MATLAB APPLICATION OF THE LEAST SQUARES MONTE CARLO SIMULATION METHOD ON PRICING AMERICAN-STYLE OPTIONS

TZATZIMAKI A. ALEXANDRA

Dissertation submitted

at the Department of Accounting and Finance

in partial fulfillment of the necessary prerequisites

for the acquisition of the MSc Degree

Athens

[November, 2016]

We approve the dissertation of TZATZIMAKI A. ALEXANDRA

TSEKREKOS ANDRIANOS

KAVOUSANOS EMMANOYIL

CHALAMANDARIS GEORGIOS

November, 2016

ΒΕΒΑΙΩΣΗ ΕΚΠΟΝΗΣΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

«Δηλώνω υπεύθυνα ότι η συγκεκριμένη πτυχιακή εργασία για τη λήψη του Μεταπτυχιακού Διπλώματος Ειδίκευσης στη Λογιστική και Χρηματοοικονομική έχει συγγραφεί από εμένα προσωπικά και δεν έχει υποβληθεί ούτε έχει εγκριθεί στο πλαίσιο κάποιου άλλου μεταπτυχιακού ή προπτυχιακού τίτλου σπουδών, στην Ελλάδα ή στο εξωτερικό. Η εργασία αυτή έχοντας εκπονηθεί από εμένα, αντιπροσωπεύει τις προσωπικές μου απόψεις επί του θέματος. Οι πηγές στις οποίες ανέτρεξα για την εκπόνηση της συγκεκριμένης διπλωματικής αναφέρονται στο σύνολό τους, δίνοντας πλήρεις αναφορές στους συγγραφείς, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο».

ΤΖΑΤΖΙΜΑΚΗ Α. ΑΛΕΞΑΝΔΡΑ

CONTENTS

1. ABSTRACT

The aim of this dissertation is the modeling of the algorithm that Longstaff & Schwartz proposed in 2001 as an alternative method for pricing American-style options. The algorithm, known as the Least-Squares Monte Carlo method, is named after the innovative techniques its instigators combined as the tool of their valuation. The key of their innovation is the usage of a simple Least-Squares regression in order to approximate the continuation value, while the path/s of the stochastic variable/s is/are simulated under the desired stochastic process using the Monte Carlo simulation method. The application of the method here expands in two different models; one containing one stochastic variable, the underlying asset's price, and a second one containing two stochastic variables, the underlying asset's price and the interest rate. Due to the nature of the algorithm, it is easy applicable under various conditions and, furthermore, under various types of options. In cases that traditional methods cannot produce a valid, regarding computational time and accuracy, result or even cannot produce a result at all, such as path-dependent options with multiple stochastic factors, the algorithm of Longstaff & Schwartz (2001) makes the pricing possible. Reaching a conclusion, it was quite impressive to end up having exactly the same price for different type and number of basis functions, proving that the algorithm is pretty robust yielding fully consistent results in all possible cases. In addition, in general context and in most cases, when the underlying asset's price follows a Geometric Brownian Motion, the option found out worthing more than that under any other stochastic process. MatLab was the tool used in this application of the LSM algorithm in a programming environment.

2. INTRODUCTION

Options are the main products of financial engineering. Traded either on exchanges or in over-the-counter markets, they are divided into two types: calls and puts. The former give their owner the right to buy a prespecified quantity of the underlying asset at a prespecified price at the expiration date of the option or over some prespecified time period, while the latter give their owner the right to sell a prespecified quantity of the underlying asset at a prespecified price at the expiration date of the option or over some prespecified time period.

Options, nowadays, are the key instrument for hedging and speculation. Concerning their hedging potential, the market players, individuals or institutionals, use them as an insurance policy or/and as a limitation of their losses, meaning that the payoff from the derivative can counterpoise the loss of an investment from an adverse movement of the market. Concerning their speculation potential, they can be used as a kind of betting in order to make profit "out of nowhere", requiring forecasting the direction towards which the market will move, the magnitude and the timing of anticipated changes. Their main advantage is as much the fact that the profit can derive even from a down-movement of the market as that the owner can avoid the great loss from a down-movement of the market by not exercising it, as long as he/she has only the right, not the obligation, to exercise it.

There are, generally, two kinds of options: European and American. European are the options that are to be executed only at their expiration date. On the other hand, American options are executable at different and multiple times, prespecified or not, during their lifetime. Compared to their European counterparts, they are much more popular, as they are traded in a wide range of markets, such us equity, commodity, foreign exchange and credit, and much more flexible. Indicatively, such financial products are call and put options on dividend-paying stocks, put options on nondividend-paying stocks, foreign exchange options, commodity options, commodity future options and index options.

American options, as the markets develop and evolve, are becoming more complex and sophisticated, making their valuation more and more difficult, insomuch no analytic

formula, like that of Black & Scholes (1973)[1] for the European ones, exists. Their valuation is much more complicated than that of the European options, because the potential of early exercise has to be measured as well. That's the need that a lot of papers, authors and students, theoreticians and practitioners, have been called to satisfy by producing models of pricing that kind of options either via numerical or analytical approximations[2].

The existing numerical procedures for pricing American options are the following:

- Binomial Tree by Cox, Ross & Rubinstein (1979)[3]
  This is the ideal and most efficient way of pricing American-style options.
  The general scheme, which basically describes the different paths that the stock price might follow during the life of the option, under which Cox, Ross and Rubinstein worked is as follows: the time until the expiration date of the option is divided into a large numbers of time intervals (the smallest the time interval the more accurate the price calculation.)[4]. At each time interval, there is a possibility of an up movement for the stock price by a certain percentage amount at the next time interval and that of a down movement, by a certain percentage amount as well. The value of the option at each node depends on the type of the option, with the payoff of a call to be max(St-K,0) and that of a put max (K-St,0)[5]. The price is then derived through backward induction, working from the end to the beginning. At each node, the value of the option is the maximum value between the continuation value, meaning the value that the option would have if exercised immediately, and the discounted expected value that it would have if it was held another timer period.

1 See Black F. & M. Scholes, 1973, "The Pricing of Options and Corporate Liabilities", The Journal of Political Economy, Volume 81, Issue 3, 637-654

2 See Literature Review in page 17.

3 See Cox J. C., S. Ross, M. Rubinstein, 1979, "Option Pricing: A Simplified Approach", Journal of Financial Economics, 7, 229-263

4 For European calls and puts, the result of the binomial tree method converges with that of the Black & Scholes valuation for very small time intervals.

5 Where K: strike price and St: the price of the underlying asset at time t

Another type of tree regards the trinomial tree. The only difference, compared to the binomial tree, is the fact that at each node, the stock price can either go up, stay stable or go down. The rest methodology is as described before.

- Finite Differences Methods by Brennan & Schwartz (1977)[6]

  That's a method that works by solving the differential equation the option satisfies. By converting the equation into a set of difference equations and by solving them iteratively, the price of the option then arises. The computation is conducted backwards, just like the tree approach. There are two kinds of finite differences methods: the explicit finite difference method, which is functionally the same as using the trinomial tree that was described above, and the implicit finite difference method, which, leaving aside the fact that is much more complicated, has the advantage that convergence comes off without taking any special precautions. The explicit finite difference method solves the partial differential equation describing option value evolution.

  This method is suggested for pricing American-style options, like the binomial tree method, but has the disadvantage of a very large increase in the computational time when applied for more than one state variables.

  Neither the binomial tree method nor the finite differences method are applicable for cases of options with path-dependent payoff.

- Monte Carlo Simulation

  That's the method that solves the problem of pricing path-dependent titles. The methodology underlying this method is the following:

  1. Sample a random path (such as Arithmetic Brownian Motion, Geometric Brownian Motion or Ornstein Uhlenberg) for the underlying asset price S using risk-neutral probabilities.
  2. Calculate the payoff from the derivative.
  3. Repeat step 1 and 2 to get many sample values of the payoff from the derivative.

6 See Brennan M. J. & E. S. Schwartz, May 1977, "The Valuation of American Put Options", The Journal of Finance, Volume 32, Issue 2, p. 449-462.

4. Calculate the mean of the sample payoffs.
5. Discount this expected payoff at the risk-free rate to get an estimate of the value of the derivative.

As implied, the application of Monte Carlo simulation only narrows down to the production of random numbers ~ N $(0,1)$[7], the quantity of which depends on the accuracy of the result one needs and the computational power that is available, and then just considers the price of the option as the expected present value of its future payoff. One of the pros of this procedure is that it's the only one, of those described, that can handle many stochastic variables, while one of the cons is that is very time-consuming and cannot easily handle cases with early exercise opportunities. It works forwards, from the beginning till the end of the life of the option and is the most flexible of all the methods described.

Concerning all the above characteristics of each method, it is logical to conclude that the final choice of the method used to price an American-style option depends on the special characteristics that the specific call or put has and the accuracy required. That is, it depends on the path the underlying asset follows, whether it incorporates all the history of the asset or not, on the number of underlying variables, on the number of the variables that are not deterministic, but stochastic. Such factors are the stock price, the interest rates, the volatility and the dividend yield.

In practice, a call or a put may feature one or more contradicting characteristics. For example, which is the most suitable method for valuating an American-style put, the payoff of which depends on the value of it in the previous time step, in a market with stochastic interest rates? American-style options are best priced via trees and finite differences methods, while path-dependent options via Monte Carlo Simulation.

The solution to the problem described above is the Least-Squares Monte Carlo Simulation. The algorithm was introduced in 2001 by Longstaff & Schwartz and is used to price American-style options via Monte Carlo Simulation. More specific, a least-squares analysis is used to determine the best-fit relationship between the value of continuing and the values of relevant variables at each time the holder of the option has

7 Supported by the Law of Large Numbers.

to decide whether to execute his/her right to buy/sell or not. Because of it dimensional flexibility it can be widely applied to a large number of complex and general options.

This dissertation is an effort to apply the method that Longstaff & Schwartz proposed in 2001 for valuing American-style options using MatLab programming. Two algorithms, containing one stochastic factor and two stochastic factors respectively, will be developed in MatLab, modeling this very popular issue.

Starting with a historical flashback, the evolution, from general Monte Carlo approaches to the most modern theories based on the model of Longstaff & Schwartz, will be shown.

3. LITERATURE REVIEW

In 1987 G. Barone-Adesi & R. Whaley were dealing with the valuation of exchange-traded American-style derivatives, calls and puts, written on commodities and commodity future contracts. So far, little research had been done on pricing options with early exercise features embedded in their price. Notwithstanding that methods such as binomial trees, finite differences and compound-option approximations were yielding accurate results, the fact that they were very time consuming and expensive to use, led to the need for an alternative.

Using the same assumptions as the ones of Black, Scholes and Merton (1973) and taking in consideration the cost of carry[8], they developed the "quadratic" approximation, which is applicable and useful in pricing futures options and stock options as well as options on foreign currencies, on stock indexes with continuous dividend yields, on precious metals and on long-term debt instruments with continuous coupon yields. After examining previous models, numerical, analytical or heuristic, they conclude that their algorithm avoids all the problems charged on its ancestors. For example, the approximation method of Geske and Johnson (1984) requires the evaluation of cumulative bivariate, trivariate and even higher order multivariate normal density functions, which needs quite an equipment to be applied and sometimes is, inevitably, infeasible. In addition, Johnson's (1983) technique is not applicable in general commodity pricing approximations.

The next part of their work is the comparison of their technique to the ones mentioned earlier separately for each kind of instrument being valued. The results are being consisted with the results of accurate methods, such as finite differences.

The main conclusion of their project refers to the categorization of the best method for option pricing depending on the time till expiration. For commodity options with less than one year to expiration, quadratic approximation best fits, while times to expiration beyond one year, concerning pricing accuracy, binomial trees or finite differences should be used.

[8] Cost of insurance, storage, deterioration.

J. Tilley in 1993 was the first researcher to publish a solution to the problem of valuing an American-style option taking advantage of Monte Carlo simulation. As the first broadly accepted serious attempt to price American options via simulation, he demonstrates the use of his algorithm by using an American option on an underlying instrument or asset for which the arbitrage-free probability distribution of paths through time can be simulated.

Having in mind, firstly, that people or organizations in the field, like institutions, financial intermediaries, brokers or dealers, need a single tool for valuation and breakdown and, secondly, the fact that technological evolution of his time had made powerful computers, servers and processors available for retail use, he demonstrated the use of an algorithm based on simulation.

The procedure followed requires the simulation of a finite number of paths and the computation of the option price from that sample. There is an "exercise-or-hold" indicator variable that takes the value "0" if the owner keeps the option for another time period and the value "1" if the owner exercises it at period t. The first step of the calculation is to compute the present value of the asset's cash flow along each path and then average across all of them. The cash flows on nodes where the option was not exercised are 0, while at all the other nodes its equal to its intrinsic value.

The whole discussion regards the estimation of the indicator mentioned above. Its computation mimics the backward induction of the familiar and widely used, now and then, binomial tree method. At each time step, a decision has to be made: whether to exercise or hold the option. The comparison that drives this decision chooses the maximum between the "holding value", which is just the intrinsic value, and the "exercise value" which is the present value of the expected one period ahead value of the option.

The source of the bias arising from the method of Tilley is the fact that the finite number of samples that he has taken into consideration cannot reach "perfect optimization", providing no proof of convergence. In addition, the paper does not deal with all the difficulties, at the optimal exercise-or-not decision, that would arise in a multidimensional problem, but just indicates that a minor transformation of the algorithm could give a solution to them.

Another group of scientists that realized the need for a method to price securities with early exercise opportunities is that of M. Broadie & P. Glasserman (1997). They designed a valuation algorithm, which can be applied to models with more than one state variables with, sometimes, path-dependencies. In realistic world, besides, models require at least three state variables.

Their algorithm reaches a specific price for the security being valued via the mean of an upper and a lower bound. It creates, in other words, a confidence interval for the option price under question, by combining the two biased estimators of the highest and the lowest value of the option. Both biased estimators are asymptotically unbiased as the size of the sample increases.

The application of the algorithm in a call option on a single asset with one state variable has shown that because of the bias of the estimators the intervals end up being conservative. The true value of the option was contained in the interval more times than the interval suggested.

On the other hand, tests on higher dimension problems brought conclusions regarding another field. They clearly demonstrate the need for variance reduction techniques, although the method is very promising for pricing American-style securities with multiple state variables.

The main problem of their technique is the limitation regarding the exercise opportunities. As the algorithm was designed, it is able to provide results only for a finite number of exercise dates, while continuous exercise is a feature available in a pretty large number of options. Even worse, the computational cost increases exponentially as the number of exercise opportunities increases.

The case they basically proved with their work is that there can’t be an unbiased estimator for the value of the right to early exercise or not, but a lot of extensions are to be done.

2001 was the year that Longstaff and Schwartz introduced their algorithm to the public, giving a solution to the problem, easy to apply and with real results. Supporting simulation as the ideal method for dealing with problems of that kind, they cite its advantages early in their paper. Simulation is applicable for problems of higher

dimensions, with multiple factors, path-dependent and/or American-style options with state variables that follow general stochastic processes and allow parallel computing.

Their algorithm is based on the least-squares regression framework using cross-sectional information and operates as follows:

The path the stock price follows is presented just like that of a binomial tree and is used to construct the tree for the payoff of the option. At each node, a decision has to be made: whether to execute the option or not. At this point the innovative technique of the authors takes place. The continuation value should be compared with the payoff of the option at the specific moment. The conditional expectation function is approached by regressing the ex post realized payoffs from continuation on functions of the values of the state variables only for the in-the-money- paths of the price of the underlying asset. Given the payoff, the continuation value is computed via the conditional expectation function and the last step is just to compare these two and eventually estimate the optimal stopping rule. The procedure is repeated recursively from the end to the beginning of the tree, going practically back in time, and finally the price is obtained by discounting the obtained cash flows to time zero.

Longstaff & Schwartz use a lot of examples to illustrate their approach. Firstly, they calculate the price of a simple American put option, just in order to prove that the algorithm is easily implemented and only needs a simple regression. Their second example refers to an exotic-American-Bermudan option and they show that the results they obtain are very similar to those of the finite differences method, with differences typically less than two or three cents per 100$ notional value, both positive and negative. Thirdly, they price a cancelable index amortizing swap and their conclusion fully coincides with the previous one. Then they use an American option on an asset that follows a jump-diffusion process and, lastly, a deferred American swaption in a 20-factor string model where each point on the interest rate curve is a separate factor.

The numerical and implementation issues they discuss at the end of their paper were the food for thought for the generations that followed. Their algorithm could be extended for higher dimensional problems or for various least squares methods of regression depending on the special features of the instrument being valued. In addition, the choice of the basis function for the state variables is very important and is open to

a lot of experimentations. They, lastly, propose that the computational speed is a sector that could be further examined, mostly by combining different CPUs for extra speed.

Alongside with Longstaff & Schwartz, in July of 2001, J. Tsitsiklis and B. Van Roy published another paper dealing with pricing complex American-style options.

The "curse of dimensionality" is being touched on in this paper, that is the fact that the size of the state space grows exponentially in the number of variables involved. Multiple sources of uncertainty contradict with the parsimonious schemes that are needed in order to solve a problem such as the pricing of an American-style derivative. As a result, the computational requirements of such a problem may become prohibitive.

Concerning about real world financial contracts, they work on finite horizon problems. Value functions are estimated for each time period and states are mapped based on those functions giving the future payoffs. These payoffs are then compared with the payoffs of instant exercising. The maximum of those two is being chosen and via the popular method of backward induction, the value of the option is being computed.

More specific, the function of approximation is produced based on hand-crafted features and, most of the times, on whatever human experience or intelligence is available. The features are linearly, which is used in the paper discussed, or not combined. Basis functions are generalized over both state space and time. Such method can have a large approximation error, but that's a problem that could be solved by simulating the state process by using the underlying risk-neutral probability distribution, where the approximation error remains within some specific boundaries.

They conclude that their paper provides the theoretical support for the effectiveness of this particular support of state sampling.

The year that followed, E. Clement, D. Lamberton and P. Protter worked on the two aforementioned papers that were introduced in 2001. Their approach distinguishes two types of approximation based on the method of Least Squares Monte Carlo simulation. Firstly, they project on a finite set of basis functions and replace analogously the conditional expectations in the dynamic programming and, secondly, the compute the value function of the first approximation using Monte Carlo simulation and least squares regression. What they do, basically, is choose the basis function and run the Monte Carlo procedure.

Through an extremely large number of mathematical proofs, they prove that as the number of functions goes to infinity, the value function of approximation converges with the value function of the initial stopping problem. On the other hand, for a finite set of basis functions, the value function of the first approximation converges to the value function of the second approximation. Their work is completed by a type of central limit theorem they develop for the rate of convergence of the Monte Carlo procedure, that is the second approximation, proving that the error is asymptotically normalized.

In order to connect their method with those of Longstaff & Schwartz and Tsitsiklis & Van Roy, they argue that their base was, of course, the algorithm of the first two authors, but it is applicable to that of the other two authors in order to analyze the rate of convergence of their algorithm.

In the same year, 2002, L. C. G. Rogers, from university of Cambridge, examined another side of the optimal exercise strategy in pricing American-style options based on a dual characterization of the problem. The papers introduced so far compute an approximate value for the price of the option, that converges as the number of estimated paths increases and which operates as the lower bound for the actual value of it. This paper, on the other hand, without trying to estimate an optimal exercise strategy, comes up with an estimation for the upper bound of the price of the option and thus provides a method that is very useful and profitable as a hedging instrument.

The two methods, the one that produces the lower bound and the one that produces the upper bound, are to be used by different parties of the option transaction. The writer needs to know the upper bound of the price, while the buyer the lower bound of the it.

The upper bound is then used for the optimal choice of the Langrangian martingale. The price that is obtained by the method is expressed as the infimum of a family of expectations, the infimum being taken over the class of Langrangian martingales.

The authors take advantage of four examples, an American put on a single asset, American min puts on n assets, Bermudan max calls on n assets and an American-Bermudan-Asian option, in order to take this one step further. The results of the examples are in fact in 1-2% divergence from the other numerical methods. Errors of this order are to be expected, regarding the estimates of volatilities or the assumption of constant interest rates, and are the main sector for improvement for the method.

Another sector to be improved is the large MAD figures that result from this hedging policy, making it a bit of misnomer.

Concerning a different field of studying the Monte Carlo approaches in American-style options valuation, M. Moreno & J. Navias (2003) examine the robustness of the Least Squares Monte Carlo algorithm of Longstaff & Schwartz relatively to the type and number of basis functions chosen.

The authors apply the LSM algorithm to price and compare the results obtained for an American put option, a Bermuda call option on the maximum of five assets and an American-Bermudan-Asian option, just like Longstaff and Schwartz did in 2001. They, also, compute in- and out-of-sample option prices as well as the standard errors for different number and types of the basis functions.

Among the basis functions they, in the first place, refer to are Power, Legendre, Laguerre, Hermite A & B, Chebyshev 1st kind A, B & C and 2nd kind A & B Polynomials. Linear regression is the projection of the dependent variable on the function produced by the independent ones. Thus, overlapping functions are excluded. In addition, they prove that the coefficients of each of the polynomial family chosen form a non-singular matrix with respect to the power function. Consequently, the result obtained from any of the basis functions should be identical to the others, since they produce the same span.

Practically, for a fixed number of terms, the results they obtain have small differences due to numerical errors mostly in the least-squares routine, not the LSM method. In general, the LSM algorithm slightly underprices the option. For reasonable degree of the polynomial the outcomes of the algorithm are quite robust, increasing though the degree more than 20, can cause problems to the regression. Regarding the type of the basis function it's clear for an American put option but the choice becomes more complex as the option becomes more complex and the authors find that the final result for a complex option may have small differences among different basis functions chosen.

The importance of the LSM method that Longstaff & Schwartz proposed in 2001 is established once again by the research of L. Stentoft in 2004. He makes a thorough examination of the study and suggests a numerically simpler specification for the cross-sectional regression than the one used in the paper of Longstaff & Schwartz. In addition,

he emphasizes in the trade-off between computational time and precision and gives details on how to handle multiple stochastic factors, by introducing simulation as the solution to the 'curse of dimensionality'.

The purpose of this paper is mainly to give answer to three questions arising from the paper of 2001: examine the results of using different numbers of regressors and paths, try alternative specifications for the cross-sectional analysis and put attention in the trade-off between computational time and precision.

Regarding the first goal, he finds that the conditional expectation function can be well approximated as the number of regressors and paths tend to infinity. More specific, he finds that, for out-of-the-money options, the effect of increasing the number of regressors is much more important than that of the paths. Though, the convergence is not guaranteed when the number of regressors is increased irrespectively of the number of paths used.

Subsequently, he criticizes the choice of the Laguerre polynomial family, that Longstaff & Schwartz chose, taking position in favor of the general Chebyshev family and the shifted Legendre family.

He, also, emphasizes on the computational time, which is a real-world problem and as that it should be dealed with. He uses the RSME as a measure for precision and states that when choosing the specification of the cross-sectional analysis, one should balance the precision obtained with the time it took to be calculated, in order to specify not only the most accurate way of solving the problem, but also the most efficient.

The author closes his examination by analyzing how the algorithm treats multiple stochastic factors and concludes that it should be preferred to the Binomial Model for high dimensional problems. He demonstrates his analysis through an example generalizing it to as many stochastic factors as one wants.

One of the most modern papers, the one of N. Areal, A. Rodigues & M. Armada in 2008, proposes improvements in the Least Squares Monte Carlo approach. They state that simulation is the most flexible of all the pricing American-style options methods, because it is adaptable to different stochastic processes, multiple underlying assets, path dependence, exotic options, when discrete dividends are considered or with multi-state variable options. One of the main drawbacks of the method is the advanced

computational requirement it takes, the low speed and the difficulty in dealing with free-boundary problems, all problems, of course, that are manageable with the high-tech computers that were, by then, and are, now, being developed.

In their paper, different regression algorithms are being tested, such as varying polynomial families as the basis function, in order to suggest a way for the reduction of the execution time of the algorithm to work. A couple of variance reduction techniques are being used and an extended algorithm regarding compound and mutually exclusive option is being proposed[9].

Two of the regression methods that are mentioned in the paper is the LFIT and the SVD. Regarding the first one, an alternative form of it is being suggested here, designated as the Continuation value by Conditional Estimation. The comparison that has to be done at each step contains the continuation value given by the least-squares regression and the present value of continuation estimated using the previous time step regression coefficients. All the results derived from that method are being evaluated by a benchmark, which is usually the price given by the binomial model. The SVD, on the other hand, is a singular value decomposition algorithm.

The algorithm then is open to some improvements, the variance reduction being the most useful. Various techniques exist, such as, antithetic variable, control variate, quasi-random numbers and the method of moments. The authors describe the control variate technique and the method of moments. In this way, the algorithm allows a faster regression with no significant loss of accuracy.

The tests they conduct refer to vanilla options and portfolios of options, and show that when time is of importance the simple Powers polynomial family should be chosen. When accuracy is of importance, the basis function should be chosen based on the specific characteristics of the option valued and the number of paths should be increased.

The final paper that is of relevance to comment here is that of M. Cerrato in 2009. Accepting the fact that simulation is a pretty active research area in recent years, he, in

[9] The original algorithm was first proposed in Andrea Gamba, December 2003, "Real Options: A Monte Carlo Approach" and a more extensive analysis regarding this one ore the one presented in the paper of N. Areal, A. Rodigues & M. Armada is out of the scope of this dissertation.

turn, proves that least squares estimators underestimates the price of the option and explores varying variance reduction techniques. He also introduces extensions to two older algorithms; that of pricing American options of Longstaff & Schwartz now covers the case of stochastic volatility, while that of Glasserman and Yu[10] embodies innovative techniques in pricing Asian and basket options.

The implementation of variance reduction techniques in the paper is extensive and proves that one can reduce the probability of choosing sub-optimal exercise decisions and, therefore, reduce the option price bias. Their method consists of a martingale approach and provides just a guidance to the traders, not the optimal methodology to be used, as long as it generates standard errors and root mean squared errors that are of the same order of magnitude.

The novel approach they introduce for pricing American-style options is quite precise and efficient compared to existing methods, while, regarding general applications, he concludes that a desirable level of accuracy could be obtained using three basis functions, 100,000 replications and the control variate technique.

[10] See Glasserman P. & B. Yu, 2004b, "Simulation for American Options: Regression Now or Regression Later? »

4. METHODOLOGY

4.1 Notation

t: time period

T: horizon of the option

K: strike price

S_0: the initial price of the underlying asset

S_t: the price of the underlying asset at time t

r_f: the risk-free interest rate

r_0: the initial price of the interest rate

r_t: the interest rate at time t

$sigma_1$: the volatility of the underlying asset

$sigma_2$: the volatility of the interest rate

d: the dividend yield of the underlying asset

m: the max-exponent of the basis function

n: the number of periods

M: number of paths[11]

a_1: speed of reversion of the underlying asset's price

b_1: mean reversion coefficient of the underlying asset's price

a_2: speed of reversion of the interest rate

b_2: mean reversion coefficient of the interest rate

c: the value of the call

[11] Concerning the trade-off between accuracy and execution time, M should be chosen carefully. The faster one gets the results, the faster he/she trades, competes and achieves goals, thus a part of accuracy could be sacrificed for the state of speed.

p: the value of the put

aBm: Arithmetic Brownian Motion

gBm: Geometric Brownian Motion

ou: Ornstein Uhlenberg

sp: Simple Powers

le: Legendre polynomial family

la: Laguerre polynomial family

he: Hermite polynomial family

4.2 Stochastic processes

Stochastic process is the process that any variable with a value changing over time follows. Generally in financial engineering and more specific in option pricing, stochastic processes are the key to simulate the path that the underlying asset follows. They are classified, firstly, into continuous time and discrete time and, secondly, into continuous variable and discrete variable. A continuous time stochastic process is one where the value of the variable can change at any time throughout the time interval, while at a discrete time stochastic process the value of the variable can change only at specific and certain points of time. On the other hand, a continuous variable stochastic process is one where the variable can take any value within a certain range, while at a discrete variable stochastic process the variable only takes specific and predetermined values.

In practice, continuous time and continuous variable stochastic processes are not observable. Trading is available only when exchanges are open and the stock prices are restricted to fixed values. That's why, in literature, the processes are considered continuous time and discrete variable, but, in practice, it is sometimes convenient to consider the discrete time version too along with the discrete variable.

To be more specific and precise, regarding the topic of this thesis, when one has to deal with an optimal stopping problem, an allowance should and is done. The option is

treated as a Bermudan one. That is, it is supposed to be executable at specific moments during its lifetime. The effect of this modification is that the value of the option is often diminished. Though, the difference between theoretical and practical value is small and vanishes as the number of allowable exercise times tends to infinity

i. (Arithmetic) Brownian Motion

A standard one-dimensional Brownian motion on [0,T] is a stochastic process $\{W_t, 0 \leq t \leq T\}$ that is a particular type of Markov stochastic process and has the following properties:

- $W_0=0$
- the mapping $t \rightarrow T$ is, with probability 1, a continuous function on [0,T]
- the increments $\{W_0-W_1, W_1-W_2, \ldots, W_{T-1}-W_T\}$ are independent
- $W_t \sim N(0,1)$

For constants μ, $\sigma>0$, a process X_t is called a Brownian Motion with drift μ and diffusion coefficient σ^2 ($X_t \sim BM(\mu, \sigma^2)$) if

$$\frac{X_t - \mu t}{\sigma}$$

is a standard Brownian Motion.

Thus, X may be constructed from a standard Brownian Motion W by setting:

$$X_t = \mu t + \sigma W_t$$

So, $X_t \sim N(\mu t, \sigma^2 t)$.

In addition, X solves the stochastic differential equation:

$$dX_t = \mu dt + \sigma dW_t$$

For deterministic but time-varying μ_t and σ_t the following Brownian Motion is defined through the SDE:

$$dX_t = \mu_t dt + \sigma_t dW_t$$

Now, for a standard Brownian Motion set $t_0=0$ and $W_0=0$ and let $Z_1, \ldots, Z_n$ be independent, standard normal random variables. Subsequent values are generated as follows:

$$W_{t_{i+1}} = W_{t_i} + \sqrt{t_{i+1} - t_i} Z_{i+1}$$

For $X \sim BM(\mu, \sigma^2)$ with constant μ, σ and given X_0 set:

$$X_{t_{i+1}} = X_{t_i} + \mu(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i} Z_{i+1}$$

(1)

ii. Geometric Brownian Motion

A stochastic process X_t is a Geometric Brownian Motion if $\log X_t$ is a Brownian Motion with initial value $\log X_0$. Thus, all methods for simulating Brownian motion are applied to Geometric Brownian Motion via exponentiation. It's the most fundamental model for the value of a financial asset because of its centrality. One of its basic features is that it cannot take negative values due to the exponential function that is always positive. That's very important when simulating stock prices or any other limited liability asset that are by definition positive.

Suppose W is a standard Brownian Motion and X satisfies:

$$dX_t = \mu_t dt + \sigma_t dW_t,$$

so that $X \sim BM(\mu, \sigma^2)$.

If $S \sim GBM(\mu, \sigma^2)$ and if S has initial value S_0, taking as granted that the increments of W are independent and normally distributed, this provides a recursive procedure for simulating values of S at $0=t_0<t_1<t_2<\ldots<T$:

$$S_{t_{i+1}} = S_{t_i} e^{(\mu - \frac{\sigma^2}{2})(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i} Z_{i+1}}, \quad i=0, 1, 2, \ldots, n$$

(2)

With $Z_1, Z_2, \ldots, Z_n$ independent standard normal.

iii. Ornstein Uhlenberg

The classical model of Vasicek[12] gives the following equation

$$dr_t = a(b - r_t)dt + \sigma dW_t$$

where W is a standard Brownian Motion and a, b, σ are positive constants. One may notice that the sign of the drift $b-r_t$ depends on the relation between b and r_t. If $b>r_t$, it is positive and vice versa. Thus, r_t is pulled toward level b, which is generally stated as mean reversion, that is the long-run interest rate level, while a is the speed at which r_t is pulled toward b.

To simulate r at times $0=t_1<t_2<\ldots<T$ and in the special case that bt≡b the following analytic approximation is used:

$$r_{t_{i+1}} = e^{-a(t_{i+1}-t_i)}r_{t_i} + b\left(1 - e^{-a(t_{i+1}-t_i)}\right) + \sigma\sqrt{\frac{1}{2a}(1 - e^{-2a(t_{i+1}-t_i)})}Z_{i+1}$$

(3)

Where $Z_1, \ldots, Z_n$ are independent draws from N(0, 1).

4.3 Polynomial Families

A polynomial family, in mathematics, is an orthogonal polynomial sequence, such that any two different polynomials in the sequence are orthogonal to each other under some inner product.

Among the most common and widely used polynomial families that are used not only in mathematics, but in other scientific fields as well, are the (weighted or generalized) Laguerre, the Hermite, the Legendre, the Chebyshev, the Gegenbauer and the Jacobi polynomial families.

In this dissertation, the following sequences are used:

[12] See O. A. Vasicek, 1977, "An equilibrium characterization of the term structure", Journal of Financial Economics, 5:177-188

i. Legendre polynomial family

Considering P(m,x) as the m^{th} degree Legendre polynomial of x, the following recursion formula gives the analytical solution of the Legendre sequence:

$$P(m,x) = \frac{2m-1}{m} xP(m-1,x) - \frac{m-1}{m} P(m-2,x)$$

while P(0,x)=1 and P(1,x)=x.

They are orthogonal on the interval [-1,1].

ii. Laguerre polynomial family

Considering L(m,x) as the m^{th} degree Laguerre polynomial of x, the following recursion formula gives the analytical solution of the Laguerre sequence:

$$L(m,x) = \frac{(2m-1-x)L(m-1,x) - (m-1)L(m-2,x)}{m}$$

while L(0,x)=1 and L(1,x)=1-x.

iii. Hermite polynomial family

Considering H(m,x) as the m^{th} degree Hermite polynomial of x, the following recursion formula gives the analytical solution of the Hermite sequence:

$$H(m,x) = 2xH(m-1,x) - 2(m-1)H(m-2,x)$$

while H(0,x)=1 and P(1,x)=2x.

They are orthogonal on the real line.

Symbolizing as $F(\omega;t_k)$ the value of the continuation and A(m,x) the m^{th} degree polynomial of the A selected polynomial family at x the equation specified from the liner regression should be of the following form:

$$F(\omega; t_k) = \sum_{j=0}^{m} a_j \, L(j, x)$$

where α_j coefficients are constants.

4.4 Risk neutral valuation

In a world of risk-neutral investors, the choice between two similar investments yielding the same expected rate of return and only differ in the risk of action taken is uniform, insomuch everybody would pick the least risky one. That is, no investor would ask for a higher rate of return for holding risky assets. Although pretty logical, this principle creates obstacles regarding the fact that expectations of the rate of return are unobservable. As a result, there is no proper discounting factor that will accurately depict the risk profile both the investor and the investment.

The solution to the aforementioned problem is called risk-neutral valuation and operates as follows:

a. Consider the risk-free interest rate as the expected return for all assets.
b. Value each payoff by discounting each expected value at the risk-free interest rate.

The only condition is no arbitrage in the market.

With these data in mind, the drift term μ equals the risk-free interest rate.

4.5 Random numbers

The kernel of all Monte Carlo simulation procedures is the generation of one or more series of random numbers. Modern random number generators, though, produce numbers that only mimic the randomness and are not genuinely random. That's

explained by the fact that all those random number generator algorithms are deterministic and as a result the numbers they produce are deterministic as well. For the shake of each procedure one follows and because the imitation is very efficient, the produced numbers are treated as genuinely random, allowing he/she to apply tools from probability and statistics to analyze them.

A generator of random numbers produces a sequence of random numbers $Z_1, Z_2, \ldots, Z_n$ with the properties:

- each $Z_1, Z_2, \ldots, Z_n$ is uniformly distributed between 0 and 1
- the Z_i are mutually independent

The first property can be extended, as any sample randomly produced from the unit interval can be transformed into sample from any other distributions using the appropriate techniques and subsequently conduct stochastic simulations. Uniformity is the case that the fraction of the values falling in any subinterval of the unit interval should be approximately the length of the interval. The second property implies that all pairs of numbers are uncorrelated with each other and unpredictable given previous values, meaning there is no distinguishable pattern among the values.

For the construction of a random number generator, well, the following must be considered: the period length, the reproducibility, the speed, the portability and the randomness. Generally, generators that produce as many as possible distinct values before repeating, that are easy to produce twice or more times the same series of random numbers, that are fast, that work properly on any computing platform and that are tested theoretically and statistically for their randomness, are the ones preferred.

4.6 Variance reduction techniques

Supposing the simulation is carried out as described so far, that is via random numbers, a significant large amount of trials should be done to estimate a quite accurate price for the option being priced. The random outputs create a variance that is of great importance. This is, of course, very expensive in terms of computational time, due to the square root convergence of the algorithm, so that's why several variance reduction techniques exist, which operate without disturbing the expectation.

i. Antithetic Variable

In this technique, two values for the derivative is being calculated. The first one is calculated in the usual way, while the second one is calculated using the opposite sign of all the samples from standard normal distributions. The final price of the option is the average of those two. The success of this method relies on the fact that if the value from the first computation is, say, above the true price, the corresponding value from the second computation will be below the true price and vice versa. The final result, consequently, will balance around the true price.

ii. Control Variate

This method premises the existence and use of a similar derivative to the one being valued, for which an analytic solution is available. By computing the difference between the price obtained from the analytical solution and the price obtained by the Monte Carlo simulation, one knows the difference between the value for the derivative being priced obtained from the simulation and the real one. For the correct execution of the method, one should use the same random number stream for the pricing of both derivatives.

iii. Importance Sampling

In order to avoid computational mistakes or misleads because of, for example deep-out-of-the-money options, this method estimates the value based only on the paths that are in-the-money, excluding the out-of-the-money or at-the-money ones.

iv. Stratified Sampling

This technique functions as follows: all the samples taken from the appropriate probability distribution are then divided into a quite large number, for example

1,000, of equal smaller intervals and a representative value of each interval is being chosen[13]. The option is, subsequently, valued under these representative values.

v. Moment Matching

The method of "Moment Matching" involves adjusting the samples taken from the appropriate distribution so that the first, second and, possibly, higher moments are matched. As a method, it has the disadvantage of redundant memory usage. That's why, it is quite often to be used combined with the "Antithetic Variable" technique, which by definition matches all odd moments. What "Moment Matching" technique further needs to do, is just calculate the second and, possible, the fourth moment.

vi. Using Quasi-Random Sequences

It is quite similar to the "Stratified Sampling" except to the fact that the samples are taken in such a way in order to always fill in gaps between existing samples and not randomly. Thus, at each stage of the simulation, the sampled intervals are almost evenly divided throughout the probability space.

The method used here is the first one, the "Antithetic Variable", and further explanation will be given later in the dissertation.

4.7 The Least Squares Monte Carlo Algorithm of Longstaff & Schwartz

The Least Squares Monte Carlo method for pricing American-style options that Longstaff & Schwartz proposed in 2001 is the objective of the present dissertation. It is an algorithm that uses backward induction for the estimation of the option price, while the optimal exercise strategy is obtained via the combination of the Monte Carlo simulation with the least squares regression. Its main purpose is to provide a pathwise

[13] Typically, the mean or the median.

approximation of the optimal stopping rule that maximizes the value of the American option.

More specific, the desired number of sample paths is simulated in the beginning. Every option is considered as a Bermudan-style one in this approach. In other words, the time steps are assumed to be discrete allowing early exercise. In practice, however, there are a lot of options that are continuously exercisable, but that's a case that can be covered by the LSM algorithm taking the number of exercise dates to be sufficiently large.

At each time step, a least squares regression takes place according to the no-arbitrage valuation theory. That is, the discounted optimal payoffs from continuation of the next time step are regressed on the desired set of the basis functions chosen towards underlying asset prices. The key here is that Longstaff and Schwartz allow in the regression only the paths of the underlying asset price that are in-the-money and that the values obtained are used only for comparison. In other words, paths that do not generate any cash flows are neglected from the regression.

At the final date, the holder of the option exercises it if it's in-the-money or allows it to expire if it's out-of-the-money. At every other time step, a comparison is done between the fitted value from the regression and the expected payoff from immediate exercise. If the fitted value from the regression is larger than the payoff from immediate exercise, then the optimal strategy is to hold the option alive from at least one more time step. In the opposite case, even if the two values are equal, the holder of the option should immediately exercise it in order to benefit from the instant payoff.

After completing all the steps described above, the lattice of the optimal exercise strategy and that of the optimal cash flows at each time step will have been constructed. Each optimal payoff is, then, discounted back to time zero and, finally, the average of all payoffs produces the value of the option.

The algorithm is easily transformed and adjusted to different data and cases, such as exotic options, multidimensional American options or American options with jump diffusions.

4.8 MatLab Implementation of LSM Algorithm

MatLab, which stands for MATrix LABoratory, is a multi-paradigm numerical computing environment and a fourth-generation programming language. As a high-performance language, it integrates computation, visualization and programming environment for technical computing. It covers a wide variety of scientific fields, such as engineering, science, economics, statistics, econometrics, finance, through special packages called toolboxes. MatLab allows matrix manipulations, plotting of functions and data, implementation of algorithms and creation of user interfaces in combination with other programming languages.

Throughout this dissertation MatLab version R2015b has been used. No package in MatLab exists for the application of the Least Squares Monte Carlo algorithm. One can find the complete MatLab code for the implementation, that will be described from here on, in the Appendix[14].

i. Model with one stochastic variable

The first case of the implementation of the Least Squares Monte Carlo algorithm in MatLab environment developed, contains the creation of an algorithm that prices an American-style option, either a call or a put, under the assumption that the underlying asset's price follows a stochastic process, while all the other variables remain stable. The development took place in two stages: firstly, a MatLab function that simulates the underlying stock's price path was developed and, secondly, a MatLab function that prices the option completed the valuation.

More specific, the function that simulates the path that the underlying asset price follows is called *path*. The inputs of this function are S_0, r_f, d, $sigma_1$, T, n, M, a_1, b_1 and the flag. Depending on the stated flag, the function demonstrates the simulation of the requested path either as an Arithmetic Brownian Motion, a Geometric Brownian Motion or an Ornstein Uhlenberg process. Consequently, the output of the function is a M*n matrix S_t, with the features the user set as inputs.

[14] See page 61.

Equation (1) is the analytical form of an Arithmetic Brownian Motion in a discrete time environment. Though, one of the toolboxes of MatLab Software is the Financial Toolbox, which supports plenty of SDE models and the Monte Carlo Simulation of them. Thus, the Arithmetic Brownian Motion path was constructed automatically by the *bm* constructor of MatLab. Using, r_f-d as the drift parameter and $sigma_1$ as the scale parameter under the risk-neutral valuation framework, a vector-valued Arithmetic Brownian motion object was constructed, with start state S_0. The MatLab *simulate* function was then used in order to simulate the M sample paths requested from the user. At this point, it was found useful to induce negative dependence between paired input samples, meaning use one of the, previous discussed, control variance techniques. The most common of them, and the one used here, is the Antithetic Variable one. Lastly, by using a *for* loop, the complete path St was constructed, that is, all the M*n cells of it were filled with the simulated values of the underlying stock price.

The method described above is exactly the same with the one used in order to simulate the Geometric Brownian Motion path. As stated before, the Geometric Brownian Motion is an exponentiated Arithmetic Brownian Motion. Thus, the Financial Toolbox of MatLab contains a predetermined function to simulate this motion as well. Regardless the fact that equation (2) gives the analytical form of a Geometric Brownian Motion in a discrete time environment, it was found preferable to take advantage of the automatic way, that the MatLab Software offers to carry out the desired simulation. The *gbm* constructor was used in the first place and, subsequently, the simulation was developed again by the *simulate* function, using the same inputs and the same control variance technique.

The simulation of the Ornstein Uhlenberg process differs a little. There is no built-in function in MatLab to demonstrate it. So, equation (3) was used in the implementation here. The start state S_0 is the same input as the one used in the previous simulations, while now there is need for the specification of two more variables, a_1 and b_1, which are the constants that particularize each Ornstein-Uhlenberg process. a_1 symbolizes the speed at which the value being simulated is reverting around the mean b_1. So, by the

use of a *for* loop and after producing the needed normally distributed random numbers, the simulated path S_t is ready for further use by the next function developed here.

Next, theoretically, a function that uses as an input the path S_t, that was previously produced, would be needed in order to approximate the value of the given option. The function developed here is named *price* and uses as input, apart from the path S_t, K, T, n, r_f, M, m, the basis and the flag. Regarding the basis, the user needs to choose between the simple powers, the Legendre polynomial family, as well as the Laguerre and the Hermite polynomial families, as the basis function for the least squares regression that will take place. Regarding the flag, the type of the option being priced is specified here, that is, whether it is a call or a put.

The necessary matrixes OptExe and OptCF are firstly defined, determining the optimal exercise strategy and the cash flows that the holder will receive following the optimal exercise strategy, respectively, by the completion of the algorithm. In the optimal exercise strategy matrix '1' means that the option is optimal to be exercised and '0'that the holder should wait at least one more period before exercising.

The next part of the code contains a *for* loop, as it is useful to work on each time period separately and successively, starting from the end. So, all the work done from here on refers to the specific column of the time period, that the step of the loop defines. Though, that means, that, since the algorithm is recursive, several intermediate matrices will be constructed during the execution of the loop.

The optimal cash flow matrix is, in the beginning, constructed by each cell taking the value of the maximum cash flow between $K-S_t$ or S_t-K, depending on the type of the option, and 0. For all the cells that their cash flow is different from '0', the corresponding cells of the optimal exercise strategy matrix take the value '1'. In fact, the exercise strategy and the cash flows of the last time period are identical to the ones that the holder would have if the option was European-style rather than American-style. Another matrix called DisCF is then constructed containing the discounted cash flows of the corresponding cells of the optimal cash flow matrix.

Next step is the least squares regression. Firstly, all the in-the-money paths of the optimal cash flow matrix are determined, because only in the case that the option is in-the-money at a time period, the holder needs to decide whether to execute it or not. As well, the choice of, only, the in-the-money paths limits the region over which the conditional expectation function should be computed, resulting in much fewer basis functions needed to obtain an accurate price. The basis the user has chosen defines the procedure that is then followed. Of a programming point of view, the simplest case is that of the simple powers as the basis of the regression. The MatLab *polyfit* function automatically produces the coefficients, in descending powers, of the polynomial of degree m along with a constant that is the best fit, in a least squares sense, of the in-the-money paths of time T on the discounted corresponding paths of time T+1.

Apart from the simple powers as the basis of the regression, the user may also choose among the Legendre, the Laguerre or the Hermite polynomial family as basis. The implementation is common between all three families. The built-in functions of MatLab, *LegendreP*, *LaguerreL* and *HermiteH*, are used to estimate the m^{th} degree Legendre, Laguerre and Hermite polynomial, respectively, for each stock price value in matrix S_t for the in-the-money paths, of course. Though, for the regression, it is necessary to estimate all the previous degrees of each polynomial family, meaning from degree 1 until degree m. All the estimated polynomials of each family are then used as the independent variables of the regression along with a constant. The dependent variable is, once again, the corresponding discounted cash flows of the next time period. For the implementation of this procedure the MatLab *regress* function was used, which returns a m*1 matrix of the coefficient estimates for a multilinear regression of the discounted responses on the polynomial predictors.

Through some numerical and algorithmical manipulations, which are out of the scope of this dissertation, the value of continuation is, then, computed substituting each value of the stock price into the, estimated from the regression, conditional expectation function. That value is stored in the Cont matrix and should be compared with the corresponding intrinsic value $K-S_t$ or S_t-K, that is the immediate exercise value, depending on the type of the option, in order to determine whether the option should be

exercised or not. If the immediate exercise value is larger than the continuation value then the respective cell of the optimal cash flow matrix takes that value and the one of the optimal exercise strategy takes the value '1'. On the other hand, if the continuation value is larger than the immediate exercise value, both cells take the value '0'. That's the end of the *for* loop.

Another numerical manipulation takes place now, consisting of a *for* loop, once again. It satisfies the premise that the option can only be exercised once in its lifetime. As a result, the loop turns into '0' the value of all the cells, in the optimal cash flow and optimal exercise strategy matrix, after the first determined execution of the option, starting from time zero this time.

The OptCash matrix is then discounted cell by cell, resulting in the OptCashDis matrix, which is summed and averaged in order to calculate the final value of the either call or put.

ii. Model with two stochastic variables

The second case of the implementation of the Least Squares Monte Carlo algorithm developed, contains the creation of an algorithm that prices an American-style option, either a call or a put, under the assumption that both the underlying asset's price and the interest rate follow a stochastic process, while all the other variables remain stable. The development took place in two stages: firstly, a MatLab function that simulates the underlying stock's price path along with that of the interest rate was developed and, secondly, a MatLab function that prices the option completed the valuation.

The general context of the second implementation is the same as that of the first. Thus, the points that remain the same will not be decomposed thoroughly. Only the shift points will be discussed analytically.

The first function that was developed at this point is fully consistent with that of the first implementation. It is called *path2* and takes as input S_0, r_0, d, $sigma_1$, $sigma_2$, T, n, M, a_1, b_1, a_2, b_2 and the flag. Though, the outputs that it gives are, apart from the path of the stock price S_t, the path of the interest rate as well. The stock price is allowed, once again, to follow either an Arithmetic Brownian Motion, a Geometric Brownian Motion or an Ornstein Uhlenberg process, which is defined by the flag, while the interest rate follows an Ornstein Uhlenberg process.

The first point at which this application of the algorithm differentiates from the first one is the way the stochastic processes are simulated. The built-in functions of MatLab, such as *bm* and *gbm,* that were used before are now useless, because of the fact that having to deal with two stochastic variables means that one has to deal with their correlation as well. Their correlation is reflected at the random numbers used to simulate each path.

More specific, let's denote the following random numbers:

$$\varepsilon_t \sim N(0,1)$$
$$\hat{w}_t \sim N(0,1)$$

The size of the matrices with the random numbers produced is (M+1)*n.

After calculating the correlation ρ between those two families of random numbers, one more family w_t is produced following the equation:

$$w_t = \rho^2 \varepsilon_t + (1 - \rho^2)\hat{w}_t$$

The ε_t and the w_t random numbers are used to simulate the path of the stock price and the interest rate respectively. Depending on the flag that the user has set as an input, the

stock price may follow either an Arithmetic Brownian Motion with the following analytic approximation:

$$S_{t_{i+1}} = S_{t_i} + (r_{t_{i+1}} - d)(t_{i+1} - t_i) + sigma_1\sqrt{t_{i+1} - t_i}\varepsilon_{i+1}$$

or a Geometric Brownian Motion with the following analytic approximation:

$$S_{t_{i+1}} = S_{t_i} e^{(r_{t_{i+1}} - d - \frac{{sigma_1}^2}{2})(t_{i+1} - t_{i)} + sigma_1\sqrt{t_{i+1} - t_i}\varepsilon_{i+1}}$$

or an Ornstein Uhlenberg process with the following approximation:

$$S_{t_{i+1}} = e^{-a_1(t_{i+1} - t_i)}S_{t_i} + b_1\left(1 - e^{-a_1(t_{i+1} - t_i)}\right) + sigma_1\sqrt{\frac{1}{2a_1}(1 - e^{-2a_1(t_{i+1} - t_i)})}\varepsilon_{i+1}$$

While the interest rate follows an Ornstein Uhlenberg process with the following analytic approximation:

$$r_{t_{i+1}} = e^{-a_2(t_{i+1} - t_i)}r_{t_{i+1}} + b_2\left(1 - e^{-a_2(t_{i+1} - t_i)}\right) + sigma_2\sqrt{\frac{1}{2a_2}(1 - e^{-2a_2(t_{i+1} - t_i)})}w_{i+1}$$

Both approximations are based on a start state S_0 and r_0 which determines the whole first column of each of the tables S_t and r_t, that are finally exported from the function.

Having produced the simulated path for both the stock price and the interest rate, the second stage of the implementation includes, once again, the valuation of the option via a function called *price2*, that uses as input the following data: S_t, K, T, r_f, r_t, M, n, m, flag. The flag feature determines whether the option being valued is a call or a put.

The general logic and the specific steps followed are identical to those of the *price* function that was developed in the first implementation. Differentiation derives from the regression. First of all, as basis function, only the simple powers one has been developed at this point. The way, though, that the coefficients are produced, cannot be the same as previous, as with two state variables the set of basis function should also include cross-products of the univariate terms.

General principles say that the bases for m-variate functions are formed by the m-fold tensor product of univariate orthogonal polynomials. Though, that way, the number of the tensor products grows exponentially with the dimensionality of the problem. As Judd (1998) indicates the approximation given by the complete set of polynomials is as good as the one given by the m-fold tensor product with much fewer elements. Thus, following the work of A. Tsekrekos, M. Shackleton and R. Wojakowski (2012), instead of using the complete set of polynomials as regressors in the approximation of the continuation value function, consuming very large computing power, the tensor products were used. The condition set here in the choice of the specific tensor products is the sum of the exponents of the univariate terms to be smaller or equal to the max exponent of the basis function, m, inputted by the user. With that in mind, the regression was implemented for the cases of m being equal to 1, 2, 3 or 4.

The application from here on is exactly the same with the univariate model explained before.

5. RESULTS

i. Model with one stochastic variable

For an ***at-the-money option*** with S_0=100, K=100, r_f=0.06, d=0.05, sigma$_1$=0.02,T=10, M=10.000, n=120[15], a_1=0.5, b_1=100, the following results are obtained:

	sp	le	la	he
call aBm				
m=1	0.0387	0.0387	0.0387	0.0387
m=2	0.0387	0.0387	0.0387	0.0387
m=3	0.0387	0.0387	0.0387	0.0387
m=4	0.0387	0.0387	0.0387	0.0387
m=5	0.0387	0.0387	0.0387	0.0387
m=10	0.0387	0.0387	0.0387	0.0387
m=15	0.0387	0.0387	0.0387	0.0387
call gBm				
m=1	3.7302	3.7302	3.7302	3.7302
m=2	3.7302	3.7302	3.7302	3.7302
m=3	3.7302	3.7302	3.7302	3.7302
m=4	3.7302	3.7302	3.7302	3.7302
m=5	3.7302	3.7302	3.7302	3.7302
m=10	3.7302	3.7302	3.7302	3.7302
m=15	3.7302	3.7302	3.7302	3.7302
call ou				
m=1	0.1138	0.1138	0.1138	0.1138
m=2	0.1138	0.1138	0.1138	0.1138
m=3	0.1138	0.1138	0.1138	0.1138
m=4	0.1138	0.1138	0.1138	0.1138
m=5	0.1138	0.1138	0.1138	0.1138
m=10	0.1138	0.1138	0.1138	0.1138
m=15	0.1138	0.1138	0.1138	0.1138
put aBm				
m=1	0.0376	0.0376	0.0376	0.0376
m=2	0.0376	0.0376	0.0376	0.0376
m=3	0.0376	0.0376	0.0376	0.0376
m=4	0.0376	0.0376	0.0376	0.0376
m=5	0.0376	0.0376	0.0376	0.0376
m=10	0.0376	0.0376	0.0376	0.0376
m=15	0.0376	0.0376	0.0376	0.0376

[15] Meaning that d_t=1/12, that is one month.

put gBm				
m=1	3.8259	3.8259	3.8259	3.8259
m=2	3.8259	3.8259	3.8259	3.8259
m=3	3.8259	3.8259	3.8259	3.8259
m=4	3.8259	3.8259	3.8259	3.8259
m=5	3.8259	3.8259	3.8259	3.8259
m=10	3.8259	3.8259	3.8259	3.8259
m=15	3.8259	3.8259	3.8259	3.8259
put ou				
m=1	0.1235	0.1235	0.1235	0.1235
m=2	0.1235	0.1235	0.1235	0.1235
m=3	0.1235	0.1235	0.1235	0.1235
m=4	0.1235	0.1235	0.1235	0.1235
m=5	0.1235	0.1235	0.1235	0.1235
m=10	0.1235	0.1235	0.1235	0.1235
m=15	0.1235	0.1235	0.1235	0.1235

For an ***out-of-the-money call option*** with S_0=60, K=100, r_f=0.06, d=0.05, sigma$_1$=0.02, T=10, M=10.000, n=120, a_1=0.5, b_1=100, the following results are obtained:

	sp	**le**	**la**	**he**
call aBm				
m=1	0	0	0	0
m=2	0	0	0	0
m=3	0	0	0	0
m=4	0	0	0	0
m=5	0	0	0	0
m=10	0	0	0	0
m=15	0	0	0	0
call gBm				
m=1	0.8393	0.8393	0.8393	0.8393
m=2	0.8393	0.8393	0.8393	0.8393
m=3	0.8393	0.8393	0.8393	0.8393
m=4	0.8393	0.8393	0.8393	0.8393
m=5	0.8393	0.8393	0.8393	0.8393
m=10	0.8393	0.8393	0.8393	0.8393
m=15	0.8393	0.8393	0.8393	0.8393
call ou				
m=1	0.0941	0.0941	0.0941	0.0941
m=2	0.0941	0.0941	0.0941	0.0941
m=3	0.0941	0.0941	0.0941	0.0941
m=4	0.0941	0.0941	0.0941	0.0941
m=5	0.0941	0.0941	0.0941	0.0941
m=10	0.0941	0.0941	0.0941	0.0941
m=15	0.0941	0.0941	0.0941	0.0941

For an ***in-the-money put option*** with S_0=60, K=100, r_f=0.06, d=0.05, sigma$_1$=0.02, T=10, M=10.000, n=120, a_1=0.5, b_1=100, the following results are obtained:

	sp	le	la	he
put aBm				
m=1	39.7997	39.7997	39.7997	39.7997
m=2	39.7997	39.7997	39.7997	39.7997
m=3	39.7997	39.7997	39.7997	39.7997
m=4	39.7997	39.7997	39.7997	39.7997
m=5	39.7997	39.7997	39.7997	39.7997
m=10	39.7997	39.7997	39.7997	39.7997
m=15	39.7997	39.7997	39.7997	39.7997
put gBm				
m=1	39.7507	39.7507	39.7507	39.7507
m=2	39.7507	39.7507	39.7507	39.7507
m=3	39.7507	39.7507	39.7507	39.7507
m=4	39.7507	39.7507	39.7507	39.7507
m=5	39.7507	39.7507	39.7507	39.7507
m=10	39.7507	39.7507	39.7507	39.7507
m=15	39.7507	39.7507	39.7507	39.7507
put ou				
m=1	24.1471	24.1471	24.1471	24.1471
m=2	24.1471	24.1471	24.1471	24.1471
m=3	24.1471	24.1471	24.1471	24.1471
m=4	24.1471	24.1471	24.1471	24.1471
m=5	24.1471	24.1471	24.1471	24.1471
m=10	24.1471	24.1471	24.1471	24.1471
m=15	24.1471	24.1471	24.1471	24.1471

For an ***in-the-money call option*** with S_0=150, K=100, r_f=0.06, d=0.05, sigma$_1$=0.02, T=10, M=10.000, n=120, a_1=0.5, b_1=100, the following results are obtained:

	sp	**le**	**La**	**he**
call aBm				
m=1	49.7515	49.7515	49.7515	49.7515
m=2	49.7515	49.7515	49.7515	49.7515
m=3	49.7515	49.7515	49.7515	49.7515
m=4	49.7515	49.7515	49.7515	49.7515
m=5	49.7515	49.7515	49.7515	49.7515
m=10	49.7515	49.7515	49.7515	49.7515
m=15	49.7515	49.7515	49.7515	49.7515
call gBm				
m=1	49.8750	49.8750	49.8750	49.8750
m=2	49.8750	49.8750	49.8750	49.8750
m=3	49.8750	49.8750	49.8750	49.8750
m=4	49.8750	49.8750	49.8750	49.8750
m=5	49.8750	49.8750	49.8750	49.8750
m=10	49.8750	49.8750	49.8750	49.8750
m=15	49.8750	49.8750	49.8750	49.8750
call ou				
m=1	30.1731	30.1731	30.1731	30.1731
m=2	30.1731	30.1731	30.1731	30.1731
m=3	30.1731	30.1731	30.1731	30.1731
m=4	30.1731	30.1731	30.1731	30.1731
m=5	30.1731	30.1731	30.1731	30.1731
m=10	30.1731	30.1731	30.1731	30.1731
m=15	30.1731	30.1731	30.1731	30.1731

For an ***out-of-the-money put option*** with S_0=150, K=100, r_f=0.06, d=0.05, $sigma_1$=0.02, T=10, M=10.000, n=120, a_1=0.5, b_1=100, the following results are obtained:

	sp	**le**	**La**	**he**
put aBm				
m=1	0	0	0	0
m=2	0	0	0	0
m=3	0	0	0	0
m=4	0	0	0	0
m=5	0	0	0	0
m=10	0	0	0	0
m=15	0	0	0	0
put gBm				
m=1	1.5113	1.5113	1.5113	1.5113
m=2	1.5113	1.5113	1.5113	1.5113
m=3	1.5113	1.5113	1.5113	1.5113
m=4	1.5113	1.5113	1.5113	1.5113
m=5	1.5113	1.5113	1.5113	1.5113
m=10	1.5113	1.5113	1.5113	1.5113
m=15	1.5113	1.5113	1.5113	1.5113
put ou				
m=1	0.1026	0.1026	0.1026	0.1026
m=2	0.1026	0.1026	0.1026	0.1026
m=3	0.1026	0.1026	0.1026	0.1026
m=4	0.1026	0.1026	0.1026	0.1026
m=5	0.1026	0.1026	0.1026	0.1026
m=10	0.1026	0.1026	0.1026	0.1026
m=15	0.1026	0.1026	0.1026	0.1026

ii. Model with two stochastic variables

For an ***at-the-money option*** with S_0=100, K=100, r_0=0.1, r_f=0.06, d=0.05, $sigma_1$=0.02, $sigma_2$=0.15, T=10, M=10.000, n=120, a_1=0.1, b_1=100, a_2=0.001, b_2=0.1 the following results are obtained:

	aBm	**gBm**	**ou**
call			
m=1	0.0122	1.2298	0.0137
m=2	0.0122	1.2298	0.0137
m=3	0.0122	1.2298	0.0137
m=4	0.0122	1.2298	0.0137
put			
m=1	0.0209	1.0804	0.0235
m=2	0.0209	1.0804	0.0235
m=3	0.0209	1.0804	0.0235
m=4	0.0209	1.0804	0.0235

For an ***out-of-the-money call option*** with S_0=60, K=100, S_0=60, r_0=0.1, r_f=0.06, d=0.05, $sigma_1$=0.02, $sigma_2$=0.15, T=10, M=10.000, n=120, a_1=0.1, b_1=100, a_2=0.001, b_2=0.1, the following results are obtained:

call	**aBm**	**gBm**	**Ou**
m=1	0	2.0468	0.0112
m=2	0	2.0468	0.0112
m=3	0	2.0468	-
m=4	0	2.0468	-

For an ***in-the-money put option*** with S_0=60, K=100, S_0=60, r_0=0.1, r_f=0.06, d=0.05, $sigma_1$=0.02, $sigma_2$=0.15, T=10, M=10.000, n=120, a_1=0.1, b_1=100, a_2=0.001, b_2=0.1, the following results are obtained:

put	**aBm**	**gBm**	**Ou**
m=1	39.8024	39.5458	36.0193
m=2	39.8024	39.5458	36.0193
m=3	39.8024	39.5458	36.0193
m=4	39.8024	39.5458	36.0193

For an ***in-the-money call option*** with S_0=150, K=100, S_0=150, r_0=0.1, r_f=0.06, d=0.05, $sigma_1$=0.02, $sigma_2$=0.15, T=10, M=10.000, n=120, a_1=0.1, b_1=100, a_2=0.001, b_2=0.1, the following results are obtained:

call	aBm	gBm	ou
m=1	49.7497	50.3912	45.0118
m=2	49.7497	50.3912	45.0118
m=3	49.7497	50.3912	45.0118
m=4	49.7497	50.3912	45.0118

For an ***out-of-the-money put option*** with S_0=150, K=100, S_0=150, r_0=0.1, r_f=0.06, d=0.05, $sigma_1$=0.02, $sigma_2$=0.15, T=10, M=10.000, n=120, a_1=0.1, b_1=100, a_2=0.001, b_2=0.1, the following results are obtained:

put	aBm	gBm	ou
m=1	0.0100	1.7856	0.0210
m=2	-	-	-
m=3	-	-	-
m=4	-	-	-

6. CONCLUSION

The Least-Squares Monte Carlo algorithm of Longstaff & Schwartz provides a straightforward approximation of the price of a complex American-style option. This dissertation is the proof that the algorithm's results are robust, uniform and united. Moreover, it provides the potential to examine the differentiation and escalation of the price of the option being valued, based on the different stochastic process the stochastic variables may follow and on the different intrinsic values the option may has at time zero.

More specific, the results of the previous section indicate that the value of the option is identical regardless the type or the number of basis functions being used. For the same group of simulated paths regarding the stochastic variable/s, the price of the title is the same, with precision of four decimal places, for m being equal to 1, 2, 3, 4, 5, 10 or 15 and for the basis being either Simple Powers or Legendre, Laguerre or Hermite polynomial family. The theoretical background in favor of these results indicates that for a given degree any polynomial can be expressed as a linear combination of the others and, thus, the consistency of the prices arising is expected.

Another observation, based on the previous indicative results, that can be conducted, is the escalation of the price of options that differ only at the stochastic process the stochastic variable/s follow. The results regarding at-the-money and out-of-the-money options are consistent and show that always the option with an underlying asset's price following a Geometric Brownian Motion is more expensive than an option with an underlying asset's price following an Ornstein Uhlenberg process, which is more expensive than an option with an underlying asset's price following an Arithmetic Brownian Motion. On the other hand, for in-the-money calls, those with an underlying asset's price following a Geometric Brownian Motion are more expensive than those with an underlying asset's price following an Arithmetic Brownian Motion, which are more expensive than those with an underlying asset's price following an Ornstein Uhlenberg process. The in-the-money puts differ from the corresponding calls, with those with an underlying asset's price following an Arithmetic Brownian Motion being more expensive than those with an underlying asset's price following an Geometric Brownian Motion, which are more expensive than those with an underlying asset's price following an Ornstein Uhlenberg process.

Closing, the results provided here are in full accordance with the general principle stating that moving from deep-out-of-the-money towards deep-in-the-money areas of the underlying asset's price, the value of the option increases.

7. WEAKNESSES

The only, among so many, variance reduction technique used in the implementation in this dissertation is the antithetic variable one. Though, depending on the specific area or subject covered, another or plenty of other variance reduction techniques may be either more beneficial individually or add some benefit to the antithetic variable that was used here. So, it cannot be said that everything possible was done regarding variance reduction.

The main problem, though, is the fact that, concerning the second implementation under the two stochastic variables, no general solution is available. More specific, a code was developed to price an American-style option in the case that the max-exponent of the basis function is either 1, 2, 3 or 4. Further application, for degrees larger than 4 was not produced due to lack of time.

8. FUTURE EXTENSIONS

The implementation of the Least Squares Monte Carlo algorithm in this dissertation was narrow in terms of stochastic variables. Lack of time, prevented the expansion of the application in stochastic volatility or even in stochastic dividend yield rate. Both are extensions that are applicable to the written code and that would benefit the implementation of the Least Squares Monte Carlo algorithm of Longstaff & Schwartz.

Moreover, regarding the polynomial families used as basis function for the regression in order to estimate the conditional expectation function of the continuation value, there are plenty of them that were not used, but that would be of benefit to the results to be used in the future.

Having in mind, the automated world we live in, as well as the need of people with no university degrees to be able to value an investment that has been proposed to them or even an option they are likely to buy or sell, the algorithm presented here in a MatLab environment would be very useful and widely accepted in a user-friendly environment. That is, developing a user interface for the algorithm could result in an application used by a wide variety of people and in a wide variety of situations.

APPENDIX

i. *path*

```
function [St] = path(S0,rf,d,sigma1,T,n,M,a1,b1,flag)

if strcmp(flag,'gBm')

    for j=1:M
        GBM=gbm(rf-d,sigma1,'StartState',S0);
    end

    [S]=GBM.simulate(n,'DeltaTime',T/n,'nTrials',M,'Antithetic',true);

    for i=1:n
            St(:,i)=(S(i,:,:));
    end

elseif strcmp(flag,'aBm')

    for j=1:M
        ABM=bm(rf-d,sigma1,'StartState',S0);
    end

    [S]=ABM.simulate(n,'DeltaTime',T/n,'nTrials',M,'Antithetic',true);

    for i=1:n
        St(:,i)=(S(i,:,:));
    end

elseif strcmp(flag,'ou')
    St=zeros(M,n);
    St(:,1)=S0;

        for a=1:n-1
            z=randn(M+1,n);
            for j=1:M-1
                St(j,a+1)=exp(-a1)*St(j,a)+b1*(1-exp(-a1))+...
                    sigma1*sqrt((1/(2*a1))*(1-exp((-2)*a1)))*z(j,a+1);
            end
        end

else
    msg = 'Error occurred.Please select between gBm,aBm and ou.';
    error(msg)
end


end
```

ii. *price*[16]

```
function [c,p] = price(St,K,T,n,rf,M,m,basis,flag)

    OptCF=zeros(M,n);
    OptExe=zeros(M,n);

    for a=n:-1:2
        OptCF(:,a)=max(St(:,a)-K,0);
        f1=find(OptCF(:,a)~=0);
        OptExe(f1,a)=1;
        DisCF(:,a)=OptCF(:,a)*exp(-rf*(T/n));

        f2=find(St(:,a-1)>=K);

            if strcmp(basis,'sp')
                coeff=polyfit(St(f2,a-1),DisCF(f2,a),m);

                for b=1:m
                    s1(:,b)=St(:,a-1)*(coeff(:,b)^(m-b+1));
                end

                s1(:,m+1)=coeff(:,m+1);

                Cont=sum(s1,2);

            else
                x=[];
                for b=1:m+1
                    if strcmp(basis,'le')
                        x(:,b)=legendreP(b-1,St(f2,a-1));
                    elseif strcmp(basis,'la')
                        x(:,b)=laguerreL(b-1,St(f2,a-1));
                    elseif strcmp(basis,'he')
                        x(:,b)=hermiteH(b-1,St(f2,a-1));
                    end
                end

                coeff=regress(DisCF(f2,a),x);

                for c=1:m+1
                    s1(:,c)=St(:,a-1)*coeff(c,:);
                end

                Cont=sum(s1,2);
            end

        if Cont>=OptCF(:,a-1)
            OptCF(:,a-1)=0;
            OptExe(:,a-1)=0;
        else
            OptCF(:,a-1)=OptCF(:,a-1);
            OptExe(:,a-1)=OptExe(:,a-1);
        end
    end
```

[16] Only the case the user has chosen to value a call is presented.

```
    for c=2:n
        for d=1:M
            if OptExe(d,c)==1
                for e=c+1:n
                    OptExe(d,e)=0;
                    OptCF(d,e)=0;
                end
            else
                continue
            end
        end
    end

    for a=2:n
        OptCashDis(:,a)=OptCF(:,a)*exp(-rf*((a-1)*(T/n)));
    end

    c=sum(sum(OptCashDis)')/M
    p=0;
end
```

iii. *path2*

```
function [St,rt] = path2(S0,r0,d,sigma1,sigma2,T,n,M,a1,b1,a2,b2,flag)

       rt=zeros(M,n);
    rt(:,1)=r0;
    St=zeros(M,n);
    St(:,1)=S0;

    z=randn(M+1,n);
    w=randn(M+1,n);
    p=corrcoef(z,w);

    for ii=1:n
        for jj=1:M
            u(jj,ii)=(p(1,2)^2)*z(jj,ii)+(1-(p(1,2)^2))*w(jj,ii);
        end
    end

        for ii=1:n-1
            for jj=1:M-1
                rt(jj,ii+1)=exp(-a2)*rt(jj,ii)+b2*(1-exp(-a2))+...
                    sigma2*sqrt((1/(2*a2))*(1-exp((-2)*a2)))*u(jj,ii+1);

                if strcmp(flag,'gBm')
                    St(jj,ii+1)=St(jj,ii)*exp((rt(jj,ii+1)-((sigma1^2)/2)-
d)*(T/n)+sigma1*(sqrt(T/n))*z(jj,ii+1));
                elseif strcmp(flag,'aBm')
                    St(jj,ii+1)=St(jj,ii)+(rt(jj,ii+1)-
d)*(T/n)+sigma1*(sqrt(T/n))*z(jj,ii+1);
                elseif strcmp(flag,'ou')
                    St(jj,ii+1)=exp(-a1)*St(jj,ii)+b1*(1-exp(-
a1))+sigma1*sqrt((1/(2*a1))*(1-exp((-2)*a1)))*z(jj,ii+1);
                else
                    msg = 'Error occurred.Please select between gBm,aBm and
ou.';
                    error(msg)
                end
            end
        end

end
```

iv. *price2*[17]

```
function [c,p] = price2(St,K,T,rf,rt,M,n,m,flag)

OptCF=zeros(M,n);
OptExe=zeros(M,n);

for a=n:-1:2
    OptCF(:,a)=max(K-St(:,a),0);
    f1=find(OptCF(:,a)~=0);
    OptExe(f1,a)=1;
    DisCF(:,a)=OptCF(:,a)*exp(-rf*(T/n));

    f2=find(St(:,a-1)<=K);

    X=[];
    Y=[];
    Z=[];
    switch m
        case 1
            REGRESSORS=[ones(length(St(f2,a)),1) St(f2,a) rt(f2,a)];
        case 2
            for h=0:m
                j=m-h;
                X(:,h+1)=(St(f2,a).^h).*(rt(f2,a).^j);
            end

            REGRESSORS=[ones(length(X),1) St(f2,a) rt(f2,a) X];
        case 3
            for h=0:m
                j=m-h;
                X(:,h+1)=(St(f2,a).^h).*(rt(f2,a).^j);
            end

            for h=0:m-1
                j=m-h-1;
                Y(:,h+1)=(St(f2,a).^h).*(rt(f2,a).^j);
            end

        REGRESSORS=[ones(length(X),1) St(f2,a) rt(f2,a) X Y];
        case 4
            for h=0:m
                j=m-h;
                X(:,h+1)=(St(f2,a).^h).*(rt(f2,a).^j);
            end

            for h=0:m-1
                j=m-h-1;
                Y(:,h+1)=(St(f2,a).^h).*(rt(f2,a).^j);
            end

            for h=0:m-2
                j=m-h-2;
```

[17] Only the case the user has chosen to value a put is presented.

```
                Z(:,h+1)=(St(f2,a).^h).*(rt(f2,a).^j);
            end

        REGRESSORS=[ones(length(X),1) St(f2,a) rt(f2,a) X Y Z];
    end

    coeff=regress(DisCF(f2,a),REGRESSORS);

    for ii=1:m+1
        s1(:,ii)=St(:,a-1)*coeff(ii,:);
    end

    Cont=sum(s1,2);

    if Cont>=OptCF(:,a-1)
        OptCF(:,a-1)=0;
        OptExe(:,a-1)=0;
    else
        OptCF(:,a-1)=OptCF(:,a-1);
        OptExe(:,a-1)=OptExe(:,a-1);
    end
    end

for ii=2:n
    for jj=1:M
    if OptExe(jj,ii)==1
        for e=ii+1:n
            OptExe(jj,e)=0;
            OptCF(jj,e)=0;
        end
    else
        P=[];
    end
    end
end

    OptCashDis(:,:)=OptCF(:,:)*exp(-rf*((a-1)*(T/n)));

    c=0;
    p=sum(sum(OptCashDis)')/M
```

REFERENCES

Areal, N., Rodrigues, A., & Armada, M. R. (2008). *Improvements to the Least Squares Monte Carlo Option Valuation Method.* Portugal.

Barone-Adesi, G., & Whaley, R. E. (1987, June). Efficient Analytic Approximation of American Option Values . *The Journal of Finance*, pp. 301-320.

Broadie, M., & Glasserman, P. (1997). Pricing American-style securities using simulation. *Journal of Economic Dynamics and Control*, pp. 1323-1352.

Cerrato, M. (2009, September 8). *Valuing American Derivatives by Least Squares Methods.*

Clement, E., Lamberton, D., & Protter, P. (2002). An analysis of a least squares regression method for American option pricing. *Finance and Stochastic*, pp. 449-471.

Coskan, C. (2006). *Pricing American Options by Simulation.* Istanbul.

Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering.* New York: Springer.

Hull, J. C. (n.d.). *Option, Futures and Other Derivatives.* Pearson Education.

Judd, K. L. (1998). *Numerical Methods in Economics.* Cambridge: MIT Press.

Longstaff, F. A., & Schwartz, E. S. (2001, Spring). Valuing American Options by Simulation: A Simple Least-Squares Approach. *Review of Financial Studies*, pp. 113-147.

Miecielica, T. I. (2012). *Pricing of American Options by means of Least-Squares Monte Carlo methods in a stochastic volatility and interest rates framework.* Aarhus.

Moreno, M., & Navas, J. F. (2003, March 14). On the Robustness of Least-Squares Monte Carlo (LSM) for Pricing American Derivatives. *Review of Derivatives Research*.

Rogers, L. C. (2002, July). Monte Carlo Valuing of American Options. *Mathematical Finance*.

Stentoff, L. (2004). Assesing the Least Squares Monte-Carlo Approach to American Option Valuation. *Review of Derivatives Research*, pp. 129-168.

Tilley, J. A. (1993). *Valuing American Option in a Path Simulation Model.*

Tsekrekos, A. E., Shackleton, M. B., & Wojakowski, R. (2012). Evaluating Natural Resource Investments under Different Model Dynamics: Managerial Insights. *European Financial Management*, pp. 543-577.

Tsitsiklis, J. N., Fellow, IEEE, & Roy, B. V. (2001, July). Regression Methods for Pricing Comple American-style Options. *IEEE Transactions on Neural Networks*.