Practical 4: Basket and Barrier Options

Lukas Fernández, Alejandro Paniagua, Raúl Pienda

August 11, 2025

Abstract

This report explores the pricing of basket and barrier options using Monte Carlo simulation techniques. We implement control variates, antithetic variables, and bias correction methods to improve estimator performance and accuracy.

Introduction

In quantitative finance, basket and barrier options are path-dependent derivatives that require simulation-based approaches for pricing. This practical is divided into two main parts: Basket Options and Barrier Options.

1 Basket Options

1.1 Overview

We price multidimensional basket options using arithmetic payoffs of the form:

$$\psi = \max\left(0, -K + \sum_{i=1}^{n} w_i S_T^{(i)}\right)$$

Two examples are used: a synthetic basket BASKET(n), and a real basket TECH4.

1.2 Preliminaries

We implemented two MATLAB functions one for the case of a BASKET(n) and the other for a real basket TECH4.

The code used by both scripts is exactly the same except for the initial parameters related to the stocks being simulated: $S_0, K, r...$

In order to simulate the stock evolution, we have used the following formula, which directly computes the value of the correlated S_i for the entire basket at any time t after the start time.

In our case we set $t = T$, since we want to compute the expected payoffs to value the price of our basket option:

$$S_j(T) = S_j(0)e^{\left((r-\frac{\sigma_j^2}{2})T + \sigma_j\sqrt{T}(L\xi)_j\right)}$$

Where L is the Cholesky factor of the correlation matrix and ξ is a vector whose size is the total number of stocks being simulated and that is composed of independent standard normal variables. In the formula, $(L\xi)_j$ denotes the jth component of the product of L and ξ.

Once we have the value of our simulated GBMs at expiry, we can calculate the value of the payoffs using the function ψ as explained in the previous section, that is:

$$\psi = \max\left(0, -K + \sum_{i=1}^{n} w_i S_T^{(i)}\right)$$

Finally, since the price of the basket option is calculated as:

$$e^{-rT}\mathbb{E}\left[\psi\right]$$

We can approximate the value of this expectation by using the montecarlo estimator for the payoffs, which is obtained by performing the mean over the payoffs obtained for a large number of simulations. The resulting code is then the following:

```
%% CORRELATED GBM PARAMETERS:
N = 1000000;
V = chol(corr,'lower'); %cholesky decomposition to simulate correlated brownians


%% MAIN PRICING SIMULATION:

Payoffs = zeros(N,1); %observation payoffs

for i=1:N %simulate N basket outcomes
    %% USING ANALYTIC SOLUTION
    corr_N = V*randn(num_stocks,1); %correlated normals

    S = S0 .* exp((r-0.5*sigma.^2)*T + sigma *sqrt(T) .*corr_N);

    Payoffs(i) = max(weights*S-K,0); %compute payoff of this outcome


end

price = exp(-r*T)*mean(Payoffs)
```

Before this code we would have to specify the desired initial parameters to use for the stocks simulations, which were indicated in the statement for Basket(n) and TECH4.

1.3 Exercise 1

In this exercise, we shall formulate the method of multiple control variates. Let $Y_1, \ldots, Y_n$ be multiple, *not necessarily independent nor uncorrelated* control variates for the random variable X. Their expected values are assumed to be known. Let us define:

$$\hat{X} = X + \sum_{i=1}^{n} \lambda_i (Y_i - \mathbb{E}[Y_i]).$$

i) Prove that $\mathbb{E}[\hat{X}] = \mathbb{E}[X]$ regardless of $\lambda_1, \ldots, \lambda_n$, but

$$\mathbb{V}[\hat{X}] = \mathbb{V}[X] + 2\sum_{i=1}^{n} \lambda_i \mathrm{Cov}[X, Y_i] + \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j \mathrm{Cov}[Y_i, Y_j].$$

Lets first prove that $\mathbb{E}[\hat{X}] = \mathbb{E}[X]$:

$$\begin{aligned}\mathbb{E}[\hat{X}] &= \mathbb{E}\left[X + \sum_{i=1}^{n} \lambda_i (Y_i - \mathbb{E}[Y_i])\right] \\ &= \mathbb{E}[X] + \sum_{i=1}^{n} \lambda_i \,\mathbb{E}\left[Y_i - \mathbb{E}[Y_i]\right]\end{aligned}$$

$$= \mathbb{E}[X] + \sum_{i=1}^{n} \lambda_i \left(\mathbb{E}[Y_i] - \mathbb{E}[Y_i]\right)$$

$$= \mathbb{E}[X] + \sum_{i=1}^{n} \lambda_i \cdot 0 = \mathbb{E}[X]$$

Now, for $\mathbb{V}[\hat{X}]$ we have:

We will utilize one of the properties of the variance for the proof, namely

$$\mathbb{V}(\sum_{i=1}^{n} a_i X_i) = \sum_{i,j=1}^{n} a_i a_j \text{Cov}(X_i, X_j)$$

Taking into account this property for $\hat{X}$ we get:

$$\mathbb{V}[\hat{X}] = \mathbb{V}\left[X + \sum_{i=1}^{n} \lambda_i (Y_i - \mathbb{E}[Y_i])\right] =$$

$$= \mathbb{V}\left[X + \sum_{i=1}^{n} \lambda_i Y_i\right] =$$

$$= \text{Cov}(X, X) + 2\sum_{i=1}^{n} \lambda_i \text{Cov}(X, Y_i) + \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j \text{Cov}(Y_i, Y_j) =$$

$$= \mathbb{V}[X] + 2\sum_{i=1}^{n} \lambda_i \text{Cov}[X, Y_i] + \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j \text{Cov}[Y_i, Y_j]$$

Where we have used the mentioned property for the third step. Notice also that for the second step we have discarded the expectation terms, because the variance of a random variable is invariant under the addition or substraction of constant terms.

ii) Assuming fixed covariances, let $f(\lambda_1, \ldots, \lambda_n) = \mathbb{V}[\hat{X}]$, and $(\lambda_1^*, \ldots, \lambda_n^*) \in \mathbb{R}^n$ be the critical points of f. Show that they verify the linear system

$$M(\lambda_1^*, \ldots, \lambda_n^*)^T = (-\text{Cov}[X, Y_1], \ldots, -\text{Cov}[X, Y_n])^T,$$

where M is the $n \times n$ matrix with entries $M_{ij} = \text{Cov}[Y_i, Y_j]$.

In order to show this result, we first need to compute $\frac{df}{d\lambda_i}$:

$$\frac{df}{d\lambda_i} = \frac{d}{d\lambda_i}\left[\mathbb{V}[X] + 2\sum_{i=1}^{n} \lambda_i \text{Cov}[X, Y_i] + \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j \text{Cov}[Y_i, Y_j]\right] =$$

$$= 2\ \text{Cov}[X, Y_i] + 2\sum_{j=1}^{n} \lambda_j \text{Cov}(Y_i, Y_j)$$

In order to find local minima in terms of λ_i we then set $\frac{df}{d\lambda_i} = 0$ and obtain:

$$\frac{df}{d\lambda_i} = 2\ \text{Cov}[X, Y_i] + 2\sum_{j=1}^{n} \lambda_j \text{Cov}(Y_i, Y_j) = 0$$

So the local minimum for λ_i is the value that satisfies:

$$\sum_{j=1}^{n} \lambda_j \text{Cov}(Y_i, Y_j) = \text{Cov}[X, Y_i]$$

To formerly check that this is a local minimum, we can compute $\frac{d^2 f}{d\lambda_i^2}$:

$$\frac{d^2 f}{d\lambda_i^2} = 2\ \mathrm{Cov}(Y_i, Y_i) = 2\mathbb{V}[Y_i] \geq 0$$

By setting $\frac{df}{d\lambda_i} = 0\ \forall i$ we then obtain the global minimum, which can be described as:

$$\underbrace{\begin{bmatrix} \mathrm{Cov}[Y_1, Y_1] & \mathrm{Cov}[Y_1, Y_2] & \cdots & \mathrm{Cov}[Y_1, Y_n] \\ \mathrm{Cov}[Y_2, Y_1] & \mathrm{Cov}[Y_2, Y_2] & \cdots & \mathrm{Cov}[Y_2, Y_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[Y_n, Y_1] & \mathrm{Cov}[Y_n, Y_2] & \cdots & \mathrm{Cov}[Y_n, Y_n] \end{bmatrix}}_{M} \begin{bmatrix} \lambda_1^* \\ \lambda_2^* \\ \vdots \\ \lambda_n^* \end{bmatrix} = - \begin{bmatrix} \mathrm{Cov}[X, Y_1] \\ \mathrm{Cov}[X, Y_2] \\ \vdots \\ \mathrm{Cov}[X, Y_n] \end{bmatrix}$$

iii) What are the required conidtions for $(\lambda_1^*, \ldots, \lambda_n^*)$ to be the unique minimiser of f.

We observe that the value for the minimiser $(\lambda_1^*, \ldots, \lambda_n^*)$ is obtained from a linear system of equations. Therefore if the matrix M used in the system has the trivial kernel (its kernel is just composed of 0), the solution to the system will be unique. That is:

Let $M\,\vec{x} = b$ be a linear system and $ker(M) = \{0\}$, then the solution $\vec{x}$ to the system is unique.

Proof: (By Contradiction)

Assume that $ker(M) = \{0\}$ and $\vec{u}$, $\vec{v}$ are two distinct vectors such that $M\vec{u} = b,\ M\vec{v} = b$

We then have that $M\vec{u} = M\vec{v}$

Therefore $M\,(\vec{u} - \vec{v}) = 0 \iff (\vec{u} - \vec{v}) \in ker(M)$

But since $\vec{u} \neq \vec{v} \implies (\vec{u} - \vec{v}) \neq 0$ which is a contradiction with $ker(M) = \{0\}$.

Then, if the matrix M of our system has a trivial kernel, the minimiser will be unique. This can be achieved for example, when M is a positive definite matrix.

1.4 Exercise 2: Simplest Control Variates

- **i)** Implement the above control variates, and use them on BASKET(5) and TECH4. Make sure that the variance reduction is as expected theoretically. Compute the correlation between the score and the "total control variate", defined as:

$$Y_i = w_i e^{-rT} S_T^{(i)}$$

where w_i is the corresponding asset weight, r the risk-free rate, T the maturity, and $S_T^{(i)}$ the simulated terminal price of asset i.

Let us delve into the implementation of control variates where we will focus on the fundamental concepts that our methodology relies on. Firstly, initialize all the constants for either BASKET(5) or TECH4. Firstly, before delving into the theoretical part, let us remark that we have separated the code into two components, firstly a training phase where we find the optimal lambda star and an evaluation phase where we apply this lambda star.

After this we delve into the section where theory comes into play.

So we must construct a control variate for each stock in each Monte Carlo iteration based on the discounted terminal prices of each asset in the basket. In MATLAB:

```
controls(i, :) = weights .* (exp(-r * T) * ST');
```

Here we have the ST' is the transpose of the vector of terminal prices, making it a row vector.

We also calculate the score, which corresponds to the discounted payoff of the basket call option. It is defined as:

$$X = e^{-rT} \cdot \max\left(\sum_{i=1}^{n} w_i S_T^{(i)} - K,\, 0\right)$$

This in MATLAB is implemented as follows:

```
basket_value = weights * ST;
payoff = max(basket_value - K, 0);
scores_train(i) = exp(-r * T) * payoff;
```

This value is stored in the vector `scores_train`. This vector accumulates all the Monte Carlo realizations of the payoff estimator.

For the purpose of this we will call scores_train as Xc_train = scores_train and also Yc_train = controls_train.

Now is where the essence of control variates comes into play: we have two fors both looping through the number of stocks but one with index i and another with index j.

Then, we compute the empirical covariance between X and each component of $\mathbf{Y}$, which results in a vector $\mathbf{c} \in \mathbb{R}^d$:

```
covXY = mean(Xc_train .* Yc_train(:,i)) - mean(Xc_train) * mean(Yc_train(:,i));
```

We store this in the vector c:

```
c(i) = covXY;
```

This gives:

$$\mathbf{c} = \mathrm{Cov}(X, \mathbf{Y}) \in \mathbb{R}^d$$

Next, we compute the empirical covariance matrix M of the control variates:

```
covY = mean(Yc\_train(:,i) .* Yc\_train(:,j)) -

mean(Yc\_train(:,i)) * mean(Yc\_train(:,j));
```

We store this in the matrix M:

```
M(i,j) = covY;
```

Finally, we compute the optimal control variate weights $\boldsymbol{\lambda}^* \in \mathbb{R}^d$ as:

```
lambda_star = M \ c;
```

This solves the linear system:

$$M\boldsymbol{\lambda}^* = \mathbf{c} \quad \Rightarrow \quad \boldsymbol{\lambda}^* = \text{-}\mathrm{Cov}(\mathbf{Y})^{-1} \cdot \mathrm{Cov}(X, \mathbf{Y})$$

The vector $\boldsymbol{\lambda}^*$ gives the optimal linear combination of the control variates that minimizes the variance of the adjusted estimator.

Note that we don't have to worry about the problem of the control variates not being optimal as shown in the first exercise, this is a unique minimizer of the variance. This furthermore implies that if a control variate isn't making the variance decrease for some reason, its corresponding coefficient would be zero.

Now we enter into phase two of the code the evaluation:

So, essentially, we simulate for t_trials where the inner components of the for are the monte carlo simulation

Then, we compute the control correction for each Monte Carlo path using the optimal weight vector $\boldsymbol{\lambda}^*$:

```
control_combo = (controls - EY) * lambda_star;
```

This gives the scalar correction:

$$\text{correction}_i = \boldsymbol{\lambda}^{*\top} \cdot (\mathbf{Y}_i - \mathbb{E}[\mathbf{Y}])$$

We subtract this correction from the original payoff to obtain the adjusted estimator:

```
X_adj = scores - control_combo;
```

Which corresponds to:

$$X_{\text{adj}} = X - \boldsymbol{\lambda}^{*\top} (\mathbf{Y} - \mathbb{E}[\mathbf{Y}])$$

This adjusted estimator preserves unbiasedness while reducing variance, assuming the control variates are correlated with the score.

Variance Reduction Evaluation

To evaluate the effectiveness of the control variates, we compare the variance of the adjusted estimator with that of the plain Monte Carlo estimator. We also compute the theoretical variance of the adjusted estimator:

```
theoretical_var = var_plain - c' * (M \ c);
```

This corresponds to:

$$\text{Var}(X_{\text{adj}}) = \text{Var}(X) - \boldsymbol{c}^\top \cdot M^{-1} \cdot \boldsymbol{c}$$

Then, we define the empirical and theoretical variance reduction factors (VRFs) as follows:

```
vrf_empirical = var_plain / var_cv;
vrf_theoretical = var_plain / theoretical_var;
```

$$\text{VRF}_{\text{empirical}} = \frac{\text{Var}(X)}{\text{Var}(X_{\text{adj}})}, \quad \text{VRF}_{\text{theoretical}} = \frac{\text{Var}(X)}{\text{Var}(X) - \boldsymbol{c}^\top M^{-1} \boldsymbol{c}}$$

These two quantities allow us to compare the observed variance reduction against the theoretical prediction. In our simulations, the empirical variance reduction factor closely matches the theoretical one, confirming the effectiveness and correctness of the control variate methodology.

Furthermore, we also compute the correlation coefficient κ between the score and the control correction:

$$\kappa = \text{Corr}(X, \boldsymbol{\lambda}^{*\top}(\mathbf{Y} - \mathbb{E}[\mathbf{Y}]))$$

Sample Size N	**Mean** κ	**Std** κ
1000	0.9191	0.0054
5000	0.9184	0.0025
10000	0.9188	0.0018
50000	0.9185	0.0008
100000	0.9186	0.0005
500000	0.9185	0.0002

Table 1: Correlation κ between raw estimator and control variate correction

Sample Size N	Empirical VRF (MC / CV)
1000	4.7657
5000	5.4538
10000	4.3431
50000	5.1694
100000	6.6675
500000	6.9478

Table 2: Empirical variance reduction factor for different values of N over 100 trials.

- **ii)** Plot CPU time vs variance (with/without control variates).
- **ii) Variance vs. CPU Time**

(ii) CPU Time vs. Variance Analysis

In this experiment, we investigate the relationship between computational time and the variance of the Monte Carlo estimator, both with and without the use of control variates. According to theory, the variance of the Monte Carlo estimator should decrease at a rate of $\mathcal{O}(1/N)$ as the number of samples N increases. This should also translate to a decreasing trend of variance with respect to CPU time in log-log scale.

To validate this, we evaluated the basket option price for several increasing values of N, using 100 repeated trials per N to estimate the variance of the mean estimator. The results are shown in Figure 1.

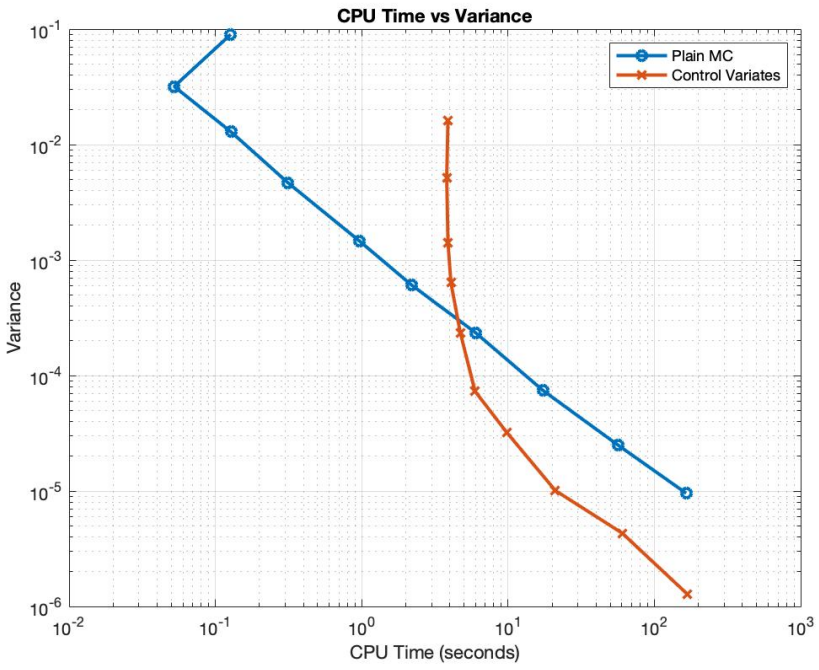


Figure 1: CPU Time vs Variance for plain Monte Carlo and control variates. Log-log plot over 10 values of N ranging from 10^2 to 10^6.

- **iii) Effect of Increasing Basket Dimension n and Normality of Errors**

  We observe the following:

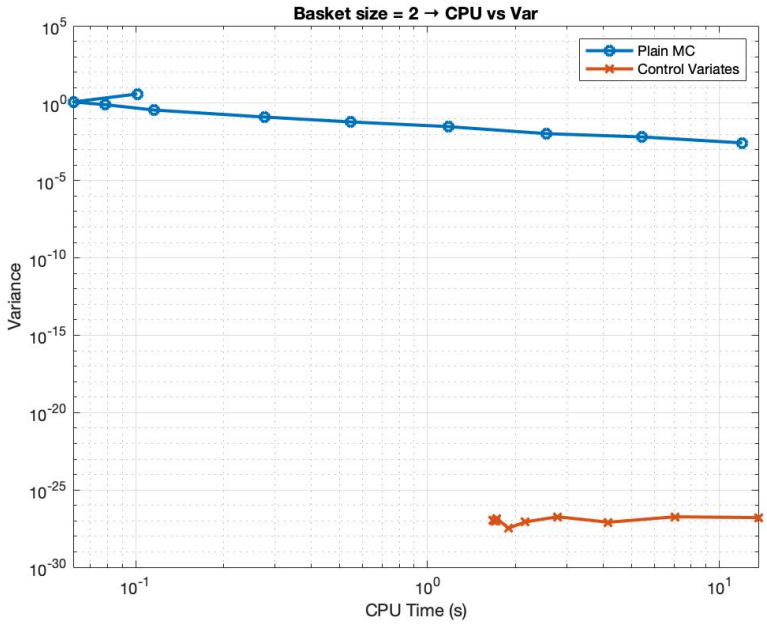


Figure 2: Basket size 2 CPU vs variance N for BASKET(n) (log-log scale).

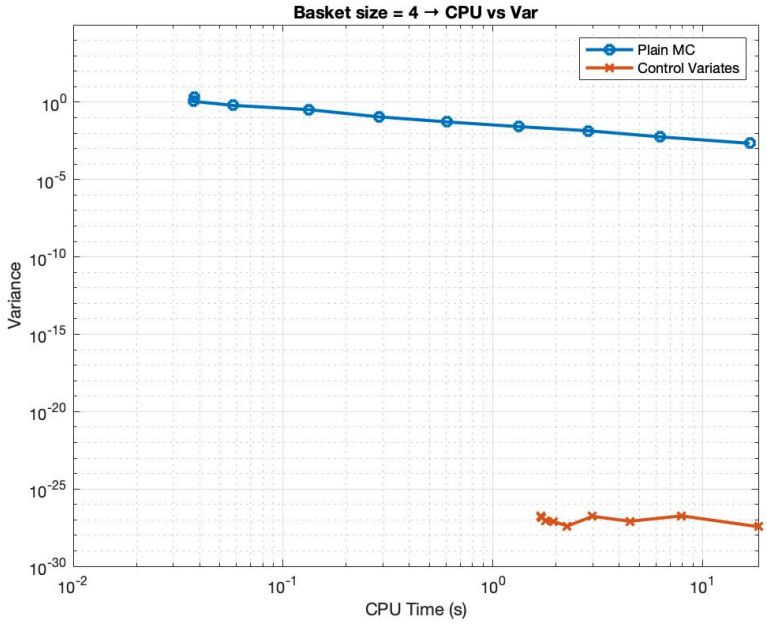


Figure 3: Basket size 4 CPU vs variance N for BASKET(n) (log-log scale).

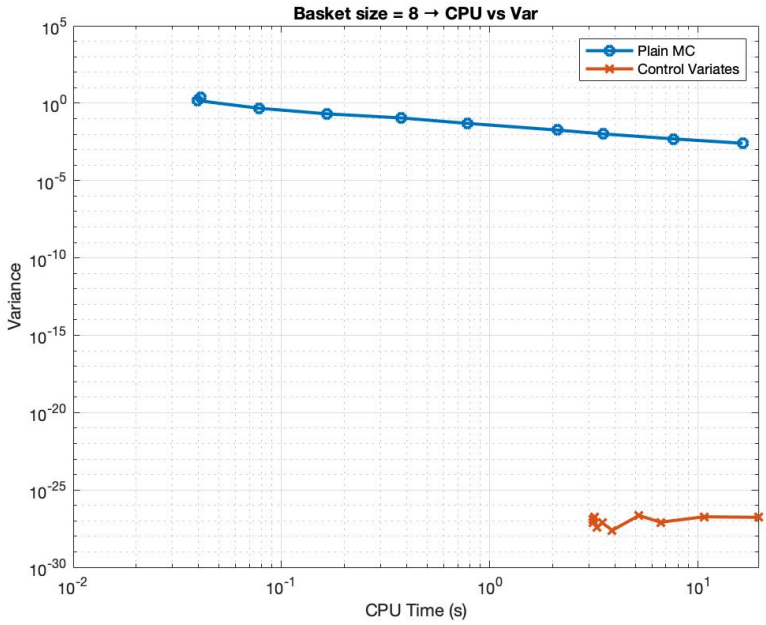


Figure 4: Basket size 8 CPU vs variance N for BASKET(n) (log-log scale).

Normality of Empirical Errors

To verify the normality assumption required by confidence intervals (e.g., 99.7%), we analyze the empirical distribution of the errors for the control variate estimator:

$$\text{error}_i = X_{\text{adj},i} - \mathbb{E}[X_{\text{adj}}]$$

The histogram of errors is shown in Figure 5, along with a fitted normal distribution.

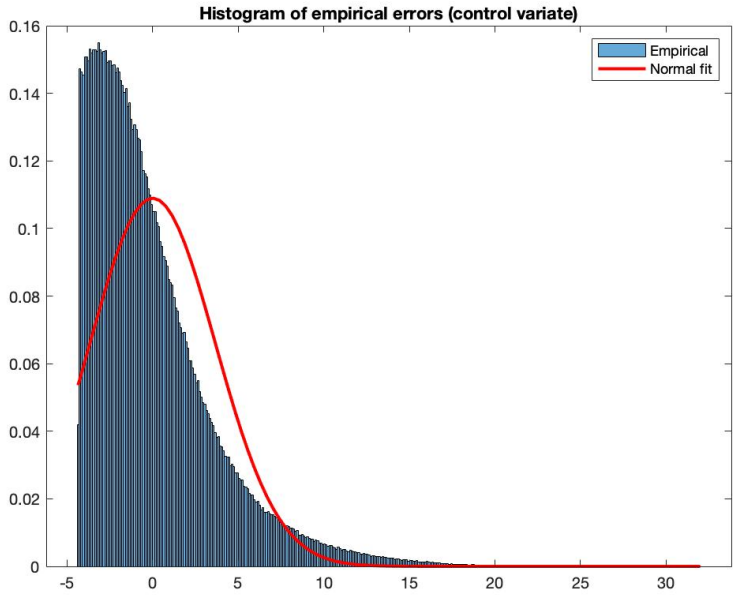


Figure 5: Histogram of empirical errors for the control variate estimator with normal fit.

This distribution does not fit because in reality we are not trying to graph the distribution of the errors, instead we want to use the Central Limit Theorem to graph the mean of the errors.

Later on we had some troubles with the mean of the variance as we were obtaining odd values, but they ended up being logical. Namely, we were obtaining that the standard deviation of the final control variate and the Monte Carlo approximation were the same. This is obviously incorrect, but let us delve into this.

The problem arose from that we were using the same data to approximate the value of lambda star and then applying the control variates to it.

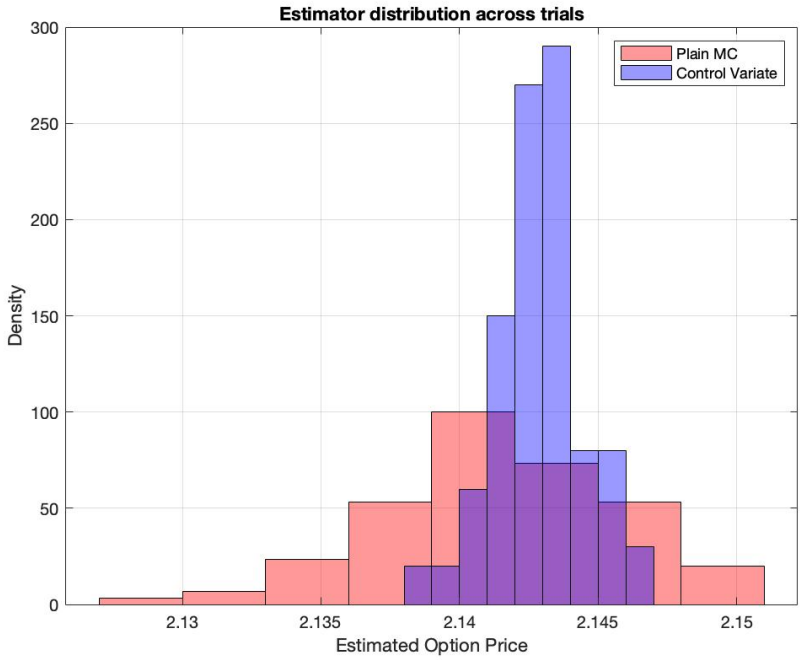


Figure 6: Distribution of the estimated option prices using both plain Monte Carlo (red) and Control Variates (blue).

Figure 6 shows the distribution of the estimated option price obtained using 100 independent trials for both the plain Monte Carlo method and the Control Variates technique. It is clearly observed that the distribution for the Control Variates estimator is significantly narrower and more concentrated around the mean.

Conclusion: the control variate estimator not only reduces variance significantly, but also produces errors that are well-behaved and approximately Gaussian in practice, enabling reliable uncertainty quantification.

- **iv)** Analyze how results vary with strike K.

  In this exercise, we analyze how the efficiency of the control variate method varies with the strike price K in the context of basket options. We compare the performance of plain Monte Carlo (MC) simulation and the control variate approach for different strike prices, ranging from deep in-the-money (ITM) to deep out-of-the-money (OTM).

  We simulate the payoff of a European call on a basket of five assets with the following parameters: $T = 1$, $r = 0.1$, $S_0 = 100$, $\sigma = 0.2$, $\rho = 0.5$, and equal weights. A Cholesky decomposition is used to introduce correlation between asset returns. The basket value is approximated as $\bar{S}_0 = \mathbf{w}^\top \mathbf{S}_0$ and the strikes are chosen as:

$$K \in \{0.7\bar{S}_0, 0.85\bar{S}_0, \bar{S}_0, 1.15\bar{S}_0, 1.3\bar{S}_0\}$$

Control variate method. The control variate is constructed as a linear combination of the discounted terminal prices of the individual assets. The optimal control weights λ^* are computed as:

$$\lambda^* = \Sigma_Y^{-1} \cdot \mathrm{Cov}(X, Y)$$

where X is the basket payoff and Y is the vector of control variables. The adjusted estimator is:

$$\hat{X}_{\mathrm{cv}} = X - \lambda^*(Y - \mathbb{E}[Y])$$

Variance and error reduction. The following figure shows the variance of the estimator with and without control variates as a function of the strike price:

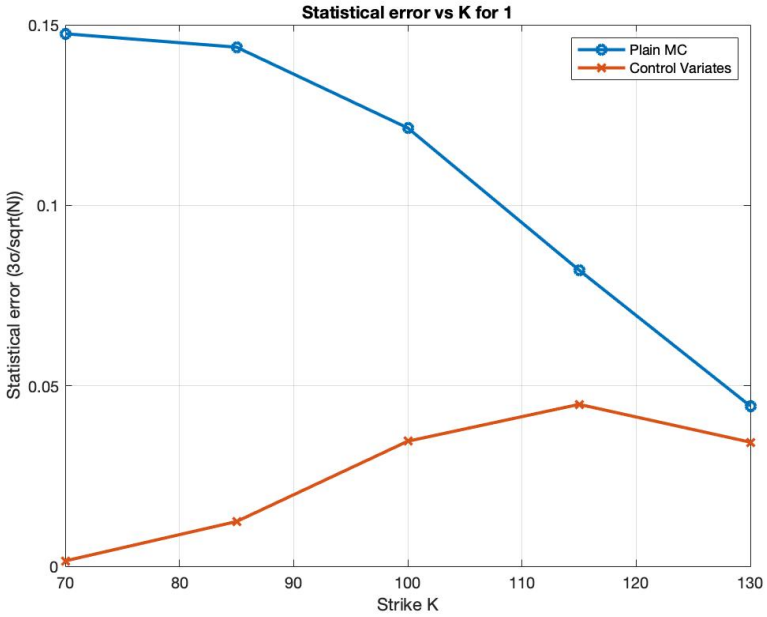


Figure 7: Variance of the estimator vs. strike K. Control variates significantly reduce variance for all strikes.

We observe that the control variate method consistently achieves a substantial variance reduction, especially for low strike prices where the payoff is large and more correlated with the control.

Correlation analysis. The next figure shows the correlation coefficient κ between the original payoff and the control variate as a function of K:

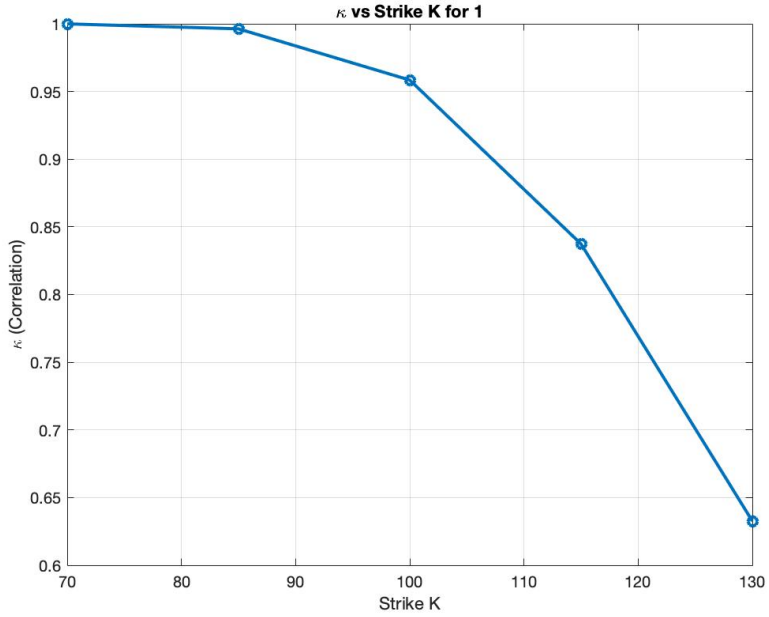


Figure 8: Correlation κ between the control variate and the payoff. High correlation leads to greater variance reduction.

The correlation κ decreases as K increases. This is expected since higher strikes correspond to lower (and rarer) payoffs, which are less aligned with the behavior of the basket's terminal asset prices.

Statistical error comparison. Finally, the figure below shows the statistical error (measured as $3\sigma/\sqrt{N}$) for both plain Monte Carlo and the control variate method:

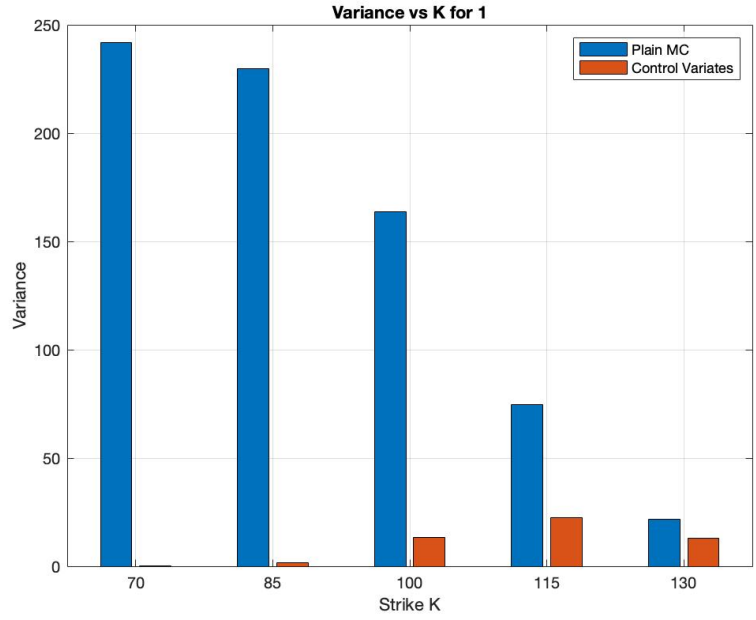


Figure 9: Statistical error vs. strike K. Control variates offer better accuracy with the same number of simulations.

For low and medium strikes, the control variate method drastically outperforms plain MC. As K increases and the option becomes more out-of-the-money, the benefit of control variates diminishes due to reduced correlation.

Conclusion. The control variate method is highly effective when the payoff is sufficiently correlated with the discounted terminal values of the assets, which is typically the case for low-to-moderate strike prices. This experiment illustrates the importance of correlation in the design of efficient variance reduction techniques.

v) Combining Control Variates with Antithetic Variables

In this experiment, we explore whether the use of control variates (CV) can be effectively combined with antithetic variables (AV) to further reduce the statistical error and variance in Monte Carlo simulation of a basket option. We evaluate four methods: Plain Monte Carlo (MC), Control Variates (CV), Antithetic Variables (AV), and their combination (CV + AV). For each method, we compute the estimator 100 times using $N = 10^4$ simulations and record the variance, standard error, and CPU time.

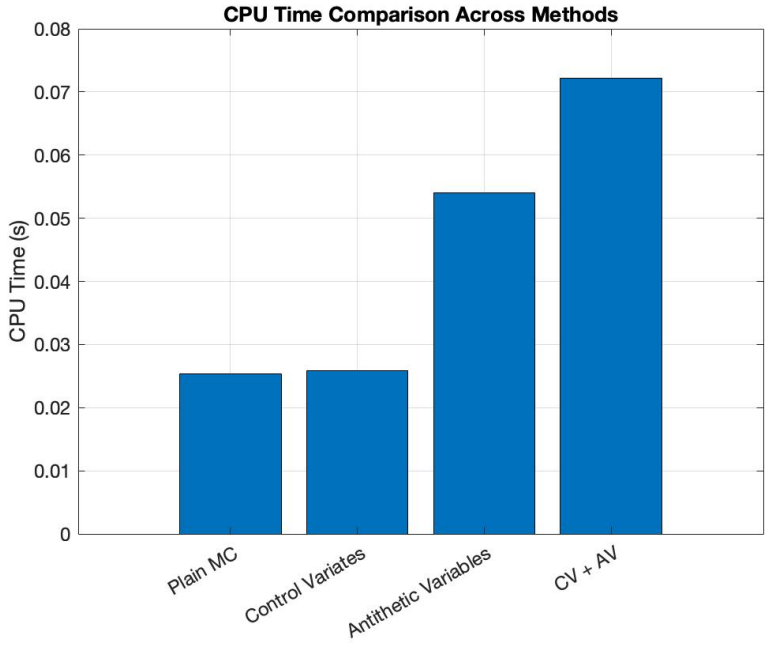


Figure 10: CPU Time Comparison Across Methods

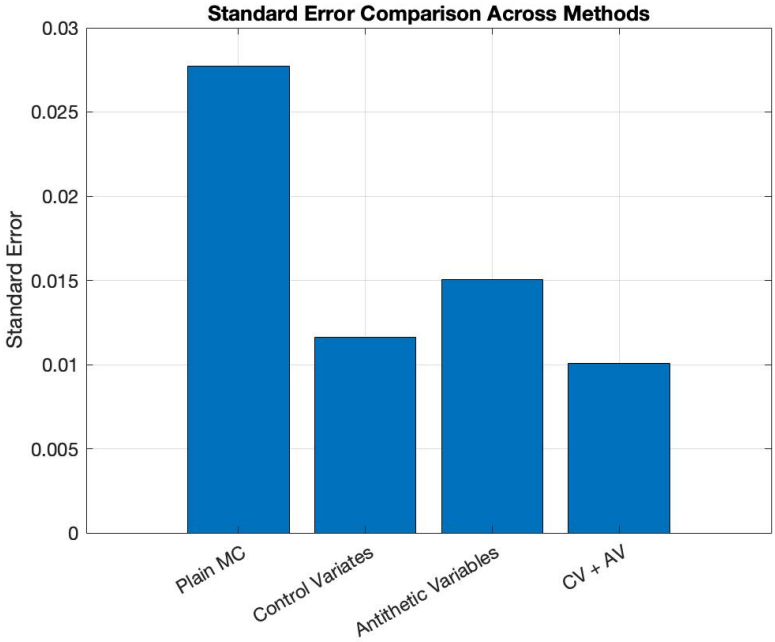


Figure 11: Standard Error Comparison Across Methods

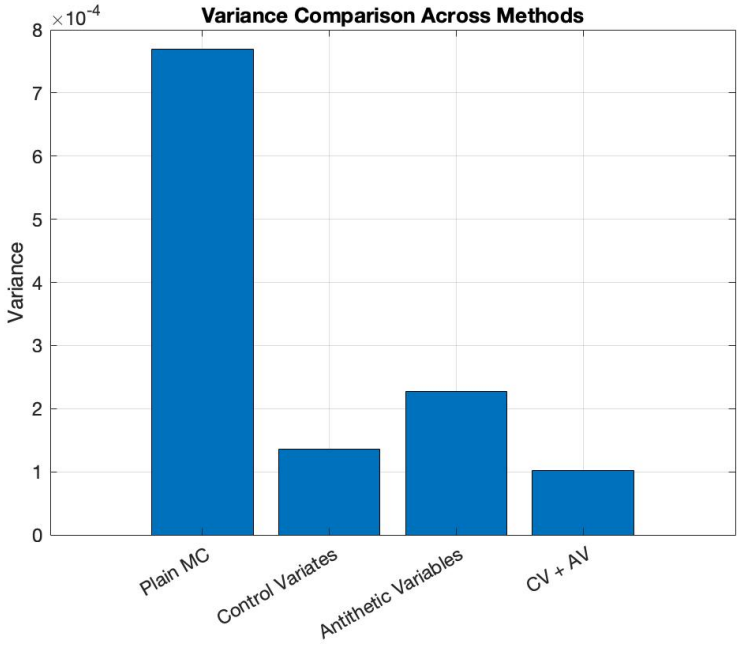


Figure 12: Variance Comparison Across Methods

Variance vs. Sample Size. The figure below shows the variance of the estimator as a function of the number of simulations N:

Method	Variance	Std. Error	CPU Time (s)
Plain MC	0.0010	0.0308	0.0344
Control Variates (CV)	0.0001	0.0116	0.0291
Antithetic Variates (AV)	0.0002	0.0153	0.0573
CV + AV	0.0001	0.0099	0.0626

Table 3: Comparison of variance reduction techniques using control variates (CV), antithetic variables (AV), and their combination (CV+AV). Results are averaged over 100 trials.

Discussion

As shown in Figures 12, 11, and 10, combining both control variates and antithetic variables yields the smallest statistical error and variance across all methods, though at the cost of increased computational time. The standard error is reduced by over 60% when using CV + AV compared to Plain MC. This is consistent with theoretical expectations: control variates improve accuracy by exploiting correlation with known quantities, while antithetic variables reduce variance by introducing negatively correlated samples.

While CV alone provides substantial improvement at minimal extra cost, the inclusion of AV doubles the sample size per iteration, explaining the higher CPU time. Nevertheless, the tradeoff between accuracy and time may be worth it in applications where precision is critical.

This hybrid technique provides a powerful variance reduction strategy for basket option pricing problems.

1.5 Exercise 3: Pellizzari Control Variates

The ith Pellizzari has the following form:

$$Y_i = e^{-rT} \max\left(0, -K + w_i S_T^{(i)} + \sum_{j \neq i} w_j \mathbb{E}[S_T^{(j)}]\right) i = 1, ..., n \tag{1}$$

where the stock risk neutral prices sdes are :

$$\frac{dS_i}{S_i} = rdt + dW \tag{2}$$

It is basically the payoff of a basket option discounted to today's price, assuming that all but the ith stock have ended up at their expected prices. Then, it makes sense that it will be correlated with the discounted payoff. For instance, when the ith stock goes above its mean then on average the payoff should tend to increase as it will probably be the case for the mean expected payoff of the ith Pezziralli variable. This is just an informal argument to justify the presence of a correlation between these variables, which is half of what makes a control variate useful. In addition we also must be able to compute the mean which can be deduced after algebraic manipulation to leave it on a familiar form:

$$E[Y_i] = e^{-rT} w_i \max(0, S_T^{(i)} - \hat{k_i}) \tag{3}$$

where the constant $\hat{k_i}$ has the following form:

$$\hat{k_i} = \frac{K + w_i E[S_T^{(i)}] - \sum_{j=1}^{n} w_j E[S_T^{(j)}]}{w_i} \tag{4}$$

In addition we know that $E[S_T^i] = S_0^i e^{rT}$. Now, we notice that the expected value of the control variate in (3) can be interpreted as the price of a European call option with current price S_0^i, time to expiration T and a "fictitious" strike price of $\hat{k_i}$ \$.

However, throughout the development of the exercise we have found out that there is a catch here! This will only be the case whenever $\hat{k_i}$ is nonnegative which is equivalent to the condition we derived on how large the original strike price has to be in relation to the other parameters:

$$K > \frac{n-1}{n} e^{rT} \sum_i w_i S_0^i \tag{5}$$

This is an important thing to take into account when implementing the pricing algorithm using this control variate as it can lead to nonsense error if not taken into account. So in this sense the control variate is limited to situations where (5) holds only. However, we have found that it is really efficient as opposed to crude Monte Carlo...

i) Implement the above control variates, and use them on BASKET(5) and TECH4. Make sure that the variance reduction is as expected theoretically. Compute the correlation between the score and the "total control variate"

For the implementation we first obtain the approximate linear system to find the maximal variance reduction in our total control variate which has the form:

$$Z = e^{-rT} C_T + \sum_i \lambda_i (e^{-rT} Y_i - E[Y_i]) \tag{6}$$

In code we obtained the system for the best choice of lambdas by estimating the covariances after running an initial simulation of 5000 realisations as follows:

```
%% CONTROL VARIATE SETUP : FINDING OPTIMAL LAMBDAS
% Solving approximate linear system by approximating covariance matrix for control v
mean_maturity = S0*exp(r*T); % Mean stock prices at maturity

% tsetup = tic;  % Start timer for MMC setup
for i=1:5000
    for j=1:n
        S = S + r*S*dt + sqrt(dt)*S.*V*randn(num_stocks,1);
        S = max(S,0);
    end
    Payoffs(i,1) = max(weights*S-K,0);

    for p=1:num_stocks
        Temp = mean_maturity;
        Temp(p,1) = S(p,1);
        Perizzellis(i,p) = max(weights*Temp-K,0);
    end
    S(:,1) = S0;
end

% Estimate covariances
M = cov(Perizzellis);
b = zeros(num_stocks,1);
for p=1:num_stocks
    cov_value = (-1)*cov(Payoffs, Perizzellis(:,p));
    b(p,1) = cov_value(1,2);
end
lambda = linsolve(M,b); % Solution of the system

% Expected variance reduction should be:
theoretical_var_red = 100*(b'*inv(M)*(-1)*b)/(var(Payoffs));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

We also take into account the theoretical reduction to validate our implementation.

We calculated the estimates with both crude Monte Carlo (sample mean of the payoffs) and the sample mean estimator of (6) over a series of sample sizes ranging from 1000 up to 10000. We estimated the standard error in each case and calculated the observed variance reduction for each case, comparing across each method and with the theoretical reduction for the optimal choice of coefficients, obtaining the following results:

	MC	MC_deviation	MMC	MMC_deviation	Var reduction (%)
1000	12.4429	0.4095	11.7192	0.1127	-92
2000	11.0934	0.2747	11.7279	0.0827	-90
3000	12.0015	0.2308	11.8131	0.0666	-91
4000	11.9716	0.2039	11.9130	0.0582	-91
5000	11.8433	0.1831	11.8376	0.0522	-91
6000	11.8526	0.1635	11.7907	0.0480	-91
7000	11.9195	0.1535	11.7603	0.0433	-92
8000	11.8982	0.1453	11.8876	0.0417	-91
9000	11.9792	0.1365	11.8580	0.0382	-92
10000	12.0255	0.1284	11.8088	0.0364	-91

Figure 13: Basket(5)

The results are astonishingly better for MMC (Pezzirelli) as opposed to MC. The variance reduction is independent of the sample size as expected and in fact it is over the theoretical one which we found to be about -89 percent.

Also, considering the fact that 3 standard deviations from the mean are extremely rare, we see that with high confidence, the method of Pezzirelli will give an error of less than 34 cents from only using 1000 simulations as opposed to that of crude Monte Carlo which barely achieves that accuracy (still 4 cents off) when the simulations are pumped all the way up to 10000 instead! Thus, with Pezzirelli method we obtain the same error as the basic Monte Carlo but using 10 times less resources- this is an spectacular 10x speed up !

Similar conclusions hold on the Tech basket:

	MC	MC_deviation	MMC	MMC_deviation	Var reduction (%)
1000	2.2396	0.1026	2.2825	0.0271	-93
2000	2.3904	0.0708	2.2909	0.0188	-92
3000	2.1953	0.0537	2.2642	0.0150	-92
4000	2.2810	0.0481	2.2536	0.0131	-92
5000	2.2096	0.0422	2.2588	0.0117	-92
6000	2.3282	0.0402	2.2714	0.0107	-92
7000	2.2527	0.0369	2.2890	0.0101	-92
8000	2.2628	0.0345	2.2791	0.0093	-92
9000	2.3114	0.0328	2.2839	0.0088	-92
10000	2.2227	0.0300	2.2692	0.0083	-92

Figure 14: Tech4

Here the same speed up is obtained but the rate of convergence looks higher than that of Basket(5) across both methods. In particular, we are strongly confident of making an error of less than 9 cents using only 1000 simulations of MMC which takes about 10 times more simulations for that same accuracy level using the crude MC estimator.

In addition, we computed the correlation between the score and the total control variate which should be negatively correlated with the score if the implementation was correct as this would mean that the overshots of one are the undershots of the other, thus ensuring an effective variance reduction and a correct implementation in our code.

We used the following lines of code in Matlab to do this, using the function coerff once the simulations had taken place:

```
[a, p] = corrcoef(Payoffs, MMC_Payoffs);
kappa(s) = a(1, 2);
```

Giving us a value of -0.96 ,which were good news for us.

ii) CPU time VS Variance

Of course, the tradeoff for this speed up is an extra time cost for setting up the control variates as explained above and in general observing from the total estimator which sums up several values. The second component of the time cost turn out to be negligible in comparison to MC and the set up part (solving the linear system for optimal lambdas). We timed the executions during our simulations for different sample sizes and reflect our conclusions on the following 'path-like' plot, where each point corresponds to a particular (execution time,variance) pair for a method (red for MC and blue for MMC) together with a mark on top which corresponds to the associated sample size factor of 1000:

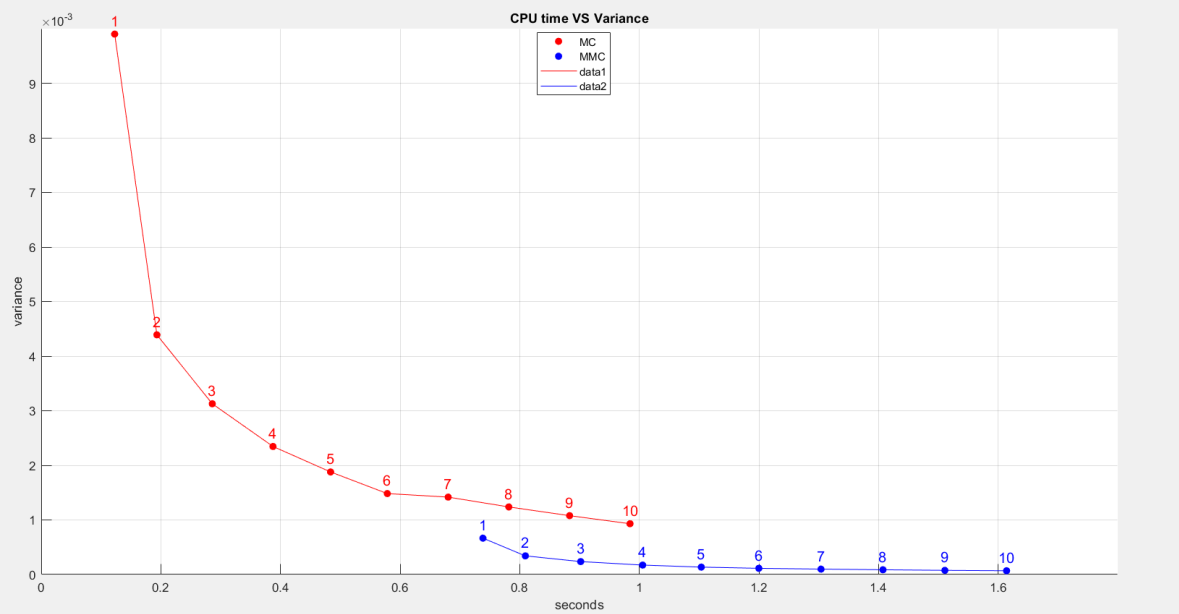


Figure 15: Efficiency Plot

Aside from the initial setup time for the control variates, which is about 0.65 seconds which causes offset the lateral shift, we see that the fundamental execution time is pretty much the same for both methods. As it was said in the beginning, but can be now equivalently observed here, the MMC method achieves a higher accuracy in any case and outperforms MC in accuracy in the smallest sample size which not even the largest one (10000) is achieved by MC. Hence, MMC is overall a better choice since it not only requires less simulations to reach a desired error threshold but also does it in less time.

iii) We approximated the distributions of both estimators with a particular sample size of N = 1000 on Basket(10) by drawing from each estimator several times. If the claim that MMC was not a smarter choice than MC becomes clear after seeing how the histogram of MMC shows much less variability where in both cases about 99.7 of the observed errors are within 3 standard deviations of the target mean. This is just as the Central Limit Theorem but the important thing to observe is that the deviation for the MMC estimator is qualitatively smaller than that of the MC estimator, as we have been showing throughout the exercise.

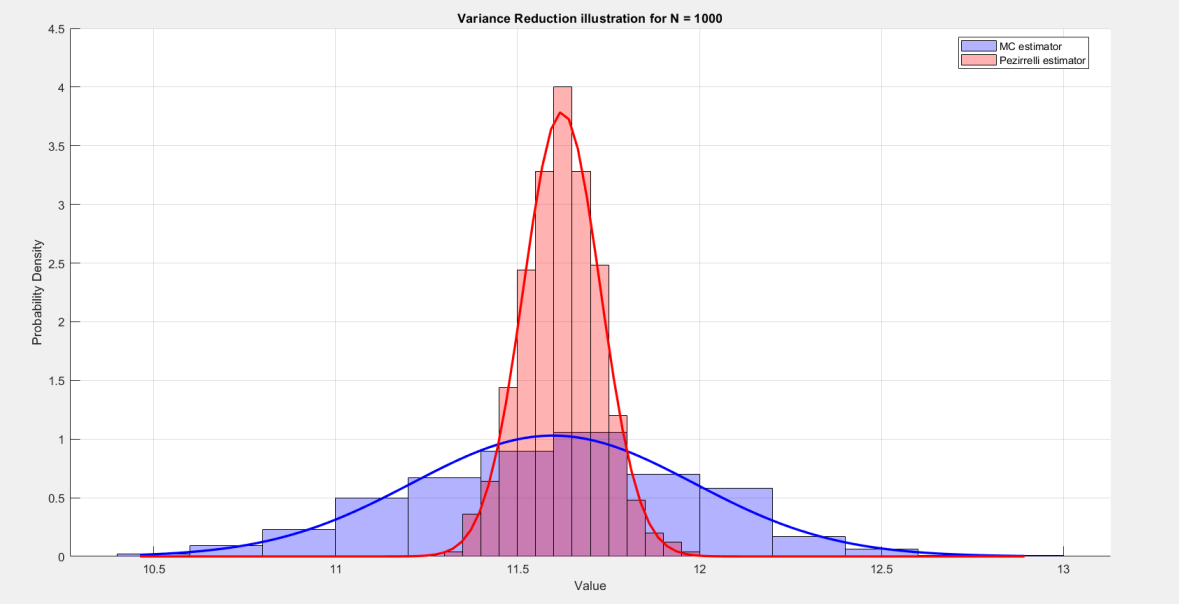


Figure 16: Esimators distribution

Moreover, we computed the variance reduction and correlation coefficient for varying basket sizes from $n = 1, ..10$ as this is the largest possible range such that Pezzirelli variates work ,recalling the constraint (1) we found. The average variance reduction was of 95 percent. Equivalently, the correlation coefficient κ stayed almost constant , being around -0.96.

iv) Behaviour for different K

- When K is much larger than that of the underlying now ie: the option is way OTM , the distribution of the payoff will accumulate most of its probability density around 0 as the most likely case is that the option expires worthless. For this reason, both estimators will not exhibit much statistical error as it is an edge case. This also explains that the difference between the standard payoff and the mean used by Pezzirelly wont have room for being correlated as they will just be the degerate variable 0 (for large K). In conclusion, the use effectiveness of both techniques becomes nearly perfect with few sample sizes but throughout the increasing of K the correlation decreases in magnitude which leads to an overall reduction of efficiency in the control variates or equivalently a less notable variance reduction.

- When K goes to 0 the option becomes an ITM option. The least that it can go is about 89 units which we obtained from the constraint we found (1). Around that minimum, the option is really is ITM by about 10 units up. In such a case we observe the best variance reduction of about -97 percent with high correlation coefficient of about -0.99. This is because the control variate technique really outperforms the MC in this edge case as the control variates are very unlikely to give a 0 value at expiry since the overall expectation of the basket at maturity will be 110. But fixing all but one will then surely reduce the chances of the value of the control variate being very different from the mean one.

v) Can you combine these control variates with antithetic variables? If yes, show the improvements, in terms of the statistical error, variance reduction amd computational time.

If we wanted to combine the total pezzirelli estimator together with the antithetic variables technique we would like to use only $N/2$ outcomes to simulate them and then reuse these together with the inverse cdf of the estimator to obtain the correlated estimates. However, we need to have the inverse cdf which we do not have in this case as the total Pezzirelli estimator we constructed is a complex combination of payoffs whose cdf we clearly dont know for this case.

2 Barrier Options

2.1 Overview

We study the numerical pricing of down-and-out barrier options under GBM and OU processes, addressing issues like bias, timestep dependency, and Brownian bridge techniques.

2.2 Exercise 4: Down-and-Out Option PDE

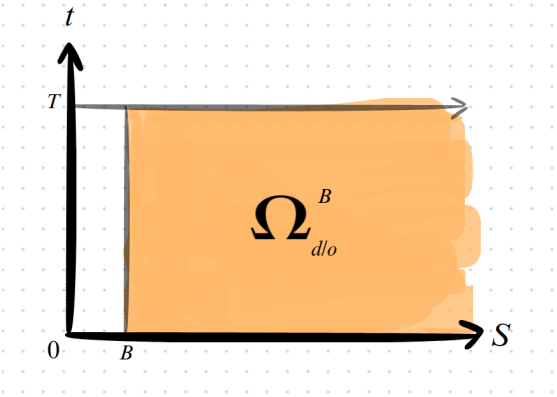
- **i)** Write down the complete boundary value problem (BVP) for $C_{d/o}$: PDE, initial/terminal conditions (if any), boundary conditions (BCs) (if any). Sketch the domain of that BVP, $\Omega^B_{d/o}$. Explain succinctly if and why the BVP is well posed, and what it means. Repeat everything for the vanilla Call (i.e. without barrier) of solution C_v. Briefly comment on the differences between the BVP for $C_{d/o}$ and for C_v.

  Assume that $S_0 > B$ otherwise the option would be worthless. We then have the following boundary value problem for $C_{d/o}$:

$$\begin{cases} \dfrac{\partial C_{d/o}}{\partial t} + \dfrac{\sigma^2 S^2}{2}\dfrac{\partial^2 C_{d/o}}{\partial S^2} + rS\dfrac{\partial C_{d/o}}{\partial S} - rC_{d/o} = 0, \\ C_{d/o}(T,S) = (S-K)^+, \\ \lim\limits_{S\to\infty} C_{d/o}(t,S) = S, \\ C_{d/o}(t,B) = 0 \end{cases}$$

  For the domain $\Omega^B_{d/o}$ we have:

$$\Omega^B_{d/o} = \{\ (t,S) \in [0,T] \times [B,\infty)\ \}$$



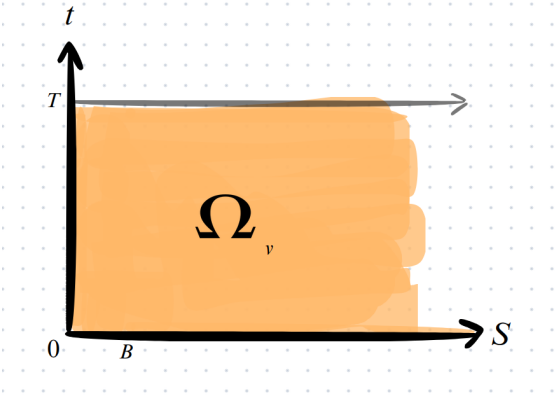
  The BVP is well-posed (it has a solution), this is because we have a parabolic PDE with boundary and terminal conditions whose domain is regular.

  Since the sign of the time derivative term is positive and we have a terminal condition (instead of initial), then the PDE will have a solution.

  For a Vanilla option we would have the following:

$$\begin{cases} \dfrac{\partial C_v}{\partial t} + \dfrac{\sigma^2 S^2}{2}\dfrac{\partial^2 C_v}{\partial S^2} + rS\dfrac{\partial C_v}{\partial S} - rC_v = 0, \\ C_v(T,S) = (S-K)^+, \\ \lim\limits_{S\to\infty} C_v(t,S) = S, \\ C_v(t,0) = 0 \end{cases}$$

$$\Omega_v = \{\ (t,S) \in [0,T] \times [0,\infty)\ \}$$



It is also well posed for the same reason we had before. We have a parabolic equation backwards in time with terminal condition.

We observe that both equations are exactly the same except for the boundary conditions related with the underlying stock.

In the vanilla option the lower boundary condition is set at S = 0, while for the barrier option it is set at S = B.

We can interpret these two conditions as the option becoming worthless when the stock touches 0 in the vanilla option, while for the barrier option it becomes worthless when the stock hits B, that is, when the option touches the barrier.

- **ii)** Prove that, if $V(t,S)$ solves the Black-Scholes PDE *(only the PDE, not the BVP!)*, then there is a unique value of γ such that the function $W(t,S) = S^\gamma V(t, H^2/S)$. Is there any restriction on the number H? *Hint: let $z = H^2/S$ and express all coefficients and derivatives in terms of z; then use the fact that $\partial V(t,z)/\partial t + (1/2)(\sigma z)^2 \partial^2 V(t,z)/\partial z^2 + \ldots = 0$.*

Let $z = \frac{H^2}{S}$, we can have that $W(t,S) = S^\gamma V(t, H^2/S) = S^\gamma V(t,z)$.

Now, the partial derivatives with the new notation result as follows:

$$\frac{dW}{dt} = S^\gamma \frac{dV}{dt}(t,z)$$

$$\frac{dW}{dS} = S^{\gamma-1}\left(\gamma\, V(t,z) - z\frac{dV}{dz}\right)$$

$$\frac{d^2W}{dS^2} = S^{\gamma-2}\left[(\gamma-1)\,\gamma\, V(t,z) - 2(\gamma-1)z\frac{dV}{dz}(t,z) + z^2\frac{d^2V}{dz^2}(t,z)\right]$$

When plugging these terms into the Black-Scholes PDE we obtain:

$$S^\gamma \frac{dV}{dt} + \frac{\sigma^2 S^2}{2} S^{\gamma-2}\left[(\gamma-1)\,\gamma\, V(t,z) - 2(\gamma-1)z\frac{dV}{dz}(t,z) + z^2\frac{d^2V}{dz^2}(t,z)\right] + rS^{\gamma-1}S\left(\gamma\, V(t,z) - z\frac{dV}{dz}\right) - rS^\gamma V = 0$$

$$\frac{dV}{dt} + \frac{\sigma^2}{2}\left[(\gamma-1)\,\gamma\, V(t,z) - 2(\gamma-1)z\frac{dV}{dz}(t,z) + z^2\frac{d^2V}{dz^2}(t,z)\right] + r\left(\gamma\, V(t,z) - z\frac{dV}{dz}\right) - rV = 0$$

By distributing the terms we arrive to the expression:

$$\left(\frac{dV}{dt} + rz\frac{dV}{dz} + \frac{\sigma^2}{2}z^2\frac{d^2V}{dz^2} - rV\right) + \frac{\sigma^2}{2}(\gamma-1)\gamma V - \sigma^2(\gamma-1)z\frac{dV}{dz} + r\gamma V - 2rz\frac{dV}{dz} = 0$$

Since V obeys the Black-Scholes PDE, we can eliminate the leftmost expression in the parenthesis.

$$\frac{\sigma^2}{2}(\gamma-1)\gamma V - \sigma^2(\gamma-1)z\frac{dV}{dz} + r\gamma V - 2rz\frac{dV}{dz} = 0$$

By working around with the terms of this expression and grouping them together we finally arrive to:

$$\left(\gamma V - 2z\frac{dV}{dz}\right)\left(\frac{\sigma^2}{2}(\gamma-1) + r\right) = 0$$

By making zero the expression in the right, we finally arrive to the value of γ such that W satisfies the Black-Scholes PDE:

$$\gamma = 1 - \frac{2r}{\sigma^2}$$

Regarding the restrictions on the number H, since H will be used to characterize the barrier of our down-and-out option later. We have to impose that $S_0 > H$ since otherwise the option would be worthless.

Notice also that since the trick we will use for valuing the option in the next items ($C_{d/o} = C_v - \frac{W}{B^\gamma}$) is derived from the method of images, we will have that S and B^2/S lie in opposite sides of the line $S = B$. This will be useful for achieving the desired conditions when the stock touches the barrier.

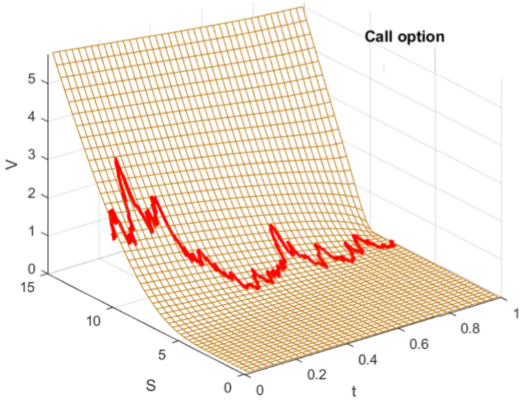
Finally we would also like to remark that $H > 0$ in general, since $H \leq 0$ would be the same as having no barrier, i.e. standard Black-Scholes.

- **iii)** Show that $V(t,H) = W(t,H)/H^\gamma$. This is obvious by just substituting for the formula we have for W:

$$\frac{W(t,H)}{H^\gamma} = \frac{1}{H^\gamma}H^\gamma V\left(t, \frac{H^2}{H}\right) = V(t,H).$$

- **iv)** Compute $\lim_{S\to+\infty} W(t,S)/H^\gamma$; and also $W(T,S)/H^\gamma$ (sketch it).

Since V describes a european call, we have that $V(t,0) = 0\ \forall t$. Also notice that the price of a call rapidly goes to 0 as S approximates 0, this can be observed in a plot of the value of a call, like the one below:



From this image, we can clearly observe that the price of the call is somewhat constant around $S = 0$, because of this, we expect the value of $\frac{dV}{dS}$ to be also really close to 0 as $S \longrightarrow 0$.
We will take into account this fact for the calculation of the limit:

$$lim_{S\to+\infty} W(t,S)/H^\gamma = lim_{S\to+\infty} \left(\frac{S}{H}\right)^\gamma V\left(t, \frac{H^2}{S}\right)$$

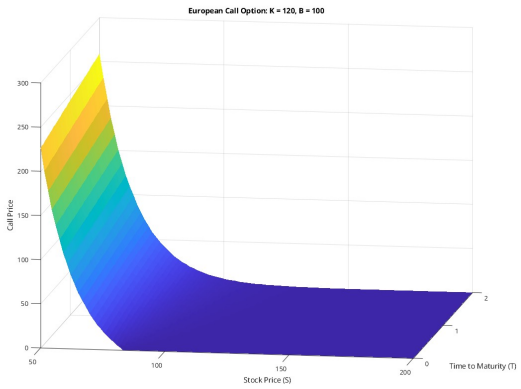
$$= H^\gamma \, lim_{S\to+\infty} \frac{V\left(t, \frac{H^2}{S}\right)}{\left(\frac{H^2}{S}\right)^\gamma} = H^\gamma \, lim_{z\to 0^+} \frac{V(t,z)}{z^\gamma} =$$

$$= H^\gamma \, lim_{z\to 0^+} \frac{\frac{dV}{dz}}{\gamma z^{\gamma-1}} = \frac{H^\gamma}{\gamma} \, lim_{z\to 0^+} z^{\frac{2r}{\sigma^2}} \frac{dV}{dz} = \frac{H^\gamma}{\gamma} 0 \cdot 0 = 0$$

For the value of $W(T,S)/H^\gamma$ we have:

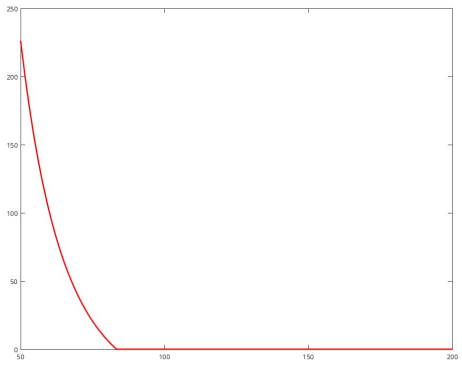
$$W(T,S)/H^\gamma = \left(\frac{S}{H}\right)^\gamma V(T, H^2/S) = \left(\frac{S}{H}\right)^\gamma (H^2/S - K)_+$$

For the sketches we have used a simple matlab script that will plot the results of the the limit and the value at expiry:



In this image we observe that the limit clearly goes to 0.

For the value at expiry we have:



We observe that when S surpasses 83.333, the value becomes zero. This is because $S = 83.333$ is the value for which $H^2/S = K$ and from that point onwards $H^2/S < K$ which makes the option worthless at expiry (its values becomes constant at 0).

- **v)** What is the BVP that $W(t,S)$ solves in $\Omega^B_{d/o}$? Prove, then, that

$$C_{d/o}(t,S) = C_v(t,S) - (S/B)^{1-2r/\sigma^2} C_v(t, B^2/S).$$

Check the formula with Matlab's built-in functions for vanilla and for the barrier. Can we also use this formula if $B > K$? Why? *Hint:* $\Omega^B_{d/o}$ *changes.*

For the BVP of $W(t,S)$ we have:

$$\begin{cases} \dfrac{\partial W}{\partial t} + \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 W}{\partial S^2} + rS\dfrac{\partial W}{\partial S} - rW = 0, \\ W(T,S) = S^\gamma \left(\dfrac{B^2}{S} - K\right)_+ < S^\gamma (B-K)_+ = 0, \\ \lim\limits_{S\to\infty} W(t,S) = 0, \\ W(t,B)/B^\gamma = V(t,B) \end{cases}$$

Now we prove that $C_{d/o}(t,S) = C_v(t,S) - (S/B)^{1-2r/\sigma^2} C_v(t, B^2/S)$.

First, $C_{d/o}$ satisfies the PDE, since it is just a linear operation on C_v & W, and both already saistfy the PDE, so any linear combination on them will satisfy the PDE too!

Now we check what occurs for the boundary and expiry conditions:

$$C_{d/o}(T,S) = C_v(T,S) - \frac{1}{B^\gamma} W(T,S) = C_v(T,S) = (S-K)_+$$

$$\lim_{S\to\infty} C_{d/o}(t,S) = lim_{S\to\infty} C_v(t,S) - lim_{S\to\infty} W(t,S)/B^\gamma = S - 0 = S$$

$$C_{d/o}(t,B) = C_v(t,B) - W(t,B)/B^\gamma = C_v(t,B) - C_v(t,B) = 0$$

Concluding the proof.

We cannot use this formula for $B > K$, since then the terminal condition $W(T,S)$, which was bounded by $(B-K)_+$ can now be distinct from 0, and then the terminal condition for $C_{d/o}$ will no longer be satisfied.

In order to check the correct functionality of the formula, we have the following simple MATLAB script:

```
Rates = 0.035;
Settle = datetime(2015,1,1);
Maturity = datetime(2016,1,1);
Compounding = -1;
Basis = 1;

RateSpec = intenvset('ValuationDate', Settle, 'StartDates', Settle, 'EndDates', Maturity, 'Rates', Rates, 'Compounding', Compounding, 'Basis', Basis);

AssetPrice = 50;
Volatility = 0.30;
StockSpec = stockspec(Volatility, AssetPrice);

Strike = 50;
OptSpec = 'call';
Barrier = 45;
BarrierSpec = 'DO';

Vanilla = blsprice(AssetPrice, Strike, Rates, yearfrac(Settle, Maturity), Volatility);
gamma = 1 - 2*Rates/(Volatility^2);
W_B = ((AssetPrice / Barrier)^gamma) * blsprice((Barrier^2)/AssetPrice, Strike, Rates, yearfrac(Settle, Maturity), Volatility);
Price = barrierbybls(RateSpec, StockSpec, OptSpec, Strike, Settle, Maturity,  BarrierSpec, Barrier);
C_do = Vanilla - W_B;

fprintf("Price using barrierbybls: "); disp(Price);

fprintf("\nPrice using our formula: ");disp(C_do);
```

The code is really simple, we have different variables that represent the different terms used in the formulas.

Rates = r

Maturity - Settle = T

AssetPrice = S_0

Volatility = σ

Strike = K

Barrier = B

We then use two commands, blsprice which is Black-Scholes pricing for european options, and barrierbybls (Barrier by Black-Scholes) which is used to compute directly the price of the barrier.

We will compute the price of the barrier directly with barrierbybls and compare the result with the value obtained by using:

$$C_{d/o}(t,S) = C_v(t,S) - (S/B)^{1-2r/\sigma^2} C_v(t, B^2/S)$$

The output obtained is:

```
Price using barrierbybls:     4.4285

Price using our formula:     4.4285
```

Which confirms that our formula is correct.

- **vi)** Does the previous equality hold if the underlying sheds dividends? And if $r = r(t)$? (but deterministic)? And if $\sigma = \sigma(t)$? (deterministic).

The previous equality will not hold if there are dividends, since the price for C_v and W is derived from the Black-Scholes formula with no dividends.

In order to add the effect of the dividends to the model we would have to modify the Black-Scholes PDE and use:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V}{\partial S^2} + (r-q) S \frac{\partial V}{\partial S} - rV = 0,$$

Where the dividends are modeled as a time-continuous cashflow $q > 0$.

Adding this new term can modify some elements of our solution for the barrier option, such as the value obtained for γ so that W satisfies the PDE.

By using non-constant r or σ the previous equality may also not hold.

This is because we have to take into account the effect that having $r(t)$ and $\sigma(t)$ has in the partial derivatives with respect to time in the PDE.

With the original Black-Scholes solution for an option namely:

The solution is then given by:

$$V_c(t,S) = S(t)F_{\mathcal{N}(0,1)}(d_1) - K\mathrm{e}^{-r(T-t)}F_{\mathcal{N}(0,1)}(d_2),$$

with

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

If we replace the terms r and σ by $r(t)$ and $\sigma(t)$, when we compute $\frac{dV}{dt}$ we will no longer obtain the same result as before. Now some new terms appear resulting from the effect of r and σ being time dependent. Which completely breaks or previous formula.

- **vii)** Using only Matlab blsprice as blackbox, write a little Matlab script which displays the 3D solution of $C_v(t,S)$ for $K < B$ and $K > B$.

```
% Parameters
r = 0.05;              % Risk-free rate
sigma = 0.2;           % Volatility
B = 100;               % Reference barrier value
T = linspace(0.01, 2, 50);      % Time to maturity
S = linspace(50, 150, 100);     % Underlying prices

% Create grids
[S_grid, T_grid] = meshgrid(S, T);

% Prepare figure
figure;
```

```
%% CASE 1: K < B
K1 = 80;
C1 = zeros(size(S_grid));
for i = 1:numel(S_grid)
    C1(i) = blsprice(S_grid(i), K1, r, T_grid(i), sigma);
end

% Plot CASE 1
subplot(1, 2, 1);
surf(S_grid, T_grid, C1, 'EdgeColor', 'none');
xlabel('Stock Price (S)');
ylabel('Time to Maturity (T)');
zlabel('Call Price');
title(['European Call: K = ' num2str(K1) ', B = ' num2str(B)]);
view(45, 30);
hold on;

% Barrier reference
fill3([B B B B], [min(T) max(T) max(T) min(T)], ...
      [min(C1(:)) min(C1(:)) max(C1(:)) max(C1(:))], ...
      [0.8 0.8 0.8], 'FaceAlpha', 0.3, 'EdgeColor', 'none');
text(B + 1, min(T), max(C1(:))*0.9, 'Barrier B', 'Color', 'k');
```

```
%% CASE 2: K > B
K2 = 120;
C2 = zeros(size(S_grid));
for i = 1:numel(S_grid)
    C2(i) = blsprice(S_grid(i), K2, r, T_grid(i), sigma);
end

% Plot CASE 2
subplot(1, 2, 2);
surf(S_grid, T_grid, C2, 'EdgeColor', 'none');
xlabel('Stock Price (S)');
ylabel('Time to Maturity (T)');
zlabel('Call Price');
title(['European Call: K = ' num2str(K2) ', B = ' num2str(B)]);
view(45, 30);
hold on;

% Barrier reference
fill3([B B B B], [min(T) max(T) max(T) min(T)], ...
      [min(C2(:)) min(C2(:)) max(C2(:)) max(C2(:))], ...
      [0.8 0.8 0.8], 'FaceAlpha', 0.3, 'EdgeColor', 'none');
text(B + 1, min(T), max(C2(:))*0.9, 'Barrier B', 'Color', 'k');
```

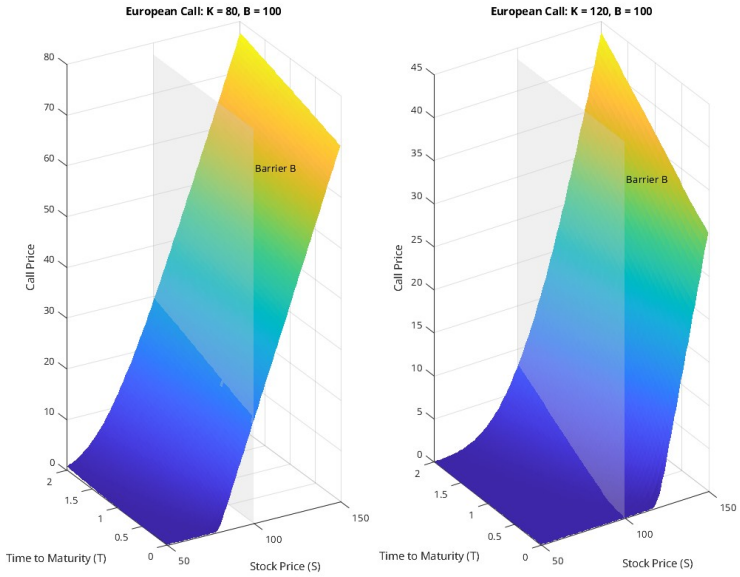
This simple code works as follows:

In the first block of code we define the parameters to use $(r, \sigma, B, T...)$.

In the second block we compute $C_v(t, S)$ for $K < B$. (K = 80) We plot the values obtained using the "surf" command and we also plot the location of the barrier for easier visualization.

The third plot is exactly the same as the previous but setting K = 120, so that $K > B$. We will also plot the location of the barrier.

We have combined both plots into a single image using the "subplot" command. The results are the following:



3 Exercise 5: OU Barrier Option

i) Simulate option with and without barriers.

Because we know that the random error made by estimating the mean of X using the sample mean of the Euler-Maruyama trajectories with a step size of h and sample size N has the form:

$$E(X) - \hat{X}_{h,N} \sim \mathcal{N}(Ch^{\delta}, \frac{\mathrm{Var}(X)}{N})$$

for a method of order $\delta > 0$, we used the provided code in order to estimate the bias error of the method, which refers to the deterministic error made by discretizing the SDE for X:

$$\text{bias\_error} = E(|E(X) - \hat{X}_{h,N}|) = Ch^{\delta}$$

by choosing $N = 10^6$, which ensures that the statistical error is negligible relative to the deterministic bias one we are interested in.

Moreover, because we know from the theory that raw Maruyama has a weak convergence rate of $\delta = 1$ on vanilla options and $\delta = 0.5$ on barrier options (like the down-out case here) regardless of the underlying, we used the code `ousscratch.m`, which estimates the bias error by using δ and the sample means with h and $h/2$.

We then plotted the estimates for varying time steps on a log-log plot, and as we expected, the logarithm of errors decreases to 0 at constant rates of 1 and 0.5, aligning with theoretical expectations:

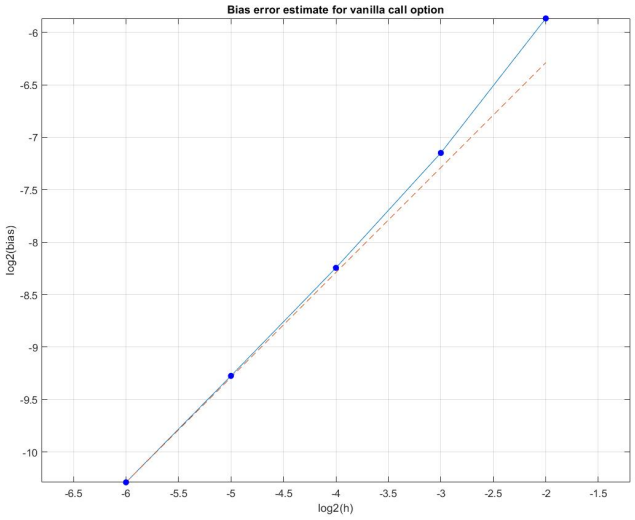


Figure 17: Order 1 for vanilla options (reasonable)

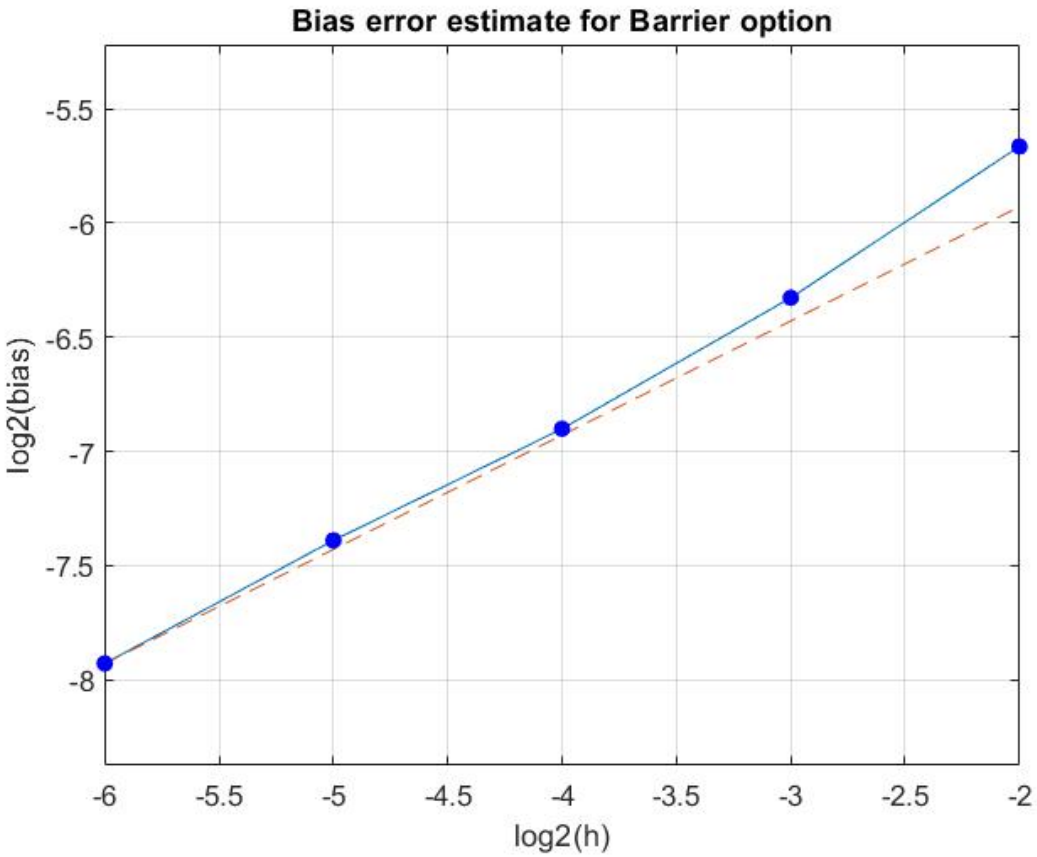


Figure 18: Order 0.5 for vanilla options (very slow)

i) Apply Brownian bridge and barrier shifting correction. We implemented and optimized the way in which we performed the simulations adapted to the Maruyama + Brownian Bridge technique using the following code in Matlab:

```
N = 10000000;
```

```
for i=1:length(M) %different time step
    dt= T/M(i);
%     S= NaN(N,M(i)+1);
    S = zeros(N,2);
    S(:,1)= S0;
    B= B0;
    %B= B0+0.5826*sigma*sqrt(h); %upwards in this problem
    %% EULER MARUYAMA WITH BROWNIAN BRIDGE
    product = ones(N,1); %compound product from brownian bridge method discount
        factor
%     dW = randn(N,M(i)); %wiener increments matrix (each row is a path)
    for j=1:M(i)
        %% maruyama+bbridge using only N*2 matrix (current and next)
        dW = randn(N,1);
        S(:,2) = S(:,1) + kappa*(theta-S(:,1))*dt + sigma*sqrt(dt)*dW(:,1);
        product = product .*(1-exp((-2*max(0,S(:,2)-B0).* max(0,S(:,1)-B0))/(sigma
            *sigma*dt)));
        S(:,1) = S(:,2);
%         S(:,j+1)= S(:,j) + kappa*(theta-S(:,j))*dt + sigma*sqrt(dt)*dW(:,1);
%         t = j*dt;
%         std =  sqrt((sigma^2*(1-exp(-2*kappa*t)))/(2*kappa));
%         S(:,j+1) = (S0*exp(-kappa*t)+theta*(1-exp(-kappa*t)))*ones(N,1)+ std*
    randn(N,1);
%         product = product .* (1-exp((-2*max(0,S(:,j+1)-B0).* max(0,S(:,j)-B0))/(
    sigma*sigma*dt)));
    end
    payoff= zeros(N,1);
    payoff= exp(-r*T)*max(0,S(:,end)-K).*product; %discounted payoffs
    V(i)= mean(payoff); %option price estimate
    varsc(i)= var(payoff);
    ster(i)= 3*sqrt(varsc(i)/N); %standard error (99 % sure not more than this)
end
```

We decide to show extra lines of commented code to emphazise how we attacked the problem initially, where we would compute the payoffs inspired by some of the vectorized code in the derfault file of the exercise but then realised that we could keep the vectorisation spirit in the Brownian bridge upgrade but without needing to rely on a full matrix with paths data in each row. Essentially we just need to have two column vectors: one for the prices in the previous instant and the second with the next time prices, since this is enough to compute the dying probabilities in Brownian bridge. Finally note that we use a sample size of 10 million paths and execute the respective estimates using different step sizes which are inside the row vector M

ii) Brownian bridge implementation and error analysis:

From Mike Gile's lecture slides we knew a priori that the addition of Brownian bridge should increase the ridiculously slow 0.5 order of convergence in barrier options (as seen above) up to a decent order 1. However, in order to not to blindly apply the heuristic bias estimate formula obtained by time step halving given by:

$$e_{h,N} = Ch^{\delta} \approx \frac{\hat{X}_{h/2,N} - \hat{X}_{h,N}}{1 - 2^{-\delta}} \tag{7}$$

which requires knowledge about the weak convergence rate δ.

We noticed that for large N, the bias is the predominant error and we can use (7) with step size $h/2$ to obtain the expression:

$$e_{h/2,N} = C(\frac{h}{2})^{\delta} \approx \frac{\hat{X}_{h/4,N} - \hat{X}_{h/2,N}}{1 - 2^{-\delta}} \tag{8}$$

Hence, if we divide (7) by (8) and take log2 on both sides we obtain:

$$\delta \approx \frac{\hat{X}_{h/2,N} - \hat{X}_{h,N}}{\hat{X}_{h/4,N} - \hat{X}_{h/2,N}} \tag{9}$$

And we can certainly use our code to then check whether that ratio is close to 1 for several exeuctions in Matlab, ensuring that the sample size is large so that no important fluctuations occur, away from the target order δ.

This was used to justify and statistically prove that the Maruyama + Brownian bridge method has indeed order 1 as hinted in Mike Gile's slides. We did this to not blindly apply the formula (6) for the bias estimation as that only makes sense to do when one knows the order of convergence.

Running the code above 10 times with varying step sizes to use (9) and a sample size of 10 million we calculated the ratios in each case and found them to be really close to 1 , "proving" (at least an statistical proof) that the weak convergence is of order 1 in this new scheme!

```
C =

   1.037534039186448
   0.952399695574950
   0.930284782630056
   0.890796467752352
   0.943431739434936
   0.971552326188480
   1.014532482563485
   0.865538470870799
   1.035540413844094
   0.808670696257613
```

Figure 19: 10 ratios realisations with N = 10M

```
>> mean(C)

ans =

   0.945028111430321

>> var(C)

ans =

   0.005601276482084

>> sqrt(var(C))

ans =

   0.074841676104188
```

Figure 20: Mean,Variance of the ratios

Following this discovery that the method was of order 1 we were in good position to estimate the bias errors with each time step using time step halving. Then we obtained the following error estimates in a log log plot:

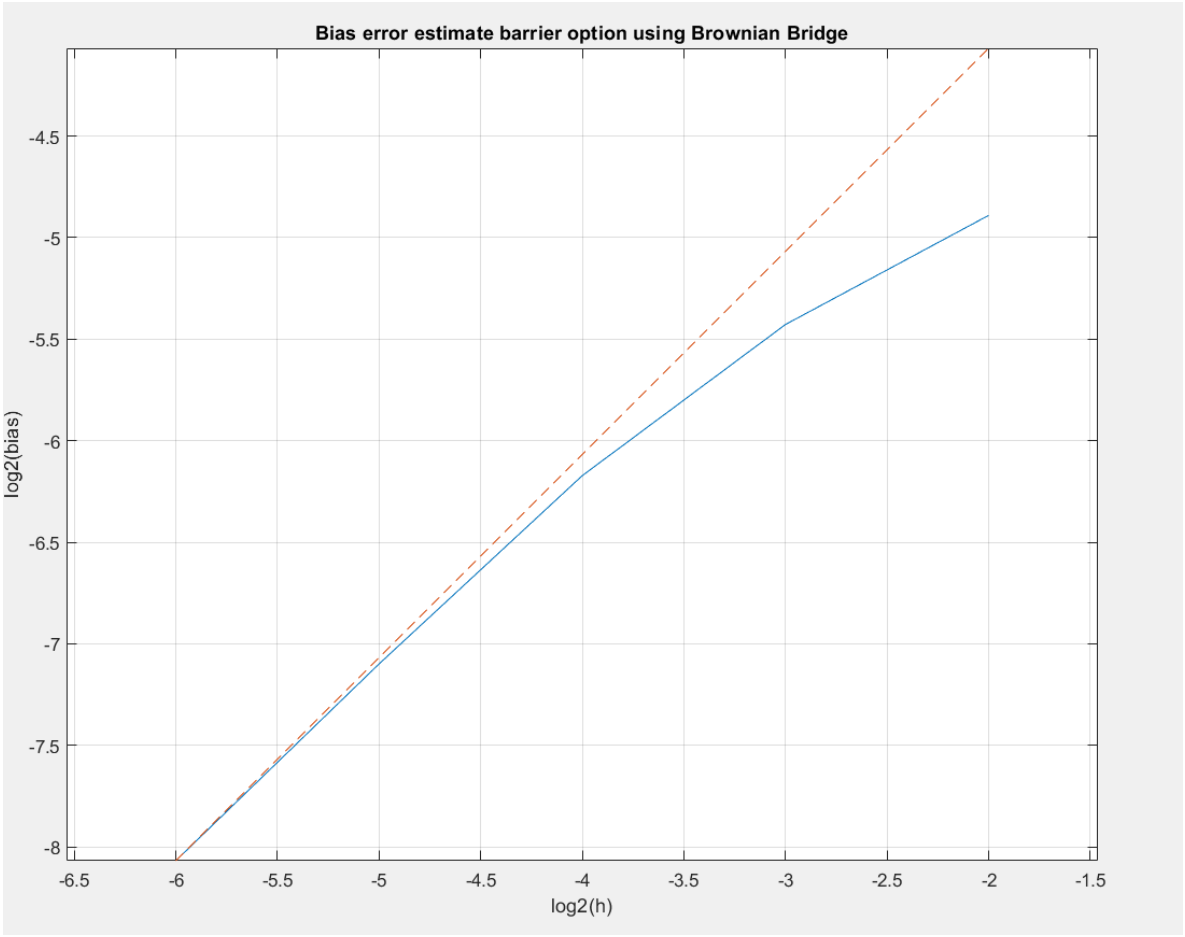


Figure 21: Maruyama with Bridge becomes order 1 !

iii) Barrier shifting analysis:

Since we're dealing with a down-out option, we decided to raise the barrier to make sure we account for price movements between the moments we check. In reality, the asset price moves continuously, but since we're working with a step-by-step simulation, there's a chance we might miss brief drops below the barrier that should knock out the option.

By increasing the barrier, we're making the system more cautious, so that paths that might have briefly crossed below in real life actually get knocked out in our simulation. This helps ensure our results reflect the true behavior of the option, rather than making it seem like fewer knockouts happen than they really should.

The results were similar to those for the Brownian bridge incorporation, obtaining a weak convergence rate of 1, after verifying by computing our technique (9) and later plotting the convergence with respect to the time steps in a log log plot:

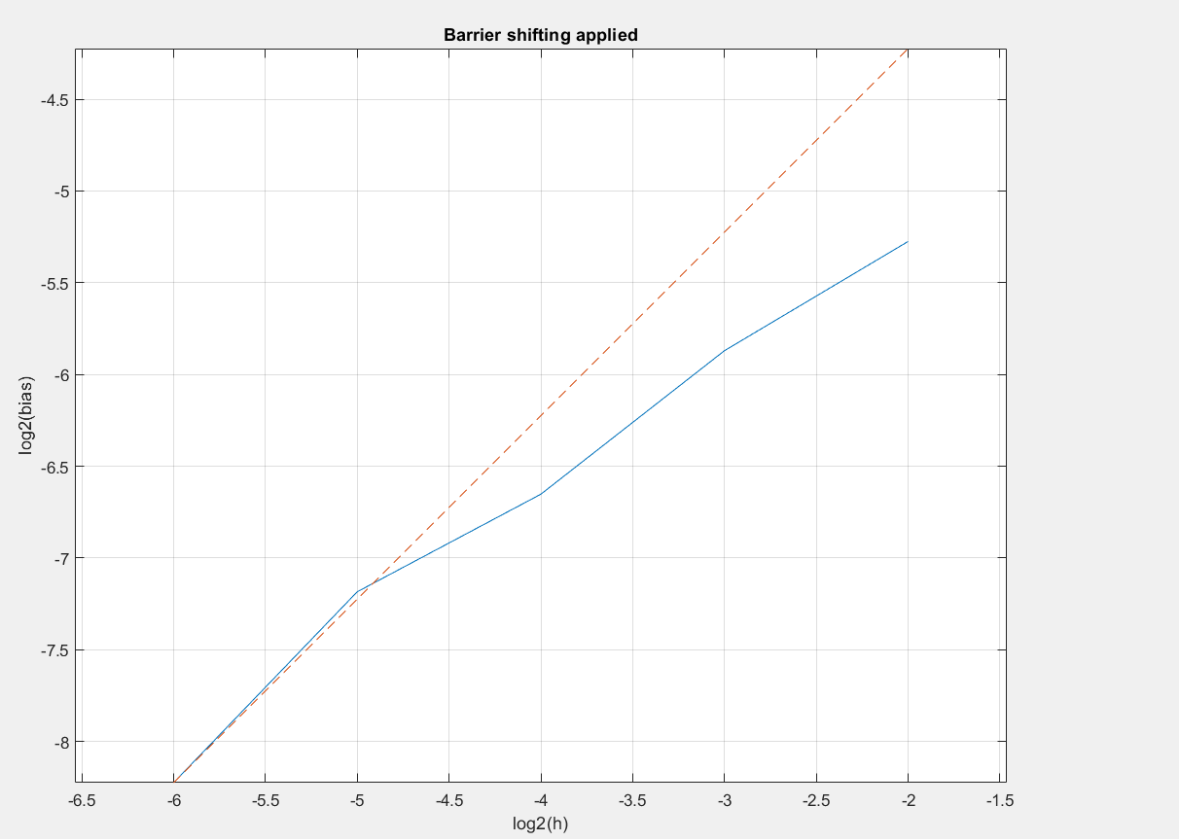


Figure 22: Increasing barrier keeps order 1

Exercise 6: Barrier Option under GBM

Let the underlying asset follow a Geometric Brownian Motion with parameters $S_0 = 100$, $K = 100$, $B = 85$, $r = 0.1$, $\sigma = 0.3$, and $T = 0.2$. The goal is to evaluate the price of a European down-and-out call option under different numerical approaches and compare them to the known analytical value $\boxed{C_{d/o}^{\text{exact}} = 6.3076}$.

(i) Exact Price via Reflection Principle

The exact price can be derived using the image method. For $B < S_0 < K$, the price of a down-and-out call is given by:

$$C_{d/o}(S_0) = C_{\text{BS}}(S_0) - \left(\frac{S_0}{B}\right)^{\alpha} C_{\text{BS}}\left(\frac{B^2}{S_0}\right)$$

where $\alpha = 1 - \frac{2r}{\sigma^2}$ and $C_{\text{BS}}(S)$ is the Black-Scholes price of a European call with spot S. The result using MATLAB confirms:

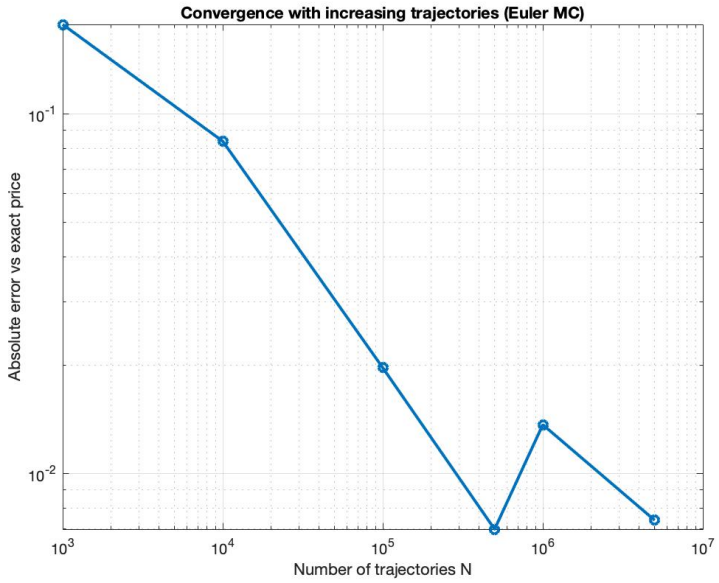
$$\boxed{C_{d/o} = 6.3076}$$

(ii) Monte Carlo using Euler-Maruyama on S_t

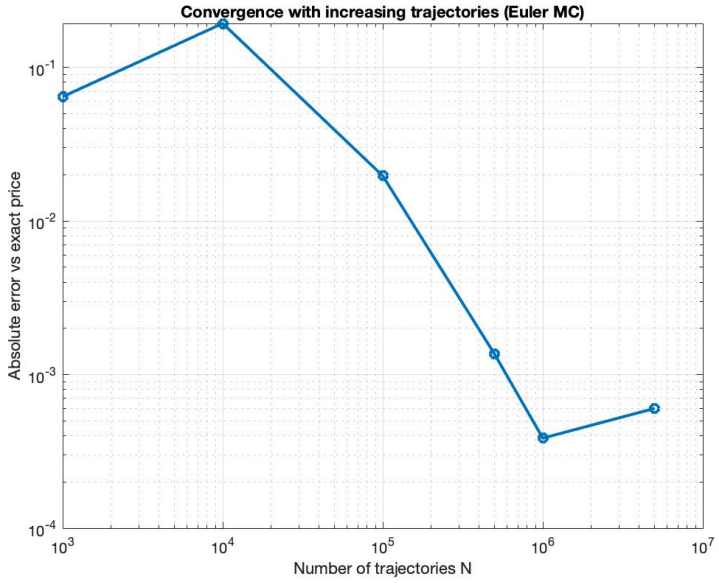
A basic Euler-Maruyama discretization of S_t was used with $M = 100$ time steps and varying number of trajectories N. If the barrier is breached at any step, the payoff is set to zero. Otherwise, the standard discounted call payoff is used.

Setup. We fixed the number of time steps to $M = 100$ and varied the number of simulated trajectories N in:

$$N \in \{10^3, 10^4, 10^5, 5 \times 10^5, 10^6, 5 \times 10^6\}$$



The results show convergence of the Monte Carlo estimate as N increases, with fluctuations due to variance and discretization bias. The following plot confirms improved behavior for smaller time steps essentially removing the fluctuations due to the error coming from the timestep:



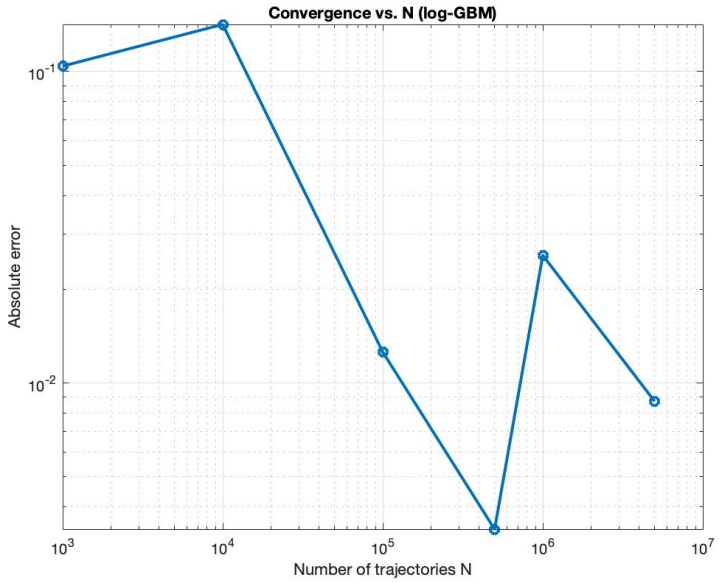
(iii) Monte Carlo using log-SDE

In this variant, the simulation is performed directly on the logarithm of the asset price, i.e., on $X_t = \log S_t$. The discretized log-SDE is given by:

$$X_{t+1} = X_t + \left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z_t,$$

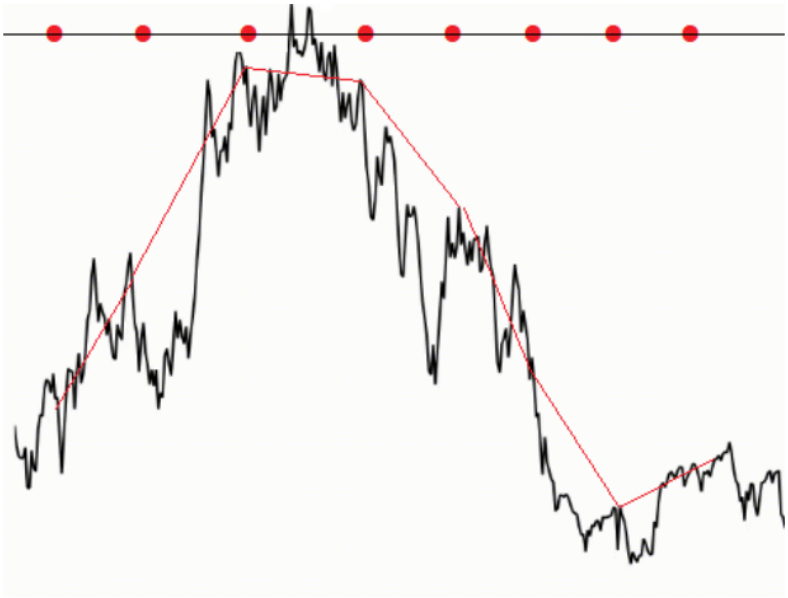
where $Z_t \sim \mathcal{N}(0,1)$. This formulation improves numerical stability.

To enforce the barrier condition in down-and-out options, we monitor $S_t = \exp(X_t)$ at each discrete time step. The payoff is computed as in (ii), meaning the option is only valid if the asset price never breaches the barrier B during the simulation.



The convergence trend is similar to that of the standard Euler scheme (ii), though the logarithmic formulation offers more robustness in terms of numerical stability. However, the method still suffers from discretization bias in the barrier monitoring. Specifically, we may miss barrier crossings that occur between discrete time steps, since we do not model the underlying Brownian bridge.

This can result in an underestimation of knock-outs, leading to a biased overestimation of the option price. The figure below illustrates this issue: although the simulated points remain above the barrier, the true continuous path may have crossed it.



(iv) Monte Carlo Simulation with Barrier Shifting (log-GBM)

This experiment refines the approach from Exercise 6(iii) by introducing a barrier shifting technique to mitigate the bias caused by discrete monitoring of the barrier. Essentially what we are doing is considering the barrier to be slightly downwards to account for possible missed barrier crossings between timestep

Method. We simulate the asset using the logarithmic SDE for GBM:

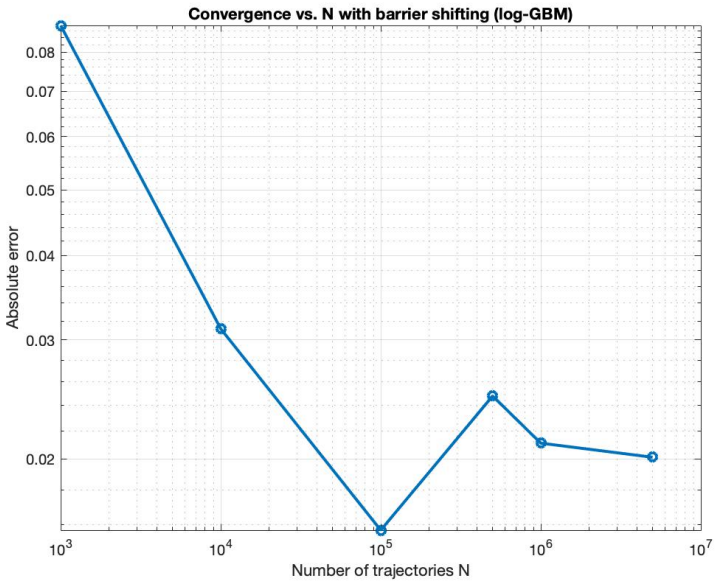
$$\log S_{t+1} = \log S_t + \left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} Z_t, \quad Z_t \sim \mathcal{N}(0,1)$$

The original barrier level B is replaced by an effective barrier:

$$B_{\text{eff}} = B \cdot \exp(-0.5826 \cdot \sigma \cdot \sqrt{\Delta t})$$

which lowers the threshold at each discrete time step to partially correct the missed-barrier-breach bias. If $S_t \leq B_{\text{eff}}$ at any time step, the payoff is zero; otherwise, we compute the standard discounted call payoff.

Results.



The plot shows the absolute error between the Monte Carlo estimates and the exact solution. As expected, the error decreases significantly with increasing N at first, but stabilizes around 0.015 for large values of N. This behavior is typical of simulations affected by residual bias: increasing the number of trajectories reduces the statistical variance, but not the discretization error introduced by the time step M.

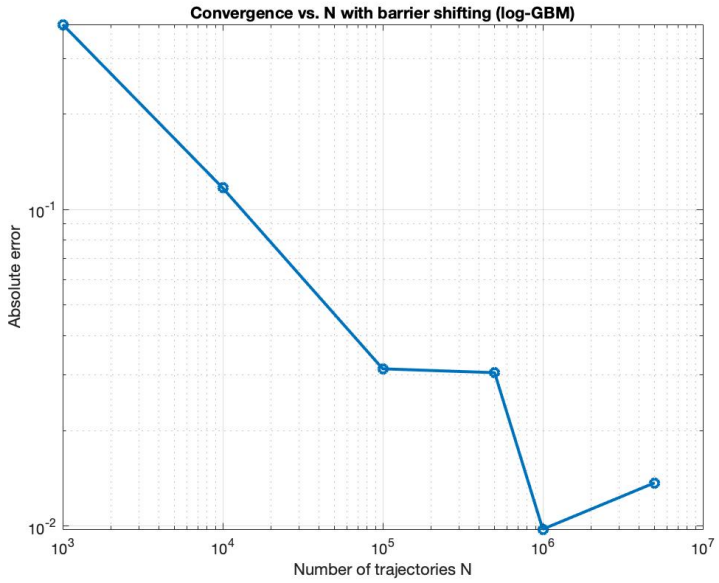
Discussion. The barrier shifting method effectively improves accuracy over the naïve log-GBM method from Exercise 6(iii), particularly at moderate N. However, because the time step M is fixed, the bias due to discrete monitoring remains non-negligible. This explains why the error plateaus beyond a certain number of trajectories. While the error does not vanish entirely, it remains significantly below that of the Euler or basic log-GBM methods without correction.

This method strikes a good balance between efficiency and accuracy and is simple to implement. However, techniques like the Brownian bridge correction (Exercise 6(v)) offer even better bias control at similar computational cost.

Refinement with Smaller Time Step. To investigate the impact of discretization bias, we repeated the simulation using a fivefold finer time resolution (i.e., increased M from 100 to 500). The barrier was re-shifted accordingly at each time step, using:

$$B_{\text{eff}} = B \cdot \exp(-0.5826 \cdot \sigma \cdot \sqrt{\Delta t})$$

This change significantly reduced the time-discretization bias, as evidenced in the following plot:



Compared to the previous figure, the absolute error now reaches values below 10^{-2} for $N \geq 10^6$, clearly outperforming the coarser time step. The convergence is more consistent, and the error plateau observed earlier is greatly diminished.

(v) Monte Carlo Simulation with Brownian Bridge Correction (log-GBM)

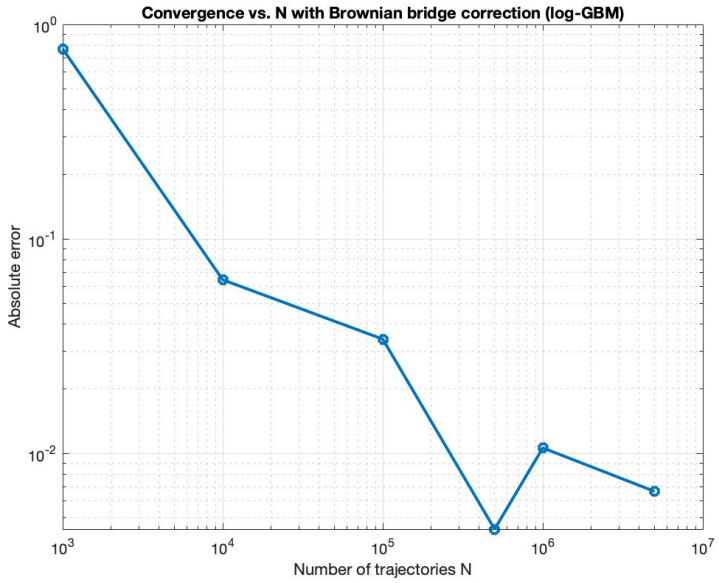
This experiment introduces a more sophisticated correction for barrier crossings in discrete-time simulations: the Brownian bridge method. Unlike the simple barrier shifting heuristic, this approach estimates the probability that the continuous path crosses the barrier between two discrete points.

Method. We simulate the asset using the logarithmic GBM model, but at each time step we estimate the probability that the path has crossed the barrier between X_t and X_{t+1} using the Brownian bridge formula. The probability that a Brownian bridge between two points X and X_{next} stays above the barrier $\log B$ is:

$$P_{\text{survival}} = 1 - \exp\left(-\frac{2(X - \log B)(X_{\text{next}} - \log B)}{\sigma^2 \Delta t}\right)$$

If either endpoint is already below the barrier, or a uniform random variable is below $1 - P_{\text{survival}}$, the trajectory is considered breached and discarded.

Results.



The figure shows a significant reduction in absolute error compared to previous methods. The Brownian bridge correction effectively eliminates most of the bias caused by discrete monitoring of the barrier, allowing the Monte Carlo estimate to converge to the correct value as N increases.

Discussion. This method demonstrates superior accuracy even for moderate values of N. Errors consistently fall below 10^{-2} for $N \geq 10^5$, and convergence is more stable and predictable than with barrier shifting. While slightly more complex computationally, the Brownian bridge offers an excellent trade-off between precision and performance, especially when high accuracy is desired without reducing the time step M.

Among all previous approaches, this one is the most faithful to the continuous-time barrier condition while still using discrete simulation steps.

(vi) Monte Carlo with Exact GBM and Analytical Survival Probability

In this approach, we simulate the terminal value S_T of the asset using the exact solution of the geometric Brownian motion (GBM), thus avoiding discretization via Euler schemes. To account for the down-and-out barrier feature, we introduce a survival probability that analytically models the likelihood that the path did not touch the barrier B during the time interval $[0, T]$.

The analytical survival probability is given by:

$$P_{\text{survival}} = 1 - \exp\left(-\frac{2\log(S_0/B)\log(S_T/B)}{\sigma^2 T}\right)$$

This survival probability comes from the reflection principle. A rejection step is then used: for each simulated path, a uniform random number $U \sim \mathcal{U}(0,1)$ is drawn, and the path is accepted only if $U < P_{\text{survival}}$. The final payoff is computed as:

$$\text{Payoff} = e^{-rT} \cdot \max(S_T - K, 0) \cdot \mathbf{1}_{\text{alive}}$$

This method is highly efficient and avoids the underestimation of barrier hits that often arises from time discretization. The following plot shows the absolute error of the Monte Carlo estimator for varying numbers of trajectories N compared to the known exact price of the option (6.3076):

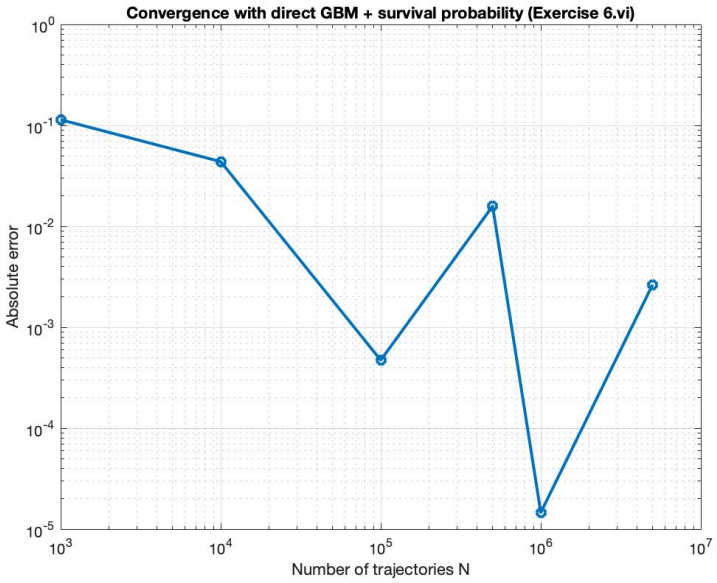


Figure 23: Convergence of the Monte Carlo estimator with exact GBM sampling and survival probability.

As shown in Figure 23, the absolute error generally decreases as N increases, following the expected $\mathcal{O}(1/\sqrt{N})$ convergence rate. Notably, some fluctuations are observed (e.g., at $N = 5\times10^5$ and 5×10^6), which are due to the inherent randomness of Monte Carlo simulations. The lowest error is achieved at $N = 10^6$, where the method provides highly accurate results.

This confirms that using exact GBM terminal sampling in combination with the analytical survival probability leads to a robust and accurate pricing scheme for barrier options.

(vii) Efficiency comparison across barrier option methods

Each method is implemented as follows:

- **Euler-S:** Standard Euler-Maruyama discretization applied directly to S_t.
- **Euler-logS:** Euler discretization of $\log S_t$, to improve numerical stability.
- **Shifting:** Barrier shifting approximation using an effective barrier B_{eff}.
- **Bridge:** Brownian bridge-based correction that estimates barrier crossing probabilities between time steps.
- **Exact:** Direct sampling of the terminal value S_T and rejection based on an analytical survival probability formula.

For each method, simulations are performed for different values of N (from 10^3 to 5×10^6). For each run, we compute:

- The **absolute error**, i.e., $|\hat{P} - \texttt{exact}|$
- The **computational time** in seconds (measured with `tic` and `toc`).

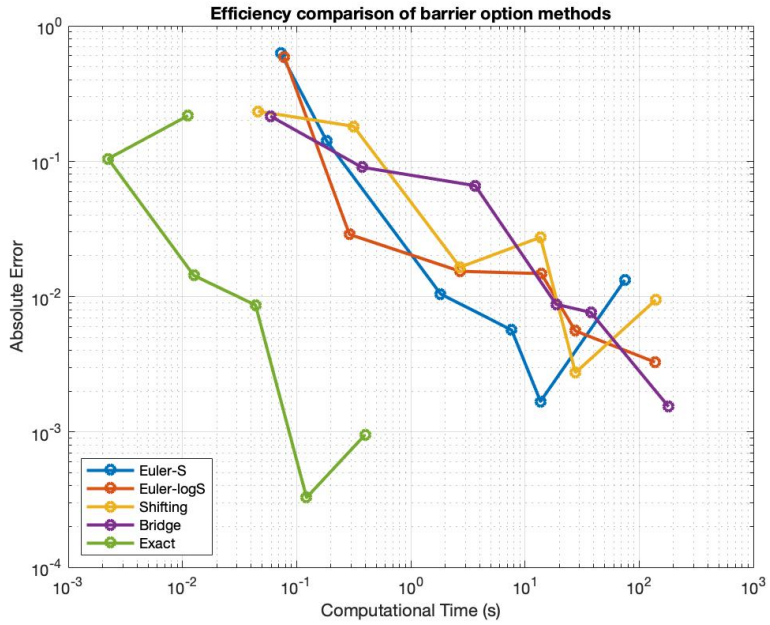


Figure 24: Efficiency comparison of barrier option pricing methods. Log-log plot of absolute error vs computational time.

Discussion: Figure 24 shows the trade-off between computational cost and accuracy for each method.

- The **Exact** method (green) clearly outperforms the rest: it reaches the lowest error with the least computation time. This is due to its use of exact sampling and analytical correction for barrier crossing.
- **Bridge** (purple) performs well in accuracy, but requires significantly more computation, especially for large N.
- **Shifting** (yellow) improves slightly over naive Euler, but may introduce bias due to the approximation of the effective barrier.
- **Euler-logS** (red) and **Euler-S** (blue) are the least efficient. They require many trajectories to reduce the error and are sensitive to discretization effects, especially when simulating barrier crossings.

(viii) Can you get even better speed with variance reduction?

Yes, variance reduction techniques can significantly improve the efficiency of Monte Carlo estimators, allowing us to achieve the same level of accuracy with fewer trajectories — and thus faster computation times.

In Exercise 6(vi), we already use a highly efficient approach by sampling the terminal value S_T exactly from the geometric Brownian motion (GBM) and applying an analytical survival probability. However, additional improvements can still be achieved using the following variance reduction techniques:

- **Antithetic variates:** For each standard normal sample Z, we also simulate $-Z$, and average the two payoffs. These two paths are negatively correlated, which reduces the variance of the estimator without increasing the number of random samples.
- **Control variates:** We can use the European call option payoff (whose expected value under Black-Scholes is known) as a control variate. The barrier option and the vanilla call are correlated, so adjusting the barrier estimator by the deviation in the control improves accuracy.

Conclusion: The experiment confirms that methods incorporating analytical information about barrier crossings (Bridge and Exact) achieve better accuracy-efficiency trade-offs. In particular, the Exact method is the best candidate when performance is critical.

4 Conclusion

Summarize findings, compare methods, and discuss implications of variance reduction and bias correction in practice.