An Exploration of Convergence in the Least Squares Monte Carlo Algorithm for Evaluating American Put Options

Affaan Ali, Seamus Lam, David Liu, Edwynn Turner

University of Ottawa

MAT 4372: Financial Mathematics

Dr. François-Michel Boire

April 17, 2025

Table of Contents

Abstract

The report presents an analysis of how varying parameters of the Least Squares Monte Carlo (LSMC) algorithm affect the absolute convergence of the value of an estimated American put option with respect to the underlying value derived from the binomial model. Parameters for the models are obtained from the S&P 500 index and United States treasury bill rate data. The number of simulation paths, the choice of polynomials, and the degree of the polynomials were all considered as variables within the algorithm. Particularly, the effectiveness of four functions as bases for the regression was explored: standard, Laguerre, and Hermite polynomials, in addition to custom functions using European option payoffs. Further, additional analysis regarding the number of possible exercise dates and the degree to which the put is In-The-Money are briefly touched upon. An At-The-Money American put was selected as a base case, with a possibility of M=50 early exercise dates; however, analysis suggested that puts deeper In-The-Money and with fewer possible early exercise dates would have contributed to lower absolute errors. Results indicated that the choice of polynomial had a negligible effect on overall convergence, while second-degree and higher degree polynomials displayed slightly reduced absolute errors. The custom functions also outperformed both the standard and orthogonal polynomials. A question manifesting itself throughout the study concerns how one can increase the efficiency of simulation whilst still minimizing errors within the results.

Introduction

While the study of pricing European Options using the Black and Scholes model has become commonplace, appearing in all introductory finance textbooks that introduce option trading, the analysis of pricing their American counterparts is much more complex, and particularly beyond the scope of the undergraduate level. Difficulty arises in determining optimal early exercising of the option, which can take place over an interval of time up until or at maturity. Research into the matter dates back to the 1970s, but has become more prevalent within the past few decades, with a few divergent methods for valuation being proposed. Perhaps the most studied of them in recent years is the Least Squares Monte Carlo (LSMC) algorithm, first proposed by Longstaff and Schwartz in 2001. Within the paper, a plethora of factors and parameters are presented for the algorithm, without a comprehensive discussion as to what yields the most accurate results, or whether there even would be any perceivable change at all amongst them. Upon our first reading of the paper, this was a question that naturally materialized. By consequence, the topic of LSMC not only presented itself as an interesting and challenging extension of the skills learned within this class, branching itself into a new topic never seen within our finance courses either, but also allowed for a significant degree of experimentation and exploration in endeavouring to answer our initial queries. The scope of this report is to determine how combinations of the number of randomly simulated paths, the type of polynomials used, and the degree of those polynomials will affect the convergence of the LSMC algorithm in valuing American put options. Using real-world data from the S&P 500 stock index, over a time frame from 2009 to 2019, we will estimate the true value of an American put option, using the convergence of a binomial tree with an increasing number of periods. Subsequently, we will compare the absolute error of an American put option estimated using the LSMC algorithm, whilst varying the parameters of the algorithm. We wish to test whether there is indeed a noticeable increase in convergence accuracy and error reduction that arises from the polynomials, their degrees, or the number of simulations. Throughout the project, we will design and test codes for both the binomial and LSMC methods, providing an assessment of their effectiveness and limitations.

Background

S&P 500 Stock Index

Standard and Poor's 500 (S&P 500) stock index is generally used as a benchmark and a statistical summary of how the 500 largest publicly traded companies in the U.S. are performing. It was selected as the source of data guiding the paper's analysis.

Its value represents the market capitalization weighted measure of the 500 companies in the index, where larger companies have more statistical influence (Kenton, 2024). This signifies that each company's weight in the index is proportional to its total market value, computed by multiplying the number of outstanding shares by its stock price. This ensures that larger companies such as Microsoft, NVIDIA, and Amazon exert significantly more influence on the index's movements. The final index value is scaled using a divisor that is not released to the public (Kenton, 2024).

American Option Valuation

We are concerned with two classifications of options: American options and European options.

An American option gives the holder the right to buy or sell an underlying asset at strike price K any time before or on the date of maturity, whereas a European option only allows the holder to exercise their right to buy or sell the asset at maturity. The primary difference between the two lies in their timing, which forms one of the central focuses of our analysis throughout this report. In contrast to the valuation of European options, which has a closed-form solution, such a solution does not exist for their American counterparts, unless strict and unrealistic restrictions are applied (Zhao, 2018). The first solution, the binomial model, was presented in 1979, by Cox, Ross, and Rubinstein. A detailed description of this method follows below. To this day, the binomial method is still considered to produce the most accurate result (Zhao, 2018); hence, it has been chosen as the basis for comparison. A similar trinomial method has also been proposed, which implements additional horizontal jumps to the binomial model. Some researchers, such as Horasanli in 2007, have suggested that this will reach equally accurate convergence but at a faster rate. Other American option valuation methods include the finite difference techniques theorized by Schwartz in 1977, which was further developed by Hull and White in 1990, and the quadratic approximation, proposed by Barone-Adesi and Whaley in 1987. The latter, while running efficiently, lacks convergence accuracy. One of the earliest proposed usages of Monte Carlo techniques was that of Tilley in 1993, which employs a bundling technique for option pricing. Nonetheless, this has since been proven unsatisfactory, due to biased estimators and its inability to be generalized to higher dimensions.

Binomial Model

The binomial model is a discrete-time model used to value options based on possible future prices of an asset, returning the expected discounted payoff under the risk-neutral measure. The movement of the underlying asset is calculated using risk-neutral probability ***p*** and its complement ***q***, and its ***u*** (up) and ***d*** (down) factors that change its value at each time step. The expected payoff is discounted accordingly through a constant risk-free interest rate ***r*** applied to each time step.

The binomial model also assumes a recombining structure with unchanging ***u, d,*** and ***r*** parameters throughout its time to maturity, which is unrealistic in most practical cases.

In our binomial model, we are concerned with only two **tradable assets**:

1. **A risky asset**
2. **A risk-free asset**

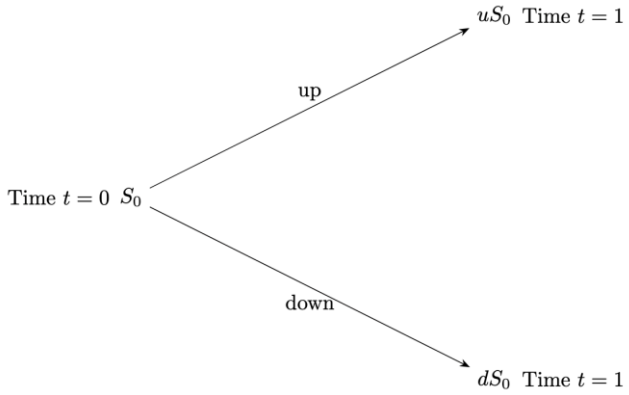
We assume that these assets can be bought or sold on demand at any time depending on if it is subject to an American option or a European option. These assets are typically used in the construction of portfolios.

For options in the binomial model, we are strictly concerned with **call** and **put** options.

1. **Call option** gives the holder the right, but not the obligation, to buy an asset at strike price ***K***.
2. **Put option** gives the holder the right, but not the obligation, to sell an asset at strike price ***K***.

Examining the process of a binomial model, let us first consider time steps: $t = 0$ and $t = 1$. Suppose $t = 0$ is the present, and $t = 1$ represents the future by 1 time step. The risky asset has a known value of S_0, which denotes the price of the risky asset at $t = 0$. At $t = 1$, the risky asset will have two distinct values S_1: $\boldsymbol{uS_0}$ and $\boldsymbol{dS_0}$, representing the possible values of S_1 after an upward or downward movement, respectively.

These different movement possibilities make up our binomial tree, which is a visually convenient way of depicting the different price paths of all possible combinations of sequences of ***u*** and ***d*** for all $t \leq T$, where T denotes the time of maturity. So, for our current example:



In our case, we assume that $uS_0 > dS_0$. And for our risk-free asset, we are likewise given the initial value B_0, which denotes the value of the risk-free asset at time $t = 0$. At time $t = 1$, assuming discretely compounded interest, our initial value is multiplied by a factor of $(1 + r)$, where *r* denotes the risk-free interest rate of our risk-free asset. For a larger number of time steps, this amount is compounded at each step until maturity.

Risk-neutral probability ***p*** denotes the probability that our risky asset increases by a factor of ***u*** under the risk-neutral measure. It is computed as $p = \frac{(1 + r) - d}{u - d}$

At maturity, we are given terminal payoffs. Payoff formulas differ based on the type of option that is subject to our model. Payoffs at maturity are $V_T = max(S_T - K, 0)$ for **call options**, and $V_T = max(K - S_T, 0)$ for **put options**. S_T denotes the possible values of the risky asset based on different sequences of ***u*** and ***d*** movements.

From this, we can now compute the expected discounted future payoff using backwards induction. The backwards induction formula is denoted as

$$V_t = \frac{1}{1 + r}\left[p\, V_{t+1}^{\uparrow} + (1 - p)V_{t+1}^{\downarrow}\right].$$

It provides the discounted expected payoff at each time step $t \leq T$, depending on the sequence of ***u*** and ***d*** factors.

For our example, we have two values of S_1 which gives us two possible payoffs: $V_1^{\uparrow}$ and $V_1^{\downarrow}$. From this, we can compute V_0 using $V_0 = \frac{1}{1 + r}\left[p\, V_1^{\uparrow} + (1 - p)V_1^{\downarrow}\right]$, giving us the discounted expected payoff for *t = 1* at *t = 0*.

The primary shortcoming of this model is the fact that it is computed discretely, being unable to capture more intricate movements in continuous time. For a European option, the binomial model is convenient because of the time of exercise, which is at maturity. This way, the binomial model only needs to compute one possible payoff at maturity. But for an American option that allows exercise at any point prior to maturity, the binomial model's performance quality will depend on more time steps, which comes at a cost of greater computational complexity. This trade-off between accuracy and efficiency is important when valuing American options, as capturing early exercise opportunities often requires a significantly finer time grid.

The binomial model also assumes a recombining structure with unchanging ***u, d,*** and ***r*** parameters throughout its time to maturity, which is in most, if not all practical cases, unrealistic.

Least Squares Monte Carlo Algorithm

In mentioning the shortcomings of the binomial model, we are left to explore alternative ways of valuing American options that both reduce computational intensity and open more doorways to statistical analysis.

Longstaff and Schwartz' (2001) Least Squares Monte Carlo (LSMC) method addresses both the problem of computational efficiency and the challenge of accurately estimating the continuation value of American options. It has been shown to outperform other estimation techniques both on the basis of speed and accuracy (Zhao, 2018). Assuming the preliminary information from the Binomial Model section above, the process of LSMC is as follows:

1. Simulating asset price paths: This involves generating a large number of possible price trajectories for the underlying asset using Monte Carlo simulation under the risk-neutral measure. At each discrete time step, the asset price is updated based on a stochastic process such as geometric Brownian motion. We will indeed employ Brownian motion in our code.
2. Determine payoff for each price path at maturity: Payoffs are computed using $max(K - S_T, 0)$ for American put options, which are most relevant to our discussion.
3. Identify the In-the-Money (ITM) paths. At each time step *t*, a path is considered in-the-money if immediate exercise yields a positive payoff $V_t > 0$.
4. Estimate continuation values: The continuation value at time *t* is an estimate of the expected discounted value of the option at the next time step *t + 1*. This is conducted through **linear regression**, which fits a function, traditionally a polynomial, that minimizes the sum of squared errors between predicted and actual continuation values. Polynomial degrees can vary, with higher degrees fitting training data more precisely but at a higher risk of overfitting. The linear regression provides an approximation of the conditional expectation $E\left[e^{\{-r\,\Delta t\}}\,V_{t+1} \mid S_t\right]$, which represents the continuation value of the option at time *t* under the risk-neutral measure.
5. Compare continuation value and immediate exercise value for each path: For each simulated path, we find the earliest time step *t* such that
$$immediate\ exercise\ value > continuation\ value,$$
which defines the optimal exercise time. We then record the payoffs for each of these paths from their optimal stopping times and set all future cash flows on that path to zero. Finally, we discount all payoffs to the present value and take their average to estimate the option price.

The LSM method is generally robust, particularly with respect to the varying types and number of polynomials used for the regression step. This signifies that the estimators obtained vary only marginally with the different polynomial types, quantities, and degrees. This is especially true for American options, where tests consistently give estimators with low variance (Moreno and Navas, 2003).

In theory, the reliability of the LSMC estimator improves as the number of simulated paths increases. This is further strengthened by LSMC's asymptotic behaviour. By the Law of Large Numbers, as the number of simulated paths $N \to \infty$, the LSMC estimator converges to the true expected option value under the risk-neutral measure (Stentoft, 2004). Additionally, by the Central Limit Theorem, the distribution of the Monte Carlo estimator approaches a normal distribution (Stentoft, 2004), allowing the construction of confidence intervals to quantify the precision of the estimate.

Longstaff and Schwartz (2001) have demonstrated that the LSMC method is effective for both single and multifactor problem settings. Even in high-dimensional problems, it remains computationally feasible and accurate within reasonable model parameters.

However, just like any other regression-based method, LSMC is not immune to overfitting. The use of too many basis functions at high degrees may begin to fit random noise in the simulated paths rather than the true continuation value. Significant errors in estimates can lead to biased and unstable pricing predictions. This issue becomes more pronounced in higher-dimensional settings or when pricing complex American-style derivatives. Nonetheless, to mitigate the risk of overfitting associated with the use of a single set of basis functions, we can use a model averaging approach for the LSMC method (Wei and Zhu, 2022). This involves combining predictions from multiple regression models with varying basis functions and averaging them to reduce the variance of the estimator. This not only improves prediction accuracy but further enhances the robustness of the LSMC algorithm.

Polynomials and Orthogonal Bases

Polynomials play a crucial role in LSMC as they are at the core of the regressions conducted throughout the algorithm. At each time step, the continuation value of the option under the risk-neutral measure is approximated using conditional expectation; this is completed in accordance with future discounted values of the option which are regressed on a set of basis functions. Therefore, the choice of basis functions selected is important as it will help determine the efficacy and efficiency of the process. The classification of basis functions that are most often used for LSMC is known as orthogonal basis functions; therefore, it is important to briefly explore these functions and how they are characterized, since they differ in use and definition from regular polynomials.

The guiding idea behind orthogonal polynomials is quite simple; they are defined as families of polynomials that have an inner product of 0. More specifically, polynomial families (in this case f and g) are considered orthogonal on an interval $[a, b]$ with respect to a weighting function if

$\int_a^b f(x)g(x)w(x) = 0$; where $w(x)$ is the non-negative weighting function.

Orthogonal polynomials are useful across many disciplines, with specific families of orthogonal polynomials having applications in various fields such as Chemistry, Physics, Engineering, and Mathematics (Ikekwere, 2023). Core properties of orthogonal polynomials include their linear

independence amongst members of the polynomial family, and most importantly for our application, when used in regression analysis, they allow for uncorrelation, unlike other polynomials. In fact, literature in regression analysis often favours an orthogonal polynomial regression due to advantages such as better curve fitting for interpolation and extrapolation. Regression completed with orthogonal polynomials allows for parameter estimates that do not depend on the degree of the polynomial, thus resulting in more stable estimates of coefficients in general. Further, the computational time can be almost twice as fast when using orthogonal polynomials compared to ordinary polynomials (Narula, 1979).

Returning to the topic of LSMC, given the aforementioned advantages of orthogonal polynomials, it will be of interest for this project to test both regular polynomials in addition to orthogonal polynomials. Given the importance of fitting procedures, such as assuring that one is not over-fitting during the regression, orthogonal polynomials should prove advantageous according to previous literature.

In terms of the polynomials used for this project, the set of basis functions for the LSMC regressions will consist firstly of standard polynomials (see **Table 2** in the Results section), in addition to two families of orthogonal polynomials which are as follows:

1. **Laguerre Polynomials** - A family of orthogonal polynomials that are non-trivial solutions to a specific second-order linear ordinary differential equation in the general form

   $$xy'' + (1 - x)y' + \lambda y = 0,$$

   where parameter $\lambda \in \mathbb{R}$. Laguerre polynomials are known to handle exponential decay especially well. The generating function for Laguerre polynomials can be seen using a Rodrigues representation which is given as

   $$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}(e^{-x}x^n)$$

2. **Hermite Polynomials** – A family of orthogonal polynomials that are orthogonal with respect to a weighting function of e^{-x^2} and over $(-\infty, \infty)$. The generating function for Hermite polynomials can be seen using a Rodrigues representation which is given as

   $$H_n(x) = (-1)^n e^{x^2}\frac{d^n}{dx^n}\left(e^{-x^2}\right)$$

   Since Hermite polynomials are orthogonal, they satisfy the orthogonality property

   $\int_{-\infty}^{\infty} H_m(x)H_n(x)e^{-x^2}\,dx = 0$, where $m \neq n$ (Arfken, 1985).

Tables 3 and **4**, presented within our Results section, display the first six Laguerre and Hermite polynomials up until $n = 5$, which were used as basis functions within the regression.

Methodology

Resulting from the characteristic of an American call option that early exercise is never optimal, assuming the absence of dividends being paid on the underlying stock, its value would be treated no differently than that of a European call option with the same parameters. Nonetheless, this does not hold for an American put option, whose value will always be larger than or equal to a respective European put. Particularly within a low volatility environment, early exercise is not uncommon. For this reason, it is important to test the LSMC algorithm on an American put option, which cannot be valued with the Black and Scholes model used for its European counterpart.

Testing the “true” value of the American Put option will be completed using the binomial model, as described extensively within the Background, due to its accuracy. This does not require randomly simulated Brownian motions but instead is based on upturns and downturns in the stock price that are calculated directly from the volatility of the underlying stock and the length of the time increments between each period. Risk-neutral probabilities are also estimated, incorporating these upturns and downturns, coupled with the risk-free interest rate prevailing in the market. Therefore, the first step is to extract these required parameters directly from market data. For further coding details, please refer to **Appendix A**.

For our market data, the timeframe from January 2009 to December 2019 was selected, for a total of 11 years. This period has relatively low volatility and the absence of widespread economic shocks, such as the 2007-2008 Financial Crisis and the COVID-19 pandemic, which took place immediately before and after our chosen timeframe. Both had far-reaching economic impacts, including plummeting stock markets and a significant lowering of interest rates through expansionary monetary policy, which would have made the results from our analysis more variable. The selected period is characterised by overall steady economic growth, as observable with the upward trend in the S&P 500 Stock Index.

Risk-Free Interest Rate Data Collection

While an interest that is truly “risk-free” is unobservable in the market, the closest approximation is the 3-month treasury bill (T-Bill) rate. Given that we are analyzing the S&P 500 Index, which considers the largest companies traded on American stock exchanges, we decided it would be more fitting to select the American Federal T-Bill rate. This represents the rate on short-term debt obligations backed by the United States Treasury. Due to their short maturity, these are deemed to be low risk. Data was collected from Bloomberg software, which allows for the retrieval of the rate at a daily frequency.

As observable in **Figure 1** below, the 3-Month Treasury Bill Rate remained stable between 0.2% and 0.8% between 2010 to 2015. Throughout 2016 to early 2019, the interest rate rose to nearly 2.5%. The Federal Reserve cited reasons for this, including promotion of job growth, concerns of inflation amidst the growing economy, and efforts to increase the savings rate (The New York Times).

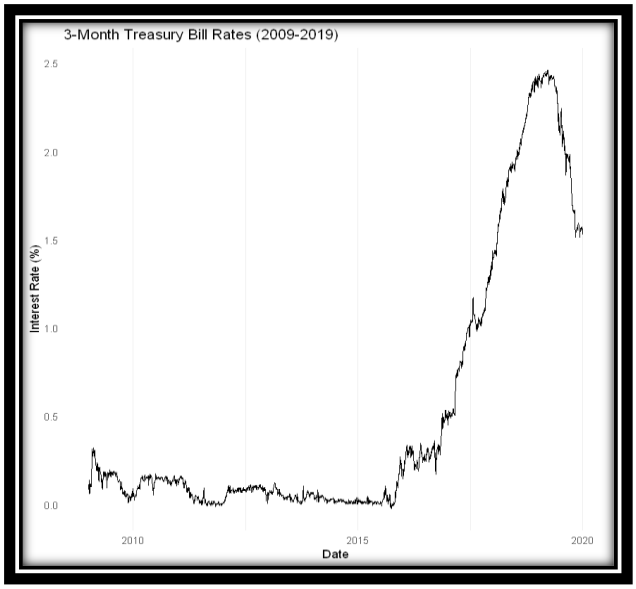


Figure 1: Risk Free Rate as estimated by U.S. 3-Month Treasury Bill Rate (Bloomberg)

While both the binomial model and the Monte Carlo algorithm could be adjusted to account for stochastic interest rates, to better approximate the values of the underlying options, we shall not do so here. It is not particularly relevant to the comparison of varying polynomials within the least squares regression and thus is beyond the scope of our immediate analysis. Nonetheless, this would be an important extension to apply for further analysis in the future, and would be imperative if the focus of this project were instead on interpreting the historical data. As a result, for the purpose of our testing, to estimate the risk-free rate of return for our model we will merely take the mean rate across the entire period. This will consider both the low interest rates from 2009 to 2015, and the rising rates following. Due to the higher frequency of low interests, as pictured below in **Figure 2**, this will have an overall higher weighting within the mean, bringing it down in value.

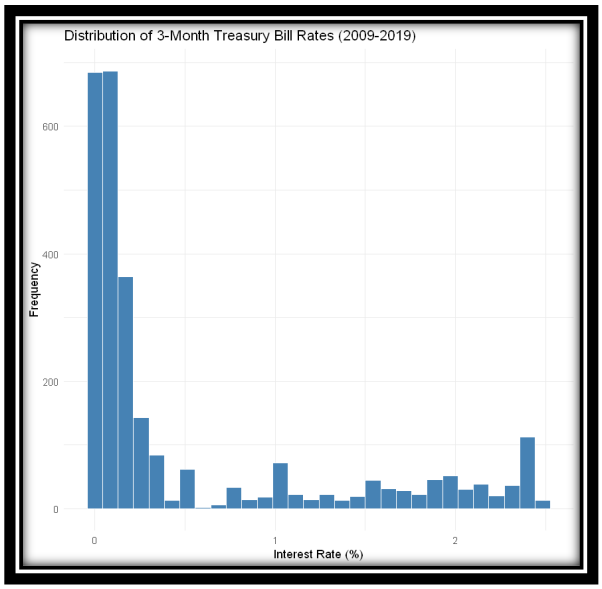


Figure 2: Frequency of Risk-Free Interest Rates over the 2009-2019 timeframe.

As a result, we calculated the average 3-month Treasury rate from 2009 to 2019 to be 0.5290473%. By consequence, our estimate for the risk-free interest rate in our model will also be this 0.5290473%. In our binomial model and random Brownian motion samples, we set parameter $\mathrm{r} = 0.005290473$ and we will make the assumption of continuous compounding.

S&P 500 Stock Index Data Collection

Secondly, in order to employ the binomial and Least Squares Monte Carlo models, we require an estimate for the volatility of the underlying stock, in our case the S&P 500 stock index, coupled with an initial value for the stock price, $S(0)$. As with the treasury bill rates, we extracted the required stock price data from Bloomberg, which presents a daily frequency recorded at the closing price. Prices were kept in terms of U.S. dollars (USD), in line with the American companies and the US Treasury Bill rates.

As observable in **Figure 3**, the index experienced significant and sustained growth over the timeframe. From 2009 to the end of 2019, the price of the S&P Index grew by over 200%. There is a clear deterministic trend within the data, which appears to be quite linear. While there are some noticeable dips within the market, they are all short-lived and the growth follows again subsequently. While this cannot be used to directly to extract the volatility, we can nonetheless explicitly read the initial price for our underlying asset: $S(0) = 931.8$.

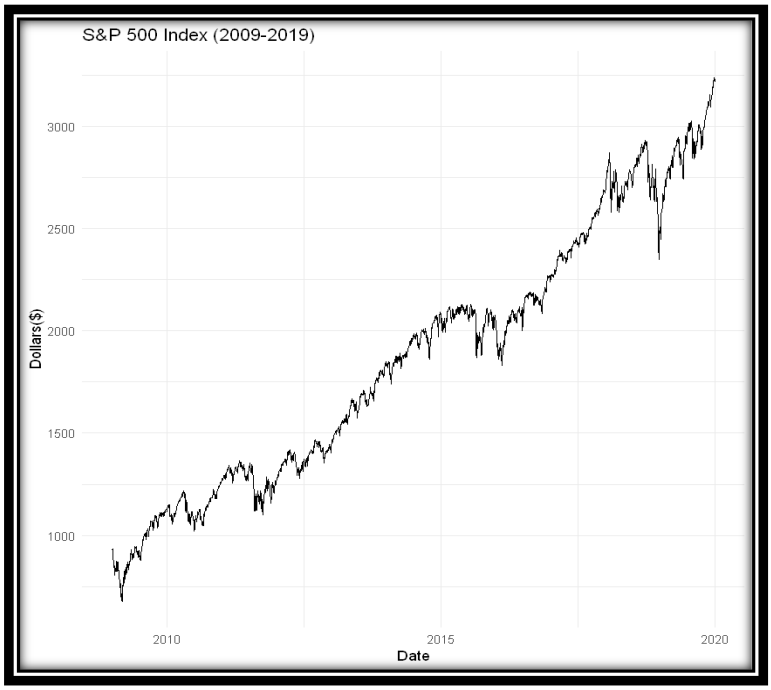


Figure 3: Growth of the S&P 500 Index from 2009 to 2019 (Bloomberg)

The stock price of the S&P 500 index can be modelled as a geometric Brownian motion of the form

$$S(t) = S(0)e^{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W(t)}$$

where $W(t)$ is a Brownian motion. Nonetheless, this $S(t)$ is not distributed as a martingale, unless $\mu = 0$. While we have calculated the value of $S(0)$, in particular we are looking for the volatility parameter σ. To proceed, we found the log-returns. The log-returns are defined as

$$\log returns = R_j = \log\left(\frac{S_{j+1}}{S_j}\right) = \sigma\left(W(t_{j+1}) - W(t_j)\right) + \left(\mu - \frac{\sigma^2}{2}\right)(t_{j+1} - t_j)$$

where t_{j+1} and t_j are sequential points within our partition of data from 2009 to 2019. Since we used a daily frequency, each t_j represents a different day. We also know that R_j should have a normal distribution $R_j \sim N\left(\left(\mu - \frac{\sigma^2}{2}\right)(t_{j+1} - t_j), \sigma^2(t_{j+1} - t_j)\right)$.

Using the log return formula defined above, we calculated the log returns of the S&P 500 index for each individual day within the timeframe, the results of which are displayed in **Figure 4**.

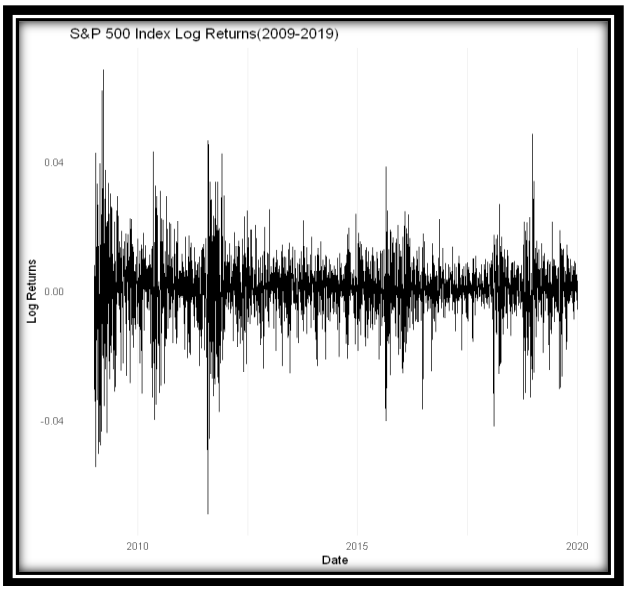


Figure 4: Log Returns of the S&P 500 Index from 2009 to 2019

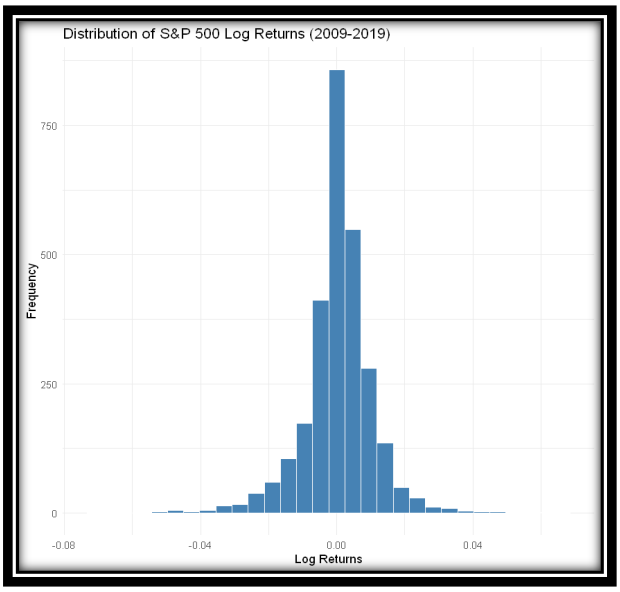


Figure 5: Histogram Distribution of S&P 500 Index Log Returns

Further, we stated the assumption above that the log returns should have a normal distribution; hence, we shall briefly verify this assumption with a histogram of the returns above in **Figure 5**. While not exact, as there appears to be a slight negative skew, the yearly log returns on the S&P 500 do in fact appear to have an approximate normal distribution. The mean also seems to be approximately 0 (it was calculated as 0.0004493533). Our goal was to find an estimate for the volatility parameter, σ, which now can simply be approximated using the standard deviation of the log-returns. Using R, we can conclude that the volatility of the S&P 500 index is approximately $\sigma = 0.01026789$.

Therefore, in summary, through the data for US Treasury Bills and the S&P 500 Stock Index, we have determined the following parameters:

- ✓ The risk-free rate: $\boldsymbol{r = 0.005290473}$
- ✓ The initial value of our underlying stock: $\boldsymbol{S(0) = 931.8}$
- ✓ The volatility of our underlying stock: $\boldsymbol{\sigma = 0.01026789}$

Binomial Tree Convergence

In order to have a basis for comparing the error within our results, we now estimated the binomial model for an At-The-Money (ATM) American put option. While it would be an interesting project to analyze the functionality of the LSMC algorithm with respect to varying degrees of options being In-The-Money or Out-of-The-Money, since that is not our focal point, we shall instead set our base case as the ATM option, in which the strike price and initial price of the stock are identical. This should provide us with a general overarching analysis of the effects of polynomials, their degrees, and the size of the number of paths in our Monte Carlo algorithm. Nonetheless, the Discussion section will briefly outline the effects this may have had on the eventual results. The goal is to increase the number of periods within the binomial model, so that this discrete model converges to the values of a continuous time model. Hence, the number of periods, denoted N, will be increased in size until we observe little variation in the value of the option. We coded a function to simulate such a binomial model, which can be found within **Appendix B**. Using the model parameters extracted from the S&P 500 data, we were able to estimate the parameters of the binomial model: the upturn factor u, the downturn factor d, and the risk neutral probability p. First, it is important to note that for the total time period of our binomial model we selected T=1. While we could have used T=11 for example, representing the 11 years for which we observed the data, given we will be increasing the number of time steps within the model it will not matter. Regardless of the value of T, whether it be 1 or 11, the length of each time step, given by the time length divided by the number of periods, will approach zero as the number of periods increases.

$$dt = \frac{T}{N} \longrightarrow 0 \; as \; N \longrightarrow \infty$$

Hence, a value of T=1 was selected for overall simplicity.

Next, within the binomial code, the upturn and downturn factors were calculated using the frequently employed

$$u = e^{\sigma\sqrt{dt}}$$

$$d = e^{-\sigma\sqrt{dt}}$$

while the risk-neutral probability of upturn and downturn were calculated as

$$p = \frac{e^{rdt} - d}{u - d}$$
$$q = 1 - p$$

At each step of the binomial tree, our code compares the present value of holding the option versus the payoff of immediately exercising the put option, which is given by $\max\{0, K - S(t)\}$.

For discounting, we assumed continuous compounding at the risk-free rate derived from the US Treasury Bill data, 0.5290473%. For full coding details see **Appendix B**.

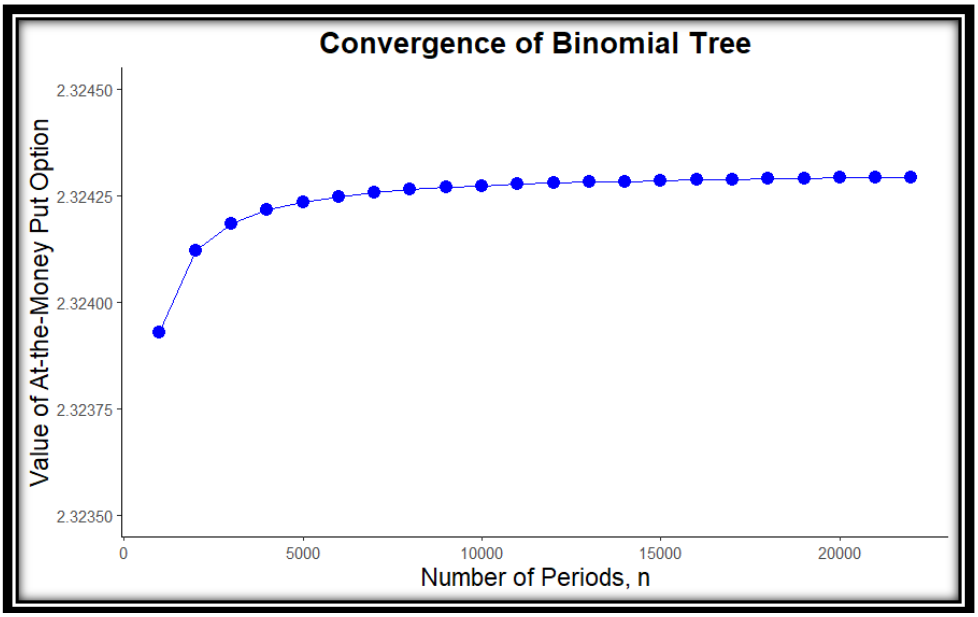


Figure 6: Binomial Convergence of ATM American Put Option

For our base value of the ATM American put option, we wanted an accuracy of 5 decimal points; therefore, we tested for convergence of the binomial with an increasing number of time steps, N, ranging from 1,000 to 22,000, until a precise value was reached. Further, within our function, the strike price, K, was set equal to the initial value of the S&P 500 Index, $S(0) = 931.8$. **Figure 6** above displays the convergence to 5 decimal points.

As the number of periods in the Binomial Model increases, the value of the ATM American put option evidently converges. Even after approximately 10,000 periods, we view almost negligible change. After the entire 22,000 periods have been added, the value converges to $P(0) = \$2.32429$. This can be thought of as the expected value of the present discounted option payoffs, and is also our desired benchmark with which we can compare the results from the Least Squares Monte Carlo algorithm.

It is important to note, from a logistical perspective, that the code for this binomial model took upwards of 5 hours to run when we included such a large number of periods. Therefore, in practice, such a method may not be feasible due to its high cost. It is for this reason that alternative methods, such as the Least Squares Monte Carlo method, would be employed due to their heightened efficiency.

In summary, we have determined that the "true" value of the At-The-Money American Put option on the S&P 500 Index, within our time frame, should be:

- ✓ $\boldsymbol{P(0) = \$2.32429}$

Coding the Least Squares Monte Carlo Algorithm

The entirety of the code for the Least Squares Monte Carlo Algorithm can be found within **Appendix C**. We have defined our function in terms of the model parameters displayed in **Table 1**.

Table 1: Least Squares Monte Carlo Model Parameters

Symbol (as written in code)	Description	Value
K	Strike Price	931.8 (At-The-Money)
M	Number of Possible Exercise Dates	50 (explained below)
N	Number of Simulations	Range: 1,000 –1,000,000
r	Risk-Free Interest Rate	0.005290473
S0	Initial Stock Price	931.8
sigma	Volatility of the Stock	0.01026789
polynomial	The Type of Polynomials Used in the Least Squares Regression	▪ Standard ▪ Laguerre ▪ Hermite ▪ European Options

The first step is to vectorize the N random Brownian motion simulations, each with M time steps, thus creating an NxM stock price matrix. In particular, we selected Brownian motions without any drift, of the form

$$S_n(t_m) = S_n(t_{m-1})e^{\sigma\sqrt{dt}Z_m}$$

where $Z \sim N(0,1)$ is a standard normal distribution and $1 \leq m \leq M$ is the column number of the stock price matrix. We created a total of N such paths, that is $1 \leq n \leq N$. In practice, random standard normal values were simulated to fill in the NxM stock price matrix. Then, the above formula was applied, commencing with $S_n(0)$ and employing a cumulative sum of the m random standard normal values within row n:

$$S_n(t_m) = S_n(0)e^{\sigma\sqrt{dt}[\sum_{j=1}^{m} Z_j]}$$

With the Brownian motions simulated and hence prices populating the stock price matrix, we created an MxN cash flow matrix. For the final row of the matrix, row M, we defined the payoffs for each row as $\max\{K - S_n(t_M), 0\}$. Working backwards one column at a time, X variables are set as the value of the stock prices in the $m-1^{th}$ column, while Y values are set as the one period discount of the positive cash flows in the m^{th} column, using the continuously compounded e^{-rdt} as the discount factor. For any entries of the matrix when the X value is higher than the strike price, $X > K$, it is replaced by an N/A, so that we do not apply the regression to these entries, but rather leave these entries in the $m-1^{th}$ column of the cash flow matrix as 0. Now we can apply the ordinary least squares linear regression for each column, inputting the polynomial variable of choice, depicted later in **Tables 2** and **4**, to provide the structure of the regressors X against the response variable Y.

We measure two values for each entry: 1) The value of immediate exercise, $K - S_n(t_{m-1})$ and 2) The continuation value. This continuation value is the estimated linear regression model evaluated at the stock price in the corresponding $S_n(t_{m-1})$ entry of the stock price matrix. The larger of these two values for each row is selected to be the entries in the $m-1^{th}$ column of

the cash flow matrix. This process is inducted backwards throughout the matrix until we reach the first column, $m = 1$.

The final section of the code involves the discounting procedure. The code ignores all zero entries beginning each row. Upon reaching the first non-zero entry, in column j, it is discounted back to time zero by e^{-rj}. Any further entries within this row will not be considered and are set to zero, ensuring that only the first non-zero entry in each row will be discounted. Since the option has already been exercised at time t_j, it will no longer have value in subsequent periods. Once all rows have a present discounted value, taking the mean over all N rows will yield the estimated value for the American Option.

With the Monte Carlo Code created, the next step was to determine the number of fixed time steps or possible exercise dates, M, within the LSMC algorithm. For testing polynomial types and degrees, coupled with number of simulations, the possible dates for exercise will be held constant. Since we were striving to emulate an American put option, which can be exercised at any point up to and at maturity, we initially tried using a large number of exercise dates, thinking this would be most accurate. Nonetheless, in doing so, we encountered many problems. In fact, with $M > 1{,}000$ the LSMC result was not converging even roughly to the base case binomial value. Instead, we will have to model a "Bermudan Put Option", which allows the holder to exercise the option on a discrete number of dates before maturity. **Figure 7** and **Figure 8** display the effects of varying the number of possible exercise dates on the price of the ATM put option and the degree of error relative to our binomial base case. It was tested with two to five degrees of our standard polynomial. Coded details can be found in **Appendix D**.

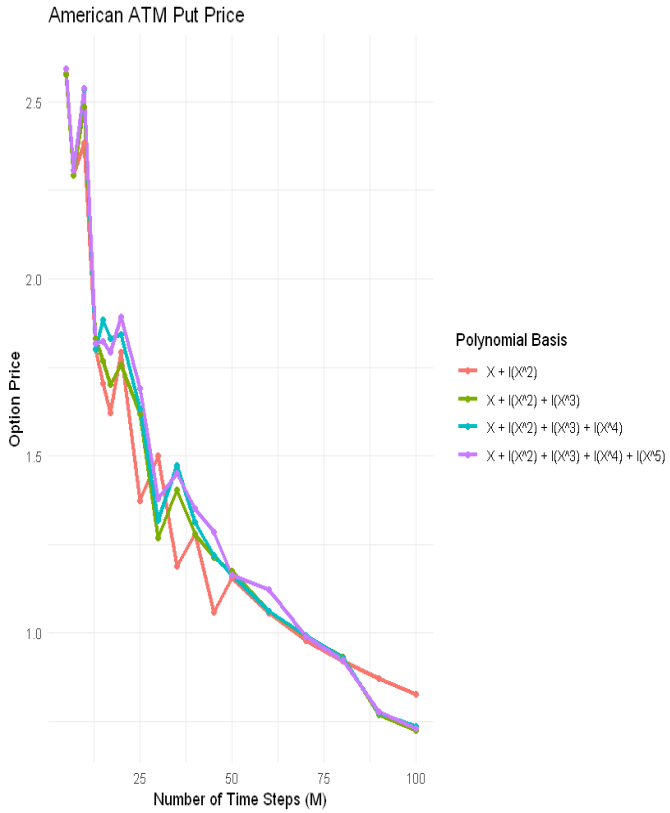


Figure 7: Effect of Possible Exercise Dates on the Price of the ATM Put Option

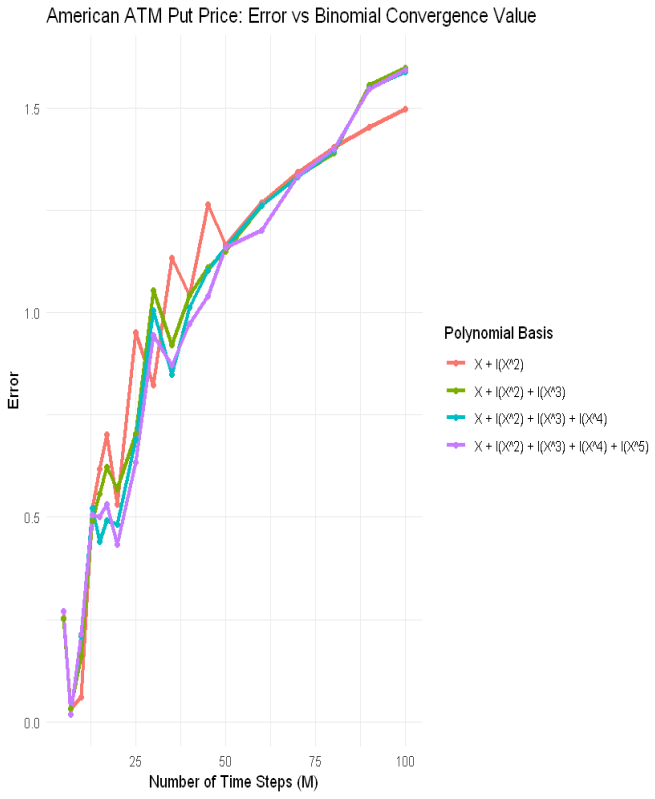


Figure 8: Effect of Possible Exercise Dates on the Error Relative to the Binomial Base Case

Evidently, as the number of possible exercise dates decreases, the value of the ATM put option begins to rise towards the value of $P(0) = \$2.32429$, and the value of the absolute error term begins to fall closer to 0. Across all the degrees of the polynomials, this result remains consistent, with the lower-degree polynomials having slightly larger errors than those with a higher degree. This result occurs due to the structure of the algorithm; the larger the number of possible exercise dates, the more regressions will be run. Each regression bears with it a degree of error, which will be accumulated with higher values of M. If M falls all the way to 1, then in fact we are no longer measuring an American or Bermudan option, but rather a European one. Here, there are actually no regressions taking place, which implies there will be no error and the value will converge most closely. Nonetheless, if we were to select $M = 1$ as our number of possible exercise dates, while that would yield precision in estimation, that would defeat the purpose of our experiment. We shall instead choose $M = 50$, primarily as the number of trading weeks throughout a year, but also because we only observe a moderate degree of error arising from this amount. Hence, rather than modelling a true American option, in technicality we are modelling a Bermudan put option with 50 possible exercise dates. Nonetheless, for the sake of this report, we can treat this as an estimation of the American put option value.

With the code designed, and the number of possible exercise dates fixed at $M = 50$, we are now able to test the impact of varying the *N* and *polynomial* parameters within our coded LSMC function. We shall set the variable parameters as follows:

1) As a **number of random simulations**, $\boldsymbol{N}$, we selected the set:
$$\{1{,}000;\ 5{,}000;\ 10{,}000;\ 50{,}000;\ 100{,}000;\ 500{,}000;\ 1{,}000{,}000\}$$
Small incremental changes are unlikely to bear significant effects to the convergence of the algorithm; hence, larger increments were chosen to capture the global trends. A logarithmic scale will be used to graphically illustrate the changes.
2) As a **choice of polynomials**, standard polynomials were employed as a base case. While not orthogonal polynomials themselves, in order to analyze whether more complex polynomials would indeed reduce the absolute error, it was imperative to begin with the most simplified base case possible. The performance of the other polynomials can thus be compared in relation to these. Laguerre and Hermite polynomials were selected as the orthogonal polynomials to be tested. All three are mentioned explicitly within the Longstaff and Schwartz paper, and said to “also give accurate results” (Longstaff & Schwartz, 123). Finally, as an extension, we will employ custom functions composed of European put and call values at each time step within the model.
3) As the **degree of the polynomials**, integers between 2 and 9, both odd and even, were selected for the standard polynomials and integers between 2 and 5 were selected for the orthogonal polynomials. Naturally, the degree of the polynomial, p, must be finite: $p < \infty$, and in order to accurately estimate the value of continuation, a degree of $p = 1$ would not suffice. Any larger degrees of a polynomial would be superfluous, and even in practice 9 degrees would be unlikely to be used. With the goal of the Monte Carlo algorithm being to employ an efficient method for valuing American options, parsimony should ultimately be strived for within the ordinary least squares regressions. It is important to note that while for the standard polynomials the degree refers explicitly to the highest power of the function polynomial function used as the regressor, for the Laguerre and Hermite polynomials, the degree refers to the number of orthogonal bases that will be used within the regression. That is, for a degree of 2, we would employ the zeroth, first, and second Laguerre or Hermite

polynomials as basis functions; for a degree of 3, we would employ the zeroth, first, second, and third Laguerre or Hermite polynomials as basis functions; and so forth. A complete list of the basis functions for the Laguerre and Hermite polynomials can be found in **Tables 3** and **4** below.

Results

We shall first graphically present the results for the three forms of polynomials we employed: standard polynomials, Laguerre Polynomials, and Hermite Polynomials. Required codes for the functions can be found in **Appendix E**, and the codes for the three-dimensional representations can be found in **Appendix F**. Subsequently, additional polynomials created using European options will be touched upon.

Standard Polynomials

Figures 9 and **10** display the results for the standard base polynomial of degrees ranging from 2 to 9, and simulated paths ranging from 1,000 to 1,000,000.

Table 2: The Eight Standard Polynomials Used as Regressors within the LSMC Algorithm

Degree	Equation of Regressor
2	$X + X^2$
3	$X + X^2 + X^3$
4	$X + X^2 + X^3 + X^4$
5	$X + X^2 + X^3 + X^4 + X^5$
6	$X + X^2 + X^3 + X^4 + X^5 + X^6$
7	$X + X^2 + X^3 + X^4 + X^5 + X^6 + X^7$
8	$X + X^2 + X^3 + X^4 + X^5 + X^6 + X^7 + X^8$
9	$X + X^2 + X^3 + X^4 + X^5 + X^6 + X^7 + X^8 + X^9$

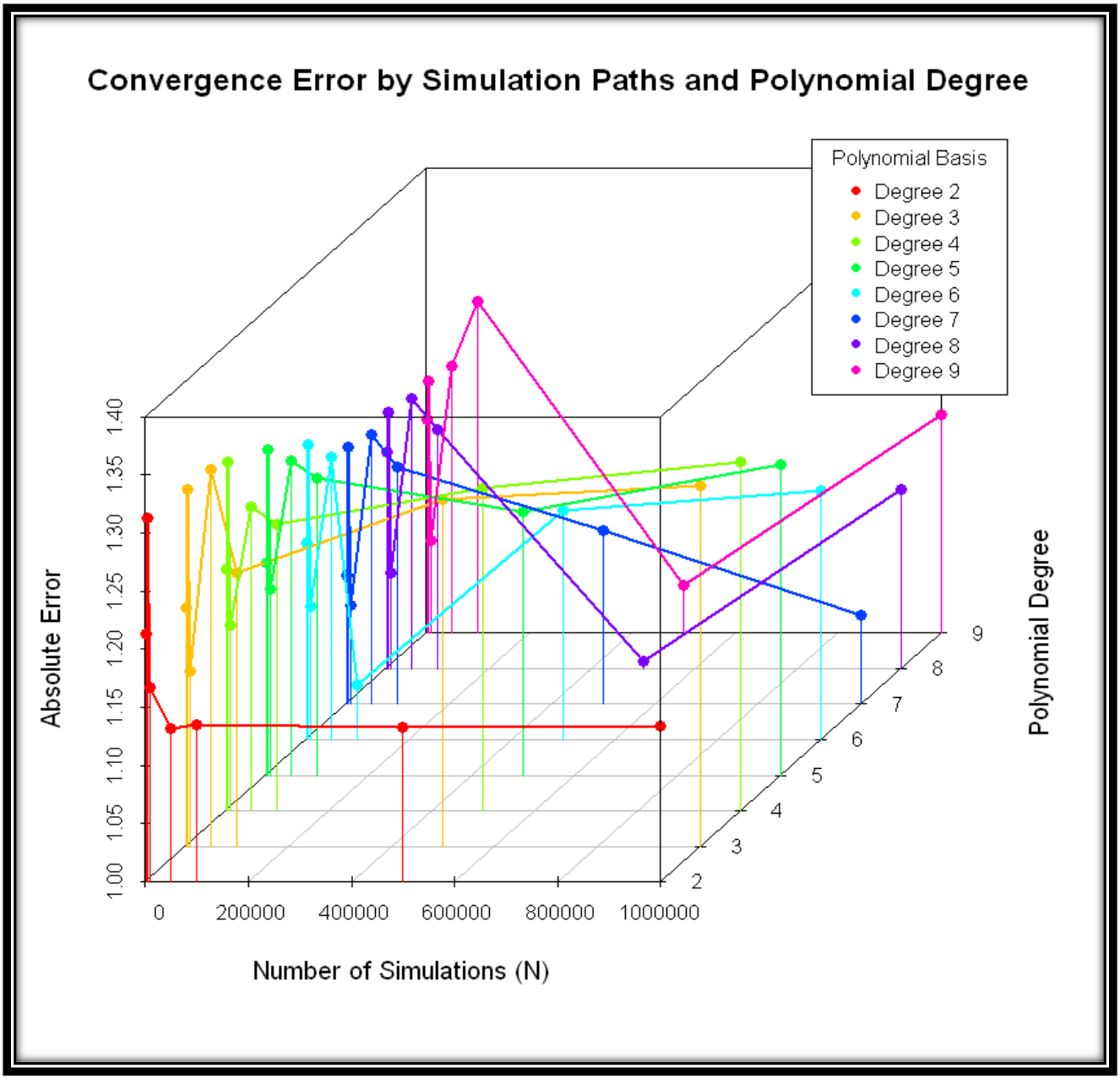


Figure 9: Standard Polynomial LSMC Convergence Error with Respect to the Number of Simulation Paths and the Degree of the Polynomials

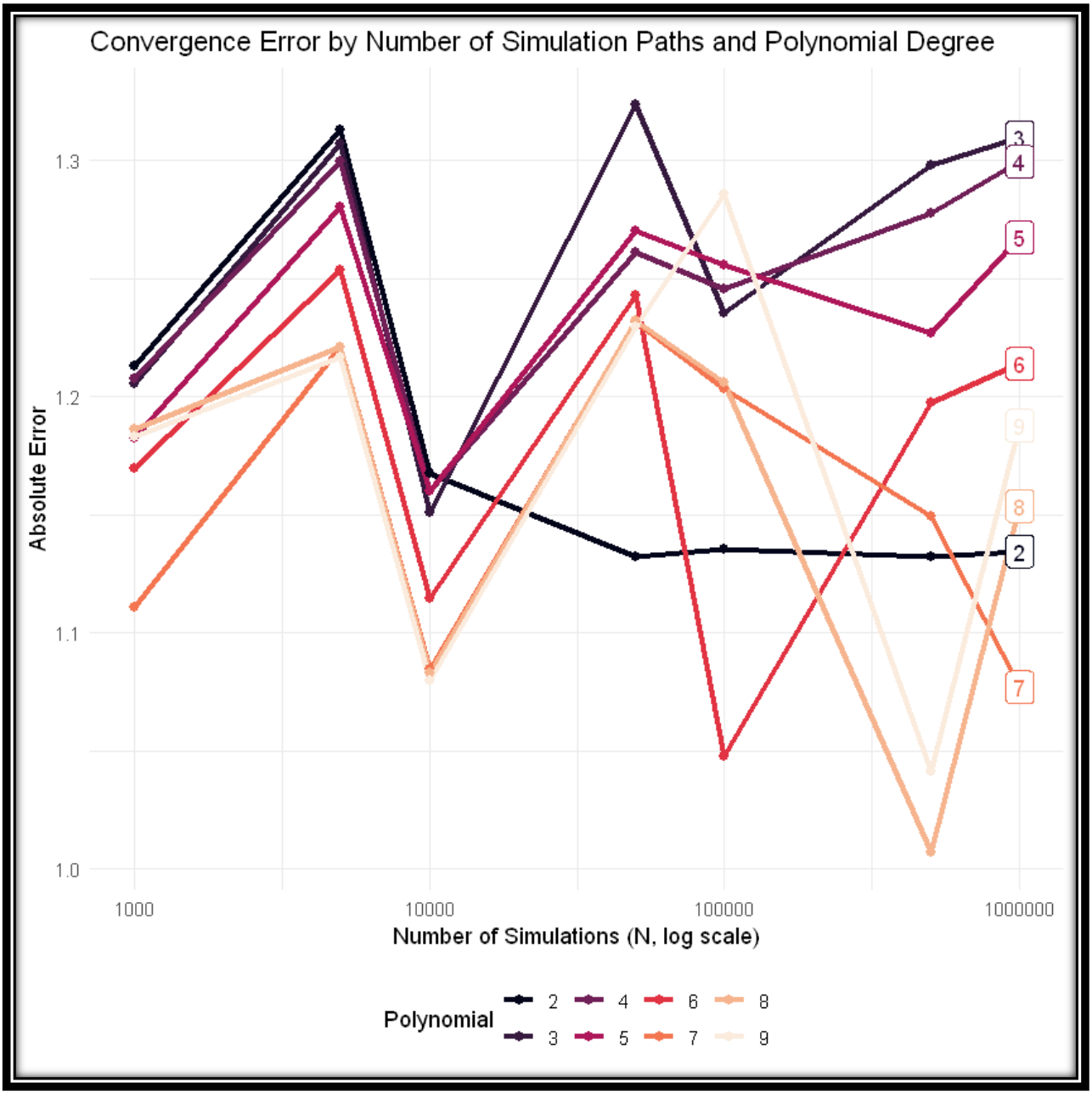


Figure 10: Standard Polynomial LSMC Convergence Error with Respect to the Number of Simulation Paths

Laguerre Polynomials

Figures 11 and **12** display the convergence of the LSMC algorithm, using Laguerre polynomials as the orthogonal bases for the regression. All degrees will use the basis elements for $n = 0,1$. As we increase the degrees from 2 to 5, we will add additional basis elements as regressors. For degree 5, all Hermite polynomials for $n = 0$ to $n = 5$ will be used as regressors. Thus, by Laguerre degree we are referring to basis elements up to and including the n^{th} Laguerre polynomial.

Table 3: The Laguerre Polynomials Used as Basis Functions Within the LSMC Algorithm

n	Laguerre Polynomial Basis Functions
0	1
1	$-X+1$
2	$\frac{1}{2}(X^2-4X+2)$
3	$\frac{1}{6}(-X^3+9X^2-18X+6)$
4	$\frac{1}{24}(X^4-16X^3+72X^2-96X+24)$
5	$\frac{1}{120}(-X^5+25X^4-200X^3+600X^2-600X+120)$

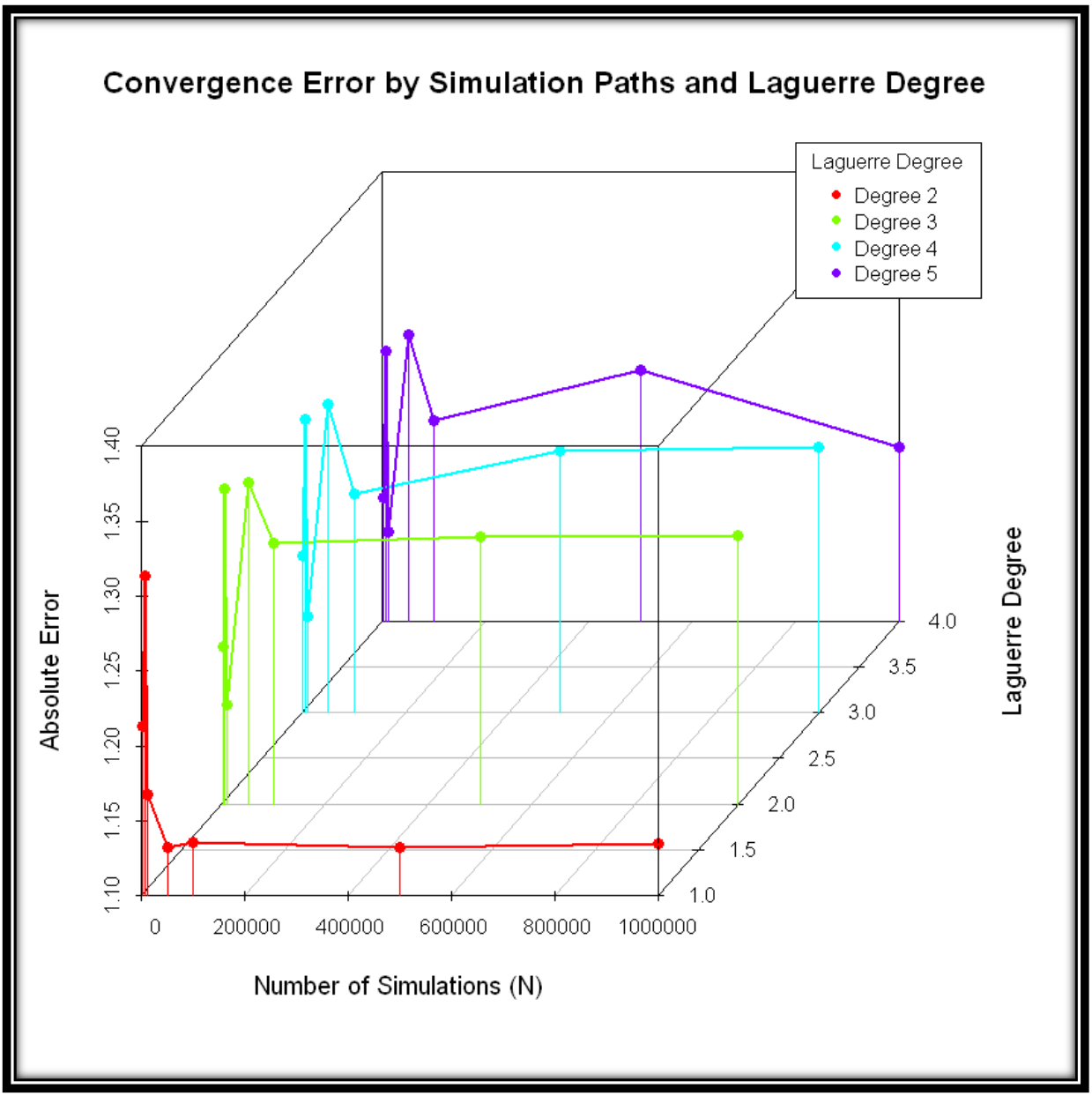


Figure 11: Laguerre Polynomial LSMC Convergence Error with Respect to the Number of Simulation Paths and the Laguerre Degree

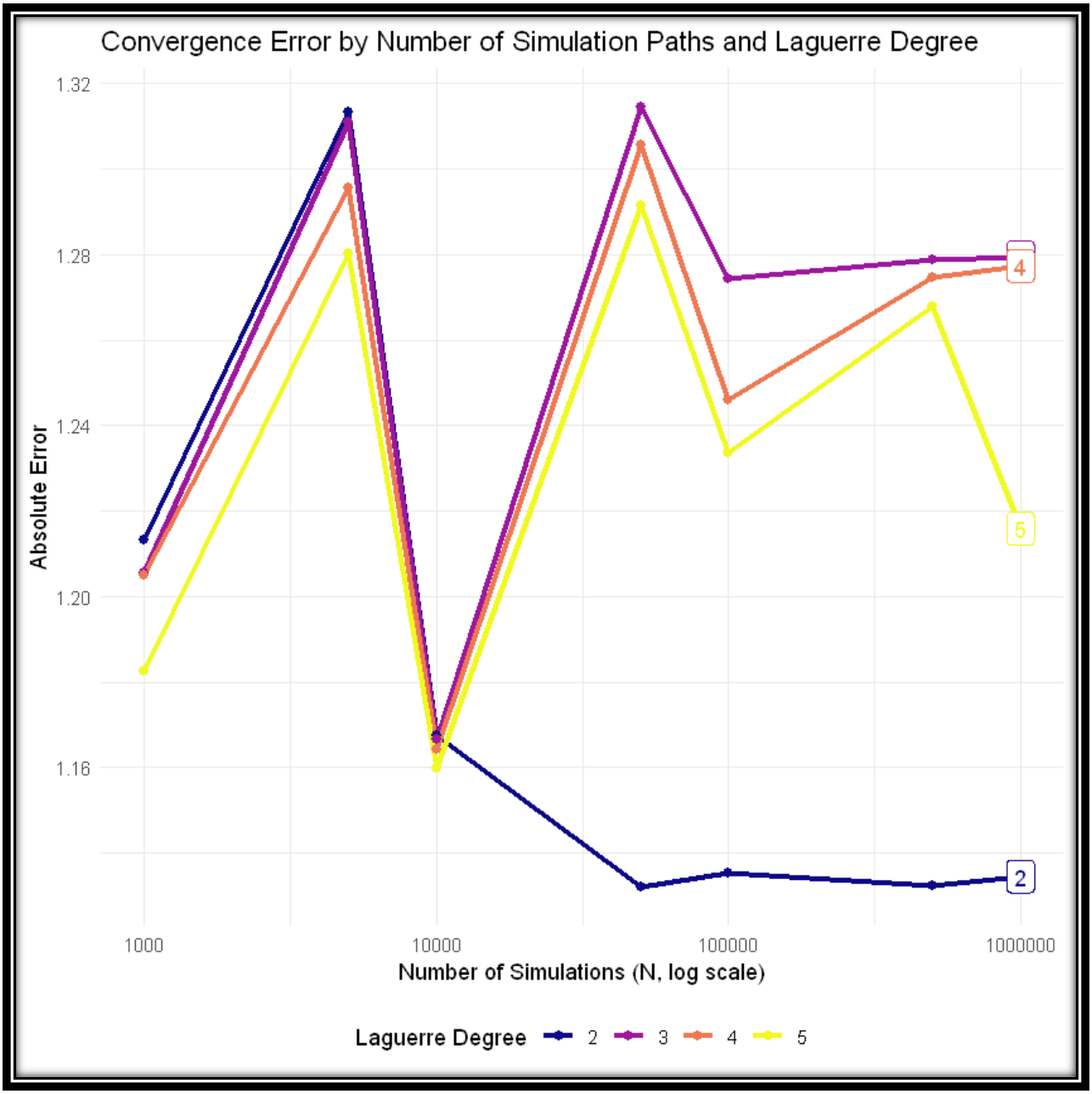


Figure 12: Laguerre Polynomial LSMC Convergence Error with Respect to the Number of Simulation Paths

Hermite Polynomials

Figures 13 and **14** display the convergence of the Least Squares Monte Carlo algorithm, using Hermite polynomials as the orthogonal bases for the regression. As with the Laguerre polynomials, all degrees will use the basis elements for $n = 0,1$. Again, in increasing the degrees from 2 to 5, we will add additional basis elements as regressors. For degree 5, all Hermite polynomials for $n = 0$ to $n = 5$ will be used as regressors. Thus, by Hermite degree we are referring to basis elements up to and including the n^{th} Hermite polynomial.

Table 4: The Hermite Polynomials Used as Basis Functions Within the LSMC Algorithm

n	Hermite Polynomial Basis Functions
0	1
1	X
2	$X^2 - 1$
3	$X^3 - 3X$
4	$X^4 - 6X^2 + 3$
5	$X^5 - 10X^3 + 15X$

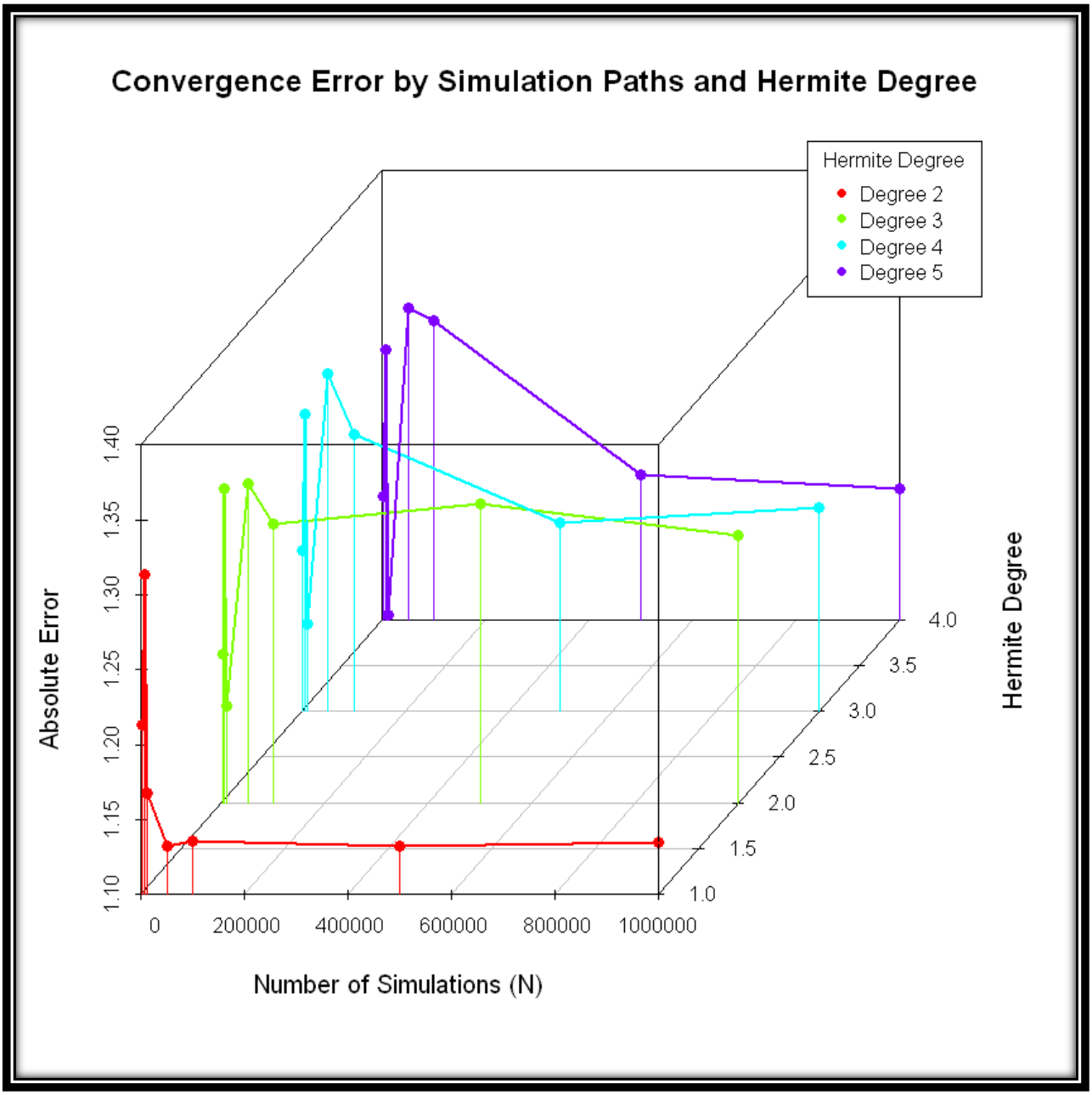


Figure 13: Hermite Polynomial LSMC Convergence Error with Respect to the Number of Simulation Paths and the Hermite Degree

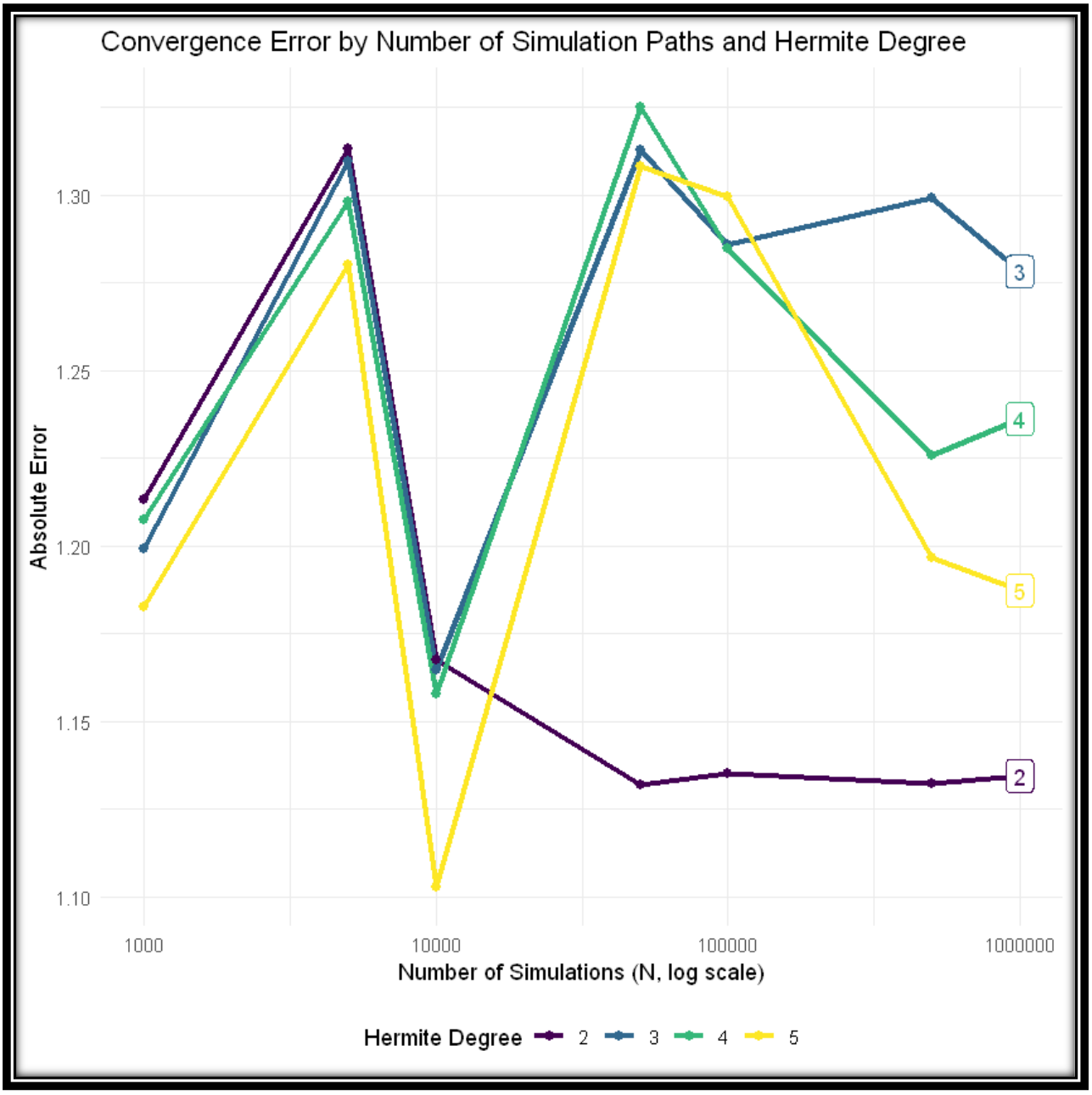


Figure 14: Hermite Polynomial LSMC Convergence Error with Respect to the Number of Simulation Paths

Functions of European Put and Call Values

As an additional extension beyond the standard and orthogonal polynomials, we wish to test polynomials that are functions of the prices of European Put and Call values. While this may be more involved than the previous polynomials, we are able to employ the Black and Scholes formula which is efficient to run. The types of polynomials can be chosen somewhat arbitrarily, as there is a vast number of possibilities; we shall take four, displayed in **Table 5** below. Just as with the other polynomials, we begin with second-degree polynomials. For further information on the coding procedure, please refer to **Appendix G**.

Table 5: Functions of European Option Values Used as Regressors in the LSMC Algorithm

Notation	Functions of European Put and Call Values
put_call_quad	$S + C + P + S^2 + C^2 + P^2$
put_only	$S + P + S^2 + P^2$
put_call_cubic	$S + C + P + S^2 + C^2 + P^2 + C^3 + P^3$
put_call_quad_int	$S + C + P + S^2 + C^2 + P^2 + S(C) + S(P)$

Here, S represents the stock price at a particular time step within the LSMC Algorithm, as determined by the random Brownian simulations. In fact, it acts identically to the X variables within the standard polynomial regressors, since those assumed the value of the underlying stock price at each step as well. C and P represent the value of the American call and put options, respectively, as calculated by the Black and Scholes formula:

$$C(S,T) = SN(d_1) - Ke^{-rT}N(d_2).$$

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

Almost all parameters in their calculation will be identical to those we are testing for the American options: K, r, σ. These will also remain constant throughout all regressions, since we assumed both volatility and interest rate are not stochastic. Of course, given the Black and Scholes formula also explicitly depends on the stock price at the time of valuation, S, the values of C and P will depend on the randomly simulated Brownian paths at each step of the algorithm. The one discrepancy to note is that while the maturity of the European options will still be the last possible exercise date of the American option, T will now be the time remaining before maturity, given by $(M - m + 1)dt$. Naturally, this will change depending on at which m of the $M = 50$ possible exercise dates for the American option we are valuing the European option. Hence, for each individual regression within the algorithm, the values of the European puts and calls will be different, adjusting to the time to maturity and the underlying stock price at each step.

Convergence to the Binomial base case was conducted in a congruent manner to other polynomials, with all parameters of our Least Squares Monte Carlo algorithm remaining the same, and the polynomial parameter being replaced by our functions of European options. **Figure 15** displays the results compared to the second-degree standard polynomial.

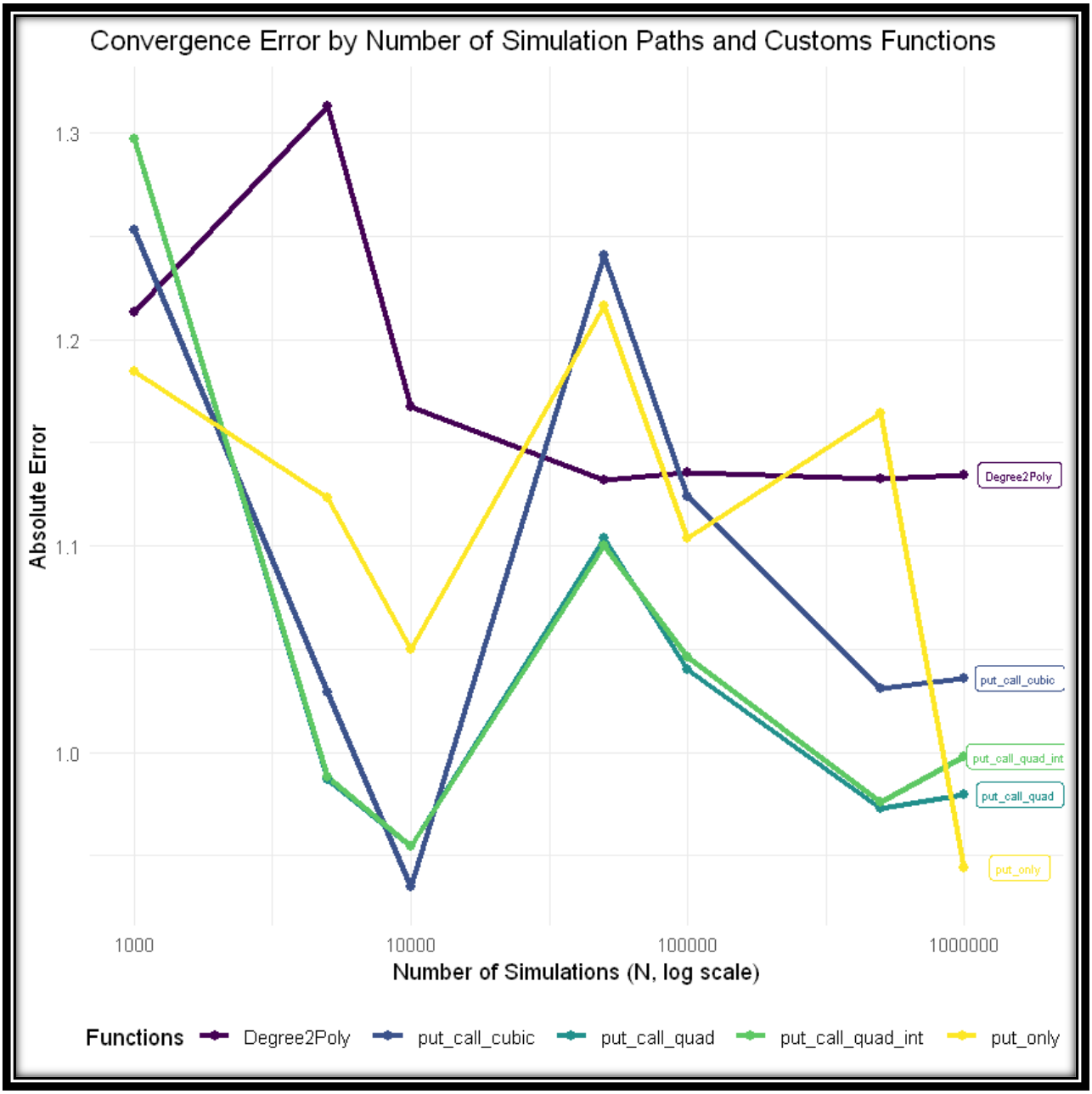


Figure 15: Functions of European Options LSMC Convergence Error with Respect to the Number of Simulation Paths

Discussion

Immediately, it is important to note that across all simulations there was a relatively high convergence error. Amongst all of the standard, Laguerre, and Hermite polynomial degrees, the smallest error was approximately 1.10, signifying no precise convergence was observed. A potential reason we have flagged for this is the base model being set At-The-Money. While these should display convergence nonetheless, given that approximately half of the Brownian

simulations will end Out-of-The-Money, by consequence, only around half of the regressors will actually be run. The fewer regressors within our algorithm, the less precise the estimation of the continuation value. **Figure 16** below displays the relative convergence error of the standard polynomial of degree two, in relation to various strike prices. Note, as the strike prices are raised above the initial value of the stock, $S(0) = 931.8$, from $K = 935$ to $K = 980$, the American put option becomes increasingly deeper In-The-Money.

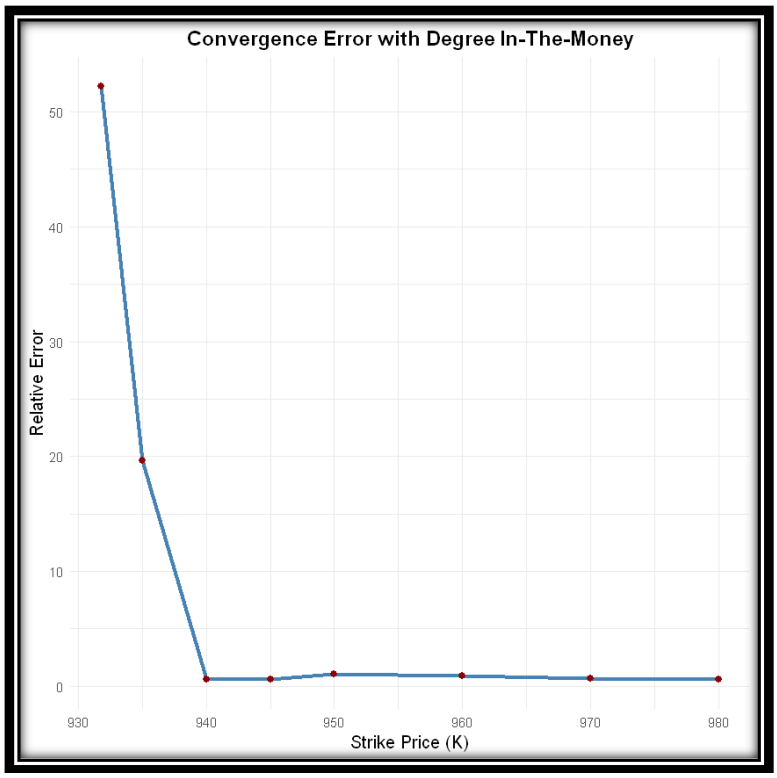


Figure 16: Convergence of Second-degree Standard Polynomial in Least Squares Monte Carlo Algorithm with Respect to Various Strike Prices

Relative error was calculated as follows:

$$Relative\ Error = \left|\frac{Binomial\ Value - Monte\ Carlo\ Value}{Binomial\ Value}\right| (100\%)$$

Evidently as the American Put option becomes deeper In-The-Money, its convergence error relative to the binomial model approaches 0. In fact, for a strike price of $K = 980$, the LSMC algorithm almost converges identically to the value of the underlying binomial model. Hence, by simulating all of our models ATM this may partially explain why we have observed such imprecise convergence overall. Furthermore, formerly presented **Figures 7** and **8** also provide a possible explanation for the heightened errors observed; had we reduced the number of possible exercise dates from $M = 50$ to either $M = 20$ or $M = 10$, again we would have likely observed our Monte Carlo simulated American put option converging more closely to the underlying Binomial Model.

Despite the large margin of error, we will nonetheless examine possible trends within the number of simulations, N, the degree of the polynomials, and the choice of polynomials.

Number of Simulations

The observed relationship between absolute error and number of simulation paths was not as clear as initially expected. While for the most part we do in fact see a negative correlation between the two, in which the error decreases with the number of simulated paths, or congruently the convergence to the binomial model is more precise, this evidently did not hold for certain larger values of N. Notably, for $N = 5{,}000;\ N = 50{,}000$, across all the standard, Laguerre, and Hermite polynomials, and all their degrees other than 2, the relative errors experience significant and seemingly inexplicable surges. For the Laguerre and Hermite polynomials, in addition to our custom European option polynomials, the $N = 10{,}000$ noticeably produced the strongest convergence to the binomial baseline overall. This perhaps suggests that running a copious quantity of paths did not necessarily produce better results in our simulations. Further, the change in error between $N = 500{,}000$ and $N = 1{,}000{,}000$ values was not as pronounced as those for $N < 100{,}000$. While initially this may seem as if for larger values of N the LSMC algorithm will converge to a value, which we know to be true, this alone does not provide sufficient evidence in favour of this. The only unmistakable trend that emerged concerning the number of simulation paths was in relation to the second-degree polynomials. Across all types of second-degree polynomials, the relative error peaked at $N = 5000$; however, thereafter steadily decreased until $N = 1{,}000{,}000$, with little variation in error for $N > 50{,}000$. Only with these second-degree polynomials was a clearer convergence visible in correlation with the number of simulated paths in the LSMC algorithm. We cannot draw any substantial claims with regard to the number of simulated paths, due to the large number of fluctuations in error. Perhaps for values of $N < 1000$, the errors would be even larger, but beginning at $N = 1000$ and increasing values thereafter, did not directly seem to affect convergence to the binomial model. While ambiguous, it would seem as if the fluctuations in error were not necessarily directly caused by the number of simulations, but rather by other factors at play within the model. Perhaps, for example, the set seed used in the code had particular extreme values for the $N = 5{,}000$ simulation; however, one would think this number of simulations would nullify such effects.

Degrees of Polynomials

The degree of the polynomial displayed much clearer results than the number of simulated paths. Among all choices of polynomials, second-degree polynomials produced the most accurate convergence, followed next by the highest degree simulated (whether that be nine of five), and then a decreasing order of the following degrees thereafter. By result, the third-degree polynomials produced the largest errors on all occasions, followed by the fourth-degree polynomials, and so forth. This same relationship amongst the degrees was observed across all numbers of simulated paths, for N, as discussed above. The slightly smaller errors with higher degree polynomials make intuitive sense, given that such regressions allow for more closely fitted values. Evidently here, this effect overweighed the additional errors that may have arisen from overfitting. The sole noteworthy deviations occurred within the standard polynomials, in which for $N = 100{,}000$ the sixth-degree polynomial outperformed all other polynomials, and for $N = 1{,}000{,}000$, the seventh-degree polynomial outperformed all other polynomials, in both cases by quite a significant margin. In fact, along with the eighth- and ninth-degree polynomials

for $N = 500{,}000$, they are the only polynomials of any degree, across all choices of polynomials, to have even outperformed the second-degree polynomial for $N \geq 100{,}000$. To some extent, this could thus be viewed as an outlier case, although perhaps this too may occur for larger bases of the Laguerre and Hermite polynomials as well, if we allowed for such. Nonetheless, in general, it is substantial to claim that for a large number of simulation paths, it is in fact the second-degree polynomial that has the most accurate convergence. While the higher degrees for the standard polynomial may have slightly less error, for parsimony of the regression, the second-degree would be preferred. This reduces the risk of overfitting, and would be particularly useful within the standard polynomials which themselves are not orthogonal. It is also important to note that the magnitude of the differences in the convergence was not significant, and even the lowest errors found in the second-degree polynomial were statistically significant from the true value of the American option. Hence, the convergence of the degrees was primarily evaluated relative to each other.

Choice of Polynomial

Again, no clear conclusion presented itself with regard to whether one choice of the standard, Laguerre, or Hermite polynomials was noticeably more effective than the others. The general error band, relative to the binomial convergence, was from $1.1 - 1.32$ for all three choices. The second-degree polynomials for each choice of polynomial, as aforementioned, were generally the most accurate, with errors around 1.13. In fact, using the second-degree standard, Laguerre, and Hermite polynomials yielded identical results. This intuitively makes sense provided that the Laguerre and Hermite polynomials use fewer number of basis functions in the second-degree and hence are not vastly different than the standard polynomial being used as the regressor. Because we tested a large number of degrees for the standard polynomial, some of the higher degrees outperformed when we employed a large number of simulated paths, but even so, the reduction in error was by no means significant in magnitude. Rather than discerning whether one polynomial was superior compared to the others, we would reach the conclusion that the polynomial choice within the regression is not a significant contributing factor to the error in convergence. For this reason, the standard polynomials, which were the simplest to code, and are conceptually the most straightforward, seem to produce equally valid results. While from a mathematical standpoint, the orthogonal polynomials are optimal and should in theory produce the best curve fitting, as discussed within the Background, the results of our simulations suggest that there may in fact not be a large discrepancy between them.

Nonetheless, the most promising results of convergence were observed within our custom functions composed of the European Put and Call values, which nearly all consistently outperformed the standard second-degree polynomial. Overall, these almost congruently followed the aforementioned trends in the number of simulation paths and degrees, but with slightly more accurate convergence. The European put and call quadratic polynomials, both with and without interactions, were the most reliable of all, continuously having the lowest convergence errors. While having more volatility across different values of N, the put-only function yielded the lowest convergence error when $N = 1{,}000{,}000$, and surprisingly the put and call cubic function when $N = 10{,}000$ yielded the singular lowest convergence error of all simulations. While we may not have tested as large of degrees of these custom functions, this would be an important area for future analysis, given the promising results. The largest concern however, is the complexity of the required function. Coding these custom functions required a noticeably more complex model and demanded further code to be incorporated. Hence, we

would need to analyze the trade-off between slightly more accurate convergence and significantly higher complexity.

Conclusion

Upon designing the project, the expectation was that we would observe slight, albeit apparent, impacts of varying the number of simulated Brownian paths and altering the choice and degree of polynomials being employed within the least squares regression. Nonetheless, no conclusions were so unambiguous, and in practice only a few significant trends could be extracted. Among all simulations of the At-The-Money American put option using the Least Squares Monte Carlo algorithm, the errors relative to our binomial base case were still relatively substantial. It would be an overstatement to claim that precise convergence was observed across all parameters and models employed. As noted, had we selected an unrealistically small value of the number of possible exercise dates, M, or simulated an option that was very deep in the money, $K \gg 931.8$, it is likely that the convergence as a whole would have been significantly more precise. By setting $M = 50$ and $K = 931.8$ as our base model, a noticeable pattern emerged with regard to convergence and polynomial degree. Second-degree polynomials generally had the smallest relative error, followed by the largest degree we tested, 9 for standard polynomials and 5 for orthogonal polynomials. The particular choice of standard, Laguerre, and Hermite polynomials to be used in the least squares regressions had a negligible impact on the overall convergence error. Nonetheless, upon testing custom polynomials in which the values of European calls and puts were employed as regressors, a clear reduction of error was visible relative to the orthogonal polynomials. Most perplexing of all was the observed relation between error and the number of randomly simulated binomial paths, which was highly volatile. Certain arbitrary values of N, such as $5{,}000$ and $50{,}000$ yielded large spikes in error, while $1{,}000$ and $10{,}000$ were much lower. While there was an eventual underlying negative relation, particularly among the larger number of simulations, it was not clearly defined.

At least exclusively from our results, it would seem as if second-degree polynomials with a large number of simulation paths, such as $N \geq 100{,}000$, has produced the most accurate convergence relative to our binomial base case, coupled with those of the highest degrees. Since parsimony is often strived for in regression, higher-degree polynomials would potentially be unnecessary. The choice of polynomial does not seem to have a significant impact, hence standard polynomials could be chosen for their simplicity or alternatively Laguerre or Hermite bases could be chosen for their orthogonality. If one is willing to further sacrifice a degree of simplicity, functions of European options seem to provide the most accurate results in absolute terms. Thus, it should be at the discretion of the researcher, and an individual cost-benefit analysis, as to whether they would assume a more complex model, including higher costs to run, for marginally more accurate results.

Areas for Extension

There are a plethora of areas that would have been fascinating to explore further. Regrettably, given the time constraints, for the sake of overall cohesion within the project, we were unable to do so.

- Stochastic interest rates and volatility: As noted within the methodology, interest rate hikes were observed late in 2016. For simplicity of our model, we merely took the mean interest rate across the 11-year timeframe; nonetheless, in practice, it would be possible to incorporate a stochastic interest rate into both the binomial model and the LSMC algorithm. For the latter, discount factors between each column of our cash flow matrix could vary. A similar technique could be applied for the volatility of the S&P 500 stock index, which would modify the stock price index within our code. Both would have significantly complicated our model, but would nonetheless provide a more accurate representation of true market conditions.
- Testing American options that are varying degrees In- and Out-of-the-Money: While there remained a substantial margin of error for all simulations of the ATM American put option, from the brief tests we ran in **Figure 16**, we hypothesize that the deeper In-the-Money we set the exercise price, the more accurate the convergence of the Least Squares Monte algorithm may be. Given that we only run regressions on X values that are ITM, if we have already begun ITM then there is a larger probability that more of the randomly simulated paths will end ITM as well. Hence, we would be running linear regressions with a greater number of sample observations, which should yield more accurate results. The opposite holds for options OTM, which would have smaller sample sizes being used within the regressions, and resultingly a more significant margin of error. While we ran a few preliminary tests of this with our code, it was not something we went too in-depth with. It would be fascinating to test whether ITM or OTM options change the trends we observed in polynomials and their degrees. That is, would being ITM affect how quickly the LSMC converges?
- Analyzing the efficiency of the code: The primary metric for analysis we employed throughout the paper is that of the convergence in absolute terms relative to the binomial base case. Of course this is essential; however, as aforementioned with regards to the cost-benefit analysis, in practice researchers also seek to render the process more efficient. If this were not a consideration, then the binomial model, despite taking hours to run, would in fact be optimal. Hence, another important metric would be that of time to run each code. In further extension, we could test how much more efficient running the code is at the various number of simulation paths, at the various degrees, and even for the polynomials used. If, for example, the functions of European Options do not require much more time to run, then it is likely that this could be the most effective method. Therefore, while our metric was a solid foundation, the true effectiveness of these models would be discerned through a comparison of absolute error together with operational efficiency. Further, other methods to improve efficiency and convergence could be explored, such as important sampling or bootstrapping techniques.
- Further analysis of functions of European Options: While this was a topic touched upon at the end of our results, given the lower errors it produced there is a great deal more to explore on the topic, whether this be increasing the degrees, examining the difference between Put-only or Call-only functions, and so forth. While there exists much literature on the classic orthogonal polynomials, the functions of European options seem to be much less explored.

References

Arfken, G. (1985). "Hermite Functions." §13.1 in Mathematical Methods for Physicists, 3rd ed. Orlando, FL: Academic Press, pp. 712-721.

Bloomberg L.P. (2025). *Stock Price Graph for S&P 500. 01/01/2009 to 12/31/2019*. Retrieved March 24, 2025 from Bloomberg terminal.

Bloomberg L.P. (2025). *U.S. 3-Month Treasury Bill Rate Graph. 01/01/2009 to 12/31/2019*. Retrieved March 24, 2025 from Bloomberg terminal.

Elliot, R. J., & van der Hoek, J. (2006). 2.1 The Basic Model. In *Binomial Models in Finance* (pp. 13-18). essay, Springer.

Ikekwere, E. (2023). Orthogonal Polynomials: Essential Tools and Applications in Modern Mathematics. Journal of Science Technology and Education, 11(3).

Moreno, M., & Navas, J. F. (2003). *On the robustness of least-squares Monte Carlo (LSM) for pricing American derivatives*. Review of Derivatives Research, 6(2), 107–128. https://doi.org/10.1023/A:1027340210935

Longstaff, F. A., & Schwartz, E. S. (2001). Valuing American Options by Simulation: A Simple Least Squares Approach. *The Review of Financial Studies, 14*(1), 113-147. https://doi.org/10.1093/rfs/14.1.113

Narula, S. C. (1979). Orthogonal Polynomial Regression. *International Statistical Review*, 47(1), 31-36. https://doi.org/10.2307/1403204

Kenton, W. (2024, June 12). *S&P 500 index: What it's for and why it's important in investing*. Investopedia. https://www.investopedia.com/terms/s/sp500.asp

Stentoft, L. (2004). Convergence of the least squares Monte Carlo approach to American option valuation. *Management Science, 50*(9), 1193–1203. https://www.jstor.org/stable/30046227

The New York Times. (2018). *How the Fed's interest rate increases affect you*. Retrieved from https://www.nytimes.com/interactive/2018/business/economy/fed-rates-powell.html

Wei, W., & Zhu, D. (2022). Generic improvements to least squares Monte Carlo methods with applications to optimal stopping problems. *European Journal of Operational Research, 298*(3), 1132–1144. https://doi.org/10.1016/j.ejor.2021.08.016

Zhao, J. (2018). American option valuation methods. International Journal of Economics and Finance, 10(5).

RCode For Monte Carlo Least Squares

2025-04-17

(Appendix A) Estimation of Parameters

Estimation of Parameters Notebook Github

(Appendix B) Binomial Model Code

Function to price an American put using a Binomial Tree

```
binomial_tree_american_put <- function(S0, K, T, r, sigma, M) {
  dt <- T / M  #Time Step Size
  u <- exp(sigma * sqrt(dt)) # Upturn factor
  d <- exp(-sigma * sqrt(dt)) # Downward factor
  p <- (exp(r * dt) - d) / (u - d) # Risk-neutral probability

  # Stock price tree
  stock_tree <- matrix(0, nrow = M + 1, ncol = M + 1) # Create matrix to store stock prices
  for (j in 0:M) {
    for (i in 0:j) {
      stock_tree[i + 1, j + 1] <- S0 * u^i * d^(j - i)
      # Number of steps multiplied by upturn and downturn factors
    }
  }
  # Option value tree
  option_tree <- matrix(0, nrow = M + 1, ncol = M + 1) # Matrix to store option values
  option_tree[, M + 1] <- pmax(K - stock_tree[, M + 1], 0)
  # Payoff is max(K - S, 0) at maturity, applied to entire column M+1

  # Work backward through the tree
  for (j in (M - 1):0) { # Go through each column in reverse order starting at M-1
    for (i in 0:j) { # Go though each row in the column
      continuation <- exp(-r*dt)*(p*option_tree[i+2,j+2]+(1-p)*option_tree[i+1,j+2])
      exercise <- pmax(K - stock_tree[i + 1, j + 1], 0)
      option_tree[i + 1, j + 1] <- pmax(continuation, exercise)
    }
  }

  return(option_tree[1, 1])  # Option price at root
}
```

(Appendix C) Monte Carlo Algorithm

Function to price an American put using Monte Carlo simulations

```
price_american_put_longstaff_schwartz_MC <- function(K, M, N, r, S0,sigma, polynomial) {

  dt <- 1/M
  discount <- exp(-r * dt)
  set.seed(123)
  Z <- matrix(rnorm(N * M), nrow = N, ncol = M) # Vectorize Brownian Motion Simulation
  S <- S0 * exp(sigma * sqrt(dt) * t(apply(Z, 1, cumsum)))

  Cash_flow <- matrix(0, nrow = N, ncol = M)
  Cash_flow[, M] <- pmax(K - S[, M], 0)

  # Cash Flows at each time step
  for (m in M:2) {
    X <- S[, m-1]
    Y <- Cash_flow[, m] * discount
    XY <- cbind(X, Y)
    XY[X > K, ] <- NA

    if (all(is.na(XY))) {
        Cash_flow[, m-1] <- 0  # Skip regression if no in the money paths
        next
    }
    regression <- lm(polynomial, data = as.data.frame(XY))
    immediate_exercise <- pmax(K - S[, m-1], 0)
    continuation <- predict(regression, newdata = as.data.frame(X))
    Cash_flow[, m-1] <- ifelse(continuation < immediate_exercise, immediate_exercise, 0)
  }

  # Discounting cash flows
  for (i in 1:nrow(Cash_flow)) {
    for (j in 1:ncol(Cash_flow)) {
      if (Cash_flow[i, j] != 0) {
        Cash_flow[i, j] <- Cash_flow[i, j] * round(exp(-r * j), 5)
        if (j < ncol(Cash_flow)) {
          Cash_flow[i, (j+1):ncol(Cash_flow)] <- 0
        }
        break
      }
    }
  }

  option_price <- mean(rowSums(Cash_flow))
  return(option_price)
}
```

(Appendix D) Effect of Time Steps M

Effect of Time Steps M Notebook Github

(Appendix E) Monte Carlo Algorithm using Laguerre and Hermite polynomials

Function to price an American put using Laguerre polynomials

```
price_american_put_Laguerre <- function(K, M, N, r, S0, sigma, degree) {

  dt <- 1/M
  discount <- exp(-r * dt)
  set.seed(123)
  Z <- matrix(rnorm(N * M), nrow = N, ncol = M)
  S <- S0 * exp(sigma * sqrt(dt) * t(apply(Z, 1, cumsum)))

  Cash_flow <- matrix(0, nrow = N, ncol = M)
  Cash_flow[, M] <- pmax(K - S[, M], 0)

  laguerre_basis <- function(x, d) {
    L <- list()
    L[[1]] <- rep(1, length(x))                                    # L0
    if (d >= 1) L[[2]] <- -x + 1                                    # L1
    if (d >= 2) L[[3]] <- 0.5 * (x^2 - 4*x + 2)                     # L2
    if (d >= 3) L[[4]] <- ( -x^3 + 9*x^2 - 18*x + 6 ) / 6           # L3
    if (d >= 4) L[[5]] <- (x^4 - 16*x^3 + 72*x^2 - 96*x + 24) / 24 # L4
    if (d >= 5) L[[6]] <- (-x^5 + 25*x^4 - 200*x^3 + 600*x^2 - 600*x + 120) / 120  # L5
    do.call(cbind, L[1:(d+1)])
  }

  for (m in M:2) {
    X <- S[, m-1]
    Y <- Cash_flow[, m] * discount

    in_the_money <- X < K
    if (sum(in_the_money) == 0) {# Skip regression if no in the money paths
      Cash_flow[, m-1] <- 0
      next
    }

    X_in <- X[in_the_money]
    Y_in <- Y[in_the_money]

    laguerre_features <- laguerre_basis(X_in, degree) # Create matrix with laguerre degree
    regression_df <- data.frame(Y = Y_in, laguerre_features)
    colnames(regression_df) <- c("Y", paste0("L", 0:degree))

    regression_formula <- as.formula(paste("Y ~", paste(colnames(regression_df)[-1], collapse = " + ")))
    regression <- lm(regression_formula, data = regression_df)

    all_features <- laguerre_basis(X, degree)
    colnames(all_features) <- paste0("L", 0:degree)
    continuation <- predict(regression, newdata = as.data.frame(all_features))

    immediate_exercise <- pmax(K - X, 0)
    Cash_flow[, m-1] <- ifelse(continuation < immediate_exercise, immediate_exercise, 0)
```

```
  }

  # Discounting cash flows
  for (i in 1:N) {
    for (j in 1:M) {
      if (Cash_flow[i, j] != 0) {
        Cash_flow[i, j] <- Cash_flow[i, j] * round(exp(-r * j), 5)
        if (j < M) Cash_flow[i, (j+1):M] <- 0
        break
      }
    }
  }

  return(mean(rowSums(Cash_flow)))
}
```

Function to price an American put using Hermite Polynomials

```
price_american_put_Hermite <- function(K, M, N, r, S0, sigma, degree) {

  dt <- 1/M
  discount <- exp(-r * dt)
  set.seed(123)
  Z <- matrix(rnorm(N * M), nrow = N, ncol = M)
  S <- S0 * exp(sigma * sqrt(dt) * t(apply(Z, 1, cumsum)))

  Cash_flow <- matrix(0, nrow = N, ncol = M)
  Cash_flow[, M] <- pmax(K - S[, M], 0)

  hermite_basis <- function(x, d) {
    H <- list()
    H[[1]] <- rep(1, length(x))                                    # H0
    if (d >= 1) H[[2]] <- x                                         # H1
    if (d >= 2) H[[3]] <- x^2 - 1                                   # H2
    if (d >= 3) H[[4]] <- x^3 - 3*x                                 # H3
    if (d >= 4) H[[5]] <- x^4 - 6*x^2 + 3                          # H4
    if (d >= 5) H[[6]] <- x^5 - 10*x^3 + 15*x                      # H5
    do.call(cbind, H[1:(d+1)])
  }

  for (m in M:2) {
    X <- S[, m-1]
    Y <- Cash_flow[, m] * discount

    in_the_money <- X < K
    if (sum(in_the_money) == 0) {
      Cash_flow[, m-1] <- 0
      next
    }

    X_in <- X[in_the_money]
    Y_in <- Y[in_the_money]
```

```
    hermite_features <- hermite_basis(X_in, degree)# Create matrix with Hermite degree
    regression_df <- data.frame(Y = Y_in, hermite_features)
    colnames(regression_df) <- c("Y", paste0("H", 0:degree))

    regression_formula <- as.formula(paste("Y ~", paste(colnames(regression_df)[-1], collapse = " + ")))
    regression <- lm(regression_formula, data = regression_df)

    all_features <- hermite_basis(X, degree)
    colnames(all_features) <- paste0("H", 0:degree)
    continuation <- predict(regression, newdata = as.data.frame(all_features))

    immediate_exercise <- pmax(K - X, 0)
    Cash_flow[, m-1] <- ifelse(continuation < immediate_exercise, immediate_exercise, 0)
  }

  # Discounting cash flows
  for (i in 1:N) {
    for (j in 1:M) {
      if (Cash_flow[i, j] != 0) {
        Cash_flow[i, j] <- Cash_flow[i, j] * round(exp(-r * j), 5)
        if (j < M) Cash_flow[i, (j+1):M] <- 0
        break
      }
    }
  }

  return(mean(rowSums(Cash_flow)))
}
```

(Appendix F) Convergence plots for Regular, Laguerre, and Hermite Polynomials

Convergence plots for Regular, Laguerre, and Hermite Polynomials Notebook Github

(Appendix G) Monte Carlo Algorithm using Custom Polynomials

Black-Scholes formula for European call and put options

```
# Black-Scholes formula for European call
bs_call <- function(S, K, r, T, sigma) {
  set.seed(123)
  d1 <- (log(S / K) + (r + 0.5 * sigma^2) * T) / (sigma * sqrt(T))
  d2 <- d1 - sigma * sqrt(T)
  S * pnorm(d1) - K * exp(-r * T) * pnorm(d2)
}

# Black-Scholes formula for European put
bs_put <- function(S, K, r, T, sigma) {
  set.seed(123)
  d1 <- (log(S / K) + (r + 0.5 * sigma^2) * T) / (sigma * sqrt(T))
  d2 <- d1 - sigma * sqrt(T)
```

```
  K * exp(-r * T) * pnorm(-d2) - S * pnorm(-d1)
}
```

Function to price an American put using European call and put options

```
price_american_put_longstaff_schwartz_MC_euro <- function(K, M, N, r, S0, sigma, polynomial) {
  dt <- 1 / M
  discount <- exp(-r * dt)
  set.seed(123)
  Z <- matrix(rnorm(N * M), nrow = N, ncol = M)
  S <- S0 * exp(sigma * sqrt(dt) * t(apply(Z, 1, cumsum)))

  Cash_flow <- matrix(0, nrow = N, ncol = M)
  Cash_flow[, M] <- pmax(K - S[, M], 0)

  for (m in M:2) {
    X <- S[, m - 1]
    T_remaining <- (M - m + 1) * dt

    call_bs <- bs_call(X, K, r, T_remaining, sigma)
    put_bs <- bs_put(X, K, r, T_remaining, sigma)

    df_reg <- data.frame( # Create regressors using custom functions
      S = X,
      call = call_bs,
      put = put_bs,
      Y = Cash_flow[, m] * discount
    )

    df_reg[X > K, ] <- NA

    if (all(is.na(df_reg))) {
      Cash_flow[, m - 1] <- 0
      next
    }

    regression <- lm(polynomial, data = df_reg)
    immediate_exercise <- pmax(K - X, 0)

    df_pred <- data.frame(
      S = X,
      call = call_bs,
      put = put_bs
    )
    continuation <- predict(regression, newdata = df_pred)
    Cash_flow[, m-1] <- ifelse(continuation < immediate_exercise, immediate_exercise, 0)
  }

  # Discounting cash flows
  for (i in 1:nrow(Cash_flow)) {
    for (j in 1:ncol(Cash_flow)) {
      if (Cash_flow[i, j] != 0) {
        Cash_flow[i, j] <- Cash_flow[i, j] * round(exp(-r * j * dt), 5)
```

```
        if (j < ncol(Cash_flow)) {
          Cash_flow[i, (j + 1):ncol(Cash_flow)] <- 0
        }
        break
      }
    }
  }

  return(mean(rowSums(Cash_flow)))
}
```

(Appendix H) Covergence Plots using European Options as Regressors

Convergence Plots using European Options as Regressors Notebook Github

(Appendix I) Convergence Error with Degree In-The-Money

Convergence Error with Degree In-The-Money Notebook Github