Mean-field neural networks: learning mappings on Wasserstein space *†

Huyên Pham‡ Xavier Warin §

October 27, 2022

Abstract

We study the machine learning task for models with operators mapping between the Wasserstein space of probability measures and a space of functions, like e.g. in mean-field games/control problems. Two classes of neural networks based on bin density and on cylindrical approximation, are proposed to learn these so-called mean-field functions, and are theoretically supported by universal approximation theorems. We perform several numerical experiments for training these two mean-field neural networks, and show their accuracy and efficiency in the generalization error with various test distributions. Finally, we present different algorithms relying on mean-field neural networks for solving time-dependent mean-field problems, and illustrate our results with numerical tests for the example of a semi-linear partial differential equation in the Wasserstein space of probability measures.

1 Introduction

Deep neural networks have been successfully used for approximating solutions to high dimensional partial differential equations (PDEs) and control problems, and various methods either based on physics informed representation ([17], [18]), or probabilistic and backward stochastic differential equations (BSDEs) representation ([8], [7], [13]) have been recently developed in the literature, see e.g. the survey papers [2] and [10].

In the last years, a novel class of control problems has emerged with the theory of mean field game/control dealing with models of large population of interacting agents. Solutions to mean-field problems are represented by functions that depend not only on the state variable of the system, but also on its probability distribution, representing the population state distribution, and can be characterized in terms of PDEs in the Wasserstein space of probability measures (called Master equation) or BSDEs of McKean-Vlasov (MKV) type, and we refer to the two-volume monograph [4], [5] for a comprehensive treatment of this topic. In such problems, the input is a probability measure on $\mathbb{R}^d$, hence valued in the infinite dimensional Wasserstein space, and the output is a function defined on the support of the input probability measure.

*This work is supported by FiME, Laboratoire de Finance des Marchés de l'Energie, and the "Finance and Sustainable Development" EDF - CACIB Chair.

†We thank Maximilien Germain and Mathieu Laurière for helpful discussions.

‡LPSM, Université Paris Cité, & FiME pham at lpsm.paris

§EDF R&D & FiME xavier.warin at edf.fr

In order to approximate numerically mean field problems, it is standard and natural in view of propagation of chaos to consider a N-particle approximation of the McKean-Vlasov system, and this approach has been indeed employed in [6], and [9] for reducing the problem to a finite, but possibly very high-dimensional problem. Actually, in the latter paper, symmetry of the N-particle system is exploited in the numerical resolution by using a specific class of neural networks, called DeepSets [20], which allows to reduce significantly the computational complexity. However, this finite dimensional particle approximation of the infinite dimensional mean field problem provides a solution to the Master equation or to the MKV BSDE only for a given initial distribution of the particles, but cannot yield a solution when varying the initial distribution, and thus does not give an approximation of the mean-field function, i.e. the mapping between the space of probability measures and the output space of functions.

In this paper, we aim to approximate the infinite dimensional mean-field function by proposing two classes of neural network architectures. The first approach starts from the approximation of a probability measure with density by a piecewise constant density function on some given fixed partition of size K of a truncated support of the measure, called bins, see Figure 1 in the case of a Gaussian distribution. This allows us to approximate the infinite dimensional mapping by a function that maps an input space of dimension K corresponding to the bin density weights that can be learned by a standard deep neural network. We show a universal approximation theorem that justifies theoretically the use of such bin density neural network. The second approach maps directly probability measures as input but through a finite-dimensional neural network function in cylindrical form, for which we also state a universal approximation theorem.

Next, we show how to effectively learn mean-field function by means of these two classes of mean-field neural networks. This is achieved by generating a data set consisting of simulated probability measures following two proposed methods, and then by training via stochastic gradient method the parameters of the mean-field neural networks. We perform several numerical tests for illustrating the efficiency and accuracy of these two mean-field neural networks on various examples of mean-field functions, and we validate our results on different test distributions by computing the generalization error.

As an application of these mean-field neural networks, we consider dynamic mean-field problems arising typically from mean-field type control, and design different algorithms of local or global type, based on regression or BSDE representation, for computing the solution. We illustrate the performance of our algorithms with the example of a semi-linear PDE on the Wasserstein space. More applications and examples from mean-field control problems and Master equations are investigated in a forthcoming companion paper where we provide a global comparison of the different neural network algorithms.

The paper is organized as follows. In Section 2, we formulate the learning problem, present two network architectures: bin-density and cylindrical neural networks, and explain the data generation and training procedures. Numerical tests are developed in Section 3, and applications to time dependent mean-field problems are given in Section 4 with various algorithms and numerical results. The proofs of the universal approximation theorem for mean-field neural networks are postponed in Appendix A.

Notations. Denote by $\mathcal{P}_2(\mathbb{R}^d)$ the Wasserstein space of square integrable probability measures equipped with the 2-Wasserstein distance $\mathcal{W}_2$. Given $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, we denote by

$L^2(\mu)$ as the set of measurable functions ϕ on $\mathbb{R}^d$ s.t.

$$|\phi|_\mu^2 \;:=\; \int |\phi(x)|^2 \mu(\mathrm{d}x) \;<\; \infty.$$

(Here $|.|$ denotes the Euclidian norm). Given some $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, and ϕ a measurable function on $\mathbb{R}^d$ with quadratic growth condition, hence in $L^2(\mu)$, we set: $\mathbb{E}_{X\sim\mu}[\phi(X)] := \int \phi(x)\mu(\mathrm{d}x)$. We also denote by $\bar{\mu} := \mathbb{E}_{X\sim\mu}[X]$.

2 Learning mean-field functions

Given a function V on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$, valued on $\mathbb{R}^p$, with quadratic growth condition w.r.t. the first argument in $\mathbb{R}^d$, we aim to approximate the infinite-dimensional mapping

$$\mathcal{V} : \mu \in \mathcal{P}_2(\mathbb{R}^d) \longmapsto\; V(\cdot,\mu) \in L^2(\mu), \tag{2.1}$$

called mean-field function, by a map $\mathcal{N}$ constructed from suitable classes of neural networks. The mean-field network $\mathcal{N}$ takes inputs composed of two parts: μ a probability measure and x in the support of μ, and outputs $\mathcal{N}(\mu)(x)$. The quality of this approximation is measured by the error:

$$\begin{aligned} L(\mathcal{N}) \;&:=\; \int_{\mathcal{P}_2(\mathbb{R}^d)} \mathcal{E}_{\mathcal{N}}(\mu)\nu(\mathrm{d}\mu), \\ \text{with}\quad \mathcal{E}_{\mathcal{N}}(\mu) \;&:=\; \big|\mathcal{V}(\mu) - \mathcal{N}(\mu)\big|_\mu^2 \;=\; \mathbb{E}_{X\sim\mu}\big|V(X,\mu) - \mathcal{N}(\mu)(X)\big|^2, \end{aligned}$$

where ν is a probability measure on $\mathcal{P}_2(\mathbb{R}^d)$, called training measure. The learning of the mean-field functional $\mathcal{V}$ will be then performed by minimizing over the parameters of the neural network operator $\mathcal{N}$ the loss function

$$L_M(\mathcal{N}) \;:=\; \frac{1}{M}\sum_{m=1}^{M} \mathcal{E}_{\mathcal{N}}(\mu^{(m)}), \tag{2.2}$$

where $\mu^{(m)}$, $m = 1,\dots,M$ are training samples of ν. We denote by $\widehat{\mathcal{N}}^M$ the learned functional from this minimization problem, and for test data μ^{test} (different from the training data set $(\mu^{(m)})_m$), we shall compute the test (generalization) error $\mathcal{E}_{\widehat{\mathcal{N}}^M}(\mu^{test})$.

2.1 Neural networks approximations

Bin density-based approximation. Let us denote by $\mathcal{D}_2(\mathbb{R}^d)$ the subset of probability measures μ in $\mathcal{P}_2(\mathbb{R}^d)$ which admit density functions p^μ with respect to the Lebesgue measure λ_d on $\mathbb{R}^d$. Fix $\mathcal{K}$ as a bounded rectangular domain in $\mathbb{R}^d$, and divide $\mathcal{K}$ into a number K of bins, $\mathrm{Bin}(k)$, $k = 1,\dots,K$: $\cup_{k=1}^K \mathrm{Bin}(k) = \mathcal{K}$, of center x_k, and with same area size $h = \lambda_d(\mathcal{K})/K$. Given $\mu \in \mathcal{D}_2(\mathbb{R}^d)$, we consider the bin approximation of its density function (see figure 1), that is the truncated piecewise-constant density function defined by

$$\hat{\mathrm{p}}_{\mathcal{K}}^\mu(x) \;=\; p_k^\mu := \frac{\mathrm{p}^\mu(x_k)}{\sum_{k=1}^K \mathrm{p}^\mu(x_k)h}, \;\text{ if } x \in \mathrm{Bin}(k),\; k = 1,\dots,K, \quad \hat{\mathrm{p}}_{\mathcal{K}}^\mu(x) = 0, \quad x \in \mathbb{R}^d \setminus \mathcal{K},$$

set $\boldsymbol{p}^\mu := (p_k^\mu)_{k\in[\![1,K]\!]}$, which lies in $\mathcal{D}_K := \{\boldsymbol{p} = (p_k)_{k\in[\![1,K]\!]} \in \mathbb{R}_+^K : \sum_{k=1}^K p_k h = 1\}$, and called density bins of the probability measure in $\mathcal{D}_2(\mathbb{R}^d)$ of density function $\hat{\mathrm{p}}_{\mathcal{K}}^\mu$, denoted by $\hat{\mu}^K$ with support on $\mathcal{K}$

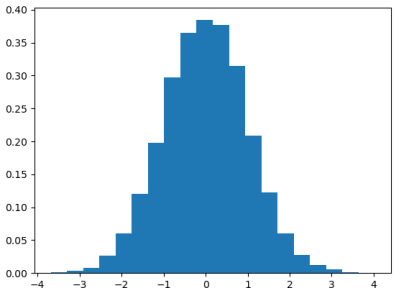


Figure 1: Bin approximation of a Gaussian distribution.

Conversely, given $\boldsymbol{p} = (p_k)_{k\in[\![1,K]\!]} \in \mathcal{D}_K$, we can associate the piecewise-constant density function defined on $\mathbb{R}^d$ by

$$\mathrm{p}(x) = p_k, \quad \text{if } x \in \mathrm{Bin}(k),\ k = 1,\dots,K, \quad \mathrm{p}(x) = 0, \quad x \in \mathbb{R}^d \setminus \mathcal{K}. \tag{2.3}$$

We then denote by $\mu = \mathcal{L}_D(\boldsymbol{p})$ the bin density probability measure on $\mathcal{P}_2(\mathbb{R}^d)$ with piecewise-constant density function p as in (2.3), hence with support on $\mathcal{K}$, and we note that $\hat{\mu}^K = \mathcal{L}_D(\boldsymbol{p}^\mu)$.

A mean-field density-based network is an operator on $\mathcal{D}_2(\mathbb{R}^d)$ in the form

$$\mathcal{N}_D(\mu) \quad = \quad \Phi(\cdot, \boldsymbol{p}^\mu),$$

where $\Phi = \Phi_\theta$ is a neural network function from $\mathbb{R}^d \times \mathcal{D}_K$ into $\mathbb{R}^p$, whose architecture can be constructed as follows:

(i) Classical feedforward neural network

(ii) DeepOnet structure (see [16]): $\Phi_\theta(x, \boldsymbol{p}) = \sum_{\ell=1}^L \mathrm{b}_\ell \mathrm{t}_\ell$, where $(\mathrm{b}_\ell)_\ell$ is the ouput of a branch net with input $\boldsymbol{p} = (p_k)_k$ representing the sensors, and $(\mathrm{t}_\ell)_\ell$ is the output of a trunk net with input x.

(iii) Other structures like the networks developed in [19] for a differentiated treatment of uncertainties and storage level in reservoir optimization.

Let us denote by $\mathcal{D}_c(\mathcal{K})$ the subset of elements μ in $\mathcal{D}_2(\mathbb{R}^d)$ with support in $\mathcal{K}$, with continuous density functions p^μ, and for $\mu \in \mathcal{D}_c(\mathcal{K})$, we set $\omega_{\mathcal{K}}^\mu$ as its modulus of uniform continuity on $\mathcal{K}$. Given a modulus of continuity $\bar{\omega}$, i.e. a nondecreasing function on $\mathbb{R}_+$ s.t. $\lim_{t\downarrow 0} \bar{\omega}(t) = \bar{\omega}(0) = 0$, we denote by $\mathcal{D}_c^{\bar{\omega}}(\mathcal{K})$ the subset of elements $\mu \in \mathcal{D}_c(\mathcal{K})$ such that $\omega_{\mathcal{K}}^\mu \le \bar{\omega}$ on a neighborhood of $t = 0$.

The justification for the use of bin-density neural networks is due to the following universal approximation theorem.

Theorem 2.1. *Let $\mathcal{K}$ be bounded rectangular domain in $\mathbb{R}^d$, $\bar{\omega}$ a modulus of continuity, and V a continuous function on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$. Then, for all $\varepsilon > 0$, there exists $K \in \mathbb{N}^*$, and Φ a neural network on $\mathbb{R}^d \times \mathbb{R}^K$ such that*

$$\left|V(x,\mu) - \Phi(x, \boldsymbol{p}^\mu)\right| \le \varepsilon, \quad \forall x \in \mathcal{K},\ \mu \in \mathcal{D}_c^{\bar{\omega}}(\mathcal{K}).$$

Proof. See Appendix A. □

Cylindrical approximation. A mean-field cylindrical neural network is an operator on $\mathcal{P}_2(\mathbb{R}^d)$ in the form

$$\mathcal{N}_C(\mu) = \Psi_\theta(\cdot, <\varphi_\theta, \mu>),$$

where Ψ_θ is a feedforward network function from $\mathbb{R}^d \times \mathbb{R}^k$ into $\mathbb{R}^p$, and φ_θ is another feedforward network function from $\mathbb{R}^d$ into $\mathbb{R}^k$ (called latent space). Here we denote $<\varphi_\theta, \mu> := \int \varphi_\theta(x)\mu(\mathrm{d}x) = \mathbb{E}_{X\sim\mu}[\varphi_\theta(X)]$. By misuse of language, we call $(\Psi_\theta, \varphi_\theta)$ such cylindrical neural network with φ_θ the inner network, and Ψ_θ the outer network.

We state a universal approximation theorem that justifies the use of cylindrical mean-field neural networks. It is stated with an L^2-distance, which is, in practice, the distance that is minimized during the training process.

Theorem 2.2. *Let ν be a probability measure on $\mathcal{P}_2(\mathbb{R}^d)$, and V be a continuous function from $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$ into $\mathbb{R}^p$ s.t. $\|V\|^2_{L^2(\nu)} := \int_{\mathcal{P}_2(\mathbb{R}^d)} |V(.,\mu)|^2_\mu \nu(\mathrm{d}\mu) < \infty$. Then, for all $\varepsilon > 0$, there exists $k \in \mathbb{N}$, Ψ a neural network from $\mathbb{R}^d \times \mathbb{R}^k$ into $\mathbb{R}^p$, φ a neural network from $\mathbb{R}^d$ into $\mathbb{R}^k$ such that*

$$\int_{\mathcal{P}_2(\mathbb{R}^d)} \left|V(.,\mu) - \Psi(.,<\varphi,\mu>)\right|^2_\mu \nu(\mathrm{d}\mu) \leq \varepsilon.$$

Proof. See Appendix A. □

2.2 Data generation

The training of neural networks for approximating mean-field function relies on samples of probability measures μ and of random state value X distributed according to μ. We propose two methods.

1. We draw a random grid $\boldsymbol{x} = (x_k)_{k\in[\![1,K]\!]}$ of K points in $\mathbb{R}^d$, according to some probability measure on $(\mathbb{R}^d)^K$, as well as a random element $\boldsymbol{\pi} = (\pi_k)_{k\in[\![1,K]\!]}$ in the simplex $\mathcal{S}_K = \{\boldsymbol{\pi} = (\pi_k)_{k\in[\![1,K]\!]} \in \mathbb{R}^K_+ : \sum_{k=1}^K \pi_k = 1\}$. This can be done for example from a sample $e_1, \ldots, e_K$ of positive random variables according to an exponential law, and by setting $\pi_k = \frac{e_k}{\sum_{k=1}^K e_k}$, $k = 1, \ldots, K$. This generates a (random) quantized probability measure:

$$\mathcal{L}^{\boldsymbol{x}}_Q(\boldsymbol{\pi}) \;:=\; \sum_{k=1}^K \pi_k \delta_{x_k},$$

that is the discrete probability measure with support on the grid $\boldsymbol{x}$ and with probability weights $\boldsymbol{\pi}$.

2. We draw random vector $\boldsymbol{p} = (p_k)_{k\in[\![1,K]\!]}$ in $\mathcal{D}_K$. This can be done for example from a sample $e_1, \ldots, e_K$ of positive random variables according to an exponential law, and by setting $p_k = \frac{e_k}{\sum_{k=1}^K e_k h}$, $k = 1, \ldots, K$. This generates (random) bin density probability measure $\mu = \mathcal{L}_D(\boldsymbol{p})$, whose cumulative distribution function is given in the one-dimensional case ($d = 1$, $\mathrm{Bin}(k) = [x_{k-1}, x_k)$, $k = 1, \ldots, K$) by

$$F_{\boldsymbol{p}}(x) = \begin{cases} 0, & x < x_0 \\ \displaystyle\sum_{j=1}^{k-1} p_j h_j \;+\; p_k(x - x_{k-1}), & x \in \mathrm{Bin}(k),\ k = 1, \ldots, K, \\ 1, & x \geq x_K, \end{cases}$$

with the convention that $\sum_{j=1}^{k-1} = 0$ for $k = 1$. Its inverse function is then explicitly given by

$$F_{\boldsymbol{p}}^{-1}(u) = x_{k_u-1} + \frac{u - \sum_{j=1}^{k_u-1} p_j h_j}{p_{k_u}}, \quad u \in [0,1],$$
$$\text{with} \qquad k_u := \inf\{k \in [\![1, K]\!] : \sum_{j=1}^{k} p_j h_j \geq u\}.$$

We then draw an uniform random variable U on $[0,1]$, which generates a random variable $X = F_{\boldsymbol{p}}^{-1}(U)$ distributed according to $\mathcal{L}_D(\boldsymbol{p})$.

2.3 Training mean-field neural networks

According to the choice of the mean-field neural network, the training for learning the mean-field operator $\mathcal{V}$ in (2.1) is performed as follows:

1. *Bin density-based neural network.* We draw a sample $\boldsymbol{p}^{(m)}$ of vectors in $\mathcal{D}_K$, which generates a sample of bin density probability measures $\mu^{(m)} = \mathcal{L}_D(\boldsymbol{p}^{(m)})$, $m = 1, \ldots, M$. By noting that the density bins of the density probability measure $\mu^{(m)}$ is $\boldsymbol{p}^{(m)}$, the training of the bin density-based neural network $\mathcal{N}_D$ from the minimization of the loss function in (2.2) consists in minimizing over the parameters θ of a feedforward neural network Φ_θ on $\mathbb{R}^d \times \mathcal{D}_K$ the loss function

$$L_D(\theta) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{X \sim \mu^{(m)}} \big| V(X, \mu^{(m)}) - \Phi_\theta(X, \boldsymbol{p}^{(m)}) \big|^2.$$

   For the approximation of this expectation $\mathbb{E}_{X \sim \mu^{(m)}}[.]$ when applying SGD we shall use for each m, a batch $X^{(n)}$, $n = 1, \ldots, N$, of samples of $X \sim \mu^{(m)}$. Notice that this method works effectively in dimension $d = 1$ in order to be able to simulate X.

2. *Cylindrical neural network.* We draw a sample $\mu^{(m)}$, $m = 1, \ldots, M$, of probability measures in $\mathcal{P}_2(\mathbb{R}^d)$, either according to discrete probability measures $\mu^{(m)} = \mathcal{L}_Q^{\boldsymbol{x}^{(m)}}(\boldsymbol{\pi}^{(m)})$, or to bin density probability measures $\mu^{(m)} = \mathcal{L}_D(\boldsymbol{p}^{(m)})$, and minimize over the parameters θ of a cylindrical neural network $(\Psi_\theta, \varphi_\theta)$ the loss function:

$$L_C(\theta) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{X \sim \mu^{(m)}} \Big| V(X, \mu^{(m)}) - \Psi_\theta\big(X, \mathbb{E}_{X \sim \mu^{(m)}}[\varphi_\theta(X)]\big) \Big|^2.$$

   Again, when applying SGD for the approximation of this expectation $\mathbb{E}_{X \sim \mu^{(m)}}[.]$, we shall use for each m, a batch $X^{(n)}$, $n = 1, \ldots, N$, of samples of $X \sim \mu^{(m)}$.

3 Numerical tests

We test our two choices of mean-field neural networks by computing the corresponding training error and test (generalization) error for different cases of mean-field functions V on $\mathbb{R} \times \mathcal{P}_2(\mathbb{R})$.

We consider the three following cases of mean-field functions:

A. *Case A: a quadratic function of the measure*

$$V(x,\mu) = \; x + \bar{\mu} + 2\mathrm{Var}(\mu),$$

where $\bar{\mu} := \mathbb{E}_{X\sim\mu}[X]$, $\mathrm{Var}(\mu) := \mathbb{E}_{X\sim\mu}[X^2] - |\bar{\mu}|^2$.

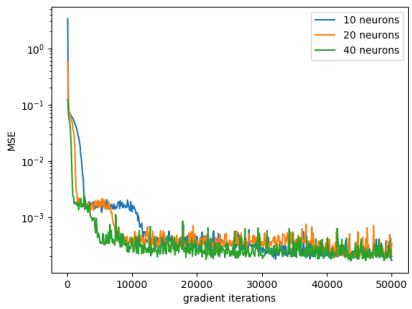
B. *Case B : a first-order mean-field interaction*

$$V(x,\mu) = \int (x-y)^2 \mu(\mathrm{d}y) \; = \; x^2 - 2x\bar{\mu} + \mathbb{E}_{X\sim\mu}[X^2].$$

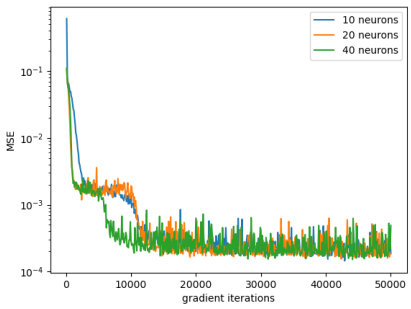
C. *Case C : a second-order mean-field interaction*

$$V(x,\mu) = \int\int (x-y-z)^2 \mu(\mathrm{d}y)\mu(\mathrm{d}z) \; = \; x^2 - 4x\bar{\mu} + 2\mathbb{E}_{X\sim\mu}[X^2] + 2|\bar{\mu}|^2.$$

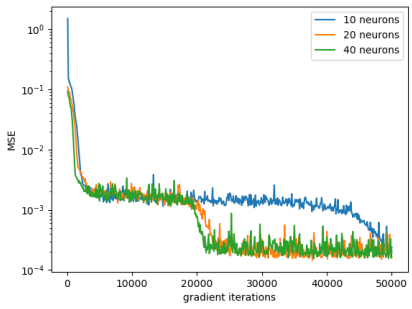
We first want to illustrate the convergence of the bin method with classical feedforward neural network for different hyperparameters. We suppose that the data are generated in all cases with the second method described in section 2.2 using $K = 100$ bins. We take $\hat{M} = 20$ distributions as the batch size during training with $N = 50000$, and give the convergence of the ADAM methods [14] by plotting the mean square error (MSE) every 100 iterations using $M = 1000$ testing distributions. The initial learning rate is 10^{-3} and we use tensorflow [1].



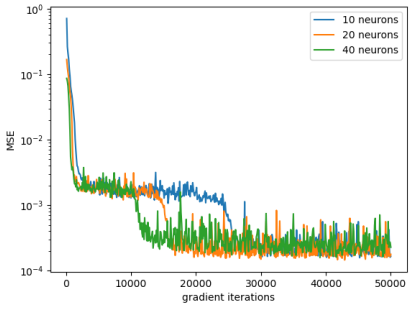
ReLu, 2 hidden layers



ReLu, 3 hidden layers

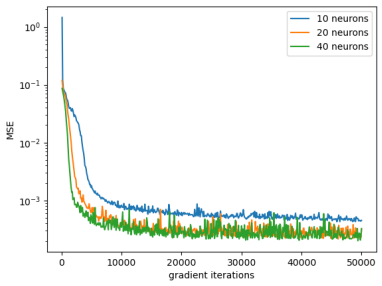


Tanh, 2 hidden layers

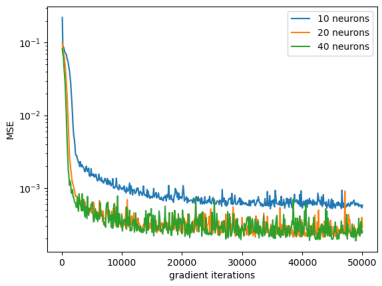


Tanh, 3 hidden layers

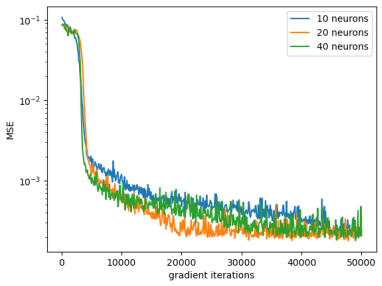
Figure 2: Bin approximation: Training error for case A depending on the number of neurons.



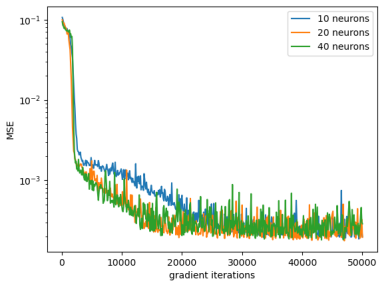
ReLu, 2 hidden layers



ReLu, 3 hidden layers

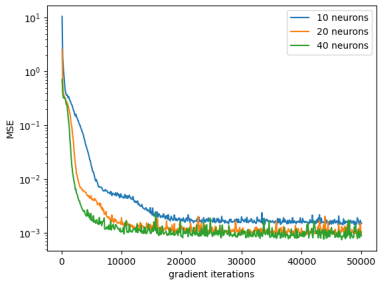


Tanh, 2 hidden layers

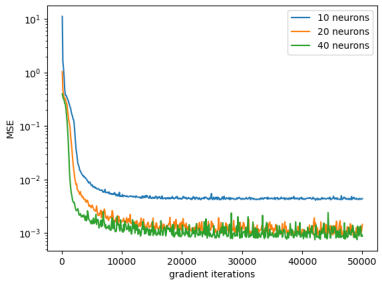


Tanh, 3 hidden layers

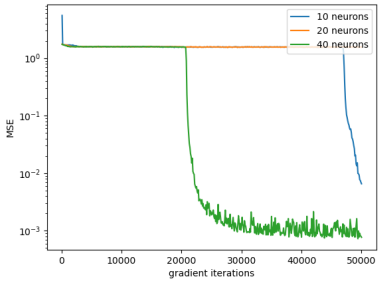
Figure 3: Bin approximation: Training error for case B depending on the number of neurons.



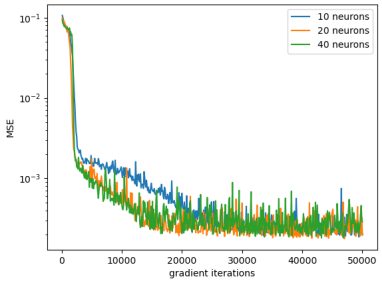
ReLu, 2 hidden layers



ReLu, 3 hidden layers



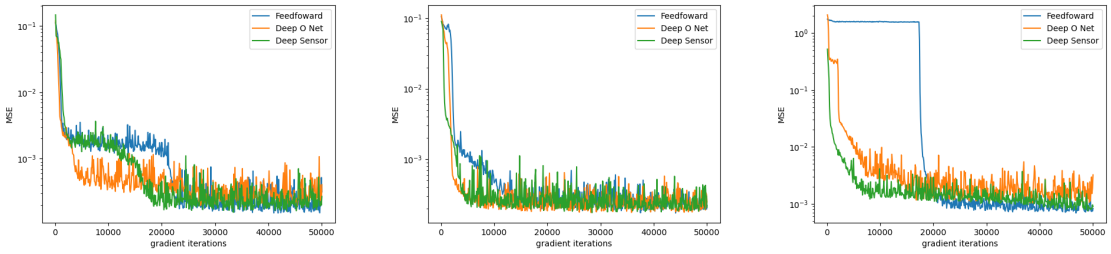
Tanh, 2 hidden layers



Tanh, 3 hidden layers

Figure 4: Bin approximation: Training error for case C depending on the number of neurons.

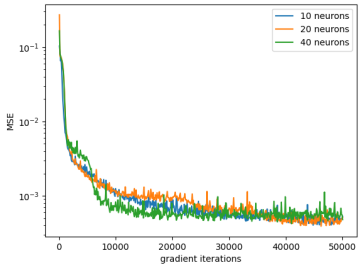
Results for the bin approximation on Figures 2, 3 and 4 indicate that three layers with 20 neurons and a tanh activation function is a good choice. We may also wonder if another architecture for neural network may improve the results: we test the DeepONet network and the network developed in paragraph 3.2 in [19] (Deep Sensor in the graphs). Results with three hidden layers with 20 neurons for each network, the tanh activation function, are given on Figure 5, and show that other networks do not seem to improve the feedforward results. In the sequel we only use the classical feedforward network.



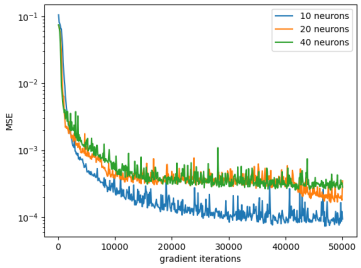
Case A Case B Case C

Figure 5: Comparison of different networks for bin approximation.

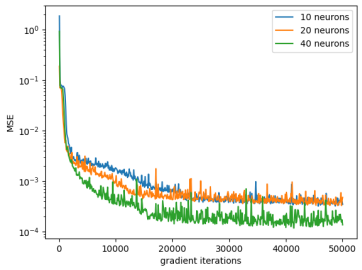
Next, we consider the cylindrical approximation, which uses two networks: One inner network φ having L layers of k neurons (so with output in dimension k), and one outer network Ψ having L hidden layers with Q neurons. The convergence of the training error is illustrated in Figure 6 for the case B: it indicates that a tanh activation function using 2 layers, $Q = 10$, $k = 20$ is a good choice. This result is confirmed on test case A and C but not reported here.



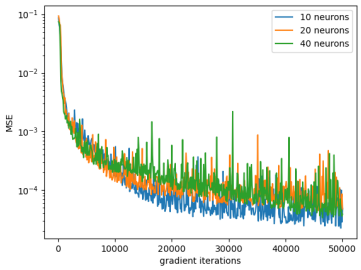
ReLu, $L = 2$, $Q = 10$



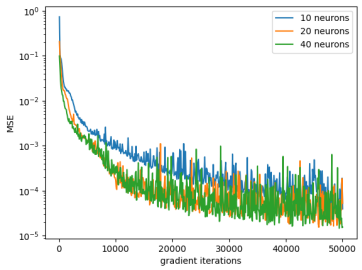
ReLu, $L = 2$, $Q = 20$



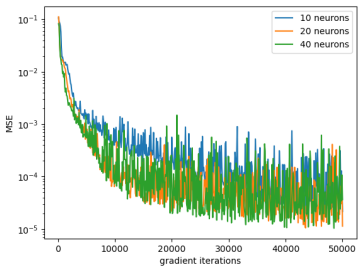
ReLu, $L = 3$, $Q = 10$



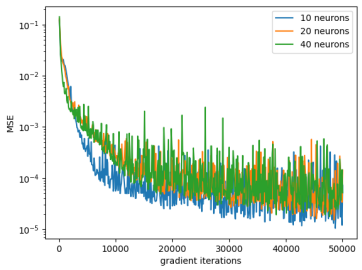
ReLu, $L = 3$, $Q = 20$



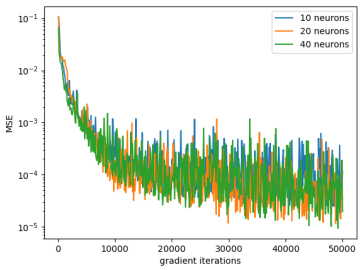
Tanh, $L = 2$, $Q = 10$



Tanh, $L = 2$, $Q = 20$



Tanh, $L = 3$, $Q = 10$



Tanh, $L = 3$, $Q = 20$

Figure 6: Cylinder network: Training error for case B depending on the number of neurons $k = 10$, 20 or 40 of the inner network.

In the sequel, all results are obtained using the previously fitted networks. We have shown that the global training errors decrease correctly, and we shall now compute the error for various given test distributions, and for the two choices of mean-field neural networks:

1. *Bin density-based neural network.* We estimate the density bins $\boldsymbol{p}^{test}$ of μ^{test} from samples $X^{(n)}$, $n = 1, \dots, N$, of μ^{test} as

$$p_k^{test} = \frac{\#\{n \in [\![1, N]\!] : \mathrm{Proj}_{\mathcal{K}}(X^{(n)}) \in \mathrm{Bin}(k)\}}{Nh}, \quad k = 1, \dots, K,$$

where $\mathrm{Proj}_{\mathcal{K}}(.)$ is the projection on $\mathcal{K}$. We then compute the error test

$$\begin{aligned} \mathcal{E}_{\widehat{\mathcal{N}}^M}(\mu^{test}) &= \mathbb{E}_{X\sim\mu^{test}}\big|V(X, \mu^{test}) - \Phi_{\hat{\theta}_M}(X, \boldsymbol{p}^{test})\big|^2 \\ &\simeq \frac{1}{N}\sum_{n=1}^{N}\big|V(X^{(n)}, \mu^{test}) - \Phi_{\hat{\theta}_M}(X^{(n)}, \boldsymbol{p}^{test})\big|^2. \end{aligned}$$

2. *Cylindrical neural network.* From samples $X^{(n)}$, $n = 1, \dots, N$, of μ^{test}, we shall next compute the error test as

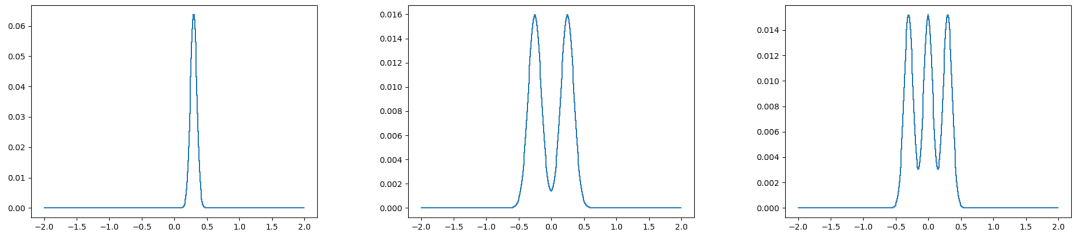
$$\begin{aligned} \mathcal{E}_{\widehat{\mathcal{N}}^M}(\mu^{test}) &= \mathbb{E}_{X\sim\mu^{test}}\big|V(X, \mu^{test}) - \Psi_{\hat{\theta}_M}\big(X^{(n)}, \mathbb{E}_{X\sim\mu^{test}}[\varphi_{\hat{\theta}_M}(X)]\big)\big|^2 \\ &\simeq \frac{1}{N}\sum_{n=1}^{N}\big|V(X^{(n)}, \mu^{test}) - \Psi_{\hat{\theta}_M}\big(X^{(n)}, \frac{1}{N}\sum_{i=1}^{N}\varphi_{\hat{\theta}_M}(X^{(i)})\big)\big|^2. \end{aligned}$$

We test three distributions of $X^{test} \sim \mu^{test}$ plotted in Figure 7 and given by:

(i) Test 1 : Gaussian with $\bar{\mu}^{test} = 0.3$, $std(\mu^{test}) = 0.05$.

(ii) Test 2 : Mixture of two gaussians: $X_0 = P(-a + bY) + (1 - P)(-a + b\bar{Y})$ with P Bernoulli random variable with parameter $\frac{1}{2}$, $a = 0.25$, $b = 0.1$, $Y, \bar{Y} \sim \mathcal{N}(0, 1)$, and independent.

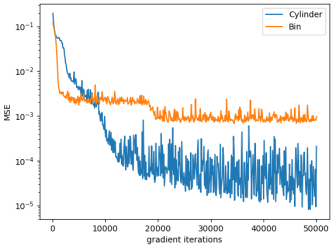
(iii) Test 3 : Mixture of three gaussians: $X_0 = a[-1_{\lfloor 3U \rfloor = 0} + 1_{\lfloor 3U \rfloor = 1}] + bY$ with $U \sim \mathcal{U}(0, 1)$, $a = 0.3$, $b = 0.07$, $Y \sim \mathcal{N}(0, 1)$ independent of U.



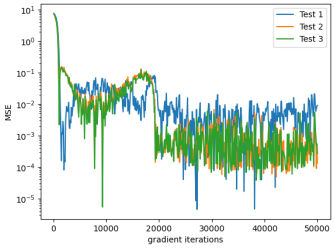
Test 1 Test 2 Test 3

Figure 7: Distributions used to test approximation algorithms.

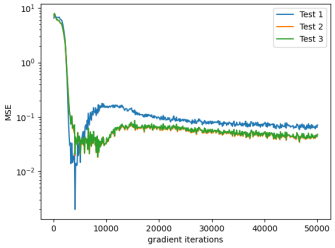
On Figures 8 and 9, we plot for different values of N, keeping $\hat{M} = 20$, the training MSE obtained by the algorithms and the error associated to the test distributions. Globally, the bin approximation is more sensitive to the N parameter and results are less good than with the cylindrical approximation. Not surprisingly, the training size N has to be taken large to get good results for both methods.



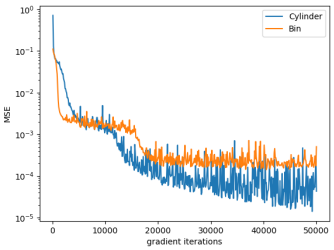
Global MSE : N=10000



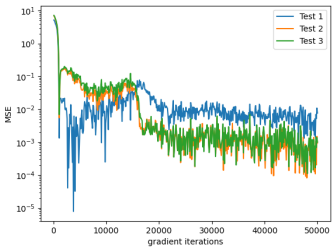
MSE with bins : N=10000



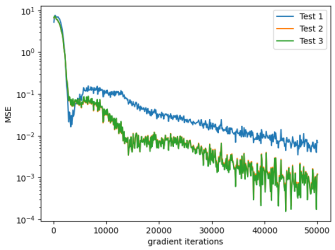
MSE with cylinder : N=10000



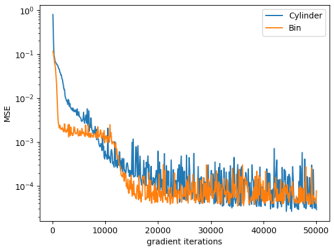
Global MSE : N=50000



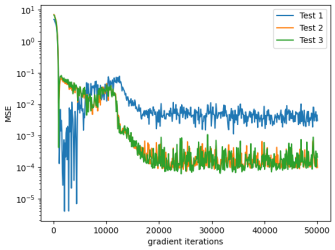
MSE with bins : N=50000



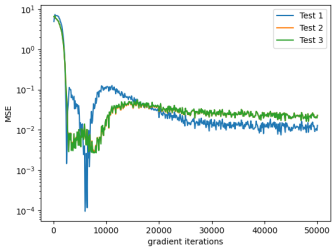
MSE with cylinder : N=50000



Global MSE : N=250000

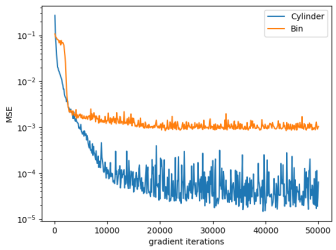


MSE with bins : N=250000

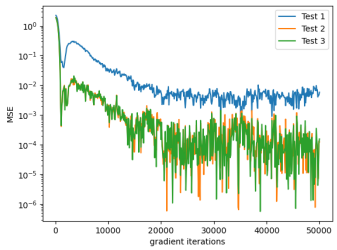


MSE with cylinder : N=250000

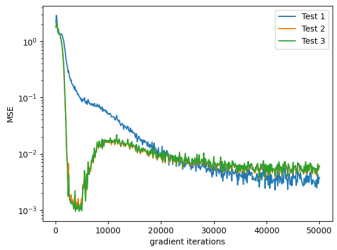
Figure 8: Case A: comparing bin to cylinder methods. Convergence for given distributions.



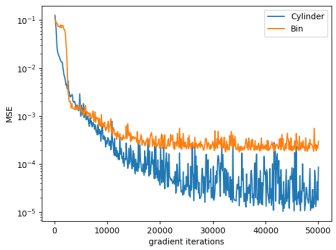
Global MSE : N=10000



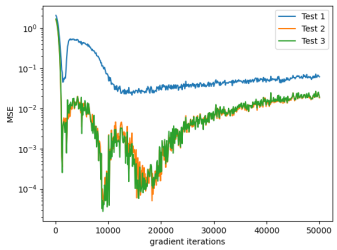
MSE with bins : N=10000



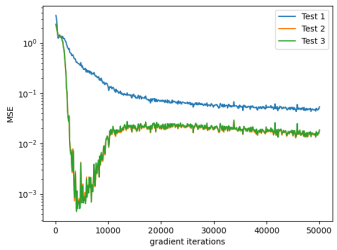
MSE with cylinder : N=10000



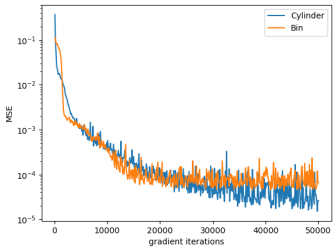
Global MSE : N=50000



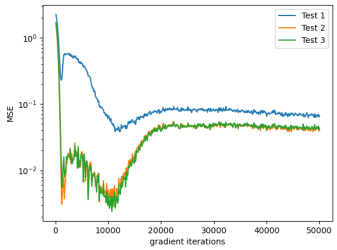
MSE with bins : N=50000



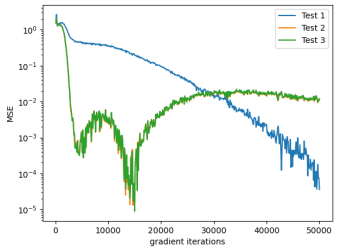
MSE with Cylinder : N=50000



Global MSE : N=250000

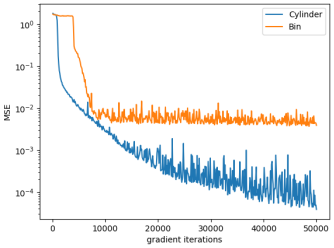


MSE with bins : N=250000

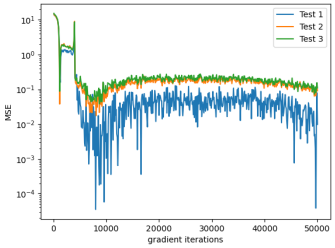


MSE with Cylinder : N=250000

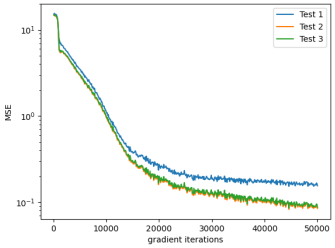
Figure 9: Case B : comparing bin to cylinder methods. Convergence for given distributions.



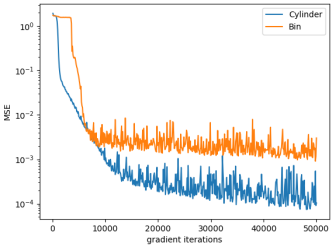
Global MSE : N=10000



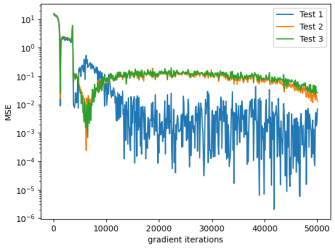
MSE with bins : N=10000



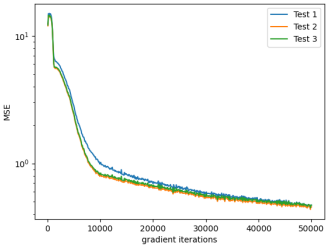
MSE with cylinder : N=10000



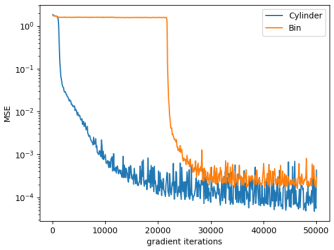
Global MSE : N=50000



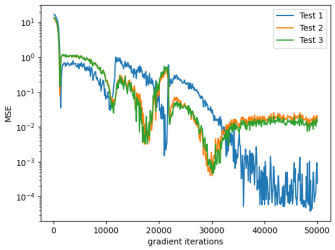
MSE with bins : N=50000



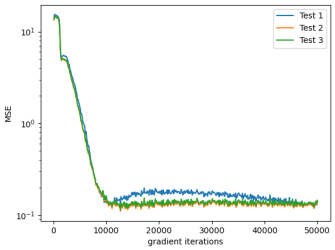
MSE with Cylinder : N=50000



Global MSE : N=250000



MSE with bins : N=250000



MSE with Cylinder : N=250000

Figure 10: Case C : comparing bin to cylinder methods. Convergence for given distributions.

Finally, we consider two other mean-field functions

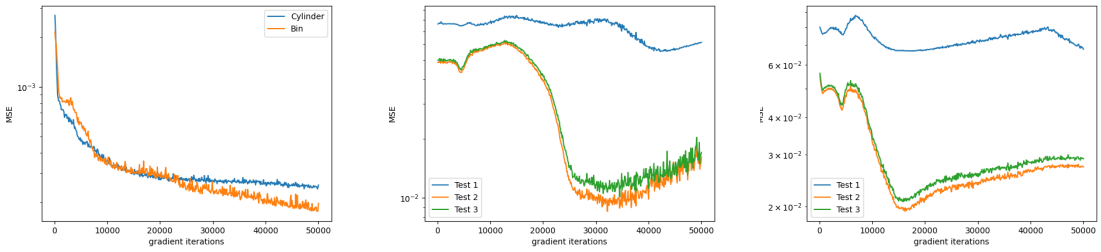
D. *Case D: mediane function*

$$V(x,\mu) = \int |x-y|\mu(dy) \;=\; \mathbb{E}_{X\sim\mu}|x-X|.$$

E. *Case E : cumulative distribution function*

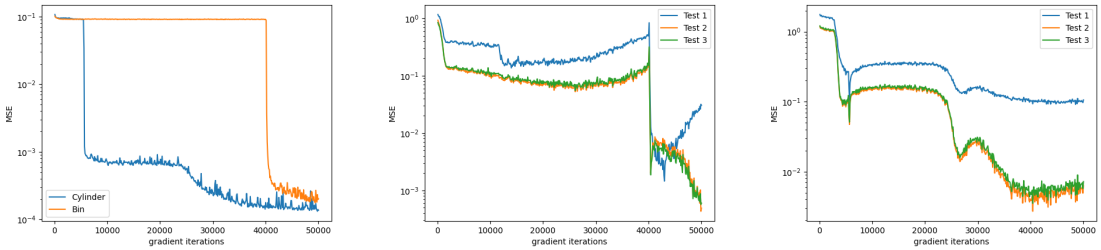
$$V(x,\mu) = \;\mu(-\infty,x] \;=\; \mathbb{E}_{X\sim\mu}[1_{X\leq x}]$$

Notice that, in these two cases, the V function cannot be expressed as a function of the moments of the distribution and Tensorflow cannot vectorize the calculation, which makes the calculation times exploding. We have to limit the number N to 10000.



Global MSE : N=10000 MSE with bins : N=10000 MSE with cylinder : N=10000

Figure 11: Case D: Comparing bin to cylinder methods.



Global MSE : N=10000 MSE with bins : N=10000 MSE with cylinder : N=10000

Figure 12: Case E: Comparing bin to cylinder methods.

Clearly, we see on Figures 11 and 12 that the limited batch size on cases C and D prevents a good accuracy for the approximation of the tested distributions.

4 Algorithms for dynamic mean-field problems

In dynamic mean-field problems (over finite horizon), like mean-field control/game, the solution (value function, control) is time-dependent, and function of some state process and its probability distribution. It is then defined on $\mathcal{T} \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$, where $\mathcal{T}$ is an interval of the form $[0, T]$ in a continuous-time problem, or a discrete time grid $\mathcal{T} = \{t_i, i = 0, \ldots, N_T\}$ in a discrete-time problem. Denoting by U this time-dependent mean-field function, we aim to approximate by learning the time-dependent functional

$$\mathcal{U}(t) : \mu \in \mathcal{P}_2(\mathbb{R}^d) \longmapsto U(t, \cdot, \mu) \in L^2(\mu), \qquad \text{for } t \in \mathcal{T}.$$

The solution U is typically characterized via a Master Bellman PDE, a dynamic programming formula, or a Backward stochastic differential equations of McKean-Vlasov type, and after time discretization (in the case of a continuous-time problem) on a grid $\{t_i, i = 0, \ldots, N_T\}$, one can design algorithms for learning the operators $\mathcal{U}_i := \mathcal{U}(t_i)$, $i = 0, \ldots, N_T$. These machine learning algorithms are either of global or local type, and are described in the next paragraphs.

4.1 Local algorithms

In backward recursion local algorithm, arising from dynamic programming, given an approximation at time t_{i+1} of the mean-field operator $\mathcal{U}_{i+1}$ (e.g. the value function and or the feedback control) by a mean-field neural network function $\widehat{\mathcal{N}}_{i+1}$ as described in the previous section, we aim to learn at time t_i a mean-field neural network function $\mathcal{N}_\theta$ by minimizing over θ a loss function in the form

$$L_i(\theta) \;=\; \mathbb{E}\big[H\big(X_i, \mu_i, \mathcal{N}_\theta(\mu_i)(X_i), \widehat{\mathcal{N}}_{i+1}(\mu_{i+1})(X_{i+1})\big)\big],$$

for some function H, and we then update $\widehat{\mathcal{N}}_i = \mathcal{N}_{\hat\theta_i}$ where $\hat\theta_i$ is the resulting optimal parameter from this minimization problem. In the above expectation for applying SGD, μ_i is sampled according to the data generation as described in the previous section, X_i is sampled according to μ_i, X_{i+1} is given by a dynamics in the form:

$$X_{i+1} \;=\; F_i(X_i, \mu_i, \mathcal{N}_\theta(\mu_i)(X_i), \varepsilon_{i+1}),$$

and μ_{i+1} is the law of X_{i+1}. In practice, μ_{i+1} has to be estimated/approximated from samples of X_{i+1}, and the suitable method will be chosen depending on the adopted class of mean-field neural networks.

1. *Bin density-based neural network*: $\mathcal{N}_\theta(\mu) = \Phi_\theta(., \boldsymbol{p}^\mu)$. We sample probability measures $\mu_i^{(m)} = \mathcal{L}_D(\boldsymbol{p}^{(m)})$ in $\mathcal{D}_2(\mathbb{R})$ from samples $\boldsymbol{p}^{(m)} = (p_k^{(m)})_{k\in[\![1,K]\!]}$, $m = 1, \ldots, M$, in $\mathcal{D}_K$, and then approximate the loss function L_i as

$$\begin{aligned} & L_i(\theta) \\ \simeq\; & \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} H\big(X_i^{(m),(n)}, \mu_i^{(m)}, \Phi_\theta(X_i^{(m),(n)}, \boldsymbol{p}^{(m)}), \Phi_{\hat\theta_{i+1}}(X_{i+1}^{(m),(n)}, \hat{\boldsymbol{p}}^{(m)})\big) \end{aligned}$$

   where $X_i^{(m),(n)}$, $n = 1, \ldots, N$, are sampled from $\mathcal{L}_D(\boldsymbol{p}^{(m)})$,

$$X_{i+1}^{(m),(n)} \;=\; F(X_i^{(m),(n)}, \mathcal{L}_D(\boldsymbol{p}^{(m)}), \Phi_\theta(X_i^{(m),(n)}, \boldsymbol{p}^{(m)}), \varepsilon_{i+1}),$$

   and $\hat{\boldsymbol{p}}^{(m)} = (\hat p_k^{(m)})_{k\in[\![1,K]\!]}$ are the estimated density bins in $\mathcal{D}_K$ of $X_{i+1}^{(m),(n)}$ (truncated on $\mathcal{K} = [x_0, x_K]$), namely:

$$\hat p_k^{(m)} \;=\; \frac{\#\{n \in [\![1,N]\!] : \mathrm{Proj}_{\mathcal{K}}(X_{i+1}^{(m),(n)}) \in \mathrm{Bin}(k)\}}{Nh}, \quad k = 1, \ldots, K,$$

   where $\mathrm{Proj}_{\mathcal{K}}(.)$ is the projection on $\mathcal{K}$.

2. *Cylindrical neural network*: $\mathcal{N}_\theta(\mu) = \Psi_\theta(., <\varphi_\theta, \mu>)$. We sample probability measures $\mu^{(m)} = \mathcal{L}_D(\boldsymbol{p}^{(m)})$, $m = 1, \ldots, M$, so that the loss function is approximated as

$$\begin{aligned} & L_i(\theta) \\ \simeq\; & \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} H\big(X_i^{(m),(n)}, \mu_i^{(m)}, \Psi_\theta(X_i^{(m),(n)}, \hat{\mathbb{E}}[\varphi_\theta(X_i^{(m)})]), \Psi_{\hat\theta_{i+1}}(X_{i+1}^{(m),(n)}, \hat{\mathbb{E}}[\varphi_{\hat\theta_{i+1}}(X_{i+1}^{(m)})])\big), \end{aligned}$$

   where $X_i^{(m),(n)}$, $n = 1, \ldots, N$, are sampled from $X_i^{(m)} \sim \mathcal{L}_D(\boldsymbol{p}^{(m)})$, $X_{i+1}^{(m),(n)}$, $n = 1, \ldots, N$, are sampled as

$$X_{i+1}^{(m),(n)} \;=\; F_i(X_i^{(m),(n)}, \mu_i^{(m)}, \Psi_\theta(X_i^{(m),(n)}, \hat{\mathbb{E}}[\varphi_\theta(X_i^{(m)})]), \varepsilon_{i+1}^{(n)}),$$

   with $\hat{\mathbb{E}}[.]$ denoting the empirical expectation:

$$\hat{\mathbb{E}}[\varphi_\theta(X_i^{(m)})] \;=\; \frac{1}{N}\sum_{n=1}^{N}\varphi_\theta(X_i^{(m),(n)}), \quad \hat{\mathbb{E}}[\varphi_{\hat\theta_{i+1}}(X_{i+1}^{(m)})] \;=\; \frac{1}{N}\sum_{n=1}^{N}\varphi_{\hat\theta_{i+1}}(X_{i+1}^{(m),(n)}).$$

4.2 Global algorithms

In global algorithms, we approximate at any time t_i, $i = 0, \ldots, N_T$, the mean-field operators $\mathcal{U}_i$ by mean-field networks $\mathcal{N}_{\theta_i}$ that are learned simultaneously by minimizing over $\boldsymbol{\theta} = (\theta_i)_i$ a global loss function in the form

$$L(\boldsymbol{\theta}) \quad = \quad \mathbb{E}\Big[\sum_{i=0}^{N_T-1} \ell_i\big(X_i, \mu_i, \mathcal{N}_{\theta_i}(\mu_i)(X_i)\big) + g(X_{N_T}, \mu_{N_T})\Big]$$

for some loss functions ℓ_i, and g. In the above expectation for applying SGD, μ_0 is sampled according to the data generation as described in the previous section, X_0 is sampled according to μ_0, and for $i = 0, \ldots, N_T - 1$, X_{i+1} are given by a dynamics in the form

$$X_{i+1} \quad = \quad F_i(X_i, \mu_i, \mathcal{N}_{\theta_i}(\mu_i)(X_i), \varepsilon_{i+1}),$$

where μ_i is the law of X_i. In practice, for $i = 1, \ldots, N_T$, μ_i has to be estimated/approximated from samples of X_i, and the suitable method will be chosen depending on the adopted class of mean-field neural networks.

1. *Bin density-based neural network*: $\mathcal{N}_\theta(\mu) = \Phi_\theta(., \boldsymbol{p}^\mu)$. We sample probability measures $\mu_0^{(m)} = \mathcal{L}_D(\boldsymbol{p}^{(m)})$ in $\mathcal{D}_2(\mathbb{R})$ from samples $\boldsymbol{p}^{(m)} = (p_k^{(m)})_{k\in[\![1,K]\!]}$, $m = 1, \ldots, M$, in $\mathcal{D}_K$, and then approximate the global loss function as

$$\begin{aligned} L(\boldsymbol{\theta}) \quad \simeq \quad & \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \Big[\ell_0(X_0^{(m),(n)}, \mu_0^{(m)}, \Phi_{\theta_0}(X_0^{(m),(n)}, \boldsymbol{p}^{(m)})\big) \\ & + \sum_{i=1}^{N_T-1} \ell_i\big(X_i^{(m),(n)}, \hat{\mu}_i^{(m)}, \Phi_{\theta_i}(X_i^{(m),(n)}, \hat{\boldsymbol{p}}_i^{(m)}\big) + g(X_{N_T}^{(m),(n)}, \hat{\mu}_{N_T}^{(m)})\Big] \end{aligned}$$

where $X_0^{(m),(n)}$, $n = 1, \ldots, N$ are sampled from $X_0^{(m)} \sim \mu_0^{(m)}$, for $i = 0, \ldots, N_T - 1$, $X_{i+1}^{(m),(n)}$, $n = 1, \ldots, N$ are sampled as

$$X_{i+1}^{(m),(n)} \quad = \quad F_i(X_i^{(m),(n)}, \hat{\mu}_i^{(m)}, \Phi_{\theta_i}(X_i^{(m),(n)}, \hat{\boldsymbol{p}}_i^{(m)}), \varepsilon_{i+1}^{(n)}),$$

with $\hat{\mu}_i^{(m)} = \mathcal{L}_D(\hat{\boldsymbol{p}}_i^{(m)})$, $\hat{\boldsymbol{p}}_0^{(m)} = \boldsymbol{p}^{(m)}$, and $\hat{\boldsymbol{p}}_i^{(m)} = (\hat{p}_{i,j}^{(m)})_{j\in[\![1,K]\!]}$ are the estimated density weights in $\mathcal{D}_K$ of $X_i^{(m),(n)}$, $i = 1, \ldots, N_T$ (truncated on $\mathcal{K} = [x_0, x_K]$), namely:

$$\hat{p}_{i,j}^{(m)} \quad = \quad \frac{\#\{n \in [\![1, N]\!] : \mathrm{Proj}_{\mathcal{K}}(X_i^{(m),(n)}) \in \mathrm{Bin}(j)\}}{Nh_j}, \quad j = 1, \ldots, K,$$

where $\mathrm{Proj}_{\mathcal{K}}(.)$ is the projection on $\mathcal{K}$.

2. *Cylindrical neural network*: $\mathcal{N}_\theta(\mu) = \Psi_\theta(., < \varphi_\theta, \mu >)$. We sample probability measures $\mu_0^{(m)}$ say according to Bin density measures $\mathcal{L}_D(\boldsymbol{p}^{(m)})$, and then minimize over the parameters $\boldsymbol{\theta} = (\theta_i)$ the approximate global loss function

$$\begin{aligned} L(\boldsymbol{\theta}) \quad \simeq \quad & \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \Big[\ell_0(X_0^{(m),(n)}, \mu_0^{(m)}, \Psi_{\theta_0}(X_0^{(m),(n)}, \hat{\mathbb{E}}[\varphi_\theta(X_0^{(m)})])\big) \\ & + \sum_{i=1}^{N_T-1} \ell_i\big(X_i^{(m),(n)}, \hat{\mu}_i^{(m)}, \Psi_{\theta_i}(X_i^{(m),(n)}, \hat{\mathbb{E}}[\varphi_\theta(X_i^{(m)})]\big) + g(X_{N_T}^{(m),(n)}, \hat{\mu}_{N_T}^{(m)})\Big], \end{aligned}$$

where $X_0^{(m),(n)}$, $n = 1, \ldots, N$ are sampled from $X_0^{(m)} \sim \mu_0^{(m)}$, for $i = 0, \ldots, N_T - 1$, $X_{i+1}^{(m),(n)}$, $n = 1, \ldots, N$ are sampled as

$$X_{i+1}^{(m),(n)} \quad = \quad F_i(X_i^{(m),(n)}, \hat{\mu}_i^{(m)}, \Psi_{\theta_i}(X_i^{(m),(n)}, \hat{\mathbb{E}}[\varphi_\theta(X_i^{(m)})]), \varepsilon_{i+1}),$$

$\hat{\mu}_i^{(m)} = \mathcal{L}_D(\hat{\boldsymbol{p}}_i^{(m)})$, $\hat{\boldsymbol{p}}_0^{(m)} = \boldsymbol{p}^{(m)}$, and $\hat{\boldsymbol{p}}_i^{(m)} = (\hat{p}_{i,j}^{(m)})_{j \in [\![1,K]\!]}$ are the estimated density weights in $\mathcal{D}_K$ of $X_i^{(m),(n)}$, $i = 1, \ldots, N_T$ (truncated on $\mathcal{K} = [x_0, x_K]$), namely:

$$\hat{p}_{i,j}^{(m)} \quad = \quad \frac{\#\{n \in [\![1, N]\!] : \mathrm{Proj}_{\mathcal{K}}(X_i^{(m),(n)}) \in \mathrm{Bin}(j)\}}{N h_j}, \quad j = 1, \ldots, K,$$

with $\hat{\mathbb{E}}[.]$ denoting the empirical expectation:

$$\hat{\mathbb{E}}[\varphi_\theta(X_i^{(m)})] \quad = \quad \frac{1}{N} \sum_{n=1}^{N} \varphi_\theta(X_i^{(m),(n)}).$$

Remark 4.1. *For global algorithms, we can avoid the approximation of the mean-field function at each single date t_i, $i = 0, \ldots, N_T - 1$, by learning directly the mean field function which also takes the time as argument. Hence, instead of having a different mean-field neural network $\mathcal{N}_{\theta_i}$ for each date t_i, we learn with a single time dependent neural network $\mathcal{N}(t, .)$, which is used for all dates, as illustrated on an example in the next section. This gives generally more stable results, see e.g. [7].*

4.3 A toy example of semi-linear PDE on Wasserstein space

Let us consider the linear differential operator on $[0, T] \times \mathbb{R} \times \mathcal{P}_2(\mathbb{R})$ associated to the mean-field stochastic differential equation (SDE):

$$\mathrm{d}X_t \quad = \quad \kappa(\mathbb{E}[X_t] - X_t)\mathrm{d}t + \sigma \mathrm{d}W_t,$$

for some positive constants κ, σ, and given by

$$\begin{aligned} \mathcal{L}v(t, x, \mu) \quad = \quad & \frac{\partial v}{\partial t}(t, x, \mu) + \kappa(\bar{\mu} - x)\frac{\partial v}{\partial x}(t, x, \mu) + \frac{1}{2}\sigma^2 \frac{\partial^2 v}{\partial x^2}(t, x, \mu) \\ & + \mathbb{E}_{\xi \sim \mu}\big[\kappa(\bar{\mu} - \xi)\partial_\mu v(t, x, \mu)(\xi) + \frac{1}{2}\sigma^2 \partial_{x'}\partial_\mu v(t, x, \mu)(\xi)\big], \end{aligned}$$

where $x' \mapsto \partial_\mu v(t, x, \mu)(x')$ is the Lions derivative of $\mu \mapsto v(t, x, \mu)$ (see [4]).

Given a C^2 function w on $\mathbb{R}^d$ with quadratic growth condition, let us define the function f on $[0, T] \times \mathbb{R} \times \mathcal{P}_2(\mathbb{R}) \times \mathbb{R}$ by

$$\begin{aligned} f(t, x, \mu, y) = \; & e^{T-t}\mathbb{E}_{\xi \sim \mu}\Big[(w - \sigma^2 D_{xx} w)(x - \xi) + \kappa(x - \xi) D_x w(x - \xi)\Big] \\ & - \; a\Big(\mathbb{E}_{\xi \sim \mu}\big[e^{T-t} w(x - \xi)\big]\Big)^2 + a y^2, \end{aligned} \tag{4.1}$$

for some positive constant a, and consider the semi-linear PDE on $[0, T] \times \mathbb{R} \times \mathcal{P}_2(\mathbb{R})$:

$$\left\{ \begin{array}{rcll} \mathcal{L}v + f(t, x, \mu, v) & = & 0, & (t, x, \mu) \in [0, T) \times \mathbb{R} \times \mathcal{P}_2(\mathbb{R}), \\ v(T, x, \mu) & = & g(x, \mu), & (x, \mu) \in \mathbb{R} \times \mathcal{P}_2(\mathbb{R}), \end{array} \right. \tag{4.2}$$

where $g(x,\mu) := \mathbb{E}_{\xi\sim\mu}[w(x-\xi)]$. By construction, the solution to the PDE (4.2) is explicitly given by $v(t,x,\mu) = e^{T-t}\mathbb{E}_{\xi\sim\mu}[w(x-\xi)]$, and this will serve as benchmark for evaluating the accuracy of our algorithms in the numerical resolution of the PDE (4.2).

Recall that the PDE (4.2) has the following probabilistic representation: by considering the pair of processes (Y,Z) given by

$$Y_t \;=\; v(t,X_t,\mathbb{P}_{X_t}), \qquad Z_t \;=\; \sigma\frac{\partial v}{\partial x}(t,X_t,\mathbb{P}_{X_t}), \quad 0\le t\le T,$$

we see by Itô's formula that it satisfies the Backward SDE

$$\mathrm{d}Y_t = \; -f(t,X_t,\mathbb{P}_{X_t},Y_t)\mathrm{d}t + Z_t\mathrm{d}W_t, \quad Y_T = g(X_T,\mathbb{P}_{X_T}). \tag{4.3}$$

Local Algorithms. We consider a time grid $\mathcal{T} = \{t_i, i=0,\ldots,N_T\}$ of $[0,T]$ with mesh size $\Delta t_i = t_{i+1}-t_i$, and consider two local algorithms for approximating v on $\mathcal{T}\times\mathbb{R}\times\mathcal{P}_2(\mathbb{R})$. In the first approach, we start from the expectation representation arising from (4.3):

$$v(t_i,X_{t_i},\mathbb{P}_{X_{t_i}}) \;=\; \mathbb{E}\Big[v(t_{i+1},X_{t_{i+1}},\mathbb{P}_{X_{t_{i+1}}}) + \int_{t_i}^{t_{i+1}} f(s,X_s,\mathbb{P}_{X_s},v(s,X_s,\mathbb{P}_{X_s}))\mathrm{d}s\Big|X_{t_i}\Big],$$

which leads to the backward regression algorithm: starting from $\hat{\mathcal{U}}_{N_T}(\mu)(x) = g(x,\mu)$, we approximate v at any time t_i, $i = N_T-1,\ldots,0$, by mean-field neural networks $\mathcal{U}_{\theta_i}$, and minimize the local loss regression function

$$L_i^R(\theta_i) \;=\; \mathbb{E}\Big|\hat{\mathcal{U}}_{i+1}(\mu_{i+1})(X_{i+1}) - \mathcal{U}_{\theta_i}(\mu_i)(X_i) + f(t_i,X_i,\mu_i,\mathcal{U}_{\theta_i}(\mu_i)(X_i))\Delta t_i\Big|^2,$$

by updating $\hat{\mathcal{U}}_i = \mathcal{U}_{\hat{\theta}_i}$ with $\hat{\theta}_i$ the resulting "optimal" parameter, and where we sample μ_i, $X_i\sim\mu_i$, with $(X_i)_i$ given by the Euler scheme of the mean-field SDE:

$$X_{i+1} \;=\; X_i + \kappa(\bar{\mu}_i - X_i)\Delta t_i + \sigma\Delta W_{t_i}, \quad \Delta W_{t_i} := W_{t_{i+1}} - W_{t_i}.$$

Alternately, by relying directly on the time discretization of the BSDE (4.3), and following the idea in [13], we approximate v and its gradient $\sigma D_x v$ at any time t_i by mean-field neural networks $\mathcal{U}_{\theta_i}$ and $\mathcal{Z}_{\theta_i}$, and minimize the loss function

$$\begin{aligned} L_i^{BSDE}(\theta_i) \;=\; \mathbb{E}\Big|&\hat{\mathcal{U}}_{i+1}(\mu_{i+1})(X_{i+1}) - \mathcal{U}_{\theta_i}(\mu_i)(X_i) \\ &+ f(t_i,X_i,\mu_i,\mathcal{U}_{\theta_i}(\mu_i)(X_i))\Delta t_i - \mathcal{Z}_{\theta_i}(\mu_i)(X_i)\Delta W_{t_i}\Big|^2. \end{aligned}$$

Global Algorithms. We propose two global methods. In the first one, we approximate v at any time t_i, $i=0,\ldots,N_T$, by a time dependent mean-field neural networks $\mathcal{U}_\theta(t,.)$, and minimize over θ the global loss regression function:

$$\begin{aligned} L^R(\theta) \;=\; \mathbb{E}\Big[&\big|g(X_{N_T},\mu_{N_T}) - \mathcal{U}_\theta(t_{N_T},\mu_{N_T})(X_T)\big|^2 \\ &+ \sum_{i=0}^{N_T-1}\big|\mathcal{U}_\theta(t_{i+1},\mu_{i+1})(X_{i+1}) - \mathcal{U}_\theta(t_i,\mu_i)(X_i) + f(t_i,X_i,\mu_i,\mathcal{U}_\theta(t_i,\mu_i)(X_i))\Delta t_i\big|^2\Big]. \end{aligned}$$

Alternately, following the idea in [8], we approximate v at time $t = 0$ by a mean-field neural networks $\mathcal{U}_{\bar{\theta}}(.)$, and its gradient $\sigma D_x v$ at any time t_i by a time dependent mean-field neural networks $\mathcal{Z}_{\tilde{\theta}}(t, .)$ by minimizing over $\theta = (\bar{\theta}, \tilde{\theta})$ the global loss function:

$$L^{BSDE}(\theta) \quad = \quad \mathbb{E}\Big|Y^{\theta}_{N_T} - g(X_T, \mathbb{P}_{X_T})\Big|^2,$$

where Y^{θ} is given by

$$Y^{\theta}_{i+1} \quad = \quad Y^{\theta}_i - f(t_i, X_i, \mu_i, Y^{\theta}_i)\Delta t_i + \mathcal{Z}_{\tilde{\theta}}(t_i, \mu_i)(X_i)\Delta W_{t_i}, \quad i = 0, \ldots, N_T - 1,$$

starting from $Y^{\theta}_0 = \mathcal{U}_{\bar{\theta}}(\mu_0)(X_0)$, from samples of μ_0, and $X_0 \sim \mu_0$.

Remark 4.2. *The algorithm in [6] gives a solution to the master equation only for a given initial distribution while the algorithms presented here permit to solve the problem for all initial distributions. Furthermore, with local algorithms, we are able to obtain the solution depending on x and μ at each time step.*

Tests. For the numerical tests, we choose $T = 0.1$, $\kappa = 0.2$, $\sigma = 0.5$, $w(x) = \cos(x)$, and $a = 0.1$ in (4.1), and the hyperparameters of the networks are as previously defined. We take $\hat{M} = 10$, $N = 100000$, a number of bins of $K = 200$ with a domain $\mathcal{K} = [-1.3, 1.3]$, $8E4$ gradient iterations at each optimization with an initial learning rate of $1E{-}3$ for the ADAM method. After optimization, we calculate the value function using the network at time $t = 0$ (bin or cylindrical), and then estimate the associated MSE for various test distributions μ^{test} as in Section 3, and following the local/global regression/BSDE algorithms. The results are reported in Tables 1 and 2, and show that the local BSDE algorithm with cylindrical neural networks provides the best results.

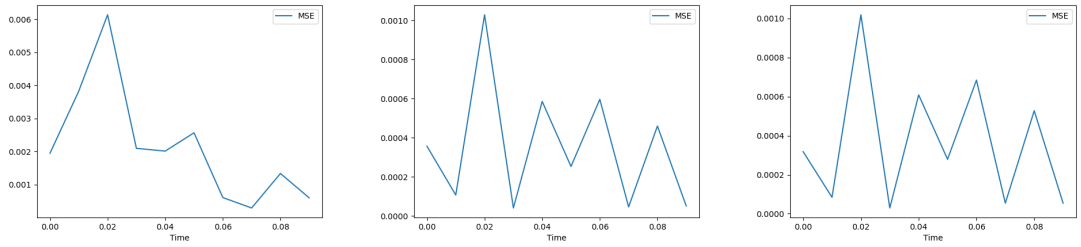
Method	Network	Test 1	Test 2	Test 3
Global	Bins	1.15E-01	5.06E-02	5.38E-02
Global	Cylinder	3.45E-03	2.88E-03	3.00E-03
Local	Bins	2.31E-02	5.89E-03	6.55E-03
Local	Cylinder	7.20E-04	5.20E-04	4.84E-04

Table 1: Regression approach : MSE at time 0 for PDE resolution.

Method	Network	Test 1	Test 2	Test 3
Global	Bins	9.90E-03	7.87E-04	7.89E-04
Global	Cylinder	6.26E-03	2.23E-03	2.40E-03
Local	Bins	5.38E-02	2.78E-02	3.02E-02
Local	Cylinder	1.95E-03	3.57E-04	3.18E-04

Table 2: BSDE approach : MSE at time 0 for PDE resolution.

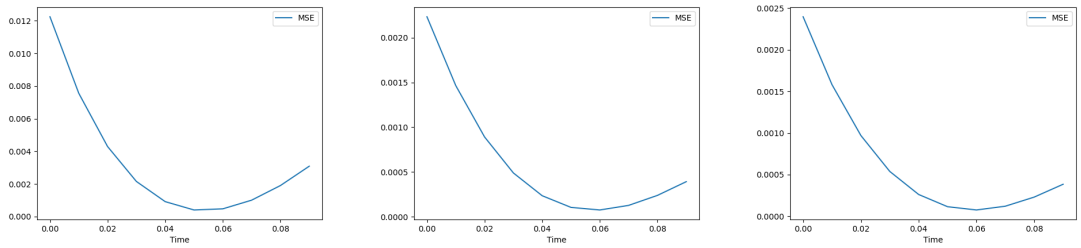
We plot in Figure 13 the MSE error at different time steps when using the local BSDE algorithm with cylindrical mean-field neural networks.



Test 1 Test 2 Test 3

Figure 13: Local BSDE approach with cylindrical NN: MSE at different time steps.

Finally, we plot in Figure 14 the MSE error at different time steps when using the global BSDE algorithm with cylindrical mean-field neural networks.



Test 1 Test 2 Test 3

Figure 14: Global BSDE approach with cylindrical NN: MSE at different time steps.

Remark 4.3. *The use of a single network in global algorithms permits to have a smooth time representation of the value function as shown on Figure 14.*

A Proofs of universal approximation theorems for mean-field neural networks

A.1 Proof of Theorem 2.1

Let $\varepsilon > 0$ be given arbitrarily. Fix $\mathcal{K}$ a bounded rectangular domain in $\mathbb{R}^d$, and divide it into $K = \bar{K}^d$ bins: $\mathrm{Bin}(k)$, $k = 1, \ldots, K$, of center x_k, and same area size $h = \lambda_d(\mathcal{K})/K$, where λ_d is the Lebesgue measure on $\mathbb{R}^d$. Denote by diam_k the diameter of $\mathrm{Bin}(k)$, and notice that $\mathrm{diam}_k \leq \mathrm{diam}(\mathcal{K})/\bar{K}$, where $\mathrm{diam}(\mathcal{K})$ is the diameter of $\mathcal{K}$. Let V be a continuous function on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$.

Step 1. For $\mu \in \mathcal{D}_c(\mathcal{K})$ with density p^μ, denote by $\hat{\mu}^K = \mathcal{L}_D(\boldsymbol{p}^\mu)$ the probability measure with bin density $\boldsymbol{p}^\mu$ in $\mathcal{D}_K$. Since μ, $\hat{\mu}^K$ are supported on the compact set $\mathcal{K}$, they lie in $\mathcal{P}_1(\mathbb{R}^d)$ the set of probability measures with finite first moment. From the Kantorovich-Rubinstein dual representation of the 1-Wasserstein distance, we have

$$\mathcal{W}_1(\mu, \hat{\mu}^K) \quad \leq \quad \sup_{\phi} \int_{\mathcal{K}} \phi(x)(\mathrm{p}^\mu(x) - \hat{\mathrm{p}}^\mu_{\mathcal{K}}(x))\mathrm{d}x,$$

where $\hat{\mathrm{p}}^\mu_{\mathcal{K}}(x) = \sum_{k=1}^K \frac{\mathrm{p}^\mu(x_k)}{N_K} 1_{x\in \mathrm{Bin}(k)}$, with $N_K = \sum_{k=1}^K \mathrm{p}^\mu(x_k)h$, and the supremum is taken over all Lipschitz continuous functions ϕ on $\mathcal{K}$ with Lipschitz constant bounded by 1, and where we can assume w.l.o.g. that $\phi(x_0) = 0$ for some fixed point x_0 in $\mathcal{K}$. We then have

$$\begin{aligned}
\mathcal{W}_1(\mu, \hat{\mu}^K) &\leq \sup_\phi \sum_{k=1}^K \int_{\mathrm{Bin}(k)} \phi(x)\big(\mathrm{p}^\mu(x) - \frac{\mathrm{p}^\mu(x_k)}{N_K}\big)\mathrm{d}x \\
&\leq \sup_\phi \sum_{k=1}^K \Big[\int_{\mathrm{Bin}(k)} \phi(x)\big(\mathrm{p}^\mu(x) - \mathrm{p}^\mu(x_k)\big)\mathrm{d}x + \int_{\mathrm{Bin}(k)} \phi(x)\mathrm{p}^\mu(x_k)\frac{N_K - 1}{N_K}\mathrm{d}x\Big] \\
&\leq \mathrm{diam}(\mathcal{K})h \sum_{k=1}^K \omega^\mu_{\mathcal{K}}(\mathrm{diam}_k) + \mathrm{diam}(\mathcal{K})|N_K - 1| \\
&\leq 2\mathrm{diam}(\mathcal{K})\lambda_d(\mathcal{K})\omega^\mu_{\mathcal{K}}\big(\frac{\mathrm{diam}(\mathcal{K})}{K^{\frac{1}{d}}}\big),
\end{aligned}$$

where we used in the third inequality the fact that $|\phi(x)| \leq |x - x_0| \leq \mathrm{diam}(\mathcal{K})$, for any $x \in \mathcal{K}$, and $|\mathrm{p}^\mu(x) - \mathrm{p}^\mu(x_k)| \leq \omega^\mu_{\mathcal{K}}(\mathrm{diam}_k)$ for any $x \in \mathrm{Bin}(k)$, and in the fourth inequality the fact that $\mathrm{diam}_k \leq \mathrm{diam}(\mathcal{K})/K^{\frac{1}{d}}$, $Kh = \lambda_d(\mathcal{K})$, and the relation

$$\big|1 - N_k\big| \;=\; \Big|\sum_{k=1}^K \int_{\mathrm{Bin}(k)} \big[\mathrm{p}^\mu(x) - \mathrm{p}^\mu(x_k)\big]\mathrm{d}x\Big| \;\leq\; \lambda_d(\mathcal{K})\omega^\mu_{\mathcal{K}}\big(\frac{\mathrm{diam}(\mathcal{K})}{K^{\frac{1}{d}}}\big).$$

By noting that $\mathcal{W}_2(\mu, \hat{\mu}^K) \leq \sqrt{\mathrm{diam}(\mathcal{K})\mathcal{W}_1(\mu, \hat{\mu}^K)}$, this shows that

$$\sup_{\mu\in\mathcal{D}_c^{\bar{\omega}}(\mathcal{K})} \mathcal{W}_2(\mu, \hat{\mu}^K) \to\; 0, \quad \text{as } K \to \infty. \tag{A.1}$$

On the other hand, by Lemma 5.7 and Proposition 5.3 in [3], the set $\mathcal{D}_c(\mathcal{K})$ is relatively compact in $\mathcal{P}_2(\mathbb{R}^d)$, and thus V is uniformly continuous on $\mathcal{K} \times \mathcal{D}_c(\mathcal{K})$. From (A.1), it follows that there exists $K \in \mathbb{N}^*$, such that

$$|V(x, \mu) - V(x, \hat{\mu}^K)| \leq\; \frac{\varepsilon}{2}, \quad \forall x \in \mathcal{K},\; \mu \in \mathcal{D}_c^{\bar{\omega}}(\mathcal{K}). \tag{A.2}$$

Step 2. Denote by V_K the function defined on $\mathbb{R}^d \times \mathcal{D}_K$ by

$$V_K(x, \boldsymbol{p}) \;=\; V(x, \mathcal{L}_D(\boldsymbol{p})), \quad (x, \boldsymbol{p}) \in \mathbb{R}^d \times \mathcal{D}_K,$$

where we recall that $\mathcal{D}_K = \{\boldsymbol{p} = (p_k)_{k\in[\![1,K]\!]} \in \mathbb{R}_+^K : \sum_{k=1}^K p_k h = 1\}$, and $\mathcal{L}_D(\boldsymbol{p})$ is the probability measure with bin density $\boldsymbol{p} \in \mathcal{D}_K$. It is clear that when $(\boldsymbol{p}^n)_n$ converges to $\boldsymbol{p}$ in $\mathcal{D}_K$, then $\mathcal{L}_D(\boldsymbol{p}^n)$ converges weakly to $\mathcal{L}_D(\boldsymbol{p})$, and thus V_K is continuous on $\mathbb{R}^d \times \mathcal{D}_K$. By the classical universal approximation theorem for finite-dimensional functions (see [12]), there exists a feedforward neural network Φ on $\mathbb{R}^d \times \mathcal{D}_K$ such that

$$|V_K(x, \boldsymbol{p}) - \Phi(x, \boldsymbol{p})| \leq\; \frac{\varepsilon}{2}, \quad \forall x \in \mathcal{K},\; \boldsymbol{p} \in \mathcal{D}_K. \tag{A.3}$$

We conclude that for all $x \in \mathcal{K}$, $\mu \in \mathcal{D}_c^{\bar{\omega}}(\mathcal{K})$,

$$\big|V(x, \mu) - \Phi(x, \boldsymbol{p}^\mu)\big| \leq\; \big|V(x, \mu) - V(x, \hat{\mu}^K)\big| + \Big|V_K(x, \boldsymbol{p}^\mu) - \Phi(x, \boldsymbol{p}^\mu)\Big| \;\leq\; \varepsilon.$$

by noting that $V(., \hat{\mu}^K) = V_K(., \boldsymbol{p}^\mu)$, and using (A.2)-(A.3).

A.2 Proof of Theorem 2.2

Step 1: Approximation theorem on compact set. Let $\varepsilon > 0$ be given arbitrarily. Fix $\mathcal{K}$ a compact set of $\mathbb{R}^d$, and let V be a continuous function on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$. By the density of cylindrical polynomial function with respect to mean-field functions, see Lemma 3.12 in [11], for all $\varepsilon > 0$, there exists $k \in \mathbb{N}^*$, P a polynomial function from $\mathbb{R}^d \times \mathbb{R}^k$ into $\mathbb{R}$, Q a polynomial function from $\mathbb{R}^d$ into $\mathbb{R}^k$ s.t.

$$\big|V(x,\mu) - P(x, < Q, \mu >)\big| \leq \frac{\varepsilon}{3}, \quad \forall x \in \mathcal{K},\ \mu \in \mathcal{P}(\mathcal{K}).$$

Now, by the uniform continuity of P on compact sets of $\mathbb{R}^d \times \mathbb{R}^k$, and the classical universal approximation theorem for finite-dimensional functions applied to Q, there exists a feedforward neural network φ from $\mathbb{R}^d$ into $\mathbb{R}^k$ such that

$$\big|P(x, < Q, \mu >) - P(x, < \varphi, \mu >)\big| \leq \frac{\varepsilon}{3}, \quad \forall x \in \mathcal{K}, \mu \in \mathcal{P}(\mathcal{K}),$$

by noting that one can find some compact set $\mathcal{Y}$ (depending on Q and $\mathcal{K}$) of $\mathbb{R}^k$ such that $< Q, \mu >$ and then $< \varphi, \mu >$ lie in $\mathcal{Y}$ for all $\mu \in \mathcal{K}$. Next, we invoke again the classical universal approximation theorem for finite-dimensional functions to get the existence of a feedforward neural network Ψ on $\mathbb{R}^d \times \mathbb{R}^k$ such that

$$\big|P(x,y) - \Psi(x,y)\big| \leq \frac{\varepsilon}{3}, \quad \forall (x,y) \in \mathcal{K} \times \mathcal{Y}.$$

We conclude that for all $x \in \mathcal{K}$, $\mu \in \mathcal{P}(\mathcal{K})$,

$$\begin{aligned}
&\big|V(x,\mu) - \Psi(x, < \varphi, \mu >)\big| \\
&\leq \big|V(x,\mu) - P(x, < Q, \mu >)\big| + \big|P(x, < Q, \mu >) - P(x, < \varphi, \mu >)\big| \\
&\quad + \big|P(x, < \varphi, \mu >) - \Psi(x, < \varphi, \mu >)\big| \leq \varepsilon.
\end{aligned}$$

Step 2: Approximation theorem in L^2. Let $\varepsilon > 0$ be given arbitrarily, and ν be a probability measure on $\mathcal{P}_2(\mathbb{R}^d)$. Given $M > 0$, we truncate the function V by defining V_M on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$ as

$$V_M(x,\mu) = \begin{cases} V(x,\mu), & \text{if } |V(x,\mu)| \leq M, \\ M \frac{V(x,\mu)}{|V(x,\mu)|}, & \text{if } |V(x,\mu)| > M, \end{cases}$$

so that $|V_M(x,\mu)| \leq M$ for all $x \in \mathbb{R}^d$, $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. It is clear that V_M converges pointwise to V as M goes to infinity, and thus by the dominated convergence theorem $\|V - V_M\|^2_{L^2(\nu)}$ converges to zero. We can thus choose $M > \frac{\sqrt{\varepsilon}}{4}$ so that

$$\|V - V_M\|^2_{L^2(\nu)} = \int_{\mathcal{P}_2(\mathbb{R}^d)} |V(.,\mu) - V_M(.,\mu)|^2_\mu \nu(\mathrm{d}\mu) \leq \frac{\varepsilon}{8}. \tag{A.4}$$

Next, we consider some compact set $\mathcal{K}$ of $\mathbb{R}^d$ such that $\nu(\mathcal{P}_2(\mathbb{R}^d) \setminus \mathcal{P}(\mathcal{K})) \leq \varepsilon/(80M^2)$, and we note that V_M is continuous on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$. We can then apply the universal approximation theorem in Step 1, to get the existence of a cylindrical mean-field neural network $(\tilde{\Psi} : \mathbb{R}^d \times \mathbb{R}^k \mapsto \mathbb{R}^p, \varphi : \mathbb{R}^d \mapsto \mathbb{R}^k)$ s.t.

$$\big|V_M(x,\mu) - \tilde{\Psi}(x, < \varphi, \mu >)\big| \leq \frac{\sqrt{\varepsilon}}{4}, \quad \forall x \in \mathcal{K},\ \mu \in \mathcal{P}(\mathcal{K}). \tag{A.5}$$

This implies in particular that

$$\begin{aligned}\left|\tilde{\Psi}(x,<\varphi,\mu>)\right| &\leq |V_M(x,\mu)| + \left|V_M(x,\mu) - \tilde{\Psi}(x,<\varphi,\mu>)\right| \\ &\leq M + \frac{\sqrt{\varepsilon}}{4} < 2M, \qquad x \in \mathcal{K},\ \forall \mu \in \mathcal{P}(\mathcal{K}). \end{aligned} \tag{A.6}$$

By the clipping lemma C.1 in [15], there exists a neural network $\gamma : \mathbb{R}^p \mapsto \mathbb{R}^p$, such that

$$\left\{ \begin{array}{rcl} |\gamma(y) - y| & \leq & \frac{\sqrt{\varepsilon}}{4}, \quad \text{if } |y| \leq M + \frac{\sqrt{\varepsilon}}{4}, \\ |\gamma(y)| & \leq & 2M, \quad \forall y \in \mathbb{R}^p\,. \end{array} \right. \tag{A.7}$$

(Actually, when $p = 1$, one can simply take $\gamma(y) = \min[\max[y, -2M], 2M])$. Define now the neural network $\Psi : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^p$ by $\Psi = \gamma \circ \tilde{\Psi}$. It is then bounded from above by

$$|\Psi(x,z)| \leq 2M, \quad \forall (x,z) \in \mathbb{R}^d \times \mathbb{R}^k,$$

and satisfies for all $x \in \mathcal{K}$, $\mu \in \mathcal{P}(\mathcal{K})$,

$$\begin{aligned}\left|V_M(x,\mu) - \Psi(x,<\varphi,\mu>)\right| &\leq \left|V_M(x,\mu) - \tilde{\Psi}(x,<\varphi,\mu>)\right| \\ &\quad + \left|\gamma \circ \tilde{\Psi}(x,<\varphi,\mu>) - \tilde{\Psi}(x,<\varphi,\mu>)\right| \\ &\leq \frac{\sqrt{\varepsilon}}{4} + \frac{\sqrt{\varepsilon}}{4} = \frac{\sqrt{\varepsilon}}{2}, \end{aligned}$$

by (A.5), (A.6), and (A.7). It follows that

$$\begin{aligned}\int_{\mathcal{P}_2(\mathbb{R}^d)} \left|V_M(.,\mu) - \Psi(.,<\varphi,\mu>)\right|_\mu^2 \nu(\mathrm{d}\mu) &\leq \int_{\mathcal{P}(\mathcal{K})} \left|V_M(.,\mu) - \Psi(.,<\varphi,\mu>)\right|_\mu^2 \nu(\mathrm{d}\mu) \\ &\quad + 2\int_{\mathcal{P}_2(\mathbb{R}^d)\setminus\mathcal{P}(\mathcal{K})} (|V_M(.,\mu)|_\mu^2 + |\Psi(.,<\varphi,\mu>)|_\mu^2)\nu(\mathrm{d}\mu) \\ &\leq \frac{\varepsilon}{4} + 2(M^2 + 4M^2)\frac{\varepsilon}{80M^2} = \frac{3\varepsilon}{8}. \end{aligned}$$

We conclude with (A.4) that

$$\begin{aligned}\int_{\mathcal{P}_2(\mathbb{R}^d)} \left|V(.,\mu) - \Psi(.,<\varphi,\mu>)\right|_\mu^2 \nu(\mathrm{d}\mu) &\leq 2\int_{\mathcal{P}_2(\mathbb{R}^d)} |V(.,\mu) - V_M(.,\mu)|_\mu^2 \nu(\mathrm{d}\mu) \\ &\quad + 2\int_{\mathcal{P}_2(\mathbb{R}^d)} \left|V_M(.,\mu) - \Psi(.,<\varphi,\mu>)\right|_\mu^2 \nu(\mathrm{d}\mu) \\ &\leq 2\frac{\varepsilon}{8} + 2\frac{3\varepsilon}{8} = \varepsilon. \end{aligned}$$

References

[1] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[2] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck. "An overview on deep learning-based approximation methods for PDEs". In: *arXiv:2012.12348v1* (2020).

[3] P. Cardaliaguet. *Notes on mean field games*. Tech. rep. 2013.

[4] R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games: vol. I, Mean Field FBSDEs, Control, and Games,* Springer, 2018.

[5] R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games: vol. II, Mean Field FBSDEs, Control, and Games,* Springer, 2018.

[6] R. Carmona and M. Laurière. "Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: II- the finite horizon case". In: *arXiv:1908.01613, to appear in The Annals of Applied Probability* (2019).

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

[8] W. E, J. Han, and A. Jentzen. "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations". In: *Communications in Mathematics and Statistics* 5.4 (2017), pp. 349–380.

[9] M. Germain, M. Laurière, H. Pham, and X. Warin. "DeepSets and derivative networks for solving symmetric PDEs". In: *Journal of Scientific Computing* 91 (2022).

[10] M. Germain, H. Pham, and X. Warin. "Neural networks based algorithms for stochastic control and PDEs in finance". In: *arXiv:2101.08068 to appear in Machine Learning And Data Sciences For Financial Markets: A Guide To Contemporary Practices*. Ed. by A. Capponi and C.A. Lehalle. Cambridge University Press, 2022.

[11] X. Guo, H. Pham, and X. Wei. "Itô's formula for flows of semimartingales". In: *arXiv: 2010.05288* (2022).

[12] K. Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". In: *Neural Networks* 4 (1991), pp. 251–257.

[13] C. Huré, H. Pham, and X. Warin. "Deep backward schemes for high-dimensional nonlinear PDEs". In: *Mathematics of Computation* 89.324 (2020), pp. 1547–1579.

[14] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[15] S. Lantahler, S. Mishra, and G. Karniadakis. "Error estimates for DeepOnets: a deep learning framework in infinite dimension". In: *Transactions of Mathematics and its Applications* 6 (2022), pp. 1–141.

[16] L. Lu, P. Jin, and G. Karniadakis. "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators". In: *arXiv preprint arXiv:1910.03193* (2019).

[17] M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear PDEs". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[18] J. Sirignano and K. Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375 (2018), pp. 1339–1364.

[19] X. Warin. "Reservoir optimization and Machine Learning methods". In: *arXiv preprint arXiv:2106.08097* (2021).

[20] M. Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3391–3401.