DEEP LEARNING ALGORITHMS FOR FBSDES WITH JUMPS: APPLICATIONS TO OPTION PRICING AND A MFG MODEL FOR SMART GRIDS *

Clemence Alasseur † Zakaria Bensaid ‡ Roxana Dumitrescu § Xavier Warin ¶

ABSTRACT. In this paper, we introduce various machine learning solvers for (coupled) forward-backward systems of stochastic differential equations (FBSDEs) driven by a Brownian motion and a Poisson random measure. We provide a rigorous comparison of the different algorithms and demonstrate their effectiveness in various applications, such as cases derived from pricing with jumps and mean-field games. In particular, we show the efficiency of the deep-learning algorithms to solve a coupled multi-dimensional FBSDE system driven by a time-inhomogeneous jump process with stochastic intensity, which describes the Nash equilibria for a specific mean-field game (MFG) problem for which we also provide the complete theoretical resolution. More precisely, we develop an extension of the MFG model for smart grids introduced in [Ala+23] to the case when the random jump times correspond to the jump times of a doubly Poisson process. We first provide an existence result of an equilibrium and derive its semi-explicit characterization in terms of a multi-dimensional FBSDE system in the linear-quadratic setting. We then compare the MFG solution to the optimal strategy of a central planner and provide several numerical illustrations using the deep-learning solvers presented in the first part of the paper.

Keywords: Machine learning; Solver; FBSDE with jumps; Deep BSDE; Pricing; Mean-field games; Cox process; Demand side management

1 Introduction

This paper is devoted to the numerical resolution of a coupled system of forward-backward stochastic differential equations (in short FBSDEs) with jumps of the form:

$$\begin{cases} dX_t = b(t, X_t, Y_t)dt + \sigma(t, X_t)dW_t + \int_{\mathbb{R}^d\setminus\{0\}} \beta(t, X_{t^-}, e)\tilde{\mathcal{J}}(dt, de), \\ dY_t = -f(t, X_t, Y_t)dt + Z_t dW_t + \int_{\mathbb{R}^d\setminus\{0\}} U_t(e)\tilde{\mathcal{J}}(dt, de), \qquad t \in [0, T], \\ X_0 = \xi, \quad Y_T = g(X_T), \end{cases} \tag{1}$$

where the functions b, σ, β, f, g, as well as the the initial condition ξ satisfy appropriate assumptions which ensure the well-posedness of the system (1).

*This work has received financial support from FIME (Finance for Energy Markets) research initiative of the Institut Europlace de Finance, which is gratefully acknowledged.

†EDF R&D & FiME, 91120 Palaiseau, France, email: clemence.alasseur@edf.fr

‡Laboratoire Manceau de Mathématiques & FR CNRS N° 2962, Institut du Risque et de l'Assurance, Université Le Mans, France, email: zakaria.bensaid@univ-lemans.fr

§Department of Mathematics, King's College London, United Kingdom, email: roxana.dumitrescu@kcl.ac.uk

¶EDF R&D & FiME, 91120 Palaiseau, France, email: xavier.warin@edf.fr

This kind of equations are linked to a class of (deterministic) partial integro-differential equations, which are non-local and take the following form:

$$\begin{cases} \frac{\partial u}{\partial t}(t,x) + \mathcal{L}u(t,x) + f\left(t,x,u(t,x)\right) = 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d, \\ u(T,x) = g(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{2}$$

where the second-order nonlocal operator $\mathcal{L}$ is defined as follows:

$$\begin{aligned} \mathcal{L}u(t,x) &= \langle b(t,x,u(t,x)), D_x u(t,x)\rangle + \frac{1}{2}\langle D^2_{xx} u(t,x)\sigma(t,x), \sigma(t,x)\rangle \\ &+ \int_{\mathbb{R}^d\setminus\{0\}} \left(u(t,x+\beta(t,x,e)) - u(t,x) - \langle D_x u(t,x), \beta(t,x,e)\rangle\right)\nu(de). \end{aligned}$$

Indeed, it is known that, under mild assumptions, $Y_t = u(t,X_t)$, where u corresponds to the viscosity solution of (2). We refer to [PP90] for a rigorous connection between PDEs and FBSDEs in a Markovian setting in the case of decoupled system of FBSDEs and Brownian filtration, further extended to the case with jumps in [BBP97] and to the case of coupled FBSDEs with jumps in [Zhe99; Zhe03].

Literature review. The resolution of partial differential equations (in short PDEs) by standard techniques as finite difference methods becomes unfeasible beyond dimension 3. An alternative method to solve nonlinear PDEs in dimension above 4 is based on the backward stochastic differential equation (in short BSDE) representation of semilinear PDEs: using the time discretization scheme proposed in [BN04], some effective algorithms based on regression have been developed in [GLW05], [LGW06] and has led to a lot of research as shown for example in [GT16]. This regression technique uses some basis functions that can be either some global polynomials as in [LS01] or some local polynomials as proposed in [BW12]: therefore this methodology still faces the curse of dimensionality and can only solve some problems in dimension below 7 or 8.

Over the past few years, machine learning methods have shown exceptional promise to solve high-dimensional nonlinear PDEs (see e.g. [DO16; HJW17; CMW19]). Machine learning methods have emerged since the pioneering papers by [HJW17] and [SS18], and have shown their efficiency for solving high-dimensional nonlinear PDEs by means of neural networks approximation. [SS18] proposes the so-called Deep Galerkin Method which uses the automatic numerical differentiation of the solution to solve the PDE on a finite domain. The authors prove the convergence of their method, but without information on the rate of convergence. An alternative methodology to solve PDEs in high-dimension is based on the BSDE representation of the solution of the PDE and deep learning approximations (see e.g. [HJW17; BEJ19; HL20; Ji+20; GPW21]). Two main classes of algorithms have been developed. The first class is based on the *global* approach, which was initially proposed in [HJW17] to tackle semi-linear PDEs. It consists in the training of as many neural networks as time steps by solving in a forward way the backward representation of the PDE solution. The Z_t process is represented by a different neural network Z_i^θ with parameters θ at each date t_i. Instead of solving the BSDE starting from the terminal condition, the method writes it down as a forward equation and an optimization problem aiming to reach the terminal condition $g(X_T)$ by minimizing a mean squared error $\mathbb{E}[|Y_T - g(X_T)|^2]$. It allows to solve PDEs in high dimension and a convergence study of Deep BSDE is conducted in [HL20]. In [BEJ19], this approach has been extended to fully nonlinear equations. Furthermore, [CMW19] shows that using a single network across all dates is more efficient, and additionally introduces a fixed-point algorithm to resolve semi-linear PDEs.

The second class of algorithms is based on the *local* approach, first proposed in [HPW20], which consists in solving local optimization problems at each time step in a backward manner. Unlike the global method, the local method involves successive optimization problems of moderate dimension. At each time step, local neural networks are trained, thus it results in as many learning problems as time steps with two neural networks (in the setting of a Brownian filtration). This process is further simplified by utilizing strategies inspired from the standard backward resolution of BSDEs with conditional expectations and regression techniques from [BN04; GLW05; LGW06; BD07]. The resulting solver was named the Deep Backward Dynamic Programming (DBDP) solver. Furthermore, the methodology is then expanded to handle fully nonlinear PDEs in [PWG21] by merging

it with ideas proposed in [Bec+19]. Additionally, extensive tests performed in [HPW20] indicate that the local method yields superior results compared to the global one, such as [PWG21] for fully nonlinear dynamics. A more robust machine learning solver, called deep backward multi-step scheme (MDBDP), was introduced in [GPW21] that builds on the Linear Regression Multi step-forward Dynamic Programming (MDP) scheme for discrete BSDEs introduced in [GT16]. According to the authors, the multi-step scheme yields the best performance when compared to other algorithms in the *local* approach.

Machine learning techniques to solve coupled FBSDEs within a Brownian filtration are explored in [HL20] and [Ji+20]. The resulting algorithms are all rooted in the *global* approach first introduced in [HJW17].

The resolution of partial integro-differential equations (in short PIDEs) has been much less regarded in the literature, even in the decoupled case, the main approaches to solve them being based on the finite-difference methods (see e.g. [VC05]) and the probabilistic representation of the solution in terms of a FBSDE system, a discrete time approximation of the associated decoupled forward-backward SDE with jumps being proposed in [BE08]. As it can be noticed above, the literature on machine learning solvers for standard PDEs is quite rich by now. In contrast, the case of integro-differential PDEs has received very little attention. Several algorithms have been recently proposed in: [BCN22; FK22; GPP22; LG23]). In these papers, the deep-learning solvers are based on the approximation of the solution of the PIDE and, for the gradient, either another neural network is employed [BCN22; FK22; LG23], or the Automatic differentiation in TensorFlow is applied [GPP22].

Contributions. The aim of our paper is to develop deep-learning solvers for the (coupled) FBSDE system (1) and a specific multi-dimensional coupled FBSDE system driven by a time-inhomogeneous Poisson process with stochastic intensity which is shown to solve an extended version of the MFG model in [Ala+23]. Our main contributions are the following:

- In the first part of the paper, we introduce five different algorithms to solve the system (1). Furthermore, we propose two different variants of the DBDP and MDBDP solvers to handle the jumps part. We emphasize that most of the literature on deep learning solvers for FBSDEs with jumps does not treat the fully coupled case and that the algorithms developed in this paper are new also in the context of decoupled FBSDEs with jumps.

- We provide a rigorous numerical comparison between all methods in terms of computation time, stability and convergence for different pricing models, which require solving a decoupled FBSDE system. To assess the performance of the algorithms for coupled FBSDEs, we purposefully introduce an equivalent coupled FBSDE system in the context of pricing which uses the explicit form of the analytical solution already known. This allows to benchmark the deep learning solvers in different settings.

- In the second part of the paper, we develop an extension of the MFG model introduced in [Ala+23] to the case when the random jump times correspond to the jumps times of a doubly Poisson process. We first provide an existence result of an equilibria by using the stochastic maximum principle and derive its semi-explicit characterization in terms of a multi-dimensional coupled FBSDE system driven by a Cox process in the linear-quadratic setting. We then compare the MFG solution to the optimal strategy of a central planner.

- We build a numerical algorithm based on the deep-learning solvers presented in the first part of the paper to solve the multi-dimensional coupled FBSDE systems driven by a Cox process, which characterize the Nash equilibrium for the MFG problem and the mean-field optimal control of the central planner. In particular, we numerically demonstrate the robustness of our deep learning-based numerical methods in handling time-inhomogeneous jump processes with stochastic intensity.

The paper is organized as follows: in Section 2.1, we give some Preliminaries on existence and uniqueness results for (coupled) FBSDEs with jumps and on neural networks. In Section 2.2, we introduce the five different deep learning solvers for (coupled) FBSDEs with jumps. In Section 2.3, we perform several numerical tests for examples derived from option pricing and provide a detailed analysis and comparison between the different algorithms. In Section 3.1, we present the mean-field game model. In Section 3.2, we characterize the mean-field equilibria, and in Section 2.3 we study the related problem of a central planner and characterize the mean-field optimal control (MFC). Section 3.4. is devoted to the implementation of the Deep learning solvers for the multi-dimensional fully coupled FBSDE system characterizing the MFG equilibria (and the MFC optimal control), as well as to the comparison between the different algorithms. Finally, in Section 3.5 we provide an interpretation of the numerical results from an economic perspective.

2 General Deep Learning algorithms for coupled FBSDEs with jumps

This section is devoted to the presentation of different deep learning algorithms for coupled FBSDEs with jumps and of their performance on several numerical examples. We shall start with some preliminaries.

2.1 Preliminaries

In this subsection, we first introduce some notation, as well as some existence and uniqueness results related to coupled FBSDEs with jumps. We then focus on neural networks which are used to solve numerically the FBSDE system.

Coupled FBSDEs with jumps. Fix a time horizon $T < \infty$ and let $(\Omega, \mathbb{F}, \mathbb{P})$ be a complete probability space. Let W_t be a d-dimensional Brownian motion and $\mathcal{J}(dt, de)$ an independent Poisson random measure with compensator $\nu(de)dt$ such that $\nu(de)$ is a σ-finite measure on $\mathbb{R}^d \setminus \{0\}$, equipped with its Borel field $\mathcal{B}(\mathbb{R}^d \setminus \{0\})$. Let $\tilde{\mathcal{J}}$ be the compensated jump measure, i.e. $\tilde{\mathcal{J}}(dt, de) := \mathcal{J}(dt, de) - \nu(de)dt$. Let $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$ be the (completed) filtration associated with W and $\mathcal{J}$. Assume that ν satisfies the condition

$$\int_{\mathbb{R}^d\setminus\{0\}} (1 \wedge |e|^2)\nu(de) < \infty.$$

We now introduce the following spaces, using the usual inner product and the Euclidean norm in $\mathbb{R}^d, \mathbb{R}^k$ and $\mathbb{R}^{k\times d}$, respectively.

- $L^2(\mathcal{G}, \mathbb{R}^d)$ is the set of $\mathbb{R}^d$-valued random variables ξ which are $\mathcal{G}$-measurable such that $\mathbb{E}\left[|\xi|^2\right] < +\infty$, where $\mathcal{G}$ is a sub-σ-algebra of $\mathcal{F}_T$.
- $\mathcal{S}^2$ is the set of $\mathbb{F}$-adapted càdlàg $\mathbb{R}^k$-valued processes Y such that $\mathbb{E}\left[\sup_{0\le t\le T} |Y_t|^2\right] < +\infty$.
- $\mathcal{H}^2$ is the set of $\mathbb{F}$-predictable $\mathbb{R}^{k\times d}$-valued processes Z such that $\|Z\|^2 := \mathbb{E}[\int_0^T |Z_t|^2 dt] < +\infty$.
- $\mathcal{H}^2_\nu$ is the set of $\mathcal{P} \otimes \mathcal{B}(\mathbb{R}^d \setminus \{0\})$-measurable maps U taking values in $\mathbb{R}^k$ such that $\|U\|^2_\nu := \mathbb{E}[\int_0^T \int_{\mathbb{R}^d\setminus\{0\}} |U_t(e)|^2 \nu(de)dt] < +\infty$, where $\mathcal{P}$ denotes the σ-field of $\mathbb{F}$-predictable subsets of $\Omega \times [0, T]$.

We now introduce the following coupled FBSDE system with jumps:

$$\begin{cases} dX_t = b(t, X_t, Y_t)dt + \sigma(t, X_t)dW_t + \int_{\mathbb{R}^d\setminus\{0\}} \beta(t, X_{t^-}, e)\tilde{\mathcal{J}}(dt, de), \\ dY_t = -f(t, X_t, Y_t)dt + Z_t dW_t + \int_{\mathbb{R}^d\setminus\{0\}} U_t(e)\tilde{\mathcal{J}}(dt, de), \\ X_0 = \xi, \quad Y_T = g(X_T), \end{cases} \qquad t \in [0, T], \tag{3}$$

where the functions $b : [0,T] \times \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$, $\sigma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, $\beta : [0,T] \times \mathbb{R}^d \times \mathbb{R}^d \setminus \{0\} \to \mathbb{R}^d$, $f : [0,T] \times \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^k$ and $g : \mathbb{R}^d \to \mathbb{R}^k$ are measurable maps which have to satisfy the following assumptions ensuring the well-posedness of the FBSDE system.

Assumption 2.1. *(i) b and f are uniformly Lipschitz with respect to (x, y), and there exists $\rho : \mathbb{R}^d \setminus \{0\} \to \mathbb{R}^+$ with $\int_{\mathbb{R}^d\setminus\{0\}} \rho^2(e)\nu(de) < +\infty$ such that for any $t \in [0,T]$, $x, \bar{x} \in \mathbb{R}^d$, and $e \in \mathbb{R}^d \setminus \{0\}$, we have:*

$$|\beta(t, x, e) - \beta(t, \bar{x}, e)| \leq \rho(e)|x - \bar{x}|.$$

(ii) σ and g are uniformly Lipschitz with respect to $x \in \mathbb{R}^d$.

(iii) Furthermore,

$$\int_0^T |b(s, 0, 0)|^2 ds + \int_0^T |f(s, 0, 0)|^2 ds + \int_0^T \int_{\mathbb{R}^d\setminus\{0\}} |\beta(s, 0, e)|^2 \nu(de) ds < \infty.$$

Given an $k \times d$ full-rank matrix G and G^T being the transposed matrix of G, we define:

$$\pi = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } \mathbb{R}^d \times \mathbb{R}^k, \quad A(t, \pi) = \begin{pmatrix} -G^T f \\ Gb \end{pmatrix}(t, \pi) \text{ in } \mathbb{R}^d \times \mathbb{R}^k.$$

For any $\pi = (x, y)$, and $\bar{\pi} = (\bar{x}, \bar{y})$, let us denote $\tilde{x} = x - \bar{x}$, and $\tilde{y} = y - \bar{y}$. We also assume the following monotonicity conditions hold .

Assumption 2.2. *There exists $\gamma_1, \gamma_2, \mu_1$ non negative constants with $\gamma_1 + \gamma_2 > 0$, $\gamma_2 + \mu_1 > 0$, such that*

(i) $\langle A(t, \pi) - A(t, \bar{\pi}), \pi - \bar{\pi} \rangle \leq -\gamma_1 |G\tilde{x}|^2 - \gamma_2 |G^T \tilde{y}|^2$.

(ii) $\langle g(x) - g(\bar{x}), G(x - \bar{x}) \rangle \geq \mu_1 |G\tilde{x}|^2$,

Moreover, we have $\gamma_1 > 0, \mu_1 > 0$ (respectively, $\gamma_2 > 0$) when $k > d$ (respectively, $k < d$).

Assumption 2.3. *Assume that b, f, σ and β are uniformly $\frac{1}{2}$-Hölder continuous in time.*

Assumption 2.4. *Assume that $k = 1$ and for any $t \in [0,T]$, $(x, y) \in \mathbb{R}^{d+1}$, we have:*

$$|b(t, x, y)| + |f(t, x, y)| + |\sigma(t, x)| + |g(x)| \leq K(1 + |x| + |y|),$$

and there exists $\rho : \mathbb{R}^d \setminus \{0\} \to \mathbb{R}^+$ with $\int_{\mathbb{R}^d\setminus\{0\}} \rho^2(e)\nu(de) < +\infty$ such that for any $t \in [0,T]$, $x \in \mathbb{R}^d$, and $e \in \mathbb{R}^d \setminus \{0\}$, we have $|\beta(t, x, e)| \leq \rho(e)(1 + |x|)$.

We now give two well-posedness results for the FBSDE (3), as well as a decoupling field representation of the Y-component of the system, which follow from [Zhe99], [LW14], and [BE08].

Theorem 2.1 (*Well-posedness for arbitrary time horizon*). *Under the Assumptions 2.1 and 2.2, for any* $\xi \in L^2(\mathcal{F}_0, \mathbb{R}^d)$*, FBSDE (3) has a unique solution* $(X, Y, Z, U) \in \mathcal{S}^2 \times \mathcal{S}^2 \times \mathcal{H}^2 \times \mathcal{H}^2_\nu$.

We give here an alternative existence and uniqueness result for a fully-coupled FBSDE in small time.

Theorem 2.2 (*Well-posedness in small time*). *Under the Assumptions 2.1 and 2.4, there exists a constant* $\delta_0 > 0$ *only depending on* ρ*,* K*, and the Lipschitz constants of* b*,* σ*, and* f *such that for every* $0 \leq \delta \leq \delta_0$*, and* $\xi \in L^2(\mathcal{F}_t, \mathbb{R}^d)$*, FBSDE (3) has a unique solution* $(X_s, Y_s, Z_s, U_s)_{s\in[t,t+\delta]}$ *on the time interval* $[t, t+\delta]$.

Let us introduce the decoupling field:

$$u(t, x) = Y^{t,x}_s|_{s=t}, \quad (t, x) \in [0, T] \times \mathbb{R}^d,$$

where $Y^{t,x}$ is the solution of the FBSDE (3) with the initial condition $X_t = x$. Using the Markov property of the forward component X of the system (3) and the continuity of the function u with respect to x, it is shown in [LW14] that, under the above assumptions, for any $(t, \xi) \in [0, T] \times L^2(\mathcal{F}_t, \mathbb{R}^d)$, we have

$$Y^{t,\xi}_s = u(s, X_s),\ t \leq s \leq T, \quad \mathbb{P} - \text{a.s.} \tag{4}$$

where X is the solution of the SDE with initial state ξ at time t and $Y^{t,\xi}$ the associated backward component of the FBSDE system (3). Furthermore, under the given assumptions on the coefficients, u is uniformly Lipschitz, and has linear growth with respect to $x \in \mathbb{R}^d$. Finally, by the assumption (2.3), we recover the $\frac{1}{2}$-Hölder continuity of the decoupling field u with respect to time which is essential for the discrete approximation discussed later. We also have the following representation for the component U of the solution: for all $(t, e) \in [0, T] \times \mathbb{R}^d \setminus \{0\}$,

$$U_t(e) = u(t, X_{t^-} + \beta(t, X_{t^-}, e)) - u(t, X_{t^-}), \quad \mathbb{P} - \text{a.s.}$$

The second part of the preliminaries concentrates on a brief introduction to neural networks.

<u>*Neural networks.*</u> We consider a feedforward neural network, denoted by Φ^θ, which approximates the processes of interest. Let d_0 be the input dimension, and d_1 be the output dimension. We fix an integer $L \geq 2$ to represent the total number of layers, including the input and output layers. We define m to be the number of neurons on each hidden layer, and for simplicity, we set $m_0 = d_0$ and $m_{L-1} = d_1$.

The feedforward neural network is defined as the composition of affine transformations and nonlinear activation functions. Specifically, we have:

$$\Phi^\theta = A_L \circ \sigma_a \circ A_{L-1} \circ \cdots \circ \sigma_a \circ A_1,$$

where σ_a is a component-wise activation function, A_1 is a mapping from $\mathbb{R}^{d_0}$ to $\mathbb{R}^m$, A_L is a mapping from $\mathbb{R}^m$ to $\mathbb{R}^{d_1}$ and for $l = 2$ to $L - 1$, A_l is a mapping from $\mathbb{R}^m$ to $\mathbb{R}^m$.

We represent each affine function A_l as $A_l(x) = W_l x + \beta_l$, where W_l is a matrix of weights and β_l is a vector of biases.

The neural network has parameters θ, which include all the weights and biases of the affine functions. The total number of parameters is $N^L_{d_0,m,d_1} = (d_0 + 1)m + (L - 2)m(1 + m) + (m + 1)d_1$, where m is the number of neurons on each hidden layer.

We denote by $\mathcal{NN}_\infty$ the set of such functions Φ^θ. To restrict the number of neurons per layer, we introduce $\mathcal{NN}_p$ the set of neural networks with at most $p \in \mathcal{N}$ neurons per hidden layer and $L-1$ hidden layers. We recall here the two following approximation theorems.

Theorem 2.3 (Universal Approximation Theorem)**.** *Assume that the function σ_a is non constant and bounded. Let μ denote a probability measure on $\mathbb{R}^d$, then for any $L \geq 2$, $\mathcal{NN}_\infty$ is dense in $L^2(\mathbb{R}^d, \mu)$.*

Theorem 2.4 (Universal Approximation Theorem)**.** *Assume that the function σ_a is non constant, bounded and a continuous function, then when $L = 2$, $\mathcal{NN}_\infty$ is dense in $C(\mathbb{R}^d)$ for the topology of the uniform convergence on compact sets.*

2.2 Deep Learning algorithms

We introduce here five *deep learning algorithms* to solve the coupled system of forward-backward SDEs with jumps (1) in the case of jumps with finite activity (i.e. $\lambda = \int_{\mathbb{R}^d \setminus \{0\}} \nu(de) < \infty$).

Let us first define the Lévy process J associated with the Poisson random measure $\mathcal{J}$, which is given, for $0 \leq t \leq T$, by

$$J_t := \int_0^t \int_{\mathbb{R}^d \setminus \{0\}} e\mathcal{J}(ds, de). \tag{5}$$

and $\Delta J_t := J_t - J_{t^-}$, for all $t > 0$. We also introduce the following Poisson process, denoted by N_t:

$$N_t := \int_0^t \int_{\mathbb{R}^d \setminus \{0\}} \mathcal{J}(ds, de).$$

The Poisson Process (N_t) has the intensity λt.

By defining the map $\bar{b}(t, x, y) := b(t, x, y) - \int_{\mathbb{R}^d \setminus \{0\}} \beta(t, x, e)\nu(de)$, we observe that the FBSDE (3) system can be written as:

$$\begin{cases} dX_t = \bar{b}(t, X_t, Y_t)dt + \sigma(t, X_t)dW_t + \int_{\mathbb{R}^d \setminus \{0\}} \beta(t, X_{t^-}, e)\mathcal{J}(dt, de), \\ dY_t = -f(t, X_t, Y_t)dt + Z_t dW_t + \int_{\mathbb{R}^d \setminus \{0\}} U_t(e)\tilde{\mathcal{J}}(dt, de), \qquad t \in [0, T], \\ X_0 = \xi, \quad Y_T = g(X_T). \end{cases} \tag{6}$$

Remark 1. *The numerical approximation of the compensator of the jump part of the forward component in the drift $\bar{b}$ can be done through numerous methods. For example, explicit integration with respect to the intensity measure, Monte Carlo estimation, or the methods used in [VC05].*

By using (4), the FBSDE system (6) reads as follows:

$$\begin{cases} X_t = \xi + \int_0^t \bar{b}(s, X_s, u(s, X_s))ds + \int_0^t \sigma(s, X_s)dW_s + \int_0^t \int_{\mathbb{R}^d \setminus \{0\}} \beta(s, X_{s^-}, e)\mathcal{J}(ds, de), \\ u(t, X_t) = g(X_T) - \int_t^T f(s, X_s, u(s, X_s))ds + \int_t^T Z_s dW_s + \int_t^T \int_{\mathbb{R}^d \setminus \{0\}} U_s(e)\tilde{\mathcal{J}}(ds, de). \end{cases} \quad t \in [0, T]. \tag{7}$$

Discrete-time approximation. Let us consider a uniform time grid $\pi := \{t_0, t_1, ..., t_M\}$ where $t_i := i\frac{T}{M}$ for $i \in \{0, 1, ..., M\}$ and $\Delta t_i := t_{i+1} - t_i$ represents the constant time step size. We also define the Brownian increment ΔW_i as $\Delta W_i := W_{t_{i+1}} - W_{t_i}$ and the Poisson increment $\Delta N_i := N_{t_{i+1}} - N_{t_i}$, which follows a Poisson distribution with mean $\lambda \Delta t_i$. Finally, for a fixed $i \in \{0, 1, \ldots, M-1\}$, denote by $(\Delta J_l^i)_{l \in [1, \Delta N_i]}$ the lth jump of the process (J_t) given by (5) which occurs on the time interval $]t_i, t_{i+1}]$.

To give the intuition about the approximation of the backward SDE of the system (7) and, in particular, about the treatment of the jump part, we first introduce the following continuous-time process $(\bar{X}_t^\pi)_{t\in[0,T]}$:

$$\bar{X}_t^\pi := X_{t_i} + \int_{t_i}^t \int_{\mathbb{R}^d\setminus\{0\}} \beta(t_i, \bar{X}_{t_i}^\pi, e)\mathcal{J}(ds, de), \quad \forall t \in [t_i, t_{i+1}[, \forall i \in [|0, M-1|],$$

and the process $(\bar{U}_t^\pi)$ which is defined as follows

$$\bar{U}_t^\pi(e) := u(t_i, \bar{X}_{t^-}^\pi + \beta(t_i, \bar{X}_{t_i}^\pi, e)) - u(t_i, \bar{X}_{t^-}^\pi), \quad \forall t \in [t_i, t_{i+1}[, \forall i \in [|0, M-1|].$$

We can observe that, for a number of time steps M sufficiently large, we have the following approximation:

$$u(t_i, X_{t_i}) \approx u(t_{i+1}, X_{t_{i+1}}) + f(t_i, X_{t_i}, u(t_i, X_{t_i}))\Delta t_i - \bar{Z}_i^\pi \Delta W_i - \int_{t_i}^{t_{i+1}} \int_{\mathbb{R}^d\setminus\{0\}} \bar{U}_s^\pi(e)\tilde{\mathcal{J}}(ds, de),$$

with $\bar{Z}_i^\pi := \frac{1}{\Delta t_i}\mathbb{E}[\int_{t_i}^{t_{i+1}} Z_s ds|\mathcal{F}_{t_i}]$. Note that the integral of $(\bar{U}_t^\pi)$ with respect to the Poisson measure $\mathcal{J}$ admits the representation:

$$\begin{aligned}
\int_{t_i}^{t_{i+1}} \int_{\mathbb{R}^d\setminus\{0\}} \bar{U}_s^\pi(e)\mathcal{J}(ds, de) &= \int_{t_i}^{t_{i+1}} \int_{\mathbb{R}^d\setminus\{0\}} u(t_i, \bar{X}_{s^-}^\pi + \beta(t_i, \bar{X}_{t_i}^\pi, e)) - u(t_i, \bar{X}_{s^-}^\pi)\mathcal{J}(ds, de) \\
\text{(by definition of } \bar{X}^\pi) &= \sum_{k=1}^{\Delta N_i} u(t_i, X_{t_i} + \sum_{l=1}^{k} \beta(t_i, X_{t_i}, \Delta J_l^i)) - u(t_i, X_{t_i} + \sum_{l=1}^{k-1} \beta(t_i, X_{t_i}, \Delta J_l^i)), \\
&= u(t_i, X_{t_i} + \sum_{l=1}^{\Delta N_i} \beta(t_i, X_{t_i}, \Delta J_l^i)) - u(t_i, X_{t_i}).
\end{aligned}$$

By using the Euler scheme to approximate the solution X_t of the SDE, we are led to the following discrete time approximation of the solution of the FBSDE system (6):

$$\begin{cases}
X_{i+1}^\pi = X_i^\pi + \bar{b}(t_i, X_i^\pi, u(t_i, X_i^\pi))\Delta t_i + \sigma(t_i, X_i^\pi)\Delta W_i + \sum_{l=1}^{\Delta N_i} \beta(t_i, X_i^\pi, \Delta J_l^i), \\
u(t_i, X_i^\pi) \approx u(t_{i+1}, X_{i+1}^\pi) + f(t_i, X_i^\pi, u(t_i, X_i^\pi))\Delta t_i - Z_i^\pi \Delta W_i - \left(u(t_i, X_i^\pi + \sum_{l=1}^{\Delta N_i} \beta(t_i, X_i^\pi, \Delta J_l^i)) - u(t_i, X_i^\pi) \right) \\
+\mathbb{E}\left[u(t_i, X_i^\pi + \sum_{l=1}^{\Delta N_i} \beta(t_i, X_i^\pi, \Delta J_l^i)) - u(t_i, X_i^\pi) \Big| \mathcal{F}_{t_i} \right], \\
Z_i^\pi = \mathbb{E}\left[u(t_{i+1}, X_{i+1}^\pi) \frac{\Delta W_i}{\Delta t_i} | \mathcal{F}_{t_i} \right], \\
X_0^\pi = \xi, \quad u(t_M, X_M^\pi) = g(X_M^\pi), \\
i = 0, \cdots, M-1.
\end{cases} \tag{8}$$

Remark 2. • *The approximation proposed here for the jumps part for the backward component is different from the one proposed in [GPP22] and is particularly well-suited for the deep learning framework. In this approach, the neural networks deal with the sum of the jumps rather than handling each jump individually which can be problematic for large intensities given a small number*

of time steps. The numerical tests that we have conducted show the robustness of this approximation. [1]

- *To handle the case when the Poisson increment* ΔN *equals* 0*, we adopt the convention* $\sum_{l=1}^{0} \phi_l := 0$*, where* $(\phi)_l$ *represents a sequence of random variables.*

From the discrete-time FBSDE to neural networks. We denote by $\mathcal{U}^\theta$ the network function approximating the decoupling field u, $\mathcal{Z}^\theta$ the network function approximating the process Z, and $\mathcal{W}^\theta$ the network function approximating the function $t, x, y \to u(t, x + y) - u(t, x)$. We present two families of algorithms : one is based on the representation (8) (below denoted by first class of algorithms) and the second one relies on the regression methods (denoted by second class of algorithms).

2.2.1 First class of deep-learning algorithms.

In this part, we introduce the deep-learning algorithms based on the representation (8) (with possibly two variants depending on the algorithm).

1. Global solver. This algorithm extends to the case of jumps and fully coupled setting the *Deep BSDE* solver developed in [HJW17], where each neural network takes t (i.e. time) as input (see [CMW19]). In our setting, we use three networks: $\mathcal{Y}^\theta$ to approximates the initial condition of the backward component, $\mathcal{Z}^\theta$ to approximate the control Z and $\mathcal{W}^\theta$ to approximate the jump part in equation (8) leading to

$$Y^\pi_{i+1} \approx Y^\pi_i - f(t_i, X^\pi_i, Y^\pi_i)\Delta t_i + \mathcal{Z}^\theta(t_i, X^\pi_i)\Delta W_i + \mathcal{W}^\theta(t_i, X^\pi_i, \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J^i_l)) - \mathbb{E}\left[\mathcal{W}^\theta(t_i, X^\pi_i, \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J^i_l))\big|\mathcal{F}_{t_i}\right],$$

where $\tilde{\beta}_i(\Delta J^i_l) = \beta(t_i, X^\pi_i, \Delta J^i_l)$. Notice that the network $\mathcal{W}^\theta$ has to depend on t, X^π, and $\sum_s \tilde{\beta}_s(\Delta J_s)$.

Observe that we have the following result to compute the conditional expectation by means of Monte Carlo on each trajectory of the batch:

$$\mathbb{E}\left[\mathcal{W}^\theta(t_i, X^\pi_i, \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J^i_l))\big|\mathcal{F}_{t_i}\right] = \Theta(t_i, X^\pi_i),$$

$$\text{where} \qquad \Theta(t, x) = \mathbb{E}\left[\mathcal{W}^\theta(t, x, \sum_{l=1}^{\Delta N_i} \beta(t, x, \Delta J^i_l))\right], \quad \forall (t, x) \in [0, T] \times \mathbb{R}^d.$$

Thus, we choose small batch sizes during the gradient descent in order to estimate the compensator with a large number of Monte Carlo simulations for each sample of the batch.

Let $\theta = (\theta_0, \theta_1, \theta_2)$, where $\theta_0 \in \mathbb{R}^{N^L_{d,m,k}}$ are the parameters of the network function $\mathcal{Y}^{\theta_0}$, $\theta_1 \in \mathbb{R}^{N^L_{d+1,m,kd}}$, are the parameters of the network function $\mathcal{Z}^{\theta_1}$, $\theta_2 \in \mathbb{R}^{N^L_{2d+1,m,k}}$ are the parameters of the network function $\mathcal{W}^{\theta_2}$. This method consists in training the neural networks by solving in a forward way the backward representation of the solution, i.e. instead of solving the BSDE starting from the terminal condition, one estimates Y_0 with $\mathcal{Y}^{\theta_0}(\xi)$ and solves the forward optimization problem with the aim of minimizing $\mathbb{E}\left[|Y_T - g(X_T)|^2\right]$. The *Global solver* is detailed in Algorithm 1.

[1] The full convergence of the algorithms proposed in this paper will be provided in an upcoming paper.

Algorithm 1: Global solver

for $m = 0, \dots, K$ **do**

  $\forall j \in [|1, B|]$ sample ξ_j from the law of ξ, and set $X_0^j(\theta) = \xi_j$, $Y_0^j(\theta) = \mathcal{Y}^{\theta_0}(\xi_j)$;

  for $i = 0, \dots, M-1$ **do**

    for $j = 0, \dots, B$ **do**

      Sample ΔW_i^j from a Gaussian vector, sample ΔN_i^j from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta J_l^{i,j})_{l=1,\cdots,\Delta N_i^j}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.

      $$X_{i+1}^j(\theta) = X_i^j(\theta) + \bar{b}(t_i, X_i^j(\theta), Y_i^j(\theta))\Delta t_i + \sigma(t_i, X_i^j(\theta))\Delta W_i^j + \sum_{l=1}^{\Delta N_i^j} \beta(t_i, X_i^j(\theta), \Delta J_l^{i,j})$$

    end

    for $k = 0, \dots, A$ **do**

      Sample $\Delta \bar{N}_i^k$ from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta \bar{J}_l^{i,k})_{l=1,\cdots,\Delta \bar{N}_i^k}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.

    end

    for $j = 0, \dots, B$ **do**

      $$\begin{aligned} Y_{i+1}^j(\theta) = & Y_i^j(\theta) - f(t_i, X_i^j(\theta), Y_i^j(\theta))\Delta t_i + \mathcal{Z}^{\theta_1}(t_i, X_i^j(\theta))\Delta W_i^j \\ & + \mathcal{W}^{\theta_2}(t_i, X_i^j(\theta), \sum_{l=1}^{\Delta N_i^j} \tilde{\beta}_i^j(\Delta J_l^{i,j})) - \frac{1}{A} \sum_{k=1}^{A} \left[\mathcal{W}^{\theta_2}(t_i, X_i^j(\theta), \sum_{l=1}^{\Delta \bar{N}_i^k} \tilde{\beta}_i^j(\Delta \bar{J}_l^{i,k})) \right], \end{aligned}$$

    end

  end

  $\phi(\theta) = \frac{1}{B} \sum_{j=1}^{B} \left| Y_M^j(\theta) - g(X_M^j(\theta)) \right|^2$.

  $\theta = \theta - r_m \nabla \phi(\theta)$

end

2. SumLocal solver. The second algorithm we develop extends the one introduced in [HPW20]. The setting with jumps is more involved and we propose two variants of this algorithm to deal with the jumps part of the backward component. The Y component is approximated by a neural network $\mathcal{U}^\theta$ and the two variants of the algorithm can be written as follows:

(i) We can directly use the system (8) giving the *SumLocal1* solver:

$$\begin{aligned}\mathcal{U}^\theta(t_{i+1}, X^\pi_{i+1}) \approx \;& \mathcal{U}^\theta(t_i, X^\pi_i) - f(t_i, X^\pi_i, \mathcal{U}^\theta(t_i, X^\pi_i))\Delta t_i + \mathcal{Z}^\theta(t_i, X^\pi_i)\Delta W_i + \\ & \mathcal{U}^\theta(t_i, X^\pi_i + \sum_{p=1}^{\Delta N_i} \beta(t_i, X^\pi_i, \Delta J^i_p)) - \mathbb{E}\left[\mathcal{U}^\theta(t_i, X^\pi_i + \sum_{p=1}^{\Delta N_i} \beta(t_i, X^\pi_i, \Delta J^i_p))\Big|\mathcal{F}_{t_i}\right].\end{aligned}$$

(ii) Or, as in the *Global solver*, we can use a network $\mathcal{W}^\theta$ for the jump part, which gives the *SumLocal2* solver :

$$\begin{aligned}\mathcal{U}^\theta(t_{i+1}, X^\pi_{i+1}) \approx \;& \mathcal{U}^\theta(t_i, X^\pi_i) - f(t_i, X^\pi_i, \mathcal{U}^\theta(t_i, X^\pi_i))\Delta t_i + \mathcal{Z}^\theta(t_i, X^\pi_i)\Delta W_i + \\ & \mathcal{W}^\theta(t_i, X^\pi_i, \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J^i_l)) - \mathbb{E}\left[\mathcal{W}^\theta(t_i, X^\pi_i, \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J^i_l))\Big|\mathcal{F}_{t_i}\right].\end{aligned}$$

Let $\theta = (\theta_0, \theta_1, \theta_2)$ where $\theta_0 \in \mathbb{R}^{N^L_{d+1,m,kd}}$ are the parameters of the network function $\mathcal{Z}^{\theta_0}$, $\theta_1 \in \mathbb{R}^{N^L_{1+2d,m,k}}$ are the parameters of the network function $\mathcal{W}^{\theta_1}$, and $\theta_2 \in \mathbb{R}^{N^L_{d+1,m,k}}$ are the parameters of the network function $\mathcal{U}^{\theta_2}$. We detail the *SumLocal2 solver* in Algorithm 2.

3. SumMultiStep solver. The third algorithm represents a multistep version of the previous one, and extends the solver proposed in [GPW21] to the jumps setting. It also has two versions, both based on the representation (8), for *SumMultiStep1* we approximate the jumps part in the backward SDE as in (i) above and for *SumMultiStep2* we approximate the jumps part in the backward SDE as in (ii) above. Let $\theta = (\theta_0, \theta_1, \theta_2)$ where $\theta_0 \in \mathbb{R}^{N^L_{d+1,m,kd}}$ are the parameters of the network function $\mathcal{Z}^{\theta_0}$, $\theta_1 \in \mathbb{R}^{N^L_{1+2d,m,k}}$ are the parameters of the network function $\mathcal{W}^{\theta_1}$, and $\theta_2 \in \mathbb{R}^{N^L_{d+1,m,k}}$ are the parameters of the network function $\mathcal{U}^{\theta_2}$. The *SumMultiStep2 solver* is described in detail in Algorithm 3.

Algorithm 2: SumLocal solver (SumLocal2 variant).

for $m = 0, \dots, K$ **do**

  Set $\forall j \in [|1, B|]$ $X_0^j(\theta) = x_0$;

  for $i = 0, \dots, M-1$ **do**

    for $j = 1, \dots, B$ **do**

      Sample ΔW_i^j from a Gaussian vector, sample ΔN_i^j from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta J_l^{i,j})_{l=1,\cdots,\Delta N_i^j}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.

$$X_{i+1}^j(\theta) = X_i^j(\theta) + \bar{b}(t_i, X_i^j(\theta), \mathcal{U}^{\theta_2}(t_i, X_i^j(\theta)))\Delta t_i + \sigma(t_i, X_i^j(\theta))\Delta W_i^j + \sum_{l=1}^{\Delta N_i^j} \beta(t_i, X_i^j(\theta), \Delta J_l^{i,j}).$$

    end

    for $k = 0, \dots, A$ **do**

      Sample $\Delta \bar{N}_i^k$ from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta \bar{J}_l^{i,k})_{l=1,\cdots,\Delta \bar{N}_i^k}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.

    end

  end

- $\phi_{\text{local}}(\theta) = \sum_{i=0}^{M-2} \left(\frac{1}{B} \sum_{j=1}^{B} \Big| \mathcal{U}^{\theta_2}(t_{i+1}, X_{i+1}^j(\theta)) - \mathcal{U}^{\theta_2}(t_i, X_i^j(\theta)) + f(t_i, X_i^j(\theta), \mathcal{U}^{\theta_2}(t_i, X_i^j(\theta)))\Delta t_i - \mathcal{Z}^{\theta_0}(t_i, X_i^j(\theta))\Delta W_i^j - \mathcal{W}^{\theta_1}(t_i, X_i^j(\theta), \sum_{l=1}^{\Delta N_i^j} \tilde{\beta}_i^j(\Delta J_l^{i,j})) + \frac{1}{A} \sum_{k=1}^{A} \left[\mathcal{W}^{\theta_2}(t_i, X_i^j(\theta), \sum_{l=1}^{\Delta \bar{N}_i^k} \tilde{\beta}_i^j(\Delta \bar{J}_l^{i,k})) \right] \Big|^2 \right)$

- $\phi_{\text{final}}(\theta) = \frac{1}{B} \sum_{j=1}^{B} \Big| g(X_M^j(\theta)) - \mathcal{U}^{\theta_2}(t_{M-1}, X_{M-1}^j(\theta)) + f(t_{M-1}, X_{M-1}^j(\theta), \mathcal{U}^{\theta_2}(t_{M-1}, X_{M-1}^j(\theta)))\Delta t_{M-1} - \mathcal{Z}^{\theta_0}(t_{M-1}, X_{M-1}^j(\theta))\Delta W_{M-1} - \mathcal{W}^{\theta_1}(t_{M-1}, X_{M-1}^j(\theta), \sum_{l=1}^{\Delta N_{M-1}^j} \tilde{\beta}_i(\Delta J_l^{M-1,j})) + \frac{1}{A} \sum_{k=1}^{A} \left[\mathcal{W}^{\theta_2}(t_{M-1}, X_{M-1}^j(\theta), \sum_{l=1}^{\Delta \bar{N}_{M-1}^k} \tilde{\beta}_{M-1}^j(\Delta \bar{J}_l^{M-1,k})) \right] \Big|^2$

  $\phi(\theta) = \phi_{\text{local}}(\theta) + \phi_{\text{final}}(\theta)$

  $\theta = \theta - r_m \nabla \phi(\theta)$

end

Algorithm 3: The **SumMultiStep** solver (SumMultiStep2 variant).

for $m = 0, \dots, K$ **do**
  Set $\forall j \in [|1, B|]$ $X_0^j(\theta) = x_0$;
  for $i = 0, \dots, M-1$ **do**
    for $j = 1, \dots, B$ **do**

$$\psi_i^j(\theta) = \mathcal{U}^{\theta_2}(t_i, X_i^j(\theta))$$

Sample ΔW_i^j from a Gaussian vector, sample ΔN_i^j from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta J_l^{i,j})_{l=1,\cdots,\Delta N_i^j}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.

    end
    for $s = 0, \dots, A$ **do**
      Sample $\Delta \bar{N}_i^s$ from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta \bar{J}_l^{i,s})_{l=1,\cdots,\Delta \bar{N}_i^s}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.
    end
    for $k = 0, \dots, i$ **do**
      for $j = 1, \dots, B$ **do**

$$\psi_k^j(\theta) = \psi_k^j(\theta) - f(t_i, X_i^j(\theta), \mathcal{U}^{\theta_2}(t_i, X_i^j(\theta)))\Delta t_i + \mathcal{Z}^{\theta_0}(t_i, X_i^j(\theta))\Delta W_i^j + \mathcal{W}^{\theta_1}(t_i, X_i^j(\theta), \sum_{l=1}^{\Delta N_i^j} \tilde{\beta}_i(\Delta J_l^{i,j})) - \frac{1}{A} \sum_{s=1}^{A} \left[\mathcal{W}^{\theta_2}(t_i, X_i^j(\theta), \sum_{l=1}^{\Delta \bar{N}_i^s} \tilde{\beta}_i^j(\Delta \bar{J}_l^{i,s})) \right].$$

      end
    end
    for $j = 1, \dots, B$ **do**

$$X_{i+1}^j(\theta) = X_i^j(\theta) + \bar{b}(t_i, X_i^j(\theta), \mathcal{U}^{\theta_2}(t_i, X_i^j(\theta)))\Delta t_i + \sigma(t_i, X_i^j(\theta))\Delta W_i^j + \sum_{l=1}^{\Delta N_i^j} \beta(t_i, X_i^j(\theta), \Delta J_l^{i,j}).$$

    end
  end
  $\phi(\theta) = \sum_{i=0}^{M-1} \left(\frac{1}{B} \sum_{j=1}^{B} \left| \psi_i^j(\theta) - g(X_M^j(\theta)) \right|^2 \right)$
  $\theta = \theta - r_m \nabla \phi(\theta)$
end

2.2.2 Second class of deep-learning algorithms.

In this second part, we introduce the deep-learning algorithms based on the regression methods.
The following algorithms exploit the fact the driver does not depend on Z and U, thus use a single network $\mathcal{U}^\theta$ to approximate the Y component of the solution. By conditionning the backward component in (8), we obtain

$$\mathcal{U}^\theta(t_i, X_i^\pi) \approx \mathbb{E}\left[\mathcal{U}^\theta(t_{i+1}, X_{i+1}^\pi) + f(t_i, X_i^\pi, \mathcal{U}^\theta(t_i, X_i^\pi))\Delta t_i | \mathcal{F}_{t_i}\right].$$

1. SumLocalReg solver. The first algorithm based on the regression methods is the *SumLocalReg solver*, which is a neural network version of the algorithms developed in [GLW05], [LGW06]. It is described in detail in Algorithm 4.

Algorithm 4: The **SumLocalReg solver**.

for $m = 0, \dots, K$ **do**
  Set $\forall j \in [|1, B|]$ $X_0^j(\theta) = x_0$;
  for $i = 0, \dots, M-1$ **do**
    for $j = 1, \dots, B$ **do**
      Sample ΔW_i^j from a Gaussian vector, sample ΔN_i^j from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta J_l^{i,j})_{l=1,\cdots,\Delta N_i^j}$ from the distribution $\frac{\nu(de)}{\lambda}\mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.

$$X_{i+1}^j(\theta) = X_i^j(\theta) + \bar{b}(t_i, X_i^j(\theta), \mathcal{U}^\theta(t_i, X_i^j(\theta)))\Delta t_i + \sigma(t_i, X_i^j(\theta))\Delta W_i^j + \sum_{l=1}^{\Delta N_i^j} \beta(t_i, X_i^j(\theta), \Delta J_l^{i,j}).$$

    end
  end

$$\phi_{\text{local}}(\theta) = \sum_{i=0}^{M-2}\left(\frac{1}{B}\sum_{j=1}^{B}\left|\mathcal{U}^\theta(t_{i+1}, X_{i+1}^j(\theta)) - \mathcal{U}^\theta(t_i, X_i^j(\theta)) + f(t_i, X_i^j(\theta), \mathcal{U}^\theta(t_i, X_i^j(\theta)))\Delta t_i\right|^2\right)$$

$$\phi_{\text{final}}(\theta) = \frac{1}{B}\sum_{j=1}^{B}\left|g(X_M^j(\theta)) - \mathcal{U}^\theta(t_{M-1}, X_{M-1}^j(\theta)) + f(t_{M-1}, X_{M-1}^j(\theta), \mathcal{U}^\theta(t_{M-1}, X_{M-1}^j(\theta)))\Delta t_{M-1})\right|^2$$

  $\phi(\theta) = \phi_{\text{local}}(\theta) + \phi_{\text{final}}(\theta)$

  $\theta = \theta - r_m \nabla\phi(\theta)$
end

2. SumMultiStepReg solver. The second algorithm is the *SumMultiStepReg solver*, which is a multistep version of the previous one, in the same spirit as in [BD07]. It is described in detail in Algorithm 5.

Algorithm 5: The **SumMultiStepReg** solver

for $m = 0, \dots, K$ **do**
  Set $\forall j \in [|1, B|]$ $X_0^j(\theta) = x_0$;
  for $i = 0, \dots, M-1$ **do**
    for $j = 1, \dots, B$ **do**

$$\psi_i^j(\theta) = \mathcal{U}^\theta(t_i, X_i^j(\theta))$$

      Sample ΔW_i^j from a Gaussian vector, sample ΔN_i^j from a Poisson distribution $\mathcal{P}(\lambda \Delta t_i)$ and sample each element of the jumps sequence $(\Delta J_l^{i,j})_{l=1,\cdots,\Delta N_i^j}$ from the distribution $\frac{\nu(de)}{\lambda} \mathbb{1}_{\mathbb{R}^d \setminus \{0\}}$.
    end
    for $k = 0, \dots, i$ **do**
      for $j = 1, \dots, B$ **do**
        $\psi_k^j(\theta) = \psi_k^j(\theta) - f(t_i, X_i^j(\theta), \mathcal{U}^\theta(t_i, X_i^j(\theta)))\Delta t_i$
      end
    end
    for $j = 1, \dots, B$ **do**

$$X_{i+1}^j(\theta) = X_i^j(\theta) + \bar{b}(t_i, X_i^j(\theta), \mathcal{U}^\theta(t_i, X_i^j(\theta)))\Delta t_i + \sigma(t_i, X_i^j(\theta))\Delta W_i^j + \sum_{l=1}^{\Delta N_i^j} \beta(t_i, X_i^j(\theta), \Delta J_l^{i,j}).$$

    end
  end
  $\phi(\theta) = \sum_{i=0}^{M-1} \left(\frac{1}{B} \sum_{j=1}^{B} \left| \psi_i^j(\theta) - g(X_M^j(\theta)) \right|^2 \right)$
  $\theta = \theta - r_m \nabla \phi(\theta)$
end

Remark 3. *As proposed in [GPP22] in the case of decoupled FBSDEs, another method to estimate the compensator is to consider an additional neural network function $\mathcal{C}^{\theta_3}$ approximating the compensator $t, x \to \int_{\mathbb{R}^d \setminus \{0\}} (u(t, x+e) - u(t,x))\, \nu(de)$ by adding an additional penalty term to the original loss function. Hence, the conditional expectation $\mathbb{E}\left[\mathcal{W}^{\theta_2}(t_i, X_i, \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J_l^i)) \middle| \mathcal{F}_{t_i}\right]$ is estimated by $\mathcal{C}^{\theta_3}(t_i, X_i)\Delta t_i$ at each time step by optimizing the following penalty function:*

$$\sum_{i=0}^{M-1} \left[\left| \mathcal{W}^{\theta_2}(t_i, X_i(\theta), \sum_{l=1}^{\Delta N_i} \tilde{\beta}_i(\Delta J_l^i)) - \mathcal{C}^{\theta_3}(t_i, X_i(\theta))\Delta t_i \right|^2 \right]. \tag{9}$$

We tested this approximation method in the coupled case for the Global method and both local methods, and the algorithms lacked accuracy due to the presence of an additional neural network and an additional

term in the loss function. Our approach is based on the approximation of the conditional expectation as in (9) *to directly estimate the backward component Y in the SumLocalReg and SumMultiStepReg algorithms, which yields better results.*

Remark 4. *The above algorithms can be used to handle the general case of jumps with infinite activity, after truncating the small jumps as in [DRZ21; GPP22].*

2.3 Numerical tests for option pricing

In this subsection, we aim to assess the performance of the deep learning algorithms discussed in the preceding section in the context of pricing European options in three different financial models: the Black-Scholes (BS) model (without jumps), the Merton (MJ) model (with jumps with finite activity), and the Variance Gamma (VG) model (with jumps with *infinite* activity). Indeed, we can adapt the algorithms presented in the finite-activity setting to the pricing of European-options under the exponential Variance-Gamma model.

We first set the hyper-parameters for the *Global* solver and both variants of the *SumLocal* and *SumMultiStep* solvers where NbTraining corresponds to the number of gradient iterations of the Adam stochastic gradient descent algorithm [KB14].

Parameter	value
m	21
L	2
NbTraining	12000

Parameter	value
A	5000
B	10
σ_a	tanh

Table 1: Hyper-parameters for the first class of deep-learning algorithms

Furthermore, the specific parameters of the regression methods (since the compensator is not computed) are:

Parameter	value
m	21
L	2
NbTraining	12000

Parameter	value
B	10 000
σ_a	tanh

Table 2: Hyper-parameters for the second class of deep-learning algorithms

The algorithms are implemented in Python with *Tensorflow* library. Each numerical experiment is conducted using GPU Tesla T4-PCIE-16GB. The code for the numerical experiments of the pricing and Mean Field Game (MFG) models can be accessed at the following URL: `https://github.com/ZakariaBensaid/DeepFBSDEJSolvers`.

In the Black-Scholes and Merton models, we compare the results we get by implementing our deep learning algorithms with those obtained by using the well-known *closed formula* of the solutions of the PDE, respectively PIDE. In the Variance Gamma model, we compare our results with those obtained by using the *inverse Fourier* method computed with the Fast Fourier Transform algorithm.

2.3.1 The models

We present below the three models.

The Black Scholes model *(No jumps)* The BS model proposes to model the underlying asset S_t under the risk neutral probability measure $\mathbb{Q}$ following a geometric Brownian motion with the following dynamics: $S_t = s\exp((r-\frac{\sigma^2}{2})t+\sigma W_t)$. The problem of pricing an European call option in the BS model translates to the following FBSDE:

$$\begin{cases} dS_t = S_t(rdt + \sigma dW_t), \\ -dY_t = -rY_t dt - Z_t dW_t, \qquad t \in [0,T], \\ S_0 = s, \quad Y_T = (S_T - K)^+. \end{cases} \tag{10}$$

where K is the strike price. More precisely, the price of the European option at time t is given by Y_t. Furthermore, it is known that there exists a function $\bar{u}$ such that $Y_t = \bar{u}(t, S_t)$, where the function $\bar{u}$ solves a specific PDE.

To test the performance of the algorithms in a coupled setting, we propose below a forward-backward system for which the forward component has an additional term coupled to the backward component. More precisely, we consider the following system

$$\begin{cases} dX_t = X_t(rdt + \sigma dW_t) + a|Y_t - \bar{u}(t, X_t)|dt, \\ -dY_t = -rY_t dt - Z_t dW_t, \qquad t \in [0,T], \\ X_0 = S_0, \quad Y_T = (X_T - K)^+. \end{cases} \tag{11}$$

where $\bar{u}$ is the analytical solution of the PDE in the decoupled case. In the case of a small time maturity, Theorem 2.2 guarantees that the system (11) admits an unique solution for which the backward component Y provides the price of the european call option. This applies for all the models below.

Model parameters. For the numerical implementation, we set $T = 1$, $M = 50$ steps, the interest rate $r = 0.1$, the diffusion volatility $\sigma = 0.3$, the strike price $K = 0.9$, the spot price $S_0 = 1$, and the coupling linearity coefficient when non-null $a = 0.1$.

Merton model *(Jumps with finite activity)* Merton's [Mer76] approach proposes to ignore risk premia for jumps, this assumption leading to a specific choice for pricing and hedging. To describe the model, we assume that the underlying asset S_t under the risk neutral probability measure $\mathbb{Q}$ follows the dynamics $S_t = S_0\exp((r-\sigma^2/2-m)t + \sigma W_t + \sum_{i=1}^{N_t} Y_i)$, where N_t, Y_i are independent from W_t and N_t is a Poisson process with intensity λt. The random variables Y_i are i.i.d. and follow a $\mathcal{N}(\alpha, \xi^2)$ distribution. The constant m is chosen such that the process $\tilde{S}_t = S_t e^{-rt}$ is a martingale under $\mathbb{Q}$ and is given by $m := \lambda\mathbb{E}[e^{Y_i} - 1]$.

As above, under appropriate assumptions on the coefficients, we can express the problem of pricing an European call option in the Merton model in terms of the following coupled FBSDE:

$$\begin{cases} dX_t = X_{t^-}(rdt + \sigma dW_t + \int_{\mathbb{R}^\star}(e^e - 1)\tilde{\mathcal{J}}(dt, de)) + a|Y_t - \bar{u}(t, X_t)|dt, \\ -dY_t = -rY_t dt - Z_t dW_t - \int_{\mathbb{R}^\star} U_t(e)\tilde{\mathcal{J}}(dt, de), \qquad t \in [0,T], \\ X_0 = S_0; \quad Y_T = (X_T - K)^+. \end{cases} \tag{12}$$

where $\bar{u}$ is the analytical solution of the partial integro differential equation associated to the decoupled FBSDE and $\tilde{\mathcal{J}}(dt, de)$ is the compensated jump measure associated with the compound Poisson process $\sum_{i=1}^{N_t} Y_i$ with intensity measure $\nu(de)$, where ν is given by :

$$\nu(de) = \frac{\lambda}{\xi\sqrt{2\pi}} \exp\left(-\frac{(e-\alpha)^2}{2\xi^2}\right) de.$$

Model parameters. For this example, we set $T = 1$, $M = 50$ steps, the interest rate $r = 0.1$, the diffusion volatility $\sigma = 0.3$, the jumps intensity $\lambda = 3$, the parameters of the jumps distribution $\alpha = 0$ and $\xi = 0.2$, the strike price $K = 0.9$, the initial condition $X_0 = 1$, and the coupling linearity coefficient when non-null $a = 0.1$.

It can be observed that the algorithms introduced in the previous section in the case of finite activity jumps can be used to compute the solution of coupled FBSDEs in some particular case of infinite activity jumps. In particular, in the case when $\beta(t, x, e) = \gamma(x) \cdot e$, the jump process J_t has finite variation and its jumps between two consecutive points on the grid can be simulated. In particular, this can be implemented for jump models based on the Brownian subordination, such as the Gamma or Variance-Gamma processes. We present below the results we obtained for the Variance-Gamma model.

Variance Gamma model *(Jumps with infinite activity)* The Variance-Gamma process is a Lévy process with *infinite activity jumps*, where the jumps part J_t have a Variance-gamma law $VG(\bar{\sigma}, \kappa, \theta)$ (see e.g. [MS90]). Its characteristic function is

$$\mathbb{E}[e^{iuJ_t}] = \left(1 - iu\theta\kappa + \frac{1}{2}\bar{\sigma}^2\kappa u^2\right)^{-\frac{t}{\kappa}}.$$

The variance-gamma process can be characterized as a time changed Brownian motion with drift, i.e.

$$J_t = \theta T_t + \bar{\sigma} W_{T_t},$$

where W_t is a standard Brownian motion, $T_t \sim \Gamma(t, \kappa t)$ and θ, κ, $\bar{\sigma}$ are given constants. The intensity measure of a Variance-Gamma process is given by

$$\nu(de) = \frac{\exp\left(\frac{\theta e}{\bar{\sigma}^2}\right)}{\kappa|e|} \exp\left(-\frac{\sqrt{\frac{2}{\kappa} + \frac{\theta^2}{\bar{\sigma}}^2}}{\bar{\sigma}}|e|\right) de$$

Under the risk neutral probability measure $\mathbb{Q}$, we assume that the underlying asset S_t follows the dynamics

$$S_t = S_0 \exp((r + \omega)t + J_t),$$

where

$$\omega = \kappa^{-1} \log\left(1 - \frac{1}{2}\bar{\sigma}^2\kappa - \theta\kappa\right).$$

Similarly to the BS and MJ models, we consider below the following coupled FBSDE system which, under appropriate assumptions on the coefficients, provides the price of an European call option in the VG model:

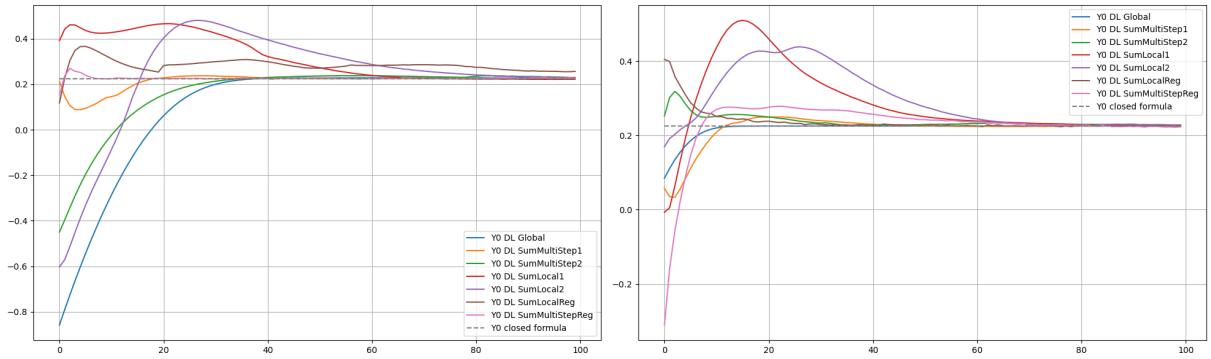
$$\begin{cases} dX_t = X_{t^-}(rdt + \int_{\mathbb{R}^\star}(e^e - 1)\tilde{\mathcal{J}}(dt, de)) + a|Y_t - \bar{u}(t, X_t)|dt, \\ -dY_t = -rY_t dt - \int_{\mathbb{R}^\star} U_t(e)\tilde{\mathcal{J}}(dt, de), \qquad t \in [0, T], \\ X_0 = S_0; \quad Y_T = (X_T - K)^+, \end{cases} \tag{13}$$

where $\tilde{\mathcal{J}}(dt, de)$ is the compensated jump measure associated with the variance Gamma process J_t, and $\bar{u}$ is the analytical solution of the PIDE in the decoupled case.

Model parameters. For this example, we set $T = 1$, $M = 30$ steps, the interest rate $r = 0.1$, the time-scaled Brownian motion drift and volatility $\theta = -0.1$ and $\bar{\sigma} = 0.2$, the variance of the Gamma process $\kappa = 0.1$, the strike price $K = 0.9$, the initial condition $X_0 = 1$, and the coupling linearity coefficient when non-null $a = 0.1$.

2.3.2 Results

On Figure 1, 2, 3, we plot the convergence of the different algorithms for the BS model, the Merton model and the Variance Gamma model, for $a = 0$ (the decoupled case) and a different from 0 (the coupled case). Hence, we plot the evolution of Y_0 through 100 epochs for BS and 120 epochs for MJ and VG. Notice that 100 gradient descents are performed between 2 epochs for BS, MJ and VG.

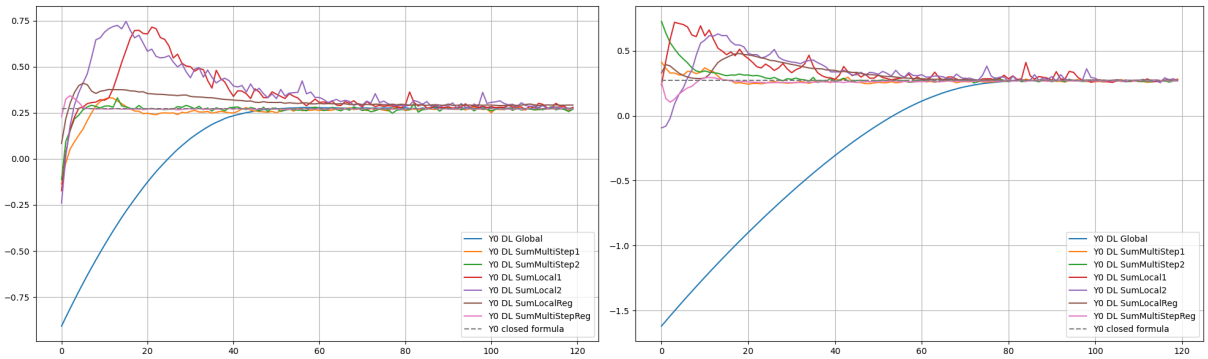


(a) Coupled case ($a = 0.1$)

(b) Decoupled case ($a = 0$)

Figure 1: Convergence of the 7 algorithms in the BS model

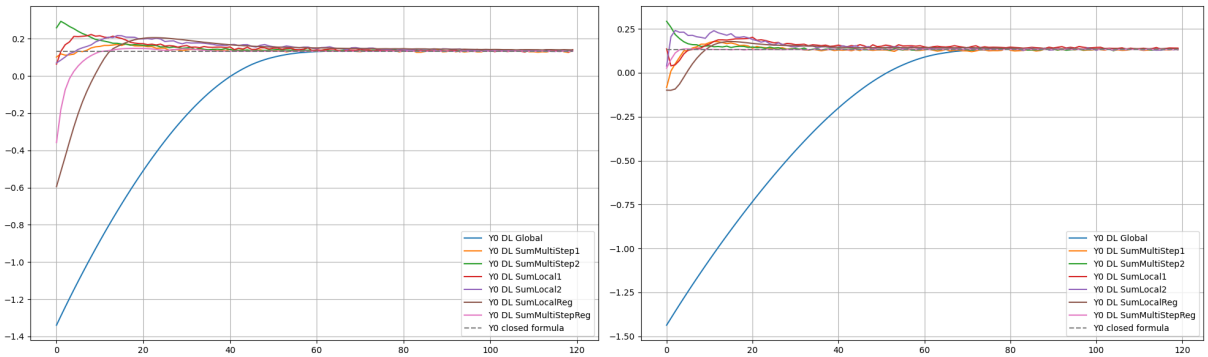
Figure 1 illustrates the convergence of the European call price Y_0 in the Black-Scholes model in both, the coupled and decoupled cases. This figure demonstrates that all methods converge smoothly in the decoupled case. In the coupled system, all methods also converge smoothly except for SumLocalReg, which stagnates between 0.255 and 0.256, instead of converging to the true value 0.225.



(a) Coupled case ($a = 0.1$) (b) Decoupled case ($a = 0$)

Figure 2: Convergence of the 7 algorithms in the Merton model

Figure 2 illustrates the convergence of the value of the European call price Y_0 for the Merton model, in both the coupled and decoupled cases. This model allows us to test the performance of our algorithms in a setting that involves a jump diffusion model with finite activity. It can be observed that all the algorithms converge quickly, requiring only 80 epochs, except for SumLocalReg in the coupled case. Similar to its performance in the BS model, SumLocalReg is very unstable and far from the true value.



(a) Coupled case ($a = 0.1$) (b) Decoupled case ($a = 0$)

Figure 3: Convergence of the 7 algorithms in the Variance Gamma model

Figure 3 illustrates the convergence of the value of the European call price Y_0 for the Variance Gamma model, in both the coupled and decoupled cases. This model allows us to test the performance of our algorithms in a model with pure jumps with infinite activity. We observed that all the algorithms were consistent, stable and relatively quick except for the SumLocalReg algorithm.

To focus on the processing times intrinsic to the training process, we first present the computation times and results for the Merton and Variance Gamma models with the parameter $a = 0$. This removes the additional computation time required for the analytical solution u that is not part of the training process.

Model	DL Methods						
	Global	MultiStep1	MultiStep2	SumLocal1	SumLocal2	SumLocalReg	MultiStepReg
MJ	874s	941s	928s	921s	902s	622s	642s
VG	634s	687s	692s	673s	670s	666s	702s

Table 3: Computation times in seconds for different DL methods after 12000 training steps

As shown in Table 3, the performance of the different deep learning methods for the Merton and Variance Gamma models was evaluated based on the computation time required for 12000 training steps. Overall, the results indicate that the MultiStepReg and SumLocalReg methods were the most time-efficient for the Merton model. Nonetheless, the discrepancies are not significant enough to base our choice on the computation time only. Thus, we present some accuracy and convergence results.

In Table 4, we present the results obtained for $a = 0$ after 12000 training steps.

Model	DL Methods						
	Global	MultiStep1	MultiStep2	SumLocal1	SumLocal2	SumLocalReg	MultiStepReg
MJ	0.271	0.273	0.266	0.276	0.270	0.272	0.267
VG	0.133	0.132	0.137	0.141	0.130	0.135	0.132

Table 4: Y_0 for $a = 0$ and different DL methods. The reference value for the Merton and Variance Gamma models are 0.271 respectively 0.133. The green color corresponds to an error less than 4.10^{-3} and the red color to an error larger than 4.10^{-3}.

In Table 5, we present the results obtained for $a = 0.1$ after 12000 training steps.

Model	DL Methods						
	Global	MultiStep1	MultiStep2	SumLocal1	SumLocal2	SumLocalReg	MultiStepReg
MJ	0.273	0.274	0.269	0.280	0.273	0.292	0.272
VG	0.133	0.136	0.135	0.141	0.135	0.140	0.132

Table 5: Y_0 for $a = 0.1$ and different DL methods. The reference value for the Merton and Variance Gamma models are the same as in Table 4.

2.3.3 Conclusion

The results above show that neural network methods can solve coupled FBSDEs with jumps with finite activity (or a particular class of jumps with infinite activity as explained above) issued from pricing models. After various benchmarks, we observe that

1. The *Global* method is consistent, stable and relatively robust compared to the other methods when it comes to the calibration of the hyper-parameters (especially the learning rate).

2. The first variant *SumLocal1* of the local methods is not very accurate in the decoupled and coupled cases, whereas the regression version *SumLocalReg* is accurate in the decoupled case and faces more difficulties in the coupled case. On the other hand, *SumLocal2* performs well in the decoupled and coupled cases with the fine-tuned hyper-parameters which have an important impact on the accuracy of the local algorithms.

3. All the MultiStep variants *MultiStep1*, *MultiStep2* and the regression version *MultiStepReg*, perform very well conditionally on finding the adequate hyperparameters that depend on the parameters of the models.

4. Finally, *MultiStepReg* provides the best computation speed and a good accuracy without the need to estimate the compensator using Monte Carlo or additional networks which will add extra biases.

3 Application to an MFG model with jumps for smart grids

In this section, we develop a generalized version of the model introduced in [Ala+23] (which is also related to the MFG models presented in [ATM19; MMS19]), which we solve numerically using the deep learning algorithms introduced in Section 2.2. We consider an energy system with n consumers who are linked by a Demand Side Management (DSM) contract, i.e. they agree to diminish, at random times, their aggregated power consumption by a predefined volume during a predefined duration. Their failure to deliver the service is penalised via the difference between the sum of the n power consumptions and the contracted target. The jumps are supposed to come from a Cox process with a *stochastic intensity process*, in contrast with [Ala+23] where the intensity is only assumed to be constant. From a modeling perspective, this generalization is important since it allows to capture the dependence of the intensity on e.g. the aggregated consumption, which implies that the jumps arrive with a higher rate when the demand is at its peak. This is when the demand is at its peak that the power system is more likely to benefit from a reduction of this power demand so that it reduces the cost of production. Furthermore, compared to [Ala+23] where the contracted target is a constant, we consider here the general case of a *stochastic target process*. When $n \to \infty$, the problem can be written in terms of a Mean-Field Game model with interaction on the control.

3.1 Extended MFG model for Demand Side Management with Cox process

In this subsection, we first briefly describe the model in the setting of a finite population of players, and then present the MFG formulation.

Model with n-consumers. We assume that there are two types of consumers (*active consumers* and *standard consumers*). An *active consumer* $i = 1, \dots, n$ enters a demand side management contract (DSM) and is characterized by two state variables (Q^i, S^{α^i}). The variable Q_t^i denotes the instantaneous electricity consumption of consumer i at time t, representing the required electricity volume. *Active consumers* can deviate from their natural power demand by an amount α_t^i, their total instantaneous consumption being $(Q_t^i + \alpha_t^i)dt$. In case the instantaneous effort $\alpha_t^i > 0$ (resp. < 0), the consumer is anticipating (resp. postponing) specific activities which require energy, which implies that consumption is increased (resp. decreased) compared to the natural demand. The total deviation in consumption from natural power demand up to time t is represented by $S_t^{\alpha^i}$. The second type of consumers is represented by the *standard consumers*, for $i = n+1, \dots, n+n'$, who do not optimize their consumption. They are characterized

by a single state variable $Q^{i,st}$ corresponding to the instantaneous consumption of consumer i at time t. More precisely, the dynamics of the consumption (resp. total deviation in consumption) for consumer $i = 1, \dots, n$ with DSM contract are given by

$$\begin{aligned} dQ_t^i &= \mu(t, Q_t^i)dt + \sigma(t, Q_t^i)dW_t^i + \sigma^0(t, Q_t^i)dW_t^0, \quad Q_0^i = q_0^i, \\ dS_t^{\alpha^i, i} &= \alpha_t^i dt, \quad S_0^i = s_0^i, \end{aligned}$$

while those for any standard consumer $i = n+1, \dots, n+n'$ are

$$dQ_t^{st,i} = \mu^{st}(t, Q_t^{st,i})dt + \sigma^{st}(t, Q_t^{st,i})dW_t^i + \sigma^{st,0}(t, Q_t^{st,i})dW_t^0, \quad Q_0^{st,i} = q_0^{st,i}.$$

The processes $W_t^0, W_t^1, \dots, W_t^{n+n'}$ appearing above are assumed to be independent Brownian motions, and the functions $\mu, \mu^{st}, \sigma, \sigma^{st}, \sigma^0, \sigma^{st,0}$ are such that the above stochastic differential equations admit strong solutions.

The demand side management contract incorporates *real-time pricing* and an *interruptible load* feature. First, *real-time pricing* refers to the fact that consumers are charged at a spot price p which depends on the total consumption, having the role to incentivize the active consumers to reduce their consumption when it becomes too high. The associated *power cost* c_t^i is a function of the total consumption of the standard consumers and those with a DSM contract.

The interruptible load part of the contract is described as follows. At random times indicated by the Transmission System Operator (TSO) in case of supply-demand imbalance, the total consumption of the active consumers $\sum_i (Q_t^i + \alpha_t^i)$ has to match a target process α_t^{tg}, which could represent e.g. a fraction of the usual consumption. The target is maintained for a specific duration, and each agent is penalized if the total response differs from the required level of demand. The corresponding *divergence cost* d^i is expressed as a function of the total consumption of the *active consumers*.

The DSM contract also includes: an *inconvenience cost* g (associated with the efforts made by consumers to control their consumption, which increases with the instantaneous effort α^i and the accumulated deviations S^{α^i}), *a demand charge cost* l and a *terminal cost function* h (which penalizes any excess or shortfall of energy consumption during the period, as it indicates that the agent did not acquire the exact amount of energy needed during the specified time frame).

MFG formulation. To present the model in the MFG setting, we first introduce the probabilistic setup.

Probabilistic setup. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a complete probabilistic space. We assume that all stochastic processes are defined on a finite time horizon $[0, T]$ with $T > 0$.

Suppose W^0 is a Brownian motion on this space on $[0, T]$ and $\mathbb{G}^0 \triangleq (\mathcal{G}_t^0)_{t\in[0,T]}$ is the filtration generated by W^0 augmented by the $\mathbb{P}$-null sets. Let N^0 be a *doubly stochastic Poisson process* (or a *Cox* Process) with a $\mathbb{G}^0$-predictable non-negative intensity $\lambda^0 := (\lambda_t^0)_{t\in[0,T]}$. In relation to N^0, we denote by $\mathbb{D}^0 \triangleq (\mathcal{D}_t^0)_{t\in[0,T]}$ the filtration generated by the *Cox* Process N^0 augmented with the $\mathbb{P}$-null sets. Let $\mathbb{F}^0 = (\mathcal{F}_t^0)_{t\in[0,T]}$ denote the filtration $\mathbb{F}^0 = \mathbb{G}^0 \vee \mathbb{D}^0$, i.e. the smallest filtration containing $\mathbb{G}^0$ and $\mathbb{D}^0$. In our setting, $\mathbb{F}^0$ plays the role of the *common noise filtration*.

Assume that $\mathbb{E}[\int_0^T \lambda_s^0 ds] < \infty$ for all $t \in [0, T]$, from which it follows that the *compensated Poisson process* $\tilde{N}_t^0 := N_t^0 - \int_0^t \lambda_s^0 ds$ is a $\mathbb{F}^0$-martingale.

We also introduce the Brownian motions W and $\bar{W}$ (representing the *idiosyncratic noises* of the active and standard consumers), which are independent of W^0 and N^0. Let $\mathbb{G} \triangleq (\mathcal{G}_t)_{t\in[0,T]}$ denote the filtration generated by W and $\bar{W}$, augmented with the $\mathbb{P}$-null sets. We denote by $\mathbb{F}^W$ the (completed) filtration

generated by W. Let (s_0, q_0) be two random variables independent of all the above processes. Finally, let $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$ be the smallest filtration containing $\mathbb{F}^0$, $\mathbb{G}$, and the information generated by (s_0, q_0).

Representative consumer with DSM contract and representative standard consumer. The *representative consumer* involved in the DSM contract is characterized by two state variables (Q, S^α), with Q_t representing the instantaneous volume of electricity needed at time t and S_t^α the accumulated deviation of electricity from the natural consumption, which is controlled by a control process (α_t). The dynamics of the state variables of the representative consumer with *DSM* contract are given by

$$\begin{cases} dQ_t &= \mu(t, Q_t)dt + \sigma(t, Q_t)dW_t + \sigma^0(t, Q_t)dW_t^0, \quad Q_0 = q_0, \\ dS_t^\alpha &= \alpha_t dt, \quad S_0 = s_0, \end{cases} \tag{14}$$

where (α_t) represents the instantaneous effort.

The *representative standard consumer* is characterized by only one state variable Q^{st} representing their usual consumption. The dynamics of the standard consumption is then given by

$$dQ_t^{st} = \mu^{st}(t, Q_t^{st})dt + \sigma^{st}(t, Q_t^{st})d\bar{W}_t + \sigma^{st,0}(t, Q_t^{st})dW_t^0, \quad Q_0^{st} = q_0^{st}, \tag{15}$$

where all the above coefficients are continuous in (t, x) and Lipschitz continuous with respect to x, uniformly in t.

Optimization problem consumer with DSM contract and MFG equilibrium. As explained in the description of the n-player model, the demand side management model considered in this paper includes *dynamic pricing* and *an interruptible load* feature. To describe the interruptible load part of the contract, let $(\alpha_t^{tg})_{t\in[0,T]}$ be a given $\mathbb{G}^0$-adapted process. At random times indicated by the operator system in charge of the production-consumption balance, the aggregated power deviation of the consumption has to match the stochastic contracted target process α_t^{tg} for a predefined time period $\theta > 0$. The random times correspond to the jump times of the Cox process (N_t^0). In case the target is not achieved during the period θ, then the representative consumer is penalized. We introduce the process R which measures the time since the last DSM jump occurred. Thus, the dynamics of R are given by

$$dR_t = dt - R_{t^-}dN_t^0, \quad R_0 = 2\theta.$$

Fix $\xi = (\xi_t)_{t\in[0,T]}$ a $\mathbb{F}^0$-adapted process which represents a *predetermined* power deviation and $\alpha \in \mathcal{A}$, where $\mathcal{A}$ is the set of all real-valued $\mathbb{F}^W \vee \mathbb{G}^0$-progressively measurable processes α such that $\mathbb{E}[\int_0^T \alpha_t^2 dt] < +\infty$ and $\mathbb{E}[|\alpha_\tau|\mathbb{1}_{\tau<\infty}] < +\infty$ for all $\mathbb{F}^0$-stopping times τ with values in $[0, T] \cup \{+\infty\}$. This set is called the set of *admissible* controls. The divergence cost is then defined as follows:

$$d_t^{\alpha,\xi} = (Q_t + \alpha_t - \alpha_t^{tg}) f\left(\mathbb{E}[Q_t|\mathcal{F}_t^0] + \xi_t - \alpha_t^{tg}\right) J_t^\theta,$$

where $J_t^\theta = \mathbb{1}_{R_t\leq\theta}$ (i.e. J_t^θ is equal to one during interruptible load contract activation and zero otherwise) and f is a convex growing function such as $f(0) = 0$.

The second component of the *DSM contract* is represented by the power cost $c_t^{\alpha,\xi}$, which defines the *dynamic pricing rule* and is defined as

$$c_t^{\alpha,\xi} = (Q_t + \alpha_t)p\left(\pi\mathbb{E}[Q_t^{st}|\mathcal{F}_t^0] + (1-\pi)(\mathbb{E}[Q_t|\mathcal{F}_t^0] + \xi_t)\right),$$

where p represents the spot price functional of the power system at which the consumers are charged, and π represents the proportion of standard consumers with respect to DSM consumers in the total population.

Finally, we introduce the *inconvenience cost* $g(\alpha_t, S_t^\alpha, Q_t)$ (with g convex in α_t and S_t^α), the cost l representing the demand charge component of the retail tariff structure, and the *terminal cost* $h(S_T^\alpha)$.

We can now formulate the MFG problem. For a fixed process ξ, the active consumer is optimizing the following functional:

$$J(\alpha;\xi) = \mathbb{E}\left[\int_0^T \left\{g(\alpha_t, S_t^\alpha, Q_t) + l(Q_t+\alpha_t) + c_t^{\alpha,\xi} + d_t^{\alpha,\xi}\right\} dt + h(S_T^\alpha)\right].$$

Therefore, the optimization problem of the representative consumer can be then written as follows

$$V^{MFG}(\xi) = \inf_{\alpha\in\mathcal{A}} J^{MFG}(\alpha;\xi) \tag{16}$$

Definition 3.1 (Mean-field Nash Equilibrium)**.** The solution $\alpha^\star$ to problem (16) is called a mean field *Nash equilibrium* if $\mathbb{E}[\alpha_t^\star|\mathcal{F}_t^0] = \xi_t$ a.s. for all $0 \le t \le T$.

Remark 5. *Notice that, in contrast to [Ala+23] where the target α^{tg} is only considered to be a constant, we consider here a $\mathbb{G}^0$-adapted target process $\alpha^{tg} = (\alpha_t^{tg})_{t\in[0,T]}$. We also assume that the jump times correspond to the ones of a Cox process (time non-homogeneous Poisson process), compared to [Ala+23] where it is supposed that they come from a Poisson process.*

3.2 Characterization of the MFG equilibrium with Cox Process

In this Section, we first provide a characterization of the MFG equilibrium in a general setting, and then focus on the linear-quadratic model. We introduce the following sets, which will be used throughout the rest of the paper:

- $\mathcal{S}^2$ is the set of $\mathbb{F}$-adapted càdlàg real-valued processes Y such that $\mathbb{E}\left[\sup_{0\le t\le T}|Y_t|^2\right] < +\infty$.
- $\mathcal{H}^2$ is the set of $\mathbb{F}$-predictable real-valued processes q such that $\|q\|^2 := \mathbb{E}[\int_0^T |q_t|^2 dt] < +\infty$.
- $\mathcal{H}^2_{\lambda^0}$ is the set of $\mathbb{F}$-predictable real-valued processes ν^0 such that $\|\nu^0\|^2_{\lambda^0} := \mathbb{E}[\int_0^T |\nu_t^0|^2\lambda_t^0 dt] < +\infty$.
- $L^2(\mathcal{F}_T)$ is the set of $\mathcal{F}_T$-measurable real-valued random variables ξ such that $\mathbb{E}[|\xi|^2] < +\infty$.

In the sequel, given a $\mathcal{B}([0,T])\otimes\mathcal{F}$-measurable process X such that $E[|X_\tau|1_{\tau<\infty}] < \infty$ for all $\mathbb{F}^0$-stopping times τ with values in $[0,T]\cup\{+\infty\}$, we will denote by $\widehat{X}$ the optional projection of X with respect to the filtration $\mathbb{F}^0$, i.e. $\widehat{X}$ is the unique (up to indistinguishability) $\mathbb{F}^0$-optional process such that $\widehat{X}_\tau 1_{\tau<\infty} = \mathbb{E}\left[X_\tau 1_{\tau<\infty}|\mathcal{F}_\tau^0\right]$ a.s. for all $\mathbb{F}^0$-stopping times τ with values in $[0,T]\cup\{+\infty\}$ (cf. Section 2 in [BY78]).

Assumption 3.1. *We make the following assumptions:*

- *g, l and h have at most quadratic growth and are strictly convex.*
- *p and f have at most linear growth.*
- *g, p, f, l and h are differentiable.*

We now give the following characterization of a MFG equilibrium.

Theorem 3.1 (Characterization of the mean-field game equilibrium)**.** *Let $\hat{\xi}$ be a given $\mathbb{F}^0$-adapted real valued process and $x_0 = (s_0, q_0, q_0^{st})$ be a random vector independent of $\mathbb{F}^0$. Assume that $\alpha \to J^{MFG}(\alpha; \hat{\xi})$ is strictly convex. If there exists a control $\alpha^\star \in \mathcal{A}$ which minimizes $\alpha \to J^{MFG}(\alpha; \hat{\xi})$ and if $(S^{\alpha^\star}, Q, Q^{st})$ is the state process associated to the initial condition x_0, optimal control $\alpha^\star$ and the dynamics* (14)-(15)*, then there exists a unique solution $(Y^\star, q^{0,\star}, q^\star, \nu^{0,\star}) \in \mathcal{S}^2 \times (\mathcal{H}^2)^2 \times \mathcal{H}^2_{\lambda^0}$ of the following BSDE with jumps:*

$$\begin{cases} -dY_t^\star = \partial_\alpha g(\alpha_t^\star, S_t^{\alpha^\star}, Q_t)dt - q_t^{0,\star} dW_t^0 - q_t^{st,\star} d\bar{W}_t - \nu_t^{0,\star} d\tilde{N}_t^0, \\ Y_T^\star = \partial_x h(S_T^{\alpha^\star}), \end{cases} \tag{17}$$

satisfying the coupling condition

$$\partial_\alpha g(\alpha_t^\star, S_t^{\alpha^\star}, Q_t) + \partial_\alpha l(Q_t + \alpha_t^\star) + p(\pi \widehat{Q}_t^{st} + (1-\pi)(\widehat{Q}_t + \hat{\xi}_t)) + Y_t^\star + J_t^\theta f(\widehat{Q}_t + \hat{\xi}_t - \alpha_t^{tg}) = 0 \tag{18}$$

Conversely, assume that there exists $(\alpha^\star, S^{\alpha^\star}, Y^\star, q^{0,\star}, q^\star, \nu^{0,\star}) \in \mathcal{A} \times (\mathcal{S}^2)^2 \times (\mathcal{H}^2)^2 \times \mathcal{H}^2_{\lambda^0}$ satisfying the FBSDE (17) *and the coupling condition* (18)*, then $\alpha^\star$ is the optimal control minimizing $\alpha \to J^{MFG}(\alpha; \hat{\xi})$ and $S^{\alpha^\star}$ is the optimal trajectory.*

If additionally $\widehat{\alpha}_t^\star = \hat{\xi}_t$ a.s for all $t \in [0,T]$, then $\alpha^\star$ is a Mean-field equilibrium.

Proof. Under Assumption 3.1 and using similar arguments as in [DQS17] to prove existence and uniqueness results for BSDEs driven by Cox processes, we conclude that the BSDE defined in the theorem is well-posed. Using this result, the proof follows the same steps as in Theorem 3.1 in [Ala+23] , and we therefore omit it.

□

Semi-explicit representation of the MFG equilibrium in the linear quadratic case We shall provide here a semi-explicit characterization of the equilibrium in the linear-quadratic setting, which is ensured by the following assumption.

Assumption 3.2. *Let $(\chi_t)_{t\in[0,T]}$ and $(\chi_t^{st})_{t\in[0,T]}$ be two continuous deterministic processes. We suppose that the following assumptions are satisfied:*
1. $\mu(t,q) = \mu(\chi_t - q)$, $\mu^{st}(t,q) = \mu^{st}(\chi_t^{st} - q)$, $\sigma(t,q) = \sigma$, $\sigma^{st}(t,q) = \sigma^{st}$, $\sigma^{st,0}(t,q) = \sigma^{st,0}$, and $\sigma^0(t,q) = \sigma^0$, with $\mu, \mu^{st}, \sigma, \sigma^0, \sigma^{st} > 0$.
2. $g(a,s,q) = \frac{A}{2}a^2 + \frac{C}{2}s^2$ with $A, C > 0$.
3. $l(x) = \frac{K}{2}x^2$ with $K \geq 0$.
4. $f(a) = f_0 + f_1 a$ with $f_i \in \mathbb{R}, i = 0, 1$ and $f_1 \geq 0$.
5. $p(q) = p_0 + p_1 q$ with $p_0 \in \mathbb{R}$, and $p_1 > 0$.
6. $h(s) = h_0 + h_1 s + \frac{h_2}{2}s^2$, with $h_i \in \mathbb{R}, i = 0, 1, 2$ and $h_2 \geq 0$.

Following the approach used in [Ala+23] in the particular case of a Poisson process, in the linear-quadratic setting we look for solutions taking the form:

$$\widehat{Y}_t = \bar{\phi}_t S_t^{\widehat{\alpha}} + \bar{\psi}_t \text{ and } Y_t = \phi_t S_t^{\alpha} + \psi_t,$$

with $(\bar{\phi}_t, 0, \widehat{\xi}_t^0, \widehat{\xi}_t^{0,N})$, $(\bar{\psi}_t, \widehat{\eta}_t^0, 0, \widehat{\eta}_t^{0,N})$, $(\phi_t, 0, 0, 0)$ and $(\psi_t, \eta_t^0, \eta_t, \eta_t^{0,N})$ the unique solutions in $\mathcal{S}^2 \times (\mathcal{H}^2)^2 \times \mathcal{H}^2_{\lambda^0}$ of the following BSDEs driven by a Cox process:

$$d\phi_t = \left(-C + \frac{1}{A+K}\phi_t^2\right) dt, \quad \phi_T = h_2,$$

$$
\begin{aligned}
d\psi_t &= \frac{\phi_t}{A+K}\Big[KQ_t + p_0 + \pi p_1 \widehat{Q}_t^{st} + ((1-\pi)p_1 + K)(\widehat{Q}_t + \widehat{\alpha}_t) + \\
&\quad J_t^\theta(f_0 + f_1(\widehat{Q}_t + \widehat{\alpha}_t - \alpha_t^{tg})) + \psi_t\Big]\,dt + \eta_t^0 dW_t^0 + \eta_t dW_t + \eta_t^{0,N} d\tilde{N}_t^0, \quad \psi_T = h_1, \\
d\bar{\phi}_t &= \left(-C + \frac{1}{K_t^\theta}\bar{\phi}_t^2\right) + \widehat{\xi}_t^0 dW_t^0 + \widehat{\xi}_t^{0,N} d\tilde{N}_t^0, \quad \bar{\phi}_T = h_2, \\
d\bar{\psi}_t &= \frac{\bar{\phi}_t}{K_t^\theta}\Big[p_0 + \pi p_1 \widehat{Q}_t^{st} + ((1-\pi)p_1 + K)\widehat{Q} + J_t^\theta(f_0 + f_1(\widehat{Q}_t - \alpha_t^{tg})) + \bar{\psi}_t\Big]\,dt \\
&\quad + \widehat{\eta}_t^0 dW_t^0 + \widehat{\eta}_t^{0,N} d\tilde{N}_t^0, \quad \bar{\psi}_T = h_1,
\end{aligned}
$$

where $K_t^\theta = A + K + (1-\pi)p_1 + f_1 J_t^\theta$.

The wellposedness of the above BSDEs follows by an adaptation of the theorems provided in [DQS17]. By using the ansatz and replacing it in the projected coupling condition, we obtain:

$$
\widehat{\alpha}_t = -\frac{1}{K_t^\theta}\left(p_0 + \pi p_1 \widehat{Q}^{st} + ((1-\pi)p_1 + K)\widehat{Q}_t + \widehat{Y}_t + (f_0 + f_1(\widehat{Q}_t - \alpha_t^{tg}))J_t^\theta\right). \tag{19}
$$

Finally, by using the expression of $\widehat{\alpha}$ and again the ansatz and the coupling condition, we finally obtain that the *MFG equilibrium* α admits the following representation:

$$
\begin{aligned}
\alpha_t = -\frac{1}{A+K}\Big(&KQ_t + p_0 + \pi p_1 \widehat{Q}^{st} + ((1-\pi)p_1 + K)(\widehat{Q}_t + \widehat{\alpha}_t) + Y_t + (f_0 + f_1(\widehat{Q}_t + \widehat{\alpha}_t \\
&- \alpha_t^{tg}))J_t^\theta\Big).
\end{aligned} \tag{20}
$$

3.3 Aggregator problem and Price of Anarchy

In this part, we consider the point of view of an aggregator and characterize his optimal strategy, as well as discuss the related price of anarchy.

Aggregator problem We now introduce the following mean-field control problem of an aggregator who plays the role of a central planner who coordinates all the DSM consumers in the system, without taking into account the non-active consumers. The associated value function of the aggregator is given by

$$
V^{MFC^{agg}} = \inf_{\alpha \in \mathcal{A}} \mathbb{E}\left[\int_0^T \left\{g(\alpha_t, S_t^\alpha, Q_t) + l(Q_t + \alpha_t) + c_t^{\alpha,\widehat{\alpha}} + d_t^{\alpha,\widehat{\alpha}}\right\} dt + h(S_T^\alpha)\right].
$$

The solution to this optimization problem is called the MFC^{agg} optimal control. Using a similar proof to the one of Theorem 3.1, we have the following characterization of the optimal control.

Theorem 3.2 (Characterization of the aggregator's mean field control)**.** *Let $x_0 = (s_0, q_0, q_0^{st})$ be a random vector independent of $\mathbb{F}^0$. Assume that the map $\alpha \mapsto J^{MFC}(\alpha)$ is strictly convex. If there exists a control $\alpha^\star \in \mathcal{A}$ which minimizes the map $\alpha \mapsto J^{MFC}(\alpha)$ and if $(S^{\alpha^\star}, Q, Q^{st})$ is the state process associated to the initial condition x_0, control $\alpha^\star$ and the dynamics (14)-(15), then there exists a unique solution $(Y^\star, q^{0,\star}, q^\star, \nu^{0,\star}) \in \mathcal{S}^2 \times (\mathcal{H}^2)^2 \times \mathcal{H}^2_{\lambda^0}$ of the BSDE with jumps*

$$
\begin{cases}
-dY_t^\star = \partial_x g(\alpha_t^\star, S_t^{\alpha^\star}, Q_t)dt - q_t^{0,\star} dW_t^0 - q_t^\star dW_t - \nu_t^{0,\star} d\widetilde{N}_t^0, \\
Y_T^\star = \partial_x h(S_T^{\alpha^\star}),
\end{cases} \tag{21}
$$

satisfying the coupling condition

$$\begin{aligned}\partial_\alpha g(\alpha_t^\star, S_t^{\alpha^\star}, Q_t) + \partial_\alpha l(Q_t + \alpha_t^\star) + p\left(\pi\widehat{Q}_t^{st} + (1-\pi)(\widehat{Q}_t + \widehat{\alpha}_t^\star)\right) \\ +(\widehat{Q}_t + \widehat{\alpha}_t^\star)\partial_\alpha p(\pi\widehat{Q}_t^{st} + (1-\pi)(\widehat{Q}_t + \widehat{\alpha}_t^\star)) \\ +Y_t^\star + J_t^\theta f(\widehat{Q_t} + \widehat{\alpha}_t^\star - \alpha^{tg}) + J_t^\theta(\widehat{Q_t} + \widehat{\alpha}_t^\star - \alpha^{tg})\partial_\alpha f(\widehat{Q_t} + \widehat{\alpha}_t^\star - \alpha_t^{tg}) = 0, \end{aligned} \tag{22}$$

with $\widehat{\alpha}^\star$ the optional projection of $\alpha^\star$ with respect to $\mathbb{F}^0$. Conversely, assume that there exists $\left(\alpha^\star, S^{\alpha^\star}, Y^\star, q^{0,\star}, q^\star, \nu^{0,\star}\right) \in \mathcal{A} \times (\mathcal{S}^2)^2 \times (\mathcal{H}^2)^2 \times \mathcal{H}^2_{\lambda^0}$ satisfying the coupling condition (22), as well as the FBSDE (14)-(21), then $\alpha^\star$ is the optimal control minimizing the map $\alpha \mapsto J^{MFC}(\alpha)$ and $S^{\alpha^\star}$ is the optimal trajectory.

Remark 6. *As observed in [Ala+23], by comparing the coupling conditions (18) and (22), the optimal control for the MFC^{agg} problem in the linear quadratic setting with pricing rules $p^{MFC^{agg}}(\pi\widehat{Q}^{st} + (1-\pi)Q) = p_0 + p_1(\pi\widehat{Q}^{st} + (1-\pi)Q)$ and $f^{MFC^{agg}}(Q) = f_0 + f_1 Q$ corresponds to the MFG equilibrium for the problem with pricing rules $p^{MFG}(\pi\widehat{Q}^{st} + (1-\pi)Q) = p_0 + 2p_1(1-\pi)Q + p_1\pi\widehat{Q}^{st}$ and $f^{MFG}(Q) = f_0 + 2f_1 Q$.*

Price of Anarchy The price of anarchy is defined as the ratio of a worst case social cost computed for a mean field game equilibrium to the optimal social cost as computed by a central planner.

For our problem, the expression for the price of anarchy takes the following form:

$$\text{PoA} = \frac{V^{MFG}(\widehat{\alpha}^\star)}{V^{MFC^{agg}}},$$

where $\alpha^\star$ is the MFG Nash equilibrium.

3.4 Deep learning algorithms for the MFG and MFC problem

In this section, we design several numerical algorithms to compute in the linear-quadratic setting the MFG equilibrium and the mean-field optimal control for the aggregator's problem. The algorithms are based on the machine learning solvers introduced in the first part of the paper and extended to the case of a time-inhomogeneous Poisson process with *stochastic intensity* of jumps.

Characterization of the MFG equilibria via a multi-dimensional coupled FBSDE with jumps Using the results from the previous section, the MFG equilibria in the linear-quadratic case can be

expressed through the following multi-dimensional fully-coupled FBSDE system:

$$\textbf{(MFG)}\begin{cases} dQ_t = \mu(\chi_t - Q_t)dt + \sigma dW_t + \sigma^0 dW_t^0, \\ d\widehat{Q}_t = \mu(\chi_t - \widehat{Q}_t)dt + \sigma^0 dW_t^0, \\ dQ_t^{st} = \mu^{st}(\chi_t^{st} - Q_t^{st})dt + \sigma^{st} d\bar{W}_t + \sigma^{st,0} dW_t^0, \\ d\widehat{Q}_t^{st} = \mu^{st}(\chi_t^{st} - \widehat{Q}_t^{st})dt + \sigma^{st,0} dW_t^0, \\ dR_t = dt - R_{t^-} dN_t^0, \\ dS_t^{\alpha^\star} = -\frac{1}{A+K}\Big(KQ_t + p_0 + \pi p_1 \widehat{Q}^{st} + ((1-\pi)p_1 + K)(\widehat{Q}_t + P(t,\widehat{Q}_t,\widehat{Q}_t^{st},\widehat{Y}_t,R_t)) + Y_t + (f_0 + f_1(\widehat{Q}_t \\ + P(t,\widehat{Q}_t,\widehat{Q}_t^{st},\widehat{Y}_t,R_t) - \alpha_t^{tg}))\mathbb{1}_{R_t \le \theta}\Big)\, dt, \\ dS_t^{\widehat{\alpha}^\star} = P(t,\widehat{Q}_t,\widehat{Q}_t^{st},\widehat{Y}_t,R_t)dt, \\ -dY_t = CS_t^{\alpha^\star} dt - q_t^0 dW_t^0 - q_t dW_t - \nu_t^0 d\tilde{N}_t^0, \\ -d\widehat{Y}_t = CS_t^{\widehat{\alpha}^\star} dt - \widehat{q}_t^0 dW_t^0 - \widehat{\nu}_t^0 d\tilde{N}_t^0, \\ Q_0 = q_0, \quad Q_0^{st} = q_0^{st}, \quad R_0 = 2\theta, \quad S_0^{\alpha^\star} = s_0, \quad Y_T = h_1 + h_2 S_T^{\alpha^\star}, \\ \widehat{Q}_0 = q_0, \quad \widehat{Q}_0^{st} = q_0^{st}, \quad S_0^{\widehat{\alpha}^\star} = s_0, \quad \widehat{Y}_T = h_1 + h_2 S_T^{\widehat{\alpha}^\star}, \end{cases} \tag{23}$$

where

$$P(t,\widehat{Q}_t,\widehat{Q}_t^{st},\widehat{Y}_t,R_t) := -\frac{1}{A+K+(1-\pi)p_1 + f_1\mathbb{1}_{R_t\le\theta}}\Big(p_0 + \pi p_1\widehat{Q}^{st} + ((1-\pi)p_1+K)\widehat{Q}_t + \widehat{Y}_t \\ +(f_0 + f_1(\widehat{Q}_t - \alpha_t^{tg}))\mathbb{1}_{R_t\le\theta}\Big).$$

We are thus led to solve a fully-coupled multi-dimensional FBSDE driven by a doubly stochastic Poisson process (which admits an unique solution, by using the results from the part on the *Characterization of the MFG equilibrium in the linear quadratic case* from subsection 3.2). For a generic fully-coupled system of FBSDEs driven by a doubly Poisson process, the discretized version takes the form

$$\begin{cases} X_{i+1}^\pi = X_i + b(t_i, X_i^\pi, Y_i)\Delta t_i + \sigma(t_i, X_i^\pi)\Delta W_i + \sigma^0(t_i, X_i^\pi)\Delta W_i^0 + \beta(t_i, X_i^\pi)dN_i^0, \\ Y_{i+1}^\pi \approx Y_i^\pi - f(t_i, X_i^\pi, Y_i^\pi)\Delta t_i + Z_i^\pi \Delta W_i + Z_i^{0,\pi}\Delta W_i^0 + U_i^\pi dN_i^0 - U_i^\pi \lambda_i^0 \Delta t_i, \\ X_0^\pi = \xi, \quad Y_M^\pi = g(X_M^\pi), \\ i = 0, \cdots, M-1. \end{cases} \tag{24}$$

Notice that in this setting, we do not require specific methods to estimate the compensator as it is directly given in this model. Thus, there is no need to test the two variants for the *Sumlocal* and *SumMultiStep* methods when comparing the deep learning solvers.

Furthermore, we make the following assumption on the intensity λ^0:

Assumption 3.3. *There exists a continuous function* $\bar{\lambda} : \mathbb{R} \mapsto \mathbb{R}$ *such that*

$$\lambda_t^0 = \bar{\lambda}(\widehat{Q}_t). \tag{25}$$

Finally, in view of Remark 6, the computation of the mean-field optimal control of the aggregator can be done through the same multi-dimensional fully-coupled FBSDE system (23), but with different coefficients.

Numerical results and comparison between the deep-learning solvers. We shall now perform a detailed analysis of the convergence and stability of the five different algorithms to solve the FBSDE system associated to the computation of the MFG equilibria. To do so, we first set the hyper-parameters.

Parameter	value
m	20
L	2
NbTraining	10000

Parameter	value
B	64
lRate	0.01/0.007
σ_a	tanh

The compensator having an analytical form, we have adapted the algorithms and have taken batches of small sizes. In contrast to [Ala+23], we consider a random intensity of jumps (λ_t^0) and a target process (α_t^{tg}) which are given by

$$\lambda_t^0 = e^{-\frac{\gamma}{2}}(e^{\gamma \widehat{Q}_t} - 1), \quad \alpha_t^{tg} = \beta \mathbb{E}\left[\widehat{Q}_t\right].$$

The other parameters associated to the model are inspired from [Ala+23] and are given below:

Parameter	value
T	2 days
n_{steps}	96 half-hours
A	150
C	80
K	50
χ^{st}	χ
$\sigma^0 = \sigma^{st,0}$	0.1

Parameter	value
σ	0.3
σ^{st}	0
μ	5
$h_0 = h_1$	0
h_2	600
θ	0.12 hours
s_0	0

Parameter	value
f_0	0
f_1	10000
p_0	6.16 €/MWh
p_1	87.43 €/MWh^2
$q_0 = q_0^{st}$	χ_0
γ	30
β	0.8

The function (χ_t) corresponds to the consumption seasonality observed from the data from [Ala+23].

We shall now compare the initial values of the backward components through each epoch. Notice that, between two epochs, 100 stochastic gradient descents are performed.

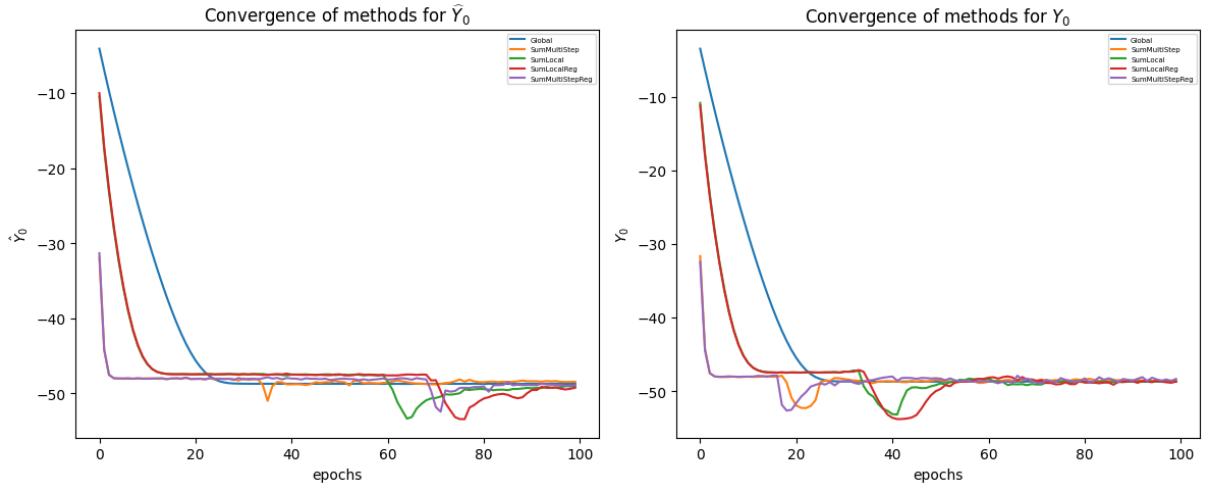


Figure 4: Convergence of the 5 algorithms in the MFG model

According to various benchmarks and the convergence results in Figure 4, it is obvious that the comparison results are similar to the pricing models. The *Global* method is the most stable one and provides a good approximation with a large learning rate which makes up for the problem of initializing low values. *MultiStep* method and its regression version *MultiStepReg* converge after few epochs with a learning rate considerably lower than the one used in the Global method. They present a good trade-off between convergence speed and stability. Finally, *SumLocal* method and its regression version performed poorly in terms of stability as is obvious from the figures above.

Remark 7. *In the particular case of a Poisson process with a constant intensity and a constant consumption rate target α^{tg}, we have compared our results with the ones obtained in [Ala+23] (which were obtained by combining a tree approximation of the martingales, as in e.g. [DL16a; DL16b], and the Monte Carlo method). We observed that our results were coherent with the ones provided in [Ala+23].*

3.5 Interpretation of numerical results from a modelling perspective

In this Section, we provide an economic interpretation of our numerical results, which are computed using the *Global method*. The results are illustrated on two typical customers whose power consumption are represented in Figure 5. Consumer 2 shows a typical power consumption profile with two peaks of consumption in the morning and in the evening. Consumer 2 needs more electricity during the first day than the consumers' average consumption, whereas Consumer 1 consumes very little during the first morning. As expected, the intensity of jumps is very high when the consumption is at its highest level.

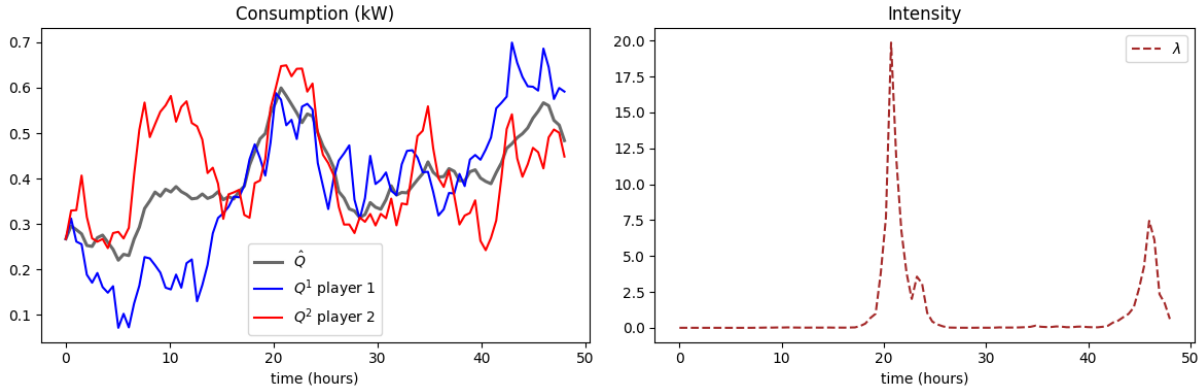


Figure 5: Trajectories over 48 hours of the consumption for 2 different consumers in kW (upper figure left) and the common intensity of jumps for the divergence costs (upper figure right).

The following results present how these two typical consumers optimize their consumption when two activations of the DSM contract happen following the DSM activation scenario presented in Figure 6 .

The illustrations show that the consumers react as expected: when they are exposed to dynamic pricing only (no activation of jump DSM), they smooth their consumption over the period as illustrated in Figure 7 (b). When they are exposed to divergence cost only, their average consumption perfectly matches the random target α^{tg}, whereas the individual consumption of the consumers can differ from the target.

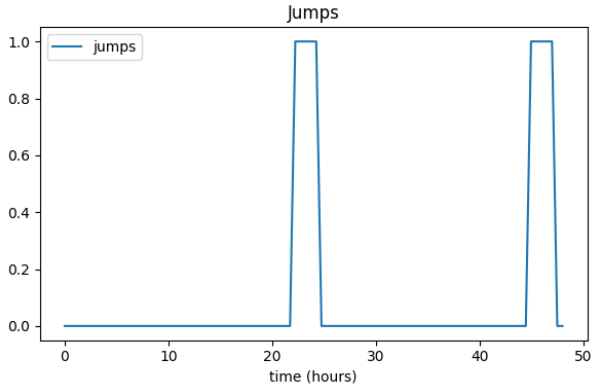
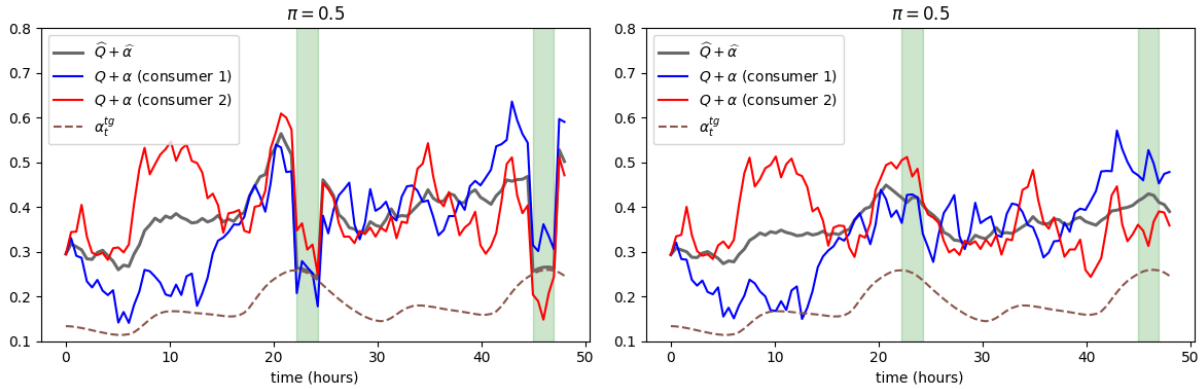


Figure 6: One trajectory of DSM activation jumps issued from the intensity presented in the previous figure.

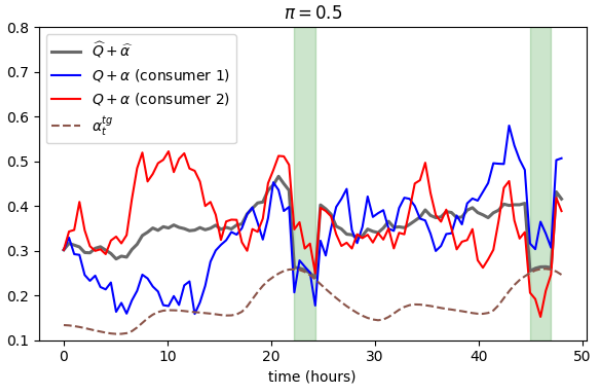


(a) With divergence cost and no dynamic pricing

(b) Dynamic pricing and without divergence cost

Figure 7: Trajectories of $\widehat{Q} + \widehat{\alpha}$ and $Q + \alpha$ (in kW) for two consumers in the MFG setting when these consumers have no dynamic pricing but only the control with respect to the divergence cost (a) and when have dynamic pricing only (b). DSM activations are represented by the green bar.

When consumers are exposed to both dynamic pricing and divergence cost activation, they combine the two behaviours observed above. Their resulting consumption is presented in Figure 8.



(a) With divergence cost and dynamic pricing

Figure 8: Trajectories of $\widehat{Q} + \widehat{\alpha}$ and $Q + \alpha$ (in kW) for two consumers in the MFG setting.

We then analyze how the spot price reacts to the DSM contract (see Figure 9). We can observe that the proportion of consumers with DSM contract (the lower π, the more widespread the DSM contract within the global population) directly impacts how much spot price is smoothed and how much peak prices are reduced.

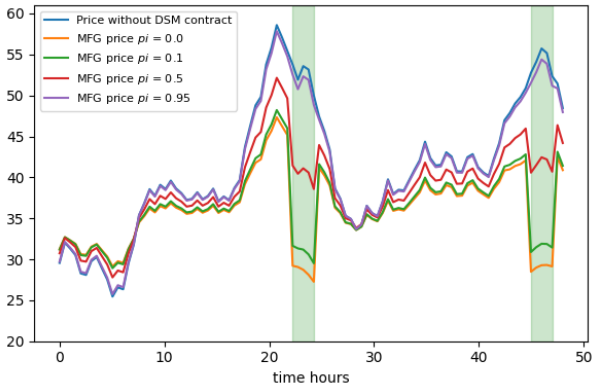


Figure 9: Trajectories of the price p for four different proportions of active consumers in the MFG setting.

A comparison between MFCagg and MFG. We now provide a comparative analysis between the levels of the consumption and corresponding prices in the case when the optimization problem is either implemented from an aggregator perspective, i.e. a MFC problem, or is solved in the MFG setting. We can observe that when consumers are not selfishly optimizing their power consumption, but are guided by an aggregator they make greater effort to reduce their consumption (see figure 10). This efficiency can be attributed to better coordination among consumers in the MFC problem. Naturally, as the prices follow the same trend as the power consumption, we observe that the prices are cheaper when there is an aggregator.

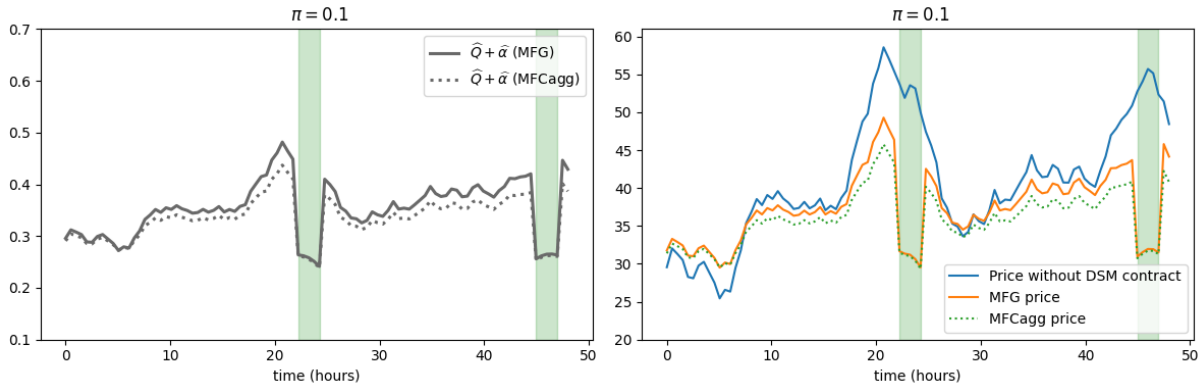


Figure 10: Trajectories of price p (right) and $Q + \alpha$ in kW (left) for MFG setting (plain lines) compared to MFCagg setting (dotting lines)

We also perform numerical computations of the PoA. As expected, the PoA (see Table 6) increases with the proportion of customers who have a DSM contract in the population and is strictly superior to 1 when π is low enough. When we consider $\pi = 0.95$, the impact of the MFG optimization compared to MFC is indeed very little as the proportion of DSM consumers in the population is too low to impact the Price Of Anarchy.

Table 6: PoA with standard prices ($p1 = 87.43$).

	$\pi = 0$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.95$
V^{MFG}	34.104 ($\pm$0.030)	34.354 ($\pm$0.030)	35.387 ($\pm$0.031)	36.601 ($\pm$0.032)
$V^{MFC^{agg}}$	33.519 ($\pm$0.029)	33.876 ($\pm$0.029)	35.226 ($\pm$0.030)	36.599 ($\pm$0.032)
PoA	1.017465	1.014111	1.004558	1.000072

It can also be observed that the PoA is sensitive to the different parameters of the model. In particular, by varying the coefficient p_1, we remark that if the spot price becomes much higher, the PoA increases as well as illustrated in Table 7.

Table 7: PoA with high prices ($p1 = 1000$).

	$\pi = 0$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.95$
V^{MFG}	139.973 ($\pm$0.127)	144.598 ($\pm$0.132)	164.214 ($\pm$0.157)	178.435 ($\pm$0.210)
$V^{MFC^{agg}}$	119.421 ($\pm$0.097)	126.039 ($\pm$0.105)	154.433 ($\pm$0.145)	179.536 ($\pm$0.214)
PoA	1.172094	1.142468	1.063333	0.993867

References

[Ala+23] Clemence Alasseur, Luciano Campi, Roxana Dumitrescu, and Jia Zeng. "MFG model with a long-lived penalty at random jump times: application to demand side management for electricity contracts". In: *Annals of Operations Research* (2023), pp. 1–29.

[ATM19] Clemence Alasseur, Imen Ben Tahar, and Anis Matoussi. *An Extended Mean Field Game for Storage in Smart Grids*. 2019. arXiv: 1710.08991 [math.PR].

[BBP97] Guy Barles, Rainer Buckdahn, and Etienne Pardoux. "Backward stochastic differential equations and integral-partial differential equations". In: *Stochastics and Stochastic Reports* 60 (1997), pp. 57–83.

[BCN22] Erhan Bayraktar, Asaf Cohen, and April Nellis. "A neural network approach to high-dimensional optimal switching problems with jumps in energy markets". In: *arXiv preprint arXiv:2210.03045* (2022).

[Bec+19] Christian Beck, Sebastien Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. "Deep splitting method for parabolic PDEs". In: *arXiv preprint arXiv:1907.03452* (2019).

[BEJ19] Christian Beck, Weinan E, and Arnulf Jentzen. "Machine Learning Approximation Algorithms for High-Dimensional Fully Nonlinear Partial Differential Equations and Second-order Backward Stochastic Differential Equations". In: *J. Nonlinear Sci.* 29.4 (2019), pp. 1563–1619. ISSN: 1432-1467. DOI: 10.1007/s00332-018-9525-3.

[BD07] Christian Bender and Robert Denk. "A forward scheme for backward SDEs". In: *Stochastic processes and their applications* 117.12 (2007), pp. 1793–1812.

[BE08] Bruno Bouchard and Romuald Elie. "Discrete-time approximation of decoupled Forward–Backward SDE with jumps". In: *Stochastic Processes and their Applications* 118.1 (2008), pp. 53–75. ISSN: 0304-4149. DOI: https://doi.org/10.1016/j.spa.2007.03.010. URL: https://www.sciencedirect.com/science/article/pii/S030441490700052X.

[BN04] Bruno Bouchard and Touzi Nizar. "Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations". In: *Stochastic Process. Appl.* 111.2 (2004), pp. 175–206.

[BW12] Bruno Bouchard and Xavier Warin. "Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods". In: *Numerical methods in finance*. Springer, 2012, pp. 215–255.

[BY78] Pierre Bremaud and Marc Yor. "Changes of filtrations and of probability measures". In: *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 45 (1978), pp. 269–295.

[CMW19] Quentin Chan-Wai-Nam, Joseph Mikael, and Xavier Warin. "Machine learning for semi linear PDEs". In: *Journal of Scientific Computing* 79.3 (2019), pp. 1667–1712.

[DO16] Jerome Darbon and Stanley Osher. "Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere". In: *Research in the Mathematical Sciences* 3 (2016), p. 19.

[DL16a] Roxana Dumitrescu and Celine Labart. "Numerical approximation of doubly reflected BSDEs with jumps and RCLL obstacles". In: *Journal of Mathematical Analysis and Applications* 442.1 (2016), pp. 206–243.

[DL16b] Roxana Dumitrescu and Celine Labart. "Reflected scheme for doubly reflected BSDEs with jumps and RCLL obstacles". In: *Journal of Computational and Applied Mathematics* 296 (2016), pp. 827–839.

[DQS17] Roxana Dumitrescu, Marie-Claire Quenez, and Agnes Sulem. "BSDEs with default jump". In: *Computation and Combinatorics in Dynamics, Stochastics and Control* 13 (2017).

[DRZ21] Roxana Dumitrescu, Christoph Reisinger, and Yufei Zhang. "Approximation schemes for mixed optimal stopping and control problems with nonlinear expectations and jumps". In: *Applied Mathematics & Optimization* 83 (2021), pp. 1387–1429.

[FK22] Rüdiger Frey and Verena Köck. "Convergence Analysis of the Deep Splitting Scheme: the Case of Partial Integro-Differential Equations and the associated FBSDEs with Jumps". In: *arXiv preprint arXiv:2206.01597* (2022).

[GPW21] Maximilien Germain, Huyen Pham, and Xavier Warin. "Approximation Error Analysis of Some Deep Backward Schemes for Nonlinear PDEs". In: *SIAM Journal on Scientific Computing* 44.1 (2021), A28–A56.

[GPP22] Alessandro Gnoatto, Marco Patacca, and Athena Picarelli. "A deep solver for BSDEs with jumps". In: (Nov. 2022). DOI: 10.48550/arXiv.2211.04349.

[GLW05] Emmanuel Gobet, Jean-Philippe Lemor, and Xavier Warin. "A regression-based Monte Carlo method to solve backward stochastic differential equations". In: (2005).

[GT16] Emmanuel Gobet and Plamen Turkedjiev. "Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions". In: *Mathematics of Computation* 85.299 (2016), pp. 1359–1391.

[HJW17] Jiequn Han, Arnulf Jentzen, and E Weinan. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (2017), pp. 8505–8510.

[HL20] Jiequn Han and Jihao Long. "Convergence of the Deep BSDE Method for Coupled FBSDEs". In: *Probability, Uncertainty and Quantitative Risk* 5.1 (2020), pp. 1–33. DOI: https://doi.org/10.1186/s41546-020-00047-w.

[HPW20] Come Hure, Huyen Pham, and Xavier Warin. "Deep backward schemes for high-dimensional nonlinear PDEs". In: *Mathematics of Computation* 89.324 (2020), pp. 1547–1579.

[Ji+20] Shaolin Ji, Shige Peng, Ying Peng, and Xichuan Zhang. "Three Algorithms for Solving High-Dimensional Fully Coupled FBSDEs Through Deep Learning". In: *IEEE Intelligent Systems* 35.3 (2020), pp. 71–84. DOI: 10.1109/MIS.2020.2971597.

[KB14] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[LGW06] Jean-Philippe Lemor, Emmanuel Gobet, and Xavier Warin. "Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations". In: *Bernoulli* 12.5 (2006), pp. 889–916.

[LW14] Juan Li and Qingmeng Wei. "Lp estimates for fully coupled FBSDEs with jumps". In: *Stochastic Processes and their Applications* 124.4 (2014), pp. 1582–1611. ISSN: 0304-4149. DOI: https://doi.org/10.1016/j.spa.2013.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0304414913003025.

[LG23] Xiangdong Liu and Yu Gu. "Study on Pricing of High Dimensional Financial Derivatives Based on Deep Learning". In: (2023).

[LS01] Francis A. Longstaff and Eduardo S. Schwartz. "Valuing American options by simulation: a simple least-squares approach". In: *The Review of Financial Studies* 14.1 (2001), pp. 113–147.

[MS90] Dilip Madan and Eugene Seneta. "The Variance Gamma (V.G.) model for share market returns". In: *The Journal of Business* 63.4 (1990), pp. 511–524.

[MMS19] Anis Matoussi, Arij Manai, and Rym Salhi. "Mean-Field Backward-Forward SDE with Jumps and Storage problem in Smart Grids". preprint. June 2019. URL: https://hal.science/hal-02160898.

[Mer76] Robert Merton. "Option pricing when underlying stock returns are discontinuous". In: *Journal of Financial Economics* 3 (1976), pp. 125–144.

[PP90] Etienne Pardoux and Shige Peng. "Adapted solution of a backward stochastic differential equation". In: *Systems & Control Letters* 14.1 (1990), pp. 55–61.

[PWG21] Huyen Pham, Xavier Warin, and Maximilien Germain. "Neural networks-based backward scheme for fully nonlinear PDEs". In: *SN Partial Differential Equations and Applications* 2.16 (2021).

[SS18] Justin Sirignano and Konstantinos Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of computational physics* 375 (2018), pp. 1339–1364.

[VC05] Ekaterina Voltchkova and Rama Cont. "Integro-Differential Equations for Option Prices in Exponential Lévy Models". In: *Finance and Stochastics* 9 (Feb. 2005), pp. 299–325. DOI: 10.1007/s00780-005-0153-z.

[Zhe99] Wu Zhen. "Forward-backward stochastic differential equations with Brownian motion and poisson process". In: *Acta Mathematicae Applicatae Sinica* 15 (1999), pp. 433–443. DOI: 10.1007/BF02684045. URL: https://doi.org/10.1007/BF02684045.

[Zhe03] Wu Zhen. "Fully coupled FBSDE with Brownian motion and Poisson process in stopping time duration". In: *Journal of the Australian Mathematical Society* 74 (2003), pp. 249–266.