COPENHAGEN BUSINESS SCHOOL

MSc in Advanced Economics and Finance

Master's Thesis

Monte Carlo Methods for American Option Pricing

Copenhagen
Business School
HANDELSHØJSKOLEN

Alberto Barola

February 2013

Academic Supervisor	Jesper Lund
	Department of Finance
Number of characters	145714
Number of pages	79

Abstract

The Monte Carlo approach has proved to be a valuable and flexible computational tool in modern finance. A number of Monte Carlo simulation-based methods have been developed within the past years to address the American option pricing problem.

The aim of this thesis is to present and analyze three famous simulation algorithms for pricing American style derivatives: the stochastic tree; the stochastic mesh and the least squares method (LSM). We first present the mathematical descriptions underlying these numerical methods. We then proceed to analyze the impact of efficiency enhancement and variance reduction techniques on the algorithms performance. At the end we test the selected methods on a common set of problems in order to be able to assess the strengths and weaknesses of each approach as a function of the problem characteristics. The results are compared and discussed on the basis of estimates precision and computation time.

Overall the simulation framework seems to work considerably well in valuing American style derivative securities. When multi-dimensional problems are considered, simulation based methods seem to be the best solution to estimate prices since the general numerical procedures of finite difference and binomial trees become impractical in these specific situations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction

The valuation of early-exercisable options, or American style options, is a task of major importance in derivatives pricing because these types of instruments are found in all major financial markets including equity, commodity, foreign exchange, credit and convertible. Even though some authors presented analytical approximations[1] for certain early-exercisable options, exact closed-form solutions are in general not existent for these instruments. Therefore the primary methods for pricing such derivatives are binomial trees and finite difference. The finite difference method estimates the solution of a partial differential equation (PDE) by discretizing the solution space into a grid and then solving the PDE by recursion. The tree method also discretizes the solution space although the discretization is chosen to represent the distribution of the underlying stock price process, instead of the entire solution space. The shortcoming is that these techniques become computationally prohibitive when they are generalized to handle multiple dimensions, with computation time typically increasing exponentially with the number of state variables[2].

The use of simulation methods in security pricing may offer a number of decisive advantages. First of all the convergence rate of the Monte Carlo approach is independent of the number of state variables which makes the method computationally appealing for solving high-dimensional problems. Moreover the method is flexible with respect to the evolution of the state variables offering the opportunity to price derivative securities with more complex process dynamics. At the end path dependency features can easily be incorporated in a Monte Carlo pricing framework. On the other hand, the major drawback of simulation procedures is the difficulty in dealing with the early exercise feature of American options. In particular, standard simulation procedures are forward algorithms, that is, the trajectories of the state variables are simulated forward in time. Then, given a pre-specified exercise policy, a path price is determined for each trajectory. The average of path prices gives an unbiased estimate of the derivative security price. By contrast, pricing methods for American style derivatives are generally backward algorithms. That is to say, the optimal exercise strategy at maturity is determined and proceeding backward in time dynamic programming determines the optimal exercise strategy and the corresponding price at previous dates. The problem of using simulation methods to price American options stems from the difficulty in applying forward-based procedures to problems that necessitate a backward procedure to be solved.

[1] See Geske and Johnson (1984) and Barone-Adesi and Whaley (2012).
[2] High-dimensionality arises in pricing options on multiple underlying assets and in pricing options in models that capture many sources of risk, such as stochastic volatility, interest rates and exchange rates.

Nevertheless many different approaches have been proposed. Dynamic programming algorithms have been developed where the value of the option is computed at maturity and then, by recursively moving back in time, the price at time zero is estimated. State space partitioning uses a finite state dynamic program to approximate the value of the option. More precisely these algorithms partition the space of the underlying asset (state space) into a tractable number of cells and they compute an approximate early exercise strategy that is constant over those cells. If the partition is appropriately chosen, the approximate strategy will be close to the actual optimal strategy. The option is then estimated with backward in time procedures consistent with the approximated exercise strategy.

This thesis will concentrate and analyze three famous algorithms belonging to two alternative classes:

- **Mesh and tree methods**: proposed by Broadie and Glasserman (1996, 2004), simulate trees (or mesh) of prices. From this structure we can generate two estimates of the asset price: one biased high and the other biased low. Both estimates are asymptotically unbiased and converge to the true price.
- **Regression-Based method**: proposed by Longstaff and Schwartz (2001), uses least-squares regression on polynomials to approximate the early exercise boundary.

The overall goal is to use a set of specific pricing problems to compare and contrast the strengths and weaknesses of the three algorithms considered. In order to do so, an initial analysis of each algorithm is presented with the implementation of efficiency enhancement and variance reduction techniques. Then the algorithms are used to estimate the prices of option contracts with different characteristics and their efficiency in terms of both estimates accuracy and computation time is discussed.

Research Question

As already presented, Monte Carlo simulation is gaining importance for the solution of American option pricing problems due to its flexibility and adaptability to complicated underlying models and derivatives contracts. Extensive analysis has been performed in order to produce efficient pricing algorithms capable of properly quantifying the premium for the early exercise opportunity. However all these techniques present specific advantages and disadvantages that may influence the applicability of the models in relation to the specific characteristics of the considered problem . Our aim is to perform an in depth analysis of the three algorithms chosen in order to study their strengths and weaknesses. Then we will proceed to test them on a selected set of pricing problems to see how well they perform in each single case.

The general overall question, which will be answered throughout the whole thesis, is:

- **What are the specific strengths and weaknesses of the stochastic tree, stochastic mesh and least-squares method algorithms and how do they impact the model applicability to specific pricing problems?**

In order to answer this question several sub questions are developed throughout the thesis and can be summarized as follow:

- **Are the methods eligible for the application of efficiency enhancement or variance reduction techniques?**
- **How can the produced estimates be considered efficient in relation to both precision and computation time?**
- **How do the algorithms behave when multiple state variable problems are considered?**

Literature Review

As already discussed, the pricing of American style derivatives is typically accomplished by means of numerical methods. The two most common approaches are finite difference methods and binomial trees. Brennan and Schwartz introduced the finite difference method in 1978. Another example of the explicit finite difference method is presented more recently in Hull and White (1990). The binomial tree was first introduced by Cox, Ross and Rubenstein (1979). The method has been very successful for pricing claims contingent on a single state variable which follows a geometric Brownian motion. The procedure was later extended by Hull and White (1993) into a trinomial method. Broadie and Detemple (1996) provides a recent comparison of various (non-simulated-based) existing methods in pricing standard American call and put options written on a single underlying dividend-paying asset.

Simulation methods for asset pricing were introduced to finance by Boyle (1977). For more information on the use of Monte Carlo simulation for derivatives pricing see also Boyle, Broadie and Glasserman (1997) which discuss some of the recent applications of the method with emphasis on improvements in efficiency.

Considering American option pricing problems, several classes of algorithms have been developed. The work of Tilley (1993) provides an estimate of the American option price through the use of a bundling technique and a backward induction algorithm. The approach of Grant, Vora and Weeks (1996, 1997) is the development of sequential algorithms in which the exercise boundary is estimated first and then the option price is calculated based on the obtained optimal boundary. Ibanez and Zapatero (2004) developed a fix-point algorithm to find the critical points where the exercise payoff equals the expected holding value. Broadie and Glasserman (1997, 2004) proposed two

algorithms – one based on a non-recombining tree and the other based on a stochastic mesh – whose objective is to find efficient upper and lower bounds from simulated paths that converge asymptotically to the true price. A different class of algorithms such as the one of Tsitsiklis and Van Roy (1999) combines simulation with regression on a set of basis functions to develop option price approximations. Other examples are the works of Carriere (1996), who uses a sequential nonlinear regression algorithm on spline functions to approximate conditional expectations for the holding values at early exercise points; and Longstaff and Schwartz (2001) that use least-squares regression on polynomials to approximate the early exercise boundary. At the end, Barraquand and Martineau (1995) developed a pricing algorithm that uses a state aggregation technique where a subset of the full state definition is used to make the early exercise decision. A comparison of some simulation based methods with numerical results is presented in the paper of Fu et al. (2001).

Thesis Structure

The rest of the thesis is organized as follows. In Part I the theoretical framework concerning derivatives pricing is presented. Moreover, the American option pricing problem is specified and the numerical procedures of binomial tree and finite difference are described. In Part II the Monte Carlo method is presented as well as the stochastic tree, stochastic mesh and LSM algorithms. A description of the efficiency enhancement techniques is also provided. The numerical analysis is developed in Part III where the algorithms are first analyzed singularly to study the impact of efficiency enhancement techniques and then they are implemented in three different pricing problems to compare their results. In Part IV the results of the overall analysis are discussed and concluding remarks are made.

Part I: Principles of Derivatives Pricing

In this part we will describe some important principles of the derivatives pricing theory[3], especially those that determine the applicability of computer simulation to pricing problems. Three concepts are of particular importance:

1. If a derivative security can be perfectly replicated through trading in other assets then the price of the derivative security is equivalent to the cost of the replicating trading strategy.
2. Discounted asset prices are martingales under a probability measure associated with the choice of a discount factor. Prices are expectations of discounted payoffs under such a martingale measure.
3. In a complete market, any payoff can be replicated through a trading strategy, and the martingale measure associated with a discount factor is unique.

The first principle gives an intuition about what the price of a derivative security has to be but it says little about how this price might be evaluated. The problem boils down to find a hedging strategy and then determine the cost of implementing this strategy. The second principle explains how to represent prices as expectations which lend themselves to evaluation through computer simulation. To do so it is necessary to describe the asset price dynamics under a risk adjusted probability measure. The third principle may be viewed as describing conditions under which the price of a derivative security is determined by the prices of other assets so that the first and second principles apply.

In the subsequent sections we will present the problem of valuing American style derivatives and illustrate two among the most common methods used in practice.

1.1 The Black and Scholes Model

In the early 1970s Fisher Black, Myrion Scholes and Robert Merton made a major breakthrough in the pricing of stock options[4]. The model revolutionized the way that traders price and hedge options. Today the Black and Scholes model is one of the basic building blocks of derivatives theory. Before presenting the model we state the assumptions needed for the derivation:

- There is a risk-free asset that earns a constant rate of return equal to r per unit of time.
- Trading takes place continuously.
- There are no transaction costs.

[3] A proper development of the theory and of the tools needed to state precisely its main results is beyond the scope of this study therefore we assume familiarity with the basic ideas of mathematical finance and refer the reader to Neftci (2000) and Musiela and Rutkowski (2011) for further background.
[4] See Black and Scholes (1973) and Merton (1973).

- There are no dividends on the underlying asset.

Let's start by calling $V(S_t, t)$ the time t price of a call option written on a non-dividend paying stock. The option has a strike price K and a time to maturity T. We then denote by Π the value of a portfolio of one long option position and a short position of some quantity delta Δ of the underlying asset, thus

$$\Pi = V(S_t, t) - \Delta S_t \,. \tag{1.1}$$

Further we assume that the underlying asset follows a geometric Brownian motion process

$$dS = \mu S dt + \sigma S dz$$

where z represents a standard Brownian motion. Over time, the change in value of the portfolio Π is due partly to the the change in the option value and partly to the change in the underlying asset

$$d\Pi = dV - \Delta dS \,.$$

Notice that Δ has not changed over the time step since it is impossible to anticipate the change in S. Using Itô lemma we can express the change in the option as

$$dV = \frac{\partial V}{\partial t} dt + \frac{\partial V}{\partial S} dS + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} dt \,.$$

Therefore the portfolio change may be expressed as

$$d\Pi = \frac{\partial V}{\partial t} dt + \frac{\partial V}{\partial S} dS + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} dt - \Delta dS \,. \tag{1.2}$$

In equation (1.2) it is possible to distinguish two types of terms, the deterministic and the random. The deterministic terms are those in dt and the random terms are those in dS. These random terms represent the risk in our portfolio. By choosing appropriately Δ it is possible to reduce or even eliminate the risk. Since the random terms in (1.2) are

$$\left(\frac{\partial V}{\partial S} - \Delta\right) dS$$

by choosing

$$\Delta = \frac{\partial V}{\partial S} \tag{1.3}$$

all the randomness in the portfolio is reduced to zero.
By choosing Δ as in (1.3) the change in the portfolio Π simplifies to

$$d\Pi = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right) dt \tag{1.4}$$

which is completely riskless. Therefore the portfolio change must be equalized to the change for risk free securities

$$d\Pi = r\Pi\, dt \tag{1.5}$$

where r is the risk free rate. This is an example of the no arbitrage principle and if equality (1.5) is not respected there would be free profit opportunities in the market[5].

By substituting (1.1), (1.3) and (1.4) into (1.5) we find that

$$\left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right) dt = r\left(V - S\frac{\partial V}{\partial S}\right) dt$$

and dividing by dt and rearranging we get

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{1.6}$$

which is the Black and Scholes partial differential equation. The solution of equation (1.6) leads to the call e put option prices

$$\begin{aligned} c(S(t),t) &= S(t)N(d_1) - Ke^{-r(T-t)}N(d_2) \\ p(S(t),t) &= Ke^{-r(T-t)}N(-d_2) - S(t)N(-d_1) \end{aligned} \tag{1.7}$$

where

$$d_1 = \frac{\ln(S(t)/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \qquad d_2 = d_1 - \sigma\sqrt{T-t}$$

K is the strike price of the option, T is the expiration date and $N(\cdot) =$ normal cumulative distribution function.

By analyzing equations (1.7) it is possible to notice that they do not depend on the drift μ since any dependence on the drifted components dropped out at the same time as we eliminated the dS components of the portfolio. The economic argument for this is that since we can perfectly hedge

[5] Imagine for a moment that the return on the portfolio was greater than r. In this case the investors could borrow from the bank at the risk free rate r, invest in the risk free option/stock portfolio and earn a sure profit. The reverse strategy would apply with the same result if the portfolio return was lower than the risk free rate. At this point we say that the actions of the investors buying and selling to exploit arbitrage opportunities will move the price of the option in the direction that eliminates arbitrages.

the option with the underlying we should not be rewarded for taking unnecessary risk; only the risk-free rate of return is in the equations. Moreover the Black and Scholes argument may be used to replicate an option just by continuously buying and selling the underlying stock and the risk free asset. This leads to the idea of complete market. In a complete market an option can be replicated by trading in the underlying and the risk free asset thus making the option redundant. The completeness of the market together with the absence of arbitrage permit to price any contingent claim.

1.2 Arbitrage and Risk Neutral Pricing

Section 1.1 outlined an argument showing how the existence of a trading strategy that replicates the payoffs of a derivative security determines its price. This argument leads to the derivation of a partial differential equation describing the derivative price dynamics. However if the asset price dynamics are sufficiently complex, a PDE characterizing the derivative price may be difficult to solve or may even fail to exist[6]. In these settings computer simulation is likely to be useful. In order to be applied, simulation requires a more convenient representation of derivatives prices. The following sections will develop the representation of the derivative price as an expectation of random objects that can be simulated.

1.2.1 Arbitrage and Stochastic Discount Factors

Consider an economy with d assets whose prices $S_i(t), i = 1, \dots, d$ are described by a system of stochastic differential equations

$$\frac{dS_i(t)}{S_i(t)} = \mu_i(S(t), t)dt + \sigma_i(S(t), t)^T dz^o(t)\,, \tag{1.8}$$

where $z^o$is a k-dimensional Brownian motion, each σ_i taking values in $\Re^k$ and each u_i scalar valued. We assume that u_i and σ_i are deterministic functions of the current state $S(t) = \big(S_1(t), \dots, S_d(t)\big)^T$and time t. We denote with P_o the probability measure under which these dynamics are specified. It describes real world probabilities and the empirical dynamics of asset prices.

A portfolio is characterized by a vector θ with θ_i representing the number of units invested in the ith asset. The value of the portfolio at time t is $\theta^T S(t)$. A trading strategy is characterized by a

[6] This is the case for derivatives whose payoffs depend on the path of the underlying asset and not simply its terminal values or derivatives whose payoffs depend on multiple underlying assets.

stochastic process $\theta(t)$ of portfolio vectors[7]. A trading strategy is self-financing if it satisfies the following equality

$$\theta(t)^T S(t) - \theta(0)^T S(0) = \int_0^t \theta(u)^T dS(u) \quad for\ all\ t\ .$$

The left side represents the change in the portfolio value from time 0 to time t and the right hand side gives the gains from continuous trading over the time interval. Thus neither profits are withdrawn from the portfolio neither new funds are added.

A self-financing trading strategy $\theta(t)$ is called an arbitrage if either of the following conditions holds for a fixed time t:

(i) $\theta(0)^T S(0) < 0$ and $P_o(\theta(t)^T S(t) \geq 0) = 1$;

(ii) $\theta(0)^T S(0) = 0$, $P_o(\theta(t)^T S(t) \geq 0)$=1, and $P_o(\theta(t)^T S(t) > 0) > 0$.

In the first condition the trading strategy transforms a negative initial investment into a nonnegative final payoff with probability 1. In the second condition a zero initial investment gives a nonnegative final payoff that is positive with positive probability. Each of these cases corresponds to an opportunity to create something out of nothing and is incompatible with economic equilibrium.

Now denote with $V(t)$ an attainable price process if $V(t) = \theta(t)^T S(t)$ for some self-financing trading strategy θ[8]. With the strictly positive process $Z(t)$ we indicate a stochastic discount factor if the ratio $V(t)/Z(t)$ is a martingale for every attainable price process $V(t)$; that is

$$\frac{V(t)}{Z(t)} = E_o\left[\frac{V(T)}{Z(T)}|\mathcal{F}_t\right] \quad for\ t < T \tag{1.9}$$

where E_o is the expectation taken under the probability P_o and $\mathcal{F}_t$ represents the history of the Brownian motion z^o up to time t. $Z(t)$ is further assumed to be $\mathcal{F}_t$ measurable meaning that the value of $Z(t)$ is determined by the history of the Brownian motion up to time t. Given this, it is possible to rewrite equation (1.9) as

$$V(t) = E_o\left[V(T)\frac{Z(t)}{Z(T)}|\mathcal{F}_t\right]. \tag{1.10}$$

[7] If we fix the portfolio holdings at $\theta(t)$ over the interval $[t, t+h]$ the change in the value of the portfolio is given by $\theta(t)^T[S(t+h) - S(t)]$. Thus in continuous time we may describe the gains from trading over $[0, t]$ through the stochastic integral

$$\int_0^t \theta(u)^T dS(u)\ .$$

[8] For example a European option can be replicated by trading in the underlying asset precisely if its payoff at maturity T coincides with the value $V(T)$ of some attainable price process at time T.

Equation (1.10) presents the true meaning of stochastic discount factor: the price $V(t)$ is just the expected discuounted value of the price $V(T)$ if we discount using $Z(t)/Z(T)$. Any constant multiple of a stochastic discount factor is itself a stochastic discount factor so by using the normalization $Z(0) \equiv 1$ equation (1.10) can be written as

$$V(0) = E_o\left[\frac{V(T)}{Z(T)}\right]. \tag{1.11}$$

If we imagine that $V(t)$ represents the time t price of a call option on the ith stock with maturity T and strike price K then $V(T) = (S_i(T) - K)^+$. According to equation (1.11) the terminal value $V(T)$ determines the initial value $V(0)$ through stochastic discounting. However it is important to highlight how the initial price $V(0)$ differs from the standard expected payoff $E_o[V(T)]$: first, because of time value of money investors will not value equally payoffs received today with payoffs to be received in the future; second, in a world where investors are risk-averse risky payoffs should be more heavily discounted in valuing a security and this task may not be accomplished by a deterministic discount factor.

At the end the existence of a stochastic discount factor rules out arbitrage. If θ is a self financing trading strategy, the process $\theta(t)^T S(t)$ is an attainable price process and the ratio $\theta(t)^T S(t)/Z(t)$ must be a martingale. In particular

$$\theta(0)^T S(0) = E_o\left[\frac{\theta(T)S(T)}{Z(T)}\right].$$

Now recalling that Z is nonnegative we can compare this result with the two conditions (i) and (ii). If $\theta(t)^T S(t)$ is almost surely positive it is impossible for $\theta(0)^T S(0)$ to be negative; if $\theta(t)^T S(t)$ is positive with positive probability and almost surely nonnegative, then $\theta(0)^T S(0) = 0$ is impossible. Thus we may conclude that the existence of a stochastic discount factor precludes arbitrage in the market. It is less obvious that the converse also holds: under a variety of technical conditions on asset price dynamics and trading strategies it has been shown that the absence of arbitrage implies the existence of a stochastic discount factor[9].

1.2.2 Risk Neutral Pricing

Imagine that among the d assets described in (1.8) there is one that is risk free and call it $\beta(t)$. Its parameters will be $\sigma_i = 0$ and $\mu_i = r$. Its price dynamics will be described by the equation $d\beta(t)/\beta(t) = rdt$ with solution $\beta(t) = \beta(0)e^{rt}$; $\beta(0)$ is considered to be fixed at 1. The process

[9] This equivalence is often called the Fundamental Theorem of Asset Pricing. The presentation of the Theorem is beyond the scope of this study however the interested reader may refer to Duffie (2008) and Musiela and Rutkowski (2011) for background references.

$\beta(t)$ is an attainable price process[10] and if the market admits a stochastic discount factor $Z(t)$ the process $\beta(t)/Z(t)$ is a martingale and its initial value is $\beta(0)/Z(0) = 1$. Moreover the martingale is positive because both $\beta(t)$ and $Z(t)$ are positive.

Any positive martingale with an initial value of 1 defines a change of probability measure. For each interval $[0, T]$ the process $\beta(t)/Z(t)$ defines a new measure P_β through the Radon-Nikodym derivative

$$\left(\frac{dP_\beta}{dP_o}\right)_t = \frac{\beta(t)}{Z(t)}, \quad 0 \le t \le T\,;$$

this implies (cf. Appendix A) that for any event $A \epsilon \mathcal{F}_t$

$$P_\beta(A) = E_o\left[1_A \cdot \left(\frac{dP_\beta}{dP_o}\right)_t\right] = E_o\left[1_A \cdot \frac{\beta(t)}{Z(t)}\right],$$

where 1_A is the indicator of the event A. The new measure defines a new expectation operator

$$E_\beta[X] = E_o\left[X\frac{\beta(t)}{Z(t)}\right] \tag{1.12}$$

which is true for any nonnegative X measurable with respect to $\mathcal{F}_t$. The probability measure P_β is called the risk neutral measure and is equivalent to P_o in the sense that they both agree about which events are impossible[11]. The risk neutral measure is a particular choice of equivalent martingale measure. Give the results in (1.12) it is possible to rewrite equation (1.11) in the following way

$$V(0) = E_\beta\left[\frac{V(T)}{\beta(T)}\right] = e^{-rT}E_\beta[V(T)]\,. \tag{1.13}$$

The transformation in (1.13) is the core of derivatives pricing by simulation. The current price is expressed as the expected present value of the terminal payoffs discounted at the risk free rate r rather than through the stochastic discount factor Z. Since the expectation is taken under $P_{\beta,}$ to estimate the expectation it is necessary to simulate under P_β rather than P_o. These points are crucial to the implementation of simulation because : first, the dynamics of $Z(t)$ are generally unknown and difficult to model; second, the asset price dynamics are more easily described under the risk-neutral measure than under the objective probability measure.

[10] It corresponds of investing an initial amount of 1 in the risk free asset and continuously reinvesting all the gains in this single asset.
[11] $P_\beta(A) = 0$ if and only if $P_o(A) = 0$.

With respect to the second point we provide some further explanations. Equation (1.13) generalizes to

$$V(t) = E_\beta \left[V(T) \frac{\beta(t)}{\beta(T)} | \mathcal{F}_t \right], \quad t < T,$$

with $V(t)$ being an attainable price process. Specifically since every stock price $S_i(t)$ is an attainable price process, each ratio $S_i(t)/\beta(t)$ is a martingale under P_β. Price dynamics under the risk neutral measure imply specific dynamics that make the ratios $S_i(t)/\beta(t)$ martingales. In particular if the asset prices in (1.8) could be expressed as

$$\frac{dS_i(t)}{S_i(t)} = rdt + \sigma_i(S(t), t)^T dz(t), \tag{1.14}$$

with z a standard k-dimensional Brownian motion under the probability measure P_β, then

$$d\left(\frac{S_i(t)}{\beta(t)}\right) = \left(\frac{S_i(t)}{\beta(t)}\right) \sigma_i(S(t), t)^T dz(t),$$

so $S_i(t)/\beta(t)$ would indeed be a martingale under P_β. The model in (1.14) is way more simple to be specified than the one in (1.8) because all drifts are set equal to the risk free rate r.[12] This explains the name risk neutral because in a world of risk neutral investors the rate of return on risky assets would be the same as the risk free rate. By comparing equation (1.14) with (1.8) it is possible to notice that the two are consistent if

$$dz(t) = dz^o(t) + \nu(t)dt \quad \text{for some } \nu \text{ satisfying } u_i = r + \sigma_i^T \nu, \quad i = 1, \dots, d, \tag{1.15}$$

because by making this substitution in (1.14) yields exactly (1.8)[13]. The condition in (1.15) states that the objective and risk neutral measures are related through a change of drift in the driving Brownian motion[14]. In particular the diffusion terms in (1.14) are equal to the real world ones. It ensures that the coefficients required to describe the dynamics of asset prices under the risk neutral measure can be estimated from data observed under the real world measure.

We now briefly summarize the pricing of derivative securities through the risk neutral measure with simulation. This procedure will be at the core of Monte Carlo methods described in Part II and

[12] The potentially difficult drifts in (1.8) are therefore irrelevant to the asset price dynamics under the risk neutral measure.

[13] $\frac{dS_i(t)}{S_i(t)} = rdt + \sigma_i(S(t), t)^T [dz^o(t) + \nu(t)dt] = (r + \sigma_i(S(t), t)^T \nu(t))dt + \sigma_i(S(t), t)^T dz^o(t)$
$= \mu_i(S(t), t)dt + \sigma_i(S(t), t)^T dz^o(t).$

[14] According to the Girsanov Theorem any measure equivalent to P_o must be related to P_o in this way. For a brief exposition of the Theorem the reader may see Appendix A. For background references on the Theorem the reader my refer to Neftci (2000) and Musiela and Rutkowski (2011).

implemented in the numerical analysis of Part III. Consider a derivative security with a payoff at maturity T specified through a function f of the price of the underlying asset as in the case of a standard call or put. To price the derivative we model the dynamics of the stock prices under the risk neutral measure typically through the choice of the drift. This ensures that discounted asset prices are martingales. The derivative price is then given by $E_{\beta}[e^{-rT}f(S(T))]$. To evaluate this expectation we simulate according to risk neutral dynamics the paths of the underlying assets over the interval $[0,T]$. On each path we calculate the discounted payoff $e^{-rT}f(S(T))$; at the end by taking average across paths we produce an estimate of the derivative price.

1.3 Pricing American Options

1.3.1 Problem Formulation

American type derivative securities contain implicit or explicit options which can be exercised before expiration date if desired.

Consider an American option written on a stock. The option can be exercised at any time between the present date $t = 0$ and time to maturity $t = T$. The holder will exercise this option if he thinks that it is better to do so, rather than waiting until expiration. The right of early exercise may have some additional value and pricing methods for American style securities must take this into account.

Let $\tau \in [0,T]$ represents the early exercise date. Given the information set at time t I_t we will be able to tell whether the option has already been exercised or not. With American type securities τ is in general random. The option price may be described by the function $F(S_t,t)$. The underlying stock price is assumed to behaves in continuous time as a geometric Brownian motion

$$dS = \mu S dt + \sigma S dz\,.$$

The price of the derivative can be expressed using the risk neutral measure Q but this time the security holder does not have to wait until expiration T to exercise the option. He will exercise the option as soon as it is more profitable to do so than waiting until expiration. In other word by waiting until expiration the asset will be worth at time t

$$F(S_t,t) = E_t^Q\left[e^{-r(T-t)}max(S_T - K, 0)\right].$$

If the option is exercised early its value would be

$$F(S_t,t) = E_t^Q\left[e^{-r(\tau-t)}max(S_\tau - K, 0)\right].$$

Therefore the pricing problem at time $t = 0$ becomes the calculation of

$$F(S_0, 0) = \sup_{\tau \in [0,T]} \left[E_0^Q [e^{-r\tau} max(S_\tau - K, 0)] \right].$$

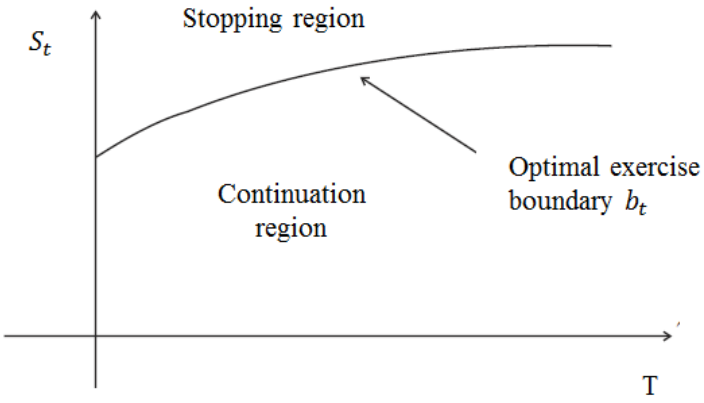
This supremum is achieved by an optimal stopping time τ^* that has the form

$$\tau^* = inf\{t \geq 0 : S_t \geq b_t^*\},$$

for some optimal exercise boundary b^* . An Example is provided in Figure 1.1 For additional literature on the subject the reader may refer to Neftci (2000) or Glasserman (2004).

Since the asset price at which the early exercise is optimal isn't known in advance it has to be found as part of the solution problem. For this reason an analytic solution to the problem does not exist except for very special cases[15]. Thus the general practice is to implement numerical procedures to generate accurate estimates of the option price. Two of the most widely used techniques are described next.

Figure 1.1: Exercise Boundary for an American Call Option



The option is exercised at τ^*, that is the first time the stock price reaches the boundary.

1.3.2 Binomial Tree

A useful and very popular technique for pricing stock options involves constructing a binomial tree. This is a diagram that represents different possible paths that might be followed by the underlying stock price over the life of the option.

[15] With a finite number of exercise opportunities the true value of the option can be obtained from the formula in Geske and Johnson (1984). For the case where the asset pays exactly one known discrete dividend during the lifetime of the option a solution for the option price is provided by the formula of Roll (1977) and Whaley (1981).

Start by dividing the life of the option into a large number of intervals n of equal length Δt. At each period, the stock may move from the initial value S to either up (Su) with probability q or down (Sd) with probability $1-q$. The interest rate is constant and is denoted by r[16].

Considering only one period to expiration, an hypothetical call option written on the asset will have at expiration the payoffs $C_u = \max\ [0, Su - K]$ if the stock price goes to Su and $C_d = \max\ [0, Sd - K]$ if the stocks decreases to Sd. If a portfolio can be constructed with Δ shares of the stock and an amount D invested at the risk free rate it will have a cost of $\Delta S + D$ and a future value of either $\Delta Su + rD$ or $\Delta Sd + rD$. It is possible to select the values of Δ and D in order to replicate the final payoffs of the option independently of the underlying outcome. These are[17]

$$\Delta = \frac{C_u - C_d}{(u - d)S} \qquad D = \frac{uC_d - dC_u}{(u-d)r}\ . \tag{1.16}$$

To preclude the existence of arbitrage opportunities in the market the call price C cannot be less that the current value of the hedging portfolio $\Delta S + D$. Therefore it must be true that

$$C = \Delta S + D = \frac{C_u - C_d}{(u-d)} + \frac{uC_d - dC_u}{(u-d)r} = \frac{[pC_u + (1-p)C_d]}{r}$$
$$p \equiv \frac{r-d}{u-d} \qquad 1 - p \equiv \frac{u-r}{u-d}. \tag{1.17}$$

Equation (1.17) represents the value of the call option one period before expiration[18]. The two most important features of the formula are that the real world probabilities q do not appear and the call price does not depend on investors' attitudes towards risk. This means that whatever different subjective probabilities about future stock movements investors have they will agree on the relationship between C, S, u, d and r. Therefore the formula would work whether investors are risk-averse or risk-preferring. Finally p is always greater than zero and less than one showing the basic properties of a probability measure. Specifically p is the value q would have in equilibrium if investors were risk-neutral[19]. Hence the call can be interpreted as the expectation of its discounted future values in a risk-neutral world (Cox et al. 1979). The procedure just presented can be reiterated several times for

[16] The condition $u > r > d$ is required in order to avoid riskless arbitrage opportunities involving only the stock and the riskless borrowing-lending.

[17] The values are obtained by solving the system of equations

$$\Delta uS + rD = C_u$$
$$\Delta dS + rD = C_d\ .$$

[18] In the present case with no dividends the option value will always be greater than $S - K$ as long as the interest rate is positive (Cox et al. 1979). Early exercise therefore is never optimal.

[19] The expected return on the stock would then be the risk free rate so

$$q(uS) + (1-q)(dS) = rS$$
$$q = \frac{r-d}{u-d} = p\ .$$

pricing call options with more than one period to expiration. Starting from the maturity date and then going backward toward time zero the steps are always the same: 1) construct a hedging portfolio that perfectly replicates the payoffs in the future 2) set the option price equal to the value of the hedging portfolio.[20] A complete example is shown in the Appendix B.

When the number of intervals considered increases and their length shrinks it is important to analyze the behavior of the interval-dependent variables *p, u, d*. With respect to the definition of *p, u* and *d* an important assumption about the behavior of the stochastic process of the underlying stock has to be done. In this section, an analysis is presented when it is assumed that the stochastic process is continuous[21] as $n \to \infty$. The parameters *p, u* and *d* must be chosen in a way to determine the right values of the stock expected return and variance at the end of each interval Δt. Given the assumption of risk neutrality the one period expected return of the stock is equal to the risk free rate $r\Delta t$ and the expected future stock price is $Se^{r\Delta t}$. So

$$Se^{r\Delta t} = pSu + (1-p)Sd$$

and

$$e^{r\Delta t} = pu + (1-p)d. \tag{1.18}$$

The stochastic process assumed implies a one period variance of $\sigma^2\Delta t$. Therefore it follows that[22]

$$pu^2 + (1-p)d^2 - [pu + (1-p)d]^2 = \sigma^2\Delta t$$

by plugging in *p* from (1.18)

$$e^{r\Delta t}(u+d) - ud - e^{2r\Delta t} = \sigma^2\Delta t\,. \tag{1.19}$$

Together with equations (1.18) and (1.19) Cox et al. (1979) introduced a third condition in order to be able to calculate *p, u, d* which is[23]

$$u = \frac{1}{d}\,. \tag{1.20}$$

The three conditions (1.18), (1.19) and (1.20) imply that[24]:

[20] There is an important difference with more than one period before expiration. In order to lock in the riskless profit by selling an overpriced option and buying the hedging portfolio an investor will have to readjust his position in the hedging portfolio every period. In this way he avoids to incur losses if the value of the call is still in disequilibrium after one period and the position cannot be liquidated.

[21] A geometric Brownian motion.Cox et al. (1979) presented also the case where the underlying stochastic process is a jump process.

[22] The variance of a variable X is equal to $E(X^2) - [E(X)]^2$.

[23] The additional assumption is introduced in order to obtain a tree that recombines, so that the effect of a down movement followed by an up movement is the same as the effect of an up movement, followed by a down movement.

$$p = \frac{a-d}{u-d} \quad (1.21) \qquad u = e^{\sigma\sqrt{\Delta t}} \quad (1.22) \qquad d = e^{-\sigma\sqrt{\Delta t}} \quad (1.23)$$

and

$$a = e^{r\Delta t}\,. \quad (1.24)$$

Finally Cox et al. (1979) provided an exhaustive demonstration that the binomial model for stock prices with the stated parameters includes the lognormal distribution as a limiting case thus proving that the binomial method and the Black and Scholes formula should coincide as the number of time interval goes to infinity.

1.3.2.1 Options on a Stock Paying Continuous Dividends

Consider a stock paying a continuous dividend yield δ. The total return from capital gains and dividends in a risk neutral world is r. Since the dividends provide a return of δ the capital gain return has to be $r - \delta$. If the stock starts at S_0, its expected value after on time step of length Δt must be $S_0 e^{(r-\delta)\Delta t}$ which means

$$pS_0u + (1-p)S_0d = S_0e^{(r-\delta)\Delta t}$$

so that

$$p = \frac{e^{(r-\delta)\Delta t} - d}{u - d}\,.$$

As in the case of an option on a non-dividend paying asset it is possible to match the volatility by setting $u = e^{\sigma\sqrt{\Delta t}}$ and $u = \frac{1}{d}$. This means that it is possible to use equations (1.20), (1.21), (1.22) and (1.24) except that equation (1.24) becomes $a = e^{(r-\delta)\Delta t}$.

1.3.2.2 Valuing American Options Using the Binomial Tree

European options can be exercised only at maturity but an American option can be exercised anytime. The method used to evaluate an American style derivative is very similar to that for the European option except that it is necessary to incorporate the early exercise feature. In the case of American options the value of the option at any node in the binomial tree is the greater between the value calculated by backward induction (like in the European option case) and the payoff from early exercise. Specifically at time t and node i

[24] Equations (1.21) and (1.24) satisfy exactly condition (1.18). Moreover according to the Taylor expansion where the terms of order higher than Δt are ignored, equation (1.22) implies that $u = 1 + \sigma\sqrt{\Delta t} + \frac{1}{2}\sigma^2\Delta t$ and equation (1.23) implies $d = 1 - \sigma\sqrt{\Delta t} + \frac{1}{2}\sigma^2\Delta t$, with $e^{r\Delta t} = 1 + r\Delta t$ and $e^{2r\Delta t} = 1 + 2r\Delta t$. It follows that when the terms of order higher than Δt are ignored the equations (1.22) and (1.23) satisfy condition (1.19). Hull (2009).

$$C_t^i = max\left[max\left(S_t^i - K, 0\right), \left[pC_{t+1}^u + (1-p)C_{t+1}^d\right]/r\right].$$

1.3.3 Finite Difference

Finite difference methods value a derivative security by solving the differential equation that the derivative satisfies. The differential equation is converted into a set of difference equations which are solved recursively.

Consider an option price f that depends on a single stochastic variable S which follows the specific stochastic process

$$dS = \mu S dt + \sigma S dz\,.$$

The differential equation that the option must satisfy is the well-known Black and Scholes (1973) differential equation[25]:

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf\,. \tag{1.25}$$

To implement the explicit finite difference method we first define a small time change and a small change in the stock price as Δt and ΔS respectvely. Then a grid is constructed to estimate the value of f when the stock price is $0, \Delta S, 2\Delta S, \dots, S_{max}$ ($M+1$ steps in total) and the time is equal to $0, \Delta t, 2\Delta t, \dots, T$ ($N+1$ steps in total). In the dimensions, the expiration date T determines the maximum time allowed. In the stock price state, limited liability defines the lower absolute value as 0 and S_{max} is the upper bound generally defined by derivatives conditions[26]. Figure A3 in Appendix C provides an example of finite difference grid. The point (i, j) in the grid refers to time $i\Delta t$ and stock price $j\Delta S$ or simply (S_j). f_{ij} is the option price at the point (i, j). The partial derivatives of f at node $(i-1, j)$ can be approximated as follows:

$$\frac{\partial f}{\partial S} = \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S} \tag{1.26}$$

$$\frac{\partial^2 f}{\partial S^2} = \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{\Delta S^2} \tag{1.27}$$

$$\frac{\partial f}{\partial t} = \frac{f_{i,j} - f_{i-1,j}}{\Delta t}\,. \tag{1.28}$$

Substituting (1.26), (1.27), and (1.28) into (1.25) gives

[25] The Black and Scholes differential equation is a particular case of the general differential equation derived by Garman (1976) that a derivative security which depends on a single stochastic variable should satisfy:

$$\frac{\partial f}{\partial t} + (\mu - \lambda\sigma)S\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

where λ is the market price of risk of S. For a detailed derivation and analysis the reader may refer to the specific Garman paper.

[26] For puts S_{max} is the stock price above which $\partial f/\partial S$ approaches zero, and for calls S_{max} is the stock price above which $\partial f/\partial S$ approaches one. The convergence rate to either one or zero of the hedge ratio may be determined from the Black-Scholes European solution. A useful rule of thumb is that the stock price will be about one-and-a-half times the exercise price for these derivatives conditions to be satisfied. Geske and Shastri (1985).

$$f_{i-1,j} = a_{j-1}f_{i,j-1} + a_j f_{ij} + a_{j+1}f_{i,j+1} \tag{1.29}$$

where

$$a_{j-1} = \frac{1}{1+r\Delta t}\left[-\frac{rS_j\Delta t}{2\Delta S} + \frac{1}{2}\frac{S_j^2\sigma^2\Delta t}{\Delta S^2}\right] \qquad a_j = \frac{1}{1+r\Delta t}\left[1 - \frac{S_j^2\sigma^2\Delta t}{\Delta S^2}\right]$$

$$a_{j+1} = \frac{1}{1+r\Delta t}\left[\frac{rS_j\Delta t}{2\Delta S} + \frac{1}{2}\frac{S_j^2\sigma^2\Delta t}{\Delta S^2}\right].$$

Equation (1.29) is the core of the explicit finite difference method. It relates the derivative price at t_{i-1} to three possible values of it at time t_i. The option value is know with certainty at expiration T[27]. The price at time *t* can be obtained by using (1.29) recursively to work backward from time T to time t[28]. The explicit finite difference method has several features in common with a trinomial lattice. In fact through some arrangements equation (1.29) may be rewritten as:

$$f_{i-1,j} = \frac{1}{1+r\Delta t}\left[p_{j,j-1}f_{i,j-1} + p_{jj}f_{ij} + p_{j,j+1}f_{i,j+1}\right] \tag{1.30}$$

where

$$p_{j,j-1} = -\frac{rS_j\Delta t}{2\Delta S} + \frac{1}{2}\frac{S_j^2\sigma^2\Delta t}{\Delta S^2} \qquad p_{jj} = 1 - \frac{S_j^2\sigma^2\Delta t}{\Delta S^2} \qquad p_{j,j+1} = \frac{rS_j\Delta t}{2\Delta S} + \frac{1}{2}\frac{S_j^2\sigma^2\Delta t}{\Delta S^2}. \tag{1.31}$$

The values of $p_{j,j-1}$, $p_{jj}$and $p_{j,j+1}$ can be interpreted as the probabilities of moving from node S_j to nodes S_{j-1}, S_j and S_{j+1} respictively during a time step Δt[29] (Figure A3). In this way equation (1.30) gives the option price at time t_i as its expected value at time t_{i+1} in a risk neutral world discounted back at the risk free rate of interest.

1.3.3.1 Valuing American Options Using Finite Differences

In order to value an American style derivative the possibility of early exercise must be included in the procedure. This may be incorporated by checking at each point in the grid if the continuation value $f_{i-1,j}$ is lower than the payoff from early exercise $j\Delta S - K$[30]. If that is the case early exercise is

[27] For a call option the payoff at maturity is equal to $(S_T - K, 0)^+$ therefore $f_{N,j} = \left(S_j - K, 0\right)^+ \; j = 0, \dots, M$. When the stock price is zero the call value is zero, hence $f_{i,0} = 0 \;\; i = 0, \dots N$. On the other hand, assuming that S_{max} has been chosen to be high enough to have the option deep in the money, we have $f_{i,M} = (S_M - K) \;\; i = 0, \dots, N$. These three conditions define the value of the option in the three borders of the grid where $S = 0, S = S_{max}$ and $t = T$.

[28] The presentation here assumes that f is a European-style derivative security. The arguments can be easily extended to other situation as we will see shortly.

[29] They all sum to one and for the time interval Δt they predict a drift rate for the stock price equal to $rS\Delta t$. This is the expected return in a risk neutral world.

[30] The specific example refers to a call option.

convenient and $f_{i-1,j}$ is replaced by $j\Delta S - K$. Proceeding recursively from maturity backward in time it is possible to value the option at each point in the grid.

1.3.3.2 The Transformation of Variables

Brennan and Schwartz (1978) suggested that it is more efficient to use $\ln(S)$ rather than S as the underlying variable when finite difference methods are applied. This is because when σ is constant, the instantaneous standard deviation of $\ln(S)$ is constant, therefore the standard deviation of $\Delta \ln(S)$ is independent of S_t. Using the log transformation, the partial differential equation that the derivative price should satisfy becomes:

$$\frac{\partial f}{\partial t} + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial f}{\partial \ln(S)} + \frac{1}{2}\sigma^2 \frac{\partial^2 f}{\partial \ln(S)^2} = rf \, .$$

The state variable $\ln(S)$ can be modeled in the same way as was done for S with a finite difference grid. In this way the probabilities in (1.31) become:

$$p_{j,j-1} = -\frac{\Delta t}{2\Delta \ln(S)}(r - \sigma^2/2) + \frac{1}{2}\frac{\sigma^2 \Delta t}{\Delta \ln(S)^2}$$
$$p_{j,j+1} = \frac{\Delta t}{2\Delta \ln(S)}(r - \sigma^2/2) + \frac{1}{2}\frac{\sigma^2 \Delta t}{\Delta \ln(S)^2} \quad p_{j,j} = 1 - \frac{\Delta t}{\Delta \ln(S)^2}\sigma^2 \, . \tag{1.32}$$

An important feature of the new probabilities in (1.32) is that they are independent from the stock price at time t and they are constant at each node[31].

1.3.3.3 Stability and Convergence of the Estimator

When the explicit finite difference method is utilized it is important to be sure that the derivative price converges to its true value as the time steps and the steps in the state variable tend to zero. From a theorem in Ames (1992), a sufficient condition for convergence is that the probability coefficients $p_{j,j-1}$, $p_{j,j}$ and $p_{j,j+1}$ be all positives as Δt and ΔS ($\Delta \ln(S)) \to 0$. According to Brennan and Schwartz (1978) this is a sufficient condition also for stability[32]. For the log transformed finite different method the non-negativity condition is satisfied if[33]

[31] By using the log transformation together with an appropriate choice of the parameter defining the stock increments it is possible to calculate the option price in the same way as the trinomial extension of the Cox, Ross and Rubenstein (1979) binomial lattice. See Hull (2009) for a complete presentation.
[32] See McCracken and Dorn (1965). A stable procedure is one where the results are relatively insensitive to round-off and other small computational errors Hull and White (1990).
[33] See Brennan and Shwartz (1978).

$$\Delta t \leq \frac{\sigma^2}{\left(r - \frac{1}{2}\sigma^2\right)^2} \qquad \Delta \ln(S) \leq \frac{\sigma^2}{|r - \frac{1}{2}\sigma^2|}.$$

Hull (2009) provided evidence that the numerically most efficient choice of state variable step is $\Delta \ln(S) = \sigma\sqrt{3\Delta t}$.

Therefore by using the log transformed finite different method one is able to produce a consistent estimate of the derivative price. This is not always the case for the standard method. This is due to the dependence of the probability coefficients on the stock price level at time t. It may happen that for some very high levels of S_t some probabilities are negatives as well as the resulting derivative estimates. This may create abnormal oscillations in the algorithm and the consequent lack of convergence. At the end it is important to mention that there exists another finite difference method, the so called "implicit finite different method. The procedure is very similar to the explicit method except that the implicit procedure relates $f_{i+1,j}$ to $f_{i,j-1}$, $f_{i,j}$ and $f_{i,j+1}$. For an exhaustive presentation of this method the reader may refer to Brennan and Schwartz (1978).

PART II: Monte Carlo Methods for American Option Pricing

2.1 The Monte Carlo Method

In this section the Monte Carlo Method is described[34]. In an experiment the probability of an event is its volume relative to that of a universe of possible alternative outcomes. Monte Carlo uses this identity in estimating the probability by calculating the volume of a set. This means sampling randomly from the universe of possible outcomes and taking the random observations that fall in a given set as an estimate of the set's volume. The law of large numbers predicts that the estimate converges to its true value as the sample size increases. In finance the Monte Carlo method is particularly useful because of its ability to estimate integrals. Considering the integral[35]:

$$\int_A f(x)g(x)dx = \alpha$$

where $f(x)$ is an arbitrary function and $g(x)$ is a probability density function with

$$\int_A g(x)dx = 1\,.$$

If it is possible to sample values (x_i) from the density function $g(x)$, by evaluating the function $f(x)$ at n of these random points and averaging them it is possible to produce the Monte Carlo estimate:

$$\hat{\alpha}_n = \frac{1}{n}\sum_{i=1}^{n} f(x_i)\,. \tag{2.1}$$

The standard deviation of the estimate is given by

$$s_f = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(f(x_i) - \hat{\alpha}_n)^2}\,.$$

[34] The reader may refer to Hammersley and Handscomb (1965) for a complete and excellent exposition of the Monte Carlo method. Additional useful references are Meyer (1957) and Shreider (1966).
[35] A denotes the range of integration.

If f is integrable over A then the strong law of large numbers ensures that $\hat{\alpha}_n \to \alpha$ with probability 1 as $n \to \infty$. Moreover the error $\hat{\alpha}_n - \alpha$ is approximately normally distributed with mean 0 and standard deviation $s_f/\sqrt{n}$. The form of the standard error $s_f/\sqrt{n}$ is an important feature of the Monte Carlo method. The denominator implies a squared root convergence rate of the estimator which is rather slow. However the $O\left(n^{-\frac{1}{2}}\right)$ convergence rate holds even when the estimate is an integral over d dimensions thus making the Monte Carlo method a competitive tool in the estimation of these particular integrals[36].

In Part I it has been showed that under some circumstances the price of a derivative security can be represented as an expected value. In some cases if we were to write the expectation as an integral we would find that its dimension is large depending on how many stochastic processes are involved in the problem. These are all suitable scenarios where the Monte Carlo method becomes attractive.

For the most part there is nothing that can be done to overcome the rather slow rate of convergence characteristic of Monte Carlo. However the precision of the estimator may be improved by reducing the standard deviation s_f. The techniques used for this task are known as variance reduction techniques and a pair of them is discussed next.

2.1.1 Variance Reduction Techniques

In this section we first discuss the role of variance reduction in meeting the broader objective of improving the computational efficiency of Monte Carlo simulations. We then discuss two specific examples of variance reduction techniques: control variate and antithetic variates and illustrate their applications in pricing problems[37].

2.1.1.1 Variance Reduction and Efficiency Improvement

Suppose we want to estimate the derivative price α. Equation (2.1) provides the estimator of α based on *n* Monte Carlo replications. The standard error of the estimator is $s_f/\sqrt{n}$. This implies that decreasing the standard deviation by a factor of 10 while leaving everything else unchanged is equivalent to increasing the number of simulations by a factor of 100 in terms of error reduction.

Suppose now that we have at our disposal two unbiased[38] Monte Carlo estimators $\hat{\alpha}^1$ and $\hat{\alpha}^2$ with standard deviations σ_1 and σ_2 respectively but with $\sigma_1 < \sigma_2$. From the previous observation, given a number *n* of replications, $\hat{\alpha}_1$ will present a lower error than $\hat{\alpha}_2$. However this analysis oversimplifies

[36] As a quick comparison the error in a product trapezoidal rule in *d* dimensions is $O\left(n^{-2/d}\right)$.

[37] Despite the numerous variance reduction techniques available for derivatives pricing simulation only the control variate and antithetic variates are going to be analyzed in this thesis. This is mainly due to their wide application in financial pricing problems. For an analysis of alternative variance reduction techniques the reader may refer to Glasserman (2004) and Charnes (2000).

[38] $E[\hat{\alpha}_1] = E[\hat{\alpha}_2] = \alpha$.

the comparison because it does not take into account eventual differences in the computational time required by the estimators. Smaller variance is not a sufficient ground for preferring one estimator over the other if its calculation is more time-consuming. A more exhaustive procedure is required to compare estimators with different computational requirements as well as different variances.

Suppose that $\tau_i \;\; i = 1, 2$ is the time required to generate one replication of $\hat{\alpha}_1$ and $\hat{\alpha}_2$ respectively. With a given time budget t the number of replications of $\hat{\alpha}_i$ is t/τ_i. The two estimators with computing time t are:

$$\frac{\tau_1}{t}\sum_{i=1}^{t/\tau_1} \hat{\alpha}_i^{(1)} \qquad \text{and} \qquad \frac{\tau_2}{t}\sum_{i=1}^{t/\tau_2} \hat{\alpha}_i^{(2)}$$

and for large enough t they are normally distributed with mean α and standard deviations

$$\sigma_1\sqrt{\frac{\tau_1}{t}} \qquad \text{and} \qquad \sigma_2\sqrt{\frac{\tau_2}{t}}\,.$$

Boyle et al. (1997) provided the condition that should be satisfied in order to prefer the first estimator over the second:

$$\sigma_1^2\tau_1 < \sigma_2^2\tau_2\,. \tag{2.2}$$

According to (2.2) it is reasonable to take the product of variance and work per run as a measure of efficiency[39].

2.1.1.2 Control Variate

The method of control variate is among the most widely applicable, easiest to use and effective of the variance reduction techniques.[40] It generally exploits information about the errors in estimates of known quantities to reduce the error in an estimate of an unknown quantity.

Suppose again we want to estimate the derivative price α. The Monte Carlo estimator from n independent and identically distributed replications $\alpha_1, \dots, \alpha_n$[41] is $\hat{\alpha} = (\alpha_1 + \cdots + \alpha_n)/n$.

Imagine now, that for each replication it is possible to calculate another output X_i along with α_i. The pairs $(X_i, \alpha_i)\; i = 1, \dots, n$ are i.i.d. and suppose that the expectation $E[X]$ is known. Thus for any fixed b it is possible to calculate

$$\alpha_i(b) = \alpha_i - b(X_i - E[X])$$

[39] The principle expressed in (2.2) dates at least to Hammersley and Handscomb (1965). The interested reader may refer to Glynn and Whitt (1992) for a further in-depth examination.
[40] For a general background on control variate see Lavenberg and Welch (1981). The earliest application of this technique to option pricing is Boyle (1977). For examples of control variate applications in finance see Broadie and Glasserman (1996), Duan (2006) and Kemma and Vorst (1990).
[41] α_i may be the discounted payoff of a derivative security on the ith simulated path.

for each replication i. The control variate Monte Carlo estimator would then be

$$\hat{\alpha}(b) = \hat{\alpha} - b\big(\hat{X} - E[X]\big) = \frac{1}{n}\sum_{i=1}^{n}\big(\alpha_i - b(X_i - E[X])\big) \tag{2.3}$$

and the observed error $\hat{X} - E[X]$ is used to control the estimate $E[\alpha]$. The control variate estimator (2.3) is unbiased[42] and consistent[43].

The variance of each replication $\alpha_i(b)$ is

$$Var[\alpha_i(b)] = Var[\alpha_i - b(X_i - E[X])] = \sigma_\alpha^2 - 2b\sigma_\alpha\sigma_X\rho + b^2\sigma_X^2 \equiv \sigma^2(b) \tag{2.4}$$

where $\sigma_X^2 = Var[X]$, $\sigma_\alpha^2 = Var[\alpha]$ and ρ is the correlation between X and α. Comparing the control variate estimator variance $\sigma^2(b)/n$ and the ordinary Monte Carlo estimator variance σ^2/n it is possible to infer that the control variate estimator $\hat{\alpha}(b)$ has smaller variance than the standard estimator $\hat{\alpha}$ if $b^2\sigma_X < 2b\sigma_\alpha\rho$.

The optimal coefficient b^* that minimizes the variance in (2.4) is

$$b^* = \frac{\sigma_\alpha}{\sigma_X}\rho = \frac{Cov[X,\alpha]}{Var[X]}. \tag{2.5}$$

Plugging this value into (2.4) and arranging it is possible to find the ratio of the variance of the optimally controlled estimator to the one of the uncontrolled estimator

$$\frac{Var\big[\hat{\alpha} - b^*\big(\hat{X} - E[X]\big)\big]}{Var[\hat{\alpha}]} = 1 - \rho^2. \tag{2.6}$$

From equation (2.6) it is possible to observe some important features of the method:

- With the optimal coefficient b^*, the effectiveness of a control variate is determined by the size of the correlation between α and X. Moreover the sign of the correlation is irrelevant.
- If the computational effort required per replication is roughly the same with and without the control variate then equation (2.6) measures the computational improvement from the use of a control variate. Specifically $n/(1-\rho^2)$ replications of α_i are necessary to achieve the same variance as *n* replications of the control variate estimator.

[42] $E[\hat{\alpha}(b)] = E\big[\hat{\alpha} - b\big(\hat{X} - E[X]\big)\big] = E[\hat{\alpha}] = E[\alpha]$. Glasserman (2004).

[43] With probability 1

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\alpha_i(b) = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\big(\alpha_i - b(X_i - E[X])\big) = E[\alpha - b(X - E[X])] = E[\alpha] = E$$

Glasserman (2004).

- The variance reduction factor $1/(1-\rho^2)$ increases very sharply as $|\rho|$ approaches 1 and it drops off quickly as $|\rho|$ decreases away from 1.

In practice since $E[\alpha]$ is unknown it is likely that also σ_α and ρ are unknown thus creating a problem in the calculation of the optimal coefficient b^*. However it is still possible to get the most of the benefit of a control variate by using an estimate of b^*. Replacing the population parameters in (2.5) with their sample counterparts yields the estimate[44]

$$\hat{b}_n = \frac{\sum_{i=1}^{n}(X_i - \hat{X})(\alpha_i - \hat{\alpha})}{\sum_{i=1}^{n}(X_i - \hat{X})^2}. \tag{2.7}$$

The expression in (2.7) is the slope of the least-squares regression line through the points $(X_i, \alpha_i)\ i = 1, \dots, n$ which can be calculated easily.

2.1.1.3 Antithetic Variates

The method of antithetic variates is one of the simplest and widely used techniques in financial pricing problems[45] . It attempts to reduce the estimator variance by introducing negative dependence between pairs of replications.

The core observation is that if a random variable U is uniformly distributed over [0, 1] then $1-U$ is too. Thus if we generate paths using as inputs $U_1, \dots, U_n$ it is possible to generate additional paths using $1-U_1, \dots, 1-U_n$ without changing the law of the simulated process. The combinations of variables $(U_i, 1-U_i)$ constitute antithetic pairs in the sense that generally a large value of one is followed by a small value of the other.

These observations extend to other distributions through the inverse transform method[46]. Specifically, in a simulation of stochastic processes based on independent standard normal random variables, antithetic pairs may be constructed by pairing a sequence $Z_1, \dots, Z_n$ e of i.i.d. $N(0,1)$ variables with the sequence $-Z_1, \dots, -Z_n$ of i.i.d. $N(0,1)$ variables. If the $Z_i s$ are used to simulate the increments of a Brownian path the $-Z_i s$ simulate the increments of the reflection of the path about the origin.

[44] Dividing numerator and denominator by *n* and applying the strong law of large numbers shows that $\hat{b}_n \to b^*$ with probability 1.

[45] For a more general formulation of antithetic sampling see Glasserman (2004) and Hammersley and Handscomb (1965). Boyle (1977) is an early application in finance. Other works on antithetic variates include Rubinstein et al. (1985) and Hammersley and Morton (1956).

[46] $F^{-1}(U)$ and $F^{-1}(1-U)$ both have the same distribution but are antithetic to each other because F^{-1} is monotone. For a distribution symmetric about the origin $F^{-1}(1-u)$ and $F^{-1}(u)$ have the same magnitude but opposite signs. Glasserman (2004).

To analyze the variance reduction produced by the method imagine we have to estimate an expectation $E[Y]$ and that using antithetic sampling we produce a series of pairs of observations $(Y_1, \tilde{Y}_1), \dots, (Y_n, \tilde{Y}_n)$. The procedure presents the following characteristics:

- the pairs $(Y_1, \tilde{Y}_1), \dots, (Y_n, \tilde{Y}_n)$ are i.i.d.;
- For each i, Y_i and $\tilde{Y}_i$ have the same distribution, though they are not independent.

The antithetic variates estimator is defined as

$$\hat{Y}_{AV} = \frac{1}{2n}\left(\sum_{i=1}^{n} Y_i + \sum_{i=1}^{n} \tilde{Y}_i\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i + \tilde{Y}_i}{2}\right). \tag{2.8}$$

The central limit theorem implies that

$$\frac{\hat{Y}_{AV} - E[Y]}{\sigma_{AV}/\sqrt{n}} \Rightarrow N(0,1)$$

with

$$\sigma_{AV}^2 = Var\left[\frac{Y_i + \tilde{Y}_i}{2}\right].$$

As for the case of the Monte Carlo estimator the limit distribution continues to hold if we replace the population standard deviation σ_{AV} with the sample standard deviation of the n values[47]. This justify the construction of a $1 - \delta$ confidence interval of the form[48]

$$\hat{Y}_{AV} \pm z_{\delta/2}\frac{s_{AV}}{\sqrt{n}}.$$

Before comparing the antithetic variates estimator with the Monte Carlo one we make the assumption that the computational time required to simulate a pair $(Y_i, \tilde{Y}_i)$ is approximately twice the effort required to simulate a single observation Y_i[49]. Following this assumption it is reasonable to compare the variance of $\hat{Y}_{AV}$ with the variance of a Monte Carlo estimator based on $2n$ independent replications. In particular antithetic variates reduce variance if:

[47] $s_{AV} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \hat{Y}_{AV})^2}$.

[48] $1 - \Phi(z_{\delta/2}) = \delta/2$.

[49] In this case the computation saving from flipping the signs on the generated sequence of random variables $Z_1, \dots, Z_n$ rather than generating new variables is completely ignored. This is appropriate if the computational time for the random variable generation represents only a small fraction of the time required to compute an estimate Y_i.

$$Var\left[\hat{Y}_{AV}\right] < Var\left[\frac{1}{2n}\sum_{i=1}^{2n} Y_i\right]$$

or

$$Var[Y_i + \tilde{Y}_i] < 2Var[Y_i]$$

$$2Var[Y_i] + 2Cov\left[Y_i, \tilde{Y}_i\right] < 2Var[Y_i]$$

$$Cov\left[Y_i, \tilde{Y}_i\right] < 0\,.$$

The condition requires that the negative dependence of the input random variables produces negative covariance between the estimates of two paired replications. A sufficient condition ensuring this is monotonicity of the mapping function from inputs to outputs defined in the algorithm.[50] Therefore if $Y = f(Z_1, \dots, Z_d)$ for some increasing function f then $\tilde{Y} = f(-Z_1, \dots, -Z_d)$ is a decreasing function of $(Z_1, \dots, Z_d)$. This argument can be adapted to show that the method of antithetic variates increases efficiency in pricing options that depend monotonically on inputs (e.g. European, American or Asian options).

2.2 Monte Carlo Algorithms for American Option Pricing

This section describes several methods to deal with American option pricing problems using simulation and presents their strengths and weaknesses. The methods differ in the restrictions they impose on the number of exercise opportunities, the information they require about the underlying processes and the extent to which they aim to compute the exact price or produce just a reasonable approximation.

Any general simulation method for pricing American style derivatives requires substantial computational effort and certainly is the case for the methods discussed in this section.

An important topic that will be discussed is the analysis of the sources of high and low bias that affect all methods for pricing American options by simulation. High bias results from using information about the future in exercise decisions, and this is the result of applying backward induction to the simulated paths. Low bias is the direct result of following a sub-optimal exercise strategy. Some methods mix the two sources of bias, but we will see that by separating them it is often possible to produce a pair of estimates containing the optimal value.

[50] More precisely if the simulation inputs are independent random variables $Z_1, \dots, Z_m$, Y is an increasing function of these inputs and $\tilde{Y}$ is a decreasing one then $E[Y\tilde{Y}] \leq E[Y]E[\tilde{Y}]$, i.e. $Cov[Y\tilde{Y}] = E[Y\tilde{Y}] - E[Y]E[\tilde{Y}] \leq 0$.

In discussing simulation methods for pricing American options we restrict ourselves to options that can be exercised only at fixed set of exercise opportunities $t_1 < t_2 < \cdots < t_m$. In some cases such a restriction is part of the option contract (Bermudan options); in others it is known in advance that exercise is suboptimal at all but a finite set of dates, such as dividend payment dates. One may alternatively view the restriction to a finite set of exercise dates as an approximation to a contract allowing continuous exercise, in which case one would want to consider the effect of letting m increase to infinity. Because even the finite-exercise problem poses a significant challenge to Monte Carlo we focus exclusively on this case.

2.2.1 Stochastic Tree Method of Broadie and Glasserman

The major difficulty in valuing assets with early exercise features is the need to estimate an optimal exercise policy. Generally, pricing methods for these assets are backward algorithms. Starting from maturity where the optimal exercise policy is known the algorithm proceeds backward in time and calculates the optimal exercise strategy and the corresponding price via dynamic programming. Broadie and Glasserman (1997) argued that there can be no general method for producing unbiased simulation estimators of American option values[51]. For this reason they suggest two estimates of the asset price based on random samples of future state trajectories and increasingly refined approximations to optimal exercise decisions. Both estimates are biased, one high and the other low, however they are asymptotically unbiased and converge to the true price. A combination of two biased estimators can be almost as effective as a single unbiased one. Imagine that $\widehat{\Theta}(b)$ and $\hat{\theta}(b)$ are each sample means of *n* independent replications for each value of a simulation parameter b and they are respectively biased high and low

$$E\left[\widehat{\Theta}(\mathrm{b})\right] \geq \mathrm{V}_0 \geq \mathrm{E}\left[\hat{\theta}(\mathrm{b})\right]. \tag{2.9}$$

If for some halfwidth $H_n(b)$,

$$\hat{\theta}(b) \pm H_n(b) \tag{2.10}$$

is a valid 95% confidence interval for $\mathrm{E}[\widehat{\Theta}(\mathrm{b})]$ and if

[51] The American option pricing problem consists in finding $C = \max_\tau E[e^{-r\tau}(S_\tau - K)^+]$ over all stopping times $\tau \leq T$. If the optimal stopping policy were known the path estimate would have been $e^{-r\tau}(S_\tau - K)^+$. Since the optimal policy is not known and must be determined via the simulation, the path estimate would be $\max_{i=1,\ldots,m} e^{-rt_i}\left(S_{t_i} - K\right)^+$. This path estimate tends to overestimate the option value $\max_{i=1,\ldots,m} e^{-rt_i}\left(S_{t_i} - K\right)^+ \geq e^{-r\tau}(S_\tau - K)^+$. For an illustration of this phenomenon one must consider the case where following the optimal but unknown exercise strategy, the option would be exercised at maturity since the path never entered the optimal exercise region but the perfect foresight strategy would achieve a higher path value by exercising at time where the stock price is at path maximum.

$$\hat{\theta}(b) \pm L_n(b) \tag{2.11}$$

is a valid 95% confidence interval for $E[\hat{\theta}(b)]$, then by using the lower limit of the low estimator and the upper limit of the high estimator it is possible to construct the confidence interval

$$\left(\hat{\theta}(b) - L_n(b), \hat{\Theta}(b) + H_n(b)\right) \tag{2.12}$$

containing the unknown value V_0 with probability at least 90%. Moreover the confidence interval may be shrunk to the point V_0 in the limit as the computational effort grows because the inequalities in (2.9) become equalities as $b \to \infty$ and the interval halfwidths $H_n(b)$ and $L_n(b)$ go to zero as $n \to \infty$. Thus combining the two estimates it is possible to obtain a confidence interval for the asset price.

The main shortcoming of the stochastic tree is that its computational requirements grow exponentially in the number of exercise dates *m*, so the method is best applicable when *m* is small.

2.2.1.1 Constructing the Tree

The stochastic tree method is based on simulating a tree of paths of the underlying Markov chain $X_0, X_1, \dots, X_m$ (Glasserman (2004)). A branching parameter $b \geq 2$ that determines the number of branches per node must be determined. Starting at the initial node b independent successor states $X_1^1, \dots, X_1^b$ are generated all following the law of X_1. Then from each X_1^i additional b independent successors $X_2^{i1}, \dots, X_2^{ib}$ are generated from the conditional law of X_2 given $X_1 = X_1^i$. The procedure is reiterated until the last node of the tree.

Figure A4 in Appendix D is an example of stochastic tree with $m = 2$ and $b = 3$, the connections between the nodes indicate the dependence structure of the stock prices. At the *m*th time step there are b^m nodes and this is why the computational cost grows exponentially with the number of the time steps. A generic node in the tree at time step t is denoted by $X_t^{j_1 j_2 \dots j_i}$ (Glasserman (2004))[52]. It should be noted that the nodes at a fixed time appear according to the order in which they are generated and not according to their node values as would be the case in a lattice method[53].

2.2.1.2 High Estimator

The high estimator is defined as the option value estimated using a dynamic programming algorithm to the simulated stochastic tree. The option value is known with certainty at the terminal date. Then, at each prior date, the option value is calculated as the maximum of the immediate exercise

[52] The superscript indicates that the node is reached by following the j_1th branch out of X_0, the j_2th branch out of the next node, and so on.

[53] In a nonrecombining tree as well as in a binomial lattice the placement of the nodes is deterministic; see, for example Heat et al. (1990).

value and the expectation of the succeeding discounted option values. The estimator is finally equal to the estimated option value at the initial node. More formally:

$$\Theta_T^{i_1 \dots i_T} = h_T(X_T^{i_1 \dots i_T}) \tag{2.13}$$

and

$$\Theta_t^{i_1 \dots i_t} = \max\left\{ h_t\left(X_t^{i_1 \dots i_t}\right), \frac{1}{b}\sum_{j=1}^{b} e^{-R_{t+1}^{i_1 \dots i_t j}} \Theta_{t+1}^{i_1 \dots i_t j} \right\} \quad for \quad t = 0, \dots, T-1 \tag{2.14}$$

where $h_t(s)$ is the payoff from exercise at time t in state s and e^{-R_t} is the discount factor from t to $t-1$. An example of the calculation of the high estimator is given in Figure A5 in Appendix D.

We refer to this as the high estimator because it gives an estimate of the true option price which is biased upward. This can be deducted from the fact that the simulated tree does not perfectly represent the distribution of stock prices. Therefore if at some node future stock prices are too high, the dynamic programming algorithm may choose not to exercise and receive a value higher than the 'optimal' decision to exercise. On the other hand, if future prices are too low, the algorithm may exercise immediately even when the optimal decision would be to not exercise. In synthesis the algorithm takes advantage of knowledge of future to overestimate the option value. An induction argument may be used to demonstrate the effective bias of the estimator[54]

$$E\left[\Theta_t^{i_1 \dots i_t} | X_t^{i_1 \dots i_t}\right] \geq C\left(X_t^{i_1 \dots i_t}\right). \tag{2.15}$$

Using (2.14) it is possible to derive

$$\begin{aligned} E\left[\Theta_t^{i_1 \dots i_t} | X_t^{i_1 \dots i_t}\right] &= E\left[\max\left\{ h_t\left(X_t^{i_1 \dots i_t}\right), \frac{1}{b}\sum_{j=1}^{b} e^{-R_{t+1}^{i_1 \dots i_t j}} \Theta_{t+1}^{i_1 \dots i_t j} \right\} | X_t^{i_1 \dots i_t}\right] \\ &\geq \max\left\{ h_t\left(X_t^{i_1 \dots i_t}\right), E\left[\frac{1}{b}\sum_{j=1}^{b} e^{-R_{t+1}^{i_1 \dots i_t j}} \Theta_{t+1}^{i_1 \dots i_t j} | X_t^{i_1 \dots i_t}\right]\right\} \\ &= \max\left\{ h_t\left(X_t^{i_1 \dots i_t}\right), E\left[e^{-R_{t+1}^{i_1 \dots i_t j}} \Theta_{t+1}^{i_1 \dots i_t 1} | X_t^{i_1 \dots i_t}\right]\right\} \\ &\geq max\left\{ h_t\left(X_t^{i_1 \dots i_t}\right), E\left[e^{-R_{t+1}^{i_1 \dots i_t j}} C_{t+1}\left(X_{t+1}^{i_1 \dots i_t 1}\right) | X_t^{i_1 \dots i_t}\right]\right\} \end{aligned}$$

[54] Equation (2.15) holds with equality at every terminal node because of equation (2.13) and the fact that $C_T(X_T^{i_1 \dots i_T}) = h_T(X_T^{i_1 \dots i_T})$. The following steps will demonstrate that if (2.15) holds at time $t+1$ it will hold also at time t.

$$= C_t\left(X_t^{i_1 \dots i_t}\right).$$

The first passage uses (2.14), the second is an application of Jensen's inequality, the third step stems from the fact that the b successors of each node are conditionally i.i.d., the fourth step applies the induction hypothesis at $t+1$and the last step is just the true value of the option at time t. Although biased, the high estimator is consistent, that is $E[\Theta_0] \to C_0(X_0)$ as the branching parameter b goes to infinity. This result is formalized in Theorem 1 of Broadie and Glasserman (1997)[55].

A simple way to compute the upper limit of the confidence interval in (2.12) is to fix the branching parameter *b* and to replicate the random tree *n* times. In this way calling $\overline{\Theta}_0(n,b)$ the mean of the *n* replications of Θ_0 and $std_\Theta(n,b)$ their sample standard deviation it is possible to calculate an asymptotically valid[56] $1-\delta$ confidence interval[57] for $\mathrm{E}[\Theta_0]$

$$\overline{\Theta}_0(n,b) \pm z_{\delta/2}\frac{std_\Theta(n,b)}{\sqrt{n}}.$$

2.2.1.3 Low Estimator

The high bias of the estimator described in the previous section may be attributed to its use of the same information in deciding whether to exercise as well as in estimating the continuation value. In this way the continuation value is based on successor nodes so it is unfairly peeking into the future in making the decision. To remove the source of bias the branches at each node are separated into two sets. The first set of branches (Y_1^i) is used for the exercise decision, and the second set (Y_2^i) is used to estimate the continuation value, if necessary. The resulting estimator[58] would be:

$$\tilde{v}\begin{cases} a, & if\ \bar{Y}_1 \le a \\ \bar{Y}_2, & if\ \bar{Y}_1 > a \end{cases},$$

its expectation is

$$E[\tilde{v}] = P(\bar{Y}_1 \le a)a + \big(1 - P(\bar{Y}_1 \le a)\big)E[Y] \le max(a, E[Y])$$

so the estimator is indeed biased low even though consistent[59].

To intuitively understand the source of bias one should consider the time just prior to expiration where the decision is based on unbiased information from the maturity date. With a finite sample,

[55] For a formal proof the reader may refer to Appendix A of Broadie and Glasserman (1997).
[56] For large n.
[57] In the formula $z_{\delta/2}$ denotes the $1-\delta/2$ quantile of the standard normal distribution.
[58] In the formula a is considered to be the immediate exercise value and $\bar{Y}_1$ and $\bar{Y}_2$ are the sample means of the two sets.
[59] If $a \neq E[Y]$ then $P(\bar{Y}_1 \le a) \to \mathbf{1}\{E[Y] < a\}$ and $E[\tilde{v}] \to max(a, E[Y])$ as the number of replications used to calculate $\bar{Y}_1$ increases. If the number of replications used to calculate $\bar{Y}_2$ also increases then $\tilde{v} \to max(a, E[Y])$. Glasserman (2004).

there is a positive probability of inferring a suboptimal decision. In this case the value assigned to this node will be an unbiased estimate of the lower value associated with the incorrect decision. The estimator is a weighted average of an unbiased estimate (based on the correct decision) and an estimate which is biased low (based on the incorrect decision). The net effect is a low biased estimate.

Broadie and Glasserman (1997) used a slightly different estimator. At each node, one branch was used to estimate the continuation value and the other $b-1$ branches were used for the exercise decision. The process was reiterated b times always changing the branch chosen for the continuation value. The b values obtained were then averaged to determine the option value estimate at the node. Formally:

$$\theta_T^{i_1 \dots i_T} = h_T\left(X_T^{i_1 \dots i_T}\right) \tag{2.16}$$

$$\eta_t^{i_1 \dots i_t j} = \begin{cases} h_t\left(X_t^{i_1 \dots i_t}\right) & if\ h_t\left(X_t^{i_1 \dots i_t}\right) \geq \dfrac{1}{b-1}\displaystyle\sum_{\substack{i=1 \\ i \neq j}}^{b} e^{-R_{t+1}^{i_1 \dots i_t i}}\, \theta_{t+1}^{i_1 \dots i_t i} \\ e^{-R_{t+1}^{j}}\, \theta_{t+1}^{i_1 \dots i_t j} & if\ h_t\left(X_t^{i_1 \dots i_t}\right) < \dfrac{1}{b-1}\displaystyle\sum_{\substack{i=1 \\ i \neq j}}^{b} e^{-R_{t+1}^{i_1 \dots i_t i}}\, \theta_{t+1}^{i_1 \dots i_t i} \end{cases} \qquad for\ j = 1, \dots, b \tag{2.17}$$

$$\theta_t^{i_1 \dots i_t} = \frac{1}{b}\sum_{j=1}^{b} \eta_t^{i_1 \dots i_t j} \qquad for\ t = 0, \dots, T-1\,. \tag{2.18}$$

An example of the calculation of the low estimator is illustrated in Figure A6 in Appendix D.

Theorem 3 of Broadie and Glasserman (1997) establishes the convergence in probability and in norm of θ_0 to the true value $C_0(X_0)$ and the Theorem 4 establishes the low bias of the estimator[60]. Using n independent replications of the random tree it is possible to construct a $1-\delta$ confidence interval for $E[\theta_0]$

$$\bar{\theta}_0(n,b) \pm z_{\delta/2}\frac{std_\theta(n,b)}{\sqrt{n}}.$$

Combining this result with the one previously calculated for the high estimator we get

$$\left(\bar{\theta}_0(n,b) - z_{\delta/2}\frac{std_\theta(n,b)}{\sqrt{n}}, \bar{\Theta}_0(n,b) + z_{\delta/2}\frac{std_\Theta(n,b)}{\sqrt{n}}\right)$$

[60] These generalization are subordinated to the conditions that both the b_1 branches used to determine the exercise decision and the b_2 branches used to evaluate the resulting payoff go to infinity.

which is a $1-\delta$ confidence interval for the true option price $C_0(X_0)$. At the end since $E[\Theta_0]$ and $E[\theta_0]$ both approach $C_0(X_0)$ as the branching parameter increases it is possible to make the confidence interval as tight as we want by increasing *n* and *b*.

2.2.1.4 Potential Enhancements of the Random Tree Method

Broadie et al. (1997) suggest variance reduction techniques to improve the efficiency of the estimators as well as pruning techniques to reduce the computational burden of the method.

2.2.1.4.1 Pruning

Pricing European style derivatives is easier than pricing their American counterparts. This is because no optimization is involved in the European price, and so an analytical formula exists. It is therefore convenient to exploit information from the European case in pricing the American option. In particular the European price can be used for pruning i.e., decreasing the number of nodes in the simulated tree.

2.2.1.4.2 Pruning at the Last Step

Imagine we consider the penultimate exercise opportunity $T-1$. The optimal decision depends on which is greater between the immediate exercise value $h_{T-1}\left(X_{T-1}^{i_1\dots i_{T-1}}\right)$ and the continuation value $g_{T-1}\left(X_{T-1}^{i_1\dots i_{T-1}}\right)$[61]. But at this specific time the continuation value is just equal to the price of a European option initiated at time $T-1$, maturing at time T and with an initial stock price of $X_{T-1}^{i_1\dots i_{T-1}}$ which we denote by $l_T\left(X_{T-1}^{i_1\dots i_{T-1}}\right)$. Computing this value directly is fast end eliminates the need to branch at the penultimate node. The low and high estimators are both set to $max\left\{h_{T-1}\left(X_{T-1}^{i_1\dots i_{T-1}}\right), l_T\left(X_{T-1}^{i_1\dots i_{T-1}}\right)\right\}$ and the number of nodes required to simulate the whole stochastic tree reduces to b^{m-1}.

2.2.1.4.3 Antithetic Branching

Antithetic branching represents a variance reduction technique that can be used to reduce the standard errors of the estimators. In order for it to be implemented it is necessary to divide the branches emanating from each node into two equal sets. The stock prices for the first set are simulated in the usual procedure, while the second half is obtained by changing the signs of all normal random

[61] This is implicitly estimated by the two estimators θ and Θ at each node.

variables in the first set. These steps are repeated at each following node[62] (see Figure A7 in Appendix D).

For its implementation this method requires the branching parameter b to be even and divisible into two equal sets of size $b/2$. Imagine that $X_{t+1}^{i_1 \dots i_t j}$ is the j^{th} sample value, $j = 1, \dots, b/2$, which is simulated using the standard normal random variable $Z_t^{i_1 \dots i_t j}$. The other samples values for $j = b/2 + 1, \dots, b$ are simulated using $Z_t^{i_1 \dots i_t j} = -Z_t^{i_1 \dots i_t j - b/2}$. The advantage of using antithetic branching lies in the possibility of balancing low stock prices with high stock prices thus reducing the bias of the estimators. Moreover it reduces the computation time required by diminishing the number of simulated random variables by half.

This technique does not produce any change for the high estimator which is calculated using (2.13) and (2.14). However it produces a slight change in the calculation of the low estimator:

$$\theta_T^{i_1 \dots i_T} = h_T\left(X_T^{i_1 \dots i_T}\right)$$

$$\eta_t^{i_1 \dots i_t j} = h_t\left(X_t^{i_1 \dots i_t}\right)$$

if

$$h_t\left(X_t^{i_1 \dots i_t}\right) \geq \frac{1}{b-2} \sum_{\substack{i=1 \\ i \neq j}}^{b/2} \left[e^{-R_{t+1}^{i_1 \dots i_t i}} \theta_{t+1}^{i_1 \dots i_t i} + e^{-R_{t+1}^{i_1 \dots i_t i+b/2}} \theta_{t+1}^{i_1 \dots i_t i+b/2} \right]$$

otherwise

$$\eta_t^{i_1 \dots i_t j} = \frac{1}{2}\left[e^{-R_{t+1}^{j}}\, \theta_{t+1}^{i_1 \dots i_t j} + e^{-R_{t+1}^{j+b/2}}\, \theta_{t+1}^{i_1 \dots i_t j+b/2} \right]$$

$$for\ j = 1, \dots, b\ /2$$

$$\theta_t^{i_1 \dots i_t} = \frac{2}{b} \sum_{j=1}^{b/2} \eta_t^{i_1 \dots i_t j} \qquad for\ t = 0, \dots, T-1\,.$$

At each node the branches are divided into two sets consisting of 2 and $b-2$ branches. The two branches constitute an antithetic pair and the continuation value is taken to be the average of the

[62] This approach is slightly different from the technique used to reduce variance for simulated European option prices. In the European case for every simulated path, another is generated using the corresponding antithetic variates (See, e.g., Hull (2009)).

discounted values of this two estimates. The continuation decision is based on the remaining branches[63].

2.2.2 Stochastic Mesh Method

Like the stochastic tree algorithm the stochastic mesh method is designed to solve a randomly sampled dynamic programming problem to approximate the price of an American option. The value of the derivative starting at time t in state x is[64] :

$$Q(t,x) = max(h(t,x), E[Q(t+1, S_{t+1})|S_t = x]) \quad for\ t < T \tag{2.19}$$

$$Q(T,x) = h(T,x)\ .$$

The vector of state variables X_t is governed by risk neutral probabilities and $h(t,x)$ gives the payoff in state x at time t, discounted at time 0. The objective is to calculate $Q \equiv Q(0, S_0)$.

The mesh estimator begins by generating independent random vectors $X_{t,i}\ for\ i = 1, \dots, b\ and\ t = 1, \dots, T$[65]. Then the estimator is given by:

$$\hat{Q}(T, X_{T,i}) = h(T, X_{T,i}) \ \ for\ i = 1, \dots, b \tag{2.20}$$

$$\hat{Q}(t, X_{t,i}) = max\left(h(t, X_{t,i}), \frac{1}{b}\sum_{j=1}^{b} \hat{Q}(t+1, X_{t+1,j}) w(t, X_{t,i}, X_{t+1,j}) \right)$$

$$for\ i = 1, \dots, b\ and\ t = 1, \dots, T-1$$

$$\hat{Q}(0, X_0) = \frac{1}{b}\sum_{j=1}^{b} \hat{Q}(1, X_{1,j})$$

where $w(t, X_{t,i}, X_{t+1,j})$ represents the weight attached to the arc joining node $X_{t,i}$ with node $X_{t+1,j}$[66].

An important feature of the stochastic mesh is that in estimating the option price at a node at time t, the mesh uses values from all the nodes at time $t+1$ and not just the successor of the initial node. Due to this property it maintains the number of nodes at each time step fixed thus avoiding the exponential growth characteristic of the stochastic tree. An illustration of the mesh is given in Figure A8 in Appendix E.

[63] For a further exposition of the estimator look at Broadie and Glasserman (1997).
[64] $S_1, \dots, S_T$ is a Markov process with fixed initial state S_0. In the following analysis we will denote the simulated state variable process as $X_1, \dots, X_T$.
[65] $X_{0,1} = S_0$.
[66] $X_{t,i}$ and $X_{t+1,j}$ need not to be on the same path for $w(t, X_{t,i}, X_{t+1,j})$ to be nonzero. In this way it is possible to forget which nodes were on the same paths and treat every node at time $t+1$ as a potential successor of every node at time *t.*

2.2.2.1 Random Vectors Generation and Weights Determination

In the following analysis the conditional density from node $X_{t,i}$ to $X_{t+1,j}$ will be denoted by $f(t, X_{t,i}, X_{t+1,j})$ and the marginal density of $X_{t,i}$ will be denoted by $g(t,\cdot)$. The mesh construction is assumed to be Markovian in the sense that for $t = 1, \dots, T$ the vectors $X_{t,i},\ i = 1, \dots, b$ are generated as independent and identically distributed samples from some density function $g(t,\cdot)$.

From the American option value at time *t* presented in (2.19) the objective is to approximate the true option price at each state at time *t*, $Q(t, X_{t,i}), i = 1, \dots, b$ by using the information generated from the mesh estimates of the option price at time $t+1$ $\hat{Q}(t+1, X_{t+1,j})$ for $j = 1, \dots, b$. More specifically an estimate of $E[Q(t+1, X_{t+1,j})|X_{t,i}], i = 1, \dots, b$ must be produced using the information $\hat{Q}(t+1, X_{t+1,j})$. The main issue is that the conditional density of $X_{t+1,j}$ is $f(t, X_{t,i}, X_{t+1,j})$ while the mesh nodes $X_{t+1,j}, j = 1, \dots, b$ were all sampled from the marginal density function $g(t+1,\cdot)$. Broadie and Glasserman (2004) showed that it is possible to overcome this problem and produce a consistent estimate of $E[Q(t+1, X_{t+1,j})|X_{t,i}], i = 1, \dots, b$ by adjusting the weight that each observation at time $t+1$ will be assigned in the calculation of the expectation. In particular:

$$\begin{aligned} E[Q(t+1, X_{t+1,j})|X_{t,i}] &= \int Q(t+1,u)\, f(t, X_{t,i}, u)\, du \\ &= \int Q(t+1,u) \frac{f(t, X_{t,i}, u)}{g(t+1,u)} g(t+1,u)\, du \\ &\equiv E\left[Q(t+1, X_{t+1,j}) \frac{f(t, X_{t,i}, X_{t+1,j})}{g(t+1, X_{t+1,j})} \right]. \end{aligned}$$

Given this result and defining the mesh estimator as:

$$\hat{Q}(t, X_{t,i}) = max\left(h(t, X_{t,i}), \frac{1}{b} \sum_{j=1}^{b} \hat{Q}(t+1, X_{t+1,j}) w(t, X_{t,i}, X_{t+1,j}) \right)$$

$$for\ i = 1, \dots, b\ and\ t = 1, \dots, T-1$$

where

$$w(t, X_{t,i}, X_{t+1,j}) = \frac{f(t, X_{t,i}, X_{t+1,j})}{g(t+1, X_{t+1,j})}$$

it is possible to produce an approximation of the derivative price $Q(t, X_{t,i})$ through the mesh simulation[67].

2.2.2.2 Selection of the Mesh Density

The correct selection of the density used to generate the mesh is essential to exploit efficiencies in the computation of the estimates. In order to present the impact that the density function may have on the estimator's variance Broadie and Glasserman (2004) considered the pricing through the mesh of a European style derivative with three years to maturity[68]. They showed that $\hat{Q}(0, S_0)$ may be written as

$$\hat{Q}(0, S_0) = \frac{1}{b}\sum_{j_1=1}^{b} \frac{f(0, S_0, X_{1,j_1})}{g(1, X_{1,j_1})}\hat{Q}(1, X_{1,j_1})$$

$$= \frac{1}{b}\sum_{j_1=1}^{b} \frac{f(0, S_0, X_{1,j_1})}{g(1, X_{1,j_1})}\left[\frac{1}{b}\sum_{j_2=1}^{b} \frac{f(1, X_{1,j_1}, X_{2,j_2})}{g(2, X_{2,j_2})}\hat{Q}(2, X_{2,j_2})\right]$$

$$= \frac{1}{b}\sum_{j_3=1}^{b} h(T, X_{T,j_3})\left[\frac{1}{b}\sum_{j_2=1}^{b} \frac{f(2, X_{2,j_2}, X_{T,j_3})}{g(T, X_{T,j_3})}\left[\frac{1}{b}\sum_{j_1=1}^{b} \frac{f(1, X_{1,j_1}, X_{2,j_2})}{g(2, X_{2,j_2})}\frac{f(0, S_0, X_{1,j_1})}{g(1, X_{1,j_1})}\right]\right].$$

The last line demonstrates that the mesh estimator is just a linear combination of the maturity payoffs. Generalizing for an arbitrary *T* the European option price can be written as:

$$\hat{Q}(0, S_0) = \frac{1}{b}\sum_{j_T=1}^{b} h(T, X_{T,j_T})L(T, j_T) \tag{2.21}$$

where

$$L(T, j_T) = \frac{1}{b^{T-1}}\sum_{j_1,\ldots,j_{T-1}}^{b}\left(\prod_{i=1}^{T} \frac{f(i-1, X_{i-1,j_{i-1}}, X_{i,j_i})}{g(i, X_{i,j_i})}\right). \tag{2.22}$$

[67] The choice of the weights assumes that the transition density $f(t, x, \cdot)$ of the underlying state variable is known or easy to be evaluated numerically. The interested reader may refer to the paper of Broadie, Glasserman and Ha (2000) for an alternative selection of weights that do not depend on densities.

[68] Given the impossibility of early exercise the mesh estimator for the option would be

$$\hat{Q}(t, X_{t,i}) = \frac{1}{b}\sum_{j=1}^{b} \hat{Q}(t+1, X_{t+1,j})\frac{f(t, X_{t,i}, X_{t+1,j})}{g(t+1, X_{t+1,j})} \quad \text{with} \quad \hat{Q}(T, X_{T,i}) = h(T, X_{t,i}).$$

Therefore the likelihood ratio in (2.22) can be interpreted as the weight associated with the j_T node in the mesh[69]. Broadie and Glasserman (2004) demonstrated in their Proposition 1 that the variance of the likelihood ratios grows exponentially with the time to maturity[70]. Given this fact, an inappropriate choice of the mesh density may lead to extremely imprecise estimates as the maturity of the derivative instrument increases. For this reason the authors suggest a method that completely eliminates the variance by making the likelihood ratio coefficients constant and equal to one. They manage to achieve this result by setting

$$g(t,u) = f(0,S_0,u) \quad for\ t = 1 \tag{2.23}$$

$$g(t,u) = \frac{1}{b}\sum_{j=1}^{b} f\big(t-1, X_{t-1,j}, u\big) \quad for\ t = 2,\dots,T\,. \tag{2.24}$$

Using the average density functions in (2.23) and (2.24) each $L(T,j)$ is identically and equal to one[71]. Consequently the mesh estimate of a European option price is just the average of the discounted terminal payoffs

$$\frac{1}{b}\sum_{i=1}^{b} h\big(T, X_{T,j}\big)\,.$$

The intuition behind the use of the average density method is to generate *b* independent paths of the underlying asset and then "forgetting" which nodes were on which paths[72].

By using this result in equation (2.20) the weight used in the in the mesh estimator for the transition from $X_{t,i}\ \ i = 1,\dots b$ on the ith path to $X_{t+1,j}\ \ j = 1,\dots b$ on the jth path becomes:

$$w\big(t, X_{t,i}, X_{t+1,j}\big) = \frac{f\big(t, X_{t,i}, X_{t+1,j}\big)}{b^{-1}\sum_{k=1}^{b} f\big(t, X_{t,k}, X_{t+1,j}\big)}\,. \tag{2.25}$$

This is the construction that will be implemented in the numerical analysis.

[69] It is natural to expect that the variance of the estimator $\hat{Q}(0,S_0)$ comes from the likelihood ratio multiplying the payoff function rather than the payoff function itself.

[70] Specifically they showed that $Var[L(t+1)] \geq a\lambda^t$ for some $a > 0$ and $\lambda > 1$. For a more detailed exposition of the assumptions made to reach the result and its complete demonstration the reader may refer to the appendix of Broadie and Glasserman (2004).

[71] Proposition 2 in Broadie and Glasserman (2004). A complete proof may be found in the appendix of the paper.

[72] Suppose we choose a node $X_{t,j}$ randomly and uniformly from the *b* nodes at time *t* and generate a successor by sampling from the transition density $f(t, X_{t,j}, \cdot)$. The procedure is repeated to generate a total of *b* nodes at time $t+1$ (we sample with replacement from the nodes at time *t* and generate a successor from each selected node). Thus conditional on $\{X_{t,1},\dots,X_{t,b}\}$ the node at time $t+1$ are i.i.d. with density equal to (2.24).

2.2.2.3 High Bias

The estimator produced by the stochastic mesh is biased high. To demonstrate this, an induction argument, used in Glasserman (2004), will be presented. The initial hypothesis is that if for some *t*

$$E\left[\hat{Q}_{t+1,j}|X_t\right] \geq Q_{t+1,j}, \qquad j = 1, \dots, b \tag{2.26}$$

then the same should be true for all times smaller than *t*[73]. By using (2.20) together with Jensen's inequality it is possible to write:

$$E\left[\hat{Q}\left(t, X_{t,i}\right)|X_t\right] \geq \max\left\{h\left(X_{t,i}\right), \frac{1}{b}\sum_{j=1}^{b} E\left[\hat{Q}\left(t+1, X_{t+1,j}\right)w\left(t, X_{t,i}, X_{t+1,j}\right)|X_t\right]\right\}. \tag{2.27}$$

By further conditioning on X_{t+1} and assuming that $w\left(t, X_{t,i}, X_{t+1,j}\right)$ is a deterministic function of X_t and X_{t+1}[74] it is possible to write the expectation on the right hand side of (2.27) as:

$$\begin{aligned} E\left[w\left(t, X_{t,i}, X_{t+1,j}\right)\hat{Q}\left(t+1, X_{t+1,j}\right)|X_t, X_{t+1}\right] &= w\left(t, X_{t,i}, X_{t+1,j}\right)E\left[\hat{Q}\left(t+1, X_{t+1,j}\right)|X_t, X_{t+1}\right] \\ &= w\left(t, X_{t,i}, X_{t+1,j}\right)E\left[\hat{Q}\left(t+1, X_{t+1,j}\right)|X_{t+1}\right] \end{aligned} \tag{2.28}$$

where the second equality derives from the generation of i.i.d. nodes $X_{t,i},\ i = 1, \dots, b\ for\ t = 1, \dots T$. Applying the induction hypothesis (2.26) to (2.28) we get:

$$E\left[w\left(t, X_{t,i}, X_{t+1,j}\right)\hat{Q}\left(t+1, X_{t+1,j}\right)|X_t, X_{t+1}\right] \geq w\left(t, X_{t,i}, X_{t+1,j}\right)Q\left(t+1, X_{t+1,j}\right).$$

Using this result together with the reasonable assumption that if we knew the true option value at time $t+1$ the expected weighted average calculated at a node at time *t* would be the true continuation value[75] it is possible to write:

$$\begin{aligned} \frac{1}{b}\sum_{j=1}^{b} E\left[w\left(t, X_{t,i}, X_{t+1,j}\right)\hat{Q}\left(t+1, X_{t+1,j}\right)|X_t\right] &\geq \frac{1}{b}\sum_{j=1}^{b} E\left[w\left(t, X_{t,i}, X_{t+1,j}\right)Q\left(t+1, X_{t+1,j}\right)|X_t\right] \\ &= C_t\left(X_{t,i}\right). \end{aligned}$$

At the end by using this result into (2.27) it is possible to conclude that:

$$E\left[\hat{Q}\left(t, X_{t,i}\right)|X_t\right] \geq \max\left\{h\left(X_{t,i}\right), C_t\left(X_{t,i}\right)\right\} = Q\left(t, X_{t,i}\right)$$

[73] Equation 2.26 holds with equality at $t = T - 1$.
[74] This assumption is always true in the present context where the weights are determined by density functions.
[75] Mathematically it would be $\frac{1}{b}\sum_{j=1}^{b} E\left[w\left(t, X_{t,i}, X_{t+1,j}\right)Q\left(t+1, X_{t+1,j}\right)|X_t\right] = C_t\left(X_{t,i}\right)$ where $C_t\left(X_{t,i}\right)$ is the option continuation value.

which is what was intended to be shown.[76]

Finally with respect to the consistency of the estimator Broadie and Glasserman (2004) in their Theorem 2 proved the convergence in probability of the estimator to the true option price by showing that $E\left[\hat{Q}(0,S_0)\right] \to Q(0,S_0)$ as $b \to \infty$[77].

2.2.2.4 Low-Biased Estimator

As in the case of the stochastic tree a low biased estimator may be computed. Then by combining it with the mesh estimator it is possible to produce a confidence interval for the option price. To generate such estimator we simulate additional independent trajectories of the underlying process S_t until the exercise region determined by the mesh is reached. An important feature of this estimator is that the mesh is used only to determine when to exercise. The payoff in case of exercise is determined by the independent path and it is not a value estimated from the mesh.

Starting from t_0 we begin the simulation of a new path of S_t over many time steps $t = 1, \dots, T$. Along this path, the approximate optimal policy determined by the mesh is

$$\hat{\tau}(S) = min\left\{t: h(t,S_t) \geq \sum_{j=1}^{b} w\left(t,S_t,X_{t+1,j}\right)\hat{Q}\left(t+1,X_{t+1,j}\right)\right\}$$

where $w\left(t,S_t,X_{t+1,j}\right)$ is the weight from node S_t on the independent path to node $X_{t+1,j}$ in the mesh and $\hat{Q}\left(t+1,X_{t+1,j}\right)$ are calculated using equation (2.20). The path estimator is therefore

$$\hat{q} = h(\hat{\tau},S_{\hat{\tau}})\,.$$

In practice, after the simulation of the independent trajectory at each node we use (2.20) with the new weights[78] to estimate the option continuation value at that node and repeat this procedure until the estimated continuation value is lower than or equal to the exercise value, i.e. the exercise region is reached. The procedure is repeated over many paths and then averaged in order to produce an estimate of $\hat{q}$. An illustrated example for three independent paths can be found in Figure A9 in Appendix E.

An important insight is that the original mesh determines the optimal exercise boundary at all possible states and not only to those corresponding to nodes in the mesh. For this reason the exercise region determined by the mesh is not the optimal exercise region. Since it cannot be better than the optimal region because it is based on a finite sample the average payoff generated by this procedure

[76] For a more detailed proof of the high bias of the mesh estimator the reader may refer to the appendix of the Broadie and Glasserman (2004) paper.
[77] For a complete proof of this result the reader may refer to the appendix of the Broadie and Glasserman paper or to Avramidis and Matzinger (2002). In the latter paper the authors derive a probabilistic upper bound on the error in the mesh estimator and use it to prove convergence.
[78] The functional form of the weights is (2.25).

cannot be greater than the average payoff determined optimally. We therefore conclude that that estimate produced by the path estimator is biased low[79].The convergence of the path estimator to the true option price is stated in Theorem 4 in Broadie and Glasserman (2004). Specifically $E[\hat{q}] \rightarrow Q(0, S_0)$ as $b \rightarrow \infty$.[80]

2.2.2.5 Efficiency Enhancements

2.2.2.5.1 Bias Reduction for the Mesh Estimator (Low Estimator)

The idea driving the construction of a new low biased estimate resembles the one already used for the low estimator in the stochastic tree, i.e. using disjoint sets of points for the estimation of the optimal exercise policy and the estimation of continuation values.

Suppose that $I \subseteq b$ denotes an arbitrary subset of all indices and $I' = b - I$ is its complement with respect to b. The estimate of the continuation value at time t is defined using only the points in I at $t+1$:

$$\hat{c}(t, X_{t,i}, I) = \frac{1}{|I|} \sum_{j \in I} \hat{Q}(t+1, X_{t+1,j})\, w(t, X_{t,i}, X_{t+1,j}),$$

where the weights are defined as in (2.25). The consequent estimate of the option price at node $X_{t,i}$ is

$$\hat{q}_L(t, X_{t,i}, I) = \begin{cases} h(t, X_{t,i}) & if\ h(t, X_{t,i}) \geq \hat{c}(t, X_{t,i}, I) \\ \hat{c}(t, X_{t,i}, I') & otherwise \end{cases}.$$

To maximize the usage of known information at stage $t+1$ on the estimation at stage t the low estimator is set equal to the average of b copies of $\hat{q}_L$ where the jth copy uses $b_{-j} \equiv b - \{j\}$ in place of I. The low estimator is therefore defined recursively

$$\hat{q}_L(T, X_{T,i}) = h(T, X_{T,i}) \quad for\ i = 1, \dots, b \tag{2.29}$$

$$\hat{q}_L(t, X_{t,i}) = \frac{1}{b} \sum_{j=1}^{b} \hat{q}_L\,(t, X_{t,i}, b_{-j}) \quad for\ t = T-1, \dots, 0 \ \ and\ \ for\ i = 1, \dots, b\,.$$

In their Theorem 1 Avramidis and Hyden (1999) showed the low bias of the estimator, i.e.

$$E[\hat{q}_L(t,x)] \leq q(t,x).$$

[79] The reader may refer to Theorem 3 of Broadie and Glasserman (2004) for a complete demonstration of this property of the path estimator.
[80] The path estimator depends on the parameter b through the mesh policy which itself depends on b.

Combining the mesh estimator end the low estimator just described we derive the average estimator of the derivative price that will be used in the numerical analysis

$$\hat{q}_A(T, X_{T,i}) = h(T, X_{T,i}) \quad for\ i = 1, \dots, b$$

$$\hat{q}_A(t, X_{t,i}) = \frac{1}{2}\hat{Q}(t, X_{t,i}) + \frac{1}{2}\hat{q}_L(t, X_{t,i}) \quad for\ t = T-1, \dots, 0 \ \ and \ \ for\ i = 1, \dots, b$$

where $\hat{Q}(t, X_{t,i})$ and $\hat{q}_L(t, X_{t,i})$ are respectively the mesh and low estimators defined in (2.20) and (2.29).

2.2.2.5.2 Control Variate, Pruning and Antithetic Variates

Together with the low estimator, additional techniques will be used in the following analysis to improve the efficiency of the stochastic mesh algorithm. In particular pruning and antithetic variates will be implemented following the same procedures already presented for the stochastic tree.

With respect to control variate, the procedure used is fairly standard. N independent meshes will be generated to produce estimates $\hat{Q}^{(i)}, i = 1, \dots, N$ of the option price. Imagine that the quantity $u = u(0, S_0)$ is known and can be easily computed analytically. u might represent the value of a European option $E[h(T, S_T)]$. Each mesh is then used to produce an unbiased estimate $\hat{u}^{(i)}, i = 1, \dots, N$ of u using equations (2.21) and (2.22). Then the controlled estimator will be:

$$\frac{1}{N}\sum_{i=1}^{N}\hat{Q}^{(i)} - \beta\left(\frac{1}{N}\sum_{i=1}^{N}\hat{u}^{(i)} - u\right). \tag{2.30}$$

The coefficient β can be estimated by solving a least squares problem[81].

2.2.3 The Longstaff-Schwartz Algorithm (LSM)

The Longstaff-Schwartz algorithm is one of the most popular among practitioners, particularly for the pricing of American options on more than one underlying asset.

We first present the valuation framework and notation necessary to describe the algorithm. We assume an underlying complete probability space $(\Omega, \mathcal{F}, \mathrm{P})$[82] and a finite time horizon $[0, T]$. We define an augmented filtration generated by the price processes of the underlying securities $F = \{\mathcal{F}_t;\ t \in [0, T]\}$. $C(\omega, s; t, T)$ will represent the cash flow path generated by the option,

[81] β is chosen to solve

$$\min_{\alpha,\beta}\frac{1}{N}\sum_{i=1}^{N}\left[\hat{Q}^{(i)} - (\alpha + \beta\hat{u}^{(i)})\right]^2.$$

[82] Ω is the set of all possible states of the stochastic economy, $\mathcal{F}$ is the sigma field of distinguishable events at time T and P is the probability measure defined on the elements of $\mathcal{F}$.

conditional of not having exercised the option before or at time t and on following the optimal exercise strategy $s, t < s < T$.

The objective is to provide an approximation of the optimal stopping rule that maximizes the value of the American option.

At expiration the holder will exercise the option depending on it being in the money or not. At any prior time t_k the holder must choose whether to exercise immediately or to continue the life of the option. The value of the option is maximized if the investor exercises as soon as the immediate exercise cash flow is greater than or equal to the continuation value. However the cash flows from continuation are not known at time t_k. Thus the continuation value is given by taking expectation of the remaining discounted cash flows $C(\omega, s; t_k, T)$ with respect to the risk neutral pricing measure Q. Specifically

$$F(\omega; t_k) = E_Q\left[\sum_{j=k+1}^{K} exp\left(-\int_{t_k}^{t_j} r(s)\, ds\right) C\big(\omega, t_j; t_k, T\big)|\mathcal{F}_{t_k}\right]. \tag{2.31}$$

The problem of optimal exercise reduces to comparing the immediate exercise value with this conditional expectation and exercise as soon as the former is greater than or equal to the latter.

The LSM algorithm uses least squares to approximate the conditional expectation function at every exercise date $t_{T-1}, \dots, t_1$. It uses a backward induction method since the path of cash flows $C(\omega, s; t, T)$ is defined recursively. Specifically at time t_{k-1} the unknown functional form of (2.31) can be approximated by a linear combination of a countable set of $\mathcal{F}_{t_{k-1}}$ measurable basis functions[83].

Longstaff and Schwartz (2001) suggested to use as basis functions the weighted Laguerre polynomials[84]:

$$L_0(X) = \exp(-X/2),$$

$$L_1(X) = \exp(-X/2)(1 - X),$$

$$L_2(X) = \exp(-X/2)(1 - 2X + X^2/2),$$

$$L_n(X) = \exp(-X/2)\frac{e^x}{n!}\frac{d^n}{dX^n}(X^n e^{-X}).$$

[83] Longstaff and Schwartz (2001) justify that we can follow this approach because of the assumption that the conditional expectation is an element of L^2, the space of square-integrable functions relative to some measure. Since L^2 is a Hilbert space it has a countable orthonormal basis and the conditional expectation can be represented as a linear function of the elements of the basis.

[84] In these examples X may be assumed to be the state variable underlying the derivative contract.

Other types of basis functions include the Hermite, Legendre and Jacobi polynomials[85]. With this specification $F(\omega; t_{k-1})$ can be approximated by a set of $M < \infty$ basis functions:

$$F_M(\omega; t_{k-1}) \approx \sum_{j=0}^{M} \beta_j L_j\left(X_{t_{k-1}}\right). \tag{2.32}$$

The vector β is given by

$$\beta = \left(E\left[L\left(X_{t_{k-1}}\right)_{(M+1)\times N} L\left(X_{t_{k-1}}\right)^T_{(M+1)\times N}\right]\right)^{-1} E\left[L\left(X_{t_{k-1}}\right)_{(M+1)\times N} V_{t_{k-1}}\left(X_{t_{k-1}}\right)_{N\times 1}\right] \equiv B_L^{-1} B_{LV}$$

where $V_{t_{k-1}}\left(X_{t_{k-1}}\right)$ are the discounted values of $C(\omega, s; t_{k-1}, T)$ at time t_{k-1} for the N paths included in the regression. B_L^{-1} is a $(M+1)\times(M+1)$ matrix (assumed nonsingular) and B_{LV} a vector of length $(M+1)$. The coefficients in β can be estimated from observations of pairs $\left(X_{t_{k-1},i}, V_{t_{k-1}}\left(X_{t_{k-1},i}\right)\right), i = 1, \dots, N$ each consisting of the state variable at time t_{k-1} and the corresponding discounted option value at time t_k. The least squares estimate of β is given by

$$\hat{\beta} = \hat{B}_L^{-1} \hat{B}_{LV}$$

where $\hat{B}_L^{-1}$ and $\hat{B}_{LV}$ are the sample counterparts of B_L^{-1} and B_{LV}. Therefore $F_M(\omega; t_{k-1})$ is estimated[86] by regressing the discounted values of $C(\omega, s; t_{k-1}, T)$ onto the basis functions for the paths where the option is in the money[87] at time t_{k-1}.

Once the conditional expectation function at time t_{k-1} is estimated the exercise decision is determined by comparing the immediate exercise value of each of the in the money paths with the expected continuation value $\hat{F}_M(\omega; t_{k-1})$. Once the exercise decision is identified the path of cash flows $C(\omega, s; t_{k-2}, T)$ can then be approximated. This procedure is reiterated recursively by proceding backward in time until all the exercise decisions for each exercise time and along each path have been determined. The American option is then valued by starting at time zero, moving forward along each path until the exercise time occurs, discounting the respective cash flow from exercise back to time zero and then taking the average over all the simulated paths. An example of pseudo-algorithm is presented in Appendix F.

[85] These functions are described in Abramowitz and Stegun (1965). In the following numerical analysis the LSM algorithm will be implemented mostly by using Laguerre, Hermite polynomials and basis functions suggested in Rasmussen (2005).

[86] Longstaff and Schwartz (2001) showed that under the described framework the fitted value of the regression $\hat{F}_M(\omega; t_{k-1})$ converges in mean square and in probability to $F_M(\omega; t_{k-1})$ as the number N of (in the money) paths in the simulation goes to infinity.

[87] The choice to use only in the money paths for the estimation is motivated by the argument that exercise decision is only relevant when the option is in the money. This limits the number of basis functions to be calculated thus reducing the computation time required to estimate the coefficients.

2.2.3.1 Convergence Theory

The analysis of the convergence of the Longstaff-Schwartz algorithm is far from trivial. Clement et al. (2002) provide an in depth study of the convergence properties, by proving that convergence actually takes place and by deriving the correspondent rate of convergence. Moreover Glasserman and Yu (2004) give an analysis of the relationship between the number of paths required for a particular number of basis functions in order to ensure convergence.

The Longstaff-Schwartz algorithm presents two major sources of error coming from two approximations. First we replace the conditional expectation (2.31) in the dynamic programming principle by projections on a finite set of functions taken from a suitable basis. Using this approximation to the continuation value we find an approximated and sub-optimal stopping rule for the option. Second we use Monte Carlo simulation and least squares regression to compute the value function of the first approximation. Specifically the coefficients β_j are being approximated by a Monte Carlo regression, therefore we are using a set of $M+1$ $\hat{\beta}_j$ whose MSE will be related to the number of paths that are used in the approximation. Moreover once the coefficients are estimated we perform a Monte Carlo simulation to estimate the true option price at time zero V_0 which introduces further error.

With respect to the first approximation, Clement et al. (2002) showed that, if the sequence of basis functions is total in a suitable L^2-function space

$$\lim_{M\to\infty} F_M(\omega; t_k) = F(\omega; t_k) \quad in \ L^2 .$$

This covers the first source of error by proving that the estimated option price V_0^M converges to the true price V_0 as the number of basis functions employed increases. Considering the second approximation they showed in their Theorem 3.2 that the estimate produced by a Monte Carlo simulation of N paths $\hat{V}_0^{M,N}$ converges to V_0^M as the number N goes to infinity

$$\lim_{N\to\infty} \hat{V}_0^{M,N} = V_0^M .$$

Clement et al. (2002) showed that the rate of convergence towards V_0^M is $O\left(N^{-1/2}\right)$. The problem is that we don't have a similar result for the number of basis functions.

In Glasserman and Yu (2004) the relationship between the number of paths and the number of basis functions was analyzed. They studied the MSE of the β_j coefficients and derived bounds on the number of paths required for this quantity to converge in the cases where the underlying asset is a Brownian motion or a geometric Brownian motion. In both cases they found that the number of paths required for convergence of the MSE grew exponentially with the number of basis functions. Their results suggest a careful choice in the number of paths used in the simulation as well as an analysis of the precision and stability of the estimator.

2.2.3.2 Efficiency Enhancements

2.2.3.2.1 Control Variate

As we already presented earlier, to reduce the variance of the Monte Carlo estimates we can use a control variate. Specifically, in the numerical analysis, the control variate technique will be used to reduce the variance of the option continuation value. The new estimate using control variate is then given by

$$F(t)^{(b)CV} = \frac{1}{b}\sum_{i=1}^{b} F(i,t) + \theta_t \left(Y_t^{(b)} - E_t^Q[Y]\right)$$

$$Y_t^{(b)} = \frac{1}{b}\sum_{i=1}^{b} Y^i .$$

θ_t is some appropriately chosen $\mathcal{F}_t$ measurable random variable and Y^i is the ith observation of a random variable for which it is easily possible to compute the conditional expectation at time t.

As we have already analyzed in section 2.1.1.2 the optimal choice of θ_t is given by

$$\theta_t^* = -\frac{Cov_t[F(t), Y]}{Var_t[Y]}.$$

Since in practice it is seldom possible to determine θ_t^* analitycally we will estimate it through the Monte Carlo simulation. Specifically the estimate of θ_t^* using b independent paths will be

$$\theta_t^{(b)} = -\frac{(FY)_t^{(b)} - F(t)^{(b)} Y_t^{(b)}}{(Y_t^2)^{(b)} - \left(Y_t^{(b)}\right)^2}$$

where

$$(FY)_t^{(b)} = \frac{1}{b}\sum_{i=1}^{b} F(i,t) Y^i,$$

$$(Y_t^2)^{(b)} = \frac{1}{b}\sum_{i=1}^{b} \left(Y^i\right)^2.$$

Rasmussen (2005) provides a good candidate for Y. Let $\{Y_t\}_{0 \le t \le T}$ be the discounted value process of a self-financing portfolio closely matching the payoff of the corresponding European option. Under the risk neutral measure Q such a process $\{Y_t\}_{0 \le t \le T}$ is a martingale and hence satisfy

$$Y_t = E_t^Q[Y_T] .$$

The main idea of Rasmussen (2005) is to use the following control variate

$$Y_s = E_s^Q[Y_T]$$

where s is the stopping time corresponding to the exercise of the American option. This can be justified by the Optimal Sampling Theorem which ensures that

$$Y_t = E_t^Q[Y_s] . \tag{2.33}$$

In the case where we can determine the value of the self-financing portfolio that exactly matches the value of the European option, i.e. we can easily value the European option analytically, the American option control variate is given by

$$Y_s = E_s^Q[C_T * exp(-rT)] = C_s * ex\,p(-rs) . \tag{2.34}$$

This is the discounted European option price measured at the exercise time of the American option.
In Appendix F a pseudo algorithm of the control variate implementation is described.

2.2.3.2.2 The Control Variate Improved LSM Approach

In this section we consider the improvement of the LSM approach by applying control variate to the least-squares regression. We denote this suggestion the CV-LSM approach.

The procedure requires the replacement of the option payoff $W_s = C(\omega, s, T)$ from following the exercise strategy s, with a random variable Z_s with the same time t conditional expectation but with a smaller conditional variance. Rasmussen (2005) proved in his Proposition 2 that this method improves the efficiency of the least squares projection without changing the bias[88].

A suitable candidate for Z_s is given by the control variate adjusted payoff

$$Z_s = W_s + \theta_{t_k}\left(Y_s - E_{t_k}^Q[Y_s]\right)$$

[88] Rasmussen (2005) Proposition 2. Let P_t^M denote the projection onto the $\mathcal{F}_t$-measurable $M+1$ basis functions. Assume we have a $\mathcal{F}_t$-measurable random variable Z_s with the following two properties

$$E_t^Q[Z_s] = E_t^Q[W_s]$$
$$Var_t^Q[Z_s] \le Var_t^Q[W_s]$$

then

$$E_t^Q\left[P_t^M Z_s - E_t^Q[Z_s]\right] = E_t^Q\left[P_t^M W_s - E_t^Q[W_s]\right]$$
$$E_t^Q\left[\left(P_t^M Z_s - E_t^Q[Z_s]\right)^2\right] \le E_t^Q\left[\left(P_t^M W_s - E_t^Q[W_s]\right)^2\right] .$$

where Y_s is the control variate defined in equation (2.34) and θ_t is a $\mathcal{F}_t$- measurable random variable. Imagine for now that θ_t is given and let P_t^M denote the projection onto the $\mathcal{F}_t$-measurable $M+1$ basis functions, we get by the linearity of projections that

$$P_{t_k}^M Z_s = P_{t_k}^M \left(W_s + \theta_{t_k} \left(Y_s - E_{t_k}^Q [Y_s] \right) \right)$$

$$= P_{t_k}^M W_s + \theta_{t_k} \left(P_{t_k}^M Y_s - P_{t_k}^M E_{t_k}^Q [Y_s] \right). \tag{2.35}$$

Hence the projection of Z_s is just a linear combination of the projections of W_s, Y_s and $E_{t_k}^Q [Y_s]$. By making the simplifying assumption that the time t_k conditional expectation of the control variate $Y_{t_k} = E_{t_k}^Q [Y_s]$ is in the span of the $M+1$ basis functions[89] it is possible to modify (2.35) to

$$P_{t_k}^M Z_s = P_{t_k}^M W_s + \theta_{t_k} \left(P_{t_k}^M Y_s - Y_{t_k} \right). \tag{2.36}$$

Considering now the optimal functional form of θ_{t_k} we get

$$\theta_{t_k}^* = -\frac{Cov_{t_k}^Q [W_s, Y_s]}{Var_{t_k}^Q [Y_s]} = -\frac{E_{t_k}^Q [W_s, Y_s] - E_{t_k}^Q [W_s] E_{t_k}^Q [Y_s]}{E_{t_k}^Q [Y_s^2] - E_{t_k}^Q [Y_s]^2} = -\frac{(WY)_{t_k} - W_{t_k} Y_{t_k}}{(Y^2)_{t_k} - \left(Y_{t_k} \right)^2}$$

where Y_{t_k} is defined in (2.33) and

$$W_{t_k} = E_{t_k}^Q [W_s] = F(\omega, t_k)$$

$$(Y^2)_{t_k} = E_{t_k}^Q [Y_s^2]$$

$$(WY)_{t_k} = E_{t_k}^Q [W_s Y_s].$$

As in (2.32) we assume that the time t_k conditional expectation of Y_{t_k}, $(Y^2)_{t_k}$ and $(WY)_{t_k}$ can also be expressed as a countable sum of the same set of $\mathcal{F}_t$-measurable basis functions[90]. Then we approximate the conditional expectations by using a set of $M+1$ functions. Hence we have

$$Y_{t_k}^M = \sum_{j=0}^M b_j L_j \left(X_{t_k} \right), \quad (Y^2)_{t_k}^M = \sum_{j=0}^M c_j L_j \left(X_{t_k} \right), \quad (WY)_{t_k}^M = \sum_{j=0}^M d_j L_j \left(X_{t_k} \right). \tag{2.37}$$

Given the approximations in (2.37) it is now possible to estimate $\theta_{t_k}^*$ by $\theta_{t_k}^M$

[89] This is equivalent to $Y_{t_k} = P_{t_k}^M E_{t_k}^Q [Y_s]$.

[90] The basis functions used in Rasmussen (2005) and in the following numerical analysis are different from the polynomials presented by Longstaff and Schwartz (2001). They are very problem specific and will be presented in the following analysis during the description of the options considered.

$$\theta_{t_k}^M \equiv -\frac{(WY)_{t_k}^M - F_M(\omega; t_k) Y_{t_k}^M}{(Y^2)_{t_k}^M - \left(Y_{t_k}^M\right)^2}. \tag{2.38}$$

The approximation of the time t_k conditional expectation of the payoff following the strategy s will be given by

$$F_M^{CV}(\omega; t_k) = F_M(\omega; t_k) + \theta_{t_k}^M \left(Y_{t_k}^M - Y_{t_k}\right)$$

where the first term in the right hand side is just the original LSM approximation (2.32).

To implement the CV-LSM approach we use the same observations of the Monte Carlo simulated state variables already used to compute (2.32). We denote as $b_j^{(N)}$, $c_j^{(N)}$ and $d_j^{(N)}, j = 0,1, \dots, M$ the coefficients determined from the least-squares regression using only the N in-the-money paths. Thus together with (2.32) the approximations required to calculate the coefficient in (2.38) will be

$$Y_{t_k}^{M(N)} = \sum_{j=0}^M b_j^{(N)} L_j\left(X_{t_k}\right), \quad (Y^2)_{t_k}^{M(N)} = \sum_{j=0}^M c_j^{(N)} L_j\left(X_{t_k}\right),$$
$$(WY)_{t_k}^{M(N)} = \sum_{j=0}^M d_j^{(N)} L_j\left(X_{t_k}\right).$$

Plugging them into (2.38) provides an approximation of $\theta_{t_k}^M$

$$\theta_{t_k}^{M(N)} \equiv -\frac{(WY)_{t_k}^{M(N)} - \hat{F}_M(\omega; t_k) Y_{t_k}^{M(N)}}{(Y^2)_{t_k}^{M(N)} - \left(Y_{t_k}^{M(N)}\right)^2}.$$

At the end the CV-LSM approximation of the option continuation value using $M + 1$ basis functions is given by

$$F_M^{CV}(\omega; t_k) = \hat{F}_M(\omega; t_k) + \theta_{t_k}^{M(N)} \left(Y_{t_k}^{M(N)} - Y_{t_k}\right).$$

From a computational point of view extending the LSM approach to the CV-LSM approach requires a few steps. First of all it is necessary to sample the control variate Y_s whenever the American option is exercised[91]. Second we need to evaluate the known time t conditional expectation of the control vartiate Y_{t_k} for each observation where the option is in the money. The last step is to include four rather than one dependent variables in the least-squares regression. Therefore the computational cost of solving the least-squares equation is considered fixed. The reader may refer to appendix F for an example of pseudo-algorithm of the CV-LSM approach.

[91] This will not add extra cost if the control variate is already used to improve the valuation.

PART III: Numerical analysis

3.1 Efficiency Improvements Evaluation

Before proceeding to the pricing of derivative claims, it is necessary to study the efficiency improvements achieved over the standard versions of the pricing algorithms through the implementation of the techniques described in Part II.

The following analysis will be based on the pricing of an American call option on a single asset which pays continuous dividends and whose price is governed by a geometric Brownian motion process.

The price of the underlying asset S_t under the risk neutral measure satisfies the stochastic differential equation

$$dS_t = S_t[(r-\delta)dt + \sigma dz_t]$$

where z_t is a standard Brownian motion process, r is the risk free interest rate, δ is the continuous dividend rate and σ is the volatility parameter[92]. Given S_{i-1}, S_i can be simulated using

$$S_i = S_{i-1}e^{(r-\delta-\sigma^2/2)(t_i-t_{i-1})+\sigma\sqrt{t_i-t_{i-1}}Z},$$

where Z is a standard normal random variable.

The relevant parameters were chosen in the following manner: risk free rate $r = 0.05$; dividend yield $\delta = 0.1$; volatility $\sigma = 0.2$; strike price $K = 100$; initial stock price $S_0 = 100$; time to expiration $T = 1$ and four exercise opportunities specifically at $t = 0, T/3, 2T/3, T$. For each algorithm the branching parameter b, the number of simulation runs n and the number of estimates used to calculate the presented performance measures is specified in each table.

Since we report point estimates performance, for the stochastic tree and stochastic mesh algorithms the point estimate is given by the simple average of the high biased and low biased estimators[93].

With a fixed number of exercise opportunities the true value of the call option can be obtained from the analytical formula in Geske and Johnson (1984). To measure the algorithms performance we

[92] Under these assumptions $\ln(S_i/S_{i-1})$ is normally distributed with mean $(r-\delta-\sigma^2/2)(t_i-t_{i-1})$ and variance $\sigma^2(t_i-t_{i-1})$.

[93] $\frac{1}{2}\theta_0 + \frac{1}{2}\Theta_0$ for the stochastc tree, $\frac{1}{2}\hat{Q}(0,S_0) + \frac{1}{2}\hat{q}$ for the stochastic mesh.

follow the literature and choose the following measures: relative error[94]; relative standard error and root mean squared error (RMSE).

3.1.1 Stochastic Tree

The results concerning the stochastic tree algorithm are presented in Table 1 and Figure 3.1.

Table: 1 Stochastic Tree Efficiency Improvements Evaluation

	b=50	b=75	b=100
Relative Error (Original ST)	0.151%	0.298%	0.255%
Relative Error (Pruning)	0.341%	0.174%	0.005%
Relative Error (Pruning + Antithetic Branching)	0.187%	0.032%	0.004%
Relative Error (Pruning + Antithetic + Control)	0.041%	0.019%	0.012%
Relative SE (Original ST)	34.073%	24.903%	20.159%
Relative SE (Pruning)	29.084%	20.130%	17.816%
Relative SE (Pruning + Antithetic Branching)	18.277%	16.204%	11.236%
Relative SE (Pruning + Antithetic + Control)	2.363%	2.342%	1.832%
RMSE (Original ST)	9.787%	7.317%	5.944%
RMSE (Pruning)	8.532%	5.864%	5.105%
RMSE (Pruning + Antithetic Branching)	5.355%	4.648%	3.219%
RMSE (Pruning + Antithetic + Control)	0.717%	0.680%	0.530%

Option parameters: S=100, $K = 100$, $n = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T. Each row was constructed with 25 estimates of the derivative price.
For each measure rows 1, 2, 3 and 4 list the relevant results for standard algorithm, pruning, antithetic branching with pruning and antithetic branching with pruning and control variate respectively. European option was used as control variate.
To calculate Relative Error and RMSE the estimated price has been compared with the true price of 5.731 estimated with the formula in Geske and Johnson (1984).

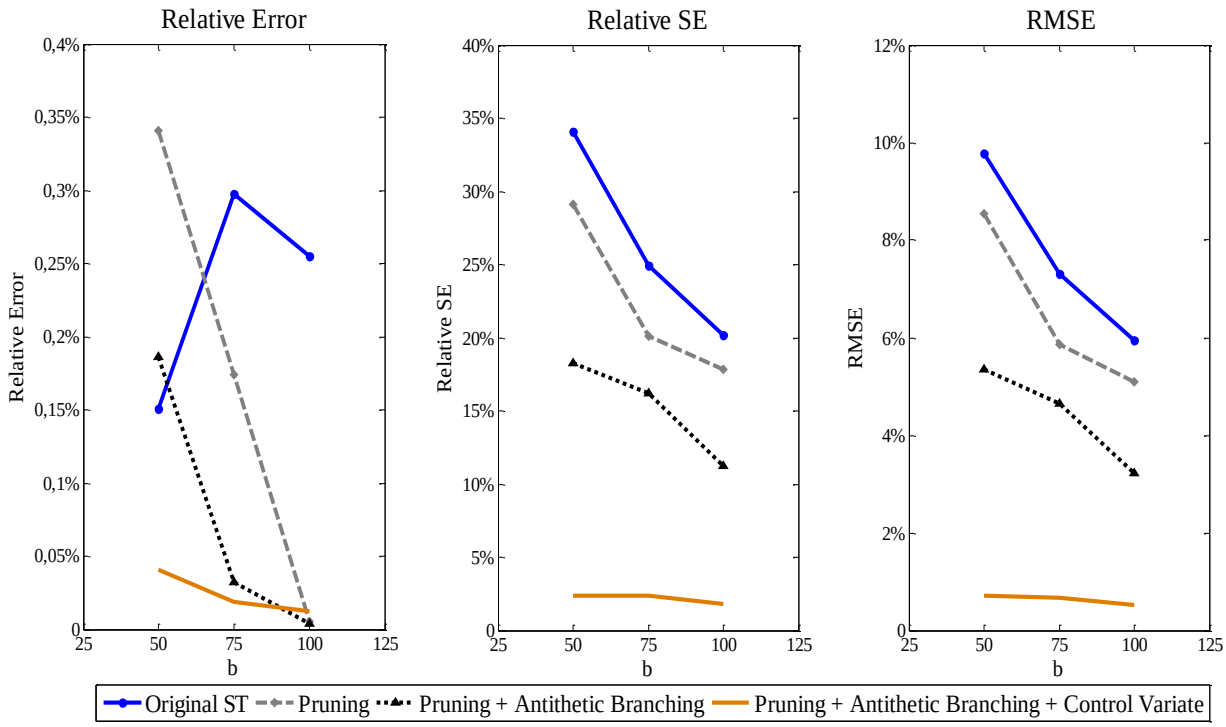
For this algorithm the number of simulation runs for each estimate is $n = 100$ and each row in the table was contructed by using 25 estimates of the derivative price. For each measure rows 1, 2, 3 and 4 list the relevant results for standard algorithm, pruning, antithetic branching with pruning and antithetic branching with pruning and control variate respectively. The European option value is used as the control variate.

The standard implementation of the algorithm even though presents a limited relative error of the estimates suffers from a consistent volatility. The general trend that can be observed in each row of Table 1 and graphically in Figure 3.1 is that by increasing the branching parameter b and consequently the computation time required for each estimate the results obtained improve both in terms of precision and variance reduction. The same result is obtained when efficiency enhancement techniques are added to the algorithm with a dramatic reduction of both the distance from the true value as well as the dispersion of the estimates. The best results are obtained when all the techniques are employed

[94]The numbers for relative error are defined by $|C - \hat{C}|/C$ where C is the true option value and $\hat{C}$ is the simulated point estimate.

with a relative error and RMSE lower than 1% and a relative SE lower than 2.5%. For this reason this will be the algorithm used when we will deal with the prices of derivative claims in the following sections. In Appendix G (Figure A10) the algorithm distribution is presented and discussed.

Figure 3.1: Stochastic Tree Algorithm: Graphical Analysis



Data are from Table 1.

Overall the results are perfectly in line to what predicted by the theory, i.e. the efficiency of the estimates improves as the branching parameter b (computation time) increases or, keeping the branching parameter fixed, efficiency enhancement techniques are added to the algorithm.

3.1.2 Stochastic Mesh

The results concerning the stochastic mesh algorithm are presented in Tables 2-3 and Figures 3.2-3.3. As for the stochastic tree the number of simulation runs for each estimate is $n = 100$ and each row in the table was constructed by using 25 estimates of the derivative price. In Table 3 for each measure rows 1, 2, 3 and 4 list the relevant results for the standard algorithm, pruning, pruning with control variate and antithetic variates with pruning and control variate respectively. The European option value is used as the control variate.

Before analyzing the efficiency enhancements produced by variance reduction techniques we first study separately in Table 2 and Figure 3.2 the low estimator performance compared to the one of the path estimator of the standard algorithm.

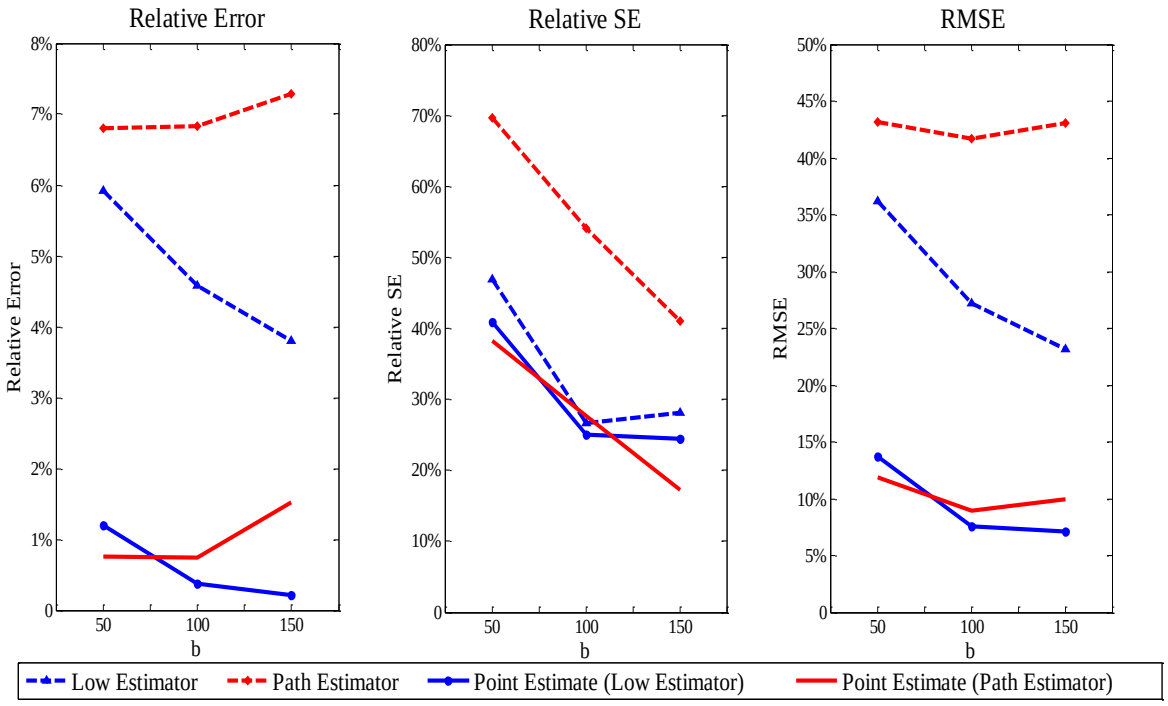
It is possible to notice that the standard path estimator presents a considerably higher bias and variance compared with the low estimator. These properties continue to hold when the two low biased estimators are used to produce the point estimates.

Table 2: Stochastic Mesh Efficiency Improvements Evaluation (Low vs. Path Estimator)

	b=50	b=100	b=150
Relative Error Low Estimator	5.922%	4.574%	3.804%
Relative Error Path Estimator	6.799%	6.833%	7.280%
Relative Error Point Estimate (Low Estimator)	1.194%	0.382%	0.218%
Relative Error Point Estimate (Path Estimator)	0.755%	0.747%	1.520%
Relative SE Low Estimator	46.971%	26.684%	28.178%
Relative SE Path Estimator	69.604%	54.112%	40.956%
Relative SE Point Estimate (Low Estimator)	40.889%	25.052%	24.445%
Relative SE Point Estimate (Path Estimator)	38.228%	27.741%	17.309%
RMSE Low Estimator	36.225%	27.209%	23.142%
RMSE Path Estimator	43.173%	41.740%	43.119%
RMSE Point Estimate (Low Estimator)	13.689%	7.532%	7.130%
RMSE Point Estimate (Path Estimator)	11.856%	8.978%	9.987%

Option parameters: S=100, $K = 100, n = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3,$ and T. Each row was constructed with 25 estimates of the derivative price.
For each statistic the first two lines compare the performances of the low and the path estimators. The third and fourth lines compare the point estimates computed with the mesh estimator together with low and path estimators respectively.
To calculate Relative Error and RMSE the estimated price has been compared with the true price of 5.731 estimated with the formula in Geske and Johnson (1984).

Figure 3.2: Stochastic Mesh Algorithm: Graphical Analysis (Low vs. Path Estimator)



Data are from table 2.

Therefore the recursively averaged mesh estimator $\hat{q}_A$ presented in section 2.2.2.5.1 appears to be the best point estimator for a general purpose mesh implementation. For this reason this will be the estimator used in the rest of the analysis in Table 3 and Figure 3.3.

Table 3: Stochastic Mesh Efficiency Improvements Evaluation

	b=50	b=100	b=150
Relative Error Point Estimate	1.194%	0.382%	0.218%
Relative Error Point Estimate (Pruning)	0.246%	0.164%	0.046%
Relative Error Point Estimate (Pruning + Control)	0.120%	0.111%	0.110%
Relative Error Point Estimate (Pruning + Control + Antithetic)	0.489%	0.322%	0.083%
Relative SE Point Estimate	40.889%	25.052%	24.445%
Relative SE Point Estimate (Pruning)	43.370%	24.949%	19.053%
Relative SE Point Estimate (Pruning + Control)	21.846%	20.712%	13.693%
Relative SE Point Estimate (Pruning + Control + Antithetic)	21.214%	12.409%	9.683%
RMSE Point Estimate	13.689%	7.532%	7.130%
RMSE Point Estimate (Pruning)	12.477%	7.199%	5.464%
RMSE Point Estimate (Pruning + Control)	6.305%	5.962%	3.969%
RMSE Point Estimate (Pruning + Control + Antithetic)	6.720%	4.017%	2.818%

Option parameters: S=100, $K = 100, n = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T. Each row was constructed with 25 estimates of the derivative price. For each measure rows 1, 2, 3 and 4 list the relevant results for standard algorithm, pruning, pruning with control variate and antithetic variates with pruning and control variate respectively. European option was used as control variate.
To calculate Relative Error and RMSE the estimated price has been compared with the true price of 5.731 estimated with the formula in Geske and Johnson (1984).

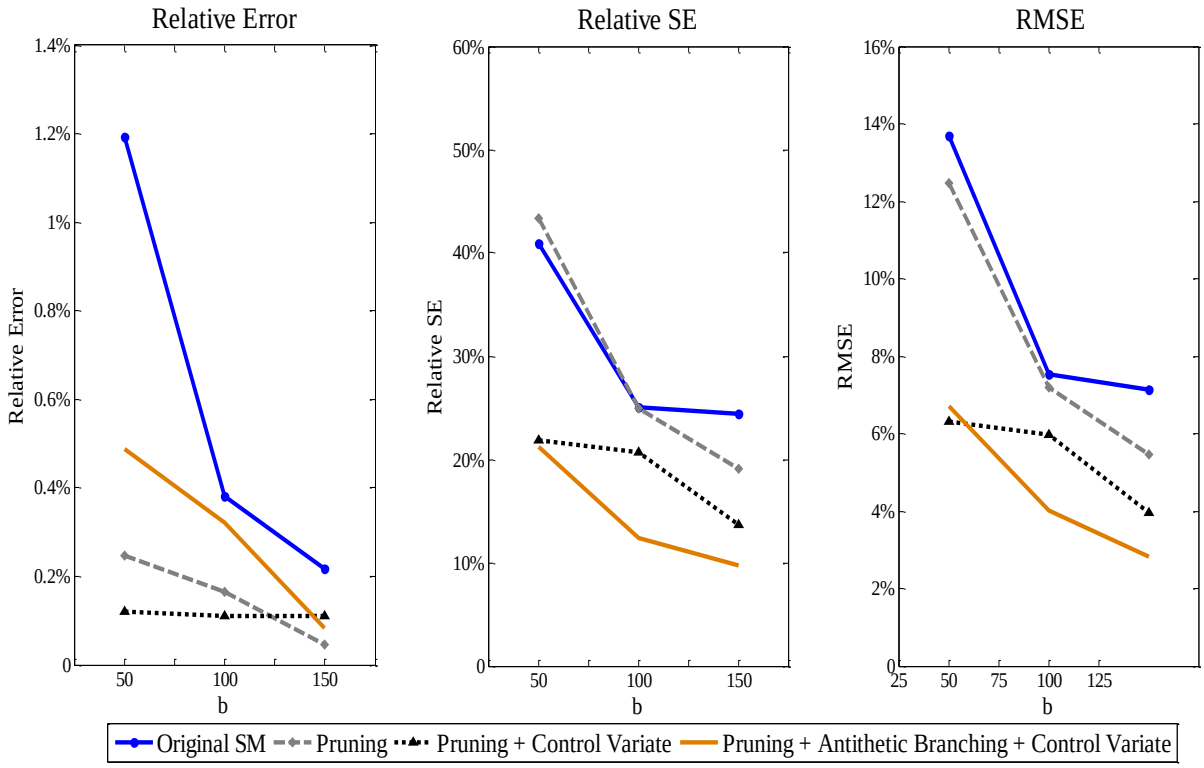
The results in Table 3 and Figure 3.3 resemble the ones obtained for the stochastic tree and are in line with theoretical predictions. In fact the efficiency of the algorithm increases as the branching parameter increases or efficiency enhancement techniques are added to the estimation procedure[95].

The best results are reached when pruning, antithetic variates and control variate are implemented with a relative error of less than 1%, a RMSE of 2.818% and a relative SE of 9.683%. Therefore this specific algorithm will be used in the pricing of derivative claims in the next sections. A discussion about the algorithm distribution is presented in Appendix G (Figure A11).

At the end it is important to highlight the fact that even though the two best estimators present a relative error of less than 1% the stochastic tree seems to be more accurate than the stochastic mesh as demonstrated by a lower relative SE and RMSE. This argument may lead to prefer the stochastic tree over the mesh in pricing problems. This issue will be further analyzed in the next sections where different types of derivative claims are considered because it is reasonable to think that the preference of an algorithm over the other may highly depend on the characteristics of the problem considered.

[95] In unreported tests, the same analysis has been extended singularly to the mesh and low estimators because asymmetric improvements on these estimators may mislead the interpretation of the results in the point estimate analysis. This, in turn, may lead to wrong conclusions about the efficiency of the algorithm. However the results of the separate analyses are aligned with the theory for both the mesh and the low estimators confirming the validity of the approach.

Figure 3.3: Stochastic Mesh Algorithm: Graphical Analysis



Data are from Table 3.

3.1.3 LSM

The results concerning the LSM algorithm are presented in Table 4 and Figure 3.4. Each row in the table was constructed by using 25 estimates of the derivative price. In Table 4 for each measure rows 1, 2 and 3 list the relevant results for standard LSM, control variate and control variate improved LSM (CVLSM) respectively. The European option value is used as the control variate. For the control variate and CVLSM algorithms pruning and antithetic variates have been implemented.

For the standard LSM the basis functions used in the regression are a constant and the first three Laguerre polynomials as described in section 2.2.3. However for the remaining two algorithms we followed Rasmussen (2005) and used the following basis functions:

$$L_0(X) = K,$$

$$L_1(X) = X,$$

$$L_2(X) = BS(X,t)$$

$$L_3(X) = XBS(X,t).$$

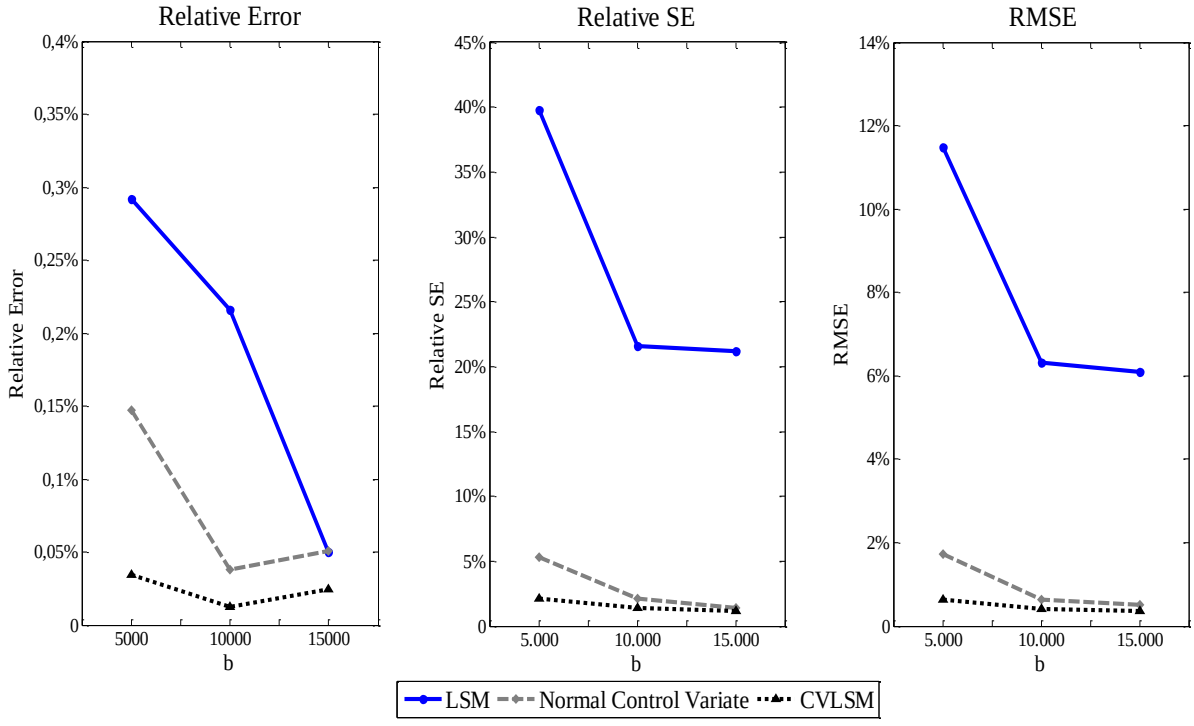
where K is the strike price, X is the current asset price and $BS(X,t)$ is the Black and Scholes European option price for a call option expiring at T with the same parameters as the American option.

Table 4: LSM Algorithm Efficiency Improvements Evaluation

	b=5000	b=10000	b=15000
LSM Relative Error	0.292%	0.216%	0.050%
LSM Relative Error Control	0.147%	0.038%	0.051%
CVLSM Relative Error	0.034%	0.012%	0.024%
LSM Relative SE	39.791%	21.592%	21.220%
LSM Relative SE Control	5.277%	2.113%	1.416%
CVLSM Relative SE	2.134%	1.432%	1.169%
LSM RMSE	11.491%	6.323%	6.084%
LSM RMSE Control	1.730%	0.644%	0.500%
CVLSM RMSE	0.642%	0.416%	0.363%

Option parameters: S=100, $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T. Each row was constructed with 25 estimates of the derivative price.
For each measure the first line was estimated using the original LSM algorithm, in the second line standard control variate was implemented and in the third the control variate improved LSM approach was used. European option was used as control variate.
To calculate Relative Error and RMSE the estimated price has been compared with the true price of 5.731 estimated with the formula in Geske and Johnson (1984).

Figure 3.4: LSM Algorithm: Graphical Analysis



Data are from Table 4.

The results demonstrate how the used techniques are effective in improving the efficiency of the estimator. In particular for the CVLSM algorithm the relative error is lower than 0.1%, the RMSE is

lower than 1% and the relative SE is lower than 2.5%. This will be the algorithm employed in the next sections. Appendix G (Figure A12) presents and analyses the algorithm distribution.

Overall the LSM algorithm seems the one that produces, among the methods considered, the most accurate estimates of the derivative claim. However the statement must be verified with other types of derivative securities before reaching a general conclusion.

3.2 Valuation of Derivative Securities

In the next sections we present the computational results obtained through the implementation of the various algorithms [96]on the pricing of different types of options. The purpose of these numerical experiments is to try to confirm and/or determine what are the strengths of each method in terms of ease of implementation, applicability, flexibility and computational efficiency.

3.2.1 Computational Testbed

The testbed consists of American call options on a single and on multiple underlying assets (2, 5 assets). These tests have already been used by Broadie and Glasserman (1997, 2004) and Longstaff and Schwartz (2001).

3.2.1.1 Testbed Set 1: Call Option with Continuous Dividends

Testbed set 1 is a call option (payoff $C = (S - K)^+$) on a single stock that pays continuous dividends at a rate δ but early exercise is restricted to discrete points $\{t = 0, T/3, 2T/3, T\}$. A complete presentation of this type of derivative has been already provided in section 3.1 therefore we will just discuss the specific set of parameters used in the analysis: time horizon $T = 1$ year; dividends $\delta = 0.1$; risk free rate $r = 0.05$; volatility $\sigma = 0.2$; strike price $K = 100$; initial stock price $S_0 = 70, 80, 90, 100, 110, 120$.

3.2.1.2 Testbed Sets 2-3: Max Options on Multiple Underlying Assets

Testbed sets 2-3 are call options on the maximum of k assets $(k = 2, 5)$. The payoff upon exercise of these options is $\left(max_{i=1,\dots,k} S^i - K\right)^+$. Under the risk-neutral measure asset prices are assumed to follow correlated geometric Brownian motion processes

$$dS_t^i = S_t^i\left[(r - \delta_i)dt + \sigma_i dz_t^i\right],$$

where z_t^i is a standard Brownian motion and the instantaneous correlation of z^i and z^j is ρ_{ij}. For simplicity in our numerical analysis we take $\delta_i = \delta$ and $\rho_{ij} = \rho$ for all $i, j = 1, \dots, k$ and $i \neq j$.

[96] All the algorithms are implemented in Matlab on a PC with a 2.1GHz Intel Core i7-3612QM processor.

The parameters used in the analysis are: $k = 2,5$; $T = 1$; $r = 0.05$; $\sigma = 0.2$; $\rho = 0, 0.3$; $K = 100$; $S_0^1 = S_0^2 = S_0^3 = S_0^4 = S_0^5 = 70, 80, 90, 100, 110, 120$.

3.2.2 American Option on a Single Asset

The results concerning the pricing of the American option on a single asset are shown in Tables 5 and 6.

For the stochastic tree and stochastic mesh algorithms the efficiency enhancement techniques of pruning, antithetic branching (antithetic variates) and control variate were implemented where we used as a control the value of a European option. Moreover for these algorithms we used 100 simulation runs for each estimate ($n = 100$). The control variate improved LSM algorithm (CVLSM) also benefited from pruning and antithetic variates. Each line was calculated from $N = 25$ estimates of the derivative price.

The confidence intervals are given by

$$\left(\theta - z_{\delta/2}\frac{std(\theta)}{\sqrt{N}}, \Theta + z_{\delta/2}\frac{std(\Theta)}{\sqrt{N}}\right)$$

$$\left(\hat{q} - z_{\delta/2}\frac{std(\hat{q})}{\sqrt{N}}, \hat{Q}(0,S_0) + z_{\delta/2}\frac{std(\hat{Q}(0,S_0))}{\sqrt{N}}\right)$$

for the stochastic tree and the stochastic mesh respectively. $z_{\delta/2}$ is the $1-\delta/2$ quantile of the normal distribution and $s(\cdot)$ are the sample standard deviations of the estimators. The point estimate is just the simple average of the high biased and low biased estimators as presented in section 3.1. The true value is computed with the analytical formula from Geske and Johnson (1984).

Concerning the stochastic tree and stochastic mesh, the results are consistent with the theoretical developments in Part II. The θ and $\hat{q}$ estimators are indeed biased low while the Θ and $\hat{Q}(0,S_0)$ estimators are biased high with $\theta < \Theta$ and $\hat{q} < \hat{Q}(0,S_0)$. Moreover all the estimators converge to the true price as the branching parameter is increased. Nevertheless the stochastic tree algorithm presents lower standard errors (for $b = 150$ range 0.0003-0.0031) than the stochastic mesh (for $b = 150$ range 0.0002-0.0072) and consequently tighter confidence intervals[97].

The CVLSM confirms its efficiency in the approximation of the option price as demonstrated by estimates which are really close to the true values and by considerably low standard errors (with $b = 10000$ range 0.0001-0.001). In order to compare the algorithms, Tables 7 and 8 present some useful statistics. By looking at relative errors[98] we can infer that all the algorithms do a great job

[97] As Broadie and Glasserman (1997) pointed out because of the bias of the estimators, the reported confidence intervals are conservative meaning that the true value will fall in the confidence interval more times, on average that suggested by the confidence level.

[98] The reported relative errors are based on more significant digits than are shown in the tables.

in the estimation of the derivative price with all the values being under 0.5%[99]. The average time column shows the computation time (in seconds) required to produce a single estimate for the algorithm considered.

Table 5: American Call Option on a Single Asset

b	Algorithm	Stock	High Estimator	SE	Low Estimator	SE	Point Estimate	SE	90% Confidence Interval		True Value
50	ST	70	0.121	0.0005	0.121	0.0005	0.121	0.0005	0.120	0.122	0.121
50	SM	70	0.122	0.0002	0.120	0.0005	0.121	0.0003	0.119	0.122	0.121
100	ST	70	0.121	0.0002	0.121	0.0002	0.121	0.0002	0.121	0.122	0.121
100	SM	70	0.122	0.0001	0.120	0.0003	0.121	0.0002	0.120	0.122	0.121
150	ST	70	0.121	0.0003	0.121	0.0003	0.121	0.0003	0.121	0.122	0.121
150	SM	70	0.122	0.0001	0.121	0.0003	0.121	0.0002	0.121	0.122	0.121
5000	CVLSM	70	-	-	-	-	0.121	0.0001	-	-	0.121
10000	CVLSM	70	-	-	-	-	0.121	0.0001	-	-	0.121
50	ST	80	0.669	0.0016	0.669	0.0016	0.669	0.0016	0.667	0.672	0.670
50	SM	80	0.686	0.0010	0.654	0.0020	0.670	0.0014	0.651	0.687	0.670
100	ST	80	0.671	0.0011	0.671	0.0011	0.671	0.0011	0.669	0.673	0.670
100	SM	80	0.682	0.0006	0.659	0.0011	0.671	0.0008	0.657	0.683	0.670
150	ST	80	0.672	0.0009	0.672	0.0009	0.672	0.0009	0.670	0.673	0.670
150	SM	80	0.680	0.0005	0.659	0.0011	0.670	0.0007	0.657	0.681	0.670
5000	CVLSM	80	-	-	-	-	0.670	0.0003	-	-	0.670
10000	CVLSM	80	-	-	-	-	0.670	0.0001	-	-	0.670
50	ST	90	2.305	0.0024	2.301	0.0024	2.303	0.0024	2.297	2.309	2.303
50	SM	90	2.400	0.0024	2.238	0.0039	2.319	0.0028	2.232	2.404	2.303
100	ST	90	2.305	0.0017	2.303	0.0018	2.304	0.0018	2.300	2.308	2.303
100	SM	90	2.368	0.0017	2.255	0.0037	2.311	0.0025	2.249	2.371	2.303
150	ST	90	2.301	0.0019	2.299	0.0019	2.300	0.0019	2.296	2.304	2.303
150	SM	90	2.356	0.0017	2.271	0.0036	2.314	0.0026	2.265	2.359	2.303
5000	CVLSM	90	-	-	-	-	2.303	0.0006	-	-	2.303
10000	CVLSM	90	-	-	-	-	2.302	0.0004	-	-	2.303

Option parameters: S=Stock Price as indicated in the table, $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3,$ and T. Each row was constructed with 25 estimates of the option price. European option was used as control variate. True prices were estimated with the formula in Geske and Johnson (1984).

An analysis of this measure highlights an expected direct relationship between the branching parameter and the computation time required, as well as the expected increase in computation time for the CVLSM as the option moves further into the money. This is because a higher number of in the money paths are included in the regression. Overall it is possible to notice the computation time handicap of the stochastic mesh compared to the tree in the estimation of this particular option; in fact it takes from 7 $(b = 50)$ to 20 $(b = 150)$ times more than the tree to produce an estimate. The CVLSM presents computation times which are similar to those of the tree but it shows a little

[99] Except two exceptions for the SM of 0.6034% and 0.6862% which remain anyway under the 1%.

advantage in pricing out of the money options as well as a slight disadvantage with in the money options where the percentage of paths included in the regression increases substantially.

Table 6: American Call Option on a Single Asset

b	Algorithm	Stock	High Estimator	SE	Low Estimator	SE	Point Estimate	SE	90% Confidence Interval		True Value
50	ST	100	5.739	0.0015	5.718	0.0015	5.729	0.0015	5.716	5.742	5.731
50	SM	100	5.962	0.0066	5.569	0.0128	5.766	0.0093	5.548	5.973	5.731
100	ST	100	5.737	0.0008	5.727	0.0009	5.732	0.0009	5.725	5.739	5.731
100	SM	100	5.877	0.0037	5.613	0.0091	5.745	0.0061	5.598	5.883	5.731
150	ST	100	5.733	0.0007	5.726	0.0007	5.729	0.0007	5.725	5.734	5.731
150	SM	100	5.851	0.0037	5.641	0.0069	5.746	0.0050	5.630	5.857	5.731
5000	CVLSM	100	-	-	-	-	5.730	0.0012	-	-	5.731
10000	CVLSM	100	-	-	-	-	5.730	0.0008	-	-	5.731
50	ST	110	11.359	0.0025	11.309	0.0034	11.334	0.0029	11.304	11.363	11.341
50	SM	110	11.632	0.0100	11.119	0.0142	11.375	0.0113	11.096	11.648	11.341
100	ST	110	11.351	0.0016	11.330	0.0018	11.341	0.0017	11.328	11.354	11.341
100	SM	110	11.521	0.0071	11.166	0.0113	11.344	0.0087	11.148	11.533	11.341
150	ST	110	11.347	0.0012	11.334	0.0013	11.340	0.0013	11.331	11.349	11.341
150	SM	110	11.470	0.0062	11.185	0.0094	11.327	0.0072	11.170	11.480	11.341
5000	CVLSM	110	-	-	-	-	11.337	0.0022	-	-	11.341
10000	CVLSM	110	-	-	-	-	11.339	0.0010	-	-	11.341
50	ST	120	20.024	0.0076	19.984	0.0100	20.019	0.0061	19.968	20.037	20.000
50	SM	120	20.087	0.0050	20.040	0.0038	20.063	0.0041	20.034	20.095	20.000
100	ST	120	19.994	0.0052	19.990	0.0057	20.001	0.0036	19.981	20.002	20.000
100	SM	120	20.022	0.0033	20.013	0.0025	20.017	0.0029	20.008	20.028	20.000
150	ST	120	19.998	0.0042	19.997	0.0043	20.003	0.0031	19.990	20.005	20.000
150	SM	120	20.006	0.0011	20.003	0.0010	20.004	0.0010	20.001	20.008	20.000
5000	CVLSM	120	-	-	-	-	20.000	0.0000	-	-	20.000
10000	CVLSM	120	-	-	-	-	20.000	0.0000	-	-	20.000

Option parameters: S=Stock Price as indicated in the table, $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T. Each row was constructed with 25 estimates of the option price. European option was used as control variate. True prices were estimated with the formula in Geske and Johnson (1984).

Considering the RMSE the results are similar to what already inferred for computation time, with the stochastic tree and CVLSM presenting better indicators than the mesh. However all the RMSE values are typically very small indicating a discrete level of accuracy in the estimates. At the end the last column presents the results relative to the coefficient described in equation (2.2) in section 2.1.1.1. It measures the efficiency of a certain algorithm in the presence of a time budget in the estimation. Again keeping fixed the algorithm parameters the stochastic tree presents an advantage over the mesh (lower coefficients); however the interesting result is that by increasing the branching parameter the coefficient gets worse for both ST and SM indicating that an increase in precision is more than outweighed by an increase in computation time.

Table 7: American Call Option on a Single Asset Statistics

b	Algorithm	Stock	Relative Error	Average Time	RMSE	Estimator Variance	Efficiency Coefficient
50	ST	70	0.063%	2.55	0.23%	0.000005605	0.0000143032
50	SM	70	0.221%	18.16	0.14%	0.000001908	0.0000346494
100	ST	70	0.401%	9.28	0.12%	0.000001298	0.0000120406
100	SM	70	0.000%	123.92	0.08%	0.000000723	0.0000895615
150	ST	70	0.276%	20.09	0.13%	0.000001782	0.0000357976
150	SM	70	0.335%	403.73	0.09%	0.000000755	0.0003047376
5000	CVLSM	70	0.133%	0.08	0.05%	0.000000220	0.0000000167
10000	CVLSM	70	0.185%	0.12	0.04%	0.000000084	0.0000000099
50	ST	80	0.109%	2.54	0.77%	0.000060794	0.0001546597
50	SM	80	0.032%	18.00	0.67%	0.000046243	0.0008321558
100	ST	80	0.136%	9.27	0.53%	0.000028420	0.0002633958
100	SM	80	0.114%	117.06	0.38%	0.000014728	0.0017241173
150	ST	80	0.286%	20.14	0.48%	0.000020536	0.0004135018
150	SM	80	0.041%	399.58	0.36%	0.000013183	0.0052678041
5000	CVLSM	80	0.016%	0.27	0.13%	0.000001796	0.0000004870
10000	CVLSM	80	0.018%	0.54	0.07%	0.000000495	0.0000002653
50	ST	90	0.006%	2.53	1.20%	0.000148879	0.0003759214
50	SM	90	0.686%	18.05	2.11%	0.000202257	0.0036513761
100	ST	90	0.031%	9.30	0.86%	0.000077319	0.0007189238
100	SM	90	0.359%	125.56	1.48%	0.000158520	0.0199031431
150	ST	90	0.126%	20.09	0.96%	0.000087636	0.0017609101
150	SM	90	0.458%	400.29	1.65%	0.000168180	0.0673216560
5000	CVLSM	90	0.020%	1.86	0.30%	0.000008908	0.0000165640
10000	CVLSM	90	0.036%	3.87	0.20%	0.000003296	0.0000127521

Statistics relative to Table 5.

For this reason, in order to achieve the best results in the presence of a time constraint, it would be optimal to keep the branching parameter relatively low while increasing the number of simulation runs n. CVLSM presents the lowest efficiency coefficient for all the options considered being the best choice in the presence of time constraints.

Overall we can conclude that in the case of a derivative security dependent on a single state variable all the considered algorithms produce satisfactory estimates; however the stochastic tree and CVLSM seem to present advantages over the mesh in terms of lower computation time and better precision.

Table 8: American Call Option on a Single Asset Statistics

b	Algorithm	Stock	Relative Error	Average Time	RMSE	Estimator Variance	Efficiency Coefficient
50	ST	100	0.038%	2.51	0.76%	0.000055638	0.0001399174
50	SM	100	0.603%	18.00	5.73%	0.002174255	0.0391453043
100	ST	100	0.016%	9.24	0.44%	0.000019261	0.0001779541
100	SM	100	0.243%	124.95	3.29%	0.000926365	0.1157490173
150	ST	100	0.030%	20.03	0.40%	0.000013457	0.0002695114
150	SM	100	0.263%	419.29	2.87%	0.000618887	0.2594903623
5000	CVLSM	100	0.024%	6.28	0.59%	0.000034816	0.0002185611
10000	CVLSM	100	0.014%	12.79	0.41%	0.000017029	0.0002178798
50	ST	110	0.060%	2.49	1.56%	0.000203047	0.0005064863
50	SM	110	0.302%	18.03	6.49%	0.003163657	0.0570527212
100	ST	110	0.002%	9.09	0.83%	0.000071217	0.0006477099
100	SM	110	0.024%	121.89	4.28%	0.001900510	0.2316565848
150	ST	110	0.006%	19.79	0.63%	0.000040696	0.0008055342
150	SM	110	0.120%	404.11	3.79%	0.001305620	0.5276089793
5000	CVLSM	110	0.031%	11.85	1.12%	0.000118663	0.0014066553
10000	CVLSM	110	0.014%	24.27	0.53%	0.000027011	0.0006554395
50	ST	120	0.096%	2.47	3.57%	0.000943738	0.0023328692
50	SM	120	0.316%	18.06	6.63%	0.000418355	0.0075561278
100	ST	120	0.004%	8.98	1.79%	0.000332095	0.0029829554
100	SM	120	0.087%	124.25	2.23%	0.000204730	0.0254383501
150	ST	120	0.016%	19.54	1.56%	0.000242204	0.0047316315
150	SM	120	0.021%	412.73	0.66%	0.000026989	0.0111394133
5000	CVLSM	120	0.000%	16.12	0.00%	0.000000000	0.0000000000
10000	CVLSM	120	0.000%	45.45	0.00%	0.000000000	0.0000000000

Statistics relative to Table 6.

3.2.3 American Option on the Max of Two Assets

The results about the pricing of the American option on the max of two assets are shown in Tables 9 and 10. In this particular case a time budget of approximately 450 seconds (570 seconds for Table 10) has been established. Based on the results about efficiency coefficients for the stochastic tree and the stochastic mesh algorithms discussed in section 3.2.2 we chose a relatively low branching parameter $b = 50$ and the time budget left was devoted to increase the number of simulation runs n. For both the methods antithetic branching (antithetic variates) and control variate were implemented.

We use as control variate the European option on the max of two assets.[100] The same option was used in pruning even though we calculate also an alternative version of the tree without it. The CVLSM was implemented with pruning, control variate and antithetic variates techniques.

Table 9: American Max-Option on Two Assets

b	n	Algorithm	Stock Price	High Estimator	SE	Low Estimator	SE	Estimator	SE	90% Confidence interval		True Value
50	100	ST	70	0.241	0.0009	0.241	0.0009	0.241	0.0009	0.240	0.243	0.241
50	285	ST*	70	0.243	0.0009	0.233	0.0010	0.238	0.0010	0.232	0.245	0.241
50	1465	SM	70	0.242	0.0001	0.238	0.0002	0.240	0.0001	0.238	0.242	0.241
244000	1	CVLSM	70	-	-	-	-	0.241	0.0000	-	-	0.241
50	100	ST	80	1.316	0.0018	1.315	0.0018	1.316	0.0018	1.312	1.319	1.312
50	285	ST*	80	1.314	0.0016	1.297	0.0017	1.305	0.0017	1.294	1.316	1.312
50	1465	SM	80	1.343	0.0003	1.267	0.0008	1.305	0.0005	1.266	1.343	1.312
144100	1	CVLSM	80	-	-	-	-	1.312	0.0001	-	-	1.312
50	100	ST	90	4.331	0.0038	4.325	0.0039	4.328	0.0039	4.318	4.337	4.333
50	285	ST*	90	4.351	0.0017	4.317	0.0017	4.334	0.0017	4.314	4.353	4.333
50	1465	SM	90	4.530	0.0011	4.162	0.0018	4.346	0.0013	4.159	4.531	4.333
35012	1	CVLSM	90	-	-	-	-	4.332	0.0005	-	-	4.333
50	100	ST	100	10.047	0.0035	10.024	0.0038	10.036	0.0036	10.018	10.053	10.043
50	285	ST*	100	10.076	0.0010	10.007	0.0012	10.041	0.0011	10.005	10.077	10.043
50	1465	SM	100	10.557	0.0021	9.684	0.0036	10.120	0.0026	9.678	10.560	10.043
13916	1	CVLSM	100	-	-	-	-	10.039	0.0013	-	-	10.043
50	100	ST	110	18.150	0.0022	18.103	0.0025	18.126	0.0023	18.099	18.154	18.125
50	285	ST*	110	18.179	0.0015	18.074	0.0016	18.127	0.0016	18.071	18.182	18.128
50	1465	SM	110	19.004	0.0035	17.563	0.0066	18.283	0.0048	17.552	19.009	18.127
9632	1	CVLSM	110	-	-	-	-	18.119	0.0021	-	-	18.127
50	100	ST	120	27.637	0.0019	27.572	0.0021	27.604	0.0020	27.568	27.640	27.608
50	285	ST*	120	27.671	0.0018	27.539	0.0019	27.605	0.0019	27.535	27.674	27.604
50	1465	SM	120	28.851	0.0040	26.810	0.0068	27.831	0.0051	26.799	28.858	27.604
8478	1	CVLSM	120	-	-	-	-	27.593	0.0037	-	-	27.604

Option parameters: $S^1 = S^2 =$ Stock Price as indicated in the table. Also $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$, $\rho = 0$, and four exercise opportunities at times $0, T/3, 2T/3,$ and T. Each row was constructed with 25 estimates of the option price.
(All algorithms implemented antithetic branching (antithetic variates) and control variate; Rainbow European option on the max of two assets was used as control variate in all the simulations; pruning at the last step was used for ST, SM and CVLSM; ST* is the tree algorithm without the implementation of pruning; simulation time was approximately 7.7 minutes per estimation on a PC with a 2.1GHz Intel Core i7-3612QM processor).

In this example, due to the presence of multiple state variables, we followed Rasmussen (2005) and used the following basis functions in the regression approximation:

[100] An analytical formula is provided by Stulz (1982).

$$L_0(X^1, X^2) = K,$$
$$L_1(X^1, X^2) = X^*$$
$$L_2(X^1, X^2) = (X^*)^2$$
$$L_3(X^1, X^2) = BS(X^*, t)$$
$$L_4(X^1, X^2) = \left(BS(X^*, t)\right)^2$$
$$L_5(X^1, X^2) = C^{max}(X^*, X^{**}, t)$$
$$L_6(X^1, X^2) = \left(C^{max}(X^*, X^{**}, t)\right)^2$$
$$L_7(X^1, X^2) = X^{**}$$
$$L_8(X^1, X^2) = X^* X^{**}$$

Table 10: American Max-Option on Two Assets

b	n	Algorithm	Stock Price	High Estimator	SE	Low Estimator	SE	Estimator	SE	90% Confidence interval		True Value
50	100	ST	70	0.238	0.0005	0.238	0.0005	0.238	0.0005	0.238	0.238	0.237
100	25	ST	70	0.235	0.0008	0.235	0.0008	0.235	0.0008	0.234	0.236	0.237
50	285	ST*	70	0.238	0.0007	0.229	0.0007	0.233	0.0007	0.229	0.238	0.237
244000	1	CVLSM	70	-	-	-	-	0.237	0.0000	-	-	0.237
50	100	ST	80	1.258	0.0020	1.258	0.0020	1.258	0.0020	1.258	1.258	1.259
100	25	ST	80	1.254	0.0032	1.254	0.0032	1.254	0.0032	1.249	1.259	1.259
50	285	ST*	80	1.264	0.0013	1.249	0.0013	1.257	0.0013	1.249	1.264	1.259
154200	1	CVLSM	80	-	-	-	-	1.259	0.0001	-	-	1.259
50	100	ST	90	4.074	0.0040	4.068	0.0040	4.071	0.0040	4.068	4.074	4.077
100	25	ST	90	4.078	0.0049	4.075	0.0050	4.077	0.0050	4.067	4.086	4.077
50	285	ST*	90	4.089	0.0014	4.058	0.0014	4.074	0.0014	4.058	4.089	4.077
43000	1	CVLSM	90	-	-	-	-	4.075	0.0003	-	-	4.077
50	100	ST	100	9.370	0.0027	9.349	0.0028	9.360	0.0027	9.349	9.370	9.361
100	25	ST	100	9.369	0.0038	9.359	0.0039	9.364	0.0039	9.353	9.376	9.361
50	285	ST*	100	9.390	0.0010	9.328	0.0010	9.359	0.0010	9.328	9.390	9.361
17800	1	CVLSM	100	-	-	-	-	9.357	0.0013	-	-	9.361
50	100	ST	110	16.946	0.0014	16.904	0.0015	16.925	0.0014	16.904	16.946	16.924
100	25	ST	110	16.935	0.0026	16.913	0.0028	16.924	0.0027	16.909	16.939	16.924
50	285	ST*	110	16.971	0.0012	16.878	0.0013	16.924	0.0012	16.878	16.971	16.924
12000	1	CVLSM	110	-	-	-	-	16.923	0.0015	-	-	16.924
50	100	ST	120	26.008	0.0013	25.946	0.0016	25.977	0.0014	25.946	26.008	25.980
100	25	ST	120	25.998	0.0027	25.966	0.0030	25.982	0.0028	25.962	26.002	25.980
50	285	ST*	120	26.041	0.0021	25.923	0.0021	25.982	0.0021	25.923	26.041	25.980
10000	1	CVLSM	120	-	-	-	-	25.967	0.0040	-	-	25.980

Option parameters: $S^1 = S^2 =$ Stock Price as indicated in the table. Also $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$, $\rho = 0.3$, and four exercise opportunities at times $0, T/3, 2T/3$, and T. Each row was constructed with 25 estimates of the option price.

(All algorithms implemented antithetic branching (antithetic variates); Rainbow European option on the max of two assets was used as control variate in all the simulations; pruning at the last step was used for ST and CVLSM; ST* is the tree algorithm without the implementation of pruning; simulation time was approximately 9.5 minutes per estimation on a PC with a 2.1GHz Intel Core i7-3612QM processor).

where $X^*$and X^{**} are the highest and second highest asset prices respectively, $BS(X^*, t)$ is the Black and Scholes call option price on the asset with the highest price and finally $C^{max}(X^*, X^{**}, t)$ is the two assets max-call option on assets with the highest and second highest prices.

Table 11: American Max-Option on Two Assets Statistics

Algorithm	Stock Price	Relative Error	Average Time per Estimate	RMSE
ST	70	0.014%	450.74	0.45%
ST*	70	1.191%	431.88	0.56%
SM	70	0.260%	457.83	0.09%
CVLSM	70	0.158%	459.97	0.04%
ST	80	0.241%	454.50	0.92%
ST*	80	0.565%	414.90	1.11%
SM	80	0.531%	466.43	0.73%
CVLSM	80	0.024%	497.09	0.05%
ST	90	0.119%	455.59	1.96%
ST*	90	0.009%	401.39	0.84%
SM	90	0.289%	467.67	1.41%
CVLSM	90	0.034%	475.83	0.30%
ST	100	0.074%	457.69	1.93%
ST*	100	0.020%	405.68	0.58%
SM	100	0.768%	463.25	7.81%
CVLSM	100	0.048%	483.76	0.81%
ST	110	0.008%	466.03	1.13%
ST*	110	0.005%	432.99	0.78%
SM	110	0.863%	462.12	15.82%
CVLSM	110	0.044%	491.12	1.31%
ST	120	0.013%	458.73	1.03%
ST*	120	0.003%	431.37	0.92%
SM	120	0.821%	481.08	22.81%
CVLSM	120	0.038%	497.72	2.08%

Statistics relative to Table 9.

Confidence intervals are computed as explained in section 3.2.2. Because the intervals are conservative the actual convergence is much greater than the nominal convergence. The point estimates are the averages of the corresponding high and low estimators. In Table 10 the true values are form Broadie and Glasserman (1997); in Table 9 we used as true values the estimates produced by the CVLSM algorithm with a branching parameter of $b = 1000000$[101].

[101] Comparison with the true values reported in Broadie and Glasserman (1997) indicated that this procedure resulted in an error of about 0.001.

Table 9 presents the results for uncorrelated underlying assets ($\rho = 0$). All the estimates present very limited standard errors (range 0.0001 – 0.0051) with the stochastic tree having lower errors than the mesh and CVLSM for at the money and in the money options.

Table 12: American Max-Option on Two Assets Statistics

Algorithm	Stock Price	Relative Error	Average Time per Estimate	RMSE
ST	70	0.311%	564.48	0.26%
ST (b=100)	70	0.836%	587.11	0.44%
ST*	70	1.607%	543.01	0.52%
CVLSM	70	0.204%	572.71	0.05%
ST	80	0.065%	564.72	1.00%
ST (b=100)	80	0.400%	586.38	1.68%
ST*	80	0.194%	545.97	0.71%
CVLSM	80	0.019%	592.00	0.04%
ST	90	0.148%	568.79	2.09%
ST (b=100)	90	0.008%	567.26	2.48%
ST*	90	0.082%	548.48	0.76%
CVLSM	90	0.047%	562.43	0.26%
ST	100	0.013%	611.06	1.37%
ST (b=100)	100	0.037%	559.14	1.97%
ST*	100	0.023%	498.28	0.54%
CVLSM	100	0.044%	580.90	0.76%
ST	110	0.006%	590.38	0.72%
ST (b=100)	110	0.000%	559.64	1.34%
ST*	110	0.003%	558.26	0.62%
CVLSM	110	0.005%	560.05	0.74%
ST	120	0.011%	601.60	0.74%
ST (b=100)	120	0.008%	570.93	1.42%
ST*	120	0.008%	557.65	1.06%
CVLSM	120	0.051%	558.25	2.35%

Statistics relative to Table 10.

Moreover the stochastic tree presents narrower confidence intervals than then mesh due to a mix of lower biases and lower standard errors in the estimators. An interesting finding is the computation time advantage of the stochastic mesh over the stochastic tree when the problem considered has multiple state variables. This can be inferred by the higher number of simulation runs necessary to cover the time budget for the stochastic mesh relative to the stochastic tree (1465 SM, 100 ST, 285 ST*) indicating a lower time cost per single run for the mesh. Moreover it is possible to notice how the pruning technique in this particular case increases the computation time for each estimate rather than decreasing it ($n = 100$ ST, $n = 285$ ST*). An in depth analysis of the algorithm showed that this

result is due to the high computational time required to compute analytically[102] the option on the max of two assets used in pruning.

The statistics in Table 11 signal the discrete precision of the estimates with a relative error lower than 1% for all the observations except one. With respect to the RMSE the stochastic tree and CVLSM algorithms show considerably low values of the indicator (approximately 2% or below) thus evidencing very limited dispersion of the estimates around the true values. For the stochastic mesh however the RMSE increases as the option moves further into the money. This can be motivated by the fact that the absolute amount of bias in the estimates increases as the option value gets bigger.

For a further in depth analysis of the algorithms we studied the two separate components of the mean squared error (MSE): variance, and squared of bias of the estimates[103]. Tables 13, 14 and 15 present the results. It is possible to notice how the stochastic tree method may substantially gain in efficiency by reducing the estimator variance maybe with additional variance reduction techniques. On the other hand the stochastic mesh algorithm is affected by significant bias that reduces its efficiency (approximately 90% of the MSE). Due to this specific issue the attention should be concentrated on bias reduction techniques in order to further improve the algorithm results. The CVLSM presents a balanced relationship between variance and bias showing potential improvements from both variance reduction as well as bias reduction techniques. However with such a low RMSE as presented in Table 11 we may state that the algorithm already achieves a satisfactory level of accuracy.

Tables 10 and 12 presents results and statistics when the correlation coefficient considered is $\rho = 0.3$. In this specific example the stochastic mesh price was not estimated due to the complexity in the derivation of conditional transition densities required for the weights when the two assets are correlated[104]. We substituted the mesh with a stochastic tree algorithm with an increased branching parameter $b = 100$. The results are very similar to the ones already discussed for the uncorrelated assets with very limited relative errors and RMSE as a proof of the estimates accuracy. An interesting remark however is the relative lower accuracy of the stochastic tree algorithm with the higher branching parameter compared to the standard version with $b = 50$.

[102] Using the Stulz formula.

[103] $MSE(\hat{X}) = Var(\hat{X}) + \left(Bias(\hat{X}, X)\right)^2$.

[104] For uncorrelated assets we followed Glasserman (2004) and the transition densities from one node to another were just the product of the one-dimensional densities. Specifically from one generic node $x = (x_1, x_2)$ to another $y = (y_1, y_2)$ over a time interval of length Δt the transition density is

$$f(x,y) = \prod_{i=1}^{2} \frac{1}{\sigma\sqrt{\Delta t} y_i} \phi\left(\frac{\log(y_i/x_i) - \left(r - \delta - \frac{1}{2}\sigma^2\right)\Delta t}{\sigma\sqrt{\Delta t}}\right)$$

with ϕ the standard normal density. This procedure was not appropriate when the correlation between assets had to be taken into account. To solve this problem it would have been possible to calculate the mesh weights with the methods described in Broadie et al. (2000) that do not use transition densities. However this task is beyond the scope of the study.

This can be noticed by the higher standard errors and RMSE values for the former algorithm compared to the latter. This confirms the fact that under time constraints a better accuracy is reached with the stochastic tree by increasing the number of simulation runs rather than the branching parameter.

Table 13: Stochastic Tree MSE Decomposition

MSE Decomposition					
Stochastic Tree					
Stock Price	MSE	VAR	BIAS	%VAR	% BIAS
70	1.32E-05	1.18E-05	1.44E-06	89%	11%
80	1.61E-04	1.56E-04	5.59E-06	97%	3%
90	5.17E-04	4.88E-04	2.98E-05	94%	6%
100	2.74E-04	2.74E-04	6.46E-07	100%	0%
110	1.38E-04	1.13E-04	2.56E-05	81%	19%
120	1.15E-04	1.15E-04	2.76E-08	100%	0%

Statistics relative to Table 9.

Table 14: CVLSM MSE Decomposition

MSE Decomposition					
CVLSM					
Stock Price	MSE	VAR	BIAS	%VAR	% BIAS
70	1.52E-07	6.66E-09	1.46E-07	4%	96%
80	2.39E-07	1.42E-07	9.71E-08	59%	41%
90	8.89E-06	6.76E-06	2.13E-06	76%	24%
100	6.55E-05	4.23E-05	2.32E-05	65%	35%
110	1.72E-04	1.10E-04	6.24E-05	64%	36%
120	4.34E-04	3.23E-04	1.10E-04	75%	25%

Statistics relative to Table 9.

Table 15: Stochastic Mesh MSE Decomposition

MSE Decomposition					
Stochastic Mesh					
Stock Price	MSE	VAR	BIAS	%VAR	% BIAS
70	7.99E-07	4.07E-07	3.92E-07	51%	49%
80	5.36E-05	5.08E-06	4.85E-05	9%	91%
90	1.98E-04	4.08E-05	1.57E-04	21%	79%
100	6.10E-03	1.56E-04	5.94E-03	3%	97%
110	2.50E-02	5.56E-04	2.45E-02	2%	98%
120	5.20E-02	6.18E-04	5.14E-02	1%	99%

Statistics relative to Table 9.

3.2.4 American Option on the Max of Five Assets

The last problem considered consists in the pricing of an American option on the maximum of five assets. Results may be found in Tables 16 and 17.

Stochastic tree was implemented with the use of the antithetic branching technique. Antithetic variates were used for LSM and stochastic mesh. In this example pruning could not be implemented due to the impossibility of calculating the 5 assets European option analytically. For the same reason it is difficult to implement the control variate technique. To reach a satisfactory level of variance reduction it would be necessary to find one or multiple controls that are strongly correlated with the option under consideration. Therefore the presented results do not employ the control variate technique[105].

In selecting the basis functions involved in the regression estimation for the LSM algorithm we followed Longstaff and Schwartz (2001) and chose a constant, the first five Hermite polynomials in the maximum of the values of the five assets, the four values and squares of the values of the second through fifth highest asset prices, the product of the highest and second highest, second highest and third highest, third highest and fourth highest, fourth highest and fifth highest and finally, the product of all five assets values for a total of 19 basis functions.

Relative errors and RMSE are not reported because the true price is unknown. A multinomial lattice with k assets and n time steps has on the order of n^k terminal nodes. With $k = 5$ the computations are prohibitive for n as small as 50. And, even if the computations could be done, the resulting values would not be accurate.

From Table 16 it is possible to notice how the LSM presents consistently the lowest standard errors among all the options considered. The stochastic tree and stochastic mesh present standard errors of similar size. It is interesting to notice how tight the confidence intervals produced by the tree are. In fact the interval half widths are within 1% of the midpoint of the interval except for $S = 70$ which is 1.77%. This provides a very good indication about the possible values of the true price.
The estimates produced by the mesh when the branching parameter is $b = 50$ present considerable bias as evidenced by the fact that the point estimates always lie outside the confidence intervals produced by the tree (except for very in the money options). The efficiency can be improved by increasing the branching parameter. When $b = 100$ the gap between the confidence intervals of the true price and the point estimates is considerable reduced. However the cost is a discrete increase in

[105] In unreported tests we tried to implement control variate by using as controls a rainbow option on the max of the first two assets or a combination of pairs of rainbow options involving four assets. However the variance reduction gains were more than outweighed by the additional computation time required. Broadie and Glasserman (1997) used as control an estimate of a European option on 5 assets. However to do so 5-variate normal probabilities need to be evaluated and this is a too complex task which is beyond the scope of the study.

the computation time required per estimate. The results confirm the necessity of variance reduction and efficiency improvement techniques for the mesh algorithm to produce acceptable estimates.

Table 16: American Max-Option on Five Assets

b	n	Algorithm	Stock Price	High Estimator	SE	Low Estimator	SE	Point Estimate	SE	90% Confidence Interval		Average Time
50	100	ST	70	0.600	0.002	0.586	0.002	0.593	0.002	0.582	0.603	249
50	450	SM	70	0.665	0.003	0.468	0.003	0.567	0.003	0.463	0.671	233
100	100	SM	70	0.669	0.005	0.487	0.004	0.578	0.004	0.480	0.677	299
250000	1	LSM	70	-	-	-	-	0.595	0.001	-	-	10
50	100	ST	80	3.105	0.006	3.085	0.005	3.095	0.005	3.076	3.114	250
50	450	SM	80	3.636	0.008	2.278	0.005	2.957	0.006	2.270	3.649	235
100	100	SM	80	3.616	0.014	2.437	0.010	3.027	0.011	2.421	3.639	295
250000	1	LSM	80	-	-	-	-	3.077	0.015	-	-	33
50	100	ST	90	9.251	0.011	9.204	0.011	9.228	0.011	9.186	9.269	249
50	450	SM	90	11.136	0.016	6.918	0.010	9.027	0.012	6.900	11.163	238
100	100	SM	90	10.889	0.015	7.457	0.013	9.173	0.011	7.435	10.913	297
250000	1	LSM	90	-	-	-	-	9.245	0.004	-	-	111
50	100	ST	100	18.744	0.012	18.666	0.012	18.705	0.012	18.646	18.764	249
50	450	SM	100	22.529	0.018	14.596	0.013	18.563	0.013	14.576	22.558	237
100	100	SM	100	21.974	0.033	15.599	0.022	18.787	0.023	15.564	22.028	295
250000	1	LSM	100	-	-	-	-	18.692	0.005	-	-	184
50	100	ST	110	29.691	0.012	29.594	0.012	29.643	0.012	29.575	29.711	249
50	450	SM	110	35.668	0.026	23.862	0.020	29.765	0.018	23.829	35.711	238
100	100	SM	110	34.697	0.035	25.232	0.026	29.964	0.026	25.189	34.754	295
250000	1	LSM	110	-	-	-	-	29.641	0.006	-	-	199
50	100	ST	120	41.002	0.017	40.890	0.017	40.946	0.017	40.862	41.031	250
50	450	SM	120	49.288	0.024	33.734	0.022	41.511	0.015	33.697	49.327	237
100	100	SM	120	48.011	0.042	35.338	0.040	41.675	0.037	35.272	48.081	295
250000	1	LSM	120	-	-	-	-	40.968	0.007	-	-	191

Option parameters: $S^1 = S^2 = S^3 = S^4 = S^5 =$ Stock Price as indicated in the table. Also $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$, $\rho = 0$, and four exercise opportunities at times $0, T/3, 2T/3,$ and T. Each row was constructed with 25 estimates of the option price.
(All algorithms implemented antithetic branching (antithetic variates); simulation were run on a PC with a 2.1GHz Intel Core i7-3612QM processor).

Considering the average computation time per estimate, again the LSM presents considerable advantages over the remaining algorithms. This is true even for in the money options where the high percentage of in the money paths causes an increase in the average computation time per estimate. Again the SM presents considerably lower computation time required for each simulation run than the tree. However the lower precision in the estimates compensates more than that. At the end the average computation time per estimate of 250 seconds for the stochastic tree represents a reasonable time for implementing the method in real life pricing problems.

The results for correlated assets are presented in Table 17. As for the two assets case the mesh estimates have been replaced by a tree algorithm with a higher branching parameter. The increased

Table 17: American Max-Option on Five Assets

b	n	Algorithm	Stock Price	High Estimator	SE	Low Estimator	SE	Point Estimate	SE	90% Confidence Interval		Average Time
50	100	ST	70	0.558	0.002	0.546	0.002	0.552	0.002	0.543	0.561	248
76	100	ST	70	0.554	0.002	0.547	0.002	0.550	0.002	0.544	0.556	600
25000	1	LSM	70	-	-	-	-	0.549	0.001	-	-	10
50	100	ST	80	2.709	0.005	2.691	0.005	2.700	0.005	2.683	2.718	250
76	100	ST	80	2.712	0.004	2.701	0.004	2.707	0.004	2.694	2.719	604
25000	1	LSM	80	-	-	-	-	2.693	0.013	-	-	30
50	100	ST	90	7.836	0.009	7.800	0.009	7.818	0.009	7.786	7.850	250
76	100	ST	90	7.831	0.007	7.808	0.007	7.820	0.007	7.796	7.844	599
25000	1	LSM	90	-	-	-	-	7.807	0.003	-	-	89
50	100	ST	100	15.904	0.008	15.841	0.008	15.904	0.008	15.829	15.916	249
76	100	ST	100	15.917	0.009	15.876	0.009	15.897	0.009	15.862	15.931	597
25000	1	LSM	100	-	-	-	-	15.884	0.004	-	-	150
50	100	ST	110	25.824	0.012	25.739	0.012	25.824	0.012	25.720	25.843	249
76	100	ST	110	25.830	0.011	25.775	0.011	25.802	0.011	25.757	25.848	614
25000	1	LSM	110	-	-	-	-	25.794	0.004	-	-	173
50	100	ST	120	36.548	0.018	36.447	0.018	36.548	0.018	36.417	36.578	249
76	100	ST	120	36.523	0.014	36.456	0.014	36.489	0.014	36.433	36.546	607
25000	1	LSM	120	-	-	-	-	36.492	0.005	-	-	190

Option parameters: $S^1 = S^2 = S^3 = S^4 = S^5 =$ Stock Price as indicated in the table. Also $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$, $\rho = 0.3$, and four exercise opportunities at times $0, T/3, 2T/3$, and T. Each row was constructed with 25 estimates of the option price.
(All algorithms implemented antithetic branching (antithetic variates); simulation were run on a PC with a 2.1GHz Intel Core i7-3612QM processor).

efficiency is reflected in lower standard errors and very narrow confidence intervals with the interval half widths being within 0.5% of the midpoint of the interval except for $S = 70$ which is 1.16%. However the cost of these gains is a considerable increase in the computation time per estimate which increased by a factor of 2.4 (600 seconds for $b = 76$ versus 250 seconds for $b = 50$) .

As a final step we analyzed the convergence rate of the LSM algorithm to see if the use of a high number of basis functions in the regression approximation, especially for the 5 assets case, compromises the stability of the algorithm. The results are presented in Table 18 and graphically in Figures 3.5, 3.6 and 3.7.

The important thing to notice is that the SE for the multidimensional cases are not exponentially larger (if comparing for the same number of paths) than in the 1 dimensional case, which demonstrate the power of the Longstaff Schwartz method. This indicates that it is possible to reach the same amount of accuracy as in the 1 dimensional case by slightly increasing the number of paths. This was the procedure implemented in the analysis which resulted in discretely small standard errors. Overall

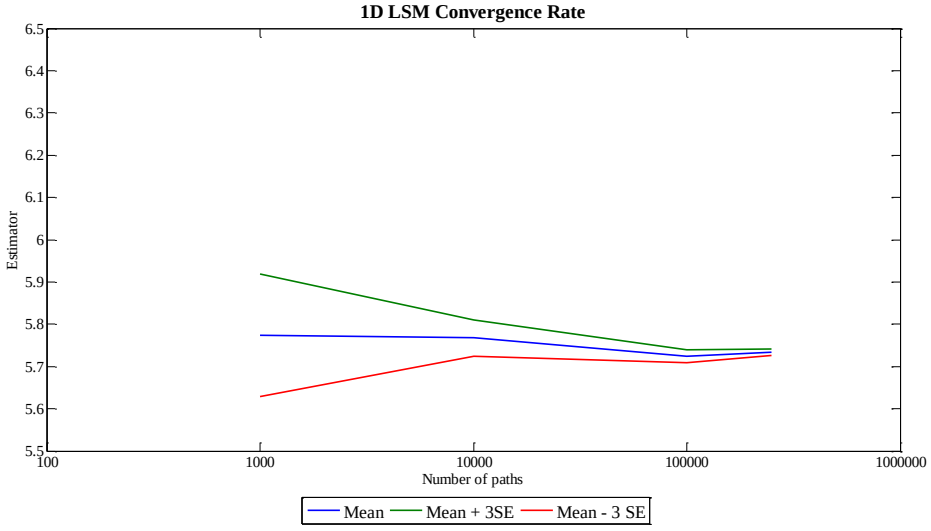
the results confirm a good stability of the LSM estimator and a rate of convergence that seems similar across the various dimensions considered[106]. This constitutes an important robustness test in favor of the methodology implemented in the numerical analysis.

Table 18: Convergence Results for the LSM Algorithm

Dimensions	Basis Functions	Estimate	b=1000	b=10000	b=100000	b=250000
1D	4	Mean	5.7739	5.7672	5.7237	5.7330
1D	4	SE	0.04844	0.01433	0.00516	0.00250
2D	10	Mean	10.0346	10.0461	10.0245	10.0249
2D	10	SE	0.05617	0.01541	0.00591	0.00453
5D	19	Mean	18.9018	18.7229	18.6857	18.6917
5D	19	SE	0.07375	0.02307	0.00774	0.00468

Option considered in the estimation: S=100, $K = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T. In the cases of 2 and 5 dimensions $\rho = 0$, $S^1 = S^2 = S^3 = S^4 = S^5 = 100$. Each row was constructed with 25 algorithm runs for each branching parameter considered.

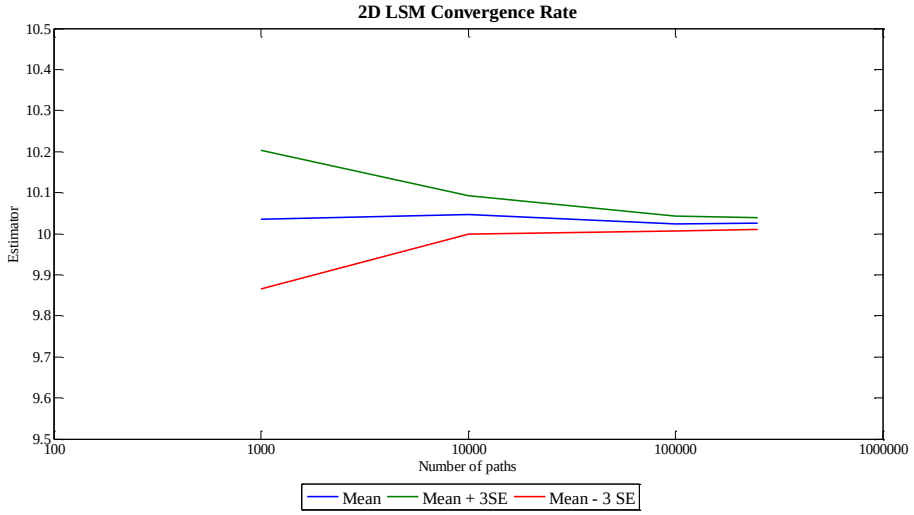
Figure 3.5: Convergence of the LSM Algorithm for an Option on a Single Asset



Source: Personal processing.

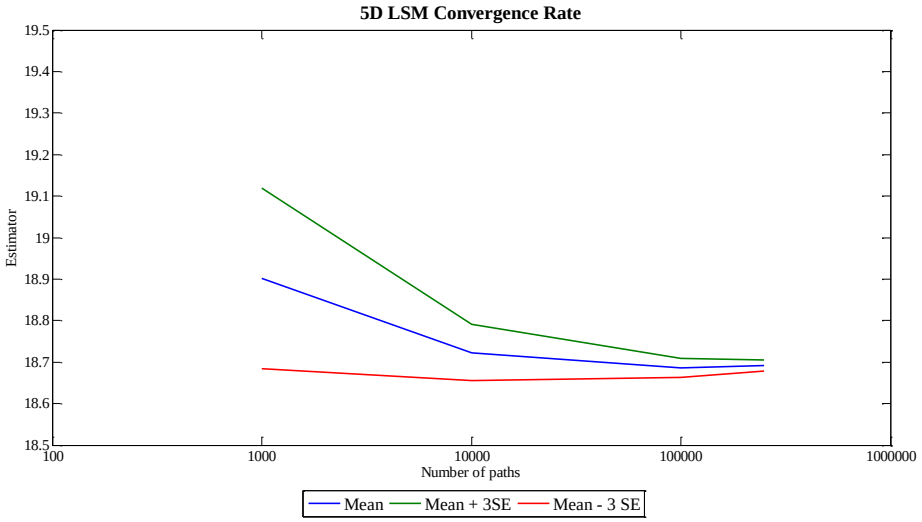
[106] The SE decreases approximately by a factor of 3 for every factor of 10 that we increase the number of paths by. This is true for all the dimensions considered.

Figure 3.6: Convergence of the LSM Algorithm for an Option on the Max of Two Assets



Source: Personal processing.

Figure 3.7: Convergence of the LSM Algorithm for an Option on the Max of Five Assets



Source: Personal processing.

PART IV: Discussion and Conclusions

4.1 Discussion of the Results

Our analysis confirmed the crucial importance of variance reduction and efficiency enhancement techniques for the implementation of the methods considered. The stochastic mesh, stochastic tree and LSM algorithms in their standard versions suffer from a considerable variation in the estimates. This may jeopardize the accuracy of the results if an adequately high computation time is not devoted to the simulation. The techniques of pruning, antithetic branching (antithetic variates) and control variate worked remarkably well in improving the efficiency. In fact, for the stochastic tree and LSM, they were responsible for approximately a reduction of 90% in the relative standard errors and RMSE compared to the standard versions. The results were slightly worse for the stochastic mesh which benefited only by an error reduction of about 50%. This indicates that the mesh estimator presents substantial room for accuracy improvements that need additional study. Among the algorithms considered the LSM seemed the one who showed the best improvements in accuracy even though the stochastic tree presented statistics of similar size but slightly worse. Overall the efficiency gains resulting from the implementation of the described techniques may permit to lower the value of the branching parameter and the number of simulation runs necessary to reach an acceptable level of precision in a restricted time interval.

Considering the one dimensional option pricing problem, the analysis suggested that the LSM algorithm performs best in terms of both accuracy and computational efficiency even though its computation time increases as the option moves further into the money. The stochastic tree and stochastic mesh also provided satisfactory estimates with a relative error lower than 1% and extremely limited standard errors. However the tree seems to be preferred for the determination of a confidence interval because, due to its lower bias and standard error in the estimators, it provides a tighter interval than the mesh. The interesting result concerns the relation between the branching parameter b, the number of simulation runs n and the algorithms efficiency under time constraints. In fact it has been pointed out that both the tree and the mesh would gain more in terms of accuracy form an increase in the simulation runs rather than an increase in the number of simulated paths when external constraints limit the time availability for the simulation. The result follows from the fact that, increasing the branching parameter, the gain in variance reduction is more than outweighed by the increase in computation time. This point may be of particular importance for practitioners implementing these methods since they often have to come up with reasonable price estimates or confidence intervals in a limited time.

The 2-dimensional option pricing problem demonstrated one of the advantages of the Monte Carlo method, which is the ability to efficiently deal with complex problems under a realistic time budget. All the produced estimates presented considerably low standard errors and relative errors lower than 1%. The stochastic tree and LSM also had low RMSE for all the options considered. An important remark is the difference in time efficiency between the stochastic tree and stochastic mesh when the number of state variables changes. The analysis pointed out how the stochastic mesh became computationally more efficient than the tree when dealing with multi-dimensional problems. Unfortunately, unless additional efficiency enhancement techniques are implemented, the mesh presented a considerable bias and estimates variance that more than outweighed the time advantage. The MSE decomposition analysis provided good information about the sources of error for each algorithm. Specifically the mesh suffered from a considerable bias which is responsible of about 90% of the error while for the tree 90% of error was due to estimator variance. This suggests the need to concentrate in future research on enhancements that may reduce the bias for the stochastic mesh and on application of additional variance reduction techniques for the stochastic tree.

The 5-dimensional option demonstrated that for extremely complex pricing problems, where analytical solutions are non-existent and traditional numerical procedures break down, the Monte Carlo method is the only alternative available to compute a price estimate. The produced estimates were characterized by standard errors generally lower than 1%. This in spite of the fact that control variate techniques were not implemented due to the difficulty to find a mix of controls capable of producing consistent variance reductions without increasing computation time exponentially. For this particular case the mesh estimator showed substantial bias. This makes it unsuitable for practical problems unless a large amount of time required to improve accuracy by increasing the branching parameter is available for the simulation. On the other hand, the stochastic tree and LSM algorithms proved their ability of producing accurate point estimates and confidence intervals in realistic time windows thus confirming to be important tools for the analysis of complex exotic derivatives.

At the end, the convergence properties of the LSM algorithm were studied. The analysis found good convergence of the estimates across various numbers of basis functions so long as sufficiently many paths were used in the simulation. For the multi-dimensional options, it has been shown that an advantage of the LSM algorithm is that the number of extra paths required to obtain a certain level of accuracy does not increase exponentially with the dimensionality of the problem.

4.2 Limitations

The results considered in the study are limited to a finite number of exercise opportunities. The limited technology used to run the simulation together with the exponential growth of computation time for the stochastic tree algorithm with the increase of exercise times limited the number of

opportunities that was possible to consider. In practice, many securities offer limited exercise opportunities. In the case of an equity call option written on an underlying that pays discrete dividends, early exercise is optimal just prior to ex-dividend dates. This implies that for options whose maturity is one year or less there will be generally at most four opportunities for optimal early exercise.[107] Another example is over-the-counter options which often have structured exercise opportunities, that is exercise is allowed only at a fixed number of predetermined dates.

Nevertheless, in order to implement the described methods on derivatives that allow for continuous exercise, two solutions may be adopted. The first is to use an extrapolation procedure. Broadie and Glasserman (1997) implemented the Richardson extrapolation to approximate the value of a continuous exercisable option with a finite set of Bermudan options. Their results were quite accurate however the risk is that this specific method uses an approximation of already approximated values (the Bermudan options) with the possibility that the errors in the option estimates compound when used to produce the American option price. This may jeopardize the efficiency of the procedure especially for complex derivatives. The second solution is to use parallel computing to reduce the estimation time of the procedure. With the extraordinary advance in technology infrastructures implemented by financial institutions this solution seems to be the preferred one for the pricing of complex American style derivatives.

With respect to the control variate techniques the analysis showed that considerable variance reduction gains are achieved only when a suitable mix of controls which presents a high correlation (positive or negative) with the derivatives under consideration is determined. Otherwise the additional computational time required for the estimation of the control prices may more than outweigh the gains from a decreased variance. Therefore, especially when dealing with derivatives characterized by different sources of risk or complex payoffs, an in depth ex ante analysis is required to determine if a proper mix of controls is available.

At the end, the obtained results are partly dependent on the implicit assumptions underlying the models and the inputs. In this study we assumed that stock prices may be well described by geometric Brownian motions and that volatilities and the risk free rate are constant over the life of option. These assumptions may be too simplistic since asset price returns are characterized by higher kurtosis ("fat tails") than the normal distribution and in addiction they occasionally present discontinuities. Moreover, both volatility and the risk free rate change over time thus presenting a stochastic behavior. These issues may be overcome thanks to the flexibility of the Monte Carlo simulation. In particular,

[107] For the Italian derivative market (IDEM) much of the options on single stocks are quoted by authorized market makers for the four closest trimestral expirations plus the two closest monthly expirations. Thus, no options are quoted regularly on the Italian exchange with expiration beyond one year. Only for a restricted group of very liquid stocks (8 stocks) market makers are obliged to quote options with expirations up to two years. However the trading volumes for these derivative securities are concentrated on the options with shorter maturities because for longer dated options the real challenge is a proper estimate of implied volatility which may have enormous impact on prices since the high vega of the options.

alternative processes may be simulated for the stock prices to match more closely empirical distributions and new stochastic processes may be introduced to describe the dynamics of volatility and risk free rate. This can be done without the need to completely discard or change the model under consideration. However it will be the task of future research to analyze how such changes may impact the results.

4.3 Conclusions

American-style securities are increasingly common. With this comes a growing need for methods to price and hedge these securities especially when their values depend on multiple assets or multiple state variables.

This thesis provides an extensive analysis of three famous algorithms for pricing such securities through Monte Carlo simulation: the stochastic tree, the stochastic mesh and the least-squares method (LSM).

Practical success of the methods depends critically on the use of effective variance reduction and efficiency enhancement techniques. On a preliminary study it has been showed how the implementation of these techniques may substantially improve the performance of the algorithms. The error reduction is substantial for the stochastic tree and the LSM with more than 90% of the error eliminated while for the stochastic mesh the error reduction is limited to 50%.

Algorithms performance is further analyzed using a number of realistic examples including the valuation of an American call option on a dividend paying asset and options written on the maximum of several assets. The LSM resulted to be best algorithm in terms of both accuracy of the estimates and computation time across the examples considered. This in spite of the fact that the computational effort increases for deep in the money options. The stochastic tree also presented a considerable amount of estimates precision that however should be traded-off with a higher computational cost especially for multi-dimensional problems. The stochastic mesh, even though presented attractive time efficiency especially when there were multiple sources of risk in the problem, it suffered from discrete variance and bias in the estimators which negatively impacted the final accuracy of the estimates.

The convergence analysis of the LSM algorithm demonstrated good convergence properties of the estimates. Specifically it showed that to obtain a certain level of accuracy the number of simulated paths does not need to increase exponentially with the dimensionality of the problem.

Overall the simulation framework seems to work considerably well in valuing American style derivative securities and when multi-dimensional problems are considered, simulation based methods seem to be the best solution to estimate prices since the general finite difference and binomial trees techniques become impractical in these specific situations.

This work can be extended in several directions. Alternative variance reduction techniques, including other control variates could be tested. Quasi Monte Carlo methods are another promising avenue of exploration to reduce both variance and computational time. Moreover investigation of alternative processes that can be used for the simulation of asset prices as well as models that include stochastic volatility and interest rate should be analyzed. At the end, an important topic not investigated here is the calculation of price sensitivities for hedging purpose. In real life problems, the knowledge about risk exposures is as important as the knowledge about fair prices.

The future research should focus on these issues in order to expand the applicability and improve the performance of simulation methods for valuing and risk managing complex derivatives.

References

Abramowitz, M. & Stegun, I.A. 1965, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, Dover publications.

Ames, W. "Numerical methods for partial differential equations. 1992", .

Avramidis, A.N. & Hyden, P. 1999, "Efficiency improvements for pricing American options with a stochastic mesh", *Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future-Volume 1*ACM, , pp. 344.

Avramidis, A.N. & Matzinger, H. 2002, "Problems in financial engineering: convergence of the stochastic mesh estimator for pricing American options", *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*Winter Simulation Conference, , pp. 1560.

BARONE-ADESI, G. & Whaley, R.E. 2012, "Efficient analytic approximation of American option values", *The Journal of Finance*, vol. 42, no. 2, pp. 301-320.

Barraquand, J. & Martineau, D. 1995, "Numerical valuation of high dimensional multivariate American securities", *Journal of Financial and Quantitative Analysis*, vol. 30, no. 03, pp. 383-405.

Black, F. & Scholes, M. 1973, "The pricing of options and corporate liabilities", *The journal of political economy*, , pp. 637-654.

Boyle, P., Broadie, M. & Glasserman, P. 1997, "Monte Carlo methods for security pricing", *Journal of Economic Dynamics and Control*, vol. 21, no. 8-9, pp. 1267-1321.

Boyle, P.P. 1977, "Options: A monte carlo approach", *Journal of Financial Economics*, vol. 4, no. 3, pp. 323-338.

Brennan, M.J. & 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.

Broadie, M. & Detemple, J. 1996, "American option valuation: new bounds, approximations, and a comparison of existing methods", *Review of Financial Studies*, vol. 9, no. 4, pp. 1211-1250.

Broadie, M. & Glasserman, P. 2004, "A stochastic mesh method for pricing high-dimensional American options", *Journal of Computational Finance*, vol. 7, pp. 35-72.

Broadie, M. & Glasserman, P. 1997, "Pricing American-style securities using simulation", *Journal of Economic Dynamics and Control*, vol. 21, no. 8-9, pp. 1323-1352.

Broadie, M. & Glasserman, P. 1996, "Estimating security price derivatives using simulation", *Management science*, , pp. 269-285.

Broadie, M., Glasserman, P. & Ha, Z. 2000, "Pricing American options by simulation using a stochastic mesh with optimized weights", *NONCONVEX OPTIMIZATION AND ITS APPLICATIONS*, vol. 49, pp. 26-44.

Broadie, M., Glasserman, P. & Jain, G. 1997, "Enhanced Monte Carlo estimates for American option prices", *The Journal of Derivatives*, vol. 5, no. 1, pp. 25-44.

Carriere, J.F. 1996, "Valuation of the early-exercise price for options using simulations and nonparametric regression", *Insurance: mathematics and Economics*, vol. 19, no. 1, pp. 19-30.

Charnes, J.M. 2000, "Using simulation for option pricing", *Simulation Conference Proceedings, 2000. Winter*IEEE, , pp. 151.

Clément, E., Lamberton, D. & Protter, P. 2002, "An analysis of a least squares regression method for American option pricing", *Finance and Stochastics*, vol. 6, no. 4, pp. 449-471.

Cox, J.C., Ross, S.A. & Rubinstein, M. 1979, "Option pricing: A simplified approach", *Journal of Financial Economics*, vol. 7, no. 3, pp. 229-263.

Duan, J.C. 2006, "The GARCH option pricing model", *Mathematical finance*, vol. 5, no. 1, pp. 13-32.

Duffie, D. 2008, *Dynamic asset pricing theory*, Princeton University Press.

Fu, M.C., Laprise, S.B., Madan, D.B., Su, Y. & Wu, R. 2001, "Pricing American options: A comparison of Monte Carlo simulation approaches", *Journal of Computational Finance*, vol. 4, no. 3, pp. 39-88.

Garman, M.B. 1976, *A General Theory of Asset Valuation under Diffusion State Processes*, No 50, Research Program in Finance Working Papers, University of California at Berkeley., .

Geske, R. & Johnson, H.E. 1984, "The American put option valued analytically", *Journal of Finance*, , pp. 1511-1524.

Geske, R. & Shastri, K. 1985, "Valuation by approximation: a comparison of alternative option valuation techniques", *Journal of Financial and Quantitative Analysis*, vol. 20, no. 01, pp. 45-71.

Glasserman, P. 2004, *Monte Carlo methods in financial engineering*, Springer Verlag.

Glasserman, P. & Yu, B. 2004, "Number of paths versus number of basis functions in American option pricing", *The Annals of Applied Probability*, vol. 14, no. 4, pp. 2090-2119.

Glynn, P.W. & Whitt, W. 1992, "The asymptotic efficiency of simulation estimators", *Operations research*, vol. 40, no. 3, pp. 505-520.

Grant, D., Vora, G. & Weeks, D. 1997, "Path-dependent options: Extending the Monte Carlo simulation approach", *Management Science*, , pp. 1589-1602.

Grant, D., Vora, G. & Weeks, D. 1996, "Simulation and the Early Exercise Option Problem", *J.OF FINANCIAL ENGINEERING*, vol. 5, no. 3.

Hammersley, J.M., Handscomb, D.C. & Weiss, G. 1965, "Monte carlo methods", *Physics Today*, vol. 18, pp. 55.

Hammersley, J. & Morton, K. 1956, "A new Monte Carlo technique: antithetic variates", *Mathematical Proceedings of the Cambridge Philosophical Society*Cambridge Univ Press, , pp. 449.

Heath, D., Jarrow, R. & Morton, A. 1990, "Bond pricing and the term structure of interest rates: A discrete time approximation", *Journal of Financial and Quantitative Analysis*, vol. 25, no. 4, pp. 419-440.

Hull, J. 2009, *Options, futures and other derivatives*, Pearson Prentice Hall.

Hull, J. & White, A. 1993, "One-factor interest-rate models and the valuation of interest-rate derivative securities", *Journal of financial and quantitative analysis*, vol. 28, no. 2.

Hull, J. & White, A. 1990, "Valuing derivative securities using the explicit finite difference method", *Journal of Financial and Quantitative Analysis*, vol. 25, no. 1, pp. 87-100.

Ibanez, A. & Zapatero, F. 2004, "Monte Carlo valuation of American options through computation of the optimal exercise frontier", *Journal of Financial and Quantitative Analysis*, vol. 39, no. 2.

Kemna, A.G.Z. & Vorst, A. 1990, "A pricing method for options based on average asset values", *Journal of Banking & Finance*, vol. 14, no. 1, pp. 113-129.

Lavenberg, S.S. & Welch, P.D. 1981, "A perspective on the use of control variables to increase the efficiency of Monte Carlo simulations", *Management Science*, vol. 27, no. 3, pp. 322-335.

Longstaff, F.A. & 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.

McCracken, D.D. & Dorn, W.S. 1965, "Numerical methods and FORTRAN programming with applications in engineering and science", *Numerical methods and FORTRAN programming with applications in engineering and science, by McCracken, Daniel D.; Dorn, William S.New York, Wiley [1965]*, vol. 1.

Merton, R.C. 1973, "Theory of rational option pricing", *The Bell Journal of Economics and Management Science*, , pp. 141-183.

Meyer, H.A. 1957, "Symposium on Monte Carlo Methods", *Bull.Amer.Math.Soc.63 (1957), 157-160.DOI: 10.1090/S0002-9904-1957-10117-7 PII: S*, vol. 2, no. 9904, pp. 10117-10117.

Musiela, M. & Rutkowski, M. 2011, *Martingale methods in financial modelling*, Springer.

Neftci, S.N. 2000, *An introduction to the mathematics of financial derivatives*, Academic Press.

Rasmussen, N.S. 2005, "Control variates for Monte Carlo valuation of American options", *Journal of Computational Finance*, vol. 9, no. 1, pp. 83.

Roll, R. 1977, "An analytic valuation formula for unprotected American call options on stocks with known dividends", *Journal of Financial Economics*, vol. 5, no. 2, pp. 251-258.

Rubinstein, R.Y., Samorodnitsky, G. & Shaked, M. 1985, "Antithetic variates, multivariate dependence and simulation of stochastic systems", *Management science*, vol. 31, no. 1, pp. 66-77.

Shreider, Y.A., Buslenko, N.P. & Parkyn, D. 1966, *The Monte Carlo method: the method of statistical trials*, Pergamon Press New York.

Stulz, R.M. 1982, "Options on the minimum or the maximum of two risky assets: analysis and applications", *Journal of Financial Economics*, vol. 10, no. 2, pp. 161-185.

Tilley, J.A. 1993, "Valuing American options in a path simulation model", *Transactions of the Society of Actuaries*, vol. 45, no. 83, pp. 104.

Tsitsiklis, J.N. & Van Roy, B. 1999, "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives", *Automatic Control, IEEE Transactions on*, vol. 44, no. 10, pp. 1840-1851.

Whaley, R.E. 1981, "On the valuation of American call options on stocks with known dividends", *Journal of Financial Economics*, vol. 9, no. 2, pp. 207-211.

Appendix

APPENDIX A: Change of Measure

In presenting the procedure we follow Glasserman (2004).

Let X be a non-negative random variable on $(\Omega, \mathcal{F}, P)$ with $E[X] = 1$. Define $Q: \mathcal{F} \to [0,1]$ by setting

$$Q(A) = E[1_A X] = \int_A X(\omega) dP(\omega), \quad A \in \mathcal{F}, \tag{A.1}$$

where 1_A represents the indicator function of the set A. It is possible to verify that the function Q is a probability measure on $(\Omega, \mathcal{F})$. It is continuous in P in the sense that

$$Q(A) > 0 \Rightarrow P(A) > 0 \quad for\ every\ A \in \mathcal{F}.$$

The Radon-Nikodym Theorem states that all such measures arise in this way. Specifically if P and Q are probability measures on $(\Omega, \mathcal{F})$ and if Q is continuous with respect to P than there exist a unique random variable X which justifies the equality in (A.1). Since Q is a probability measure the following equality must holds

$$P(X \geq 0) = 1 \Rightarrow \int_\Omega X(\omega) dP(\omega) = 1.$$

The variable X is generally represented as dQ/dP and called the Radon-Nikodym derivative or likelihood ratio of Q with respect to P. If P is also absolutely continuous with respect to Q then P and Q are equivalent in the sense that they agree about which events have probability zero.

To illustrate these ideas suppose the random variable Y on $(\Omega, \mathcal{F})$ has a density g under the measure P

$$P(Y \leq y) = \int_{-\infty}^{y} g(x) dx\,.$$

Consider now f to be another density function satisfying the property $g(y) = 0 \Rightarrow f(y) = 0$ and define a new probability measure Q on $(\Omega, \mathcal{F})$ by setting

$$Q(A) = E_P\left[1_A \frac{f(Y)}{g(Y)}\right],$$

where E_P is the expectation operator under the measure P. The ratio $f(y)/g(y)$ is considered to be equal to 1 whenever both terms are 0. It follows that

$$P(f(Y)/g(Y) \geq 0) = 1 \Rightarrow \int_\Omega \frac{f\big(Y(\omega)\big)}{g\big(Y(\omega)\big)} dP(\omega) = \int_{-\infty}^{\infty} \frac{f(x)}{g(x)} g(x) dx = 1\,.$$

Under the probability measure Q the event $(Y \leq y)$ has probability

$$Q(Y \le y) = \int_{(\omega: Y(\omega) \le y)} \frac{f\big(Y(\omega)\big)}{g\big(Y(\omega)\big)} dP(\omega)$$
$$= \int_{-\infty}^{y} \frac{f(x)}{g(x)} g(x) dx$$
$$= \int_{-\infty}^{y} f(x) dx \,.$$

Thus under the new measure Q the random variable Y has density f.

As an example consider g to be the standard normal density and let f be the normal density with mean μ and unit variance. It follow that $f(x)/g(x) = exp(\mu x - \mu^2/2)$. If Y follows the standard normal distribution under the measure P and if we define Q by setting

$$\frac{dQ}{dP} = e^{-\frac{1}{2}u^2 + \mu Y},$$

then Y has mean μ under the new measure Q and $Y - \mu$ has the standard normal distribution under Q. Similarly if, under some probability measure P_n, the random variables $Y_1, \dots, Y_n$ are independent with densities $g_1, \dots, g_n$, and if Q_n is defined as follow

$$\frac{dQ_n}{dP_n} = \prod_{i=1}^{n} \frac{f_i(Y_i)}{g_i(Y_i)},$$

then under Q_n, the Y_i are independent with densities f_i. In particular if the Y_i are independent standard normal under P_n and we set

$$\frac{dQ_n}{dP_n} = exp\left(-\frac{1}{2} \sum_{i=1}^{n} u_i^2 + \sum_{i=1}^{n} u_i Y_i \right),$$

then $Y_1 - \mu_1, \dots, Y_n - \mu_n$ are independent standard normal under Q_n.

The main interest in measure transformation is the change of measure that modify the drift of a Brownian motion.

Girsanov's Theorem

In presenting the theorem formulation we follow Glasserman (2004).

Let's start with a generalization of the transformation in (A.1). For the filtration $\{\mathcal{F}_t\}$ of $(\Omega, \mathcal{F}, P)$ let P_t denote the restriction of P to $\mathcal{F}_t$. Let $\{X(t), t \in [0,T]\}$ be a non-negative martingale with respect to $\{\mathcal{F}_t\}$ and suppose $E[X(T)] = 1$. With Q_t define a probability measure on $\mathcal{F}_t$ by setting

$$Q_t(A) = E_P[1_A X(t)] = E_{P_t}[1_A X(t)], \;\; A \in \mathcal{F}_t;$$

that is for each $t \in [0,T]$

$$\frac{dQ_t}{dP_t} = X(t).$$

Then Q_t is the restriction of Q_T to $\mathcal{F}_t$because for any $A \epsilon \mathcal{F}_t$

$$Q_T(A) = E_P[1_A X(T)] = E_P\left[1_A E_P[X(T)|\mathcal{F}_t]\right] = E_P[1_A X(t)] = Q_t(A).$$

In this sense the measures $\{Q_t, t \in [0,T]\}$ are consistent. If X is strictly positive then Q_t and P_t are equivalent for all t.

Suppose now that P and Q are equivalent probability measure on $(\Omega, \mathcal{F})$ with Radon-Nikodym derivative dQ/dP. Define

$$\left(\frac{dQ}{dP}\right)_t = E_P\left[\frac{dQ}{dP}|\mathcal{F}_t\right], \qquad t \in [0,T].$$

We claim that the ratio $(dQ/dP)_t$ is a martingale equals to the Radon-Nikodym derivative of the restriction of Q to $\mathcal{F}_t$ with respect to the restriction of P to $\mathcal{F}_t$. From the definition it is possible to infer the martingale property. For the second claim observe that for any $A \epsilon \mathcal{F}_t$,

$$E_P\left[1_A\left(\frac{dQ}{dP}\right)_t\right] = E_P\left[E_P\left[1_A \frac{dQ}{dP}|\mathcal{F}_t\right]\right] = E_P\left[1_A \frac{dQ}{dP}\right] = Q(A).$$

In summary a nonnegative, unit-mean martingale defines a consistent (with respect to $\{\mathcal{F}_t, t \in [0,T]\}$) family of probability measure and conversely the Radon-Nikodym derivatives for such a family define a unit-mean, nonnegative martingale.

In the following let $\{z(t), t \in [0,T]\}$ denote a standard k-dimensional Brownian motion on $(\Omega, \mathcal{F}, P)$ and let $\{\mathcal{F}_t^z, t \in [0,T]\}$ denote the filtration generated by z augmented to include all subsets of sets having P −probability 0.

Theorem (Girsanov Theorem): Let γ be a process adapted to $\{\mathcal{F}_t^z\}$ satisfying

$$\int_0^t \|\gamma(u)\|^2 du < \infty \ , for\ t \in [0,T],$$

and let

$$X(t) = exp\left(-\frac{1}{2}\int_0^t \|\gamma(u)\|^2 du + \int_0^t \gamma(u) dz(u)\right). \tag{A.2}$$

If $E_P[X(T)] = 1$, then $\{X(t), t \epsilon [0,T]\}$ is a martingale and the measure Q on $(\Omega, \mathcal{F}_T^z)$ defined by

$$\frac{dQ}{dP} = X(T)$$

is equivalent to P. Under Q, the process

$$z^Q(t) \triangleq z(t) - \int_0^t \gamma(u)du, \quad t \in [0,T]$$

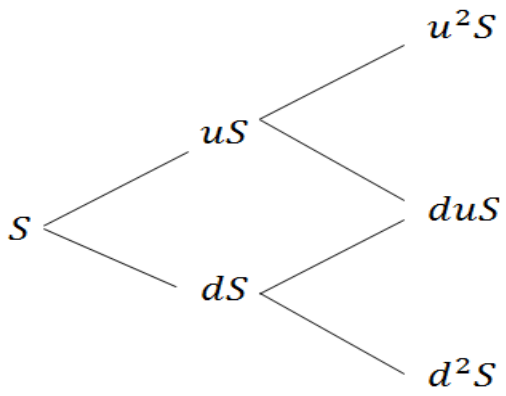
is a standard Brownian motion with respect to $\{\mathcal{F}_t^z\}$. (ii) Conversely if Q is a probability measure on $(\Omega, \mathcal{F}_T^z)$ equivalent to the restriction of P to $\mathcal{F}_T^z$, then $(dQ/dP)_t$ admits the representation in (A.2) for some γ and z^Q is a standard Brownian motion under Q.

Thus the change of measure associated with a change of "drift" in a Brownian motion over a finite horizon is an absolutely continuous change of measure, and every absolutely continuous change of measure for Brownian motion is of this type.

APPENDIX B: Call Option with Two Periods Remaining Before Expiration

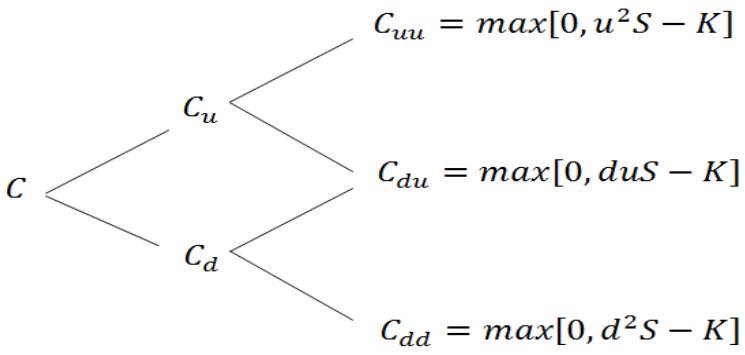
Figure A1 gives a graphical view of the three possible values that the underlying stock may have after the two periods. In a similar way Figure A2 gives the three possible payoffs for the call option (C_{dd} stands for the value of the call two periods from now if the stock moved downward in each period. The other payoffs follow the same logic).

Figure A1: Underlying Price Dynamics in a Two Periods Binomial Tree



Source: Personal processing.

Figure A2: Option Payoffs in a Two Periods Binomial Tree



Source: Personal processing.

One period from now there will be only one period left before expiration therefore we will face a problem identical to the one already solved in Section 1.3.2**.** From the previous analysis we know that the payoff values one period before expiration are equal to:

$$C_u = [pC_{uu} + (1-p)C_{ud}]/r$$

$$C_d = [pC_{du} + (1-p)C_{dd}]/r\ .$$

Now it is possible to form an hedging portfolio with ΔS in stocks and D in bonds whose value will replicate the option payoffs at the end of the first period. The functional form of Δ and D remains unchanged. Therefore by plugging in (1.16) the new values of C_d and C_u it is possible to obtain:

$$\Delta = \frac{C_u - C_d}{(u-d)S} = \frac{p(C_{uu} - C_{du}) + (1-p)(C_{ud} - C_{dd})}{r(u-d)S}$$

$$D = \frac{uC_d - dC_u}{(u-d)r} = \frac{p(uC_{du} - dC_{uu}) + (1-p)(uC_{dd} - dC_{ud})}{r^2(u-d)}\ .$$

Again in order to avoid arbitrage opportunities the price of the call should be equal to the price of the hedging portfolio. So

$$C = \Delta S \quad + D = \frac{C_{uu}(pr - pd)}{r^2(u-d)} + \frac{C_{ud}(p(u-r) + (1-p)(r-d)}{r^2(u-d)} + \frac{C_{dd}((1-p)(u-r))}{r^2(u-d)}$$

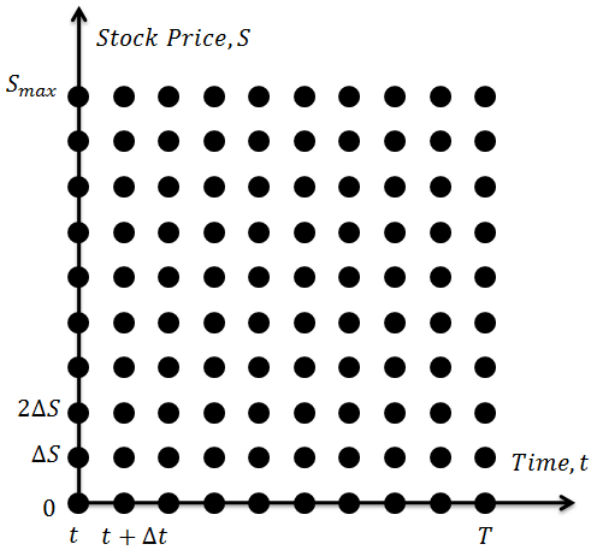
$$= \frac{[p^2C_{uu} + 2p(1-p)C_{ud} + (1-p)^2C_{dd}]}{r^2}$$

$$= \frac{\left[p^2 max[0, u^2S - K] + 2p(1-p)max[0, duS - K] + (1-p)^2 max[0, d^2S - K]\right]}{r^2}.$$

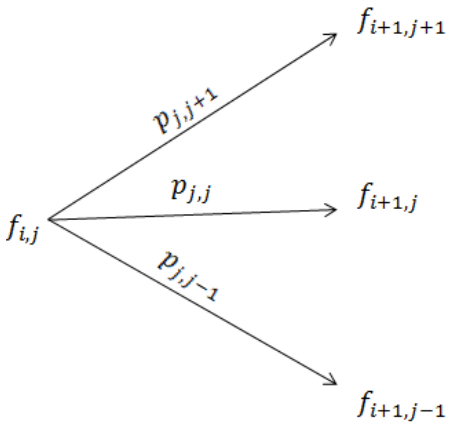
The last expression gives the exact value of the call since it is always greater than $S - K$ as r is greater than one and the stock pays no dividend (Cox et al. 1979). The stated procedure may be used recursively to price option with arbitrary *n* time periods before expiration.

APPENDIX C: Finite Difference Figures

Figure A3: Explicit Finite Difference



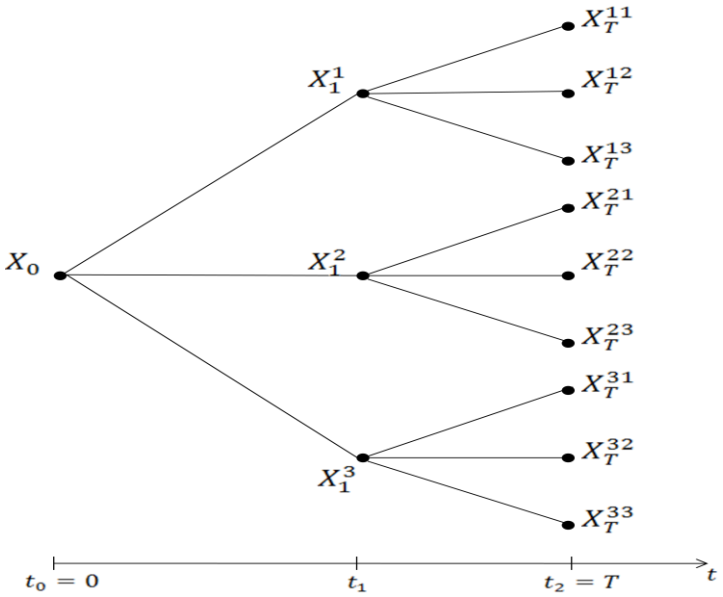
Source: Personal processing.



Source: Personal processing.

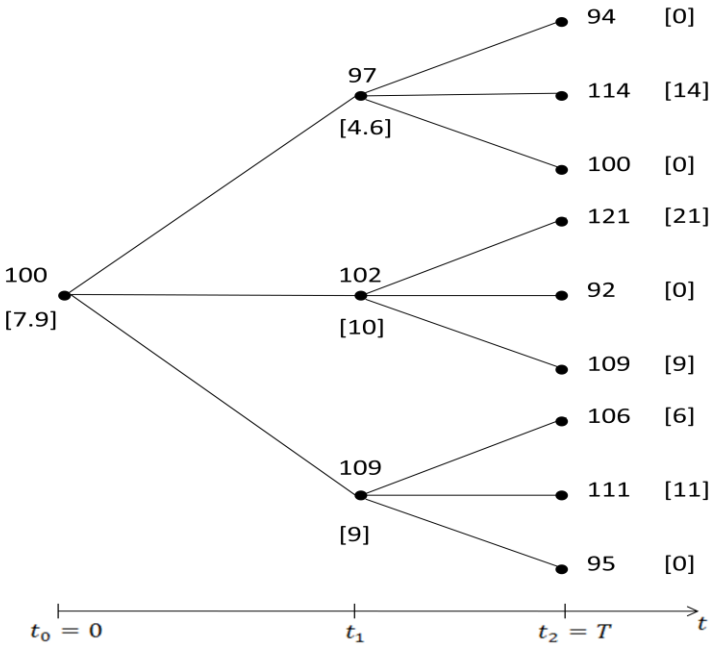
APPENDIX D: Stochastic Tree Figures

Figure A4: Simulated Tree for m=2 b=3



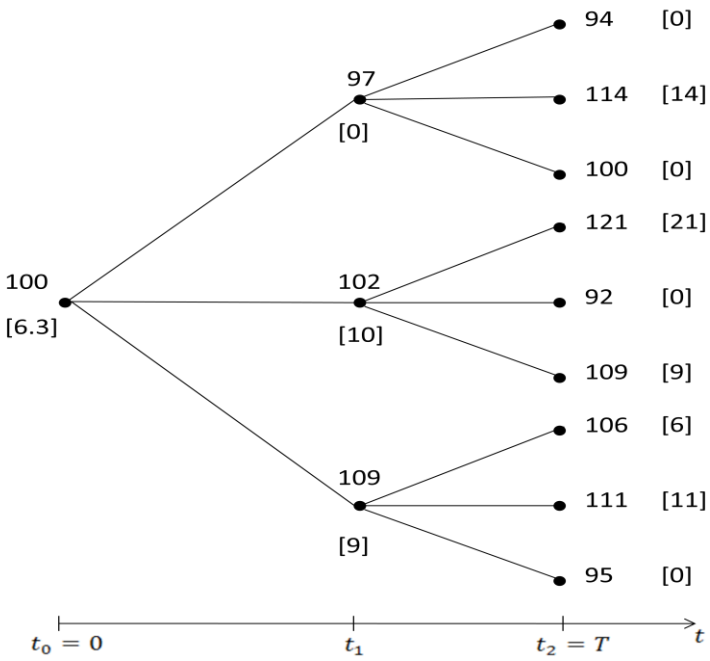
Source: Personal processing.

Figure A5: Example of High Estimator Calculation for a Call Option on a Single Underlying Asset and a Strike Price of 100



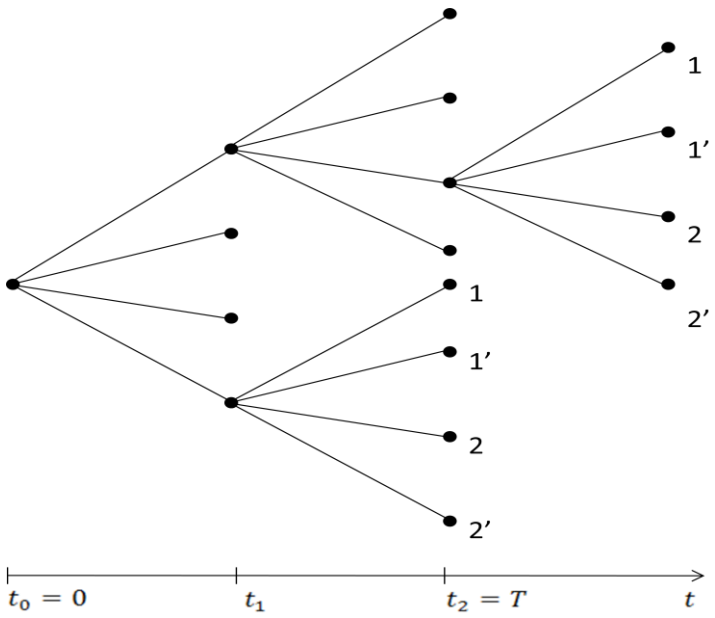
Labels at each node show the levels of the underlying asset and [in brackets] the values of the high estimator.
Source: Personal processing.

Figure A6: Example of Low Estimator Calculation for a Call Option on a Single Underlying Asset and a Strike Price of 100



Labels at each node show the levels of the underlying asset and [in brackets] the values of the low estimator.
Source: Personal Processing.

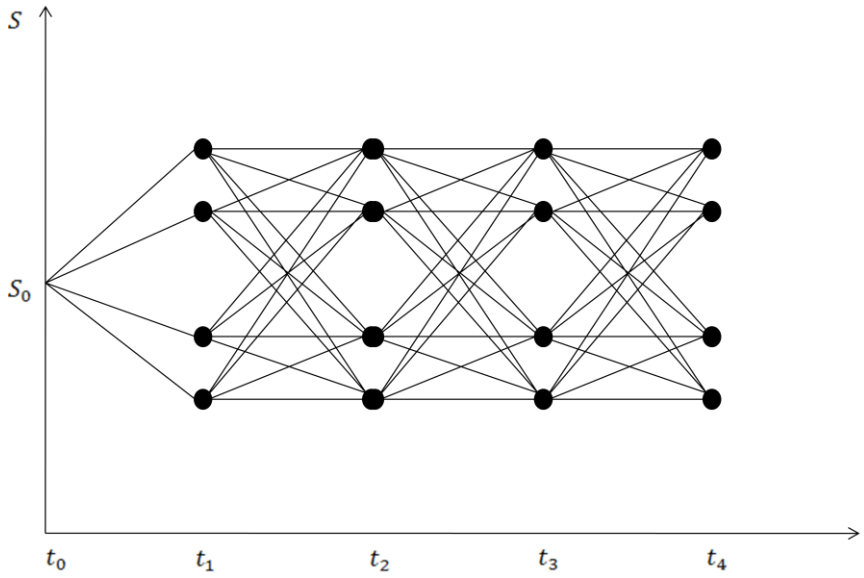
Figure A7: Stochastic Tree with Antithetic Branching



b=4 and 1’ and 2’ are generated using the antithetic random variables of 1 and 2 respectively.
Source: Personal Processing.

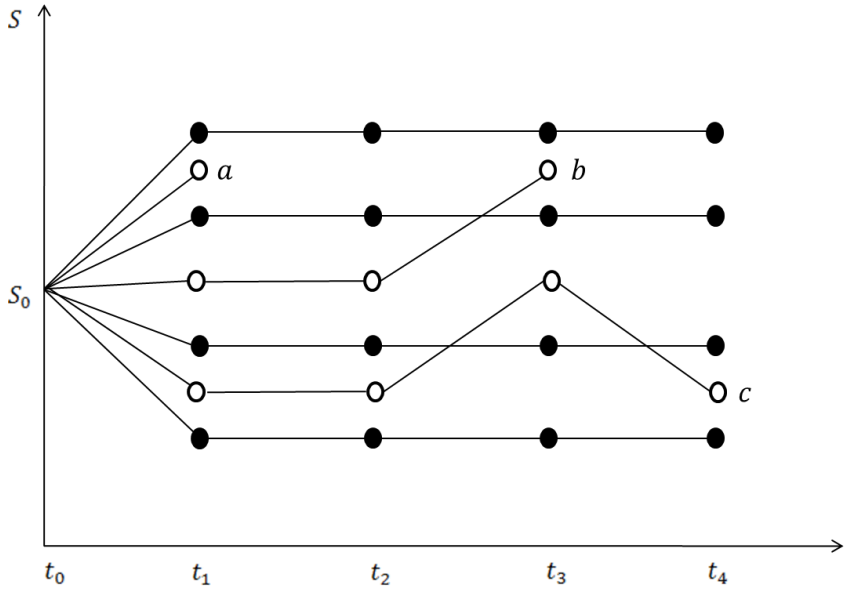
APPENDIX E: Stochastic Mesh Figures

Figure A8: Mesh Illustrated for n=1 T=4 and b=4



Source: Personal processing.

Figure A9: Schematic Diagram of Three Randomly Generated Paths a, b and c for the Path Estimator



The paths terminate upon entry into the exercise region estimated by the mesh.
Source: Personal processing.

APPENDIX F: LSM Algorithm

Valuation of the Bermudan Call Option

The following steps describe the procedure for implementing the LSM algorithm in the pricing of a Bermudan option with three exercises opportunities t_0, t_1 and $T = t_2$. K is the option strike price, r is the risk free rate and $X_{t,i}$ is the value of the underlying asset at time t in state i.

Step A.1 Sample b independent paths of the underlying asset S at time t_0, t_1 and t_2.

Step B.1 For each path set $C(i,s;t_2,T) = max\big(X_{t_2,i} - K, 0\big)\ i = 1, \dots, b$

Step C.1 For the paths where $\big(X_{t_1,i} - K\big) > 0$, project $C(i,s;t_2,T) * exp\big(-r * (t_2 - t_1)\big)$ onto the basis functions $L_j\big(X_{t_1,i}\big)\ j = 1, \dots, M$ using the method of Least Squares.

Step D.1 For the paths where $\big(X_{t_1,i} - K\big) > 0$ and $\big(X_{t_1,i} - K\big) > \hat{F}_M(i;t_1)$ set $C(i,s;t_1,T) = \big(X_{t_1,i} - K\big)$, where $\hat{F}_M(i;t_1)$ is the approximation of the continuation value obtained from the regression using the N in the money paths. For the remaining paths set $C(i,s;t_1,T) = C(i,s;t_2,T) * exp\big(-r * (t_2 - t_1)\big)$.

Step E.1 For each path set $F(i,t_0) = C(i,s;t_1,T) * exp\big(-r * (t_1 - t_0)\big)$, compute $F(t_0) = \frac{1}{b}\sum_{i=1}^{b} F(i,t_0)$.

Step F.1 Finally exercise if $\big(X_{t_0} - K\big) > F(t_0)$. The brute force LSM estimate of the Bermudan option is hence given by $V_0 = max\Big(X_{t_0} - K, F(t_0)\Big)$.

Application of Control Variate

The following steps need to be integrated with the preceding algorithm.

Step B.2 For each path set $Y_{t_2}^i = max\big(X_{t_2,i} - K, 0\big)\ i = 1, \dots, b$

Step D.2 For the paths where $\big(X_{t_1,i} - K\big) > 0$ and $\big(X_{t_1,i} - K\big) > \hat{F}_M(i;t_1)$ set $Y_{t_1}^i = BS\big(X_{t_1,i}, t_1\big)$ where $BS\big(X_{t_1,i}, t_1\big)$ is the Black and Scholes price of a European call option expiring at time T with the same parameters as the Bermudan option. For the remaining paths set $Y_{t_1}^i = Y_{t_2}^i * exp\big(-r * (t_2 - t_1)\big)$.

Step E.2 For each path set $Y_{t_0}^i = Y_{t_1}^i * exp\big(-r * (t_1 - t_0)\big)$ and compute $Y_{t_0}^{(b)} = \frac{1}{b}\sum_{i=1}^{b} Y_{t_0}^i$

$(FY)_{t_0}^{(b)} = \frac{1}{b}\sum_{i=1}^{b} F(i,t_0)Y_{t_0}^{i}$ and $(Y^2)_{t_0}^{(b)} = \frac{1}{b}\sum_{i=1}^{b}\left(Y_{t_0}^{i}\right)^2$.

Step E.3 To adjust the Monte Carlo estimate with the control variate, compute $\theta_{t_0}^{(b)} = -\frac{(FY)_{t_0}^{(b)} - F(t_0)Y_{t_0}^{(b)}}{(Y^2)_{t_0}^{(b)} - \left(Y_{t_0}^{(b)}\right)^2}$ and set $F(t_0)^{CV} = F(t_0) + \theta_{t_0}^{(b)}\left(Y_{t_0}^{(b)} - BS\left(X_{t_0}, t_0\right)\right)$.

Step F.2 Finally exercise if $\left(X_{t_0} - K\right) > F(t_0)^{CV}$. The control variate adjusted LSM estimate of the Bermudan option is given by $V_0 = max\left(X_{t_0} - K, F(t_0)^{CV}\right)$.

Control Variate Improved LSM (CV-LSM)

Step C.0 For the paths where $\left(X_{t_1,i} - K\right) > 0$, project $Y_{t_2}^{i} * exp\left(-r * (t_2 - t_1)\right)$, $\left(Y_{t_2}^{i}\right)^2 * exp\left(-r * (t_2 - t_1)\right)$ and $Y_{t_2}^{i} * C(i,s;t_2,T) * exp\left(-r * (t_2 - t_1)\right)$ onto the same set of basis functions used for the LSM.

Step D.0 For the paths where $\left(X_{t_1,i} - K\right) > 0$ compute $F_M^{CV}(i;t_1) = \hat{F}_M(i;t_1) + \theta_{t_1}^{M(N)}\left(Y_{t_1}^{M(N)} - BS\left(X_{t_1,i}, t_1\right)\right)$ where $\theta_{t_1}^{M(N)} \equiv -\frac{(WY)_{t_1}^{M(N)} - \hat{F}_M(i;t_1)Y_{t_1}^{M(N)}}{(Y^2)_{t_1}^{M(N)} - \left(Y_{t_1}^{M(N)}\right)^2}$ and $(WY)_{t_1}^{M(N)}$, $\hat{F}_M(i;t_1)Y_{t_1}^{M(N)}$, $(Y^2)_{t_1}^{M(N)}$ and $Y_{t_1}^{M(N)}$ are the approximations of the conditional expectations from the regression.

Step D.1* For the paths where $\left(X_{t_1,i} - K\right) > F_M^{CV}(i;t_1)$ set $C(i,s;t_1,T) = \left(X_{t_1,i} - K\right)$ and for the remaining paths set $C(i,s;t_1,T) = C(i,s;t_2,T) * exp\left(-r * (t_2 - t_1)\right)$.

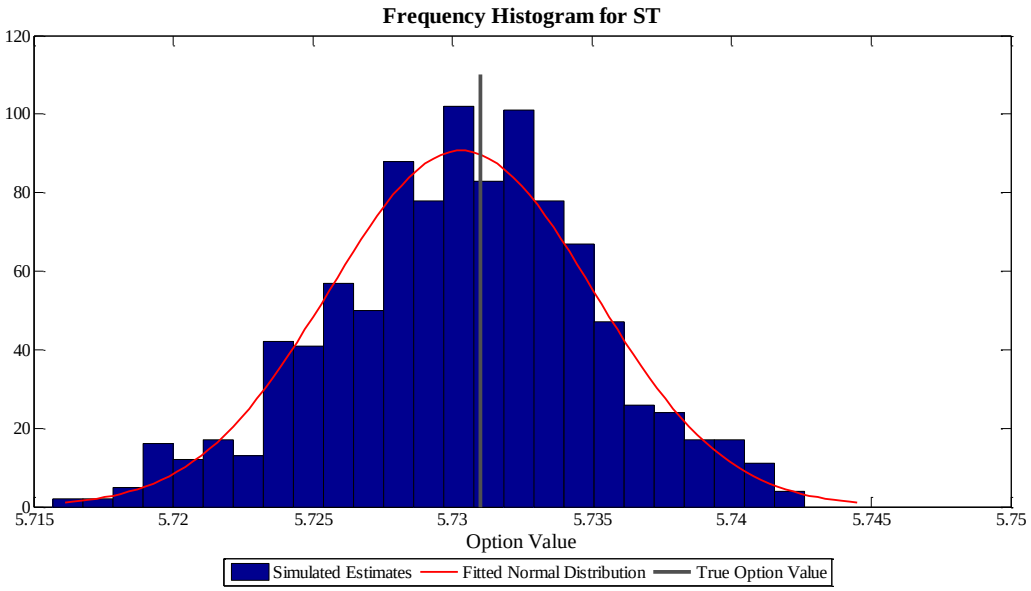
Step D.2* For the paths where $\left(X_{t_1,i} - K\right) > 0$ and $\left(X_{t_1,i} - K\right) > F_M^{CV}(i;t_1)$ set $Y_{t_1}^{i} = BS\left(X_{t_1,i}, t_1\right)$. For the remaining paths set $Y_{t_1}^{i} = Y_{t_2}^{i} * exp\left(-r * (t_2 - t_1)\right)$.

APPENDIX G: Algorithm Distributions

In this appendix the distributions of the price estimates for the three considered algorithms are presented. The option under consideration is the same as the one treated in sections 3.1.1, 3.1.2 and 3.1.3. Each figure has been constructed by calculating the option premium 1000 times. The red lines are probability density functions for normal distributions having means and variances equal to the ones of the samples.

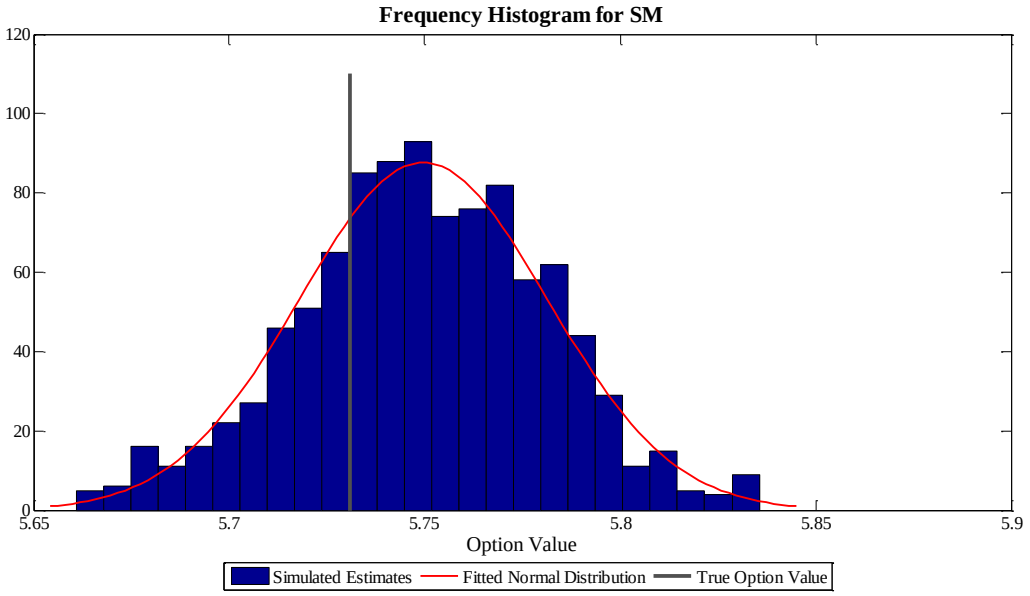
Considering the stochastic tree it is possible to notice that the estimator is fairly centered on the true option value and presents relatively small dispersion with 90% of the estimates lying in the tight interval 5.722 - 5.7382 (1.6 cents). The Lilliefors and Jarque-Bera tests confirm the normal distribution of our sample at 0.1% significance level. The results are a bit different for the stochastic mesh. Its distribution presents a positive bias as evinced by a distribution centered to the right of the true option price. Moreover there is considerable dispersion with 90% of the estimates falling in the wider interval 5.6948 – 5.7985 (10.39 cents). Also for this sample normality is confirmed at 0.1% significance. The CVLSM algorithm while presenting a slightly negative bias, as expected since its approximation method of the exercise strategy may lead to sub-optimal decisions, seems the algorithm with the lowest dispersion around the mean with 90% of the estimates falling in the interval 5.7239 – 5.7361 (1.22 cents). Normality is confirmed at 0.1% confidence level.

Figure A10: Stochastic Tree Estimator Distribution



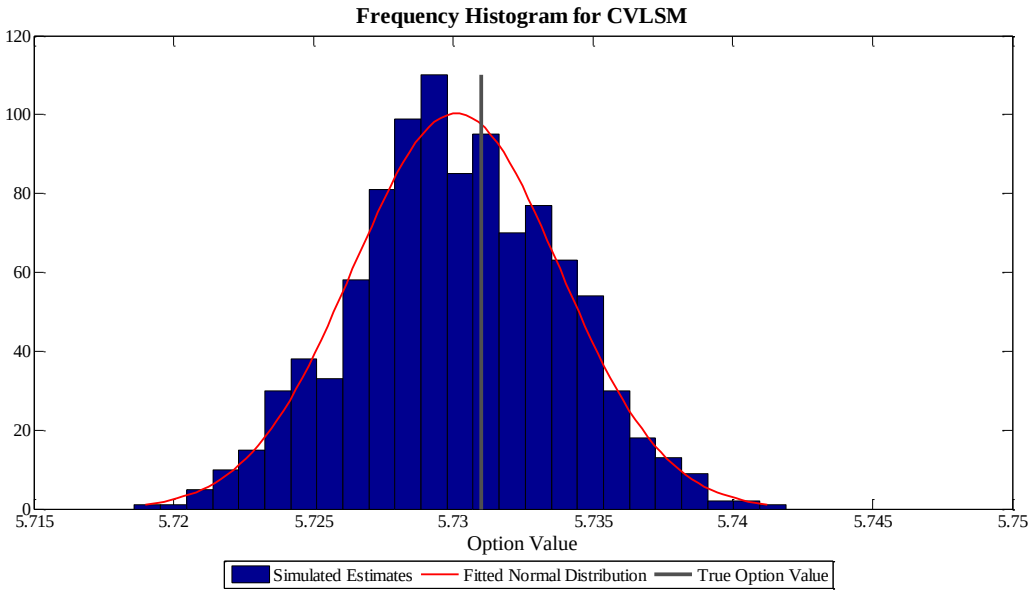
Option parameters: S=100, $K = 100$, $n = 100$, $b = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3,$ and T.
The algorithm was implemented with pruning antithetic branching and control variate. The histogram was produced with 1000 estimates.

Figure A11: Stochastic Mesh Estimator Distribution



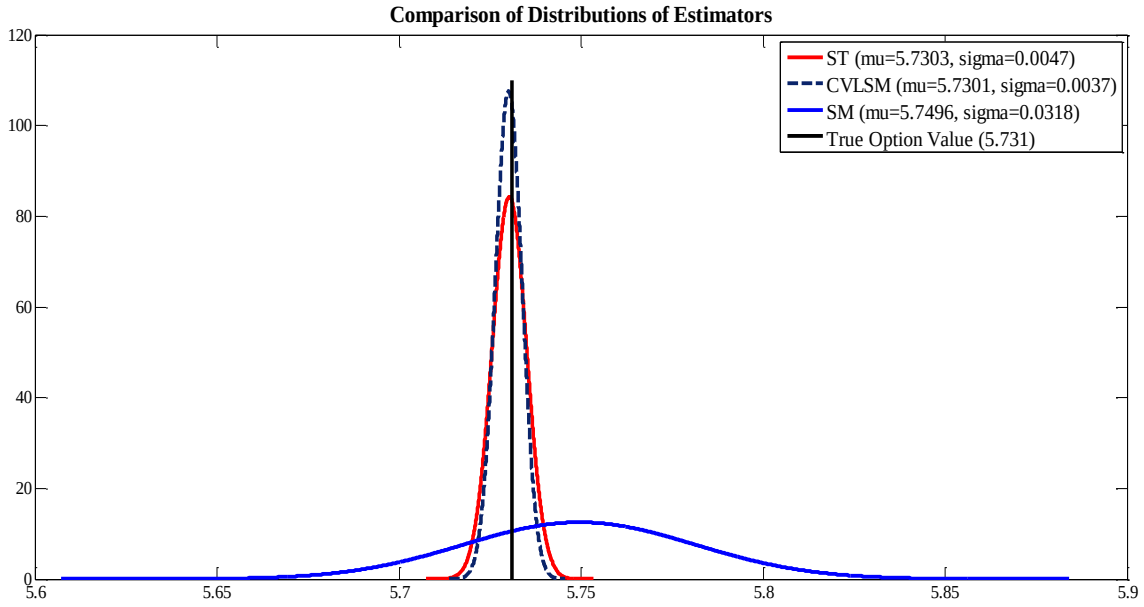
Option parameters: S=100, $K = 100$, $n = 100$, $b = 100$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T.
The algorithm was implemented with pruning antithetic variates and control variate. The histogram was produced with 1000 estimates.

Figure A12: CVLSM Estimator Distribution



Option parameters: S=100, $K = 100$, $b = 10000$, $r = 0.05$, $\delta = 0.10$, $T = 1$, $\sigma = 0.2$ and four exercise opportunities at times $0, T/3, 2T/3$, and T.
The algorithm was implemented with pruning antithetic variates and control variate. The histogram was produced with 1000 estimates.

Figure A13: Distributions Comparison



The figure plots on one graph the distributions of the three algorithms previously presented.

Figure A13 compares the three distributions. It is possible to notice how the ST and CVLS present similar distributions having small variance and extremely low bias while the mesh algorithm has a positive bias (even though still lower than 1%) and quite a big variance (approximately 10 times the one of the other algorithms).

APPENDIX H: Matlab Code

Single Asset Option Price Estimation with the Stochastic Tree Algorithm

```
clear all
K=100; % Strike Price
delta=0.1; % Dividend Yield
T=1; % Time to maturity
sigma=0.2; % Volatility
t=[0;1/3;2/3;1]; % Exercise opportunities for the Bermudan option
r=0.05; % Risk-free rate
n=100; % Monte Carlo reiterations
b =50; % Branching parameter
S=[70;80;90;100;110;120]; % Initial stock price
truevalue=[0.121;0.670;2.303;5.731;11.341;20.000];
o=length(S);
g=25; % Number of estimates

% variables definition
estimator1= zeros(1,o);
highestimator= zeros(1,o);
lowestimator= zeros(1,o);
serrh=zeros(1,o);
serrl=zeros(1,o);
serr1=zeros(1,o);
RMSE=zeros(1,o);
conflow1= zeros(1,o);
confhigh1= zeros(1,o);
realerror1=zeros(1,o);
time=zeros(1,o);
he=zeros(g,o);
le=zeros(g,o);
pointestimate= zeros(g,o);

for q=1:o
tic
for h=1:g
lowest1=zeros(n,1);
lowest=zeros(n,1);
highest1=zeros(n,1);
highest=zeros(n,1);
for f=1:n
%generate the tree
S1=zeros(b,1);
S3=zeros(b,1);
S2=zeros(b*b,1);
S4=zeros(b*b,1);

for i=1:b/2
    x=normal(1,0,1);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*(-x));
    % generate the tree on which the European option control variate
    % is valued
    S3(i)=S(q)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S3(i+b/2)=S(q)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*(-x));
```

```
end
for i=1:b
for k=1:b/2
    x=normal(1,0,1);
    S2(b*(i-1)+k)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x);
    S2(b*(i-1)+k+b/2)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x));
    % generate the tree on which the European option control variate
    % is valued
    S4(b*(i-1)+k)=S3(i)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S4(b*(i-1)+k+b/2)=S3(i)*exp((r-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x));
end
end

% Calculate the high estimator
% Calculate the value of a European call starting at T-1 and maturing at T
finpay = blsprice(S2, K, r, 1/3 ,sigma , delta);
est=zeros(1,b*b);
for i=1:(b*b)
est(i)=max(S2(i)-K,finpay(i));
end

ini2=1:b:length(S2);
est2=zeros(b,1);
for i=1:(b)
cont2=(sum(est(ini2(i):ini2(i)+b-1))*exp(-r*(1-2/3)))/b;
est2(i)=max(S1(i)-K,cont2);
end
% true BS price used in the control variate
callcontrol=blsprice(S(q), K, r, 2/3 ,sigma );
%Control variate estimate
optionvalue4=zeros(1,b*b);
for i= 1:b*b
optionvalue4(i)=max(S4(i)-K,0);
end
expectedvalue4=mean(optionvalue4);
price4=(expectedvalue4*exp(-r*(1-1/3)));

cont3=(sum(est2)*exp(-r*(1-2/3)))/b;
highest1(f)=max(S(q)-K,cont3);
% Control variate high estimator
highest(f)=highest1(f)+(callcontrol-price4);

% Calculate the low estimator
% For time T-1 the low estimator and the high estimator are equal therefore
% I started the calculation of the low estimator for T-2
est4=zeros(b,1);

for i=1:b
 nj2=zeros(1,b/2);
for j=1:b/2
  if j==1
        cont5= ((sum(est(ini2(i)+j:(i*b)-b/2))*exp(-r*(1-2/3)))...
            +(sum(est((i*b)-(b/2-2):(i*b)))*exp(-r*(1-2/3))))/(b-2);
    else
        if j==b/2
```

```
    cont5=(sum(est(ini2(i):b*(i-1)+(b/2-1)))*exp(-r*(1-2/3))...
        +(sum(est(b*(i-1)+(b/2+1):(i*b)-1)))*exp(-r*(1-2/3)))/(b-2);
    else
    cont5=((sum(est(ini2(i):b*(i-1)+j-1))...
        +sum(est(b*(i-1)+j+1:b*(i-1)+j+(b/2-1)))...
        +sum(est(b*(i-1)+j+(b/2+1):i*b)))*exp(-r*(1-2/3)))/(b-2);
        end
   end
if max(S1(i)-K,0)>= cont5
    nj2(j)= max(S1(i)-K,0);
else
    nj2(j) = (est(b*(i-1)+j)*exp(-r*(1-2/3))...
        +est(b*(i-1)+j+b/2)*exp(-r*(1-2/3)))/2;

end
end
est4(i,1) = sum(nj2(1:b/2))*2/b;
end
 nj3=zeros(1,b/2);
for j=1:b/2
if j==1
        cont6= ((sum(est4(1+j:b-b/2))*exp(-r*(1-2/3))...
            +(sum(est4(b-(b/2-2):b)))*exp(-r*(1-2/3))))/(b-2);
    else
        if j==b/2
    cont6=(sum(est4(1:b/2-1))*exp(-r*(1-2/3))...
        +(sum(est4(b/2+1:b-1)))*exp(-r*(1-2/3)))/(b-2);
    else
    cont6=((sum(est4(1:j-1))...
        +sum(est4(j+1:j+(b/2-1)))...
        +sum(est4(j+(b/2+1):b)))*exp(-r*(1-2/3)))/(b-2);
        end
end
if max(S(q)-K,0)>= cont6
    nj3(j)= max(S(q)-K,0);
else
    nj3(j) = (est4(j)*exp(-r*(1-2/3))+est4(j+b/2)*exp(-r*(1-2/3)))/2;

end
end
lowest1(f)=sum(nj3(1:b/2))*2/b;
% Control variate low estimator
lowest(f)=lowest1(f)+(callcontrol-price4);

end
%expected value high estimator
he(h,q)=mean(highest);
%expected value low estimator
le(h,q)=mean(lowest);
%average estimator
pointestimate(h,q)= 0.5*max(S(q)-K,le(h,q))+0.5*he(h,q);

end
% Final Low Estimator
lowestimator(1,q)=mean(le(:,q));
% Final High Estimator
highestimator(1,q)=mean(he(:,q));
% Final Point Estimator
```

```
estimator1(1,q)= mean(pointestimate(:,q));
%SE Point Estimator
serr1(1,q)=std(pointestimate(:,q))/sqrt(g);
%SE Low Estimator
serrl(1,q)=std(le(:,q))/sqrt(g);
%SE High Estimator
serrh(1,q)=std(he(:,q))/sqrt(g);
%RMSE
RMSE(1,q)=sqrt(mean((pointestimate(:,q)-truevalue(q)).^2));
% 95% quantile
x = norminv(0.95,0,1);
% confidence interval
conflow1(1,q) =lowestimator(1,q) - x*serrl(1,q);
confhigh1(1,q)= highestimator(1,q)+ x*serrh(1,q);
realerror1(1,q)=abs(estimator1(1,q)-truevalue(q))/truevalue(q);
% cpu time
time(q)=toc;

end
%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Monte Carlo Reiteration';
str2 = [name,'   ', num2str(n)];
disp(str2);
name = 'Number of Estimates';
str = [name,'   ', num2str(g)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price    High Estimator      SE      Low Estimator',...
    '      SE         Point Estimator      SE',...
    '       Confidence Interval     True Value   Rel Error',...
    '      RMSE       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g        %-8.3g      %-8.4g      %-8.3g',...
    '        %-8.4g        %-8.3g    %-8.4g  %-8.4g       %-8.4g',...
    '    %-8.3g       %-8.3g             %-8.4g\n'],...
label(i,:),highestimator(i),serrh(i),lowestimator(i),serrl(i),...
estimator1(i),serr1(i),conflow1(i),confhigh1(i),truevalue(i),...
realerror1(i),RMSE(i),time(i))
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

Single Asset Option Price Estimation with the Stochastic Mesh Algorithm

```
clear all
K=100; % Strike Price
delta=0.1; % Dividend Yield
T=1; % Time to maturity
sigma=0.2; % Volatility
t=[0;1/3;2/3;1]; % Exercise opportunities for the Bermudan option
r=0.05; % Risk-free rate
n=100; % Monte Carlo reiteration
b =50; % Branching parameter
S=[70;80;90;100;110;120]; % Initial stock price
truevalue=[0.121;0.670;2.303;5.731;11.341;20.000];
o=length(S);
g=25; % Number of estimates

%variable generation
lowestimator1=zeros(g,o);
pointestimateouter=zeros(g,o);
highest=zeros(1,o);
lowest=zeros(1,o);
serrl=zeros(1,o);
serrh=zeros(1,o);
RMSE=zeros(1,o);
estimator1=zeros(1,o);
serr1=zeros(1,o);
estimatorouter=zeros(g,o);
realerror=zeros(1,o);
realerrorouter=zeros(1,o);
conflow=zeros(1,o);
confhigh=zeros(1,o);
time=zeros(1,o);

for q=1:o
tic

for h=1:g
Q0=zeros(1,n);
L0=zeros(1,n);
Call=zeros(1,n);
for f=1:n
%Generate the mesh
S1=zeros(b,1);
S2=zeros(b,1);
S3=zeros(b,1);
for i=1:b/2
    x=normal(1,0,1);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x));
    x=normal(1,0,1);
    S2(i)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S2(i+b/2)=S1(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x));
end

%Calculate the mesh
Q2=max((S2-K)*exp(-r*(1-1/3)),blsprice(S2, K, r, 1/3 ,sigma , delta)...
    *exp(-r*(1-1/3)));
```

```
Call2=blsprice(S2, K, r, 1/3 ,sigma , delta)*exp(-r*(1-1/3));


%Calculate the weight at time T-2
%first the transition densities
density2 =zeros(b,b);
for k=1:b
for i=1:b
   density2(i,k)=1/(sigma*sqrt(1/3)*S2(k))*normpdf((log(S2(k)/S1(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
end
end

%Calculate the weights
weights2=zeros(b,b);
for k=1:b
for i=1:b
    weights2(i,k)=(density2(k,i))/(sum(density2(1:b,i))/b);
end
end

% Now the estimator at time T-2
Q1=zeros(b,1);
est2=zeros(b,1);
continuation2=zeros(1,b);
Call1=zeros(b,1);
for i=1:b

continuation2(i)= ((sum(Q2.*weights2(1:b,i))/b));
%Mesh estimator
Q1(i)=max(max(S1(i)-K,0)*exp(-r*(1-2/3)),continuation2(i));

%Control variate part European call estimate at time T-2
Call1(i)=(sum((Call2).*weights2(1:b,i))/b);
end

% Now the estimator at time 0
continuation3= ((sum(Q1)/b));
Q0(f)=max(S(q)-K,continuation3);

%Control variate part
Call(f)=(sum(Call1)/b);
% True BS price used in the control variate
truecall=blsprice(S(q), K, r, 1, sigma, delta);


%Calculate the low estimator
lowestimator=max((S2-K)*exp(-r*(1-1/3)),...
    blsprice(S2, K, r, 1/3 ,sigma , delta)*exp(-r*(1-1/3)));
%variable generation
cont=zeros(b,b/2);
exercise=zeros(b,b/2);
weightcont=zeros(2,b);
weights3=zeros(b-2,b);
weightcont2=zeros(2,b);
weights4=zeros(b-2,b);
weightcont3=zeros(2,b);
```

```
%Calculate the low estimator at Time T-2
  for j=1:b/2
  if j==1
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      for k=1:b
      weightcont(1,k)=(density2(k,1))/(sum(density2(1:b,1))/(b));
      weightcont(2,k)=(density2(k,b/2+1))/(sum(density2(1:b,b/2+1))/(b));
      % These weights are used to check if to continue or not
      for i=1:(b-2)/2
      weights3(i,k)=(density2(k,i+1))/(sum(density2(1:b,i+1))/(b));
      weights3(i+(b/2)-1,k)=(density2(k,i+1+b/2))/...
          (sum(density2(1:b,i+1+b/2))/(b));
      end

      cont(k,j)= (sum(lowestimator(2:b/2).*weights3(1:(b-2)/2,k))...
          +sum(lowestimator((b/2)+2:b).*weights3(((b-2)/2)+1:b-2,k)))...
          /(b-2);
      exercise(k,j)=((lowestimator(1)*weightcont(1,k))...
          +(lowestimator(1+b/2)*weightcont(2,k)))/2;
      end
  else
      if j==b/2
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      for k=1:b
      weightcont2(1,k)=(density2(k,b/2))/(sum(density2(1:b,b/2))/(b));
      weightcont2(2,k)=(density2(k,b))/(sum(density2(1:b,b))/(b));
      % These weights are used to check if to continue or not
      for i=1:(b-2)/2
      weights4(i,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      weights4(i+(b/2)-1,k)=(density2(k,i+b/2))/...
          (sum(density2(1:b,i+b/2))/(b));
      end
      cont(k,j)= (sum(lowestimator(1:(b/2)-1).*weights4(1:(b-2)/2,k))...
          +sum(lowestimator((b/2)+1:b-1).*weights4(((b-2)/2)+1:b-2,k)))/...
          (b-2);
      exercise(k,j)=((lowestimator(b/2)*weightcont2(1,k))...
          +(lowestimator(b)*weightcont2(2,k)))/2;
      end
      else
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      for k=1:b
      weightcont3(1,k)=(density2(k,j))/(sum(density2(1:b,j))/(b));
      weightcont3(2,k)=(density2(k,j+b/2))/(sum(density2(1:b,j+b/2))/(b));
      % These weights are used to check if to continue or not
      for i=1:j-1
      weights5(i,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      for i=j+1:j+b/2-1
      weights6(i-j,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      for i=j+b/2+1:b
      weights7(i-(j+b/2),k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      cont(k,j)= (sum(lowestimator(1:j-1).*weights5(1:j-1,k))...
```

```
          +sum(lowestimator(j+1:j+b/2-1).*weights6(1:b/2-1,k))...
          +sum(lowestimator(j+b/2+1:b).*weights7(1:b/2-j,k)))/(b-2);
      exercise(k,j)=(lowestimator(j)*weightcont3(1,k)...
          +lowestimator(j+b/2)*weightcont3(2,k))/2;
      end
      end
  end
  end
  %Calculate the estimator
  % Check whether to exercise of not
  lowestimator2=zeros(1,b);
  est=zeros(b,b/2);
  for i=1:b
      for k=1:b/2
      if (max(S1(i)-K,0)*exp(-r*(1-2/3)))>= cont(i,k);
          est(i,k)=max(S1(i)-K,0)*exp(-r*(1-2/3));
      else
          est(i,k)=exercise(i,k);
      end
      end
  lowestimator2(i)=sum(est(i,1:b/2))*2/b;

  end
%  low estimator at time zero
L0(f)=max(S(q)-K,(sum(lowestimator2)/b));

end

%Estimating beta
%generating the matrix
F=zeros(n,1);
G=zeros(n,2);
H=zeros(n,1)+1;
for k= 1:n
    F(k)= (Q0(k));
    G(k,1:2)= [H(k),Call(k)];
end
%These are the regression coefficients
beta=regress(F,G);


%high estimator corrected with outer control
estimatorouter(h,q)=mean(Q0)-beta(2)*(mean(Call)-truecall);

%Here few lines to calculate the beta and the corrected low estimator
%generating the matrix
F1=zeros(n,1);
G1=zeros(n,2);
H1=zeros(n,1)+1;
for k= 1:n
    F1(k)= (L0(k));
    G1(k,1:2)= [H1(k),Call(k)];
end
%These are the regression coefficients
beta=regress(F1,G1);
%This is the corrected low estimator
lowestimator1(h,q)=mean(L0)-beta(2)*(mean(Call)-truecall);
%Point Estimator
```

```
pointestimateouter(h,q)=0.5*estimatorouter(h,q)+0.5*lowestimator1(h,q);

end
% Final Point Estimator
estimator1(1,q)=mean(pointestimateouter(:,q));
% Final High Estimator
highest(1,q)=mean(estimatorouter(:,q));
% Final Low Estimator
lowest(1,q)=mean(lowestimator1(:,q));
%SE High Estimator
serrh(1,q)=std(estimatorouter(:,q))/sqrt(g);
%SE Low Estimator
serrl(1,q)=std(lowestimator1(:,q))/sqrt(g);
% SE Point Estimate
serr1(1,q)=std(pointestimateouter(:,q))/sqrt(g);
%RMSE
RMSE(1,q)=sqrt(mean((pointestimateouter(:,q)-truevalue(q)).^2));
realerror(1,q)= abs(estimator1(1,q)-truevalue(q))/truevalue(q);

% 95% quantile
x = norminv(0.95,0,1);

% confidence interval
conflow(1,q)=lowest(1,q)- x*serrl(1,q);
confhigh(1,q)= highest(1,q)+ x*serrh(1,q);
% cpu time
time(q)=toc;
end
%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Monte Carlo Reiteration';
str2 = [name,'   ', num2str(n)];
disp(str2);
name = 'Number of Estimates';
str = [name,'   ', num2str(g)];
disp(str);
disp('   ');


% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price    High Estimator      SE      Low Estimator',...
    '      SE         Point Estimator      SE',...
    '       Confidence Interval     True Value   Rel Error',...
    '        RMSE       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g        %-8.3g      %-8.4g      %-8.3g',...
    '        %-8.4g        %-8.3g      %-8.4g  %-8.4g       %-8.3g',...
    '    %-8.3g       %-8.3g             %-8.4g\n'],...
label(i,:),highest(i),serrh(i),lowest(i),serrl(i),estimator1(i),...
serr1(i),conflow(i),confhigh(i),truevalue(i),realerror(i),RMSE(i),time(i))
```

```
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

Single Asset Option Price Estimation with the CVLSM Algorithm

```
clear all
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1]; % Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
reiteration=25;% Number of estimates
b=10000;% Branching parameter
S0=[70;80;90;100;110;120]; % Initial stock price
K0=[100;100;100;100;100;100];% Strike Price
K=K0./K0;
S=S0./K0;
truevalue=[0.121;0.670;2.303;5.731;11.341;20.000];
o=length(S0);

% variable generation
option=zeros(reiteration,o);
estimator=zeros(1,o);
serr=zeros(1,o);
realerror=zeros(1,o);
RMSE=zeros(1,o);
time=zeros(1,o);
percitm=zeros(1,o);
inthemoneyperc=zeros(1,o);

for j=1:o
tic

for f=1:reiteration
%Generate the stock paths
S1=zeros(b,1);
S2=zeros(b,1);
for i=1:b/2
    x=normal(1,0,1);
    S1(i)=S(j)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S1(i+b/2)=S(j)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x));
    x=normal(1,0,1);
    S2(i)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x);
    S2(i+b/2)=S1(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x));
end

% Calculate the option price
% At T-1 the calculations are straightforward thanks to pruning
Yt2=blsprice(S2,K(j),r,1/3,sigma,delta);
Lt2=max(max(S2-K(j),0),blsprice(S2,K(j),r,1/3,sigma,delta));
```

```
%At time T-2 we need first to see where the option is in the money
indicator=zeros(b,1);
inthemoney=max(S1-K(j),0);
for i=1:b
     if inthemoney(i)>0
        indicator(i)=1;
     end

end

%Generate the basic functions
X2=zeros(b,4);
Y=Lt2*exp(-r*1/3);%F
Y1=Yt2*exp(-r*1/3); %Y
Y12=(Yt2.^2)*exp(-r*1/3);%Y^2
Y13=(Lt2.*Yt2)*exp(-r*1/3);%FY
S22=S1;
for i=1:b
    if indicator(i)==1
        X2(i,1)=K(j);
        X2(i,2)=S22(i);
        X2(i,3)=blsprice(S22(i),K(j),r,2/3,sigma,delta);
        X2(i,4)=S22(i)*blsprice(S22(i),K(j),r,2/3,sigma,delta);
    end
end

%These lines delete the zero rows from the regression matrices
X3 = X2(any(X2,2),:);
Y2= Y(any(X2,2),:);%F
Y21= Y1(any(X2,2),:);%Y
Y22= Y12(any(X2,2),:);%Y^2
Y23= Y13(any(X2,2),:);%FY

beta=inv(X3'*X3)*X3'*Y2; %F
beta1=inv(X3'*X3)*X3'*Y21;%Y
beta2=inv(X3'*X3)*X3'*Y22;%Y^2
beta3=inv(X3'*X3)*X3'*Y23;%FY
Continuation=X3*beta;%F
Continuation1=X3*beta1;%Y
Continuation2=X3*beta2;%Y^2
Continuation3=X3*beta3;%FY
coeff=-(Continuation3-Continuation.*Continuation1)./...
    (Continuation2-(Continuation1.^2));
% control variate adjusted continuation value
Contval=Continuation+coeff.*(Continuation1-X3(:,3));
continuation1=zeros(b,1);

S21=S1;
S21(indicator==0)=[];
for i=1:length(Y2)
    n=find(S1==S21(i));
    continuation1(n)=Contval(i);
end
 Lt1=Lt2*exp(-r*1/3);
 Yt1=Yt2*exp(-r*1/3);
```

```
 for i=1:b
    if max(S1(i)-K(j),0)>0
        if max(S1(i)-K(j),0)>continuation1(i)
            Lt1(i,1)=max(S1(i)-K(j),0);
            Yt1(i,1)=blsprice(S1(i),K(j),r,2/3,sigma,delta);
        end
    end
end

Lt0=Lt1*exp(-r*1/3);
Yt0=Yt1*exp(-r*1/3);

est=mean(Lt0);
Y0=mean(Yt0);
LY=mean(Lt0.*Yt0);
Ysquared=mean(Yt0.^2);
coefficient=-(LY-est*Y0)/(Ysquared-(Y0^2));
% control variate adjusted continuation value
estimator1=est+coefficient*(Y0-blsprice(S(j),K(j),r,1,sigma,delta));
% Point estimate
option(f,j)=max(max(S(j)-K(j),0),estimator1)*100;
% n. of In-the-money paths
inthemoneyperc(f,j)=sum(indicator);
end
% Final Estimator
estimator(1,j)= mean(option(:,j));
% SE Estimator
serr(1,j)=std(option(:,j))/sqrt(reiteration);
realerror(1,j)=abs(estimator(1,j)-truevalue(j))/truevalue(j);
% RMSE
RMSE(1,j)=sqrt(mean((option(:,j)-truevalue(j)).^2));
% In-the-money paths as a percentage of the branching parameter
percitm(j)=mean(inthemoneyperc(:,j))/b;
% cpu time
time(j)=toc;
end

%Building the table
dataLength = length(S0);
%Building the table
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Number of Estimates';
str = [name,'   ', num2str(reiteration)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price   Estimator         SE              True Value',...
    '     Rel Error     RMSE       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g      %-8.3g            %-8.4g',...
```

```
    '      %-8.3g      %-8.3g             %-8.4g\n'],...
label(i,:),estimator(i),serr(i),truevalue(i),realerror(i),RMSE(i),time(i))
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1)
```

2 Assets Max Option Price Estimation with the Stochastic Tree Algorithm

```
clear all
K=100;% Strike Price
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1];% Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
n=100;% Monte Carlo reiterations
b =50;% Branching parameter
S=[70;80;90;100;110;120]; % Asset 1 Initial Stock price
S0=[70;80;90;100;110;120]; % Asset 2 Initial Stock price
truevalue=[0.237;1.259;4.077;9.361;16.924;25.980];
o=length(S);
rho=0.3; % correlation between assets
g=25;% Number of estimates

%Creation of the ratespec for the formula of the European
% rainbow option used for pruning
Settle = 'Jun-1-2008';
Maturity = 'Oct-1-2008';
Rates = r;
Basis = 1;
NumDays = daysdif(Settle, Maturity, Basis);
RateSpec = intenvset('ValuationDate', Settle, 'StartDates', Settle,...
'EndDates', Maturity, 'Rates', Rates, 'Compounding', -1, 'Basis', Basis);
%Creation of the ratespec for formula of the European rainbow option used
%for control variate
Settle1 = 'Jun-1-2008';
Maturity1 = 'Feb-1-2009';
Rates1 = r;
Basis1 = 1;
NumDays1 = daysdif(Settle1, Maturity1, Basis1);
RateSpec1 = intenvset('ValuationDate', Settle1, 'StartDates', Settle1,...
'EndDates', Maturity1, 'Rates', Rates1,'Compounding', -1, 'Basis',...
Basis1);

%variable definition
estimator1= zeros(1,o);
highestimator= zeros(1,o);
lowestimator= zeros(1,o);
serrh1= zeros(1,o);
serrl1= zeros(1,o);
serr1=zeros(1,o);
RMSE=zeros(1,o);
conflow1= zeros(1,o);
confhigh1= zeros(1,o);
```

```
realerror1=zeros(1,o);
time=zeros(1,o);
he=zeros(g,o);
le=zeros(g,o);
pointestimate= zeros(g,o);

for q=1:o
tic
% Control variate constant details
Sigma11 = sigma;
Sigma21 = sigma;
Div11 = 0;
Div21 = 0;
StockSpec11 = stockspec(Sigma11, S(q), 'continuous', Div11);
StockSpec21 = stockspec(Sigma21, S0(q), 'continuous', Div21);
OptSpec = 'call';
% true European price used in the control variate
callcontrol = maxassetbystulz(RateSpec1, StockSpec11, StockSpec21,...
Settle1, Maturity1, OptSpec, K, rho);
%variable definition
lowest1=zeros(n,1);
highest1=zeros(n,1);
highest=zeros(n,g);
lowest=zeros(n,g);

for h=1:g
for f=1:n
%generate the tree for Asset 1
S1=zeros(b,1);
S3=zeros(b,1);
S2=zeros(b*b,1);
S4=zeros(b*b,1);

%generate the tree for Asset 2
S12=zeros(b,1);
S32=zeros(b,1);
S22=zeros(b*b,1);
S42=zeros(b*b,1);

m=[0,0];
vcv = [1 rho; rho 1];

for i=1:b/2
    x=normal(1,m,vcv);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S12(i)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S12(i+b/2)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
    S3(i)=S(q)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S3(i+b/2)=S(q)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S32(i)=S0(q)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S32(i+b/2)=S0(q)*exp((r-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
end
for i=1:b
```

```
for k=1:b/2
    x=normal(1,m,vcv);
    S2(b*(i-1)+k)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(1));
    S2(b*(i-1)+k+b/2)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S22(b*(i-1)+k)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(2));
    S22(b*(i-1)+k+b/2)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S4(b*(i-1)+k)=S3(i)*exp((r-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(1));
    S4(b*(i-1)+k+b/2)=S3(i)*exp((r-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S42(b*(i-1)+k)=S32(i)*exp((r-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(2));
    S42(b*(i-1)+k+b/2)=S32(i)*exp((r-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
end
end

% Calculate the high estimator
% Generation of the inputs for the rainbow option used in pruning
AssetPrice1 = S2;
AssetPrice2 = S22;
Sigma1 = sigma;
Sigma2 = sigma;
Div1 = delta;
Div2 = delta;

StockSpec1 = stockspec(Sigma1, AssetPrice1, 'continuous', Div1);
StockSpec2 = stockspec(Sigma2, AssetPrice2, 'continuous', Div2);
OptSpec = 'call';

est =max(max(S2,S22)-K, maxassetbystulz(RateSpec, StockSpec1,
StockSpec2,...
Settle, Maturity, OptSpec, K, rho));

ini2=1:b:length(S2);
est2=zeros(b,1);
for i=1:(b)
cont2=(sum(est(ini2(i):ini2(i)+b-1))*exp(-r*(1-2/3)))/b;
est2(i)=max(max(S1(i),S12(i))-K,cont2);
end
%Control variate estimate
optionvalue4=max(max(S4,S42)-K,0);
expectedvalue4=mean(optionvalue4);
price4=(expectedvalue4*exp(-r*(1-1/3)));

cont3=(sum(est2)*exp(-r*(1-2/3)))/b;
highest1(f)=max(max(S(q),S0(q))-K,cont3);
% Control variate high estimator
highest(f,h)=highest1(f)+(callcontrol-price4);

% Calculate the low estimator
% For time T-1 the low estimator and the high estimator are equal therefore
% I started the calculation of the low estimator for T-2
est4=zeros(b,1);
```

```
for i=1:b
nj2=zeros(1,b/2);
for j=1:b/2
   if j==1
        cont5= ((sum(est(ini2(i)+j:(i*b)-b/2))*exp(-r*(1-2/3)))...
            +(sum(est((i*b)-(b/2-2):(i*b)))*exp(-r*(1-2/3))))/(b-2);
    else
        if j==b/2
    cont5=(sum(est(ini2(i):b*(i-1)+(b/2-1)))*exp(-r*(1-2/3))...
        +(sum(est(b*(i-1)+(b/2+1):(i*b)-1)))*exp(-r*(1-2/3)))/(b-2);

    else
    cont5=((sum(est(ini2(i):b*(i-1)+j-1))...
        +sum(est(b*(i-1)+j+1:b*(i-1)+j+(b/2-1)))...
        +sum(est(b*(i-1)+j+(b/2+1):i*b)))*exp(-r*(1-2/3)))/(b-2);
        end
   end
if max(max(S1(i),S12(i))-K,0)>= cont5
    nj2(j)= max(max(S1(i),S12(i))-K,0);
else
    nj2(j) = (est(b*(i-1)+j)*exp(-r*(1-2/3))...
        +est(b*(i-1)+j+b/2)*exp(-r*(1-2/3)))/2;

end
end
est4(i,1) = sum(nj2(1:b/2))*2/b;
end
nj3=zeros(1,b/2);
for j=1:b/2
if j==1
        cont6= ((sum(est4(1+j:b-b/2))*exp(-r*(1-2/3))...
            +(sum(est4(b-(b/2-2):b)))*exp(-r*(1-2/3))))/(b-2);
    else
        if j==b/2
    cont6=(sum(est4(1:b/2-1))*exp(-r*(1-2/3))...
        +(sum(est4(b/2+1:b-1)))*exp(-r*(1-2/3)))/(b-2);
    else
    cont6=((sum(est4(1:j-1))...
        +sum(est4(j+1:j+(b/2-1)))...
        +sum(est4(j+(b/2+1):b)))*exp(-r*(1-2/3)))/(b-2);
        end
end
if max(max(S(q),S0(q))-K,0)>= cont6
    nj3(j)= max(max(S(q),S0(q))-K,0);
else
    nj3(j) = (est4(j)*exp(-r*(1-2/3))+est4(j+b/2)*exp(-r*(1-2/3)))/2;

end
end
lowest1(f)=sum(nj3(1:b/2))*2/b;
% Control variate low estimator
lowest(f,h)=lowest1(f)+(callcontrol-price4);

end
%expected value high estimator
he(h,q)=mean(highest(:,h));
%expected value low estimator
le(h,q)=mean(lowest(:,h));
```

```
%average estimator
pointestimate(h,q)= 0.5*max(max(S(q),S0(q))-K,le(h,q))+0.5*he(h,q);
end
% Final Point Estimator
estimator1(1,q)= mean(pointestimate(:,q));
% Final High Estimator
highestimator(1,q)= mean(he(:,q));
% Final Low Estimator
lowestimator(1,q)= mean(le(:,q));
%SE Point Estimator
serr1(1,q)=std(pointestimate(:,q))/sqrt(g);
%SE High Estimator
serrh1(1,q)=std(he(:,q))/sqrt(g);
%SE Low Estimator
serrl1(1,q)=std(le(:,q))/sqrt(g);
%RMSE
RMSE(1,q)=sqrt(mean((pointestimate(:,q)-truevalue(q)).^2));
% 95% quantile
x = norminv(0.95,0,1);
% confidence interval
conflow1(1,q) =(lowestimator(1,q)- x*serrl1(1,q));
confhigh1(1,q)= highestimator(1,q)+ x*serrh1(1,q);
realerror1(1,q)=abs(estimator1(1,q)-truevalue(q))/truevalue(q);
% cpu time
time(q)=toc;
end

%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Monte Carlo Reiteration';
str2 = [name,'   ', num2str(n)];
disp(str2);
name = 'Number of Estimates';
str = [name,'   ', num2str(g)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price    High Estimator      SE      Low Estimator',...
    '      SE         Point Estimator      SE',...
    '       Confidence Interval     True Value   Rel Error',...
    '      RMSE       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g        %-8.3g      %-8.4g      %-8.3g',...
    '        %-8.4g        %-8.3g    %-8.4g  %-8.4g       %-8.4g',...
    '    %-8.3g       %-8.3g             %-8.4g\n'],...
label(i,:),highestimator(i),serrh1(i),lowestimator(i),serrl1(i),...
estimator1(i),serr1(i),conflow1(i),confhigh1(i),truevalue(i),...
realerror1(i),RMSE(i),time(i))
end
```

```
disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

2 Assets Max Option Price Estimation with the Stochastic Mesh Algorithm

```
clear all
K=100;% Strike Price
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1];% Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
n=1465; % Monte Carlo reiteration
b=50;% Branching parameter
S=[70;80;90;100;110;120];% Asset 1 Initial stock price
S0=[70;80;90;100;110;120];% Asset 2 Initial stock price
truevalue=[0.241;1.312;4.333;10.0431;18.12498;27.60773];
o=length(S);
g=25;% Number of estimates
rho=0; % correlation between assets
% random variable generation details
m=[0,0]; % drift
vcv = [1 rho; rho 1]; % covariance matrix

%Creation of the ratespec for the formula of the European rainbow
% option used for pruning
Settle = 'Jun-1-2008';
Maturity = 'Oct-1-2008';
Rates = r;
Basis = 1;
NumDays = daysdif(Settle, Maturity, Basis);
RateSpec = intenvset('ValuationDate', Settle, 'StartDates', Settle,...
'EndDates', Maturity, 'Rates', Rates, 'Compounding', -1, 'Basis', Basis);

%Creation of the ratespec for the formula of the European rainbow
% option used for control variate
Settle1 = 'Jun-1-2008';
Maturity1 = 'Jun-1-2009';
Rates1 = r;
Basis1 = 1;
NumDays1 = daysdif(Settle1, Maturity1, Basis1);
RateSpec1 = intenvset('ValuationDate', Settle1, 'StartDates', Settle1,...
'EndDates', Maturity1, 'Rates', Rates1, 'Compounding', -1,...
'Basis', Basis1);

%variable generation
lowestimator1=zeros(g,o);
lowestimator12=zeros(g,o);
pointestimateouter=zeros(g,o);
estimator=zeros(g,o);
estimatorouter=zeros(g,o);
conflow1=zeros(1,o);
confhigh1=zeros(1,o);
```

```
realerror1=zeros(1,o);
estimator1=zeros(1,o);
highestimator=zeros(1,o);
lowestimator21=zeros(1,o);
serr1=zeros(1,o);
serrh1=zeros(1,o);
serrl1=zeros(1,o);
time=zeros(1,o);
RMSE=zeros(1,o);

for q=1:o
tic
% Control variate constant details
Sigma11 = sigma;
Sigma21 = sigma;
Div11 = delta;
Div21 = delta;
StockSpec11 = stockspec(Sigma11, S(q), 'continuous', Div11);
StockSpec21 = stockspec(Sigma21, S0(q), 'continuous', Div21);
OptSpec = 'call';
% true European price used in the control variate
truecall = maxassetbystulz(RateSpec1, StockSpec11, StockSpec21,...
Settle1, Maturity1, OptSpec, K, rho);

Call=zeros(n,g);
for h=1:g
Q0=zeros(n,1);
L0=zeros(n,1);
for f=1:n
%Generate the mesh
S1=zeros(b,1);
S12=zeros(b,1);
S2=zeros(b,1);
S22=zeros(b,1);

for i=1:b/2
    x=normal(1,m,vcv);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S12(i)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S12(i+b/2)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
    x=normal(1,m,vcv);
    S2(i)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S2(i+b/2)=S1(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S22(i)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S22(i+b/2)=S12(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
end

% Generation of the inputs for the rainbow option used in pruning
AssetPrice1 = S2;
AssetPrice2 = S22;
Sigma1 = sigma;
Sigma2 = sigma;
Div1 = delta;
```

```
Div2 = delta;

StockSpec1 = stockspec(Sigma1, AssetPrice1, 'continuous', Div1);
StockSpec2 = stockspec(Sigma2, AssetPrice2, 'continuous', Div2);
OptSpec = 'call';

% Calculate the mesh
% At T-1 the calculations are straightforward thanks to pruning
Call2=maxassetbystulz(RateSpec, StockSpec1, StockSpec2,...
Settle, Maturity, OptSpec, K, rho);
Q2 =max((max(S2,S22)-K)*exp(-r*(1-1/3)),Call2*exp(-r*(1-1/3)));

%Calculate the weight at time T-2
%first the transition densities

density2 =zeros(b,b);
density21 =zeros(b,b);
density12 =zeros(b,b);

for k=1:b
for i=1:b
   % this is the one dimensional density for the first asset
   density12(i,k)=1/(sigma*sqrt(1/3)*S2(k))*normpdf((log(S2(k)/S1(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the second asset
   density21(i,k)=1/(sigma*sqrt(1/3)*S22(k))*normpdf((log(S22(k)/S12(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % conditional transition densities from one node to another are the
   % product of one-dimensional densities
   density2(i,k)=density12(i,k)*density21(i,k);

end
end

%Calculate the weights
weights2=zeros(b,b);

for k=1:b
for i=1:b
    weights2(i,k)=(density2(k,i))/(sum(density2(1:b,i))/b);
end
end

% Now the estimator at time T-2
Q1=zeros(b,1);
Call1=zeros(b,1);
est2=zeros(b,1);
continuation2=zeros(1,b);
for i=1:b

continuation2(i)= ((sum(Q2.*weights2(1:b,i))/b));

%Mesh estimator
Q1(i)=max(max(max(S1(i),S12(i))-K,0)*exp(-r*(1-2/3)),continuation2(i));

%Control variate part European call estimate at time T-2
```

```
Call1(i)=(sum((Call2*exp(-r*(1-1/3))).*weights2(1:b,i))/b);

end

% Now the estimator at time 0
continuation3= ((sum(Q1)/b));
Q0(f)=max(max(S(q),S0(q))-K,continuation3);
%Control variate part
Call(f,h)=(sum(Call1)/b);

%Calculate the low estimator
lowestimator=Q2;
%variable generation
cont=zeros(b,b/2);
exercise=zeros(b,b/2);
weightcont=zeros(2,b);
weights3=zeros(b-2,b);
weightcont2=zeros(2,b);
weights4=zeros(b-2,b);
weightcont3=zeros(2,b);
%Calculate the low estimator at Time T-2
  for j=1:b/2
  if j==1
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      for k=1:b
      weightcont(1,k)=(density2(k,1))/(sum(density2(1:b,1))/(b));
      weightcont(2,k)=(density2(k,b/2+1))/(sum(density2(1:b,b/2+1))/(b));
      % These weights are used to check if to continue or not
      for i=1:(b-2)/2
      weights3(i,k)=(density2(k,i+1))/(sum(density2(1:b,i+1))/(b));
      weights3(i+(b/2)-1,k)=(density2(k,i+1+b/2))...
          /(sum(density2(1:b,i+1+b/2))/(b));
      end

      cont(k,j)= (sum(lowestimator(2:b/2).*weights3(1:(b-2)/2,k))...
          +sum(lowestimator((b/2)+2:b).*weights3(((b-2)/2)+1:b-2,k)))...
          /(b-2);
      exercise(k,j)=((lowestimator(1)*weightcont(1,k))...
          +(lowestimator(1+b/2)*weightcont(2,k)))/2;
      end

  else
      if j==b/2
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      for k=1:b
      weightcont2(1,k)=(density2(k,b/2))/(sum(density2(1:b,b/2))/(b));
      weightcont2(2,k)=(density2(k,b))/(sum(density2(1:b,b))/(b));
      % These weights are used to check if to continue or not
      for i=1:(b-2)/2
      weights4(i,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      weights4(i+(b/2)-1,k)=(density2(k,i+b/2))...
          /(sum(density2(1:b,i+b/2))/(b));
      end
      cont(k,j)= (sum(lowestimator(1:(b/2)-1).*weights4(1:(b-2)/2,k))...
          +sum(lowestimator((b/2)+1:b-1).*weights4(((b-2)/2)+1:b-2,k)))...
          /(b-2);
```

```
      exercise(k,j)=((lowestimator(b/2)*weightcont2(1,k))...
          +(lowestimator(b)*weightcont2(2,k)))/2;
      end

      else
      % These weights are used  for calculating the estimator if
      % the choice is to continue
      for k=1:b
      weightcont3(1,k)=(density2(k,j))/(sum(density2(1:b,j))/(b));
      weightcont3(2,k)=(density2(k,j+b/2))/(sum(density2(1:b,j+b/2))/(b));

      % These weights are used to check if to continue or not
      for i=1:j-1
      weights5(i,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      for i=j+1:j+b/2-1
      weights6(i-j,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      for i=j+b/2+1:b
      weights7(i-(j+b/2),k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      cont(k,j)= (sum(lowestimator(1:j-1).*weights5(1:j-1,k))...
          +sum(lowestimator(j+1:j+b/2-1).*weights6(1:b/2-1,k))...
          +sum(lowestimator(j+b/2+1:b).*weights7(1:b/2-j,k)))/(b-2);
      exercise(k,j)=(lowestimator(j)*weightcont3(1,k)...
          +lowestimator(j+b/2)*weightcont3(2,k))/2;
      end
      end
  end

  end
  %Calculate the estimator
  % Check whether to exercise of not
  lowestimator2=zeros(1,b);
  est=zeros(b,b/2);
  for i=1:b
      for k=1:b/2
      if (max(max(S1(i),S12(i))-K,0)*exp(-r*(1-2/3)))>= cont(i,k);
          est(i,k)=max(max(S1(i),S12(i))-K,0)*exp(-r*(1-2/3));
      else
          est(i,k)=exercise(i,k);
      end
      end
  lowestimator2(i)=sum(est(i,1:b/2))*2/b;

  end

%  low estimator at time zero
L0(f)=max(max(S(q),S0(q))-K,(sum(lowestimator2)/b));

end

%Estimating beta
%generating the matrix
F=zeros(n,1);
G=zeros(n,2);
H=zeros(n,1)+1;
```

```
for k= 1:n
    F(k)= (Q0(k));
    G(k,1:2)= [H(k),Call(k,h)];
end
%These are the regression coefficients
beta=regress(F,G);

%high estimator corrected with outer control
estimator(h,q)=mean(Q0);
estimatorouter(h,q)=mean(Q0)-beta(2)*(mean(Call(:,h))-truecall);

%Here few lines to calculate the beta and the corrected low estimator
%generating the matrix
F1=zeros(n,1);
G1=zeros(n,2);
H1=zeros(n,1)+1;
for k= 1:n
       F1(k)= (L0(k));
       G1(k,1:2)= [H1(k),Call(k,h)];
end
%These are the regression coefficients
beta=regress(F1,G1);
lowestimator12(h,q)=mean(L0);
%This is the corrected low estimator
lowestimator1(h,q)=mean(L0) -beta(2)*(mean(Call(:,h))-truecall);
%Point Estimator
pointestimateouter(h,q)=0.5*estimatorouter(h,q)+0.5*lowestimator1(h,q);

end
% Final Point Estimator
estimator1(1,q)= mean(pointestimateouter(:,q));
% Final High Estimator
highestimator(1,q)= mean(estimatorouter(:,q));
% Final Low Estimator
lowestimator21(1,q)= mean(lowestimator1(:,q));
% SE Point Estimate
serr1(1,q)=std(pointestimateouter(:,q))/sqrt(g);
%SE High Estimator
serrh1(1,q)=std(estimatorouter(:,q))/sqrt(g);
%SE Low Estimator
serrl1(1,q)=std(lowestimator1(:,q))/sqrt(g);
realerror1(1,q)=abs(estimator1(1,q)-truevalue(q))/truevalue(q);
%RMSE
RMSE(1,q)=sqrt(mean((pointestimateouter(:,q)-truevalue(q)).^2));
% 95% quantile
x = norminv(0.95,0,1);
% confidence interval
conflow1(1,q) =(lowestimator21(1,q)- x*serrl1(1,q));
confhigh1(1,q)= highestimator(1,q)+ x*serrh1(1,q);
% cpu time
time(q)=toc;
end

%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
```

```
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Monte Carlo Reiteration';
str2 = [name,'   ', num2str(n)];
disp(str2);
name = 'Number of Estimates';
str = [name,'   ', num2str(g)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price    High Estimator      SE      Low Estimator',...
    '      SE         Point Estimator      SE',...
    '       Confidence Interval     True Value   Rel Error',...
    '      RMSE       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g        %-8.3g      %-8.4g      %-8.3g',...
    '        %-8.4g        %-8.3g     %-8.4g  %-8.4g       %-8.3g',...
    '    %-8.3g      %-8.3g             %-8.4g\n'],...
label(i,:),highestimator(i),serrh1(i),lowestimator21(i),serrl1(i),...
estimator1(i),serr1(i),conflow1(i),confhigh1(i),truevalue(i),...
realerror1(i),RMSE(i),time(i))
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

2 Assets Max Option Price Estimation with the CVLSM Algorithm

```
clear all
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1]; % Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
reiteration=25;% Number of estimates
b1=10000;% Branching parameter
S0=[70;80;90;100;110;120];% Asset 1 Initial stock price
S01=[70;80;90;100;110;120];% Asset 2 Initial stock price
K0=100;% Strike Price
K=K0/K0;
S=S0/K0;
S02=S01/K0;
truevalue=[0.241;1.259;4.077;9.361;16.924;25.980];
o=length(S);
rho=0.3; % correlation between assets
m=[0,0]; % drift
vcv = [1 rho; rho 1]; % covariance matrix

%Creation of the ratespec for the formula of the European rainbow
% option used for pruning
Settle = 'Jun-1-2008';
Maturity = 'Oct-1-2008';
```

```
Rates = r;
Basis = 1;
NumDays = daysdif(Settle, Maturity, Basis);
RateSpec = intenvset('ValuationDate', Settle, 'StartDates', Settle,...
'EndDates', Maturity, 'Rates', Rates, 'Compounding', -1, 'Basis', Basis);
%Creation of the ratespec for the formula of the European rainbow
% option used at T=2/3
Settle1 = 'Jun-1-2008';
Maturity1 = 'Feb-1-2009';
Rates1 = r;
Basis1 = 1;
NumDays1 = daysdif(Settle1, Maturity1, Basis1);
RateSpec1 = intenvset('ValuationDate', Settle1, 'StartDates', Settle1,...
'EndDates', Maturity1, 'Rates', Rates1, 'Compounding', -1, 'Basis',...
Basis1);
%Creation of the ratespec for the formula of the European rainbow
% option used for control variate
Settle2 = 'Jun-1-2008';
Maturity2 = 'Jun-1-2009';
Rates2 = r;
Basis2 = 1;
NumDays2 = daysdif(Settle2, Maturity2, Basis2);
RateSpec2 = intenvset('ValuationDate', Settle2, 'StartDates', Settle2,...
'EndDates', Maturity2, 'Rates', Rates2, 'Compounding', -1, 'Basis',...
Basis2);

Sigma13 = sigma;
Sigma23 = sigma;
Div13 = delta;
Div23 = delta;

% variable generation
option=zeros(reiteration,o);
inthemoneyperc=zeros(reiteration,o);
estimator=zeros(o,1);
serr=zeros(o,1);
realerror1=zeros(o,1);
time=zeros(o,1);
RMSE=zeros(o,1);
percitm=zeros(o,1);

% Adjustment coefficients to match the established time budget
for q=1:o
tic
if S0(q)==70
    b= b1*24.4;

else
    if S0(q)==80
    b=b1*15.42;

    else
    if S0(q)==90
        b=b1*4.3;

    else
    if S0(q)==100
        b=b1*1.78;
```

```
    else
    if S0(q)==110
        b=b1*1.2;

    else
        if S0(q)==120
        b=b1*1;

        end
    end
    end
    end
    end
end

% Control variate constant details
Sigma12 = sigma;
Sigma22 = sigma;
Div12 = delta;
Div22 = delta;
StockSpec12 = stockspec(Sigma12, S(q), 'continuous', Div12);
StockSpec22 = stockspec(Sigma22, S02(q), 'continuous', Div22);
OptSpec = 'call';
% true European price used in the control variate
callcontrol = maxassetbystulz(RateSpec2, StockSpec12, StockSpec22,...
Settle2, Maturity2, OptSpec, K, rho);

for f=1:reiteration
%Generate the stock paths
S1=zeros(b,1);
S2=zeros(b,1);
S12=zeros(b,1);
S22=zeros(b,1);

for i=1:b/2
    x=normal(1,m,vcv);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S12(i)=S02(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S12(i+b/2)=S02(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
    y=normal(1,m,vcv);
    S2(i)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*y(1));
    S2(i+b/2)=S1(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-y(1)));
    S22(i)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*y(2));
    S22(i+b/2)=S12(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-y(2)));
end

% Generation of the inputs for the rainbow option used in pruning
AssetPrice1 = S2;
AssetPrice2 = S22;
Sigma1 = sigma;
Sigma2 = sigma;
```

```
Div1 = delta;
Div2 = delta;

StockSpec1 = stockspec(Sigma1, AssetPrice1, 'continuous', Div1);
StockSpec2 = stockspec(Sigma2, AssetPrice2, 'continuous', Div2);
OptSpec = 'call';

% Calculate the option price
% At T-1 the calculations are straightforward thanks to pruning
Yt2=maxassetbystulz(RateSpec, StockSpec1, StockSpec2,...
Settle, Maturity, OptSpec, K, rho);

Lt2=max(max(max(S2,S22)-K,0),Yt2);

%At time T-2 we need first to see where the option is in the money
indicator=zeros(b,1);
inthemoney=max(max(S1,S12)-K,0);
for i=1:b
     if inthemoney(i)>0
        indicator(i)=1;
     end

end

%Generate the basic functions
X2=zeros(b,9);
Y=Lt2*exp(-r*1/3);%F
Y1=Yt2*exp(-r*1/3); %Y
Y12=(Yt2.^2)*exp(-r*1/3);%Y^2
Y13=(Lt2.*Yt2)*exp(-r*1/3);%FY
S23=S1;
S24=S12;
for i=1:b
    if indicator(i)==1
        X2(i,1)=1;
        X2(i,2)=max(S23(i),S24(i));
        X2(i,3)=max(S23(i),S24(i))^2;
        X2(i,4)=blsprice(max(S23(i),S24(i)),K,r,2/3,sigma,delta);
        X2(i,5)=blsprice(max(S23(i),S24(i)),K,r,2/3,sigma,delta)^2;
            AssetPrice13 = S23(i);
            AssetPrice23 = S24(i);
            StockSpec13 = stockspec(Sigma13, AssetPrice13,...
                'continuous', Div13);
            StockSpec23 = stockspec(Sigma23, AssetPrice23,...
                'continuous', Div23);
            OptSpec = 'call';

        X2(i,6)=maxassetbystulz(RateSpec1, StockSpec13, StockSpec23,...
            Settle1, Maturity1, OptSpec, K, rho);
        X2(i,7)=X2(i,6)^2;
        X2(i,8)=min(S23(i),S24(i));
        X2(i,9)=S23(i)*S24(i);

    end
end

%These lines delete the zero row from the regression matrix
```

```
X3 = X2(any(X2,2),:);
Y2= Y(any(X2,2),:);%F
Y21= Y1(any(X2,2),:);%Y
Y22= Y12(any(X2,2),:);%Y^2
Y23= Y13(any(X2,2),:);%FY

beta=inv(X3'*X3)*X3'*Y2; %F
beta1=inv(X3'*X3)*X3'*Y21;%Y
beta2=inv(X3'*X3)*X3'*Y22;%Y^2
beta3=inv(X3'*X3)*X3'*Y23;%FY
Continuation=X3*beta;%F
Continuation1=X3*beta1;%Y
Continuation2=X3*beta2;%Y^2
Continuation3=X3*beta3;%FY
coeff=-(Continuation3-Continuation.*Continuation1)./...
    (Continuation2-(Continuation1.^2));
% control variate adjusted continuation value
Contval=Continuation+coeff.*(Continuation1-X3(:,6));
continuation1=zeros(b,1);
S25=S1;
S25(indicator==0)=[];
for i=1:length(Y2)
    n=find(S1==S25(i));
    continuation1(n)=Contval(i);
end
 Lt1=Lt2*exp(-r*1/3);
 Yt1=Yt2*exp(-r*1/3);

 for i=1:b
    if max(max(S1(i),S12(i))-K,0)>0
        if max(max(S1(i),S12(i))-K,0)>continuation1(i)
            Lt1(i,1)=max(max(S1(i),S12(i))-K,0);
            % Generation of the inputs for the rainbow option
            AssetPrice11 = S1(i);
            AssetPrice21 = S12(i);
            Sigma11 = sigma;
            Sigma21 = sigma;
            Div11 = delta;
            Div21 = delta;

            StockSpec11 = stockspec(Sigma11, AssetPrice11,...
                'continuous', Div11);
            StockSpec21 = stockspec(Sigma21, AssetPrice21,...
                'continuous', Div21);
            OptSpec = 'call';

            Yt1(i,1)=maxassetbystulz(RateSpec1, StockSpec11,...
                StockSpec21,Settle1, Maturity1, OptSpec, K, rho);

        end
    end
end

Lt0=Lt1*exp(-r*1/3);
Yt0=Yt1*exp(-r*1/3);

est=mean(Lt0);
```

```
Y0=mean(Yt0);
LY=mean(Lt0.*Yt0);
Ysquared=mean(Yt0.^2);
coefficient=-(LY-est*Y0)/(Ysquared-(Y0^2));
% control variate adjusted continuation value
estimator1=est+coefficient*(Y0-callcontrol);
% Point estimate
option(f,q)=max(max(max(S(q),S02(q))-K,0),estimator1)*100;
% n. of In-the-money paths
inthemoneyperc(f,q)=sum(indicator);
end
% Final  Estimator
estimator(q)= mean(option(:,q));
% SE Estimator
serr(q)=std(option(:,q))/sqrt(reiteration);
realerror1(q)=abs(estimator(q)-truevalue(q))/truevalue(q);
%RMSE
RMSE(q)=sqrt(mean((option(:,q)-truevalue(q)).^2));
% In-the-money paths as a percentage of the branching parameter
percitm(q)=mean(inthemoneyperc(:,q))/b1;
% cpu time
time(q)=toc;
end

%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Number of Estimates';
str = [name,'   ', num2str(reiteration)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price   Estimator      SE        True Value'...
    '   Rel Error   Average time per estimate   RMSE\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g   %-8.3g       %-8.4g    %-8.3g',...
    '            %-8.4g           %-8.3g\n'],...
label(i,:),estimator(i),serr(i),truevalue(i),realerror1(i),...
time(i)/reiteration,RMSE(i))
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

5 Assets Max Option Price Estimation with the Stochastic Tree Algorithm

```
clear all
```

```
K=100;% Strike Price
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1];% Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
n=100;% Monte Carlo reiterations
b =50;% Branching parameter
S=[70;80;90;100;110;120];% Asset 1 Initial Stock price
S0=S;% Asset 2 Initial Stock price
S01=S;% Asset 3 Initial Stock price
S02=S;% Asset 4 Initial Stock price
S03=S;% Asset 5 Initial Stock price
o=length(S);
rho=0.3; % correlation between assets
g=25;% Number of estimates

%variable definition
averageestimator=zeros(o,1);
serrdef=zeros(o,1);
averagehighestimator=zeros(o,1);
serrhdef=zeros(o,1);
averagelowestimator=zeros(o,1);
serrldef=zeros(o,1);
he=zeros(g,o);
le=zeros(g,o);
pointestimate=zeros(g,o);
conflow=zeros(o,1);
confhigh=zeros(o,1);
time=zeros(o,1);

for q=1:o
highest=zeros(n,g);

tic
for h=1:g
call=zeros(1,n);
lowest=zeros(1,n);
for f=1:n
%generate the tree
S1=zeros(b,1);
S2=zeros(b*b,1);
S3=zeros(b*b*b,1);
% underlying 2
S12=zeros(b,1);
S22=zeros(b*b,1);
S32=zeros(b*b*b,1);
% underlying 3
S13=zeros(b,1);
S23=zeros(b*b,1);
S33=zeros(b*b*b,1);
% underlying 4
S14=zeros(b,1);
S24=zeros(b*b,1);
S34=zeros(b*b*b,1);
% underlying 5
S15=zeros(b,1);
S25=zeros(b*b,1);
```

```
S35=zeros(b*b*b,1);

m=[0,0,0,0,0];
vcv = [1 rho rho rho rho; rho 1 rho rho rho; rho rho 1 rho rho;...
    rho rho rho 1 rho; rho rho rho rho 1];

for i=1:b/2
    x=normal(1,m,vcv);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S12(i)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S12(i+b/2)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
    S13(i)=S01(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
    S13(i+b/2)=S01(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(3)));
    S14(i)=S02(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
    S14(i+b/2)=S02(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(4)));
    S15(i)=S03(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(5));
    S15(i+b/2)=S03(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(5)));

end
for i=1:b
for k=1:b/2
    x=normal(1,m,vcv);
    S2(b*(i-1)+k)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(1));
    S2(b*(i-1)+k+b/2)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S22(b*(i-1)+k)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(2));
    S22(b*(i-1)+k+b/2)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S23(b*(i-1)+k)=S13(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(3));
    S23(b*(i-1)+k+b/2)=S13(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(3)));
    S24(b*(i-1)+k)=S14(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(4));
    S24(b*(i-1)+k+b/2)=S14(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(4)));
    S25(b*(i-1)+k)=S15(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(5));
    S25(b*(i-1)+k+b/2)=S15(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(5)));

end
end

for i=1:b*b
for k=1:b/2
    x=normal(1,m,vcv);
    S3(b*(i-1)+k)=S2(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(1));
    S3(b*(i-1)+k+b/2)=S2(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
```

```
        *sqrt(1-2/3)*(-x(1)));
    S32(b*(i-1)+k)=S22(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(2));
    S32(b*(i-1)+k+b/2)=S22(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S33(b*(i-1)+k)=S23(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(3));
    S33(b*(i-1)+k+b/2)=S23(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(3)));
    S34(b*(i-1)+k)=S24(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(4));
    S34(b*(i-1)+k+b/2)=S24(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(4)));
    S35(b*(i-1)+k)=S25(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(5));
    S35(b*(i-1)+k+b/2)=S25(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(5)));
end
end

% Calculate the high estimator
% Calculate option payoff at expiration
X=[S3,S32,S33,S34,S35];
finpay= max(max(X,[],2)-K,0);

% At time T-1
ini=1:b:length(S3);
est=zeros(b*b,1);

for i=1:(b*b)
cont=(sum(finpay(ini(i):ini(i)+b-1))*exp(-r*(1-2/3)))/b;
 Y=[S2(i),S22(i),S23(i),S24(i),S25(i)];
est(i)=max(max(Y)-K,cont);
end

% At time T-2
ini2=1:b:length(S2);
est2=zeros(b,1);
for i=1:(b)
cont2=(sum(est(ini2(i):ini2(i)+b-1))*exp(-r*(1-2/3)))/b;
Y2=[S1(i),S12(i),S13(i),S14(i),S15(i)];
est2(i)=max(max(Y2)-K,cont2);
end

cont3=(sum(est2)*exp(-r*(1-2/3)))/b;
X2=[S(q),S0(q),S01(q),S02(q),S03(q)];
% high estimator at time 0
highest(f,h)=max(max(X2)-K,cont3);

% Calculate the low estimator
est5=zeros(b*b,1);
for i=1:b*b
nj2=zeros(1,b/2);
for j=1:b/2
  if j==1
        cont5= ((sum(finpay(ini(i)+j:(i*b)-b/2))*exp(-r*(1-2/3)))...
            +(sum(finpay((i*b)-(b/2-2):(i*b)))*exp(-r*(1-2/3))))/(b-2);
    else
```

```
        if j==b/2
    cont5=(sum(finpay(ini(i):b*(i-1)+(b/2-1)))*exp(-r*(1-2/3))...
        +(sum(finpay(b*(i-1)+(b/2+1):(i*b)-1)))*exp(-r*(1-2/3)))/(b-2);

    else
    cont5=((sum(finpay(ini(i):b*(i-1)+j-1))...
        +sum(finpay(b*(i-1)+j+1:b*(i-1)+j+(b/2-1)))...
        +sum(finpay(b*(i-1)+j+(b/2+1):i*b)))*exp(-r*(1-2/3)))/(b-2);
        end
   end
   X1=[S2(i),S22(i),S23(i),S24(i),S25(i)];
if max(max(X1)-K,0)>= cont5
    nj2(j)= max(max(X1)-K,0);
else
    nj2(j) = (finpay(b*(i-1)+j)*exp(-r*(1-2/3))...
        +finpay(b*(i-1)+j+b/2)*exp(-r*(1-2/3)))/2;

end
end
est5(i,1) = sum(nj2(1:b/2))*2/b;
end

est4=zeros(b,1);
for i=1:b
for j=1:b/2
if j==1
        cont5= ((sum(est5(ini2(i)+j:(i*b)-b/2))*exp(-r*(1-2/3)))...
            +(sum(est5((i*b)-(b/2-2):(i*b)))*exp(-r*(1-2/3))))/(b-2);
    else
        if j==b/2
    cont5=(sum(est5(ini2(i):b*(i-1)+(b/2-1)))*exp(-r*(1-2/3))...
        +(sum(est5(b*(i-1)+(b/2+1):(i*b)-1)))*exp(-r*(1-2/3)))/(b-2);

    else
    cont5=((sum(est5(ini2(i):b*(i-1)+j-1))...
        +sum(est5(b*(i-1)+j+1:b*(i-1)+j+(b/2-1)))...
        +sum(est5(b*(i-1)+j+(b/2+1):i*b)))*exp(-r*(1-2/3)))/(b-2);
        end
end
   Y3=[S1(i),S12(i),S13(i),S14(i),S15(i)];
if max(max(Y3)-K,0)>= cont5
    nj2(j)= max(max(Y3)-K,0);
else
    nj2(j) = (est5(b*(i-1)+j)*exp(-r*(1-2/3))...
        +est5(b*(i-1)+j+b/2)*exp(-r*(1-2/3)))/2;

end
end
est4(i,1) = sum(nj2(1:b/2))*2/b;
end
nj3=zeros(1,b/2);
for j=1:b/2
if j==1
        cont6= ((sum(est4(1+j:b-b/2))*exp(-r*(1-2/3))...
            +(sum(est4(b-(b/2-2):b)))*exp(-r*(1-2/3))))/(b-2);
    else
        if j==b/2
    cont6=(sum(est4(1:b/2-1))*exp(-r*(1-2/3))...
```

```
        +(sum(est4(b/2+1:b-1)))*exp(-r*(1-2/3)))/(b-2);
    else
    cont6=((sum(est4(1:j-1))+sum(est4(j+1:j+(b/2-1)))...
        +sum(est4(j+(b/2+1):b)))*exp(-r*(1-2/3)))/(b-2);
        end
end
if max(max(X2)-K,0)>= cont6
    nj3(j)= max(max(X2)-K,0);
else
    nj3(j) = (est4(j)*exp(-r*(1-2/3))+est4(j+b/2)*exp(-r*(1-2/3)))/2;

end
end
lowest(f)=sum(nj3(1:b/2))*2/b;

end
%expected value high estimator
he(h,q)=mean(highest(:,h));
%expected value low estimator
le(h,q)=mean(lowest);
%average estimator
pointestimate(h,q)= 0.5*max(max(X2)-K,le(h,q))+0.5*he(h,q);
end
% Final Point Estimator
averageestimator(q)=mean(pointestimate(:,q));
%SE Point Estimator
serrdef(q)=std(pointestimate(:,q))/sqrt(g);
% Final High Estimator
averagehighestimator(q)=mean(he(:,q));
%SE High Estimator
serrhdef(q)=std(he(:,q))/sqrt(g);
% Final Low Estimator
averagelowestimator(q)=mean(le(:,q));
%SE Low Estimator
serrldef(q)=std(le(:,q))/sqrt(g);

% 95% quantile
x = norminv(0.95,0,1);
% confidence interval
conflow(q)=averagelowestimator(q)- x*serrldef(q);
confhigh(q)= averagehighestimator(q)+ x*serrhdef(q);
% cpu time
time(q)=toc;
end
%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Monte Carlo Reiteration';
str2 = [name,'   ', num2str(n)];
disp(str2);
name = 'Number of Estimates';
str = [name,'   ', num2str(g)];
```

```
disp(str);
disp('   ');

% This code is used to generate the table of results
label = char('70','80','90','100','110','120');
fprintf(['Stock Price   High Estimator   SE         Low Estimator',...
    '       SE         Estimator         SE',...
    '            Confidence Interval',...
    '       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g      %-8.3g        %-8.4g      %-8.3g',...
    '        %-8.4g      %-8.3g          %-8.4g  %-8.4g',...
    '               %-8.4g\n'],...
label(i,:),averagehighestimator(i),serrhdef(i),averagelowestimator(i),...
serrldef(i),averageestimator(i),serrdef(i),conflow(i),confhigh(i),time(i))
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

5 Assets Max Option Price Estimation with the Stochastic Mesh Algorithm

```
clear all
K=100;% Strike Price
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1];% Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
n=100;% Monte Carlo reiterations
b=100;% Branching parameter
S=[70;80;90;100;110;120];% Asset 1 Initial Stock price
S0=S;% Asset 2 Initial Stock price
S01=S;% Asset 3 Initial Stock price
S02=S;% Asset 4 Initial Stock price
S03=S;% Asset 5 Initial Stock price
o=length(S);
rho=0; % correlation between assets
g=25;% Number of estimates

%variable generation
estimator=zeros(g,o);
lestimator=zeros(g,o);
optionprice=zeros(g,o);
time=zeros(1,o);
highestimate=zeros(1,o);
serrh=zeros(1,o);
lowestimate=zeros(1,o);
serrl=zeros(1,o);
pointestimate=zeros(1,o);
serr=zeros(1,o);
conflow=zeros(1,o);
confhigh=zeros(1,o);

for q=1:o
```

```
tic
for h=1:g
Q0=zeros(1,n);
lest=zeros(1,n);
L0=zeros(1,n);
for f=1:n
%generate the tree
S1=zeros(b,1);
S2=zeros(b,1);
S3=zeros(b,1);
% underlying 2
S12=zeros(b,1);
S22=zeros(b,1);
S32=zeros(b,1);
% underlying 3
S13=zeros(b,1);
S23=zeros(b,1);
S33=zeros(b,1);
% underlying 4
S14=zeros(b,1);
S24=zeros(b,1);
S34=zeros(b,1);
% underlying 5
S15=zeros(b,1);
S25=zeros(b,1);
S35=zeros(b,1);

m=[0,0,0,0,0];
vcv = [1 rho rho rho rho; rho 1 rho rho rho; rho rho 1 rho rho; ...
    rho rho rho 1 rho; rho rho rho rho 1];

for i=1:b/2
    x=normal(1,m,vcv);
    S1(i)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S1(i+b/2)=S(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S12(i)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S12(i+b/2)=S0(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
    S13(i)=S01(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
    S13(i+b/2)=S01(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(3)));
    S14(i)=S02(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
    S14(i+b/2)=S02(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(4)));
    S15(i)=S03(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *x(5));
    S15(i+b/2)=S03(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(5)));

    x=normal(1,m,vcv);
    S2(i)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S2(i+b/2)=S1(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S22(i)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S22(i+b/2)=S12(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S23(i)=S13(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
```

```
    S23(i+b/2)=S13(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(3)));
    S24(i)=S14(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
    S24(i+b/2)=S14(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(4)));
    S25(i)=S15(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(5));
    S25(i+b/2)=S15(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(5)));

    x=normal(1,m,vcv);
    S3(i)=S2(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S3(i+b/2)=S2(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S32(i)=S22(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*x(2));
    S32(i+b/2)=S22(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S33(i)=S23(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
    S33(i+b/2)=S23(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(3)));
    S34(i)=S24(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
    S34(i+b/2)=S24(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(4)));
    S35(i)=S25(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(5));
    S35(i+b/2)=S25(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(5)));
end

%Calculate the mesh estimator
% Calculate option payoff at expiration
X=[S3,S32,S33,S34,S35];
Q3=max(max(X,[],2)-K,0)*exp(-r*(1));

%Calculate the weight at time T-1
%first the transition densities

density =zeros(b,b);
density1 =zeros(b,b);
density12 =zeros(b,b);
density13 =zeros(b,b);
density14 =zeros(b,b);
density15 =zeros(b,b);

for k=1:b
for i=1:b
   % this is the one dimensional density for the first asset
   density1(i,k)=1/(sigma*sqrt(1/3)*S3(k))*normpdf((log(S3(k)/S2(i))-...
       (r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the second asset
   density12(i,k)=1/(sigma*sqrt(1/3)*S32(k))*normpdf((log(S32(k)/S22(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the third asset
   density13(i,k)=1/(sigma*sqrt(1/3)*S33(k))*normpdf((log(S33(k)/S23(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the fourth asset
   density14(i,k)=1/(sigma*sqrt(1/3)*S34(k))*normpdf((log(S34(k)/S24(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the fifth asset
```

```
   density15(i,k)=1/(sigma*sqrt(1/3)*S35(k))*normpdf((log(S35(k)/S25(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % conditional transition densities from one node to another are the
   % product of one-dimensional densities
   density(i,k)=density1(i,k)*density12(i,k)*density13(i,k)...
       *density14(i,k)*density15(i,k);
end
end

% Calculate the weights
weights=zeros(b,b);

for k=1:b
for i=1:b
    weights(i,k)=(density(k,i))/(sum(density(1:b,i))/b);
end
end

% Now the estimator at time T-1
Q2=zeros(b,1);
for i=1:b
    Y=[S2(i),S22(i),S23(i),S24(i),S25(i)];
    Q2(i)=max(max(max(Y)-K,0)*exp(-r*(1-1/3)),(sum(Q3.*weights(1:b,i))/b));
end

%Calculate the weight at time T-2
%first the transition densities

density2 =zeros(b,b);
density21 =zeros(b,b);
density22 =zeros(b,b);
density23 =zeros(b,b);
density24 =zeros(b,b);
density25 =zeros(b,b);

for k=1:b
for i=1:b
    % this is the one dimensional density for the first asset
   density21(i,k)=1/(sigma*sqrt(1/3)*S2(k))*normpdf((log(S2(k)/S1(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the second asset
   density22(i,k)=1/(sigma*sqrt(1/3)*S22(k))*normpdf((log(S22(k)/S12(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the third asset
   density23(i,k)=1/(sigma*sqrt(1/3)*S23(k))*normpdf((log(S23(k)/S13(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the fourth asset
   density24(i,k)=1/(sigma*sqrt(1/3)*S24(k))*normpdf((log(S24(k)/S14(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % this is the one dimensional density for the fifth asset
   density25(i,k)=1/(sigma*sqrt(1/3)*S25(k))*normpdf((log(S25(k)/S15(i))...
       -(r-delta-sigma^2/2)*1/3)/(sigma* sqrt(1-2/3)));
   % conditional transition densities from one node to another are the
   % product of one-dimensional densities
   density2(i,k)=density21(i,k)*density22(i,k)*density23(i,k)...
       *density24(i,k)*density25(i,k);
```

```
end
end

%Calculate the weights
weights2=zeros(b,b);

for k=1:b
for i=1:b
    weights2(i,k)=(density2(k,i))/(sum(density2(1:b,i))/b);
end
end

% Now the estimator at time T-2
Q1=zeros(b,1);
for i=1:b
    Y2=[S1(i),S12(i),S13(i),S14(i),S15(i)];
    Q1(i)=max(max(max(Y2)-K,0)*exp(-r*(1-2/3)),(sum(Q2.*weights2(1:b,i))...
        /b));

end

% Now the estimator at time 0
X2=[S(q),S0(q),S01(q),S02(q),S03(q)];
Q0(f)=max(max(X2)-K,(sum(Q1)/b));

%Calculate the low estimator
%Calculate the low estimator at time T-1
%variable generation
cont=zeros(b,b/2);
exercise=zeros(b,b/2);
weightcont=zeros(2,b);
weights3=zeros(b-2,b);
weightcont2=zeros(2,b);
weights4=zeros(b-2,b);
weightcont3=zeros(2,b);

  for j=1:b/2
  if j==1
      for k=1:b
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      weightcont(1,k)=(density(k,1))/(sum(density(1:b,1))/(b));
      weightcont(2,k)=(density(k,b/2+1))/(sum(density(1:b,b/2+1))/(b));

      for i=1:(b-2)/2
      % These weights are used to check if to continue or not
      weights3(i,k)=(density(k,i+1))/(sum(density(1:b,i+1))/(b));
      weights3(i+(b/2)-1,k)=(density(k,i+1+b/2))/...
          (sum(density(1:b,i+1+b/2))/(b));
      end

      cont(k,j)= (sum(Q3(2:b/2).*weights3(1:(b-2)/2,k))...
          +sum(Q3((b/2)+2:b).*weights3(((b-2)/2)+1:b-2,k)))/(b-2);
      exercise(k,j)=((Q3(1)*weightcont(1,k))+(Q3(1+b/2)...
          *weightcont(2,k)))/2;
      end
```

```
else
    if j==b/2
    for k=1:b
    % These weights are used  for calculating the estimator if the
    % choice is to continue
    weightcont2(1,k)=(density(k,b/2))/(sum(density(1:b,b/2))/(b));
    weightcont2(2,k)=(density(k,b))/(sum(density(1:b,b))/(b));
    for i=1:(b-2)/2
    % These weights are used to check if to continue or not
    weights4(i,k)=(density(k,i))/(sum(density(1:b,i))/(b));
    weights4(i+(b/2)-1,k)=(density(k,i+b/2))/(sum(density(1:b,i+b/2))...
        /(b));
    end
    cont(k,j)= (sum(Q3(1:(b/2)-1).*weights4(1:(b-2)/2,k))...
        +sum(Q3((b/2)+1:b-1).*weights4(((b-2)/2)+1:b-2,k)))/(b-2);
    exercise(k,j)=((Q3(b/2)*weightcont2(1,k))+(Q3(b)...
        *weightcont2(2,k)))/2;
    end

    else
    for k=1:b
    % These weights are used  for calculating the estimator if the
    % choice is to continue
    weightcont3(1,k)=(density(k,j))/(sum(density(1:b,j))/(b));
    weightcont3(2,k)=(density(k,j+b/2))/(sum(density(1:b,j+b/2))/(b));

    % These weights are used to check if to continue or not
    for i=1:j-1
    weights5(i,k)=(density(k,i))/(sum(density(1:b,i))/(b));
    end
    for i=j+1:j+b/2-1
    weights6(i-j,k)=(density(k,i))/(sum(density(1:b,i))/(b));
    end
    for i=j+b/2+1:b
    weights7(i-(j+b/2),k)=(density(k,i))/(sum(density(1:b,i))/(b));
    end
    cont(k,j)= (sum(Q3(1:j-1).*weights5(1:j-1,k))...
        +sum(Q3(j+1:j+b/2-1).*weights6(1:b/2-1,k))...
        +sum(Q3(j+b/2+1:b).*weights7(1:b/2-j,k)))/(b-2);
    exercise(k,j)=(Q3(j)*weightcont3(1,k)+Q3(j+b/2)*weightcont3(2,k))/2;
    end
    end
end

end
%Calculate the estimator
% Check whether to exercise of not
lowestimator=zeros(b,1);
est=zeros(b,b/2);
for i=1:b
    Y=[S2(i),S22(i),S23(i),S24(i),S25(i)];
    for k=1:b/2
    if (max(max(Y)-K,0)*exp(-r*(1-1/3)))>= cont(i,k);
        est(i,k)=max(max(Y)-K,0)*exp(-r*(1-1/3));
    else
        est(i,k)=exercise(i,k);
    end
    end
```

```
  lowestimator(i)=sum(est(i,1:b/2))*2/b;

  end

%Calculate the low estimator at time T-2
  for j=1:b/2
  if j==1
      for k=1:b
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      weightcont(1,k)=(density2(k,1))/(sum(density2(1:b,1))/(b));
      weightcont(2,k)=(density2(k,b/2+1))/(sum(density2(1:b,b/2+1))/(b));
      for i=1:(b-2)/2
      % These weights are used to check if to continue or not
      weights3(i,k)=(density2(k,i+1))/(sum(density2(1:b,i+1))/(b));

      weights3(i+(b/2)-1,k)=(density2(k,i+1+b/2))...
          /(sum(density2(1:b,i+1+b/2))/(b));
      end

      cont(k,j)= (sum(lowestimator(2:b/2).*weights3(1:(b-2)/2,k))...
          +sum(lowestimator((b/2)+2:b).*weights3(((b-2)/2)+1:b-2,k)))...
          /(b-2);
      exercise(k,j)=((lowestimator(1)*weightcont(1,k))...
          +(lowestimator(1+b/2)*weightcont(2,k)))/2;
      end

  else
      if j==b/2
      for k=1:b
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      weightcont2(1,k)=(density2(k,b/2))/(sum(density2(1:b,b/2))/(b));
      weightcont2(2,k)=(density2(k,b))/(sum(density2(1:b,b))/(b));
      for i=1:(b-2)/2
      % These weights are used to check if to continue or not
      weights4(i,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));

      weights4(i+(b/2)-1,k)=(density2(k,i+b/2))...
          /(sum(density2(1:b,i+b/2))/(b));
      end
      cont(k,j)= (sum(lowestimator(1:(b/2)-1).*weights4(1:(b-2)/2,k))...
          +sum(lowestimator((b/2)+1:b-1).*weights4(((b-2)/2)+1:b-2,k)))...
          /(b-2);
      exercise(k,j)=((lowestimator(b/2)*weightcont2(1,k))...
          +(lowestimator(b)*weightcont2(2,k)))/2;
      end

      else
      for k=1:b
      % These weights are used  for calculating the estimator if the
      % choice is to continue
      weightcont3(1,k)=(density2(k,j))/(sum(density2(1:b,j))/(b));
      weightcont3(2,k)=(density2(k,j+b/2))/(sum(density2(1:b,j+b/2))/(b));

      % These weights are used to check if to continue or not
      for i=1:j-1
```

```
      weights5(i,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      for i=j+1:j+b/2-1
      weights6(i-j,k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      for i=j+b/2+1:b
      weights7(i-(j+b/2),k)=(density2(k,i))/(sum(density2(1:b,i))/(b));
      end
      cont(k,j)= (sum(lowestimator(1:j-1).*weights5(1:j-1,k))...
          +sum(lowestimator(j+1:j+b/2-1).*weights6(1:b/2-1,k))...
          +sum(lowestimator(j+b/2+1:b).*weights7(1:b/2-j,k)))/(b-2);
      exercise(k,j)=(lowestimator(j)*weightcont3(1,k)...
          +lowestimator(j+b/2)*weightcont3(2,k))/2;
      end
      end
  end

  end
  %Calculate the estimator
  % Check whether to exercise of not
  lowestimator2=zeros(1,b);
  est=zeros(b,b/2);
  for i=1:b
      Y2=[S1(i),S12(i),S13(i),S14(i),S15(i)];
      for k=1:b/2
      if (max(max(Y2)-K,0)*exp(-r*(1-2/3)))>= cont(i,k);
          est(i,k)=max(max(Y2)-K,0)*exp(-r*(1-2/3));
      else
          est(i,k)=exercise(i,k);
      end
      end
  lowestimator2(i)=sum(est(i,1:b/2))*2/b;
  end

%  low estimator at time zero
L0(f)=max(max(S(q),S0(q))-K,(sum(lowestimator2)/b));

end

estimator(h,q)=mean(Q0);
lestimator(h,q)=mean(L0);
optionprice(h,q)=0.5*estimator(h,q)+0.5*lestimator(h,q);
end
% Final High Estimator
highestimate(q)=mean(estimator(:,q));
%SE High Estimator
serrh(q)=std(estimator(:,q))/sqrt(g);
% Final Low Estimator
lowestimate(q)=mean(lestimator(:,q));
%SE Low Estimator
serrl(q)=std(lestimator(:,q))/sqrt(g);
% Final Point Estimator
pointestimate(q)=mean(optionprice(:,q));
% SE Point Estimate
serr(q)=std(optionprice(:,q))/sqrt(g);

% 95% quantile
x = norminv(0.95,0,1);
```

```
% confidence interval
conflow(q)=max(max(X2)-K,lowestimate(q)- x*serrl(q));
confhigh(q)= highestimate(q)+ x*serrh(q);
% cpu time
time(q)=toc;
end

%Building the table
dataLength = length(S);
%These are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Monte Carlo Reiteration';
str2 = [name,'   ', num2str(n)];
disp(str2);
name = 'Number of Estimates';
str = [name,'   ', num2str(g)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf(['Stock Price   High Estimator   SE         Low Estimator',...
    '       SE         Estimator         SE',...
    '           Confidence Interval',...
    '       Average time per Estimate(in sec)\n']);
for i=1:dataLength
fprintf(['%8s        %-8.4g      %-8.3g        %-8.4g      %-8.3g',...
    '        %-8.4g      %-8.3g          %-8.4g  %-8.4g',...
    '             %-8.4g\n'],...
label(i,:),highestimate(i),serrh(i),lowestimate(i),serrl(i)...
,pointestimate(i),serr(i),conflow(i),confhigh(i),time(i))
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```

5 Assets Max Option Price Estimation with the LSM Algorithm

```
clear all
delta=0.1;% Dividend Yield
T=1;% Time to maturity
sigma=0.2;% Volatility
t=[0;1/3;2/3;1]; % Exercise opportunities for the Bermudan option
r=0.05;% Risk-free rate
reiteration=25;% Number of estimates
b=25000;% Branching parameter
S01=[70;80;90;100;110;120];% Asset 1 Initial stock price
S02=S01;% Asset 2 Initial Stock price
S03=S01;% Asset 3 Initial Stock price
S04=S01;% Asset 4 Initial Stock price
```

```
S05=S01;% Asset 5 Initial Stock price
K0=100;% Strike Price
K=K0/K0;
s=S01/K0;
s2=S02/K0;
s3=S03/K0;
s4=S04/K0;
s5=S05/K0;
o=length(s);
rho=0; % correlation between assets
m=[0,0,0,0,0];
vcv = [1 rho rho rho rho; rho 1 rho rho rho; rho rho 1 rho rho;...
    rho rho rho 1 rho; rho rho rho rho 1];

% variable generation
option=zeros(reiteration,o);
inthemoneyperc=zeros(reiteration,o);
estimator=zeros(o,1);
serr=zeros(o,1);
time=zeros(o,1);
percitm=zeros(o,1);

for q=1:o
tic
for f=1:reiteration
%generate the tree
S1=zeros(b,1);
S2=zeros(b,1);
S3=zeros(b,1);
% underlying 2
S12=zeros(b,1);
S22=zeros(b,1);
S32=zeros(b,1);
% underlying 3
S13=zeros(b,1);
S23=zeros(b,1);
S33=zeros(b,1);
% underlying 4
S14=zeros(b,1);
S24=zeros(b,1);
S34=zeros(b,1);
% underlying 5
S15=zeros(b,1);
S25=zeros(b,1);
S35=zeros(b,1);

for i=1:b/2
    x=normal(1,m,vcv);
    S1(i)=s(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S1(i+b/2)=s(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(1)));
    S12(i)=s2(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S12(i+b/2)=s2(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(2)));
    S13(i)=s3(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
    S13(i+b/2)=s3(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(3)));
    S14(i)=s4(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
```

```
    S14(i+b/2)=s4(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(4)));
    S15(i)=s5(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(5));
    S15(i+b/2)=s5(q)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)...
        *(-x(5)));

    x=normal(1,m,vcv);
    S2(i)=S1(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S2(i+b/2)=S1(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S22(i)=S12(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S22(i+b/2)=S12(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S23(i)=S13(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
    S23(i+b/2)=S13(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(3)));
    S24(i)=S14(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
    S24(i+b/2)=S14(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(4)));
    S25(i)=S15(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(5));
    S25(i+b/2)=S15(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(5)));

    x=normal(1,m,vcv);
    S3(i)=S2(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(1));
    S3(i+b/2)=S2(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(1)));
    S32(i)=S22(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(2));
    S32(i+b/2)=S22(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(2)));
    S33(i)=S23(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(3));
    S33(i+b/2)=S23(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(3)));
    S34(i)=S24(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(4));
    S34(i+b/2)=S24(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(4)));
    S35(i)=S25(i)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma*sqrt(1-2/3)*x(5));
    S35(i+b/2)=S25(i+b/2)*exp((r-delta-sigma^2/2)*(1-2/3)+sigma...
        *sqrt(1-2/3)*(-x(5)));

end

% Calculate option payoff at expiration
X=[S3,S32,S33,S34,S35];
Lt2=max(max(X,[],2)-K,0);

%At time T-1 we need first to see where the option is in the money
indicator=zeros(b,1);
X1=[S2,S22,S23,S24,S25];
inthemoney=max(max(X1,[],2)-K,0);
for i=1:b
     if inthemoney(i)>0
        indicator(i)=1;
     end

end
```

```
%Generate the basic functions
X2=zeros(b,19);
Y=Lt2*exp(-r*1/3);%F

for i=1:b
    if indicator(i)==1
    Y0=[S2(i),S22(i),S23(i),S24(i),S25(i)];
    Y1=sort(Y0,'descend');
        X2(i,1)=1;
        X2(i,2)=Y1(1);
        X2(i,3)=Y1(1)^2-1;
        X2(i,4)=Y1(1)^3-3*Y1(1);
        X2(i,5)=Y1(1)^4-6*Y1(1)^2+3;
        X2(i,6)=Y1(1)^5-10*Y1(1)^3+15*Y1(1);
        X2(i,7)=Y1(2);
        X2(i,8)=Y1(2)^2;
        X2(i,9)=Y1(3);
        X2(i,10)=Y1(3)^2;
        X2(i,11)=Y1(4);
        X2(i,12)=Y1(4)^2;
        X2(i,13)=Y1(5);
        X2(i,14)=Y1(5)^2;
        X2(i,15)=Y1(1)*Y1(2);
        X2(i,16)=Y1(2)*Y1(3);
        X2(i,17)=Y1(3)*Y1(4);
        X2(i,18)=Y1(4)*Y1(5);
        X2(i,19)=Y1(1)*Y1(2)*Y1(3)*Y1(4)*Y1(5);
    end
end

%These lines delete the zero rows from the regression matrix
X3 = X2(any(X2,2),:);
Y2= Y(any(X2,2),:);%F
beta=inv(X3'*X3)*X3'*Y2; %F
Continuation=X3*beta;%F
Contval=Continuation;
continuation1=zeros(b,1);
S26=S2;
S26(indicator==0)=[];
for i=1:length(Y2)
    n=find(S2==S26(i));
    continuation1(n)=Contval(i);
end
 Lt1=Lt2*exp(-r*1/3);
 for i=1:b
     Y0=[S2(i),S22(i),S23(i),S24(i),S25(i)];
    if max(max(Y0)-K,0)>0
        if max(max(Y0)-K,0)>continuation1(i)
            Lt1(i,1)=max(max(Y0)-K,0);

        end
    end
end

%At time T-2 we need first to see where the option is in the money
indicator=zeros(b,1);
X1=[S1,S12,S13,S14,S15];
inthemoney=max(max(X1,[],2)-K,0);
```

```
for i=1:b
     if inthemoney(i)>0
        indicator(i)=1;
     end
end

%Generate the basic functions
X2=zeros(b,19);
Y=Lt1*exp(-r*1/3);

for i=1:b
    if indicator(i)==1
    Y0=[S1(i),S12(i),S13(i),S14(i),S15(i)];
    Y1=sort(Y0,'descend');
        X2(i,1)=1;
        X2(i,2)=Y1(1);
        X2(i,3)=Y1(1)^2-1;
        X2(i,4)=Y1(1)^3-3*Y1(1);
        X2(i,5)=Y1(1)^4-6*Y1(1)^2+3;
        X2(i,6)=Y1(1)^5-10*Y1(1)^3+15*Y1(1);
        X2(i,7)=Y1(2);
        X2(i,8)=Y1(2)^2;
        X2(i,9)=Y1(3);
        X2(i,10)=Y1(3)^2;
        X2(i,11)=Y1(4);
        X2(i,12)=Y1(4)^2;
        X2(i,13)=Y1(5);
        X2(i,14)=Y1(5)^2;
        X2(i,15)=Y1(1)*Y1(2);
        X2(i,16)=Y1(2)*Y1(3);
        X2(i,17)=Y1(3)*Y1(4);
        X2(i,18)=Y1(4)*Y1(5);
        X2(i,19)=Y1(1)*Y1(2)*Y1(3)*Y1(4)*Y1(5);
    end
end

%These lines delete the zero rows from the regression matrix
X3 = X2(any(X2,2),:);
Y2= Y(any(X2,2),:);%F
beta=inv(X3'*X3)*X3'*Y2; %F
Continuation=X3*beta;%F
Contval=Continuation;
continuation1=zeros(b,1);
S16=S1;
S16(indicator==0)=[];
for i=1:length(Y2)
    n=find(S1==S16(i));
    continuation1(n)=Contval(i);
end
 Lt0=Lt1*exp(-r*1/3);

 for i=1:b
     Y0=[S1(i),S12(i),S13(i),S14(i),S15(i)];
    if max(max(Y0)-K,0)>0
        if max(max(Y0)-K,0)>continuation1(i)
            Lt0(i,1)=max(max(Y0)-K,0);

        end
```

```
    end
end

Lt=Lt0*exp(-r*1/3);
est=mean(Lt);
estimator1=est;
X5=[s(q),s2(q),s3(q),s4(q),s5(q)];
% Point estimate
option(f,q)=max(max(max(X5)-K,0),estimator1)*100;
% n. of In-the-money paths
inthemoneyperc(f,q)=sum(indicator);
end
% Final Estimator
estimator(q)= mean(option(:,q));
% SE Estimator
serr(q)=std(option(:,q))/sqrt(reiteration);
% In-the-money paths as a percentage of the branching parameter
percitm(q)=mean(inthemoneyperc(:,q))/b;
% cpu time
time(q)=toc;
end

%Building the table
dataLength = length(s);
%This are lines used to space between an output and the preceding one
disp('   ');
disp('   ');
disp('   ');
name = 'Branching Parameter';
str1 = [name,'   ', num2str(b)];
disp(str1);
name = 'Number of Estimates';
str = [name,'   ', num2str(reiteration)];
disp(str);
disp('   ');

% This code is used to generate the table of results.
label = char('70','80','90','100','110','120');
fprintf('Stock Price   Estimator      SE   Average time per estimate\n');
for i=1:dataLength
fprintf('%8s        %-8.4g   %-8.3g            %-8.4g\n',...
label(i,:),estimator(i),serr(i),time(i)/reiteration)
end

disp('   ');
% Generate the output for the total cpu time computation
name = 'Time Elapsed for constructing the entire table (in sec)';
str1 = [name,'   ', num2str(sum(time))];
disp(str1);
```