Algorithms Implementation for Computing Bermudan Options Price and CVA

Final Degree Internship Report

Ricardo Rincón García

Directed by Bernard LAPEYRE
Marouan IBEN TAARIT

October 3, 2015

Mathématiques Appliquées et Calcul Scientifique

Thanks to an agreement with

Remerciements

J'adresse tout d'abord mes plus sincères remerciements aux personnes travaillant au CERMICS qui, au cours de ce stage, m'ont aidé et soutenu dans les activités quotidiennes et ont ainsi contribué à sa réussite.

Je remercie plus particulièrement mes encadrants Bernard LAPEYRE et Marouan IBEN TAARIT pour m'avoir fait confiance sur ce projet et pour le temps qu'ils m'ont dédié malgré leur emplois du temps chargés.

Je tiens à remercier, pour leur disposition à m'aider et leur leçons de français, les autres stagiaires et doctorants présents pendant mon stage. Je citerai spécialement Henri GERARD, Valentin FOUCHER et Houzhi LI.

Je remercie également Ahmed KEBAIER pour m'avoir ouvert la porte à cette opportunité, Isabelle SIMUNIC pour son aide précieuse dans le domaine administratif, Antonino ZANETTE pour sa patience à l'heure d'intégrer les codes sur Premia et Laurent TOURNIER pour son égard en tant que tuteur du stage à Paris 13.

Je tiens aussi à adresser mes remerciements à Olivier LAFITTE, Emmanuel AUDUSSE, et plus particulièrement à Adolfo QUIRÓS, sans lesquels je ne serais pas ici.

Enfin, j'adresse une grosse pensée à mes parents, ma sœur, ma famille et mes amis qui m'ont encouragé et apporté leur soutien durant mon expérience parisienne, ainsi que ceux avec lesquels j'ai partagé mes doutes académiques.

Merci à tous.

"Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better."

Samuel Beckett

Abstract

This report presents the labour developed during six months of internship in the research laboratory CERMICS working on the study of the Stochastic Grid Bundling Method (SGBM) [7]. This method combines the techniques of simulation, regression, and bundling. Initially, we analyse the use of SGBM for pricing multi-dimensional Bermudan options.

Subsequently we tackle the pricing of Credit Valuation Adjustment (CVA) on the backward dynamics framework provided by the SGBM as exposed in [4]. The valuation of the CVA has become a fundamental task these days due to the introduction of Basel III. Therefore, the possibility to integrate the computation of exposure profiles in the SGBM means a great advantage of this algorithm.

Computational results for several multi-dimensional European and Bermudan options show the efficacy of the method studied for pricing purposes. Furthermore, we give examples of calculating exposure and CVA for European and Bermudan options under the Black-Scholes and Heston asset dynamics.

Key words — pricing, European options, Bermudan options, Stochastic Grid Bundling Method (SGBM), Monte Carlo simulation, least-squares-polynomial-approximation, expected exposure, Credit Valuation Adjustment (CVA), Debit Valuation Adjustment (DVA), Black-Scholes model, Heston model.

Contents

1 Introduction

During my internship I have studied and implemented in Scilab and C the Stochastic Grid Bundling Method (SGBM) proposed by Shashi Jain and Cornelis W. Oosterlee (2013) [7]. Coming face to face with both high dimensionality and path-dependent nature when pricing Bermudan options under multi-dimensional stochastic processes, Monte Carlo simulation methods appears to represent an interesting solution. The reason is that these methods, based on stochastic sampling of the underlying state vector, have a convergence rate independent of the dimension of the problem. In addition, pricing of Bermudan options, the discrete version of the American-style options in which the holder can choose the time of exercise, needs to take into account an optimal exercise policy. A dynamic programming approach serves for this purpose. The SGBM exposed in [7] solves the difficulty of combining these two aspects at the same time for pricing Bermudan basket options. The method employs techniques of simulation, regression, and bundling.

The second task accomplished in my internship has consisted of using the SGBM for computation of exposure profiles. The possibility to integrate the computation of exposure profiles in the SGBM thanks to its structure is proposed by Feng and Oosterlee (2014) [4]. This possibility means a great advantage of this algorithm due to the current importance of pricing the Credit Valuation Adjustment (CVA), which needs the exposure values to be computed. We apply the SGBM with European and Bermudan basket derivative products under Black-Scholes dynamics, and options with a single asset under stochastic volatility dynamics, the Heston model, in order to increase our range of financial models tackled.

My last task during the internship has comprised the understanding and translation from R to C code of a method proposed by my internship directors Bernard Lapeyre and Marouan Iben Taarit (2015) [8]. They establish a forward representation of a derivative's expected exposure, with the key idea of considering the expected exposure as the price of a compound option, using Dupire-like arguments.

This report is organized as follows. After presenting the organizations involved in my internship in Chapter 2, we describe the SGBM and we provide numerical experiments of increasing complexity in Chapter 3 in order to discuss some features of the method. Chapter 4 gives the framework to apply the SGBM for computing exposure profiles for European and Bermudan options, and shows computational aspects and numerical examples for Black-Scholes and Heston models. In Chapter 5, we present the approach for computing derivatives exposure proposed by B. Lapeyre and M. Iben Taarit and we find again some of their numerical results. Conclusions are finally addressed in Chapter 6.

2 CERMICS

This internship has been developed in the CERMICS, has been framed in the Math-Risk Project of INRIA, and has contributed to the software Premia. These three involved parts are presented in this chapter.

2.1 CERMICS

CERMICS[1] ('Centre d'Enseignement et de Recherche en Mathématiques et Calcul Scientifique') is a laboratory of École des Ponts ParisTech, hosting joint research teams with INRIA and University of Marne-la-Vallée[2]. It is located at École des Ponts ParisTech in Champs-sur-Marne. The scientific activity of CERMICS covers several domains in scientific computing, modelling, and optimization.

The laboratory, whose direction is ensured by Jean-François Delmas, is organized in three research teams :

- **Modelling, Analysis, and Simulation group.** The scientific themes of this group are focused on the mathematical study, the numerical analysis, and the simulation of mechanics and physics equations. The areas of application are :
  - Molecular and multi-scale simulation, in particular, coupling the microscopic-scale models (quantum physics and statistics) and the macroscopic-scale models. The mathematical tools used are varied: partial differential equations analysis, spectral analysis, stochastic processes analysis, variational methods, etc.
  - Fluid and solid mechanics for which mathematical models and numerical methods in a more macroscopic scale are developed. These works cover, on one hand, finite element methods, discontinuous Galerkin methods, higher-order hybrid methods, and error estimation a posteriori.
  - PDE and materials. Dislocation dynamics modelled at different scales and open boundary problems are studied.
- **Optimization and Systems group.** Focused on optimization and its applications. Its main expertise domains are stochastic dynamic optimization and discrete optimization. The group has established long-lasting cooperations with the industrial

[1] `http://cermics.enpc.fr/`
[2] `http://www.u-pem.fr/`

world, the national and international academic community. Applications cover the management of large-scale energy systems under uncertainty, the sustainable management of biodiversity, and the management and design of transport systems. For this, the group develops numerical methods and scientific softwares.

- **Applied Probability group.** The objectives of the applied probability team are :
  - The modelling, conception, and study of computational algorithms, from probabilities or using them, in the following areas: financial mathematics, physics, chemistry, biologist, and engineering sciences.
  - The study of the probabilistic interpretation of partial differential equations.

Both of my supervisors, Bernard Lapeyre and Marouan Iben Taarit, are part of the Applied Probability group as researcher and PhD student respectively.

In addition, the CERMICS is connected to the École Doctorale MSTIC[3] and it adheres to the SMAI[4]. The laboratory also contributes to the Labex Bézout[5] at the interface between mathematics and computer science since 2011 and to the Labex MMCD[6] since 2012.

Figure 2.1: Centre d'Enseignement et de Recherche en Mathématiques et Calcul Scientifique.

2.2 INRIA

INRIA[7] ('Institut national de recherche en informatique et en automatique') is a public science and technology institution which promotes "scientific excellence for technology transfer and society".

[3] http://www.univ-paris-est.fr/fr/index.html
[4] Société de Mathématiques Appliquées et Industrielles http://smai.emath.fr/?lang=fr
[5] Models and algorithms: from discrete to continuous http://www.univ-paris-est.fr/fr/bezout-modeles-et-algorithmes-du-discret-au-continu/
[6] Multi-Scale Modelling & Experimentation of Materials for Sustainable Construction http://www.univ-paris-est.fr/fr/labex-mmcd-modelisation-experimentation-pour-la-construction-durable/
[7] http://www.inria.fr/

Graduates from the world's top universities, INRIA's 2,700 employees rise to the challenges of digital sciences. Research at INRIA is organised in "project teams" which bring together researchers with complementary skills to focus on specific scientific projects. With this open, agile model, INRIA is able to explore original approaches with its partners in industry and academia and provide an efficient response to the multidisciplinary and application challenges of the digital transformation. The source of many innovations that add value and create jobs, INRIA transfers expertise and research results to companies (startups, SMEs[8] and major groups) in fields as diverse as healthcare, transport, energy, communications, security, and privacy protection, smart cities and the factory of the future.

From the infancy of computer science to the digital dominance of today, INRIA's history dates back more than forty years. Below are a few key figures[9] about the National Institute, which is placed under the supervision of the French Ministries of Research and Industry.

- Scientific activities
  - 170 INRIA project-teams
  - 4,500 scientific publications per year
  - 300 PhDs supported
- Industrial relations
  - 370 active patents
  - 143 software programmes registered with France's Software Protection Agency in 2014
  - 24 start-ups INRIA since 2010
- Structure
  - 8 research centres located throughout France (Rocquencourt, Rennes, Sophia Antipolis, Grenoble, Nancy, Bordeaux, Lille, and Saclay) and a head office in Rocquencourt, near Paris.
- Human resources
  - 2700 members of staff
  - including 1,800 scientists
- Budgetary resources
  - Total budget: 230 m €
  - Proportion self-financed: 37%

[8] Small and medium-sized enterprises
[9] 2014 figures

Figure 2.2: Institut National de Recherche en Informatique et en Automatique.

2.2.1 Math-Risk

Math-Risk is a research team led by Agnès Sulem-Bialobroda for a mathematical risk handling and it is based on the former project team Mathfi (2000-2011). It is a joint team between, INRIA Paris-Rocquencourt, Ecole des Ponts et Chaussées (CERMICS laboratory) and the Université Paris-Est Marne la Vallée (LAMA[10] laboratory).

The Mathfi project team has extensively contributed to the development of modelling and computational methods for the pricing and hedging of increasingly complex financial products. Since the crisis of 2008, there has been a critical reorientation of research priorities in quantitative finance with emphasis on risk. Financial mathematics is facing the following new evolutions:

(i) The complete market modelling has become unsatisfactory to provide a realistic picture of the market and is replaced by incomplete and multidimensional models which lead to new modelling and numerical challenges.

(ii) Quantitative measures of risk coming from the markets, the hedging procedures, and the lack of liquidity are crucial for banks.

(iii) Uncontrolled systemic risks may cause planetary economic disasters, and require better understanding.

(iv) Deregulation of stock markets and its consequences lead to study high frequency trading.

Research themes

To meet these new issues, Math-Risk proposes to focus on dependence modelling, systemic risk, market microstructure modelling, and risk measures. The research on the areas related to the current expertise of the team in modelling and numerical analysis remains active in this new context, motivated by new issues. Mathematical tools for dealing with the new challenges of financial modelling are stochastic modelling, stochastic analysis, in particular stochastic (partial) differential equations and various aspects of stochastic control of these equations, and advanced numerical probability for effective solutions.

[10] `http://www.u-pem.fr/recherche/unites-de-recherche/lama-umr-8050-cnrs/`

2.3 Premia

Since the beginning our aim was to integrate the methods code programmed during this internship in Premia[11]. In addition, we needed to verify our results with an evidence-based source in the process so we did it with other methods already present in this powerful tool for professional financial research teams. Therefore, a presentation of this tool and its basic working principle is deserved.

Premia is a software designed for option pricing, hedging, and financial model calibration. It is provided with its C/C++ source code and an extensive scientific documentation.

Efficient computations of prices and hedges for derivative products are major issues for financial institutions. The development of increasingly complex financial products requires the use of advanced stochastic and numerical analysis techniques. A consortium of banks[12] have been using Premia since its beginning in 1999 and have brought important contributions to the project.

After the release of each new version of Premia, two-year-older versions become available on its web site (Figure 2.3) and can be freely downloaded for academic purposes.

Premia is developed by the Math-Risk team which gathers research scientists from INRIA (the French national institute for research in computer science and control), École des Ponts ParisTech (CERMICS), and the University of Marne la Vallée (Figure 2.4).

Figure 2.3: Premia's home page.

Its working interface can be seen in Figure 2.5 for the pricing of an European put on several assets under the Black-Scholes model and in Figure 2.6 for the pricing of an Asian call on a single asset under the Heston model. We can appreciate in these figures all the possible parameters to modify at user's whim.

[11] `https://www.rocq.inria.fr/mathfi/Premia/`

[12] Premia is developed in interaction with a consortium of financial institutions or departments presently composed of: Crédit Agricole Corporate and Investment Bank, Natixis, Société Générale, Raiffeisen Zentralbank and Bank Austria. The participants of the consortium finance the development of Premia (by contributing to the salaries of expert engineers hired by the Math-Risk project every year to develop the software) and help to determine the directions in which the project evolves.

Figure 2.4: Premia's creator institutions.

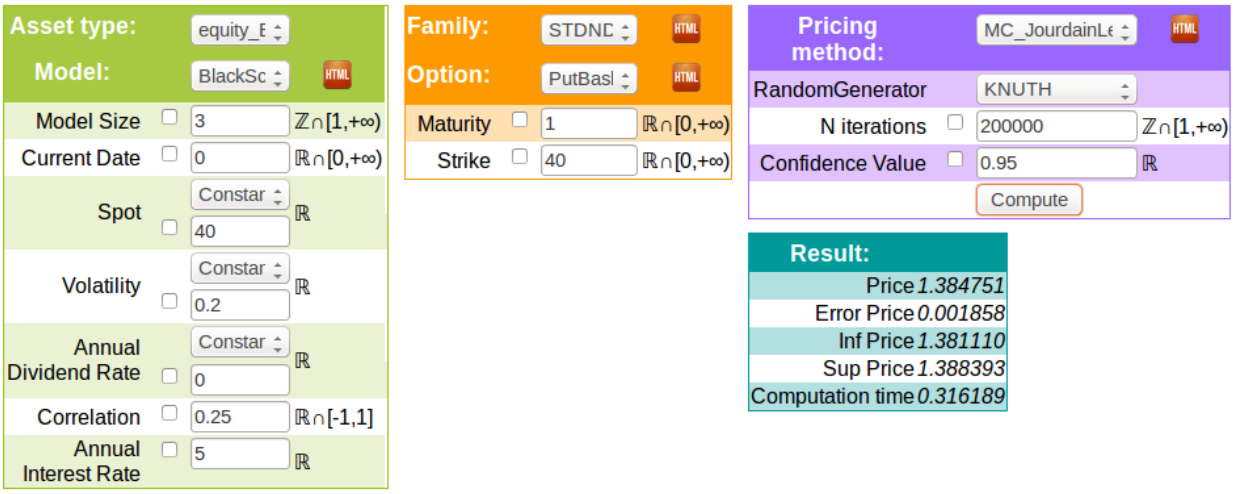


Figure 2.5: Example of European put pricing with Premia.

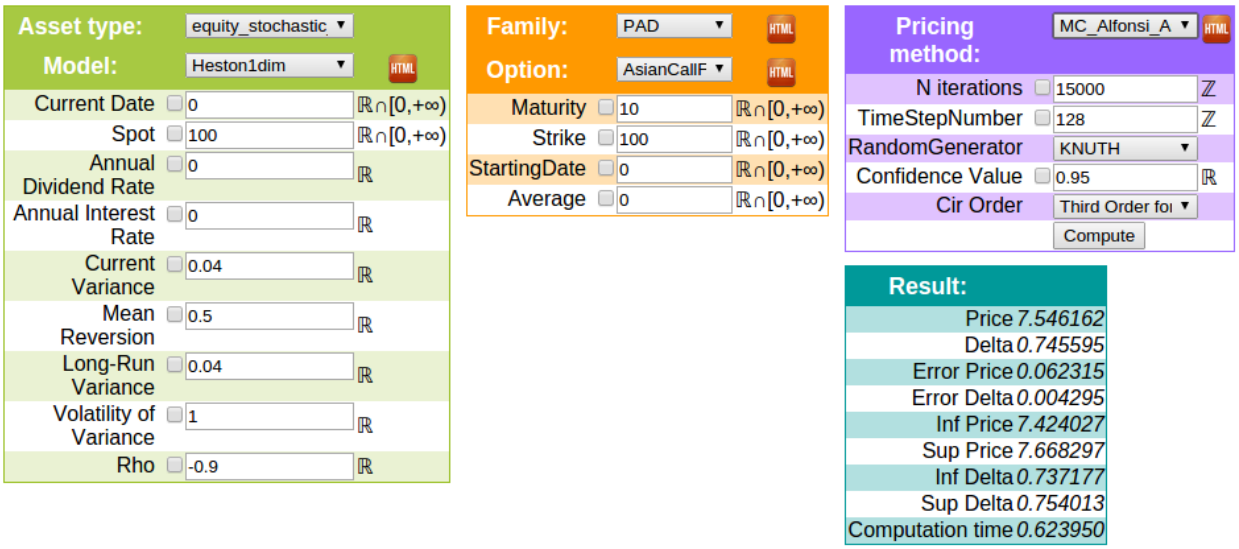


Figure 2.6: Example of Asian call pricing with Premia.

3 The Stochastic Grid Bundling Method: Efficient Pricing of Bermudan Options

Shashi Jain and Cornelis W. Oosterlee (2013) [7] proposed the Stochastic Grid Bundling Method (SGBM) for pricing of Bermudan options with several underlying assets. The steps involved in the SGBM algorithm, are:

- Step I: Generating grid points
- Step II: Option value at terminal time
- Step III: Bundling
- Step IV: Mapping high-dimensional state space to a low-dimensional space
- Step V: Computing the continuation and option values at t_{m-1}

We perform initially the first two steps and then, starting from t_M and moving backwards in time, steps III to V are performed for each time step, t_m, $m \leq M$. The next sections retake the theoretical support exposed in [7] and for more details and the proofs of all theoretical results one can look into the paper.

As mentioned before, the study and understanding of the SGBM represents the first task of my internship. I started working on the clustering algorithms isolated from the financial environment and later I came into the other parts involved in this pricing method. Pseudocodes of the algorithmic process followed are presented in Appendix D. Every graph displayed from here has been produced with the code I have done during the internship, which version in C is meant to be in Premia v.18 (see Appendix C). We discuss the outcome obtained for some examples of increasing complexity in the Numerical experiments section.

3.1 Stochastic Grid Bundling Method

We start providing the definition of the Bermudan options pricing problem and setting up the notations given in [7] and used in this section, and then explaining the steps involved in the SGBM algorithm.

3.1.1 Problem formulation

We assume a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and we consider the time duration $[0, T]$, finite, corresponding to the derivative product's life. Ω is the set of all possible realizations of the stochastic market in this temporal space and $\mathcal{F}_t$, the information structure in this market, $t \in [0, T]$. $\mathbb{P}$ is the risk-neutral probability measure on elements of $\mathcal{F}$. In this frame, $\mathbf{S}_t = (S_t^1, \ \ldots, \ S_t^d) \in \mathbb{R}^d$, represents the market state variable at each time $t \in [t_0 = 0, \ \ldots, \ t_m, \ \ldots, \ t_M = T]$, discrete.

Denoting r_t the instantaneous risk-free rate of return, which we consider here constant, we define the discount as :

$$D_{t_m} = \frac{B_{t_m}}{B_{t_{m+1}}},$$

where $B_t = \exp\left(\int_0^t r_s ds\right)$ is the risk-less savings account function. We call the intrinsic value of the option $h(\mathbf{S}_t)$, which indicate that the holder of the option receives $\max(h_t, 0)$, if the option is exercised at time t. We are interested in finding the value of the option at the initial state $\mathbf{S}_{t_0}$, which can be computed as :

$$V_{t_0}(\mathbf{S}_{t_0}) = \max_{\tau} \mathbb{E}\left[\frac{h(\mathbf{S}_\tau)}{B_\tau}\right],$$

where τ is a stopping time taking values in the discrete set $\{0, \ t_1, \ \ldots, \ T\}$. The value of the option at its maturity T corresponds to the product's pay-off,

$$V_T(\mathbf{S}_T) = \max(h(\mathbf{S}_T), 0). \tag{3.1}$$

The conditional continuation value Q_{t_m}, i.e. the expected pay-off at time t_{m+1}, is given by:

$$Q_{t_m}(\mathbf{S}_{t_m}) = D_{t_m} \mathbb{E}[V_{t_{m+1}}(\mathbf{S}_{t_{m+1}}) | \mathbf{S}_{t_m}]. \tag{3.2}$$

For Bermudan options, the value of the option at time t_m and state $\mathbf{S}_{t_m}$ is calculated as :

$$V_{t_m}(\mathbf{S}_{t_m}) = \max(h(\mathbf{S}_{t_m}), \ Q_{t_m}(\mathbf{S}_{t_m})). \tag{3.3}$$

3.1.2 Initial steps

We start by doing a Monte Carlo simulation, following the indicated scheme for the required financial model, of the stochastic paths. In Appendix A we can see the C code to generate grid points on a single or multiple assets under Black-Scholes dynamics, which is the model treated in this Chapter. Simulating independent copies of the market state variable, $\{\mathbf{S}_{t_0}(n), ..., \mathbf{S}_{t_M}(n)\}$, $n = 1, ..., N$, is a forward process in which all the paths start from the same initial state $\mathbf{S}_{t_0}$. Therefore, our grid is :

$$\begin{pmatrix} \mathbf{S}_{t_0} & \mathbf{S}_{t_1}(1) & \cdots & \cdots & \mathbf{S}_{t_M}(1) \\ \mathbf{S}_{t_0} & \mathbf{S}_{t_1}(2) & \cdots & \cdots & \mathbf{S}_{t_M}(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{S}_{t_0} & \mathbf{S}_{t_1}(N) & \cdots & \cdots & \mathbf{S}_{t_M}(N) \end{pmatrix}.$$

Subsequently the option value for all paths at terminal time is computed as :

$$V_{t_M}(\mathbf{S}_{t_M}) = \max\left(h(\mathbf{S}_{t_M}), 0\right).$$

Figure 3.1 shows a $N = 10,000$ paths simulation of a single asset under Black-Scholes model for data set 1 in Table 3.1 (table further on this report introducing the parameters for the numerical examples under Black-Scholes model presented).

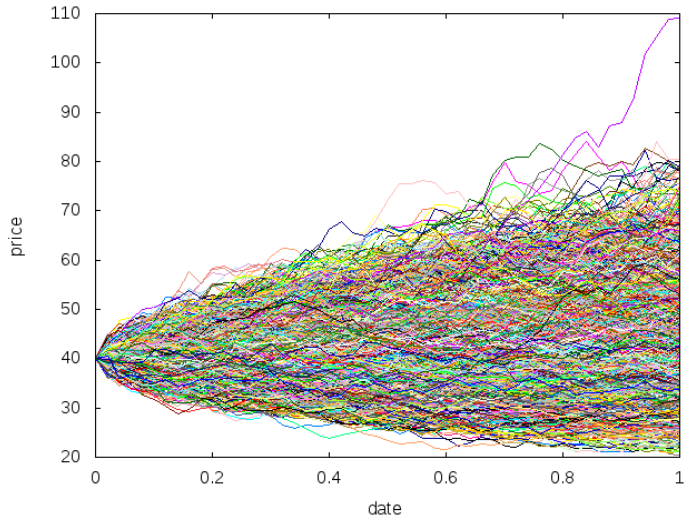


Figure 3.1: Plot of generated paths.

3.1.3 Bundling

We want to sample the distribution of $\mathbf{S}_{t_m}$ conditional on the state $\mathbf{S}_{t_{m-1}}$, in a backward loop. To accomplish this, the SGBM firstly clusters using some measure of proximity the grid points at t_{m-1} into ν non-overlapping partitions. Then, we sample $\mathbf{S}_{t_m}$ with those paths originated from the bundle that contains $\mathbf{S}_{t_{m-1}}$. We have considered the three different approaches for partitioning proposed by Jain and Oosterlee (2013) [7] for SGBM and we explain them next.

K-means clustering algorithm

The objective of this algorithm is to cluster points so as to minimize the sum of squares within clusters, i.e :

$$\underset{\mathcal{B}_{t_{m-1}}}{\arg\min} \sum_{\beta=1}^{\nu} \left(\sum_{\mathbf{S}_{t_{m-1}}(n) \,\in\, \mathcal{B}_{t_{m-1}}(\beta)} \|\mathbf{S}_{t_{m-1}}(n) - \mu_\beta\|^2 \right),$$

where μ_β is the mean of the points into each of the ν non-overlapping bundles β, $\mathcal{B}_{t_{m-1}}(\beta)$. The algorithm (Lloyd 1982 [9]) uses an iterative refinement technique. Initially, we take ν aleatory points from the N points to cluster as bundle centroids, $\mu_1^1, ..., \mu_\nu^1$. Subsequently, we perform the following steps alternately :

- **Step 1** : Assign grid points to the set whose mean is closest to it.

$$\mathcal{B}_{t_{m-1}}^{(l)}(\beta) = \{\mathbf{S}_{t_{m-1}}(n) : \|\mathbf{S}_{t_{m-1}}(n) - \mu_\beta^{(l)}\|^2 \leq \|\mathbf{S}_{t_{m-1}}(n) - \mu_j^{(l)}\|^2, \forall\ 1 \leq j \leq \nu\},$$

  where grid point $\mathbf{S}_{t_{m-1}}(n)$ is assigned to just one bundle.

- **Step 2** : If the assignment of the grid points does not change anymore from a previous iteration the process has converged, else the means are updated into each of the new clusters as :

$$\mu_\beta^{(l+1)} = \frac{1}{|\mathcal{B}_{t_{m-1}}^{(l)}(\beta)|} \sum_{\mathbf{S}_{t_{m-1}}(n)\ \in\ \mathcal{B}_{t_{m-1}}^{(l)}(\beta)} \mathbf{S}_{t_{m-1}}(n),$$

  where $|\mathcal{B}_{t_{m-1}}^{(l)}(\beta)|$ is the cardinal of the set $\mathcal{B}_{t_{m-1}}^{(l)}(\beta)$.

As we will see in 'Numerical experiments' section, this is computationally the most expensive algorithm of the three presented here. In order to not prejudice *k-means clustering* with respect to the others, especially in high dimensions, we can set a maximum number of iterations before stopping performing those two steps, without having converged. Even in this case, we find an accurate price.

In addition, we can specify directly centroids and just distribute grid points in clusters as indicated in step 1. We use this procedure when computing path estimator with centroids previously found in calibration phase. Figure 3.2 shows the bundles obtained using the *k-means clustering* algorithm for a two dimension grid of normally distributed points.

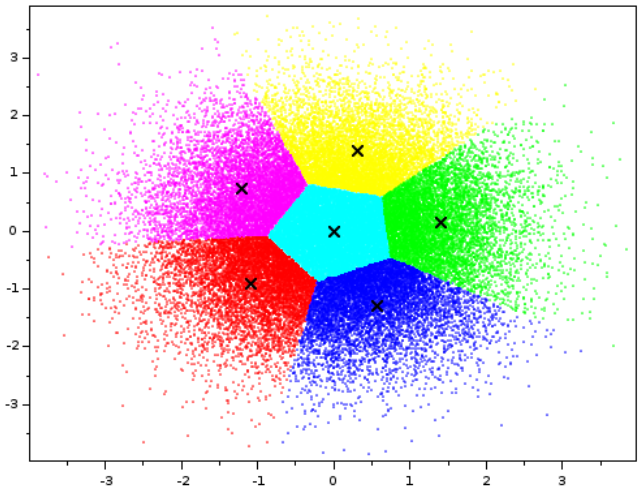


Figure 3.2: Bundling of grid points in a two-dimensional space using *k-means clustering*. The grid points are bundled into 6 non-overlapping partitions.

Recursive bifurcation

As convergence when facing high dimension state vectors in k-means algorithm can be slow, Jain and Oosterlee (2013)[1] propose fast practical schemes to cluster grid points based on proximity :

- **Step 1** : Compute the mean of the grid points along each dimension,

$$\mu_\delta = \frac{1}{N}\sum_{n=1}^{N} S^\delta_{t_{m-1}}(n),\ \ \delta = 1, ..., d.$$

- **Step 2** : Bundle separately along each dimension the grid points by dividing the grid into 2^d sets according to :

$$A_\delta = \{\mathbf{S}_{t_{m-1}}(n) : S^\delta_{t_{m-1}}(n) > \mu_\delta,\ \ n = 1, ..., N\},$$

$$\bar{A}_\delta = \{\mathbf{S}_{t_{m-1}}(n) : S^\delta_{t_{m-1}}(n) \leq \mu_\delta,\ \ n = 1, ..., N\},$$

  where $\delta = 1, ..., d$.

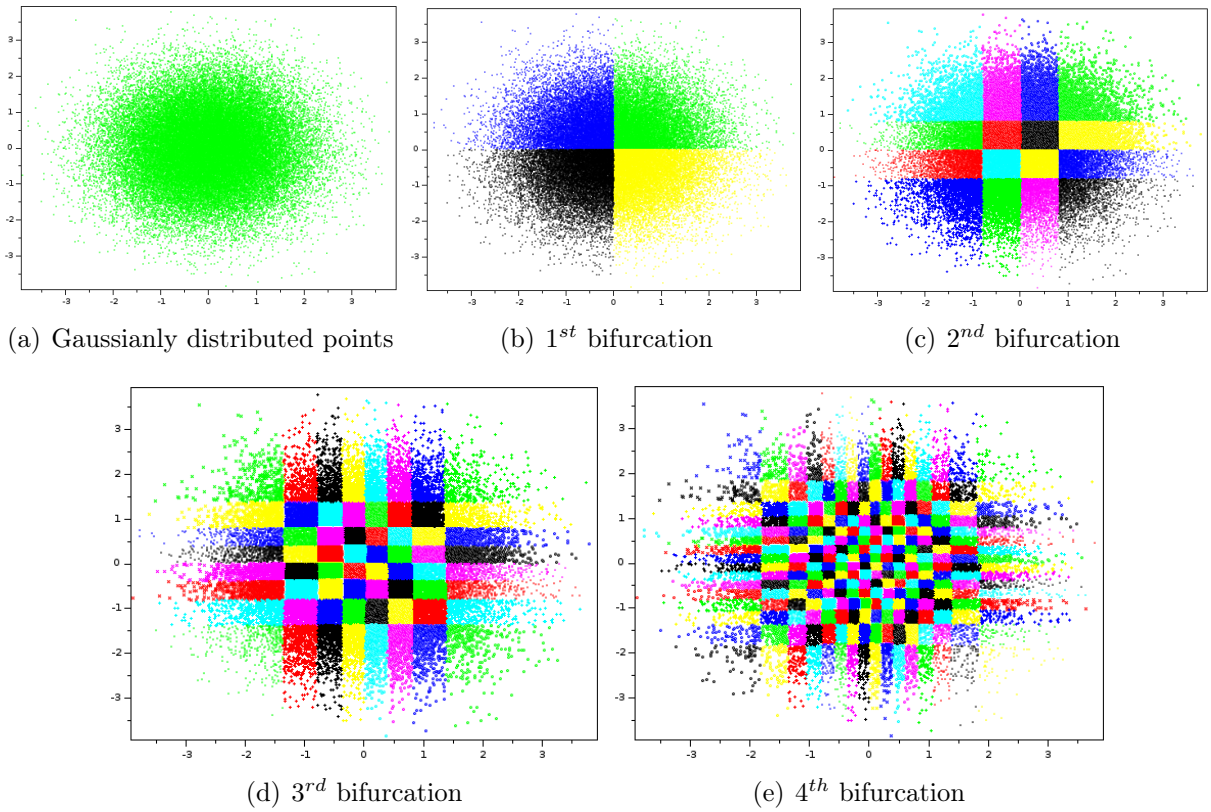
- **Step 3** : The 2^d unique non-overlapping clusters are obtained intersecting these sets as follows :

$$\begin{aligned}
\mathcal{B}_{t_{m-1}}(1) &= A_1 \cap A_2 \cap ... \cap A_d,\\
\mathcal{B}_{t_{m-1}}(2) &= \bar{A}_1 \cap A_2 \cap ... \cap A_d,\\
\mathcal{B}_{t_{m-1}}(3) &= A_1 \cap \bar{A}_2 \cap ... \cap A_d,\\
&\vdots\\
\mathcal{B}_{t_{m-1}}(2^d) &= \bar{A}_1 \cap \bar{A}_2 \cap ... \cap \bar{A}_d,
\end{aligned} \tag{3.4}$$

We can continue performing as much iterations as we want of previous steps to split further each bundle. Figure 3.3 shows an example of bundling grid points, $N = 50,000$, in a two-dimensional space with the recursive bifurcation approach. In first iteration we divide in two each dimension obtaining 4 clusters. Then, each of these partitions follows the same procedure, resulting in 16 partitions in the second iteration, 64 in the third iteration, and 256 bundles in the fourth iteration. Abstracting a formula, after p iterations the number of bundles will be $(2^d)^p$. As we can appreciate, this scheme will be less interesting, even unworkable, with increasing dimensions of the problem, as the number of clusters obtained will be too large even after just one iteration.

Recursive bifurcation of reduced state space

We can also bundle the grid points based on proximity of the reduced state space $h(\mathbf{S}_{t_{m-1}})$, i.e. starting by using the payoff as a mapping function and then employing the clustering procedure of the *recursive bifurcation* to the mapped points (which belong to dimension $d = 1$). However, in terms of programming, we do not call the *recursive bifurcation* function after obtaining the reduced state space, because of feasible optimizations in the algorithm and memory allocation. The number of bundles obtained after p iterations in this case will be 2^p. Figure 3.4 shows an example of bundling of uniformly distributed points in $[0, 1] \times [0, 1]$. It results in 8 partitions with three iterations of the method, 2^3.



(a) Gaussianly distributed points (b) 1^{st} bifurcation (c) 2^{nd} bifurcation

(d) 3^{rd} bifurcation (e) 4^{th} bifurcation

Figure 3.3: Clustering of Gaussianly distributed points in a two-dimensional space.

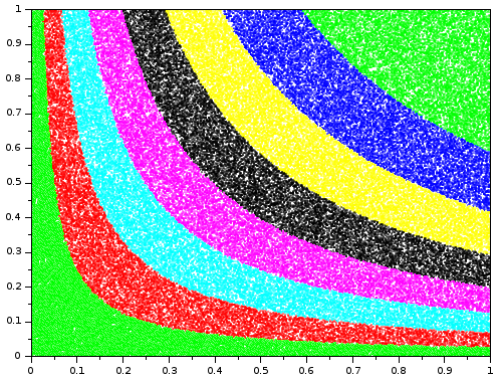


Figure 3.4: Bundling into 8 non-overlapping clusters of grid points using *recursive bifurcation of reduced state space* with the geometric mean as mapping function.

3.1.4 Mapping high-dimensional state space to a low-dimensional space

Corresponding to each bundle $\mathcal{B}_{t_{m-1}}(\beta)$, $\beta = 1, ..., \nu$, a parametrized value function $Z : \mathbb{R}^d \times \mathbb{R}^K \to \mathbb{R}$, is introduced. This approximation of the value function assigns values $Z(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m})$ to states $\mathbf{S}_{t_m}$ in order to better deal with the large dimension of the state space. $\alpha^{\beta}_{t_m} \in \mathbb{R}^K$ is a vector of free parameters and we aim to choose, for each t_m and bundle β, a parameter vector so that $Z(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}) \approx V_{t_m}(\mathbf{S}_{t_m})$.

We are said to use basis functions that map the state space from $\mathbb{R}^d$ to $\mathbb{R}$, to approximate the value functions. The function $Z(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m})$ projecting the option values onto the span of ϕ is restricted to a linear combination of basis functions and, can be approximated by :

$$Z(\mathbf{S}_{t_m}, \hat{\alpha}^{\beta}_{t_m}) = \sum_{k=1}^{K} \hat{\alpha}^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m}), \tag{3.5}$$

satisfying,

$$\arg\min_{\hat{\alpha}^{\beta}_{t_m}} \sum_{n=1}^{|\mathcal{B}_{t_{m-1}}(\beta)|} \left(V_{t_m}(\mathbf{S}_{t_m}(n)) - \sum_{k=1}^{K} \hat{\alpha}^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m}(n)) \right)^2. \tag{3.6}$$

Therefore, the parametrized function $Z(\mathbf{S}_{t_m}, \hat{\alpha}^{\beta}_{t_m})$ is computed, corresponding to each bundle $\mathcal{B}_{t_{m-1}}(\beta)$, using ordinary least squares regression, so that:

$$V_{t_m}(\mathbf{S}_{t_m}(n)) = Z(\mathbf{S}_{t_m}(n), \hat{\alpha}^{\beta}_{t_m}) + \epsilon^{\beta}_{t_m}, \tag{3.7}$$

where $\mathbf{S}_{t_{m-1}}(n) \in \mathcal{B}_{t_{m-1}}(\beta)$ and $\epsilon^{\beta}_{t_m}$ is the error made in the regression. In theoretical terms, it is assumed that $\mathbb{E}[\epsilon^{\beta}_{t_m} | \mathbf{S}_{t_{m-1}}(n)] = 0$.

Figure 3.5 shows an example of clustering into 5 bundles, with k-means algorithm, of the paths at each time step when computing calibration for the SGBM. Calibration is how we call to the phase in which we calculate the vector of free parameters $\alpha^{\beta}_{t_m}$ for each t_m and bundle β by performing a least squares regression. These parameters are stored and used to compute the estimators of the price. For this example we take data set 1 in Table 3.1 and $N = 20,000$.

Figure 3.6 shows the evolution of the option value function for a put on a single asset, clustering the paths at each time step in 8 bundles with k-means, when computing calibration for SGBM. We obtain the graphics in backward, but we display them in the forward temporal order. The black line correspond to the linear regression and it can be appreciated how it tends to the payoff from buying a put when the date approaches to the option maturity, $T = t_M$. For this example we take data set 1 in Table 3.1 and $N = 50,000$.

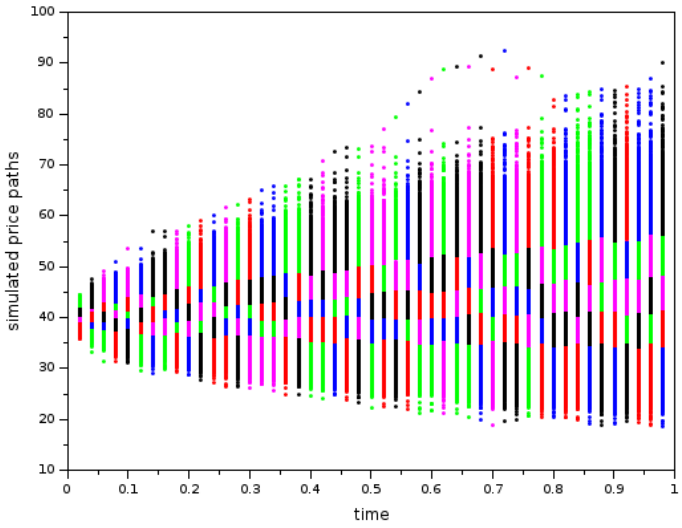


Figure 3.5: Clustering of the paths at each time step.

3.1.5 The continuation and option values at t_{m-1}

Now, using the parametrized option value function $Z(\mathbf{S}_{t_m}, \hat{\alpha}^{\beta}_{t_m})$ corresponding to bundle $\mathcal{B}_{t_{m-1}}(\beta)$, the continuation values for the paths into this cluster are approximated by :

$$\hat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = D_{t_{m-1}}\mathbb{E}[Z(\mathbf{S}_{t_m}, \hat{\alpha}^{\beta}_{t_m})|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)], \tag{3.8}$$

where $\mathbf{S}_{t_{m-1}}(n) \in \mathcal{B}_{t_{m-1}}(\beta)$, $n = 1, ..., N$, $\beta = 1, ..., \nu$. Using Equation (3.5), this can be written as:

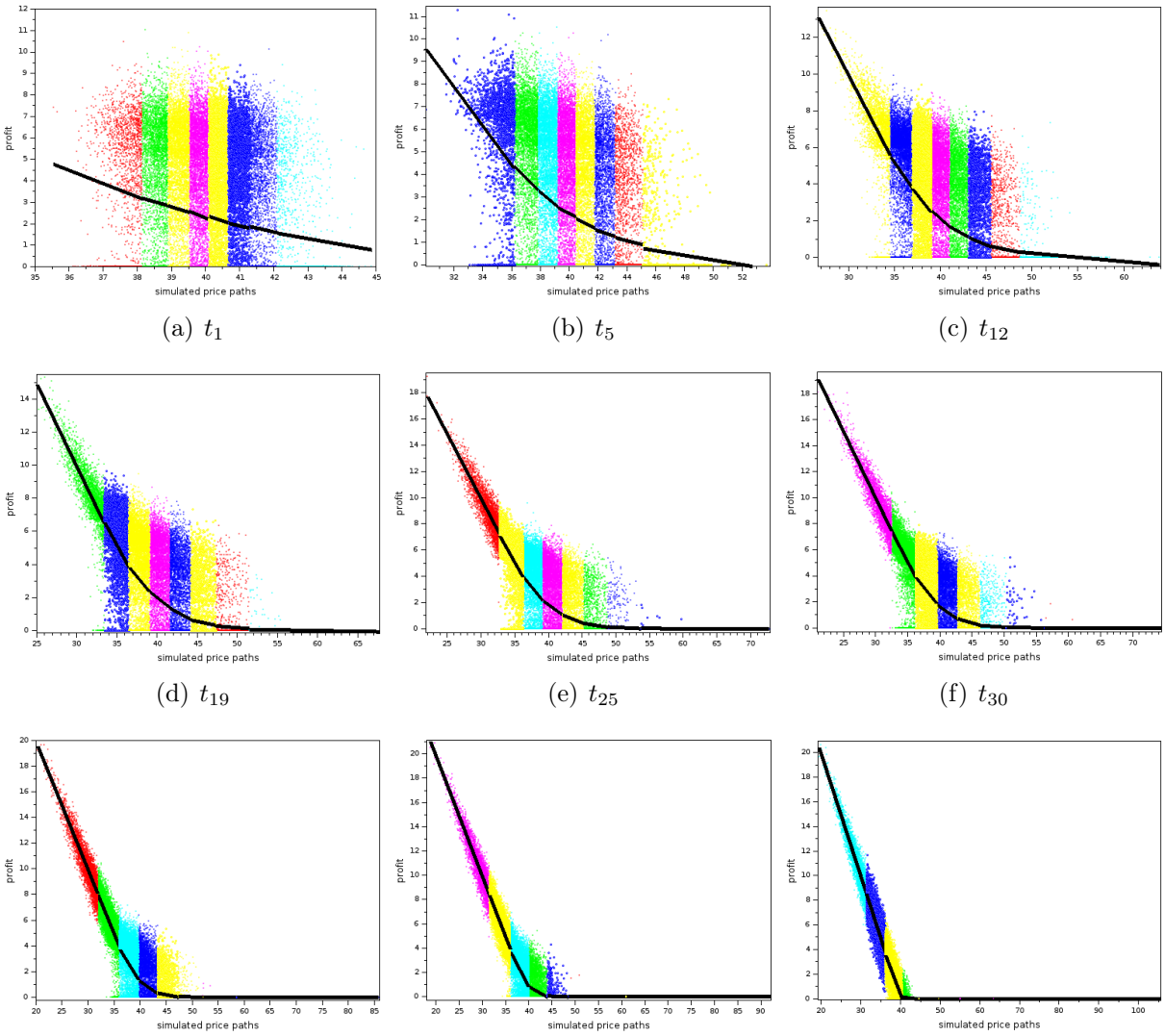
$$\begin{aligned}\hat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) &= D_{t_{m-1}}\mathbb{E}\left[\left(\sum_{k=1}^{K}\hat{\alpha}^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m})\right)|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)\right] \\ &= D_{t_{m-1}}\sum_{k=1}^{K}\hat{\alpha}^{\beta}_{t_m}(k)\mathbb{E}[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)].\end{aligned} \tag{3.9}$$

The **direct estimator** of the option values for the paths at t_{m-1} is defined as :

$$\hat{V}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \max(h(\mathbf{S}_{t_{m-1}}(n)),\ \hat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n))),$$

where $n = 1, \ldots, N$. The direct estimator is said to be an upper bound, $\mathbb{E}[\hat{V}_{t_0}(\mathbf{S}_{t_0})] \geq V_{t_0}(\mathbf{S}_{t_0})$, converging to the true price when simulating an increasing number of paths and using an increasing number of bundles to cluster paths at each time step.

We remark here that the theoretical support provided in [7] to show that the direct estimator is an upper bound (biased high) is based on the assumption that $\mathbb{E}[\epsilon^{\beta}_{t_m}|\mathbf{S}_{t_{m-1}}(n)] = 0$, which is theoretically right. However, numerically, if the regression is not perfectly performed, we will not obtain such an unbiased estimate in (3.7). Moreover, at this point, Jain and Oosterlee [7] indicate choosing the vector of basis functions ϕ such that it represents the most salient properties of a given state and ideally such that $\mathbb{E}[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)]$



(a) t_1 (b) t_5 (c) t_{12}

(d) t_{19} (e) t_{25} (f) t_{30}

(g) t_{38} (h) t_{44} (i) $t_{M-1} = t_{49}$

Figure 3.6: Regression evolution at different epochs t_m. Corresponding dates are displayed under each plot.

is known in a closed form, or has an analytic approximation, to use it. They point that $h(\cdot)$ is usually an important basis function and we employ their same vector of basis functions, but we do not utilize the closed form of $\mathbb{E}[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)]$ corresponding to each kind of option pricing. We use directly the first right hand side in (3.9). Therefore, it is not strange that, in our numerical examples, direct estimator is not sometimes an upper bound as it is shown in [7] experiments.

3.1.6 Computing path estimator

After finishing previous procedure to compute the direct estimator, we simulate a new set of paths and we develop a lower bound estimator based on these new paths. Using the same scheme followed in prior generation of grid points, we simulate $\mathbf{S}(n) =$

$\mathbf{S}_{t_1}(n),\ \ldots,\ \mathbf{S}_{t_M}(n), n = 1,\ \ldots,\ N_L$. Along each path, the approximate optimal policy exercises at,

$$\hat{\tau}^*(\mathbf{S}(n)) = \min\{t_m : h(\mathbf{S}_{t_m}(n)) \geq \hat{Q}_{t_m}(\mathbf{S}_{t_m}(n)), m = 1,\ \ldots,\ M\},$$

where $\hat{Q}_{t_m}(\mathbf{S}_{t_m}(n))$ is computed using Equation (3.9), which means to compute another backward loop including bundling again, but using directly the parameters $\alpha^{\beta}_{t_m}$ stored before for the regression step.

The **path estimator** $\underline{V}_{t_0}(\mathbf{S}_{t_0})$, lower bound respect to the true option value, is :

$$\underline{V}_{t_0}(\mathbf{S}_{t_0}) = \lim_{N_L} \frac{1}{N_L} \sum_{n=1}^{N_L} h(\mathbf{S}_{\hat{\tau}^*(\mathbf{S}(n))}) \leq V_{t_0}(\mathbf{S}_{t_0}).$$

3.1.7 Pros and cons of SGBM

Other advantages of this method include the possibility to use it to compute a duality-based upper bound estimator and its capacity to compute Greeks at the same time, typically requires much more computing time than pricing the contract. However, due to the limited period of the internship, we have not had the time to tackle these powerful properties.

As a 'weakness' of the SGBM presented in [7] we can mention the closed formulas utilized while pricing different payoffs. It is necessary to compute the moments involved in calculation of the continuation value differently each time that we want to experiment with another payoff. We avoid the necessity of the moments using directly (3.9) first right hand side instead of the second one, and we employ the same method to price all the payoffs present in the 'Numerical experiments' section.

3.2 Numerical experiments

We illustrate in this section the performance of the SGBM by pricing different types of Bermudan options and European options. We compare our results with the ones found by S. Jain and C. W. Oosterlee [7] and we discuss for different number of assets priced the role of bundling in computing the option price. The pricing results of SGBM are also compared against other methods.

In following examples, all underlying assets follow the standard single or multi-asset Black-Scholes model, see Appendices A and B. Unless specified otherwise, we use MERSENNE as PNL random generator; $N = 50,000$ paths for computing the calibration, the direct estimator, and the early exercise policy; and $N_L = 200,000$ paths for computing the path estimator. In our code, we can easily choose to use a completely different set of paths for the path estimator or to share the paths simulated for the direct estimator. As in initial tests, we found better pricing results sharing the paths, the first 50,000 paths are common for the examples shown here. In case of the k-means clustering algorithm, we fix to 100 the maximum number of iterations before to stop without convergence.

In addition, we point out that, due to the fine scale employed, sometimes a mili-scale, we may not consider the prices achieved as precise as in fact they are. But we believe that a fine scale is a better choice to appreciate special features of every plot.

The parameter sets used for the different problems are presented in Table 3.1. Remark that in Premia interface there is an "Annual interest rate" parameter, see Figures 2.5 and 2.6; whereas for us, r, the drift of the process for the stock, is continuous compounding[1]. So we have to use the formula $(1+R)^T = e^r$, where T is the maturity, to use the same cases. For example, for $r = 0.06$ and $T = 1$, we have $R = e^{0.06} - 1 = 0.06183654 \rightarrow 6.183654\%$ as Premia parameter.

	$S^{\delta}_{t_0}$	K	r	q_δ	σ_δ	ρ_{ij}	T	M
Set 1	40	40	0.06	-	0.2	-	1	50
Set 2	40	40	0.06	0	0.2	0.25	1	10

Table 3.1: Parameter values used in the examples.

3.2.1 Bermudan options on single asset

We begin with a Bermudan put on a single asset, where the risk-neutral asset price follows the stochastic differential equation

$$dS_t = rS_t dt + \sigma S_t dW_t, \tag{3.10}$$

r being the continuously compounded risk-free interest rate, σ the annualized volatility (both constant), W_t is the standard Brownian motion. The option can be exercised a finite number of times per year, M, including the final expiration time $t_M = T$. As basis functions we use $\phi_k(\mathbf{S}_{t_m}) = \mathbf{S}^{k-1}_{t_m}$, where $k = 1, \ldots, 3$, unless specified otherwise.

First, the convergence of the three clustering algorithms and their corresponding computational times are compared. Figures 3.7 (a) and (b) show the convergence with an increasing number of bundles for the two schemes. Being in dimension one, it needs to be noticed that for recursive bifurcation (RB) and recursive bifurcation of reduced state space (RBRSS) we obtain the same pricing results because the mapping function does not change the actual state space, we already are in dimension 1. Therefore we show their convergence results all at once, but we display both computational times, Figure 3.7 (c), because they are not exactly the same due to algorithm performances. A highly accurate option price, computed using the COS method [3], is given as reference in [7], so we also use it as reference. We add equally as reference here the price computed with Monte Carlo (MC) Longstaff Schwartz method in Premia for Black-Scholes options on single asset. Figures 3.7 (c) and (d) compare the total computational times. Fast convergence with increasing bundles for lower computational time makes recursive bifurcation methods the prioritized methods in this case. In

[1] "The process of earning interest on top of interest. The interest is earned constantly, and immediately begins earning interest on itself." investopedia.com

case of the k-means clustering algorithm, [7] establishes first the use of a set of 5,000 training paths to obtain the optimal centroids corresponding to each bundle. However, it is not indicated if the time to do such optimization is taken into account in the computational time displayed. Differences in k-means clustering computational time between 3.7 (c) and (d) are supposed to come from this point, because we just take the centroids aleatory.

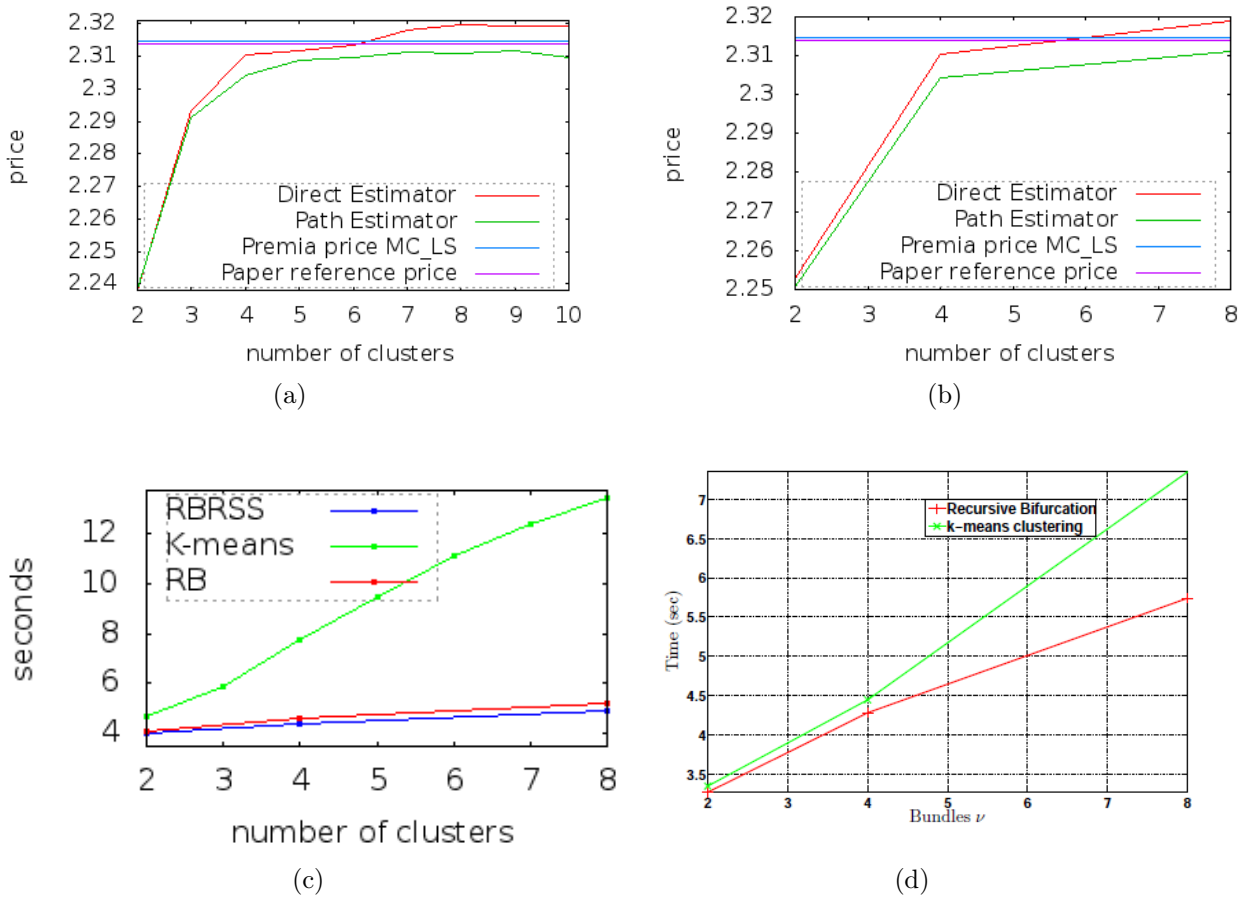


Figure 3.7: Option price for a put on a single asset, corresponding to different numbers of bundles used, when (a) k-means clustering is used and (b) recursive bifurcation scheme is used to partition the state space. Parameter set 1 in Table 3.1 is employed. The true option value is 2.3140. (c) Total computational time with our algorithms and (d) total computational time presented in [7].

We have also studied how affected by the number of basis functions (approximation dimension) the pricing with SGBM is. Figures 3.8 (a) and (b) show the evolution of the price w.r.t. this parameter for the two methods. In Figures 3.8 (c) and (d) we display the approximation absolute error, $|\text{RefPrice} - \text{Estimator}|$, where the vertical bars denote the absolute value and RefPrice the paper reference price, 2.3140.

The convergence of the bundling schemes when increasing the number of Monte Carlo simulated paths is additionally studied. Figures 3.9 (a) and (b) show the estimators evolution with an increasing number of Monte Carlo simulations for the two methods. We keep proportional the relation between N, the number of paths for computing the calibration

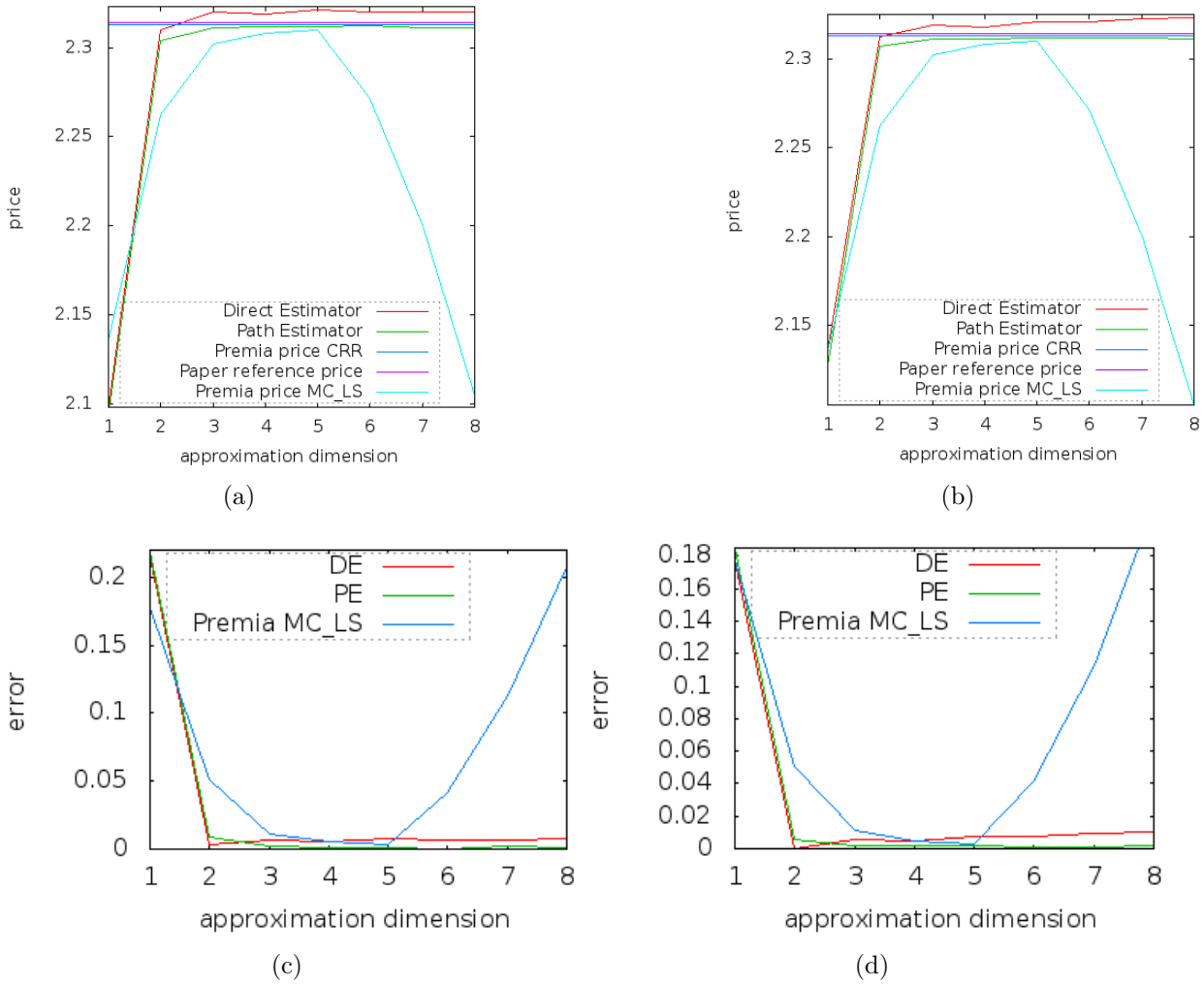
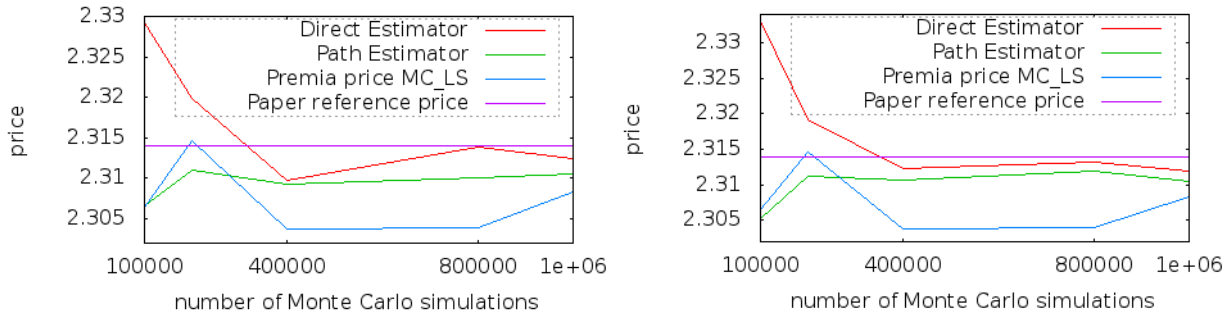


Figure 3.8: Option price for a put when modifying the number of basis functions to parametrize when (a) k-means clustering is used and (b) RB scheme is used to partition the state space. (c) Approximation absolute error with k-means clustering and (d) recursive bifurcation scheme. Parameter set 1 in Table 3.1 is employed and the number of clusters is fixed to 8. MC Longstaff Schwartz method in Premia is also compared.

and direct estimator, and N_L, the number of paths for computing the path estimator, so that $N_L = 4 \times N$. Graphics x axis represents N_L. We validate that compared to another regression-based method, the Least Squares Method, Longstaff and Schwartz (2001) [10], the approximate option values computed using SGBM have lower numerical noise.

Finally, we examine variance of direct and path estimators. It is pointed in [7] that the direct estimator (DE) has a significantly lower variance when compared to the path estimator (PE), however for our version we find quite similar low variances for both of them, Table 3.2. Figure 3.10 displays a series of $1,000$ computed direct and path estimators using a random generator whose seed is initialised aleatory each iteration with the current calendar time (on Unix systems, seconds since 1st January 1970). Parameter set 1 in Table 3.1 is employed and the number of clusters is fixed to 8 for this experiment.



(a) (b)

Figure 3.9: Option price for a put when increasing the number of MC simulations when (a) k-means is used and (b) RB scheme is used to partition the state space. Parameter set 1 in Table 3.1 is employed and the number of clusters is fixed to 8. MC Longstaff Schwartz method in Premia is also compared.

	Mean DE	**Var DE**	**Mean PE**	**Var PE**
k-means	2.31832	1.4317e-04	2.31101	3.7988e-05
RB	2.31845	1.5182e-04	2.31151	3.7484e-05

Table 3.2: Means and variances values for direct and path estimators with k-means clustering and recursive bifurcation schemes. The sample size is 1,000. The true option value is 2.3140.

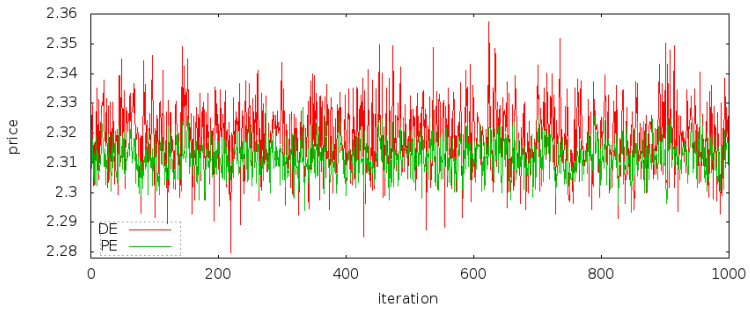


Figure 3.10: Plot of a series of 1,000 computed direct and path estimators.

3.2.2 Geometric Basket Option

Consider now the pricing of a Bermudan option on the geometric average of d assets, the put has intrinsic value :

$$h(\mathbf{S}_{t_m}) = K - (\prod_{\delta=1}^{d} S_{t_m}^{\delta})^{\frac{1}{d}}.$$

The asset prices are assumed to follow correlated geometric Brownian motion processes, i.e.

$$\frac{dS_t^{\delta}}{S_t^{\delta}} = (r - q_{\delta})dt + \sigma_{\delta} dW_t^{\delta}, \tag{3.11}$$

where each asset pays a dividend at a continuous rate of q_{δ}, again different with respect to the Premia interface "Annual dividend rate" parameter. W_t^{δ}, $\delta = 1, \ldots, d$, are standard

Brownian motions and ρ_{ij} is the instantaneous correlation coefficient between W_t^i and W_t^j, see Appendix A and B. As basis functions we use

$$\phi_k(\mathbf{S}_{t_m}) = \left((\prod_{\delta=1}^{d} S_{t_m}^{\delta})^{\frac{1}{d}} \right)^{k-1}, k = 1, \ \ldots, \ 3.$$

As a benchmark result for the algorithm, [7] takes the price of the problem reduction to a one-dimensional problem, using again the COS method [3]. We also take that price as reference.

For a five-dimensional problem, Figure 3.11 shows the convergence of the DE and PE with an increasing number of clusters for the different bundling schemes. For the recursive bifurcation of the reduced state space (RBRSS), the geometric mean of the asset prices is used to map the high-dimensional state space. RBRSS leads to better convergence, is computationally most efficient, and, has a great flexibility on the choice of the number of bundles. There is no graphic for RB because it would be just a point corresponding to estimators when we take the $2^d = 2^5 = 32$ clusters resulting in the first bifurcation, so we could not appreciate any convergence. Figure 3.11 (c) and (d) compare the computational times corresponding to the bundling schemes used.

The convergence of the bundling schemes when increasing the number of Monte Carlo simulated paths for the five-dimensional problem is studied as for Bermudan options on single asset. Figures 3.12 (a), (b) and (c) show the estimators evolution with an increasing number of Monte Carlo simulations for the three clustering methods.

Figures 3.13 (a) and (b) compare the convergence for a geometric basket on 15 assets, when k-means and RBRSS are used for clustering. We do not use RB here due to the high dimension of the problem. We obtain $2^d = 2^{15} = 32,768$ bundles just with one iteration of RB, so having $N = 50,000$ grid points, a significant number of the clusters do not contain enough points. Figures 3.13 (c) and (d) give the total computational time for the two methods. Notice that both k-means computational times are comparable, however, our RBRSS computational time halve its peer in the paper. We study as well in this case the convergence of the bundling schemes when increasing the number of Monte Carlo simulated paths, Figures 3.14 (a) and (b).

Table 3.3 compares direct and path estimator variances and Table 3.4 displays the results when using two different established random generators of PNL for the 15 assets case. The results correspond to 32 bundles.

	Mean DE	**Var DE**	**Mean PE**	**Var PE**
k-means	1.11854	4.2740e-05	1.11468	1.0243e-05
RBRSS	1.12125	4.0163e-05	1.117	1.0817e-05

Table 3.3: Means and variances values for direct and path estimators with k-means clustering and RBRSS scheme. The sample size is $1,000$. The true option value is 1.1190.

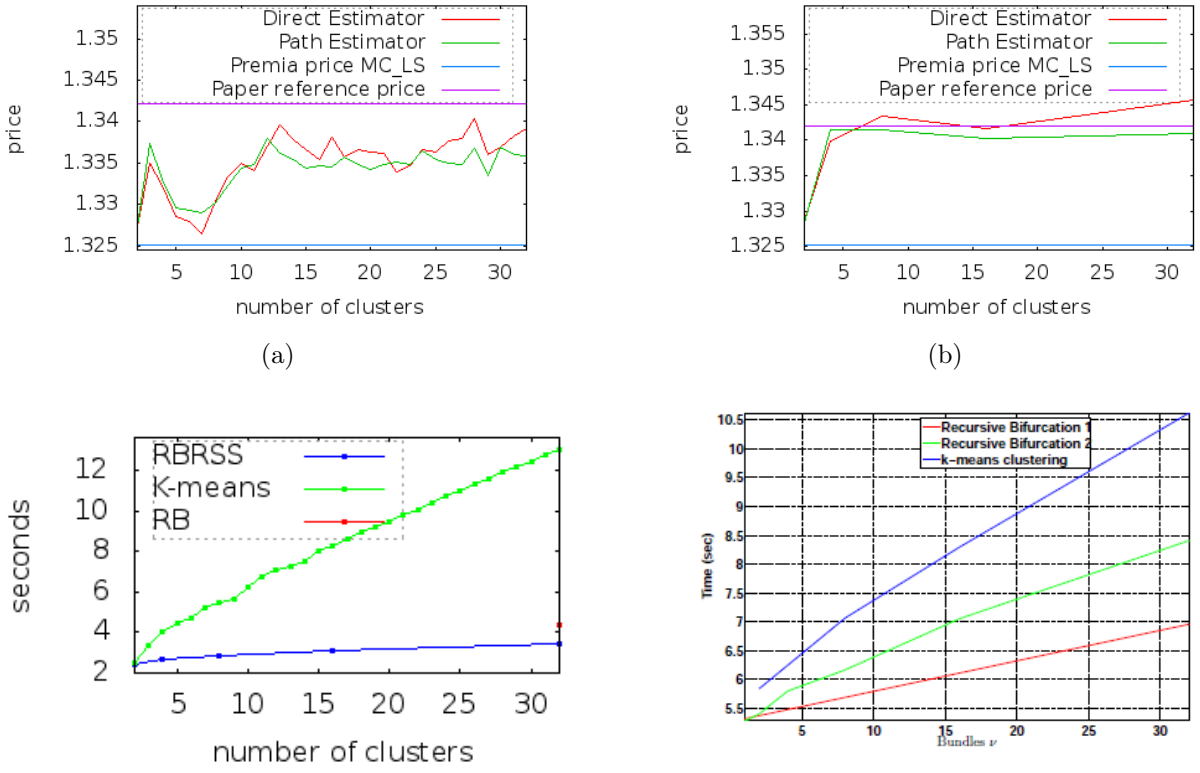


Figure 3.11: Put on geometric average of 5 assets pricing when (a) k-means is used, and (b) RBRSS is used, to partition the state space. Parameter set 2 from Table 3.1 is employed. The true option price is 1.3421. (c) Total computational times with our algorithms and (d) total computational times presented in [7].

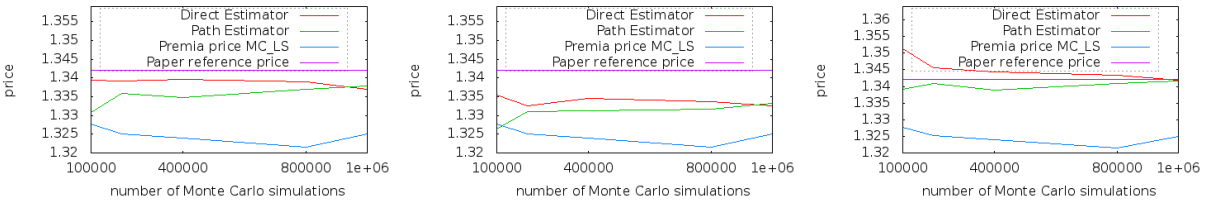


Figure 3.12: Option price for a put when increasing the number of MC simulations when (a) k-means is used, (b) RB in high-dimensions is used and (c) RBRSS is used, to partition the state space. Parameter set 2 in Table 3.1 is employed and the number of clusters is fixed to 32. MC Longstaff Schwartz method in Premia is also compared.

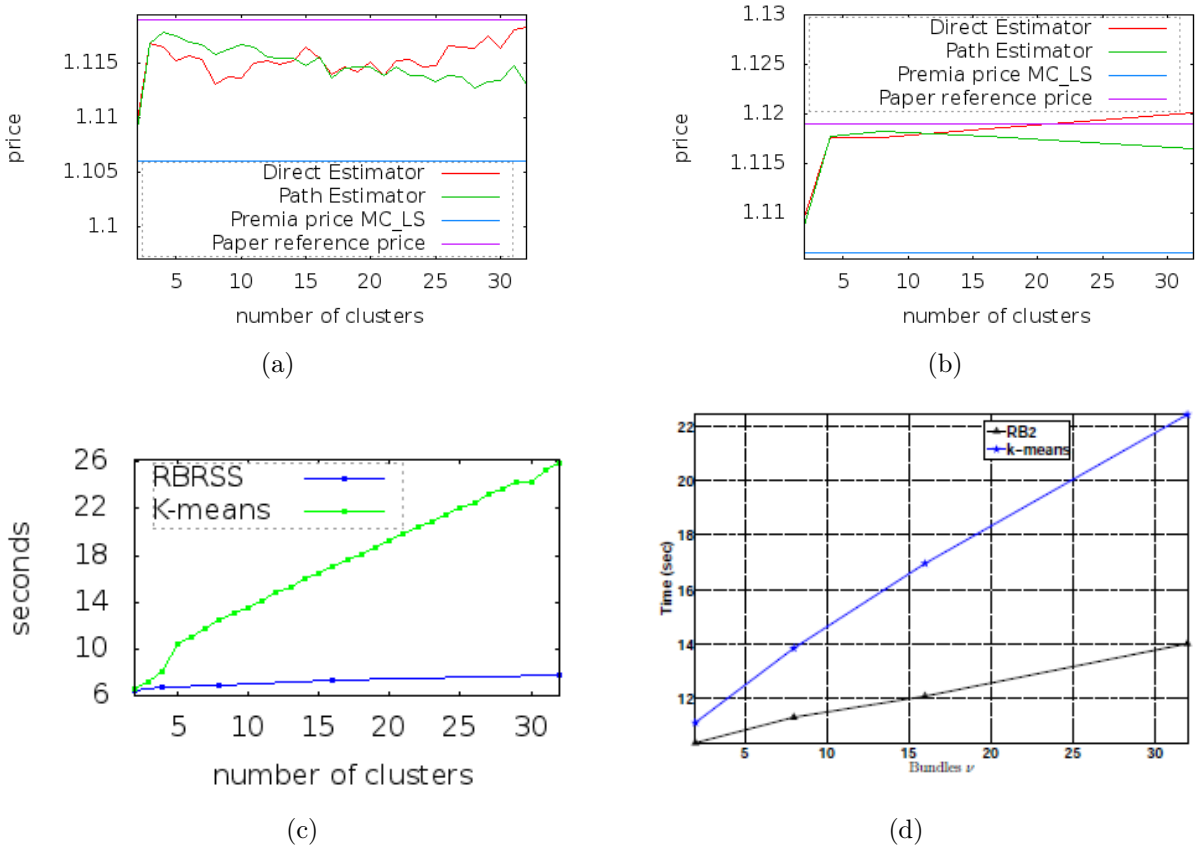


Figure 3.13: Option value for a put on geometric average of 15 assets, when (a) k-means is used and (b) RBRSS scheme is used, for bundling. The reference option price is 1.1190. Parameter set 2 in Table 3.1 is employed. Total computational times (c) with our algorithms and (d) presented in [7].

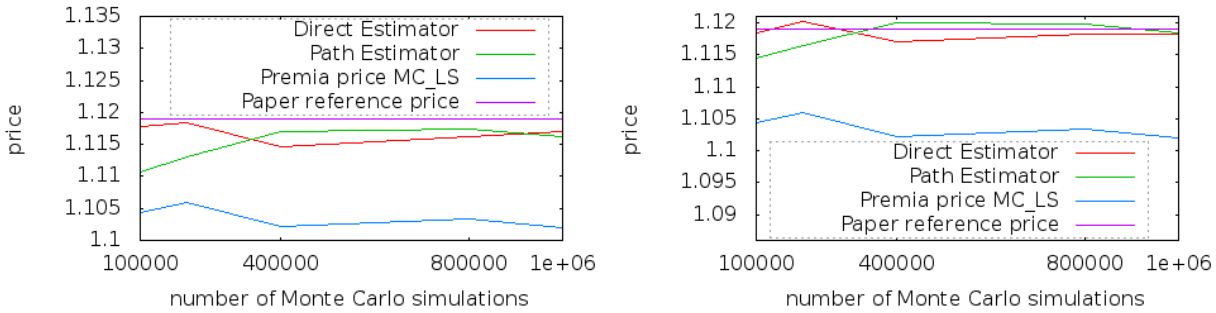


Figure 3.14: Option price for a put when increasing the number of MC simulations when (a) k-means clustering is used, and (b) RBRSS is used, to partition the state space. Parameter set 2 in Table 3.1 is employed and the number of clusters is fixed to 32. MC Longstaff Schwartz method in Premia is also compared.

	k-means DE	RBRSS DE	k-means PE	RBRSS PE
Knuth	1.1232	1.12415	1.11563	1.11585
Mersenne	1.1184	1.12013	1.11312	1.11652

Table 3.4: Direct and path estimators for Knuth and Mersenne generators with k-means and RBRSS.

3.2.3 Arithmetic Basket Option

The intrinsic value function for an arithmetic average put option on $d-$assets is given by :

$$h(\mathbf{S}_{t_m}) = K - (\frac{1}{d}\sum_{\delta=1}^{d} S_{t_m}^{\delta}).$$

The asset prices follow Equation (3.11) dynamics and as basis functions we use :

$$\phi_k(\mathbf{S}_{t_m}) = (\frac{1}{d}\sum_{\delta=1}^{d} S_{t_m}^{\delta})^{k-1}, k = 1, \ \ldots, \ 3.$$

Figure 3.15 (a) and (b) show the convergence of the method with an increasing number of bundles for an arithmetic basket on 15 assets. We just consider k-means and RBRSS for bundling, as in the case of RB we have the same problem previously commented of a too high number of clusters. The arithmetic mean of the asset prices is used to map the high-dimensional state space for RBRSS. Figure 3.15 (c) displays the computational times corresponding to different numbers of bundles, which is still in seconds, with the RBRSS being computationally most efficient.

3.2.4 European options

Additionally, in order to use the method to price as much as possible different products, we study the pricing of European options on $d-$assets, where the asset prices follow the dynamics given by Equation (3.11). Another important reason to comprise the computational results of European basket options is that it has helped us to guarantee the correct implementation of certain parts of the code, since it is a more basic case (see Appendix D). As for Bermudan arithmetic basket options, the intrinsic value function of an European put option on $d-$assets, is

$$h(\mathbf{S}_{t_M}) = K - (\frac{1}{d}\sum_{\delta=1}^{d} S_{t_M}^{\delta}),$$

but it can be exercised only at its maturity, $t_M = T$. The product's pay-off is $V_T(\mathbf{S}_T) = \max(h(\mathbf{S}_{t_M}), \ 0)$. The difference resides in the possible exercise dates. We also use the same basis functions as for the arithmetic Bermudan options and the arithmetic mean of the asset prices to map the high-dimensional state space to the single-dimensional space for recursive bifurcation of reduced state space.

Figure 3.16 (a) shows the convergence of the method with an increasing number of bundles

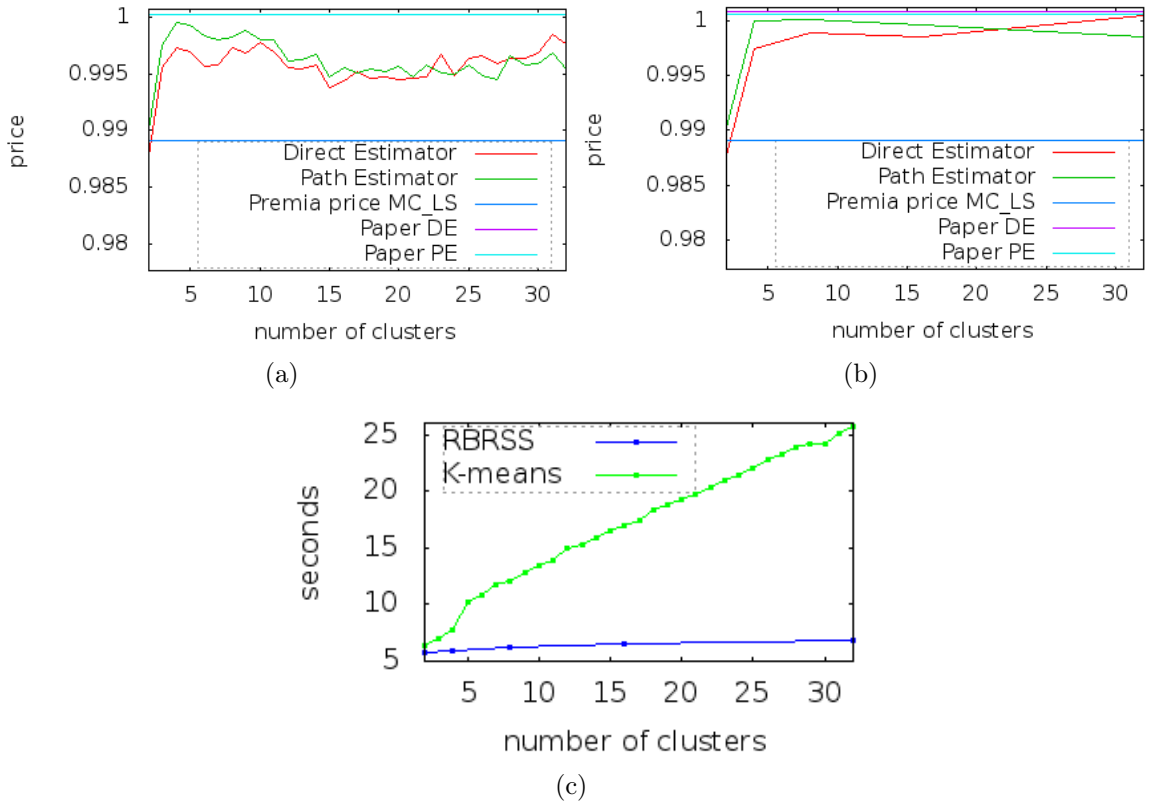


Figure 3.15: Option value for a put on arithmetic average of 15 assets, when (a) k-means clustering is used and (b) RBRSS scheme is used, for bundling. The paper results reported correspond to the case of 32 bundles. The parameter values from set 2 in Table 3.1 are employed. (c) Computational times corresponding to different numbers of bundles.

for an European basket on 5 assets and, equally, in (b) for an European basket on 15 assets. The pricing of European options is not affected by the clustering method and by the number of clusters neither, so neither by the regression, which can be better understood with pseudocodes in Appendix D.

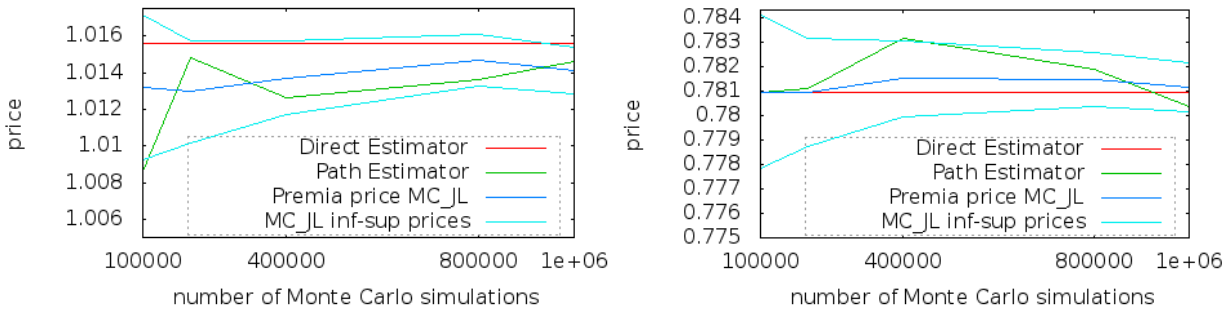


Figure 3.16: Option price when increasing the number of MC simulations for an European put on (a) 5 assets, and (b) 15 assets. Parameter set 2 in Table 3.1 is employed. Monte Carlo Jourdain Lelong method in Premia is also compared, using 95% confidence interval.

4 The SGBM: Monte Carlo Calculation of Exposure Profiles for Bermudan Options

This task of my internship has consisted of, firstly, understand what the exposure and the Credit Valuation Adjustment (CVA) are, why it is so important to compute them these days, and how we can compute exposure profiles using the advantageous structure of the SGBM as proposed by Feng and Oosterlee (2014) [4]. In order to show the SGBM working on this purpose, we apply it to European and Bermudan multiple asset derivative contracts under Black-Scholes dynamics and options with a single asset under stochastic volatility dynamics, the Heston model. There is also a focus on the use of the SGBM for computation of exposure profiles of options with a single asset under stochastic interest rate dynamics, the Heston Hull-White model (HHW), in [4]. The development of this case is contemplated as well previous to the end of the internship, but it will not be included in the present report.

Following the same planning as [4], we start introducing the concepts and mathematical formulation of CVA and exposure and subsequently we explain the base to employ the SGBM to compute the CVA.

4.1 CVA and exposure

The introduction of Basel III[1] has made the pricing of CVA a fundamental duty to all the players in financial world these days. There are three key elements in assessing CVA :

(1) Loss given default, which is the percentage of loss in a default event.

(2) Expected exposure, which quantifies the expectation of the losses at a future moment.

(3) Probability of the counterparty default.

We assume here independence of these three elements, but in a general real-life situation, they are correlated.

The credit exposure is the economic loss for a contract holder in case of counterparty default without any recovery. The problem comes because the market moves unpredictably

[1] A comprehensive set of reform measures designed to improve the regulation, supervision and risk management within the banking sector. The Basel Committee on Banking Supervision published the first version of Basel III in late 2009, giving banks approximately three years to satisfy all requirements. Largely in response to the credit crisis, banks are required to maintain proper leverage ratios and meet certain capital requirements. `http://www.investopedia.com/terms/b/basell-iii.asp`

and so the future exposure value is uncertain. A solution is to create an empirical exposure density by Monte Carlo simulation of a big number of asset paths. Afterwards, we compute the market values of the contract for each grid point, for example, as with the SGBM. Thus, we have all the information needed to calculate the exposure at each grid point as well.

As it is logical, the contract holder suffers a loss only if the derivative has a positive market value when the counterparty defaults. Therefore, the option exposure is defined as,

$$E(t, \mathbf{X}_t) := \max(V(t, \mathbf{X}_t), 0), \tag{4.1}$$

where $V(t, \mathbf{X}_t)$ is the option value at time t depending on market state variable $\mathbf{X}_t$, which follows the corresponding stochastic dynamics (Black-Scholes or Heston in our case).

The expectation of the exposure, Expected Exposure (EE), is an estimate of the expected value of loss and its mathematical expression conditioned on the initial market state is :

$$EE(t) := \mathbb{E}^{\mathbb{Q}}[E(t, \mathbf{X}_t) \mid \mathbf{X}_0], \tag{4.2}$$

where $\mathbb{Q}$ is the risk-neutral measure. The discounted EE at time t is then given by :

$$EE^*(t) := \mathbb{E}^{\mathbb{Q}}\left[D(0,t)E(t, \mathbf{X}_t) \mid \mathbf{X}_0\right], \tag{4.3}$$

where the discount factor is defined by

$$D(s,t) := \exp\left(-\int_s^t r_u du\right), \ s < t, \tag{4.4}$$

with the interest rate r_u at time u.

As we are confronted with Black-Scholes and Heston models, where the interest rate is deterministic, it can be written $EE^*(t) = D(0,t)EE(t)$. EE(t) represents the future exposure and $EE^*(t)$, the current value of the future exposure.

The formula to compute the CVA provided in [5], assuming independence between exposure values and default events, is written as follows :

$$CVA = (1-\delta)\int_0^T EE^*(t)dPD(t), \tag{4.5}$$

where δ is the recovery rate and PD(s) is the default probability function,

$$PD(t) = 1 - \exp\left(-\int_0^t h(t)dt\right), \tag{4.6}$$

where $h(t)$ is called the intensity.

As simulation is done in a discrete time grid, we need a discrete version formula to compute the CVA, which can be given as :

$$CVA \approx (1-\delta)\sum_{m=0}^{M-1} EE^*(t_m)(PD(t_{m+1}) - PD(t_m)). \tag{4.7}$$

4.2 SGBM to compute CVA

The holder receives the payoff value, $g(S_m)$, when the option is exercised. When the option contract is still alive ($t_m < \tau_m$), the discounted option value, the continuation value w.r.t. state vector $\mathbf{X}_m$, is

$$\hat{Q}_m(\mathbf{X}_m) := \mathbb{E}^{\mathbb{Q}}\left[V_{m+1}(\mathbf{X}_{m+1}) \cdot D(t_m, t_{m+1})|X_m\right], \tag{4.8}$$

where $V_{m+1}(\cdot)$ represents the option value at time t_{m+1}.

For European options, denoting $S_m = \exp(x_m)$ as the underlying asset variable at time t_m, the option value equals the continuation value before maturity and the holder receives the payoff value only at maturity, i.e.

$$V_m^{Euro}(\mathbf{X}_m) = \begin{cases} g(S_M), & \text{for } t_M, \\ \hat{Q}_m(\mathbf{X}_m), & \text{for } t_m \in \mathcal{T} - t_M. \end{cases} \tag{4.9}$$

For Bermudan options, we assume that the credit information of the other party does not influence the exercise decision of the option holder. At each exercise date the holder compares the payoff value with the continuation value of the option, based on the currently available information. The holder keeps the option until the payoff value is higher. When the option is still alive at time t_m, denoting $\mathcal{T}_e$ the exercise dates, the option can be computed via

$$V_m^{Berm}(\mathbf{X}_m) = \begin{cases} \max\{\hat{Q}_m(\mathbf{X}_m), g(S_m)\}, & \text{for } t_m \in \mathcal{T}_e, \\ \hat{Q}_m(\mathbf{X}_m), & \text{for } t_m \in \mathcal{T} - \mathcal{T}_e. \end{cases} \tag{4.10}$$

On the other hand, the exposure value becomes 0 when the option is exercised as there is not possible economic loss for the contract holder any longer, $E_M = 0$. Equally, for Bermudan options, after being exercised at time t_m the exposure later is 0. By definition, the value of the exposure can be represented mathematically as :

$$E_m(\mathbf{X}_m) = \begin{cases} 0, & \text{when the option is exercised,} \\ V_m(\mathbf{X}_m), & \text{when the option is alive.} \end{cases} \tag{4.11}$$

where $E_m(\cdot)$ represents the exposure at time t_m, $m = 1, 2, \ldots, M-1$.

Having computed the exposure values for all the simulated paths at times t_m, $m = 0, \ldots, M-1$, the EE value at time t_m is approximated as an average of them :

$$EE(t_m) \approx \frac{1}{N} \sum_{i=1}^{N} E_m(\hat{\mathbf{x}}_m(i)), \tag{4.12}$$

where N represents the number of paths and $\hat{\mathbf{x}}_m(i)$, $i = 1, \ldots, N$, the values of the state variables of the i-th path at t_m. As we previously mentioned, since the interest rate is deterministic in our case, the discounted exposure is the product of the discount factor and the precedent EE value and so we also can obtain the CVA, the direct estimator CVA.

4.2.1 Backward iteration for exposure values

We provide here the procedure for calculating the exposure values in a backward iteration, starting at final time T. This procedure corresponds to the one introduced in [4] as well and uses the results explained in Chapter 3.

At time t_M, as we mentioned before, the exposure values are 0 as there is not possible economic loss for the holder any longer. For each path, the option value is calculated as $V_M(\hat{\mathbf{x}}_M(i))$, with the corresponding formulas of European or Bermudan options, (4.9) and (4.10).

At time t_{M-1}, for all the paths, the continuation values $\hat{Q}_{M-1}(\hat{\mathbf{x}}_{M-1})$, $i = 1, \ldots, N$, can be calculated with (3.9), and so the option values, $V_{M-1}(\hat{\mathbf{x}}_M(i))$, (4.9) or (4.10), and the exposure values, $E_{M-1}(\hat{\mathbf{x}}_M(i))$, (4.11), can be also computed.

Then the iteration goes backward in time repeating the bundling and regression exposed in Chapter 3 to compute the continuation, option, and exposure values at each time step until we arrive to the initial time. When we arrive at t_0, we have the option and exposure values of every grid point.

For Bermudan options, we also need to take into account the optimal early-exercise strategy in this case. If the option is still alive at t_m, both option and exposure values are set to the corresponding continuation value. Then, the payoff value is calculated for each path, and compared with the continuation value to determine if the option should be exercised. If yes, the exposure at this path from time t_m will be 0, and the option value at time t_m corresponds to the payoff value. The EE function is then written as :

$$EE(t_m) \approx \frac{1}{N} \sum_{\tau_i > t_m} (D(t_m, \tau_i) \cdot \text{cash-flow}(i)), \tag{4.13}$$

where τ_i is the exercise time of the i-th path, and the cash-flow is the payoff value at time τ_i, $g(S_{\tau_i}(i))$, with $S_{\tau_i}(i)$ the value of the stock of the i-th path at time τ_i. The EE values calculated with precedent expression, provide the path estimator CVA, which considers the obtained "optimal" exercise strategy. For computing the path estimator CVA, we simulate a new set of paths and we employ again, in a backward procedure, the same regression coefficients for each bundle that we used to compute direct estimator (and that we have previously saved at each time step). In Appendix D we can see the pseudocode with the procedure to compute exposure profiles in the path estimator case.

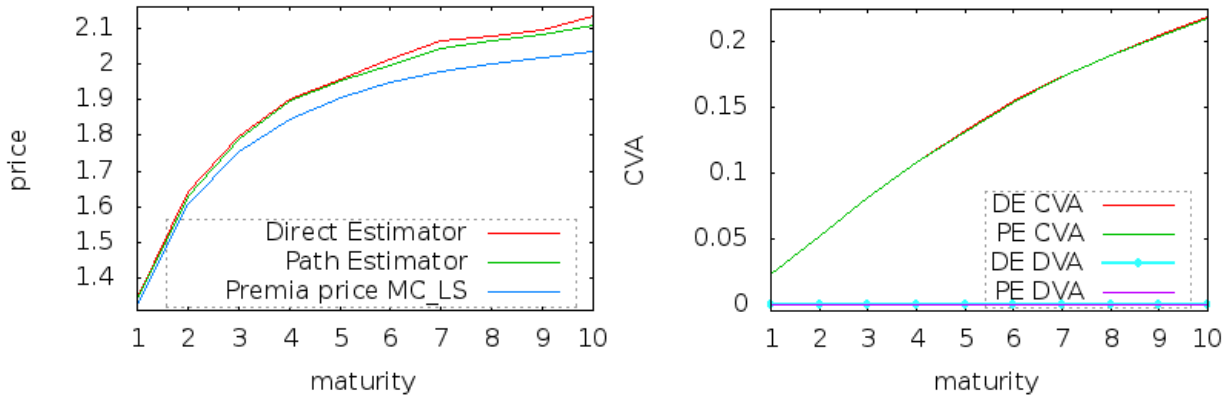
4.3 Black-Scholes model

We retake Bermudan options on the geometric average of several assets exposed in Section 3.2.2, with all underlying assets following the standard multi-asset Black-Scholes model (geometric Brownian motion). We provide also an example of European options.

4.3.1 Numerical experiments

For the examples considered, unless specified otherwise, we use MERSENNE as PNL random generator, $N = 50,000$ paths for computing the direct estimator and the early exercise policy, and $N_L = 200,000$ paths for calculate the path estimator, where the first 50,000 paths are shared. We utilise RBRSS as clustering method and we fix the number of clusters to 32 for all the examples here, because we previously validated it to be the ideal number to have convergence when pricing basket options of 5 assets. We take a recovery rate of 40%, i.e. $\delta = 0.4$, and we set constant the hazard rate, $h = 0.03$, for CVA and DVA computation.

Figure 4.1 (a) shows the evolution of the direct estimator and path estimator prices with an increasing maturity for a Bermudan option on the geometric average of 5 assets. Figure 4.1 (b) displays the corresponding evolution of CVA and DVA values of the direct and path estimator. Parameter set 2 in Table 3.1 is employed, with increasingly maturity and keeping proportional per year the number of option exercise dates, M, i.e. $M = 10 \times T$.



(a) (b)

Figure 4.1: (a) Price and (b) CVA/DVA, for a Bermudan put on 5 assets under Black-Scholes model with increasingly maturity until ten years. Monte Carlo Longstaff Schwartz method in Premia is also compared in (a).

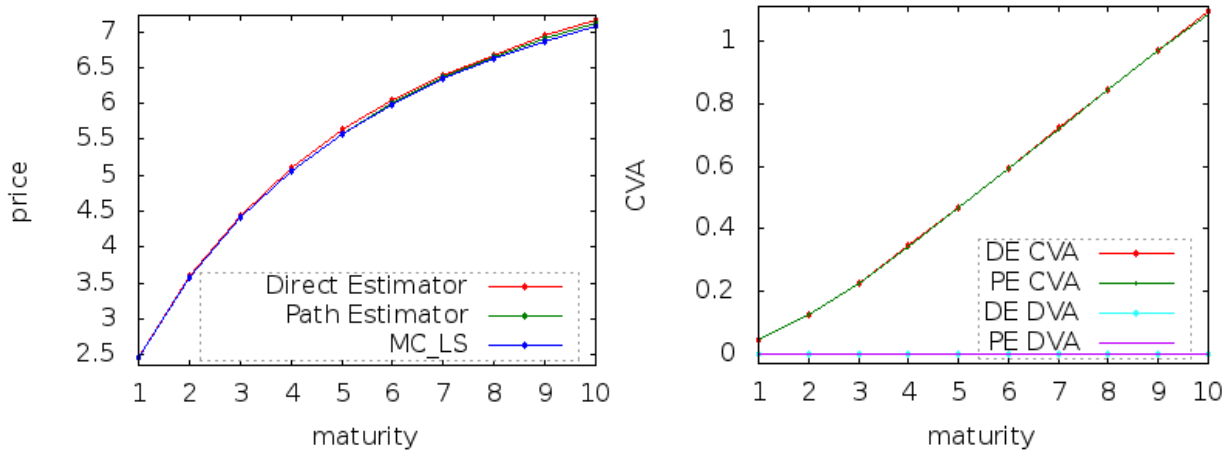
When two firms are both including CVA in fair value of the new trade, they will not agree on the price unless they include the other firm's CVA as well[2]. The term DVA, Debt or Debit Valuation Adjustment, has been adopted to mean the other firm's CVA. DVA is an adjustment used to price OTC (over-the-counter) derivatives trades based on both parties' credit risk. In this way, the expected exposure previously explained to compute CVA is also called Expected Positive Exposure (EPE), i.e. EPE is our average exposure to the counterparty default; and the Expected Negative Exposure (ENE) is the equivalent of EPE from the perspective of our counterparty, i.e. the expected exposure needed to compute DVA. In Figure 4.1 (b), DVA values are always 0 because for this product there is not going to be ENE different from 0. We expected that since we take $\max(\cdot, 0)$ in (4.13) cash-flow formula and so there is no negative part, however, we display DVA as information and to discard errors in the method's code. DVA will be more interesting to compute in the case

[2] `http://www.prmia.org/sites/default/files/references/SokolPresentation.pdf`

of financial products such as swaps[3], where it is possible to have negative cash flows and so, ENE different from 0.

Due to the recent establishment of the necessity to compute the CVA, there is not free sources available to compare CVA/DVA values. Therefore, we have just validated our implementation with cases used in reliable reference papers and prices with Premia.

Figures 4.2 show the same results of Figures 4.1 for the same kind of option and parameters, but including a dividend in this case, $q_d = \log(1.08)$. The equivalent Annual Dividend Rate parameter in Premia for the MC Longstaff Schwartz method, which is also compared in Figure 4.2 (a), is 8%. As it can be appreciate and seems logical, prices are higher and the adjustment CVA too, since loss given default would have more impact.



(a) (b)

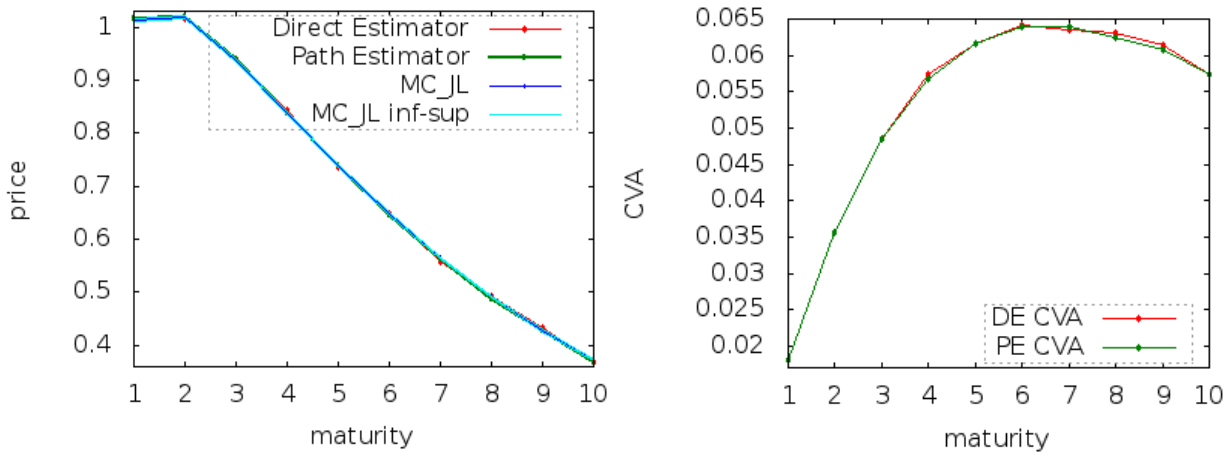
Figure 4.2: Evolution of (a) price (estimators) and (b) CVA/DVA, for a Bermudan put on the geometric average of 5 assets which pays dividends.

In Figure 4.3 (a) we can see the evolution of the direct and path price estimators with an increasing maturity for an European option on the arithmetic average of 5 assets. Monte Carlo Jourdain Lelong method in Premia using 95% confidence interval is also compared in this plot. Figure 4.3 (b) displays the corresponding progression of the CVA value of these estimators. Parameter set 2 in Table 3.1 is employed, with increasingly maturity. This time, the proportionality of the number of option exercise dates is not relevant because the holder receives the payoff value only at maturity.

4.4 Heston model

The Heston stochastic volatility model has been one of the most popular extensions to the Black-Scholes model in finance since its appearance in a paper of Steven L. Heston [6] in 1993. Instead of assuming volatility as a constant, the Heston model assumes that variance,

[3]A swap is a derivative in which two counterparties exchange streams of cash flows.



(a) (b)

Figure 4.3: (a) Price and (b) CVA, for an European put on 5 assets under Black-Scholes model with increasingly maturity until ten years.

or the square of volatility, follows the square root diffusion process (also known as the Cox-Ingersoll-Ross (CIR) (1985) [2] process in interest rate modelling), which has the attractive properties of being non-negative and mean-reverting.

We consider the 2-d state variable $\mathbf{X}_t = [x_t, v_t]^T$, with $x_t = \log(S_t)$ the log-asset variable and v_t the variance variable. The dynamics of the Heston stochastic volatility model, are given by the coupled two-dimensional stochastic differential equations :

$$\begin{aligned} dv_t &= \kappa(\theta - v_t)dt + \gamma\sqrt{v_t}dW_t^v \\ dx_t &= \left(r - \tfrac{1}{2}v_t\right)dt + \sqrt{v_t}dW_t^x, \end{aligned} \tag{4.14}$$

where W_t^x and W_t^v are two correlated Brownian motions in the time variable t, and $dW_t^x \times dW_t^v = \rho_{x,v}dt$. κ, θ, γ are positive constants. κ represents the mean reversion speed for the variance, θ is the mean reversion level for the variance, and γ is the volatility of the variance, called vol-of-vol parameter. r, the drift of the process for the stock or constant interest rate, is a non-negative constant, and the correlation between the two Brownian motions ρ is a constant in $[-1, 1]$. The initial conditions x_0 and v_0 are assumed to be strictly positive.

The option dynamics is a function of the log-asset price and variance. For orders $p = \{0, 1, 2, 3\}$, corresponding to approximation dimensions $\{1, 2, 3, 4\}$, the set of basis functions used in regression step of the SGBM is presented in Table 4.1. Values for the variance are much lower than values for the log-asset, and so, coefficients found in regression would be too much conditioned by the big values. Therefore, before performing regression at each time step, we normalize log-asset and variances values by dividing for the corresponding in each case max simulated value.

4.4.1 Simulation schemes for paths under the Heston model

We explain in this section how we have simulated paths under the Heston model, employing two different schemes. First we used Euler scheme to test the SGBM and fit the necessary

order p of the polynomial space	basis functions
0	$\{ 1 \}$
1	$\{ 1, x, v \}$
2	$\{ 1, x, v, x^2, x \cdot v, v^2 \}$
3	$\{ 1, x, v, x^2, x \cdot v, v^2, x^3, x^2 \cdot v, x \cdot v^2, v^3 \}$

Table 4.1: The basis functions of order p.

changes with respect to the Black-Scholes SGBM algorithm implementation. We simulate $x_t = \log(S_t)$, the log-asset variable, and v_t, the variance variable, following :

$$\begin{cases} v_0 &= V_0 \\ v_{i+1} &= v_i + \kappa\ (\theta - v_i)\ \Delta t_i + \gamma\ \sqrt{v_i}\sqrt{\Delta t_i} W_i^v \end{cases}$$

$$\begin{cases} x_0 &= X_0 \\ x_{i+1} &= x_i + (r - q - \frac{1}{2} v_i)\Delta t_i + \sqrt{v_i}\sqrt{\Delta t_i} W_i^x \end{cases}$$

where q is the dividend paid at a continuous rate. Δt_i means $t_{i+1} - t_i$, and x_i and v_i represent x_{t_i} and v_{t_i}, respectively. As we mentioned while introducing this model, the initial conditions x_0 and v_0 are assumed to be strictly positive.

We notice that we could have problems if $v_i < 0$ because it is inside a square root. To avoid this possibility, we take $|v_i|$ inside the square root. There is also an alternative solution for this situation consisting in taking $\max(v_i, 0)$ inside the sqrt function. These are acceptable solutions and provide a valid simulation, used even in real world by banks. However, there is a better approach.

Once we proved the SGBM was completely adapted to the Heston model requirements, we changed the Euler scheme for the one proposed by Damiano Brigo and Aurélien Alfonsi[4] in [1]. In order to ensure positivity of v_i, they explain their Euler Implicit Positivity-Preserving Scheme, which consist in simulate v_{i+1} as follows :

$$v_{i+1} = \left(\frac{\gamma(W_{i+1}^v - W_i^v)\ +\ \sqrt{\gamma^2 (W_{i+1}^v - W_i^v)^2\ +\ 4(v_i\ +\ (\kappa\theta - \frac{\gamma^2}{2})\Delta t_i)(1\ +\ \kappa\Delta t_i)}}{2(1\ +\ \kappa\Delta t_i)} \right)^2,$$

where $(W_{i+1}^v - W_i^v) = \sqrt{\Delta t_i}\ W_i^v$. We just need to respect $2\kappa\theta > \gamma^2$ to use this scheme, which is a much less restrictive condition.

4.4.2 Numerical experiments

For the cases exposed in this section, unless specified otherwise, we use again MERSENNE as PNL random generator, $N = 50,000$ paths for computing the direct estimator and the early exercise policy, and $N_L = 200,000$ paths for computing the path estimator, with the first 50,000 paths shared. We fix the number of clusters to 8 for all the examples here because, as we appreciate for pricing Bermudan options on a single asset in previous chapter,

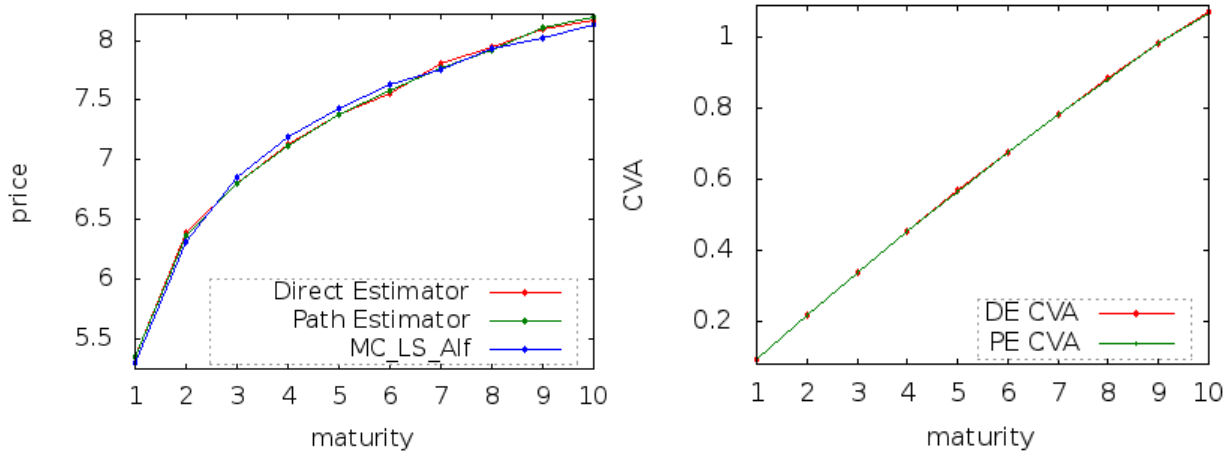
[4] A. Alfonsi is part of the Applied Probability group in the CERMICS as researcher.

it is the ideal clustering to converge to an accurate price in this case. Additionally, we do not utilize any more recursive bifurcation of reduced state space because we just have an asset. We take a recovery rate of 40%, i.e. $\delta = 0.4$, and we set constant the hazard rate, $h = 0.03$, for CVA computation. As DVA is still always 0, we do not display it this time. The parameter set employed for options under the Heston model is presented in Table 4.2.

S_{t_0}	V_{t_0}	q	r	ρ	κ	θ	γ	T	K	M
40	0.2	0	log(1.05)	0.25	1	0.07	0.1	1	40	24

Table 4.2: Parameter values for the Heston model experiments.

Figure 4.4 (a) shows the evolution of the direct estimator and path estimator prices with an increasing maturity for a Bermudan option on a single asset under the Heston model. Figure 4.4 (b) displays the corresponding evolution of the CVA value of the direct and path estimator.



(a) (b)

Figure 4.4: (a) Price and (b) CVA, for a Bermudan put on a single asset under the Heston model. MC Longstaff Schwartz method in Premia is also compared in (a). Parameters values in Table 4.2 are employed, with increasingly maturity until ten years and keeping proportional per year the number of exercise dates, $M = 24 \times T$.

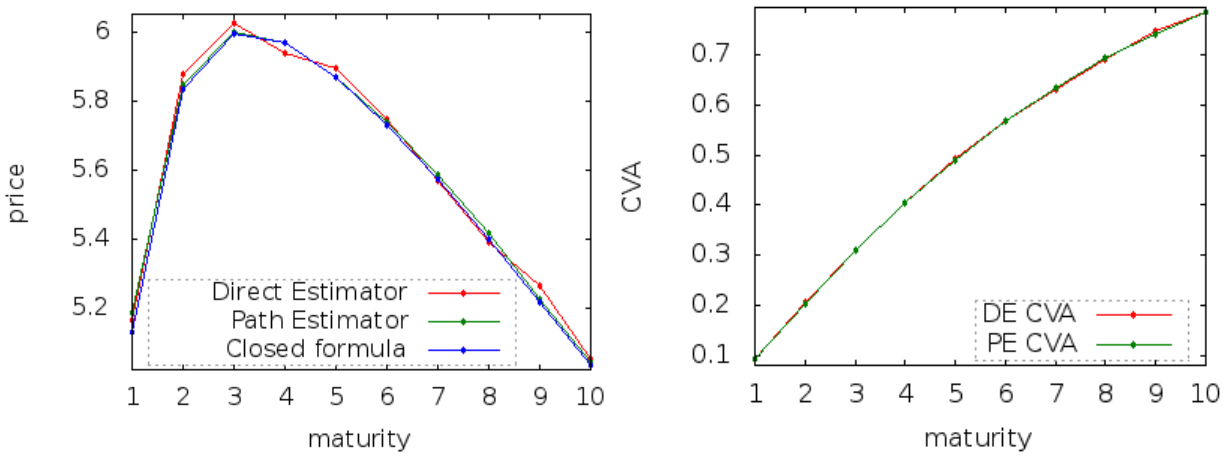
Differences between the use of k-means or recursive bifurcation (RB) for clustering and the simulation of paths by Euler or Alfonsi scheme, when pricing a Bermudan put under the Heston model, are exposed in Table 4.3. We take as reference price for these results the MC Longstaff Schwartz with Alfonsi scheme simulation Premia price, 5.3120. Therefore, we highlight that k-means clustering is not so accurate for options under the Heston model, which could explain why it is not used in [4].

In Figure 4.5 (a) we can see the evolution of the direct and path price estimators with an increasing maturity for an European option under the Heston model. The price from the existing closed formula computed in Premia is also compared in this plot. Figure 4.5 (b)

	DE k-means	DE RB	PE k-means	PE RB
Euler	5.40561	5.34642	5.38082 (8.11 s)	5.35056 (2.94 s)
Alfonsi	5.42828	5.36977	5.40773 (8.12 s)	5.37607 (3.10 s)

Table 4.3: Direct and path estimators with k-means and RB as clustering method and Euler and Alfonsi schemes to simulate paths. The total computation time in each case is also included.

displays the corresponding progression of the CVA of these estimators. Parameter set in Table 4.2 is employed. Neither the number of clusters nor the number of exercise dates affect to these outcomes. In addition, remark that, even if the direct and path estimators are sometimes "quite" separated, both produce almost the same CVA.



(a) (b)

Figure 4.5: Evolution of (a) price and (b) CVA, for an European put on a single asset under the Heston model with increasingly maturity until ten years.

5 A Forward Solution for Computing Derivatives Exposure

The quantification of the exposure to a counterparty's default represents the main function of the valuation and management of counterparty risk. This exposure is expressed as the expectation of entering cash flows at a chosen set of observation dates ending at the final date of the contract. The exposure expression is provided under a risk probability measure, which, for B. Lapeyre and M. Iben Taarit (2015) [8], is the risk neutral probability.

Considering the expected exposure as the price of a compound option[1], they give a forward representation of a derivative's expected exposure using Dupire-like arguments in [8]. They show then that this approach reduces considerably the computation efforts with several numerical examples.

After study the proposed forward representation to understand its workings, my task here has included to rediscover their computation outcomes translating their method to C code from the R code proportioned by Marouan Iben Taarit. The purpose is again to add the C code to Premia v.18. We can not come into more specifications about their approach because the paper is still not published since they are working on last details.

The immediate theoretical interests and practical perspectives that they highlight for the forward representation in [8] are : First, it gives a payoff-free analytical representation that can be easily used to compute the expected exposure, as a function of the observation date, the contract's Delta, and the underlying volatility. An advantage of the forward representation is that the method can be also used for a fast approximation of the expected exposure Greeks. Second, the forward solution is incremental. Therefore, the computation effort benefit from being concentrated on the exposure increments between each couple of consecutive observation dates. Thus, one can freely set the own observation dates, correspondingly to a hedging strategy or default believes, without impacting the accuracy of the exposure calculation. More precisely, the forward representation can be written as an incremental scheme that gives the entire expected exposure trajectory, incrementally, over several observation dates.

[1] An European Call option on the residual contract with a 0-strike price and maturing at the default observation date.

5.1 Numerical experiments

This section shows the results found with C code, meant to be the same as in [8]. The performance of both codes is also compared and some numerical aspects related to the implementation of the forward representation are discussed. We just address the case of an Equity Forward under the Black-Scholes model, because I still work on this task during these last weeks.

5.1.1 Equity Forward under the Black-Scholes model

Considering an Equity forward contract for which the expected exposure is given explicitly. We refer by S to the price of an Equity asset and by X its logarithm. We assume that X has the following dynamics

$$\begin{cases} dX_s &= (r - q - \frac{1}{2}\sigma^2)ds + \sigma dW_s, \ s \geq t \\ X_t &= x \end{cases} \tag{5.1}$$

where r is the risk-free rate, q is the continuous dividend rate and σ the instantaneous volatility. The price V of the forward contract is given by

$$V(t,x;t) = \mathbb{E}_t \left[D(t,T) \left(e^{X_T^{t,x}} - K \right) \right] = e^{-q(T-t)} e^x - e^{-r(T-t)} K. \tag{5.2}$$

At an intermediate maturity s, the expected exposure for this contract is given by the Black-Scholes Call formula :

$$\begin{aligned} EE(t,x;s) &= \mathbb{E}_t \left[e^{-r(s-t)} \left(e^{-q(T-s)} e^{X_T^{s,\, X_s^{t,x}}} - e^{-r(T-s)} K \right)^+ \right] \\ &= e^{-q(T-s)} Call_{BS}(t, e^x; s, K_s), \end{aligned} \tag{5.3}$$

where $K_s = e^{-(r-q)(T-s)} K$, and

$$\begin{cases} Call_{BS}(t, e^x; s, K_s) &= e^{-q(T-t)} e^x \mathcal{N}(d_1) - K_s e^{-r(T-t)} \mathcal{N}(d_1 - \sigma\sqrt{T-t}) \\ d_1 &= \dfrac{ln(\frac{e^x e^{-q(T-t)}}{K e^{-r(T-t)}})}{\sigma\sqrt{T-t}} + \dfrac{1}{2}\sigma\sqrt{T-t} \end{cases} \tag{5.4}$$

The expected exposure is computed at different observation dates s_i, with the number of payment dates equal to 1. The simulation parameters are taken as follow in Table 5.1.

S	K	T	s_i	r	q	σ
1	ATM	5 Y	T $\times \frac{i}{N}$	1%	5%	50%

Table 5.1: Parameter values used for an Equity Forward.

Figure 5.1 shows the results presented in [8] of the forward solution versus the exact one obtained through the closed formula. Figure 5.2 displays the reproduction of these results that

we find with the code in C. The execution times of both solutions with both programming languages are introduced in Table 5.2. As we previously mention, their approach reduces considerably the computation efforts. On the other hand, even if all results are obtained in less than a second, we can also appreciate how fast is C code and imagine how useful it can be for this kind of algorithms where performance is so important and represents money.

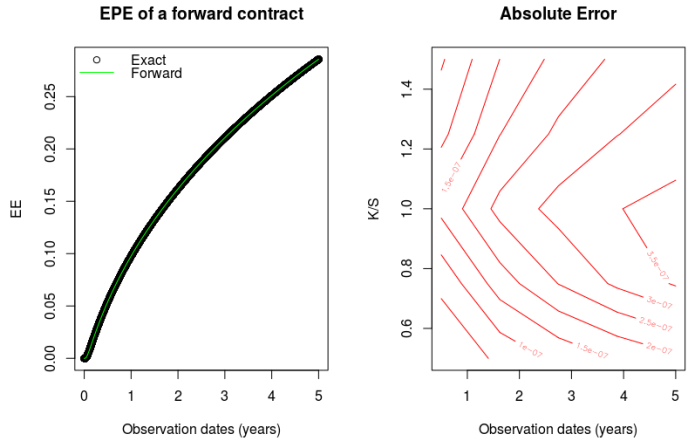


Figure 5.1: Expected Exposure of an Equity Forward: Expected Exposure Profile (left), Absolute Error (right).

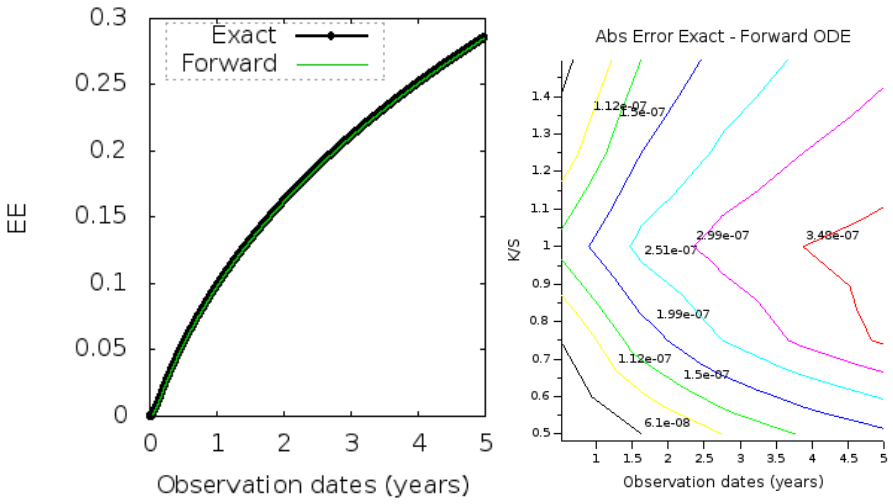


Figure 5.2: Expected Exposure of an Equity Forward: Expected Exposure Profile (left), and contour lines of Absolute Error (right).

	R	C
Exact solution	426 ms	170 ms
Forward solution	135 ms	100 ms

Table 5.2: Execution times of the exact and forward solutions with R and C code, for a number of observation dates, $N = 10^6$.

6 Conclusions

The main subject of my internship has been the Stochastic Grid Bundling Method (SGBM), introduced in [7] for approximating the value of Bermudan options. We summarize here some ideas that we have extracted from the numerical experiments made. The first one, and the key concept of the SGBM, is that we are interested in clustering paths. This takes us computation time, but we regain part by doing regression of smaller size into each cluster, and we gain accuracy as shown in figures of Chapter 3. So the algorithm that we have coded is computationally competitive with respect to other methods existing in this domain. Secondly, remark that, even if the thought that increasing the polynomial order of the approximation leads to a better approach is logical, the truth is that, numerically, an order greater than four comes of a distancing from the right price. That is the case for Monte Carlo Longstaff Schwartz method. On the other hand, with an appropriate number of clusters fixed, the SGBM is not affected equally for this parameter evolution. Finally, as it was not unexpected, with an increasing number of Monte Carlo simulations we obtain more accurate pricing results. However, as the random generator takes part into the outcomes, occasionally we can not find a perfect convergence when testing this aspect.

Taking advantage of the SGBM algorithmic structure lets us apply it for the computation of exposure profiles [4] and makes this method doubly interesting because we are able to assess the currently precious CVA as well. In this case, we enrich our range of financial asset dynamics studied, employing the Black-Scholes model, constant volatility, and the Heston model, stochastic volatility, in our numerical examples.

The study of SGBM has carried me to face different types of bundling algorithms, a key idea in the learning machine field. Another example that shows how extended and important this domain has become these days. In addition, to improve the computational efficiency of the SGBM, impacted by the number of clusters used, a parallelization of the algorithm will be interesting.

Apart from the conclusions concerning the themes developed during the internship, there are also some other reflections which are worth doing. This internship has allowed to introduce myself more in the research world in a unique frame, representing an opportunity to apply the knowledge I have acquired in my academic training and to delve into mathematical finance. I could work on three papers due to a continuous labour and thanks to an excellent guide, facing difficulties as well, but concluding in a rewarding experience. This internship represents the closure of a period of my life and I am ready and excited about what is coming next.

Bibliography

[1] Damiano Brigo and Aurélien Alfonsi. Credit default swaps calibration and option pricing with the ssrd stochastic intensity and interest-rate model. *Finance and Stochastics*, 9, 2005.

[2] John C. Cox, Jonathan E. Ingersoll, and Stephen A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407, 1985.

[3] Fang Fang and Cornelis W. Oosterlee. Pricing early-exercise and discrete barrier options by fourier-cosine series expansions. *Numerische Mathematik*, 114(1):27–62, 2009.

[4] Qian Feng and Cornelis W. Oosterlee. Monte carlo calculation of exposure profiles and greeks for bermudan and barrier options under the heston hull-white model. *Available at SSRN 2494233*, 2014.

[5] Jon Gregory. *Counterparty Credit Risk: The new challenge for global financial markets*, volume 470. John Wiley & Sons, 2010.

[6] Steven L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of financial studies*, 6(2):327–343, 1993.

[7] Shashi Jain and Cornelis W. Oosterlee. The stochastic grid bundling method: Efficient pricing of bermudan options and their greeks. *Available at SSRN 2293942*, 2013.

[8] Bernard Lapeyre and Marouan Iben Taarit. A forward solution for computing derivatives exposure. 2015.

[9] Stuart P. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[10] Francis A. Longstaff and Eduardo S. Schwartz. Valuing american options by simulation: A simple least-squares approach. *Review of Financial studies*, 14(1):113–147, 2001.

[11] Jean-Jacques Ruch. Vecteurs gaussiens. *Préparation à l'Agrégation Bordeaux 1*, 2012 - 2013.

A Code to generate paths

This appendix introduces the code in C of the scheme to generate grid points for single and multiple assets under Black and Scholes model. We can see in Appendix B some mathematical details about correlation when multiple assets are simulated.

Listing A.1: Generation of grid points algorithm

```
/**
 * @brief Generate the stochastic paths.
 *
 * @param [out] SPaths Simulated stochastic paths.
 * @param [in] generator Random generator.
 */
static void getTrajectories(PnlMat** SPaths, PnlRng* generator)
{
    assert(SPaths != NULL);
    assert(generator != NULL);

    int i, j;
    double dt;

    if (InstrumentType == SINGLE)
    {
        double auxValue;
        PnlMat* SPathsAux = pnl_mat_new();
        PnlVect* initialV = pnl_vect_create_from_scalar
                            (
                                NRepl,
                                log(S0[0])
                            );
        pnl_mat_rng_normal(SPathsAux, NRepl, numDates, generator);
        pnl_mat_set_col(SPathsAux, initialV, 0);

        for (j = 1; j < numDates; j++)
        {
            dt = Dates[j] - Dates[j-1];
            for (i = 0; i < NRepl; i++)
            {
```

```
                auxValue = (r - CDividendR[dim-1] -
                            0.5*SQR(sigma[0]))*dt +
                           sigma[0]*sqrt(dt)*MGET(SPathsAux,i,j);
                MLET(SPathsAux, i, j) = auxValue;
            }
        }
        pnl_mat_cumsum(SPathsAux, 'c');

        for (j = 0; j < numDates; j++)
        {
            pnl_mat_resize(SPaths[j], NRepl, dim);
            for (i = 0; i < NRepl; i++)
                MLET(SPaths[j], i, dim-1) =
                  exp(MGET(SPathsAux, i, j));
        }

        pnl_vect_free(&initialV);
        pnl_mat_free(&SPathsAux);
    }
    else
    {
        int k, l;
        double depBrow;
        double SigmaMatValue = 0.;
        PnlMat* SPathsAux = pnl_mat_create(dim, numDates);
        PnlMat* NormalMat = NULL;

        for (j = 0; j < NRepl; j++)
        {
            for (l = 0; l < dim; l++)
                MLET(SPathsAux, l, 0) = log(S0[l]);
            NormalMat = pnl_mat_new();
            pnl_mat_rng_normal(NormalMat, dim, numDates,
                               generator);
            for (i = 1; i < numDates; i++)
            {
                dt = Dates[i] - Dates[i-1];
                for (k = 0; k < dim; k++)
                {
                    depBrow = 0.0;
                    for (l = 0; l < dim; l++)
                        depBrow += MGET(SigmaMat, k, l)*
                                   MGET(NormalMat, l, i);
                    depBrow = depBrow / sigma[k];
                    MLET(SPathsAux,k,i) =
                      (r - CDividendR[k] - 0.5*SQR(sigma[k]))*dt +
```

```
                        sigma[k]*sqrt(dt)*depBrow;
                }
            }
            pnl_mat_cumsum(SPathsAux, 'c');

            for (i = 0; i < numDates; i++)
            {
                if (j == 0)
                    pnl_mat_resize(SPaths[i], NRepl, dim);
                for (k = 0; k < dim; k++)
                    MLET(SPaths[i], j, k) =
                      exp(MGET(SPathsAux, k, i));
            }
            pnl_mat_free(&NormalMat);
        }
        pnl_mat_free(&SPathsAux);
    }

    return;
}

/**
 * @brief Compose the ensemble of exercise and discretization
 *        dates without repetitions.
 */
static void setDates()
{
    setDiscretDates(Nsteps);
    if (NumExerciseDates != 0)
    {
        setExerciseDates(NumExerciseDates);
        numDates = Nsteps + NumExerciseDates;
    }
    else
        numDates = Nsteps + 1;

    Dates = (double*) malloc(numDates*sizeof(double));
    if (Dates == NULL)
        abort();
    memcpy(Dates, DiscretDates, (Nsteps+1)*sizeof(double));
    if (NumExerciseDates != 0)
    {
        memcpy(&Dates[Nsteps+1], &ExerciseDates[1],
                (NumExerciseDates-1)*sizeof(double));

        // Quick-Sort
```

```
        qsort(Dates, numDates, sizeof(double), comparer);
        // remove repetitions
        int i;
        int count = 1;
        for (i = 1; i < numDates; i++)
        {
            if (Dates[i] != Dates[i-1])
            {
                Dates[count] = Dates[i];
                count++;
            }
        }
        numDates = count;
    }

    return;
}

/**
 * @brief Create the correlation matrix.
 */
static void setRhoMat()
{
    rhoMat = pnl_mat_create_from_scalar(dim, dim, rho);
    pnl_mat_set_diag(rhoMat, 1., 0);
}

/**
 * @brief Create Sigma matrix by doing Cholesky decomposition.
 */
static void setSigmaMat()
{
    int i, k;
    SigmaMat = pnl_mat_create(dim, dim);

    for (i = 0; i < dim; i++)
        for (k = 0; k < dim; k++)
            MLET(SigmaMat, i, k) = sigma[i]*sigma[k]*
                                   MGET(rhoMat, i, k);

    if (pnl_mat_chol(SigmaMat) == FAIL)
        abort();
}
```

B Correlation between Brownian motions

In this appendix we introduce some mathematical support related with correlation for simulating paths when pricing options of several assets. I have studied the corresponding basis theory presented at the beginning of this appendix in different courses of probability during my academic training as well as discussing with my director Bernard Lapeyre about them during my internship. Precise theorems, propositions, proofs, and definitions exposed here are mostly taken from [11] and I was confronted to have a good command of these results and proofs. Apart from being useful knowledge, understanding of these concepts has been a fundamental part to justify which values are acceptable for correlation and for this reason I include them in this appendix. It also shows again that it was not at all an internship merely in computer science or just programming.

Definition B.1. *A symmetric real matrix M is said to be positive semi-definite if and only if it verifies one of the two following equivalent properties :*

1. *For all column vector V we have $V^t MV \geq 0$*
2. *All its eigenvalues are positive or nil i.e. $Sp(M) \subset [0, +\infty[$*

The spectral theorem implies that if M is symmetric positive semi-definite, then there exists an orthogonal matrix P and a diagonal matrix $D = Diag(\lambda_1, \ \ldots, \ \lambda_d)$ (positive) such that $M = PDP^t$. From that we can deduce that there exists a matrix A (not unique in general), such that $M = AA^t$; for example $A = PDiag(\sqrt{\lambda_1}, \ \ldots, \ \sqrt{\lambda_d})$. A is called a squared root of M.

Theorem B.1. *The covariance matrix of a vector $X = (X_1, \ \ldots, \ X_d)$ is a symmetric positive semi-definite matrix.*

Proof. By construction the covariance matrix is symmetric. It is written as the product of a vector and its transpose (the mean is applied then to each component and therefore symmetry is not changed). For the second point we only need to notice that if M is the covariance matrix of the random vector X and if V is a constant vector of $\mathbb{R}^d$, taking into account the bilinearity of the covariance, then

$$V^t MV = \sum_{i,j=1}^{d} v_i v_j Cov(X_i, X_j) = Cov(Z, Z) = Var(Z) \geq 0, Z = v_1 X_1 + v_2 X_2 + \ \ldots + v_d X_d.$$

□

Theorem B.2. *If X is a random (column) vector in $\mathbb{R}^p$ with mean vector m and covariance matrix Γ. Then, if A is a real matrix $q \times p$, the random vector AX in $\mathbb{R}^q$ has Am as mean vector and $A\Gamma A^t$ as covariance matrix.*

Proof. It is a simple consequence of the expectation linearity. For the mean we have :

$$\mathbb{E}[AX] = A\mathbb{E}[X] = Am$$

and for the covariance matrix :

$$\mathbb{E}[(AX - Am)(AX - Am)^t] = \mathbb{E}[A(X-m)(X-m)^t A^t] = A\mathbb{E}[(X-m)(X-m)^t]A^t = A\Gamma A^t.$$

□

Theorem B.3. *Every symmetric positive semi-definite matrix Γ of $d \times d$ dimension is the covariance matrix of a random vector of $\mathbb{R}^d$.*

Proof. Let A be a matrix square root of Γ. We denote X a random vector of $\mathbb{R}^d$ of which components are independent, with mean 0 and variance 1. Therefore the expectation of X is the zero vector, and the covariance matrix is equal to Id_d. The random vector AX is then centred of covariance matrix $AI_dA^t = AA^t = \Gamma$. □

Definition B.2. *A random vector of $\mathbb{R}^d$ is a Gaussian random vector if and only if every linear combination of its components is a Gaussian real-valued random variable, i.e. :*

$$\forall a \in \mathbb{R}^d, a^t X \overset{\mathcal{L}}{\sim} \mathcal{N}(m, \sigma^2)$$

Theorem B.4. *Let X be a random vector of $\mathbb{R}^d$ with mean m and covariance matrix Γ. The following assertions are equivalent :*

- *The vector X is a Gaussian vector.*
- *The distribution of vector X is the same as the distribution of vector $m + AZ$, where Z is a random vector of $\mathbb{R}^d$ with independent components of law $\mathcal{N}(0,1)$ and A is a matrix square root of Γ.*

Proof. The equivalence comes from theorem B.2 and the properties of the symmetric positive semi-definite matrices. □

The distribution of a Gaussian vector is characterised by its mean vector m and its covariance matrix Γ. It is called Gaussian distribution in $\mathbb{R}^d$ and it is denoted by $\mathcal{N}(m, \Gamma)$. The distribution $\mathcal{N}(0, I_d)$ is called standard Gaussian distribution in $\mathbb{R}^d$. A Gaussian vector following this distribution is called standard Gaussian random vector.

Proposition B.1. *Every symmetric positive semi-definite matrix Γ of dimension $d \times d$ is the covariance matrix of a Gaussian random vector of $\mathbb{R}^d$.*

Proof. Let Z be the $\mathbb{R}^d$ Gaussian vector of which every component are independents and follow distribution $\mathcal{N}(0,1)$. It is a standard Gaussian random vector. From the theorem B.2 and the theorem B.4, AZ is a $\mathbb{R}^d$ Gaussian vector of distribution $\mathcal{N}(0, \Gamma)$. □

In our case, the asset prices are assumed to follow correlated geometric Brownian motion processes, i.e.

$$\frac{dS_t^i}{S_t^i} = (r - q_i)dt + \sigma_i dW_t^i$$

where each asset pays a dividend at a continuous rate of q_i that we take equal to 0 here for simplicity. W_t^i, $i = 1, \ldots, d$, are standard Brownian motions and the instantaneous correlation coefficient between W_t^i and W_t^j is ρ_{ij}, i.e. $d\langle W_t^i, W_t^j\rangle = \rho_{ij}dt$, i.e. the process $(W_t^i W_t^j - \rho t, t \geq 0)$ is a martingale and $\mathbb{E}[W_t^i W_t^j] = \rho_{ij} \times t$. We define :

$$\rho = (\rho_{ij}),\ 1 \leq i \leq d,\ 1 \leq j \leq d = \begin{pmatrix} 1 & \rho_0 & \cdots & \cdots & \rho_0 \\ \rho_0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \rho_0 \\ \rho_0 & \cdots & \cdots & \rho_0 & 1 \end{pmatrix},\ \rho_0 \in [-1, 1], \tag{B.1}$$

which must be a symmetric positive semi-definite matrix, $\rho_{ij}dt = d\langle W^i, W^j\rangle_t$. We have then :

$$\begin{aligned} \frac{dS_t^i}{S_t^i} &= rdt + (\Sigma\ d\bar{W}_t)_i \\ &= rdt + \sigma_i dW_t^i \end{aligned} \tag{B.2}$$

where $\bar{W}_t$ is an independent Brownian motion of dimension $d \times 1$, and Σ is a matrix $d \times d$ that we obtain as follows. Equalising terms in (B.2) we arrive to :

$$dW_t^i = \frac{\sum_{j=1}^d \Sigma_{ij} d\bar{W}_t^j}{\underbrace{\sqrt{\sum_{j=1}^d \Sigma_{ij}^2}}_{(=\sigma_i)}}$$

Then, according to what we have exposed before,

$$d < W_t^i, W_t^k >= \rho_{ik}dt = (\frac{1}{\sigma_i \sigma_k} \sum_{j=1}^d \Sigma_{ij}\Sigma_{kj})dt$$

which can be expressed as :

$$\Sigma\Sigma^* = (\sigma_i \sigma_k \rho_{ik}),\ 1 \leq i \leq d,\ 1 \leq k \leq d. \tag{B.3}$$

Finally, doing Cholesky factorization in right hand side of (B.3) we find Σ, lower triangular matrix. In order to apply Cholesky factorization we need a symmetric positive definite matrix. $\sigma_i \geq 0$, for $1 \leq i \leq d$ and $\rho_0 \in [-1, 1]$. Therefore, the only inconsistency with problem setting can be caused by certain values of ρ_0 with which ρ would not be a correlation matrix and Cholesky factorization could not be done.

We can express :

$$\rho = (1 - \rho_0)Id + \rho_0 M, \tag{B.4}$$

where

$$M = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

Doing spectral analysis of M, $M\vec{x} = \vec{0} \Leftrightarrow x_1 + \ \ldots \ + x_d = 0$, the kernel of M has dimension $d - 1$, and the eigenvalues of M are $\lambda_1 = 0$ and $\lambda_2 = tr(M) = d$, where $tr(M)$ is the trace of M[1]. So, to be a positive semi-definite matrix, conditions upon eigenvalues of ρ are :

$$(1 - \rho_0) \geq 0 \longrightarrow \rho_0 \leq 1$$
$$(1 - \rho_0) + \rho_0 d \geq 0 \longrightarrow 1 + \rho_0(d - 1) \geq 0 \longrightarrow \rho_0 \geq \frac{-1}{d - 1}$$

Therefore, we obtain for the correlation value that we have to impose not only $\rho_0 \in [-1, 1]$, but $\rho_0 \geq \frac{-1}{d-1}$ in order to have a problem setting which makes sense.

[1]The trace of a matrix is the sum of the (complex) eigenvalues, and it is invariant with respect to a change of basis.

C Code and documentation

Apart from the code showed in Appendix A, we do not introduce the code made during this internship in the present report. The reasons are both its volume and the fact that it belongs to the new version of Premia, which will not become freely available on its web site for up to two years. However, in this Appendix we bring up some relevant aspects of the work made in the code subject.

In total, approximately more than 1,000 lines in Scilab and about 3,000 - 4,000 lines of C code have been programmed to accomplish the main goals of the internship. At the beginning, I was proposed to start programming in Scilab[1] in order to familiarise myself with clustering methods involved in SGBM and SGBM in general. Because Scilab is an interpreted language[2], understanding of the operation and error detection were easy. After this phase, we proceeded with C implementation, which was the real aim to add the programs in Premia. As it is well known, C is a compiled language, so we could execute the tests faster, even increasing the number of Monte Carlo simulations performed. For our C code, we have used a numerical library for C and C++ programmers, PNL[3]. PNL is a free software with a wide range of routines available on topics as cumulative distribution functions, fast Fourier transform, Laplace inversion, least-squares fitting, linear algebra, MPI bindings, multidimensional root finding, multivariate polynomial regression, numerical integration, optimization with inequality constraints, complex numbers, and random number generators. I had previously worked with PNL during the development of my Final Degree Project.

In addition, C implementation has been exhaustively documented in English by specifying description, input and output parameters, and return values of each function, as well as inside function comments. It will be useful in case of someone needs to continue working on these methods or for academic purposes. To produce this code documentation, Doxygen[4], the de facto standard tool for generating documentation from annotated C++ sources[5], has

[1] Scilab is free and open source software for numerical computation providing a powerful computing environment for engineering and scientific applications. Scilab includes hundreds of mathematical functions. It has a high level programming language allowing access to advanced data structures, 2-D and 3-D graphical functions. `http://www.scilab.org/`

[2] An interpreted language is a programming language for which most of its implementations execute instructions directly, without previously compiling a program into machine-language instructions. Another examples of interpreted languages are Python and MATLAB.

[3] `http://pnl.gforge.inria.fr/dokuwiki/doku.php`

[4] `http://www.stack.nl/~dimitri/doxygen/`

[5] It also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D.

been used. We have generated with Doxygen an on-line documentation browser in HTML from our set of documented source files. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code. Figure C.1 shows an example of the HTML documentation created for two of our functions.

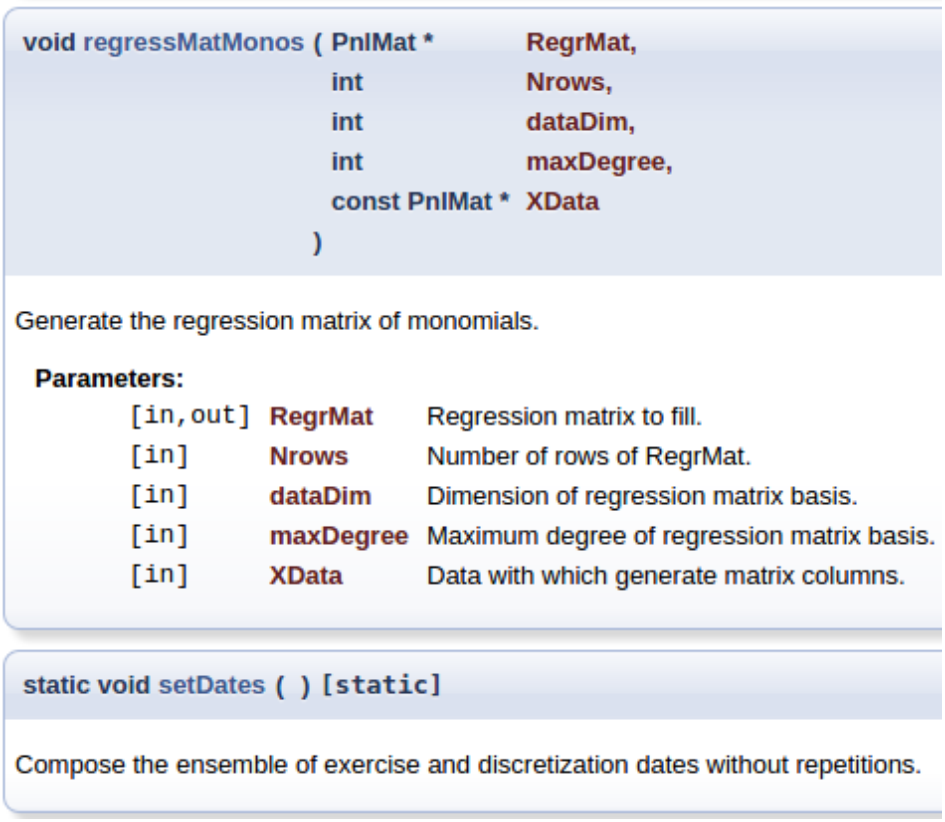


Figure C.1: Functions' HTML documentation generated with Doxygen.

D Pseudocodes of SGBM

We exhibit in this appendix the pseudocodes of the three main functions involved in SGBM in order to summarise in few lines how it works. The pseudocodes should help to clarify the cycle of SGBM described in Section 3.1, steps II to V.

Listing D.1: calibration - Compute calibration of a model.

```
/**
 * Input Parameters:
 *   Simulated stochastic paths.
 * Returns:
 *   Coefficients found in each time step for each cluster.
 *   Centres, if k-means clustering algorithm, for each time step.
 */

Initialize exercise time at expiration
Compute intrinsic value of the option at terminal time t_M = T

for date = t_{M-1} to date t_1

    Update the intrinsic value, multiply it by the discount of
    one time step

    Cluster the paths at date
    for every cluster
        Build regression matrix with paths data of this cluster

        Solve the linear system Ax = b in the least square
        sense, i.e. x = arg min_U || A * u - b ||^2, where
        A is the regression matrix built and b the intrinsic value

        Set continuation value to the result of A * x

    if exercise option type = American option
        Compute option value at date

        if option value > continuation value
           Set intrinsic value to option value
           Update exercise time to date, the optimal stopping time
```

Listing D.2: getDirectEstimator - Compute the direct estimator.

```
/**
 * Input Parameters:
 *   Simulated stochastic paths.
 *   Coefficients found in each time step for each cluster.
 *   Centres, if k-means clustering algorithm, for each time step.
 */

Cluster the paths at date t_1

for every cluster
    Build regression matrix with paths data of this cluster

    Set continuation value to the result of A * x, where A is the
    regression matrix built and x the coefficients from
    calibration phase

Update continuation value, multiply it by the discount of one
time step

Compute option value at initial date, t_0

Set direct estimator to max between mean of continuation value
and option value at t_0
```

Listing D.3: getPathEstimator - Compute the path estimator.

```
/**
 * Input Parameters:
 *   Simulated stochastic paths.
 *   Coefficients found in each time step for each cluster.
 *   Centres, if k-means clustering algorithm, for each time step.
 */

Initialize exercise time at expiration
Compute intrinsic value of the option at terminal time t_M = T

Compute exposures for CVA at T

for date = t_{M-1} to date t_1

    Update the intrinsic value, multiply it by the discount of
    one time step

    Cluster the paths at date
    for every cluster
```

```
        Build regression matrix with paths data of this cluster

        Set continuation value to the result of A * x, where A is
        the regression matrix built and x the coefficients from
        calibration phase

    if exercise option type = American option
        Compute option value at date

        if option value > continuation value
            Set intrinsic value to option value
            Update exercise time to date, the optimal stopping time

    Compute exposures for CVA at date

Update intrinsic value, multiply it by the discount of one
time step

Compute exposures for CVA at t_0

Set path estimator to the mean of the intrinsic value
```