Correctly Pricing Continuous Barrier Contracts

Felix EYCHENNE (CID: 02299994)
Imperial College London
Department of Mathematics

August 2023

Declaration & acknowledgment

The work contained in this thesis is my own unless otherwise stated. I would like to thank Mr Galbraith for giving me the opportunity to work on this project and for all his valuable contributions.

Abstract

This project shows how to price contracts on assets with a barrier using the Monte-Carlo paradigm. Because we suppose the barrier continuous and not discrete we cannot price those contracts using only discrete monitoring of the underlying (i.e. looking if the sampled paths breached the barrier at discrete times). Doing so underestimates the probability of an asset getting knocked-out (or knocked-in) and results in a positive bias in the simulated price. We show first how to offset this bias in the case of a single underlying using known techniques and details how to modify the Monte-Carlo paradigm to quicken the pricing. We then show that this bias is still present in the case of several underlyings each with its own barrier and outlines techniques to get the right price. We also show how to greatly speed up the pricing by combining one of those techniques with a modified Monte-Carlo algorithm. Lastly, we introduce an alternative method to price barrier contracts on two correlated assets that relies on the Law of Large numbers and uses a lot of pre-processing calculations so that the actual pricing is very reliable and actually quicker than other known techniques. We discuss it in detail and propose several axes of research to adapt it for more than two assets. All the techniques and discussions in this paper are illustrated with numerical results and figures.

Keywords: Barrier options; Brownian bridge; Monte-Carlo; Barrier shifting; Copula; Correlation; Neural Network

.

Contents

List of Figures

Introduction

Banks and financial institutions are always in a competition to find better ways to price financial products. Among these products, derivatives are contracts tied to different financial assets, and their potential payouts are only limited by traders' imaginations. Due to the complexity and diversity of these contracts, there has been a growing need for faster and more accurate numerical methods in recent decades. One of the popular methods is the Monte Carlo approach, which has become a key player in finance. It has been extensively studied and is now well-understood, making it handy for pricing various derivatives. However, the Monte Carlo method has its limitations, both in terms of performance and conceptually. These limitations require specific solutions, especially for contracts like barrier options, which are among the most commonly traded derivatives. Barrier options provide a payout if an underlying asset does not cross a certain level during the contract's duration; otherwise, there is no payout. Because of the way Monte Carlo works, it can be tricky to efficiently and reliably price these contracts. That is why various numerical techniques have been developed to tackle this challenge.

In this work, we focus on addressing these issues for barrier options. We start with the basics in Chapter 1, looking at barrier contracts related to a single asset. We detail the Brownian bridge techniques and the barrier shifting techniques. We will also show how the Brownian bridge technique can be combined with the quite recent Multi-Level Monte-Carlo paradigm to create a dependable way of pricing barrier options.

In Chapter 2, we expand our scope to contracts involving multiple assets with correlation structures. Specifically, we detail the copula method that derives two bounds for the price of the option and we see how the barrier shifting techniques can be adapted in this case. Lastly, we merge the Multi-Level Monte-Carlo paradigm and the copula technique to allow for fast and accurate pricing.

Finally, the third chapter details an alternative technique to price those contracts that makes up to some of the weaknesses of the existing ones in the case of two correlated assets. It uses a lot of pre-processing to produce a mathematical object then used to price those contracts. It also discusses how to extend this technique for more than 2 assets and proposes axes of reflexion.

Chapter 1

The single asset case

This chapter deals with the single asset case. We start with a contextualization of the problem, then show how to solve the simplest case (i.e a Brownian motion). We then build on complexity and show how to tackle real-world situations.

1.1 Barrier options: first results

We start with basic definitions. Let us consider an asset S (real or fictional): this asset can be a stock (i.e. Tesla) or an index (i.e. S&P 500). A digital option on this asset gives at maturity a payoff P that is determined at the settlement date or at maturity given that the asset did/ did not cross a certain region during the lifetime of the option.

1.1.1 Hypothesis and mathematical context

Most of the time this region is a constant barrier B (discrete or continuous), and we call a down-and-out digital option on the asset the digital option that pays P if the asset did not go under B during the lifetime of the option and 0 otherwise. This specific form of contract (digital down-and-out) will be one of our main focus in this thesis. More specifically, suppose the dynamic of the asset price follows the standard model

$$dS_t = r_t S_t dt + \sigma_t S_t dW_t^{\mathcal{Q}} \tag{1.1}$$

where r is the interest rate process, σ is the volatility process and W^Q is a single Brownian motion taken under the standard risk-neutral measure $\mathcal{Q}$. To keep the results of this work general enough and simplify the notations, we do not take into account credit risk and dividend in (1.1). Suppose at t the present date the asset price is known equal to $S(t) = S_t$. A digital barrier option with maturity T, domain $\mathcal{D}$ and notional P has its payoff given by

$$Payoff = \begin{cases} P \text{ if S stayed in } \mathcal{D} \text{ between t and T} \\ 0 \text{ otherwise} \end{cases} \tag{1.2}$$

If the domain is a single constant line B (continuous or not) below S_t as is often the case and will be our main focus, the equation (1.2) becomes

$$Payoff = \begin{cases} P \text{ if S stayed above B when the barrier is monitored between t and T} \\ 0 \text{ otherwise} \end{cases} \tag{1.3}$$

One should pay extra attention if the barrier is continuously monitored or is discretely monitored. The latter case corresponds to the case where the barrier becomes a set of point: the asset is therefore allowed to go under the constant level B between two points. In this thesis, we will tackle the case where B is continuous which implies that the asset should not go below B independently of the monitoring points chosen between t and T. The reason why we do not deal with the discrete case is because Monte-Carlo pricing for this kind of barrier is straightforward. Therefore from now onward and except mention of the contrary B designs a continuous barrier below S_t and (1.3) simply becomes

$$Payoff = \begin{cases} P \text{ if S stayed above B between t and T} \\ 0 \text{ otherwise} \end{cases}$$

Another contract of interest for us is the down-and-out call option. Given B, T, a notional P and a strike K, the payoff can be written as

$$Payoff = \begin{cases} P \times (S_T - K)^+ \text{ if S stayed above B when the barrier is monitored between t and T} \\ 0 \text{ otherwise} \end{cases} \tag{1.4}$$

The difference between (1.4) and (1.3) lies simply in the payoff but there is no major difference between the two (given we can simulate S until T). To simplify things further, notice that under our hypothesis the price of a barrier option with notional P is simply P times the price of the same barrier option with notional 1 (to notice this simply write the payoff and use a non-arbitrage argument). Therefore from now on we will only be interested in the price of barrier options with notional 1: given the dynamics of the underlying, the price of a barrier option is now a function only of the barrier level B, the maturity T and of K if it is not of digital type.

1.1.2 Closed form formula under the Black-Scholes model

As for the vast majority of contracts, there is no closed-form formula for the price of barrier options, digital or not under (1.1). One has to rely on numerical methods such as PDE, Monte-Carlo, etc. This work deals with the latter and we need to compare our results with benchmark ones under different configurations of K, B and T to assess the quality of our methods. The Black-Scholes model provides a good starting point to do so since there exists closed-form formulas. The derivation of those results can be found in [1] for the digital down-and-out option and in [2] (Chapter 25, p.579) for the down-and-out option and we give here the desired results.

Digital down-and-out barrier option

We let $\tau = T - t$ and the value of the option at time t is given by

$$DDO(S_t, \tau, B) = D_K^+(S, \tau) - \mathcal{I}(D_K^+(S, \tau)) \tag{1.5}$$

where $D_K^+(S, \tau) = e^{-r\tau}\mathcal{N}(d_-)$ and $\mathcal{I}(D_K^+(S, \tau)) = (\frac{S}{B})^{2\alpha} D_K^+(\frac{B^2}{S}, \tau)$ with $\alpha = \frac{1}{2} - \frac{r}{\sigma^2}$.

Down-and-out barrier option

Let $C_{BS}(K, S_t, \tau)$ be the price of a call under the Black-Scholes model with the usual notations. Then the price of the option is given by:

$$DOC(S_t, \tau, B, K) = C_{BS}(K, S_t, \tau) - (\frac{S_t}{B})^{1-2\frac{r}{\sigma^2}} C_{BS}(K, \frac{B^2}{S_t}, \tau) \tag{1.6}$$

Note that those formulas also allow to compute the price of a digital down-and-in barrier options and down-and-in barrier options using the (model independent) no-arbitrage relations:

$$1 = DDO(S_t, \tau, B) + DDI(S_t, \tau, B) \tag{1.7}$$

and

$$C_{BS}(K, S_t, \tau) = DOC(S_t, \tau, B, K) + DIC(S_t, \tau, B, K) \tag{1.8}$$

where DDI stands for "digital down-and-in" and DIC for "down-and-in call".

Note: In the case of a continuous dividend yield q, just replace r in the above relations by r-q.

1.2 The naive pricing procedure

This section details the numerical procedure used in this work and gives an overview of the principal limitations to overcome. We for now show how to price the barrier contracts using crude Monte-Carlo, and will refine our methodology in subsequent sections.

1.2.1 The Euler-Maruyama scheme

In this work, we chose to simulate (1.1) with the Euler-Maruyama scheme. Although it is the simplest numerical scheme to simulate a SDE, it is robust and very generic when the Milstein method already requires the volatility term in (1.1) to be C^1 which implies obvious limitations when one is not working with the simpler Black-Scholes model. To obtain good results using the Euler-Maruyama scheme, one only needs to make sure the drift and volatility terms in (1.1) verify the following assumption (given in [3] (page 1)). If we suppose they are of the standard form $(t,x) \longrightarrow r(t,x)$ and $(t,x) \longrightarrow \sigma(t,x)$ then:

Assumption 1. *The coefficients in* (1.1) *are globally Lipschitz and they satisfy a linear growth condition; there exists K such that for all $x, x' \in \mathbb{R}$:*

$$|r(t,x)x - r(t,x')x'| + |\sigma(t,x)x - \sigma(t,x')x'| \leqslant K|x - x'|$$

and

$$|r(t,x)x| + |\sigma(t,x)x| \leqslant K(1 + |x|)$$

In addition, S_t is independent of W^Q and $\mathbb{E}(S_t^2) < \infty$.

This assumption seems perfectly reasonable for most of the drift and volatility terms one is expected to work with in finance. Under this assumption, we recall the two main theorems related to the Euler-Maruyama scheme, where a formal derivation can be found in [3] (pages 2-3) for the strong convergence and in [4] (pages 35-41) for the weak convergence for example:

Theorem 1. *If Δt is the time step used in the Euler-Maruyama scheme and $\hat{S}^{\Delta t}$ designs the simulated piece wise constant asset path under this scheme, then under assumption 1 there exists a positive constant C such that*

$$\underset{s \in [t,T]}{sup}\ \mathbb{E}(\hat{S}^{\Delta t}(s) - S(s)) \leqslant C\sqrt{\Delta t}$$

In other words the Euler-Maruyama scheme is strongly convergent of order $\sqrt{\Delta t}$, which means under our assumption and given a sufficiently small time step the simulated path and the exact solution should not differ significantly. We also have the following theorem this time for the weak convergence

Theorem 2. *If Δt is the time step used in the Euler-Maruyama scheme and $\hat{S}^{\Delta t}$ represents the simulated piecewise constant asset path under this scheme, then under assumption 1, there exists a positive constant C such that*

$$\mathbb{E}(f(\hat{S}^{\Delta t}(T)) - f(S(T))) \leqslant C\Delta t$$

where f is a Lipschitz function.

The latter theorem deals with statistical properties of the simulated random variable S at time T. Intuitively, this means that the convergence for a non path-dependent option should be really good (in particular there is no difference between the Milstein and the Euler-Maruyama scheme in this regard).

However we deal with barrier options which are by nature path-dependent, and the first theorem comes into play. Since we would like the error related to the numerical scheme as small as possible, this implies choosing a fine time grid. We therefore choose to simulate $\hat{S}^{\Delta t}$ with a daily time step: if we want to price a contract with maturity $T-t$ where $T-t$ refers to the number of days between settlement and maturity, then every path in our Monte-Carlo simulations will consist of $T-t+1$ points. More precisely, we set $\Delta t = 1/365$ which is realistic for most applications.

1.2.2 The Monte-Carlo simulations

Let us show how to price a digital down-and-out barrier option naively, with lower barrier B. Once we know how to simulate our asset paths, we can sample a number $N >> 1$ of them. Each path will then be used to compute a payoff: averaging them will give us our first price estimation of the barrier contract. More specifically, if $\hat{S_i^{\Delta t}}$ refers to the i-th simulated path simulated as described in 1.2.1, then we are able to compute the non-discounted estimator

$$\mathbb{E}(1_{\nexists s\in[t,T] \text{ s.t } S(s)<B}) \approx \frac{1}{N}\sum_{i=1}^{N} 1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_i^{\Delta t}(s)<B} \tag{1.9}$$

It should be noted in the latter equation that in the right-hand term "$1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_i^{\Delta t}(s)<B}$" can be replaced by "$1_{\nexists j\in\{0,T\} \text{ s.t } \hat{S}_i^{\Delta t}(j\Delta t)<B}$" since our Euler-Maruyama scheme is piece-wise constant as described in 1.2.1. If we want to price a down-and-out call of strike K naively, then we adapt equation (1.9):

$$\mathbb{E}((S_T-K)^+ 1_{\nexists s\in[t,T] \text{ s.t } S(s)<B}) \approx \frac{1}{N}\sum_{i=1}^{N}(\hat{S}_i^{\Delta t}(T)-K)^+ 1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_i^{\Delta t}(s)<B} \tag{1.10}$$

and the same remark holds.
The choice of N in (1.9) and (1.10) is directly related to the convergence properties of the Monte-Carlo method. It is well known (see for example [16]) that this method accuracy increases with the square root of the number of path: to double the accuracy one needs to quadruple the number of paths. In other words, if $\hat{\sigma_N}$ designs the empirical standard deviation of a sample of size N, one can prove this estimator is unbiased which leads to the standard approximated 95% (for example) confidence interval for our Monte-Carlo estimation of the variable X

$$[\hat{X_N} - 1.96\frac{\hat{\sigma_N}}{\sqrt{N}}, \hat{X_N} + 1.96\frac{\hat{\sigma_N}}{\sqrt{N}}] \tag{1.11}$$

Here $\hat{X_N}$ denotes the empirical mean of the sample: $\mathbb{E}(\hat{X_N}) = \mathbb{E}(X)$. Simulation-wise, the computational time required to check whether a path crossed the level B is negligible compared to the time required to sample the N paths: computing the payoff in equation (1.9) and (1.10) is very fast.

1.2.3 First results with naive pricing

To see whether the pricing methodology described in 1.2.1 and 1.2.2 yields prices close to the benchmark, we will price contracts under the Black-Scholes model. We start with $r = 0.05$, $S(t) = 100$, $\sigma = 0.3$, $B = 80$, $T-t$=52 weeks and $K = 100$. The equation (1.6) gives a benchmark price of 13.24. We set $N = 100000$ which is standard in most applications and we plot an histogram of 500 simulated prices:

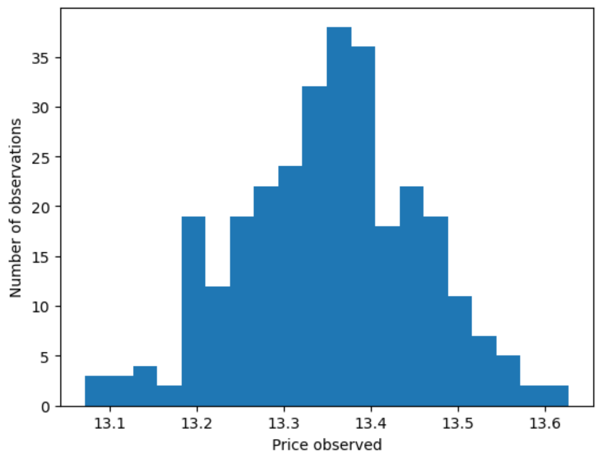


Figure 1.1: Histogram of prices obtained with naive method, mean price of 13.35

As shown in the latter figure, there is a bias. This bias cannot possibly come from the Monte-Carlo method since the empirical mean is an unbiased estimator of the price, which means it is either coming from the Euler-Maruyama scheme or the way we evaluate our payoff. Under the specifications of 1.2.1, the strong convergence theorem guarantees the expected maximal error between our simulated paths and true paths of the asset is of order $O(\sqrt{\Delta t}) \approx O(0.05)$. Moreover, the weak convergence theorem guarantees the expected error between our final value for the simulated paths and true paths to be of order $O(\Delta t) \approx O(0.003)$. Therefore the weak error should not play a role here while the strong error would only account for the very simulated paths that do/do not break the barrier B while the true path do not/do break it. We can hardly imagine the proportion of those paths to be sufficient enough to cause this kind of massive "error". Moreover, if we price the same contract by just setting B=95, we get a benchmark price of 5.50 and the below histogram of simulated prices:

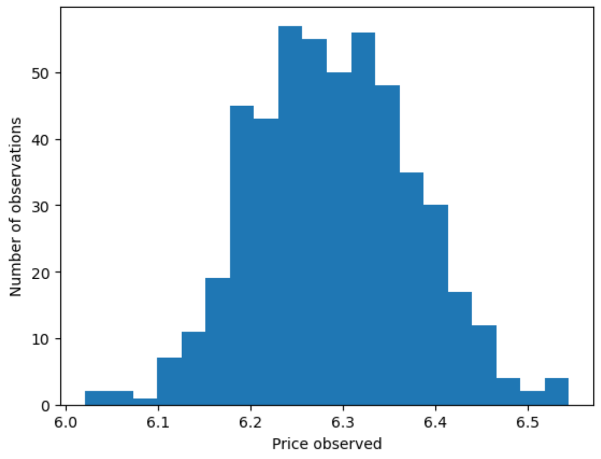


Figure 1.2: Histogram of prices obtained with naive method, mean price of 6.31

The error is way bigger and cannot be due purely to the Euler scheme. More generally, the closer the barrier is to the initial spot, the wider the bias. This bias is also not specific to a down-and-out call: pricing a digital down-and-out in the same fashion results in the same price gap. We keep the same characteristics and get a benchmark price for a digital down-and-out of 0.132. Below is the histogram of simulated prices for the same contract:

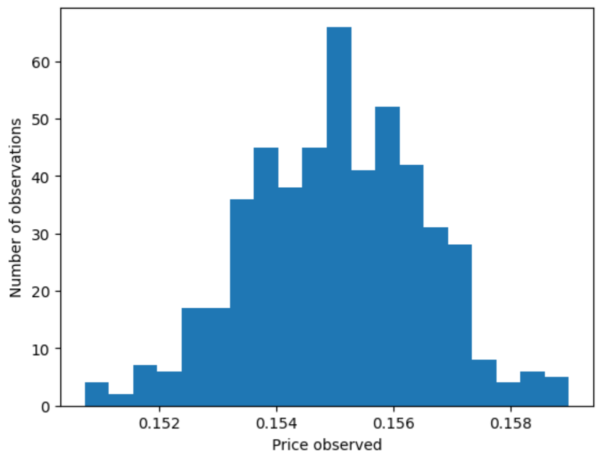


Figure 1.3: Histogram of prices obtained with naive method, mean price of 0.156

Actually, the digital down-and-out call gives us more insight on what is happening: this contract price should be exactly equal to the probability of the asset not crossing the barrier during the lifetime of the option.

1.2.4 Analysis of the naive method

Getting a simulated higher price for both contracts means that we are overestimating the probability of the asset not getting knocked-out. The answer lies in the way we compute the payoff: by only monitoring the asset level relative to the barrier level for every sampled point for each of the simulated paths, we totally discard the asset's behavior between those points. This would be acceptable for a discrete barrier but falls off in the continuous case. The intuitive way to overcome this bias would be to increase the number of sampling points in the simulation of (1.1) and use this to compute (1.9) and (1.10) again. However this is a bad approach for numerous reasons: by doing so we trade the computational efficiency for the accuracy when we would like a fast and accurate way to price those kind of contracts and most importantly this bias is not easy to remove entirely. The time step we chose in 1.2.1 is already small enough for practical applications and sampling (a lot) more points for each path seems like a waste. What if we cannot even set the time step in the Monte-Carlo simulation because the pricing engine is separate from the path generator engine? Moreover, this bias does not disappear fast when we increase the number of time steps: we refer to [6] (Theorem 2.3) for a proof of this result. We state a key point from this article:

Theorem 3. *For a digital down-and-out or a down-and-out call, if P_{bench} designs the true value of the contract and $P_{MC}^{\Delta t}$ the naive value obtained following the procedure outlined in 1.2.1 and 1.2.2, then we have the following result*

$$P_{bench} - P_{MC}^{\Delta t} = O(\sqrt{\Delta t}) \tag{1.12}$$

We need to think of other ways to overcome this bias. Furthermore, to speed up calculations, practitioners often want to do the inverse and reduce the number of monitoring points: given a collection of paths, we aim to compute the price of the contract using a time step bigger than the one used to simulate the paths. For example we can think of a weekly time step, a fortnightly time step or even a mix of different time step for different periods of the contract. One can only expect the bias to grow with the time step using the naive Monte-Carlo method hence the importance of new techniques to address this problem. We want to emphasize that the latter point is independent of the simulated paths: we still simulate the paths with daily time steps following 1.2.1 but we rather aim to reduce the computational expenses to get equations (1.9) and (1.10) because we do not want to introduce undesired statistical bias with the Euler-Maruyama scheme (i.e. we want to keep both strong and weak errors as low as possible). The next section deals with techniques that address this challenge.

1.3 Removing the bias in the single asset case

This section deal with two techniques that address the problems outlined in 1.2.4. We start with the Brownian bridge technique and then deal with the family of barrier shifting techniques. For each of these techniques, we start with the simpler Black-Scholes case then show how to adapt them to more complex/ diversified cases. Numerical examples are provided to illustrate the strengths/ weaknesses of those methods.

1.3.1 The Brownian bridge technique

All the results of this section are taken from [5] (pages 3-11). The main idea of the Brownian bridge technique is the following: we can approximate the probability that the asset breaches its barrier between two time steps and use this probability in the Monte-Carlo payoff. Suppose we sit at time t_i and we know the value of $\hat{S}^{\Delta t}$ at time t_i and t_{i+1}. Following 1.2.1, we got that

$$\hat{S}^{\Delta t}(t_{i+1}) = \hat{S}^{\Delta t}(t_i)(1 + r(t_i, \hat{S}^{\Delta t}(t_i))\Delta t + \sigma(t_i, \hat{S}^{\Delta t}(t_i))\sqrt{\Delta t}Z) \tag{1.13}$$

where $Z \sim N(0,1)$ is a standard normal variable. Given that Δt is small enough, this is under assumption 1 a good approximation of the true behavior of the asset. To compute the probability of exit of $\hat{S}^{\Delta t}$ between t_i and t_{i+1} we could say it follows a Brownian motion with drift equal to $r(t_i, \hat{S}^{\Delta t}(t_i))$ and volatility equal to $\sigma(t_i, \hat{S}^{\Delta t})$ then apply the standard results (as outlined in [5] p.5); for a Brownian motion with drift μ, volatility σ, whose value is x at time t and y at time T, the probability of it breaching the constant level B between t and T is given by:

$$p(x, y, T-t, \sigma, B) = \exp(-2\frac{(x-B)(y-B)}{\sigma^2(T-t)}) \tag{1.14}$$

Note that in (1.14) we assume x and y are above B, otherwise this probability is equal to 1. Also note that this probability does not depend on μ which may seem counter-intuitive but is normal since y already depends on μ. We can then replace x and y by the values of $\hat{S}^{\Delta t}$ at time t_i and t_{i+1} and the volatility at time t_i to get an estimation of the asset breaching its barrier in the time interval. However, this approximation is not that good in the case of a spot-dependent volatility because the volatility is not constant on the time interval. Using Ito's lemma classically yields that the log-price follows a Brownian motion (not geometric) on $[t_i, t_{i+1}]$ which yields a better approximation of the probability of exit. We will therefore work with the probability:

$$p(x, y, T-t, \sigma, B) = \exp(-2\frac{\max(\log(x) - \log(B), 0)\max(\log(y) - \log(B), 0))}{\sigma^2 \Delta t}) \tag{1.15}$$

where x and y should be replaced by the values of $\hat{S}^{\Delta t}$ at time t_i and t_{i+1}. Therefore, for a sampled path $\hat{S}_j^{\Delta t}$ with S following (1.1) the probability of it not breaching its barrier between t_i and t_{i+1} is given by $1 - p(\hat{S}_j^{\Delta t}(t_i), \hat{S}_j^{\Delta t}(t_{i+1}), \Delta t, \sigma(t_i, \hat{S}_j^{\Delta t}(t_i)), B)$ and the probability of it not breaching its barrier between t and T is given by:

$$\nu(\hat{S}_j^{\Delta t}) = \prod_{i=0}^{M-1}(1 - p(\hat{S}_j^{\Delta t}(t_i), \hat{S}_j^{\Delta t}(t_{i+1}), \Delta t, \sigma(t_i, \hat{S}_j^{\Delta t}(t_i)), B)) \tag{1.16}$$

where the t_i's are the monitoring points (between t and T) and M is the number of monitoring points. We can therefore rewrite the payoff for the digital down-and-out taking into account (1.16):

$$\mathbb{E}(1_{\nexists s \in [t,T] \text{ s.t } S(s)<B}) \approx \frac{1}{N}\sum_{j=1}^{N} 1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_j^{\Delta t}(s)<B}\,\nu(\hat{S}_j^{\Delta t}) \tag{1.17}$$

and for the down-and-out call:

$$\mathbb{E}(1_{\nexists s\in[t,T] \text{ s.t } S(s)<B}) \approx \frac{1}{N}\sum_{j=1}^{N}(\hat{S}_j^{\Delta t}(T)-K)^+ 1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_j^{\Delta t}(s)<B}\nu(\hat{S}_j^{\Delta t}) \tag{1.18}$$

This is still an approximation because each of the probabilities in (1.16) assumes the volatility stays constant equal to the left point for each time interval, but we expect it to perform way better overall than the naive pricing method. In particular, in the case where the volatility term in (1.1) is constant, then (1.17) and (1.18) should be unbiased.
To show this, we price the same contracts than in 1.2.4. For a down-and-out call with r=0.05, $S(t)=100$, $\sigma=0.3$, B=80, $T-t$=52 weeks and K=100 we get using (1.18) the following histogram of prices where every price has been obtained using N= 100000 simulations, and the monitoring time step is first set to one day (i.e we check every day if the barrier had been crossed and every probability in (1.16) is computed on a time interval of 1 day):

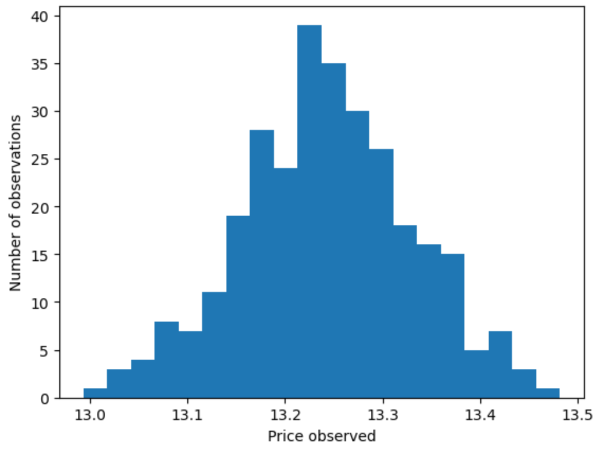


Figure 1.4: Histogram of prices obtained with the Brownian bridge method, mean price of 13.23

The empirical mean of the simulated prices is equal to the benchmark price: the Brownian bridge technique successfully removed the bias.
We also want to make sure this technique works fine when the barrier is close to the initial spot, so we keep the same contract characteristics and only set B=95 (the bias observed with the naive method was bigger). We get the following histogram of prices:

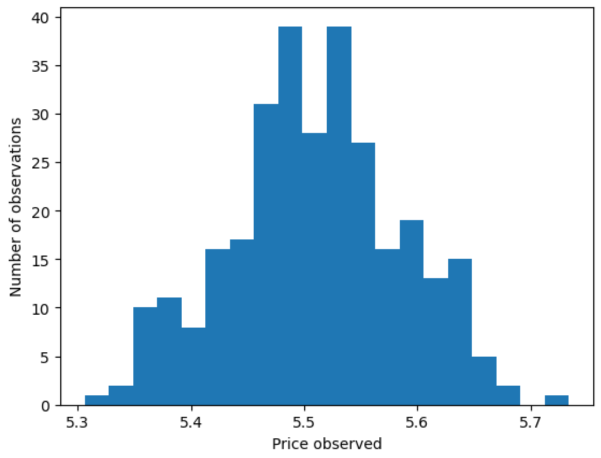


Figure 1.5: Histogram of prices obtained with the Brownian bridge method, mean price of 5.51

Once again there is no bias. We price the exact same contract but we now take monitoring time steps equal to one week to show that the estimator (1.18) is still unbiased as outlined above. We get the following histogram:

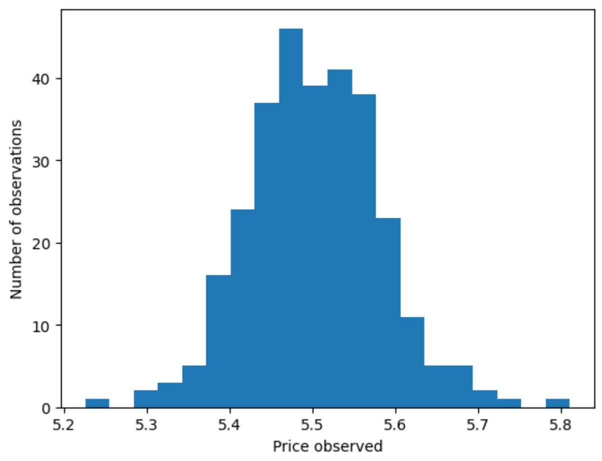


Figure 1.6: Histogram of prices obtained with Brownian Bridge method, mean price of 5.51

The advantage of taking less time step in the computation of (1.18) is that it decreases the computational time required to compute the probabilities and check whether the asset breached its barrier. Taking a weekly time step instead of a daily one cuts this computational time by seven! This can be used in real-word applications when the need for speed is often crucial for traders, because the left-point volatility approximation is still a very good approximation in most cases and the volatility mean is not expected to change by a lot on a given time interval. However this is the responsibility of the model designer to make sure additional checks are implemented to prevent ill cases to have an influence on the final price (e.g. by using adaptive time steps when the volatility regime is modified or when jumps occur in the simulation). To finish with, we price a digital down-and-out with the same characteristics as in 1.3.1 using (1.17) (B=95 and a monitoring time step of seven days). We get the following histogram of prices (which we recall can be seen as the empirical probability the asset does not breach its barrier between t and T):

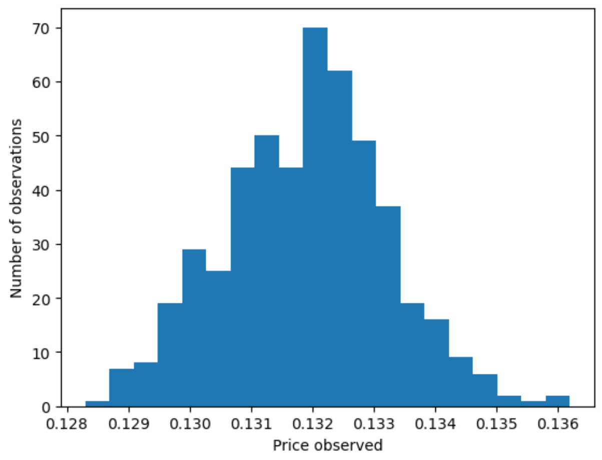


Figure 1.7: Histogram of prices obtained with Brownian Bridge method, mean price of 0.132

Maybe more than for the down-and-out call case, the latter shows that the exit probability estimation is spot on (the benchmark price being 0.132).

1.3.2 The barrier shifting techniques

All the results of this section are taken from [5] (pages 11-32). Unlike the Brownian bridge technique, the family of barrier shifting techniques do not modify the way we evaluate the payoff in (1.9) and (1.10). What is done instead is modifying the barrier level to offset the bias: we aim to shift the barrier level just by the right amount so that we offset the bias induced by only discretely monitoring the barrier. This family of techniques in fact aims to connect the price of a continuously monitored barrier contract and the price of a discretely monitored barrier contract (which is straightforward to simulate).

BAST

Originally developed in [8], the BAST (BArrier Shifting Technique) shifts the barrier by an amount that at least partly depends of the asset level, barrier level, volatility level and time step. We recall the main theorem from [8] (Theorem 1.1 p.327):

Theorem 4. *Let $V_M(B)$ be the price of a discretely monitored (M monitoring points) knock-in or knock-out barrier option (digital or not) with maturity T and volatility σ. Let $V(B)$ be the price of the continuously monitored option with the same characteristics. Then*

$$V_M(B) = V(Be^{\pm\beta\sigma\sqrt{\frac{T}{M}}}) + o(\frac{1}{\sqrt{M}}) \tag{1.19}$$

where - applies if the contract is of the down-and-out type and + applies otherwise. $\beta \approx 0.5826$ is here kept constant.

This is already an improvement compared to the crude method since the left-hand side term in (1.19) can be computed exactly and the total error is now in o rather than in O. This theorem implicitly assumes that the volatility is constant through the life of the option but is still applicable when the volatility is allowed to be a more general process as in (1.1) by using different shifting amounts for different periods. More specifically, if $\sigma = \sigma(t, S)$, then after simulating the paths as described in 1.2.1 we shift the barrier differently for every time interval for every path. Obviously there is no need to do so within the Black-Scholes framework. To show how this method perform, we price the same options as before. Starting with the DOC with barrier level 80, we get the following histogram of prices:

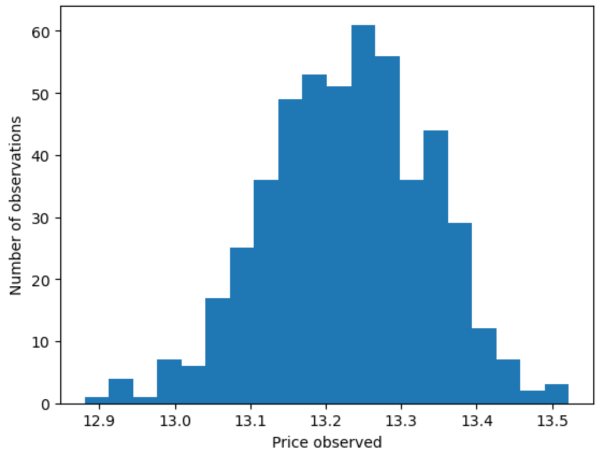


Figure 1.8: Histogram of prices obtained with BAST, mean price 13.25

And indeed we verify that the empirical mean corresponds to the benchmark price (13.24). Now we price the same option setting B=95 and we get:

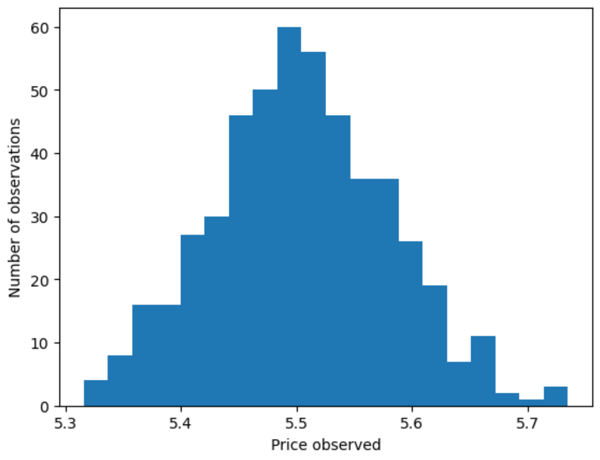


Figure 1.9: Histogram of prices obtained with BAST, mean price of 5.51

Also note that (1.19) holds for digital barrier options as well, as pointed out in [8] (p.343). We price the usual digital down-and-out option and get the following histogram:

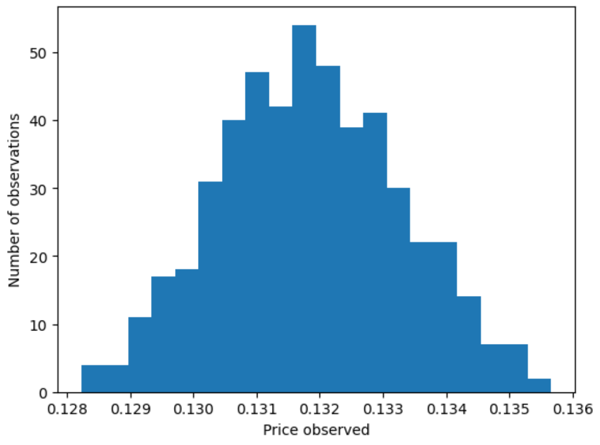


Figure 1.10: Histogram of prices obtained with BAST, mean price of 0.132

However note that this technique, at least following theorem 4, is not perfect. To see this, we price a down-and-out call with strike 100 and barrier level 99. This is quite an extreme case but we would expect a good technique to perform well regardless. The benchmark price is 1.23 (all other parameters are kept constant), and we get the histogram of price:

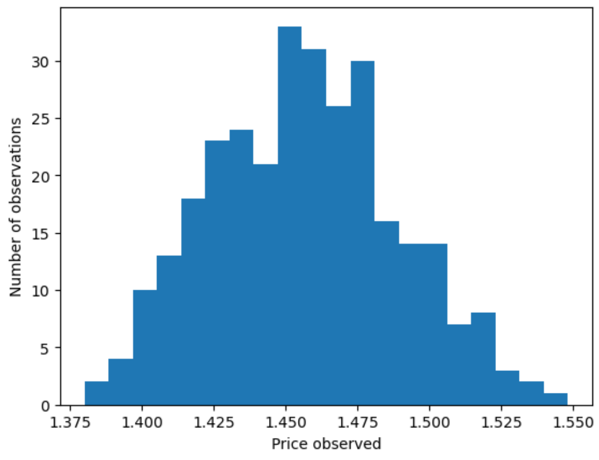


Figure 1.11: Histogram of prices obtained with BAST, mean price of 1.46

This time, the empirical mean is not spot on. This partly has to do with the expected overshoot as outlined in [5] (pages 14-16). This quantity actually depends on the relative distance between the logarithmic value of the initial spot and the logarithmic value of the barrier $u_M = \frac{log(S_t)-log(B)}{\sigma\sqrt{\frac{T}{M}}}$ and converges very fast to $\beta = 0.5826$ when u_M increases. Here σ is the volatility of the asset, T the maturity of the contract and M the number of monitoring points of the barrier. However in [5], Gobet proposes to refine this constant β to take the expected overshoot into account. There is no closed-form formula for this expected overshoot but [5] (p. 23) showed that it could be very well approximated by the following functional:

$$\hat{y}(u_M) \approx 0.5826 + 0.1245\text{exp} \times (-2.7u_M^{1.2}) \tag{1.20}$$

with less than 1% error. This allows us to refine theorem 4 and to move on to ABAST.

ABAST

Following what we just outlined and following [5] (p. 21 and remark of p.23), we can now state a refined version of the theorem 4:

Theorem 5. *Let $V_M(B)$ be the price of a discretely monitored (M+1 time steps) knock-in or knock-out barrier option (digital or not) with maturity T and volatility σ. Let V(B) be the price of the continuously monitored option with the same characteristics. Then*

$$V_M(B) = V(Be^{\pm\hat{y}(u_M)\sigma\sqrt{\frac{T}{M}}}) + o(\frac{1}{\sqrt{M}}) \tag{1.21}$$

where - applies if the contract is of the down-and-out type and + applies otherwise. Here u_M stands for $\frac{log(S_t)-log(B)}{\sigma\sqrt{\frac{T}{M}}}$.

We can now try to price the contract that gave us trouble earlier ($S(t) = 100$, $T - t = 1$ year, $\sigma = 0.3$, $r = 0.05$, $K = 100$ and $B = 99$). We get the following histogram of price:

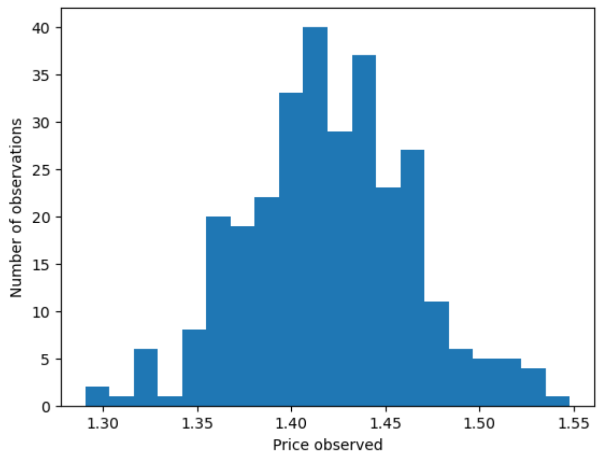


Figure 1.12: Histogram of prices obtained with ABAST, mean price of 1.41

As we can see, there is a slight improvement towards the true value but this is still far from perfect. Another improvement we can add is to move the barrier differently for each time step for every path, so that the barrier shifting is tailored for every time interval. For the monitoring period, we stay with $\Delta t = \frac{1}{365}$ and we get a mean price of 1.39 (0.02).
To understand while we are not able to fit the benchmark price perfectly, we refer to the original derivation of the BAST theorem in the well-known article from [8] (Appendix A) . The error comes from the term $o(\frac{1}{\sqrt{m}})$ and it is true it behaves well when we do not deal with ill cases; however this error depends also of the term u_M and it can quickly become a problem.

Here we should point out that quite surprisingly we found the barrier shifting techniques to behave rather poorly on a wide range of examples. For the down-and-out call option, there is a bias that becomes predominant whenever u_M shrinks. We reproduced the numerical examples given in [8] (Section 2, numerical results) that seem to show those techniques are almost ideal for most application. However, and this is something we highlight further, all of the numerical examples given there are with a strike set to K=100 i.e. for at-the-money contracts. Consider a path $\hat{S}^{\Delta t}$ and suppose we monitor it with a period $\Delta t'$ to evaluate its payoff for a down-and-out call option. The final Monte-Carlo payoff is simply the average of all the payoffs corresponding to the paths 1,...,N. If we shift the barrier at the beginning or for every time interval corresponding to the monitoring periods, the path's payoff given by $(\hat{S}^{\Delta t}(T) - K)^{+} 1_{\nexists s \in [t,T] \text{ s.t the path breaches its barrier}}$ will only differ when using crude Monte-Carlo or the barrier shifting techniques in the following case: the path breaches its shifted barrier AND does not breach its barrier AND does not end up below K. In other words, the proportion of the paths making a difference in the final payoff in the barrier shifting case is a decreasing function of K. For a given barrier level B, the impact of the term $o(\frac{1}{\sqrt{M}})$ will therefore be reduced when K increases. To show this, we price a digital down-and-out option with B=99. The benchmark price (i.e. the discounted probability that a path gets knocked-out) is equal to 0.026. We use the ABAST method to price this contract and re-asjust the barrier or every interval of every path. We get a mean price of 0.031(0.002) which means we overestimate this probability by almost 20% ! Clearly this range of method becomes obsolete when u_M becomes small: the ABAST adjustment is not enough to fit the true probability of the continuous paths getting knocked-out.
Finally, we would like to assess whether the estimates we obtain by using the barrier shifting techniques remain accurate when we decrease the monitoring frequency. We price a down-and-out call with K=10, B=95 and the usual other characteristics by taking fortnightly time steps rather than daily ones, i.e $\Delta t' = \frac{14}{365}$. The benchmark price is 17.33 and we get a simulated price of 18.23 (0.07) using ABAST. This in particular shows that one should be careful whenever there is a need to speed up computations while using barrier shifting techniques: as soon as u_M becomes relatively small the bias becomes overwhelming. We plot below

for the same contract the evolution of the simulated price with the time step:

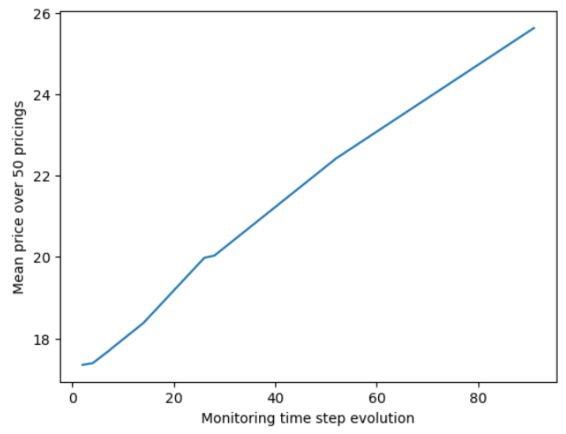


Figure 1.13: Evolution of the contract price using ABAST as a function of the monitoring time step. True price of 17.33

There is clearly an overestimation of the price that does not behave in $o(\frac{1}{\sqrt{M}})$ at all. Note that if the barrier level and strike level are both quite far from the initial spot price, then this method works as intended, even for large monitoring time steps. For example, if $K = 100$ and $B = 80$ the same experiment yields the following plot:

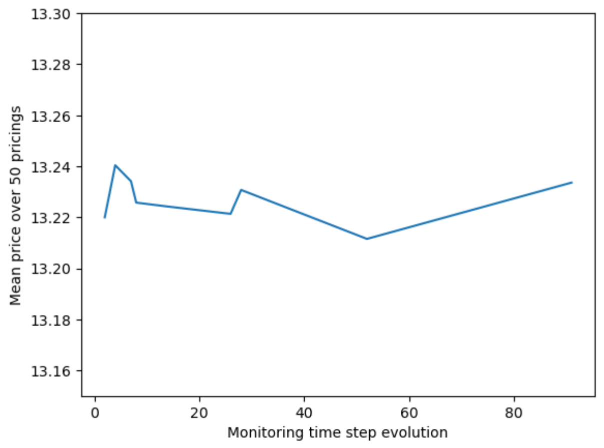


Figure 1.14: Price evolution with the time step

To conclude, the barrier shifting techniques are decent techniques that will yield a better estimate than the crude Monte-Carlo one, but are far from perfect especially when one wants to reduce the computational time by taking bigger monitoring time steps or when one wants to price contracts that are tricky in the sense that the initial spot is close to the barrier level and the contract is relatively far out of the money.

1.4 The Multi-Level Monte-Carlo estimator

In the last section, we have seen how to remove the bias induced by the discrete monitoring of the barrier. Most of the time, the gross of the computational time required to price a barrier contract lies in sampling the paths necessary to the Monte-Carlo estimate. This is done using 1.2.1 and taking a small time step for the simulation is mandatory since we want to keep the error due to the Euler-Maruyama scheme as low as possible i.e. we want the simulated paths to be as close as possible to the true paths. Of course, the more points are used for each path the longer the whole pricing procedure will take. We can approximate the global error of the whole pricing procedure by writing, as pointed out in [9] (equation 7 p.5):

$$a_M - E(X_T) = a_M - E(X_N) + E(X_N - X_T) \tag{1.22}$$

where X_T is the "true" random variable whose expectancy we are trying to compute, X_N is a random variable obtained using Euler-Maruyama that approximates X_T and a_M designs the estimation of $E(X_N)$ obtained by using Monte-Carlo with M paths. The left-hand side of (1.22) is the total error of our procedure and the right-hand side shows this error is due to the Euler-Murayama discretization and the Monte-Carlo procedure. As pointed out before, the Monte-Carlo error tends to decrease when M increases asymptotically in $O(\frac{1}{\sqrt{M}})$ and the other term decreases with the step size we use in Euler-Maruyama at a rate that is approximately equal to $O(\Delta t)$ (even though to price barrier contracts this is not exactly true since strong convergence properties of the scheme theoretically play a role in the final estimate, but we will come to this later). We can therefore state that for a non path-dependent option, if we want to achieve a level of accuracy ϵ, then we should make sure both the approximation scheme error due to the numerical scheme and the error due to Monte-Carlo scale accordingly. Therefore, M should scale like ϵ^{-2} and Δt like ϵ^{-1}. Since the computational cost per path scales like $\frac{1}{\Delta t}$ it means that to reach accuracy ϵ the total cost of the pricing algorithm behaves in $O(\epsilon^{-3})$.
To reduce this cost, we can therefore use a method with better weak convergence order, so that we need to sample less points for every path to achieve the same order of error. However such methods often require additional constraints on the coefficients of (1.1) than those induced by Assumption 1 and the Euler-Maruyama scheme. The Multi-level Monte-Carlo (MLMC in short) method, introduced in the paper [10] takes advantages of both weak and strong convergence properties of the numerical scheme to reduce the overall cost to achieve accuracy ϵ. For a non-path dependent option sampled using the Euler-Maruyama scheme Giles shows in [10] (pages 609-610) that the MLMC framework achieves an overall cost that scales like $O(\epsilon^{-2}(\log(\epsilon)^2))$.

1.4.1 Construction of the MLMC estimator

Let f be a utility function. Suppose we want to estimate the quantity $E(f(X_T))$ using Monte-Carlo and that we do not possess a closed-form formula for $w \longrightarrow X_T(w)$. We therefore have to simulate the variable X_T for a set of possible w to then evaluate the empirical mean using Monte-Carlo. Let $M > 1$ whose precise value does not matter for the asymptotic analysis and let $\Delta t_l = \frac{T}{M^l}$ be the time step at level l, where $l \in \{1, ..., L\}$ with L an integer. Let $\hat{P}_l$ be the random variable that approximates $f(X_T)$ using the time step Δt_l. The MLMC takes advantage of the following equation:

$$\mathbb{E}(\hat{P}_L) = \mathbb{E}(\hat{P}_0) + \sum_{l=1}^{L} \mathbb{E}(\hat{P}_l - \hat{P}_{l-1}) \tag{1.23}$$

For every level l, we estimate $\mathbb{E}(\hat{P}_L)$ using a Monte-Carlo estimator so that the final estimator reads, if we use N_l paths for the level l:

$$\mathbb{E}(\hat{P}_L) \approx \frac{1}{N_0} \sum_{i=1}^{N_0} \hat{P}_0(w_{0,i}) + \sum_{l=1}^{L} \frac{1}{N_l} \sum_{i=1}^{N_l} (\hat{P}_l(w_{l,i}) - \hat{P}_{l-1}(w'_{l,i})) \tag{1.24}$$

It should be noted in equation (1.24) that for every level l, we compare the estimator $\hat{P}_l$ and $\hat{P}_{l-1}$ on the "same" realizations $w_{l,i}$ and $w'_{l,i}$ every time which is a vital feature of the MLMC method. By same realization we mean that the same simulated Brownian motion is used at level l for both $\hat{P}_l$ and $\hat{P}_{l-1}$ for $i = \{1, ...N_l\}$. In equations, let $l > 0$ and suppose we sample a Brownian Motion using normal realizations at resolution $l - 1$:

$$W_{l-1}^i(j\Delta t_{l-1}) = \sum_{k=0}^{2j} Z_{k,i}\sqrt{\Delta t_l} \tag{1.25}$$

where $Z_k \sim N(0,1)$ are independent random variables and $j \in \{0,..,T^{-1}M^{l-1}\}$. We then let the corresponding discretized Brownian Motion at level l be:

$$W_l^i(j\Delta t_l) = \sum_{k=0}^{j} Z_{k,i}\sqrt{\Delta t_l} \tag{1.26}$$

with $j \in \{0,..,T^{-1}M^l\}$. The "fine" paths $w_{l,i}$ are then built using increments of the Brownian motion (1.26) and the "coarse" paths $w'_{l,i}$ using increments of (1.25) and 1.2.1. The simulated paths are very close, although of different resolutions, and it is how they differ that matters when using this method.

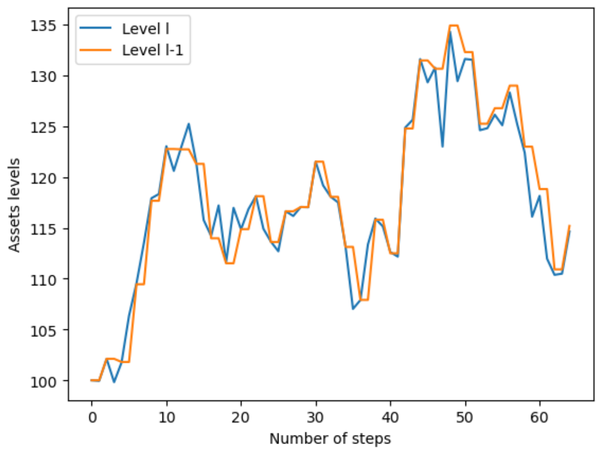


Figure 1.15: Path at a level of resolution l vs resolution l-1

1.4.2 Properties of the MLMC estimator

We now state the general theorem for the MLMC method from [10] (theorem 3.1 p.609) which will allow us to choose the number of levels L we use and the number of paths per level N_l for each of those levels.

Theorem 6. *Let P be the random variable whose mean we want to estimate and let Y_l be independent estimators at level $\{0,...,L\}$ (not necessarily the ones build in the last subsection). If there exists positive constants α, β, γ, c_1,c_2 and c_3 such that $\alpha \geqslant \frac{1}{2}min(\beta,\gamma)$ and*

1. $|\mathbb{E}(P_l - P)| \leqslant c_1 2^{-\alpha l}$
2. $\mathbb{E}(Y_0) = \mathbb{E}(P_0)$ *and* $\mathbb{E}(Y_l) = \mathbb{E}(P_l - P_{l-1})$ *for* $l > 0$.
3. $var(Y_l) \leqslant c_2 N_l^{-1} 2^{-\beta l}$
4. $\mathbb{E}(C_l) \leqslant c_3 N_l 2^{\gamma l}$, *where* C_l *is the complexity of* Y_l

then there exists a positive constant c_4 such that for any $\epsilon < e^{-1}$ then there exists values L and N_l for which the MLMC estimator

$$Y = \sum_{i=0}^{L} Y_i$$

has a mean square error with bound

$$\mathbb{E}((Y - \mathbb{E}(P))^2) < \epsilon^2$$

with a computational complexity C with bound

$$\mathbb{E}(C) \leqslant \begin{cases} c_4\epsilon^{-2}, & \beta > \gamma \\ c_4\epsilon^{-2}log(\epsilon)^2, & \beta = \gamma \\ c_4\epsilon^{-2-\frac{(\gamma-\beta)}{\alpha}}, & \beta < \gamma \end{cases} \tag{1.27}$$

One can apply this theorem to prove that using the MLMC estimator and the construction of the estimators $\hat{P}_l$ of the last subsection for a Lipschitz payoff (i.e. European put an call options that are not path-dependent) indeed yields a complexity in $O(\epsilon^{-2}\log(\epsilon)^2)$ (case $\beta = \gamma$). One can also use a more refined numerical scheme such as the Milstein scheme to attain $O(\epsilon^{-2})$ but we do not do it here for the reasons outlined above (see [11] p.18).
Needless to say we are not in the case of a Lipschitz payoff. The problem with barrier options lies in the point 3 of 6: a path can totally get knocked-out at the level l and not at the level l-1 which results in a $O(1)$ difference in payoff and this proportion of path, intuitively low when u_M is low can increase with u_M. In [12] (pages 6-8), it is proven that for any $\delta > 0$, we have

$$\mathbb{E}((P - \hat{P})^2) = O(h^{\frac{1}{2}-\delta}) \tag{1.28}$$

which by using an upper bound of the variance and Minkowski's inequality is enough to prove that the estimator built in subsection 1.4.1 achieves an overall complexity that is bounded by $O(\epsilon^{-2-\frac{\delta-\frac{1}{2}}{\alpha}})$ for any $\delta > 0$ which is still an improvement compared to the standard Monte-Carlo method. Here $\hat{P}$ is the approximation of P using Euler-Maruyama. We will see in the next section how much better this method performs.

1.4.3 Pricing using the MLMC method

This section outlines the results obtained by mixing the MLMC technique and the Brownian Bridge technique. The process is the same when we want to combine the MLMC method with the barrier shifting technique but we focus on the former since it gave the best results in 1.2.4. We do not use the estimators defined in 1.4.1 but rather the estimators Y_l defined in 1.3.1 meaning for each time step Δt_l we use the discounted estimators given by (1.17) and (1.18) and we will use (1.24) to approximate the value of the down-and-out barrier options. We set $M = 2$ and process in the following fashion: we first define a level of accuracy ϵ we want to reach. We set L to be the first integer such that $2^L \geqslant T$. We then run "pilot" runs to compute an estimation of the cost and of the variance of the estimators Y_l for each level $l \in \{0, .., L\}$. In practice we impose a run time limit τ to perform those pilot runs at each level: the first estimated path is a good estimator of the time required to sample a path at resolution l and evaluate its payoff using Y_l. Then, we sample enough paths at level l so that the overall run time of the pilot runs at level l is approximately equal to τ. Since all the levels are independent, we can write:

$$\text{var}(Y_0 + \sum_{i=1}^{L} Y_i) = \sum_{i=0}^{L} \text{var}(Y_i) \tag{1.29}$$

Now using that $Y_l = \frac{1}{N_l}\sum_{i=1}^{N_l}(\hat{P}_l(w_{l,i}) - \hat{P}_{l-1}(w'_{l,i}))$ where P_l corresponds to the estimator defined by (1.17) for digital barrier option and (1.18) for regular ones we get that the variance at level l is simply the empirical one we found through our pilot runs divided by N_l. Since we want to achieve:

$$\text{var}(Y) < \epsilon^2 \tag{1.30}$$

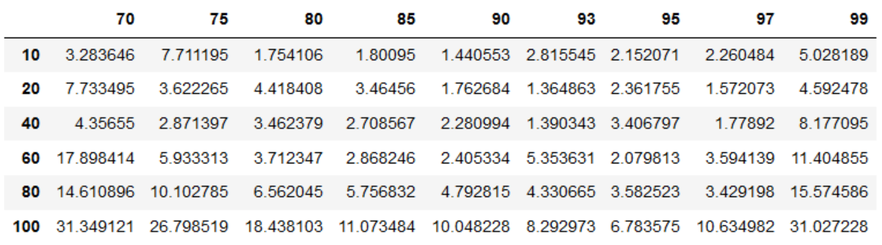
we need to make sure the sum of the variance across each level does not exceed ϵ^2. Since the cost is also given by those pilot runs, we solve a classic optimization problem: the output of this optimization is the necessary number of levels L to use as well with the number of paths N_l to use across every level. If we denote by $\hat{C}_l$ the empirical cost per path at level l (in seconds) then the objective function we want to minimize is:

$$J(N_0, .., N_L) = \sum_{i=0}^{N} C_i N_i \tag{1.31}$$

subject to $\mathrm{var}(Y) < \epsilon^2$ and $N_l \geqslant 10$ for each level (to not get outlier paths one one level). Note that this pre-analysis also allows to compute an empirical β and γ and *at posteriori* an empirical α.

Results without variance reduction

We now present the numerical results corresponding to this method. We do not yet try to reduce the variance and stick to what has been said before. What really matters here is the ratio between the computational time using standard Euler-Maruyama/ Standard Monte-Carlo and the MLMC method for a given level of accuracy ϵ. Below is the table summing up those results for down-and-out options for $\epsilon = 0.15$. We set $S(t) = 100$, $r = 0.05$, $T - t = 52$ weeks and a constant volatility equal to 0.3. We price down-and-out call options and we make the strike K (index) and barrier level B (columns) vary:



	70	75	80	85	90	93	95	97	99
10	3.283646	7.711195	1.754106	1.80095	1.440553	2.815545	2.152071	2.260484	5.028189
20	7.733495	3.622265	4.418408	3.46456	1.762684	1.364863	2.361755	1.572073	4.592478
40	4.35655	2.871397	3.462379	2.708567	2.280994	1.390343	3.406797	1.77892	8.177095
60	17.898414	5.933313	3.712347	2.868246	2.405334	5.353631	2.079813	3.594139	11.404855
80	14.610896	10.102785	6.562045	5.756832	4.792815	4.330665	3.582523	3.429198	15.574586
100	31.349121	26.798519	18.438103	11.073484	10.048228	8.292973	6.783575	10.634982	31.027228

Figure 1.16: Ratio of the computational times: SMC over MLMC methods

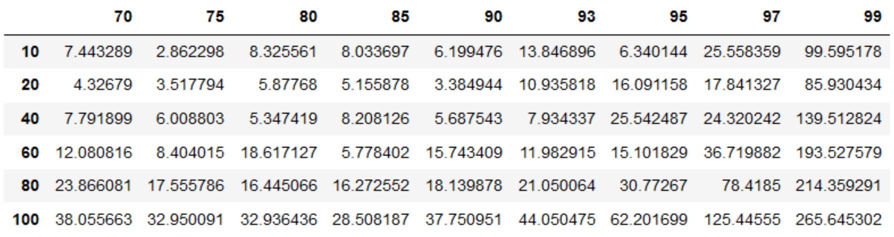
As we can see, the MLMC method outperforms the standard Monte-Carlo procedure. It is also interesting to notice that this ratio differs a lot with the barrier level B and strike K.

Results with variance reduction

Finally, we show that the variance of the estimator Y can be improved by modifying slightly the estimators Y_l as shown in [11] (pages 24-28). Let us sum up the method. In the last paragraph, the estimator Y_l is built using the difference of the estimator defined in (1.17) and (1.10) on the path defined with the Brownian motion sampled as in (1.26) and (1.25). We would like to include the information at level l for the fine estimator in the coarse estimator at the same level. To do so, we refine the coarse Brownian path at level l by sampling a point using a Brownian bridge construction. More specifically, we still sample the coarse Brownian path using (1.25) and build the asset path according to (1.2.1) but for every $j \in \{1, ..., (2^l - 1)T^{-1}\}$ we set the following value for the asset path $i \in \{1, N_l\}$

$$\begin{aligned} \hat{S}_i^{\Delta t_{l-1}}(j\Delta t_l) = \frac{1}{2}\Big(& \hat{S}_i^{\Delta t_{l-1}}((j-1)\Delta t_l) + \hat{S}_i^{\Delta t_{l-1}}((j+1)\Delta t_l) \\ & - \hat{S}_i^{\Delta t_{l-1}}((j-1)\Delta t_l)\sigma((j-1)\Delta t_{l-1}, \hat{S}_i^{\Delta t_{l-1}}((j-1)\Delta t_l)) \\ & \times (W_{l-1}^i((j+1)\Delta t_l) - 2W_l^i((j+1)\Delta t_l) + W_{l-1}^i((j-1)\Delta t_l))\Big) \end{aligned} \tag{1.32}$$

As pointed out in [11], modifying the coarse estimator at every level $l > 1$ does not modify the equality (1.23) and guarantees convergence (provided we satisfy theorem 6). Intuitively this reduces the variance because a path getting knocked out at the fine level l is now almost sure to be knocked out at the coarse level l for the estimator Y_l. However as pointed out in [11] (pages 31-32), finding estimators in such a way for other problems is not guaranteed at all and is still an area of research especially in the case of multi-dimensional SDE's. We now conduct the same experiments as in the last section to find the computational time ratio between the standard Monte-Carlo method and this modified estimator:



	70	75	80	85	90	93	95	97	99
10	7.443289	2.862298	8.325561	8.033697	6.199476	13.846896	6.340144	25.558359	99.595178
20	4.32679	3.517794	5.87768	5.155878	3.384944	10.935818	16.091158	17.841327	85.930434
40	7.791899	6.008803	5.347419	8.208126	5.687543	7.934337	25.542487	24.320242	139.512824
60	12.080816	8.404015	18.617127	5.778402	15.743409	11.982915	15.101829	36.719882	193.527579
80	23.866081	17.555786	16.445066	16.272552	18.139878	21.050064	30.77267	78.4185	214.359291
100	38.055663	32.950091	32.936436	28.508187	37.750951	44.050475	62.201699	125.44555	265.645302

Figure 1.17: Ratio of the computational times: SMC over MLMC methods

The computational gain is way higher than without variance reduction: modifying the path as outlined above allows the method to perform almost as good as with a Lipschitz payoff.

Chapter 2

The Multi-Asset case

The last chapter dealt with one asset only. Naturally a lot of contracts involve more than one underlying asset. We present a few of them in what follows. We then show how to adapt 1.2.1 in this setting to take into account correlation between the assets and present two techniques to price those contracts. Finally we show how to use the MLMC method in this case.

2.1 Contextualization

A lot of the barrier contracts that are traded actually involve more than one asset. We consider n assets $S^i, i \in \{1, n\}$ where every asset follows the standard model:

$$dS_t^i = r_t^i S_t^i dt + \sigma_t^i S_t^i dW_t^{\mathcal{Q},i} \tag{2.1}$$

where the drift and volatility processes for each of the assets are supposed to satisfy Assumption 1. The Brownian motions $W^{\mathcal{Q},i}$ are not necessarily independent and therefore we assume that for $i, j \in [\![1..n]\!]^2$ there exists a correlation function $\rho_{i,j}$ such that for $s \in [t, T]$ we have

$$dW_s^{\mathcal{Q},i} dW_s^{\mathcal{Q},j} = \rho_{i,j}(s) \tag{2.2}$$

We can rewrite the $n + n^2$ equations from (2.2) and (2.1) with the following system:

$$dS_t = r_t dt + \Sigma(t)\sigma_t dW_t^{\mathcal{Q}} \tag{2.3}$$

Here S_t, r and σ should be understood as $n \times 1$ vectors and Σ as the Cholesky decomposition of the $n \times n$ correlation matrix, which is positive-definite with all diagonal entries equal to 1. Also $W^{\mathcal{Q}}$ should be understood as a $n \times 1$ vector of uncorrelated Brownian motions. Note in (2.2) the correlation functions are supposed to be deterministic functions of the time but in practice it could be dependent of more terms/ stochastic; it does not affect the following results. We will consider the following type of contract throughout our numerical study: a down-and-out call on the first asset that pays $(S^1(T) - K)^+$ conditional on none of the asset touching its barrier B^i on $[t, T]$. It is enough to consider this type of payoff since the study is more focused on getting a good approximation of the probability of exit on $[t, T]$ rather than on the payoff. Unfortunately, there is no closed-form formula for the price of such contracts even under the simpler Black-Scholes model but we refer to [7] (annex) for a whole table of such prices for a wide range of contracts.

2.2 Simulation and naive pricing

For the same reasons as the ones outlined in Chapter 1, we do not consider a Milstein simulation of the stochastic system (2.1) since this is often an unrealistic approach that requires to know about partial

derivatives of the σ vector and requires simulation of the Levy areas (see [13] pages 10-11). We instead simply use (1.2.1) for each of the simulated assets which preserves the rate of strong/weak convergence of the total simulation.
The (not discounted) payoff for the down-and-out call contract on the first asset therefore reads in the naive setting:

$$\mathbb{E}((S^1(T)-K)^+1_{\nexists s\in[t,T] \text{ s.t } S^j(s)<B_j \quad \forall j\in\{1,..,n\}} \approx \frac{1}{N}\sum_{i=1}^{N}(\hat{S}_i^{\Delta t,1}(T)-K)^+1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_i^{\Delta t,j}(s)<B_j \quad \forall j\in\{1,..,n\}} \tag{2.4}$$

where $\hat{S}_i^{j,\Delta t}$ designs the simulated i-th path of the j-th asset. The results we outlined in 1.2.1 and 1.2.2 still hold for this procedure. For the simulation of the paths, we still take $\Delta t = \frac{1}{365}$ for all of the assets and use $N = 100000$ paths. The first derived numerical results are for 3 assets with, $S^i(t) = 100$, $T-t = 1$ year, $r^i = 0.05$, $K = 100$, $\sigma_i = 0.4$, $B^i = 80$ and $\rho_{i,j} = 0.5$ if $i \neq j$. As shown in [7] (p.19), the benchmark price is equal to 7.60. We plot an histogram obtained with 500 realizations of the naive procedure with the above parameters:

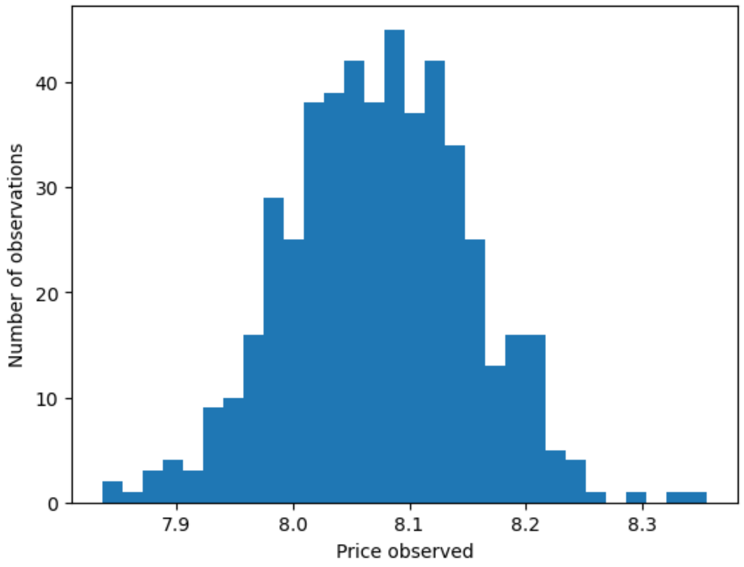


Figure 2.1: Naive price obtained with Monte-Carlo, mean price of 8.10

As we can see, we encounter the very same phenomenon outlined in 1.2.4. The naive procedure underestimates the probability of ones of the asset getting knocked-out and we show that this bias increases when the mean of the quantities $u_N^i = \frac{\log(\frac{S^i(t)}{B_i})}{\sigma_i\sqrt{\Delta t'}}$ (where $\Delta t'$ is the monitoring period that is greater or equal than the simulation period Δt) increases by pricing the same contract with $B^i = 95$: the benchmark price is 0.77 and we get the following price histogram:

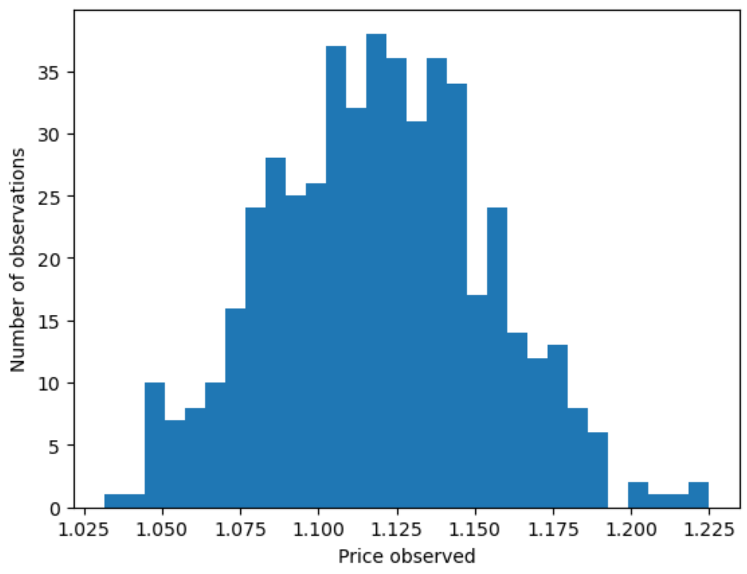


Figure 2.2: Naive price obtained with Monte-Carlo, mean price of 1.12

This bias again increases when K decreases. This happens because roughly speaking the proportion of path breaching their barrier and ending above the barrier vector B increases which means there is a higher margin for error. Again we need techniques that allow us to take into account this probability of exit between every interval of the form $[j\Delta t', (j+1)\Delta t']$. The difficulty that arises in this situation is, unlike for the single asset case, that the correlation structure has to be taken into account. More specifically, if two assets S^i and S^j have a correlation ρ constant on $[k\Delta t', (k+1)\Delta t']$ then the probability of any of them getting knocked-out on this time interval is different than the one under correlation 0 i.e. if the assets were independent. Consider the following notations (we will use those notations again later):

$$\begin{aligned} E_i([k\Delta t',(k+1)\Delta t'], B^i) = \Big\{ &\text{The } i\text{-th asset breaches its barrier } B^i \\ &\text{during } [k\Delta t', (k+1)\Delta t'], \text{ knowing the values of } S^i \text{ at the two endpoints} \Big\} \end{aligned} \tag{2.5}$$

If there is no confusion on the interval of time $[k\Delta t', (k+1)\Delta t']$ and the level of the barrier B^i we will simply write E_i. Then, we write:

$$P^\rho(E_i \cup E_j) \tag{2.6}$$

as the probability of at least one of the assets S^i or S^j getting knocked-out on $[k\Delta t', (k+1)\Delta t']$ if we suppose they have correlation ρ on this interval of time. What we said earlier is that under the assumption $\rho \neq 0$ on $[k\Delta t', (k+1)\Delta t']$ we generally have:

$$P^\rho(E_i \cup E_j) \neq P^0(E_i \cup E_j) \tag{2.7}$$

To see this, one can of course think of the pathological case where the assets would be correlated with $\rho \approx 1$. In this case we simply get:

$$P^\rho(E_i \cup E_j) \approx P(E_i) \neq P^0(E_i \cup E_j) \tag{2.8}$$

in general. Of course this is true for every value of ρ that is not 0. This in particular means that if we think about the Brownian bridge technique outlined in 1.3.1, then we cannot adapt it easily in this situation if the correlation is not zero. If the correlation is zero, we can write

$$P^0(E_i \cup E_j) = P(E_i) + P(E_j) - P(E_i)P(E_j) \tag{2.9}$$

and therefore we can compute analytically the latter equation since every $P(E_i)$ and $P(E_j)$ are given by (1.15) (given there is a geometric Brownian motion part in the volatility terms of S^i and S^j). This generalizes for more than 2 assets because the probability of any of the asset getting knocked-out will ultimately simplify to terms that can be directly computed using (1.15). However, in the general case where the correlation matrix differs from the identity matrix, the Brownian bridge method cannot be used to provide unbiased estimation of the price and we need other ways to compute it. Below we show how evolves $P^\rho(E_i \cup E_j)$ when the initial and terminal values of S^i and S^j are $\begin{pmatrix}100\\100\end{pmatrix}$ on a one day interval, and when the volatilities are constant equal to 0.4 with $r = 0.05$ for both of the assets. We also set $B = \begin{pmatrix}98\\98\end{pmatrix}$:

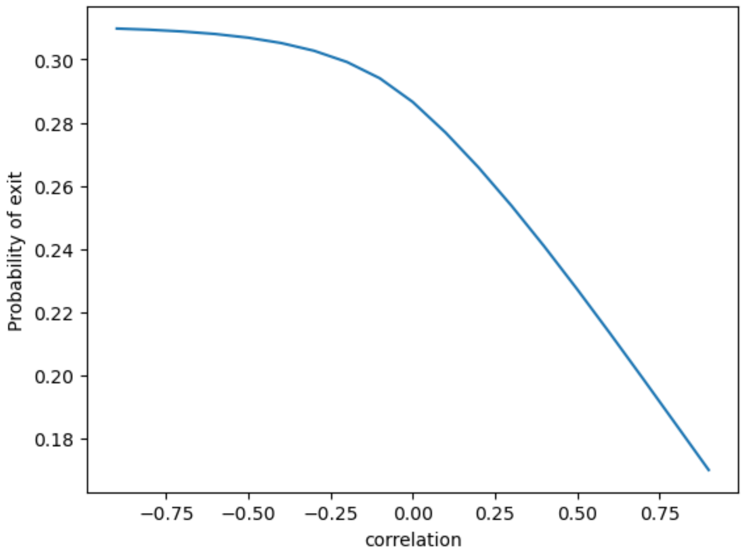


Figure 2.3: Probability of exit with correlation

2.3 Removing the bias in the multi-asset case

2.3.1 The copula method

A way to get the price of those contracts is given in [7]. All the results that follow are from this article. The copula method introduced here does not assume anything more about the simulation procedure or the coefficients in (2.1) than the assumptions we already provided. This method gives up on trying to derive an unbiased estimator of the option price and rather derives two biased estimators that bound the option price and converge towards it. This method is based on the following result from [7] (Theorem p.7) we recall here:

Theorem 7. *Let $A_1, ..., A_n$ be events in a probability space such that $P(A_i) = a_i$, $i \in \{1, ..., n\}$. Then we got the Fréchet's bounds for the probability of the intersection of those events:*

$$\max[0, \sum_{i=0}^{n} a_i - (n-1)] \leqslant P(A_1 \cap ... \cap A_n) \leqslant \min_{i=1,..,n} a_i \tag{2.10}$$

If we think of the events E_i defined above, what we really want is to get those bounds for the probability that none of the paths corresponding to the assets S^j where $j \in \{1, n\}$ gets knocked-out on $[t, T]$ (which would be what to plug in the Brownian bridge method if we knew how to compute it). This probability is given by (for n assets on a time interval corresponding to the monitoring period):

$$P(\overline{\cup_{i=1}^{N} E_i}) = P(\cap_{i=1}^{N} \overline{E_i}) \tag{2.11}$$

Therefore applying 2.10 to (2.11) yields the following equation:

$$\max[0, \sum_{i=0}^{n} P(\overline{E_i}) - (n-1)] \leqslant P(\cap_{i=1}^{N} \overline{E_i}) \leqslant \min_{i=1,..,n} P(\overline{E_i}) \tag{2.12}$$

Now let

$$P_L([k\Delta t', (k+1)\Delta t'], S(k\Delta t'), S((k+1)\Delta t'), B) = \max[0, \sum_{i=0}^{n} P(\overline{E_i([k\Delta t', (k+1)\Delta t'], B_i)}) - (n-1)] \tag{2.13}$$

and

$$P_U([k\Delta t', (k+1)\Delta t'], S(k\Delta t'), S((k+1)\Delta t'), B) = \min_{i=1,..,n} P(\overline{E_i([k\Delta t', (k+1)\Delta t'], B_i)}) \tag{2.14}$$

design the lower and upper bounds for the probability of not getting knocked-out for the process S ($n \times 1$ vector) between time $[k\Delta t', (k+1)\Delta t']$ that corresponds to a monitoring period, with lower barriers B ($n \times 1$ vector). To simplify the notations, we will write $P_L(S, k\Delta t')$ and $P_U(S, k\Delta t')$ to refer to (2.13) and (2.14) when there is no confusion about the other parameters. Now if we let $\{\hat{S}_i^{\Delta t}\}_{i=1,..,N}$ be the simulated sample of (2.1) using Euler-Maruyama and with M monitoring points, we define the lower biased estimator of the price of the option (not discounted) to be

$$E_L = \frac{1}{N} \sum_{i=1}^{N} \left((\hat{S}_i^{\Delta t,1}(T) - K)^+ \cdot 1_{\nexists s \in [t,T] \text{ s.t } \hat{S}_i^{\Delta t,j}(s) < B_j \quad \forall j \in \{1,...,n\}} \right) \times \prod_{k=0}^{M-1} P_L(\hat{S}_i^{\Delta_t}, k\Delta t') \tag{2.15}$$

and the upper biased estimator (not discounted) to be:

$$E_U = \frac{1}{N} \sum_{i=1}^{N} \left((\hat{S}_i^{\Delta t,1}(T) - K)^+ \cdot 1_{\nexists s \in [t,T] \text{ s.t } \hat{S}_i^{\Delta t,j}(s) < B_j \quad \forall j \in \{1,...,n\}} \right) \times \prod_{k=0}^{M-1} P_U(\hat{S}_i^{\Delta_t}, k\Delta t') \tag{2.16}$$

Note that the expressions (2.15) and (2.16) are correct since the simulated samples are piece-wise constants on $[t, T]$. Thanks to theorem 2.10, we know that the true price of the option lies within those bounds. When the monitoring period decreases, we expect the two estimators to converge towards one another which will allow us to estimate the price of the option. In [7], Shevchenko showed on a variety of examples that this convergence was indeed very fast in the sense that even a large monitoring period can yield meaningful results when one uses for example the midpoint estimator given by:

$$E_M = \frac{E_L + E_U}{2} \tag{2.17}$$

Moreover, Shevchenko also shows how to adapt those estimators in the case of other contracts such as when some of the assets have two barriers. Truth is that this method is flexible and even though the convergence of the biased estimators towards the price is slower when ρ gets close to 1, it is still very much sufficient for the realistic cases one can encounter in finance. Another advantage of this method is that we do not actually need to know the correlation between the assets to estimate the price. It means that even if we need to know the correlation matrix to simulate the asset paths, we can do without it in the pricing setting. We now price a call on the first asset conditional on none of 3 assets getting knocked-out with $\Delta t' = \frac{1}{365}$, $S_t^i = 100$, $T - \tau = 1$ year, $r^i = 0.05$, $K = 100$, $\sigma_i = 0.4$, $B_i = 80$ and $\rho_{i,j} = 0.5$ if $i \neq j$. As shown in [7] (p.19), the benchmark price is equal to 7.60. We get the following histogram of price:

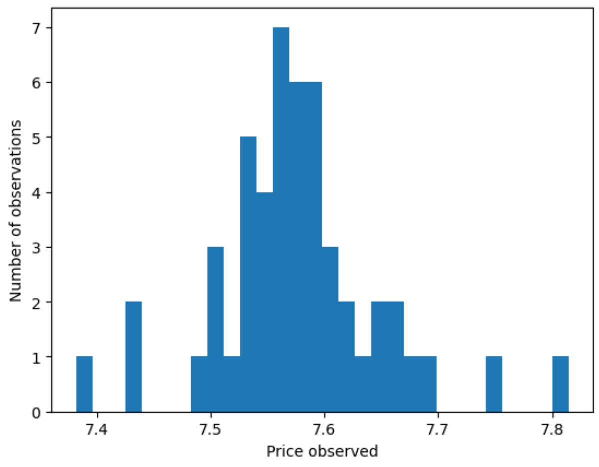


Figure 2.4: Copula price obtained with Monte-Carlo, mean price of 7.59

This time the average price is spot on. We do the same when $B_i = 95$ for each of the asset and get the following histogram of prices (benchmark price of 0.77):

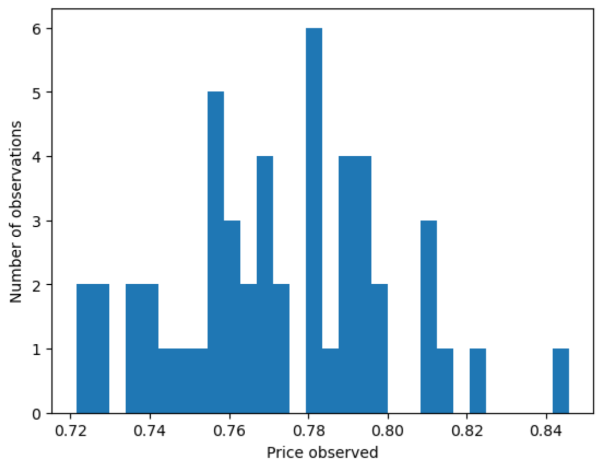


Figure 2.5: Copula price obtained with Monte-Carlo, mean price of 0.78

Again there is no more bias. In fact the convergence stays very good even if we take bigger time steps e.g $\Delta t' = \frac{7}{365}$ for example. We could take even bigger time steps but one has to keep in mind that in real situations we deal with volatility functions that are not constant and taking bigger time steps might introduce a bigger error due to the left-point approximation of the volatility for each time step. Nevertheless we show that even for very pathological cases those estimators stay relatively stable when we increase $\Delta t'$. We set $B_i = 95 \quad \forall i \in [1, 3]$ and we now plot the evolution of the upper and lower copula estimators for this contract as a function of the monitoring time step.

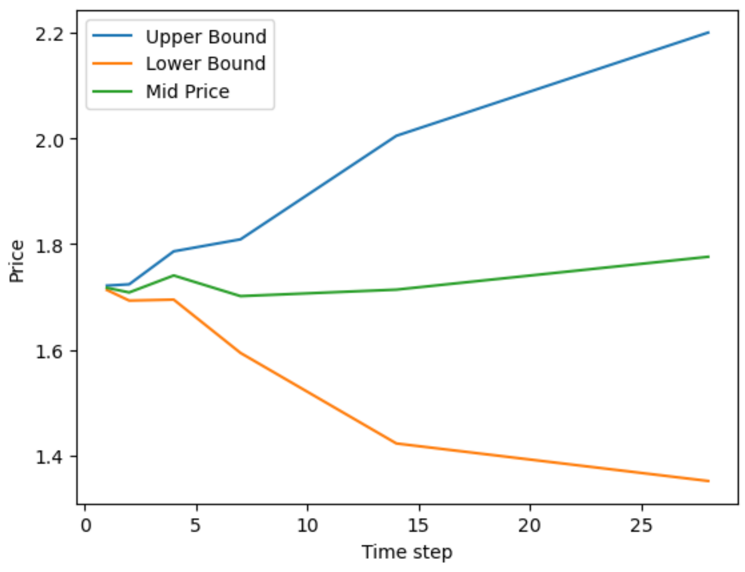


Figure 2.6: Copula prices obtained with Monte-Carlo as a function of the time step

Even for a very ill case the price stays relatively stable with the monitoring time step $\Delta t'$ so we would expect this to hold as well for other contracts. For example, if we take $B^i = 80$ and $K = 100$ then the same experiment shows:

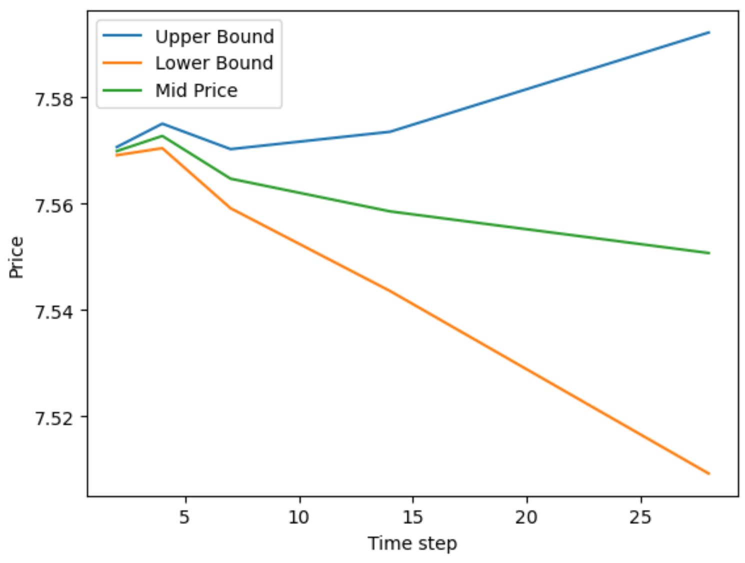
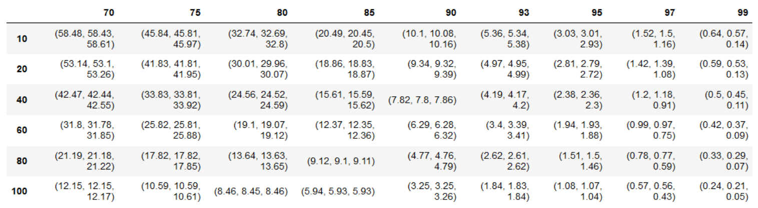


Figure 2.7: Copula prices obtained with Monte-Carlo as a function of the time step

And we can see the lower and upper bounds are way closer as the monitoring time step increases as with the previous case.

2.3.2 The barrier shifting techniques for the multi-asset case

In [5] (p.27), Gobet generalizes the barrier shifting method for more than one asset by stating that "Contrary to the Brownian bridge techniques, we do not need to take into account the correlation between assets to adjust the barriers" and then provides a range of numerical experiments to confirm this theoretical choice. Namely every barrier is shifted as in 1.3.2 independently of if we pick the BAST and ABAST technique and independently of the correlation between the assets (which makes sense since the barrier shifting techniques are more concerned about the normal increments rather than the initial and terminal values for each time interval). We indeed implemented this and got the same results as in [5] (Table 6 p.29). However and once again, the argument given in 1.2.4 seems to hold. Choosing a strike $K = 100$ and a low level for the barriers is not really relevant when we want to assess whether the barrier shifting techniques perform well in the multi-asset setting. For example, let $d = 3$ be the number of assets, $S^i(t) = 100$, $T - \tau = 1$ year, $r^i = 0.05$, $K = 10$, $\sigma_i = 0.3$, $B_i = 95$, $\Delta t' = \frac{7}{365}$ and $\rho_{i,j} = 0.5$. The benchmark price is 3.14 and the BAST yields a mean price of 3.25 while the ABAST yields a mean price of 3.23. Those method fails for this example. Nevertheless they still performs well on contracts that are not too exigent in the sense that either the mean of the quantities $u_{M,i} = \frac{log(S^i(t))-log(B^i)}{\sigma\sqrt{\frac{T}{M}}}$ is quite low and/ or K is large which affects the approximation as discussed in 1.2.4. Again, we set $d = 3$ the number of assets, $S^i_t = 100$, $T - \tau = 1$ year, $r^i = 0.05$, $\sigma_i = 0.3$, $\Delta t' = \frac{7}{365}$ and $\rho_{i,j} = 0.5$. We make K (index) and B^i (columns) vary and we compare the mean price obtained over 50 repetitions using BAST (left), ABAST (middle) and the copula method (right):



	70	75	80	85	90	93	95	97	99
10	(58.48, 58.43, 58.61)	(45.84, 45.81, 45.97)	(32.74, 32.69, 32.8)	(20.49, 20.45, 20.5)	(10.1, 10.08, 10.16)	(5.36, 5.34, 5.38)	(3.03, 3.01, 2.93)	(1.52, 1.5, 1.16)	(0.64, 0.57, 0.14)
20	(53.14, 53.1, 53.26)	(41.83, 41.81, 41.95)	(30.01, 29.96, 30.07)	(18.86, 18.83, 18.87)	(9.34, 9.32, 9.39)	(4.97, 4.95, 4.99)	(2.81, 2.79, 2.72)	(1.42, 1.39, 1.08)	(0.59, 0.53, 0.13)
40	(42.47, 42.44, 42.55)	(33.83, 33.81, 33.92)	(24.56, 24.52, 24.59)	(15.61, 15.59, 15.62)	(7.82, 7.8, 7.86)	(4.19, 4.17, 4.2)	(2.38, 2.36, 2.3)	(1.2, 1.18, 0.91)	(0.5, 0.45, 0.11)
60	(31.8, 31.78, 31.85)	(25.82, 25.81, 25.88)	(19.1, 19.07, 19.12)	(12.37, 12.35, 12.36)	(6.29, 6.28, 6.32)	(3.4, 3.39, 3.41)	(1.94, 1.93, 1.88)	(0.99, 0.97, 0.75)	(0.42, 0.37, 0.09)
80	(21.19, 21.18, 21.22)	(17.82, 17.82, 17.85)	(13.64, 13.63, 13.65)	(9.12, 9.1, 9.11)	(4.77, 4.76, 4.79)	(2.62, 2.61, 2.62)	(1.51, 1.5, 1.46)	(0.78, 0.77, 0.59)	(0.33, 0.29, 0.07)
100	(12.15, 12.15, 12.17)	(10.59, 10.59, 10.61)	(8.46, 8.45, 8.46)	(5.94, 5.93, 5.93)	(3.25, 3.25, 3.26)	(1.84, 1.83, 1.84)	(1.08, 1.07, 1.04)	(0.57, 0.56, 0.43)	(0.24, 0.21, 0.05)

Figure 2.8: Comparison of prices obtained with BAST, ABAST and the copula method

BAST and ABAST perform well except when K is low and/or the barrier level is high (the mean of the quantities $u_{M,i}$ decreases).

2.4 The MLMC copula estimator

This last section of this chapter shows that using the copula method in conjunction with the MLMC method outperforms the performance of the copula method in itself. We first detail how to build this estimator then show how to reduce its variance.

2.4.1 Construction of the MLMC copula estimator

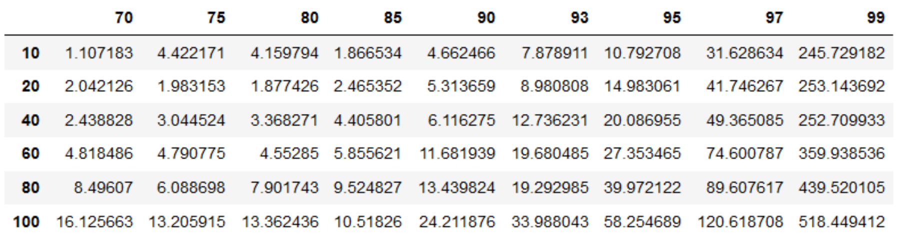
Suppose we are still in the setting of this chapter and would like to price a down-and-out option with a conditional payoff depending on the terminal values of the system (2.1). We have shown that the copula estimator was a solid method to price this kind of contract. Nevertheless, the numerical analysis conducted in 1.4 still holds and the overall cost of pricing those contracts using Euler-Maruyama and standard Monte-Carlo for a given level of accuracy ϵ scales worst than $O(\epsilon^{-3})$ (because the payoff is not Lipschitz). This can become extremely painful because it implies that to reduce the variance by a factor 2 one has to multiply the number of paths used in the Monte-Carlo pricing by a factor higher than 4 (factor whose exact value is contract-dependent). Therefore, and if we can apply theorem 6, the MLMC setting seems particularly appealing in this context.
Suppose we want to price a contract of this type with maturity T. As before, let $M = 2$ and $\Delta t_l = \frac{T}{2^l}$ the time step at level l. We define Y_0 to be the mid-point copula estimator at level zero for this contract, and let for $l > 1$ Y_l be the difference between the mid-point copula estimator at resolution l and at resolution $l - 1$. Namely, we can write:

$$Y_l(\hat{S}^{\Delta t_l}, \hat{S}^{\Delta t_{l-1}}) = E_M(\hat{S}^{\Delta t_l}) - E_M(\hat{S}^{\Delta t_{l-1}}) \tag{2.18}$$

where $\hat{S}^{\Delta t_l}$ and $\hat{S}^{\Delta t_{l-1}}$ are simulations of the system (2.1) using Euler-Maruyama with time step Δt_l and Δt_{l-1} for the same sampled Brownian motions vectors as shown in (1.26) and (1.25) (every corresponding component of the vector $W^{\Delta t_l, \mathcal{Q}}$ and $W^{\Delta t_{l-1}, \mathcal{Q}}$ are sampled this way). The mid-point MLMC copula estimator is then:

$$Y = \sum_{i=0}^{L} Y_l \tag{2.19}$$

Of course, we still need to get L and the number N_l of paths per level. This is done following the exact same procedure as the one in 1.4.3 namely using pilot runs for each level. This pricing method will converge towards the true value of the option if we can satisfy the hypothesis outlined in theorem 6. To show that this method indeed improves the computational time for a desired level of accuracy ϵ fixed at 0.25 we provide a table that outlines the ratio between the computational time using this method and the standard method of section 2.3.1. We take three assets with $\sigma^i = 0.4$, $r^i = 0.05$, $\rho_{i,j} = 0.5 \quad \forall i \neq j$, $S^i(t) = 100$, $T - t = 52$ weeks and we make the strike (index) and the barrier levels (index and we suppose them equal) vary:



	70	75	80	85	90	93	95	97	99
10	1.107183	4.422171	4.159794	1.866534	4.662466	7.878911	10.792708	31.628634	245.729182
20	2.042126	1.983153	1.877426	2.465352	5.313659	8.980808	14.983061	41.746267	253.143692
40	2.438828	3.044524	3.368271	4.405801	6.116275	12.736231	20.086955	49.365085	252.709933
60	4.818486	4.790775	4.55285	5.855621	11.681939	19.680485	27.353465	74.600787	359.938536
80	8.49607	6.088698	7.901743	9.524827	13.439824	19.292985	39.972122	89.607617	439.520105
100	16.125663	13.205915	13.362436	10.51826	24.211876	33.988043	58.254689	120.618708	518.449412

Figure 2.9: Ratio between the pricing time using the standard Copula method and the modified MLMC Copula method

As shown, this method outperforms the standard method by a significant margin. Again this ratio seems to be a function of the strike K and the barrier levels B^i.

2.4.2 Variance reduction of the MLMC copula estimator

Finally, we show here that using MLMC and the copula method together also allows for further variance reduction for the same computational time. In 1.4.3 of chapter 1, we slightly modify the default estimators by using mid-points for the coarse paths using a Brownian bridge construction. While this works in this particular case for the Brownian bridge estimator (1.17) and (1.18) because then the relation:

$$\mathbb{E}(Y_{l-1}^f) = \mathbb{E}(Y_l^c) \tag{2.20}$$

where Y_l^f is the fine estimator at level $l-1$ and Y_l^c is the coarse estimator at level l is satisfied for $l \geqslant 1$, it cannot be done for the copula MLMC estimator since adding points in such a fashion makes E_l and E_U in the coarcer levels converge faster towards the true value and therefore (2.20) is no longer satisfied. We therefore refer to [14] who shows that using antithetic paths allows for variance reduction of the MLMC estimator (in a setting that is quite general). Namely for a level l, once the fine and coarce Brownian paths $W_{l,1}$ and W_{l-1} are built using (1.26) and (1.25), we construct another fine Brownian path that is the reflection of the original fine path at every odd point (obtained by swapping every increment after the first one):

$$W_{l,2}^i(j\Delta t_l) = \begin{cases} W_{l,1}^i(j\Delta t_l) & \text{if } j \text{ is even} \\ W_{l,1}^i((j-1)\Delta t_l) + Z_{j+1} & \text{if } j \text{ is odd} \end{cases} \tag{2.21}$$

and we do that for every asset $S^i, i \in \{1, ..., n\}$. This gives us a collection of paths $S^{\Delta t_l,1}$, $S^{\Delta t_l,2}$ and $S^{\Delta t_{l-1}}$ Then the estimator at level $l \geqslant 1$ is modified to be:

$$Y_l(\hat{S}^{\Delta t_l,1}, \hat{S}^{\Delta t_l,2}, \hat{S}^{\Delta t_{l-1}}) = \frac{E_M(\hat{S}^{\Delta t_l,1}) + E_M(\hat{S}^{\Delta t_l,2})}{2} - E_M(\hat{S}^{\Delta t_{l-1}}) \tag{2.22}$$

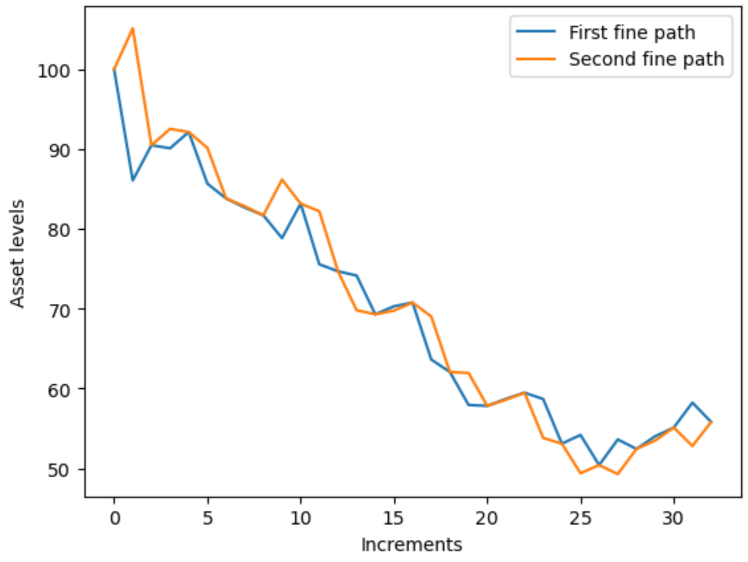
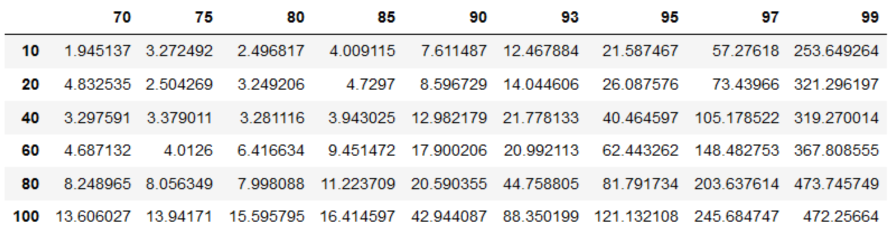


Figure 2.10: A fine path and its reflected path obtained by swapping Brownian increments

Those new estimators Y_l are valid because equality (2.20) is respected. To show how the variance is reduced, we conduct the same experiments as in the last subsection:



	70	75	80	85	90	93	95	97	99
10	1.945137	3.272492	2.496817	4.009115	7.611487	12.467884	21.587467	57.27618	253.649264
20	4.832535	2.504269	3.249206	4.7297	8.596729	14.044606	26.087576	73.43966	321.296197
40	3.297591	3.379011	3.281116	3.943025	12.982179	21.778133	40.464597	105.178522	319.270014
60	4.687132	4.0126	6.416634	9.451472	17.900206	20.992113	62.443262	148.482753	367.808555
80	8.248965	8.056349	7.998088	11.223709	20.590355	44.758805	81.791734	203.637614	473.745749
100	13.606027	13.94171	15.595795	16.414597	42.944087	88.350199	121.132108	245.684747	472.25664

Figure 2.11: Ratio between the pricing time using the standard Copula method and the modified MLMC Copula method with variance reduction

This shows that this new MLMC estimator performs better than the one in the last subsection for all the range of contracts tested here.

Chapter 3

Pricing Barrier contracts using pre-processing

This chapter presents an alternative to pricing barrier contracts in multi-dimension. We start by giving a contextualization of the problem then show how to approach probabilities of exit on a time interval by solving an (convex) optimization problem. We use this to build a 2 assets contract-specific surface which allows us to price the latter in the case of assets following the Black-Scholes dynamics. We then show how to adapt this to contracts who follow more general volatility dynamics and conduct in-depth analysis of this numerical method. Finally, we discuss how this method can be adapted for more than two assets and give axes of research.

3.1 Contextualization of the problem

We have seen in the last chapter that the copula method was a fast and reliable method to price multidimensional barrier contracts. However this method suffers some performance drawbacks because one needs to derive a lower and upper bound on the price of the contract. Also, this method does not allow the user to get some useful metrics such as the average crossing time of the asset and does not allow for a path-by-path analysis. Therefore, we would like to find another way to estimate the price of such contracts that allows for this and reduce the computing time. The method we will develop in the following sections can be seen as an extension of the Brownian Bridge method in the multidimensional case. As outlined before, we will start by considering a simplified case of (2.1) and will build on complexity. Specifically, suppose $n = 2$ and the two assets follow a geometric Brownian motion with constant correlation coefficient on $[t, T]$:

$$d\begin{pmatrix} S_t^1 \\ S_t^2 \end{pmatrix} = \begin{pmatrix} r^1 S_t^1 \\ r^2 S_t^2 \end{pmatrix} dt + \Sigma \begin{pmatrix} \sigma^1 S_t^1 dW_t^{1,\mathcal{Q}} \\ \sigma^2 S_t^2 dW_t^{2,\mathcal{Q}} \end{pmatrix} \tag{3.1}$$

where Σ is the Cholesky decomposition of the constant correlation matrix

$$\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \tag{3.2}$$

We will focus on a down-and-out call with strike K on the first asset than pays $(S^1(T) - K)^+$ conditional on none of the assets breaching their constant barrier on [t,T] but the following reasoning can be adapted to other type of payoffs.

3.2 Approximation of the probability of exit

3.2.1 Volatility equal to 1

In this section, we use the notations defined in section 2.2. We namely want to approximate (2.6) given an interval of time $[k\Delta t', (k+1)\Delta t']$ and initial and terminal values of the process (3.1) on this interval. What follows is entirely taken from [15]. To do this we will approximate the probability that both assets get knocked-out on this interval then use the relation:

$$P^{\rho}(E_1 \cup E_2) = P(E_1) + P(E_2) - P^{\rho}(E_1 \cap E_2) \tag{3.3}$$

Suppose as of now that both volatility functions are equal to 1. Denote by

$$X = \begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} \tag{3.4}$$

the logarithmic values of the assets on $[t, T]$. The process X follows a standard Brownian motion on $[t, T]$ whose drift is given by applying the Girsanov's theorem on S. If the process S is simulated perfectly through (3.1) then there is an equivalence between simulating S or X but for a maximum of generality we will in our simulations use Euler-Maruyama to simulate (3.1) and then simply take the logarithmic vector to get X. We do not detail the construction of the approximation (see [15] Section 2 pages 27-72) to not introduce repetition but we give the necessary simulation steps. Let $K = ((\Sigma\Sigma^T)^{-1})^{\frac{1}{2}}$ and this is a symmetric matrix of the form

$$K = \begin{pmatrix} k_1 & k_2 \\ k_2 & k_3 \end{pmatrix} \tag{3.5}$$

If we denote by $x \in \mathbb{R}^2$ the values that takes the process X at time $k\Delta t'$, by $y \in \mathbb{R}^2$ the values it takes at time $(k+1)\Delta t'$ and by $c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \log(B_1) \\ \log(B_2) \end{pmatrix}$ the value of the barriers, then we define the new vectors $x' = Kx$ and $y' = Ky$ and we define two hyper-planes that correspond to the boundaries of the open set ([15] p.38) :

$$F = (Kx, x \in D) \tag{3.6}$$

where D is

$$D = ((x_1, x_2), x_1 > c_1, x_2 > c_2) \tag{3.7}$$

Those equations are given by

$$k_3 z_1 - k_2 z_2 = c_1(k_1 k_3 - k_2^2) \tag{3.8}$$

$$-k_2 z_1 + k_1 z_2 = c_2(k_1 k_3 - k_2^2) \tag{3.9}$$

Now we denote by J_1 (resp. J_2) the points $\in \mathbb{R}^2$ that satisfy equation (3.8) (resp. (3.9)). The optimization problem we have to solve is in two parts ([15] p.46):

- Find

$$u(x, y, c)_1 = \inf_{\phi_1 \in J_1, \phi_2 \in J_2} (||x' - \phi_1|| + ||\phi_1 - \phi_2|| + ||\phi_2 - y'||) \tag{3.10}$$

  and

$$u(x, y, c)_2 = \inf_{\phi_1 \in J_2, \phi_2 \in J_1} (||x' - \phi_1|| + ||\phi_1 - \phi_2|| + ||\phi_2 - y'||) \tag{3.11}$$

- Take the minimum of those quantities $u(x, y, c)$.

In [15] (Lemma 2.2.1 p.53), it is proven that the two problems (3.10) and (3.11) are convex in the variables ϕ_1 and ϕ_2 which makes this optimization quite quick to perform as well with admitting a unique solution. Now the approximation of the probability that both components of X get knocked-out on $[k\Delta t', (k+1)\Delta t']$ knowing initial values x and terminal values y is given by ([15] p.41) :

$$p(x, y, \Delta t', 1, 1, c_1, c_2) = \exp(-\frac{u(x, y, c)}{\Delta t'}) \tag{3.12}$$

assuming of course that the initial values of x and y do not imply they have both crossed their barriers (in this case we set $p(x, y, \Delta t', 1, 1, c_1, c_2) = 1$).

3.2.2 Different volatility

When the volatilities in (3.1) are no longer equal to 1 but are equal to σ^1 and σ^2, we need to adapt the problem. Intuitively a Brownian motion with volatility σ will have normal increments with standard deviation σ, so that re-scaling the real value axis by $\frac{1}{\sigma}$ gives a standard Brownian motion. Whence, a Brownian motion that has an initial value x and a terminal value y over $[k\Delta t', (k+1)\Delta t']$ with a constant barrier c_1 and a volatility σ can be seen as a standard Brownian motion with initial value $\frac{x}{\sigma}$, terminal value $\frac{y}{\sigma}$ and barrier level $\frac{c_1}{\sigma}$ over this interval. This means that if X^1 has volatility σ^1 and X^2 has volatility σ^2 then we only need to re-scale x_1, y_1 and c_1 by $\frac{1}{\sigma^1}$ and x_2, y_2 and c_2 by $\frac{1}{\sigma^2}$ and then use 3.2.1 to get (3.12).

3.2.3 Limitations of this method

Using the above, it is theoretically possible knowing Σ to price a contract whose underlying assets follow the dynamic (3.1). However and despite the convexity of the optimization problem this is way too slow for real applications. Using an industry standard number of path equal to $N = 100000$ and using a monitoring period of say $\Delta t' = \frac{7}{365}$ roughly yields for a one year contract with 2 assets $2 \times 52 \times N{=}10^7$ optimizations of this sort to perform. On our machine (8 × Intel® Core™ i5-1155G7 Processor) this optimization takes approximately 5 ms to perform which means it would take approximately 10^5 s for one contract. Obviously this is not realistic and this method cannot be applied in this crude form: we therefore need to think of other way to proceed.

3.3 The correlation surface

In this section we construct a surface that uses last section and the law of Large Numbers. We first build that surface for specific contracts then propose a transformation that allows to use one surface for a wide range of contracts. Finally, we show that this new method behaves very well even for problematic contracts and discuss some of the shortcomings of this method and discuss what could possibly be done to overcome them.

3.3.1 On a single interval

Suppose once more the assets follow (3.1) and that we sit on the time interval $[k\Delta t', (k+1)\Delta t']$. The law of the vector $X = \log(\mathrm{S})$ at $(k+1)\Delta t'$ knowing what values X takes at $k\Delta t'$ is given by:

$$X_{(k+1)\Delta t'} | X_{k\Delta t'} \hookrightarrow \mathbf{N}(\begin{pmatrix} X^1_{k\Delta t'} + (r^1 - \frac{{\sigma^1}^2}{2})\Delta'_t \\ X^2_{k\Delta t'} + (r^2 - \frac{{\sigma^2}^2}{2})\Delta'_t \end{pmatrix}, \Sigma\sqrt{\Delta t'}) \tag{3.13}$$

Now, this means that we can sample points from the above distribution (3.13). Sampling enough points in $\mathbf{R}^2$ allows us to find desirable quantities but in particular allows us to use the transfer theorem for any function. Let us consider the following function:

$$f(X,Y) = \frac{P^{\rho}(E_1 \cup E_2)}{P^0(E_1 \cup E_2)}(X,Y) \tag{3.14}$$

where the vectors X and Y design the initial and terminal values of the process (3.1) given some realization of the process. In particular, f should be understood as "the ratio between the probability of exit under correlation ρ and zero of the process S given initial and terminal values on the interval $[k\Delta t', (k+1)\Delta t']$". The transfer theorem then states if we are given some initial values X that:

$$\mathbb{E}(f(X,Y)|X) = \int_{\mathbb{R}^2} f(X,y)dP(y) \tag{3.15}$$

where P is the density given in (3.13). Of course, to get (3.15) we could use a 2 dimensional-grid or simply (which is better for higher dimensions) sample points with (3.13) and then use Monte-Carlo to get an approximation of (3.15). The function $(X,Y) \longrightarrow f(X,Y)$ is equal to 1 if any component of X or Y is below the barrier vector B which makes the latter attractive. Now that we get a good estimation of (3.15) we can use the law of Large Numbers to state that on average the function $Y \longrightarrow P^0(E_1 \cup E_2)(X,Y) \times \mathbb{E}(f(X,Y)|X)$ should be a good approximation of the function $Y \longrightarrow P^{\rho}(E_1 \cup E_2)(X,Y)$. This is what we will do to estimate the price of the option.

3.3.2 The contract-specific surface

Let $\rho = 0.5$ through this section. Suppose again that the assets dynamics follow (3.1) and suppose we got a constant barrier level $B = \begin{pmatrix} B^1 \\ B^2 \end{pmatrix}$ for the assets for a down-and-out call on S^1 with strike K, maturity T and monitoring period $\Delta t'$. We create a grid of "initial values" in the following way:

$$\Theta = \{(X_1, X_2) \in [B^1, \infty[\times [B^2, \infty[\} \tag{3.16}$$

Of course numerically we replace $[B^1, \infty[$ by $\{B^1, B^1 + \text{step}, ..., B^1 + K_1 \times \text{step}$ and $[B^2, \infty[$ by $\{B^2, B^2 + \text{step}, ..., B^2 + K_2 \times \text{step}$ where K_1 and K_2 can be taken as upper bounds of the price of the assets on $[t, T]$ using a standard deviation argument (or something else)) and "step" corresponds to the resolution of the grid. We then apply the procedure outlined in 3.3.1 for every point of this grid which gives a surface where every point of this surface is the Monte-Carlo estimate of (3.15) for this initial value. Below we show a specific surface where $B^1 = B^2 = 90$, $\sigma^1 = \sigma^2 = 0.3$, $r^1 = r^2 = 0.05$, $S^1(t) = S^2(t) = 100$, $T - \tau = 52$ weeks and $\Delta t' = \frac{7}{365}$:

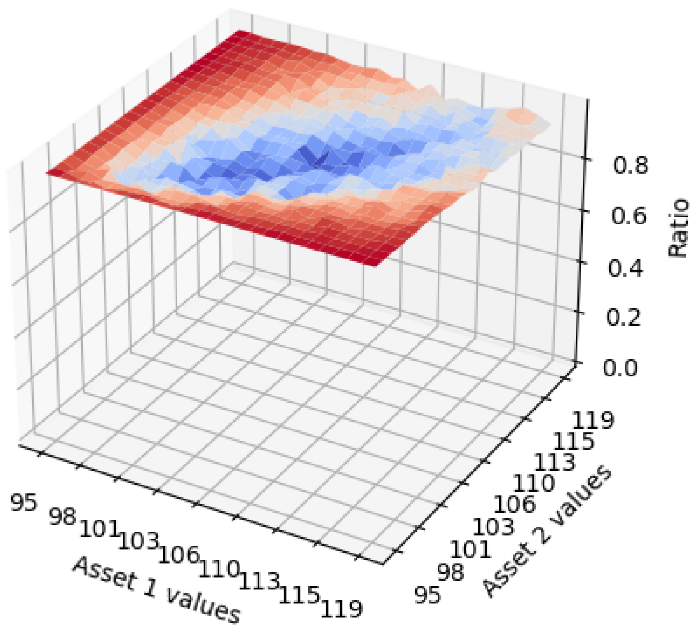


Figure 3.1: Contract specific surface

As expected this surface is symmetric. We used $N' = 50$ for the Monte-Carlo simulation for every point in Θ which means we sampled 50 points in (3.13) and solved less than 50 optimization problems for every $X \in \Theta$ (we recall that $f = 1$ if any component of X or Y is below the barrier level B which means there is no need to solve an optimization problem in this case). Because we want the quantity $\mathbb{E}(f(X,Y)|X)$ for every possible initial point X that the asset price can take in the Monte-Carlo simulations, we use a standard interpolator from the `scipy`: `RegularGridInterpolator`. To extrapolate, we say that outside of Θ the ratio is equal to 1 which is makes sense looking at the surface. To price the contract using the surface we simulate our paths as with the naive pricing procedure and for every interval of every path we compute $P^0(E_1 \cup E_2)$ for which there is a closed-form formula. Then we simply multiply the latter by $\mathbb{E}(f(X,Y)|X)$ for the initial values X at the left-hand of the time interval. The price of the contract is then given by the formula:

$$E_{SC} = \frac{1}{N}\sum_{i=1}^{N}\big((\hat{S}_i^{\Delta t,1}(T) - K)^+ \cdot 1_{\nexists s\in[t,T] \text{ s.t } \hat{S}_i^{\Delta t,j}(s)<B_j \quad \forall j\in\{1,\ldots,n\}}\big) \times \text{Discount}(\hat{S}_i^{\Delta_t}) \tag{3.17}$$

where M is the number of monitoring points and N is the number of simulated paths and where

$$\text{Discount}(\hat{S}_i^{\Delta_t}) = \prod_{k=0}^{M-1} P^0(E_1 \cup E_2)\mathbb{E}(f(\hat{S}_i^{\Delta t}(k\Delta t'), \hat{S}_i^{\Delta t}((k+1)\Delta t')))|\hat{S}_i^{\Delta t}(k\Delta t'))) \tag{3.18}$$

We choose a strike $K = 10$ and the benchmark price for this contract is 16.70 (obtained using the copula method from Chapter 2). We get the following histogram of price with our method and 50 repetitions:

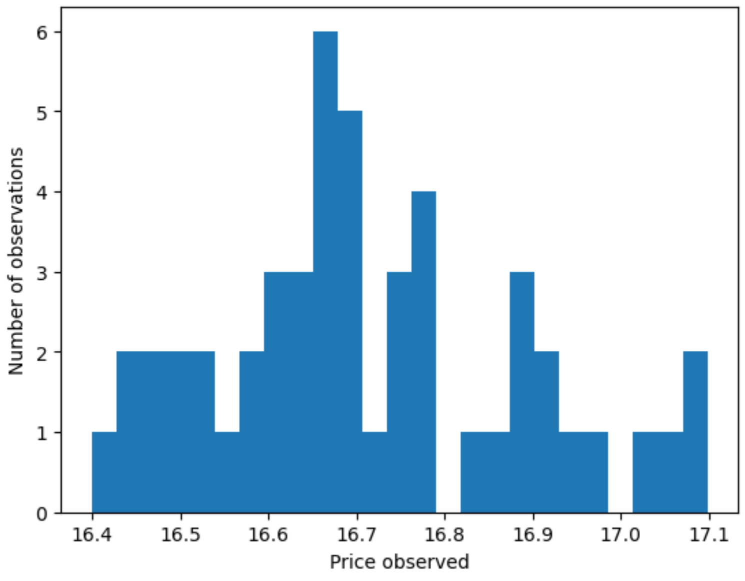


Figure 3.2: Histogram of prices. Mean price is 16.71

The mean price is spot on. Note also that obviously we do not need to re-calibrate the surface with K changing.
Also, unlike the copula method, doing so allows us to compute other metrics such as the average crossing-time of the barrier, etc.

3.3.3 From a contract-specific surface to a general surface

The last subsection shows that it is possible to get the price of a barrier option using this method. However and from a practical point of view, having to calibrate a surface for every contract is prohibitive in terms of computational time. Also, all the latter has been developed in the case where the assets follow the dynamics given by (3.1). Usually practitioners use local volatility models or even stochastic local volatility models for the asset's dynamics. To overcome this and in the case where S^1 has a volatility function $(t,x) \longrightarrow \sigma^1(t,x)$ and S^2 has a volatility function $(t,x) \longrightarrow \sigma^2(t,x)$ we could compute M correlation surfaces following 3.3.2 where each correlation surface corresponds to the spot-dependent volatility functions $x \longrightarrow \sigma^1(k\Delta t', x)$ and $x \longrightarrow \sigma^2(k\Delta t', x)$ where k varies from 0 to $M-1$ and $\Delta t'$ is the monitoring period. Even if this allows us to theoretically get the contract fair price it is computationally very expensive and we see no plausible way to extend this when the volatility functions are no longer deterministic.
Suppose we are given a vector of initial values $S(k\Delta t')$ and a vector of terminal values $S((k+1)\Delta t')$ for the time interval $[k\Delta t', (k+1)\Delta t']$ for the process S that has the dynamics on [t,T] given by (3.1) but with volatility processes given by σ_t^1 and σ_t^2 that are general (deterministic or not). We approximate the volatility functions on the interval $[k\Delta t', (k+1)\Delta t']$ by their left values and we denote by σ^1 and σ^2 their approximated values on this interval of time. To compute the probability that both components of the process cross the barrier level vector B on the time interval we use the log-levels of the price and the barriers as described in 3.2. However, the log-price dynamics of the assets over $[k\Delta t', (k+1)\Delta t']$ is governed by an arithmetic Brownian motion and is therefore not spot-dependent which means the probability of exit over $[k\Delta t', (k+1)\Delta t']$ as given in (3.12) is a function of the distance between the logarithmic value of S and the logarithmic value of B. Also and as pointed out before, this distance does not necessarily have to be the absolute distance but can also take into account the volatility of the assets on this time interval as well as the monitoring period $\Delta t'$. The standard deviation of S^1 over this time interval being equal to $\sigma^1\sqrt{\Delta t'}$ and the one of S^2 being equal to $\sigma^2\sqrt{\Delta t'}$ we come with the following theorem:

Theorem 8. *The probability of exit of S $P^\rho(E_1 \cup E_2)$ over the time interval $[k\Delta t', (k+1)\Delta t']$ with* *barrier level B knowing initial and terminal values is equal to the probability of exit of a 2-dimensional standard Brownian motion with barrier level 0, with same correlation matrix with initial values* $X = \begin{pmatrix} \frac{log(S^1(k\Delta t') - log(B^1)}{\sigma^1\sqrt{\Delta t'}} \\ \frac{log(S^2(k\Delta t') - log(B^2)}{\sigma^2\sqrt{\Delta t'}} \end{pmatrix}$ *and terminal values* $Y = \begin{pmatrix} \frac{log(S^1((k+1)\Delta t') - log(B^1)}{\sigma^1\sqrt{\Delta t'}} \\ \frac{log(S^2((k+1)\Delta t') - log(B^2)}{\sigma^2\sqrt{\Delta t'}} \end{pmatrix}$ *over the time interval* $[0, 1]$.

Proof. Subsections (3.2.2) and (3.2.1) show that discounting the initial and terminal values x' and y' in the optimization problems (3.10) and (3.11) yields the same exact solution: if $x = \begin{pmatrix} \log(S^1(k\Delta t')) \\ \log(S^2(k\Delta t')) \end{pmatrix}$, $y = \begin{pmatrix} \log(S^1((k+1)\Delta t')) \\ \log(S^2((k+1)\Delta t')) \end{pmatrix}$, $c_1 = \log(B^1)$ and $c_2 = \log(B^2)$ then we have

$$p(x, y, \Delta t', \sigma^1, \sigma^2, c_1, c_2) = p(X, Y, 1, 1, 1, 0, 0)$$

$\square$

Suppose throughout the rest of this section that $r^1 = \frac{\sigma^{1}}{2}^2$ and $r^2 = \frac{\sigma^{2}}{2}^2$ so that the log-process of (3.1) has zero drift. This theorem is useful for us because it means that given a correlation matrix we only need to build one surface for the process X with dynamics given by:

$$d\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} = \Sigma \begin{pmatrix} dW_t^{1,\mathcal{Q}} \\ dW_t^{2,\mathcal{Q}} \end{pmatrix} \tag{3.19}$$

over $[0, 1]$ and barrier level $B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Given a grid

$$\Theta = \{(X_0^1, X_0^2) \in [0, \infty[\times[0, \infty[\} \tag{3.20}$$

we discretize it numerically and we sample points according to the following distribution

$$X_1|X_0 \hookrightarrow \mathbf{N}(\begin{pmatrix} X_0^1 \\ X_0^2 \end{pmatrix}, \Sigma) \tag{3.21}$$

for every initial value in discretized Θ and then compute the Monte-Carlo approximation of the ratio function f under correlation 0 and ρ knowing this initial value. We therefore get a surface for this process as shown below:

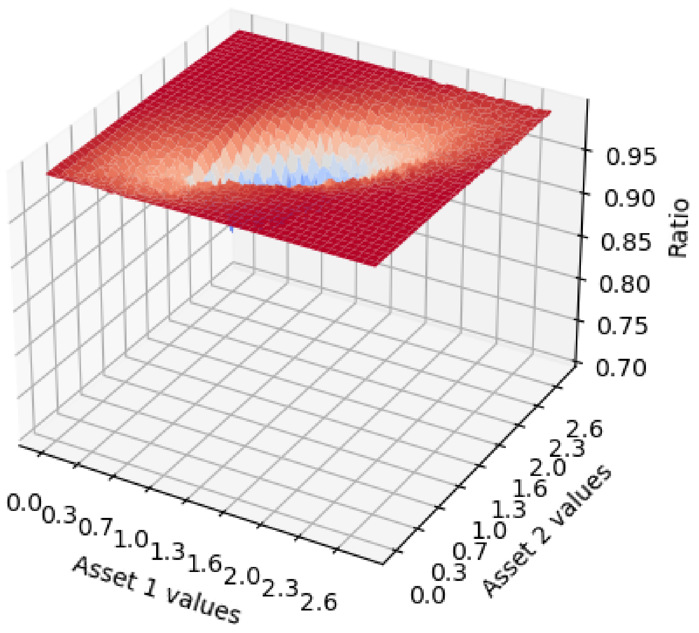


Figure 3.3: General surface

We interpolate and extrapolate this surface as done previously for the contract-specific surface. Now if we want to price a contract where the underlying dynamics are given by (3.1) but where we allow the volatility functions to be more general than in the Black-Scholes setting and the barrier level to be time-dependent, we proceed as described in 3.3.2 but we modify the discount function: if we denote by $(x, y) \longrightarrow \gamma(x, y)$ the surface function we write for a simulated path:

$$\text{Discount}(\hat{S}_i^{\Delta_t}) = \prod_{k=0}^{M-1} P^0(E_1 \cup E_2)\gamma(\beta(\hat{S}_i^1(k\Delta t'), B^1(k\Delta t'), \sigma^1_{k\Delta t'}, \Delta t'), \beta(\hat{S}_i^2(k\Delta t'), B^2(k\Delta t'), \sigma^2_{k\Delta t'}, \Delta t')) \tag{3.22}$$

where

$$\beta(x, y, z, t) = \frac{\log(x) - \log(y)}{z\sqrt{t}} \tag{3.23}$$

is the "normalization" function. This surface can virtually be used for any 2-dimensional process with this specific correlation matrix and as long as the probability of exit over a time interval can be computed using (3.2). Moreover this approach is fairly general and can be coupled with variance reduction techniques. Below we give a table that compares the results and computing time we obtain with our technique against the copula technique for different monitoring periods and barrier levels for down-and-out barrier contracts. We take 2 assets with $S^i(t) = 100$, $\rho_{1,2} = 0.5$, $T - t = 52$ weeks, $r^i = 0.05$ and $\sigma^i = 0.3$. We make the barrier levels vary (columns) as well as the strike level (index). Each cell is computed using 50 repetitions and gives respectively the mean price, the standard deviation and the average time taken to sample the path and compute the payoff:

	70	75	80	85	90	93	95	97	99
10	(66.07, 0.2, 7.49)	(55.02, 0.28, 5.74)	(42.34, 0.26, 3.62)	(29.24, 0.18, 3.62)	(16.68, 0.21, 3.36)	(9.96, 0.16, 3.26)	(6.13, 0.14, 3.37)	(2.89, 0.06, 3.2)	(0.53, 0.03, 3.18)
20	(59.93, 0.21, 4.54)	(49.97, 0.24, 4.25)	(38.69, 0.22, 3.98)	(26.87, 0.25, 3.65)	(15.42, 0.14, 3.32)	(9.27, 0.13, 3.17)	(5.67, 0.11, 1303.56)	(2.68, 0.07, 3.08)	(0.49, 0.02, 3.28)
40	(47.63, 0.16, 4.36)	(40.31, 0.19, 4.2)	(31.53, 0.2, 3.95)	(22.09, 0.16, 3.72)	(12.81, 0.13, 3.62)	(7.75, 0.12, 3.51)	(4.75, 0.12, 3.36)	(2.28, 0.07, 3.34)	(0.43, 0.02, 3.29)
60	(35.35, 0.15, 4.51)	(30.41, 0.14, 4.56)	(24.3, 0.17, 4.2)	(17.34, 0.14, 4.23)	(10.24, 0.13, 4.07)	(6.26, 0.1, 3.86)	(3.88, 0.08, 3.77)	(1.83, 0.05, 3.55)	(0.35, 0.02, 3.34)
80	(23.22, 0.13, 4.8)	(20.66, 0.12, 4.41)	(17.08, 0.13, 4.18)	(12.61, 0.1, 3.85)	(7.65, 0.09, 3.65)	(4.76, 0.08, 3.53)	(2.97, 0.05, 3.5)	(1.44, 0.04, 3.51)	(0.27, 0.02, 3.43)
100	(13.01, 0.1, 4.61)	(11.94, 0.1, 5.04)	(10.31, 0.09, 4.52)	(7.98, 0.08, 4.19)	(5.11, 0.07, 3.85)	(3.27, 0.06, 3.83)	(2.08, 0.05, 3.73)	(1.02, 0.03, 3.63)	(0.2, 0.01, 3.64)

Figure 3.4: Statistics using the correlation surface method

	70	75	80	85	90	93	95	97	99
10	(66.08, 0.24, 6.27)	(54.87, 0.26, 4.04)	(42.4, 0.23, 4.26)	(29.16, 0.2, 4.13)	(16.67, 0.19, 4.34)	(10.04, 0.18, 4.26)	(6.1, 0.11, 4.32)	(2.87, 0.08, 4.53)	(0.5, 0.02, 4.56)
20	(59.94, 0.21, 4.37)	(50.04, 0.23, 4.16)	(38.72, 0.17, 4.46)	(26.82, 0.23, 4.27)	(15.39, 0.15, 4.35)	(9.27, 0.12, 4.32)	(5.68, 0.11, 4.34)	(2.68, 0.08, 4.44)	(0.47, 0.02, 4.42)
40	(47.62, 0.17, 4.58)	(40.24, 0.16, 4.42)	(31.51, 0.18, 4.44)	(22.15, 0.15, 4.46)	(12.8, 0.17, 4.65)	(7.74, 0.11, 4.54)	(4.78, 0.09, 4.81)	(2.27, 0.05, 5.23)	(0.4, 0.02, 4.96)
60	(35.34, 0.14, 4.95)	(30.47, 0.17, 4.56)	(24.36, 0.15, 4.72)	(17.35, 0.19, 4.6)	(10.22, 0.12, 4.67)	(6.25, 0.1, 4.59)	(3.88, 0.05, 4.6)	(1.83, 0.04, 4.57)	(0.33, 0.01, 4.7)
80	(23.18, 0.1, 4.63)	(20.66, 0.14, 4.68)	(17.1, 0.13, 5.37)	(12.6, 0.11, 5.01)	(7.64, 0.08, 4.81)	(4.74, 0.07, 4.87)	(3.0, 0.06, 4.9)	(1.44, 0.04, 4.92)	(0.25, 0.01, 4.15)
100	(13.04, 0.11, 3.76)	(12.0, 0.09, 3.78)	(10.29, 0.08, 4.02)	(7.98, 0.07, 3.89)	(5.11, 0.07, 3.9)	(3.26, 0.06, 3.73)	(2.08, 0.05, 3.7)	(1.03, 0.03, 3.89)	(0.18, 0.01, 3.75)

Figure 3.5: Statistics using the copula method

As one can see this method performs fairly well for all the contracts considered and outperforms the copula method in terms of speed; there is no need to compute two estimators with this method.
One could argue with the results above that the prices are spot on because $\Delta t'$ is relatively small and so for each time interval of every path we have $P^0(E_1 \cup E_2) \approx P^\rho(E_1 \cup E_2)$. The next graph shows how the mean price evolves as $\Delta t'$ increases with our method against one where we only use $P^0(E_1 \cup E_2)$ to estimate the probability of exit on the intervals. We take 2 assets with $S^i(t) = 100$, $\rho_{1,2} = 0.5$, $T - t = 52$ weeks, $r^i = 0.045$, $\sigma^i = 0.3$, $K = 50$ and $B^i = 90$ for $i \in \{1, 2\}$:

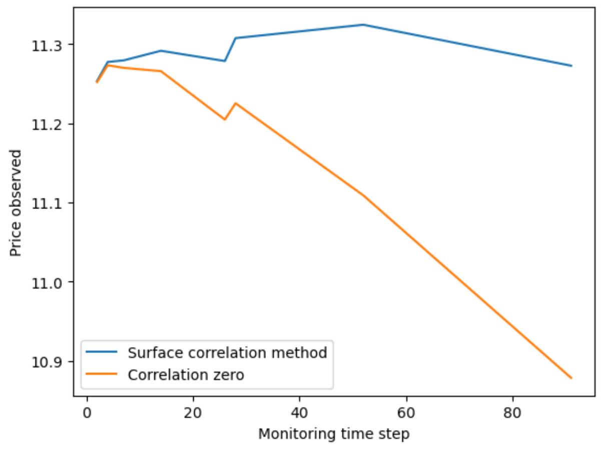


Figure 3.6: Price evolution with the time step using the surface correlation technique

The price computed with this method stays approximately the same when the monitoring time step changes. To compare, the same experiment gives with the copula technique and the mid-point estimator the following graph:

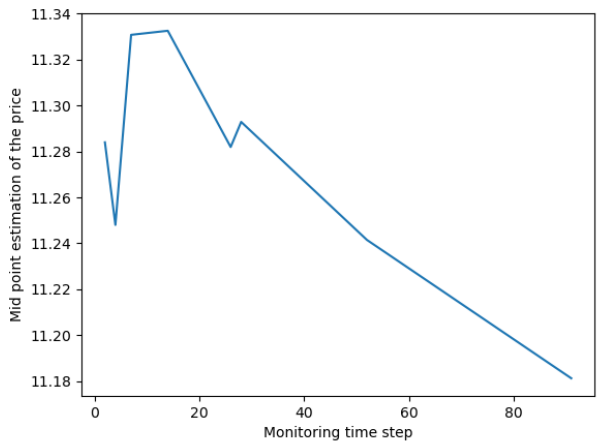


Figure 3.7: Price evolution with the time step using the copula method

Because of this transformation, the surface we use to price those contracts only depends of the correlation matrix. Specifically the volatility levels, barrier levels and monitoring period do not have any influence anymore on the pricing; we use their left-hand approximations to compute the coordinates of the ratio estimate on the surface through the function β. Therefore we can handle other dynamics for the assets: local volatility, stochastic local volatility... with no more efforts than in the Black-Scholes case. We can use different monitoring periods for different periods of the life-time of the contract and the barrier level can also be time-dependent: all that matters is the relative distance spot-barrier i.e. the quantities $u_{M,i}$. The only quantity that can trigger issues are the drift functions: we will see in the next section how to adapt the latter.

3.3.4 Discussion and extension

The main drawback of this method is that we need to calibrate this surface for every possible correlation matrix which means we need to calibrate it for every $\rho \in [-1, 1]$. Also, we said that this surface could be used to price any contract when the underlying assets follow general dynamics in terms of drift and volatility functions; while in practice this is true, in theory the drift functions should modify the way we compute $\mathbb{E}(f(Y, X)|X)$. To finish with, we did all of the above with the 2 assets. How should one adapt the above for more underlyings ? The next section details those issues and proposes solutions.

3.4 Beyond the correlation surface

3.4.1 Smoothing and interpolation

Suppose we have a surface as explained in 3.3.3 for a certain ρ. Because we use Monte-Carlo to get estimates of (3.15) there is some noise in the estimates. Below is shown the estimates of the surface shown in 3.3.3 for $X^1 = 1$ when X^2 varies from zero to 3 and $\rho = 0.5$:

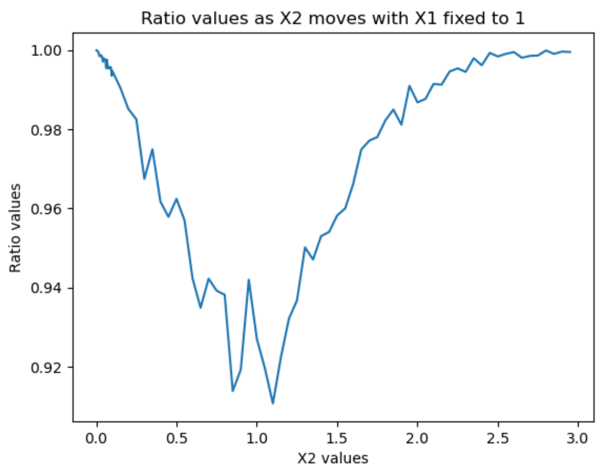


Figure 3.8: Variation of the estimates with X^1 fixed and X^2 varying

To get rid of the noise and prepare interpolation inter-surfaces, we use a Salgov filter. We plot the same graph with the smoothed data on top of it:

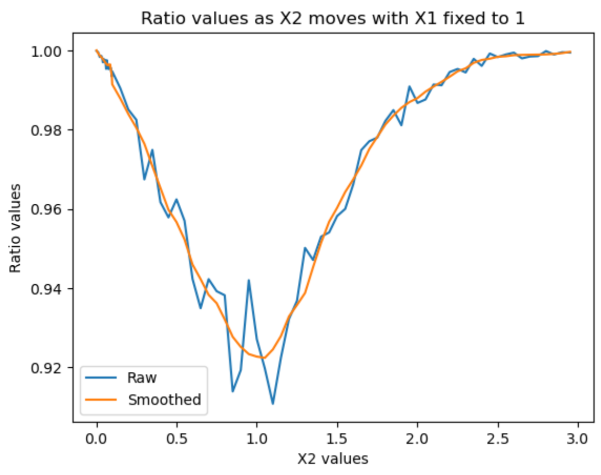
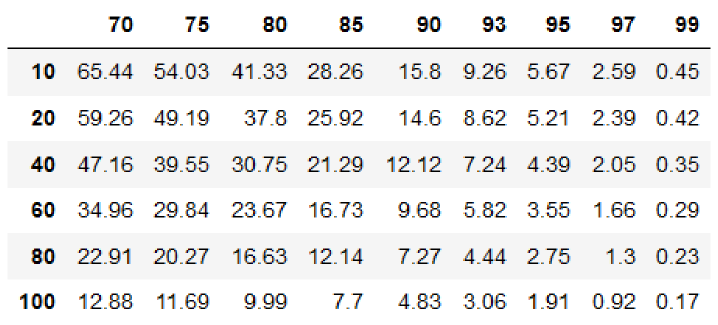


Figure 3.9: Variation of the estimates with X^1 fixed and X^2 varying

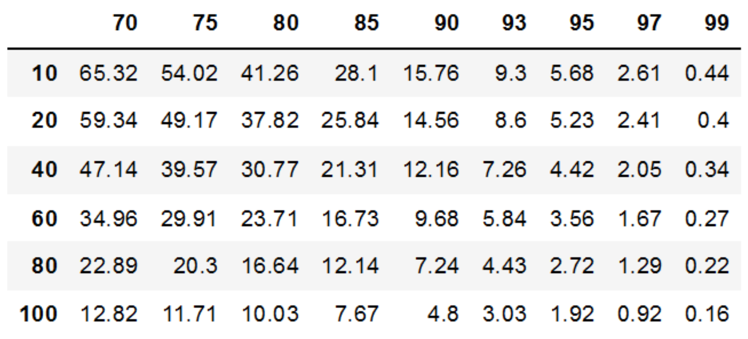
To be able to price any contract for any correlation matrix (i.e. any ρ) we first build a surface for every $\rho \in \alpha = \{-1, -0.9, ..., 0.9, 1\}$ with the same Θ grid every time. We smooth them using Salgov filter then interpolate the surfaces along $\{-1, -0.9, ..., 0.9, 1\}$ so that if we have any $\rho \in [-1, 1]$ we are able to retrieve the corresponding surface and estimates. Of course this section only outlines the idea and a way to proceed but one can think of many other ways to proceed using Machine Learning, other ways to interpolate, thinner α... There is virtually no limit to the amount of surfaces one can compute and the degree of accuracy (N') to do so because all of this computational time only has to be spent once and is totally separate from the pricing. Below we show the same table as in the last subsection for $\rho_{1,2} = 0.45$ where the surface estimates used in the pricing have been obtained through linear interpolation. We set $S^i(t) = 100$, $r^i = 0.05$, $\sigma^i = 0.3$, $T - t = 52$ weeks, and we make K (index) and the equal barrier levels B^i (columns) vary:



	70	75	80	85	90	93	95	97	99
10	65.44	54.03	41.33	28.26	15.8	9.26	5.67	2.59	0.45
20	59.26	49.19	37.8	25.92	14.6	8.62	5.21	2.39	0.42
40	47.16	39.55	30.75	21.29	12.12	7.24	4.39	2.05	0.35
60	34.96	29.84	23.67	16.73	9.68	5.82	3.55	1.66	0.29
80	22.91	20.27	16.63	12.14	7.27	4.44	2.75	1.3	0.23
100	12.88	11.69	9.99	7.7	4.83	3.06	1.91	0.92	0.17

Figure 3.10: Pricing with linearly interpolated estimates for $\rho = 0.45$

For the benchmark prices, we use the copula method:



	70	75	80	85	90	93	95	97	99
10	65.32	54.02	41.26	28.1	15.76	9.3	5.68	2.61	0.44
20	59.34	49.17	37.82	25.84	14.56	8.6	5.23	2.41	0.4
40	47.14	39.57	30.77	21.31	12.16	7.26	4.42	2.05	0.34
60	34.96	29.91	23.71	16.73	9.68	5.84	3.56	1.67	0.27
80	22.89	20.3	16.64	12.14	7.24	4.43	2.72	1.29	0.22
100	12.82	11.71	10.03	7.67	4.8	3.03	1.92	0.92	0.16

Figure 3.11: Pricing with the copula method for $\rho = 0.45$

The prices are sticking to the benchmark and this demonstrates the generalization and applicability of this technique.

3.4.2 Offsetting the drift

In the last section, we have been elusive on the impact of the drift functions on the pricing by setting this drift to zero for the logarithmic price process. Given an initial vector of values X we sample points according to (3.21) which yields terminal values for a two-dimensional arithmetic Brownian motion with zero drift which corresponds to the case where the two assets follow a log-normal distribution with drift equal respectively to $\frac{{\sigma^1}^2}{2}$ and $\frac{{\sigma^2}^2}{2}$ using Ito's lemma. This means that any estimates computed in the last section 3.3.3 assumes that the assets drift is constant equal to the above on every monitoring period $[k\Delta t', (k+1)\Delta t']$. But if the first asset has a drift r^1 and the second asset had a drift r^2 on a time interval of length $\Delta t'$ then we have for S^1 for example:

$$\frac{S^1((k+1)\Delta t')}{S^1(k\Delta t')}|S^1(k\Delta t') \hookrightarrow \text{lognormal}((r^1 - \frac{{\sigma^1}^2}{2})\Delta t', (\sigma^1)^2\Delta t') \tag{3.24}$$

which means we should sample points for the generic surface according to the following distribution (by taking the logarithms in the above expression and using β):

$$X_1|X_0 \hookrightarrow \mathbf{N}(\begin{pmatrix} X_0^1 + (r^1 - \frac{{\sigma^1}^2}{2})\frac{\sqrt{\Delta t'}}{\sigma^1} \\ X_0^2 + (r^2 - \frac{{\sigma^2}^2}{2})\frac{\sqrt{\Delta t'}}{\sigma^2} \end{pmatrix}, \Sigma) \tag{3.25}$$

to obtain our estimates. This differs of what we did since we sampled points for the generic surfaces according to (3.21). In practice, the term $(r^1 - \frac{{\sigma^1}^2}{2})\frac{\sqrt{\Delta t'}}{\sigma^1}$ is so small it can be neglected (with $r = 0.1$,

$\sigma^1 = 0.3$ and $\Delta t' = \frac{7}{365} = 1$ week it is equal to 0.035) when we estimate the ratio from the surface (the pricing tables shown above in 3.3.3 show this technique still provides prices that are very close to the benchmark ones). To take into account this drift, we do not modify the way the surfaces in 3.3.3 are built but the way we look at them. By setting $X_0^{1'} = X_0^1 + (r^1 - \frac{{\sigma^1}^2}{2})\frac{\sqrt{\Delta t'}}{\sigma^1}$ and $X_0^{2'} = X_0^2 + (r^2 - \frac{{\sigma^2}^2}{2})\frac{\sqrt{\Delta t'}}{\sigma^2}$ then we can refer to the vector of coordinates X_0' to estimate the probability of exit since X_1 is obtained from X_0 using (3.21). It simply means we modify the transformation function in the following way:

$$\beta(x, y, z, t, r) = \frac{\log(x) + (r - \frac{z}{2}^2)t - \log(y)}{z\sqrt{t}} \tag{3.26}$$

and use again (3.22) to price the option. To show how our pricing results are modified when doing the latter compared with our methodology from 3.3.3, we set $S^i(t) = 100$, $\rho_{i,j} = 0.5$, $T - \tau = 52$ weeks, $r^i = 0.5$, $\sigma^i = 0.3$, $K = 50$ and $B^i = 97$. Then we plot the evolution of the price as a function of $\Delta t'$ for the price obtained using (3.26) and (3.23):

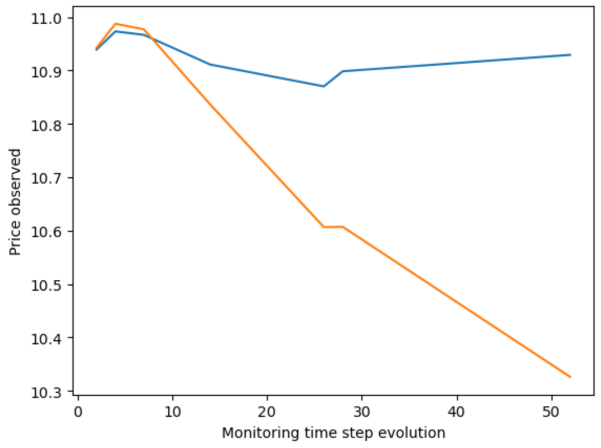


Figure 3.12: Adjusting the drift vs not adjusting the drift

Not adjusting for the drift when using (3.22) causes the computed price to deviate from the true price (10.96) when $\Delta t'$ increases but adjusting for the drift makes it stay at the same level regardless of the monitoring time step. Note however that $r^i = 0.5$ is unlikely to happen in practice and for more general volatility processes it is not recommended to have a large $\Delta t'$ anyways.

3.4.3 The method in higher dimensions

So far we have developed this technique for two correlated assets. We have shown how to deal with more general processes than the Black-Scholes one and have shown that this method is faster than the copula method for the same level of accuracy. If we want to make this technique work for a number n of correlated assets we need to adapt the above framework. The idea stays the same: given a correlation matrix $P \in \mathbb{R}^{n \times n}$ we create a grid $\Theta = \{(X_0^1, X_0^2, ..., X_0^n) \in [0, \infty[^n\}$, discretize it and for every point X in the latter we compute an approximation of $\mathbb{E}(f(X, Y)|X)$ where f is now

$$f(X, Y) = \frac{P^\rho(\bigcup_{i=1}^n E_i)}{P^0(\bigcup_{i=1}^n E_i)}(X, Y) \tag{3.27}$$

Then using the β function and interpolation we should theoretically be able to price any contract with a continuous barrier using the same procedure as in 3.3.3. To compute an approximation of $P^\rho(\bigcup_{i=1}^n E_i)$ we refer to [15] (pages 49-55); this is the same concept as the one developed in 3.2 but for vectors of n initial and terminal values. Obviously and even though constructing those n-dimensional surfaces for every P matrix is completely separate from the pricing aspect, there is an explosion of the computational time.

Whereas in dimension 2 computing 20 of those surfaces is virtually enough to price any contract with 2 correlated Brownian motions (by interpolating the surfaces) this is not the case anymore for n assets. If we restrict ourselves to correlation values in $\{-1, -0.9, ..., 0.9, 1\}$ we can build roughly $21^{\frac{n(n-1)}{2}}$ correlation matrix (minus the repetitions). Besides every one of those correlation matrix has its own grid Θ and every estimate in this discretized grid is obtained solving N' optimization problems who take way longer to perform than for the 2-dimensional case. Therefore trying to extend this method crudely for n correlated assets is a bad idea. To extend this method in higher dimension we therefore propose two approaches. The first one would be to find a way to parametrize the function f i.e find a good approximation of this function in terms of ρ and the initial and terminal values X, Y and extend it for $n \geqslant 2$. Because our numerical experiments show this function is smooth in those parameters we believe it would be possible to do so. A good approximation would allow to overcome those numerical issues when computing the ratio estimates. A parametrization of the surfaces in 3.3.3 should also be possible thanks to their smoothness which would lead to the same conclusion. The second approach is to approximates the probability of exit $P(\bigcup_i = 1^n E_i)$ by the one of the worst k performers with $k < n$. By worst performers we mean the assets with higher quantities u_M: to approximate this probability use either the method developed in 3.3.3 or a parametrization of this probability (or of f). Those approaches are not mutually exclusive and could lead to faster, more accurate pricing while retaining the advantages of the latter method (price stable with monitoring period, etc...).

3.5 A machine learning approach as an alternative

This last section proposes an alternative to what has been done before. Instead of approximating the probability of exit $P^\rho(E_1 \cup E_2)$ by $P^0(E_1 \cup E_2)\mathbb{E}(f(\beta(X), \beta(Y))|\beta(X))$ we will rather approximate it by $P^0(E_1 \cup E_2)\hat{f}(\beta(X), \beta(Y))$ where $\hat{f}$ is a fitted machine learning model that approximates f. To be more specific, given a correlation matrix, we sample points according to the distribution (3.21) and use those points as features for the model we want to fit. Then when we want to price a barrier contract on underlyings sharing this correlation matrix we use this machine learning model to adjust the probability of exit under correlation zero to get the one under correlation ρ. The Monte-Carlo payoff for a down-and-out call therefore reads in this setting:

$$E_{ML} = \frac{1}{N}\sum_{i=1}^{N}\left((\hat{S}_i^{\Delta t,1}(T) - K)^+ \cdot 1_{\nexists s \in [t,T] \text{ s.t } \hat{S}_i^{\Delta t, j}(s) < B_j \quad \forall j \in \{1,...,n\}}\right) \times \text{DiscountML}(\hat{S}_i^{\Delta_t}) \tag{3.28}$$

with:

$$\text{DiscountML}(\hat{S}_i^{\Delta_t}) = \prod_{k=0}^{M-1} P^0(E_1 \cup E_2)\hat{f}(\beta(S_i^{\Delta t}(k\Delta t')), \beta(S_i^{\Delta t}((k+1)\Delta t'))) \tag{3.29}$$

This alternative to the surface technique has some advantages: the use of a machine learning model allows for a faster calibration given the model captures the distribution well. It also allows us to ignore the influence of the drift since the function f is not drift-dependent (since the impact of the drift is entirely captured by the terminal point Y) even though $X \longrightarrow \mathbb{E}(f(\beta(X), \beta(Y))|\beta(X))$ is.

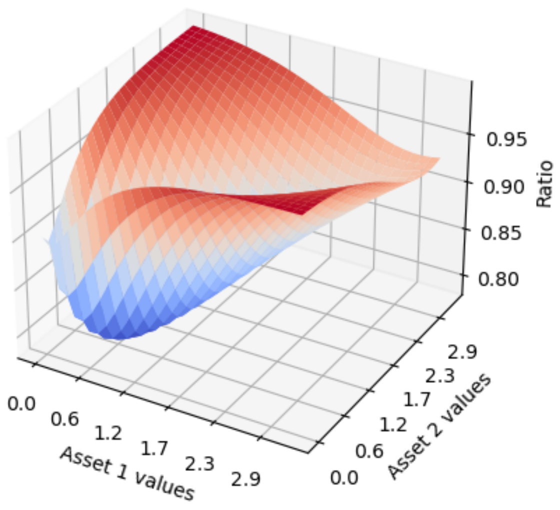
3.5.1 Features and calibration

As mentioned before, the essential features of the model should be the initial vector X and terminal vector Y. We randomly sample points X using an uniform distribution over $[0, 6]^2$ which is the standard image interval for β. Then for each point X we sample Y using (3.21). To speed up and improve the quality of the fit, we compute for each pair of initial and terminal values the quantities $P(E_1)$, $P(E_2)$ and $P^0(E_1 \cup E_2)$. We pass $P(E_1)$ and $P(E_2)$ as features for the model. If either $P(E_1) < \alpha$ or $P(E_2) < \alpha$

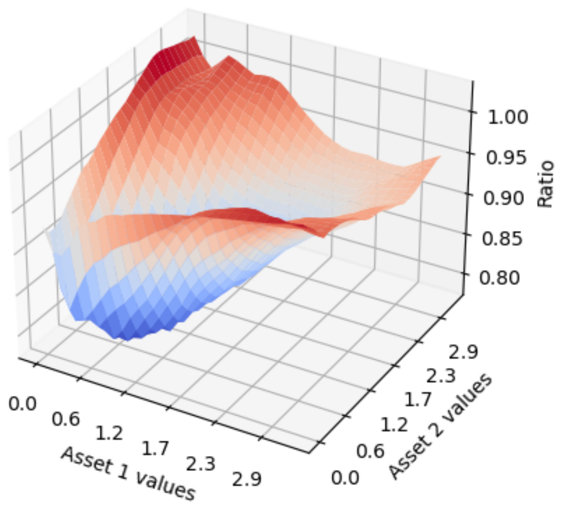
with α arbitrary small (say 10^{-3}) then we can reasonably assume that $P^0(E_1 \cup E_2) \approx P^\rho(E_1 \cup E_2)$ and remove this pair of initial/terminal values from the training sample. We do this as well for any sampled point Y that has a component at least below the barrier. We use the package `scikit_learn` for the models calibration.

3.5.2 Which model to use ?

The fitted model should be accurate but perhaps most importantly should be fast to make new predictions. To assess the quality of the model we can sample points as with the training set and then compare the predictions to the test target values. Also the model should correctly capture the dynamics of the ratio distribution which means that the accuracy of the predictions should stay relatively stable when X changes. We found that using a neural network yielded the best results; while a KNN algorithm might seem a better choice because we are not limited in the number of points we sample to fit the model its prediction time makes it a prohibitive choice. Below we show how a neural network trained with a dataset of 300000 samples approximates $Y \longrightarrow f((1,1),.)$:



(a) Actual values of f



(b) Predicted values of f

Figure 3.13: Comparison of Actual and Predicted Values

Of course, it is possible to achieve better results using other algorithms, or using separate neural networks for different partition of the discretized image space of β. On a test set build in the same fashion as the training set, the latter model performs very well in term of prediction quality as the below histogram shows with an error that appears centered around 0:

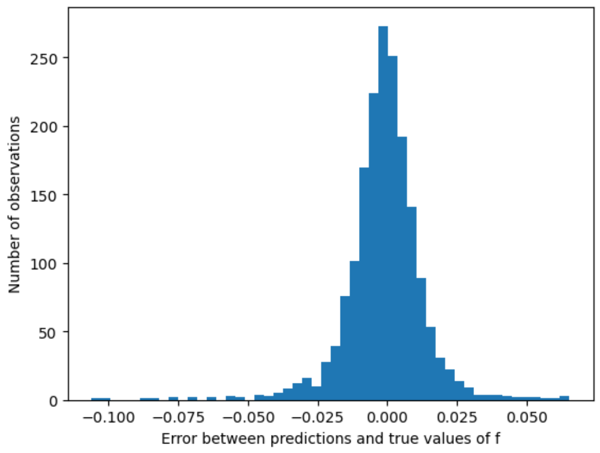
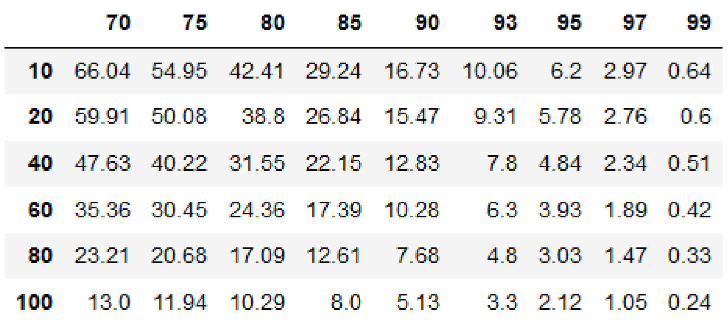


Figure 3.14: Predicted values vs actual values of f

3.5.3 Pricing results

We use this algorithm to price the same contracts as with the correlation surface technique obtained in 3.3.3. Namely, we take 2 assets with $S^i(t) = 100$, $\rho_{1,2} = 0.5$, $T - t = 52$ weeks, $r^i = 0.05$ and $\sigma^i = 0.3$. We make the barrier levels vary (columns) as well as the strike level (index) and get the table below. We take a large monitoring time step $\Delta t' = \frac{26}{365}$ so that the prices we compute are different from the ones we would get discounting only with $P^0(E_1 \cup E_2)$ for each time step of each simulated path:



	70	75	80	85	90	93	95	97	99
10	66.04	54.95	42.41	29.24	16.73	10.06	6.2	2.97	0.64
20	59.91	50.08	38.8	26.84	15.47	9.31	5.78	2.76	0.6
40	47.63	40.22	31.55	22.15	12.83	7.8	4.84	2.34	0.51
60	35.36	30.45	24.36	17.39	10.28	6.3	3.93	1.89	0.42
80	23.21	20.68	17.09	12.61	7.68	4.8	3.03	1.47	0.33
100	13.0	11.94	10.29	8.0	5.13	3.3	2.12	1.05	0.24

Figure 3.15: Pricing using a neural network

The benchmark table is given in 3.3.3. We can see this method performs fairly well for most of the entries; the relatively large differences in the last columns is mostly due to the large values of $u_{M,i}$. Also as outlined in 3.4.2 another transformation needs to be done to offset the drift. However using a fitted machine learning model as detailed above should allow us to price a contract correctly no matter the drifts function for the asset.

The results show this method is also general and applicable in fairly generic situations. To be able to price any contract for any $\rho \in [-1, 1]$ we can fit several machine learning algorithms for a discretized range of correlation value or fit a general machine learning model for sampled points in the β space where ρ is randomly sampled in $[-1, 1]$ and used as a feature. Doing the latter allows to only fit one machine learning model to price any contract but the quality of the fit might get worse since the model does not learn specifically for a given correlation value; therefore when we were sampling random points to fit the algorithm before one might need to be more careful in constructing the training set to make sure the accuracy of the model is somewhat preserved. However this is an interesting approach: finding an adequate training set to minimize the test error across all correlation values would allow for an extension of this solution to higher dimensions while keeping a low training time.

Conclusion

The first chapter of this master's thesis addressed the single-asset case. We introduced the Brownian bridge techniques and the barrier shifting techniques, and we demonstrated that the former was more reliable than the latter. Building on the concepts presented in [10], we illustrated that combining the Multi-Level Monte-Carlo paradigm with Brownian bridge techniques enabled rapid and accurate pricing.

The second chapter outlined the copula method and evaluated the performance of barrier shifting techniques in the context of multi-asset scenarios. We implemented Multi-Level Monte-Carlo estimators using the copula method and observed a significant increase in computational time.

In the final chapter, we utilized the approximation technique for exit probabilities from [15] to construct mathematical models for pricing barrier contracts on assets through the application of the law of large numbers. We provided a detailed exposition of this alternative technique and discussed potential extensions, particularly in multi-asset scenarios. Additionally, we explored the potential application of machine learning techniques to this problem and considered its potential extensions.

The aim was to comprehensively investigate various techniques and methodologies for pricing barrier options across different asset scenarios, providing insights into both traditional and innovative approaches.

Bibliography

[1] Kyng Tim, Konstandatos Otto *Images and Barriers on the Road to Real Options.* Journal of Real Options, 12(3), 123-136. 2021.

[2] Hull, John C. *Options, Futures and Other Derivatives: Global Edition.* 2010

[3] Vaes, Urbain. *Lecture Notes on Topic.* URL: `https://urbain.vaes.uk/static/teaching/lectures/build/lectures-w6.pdf`

[4] Bally, Vlad, and Denis Talay. *The Euler Scheme for Stochastic Differential Equations: Error Analysis with Malliavin Calculus.* Mathematics and Computers in Simulation, 38(1-3), 35-41. 1995.

[5] Gobet Emmanuel *Advanced Monte Carlo methods for barrier and related exotic options.* 2008

[6] Gobet, E. and Menozzi, S. *Discrete Sampling of Functionals of Itô Processes.* Séminaire de Probabilités XL.

[7] Shevchenko, Pavel V. *Addressing the Bias in Monte Carlo Pricing of Multi-Asset Options with Multiple Barriers through Discrete Sampling.* The Journal of Computational Finance, 6(3), 1-20. January 2003.

[8] Broadie, M., Glasserman, P., and Kou, S. *A Continuity Correction for Discrete Barrier Options.* 1997.

[9] Higham, Desmond J. *An Introduction to Multilevel Monte Carlo for Option Valuation.*

[10] Giles, Michael B. *Multilevel Monte Carlo Path Simulation.* Oxford University Mathematical Institute, and Oxford—Man Institute of Quantitative Finance.

[11] Giles, Michael B., Debrabant, Kristian, and Rößler, Andreas. *Analysis of Multilevel Monte Carlo Path Simulation Using the Milstein Discretisation.*

[12] Giles, Mike, Higham, Desmond J., and Mao, Xuerong. *Analyzing Multi-level Monte Carlo for Options with Non-globally Lipschitz Payoff.* August 2008.

[13] Giles, Mike. *Multilevel Monte Carlo for Multi-dimensional SDEs.* Oxford University Mathematical Institute, Oxford-Man Institute of Quantitative Finance.

[14] Giles, Michael B. and Szpruch, Lukasz. *Antithetic Multilevel Monte Carlo Estimation for Multi-dimensional SDEs without Levy Area Simulation.*

[15] Huh, Joonghee. *Computation of Multivariate Barrier Crossing Probability, and Its Applications in Finance.*

[16] Lapeyre, Bernard. *Introduction to Monte-Carlo Methods.* Halmstad, January 2007.