Numerical Methods for High Dimensional BSDEs

by

Majdi Rabia

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science

in

Applied Probability and Statistics

Supervisors

Alexandre Thiery Chao Zhou

Department of Applied Probability and Statistics

National University of Singapore

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.
This thesis has also not been submitted for any degree in any university previously

Majdi Rabia
April, 2017

Acknowledgements

I would like to thank the following people for their unwavering support during the writing of this MSc thesis research.

To my supervisors, Alexandre Thiery (Department of Applied Probability and Statistics, NUS) and Chao Zhou (Department of Mathematics, NUS), for their constant guidance and support in supervising this thesis. I would have never tried some computational methods (Random Forest, Mesh, ...) and understood the practical meaning of financial mathematics without them. Thank you for always being patient with me.

To my parents and siblings, for their patience and absolute faith in me. Even if they do not understand most of these lines, they have always supported me in everything I have done.

To my flatmates and friends during this period, who have helped in the process of writing, giving me honest remarks and useful advices. Special mentions to Hugo Gerard, Olive Morice, Ludovic Martin and Andy Issenjou for their reading and reviewing.

Contents

List of Tables

List of Figures

Abstract

Option Pricing is a well-known subject in the literature since Black and Scholes model first appeared in 1973. However, with the emergence of robust processors, pricing basket options (options on multiple assets) in finance or solving optimisation problems of diversified portfolios became less time-consuming. This is what we aim for in this MSc thesis: explore the currently used computational methods and try new ones for an already settled theory, the High Dimensional Backward Stochastic Differential Equation (HD BSDEs).
This special kind of Stochastic Differential Equation, is useful for problems involving final condition hypotheses. Hence, from an ending point, we work backward to an optimal initial solution.

Chapter 1

Introduction

Throughout this paper we consider $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$ a complete probability space on which a d-dimensional Brownian Motion $B = (B_t)_{0\leq t\leq T}$ is defined, where T is a positive constant and $(\mathcal{F}_t)_t$ is the natural augmented filtration.
We will moreover make use of the following spaces

- $\mathbb{L}^2 = \{X \text{ r.v., such that } \mathbb{E}[X^2] < \infty\}$
- $\mathcal{H}^2 = \left\{Z, s.t\ \mathbb{E}[\int_0^T |Z_s^2|] < \infty\right\}$
- $\mathcal{S}^2 = \left\{Y, s.t\ \mathbb{E}[\sup_{t\leq T} Y_t] < \infty\right\}$

1.1 SDE

In this section, we recall the basic tools from stochastic differential equations

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dB_t \quad t \in [0, T] \tag{1.1}$$

where $T > 0$ is a given maturity date. Here, μ and σ are $\mathcal{F} \times \mathcal{B}(\mathbb{R}^d)$ measurable functions from $[0, T] \times \mathbb{R}^d$ to $\mathbb{R}^d$ and $\mathcal{M}_d(\mathbb{R})$, respectively.

Definition 1.1.1. A strong solution of 1.1 is a $\mathcal{F}$ measurable process X such that $\int_0^T (|\mu(t, X_t)| + |\sigma(t, X_t)|^2)dt < \infty \quad a.s.$ and

$$X_t = X_0 + \int_0^t \mu(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s, \quad t \in [0, T] \tag{1.2}$$

1.2 BSDE

1.2.1 Problem

All over this paper, we will mainly be interested on the following so called backward stochastic differential equation (BSDE).

$$\begin{aligned} dX_t &= \mu(t, X_t)dt + \sigma(t, X_t)dB_t \\ -dY_t &= f(t, X_t, Y_t, Z_t)dt - Z_t dB_t \\ Y_T &= \xi \end{aligned} \tag{1.3}$$

In this representation, $(X_t)_{0 \leq t \leq T}$ is the forward d-dimensional process, $(Y_t)_{0 \leq t \leq T}$ is the backward component, where f is the driver of the BSDE, ξ a measurable function with respect to the filtration generated by the Brownian Motion B, and (Y_t, Z_t) is a pair of square-integrable adapted processes satisfying the equation.

1.2.2 Motivation

BSDEs are quite present in literature, especially since its introduction in the linear case by Bismut in 1973 [3] and Pardoux and Peng [25] in the general case. Different methods have been used over the years, from Least-Squares Monte Carlo [2] , to finite difference method [22], [24], Control Variates [19], Stochastic Mesh [17], [18], Malliavin Calculus [29], Basis function [19], Quantization [16], Cubature [8] or PDE methods([10]).
Given this rich literature and concrete applications in pricing and optimization problems in financial mathematics, we implemented different methods, selected from above, and added Decisional Trees.
In this MSc thesis, we will mainly focus on Least-Square regression, Stochastic Mesh, and Tree regression methods (Random Forest). The basic idea in the latter is to generate multiple independent decision trees, and average the predictions given. This method has not been used in the literature yet from our best knowledge, given maybe the non established exact theory about randomized trees, especially for bias and variance. Most of the analysis has been about optimizing accuracy and time via hyper-parameters to have better performances than literature methods in high dimensions.

Chapter 2

SDE

2.1 Discretization of forward SDE

In this section, we look at a method that approximates the numerical solution of forward SDE 1.1
Methods differ according to their type of convergence. Hence, we define two types of convergence.

Definition 2.1.1. (Strong convergence of order α) A time-discretized approximation X_i^π of a continuous-time process X, is said to be of general strong order of convergence α to X with Δt if there exists a constant $C \in \mathbb{R}^+$, such that

$$\mathbb{E}[|X_i^\pi - X_i|] \leq C\Delta t^\alpha$$

Definition 2.1.2. (Weak convergence of order α) A time-discretized approximation X_i^π of a continuous-time process X, is said to be of general strong order of convergence α to X with Δt if there exists a constant $C \in \mathbb{R}^+$, such that for every $\phi \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R})$ with polynomial growth

$$|\mathbb{E}[\phi(X_i^\pi)] - \mathbb{E}[\phi(X_i)]| \leq C\Delta t^\alpha$$

Euler-Maruyama method Let us divide $(0,T)$ into subintervals $(t_{i-1}, t_i)_{1\leq i\leq m}$, and set $\Delta t_i = t_i - t_{i-1}$, $\Delta B_i = B_{t_i} - B_{t_{i-1}}$, and $\Delta = \max_i \Delta t_i$.
The Euler-Maruyama approximation of X is a continuous stochastic process X^π satisfying the iterative scheme

$$X^\pi_{t_{i+1}} = X^\pi_{t_i} + \mu(t_i, X^\pi_{t_i})\Delta t_{i+1} + \sigma(t_i, X^\pi_{t_i})\Delta B_{t_{i+1}}$$

with $X^\pi_0 = X_0$.

Proof.

Lemma 2.1.1 (Itô's Lemma). *Assume X_t is an Itô drift-diffusion process that satisfies the stochastic differential equation*

$$dX_t = \mu_t\, dt + \sigma_t\, dB_t$$

where B_t is a Standard Brownian Motion. If $f(t,x)$ is a twice-differentiable scalar function, then

$$df(t,X_t) = \frac{\partial f}{\partial t}(t,X_t)dt + \frac{\partial f}{\partial x}(t,X_t)dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(t,X_t)\sigma_t^2 dt.$$

Applying this Lemma to μ and σ, we get the so called Itô-Taylor expansion :

$$\begin{aligned}
\mu(t,X_t) &= \mu(0,X_0) + \int_0^t \frac{\partial \mu}{\partial s}(s,X_s)ds + \int_0^t \frac{\partial \mu}{\partial x}(s,X_s)dX_s + \int_0^t \frac{1}{2}\frac{\partial^2 \mu}{\partial x^2}(s,X_s)\sigma_s^2 ds \\
\sigma(t,X_t) &= \sigma(0,X_0) + \int_0^t \frac{\partial \sigma}{\partial s}(s,X_s)ds + \int_0^t \frac{\partial \sigma}{\partial x}(s,X_s)dX_s + \int_0^t \frac{1}{2}\frac{\partial^2 \sigma}{\partial x^2}(s,X_s)\sigma_s^2 ds
\end{aligned}$$

Replacing in (1.2)

$$\begin{aligned}
X_t = X_{t-\delta t} &+ \int_{t-\delta t}^t (\mu(t-\delta t, X_{t-\delta t}) + \int_{t-\delta t}^t \frac{\partial \mu}{\partial s}(s,X_s)ds + \int_{t-\delta t}^t \frac{\partial \mu}{\partial x}(s,X_s)dX_s + \\
&\int_{t-\delta t}^t \frac{1}{2}\frac{\partial^2 \mu}{\partial x^2}(s,X_s)\sigma_s^2 ds)dt + \int_{t-\delta t}^t (\sigma(t-\delta t, X_{t-\delta t}) + \int_{t-\delta t}^t \frac{\partial \sigma}{\partial s}(s,X_s)ds + \\
&\int_{t-\delta t}^t \frac{\partial \sigma}{\partial x}(s,X_s)dX_s + \int_{t-\delta t}^t \frac{1}{2}\frac{\partial^2 \sigma}{\partial x^2}(s,X_s)\sigma_s^2 ds)dB_t \\
&= X_{t-\delta t} + \mu(t-\delta t, X_{t-\delta t})\int_{t-\delta t}^t dt + \sigma(t-\delta t, X_{t-\delta t})\int_{t-\delta t}^t dB_t + h(\delta t, t, X_{t-\delta t}, X_t) \\
&= X_{t-\delta t} + \mu(t-\delta t, X_{t-\delta t})\delta t + \sigma(t-\delta t, X_{t-\delta t})(B_t - B_{t-\delta t}) + h(\delta t, t, X_{t-\delta t}, X_t)
\end{aligned}$$

where h is a remainder regrouping all the double integrals.
One can show this term converges to 0 when δt goes to 0.

□

Chapter 3

First Order BSDE

3.1 Existence and Uniqueness

We rewrite 1.3 as

$$Y_t = \xi + \int_0^T f(s, Y_s, Z_s)ds - \int_0^T Z_s dB_s \quad t \leq T \quad \mathbb{P} - a.s. \tag{3.1}$$

Theorem 1. Assume that $\{g \to f(t, 0, 0),\ t \in [0, T]\} \in \mathcal{H}^2$ and, for some constant $C > 0$, $|f(t, y, z) - f(t, y_0, z_0)| \leq C(|y - y_0| + |z - z_0|)$ $dt \times dP - a.s$ $\forall t \in [0, T]$ and $(x, y, z), (x_0, y_0, z_0) \in \mathbb{R}^n \times \mathbb{R}^{n \times d}$. Then, $\forall \xi \in \mathbb{L}^2$, there is a unique solution $(Y, Z) \in \mathcal{S}^2 \times \mathcal{H}^2$ to the BSDE (f, ξ)

3.2 First order BSDE and semi-linear PDE

Let us consider the semilinear PDE

$$\partial_t u + \mathcal{L}u + f(t, x, u(t, x), \sigma(t, x)^T D_x u(t, x)) = 0, \ \ (t, x) \in [0, T) \times \mathbb{R}^d \tag{3.2}$$

with the terminal condition $u(T; x) = g(x)$ and $\mathcal{L}$ the Itô generator of X. Such an equation appears when we consider a stochastic control problem with no control on the diffusion coefficient. $(Y_t = u(t; X_t); Z_t = \sigma(t, x)^T D_x u(t; X_t))$ is a solution to the BSDE. To be precise, a straightforward application of Itô's lemma gives the following:

Proposition 3.2.1. *Generalization of Feynman-Kac's formula :*
Let u be a function of $\mathcal{C}^{1,2}$ satisfying (here) and suppose that there exists a constant C such that, for each $(t; x) \in [0; T] \times \mathbb{R}^d$

$$|\sigma(t, x)^T D_x u(t; x)| \leq C(1 + |x|) \tag{3.3}$$

Then $(Y_t = u(t; X_t);\ \ Z_t = \sigma(t,x)^T D_x u(t; X_t))$ *is the unique solution to 1- BSDE [12]*

3.3 Reflected Backward Stochastic Differential Equation (RBSDE)

Some problems require the solution to remain above or under a certain stochastic process. Hence, the previous simple BSDE analysis cannot be implied, and we have to take into account this new constraint. Most famous application for RBSDE is the pricing of American Options (see [4], [27] and [17] for more details about American Option problem formulation and pricing).

N. El Karoui, C. Kapoudjian, E. Pardoux,S. Peng and M. C. Quenez introduced this notion in 1997 with a one-dimensional RBSDE([11]).

RBSDE with lower boundary problem can be formulated as

$$\begin{cases} Y_t = \xi + \int_t^T f(s, Y_s, Z_s)ds - \int_t^T Z_s dB_s + K_T - K_t \\ Y_t \geq L_t \ \text{ constrained value process} \\ \quad \int_0^T (Y_t - L_t)dK_t = 0, \quad 0 \leq t \leq T \end{cases} \tag{3.4}$$

And a solution of such an equation is a triple processes (Y, Z, K) with values in $\mathbb{R} \times \mathbb{R}^d \times \mathbb{R}_+$.

3.4 Discretization

3.4.1 BSDE

We have $Y_T = \xi$, so we focus on a backward simulation.

Considering a simulation of the forward process $(X_t)_t$ using Euler-Maryuma method, and using same notations, we denote X_{t_n} and $Y_{t_n} = \xi(S_{t_n})$ the final state.

The following part explains how we can get a backward algorithm to approximate (Y_t) and (Z_t).

- If we multiply 1.3 by dB_t, we get :

$$-dY_t dB_t = -Z_t dt$$
$$(Y_{t_i} - Y_{t_{i+1}})\Delta B_{t_i} = -Z_{t_i}\Delta t_i$$

Taking the expectation given the information at time t_i, and using Y_{t_i} and $Z_{t_i}\Delta t_i$ being $\mathcal{F}_{t_i}$ - measurable, we get :

$$Z_{t_i} = \frac{1}{\Delta t_i}\mathbb{E}[Y_{t_{i+1}}\Delta B_{t_i}|\mathcal{F}_{t_i}]$$

- Taking conditional expectation given the information at time t_i using 1.3

$$\mathbb{E}[Y_{t_i}|\mathcal{F}_{t_i}] - \mathbb{E}[Y_{t_{i+1}}|\mathcal{F}_{t_i}] = \mathbb{E}[f(t_i, S_{t_i}, Y_{t_i}, Z_{t_i})\Delta t_i|\mathcal{F}_{t_i}] - \mathbb{E}[Z_{t_i}\Delta B_{t_i}|\mathcal{F}_{t_i}]$$

Z_{t_i} being $(\mathcal{F}_{t_i})$ - measurable:

$$\mathbb{E}[Z_{t_i}\Delta B_{t_i}|\mathcal{F}_{t_i}] = Z_{t_i}\mathbb{E}[\Delta B_{t_i}|\mathcal{F}_{t_i}]$$

Finally :

$$\mathbb{E}[\underbrace{Y_{t_i}}_{\mathcal{F}_{t_i} measurable}|\mathcal{F}_{t_i}] = \mathbb{E}[Y_{t_{i+1}}|\mathcal{F}_{t_i}] + \underbrace{\mathbb{E}[f(t_i, S_{t_i}, Y_{t_i}, Z_{t_i})\Delta t_i|\mathcal{F}_{t_i}]}_{=f(t_i,S_{t_i},Y_{t_i},Z_{t_i})\Delta t_i \quad by \mathcal{F}_{t_i} measurability}$$

Given that Y_{t_i} appears on both sides, the previous scheme is implicit, so we can use the following explicit *scheme* to fulfil this step :

$$Y_{t_i} = \mathbb{E}[Y_{t_{i+1}}|\mathcal{F}_{t_i}] + f(t_i, S_{t_i}, Y_{t_{i+1}}, Z_{t_i})\Delta t_i$$

Given Y_{t_n} we can get Y_0 using the previous discretization backwardly.

3.4.2 RBSDE

Let the below boundary be the process $\xi(X_t)$, $\forall t \in [0, T]$. Assuming that, 3.4 becomes

$$\begin{cases} Y_t = \xi - \int_t^T Z_s dB_s + K_T - K_t \\ Y_t \geq \xi(X_t) \\ \quad \int_0^T (Y_t - \xi(X_t))dK_t = 0, \quad 0 \leq t \leq T \end{cases} \tag{3.5}$$

Gobet and Lemor give an approximation procedure in [21]. The reader can refer to this article for more information, as we will follow this procedure in our simulations.

$$\begin{aligned} Z_{t_i} &= \frac{1}{\Delta t_i}\mathbb{E}[Y_{t_{i+1}}\Delta B_{t_i}|\mathcal{F}_{t_i}] \\ Y_{t_i} &= \mathbb{E}[Y_{t_{i+1}}|\mathcal{F}_{t_i}] + f(t_i, Y_{t_{i+1}}, Z_{t_i})\Delta t_i \\ Y_{t_i} &= \max(Y_{t_i}, \xi(t_i, X_{t_i})) \\ Y_{t_m} &= \xi(t_m, X_{t_m}) \end{aligned}$$

Chapter 4

Regression

4.1 Mesh Method

The way to construct a mesh is given by the following figure(4.1), as presented by Glasserman and Broadie in [5] . We simulate N independent forward paths, each path X_i containing m time steps $X_{i,0}$, $X_{i,1}$, ..., $X_{i,m-1}$. Then, we omit the connection between the paths nodes, i.e. we forget which node at time step j generates the one at time $j+1$. For the backward process, we connect all the paths thereafter, giving weights to each connection.

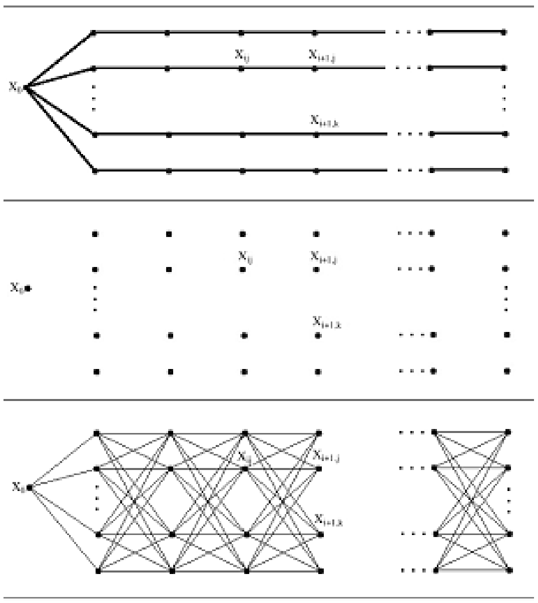


Figure 4.1: Mesh

The weights are related to the probability density function. Intuitively, given a node (i, j), and according to the SDE (1), some $(k, j+1)$ are more likely to be reached by the stock path than others. An important issue of the stochastic mesh method is to determine those weights.

4.1.1 likelihood Ratio Weights

Suppose that we want to evaluate the following $C(t_{i+1}, x) = \mathbb{E}[h(t_{i+1}, X_{i+1})|X_i = x]$ conditional expectation where h is smooth enough to ensure the existence. Let us denote by $(f(t_i, X_i, X_{i+1}))_{i \in 1,\dots,m-1}$ the transition densities of Markov Chain $(X_i)_{i\in 1,\dots,m}$, and $g_(t_i, X_i)$ density function of X_i. Then, by definition,

$$\mathbb{P}[X_{i+1} \in A | X_i = x] = \int_A f_i(x, y)dy$$

which leads to

$$\begin{aligned}
\mathbb{E}[h(t_{i+1}, X_{i+1})|X_i = x] &= \int h(t_{i+1}, y) f(t_i, x, y) dy \\
&= \int h(t_{i+1}, y) \frac{f(t_i, x, y)}{g(t_{i+1}, y)} g(t_{i+1}, y) dy \\
&= \mathbb{E}[h(t_{i+1}, X_{i+1}) \frac{f(t_i, x, X_{i+1})}{g(t_{i+1}, X_{i+1})}] \qquad (4.1)
\end{aligned}$$

Defining now

$$\begin{aligned}
\hat{C}(t_i, X_{i,j}) &= \frac{1}{N} \sum_{k=1}^{N} h(t_{i+1}, X_{i+1}) \omega_{i,j}^k \\
\omega_{i,j}^k &= \frac{f(t_i, X_{i,j}, X_{i+1,k})}{g(t_{i+1}, X_{i+1,j})}
\end{aligned}$$

$\widehat{C}$ is an unbiased estimator of C when $N \to \infty$, with a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{N}})$.
Using definition of g,

$$\begin{aligned}
g(t_{i+1}, X_{i+1} = y) &= \int f(t_i, x, y) g(t_i, x) dx \\
&= \mathbb{E}[f(t_i, X_i, y)] \qquad (4.2)
\end{aligned}$$

Finally, using a second time a Monte-Carlo estimator for the previous density function, we simulate the weights by :

$$\omega_{i,j}^k = \frac{f(t_i, X_{i,j}, X_{i+1,k})}{\frac{1}{N}\sum_{j=1}^N f(t_i, X_{i,j}, X_{i+1,k})}$$

As a reminder, our main problem is calculation of :

$$\begin{cases} \mathbb{E}[Y_{i+1}|X_i] \\ \mathbb{E}[Y_{i+1}\Delta B_{i+1}|X_i] \end{cases}$$

So adapting our previous calculus

$$\begin{cases} \widehat{Y_{i,j}} = \dfrac{1}{N}\displaystyle\sum_{k=1}^N \widehat{Y}_{i+1,k}\omega_{i,j}^k \\ \widehat{Z}_{i,j} = \dfrac{1}{N}\displaystyle\sum_{k=1}^N \widehat{Y}_{i+1,k}\Delta B_{i+1,j}^k \omega_{i,j}^k \end{cases}$$

are highly biased estimators of our two conditional expectations (see [18] for proof)

Remark 4.1.1. The reader can refer to [17] for another method for weights computation, based on optimization of entropy function.

4.1.2 Limits

Mesh Method presents two main drawbacks, especially when used in higher dimension. First, computing the weights requires transition densities, which is not always available (can be computed approximatively though, using the Euler scheme, when SDE is not a geometric Brownian motion). Second, its time complexity of $\mathcal{O}(dmN^2)$, where d is the dimension, m the number of steps used for discretization, and N the number of samples, limits the use of big samples especially. For instance, 10^4 particles would require in Python 10GB of memory in dimension four. Indeed, one float takes 24 bytes in memory, so generating 10^4 particles requires at every step the construction of an $N \times N$ matrix of weights, taking then $24 \times N^2 = 2.4.10^9$ bytes in memory. Hence, working with a 4-dimensional problem would require 10GB memory ...

4.2 Least Square Monte-Carlo

A Least Square Monte-Carlo can be used for the simulation of conditional expectation of the form $\mathbb{E}[Y|X]$, when X and Y are square integrable random

variables, and a set of numerical values (X, Y) is available.
Longstaff and Schwartz made this method popular in financial mathematics for the pricing of American Options, which is build upon a basis projection, i.e upon the statement $\mathbb{E}[Y|X] = h(X)$.
h minimizes what we call a least square function, i.e.

$$h = \arg\min\{\mathbb{E}[|Y - f(X)|^2] \quad ,\text{f s.t} \quad \mathbb{E}[|f(X)|^2] < \infty\} \tag{4.3}$$

This infinite-dimensional problem can be restricted to smaller set of functions to look for. Our simulations for one asset will be using polynomials.
Although very easy to implement in practice, this kind of function basis has a major flaw. For a given number of particles it is not easy to find an optimal degree of the functional basis. This is due to rare events that the polynomials try to fit, leading to some oscillating representation of the function (see below 4.2).

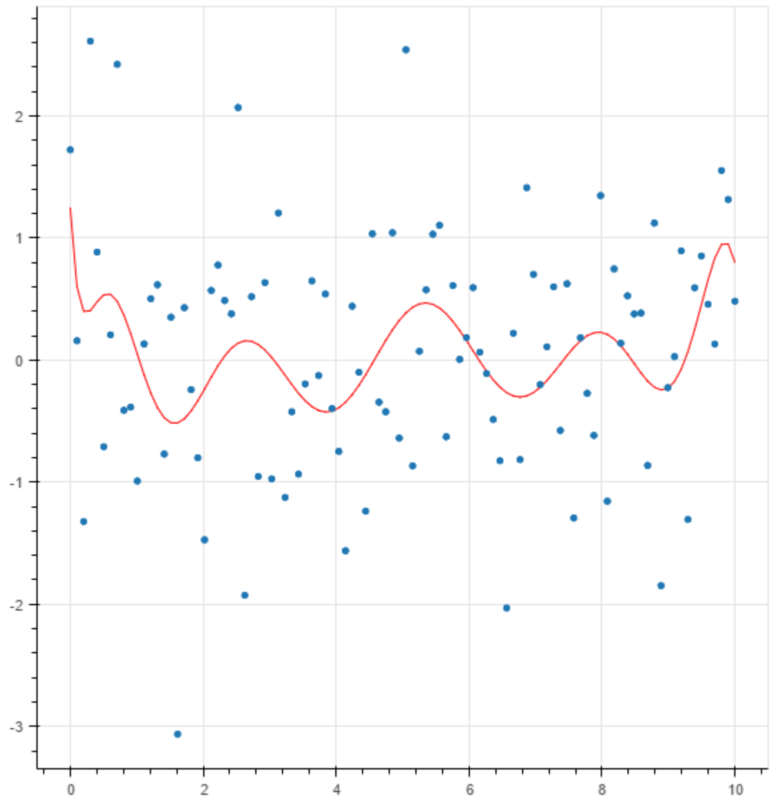


Figure 4.2: Example of polynomial fitting with degree 10 on a random dataset

Remark 4.2.1. We will only use this method for one dimensional example. Indeed, basis projection using polynomials in higher dimension would require too much computational time, and one can show time complexity with number of features d is $\mathcal{O}(d^2)$ ...

4.3 Random Forest Regression

The following section is the main contribution to BSDE analysis, as it uses a machine learning method, Random Forest (Randomized Tree Decision), for the regression step. We will first focus on a quick analysis of decision trees. From there, we will explain Randomized Trees methods using a housing market data in Iowa. We will mainly focus on hyperparameters tuning and complexity of the different Random Trees methods.

4.3.1 Decision Tree

Machine learning research has given plenty of new methods for classification and regression problems. Most effective remain the tree based methods, intuitive and reliable for almost any kind of data.
Following Gilles Louppe ([15])

Definition 4.3.1. A tree is a graph $G = (V, E)$ in which any two vertices (or nodes) are connected by exactly one path.

Definition 4.3.2. A rooted tree is a tree in which one of the nodes has been designated as the root. In our case, we additionally assume that a rooted tree is a directed graph, where all edges are directed away from the root.

Definition 4.3.3. If there exists an edge from t1 to t2 (i.e., if (t1, t2) 2 E) then node t1 is said to be the parent of node t2 while node t2 is said to be a child of node t1.

Definition 4.3.4. In a rooted tree, a node is said to be internal if it has one or more children and terminal if it has no children. Terminal nodes are also known as leaves.

Definition 4.3.5. A binary tree is a rooted tree where all internal nodes have exactly two children.

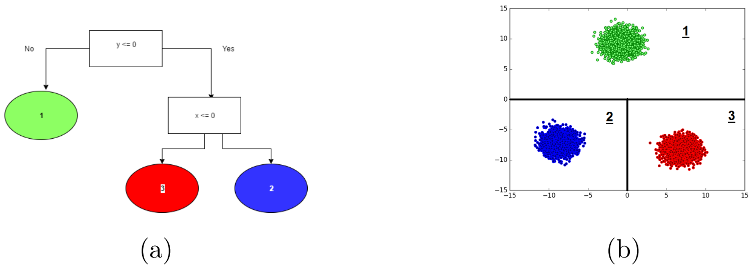


Figure 4.3: A binary tree built for a classification $(1, 2, 3)$ problem from an input space $[-15, 15] \times [-15, 25]$

4.3.2 Randomized Forest

Random Forests make use of the previous Binary trees, generating multiple kind and averaging over them. From Decision trees, we get low bias and high variance, and to overcome this issue, generating random decision trees (i.e random initial state of the tree, random features, ...) and averaging over them makes it possible to get better bias-variance results. The theoretical properties and statistical mechanisms that drive the algorithm are still not clearly and entirely understood. Random forests indeed evolved from empirical successes rather than from a sound theory. As such, various parts of the algorithm remain heuristic rather than theoretically motivated.
We will mainly focus on empirical results but for time complexity and hyper-parameters explanations, the reader can refer to Appendix B.

4.4 Derivative

Another method we will use in the simulations is the derivative one. As explained in 1-BSDE section, $(Y_t = u(t; X_t); Z_t = \sigma(t, x)^T D_x u(t; X_t))$ is solution to (8), with u solution to semi-linear PDE (9). Given our algorithm process

Algorithm 1 BSDE Algorithm

```
1: procedure BSDE
2:     for $t \in \{T-1, \cdots, 0\}$ do
3:         $Z[t] = \frac{1}{\Delta t}\mathbb{E}_t[Y_{t+1}\Delta B_t]$
4:         $Y[t] = \mathbb{E}_t[Y_{t+1} + f(t, X_t, Y_{t+1}, Z_t)\Delta t]$
5:     end for
6: end procedure
```

for all t we can compute once the previous algorithm, and enhance it by taking $Y_t = \phi(t, X_t)$ with ϕ smooth enough to be derivable, and derive from there $Z_t = \nabla\phi(t, X_t)$. Hence

Algorithm 2 BSDE Algorithm

```
1: procedure BSDE
2:     for $t \in \{T-1, \cdots, 0\}$ do
3:         $Z[t] = \frac{1}{\Delta t}\mathbb{E}_t[Y_{t+1}\Delta B_t]$ by RandomForest for instance
4:         $Y[t] = \mathbb{E}_t[Y_{t+1} + f(t, X_t, Y_{t+1}, Z_t)\Delta t]$
5:         for __ = 1:M do
6:             Get $\phi$ such that $Y_t \sim \phi(t, X_t)$
7:             $Y_{new} = \phi(t, X_t)$
8:             $Z_{new} = \nabla\phi(t, X_t)$ gives a new smoother $Z$
9:             $Y[t] = \mathbb{E}_t[Y_{t+1} + f(t, X_t, Y_{t+1}, Z_{new})\Delta t]$
10:        end for
11:    end for
12: end procedure
```

Z being a very noisy process, due to the factor $\frac{\Delta B_t}{\Delta t}$, especially when $\Delta t \to 0$, it seemed relevant to get the smoothest approximation possible. In our simulation, we use a kernel regression to recover a smooth ϕ function. Kernel regression consists on a sum of smooth functions, each one giving an approximation around a point (X_i, Y_i). We assume that points that are close together are similar, a kernel defines then weights that decrease in a smooth fashion as one moves away from the target point.

Denoting $K\left(\frac{X-X_i}{l}\right)$ the kernel function for the point (X_i, Y_i), where l controls the scale and length, we define the weight $\omega(X, X_i) = \frac{K(\frac{X-X_i}{l})}{\sum_{i=1}^{N} K(\frac{X-X_i}{l})}$ such that

$\phi(X) = \sum_{i=1}^{N} \omega(X, X_i) Y_i$.

Remark 4.4.1. $\sum_{i=1}^{N} \omega(X, X_i) = 1$

We use in our simulations a Gaussian Kernel with scaling parameter l, i.e

$$K\left(\frac{X - X_i}{l}\right) = \exp^{-\frac{||X - X_i||^2}{2l^2}}$$

Example 4.4.2. Let Y be a random variable defined by $Y_x = \sin(\frac{x}{10}) + \frac{x}{50} + 0.3\exp^{-x} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.09)$ and $x \in [0, 100]$. Let X be a linear space division of $[0, 100]$ interval with 200 points. Applying a Kernel Regression on (X, Y_X) gives

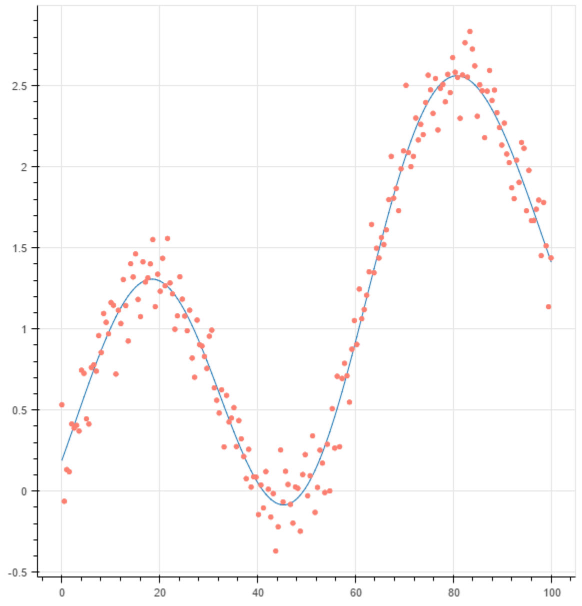


Figure 4.4: Example of Gaussian Kernel Regression

Remark 4.4.3. In one dimension, this algorithm requires the construction of an $N \times N$ matrix of weights. To avoid this $\mathcal{O}(N^2)$ we can consider the '$\mathcal{K}$ Nearest Neighbours' (KNN) instead of all the points when computing the weights. Hence, $\phi(X) = \sum_{i \in \mathcal{S}_{\mathcal{K}}} \omega(X, X_i) Y_i$, where $\mathcal{S}_{\mathcal{K}}$ is a subset of $\{1, \cdots, N\}$ with indices of the KNN.

4.5 Picard Iteration

The noisy Z_t process can be better approximated iteratively using a Picard iteration. This idea has been inspired from E.Gobet [20] and was added as an option in our code. Hence, the backward algorithm process with Picard iteration

Algorithm 3 BSDE Algorithm with Picard iteration

```
 1: procedure BSDE
 2:     for $t \in \{T-1, \cdots, 0\}$ do
 3:         Y_new = 0
 4:         $Z_t = \frac{1}{\Delta t}\mathbb{E}_t[Y_{t+1}\Delta B_t]$ by RandomForest for instance
 5:         $Y_t = \mathbb{E}_t[Y_{t+1} + f(t, X_t, Y_{t+1}, Z_t)\Delta t]$
 6:         for __ = 1:n_picard do
 7:             $Z_t = \frac{1}{\Delta t}\mathbb{E}_t[(Y_{t+1} - Y_t)\Delta B_t]$
 8:             $Y_{new} = \mathbb{E}_t[Y_{t+1} + f(t, X_t, Y_{t+1}, Z_t)\Delta t]$
 9:             $Y_t = Y_{new}$
10:         end for
11:     end for
12: end procedure
```

4.6 Time Complexity

Time complexities presented below in table 4.1 differ with the method used. Hence, the preferred LSM when dimensions are low, as every step is asymptotically $\mathcal{O}(N)$. Then comes mesh and derivative methods, using $N \times N$ matrices, which barres us from using high amount of samples. Finally, as explained in Appendix B, Random Forest time complexity can not be obtained theoretically, but lower bounds and upper bounds exist, depending on number of trees generated and number of features used.

Mesh	$\mathcal{O}(dn_{picard}mN^2)$
Random Forest [1] (worst case - best case)	$\mathcal{O}(n_{trees}Kn_{picard}mN^2\log(N))$ $\mathcal{O}(n_{trees}Kn_{picard}mN\log(N)^2)$
Derivative	$\mathcal{O}(Kn_{picard}mN^2)$
LSM	$\mathcal{O}(d^2n_{picard}mN)$

Table 4.1: Time Complexity with number of samples M, number of paths m, number of picard iteration $n_{pricard}$, dimension d, K the number of selected features, and n_{trees} the number of trees in the Random Forest

Chapter 5

Applications

5.1 Introduction

The following simulations are computed on Python 3.5, with ten CPUs available in the Department of Applied Probability and Statistics in NUS.
Each example relies on a theoretical solution, or a comparison found in BSDE literature.
We will give for every example the solution given by the previous cited methods, with its 95% confidence interval, and the average time. When not specified, we run #30 simulations for every average given, and use a Picard iteration number of three.
We apply BSDE theory on European and American option pricing, respectively on one and multiple assets.
Assets follow a Black-Scholes model

$$\frac{dX_t^i}{X_t^i} = (\mu_t^i - q^i)dt + \sigma_t^i dB_t^i$$

where μ denotes the drift rate, σ the volatility, q the dividend (will be equal to zero in application, unless stated by the author) and X the asset price.
First example analysis will be exhaustive, especially comparison of different methods implemented for regression step. For instance, we will detail how Random Forest parameters have been calibrated for a European call option with different interest rates, and will follow the same idea for the next applications.
Moreover, all the results in table are given with a standard deviation.

5.1.1 Bid-ask model

Suppose the dynamic of the underlying assets to be :

$$\frac{dX_t^i}{X_t^i} = (\mu_t^i - q^i)dt + \sigma_t^i dB_t^i \tag{5.1}$$

$$\left\langle dB_t^i, dB_t^j \right\rangle = \begin{cases} dt \quad \text{if} \quad i = j \\ \rho dt \quad \text{else} \end{cases} \tag{5.2}$$

and let the agent be able to lend (r) and borrow (R) money using two market accounts.

$$\begin{cases} d\alpha_t = R\alpha_t dt \\ d\beta_t = r\beta_t dt \end{cases} \tag{5.3}$$

Let Y_t be the portfolio value at time t and Φ_t^i the amount of shares invested in stock X_t^i at time t. Intuitively, if the difference $Y_t - \sum_{i=1}^p \Phi_t^i$ is positive, the agent can lend to the bank. Inversely, if this difference is negative, the agent would rather borrow money. Hence the variations of dY_t

$$dY_t = \sum_{i=1}^p \Phi_t^i \frac{dX_t^i}{X_t^i} + (Y_t - \sum_{i=1}^p \Phi_t^i)^+ r dt - (Y_t - \sum_{i=1}^p \Phi_t^i)^- R dt$$

We get then the following BSDE

$$-dY_t = f(t, Y_t, Z_t)dt - Z_t dW_t \tag{5.4}$$

where

$$\begin{cases} f(t, Y_t, Z_t) = Z_t.\theta - r.Y + (R - r)(Y - \sum_{i=1}^d (\sigma^{-1} Z_t)_i)^- \\ Z_t = \sigma.\Phi_t \\ \theta = \sigma^{-1}(\mu - r)\mathbb{1} \end{cases} \tag{5.5}$$

5.1.2 Credit Valuation Adjustment (CVA)

Using same notations than Guyon and Labordère [23], CVA can be transformed into a BSDE problem

$$\begin{cases} -dY_t = f(t, Y_t, Z_t)dt - Z_t dW_t \\ f(t, Y_t, Z_t) = \beta(Y_t^+ - Y_t) \end{cases}$$

where $\beta = \lambda_C(1 - R)$ with R the recovery rate (usually equal to 0.4), and λ_C the intensity of the Poisson jump process associated to the CVA problem(see [23]) , with the corresponding semi-linear PDE

$$\begin{aligned} \partial_t u + \mathcal{L}u + \beta(u^+ - u) &= 0 \\ u(T, x) &= g(x) \end{aligned}$$

where $\mathcal{L}$ is the Itô generator of a diffusion X_t.
There is no process Z to regress here, but we will analyse how Random Forest performs on $\mathbb{E}_t[Y^+_{t+\Delta t}]$, comparing the results to Guyon and Labordère([23]).

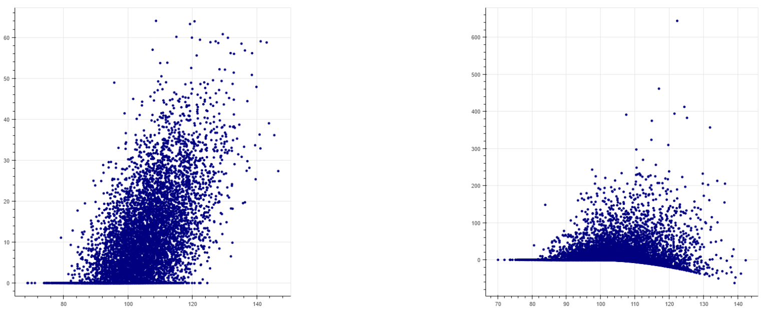
5.2 One dimension

5.2.1 Bid-ask European call option

Referring to Gobet, Lemor and Warin's simulations in [13], we take the following numerical values

X_0	K	σ	r	R	μ
100	100	0.2	0.04	0.06	0.06

We compute the pricing of an European call (i.e pay-off $(X_T - K)^+$) on a single asset associated. This problem has a theoretical solution, given by the Black-Scholes model with R used as free-risk rate. Indeed, to hedge himself, the financial seller will borrow money to buy assets, hence the use of R.



(a) Y_t against $X_{t-\Delta t}$ (b) $Y_t \frac{\Delta B_t}{\Delta t}$ against $X_{t-\Delta t}$

Figure 5.1: Regression at time $t = T - \Delta t$

figure 5.1 shows the first regression step implied by BSDE algorithm. We can notice the noisy process on the right figure due to $\frac{\Delta B_t}{\Delta t}$ term. The bigger the difference between the two rates, the more important the approximation error is on Z_t process.

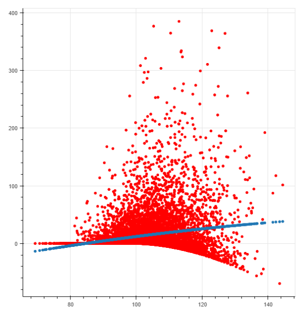
LSM

Main parameter when having polynomial basis projection is the degree. The following table gives the simulation results according to number of samples and this degree parameter.

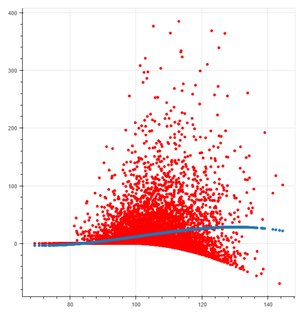
Table 5.1: LSM

degree	N = 100	N = 1000	N = 10000	N = 100000
1	7.166 (1.004)	7.136 (0.24)	7.229 (0.084)	7.242 (0.032)
2	7.156 (0.936)	7.339 (0.326)	7.254 (0.106)	7.272 (0.025)
3	7.296 (0.834)	7.191 (0.319)	7.191 (0.097)	7.17 (0.029)
4	7.194 (0.889)	7.24 (0.354)	7.17 (0.106)	7.177 (0.027)
5	7.222 (0.981)	7.159 (0.248)	7.155 (0.074)	7.168 (0.029)
6	7.225 (0.893)	7.198 (0.298)	7.164 (0.081)	7.155 (0.026)
7	7.188 (1.069)	7.143 (0.319)	7.145 (0.085)	7.168 (0.031)
8	7.368 (0.933)	7.071 (0.258)	7.154 (0.089)	7.164 (0.032)
9	7.45 (1.034)	7.156 (0.302)	7.164 (0.091)	7.163 (0.034)
10	6.984 (0.986)	7.227 (0.348)	7.18 (0.086)	7.16 (0.025)
11	7.188 (1.093)	7.157 (0.266)	7.195 (0.104)	7.169 (0.029)
12	7.187 (0.883)	7.141 (0.19)	7.166 (0.085)	7.164 (0.026)

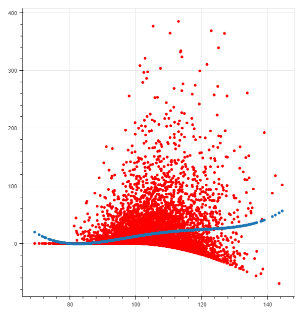
Paradoxically, accuracy does not increase with polynomial degree. Even if approximation is increased for central data points, peripheral points can undergo high variance issues (cf 5.2 below).
Theoretical price of 7.15 is within reach with 100.000 samples and degree 6.



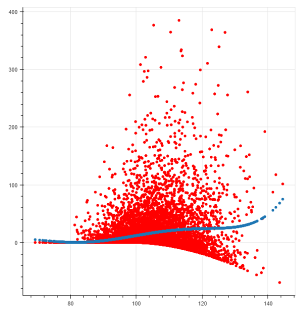
(a) d =2



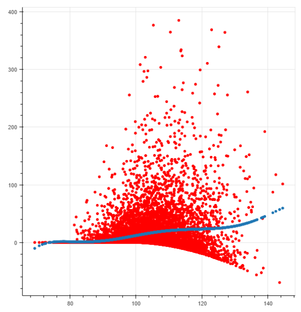
(b) d=3



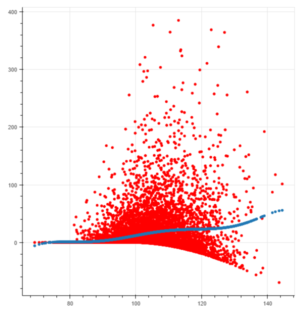
(c) d=4



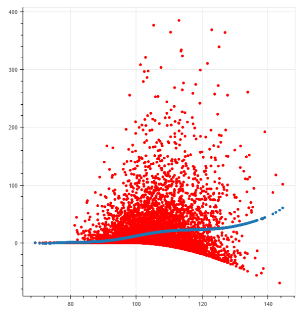
(d) d=5



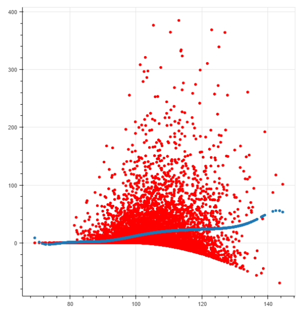
(e) d=6



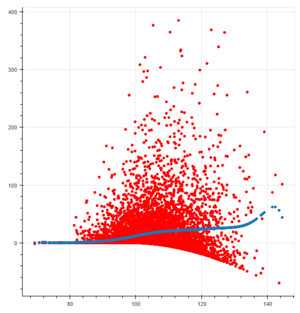
(f) d=7



(g) d=8



(h) d=9



(i) d=10

Figure 5.2: Display of the regression step using LSM method (in blue) with different polynomial degrees and display of $Y_t \frac{\Delta B_t}{\Delta t}$ with respect to, both with respect to X_t

Regression of noisy process Z with polynomials does not cover all the noisy process. Moreover, peripheral points are confronted to high variance depending on the degree.

Mesh

With Black-Scholes formulation, density function takes the following expression

$$f(t_i, X_{i,j}, X_{i+1,k}) = \frac{1}{\sigma\sqrt{\Delta t}}\phi\left(\frac{\ln(\frac{X_{i+1,k}}{X_{i,j}} - (\mu - \frac{1}{2}\sigma^2))}{\sigma\sqrt{\Delta t}}\right) \tag{5.6}$$

We inject then in weights expression 4.3 and draw results in following table 5.2. Prices computed are given with respect to N samples and m discretization time.

Table 5.2: Mesh with number of paths and time discretization

m	N = 100	N = 1000	N = 2000	N = 4000	N = 10000
4	7.249 (0.951)	7.174 (0.309)	7.145 (0.219)	7.151 (0.153)	7.152 (0.096)
6	7.184 (1.051)	7.166 (0.324)	7.155 (0.214)	7.157 (0.154)	7.152 (0.098)
8	7.17 (0.97)	7.143 (0.328)	7.163 (0.235)	7.16 (0.156)	7.157 (0.098)
10	7.186 (1.109)	7.19 (0.315)	7.141 (0.219)	7.165 (0.161)	7.145 (0.099)
12	7.171 (1.015)	7.147 (0.316)	7.147 (0.22)	7.144 (0.169)	7.161 (0.101)

Remark 5.2.1. As the number of samples increases, standard deviation decreases as expected, and price seems to converge to theoretical 7.15. Moreover, when number of paths is big, like 12, we notice an inaccuracy, due to terms in $\frac{1}{\Delta t}$ becoming bigger, thus making Z even more noisy.

The following figure show one regression using this mesh approximation.

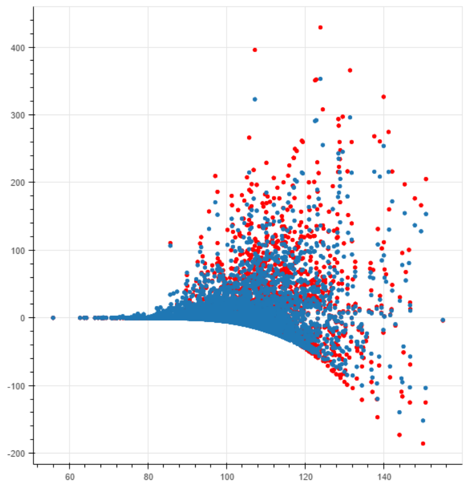


Figure 5.3: Display of the regression step using mesh method (in blue) and display of $Y_t\frac{\Delta B_t}{\Delta t}$ (red) , both with respect to X_t

Random Forest

Table 5.3: Random Forest with number of paths and number of trees generated

n_trees	N = 100	N = 1000	N = 10000	N = 100000
10	6.917 (0.988)	6.984 (0.322)	7.207 (0.117)	7.238 (0.031)
50	7.187 (1.208)	7.09 (0.295)	7.221 (0.132)	7.218 (0.026)
100	7.229 (1.057)	7.069 (0.375)	7.159 (0.107)	7.211 (0.036)
150	7.26 (0.921)	7.055 (0.307)	7.202 (0.086)	7.221 (0.028)
200	6.61 (0.836)	7.209 (0.242)	7.198 (0.103)	7.226 (0.026)

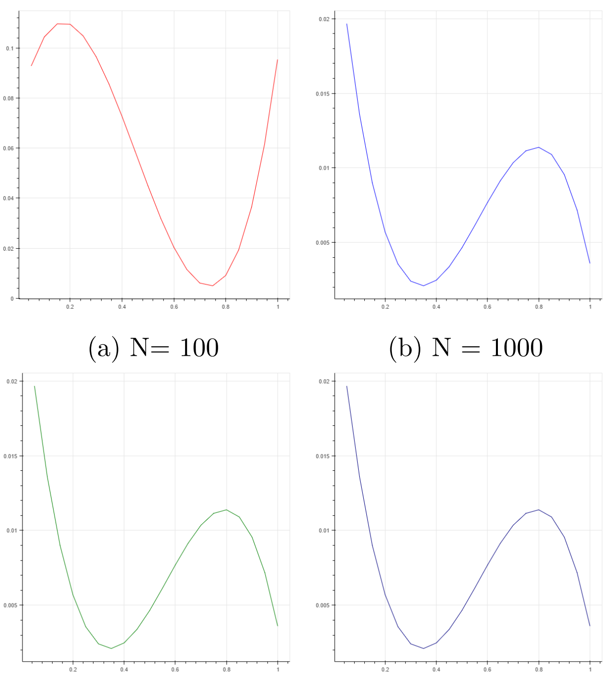
Obviously, the more the number of trees generated, the smaller the variance. Unfortunately, playing only with the number of trees gives a highly biased result. Indeed, with 200 trees and 10^5 samples, we get $Y_0 = 7.226$, far from the expected 7.15.
Referring the reader to Appendix B, multiple sensitive parameters can be the source of this bias. We focus on the most sensitive one, the number of maximum leafs allowed in the tree (related then to depth of the tree).

Remark 5.2.2. Even though generating 200 trees does not provide a better result, Appendix B shows this parameter has to be taken bigger than 150 at least. We will fix it to 200 and try to calibrate the number of maximum allowed leafs.

Table 5.4: Random Forest with number of samples N and maximum number of leafs(in percentage of N)

max_leafs	N = 100	N=1000	N=10000	N=50000
5%	7.115 (0.795)	7.027 (0.265)	7.096 (0.101)	7.101 (0.036)
10%	6.717 (1.031)	7.023 (0.316)	7.073 (0.1)	7.062 (0.049)
15%	6.767 (0.944)	7.017 (0.291)	7.063 (0.117)	7.069 (0.036)
20%	6.796 (0.773)	7.114 (0.244)	7.074 (0.078)	7.101 (0.047)
25%	6.922 (0.977)	7.2 (0.323)	7.115 (0.09)	7.127 (0.046)
30%	6.71 (1.235)	7.139 (0.375)	7.119 (0.092)	7.159 (0.045)
35%	6.881 (1.046)	7.168 (0.338)	7.173 (0.109)	7.192 (0.044)
40%	7.244 (1.146)	7.182 (0.289)	7.218 (0.118)	7.195 (0.041)
45%	6.981 (0.85)	7.243 (0.312)	7.218 (0.108)	7.218 (0.04)
50%	7.281 (1.091)	7.234 (0.367)	7.221 (0.091)	7.222 (0.051)
55%	6.893 (0.954)	7.246 (0.325)	7.224 (0.089)	7.226 (0.05)
60%	7.164 (0.932)	7.083 (0.285)	7.23 (0.109)	7.23 (0.051)
65%	7.069 (0.99)	7.239 (0.333)	7.226 (0.081)	7.222 (0.042)
70%	7.158 (1.035)	7.206 (0.378)	7.247 (0.078)	7.233 (0.037)
75%	7.211 (1.138)	7.211 (0.267)	7.206 (0.107)	7.215 (0.041)
80%	6.934 (1.027)	7.291 (0.226)	7.237 (0.097)	7.224 (0.05)
85%	6.878 (1.016)	7.287 (0.326)	7.242 (0.097)	7.222 (0.044)
90%	6.939 (1.134)	7.255 (0.321)	7.219 (0.101)	7.234 (0.04)
95%	7.119 (1.14)	7.206 (0.35)	7.217 (0.089)	7.238 (0.049)
100%	6.834 (0.99)	7.199 (0.379)	7.234 (0.109)	7.243 (0.037)

Let's draw the squared error from theoretical price of 7.15, for every N, ie $||Y_0(N) - 7.15||^2$. We smooth the results with a polynomial fit of degree 3.

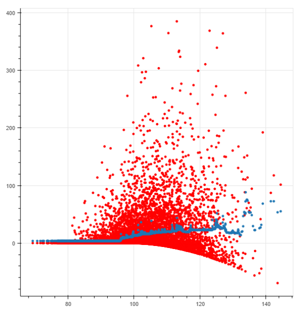


(a) N= 100 (b) N = 1000

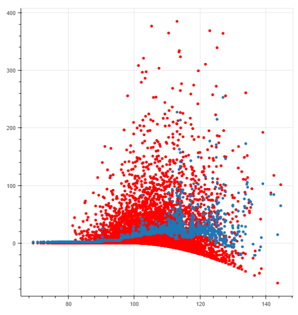
(c) N = 10.000 (d) N = 50.000

Figure 5.4: Pricing error with number of maximum leafs for different number of samples N

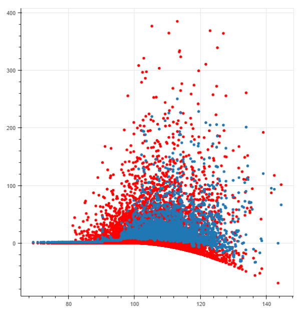
Apart from N=100, the number of maximum leafs seems to be optimal when limited to 30%-35% of the samples N.
Higher values must be due to over-fitting, when smaller values are due to under-fitting. See Appendix B for more details.



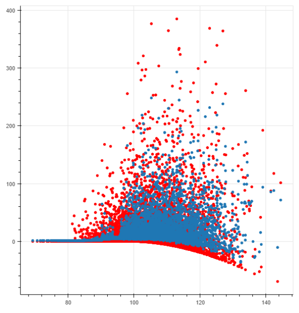
(a) b=10



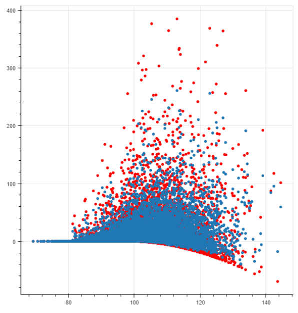
(b) b=100



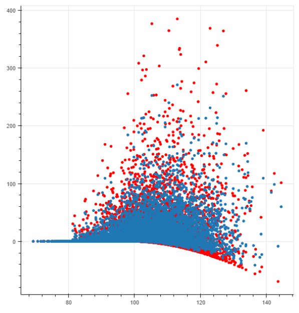
(c) b=500



(d) b=1000



(e) b=1000



(f) b=10000

Figure 5.5: Display of the regression step using RF method (in blue) and display of $Y_t \frac{\Delta B_t}{\Delta t}$ (red) , both with respect to X_t, fixing the number of samples to 10.000 and playing with number of leafs b

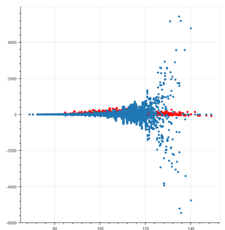
Derivative

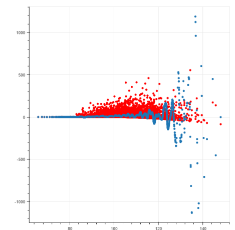
Processing in a first time a Random Forest, we follow the derivative algorithm presented in previous chapter, using kernel regression.
Let us draw prices computed with respect to scaling parameter l, and samples N.

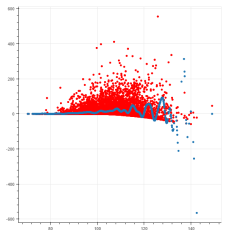
Table 5.5: Derivative method with number of paths and scale parameter l

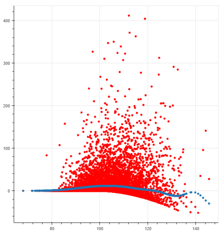
l	N = 100	N = 500	N = 1000	N = 4000	N = 10000
0.1	7.54 (1.156)	7.26 (0.383)	7.245 (0.286)	7.278 (0.174)	7.231 (0.102)
0.2	6.944 (0.933)	7.223 (0.376)	7.224 (0.299)	7.276 (0.176)	7.249 (0.118)
0.3	7.232 (0.848)	7.206 (0.425)	7.284 (0.354)	7.273 (0.131)	7.253 (0.103)
0.4	7.078 (1.123)	7.232 (0.469)	7.307 (0.372)	7.325 (0.182)	7.266 (0.114)
0.5	6.989 (0.958)	7.218 (0.491)	7.243 (0.304)	7.327 (0.184)	7.275 (0.123)
1	7.257 (1.007)	7.406 (0.436)	7.366 (0.271)	7.241 (0.175)	7.278 (0.112)
2	7.522 (0.932)	7.37 (0.436)	7.387 (0.255)	7.241 (0.157)	7.274 (0.097)
5	7.149 (0.993)	7.471 (0.48)	7.264 (0.286)	7.275 (0.169)	7.316 (0.1)
10	7.509 (0.787)	7.429 (0.355)	7.303 (0.264)	7.353 (0.167)	7.339 (0.098)
20	7.607 (0.972)	7.312 (0.449)	7.339 (0.463)	7.44 (0.167)	7.441 (0.095)
50	7.126 (0.939)	7.304 (0.412)	7.536 (0.345)	7.456 (0.17)	7.462 (0.097)
100	7.532 (1.328)	7.486 (0.456)	7.46 (0.346)	7.488 (0.149)	7.473 (0.085)
200	7.29 (0.754)	7.46 (0.492)	7.421 (0.227)	7.468 (0.173)	7.472 (0.123)

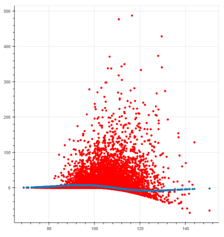
Despite a smoother process Z and calibration of scaling parameter l, results are too far from expectation of 7.15, both biased and with high variance.
The following figures mostly explain high variances noticed before.

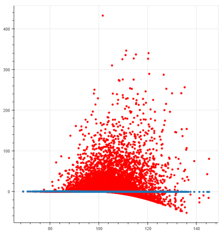

(a) l = 0.1


(b) l = 0.5


(c) l = 1


(d) l = 5


(e) l = 10


(f) l = 100

Summary

Specifically to each method, table (5.6) gives the best calibration, and main statistical measures, with the time average of one run. Results correspond to the same number of time step, $m = 6$.

Table 5.6: Bid-ask European call option with one asset

Method	European Call			

	Random Forest
	$(200, 3000)$
	N = 10.000
		Least-Square
		degree = 6
		N = 100.000
			Mesh
			
			$N = 10000$
				Derivative
				l = 1.0
				$N = 4000$

mean	7.142	7.155	7.152	7.231
95% CI	$[7.134, 7.150]$	$[7.152, 7.159]$	$[7.117, 7.187]$	$[7.191, 7.271]$
time average	35s	5s	30s	4s

Remark 5.2.3. C.I stand for confidence interval, while (200, 3000) corresponds to number of trees generated and maximum number of leafs allowed respectively.

Remark 5.2.4. Random Forest performs well in accuracy, but time computation is high compared to the other methods. However, with higher dimensions, this method will prove efficient...

Remark 5.2.5. For the following applications, an equivalent table to 5.6 will be given only. However, the same calibrations have been run to obtain the best hyper-parameters.

5.2.2 Bid-ask European call combination

Once again, we take an example from Gobet, Lemor and Warin's 's article [13], where we consider still two different rates (bid-ask model). This time, the differences $R - r$ and $\mu - r$ are larger, as $\mu = 0.05$, $r = 0.01$ and $R = 0.06$. This emphasizes the role of the noisy process Z during the regression steps. We keep working with a 20% volatility $\sigma = 0.02$, no dividend yield $q = 0$, and an initial asset value $X_0 = 100$. We consider a call combination derivative, with pay-off $(X_T - K_1)^+ - 2(X_T - K_2)^+$, where $K_1 = 95$ and $K_2 = 105$.

In their simulations, Gobet, Lemor and Warin's use projections on function basis to approximate the conditional expectations. They make use of hypercubes partitions, Voronoi partitions (VP) and global polynomials.
There is no theoretical price for this example, as the non-linearity of driver has a real impact. The option buyer has alternatively to borrow and lend money to hedge his position.
However, a simulation price given by Hypercubes and VP seem to converge to 2.95, which will be taken as reference. Applying our derivative, stochastic mesh, LSM and Random Forest methods to this example, we get

Table 5.7: Bid-ask European call combination option with one asset

	Call combination			

Method	Random Forest
	$(200, 3000)$
	N = 10.000
		Least-Square
		degree = 6
		N = 100.000
			Mesh
			
			$N = 10000$
				Derivative
				l= 1.0
				$N = 4000$

mean	2.969	2.938	2.921	2.940
95% C.I	$[2.965, 2.974]$	$[2.937, 2.940]$	$[2.904, 2.938]$	$[2.912, 2.969]$
time average	45s	5s	30s	9s

Remark 5.2.6.

5.2.3 CVA

Following Guyon and Labordère example [23], i.e with $X_0 = 1.$, $\sigma = 0.2$, $\mu = r = 0.$, $g : x \to 1 - 2.\mathbb{1}_{x>1}$ (final condition) and $\beta = 10\%$

Table 5.8: CVA via BSDE compared to Guyon and Labordère PDE method

Maturity (years)	PDE with poly	Random Forest (std)
2	11.62	11.663(0.011)
4	16.54	16.613 (0.009)
6	20.28	20.595(0.010)
8	23.39	23.587(0.012)
10	26.11	26.107 (0.009)

Remark 5.2.7. We train 150 trees on $N = 10.000$ particles generated, and to avoid overfitting, we limit the depth of trees to 1000 leafs, i.e 10% of the samples. Time average for a run is around 6s.

5.2.4 American Call Option

We consider an American maximum call option with ten exercising

r	σ	q	T	K	X_0
0.05	0.2	0.1	3	100	100

with final pay-off $Y_T = (X_T - K)^+$. We take the results by Glasserman (2004) [17] as a comparison, who get a price of 7.98 using a binomial lattice.

Table 5.9: Call combination of geometric average

	Methods tested	
Method	LSM	Random Forest
	$N = 100.000$	$(200, 850)$
		$N = 10.000$
mean	7.977	7.990
95% C.I	[7.974, 7.988]	[7.938, 8.041]
time average	2s	20s

5.3 High Dimension

5.3.1 Geometrical average on seven assets following bid-ask model

After managing to tune our Random Forest hyperparameters in one dimension cases, we now turn to more sophisticated problems. First case which drew our attention is the pricing of a Geometrical Average call option on seven assets. This derivative has a payoff $((\prod_{i=1}^{7} X_T^{(i)})^{\frac{1}{7}} - K)^+$. Taking seven independent assets in Black-Scholes model, this problem is equivalent to taking a one dimensional European call.
Indeed, let us assume $\forall i \in \{1, \cdots, d\}$, $X_T^i = X_0^i \exp^{(r-\frac{\sigma^2}{2})T+\sigma B_T^i}$. Then

$$
\begin{aligned}
(\prod_{i=1}^{7} X_T^{(i)})^{\frac{1}{d}} &= X_0 \exp^{(r-\frac{\sigma^2}{2})T+\frac{\sigma}{d}\sum_{i=1}^{d} B_T^i} \\
&\overset{\mathcal{L}}{\sim} X_0 \exp^{(r-\frac{\sigma^2}{2})T+\frac{\sigma}{d}\sum_{i=1}^{d} \mathcal{N}(0,T)} \\
&\overset{\mathcal{L}}{\sim} X_0 \exp^{(r-\frac{\sigma^2}{2})T+\frac{\sigma}{d}\mathcal{N}(0,dT)} \\
&\overset{\mathcal{L}}{\sim} X_0 \exp^{(r-\frac{\sigma^2}{2})T+\frac{\sigma}{\sqrt{d}}\mathcal{N}(0,T)}
\end{aligned}
$$

Taking

$$
\tilde{r} - \frac{\tilde{\sigma}^2}{2} = r - \frac{\sigma^2}{2} \tag{5.7}
$$

$$
\tilde{\sigma} = \frac{\sigma}{\sqrt{d}} \tag{5.8}
$$

$$
\tilde{r} = r + \frac{\sigma^2}{2}\frac{d-1}{d} \tag{5.9}
$$

$$
\tilde{\sigma} = \frac{\sigma}{\sqrt{d}} \tag{5.10}
$$

$$
(\prod_{i=1}^{7} X_T^{(i)})^{\frac{1}{d}} \overset{\mathcal{L}}{\sim} X_0 \exp^{(\tilde{r}-\frac{\tilde{\sigma}^2}{2})T+\tilde{\sigma}\mathcal{N}(0,T)}
$$

Hence, a theoretical price is given by usual Black-Scholes formula

$$
Y_0 = BS(T, K, X_0, \tilde{r}, \tilde{\sigma}) = 3.308
$$

where $X_0 = 100\mathbb{1}_d$ ($\mathbb{1}_d$ being the unit vector with d ones), $K = 100.$, $\sigma = 0.2Id$, $r = 0.04$, $R = 0.06$, $\mu = 0.06\mathbb{1}_d$ and $q = 0$.

Random Forest Analysis

K	b= 200	b = 500	b = 1000	b = 1500	b=2000
1	3.29 (0.034)	3.256 (0.035)	3.238 (0.032)	3.274 (0.037)	3.272 (0.039)
2	3.286 (0.038)	3.25 (0.029)	3.279 (0.032)	3.294 (0.032)	3.317 (0.028)
3	3.266 (0.033)	3.256 (0.033)	3.293 (0.038)	3.327 (0.033)	3.345 (0.029)
4	3.278 (0.044)	3.266 (0.03)	3.297 (0.035)	3.336 (0.028)	3.359 (0.032)
5	3.263 (0.039)	3.272 (0.038)	3.317 (0.035)	3.344 (0.042)	3.375 (0.027)
6	3.273 (0.04)	3.267 (0.037)	3.32 (0.041)	3.357 (0.043)	3.38 (0.05)
7	3.257 (0.034)	3.289 (0.041)	3.32 (0.041)	3.34 (0.037)	3.369 (0.036)

Table 5.10: Random Forest with maximum number of features K and maximum number of leafs b

Given the uncorrelated assets, number of features does not seem to affect so much the pricing. However it seems to affect the number of leafs to calibrate. Literature [15] advices an optimal number of features $K = \frac{d}{3}$ to take when running a Random Forest.

K	b= 200	b = 500	b = 1000	b = 1500	b=2000
1	75.03	75.28	78.15	80.18	83.77
2	73.88	77.22	98.62	91.44	94.14
3	83.18	87.33	89.55	94.38	95.73
4	82.91	87.58	92.49	94.46	100.41
5	83.5	90.26	96.04	102.63	106.44
6	86.12	93.37	102.63	122.22	112.97
7	89.46	100.47	109.51	115.49	119.11

Table 5.11: Random Forest time computation with dimension d and maximum number of leafs in seconds

Results

Table 5.12: Bid-ask Geometric Average European Call option on 7 assets

	Geometric Average European Call		
Method	Random Forest	Mesh	Derivative
	$(200, 700)$		l = 1.0
	N = 10.000	N = 10.000	$N = 4000$
mean	3.299	3.318	3.330
95% confidence interval	$[3.296, 3.304]$	$[3.310, 3.327]$	$[3.260, 3.401]$
time average	1min10s	2min20s	40s

Remark 5.3.1. Even if Derivative method performs better in matter of time, its variance is too high compared to Random Forest one. Mesh method performs well in term of accuracy, but time computation does not challenge Random Forest

Remark 5.3.2. For Mesh in high dimension, density functions can be expressed as the product of one-dimensional densities.

5.3.2 Geometrical average on twenty assets following bid-ask European call combination model

Expected 5.70, using the equivalent problem of a call combination on one asset. As this problem does not have a theoretical value, and literature does not offer comparison, we take the result obtained by Random Forest ran on 100.000 samples (which was the most reliable method in one dimension).

Table 5.13: Call combination of geometric average

Method	Methods tested		
	Random Forest $(200, 700)$ $N = 10.000$	Mesh $N = 4000$	Derivative $l = 1.$ $N = 4000$
mean	5.686	5.688	5.6694
95% confidence interval	[5.675, 5.697]	[5.684, 5.693]	[5.691, 5.708]
time average	1min52s	5min48s	3min

Remark 5.3.3. We limited every tree generated to four features chosen randomly, which allowed a faster computation than mesh, and with more samples ! We can notice a good estimation provided by derivative method when dimension is higher.

5.3.3 Max Call Option on 5 assets

We consider an American maximum call option with ten exercising

r	σ	q	T	K	X_0
0.05	0.2	0.05	3	100	100

with final pay-off $Y_T = \left(\underset{i}{max} X_T^{(i)} - K\right)^+$. We take the results by Broadie and Glasserman (2004) [5] as a comparison, who get a price of 23.052 using stochastic mesh.

Method	Methods tested		
	Random Forest $N = 10.000$ $(200, 700)$	Mesh $N = 4000$	Derivative $N = 4000$ $l = 1.$
mean	23.016	22.949	22.913
95% confidence interval	[22.988, 23.044]	[22.905, 22.992]	[22.844, 22.982]
time average	20s	29s	14s

Table 5.14: Max Call option on 5 assets

Remark 5.3.4. This 5 dimensional problem has a linear driver, thus a low computation obtained. However, Mesh and derivative seem far from price provided by Glasserman.

5.3.4 Exchange Option between two correlated assets

We consider in this model two correlated assets $X^{(1)}$ and $X^{(2)}$ to define an exchange pay-off $\xi(X_T^{(1)}, X_T^{(2)}) = \left(X_T^{(1)} - X_T^{(2)}\right)^+$. This 2-dimensional problem is really interesting, as it has a theoretical solution, and is going to test how Random Forest performs here when there is correlation. This theoretical solution , called Margrabe Formula is given by Poulsen and Rolf in [26]. Let us assume that the interest rate is constant (r) and that the underlying assets follow correlated ($\langle dB_t^{(1)}, dB_t^{(2)} \rangle = \rho dt$) geometric Brownian motions under the risk-neutral measure.
Time t-value $Y_t = u(t, X_t^{(1)}, X_t^{(2)})$ of this exchange option is

$$Y_t = X_t^{(1)} \exp^{(\mu_1 - r)(T-t)} \Phi(d_+) - X_t^{(2)} \exp^{(\mu_2 - r)(T-t)} \Phi(d_-)$$

where

$$d_\pm = \frac{\ln\left(\frac{X_t^{(1)}}{X_t^{(2)}}\right) + (\mu_1 - \mu_2 \pm \frac{\tilde{\sigma}^2}{2})}{\tilde{\sigma}\sqrt{T-t}}$$

and

$$\tilde{\sigma} = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho}$$

Table 5.15: Bid-ask European call option with one asset

	Random Forest (Trees, Max_leafs)		
	$(200, 1000)$	$(200, 2000)$	$(200, 3000)$
mean	4.077	4.128	4.181
95% confidence interval	[4.044, 4.091]	[4.109, 4.147]	[4.163, 4.199]
average time	55s	60s	65s

Remark 5.3.5. Random Forest performs well with limitation to 20% of leafs for these two correlated assets problem. Hence the importance of prior calibration when using this method.
We do not run a mesh for this example as density function for such correlated paths does not exist analytically.

5.3.5 Exchange Option between the average of twenty correlated assets

We analyse in this case an equivalent of previous exchange option, but with twety correlated assets. The pay-off $\xi(X_T^{(1)}, \cdots, X_T^{(20)}) = \left(\prod_{i=1}^{10} X_T^{(i)} - \prod_{i=11}^{20} X_T^{(i)}\right)^+$. For the rest, the parameters are equivalent to previous case.

Table 5.16: Bid-ask European call option with one asset

	Random Forest (Trees, Max_features)			
	$(200, 1)$	$(200, 3)$	$(200, 6)$	(200, 10)
mean	1.395	1.351	1.360	1.362
95% CIl	[1.367, 1.411]	[1.347, 1.357]	[1.357, 1.363]	[1.361, 1.363]
average time	1mn10s	2mn11s	3mn30s	5mn

A theoretical price can be derived like using previous formula with equivalent ρ and σ_1, σ_2 especially. This formula gives an equivalent price of 1.356, which compares well to simulations with more than 3 features.

Chapter 6

Second Order BSDE

This section presents an extension to 1-BSDE models, with a direct application on non-linear PDE. Cheridito, Soner, Touzi and Victoir [6] introduced the notion of Second Order BSDEs (2-BSDEs).
We will give a quick definition of the problem, before explaining the link with non-linear PDE. We will finish with an example of discretization for this type of BSDE. For a larger review of the theory of 2-BSDEs, we refer to Soner, Touzi and Zhang [28].

6.1 Definition

We now introduce second order BSDEs for which the corresponding PDE can be non-linear in the second order derivatives and are therefore connected to HJB equations with a control on the diffusion coefficient. Examples of such HJB equations include the Black-Scholes-Barenblatt equation. The following definition is given by [6]

Definition 6.1.1. Let $(s,x) \in [0,T) \times \mathbb{R}^d$ and $(Y_t, Z_t, \Gamma_t, \alpha_t)_{t\in[0,T]}$ be a quadruple of $(\mathcal{F}_t)$-adapted processes taking values in $\mathbb{R}$, $\mathbb{R}^d$, $\mathcal{S}^d$ [1], and $\mathbb{R}^d$ respectively. We call (Y, Z, Γ, α) a solution to a 2-BSDE corresponding to $(X^{s,x}, f, g)$ [2] if

[1]Space of symmetrical matrices with dimension
[2]$X_s = x$

$$\begin{aligned} dY_t &= -f(t, X_t^{s,x}, Y_t, Z_t, \Gamma_t)dt + Z_t' \diamond dX_t^{s,x} \quad t \in [s,T) \\ dZ_t &= \alpha_t dt + \Gamma_t dX_t^{s,x} \quad t \in [s,T) \\ Y_T &= g(X_T^{s,x}) \end{aligned} \tag{6.1}$$

where $\diamond$ represents the Stratanovich integral, related to Itô integration by

$$\begin{aligned} Z_t' \diamond dX_t^{s,x} &= Z_t' dX_t^{s,x} + \frac{1}{2} d\langle Z, X_t^{s,x}\rangle_t \\ &= Z_t' dX_t^{s,x} + \frac{1}{2} Tr\left[\Gamma_t d\langle X_t^{s,x}, X_t^{s,x}\rangle_t\right] \\ &= Z_t' dX_t^{s,x} + \frac{1}{2} Tr\left[\Gamma_t \sigma(t, X_t^{s,x})\sigma(t, X_t^{s,x})'\right] dt \end{aligned}$$

6.2 Existence and Uniqueness

The reader can refer to "Wellposedness of Second Order Backward SDEs" [28] article which provides an existence and uniqueness theory.

6.3 Second order BSDE and non-linear PDE

Let us consider the non-linear PDE

$$\partial_t u + \mathcal{L}u + f(t, x, u, D_x u(t,x), D_x^2 u(t,x)) = 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d \tag{6.2}$$

with the terminal condition $u(T;x) = g(x)$ and $\mathcal{L}$ the Itô generator of X. Such an equation appears when we consider a stochastic control problem with no control on the diffusion coefficient. ($Y_t = u(t;X_t)$, $Z_t = D_x u(t;X_t)$, $\Gamma_t = D_x^2 u(t;X_t)$, $\alpha_t = (\partial_t + \mathcal{L})D_x u(t, X_t)$) is a solution to the 2-BSDE under some assumptions

Proposition 6.3.1. *Generalization of Feynman-Kac's formula :*
Let u be a function of $\mathcal{C}^{1,2}$ satisfying (6.2) and suppose that there exists a constant C such that, for each $(t;x) \in [0;T] \times \mathbb{R}^d$

$$|\sigma(t,x)^T D_x u(t;X_t)| \leq C(1 + |x|) \tag{6.3}$$

Then $(Y_t = u(t;X_t); Z_t = D_x u(t;X_t), \Gamma_t = D_x^2 u(t,x))$ is the unique solution to 2- BSDE (6.1).

6.4 Discretization

Using the previous tricks (multiplying by dB_t in dZ_t expressions and using measurability properties) gives us the following discretization for the 2-BSDE :

$$
\begin{aligned}
Y_{t_n} &= g(X_{t_n}) \\
Z_{t_n} &= Dg(X_{t_n}) \\
\Gamma_{t_i} &= \frac{1}{\Delta t_i}\mathbb{E}[Z_{t_{i+1}}\Delta B_{t_i}^T|\mathcal{F}_{t_i}]\sigma(t_i, X_{t_i})^{-1} \\
Z_{t_i} &= \sigma(t_i, X_{t_i})'^{-1}\frac{1}{\Delta t_i}\mathbb{E}[Y_{t_{i+1}}\Delta B_{t_i}|\mathcal{F}_{t_i}]
\end{aligned}
$$

$$
Y_{t_i} = \mathbb{E}[Y_{t_{i+1}}|\mathcal{F}_{t_i}] + (f(t_i, S_{t_i}, Y_{t_{i+1}}, Z_{t_i}, \Gamma_{t_i}) - \mathtt{tr}[\sigma(t_i, X_{t_i})\sigma(t_i, X_{t_i})'\Gamma_{t_i}])\Delta t_i
$$

Given Y_{t_n} we can get Y_0 using the previous discretization backwardly.

6.5 Application : Portfolio Optimization

Arash Fahim, Nizar Touzi and Xavier Warin analysed an example of 2-BSDE, the continuous-time portfolio optimization in financial mathematics ([1]). Two dimensional and Five dimensional examples are solved via non-linear PDE. We tried to solve the problems using 2-BSDE method presented previously, but results obtained do not match exactly the author's one. The reader can refer to the code we released on github (*https://github.com/MajdiRabia/BSDE.git*) to help fixing it, as second derivative Γ can be approximated well using Random Forest regression.

Chapter 7

Conclusion

After reviewing BSDE's theory, we gave here several regression methods to compute conditional expectatiosn appearing in the backward algorithm. Relying on existent methods found in literature, such as Stochastic Mesh by Broadie and Glasserman [18] and Least Square [2], we computed different approaches, Random Forest, Gradient, and Derivative, for comparison purposes. We focused on accuracy (i.e mean and variance), and time analysis to draw simulation conclusions.
In one dimension, fastest method remains LSM, with promising accuracy when calibrated with a polynomial of degree 6. Random Forest performs well in accuracy terms but drawing several trees slows down the process. Derivative method, even after scale parameter tuning, does not offer reliable results, and even if it was accurate enough, computing $N \times N$ matrices costs time. Mesh method applied with amount of samples costs time but provides a good accuracy. Differentiating these methods happens when considering higher dimensions.
In Higher dimension, Random Forest flexibility to choose a certain amount of features for each tree generated, and calibrated with certain percentage of maximum leafs, gives a really good accuracy and computational time.
However, this tree based method requires several steps of calibration, and when no theoretical solution exists, or no bounds can be give, it remains difficult to use.

Chapter 8

Future Work

Mostly computational, this MSc thesis reviews most numerical methods to solve BSDEs, adding a machine learning method (Random Forest) to price higher dimensional options. Given the efficiency of high dimensional pricing with Random Forest, both in accuracy and time computation, compared to stochastic mesh especially, one can now focus on a better calibration method.
Indeed, had it not been for theoretical solution to our examples, or comparisons in literature, tuning the Random Forest parameters would have been trickier. Depth of trees, number of features, number of leafs are all sensitive parameters. We suggest then to the reader some ideas that could be tried, using the code we created and released on github(*https://github.com/MajdiRabia/BSDE.git*).
First, 'RandomizedsearchCV' and 'RandomizedsearchCV' are two cross-validation methods existing on scikit-learn package with Python. These two methods could be used during regression step, to calibrate the best hyper-parameters itself. Though it would be timely consuming, it would prove efficient when no theroy exists for a pricing derivative !
Second, reader can improve the existing code for the implemented 2-BSDE computation. The current code is only adapted to examples from Warin, Touzi and Fahim [28] .If a generic code could be implemented would prove powerful to solve any kind of non-linear PDE, as long as it respects initial conditions.

Appendices

Appendix A

Model Assessment and Selection

Inspired by Hastie, Tibshirani and Friedman [14], we present and describe variables used for regression model accuracy measurements. These key methods enabled the comparisons made during financial applications in part 4.

A.1 Bias, Variance and Loss Mean Squared function

Definition A.1.1. Bias
Let $\hat{\mu}$ be an estimator for $\mu = \mathbb{E}[X]$ where X is an integrable random variable. $\hat{\mu}$ is a an unbiased estimator if $\mathbb{E}[\hat{\mu}] = \mu$. Inversely, μ is a biased estimator if the difference between $\mathbb{E}[\hat{\mu}]$ and μ is not zero.

Definition A.1.2. Variance Variance is the amount that the estimate of the target function will change if different training data was used. $Var(\hat{\mu}) = \mathbb{E}[\hat{\mu}^2] - \mathbb{E}[\hat{\mu}]^2$

Given a target variable Y, features $X = (X_1, \cdots, X_p)$ and a regression model $\hat{f}$, estimated from a sample set $\mathcal{S}$, we denote $\mathcal{L}$ the loss mean squared function, measuring prediction errors between Y and $\hat{f}(X)$, and defined by

$$\mathcal{L}(Y, \hat{f}(X)) = ||Y - \hat{f}(X)||^2$$

A.2 Bias-Variance trade-off

In statistics, bias-variance is the dilemma consisting on reducing simultaneously two sources of error, bias and variance.
When interested on the expected value of mean squared loss function, one is faced to generalized problem of $\mathbb{E}[(\mu - \hat{\mu})^2]$, i.e

$$\begin{aligned}\mathbb{E}[(\mu - \hat{\mu})^2] &= \mathbb{E}[(\mu - \mathbb{E}[\hat{\mu}] + \mathbb{E}[\hat{\mu}] - \hat{\mu})^2] \\ &= Bias^2(\hat{\mu}) + Var(\hat{\mu}) + Var(\mu)\end{aligned}$$

Most famous Bias-variance trade-off appears in Ridge and Lasso regression models. When one wants to decrease variance of the model, bias will increase. The bias-variance decomposition framework can be used as a tool for diagnosing under-fitting and over-fitting.

Appendix B

Random Forest

B.1 Complexity

Let us consider p predictor variables $(x_1, \cdots, x_p) \in D \subset \mathbb{R}^p$ of a response variable y. The goal of this subsection is to determine the time complexity of the Random Forest model, given N samples data.
Let $T(N)$ denote the time for building one decision tree.

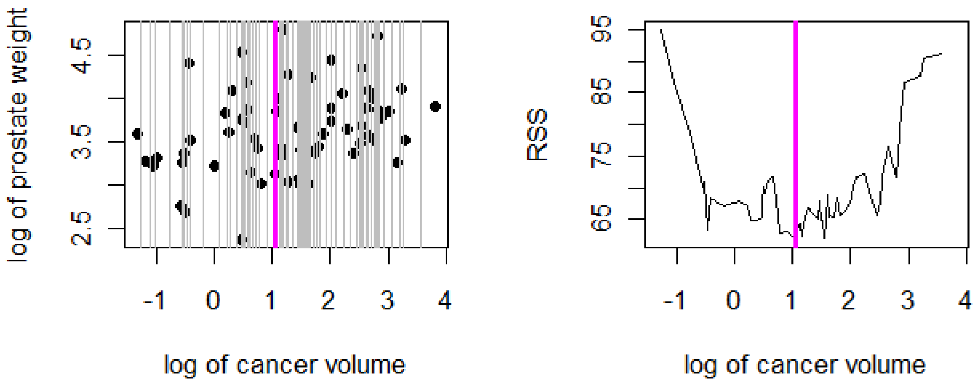


Figure B.1: Choosing the best vertical split using RSS measure

$T(N)$ can be approximated recursively. Indeed, we can grow a decision tree in three steps : find the best splitting, grow a left child tree, and grow a right child tree.
Let us denote the time complexity of these steps by $s(N)$, $T_{left}(N)$ and $T_{right}(N)$ respectively. Hence : $T(N) = s(N) + T_{left}(N) + T_{right}(N)$. We will present both best case and worst case complexity for every step, and we will assume the time

complexity to lay in between.

splitting and partitioning The partitioning step requires at least an iteration over the N samples, so we can already set a linear lower bound on the time complexity within a node. Finding a split can be done using Figure B.1 (example from [9]) method, i.e computing a Residual Sum of Squares (RSS) and minimize it. This operation requires to sort the N samples for every predictor $x_i, i \in \{1, \cdots, p\}$. Time complexity to sort a list with size N is at worst $\mathcal{O}(N)$, and at best $\mathcal{O}(\log N)$. Combining both splitting and partitioning, time complexity is at worst $\mathcal{O}(pN^2)$ and at best $\mathcal{O}(pN \log N)$. Random Forest can limit the search over $K \leq p$ features, which reduces this complexity between $\mathcal{O}(KN \log N)$ and $\mathcal{O}(KN^2)$.

Child left and right trees The best case for growing these two child trees is intuitively $\frac{N}{2}$. The worst case would be having a tree with one pure leaf and $N-1$ samples in the other tree.
Hence,

$$s(N) + T_{left}(\frac{N}{2}) + T_{right}(\frac{N}{2}) \leq T(N) \leq s(N) + T_{left}(1) + T_{right}(N-1)$$

Or

$$s(N) + 2.T(\frac{N}{2}) \leq T(N) \leq s(N) + T(1) + T(N-1)$$

Having a recurrence equation, let us make use of the following theorem.

Theorem B.1.1. *Master Theorem*

$$T(n) = \begin{cases} c & \textit{if } n < d, \\ aT(n/b) + f(n) & \textit{if } n \geq d, \end{cases}$$

where $a \geq 1, b > 1,$ *and* d *are integers and* c *is a positive constant. Let* $\nu = \log_b a$.

Case (i) $f(n)$ **is "definitely smaller" than** n^ν**:** *If there is a small contant* $\epsilon > 0$, *such that* $f(n) \preceq n^{\nu-\epsilon}$, *that is,* $f(n) \prec n^\nu$, *then* $T(n) \sim n^\nu$.

Case (ii) $f(n)$ **is "similar in size" to** n^ν**:** *If there is a constant* $k \geq 0$, *such that* $f(n) \sim n^\nu (\log n)^k$, *then* $T(n) \sim n^\nu (\log n)^{k+1}$.

Case (iii) $f(n)$ **is "definitely larger" than** n^ν**:** *If there are small constants* $\epsilon > 0$ *and* $\delta < 1$*, such that* $f(n) \succeq n^{\nu+\epsilon}$ *and* $af(n/b) \leq \delta f(n)$*, for* $n \geq d$*, then* $T(n) \sim f(n)$*.*

Lemma B.1.2. *For* $\frac{s(N)}{K} = \mathcal{O}(N \log N)$ *and* $T(N) = 2.T(\frac{N}{2}) + s(N)$*, time complexity for building a decision tree is* $T(N) \sim K.N. \log^2 N$*.*

Proof. Apply Theorem B.1.1 with $a = b = 2$, $d = 1$, $k = 1$ and $T(N) \equiv \frac{T(N)}{K}$ □

Lemma B.1.3. *For* $\frac{s(N)}{K} = \mathcal{O}(N \log N)$ *and* $T(N) = s(N) + T(1) + T(N - 1)$*, time complexity for building a decision tree is* $T(N) \sim K.N. \log^2 N$ *(upper bound) and* $T(N)$*.*

Proof. Let us rewrite in a first time the expression of $T(N)$ in a better form to make use of Theorem B.1.1.

$$\frac{s(N)}{K} = \mathcal{O}(N \log N)$$

Let us assume there exists C_1, C_2 such that $C_1 N \log N \leq \frac{s(N)}{K} \leq C_2 N \log N$. Hence, using the right side,

$$\begin{aligned}\frac{T(N)}{K} &= \frac{s(N)}{K} + \frac{T(1)}{K} + \frac{T(N-1)}{K} \\ &\leq C_2 N \log N + \frac{T(1)}{K} + \frac{T(N-1)}{K}\end{aligned}$$

Let $t(N) = \frac{T(N)}{K}$.

$$\begin{aligned}
t(N) &\leq C_2 N \log N + t(1) + t(N-1) \\
\iff t(N) - t(N-1) &\leq C_2 N \log N + t(1) \\
\Rightarrow \sum_{j=2}^{N} t(j) - t(j-1) &\leq (N-1)t(1) + C_2 \sum_{j=2}^{N} j \log j \\
\Rightarrow t(N) &\leq Nt(1) + C_2 \log N \sum_{j=1}^{N} j \\
\Rightarrow t(N) &\leq Nt(1) + C_2 \log N \frac{N(N+1)}{2} \\
\Rightarrow t(N) &= \mathcal{O}(N^2 \log N) \\
\Rightarrow T(N) &= \mathcal{O}(KN^2 \log N)
\end{aligned}$$

Similarly, using the left side gives a lower bound

$$\begin{aligned}
& t(N) \geq C_1 N \log N + t(1) + t(N-1) \\
\iff\ & t(N) - t(N-1) \geq C_1 N \log N + t(1) \\
\Rightarrow & \sum_{j=2}^{N} t(j) - t(j-1) \geq (N-1)t(1) + C_1 \sum_{j=2}^{N} j \log j \\
& \Rightarrow t(N) \geq N t(1) + C_1 \sum_{j=2}^{N} j \\
& \Rightarrow t(N) \geq N t(1) + C_1 (\frac{N(N+1)}{2} - 1) \\
& \Rightarrow t(N) \succeq N^2 \\
& \Rightarrow T(N) \succeq N^2
\end{aligned}$$

□

Finally, growing n_{trees} random trees has in the best case a time complexity $\Theta(n_{trees} K N \log^2 N)$ and in the worst case $\mathcal{O}(n_{trees} K N^2 \log N)$.

Prediction time complexity Iterating over every tree generated by the Random Forest, predicting depends on the depth of the tree. Similarly to previous analysis, we can lower bound and upper bound the depth of one tree. Indeed, let $D(N)$ denote the depth of a tree generated in the Forest.

- Best case is again a $\frac{N_{at\_node\_i}}{2}$ split at every node i, which gives the following recurrence equation for prediction:

$$\begin{cases} D(1) = 1 \\ D(N) = 1 + D(\frac{N}{2}) + D(\frac{N}{2}) = 1 + 2.D(\frac{N}{2}) \end{cases} \tag{B.1}$$

  A quick application of Theorem B.1.1 with $a = b = 2$ and $k = 0$ gives $D(N) \sim \log N$

- Worst case is a split with one simple leaf and a tree with $N - 1$ samples, which gives the following recurrence equation :

$$\begin{cases} D(1) = 1 \\ D(N) = 1 + D(1) + D(N-1) \end{cases}$$

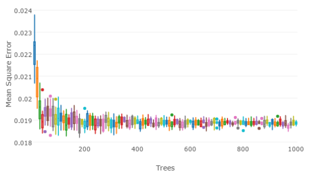
$$
\begin{aligned}
D(N) - D(N-1) &= 1 + D(1) \\
D(N) &= (N+1)D(1) + N \\
D(N) &= \mathcal{O}(N)
\end{aligned}
$$

B.2 Hyperparameters tuning

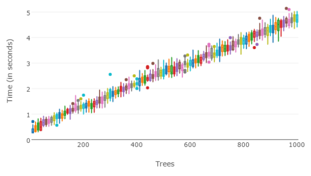
Using a Random Forest on a dataset $\mathcal{S}$ requires the tuning of several parameters, which have impact on time computation, bias and variance. Let's analyse these parameters on a housing dataset, imported from a Kaggle competition [7], with 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. The reader can refer to [14] for a similar analysis on Boston's housing market.
We use for our simulations the *RandomForestRegressor* method from *scikit-learn* library, available on python.
In the following, the mean square error of the regression will be given by the mean of $||y_{predicted} - y||_2^2$

B.2.1 Number of trees generated

We analyse here the MSE against number of trees generated, and the computational time with respect to the same parameter.



(a) Box plot : MSE with increasing number of trees



(b) Time with Trees increasing

Figure B.2: Analysis of number trees in a Random Forest

A expected, the more trees, the smaller the MSE. Time is linear with number of trees, conformal to previous section analysis.

B.2.2 Depth of the trees

This parameter controls the data fit. Indeed, the deeper the tree, and the more over-fitted the model is. Inversely, the smaller the tree, the more under-fitted the model is. There is no theoretical value to take for the depth of a tree. Only analysis of the data can give an answer. Using a Cross Validation method can help figure out which value to take for this parameter.

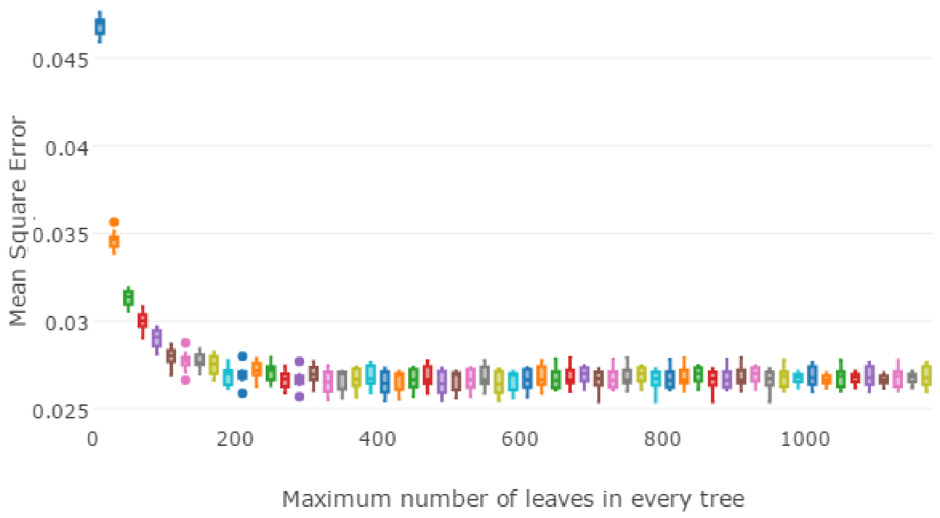


Figure B.3: Box plot giving the mean square error given by Random forest with increasing number of leafs

For the Iowa data available with 1429 samples, taking more and more leafs and depth helps reducing the MSE.

B.2.3 Variable Importance and features selection

Tuning the max_features hyperparameter

Without doubt the most efficient parameter for high dimensional computation is the number of features taken into account when generating a tree. Indeed, when growing multiple trees, one can limit to K features, randomly chosen, and let the Random Forest process happens. Theory for this kind of feature selection in Random Forest is not abundant yet, but here is a quick analysis on our Iowa data of housing market.

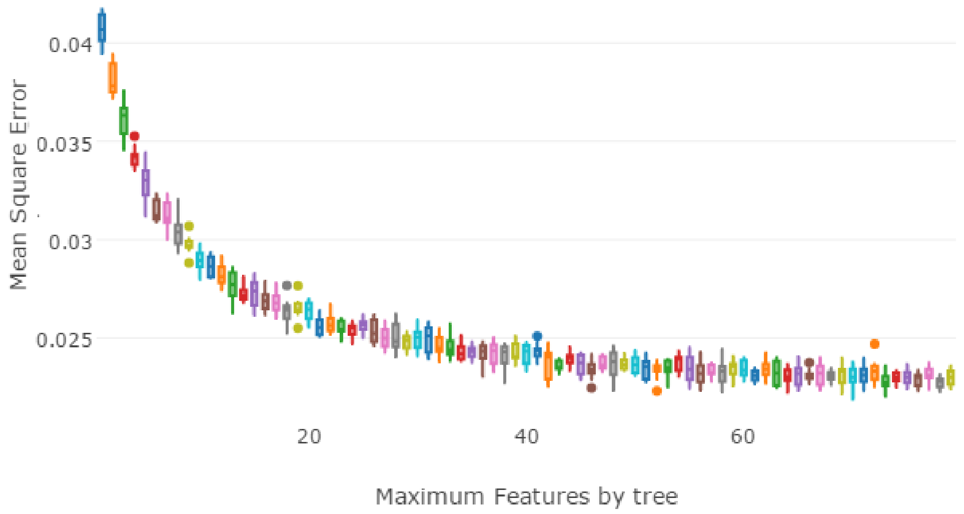


Figure B.4: Box plot giving the mean square error of regression given by Random forest with increasing number of features

Of course, as expected, the bigger the number of features, the better the estimation. However, to get competitive time computation, one can prefer fewer precision, and limit to 40 features in this case, out of 79.
Moreover, the previous graph provides a number of features chosen randomly between the 79 available. But, once can analyse the features first, and make a feature selection, explained in the following part.

Selecting the best features : Variable Importance

One can reduce the high dimensionality of a regression problem by selecting the 'best' features.
Measuring this so called *Variable Importance* makes it possible to rank the features by their importance. The difficult part here is the way to measure a feature's impact on the regression model. Intuitively, the more an input X_i is important, the worse the accuracy when removing it from the regression.
Thus when training a tree, it can be computed how much each feature decreases the impurity in a tree, called impurity score and known as *Gini impurity* for classification and variance for regression.
here method in *Scikit-Learn* Random Forest Regressor class gives this information. Drawing in the previous example a variable importance histogram gives

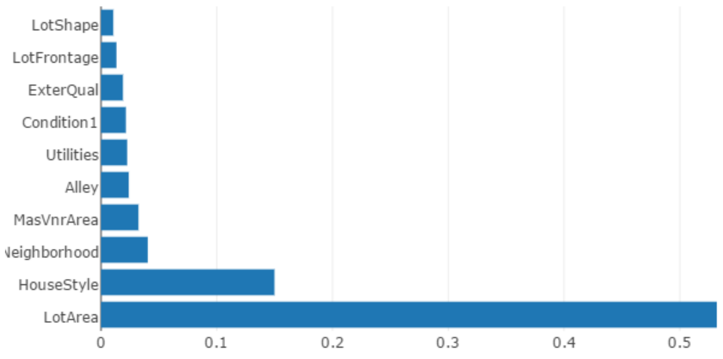


Figure B.5: First ten most important variables given by Random Forest

Obviously the *LotArea* represents the most important variable here.
Had it been that easy, we would then select the best features and rerun a Random Forest. But, redundant information between features affects impurity measurements and thus its ranking. This is not an issue for prediction and avoiding over fitting, but can lead to misinterpretation about a label importance.
For instance

Example B.2.1. let X_0, X_1, X_2, X_3, X_4 and X_5 be six random variables having Gaussian distribution with mean 0 and variance 1. To add some correlation, we choose to add X_0 to every X_i, $\forall i \in \{1, \cdots, 5\}$. Let Y be the sum of the correlated random variables X_i, $\forall i \in \{1, \cdots, 5\}$, i.e $Y = \sum_{i=1}^{5} X_i$. It is clear here every input X_i has the same impact on the output Y. However, computing a Random Forest on 10.000 samples of (X, Y), X being the concatenate of X_i, $i \in \{1, \cdots, 5\}$, gives the following Variable Importance bar plot

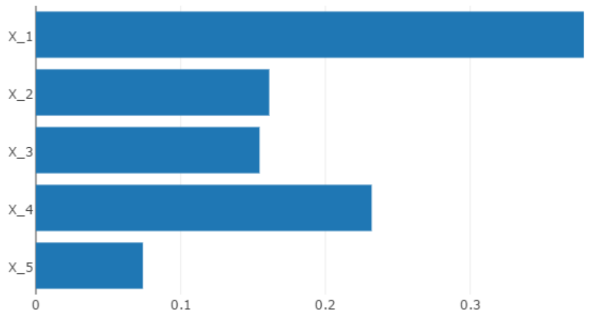


Figure B.6: Variable Importance with 5 correlated features X_i,$i \in \{1, \cdots, 5\}$ respectively taking importances 0.3785, 0.1612, 0.1545, 0.2319, 0.0739.

X_1 is five times 'more important' than X_5, which does not correspond to reality, as all inputs have the same impact on output Y. Random Forest seems to keep in 'mind' the information it gets, and if not needed, it does reduce its importance.

Let's analyse this on the previous housing example :
First, let's draw the correlation matrix between the inputs (we draw only a part here for better visualization):

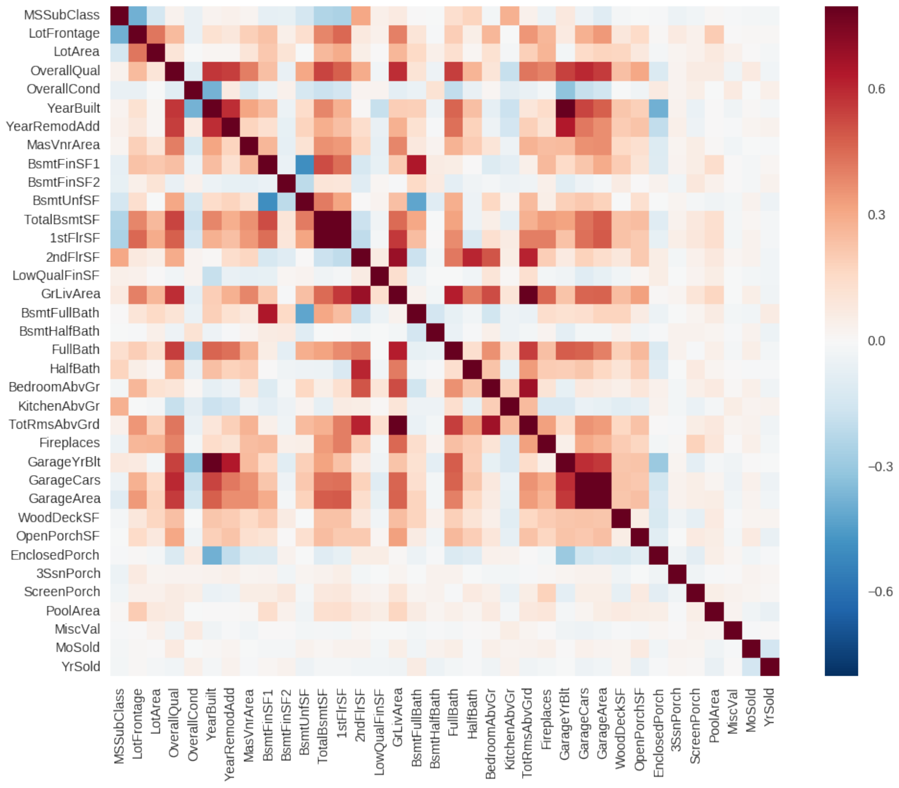


Figure B.7: Correlation matrix between the features of our housing market data

We can see *LotFrontage*(Linear feet of street connected to property) and *LotArea*("Lot size in square feet") are highly correlated (intuitively and by matrix correlation). Growing a Random Forest gives the following Variable Importance ranking

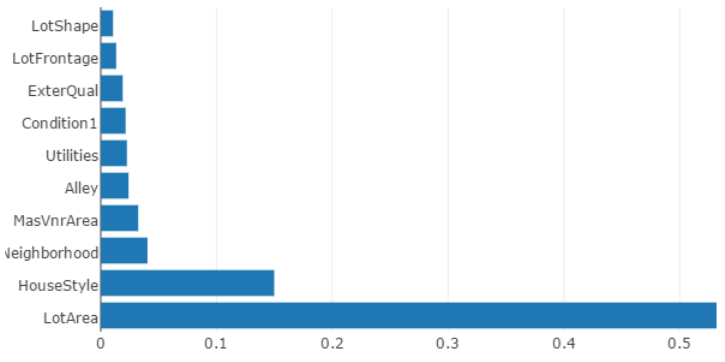


Figure B.8: Correlation matrix between the features of our housing market data

The randomness of the algorithm managed here to make the best of both provided information in *LotFrontage* and *LotArea* and bucket it into the 'most important' one : *LotArea.* Several analysis can be made on features to make the best selection, but this example illustrates how powerful Random Forest can be for regression problems...

Bibliography

[1] Arash Fahim, N. T. and Warin, X. (2011). A probabilistic numerical method for fully nonlinear parabolic pdes. *The Annals of Applied Probability*, 21(4):1322–1364.

[2] Bender, C. and Steiner, J. (2010). Least-squares monte carlo for backward sdes.

[3] Bismut, J. (1973). Analyse convexe et probabilités, phd.

[4] Bouchard B., W. X. Monte-carlo valuation of american options: facts and new algorithms to improve existing methods.

[5] Broadie, M., Glasserman, P., et al. (2004). A stochastic mesh method for pricing high-dimensional american options. *Journal of Computational Finance*, 7:35–72.

[6] Cheridito, P., S. H. and Touzi, N., V. (2007). Second order bsde's and fully nonlinear pde's. *Communications in Pure and Applied Mathematics*, pages 1081–1110.

[7] Cock, D. D. (2016). House prices. *https://www.kaggle.com/c/house-prices-advanced-regression-techniques.*

[8] Crisan, D. and Manolarakis, K. (2014). Second order discretization of backward sdes and simulation with the cubature method. *Annals of Applied Probability*, 24(2):652–678.

[9] Cutler, A. (2010). Random forests for regression and classification.

[10] Delarue, F. and Menozzi, S. (2006). A forward–backward stochastic algorithm for quasi-linear pdes. *The annals of Applied Probability*, 16(1):140–184.

[11] El Karoui, N., Kapoudjian, C., Pardoux, E., Peng, S., and Quenez, M.-C. (1997a). Reflected solutions of backward sde's, and related obstacle problems for pde's. *the Annals of Probability*, pages 702–737.

[12] El Karoui, N., Peng, S., and Quenez, M. C. (1997b). Backward stochastic differential equations in finance. *Mathematical finance*, 7(1):1–71.

[13] Emmanuel Gobet, J.-P. L. and Warin, X. (2005). A regression-based monte carlo method to solve backward stochastic differential equations. *Annals of Applied Probability*, 15(3):2172–2202.

[14] Friedman, T. H. R. T. J. (2008). The elements of statistical learning. *Springer Series in Statistics*, Second Edition:587–604.

[15] Gilles, L. (2015). Understanding random forests. *PhD dissertation.*

[16] Gilles Pagès, A. S. (2015). Improved error bounds for quantization based numerical schemes for bsde and nonlinear filtering.

[17] Glasserman, P. (2004). Monte carlo methods in financial engineering. pages 421–426, 443–.

[18] Glasserman, P. and M.Broadie (1997). A stochastic mesh method for pricing high-dimensional american options. *Economics and Finance*, pages 421–426, 443–.

[19] Gobet, E. and Labart, C. (2010). Solving bsde with adaptive control variate. *Journal of Numerical Analysis*, 48(1):257–277.

[20] GOBET, E. and LABART, C. (2010). Solving bsde with adaptive control variate.

[21] Gobet, E. and Lemor, J.-P. (2008). Numerical simulation of bsdes using empirical regression methods: theory and practice. *arXiv preprint arXiv:0806.4447.*

[22] Guo, G. (2015). Finite differences method for bsdes in finance.

[23] Julien, G. and Pierre-Henry, L. (2013). Non linear option pricing. page 180.

[24] Milstein, G. N. and Tretyakov, M. V. (2006). Numerical algorithms for forward-backward stochastic differential equations. *Journal of Computer Science and Computational Mathematics*, 48(2):561–582.

[25] Pardoux and Peng, S. (1990). Adapted solution of a backward stochastic differential equation. *System and Control Letters*, 14:55–61.

[26] Poulsen, R. (2009). Margrabe formula. *Encyclopedia of Quantitative Finance.*

[27] Rogers, L. C. G. (2016). Bermudan options by simulation.

[28] Soner, Touzi, Z. (2010). Wellposedness of second order backward sdes.

[29] Yaozhong Hu, D. N. and Song, X. (2011). Malliavin calculus for backward stochasticdifferential equations and application to numerical solutions. *The annals of Applied Probability*, 21(6):2379–2423.