Valuation of American Options using a Grid Lattice Algorithm

Degree Dissertation

for the

Master Examination in Economics and Finance

at the

Faculty of Economics and Social Sciences

of the

Eberhard Karls Universität

Tübingen

Examiner:

Prof. Dr.-Ing. Rainer Schöbel

Submitted by:

Bao Tran Tong

Born in 10 March 1991

Date of submission: 19 March 2018

Abstract

In this thesis, we address the problem of pricing American options. We introduce here a grid lattice algorithm, which is simple and easy to implement computationally. The key idea is to build a time adjusted grid lattice with time discretization approach, then use backward induction procedure to calculate the option price. In order to illustrate the efficiency of this algorithm, we price both European and American options and compare the results with those from other numerical methods. The results will be analyzed in terms of accuracy and computation time. We also examine the convergence behavior of option prices evaluated by the lattice algorithm. The analysis suggests that the grid lattice algorithm is superior to other numerical methods, since it demonstrates a quick convergence and high accuracy with low computational time.

Keywords: Grid lattice algorithm, American option, European option

Acknowledgements

I would like to express my great gratitude to my supervisor Prof. Dr.-Ing. Rainer Schöbel for the useful comments, remarks and engagement through the writing process of this master thesis. His patience, motivation, enthusiasm, immense knowledge and sense of humor are always a great support to me in completing my thesis. Furthermore I would like to thank Mrs. Klöckner for helping me making appointments with Prof. Schöbel.

Also, I would like to thank my parents, my sister, and my grandparents for their continuous encouragement and support throughout my years of study and through the process of writing this thesis.

Contents

Abstract ii

Acknowledgements iii

1 Introduction 1

1.1 The Background to the Research 1

1.2 Problem Formulation 3

1.3 Organization of the Thesis 8

2 Theoretical Foundations 9

2.1 Finite Difference Scheme 9

2.2 Least Squares Monte Carlo Simulation 15

2.3 Grid Lattice Algorithm 21

3 Numerical Implementations 27

3.1 Implementation of Finite Difference Scheme 27

3.2 Implementation of Least Squares Monte Carlo Method 31

3.3 Implementation of Grid Lattice Algorithm 34

3.4 Implementation of the Main Program 38

3.4.1 Test Design for European Options 38

3.4.2 Test Design for American Options 40

4 Research Results and Discussion 43

4.1 Efficiency Test for European Options 43

4.2 Efficiency Test for American Options 49

Appendix **56**

A Results from Matlab Implemetations **56**

A.1 European put option: Test 1 56

A.2 European put option: Test 2 59

A.3 American put option: Test 1 60

A.4 American put option: Test 2 63

Bibliography **64**

List of Figures

2.1 Function u(x) 10
2.2 Retangular grid in finite difference method 11
2.3 Grid lattice for the underlying process y_t over two discrete time periods 25

3.1 Implementation of Implicit Scheme (Part 1) 29
3.2 Implementation of Implicit Scheme (Part 2) 30
3.3 Implementation of LSM method (Part 1) 32
3.4 Implementation of LSM method (Part 2) 33
3.5 Implementation of Grid Lattice Algorithm (Part 1) 35
3.6 Implementation of Grid Lattice Algorithm (Part 2) 37
3.7 Implementation of Black-Scholes Model 39
3.8 Implementation of European Option Test 40
3.9 Implementation of American Option Test 41

4.1 Convergence of a European put option using different grid spaces in the lattice algorithm 44
4.2 Relationship between European put option price and number of determination dates using different grid spaces in the lattice algorithm . 46
4.3 Convergence result of an American put option using different grid spaces in the lattice algorithm 50
4.4 Convergence result of an American put option using different grid spaces in the lattice algorithm 51

List of Tables

1.1 Itô multiplication rule for dS^2 5

2.1 Linear system to solve in implicit scheme 13
2.2 Stock price simulation paths 16
2.3 Cash flows at time t=3 16
2.4 Values of vector X and Y for regression at time t=2 17
2.5 Optimal early exercise decision at time 2 18
2.6 Cash flows at time t=2 18
2.7 Values of vector X and Y for regression at time t=1 19
2.8 Optimal early exercise decision at time 1 19
2.9 Stopping rule 20
2.10 Option cash flow matrix 20

3.1 Summary of Matlab files 42

4.1 Input data for a European put option 43
4.2 Convergence result of a European put option using different grid spaces in the lattice algorithm 45
4.3 Various settings to compare the lattice algorithm and BSM in pricing European put option 47
4.4 Comparison in pricing European put options between Black-Scholes formula and lattice algorithm 48
4.5 Input data for an American put option 49
4.6 Convergence result of an American put option using different grid spaces in the lattice algorithm 51

4.7 Various settings to compare the lattice algorithm with other two methods in pricing American put option 52

4.8 Comparison in pricing American put options among finite difference method, least squares Monte Carlo simulation, and lattice algorithm 54

Chapter 1

Introduction

1.1 The Background to the Research

Options are very popular financial products. Most stock exchanges around the world offer options. Initially, people use options as a tool to hedge the risk of price fluctuations. However, many financial firms nowadays use options to speculate on prices. This activity of financial institutes has created very dynamic markets for options. There are two type of options: put option and call option. Put option offers its owner the right to sell the underlying asset at the exercise price. Call option offers its owner the right to buy the underlying asset at the predetermined exercise price. There is also a difference between long position and short position in option trading. Long options means that the holders have the right but not the obligation to exercise the option, whereas the short position obliges the holders to exercise the option if the person who holds the long position decides to exercise the option. We also can categorize options into American options and European options. European options can only be exercised at the maturity. In contrast, American options can be exercised at any time up to the maturity date. Due to this flexibility property, the price of American option is equal or higher than that of European option.

Due to the nature of European options, pricing European option is much simpler than pricing American options. Black-Scholes model (also called Black-Scholes-Merton) (1973) are developed by Black and Scholes to value European options. This

model offers a closed-form solution for European option price. According to this model, theoretical value of European-style options can be calculated using current stock price, option strike price, expected dividend, interest rates, time to maturity, and expected volatility. However, valuing American options is more complicated. The possibility of early exercise in American options leads to complications for analytic calculations. Many methods are developed to estimate the value of American options. These methods work basing on the fact that American option price satisfies a partial differential equations (PDE) with some boundary conditions. Methods dealing with American options can be analytical approximations or numerical methods. Analytical approximations are developed by Geske and Johnson (1984), MacMillan (1986), Barone-Adesi and Whaley (1987), and Ju (1998). Zhu (2006) contributed a remarked results in valuing American options by using Taylor series with infinitely many terms. By this approach, Zhu (2006) found an exact and explicit solution of the Black-Scholes equation for the valuation of American put options. However, this method confronts the difficulty in numerical implementation. The problem arises due to the fact that infinite sum is likely to yield many computation errors. Zhao and Wong (2012) extends Zhu's work (2006) to price American options under general diffusion processes.

Recently, the area of numerical methods attracts many financial researchers. For example, Brennan and Schwartz (1978), and Courtadon (1982) used finite difference methods for option valuation. Cox (1979) values American options using binomial tree method. Extending binomial tree method, Jarrow and Rudd (1983), Hull and White (1999), and Boyle [2] created the trinomial model for option pricing by adjusting some parameters of binomial model. Trinomial model is more efficient than the binomial model since it approaches the accurate value faster when three-pronged path is used instead of two-pronged path. Jacka (1991), Kim (1990), and Carr et al. (1992) expresses the value of American option through an integral formula, in which American option is the sum of corresponding European option and integral function of free boundary, then solve for optimal exercise boundary and option price using recursive numerical algorithm. Monte Carlo simulation method of Grant et al. (1996)

produces very good results in valuing American options. Longstaff and Schwartz (2001) adapted Monte Carlo simulation method, and used least squares technique to solve American option pricing problem quite successfully. Zhu (2006) uses the Laplace transform method for American option problem. The common ground for Monte Carlo simulation method, Longstaff and Schwartz least squares method, and Laplace transform method is the time-recursive technique. The general idea is to discretize the lifetime of an option and find its optimal exercise values backwardly in time. At each time step, we need to calculate the value of American option. This process is repeated until we reach the starting point of the time line. With the help of computers, we can easily obtain those values with short computation time and minimal pricing errors. Because numerical methods are based on the capacity of computer and on the way we discretize time, we need to choose the appropriate discretization and appropriate computers with enough capacity to perform the tasks. Other issues with numerical methods are the convergence, consistency and stability of these methods. We also need to consider the trade-off between the accuracy and time consumption of each method in producing the pricing approximations.

In this thesis, I want to concentrate on grid lattice algorithm, a numerical method used to price American options. The performance of grid lattice algorithm will be compared to those of finite difference method and least squares Monte Carlo simulation. In order to do so, we calculate prices of both European and American options using a specific set of parameters for each type of options. The results produced by these three methods are analyzed in terms of accuracy and efficiency. Without loss of generality, we only consider pricing put option in this thesis, since pricing call options can be done using the same process with a slightly change according to the nature of call options.

1.2 Problem Formulation

In this section, we will go into details a mathematical formula for finding fair value of a contingent claim. American option and European option in our current case are

specific examples of contingent claims. Suppose that we have a contingent claim or a financial asset $P(S,t)$ with strike price K and time to maturity T. This contingent claim is written on the underlying stock S. Other parameters in pricing a contingent claim can be defined as follows:

S: market price of the underlying stock
r: continuously compounded risk free-rate of interest
μ: expected rate of return of the underlying stock
q: continuous dividend rate
σ: volatility of the underlying stock
$P(S,t)$: value of the contingent claim at time t

An important assumption when pricing contingent claims is that, the pricing of contingent claims occurs in the risk-neutral world, which is governed by the Martingale measure. Another assumption is, the underlying stock price S follows a geometric Brownian motion. The stock dynamics in the risk neutral world can be presented by the following equation:

$$dS = \mu S dt + \sigma S dW \tag{1.2.1}$$

where W is the standard Brownian motion.

In the above equation, (dS) is the increment of the risk factor, $(\mu S dt)$ is the drift term and $(\sigma S dW)$ is the diffusion term. If there is a continuous dividend rate paid on the underlying, the drift term will become $((\mu - q)Sdt)$ instead of $(\mu S dt)$. However, in this thesis, we assume that there is no dividend payment. $P(S,t)$ is a contingent claim on the risk factor of stock price. Using Itô's lemma, we can derive the dynamics of the fair value of the contingent claim $P(S,t)$:

$$dP = P_t dt + P_s dS + \frac{1}{2} P_{ss} dS^2 \tag{1.2.2}$$

where P_t: first derivative of $P(S,t)$ with respect to time t
P_s: first derivative of $P(S,t)$ with respect to S

P_{ss}: second derivative of $P(S,t)$ with respect to S.

Itô multiplication rule for dS^2 is presented in the following table:

	dt	**dW**
dt	0	0
dW	0	dt

Table 1.1: Itô multiplication rule for dS^2

Applying Itô multiplication rule for dS^2, we get:

$$dS^2 = (\mu S dt + \sigma S dW)^2 = \sigma^2 S^2 dt \tag{1.2.3}$$

Inserting the formula of dS in equation (1.2.1) and dS^2 in equation (1.2.3) into equation (1.2.2) , we have:

$$dP = P_t dt + P_s(\mu S dt + \sigma S dW) + \frac{1}{2} P_{ss}\sigma^2 S^2 dt \tag{1.2.4}$$

$$dP = (P_t + P_s \mu S + \frac{1}{2} P_{ss}\sigma^2 S^2)dt + \sigma S P_s dW \tag{1.2.5}$$

Equation (1.2.5) represents the dynamics of the asset or option in our current case. In this equation, we denote:

$$(P_t + P_s \mu S + \frac{1}{2} P_{ss}\sigma^2 S^2) = \beta(S,t)P$$

$$\sigma S P_s = \Psi(S,t)P$$

With this notation, we get:

$$dP = \beta(S,t)P dt + \Psi(S,t)P dW \tag{1.2.6}$$

where

$$\beta = \beta(S,t) = \frac{1}{P}(P_t + P_s \mu S + \frac{1}{2} P_{ss}\sigma^2 S^2) \tag{1.2.7}$$

$$\Psi = \Psi(S,t) = \frac{1}{P}\sigma S P_s \tag{1.2.8}$$

The dynamics of asset $P(S,t)$ in equation (1.2.6) is comparable to the dynamics of the underlying stock S in equation (1.2.1).

$$dS = \mu S dt + \sigma S dW$$
$$dP = \beta P dt + \Psi P dW$$

Using the definition of β in equation (1.2.7), we get the Partial Differential Equation (PDE):

$$\frac{1}{2}P_{ss}\sigma^2 S^2 + P_s \mu S + P_t - \beta P = 0 \tag{1.2.9}$$

The problem for $P(S,t)$ is not well-posed since we do not know $\beta(S,t)$. To solve this problem, we have two solutions basing on two cases:

Case 1: Complete market case

In complete market case, the underlying rick factor is a continuously traded asset. For example, stocks of big companies are traded regularly on stock exchanges, and they have high market liquidity. We assume that investors are risk-averse, therefore, the excess return per unit of risk is constant for assets with identical risk factors. It means that, a put option written on an underlying stock has the same excess return per unit with the stock itself. This leads to the following relationship:

$$\frac{\mu - r}{\sigma} = \frac{\beta(S,t) - r}{\Psi(S,t)} = \lambda(S,t) \tag{1.2.10}$$

where $\lambda(S,t)$ is the market price of risk.

Substituting the formulas for $\beta(S,t)$ and $\Psi(S,t)$ into equation (1.2.10), we have:

$$\frac{\mu - r}{\sigma} = \frac{\frac{1}{P}(P_t + P_s \mu S + \frac{1}{2}P_{ss}\sigma^2 S^2) - r}{\frac{1}{P}\sigma S P_s} \tag{1.2.11}$$

$$\frac{\mu - r}{\sigma} = \frac{P_t + P_s \mu S + \frac{1}{2}P_{ss}\sigma^2 S^2 - rP}{\sigma S P_s} \tag{1.2.12}$$

$$(\mu - r) S P_s = P_t + P_s \mu S + \frac{1}{2}P_{ss}\sigma^2 S^2 - rP \tag{1.2.13}$$

Rearrange we have:

$$\frac{1}{2}P_{ss}\sigma^2S^2 + rP_sS + P_t - rP = 0 \tag{1.2.14}$$

Our duty is to solve (1.2.14) in addition with some boundary conditions, which depends on the nature of the derivative that we need to determine its value.

Case 2: Incomplete market case

In incomplete market case, the underlying risk factor is not continuously traded. For example, it could be interest rate or stocks that have very limited liquidity. We have the following relationship:

$$\frac{\beta(S,t) - r}{\Psi(S,t)} = \lambda(S,t) \tag{1.2.15}$$

where $\lambda(S,t)$ comes from outside information.

Using formulas for $\beta(S,t)$ and $\Psi(S,t)$ as above, we have:

$$\frac{\frac{1}{P}(P_t + P_s\mu S + \frac{1}{2}P_{ss}\sigma^2S^2) - r}{\frac{1}{P}\sigma SP_s} = \lambda(S,t) \tag{1.2.16}$$

$$(P_t + P_s\mu S + \frac{1}{2}P_{ss}\sigma^2S^2) - rP = \lambda\sigma SP_s \tag{1.2.17}$$

$$\frac{1}{2}P_{ss}\sigma^2S^2 + (\mu - \lambda\sigma)SP_s + P_t - rP = 0 \tag{1.2.18}$$

In the scope of this thesis, we only consider the case of complete market. Therefore, we need to price the asset that satisfies the following PDE:

$$\frac{1}{2}\sigma^2S^2P_{ss} + rSP_s + P_t - rP = 0 \tag{1.2.19}$$

Both European option and American option satisfy the above PDE. In case of European option, it can be only exercised at the maturity, so pricing European option is much simpler than pricing American option. To price European option, beside the specified PDE in (1.2.19), we need some boundary conditions. If it is a European put option, we have the following condition:

$$P(S,T) = max(K - S(T), 0) \tag{1.2.20}$$

In order to solve the following system analytically,

$$\frac{1}{2}\sigma^2 S^2 P_{ss} + rSP_s + P_t - rP = 0$$
$$P(S,T) = max(K - S(T), 0)$$

we can use some methods such as transformation of variables, Feyman-Kac theorem, etc. However, the discussion of these methods is out of scope of this thesis. Here we only focus on numerical methods to solve the above PDE.

1.3 Organization of the Thesis

This thesis is organized as follows. Chapter 1 will highlight the background to the research and the mathematics of option pricing. In Chapter 2, we focus on theoretical aspects of three numerical methods used in this thesis: finite difference method, least squares Monte Carlo simulation, and the grid lattice algorithm. Chapter 3 illustrates the details of numerical implementations. In particular, we demonstrate how these numerical methods are applied using Matlab programming. Chapter 4 presents the research results from running Matlab programs and discussion in details in order to extract the main findings in this thesis. Conclusion section gives concluding remarks and suggests what can be done further in the related research topic.

Chapter 2

Theoretical Foundations

In this chapter, we will go through the theoretical foundations of all three numerical methods discussed in this thesis. Basically, we will see how these methods solve the problem of pricing European option and American option. Three methods are Finite difference scheme, Least square Monte Carlo simulation, and Grid lattice algorithm.

2.1 Finite Difference Scheme

Numerical methods are frequently used to solve partial differential equations by John von Neumann in the mid-1940s. Numerical methods gain popularity due to their practical applications. There are two most important numerical methods: Finite difference method and Finite Element Method. In the scope of this thesis, we only discuss the Finite difference method. Finite difference method becomes more popular after Second World War, with the widespread use of computers. Finite difference theory made further progress for initial values problem and parabolic problems in the 1950s and 1960s, when the concept of stability is discovered in the Lax equivalence theorem.

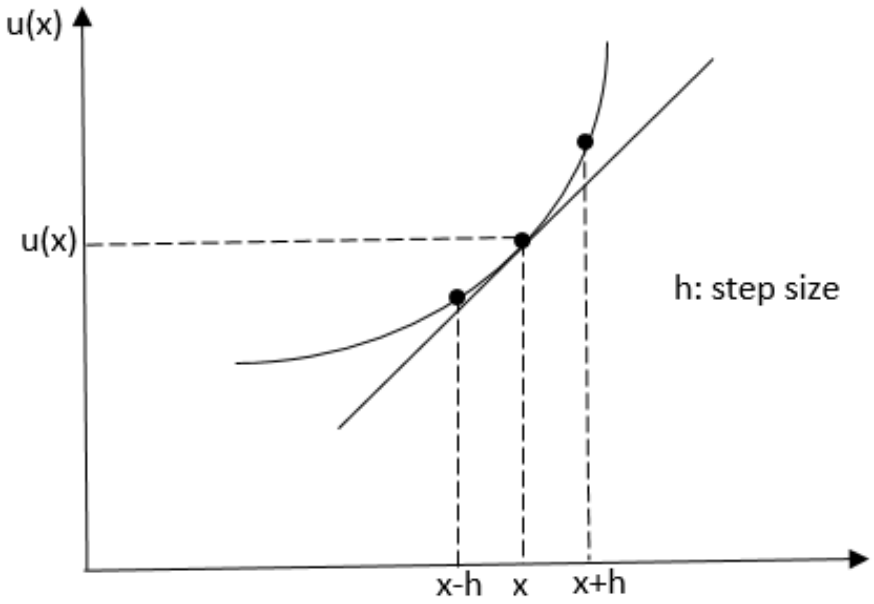
For our valuation problem, we need to price the derivative $P(S,t)$ on an underlying stock S. This derivative satisfies the Black-Scholes-Merton PDE that we derive

in Chapter 1:

$$\frac{1}{2}\sigma^2 S^2 P_{SS}(S,t) + rSP_S(S,t) - rP(S,t) + P_t(S,t) = 0 \tag{2.1.1}$$

As we can see in the PDE, there are some derivatives: $P_{SS}(S,t)$, $P_S(S,t)$, and $P_t(S,t)$. The idea of finite difference method is to approximate these derivatives in the above equation using differential quotients. We have two dimensions: time and space. The domain is partitioned in time and in space, and approximations of the solution are computed at the time and space points. We have three main finite difference schemes: explicit scheme, implicit scheme, Crank-Nicolson scheme. Consider one-dimensional case with a function $u(x)$.

Figure 2.1: Function u(x)



The core idea behind any finite difference scheme is linked to the definition of the derivative of a smooth function u at a point $x \in \mathbb{R}$. Approximations for the first derivative and second derivative can be computed as follows:

1. Forward difference approximation:

$$u_x = \frac{du}{dx} \approx \frac{1}{h}[u(x+h) - u(x)] + error \tag{2.1.2}$$

2. Backward difference approximation:

$$u_x = \frac{du}{dx} \approx \frac{1}{h}[u(x) - u(x-h)] + error \tag{2.1.3}$$

3. Central difference approximation:

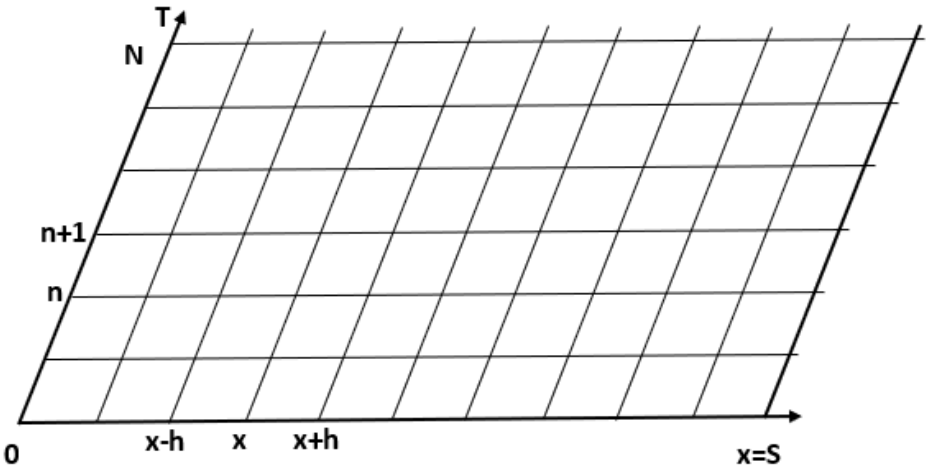
$$u_x = \frac{du}{dx} \approx \frac{1}{2h}[u(x+h) - u(x-h)] + error \tag{2.1.4}$$

4. Approximation for second derivative:

$$u_{xx} = \frac{d^2u}{dx^2} \approx \frac{1}{h^2}[u(x+h) - 2u(x) + u(x-h)] + error \tag{2.1.5}$$

When h approaches to zero, the quotients on the right-hand side of the above equations give good approximations of the derivatives. An approximation is good when the error in the approximation tends towards zero when h goes close to zero. It is possible to estimate the error using Taylor series expansion, in case the function $u(x)$ is smooth enough in the neighborhood of x.

Figure 2.2: Retangular grid in finite difference method



In this thesis, we only focus on Implicit difference scheme. To solve the PDE, we establish a rectangular grid with two dimensions: time and space as illustrated in Figure 2.2. The horizontal line represents the stock price S and the vertical line represents time line. In order to implement a finite difference scheme, we define two new independent variables x and τ:

$$\begin{aligned} x = S = ih \qquad & \forall i \in [0, M] \quad with \quad x_{max} = M.h \\ \tau = T - t = nk \quad & \forall n \in [0, N] \quad with \quad T = N.k \end{aligned}$$

where i and n are index numbers, h is the space step size and k is the time step size.

Discretizing the PDE (2.1.1) at the point $(x = ih, \tau = nk)$, we have:

$$a(i,n)u_{xx}(i,n) + b(i,n)u_x(i,n) + c(i,n)u(i,n) - u_\tau(i,n) = 0 \tag{2.1.6}$$

where

$$\begin{aligned} a(i,n) &= \frac{1}{2}\sigma^2 i^2 h^2 \\ b(i,n) &= rih \\ c(i,n) &= -r = const \end{aligned}$$

We use the following difference approximations for implicit scheme:

$$\begin{aligned} (u_{xx})_i^{n+1} &= \frac{1}{h^2}(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}) + \sigma(h^2) \\ (u_x)_i^{n+1} &= \frac{1}{2h}(U_{i+1}^{n+1} - U_{i-1}^{n+1}) + \sigma(h^2) \\ (u_\tau)_i^{n+1} &= \frac{1}{k}(U_i^{n+1} - U_i^n) + \sigma(k) \\ (U_i)^{n+1} &= (U_i)^{n+1} \end{aligned}$$

Plugging these approximations to the PDE, we have the solution of the implicit difference approximation at $(x = ih, \tau = (n+1)k)$:

$$d_1(i,n+1)U_{i-1}^{n+1} + d_2(i,n+1)U_i^{n+1} + d_3(i,n+1)U_{i+1}^{n+1} = d_4(i,n+1) \tag{2.1.7}$$

where:

$$\begin{aligned} d_1(i,n+1) &= -a\frac{k}{h^2} + b\frac{k}{2h} \\ &= -\frac{1}{2}\sigma^2 i^2 k + \frac{1}{2}rik \\ d_2(i,n+1) &= a\frac{2k}{h^2} - ck + 1 \\ &= \sigma^2 i^2 k + rk + 1 \\ d_3(i,n+1) &= -a\frac{k}{h^2} - b\frac{k}{2h} \\ &= -\frac{1}{2}\sigma^2 i^2 k - \frac{1}{2}rik \\ d_4(i,n+1) &= U_i^n \end{aligned}$$

$\forall i \in [1, M-1]$ and $n \in [0, N-1]$

We get a linear system to solve:

i	**0**	**1**	**2**	**3**	. . .	**M-1**	**M**	**RHS**
1	$d_1(1)$	$d_2(1)$	$d_3(1)$					$d_4(1)$
2		$d_1(2)$	$d_2(2)$	$d_3(2)$				$d_4(2)$
.								.
.								.
.								.
M-1					$d_1(M-1)$	$d_2(M-1)$	$d_3(M-1)$	$d_4(M-1)$

Table 2.1: Linear system to solve in implicit scheme

In order to solve the PDE, we still need some boundary conditions. These conditions differ depending on the type of the contingent claims. For example, in case of European put option, we have the boundary conditions in implicit difference scheme as follows:

1. $n = 0$: $U_i^0 = max(K - ih, 0) \quad \forall i \in [0, M]$
2. $i = 0$: $d_2 U_1^{n+1} + d_3 U_2^{n+1} = d_4^*$
   with $d_4^*(1, n+1) = d_4(1, n+1) - d_1(1, n+1)U_0^{n+1}$

3. $i = M$: $d_1^* U_{M-1}^{n+1} + d_2 U_M^{n+1} = d_4$
   with $d_1^*(M, n+1) = d_1(M, n+1) + d_3(M, n+1)$

Writing the equation (2.1.7) explicitly for each space level i, combined with the above boundary conditions, we have the following equation system:

$$\underbrace{\begin{pmatrix} d_2(1) & d_3(1) & & & & & \\ d_1(2) & d_2(2) & d_3(2) & & & & \\ & d_1(3) & d_2(3) & d_3(3) & & & \\ & & & \ddots & & & \\ & & & d_1(M-1) & d_2(M-1) & d_3(M-1) & \\ & & & & d_1^*(M) & d_2(M) & \end{pmatrix}}_{\underline{\underline{A}}} \underbrace{\begin{pmatrix} U_1^{n+1} \\ U_2^{n+1} \\ U_3^{n+1} \\ \vdots \\ U_{M-1}^{n+1} \\ U_M^{n+1} \end{pmatrix}}_{\underline{u}} = \underbrace{\begin{pmatrix} d_4^*(1) \\ d_4(2) \\ d_4(3) \\ \vdots \\ d_4(M-1) \\ d_4(M) \end{pmatrix}}_{\underline{d}}$$

At each time level, in order to solve linear system $\underline{\underline{A}} * \underline{u} = \underline{d}$, we implement three following steps:

Step 1: LR decomposition of $\underline{\underline{A}}$: $\underline{\underline{L}} * \underline{\underline{R}} = \underline{\underline{A}}$

Step 2: Find solution $\underline{\underline{L}} * \underline{y} = \underline{d}$ where $\underline{y} = \underline{\underline{R}} * \underline{u}$

Step 3: Find solution $\underline{\underline{R}} * \underline{u} = \underline{y}$

This process has to be repeated until the desired time level is reached.

For American put option, the problem is more complicated. At each nod on the rectangular grid, we compute the non-exercise value (or the value of corresponding European option using the above procedure) and the exercise value. Afterwards, we compare the non-exercise value and the exercise value. The value of American option will be the greater of those two values. This procedure will be illustrated in more details in Chapter 3, where the Matlab implementation is explained.

2.2 Least Squares Monte Carlo Simulation

Monte Carlo simulation is one of the most popular numerical method in option pricing. This method is invented by Stanislaw Ulam, a scientist in the fields of mathematics and nuclear physics. Ever since, Monte Carlo simulation is used in many areas, not only in mathematics and physics. Financial researchers use Monte Carlo simulation as an essential tool in option pricing and in risk management. Basically, Monte Carlo simulation deals with any problem having a probabilistic interpretation. Monte Carlo simulation becomes an important and powerful tool to deal with pricing path-dependent options. Valuing a derivative security by Monte Carlo simulation typically involves simulating paths of the stochastic process that describes the evolution of the underlying asset prices. The main idea is to represent derivative prices as expectations of random objects that can be simulated.

Least Squares Monte Carlo simulation is developed by Longstaff and Schwartz in 2001. This method values American option through an optimization algorithm based on least squares regression. As the name of the method suggests, least squares Monte Carlo method combines the standard Monte Carlo method and an algorithm based on regression. According to Longstaff and Schwartz (2001), this new technique allows the consideration of multiple risks factors and accepts any type of dynamics of these factors. The dynamics can be general stochastic process such as jump diffusions, non-Markovian processes, or general semimartingales. Besides, the advantages of simulation technique are undeniable. Simulation techniques are simple and flexible.

To understand how least squares Monte Carlo method works, we examine a specific example of valuing American put option. This example is illustrated in more details in the paper *"Valuing American Options by Simulation: A Simple Least-Squares Approach"* by Longstaff and Schwartz (2001). Assume we have an American put option with strike price $K = 1.10$, written on an non-dividend-paying stock. There is three point of times in the life of this American option, times 1, 2, and 3. It means that time three is the final expiration date of the option, while

the option can be early exercised at any point of time. The risk-free rate is 6%. As a part of least squares Monte Carlo method, we simulate eight paths of stock prices under risk-neutral measure. Table 2.2 represents the result of this stock price simulation.

Path	**t=0**	**t=1**	**t=2**	**t=3**
1	1.00	1.09	1.08	1.34
2	1.00	1.16	1.26	1.54
3	1.00	1.22	1.07	1.03
4	1.00	0.93	0.97	0.92
5	1.00	1.11	1.56	1.52
6	1.00	0.76	0.77	0.90
7	1.00	0.92	0.84	1.01
8	1.00	0.88	1.22	1.34

Table 2.2: Stock price simulation paths

Our objective is to find a stopping rule that maximizes the value of the option at each point of time along each path. Assuming that the option is not exercised early, the cash flows at the date of expiration are calculated by the formula: $\max(K - S(t = 3), 0)$ with $K = 1.10$. Table 2.3 contains cash flows at time t=3.

Path	**t=1**	**t=2**	**t=3**
1	—	—	0.00
2	—	—	0.00
3	—	—	0.07
4	—	—	0.18
5	—	—	0.00
6	—	—	0.20
7	—	—	0.09
8	—	—	0.00

Table 2.3: Cash flows at time t=3

At time 2, if the put option is in the money, the option holder must decide whether to exercise the option or not. Therefore, we only consider the paths for which the option is in the money. Looking at the Table 2.2, we see that there are only 5 paths satisfying the condition. Those are paths 1, 3, 4, 6, and 7 where the stock price is smaller than the strike price of 1.10. We store the stock prices of these paths in a vector X. At time t=2, another value we need to concern is the discounted value of cash flows received at time 3 if the put option is not exercised at time 2. The discount factor is: $e^{-r} = e^{-0.06} = 0.94176$. The discounted cash flows values are stored in vector Y. The values of vector X and Y are given in Table 2.4.

Path	**S(t=2)**	**CF(t=3)**	**X**	**Y**
1	**1.08**	**0.00**	1.08	0.00 x 0.94176
2	1.26	0.00	—	—
3	**1.07**	**0.07**	1.07	0.07 x 0.94176
4	**0.97**	**0.18**	0.97	0.18 x 0.94176
5	1.56	0.00	—	—
6	**0.77**	**0.20**	0.77	0.20 x 0.94176
7	**0.84**	**0.09**	0.84	0.09 x 0.94176
8	1.22	0.00	—	—

Table 2.4: Values of vector X and Y for regression at time t=2

The purpose of constructing vectors X and Y is that we want to estimate the expected cash flow from not exercising the option at time 2. In order to do so, we regress Y on a constant, X, and X^2. Using least squares regression technique, we obtain the result: $E[Y|X] = -1.070 + 2.983X - 1.813X^2$. With this conditional expectation function, we compute the cash flow or the value of American put option from continuation by simply plugging the values of X into the function. We call these computed values "Continuation values". On the other hand, we also need to calculate the exercise value at time 2, which equals to $(1.10 - X)$. Table 2.5 shows exercise values and continuation values.

Path	Exercise	Continuation
1	0.02	0.0369
2	—	—
3	0.03	0.0461
4	**0.13**	**0.1176**
5	—	—
6	**0.33**	**0.1520**
7	**0.26**	**0.1565**
8	—	—

Table 2.5: Optimal early exercise decision at time 2

Our next task is to compare the continuation values with the corresponding exercise values in order to determine whether it is optimal to exercise the option early. Looking at Table 2.5, it is reasonable to exercise the option at time 2 for paths forth, sixth, and seventh because the exercise values are greater than the corresponding continuation values in these paths. With this results, we update our cash flows table by including the cash flows at time t=2.

Path	t=1	t=2	t=3
1	—	0.00	0.00
2	—	0.00	0.00
3	—	0.00	0.07
4	—	**0.13**	**0.00**
5	—	0.00	0.00
6	—	**0.33**	**0.00**
7	—	**0.26**	**0.00**
8	—	0.00	0.00

Table 2.6: Cash flows at time t=2

Notice that if the option is already exercised at time 2, the cash flow at time 3

equals to zero. The same process can be applied to time 1. First, we need to determine in the money paths at time 1. Second, with these paths, we define vector Y as the discounted cash flows. Note that we use the actual realized cash flows along each path, not the conditional expected values. This practice helps prevent the upward bias in the value of the option. Next, we regress the discounted cash flows Y on underlying stock price X. Values of vector X and Y at time 1 are given in Table 2.7. The resulted regression equation at time 1 is: $E[Y|X] = 2.038 - 3.335X + 1.356X^2$.

Path	**S(t=1)**	**CF(t=2)**	**X**	**Y**
1	**1.09**	**0.00**	1.09	0.00 x 0.94176
2	1.16	0.00	—	—
3	1.22	0.00	—	—
4	**0.93**	**0.13**	0.93	0.13 x 0.94176
5	1.11	0.00	—	—
6	**0.76**	**0.33**	0.76	0.33 x 0.94176
7	**0.92**	**0.26**	0.92	0.26 x 0.94176
8	**0.88**	**0.00**	0.88	0.00 x 0.94176

Table 2.7: Values of vector X and Y for regression at time t=1

Path	**Exercise**	**Continuation**
1	0.01	0.0139
2	—	—
3	—	—
4	**0.17**	**0.1092**
5	—	—
6	**0.34**	**0.2866**
7	**0.18**	**0.1175**
8	**0.22**	**0.1533**

Table 2.8: Optimal early exercise decision at time 1

We plug the values of X into this regression to obtain the estimated continuation values at time 1. We also compute the exercise values at time 1 for in the money paths. Comparing exercise values with corresponding continuation values, it is optimal to exercise for paths forth, sixth, seventh, and eighth. Table 2.8 shows the optimal early exercise decision at time 1. At this moment, we have enough information on the exercise strategy at times 1, 2, and 3. At each point of time, if the option is exercised, the value on the specific path is one, otherwise the value will be zero. Table 2.9 depicts the stopping rule for this American put option.

Path	**t=1**	**t=2**	**t=3**
1	0	0	0
2	0	0	0
3	0	0	**1**
4	**1**	0	0
5	0	0	0
6	**1**	0	0
7	**1**	0	0
8	**1**	0	0

Table 2.9: Stopping rule

Path	**t=1**	**t=2**	**t=3**
1	0	0.00	0.00
2	0	0.00	0.00
3	0	0.00	**0.07**
4	**0.17**	0.00	0.00
5	0	0.00	0.00
6	**0.34**	0.00	0.00
7	**0.18**	0.00	0.00
8	**0.22**	0.00	0.00

Table 2.10: Option cash flow matrix

With this clear stopping rule, we now can form the final cash flow matrix. The strategy is to exercise the option where there is a one on the above table. With all the available cash flows, we now value the option by discounting each cash flow in the matrix back to time zero. The final option value is the average of all the discounted cash flows.

$$\begin{aligned} V_0 &= \frac{1}{8}(0+0+0.07e^{-3r}+0.17e^{-r}+0+0.34e^{-r}+0.18e^{-r}+0.22e^{-r}) \\ &= \frac{1}{8}(0+0+0.07e^{-3*0.06}+0.17e^{-0.06}+0+0.34e^{-0.06}+0.18e^{-0.06}+0.22e^{-0.06}) \\ &= 0.1144 \end{aligned}$$

The above example demonstrates how least squares technique can be used to estimate the conditional expectation function. This function is then used to identify the exercise strategy. Least squares Monte Carlo simulation can be done easily by using Monte Carlo simulation and a simple regression.

2.3 Grid Lattice Algorithm

In finance, lattice model is used to price derivatives in discrete time scheme. Lattice model is useful in valuation of standard options such as European options, American options as well as exotic options. The general idea of lattice model is to discretize the time period from now to the maturity of options into N discrete periods. At each point of time, the model has a specific number of outcomes. The most popular lattice model is the binomial options pricing model coined by Cox, Ross, and Rubinstein (CRR) in 1979. This model states that the underlying process has two possible outcomes at each stage. Another variant is the Trinomial tree developed by Phelim Boyle in 1986, in which the underlying process has three possible outcomes at each stage. In this paper, we analyze the grid lattice model, which is also called the rectangular lattice. Grid lattice model is more flexible by allowing the underlying process to change by multiple states.

Recall that the underlying stock price follows a stochastic differential equation (SDE) under the risk-neutral measure $\mathbb{Q}$:

$$dS_t = rS_t dt + \sigma S_t d\tilde{W}_t \tag{2.3.1}$$

where $\tilde{W}(t)$ is a standard Brownian motion under the risk-neutral measure $\mathbb{Q}$. Suppose we have $\tilde{S}$ is the discounted stock price:

$$\tilde{S}_t = e^{-rt} S_t \tag{2.3.2}$$

Applying Itô's Lemma formula, we find the SDE for $\tilde{S}_t$ as follows.

$$d\tilde{S}_t = \frac{\partial \tilde{S}_t}{\partial t} dt + \frac{\partial \tilde{S}_t}{\partial S_t} dS_t + \frac{1}{2} \frac{\partial^2 \tilde{S}_t}{\partial S_t^2} dS_t^2 \tag{2.3.3}$$

We have:

$$\begin{aligned}
\frac{\partial \tilde{S}_t}{\partial t} &= (-r) e^{-rt} S_t \\
\frac{\partial \tilde{S}_t}{\partial S_t} &= e^{-rt} \\
\frac{\partial^2 \tilde{S}_t}{\partial S_t^2} &= 0
\end{aligned}$$

Plugging these results into formula (2.3.3), we get:

$$\begin{aligned}
d\tilde{S}_t &= (-r) e^{-rt} S_t dt + e^{-rt} dS_t + 0 \\
d\tilde{S}_t &= (-r) e^{-rt} S_t dt + e^{-rt} (r S_t dt + \sigma S_t d\tilde{W}_t) \\
d\tilde{S}_t &= e^{-rt} \sigma S_t d\tilde{W}_t \\
d\tilde{S}_t &= \sigma \tilde{S}_t d\tilde{W}_t
\end{aligned}$$

From the result above, we have:

$$\frac{d\tilde{S}_t}{\tilde{S}_t} = \sigma d\tilde{W}_t \tag{2.3.4}$$

To solve for $\tilde{S}_t$, we take the integral for both sides of equation (2.3.4):

$$\int_0^t \frac{d\tilde{S}_t}{\tilde{S}_t} = \int_0^t \sigma d\tilde{W}_t \tag{2.3.5}$$

The left-hand side of (2.3.5):

$$\int_0^t \frac{d\tilde{S}_t}{\tilde{S}_t} = ln\tilde{S}_t - ln\tilde{S}_0 + \frac{1}{2}\sigma^2(t-0) \tag{2.3.6}$$

$$\int_0^t \frac{d\tilde{S}_t}{\tilde{S}_t} = ln\tilde{S}_t - ln(e^0 S_0) + \frac{1}{2}\sigma^2 t \tag{2.3.7}$$

$$\int_0^t \frac{d\tilde{S}_t}{\tilde{S}_t} = ln\tilde{S}_t - ln(S_0) + \frac{1}{2}\sigma^2 t \tag{2.3.8}$$

Right-hand side of (2.3.5):

$$\int_0^t \sigma d\tilde{W}_t = \sigma(\tilde{W}_t - \tilde{W}_0) \tag{2.3.9}$$

$$\int_0^t \sigma d\tilde{W}_t = \sigma(\tilde{W}_t - 0) = \sigma\tilde{W}_t \tag{2.3.10}$$

Substituting the left-hand side and right-hand side results to equation (2.3.5), we get the following results:

$$ln\tilde{S}_t - ln(S_0) + \frac{1}{2}\sigma^2 t = \sigma\tilde{W}_t \tag{2.3.11}$$

$$ln\tilde{S}_t = lnS_0 - \frac{1}{2}\sigma^2 t + \sigma\tilde{W}_t \tag{2.3.12}$$

$$\tilde{S}_t = S_0 \exp(-\frac{1}{2}\sigma^2 t + \sigma\tilde{W}_t) \tag{2.3.13}$$

Taking the expectation of the above equation at time t_0:

$$\begin{aligned}
\mathbf{E}_{\mathbb{Q}}[\tilde{S}_t] &= \mathbf{E}_{\mathbb{Q}}[S_0 exp(-\frac{1}{2}\sigma^2 t + \sigma\tilde{W}_t)] \\
&= S_0 \frac{1}{\sqrt{2\pi t}} \int_{-\infty}^{\infty} e^{-\frac{\sigma^2}{2}t + \sigma w} e^{\frac{-w^2}{2t}} dw \\
&= S_0 \frac{1}{\sqrt{2\pi t}} \int_{-\infty}^{\infty} e^{-\frac{(w-\sigma t)^2}{2t}} dw \\
&= S_0 \frac{1}{\sqrt{2\pi t}} \int_{-\infty}^{\infty} e^{-\frac{y^2}{2t}} dw \\
&= S_0
\end{aligned}$$

where $y = w - \sigma t$

In order to numerically evaluate the integral $\int_{-\infty}^{\infty} e^{-\frac{(w-\sigma t)^2}{2t}} dw$, we choose a non-negative number A and calculate $\int_{-A}^{A} e^{-\frac{(w-\sigma t)^2}{2t}} dw$. Notice that the function $e^{-\frac{(w-\sigma t)^2}{2t}}$ is symmetric and centered at σt, it is better to evaluate the above function in the interval $[-A + \sigma t, A + \sigma t]$. In other words, we approximate $\int_{-A+\sigma t}^{A+\sigma t} e^{-\frac{(w-\sigma t)^2}{2t}} dw$. Instead of using the Brownian motion process $\tilde{W}$, we can use the process $Y_t = \tilde{W} - \sigma t$. Y_t is also a Brownian motion with drift under the risk neutral measure $\mathbb{Q}$. The function we need to estimate now is $\int_{-A}^{A} e^{-\frac{y^2}{2t}} dw$. If $\tilde{W} \in [-A + \sigma t, A + \sigma t]$ and $Y_t = \tilde{W} - \sigma t$, then we have:

$$Y_t \in [-A, A] \Leftrightarrow \tilde{W} \in [-A + \sigma t, A + \sigma t]$$

This grid lattice is called time-adjusted grid lattice. This variable transformation ensures that the distribution is symmetric and centered around zero. This property of the distribution helps increase the accuracy when we express stock price in discrete form and use numerical integration to compute the expectation (Xiao 2011). With this variable transformation, we can rewrite the equation (2.3.13) as follows.

$$\tilde{S}_t = S_0 \exp(-\frac{\sigma^2}{2} t + \sigma \tilde{W}_t)$$
$$\tilde{S}_t = S_0 \exp(\frac{\sigma^2}{2} t + \sigma \tilde{Y}_t)$$

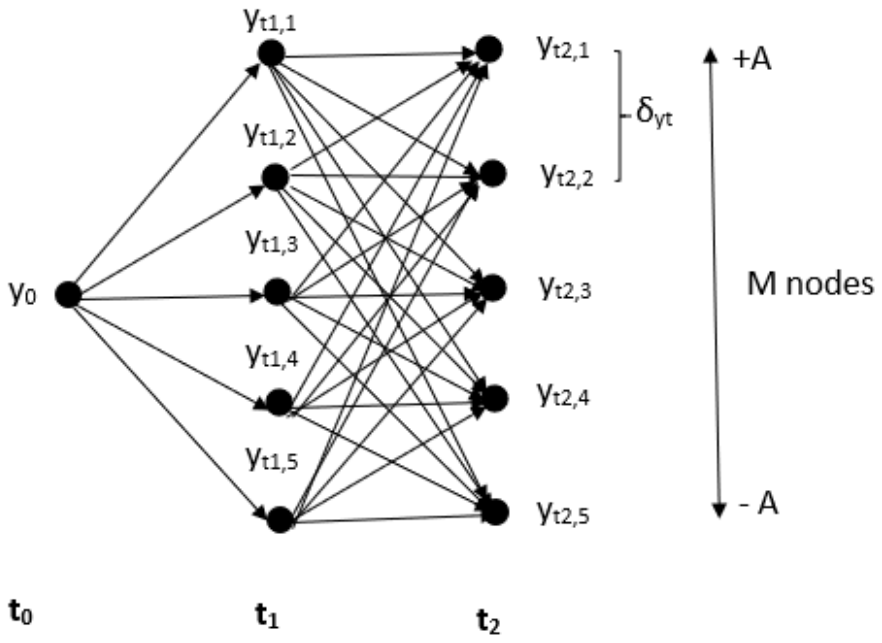
Additionally, we have the relation between S_t and $\tilde{S}_t$: $\tilde{S}_t = e^{-rt} S_t$. With this relation, we can easily find the formula for S_t:

$$S_t = \tilde{S}_t \exp(rt) = S_0 \exp(rt + \frac{\sigma^2}{2} t + \sigma Y_t) \tag{2.3.14}$$

Now we start constructing the grid lattice procedure to model the process in (2.3.14). First of all, we discretize the time period [0,T] into N periods. Specifically, the time line will be $0 = t_0 < t_1 < ... < t_N = T$. At the initial time t_0, the underlying process y_t takes initial value of y_0. At each date t_i, the underlying process y_{t_i} will be discretized into a number of vertical nodes. The number of vertical nodes can affect the convergence rate. We discretize the Brownian motion to be equally spaced as a grid of node $y_{t,j}$ where $j = 1, ..., M_t$. The equal space between the nodes $\delta_{yt} = y_{t,j} - y_{t,j-1}$ at each determination date depends on our requirement of accuracy. In the next determination date, the underlying process y_{t_i} can evolve to

any discrete state with a certain transition probability. Figure 2.3 illustrates the grid lattice with two determination dates and five discrete states at each point of time.

Figure 2.3: Grid lattice for the underlying process y_t over two discrete time periods



The discrete form of stock price conditioning on the state $y_{t,j}$ at the time t is given by:

$$S_{t,j} = S_0 \exp(rt + \frac{\sigma^2}{2}t + \sigma y_{t,j}) \tag{2.3.15}$$

Another aspect we need to consider is the transition probability. As in equation (2.3.13), the underlying state process $\tilde{W}$ is a standard Brownian motion. Therefore, the transition probability density from state $(\tilde{w}_{t,j}, t)$ to state $(\tilde{w}_{T,k}, T)$ is given by

$$p(\tilde{w}_{t,j}, t; \tilde{w}_{T,k}, T) = \frac{1}{\sqrt{2\pi(T-t)}} \exp\left(- \frac{(\tilde{w}_{T,k} - \tilde{w}_{t,j})^2}{2(T-t)} \right) \tag{2.3.16}$$

With the relation between processes Y_t and $\tilde{W}_t$: $Y_t = \tilde{W}_t - \sigma t$, the equation (2.3.16)

then becomes:

$$p(y_{t,j}, t; y_{T,k}, T) = \frac{1}{\sqrt{2\pi(T-t)}} \exp\left(- \frac{(y_{T,k} - y_{t,j} + \sigma T - \sigma t)^2}{2(T-t)} \right) \tag{2.3.17}$$

In case of American put option, the value of the option at time t_i is the maximum of the immediate exercise value and continuation value.

$$P(w, t_i) = \max((K - S_{t_i}(w))_+, \mathbf{E}_{\mathbb{Q}}[e^{-r(t_{i+1}-t_i)} P(w; t_{i+1}) | \mathcal{F}_{t_i}]) \tag{2.3.18}$$

where $\mathcal{F}$ is the filtration $\{\mathcal{F}_{t_i}\}_{0 \le t \le T}$.

At time t_i and state y_j, the value of American put option can be expressed as:

$$P(t_i; y_j) = \max((K - S_{t_i})_+, P^c(t_i; y_j)) \tag{2.3.19}$$

$P^c(t_i; y_j)$ is the continuation value in time t_i and state y_j. Its value can be expressed as follows.

$$P^c(t_i; y_j) = \frac{e^{-r(t_{i+1}-t_i)}}{\sqrt{2\pi(t_{i+1}-t_i)}} \int_{-\infty}^{\infty} P(t_{i+1}; y) e^{-\frac{(y - y_j + \sigma(t_{i+1}-t_i))^2}{2(t_{i+1}-t_i)}} dy \tag{2.3.20}$$

In order to compute the continuation value by taking the expectation of the remaining discounted cash flows under the risk measure $\mathbb{Q}$, we need numerical fast integration algorithms. In this thesis, we use the Trapezoidal Rule Integration.

Chapter 3

Numerical Implementations

In this chapter we illustrate the implementations of three numerical methods used in this thesis. The programming language used is Matlab.

3.1 Implementation of Finite Difference Scheme

For finite difference method in this thesis, we use implicit scheme. It is proven that implicit scheme is stable and consistent.

1. Consistency

Assume that $U(x,\tau)$ is the theoretical exact solution of the PDE, and $U(ih,nk)$ is the exact solution of the difference equation (h and k are step sizes). Then the truncation error reads:

$$T(ih,nk) = U(x,\tau) - U(ih,nk)$$

A numerical scheme is said to be consistent if:

$$T(ih,nk) \to 0 \text{ as } h,k \to 0$$

In words, a finite difference scheme is consistent if when we decrease the space and time step size, the truncation error approaches to zero. It means that the solution to the difference equation would approach the exact solution of the PDE.

2. Stability

Assume that U_i^n is the theoretical exact solution of the difference equation, and $\tilde{U}_i^n$ is the numerical solution of the difference equation. The difference scheme is stable if the difference

$$U_i^n - \tilde{U}_i^n < K = const$$

In words, a numerical scheme is said to be stable if the difference between the numerical solution and the theoretical exact solution remains bounded as the number of steps tends to infinity.

3. Convergence

According to Lax Equivalence Theorem, for a linear scheme, we have:

$$\text{Stability} + \text{Consistency} = \text{Convergence}$$

In order to evaluate how good a numerical method is, we need to analyze the convergence property of that method. However, it is difficult to prove convergence directly. Instead, we can prove stability and consistency as the Lax Equivalence Theorem says. Implicit finite scheme is stable and consistent, therefore convergent. With this property, we can say that it is reasonable to use implicit scheme in option pricing, in comparison with other numerical methods.

In order to price American option using implicit scheme, we create the Matlab function "*FD.m*". This function will return the price of an American put option and the computation time. This computation time will be used for purpose of comparison between numerical methods. In the code for implementing implicit scheme to price American option, there are some points needed to clarify. First of all, this is the choice of $Smax$. In the code, I choose $Smax = 2\max(S_0, K)e^{rT}$. However, we can choose any number for $Smax$, as long as it is reasonably greater than S_0. Another point is the interpolation vector Z, which is highlighted by the circle in Figure 3.1.

Figure 3.1: Implementation of Implicit Scheme (Part 1)

```
function [FD_price,time_FD] = FD(S0,M,N,T,K,sigma,r,q)
%% Inputs:
%   M: number of grid points per time level
%   N: number of time levels
%   T: time to maturity
%   K: strike price
%   sigma: volatility
%   r: interest rate;
%   q: dividend rate;
%   S0: initial stock price

tic; % Start counting computation time
h=r-q; %carry costs
Smax = 2*max(S0,K)*exp(r*T); % Maximum price considered;

%% Definition of variables
u1=zeros(M+1,1);

%% Defining the grid
hh=Smax./M; % Space step
kk=T./N; % Time step
rx=kk./hh^2;
% Initial values
S=(0:M)'.*hh;
u0=max(K-S,0); % Boundary condition: Derivative at the maturity

%% Now find points either side of the initial price,
%% so that we can calculate the price of the option via interpolation
idx = find(S < S0); idx = idx(end); a = S0-S(idx); b = S(idx+1)-S0;
Z = 1/(a+b)*[a b]; % Interpolation vector

%% Time independent coefficients
aa=0.5.*sigma^2.*(0:M)'.*(0:M)'.*hh.*hh;
bb=h.*(0:M)'.*hh;
cc=-r;
d1=-aa.*rx+0.5.*bb.*rx.*hh;
d2=2.*aa.*rx-cc.*kk+1;
d3=-aa.*rx-0.5.*bb.*rx.*hh;

%% Correction of d1 due to the right boundary at Mx (S=Smax)
d1(M+1)=d1(M+1)+d3(M+1);

%% Construction of the tridiagonal matrix A
A=diag(d1(3:M+1),-1)+diag(d2(2:M+1),0)+diag(d3(2:M),1);

% Step 1: LR-Decomposition of the system Au=d into Au=LRu=Ly=d
[L,R]=lu(A);
```

Figure 3.2: Implementation of Implicit Scheme (Part 2)

```
for nn=1:N %main loop, where nn is counted on the advanced time level

  % Left boundary for index i=0 "Dirichlet"
  u1(1)=K.*exp(-r.*nn.*kk);

  % Definition of the vector d4 (RHS of equation system)
  d4=u0;
  d4(2)=d4(2)-d1(2).*u1(1);

  % Step 2 and step 3 in one line
  u1(2:M+1)=R\(L\d4(2:M+1));

  % Timeshift
  u0=max(u1,K-S);

end

%% Extract the final price
FD_price = Z*u0(idx:idx+1);

time_FD=toc; % Finish counting computation time
end
```

At each discrete stock price in the range $[0, Smax]$, the function "$FD.m$" will return a corresponding American put option price. Therefore, when the program calculates backward to time $t_0 = 0$, we have a vector of American put prices. However, the final result that we need is just a single price for American option. Therefore, we use linear interpolation technique to extract the final single price. The choices of M (number of grid points) and N (number of time levels) are important in finite difference scheme. Bigger M and N increase the accuracy of approximation algorithm. However, there is a threshold where increasing M and N do not improve the preciseness of the final result. Instead, it will cost us the computation time and computer resources to run our function.

3.2 Implementation of Least Squares Monte Carlo Method

As in Chapter 2, we examine a specific example of using least squares Monte Carlo method to price American option. In this section, we elaborate how to implement this method by Matlab. Valuing a derivative security by Monte Carlo typically involves simulating paths of stochastic processes used to describe the evolution of the underlying asset prices. The main idea is to represent derivative prices as expectations of random objects that can be simulated. The Matlab function "*LSM.m*" illustrates the Least-squares Monte Carlo method. In the context of pricing American options, we implement the following steps:

1. Generate M stock price paths. Under the Black-Scholes framework, the underlying asset price is assumed to follow a stochastic differential equation (SDE):

$$dS(t) = rS(t)dt + \sigma S(t)dW(t) \tag{3.2.1}$$

where W is the geometric Brownian motion (Hull p.282).
The solution of the stochastic differential equation is:

$$S(T) = S_0\, exp\,([r - \tfrac{1}{2}\sigma^2]T + \sigma W(T)), \tag{3.2.2}$$

where σ is the volatility of the stock price, r - the mean rate of return, S_0 - the current price of the stock.
For the simulation purpose, we used step-by-step approach/Euler scheme:

$$S_{t+\Delta t} = S_t\, exp\,((r - \tfrac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\,\epsilon), \tag{3.2.3}$$

where $\epsilon \sim N(0,1)$.
An another important parameter in stock price simulation is the number of time steps, which is the variable N in the function "*LSM.m*". Therefore, our first task is to simulate M stock price paths, and with each path, there is N time steps. As a part of this simulation, we draw $N \times M$ normally distributed random numbers. We use here two index i and j: $1 \leq i \leq N$, $1 \leq j \leq M$ (i: index for time, j: index for stock price path).

Figure 3.3: Implementation of LSM method (Part 1)

```
function [LSM_price,time_LSM] = LSM(S0,K,r,T,sigma,N,M)
%% Inputs:
%  S0     Initial asset price
%  K      Strike Price
%  r      Interest rate
%  T      Time to maturity of option
%  sigma  Volatility of underlying asset
%  N      Number of points in time grid
%  M      Number of paths

tic; %Start counting computation time
dt = T/N;  % Time steps

%% Simulate stock prices
R = exp((r-sigma^2/2)*dt+sigma*sqrt(dt)*randn(N,M));
S = cumprod([S0*ones(1,M); R]);
S=S';
```

2. Choose the basic regression function F(x). In this thesis, the chosen regression function has the following form:

$$F(x) = p_1 + p_2(1-x) + \frac{1}{2}p_3(2-4x-x^2) \tag{3.2.4}$$

where p_1, p_2, p_3 are the parameters we need to estimate using least squares technique.

3. At the maturity of the option $t = T$, we calculate the cash flows $CF^j(t_N)$ which for an American put are $\max(K - S^j_{t_N}, 0)$ for each stock price path.

4. We move back to $t = t_{N-1}$, for each in-the-money path (where $S^j(t_N) < K$), we calculate the continuation value:

$$CV^j(t_{N-1}) = e^{-r\Delta t} CF^j(t_N)$$

where $\Delta t = \dfrac{T}{N} = t_N - t_{N-1}$

Figure 3.4: Implementation of LSM method (Part 2)

```
%% Main algorithm to value American options
CF = zeros(M,N+1); % Cash flow matrix
CF(:,end) = max(K-S(:,end),0); % Payoff at time T
for i = N:-1:2
   Idx = find(K>S(:,i)); % Find paths that are in the money at time i
   X = S(Idx,i); % Prices for in-the-money paths
   Y = CF(Idx,i+1).*exp(-r*dt); % Discounted cashflow

   R = [ones(size(X)) (1-X) 1/2*(2-4*X-X.^2)];
   beta = R\Y; % Linear regression step
   C = R*beta; % Estimated value of continuation

   E=max(K-X,0) ;    % Value of immediate excercise
   exP=Idx(E>C); % Paths where it's better to excercise
   nexP = setdiff((1:M),exP); % Rest of the paths
   CF(exP,i) = max(K-X(E>C),0); % Better to excercise, insert value in payoff vector
   CF(nexP,i) = CF(nexP,i+1).*exp(-r*dt); % Better to continue,
                                          % insert previous payoff and discount back one step
end

LSM_price = mean(CF(:,2))*exp(-r*dt);
time_LSM=toc; % Finish counting computation time
end
```

5. Implement the regression to identify the functional form of the continuation value y(S). In other words, we determine the values of three parameters p_1, p_2, p_3 in Equation (3.2.4).

6. Recalculate the continuation values using function y(S):

$$CV^j(t_{N-1}) = y(S^j_{t_{N-1}})$$

7. Identify paths where it is better to exercise the option. For paths satisfying the condition: $CV^j(t_{N-1}) < K - S^j_{t_{N-1}}$, we calculate the cash flow values $CF^j_{t_{N-1}} = K - S^j_{t_{N-1}}$ and $CF^j(t_i) = 0$ for $i > N-1$. For paths not satisfying this condition, $CF^j_{t_{N-1}} = 0$.

8. We repeat these steps when going backwards in time until we have $CF^j(t_i)$ for all i and j. Note that we calculate the continuation value $CV^j(t_i)$ before the regression to avoid upward bias in the final value of the option.

9. The option value V_0 is given by

$$V_0 = \frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N} e^{-rt_i} CF^j(t_i) \tag{3.2.5}$$

3.3 Implementation of Grid Lattice Algorithm

In this section, we build Matlab functions to price American and European put options using grid lattice algorithm. Those functions are "$GL\_AM.m$", which is used to price American put option, and "$GL\_EU.m$", which is used to price European put option. We can see that European option is a special case of American option when the option is not early exercised at any point of time. We can build a Matlab function for American option and then with a little adjustment, we can create a function for European option. Therefore, the following part will focus on explaining the procedure of lattice algorithm to value American put option.

1. **Step 1:** First of all, we need to create the lattice grid. We discretize the time dimension of the grid into N time steps. At each time step, there are M grid nodes. The grid space between two consecutive nodes is: $\delta_{yt} = y_{t,j} - y_{t,j-1}$ for $j = 1, ..., M_t$. For simplification purpose, we choose that all nodes are equally spaced and symmetric. We set that $\delta_y = \dfrac{2A}{M-1}$ and $y_{t,j} = -A + j\delta_y$ with j=1,2,...,M. It is easy to check that the largest distance between two nodes is 2A and all the grid nodes cover the interval $[-A, A]$. Refer to Figure 2.3 in Chapter 2 to easily visualize the lattice grid. With the established grid lattice, we compute the stock prices corresponding to each time point and each grid point according to the formula:

$$S_{t,j} = S_0 \exp(rt + \frac{\sigma^2}{2}t + \sigma y_{t,j}) \tag{3.3.1}$$

   We will have a matrix of stock prices with N columns and M rows.

2. **Step 2:** We calculate the value of put option at the maturity of the option t=T. At maturity, American option and European option have the same payoff, which is given by

$$P(T, y_j) = \max(K - S_T, 0) = (K - S_T)^+ \tag{3.3.2}$$

where $j = 1, ..., M$

Figure 3.5: Implementation of Grid Lattice Algorithm (Part 1)

```
function [P_Am0,time_lattice]= GL_AM(S0,M,N,T,K,sigma,r,dy)
%% Inputs:
%  S0: initial stock price
%  M: number of grid points per time level
%  N: number of time levels
%  T: time to maturity
%  K: strike price
%  sigma: volatility
%  r: interest rate
%  dy: grid space
tic; % Start counting computing time
%% Step 1: Define variables
dt = T/N;  % Time steps
t=dt:dt:T;
A=dy*(M-1)/2;
y=(-A:dy:A)';
S=zeros(M,N);
P_Am=zeros(M,N);
P_c_Am=zeros(M,N-1);

for i=1:N
   S(:,i)=S0.*exp(r*t(i)+sigma.^2/2.*t(i)+sigma.*y);
end

%% Step 2: Payoff at the maturity T
P_Am(:,end) = max(K-S(:,end),0); % Payoff at time T
```

3. **Step 3:** We go backwards to the time point $t = t_{N-1}$. At any state y_j, the value of American put option P_{Am} is the maximum between the early exercised value and continuation value $P^c(t_{N-1}, y_j)$.

$$P_{Am}(t_{N-1}, y_j) = \max\{(K - S_{t_{N-1}})^+, P^c(t_{N-1}, y_j)\} \tag{3.3.3}$$

Basing on the Trapezoidal Rule Integration and the transition probability density in equation (2.3.17), we can calculate the continuation value using the following formula:

$$P^c(t_{N-1}, y_j) = \frac{e^{-r\Delta t}}{\sqrt{2\pi\Delta t}} \int_{-\infty}^{\infty} P(T, y) \exp\left(- \frac{(y - y_j + \sigma\Delta t)^2}{2\Delta t} \right) dy \quad (3.3.4)$$

$$P^c(t_{N-1}, y_j) \approx \frac{e^{-r\Delta t}}{\sqrt{2\pi\Delta t}} \frac{\delta_y}{2} \sum_{k=2}^{M} \left\{ P(T, y_k) \exp(-\frac{(y_k - y_j + \sigma\Delta t)^2}{2\Delta t}) + P(T, y_{k-1}) \exp\left(- \frac{(y_{k-1} - y_j + \sigma\Delta t)^2}{2\Delta t} \right) \right\}$$

4. **Step 4:** In a similar manner, we find the option value at earlier nodes. Suppose we want to value the option at date t_i. The value of American put option at time t_i in any state y_j can be expressed by

$$P_{Am}(t_i, y_j) = \max\{(K - S_{t_i})^+, P^c(t_i, y_j)\} \quad (3.3.5)$$

where $P^c(t_i, y_j)$ is the discounted continuation value in state y_j at time t_i. Again, based on the transition probability density in equation (2.3.17) and the Trapezoidal Rule Integration, the value of $P^c(t_i, y_j)$ is

$$P^c(t_i, y_j) = \frac{e^{-r\Delta t}}{\sqrt{2\pi\Delta t}} \int_{-\infty}^{\infty} P_{Am}(t_{i+1}, y) \exp\left(- \frac{(y - y_j + \sigma\Delta t)^2}{2\Delta t} \right) dy \quad (3.3.6)$$

$$P^c(t_i, y_j) \approx \frac{e^{-r\Delta t}}{\sqrt{2\pi\Delta t}} \frac{\delta_y}{2} \sum_{k=2}^{M} \left\{ P_{Am}(t_{i+1}, y_k) \exp(-\frac{(y_k - y_j + \sigma\Delta t)^2}{2\Delta t}) + P_{Am}(t_{i+1}, y_{k-1}) \exp\left(- \frac{(y_{k-1} - y_j + \sigma\Delta t)^2}{2\Delta t} \right) \right\}$$

When going backward in time i=N-2, ...,1, we can implement the same procedure to calculate the discounted continuation values at each date. When we implement this procedure of valuing American option using grid lattice algorithm by Matlab, we can combine Step 3 and Step 4.

Figure 3.6: Implementation of Grid Lattice Algorithm (Part 2)

```
%% Step 3 and 4: Find the option value at time nodes N-1:-1:1
for i=N-1:-1:1 % time space
 for j=1:M % y space

  % Calculations for American option
  P_c_Am(j,i)=exp(-r.*dt)./sqrt(2.*pi.*dt).*dy./2....
  .*sum(  P_Am(2:M,i+1).*exp(-(y(2:M)-y(j)+sigma.*dt).^2./(2*dt)) ...
  +P_Am(1:M-1,i+1).*exp(-(y(1:M-1)-y(j)+sigma.*dt).^2./(2*dt))    );

  P_Am(j,i)=max(K-S(j,i),P_c_Am(j,i));

 end
end

%% Step 5: Option value at time 0
Pc0_Am=exp(-r.*dt)./sqrt(2.*pi.*dt).*dy./2....
  .*sum(  P_Am(2:M,1).*exp(-(y(2:M)+sigma.*dt).^2./(2*dt)) ...
  +P_Am(1:M-1,1).*exp(-(y(1:M-1)+sigma.*dt).^2./(2*dt))    );

P_Am0=max(K-S0,Pc0_Am);

time_lattice=toc; % Finish counting computation time

end
```

5. **Step 5:** Compute the final option price at time $t_0 = 0$. The value of American option at time $t_0 = 0$ can be given as

$$P_{Am}(0) = \max\{K - S_0, P^c(0)\} \tag{3.3.7}$$

The formula for $P^c(0)$ is as follows

$$P^c(0) = \frac{e^{-r\Delta t}}{\sqrt{2\pi\Delta t}} \int_{-\infty}^{\infty} P_{Am}(t_1, y) \exp\left(- \frac{(y + \sigma\Delta t)^2}{2\Delta t} \right) dy \tag{3.3.8}$$

$$P^c(0) \approx \frac{e^{-r\Delta t}}{\sqrt{2\pi\Delta t}} \frac{\delta_y}{2} \sum_{k=2}^{M} \left\{ P_{Am}(t_1, y_k) \exp(-\frac{(y_k + \sigma\Delta t)^2}{2\Delta t}) + P_{Am}(t_1, y_{k-1}) \exp\left(- \frac{(y_{k-1} + \sigma\Delta t)^2}{2\Delta t} \right) \right\}$$

As mentioned before, European option can be seen as a special case of American option when there is no early exercise at any point of time. Therefore, we build the Matlab function for European option basing on the function for American option. In particular, for European option, we replace the equations (3.3.3), (3.3.5). (3.3.7) by

$$P_{Eu}(t_{N-1}, y_j) = P^c(t_{N-1}, y_j) \tag{3.3.9}$$

$$P_{Eu}(t_i, y_j) = P^c(t_i, y_j) \tag{3.3.10}$$

$$P_{Eu}(0) = P^c(0) \tag{3.3.11}$$

respectively and replace P_{Am} by P_{Eu} in equations (3.3.6) and (3.3.8)

3.4 Implementation of the Main Program

In this paper, we focus on the grid lattice method to price American option. We want to compare the efficiency of this method with other two methods: least squares Monte Carlo simulation and finite different method. The efficiency can be compared by two criteria: the accuracy and the computation time. We also extend our analysis on pricing European options using grid lattice method.

3.4.1 Test Design for European Options

We can price European options using Black-Scholes closed-form solution. Suppose we need to value an European put option. Some variables involving in pricing the option are as follows.

S: Spot price of the underlying asset

K: Strike price

r: Risk-free interest rate

σ: Volatility of the underlying asset

T: Time to maturity.

We define N as the cumulative distribution function (CDF) of a standard normal

distribution.

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{-z^2}{2}} dz$$

The Matlab function "*N.m*" returns the value of $N(x)$. According to Black-Scholes model, the price of an European put option can be formulated as below

$$P(S_t, t) = N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t$$

where

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_2 &= d_1 - \sigma\sqrt{T-t} \\ &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)(T-t)\right] \end{aligned}$$

The Matlab function "*BSMP.m*" computes the value of an European put option by Black-Scholes model. It is obvious that using this closed-form solution produces

Figure 3.7: Implementation of Black-Scholes Model

```
function Put = BSMP(S0,K,T,t,q,r,sigma)
d1 = (log(S0./K)+(r-q+0.5.*sigma.*sigma).*(T-t))/(sigma.*sqrt((T-t)));
d2 = d1 - sigma.*sqrt((T-t));

Put = -S0.*exp(-q.*(T-t)).*N(-d1) + K.*exp(-r.*(T-t)).*N(-d2);
end
```

the most accurate value for European option. Moreover, because of the simplicity of the method, Matlab function "*BSMP.m*" can return the result in the shortest time. Therefore, there is no need to compare the computation time between BSM and other numerical methods. We will use BSM price as the true value of European option. In order to illustrate the accuracy of grid lattice method, we compare the result by the lattice algorithm with that from Black-Scholes model. The criteria we use is the absolute difference between two prices generated by two methods respectively.

$$Diff = \frac{|P_{BS} - P_{lat}|}{P_{BS}} \times 100\% \tag{3.4.12}$$

where

P_{BS}: Black-Scholes price (true value)

P_{lat}: option prices valued by lattice algorithm

The following figure illustrates the Matlab function "$COMPARE\_EU.m$", which contains all the computations we need for the European option test.

Figure 3.8: Implementation of European Option Test

```
function [BS,P_Eu_lattice,diff] = COMPARE_EU(S0,K,T,t,q,r,sigma,M,N,dy)
BS = BSMP(S0,K,T,t,q,r,sigma);
P_Eu_lattice=GL_EU(S0,M,N,T,K,sigma,r,dy);

diff=abs(BS-P_Eu_lattice)/BS*100;
end
```

3.4.2 Test Design for American Options

For American options, there exists no closed-form solution. Therefore, we do not have true value to use as a threshold when we compare the efficiency of three methods: finite difference method (FD), least-squares Monte Carlo technique (LSM), and grid lattice method (lat). For the accuracy evaluation, we regard the finite difference method as the benchmark. We compute the difference in price valued by the finite difference method P_{FD} and that by the least squares Monte Carlo simulation P_{LSM}.

$$Diff_{LSM} = \frac{|P_{FD} - P_{LSM}|}{P_{FD}} \times 100\% \tag{3.4.13}$$

Similarly, we compare the price produced by the finite difference method P_{FD} and that by the grid lattice algorithm P_{lat}.

$$Diff_{lat} = \frac{|P_{FD} - P_{lat}|}{P_{FD}} \times 100\% \tag{3.4.14}$$

For American options, there is a need to compare the computation time or running time among three methods. This can be done by using the stopwatch timer function 'tic' and 'toc' in Matlab.

Figure 3.9: Implementation of American Option Test

```
function
[FDprice,time_FD,LSMprice,time_LSM,diff_FD_LSM,P_Am_lattice,time_lattice,diff_FD_lattice] =
COMPARE_AM(S0,M_FD,N_FD,T,K,sigma,r,q,N_LSM,M_LSM,M_lattice,N_lattice,dy)
%% Compute American put by finite difference method
[FDprice,time_FD]=FD(S0,M_FD,N_FD,T,K,sigma,r,q);

%% Compute American put by least square Monte-Carlo method
[LSMprice,time_LSM]=LSM(S0,K,r,T,sigma,N_LSM,M_LSM);
diff_FD_LSM=abs(FDprice-LSMprice)/FDprice*100;

%% Compute American put by grid lattice method
[P_Am_lattice,time_lattice]=GL_AM(S0,M_lattice,N_lattice,T,K,sigma,r,dy);
diff_FD_lattice=abs(FDprice-P_Am_lattice)/FDprice*100;

end
```

Besides all the Matlab functions that we examine above, we have two important Matlab files: "$ACCURACY\_EU.m$" and "$ACCURACY\_AM.m$". These two Matlab files will use specific data sets together with other Matlab functions to analyze the efficiency of three numerical methods.

The following table summarizes all the Matlab files and their purposes in this thesis.

	Matlab files	**Purpose**
1	FD.m	Value American put option by finite difference method
2	LSM.m	Value American put option by least squares Monte Carlo method
3	GL_EU	Value European put option by grid lattice algorithm
4	GL_AM	Value American put option by grid lattice algorithm
5	N.m	Cumulative distribution function of a standard normal distribution
6	BSMP.m	Value American put option by Black-Scholes model
7	COMPARE_EU.m	Evaluate the accuracy of grid lattice method in pricing European option
8	COMPARE_AM.m	Compare the efficiency of three numerical methods used to price American options
9	ACCURACY_EU.m	Main file with data sets for European option
10	ACCURACY_AM.m	Main file with data sets for American option

Table 3.1: Summary of Matlab files

Chapter 4

Research Results and Discussion

In Chapter 3, we build some Matlab functions to evaluate the efficiency of three numerical methods. In this chapter, we will use some specific data sets to produce some results. Basing on these results, we can analyze the effectiveness of grid lattice algorithm in pricing American option as well as European option. Running Matlab programs can take a long time, due to big data.

4.1 Efficiency Test for European Options

In order to illustrate the convergence of the grid lattice algorithm, we choose a specific data set for European option. Table 4.1 below summarizes the input data.

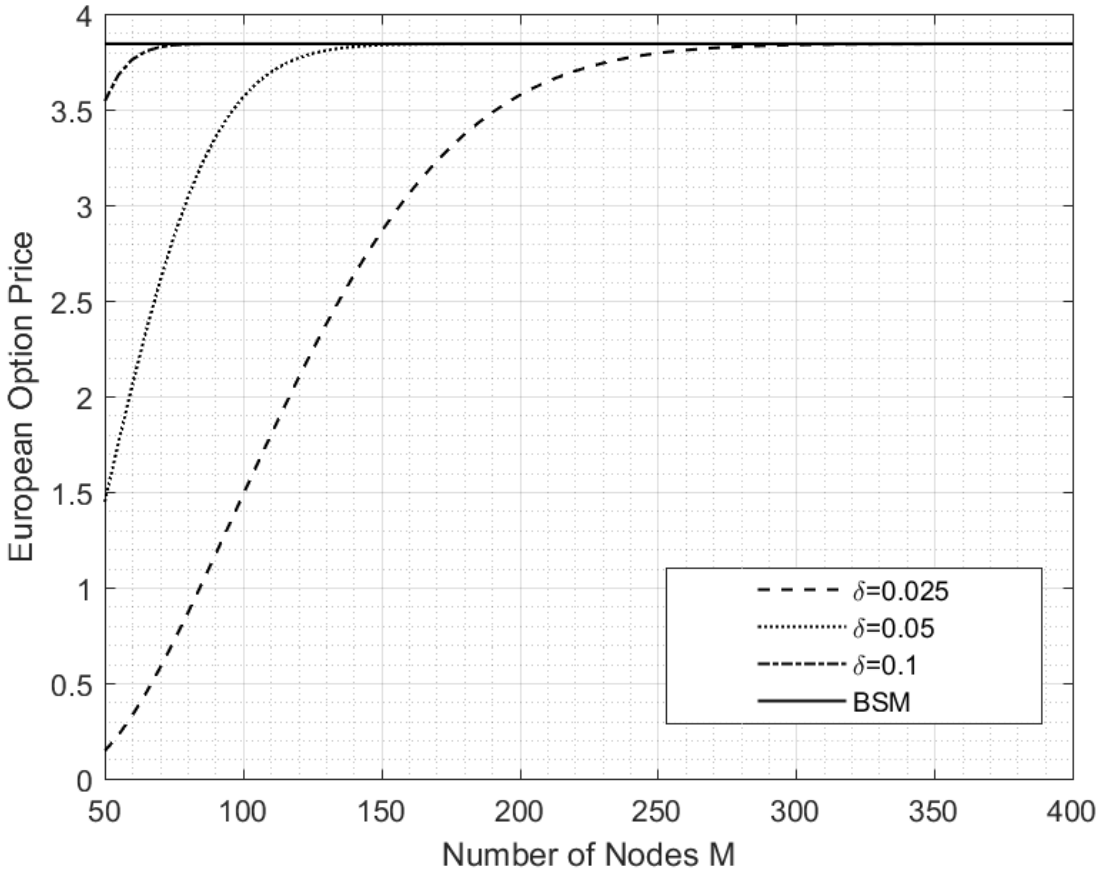
Option	S_0	**K**	**T**	**r**	**q**	σ
European Put	36	40	1	0.06	0	0.20

Table 4.1: Input data for a European put option

In grid lattice method, there is three important parameters: the number of determination dates N, the number of grid nodes M, and the grid space δ. In our analysis, we consider three small tests.

1. Test 1: We analyze the convergence of the lattice algorithm when keeping the number of determination dates constant ($N = 50$) and vary the number of grid nodes M. The number of nodes ranges from 50 to 400 in step of 5: $M \in [50 : 5 : 400]$. Besides, we consider three cases of grid space: $\delta_y = [0.025; 0.05; 0.1]$.

Figure 4.1: Convergence of a European put option using different grid spaces in the lattice algorithm



With the data in Table 4.1, the calculated BSM price is 3.8443. From Figure 4.1, we can see that the European option prices in three cases of grid space converge to the BSM price 3.8443. However, the rate of convergence is different. The higher the grid space, the faster the price converges to the true value. In particular, the dash-dot line ($\delta = 0.1$) meets the BSM line earlier than the dotted line ($\delta = 0.05$)

and the dashed line ($\delta = 0.025$).

Let have a look at the Table 4.2 to see at which number of nodes the price valued by the lattice algorithm converges to the true price. Refer to the Appendix A.1 for a full table of European prices in three cases of grid space using data in Table 4.1. From Table 4.2, we see that the European put option price converges to true value when the lattice covers approximately a certain number of standard deviations, namely $\delta \times (M-1) \approx 10$. Remember that $\delta \times (M-1) = 2A$, therefore the value of A should be large enough (at least 5 in this case) to ensure the price convergence in grid lattice algorithm.

Number of nodes M	**Grid space δ**	**Price**
> 100	0.1	3.8444
> 200	0.05	3.8446
> 375	0.025	3.8441

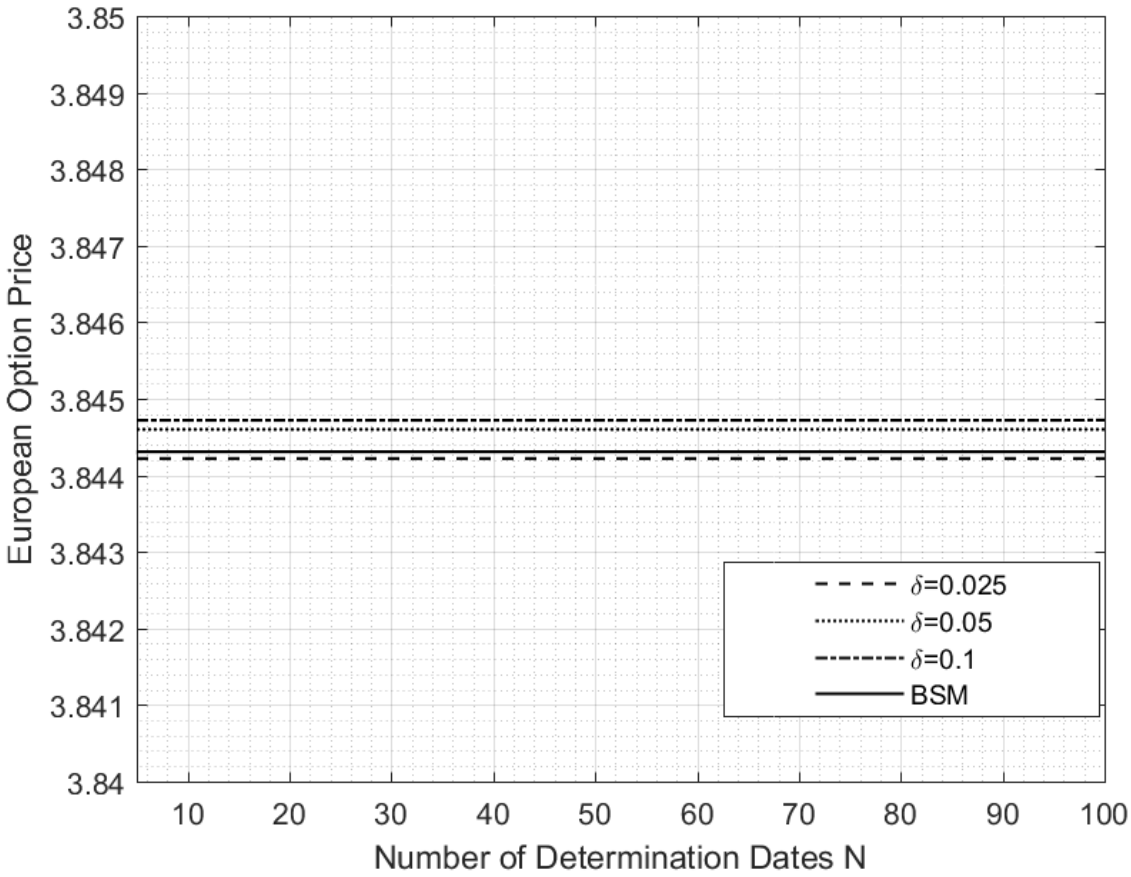
Table 4.2: Convergence result of a European put option using different grid spaces in the lattice algorithm

2. Test 2: We analyze the convergence of a European put option when we fix the number of grid nodes M=501 and vary the number of determination dates from 5 to 100 in step of 5: $N \in [5:5:100]$. We also keep three cases of the grid space.

From Figure 4.2, we can see that the prices of European option in three cases of grid space remain constant over the number of determination dates. It is reasonable because European option does not allow early exercise before the maturity. Another feature from this figure is that all three prices corresponding to three grid spaces are close to the true price BSM, although the lattice with a higher grid space tends to bring a higher option price. The full results of this test is presented in the Appendix

A.2.

Figure 4.2: Relationship between European put option price and number of determination dates using different grid spaces in the lattice algorithm



3. Test 3: We analyze the accuracy of grid lattice algorithm under various settings. Considering the convergence rate (refer to Table 4.2), we use 151 nodes with the grid space of 0.1. Other settings are summarized in the Table 4.3. The results are presented in Table 4.4. This table represents the results from running Matlab program for 20 different settings of parameters. Note that in the table, prices are rounded to the nearest ten-thousandths. However, the differences are calculated using the precise values of these prices.

Parameters	Value
S_0	[32; 36; 40; 44; 48]
σ	[0.1-0.9]
T	[1;2]
N	[50; 100]
M	151
K	40
r	0.06
q	0
δ	0.1

Table 4.3: Various settings to compare the lattice algorithm and BSM in pricing European put option

In general, the differences between the BSM prices and the corresponding prices calculated by the lattice algorithm in all scenarios are very small. For some cases, there is almost no difference between these prices. When the European put options are in the money at time zero (for $S = 32$ and $S = 36$), the grid lattice algorithm works quite well, because the differences in these cases are really low. The maximum different rate is only 0.021 % in the fourth scenario ($S_0 = 32$, $\sigma = 0.5$, $T = 2$, $N = 100$). The differences tends to increase when the European put options are at the money or out of money at time zero. However, the maximum difference is still low, which equals to 0.067% for the scenario $S = 44$, $\sigma = 0.8$, $T = 1$, and $N = 50$. In overall, the grid lattice algorithm is a good candidate in pricing European options.

S_0	σ	T	N	BS	Lattice	Difference
32	0.1	1	50	5.7454	5.7454	0.000 %
32	0.1	2	100	4.1235	4.1237	0.004 %
32	0.5	1	50	10.0666	10.0671	0.005 %
32	0.5	2	100	11.1561	11.1537	0.021 %
36	0.2	1	50	3.8443	3.8447	0.011 %
36	0.2	2	100	3.7630	3.7633	0.008 %
36	0.6	1	50	9.5460	9.5467	0.008 %
36	0.6	2	100	11.4847	11.4851	0.004 %
40	0.3	1	50	3.5574	3.5593	0.053 %
40	0.3	2	100	4.3261	4.3236	0.058 %
40	0.7	1	50	9.5004	9.5036	0.034 %
40	0.7	2	100	12.1523	12.1535	0.010 %
44	0.4	1	50	3.7828	3.7810	0.049 %
44	0.4	2	100	5.2020	5.2034	0.026 %
44	0.8	1	50	9.7541	9.7476	0.067 %
44	0.8	2	100	13.0367	13.0384	0.013 %
48	0.5	1	50	4.2397	4.2419	0.051 %
48	0.5	2	100	6.2583	6.2552	0.049 %
48	0.9	1	50	10.2093	10.2085	0.008 %
48	0.9	2	100	14.0657	14.0683	0.019 %

Table 4.4: Comparison in pricing European put options between Black-Scholes formula and lattice algorithm

4.2 Efficiency Test for American Options

To analyze the efficiency of grid lattice algorithm, we need to compare the accuracy and the computation time with those of the other two numerical methods (least squares Monte Carlo simulation and finite difference method). Due to the fact that there is no closed-form solution for American option, we use the results from finite difference method as the benchmark. First of all, we need to learn the convergence of the lattice algorithm in pricing American option. The data for American put option is in Table 4.5. Similar to analyze the grid lattice algorithm in pricing European option, we need to do some small tests.

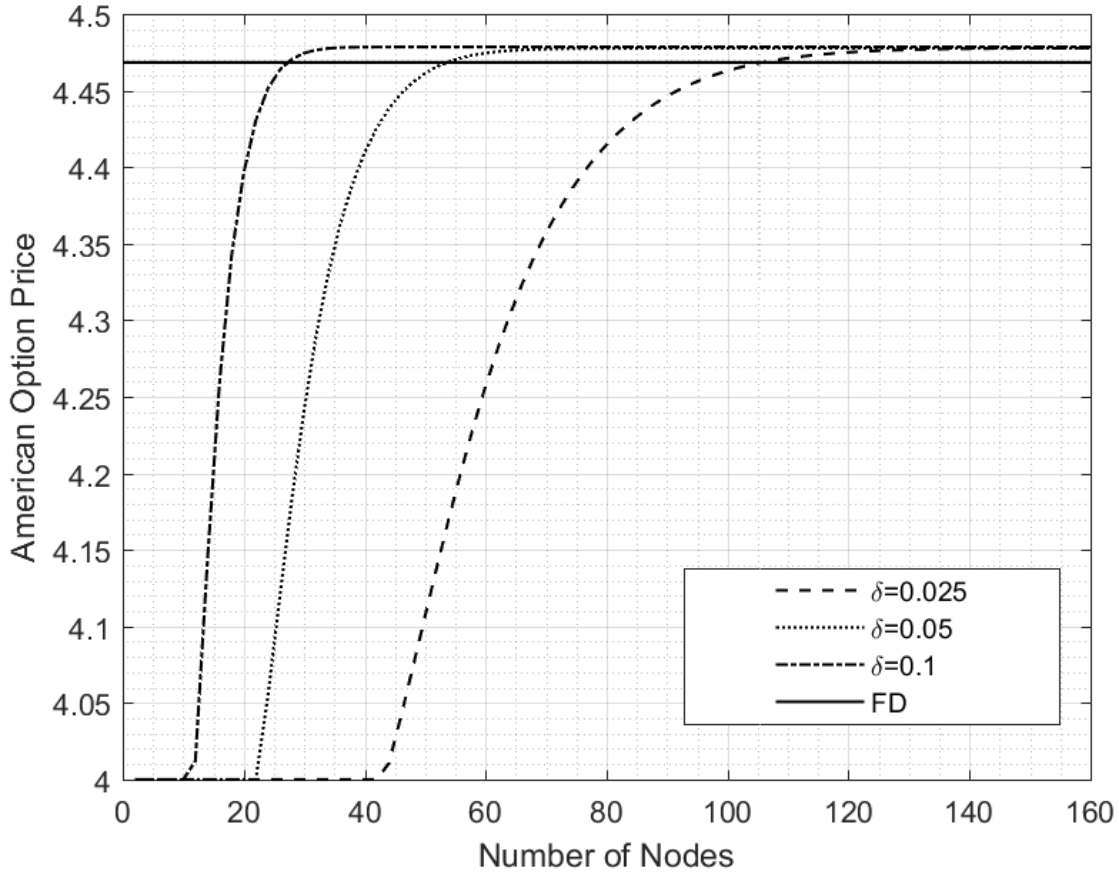
Option	S_0	**K**	**T**	**r**	**q**	σ
American Put	36	40	1	0.06	0	0.20

Table 4.5: Input data for an American put option

1. Test 1: We examine the convergence of the grid lattice algorithm by fixing the number of determination dates (N=50) and vary the number of grid nodes: M $\in [2 : 2 : 160]$. We consider three cases of grid space: $\delta_y = [0.025; 0.05; 0.1]$. For finite difference method, we use 500 time levels ($N_{FD} = 500$) and 1400 grid points per time level ($M_{FD} = 1400$). For least squares Monte Carlo method, we simulate 100000 paths of stock prices ($M_{LSM} = 100000$) with 50 grid points in each path ($N_{LSM} = 50$). The prices of American put option computed by finite difference method and least square Monte Carlo simulation are 4.4684 and 4.4788 respectively.

In Figure 4.3, we see that three lines corresponding to three grid spaces converge when the number of grid nodes increases. The convergence prices in three cases are also close to the price valued by finite difference method (solid line). This implies that the grid lattice algorithm works well in pricing American option. Moreover, the prices calculated by the lattice algorithm converge quite quickly.

Figure 4.3: Convergence result of an American put option using different grid spaces in the lattice algorithm



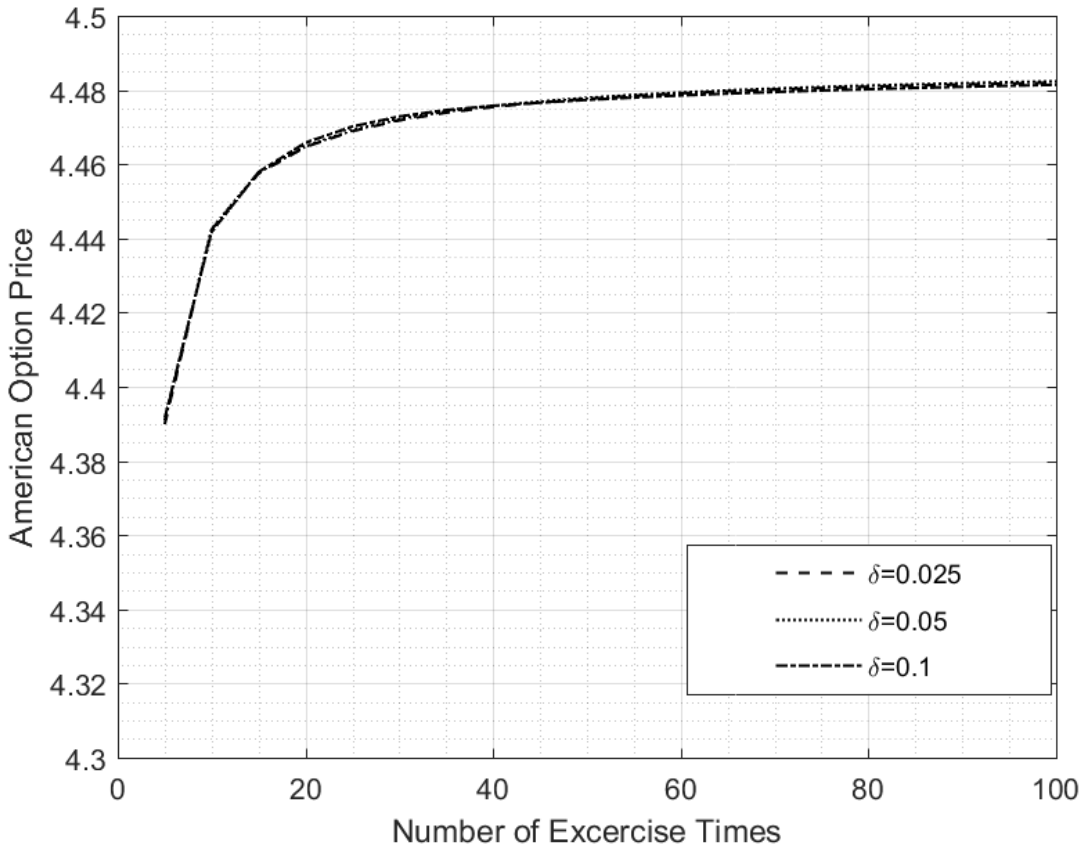
It is obvious that there is correlation between the grid space and the rate of convergence. In particular, the case with lower grid space converges faster than the case with large grid space. From Figure 4.3, we can see that the dash-dot line with $\delta = 0.1$ starts converging when the number of grid nodes reaches approximate 45 nodes. In the meantime, the dotted line ($\delta = 0.05$) and the dashed line ($\delta = 0.025$) converge at a lower speed when the number of nodes M is around 85 and 145 respectively. The full result of this test is presented in the Appendix A.3. Table 4.6 summarizes the number of nodes where the American put price starts converging in three cases of grid space. From Table 4.6, it is observable that the product $\delta \times (M-1)$ in three cases equals to a certain number: $\delta \times (M-1) = 2A \approx 5$. This confirms that the American option prices start converging when the lattice covers approximately five standard deviation.

Number of nodes	Grid space	Price
> 45	0.1	4.4785
> 85	0.05	4.4777
> 145	0.025	4.4777

Table 4.6: Convergence result of an American put option using different grid spaces in the lattice algorithm

2. Test 2: We analyze the relationship between American option price and the number of determination date. In order do so, we fix the number of grid nodes M=151 to ensure the convergence of the lattice algorithm in three cases of different grid space. The number of determination dates varies from 5 to 100 in step of 5: $N \in [5 : 5 : 100]$.

Figure 4.4: Convergence result of an American put option using different grid spaces in the lattice algorithm



The full result of this test can be found in the Appendix A.4. Figure 4.4 illustrates the convergence of the American put price with changing number of determination dates. Regardless of different grid spaces, the American option prices in three cases demonstrate the similar convergence manner. This is easy to understand, because we choose the number of grid nodes M=151 large enough to guarantee the convergence. Initially, when the number of determination dates increases, the price of American put option also increases. However, when the number of determination dates reaches a certain level, around N=50 in this case, the option prices become stable. N=50 means that according to the grid lattice method, it is possible in practice that we can exercise the option 50 times a year.

3. Test 3: We compare the accuracy and computation time in pricing American option among three numerical methods: Finite difference method, least squares Monte Carlo simulation, and grid lattice algorithm.

Parameters	**Value**
S_0	[32; 36; 40; 44; 48]
σ	[0.1-0.9]
T	[1;2]
N	[50; 100]
M	151
K	40
r	0.06
q	0
δ	0.025

Table 4.7: Various settings to compare the lattice algorithm with other two methods in pricing American put option

For finite difference method, we establish a rectangular grid with 1400 grid points per time level (M=1400) and 500 time levels (N=500). Other data for the option

such as the initial stock price, strike price, interest rate, volatility of the underlying stock, and time to maturity is the same as in the lattice algorithm. For least squares Monte Carlo method, we use 100,000 simulations paths, according to Longstaff and Schwartz (2001). Because least squares Monte Carlo method involves simulation of normally distributed random numbers, the option prices computed by LSM method can slightly change each time we run Matlab programs. The computation time is measured in the unit of seconds and it also varies each time we run Matlab program, however the difference is negligible. In addition, the computational time depends on the capacity of the computer used to run the program. However, since we use the same computer to implement three numerical methods, the results are still reliable.

Table 4.8 shows the results of three numerical methods for 20 different data settings. When the option is deep in the money at time zero, both LSM method and grid lattice algorithm do a good job in pricing option. In this case, there is no significant discrepancy in prices valued by LSM, grid lattice algorithm and the benchmark price valued by finite difference method. Particularly, if the initial stock price is less than 40, the maximum difference in price is only approximate 0.7 % for lattice algorithm. LSM and grid lattice algorithm also give the results quite close to the finite difference method when the volatility of the underlying stock is low. This fact is well demonstrated for cases with $\sigma < 0.5$, in which most differences are below 1% for both LSM and the lattice algorithm.

For cases with high volatility, the difference tends to increase as the volatility increases. Besides, in some cases, the grid lattice algorithm depicts a lower difference than that of the LSM method. In other cases, LSM method seems to work better than the grid lattice method. However, in general, the discrepancy in prices among three method is not considerable. In terms of computation time, the lattice algorithm shows the highest efficiency with very low running time. All in all, in practice, if we predetermine a specific accuracy level, the grid lattice algorithm is the best choice because of the lowest running time.

S_0	σ	T	N	FD	Time	LSM	Time	Diff (%)	Lattice	Time	Diff (%)
32	0.1	1	50	7.986	8.098	7.954	0.403	0.403	8.000	0.205	0.170
32	0.1	2	100	8.023	6.464	7.901	1.516	1.516	8.000	0.270	0.282
32	0.5	1	50	10.580	6.957	10.566	0.136	0.136	10.582	0.111	0.013
32	0.5	2	100	12.216	7.050	12.157	0.480	0.480	12.163	0.212	0.427
36	0.2	1	50	4.468	6.677	4.469	0.021	0.021	4.478	0.118	0.207
36	0.2	2	100	4.865	6.965	4.849	0.336	0.336	4.835	0.193	0.612
36	0.4	1	50	7.094	6.588	7.070	0.333	0.333	7.101	0.115	0.104
36	0.4	2	100	8.529	6.868	8.508	0.239	0.239	8.500	0.230	0.341
40	0.3	1	50	3.797	7.234	3.764	0.855	0.855	3.806	0.095	0.252
40	0.3	2	100	4.906	6.567	4.843	1.287	1.287	4.884	0.216	0.448
40	0.7	1	50	9.883	6.657	9.736	1.481	1.481	9.775	0.135	1.086
40	0.7	2	100	13.359	6.354	12.781	4.328	4.328	12.717	0.214	4.806
44	0.4	1	50	3.939	7.568	3.923	0.418	0.418	3.948	0.128	0.212
44	0.4	2	100	5.661	7.542	5.622	0.694	0.694	5.569	0.211	1.638
44	0.8	1	50	10.179	6.667	10.044	1.325	1.325	9.988	0.111	1.875
44	0.8	2	100	14.412	6.746	13.652	5.274	5.274	13.321	0.205	7.573
48	0.5	1	50	4.369	7.656	4.406	0.839	0.839	4.370	0.101	0.025
48	0.5	2	100	6.708	6.617	6.699	0.137	0.137	6.405	0.204	4.514
48	0.9	1	50	10.718	6.781	10.464	2.368	2.368	10.410	0.103	2.870
48	0.9	2	100	15.658	8.125	14.666	6.334	6.3338	13.983	0.225	10.698

Table 4.8: Comparison in pricing American put options among finite difference method, least squares Monte Carlo simulation, and lattice algorithm

Conclusions

In this thesis, we present an efficient grid lattice algorithm to price American option. By building a time adjusted grid lattice and employing backward induction technique, we create an algorithm that returns the option price with high accuracy in fast computational speed. In pricing European options, we compare the efficiency of grid lattice algorithm with the well-known Black-Scholes formula. The result shows that the grid lattice method achieves a high accuracy comparable to Black-Scholes model. For American options, we compare the lattice algorithm with the other two numerical methods: the finite difference method and the least squares Monte Carlo simulation. The results suggest that the lattice algorithm does a better job than the other two methods in term of accuracy and computational speed. The lattice algorithm also shows a fast convergence when we study the relationship between the option price and the number of determination dates or the relation between the option price and the number of grid nodes.

The grid lattice algorithm has some merits. It is less complex and easy to implement than other numerical methods. It can be applied to price other financial products which have the properties of early exercise and "backward-in-time" nature. These products can be interest rate products, callable bond, callable capped floater swap, etc. This algorithm also requires few discretization nodes as the convergence rate is quite fast. However, the accuracy of the model is still dependent on many factors: numerical integration schemes, selections of the volatility of the underlying, the dynamics model of stock price movements. Further researches can focus on these aspects of the model to improve the accuracy as well as the convergence rate.

Appendix A

Results from Matlab Implemetations

A.1 European put option: Test 1

Data for the European put option: $S = 36$, $K = 40$, r=0.06, $\sigma = 0.2$, $T = 1$. For lattice algorithm, we choose N=50 and vary the number of grid nodes $M \in [50, 400]$. Three grid spaces are chosen: $\delta = [0.025, 0.05, 0.1]$. Below are the prices of the European put option.

M	$\boldsymbol{\delta = 0.025}$	$\boldsymbol{\delta = 0.05}$	$\boldsymbol{\delta = 0.1}$
50	0.1434	1.4468	3.5435
55	0.2280	1.7590	3.6829
60	0.3309	2.0595	3.7629
65	0.4488	2.3437	3.8061
70	0.5789	2.6028	3.8273
75	0.7182	2.8357	3.8376
80	0.8646	3.0374	3.8417
85	1.0157	3.2107	3.8437

M	$\delta = 0.025$	$\delta = 0.05$	$\delta = 0.1$
90	1.1704	3.3537	3.8441
95	1.3265	3.4719	3.8446
100	1.4836	3.5649	3.8444
105	1.6396	3.6393	3.8447
110	1.7942	3.6951	3.8445
115	1.9457	3.7385	3.8447
120	2.0938	3.7694	3.8445
125	2.2368	3.7932	3.8447
130	2.3748	3.8089	3.8445
135	2.5063	3.8212	3.8447
140	2.6316	3.8284	3.8445
145	2.7495	3.8346	3.8447
150	2.8604	3.8375	3.8445
155	2.9635	3.8406	3.8447
160	3.0594	3.8414	3.8445
165	3.1474	3.8431	3.8447
170	3.2285	3.8429	3.8445
175	3.3021	3.8441	3.8447
180	3.3691	3.8435	3.8445
185	3.4293	3.8444	3.8447
190	3.4836	3.8437	3.8445
195	3.5318	3.8445	3.8447
200	3.5749	3.8438	3.8445
205	3.6127	3.8446	3.8447
210	3.6463	3.8438	3.8445
215	3.6755	3.8446	3.8447
220	3.7012	3.8438	3.8445
225	3.7232	3.8446	3.8447
230	3.7425	3.8438	3.8445
235	3.7587	3.8446	3.8447
240	3.7730	3.8438	3.8445

M	$\delta = 0.025$	$\delta = 0.05$	$\delta = 0.1$
245	3.7848	3.8446	3.8447
250	3.7952	3.8438	3.8445
255	3.8036	3.8446	3.8447
260	3.8110	3.8438	3.8445
265	3.8169	3.8446	3.8447
270	3.8221	3.8438	3.8445
275	3.8261	3.8446	3.8447
280	3.8297	3.8438	3.8445
285	3.8324	3.8446	3.8447
290	3.8349	3.8438	3.8445
295	3.8366	3.8446	3.8447
300	3.8383	3.8438	3.8445
305	3.8394	3.8446	3.8447
310	3.8406	3.8438	3.8445
315	3.8412	3.8446	3.8447
320	3.8420	3.8438	3.8445
325	3.8424	3.8446	3.8447
330	3.8430	3.8438	3.8445
335	3.8431	3.8446	3.8447
340	3.8435	3.8438	3.8445
345	3.8436	3.8446	3.8447
350	3.8439	3.8438	3.8445
355	3.8438	3.8446	3.8447
360	3.8441	3.8438	3.8445
365	3.8440	3.8446	3.8447
370	3.8442	3.8438	3.8445
375	3.8441	3.8446	3.8447
380	3.8443	3.8438	3.8445
385	3.8441	3.8446	3.8447
390	3.8443	3.8438	3.8445
395	3.8442	3.8446	3.8447
400	3.8443	3.8438	3.8445

A.2 European put option: Test 2

Data for the European put option: $S = 36$, $K = 40$, r=0.06, $\sigma = 0.2$, $T = 1$. For lattice algorithm, we choose M=501 and vary the number of determination dates $N \in [5 : 5 : 100]$. Three grid spaces are chosen: $\delta = [0.025, 0.05, 0.1]$. Below are the prices of the European put option.

N	$\boldsymbol{\delta = 0.025}$	$\boldsymbol{\delta = 0.05}$	$\boldsymbol{\delta = 0.1}$
5	3.8442	3.8446	3.8447
10	3.8442	3.8446	3.8447
15	3.8442	3.8446	3.8447
20	3.8442	3.8446	3.8447
25	3.8442	3.8446	3.8447
30	3.8442	3.8446	3.8447
35	3.8442	3.8446	3.8447
40	3.8442	3.8446	3.8447
45	3.8442	3.8446	3.8447
50	3.8442	3.8446	3.8447
55	3.8442	3.8446	3.8447
60	3.8442	3.8446	3.8447
65	3.8442	3.8446	3.8447
70	3.8442	3.8446	3.8447
75	3.8442	3.8446	3.8447
80	3.8442	3.8446	3.8447
85	3.8442	3.8446	3.8447
90	3.8442	3.8446	3.8447
95	3.8442	3.8446	3.8447
100	3.8442	3.8446	3.8447

A.3 American put option: Test 1

Data for the American put option: $S = 36$, $K = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 1$. For lattice algorithm, we choose N=50 and vary the number of grid nodes $M \in [2 : 2 : 160]$. Three grid spaces are chosen: $\delta = [0.025, 0.05, 0.1]$. Below are the prices of the American put option.

M	$\boldsymbol{\delta = 0.025}$	$\boldsymbol{\delta = 0.05}$	$\boldsymbol{\delta = 0.1}$
2	4.0000	4.0000	4.0000
4	4.0000	4.0000	4.0000
6	4.0000	4.0000	4.0000
8	4.0000	4.0000	4.0000
10	4.0000	4.0000	4.0000
12	4.0000	4.0000	4.0118
14	4.0000	4.0000	4.1423
16	4.0000	4.0000	4.2590
18	4.0000	4.0000	4.3421
20	4.0000	4.0000	4.3962
22	4.0000	4.0000	4.4304
24	4.0000	4.0543	4.4514
26	4.0000	4.1203	4.4637
28	4.0000	4.1843	4.4707
30	4.0000	4.2415	4.4745
32	4.0000	4.2907	4.4766
34	4.0000	4.3307	4.4776
36	4.0000	4.3632	4.4781
38	4.0000	4.3891	4.4783
40	4.0000	4.4095	4.4784
42	4.0000	4.4256	4.4785

M	$\delta = 0.025$	$\delta = 0.05$	$\delta = 0.1$
44	4.0105	4.4382	4.4785
46	4.0412	4.4480	4.4785
48	4.0737	4.4556	4.4785
50	4.1069	4.4614	4.4785
52	4.1398	4.4658	4.4785
54	4.1718	4.4690	4.4785
56	4.2022	4.4714	4.4785
58	4.2307	4.4732	4.4785
60	4.2571	4.4745	4.4785
62	4.2815	4.4755	4.4785
64	4.3035	4.4761	4.4785
66	4.3234	4.4766	4.4785
68	4.3413	4.4770	4.4785
70	4.3573	4.4772	4.4785
72	4.3716	4.4773	4.4785
74	4.3843	4.4775	4.4785
76	4.3957	4.4775	4.4785
78	4.4058	4.4776	4.4785
80	4.4147	4.4776	4.4785
82	4.4227	4.4776	4.4785
84	4.4298	4.4776	4.4785
86	4.4360	4.4777	4.4785
88	4.4415	4.4777	4.4785
90	4.4464	4.4777	4.4785
92	4.4507	4.4777	4.4785
94	4.4544	4.4777	4.4785
96	4.4577	4.4777	4.4785
98	4.4605	4.4777	4.4785
100	4.4630	4.4777	4.4785
102	4.4651	4.4777	4.4785
104	4.4670	4.4777	4.4785

M	$\delta = 0.025$	$\delta = 0.05$	$\delta = 0.1$
106	4.4686	4.4777	4.4785
108	4.4700	4.4777	4.4785
110	4.4712	4.4777	4.4785
112	4.4722	4.4777	4.4785
114	4.4731	4.4777	4.4785
116	4.4738	4.4777	4.4785
118	4.4745	4.4777	4.4785
120	4.4750	4.4777	4.4785
122	4.4755	4.4777	4.4785
124	4.4759	4.4777	4.4785
126	4.4762	4.4777	4.4785
128	4.4765	4.4777	4.4785
130	4.4767	4.4777	4.4785
132	4.4769	4.4777	4.4785
134	4.4771	4.4777	4.4785
136	4.4772	4.4777	4.4785
138	4.4773	4.4777	4.4785
140	4.4774	4.4777	4.4785
142	4.4775	4.4777	4.4785
144	4.4776	4.4777	4.4785
146	4.4776	4.4777	4.4785
148	4.4777	4.4777	4.4785
150	4.4777	4.4777	4.4785
152	4.4777	4.4777	4.4785
154	4.4777	4.4777	4.4785
156	4.4778	4.4777	4.4785
158	4.4778	4.4777	4.4785
160	4.4778	4.4777	4.4785

A.4 American put option: Test 2

Data for the American put option: $S = 36$, $K = 40$, r=0.06, $\sigma = 0.2$, $T = 1$. For lattice algorithm, we choose M=151 and vary the number of determination dates $N \in [5 : 5 : 100]$. Three grid spaces are chosen: $\delta = [0.025, 0.05, 0.1]$. Below are the prices of the American put option.

N	$\boldsymbol{\delta = 0.025}$	$\boldsymbol{\delta = 0.05}$	$\boldsymbol{\delta = 0.1}$
5	4.3898	4.3909	4.3916
10	4.4424	4.4425	4.4417
15	4.4575	4.4578	4.4579
20	4.4647	4.4648	4.4658
25	4.4689	4.4692	4.4702
30	4.4718	4.4721	4.4729
35	4.4739	4.4743	4.4746
40	4.4754	4.4757	4.4758
45	4.4766	4.4769	4.4766
50	4.4776	4.4779	4.4774
55	4.4784	4.4787	4.4780
60	4.4791	4.4793	4.4786
65	4.4797	4.4799	4.4791
70	4.4802	4.4804	4.4795
75	4.4806	4.4808	4.4799
80	4.4809	4.4812	4.4802
85	4.4812	4.4815	4.4806
90	4.4815	4.4818	4.4809
95	4.4818	4.4821	4.4812
100	4.4820	4.4824	4.4814

Bibliography

Barone-Adesi, G. and Whaley, R. (1987),*Efficient analytic approximation of American option values*, Journal of Finance, vol. 42, no. 2, pp. 301–320.

Boyle, P. (1988), *A lattice framework for option pricing with two state variables*, Journal of Financial and Quantitative Analysis, vol. 23, no. 1, pp. 1–12

Brennan, M. J. and Schwartz, E. S. (1978), *Finite difference methods and jump processes arising in the pricing of contingent claims: a synthesis*, Journal of Financial and Quantitative Analysis, vol 13, no. 3, pp. 461–474.

Carr, P., Jarrow, R., and Myneni, R. (1992), *Alternative characterizations of American put options*, Mathematical Finance, vol. 2, no. 2, pp. 87–106.

Courtadon, G. (1982), *A more accurate finite difference approximations for the valuation of options*, Journal of Financial and Quantitative Analysis, vol 17, no. 5, pp. 697–703.

Cox, J. C., Ross, S. A., and Rubinstein, M. (1979), *Option pricing: a simplified approach*, Journal of Financial Economics, vol. 7, no. 3, pp. 229–263.

Geske, R. and Johnson, H. (1984), *The American put option valued analytically*, Journal of Finance, vol. 39, no. 5, pp. 1511–1524.

Grant, D., Vora, G., and Weeks, D. E. (1996), *Simulation and the early-exercise option problem*, Journal of Financial Engineering, vol. 5, no. 3, pp. 211–227.

Jacka, S. D. (1991), *Optimal stopping and the American put*, Mathematical Finance, vol. 1, no. 2, pp. 1–14.

Jarrow, R. and Rudd, A. (1983), *Tests of an approximate option-valuation formula*, M. Brenner (Ed.), Option Pricing Theory and Applications, Lexington Books, Lexington, MA, pp. 81–100.

John, C. H. (1999), *Options, Futures, and Other Derivatives (fourth ed.)*, Prentice-Hall Inc., Englewood Cliffs, NJ.

Ju, N. (1998), *Pricing an American option by approximating its early exercise boundary as a multi-piece exponential function*, Review of Financial Studies, vol. 11, no. 3, pp. 627–646.

Kim, I. (1990), *The analytic valuation of American options*, The Review of Financial Studies, vol. 3, no. 4, pp. 547–572.

Liu, Z. and Pang, T. (2016), *An efficient grid lattice algorithm for pricing American-style options*, International Journal of Financial Markets and Derivatives, Vol. 5, No. 1, pp.36–55.

Longstaff, F. A. and Schwartz, E. S. (2001), *Valuing American options by simulation: a simple least-squares approach*, Review of Financial Studies, vol. 14, no. 1, pp. 113–147.

MacMillan, L. W. (1986), *An analytical approximation for the American put prices*, Advances in Futures and Options Research, vol. 1, pp. 119–139.

Xiao, T. (2011), *An efficient lattice algorithm for the LIBOR market model*, The Journal of Derivatives, vol. 19, no. 1, pp. 25–40.

Zhao, J. and Wong, H. Y. (2012), *A closed-form solution to American options under general diffusion processes*, Quantitative Finance, vol. 12, no. 5, pp. 725–737.

Zhu, S. P. (2006), *An exact and explicit solution for the valuation of American put options*, Quantitative Finance, vol. 6, no. 3, pp. 229–242.

Zhu, S. P. (2006), *A new analytical approximation formula for the optimal exercise boundary of American put options*, International Journal of Theoretical and Applied Finance, vol. 9, no. 7, pp. 1141–1177.

Declaration

I am Tong Bao Tran, student of School of Business and Economics of the University of Tübingen, declare and certify with my signature that my thesis entitled "Valuation of American Options using a Grid Lattice Algorithm" is entirely the result of my own work. I declare that the work has not be submitted for any other degree or professional qualification. I have faithfully and accurately cited all my sources, including books, journals, handouts, program codes, and unpublished manuscripts, as well as any other media, such as the Internet, letters or significant personal communications.

Tübingen, 19 March 2018

.....

Tong Bao Tran