Comparaison between the regular Multilevel estimator and Multilevel Richardson Romberg estimator

Oussama Elghodhben

A report submitted as part of the requirements for the degree
of M2 Probability and Finance
at Ecole Polytechnique & Sorbonne University
Paris, France

May 2024

Supervisor Pr. Gilles Pagès and Pr. Vincent Lemaire

Abstract

This report provides a comparative analysis of two advanced computational techniques in financial modeling: the Multilevel Monte Carlo (MLMC) method and the Multilevel Richardson-Romberg (ML2R) extrapolation. By exploring their application in pricing various financial derivatives under the framework of Geometric Brownian Motion, this study assesses their computational efficiency and accuracy. MLMC, known for its capability to reduce computational costs through a multi-tiered sampling approach, and ML2R, which integrates Richardson-Romberg extrapolation to enhance error reduction, are evaluated across different financial products. The analysis aims to determine which if the Multilevel Richardson Romberg Extrapolation offers superior performance in terms of accuracy and computational expense in comparaison to the regular Multilevel approach.

Chapter 1

Introduction

In the field of computational finance, the accuracy and efficiency of numerical methods directly influence the reliability and feasibility of financial assessments and forecasts. Traditional Monte Carlo simulations, while robust, often incur significant computational costs when high precision is required. The Multilevel Monte Carlo (MLMC) method, introduced by Mike Giles in 2008, presents a strategic innovation to address these challenges by employing a hierarchical sampling technique that significantly reduces computational overhead. Building upon this framework, the Multilevel Richardson-Romberg (ML2R) method offers a further refinement by incorporating Richardson extrapolation, targeting enhanced reduction of higher-order errors.

This report delves into a detailed comparison of these two methodologies, focusing on their application in the pricing of various financial derivatives, including standard and more complex options under Geometric Brownian Motion. Through a combination of theoretical analysis and empirical testing, the study evaluates the performance of MLMC and ML2R in terms of computational speed, cost-efficiency, and error minimization. The outcomes of this comparison are intended to guide financial analysts and researchers in choosing the most effective computational strategy for their specific modeling needs, ensuring that both precision and efficiency are optimally balanced. The findings also aim to contribute to the broader field of financial mathematics by clarifying the practical implications of adopting advanced Monte Carlo methods in real-world financial modeling scenarios.

1.1 Background

The Multilevel Monte Carlo (MLMC) methodology represents a substantial evolution in computational methods, specifically tailored to enhance the traditional Monte Carlo

simulations through methodical mathematical improvements. Each development addresses practical needs across various applications, providing nuanced solutions to complex computational challenges.

The core concept of MLMC, introduced by Mike Giles in 2008, is designed to optimize the efficiency of simulations by incorporating multiple levels of accuracy. This is elegantly captured by the formula:

$$E[P] \approx \sum_{\ell=0}^{L} \left(E[P_\ell] - E[P_{\ell-1}] \right),$$

where P_ℓ denotes the approximation at level ℓ, with $P_{-1} \equiv 0$. This method enhances accuracy by focusing on error correction at each level, efficiently minimizing redundant computations and thereby saving significant resources (*Giles, 2008*).

Introduced by Rhee and Glynn in 2015, Randomized MLMC adds stochastic control over the level selection, enhancing the flexibility of MLMC. It adapts to the unpredictable environments where computational requirements vary dynamically. The randomized MLMC estimator is given by:

$$Y = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{p_{\ell(n)}} \left(Q_{\ell(n)} - Q_{\ell(n)-1} \right),$$

where $\ell(n)$ is a level selected randomly for each sample. This method dynamically balances computational load, making it ideal for real-time data analysis and adaptive simulations (*Rhee & Glynn, 2015*).

The Multi-Index Monte Carlo (MIMC) method, proposed by Haji-Ali et al. in 2016, extends MLMC to address high-dimensional problems where errors originate from various sources. It reduces computational demands significantly while maintaining accuracy across multiple dimensions:

$$E[P] \approx \sum_{\ell \in \mathcal{L}} \Delta Q_\ell,$$

where ΔQ_ℓ signifies multi-dimensional differentials across the index set $\mathcal{L}$. This method is particularly beneficial for applications such as fluid dynamics and financial modeling (*Haji-Ali et al., 2016*).

Non-Geometric MLMC customizes the refinement strategy to match the specific characteristics of problems, optimizing computational grids based on the peculiarities of each application. This ensures that computational efforts are specifically tailored, enhancing both the efficiency and accuracy of simulations.

The Multilevel Richardson-Romberg (ML2R) method, detailed by Lemaire and Pagès in 2016, integrates Richardson extrapolation into MLMC to further reduce biases in estimates. This approach significantly improves the precision of calculations, crucial in areas such as financial derivatives pricing:

$$E[P] \approx \sum_{\ell=0}^{L} w_\ell E[P_\ell],$$

where w_ℓ are specifically designed weights to minimize higher-order biases, providing highly accurate and reliable estimates (*Lemaire & Pagès, 2016*).

1.2 Chapter List

Chapter 2 Theory. In this chapter, we focus on explaining the theoritical results for both estimatros.

Chapter 3 Methodology.In This chapter we describe our methodology for exploring the practical results in relation to the theory.

Chapter 4 Results and Comparaison. In this chapter we discuss our finding and compare them to the results of the original papers.

Chapter 5 Conclusion

Chapter 2

Thoery

2.1 Multilevel Complexity Theorem

The Multilevel Monte Carlo (MLMC) method provides a significant enhancement in computational efficiency by integrating estimates from multiple accuracy levels. The core of its effectiveness is encapsulated in the MLMC complexity theorem, which rigorously quantifies how this approach balances computational cost against achieving desired accuracy.

The theorem is formally stated as follows: Consider a sequence of approximations P_ℓ to a quantity P, where each level ℓ contributes an estimator with associated variance V_ℓ and cost C_ℓ. Suppose there exist constants $\alpha, \beta, \gamma > 0$ and $c_1, c_2, c_3 > 0$ such that the following conditions are met:

1. The bias at each level ℓ is bounded by $|E[P_\ell - P]| \leq c_1 2^{-\alpha \ell}$.
2. The variance at each level ℓ is bounded by $V_\ell \leq c_2 2^{-\beta \ell}$.
3. The computational cost at each level ℓ is bounded by $C_\ell \leq c_3 2^{\gamma \ell}$.

Then, for any $\epsilon < e^{-1}$, the total computational cost C to achieve a mean square error (MSE) less than ϵ^2 is given by:

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

where c_4 is a constant depending on c_1, c_2, and c_3, but not on ϵ.

This theorem is critical for guiding the optimal allocation of computational resources across different levels of simulation. It suggests allocating more resources to levels where the variance reduction per computational unit is highest. In practice, this often means conducting more simulations at coarser levels and fewer at finer levels, optimizing the total computational expenditure while meeting accuracy requirements.

For practical implementation, a practitioner would start by estimating the values of α, β, and γ through preliminary simulations to characterize the behavior of bias, variance, and cost as functions of ℓ. These parameters then guide the setup of the MLMC algorithm to ensure that the total cost is optimized while adhering to the specified error threshold.

The optimization problem in the heart of Multilevel Monte Carlo (MLMC) method focuses on determining the optimal number of simulations N_ℓ at each level ℓ to minimize the total computational cost while achieving a specified accuracy ϵ. This problem is crucial for ensuring the efficiency of the MLMC method, balancing computational resources effectively across different levels.

The objective of the optimization problem is to allocate the number of samples N_ℓ such that the total variance of the MLMC estimator is minimized, under the constraint that the total cost does not exceed what is necessary to achieve an error smaller than ϵ^2. The formula for N_ℓ is derived by equating the cost-efficiency of variance reduction across all levels:

$$N_\ell = \left\lceil 2\epsilon^{-2} \sqrt{V_\ell / C_\ell} \sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell} \right\rceil$$

Here, $\sqrt{V_\ell / C_\ell}$ represents the efficiency of variance reduction per unit of cost at level ℓ. The sum $\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell}$ acts as a normalization factor, ensuring that resources are allocated proportionally to the capacity of each level to efficiently reduce variance.

This optimization ensures that each level ℓ is allocated a number of samples in proportion to its effectiveness in reducing the overall variance of the estimator for each unit of computational cost spent. This strategy allows for more samples to be allocated to levels where they are most cost-effective, balancing the trade-off between computational expenditure and statistical efficiency. Such an approach is essential in applications such as financial modeling where accuracy and computational overhead are critical.

Implementation and Convergence Testing of the MLMC Algorithm

The Multilevel Monte Carlo (MLMC) method strategically utilizes a dynamic algorithm that adjusts the number of simulation samples across different levels to optimize the balance between computational cost and accuracy. This implementation is essential for minimizing the computational effort while maintaining the desired accuracy levels.

The MLMC algorithm begins with an initial setup and iteratively adjusts based on convergence criteria:

```
Algorithm 1: MLMC Algorithm
1. Start with L = 2, and an initial target of N0 samples on levels l = 0, 1, 2
2. While extra samples are needed:
    a. Evaluate extra samples on each level
    b. Compute/update estimates for Vl, l = 0, ..., L
    c. Define optimal Nl, l = 0, ..., L
    d. Test for weak convergence
    e. If not converged, set L := L + 1, and initialize target NL
```

The optimal number of samples for each level, N_ℓ, is determined by the formula:

$$N_\ell = \left\lceil 2\epsilon^{-2}\sqrt{V_\ell/C_\ell}\sum_{\ell=0}^{L}\sqrt{V_\ell C_\ell}\right\rceil$$

This formula optimizes variance reduction per unit of computational cost, ensuring efficiency in resource allocation.

A critical component of the MLMC algorithm is its convergence test, which is applied to determine when the simulation has adequately converged to halt further computations. The convergence criterion is:

$$\left|E[P_L - P_{L-1}]\right| / (2^\alpha - 1) < \frac{\epsilon}{\sqrt{2}}$$

where α represents the rate of bias decay, and ϵ is the desired accuracy. This test ensures the mean square error (MSE) of the estimator remains within acceptable bounds.

During each iteration, after updating estimates and recalculating the optimal number of samples, the algorithm assesses convergence using the maximum of the estimated differences from the last three highest levels extrapolated to the last level:

$$\max\left|E[P_L - P_{L-1}], E[P_{L-1} - P_{L-2}]2^\alpha, E[P_{L-2} - P_{L-3}]2^{2\alpha}\right|$$

This robustness check mitigates the impact of random fluctuations that might falsely indicate convergence.

2.2 ML2R Complexity Theorem

Theoretical Explanation and Complexity of the Multilevel Richardson-Romberg (ML2R) Method

The Multilevel Richardson-Romberg (ML2R) method integrates Richardson-Romberg extrapolation with Multilevel Monte Carlo (MLMC) techniques to optimize the accuracy and computational efficiency of stochastic simulations. This method is especially useful in contexts where direct simulations are prohibitively expensive.

Core Concepts:

- *Bias Decomposition*: Assumes a systematic expansion of the bias $E[Y_h] - E[Y_0]$ in powers of the discretization step h, enabling higher-order bias corrections.
- *Variance Reduction*: Utilizes differences between consecutive discretization levels to reduce variance, lessening the number of high-resolution simulations needed.
- *Richardson-Romberg Extrapolation*: Uses the expansion of the bias error to eliminate higher-order terms systematically, increasing accuracy.
- *MLMC Methodology*: Averages differences across multiple discretization levels to significantly reduce variance with fewer fine-resolution simulations.

Mathematical Formulation: The ML2R estimator is given by:

$$I_{h,R,q}^N = \frac{1}{N_1} \sum_{k=1}^{N_1} Y_{h_1}^{(k)} + \sum_{j=2}^{R} W_j \left(\frac{1}{N_j} \sum_{k=1}^{N_j} (Y_{h_j}^{(k)} - Y_{h_{j-1}}^{(k)}) \right)$$

where R is the number of levels, N_j is the number of simulations at level j, W_j are weights designed to optimize bias reduction, and h_j are the discretization steps, typically $h_j = M^{-j+1}h$ for some $M > 1$.

Complexity Analysis: The complexity of the ML2R estimator improves as the target error ϵ decreases. The complexity for achieving a root mean squared error (RMSE) ϵ is:

$$\text{Cost(ML2R)} = \begin{cases} \epsilon^{-2} & \text{if } \beta > 1, \\ \epsilon^{-2} \log(1/\epsilon) & \text{if } \beta = 1, \\ \epsilon^{-2} e^{-\frac{1-\beta}{\sqrt{\alpha}} \sqrt{2log(1/\epsilon)log(M)}} & \text{if } \beta < 1. \end{cases}$$

where β is the strong convergence rate parameter $\|Y_h - Y_0\|_2 = O(h^\beta)$.

2.3 Comparative efficiency

For $\beta < 1$: ML2R exhibits significant superiority over MLMC, with efficiency increasing exponentially as $\epsilon \to 0$. The improvement factor is given by:

$$\epsilon^{-\frac{1}{\sqrt{-\beta\alpha}}} \exp\left(-\frac{1-\beta}{\alpha\sqrt{2\log(M)}}\log(1/\epsilon)\right),$$

which becomes greater than 1 when $\epsilon \leq 2^{-\frac{2}{\alpha}}$.

For $\beta = 1$: ML2R outperforms MLMC by a factor of $\log(1/\epsilon)$, showcasing its enhanced performance as precision requirements increase.

For $\beta > 1$: Both estimators achieve a convergence rate of ϵ^{-2}, akin to an unbiased Monte Carlo simulation. However, numerical experiments suggest that ML2R has a lower constant in its error term compared to MLMC, indicating better computational efficiency even when theoretical convergence rates are identical.

Chapter 3

Methodology

We will first start by studying the performance of Multilevel Monte Carlo independently of the Multilevel Richardson Romberg Extrapolation. In fact, It is important to first understand the improvement that MLMC brings over the biased montecarlo and show that numerically. We will then move to the comparaison between the regular MLMC and the Richardson Romberg Extension Where we verify the theoritical results of the complexity theorems by applying both estimators on 3 cases where $\beta > 1$, $\beta = 1$ and $\beta < 1$

3.1 Multilevel Monte Carlo

In this section we implement the MLMC estimator using the Adaptive approach proposed by giles (2008) and improved in giles (2015). The pseudo-algorithm is as follows :

```
Algorithm 1: MLMC Algorithm
1. Start with L = 2, and an initial target of N0 samples on levels l = 0, 1, 2
2. While extra samples are needed:
    a. Evaluate extra samples on each level
    b. Compute/update estimates for Vl, l = 0, ..., L
    c. Define optimal Nl, l = 0, ..., L
    d. Test for weak convergence
    e. If not converged, set L := L + 1, and initialize target NL
```

The optimal number of samples for each level, N_ℓ, is determined by the formula:

$$N_\ell = \left\lceil 2\epsilon^{-2}\sqrt{V_\ell/C_\ell}\sum_{\ell=0}^{L}\sqrt{V_\ell C_\ell}\right\rceil$$

The convergence test is as we mentioned before

$$|E[P_L - P_{L-1}]| / (2^\alpha - 1) < \frac{\epsilon}{\sqrt{2}}$$

which is extend, for more robustness, to the maximum of the estimated differences from the last three highest levels extrapolated to the last level:

$$\max \left| E[P_L - P_{L-1}], E[P_{L-1} - P_{L-2}]2^\alpha, E[P_{L-2} - P_{L-3}]2^{2\alpha} \right|$$

. To translate this into code, we adopt a similar structure to that described in Giles (2015). The whole code is written in python in the format of a jupyter notebook.
We develop a problem independent structure throught fucntions:

- `mlmc` this function is the main driver for the algorithm, it iterates until the convergence test is satisfied. It starts with a number intial smaples and 3 levels and at each iteration estimates the number of added samples and the parameters alpha, beta and gamma if not provided. then it simulates the needed additional samples using `eval_fn` and checks the convergence test.If not converged a level is added and we iterate again. at the end it returns the quantity of interest estimation, the cost per level and the number of samples per level

- `eval_fn` this function si a simple driver to samples from the correct problem class and 3 first order moments of the samples, that serve for estimating kurtosis and consistency checking as well as estimating parameters, and the cost of simulation

- `mlmc_test` the function serves as a testing routine that estimates the parameters calculates the kurtosis per level and the several other statistics. It also simulates the `mlmc` function for differnet ϵ and return all the data for plotting.

- `mlmc_plot` uses the results from the testing routine and produces 6 convergence including weak and strong error decay, consistency check, kurtosis per level, number of samples per level for different RMSE and cost of simulation vs ϵ. which are the same plots used in giles (2015)

Then each problem is represented by a class that implement and `p_eval` method which takes as parameteres the level and the number of samples and returns the coupled estimator $Y_l = P_l^f - P_{l-1}^c$ and it's moments along with the simulation cost represented by the processor time.

3.2 MLMC vs ML2R

The second part is implementing MLMC and ML2R as described in Pages and Lemaire (2018). This implementation is not adaptive as before as it relies on first estimating the structural parameters by simulation then calculating R, h, q_j and N using the explicit expression that result from the resolution of the optimization problem. The explicit expressions are reported in the tables below.

$R(\varepsilon)$	$\left\lceil \frac{1}{2} + \frac{\log\left(\widetilde{c}^{\frac{1}{\alpha}}\mathbf{h}\right)}{\log(M)} + \sqrt{\left(\frac{1}{2} + \frac{\log\left(\widetilde{c}^{\frac{1}{\alpha}}\mathbf{h}\right)}{\log(M)}\right)^2 + 2\frac{\log(A/\varepsilon)}{\alpha\log(M)}} \right\rceil, \quad A = \sqrt{1+4\alpha}$
$h(\varepsilon)$	$\mathbf{h}/\left\lceil (1+2\alpha R)^{-\frac{1}{2\alpha R}} \varepsilon^{\frac{1}{\alpha R}} \widetilde{c}^{-\frac{1}{\alpha}} M^{\frac{R-1}{2}} \mathbf{h} \right\rceil$
$q(\varepsilon)$	$q_1 = \mu_R^*(1+\theta h^{\frac{\beta}{2}})$ $q_j = \mu_R^* \theta h^{\frac{\beta}{2}} \left(\lvert \mathbf{W}_j(R,M) \rvert \frac{n_{j-1}^{-\frac{\beta}{2}} + n_j^{-\frac{\beta}{2}}}{\sqrt{n_{j-1}+n_j}} \right),\ j = 2,\dots,R;\ \mu_R^*\ s.t. \sum_{1 \leqslant j \leqslant R} q_j = 1$
$N(\varepsilon)$	$\left(1 + \frac{1}{2\alpha R}\right) \frac{\mathrm{var}(Y_0) \left(1 + \theta h^{\frac{\beta}{2}} \sum_{j=1}^{R} \lvert \mathbf{W}_j(R,M) \rvert \left(n_{j-1}^{-\frac{\beta}{2}} + n_j^{-\frac{\beta}{2}} \right) \sqrt{n_{j-1}+n_j} \right)}{\varepsilon^2 \mu_R^*}$

Figure 3.1: Optimal parameters for the ML2R estimator (standard case).

$R(\varepsilon)$	$\left\lceil 1 + \frac{\log\left(\lvert c_1 \rvert^{\frac{1}{\alpha}}\mathbf{h}\right)}{\log(M)} + \frac{\log(A/\varepsilon)}{\alpha\log(M)} \right\rceil, \quad A = \sqrt{1+2\alpha}$
$h(\varepsilon)$	$\mathbf{h}/\left\lceil (1+2\alpha)^{-\frac{1}{2\alpha}} \varepsilon^{\frac{1}{\alpha}} \lvert c_1 \rvert^{-\frac{1}{\alpha}} M^{R-1} \mathbf{h} \right\rceil$
$q(\varepsilon)$	$q_1 = \mu_R^*(1+\theta h^{\frac{\beta}{2}})$ $q_j = \mu_R^* \theta h^{\frac{\beta}{2}} \left(\frac{n_{j-1}^{-\frac{\beta}{2}} + n_j^{-\frac{\beta}{2}}}{\sqrt{n_{j-1}+n_j}} \right),\ j = 2,\dots,R;\ \mu_R^*\ s.t. \sum_{1 \leqslant j \leqslant R} q_j = 1$
$N(\varepsilon)$	$\left(1 + \frac{1}{2\alpha}\right) \frac{\mathrm{var}(Y_0) \left(1 + \theta h^{\frac{\beta}{2}} \sum_{j=1}^{R} \left(n_{j-1}^{-\frac{\beta}{2}} + n_j^{-\frac{\beta}{2}} \right) \sqrt{n_{j-1}+n_j} \right)}{\varepsilon^2 \mu_R^*}$

Figure 3.2: Optimal parameters for the MLMC estimator (standard case).

The in the implementation we use a similar structure as before with a problem independent structure coupled with a class representatino for each problem that impelement a problem interface (not included with the code for sake of simplicity).

- `ML2R` and `MLMC` driver functions that calculate the simulation parameters R, h, q, and N and simulate from the right problem
- `W_f` calculates the Richardson Romberg weightes as describe in Pages (2007). The proposed idea i the paper was to store an offline abaqus for W for different values of M and R but for the sake of simplicity a function is used instead.
- `estimate_paremter` function that is used to empircally estimate the problem parametrs α, β, γ $var(Y_0)$ and $\theta = \sqrt{\frac{V1}{var(Y0)}}$

Finally a class for each problem is developped that implements the necessary `eval_Y` function that as before returns the coupled estimator $Y_l = P_l^f - P_{l-1}^c$ and it's moments along with the simulation cost represented by the processor time.

Chapter 4

Results and Comparaison

4.1 MLMC Estimator

We tested the convergence of the MLMC estimator for a geometric brownian motion as the theoritical price is given explicity by the black and sholes formula. We used as a payoff function a european call option payoff and a digital call option payoff. For the discretization we a used an euler scheme and a 1D milstein scheme. These test will give us all the complexity theorme cases as

- Euler discretization + european smooth payoff yields $\beta = \gamma$
- Euler discretization + digital payoff yields $\beta < \gamma$ since the discontinuity causes the coarse and fine paths to end up on either sides of the strike for $O(h^{\frac{1}{2}})$ paths and this leads to $\beta = \frac{1}{2}$
- Milstein discretization + european smooth payoff which yields $\beta > \gamma$ due the strong order of convergence of 1 given by milstein with the Lipshitz european payoff.

In the sequel only the Euler discretizatino in conjunction with european payoff is discussed. The results are well in accordance with the MLMC complexity theorems for all cases. The results of the figure below are for a GBM with parameters $S_0 = 100$, $r = 0.05$, $\sigma = 0.2$, $T = 1$ and $K = 100$

The top left plot illustrates the logarithm base 2 of the variance for both P_l and $P_l - P_{l-1}$ against the grid level. A slope of approximately -1 suggests that the variance $V_l = V(P_l - P_{l-1})$ is proportional to M^{-l}, which aligns with h_l. At grid level $l = 4$, V_l is observed to be more than 1,000 times smaller than the variance $V(P_l)$ of the standard Monte Carlo method using the same timestep.

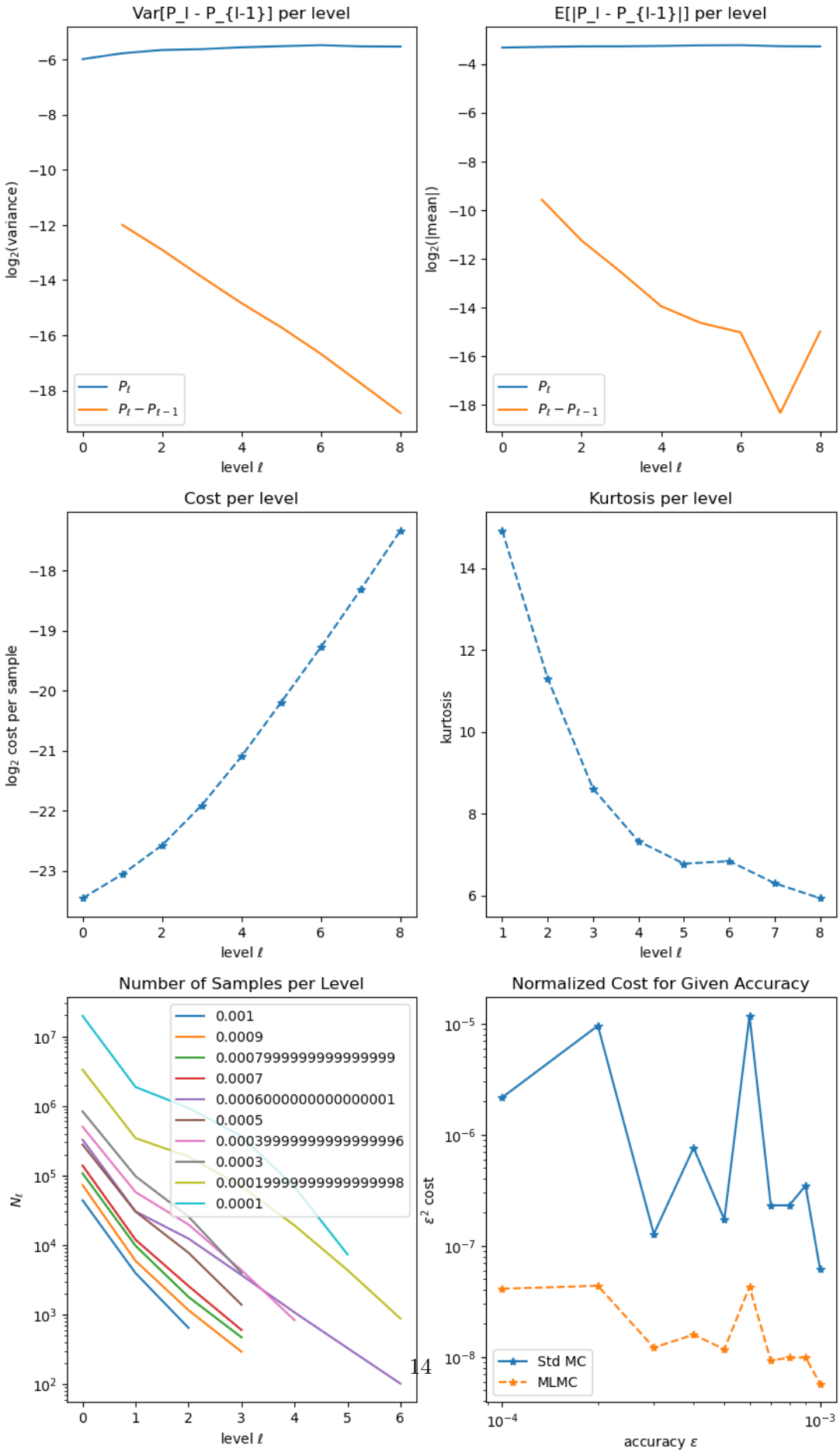


Figure 4.1: Convergence results for Euler scheme + European Call payoff

The top right plot demonstrates the mean value and correction at each level, revealing an approximately -1 slope for $E(P_l - P_{l-1})$, indicating $O(h)$ convergence. By grid level $l = 3$, the relative error $E(P - P_l)/E(P)$ is less than 10^{-3}.

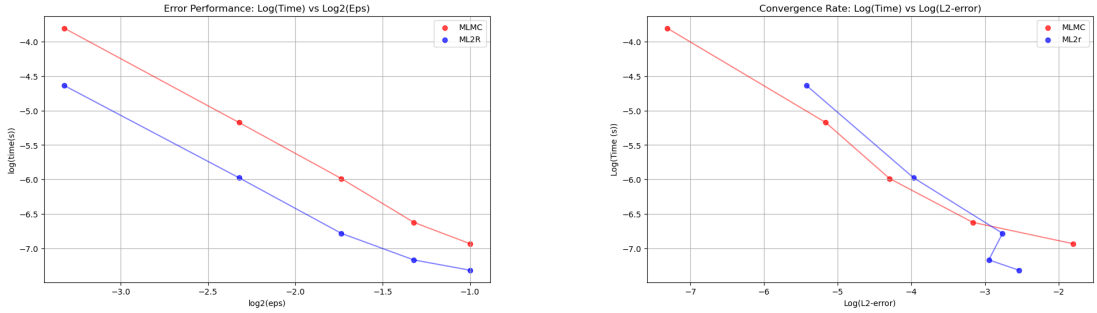
The middle right plot shows the the kurtosis decreases exponentially with level which means that the estimate is robust as variance estimation requires $O(\kappa)$ samples which is easily achievable at each iteration if the kurtosis is controlled.

In the bottom left plot, each line represents a multilevel calculation for N_l across levels $l = 0$ to L, showing decreasing N_l with increasing l due to reduced V_l and h_l. The maximum refinement level L increases as the desired accuracy ϵ improves.

The bottom right plot evaluates the computational complexity C, plotting $\epsilon^2 C$ versus ϵ. The results indicate a very slow increase in $\epsilon^2 C$ with ϵ for multilevel method, aligning with theoretical predictions of asymptotic proportionality to $\log(\epsilon)^2$. Standard Monte Carlo methods, predicted to have $\epsilon^2 C$ proportional to ϵ^{-1}.

4.2 MLMC vs ML2R

We compared the convergence of the MLMC and the ML2R estimators for a geometric brownian motion as the theoritical price is given explicity by the black and sholes formula.As before, We used as a payoff function a european call option payoff and a digital call option payoff. For the discretization we a used an euler scheme and a 1D milstein scheme. These test will give us all the complexity theorm cases.



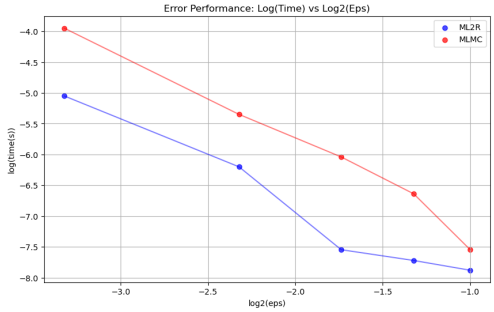
(a) CPU time (y-axis, log scale) as a function of emipirical error (x-axis, log2 scale)

(b) CPU time (y-axis, log scale) as a function of prescribed error (x-axis, log2 scale)

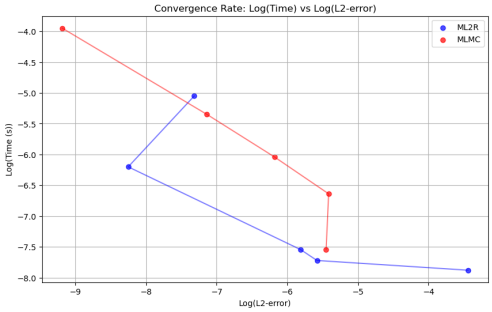
Figure 4.2: GBM with Euler Scheme + European Payoff

As the 4.2 show the ML2R estimator overperforms the MLMC estimator for a European call option on a Geomtric Brownian motion when an Euler Discretization scheme is used. in fact this is the case of $\beta = \gamma$ in the MLMC Complexiyt theorem which leads

to the ML2R estimator theoritically overperforming MLMC with a factor of $log(\epsilon)$ which explains the slight difference in cpu time.



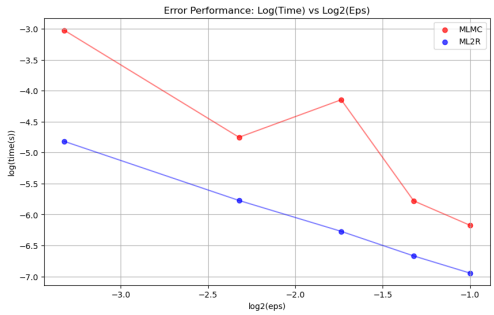
(a) CPU time (y-axis, log scale) as a function of emipirical error (x-axis, log2 scale)



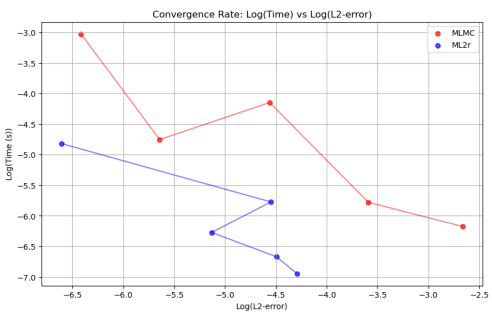
(b) CPU time (y-axis, log scale) as a function of prescribed error (x-axis, log2 scale)

Figure 4.3: GBM with Euler Scheme + digital Payoff

4.3 illustrates the problem with discontinuous payoff functions. The underlying SDE is exactly the same, but the payoff is a digital option with payoff $P(S) \equiv 10\exp(-rT)H(S_T - K)$, where $H(x)$ is the Heaviside step function. On the finer levels, most samples have fine and coarse paths which end on the same side of the strike K, and hence give $P_l - P_{l-1} = 0$. However, noting that the strong error is $O(h_l^{1/2})$, and there is a bounded density of paths terminating in the neighbourhood of K, there is therefore an $O(h_l^{1/2})$ fraction of the samples with the coarse and fine paths on either side of the strike, with $P_l - P_{l-1} = \pm 1$. This gives $V_l = O(h^{1/2})$, and this lower rate of convergence is clearly visible in plot (a). Furthermore, $E[(P_l - P_{l-1})^4] = O(h_l^{1/2})$ and so the kurtosis is $O(h_l^{-1/2})$; this increase in kurtosis is obvious in plot (d), which illustrates why this is a helpful quantity to plot. Consequently, this application has $\alpha = 1$, $\beta = \frac{1}{2}$, $\gamma = 1$. In this case the improvement factor is exponential and the importance of the ML2R extension is emphasized.



(a) CPU time (y-axis, log scale) as a function of emipirical error (x-axis, log2 scale)



(b) CPU time (y-axis, log scale) as a function of prescribed error (x-axis, log2 scale)

Figure 4.4: GBM with Milstein Scheme + European Payoff

This is the case of $\beta > \gamma$ where the theoritcally the MLMC and ML2R have the same order of convergence which is the optimal order $O(\epsilon^{-2})$ of an unbiased monte carlo estimator. As discussed in Pages and Lemaire (2017) despite the similar rates, ML2R is indicated to have a significantly lower constant associated with its error term. This conclusion is supported by numerical experiments, such as those using the Black-Scholes model discretized by the Milstein scheme. This suggests that ML2R, while matching MLMC in convergence rate under optimal conditions, may still offer more efficient computation. In this part of the analysis we try to replicate these findings.

Chapter 5

Conclusion

5.1 Conclusions

In this detailed study, we have embarked on an exhaustive comparative analysis of the Multilevel Monte Carlo (MLMC) method and the Multilevel Richardson-Romberg (ML2R) extrapolation within the domain of financial modeling. Our investigation aimed to elucidate the computational efficiency and accuracy of these methods in pricing a variety of financial derivatives, ranging from simple European payoffs to more complex derivatives under the Geometric Brownian Motion model.

The findings reveal that while MLMC offers significant improvements in computational cost by leveraging a hierarchical estimation process that refines accuracy incrementally, ML2R advances this by integrating Richardson-Romberg extrapolation, achieving higher order error reduction. This integration proves especially valuable in cases where the parameter β equals or exceeds the convergence parameter γ, demonstrating that ML2R can provide substantial gains in computational efficiency and accuracy.

Notably, in scenarios characterized by a high degree of payoff discontinuity, the advantages of ML2R become even more pronounced, offering substantial computational savings and more reliable error estimates. This is particularly evident when dealing with digital options, where the payoff discontinuity can dramatically affect the variance and bias of the estimates.

As a next step we can explore the possiblity of the development and parallelization of an adaptive version of the ML2R Estimator. One can also try differnet payoffs including path dependent option like asian or barrier options. Due to the high modularity and abstarction level of the code exploring other problems would only require modelling the coupled problem and coding inside a class as the rest of the code works independently.

Appendix A

Results

eps	L2-error	time	R	N	h
0.5	0.079012	0.000662	4.0	19462.418539	0.333333
0.4	0.052463	0.000769	4.0	36922.498080	0.500000
0.3	0.063221	0.001132	4.0	65639.996587	0.500000
0.2	0.018879	0.002538	5.0	175204.420981	0.500000
0.1	0.004418	0.009697	6.0	808002.171858	0.500000

Table A.1: ML2R: European payoff + GBM with Euler Discretization

eps	L2-error	time	R	N	h
0.5	0.164511	0.000974	3.0	19986.019244	0.5
0.4	0.042166	0.001328	3.0	31228.155069	0.5
0.3	0.013618	0.002512	3.0	81663.816970	1.0
0.2	0.005728	0.005667	4.0	142489.191866	0.5
0.1	0.000667	0.022320	4.0	569956.767465	0.5

Table A.2: MLMC: European payoff + GBM with Euler Discretization

eps	L2-error	time	R	N	h
0.5	0.031973	0.000379	4.0	690.117665	0.5
0.4	0.003773	0.000444	4.0	1078.308852	0.5
0.3	0.003001	0.000529	4.0	1916.993514	0.5
0.2	0.000261	0.002029	5.0	8392.568533	0.5
0.1	0.000664	0.006410	5.0	33570.274134	0.5

Table A.3: ML2R: Digital payoff + GBM with Euler Discretization

eps	L2-error	time	R	N	h
0.5	0.004295	0.000531	5.0	101.613378	0.500000
0.4	0.004450	0.001309	6.0	180.181221	0.333333
0.3	0.002067	0.002382	7.0	416.465039	0.333333
0.2	0.000794	0.004754	8.0	1191.503203	0.333333
0.1	0.000103	0.019252	10.0	7387.626164	0.333333

Table A.4: MLMC: Digital payoff + GBM with Euler Discretization

eps	L2-error	time	R	N	h
0.5	0.013696	0.000963	4.0	3.006031e+04	0.333333
0.4	0.011172	0.001270	4.0	7.619955e+04	0.500000
0.3	0.005907	0.001889	4.0	1.354659e+05	0.500000
0.2	0.010593	0.003111	5.0	3.311203e+05	0.500000
0.1	0.001352	0.008081	6.0	1.395892e+06	0.500000

Table A.5: ML2R: european payoff + GBM with Milstein Discretization

eps	L2-error	time	R	N	h
0.5	0.069744	0.002085	3.0	4.178661e+04	0.5
0.4	0.027621	0.003091	3.0	6.529159e+04	0.5
0.3	0.010464	0.015827	3.0	3.142609e+05	1.0
0.2	0.003545	0.008635	4.0	2.644505e+05	0.5
0.1	0.001636	0.048576	4.0	1.057802e+06	0.5

Table A.6: MLMC: european payoff + GBM with Milstein Discretization