Valuing American Options by Simulation: Least Squares Monte Carlo

Carl Adrian Møller[a], Emil Bille[a], Silke Kofoed Christiansen[a]

[a]*Department of Economics, University of Copenhagen*
Course Coordinator: Bertel Schjerning
Keystrokes: ∼ 33,500,

Abstract

We investigate the robustness and efficiency of the well-known Least Square Monte Carlo (LSM) algorithm by Longstaff and Schwartz (2001) for option valuation. To value the option, the LSM relies on backward induction and functional approximation of the continuation value. We implement and analyse the impact of using different basis functions in the LSM algorithm. In order to reduce the computational speed we propose a Brownian Bridge simulation technique for a better implementation of the LS algorithm, with and without Halton draws for the terminal stock price. From our numerical experiments using different basis functions, we find that the LSM algorithm is robust to the choice of basis functions for a fixed number of polynomial terms. However, we find that the LSM algorithm is computationally expensive, and needs a considerable duration of time to produce a robust price. The LS algorithm with Brownian Bridge (LS-BB) reduces run time and manages to produce more robust pricing results. The LS-BB further increases the accuracy of the pricing approach, as opposed to the standard LSM.

Keywords: Option pricing, Monte Carlo Simulation, LSM algorithm, Brownian Bridge

1. Introduction

The challenge of correctly pricing an American put option comes down to finding the optimal stopping rule that maximizes the value of the American option. A variety of traditional finite difference and binomial techniques have been proposed to tackle the problem. Among the most popular techniques are the Monte Carlo simulation methods. The main advantages of using Monte Carlo simulations for pricing options, is that the method does not suffer from the curse of dimensionality. However, the Monte Carlo methods cannot estimate the optimal stopping rule. Thus, only using Monte Carlo simulations are not optimal for pricing American style put options. Carriere (1996) proposed a backward induction algorithm to price put options. The algorithm estimates the optimal stopping rule by estimation the conditional expectations using splines and local regressions. Longstaff and Schwartz (2001) proposed the Least Square Monte Carlo (LSM) algorithm to price options. The algorithm overcomes the challenge in combining the Monte Carlo simulation methods, which follow a forward nature, and the backward induction methods, which follow a backward nature. The LSM is concentrated around the approximation of value functions and demonstrates the efficiency of the least-squares regression approach.

In this paper, we will investigate the robustness and efficiency of the LSM algorithm for pricing American put options. We analyze the impact of using different basis functions to solve for the expected value function in a standard LSM algorithm. The types of basis functions in this paper include plain polynomials (Polyfit), Legendre, Hermite, Chebyshev, and Laguerre. To reduce computational speed, we introduce a Brownian Bridge simulation technique when implementing the algorithm. Finally, we compare using random draws with Halton draws, when constructing the terminal stock prices. We investigate the performance of the different applications through a set of numerical experiments. To evaluate the performance we rely on the BENCHOP option prices provided by von Sydow et al. (2015).

We find that the LSM algorithm is robust to the choice of basis functions for a fixed number of polynomial terms. However, the LSM algorithm is not robust to the number of polynomial terms, as the option price generally increases with the number of polynomial terms. When evaluating the squared standard error, the optimal number of polynomial terms is conditioned on the number of simulated paths. Next, we compare the performance of the LSM algorithm with the proposed LS-BB algorithm. We find that the LS-BB is considerably faster than the LSM algorithm. This is a direct result of reducing the memory usage when implementing the algorithms. More importantly, we find that the LS-BB algorithm generally yields lower standard errors than the LSM algorithm. Thus, the LS-BB is attractive in terms of both computational speed and accuracy. Finally, there is no benefit in using

Halton draws to construct the endpoints in the LS-BB algorithm, compared to a LS-BB without using Halton draws.

2. The Model Framework

2.1. The Underlying Asset

The dynamics of the underlying asset is based on the representation of the Black-Scholes(BSM) model formalized in Black & Scholes (1973). In this model the underlying asset price S follows a geometric Brownian motion. Moreover, under the risk-neutral probability measure Q, the drift rate of the asset S is the risk-neutral interest rate r. The dynamics of underlying asset S is generated by the stochastic differential equation in 1.

$$\frac{dS_t}{S_t} = rdt + \sigma dW_t, \tag{1}$$

where S_t is the price of the underlying asset at time t, with starting point S_0; r is the risk-free rate(expected return); σ is the volatility of the underlying asset; W_t is a one-dimensional Brownian motion. The solution to the stochastic differential equation in 1, is well known to be as in Equation 2.

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

The stock price at any given time t is known to be log-normally distributed. Equivalently we can write that the log of the stock price is normally distributed with mean and variance as in equation 3.

$$\ln(S_T) \sim N\left[\ln(S_t) + \left(r - \frac{\sigma^2}{2}\right)(T-t)\ ,\ \sigma^2(T-t)\right] \tag{3}$$

We simulate the underlying stock movements under the assumption of a constant risk-free rate r, and constant volatility, σ. When simulating the paths, we move into discrete time. We chose M discrete times $0 < t_1 \leq t_2 \cdots \leq t_M = T$. Thus, each simulation of movements in the underlying stock is generated through a Markov chain:

$$S_{t_m} = S_{t_{m-1}} \cdot \exp\left\{\left(r - \frac{\sigma^2}{2}\right)(t_m - t_{m-1}) + \sigma\sqrt{t_m - t_{m-1}}Z_m\right\} \tag{4}$$

where Z_m is a Gaussian vector of independent random draws. For any path obtained from Z_m, we generate a mirror image by changing the sign of all random draws in Z_m. The mirrored draws are called antithetic draws. This technique is beneficial since it reduces the number of samples to be generated in order to obtain N paths. Using antithetic draws also improves the precision of the sample by reducing the variance of the sample. Figure 1(a) shows 50 simulated paths for a Brownian motion, where the simulation is based on 25 simulated paths plus 25 antithetic paths:

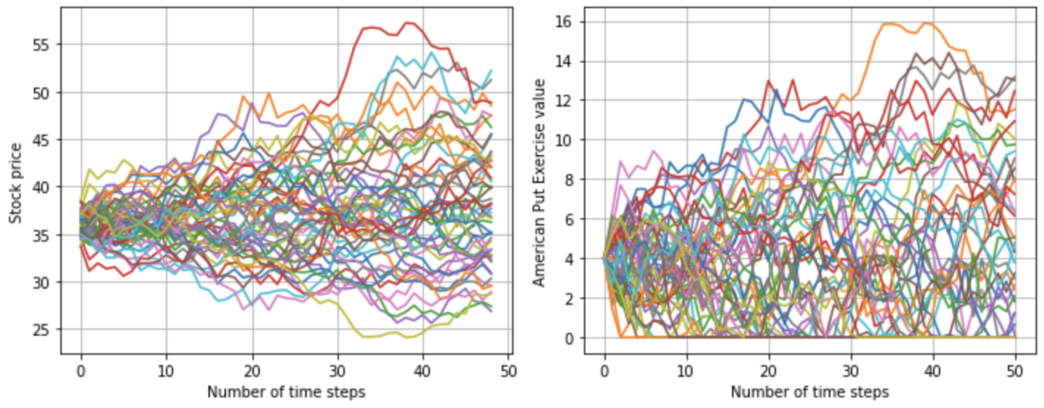


Figure 1: Simulated Paths

with drift $r = 0.05$ and diffusion coefficient $\sigma^2 = 0.02$, with $N = 50$ and $M = 50$.

2.2. The American Put Option

An American put option is a security giving the right to sell an asset, for a predetermined strike price, at any time before the option expires, given that the option is in the money (ITM). For a put option ITM means that the underlying asset is trading at a price that is lower than the strike price. The buyer of the option, in theory, continuously has to decide to whether exercise the option or keep holding it until maturity. The payoff of exercising a put option is given by:

$$g(s) = max(K - s, 0), \tag{5}$$

where K is the strike price and s is the asset price when exercised. The payoff from exercising yields the intrinsic value of the option, this is illustrated in Figure 1(b). The challenge of correctly determining the price of an American put option is to find the optimal exercise time that maximizes the value of the American option, also called the optimal stopping strategy. When the buyer has to decide if he wants to exercise at a given time, he has to take all possible future realisations into account. This complicates the valuation of the American option in contrast to European options, which can only be exercised at maturity.

2.3. The Valuation Framework

To valuate the American put option, we assume a finite time horizon $[0, T]$, and a complete probability space[1] $(\Omega, \mathcal{F}, Q)$. At maturity, T, the value of the option is simply the payoff, $g_T(s)$. At time t_{m-1}, the value of the option can be found by comparing the cashflow from exercising the put, $g_{m-1}(s)$, with the continuation value, $\mathbf{E}[V_T(S_T)|S_{t_{m-1}} = s]$. This can be formalized as the following backwards induction equation:

$$V_T(s) = g_T(s) \tag{6}$$

$$V_{t_{m-1}}(s) = \max\left[g_{t_{m-1}}(s), \mathbf{E}[V_{t_m}(S_{t_m})|S_{t_{m-1}} = s]\right], \qquad m = T, ..., 1 \tag{7}$$

The above is the formulation of the Bellman equation, and the key to finding the optimal exercise policy, and thus the value of the American option. To find $\mathbf{E}[V_{t_m}(S_{t_m})|S_{t_{m-1}} = s]$, we need to determine the future cash-flow generated by the put option. We introduce the notation $C(\omega, s; t_m, T)$ as the path of cash flows generated by the put option, conditional on (a) the option not being exercised at or prior to time t and (b) the option holder following the optimal stopping strategy for all $s, t_m < s \leq T$. Where the cash flow from immediate exercise is known to the investor at time t_m, i.e. $g_{t_m}(s)$, but the cash flows from continuation remain unknown. Following the no-arbitrage valuation theory, the value of continuation is given by taking the expectation of the remaining discounted cash flows $C(\omega, s; t_m, T)$ with respect to the risk-neutral pricing measure Q:

$$F(\omega; t_m) = \mathbf{E}[V_{t_m}(S_{t_m})|S_{t_{m-1}} = s] \tag{8}$$

$$F(\omega; t_m) = \mathbf{E}^Q\left[\sum_{j=m+1}^{M} \exp\left(-\int_{t_m}^{t_j} r(\omega, s)ds\right) C(\omega, t_j; t_m, T) \mid \mathcal{F}_{t_m}\right] \tag{9}$$

where $F(\omega; t_m)$ is the value of continuation at time t_m; $r(\omega, s)$ is the riskless discount rate; $\mathcal{F}_{t_m}$ is the information set at time t_m. In this paper we deal with a constant interest rate, where Equation 9 allows for a possibly stochastic interest rate.

[1] Ω is the set of all possible sample paths (ω); $\mathcal{F}$ is the sigma-algebra of events at time T; Q is a probability measure defined on the elements of F.

3. The LSM algorithm

The Least Square Monte Carlo (LSM) algorithm of Longstaff and Schwartz (2001) is used to solve the Bellman equation (7). The objective of the algorithm is to provide a path-wise approximation to the optimal stopping rule that maximizes the value of the American option. The LSM algorithm is based on the underlying idea that the conditional expectation can be approximated by a least-squares regression for each exercise date. The algorithm assumes that at time t_{M-1}, the continuation value , $F\left(\omega; t_{M-1}\right)$, can be expressed as a linear combination of countable sets of basis functions:

$$F\left(\omega; t_{M-1}\right) = \sum_{j=0}^{\infty} a_j p_j(X), \quad a_j \in \mathbb{R} \tag{10}$$

which is used to approximate the continuation value, by choosing a finite number of degrees in the polynomial $d < \infty$:

$$F_d\left(\omega; t_{M-1}\right) = \sum_{j=0}^{d} a_j p_j(X), \quad a_j \in \mathbb{R} \tag{11}$$

where $(p_j(\cdot))$ is the orthonormal basis functions. Once $(p_j(\cdot))$ has been specified, $F_d\left(\omega; t_{M-1}\right)$ is estimated. To be specific, the discounted cash flows, $C(\omega, s; t_{M-1}, T)$ is projected onto the basis functions, $(p_j(\cdot))$, for the paths where the option is in the money at time t_{M-1}. To illustrate how the LSM algorithm works, we price an American-style put option with a strike price of $K = 40$. Figure 2, illustrates the approximation of the continuation values using a family of Laguerre polynomials at time t_{M-1}.

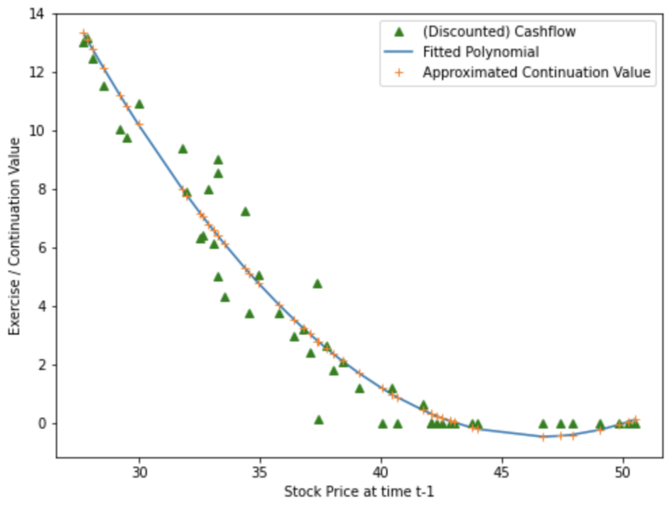


Figure 2: Functional Approximation

The simulation is based on 50 simulated paths for a Brownian motion, with drift $r = 0.06$ and diffusion coefficient $\sigma^2 = 0.02$, with $N = 50$ and $S_0 = 36$.

Early exercise is optimal when the immediate exercise value is larger than the continuation value obtained from functional approximation. The algorithm works backward, such that at each future scenario, and at each future time the option holder chooses a path given the information provided. This is illustrated for time step $m - 0$ and time step $m - 34$ below:

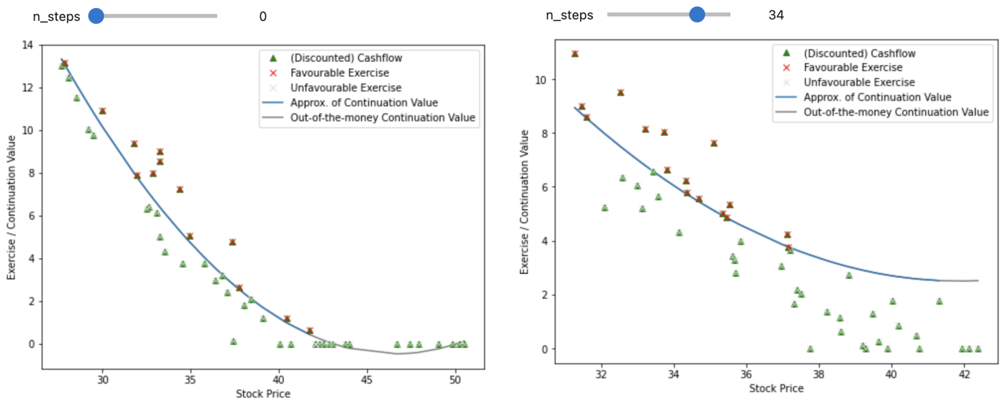


Figure 3: Optimal Stopping Rule

As the option can only be exercised once, it would be exercised on the first favourable exercise date. The option is then valued by discounting the cash flow back from the first favourable exercise date to time zero, and taking the average over all paths. The steps in the algorithm can be followed using the LSM process guide provided in section 3.2.

3.1. Choice of basis functions

In Longstaff and Schwartz, they proposed the Laguerre polynomial for the basic function, $p_j(\cdot)$. We want to test if the LSM option prices change given the choice of basis functions. We propose different methods of functional approximation including plain polynomials(Polyfit), Legendre, Hermite, Chebyshev, and Laguerre. The basis functions can be written in special cases of the Rodrigues' formula:

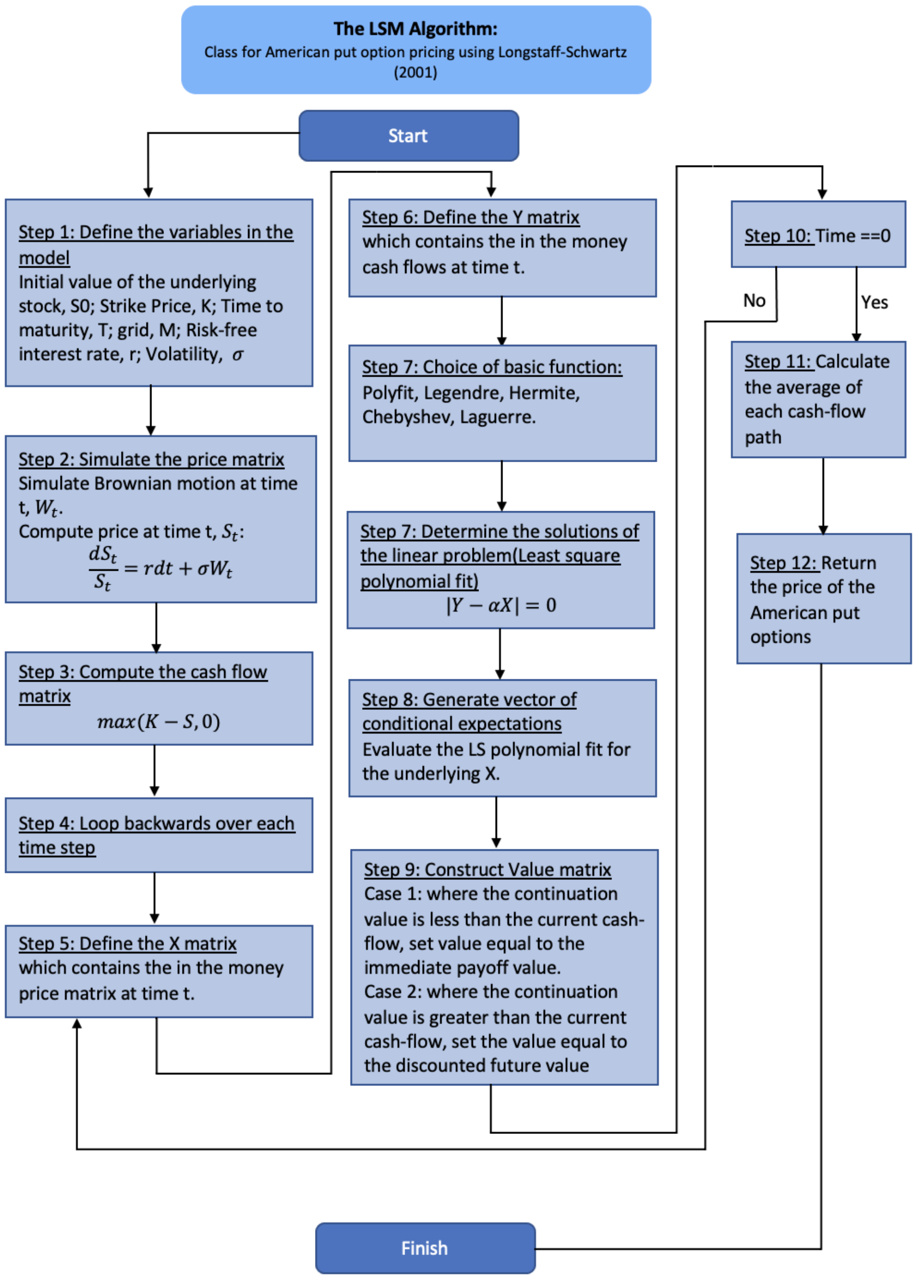
$$f_n(x) = \frac{1}{a_n g(x)} \frac{\partial^n}{\partial x^n} \left[\rho(x)(g(x))^n \right],$$

where n denotes the degree of the polynomial. The expressions for each of the polynomials are given in Table 1:

Name	$f_n(x)$	a_n	$\rho(x)$	$g(x)$
Polyfit	$P_n(x)$	$\frac{(2n)!}{n!}$	x^{2n}	1
Legendre	$L_n(x)$	$(1)^n 2^n n!$	1	$1-x^2$
Hermite	$H_n(x)$	$(-1)^n$	e^{-x^2}	1
Chebyshev	$C_n(x)$	$(-1)^n 2^n \frac{\Gamma(n+\frac{1}{2})}{\sqrt{\pi}}$	$(1-x^2)^{-\frac{1}{2}}$	$1-x^2$
Laguerre	$La_n(x)$	$n!$	e^{-x}	x

Table 1: Different basis functions

3.2. LSM pocket guide



The LSM Algorithm:
Class for American put option pricing using Longstaff-Schwartz (2001)
Start
Step 1: Define the variables in the model
Initial value of the underlying stock, S0; Strike Price, K; Time to maturity, T; grid, M; Risk-free interest rate, r; Volatility, σ
Step 2: Simulate the price matrix
Simulate Brownian motion at time t, W_t.
Compute price at time t, S_t:
$\frac{dS_t}{S_t} = rdt + \sigma W_t$
Step 3: Compute the cash flow matrix
$max(K - S, 0)$
Step 4: Loop backwards over each time step
Step 5: Define the X matrix which contains the in the money price matrix at time t.
Step 6: Define the Y matrix which contains the in the money cash flows at time t.
Step 7: Choice of basic function: Polyfit, Legendre, Hermite, Chebyshev, Laguerre.
Step 7: Determine the solutions of the linear problem(Least square polynomial fit)
$|Y - \alpha X| = 0$
Step 8: Generate vector of conditional expectations
Evaluate the LS polynomial fit for the underlying X.
Step 9: Construct Value matrix
Case 1: where the continuation value is less than the current cash-flow, set value equal to the immediate payoff value.
Case 2: where the continuation value is greater than the current cash-flow, set the value equal to the discounted future value
Step 10: Time ==0
No
Yes
Step 11: Calculate the average of each cash-flow path
Step 12: Return the price of the American put options
Finish

4. Replicating Longstaff and Schwartz (2001)

As a sanity check to see if the algorithm has been implemented correctly, we start off by replicating the results of the Longstaff and Schwartz paper. Following Longstaff and Schwartz (2001) we are interested in pricing an American-style put option on a collection of stocks following the stochastic differential equation given in equation (1). The starting point is a put option with a strike price of 40, the option is exercisable 50 times per year, the short term interest rate is 0.06 and we then vary the initial stock prices, volatility and number of years until final expiration. Specifically the initial price of the underlying asset starts at the values $S_0 = [36, 38, 40, 42, 44]$, the constant volatility takes the values $\sigma = [0.2, 0.4]$ and the years until final expiration $T = [1, 2]$. The Laguerre polynomials are used as basis functions.

parameters			sim results		LS results		comparison
S_0	σ	T	price	s.e	price	s.e.	diff in price
36	0.200	1.000	4.490	0.009	4.472	0.010	0.018
36	0.200	2.000	4.842	0.011	4.821	0.012	0.021
36	0.400	1.000	7.115	0.019	7.091	0.020	0.024
36	0.400	2.000	8.515	0.022	8.488	0.024	0.027
38	0.200	1.000	3.253	0.009	3.244	0.009	0.009
38	0.200	2.000	3.741	0.011	3.735	0.011	0.006
38	0.400	1.000	6.156	0.019	6.139	0.019	0.017
38	0.400	2.000	7.671	0.022	7.669	0.022	0.002
40	0.200	1.000	2.320	0.009	2.313	0.009	0.007
40	0.200	2.000	2.886	0.010	2.879	0.010	0.007
40	0.400	1.000	5.313	0.018	5.308	0.018	0.005
40	0.400	2.000	6.914	0.022	6.921	0.022	-0.007
42	0.200	1.000	1.625	0.008	1.617	0.007	0.008
42	0.200	2.000	2.219	0.010	2.206	0.010	0.013
42	0.400	1.000	4.592	0.017	4.588	0.017	0.004
42	0.400	2.000	6.241	0.021	6.243	0.021	-0.002
44	0.200	1.000	1.116	0.006	1.118	0.007	-0.002
44	0.200	2.000	1.698	0.009	1.675	0.009	0.023
44	0.400	1.000	3.957	0.016	3.957	0.017	-0.000
44	0.400	2.000	5.646	0.021	5.622	0.021	0.024

Table 2: Comparison of simulation results to the results of Longstaff and Schwartz

Overall we are able to replicate the results from Longstaff and Schwartz (2001). The results are not matched with total precision, but that is to be expected with the inherent randomness of monte carlo simulations. If we had the seed used by the original authors, we should in theory be able to replicate their results down to the last decimal. Our simulated results and the results from the original paper are quite close. The standard errors of the estimates are very similar to those obtained by Longstaff and Schwartz (2001), and they are low, ranging between 0.009 and 0.022. The difference in the final prices corresponds to a margin of error between between -0.7 and 2.2 cents (assuming a standard option of 100 shares of the underlying).

5. Choice of basis functions

Longstaff and Schwartz(2001) proposed using the Laguerre polynomial as the basis function mentioned before. In this paper we want to investigate the robustness of the LSM algorithm for pricing American put options numerically. To be specific, we want to vary the basis function used for the functional approximation. For simplicity, we only present the results for one American put option, i.e. one set of parameter values. The parameter set consists of $S_0 = 36, \sigma = 0.2, T = 2, r = 0.06, M = 50$. Table 4, reports the American option prices obtained by LSM using both different polynomials and increasing polynomial terms.

degree	polyfit	chebyshev	laguerre	hermite	legendre
2	4.414	4.414	4.414	4.414	4.414
3	4.429	4.429	4.429	4.429	4.429
4	4.431	4.431	4.431	4.431	4.431
5	4.459	4.459	4.459	4.459	4.459
6	4.457	4.457	4.457	4.457	4.457
7	4.467	4.467	4.467	4.467	4.467
8	4.474	4.474	4.474	4.474	4.474
9	4.472	4.472	4.472	4.472	4.472

Table 3: Simulated price of an American put option using different basis functions

For a fixed number of polynomial terms we obtain the same option price regardless of the the polynomial choice. Our results, therefore, indicate that the LSM algorithm is robust to the choice of basis functions. This is in line with existing papers, including (Longstaff and Schwartz, 2001) and (Fuentes, 2002). In the LSM algorithm, we define the Y matrix for each path using the actual realized cash flows along each path. Thus, the conditional expectations obtained from functional approximation is only used to compare the value of immediate exercise with the value from continuation. Consequently, any orthogonal polynomial can be used to derive the option price.

In Table 4 we also compare the impact of varying the number of terms in each polynomial on the option price. We find that the price of the American put option, in most cases, increase with the number of polynomial terms included. Our results, therefore, indicate that the LSM algorithm is not robust to the number of polynomial terms. The optimal number of polynomial terms is conditioned on the number of paths included in the simulation, due to problems arising from over/under-fitting. Later in the paper, such issues are addressed in more detail.

6. Backward simulation by Brownian Bridge

One drawback of the simulation approach used in the LSM-algorithm, is the increasing time it takes to compute an accurate result. Many alternatives has been proposed in order to increase computational speed, and in this paper we will focus on a backward simulation technique involving whats known as a Brownian Bridge. Specifically we combine the LSM algorithm with the Brownian Bridge simulation technique as in Chen and Zhou (2011). The main advantage of using a Brownian Bridge is that we can generate any stock price only given the start and the previous point. Thus, when simulating backwards using a brownian bridge, we don't have to save the matrix of simulated prices during all iterations. Since the LSM-algorithm uses backward induction, we can utilize the brownian bridge to simulate prices as we go back in time, and disregard them again when used for computation. This way we save both computer memory and computation time compared to the traditional LSM approach.

To simulate the stock paths backwards, we need to be able to simulate from a given terminal price S_T and for every simulated path still end up in todays stock price S_0. So we can have multiple terminal stock prices, but ultimatly, all the simulated paths need to end up in S_0.

In order to achieve this, Chen and Zhou (2011) use equation 12, to construct the brownian bridge. Here W_s denotes the value of the brownian bridge at time s, conditional on $W_0 = x$ and $W_T = y$.

$$W(s) = \frac{(T-s)W_0 + s \cdot W_T}{T} + \sqrt{\frac{(T-s)s}{T}}\epsilon \tag{12}$$

By a suitable discretization, we can then simulate a stock path using equation 12 and 2.

One way of deciding the terminal stock prices would be independent draws from the distribution as described in expression 3. With a sufficient amount of draws, this approach becomes a good approximation for all possible terminal values S_T.

Another approach could be to use pseudo-random numbers, to make sure the range of possible terminal values are filled out sufficiently. Using inverse transformation sampling, we can utilize Halton uniform draws, map them to standard normal draws, and finally take the exponential function in order to achieve log-normally distributed terminal values. These values are deterministic, and guaranteed to give a wide range of terminal stock prices. In order to illustrate the difference between these two approached, five terminal values S_T are drawn from a distribution as in expression 3, and five prices are generated from five Halton nodes with base 2. Figure 4 plots both versions of the terminal prices.

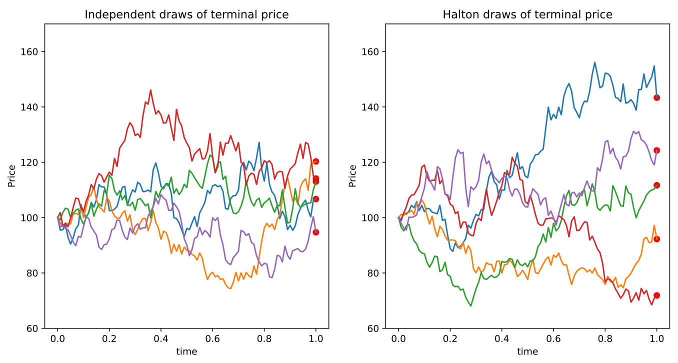


Figure 4: Halton and Independent terminal prices

The five independent draws happen to cluster together in this small simulation. The Halton draws however, are spread well out, which is expected. This leads us to hypothesise that using Halton draws for the terminal prices, could give a better approximation for the option price for a given number of simulated paths. As the number of simulated paths grows, this advantage would shrink, as the independent draws start to represent the actual log-normal distribution well.

The implementation of the Brownian Bridge approach, with and without Halton draws, are presented in pseudo-code table 2.

7. Performance

To evaluate the performance of the methods used we look at the accuracy and the compute time. There are mainly two levers that we can pull trying to improve the accuracy, increase the number of simulations (stock price paths) and increasing the degree of discretization in the time grid (number of time steps). Increasing the number of simulated paths in the monte carlo simulations should reduce the standard error of the price estimates bringing us closer to the "true" price. Increasing the number of time steps should reduce the price bias that stems from using a finite time grid when approximating the American option which can be continuously exercised. Both of these come at a tradeoff, because increasing either will increase the required computing time.

With American options, where there are no analytical solutions to the problem, it is hard to say what the "true" value is when evaluating the accuracy of the results. Thankfully von Sydow et al. (2015), the founders of the BENCHmarking project in Option Pricing (BENCHOP) provide the finance community with a set of common benchmark problems that can be used both for comparisons between methods and for evaluation of new methods. For the Black–Scholes–Merton model with one underlying asset, they provide reference values for valuation of American put options that are deep in the money (ITM), at the money (ATM) and out-of the money (OTM). It is interesting to see whether there is a difference in the performance of the methods for each type of option. Intuitively it should be easier to price options that are deep ITM because the intrinsic value of the option constitutes a larger share of the option's value compared to the extrinsic value. Following the same logic, ATM options should be slightly more difficult and deep OTM options should be the most dificult to price. Deep OTM options have no intrinsic value and their entire valuation comes from the extrinsic value, which means that the entire value of the option is based on the expected continuation value. Additionally deep OTM options will often have a very volatile return distribution, where the price will be close to 0 and then jump if the stock makes a large move.

Specifically BENCHOP provides reference values for American puts with at the strike price $K = 100$ evaluated for $S_0 \in \{90, 100, 110\}$, $\sigma = 0.15$, $r = 0.03$, $T = 1$.

S_0	90	100	110
u	10.726486710094511	4.820608184813253	1.828207584020458

Table 4: Reference values for American put price

Using their reference values, we evaluate the accuracy of the algorithm using the squared errors.

$$e_{sq} = (u - \hat{u})^2 \tag{13}$$

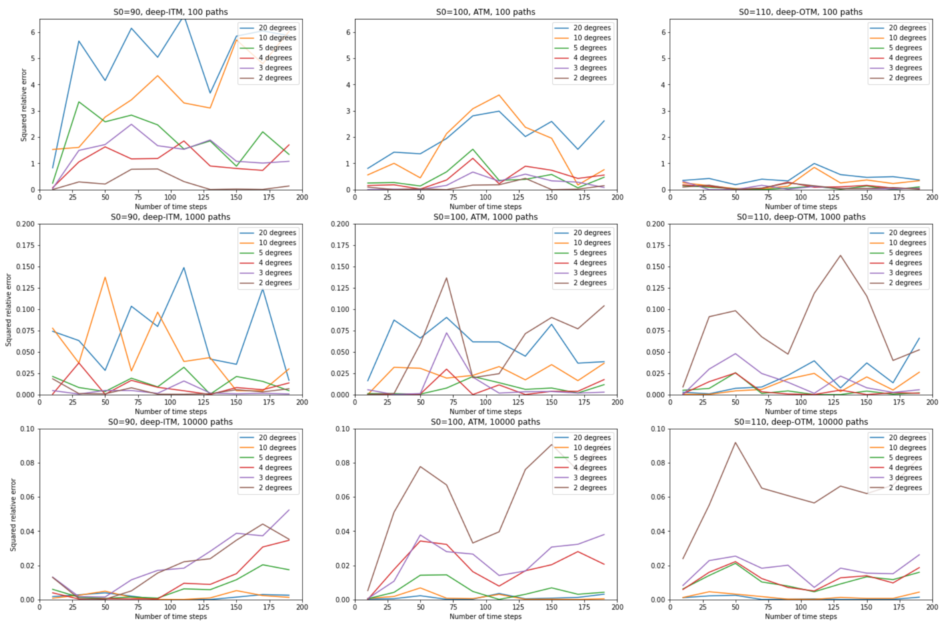


Figure 5: Squared errors

Figure 5 shows the squared errors when using the Laguerre polynomials with 2-5, 10, 20 degrees for an increasing number of time steps with 100, 1000 and 10000 simulated paths for time each step. Figure 5 yields many interesting results.

First, if the size sample being employed is small, a lower number of polynomial terms are favorable in order to reduce the squared errors. This is a direct result of the polynomials over fitting the cash-flows in the LSM algorithm. Vise versa, a high degree of polynomial reduces the squared errors for a larger sample size. Thus, the optimal number of polynomial terms is conditioned on the number of simulated paths.

Second, the performance of the LSM algorithm increase with the number of simulated paths. This is evident when comparing each row of sub plots, as a low number of simulated paths yields larger error. This is a direct result of the law of large numbers, and indicates that the LSM algorithm converges towards the true price when increasing the sample size. In relation to this, the LSM becomes more robust to the chosen number of polynomial terms when the number of simulated paths increase. Third, we note that the deep OTM put price is easier to price than the deep ITM put price. However, when increasing the number of simulated paths the squared errors decrease for the ITM put price. Our results, therefore, indicates that the deep ITM put price become easier to compute when simulating more paths.

7.1. Comparing least squares monte carlo and brownian bridge

In this section we examine the numerical performance of the proposed Longstaff Schwartz using the Brownian bridge algorithm (LS-BB) in comparison to the standard LSM algorithm. We will focus on investigating how the simulation properties of a Brownian Bridge and Halton Draws impact the performance, in terms of both accuracy and computational speed.

The goal of using a Brownian Bridge is to improve the speed of the LSM algorithm. As discussed in section 6, the needed memory storage increase with the number of time steps when using the standard LSM for pricing an American put option. This is in contrast to the LS-BB algorithm which, at least in theory, uses the same memory storing regardless of the number of time steps. Thus, we are interested in seeing if the reduction in memory storage reduces the computational speed. To do so, we present the result for American puts with the strike price $K = 100$ evaluated for $S_0 \in \{90, 100, 110\}$, $\sigma = 0.15$, $r = 0.03$, $T = 1$. For each method we use 10000 simulations and the first five Laguerre polynomial terms. Figure 6 shows the execution times when pricing an American put option using the LSM algorithm, the LS-BB, and the LS-BB with Halton nodes. The results in Figure 6 present the exponential growth in time given the logarithmic scale of the Y-axis. Our results from Figure 6 shows that the LS-BB algorithm is considerably faster than the LSM algorithm. Thus, the memory-efficient benefits of the LS-BB algorithm makes the algorithm attractive in terms of computational speed in comparison to the standard LS algorithm. We do not find an advantages in computational speed when using Halton draws to construct the terminal stock prices. This is as expected, as the purpose of using Halton draws are on the accuracy of the option price and not on the computational speed. Even

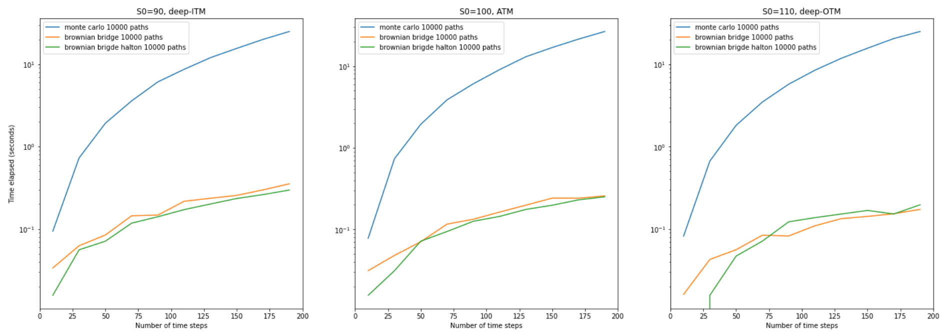


Figure 6: Time elapsed (log scale)

though, the goal of using a Brownian Bridge is to improve the implementation of the LSM algorithm, we are not interested in a trade off with the accuracy of the algorithm. To be specific, we expect the standard errors to be the same or lower for the LS-BB in comparison to the LSM algorithm. Figure 7 shows the squared standard errors when pricing an American put option using the LSM algorithm, the LS-BB, and the LS-BB with quadrature nodes. The results in Figure 7 shows that the LS-BB algorithm generally yields lower standard errors than the LSM algorithm. Our results thereby indicates that the LS-BB algorithm is superior in both computational speed and accuracy when pricing American put options. As mentioned the goal of using Halton draws to compute the terminal stock prices is to simulated better stock prices and therefore an algorithm which is more accurate. However, in terms of accuracy there is no benefit in pricing the American put option with a LS-BB with Halton draws compared to a LS-BB without Halton draw. To further investigate the advantages of LS-BB in practise compared to the LSM, we match the compute time of the methods. The LS-BB manages to produce 100.000 paths during roughly the same computation time that it takes the LSM algorithm takes to generate a price using 10.000 paths for each time

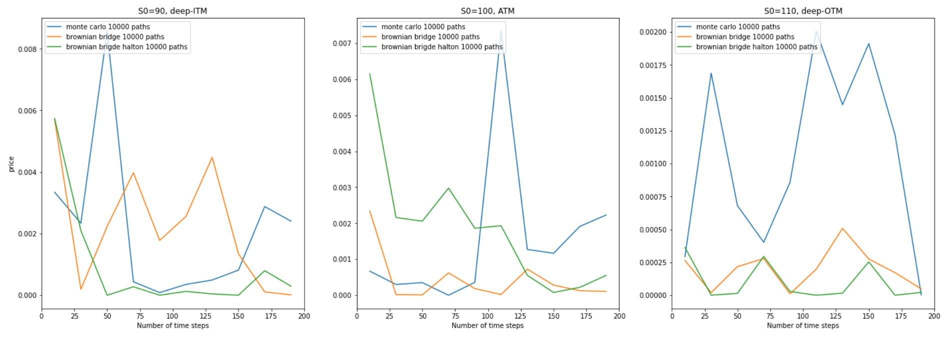


Figure 7: Simulated option price vs true value

step. The results are presented in Figure 8, where we find that the LS-BB algorithm quickly produce an accurate result.

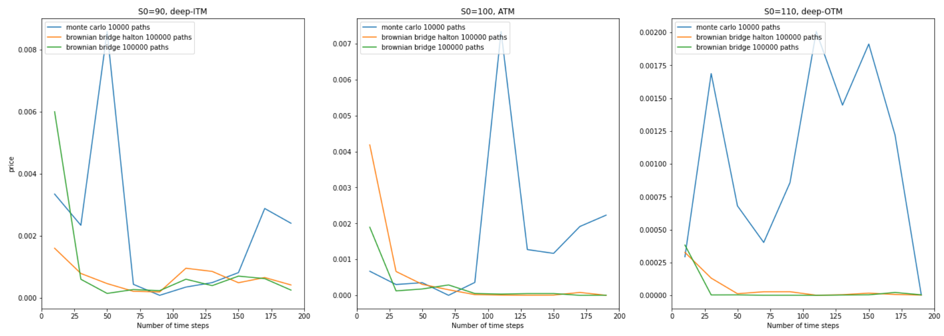


Figure 8: Matching compute time: Simulated option price vs true value

8. Conclusion

We have analyzed the impact of using different basis functions to solve for the expected value function in a standard Least Square Monte Carlo(LSM) algorithm. We have also proposed a method to reduce the computational speed of the standard LSM algorithm. The method relies on the properties of a Brownian Bridge to simulated the stock prices. The main advantage of using a Brownian Bridge is that we can generate any stock price only given the start and the endpoints. For the LSM algorithm this means that we only have to store the values needed for the current step, and disregard them after. Finally, we have proposed comparing different draws for the constructing the endpoints in the LS BB algorithm.

From numerical experiments using different basis functions, we found that (a) the LSM algorithm is robust to the choice of basis functions for a fixed number of polynomial terms, (b) the LSM algorithm is not robust to the number of polynomial terms, as the option price generally increases with the number of polynomial terms, (c) the optimal number of polynomial terms is conditioned on the number of simulated paths, (d) the LSM becomes more robust to the number of polynomial terms when the number of simulated paths increases.
Our numerical experiments shows that the computational time of the LSM algorithm, is increasing exponentially with the number of grid points. When comparing with the LS Brownian Bridge algorithm, we found that the LS BB is considerably faster than the LSM algorithm. This is a direct result of reducing the memory storage in the implementation. More importantly, we find that the LS Brownian Bridge algorithm generally yields lower standard errors than the LSM algorithm. Our results thereby indicates that the LS Brownian Bridge algorithm is superior in both computational speed and accuracy when pricing American put options. Finally, there is no benefit in using Halton draws to construct the endpoints in the LS BB algorithm, compared to a LS Brownian Bridge without using Halton draws.

Appendix A. Code

For the code used in the project, please refer to our Github.
`https://github.com/Adritch/DPtermpaper`

Appendix B. Longstaff and Schwartz Pseudo Code

Algorithm 1 Longstaff and Schwartz Method

```
1: Simulate paths $S_i(t)$ for $i = 1, 2, ..., M$ and $t = 0, t_1, t_2, ..., t_N = T$
2: Set terminal payoff $P_i(T) \leftarrow g(S_i(T))$
3: for $t$ from $t_{N-1}$ to $t_1$ do
4:     Set index $I_{ITM} = \{i_1, i_2, .., i_n\}$ where $g(S_i(t)) > 0$
5:     Set $X_i \leftarrow S_i(t)$ and $Y_i \leftarrow e^{-r\Delta t} P_i(t+1)$, where $i \in I_{ITM}$
6:     Regress $X$ on $Y$ and save coefficients $\boldsymbol{\beta}$
7:     Estimate $C(\hat{S_i}(t))$ using $\boldsymbol{\beta}$ and $X$
8:     Save $g(S_i(t))$ for $i \in I_{ITM}$
9:     for $i$ from 1 to M do
10:        if $i \in I_{ITM}$ and $g(S_i(t)) > C(\hat{S_i}(t))$ then
11:            $P_i(t) \leftarrow g(S_i(t))$
12:        else
13:            $P_i(t) \leftarrow e^{-r\Delta t} P_i(t+1)$
14:        end if
15:    end for
16: end for
17: Price $\leftarrow \frac{1}{M} \sum_{i=1}^{M} e^{-r\Delta t} P_i(1)$
```

Appendix C. LS with Brownian Bridge - Pseudo Code

Algorithm 2 LSM with Brownian Bridge

```
1:  M                                                  ▷ Number of simulations
2:  N                                                  ▷ # of timesteps
3:  if $Halton = True$ then
4:      Generate M Halton nodes
5:      Map nodes with quantile function $\rightarrow \epsilon_1, .., \epsilon_M$
6:      Set $W_i(t_N) = \epsilon_i$
7:  else
8:      Draw $W_i(t_N) \sim N(0,1)$ for $i = 1, 2, ..., M$
9:  end if
10: Generate terminal prices $S_i(t_N) = S_0 e^{(r-0.5\sigma^2)t_N + \sigma t_N W_i(t_N)}$
11: Initialise payoff vector $P_i \leftarrow g(S_i(T))$
12: for $t$ from $t_{N-1}$ to $t_1$ do
13:     Draw $M$ noise terms $\epsilon_i \sim N(0,1)$
14:     Continue bridge $W_i(t) = \frac{(T-t)W(0)+tW(t_n)}{t_n} + \sqrt{\frac{(t_n-t)t}{t_n}} \cdot \epsilon_i$
15:     Generate prices $S_i(t) = S_0 e^{(r-0.5\sigma^2)t + \sigma t W_i(t)}$
16:     Set index $I_{ITM} = \{i_1, i_2, .., i_n\}$ where $g(S_i(t)) > 0$
17:     Set $X_i \leftarrow S_i(t)$ and $Y_i \leftarrow e^{-r\Delta t} P_i$, where $i \in I_{ITM}$
18:     Regress $X$ on $Y$ and save coefficients $\boldsymbol{\beta}$
19:     Estimate $C(\hat{S_i}(t))$ using $\boldsymbol{\beta}$ and $X$
20:     for $i$ from 1 to M do
21:         if $i \in I_{ITM}$ and $g(S_i(t)) > C(\hat{S_i}(t))$ then
22:             $P_i \leftarrow g(S_i(t))$
23:         else
24:             $P_i \leftarrow e^{-r\Delta t} P_i$
25:         end if
26:     end for
27: end for
28: Price $\leftarrow \frac{1}{M} \sum_{i=1}^{M} e^{-r\Delta t} P_i$
```

References

Chen, Y., Zhou, Y., 2011. Pricing american options on assets with dividends by a brownian bridge simulation method. Proceedings of 2011 International Conference on Computer Science and Network Technology, ICCSNT 2011 1, 185–189. doi:`10.1109/ICCSNT.2011.6181937`.

Fuentes, M.M., 2002. On the robustness of least-squares monte carlo (lsm). .. credit risk models in the banking system view project macro-finance models for the term structure of interest rates view project. URL: `https://www.researchgate.net/publication/239066103`.

Longstaff, F.A., Schwartz, E.S., 2001. Valuing american options by simulation: A simple least-squares approach. Review of Financial Studies IS. No. I, 113–147.

von Sydow, L., Höök, L.J., Larsson, E., Lindström, E., Milovanović, S., Persson, J., Shcherbakov, V., Shpolyanskiy, Y., Sirén, S., Toivanen, J., Waldén, J., Wiktorsson, M., Levesley, J., Li, J., Oosterlee, C.W., Ruijter, M.J., Toropov, A., Zhao, Y., 2015. Benchop – the benchmarking project in option pricing. International Journal of Computer Mathematics 92, 2361–2379. doi:`10.1080/00207160.2015.1072172`.