MUTILEVEL MONTE CARLO ADAPTED TO BERMUDAN OPTIONS USING RANDOMIZED STOPPING RULE

Longyun Chen

Lady Margaret Hall

University of Oxford

Supervised by: Dr.Lajos Gergely Gyurkó



MSc Thesis

Trinity 2010

Acknowledgement

First, I would like to show my grateful appreciation to my supervisor Dr. Lajos Gergely Gyurko, for his patient and diligent work for my meeting every time and for his useful help and suggestions evertime I met problems. Again, I would also like to thank him for his well-organized job for us and all the help in academic and practical areas during the whole program. Secondly, I would like to appreciate Prof. Mike Giles for his lectures in Monte Carlo method and literatures which the main idea of my dissertation based on. Finally, I wish to thank to all our professors, lectures and all my classmates here for there support during my dissertation and entire MSc program this year.

Abstract

The Monte Carlo method is very widely used in option pricing in mathematical finance area. Main idea for the Monte Carlo method is to improve the accuracy of estimator by reducing variance with a fixed computational cost. Instead of European-style payoff, path-dependent option like Bermudan option in our paper is not easy to implement and reduce variance. Multi-level Monte Carlo method and Longstaff-Schwartz method are used to estimate the Bermudan option.

Key Words: Monte Carlo method, Multi-level Monte Carlo method, Longstaff-Schwartz Method, Computational Complexity(Cost), Variance Reduction, Randomized Stopping Rule.

Contents

Contents **1**

1 Introduction **3**

2 Multi-level Monte Carlo Approach **5**
2.1 Standard Monte Carlo Approach 5
2.2 Euler-Maruyama Method 5
2.3 Multi-level Monte Carlo Approach 8
2.4 Computational Cost and Complexity Theorem 9

3 Longstaff-Schwarz Method **13**
3.1 Early Exercise and Problem Formulation 13
3.2 Continuation Value and Stopping Rule 14
3.3 Longstaff-Schwartz Method 15

4 Valuing Bermudan Options based on Multi-level Monte Carlo and Randomized stopping rule **19**
4.1 Longstaff-Schwartz Method Coefficient 19
4.2 Backward and Forward Recursions 20
4.3 Numerical Pricing under Randomized Stopping Rule 21
4.4 Other Parameters Improvement 25

5 Conclusion **29**

6 Code and Appendix **31**
6.1 Generate the GBM 31

6.2 Weighted-Laguerre Explicit 31
6.3 Beta-Regression by Longstaff-Schwartz Method 32
6.4 Stopping ValueVec 33
6.5 Main Function of Multi-level Stopping Value 35

Chapter 1

Introduction

In the standard Monte Carlo approach for the option pricing under Black-Scholes model, the dynamic of share price is given by

$$dS_t = rS_t dt + \sigma S_t dW_t$$

and the computational cost under a fixed error ϵ for estimating the $E[f(S_T)]$ is given by $O(\epsilon^{-2})$. Mike Giles [2] shows that under the multi-level Monte Carlo approach, the computational cost for the fixed error ϵ is given by $O(\epsilon^{-2}(\log \epsilon - 2))$ and the the number of sample paths N_l for each level is reduced by half level by level while the multi-level is estimated unbiased since

$$E[\hat{P}_L] = E[\hat{P}_0] + \sum_{l=1}^{L} (\hat{P}_l - \hat{P}_{l-1}),$$

where $\hat{P}_l$ is the estimator of $\hat{f}(S_{T^l})$ at level l.

However, in the path-dependent case, for instance pricing the Bermudan option in this paper, the continuation value for the current share price X_t at time $t < T$ is not that easy to compute. In the paper, we apply Longstaff-Schwartz method to approximate the continuation value by a linear combination of some test functions we set. Longstaff and Schwartz[6] gave an effective set of test functions and we apply it in the paper to calculate the regression coefficient and the continuation level by backward recursion step by step.

In this paper, we extend the multi-level Monte Carlo framework to Bermudan option. Between the different levels, we introduce and apply a randomized stopping rule, which makes the lower level to be artificial and the top level to be the pure multi-level Monte Carlo approach, in order to increase correlation between adjacent levels. We aim to find an optimal set of smoothing parameters.

In Chapter 2, we introduce the standard Monte Carlo approach as well as the multi-level approach. Following Mike Gile's literature, we state and prove the theorem of computational complexity and the way to calculate the optimal N_l for each level. In Chapter 3, Longstaff-Schwartz method with the least-square method to calculate the continuation value is introduced. Also, we give the formulation problem and stopping rule for the Bermudan and Bermudan option. In Chapter 4, we introduce the randomized stopping rule to calculate the value and variance for the Bermudan option. Also, different implementations and results are given in Chapter 4 to compare the impact of different parameters.

Chapter 2

Multi-level Monte Carlo Approach

2.1 Standard Monte Carlo Approach

In mathematical finance area, Monte Carlo method is very popular in option pricing. The basic idea is that we sample a large number of random variables to get a set of possible outcomes and take their expection. The law of large numbers tells us that the estimate converges to the true value when we increase the number of samples, while the central limit theorem gives that the variance the variance/standard deviation is in a range we can compute.

For the European style payoff such as a European call, our aim is just to estimate the $E^{\mathbb{Q}}[e^{-rT}(S_T - K)^+]$ by sampling large number of paths for the share price from time 0 to T. However, for the path-dependent one, for instance Bermudan and Bermudan option which can be exercised before the expire time, we have to apply discretization to estimate the value step by step.

2.2 Euler-Maruyama Method

In standard Monte Carlo simulations which are used in computational finance, we are interested in the expected value of a quantity which is a functional of the solution to a SDE. For the simplest approximation for the scalar SDE,

$$dS_t = a(t, S_t)dt + b(t, S_t)dW_t, \quad 0 < t < T, \tag{2.1}$$

with S_0 given and the payoff function $f(S)$ with a uniform Lipschitz bound, i.e., there exists a constant c such that

$$|f(U) - f(V)| \leq c\|U - V\| \quad \forall U, V.$$

The Euler-Maruyama approximation applied to the SDE (2.1) is applied by:

$$\hat{S}_{n+1} = \hat{S}_n + a(t_n, \hat{S}_n)h + b(t_n, \hat{S}_n)\delta W_n,$$

where h is the timestep and the simplest estimate for $E[f(S_T)]$ is the mean of the payoff values $f(\hat{S}_{T/h})$ from N independent paths, i.e.,

$$\hat{Y} = N^{-1} \sum_{i=1}^{N} f(\hat{S}^i_{T/h}).$$

For a European style payoff $f(\hat{S}_T)$, the weak error is defined by

$$E[f(S_T)] - E[f(\hat{S}_{T/h})].$$

According to Mike Giles' lecture notes, if $p(S)$ is the probability density function(p.d.f.) for S_T and $\hat{p}(S)$ the p.d.f. for $\hat{S}_{T/h}$, then under appropriate conditions on $a(t, S_t)$ and $b(t, S_t)$ using Euler-Maruyama Method,

$$p(S) - \hat{p}(S) = O(h).$$

Therefore, order of weak error for European style payoff is

$$E[f(S_T)] - E[f(\hat{S}_{T/h})] = O(h).$$

We denote $V := E[f]$ as the true option value, $\hat{V} = E[\hat{f}]$ the discrete approximation, and $\hat{Y} = N^{-1}\sum_{i=1}^{N} f(\hat{f}^i)$ the Monte Carlo estimate, the Mean Square Error is given by

$$\begin{aligned} E\left[(\hat{Y}-V)^2\right] &= E\left[\left(\hat{Y}-E[\hat{f}]+E[\hat{f}]-E[f]\right)^2\right] \\ &= E\left[\left(\hat{Y}-E[\hat{f}]\right)^2\right]+\left(E[\hat{f}]-E[f]\right)^2 \\ &= N^{-1}\mathbb{V}[\hat{f}]+\left(E[\hat{f}]-E[f]\right)^2 \end{aligned}$$

where the first term is the variance of estimator and the second term is the square of bias due to weak error.

For the Euler-Maruyama Method with M time steps and a fixed computational cost proportional to $C = NM$, the Mean Square Error is approximately given by

$$\text{MSE} \approx c_1 N^{-1} + c_2 M^{-2} = c_1 N^{-1} + c_2 C^{-2} N^2.$$

where $c_1, c_2 > 0$ are constants.
To minimise the Mean Square Error, we get

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

and therefore

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

where we can see that the Monte Carlo term is twice the bias term.

2.3 Multi-level Monte Carlo Approach

Consider Monte Carlo path simulations with different timesteps $h_l = M^{-l}T$, $l = 0, 1, ..., L$. For a given Brownian path W_t, let $P := f(S_T)$, where f is the payoff function, and denote $\hat{S}_{l,M^l}$ and $\hat{P}_l$ the approximations to S_T and P using the numerical discretisation with time timestep h_l.

According to the linearity of expectation, we have

$$E[f(\hat{S}_T)] = E[\hat{P}_L] = E[\hat{P}_0] + \sum_{l=1}^{L} E[\hat{P}_l - \hat{P}_{l-1}].$$

The multi-level monte carlo approach estimates the expectations on the right hand side independently for each level in a way which minimises the computational cost at a fixed variance or reduces the variance at a fixed computational cost.

Let $\hat{Y}_0$ be the estimator of $E[\hat{P}_0]$ with N_0 samples, and $\hat{Y}_l$, $l \in [1, 2, ...L]$ be the estimator of $E[\hat{P}_l - \hat{P}_{l-1}]$ with N_l paths. A simple way to get the estimator is given by

$$\hat{Y}_l = \frac{1}{N_l} \sum_{i=1}^{N_l} (\hat{P}_l^i - \hat{P}_{l-1}^i).$$

Here we can see that $\hat{P}_l^i - \hat{P}_{l-1}^i$ comes from two different discrete approximations but the same Brownian path. It is easily implemented by constructing the Brownian increments first to simulate the discrete path leading to the value of $\hat{P}_l^i$, and then summing them together to give the discrete Brownian increments for the value of $\hat{P}_{l-1}^i$.

Now we are to consider the variance. Variance of the simple estimator is given by $V[\hat{Y}_l] = {N_l}^{-1} V_l$ where V_l is the variance of a single path. The same inverse dependence on N_l may apply in the case of a more exact estimator using stratified sampling or control variate to reduce the variance, where the variance of estimator $\hat{Y} = \sum_{l=0}^{L}$ is given by

$$V[\hat{Y}] = \sum_{l=0}^{L} {N_l}^{-1} V_l.$$

The computational cost is proportional to $\sum_{l=0}^{L} N_l {h_l}^{-1}$. So for a fixed computational the variance is minimised by choosing N_l to be proportional to $\sqrt{V_l h_l}$.

2.4 Computational Cost and Complexity Theorem

For the standard Monte Carlo approach, to make the Mean Square Error $O(\epsilon^2)$, we need that $N = O(\epsilon^{-2})$ and $h = O(\epsilon)$, which makes the computational cost to be $O(\epsilon^{-3})$. In the multi-level Monte Carlo approach, by choosing appropriate M, we can reduce the computational complexity to $O(\epsilon^{-2}(\log \epsilon)^{-2})$. To analyse the computational complexity for each level l and find the optimal number of paths N_l for each leve, Mike Giles[3] gives the following theorem and proof.

Theorem 2.1 *Let P be a functional of the solution to the following SDE,*

$$dS_t = a(t, S_t)dt + b(t, S_t)dW_t, \quad 0 < t < T$$

and $\hat{P}_l$ be the corresponding approximation by the discretisation with timestep $h_l = M^{-l}T$.
If there exists independent $\hat{Y}_l$ based on N_l sample paths, and constants $\alpha \geq \frac{1}{2}$, $\beta, c_1, c_2, c_3 > 0$ satisfying
1): $E[\hat{P}_l - P] \leq c_1 h_l^{\alpha}$,

2): $E[\hat{Y}_l] = \begin{cases} E[\hat{P}_0] & \text{if } l = 0 \\ E[\hat{P}_l - \hat{P}_{l-1}] & \text{if } l \geq 1 \end{cases}$,

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

4): The computational cost of $\hat{Y}_l$, denoted by C_l, is bounded by $C_l \leq c_3 N_l h_l^{-1}$,

then there exists a constant $c_4 > 0$ s.t. $\forall \epsilon < e^{-1}$, $\exists$ L and N_l for which the multi-level

Monte Carlo estimator $\hat{Y} = \sum_{l=0}^{L} \hat{Y}_l$ *has a MSE satisfying*

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

and a computational cost C *with the following bound*

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

Proof. Let $\lceil X \rceil$ be the integer part of X. First we choose L by satisfying

$$L = \left\lceil \frac{\log\left(\sqrt{2}c_1 T^{\alpha}\epsilon^{-1}\right)}{\alpha \log M} \right\rceil,$$

which implies $M^{-\alpha}\epsilon < \sqrt{2}c_1 h_l^{\alpha} \leq \epsilon$. According to 1) and 2), we have

$$(E[\hat{Y}] - E[P]) \leq \frac{1}{2}\epsilon^2.$$

Now we consider the different β. For $\beta = 1$, we choose

$$N_l = \left\lceil 2\epsilon^{-2}(L+1)c_2 h_l \right\rceil$$

s.t.

$$\mathbb{V}[\hat{Y}] = \sum_{l=0}^{L} N_l^{-1}\mathbb{V}_l \leq \frac{1}{2}\epsilon^2.$$

Therefore, the MSE is bounded by

$$MSE = \mathbb{V}[\hat{Y}] + (E[\hat{Y}] - E[P])^2 \leq \frac{1}{2}\epsilon^2 + \frac{1}{2}\epsilon^2 = \epsilon^2.$$

To bound the computational complexity C, first we give an upper bound on L s.t.

$$L \leq \frac{\log \epsilon^{-1}}{\alpha \log M} + \frac{\log\left(2c_1 T^{\alpha}\right)}{\alpha \log M} + 1.$$

Hence, $\forall \epsilon < e^{-1}$, it follows that

$$L + 1 \leq c_5 \log \epsilon^{-1},$$

where

$$c_5 = \frac{1}{\alpha \log M} + \max\left(0, \frac{\log(2c_1 T^\alpha)}{\alpha \log M}\right) + 2.$$

The upper bound of N_l is given by

$$N_l \leq 2\epsilon^{-2}(L+1)c_2 h_l + 1.$$

Therefore, the computational cost C is bounded by

$$\begin{aligned} C &\leq c_3 \sum_{l=0}^{L} N_l h_l^{-1} \\ &\leq c_3(2\epsilon^{-2}(L+1)^2 c_2 + \sum_{l=0}^{L} h_l^{-1}) \\ &\leq c_3(2\epsilon^{-2}(L+1)^2 c_2 + \frac{M^2}{M-1}(\frac{\epsilon}{2c_1})^{-1/\alpha}), \end{aligned}$$

since

$$\sum_{l=0}^{L} h_l^{-1} < \frac{M}{M-1} h_L^{-1} < \frac{M^2}{M-1}(\frac{\epsilon}{2c_1})^{-1/\alpha}.$$

Note that $\epsilon^{-1/\alpha} < \epsilon^{-2}(\log \epsilon)^2$, and apply the upper bound for $L+1$, we get

$$C \leq c_4 \epsilon^{-2}(\log \epsilon)^2,$$

where

$$c_4 = 2c_3 c_5 + c_3 \frac{M^2}{M-1}(2c_1)^{1/\alpha}.$$

For $\beta > 1$, we choose

$$N_l = \left\lceil 2\epsilon^{-2} c_2 (1 - M^{(\beta-1)/2})^{-1} h_l^{(\beta+1)/2} \right\rceil,$$

and it is easy to find that the MSE is still less than ϵ^2. The computational const bound is a decreasing geometric sequence, bounded by the result stated in the theorem.

For $\beta < 1$, we choose the appropriate N_l by

$$N_l = \left\lceil 2\epsilon^{-2} c_2 (1 - M^{(1-\beta)/2})^{-1} h_L^{-(1-\beta)/2} h_l^{(1+\beta)/2} \right\rceil.$$

Similarly, the MSE is also less than ϵ^2. However, the corresponding computational cost is an increasing geometric sequence, bounded by the result stated before. ■

In the case of standard Monte Carlo approach, applying Euler-Maruyama Method with Lipschitz payoff, the weak convergence is given by $O(h)$ while the strong convergence is given by $O(\sqrt{h})$ under certain conditions on $a(t, S_t)$ and $b(t, S_t)$. In the multi-level Monte Carlo approach with Euler discretisation, for a given timestep h_l, a fraction of the paths with size $O(\sqrt{h_l})$ gives a final value of $\hat{S}_{l,M^l}$ which is $O(\sqrt{h_l})$ from a discontinuity. With the Euler discretisation, the fraction of paths gives $\hat{P}_l - \hat{P}_{l-1}$ being order $O(1)$ with a probability $O(1)$, and therefore $\mathbb{V}_l = O(\sqrt{h_l})$ and $\beta = \frac{1}{2}$. Since the weak convergence is still $O(h_l)$, we have $\alpha = 1$ and the overall computational cost is $O(\epsilon^{-\frac{5}{2}})$, reducing the $O(\epsilon^{-3})$ computational cost of standard Monte Carlo approach with Euler-Maruyama Method.

Furthermore, we can reduce the computational complexity by replacing Milstein method instead of Euler-Maruyama method. And it might be also efficient by modifying the zero-mean control variate estimator to reduce the computational cost to obtain the same order of convergence for vector SDEs.

According to Mike Giles's literature, we know that the optimal M is set to be $M = 2$. Later on, we use all 2^L and 2^l for the Multi-level Monte Carlo approach to reduce the computational cost most efficiently.

Chapter 3

Longstaff-Schwarz Method

3.1 Early Exercise and Problem Formulation

Comparing to the European options that can only be exercised at the expiry time T, American options can be exercised at any time up to T while Bermudan options can be exercised at a finite sequence of times $t_i \leq T$. Our aim for the American option pricing problems can be formulated to a process $U(t)$, $0 \leq t \leq T$, representing the discounted payoff from the exercise at time t, and a class of stopping time $\mathcal{T}$. Therefore, our aim is just to find an optimal expected discounted payoff, i.e.,

$$\sup_{\tau \in \mathcal{T}} E[U(\tau)].$$

With an instantaneous short rate $r(t)$, where $0 \leq t \leq T$, the problem formulation becomes to

$$\sup_{\tau \in \mathcal{T}} E\left[e^{-\int_0^\tau r(u)du} \tilde{h}(X(\tau))\right],$$

where $\tilde{h}(X)$ is the payoff function of Bermudan put option.

Let $\tilde{h}_i$ be the payoff function as defined above for exercising at t_i, depending on only i and $\tilde{V}_m(x)$ be the value of the option at t_i given $X_i = x$, conditioning on the option has not been exercised before t_i. The values is determined by the following:

$$\tilde{V}_m(x) = \tilde{h}_m(x), \tag{3.1}$$

$$\tilde{V}_{i-1}(x) = \max\{\tilde{h}_{i-1}(x), E[B_{i-1,i}(X_i)\tilde{V}_i(X_i)|X_{i-1} = x]\}, \tag{3.2}$$

$$i = 1, 2, ..., m, \tag{3.3}$$

where $B_{i-1,i}(X_i) = \exp\left(-\int_{t_{i-1}}^{t_i} r(u)du\right)$ is the discount factor from time t_{i-1} to t_i.

3.2 Continuation Value and Stopping Rule

Now we define

$$h_i(x) := B_{0,i}(x)\tilde{h}_i(x), \quad i = 1, 2, ..., m,$$

and

$$V_i(x) := B_{0,1}(x)\tilde{V}_i(x), \quad i = 0, 1, ..., m.$$

Since $B_{0,0} = 1$, which implies $V_0(x) = \tilde{V}_0(x)$, we have

$$\begin{aligned}
V_m(x) &= h_m(x), \\
V_{i-1}(x) &= B_{0,i-1}(x)\tilde{V}_{i-1}(x) \\
&= B_{0,i-1}(x)\max\{\tilde{h}_{i-1}(x), E[B_{i-1,i}(X_i)\tilde{V}_i(X_i)|X_{i-1} = x]\} \\
&= \max\{h_{i-1}(x), E[B_{0,i-1}(x)B_{i-1,i}(X_i)\tilde{V}_i(X_i)|X_{i-1} = x]\} \\
&= \max\{h_{i-1}(x), E[V_i(X_i)|X_{i-1} = x]\}.
\end{aligned}$$

The continuation value of a Bermudan option with finite number of exercise dates $t_i, \quad i = 0, 1, ...m$ is the value of holding the option rather exercising. The continuation value at t_i is given by

$$C_i(x) = E[V_{i+1}(X_{i+1})|X_i = x], \quad i = 0, 1, ..., m-1.$$

At time $t = 0$, the option value is $C_0(X_0)$, which is the continuation value at time 0,

as we assume that the Bermudan option won't be exercised at time 0.

The value of Bermudan option V_{i+1} determines the continuation value C_i, whereas we have conversely

$$V_i(x) = \max\{h_i(x), C_i(x)\}, \quad i = 1, 2, ..., m.$$

Therefore, we have the following stopping rule,

$$\tau = \min\{i \in \{1, 2, ..., m\} : h_i(X_i) \geq C_i(X_i)\},$$

where $h_m(X) = C_m(X)$ by convention at the last time step.

For the approximation described in the following Longstaff-Schwartz Method, the approximation $\hat{C}_i$ to the continuation values determine the stopping rule through

$$\hat{\tau} = \min\{i \in \{1, 2, ..., m\} : h_i(X_i) \geq \hat{C}_i(X_i)\}.$$

3.3 Longstaff-Schwartz Method

At the maturity time $t_m = T$, the value of option is given by

$$V_m(X_m) = h_m(X_m).$$

And at time t_{m-1}, the continuation value is

$$C_{m-1}(x) = E\left[V_m(X_m)|X_{m-1} = x\right],$$

where we use a set of R basis functions $\varphi_r(x)$ to approximate in Longstaff-Schwartz Method, i.e.,

$$\hat{C}_{m-1}(x) = \sum_{r=1}^{R} \beta_r \varphi_r(x).$$

In this paper, we use the Laguerre polynomials as the R basis function:

$$\varphi_0(x) = \exp(-x/2), \tag{3.4}$$
$$\varphi_1(x) = \exp(-x/2)(1-x), \tag{3.5}$$
$$\varphi_2(x) = \exp(-x/2)(1-2x+x^2/2), \tag{3.6}$$
$$\vdots \tag{3.7}$$
$$\varphi_r(x) = \exp(-x/2)\frac{e^x}{r!}\frac{d^r}{dx^r}(x^r e^{-x}). \tag{3.8}$$

Generally, we set $0 \leq r \leq R = 6$ to calculate the R basis function.

Now we are to introduce how to get the coefficients β_r by applying the least-squares minimisation, which is a vital part of Longstaff-Schwartz Method. Basically, our aim is to minimise

$$\begin{aligned} & E\left\{\left(E[V_m(X_m)|X_{m-1}=x] - \hat{C}_{m-1}(X_{m-1})\right)^2\right\} \\ = & E\left\{\left(E[V_m(X_m)|X_{m-1}=x] - \sum_{r=1}^{R}\beta_r\varphi_r(x)\right)^2\right\} \end{aligned} \tag{3.9}$$

Differentiate equation (3.9) w.r.t. β_r, we get

$$E\left\{\left(E[V_m(X_m)|X_{m-1}=x] - \hat{C}_{m-1}(X_{m-1})\right)\varphi_r(X_{m-1})\right\} = 0.$$

Therefore,

$$E[V_m(X_m)\varphi_r(X_{m-1})] = E[\hat{C}_{m-1}(X_{m-1})\varphi_r(X_{m-1})]$$
$$= \sum_s E[\varphi_r(X_{m-1})\varphi_s(X_{m-1})]\beta_s. \tag{3.10}$$

Equivalently, we have for $i = 0, 1, ..., m-1$, the continuation value C_i is given by

$$C_i(x) = {\beta_i}^T \varphi(x),$$

where $\beta_i^T = (\beta_{i1}, \beta_{i2}(x), ..., \beta_{iR}(x))$ and $\varphi(x) = (\varphi_1(x), \varphi_2(x), ..., \varphi_R(x))^T$.

According to equation (3.10), we get

$$\beta_i = (E[\varphi(X_i)\varphi(X_i)^T])^{-1} E[\varphi(X_i)V_{i+1}(X_{i+1})] := B_{\varphi\varphi}^{-1} B_{\varphi V}.$$

Here, as defined, we can see $B_{\varphi\varphi}$ is a $R \times R$ matrix and $B_{\varphi V}$ is a $R \times 1$ row vector.

To estimate the coefficients β_{ir}, we use N independent sample paths with the observations of the pair $(X_{ij}, V_{i+1}(X_{i+1,j}))$, where $j = 1, 2, ..., N$. Therefore, the estimator $\hat{\beta}_i$ is given by

$$\hat{\beta}_i = \hat{B}_{\varphi\varphi}^{-1} \hat{B}_{\varphi V},$$

where $\hat{B}_{\varphi\varphi}$ is a $R \times R$ matrix with the ab-th entry

$$(\hat{B}_{\varphi\varphi})_{ab} = \frac{1}{N} \sum_{j=1}^{N} \varphi_a(X_{ij})\varphi_b(X_{ij})$$

and $\hat{B}_{\varphi V}$ is a $R \times 1$ row vector with the b-th entry

$$(\hat{B}_{\varphi V})_{1b} = \frac{1}{N} \sum_{j=1}^{N} \varphi_b(X_{ij}) V_{i+1}(X_{i+1,j}).$$

In practice, V_{i+1} has to be inducted backwards by $\hat{V}_{i+1}$ and the approximation of continuation value $C_i(x)$ is denoted by $\hat{C}_i(X)$.

Algorithm 3.1 *Simulate N independent sample paths $\{X_{1j}, X_{2j}, ..., X_{mj}\}$ for the underlying assets, $j = 1, 2, ..., N$.*

At terminal step m, $\hat{V}_{mj} = h_m(X_{mj})$, where $h(\cdot)$ is the payoff function, $j = 1, 2, ..., N$.

Apply the backwards induction for $i = m-1, ..., 1$.
First, use the estimated values $\hat{V}_{i+1,j}$ to get the regression coefficient β_i via $\beta_i = \hat{B}_{\varphi}^{-1}\hat{B}_{\varphi V}$;
Second, set $\hat{V}_{ij} = \max\Big\{h_i(X_{ij}), \hat{C}_i(X_{ij}))\Big\}$, $j = 1, 2, ..., N$ where $\hat{C}_i$ is the estimator of continuation value.

For the last step of backwards induction, there is no β, just set $\hat{V}_0 = \frac{1}{N}\sum_{j=1}^{N}\hat{V}_{1j}$.

Algorithm 3.2 *Simulate N independent sample paths $\{X_{1j}, X_{2j}, ..., X_{mj}\}$ for the underlying asset, $j = 1, 2, ..., N$.*

Use forwards induction.
for $i = 1, 2, ..., m-1$
Apply $\hat{C}_{ij}(X_{ij}) = \beta_i^T\varphi(X_{ij})$, $h_{ij}(X_{ij}) = payoff(X_{ij})$.
if $h_{ij}(X_{ij}) \geq \hat{C}_{ij}(X_{ij})$
return $\hat{V}_{mj} = h_{ij}(X_{ij})$

At time step m, $\hat{V}_{mj} = h_{mj}(X_{mj})$, $\hat{V}_m = \frac{1}{N}\hat{V}_{mj}$.

For the convergence result of the Least-Square method, we refer to Clement, Lamberton and Protter[9].

Chapter 4

Valuing Bermudan Options based on Multi-level Monte Carlo and Randomized stopping rule

4.1 Longstaff-Schwartz Method Coefficient

As defined in Chapter 2, we introduced the way to approximate the estimator of continuation value $C_i(X)$. In the implementation, we use a fixed β_i which is estimated based on a large number of sample paths and apply it to the further approximation for the continuation value C_i.

Since our basis functions are set to be the weighted-Laguerre Polynomials with a weight of $\exp(-x/2)$, we would not like to set the spot and strike price to be high. For $R = 5, S_0 = 11, K = 15, L = 5, \sigma = 0.4, r = 0.05$, we get the beta for an example by simulating $N = 10^5$ paths. See Appendix.

We set β^T below for a convenient view since β_i is a 5×31 matrix for the levels $L = 5$.
As we mentioned in the previous two chapters, we use the Longstaff-Schwartz Method to approximate the continuation value of Bermudan options and apply Multi-level Monte Carlo approach to do the variance reduction, with choosing the appropriate

sample paths N_l at each level l.

4.2 Backward and Forward Recursions

The Longstaff-Schwartz Method gives the way to estimate the value of Bermudan option by backwards recursion with the algorithm introduced in the last chapter. Now we implement the Longstaff-Schwartz Method for time steps 2^5 to 2^2 with $N = 10^3, 10^4, 10^5$ to see if the stopping rule we set before for 2^5 time steps works for the lower time steps 2^l where $l < 5$. For the 2^5 time steps, we apply the β getting from the method we introduced before, and for the time steps 2^l where $l < 5$, we use the corresponding β_i at each time step i. This is to some extent artificial method, but we are just to show how the Longstaff-Schwartz works for the less time steps.

Table 4.1: **Backward Recursion by Longstaff-Schwartz Method**

	$N = 10^5$	$N = 10^4$	$N = 10^3$
2^5 time steps	4.3171	4.3324	4.4367
2^4 time steps	4.3150	4.2968	4.2061
2^3 time steps	4.2940	4.2884	4.2846
2^2 time steps	4.2423	4.2469	4.1686

From the table above, we can find that the stopping rules for 2^5 works perfectly for the case of 2^l time steps where $l < 5$.

Remark 4.1 In the table above, we use also the same parameters $R = 5, T = 1, r = 0,05, \sigma = 0.4, S_0 = 11, K = 15$ to get the Bermudan put price using the β matrix saved before, seen in the Appendix. Such β can be only used from backward recursion to forward recursion with same number of test functions and corresponding parameters. If the spot and strike prices change, or others, β must be calculated and saved again to apply for the forward recursion.

4.3 Numerical Pricing under Randomized Stopping Rule

As introduced in the previous section, we set up the stopping rule such that the Bermudan option is exercised at the first exercise date which the payoff is higher than its continuation value, i.e.,

$$\tau = \min\{i \in \{1, 2, ..., m\} : h_i(X_i) \geq \hat{C}_i(X_i)\}.$$

However, in order to increase the correlation between levels l and level $l-1$, we outline a smoothing technique based on randomized stopping rule proposed by Mike Giles first. The main idea is that the Bermudan option is not surely exercised even if the payoff is greater than the continuation value, and vice versa. Instead, we introduce a probability of exercising or holding the Bermudan option based on the difference between the payoff and continuation value. In the conventional stopping rule, we exercise the Bermudan option as if the payoff is greater or equal to the continuation value and continue to the next time step if the payoff is less than the continuation value. In the randomized stopping rule, we randomize the exercise desicion in a way such that the probability of exercising or holding the option depends on the difference between the exercising payoff and continuation value.

Equivalently, the greater difference becomes, the higher probability we give for exercising the option under randonmized stopping rule. For the payoff $h_i(X_i)$ and continuation value $\hat{C}_(X_i)$, the probability of exercising the Bermudan option is given by

$$\mathbb{P}(\lambda, h_i(X_i), \hat{C}_i(X_i)) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\lambda(h_i(X_i)-\hat{C}_i(X_i))} e^{-x^2/2} dx = \Phi\Big(\lambda(h_i(X_i) - \hat{C}_i(X_i))\Big).$$

where $\Phi(\cdot)$ is the cumulative density function of a normal distributed random variable Z with mean 0 and variance 1.

With a positive parameter λ chosen, we have that the probability of exercising the Bermudan is greater than one half when $h_i(X_i) \geq \hat{C}_i(X_i)$ while it is less than when

$h_i(X_i) \leq \hat{C}_i(X_i)$. For a large $\lambda \to \infty$ chosen, the probability is close to 1 when $h_i(X_i) \geq \hat{C}_i(X_i)$ and close to 0 for $h_i(X_i) \leq \hat{C}_i(X_i)$ since $\Phi(\infty) = 1$ and $\Phi(-\infty) = 0$, which implies the stopping rule we introduced before.

Therefore, the value of a Bermudan option under the randomized stopping rule is given by

$$E\left[\sum_{i=1}^{m} C_i(X_i)\mathbb{P}(h_i(X_i), \hat{C}_i(X_i)) \prod_{j=1}^{m-1} \left[1 - \mathbb{P}(h_i(X_i), \hat{C}_i(X_i))\right]\right]. \tag{4.1}$$

First we set $\lambda = 10^6$ to show how the variance is reduced when using multi-level Monte Carlo approach. As mentioned in chapter 1, we have the expectation of estimator is given by

$$E[\hat{P}_L] = E[\hat{P}_0] + \sum_{l=1}^{L} E[\hat{P}_l - \hat{P}_{l-1}].$$

Since at the lower levels, we choose the smoothing parameters λ to be close to 0, $E[\hat{P}_l]$ for $l < L$ is purely by technique and does not correspond to the price of any derivative. However, λ at level L is chosed to be great ($>> 1$), which implies that $E[\hat{P}_L] at \text{level } L$ is by the pure multi-level Monte Carlo approach and we note that the smoothing technique above is still an unbiased estimate of the theoretical value resulted by the forward Longstaff-Schwartz method.

Meanwhile, in practice, it is no strict constraint for us to start from level 0. Given a starting level a where $0 \leq a \leq L$, the expectation is of the form

$$E[\hat{P}_L] = E[\hat{P}_a] + \sum_{l=a+1}^{L} E[\hat{P}_l - \hat{P}_{l-1}],$$

where $a = 5$ gives the standard Monte Carlo estimator without using multi-level approach.

Due to the memory constraint of Matlab, we set the total level $L = 5$, with $R =$

$5, S_0 = 11, K = 15, r = 0.05, \sigma = 0.4, T = 1$. Using the β matrix saved before, we compare the variance between the standard Monte Carlo approach. Since the simple equal-weighted sample paths N_l can not reduce the variance, what we need to do first is to find the optimal sample paths N_l for each level $l \leq L$. As introduced in the previous chapter, the number of sample paths N_l is proportional to $\sqrt{V_l h_l}$, let c_1 be the proportional constant, C the fixed computational cost, and $h_l = 2^{-l}$ in the whole set up of our simulation, hence the total computational cost is given by

$$C = c_1 \left(\sum_{i=a}^{L} \sqrt{V_l 2^{-i}} \cdot 2^l \right).$$

Therefore, the constant c_1 and corresponding number of paths N_l is given by

$$c_1 = \frac{C}{\sum_{i=a}^{L} \sqrt{V_l 2^i}},$$

and the number of sample paths under the fixed computational cost C is thus

$$N_l = \frac{C\sqrt{V_l 2^{-l}}}{\sum_{i=a}^{L} \sqrt{V_l 2^i}}.$$

The following table compares the value and variance of estimator using standard Monte Carlo approach and multi-level Monte Carlo approach with the starting level $a = 0$ and a fixed computational cost $C = 2^5 \times 10^5$. According to the formula given above, the optimal number of paths N_l is also shown in the table, as well as the variance of each level.

Here, we use the same parameters as what used to estimate beta in the previous section, i.e., $R = 5, S_0 = 11, K = 15, T = 1, r = 0.05, \sigma = 0.4$, and we use the regression coefficient β saved before, as can be seen in the Appendix.

Table 4.2: **Multi-level Estimator with large parameter** $\lambda = 10000$

	level value	level variance	N_l	estimator variance($1.0e-4$)
$l = 0$	4.0402	9.4152	541042	0.1740
$l = 1$	0.0929	2.5252	198132	0.1275
$l = 2$	0.0668	1.7200	115625	0.1488
$l = 3$	0.0588	1.0352	63427	0.1632
$l = 4$	0.0517	0.7187	37371	0.1923
$l = 5$	0.0312	0.4853	21714	0.2235
Overall	**4.3416**	N/A	N/A	**1.0292**
Standard MC	**4.3125**	5.4357	10^5	**0.5436**

From the table above we can see, the variance using $\lambda \to \infty$ for all the levels is even greater than the variance using standard Monte Carlo approach. Under such circumstance, we have to adjust the parameter λ to reduce the variance.

Now we set the λ's at levels 1 to 5 to be $\lambda = [0.01, 3, 10, 30, 100]$ respectively, where $100 >> 1$ as well at the last step, and using the same N_l as before and change the starting level to be $a = 1$, then we get the following table where we can find that the variance has been reduced to half comparing with using all large parameters as before.

Remark 4.2 In the later set up for the smoothing parameters λ, we need to ensure that the last step must not have any artificial impact in calculating. For the example with a higher spot here, $\lambda = 100$ might be ok, but later we set all $\lambda = 10000$ for the last step while $\lambda \to 0$ for the starting level.

Table 4.3: **Multi-level Estimator with smoothing parameter** λ

	λ_l	level value	level variance	N_l	estimator variance($1.0e-4$)
$l = 1$	0.01	3.0257	3.8124	379346	0.1005
$l = 2$	3	1.1417	0.9252	132143	0.0700
$l = 3$	10	0.0711	0.4389	64355	0.0682
$l = 4$	30	0.0558	0.3114	38331	0.0812
$l = 5$	100	0.0316	0.2548	24519	0.1039
Overall	N/A	**4.3260**	N/A	N/A	**0.4239**
Standard MC	∞	**4.3294**	5.2516	10^5	**0.5252**

Here we can find that with chosen set of the smoothing parameter and starting level 1, our variance is reduced to half as that compared to previous test case and succeed in reducing the variance about a quarter comparing to the standard Monte Carlo approach used under the same computational cost.

4.4 Other Parameters Improvement

However, we consider that it is still not that good for the variance reduction for the example we introduced in the previous section. Our aim now is to further improve the variance reduction under the Longstaff-Schwartz method, Multi-level Monte Carlo approach combining with the randomized stopping rule we used in the last examples. All the parameters which influences the result should be considered to change.

From the last two tables, we found out that the variance by applying the multi-level Monte Carlo approach is even larger than the variance of standard Monte Carlo approach both with the Longstaff-Schwartz method to estimate the continuation value at each time step. As we see from the last examples, the spot price $S_0 = 11$ and $S = 15$, which means the Bermudan put is well out-of-money at the starting time $T = 0$ and might give a large variance when computing in the first two steps according to its corresponding relation. First we set an at the money put with $S_0 = K = 10$, to see if the variance has been reduced from standard Monte Carlo approach to multi-level one. The following table (4.4) compares the value and variance using standard Monte Carlo approach and multi-level one with $S_0 = K = 10, R = 5, T = 1, r = 0.05, \sigma = 0.4$ and the fixed computational cost $C = 2^5 \times 10^5$.

Notice that we have to calculate the coefficients matrix β with a large number of N again since the spot and strike prices have been changed and the β matrix saved before is no longer effective.

Table 4.4: **Estimator of at-the-money Bermudan put with $\lambda = 10000$**

	level value	level variance	N_l	estimator variance($1.0e-5$)
$l = 0$	1.3109	1.4093	615397	0.4585
$l = 1$	0.0332	0.3591	155242	0.2313
$l = 2$	0.0271	0.2350	88802	0.2646
$l = 3$	0.0135	0.2819	68778	0.4099
$l = 4$	0.0079	0.1745	38257	0.4560
$l = 5$	0.0043	0.1333	23643	0.5636
Overall	**1.3968**	N/A	N/A	**2.3839**
Standard MC	**1.4093**	5.4357	10^5	**2.4658**

From the table we can see that when implementing the at-the-money put, the variance between standard Monte Carlo approach and multi-level one stand for nearly no difference. The multi-level approach still does not improve the accuracy that obviously.

Now we use all the same market parameters but change the λ to be smooth, to check if it works well for this case.

Table 4.5: **Estimator of at-the-money Bermudan put with smoothing λ**

	λ	level value	level variance	N_l	estimator variance($1.0e-5$)
$l = 0$	0.001	0.6596	0.7111	410395	0.1733
$l = 1$	1	0.6071	0.4626	234055	0.1976
$l = 2$	10	0.1025	0.1369	90046	0.1521
$l = 3$	30	0.0203	0.1274	61428	0.2075
$l = 4$	100	0.0098	0.0930	37095	0.2506
$l = 5$	10000	0.0045	0.1013	27386	0.3700
Overall	N/A	**1.4038**	N/A	N/A	**1.3510**
Standard MC	∞	**1.4093**	5.4357	10^5	**2.4658**

As expected, the parametric method of smoothing λ does improve the accuracy well. The variance has been reduced by around 40% of that by using standard Monte Carlo approach. Meanwhile, according to the result of two paris of spot and strike prices, variances of using smoothing λ have all been reduced to half or less comparing to the corresponding $\lambda \to \infty$ used. The result means that using the smoothing parameters λ is extremely important in our implementation. Therefore, in the parameter

sensitivity testing later, we all use smoothing λ with the last step very large, instead applying all the $\lambda \to \infty$.

Now we analysis the sensitivity of the test functions, both the number of test functions for Longstaff-Schwartz method itself and the correlation between the initial spot price and number of test functions used for estimating the regression coefficient β's. Recall that the R-basis function is given by a weighted-polynomial with a weight factor of $\exp(-x/2)$. With this exponential weight, the β matrix which is calculated based on the R-basis function might be very sensitive to the current share price in the simulating paths especially when the initial spot price is going large. Under this hypothesis, we set up an in-the-money Bermudan put with spot price $S_0 = 5$ and strike $K = 4$ to see if result of our experiment improves with a potential accurate impact on β. As stated before, we apply all smoothing λ in the following examples.

Table 4.6: **Estimator of Bermudan put with low spot and strike price**

	λ	level value	level variance	N_l	estimator variance($1.0e-6$)
$l = 0$	0.001	0.1272	0.0602	424181	0.1420
$l = 1$	1	0.0718	0.0246	191469	0.1282
$l = 2$	10	0.0576	0.0191	119304	0.1598
$l = 3$	30	0.0086	0.0068	50520	0.1353
$l = 4$	100	0.0035	0.0078	38087	0.2041
$l = 5$	10000	0.0013	0.0085	28191	0.3021
Overall	N/A	**0.2700**	N/A	N/A	**1.0715**
Standard MC	∞	**0.2713**	2.1770	10^5	**2.1770**

From the result, we can find that under a lower strike and spot price, the multi-level Monte Carlo approach works better than before. Comparing with standard approach, the variance has been reduced to around a half. As what mentioned before, the result does depend on the relationship between the test functions and the initial share price. When the share price goes lower, the Longstaff-Schwartz method works better in helping the multi-level approach improving the accuracy.

Since we had hypothesis that the pair of R-basis function and spot price influence the result, we now think about adjusting the R-basis function for a bit to change the

R from $R = 5$ and $R = 4$ and hope there will be further improvement.

Table 4.7: **Estimator of Bermudan put with** $R = 4$

	λ	level value	level variance	N_l	estimator variance($1.0e-6$)
$l = 0$	0.001	0.1275	0.0603	441887	0.1364
$l = 1$	1	0.0734	0.0250	201307	0.1243
$l = 2$	10	0.0601	0.0199	126820	0.1566
$l = 3$	30	0.0070	0.0062	50000	0.1235
$l = 4$	100	0.0015	0.0069	37469	0.1851
$l = 5$	10000	0.0005	0.0069	26522	0.2620
Overall	N/A	**0.2701**	N/A	N/A	**0.9878**
Standard MC	∞	**0.2691**	2.3816	10^5	**2.3816**

The table shows that our result has been improved furthermore. The variance for the multi-level approach is reduced by 60% after we change the R-basis function to be $R = 4$. As in our hypothesis, pair of R-basis function and spot price influences the multi-level result together.

For the further improvement, we might use different test functions which can be proved to be effective. However, we still do not know which parameters influence the result most and can not find an analytical way to set up the smoothing parameters λ which influences our randomized stopping rule. The result has been improved after we adjusted by different ways.

Chapter 5

Conclusion

In the thesis, we outlined and adapted the multi-level Monte Carlo approach introduced by Mike Giles to improve the accuracy in pricing a Bermudan option. For the approximation of continuation value of the Bermudan option, we introduced and applied the Longstaff-Schwartz method with a set of efficient test functions. Meanwhile, for the multi-level Monte Carlo method approach, we introduced a randomized stopping rule instead of conventional stopping rule, i.e., we set up a probability of exercising relating to the difference between the payoff and continuation value. In Chapter 4, we gave different kind of hypothesis and adjustment for the smoothing parameters to reduce the variance by the multi-level Monte Carlo approach.

For the result, we have shown that smoothing parameters for the randomized stopping rule must be used to improve the accuracy and the result was indeed improved dramatically comparing with all the large parameters used. Meanwhile, according to the randomized stopping rule and correlation between levels, our method works better for a suitable pair of the initial spot and strike price. The variance is reduced better for the at-the-money and in-the-money Bermudan put than that for a well out-of-money put. Furthermore, we have shown that there is relationship between the test function we set. The pair of share price and test function does influence the result. For some case, the variance is reduced a bit better when we reduce the number of test functions.

However, till now, we still do not know which kind of parameters influence the result

most and this problem and result have not been addressed in any literature. The best result we have got already is that the variance has been reduced to around 1/3 of that using the standard Monte Carlo approach. We suppose that for other test functions if proved to be efficient, we might improve the result furthermore.

Chapter 6

Code and Appendix

6.1 Generate the GBM

```
% GenerateTheGBMPaths.m %%%%%%%%%%%%%%
function S=GenerateTheGBMPaths(S0,sigma,r,T,W)
% Generates a set of NxM Geometric Brownian
% paths for a set of random numbers
N=size(W,2);
L=size(W,1);
deltaT=T/L;
S(1,1:N)=S0;
for m=1:L
S(m+1,:)=S(m,:).*exp((r-0.5*sigma*sigma)*deltaT+sigma*W(m,:)*sqrt(deltaT));
end
end
```

6.2 Weighted-Laguerre Explicit

```
% LaguerreExplicit.m %%%%%%%%%%%%%%
function Phi=LaguerreExplicit(R,x)
% Generates the (weighted) laguerre polynomial Phi
switch R
```

```
    case 0
        Phi=1;
    case 1
        Phi=-x+1;
    case 2
        Phi=0.5*(x.^2-4*x+2);
    case 3
        Phi=(1/6)*(-x.^3+9*x.^2-18*x+6);
    case 4
        Phi=(1/24)*(x.^4-16*x.^3+72*x.^2-96*x+24);
    case 5
        Phi=(1/120)*(-x.^5+25*x.^4-200*x.^3+600*x.^2-600*x+120);
    case 6
        Phi=(1/720)*(x.^6-36*x.^5+450*x.^4-2400*x.^3+5400*x.^2-4320*x+720);
    otherwise
        disp('Error, R is out of range');
        Phi=0;
end

Phi=exp(-0.5*x).*Phi; % weighting used in Longstaff-Schwartz

end
```

6.3 Beta-Regression by Longstaff-Schwartz Method

```
% BetaRegression.m %%%%%%%%%%%%%%
function [value,Beta]=BetaRegression(R,S0,K,sigma,r,T,W)
% Generate the Betas used in the Longstaff-Schwartz
% algorithm by regression
% Takes in a set of random numbers from which it generates paths
% Weighted Laguerre polynomials are used as the basis functions.
%R Number of basis functions
%N Number of paths
```

```
%M Number of exercise dates (approximate Bermudan with Bermudan)
N=size(W,2);
L=size(W,1);
payoff = @(x) max(K-x,0); % Put payoff
% Phase 1 : Generate N paths and approximate the coefficients Beta
S=GenerateTheGBMPaths(S0,sigma,r,T,W);
% Now step backwards from T, evaluating the beta at each time
V=payoff(S(L+1,:)');
B_phi_phi=[];
B_V_phi=[];
Beta=[];%=zeros(R,1);
A=[];
for m=L:-1:2 % might need to fix these indices
    for i=1:R
        A(:,i)=LaguerreExplicit(i-1,S(m,:)');
    end
    B_phi_phi=A'*A;
    B_V_phi=A'*V;
    %Beta=[Beta, (B_phi_phi\B_V_phi)];
    Beta=[(B_phi_phi\B_V_phi), Beta];
    %V=max(payoff(S(m,:)') ,exp(-r*T/(L))* A*Beta(:,L+1-m));
    V=max(payoff(S(m,:)') ,exp(-r*T/(L))* A*Beta(:,1));
end
value=exp(-r*T/(L))*mean(V);
%v=max(payoff(S(1,1)),mean(V));
save Beta Beta;
```

6.4 Stopping ValueVec

```
% StoppingValueVec.m %%%%%%
function [vec]=StoppingValueVec(r,sigma,T,R,S0,K,lambda,W,Beta)
payoff = @(x) max(K-x,0); % Put payoff
if size(Beta,2)==0
```

```
    S=GenerateTheGBMPaths(S0,sigma,r,T,W);
    Payoff=payoff(S(2,:));
    probability=normcdf(sqrt(2)*lambda*...
     Payoff);
    vec=exp(-r*T)*Payoff.*probability;
else

    S=GenerateTheGBMPaths(S0,sigma,r,T,W);
    N=size(W,2);
    L=size(W,1);
    Phi=[];
    Payoff=payoff(S(2:L+1,:));
    ContinuationValue=zeros(L-1,N);
    for j=2:L
      for i=1:R
        ContinuationValue(j-1,:)=ContinuationValue(j-1,:)+...
            exp(-r*T/(L))*Beta(i,j-1)*...
            LaguerreExplicit(i-1,S(j,:));
      end

    end

    ContinuationValue=[ContinuationValue;payoff(S(L+1,:))];

    probability=normcdf(sqrt(2)*lambda*...
     (Payoff-ContinuationValue));
    probability(L,:)=normcdf(sqrt(2)*lambda*...
     Payoff(L,:));



    probTemp=1-probability;
    probTemp=cumprod(probTemp,1);
```

```
    probability(2:L,:)=probability(2:L,:)...
        .*probTemp(1:L-1,:);

    discount=exp(-r*T./(1:L));

    vec=discount*(probability.*Payoff);
end


function [v]=normcdf(x)
    v=0.5*erfc(-x/sqrt(2));
```

6.5 Main Function of Multi-level Stopping Value

```
%parameters
r=0.05;
sigma=0.4;
T=1;
R=4;
S0=10;
K=10;
levels=5;
starting_level=0;
NumPaths=10000;

%W=randn(2^5,100000);
%[v,Beta]=BetaRegression(R,S0,K,sigma,r,T,W);

%multilevel MC
load Beta Beta;
MLvalue=[];
MLvariance=[];
```

```
%set up N: number of paths per level
N=zeros(levels+1,1);
for i=1:(levels+1)
    N(i)=NumPaths;
end
%N=round(N_l);
%tuning the smoothing parameters
lambda=[];
%Using smoothing lambda
%lambda=[10000 10000 10000 10000 10000 10000];
lambda=[0.001 1 10 30 100 10000];
%lambda=[0.1 0.5 4 10 100 10000];

%level 0

ind=[];
for j=1:(2^starting_level-1)
      ind(j)=j*2^(levels-starting_level);
end
beta=Beta(:,ind);
W=randn(2^starting_level,N(starting_level+1));
vec0=StoppingValueVec(r,sigma,T,R,S0,K,lambda(starting_level+1),W,beta);
MLvalue=mean(vec0);
MLvariance=var(vec0);

vec0=[];


%level i
if starting_level<levels
for i=(starting_level+1):levels
   %generate BM
    l=2^i;
```

```
    n=N(i);
    %fine scale
    W=randn(l,n);
    ind=[];
    for j=1:(2^i-1)
        ind(j)=j*2^(levels-i);
    end
    beta=Beta(:,ind);
    [vec_fine]=StoppingValueVec(r,sigma,T,R,S0,K,lambda(i+1),W,beta);

    %coarse scale
    %reducing betas
    even_ind=2:2:(l-2);
    beta=beta(:,even_ind);
    %aggregating W
    even_ind=[even_ind,l];
    odd_ind=1:2:(l-1);
    W=(W(even_ind,:)+W(odd_ind,:))/sqrt(2);
    [vec_coarse]=StoppingValueVec(r,sigma,T,R,S0,K,lambda(i),W,beta);
    MLvalue=[MLvalue; mean(vec_fine-vec_coarse)];
    MLvariance=[MLvariance;var(vec_fine-vec_coarse)];

end
end
MLvalue;
value=sum(MLvalue)
MLvariance;



h_l=[];
h_l=0.5.^(starting_level:levels);
p2=2.^(starting_level:levels)';
```

```
%proportion=[];
%proportion=(1./sqrt(h_l))*sqrt(MLvariance);
proportion=sqrt(p2.*MLvariance);
Totalcost=2^levels*NumPaths;
constant=Totalcost/sum(proportion);
N_l=[];
N_l=constant.*sqrt(MLvariance./p2);
N_l=round(N_l)
level_variance=MLvariance./N_l;
variance=sum(level_variance)
```

Table 6.1: **Longstaff-Schwartz Regression Coefficient**

3447.75	1042.04	301.73	61.51	13.149
323.45	103.06	56.17	9.65	5.686
769.64	252.01	99.52	19.9	7.316
324.19	103.76	56.25	9.61	5.557
425.67	140.14	67.49	12.4	5.952
421.97	135.67	64.96	11.41	5.605
318.26	99.68	53.84	8.56	5.009
305.64	95.11	52.3	8.12	4.855
285.86	86.68	49	7.05	4.508
223.49	62.77	40.78	4.7	3.934
164.73	39.63	32.58	2.27	3.310
149.21	32.96	29.94	1.38	3.003
146.43	30.91	28.76	0.89	2.758
121.11	20.7	24.95	-0.33	2.361
93.94	9.56	20.84	-1.57	1.987
71.27	0.24	17.32	-2.68	1.629
47.81	-10.32	13	-4.16	1.136
33.81	-16.82	10.26	-5.11	0.764
25.52	-20.78	8.46	-5.76	0.474
7.63	-28.87	5.15	-6.88	0.067
-6.73	-35.78	2.13	-7.97	-0.355
-14.9	-40.11	0.01	-8.8	-0.721
-16.82	-41.35	-0.71	-9.11	-0.962
-26.06	-46.3	-3.18	-10.15	-1.451
-33.51	-50.29	-5.14	-10.97	-1.860
-41.87	-54.81	-7.4	-11.92	-2.333
-47.99	-58.22	-9.16	-12.72	-2.786
-56.67	-62.87	-11.49	-13.7	-3.281
-68.52	-68.98	-14.49	-14.92	-3.909
-81.74	-75.78	-17.91	-16.33	-4.676
-110.52	-89.75	-24.57	-18.88	-5.957

Bibliography

[1] P.Glasserman,"Monte Carlo Methods in Financial Engineering",*Springer*.

[2] P.E.Kloeden, E.Platen, "Numerical Solution of Stochastic Differential Equations", *Springer*.

[3] M.Giles, "Multi-level Monte Carlo path simulation", *May, 2006*.

[4] A.R.Choudhury, A.King, S.Kumar, Y.Sabharwal, "Optimizations in Financial Engineering: The Least-Squares Monte Carlo method of Longstaff and Schwartz", *October, 2007*.

[5] M.Cerrato, K.Cheung, "Valuing Bermudan Style Options by Least Square Methods", *Aprial, 2006*.

[6] F.A.Longstaff, E.S.Schwartz, "Valuing Bermudan Options by Simulation: A Simple Least-Squares Approach".

[7] M.Giles, D.J.Higham, M.Mao, "Analysing Multi-level Monte Carlo for Options with Non-globally Lipschitz Payoff", *August, 2008*.

[8] Mike Giles"Lectures for Numercial Method, Monte Carlo",*University of Oxford, 2010*.

[9] E.Clement, D.Lamberton, P.Protter, "An Analysis of a Least Squares Regression Method for Bermudan Option Pricing", *December, 2001*.

[10] R.Bilger, "Valuation of Bermudan-Asian Options with the Longstaff-Schwartz Algorithm", *April, 2004*.