# Optimised Importance Sampling in Multilevel Monte Carlo


Candidate Number: 194537

Hertford College

University of Oxford

A thesis submitted in partial fulfillment of the MSc in

*Mathematical and Computational Finance*

Trinity Term 2015

# Acknowledgements

First, I would like to thank my supervisor Prof. Mike Giles for his continued support, guidance and enthusiasm for this project.

Secondly, many thanks to the people of Hertford College: you have made my experience in Oxford truly unforgettable. Especially Jamie. You have been an invaluable companion in our restless chase for unexplored libraries.

A final thanks to my family. *Sarete sempre il mio porto sicuro.*

# Abstract

This dissertation explores the remarkable variance reduction effects that can be achieved combining Multilevel Monte Carlo and Importance Sampling. The analysis is conducted within a Black-Scholes framework, focusing on pricing deep out-of-the-money options. Particular attention is addressed to the choice of the Importance Sampling measure and to the optimisation of its parameters. Numerical results show that the combination of the two methods significantly outperforms both techniques if applied separately.

Key words: Monte Carlo, Multilevel Monte Carlo, Option Pricing, Importance Sampling, Variance Reduction

# Contents

# List of Figures

# Chapter 1

# Introduction

Monte Carlo methods are a flexible approach to approximating expectations in a stochastic framework, and are widely employed in the financial industry in order to price a variety of securities. Among the advantages of this technique, we can mention relative simplicity of implementation, lower computational cost when dealing with high dimension problems (e.g. compared to Finite Difference methods) and predisposition towards parallel computing.

## 1.1 The essence of Monte Carlo methods

Let us consider an arbitrage-free complete market, where it is possible to define the so called Risk-Neutral probability measure $\mathbb{Q}$, under which the discounted price process $(e^{-rt}\, S_t)_{\,0\leq t\leq T}$ of an underlying is a martingale. By the Feynman-Kac lemma and non-arbitrage arguments, the price $V$ of a European derivative whose payoff is $P$ can be computed as:

$$V(t,s) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}P(S_T)\,|\,S_t = s\right] \tag{1.1}$$

Monte Carlo methods approximate the price $\hat{V}$ of the option by simulating a set of $N$ independent samples $\{\hat{S}_T^{(i)}\}_{i=1}^N$ drawn from the distribution of the underlying price process $S_T$ under the measure $\mathbb{Q}$ and computing the average of the payoff corresponding to each sample. Namely:

$$\hat{V} \;=\; \frac{1}{N}\sum_{i=0}^{N} e^{-r(T-t)}P\left(\hat{S}_T^{(i)}\right) \;\triangleq\; \frac{1}{N}\sum_{i=0}^{N}\hat{P}^{(i)} \tag{1.2}$$

The simple arithmetic average $\hat{V}$ is an unbiased estimator of $V(t,s)$ whose variance is $\nu^2/N$, where $\nu^2 = \mathbb{V}^{\mathbb{Q}}[\hat{P}]$

The validity of Monte Carlo methods relies on the Central Limit Theorem, which states that:

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=0}^{n}\hat{P}^{(i)}-\mathbb{E}^{\mathbb{Q}}\left[\hat{P}\right]\right)\xrightarrow[n\to\infty]{\mathcal{D}}\quad\mathcal{N}\left(0,\nu^{2}\right)\tag{1.3}$$

provided that $\nu^2 < \infty$. Therefore, it is possible to produce an asymptotically valid confidence interval for the Monte Carlo estimate:

$$\mathbb{Q}\left\{\mathbb{E}^{\mathbb{Q}}[\hat{P}]\in\mathcal{I}\right\}\triangleq\mathbb{Q}\left\{\hat{V}-3\,\nu/\sqrt{N}\leq\mathbb{E}^{\mathbb{Q}}\left[X\right]\leq\hat{V}+3\,\nu/\sqrt{N}\right\}\approx 99.7\%\tag{1.4}$$

It is evident then that the accuracy of the Monte Carlo estimate depends on the width of the confidence interval $\mathcal{I}$, which scales as $O(1/\sqrt{N})$. As well as increasing the number of samples $N$, it is possible to achieve greater accuracy by reducing the variance of the estimator $\hat{V}$. It is not surprising then that most part of the research in Monte Carlo simulations is addressed to the problem of reducing the variance.

## 1.2 Financial Background

Throughout the whole dissertation, we will consider a complete and arbitrage-free Black-Scholes market. Let $r > 0$ the risk-free interest rate and $\sigma$ the volatility, which we will assume to be constant throughout time. Let $S(t)$ be the price process of the underlying risky asset, whose evolution is described by the following SDE under the risk-neutral measure $\mathbb{Q}$:

$$dS_t\;=\;rS_tdt\;+\;\sigma S_t dW_t^{\mathbb{Q}},\qquad S_0=s\tag{1.5}$$

where $W_t^{\mathbb{Q}}$ is a $\mathbb{Q}$-Brownian motion. For each Monte Carlo sample, we will simulate a path of the SDE (1.5). We will discretise the time interval $[0;T]$ with a uniform time grid. M will denote the number of time-steps and $h \triangleq T/M$. We will make use of two different schemes to approximate the SDE (1.5):

1. Euler-Maruyama: $$\hat{S}_{n+1}\;=\;\hat{S}_n\left(1+rh+\sigma\Delta W_n\right),\quad\hat{S}_0\;=\;s\tag{1.6}$$

1. Milstein: $$\hat{S}_{n+1}\;=\;\hat{S}_n\left(1+rh+\sigma\Delta W_n+\frac{\sigma^2}{2}\left(\Delta W_n^2-h\right)\right),\;\hat{S}_0\;=\;s\tag{1.7}$$

where the Brownian increments are distributed as:

$$\Delta W_n\quad\text{i.i.d.}\;\sim\;\mathcal{N}(0,h)\tag{1.8}$$

### 1.2.1 Accuracy of the discretisation

Three key quantities to assess the accuracy of the approximation are defined as follows:

$$\text{Strong Error:} \qquad SE\left[\hat{S}_M\right] = \sqrt{\mathbb{E}\left[\|\hat{S}_M - S_T\|^2\right]} \tag{1.9}$$

$$\text{Weak Error:} \qquad WE\left[\hat{P}\right] = \mathbb{E}[\hat{P}] - \mathbb{E}\left[P\right] \tag{1.10}$$

$$\text{Root Mean Square Error:} \qquad RMSE\left[\hat{V}\right] = \sqrt{\mathbb{E}\left[\left(\hat{V} - V\right)^2\right]} \tag{1.11}$$

Since the drift and diffusion coefficients of the SDE (1.5) respect Lipschitz continuity and linear growth conditions, it is known that $WE = O(h)$, whereas $SE = O(\sqrt{h})$ for the Euler-Maruyama approximation and $SE = O(h)$ for the Milstein approximation (please refer to [KP92], chapters 9-10 and [Gla04], pp. 344-347 for proofs and further details).

On the other hand, it requires a simple piece of algebra to show that:

$$\mathbb{E}\left[\left(\hat{V} - V\right)^2\right] = \mathbb{V}[\hat{V}] + WE\left[\hat{P}\right]^2 \tag{1.12}$$

Therefore, $RMSE\left[\hat{V}\right] = O\left(1/\sqrt{N} + h\right)$.

## 1.3 Digital Put Option

Throughout the dissertation, we will consider two options as a case study to assess the performance of different methods. The first one is the European Digital Put, whose payoff has the following expression:

$$P(S_T) = \mathbb{1}_{\{S_T < K\}} \tag{1.13}$$

for a fixed strike price K. An analytic closed formula for its price can be derived, for example, by solving analytically the associated Black-Scholes PDE (refer to [WDH95] for full detail).

$$V(s,t) = \mathbb{Q}\left(S_T < K \,|\, S_t = s\right) = \Phi\left(\frac{\log(s/K) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}\right) \tag{1.14}$$

where $\Phi(\cdot)$ is the Standard Normal cumulative distribution function. We will focus on deep out-of-the-money Digital Put options, i.e. such that $S_0 \gg K$. This sort of securities are widely employed in the industry, for example as an insurance against default risk of the underlying financial asset. Ending in-the-money at maturity is a

rare event and simulations of low probability events intrinsically tend to have high variance. If we were to estimate the expected value of the indicator of a rare event (this might be the case of an out-of-the-money digital option), such that $\mathbb{P}(X = 1) = \epsilon \ll 1$ and $\mathbb{P}(X = 0) = 1 - \epsilon$, the statistical error of a Monte Carlo simulation relative to the estimate would be:

$$\frac{\mathrm{sd}[X]}{\mathbb{E}[X]} = \frac{\sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2}}{\mathbb{E}[X]} = \sqrt{\frac{\epsilon - \epsilon^2}{\epsilon^2}} \approx \sqrt{\frac{1}{\epsilon}} \tag{1.15}$$

Therefore, variance reduction is essential.

### 1.3.1 Payoff smoothing

For reasons that will be clear in chapter 4, the discontinuity in the Digital Put payoff would be highly detrimental to the application of Multilevel Monte Carlo methods. However, various payoff smoothing techniques are available. Throughout this dissertation, we will rely on the Conditional Expectation approach as explained in [Gla04] (pp. 399-400). This method consists in the following approximation:

$$\hat{P}^{CE}(\hat{S}_{T/h}) = \mathbb{E}\left[ \mathbb{1}_{\hat{S}_{T/h} < K} \,|\, \hat{S}_{T/h-1} \right] \tag{1.16}$$

A simple application of the tower rule shows that it is an unbiased estimator. Under Euler-Maruyama approximation, it is known that:

$$\hat{S}_{T/h} \,|\, \hat{S}_{T/h-1} \sim \mathcal{N}\left( (1 + rh)\, \hat{S}_{T/h-1},\, h\sigma^2\, \hat{S}^2_{T/h-1} \right) \tag{1.17}$$

Therefore, (1.16) takes the form:

$$\hat{P} = \Phi\left( \frac{K - (1 + rh)\, \hat{S}_{T/h-1}}{\sigma\sqrt{h}\, \hat{S}_{T/h-1}} \right) \tag{1.18}$$

## 1.4 Down-and-in Call Option

The second derivative we will analyse to assess the performance of different methods is the Down-and-in Call Option. This path-dependent derivative delivers the same payoff of a usual Call Option at time $T$, upon condition that the stock price $S_t$ falls below a pre-determined barrier B at least once during the option lifetime. Its payoff takes the form:

$$P\{(S_t)_{0 \le t \le T}\} = (S_T - K)^+ \, \mathbb{1}_{\{M_t < B\}}, \quad \text{where} \quad M_t = \inf_{0 \le t \le T} S_t \tag{1.19}$$

for a fixed value of the strike price K and the barrier value B.

In chapter 12 of [WDH95], the following analytic expression for the price of a Down-and-in Call Option is derived:

$$V(S_t, t) \;=\; \left(\frac{S_t}{B}\right)^{1-2r/\sigma^2} C\left(\frac{B^2}{S_t}, t\right) \tag{1.20}$$

where $C(S_t, t)$ is the Black-Scholes price for a vanilla Call with the same strike and maturity. Since the payoff is naturally formulated in continuous time, it is necessary to introduce a discretisation before in order to perform a Monte Carlo simulation. A naive approach to discretisation would be the following:

$$\hat{P} \;=\; (\hat{S}_{T/h} - K)^+ \mathbb{1}_{\mathcal{T}\neq\varnothing}, \quad \text{where} \quad \mathcal{T} \;=\; \{i = 1, \ldots, T/h : \hat{S}_i < B\} \tag{1.21}$$

However, this would lead to an undesirable weak error of order $O(\sqrt{h})$ (the same order as the average standard deviation of the discretised path $\{\hat{S}\}$ between two consequent time-steps).

### 1.4.1 Restoring the O(h) weak error

It is possible to overcome the issue by analysing the distribution of the driving Brownian Increment within a time-step conditional on its value at the extremes. Following the same approach as in chapter 6.4 of [Gla04], we will adopt the following discretisation of the option payoff, which restores a $O(h)$ order for the weak error.

$$\hat{P} \;=\; \left(\hat{S}_{T/h} - K\right)^+ \left(1 - \prod_{n=1}^{T/h}(1 - P_n)\right), \quad \text{where:} \tag{1.22}$$

$$P_n \;=\; \exp\left\{-\frac{2(\hat{S}_n - B)^+(\hat{S}_{n-1} - B)^+}{h\,\sigma^2 \hat{S}_{n-1}^2}\right\} \tag{1.23}$$

is the probability of $S_t$ crossing the barrier B in the time interval $[t_{n-1}; t_n]$ conditional on the values at the extremes. This particular choice leaves the payoff differentiable, which is beneficial for the application of Multilevel Monte Carlo.

# Chapter 2

# Importance Sampling

Throughout this section, we will present one of the most effective and at the same time delicate variance reduction techniques - Importance Sampling. The following explanation is mainly based on chapter 9 of [Owe13], section 4.6 of [Gla04] and section 2.7 of [BBG97].

Importance Sampling lays its foundations in the theory of changes of measure: in fact, it consists in drawing samples for a Monte Carlo simulation from a more convenient distribution (the concept of convenience will be clarified throughout this section), weighting them by the corresponding Radon-Nikodym derivative.

When simulating the payoff of deep out-of-the-money options, the probability of obtaining a payout is low and it is likely that most part of the simulated paths return value zero. Through a suitable change of measure, it is possible to increase the probability of a payout, allowing us to obtain more information out of each simulation run.

## 2.1 Motivations and general theory

Specifically, let us consider the problem of estimating the following quantity:

$$\mu_f \;=\; \mathbb{E}_p\left[f(X)\right] \;=\; \int_{\mathcal{D}} f(x)p(x)dx \tag{2.1}$$

where $p(\cdot)$ is the distribution of the random variable X and $\mathcal{D}$ is its support. Let us consider another distribution $s(\cdot)$ such that:

$$\mathcal{D} \subseteq \mathcal{S} = \mathrm{spt}(s), \quad \text{or equivalently that} \quad p(x) > 0 \rightarrow s(x) > 0 \tag{2.2}$$

This condition ensures that it is possible to express (2.1) as:

$$\mathbb{E}_p\left[f(X)\right] = \int_{\mathcal{D}} f(x)\,\frac{p(x)}{s(x)}\,s(x)\,dx = \int_{\mathcal{Q}} f(x)\,\frac{p(x)}{s(x)}\,s(x)\,dx = \mathbb{E}_s\left[f(X)\,\frac{p(X)}{s(X)}\right] \tag{2.3}$$

where $\mathbb{E}_s[\,\cdot\,]$ denotes expectation with respect to the measure defined by $s(\cdot)$. We define $f(X)p(X)/s(X)$, $X \sim s$ as the Importance Sampling estimator, which is unbiased for $\mu_f$. The ratio $p(x)/s(x)$ is referred to as Likelihood Ratio or Radon-Nikodym derivative.

Therefore, we define the corresponding Monte Carlo Importance Sampling estimator as:

$$\hat{V}^{IS} = \frac{1}{N}\sum_{i=1}^{N} f(X^{(i)})\,\frac{p(X^{(i)})}{s(X^{(i)})}, \quad \text{where} \quad X^{(i)} \text{ i.i.d.} \sim s \tag{2.4}$$

So far, we have just mentioned the assumptions in order for the Importance Sampling estimator to exist and to be unbiased. Let us focus on the original purpose of reducing the variance of our Monte Carlo estimate. The quantity we aim to minimise is the following:

$$\begin{gathered}\mathbb{V}_s\left[f(X)\,\frac{p(X)}{s(X)}\right] = \mathbb{E}_s\left[\left(f(X)\,\frac{p(X)}{s(X)} - \mu_f\right)^2\right] = \\ \int_{\mathcal{Q}}\left(\frac{f(x)\,p(x)}{s(x)} - \mu_f\right)^2 s(x)\,dx = \int_{\mathcal{Q}}\frac{[\,f(x)\,p(x) - \mu_f s(x)\,]^2}{s(x)}\,dx\end{gathered} \tag{2.5}$$

Let us assume that $f(x) \geq 0,\ \forall x \in \mathcal{Q}$ (it does not imply any loss of generality, since $f$ will be a discounted payoff in the following sections). The integrand in equation (2.5) is positive for all possible choices of $s(\cdot)$ and a minimum of 0 can be attained at $s(x) = f(x)p(x)/\mu_f$. Of course, such choice is not feasible, since it would require us to know $\mu_f$ beforehand in order to estimate $\mu_f$ itself. However, this approach provides useful insight about a convenient choice of $s(\cdot)$, which should satisfy:

$$s(x) \propto f(x)p(x) \tag{2.6}$$

Please refer to [KM53] for more rigorous proofs of optimality. Intuitively, we are designing $s(x)$ such that it generates more samples in the regions that contribute more to the value of $f$, or equivalently, we are choosing $s(\cdot)$ such that $f(\cdot)p(\cdot)/s(\cdot)$ is as close as possible to being constant, which implies having lower variance.

### 2.1.1 Importance Sampling with Markov Chains

The same reasoning exposed in the previous section can be extended to the case when $f(\cdot)$ is a function of a multivariate random variable. We will focus on the special case

of a discrete-time Markov Chain $\{S_n\}$. The Radon-Nikodym derivative takes then the form:

$$\frac{p(S_1, S_2, \ldots, S_n)}{s(S_1, S_2, \ldots, S_n)} = \frac{p(S_n \,|\, S_{n-1})\, p(S_1, S_2, \ldots, S_{n-1})}{s(S_n \,|\, S_{n-1})\, s(S_1, S_2, \ldots, S_{n-1})} = \ldots = \prod_{i=1}^{n} \frac{p(S_i \,|\, S_{i-1})}{s(S_i \,|\, S_{i-1})} \quad (2.7)$$

## 2.2 Importance Sampling in the Option Pricing framework

Let us clarify what Importance Sampling implies in the framework we have described in section 1.2, where the randomness driving the underlying price process $(S_t)$ is encapsulated in the Brownian Motion $W_t$. When performing Monte Carlo simulations, each payoff path is produced from a random (normal or uniform) number generator throughout the following steps:

$$\left(U^{(i)} \xrightarrow{\Phi(\cdot) \text{ inversion}}\right) Z^{(i)} \xrightarrow{\text{covar. matrix}} \{\Delta W_n^{(i)}\} \xrightarrow{\text{Euler-Maruyama}} \{\hat{S}_n^{(i)}\} \xrightarrow{e^{-rT}P(\cdot)} \hat{P}^{(i)} \quad (2.8)$$

Importance Sampling can be applied at each step of the process, provided that the distribution of the corresponding Random Variable can be estimated.

As we have seen so far, the only restriction on the choice of $s(\cdot)$ is condition (2.2) (alongside with an analysis of the tails of the two distributions). The optimality condition (2.6) does not give precise indication about the features of the new distribution function, which could belong to any family of distributions.

In order to choose what sort of change of measure to apply, we have reviewed some of the most relevant papers about Importance Sampling applied to Option Pricing. The vast majority of the literature keeps the Gaussian framework unaltered. Specifically, in [Cap08, GHS99, Aro03] the change of measure consists in adding a drift vector to the normal random vector $Z$; analogously, in [SF00], the change of drift is applied to the Brownian Increments, whereas in [VAD98] the drift term is added to the process $\log \hat{S}_n$; in [Rei93], after integrating analytically the underlying SDE, the change of measure consists in adding a drift term and scaling the variance of the underlying Brownian Motion $W_T$.

Changing the drift of a normally distributed random variable falls into the so-called Exponential Tilting approach (see [Gla04], page 260 for reference).

### 2.2.1 Our approach

Our first approach consisted in applying Importance Sampling to the Brownian Increments. Concretely, this consists in substituting $\Delta W_n$ in the Euler-Maruyama

updating of $\hat{S}_n$ with:

$$\Delta W_n^{IS} \;=\; \tilde{\mu}_n\, h \;+\; \tilde{\sigma}\Delta W_n \tag{2.9}$$

From now onwards, we will use the notations $\Delta W^{IS}$ and $\Delta W$ to specify what measure the Brownian increments have been drawn from. The resulting Radon-Nikodym derivative is:

$$RN \;=\; \prod_{n=1}^{T/h} RN_n \;=\; \prod_{n=1}^{T/h} \frac{q\left(\Delta W_n^{IS}\right)}{s\left(\Delta W_n^{IS}\right)}, \quad \text{where:} \tag{2.10}$$

$$q(x) \;=\; \frac{1}{\sqrt{2\pi h}} \exp\left\{-\frac{x^2}{2h}\right\} \quad \text{and} \quad s(x) \;=\; \frac{1}{\sqrt{2\pi\tilde{\sigma}^2 h}} \exp\left\{-\frac{(x-\tilde{\mu}_n h)^2}{2\tilde{\sigma}^2\, h}\right\} \tag{2.11}$$

Therefore:

$$RN \;=\; \tilde{\sigma}^{T/h} \exp\left\{\sum_{n=1}^{T/h}\left(-\frac{\left(\Delta W_n^{IS}\right)^2}{2h} + \frac{\left(\Delta W_n^{IS} - \tilde{\mu}_n h\right)^2}{2\tilde{\sigma}^2\, h}\right)\right\} \tag{2.12}$$

Therefore, our Monte Carlo estimator takes the form:

$$\hat{V}^{IS} \;=\; \frac{1}{N}\sum_{i=1}^{N}\left\{\hat{P}\left(\left\{\Delta W^{IS}\right\}_{n\,=\,1,\ldots T/h}^{(i)}\right)\, RN^{(i)}\right\} \tag{2.13}$$

Throughout the rest of the dissertation, unless differently specified, we will use the following expression for the time-varying Importance Sampling drift:

$$\tilde{\mu}_n \;=\; \mu_0 \;+\; \mu_1\left(\frac{n}{M}\right) \;+\; \mu_2\left(\frac{n}{M}\right)^2 \tag{2.14}$$

This choice restricts the degrees of freedom of the drift term compared to the papers we have reviewed, where each time step was allowed to have a different value for the drift. However, our choice has the advantage of keeping the dimension of the Importance Sampling parameters state space constant, regardless of M.

In the case of Euler-Maruyama time-stepping, the distribution of $\hat{S}_n$ conditional on $\hat{S}_{n-1}$ is known. Therefore, it is possible to apply Importance Sampling to the path $\{\hat{S}_n\}$ by changing the values of the drift and volatility parameters and making use of the theory in section 2.1.1 to compute the Radon-Nikodym derivative.

### 2.2.2 Issue with changing variance

In the first simulations, we have applied a constant change of drift ($\mu_1 = \mu_2 = 0$ in equation (2.14)) and variance to the Brownian increments $\Delta W$ in the case of a Digital Put. Any changes in the drift did not alter the order of weak convergence

(hence an indication of unbiasedness) and yielded significant variance reductions when the drift was chosen to be negative. However, changing the variance of the Brownian Increments produced unsatisfying results: specifically, the variance of the Monte Carlo estimator diverged as the width of the timestep was reduced, suggesting some issues arise when taking the limit to continuous time. A more detailed analysis follows.

The SDE (1.5) allows for a closed formula analytic solution:

$$S_T \;=\; s\,\exp\left\{\left(r-\frac{\sigma^2}{2}\right)T+\sigma W_T^{\mathbb{Q}}\right\} \tag{2.15}$$

In continuous time, the effect of applying the importance sampling construction described at the beginning of this section is to replace $W_T$ with $\tilde{W}_T = \tilde{\mu}T + \tilde{\sigma}W_T$. Therefore, the Importance Sampling estimator for $S_T$ is:

$$\frac{S_T^{IS}}{s}\,\frac{s(\tilde{W}_T)}{q(\tilde{W}_T)} = \tilde{\sigma}\exp\left\{\left(r-\frac{\sigma^2}{2}\right)T+\sigma\left(\tilde{\mu}T+\tilde{\sigma}W_T\right)-\frac{\left(\tilde{\mu}T+\tilde{\sigma}W_T\right)^2}{2T}+\frac{\left(\tilde{\sigma}W_T\right)^2}{2T\tilde{\sigma}^2}\right\} \tag{2.16}$$

$$\log\left\{\frac{S_T^{IS}}{s}\,\frac{s(\tilde{W}_T)}{q(\tilde{W}_T)}\right\} = \log\tilde{\sigma}+\left(r-\frac{\sigma^2}{2}\right)T+\sigma\left(\tilde{\mu}T+\tilde{\sigma}W_T\right)-\frac{\left(\tilde{\mu}T+\tilde{\sigma}W_T\right)^2}{2T}+\frac{\left(\tilde{\sigma}W_T\right)^2}{2T\tilde{\sigma}^2} \tag{2.17}$$

Considering the Euler-Maruyama approximation and taking the logarithm of the corresponding discretised quantity $\hat{S}_M^{IS}RN$ (as defined in (2.13)) yields:

$$\begin{aligned}\log\left\{\frac{\hat{S}_M^{IS}}{s}RN\right\} \;=\; &\sum_{n=1}^{T/h}\log\left(1+rh+\sigma\left(\tilde{\mu}h+\tilde{\sigma}\Delta W_n\right)\right)\;+\\ &T/h\log\tilde{\sigma}\;+\;\sum_{n=1}^{T/h}\left\{-\frac{\left(\tilde{\mu}h+\tilde{\sigma}\Delta W_n\right)^2}{2h}+\frac{\left(\tilde{\sigma}\Delta W_n\right)^2}{2h\tilde{\sigma}^2}\right\}\end{aligned} \tag{2.18}$$

Applying Ito's lemma to the logarithm in equation (2.18) yields:

$$\begin{aligned}\log\left\{\frac{\hat{S}_M^{IS}}{s}RN\right\} \;=\; &\left(r-\frac{\sigma^2}{2}\right)T+\sigma\left(\tilde{\mu}T+\tilde{\sigma}W_T\right)\;+T/h\,o(h^{3/2})+\\ &T/h\log\tilde{\sigma}\;+\;\sum_{n=1}^{T/h}\left\{-\frac{\tilde{\mu}^2h}{2}-\tilde{\mu}\tilde{\sigma}\Delta W_n+\frac{1-\tilde{\sigma}^2}{2h}\Delta W_n^2\right\}\end{aligned} \tag{2.19}$$

As $h$ tends to zero, the last term in the summation in equation (2.18) diverges to $+\infty$ with probability 1 (in fact, the Brownian motion $W_t$ has finite quadratic variation, which is divided by the infinitesimal quantity $h$). On the other hand, setting $\tilde{\sigma} = 1$, which implies a simple change of drift, the convergence of the quantity in (2.19) to (2.16) is restored.

### 2.2.3 Brownian Bridge construction

So far, we have simply converted the normal random vector $Z$ into $\Delta W$ as $\Delta W_n = \sqrt{h} Z_n$. In order not to exclude the possibility of a change in variance for the Brownian increments, we will adopt a different approach for the conversion $Z \to \Delta W^{IS}$, based on the theory of the Brownian Bridge. More specifically, we will be adapting a technique which is more commonly used in quasi-Monte Carlo methods (please refer to sections 3.1 and 5.3 of [Gla04] for further detail).

It is known that, for a Brownian Motion $W(t)$,

$$W\left(\frac{t_{i+1}+t_i}{2}\right) \,\Big|\, W(t_i)=w_i, W(t_{i+1})=w_{i+1} \;\sim\; \mathcal{N}\left(\frac{w_i+w_{i+1}}{2}, \frac{t_{i+1}-t_i}{4}\right) \tag{2.20}$$

which has the same distribution as:

$$W\left(\frac{t_{i+1}+t_i}{2}\right) \;=\; \frac{w_i+w_{i+1}}{2} + \sqrt{\frac{t_{i+1}-t_i}{4}}\, Z\,, \qquad Z \sim \mathcal{N}(0,1) \tag{2.21}$$

This piece of theory suggests the following construction of the set $\Delta W$. After computing

$$\begin{aligned} W_M \;&=\; \sqrt{T}\, Z_1 \\ W_{M/2} \;&= 1/2\, W_M + \sqrt{T/4}\, Z_2 \\ W_{3/4M} \;&= 1/2\, \left(W_M + W_{M/2}\right) + \sqrt{T/8}\, Z_3 \\ &\dots \end{aligned} \tag{2.22}$$

we simply take differences to obtain the Brownian increments. It is possible to operate a change in variance by scaling the first normal random variable $Z_1$, which is converted into $W_M$, to $\tilde{\sigma} Z_1$. The Radon-Nikodym derivative needs to be adapted as follows. Equation (2.13) defined the Monte Carlo estimator for:

$$\mathbb{E}_s\left[\hat{P}\left(\Delta W^{IS}\right) RN\left(\Delta W^{IS}\right)\right] \;\triangleq\; \mathbb{E}_s\left[\hat{R}\left(\Delta W^{IS}\right)\right] \tag{2.23}$$

Since $\Delta W^{IS}$ is a function of $Z$, we can see (2.23) as:

$$\int_{\mathbb{R}^M} (\hat{R} \circ \Delta W^{IS})(Z)\, \phi(Z)\, dZ \tag{2.24}$$

Therefore, denoting by $\tilde{\phi}$ the density of a $\mathbb{R}^M$-dimensional set of independent standard normal variables (apart from the first which has variance $\tilde{\sigma}$), we obtain:

$$\int_{\mathbb{R}^M} (\hat{R} \circ \Delta W^{IS})(Z)\, \phi(Z)\, dZ = \int_{\mathbb{R}^M} \left((\hat{R} \circ \Delta W^{IS})(Z)\, \frac{\phi(Z)}{\tilde{\phi}(Z)}\right) \tilde{\phi}(Z)\, dZ \tag{2.25}$$

This translates into multiplying the Monte Carlo estimator in (2.13) by:

$$\frac{\phi(Z)}{\tilde{\phi}(Z)} \;=\; |\tilde{\sigma}|\, \exp\left\{(1-\tilde{\sigma}^2)\frac{Z_1^2}{2}\right\} \tag{2.26}$$

## 2.3 Simulation for a Digital Put

In the following section and throughout the rest of the dissertation we will set $T = 1$, $r = 0.05$ and $\sigma = 0.25$. Each simulation in this section will include two stages. Firstly, we simulate a set of $N_{opt}$ paths in order to run an optimisation algorithm (whose features will be described in chapter 3), which returns the near-optimal values for $(\mu_0, \mu_1, \mu_2, \tilde{\sigma})$. Secondly, we simulate $N$ paths under the Original and the Importance Sampling measures and we compute the corresponding Monte Carlo estimators. In order to avoid introducing bias in the second step, the samples used in the first step are discarded. Throughout this section we will only make use of the Euler-Maruyama approximation.

Two key quantities will be used to compare the performance of the two methods. VR is defined as the ratio of the Original measure estimator over the variance of the Importance Sampling estimator. The empirical variance will be approximated as follows:

$$\nu \approx \frac{1}{N-1}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\hat{P}^{(i)}\right)^2 - \hat{V}^2\right) \tag{2.27}$$

Please note that $\hat{P}^{(i)}$ refers to a single Monte Carlo sample in general: therefore, in the Importance Sampling framework, by $\hat{P}^{(i)}$ we mean the product of the payoff and the Radon-Nikodym derivative ($\hat{R}$ in the notation of the previous section).

The second quantity is the Weak Error (as defined in (1.10)), which allows us to check the unbiasedness of the Importance Sampling estimator.

$N_{opt}$ needs to be calibrated beforehand, implying a trade-off between the computational cost of the overhead optimisation algorithm (roughly $O(N_{opt})$) and the additional computational cost in the actual Monte Carlo simulation in order to achieve the same accuracy of an Importance Sampling simulation with optimal parameters. The following table shows the effects of a change in $N_{opt}$ on the following Monte Carlo simulation. For this simulation, we have set $S_0 = 200$, $K = 100$ and $M = 16$.

| $N_{opt}$ | $N_{steps}$ | R.T. | Rel.R.T. | $\mu_0$ | $\mu_1$ | $\mu_2$ | $\tilde{\sigma}$ | VR | $V$ | $\hat{V}^{IS}$ | $\hat{V}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $10^5$ | 5 | 44.7 s | 100% | -2.86 | -0.46 | 0.08 | 0.50 | 604 | 0.0021 | 0.0025 | 0.0026 |
| $10^4$ | 5 | 5.4 s | 12.1% | -2.88 | -0.41 | 0.11 | 0.50 | 612 | 0.0021 | 0.0025 | 0.0026 |
| 5000 | 7 | 1.9 s | 4.3% | -2.82 | -0.47 | 0.06 | 0.50 | 596 | 0.0021 | 0.0025 | 0.0026 |
| $10^{3}$ * | 9 | 0.5 s | 1.1% | -2.53 | -0.87 | -0.45 | 0.51 | 404 | 0.0021 | 0.0025 | 0.0026 |

*: The optimisation algorithm might be unstable: executing it several times, we have observed that it fails to converge in about 1 case out of 10.

A heuristic observation suggests that $N_{opt} = 5000$ is a fair compromise between the overhead computational cost and effectiveness in variance reduction. From now onwards, unless differently specified, we will employ 5000 paths in the optimisation algorithm.

In order to show the outstanding potential of Importance Sampling in terms of variance reduction, we have performed a full simulation (i.e. Optimisation + Actual Simulation) for different combinations of $S_0$ and $K$. The more the option is out-of-the-money, the more Importance Sampling turns out to be beneficial. In the following tables we have reported the values of the optimal Importance Sampling parameters, the variance ratio, the estimated values (exact, Original Measure and Importance Sampling), the Monte Carlo error of the Importance Sampling estimator and the number $N_{itm}$ of paths ending in the money under the two different measures.

Note: even when $N_{itm} = 0$, $\hat{V}$ will not be exactly zero, due to the payoff smoothing described in (1.18). It is worth noticing that $\mu_1$ and $\mu_2$ seem to give a minor contribution. Since we are dealing with a European non path-dependent option, this is unsurprising. In chapter 5 we will consider possible economisations of the model.

| $S_0$ | $K$ | N | M | $\mu_0$ | $\mu_1$ | $\mu_2$ | $\tilde{\sigma}$ | VR | $3 \cdot sd^{IS}$ | $3 \cdot sd$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 50 | $10^6$ | 16 | -2.62 | -0.68 | -0.19 | 0.51 | 519 | $1.17 \cdot 10^{-6}$ | $2.67 \cdot 10^{-5}$ |
| 125 | 50 | $10^6$ | 16 | -3.56 | -0.59 | 0.04 | 0.51 | 7654 | $2.72 \cdot 10^{-7}$ | $2.38 \cdot 10^{-5}$ |
| 150$^\star$ | 50 | $10^6$ | 16 | -4.00 | -0.82 | -0.12 | 0.48 | 152249 | $1.72 \cdot 10^{-8}$ | $6.71 \cdot 10^{-6}$ |
| 175$^\star$ | 50 | $10^6$ | 16 | -5.09 | 0.04 | 0.03 | 0.43 | 2538625 | $1.04 \cdot 10^{-9}$ | $1.66 \cdot 10^{-6}$ |
| 200$^\star$ | 50 | $10^6$ | 16 | -5.56 | 0.07 | 0.04 | 0.44 | 708932 | $7.95 \cdot 10^{-11}$ | $6.69 \cdot 10^{-8}$ |

| $S_0$ | $K$ | $V$ | $\hat{V}$ | $\hat{V}^{IS}$ | $N_{itm}$ | $N_{itm}^{IS}$ |
|---|---|---|---|---|---|---|
| 100 | 50 | 0.0021 | 0.0026 | 0.0025 | 2676 | 694919 |
| 125 | 50 | $8.75 \cdot 10^{-5}$ | $1.35 \cdot 10^{-4}$ | $1.33 \cdot 10^{-4}$ | 141 | 663217 |
| 150$^\star$ | 50 | $3.73 \cdot 10^{-6}$ | $1.01 \cdot 10^{-5}$ | $7.55 \cdot 10^{-6}$ | 12 | 633977 |
| 175$^\star$ | 50 | $1.74 \cdot 10^{-7}$ | $9.60 \cdot 10^{-7}$ | $4.87 \cdot 10^{-7}$ | 1 | 647883 |
| 200$^\star$ | 50 | $9.07 \cdot 10^{-9}$ | $2.80 \cdot 10^{-8}$ | $3.60 \cdot 10^{-8}$ | 0 | 624154 |

$^\star$ : Please refer to section 3.2.5 for clarification

The table in the previous page arouses a question worth being clarified. In section 2, we have stated that the Importance Sampling estimator is unbiased. However, we have also seen (section 1.2) that a discrete time approximation is unavoidably affected by Weak Error. According to the theory developed in section 2, we should expect this quantity to be the same both for $\hat{V}$ and $\hat{V}^{IS}$. However, we are not provided with the exact value for the Weak Error, and its estimator $|\hat{V} - V|$ is intrinsically affected by statistical error. Therefore, as long as the difference between the two estimated weak errors is smaller than $3 \cdot sd$, this discrepancy should not raise concerns about unbiasedness. A simple argument to support this assertion is that, in the case of $S_0 = 200$, $K = 50$, the estimated price without any payoff smoothing is $3.58 \cdot 10^{-8}$ and 0 with and without Importance Sampling respectively. Without Importance Sampling, the estimated Weak Error is simply the exact value $V$. Since Importance Sampling provides an estimate of the price different than 0, we should not expect the two estimators to have the same estimated Weak Error.

An interesting fact that we have observed is that performing the preliminary optimisation on a coarser time-stepping, we obtain Importance Sampling parameters that produce satisfactory results even with finer time grids. Figure 2.1 shows the dependence of the Weak Error and the Monte Carlo error on the length of the time-step $h$ with and without Importance Sampling. Optimisation has been carried out with $M = 4$, whereas the actual simulation was repeated for $M = 2, 4, 8, 16, 32, 64$.

Incidentally, figure 2.1 shows that the Weak Error of the Importance Sampling estimator has indeed $O(h)$ magnitude.

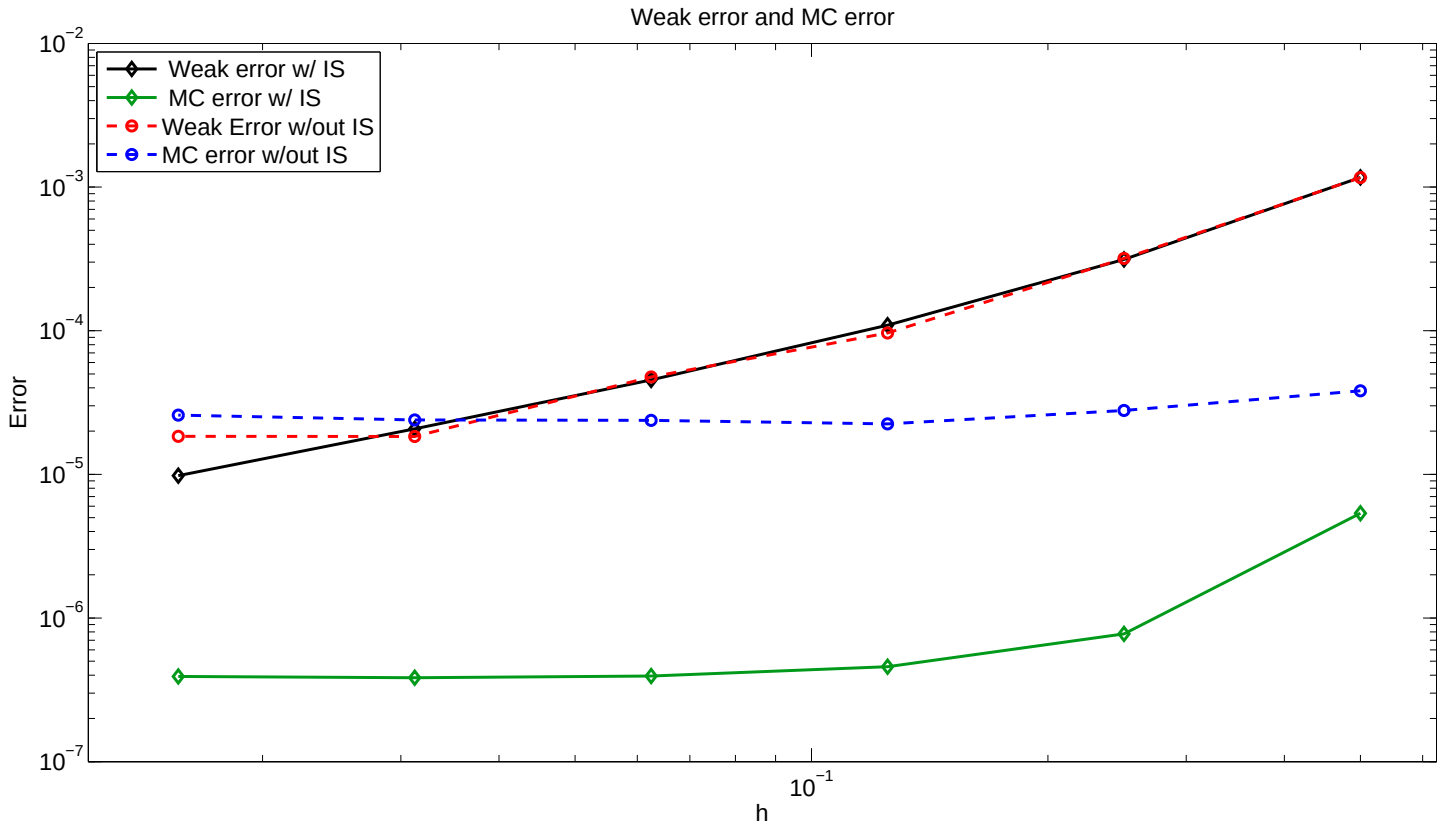


Figure 2.1: Plot of the Weak Error and the Monte Carlo Error $(3 \cdot sd)$for a Digital Put, with and without Importance Sampling. $S_0 = 125$, $K = 50$. $S_0 = 200, K = 100$. M = 2, 4, 8, 16, 32, 64 from right to left.

To provide a stronger argument in favour of the unbiasedness of our Importance Sampling estimator, we have performed 20 simulations, computed the payoff using analytic integration of the underlying SDE (namely, setting $\hat{S}_M = S_0 \exp\{(r - 1/2\,\sigma^2)T + \sigma \sum_n \Delta W_n\}$) with and without Importance Sampling and computed the Relative Root Mean Square Error (see section 1.2.1 for definitions) of the estimated price compared to the analytic price. These are the results:

| $RMSE[\hat{V}]$ | $RMSE[\hat{V}]/V$ | $RMSE[\hat{V}^{IS}]$ | $RMSE[\hat{V}^{IS}]/V$ |
|---|---|---|---|
| $2.11 \cdot 10^{-7}$ | 23.2 | $1.28 \cdot 10^{-11}$ | 0.0014 |

We conclude this section with a plot of the empiric distribution functions of $S_T$ under three different measures: the Optimal measure (as defined in (2.6)), the Importance Sampling measure and the Original Measure. As expected, the Importance Sampling density is as close to the Optimal density as its lognormal shape allows for.

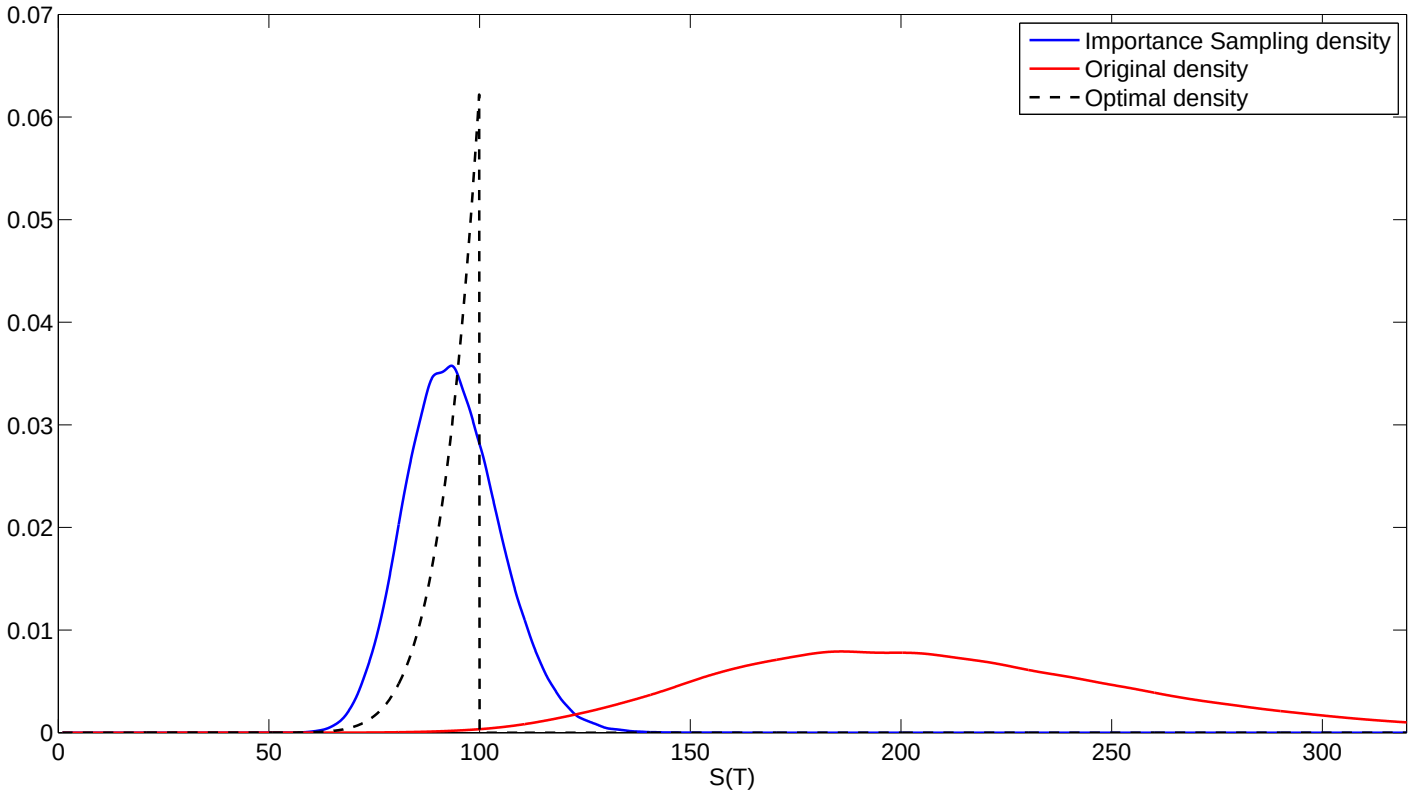


Figure 2.2: Plot of the empirical density functions (produced using the `ksdensity` kernel smoothing Matlab function) of $S_T$ under the Importance Sampling, the Original and the Optimal measures in the case of a Digital Put. $S_0 = 200$, $K = 100$.

## 2.4 Simulation for a Down-and-in Barrier Call Option

In this section we will show the results we have obtained applying Importance Sampling for pricing a Down-and-In Call. We have made use of the Milstein time-scheme on a 32-step time-grid.

As expected, since we are dealing with a path-dependent option, the parameters $\mu_1$, $\mu_2$ appear to be relevant in defining the shape of the drift term throughout time (see figure 2.4).

As in the previous section, we have reported some key quantities in a table. The difference between WE and $\text{WE}^{IS}$, which we have discussed about in the previous section, is well within $3 \cdot sd$, hence does not raise any concerns. As expected, the beneficial effects of Importance Sampling in terms of variance reduction become more evident when the probability of a payout is lower.

| $S_0$ | $K$ | B | $\mu_0$ | $\mu_1$ | $\mu_2$ | $\tilde{\sigma}$ | VR | $3 \cdot sd$ | $3 \cdot sd^{IS}$ |
|---|---|---|---|---|---|---|---|---|---|
| 100 | 100 | 80 | -2.32 | 8.63 | -4.05 | 0.60 | 9.3 | $7.9 \cdot 10^{-3}$ | $2.6 \cdot 10^{-3}$ |
| 100 | 100 | 70 | -3.40 | 9.47 | -2.38 | 0.52 | 47.6 | $1.39 \cdot 10^{-3}$ | $2.01 \cdot 10^{-4}$ |
| $^\star$100 | 100 | 60 | -6.24 | 15.77 | -4.34 | 0.29 | 299 | $6.40 \cdot 10^{-5}$ | $3.70 \cdot 10^{-6}$ |
| 100 | 110 | 90 | -1.18 | 8.62 | -5.82 | 0.58 | 7.1 | 0.0185 | 0.00695 |
| 100 | 110 | 80 | -3.06 | 12.93 | -7.39 | 0.48 | 18.7 | $5.0 \cdot 10^{-3}$ | $1.17 \cdot 10^{-3}$ |

| $S_0$ | $K$ | $B$ | $V$ | $\hat{V}$ | $\hat{V}^{IS}$ | WE | WE$^{IS}$ | $N_{itm}$ | $N_{itm}^{IS}$ |
|---|---|---|---|---|---|---|---|---|---|
| 100 | 100 | 80 | 0.4083 | 0.4169 | 0.4159 | $8.56 \cdot 10^{-3}$ | $7.59 \cdot 10^{-3}$ | 24213 | 221451 |
| 100 | 100 | 70 | 0.01689 | 0.01780 | 0.01709 | $9.10 \cdot 10^{-4}$ | $2.05 \cdot 10^{-4}$ | 1157 | 102537 |
| $^\star$100 | 100 | 60 | $1.26 \cdot 10^{-4}$ | $8.20 \cdot 10^{-5}$ | $1.29 \cdot 10^{-4}$ | $4.35 \cdot 10^{-5}$ | $3.20 \cdot 10^{-6}$ | 7 | 103241 |
| 100 | 110 | 90 | 1.7103 | 1.7265 | 1.7337 | 0.0162 | 0.0234 | 80657 | 373853 |
| 100 | 110 | 80 | 1.652 | 1.678 | 1.681 | $2.66 \cdot 10^{-3}$ | $2.97 \cdot 10^{-3}$ | 9135 | 236987 |

$^\star$ : Please refer to section 3.2.5 for clarification

We conclude this section plotting the solution $S^{det}(t)$ of the ODE: $dS^{det}(t) = (\mu_0 + r + \mu_1 t + \mu_2 t^2)\, dt$, which defines the evolution that $S(t)$ would follow in a fully deterministic case (namely, if the underlying Brownian motion was set to 0).

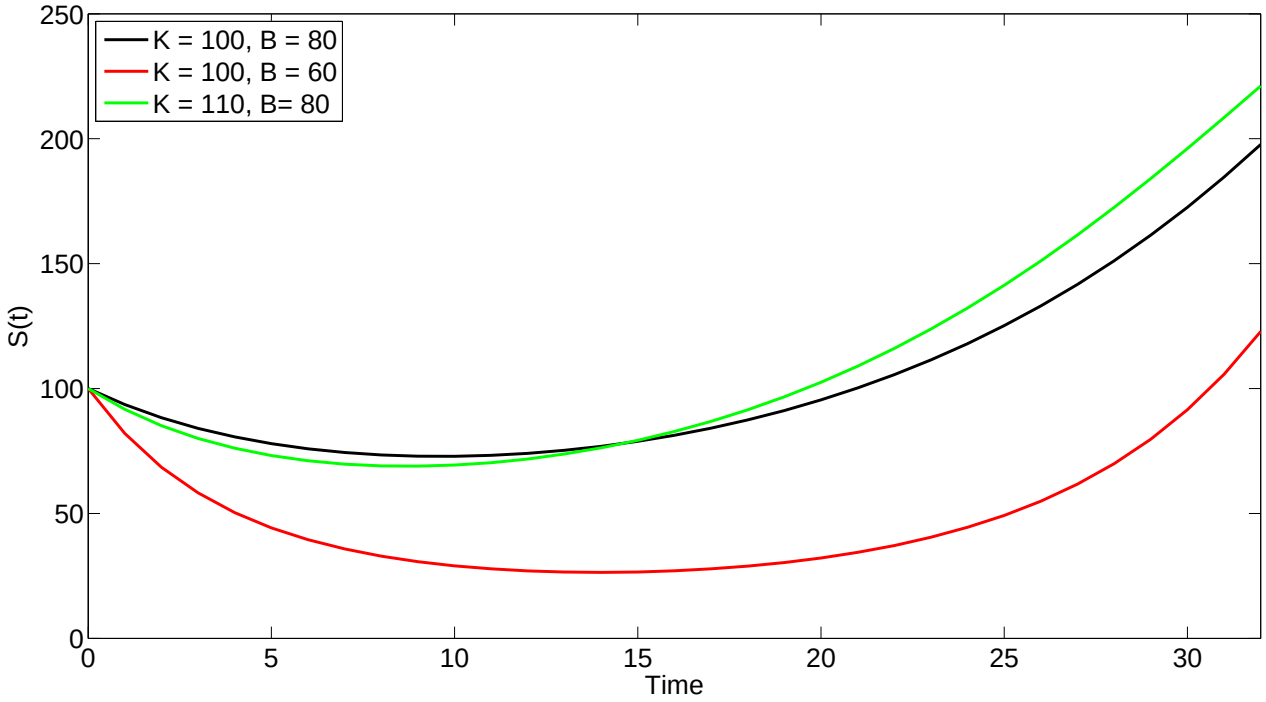


Figure 2.3: Plot of $S^{det}(t)$ for 3 different combinations of K and B.

# Chapter 3

# Optimisation techniques

In the following section, we will explain the Optimisation Algorithm that has been employed to produce the optimal values of the Importance Sampling parameters. The main references for this topic are [NW99] and [BGLS06].

We have decided to implement our own Optimisation algorithm, rather than relying on built-in packages, in order to have full control on it. Despite being suboptimal, it does not prevent us from evincing the relation between its runtime and $N_{opt}$ in relative terms. In fact, if we assume that a more efficient algorithm reduces the number of iterations and the number of function and gradient calls, still the above mentioned dependence lies in the estimation time of the objective function and its gradient.

As in [GHS99], we opted for a fully deterministic method (in the paper, the authors make use of a large deviations principle to derive the nonlinear equation governing the optimal parameters and they solve it with a fully deterministic fixed-point iteration method). Specifically, we will generate only one set of $N_{opt}$ sample paths at the beginning of the algorithm and we will use it for all estimations of the objective function or its gradient. In fact, we are interested in the dependence of the objective function on the Importance Sampling parameters and repeating the generation of random numbers at each function/gradient evaluation would affect their smoothness.

At the current state of art, other commonly used techniques are gradient-based Robbins-Monro stochastic optimisation (as in [Aro03, SF00, VAD98]) and non-linear least-squares minimisation (as in [Cap08]).

## 3.1 Objective function

The quantity we are set to minimise is the one defined in (2.5):

$$\mathbb{V}_s\left[\hat{P}\left(\Delta W^{IS}\right)\ RN\right] = \mathbb{E}_s\left[\left(\hat{P}\left(\Delta W^{IS}\right)\ RN\right)^2\right] - \left(\mathbb{E}_s\left[\hat{P}\left(\Delta W^{IS}\right)\ RN\right]\right)^2 \quad (3.1)$$

where $\Delta W^{IS}$ denotes the whole set of Brownian increments $\left\{\Delta W^{IS}\right\}_{n=1,\dots T/h}$. Since the Importance Sampling estimator is unbiased, $\mathbb{E}_s\left[\hat{P}\left(\Delta W^{IS}\right)\ RN\right]$ is constant for all choices of the Importance Sampling parameters. We will then focus on minimising the following objective function:

$$\varphi_s(x) \triangleq \mathbb{E}_s\left[\left(\hat{P}\left(\Delta W^{IS}(x)\right)\ RN(x)\right)^2\right] \tag{3.2}$$

where $x = (\mu_0, \mu_1, \mu_2, \tilde{\sigma})$.

Alternatively, we could minimise $\varphi_p(x) \triangleq \mathbb{E}_p\left[\hat{P}^2\left(\Delta W\right)\ RN(x)\right]$, since:

$$\int_{\mathbb{R}^{T/h}} \hat{P}^2\left(w\right) \frac{q^2(w)}{s^2(w,x)} s(w,x)\, dw \ = \ \int_{\mathbb{R}^{T/h}} \hat{P}^2\left(w\right) \frac{q(w)}{s(w,x)} q(w)\, dw \tag{3.3}$$

where $s(\cdot, x)$ denotes the Importance Sampling density function $s(\cdot)$ as defined in (2.11) (with the addition of the term deriving from the Brownian Bridge construction in (2.26)) when the set of Importance Sampling parameters take value $x$.

As pointed out in section 2 of [SF00] the choice between one formulation or the other has a remarkable consequence: opting for the latter, the dependence of $\varphi_p$ on $x$ comes in only through the Radon-Nikodym derivative, since $\Delta W$ is sampled from the Original measure. Since the optimisation algorithm will imply differentiating $\varphi_{(\cdot)}$ w.r.t. x, this formulation allows for non-differentiable payoffs. However, both $\varphi_s$ and $\varphi_p$ will be approximated via Monte Carlo simulation, and will therefore be affected by the same statistical error we are attempting to minimise by sampling from $s$ rather than $q$:

$$\hat{\varphi}_s(x) \ = \ \frac{1}{N_{opt}} \sum_{i=1}^{N_{opt}} \left[\left(\hat{P}\left(\Delta W^{IS\,(i)}(x)\right)\ RN^{(i)}(x)\right)^2\right] \tag{3.4}$$

Since we will deal with differentiable payoffs, we will opt for $\hat{\varphi}_s(x)$, which we will denote simply as $\varphi(x)$. This choice allows us to avoid the risk that, sampling under the Original measure, too few paths end in-the-money.

### 3.1.1 Potential improvement

We will expose, only at a theoretical level, a third possible way to estimate the objective function. Due to time constraints, we have not managed to assess its performances with numerical experiments.

Since our optimisation algorithm is iterative, we expect it to generate a sequence $\{x_n\}$ before reaching the near-optimal point. Therefore, at the $n^{th}$ iteration, $x_n$ can

be considered as a constant and $\varphi$ is a function of $x_{n+1}$. Taking one step back, let us recall what $\varphi = \hat{\varphi}_s$ is estimating:

$$\varphi_s(x_{n+1}) \;=\; \int_{\mathbb{R}^{T/h}} \hat{P}^2\,(w)\,\frac{q^2(w)}{s^2(w, x_{n+1})}\,s(w, x_{n+1})\,dw \tag{3.5}$$

We can formulate the integral in (3.5) alternatively as:

$$\int_{\mathbb{R}^{T/h}} \hat{P}^2\,(w)\,\frac{q^2(w)}{s^2(w, x_{n+1})}\,\frac{s(w, x_{n+1})}{s(w, x_n)}\,s(w, x_n)\,dw \tag{3.6}$$

Therefore, applying a reasoning similar to what we have shown in (2.25) and (2.26):

$$\varphi_s(x_{n+1}) \;=\; \mathbb{E}_{s^\star}\left[\hat{P}^2\left(\Delta W^{IS^\star}\right)\frac{q^2(\Delta W^{IS^\star})}{s\left(\Delta W^{IS^\star}, x_{n+1}\right)\,s^\star\left(\Delta W^{IS^\star}\right)}\right] \triangleq\; \varphi_{s^\star}(x_{n+1}|x_n) \tag{3.7}$$

where $s^\star(\cdot) \triangleq s(\cdot, x_n)$. This naturally leads to the estimator:

$$\hat{\varphi}_{s^\star}(x_{n+1}|x_n) = \frac{1}{N_{opt}}\sum_{i=1}^{N_{opt}}\left[\hat{P}^2\left(\Delta W^{IS^\star\,(i)}\right)\frac{q^2(\Delta W^{IS^\star\,(i)})}{s\left(\Delta W^{IS^\star\,(i)}, x_{n+1}\right)\,s^\star\left(\Delta W^{IS^\star\,(i)}\right)}\right] \tag{3.8}$$

where each $\Delta W^{IS^\star\,(i)} \sim s^\star$.

The dependence of $\hat{\varphi}(\cdot|x_n)$ on $x_{n+1}$ is entirely encapsulated in $s\left(\Delta W^{IS^\star\,(i)}, x_{n+1}\right)$, meaning that any differentiation with respect to $x_{n+1}$ will not involve differentiating the payoff. At the same time, sampling from $s(\cdot, x_n)$ reduces the statistical error of $\hat{\varphi}_{(\cdot)}$ compared to sampling from $q(\cdot)$.

## 3.2 The algorithm

We will denote the descent direction at iteration $n$ as $p_n$ and the length of the corresponding step $\alpha_n$. Therefore, we will update $x_n$ as: $x_{n+1} = x_n + \alpha_n\,p_n$. Moreover, we will set $q_n(\alpha) \triangleq \varphi(x_n + \alpha p_n)$, hence: $q_n'(\alpha) \;=\; \nabla\varphi(x_n + \alpha p_n)^T p_n$.

The extremely simplistic optimisation algorithm consists in the following procedure:

**Algorithm 1** Gradient-based optimisation

```
Compute steepest-descent direction
Perform Line Search
Update x
while not converged do
    Compute steepest-descent AND BFGS directions
    Perform Line Search for both directions
    Choose the direction that most minimises φ
    Update x
end while
```

### 3.2.1 Descent directions

The steepest-descent direction simply consists in: $p_n = -\nabla\varphi(x_n) \,/\, \|\nabla\varphi(x_n)\|$.

The BFGS method falls into the category of quasi-Newton methods. The descent direction is defined by $p_n = -B_n\nabla\varphi(x_n) \,/\, \|B_n\nabla\varphi(x_n)\|$, where $B_n$ is an approximation of the inverse of the Hessian matrix, such that it meets the so-called secant condition:

$$B_n\, y_n \;=\; s_n \qquad s_n = x_n - x_{n-1}, \quad y_n = \nabla\varphi(x_n) - \nabla\varphi(x_{n-1}) \tag{3.9}$$

The sequence $B_n$ is updated via rank-1 matrices as follows:

$$B_{n+1} \;=\; \left(\mathbb{1} - \frac{s_n y_n^T}{y_n^T s_n}\right) B_n \left(\mathbb{1} - \frac{y_n s_n^T}{y_n^T s_n}\right) \;+\; \frac{s_n s_n^T}{y_n^T s_n} \tag{3.10}$$

Since we have little (if any) knowledge about the Hessian matrix of $\varphi$, we will simply set $B_0 = \mathbb{1}$. Although steepest-descent is well known for being slow in a neighbourhood of the optimal point, it often outperforms BFGS when $x_n$ is far from the optimal point. Moreover, it is the only viable technique for the very first iteration.

### 3.2.2 Complex-step differentation

The previous sections imply the estimation of $\nabla\varphi(x)$, which has no closed formula. We will adopt the so-called Complex-Step differentiation (or Complex Variable Trick) technique. We have referred to [MSA03, ST98] for practical implementation and underlying theory.

$\varphi(x)$ can be extended to an analytic complex function $\varphi(x + iy)$, for which the following Taylor expansion holds:

$$\varphi(x + i\,\varepsilon) \;=\; \varphi(x) + i\,\varepsilon^T\,\nabla\varphi(x) - \frac{1}{2}\,\varepsilon^T\,\nabla^2\varphi(x)\,\varepsilon + o(\|\varepsilon\|^2) \tag{3.11}$$

Therefore, we can approximate $\nabla\varphi(x)$ (up to a truncation error of order $O(\|\varepsilon\|^2)$) as follows:

$$(\nabla\varphi(x))_i \approx \frac{\text{imag}\left(\varphi(x + i\,\varepsilon\,\underline{e}_i)\right)}{\varepsilon} \tag{3.12}$$

The main advantage of CVT differentiation compared to Finite Differencing approximation is the possibility of reducing the step-size without the risk of subtractive error cancellations.

In order to show the validity of this method, we have approximated the MSE of the gradient computed with complex-step differentiation compared to the same calculation using finite differencing. We have set $\varepsilon = 10^{-10}$ and used the case of the Digital Put payoff to estimate:

$$\text{MSE}\left[\nabla^{CVT}\varphi(x)\right] \approx \frac{1}{100}\sum_{i=1}^{100} \| \nabla^{CVT}\varphi(x|\Delta W^{(i)}) - \nabla^{FD}\varphi(x|\Delta W^{(i)}) \|_2^2 \tag{3.13}$$

We have obtained the following results, for $N = 10^5$, $M = 16$, $S_0 = 100$, $K = 50$:

$$\begin{aligned} x = (0,0,0,1): &\qquad \text{MSE} = 7.42 \cdot 10^{-12} \\ x = (-2.6,-0.7,-0.2,0.5): &\qquad \text{MSE} = 1.19 \cdot 10^{-14} \end{aligned} \tag{3.14}$$

The only prescription in order to apply complex-step differentiation is to re-implement a number of functions that are not defined for complex numbers. We have had to redefine $\Phi(\cdot)$ and `max` as follows:

$$\begin{aligned} \texttt{max}(x,y) &= \texttt{max}(\,\text{Re}(x), \text{Re}(y)\,) + i\,\left(\text{Im}(x)\,\mathbb{1}_{\text{Re}(x)>\text{Re}(y)} + \text{Im}(y)\,\mathbb{1}_{\text{Re}(y)>\text{Re}(x)}\right) \\ \texttt{abs}(x) &= \texttt{abs}(\,\text{Re}(x)\,) + i\,(\text{Im}(x)\,\text{sign}(x)) \\ \Phi(x) &= \Phi(\text{Re}(x)) + i\,\text{Im}(x)\,\frac{1}{\sqrt{2\pi}} e^{-\frac{\text{Re}(x)^2}{2}} \end{aligned} \tag{3.15}$$

### 3.2.3 Line search

The Line Search procedure consists in finding the step length $\alpha_n$ minimising $q_n(\alpha)$. We will use the following inexact (and, in a sense, greedy) Line Search procedure. It consists in finding an interval $[\alpha_L; \alpha_R]$ where the optimal $\alpha$ lies and in progressively narrowing it, until the midpoint satisfies sufficient decrease conditions for $q_n$ (it would be useless computational expense to search for the exact optimal point). Our stopping criterion is based on the Wolfe conditions. Namely:

$$\begin{aligned} \text{Wolfe}_1(\alpha): &\quad q(\alpha) \le q(0) + m_1\alpha\, q'(0) \\ \text{Wolfe}_2(\alpha): &\quad q'(\alpha) \ge m_2\, q'(0) \end{aligned} \tag{3.16}$$

Generally, $m_1 \in [10^{-4}; 10^{-3}]$ and $m_2 \in [0.1; 0.9]$.

**Algorithm 2** Line Search

Set $\alpha_L = 0$
Make ansatz for $\alpha_R$
**while** $q_n'(\alpha_R) \leq 0$ **do**
  **if** $\text{Wolfe}_1(\alpha_R) = \text{TRUE}$ **AND** $\text{Wolfe}_2(\alpha_R) = \text{FALSE}$ **then**
    Set $\alpha_L = \alpha_R$
  **end if**
  Set $\alpha_R = 2\alpha_R$
**end while**
Set $\alpha = 0.5 \cdot (\alpha_L + \alpha_R)$
**while** $\text{Wolfe}_1(\alpha) = \text{FALSE}$ **OR** $\text{Wolfe}_2(\alpha) = \text{FALSE}$ **do**
  Set $\alpha = 0.5 \cdot (\alpha_L + \alpha_R)$
  **if** $\text{Wolfe}_1(\alpha) = \text{FALSE}$ **then**
    Set $\alpha_R = \alpha$
  **end if**
  **if** $\text{Wolfe}_1(\alpha_R) = \text{TRUE}$ **AND** $\text{Wolfe}_2(\alpha_R) = \text{FALSE}$ **then**
    Set $\alpha_L = \alpha$
  **end if**
**end while**

For further details and proofs of convergence, please refer to [BGLS06].

The Line Search algorithm is generally the most critical part of an optimisation procedure, and poor implementation can affect convergence speed as well as robustness. For example, introducing quadratic and cubic interpolation would be a way of reducing the number of function evaluations, hence increasing speed.

We have focused on ensuring that our method is robust, especially in the search for the right bound. Our first guess is for the value of $\alpha$ such that the linear expansion of $q$ around 0 decreases by $(\varphi(x_n) - \varphi(x_{n-1}))/\varphi(x_n)$. Less careful choices might have often produced dramatic effects: in fact, we have observed that, for certain values of $x_n$, $q_n$ shows an increasing trend up to a value $\bar{\alpha}$ and suddenly drops to 0 after $\bar{\alpha}$ (this happens e.g. when $\tilde{\sigma}$ reaches 0). In a similar scenario, if our initial guess for the right bound $\alpha_R$ had been greater than $\bar{\alpha}$, the Line Search algorithm would have never terminated.

### 3.2.4 Stopping criterion

Direct observation suggest that $\varphi$ is a convex function in a neighbourhood of the optimal point. Therefore, first order conditions are sufficient to ensure optimality. We have chosen a stopping criterion based on the norm of the gradient:

$$\|\nabla\varphi(x_N)\|_2 \leq \xi \|\nabla\varphi(x_0)\|_2 \tag{3.17}$$

We have decided to operate in relative terms since the magnitude of the gradient varies significantly according to the moneyness of the option. We have set $\xi = 0.5\%$ unless differently specified.

### 3.2.5 Initial Point

So far, we have dealt with descent directions, line search and stopping criteria, but we have not addressed the question of the initial point. A natural choice would be to set $x_0 = (0, 0, 0, 1)$, which corresponds to a simulation under the Original Measure. This choice proves to be robust until a certain degree of moneyness. In sections 2.3 and 2.4 we have marked with a $^\star$ some options which were excessively out of the money. For instance, for those combinations of $S_0$ and $K$, fewer than 1 path out of $10^5$ in the Monte Carlo simulation without Importance Sampling ended in-the-money in the Digital Put case. This means that setting $x_0 = (0, 0, 0, 1)$, it is extremely likely that none of the paths in the smaller Optimisation sample ends in-the-money. Since the Optimisation algorithm is conceived to simply minimise (3.2), regardless of Weak Error, MSE or other indicators of biasedness, the iterations will tend to proceed further out-of-the-money, where eventually $\mathbb{E}_s\left[P^2(W)\, q^2(W)/s^2(W)\right]$ reaches 0, to the price of $WE = V$. This hurdle can be overcome by making an educated guess about the initial point, in order to force more paths to end in-the-money. Following a heuristic argument, in the Digital Put case we have set $x_0 = (\mu_0^0, 0, 0, 1)$, where $\mu_0^0 = \left(\log\left(K/S_0\right) - rT\right)/\sigma$, such that:

$$\mathbb{E}^{\mathbb{Q}}\left[S_0 \exp\left\{\left(r - 1/2\sigma^2\right)T + \sigma(W_T + \mu_0^0\, T)\right\}\right] = K \tag{3.18}$$

Alternatively, it is possible to run the optimisation algorithm in a less extreme scenario and use its solution as an initial guess for the actual optimisation. This is the approach we have followed for the Down-and-In Call.

# Chapter 4

# Multilevel Monte Carlo

Multilevel Monte Carlo is an extension of the Control Variate technique which was introduced by M. Giles in 2008. It enables us to reduce remarkably the computational cost required to achieve a pre-determined accuracy. The underlying intuition is to simulate a consistent number of samples at lower accuracy levels, using then fewer paths at higher accuracy as a correction term. The main references for Multilevel Monte Carlo are [Gil08, Gil15b, Gil15a].

## 4.1 General Background

Let us consider the Monte Carlo estimator we have defined in (1.2). Each sample is estimated by discretising the underlying SDE, which implies choosing the number of time-steps $M$. We will make this dependence explicit by denoting as $\hat{P}_\ell$ a sample path approximated with $\ell$ time-steps.

Given a sequence $\hat{P}_1, \ldots, \hat{P}_{L-1}, \hat{P}_L$, it takes a simple application of telescopic sum and linearity of expectation properties to obtain the following identity:

$$\mathbb{E}[\hat{P}_L] \;=\; \mathbb{E}[\hat{P}_0] \;+\; \sum_{\ell=1}^{L} \mathbb{E}[\hat{P}_\ell - \hat{P}_{\ell-1}] \tag{4.1}$$

With the usual Monte Carlo approach, we can approximate the expectation in (4.1) as:

$$\hat{V}_L^{ML} \;=\; \frac{1}{N_0} \sum_{i=1}^{N_0} \hat{P}_0^{(i)} \;+\; \sum_{\ell=1}^{L} \left( \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \left( \hat{P}_\ell - \hat{P}_{\ell-1} \right)^{(i)} \right) \tag{4.2}$$

where the number of samples $N_\ell$ per each level is allowed to be different. The notation $\left( \hat{P}_\ell - \hat{P}_{\ell-1} \right)^{(i)}$ implies that both $\hat{P}_\ell$ and $\hat{P}_{\ell-1}$ are to be evaluated using the same random sample (nevertheless, different paths on the same or on different levels

are independent). We will assume that simulating a single path has different computational costs $C_\ell$, depending on the number of time-steps $\ell$ of the approximation. In our Euler-Maruyama framework, $C_\ell$ grows with $\ell$ (the order will be estimated later). Moreover, we will admit that each of the estimators $\hat{Y}_\ell \triangleq \left(\hat{P}_\ell - \hat{P}_{\ell-1}\right)$, $\hat{Y}_0 \triangleq \hat{P}_0$ have different variance $V_\ell$. Therefore the overall variance and computational cost of the estimator $\hat{V}_L^{ML}$ are:

$$\mathbb{V}\left[\hat{V}_L^{ML}\right] = \sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell}, \qquad \text{Cost}\left[\hat{V}_L^{ML}\right] = \sum_{\ell=0}^{L} C_\ell \, N_\ell \tag{4.3}$$

For a fixed computational cost, it is possible choose the $N_\ell$'s such that the overall variance of the MLMC estimator is minimised. Applying KKT conditions to:

$$\mathcal{L}(\{N_\ell\}, \lambda) = \sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell} + \lambda \left( \sum_{\ell=0}^{L} C_\ell \, N_\ell \right) \tag{4.4}$$

we obtain that $N_\ell = \lambda \sqrt{V_\ell / C_\ell}$. It is possible to choose the constant $\lambda$ such that the overall variance is equal to $\varepsilon^2$, leading to:

$$N_\ell = \left( \frac{\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell}}{\varepsilon^2} \right) \sqrt{\frac{V_\ell}{C_\ell}} \tag{4.5}$$

The overall computational cost is then: $\left(\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell}\right)^2 / \varepsilon^2$.

We will now state a key theorem for Multilevel Monte Carlo in a general stochastic framework (for reference, see Theorem 1 in section 2 of [Gil15b]).

**Complexity Theorem.** *Let $P$ be a random variable and let $P_\ell$ be its corresponding level $\ell$ numerical approximation. Let us suppose there exist independent estimators $\hat{Y}_\ell$ based on $N_\ell$ Monte Carlo samples, each with expected cost $C_\ell$ and variance $V_\ell$, and positive constants $\alpha$, $\beta$, $\gamma$, $c_1$, $c_2$, $c_3$ such that $\alpha \geq \frac{1}{2} min(\beta, \gamma)$ . Let us suppose in addition that the following hold:*

1. $\left| \mathbb{E}[P_\ell - P] \right| \leq c_1 \, 2^{-\alpha \ell}$
2. $\mathbb{E}[\hat{Y}_\ell] = \begin{cases} P_0 & \text{if } \ell = 0 \\ P_\ell - P_{\ell-1} & \text{if } \ell > 0 \end{cases}$
3. $V_\ell \leq c_2 \, 2^{-\beta \ell}$
4. $C_\ell \leq c_3 \, 2^{\gamma \ell}$

*Then, there exists a positive constant $c_4$ such that, $\forall \varepsilon < e^{-1}$ there are values $L$ and $N_\ell$ for which the Multilevel estimator $\hat{V}^{ML} = \sum_{\ell=0}^{L} \hat{Y}_\ell$ has a Mean Square Error with bound $MSE[\hat{V}^{ML}] < \varepsilon^2$ and a computational complexity $\bar{C}$ with bound:*

$$\mathbb{E}[\bar{C}] \leq \begin{cases} c_4\, \varepsilon^{-2} & \text{if } \beta > \gamma \\ c_4\, \varepsilon^{-2} \log^2 \varepsilon & \text{if } \beta = \gamma \\ c_4\, \varepsilon^{-2-(\gamma-\beta)/\alpha} & \text{if } \beta < \gamma \end{cases}$$

Recalling the results about Weak Error in section 1.2.1 and assuming that $h_\ell = T\, 2^{-\ell}$, we can affirm that condition 1 is met, with $\alpha = 1$.

Condition 2 (unbiasedness of the telescopic sum) is naturally met when using Monte Carlo estimators. We will show in chapter 5 that this condition will not be violated when introducing Importance Sampling.

In order to check condition 3, we need to introduce some assumptions on the payoff. Let us assume that $P(\cdot)$ is Lipschitz-continuous. This means that:

$$|P(S) - \hat{P}_\ell| \leq \xi\, \|S - \hat{S}\| \tag{4.6}$$

The monotonicity property of expectation combined with the results about Strong Error that we have stated in 1.2.1 yields the following:

$$\mathbb{V}[P - \hat{P}_\ell] \leq \mathbb{E}[(P - \hat{P}_\ell)^2] \leq \xi^2\, \mathbb{E}[\|S - \hat{S}\|^2] = O(h_\ell^\beta) \tag{4.7}$$

where $\beta = 1$ for the Euler-Maruyama approximation and $\beta = 2$ in the Milstein case. An application of Cauchy-Schwartz inequality yields the result: $\sqrt{\mathbb{V}[a-b]} \leq \sqrt{\mathbb{V}[a]} + \sqrt{\mathbb{V}[b]}$. Applying it to $V_\ell$:

$$\mathbb{V}[\hat{P}_\ell - \hat{P}_{\ell-1}] = \mathbb{V}[(\hat{P}_\ell - P) - (\hat{P}_{\ell-1} - P)] \leq \left( \sqrt{\mathbb{V}[P - \hat{P}_\ell]} + \sqrt{\mathbb{V}[P - \hat{P}_{\ell-1}]} \right)^2 \tag{4.8}$$

Therefore, the values of $\beta$ in the Complexity Theorem are the ones we have mentioned a few lines above.

As far as condition 4 is concerned, since one path simulation at level $\ell$ implies $2^\ell$ iterations of fixed size block of instructions, without any nested `for` loops, $C_\ell^1 = O(2^\ell)$. Hence, $\gamma = 1$.

On the basis of the values of $\alpha$, $\beta$, $\gamma$ we have derived, Complexity theorem ensures that the computational cost to achieve $O(\varepsilon^2)$ Mean Square Error is $O(\varepsilon^{-2} \log^2 \varepsilon)$. How does this compare to standard Monte Carlo? We have seen in section 1.2.1 that the MSE of a standard Monte Carlo estimator is $O(1/N + h^2)$, which requires $N = O(\varepsilon^{-2})$ and $h = O(\varepsilon)$ to achieve our target $O(\varepsilon^2)$ MSE accuracy. The resulting computational cost is $C = O(N/h) = O(\varepsilon^{-3})$, which proves the benefit of the Multilevel approach.

### 4.1.1 Smoothing of non-Lipschitz payoffs

In the previous section we have assumed to deal with Lipschitz-continuous payoffs. However, the payoff of a Digital Put is originally discontinuous, hence non-Lipschitz. This feature has a detrimental consequence on the order of $V_\ell$. In fact:

$$\mathbb{E}[\hat{Y}_\ell^2] \;=\; O(1) \cdot \mathbb{P}\left(\hat{P}_\ell \neq \hat{P}_{\ell-1}\right) \tag{4.9}$$

Denoting by $\hat{S}^f$ the final value of $\hat{S}_n$ at the finer level $\ell$ and $\hat{S}^c$ the corresponding quantity at the coarser level $(\ell-1)$, the case when $(\hat{P}_\ell \neq \hat{P}_{\ell-1}$ occurs when $\hat{S}^f < K$ and $\hat{S}^c > K$ or vice versa. This happens for an $O(\sqrt{h_\ell})$ fraction of paths using an Euler-Maruyama time-stepping and for an $O(h_\ell)$ fraction if the Milstein approximation is used.

Therefore, the order of convergence for the variance of the Multilevel estimators $(\beta)$ decreases to 0.5 and 1 respectively. Recalling the relation between $V_\ell$ and $N_\ell$, this effect is clearly detrimental.

Applying the payoff smoothing technique explained in section 1.3.1, the difference between the payoffs corresponding to $\hat{S}^f$ and $\hat{S}^c$ around the strike K is:

$$\hat{P}_\ell - \hat{P}_{\ell-1} \;=\; O\left(\frac{\|\hat{S}^f - \hat{S}^c\|}{\sqrt{h_\ell}}\right) \;=\; \begin{cases} O\left(1\right) & \text{Euler-Maruyama} \\ O\left(\sqrt{h_\ell}\right) & \text{Milstein} \end{cases} \tag{4.10}$$

The result comes from an application of the Mean Value Theorem.

The practical implementation of the payoff smoothing explained in 1.3.1 requires some care at the coarser level. In fact, the results we have presented hold only if we re-use the first half of the increment (which is the penultimate step at the finer level). Hence:

$$\hat{P}_{\ell-1} \;=\; \Phi\left(\frac{K - \hat{S}^c_{M^c-1}\left(1 + r h_{\ell-1} + \sigma \Delta W_{M^f-1}\right)}{\sigma\sqrt{h_\ell}\hat{S}^c_{M^c-1}}\right) \tag{4.11}$$

where $M^f = 2^\ell$ and $M^c = 2^{\ell-1}$.

It is worth pointing out what we have just shown: payoff smoothing introduces tangible benefit in terms of order of variance convergence only when using Milstein discretisation. Nevertheless, differentiability is always a positive feature (for example, it would allow for a broader range of techniques if we needed to compute Greeks).

# Chapter 5

# Multilevel Monte Carlo and Importance Sampling combined

## 5.1 Preliminary clarifications

In this section we will show the benefits of combining Multilevel Monte Carlo with Importance Sampling. However, we need to clarify a few important points before showing numerical results.

### 5.1.1 The Importance Sampling Multilevel Estimators

Equation (4.2) allowed for a certain freedom of choice for the estimators $\hat{P}_\ell$ and $\hat{P}_{\ell-1}$: in fact, we only require that they do not violate the consistency of the telescopic sum in (4.1). In the standard Multilevel framework (assuming that the refinement factor is 2, as we have done so far), the Brownian Increments used in the path simulation for $\hat{S}^c$ are derived from the Brownian Increments of the finer level as follows:

$$\Delta W_{c,n} = \Delta W_{f,2n} + \Delta W_{f,2n+1}, \qquad n = 1, \dots, 2^{\ell-1} \tag{5.1}$$

Using the change in drift and variance, alongside with the Brownian Bridge construction (as we have exposed in chapter 2) leads to slightly different Radon-Nikodym derivatives for the two levels. Namely:

$$RN_\ell \;=\; |\tilde{\sigma}| \exp\left\{ (1-\tilde{\sigma}^2)\,\frac{Z_1^2}{2} \;+\; \sum_{n=1}^{2^\ell} \left( \frac{1}{2}\,\tilde{\mu}_n^2\, h_\ell - \tilde{\mu}_n \Delta W_{f,n}^{IS} \right) \right\} \tag{5.2}$$

$$RN_{\ell-1} \;=\; |\tilde{\sigma}| \exp\left\{ (1-\tilde{\sigma}^2)\,\frac{Z_1^2}{2} \;+\; \sum_{n=1}^{2^{\ell-1}} \left( \frac{1}{2}\,\tilde{\mu}_n^2\, h_{\ell-1} - \tilde{\mu}_n \Delta W_{c,n}^{IS} \right) \right\} \tag{5.3}$$

It is worth mentioning that, in the case of a digital put with the payoff smoothing (1.3.1), the last term in the summations is dropped, since we are ending our stochastic simulation at $M^f - 1$ and $M^c - 1$ respectively. The resulting Multilevel Importance Sampling estimator is:

$$\hat{V}_{ML}^{IS} = \frac{1}{N_0}\sum_{i=1}^{N_0} \hat{P}_0^{(i)} RN_0^{(i)} + \sum_{\ell=1}^{L}\left(\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\left(\hat{P}_\ell\, RN_\ell - \hat{P}_{\ell-1}\, RN_{\ell-1}\right)^{(i)}\right) \tag{5.4}$$

Recalling the unbiasedness results we have proved in chapter 2, the consistency condition follows immediately:

$$\mathbb{E}_s[\hat{P}_\ell\, RN_\ell] = \mathbb{E}_q[\hat{P}_\ell] \quad \text{and} \quad \mathbb{E}_s[\hat{P}_{\ell-1}\, RN_{\ell-1}] = \mathbb{E}_q[\hat{P}_{\ell-1}] \tag{5.5}$$

Note: in section 2.1, our derivation of the optimal Importance Sampling distribution assumed that $f(\cdot)$ is a positive function. Of course, this condition cannot be verified for the Multilevel estimator $(\hat{P}_\ell - \hat{P}_{\ell-1})$. We will simply mention that, for a general integrand function $f(\cdot)$, the Importance Sampling distribution $s(x)$ is proportional to $|f(x)|\, q(x)$ (please refer to [KM53] for full detail). Figure 5.2.1.2 will add insight into this aspect.

### 5.1.2 Choice for the penultimate step

In (4.11) we have shown how the smoothed payoff for a digital option is built at the coarser level. However, in the Importance Sampling framework this definition is not obvious, as we have to choose whether the penultimate Brownian Increment is drawn from the original or from the Importance Sampling measure. Choosing to sample from the Importance Sampling measure adds a term in the summation inside $RN_{\ell-1}$, corresponding to the step $M^f - 2$.

We will show the effects of this choice later on in this chapter.

### 5.1.3 How many degrees of freedom?

A priori, we have no knowledge about the dependence between the optimal Importance Sampling parameters and the level $\ell$. We will start allowing each level to have different parameters (which implies performing an optimisation algorithm at each step). More specifically, we will use the optimal parameters at level $(\ell - 1)$ as a starting point when optimising for level $\ell$. For practical reasons, we have reduced the tolerance $\xi$ (see section 3.2.4) to 10% for $\ell \geq 6$.

### 5.1.4 What to minimise?

When it comes to running the optimisation algorithm, we have to decide which quantity should be minimised. Since the estimators $\hat{P}_\ell$ and $(\hat{P}_\ell - \hat{P}_{\ell-1})$ have different features, we have considered minimising for $\mathbb{E}_s\left[\left(\hat{P}_\ell\, RN_\ell\right)^2\right]$ and $\mathbb{E}_s\left[\left(\hat{P}_\ell\, RN_\ell - \hat{P}_{\ell-1}\, RN_{\ell-1}\right)^2\right]$. Results for both will follow.

## 5.2 Simulations for a Digital Put. $S_0 = 100$, $K = 50$

### 5.2.1 Economised model 1: only $\mu_0$ and $\tilde{\sigma}$

We will start with an economised model, where we will assume that the Importance Sampling drift term is constant throughout time. In the following sections we will use this notation to indicate different Importance Sampling approaches.

- i/ii: Minimising $\mathbb{E}_s[(\hat{P}_\ell\, RN_\ell - \hat{P}_{\ell-1}\, RN_{\ell-1})^2]$ or $\mathbb{E}_s[(\hat{P}_\ell\, RN_\ell)^2]$
- a/b: with and without Importance Sampling in the penultimate step at the Coarse level

We have used the following quantity as a proxy for the computational cost:

$$\text{Cost} \propto \sum_{\ell=0}^{L} 2^\ell\, N_\ell \tag{5.6}$$

Note: we have excluded from all plots the values of $V_0$ and $N_0$. In fact, because of payoff smoothing, simulating at level 0 produces a constant estimator.

#### 5.2.1.1 Euler-Maruyama time-stepping

We have performed several simulations. For each, we have multiplied the payoff by 10 and set the target accuracy $\varepsilon$ to $10^{-4}$. We have observed that all approaches outperform standard Multilevel Monte Carlo and among them, i.a brings about the greatest computational savings. Results and plots follow in the next pages.

In addition, we have added contour plots of the surfaces $\mathbb{E}_s[(\hat{P}_\ell\, RN_\ell)^2]$ and $\mathbb{E}_s[(\hat{P}_\ell\, RN_\ell - \hat{P}_{\ell-1}\, RN_{\ell-1})^2]$ as functions of $(\mu_0, \tilde{\sigma})$ (note: gaps between subsequent lines in contour plots, looking like discontinuities, are simply due to our choice for non uniform spacing between levels). These plots provide further insight into the difference between approaches i and ii, showing that the optimal values for the two objective functions are effectively different, especially for small values of $\ell$ (or equivalently $M$).

**No Importance Sampling**

| $\text{Cost}_{ML}$ | $V$ | $\hat{V}_{ML}$ | $\text{WE}_{ML}$ |
|---|---|---|---|
| $1.97 \cdot 10^9$ | 0.020952 | 0.021113 | $1.62 \cdot 10^{-4}$ |

**Approach i.a**

| $\text{Cost}_{ML}^{IS}$ | $\text{Cost}_{ML}/\text{Cost}_{ML}^{IS}$ | $V$ | $\hat{V}_{ML}^{IS}$ | $\text{WE}_{ML}^{IS}$ |
|---|---|---|---|---|
| $2.82 \cdot 10^7$ | 69.9 | 0.020952 | 0.020936 | $1.56 \cdot 10^{-5}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mu_0$ | -1.64 | -1.78 | -2.15 | -2.42 | -2.62 | -2.70 | -2.78 | -2.80 | -2.83 | -2.84 |
| $\tilde{\sigma}$ | 0.90 | 0.86 | 0.65 | 0.49 | 0.34 | 0.28 | 0.19 | 0.13 | 0.09 | 0.07 |

**Approach ii.a**

| $\text{Cost}_{ML}^{IS}$ | $\text{Cost}_{ML}/\text{Cost}_{ML}^{IS}$ | $V$ | $\hat{V}_{ML}^{IS}$ | $\text{WE}_{ML}^{IS}$ |
|---|---|---|---|---|
| $4.46 \cdot 10^7$ | 44.2 | 0.020952 | 0.020993 | $4.14 \cdot 10^{-5}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mu_0$ | -2.80 | -2.96 | -2.96 | -3.05 | -3.09 | -3.10 | -3.11 | -3.12 | -3.11 | -3.11 |
| $\tilde{\sigma}$ | 0.97 | 0.76 | 0.62 | 0.47 | 0.39 | 0.38 | 0.40 | 0.42 | 0.36 | 0.30 |

**Approach i.b**

| $\text{Cost}_{ML}^{IS}$ | $\text{Cost}_{ML}/\text{Cost}_{ML}^{IS}$ | $V$ | $\hat{V}_{ML}^{IS}$ | $\text{WE}_{ML}^{IS}$ |
|---|---|---|---|---|
| $3.52 \cdot 10^8$ | 5.6 | 0.020952 | 0.021092 | $1.40 \cdot 10^{-4}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mu_0$ | -0.17 | -0.66 | -1.10 | -2.29 | -2.92 | -3.00 | -3.03 | -3.00 | -3.02 | -3.01 |
| $\tilde{\sigma}$ | 1.20 | 1.33 | 1.39 | 0.88 | 0.50 | 0.40 | 0.38 | 0.32 | 0.31 | 0.30 |

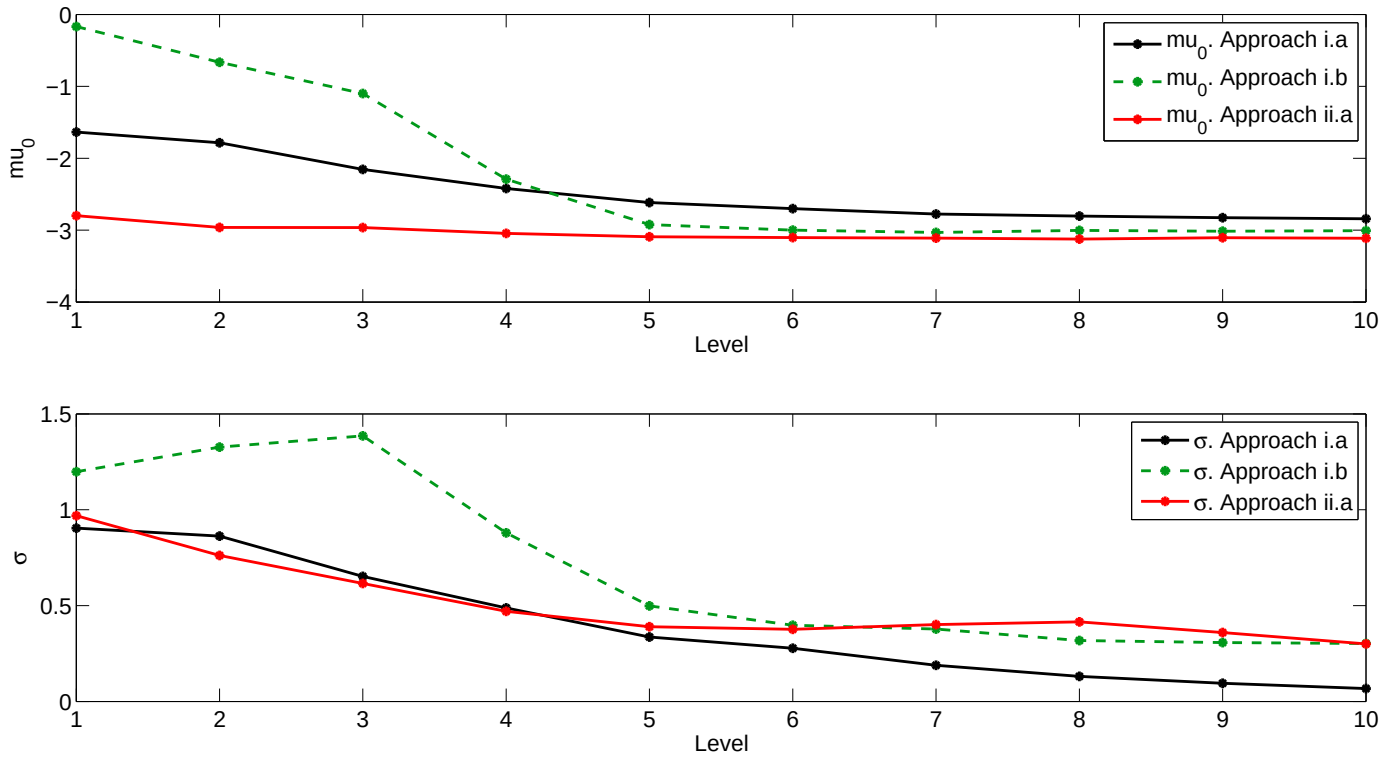


Figure 5.1: Values of $\mu_0$ (above) and $\tilde{\sigma}$ (below) for three different approaches. Euler-Maruyama scheme.

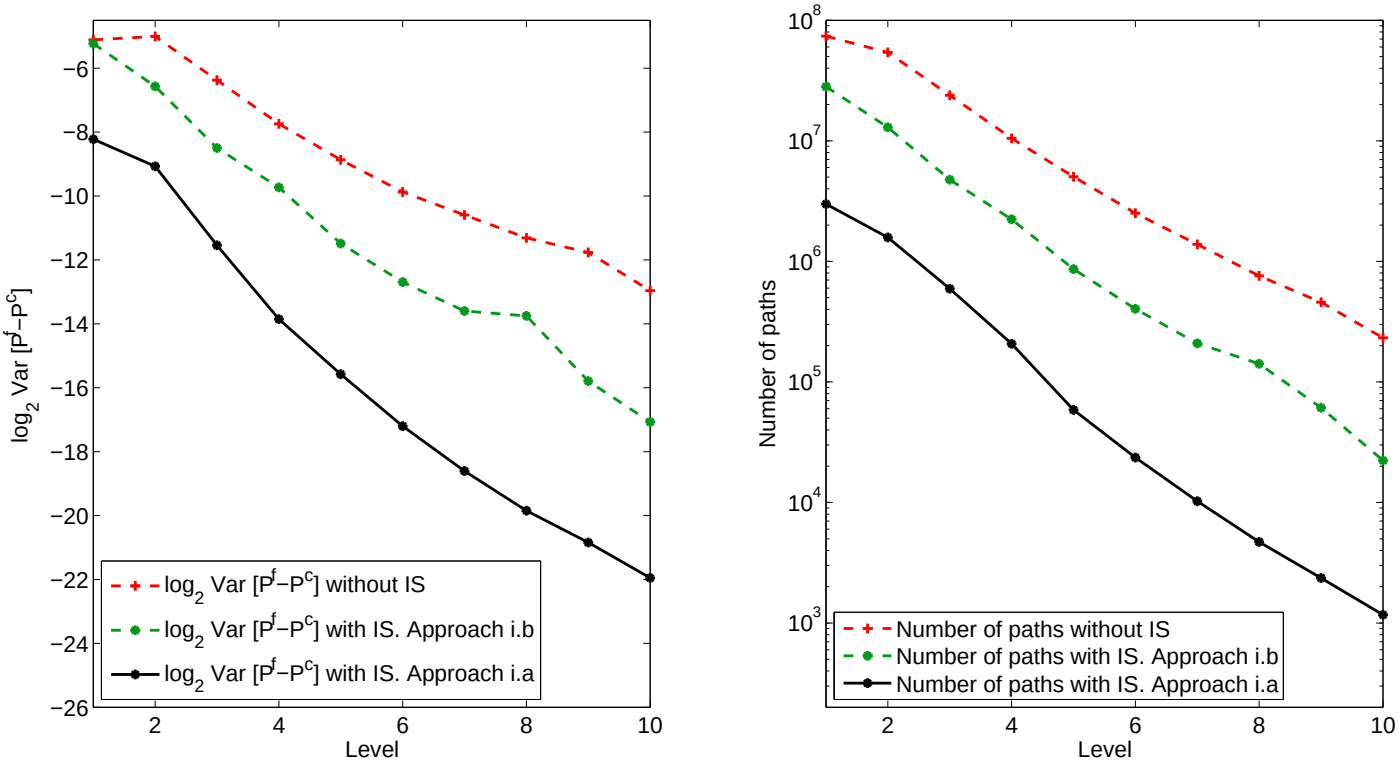


Figure 5.2: Plot of $\log_2 V_\ell$ and $N_\ell$ with (solid black) and without (dashed red) Importance Sampling. Euler-Maruyama scheme, Approach i.a and i.b

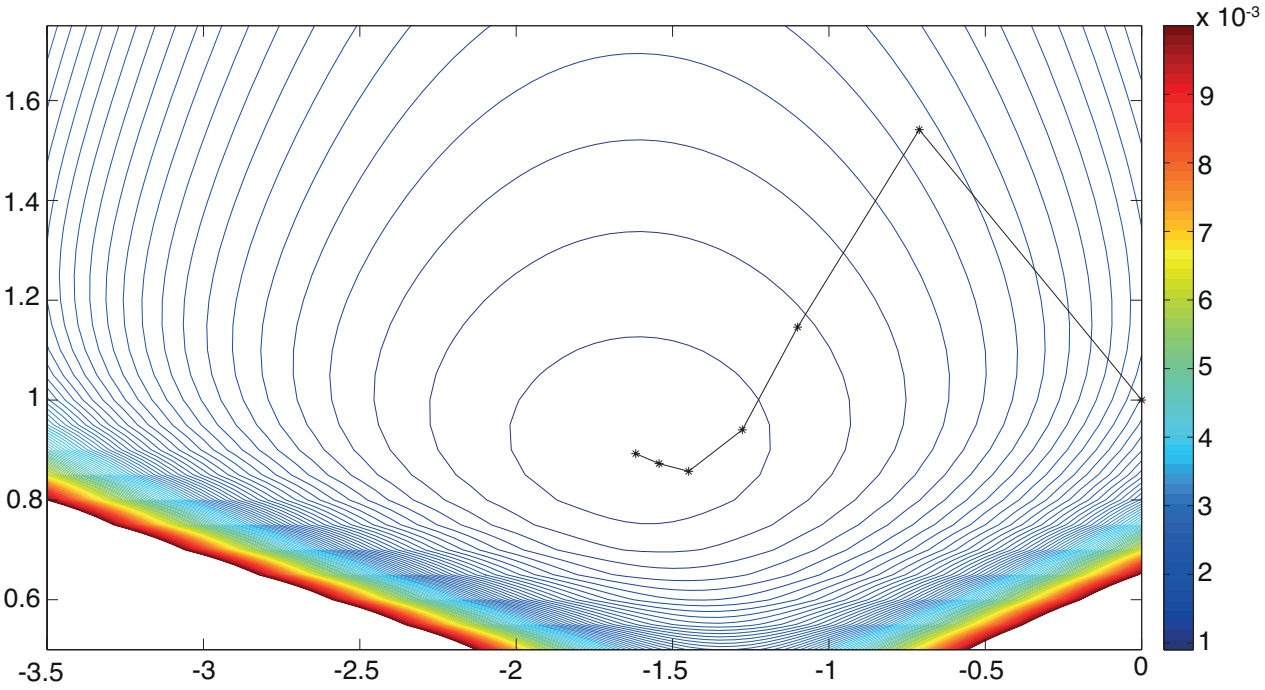


Figure 5.3: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach i. $M = 2$

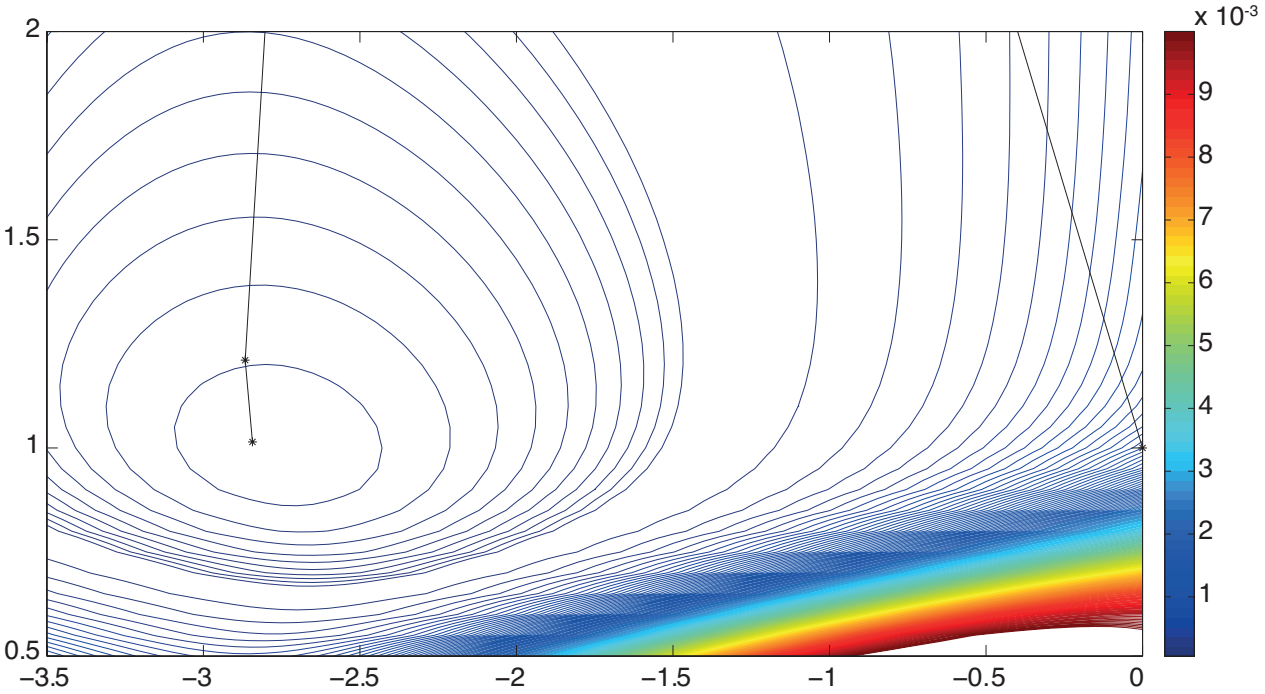


Figure 5.4: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach ii. $M = 2$

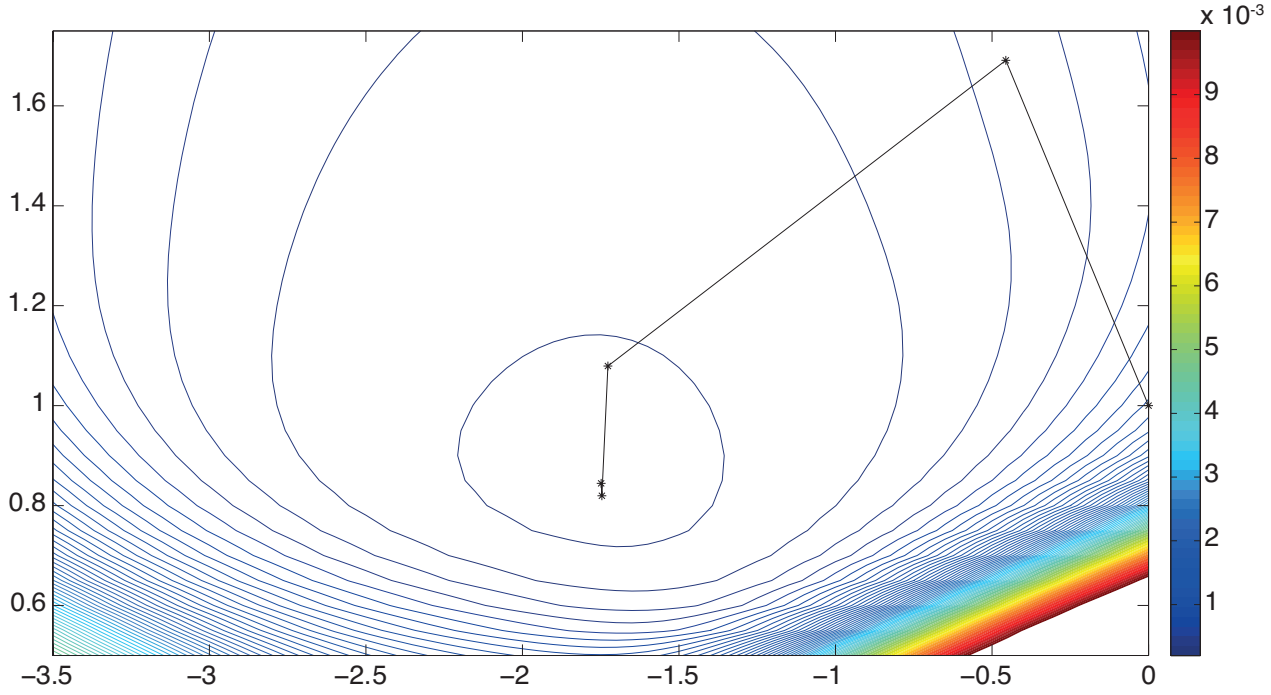


Figure 5.5: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach i. $M = 4$

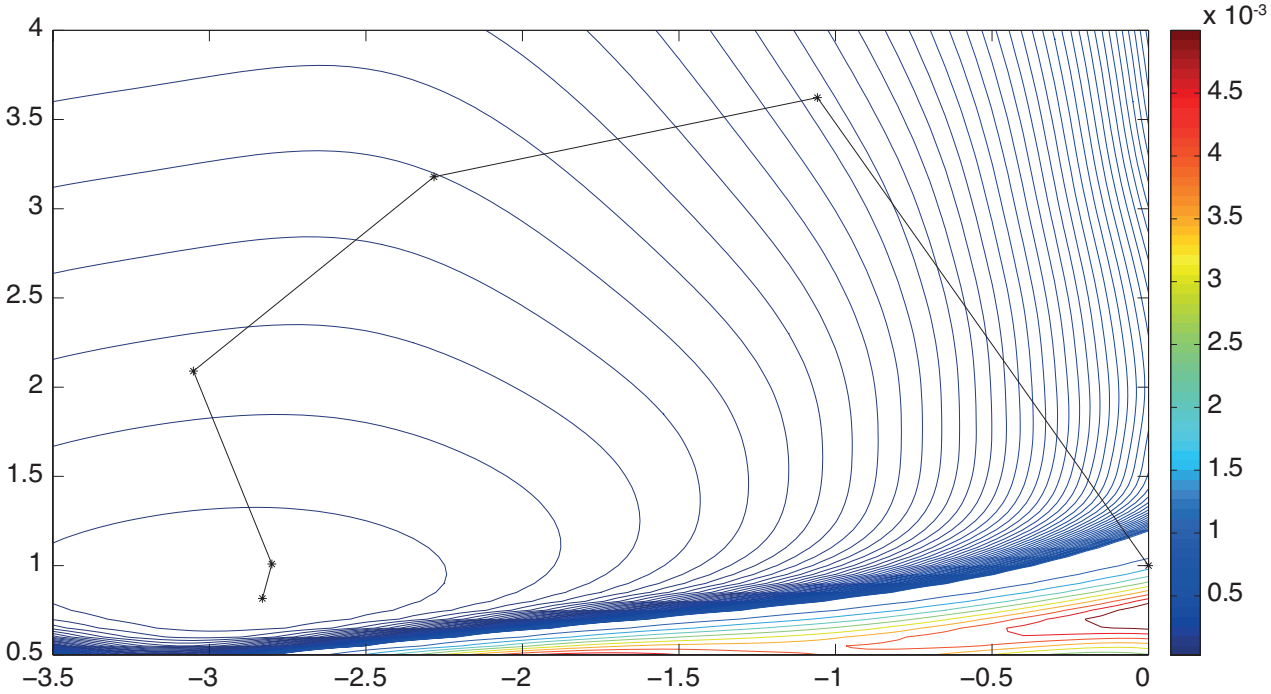


Figure 5.6: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach ii. $M = 4$

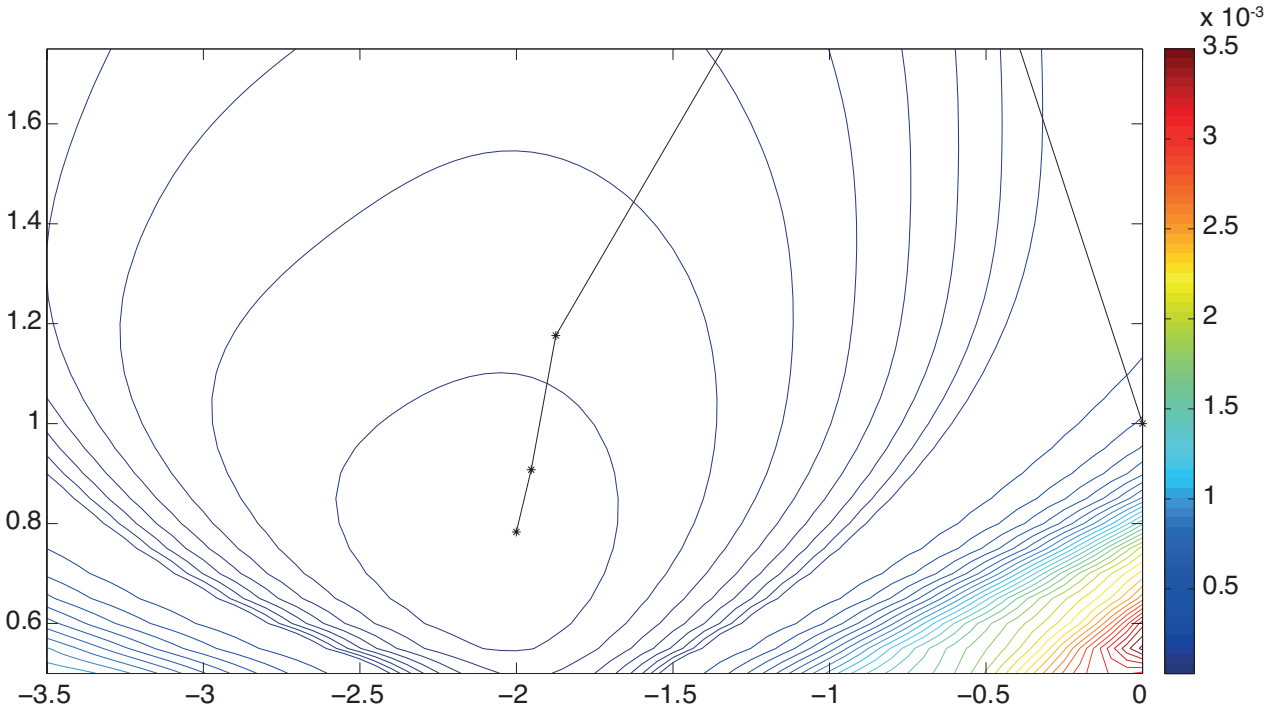


Figure 5.7: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach i. $M = 8$

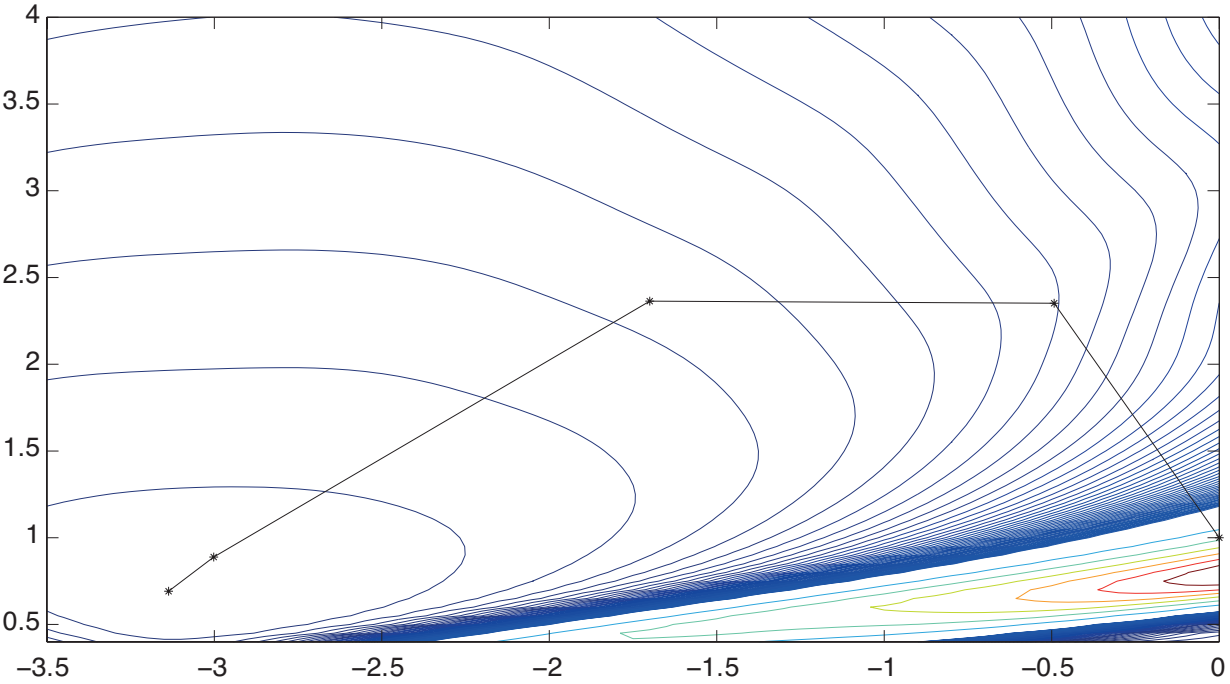


Figure 5.8: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach ii. $M = 8$

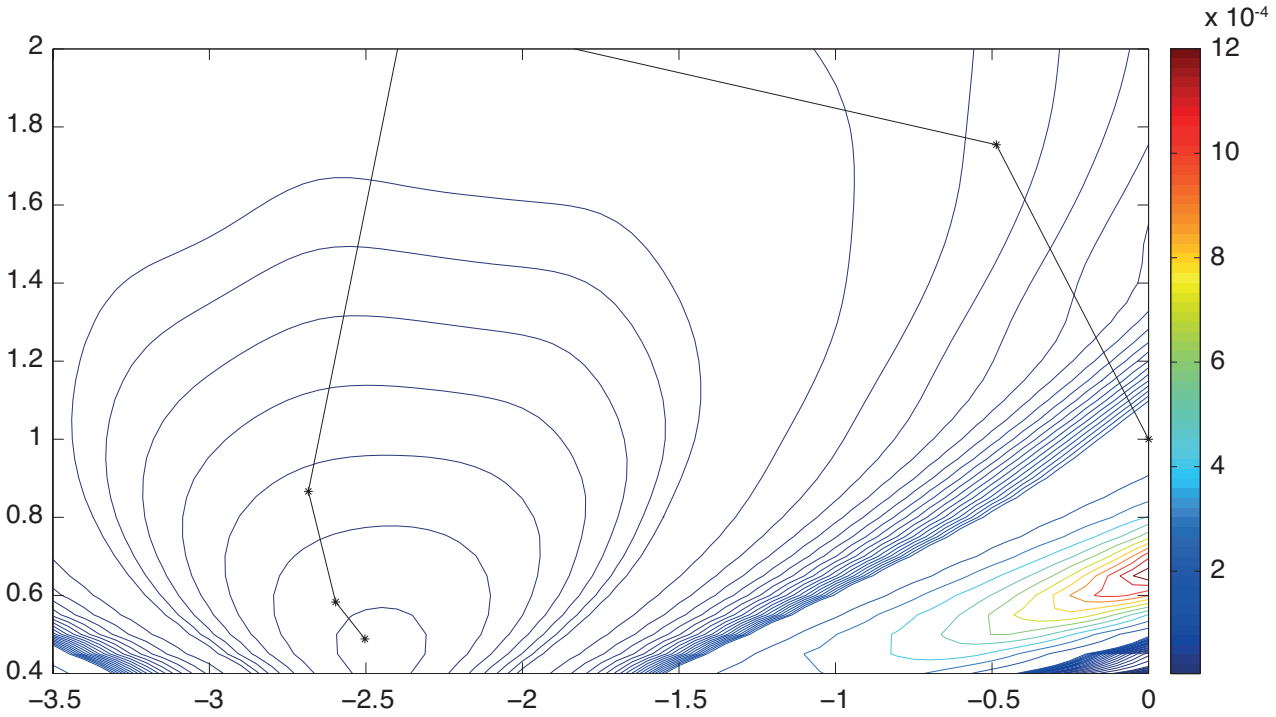


Figure 5.9: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach i. $M = 16$

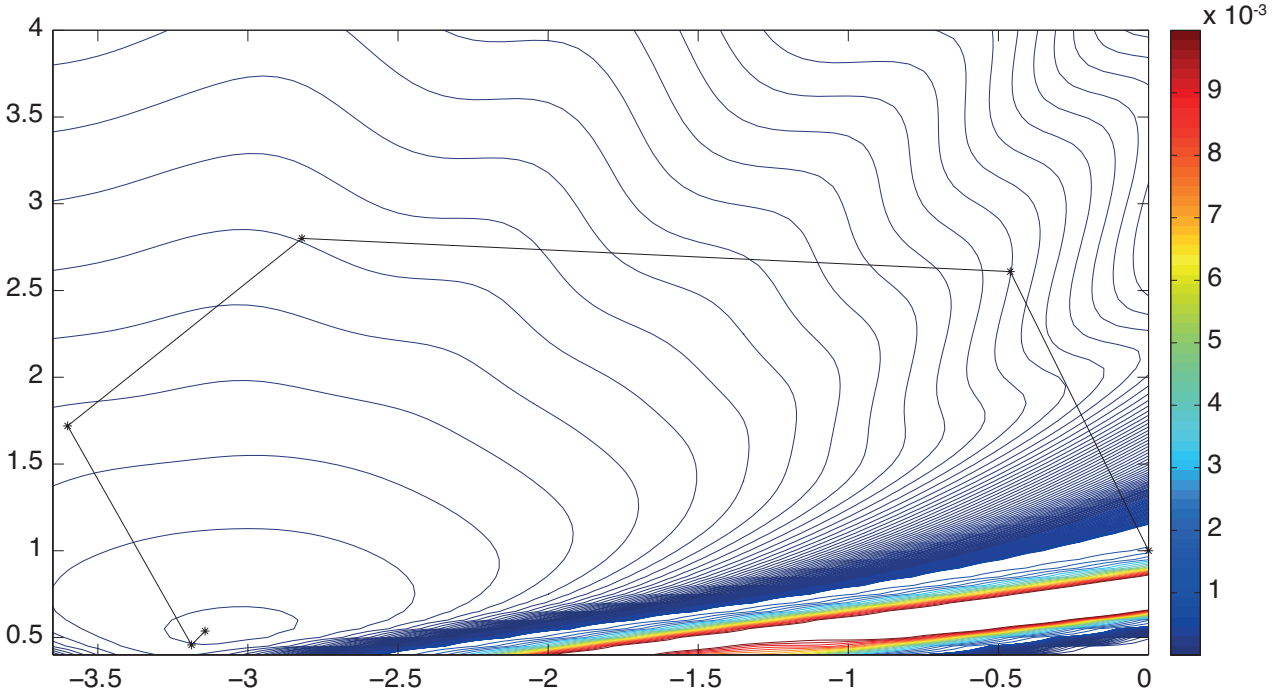


Figure 5.10: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach ii. $M = 16$

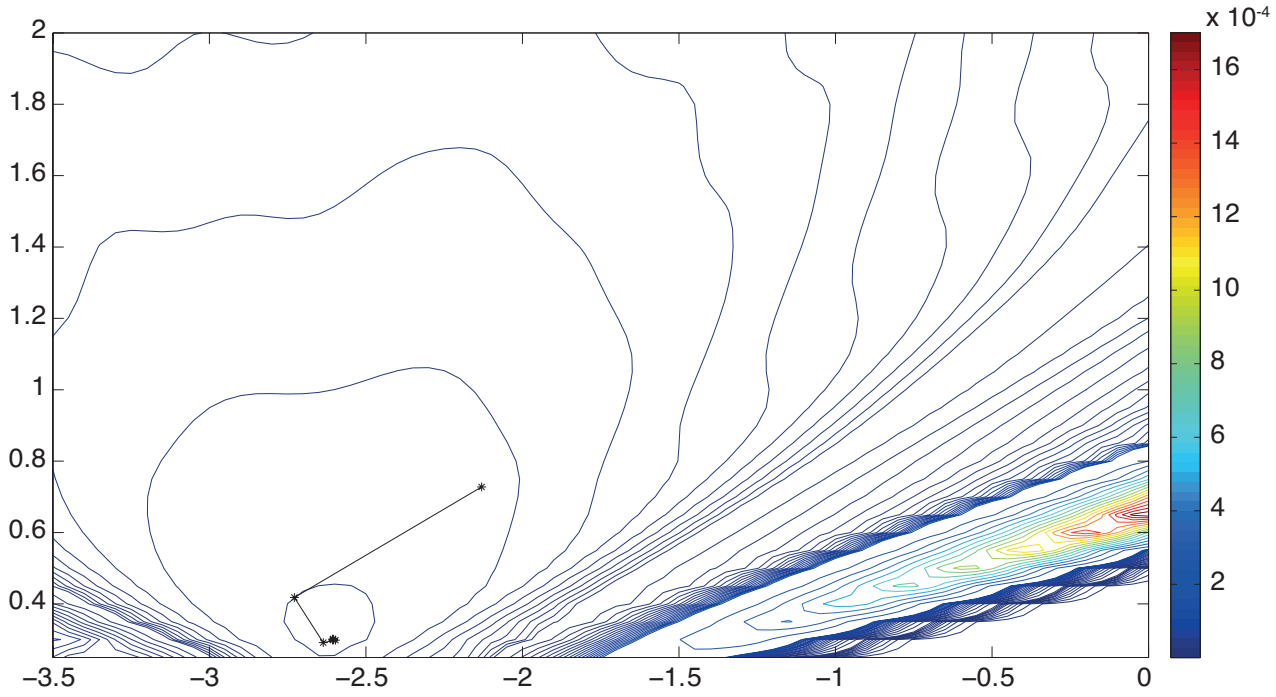


Figure 5.11: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach i. $M = 32$

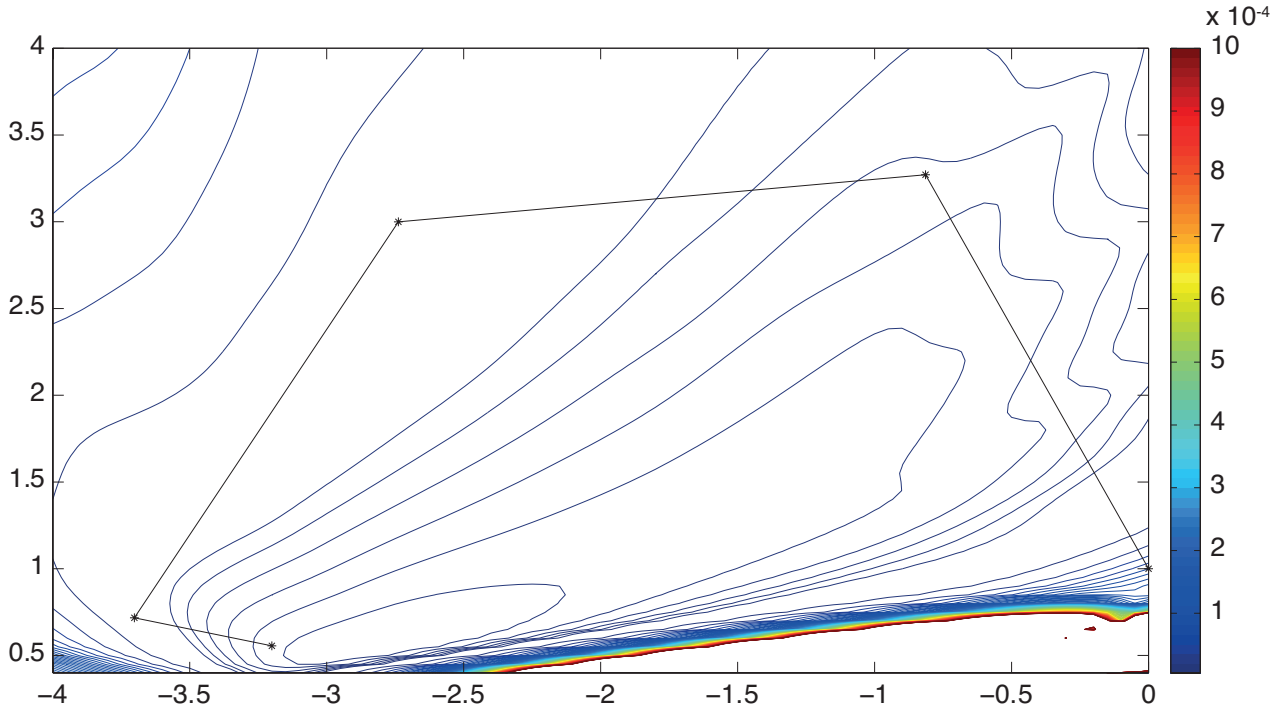


Figure 5.12: Contour plot of $\varphi(\mu_0, \tilde{\sigma})$ approach ii. $M = 32$

#### 5.2.1.2 Milstein time-stepping

We have repeated the same simulations as in the previous section using Milstein scheme. Guided by the results obtained so far, we have decided not to consider approach b. We have reported some key quantities in the following tables and plots. Moreover, we have plotted the empirical distributions of $\hat{S}^f$ at the final time-step both for Euler-Maruyama and Milstein scheme.

**No Importance Sampling**

| $\text{Cost}_{ML}$ | $V$ | $\hat{V}_{ML}$ | $\text{WE}_{ML}$ |
|---|---|---|---|
| $2.80 \cdot 10^8$ | 0.020952 | 0.020875 | $7.71 \cdot 10^{-5}$ |

**Approach i.a**

| $\text{Cost}_{ML}^{IS}$ | $\text{Cost}_{ML}/\text{Cost}_{ML}^{IS}$ | $V$ | $\hat{V}_{ML}^{IS}$ | $\text{WE}_{ML}^{IS}$ |
|---|---|---|---|---|
| $1.47 \cdot 10^7$ | 18.9 | 0.020952 | 0.020850 | $1.01 \cdot 10^{-4}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\mu_0$ | -1.74 | -1.64 | -2.12 | -2.49 | -2.65 | -2.76 | -2.80 | -2.82 |
| $\tilde{\sigma}$ | 0.92 | 0.84 | 0.67 | 0.50 | 0.36 | 0.26 | 0.19 | 0.14 |

**Approach ii.a**

| $\text{Cost}_{ML}^{IS}$ | $\text{Cost}_{ML}/\text{Cost}_{ML}^{IS}$ | $V$ | $\hat{V}_{ML}^{IS}$ | $\text{WE}_{ML}^{IS}$ |
|---|---|---|---|---|
| $2.04 \cdot 10^7$ | 13.7 | 0.020952 | 0.020906 | $4.53 \cdot 10^{-5}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\mu_0$ | -2.08 | -2.95 | -3.10 | -3.11 | -3.11 | -3.14 | -3.11 | -3.10 |
| $\tilde{\sigma}$ | 0.97 | 0.82 | 0.62 | 0.48 | 0.37 | 0.41 | 0.31 | 0.26 |

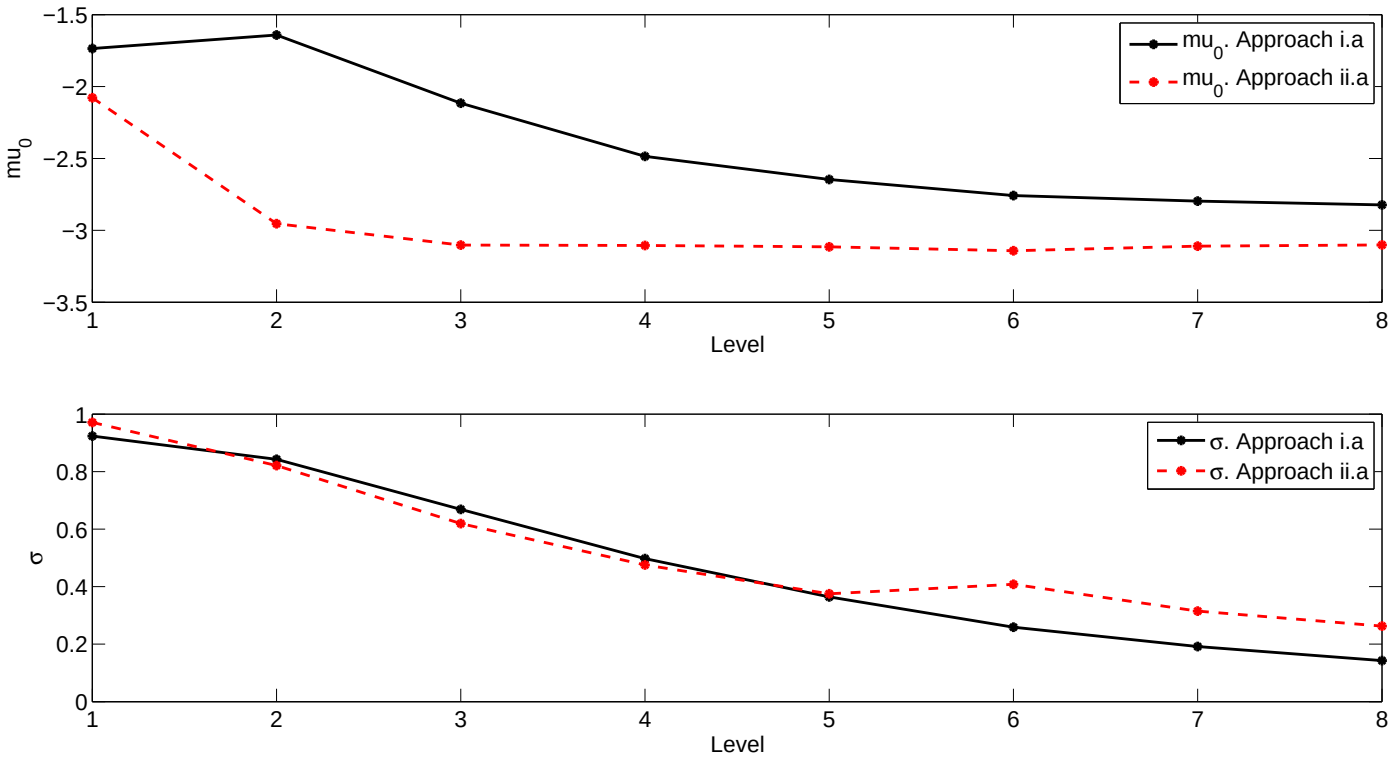


Figure 5.13: Values of $\mu_0$ (above) and $\tilde{\sigma}$ (below) for two different approaches. Milstein scheme.

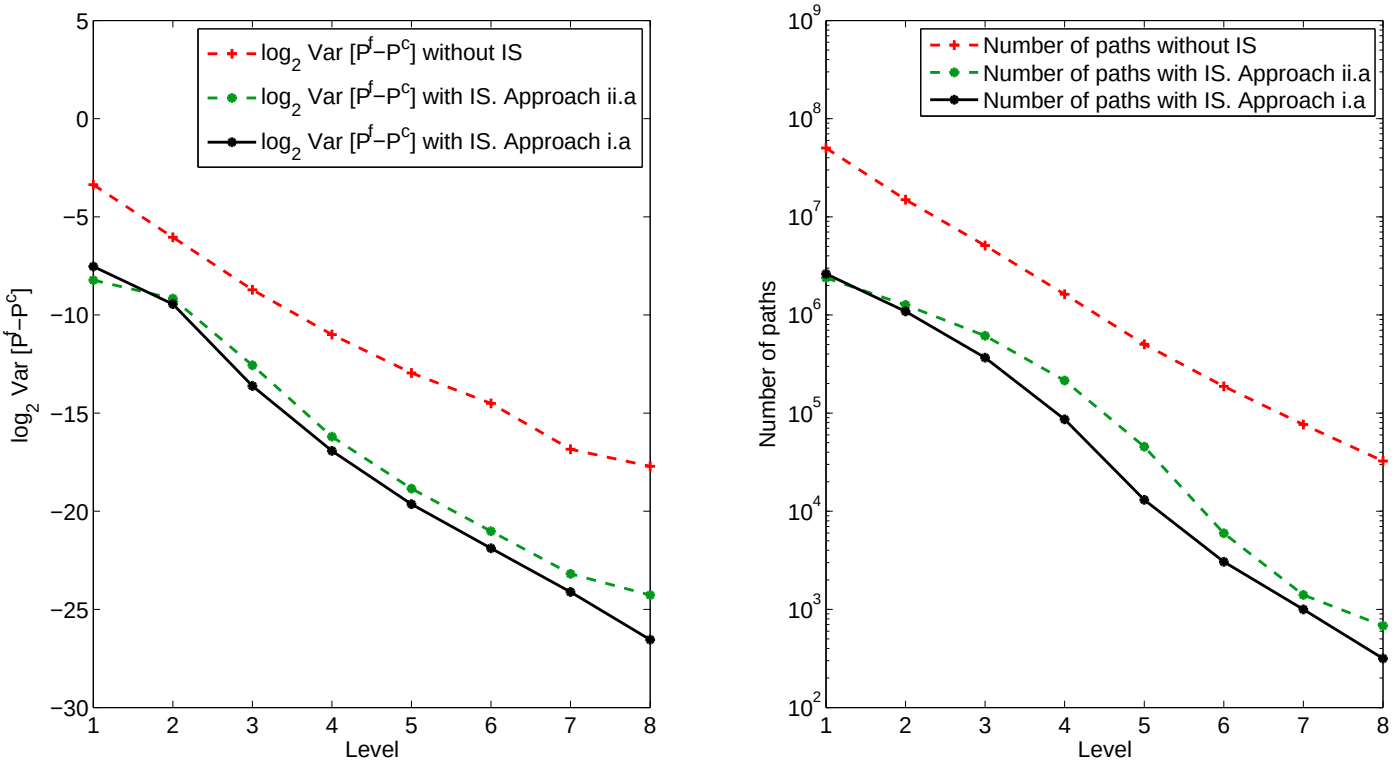


Figure 5.14: Plot of $\log_2 V_\ell$ and $N_\ell$ with (solid black) and without (dashed red) Importance Sampling. Milstein scheme, Approach i.a and ii.a

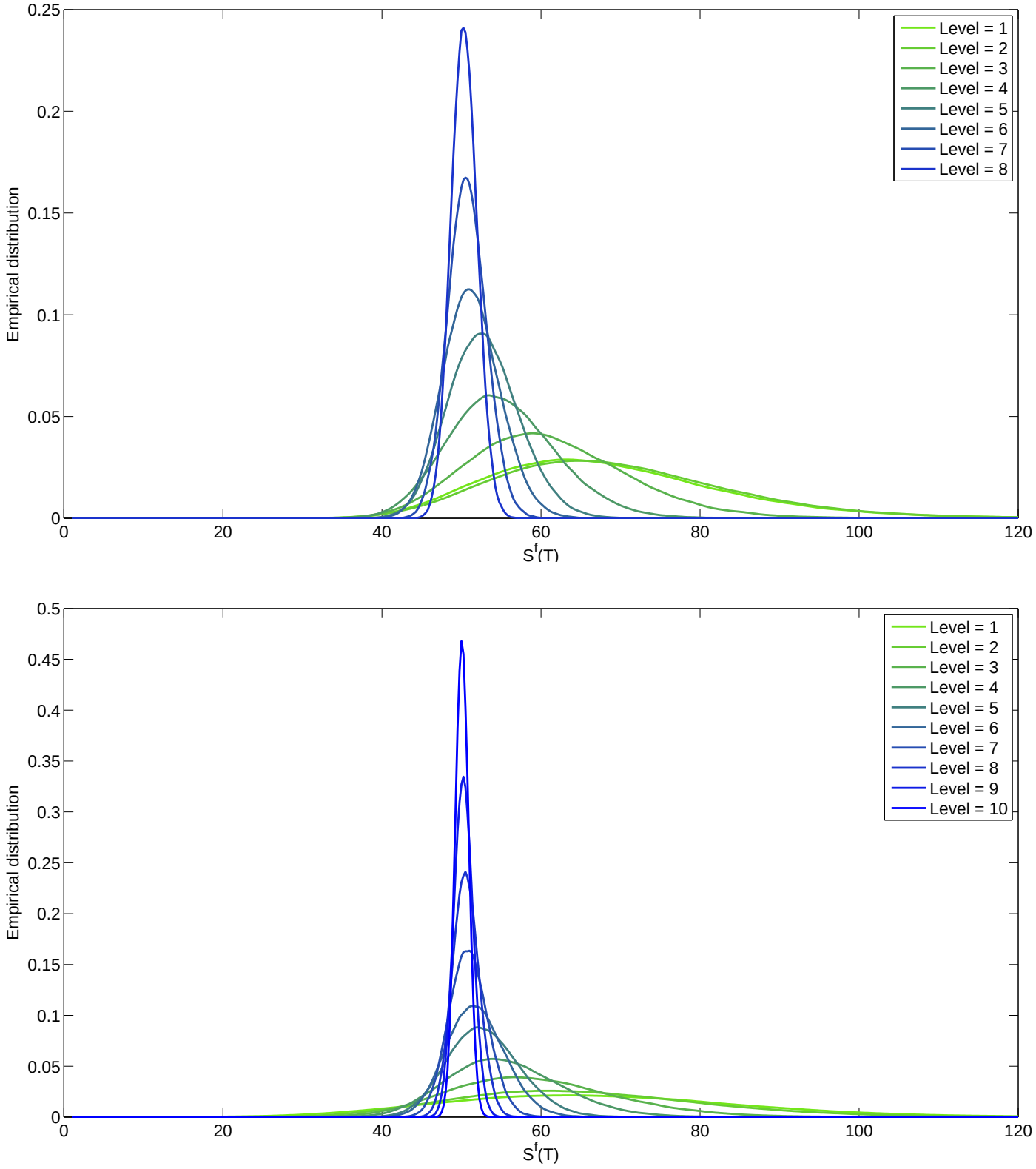


Figure 5.15: Empirical distributions of $\hat{S}^f$ on different levels. Milstein (above) and Euler-Maruyama (below) discretisation. Approach i.a

As expected, distributions exhibit a peak around $S_T = K = 50$, where $|\hat{P}_\ell - \hat{P}_{\ell-1}|$ reaches its maximum. It is also worth noticing that in both cases $\tilde{\sigma}$ exhibits a downwards trend. This is not surprising: as we have seen in section 4.1.1, the fraction of paths for which $\hat{P}_\ell$ and $\hat{P}_{\ell-1}$ differ (hence, $|\hat{P}_\ell - \hat{P}_{\ell-1}|$ reaches its maximum) decreases as $\ell$ increases. Analogously, the separation between the corresponding values of $\hat{S}^f_{Mf}$ and $\hat{S}^c_{Mc}$ decreases as $\ell$ increases. Therefore, the range of values of $\hat{S}^f_{Mf}$ which are likely to have produced a difference in the payoffs collapses around $K$ as $\ell$ increases. A more detailed explanation of this phenomenon can be found at page 259 of [Gla04].

It is also worth commenting about the greater improvement that Importance Sampling brings about when Euler-Maruyama scheme is employed. Figures 5.2.1.1 and 5.2.1.2 show that variance reduction due Importance Sampling is more significant at the finest levels. However, recalling how the optimal set of $N_\ell$ is computed (see chapter 4) and especially its $O(2^{-\beta/2\ell})$ dependence on $\beta$, we expect the computational cost to be more spread across all levels for the Euler-Maruyama approximation ($\beta = 1/2$); on the other hand, since $\beta = 3/2$ for the Milstein scheme, we expect the majority of the computational cost to be concentrated in the coarsest levels, where the variance reduction effects are less significant. The following bar plots summarise what we have just mentioned.

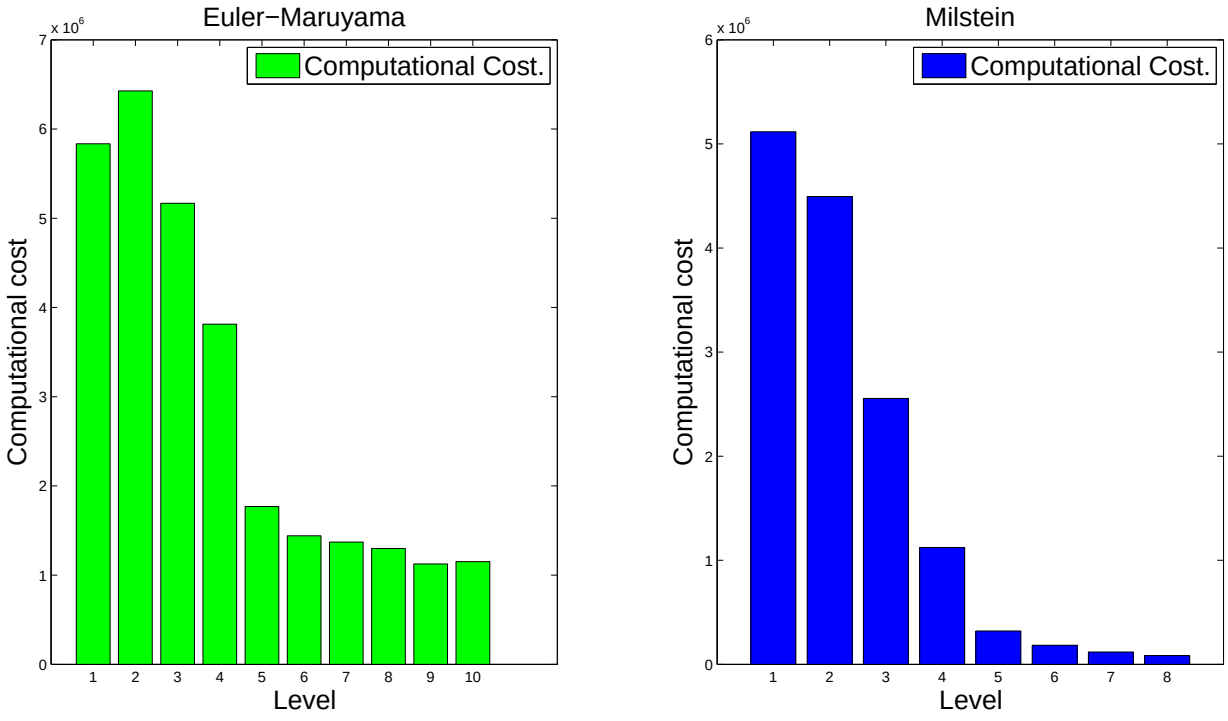


Figure 5.16: Computational cost at each level, using Importance Sampling. Euler-Maruyama (green) and Milstein (blue) schemes.

### 5.2.2 Full model: $\mu_0$, $\mu_1$, $\mu_2$, $\tilde{\sigma}$

In this section we will compare the benefits of allowing for a time-varying Importance Sampling drift parameters, in the form described in (2.14). The first experiments showed that, especially at the coarsest level, the objective function has more than one local minimum. For example, at $\ell = 2$, $\varphi(x)$ has also a local minimum at $x = (-0.15, -1.45, -2.07, 0.90)$.

In order to ensure that the optimisation algorithm identifies the global minimum, we have set $N_{opt} = 20000$. A more refined solution could make use of simulated annealing before running the optimisation algorithm.

We have repeated 5 simulation using the Milstein scheme with optimisation approach i.a, with and without time-varying drift. Target accuracy was set to $10^{-4}$. The average of the Importance Sampling parameters for the time-varying case is:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\mu_0$ | -1.33 | -4.94 | -3.51 | -3.10 | -3.10 | -3.01 | -3.05 | -3.03 |
| $\mu_1$ | -0.66 | 15.99 | 6.93 | 2.55 | 1.51 | 0.93 | 0.92 | 0.87 |
| $\mu_2$ | -0.33 | -17.27 | -6.86 | -2.06 | -0.96 | -0.63 | -0.64 | -0.69 |
| $\tilde{\sigma}$ | 0.93 | 0.86 | 0.72 | 0.50 | 0.36 | 0.28 | 0.20 | 0.14 |

We have averaged the following values over the 5 simulations:

| Model | Cost | $\hat{V}$ | $V$ | RMSE$[\hat{V}]$ |
|---|---|---|---|---|
| $\mu_0$, $\mu_1$, $\mu_2$, $\tilde{\sigma}$ | $1.41 \cdot 10^7$ | 0.020966 | 0.020951 | $2.24 \cdot 10^{-5}$ |
| $\mu_0$, $\tilde{\sigma}$ | $1.41 \cdot 10^7$ | 0.020941 | 0.020952 | $4.64 \cdot 10^{-5}$ |

The difference in RMSE and the apparent indifference in terms of computational cost led us to further investigation. We have plotted (figure 5.2.2) the average of variance of the different levels in the two cases. Estimating $\beta$ through linear regression over the 8 levels a posteriori yields $\beta = 2.8$ in the constant case and $\beta = 2.4$ in the time-varying case. Forcing this values in our MLMC algorithm (which usually computes $\beta$ through linear regression over the first 2 or 3 levels), we obtained computational costs of $1.17 \cdot 10^7$ for constant drift and $1.03 \cdot 10^7$ for time-varying drift (comparison in figure 5.2.2).

We conclude this section mentioning the possibility of a hybrid approach, using time-varying drift in the first levels and constant drift in the following. For further experiments with the Digital Put we will consider only the economised model $\mu_0$, $\tilde{\sigma}$.

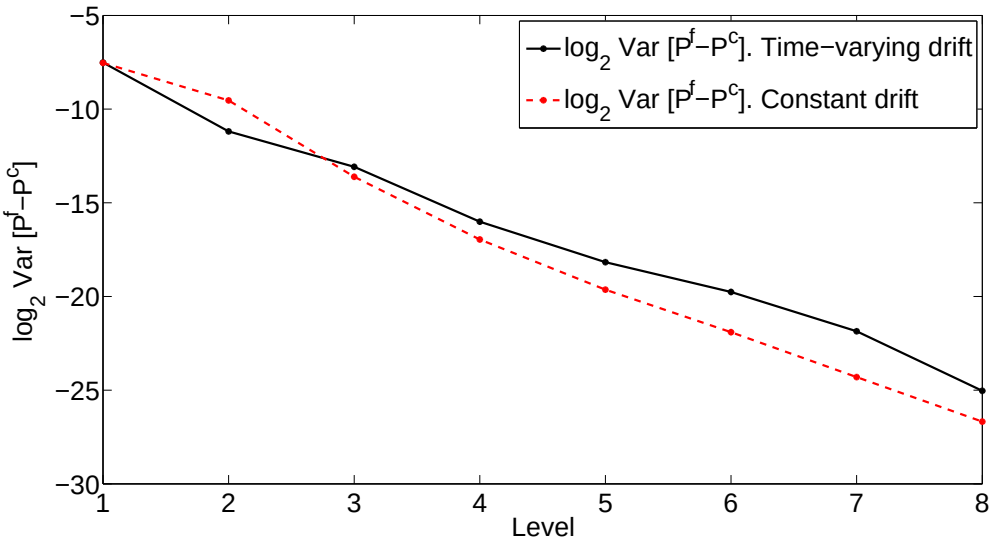


Figure 5.17: Variance of the MLMC estimator at each level. With time-varying and constant Importance Sampling drift

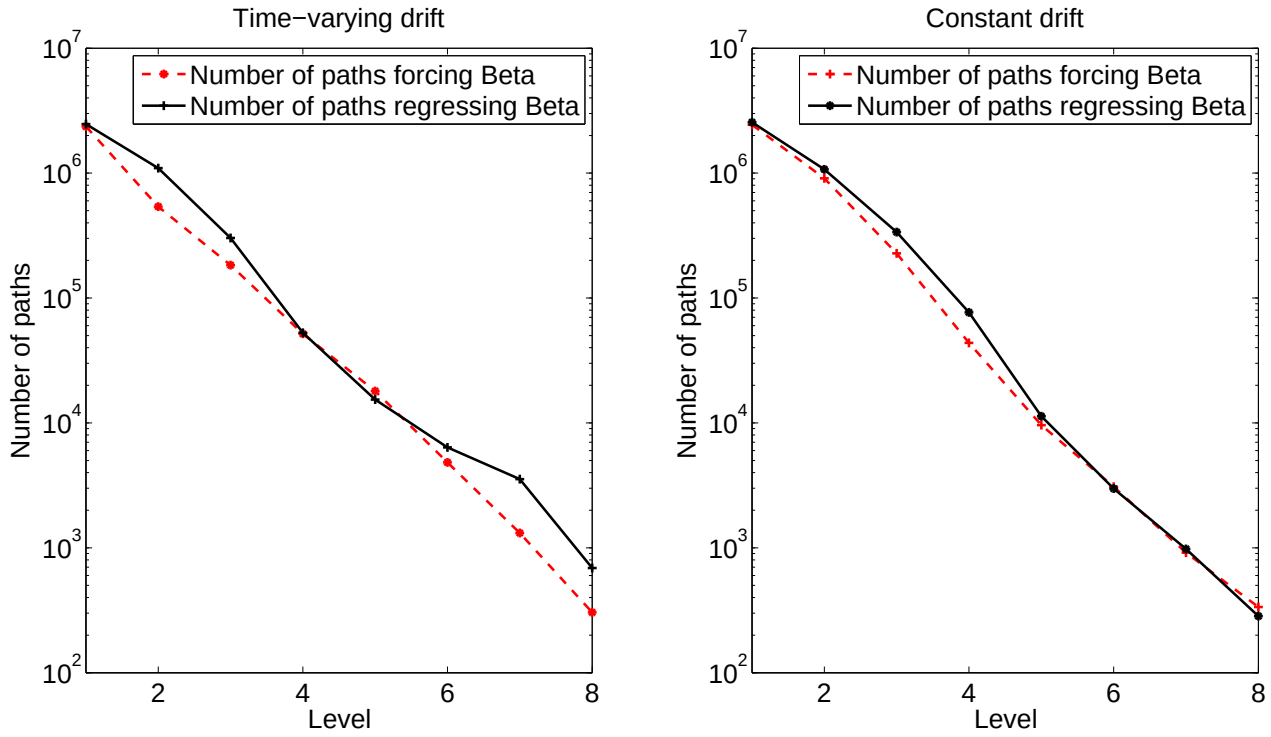


Figure 5.18: Number of paths per level. Time-varying (left) and constant (right) Importance Sampling drift. Forcing (red) and regressing (black) the value of $\beta$

### 5.2.3 Optimising only on coarser levels

So far, we have run the optimisation algorithm at each level, using the optimal parameters of the previous level as a starting point. In order to investigate the sensitivity of the computational cost to the optimality of the Importance Sampling parameters, we have tried performing optimisation only up to a level $\bar{L}$ rather than at each level. For $\ell > \bar{L}$ we have simply set $x_\ell = x_{\bar{L}}$. We have repeated the same experiment 5 times per each value of $\bar{L}$. We have used Milstein scheme and set target accuracy to $5 \cdot 10^{-5}$, resulting in 9-level estimators.

| $\bar{L}$ | Cost | Cost/(Cost$\|_{\bar{L}=8}$) | Achieved RMSE |
|---|---|---|---|
| all | $5.66 \cdot 10^7$ | 100% | $3.99 \cdot 10^{-5}$ |
| 5 | $5.78 \cdot 10^7$ | 102% | $1.93 \cdot 10^{-5}$ |
| 4 | $5.86 \cdot 10^7$ | 103% | $1.38 \cdot 10^{-6}$ |
| 3 | $6.10 \cdot 10^7$ | 108% | $3.56 \cdot 10^{-5}$ |
| 2 | $6.58 \cdot 10^7$ | 116% | $6.55 \cdot 10^{-5}$ |
| 1 | $6.22 \cdot 10^7$ | 110% | $2.78 \cdot 10^{-5}$ |

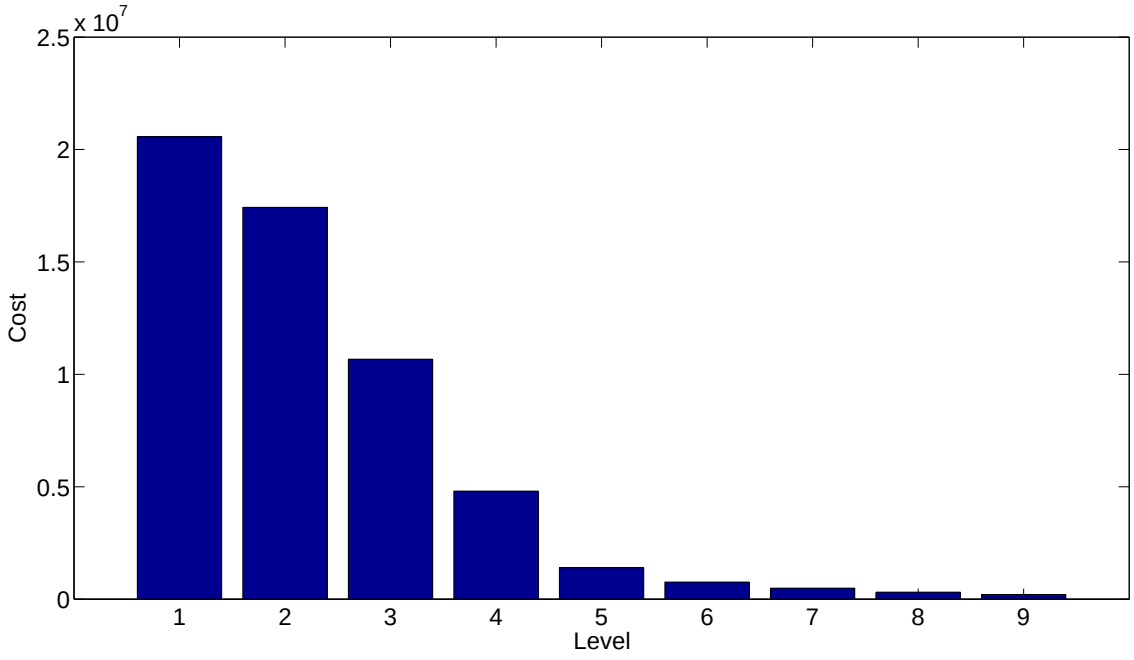


Figure 5.19: Computational Cost per each level for $\bar{L} = 9$

Note: we have decided to stop at $\bar{L} > 5$, since the standard deviation of Costs for $\bar{L} = 9$ is $7.3 \cdot 10^5$ and distinguishing between different $\bar{L}$'s would have required a higher number of experiments. We can observe that performing optimisation only on

the first 4/5 levels has marginal consequences on the benefits of the model. On the other hand, it solves an intrinsic contradiction we had not properly investigated so far: in fact, in the previous simulations, $N_\ell$ was often smaller than $N_{opt}$ on the finest levels.

### 5.2.4 Comparison between standard Importance Sampling and Multilevel Importance Sampling

In this section we will compare the performances of four methods at different levels of accuracy $\varepsilon = (5,\ 2,\ 1,\ 0.5,\ 0.2,\ 0.1) \cdot 10^{-4}$:

1. Standard Monte Carlo. Euler-Maruyama scheme.
2. Standard Importance Sampling Monte Carlo. Euler-Maruyama scheme. $(\mu_0, \tilde{\sigma})$.
3. Multilevel Monte Carlo. Milstein scheme.
4. Multilevel Importance Sampling Monte Carlo. Milstein scheme. $(\mu_0, \tilde{\sigma})$.

In figure 5.20 we have plotted Cost$\cdot \varepsilon^{-2}$ against $\varepsilon$. This choice helps showing the $O(\varepsilon^{-2})$ order of the computational cost of the Multilevel Methods. In order to compute the computational cost for Standard Monte Carlo, we have used formula (1.12). Applying a heuristic argument, we have set:

$$\frac{\nu}{N} = \frac{\varepsilon^2}{2} \qquad \text{and} \qquad WE^2 = \frac{\varepsilon^2}{2} \tag{5.7}$$

to find the values of $N$ and $M$ required to achieve the target accuracy. Computational cost has been simply obtained as Cost $= N \cdot M$. We have run simulations for Standard Monte Carlo with $N = 10^6$ in order to estimate the values of WE, WE$^{IS}$, $\nu^{IS}$ (recall section 1.1) and $\nu$.

| $M$ | WE$^{IS}$ | $\nu^{IS}$ | WE | $\nu$ |
|---|---|---|---|---|
| $2^4$ | $4.30 \cdot 10^{-3}$ | $2.3 \cdot 10^{-4}$ | $4.53 \cdot 10^{-3}$ | $1430 \cdot 10^{-4}$ |
| $2^5$ | $2.07 \cdot 10^{-3}$ | $2.05 \cdot 10^{-4}$ | $1.97 \cdot 10^{-3}$ | $1500 \cdot 10^{-4}$ |
| $2^6$ | $1.01 \cdot 10^{-3}$ | $1.95 \cdot 10^{-4}$ | $1.06 \cdot 10^{-3}$ | $1580 \cdot 10^{-4}$ |
| $2^7$ | $4.70 \cdot 10^{-4}$ | $3.2 \cdot 10^{-4}$ | $7.49 \cdot 10^{-4}$ | $1710 \cdot 10^{-4}$ |
| $2^8$ | $2.20 \cdot 10^{-4}$ | $3.4 \cdot 10^{-4}$ | $2.84 \cdot 10^{-4}$ | $1830 \cdot 10^{-4}$ |
| $2^9$ | $8.51 \cdot 10^{-5}$ | $3.55 \cdot 10^{-4}$ | $1.81 \cdot 10^{-4}$ | $1860 \cdot 10^{-4}$ |
| $2^{10}$ | $1.82 \cdot 10^{-5}$ | $3.77 \cdot 10^{-4}$ | $8.05 \cdot 10^{-5}$ | $1910 \cdot 10^{-4}$ |
| $2^{11}$ | $1.77 \cdot 10^{-5}$ | $4.26 \cdot 10^{-4}$ | $1.02 \cdot 10^{-3}$ | - |
| $2^{12}$ | $8.55 \cdot 10^{-6}$ | $4.24 \cdot 10^{-4}$ | $5.60 \cdot 10^{-5}$ | - |

We have noticed that, in the case of the Standard Monte Carlo, for $M \geq 2^9$, the magnitude of the Weak Error becomes comparable with the standard deviation of the Monte Carlo estimator, requiring then more than $N = 10^6$ samples for an accurate estimate. Due to limited computational capacity, we have opted for a linear regression based on the previous values of WE rather than direct simulation.

Figure 5.20 confirms the $O(\varepsilon^{-2})$ order of magnitude of the computational cost for Multilevel Monte Carlo. Moreover, it shows the great benefits both of Importance Sampling and Multilevel Monte Carlo, compared to Standard Monte Carlo, as well as the advantage of the combination of the two over either technique applied separately.

As we have pointed out at the end of section 5.2.1.2, the variance reduction effect of Importance Sampling for the Multilevel estimators $(\hat{P}_\ell - \hat{P}_{\ell-1})$ is more evident at the finest levels. On the contrary, the previous table showed that the variance reduction effect on the Standard Monte Carlo estimator is comparable for all values of $M$. This is the reason why Standard Importance Sampling outperforms Multilevel Monte Carlo with Importance Sampling for the lowest value of accuracy that we have set. The benefits of Multilevel Monte Carlo (both with and without Importance Sampling) in terms of computational cost reduction becomes more and more remarkable as we decrease the value of $\varepsilon$.

Note: unlike Multilevel Monte Carlo, computational cost for Standard Monte Carlo (with and without Importance Sampling) has only been worked out theoretically.

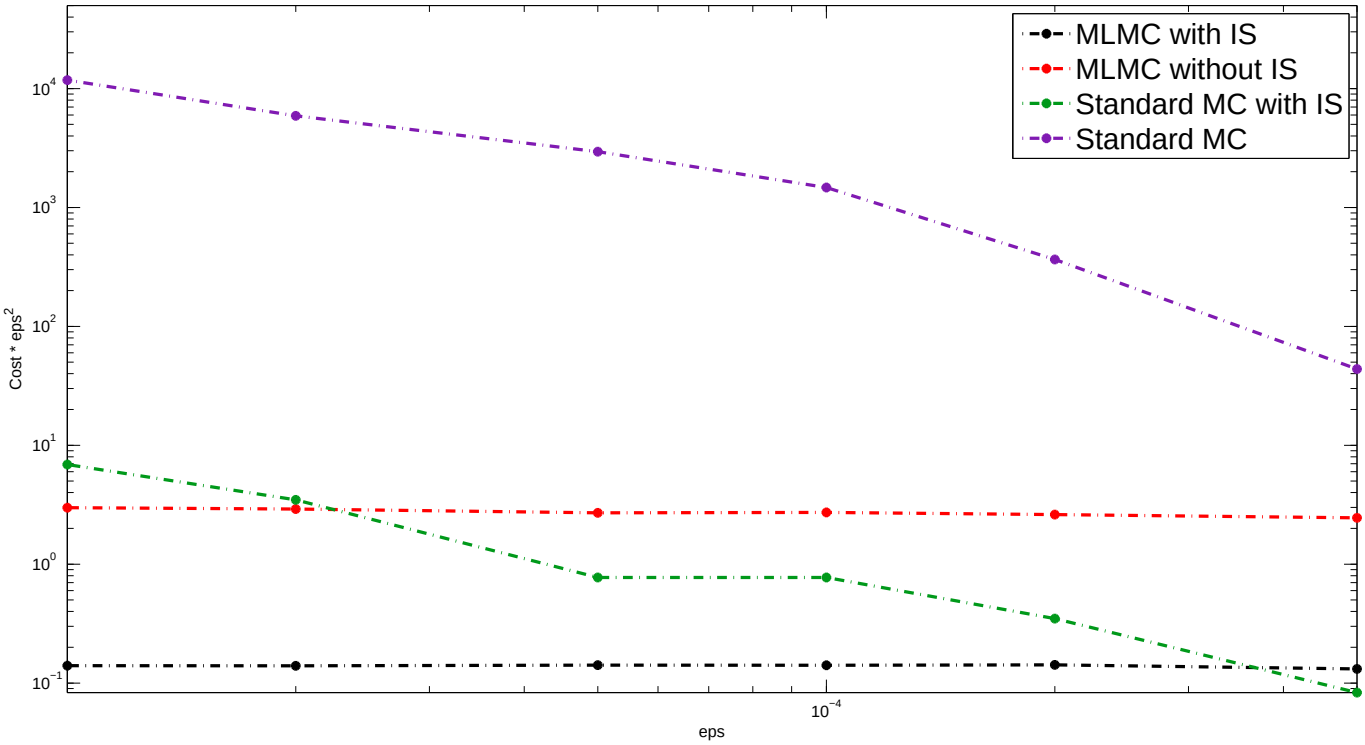


Figure 5.20: Plot of Cost·$\varepsilon^2$ vs $\varepsilon$ for Standard (purple), Multilevel (red), Standard with Importance Sampling (green) and Multilevel with Importance Sampling (black) Monte Carlo

### 5.2.5 Economised models 2 & 3: only $\mu_0$ and only $\tilde{\sigma}$

We conclude the section about the Digital Put comparing two possible economisations of the model. The former applies only a change in the variance $\tilde{\sigma}$, the latter only a change in the constant Importance Sampling drift term $\mu_0$. We have used the Milstein scheme and set target accuracy to $\varepsilon = 10^{-4}$, obtaining the following results.

Importance Sampling parameters:

| Levels | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Only $\mu_0$ | -1.77 | -1.64 | -2.04 | -2.33 | -2.62 | -2.74 | -2.75 | -2.72 |
| Only $\tilde{\sigma}$ | 1.26 | 1.49 | 1.80 | 2.61 | 2.86 | 2.60 | 2.73 | 3.98 |

As we expected, the optimal value for $\tilde{\sigma}$ is greater than 1 when the change in variance is not coupled with a change in drift. In fact, increasing the variance has the effect of driving more paths in the critical region around the strike price $K$. On the

contrary, the optimal values for $\mu_0$ are comparable to those obtained when associated with $\tilde{\sigma}$.

Key figures:

| Method | $V$ | $\hat{V}$ | WE | Cost |
|---|---|---|---|---|
| MLMC+IS, only $\mu_0$ | 0.020952 | 0.020947 | $4.02 \cdot 10^{-6}$ | $1.48 \cdot 10^{7}$ |
| MLMC+IS, only $\tilde{\sigma}$ | 0.020952 | 0.020951 | $5.94 \cdot 10^{-7}$ | $1.49 \cdot 10^{8}$ |
| MLMC+IS, $(\mu_0,\tilde{\sigma})$ | 0.020952 | 0.020889 | $6.24 \cdot 10^{-5}$ | $1.41 \cdot 10^{7}$ |
| MLMC | 0.020952 | 0.020769 | $1.83 \cdot 10^{-4}$ | $2.78 \cdot 10^{8}$ |

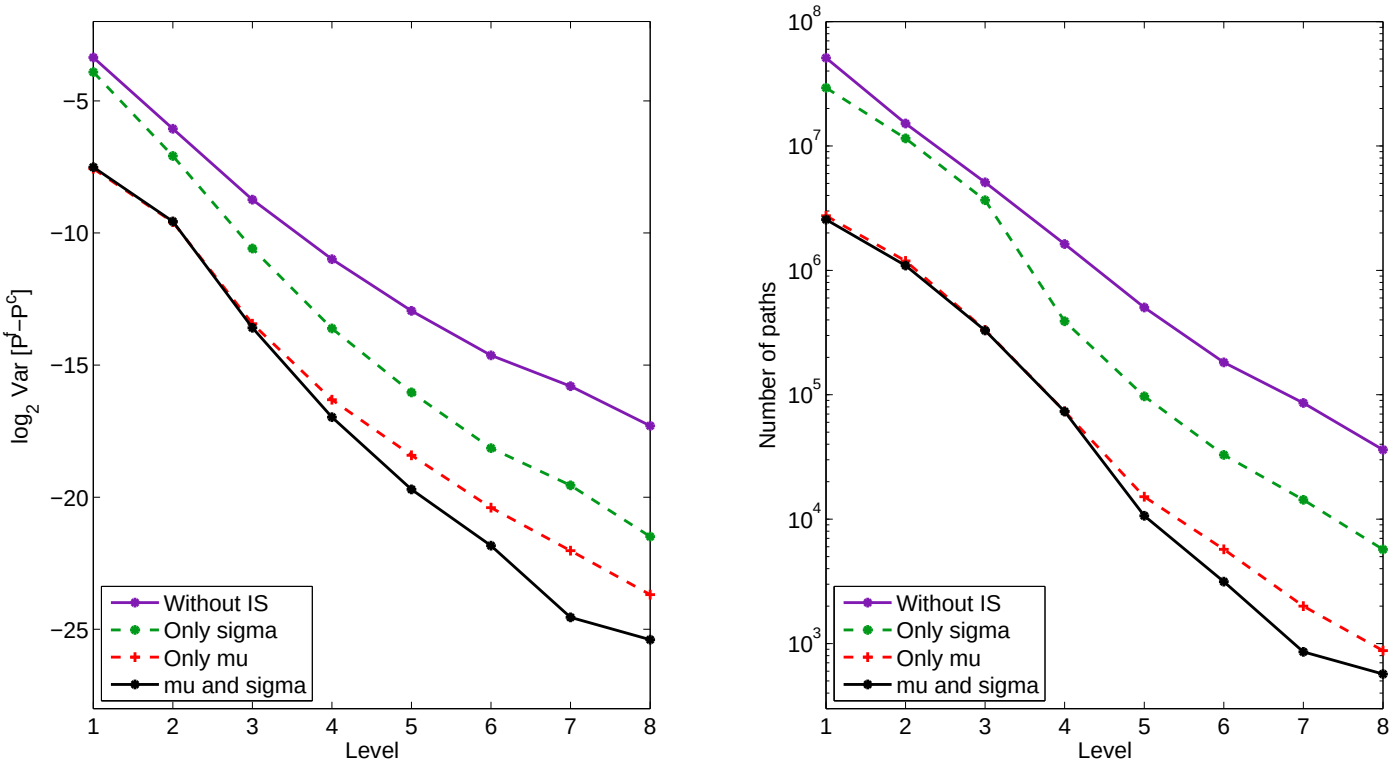


Figure 5.21: Plots of $\log_2 V_\ell$ and $N_\ell$ for Multilevel Monte Carlo without (purple) and with (only $\tilde{\sigma}$: green, only $\mu_0$: red, both: red) Importance Sampling.

Although both changes in drift and in variance have beneficial effects even when applied separately, the role of the former is more relevant. The apparently scarce advantage of choosing the full model over the one including only $\mu_0$ can be explained observing figure 5.21: the difference in $V_\ell$ between the two approaches increases with $\ell$. Therefore, the benefits of the complete models can be fully appreciated at higher levels of accuracy. Repeating the experiment with $\varepsilon = 5 \cdot 10^{-5}$ and $\varepsilon = 2.5 \cdot 10^{-5}$ required computational costs of $5.65 \cdot 10^7$ and $2.28 \cdot 10^8$ respectively for the full model and of $5.96 \cdot 10^7$ and $2.41 \cdot 10^8$ respectively using only $\mu_0$.

## 5.3 Simulations for a Down-and-In Call. $S_0 = 100,\ K = 100,\ B = 70$

In this section we have performed simulations for a Down-and-In Call with and without Importance Sampling in the Multilevel framework. We have used exclusively the Milstein scheme and we have only considered the full model, namely including ($\mu_0$, $\mu_1$, $\mu_2$, $\tilde{\sigma}$). Setting the target accuracy to $\varepsilon = 2 \cdot 10^{-4}$, we have obtained the following results:

**No Importance Sampling**

| $\text{Cost}_{ML}$ | $V$ | $\hat{V}_{ML}$ | $\text{WE}_{ML}$ |
|---|---|---|---|
| $1.36 \cdot 10^9$ | 0.016885 | 0.017047 | $1.62 \cdot 10^{-4}$ |

**Approach i.a**

| $\text{Cost}^{IS}_{ML}$ | $\text{Cost}_{ML}/\text{Cost}^{IS}_{ML}$ | $V$ | $\hat{V}^{IS}_{ML}$ | $\text{WE}^{IS}_{ML}$ |
|---|---|---|---|---|
| $9.47 \cdot 10^7$ | 14.4 | 0.016885 | 0.016863 | $2.23 \cdot 10^{-5}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $\mu_0$ | -3.24 | -5.85 | -5.35 | -4.47 | -4.12 | -3.89 | -3.89 |
| $\mu_1$ | 1.66 | 12.73 | 12.89 | 10.42 | 11.84 | 11.12 | 11.13 |
| $\mu_2$ | 4.11 | -4.56 | -4.44 | -2.06 | -4.73 | -4.09 | -4.10 |
| $\tilde{\sigma}$ | 0.49 | 0.71 | 0.56 | 0.60 | 0.60 | 0.35 | 0.38 |

**Approach ii.a**

| $\text{Cost}_{ML}^{IS}$ | $\text{Cost}_{ML}/\text{Cost}_{ML}^{IS}$ | $V$ | $\hat{V}_{ML}^{IS}$ | $\text{WE}_{ML}^{IS}$ |
|---|---|---|---|---|
| $2.04 \cdot 10^8$ | 6.7 | 0.016885 | 0.016775 | $1.11 \cdot 10^{-4}$ |

Importance Sampling parameters are:

| Level | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $\mu_0$ | -3.28 | -5.51 | -4.63 | -3.28 | -3.87 | -3.69 | -3.66 |
| $\mu_1$ | 1.67 | 12.97 | 12.75 | 7.73 | 11.54 | 10.60 | 10.53 |
| $\mu_2$ | 4.15 | -4.37 | -4.92 | -0.69 | -4.36 | -3.45 | -3.51 |
| $\tilde{\sigma}$ | 0.49 | 0.42 | 0.48 | 0.45 | 0.39 | 0.51 | 0.45 |

Plots of $N_\ell$ and $\log_2 V_\ell$ for the three methods follow:

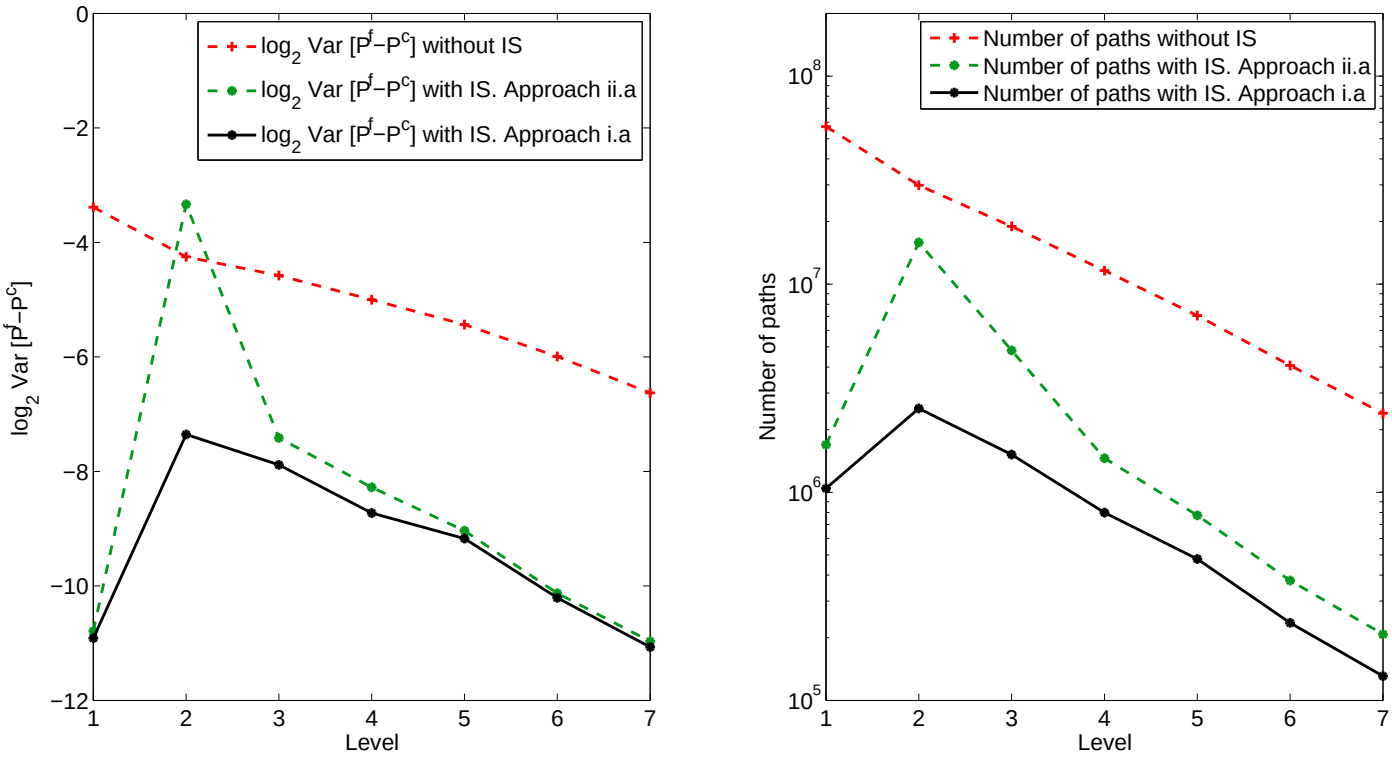


Figure 5.22: $\log_2 V_\ell$ and $N_\ell$ for Multilevel Monte Carlo, with and without Importance Sampling. Approaches i.a and ii.a

It is worth pointing out that we have decided to start from $\ell = 1$ rather than $\ell = 0$, since we are dealing with a path-dependent option. Therefore, the estimator corresponding to level 1 is a Standard Monte Carlo estimator $\hat{P}_1$.

Figure 5.22 shows that the variance reduction effect of Importance Sampling is significantly more evident for $\ell = 1$. This result is analogous to those obtained for a Digital Put, where we have observed that the magnitude of the variance reduction effect is much higher if applied to the Standard estimators $\hat{P}_\ell$ rather than to the Multilevel estimators $(\hat{P}_\ell - \hat{P}_{\ell-1})$. Moreover, both tables and plots suggest that approach i outperforms approach ii in terms of reduction of variance (especially for $\ell = 2$) and overall computational cost.

Note: the performance of the optimisation algorithm for a Down-and-In Call has been poorer than with a Digital Put. Contour plots show that the objective function requires more careful preconditioning. As a result, some of the simulation runs needed to be discarded, since the estimated Importance Sampling parameters were evidently suboptimal. We do not exclude that more refined optimisation methods may lead to further variance reduction.

### 5.3.1 Optimising only on coarser levels

In this section, we have performed an experiment similar to the one in section 5.2.3. We have set our target accuracy to be $\varepsilon = 2 \cdot 10^{-4}$ and performed 5 simulations per each value of $\bar{L}$. We will be using the same notation as in section 5.2.3.

| $\bar{L}$ | Cost | Cost/(Cost$\|_{\bar{L}=7}$) | Achieved RMSE |
|---|---|---|---|
| 7 (all) | $9.47 \cdot 10^7$ | 100% | $7.90 \cdot 10^{-5}$ |
| 6 | $1.04 \cdot 10^8$ | 110% | $6.20 \cdot 10^{-5}$ |
| 5 | $1.15 \cdot 10^8$ | 121% | $1.25 \cdot 10^{-4}$ |
| 4 | $1.04 \cdot 10^8$ | 109% | $1.80 \cdot 10^{-4}$ |
| 3 | $6.18 \cdot 10^8$ | 652% | $1.26 \cdot 10^{-4}$ |
| 2 | $4.24 \cdot 10^8$ | 447% | $1.49 \cdot 10^{-5}$ |

The results in this table suggest that setting $\bar{L} \geq 4$ allows to obtain satisfactory results. It is worth pointing out that setting $\bar{L}$ smaller than the overall number of levels is the only practically viable solution: in fact, although this approach provides suboptimal solutions, optimising at each level would often imply that $N_{opt} > N_\ell$ at the coarser levels, which is contradictory.

# Chapter 6

# Conclusions

Throughout this dissertation, we have explored the potential of a combined use of Importance Sampling and Multilevel Monte Carlo to achieve variance reduction. We have shown that this approach leads to greater benefits than either Multilevel Monte Carlo or Importance Sampling on their own. Apart from a similar application with a different approach in [Eug12], we are not aware of any other papers dealing with the combination of the two techniques.

We have tested in full detail the advantages of our method on a European option (namely, a Digital Put) and we have included results showing that remarkable variance reduction can be achieved also for path-dependent options (in our case, a Down-and-In Barrier Call).

Our approach to Importance Sampling, operating exclusively on the Brownian increments and on the normal random variables they are generated from, has been appositely designed to allow for great flexibility. In fact, it can be generalised to a wide variety of problems without needing much prior knowledge. In addition, we have presented different approaches to optimisation, which can extend the application of our model even in the case of discontinuous payoffs.

Alongside with the benefits of combining the two techniques, this dissertation has also pointed out how intrinsically delicate the implementation of Importance Sampling is and has appositely dedicated several sections to important matters that have arisen throughout the process. For instance, we have investigated the theoretical limit to a change in variance and we have proposed an alternative approach based on the Brownian Bridge construction; we have assessed the benefits of performing optimisation for $(\hat{P}_\ell - \hat{P}_{\ell-1})$ rather than for $\hat{P}_\ell$ in the Multilevel framework; we have investigated the sensitivity of our method to the choice for the penultimate step in the case of payoff smoothing.

## 6.1 Further developments

The first improvement in order to make the model we have presented applicable is to adopt a faster optimisation algorithm. Since we are dealing with a low dimensional problem, any built-in package can produce satisfactory results.

In addition, it might be worth considering different Optimisation techniques, such as the ones we have mentioned at the beginning of chapter 3.

Furthermore, it could be interesting to compare the performance of the three different versions of the objective function for our problem using a more solid optimisation algorithm.

Moreover, it could be possible to further exploit the flexibility that our model allows for: in fact, we have dealt with a SDE that might have been solved analytically. However, we have made use of the exact solution only with the purpose of making tests on the method. Therefore, our Importance Sampling Multilevel Monte Carlo approach can be applied to non-integrable SDEs, e.g. for Stochastic Volatility models.

Finally, in [GHS99] the authors have shown the benefits of combining Importance Sampling with Stratification. It might be worth investigating the advantages of this approach in the Multilevel framework.

# References


[Aro03] B. Arouna. Robbins-Monro algorithms and variance reduction in finance. *J. Comput. Finan.*, 7:1245–1255, 2003.

[BBG97] P. Boyle, M. Broadie, and P. Glasserman. Monte Carlo methods for security pricing. *Journal of Econonomic Dynamics and Control*, 21:1267–1321, 1997.

[BGLS06] J.F. Bonnans, J.C. Gilbert, C. Lemarechal, and C.A. Sagastizabal. *Numerical Optimization - Theoretical and Practical Aspects*. Springer, 2006.

[Cap08] L. Capriotti. Least-squares importance sampling for Monte Carlo security pricing. *Quantitative Finance*, 8(5):485–497, 2008.

[Eug12] T. Euget. Computing greeks with Multilevel Monte Carlo methods using importance sampling. *MSc in Mathematical and Computational Finance Thesis*, University of Oxford, 2012.

[GHS99] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Asymptotically optimal importance sampling and stratification for pricing path-dependent options. *Mathematical Finance*, 9(2):117–152, 1999.

[GI89] P. Glynn and D. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392, 1989.

[Gil08] M.B. Giles. Multilevel Monte Carlo Path Simulation. *Operations Research*, 56(3):607–617, 2008.

[Gil15a] M.B. Giles. *Monte Carlo lectures for the MSc in Mathematical and Computational Finance*. `https://people.maths.ox.ac.uk/gilesm/mc/`, Oxford University, 2014-2015.

[Gil15b] M.B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.

[Gla04] P. Glasserman. *Monte Carlo methods in Financial Engineering*. Springer, 2004.

[KM53] H. Kahn and A.W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

[KP92] P.E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1992.

[MSA03] J.R.R.A. Martins, P. Sturdza, and J.J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29:245–262, 2003.

[NW99] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 1999.

[Owe13] A. Owen. *Monte Carlo theory, methods and examples*. `http://statweb.stanford.edu/~owen/mc/`, 2013.

[OZ00] A. Owen and Y. Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

[Rei93] R. Reider. An efficient Monte Carlo technique for pricing options. *Working Paper*, Wharton School, University of Pennsylvania, 1993.

[SF00] Y. Su and M.C. Fu. Simulation in financial engineering: Importance sampling in derivative securities pricing. *Proceedings of the 32nd Conference on Winter Simulation*, pages 587–596, 2000.

[ST98] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1998.

[VAD98] F.J. Vazquez-Abad and D. Dufresne. Accelerated simulation for pricing Asian Options. *Proceedings of 1998 Winter Simulation Conference*, pages 1493–1500, 1998.

[WDH95] P. Wilmott, J. Dewynne, and S. Howison. *The Mathematics of Financial Derivatives: A Student Introduction*. Cambridge University Press, 1995.

# Appendix A

# Matlab code

This appendix includes a selection the Matlab scripts and functions that have been used to produce the results included in this dissertation.

## A.1 Approximation of $\varphi$

### A.1.1 Digital Put, Milstein scheme, approach i.a

This function approximates $\mathbb{E}_s[(\hat{P}_\ell RN_\ell - \hat{P}_{\ell-1}RN_{\ell-1})^2]$ starting from a given matrix of normal random variables $Z$. It is designed to deal with the cases of $(\mu_0, \mu_1, \mu_2, \tilde{\sigma})$, $(\mu_0, \tilde{\sigma})$, only $\mu_0$ and only $\tilde{\sigma}$.

```
function e = E2_digital_MLMC_milstein(Z,PROBparam,ISparam)

S0 = PROBparam(1); K = PROBparam(2);
r = PROBparam(3); sig = PROBparam(4);
T = PROBparam(5);
N = size(Z,2); M = size(Z,1);

P = length(ISparam);

if M<2
    error('Row dimension must be at least 2.')
end

jump = 2;
Mc = M/jump;
hf = T/M;
```

```
hc = hf*jump;
Sf_old = S0*ones(1,N);
Sf_new =  Sf_old;
Sc_new =  Sf_old;
Sc_old =  Sf_old;

switch P
case 4

a = ISparam(1); b = ISparam(2);
c = ISparam(3); sigIS = ISparam(4);

ARG_EXPc = 0.5*(-sigIS^2+1)*Z(1,:).^2;
Z(1,:) = sigIS * Z(1,:);
ARG_EXPf = ARG_EXPc;
Z = bb(Z,T);

for n = 1:Mc
    Sc_old = Sc_new;
    dWc = zeros(1,N);
    for m = 1:jump
        Sf_old = Sf_new;
        index = (n-1)*jump + m;
        rISf = a+b*(index/M)+c*(index/M)^2;
        Sf_new = Sf_old .*(1+r*hf+sig*Z(index,:)+rISf*sig*hf...
        +0.5*sig^2*((Z(index,:)+rISf*hf).^2-hf));
        ARG_EXPf = ARG_EXPf - ...
        (0.5*hf*rISf^2+rISf*Z(index,:));
        dWc = dWc + Z(index,:);
    end
    rISc = a+b*(n/Mc)+c*(n/Mc)^2;
    Sc_new = Sc_old .*(1+r*hc+sig*dWc+sig*rISc*hc...
    +0.5*sig^2*((dWc+rISc*hc).^2-hc));
    ARG_EXPc = ARG_EXPc - ...
    (0.5*hc*rISc^2+rISc*dWc);
end
```

```
last_rIS = a+b*((M-1)/M)+c*((M-1)/M)^2;

case 2

rIS = ISparam(1); sigIS = ISparam(2);

ARG_EXPc = 0.5*(-sigIS^2+1)*Z(1,:).^2;
Z(1,:) = sigIS * Z(1,:);
ARG_EXPf = ARG_EXPc;
Z = bb(Z,T);

for n = 1:Mc
    Sc_old = Sc_new;
    dWc = zeros(1,N);
    for m = 1:jump
        Sf_old = Sf_new;
        index = (n-1)*jump + m;
        Sf_new = Sf_old .*(1+r*hf+sig*Z(index,:)+rIS*sig*hf...
        +0.5*sig^2*((Z(index,:)+rIS*hf).^2-hf));
        ARG_EXPf = ARG_EXPf - ...
        (0.5*hf*rIS^2+rIS*Z(index,:));
         dWc = dWc + Z(index,:);
    end
    Sc_new = Sc_old .*(1+r*hc+sig*dWc+sig*rIS*hc...
    +0.5*sig^2*((dWc+rIS*hc).^2-hc));
    ARG_EXPc = ARG_EXPc - ...
    (0.5*hc*rIS^2+rIS*dWc);
end

last_rIS = rIS;

case 1

rIS = ISparam(1);  sigIS = 1;
```

```
ARG_EXPc = zeros(1,N);
ARG_EXPf = ARG_EXPc;
Z = bb(Z,T);

for n = 1:Mc
    Sc_old = Sc_new;
    dWc = zeros(1,N);
    for m = 1:jump
        Sf_old = Sf_new;
        index = (n-1)*jump + m;
        Sf_new = Sf_old .*(1+r*hf+sig*Z(index,:)+rIS*sig*hf...
        +0.5*sig^2*((Z(index,:)+rIS*hf).^2-hf));
        ARG_EXPf = ARG_EXPf - ...
        (0.5*hf*rIS^2+rIS*Z(index,:));
        dWc = dWc + Z(index,:);
    end
    Sc_new = Sc_old .*(1+r*hc+sig*dWc+sig*rIS*hc...
    +0.5*sig^2*((dWc+rIS*hc).^2-hc));
    ARG_EXPc = ARG_EXPc - ...
    (0.5*hc*rIS^2+rIS*dWc);
end
last_rIS = rIS;

%         Uncomment and comment the previous part to switch from
%         only drift to only variance

%         case 1
%
%         rIS = 0; sigIS = ISparam(1);
%
%         ARG_EXPc = 0.5*(-sigIS^2+1)*Z(1,:).^2;
%         Z(1,:) = sigIS * Z(1,:);
%         ARG_EXPf = ARG_EXPc;
%         Z = bb(Z,T);
%
%         for n = 1:Mc
```

```
%
%              Sc_old = Sc_new;
%              dWc = zeros(1,N);
%              for m = 1:jump
%                  Sf_old = Sf_new;
%                  index = (n-1)*jump + m;
%                  Sf_new = Sf_old .*(1+r*hf+sig*Z(index,:)...
%                      +0.5*sig^2*(Z(index,:).^2-hf));
%
%                  dWc = dWc + Z(index,:);
%              end
%              Sc_new = Sc_old .*(1+r*hc+sig*dWc...
%                  +0.5*sig^2*(dWc.^2-hc));
%          end
%      last_rIS = 0;
end

ARG_EXPf = ARG_EXPf + ...
(0.5*hf*rISf^2+rISf*Z(end,:));
ARG_EXPc = ARG_EXPc + ...
(0.5*hc*rISc^2+rISc*dWc);
ARG_EXPc = ARG_EXPc - ...
(0.5*hf*rISf^2+last_rIS*Z(end-1,:));

D = exp(-r*T);
Pf = D*Ncdf((K-Sf_old-r*hf*Sf_old)./(sig*sqrt(hf)*Sf_old));
RNf = sigIS*exp(ARG_EXPf);
dW = dWc - Z(end,:);
Pc = D*Ncdf((K-Sc_old-r*hc*Sc_old-sig*Sc_old.*...
(dW+last_rIS*hf))./(sig*sqrt(hc)*Sc_old));
RNc = sigIS*exp(ARG_EXPc);
e = sum((Pf.*RNf-Pc.*RNc).^2)/N;



end
```

### A.1.2 Down-and-In-Call, Milstein scheme, approach i.a

This function approximates $\mathbb{E}_s[(\hat{P}_\ell RN_\ell - \hat{P}_{\ell-1}RN_{\ell-1})^2]$ starting from a given matrix of normal random variables $Z$.

```
function e = E2_downandin_MLMC(Z,PROBparam,ISparam)

aIS = ISparam(1); bIS = ISparam(2);
cIS = ISparam(3); sigIS = ISparam(4);

S0 = PROBparam(1); K = PROBparam(2);
r = PROBparam(3); sig = PROBparam(4);
T = PROBparam(5); B = PROBparam(6);

N = size(Z,2); M  = 2;
nf = size(Z,1); nc = nf/M;
hf = T/nf; hc = T/nc;

sumP_IS = 0;
l = log2(nf);
Sf_new = S0*ones(1,N);
Pc = zeros(1,N);

if l == 1
    P_notcrossf = ones(1,N);
    Z(1,:) = sigIS*Z(1,:);
    RNf = sigIS*exp( - 0.5*(1-1/sigIS^2)*Z(1,:).^2);
    dWf = bb(Z,T);
    ARG_expf = zeros(1,N);
    for m = 1:nf
        Sf_old = Sf_new;
        rISf = aIS+bIS*(m/nf)+cIS*(m/nf)^2;
        Sf_new = Sf_old .*(1+r*hf+sig*dWf(m,:)+rISf*sig*hf...
        +0.5*sig^2*((dWf(m,:)+rISf*hf).^2-hf));
       P_notcrossf = P_notcrossf.*(1-exp(-2*complex_max(Sf_new-B,0)...
        .*complex_max(Sf_old-B,0)./(hf*sig^2*Sf_old.^2)));
        ARG_expf = ARG_expf - ...
```

```
        (0.5*hf*rISf^2+rISf*dWf(m,:));
    end

RNf = RNf.*exp(ARG_expf);
D = exp(-r*T);
Pf = D*complex_max(Sf_new-K,0).*(1-P_notcrossf).*RNf;

end

if l > 1
    Sc_new = Sf_new;
    P_notcrossf = ones(1,N); P_notcrossc = ones(1,N);
    Z(1,:) = sigIS*Z(1,:);
    RNf = complex_abs(sigIS)*exp( - 0.5*(1-1/sigIS^2)*Z(1,:).^2);
    RNc = RNf;
    dWf = bb(Z,T);
    ARG_expc = zeros(1,N); ARG_expf = zeros(1,N);
    for n = 1:nc
        dWc = zeros(1,N);
        Sc_old = Sc_new;
        for m = 1:M
            Sf_old = Sf_new;
            index = (n-1)*M + m;
            rISf = aIS+bIS*(index/nf)+cIS*(index/nf)^2;
            Sf_new = Sf_old .*(1+r*hf+sig*dWf(index,:)+rISf*sig*hf...
            +0.5*sig^2*((dWf(index,:)+rISf*hf).^2-hf));
            P_notcrossf = P_notcrossf.*(1-exp(-2*complex_max...
            (Sf_new-B,0).*complex_max(Sf_old-B,0)./...
            (hf*sig^2*Sf_old.^2)));
            ARG_expf = ARG_expf - ...
            (0.5*hf*rISf^2+rISf*dWf(index,:));
            dWc = dWc + dWf(index,:);
        end
        rISc = aIS+bIS*(n/nc)+cIS*(n/nc)^2;
        Sc_new = Sc_old + r*Sc_old*hc + sig*Sc_old.*dWc+sig*rISc*hc*Sc
        +0.5*sig^2*Sc_old.*((dWc+rISc*hc).^2-hc);
```

```
        ARG_expc = ARG_expc - ...
        (0.5*hc*rISc^2+rISc*dWc);
        P_notcrossc = P_notcrossc.*(1-exp(-2*complex_max...
        (Sc_new-B,0).*complex_max(Sc_old-B,0)./(hc*sig^2*Sc_old.^2)));
    end
    RNc = RNc.*exp(ARG_expc); RNf = RNf.*exp(ARG_expf);
    D = exp(-r*T);
    Pf = D*complex_max(Sf_new-K,0).*(1-P_notcrossf).*RNf;
    Pc = D*complex_max(Sc_new-K,0).*(1-P_notcrossc).*RNc;
end
sumP_IS = sumP_IS + sum((Pf-Pc).^2);
e = sumP_IS/N;
end
```

## A.2 Line Search

This function performs the Line Search described in section 3.2.3.

```
function [alpha,value] = find_alpha...
(Z,PROBparam,x0,f0,grad0,p,m1,m2,E2fcn,GRADfcn,percentage,bump)

alphaL = 0; fprime0 = (grad0'*p);
% alpha = -0.5*percentage*f0/fprime0; %safer
alpha = -percentage*f0/fprime0;

alphaR = alpha; xR = x0 + alphaR*p;
fR = feval(E2fcn,Z,PROBparam,xR);
gradR = feval(GRADfcn,Z,PROBparam,xR,bump);
fprimeR = (gradR'*p)/norm(p,2);

while fprimeR <=0
wolfe1 = (fR <= f0 + m1*alpha*fprimeR);
wolfe2 = (fprimeR >= m2*fprime0);
if wolfe1 == true && wolfe2 == false
  alphaL = alphaR;
end
if alphaR > 1e5
```

```
  alpha = +inf;  value = +inf;
  return
end
alphaR = 2*alphaR; xR = x0 + alphaR*p;
fR = feval(E2fcn,Z,PROBparam,xR);
gradR = feval(GRADfcn,Z,PROBparam,xR,bump);
fprimeR = (gradR'*p)/norm(p,2);
end

alpha = alphaL+(alphaR-alphaL)*0.5;
x1 = x0 + alpha*p; f1 = feval(E2fcn,Z,PROBparam,x1);
grad1 = feval(GRADfcn,Z,PROBparam,x1,bump);
fprime1 = (grad1'*p)/norm(p,2);
wolfe1 = (f1 <= f0 + m1*alpha*fprime0);
wolfe2 = (fprime1 >= m2*fprime0);
warning = 0;

while (~(wolfe1 == true && wolfe2 == true)) && warning < 16
if wolfe1 == true && wolfe2 == false
  alphaL = alpha;
  alpha = alphaL + 0.5*(alphaR-alphaL);
  x1 = x0 + alpha*p;
  f1 = feval(E2fcn,Z,PROBparam,x1);
  grad1 = feval(GRADfcn,Z,PROBparam,x1,bump);
  fprime1 = (grad1'*p)/norm(p,2);
  wolfe1 = (f1 <= f0 + m1*alpha*fprime0);
  wolfe2 = (fprime1 >= m2*fprime0);
elseif wolfe1 == false
  alphaR = alpha;
  alpha = alphaL + 0.5*(alphaR-alphaL);
  x1 = x0 + alpha*p;
  f1 = feval(E2fcn,Z,PROBparam,x1);
  grad1 = feval(GRADfcn,Z,PROBparam,x1,bump);
  fprime1 = (grad1'*p)/norm(p,2);
  wolfe1 = (f1 <= f0 + m1*alpha*fprime0);
  wolfe2 = (fprime1 >= m2*fprime0);
```

```
end
warning = warning+1;
end
value = f1;
if warning >=16
  alpha = +inf;  value = +inf;
end
end
```

## A.3 Multilevel Monte Carlo simulation. Digital Put. Milstein Scheme

The following function is used to perform a Multilevel Monte Carlo simulation. The target accuracy needs to be specified as an input for the nested function `mlmc_penultimate`. This function relies on three underlying routines: `mlmc_penultimate` (which performs the actual Multilevel Monte Carlo simulation), `gbm_l` (which computes estimators for $\hat{P}_\ell$, $(\hat{P}_\ell - \hat{P}_{\ell-1})$ and their squares for a single specified level) and the optimisation function `optim`.

```
function [P,Nl,var,elapsed,cost,rIS,sigIS]=...
MLMC_digital_optimincluded_milstein

tic

global S0 K T r sig insurance

S0  = 100; K   = 50; T   = 1;
r   = 0.05; sig = 0.25; insurance = 10;
[P, Nl,var,rIS,sigIS] = mlmcvar_optimincluded(1000,TARGET_ACCURACY,...
@gbm_l, 1,-1,1,@optim);
maxL = length(Nl); cost = sum((2.^(0:(maxL-1))).*Nl);
elapsed=toc;
end

function [sum1, sum2] = gbm_l(l,N,rIS,sigIS)
```

```
global S0 K T r sig insurance
M  = 2; nf = M^l; nc = nf/M;
hf = T/nf; hc = T/nc;
sum1(1:4) = 0; sum2(1:2) = 0;

for N1 = 1:10000:N

N2 = min(10000,N-N1+1);
Sf_old = S0*ones(1,N2); Sf_new = Sf_old;
Sc_old = Sf_old; Sc_new = Sc_old;
Pc = zeros(1,N2);

if l == 0
RNf = ones(1,N2);
end

if l>0
dWf = randn(nf,N2);
dWf(1,:) = sigIS*dWf(1,:);
RNf = sigIS*exp( - 0.5*(1-1/sigIS^2)*dWf(1,:).^2);
RNc = RNf; dWf = bb(dWf,T);
ARG_expc = zeros(1,N2); ARG_expf = zeros(1,N2);

for n = 1:nc

dWc = zeros(1,N2); Sc_old = Sc_new;

for m = 1:M

Sf_old = Sf_new;
index = (n-1)*M + m;
dWc = dWc + dWf(index,:);
Sf_new = Sf_old .*(1+r*hf+sig*dWf(index,:)+rIS*sig*hf...
+0.5*sig^2*((dWf(index,:)+rIS*hf).^2-hf));
ARG_expf = ARG_expf - ...
```

```
(0.5*hf*rIS^2+rIS*dWf(index,:));
end

Sc_new = Sc_old + r*Sc_old*hc + sig*Sc_old.*dWc+sig*rIS*hc*Sc_old...
+0.5*sig^2*Sc_old.*((dWc+rIS*hc).^2-hc);
ARG_expc = ARG_expc - ...
(0.5*hc*rIS^2+rIS*dWc);
end

ARG_expf = ARG_expf + ...
(0.5*hf*rIS^2+rIS*dWf(end,:));
ARG_expc = ARG_expc + ...
(0.5*hc*rIS^2+rIS*dWc);
ARG_expc = ARG_expc - ...
(0.5*hf*rIS^2+rIS*dWf(end-1,:));
RNc = RNc.*exp(ARG_expc); RNf = RNf.*exp(ARG_expf);
end

Pf = exp(-r*T)*insurance*...
Ncdf((K-Sf_old-r*hf*Sf_old)./(sig*sqrt(hf)*Sf_old)).*RNf;
if (l>0)
dW = dWc-dWf(end,:);
Pc = exp(-r*T)*insurance*Ncdf((K-Sc_old-r*hc*Sc_old-sig*Sc_old.*...
(dW+rIS*hf))./(sig*sqrt(hf)*Sc_old)).*RNc;
end
sum1(1) = sum1(1) + sum(Pf-Pc);
sum1(2) = sum1(2) + sum((Pf-Pc).^2);
sum1(3) = sum1(3) + sum((Pf-Pc).^3);
sum1(4) = sum1(4) + sum((Pf-Pc).^4);
sum2(1) = sum2(1) + sum(Pf);
sum2(2) = sum2(2) + sum(Pf.^2);
end end


function [rIS,sigIS] = optim(l,rIS,sigIS,tol)
global S0 K T r sig
```

```
Nopt = 10000;
E2fcn = @E2_digital_MLMC_milstein;
GRADfcn = @grad_CVT_digital_MLMC_milstein;
bump = 1e-10;   %bump to be used in the CVT approximation
first_percentage = 0.01;
m1G = 1e-4; m2G = 0.8;
% Parameters for the Wolfe conditions

PROBparam(1)= S0; PROBparam(2) = K; PROBparam(3) = r;
PROBparam(4) = sig; PROBparam(5) = T;

x0 = [rIS;sigIS];
Z = randn(2^l,Nopt);
f0 = feval(E2fcn,Z,PROBparam,x0);
grad0 = feval(GRADfcn,Z,PROBparam,x0,bump);
p = -grad0; p = p/norm(p,2);

alpha = find_alpha(Z,PROBparam,...
x0,f0,grad0,p,m1G,m2G,E2fcn,GRADfcn,first_percentage,bump);
x1 = x0 + alpha*p;
f1 = feval(E2fcn,Z,PROBparam,x1);
grad1 = feval(GRADfcn,Z,PROBparam,x1,bump);
H0 = eye(length(x0)); I = H0;
tol = tol*norm(grad0,2);

while norm(grad1,2)>tol
s = x1-x0; y = grad1 - grad0;
rho = 1/(y'*s);
H1 = (I-rho*s*y')*H0*(I-rho*y*s')+rho*s*s';
pBFGS = - H1*grad1; pBFGS = pBFGS/norm(pBFGS,2);
[alphaBFGS,valueBFGS] = find_alpha(Z,PROBparam,...
x1,f1,grad1,pBFGS,m1G,m2G,E2fcn,GRADfcn,abs(f1-f0)/f0,bump);
pG = -grad1; pG = pG/norm(pG,2);
[alphaG,valueG] = find_alpha(Z,PROBparam,...
x1,f1,grad1,pG,m1G,m2G,E2fcn,GRADfcn,abs(f1-f0)/f0,bump);
provBFGS = x1+alphaBFGS*pBFGS; provG = x1+alphaG*pG;
```

```
if provBFGS(end) < 0.01
valueBFGS = +inf;
end
if provG(end) < 0.01
valueG = +inf;
end
if valueBFGS<valueG
x2 = x1+alphaBFGS*pBFGS;
elseif valueBFGS>=valueG
x2 = x1+alphaG*pG;
end
x0 = x1; x1 = x2; f0 = f1;
f1 = feval(E2fcn,Z,PROBparam,x1);
grad0 = grad1; grad1 = feval(GRADfcn,Z,PROBparam,x1,bump);
H0 = H1;
end

rIS = x1(1);
sigIS = x1(2);
end
```

### A.3.1 mlmc_optimincluded

This function is an adapted version of the original one provided by Prof. Mike Giles for the paper [Gil15b] (it can be found at `https://people.maths.ox.ac.uk/gilesm/acta/mlmc.m`). We simply added a few lines to perform optimisation (function `optim`) every time a new level is added (possibly up to a certain level $\bar{L}$). Moreover, we adapted the function so that it returns the estimated optimal parameters for Importance Sampling and the estimated values of $V_\ell$. We will omit the implementation of the function.

# Appendix B

# Light tails

Throughout the whole dissertation, we have assessed in several ways the benefits of sampling from the Importance Sampling distribution. A common feature, encompassing Standard and Multilevel, Down-and-In Call and Digital Put, is the lightness of the tails. In fact, $\tilde{\sigma}$ is smaller than 1 in the vast majority of the examples we have presented.

However, it is well known that sampling from a distributions with lighter tails often produces catastrophic variance blow-ups: in fact, since the original and the Importance Sampling distribution are involved in computations as a ratio, different asymptotic behaviours can induce integrals to diverge. To be more specific, we have provided an example, inspired to the one exposed in [Owe13]. Let us consider the case when the original measure is a standard normal distribution and the Importance Sampling measure is normal with variance $\tilde{\sigma}^2$. The resulting Radon-Nikodym derivative is:

$$\frac{s(z)}{q(z)} = \exp\left\{\left(\frac{1}{2\tilde{\sigma}^2} - \frac{1}{2}\right) z^2\right\} \tag{B.1}$$

Therefore, its second moment under the Importance Sampling measure is:

$$\mathbb{E}_s\left[\left(\frac{s(Z)}{q(Z)}\right)^2\right] = \frac{1}{\sqrt{2\pi\tilde{\sigma}^2}} \int_{\mathbb{R}} \exp\left\{\left(\frac{1}{2\tilde{\sigma}^2} - 1\right) z^2\right\} dz \tag{B.2}$$

The integral in (B.2) converges only if $\tilde{\sigma} > 1/\sqrt{2}$.

The change of measure we have just described is the same as the one involved in our Brownian Bridge construction. Nevertheless we have have often dealt with values

of $\tilde{\sigma}$ consistently smaller than $1/\sqrt{2}$. We argue that in our estimators the Radon-Nikodym derivative is always multiplied by the payoff and that extreme values of $Z$, which would produce a variance blow-up in the likelihood ratio, result in zero payoff.

Throughout the dissertation, the only case when choosing small values of $\tilde{\sigma}$ has raised our concerns happened when experimenting optimisation using the objective function described in section 3.1.1. When $\tilde{\sigma}$ was set to be smaller than $1/\sqrt{2}$, either in $s$ or in $s^\star$, the function $\varphi_{s^\star}$ began to show inconsistencies. Specifically, it started exhibiting an unexpected dependence on $x_n$.

We address to further analysis the theoretical explanation of the results we have produced with light tail distributions and a numerical analysis in the case of different payoffs (e.g. a European call).