orthogonal basis functions based on Hermite polynomials, compared with the results based on weighed Laguerre polynomials and Finite Difference Method.The parameters used are $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, number of time steps $M$=32, the number of basis functions $J$=5 and the number of sub-successors $m$=100 for duality estimations. The number of main Mote Carlo paths is set as $N$=1e6 for $\beta$ estimations and $N$=1e4 for LSM and Duality estimations. The mean computing time used are listed in the last row.

| | LSM | | | DUAL | | |
|---|---|---|---|---|---|---|
| M | Weighted Laguerre | Orthoganal | Orthogonal Simplified | Weighted Laguerre | Orthoganal | Orthogonal Simplified |
| 8 | 5.405 | 5.387 | 5.376 | 5.539 | 5.562 | 5.558 |
| 16 | 5.451 | 5.436 | 5.326 | 5.587 | 5.622 | 5.872 |
| 20 | 5.439 | 5.433 | 5.266 | 5.601 | 5.643 | 6.640 |
| 24 | 5.417 | 5.432 | 5.271 | 5.612 | 5.666 | 7.041 |
| 28 | 5.436 | 5.432 | 5.224 | 5.623 | 5.681 | 7.122 |
| 32 | 5.404 | 5.403 | 5.175 | 5.635 | 5.704 | 7.208 |
| 48 | 5.388 | 5.373 | 4.913 | 5.653 | 5.724 | 9.975 |
| 64 | 5.457 | 5.422 | 4.804 | 5.665 | 5.735 | 15.000 |
| | | | | | | FDM=5.470 |

Table 5.4: Effects of time steps on the estimated lower bounds given by LSM and upper bounds given by Duality approach using the orthogonal basis functions based on Hermite polynomials, compared with the results based on weighed Laguerre polynomials. The parameters used are $S_0 = 15$, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, the number of basis functions $J$=5 and the number of sub-successors $m$=100 for duality estimations. The number of Mote Carlo paths is set as $N$=1e4 for both the pre-processing of $\beta$ estimations and the LSM and Duality calculations. The Finite Difference Method gives a value of 5.470.

It's interesting to notice that the results given by the simplified approach using the orthogonal basis functions are very sensitive with respect to time steps. As shown in the table, when the number of time steps getting larger, the results obtained from the simplified orthogonal approach start to give unacceptable accuracies, especially in duality values. Recall that the effectiveness of the simplified approach depends on the accuracy of $\beta$ estimations, which in turn depends on the similarity between the matrix $\hat{B}_\psi = \hat{\mathbb{E}}\left[\psi(S)\psi(S)^\top\right]$

and identity matrix $I$. This similarity depends on the number of Monte Carlo paths $N$, as shown in Table 5.1. In this case, we only set the number of Monte Carlo paths $N$=1e4 for the $\beta$ estimations, which will give poor accuracy. On the other hand, the regression coefficients $\beta$ are estimated by backward induction, as given by Algorithm 3.2. Thus, the errors in the estimations of $\beta$s will be accumulated step by step. This is the main reason why the results show larger errors with more time steps. To make the simplified orthogonal approach working with large number of time steps, we have to set a large number of Monte Carlo paths for the pre-processing of $\beta$ estimation. Table 5.5 shows this effect. As we can see, when we increase $N$ to be 1e6, the simplified approach worked again. This seems a main drawback of the simplified technique using the orthogonal basis functions and improvements are needed. But the technique presented here is still potentially beneficial for computational complexity reduction.

| | LSM | | | DUAL | | |
|---|---|---|---|---|---|---|
| N | Weighted Laguerre | Orthoganal | Orthogonal Simplified | Weighted Laguerre | Orthoganal | Orthogonal Simplified |
| 1e4 | 5.457 | 5.422 | 4.804 | 5.665 | 5.735 | 15.000 |
| 1e6 | 5.460 | 5.476 | 5.413 | 5.654 | 5.736 | 5.759 |

| Time used (s) | Weighted Laguerre | Orthoganal | Orthogonal Simplified |
|---|---|---|---|
| N=1e4 | 1.93 | 1.74 | 0.34 |
| N=1e6 | 215.99 | 185.49 | 40.22 |

M = 64 FDM = 5.470

Table 5.5: Increased accuracy with simplified approach when using larger number of Monte Carlo paths. The parameters used are $S_0 = 15$, $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, the number of basis functions $J$=5, the number of time steps $M$=64 and the number of sub-successors $m$=100 for duality estimations.

# Chapter 6

# Multilevel Monte Carlo Approach

Multilevel Monte Carlo (MLMC) is a technique originally suggested by Michael Giles[11] in 2008. The technique can be used to achieve greater accuracy (reduce the variance of the Monte Carlo estimators) for the same computational cost. Since its first introduction, the Multilevel Monte Carlo technique has been rapidly developed for a variety of applications in computational finance. The main idea behind MLMC is to employ multiple sets of simulations with different time-steps (called levels) of the Monte Carlo paths. By constructing an unbiased estimator using the sum of these simulations under different levels, the variance of the estimator is expected to be reduced for a fixed computational cost, given the high correlations between the simulations under different levels. In this chapter, we are going to apply the MLMC technique into our problem of pricing American options and to test its effectiveness with variance reduction.

## 6.1 An Analysis of Standard Monte Carlo

We start with the standard Monte Carlo. Under the standard Monte Carlo framework, the aim is to numerically estimate the expectation $\mathbb{E}[Y]$, where $Y = P(S)$ is a functional of a random variable $S$ that follows a stochastic differential equation given by

$$\mathrm{d}S(t) = \mu(S,t)\,\mathrm{d}t + \sigma(S,t)\,\mathrm{d}t \tag{6.1}$$

In finance, the random variable $S$ is usually the state of the asset price process and the interested functionals are usually payoff functions. In the case of pricing American options, the functional $Y$ becomes the optimal stopped payoff $h_\tau(S_\tau)$ for the lower bounds estimations under Longstaff-Schwartz approach and $\max_{i=1,\cdots,M}\{h_i(S_i) - \mathscr{M}_i^*\}$ for the upper bounds estimation under Duality approach. Without the lost of generality, we will simply denote the target functional as $Y$.

Standard Monte Carlo aims to simulate the underlying random variable $S$ by stepping, i.e., we approximate the continuous stochastic process $S_t$ with a sequence of random variables $S_{\Delta t} = \{S_0, S_1, S_2 \cdots\}$ by dividing the time horizon with step $\Delta t$. This discretization can be achieved by various stepping schemes, such as Euler-Maruyama scheme (2.6). We

denote the target functional under this discretization as $Y_{\Delta t}$ and expect $\mathbb{E}[Y_{\Delta t}] \to E[Y]$ when $\Delta t \to 0$. Then, by sampling $N$ paths of the underlying random variable $S_i$, the standard Monte Carlo estimator of the expected functional $Y$ is given by

$$\hat{Y} = \frac{1}{N}\sum_{n=1}^{N} Y_{\Delta t}^{(n)} = \frac{1}{N}\sum_{n=1}^{N} P(S_{\Delta t}^{(n)}) \tag{6.2}$$

where $Y_{\Delta t}^{(n)}$ refers the observed value with each individual sampled path $S_{\Delta t}^{(n)}$. By standard Monte Carlo results $\hat{Y} \to \mathbb{E}[Y]$, when $\Delta t \to 0$ and $N \to \infty$. But we could not make $\Delta t$ to be 0 in practice. The standard Monte Carlo is usually performed with certain $\Delta t > 0$ and finite $N$, which will produce a mean square error defined by

$$\text{MSE} \equiv \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[Y]\right)^2\right] \tag{6.3}$$

The major concern when performing Monte Carlo simulations is to make the root mean square error small, i.e. to estimate $Y$ with accuracy $\epsilon$ such that $\sqrt{\text{MSE}} \leq \epsilon$ as efficiently as possible. That is to minimize the computational complexity required to achieve the required mean square error. We can rewrite the definition of mean square error as

$$\begin{aligned}
\text{MSE} =& \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[Y]\right)^2\right] \\
=& \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[\hat{Y}] + \mathbb{E}[\hat{Y}] - \mathbb{E}[Y]\right)^2\right] \\
=& \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[\hat{Y}]\right)^2\right] + \left(\mathbb{E}[\hat{Y}] - \mathbb{E}[Y]\right)^2
\end{aligned} \tag{6.4}$$

where the first term is the variance of the Monte Carlo estimator, given by

$$\text{Var}[\hat{Y}] = \frac{1}{N^2}\text{Var}\left[\sum_{n=1}^{N} P(S_{\Delta t}^{(n)})\right] = \frac{1}{N}\text{Var}[P(S_{\Delta t})] \tag{6.5}$$

and the second term is the square of the bias due to discretization approximation. The discretization like Euler-Maruyama scheme (2.6) gives a bias $|\mathbb{E}[\hat{Y}] - \mathbb{E}[Y]| = O(\Delta t)$, and thus the mean square error given by the standard Monte Carlo will be

$$\text{MSE} = \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[Y]\right)^2\right] = O(\frac{1}{N}) + O(\Delta t^2) \tag{6.6}$$

What we need is to make the root mean square error proportional to $\epsilon$, that is $\text{MSE} = O(\epsilon^2)$, which gives

$$\begin{array}{c} 1/N = O(\epsilon^2) \\ \Delta t^2 = O(\epsilon^2) \end{array} \Rightarrow \begin{array}{c} N = O(\epsilon^{-2}) \\ \Delta t = O(\epsilon) \end{array} \tag{6.7}$$

The number of time steps for the discretization is $M = O(\Delta t^{-1})$ and thus the computational cost of standard Monte Carlo is given by

$$C = NM = O(\epsilon^{-3}) \tag{6.8}$$

On the other hand, given a fixed computational cost $C = NM$, the mean square error is approximately given by

$$\text{MSE} \approx \frac{c_1}{N} + \frac{c_2}{M^2} = \frac{c_1}{N} + \frac{c_2 N^2}{C^2} \tag{6.9}$$

where $c_1$ and $c_2$ are positive constants. To minimize MSE, we take the derivative of (6.9) with respect to $N$ and equate the resulted equation to 0 and finally get

$$N = \left(\frac{c_1 C^2}{2c_2}\right)^{1/3}, \; M = \left(\frac{2c_2 C}{c_1}\right)^{1/3} \tag{6.10}$$

and thus

$$\frac{c_1}{N} = \left(\frac{2c_1^2 c_2}{C^2}\right)^{1/3}, \; \frac{c_2}{M^2} = \left(\frac{c_1^2 c_2}{4C^2}\right)^{1/3} \tag{6.11}$$

indicating that the mean square error comes from the variance of the Monte Carlo estimator is twice as the square of the bias due to discretization.

## 6.2 Multilevel Monte Carlo

Multilevel Monte Carlo (MLMC) uses different time resolutions, i.e. number of time steps at different levels, which might have the form $M_l = 2^l, \; l = 0, 1, \cdots, L$ and the corresponding time interval is $\Delta t_l = 2^{-l}T, \; l = 0, 1, \cdots, L$. Denote $P$ as our target functional, which might be a payoff function. For American options, the target functional $P$ will be the optimal stopped payoff $h_\tau(S_\tau)$ for the lower bounds estimations under Longstaff-Schwartz approach and $\max\limits_{i=1,\cdots,M} \{h_i(S_i) - \mathscr{M}_i^*\}$ for the upper bounds estimation under Duality approach respectively.

Let $P_l$ be the approximation of our target functional on level $l$, then the expected value $\mathbb{E}[P_L]$ on the finest level can be constructed by

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}] \tag{6.12}$$

which provides a basic estimator under the MLMC framework.

The idea behind MLMC is to estimate each of the expectations on the right hand side of (6.12) independently in a way which minimizes the overall variance for a given fixed computational cost or minimizes the computational cost at a given variance requirement. Now, denote $\hat{Y}_0$ as the estimator of $\mathbb{E}[P_0]$ with $N_0$ sample paths, and $\hat{Y}_l, \; l = 1, 2, \cdots L$ be the estimator of $\mathbb{E}[P_l - P_{l-1}]$ using $N_l$ samples paths. The simplest way to construct these estimators is a mean of $N_l$ independent paths, given by

$$\hat{Y}_l = \frac{1}{N_l} \sum_{n=1}^{N_l} (P_l^{(n)} - P_{l-1}^{(n)}) \tag{6.13}$$

What's important here is that $P_l^{(n)} - P_{l-1}^{(n)}$ should come from two different discrete approximations but the same underlying Monte Carlo path[18]. By doing this, the difference is

small on finer levels and so is its variance. Hence very few samples will be required on finer levels to achieve acceptable accuracies in the expected value.

By these notations, we can finally write the MLMC estimator as

$$\hat{Y} = \sum_{l=0}^{L} \hat{Y}_l \tag{6.14}$$

It's easy to verify that

$$\mathbb{E}[\hat{Y}_l] = \frac{1}{N_l} \sum_{n=1}^{N_l} \mathbb{E}[P_l^{(n)} - P_{l-1}^{(n)}] = \mathbb{E}[P_l - P_{l-1}] \tag{6.15}$$

and

$$\mathbb{E}[\hat{Y}] = \sum_{l=0}^{L} \mathbb{E}[\hat{Y}_l] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}] = \mathbb{E}[P_L] \tag{6.16}$$

This indicates that MLMC provides an unbiased estimator with respect to the finest level approximation and thus the final accuracy depends on the accuracy of the finest level $L$. The variance of the MLMC estimator is given by

$$\mathrm{Var}\left[\hat{Y}\right] = \sum_{l=0}^{L} \mathrm{Var}\left[\hat{Y}_l\right] = \sum_{l=0}^{L} \frac{1}{N_l} \mathbb{V}_l \tag{6.17}$$

where $\mathbb{V}_l = \mathrm{Var}[P_l - P_{l-1}]$ is the variance at each level.

Michael Giles proved the following complexity theorem[11] with respect to the MLMC approach, which we give here without proof.

**Theorem 6.1.** ***Complexity theorem of MLMC***[11]
*Let $P$ denote a functional of the solution of SDE* (6.1) *for a given Brownian path $W(t)$, and let $P_l$ denote the corresponding level $l$ numerical approximation with the time discretization using interval $\Delta t_l = \frac{T}{M_l} = \frac{T}{2^l}$ steps. If there exit independent estimators $\hat{Y}_l$ based on $N_l$ Monte Carlo sample paths, and positive constants $\alpha \geq \frac{1}{2}$, $\beta$, $c_1$, $c_2$, $c_3$ such that*

*(i)* $\mathbb{E}[P_l - P] \leq c_1 \Delta t_l^{\alpha}$

*(ii)* $\mathbb{E}[\hat{Y}_l] = \begin{cases} \mathbb{E}[P_0], & l = 0 \\ \mathbb{E}[P_l - P_{l-1}], & l > 0 \end{cases}$

*(iii)* $\mathbb{E}[\hat{Y}_l] \leq c_2 N_l^{-1} \Delta t_l^{\beta}$

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

*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*

$$\hat{Y} = \sum_{l=1}^{L} \hat{Y}_l$$

*has an MSE with bound*

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

*with a computational complexity C with bound*

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

As we can see from the above theorem, the complexity of MLMC given a root mean square error $\epsilon$ is reduced to $O(\epsilon^{-2})$ compared with that of the standard Monte Carlo, which is $O(\epsilon^{-3})$. What we are more interested here is that, given a fixed computational complexity, how to minimize the variance of the combined estimator $\hat{Y}$. Recall that the mean square error is given by

$$\begin{aligned} \text{MSE} =& \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[P]\right)^2\right] \\ =& \mathbb{E}\left[\left(\hat{Y} - \mathbb{E}[\hat{Y}]\right)^2\right] + \left(\mathbb{E}[\hat{Y}] - \mathbb{E}[P]\right)^2 \end{aligned} \tag{6.18}$$

where the first term is the variance of the estimator given by

$$\text{Var}\left[\hat{Y}\right] = \sum_{l=0}^{L} \text{Var}\left[\hat{Y}_l\right] = \sum_{l=0}^{L} \frac{1}{N_l}\mathbb{V}_l = \sum_{l=0}^{L} \frac{1}{N_l}\text{Var}[P_l - P_{l-1}]$$

and the second term is the bias term. Giles[11] proved that the upper bounds of the variance and bias errors are both $\frac{1}{2}\epsilon^2$, given the conditions in Theorem 6.1 hold.

For given computational cost of $C = \sum_{l=0}^{L} N_l \Delta t_l^{-1} T$, to minimize the variance of $\hat{Y}$, we can set up a Lagrangian function to find minimum

$$L = \sum_{l=0}^{L} \frac{1}{N_l}\mathbb{V}_l + \lambda\left(\sum_{l=0}^{L} N_l \Delta t_l^{-1} T - C\right)$$

Taking the first order derivative with respect to $N_l$ and equating to 0, we get

$$N_l = \lambda^{-\frac{1}{2}}\sqrt{\frac{\mathbb{V}_l \Delta t_l}{T}}$$

therefore

$$\text{Var}\left[\hat{Y}\right] = \sum_{l=0}^{L} \frac{1}{N_l}\mathbb{V}_l = \sum_{l=0}^{L} \frac{\sqrt{\lambda T}}{\sqrt{\mathbb{V}_l \Delta t_l}}\mathbb{V}_l$$

What we want is $\text{Var}[\hat{Y}] \leq \epsilon^2/2$ and thus we can get

$$\lambda^{-\frac{1}{2}} \geq 2\epsilon^{-2}\sum_{l=0}^{L}\sqrt{\mathbb{V}_l T/\Delta t_l}$$

and the optimal number of sample paths for each level is

$$N_l = 2\epsilon^{-2}\sqrt{\mathbb{V}_l \Delta t_l}\sum_{l=0}^{L}\sqrt{\mathbb{V}_l/\Delta t_l} \tag{6.19}$$

We notice that the optimal number of level paths $N_l$ is proportional to $\sqrt{\mathbb{V}_l \Delta t_l}$. In practice, we fix the overall computational cost as $C$ and let $N_l = c_1\sqrt{\mathbb{V}_l \Delta t_l}$, thus we will have

$$C = \sum_{l=0}^{L} N_l \Delta t_l^{-1} T = c_1 \sum_{l=0}^{L}\sqrt{\frac{\mathbb{V}_l}{\Delta t_l}}T$$

Solving for $c_1$ we get

$$c_1 = \frac{C}{\sum_{l=0}^{L}\sqrt{\frac{\mathbb{V}_l}{\Delta t_l}}T}$$

Then finally, the optimal number of sample paths for each level with a fixed computational cost $C$ is

$$N_l = \frac{C\sqrt{\mathbb{V}_l \Delta t_l}}{\sum_{l=0}^{L}\sqrt{\frac{\mathbb{V}_l}{\Delta t_l}}T}$$

If the time interval for each level is given by $\Delta t_l = T/2^l$, then we finally get

$$N_l = \frac{C\sqrt{\mathbb{V}_l 2^{-l}}}{\sum_{l=0}^{L}\sqrt{\mathbb{V}_l 2^l}} \tag{6.20}$$

In practice, the variance at each level $\mathbb{V}_l$ can be estimated by a pilot run of the MLMC estimator using a fixed number of sample paths $\tilde{N}_l$ at each level. Then, the optimal number $N_l$ can be calculated using (6.20), which in turn can be used to implement the actual MLMC algorithm. Furthermore, notice that

$$\mathbb{V}_l = \mathrm{Var}[P_l - P_{l-1}] = \mathrm{Var}[P_l] + \mathrm{Var}[P_{l-1}] - \mathrm{Cov}[P_l, P_{l-1}] \tag{6.21}$$

thus the MLMC method will give better variance reduction if the approximations of our target functional $P$ are highly correlated between levels.

## 6.3 Numerical Examples

As we did in the previous chapters, we are going to apply the Multilevel Monte Carlo technique into our problem of pricing American options. We continue to use the same simple example of an American put option written on a single non-dividend paying stock with the srike $K = 20$, interest rate $r = 5\%$, volatility $\sigma = 0.4$ and maturity $T = 1$ respectively. Recall that the pricing is a 2-stage process. Firstly, we need to use the regression method to estimate the regression coefficients $\beta$ for the purpose of estimating the continuation values, which is done by the pre-processing by backward induction (Algorithm 3.2). After that, we can either estimate a lower bound of the American option price using the Longstaff-Schwartz (LSM) optimal stopping algorithm (Algorithm 3.3) or estimate an upper bound

of the price using the Duality approach through martingales (Algorithm 4.2). Usually, the first stage of $\beta$ estimations is a prerequisite for good estimations of either lower bounds or upper bounds. Thus, we are not going to test the MLMC on this stage and will use the standard Monte Carlo with the finest level settings to estimate regression coefficients $\beta$ which are available for all other levels to use. Our testing with MLMC is on the second stage, i.e. the estimations of lower or upper bounds and our target functionals at each level are given by

$$P_l = \begin{cases} h_{\tau_l^*}(S_{\tau_l^*}), \tau_l^* = \min\{\tau_i \in \{t_1, \cdots, t_{M_l}\} : h_i(S_i) \geq C_i(S_i)\}, & \text{for LSM} \\ \max\limits_{i=1,\cdots,M_l} \{h_i(S_i) - \mathscr{M}_i^*\}, & \text{for Duality} \end{cases} \tag{6.22}$$

where $M_l = 2^l,\ l \in \{1, \cdots, L\}$ is the number of time steps at each level. We fixed the overall computational as $C = NM_L$, where $N$ = 1e5 is fixed and we also fix the finest level as $L$=7. We use (6.20) to calculate the optimal number of sample paths at each level through a pilot run with $\tilde{N}_l$=1e4. Here, we start our level from $l = 1$ since $l = 0$ corresponds to an European option, which is not applicable in our problem. Our MLMC estimator thus becomes

$$\mathbb{E}[P_L] = \mathbb{E}[P_1] + \sum_{l=2}^{L} \mathbb{E}[P_l - P_{l-1}] \tag{6.23}$$

For faster processing, instead of using the weighted Laguerre polynomials, we use simple Laguerre polynomials as our basis functions. The initial stock prices were set as $S_0$=12.5, 20 and 25 respectively, representing the in-the-money, at-the-money and out-of-the-money options. The results are listed in Table 6.1.

As we can see from the results, the MLMC approach gives very close price estimations for both the lower bounds using LSM method and the upper bounds using the Duality method, which confirms that the MLMC estimator is unbiased. By comparing the variances, we notice that the variances are noticeably reduced with MLMC for the LSM estimations with the cases of at-the-money option ($S_0$=20) and out-of-the-money option ($S_0$=25). For these cases, the variances of MLMC are about only 30% of those with standard MC. This is indeed what we expected. By looking at the variances $\mathbb{V}_l$ at each level for these two cases, we notice that the variances are dramatically decreasing when we move forward to higher levels. Recall that the variances at each level depend on the correlations between the functionals at adjacent coarse and fine levels $\rho(P_l, P_{l-1})$ (see Equation (6.21)). Higher correlations between $P_l$ and $P_{l-1}$ will cause lower variances at each level, which in turn cause the optimal number of sample paths at each level $N_l$ to be smaller (recall that $N_l$ is proportional to $\sqrt{\mathbb{V}_l \Delta t_L}$). Notice that these correlations are all very close to 1 for the two cases and increase with levels, giving good overall variance reduction under MLMC. As for the in-the-money case ($S_0 = 12.5$), the situation changed. Although the correlations $\rho(P_l, P_{l-1})$ still increase with levels, they become much weaker, especially for lower levels. This in turn causes much higher level variances $\mathbb{V}_l$ compared with the other two cases. And thus, the variance under MLMC is even larger than its standard counterpart, about 170%.

**CASE1: $S_0 = 12.5$ FDM = 7.528**

| LSM | | | | DUAL | | | |
|---|---|---|---|---|---|---|---|
| level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ |
| 1 | 858970 | 17.876 | N/A | 1 | 345730 | 0.171 | N/A |
| 2 | 259758 | 3.365 | 0.902 | 2 | 268972 | 0.208 | 0.252 |
| 3 | 170270 | 2.870 | 0.902 | 3 | 153913 | 0.137 | 0.225 |
| 4 | 99154 | 1.974 | 0.922 | 4 | 93679 | 0.102 | 0.170 |
| 5 | 57649 | 1.270 | 0.943 | 5 | 62551 | 0.087 | 0.119 |
| 6 | 37267 | 1.039 | 0.949 | 6 | 41468 | 0.080 | 0.089 |
| 7 | 22379 | 0.709 | 0.962 | 7 | 28491 | 0.075 | 0.085 |
| $\text{LSM}_{\text{MLMC}}$ | 7.448 | $\mathbb{V}_{\text{MLMC}}$ | 1.52E-04 | $\text{DUAL}_{\text{MLMC}}$ | 7.737 | $\mathbb{V}_{\text{MLMC}}$ | 9.17E-06 |
| $\text{LSM}_{\text{std}}$ | 7.456 | $\mathbb{V}_{\text{std}}$ | 8.81E-05 | $\text{DUAL}_{\text{std}}$ | 7.741 | $\mathbb{V}_{\text{std}}$ | 4.06E-07 |

**CASE2: $S_0 = 20$ FDM = 2.733**

| LSM | | | | DUAL | | | |
|---|---|---|---|---|---|---|---|
| level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ |
| 1 | 1724635 | 14.287 | N/A | 1 | 370052 | 0.291 | N/A |
| 2 | 234803 | 0.535 | 0.982 | 2 | 259579 | 0.411 | 0.101 |
| 3 | 135814 | 0.347 | 0.986 | 3 | 144198 | 0.236 | 0.260 |
| 4 | 91932 | 0.325 | 0.986 | 4 | 89784 | 0.189 | 0.426 |
| 5 | 51647 | 0.204 | 0.991 | 5 | 61105 | 0.170 | 0.469 |
| 6 | 28273 | 0.139 | 0.994 | 6 | 41872 | 0.161 | 0.464 |
| 7 | 18687 | 0.111 | 0.995 | 7 | 29658 | 0.160 | 0.435 |
| $\text{LSM}_{\text{MLMC}}$ | 2.651 | $\mathbb{V}_{\text{MLMC}}$ | 3.15E-05 | $\text{DUAL}_{\text{MLMC}}$ | 3.031 | $\mathbb{V}_{\text{MLMC}}$ | 1.81E-05 |
| $\text{LSM}_{\text{std}}$ | 2.639 | $\mathbb{V}_{\text{std}}$ | 1.09E-04 | $\text{DUAL}_{\text{std}}$ | 3.031 | $\mathbb{V}_{\text{std}}$ | 1.39E-06 |

**CASE3: $S_0 = 25$ FDM = 1.306**

| LSM | | | | DUAL | | | |
|---|---|---|---|---|---|---|---|
| level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ |
| 1 | 1749086 | 9.212 | N/A | 1 | 311562 | 0.167 | N/A |
| 2 | 291308 | 0.532 | 0.973 | 2 | 267823 | 0.253 | 0.207 |
| 3 | 157964 | 0.279 | 0.982 | 3 | 159286 | 0.164 | 0.510 |
| 4 | 86714 | 0.176 | 0.987 | 4 | 96102 | 0.118 | 0.674 |
| 5 | 46791 | 0.107 | 0.992 | 5 | 61579 | 0.099 | 0.713 |
| 6 | 30232 | 0.078 | 0.994 | 6 | 41574 | 0.091 | 0.712 |
| 7 | 16042 | 0.050 | 0.996 | 7 | 28612 | 0.088 | 0.699 |
| $\text{LSM}_{\text{MLMC}}$ | 1.281 | $\mathbb{V}_{\text{MLMC}}$ | 1.88E-05 | $\text{DUAL}_{\text{MLMC}}$ | 1.695 | $\mathbb{V}_{\text{MLMC}}$ | 1.06E-05 |
| $\text{LSM}_{\text{std}}$ | 1.280 | $\mathbb{V}_{\text{std}}$ | 5.86E-05 | $\text{DUAL}_{\text{std}}$ | 1.697 | $\mathbb{V}_{\text{std}}$ | 1.41E-06 |

Table 6.1: The MLMC pricing of American put options. The parameters used in simulations are $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$, the number of basis functions $J$=5 and the number of sub-successors $m$=100 for duality estimations. The number of finest levels is set as $L$=7 and total compuational cost is fixed at $C$=$NM_L$=1e4×$2^7$. In the table, $N_l$, $\mathbb{V}_l$ and $\rho(P_l, P_{l-1})$ represent the number of sample paths at each level, the variance at each level and the correlation coefficients between the functionals at adjacent coarse and fine levels. $\text{LSM}_{\text{MLMC}}$ and $\text{LSM}_{\text{std}}$ represent the lower bound estimations using LSM method under MLMC and standard Mote Carlo respectively. $\text{DUAL}_{\text{MLMC}}$ and $\text{DUAL}_{\text{std}}$ are corresponding upper bound estimations using Duality approach. $\mathbb{V}_{\text{MLMC}}$ and $\mathbb{V}_{\text{std}}$ are corresponding variances.

Recall that higher levels, i.e. finer time steps correspond to more exercise opportunities for our American options under the question. The simulation results given here indicate that the lower bounds of the price of an American option (i.e. the optimal stopping values given by the LSM approach) are generally highly correlated with respect to the number of exercises opportunities and this correlation becomes weaker when we move from out-of-money options to in-the-money options.

What surprised us is that the MLMC method is not working well with the Duality estimations. Although it still gives an unbiased estimation and overally gives smaller variances with respect to those given by LSM, its variances are much higher if compared with those given by standard Monte Carlo, about 7.5 times for out-the-money case, 13 times for the at-the-money case and 22.5 times for the in-the-money case respectively. The direct reason behind is that there are much weaker correlations exist between $P_l$ and $P_{l-1}$ for the Duality estimations. The maximum correlation is only about 0.7 and the minimum is extremely small with a value of 0.085, which in turn causes relatively high level variances $\mathbb{V}_l$. On the other hand, the correlations $\rho(P_l, P_{l-1})$ do not generally increase with levels and they even decrease with levels for the in-the-money case, which is in contrary with our expectation. The results shown here indicates that the duality values are relatively weakly correlated with different number of exercise opportunities. The actual reason behind this is still unclear.

## 6.4 Improved MLMC for Lower Bounds Estimation using Probability Weighted Payoff Functions and Brownian Bridge Interpolations

As we have seen in the previous section, the MLMC method worked relatively poor with the lower bounds estimations under LSM approach for the case of in-the-money option. The main reason is that the correlations between the functionals at adjacent coarse and fine levels $\rho(P_l, P_{l-1})$ become relatively weaker. Recall that for the lower bounds estimation, our target functional is given by

$$P_l = h_{\tau_l^*}(S_{\tau_l^*}) = \sum_{i=1}^{M_l} h_i(S_i) \mathbb{1}_{[t_i = \tau_l^*]} \tag{6.24}$$

where $\tau_l^* = \min\{\tau_i \in \{t_1, \cdots, t_{M_l}\} : h_i(S_i) \geq C_i(S_i)\}$ is the optimal stopping time under level $l$. Thus, we can see the functional $P_l$ is non-smooth with respect to every individual simulated sample path $S_i$. Thus, in order to increase the correlations of our functionals between adjacent coarse and fine levels, we need to find a way to smooth $P_l$. This can be done by assigning a weight to the payoffs at every time steps $h_i(S_i)$ and summing them up instead of using the single stopped value with $h_\tau^*(S_\tau^*)$. We construct the weights using a probabilistic approach. As we know, an investor will choose to exercise an American option when the payoff $h_i(S_i)$ exceeds the continuation value $C_i(S_i)$ for the first time. We slightly

changed this rule by assigning an artificial conditional probability of exercising at every time step given that option has not been exercised before. This probability should depend on the difference between $h_i(S_i)$ and $C_i(S_i)$. The greater difference gives larger probability. We define this probability by

$$\begin{cases} \mathbb{P}_i(S_i) = \Phi\left(\lambda\left(h_i(S_i) - C_i(S_i)\right)\right), & i = 1, \cdots, M_l - 1 \\ \mathbb{P}_{M_l}(S_{M_l}) = 1 \end{cases} \tag{6.25}$$

where $\Phi(x)$ is the standard normal cumulative distribution function and $\lambda$ is a controlling parameter for smoothing. And we define our target functional $P_l$ as

$$P_l = \sum_{i=1}^{M_l} W_i(S_i) h_i(S_i) = \sum_{i=1}^{M_l} \left\{ \mathbb{P}_i(S_i) \prod_{j=1}^{i-1} \left[1 - \mathbb{P}_j(S_j)\right] \right\} h_i(S_i) \tag{6.26}$$

By doing this, we essentially assign a weighting on payoffs along the entire Monte Carlo path. As we can see, when $\lambda \to \infty$, the probability defined above will be 1 for any case that $h_i(S_i) > C_i(S_i)$ and 0 otherwise. Thus, for extremely large $\lambda$, the functional $P_l$ defined by (6.26) is a coincidence with the original optimal stopping value defined by (6.24). And smaller $\lambda$ will smooth the payoff along the sample path and thus increase the correlations between different levels. In practice with MLMC, to keep the estimator's unbiasedness, we will replace the original functional with (6.26) and gradually increase $\lambda$ along the levels. At lower levels, we make $\lambda$ small to increase correlations between $P_l$ and $P_{l-1}$. At highest level $L$, we set a very large $\lambda$ so as to make $P_L$ a close approximation to the original problem and our MLMC estimator $\mathbb{E}[P_L] = \mathbb{E}[P_1] + \sum_{l=2}^{L} \mathbb{E}[P_l - P_{l-1}]$ will keep to be unbiased.

We can now enhance the correlations $\rho(P_l, P_{l-1})$ further by "Brownian Bridge interpolations". As we know, the functionals at coarse level $P_{l-1}$ are constructed on Monte Carlo points $S_i^{l-1}, i \in \{0, 1, \cdots, M_{l-1} = 2^{l-1}\}$. Comparing with the functionals at fine level $P_l$, which are constructed on points $S_i^l, i \in \{0, \cdots, M_l = 2^l\}$, we have fewer number of points at coarse level. These missing points will relatively reduce the correlations $\rho(P_l, P_{l-1})$. In order to make the points at coarse level coincident to those at fine level, we use Brownian Bridge interpolations to construct extra MC points between $S_i^{l-1}$ and $S_{i+1}^{l-1}$. By definition, the Brownian Bridge interpolation at time $t \in (i\Delta t_{l-1}, (i+1)\Delta t_{l-1})$ for $i \in \{0, \cdots, 2^{l-1} - 1\}$ is given by[19]

$$S_t^{l-1} = S_i^{l-1} + \alpha(S_{i+1}^{l-1} - S_i^{l-1}) + \sigma S_i^{l-1}\left(W_t^{l-1} - W_{i\Delta t_{l-1}}^{l-1} - \alpha \Delta W_{i+1}^{l-1}\right)$$

where, $\alpha = (t - i\Delta t_{l-1})/\Delta t_{l-1}$, and $\Delta W_{i+1}^{l-1}$ is the corresponding Brownian increments at coarse level. If we construct the interpolation exactly at the mid-point of the time interval, we will have $\alpha = 1/2$ and the interpolation now becomes

$$\begin{aligned} S_{i+\frac{1}{2}}^{l-1} =& S_i^{l-1} + \frac{1}{2}(S_{i+1}^{l-1} - S_i^{l-1}) + \sigma S_i^{l-1}\left(W_{(i+\frac{1}{2})\Delta t_{l-1}}^{l-1} - W_{i\Delta t_{l-1}}^{l-1} - \frac{1}{2}\Delta W_{i+1}^{l-1}\right) \\ =& \frac{1}{2}\left(S_i^{l-1} + S_{i+1}^{l-1}\right) + \sigma S_i^{l-1}\left(\Delta W_{2i+1}^{l} - \frac{1}{2}\left(\Delta W_{2i+1}^{l} + \Delta W_{2i+2}^{l}\right)\right) \\ =& \frac{1}{2}\left[S_i^{l-1} + S_{i+1}^{l-1} + \sigma S_i^{l-1}\left(\Delta W_{2i+1}^{l} - \Delta W_{2i+2}^{l}\right)\right] \quad \text{for } i \in \{0, \cdots, 2^{l-1} - 1\} \end{aligned} \tag{6.27}$$

where $\Delta W_{2i+1}^l$ and $\Delta W_{2i+2}^l$ are corresponding Brownian increments used to construct MC points at fine level. Finally, we construct our functional $P_{l-1}$ on the MC points $S_i^{l-1}$ together with the interpolations $S_{i+1/2}^{l-1}$ using the same weighted payoff functions defined in (6.26). By doing this, we essentially introduce extra points at level $l-1$ such that the time stepping is exactly same between levels $l$ and $l-1$ and the correlations between $P_l$ and $P_{l-1}$ are expected to be increased.

The same example of the American put option was used to test the ideas. We set the controlling parameter $\lambda = \{1, 5, 10, 50, 250, 1000, 100000\}$ respectively for the different levels. And also, for improving accuracies, we simulated the MC paths using the exact SDE solution (5.13) instead of using Euler-Maruyama scheme. The results are shown in Table 6.2. We tested both the probability weighted functions with and without Brownian Bridge interpolations. As expected, the probability weighted payoff functions are better fit into the MLMC method. For the case without Brownian Bridge interpolation, the correlations $\rho(P_l, P_{l-1})$ are relatively higher with the weighted payoff functions, especially for the in-the-money case ($S_0$=12.5), giving lower variances $\mathbb{V}_l$ at each level. As a result, the overall variances are reduced by about 40%$\sim$50% compared to those with the original non-smooth payoff functions. For the in-the-money case ($S_0$=12.5), the MLMC slightly overperformed the standard MC with the use of probability weighted payoff functions while it originally underperformed the standard MC with the non-smooth payoff functions. The most promising results were provided by the Brownian Bridge interpolations. As we can see from the results, although the interpolation method produced relatively smaller estimations of the option values, the correlations $\rho(P_l, P_{l-1})$ were significantly improved with the interpolations. The correlation coefficients were all very close to 1. As a result, the variances were reduced further, giving an average reduction of about 90% compared with the standard MC method. This confirms that the weighted payoff functions jointly with the Brownian Bridge interpolation technique will significantly improve the MLMC performances.

**CASE1: $S_0 = 12.5$ FDM = 7.528 $LSM_{std} = 7.456$ $\mathbb{V}_{std} = 8.81E-05$**

| | ORIGINAL | | | WEIGHTED | | | WEIGHTED INTERPOLATION | | |
|---|---|---|---|---|---|---|---|---|---|
| level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ |
| 1 | 858970 | 17.876 | N/A | 952024 | 12.220 | N/A | 2921330 | 12.2080 | N/A |
| 2 | 259758 | 3.365 | 0.902 | 212153 | 1.210 | 0.960 | 99807 | 0.0312 | 0.9986 |
| 3 | 170270 | 2.870 | 0.902 | 133481 | 0.961 | 0.961 | 43612 | 0.0113 | 0.9994 |
| 4 | 99154 | 1.974 | 0.922 | 79237 | 0.726 | 0.968 | 29881 | 0.0107 | 0.9994 |
| 5 | 57649 | 1.270 | 0.943 | 56089 | 0.657 | 0.969 | 26756 | 0.0179 | 0.9991 |
| 6 | 37267 | 1.039 | 0.949 | 40084 | 0.763 | 0.961 | 24230 | 0.0276 | 0.9985 |
| 7 | 22379 | 0.709 | 0.962 | 26184 | 0.552 | 0.970 | 25970 | 0.0223 | 0.9988 |
| | $LSM_{MLMC}$ | | 7.448 | $LSM_{MLMC}$ | | 7.513 | $LSM_{MLMC}$ | | 7.314 |
| | $\mathbb{V}_{MLMC}$ | | 1.52E-04 | $\mathbb{V}_{MLMC}$ | | 8.67E-05 | $\mathbb{V}_{MLMC}$ | | 7.78E-06 |

**CASE2: $S_0 = 20$ FDM = 2.733 $LSM_{std} = 2.639$ $\mathbb{V}_{std} = 1.09E-04$**

| | ORIGINAL | | | WEIGHTED | | | WEIGHTED INTERPOLATION | | |
|---|---|---|---|---|---|---|---|---|---|
| level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ |
| 1 | 1724635 | 14.287 | N/A | 1942950 | 9.446 | N/A | 4264546 | 9.4384 | N/A |
| 2 | 234803 | 0.535 | 0.982 | 145718 | 0.106 | 0.995 | 143460 | 0.0211 | 0.9990 |
| 3 | 135814 | 0.347 | 0.986 | 91552 | 0.084 | 0.996 | 60162 | 0.0065 | 0.9997 |
| 4 | 91932 | 0.325 | 0.986 | 78555 | 0.127 | 0.994 | 41234 | 0.0055 | 0.9998 |
| 5 | 51647 | 0.204 | 0.991 | 51840 | 0.099 | 0.996 | 24260 | 0.0059 | 0.9997 |
| 6 | 28273 | 0.139 | 0.994 | 29798 | 0.081 | 0.996 | 10012 | 0.0032 | 0.9999 |
| 7 | 18687 | 0.111 | 0.995 | 21687 | 0.085 | 0.996 | 8898 | 0.0051 | 0.9998 |
| | $LSM_{MLMC}$ | | 2.651 | $LSM_{MLMC}$ | | 2.636 | $LSM_{MLMC}$ | | 2.576 |
| | $\mathbb{V}_{MLMC}$ | | 3.15E-05 | $\mathbb{V}_{MLMC}$ | | 1.67E-05 | $\mathbb{V}_{MLMC}$ | | 3.74E-06 |

**CASE3: $S_0 = 25$ FDM = 1.306 $LSM_{std} = 1.280$ $\mathbb{V}_{std} = 5.86E-05$**

| | ORIGINAL | | | WEIGHTED | | | WEIGHTED INTERPOLATION | | |
|---|---|---|---|---|---|---|---|---|---|
| level | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ | $N_l$ | $\mathbb{V}_l$ | $\rho(P_l, P_{l-1})$ |
| 1 | 1749086 | 9.212 | N/A | 1822150 | 5.331 | N/A | 4068118 | 5.3286 | N/A |
| 2 | 291308 | 0.532 | 0.973 | 97492 | 0.027 | 0.998 | 180223 | 0.0198 | 0.9983 |
| 3 | 157964 | 0.279 | 0.982 | 76879 | 0.036 | 0.997 | 71114 | 0.0049 | 0.9996 |
| 4 | 86714 | 0.176 | 0.987 | 58904 | 0.049 | 0.996 | 30686 | 0.0026 | 0.9998 |
| 5 | 46791 | 0.107 | 0.992 | 55953 | 0.069 | 0.994 | 19447 | 0.0028 | 0.9998 |
| 6 | 30232 | 0.078 | 0.994 | 39296 | 0.077 | 0.993 | 10134 | 0.0025 | 0.9998 |
| 7 | 16042 | 0.050 | 0.996 | 22678 | 0.042 | 0.996 | 12595 | 0.0016 | 0.9999 |
| | $LSM_{MLMC}$ | | 1.281 | $LSM_{MLMC}$ | | 1.278 | $LSM_{MLMC}$ | | 1.212 |
| | $\mathbb{V}_{MLMC}$ | | 1.88E-05 | $\mathbb{V}_{MLMC}$ | | 9.54E-06 | $\mathbb{V}_{MLMC}$ | | 2.09E-06 |

Table 6.2: The MLMC pricing of American put options using weighted payoff functions and Brownian bridge interpolations. The parameters used in simulations are $K = 20$, $r = 5\%$, $\sigma = 0.4$, $T = 1$ and the number of basis functions $J$=5. The number of finest levels is set as $L$=7 and total compuational cost is fixed at $C$=$NM_L$=1e4×$2^7$. In the table, $N_l$, $\mathbb{V}_l$ and $\rho(P_l, P_{l-1})$ represent the number of sample paths at each level, the variance at each level and the correlation coefficients between the functionals at adjacent coarse and fine levels. $LSM_{MLMC}$ and $LSM_{std}$ represent the lower bound estimations using LSM method under MLMC and standard Mote Carlo respectively. $\mathbb{V}_{MLMC}$ and $\mathbb{V}_{std}$ are corresponding variances. The controlling parameters are set as $\lambda = \{1, 5, 10, 50, 250, 1000, 100000\}$ for different levels.

# Chapter 7

# Conclusions

This thesis reviewed and implemented a number of Monte Carlo based techniques for pricing American options. The applicabilities of these techniques and the dependencies on various simulation parameters were tested. A computing saving technique using orthogonal Hermite polynomials was suggested and tested. The Multilevel Mote Carlo method was applied with the algorithms and the effectiveness on variance reduction is discussed. A smoothing technique using probability weighted payoff functions was proposed to improve the Multilevel Monte Carlo performances for pricing American options.

Chapter 2 reviewed the main theoretical results of the valuation problems of American options and the basic problem formulations were derived through the dynamic programming principle.

Chapter 3 provided an introduction of Longstaff-Schwartz method (LSM) for approximating lower bounds of the prices of American options. The least-squares based regression method was employed to estimate the conditional expectations of the value of American options. A backward induction (Algorithm 3.2) was used to estimate the optimal regression coefficients and a forward induction (Algorithm 3.3) was used to estimate the corresponding lower bounds of the values of options through a sub-optimal stopping rule. The algorithms were tested using a simple example of an American put option written on a single non-dividend paying stock. It was found that the LSM method produced similar estimated values of American options compared with those given by Finite Difference Method (FDM). The variances of the LSM estimators depend on the number of Monte Carlo paths used in the simulations. Fewer simulated paths gave much higher variances. It was found that the variances of the LSM estimators tended to be larger when the option was close to at-the-money. The effects of number of time steps (exercise opportunities) were also tested. The results showed that, when the time steps increased, the LSM estimated values were generally getting closer to the results given by FDM. The variances were not noticeably affected by the number of time steps for most cases. But for deep-in-the-money and deep-out-of-money options, higher number of time steps resulted in lower variances.

Chapter 4 reviewed and implemented the Duality approach for estimating the upper bounds of the prices of American options through martingales constructed with the approx-

imated value functions and the corresponding conditional expectations estimated by nested Monte Carlo simulations (Algorithm 4.2). Various effects of the simulation parameters were tested. The simulations results showed that the duality estimations were generally larger than the results obtained either by the LSM or by the Finite Difference Method, confirming that the Duality approach provides estimations of upper bounds of the true prices. It was found that the variances of duality estimators were generally much smaller than those of the LSM. Again, fewer simulated Monte Carlo paths caused higher variances and the highest variances tended to be with the at-the-money options. The effects of number of sub-successors for nested MC simulations was also tested. The numerical results indicated that the estimated duality values decreased when the number of sub-successors increased, giving smaller upper bounds and thus tighter ranges containing the true price, conjunct with the lower bounds given by LSM. This was the direct outcome of the better estimations of the conditional expectations. The higher number of sub-successors also resulted in lower overall variances. The effects of the number of time steps were also investigated. The results showed that the duality estimations were getting lager with more time steps, causing wider upper-lower price ranges. The reason behind this phenomena is unclear. The variances of the duality estimators were not noticeably affected by the number of time steps for most cases.

Chapter 5 suggested a computing saving technique for pricing American options. By converting the underlying asset price into a standard normal random variable and using the Hermite polynomials, we were able to construct a set of regression basis functions which are orthogonal to each other with respect to the natural distribution of the underlying asset price. This orthogonality made the expectation $\mathbb{E}\left[\psi(S)\psi(S)^\top\right]$ to be an identity matrix $I$, which in turn simplified the procedure of estimating the regression coefficients $\beta$ and thus provided large amounts of computing saving. A simple numerical example showed that the estimated matrix $\hat{B}_\psi = \hat{\mathbb{E}}\left[\psi(S)\psi(S)^\top\right]$ was very similar to an identity matrix $I$. The quality of this similarity improved with larger number of Monte Carlo paths. The estimated regression coefficients $\hat{\beta}$ with this simplified technique were reasonably close to those obtained by the standard approach but with much faster computing speed. The technique was then tested with both the algorithms of LSM and Duality approaches. Simulation results showed that this simplified technique produced similar accuracies as those under the standard approaches but with much faster speed. While giving similar accuracies, the use of orthogonal basis functions generally resulted in smaller lower bounds from LSM approach and larger upper bounds from Duality approach, indicating slightly wider lower-upper price ranges. The effects of time steps with this simplification were also investigated. It was found that the results given by the simplified approach using the orthogonal basis functions were very sensitive with respect to time steps. When the number of time steps getting larger, the results obtained from the simplified orthogonal approach started to give unacceptable accuracies if we worked with a relatively low number of main Monte Carlo paths. This was due to the built-up errors with over-simplification

coming from the relatively poor similarity between $\mathbb{E}\left[\psi(S)\psi(S)^{\top}\right]$ and the identity $I$ and the backward induction for estimating regression coefficients. This problem can be resolved by using larger number of main Monte Carlo paths.

Chapter 6 employed the Multilevel Monte Carlo (MLMC) technique with pricing American options. The effects in variance reductions for both the primal and the dual approaches were tested. The numerical examples confirmed that MLMC approach produced unbiased estimations for both the lower bounds using LSM method and the upper bounds using the Duality method. As for the lower bounds estimations through LSM approach, the variances were noticeably reduced under MLMC with the cases of at-the-money option and out-of-the-money option and gave a reduction up to 70% with respect to standard Monte Carlo. This is mainly because of the high correlations between the functionals at adjacent coarse and fine levels $\rho(P_l, P_{l-1})$. However, the situation changed for the in-the-money option, in which case the correlations became weaker and the overall variance was about 170% as large as that of the standard counterpart. The most surprising fact was that the MLMC method worked poorly with the Duality estimations. Although it still gave an unbiased estimation, its variances were much higher if compared with those given by standard Monte Carlo. The main reason behind is that there were much weaker correlations exist between $P_l$ and $P_{l-1}$ for the Duality estimations, indicating that the duality values are relatively weakly correlated with different number of exercise opportunities. A smoothing technique using artificial-probability weighted payoff functions jointly with Brownian Bridge interpolations was suggested to improve MLMC performances with the lower bounds estimations through LSM algorithm. The numerical results showed that the probability weighted payoff function, which closely approximated the non-smooth optimal stopped value obtained from LSM at higher levels, better fitted into the MLMC method. The correlations $\rho(P_l, P_{l-1})$ were relatively higher with the weighted payoff functions, especially for the in-the-money case. As a result, the overall variances were reduced by about 40%$\sim$50% compared to those with the original non-smooth payoff functions. The most promising results were obtained when we employed the method of the weighted payoff functions together with the Brownian Bridge interpolations for coarse levels. The correlations $\rho(P_l, P_{l-1})$ were significantly improved with values close to 1 for all the cases. As a result, the variances were reduced further, giving an average reduction of about 90% compared with the standard MC method. This confirms that the weighted payoff functions jointly with the Brownian Bridge interpolation technique will significantly improve the MLMC performances.

# Appendix A

# Matlab Codes

## A.1 Sub-codes and Functions

### A.1.1 Cumulative Normal Density Function

```
%Cumulative Normal Density Functions

function [ Y ] = NormalCDF( X )

Y=0.5*(1+erf(X/sqrt(2)));

end
```

### A.1.2 Regression Basis Functions Using Laguerre Polynomials

```
% J: number of basis functions

function Y=Basis_laguerre(X,J)

    Y=zeros(J,length(X));

    for j=1:J
        Y(j,:)=Laguerre(X,j-1);
    end

end

function Y=Basis_weighted_laguerre(X,J)

    Y=zeros(J,length(X));

    for j=1:J
        Y(j,:)=exp(-X./2).*Laguerre(X,j-1);
    end

end

function Y = Laguerre(X, k)
```

```
    if (k==0)
        Y=1;
    elseif (k==1)
        Y=1-X;
    else
        Y=(1/k)*((2*k-1-X).*Laguerre(X,k-1)-(k-1)*Laguerre(X,k-2));
    end
end
```

### A.1.3 Orthogonal Basis Functions Based on Hermite Polynomials

```
% r: interest rate
% q: dividend rate
% sig: volatility
% T: maturity
% M: number of time intervals
% S: stock price process S(M+1,N)
% S0: initial stock price
% m: current time index
% J: number of basis functions

function Phi=Basis_Hermite_converted(r,q,sig,T,M,S,S0,m,J)

    Phi=zeros(J,length(S));

    dt=T/M;

    i=m-1;

    mu=log(S0)+(r-q-0.5*sig^2)*i*dt;

    sigma=sig*sqrt(i*dt);

    Y=(log(S)-mu)/sigma;

    Phi(1,:)=1;

    Phi(2,:)=Y;

    for j=3:J
        Phi(j,:)=sqrt(1/(j-1))*Y.*Phi(j-1,:)-sqrt((j-2)/(j-1))*Phi(j-2,:);
    end

end
```

### A.1.4 Discounted Payoff Functions of American Put Option

```
% Discounted Payoff function of put
% K: strike
% M: number of time intervals
% N: number of MC paths

```

```
function h = Payoff_put(r,T,S,K,M,N)

    dt=T/M;
    h=zeros(M,N);
    h(M,:)=exp(-r*T)*max(K-S(M+1,:),0);

    for  i=1:M
        m=i+1;   %current time index (exercise dates)
        time=(m-1)*dt;
        h(i,:)=exp(-r*time)*max(K-S(m,:),0);
    end

end
```

### A.1.5 Estimating Regression Coefficients Using Backward Induction

```
%-------Backward induction for estimating regression coefficients--------
%-------The code is written on simple laguerre polynomials, which can-----
%-------be replaced with other interested basis functions.-----------------
%-------S: Stock price process S(M+1,N);  h: Payoff functions h(M,N)
%-------Beta: Regression coefficients Beta(J,M)

function Beta = Regression_Beta(S,h,M,N,J)

    Beta=zeros(J,M);   % regression coefficients
    c=zeros(M,N);    % continuation values

    V=h(M,:);  % set the value function to the terminal payoff

    for  i=(M-1):-1:1

        m=i+1; % current time index (exercise dates)

        % basis functions, different basis functions can
        % be used here by replacing with corresponding functions
        Phi=Basis_laguerre(S(m,:),J);

        sum1=0;
        sum2=0;

        for j=1:N
            sum1=sum1+Phi(:,j)*Phi(:,j)';
            sum2=sum2+Phi(:,j)*V(j);
        end

        Beta(:,i)=sum1\sum2;

        c(i,:)=(Beta(:,i)')*Phi;


        V=max(h(i,:),c(i,:));

    end
  end
```

### A.1.6 Regression Coefficients Using Simplified Approach Based on Orthogonality

```
function Beta_direct = Regression_Beta_Hermite_direct(r,q,sig,T,S,h,M,N,J)


    Beta_direct=zeros(J,M);


    c=zeros(M,N);    %continuation values


    V=h(M,:);  %set the value function to the terminal payoff

    for  i=(M-1):-1:1

        m=i+1; %current time index (exercise dates)
        Phi_direct=Basis_Hermite_converted(r,q,sig,T,M,S(m,:),S(1,:),m,J); ...
              %basis functions

         sum1=0;

        for j=1:N
            sum1=sum1+Phi_direct(:,j)*V(j);
        end

        Beta_direct(:,i)=(sum1)/N;

        c(i,:)=(Beta_direct(:,i)')*Phi_direct;

        V=max(h(i,:),c(i,:));

    end

end
```

### A.1.7 LSM Estimator

```
%------LSM estimator using optimal stopping rule using standard MC------
%------M: number of time intervals; N: number of MC paths
%-------S: Stock price process S(M+1,N);  h: Payoff functions h(M,N)
%-------Beta: Regression coefficients Beta(J,M)

function [Vec_rule,V,c]= LSM_Low_std_MC(S,h,M,N,J,Beta)

    V=zeros(M,N); %value function vecter
    c=zeros(M,N);    %continuation values

    for  i=1:M

        m=i+1;   %current time index (exercise dates)


```

```
16          if (i==M) %final time step treatment
17              V(i,:)=h(i,:);
18          else
19              Phi=Basis_laguerre(S(m,:),J);  % basis functions, can be ...
                    replaced with other basis functions
20              c(i,:)=(Beta(:,i)')*Phi;
21              V(i,:)=max(h(i,:),c(i,:));
22           end
23
24      end
25
26      Vec_rule=zeros(1,N); %value function vecter V(i)  optimal stopping values
27
28      for n=1:N
29          Vec_rule(n)=h(find(h(:,n)>=c(:,n),1),n);
30      end
31
32  end
```

### A.1.8 Duality Estimator

```
1   %------Estimating Duality value through martingales using standard MC------
2   %------M: number of time intervals; N: number of MC paths
3   %-------S: Stock price process S(M+1,N);  h: Payoff functions h(M,N)
4   %-------Beta: Regression coefficients Beta(J,M)
5   %-------NS: number of MC sub-sucessors for nested MC
6
7   function [Vec_da,V,c]= LSM_Dual_std_MC(r,q,sig,K,T,S,h,M,N,J,NS,Beta)
8
9       dt=T/M;
10
11      V=zeros(M,N); %value function vecter
12      c=zeros(M,N);    %continuation values
13
14      Mar=zeros(M,N); %duality martingales  M(i)
15      Da=zeros(M,N); %duality values (h(i)-M(i))
16
17
18      for  i=1:M
19
20          m=i+1;   %current time index (exercise dates)
21          time=i*dt;
22
23          if (i==M) %final time step treatment
24              V(i,:)=h(i,:);
25          else
26              Phi=Basis_laguerre(S(m,:),J); % basis functions, can be ...
                    replaced with other basis functions
27              c(i,:)=(Beta(:,i)')*Phi;
28              V(i,:)=max(h(i,:),c(i,:));
29           end
30
31          %generating sub paths for estimating dual value
32          Ss=zeros(NS,N);    % stock prices of sub-paths
33          Vs=zeros(NS,N);  %value functions of sub-paths
```

```
34          cs=zeros(NS,N);      %continuation values of sub-paths
35          hs=zeros(NS,N);      %payoff functions of sub-paths
36
37          for k=1:NS
38              dW=sqrt(dt)*randn(1,N);
39              Ss(k,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);  % generating ...
                    sub-paths using the previous stock price
40              hs(k,:)=exp(-r*time)*max(K-Ss(k,:),0);
41              if (i==M)
42                  Vs(k,:)=hs(k,:);
43              else
44                  Phi=Basis_laguerre(Ss(k,:),J); % basis functions, can be ...
                        replaced with other basis functions
45                  cs(k,:)=(Beta(:,i)')*Phi;
46                  Vs(k,:)=max(hs(k,:),cs(k,:));
47              end
48          end
49
50          Diff=V(i,:)-sum(Vs,1)/NS;    % martingale differential
51          if (i==1)
52              Mar(i,:)=Diff;
53          else
54              Mar(i,:)=Mar(i-1,:)+Diff;
55          end
56
57          Da(i,:)=h(i,:)-Mar(i,:);
58
59      end
60
61      Vec_da=max(Da,[],1); % vecoer storing duality values
62
63  end
```

### A.1.9 LSM Estimator Using Probability Weighted Payoff Functions

```
1   % LSM Estimator using smoothed Probability Weighted Payoff Functions
2   % lamda: tsmoothing controlling paramter
3
4   function [Vec_rule,V,c]= LSM_weighted(S,h,M,N,J,Beta,lamda)
5
6       V=zeros(M,N); %value function vecter
7       c=zeros(M,N);     %continuation values
8       P=zeros(M,N);    %conditional probability of exercise
9
10      sum1=0;
11      prod=1;
12
13      for  i=1:M
14
15          m=i+1;    %current time index (exercise dates)
16
17          if (i==M) %final time step treatment
18              V(i,:)=h(i,:);
19              c(i,:)=V(i,:);
20          else
```

```
21              Phi=Basis_laguerre(S(m,:),J);
22              c(i,:)=(Beta(:,i)')*Phi;
23              V(i,:)=max(h(i,:),c(i,:));
24          end
25
26          if (i == M)
27              P(i,:)=1;
28          else
29              P(i,:)=NormalCDF(lamda*(h(i,:)-c(i,:)));
30          end
31
32          if (i ~=1)
33              prod=prod.*(1-P(i-1,:));
34          end
35
36          sum1=sum1+h(i,:).*P(i,:).*prod;
37
38      end
39
40      Vec_rule=sum1; %value function vecter V(i)  weighted payoff functions
41
42  end
```

### A.1.10 FDM penalty method

```
1  % Pricing Amercian Option Using FDM Penalty Method
2
3  function V=American_put_FDM_penalty(S0,K,r,sigma,T)
4
5  %
6  % look at variation with J, with N=J
7  %
8
9  Smax = 10*S0;
10 J    = 1000;
11 N    = 500;
12
13 S  = linspace(0,Smax,J+1)';
14
15 %
16 % set up various constants
17 %
18
19 dS     = S(2)-S(1);
20 dt     = T/N;
21
22 lambda = 0.5*sigma^2*S.^2/dS^2;
23 gamma  = 0.5*r*S/dS;
24
25 %
26 % define coefficients of tridiagonal matrix
27 %
28
29 a =     lambda - gamma;
30 b = - 2*lambda - r;
```

```
c =     lambda + gamma;

%
% special treatment of last point, based on d^2u/dS^2=0
%

a(end) = - 2*gamma(end);
b(end) =   2*gamma(end) - r;
c(end) = 0;

%
% call function to create sparse tri-diagonal matrix
%

L = sparse_tri(a,b,c);
I = speye(length(S));

%
% now do time-marching
%

u = max(0,K-S);
payoff=u;

cnt = zeros(1,N);
ue  = zeros(1,N+1);
ue(1) = K;

rho = 10000;

for n = 1:N
  rhs = (I+0.5*dt*L)*u;

  u_old = u;
  conv  = 0;
  count = 0;
  while ~conv
    P  = max(0,K-S-u);
    PP = spdiags( -sign(P), 0, J+1,J+1);
    du = (I-0.5*dt*L-rho*dt*PP) \ (rhs-(I-0.5*dt*L)*u+rho*dt*P);
    u  = u + du;
    conv  = max(abs(du))<1e-10;
    count = count + 1;
  end

  cnt(n) = count;

  ue(n+1) = S(min(find(u > (K-S))));

end

u0 = u(1 + J*S0/Smax);

V=u0;

%
% function to construct tridiagonal matrix
% from 3 vectors defining its diagonals
%
```

```
90
91 function A = sparse_tri(a,b,c)
92
93 A = spdiags([c b a],[-1 0 1],length(a),length(a))';
```

## A.2 Main Codes

### A.2.1 Lower Bounds Estimation Using LSM

```
1  %Estimating Lower Bounds for American Put Option
2  %using LSM method
3
4  clear all; close all; clc;
5  rng('default')
6
7  % problem parameters
8  r=0.05;       %interest rate
9  q=0.0;        %dividend rate
10 sig=0.4;     %volativity
11 S0=20;     %initial stock values
12 K=20;       %strike
13 T=1;            %maturity
14
15 N=1e4;  % number of paths for standard MC
16 M=2^5;  % number of time steps
17
18 J=5;        %number of basis functions
19
20 dt=T/M;  %length of time step
21
22 V_fdm=American_put_FDM_penalty(S0,K,r,sig,T);
23
24 %Generating independent paths using Euler-Maruyama scheme for estimating
25 %regression coefficients using standard MC
26 S=zeros(M+1,N);  % every column corresponds to an individual path
27
28 S(1,:)=S0*ones(1,N);
29
30 for m=2:M+1
31      dW=sqrt(dt)*randn(1,N);
32      S(m,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);
33 end
34
35 h = Payoff_put(r,T,S,K,M,N); %payoff_functions
36
37 %-------Calculating Regression Coefficients with Longstaff ...
       Algorithm-------------
38 Beta = Regression_Beta(S,h,M,N,J);
39
40 %------Estimating value using optimal stopping rule using standard MC
41 %------through the 2nd set of independent Monte Carlo paths--
42
43 % generating the 2nd set of main paths
44 for m=2:M+1
45       dW=sqrt(dt)*randn(1,N);
```

```
46        S(m,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);
47  end
48
49  h = Payoff_put(r,T,S,K,M,N); %payoff_functions
50
51  % calculating optimal stopping value
52  [Vec_rule,V,c]= LSM_Low_std_MC(S,h,M,N,J,Beta);
53
54  V_rule=sum(Vec_rule)/N;
55
56  Var_rule=(sum(Vec_rule.^2)/N-V_rule^2)/N;
57
58  disp('The value of option using FDM');
59  disp(V_fdm);
60  disp('The value of option using standard LSM');
61  disp(V_rule);
62  disp('Variance LSM estimator')
63  disp(Var_rule);
```

### A.2.2 Upper Bounds Estimation Using Duality Method

```
1   %Estimating Upper Bounds for American Put Option
2   %using Duality method
3
4   clear all; close all; clc;
5   rng('default')
6
7   % problem parameters
8   r=0.05;       %interest rate
9   q=0.0;        %dividend rate
10  sig=0.4;     %volativity
11  S0=20;      %initial stock values
12  K=20;       %strike
13  T=1;          %maturity
14
15  N=1e4;  % number of paths for standard MC
16  M=2^5;  % number of time steps
17  NS=200; %number of sub-paths for duality calculation
18  J=5;        %number of basis functions
19
20  dt=T/M;  %length of time step
21
22  V_fdm=American_put_FDM_penalty(S0,K,r,sig,T);
23
24  %Generating independent paths using Euler-Maruyama scheme for estimating
25  %regression coefficients using standard MC
26  S=zeros(M+1,N);  % every column corresponds to an individual path
27
28  S(1,:)=S0*ones(1,N);
29
30  for m=2:M+1
31       dW=sqrt(dt)*randn(1,N);
32       S(m,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);
33  end
34
```

```
h = Payoff_put(r,T,S,K,M,N); %payoff_functions

%--------Calculating Regression Coefficients with Longstaff ...
    Algorithm----------------
Beta = Regression_Beta(S,h,M,N,J);

%-------Estimating duality value using standard MC
%-------through the 2nd set of independent Monte Carlo paths---

% generating the 2nd set of main paths
for m=2:M+1
      dW=sqrt(dt)*randn(1,N);
      S(m,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);
end

h = Payoff_put(r,T,S,K,M,N); %payoff_functions

% calculating Duality values
[Vec_dual,V,c]= LSM_Dual_std_MC(r,q,sig,K,T,S,h,M,N,J,NS,Beta);

V_dual=sum(Vec_dual)/N;

Var_dual=(sum(Vec_dual.^2)/N-V_dual^2)/N;

disp('The value of option using FDM');
disp(V_fdm);
disp('The Duality value of option');
disp(V_dual);
disp('Variance of Duality estimator')
disp(Var_dual);
```

### A.2.3 Multilevel Monte Carlo for Pricing American Put Options

```
% MLMC for pricing American put options
% The code is written on LSM estimator and Laguerre basis functions.
% For estimating Duality values, users can repalce the LSM estimators with
% the Duality estimators. Users can also replace the Laguerre basis ...
    functions with
% other interested basis functions.

clear all; close all; clc;
rng('default')

% problem parameters
r=0.05;      %interest rate
q=0.0;       %dividend rate
sig=0.4;    %volativity
S0=20; %initial stock values
K=20;      %strike
T=1;           %maturity

V_fdm=American_put_FDM_penalty(S0,K,r,sig,T);

L=7;   % number of levels of MLMC
N=1e5;  % number of paths for standard MC
```

```
M=2^L;  % number of time steps
C=N*M;   %Compuational cost for standard MC

NS=100; %number of sub-paths for duality calculation

J=5;       %number of basis functions

dt=T/M;  %length of time step

%Generating independent paths using Euler-Maruyama scheme for estimating
%regression coefficients using standard MC
S=zeros(M+1,N);  % every column corresponds to an individual path

S(1,:)=S0*ones(1,N);

for m=2:M+1
    dW=sqrt(dt)*randn(1,N);
    S(m,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);
end

h = Payoff_put(r,T,S,K,M,N); %payoff_functions

%-------Calculating Regression Coefficients ---------------
Beta = Regression_Beta(S,h,M,N,J);

% generating the 2nd set of main paths
for m=2:M+1
    dW=sqrt(dt)*randn(1,N);
    S(m,:)=S(m-1,:).*(1+(r-q)*dt+sig*dW);
end

h = Payoff_put(r,T,S,K,M,N); %payoff_functions

% calculating option values using standard MC
[Vec_rule,V,c]= LSM_Low_std_MC(S,h,M,N,J,Beta);

V_rule=sum(Vec_rule)/N;

Var_rule=(sum(Vec_rule.^2)/N-V_rule^2)/N;

%------MLMC starts from here--------

Nl_rule=1e4*ones(1,L); %number of independent Monte Carlo paths at each ...
    level (initial value 1e4 for pilot run)

Ml=zeros(1,L);   % number of time steps at each level
for l=1:L
    Ml(l)=2^l;
end

Yl_rule=zeros(1,L);  % this vector records estimated option values at ...
    each level
Varl_rule=zeros(1,L);  % this vector records variances of option value at ...
    each level
Cov_level=zeros(1,L); %this vector records co-variance between fine and ...
    coarse levels
Corr_level=zeros(1,L); %this vector records correlation coefficients ...
    between fine and coarse levels
```

```
75 Sum_rule=0.0;    % the sum required in the calculation of optimal number ...
       of paths
76
77
78 % MLMC pilot run
79 Nl=Nl_rule;
80 for l=1:L
81     S_fine=zeros(Ml(l)+1,Nl(l));   % stock path at level l
82     S_fine(1,:)=S0*ones(1,Nl(l));
83     Beta_fine=zeros(J,Ml(l));     % recording regression coefficients at ...
           fine level
84
85     for i=1:Ml(l)
86         ind=i*Ml(L)/Ml(l);    % global index corresponding to the global ...
               highest level
87         Beta_fine(:,i)=Beta(:,ind);   % copying golobal Betas to the ...
               local fine level (l)
88     end
89
90     if (l ~= 1)
91         S_coarse=zeros(Ml(l-1)+1,Nl(l));  %stock path at level (l-1)
92         S_coarse(1,:)=S0*ones(1,Nl(l));
93         Beta_coarse=zeros(J,Ml(l-1));    % recording regression ...
               codfficients at coarse level
94
95         for i=1:Ml(l-1)
96              ind=i*Ml(L)/Ml(l-1);    % global index corresponding to the ...
                   global highest level
97             Beta_coarse(:,i)=Beta(:,ind);   % copying golobal Betas to ...
                   the local coarse level (l-1)
98         end
99      end
100
101    dt_l=T/Ml(l);  % time interval at level l (fine level)
102
103    % generating the main MC paths
104    if (l ~=1)
105        for m=2:Ml(l-1)+1
106            m_fine=2*m-2;
107            dW1=sqrt(dt_l)*randn(1,Nl(l));
108            S_fine(m_fine,:)=S_fine(m_fine-1,:).*(1+(r-q)*dt_l+sig*dW1);
109            dW2=sqrt(dt_l)*randn(1,Nl(l));
110            S_fine(m_fine+1,:)=S_fine(m_fine,:).*(1+(r-q)*dt_l+sig*dW2);
111            S_coarse(m,:)=S_coarse(m-1,:).*(1+(r-q)*2*dt_l+sig*(dW1+dW2));
112        end
113        h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discoundted ...
               payoff functions at fine level (l)
114        h_coarse=Payoff_put(r,T,S_coarse,K,Ml(l-1),Nl(l));  % discounted ...
               payoff functions at coarse level (l-1)
115    else
116        for m=2:Ml(l)+1
117            dW=sqrt(dt_l)*randn(1,Nl(l));
118            S_fine(m,:)=S_fine(m-1,:).*(1+(r-q)*dt_l+sig*dW);
119        end
120        h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discounted ...
               payoff functions at fine level (l)
121    end
122
123    if (l==1)
```

```
124          [Vec_rule_ML,V_ML,c_ML]=...
125              LSM_Low_std_MC(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine);
126      else
127          [Vec_rule_fine,V_fine,c_fine]=...
128              LSM_Low_std_MC(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine);
129          [Vec_rule_coarse,V_coarse,c_coarse]=...
130              LSM_Low_std_MC(S_coarse,h_coarse,Ml(l-1),Nl(l),J,Beta_coarse);
131          Vec_rule_ML=Vec_rule_fine-Vec_rule_coarse;
132      end
133
134      Yl_rule(l)=sum(Vec_rule_ML)/Nl(l);
135      Varl_rule(l)=(sum(Vec_rule_ML.^2)/Nl(l)-Yl_rule(l)^2);
136
137
138      Sum_rule=Sum_rule+sqrt(Varl_rule(l)*T*Ml(l));
139
140  end
141
142  for l=1:L
143      Nl_rule(l)=int64(C*sqrt(Varl_rule(l)*T/Ml(l))/Sum_rule);
144  end
145
146  % MLMC optimal run
147  Nl=Nl_rule;
148  for l=1:L
149      S_fine=zeros(Ml(l)+1,Nl(l));   % stock path at level l
150      S_fine(1,:)=S0*ones(1,Nl(l));
151      Beta_fine=zeros(J,Ml(l));     % recording regression coefficients at ...
             fine level
152
153      for i=1:Ml(l)
154          ind=i*Ml(L)/Ml(l);    % global index corresponding to the global ...
                 highest level
155          Beta_fine(:,i)=Beta(:,ind);   % copying golobal Betas to the ...
                 local fine level (l)
156      end
157
158      if (l ~= 1)
159          S_coarse=zeros(Ml(l-1)+1,Nl(l));  %stock path at level (l-1)
160          S_coarse(1,:)=S0*ones(1,Nl(l));
161          Beta_coarse=zeros(J,Ml(l-1));    % recording regression ...
                 codfficients at coarse level
162
163          for i=1:Ml(l-1)
164               ind=i*Ml(L)/Ml(l-1);    % global index corresponding to the ...
                     global highest level
165              Beta_coarse(:,i)=Beta(:,ind);   % copying golobal Betas to ...
                     the local coarse level (l-1)
166          end
167       end
168
169      dt_l=T/Ml(l);  % time interval at level l (fine level)
170
171      % generating the main MC paths
172      if (l ~=1)
173          for m=2:Ml(l-1)+1
174              m_fine=2*m-2;
175              dW1=sqrt(dt_l)*randn(1,Nl(l));
176              S_fine(m_fine,:)=S_fine(m_fine-1,:).*(1+(r-q)*dt_l+sig*dW1);
```

```
177             dW2=sqrt(dt_l)*randn(1,Nl(l));
178             S_fine(m_fine+1,:)=S_fine(m_fine,:).*(1+(r-q)*dt_l+sig*dW2);
179             S_coarse(m,:)=S_coarse(m-1,:).*(1+(r-q)*2*dt_l+sig*(dW1+dW2));
180         end
181         h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discoundted ...
                payoff functions at fine level (l)
182         h_coarse=Payoff_put(r,T,S_coarse,K,Ml(l-1),Nl(l));  % discounted ...
                payoff functions at coarse level (l-1)
183     else
184         for m=2:Ml(l)+1
185             dW=sqrt(dt_l)*randn(1,Nl(l));
186             S_fine(m,:)=S_fine(m-1,:).*(1+(r-q)*dt_l+sig*dW);
187         end
188         h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discounted ...
                payoff functions at fine level (l)
189     end
190
191     if (l==1)
192          [Vec_rule_ML,V_ML,c_ML]=...
193              LSM_Low_std_MC(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine);
194     else
195         [Vec_rule_fine,V_fine,c_fine]=...
196             LSM_Low_std_MC(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine);
197         [Vec_rule_coarse,V_coarse,c_coarse]=...
198             LSM_Low_std_MC(S_coarse,h_coarse,Ml(l-1),Nl(l),J,Beta_coarse);
199         Vec_rule_ML=Vec_rule_fine-Vec_rule_coarse;
200         cov_l=cov(Vec_rule_fine,Vec_rule_coarse,1);
201         Cov_level(l)=cov_l(1,2);
202         Corr_level(l)=cov_l(1,2)/sqrt(cov_l(1,1)*cov_l(2,2));
203     end
204
205     Yl_rule(l)=sum(Vec_rule_ML)/Nl(l);
206     Varl_rule(l)=(sum(Vec_rule_ML.^2)/Nl(l)-Yl_rule(l)^2);
207
208     Sum_rule=Sum_rule+sqrt(Varl_rule(l)*T*Ml(l));
209
210 end
211
212 V_rule_MLMC=sum(Yl_rule);
213
214 Var_rule_MLMC=0;
215
216 for l=1:L
217     Var_rule_MLMC=Var_rule_MLMC+Varl_rule(l)/Nl(l);
218 end
219
220 disp('The value of option using FDM:');
221 disp(V_fdm);
222 disp('The value of option using standard MC');
223 disp(V_rule);
224 disp('The value of option using MLMC');
225 disp(V_rule_MLMC);
226 disp('Variance of standard MC')
227 disp(Var_rule);
228 disp('Variance of MLMC')
229 disp(Var_rule_MLMC);
230 disp('Number of MC paths at each level')
231 disp(Nl_rule);
232 disp('Variances at each level')
```

```
disp(Varl_rule);
disp('Correlations between fine and coarse levels');
disp(Corr_level);
```

### A.2.4 Multilevel Monte Carlo Using Probability Weighted Payoff Functions jointly with Brownian Bridge Interpolations

```
% MLMC for pricing American put options
% based on smoothed probability weighted payoff functions
% and the Brownian Bridge Interpolations

clear all; close all; clc;
rng('default')

% problem parameters
r=0.05;      %interest rate
q=0.0;       %dividend rate
sig=0.4;    %volativity
S0=20; %initial stock values
K=20;      %strike
T=1;           %maturity

V_fdm=American_put_FDM_penalty(S0,K,r,sig,T);

L=7;   % number of levels of MLMC
N=1e5;  % number of paths for standard MC
M=2^L;  % number of time steps
C=N*M;   %Compuational cost for standard MC

NS=100; %number of sub-paths for duality calculation

J=5;       %number of basis functions

lamda=[1,5,10,50,250,1000,100000];   % parameters for smoothing

dt=T/M;  %length of time step

%Generating independent paths
S=zeros(M+1,N);  % every column corresponds to an individual path

S(1,:)=S0*ones(1,N);

for m=2:M+1
    dW=sqrt(dt)*randn(1,N);
    S(m,:)=S(m-1,:).*exp((r-q-0.5*sig^2)*dt+sig*dW);
end

h = Payoff_put(r,T,S,K,M,N); %payoff_functions

%-------Calculating Regression Coefficients with Longstaff ...
    Algorithm-------------
Beta = Regression_Beta(S,h,M,N,J);

% generating the 2nd set of main paths
for m=2:M+1
```

```
48      dW=sqrt(dt)*randn(1,N);
49      S(m,:)=S(m-1,:).*exp((r-q-0.5*sig^2)*dt+sig*dW);
50  end
51
52  h = Payoff_put(r,T,S,K,M,N); %payoff_functions
53
54  % calculating optimal stopping value and duality value
55  [Vec_rule,V,c]= LSM_Low_std_MC(S,h,M,N,J,Beta);
56
57  V_rule=sum(Vec_rule)/N;
58
59  Var_rule=(sum(Vec_rule.^2)/N-V_rule^2)/N;
60
61  %------MLMC starts from here--------
62
63  Nl_rule=1e4*ones(1,L); %number of independent Monte Carlo paths at each ...
        level (initial value 1e4 for pilot run)
64
65  Ml=zeros(1,L);    % number of time steps at each level
66  for l=1:L
67      Ml(l)=2^l;
68  end
69
70  Yl_rule=zeros(1,L);  % this vector records estimated option values at ...
        each level
71  Varl_rule=zeros(1,L);  % this vector records variances of option value at ...
        each level
72  Cov_level=zeros(1,L); %this vector records co-variance between fine and ...
        coarse levels
73  Corr_level=zeros(1,L); %this vector records correlation coefficients ...
        between fine and coarse levels
74  Sum_rule=0.0;     % the sum required in the calculation of optimal number ...
        of paths
75
76  % MLMC pilot run
77   Nl=Nl_rule;
78  for l=1:L
79
80      S_fine=zeros(Ml(l)+1,Nl(l));    % stock path at level l
81      S_fine(1,:)=S0*ones(1,Nl(l));
82      Beta_fine=zeros(J,Ml(l));      % recording regression coefficients at ...
            fine level
83
84      for i=1:Ml(l)
85          ind=i*Ml(L)/Ml(l);     % global index corresponding to the global ...
                highest level
86          Beta_fine(:,i)=Beta(:,ind);    % copying golobal Betas to the ...
                local fine level (l)
87      end
88
89      if (l ~= 1)
90          S_coarse=zeros(Ml(l-1)+1,Nl(l));  %stock path at level (l-1)
91          S_inter=zeros(Ml(l)+1,Nl(l)); % stock path at level (l-1) ...
                combined with interpolations
92          S_coarse(1,:)=S0*ones(1,Nl(l));
93          S_inter(1,:)=S0*ones(1,Nl(l));
94          Beta_coarse=zeros(J,Ml(l-1));     % recording regression ...
                codfficients at coarse level
95
```

```
96          for i=1:Ml(l-1)
97               ind=i*Ml(L)/Ml(l-1);    % global index corresponding to the ...
                     global highest level
98              Beta_coarse(:,i)=Beta(:,ind);   % copying golobal Betas to ...
                    the local coarse level (l-1)
99          end
100      end
101
102     dt_l=T/Ml(l);  % time interval at level l (fine level)
103
104     % generating the main MC paths
105     if (l ~=1)
106         for m=2:Ml(l-1)+1
107             m_fine=2*m-2;
108             dW1=sqrt(dt_l)*randn(1,Nl(l));
109             S_fine(m_fine,:)=...
110                 S_fine(m_fine-1,:).*exp((r-q-0.5*sig^2)*dt_l+sig*dW1);
111             dW2=sqrt(dt_l)*randn(1,Nl(l));
112             S_fine(m_fine+1,:)=...
113                 S_fine(m_fine,:).*exp((r-q-0.5*sig^2)*dt_l+sig*dW2);
114             S_coarse(m,:)=...
115                 S_coarse(m-1,:).*exp((r-q-0.5*sig^2)*2*dt_l+sig*(dW1+dW2));
116
117             %Brownian interpolations
118             S_inter(m_fine,:)=...
119                 0.5*(S_coarse(m,:)+S_coarse(m-1,:)...
120                    +(S_coarse(m-1,:).*sig).*(dW1-dW2));
121             S_inter(m_fine+1,:)=S_coarse(m,:);
122         end
123         h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discoundted ...
                payoff functions at fine level (l)
124         h_coarse=Payoff_put(r,T,S_coarse,K,Ml(l-1),Nl(l));  % discounted ...
                payoff functions at coarse level (l-1)
125         h_inter=Payoff_put(r,T,S_inter,K,Ml(l),Nl(l)); % discoundted ...
                payoff functions at coarse level (l-1) combined with ...
                interpolations
126     else
127         for m=2:Ml(l)+1
128             dW=sqrt(dt_l)*randn(1,Nl(l));
129             S_fine(m,:)=S_fine(m-1,:).*exp((r-q-0.5*sig^2)*dt_l+sig*dW);
130         end
131         h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discounted ...
                payoff functions at fine level (l)
132     end
133
134     if (l==1)
135        [Vec_rule_ML,V_ML,c_ML]=...
136             LSM_weighted(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine,lamda(l));
137     else
138        [Vec_rule_fine,V_fine,c_fine]=...
139            LSM_weighted(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine,lamda(l));
140       [Vec_rule_coarse,V_coarse,c_coarse]=...
141           LSM_weighted(S_inter,h_inter,Ml(l),Nl(l),J,Beta_fine,lamda(l));
142        Vec_rule_ML=Vec_rule_fine-Vec_rule_coarse;
143     end
144
145     Yl_rule(l)=sum(Vec_rule_ML)/Nl(l);
146     Varl_rule(l)=(sum(Vec_rule_ML.^2)/Nl(l)-Yl_rule(l)^2);
147
```

```
148     Sum_rule=Sum_rule+sqrt(Varl_rule(l)*T*Ml(l));
149 end
150
151 for l=1:L
152     Nl_rule(l)=int64(C*sqrt(Varl_rule(l)*T/Ml(l))/Sum_rule);
153 end
154
155
156 % MLMC optimal run
157     Nl=Nl_rule;
158 for l=1:L
159
160     S_fine=zeros(Ml(l)+1,Nl(l));   % stock path at level l
161     S_fine(1,:)=S0*ones(1,Nl(l));
162     Beta_fine=zeros(J,Ml(l));     % recording regression coefficients at ...
            fine level
163
164     for i=1:Ml(l)
165         ind=i*Ml(L)/Ml(l);    % global index corresponding to the global ...
                highest level
166         Beta_fine(:,i)=Beta(:,ind);   % copying golobal Betas to the ...
                local fine level (l)
167     end
168
169     if (l ~= 1)
170         S_coarse=zeros(Ml(l-1)+1,Nl(l));  %stock path at level (l-1)
171         S_inter=zeros(Ml(l)+1,Nl(l)); % stock path at level (l-1) ...
                combined with interpolations
172         S_coarse(1,:)=S0*ones(1,Nl(l));
173         S_inter(1,:)=S0*ones(1,Nl(l));
174         Beta_coarse=zeros(J,Ml(l-1));    % recording regression ...
                codfficients at coarse level
175
176
177         for i=1:Ml(l-1)
178              ind=i*Ml(L)/Ml(l-1);    % global index corresponding to the ...
                     global highest level
179             Beta_coarse(:,i)=Beta(:,ind);   % copying golobal Betas to ...
                    the local coarse level (l-1)
180         end
181      end
182
183     dt_l=T/Ml(l);  % time interval at level l (fine level)
184
185     % generating the main MC paths
186     if (l ~=1)
187         for m=2:Ml(l-1)+1
188             m_fine=2*m-2;
189             dW1=sqrt(dt_l)*randn(1,Nl(l));
190             S_fine(m_fine,:)=...
191                 S_fine(m_fine-1,:).*exp((r-q-0.5*sig^2)*dt_l+sig*dW1);
192             dW2=sqrt(dt_l)*randn(1,Nl(l));
193             S_fine(m_fine+1,:)=...
194                 S_fine(m_fine,:).*exp((r-q-0.5*sig^2)*dt_l+sig*dW2);
195             S_coarse(m,:)=...
196                 S_coarse(m-1,:).*exp((r-q-0.5*sig^2)*2*dt_l+sig*(dW1+dW2));
197
198             %Brownian interpolations
199             S_inter(m_fine,:)=...
```

```
                0.5*(S_coarse(m,:)+S_coarse(m-1,:)+...
                            (S_coarse(m-1,:).*sig).*(dW1-dW2));
            S_inter(m_fine+1,:)=S_coarse(m,:);
        end
        h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discoundted ...
            payoff functions at fine level (l)
        h_coarse=Payoff_put(r,T,S_coarse,K,Ml(l-1),Nl(l));  % discounted ...
            payoff functions at coarse level (l-1)
        h_inter=Payoff_put(r,T,S_inter,K,Ml(l),Nl(l)); % discoundted ...
            payoff functions at coarse level (l-1) combined with ...
            interpolations
    else
        for m=2:Ml(l)+1
            dW=sqrt(dt_l)*randn(1,Nl(l));
            S_fine(m,:)=S_fine(m-1,:).*exp((r-q-0.5*sig^2)*dt_l+sig*dW);
        end
        h_fine=Payoff_put(r,T,S_fine,K,Ml(l),Nl(l));   % discounted ...
            payoff functions at fine level (l)
    end

    if (l==1)
         [Vec_rule_ML,V_ML,c_ML]= ...
             LSM_weighted(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine,lamda(l));
    else
        [Vec_rule_fine,V_fine,c_fine]=...
            LSM_weighted(S_fine,h_fine,Ml(l),Nl(l),J,Beta_fine,lamda(l));
         [Vec_rule_coarse,V_coarse,c_coarse]=...
             LSM_weighted(S_inter,h_inter,Ml(l),Nl(l),J,Beta_fine,lamda(l));
        Vec_rule_ML=Vec_rule_fine-Vec_rule_coarse;
        cov_l=cov(Vec_rule_fine,Vec_rule_coarse,1);
        Cov_level(l)=cov_l(1,2);
        Corr_level(l)=cov_l(1,2)/sqrt(cov_l(1,1)*cov_l(2,2));
     end

    Yl_rule(l)=sum(Vec_rule_ML)/Nl(l);
    Varl_rule(l)=(sum(Vec_rule_ML.^2)/Nl(l)-Yl_rule(l)^2);

    Sum_rule=Sum_rule+sqrt(Varl_rule(l)*T*Ml(l));
end

V_rule_MLMC=sum(Yl_rule);

Var_rule_MLMC=0;

for l=1:L
    Var_rule_MLMC=Var_rule_MLMC+Varl_rule(l)/Nl(l);
end

disp('The value of option using FDM:');
disp(V_fdm);
disp('The value of option using standard MC');
disp(V_rule);
disp('The value of option using MLMC');
disp(V_rule_MLMC);
disp('Variance of standard MC')
disp(Var_rule);
disp('Variance of MLMC')
disp(Var_rule_MLMC);
disp('Number of MC paths at each level')
```

```
254 disp(Nl_rule);
255 disp('Variances at each level')
256 disp(Varl_rule);
257 disp('Correlations between fine and coarse levels');
258 disp(Corr_level);
```

# Bibliography


[1] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[2] M. Brennan and E. Schwartz. The valuation of American put options. *Journal of Finance*, 32(2):449–462, 1977.

[3] P.A. Forsyth and K.R. Vetzal. Quadratic convergence of a penalty method for valuing American options. *SIAM Journal on Scientific Computation*, 23:2096–2123, 2002.

[4] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

[5] F.A. Longstaff and E.S. Schwartz. Valuing American options by simulation: A simple leastsquares approach. *Review of Financial Studies*, 41(1):113–147, 2001.

[6] J.N. Tsitsiklis and B. Van Roy. Regression methods for pricing complex American-style options. *Neural Networks, IEEE Transactions on*, 12(4):694–703, 2001.

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

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

[9] L. Andersen and M. Broadie. Primal-dual simulation algorithm for pricing multidimensional American options. *Management Science*, 50(9):1222–1234, 2004.

[10] D. Belomestny and J. Schoenmakers. Multilevel dual approach for pricing American style derivatives. *DFG-Schwerpunktprogramm 1324*, Preprint 111.

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

[12] D. Duffie. *Dynamic Asset Pricing Theory, 3rd ed.* Princeton University Press, 2001.

[13] G. Maruyama. Continous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4(1):48–90, 1955.

[14] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[15] S. Pilska. *Introduction to Mathematical Finance*. Blackwell Publishers, 1997.

[16] C. Hermite. Sur un nouveau dveloppement en srie de fonctions. *Comptes Rendus de l'Academie des Sciences de Paris*, 58:93–100, 1864.

[17] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Wiley-Interscience, 1953.

[18] G. Pagès. Multi-step Richardson-Romberg extrapolation: remarks on variance control and complexity. *Monte Carlo Methods and Applications*, 13(1):37–70, 2007.

[19] S.E. Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 2. Springer Verlag, 2004.