FiNANCE FOR ENERGY MARKET RESEARCH CENTRE

Numerical resolution of McKean-Vlasov FBSDEs using neural networks

Maximilien GERMAIN, Joseph MIKAEL, and Xavier WARIN,

Working Paper
RR-FiME-19-08

September 30, 2019

DAUPHINE UNIVERSITÉ PARIS

CREST Centre de Recherche en Économie et Statistique

ÉCOLE POLYTECHNIQUE

EDF

Numerical resolution of McKean-Vlasov FBSDEs using neural networks *

Maximilien Germain †, Joseph Mikael ‡, Xavier Warin §

September 30, 2019

Abstract

We propose several algorithms to solve McKean-Vlasov Forward Backward Stochastic Differential Equations. Our schemes rely on the approximating power of neural networks to estimate the solution or its gradient through minimization problems. As a consequence, we obtain methods able to tackle both mean field games and mean field control problems in high dimension. We analyze the numerical behavior of our algorithms on several examples including non linear quadratic models.

1 Introduction

This paper is dedicated to the numerical resolution in high dimension of the following McKean-Vlasov Forward Backward Stochastic Differential Equations (MkV FBSDEs)

$$\begin{cases} X_t &= \xi + \int_0^t b(s, X_s, Y_s, Z_s, \mathcal{L}(X_s), \mathcal{L}(Y_s), \mathcal{L}(Z_s))\ \mathrm{d}s + \int_0^t \sigma(s, X_s)\ \mathrm{d}W_s \\ Y_t &= g(X_T, \mathcal{L}(X_T)) + \int_t^T f(s, X_s, Y_s, Z_s, \mathcal{L}(X_s), \mathcal{L}(Y_s), \mathcal{L}(Z_s))\ \mathrm{d}s - \int_t^T Z_s\ \mathrm{d}W_s \end{cases} \tag{1}$$

with $b : \mathbb{R}\times\mathbb{R}^d\times\mathbb{R}^k\times\mathbb{R}^{k\times d}\times\mathcal{P}_2(\mathbb{R}^d)\times\mathcal{P}_2(\mathbb{R}^k)\times\mathcal{P}_2(\mathbb{R}^{k\times d}) \mapsto \mathbb{R}^d$, $\sigma : \mathbb{R}\times\mathbb{R}^d \mapsto \mathbb{R}^d$, $g : \mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d) \mapsto \mathbb{R}^k$, and $f : \mathbb{R}\times\mathbb{R}^d\times\mathbb{R}^k\times\mathbb{R}^{k\times d}\times\mathcal{P}_2(\mathbb{R}^d)\times\mathcal{P}_2(\mathbb{R}^k)\times\mathcal{P}_2(\mathbb{R}^{k\times d}) \mapsto \mathbb{R}^k$.
W_t is a d-dimensional $\mathcal{F}_t$-Brownian motion where $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ is a given filtered probability space and $T > 0$.
ξ is a given random variable in $L^2(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{R}^d)$ and $\mathcal{P}_2(\mathbb{R}^n)$ stands for the space of square integrable probability measures over $\mathbb{R}^n$ endowed with the $2-$Wasserstein distance

$$\mathcal{W}_2(\mu, \nu) = \inf\left\{\sqrt{\mathbb{E}[(X - X')^2]} \mid X, X' \in L^2(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{R}^n),\ \mathbb{P}_X = \mu,\ \mathbb{P}_{X'} = \nu\right\}. \tag{2}$$

At last $\mathcal{L}(.)$ is a generic notation for the law of a random variable.

This kind of equation is linked to some non local PDEs kwown as master equations. We refer to (Carmona and Delarue 2018b), chapter 4 and 5 of volume 2, for an introduction on the subject. In (Chassagneux, Crisan, and Delarue 2019), (Chassagneux, Crisan, and Delarue 2014), it is shown for example that under regularity conditions, when the driver b is independent of Z_t, $\mathcal{L}(Z_t)$ and when

*This work is supported by FiME, Laboratoire de Finance des Marchés de l'Energie.
†EDF R&D Maximilien.Germain at edf.fr
‡EDF R&D Joseph.Mikael at edf.fr
§EDF R&D & FiME xavier.warin at edf.fr

f does not depend on $\mathcal{L}(Z_t)$, the resolution of equation (1) provides a way to estimate the solution of the equation

$$\begin{aligned}\partial_t \mathcal{U}(t,x,\mu) + b(x,\mathcal{U}(t,x,\mu),\eta).\partial_x \mathcal{U}(t,x,\mu) + \\ \frac{1}{2}\,\mathrm{Tr}[\partial_{x,x}\mathcal{U}(t,x,\mu)\sigma\sigma^\top(x,\mu)] + f(x,\mathcal{U}(t,x,\mu),\partial_x\mathcal{U}(t,x,\mu)\sigma(x,\mu),\eta) + \\ \int_{\mathbb{R}^d} \partial_\mu \mathcal{U}(t,x,\mu)(y).b(t,y,\mathcal{U}(t,y,\eta),\eta)d\mu(y) + \\ \int_{\mathbb{R}^d} \frac{1}{2}\,\mathrm{Tr}[\partial_x\partial_\mu \mathcal{U}(t,x,\mu)(v)\sigma\sigma^\top(y,\mu)]d\mu(y) = 0,\end{aligned}$$

where η is a notation for the image of the probability measure $\mu : x \in \mathbb{R}^d \longrightarrow (x, \mathcal{U}(t,x,\eta)) \in R^{d+1}$. The equation above is non local due to the integral terms and the term $\partial_\mu \mathcal{U}(t,x,\mu)(y)$ stands for the Wasserstein derivative of $\mathcal{U}$ in the direction of the measure at point (t,x,μ) and taken at the continuous coordinate y.

Equations (1) appear as well as some probabilistic formulations of mean field games or mean field controls leading to finding the value function V of the game. Mean field games were introduced by (Lasry and Lions 2006a) and (Lasry and Lions 2006b) to model games with interactions between many similar players. In this theory, a player cannot observe the states of the other players, however his or her dynamics and cost take into account the distribution of all agents.
Two probabilistic approaches based on Forward Backward Stochastic Differential Equations can be used to solve these problems:

- A first approach called the **Pontryagin approach** consists as shown in (Carmona and Delarue 2018a) in applying the strong Pontryagin principle to these control problems. Under regularity and convexity conditions, Y_t appears to be a stochastic representation of the gradient of the value function V.
- Another approach called the **Weak approach** permits to solve the optimization problem by estimating directly Y_t as the value function V of the problem as shown in (Carmona and Lacker 2015).

The numerical resolution of equations (1) is rather difficult as far as:

- The dynamics are coupled through both the drift and the driver of the BSDE.
- The McKean-Vlasov structure of the problem requires to solve a fixed point in probability spaces.

The weak approach applied to mean field problems gives a problem in low dimension but often with quadratic coupling in Z_t appearing in the backward dynamic, whereas the Pontryagin approach gives a problem in high dimension but with a linear coupling in Y_t giving some equations easier to solve numerically.
In the case of mean field games only the law of X_t is present in the dynamic of (1). In other applications, individuals interact through their controls instead of their states as in the application of trade crowding in (Cardaliaguet and Lehalle 2018). The law of the control appears in the dynamic of (1) and may give some FBSDE depending on the law of Z_t in the weak approach or the law of Y_t in the Pontryagin approach.

In (Chassagneux, Crisan, and Delarue 2019) and (Angiuli et al. 2019), tree and grid algorithms are proposed and tested in dimension 1. It is worth mentioning that these techniques suffer from the so-called curse of dimensionality and cannot be applied when the dimension describing a player state is high (typically greater than 3 or 4). This is due to the discretization of the state space.

However, new approaches using machine learning are developed since 2017 first to solve non linear PDEs. Two kinds of methods have emerged:

- The first to appear are **global methods** first proposed in (Han, Jentzen, and E 2017) to solve semi linear PDEs. The approach is extended to full non linear equations in (Beck, E, and Jentzen 2017) and the authors show that the methodology can solve some equations in high dimension. In the methods proposed Z_t is represented by a neural network at each date. (Chan-Wai-Nam, Mikael, and Warin 2019) showed that it is more effective to used a single network for all dates and besides proposed an original algorithm to solve some the semi-linear PDEs.
- A second kind of **local methods** is based on some local optimization problems solved at each time step. Some algorithms are first proposed in (Huré, Pham, and Warin 2019) to solve semi-linear PDEs and the methodology is extended to some full non linear PDEs in (Pham and Warin 2019) by combining some ideas proposed in (Beck, Becker, et al. 2019).

Machine learning techniques to solve coupled FBSDEs are investigated by several authors in (Han and Long 2018) and (Ji et al. 2019), and a first method for McKean-Vlasov FBSDEs with delay is studied by (Fouque and Zhang 2019) for a linear quadratic equation. Similar and more general ideas are presented in (Carmona and Laurière 2019) alongside convergence results. The resulting algorithms proposed all rely on the global approach first initiated in (Han, Jentzen, and E 2017).

Our paper aims to extend these methods and to propose new ones for the resolution of McKean-Vlasov FBSDEs in high dimension. We first propose to modify the previously proposed algorithm to stabilize the convergence of the algorithm proposed in (Ji et al. 2019). Our modification permits to deal with a high variance of the estimators used in the dynamic of X_t and Y_t. Then we propose a second algorithm relaxing the fixed point iteration algorithm used adding a constraint on the distribution in the loss function optimized At last we propose a resolution scheme based on some local resolution as in (Huré, Pham, and Warin 2019).

To simplify the presentation, we consider first order models, it is to say that the dependency of the drift and cost function with respect to the laws $\mathcal{L}(X_t), \mathcal{L}(Y_t), \mathcal{L}(Z_t)$ only concerns their mean. We define μ_t as the vector $(\mathbb{E}[X_t], \mathbb{E}[Y_t], \mathbb{E}[Z_t])$. This can directly be generalized to any finite number of probability distribution moments. We provide multidimensional tests to show how these machine learning approaches can overcome the curse of dimensionality on some test cases first coming from a mean field game of controls: we solve the FBSDE derived from the weak approach and the Pontryagin approach. Then we compare all the methods on some general test cases of FBSDE involving some linear or quadratic dependence on the processes X_t, Y_t, Z_t and on their distributions.

The structure of the paper is the following: in sections 2 and 3 we describe the proposed schemes, and in section 4 we provide a numerical study of our methods in dimension 10. We show that our algorithms can solve non linear-quadratic models with small maturities. At last we discuss the opportunity to combine Neural networks with some algorithm developed in (Chassagneux, Crisan, and Delarue 2019) and extensively tested in (Angiuli et al. 2019). This tree-based-algorithm is used to permit to solve the problem with higher maturity as it is known that fixed point iteration method can only be applied on small intervals.

2 Some global solvers

In this section we propose three **global algorithms** based in the approach in (Han, Jentzen, and E 2017).

2.1 Algorithm principle

As initialization, we set $\mu_t = (x_0, 0, 0)\ \forall t \in [0, T]$.

We propose a generalized and refined version of the Algorithm 2 from (Carmona and Laurière 2019). We recall that a similar technique with additional networks is used in (Fouque and Zhang 2019) for delayed McKean-Vlasov equations but is tested only on a one dimensional linear quadratic example. Our methods also take advantage of different expectation computation methods, introduced in section 2.2. We present in section 4 several tests in dimension 10 where the laws of X, Y, Z are involved.

We consider the Euler-Maruyama discretized FBSDE system (1) on a regular time grid $t_k = \frac{kT}{N}$ for $k \in [\![0, N]\!]$:

$$\begin{cases} X_{t_{i+1}} &= X_{t_i} + b\left(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}, \mu_{t_i}\right)\ (t_{i+1} - t_i) + \sigma\left(t_i, X_{t_i}\right)(W_{t_{i+1}} - W_{t_i}) \\ Y_{t_{i+1}} &= Y_{t_i} - f\left(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}, \mu_{t_i}\right)\ (t_{i+1} - t_i) + Z_{t_i}(W_{t_{i+1}} - W_{t_i}). \end{cases} \tag{3}$$

This FBSDE is solved by the Merged Deep BSDE method introduced in (Chan-Wai-Nam, Mikael, and Warin 2019). Z_{t_i} is approximated by a feedforward neural network $Z_\theta(t_i, X_{t_i})$ and Y_0 is a variable. With this point of view, the discretized Brownian motion W_t acts as training data in the language of machine learning. The motivation for such an approximation comes from the notion of decoupling field, also used for numerical purposes in (Angiuli et al. 2019) or (Chassagneux, Crisan, and Delarue 2019), which gives the existence of functions u, v such that

$$Y_t = u(t, X_t, \mathcal{L}(X_t)),\ \ Z_t = v(t, X_t, \mathcal{L}(X_t)). \tag{4}$$

For numerical purposes, it is enough to consider Z as a function of the couple (t, X_t). In fact, the law of the solution (and therefore its moments) can be seen as a function of t. That's why we search for a representation

$$Y_t = \widetilde{u}(t, X_t),\ \ Z_t = \widetilde{v}(t, X_t). \tag{5}$$

The forward-backward system is transformed into a forward system and an optimization problem aiming to satisfy the terminal condition of the BSDE through the loss function $\mathbb{E}\left[\left(Y_T - g\left(X_T, \mu_T\right)\right)^2\right]$. The computation graph is drawn in figure 1. To simplify the graph, the expectation computation is not presented in the figure. To simplify notations, $X_i := X_{t_i}$ and similarly for Y and Z.

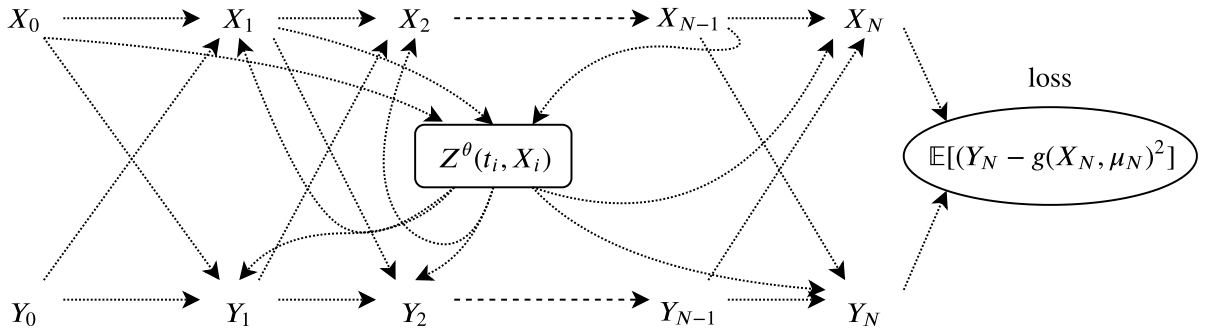


Figure 1: Simplified computation graph of our global solver

The loss is minimized with the Adam gradient descent method (Kingma and Ba 2014). In any case, the goal of our scheme is to learn both the optimal control and the distribution of X_t, Y_t, Z_t. In the following, B is the batch size, N is the number of time steps and M is the number of previous batches expectations to keep in memory. In the following, $\Delta t := t_{i+1} - t_i = \frac{T}{N}$.

We use a feedforward neural network with 3 hidden layers ($d + 10$ neurons in each) with hyperbolic tangent function as activation functions and an output layer with identity as activation. It is worth noticing that because of the coupled structure of the FBSDE, we cannot use batch normalization as far as the distribution of X_i is not stationary over time.

2.2 Estimation of the expectation

A key step for the methods is to estimate the moments of the processes X, Y, Z. It has a significant effect on the algorithms performances. We note θ_m the neural network parameters involved in the estimation and $\mu_i(\theta_m)$ the estimation of μ_{t_i} at iteration m. In the algorithms described below, the approximated processes are considered as functions of the parameters θ of the neural network.
Several methods can be used to approximate the moments of the solution involved in the stochastic McKean-Vlasov dynamics:

- **Direct**: use the empirical mean of the current batch (in a McKean Vlasov formulation) or the last batch (in a fixed point formulation). This approach requires to handle very large batches, typically of the order of B =10,000 sample paths get a reasonable approximation of the laws. This is the approach used by (Fouque and Zhang 2019) and (Carmona and Laurière 2019).

$$\mu_i(\theta_m) = \frac{1}{B}\left(\sum_{j=1}^{B} X_i^j(\theta_m), \sum_{j=1}^{B} Y_i^j(\theta_m), \sum_{j=1}^{B} Z_i^j(\theta_m)\right), \ i = 0, \cdots, N-1. \tag{6}$$

  The **direct solver** leads to algorithm 1.

Algorithm 1 Direct solver

Let y_0 be a variable in $\mathbb{R}^k$, $\mathcal{Z}_\eta(\cdot,\cdot)$ be a neural network with parameter η, defined on $\mathbb{R}^+ \times \mathbb{R}^d$ and valued in $\mathbb{R}^{k\times d}$, so that $\theta_0 = (y_0, \eta)$ is initialized with value (y_0, η_0).
for m from 0 to K **do** ▷ Stochastic gradient iterations
  Set $\forall j \in [\![1, B]\!]$, $X_0^j(\theta_m) = x_0 \in \mathbb{R}^d, Y_0(\theta_m) = \theta_{m,0} \in \mathbb{R}^k$.
  for i from 0 to $N-1$ **do**
    $\mu_i(\theta_m) = \frac{1}{B}\left(\sum_{j=1}^{B} X_i^j(\theta_m), \sum_{j=1}^{B} Y_i^j(\theta_m), \sum_{j=1}^{B} \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right)\right)$
    for j from 1 to B **do**
      Sample ξ_i^j from a d-dimensional standard Gaussian vector.
      $X_{i+1}^j(\theta_m) = X_i^j(\theta_m) + b\left(t_i, X_i^j(\theta_m), Y_i^j(\theta_m), \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right), \mu_i(\theta_m)\right)\Delta t + \sqrt{\Delta t}\,\sigma\left(t_i, X_i^j(\theta_m)\right)\xi_i^j$
      $Y_{i+1}^j(\theta_m) = Y_i^j(\theta_m) - f\left(t_i, X_i^j(\theta_m), Y_i^j(\theta_m), \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right), \mu_i(\theta_m)\right)\Delta t + \sqrt{\Delta t}\,\mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right)\xi_i^j$
    end for
  end for
  $\overline{X_N}(\theta_m) = \frac{1}{B}\sum_{j=1}^{B} X_N^j(\theta_m)$,
  $J(\theta_m) = \frac{1}{B}\sum_{j=1}^{B}\left(Y_N^j(\theta_m) - g\left(X_N^j(\theta_m), \overline{X_N}(\theta_m)\right)\right)^2$
  Calculate $\nabla J(\theta_m)$ by back-propagation.
  Update $\theta_{m+1} = \theta_m - \rho_m \nabla J(\theta_m)$.
end for

- **Dynamic**: a method which dynamically updates the estimation on $(M+1)B$ samples. The

expectations from the last M batches are kept in memory in an array

$$(\zeta_{i,r})\ \begin{matrix} i=0,\dots,N-1, \\ r=0,\dots,M-1 \end{matrix} \in \left(\mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^{k\times d}\right)^{N\times M},$$

initialized with values $(x_0, 0, 0)^{N\times M}$.
At iteration $m-1$, $\nu_i^{(m-1)}$ is defined as the empirical mean on these previous sample paths. On a new batch, the expectation is computed by averaging the previous estimation $\nu_i^{(m-1)}$ and the current batch empirical mean by the following algorithm used for $i = 0, \cdots, N-1$:

$$\begin{aligned}
\nu_i^{(m-1)} &= \frac{1}{M}\sum_{r=0}^{M-1} \zeta_{i,r}, \\
\mu_i(\theta_m) &= \frac{M\nu_i^{(m-1)} + \frac{1}{B}\left(\sum_{j=1}^B X_i^j(\theta_m), \sum_{j=1}^B Y_i^j(\theta_m), \sum_{j=1}^B Z_i^j(\theta_m)\right)}{M+1}, \\
\zeta_{i,m\%M} &= \frac{1}{B}\left(\sum_{j=1}^B X_i^j(\theta_m), \sum_{j=1}^B Y_i^j(\theta_m), \sum_{j=1}^B Z_i^j(\theta_m)\right).
\end{aligned} \tag{7}$$

The notation $m\%M$ refers to the remainder of the Euclidian division of m by M. This technique allows to use smaller batches of size 100 or 1000. Thus it is more efficient in terms of convergence speed in comparison with the direct approach. This method can be seen as a dynamic fixed point approach.

Remark 1. *The fixed point approach is known to be convergent theoretically only for small maturities. In practice, the theoretical bound on the maturity found on the simple example given for example in paragraph 3.1 in (Angiuli et al. 2019) is far too pessimistic. We will see that the restriction is not relevant on all our test cases. Other problems such as bifurcation problems will appear first.*

The **dynamic solver** is given in algorithm 2.

Algorithm 2 Dynamic solver

Let y_0 be a variable in $\mathbb{R}^k$, $\mathcal{Z}_\eta(\cdot,\cdot)$ be a neural network with parameter η, defined on $\mathbb{R}^+ \times \mathbb{R}^d$ and valued in $\mathbb{R}^{k\times d}$, so that $\theta_0 = (y_0, \eta)$ is initialized with value (y_0, η_0).
Set $\forall i \in [\![0, N-1]\!],\ \forall r \in [\![0, M-1]\!],\ \zeta_{i,r} = (x_0, 0, 0) \in \mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^{k\times d}$.
for m from 0 to K **do**
    Set $\forall j \in [\![1, B]\!],\ X_0^j(\theta_m) = x_0 \in \mathbb{R}^d, Y_0^j(\theta_m) = \theta_{m,0} \in \mathbb{R}^k$.
    for i from 0 to $N-1$ **do**
        $\mu_i(\theta_m) = \frac{1}{B}\left(\sum_{j=1}^B X_i^j(\theta_m), \sum_{j=1}^B Y_i^j(\theta_m), \sum_{j=1}^B \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right)\right)$
        $\widetilde{\mu_i}(\theta_m) = \frac{\sum_{r=0}^{M-1}\zeta_{i,r} + \mu_i(\theta_m)}{M+1}$
        for j from 1 to B **do**
            Sample ξ_i^j from a d-dimensional standard Gaussian vector.
            $X_{i+1}^j(\theta_m) = X_i^j(\theta_m) + b\left(t_i, X_i^j(\theta_m), Y_i^j(\theta_m), \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right), \widetilde{\mu_i}(\theta_m)\right)\Delta t + \sqrt{\Delta t}\,\sigma\left(t_i, X_i^j(\theta_m)\right)\xi_i^j$
            $Y_{i+1}^j(\theta_m) = Y_i^j(\theta_m) - f\left(t_i, X_i^j(\theta_m), Y_i^j(\theta_m), \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right), \widetilde{\mu_i}(\theta_m)\right)\Delta t + \sqrt{\Delta t}\,\mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right)\xi_i^j$
        end for
        $\zeta_{i,m\%M} = \mu_i(\theta_m)$
    end for
    $\overline{X_N}(\theta_m) = \frac{1}{B}\sum_{j=1}^B X_N^j(\theta_m)$,
    $J(\theta_m) = \frac{1}{B}\sum_{j=1}^B \left(Y_N^j(\theta_m) - g\left(X_N^j(\theta_m), \overline{X_N}(\theta_m)\right)\right)^2$
    Calculate $\nabla J(\theta_m)$ by back-propagation.
    Update $\theta_{m+1} = \theta_m - \rho_m \nabla J(\theta_m)$.
end for

- **Expectation**: estimate μ_t by a neural network Ψ_κ with input t and parameters κ. The loss is modified in order to force the estimation to remain close to the empirical mean of each batch at every time.

$$\mu_i(\theta_m) = \Psi_\kappa(t_i),\ i = 0, \cdots, N \tag{8}$$

In this case, a term $\mathbb{E}\left[\lambda \sum_{i=0}^{N-1}\left(\Psi_\kappa(t_i) - \frac{1}{B}\left(\sum_{j=1}^B X_i^j, \sum_{j=1}^B Y_i^j, \sum_{j=1}^B Z_i^j\right)\right)^2\right]$ is added to the loss function. We will see that in practise this method is quite involved to use because the performances heavily depend upon the choice of the parameter λ. This approach provides a relaxation of the fixed point method. The **expectation solver** is described in algorithm 3.

Algorithm 3 Expectation solver

Let y_0 be a variable in $\mathbb{R}^k$, $\mathcal{Z}_\eta(\cdot,\cdot)$ defined on $\mathbb{R}^+ \times \mathbb{R}^d$, $\Psi_\kappa(\cdot)$ defined on $\mathbb{R}^+$ be neural networks with parameters η, κ, taking values respectively in $\mathbb{R}^{k\times d}$ and $\mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^{k\times d}$, so that $\theta_0 = (y_0, \eta, \kappa)$ is initialized with value (y_0, η_0, κ_0).
for m from 0 to K **do**
  Set $\forall j \in [\![1, B]\!]$, $X_0^j(\theta_m) = x_0 \in \mathbb{R}^d, Y_0^j(\theta_m) = \theta_{m,0} \in \mathbb{R}^k$.
  for i from 0 to $N-1$ **do**
    $\mu_i(\theta_m) = \frac{1}{B}\left(\sum_{j=1}^B X_i^j(\theta_m), \sum_{j=1}^B Y_i^j(\theta_m), \sum_{j=1}^B \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right)\right)$
    for j from 1 to B **do**
      Sample ξ_i^j from a d-dimensional Gaussian vector.
      $X_{i+1}^j(\theta_m) = X_i^j(\theta_m) + b\left(t_i, X_i^j(\theta_m), Y_i^j(\theta_m), \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right), \Psi_{\theta_{m,2}}(t_i)\right)\Delta t + \sqrt{\Delta t}\,\sigma\left(t_i, X_i^j(\theta_m)\right)\xi_i^j$
      $Y_{i+1}^j(\theta_m) = Y_i^j(\theta_m) - f\left(t_i, X_i^j(\theta_m), Y_i^j(\theta_m), \mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right), \Psi_{\theta_{m,2}}(t_i)\right)\Delta t + \sqrt{\Delta t}\,\mathcal{Z}_{\theta_{m,1}}\left(t_i, X_i^j(\theta_m)\right)\xi_i^j$
    end for
  end for
  $\overline{X_N}(\theta_m) = \frac{1}{B}\sum_{j=1}^B X_N^j(\theta_m)$,
  $J(\theta_m) = \frac{1}{B}\sum_{j=1}^B \left(Y_N^j(\theta_m) - g\left(X_N^j(\theta_m), \overline{X_N}(\theta_m)\right)\right)^2 + \lambda \sum_{i=0}^{N-1}\left(\mu_i(\theta_m) - \Psi_{\theta_{m,2}}(t_i)\right)^2$
  Calculate $\nabla J(\theta_m)$ by back-propagation.
  Update $\theta_{m+1} = \theta_m - \rho_m \nabla J(\theta_m)$.
end for

We will compare the performances of these techniques on several examples in section 4.

3 A local solver

We also propose a local method inspired by the Deep Backward Dynamic Programming introduced by (Huré, Pham, and Warin 2019) and (Pham and Warin 2019). It considers local minimization problems between contiguous time steps. In this case there are as many networks as time steps. We replace a global optimization setting by a set of smaller problems.

In this method for $i \in [\![0, N-1]\!]$, Z_i and Y_i are approximated by a neural network $(Z^i_{\theta_m^i}(\cdot), Y^i_{\theta_m^i}(\cdot))$. At iteration m we simulate X_i with the previously computed parameters θ_m^i. The X_i's dynamic being frozen with parameters θ_m^i, we solve backward problems to find the θ_{m+1}^i:

- First Y^N is set to the terminal condition $g(X_N, \mu_N)$.
- For i from $N-1$ to 0 we solve successively the local backward problems

$$\min_{\theta_m^i} \mathbb{E}\left[\left(Y^{i+1}_{\theta^{i+1}_{m+1}}(X_{i+1}) - Y^i_{\theta_m^i}(X_i) + f\left(t_i, X_i, Y^i_{\theta_m^i}(X_i), Z^i_{\theta_m^i}(X_i), \mu_{t_i}\right)\Delta t - Z^i_{\theta_m^i}(X_i)\Delta W_{t_i}\right)^2\right]. \tag{9}$$

We can then update the θ value as θ_{m+1}^i. In the version of the **local solver** given in algorithm 4, we use the dynamic update of the expectations introduced previously in the dynamic solver of section 2. In this algorithm H stands for the number of gradient steps to perform at each step of the algorithm and R is the number of samples for the laws estimation.

Algorithm 4 Local solver

Let $(\mathcal{Y}^i_{\eta^i}(\cdot), \mathcal{Z}^i_{\eta^i}(\cdot))$ be some neural networks defined on $\mathbb{R}^d$ with values in $\mathbb{R}^k \times \mathbb{R}^{k\times d}$ for $i = 0, \cdots, N-1$ and initialized with values η^i_0. We note $\theta_0 = (\eta^0_0, \cdots, \eta^N_0)$.
Set $\forall i \in [\![0, N]\!],\ \forall r \in [\![0, M-1]\!],\ \zeta_{i,r} = (x_0, 0, 0) \in \mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^{k\times d}$.
for m from 0 to K **do**
  Sample ξ^j_i from a d-dimensional standard Gaussian vector, $i = 0, \cdots, N$, $j = 1, \cdots, R$.
  Set $\forall j \in [\![1, R]\!],\ X^j_0(\theta_m) = x_0 \in \mathbb{R}^d$.
  for i from 0 to N **do** ▷ Forward estimation of the laws
    $\mu_i(\theta_m) = \frac{1}{R}\left(\sum_{j=1}^R X^j_i(\theta_m), \sum_{j=1}^R \mathcal{Y}^i_{\theta_{m,i}}\left(X^j_i(\theta_m)\right), \sum_{j=1}^R \mathcal{Z}^i_{\theta_{m,i}}\left(X^j_i(\theta_m)\right)\right)$
    $V_i(\theta_m) = \frac{1}{R}\sum_{j=1}^R \left(X^j_i(\theta_m)\right)^2 - \frac{1}{R^2}\left(\sum_{j=1}^R X^j_i(\theta_m)\right)^2$
    $\widetilde{\mu_i}(\theta_m) = \frac{\sum_{r=0}^{M-1} \zeta_{i,r} + \mu_i(\theta_m)}{M+1}$
    $\zeta_{i,m\%M} = \mu_i(\theta_m)$
    for j from 1 to R **do**
      $X^j_{i+1}(\theta_m) = X^j_i(\theta_m) + b\left(t_i, X^j_i(\theta_m), \mathcal{Y}^i_{\theta_{m,i}}\left(X^j_i(\theta_m)\right), \mathcal{Z}^i_{\theta_{m,i}}\left(X^j_i(\theta_m)\right), \widetilde{\mu_i}(\theta_m)\right)\Delta t + \sqrt{\Delta t}\,\sigma\left(t_i, X^j_i(\theta_m)\right)\xi^j_i$
    end for
  end for
  for i from $N-1$ to 0 **do** ▷ Backward resolution
    $\hat{\theta}_0 = \theta_{m,i}$
    for h from 0 to $H-1$ **do** ▷ Gradient descent with simulated data for X
      for j from 1 to B **do**
        Sample Ξ^j_i, Θ^j_i from d-dimensional standard Gaussian vectors.
        $x^j_i = \mu_i(\theta_m)_0 + \sqrt{V_i(\theta_m)}\,\Theta^j_i$
        $x^j_{i+1} = x^j_i + b\left(t_i, x^j_i, \mathcal{Y}^i_{\hat{\theta}_h}\left(x^j_i\right), \mathcal{Z}^i_{\hat{\theta}_h}\left(x^j_i\right), \widetilde{\mu_i}(\theta_m)\right)\Delta t + \sqrt{\Delta t}\,\sigma\left(t_i, x^j_i\right)\Xi^j_i$
        if $i = N-1$ **then**
          $Y^j_{i+1} = g\left(x^j_N, \widetilde{\mu_N}(\theta_m)_0\right)$
        else
          $Y^j_{i+1} = \mathcal{Y}^{i+1}_{\hat{\theta}_{m,i+1}}\left(x^j_{i+1}\right)$
        end if
      end for
      $J^i(\hat{\theta}_h) = \frac{1}{B}\sum_{j=1}^B \left(f\left(t_i, x^j_i, \mathcal{Y}^i_{\hat{\theta}_h}\left(x^j_i\right), \mathcal{Z}^i_{\hat{\theta}_h}\left(x^j_i\right), \widetilde{\mu_i}(\theta_m)\right)\Delta t + Y^j_{i+1} - \mathcal{Y}^i_{\hat{\theta}_h}\left(x^j_i\right) - \sqrt{\Delta t}\,\mathcal{Z}^i_{\hat{\theta}_h}\left(x^j_i\right)\Xi^j_i\right)^2$
      Calculate $\nabla J^i(\hat{\theta}_h)$ by back-propagation.
      Update $\hat{\theta}_{h+1} = \hat{\theta}_h - \rho_h \nabla J^i(\hat{\theta}_h)$.
    end for
    $\theta_{m+1,i} = \hat{\theta}_H$
  end for
end for

Remark 2. *Because we have to learn the dynamic of the forward process, the use of a backward resolution is not as obvious as in (Huré, Pham, Bachouch, et al. 2018). We have to alternate forward dynamic estimations and backward resolutions.*

4 Numerical results

The algorithms are implemented in Python with the Tensorflow library (Abadi et al. 2016). Each numerical experiment is conducted using a node composed of 2 Intel® Xeon® Gold 5122 Processors, 192 Go of RAM, and 2 GPU nVidia® Tesla® V100 16Go. The multi-GPU parallelization on the global solver is conducted using the Horovod library (Sergeev and Del Balso 2018). The methods we test are:

- **Direct**: algorithm 1 at page 5. Batch size $B = 10000$.
- **Dynamic**: algorithm 2 at page 7. Batch size $B = 200$ and $M = 100$.
- **Expectation**: algorithm 3 at page 8. Batch size $B = 2000$.
- **Local**: algorithm 4 at page 9. Batch size $B = 300$ (Weak), $B = 100$ (Pontryagin) and $M = 20$.

If the algorithm is applied to equations coming from the Pontryagin or the Weak approach, it is specified in its name.

4.1 Linear price impact model

We use a linear-quadratic mean field game of controls model studied in (Angiuli et al. 2019) and (Carmona and Delarue 2018a) for comparison. This model is useful for numerical tests as far as the analytic solution is known. The MFG of controls model for the representative player is given by:

$$\begin{aligned} \min_{\alpha\in\mathbb{A}} \quad & \mathbb{E}\left[\int_0^T \left(\frac{c_\alpha}{2}\left\|\alpha_t\right\|^2 + \frac{c_X}{2}\left\|X_t\right\|^2 - \gamma X_t\cdot\mu_t\right)\,\mathrm{d}t + \frac{c_g}{2}\left\|X_T\right\|^2\right]. \\ \text{subject to} \quad & X_t = x_0 + \int_0^t \alpha_s\,\mathrm{d}s + \sigma\,W_t \end{aligned} \tag{10}$$

and the fixed point $\mathbb{E}[\alpha_t] = \mu_t$. In this case, the mean field interaction is exerted through the law of the control process.
The Pontryagin optimality principle gives the system:

$$\begin{cases} \mathrm{d}X_t &= -\frac{1}{c_\alpha}Y_t\,\mathrm{d}t + \sigma\,\mathrm{d}W_t \\ X_0 &= x_0 \\ \mathrm{d}Y_t &= -(c_X X_t + \frac{\gamma}{c_\alpha}\mathbb{E}[Y_t])\,\mathrm{d}t + Z_t\,\mathrm{d}W_t \\ Y_T &= c_g X_T. \end{cases} \tag{11}$$

In this case, the output Z of the neural network is a matrix of size $d\times d$ and Y is a vector of size d. The weak representation of the value function gives:

$$\begin{cases} \mathrm{d}X_t &= -\frac{1}{c_\alpha}\sigma^{-1}Z_t\,\mathrm{d}t + \sigma\,\mathrm{d}W_t \\ X_0 &= x_0 \\ \mathrm{d}Y_t &= -\left(\frac{c_X}{2}\left\|X_t\right\|^2 + \frac{\gamma}{c_\alpha}X_t\cdot\sigma^{-1}\mathbb{E}[Z_t] + \frac{1}{2c_\alpha}\left\|\sigma^{-1}Z_t\right\|^2\right)\,\mathrm{d}t + Z_t\,\mathrm{d}W_t \\ Y_T &= \frac{c_g}{2}\left\|X_T\right\|^2. \end{cases} \tag{12}$$

In this case, the output Z of the neural network is a vector of size d and Y is a scalar. Therefore we may be able work in higher dimensions.

Remark 3. *With LQ models, the dynamics of Y is linear in the Pontryagin approach and quadratic in the Weak approach. Thus the potentially high dimension of one method is counterbalanced by the complex dynamics of the other technique.*

For our numerical experiments we take $c_X = 2, x_0 = 1, \sigma = 0.7, \gamma = 2, c_\alpha = 2/3, c_g = 0.3$. If not stated otherwise, the simulations are conducted with $T = 1, d = 10, \Delta t = 0.01$.

Method \ T	0.25	0.75	1.0	1.5
Reference	**0.7709**	**0.1978**	**0.0811**	**0.0125**
Pontryagin	0.763 (1.3e-03)	0.187 (2.5e-03)	0.075 (2.7e-03)	0.012 (5.0e-03)
Dyn. Pontryagin	0.762 (2.3e-03)	0.189 (4.0e-03)	0.078 (5.5e-03)	0.013 (6.7e-03)
Exp. Pontryagin	0.765 (1.3e-03)	0.198 (6.7e-03)	0.067 (2.1e-01)	0.496 (1.0e-01)
Weak	0.778 (2.0e-03)	0.200 (1.4e-02)	0.092 (2.9e-02)	0.025 (2.0e-02)
Dyn. Weak	0.775 (4.4e-03)	0.212 (2.0e-02)	0.083 (4.1e-02)	0.016 (5.6e-02)
Exp. Weak	NC	NC	NC	NC
Pontryagin Loc.	0.767 (4.3e-04)	0.192 (8.8e-04)	0.077 (8.8e-04)	0.011 (1.3e-03)
Weak Loc.	0.786 (1.3e-03)	0.308 (1.2e-02)	0.054 (3.7e-02)	0.075 (5.2e-02)

Figure 2: Mean of $\mathbb{E}[X_T]$ over the 10 dimensions (and standard deviation) for several maturities T (2000 iterations for global methods, 6000 iterations for local methods) on the price impact model.

Pontryagin	**1877 s.**
Dyn. Pontryagin	**1336 s.**
Exp. Pontryagin	**1562 s.**
Weak	**2205s.**
Dyn. Weak	**1605 s.**
Exp. Weak	**1670 s.**
Pontryagin Loc.	**3582 s.**
Weak Loc.	**3948 s.**

Figure 3: Duration times of the methods (2000 iterations for global methods, 6000 iterations for local methods) on the price impact model.

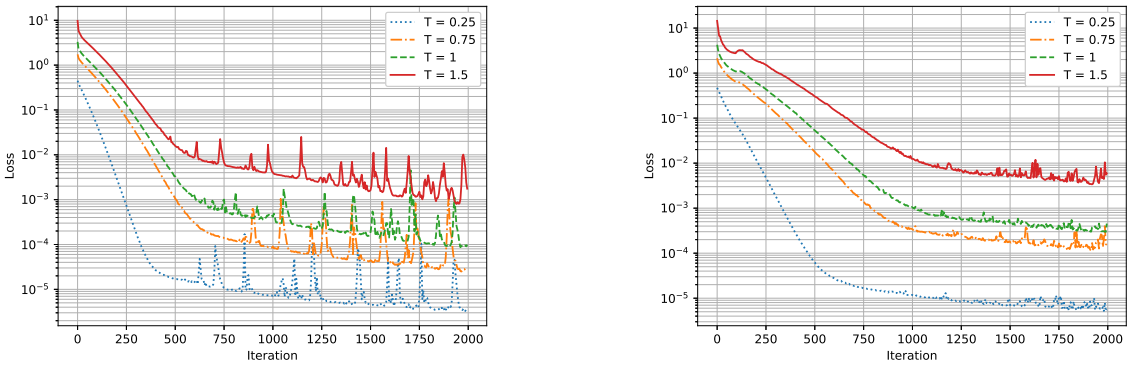


Figure 4: Learning curves for direct (left) and dynamic (right) Pontryagin method on the price impact model. The loss is the L^2 error between Y_T and the terminal condition of the backward equation.

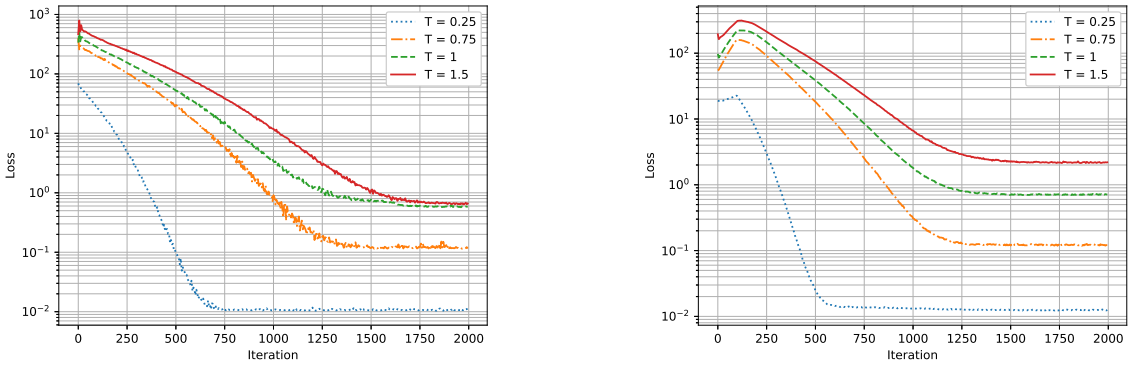


Figure 5: Learning curves for direct (left) and dynamic (right) Weak method on the price impact model. The loss is the L^2 error between Y_T and the terminal condition of the backward equation.

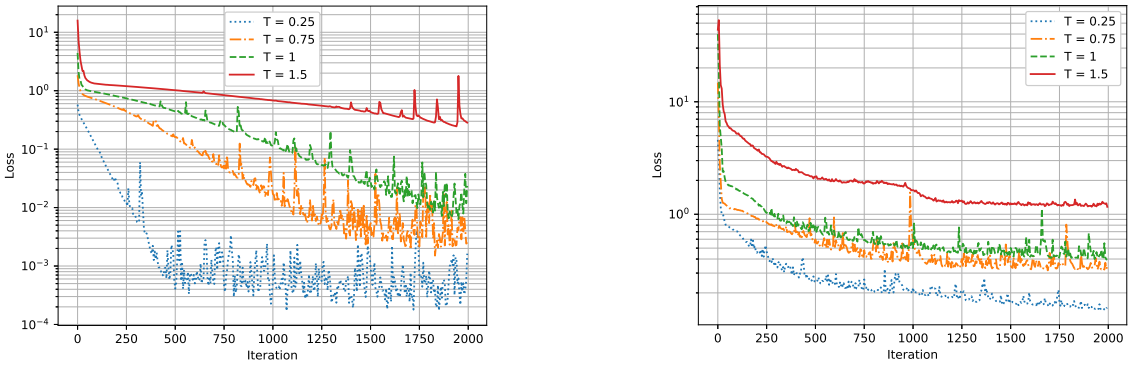


Figure 6: Learning curves for expectation Pontryagin (left) and expectation Weak (right) method on the price impact model. The loss is the L^2 error between Y_T and the terminal condition of the backward equation.

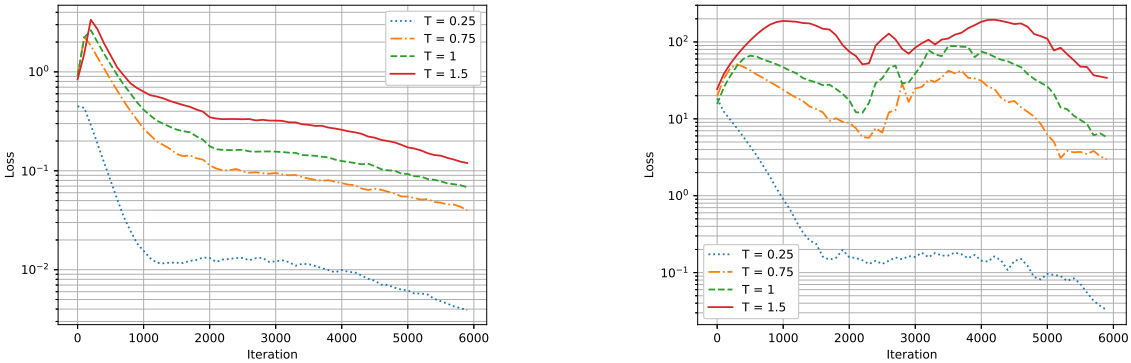


Figure 7: Learning curves for Local Pontryagin (left) and Local Weak (right) method on the price impact model. The loss is the sum of the local L^2 errors between the dynamics of Y and the Euler discretization for all time steps.

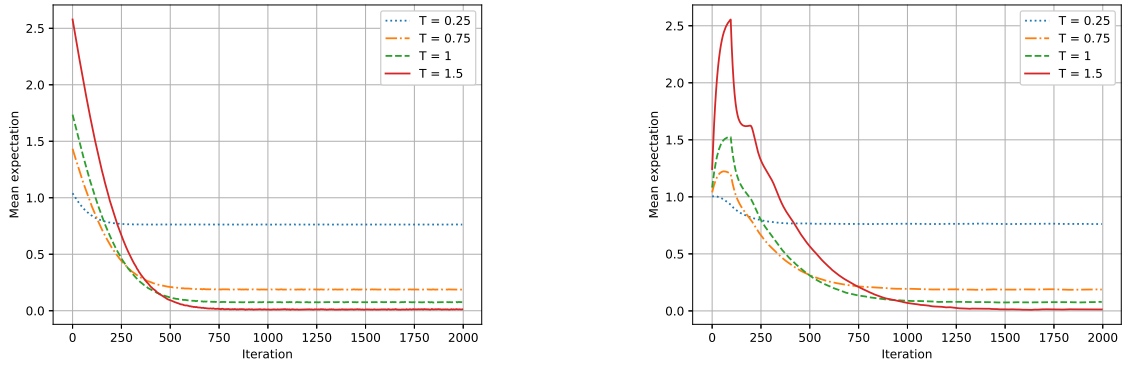


Figure 8: $\mathbb{E}[X_T]$ for direct (left) and dynamic (right) Pontryagin method on the price impact model.

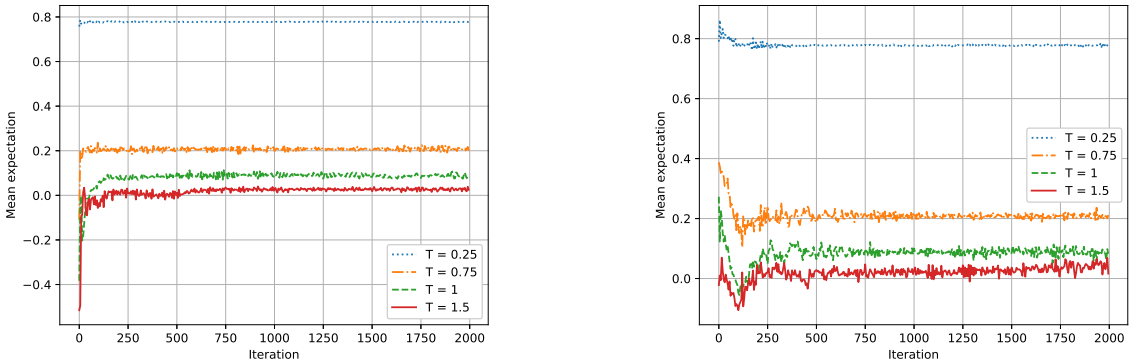


Figure 9: $\mathbb{E}[X_T]$ for direct (left) and dynamic (right) Weak method on the price impact model.

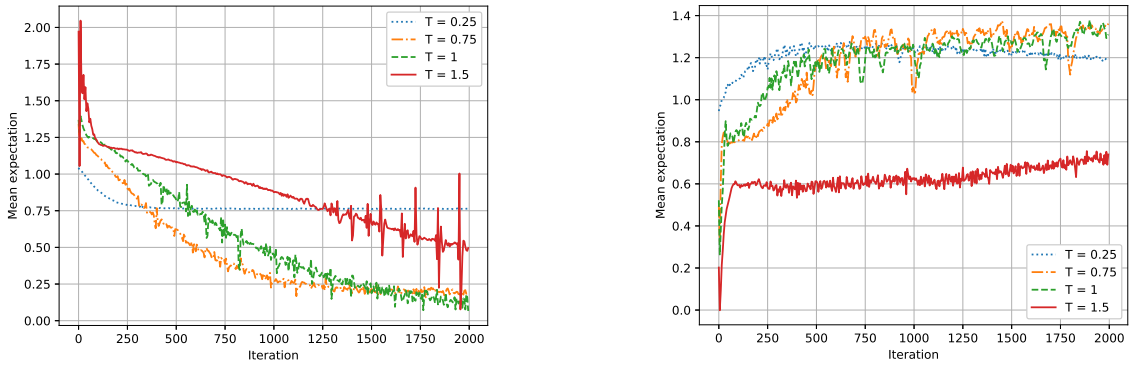
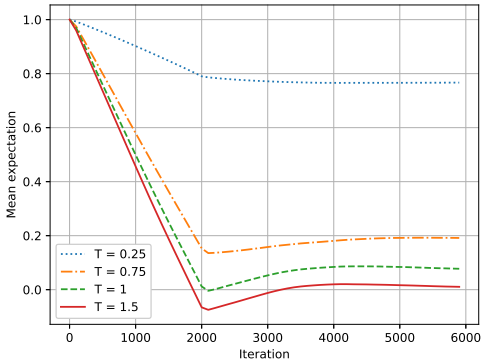


Figure 10: $\mathbb{E}[X_T]$ for expectation Pontryagin (left) and expectation Weak (right) method on the price impact model.



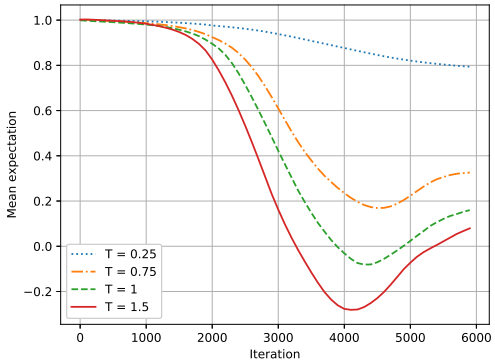
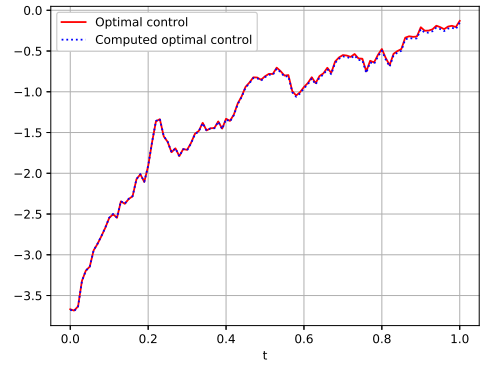


Figure 11: $\mathbb{E}[X_T]$ for Local Pontryagin (left) and Local Weak (right) method on the price impact model.



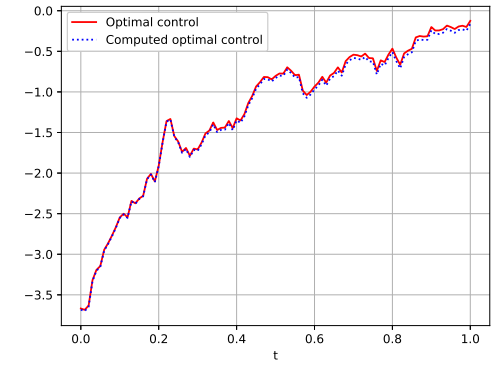
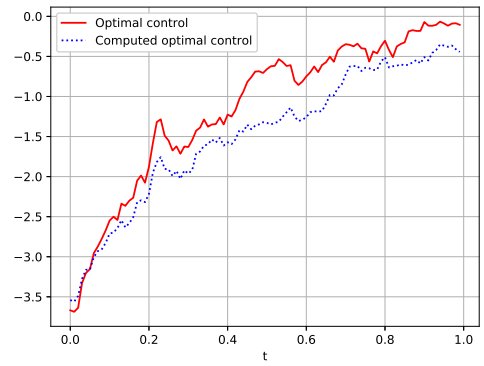


Figure 12: First coordinate of the optimal control evaluated on a sample path for direct (left) and dynamic (right) Pontryagin method after 2000 iterations on the price impact model.



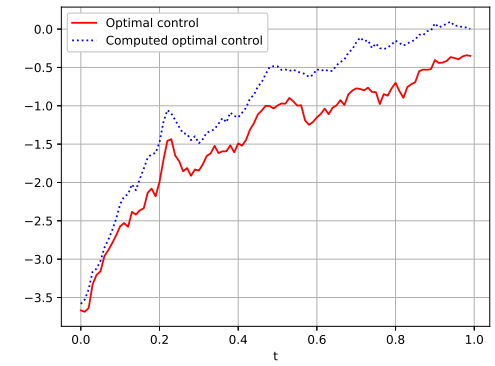


Figure 13: First coordinate of the optimal control evaluated on a sample path for direct (left) and dynamic (right) Weak method after 2000 iterations on the price impact model.

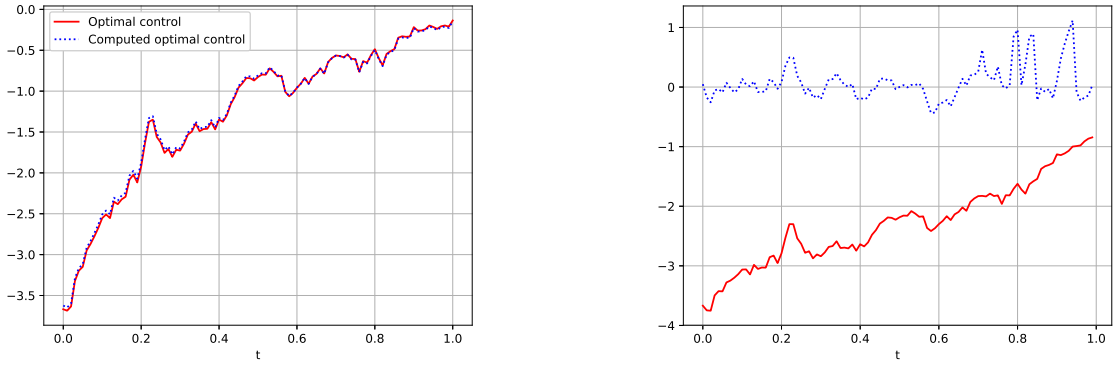


Figure 14: First coordinate of the optimal control evaluated on a sample path for expectation Pontryagin (left) and expectation Weak (right) method after 2000 iterations on the price impact model.

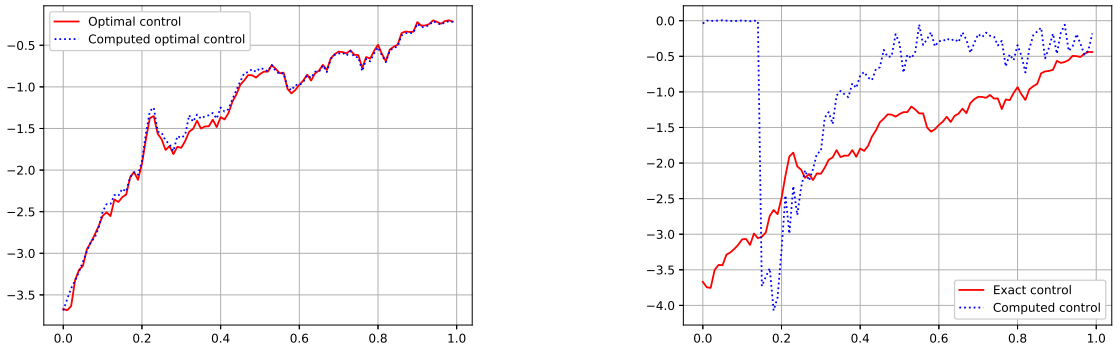


Figure 15: First coordinate of the optimal control evaluated on a sample path for Local Pontryagin (left) and Local Weak (right) Local Weak method after 6000 iterations on the price impact model.

All methods except Expectation Weak converge to the exact solution for small maturities. As we increase the maturity, the local weak solver does not converge anymore to the right solution.
For the local method it may be due to the lack of a contraction for the fixed point problem when the maturity is too high. In fact, the algorithm converges to the true solution only for small maturities. We won't test the expectation methods on the other test cases as far as they are less efficient than the other methods.

We cannot hope for more iterations to help the convergence in the Weak method as far as the loss in the learning curves of figure 5 reaches a plateau. The algorithms solving the system coming from the Pontryagin principle perform better than the others. The dynamic estimation of the expectation allows to gain training speed and to stabilize the loss.

4.2 Beyond the LQ case

In this Section we design non Linear Quadratic models in order to test the limitations of our methods. We construct general MKV FBSDES with explicit solutions following a log-normal distribution. Let X_t^i be defined by

$$\mathrm{d}X_t^i = a^i X_t^i \ \mathrm{d}t + \sigma_t^i X_t^i \ \mathrm{d}W_t^i, \tag{13}$$
$$X_0^i = \xi^i. \tag{14}$$

We obtain explicitly

$$\begin{aligned} X_t^i &= \xi^i e^{(a^i - \frac{(\sigma^i)^2}{2})t + \sigma^i W_t^i}, \\ g_t^i &:= \mathbb{E}[X_t^i] = \xi^i e^{a^i t}, \\ k_t^i &:= \mathbb{E}[(X_t^i)^2] = \xi^i e^{(2a^i + (\sigma^i)^2)t}. \end{aligned}$$

We choose $g : (t,x) \mapsto e^{\alpha t} \log\left(\prod_{i=1}^n x^i\right)$ and the following dynamic for Y_t

$$Y_t = e^{\alpha t} \log\left(\prod_i X_t^i\right) = e^{\alpha t} \sum_i \left[\log(\xi^i) + (a^i - \frac{(\sigma^i)^2}{2})t + \sigma^i W_t^i\right],$$

such that

$$\begin{aligned} c_t &:= \mathbb{E}[Y_t] = e^{\alpha t} \sum_i \left[\log(\xi^i) + \left(a^i - \frac{(\sigma^i)^2}{2}\right) t\right], \\ d_t &:= \mathbb{E}[Y_t^2] = e^{2\alpha t} \left(\left[\sum_i \left(\log(\xi^i) + a^i - \frac{(\sigma^i)^2}{2}\right) t\right]^2 + \sum_i (\sigma^i)^2 t\right). \end{aligned}$$

As we want $Y_t = u(t, X_t)$, we have $Z_t^i = \sigma_t^i X_t^i \partial_x u(t, X_t)$ following:

$$\begin{aligned} Z_t^i &= \sigma_t^i e^{\alpha t} \\ e_t^i &:= \mathbb{E}[Z_t^i] = \sigma_t^i e^{\alpha t}, \\ f_t^i &:= \mathbb{E}[(Z_t^i)^2] = (\sigma_t^i)^2 e^{2\alpha t}. \end{aligned}$$

Introducing

$$\begin{aligned} \phi(t,x) &:= \partial_t u + \sum_i a_i x_i \ \partial_{x_i} u + \sum_i \frac{(\sigma^i x_i)^2}{2} \ \partial^2_{x_i^2} u \\ &= e^{\alpha t} \left(\alpha \log\left(\prod_i x^i\right) + \sum_i \left(a^i - \frac{(\sigma^i)^2}{2}\right)\right), \end{aligned}$$

$u(t, X_t)$ solves the PDE

$$\partial_t u + \sum_i a_i x^i \ \partial_{x_i} u + \sum_i \frac{(\sigma^i)^2}{2} \ \partial^2_{x_i^2} u - \phi(t,x) = 0.$$

This semilinear PDE is related to the BSDE associated with the driver $f(t,x) = -\phi(t,x)$ for forward dynamics (13).
Using some chosen $\mathbb{R}^d$ valued functions ψ^i and $\mathbb{R}^k$ valued functions κ, we express all dynamics in a McKean-Vlasov setting:

$$\begin{cases} \mathrm{d}X_t^i & = (a^i X_t^i + \psi^i(Y_t, Z_t^i, \mathbb{E}[X_t^i], \mathbb{E}[(X_t^i)^2], \mathbb{E}[Y_t], \mathbb{E}[Y_t^2], \mathbb{E}[Z_t^i], \mathbb{E}[(Z_t^i)^2]) \\ & \quad -\psi^i\left(e^{\alpha t} \log\left(\prod_i X_t^i\right), \sigma_t^i e^{\alpha t}, g_t^i, k_t^i, c_t, d_t, e_t^i, f_t^i\right) \ \mathrm{d}t + \sigma_t^i X_t^i \ \mathrm{d}W_t^i \\ X_0^i & = \xi^i \\ \mathrm{d}Y_t & = -f(t, X_t, Y_t, Z_t, \mathbb{E}[X_t], \mathbb{E}[X_t^2], \mathbb{E}[Y_t], \mathbb{E}[Y_t^2], \mathbb{E}[Z_t], \mathbb{E}[Z_t^2]) \ \mathrm{d}t + Z_t \ \mathrm{d}W_t \\ Y_T & = e^{\alpha T} \log\left(\prod_i X_T^i\right) \end{cases} \tag{15}$$

with

$$
\begin{aligned}
&f(t, X_t, Y_t, Z_t, x_1, x_2, y_1, y_2, z_1, z_2) \\
&= -\phi(t,x) + \kappa(Y_t, Z_t, x_1, x_2, y_1, y_2, z_1, z_2) - \kappa\left(e^{\alpha t}\log\left(\prod_i X_t^i\right), \sigma_t^i e^{\alpha t}, g_t^i, k_t^i, c_t, d_t, e_t^i, f_t^i\right).
\end{aligned}
$$

and $f : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{1\times d} \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{1\times d} \times \mathbb{R}^{1\times d} \mapsto \mathbb{R}$.
We consider two models of this kind for numerical tests.

4.2.1 Linear

We consider a linear McKean-Vlasov FBSDE in Y_t, Z_t and their law dynamics for X_t and Y_t:

$$
\begin{cases}
\mathrm{d}X_t^i &= (a^i X_t^i + b(Y_t + Z_t^i + \mathbb{E}[X_t^i] + \mathbb{E}[Y_t] + \mathbb{E}[Z_t^i]) \\
&\quad -b\left(e^{\alpha t}\log\left(\prod_{i=1}^d X_t^i\right) + \sigma_t^i e^{\alpha t} + g_t^i + c_t + e_t^i\right)\,\mathrm{d}t + \sigma_t^i X_t^i\,\mathrm{d}W_t^i \\
X_0^i &= \xi^i \\
\mathrm{d}Y_t &= \left(\phi(t, X_t) + b(Y_t + \frac{1}{d}\sum_{i=1}^d Z_t^i + \frac{1}{d}\sum_{i=1}^d \mathbb{E}[X_t^i] + \mathbb{E}[Y_t] + \frac{1}{d}\sum_{i=1}^d \mathbb{E}[Z_t^i])\right. \\
&\quad \left. -b\left(e^{\alpha t}\log\left(\prod_{i=1}^d X_t^i\right) + \frac{1}{d}\sum_{i=1}^d \sigma_t^i e^{\alpha t} + \frac{1}{d}\sum_{i=1}^d g_t^i + c_t + \frac{1}{d}\sum_{i=1}^d e_t^i\right)\right)\mathrm{d}t + Z_t\,\mathrm{d}W_t \\
Y_T &= e^{\alpha T}\log\left(\prod_{i=1}^d X_T^i\right).
\end{cases}
\tag{16}
$$

We take $a = b = 0.1,\ \alpha = 0.5, \sigma = 0.4, \xi = 1$.

Method \ T	0.25	0.75	1.0	1.5
Reference	**1.0253**	**1.0779**	**1.1052**	**1.1618**
Global	1.025 (1.8e-03)	1.076 (3.3e-03)	1.095 (3.9e-03)	1.162 (7.2e-03)
Dyn. Global	1.026 (2.0e-03)	1.077 (3.6e-03)	1.105 (2.9e-03)	1.163 (4.8e-03)
Local	1.025 (2.4e-04)	1.096 (6.2e-04)	1.158 (9.6e-04)	1.310 (1.5e-03)

Figure 16: Mean of $\mathbb{E}[X_T]$ over the 10 dimensions (and standard deviation) for several maturities T (2000 iterations for global methods, 6000 iterations for local methods) on the general linear model.

Global	Dynamic Global	Local
2081 s.	**1308 s.**	**5382 s.**

Figure 17: Duration times of the methods (2000 iterations for global methods, 6000 iterations for local methods) on the general linear model.

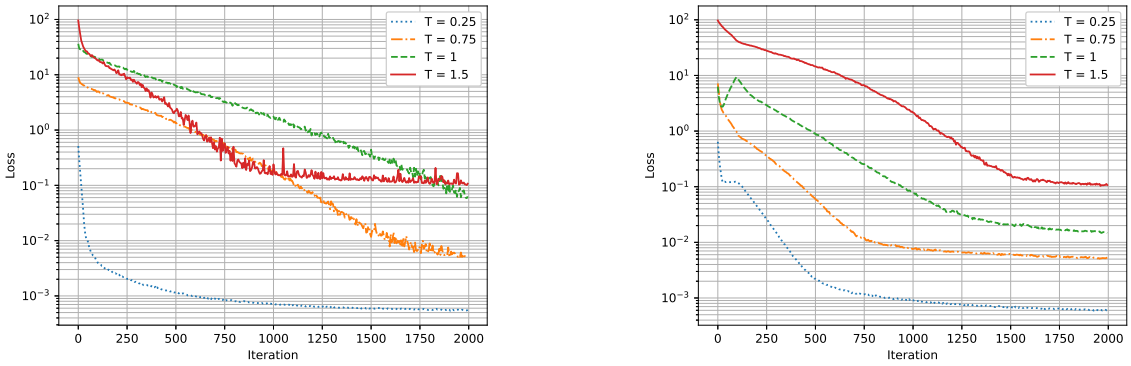


Figure 18: Learning curves for Direct Global (left) and Dynamic Global (right) method on the general linear model. The loss is the L^2 error between Y_T and the terminal condition of the backward equation.

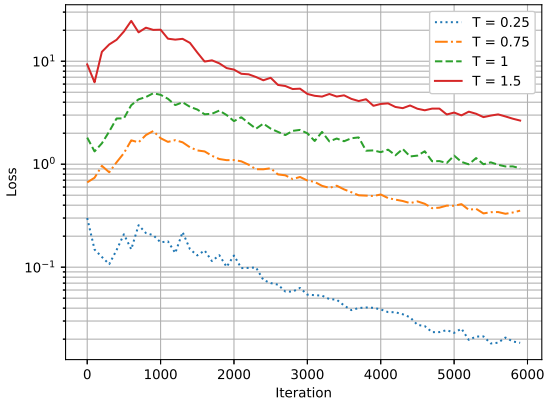


Figure 19: Learning curves for Local method on the general linear model. The loss is the sum of the local L^2 errors between the dynamics of Y and the Euler discretization for all time steps.

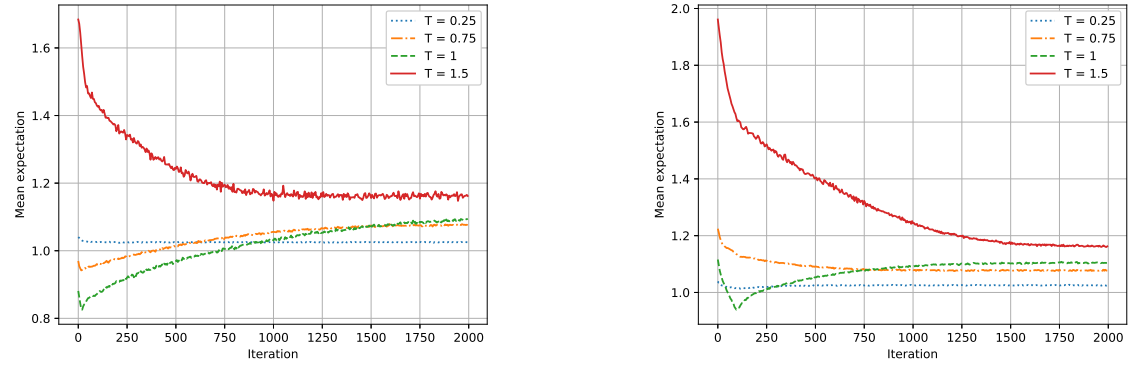


Figure 20: $\mathbb{E}[X_T]$ for Direct Global (left) and Dynamic Global (right) method on the general linear model.

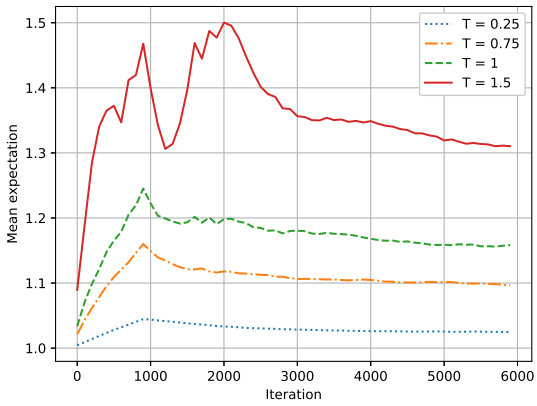


Figure 21: $\mathbb{E}[X_T]$ for Local method on the general linear model.

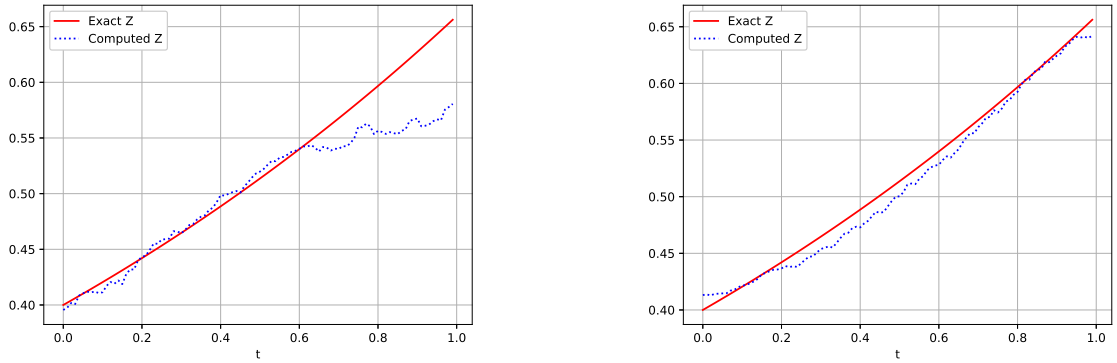


Figure 22: First coordinate of Z_t evaluated on a sample path for Direct Global (left) and Dynamic Global (right) method after 2000 iterations on the general linear model.

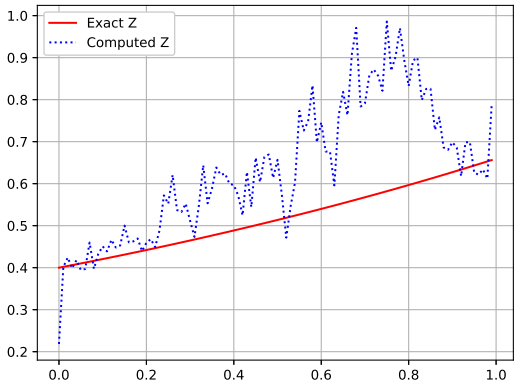


Figure 23: First coordinate of Z_t evaluated on a sample path for Local method after 6000 iterations on the general linear model.

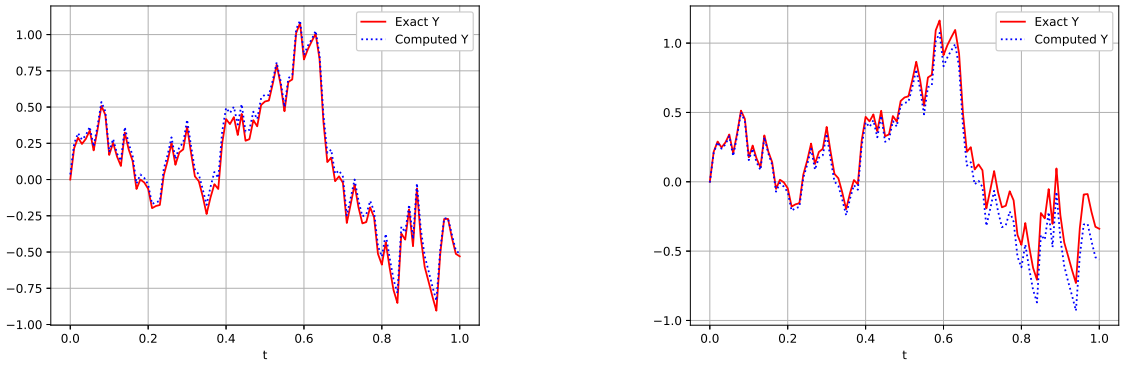


Figure 24: First coordinate of Y_t evaluated on a sample path for Direct Global (left) and Dynamic Global (right) method after 2000 iterations on the general linear model.

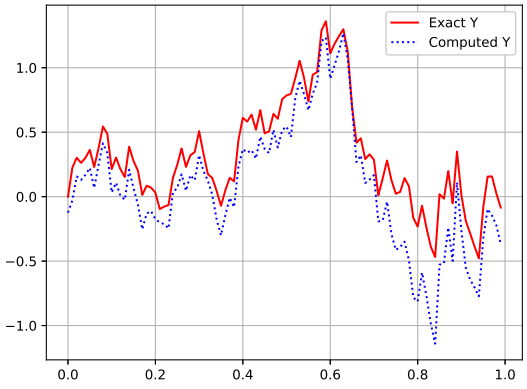


Figure 25: Y_t evaluated on a sample path for Local method after 6000 iterations on the general linear model.

The three algorithms demonstrate good performances on this test case. Both processes Y, Z are well represented by the neural network. However the local method is less precise than the global methods when the maturity grows. More iterations could improve the results given by the local method, as far as the loss hasn't reached a plateau yet in figure 19.

4.2.2 Quadratic

We consider a quadratic McKean-Vlasov FBSDE in Y_t, Z_t and their law dynamics for X_t and Y_t:

$$\begin{cases} \mathrm{d}X_t^i &= (a^i X_t^i + b(Y_t + Z_t^i + \mathbb{E}[X_t^i] + \mathbb{E}[Y_t] + \mathbb{E}[Z_t^i]) \\ &\quad -b\left(e^{\alpha t}\log\left(\prod_{i=1}^d X_t^i\right) + \sigma_t^i e^{\alpha t} + g_t^i + c_t + e_t^i\right) \\ &\quad +c\Bigg(Y_t^2 + (Z_t^i)^2 + \mathbb{E}[(X_t^i)^2] + \mathbb{E}[Y_t^2] + \mathbb{E}[(Z_t^i)^2]) \\ &\quad -c\left(e^{2\alpha t}\log\left(\prod_{i=1}^d X_t^i\right)^2 + (\sigma_t^i)^2 e^{2\alpha t} + (g_t^i)^2 + c_t^2 + (e_t^i)^2\right)\Bigg)\,\mathrm{d}t + \sigma_t^i X_t^i\,\mathrm{d}W_t^i \\ X_0^i &= \xi^i \\ \mathrm{d}Y_t &= \Bigg(\phi(t,X_t) + b(Y_t + \tfrac{1}{d}\textstyle\sum_{i=1}^d Z_t^i + \tfrac{1}{d}\sum_{i=1}^d \mathbb{E}[X_t^i] + \mathbb{E}[Y_t] + \tfrac{1}{d}\sum_{i=1}^d \mathbb{E}[Z_t^i]) \\ &\quad -b\left(e^{\alpha t}\log\left(\prod_{i=1}^d X_t^i\right) + \tfrac{1}{d}\textstyle\sum_{i=1}^d \sigma_t^i e^{\alpha t} + \tfrac{1}{d}\sum_{i=1}^d g_t^i + c_t + \tfrac{1}{d}\sum_{i=1}^d e_t^i\right) \\ &\quad +c(Y_t^2 + \tfrac{1}{d}\textstyle\sum_{i=1}^d (Z_t^i)^2 + \tfrac{1}{d}\sum_{i=1}^d \mathbb{E}[(X_t^i)^2] + \mathbb{E}[Y_t^2] + \tfrac{1}{d}\sum_{i=1}^d \mathbb{E}[(Z_t^i)^2]) \\ &\quad -c\left(e^{2\alpha t}\log\left(\prod_{i=1}^d X_t^i\right)^2 + \tfrac{1}{d}\textstyle\sum_{i=1}^d (\sigma_t^i)^2 e^{2\alpha t} + \tfrac{1}{d}\sum_{i=1}^d (g_t^i)^2 + c_t^2 + \tfrac{1}{d}\sum_{i=1}^d (e_t^i)^2\right)\Bigg)\,\mathrm{d}t + Z_t\,\mathrm{d}W_t \\ Y_T &= e^{\alpha T}\log\left(\prod_{i=1}^d X_T^i\right). \end{cases} \tag{17}$$

We take $a = b = c = 0.1,\ \alpha = 0.5, \sigma = 0.4, \xi = 1$.

Method \ T	0.25	0.75	1.0	1.5
Reference	**1.0253**	**1.0779**	**1.1052**	**1.1618**
Global	1.024 (1.8e-03)	1.065 (4.3e-03)	12.776 (3.3e-02)	DV
Dyn. Global	1.025 (2.1e-03)	1.072 (3.1e-03)	0.961 (7.0e-03)	DV
Local	1.025 (3.3e-04)	-6.763 (7.5e-04)	0.409 (2.7e-04)	DV

Figure 26: Mean of $\mathbb{E}[X_T]$ over the 10 dimensions (and standard deviation) for several maturities T (2000 iterations for global methods, 50 fixed point iterations for local methods) on the general quadratic model.

Global	Dynamic Global	Local
2072 s.	**1309 s.**	**5373 s.**

Figure 27: Duration times of the methods (2000 iterations for global methods, 6000 iterations for local methods) on the general quadratic model.

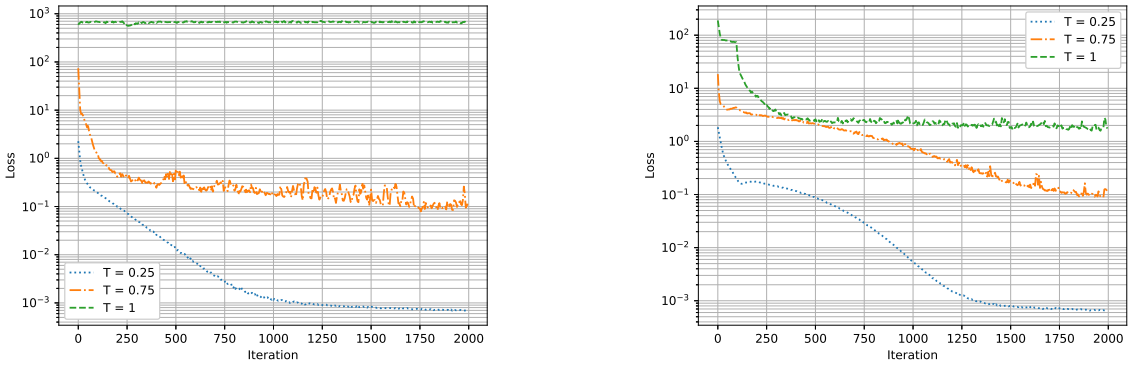


Figure 28: Learning curves for Direct Global (left) and Dynamic Global (right) method on the general linear model. The loss is the L^2 error between Y_T and the terminal condition of the backward equation.

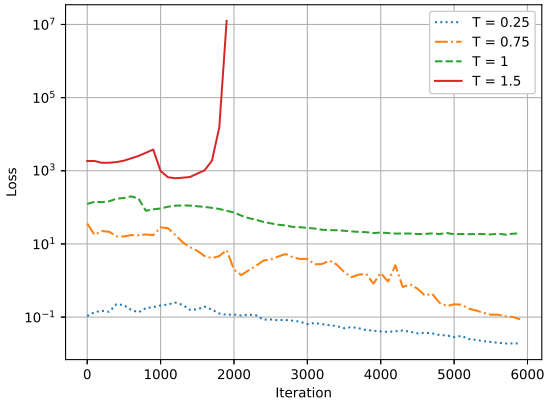


Figure 29: Learning curves for Local method on the general quadratic model. The loss is the sum of the local L^2 errors between the dynamics of Y and the Euler discretization for all time steps.

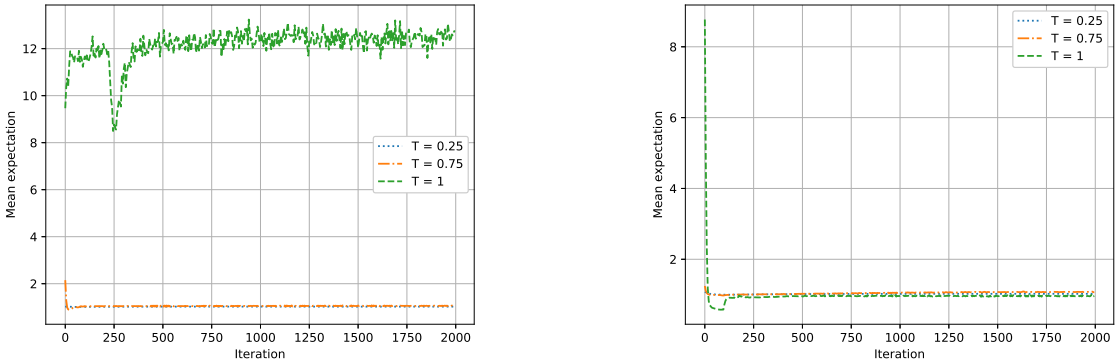


Figure 30: $\mathbb{E}[X_T]$ for Direct Global (left) and Dynamic Global (right) method on the general quadratic model.

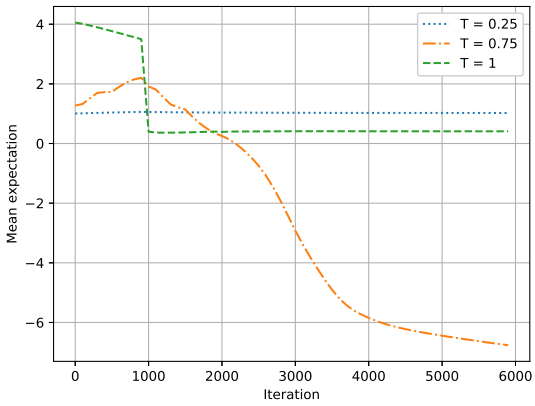


Figure 31: $\mathbb{E}[X_T]$ for Local method on the general quadratic model.

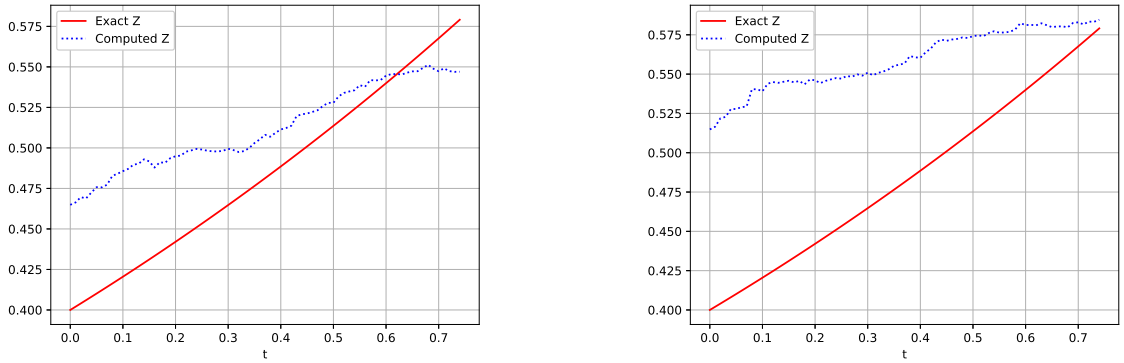


Figure 32: First coordinate of Z_t evaluated on a sample path for Direct Global (left) and Dynamic Global (right) method ($T = 0.75$) after 2000 iterations on the general quadratic model.

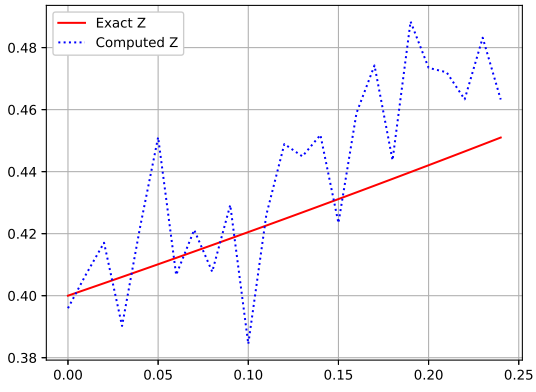


Figure 33: First coordinate of Z_t evaluated on a sample path for Local method ($T = 0.25$) after 6000 iterations on the general quadratic model.

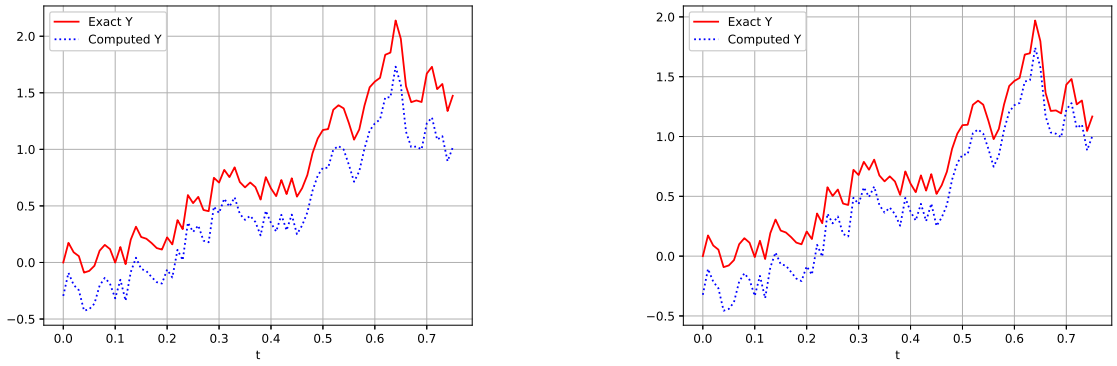


Figure 34: First coordinate of Y_t evaluated on a sample path for Direct Global (left) and Dynamic Global (right) method ($T = 0.75$) after 2000 iterations on the general quadratic model.

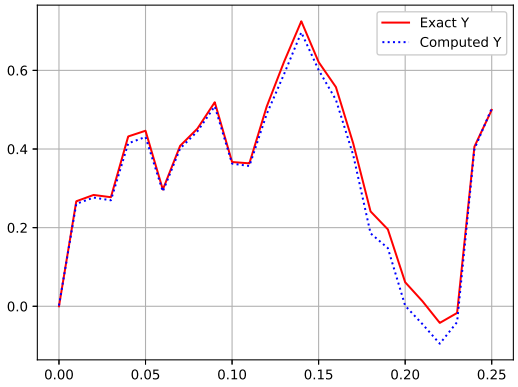


Figure 35: First coordinate of Y_t evaluated on a sample path for Local method ($T = 0.25$) after 6000 iterations on the general quadratic model.

We observe convergence for small maturities and divergence beyond $T = 1$. Note that the dynamic estimation of the expectation prevents the algorithm to explode for $T = 1$, contrarily to the direct method. However, it does not converge to the true solution in this case. Indeed the loss plateaus at the value 2 in figure 28 (right), so the terminal condition of the BSDE is not properly respected. The dynamic method also produces a better approximation of Y (see figure 34). Concerning the local method, we see in figure 31 that the estimated expectations are stable around zero for a few iterations but then become negative. It may be due to the lack of a contraction for the fixed point problem.

5 Some remark on bifurcations

As we increase the coefficient values in the FBSDE, we seem to face some bifurcations. As stated in the papers (Chassagneux, Crisan, and Delarue 2019), (Angiuli et al. 2019) we encounter for example bifurcations as we increase the parameter c_X in the price impact problem. Suspecting that these convergence problems are partly due to the fixed point iteration procedure used in their tree algorithm, they propose a numerical algorithm limiting the use of FBSDE fixed point iteration procedure to small maturities. Their algortihm is used for some tree solvers and we propose to

extend it with neural networks.
The algorithm proposed is based on some level of discretization $\{0 = T_0 < T_1, \dots, T_{L_p}\}$ where L_p stands for the number of levels. On a level $[T_k, T_{k+1}]$, we can divide the resolution period with a time step $\frac{T_{k+1}-T_k}{N_k}$. On a level we can solve the problem

$$(Y_{T_k}(\theta_k, .), \mathcal{L}(X_{T_{k+1}})) = \phi_k(\mathcal{L}(X_{T_k}), Y_{T_{k+1}}(\theta_{k+1}, .)), \tag{18}$$

where ϕ_k is an operator defined for $t \in [T_k, T_{k+1}]$ taking as arguments:

- $\mathcal{L}(X_{T_k})$ the law of an estimation of the process X_t at date T_k: this estimation is calculated by the resolution of the sub problem (18) on $[T_{k-1}, T_k]$ or taking this law equal to δ_{X_0} for $k = 0$,
- $Y_{T_{k+1}}(\theta_{k+1}, .)$ an estimation calculated by neural networks of Y_t at date T_{k+1} obtained by the resolution of the sub problem (18) on $[T_{k+1}, T_{k+2}]$ or by projection of the terminal condition g when $k = L_p$ having an estimation of $\mathcal{L}(X_T)$.

The operator ϕ_k gives back an estimation of

- $Y_{T_k}(\theta_k, .)$ solution of the equations (1) with initial X distribution $\mathcal{L}(X_{T_k})$ and terminal condition $Y_{T_{k+1}}(\theta_{k+1}, .)$ for $Y_{T_{k+1}}$,
- $\mathcal{L}(X_{T_{k+1}})$ the distribution of the forward process at date T_{k+1},

where the resolution has been achieved on each level k by optimization of some neural networks parametrized with θ_k by the solvers proposed in section 2 or 3.

Using the operators $\{\phi_k\}_{k=1,L_p}$ and iterating between levels, it is shown in (Chassagneux, Crisan, and Delarue 2019) that the algorithm converges to the solution when uniqueness is proved, the driver b is independent of Z_t, $\mathcal{L}(Z_t)$ and when f does not depend on $\mathcal{L}(Z_t)$.
We give a recursive version of the procedure in algorithm 5 where J is a parameter for convergence of iterations between levels.

Algorithm 5 Level iteration algorithm

function SOLVER($k, \mathcal{L}(X_{T_k})$)
  if $k == L_{p-1}$ **then**
    $(Y_{T_k}(\theta_k, .), \mathcal{L}(X_{T_{k+1}})) = \phi_k(\mathcal{L}(X_{T_k}), Y_{T_{k+1}}(\theta_{k+1}, .))$
  else
    Initialize $\mathcal{L}(X_{T_{k+1}}) = \mathcal{L}(X_{T_k})$
    for $1 \leq j \leq J$ **do**
      $Y_{T_{k+1}}(\theta_{k+1}, .) = \text{Solver}(k+1, \mathcal{L}(X_{T_{k+1}}))$
      $(Y_{T_k}(\theta_k, .), \mathcal{L}(X_{T_{k+1}})) = \phi_k(\mathcal{L}(X_{T_k}), Y_{T_{k+1}}(\theta_{k+1}, .))$
    end for
  end if
  return $Y_{T_k}(\theta_k, .)$
end function

We implemented the algorithm solving (18) by local and global solvers without any dynamic versions for simplicity.
Both versions work well but we do not present the results as the methodology does not permit to extend the maturity treated. For both solvers, the algorithm converges only when the direct one also converges to the true solution. Of course, because of the need to iterate between the levels, the resolution time previously limited to a few minutes explodes to hours.
These results seem to indicate that the limiting factor of the resolution is not the fixed point iteration procedure with neural networks.

Remark 4. *In (Angiuli et al. 2019), results show that the bifurcation appears later with their scheme based on trees. In fact, their scheme is more "explicit" than ours and between the local and the global solver, the local solver is more explicit than the global one. So it it seems that the more implicit the scheme is the later the bifurcation problem appears.*

6 Conclusion

We have shown that neural network methods can solve some high dimensional FBSDE of McKean-Vlasov type. Comparing the different algorithms used we find out that

- The dynamic update of the expectation is efficient in terms of computation speed (about 30% faster than direct method) and seems to smooth the learning curve.
- Pontryagin performs better than Weak for large maturities. On the contrary, the Weak approach is the best for small maturities.
- For the linear model we observe no convergence problem whereas for the quadratic one we can solve only on a small horizon.
- The local method faces more difficulties for quadratic problems than the global methods do. It also requires more iterations to converge
- The methods can be used in dimension 10, thus applied to more realistic problems than usually. For instance, in the price impact model, the number of dimensions corresponds to the number of assets involved in the trading. Thus, developing methods able to deal with problems in high dimensions can help us to handle large portfolios.
- Previously developed algorithm used to postpone the apparition of bifurcations does not provide a way to extend maturity with neural networks.

References

Abadi, Martin et al. (2016). "TensorFlow: A System for Large-scale Machine Learning". In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. OSDI'16. Savannah, GA, USA: USENIX Association, pp. 265–283. ISBN: 978-1-931971-33-1. URL: http://dl.acm.org/citation.cfm?id=3026877.3026899.

Angiuli, Andrea, Christy V. Graves, Houzhi Li, Jean-François Chassagneux, François Delarue, and René Carmona (2019). "Cemracs 2017: numerical probabilistic approach to MFG". In: *ESAIM: Proceedings and Surveys* 65, pp. 84–113. DOI: 10.1051/proc/201965084.

Beck, Christian, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld (2019). "Deep splitting method for parabolic PDEs". In: *arXiv preprint arXiv:1907.03452*.

Beck, Christian, Weinan E, and Arnulf Jentzen (2017). "Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations". In: *CoRR* abs/1709.05963.

Cardaliaguet, Pierre and Charles-Albert Lehalle (2018). "Mean field game of controls and an application to trade crowding". In: *Mathematics and Financial Economics* 12.3, pp. 335–363.

Carmona, René and François Delarue (2018a). *Probabilistic Theory of Mean Field Games with Applications I*. Springer. DOI: 10.1007/978-3-319-58920-6. URL: https://hal.archives-ouvertes.fr/hal-01868147.

Carmona, René and François Delarue (2018b). *Probabilistic Theory of Mean Field Games with Applications I-II*. Springer.

Carmona, René and Daniel Lacker (2015). "A probabilistic weak formulation of mean field games and applications". In: *Ann. Appl. Probab.* 25.3, pp. 1189–1231. DOI: 10.1214/14-AAP1020. URL: https://doi.org/10.1214/14-AAP1020.

Carmona, René and Mathieu Laurière (2019). *Convergence Analysis of Machine Learning Algorithms for the Numerical Solution of Mean Field Control and Games: II – The Finite Horizon Case*.

Chan-Wai-Nam, Quentin, Joseph Mikael, and Xavier Warin (2019). "Machine Learning for Semi Linear PDEs". In: *Journal of Scientific Computing*. DOI: 10.1007/s10915-019-00908-3.

Chassagneux, Jean-François, Dan Crisan, and François Delarue (2014). "A probabilistic approach to classical solutions of the master equation for large population equilibria". In: *arXiv preprint arXiv:1411.3009*.

— (2019). "Numerical method for FBSDEs of McKean-Vlasov type". In: *Annals of Applied Probability* 29. DOI: 10.1214/18-AAP1429.

Fouque, Jean-Pierre and Zhaoyu Zhang (2019). *Deep Learning Methods for Mean Field Control Problems with Delay*.

Han, Jiequn, Arnulf Jentzen, and Weinan E (2017). "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115. DOI: 10.1073/pnas.1718942115.

Han, Jiequn and Jihao Long (2018). *Convergence of the Deep BSDE Method for Coupled FBSDEs*.

Huré, Côme, Huyên Pham, Achref Bachouch, and Nicolas Langrené (2018). *Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis*.

Huré, Côme, Huyên Pham, and Xavier Warin (2019). *Some machine learning schemes for high-dimensional nonlinear PDEs*.

Ji, Shaolin, Shige Peng, Ying Peng, and Xichuan Zhang (2019). *Three algorithms for solving high-dimensional fully-coupled FBSDEs through deep learning*.

Kingma, Diederik and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Lasry, Jean-Michel and Pierre-Louis Lions (2006a). "Jeux à champ moyen. I – Le cas stationnaire". In: *Comptes Rendus Mathematique - C R MATH* 343, pp. 619–625. DOI: 10.1016/j.crma.2006.09.019.

— (2006b). "Jeux à champ moyen. II – Horizon fini et contrôle optimal". In: *Comptes Rendus. Mathématique. Académie des Sciences, Paris* 10. DOI: 10.1016/j.crma.2006.09.018.

Pham, Huyên and Xavier Warin (2019). "Neural networks-based backward scheme for fully nonlinear PDEs". In: *arXiv preprint arXiv:1908.00412*.

Sergeev, Alexander and Mike Del Balso (2018). *Horovod: fast and easy distributed deep learning in TensorFlow*.

Finance for Energy Market Research Centre

Institut de Finance de Dauphine, Université Paris-Dauphine

1 place du Maréchal de Lattre de Tassigny

75775 PARIS Cedex 16

www.fime-lab.org