PRICING MULTI-DIMENSION AMERICAN OPTIONS BY SIMULATION

SUN JUNHUA

(B.Sc.)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS

NATIONAL UNIVERSITY OF SINGAPORE

2004

To My Parents ...

Acknowledgements

This thesis would not have been possible without the help, invaluable suggestion and patient guidance from my supervisor, Dr. Jin Xing. Without him, I would not have learned so much. Thank you very much, sir, for everything you have done for me!

I would like to say a very special thank you to my parents. You always encourage me and support my decision, your words give me great motivation in my study and in my life.

Furthermore, I would like to express my sincere thanks to all my friends in Singapore. I may not list all your names here because there are really too many to be listed, your friendship and encouragement keep me on the hard-working track to this thesis. Thanks are also extended to others including the friendly and helpful academic and non-academic staffs in Department of Mathematics. Last but not least, I would also wish to thank the National University of Singapore for awarding me the Research Scholarship which financially supports me throughout my M.Sc. candidature.

Thanks all!

Sun Junhua

Feb 2004

Contents

Summary

Pricing American-style options in high-dimension has always been a challenging problem. Classical approaches to price options, including lattice method, finite difference, can deal with the low-dimension options pricing well. Originally, people thought that Monte Carlo simulation is not suitable for pricing American options due to the early exercise characteristic. In recent years, new approaches for pricing American option based on Monte Carlo simulation have been proposed one by one. The most popular ones are state-space partitioning, least squares Monte Carlo, and stochastic mesh method.

Our algorithm is based on state-space partitioning method. The main challenge in using this kind of methods is the selection of the state-space partitions. It's natural to imagine that as the resolution of the partitions increase, the option value by this approach will converge to the true value. However, as many scholars noted, "the number of strata required to maintain the same resolution along all dimensions grow exponentially with the number of dimension".

The algorithm we present here uses low-discrepancy sequences as "*Representative State* " to partition the state-space, so that we can deal with the pricing in high

dimensions. The low-discrepancy sequences, such as the sobol sequence, can fill in the space quickly in an efficient way. By using the low-discrepancy sequences, we can avoid the curse of high dimension. Therefore, we can practically apply the algorithm to pricing high-dimensioned American options.

List of Tables

List of Figures

People should not be praised for their virtue if they lack the energy to be wicked.

Introduction

Pricing American securities has received a tremendous amount of attention in the last 25 years, ever since the Black-Scholes model was introduced in 1973. Most of the derivatives securities traded are American style securities, so the need for a method that can generalize Black-Scholes analysis to allow for early exercise opportunities has been the subject of many researchers.

The focus of this thesis is pricing multi-dimension American options by Monte Carlo simulation methods. As the dimension of the problem increases up to 3, simulation has been proved to be a good tool for derivatives pricing. Simulation, first introduced by Boyle [4] to derivatives pricing, is becoming more and more popular method for pricing complex and exotic derivatives. The main advantage of simulation is that the computational workload grows linearly with the number of state variables. Therefore, simulation is the most popular and efficient way to approximate the high dimensional derivatives price.

Once, it was thought that Monte Carlo simulation cannot be extended to price American style options. In the past years, simulation methods for pricing American options have been proposed one by one. The most popular ones are state-space

partitioning, least square Monte Carlo, and stochastic mesh method.

This thesis is organized as follows. Chapter 1 gives a brief introduction to option and option pricing with emphasis on American option. We also present the basic idea of Monte Carlo Simulation. Chapter 2 reviews some popular option pricing models, focusing on American options. At the end of this Chapter, we present the American option pricing in dynamic programming framework. In Chapter 3 we present our state-space partition based algorithm. In Chapter 4, based on our algorithm, we will give some numerical results and compare them with the results of other methods. Appendix C is the MATLAB source code of our methods.

Chapter 1

Foundations

1.1 Introduction to Options

Option is one of the most common derivatives traded in the market. There are 2 types of options: A *call option* is a contract that give the holder the right to buy an underlying asset at a predetermined price on or before a specified date in the future. A *put option* is a contract that give the holder the right to sell an underlying asset at a predetermined price on or before a specified date in the future.

The predetermined price in the option contract is the *exercise price* or *strike price*. The date in the contract is the *maturity* or *expiration date*. The European option can be exercised only on the expiration date, while the American option can be exercised at any time up to the expiration date.

For example, let us consider a put option on Microsoft (MSFT) stock: The put option gives the right to buy one share of MSFT stock for $100 in 12 months' time. Today's MSFT stock price is $90. The '100' is called exercise price. The 12th month is called *maturity*. The MSFT stock on which the option is based is

called the underlying asset. We would like to exercise the put option at maturity if the stock price is below the strike price and not if it is above. If we use S denote the stock price, K denote the strike price. Then at maturity, the put option is worth:

$$\max\{K - S, 0\}$$

This function is called *payoff function*. The max represents the optionality.

Throughout,

- the expiration date will be denoted by T,
- the stocks price at time t by S_t, S_t can be multivariate, or vector-valued and hence applies to American options on multiple assets,
- the strike price by K,
- the interest rate by r,
- the volatility by σ.

For simplicity, throughout this thesis, the interest and volatility are constant, however, they can be made stochastic without any difficulty.

Actually, the fair value of an option in the risk-neutral world is the present value of the expected payoff at maturity under a risk-neutral probability measure.

$$\text{Option Value} = e^{-rT}\mathbb{E}[\text{payoff}(S)]$$

The option price actually is nothing but the expected discounted payoff.

1.2 The Classical Black-Scholes Option Pricing Model

Description of Asset price

Normally the price of assets is often modelled as a continuous-time stochastic process. For practical use, we usually use a stochastic differential equation (SDE) to describe this process. The general form is as follows:

$$dS = \mu(S,t)dt + \sigma(S,t)dW \tag{1.1}$$

where W is Brownian Motion, or Wiener process. $\mathbb{E}(dW) = 0$ and $\mathbb{E}(dW^2) = dt$. We can think of dW as a random variable, drawn from a normal distribution with mean 0, and variance dt.

In Black-Scholes Option Pricing Model, $\mu(S,t) = \mu$ and $\sigma(S,t) = \sigma$, (μ, σ in (1.1) are constant). Black-Scholes Model is a continuous-time model for an asset price. It's the most widely used model for equites, indices, currencies and so on.

Black-Scholes Formulae

As we metioned, in Black-Scholes /Merton model [3], [13], the price of the underlying asset, S_t is assumed to follow the Geometric Brownian Motion (GBM) process

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{1.2}$$

where r is interest rate, and σ is volatility. Both of them are assumed to be constant. W_t is a standard Brownian Motion.

The SDE (1.2) has the closed form solution:

$$S_t = S_0 exp\{(r - \frac{\sigma^2}{2})t + \sigma W_t\} \tag{1.3}$$

After building a risk free portfolio and applying no-arbitrage conditions, we can derive the following PDE for call option price $c(t, S)$, at time t, with initial stock price S:

$$\frac{\partial c}{\partial t} + rS\frac{\partial c}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 c}{\partial S^2} = rc. \tag{1.4}$$

This is the Black-Scholes-Merton PDE. The terminal boundary condition, for European call, is

$$c(T, S_T) = max\{S_T - K, 0\} \tag{1.5}$$

There is a closed form solution to the PDE (1.4) with the initial boundary condition, (1.5): the European option price at time t, with initial dividend-free stock price S, can be expressed as :

$$c(t, S) = SN(d_1) - Ke^{-r(T-t)}N(d_2)$$

where $N(\cdot)$ is the standard cumulative normal distribution function:

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

$$d_1 = \frac{ln(\frac{S}{K}) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = d_1 - \sigma\sqrt{(T-t)}$$

This is the famous Black-Scholes Formulae for European option, which gives the analytical solution to the single asset European option price.

1.3 American Option: Problem Formulation

American options are contracts that may be exercised early, prior to *maturity*. The right to exercise at any time is clearly valuable. The price, or the value of an

American option can't be less than an equivalent European option. Although, the American options give the option holder more rights, they also give the holder more problems: when to exercise ? One central problem in pricing American options is to decide when is the best time to exercise, which makes American options much more fascinating than European cousins. The American option value is maximized by an optimal exercise strategy.

A general class of American option pricing problem can be formulated through an $\mathbb{R}^d$-valued Markov process $\{S_t, 0 \leq t \leq T\}$, (with S_0 fixed), that records all relevant financial information, including the prices of underlying assets. We restrict our attention to options admitting a finite set of exercise opportunities $t_0 = 0, t_1, ..., t_N = T$. Usually, such restriction ia part of the option contract, and such option actually are called "Bermudan". When N increases to infinity, we may view the finite exercise date as an approximation to the continuous exercise date.

If exercised at time t_i, $i = 0, 1, 2, .., N$, the option pays $h(S_{t_i})$ for some known functions $h(\cdot)$ mapping $\mathbb{R}^d$ into $[0, +\infty)$. Let Γ_i denote the set of stopping times (with respect to the history of S) taking values in $\{t_i, t_{i+1}, ...t_N\}$ and define:

$$V_i^*(x) = \sup_{\tau \in \Gamma_i} \mathbb{E}^{\mathrm{Q}}[e^{-r\tau} h(S_\tau) | S_{t_i} = x], x \in \mathbb{R}^d \tag{1.6}$$

for $i = 0, 1, 2, .., N$, where Q is an appropriate risk-neutral measure, (see Duffie [10] for details on Q). $h(\cdot)$ is the payoff function, and *sup* is achieved by all stopping times $\tau \leq T$. Then $V_i^*(x)$ is the value of the option at t_i in the state x, given the option was not exercised prior to t_i.

The option values satisfy the dynamic programming equations:

$$V_N^*(x) = h_N(x) \tag{1.7}$$

$$V_i^*(x) = \max\{h_i(x), \mathbb{E}^{\mathrm{Q}}[e^{-r(t_{i+1}-t_i)} V_{i+1}^*(h(S_{t_{i+1}})) | S_{t_i} = x]\} \tag{1.8}$$

for $i = 0, 1, 2, .., N$. These can be rewritten in terms of continuation values.

$$C_i^*(x) = \mathbb{E}^{\mathrm{Q}}[e^{-r(t_{i+1}-t_i)}V_{i+1}^*(h(S_{t_{i+1}}))|S_{t_i} = x]$$

as

$$C_N^*(x) = 0 \tag{1.9}$$

$$C_i^*(x) = \max\{h(x), \mathbb{E}^{\mathrm{Q}}[e^{-r(t_{i+1}-t_i)}V_{i+1}^*(h(S_{i+1}))|S_{t_i} = x]\} \tag{1.10}$$

for $i = 0, 1, 2, .., N$. The option values satisfy

$$V_i^*(x) = \max\{h(x), C_i^*(x)\}.$$

All these can be calculated from the continuation values.

In this thesis, we only consider the case, where only a finite number of discrete exercise date $t_0 = 0, t_1, ..., t_N = T$ exist. In such case, the *sup* in (1.6) becomes *max*:

$$V_i^*(x) = \max_{\tau \in \Gamma_i} \mathbb{E}^{\mathrm{Q}}[e^{-r\tau}h(S_\tau)|S_{t_i} = x], x \in \mathbb{R}^d \tag{1.11}$$

where *max* is taken over all stopping times τ in the set Γ_i, $i = 0, 1, 2, .., N$.

The formulation (1.6) is general enough to include quite a wide range of American style option products, such as the classical American put, two-asset minimum put, five-asset minimum put as special cases. For example, in classical American put: $h(S_t) = (K - S_t)^+$, the option value(1.6) becomes:

$$\max_{\tau \in \Gamma_i} \mathbb{E}^{\mathrm{Q}}[e^{-r\tau}(K - S_\tau)^+|S_{t_i} = x] \tag{1.12}$$

1.4 Monte Carlo Simulation for Option Pricing

Monte Carlo Simulation Framework

Write

$$\text{Option Value} = e^{-rT}\mathbb{E}[\text{payoff}(S)]$$

where the expectation is with respect to the risk-neutral probability.

Monte Carlo simulation for valuing the price of the option can be implemented as follows:

1. Simulate the risk-neutral random walk, starting at today's value of the asset S_0, over the required time horizon. This time period starts today and continues until the maturity of the option. This gives one sample path of the underlying asset price path.

2. For this sample path, compute the option payoff.

3. Simulate many more such sample paths over the time horizon.

4. Compute the average payoff over all sample paths.

5. Take the present value of the average, to get the approximate option price.

In the first step of this algorithm, we need to generate sample paths of underlying stock. A simple way to approximate (1.1) is the *Euler Method*:

$$\Delta S = \mu(S,t)\Delta t + \sigma(S,t)\sqrt{\Delta t}Z(0,1)$$

where $Z(0,1)$ is drawn from a standard Normal distribution. This discretization error is $O(\Delta t)$.

Monte Carlo Method is more attractive than other methods for option pricing:

- It's easy to understand and simple to implement.
- Correlations can be easily modelled.
- To get better accuracy, just run more simulations.
- The models can be changed without much efforts.
- Path dependency can also be easily handled.
- The error convergence rate $1/\sqrt{n}$ is independent of the dimension of the problem.

The error of the Monte Carlo method goes $1/\sqrt{n}$ as a consequence of the central limit theorem. While this error may not look very impressive, it is often the best that can be managed for S which has a high dimension. For the high-dimension American Option pricing problem, the Monte-Carlo method is the most method.

Monte Carlo Method is very powerful and general. This technique carries over to exotic and path-dependent contracts. Just simply simulate paths, and corresponding cash flows, estimate the average payoff and take the present value. That's all!

Pricing American Options by Simulations

Applying Monte Carlo simulation to American option is very hard, and was considered impossible. The cash flow of American options not only depend on the price path of the underlying assets, but also depend on the decisions of the option holder. The problem lies in the time direction in which we are solving. In The formulation (1.6):

$$\sup_{\tau \in \Gamma_i} \mathbb{E}^{\mathrm{Q}}[e^{-r\tau} h(S_\tau)]$$

We need to estimate the optimal stopping time τ, which lies in the heart of the American option pricing problem. We can use simulation to estimate the expectation in (1.6) via a recursive algorithm.

For pricing high dimension American options, simulation method is a major tool we can turn to. Baraquad and Martineau [2] are perhaps the first to consider pricing high-dimension American options. Recently, appeared many new Monte Carlo methods for American option, such as Longstaff and Schwartz's [1] Least Square Monte Carlo, Broadie and Glasserman's[7] Stochastic Mesh, and Tilley's [18] State-Space partition and so on. All these methods are constructed within the backward dynamic programming framework to handle the early exercise characteristic.

Chapter 2

A Review of American Option Pricing Model

For American options a solution of the Black Scholes equation in general cannot be found analytically. The reason is that the point at which early exercise of the option at any instant of time is optimal is a priori unknown. In the framework of the PDE it can be treated as a free boundary problem. Let p be the standard American put option value. We have the Black-Scholes-Merton PDE for p :

$$\frac{\partial p}{\partial t} + rS\frac{\partial p}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 p}{\partial S^2} = rp$$

.

However, since we don't know when the option holder will exercise the option. The boundary condition becomes a " free boundary " :

$$p(t, S) \geq \max\{E - S, 0\}$$

for all $0 < t \leq T$. It is a free-boundary problem. Unfortunately, unlike the European option case, we don't have the beautiful closed form solution for p.

We now turn to numerical method for pricing American option. The popular numerical methods for pricing American options are as follows:

- Tree/Lattice Method
- Finite Difference Method
- Least Square Monte Carlo Method
- Stochastic Mesh Method
- State-Space Partition Method

All these methods have their advantages and disadvantages.

2.1 Tree/Lattice Method

Tree or Lattice method is one of the simplest and classical methods for option pricing. The classical binomial tree methods was proposed by Cox, Ross and Rubinstein, [9]. This method is based on the random walk approximation to the Brownian motion.

We consider the partition of the time horizon $[0, T]$: $t_0 = 0, t_1, t_2, ..., t_N = T$. At each time step, it is assumed that the underlying asset follows a binomial process, that is,it either jumps up by a proportion u with probability p or goes down by a proportion d with probability $1 - p$.

Like Black-Scholes model, we can build a risk free portfolio. We refer to the jth node at time ith time step as (i, j). The stock price at (i, j) is $S_0 u^i d^{i-j}$. The European call option price is

$$V_0 = e^{-r\Delta T}\mathbb{E}^{\mathrm{Q}}[h(S_T)]$$

where $h(S)$ is the European call payoff function: $h(S) = (S - K)^+$.

The value of the European call option price at expiration date T is$(S_T - K)^+$. We get

$$V_{N,j} = (S_0 u^j d^{N-j} - K)^+, j = 0, 1, 2....N$$

The probability of price going up in risk neutral is

$$p = \frac{e^{r\Delta t} - d}{u - d}, \Delta t = \frac{T}{N}.$$

For European options,we assume no early exercise. The European call option price at (i, j) node is give by

$$V_{i,j} = e^{-r\Delta t}[pV_{i+1,j+1} + (1 - p)V_{i+1,j}] \tag{2.1}$$

For American option, we can exercise the option early. We need to compare the exercise value and continuation value at every (i, j) node. The option price (2.1) becomes for American Options :

$$V_{i,j} = max\{e^{-r\Delta t}[pV_{i+1,j+1} + (1 - p)V_{i+1,j}], S_0 u^i d^{i-j} - K\} \tag{2.2}$$

The main advantage of tree methods is that it is easy to understand and simple to implement. Also if we use large number of steps, we can get good precision. However, tree methods can only handle low dimension pricing. The reason is that the tree paths increase exponentially as dimension increases.

2.2 Finite Difference Method

The price of a financial derivative,like options, can often be formulated as the solution to a parabolic PDE, such as Black-Scholes-Merton PDE, subject to boundary conditions specified by the payoff of the derivatives. The PDE can be solved numerically by a suitable finite difference method.

The first step is to introduce a grid of mesh points (ik, jh), $i, j \in Z^+$ where h and k are mesh parameters usually small. Next, approximate solutions of the PDE at these points is obtained by solving a problem, where the partial derivatives are replaced with finite differences.

When we choose different experssions which are centerd around time step $i+1$, i, or $i+\frac{1}{2}$, we can get the explicit, implicit or Crank-Nicolson finite difference scheme.

These 3 scheme can be compared in terms of convergence, consistency, and stability, as follows:

Scheme	convergence	consistency	stability
Explicit	$O((\Delta x)^2 + \Delta t)$	only if $\Delta x > \sqrt{2\Delta t}$	only if $\Delta x > \sqrt{2\Delta t}$
Implicit	$O((\Delta x)^2 + \Delta t)$	Unconditionally	Unconditionally
Crank-Nicolson	$O((\Delta x)^2 + (\frac{\Delta t}{2})^2)$	Unconditionally	Unconditionally

Table 2.1: Finite Difference Method

Just like lattice methods, finite difference methods can deal with European and American derivatives with very good precision, if we use a fine partition of time horizon and stock price range. But it's very hard to extend them to path-dependent derivatives. And also for high dimensions, the number of grid will also increase exponentially with dimension. Monte Carlo simulation methods are then needed.

2.3 Least Square Monte Carlo Method

As we know, the main problem for pricing American options is that the options can be exercised early. The option holder must decide, at each exercise date, whether to exercise the option or to wait. This decision depends on the comparison of

exercise value and continuation value. The exercise value is normally easy to determine. The "simple yet powerful" Least Square Monte Carlo Method proposed by Longstaff and Schwartz [1] attempts to approximate the price of American option using cross-sectional information from simulated paths. The optimal exercise strategy is successively approximated backwards in time on the paths by comparing the intrinsic values to the continuation values projected onto a number of basis functions over the states. In this regression analysis, they use a set of basis functions whose arguments are based on the underlying assets prices. The fitted value of these regressions is taken as the expected continuation values. By comparing these estimated continuation values with the immediate exercise values, the optimal stopping rule will be found. Experimental success is reported for the Least Square Monte Carlo method. However, in high dimension, the basis function must be chosen carefully. Clément, Lamberton, Protter [11] provided proofs of the convergence for the LSM. They show that the convergence is $\sqrt{n}$ in the number of paths used. The convergence behaviour in the number of basis function has not been determined.

As before, we assume a finite number of exercise dates $0 < t_1 < t_2 < ... < t_N = T$ in the time horizon $[0, T]$. We have a probability space,$(\Omega, \mathcal{F}, \mathrm{P})$, and a risk-neutral equivalent martingale measure (EMM) Q. Let $C(\omega, s; t, T), \omega \in \Omega, s \in (t, T]$ denote the path of option cash-flows, condition on the option being exercised after t and the option holder following the optimal stopping strategy at any time after t. The continuation value is, then, under no-arbitrage conditions, the risk-neutral expectation of the future discounted cash flows $C((\omega, s; t_i, T)$:

$$F(\omega; t_i) = \mathbb{E}^{\mathrm{Q}}[\sum_{j=i+1}^{N} exp(-\int_{t_i}^{t_j} r(\omega, s)ds)C(\omega, t_j; t_i, T)|\mathcal{F}_{t_i}] \tag{2.3}$$

where $r(\omega, s)$ is the risk-free interest rate and $\mathcal{F}_{t_i}$ is the information set at time t_i.

The idea underlying the Least Square Monte Carlo Methods (LSM) algorithm is that this conditional expectation can be approximated by a least-square regression for each exercise date. At time t_{N-1}, it is assumed that $F(\omega, t_{N-1})$ can be expressed as a linear combination of basis functions $L_j(S)$ in L^2 space.

$$F(\omega; t_i) = \sum_{j=0}^{+\infty} a_j L_j(S), a_j \in \mathbb{R}$$

That is approximated by

$$F_M(\omega; t_i) = \sum_{j=0}^{M} a_j L_j(S), a_j \in \mathbb{R}. \tag{2.4}$$

This procedure is repeated backward in time until the first exercise date.

2.4 Stochastic Mesh Method

Broadie and Glasserman [6] use a stochastic tree algorithm to give both a low-biased and high-biased estimator of the option price, both asymptotically unbiased. Their method requires an exponentially increasing amount of work in the number of exercise opportunities. In their subsequent paper [7], they present a related method based on a stochastic mesh which does not suffer from this problem. However, this method has been found by a few authors to be slow and to have a large finite-sample bias.

Like tree methods, the stochastic mesh methods approximate the American option price by solving a randomly sampled dynamic programming (DP) problem. The difference is that in valuing the option at a node at *ith* time step, the mesh use values from all nodes at time step $i + 1$, not just those that are successors of the current node. That's why it is called mesh not tree. It keeps the number of nodes

at each time step fixed, avoiding the exponential growth characteristic of a tree.

The main idea of Stochastic Mesh Method goes like this. In the mesh, we use S_{ij} to denote the jth node at ith time step, $i = 1, ..., m$ and $j = 1, ..., b$. We use $V_{i,j}$ to denote the estimated value at this node. At the terminal nodes, we set $V_{m,j} = h(X_{mj})$, we work backward by computing

$$V_{i,j} = max\{h(S_{ij}), \frac{1}{b}\sum_{k=1}^{b} W_{jk}^{i} V_{i+1,k}\} \tag{2.5}$$

where $h(\cdot)$ is the payoff function, and W_{jk}^{i} is some set of weight. At the root node, we get the option value at time 0

$$V_0 = \frac{1}{b}\sum_{k=1}^{b} V_{1k}.$$

Boyle [15] recently extended the stochastic mesh method of Broadie and Glasserman [7] with their low discrepancy mesh method. This involves generating a set of low discrepancy interconnected paths and using a dynamic programming approach to find prices on the mesh.

2.5 State-Space Partition Method

In [18], Tilley first proposed a "bundling algorithm". This is the first kind of "State space partition" method, to use simulation for American option pricing. Also another "state space partition" algorithm proposed by Barraquand and Martineau in [2] is "stratified state aggregation (SSA)" algorithm.

We still use the settings, symbols $S = (S(t_1), S(t_2), ..., S(t_N))$, S may be multi-dimensional Markov process. There are $0 = t_0 < t_1 < t_2 < ... < t_N = T$, N exercise dates.

For each exercise date, t_i, $i = 1, ..., N$, let $A_{i1}, ..., A_{ib_i}$ be a partition of the state space of $S(t_i)$ into b_i subsets. We define the *transition probabilities* :

$$p^i_{jk} = P\{S(t_{i+1}) \in A_{i+1,k} | S(t_i) \in A_{ij}\} \tag{2.6}$$

for all $j = 1, ..., b_i$, $k = 1, ..., b_{i+1}$, $i = 0, ..., N$. Take this to be 0, if $P(S(t_i) \in A_{ij}) = 0$. For $t = 0$, just let $b_0 = 1$, $A_{01} = S_0$. Then define:

$$h_{ij} = \mathbb{E}[h(S(t_i))|S(t_i) \in A_{ij}]$$

for $i = 1, ..., N$,$j = 1, ..., b_i$. Take this to be 0, if $P(S(t_i) \in A_{ij}) = 0$. We recursively have

$$V_{Nj} = h_{Nj} \tag{2.7}$$

$$V_{ij} = \max\{h_{ij}, \sum_{k=1}^{b_{i+1}} p^i_{jk} V_{i+1,k}\} \tag{2.8}$$

Here, the continuation value is computed as $\sum_{k=1}^{b_{i+1}} p^i_{jk} V_{i+1,k}$.

We can use Monte Carlo simulation to compute the transition probabilities. First, we denote N^i_{jk} as the number of paths at time t_i that move from A_{ij} to $A_{i+1,k}$. Then we can estimate the p^i_{jk} by:

$$\widetilde{p}^i_{jk} = \frac{N^i_{jk}}{\sum_{l=0}^{i} N^i_{jb_l}}$$

By using $\widetilde{p}^i_{jk}$ to approximate p^i_{jk} in (2.7),and work backward, we can get an approximation of option price: V_{01}.

Paul Glasserman pointed out "the main challenge in using any variant of this approach (state space partition methods) lies in the selection of the state-space partitions." and "...the number of strata required to maintain the same resolution along all dimensions grows exponentially with the number of dimensions."

2.6 Summary

A numerical valuation of an American option using the (1.6) involves a *sup*. For finite exercise date, it becomes maximization over the set of all stopping times τ. Since the set of possible candidates for this stopping time function τ is so huge, direct maximization is almost impossible.

Therefore, we turn to *Bellman's principle of Dynamic Programming*. We assume the underlying asset to be *Markov process*: $\{S_t, 0 \leq t \leq T\}$ and the optimal stopping time τ only depends on the time and and current S_t. Most approaches to pricing American options is based on this dynamic programming framework, evaluating backward from maturity of the option to today.

Let $h(\cdot)$ be the payoff function. $V_i(S)$ denotes the value of the option at time t_i given $S_{t_i} = S$, assuming the option has not been exercised early. What we want to know ultimately, is $V_0(S_0)$ at $t = 0$. At maturity time $t_N = T$, we have the value of V_N:

$$V_N(S_T) = h(S_T) \tag{2.9}$$

The rest of V_i can be computed backward by dynamic programming recursively:

$$V_{t-1}(x) = \max\{h(x), \mathbb{E}[e^{-r\Delta t} V_t(S_t)|S_{t-1} = x]\} \tag{2.10}$$

Actually, the conditional expectation in (2.10) is the *continuation value*, or holding value for the American options. Using (2.10), we will finally get the value $V_0(S_0)$.

2.6.1 General Algorithm for Pricing American Option by DP

We assume that the underlying asset S are Markov process $S = \{S(t_1), S(t_2), ..., S(t_N)\}$. S can be multivariate. For example, S can be 2 dimensions $S = (S^1, S^2) = \{(S^1(t_1), S^2(t_1)), ..., (S^1(t_N), S^2(t_N))\}$. There are $0 = t_0 < t_1 < t_2 < ... < t_N = T$, N exercise dates.

General Algorithm

A general algorithm for pricing American option by Monte Carlo simulation and dynamic programming is composed of the following steps:

Step 1. Simulate n independent replications (paths) of the Markov chain $\{S_j(t_1), S_j(t_2), ..., S_j(t_N)\}, j = 1, 2, ...n$ for every underlying asset S.

Step 2. At time T, compute discounted exercise value: $e^{-r\Delta t}h(S(t_N))$ of every simulated path. In standard American put option case, $h(S) = max\{K - S, 0\}$, where K is the strike price. Let Y_T be a $n \times 1$ vector:

$$Y_T = (e^{-r\Delta t}h(S(t_N)))_{n\times 1} \text{ and } V_T = Y_T.$$

Step 3. For $t = t_{N-1}, t_{N-2}..., t_1$, recursively, compute the continuation value

$$C_t(x) = \mathbb{E}[e^{-r\Delta t}V_{t+1}(S(t+1))|S(t) = x] \tag{2.11}$$

Let H_t be a $n \times 1$ vector: $(C_t(x))_{n\times 1}$ and update $V_t = max\{Y_t, H_t\}$ for every timestep t.

Step 4. Finally, the optimal value , V_0^* can be approximated as

$$V_0 = \frac{1}{n}IV_1 \tag{2.12}$$

where $I = [1, 1, ..., 1]_{1\times n}$ row vector.

We have got an approximation of American option: V_0.

2.6.2 Stopping Rules

Now we define a $n \times N$ matrix of continuation values $\mathcal{H} = [H_1, H_2, ...H_N] = \{H_{ij}\}$ and a $n \times N$ matrix of exercise values $\mathcal{Y} = [Y_1, Y_2, ..., Y_N] = \{Y_{ij}\}$. For each path, $i = 1, ..., n$ the stopping time τ_i can be calculated as follows:

Step 1. Set $\tau_i = 1$ if $Y_{i1} \geq H_{i1}$,

Step 2. For $k = 2, ..., N-1$, set $\tau_i = k$, if $Y_{i1} < H_{i1}, ..., Y_{i,k-1} < H_{i,k-1}$ and $Y_{ik} \geq H$,

Step 3. Finally, $\tau_i = N$, if $Y_{it} < H_{it}$ for all $t = 1, ..., m-1$

Now we have an approximation of optimal stopping time τ^*. Meanwhile, we get another approximation for the optimal value, denoted as $\widetilde{V_0}$:

$$\widetilde{V_0} = \frac{1}{n}\sum_{i=1}^{n} Y_{i,\tau_i} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{N} Y_{ij} 1(\tau_i = j). \tag{2.13}$$

In most cases, $\widetilde{V_0}$ is at least as good as V_0. For small number of sample paths, it's better than V_0.

2.6.3 Continuation Value

Continuation value (2.11) is expressed as a conditional expectation. Estimation of this conditional expectation is one of the most important problems in American option pricing.

In the dynamic programming framework, different approximations to the continuation value (2.11) will lead to the different pricing methods for American option, as introduced in the pervious chapter.

This is true for binomial tree method, in which the continuation value (2.11) becomes the average of the 2 children nodes in (2.2):

$$e^{-r\Delta t}[pV_{i+1,j+1} + (1-p)V_{i+1,j}]$$

And if the continuation value (2.11) is approximated by kind of regression equation, like (2.4),

$$F_M(\omega; t_i) = \sum_{j=0}^{M} a_j L_j(S), a_j \in \mathbb{R}$$

then it is the Least-Square Monte Carlo Method. In Stochastic Mesh method, (2.11) is estimated by (2.5)

$$\frac{1}{b}\sum_{k=1}^{b} W_{jk}^{i} V_{i+1,k}.$$

For State-space partition method, we use the following estimation

$$\sum_{k=1}^{b_{i+1}} p_{jk}^{i} V_{i+1,k}$$

to approximate (2.11).

Chapter 3

Low-discrepancy sequence based State-Space Partition Algorithm

3.1 Challenges

In [18], Tilley proposed "bundling algorithm" for one dimension. Tilley didn't extend his approach to the high dimension case. This is the corresponding algorithm **DP1**, in Fu [12]. Fu [12] also pointed out that in extending this "bundling algorithm" to high dimension case, there are two problem that need to be considered: *the bundling procedure* and *sorting procedure*. See Broadie and Glasserman [6], Clewlow and Strickland [8] for more discussion on this.

We try to solve this exension by applying Yakowitz's "*Representative State* " idea to Tilley 's "bundling algorithm". We use low-discrepancy sequence as "*Representative State* ". The low-discrepancy sequences, such as the Sobol sequence, can fill in the space quickly in an efficient way, even in high dimension case.

Our Algorithm's main idea comes from Yakowitz's "*Representative State* ". Yakowitz, in [20] and its counterpart theoretical work [19], gave the basic algorithm to use "*Representative State* " approach to approximate the *transition probabilities*, a similar kind of conditional expectation like our continuation value (2.11). In [20], Yakowitz proposed a idea of the state partition and give a form like (2.6) in one dimension case. However, we noticed that the choice of "*Representative State*" is a problem for applying this idea to high dimension case:

In Yakowitz's method, the representative states$\{c_j\}$ should be " found" by solving the following problem:

Choose N-tuples$\{c_v\}_{v=1}^m$ so that the value of

$$J(c_1, c_2, ..., c_m) = \sum_{j=N}^{n} [\min \|x_j - c_v\|^2]$$

is "not great". x_j is the sample point in the Monte Carlo simulation.

It is time-consuming to find all the representative states, especially in high dimension case. Our algorithm won't find such representative state. Instead, we predetermine them using low discrepancy sequence.

3.2 Assumptions and Settings

We will simulate n underlying asset price paths. We consider a complete and arbitrage free market. The assets prices all follow GBM:

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

We assume:

- There are $t = 1, 2, ..., N$, N exercise dates.

- Let $CQ = \{c_{t,1}, ..., c_{t,m}\}$ be a low-discrepancy, or quasi-Monte Carlo sequence to be used as *representative points* in t-th period, $t = 1, ..., N$, where $m = \sqrt{n}$, n is the number of paths and N is the number of periods.
- Let $S_{t,i}$ denote the stock price on i-th path and t-th period, $i = 1, ..., n$, $t = 1, ..., N$.
- Let $\{\mathcal{C}_{t,j}\}$ be the set of nearest points of $\{S_{t,i}\}$.
- Let $h(t, s)$ be the payoff from exercise at time t in state s.
- Let $q(t, s)$ be the option value at (time, state) pair (t, s).

Here are the assumption for our algorithm :

Assumption 1: The underlying assets price S_t are all Markov processes.

Assumption 2: S_t and $q(t, S_t)$ are all bounded for $t = 1, ..., N$.

This assumption can be easily relaxed.

3.3 Pricing Algorithm

We present our algorithm within the dynamic programming framework in Chapter 2. The option value at (time, state) pair (t, s) is obtained by backward dynamic programming:

$$q(N, s) = h(N, s),$$

for all s and for $t = N - 1, ..., 1$,

$$q(t, s) = \max\{h(t, s), E[q(t + 1, S_{t+1})|S_t = s]\}.$$

Algorithm: Our algorithm proceeds as follows:

STEP 1. (Path Generation) We simulate n independent replications (paths) of the underlying asset $\{S_{t,i}\}$, $i = 1, ..., n, t = 1, ..., N$, that follow GBM:

$$\Delta S_{t,i} = \mu \Delta t + \sigma \sqrt{\Delta t} Z(0,1)$$

where $Z(0,1)$ is drawn from a standard normal distribution.

At the same time, we use low-discrepancy sobol sequence $CQ = \{c_{t,1}, ..., c_{t,m}\}$ and Normal inverse function $N^{-1}(x)$ to generate our representative stock price $\{C_{t,j}\}$:

$$\Delta C_{t,j} = \mu \Delta t + \sigma \sqrt{\Delta t} N^{-1}(c_{t,j})$$

$$C_{t,j} = C_{t-1,j} + \Delta C_{t,j}$$

where at $t = 0$, $C_{0,j} = S_0$.

STEP 2. At maturity T, we first compute

$$q_n(N, s) = h(N, s),$$

for all s.

STEP 3.(Continuation Value) Then, recursively, for for $t = N - 1, ..., 1$, and every path n

$$q_n(t, s) = \max\{h(t, s), c_n(t, s)\}, \tag{3.1}$$

where $c_n(t, s)$ is called the *continuation* value, which, by our algorithm, is calculated by

$$c_n(t,s) = \frac{1}{|\mathcal{C}_n(s)|} \sum_{y \in \mathcal{C}_n(s)} q_n(t+1, S_{t+1}(y)). \tag{3.2}$$

Here $\mathcal{C}_n(s)$ is the nearest points set containing s to be constructed below and $S_{t+1}(y)$ is the stock price at period $t+1$ following y. $|\mathcal{C}_n(s)|$ is the cardinality of $c_n(t,s)$.

STEP 4. Finally, the American option price at $t = 0$ can be approximated as

$$V_0 = \frac{1}{n} \sum_{i=1}^{n} q_i(0, S_0) \tag{3.3}$$

$\mathcal{C}_n(s)$ is constructed as follows: Define:

For every $c_{t,j}$,

$$A_{j,n} = \{S_{t,i} : 1 \le i \le n \text{ and } ||S_{t,i} - c_{t,j}|| \le ||S_{t,i} - c_{t,k}||, 1 \le k \le n^{1/2}\};$$

C_n is a set of $c_{t,j}$:

$$\mathrm{C}_n = \{c_{t,j} : 1 \le j \le n^{1/2} \text{ such that } A_{j,n} \text{ has } \ge [n^{1/3}] \text{ elements}\};$$

$$\begin{aligned} c_n(x) &= c_{t,v} \text{ where } c_{t,v} \text{ is element of least index in } \mathrm{C}_n \text{ such that} \\ &\quad ||x - c_{t,v}|| \le ||x - c_{t,k}||, \text{all } c_{t,k} \in \mathrm{C}_n, \end{aligned}$$

and $\mathcal{C}_n(s)$ is the nearest points set containing s:

$$\mathcal{C}_n(s) = \{S_{t,i} \in A_{v,n} : c_n(s) = c_{t,v}\}$$

For stopping rules, we have already detailed at the end of Chapter 2. We continue to use (2.13):

$$\widetilde{V_0} = \frac{1}{n} \sum_{i=1}^{n} Y_{i,\tau_i} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{N} Y_{ij} 1(\tau_i = j).$$

And at last we our approximation to the American option price $\widetilde{V}_0$.

This is our algorithm for pricing American options, low-discrepancy sequence based state-space partition algorithm.

Chapter 4

Numerical Results

In this section, we present computational results from our algorithm and compare them with results of other authors. We choose 3 examples from other authors' papers:

1. Rogers[17] standard American put.

2. Rogers[17] 2-dim min-American put.

3. Fu [12] 5-dim min-American put.

In the following, we will consider pricing options for which the underlying state variable are log-normally distributed stock prices. The simulated paths were generated using closed form solution to the SDE:

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

i.e.

$$S_t = S_{t-1} exp\{(r - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t} Z(0,1)\}$$

where time step Δt is the time step used in approximation, $\Delta t = t_i - t_{i-1}$, r is the interest rate, σ is the volatility. Both are assumed to be constant.

4.1 Classical Standard American Put Option

Consider the standard Black-Scholes model, in which the price of the underlying asset follows a log-normal distribution. A put option has payoff funtion

$$h(S_t, t) = max\{S_t - K, 0\}$$

with a finite set of N exercise opportunities. Put options on single underlying asset.

Parameter settings

K=100; r=0.06; T=0.5; σ=0.4

Number of simulation paths= 10000;

Time Step: N=50;

Simulation prices of standard American put options on single underlying asset.

S_0	European (Black-Scholes)	American(True)	Our Results	Standard Error
80	20.6893	21.6059	21.5012	0.21
85	17.3530	18.0374	17.9921	0.19
90	14.4085	14.9187	14.8655	0.22
95	11.8516	12.2314	12.2012	0.23
100	9.6642	9.9458	9.8532	0.33
105	7.8183	8.0281	8.0009	0.15
110	6.2797	6.4352	6.3013	0.11
115	5.0113	5.1265	5.0619	0.12
120	3.9759	4.0611	4.0019	0.10

Table 4.1: Classical standard American put option

4.2 Minimum American Put on Two Assets

We consider options on n log-normal assets, given by:

$$S_i(t) = S_i(0)exp\{\sigma_i W_i(t) + (r - \sigma_i^2/2)t\}$$

for $i = 1, ...n$

The assets are assumed to be independent. The payoff function is:

$$\max_{i=1,...,n} \{K - S_i(t), 0\}$$

Parameter settings

K=100; r=0.06; T=0.5; $\sigma_1 = \sigma_2 = 0.6$

Number of simulation paths= 10000;

Time Step: N=50;

Minimum American put on two assets: $(\sigma_1 = \sigma_2 = 0.6)$

S_{01}	S_{02}	European (Black-Scholes)	FD Price	Our Results	Std Error
80	80	36.859	37.30	37.1648	0.31
80	100	31.639	32.08	31.9598	0.26
80	120	28.652	29.14	29.0720	0.27
100	100	24.728	25.06	24.8799	0.25
100	120	20.610	20.91	20.7756	0.22
120	120	15.704	15.92	15.8526	0.20

Table 4.2: Minimum American put on two sssets -1

FD price is finite difference methods price.

Parameter settings

K=100; r=0.06; T=0.5; $\sigma_1 = 0.4$, $\sigma_2 = 0.8$

Number of simulation paths= 10000;

Time Step: N=50;

Minimum American put on two assets: ($\sigma_1 = 0.4$,$\sigma_2 = 0.8$)

S_{01}	S_{02}	European (Black-Scholes)	FD price	Our Results	std Error
80	80	37.5540	38.01	38.00	0.22
80	100	31.8075	32.23	32.10	0.21
80	120	28.0900	28.54	28.44	0.19
100	80	32.8564	33.34	33.23	0.16
100	100	25.4666	25.81	25.74	0.18
100	120	20.4767	20.75	20.66	0.14
120	80	30.6872	31.21	31.11	0.13
120	100	22.4413	22.77	22.60	0.12
120	120	16.7641	16.98	16.86	0.14

Table 4.3: Minimum American put on two assets -2

4.3 Minimum American Put on Five Assets

We consider options on n log-normal assets, given by:

$$S_i(t) = S_i(0)exp\{\sigma_i W_i(t) + (r - \delta - \sigma_i^2/2)t\}$$

for $i = 1, ...5$

The assets are assumed to be independent. The payoff function is:

$$\max_{i=1,...,5}\{K - S_i(t), 0\}$$

Parameter settings

K=100; r=0.05; T=1.0; $\sigma = 0.20$, $\delta = 0.10$

Number of simulation paths= 10000;

Time Step: N=3;

S_0	American(True)	Our Results	std error
70	0.55	0.46	0.01
80	2.7	2.59	0.01
90	7.8	7.23	0.02
100	15.9	15.52	0.15
110	25.8	25.50	0.19
120	36.5	36.11	0.21
130	47.4	47.10	0.33

Table 4.4: Minimum American put on five assets

5-dim min-American put, BG prices in Michael C. Fu [12], Table 21 are reported as our "true value". Compared with earlier work's results, the accuracy of our method is not bad. Also in our experiment, we found with the increase of number of path, the option price converges.

4.4 Conclusions

The use of Monte Carlo methods for pricing derivatives is crucial not only for high-dimensional options. Pricing other exotic options such as barrier, knockout options,and path dependent options also need Monte Carlo methods.

In addition, simulation methods lend themselves to the use parallel computing. There are obvious approaches in which the algorithm can be partitioned so that computations could be carried out in parallel.

In this thesis, we have present a new method for pricing multi-dimensional American options. Our algorithm is Low-discrepancy sequence based state-space partition algorithm. Our main idea lies in using low discrepancy sequence as representative state, so that we can price high dimension American options. Numerical results show that the algorithm can be applied successfully.

Appendix A

Low Discrepancy Sequence

Low discrepancy sequence is a sequence of N distributed vectors $X^1, X^2, X^3, ...$ in the m-dimensional hypercube $I^m = [0,1]^m \subset \mathbb{R}^m$. It's also called Quasi Monte Carlo sequence, or Quasi random sequence.

Now given a sequence of such vectors, if they are well distributed, the number of points included in any subset G of I^m should be roughly proportional to its volume vol(G). Given a vector $X = x_1, x_2, x_3, ..., x_m$ consider the rectangular subset G_X as defined as

$$G_x = [0, x_1) \times [0, x_m) \times \cdots \times [0, x_m)$$

which has a volume $x_1 x_2 \cdots x_N$. If we denote by $S_N(G)$ the function counting the number of points in the sequence, which are contained in a subset $G \subset I^m$, a possible definition of discrepancy is

$$D(X^1, X^2, X^3, ...X^N) = \sup_{X \in I^m} |S_N(G_X) - N x_1 x_2 \cdots x_m|$$

Actually, the Quasi random sequence term is a bit misleading, as there is no randomness at all. Some theoretical results show that low discrepancy sequences may perform better than random Monte carlo sequence obtained through a LCG. The estimation error with Monte Carlo simulation is something like $O(1/\sqrt{N})$, where

N is the number of samples. With certain low discrepancy sequences, it can be show that the error is something like $O((lnN)^m/N)$, where m is the dimension of the low discrepancy sequences.

There are 4 popular kind of low discrepancy sequences:Halton sequence, Faure sequence, Sobol sequence, and Niederreiter. sequence. Here we introduce the basic idea of Halton and Sobol sequence. Niederreiter's [14] is a comprehensive book on low discrepancy sequences.

A.1 Halton Sequence

Halton low discrepancy sequences can be generated as follows:

Step 1 Representing an integer number n in a base b, where b is a primer number:

$$n = (\cdots d_4 d_3 d_2 d_1 d_0)_b$$

Step 2 Reflecting the digits and adding a radix point to obtain a number within the unit interval:

$$h = (0.d_0 d_1 d_2 d_3 d_4 d_5 \cdots)_b$$

More formally, if we represent an integer number n as

$$n = \sum_{k=0}^{m} d_k b^k$$

the nth number in the Halton's sequence with base b is

$$h(n, b) = \sum_{k=0}^{m} d_k b^{-(k+1)}$$

Halton sequence is the simplest to generate. However, it's not the best. For Halton sequence, the points in successive dimensions are highly correlated and in high dimensions, the initial points in the Halton sequence are clustered near zero.

The same problem arises in Faure sequence. The Faure sequence is also a general s-dimensional sequence. The first dimension of the Faure sequence is in the base p. Higher dimension are permutation of the sequence of the first dimension.

Therefore, Sobol sequence is our best choice in practice.

A.2 Sobol Sequence

In the general s-dimensional Sobol sequence, all dimensions use the primer number 2 as the base. The higher dimensions are permutation of the sequence in the first dimension. The permutation depends on a set of direction numbers and the Sobol Sequence is not uniquely defined until all these direction numbers are determined.

Consider the generation of an one-dimensional sequence x^n in the $[0, 1]$ interval. To get the nth number in the sequence, consider the binary representation of the integer n:

$$n = (\cdots d_4 d_3 d_2 d_1)_2$$

The result is obtained by computing the bitwise exclusive or of the direction numbers v_i for which $b_i \neq 0$:

$$x^n = b_1 v_1 \oplus b_2 v_2 \oplus \cdots \tag{A.1}$$

If the direction numbers are chosen properly, a low-discrepancy sequence will be generated. A direction number maybe though as a binary fraction:

$$v_i = (0.v_{i1} v_{i2} v_{i3} \cdots)_2$$

or as

$$v_i = \frac{m_i}{2^i}$$

where $m_i < 2^i$ is an odd integer.

To get direction numbers, we need to consider the primitive polynomials over the field $\mathbb{Z}_2$, i.e. polynomials with binary coefficients:

$$P = x^d + a_1 x^{d-1} + \cdots + a_{d-1}x + 1, a_k \in \{0, 1\}$$

Some primitive polynomials over the field $\mathbb{Z}_2$ are listed in [16]. Now, given the primitive polynomial of degree d, the direction numbers can be generated as :

$$v_i = a_1 v_{i-1} \oplus a_2 v_{i-2} \oplus \cdots \oplus a_{d-1} v_{i-d+1} \oplus v_{i-d} \oplus [v_{i-d}/2^d].$$

This is better expressed in integer arithmetics:

$$m_i = 2a_1 m_{i-1} \oplus 2^2 a_2 m_{i-2} \oplus \cdots \oplus 2^{d-1} a_{d-1} m_{i-d+1} \oplus 2^d m_{i-d} \oplus m_{i-d}.$$

After computing the direction numbers, we could generate a Sobol sequence according to (A.1). But, an improved method was proposed in ([5]). It has been proved that the discrepancy is not changed by using the Gray code representation of n. Gray code are :

1. A Gray code is a function mapping an integer i to a corresponding binary representation $G(i)$; the function, for a given N, is one to one for $0 \leq i \leq 2^N - 1$.

2. A Gray code representation for a integer n is obtained from its binary representation by computing

$$...g_3 g_2 g_1 = (...b_3 b_2 b_1)_2 \oplus (...b_4 b_3 b_2)_2$$

3. The main feature of such code is that the code for consecutive numbers n and n+1 diff only in one position.

Using the Gray code, we may streamline generation of a Sobol sequence. Given x^n, we have

$$x^{n+1} = x^n \oplus v_c$$

where c is the index of the rightmost zero bit b_c in the binary representation of n.

Now we put it all together. First, we generate the direct in numbers. Secondly, we initialize the sequence in some way, and apply Gray code to generate the Sobol sequence. Note the longer the sequences, the more generating numbers are needed.

Appendix B

Miscellaneous Notes

B.1 Brownian Motion

If the value of a variable changes over time in an uncertain way it is said to follow a stochastic process. This process can be discrete or continuous in time (discrete time or continuous time process) and in "space" (discrete or continuous variable). Although trading in financial markets is not continuous in time (there is no trading outside business hours at exchanges) and asset price (e.g. stock prices are quoted in fixed ticks), the continuous-time, continuous-variable process is a useful model of financial asset prices for many purposes.

A Markov process is a stochastic process where only the present value of a stochastic variable is relevant for the next value. The next value is independent of the path the present value is obtained. A Brownian Motion is a particular Markov process with a mean change of 0 and variance 1. It is also called Wiener process. If a random variable X follows a Wiener process,its changes ΔX in discrete time steps Δt can be written as

$$\Delta X = \epsilon\sqrt{\Delta t}$$

where ϵ is a standard Normal random variable. If $\Delta t \to 0$, it can be rewritten as

$$dX = \epsilon\sqrt{dt}.$$

If the development of a stochastic variable S with time t can be described as a generalized Wiener process, its differential equation can be written as:

$$dS = \mu dt + \sigma dW$$

where the parameters μ and σ are constant. μ describes the drift of the process, and σ is a measure of its variation. The differential dX is a random variable drawn from a normal distribution with mean 0 and variance dt (i.e. a Wiener process). The values of dX for different times are independent.

The prices of financial assets are usually assumed to follow more general processes where the parameters can depend on S and t.

$$dS = \mu(S,t)dt + \sigma(S,t)dW$$

These processes are called Itô processes.

B.2 Cumulative Normal Distribution

$N(x)$ is the cumulative normal distribution:

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

One numerical approximation of $N(x)$ is :

$$\begin{aligned} N(x) &= 1 - N'(x)\sum_{i=1}^{5} a_i k^i, \quad \text{if}, x \geq 0 \\ N(x) &= 1 - N(-x), \quad \text{if}, x \leq 0 \end{aligned}$$

where

$$N'(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}}$$

and

$$\begin{aligned} k &= \frac{1}{1+\gamma x} \\ \gamma &= 0.2316419 \\ a_1 &= 0.319381530 \\ a_2 &= -0.356563782 \\ a_3 &= 1.781477937 \\ a_4 &= -1.821255978 \\ a_5 &= 1.330274429 \end{aligned}$$

Appendix C

Source of Program

```
%-------------------------------------------------------------
%-------------------------------------------------------------
clear
t11=cputime;
S01=80;
S02=80;
mu=0.06;
sigma=0.6;
T=0.5;
N=25;
n=5000;

r=0.06;
X=100;
m=floor(n^(1/2));
dt = T/N;
discount = exp(-r*dt);
```

```
%-------Generating Path ------------------------------------
%---S1-----------
load sob101;
Sobolsq=sob101(1:m,1:2*N);


nudt = (mu-0.5*sigma^2)*dt;
sidt = sigma*sqrt(dt);
RandMat = Sobolsq(:,1:2:2*N-1);
Increments = [nudt + sidt*norminv(RandMat)];
LogPaths = cumsum([log(S01)*ones(m,1) , Increments] , 2);
C1 = exp(LogPaths);

Increments = [nudt + sidt*randn(n,N)];
LogPaths = cumsum([log(S01)*ones(n,1) , Increments] , 2);
SPaths1 = exp(LogPaths);

%-------------------------------------------------------------
%----S2-------------

RandMat = Sobolsq(:,2:2:2*N);
Increments = [nudt + sidt*norminv(RandMat)];
LogPaths = cumsum([log(S02)*ones(m,1) , Increments] , 2);
C2 = exp(LogPaths);

Increments = [nudt + sidt*randn(n,N)];
```

```
LogPaths = cumsum([log(S01)*ones(n,1) , Increments] , 2);
SPaths2 = exp(LogPaths);

%-------------------------------------------------------------------------

discountVet = exp(-r*dt*(1:n)'); % discount rates over different time intervals

SPaths1(:,1) = []; % get rid of starting prices
C1(:,1)=[];

SPaths2(:,1) = []; % get rid of starting prices
C2(:,1)=[];
%=========================================================================


ExerciseTime = N*ones(n,1); % first set exercise time at expiration for convenienc

NN=zeros(m,N-1);
NNCindex=zeros(m,N-1);

Vt=zeros(n,N);
Yt=zeros(n,N);
Ht=zeros(n,N);
Temp2=repmat(1:N,n,1);
Yt=exp(-Temp2*dt*r).*max(0, X - min(SPaths1,SPaths2));
%Yt(:,NSteps)=exp(-r*dt*NSteps).*max(0, X - SPaths(:,NSteps));
Vt(:,N)=exp(-r*dt*N).*max(0, X - min(SPaths1(:,N),SPaths2(:,N)));
```

```
%-------------------------------------------------------------------------



for step = N-1:-1:1

    NN=zeros(m,1);
    NNCindex=zeros(m,1);

    InMoney = find(SPaths1(:,step) >0);
    XData1 = SPaths1(InMoney,step);
    XData2 = SPaths2(InMoney,step);
    LXData=length(XData1);
    Q = Vt(:,step+1);

    LXData=n;
    %Q = Vt(InMoney,step+1);
    Q = Vt(:,step+1);
    %Q = Vt;
    %-------------------------------------------------------------
    %-------Compute Continuation Value--------------------------
    %-------------------------------------------------------------


    Qsum=zeros(m,1);

    for i1=1:LXData

        r1=(XData1(i1)-C1(:,step)).^2+(XData2(i1)-C2(:,step)).^2;
```

```
        [r1_min,position] = min(r1);
        Cindex=position;

        NNCindex(i1)=Cindex;
        NN(Cindex)=NN(Cindex)+1;
    end

    for i1=1:LXData

        Cindex1=NNCindex(i1);
        Qsum(Cindex1)=Qsum(Cindex1)+Q(i1);
    end

    aa=zeros(LXData,1);

    for i1=1:LXData

        Cindex2=NNCindex(i1);
        aa(i1)=Qsum(Cindex2)/(NN(Cindex2));
    end
    aa;

    Ht(:,step)=aa;
    Vt(:,step)=max(Yt(:,step),aa);


end % for
```

```
%-----------------------------------------------------------------
%--Stopping Times----------------------------

CashFlows = max(0, X - min(SPaths1(:,1),SPaths2(:,1)));
Stopping = zeros(n,1);    %Stopping Rules
ExerciseTime = ones(n,1); % first set exercise time at time step 1 for convenience

for t = 1:N

    if t~=N
        temp1=find(Stopping==0);
        Exercise = find(Yt(temp1,t) > Ht(temp1,t));
        k = temp1(Exercise);
        CashFlows(k) = Yt(k,t);
        ExerciseTime(k) = t;
        Stopping(k)=1;
    else
        Exercise = find(Yt(temp1,t) >= Ht(temp1,t));
        k = temp1(Exercise);
        CashFlows(k) = Yt(k,t);
        ExerciseTime(k) = t;
        Stopping(k)=1;

    end %if

end %for

price = mean(CashFlows);
[price,cputime-t11]
```

Bibliography

[1] Francis A.Longstaff and Eduardo S. Schwartz. Valuing American Options by Simulation: A Simple Least-Squares Approach. *Review of Financial Studies*, Vol.14, No.1:113–147, Spring 2001.

[2] J Barraquand and D Martineau. Numerical Valuation of High Dimensional Multivariate American Securities. *Journal of Financial and Quantitive Analysis*, 30:383–405, 1995.

[3] F. Black and M. Scholes. The Pricing of options and Corporate Liabilities. *Journal of Political Economy*, 81,3:637–654, 1973.

[4] P.M. Boyle. Options: a Monte Carlo Approach. *Journal of Financial Economics*, 4:323–338, 1979.

[5] Paul Bratley and Bennett L.Fox. Algorithm 659 Implement Sobol's Qusairandom Sequence Generator. *ACM Transactions on Mathematicial Software*, Vol 14:88–100, 1988.

[6] M Broadie and P. Glasserman. Pricing American- Style securities using simulation. *Journal of Economic Dynamics and Control*, Vol 21, No 8/9:1323–1352, 1997.

[7] Mark Broadie and Paul Glasserman. A Stochastic Mesh Method for Pricing High-Dimensional American Options. PaineWebber Working Papers in Money,Economics and Finance, #PW9804, Columbia Business School, New York, 1997.

[8] L Clewlow and C Srickland. Pricing Interest Rate Exotics by Monte Carlo Simulation. *Monte Carlo: Methodologies and Applications for Pricing and Risk Management,Dupire,B.,Editor*, pages Risk Publications, London, 1998.

[9] J.C. Cox, S.A.Ross, and M.Rubinstein. Option Pricing: A Simplified Appraoch. *Journal of Financial Economics*, 7,3:229–263, 1979.

[10] Darrell Duffie. *Dynamic Asset Pricing Theory, 3rd ed.* Princeton University Press, Princeton, NJ.

[11] P. Protter E. Clément, D. Lamberton. An Analysis of a Least Square Regression Methods for American Option Pricing. *Finance and Stochastics*, 6:449–471, 2002.

[12] Michael C. Fu, Scott B.Laprise, Dilip B.Madan, Yi Su, and Rongwen Wu. Pricing American Options: A Comparison of Monte Carlo Simulation Approaches. *Journal of Computational Finance*, Vol. 4, No. 3:39–88, Spring 2001.

[13] R.C. Merton. The Theory of Rational Option Pricing. *Bell Journal of Economics and Management Science*, 4,1:141–183, 1973.

[14] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* Society for Industrial and Applied Mathematics,.

[15] Jeremy Evnine Phelim P. Bolyle and Stephen Gibbs. Pricing American Style Options Using Low Discrepancy Mesh Methods. *Mathematics and Computers in Simulations*, Forthcoming.

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition.* Cambridge University Press, 1992.

[17] L.C.G. Rogers. Monte Carlo Valuation of AmericanOptions. *Mathematical Finance*, Vol.12, No.3:271–286, July 2002.

[18] James A. Tilley. Valuing American Options in a Path Simulation Model. *Transactions of the Society of Actuaries*, 45:83–104, 1993.

[19] Sidney J. Yakowitz. Noneparametic Estimation of Markov Transition Function. *Ann. Statist*, 7(3):671–679, 1979.

[20] Sidney J. Yakowitz. A Noneparametic Markov Model for Daily River Flow. *Water Resources Research*, Vol.15, No.5:1035–1043, October 1979.

Index

Name: Sun Junhua

Degree: Master of Science

Department: Mathematics

Thesis Title: Pricing Multi-Dimension American Options by Simulation

Abstract

Applying Monte Carlo Simulation to American option is very hard and was considered impossible. The cash flow of American options not only depends on the price path of the underlying assets but also depends on the decisions of the option holder. Our algorithm is based on State-Space Partitioning Method. The main challenge in using this kind of methods is the selection of the state-space partitions. The low-discrepancy sequences, such as Sobol sequence, can fill in the space quickly in an efficient way. The algorithm we present here uses low-discrepancy sequences as "Representative State" to partition the state-space, so that we can deal with the pricing in high dimensions.

Keywords:

Monte Carlo simulation; Quasi-Monte Carlo; American option; Option pricing ; Sobol sequence; Multiple state variables.

PRICING MULTI-DIMENSION AMERICAN OPTIONS BY SIMULATION

SUN JUNHUA

NATIONAL UNIVERSITY OF SINGAPORE

2004