Actor-critic learning algorithms for mean-field control with moment neural networks [*]

Huyên PHAM[†] Xavier WARIN [‡]

February 6, 2025

Abstract

We develop a new policy gradient and actor-critic algorithm for solving mean-field control problems within a continuous time reinforcement learning setting. Our approach leverages a gradient-based representation of the value function, employing parametrized randomized policies. The learning for both the actor (policy) and critic (value function) is facilitated by a class of moment neural network functions on the Wasserstein space of probability measures, and the key feature is to sample directly trajectories of distributions. A central challenge addressed in this study pertains to the computational treatment of an operator specific to the mean-field framework. To illustrate the effectiveness of our methods, we provide a comprehensive set of numerical results. These encompass diverse examples, including multi-dimensional settings and nonlinear quadratic mean-field control problems with controlled volatility.

Keywords: Mean-field control, reinforcement learning, policy gradient, moment neural network, actor-critic algorithms.

1 Introduction

This paper is concerned with the numerical resolution of mean-field (a.k.a. McKean-Vlasov) control in continuous time in a partially model-free reinforcement learning setting. The dynamics of the controlled mean field stochastic differential equation on $\mathbb{R}^d$ is in the form

$$\mathrm{d}X_t = b(t, X_t, \mathbb{P}_{X_t}, \alpha_t)\mathrm{d}t + \sigma(t, X_t, \mathbb{P}_{X_t}, \alpha_t)\mathrm{d}W_t, \quad 0 \leq t \leq T, \ X_0 \sim \mu_0, \tag{1.1}$$

where W is a standard d-dimensional Brownian motion on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_t, \mathbb{P})$, $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$, the Wasserstein space of square integrable probability measures, $\mathbb{P}_{X_t}$ denotes the marginal distribution of X_t at time t, and the control process α is valued in $A \subset \mathbb{R}^p$. The coefficients b, σ are in the separable form:

$$\textbf{(SC)} \qquad b(t,x,\mu,a) \ = \ \beta(t,x,\mu) + C(t,a), \qquad \sigma\sigma^{\intercal}(t,x,\mu,a) \ = \ \Sigma(t,x,\mu) + \vartheta(t,a),$$

for $(t,x,\mu,a) \in [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d) \times \mathbb{R}^p$, where β, Σ depending on the state variable and its probability distribution are unknown functions, while the coefficients C, ϑ on the control are known functions on $[0,T] \times \mathbb{R}^p$.

The expected total cost associated to a control α is given by

$$\mathbb{E}\Big[\int_0^T f(X_t, \mathbb{P}_{X_t}, \alpha_t)\mathrm{d}t + g(X_T, \mathbb{P}_{X_T}) \Big], \tag{1.2}$$

[*]This work is supported by FiME, Laboratoire de Finance des Marchés de l'Energie, and the "Finance and Sustainable Development" EDF - CACIB Chair. H. Pham is also supported by the BNP-Paribas Chair "Futures of Quantitative Finance".

[†]Corresponding author. LPSM, Université Paris Cité, & FiME pham at lpsm.paris

[‡]EDF R&D & FiME xavier.warin at edf.fr

and we denote by $(t, x, \mu) \mapsto V(t, x, \mu)$ the associated value function defined on $[0, T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$. The analytical forms of f, g are unknown, but it is assumed that given an input $(x, \mu, a) \in \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d) \times A$, we obtain (by observation or from a blackbox) the realized output costs $f(x, \mu, a)$ and $g(x, \mu)$. Such setting is partially model-free, as the model coefficients β, Σ, f, and g are unknown, and we only assume that the action functions C, ϑ on the control are known.

The theory and applications of mean-field control (MFC) problems are concerned with the study of large population models of cooperative interacting agents, and have generated a vast literature in the last decade with various applications in economics, finance, social sciences and herd behaviour. We refer to the monographs [2], [3] for a comprehensive treatment of this topic. From a numerical aspect, the main challenging issue is the infinite dimensional feature of MFC coming from the distribution law state variable. In a model-based setting, i.e., when all the coefficients b, σ, f and g are known, several recent works have proposed deep learning schemes for MFC, based on neural network approximations of the feedback control and/or the value function solution to the Hamilton-Jacobi-Bellman equation from the dynamic programming or backward stochastic differential equations (BSDEs) from the maximum principle, see [5], [8], [9], [11], [17], [16], [14].

The approximation of solutions to MFC in a (partially) model-free setting is the purpose of reinforcement learning (RL) where one learns in an unknown environment the optimal control (and the value function) by repeatedly trying policies, observing state, receiving and evaluating rewards, and improving policies. RL is a very active branch of machine learning, see the seminal reference monograph [19], and has recently attracted attention in the context of mean-field control in discrete-time, and mostly by Q-learning methods, see [6], [1], [10].

In this paper, we consider a partially model-free continuous time setting as described above, and adopt a policy gradient approach as in [7]. This relies on a gradient representation of the cost functional associated to randomized policies, which makes appear an additional operator term $\mathcal{H}$ compared to the classical diffusion setting of [12], and specific to the mean-field setting. The computational treatment of this operator $\mathcal{H}$ on functions defined on the Wasserstein space is the crucial issue, and has been handled in [7] only for one-dimensional linear quadratic (LQ) models, hence with a very particular dependence of the value function and optimal control on the law of the state. Here, we address the general dependence of the coefficients on the distribution state, and deal with the operator $\mathcal{H}$ by means of the class of moment neural networks. This class of neural networks consists of functions that depend on the measure via its first L moments, and satisfies some universal approximation theorem for functions defined on the Wasserstein space, see [15], [22]. We then design an actor-critic algorithm for learning alternately the optimal policy (actor) and value function (critic) with moment neural networks, which provides an effective resolution of MFC control problems in a (partially) model-free setting beyond the LQ setting for multivariate dynamics with control on the drift and the volatility. Our actor-critic algorithm has the structure of general actor-critic algorithms but during gradient iterations, instead of following a single state trajectory by sampling, we follow the evolution of an entire distribution that is initially randomly chosen and described by its empirical measure obtained with a large fixed number of particles. Then, the batch version of the algorithm consists in sampling and following N distributions together to estimate the gradients.

The outline of the paper is organized as follows. We recall in Section 2 the gradient representation of the functional cost with randomized policies and formulate notably the expression of the operator $\mathcal{H}$. In Section 3, we consider the class of moment neural networks, and show how it acts on the operator $\mathcal{H}$. We present in Section 4 the actor-critic algorithm, and Section 5 is devoted to numerical results for illustrating the accuracy and efficiency of our algorithm. We present various examples with control on the drift and on the volatility, and non LQ examples in a multi-dimensional setting.

Notations. The scalar product between two vectors x and y is denoted by $x \cdot y$, and $|\cdot|$ is the Euclidian norm. Let $Q = (Q_{i_1 \ldots i_q}) \in \mathbb{R}^{d_1 \times \ldots \times d_q}$ be a tensor of order q, and $P = (P_{i_1 \ldots i_p}) \in \mathbb{R}^{d_1 \times \ldots \times d_p}$ be a tensor of order $p \leq q$. We denote by $Q \circ P$ the circ product defined as the tensor in $\mathbb{R}^{d_{p+1} \times \ldots \times d_q}$ with components:

$$[Q \circ P]_{i_{p+1} \ldots i_q} = \sum_{i_1, \ldots, i_p} Q_{i_1 \ldots i_p i_{p+1} \ldots i_q} P_{i_1 \ldots i_p}.$$

When $q = p = 1$, $\circ$ is the scalar product in $\mathbb{R}^{d_1}$. When $q = 2$, $p = 1$, $Q \circ P = Q^\intercal P \in \mathbb{R}^{d_2}$ where $^\intercal$ is the transpose matrice operator. When $q = p = 2$, $\circ$ is the inner product $Q \circ P = \mathrm{tr}(Q^\intercal P)$ where tr is the trace operator. When $q = 3$, $Q \circ P$ is a vector in $\mathbb{R}^{d_3}$ for $p = 2$, and a matrix in $\mathbb{R}^{d_2 \times d_3}$ for $p = 1$.

2 Preliminaries

We adopt a policy gradient approach by searching optimal control among a parametric class of probability transition kernels π_θ from $[0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$ into $\mathbb{R}^p$, called randomized policies with densities $p_\theta(t,x,\mu,.)$ w.r.t. some measure on $\mathbb{R}^p$, and then by minimizing over the parameters $\theta \in \mathbb{R}^D$ the functional

$$\mathrm{J}(\theta) = \mathbb{E}_{\alpha\sim\pi_\theta}\Big[\int_0^T f(X_t, \mathbb{P}_{X_t}, \alpha_t)\mathrm{d}t + g(X_T, \mathbb{P}_{X_T})\Big]. \tag{2.1}$$

Here $\alpha \sim \pi_\theta$ means that at each time t, the action α_t is sampled (independently from W) from the probability distribution $\pi_\theta(.|t, X_t, \mathbb{P}_{X_t})$.

Remark 2.1. *This randomization of action is done formally via pairwise independent uniformly distributed random variables $(U_t)_t$ that are used to generate $\pi_\theta(\cdot|X_t, \mathbb{P}_{X_t})$ at any time. However, dealing with a continuum of i.i.d. random variables over the index set $[0,T]$ induces measurability issues, in the sense that for almost all ω, $t \mapsto U_t(\omega)$ is not Lebesgue measurable, see [18]. This can be overcome in theory by extending the probability space with Lebesgue measure via the so-called Fubini extension. Alternatively, one can adopt a randomization on an arbitrary discrete time grid as in Definition 2.3 in [20], and view the continuous-time sampling formulation* (2.1) *as an heuristic tool for deriving the policy gradient representation.*

Remark 2.2. *We may include an entropy (e.g. Shannon) regularization term in the functional cost* (2.1) *as proposed in [21] for encouraging exploration of randomized policies. In this case, equation* (2.1) *is modified as*

$$\mathrm{J}(\theta) = \mathbb{E}_{\alpha\sim\pi_\theta}\Big[\int_0^T f(X_t, \mathbb{P}_{X_t}, \alpha_t) - \lambda\mathcal{E}\big(\pi_\theta(.|t, X_t, \mathbb{P}_{X_t})\big)\mathrm{d}t + g(X_T, \mathbb{P}_{X_T})\Big].$$

with a Shannon entropy term

$$\mathcal{E}\big(\pi_\theta(.|t,x,\mu)\big) \;:=\; -\textstyle\int_{\mathbb{R}^p} \log p_\theta(t,x,\mu,a)\pi_\theta(\mathrm{d}a|t,x,\mu).$$

This can slightly help the convergence of the policy gradient algorithms by permitting the use of higher learning rates, but comparing the accuracy of the results with and without the entropy term on all the examples treated in this article, we don't see an improvement while using the entropy term. Therefore, we only consider in this paper exploration through the randomization of policies.

We have the gradient representation of J as derived in [7]:

$$\begin{aligned}\mathrm{G}(\theta) :=\; & \nabla_\theta \mathrm{J}(\theta) \\ =\; & \mathbb{E}_{\alpha\sim\pi_\theta}\Big[\int_0^T \nabla_\theta \log p_\theta(t, X_t, \mathbb{P}_{X_t}, \alpha_t)\big[\mathrm{d}J_\theta(t, X_t, \mathbb{P}_{X_t}) + f(X_t, \mathbb{P}_{X_t}, \alpha_t)\mathrm{d}t\big] \\ & \qquad + \int_0^T \mathcal{H}_\theta[J_\theta](t, X_t, \mathbb{P}_{X_t})\mathrm{d}t\Big],\end{aligned} \tag{2.2}$$

where $J_\theta : [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$ is the dynamic value function associated to (2.1), hence satisfying the property that

$$(\mathbf{MJ}) \quad \{J_\theta(t, X_t, \mathbb{P}_{X_t}) + \int_0^t f(X_s, \mathbb{P}_{X_s}, \alpha_s)\mathrm{d}s,\ 0 \le t \le T\} \text{ is a martingale,}$$

and $\mathcal{H}_\theta$ is the operator specific to the mean-field framework, defined by

$$\mathcal{H}_\theta[\varphi](t,x,\mu) = \nabla_\theta \mathbb{E}_{\xi\sim\mu}\Big[b_\theta(t,\xi,\mu)\cdot\partial_\mu\varphi(t,x,\mu)(\xi) + \frac{1}{2}\Sigma_\theta(t,\xi,\mu)\circ\partial_\xi\partial_\mu\varphi(t,x,\mu)(\xi)\Big] \in \mathbb{R}^D$$

with $b_\theta(t,x,\mu) = \int_A b(t,x,\mu,a)\pi_\theta(\mathrm{d}a|t,x,\mu)$, $\Sigma_\theta(t,x,\mu) = \int_A \sigma\sigma^\intercal(t,x,\mu,a)\pi_\theta(\mathrm{d}a|t,x,\mu)$. Here $\partial_\mu\varphi(t,x,\mu)(.)$ is the Lions-derivative with respect to $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, and it is a function from $\mathbb{R}^d$ into $\mathbb{R}^d$, and $\mathbb{E}_{\xi\sim\mu}[.]$ means that the expectation is taken with respect to the random variable ξ distributed according to μ.

Notice that under the structure condition **(SC)**, we have

$$b_\theta(t,x,\mu) \ = \ \beta(t,x,\mu) + C_\theta(t,x,\mu), \quad \Sigma_\theta(t,x,\mu) \ = \ \Sigma(t,x,\mu) + \vartheta_\theta(t,x,\mu)$$

where $C_\theta(t,x,\mu) := \int_A C(t,a)\pi_\theta(\mathrm{d}a|t,x,\mu)$, $\vartheta_\theta(t,x,\mu) := \int_A \vartheta(t,a)\pi_\theta(\mathrm{d}a|t,x,\mu)$ are known functions, and thus

$$\mathcal{H}_\theta[\varphi](t,x,\mu) = \nabla_\theta \mathbb{E}_{\xi\sim\mu}\Big[C_\theta(t,\xi,\mu)\cdot\partial_\mu\varphi(t,x,\mu)(\xi) + \frac{1}{2}\vartheta_\theta(t,\xi,\mu)\circ\partial_\xi\partial_\mu\varphi(t,x,\mu)(\xi)\Big].$$

3 Parametrization of actor/critic functions with moment neural networks

A moment neural network function on $[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$ of order $L\in\mathbb{N}^*$ is a parametric function in the form

$$\phi_\eta(t,x,\mu) = \ \Psi_\eta(t,x,\bar{\boldsymbol{\mu}}_L),$$

where $\bar{\boldsymbol{\mu}}_L = (\bar{\mu}^\ell)_{\ell\in\mathcal{L}}$, with $\bar{\mu}^\ell = \mathbb{E}_{\xi\sim\mu}[\prod_{i=1}^d \xi_i^{\ell_i}]$ for $\ell = (\ell_i)_{i\in[\![1,d]\!]} \in \mathcal{L} = \{\ell = (\ell_1,\ldots,\ell_d)\in\mathbb{N}^d : \sum_{i=1}^d \ell_i \leq L\}$ of cardinality L_d, and $(t,x,y)\in[0,T]\times\mathbb{R}^d\times\mathbb{R}^{L_d}\mapsto\Psi_\eta(t,x,y)$ is a classical finite-dimensional feedforward neural network with parameters η. Moment neural networks have been considered in [22] as a special case of cylindrical mean-field neural networks, and satisfy a universal approximation theorem for continuous functions on $[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$, see [15]. By abuse of notation and language, we identify ϕ_η and Ψ_η, and call them indifferently moment neural networks.

We shall parametrize the randomized policy (actor) by a Gaussian probability transition kernel in the form

$$\pi_\theta(.|t,x,\mu) \ = \ \mathcal{N}(m_\theta(t,x,\bar{\boldsymbol{\mu}}_L),\lambda\mathbb{I}_p),$$

where m_θ is a moment neural network, hence with log density:

$$\log p_\theta(t,x,\mu,a) \ = \ -\frac{1}{2}\log(2\pi\lambda) - \frac{|a-m_\theta(t,x,\bar{\boldsymbol{\mu}}_L)|^2}{2\lambda},$$

and $\lambda > 0$ is a parameter for exploration. Notice that in this case, the known functions C_θ, ϑ_θ depend on μ only though its L moments $\bar{\boldsymbol{\mu}}_L$, and by misuse of notation we also write: $C_\theta(t,x,\bar{\boldsymbol{\mu}}_L)$, $\vartheta_\theta(t,x,\bar{\boldsymbol{\mu}}_L)$.

The value function (critic) is parametrized by a moment neural network $\mathcal{J}_\eta(t,x,\mu) = \mathcal{J}_\eta(t,x,\bar{\boldsymbol{\mu}}_L)$, and by noting that the L-derivative of the moment function is given by: $\partial_\mu[\bar{\boldsymbol{\mu}}_L](\xi) = D_1(\xi)$, we see with the chain rule that

$$\begin{aligned}\partial_\mu\mathcal{J}_\eta(t,x,\mu)(\xi) &= \ D_1(\xi)\circ\nabla_y\mathcal{J}_\eta(t,x,\bar{\boldsymbol{\mu}}_L)\\ \partial_\xi\partial_\mu\mathcal{J}_\eta(t,x,\mu)(\xi) &= \ D_2(\xi)\circ\nabla_y\mathcal{J}_\eta(t,x,\bar{\boldsymbol{\mu}}_L),\end{aligned}$$

where $D_1(\xi)$ is the matrix in $\mathbb{R}^{L_d\times d}$, and $D_2(\xi)$ is the tensor in $\mathbb{R}^{L_d\times d\times d}$ with components

$$[D_1(\xi)]_{\ell i} = \ \ell_i\xi_i^{\ell_i-1}\prod_{k\neq i}\xi_k^{\ell_k}, \quad \text{for } \xi = (\xi_i)_{i\in[\![1,d]\!]},\ \ell = (\ell_i)_{i\in[\![1,d]\!]},$$

$$[D_2(\xi)]_{\ell ij} = \ \begin{cases}\ell_i(\ell_i-1)\xi_i^{\ell_i-2}\prod_{k\neq i}\xi_k^{\ell_k}, & i=j\\ \ell_i\ell_j\xi_i^{\ell_i-1}\xi_j^{\ell_j-1}\prod_{k\neq i,j}\xi_k^{\ell_k}, & i\neq j.\end{cases}$$

The expression of the operator $\mathcal{H}_\theta$ applied to the moment neural network critic function is then given by

$$\mathcal{H}_\theta[\mathcal{J}_\eta](t,x,\mu) = \nabla_\theta\Big[\mathbb{E}_{\xi\sim\mu}\big[D_1(\xi)C_\theta(t,\xi,\bar{\boldsymbol{\mu}}_L) + \frac{1}{2}D_2^\intercal(\xi)\circ\vartheta_\theta(t,\xi,\bar{\boldsymbol{\mu}}_L)\big]\cdot\nabla_y\mathcal{J}_\eta(t,x,\bar{\boldsymbol{\mu}}_L)\Big]. \quad (3.1)$$

Here $D_2^\intercal(\xi)$ is the tensor in $\mathbb{R}^{d\times d\times L_d}$ with components $[D_2^\intercal(\xi)]_{ij\ell} = [D_2(\xi)]_{\ell ij}$.

In the algorithm, we shall use the expectation of $\mathcal{H}_\theta$, which is given from (3.1) by

$$\begin{aligned}\overline{\mathcal{H}}_\theta[\mathcal{J}_\eta](t,\mu) &:= \mathbb{E}_{\xi\sim\mu}\big[\mathcal{H}_\theta[\mathcal{J}_\eta](t,\xi,\mu)\big] \\ &= \nabla_\theta\Big[\mathbb{E}_{\xi\sim\mu}\big[D_1(\xi)C_\theta(t,\xi,\bar{\boldsymbol{\mu}}_L) + \frac{1}{2}D_2^\intercal(\xi)\circ\vartheta_\theta(t,\xi,\bar{\boldsymbol{\mu}}_L)\big]\cdot\nabla_y\mathbb{E}_{\xi\sim\mu}\big[\mathcal{J}_\eta(t,\xi,\bar{\boldsymbol{\mu}}_L)\big]\Big].\end{aligned}$$

Remark 3.1. **1.** *For complexity argument, it is crucial to rely on the above expression of the operator $\mathcal{H}_\theta$ where the differentiation is taken on the expectation $\mathbb{E}_{\xi\sim\mu}[.]$, and not the reversal: expectation of the differentiation. Indeed, in the latter case, after empirical approximation of the expectation with M samples $\xi^j\sim\mu$, $j=1,\dots,M$, one should compute by automatic differentiation*

$$\nabla_\theta\big[D_1(\xi^j)C_\theta(t,\xi^j,\bar{\boldsymbol{\mu}}_L) + \frac{1}{2}D_2^\intercal(\xi^j)\circ\vartheta_\theta(t,\xi^j,\bar{\boldsymbol{\mu}}_L)\big], \quad \nabla_y\mathcal{J}_\eta(t,\xi^j,\bar{\boldsymbol{\mu}}_L), \quad j=1,\dots,M,$$

which is very costly as M is of order 10^4. In the former case, $\overline{\mathcal{H}}_\theta[\mathcal{J}_\eta](t,\mu)$ is approximated by automatic differentiation via

$$\widehat{\mathcal{H}}_\theta^M[\mathcal{J}_\eta](t,\mu) := \nabla_\theta\Big[\tfrac{1}{M}\textstyle\sum_{j=1}^M D_1(\xi^j)C_\theta(t,\xi^j,\bar{\boldsymbol{\mu}}_L) + \tfrac{1}{2}D_2^\intercal(\xi^j)\circ\vartheta_\theta(t,\xi^j,\bar{\boldsymbol{\mu}}_L)\cdot\nabla_y\tfrac{1}{M}\textstyle\sum_{j=1}^M\mathcal{J}_\eta(t,\xi^j,\bar{\boldsymbol{\mu}}_L)\Big].$$

hence saving an order M for the complexity cost. In theory, it is also possible to choose other networks for taking into account the dependency on the distribution μ: the cylindrical network proposed in [15] could be used but some automatic differentiation are then requested to calculate $\partial_\mu\mathcal{J}_\eta(t,x,\mu)(\xi)$ and $\partial_\xi\partial_\mu\mathcal{J}_\eta(t,x,\mu)(\xi)$ for each sample of ξ leading to an explosion in the computation time.

2. *In order to calculate the term $\nabla_y\mathcal{J}_\eta$, it is necessary to explore different initial distributions, otherwise $\mathcal{J}_\eta$ only depends on t and x at convergence and the gradient is impossible to estimate.*

4 Algorithm

The actor-critic method consists in two optimization stages that are performed alternately:

(1) *Policy evaluation*: given an actor policy π_θ, evaluate its cost functional with the critic function $\mathcal{J}_\eta$ that minimizes the loss function arising from the martingale property **(MJ)** after time discretization of the interval $[0,T]$ with the time grid $\{t_k = k\Delta t, k=0,\dots,n\}$:

$$L^{PE}(\eta) = \mathbb{E}_{\alpha\sim\pi_\theta}\Big[\sum_{k=0}^{n-1}\big|g_{t_n} + \sum_{l=k}^{n-1} f_{t_l}\Delta t - \mathcal{J}_\eta(t_k,X_{t_k},\mu_{t_k})\big|^2\Delta t\Big],$$

where we set $\mu_{t_l} = \mathbb{P}_{X_{t_l}}$ for the law of X_{t_l}, and $f_{t_l} = f(X_{t_l},\mu_{t_l},\alpha_{t_l})$ as the output cost at time t_l for input state X_{t_l}, law μ_{t_l}, action $\alpha_{t_l}\sim\pi_\theta(.|t_l,X_{t_l},\mu_{t_l})$, and $g_{t_n} = g(X_{t_n},\mu_{t_n})$ the terminal output cost for input X_{t_n}, μ_{t_n}.

(2) *Policy gradient*: given a critic cost function $\mathcal{J}_\eta$, update the parameter θ of the actor by stochastic gradient descent by using the gradient, which is given from (2.2) and after time discretization by

$$\begin{aligned}G(\theta) = \mathbb{E}_{\alpha\sim\pi_\theta}\Big[\sum_{k=0}^{n-1}\nabla_\theta\log p_\theta(t_k,X_{t_k},\mu_{t_k},\alpha_{t_k})\big[\mathcal{J}_\eta(t_{k+1},X_{t_{k+1}},\mu_{t_{k+1}}) - \mathcal{J}_\eta(t_k,X_{t_k},\mu_{t_k}) \\ + f_{t_k}\Delta t\big] + \mathcal{H}_\theta[\mathcal{J}_\eta](t_k,X_{t_k},\mu_{t_k})\Big].\end{aligned}$$

In the practical implementation, we take for π_θ a Gaussian distribution with mean a moment neural network m_θ, and covariance matrix λI_d, where $\lambda = \lambda(e)$ represents the exploration parameter depending on the epoch e (gradient iteration descent) and decreasing to 0.

- We start with a batch N (of order 10) of initial distributions μ_0^i, $i = 1,\dots,N$, e.g. Gaussian distributions by varying the mean and standard deviations parameters, and sample $X_0^{i,j} \sim \mu_0^i$, $j = 1,\dots,M$ with M of order 10^4. If our ultimate goal is to learn the optimal control and function value for other families of initial distributions, the initial distributions should be sampled accordingly.
- We then run by forward induction in time: for $k = 0,\dots,n-1$:
  - Empirical estimate of $\mu_{t_k}^i$ from $(X_{t_k}^{i,j})_{j\in[\![1,M]\!]}$, for $i = 1,\dots,N$.
  - Sample $\alpha_{t_k}^{i,j} \sim \pi_\theta(.|t_k, X_{t_k}^{i,j}, \mu_{t_k}^i)$, $i \in [\![1,N]\!]$, $j \in [\![1,M]\!]$ using the exploration parameter $\lambda(e)$
  - Observe running cost $f_{t_k}^{i,j} = f(X_{t_k}^{i,j}, \mu_{t_k}^i, \alpha_{t_k}^{i,j})$, and next state $X_{t_{k+1}}^{i,j}$, $i \in [\![1,N]\!]$, $j \in [\![1,M]\!]$
- Observe final cost $g_{t_n}^{i,j} = g(X_{t_n}^{i,j}, \mu_{t_n}^i)$, $i \in [\![1,N]\!]$, $j \in [\![1,M]\!]$
- Compute the empirical mean approximation of $L^{PE}(\eta)$ on all initial distributions μ_0^i, $i \in [\![1,N]\!]$:

  $$\widetilde{L}_M^{PE}(\eta) = \frac{1}{MN}\sum_{i=1}^N\sum_{j=1}^M\sum_{k=0}^{n-1}\Big|g_{t_n}^{i,j} + \sum_{l=k}^{n-1} f_{t_l}^{i,j}\Delta t - \mathcal{J}_\eta(t_k, X_{t_k}^{i,j}, \mu_{t_k}^i)\Big|^2\Delta t,$$

  and update the critic parameter by

  $$\eta \longleftarrow \eta - \rho^C\nabla_\eta\tilde{L}_M^{PE}(\eta),$$

  where ρ^C is a learning rate. Notice that the gradient is calculated by automatic differentiation.
- Compute the empirical mean approximation of $G(\theta)$ on all initial distributions μ_0^i, $i \in [\![1,N]\!]$:

  $$\widetilde{G}_M(\theta) = \nabla_\theta\frac{1}{MN}\sum_{i=1}^N\sum_{j=1}^M\Big\{\sum_{k=0}^{n-1}\log p_\theta(t_k, X_{t_k}^{i,j}, \mu_{t_k}^i, \alpha_{t_k}^{i,j})\big[\mathcal{J}_\eta(t_{k+1}, X_{t_{k+1}}^{i,j}, \mu_{t_{k+1}}^i) - \mathcal{J}_\eta(t_k, X_{t_k}^{i,j}, \mu_{t_k}^i) + f_{t_k}^{i,j}\Delta t\big]\Big\} + \widetilde{\mathcal{H}}_\theta^M[\mathcal{J}_\eta]$$

  where

  $$\widetilde{\mathcal{H}}_\theta^M[\mathcal{J}_\eta] = \nabla_\theta\Big(\frac{1}{N}\sum_{i=1}^N\Big[\sum_{k=0}^{n-1}\Big(\frac{1}{M}\sum_{j=1}^M D_1(X_{t_k}^{i,j})C_\theta(t_k, X_{t_k}^{i,j}, \mu_{t_k}^i) + \frac{1}{2}D_2^\intercal(X_{t_k}^{i,j})\circ\vartheta_\theta(t_k, X_{t_k}^{i,j}, \mu_{t_k}^i)\Big)\cdot \nabla_y\Big(\frac{1}{M}\sum_{j=1}^M \mathcal{J}_\eta(t_k, X_{t_k}^{i,j}, \mu_{t_k}^i)\Big)\Big]\Big)$$

  and update the actor parameter by

  $$\theta \longleftarrow \theta - \rho^A\widetilde{G}_M(\theta),$$

  where ρ^A is a learning rate. Again for efficiency, it is crucial to compute by automatic differentiation the gradient after computing all the different expectations as in Remark 3.1.

The output (θ^*, η^*) are the optimal parameters obtained at convergence of the algorithm.

Remark 4.1. *On simulated data, the state at date t_{n+1} is obtained from the state at date t_n by discretizing the equation* (1.1) *with an Euler scheme.*

Remark 4.2. *Compared to classical actor-critic algorithm where one samples a trajectory for a given distribution, here the batch version of the algorithm consists in sampling and following N distributions together to estimate efficiently the value function, and the policies that depend of the state distribution. For practical reinforcement learning, this means that one has to observe a batch of initial distributions for several large populations of agents.*

Remark 4.3. *In order to check that the algorithm has effectively converged to the solution, we can use the calculated control $m_{\theta^*}(t, x, \mu)$ and apply it from different initial distributions μ_0 sampled as $(X_0^j)_{j \in [\![1,M]\!]}$ in a time discretized version of* (1.2)*. Taking discrete expectation, we can compare the result obtained to $\frac{1}{M}\sum_{j=1}^{M} \mathcal{J}_{\eta^*}(0, X_0^j, \mu_0)$. When results are very close, we can suppose that the algorithm has effectively converged to the right solution.*

Remark 4.4. *In the case where we know a priori that the running cost and terminal cost functions depend on the probability distribution μ only via its moments $\bar{\boldsymbol{\mu}}_L$, then we only need to estimate the moments of $\mu_{t_k}^i$ from $(X_{t_k}^{i,j})_{j \in [\![1,M]\!]}$, since all the other coefficients in the algorithm depend upon the measure via its moments.*

5 Numerical results

Throughout this section, we use moment neural networks with 3 hidden layers and 20 neurons on each layer, and choose the activation function tanh. The exploration parameter λ is a function of the number of gradient descent iterations (epoch number $e \le \hat{N}$):

$$\lambda(e) = (\bar{\lambda} - \underline{\lambda})\Big(1 - S\big(\frac{20e - 10\hat{N}}{\hat{N}}\big)\Big) + \underline{\lambda},$$

where $\underline{\lambda} = 0.0001$ and $\bar{\lambda} = 0.1$ and S is the sigmoid function: $S(x) = \frac{1}{1+\exp(-x)}$. In other words, it is chosen so that the exploration period with λ close to 0.1 is long enough, then λ slowly decreases to 0.0001 and stays close to that value long enough. This function is plotted on Figure 1.

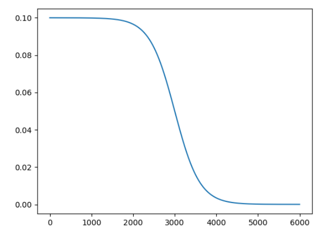


Figure 1: Exploration function λ for a number of epochs $\hat{N} = 6000$.

At a gradient descent iteration e, we sample the control as:

$$\pi_\theta(.|t, x, \mu) \sim \mathcal{N}(m_\theta(t, x, \mu), \lambda(e)\mathbb{I}_p).$$

During the gradient descent algorithm, we use the ADAM optimizer [13]. We point out that it is crucial to use two timescales approach (see [1]), for the learning rates ρ^C, ρ^A of the critic and actor updates: ρ^C should be at least one order of magnitude higher than ρ^A to get good convergence, hence the approximate critic function should evolve faster. We take a batch size $N = 10$ while the number of samples to estimate distributions is taken equal to $M = 10000$ or $M = 20000$ depending on the examples.

In the tables and figures below, we give the average analytic solution "Anal" at $t = 0$, i.e. $\mathbb{E}_{X_0 \sim \mu_0}[V(0, X_0, \mu_0)]$, and the average calculated value function "Calc": $\mathbb{E}_{X_0 \sim \mu_0}[\mathcal{J}_{\eta^*}(0, X_0, \mu_0)]$ obtained by the algorithm at $t = 0$, by varying the initial distributions μ_0. The MSE is the mean square error between the analytic and the critic value computed at $t = 0$, i.e. $\mathbb{E}_{X_0 \sim \mu_0} \big| \mathcal{J}_{\eta^*}(0, X_0, \mu_0) - V(0, X_0, \mu_0) \big|^2$, and the relative error is

$$\text{RelError} = \frac{\mathbb{E}_{X_0 \sim \mu_0} \big[\mathcal{J}_{\eta^*}(0, X_0, \mu_0) - V(0, X_0, \mu_0) \big]}{\mathbb{E}_{X_0 \sim \mu_0} [V(0, X_0, \mu_0)]}$$

We shall also plot in the one-dimensional case $d = 1$, and $A = \mathbb{R}$, the trajectories of the optimal control $t \mapsto \alpha_t^*$ vs the ones obtained from moment neural networks, i.e., $t \mapsto m_{\theta^*}(t, X_t, (\mathbb{E}[X_t^\ell])_{\ell \in [\![1,L]\!]})$.

All training times are calculated on a on GPU NVidia V100 32Go graphic card.

We consider four examples with control on the drift, including multidimensional setting and non-linear quadratic mean-field control, and one example with controlled volatility, for which we have analytic solutions to be compared with the approximations calculated from our actor-critic algorithm. We emphasize that our primary focus in on the test for the dimension related to the Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$ (which is of infinite dimension), and not on the dimension d of the state. We shall then not test dimension d up to $50, 100$ as in deep learning methods for PDEs and control without mean-field interaction, and only test for d up to 3.

5.1 Examples with controlled drift

In the four examples of this paragraph, we take $\vartheta(t, a) \equiv 0$, $C(t, a) = a$ and so $\vartheta_\theta \equiv 0$, $C_\theta = m_\theta$.

5.1.1 Systemic risk model in one dimension

We consider the model in [4]:

$$\begin{cases} b(x, \mu, a) &= \kappa(\bar{\mu} - x) + a, \quad \sigma \text{ positive constant} \\ f(x, \mu, a) &= \frac{1}{2}a^2 - qa(\bar{\mu} - x) + \frac{p}{2}(\bar{\mu} - x)^2, \qquad g(x, \mu) \;=\; \frac{c}{2}(x - \bar{\mu})^2, \end{cases}$$

for $(x, \mu, a) \in \mathbb{R} \times \mathcal{P}_2(\mathbb{R}) \times \mathbb{R}$, with some positive constants κ, q, p, $c > 0$, $q^2 \le p$. Here we denote by $\bar{\mu} := \mathbb{E}_{\xi \sim \mu}[\xi]$.

In this linear quadratic (LQ) model, the value function is explicitly given by

$$V(t, x, \mu) = K(t)(x - \bar{\mu})^2 + \sigma^2 R(t), \tag{5.1}$$

where

$$K(t) = -\frac{1}{2}\Big[\kappa + q - \sqrt{\Delta}\frac{\sqrt{\Delta}\sinh(\sqrt{\Delta}(T-t)) + (\kappa + q + c)\cosh(\sqrt{\Delta}(T-t))}{\sqrt{\Delta}\cosh(\sqrt{\Delta}(T-t)) + (\kappa + q + c)\sinh(\sqrt{\Delta}(T-t))}\Big],$$

with $\sqrt{\Delta} = \sqrt{(\kappa + q)^2 + p - q^2}$, and

$$R(t) = \frac{\sigma^2}{2}\ln\Big[\cosh(\sqrt{\Delta}(T-t)) + \frac{\kappa + q + c}{\sqrt{\Delta}}\sinh(\sqrt{\Delta}(T-t))\Big] - \frac{\sigma^2}{2}(\kappa + q)(T - t),$$

while the optimal control is given by

$$\alpha_t^* = (2K(t) + q)(\mathbb{E}[X_t] - X_t), \quad 0 \le t \le T.$$

The parameters of the model are fixed to the following values: $\kappa = 0.6$, $\sigma = 1$, $p = c = 2$, $q = 0.8$, $T = 1$. We take a number of time steps $n = 100$, $M = 10000$. At each gradient iteration, the initial distribution is sampled with

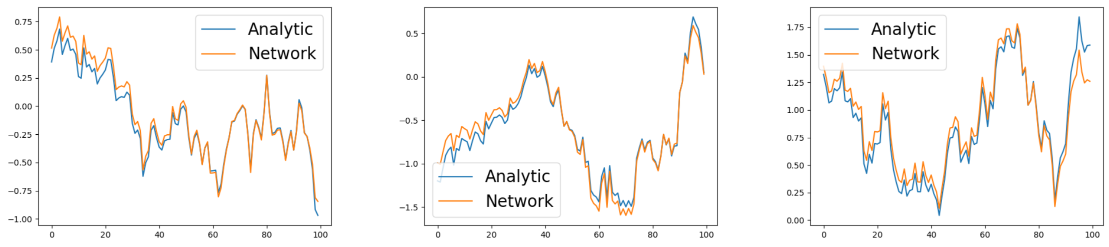
$$X_0 \sim \mu_0 = \upsilon_0 \, \mathcal{N}(0, 1),$$

where υ_0^2 is sampled at each iteration according to the uniform distribution on $[0, 1]$ for each element of the batch. In the table, we give the results obtained in simulation with $\upsilon_0^2 \in \{0, \frac{1}{10}, \ldots, 1\}$ after training with $L = 2$ moments, using $\hat{N} = 6000$ gradient iterations.

v_0^2	0.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Anal	0.3870	0.4095	0.4321	0.4546	0.4772	0.4997	0.5223	0.5448	0.5674	0.5900
Calc	0.3958	0.4198	0.4421	0.4642	0.4875	0.5112	0.5341	0.5593	0.5858	0.6082
MSE	0.0001	0.0001	0.0002	0.0002	0.0004	0.0005	0.0005	0.0006	0.0005	0.0007

Table 1: Results for the systemic model using $L = 2$, $\rho^A = 0.0005$, $\rho^C = 0.01$. Training time is 106863s.

On Figure 2, we plot 3 trajectories of the optimal control and the ones calculated with moment neural networks, and we observe that the control is very well estimated.



Trajectory 1 Trajectory 2 Trajectory 3

Figure 2: Trajectories of control with $v_0^2 = 0.9$

In this LQ example, we know that the suitable number of moments to take is $L = 2$. In the real test case, we do not know which L to take. In Table 2, we give the results with $L = 4$, which shows that the results are also very accurate and do not depend on L being small.

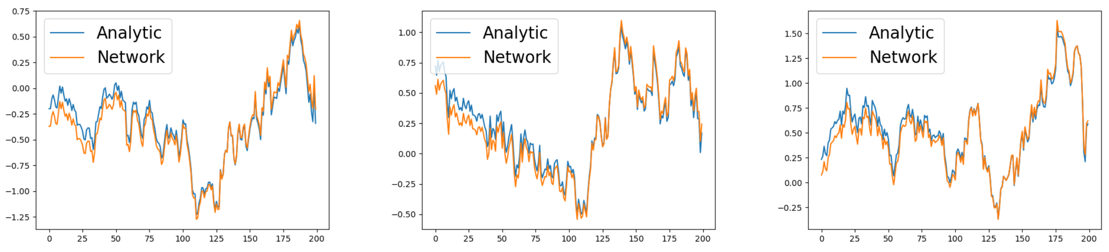
v_0^2	0.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Anal	0.3869	0.4095	0.4320	0.4546	0.4771	0.4997	0.5222	0.5448	0.5674	0.5899
Calc	0.3917	0.4159	0.4376	0.4588	0.4801	0.5023	0.5249	0.5471	0.5688	0.5891
MSE	0.0000	0.0000	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0002	0.0003

Table 2: Results for the systemic case using $L = 4$, $\rho^A = 0.0005$, $\rho^C = 0.01$. Training time is 115183s.

Finally, we can show that the function value and the control are stable by taking $n = 200$. In the table 3 we give the results obtained with $n = 200$, $L = 2$. With the same parameterization, we obtain results closer to the analytic one than with $n = 100$, reflecting the convergence of the Euler scheme to the continuous solution. Some trajectories of the control are added on figure 3.

v_0^2	0.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Anal	0.3869	0.4095	0.4320	0.4546	0.4771	0.4997	0.5222	0.5448	0.5674	0.5899
Calc	0.3850	0.4071	0.4292	0.4521	0.4745	0.4966	0.5193	0.5416	0.5648	0.5874
MSE	0.0000	0.0000	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0002	0.0003

Table 3: Results for the systemic case using $L = 2$, $n = 200$, $\rho^A = 0.0005$, $\rho^C = 0.01$. .



Trajectory 1 Trajectory 2 Trajectory 3

Figure 3: Trajectories of control with $v_0^2 = 0.9$, $n = 200$.

5.1.2 An optimal trading example

We consider an optimal trading model taken from [7]:

$$\begin{cases} b(x,\mu,a) &= a, \quad \sigma \text{ positive constant} \\ f(x,\mu,a) &= a^2 + 2Pa, \qquad g(x,\mu) \;=\; \gamma(x-\bar{\mu})^2, \end{cases}$$

for $(x,\mu,a) \in \mathbb{R} \times \mathcal{P}_2(\mathbb{R}) \times \mathbb{R}$, with $P > 0$ the constant transaction price per trading, and $\gamma > 0$ the risk aversion parameter.

In this LQ framework, the value function has the form as in (5.1) with

$$K(t) \;=\; \frac{\gamma}{1+\gamma(T-t)}, \qquad R(t) \;=\; \sigma^2 \log(1+\gamma(T-t)) - P^2(T-t),$$

while the optimal control is given by

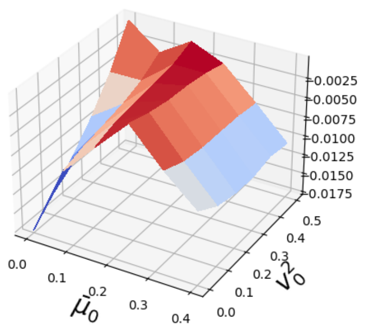
$$\alpha_t^* \;=\; -K(t)(X_t - \mathbb{E}[X_t]) - P, \quad 0 \le t \le T.$$

We take the following parameters : $P = 3, \gamma = 3, \sigma = 1, T = 0.5$, and $n = 100$, $M = 10000$. At each gradient iteration, the initial distribution is sampled with

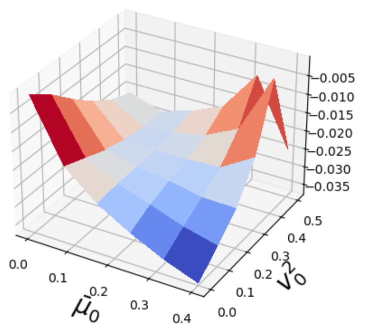
$$X_0 \sim \;\mu_0 = \bar{\mu}_0 + v_0 \mathcal{N}(0,1),$$

where $(\bar{\mu}_0, v_0^2)$ are sampled from $(0.4\mathcal{U}([0,1]), 0.5\mathcal{U}([0,1]))$ for each element of the batch.

The relative error is plotted on Figure 4 by varying $(\bar{\mu}_0, v_0)$, while the trajectories of the control (optimal vs moment neural netwok) are plotted on Figure 5. Again, in this LQ example, the suitable number of moments to take is $L = 2$, and when we increase L, convergence is more difficult to achieve and results become more instable and may depend on the run with the same hyper-parameters. Nevertheless we manage to obtain very good results with $L = 4$ as shown on Figure 4.

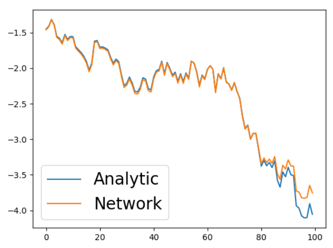


$L = 2$, training time is $110900s$

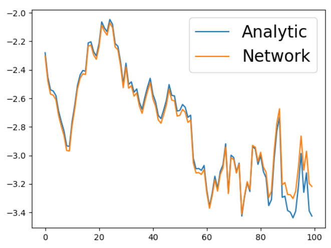


$L = 4$, Training time is $120000s$

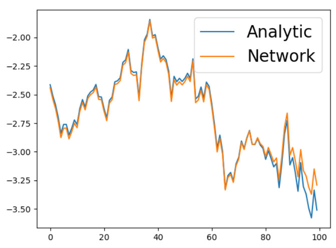
Figure 4: Relative error with $\rho^A = 0.0005$, $\rho^C = 0.01$, $\hat{N} = 9000$.



Trajectory 1



Trajectory 2



Trajectory 3

Figure 5: Controlled trajectories for $\bar{\mu}_0 = 0.16$, $v_0^2 = 0.1$, $\rho^A = 0.0005$, $\rho^C = 0.01$, $L = 2$.

5.1.3 A non linear quadratic mean-field control

We construct an ad-hoc mean-field control model with

$$\begin{cases} b(t,x,\mu,a) & = \beta(t,x,\mu) + a, \quad \sigma \text{ positive constant} \\ f(x,\mu,a) & = F(t,x,\mu) + \frac{1}{2}|a|^2, \quad g(x,\mu) = \mathbb{E}_{\xi\sim\mu}[w(x-\xi)] \end{cases}$$

for some smooth C^2 even function w on $\mathbb{R}$, e.g. $w(x) = \cos(x)$, and F is a function to be chosen later.
In this case, the optimal feedback control valued in $A = \mathbb{R}$ is given by

$$\begin{aligned} \mathfrak{a}^\star(t,x,\mu) &= \hat{\mathrm{a}}(t,x,\mathcal{U}(t,x,\mu)) = -\mathcal{U}(t,x,\mu) = -\partial_\mu v(t,\mu)(x) \\ & \quad \text{with } v(t,\mu) = \mathbb{E}_{\xi\sim\mu}[V(t,\xi,\mu)], \end{aligned}$$

and V is solution to the Master Bellman equation (see section 6.5.2 in [2]):

$$\begin{aligned} & \partial_t V(t,x,\mu) + \big(\beta(t,x,\mu) - \mathcal{U}(t,x,\mu)\big)\partial_x V(t,x,\mu) + \frac{\sigma^2}{2}\partial^2_{xx}V(t,x,\mu) \\ & + \mathbb{E}_{\xi\sim\mu}\Big[\big(\beta(t,\xi,\mu) - \mathcal{U}(t,\xi,\mu)\big)\partial_\mu V(t,x,\mu)(\xi) + \frac{\sigma^2}{2}\partial_{x'}\partial_\mu V(t,x,\mu)(\xi)\Big] \\ & \qquad + F(t,x,\mu) + \frac{1}{2}|\mathcal{U}(t,x,\mu)|^2 = 0, \end{aligned} \tag{5.2}$$

with the terminal condition $V(T,x,\mu) = g(x,\mu)$.

We look for a solution to the Master equation in the form: $V(t,x,\mu) = e^{T-t}\mathbb{E}_{\xi\sim\mu}[w(x-\xi)]$. For such function V, we have $\partial_t V(t,x,\mu) = -V$,

$$\begin{aligned} \partial_x V(t,x,\mu) &= e^{T-t}\mathbb{E}_{\xi\sim\mu}[w'(x-\xi)], \quad \partial^2_{xx}V(t,x,\mu) = e^{T-t}\mathbb{E}_{\xi\sim\mu}[w''(x-\xi)] \\ \partial_\mu V(t,x,\mu)(\xi) &= -e^{T-t}w'(x-\xi), \quad \partial_{x'}\partial_\mu V(t,x,\mu)(\xi) = e^{T-t}w''(x-\xi), \end{aligned}$$

and

$$\mathcal{U}(t,x,\mu) = e^{T-t}\mathbb{E}_{\xi\sim\mu}[w'(x-\xi) - w'(\xi-x)] = 2e^{T-t}\mathbb{E}_{\xi\sim\mu}[w'(x-\xi)] = 2\partial_x V(t,x,\mu).$$

since w is even. By plugging these derivatives expressions of V into the l.h.s. of (5.4), we then see that by choosing F equal to

$$\begin{aligned} F(t,x,\mu) &= e^{T-t}\mathbb{E}_{\xi\sim\mu}\Big[(w-\sigma^2 w'')(x-\xi) + (\beta(t,\xi,\mu) - \beta(t,x,\mu))w'(x-\xi)\Big] \\ &\quad - 2e^{2(T-t)}\mathbb{E}_{(\xi,\xi')\sim\mu\otimes\mu}\big[w'(x-\xi)w'(\xi-\xi')\big], \end{aligned}$$

the function V satisfies the Master Bellman equation, hence is the value function to the mean-field control problem.

Actually, with the choice of $w(x) = \cos(x)$, and using trigonometric relations, we have

$$\begin{aligned} F(t,x,\mu) &= \cos(x)\Big[e^{T-t}\left((1+\sigma^2)\mathbb{E}_{\xi\sim\mu}[\cos(\xi)] + \mathbb{E}_{\xi\sim\mu}[\sin(\xi)\beta(t,\xi,\mu)] - \beta(t,x,\mu)\mathbb{E}_{\xi\sim\mu}[\sin(\xi)]\right) \\ &\quad - 2e^{2(T-t)}\left(\mathbb{E}_{\xi\sim\mu}[\sin(\xi)\cos(\xi)]\mathbb{E}_{\xi\sim\mu}[\sin(\xi)] - \mathbb{E}_{\xi\sim\mu}[\sin^2(\xi)]\mathbb{E}_{\xi\sim\mu}[\cos(\xi)]\right)\Big] \\ &\quad + \sin(x)\Big[e^{T-t}\left((1+\sigma^2)\mathbb{E}_{\xi\sim\mu}[\sin(\xi)] - \mathbb{E}_{\xi\sim\mu}[\beta(t,\xi,\mu)\cos(\xi)] + \beta(t,x,\mu)\mathbb{E}_{\xi\sim\mu}[\cos(\xi)]\right) \\ &\quad - 2e^{2(T-t)}\left(\mathbb{E}_{\xi\sim\mu}[\sin(\xi)\cos(\xi)]\mathbb{E}_{\xi\sim\mu}[\cos(\xi)] - \mathbb{E}_{\xi\sim\mu}[\cos^2(\xi)]\mathbb{E}_{\xi\sim\mu}[\sin(\xi)]\right)\Big]. \end{aligned}$$

Notice that such function $F(t,x,\mu)$ has non polynomial moments dependence w.r.t. the measure μ.

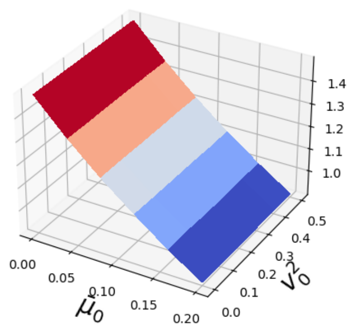
For the test case, we take

$$\beta(t,x,\mu) = \kappa(\bar{\mu} - x), \tag{5.3}$$

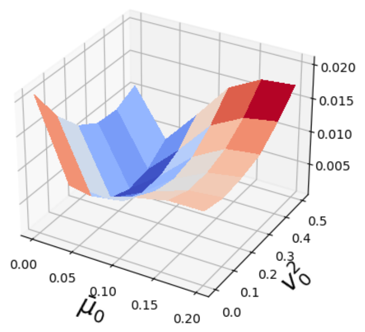
with the parameters $\kappa = \sigma = 1$, $T = 0.4$, $n = 40$, $M = 20000$. At each gradient iteration, the initial distribution is sampled with

$$X_0 \sim \mu_0 = \bar{\mu}_0 + \upsilon_0\mathcal{N}(0,1)$$

where $(\bar{\mu}_0, \upsilon_0^2)$ is sampled from $(0.2\mathcal{U}([0,1]), 0.5\mathcal{U}([0,1]))$ for each element of the batch. On Figure 6, we give the analytic solution depending on $(\bar{\mu}_0, \upsilon_0^2)$ and the relative error obtained by the algorithm. We observe that the results are quite accurate.



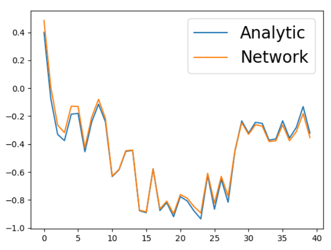
Analytic value



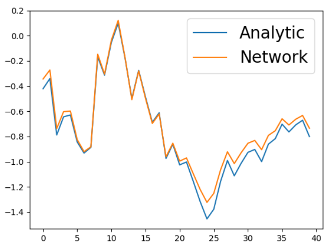
Relative error

Figure 6: Non LQ model using $\hat{N} = 9000$ gradient iterations, $L = 3$, $\rho^A = 0.0005$, $\rho^C = 0.02$. Training time is 45200s.

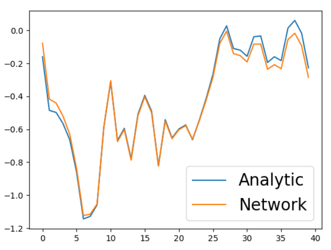
We plot in Figure 7 the trajectories of the optimal control vs the moment neural network, and observe that they are very close.



Trajectory 1



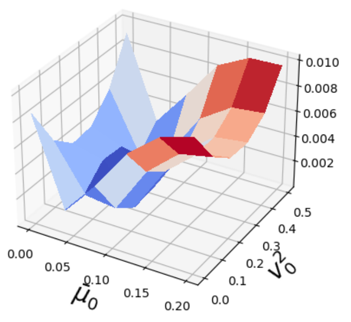
Trajectory 2



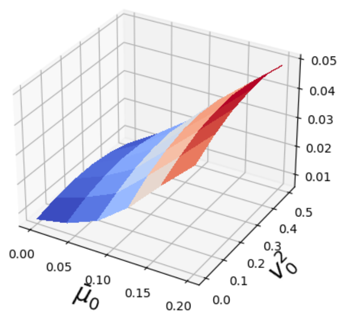
Trajectory 3

Figure 7: Control trajectories in a non LQ model using $\hat{N} = 9000$ gradient iterations, $L = 3$, $\rho^A = 0.0005$, $\rho^C = 0.02$, for $(\bar{\mu}_0, v_0^2) = (0.16, 0.1)$.

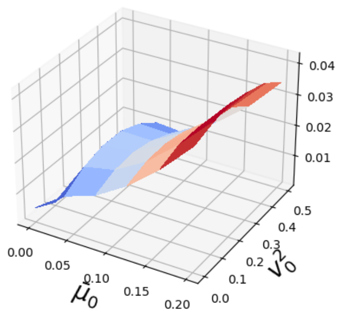
Figure 8 shows that using $L = 2$ or 3 is optimal in terms of relative error, while the convergence is more difficult to achieve for high values of L.



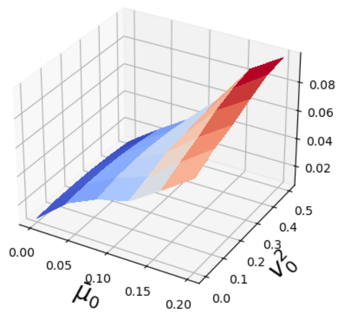
$L = 2$



$L = 4$



$L = 5$



$L = 6$

Figure 8: Relative error in a non LQ model for different values of L. $\hat{N} = 9000$ gradient iterations, $\rho^A = 0.0005$, $\rho^C = 0.02$.

5.1.4 A multi-dimensional LQ example

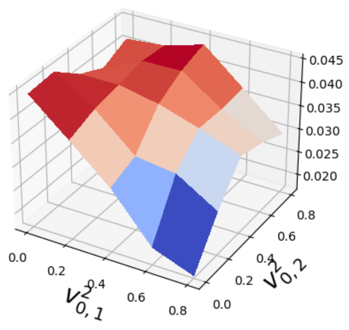
We consider a multi-dimensional extension of the LQ systemic risk model in Section 5.1.1, by supposing that on each dimension, the dynamics satisfies the same equation with independent Brownian motions, and that the cost functions are the sum over each component of the cost function in the univariate case. In this case, the value function is given by $V(t,x,\mu) = \sum_{i=1}^{d} V_1(t,x_i,\mu_i)$, for $t \in [0,T]$, $x = (x_i)_{i\in[\![1,d]\!]} \in \mathbb{R}^d$, μ_i is the i-th marginal law of $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, and V_1 is the value function in the univariate model given by (5.1).

We keep the same parameters as in Section 5.1.1, with a number of time steps $n = 50$, $M = 10000$, $L = 2$, and test in dimension $d = 2$ and 3. At each gradient iteration, the initial distribution is sampled from

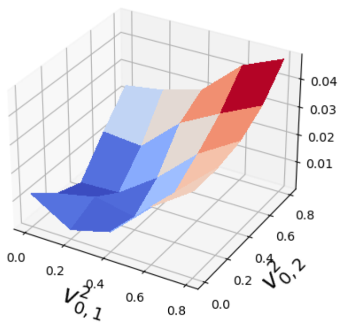
$$X_0 \sim \ \mu_0 = \mathcal{N}(0, \upsilon_0),$$

where υ_0 is the diagonal $d \times d$-matrix with diagonal elements $\upsilon_{0,i}$ sampled from $u_i\mathcal{U}([0,1])$, with constants $u_i \in [0,1]$, $i = 1,\ldots,d$, for each element of the batch.

We plot in Figure 9 the relative error in dimension $d = 2$ by varying $(\upsilon_{0,1}, \upsilon_{0,2})$, and for $L = 2$ and $L = 4$. In Figure 10, we plot the relative error in dimension $d = 3$ by varying $(\upsilon_{0,1}, \upsilon_{0,2})$, and for $\upsilon_{0,3} = 0$.



$L = 2$, $\hat{N} = 6000$, $\rho^A = 0.001$



$L = 4$, $\hat{N} = 9000$, $\rho^A = 0.0005$

Figure 9: Relative error in dimension $d = 2$, with $\rho^C = 0.01$.

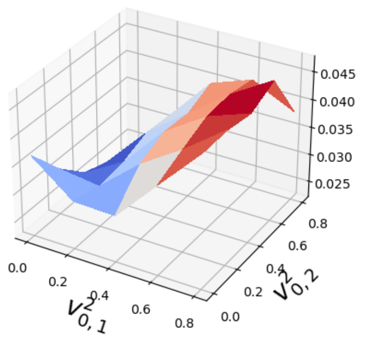


Figure 10: Relative error for $L = 2$ in dimension $d = 3$, with $\hat{N} = 9000$ gradient iterations, $\rho^A = 0.0005$, $\rho^C = 0.01$. Training time is 144000s.

5.2 A non LQ example with controlled volatility

We consider a one-dimensional model with

$$\begin{cases} b(t,x,\mu,a) & = \ \beta(t,x,\mu), \quad \sigma(t,x,\mu,a) \ = \ a, \\ f(x,\mu,a) & = \ F(t,x,\mu) + \frac{1}{2}P|a|^2 - a, \quad g(x,\mu) \ = \ \mathbb{E}_{\xi\sim\mu}[w(x-\xi)], \end{cases}$$

where P is a positive constant, w is a smooth C^2 even function on $\mathbb{R}$, e.g. $w(x) = \cos(x)$, and F is a function to be chosen later. Notice that $C_\theta \equiv 0$, and

$$\vartheta_\theta(t,x,\mu) \ = \ m_\theta(t,x,\mu)^2 + \lambda,$$

where $\lambda = \lambda(e)$ (depending on the epoch e) is the exploration parameter of $\pi_\theta(.|t,x,\mu) = \mathcal{N}(m_\theta(t,x,\mu),\lambda)$.

In this model, the optimal feedback control valued in $A = \mathbb{R}$ is given by

$$\mathfrak{a}^\star(t,x,\mu) = \ \hat{\mathrm{a}}(t,x,\partial_x\mathcal{U}(t,x,\mu)) \ = \ \frac{1}{P + \partial_x\mathcal{U}(t,x,\mu)},$$
$$\text{with} \quad \mathcal{U}(t,x,\mu) \ = \ \partial_\mu v(t,\mu)(x), \quad v(t,\mu) \ = \ \mathbb{E}_{\xi\sim\mu}[V(t,\xi,\mu)],$$

and V is solution to the Master Bellman equation:

$$\begin{aligned} & \partial_t V(t,x,\mu) + \beta(t,x,\mu)\partial_x V(t,x,\mu) + \frac{1}{2}\frac{1}{(P+\partial_x\mathcal{U}(t,x,\mu))^2}\partial^2_{xx}V(t,x,\mu) \\ & + \mathbb{E}_{\xi\sim\mu}\Big[\beta(t,\xi,\mu)\partial_\mu V(t,x,\mu)(\xi) + \frac{1}{2}\frac{1}{(P+\partial_x\mathcal{U}(t,\xi,\mu))^2}\partial_{x'}\partial_\mu V(t,x,\mu)(\xi)\Big] \\ & + F(t,x,\mu) + \frac{1}{2}\frac{P}{(P+\partial_x\mathcal{U}(t,x,\mu))^2} - \frac{1}{P+\partial_x\mathcal{U}(t,x,\mu)} \ = \ 0, \end{aligned} \tag{5.4}$$

with the terminal condition $V(T,x,\mu) = g(x,\mu)$.

We look for a solution to the Master equation in the form: $V(t,x,\mu) = e^{T-t}\mathbb{E}_{\xi\sim\mu}[w(x-\xi)]$, and by similar calculations as in Section 5.1.3, we would have $\mathcal{U}(t,x,\mu) = 2\partial_x V(t,x,\mu) = 2e^{T-t}\mathbb{E}_{\xi\sim\mu}[w'(x-\xi)]$. Therefore, with $w(x) = \cos(x)$, and by choosing F equal to

$$\begin{aligned} F(t,x,\mu) = & -\frac{P}{2(P-2e^{T-t}\mathbb{E}_{\xi\sim\mu}[\cos(x-\xi)])^2} + \frac{1}{P-2e^{T-t}\mathbb{E}_{\xi\sim\mu}[\cos(x-\xi)]} + \\ & \mathbb{E}_{\xi\sim\mu}[\cos(x-\xi)]e^{T-t}(1+\frac{1}{2}\frac{1}{(P-2e^{T-t}\mathbb{E}_{\xi\sim\mu}[\cos(x-\xi)])^2}) + \\ & e^{T-t}\mathbb{E}_{\xi\sim\mu}[(\beta(t,x,\mu)-\beta(t,\xi,\mu))\sin(x-\xi)] + \\ & e^{T-t}\mathbb{E}_{\xi\sim\mu}[\cos(x-\xi)\frac{1}{2}\frac{1}{(P-2e^{T-t}\mathbb{E}_{\xi'\sim\mu}[\cos(\xi-\xi')])^2}], \end{aligned}$$

the function V satisfies the Master Bellman equation, hence is the value function to the mean-field control problem. We notice again that such function F has generalized moments dependence, which are not polynomial.

To be easily computable, the function F can be rewritten using trigonometric relations as

$$\begin{aligned} F(t,x,\mu) = & -\frac{P}{2}\frac{1}{(P-2e^{T-t}[\cos(x)\overline{\cos}_\mu + \sin(x)\overline{\sin}_\mu])^2} + \frac{1}{P-2e^{T-t}[\cos(x)\overline{\cos}_\mu + \sin(x)\overline{\sin}_\mu]} \\ & + \ e^{T-t}[\cos(x)\overline{\cos}_\mu + \sin(x)\overline{\sin}_\mu](1+\frac{1}{2}\frac{1}{(P-2e^{T-t}[\cos(x)\overline{\cos}_\mu + \sin(x)\overline{\sin}_\mu])^2}) \\ & + \ e^{T-t}[\beta(t,x,\mu)\sin(x)\overline{\cos}_\mu - \beta(t,x,\mu)\cos(x)\overline{\sin}_\mu \\ & \quad - \ \mathbb{E}_{\xi\sim\mu}[\beta(t,\xi,\mu)\cos(\xi)]\sin(x) + \mathbb{E}_{\xi\sim\mu}[\beta(t,\xi,\mu)\sin(\xi)]\cos(x)] \\ & + \ \frac{e^{T-t}}{2}\cos(x)\mathbb{E}_{\xi\sim\mu}[\frac{\cos(\xi)}{2[P-2e^{T-t}(\cos(\xi)\overline{\cos}_\mu + \sin(\xi)\overline{\sin}_\mu)]^2}] \\ & + \ \frac{e^{T-t}}{2}\sin(x)\mathbb{E}_{\xi\sim\mu}[\frac{\sin(\xi)}{2[P-2e^{T-t}(\cos(\xi)\overline{\cos}_\mu + \sin(\xi)\overline{\sin}_\mu)]^2}] \end{aligned}$$

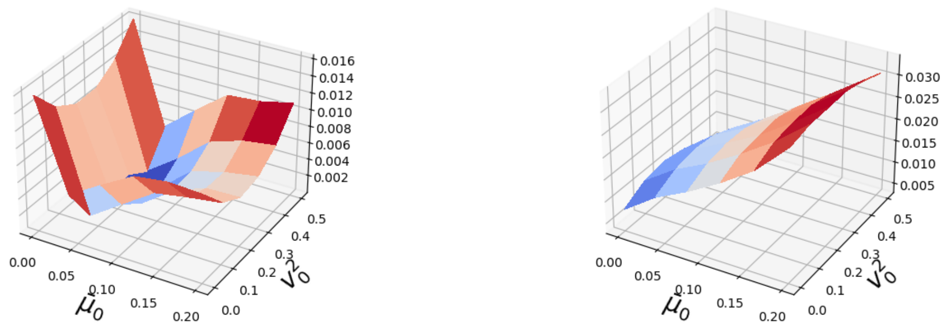
with the notations: $\overline{\cos}_\mu := \mathbb{E}_{\xi\sim\mu}[\cos(\xi)]$, $\overline{\sin}_\mu := \mathbb{E}_{\xi\sim\mu}[\sin(\xi)]$.

We take $P = 2.2e^T$ so that the control is bounded, the same trend β as in (5.3), and parameters as in section 5.1.3: $\kappa = \sigma = 1$, $T = 0.4$, $n = 40$, $M = 20000$. At each gradient iteration, the initial distribution is sampled with

$$x_0 \sim \ \mu_0 = \bar{\mu}_0 + \upsilon_0 \mathcal{N}(0,1)$$

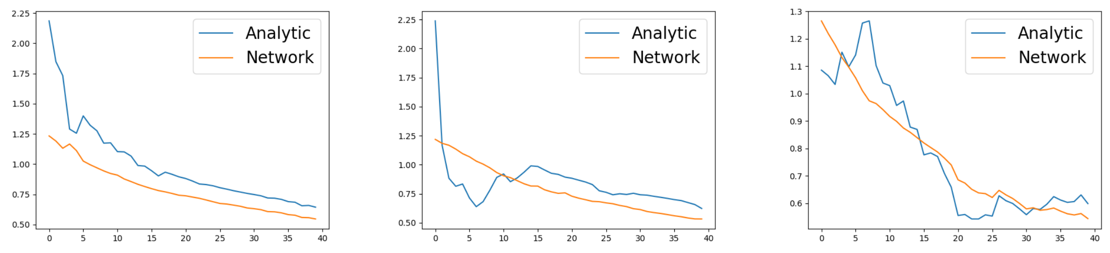
where $(\bar{\mu}_0, \upsilon_0^2)$ is sampled from $(0.2U([0,1]), 0.5U([0,1]))$. Controlling the volatility is more difficult than controlling the trend, and it is crucial for the method that ρ^A is very small. We take $\hat{N} = 9000$ gradient iterations. Training time with $L = 3$ is 67228s, while it takes 69073s for $L = 4$.

On Figure 11, we give the relative error obtained with $L = 3$ and $L = 4$. Notice that with $L = 3$, ρ^A is small and that we have to take ρ^A even smaller with $L = 4$. The relative error is small, but the control is not as well approximated as in the controlled drift example, as shown in Figure 12.



$L = 3$, $\rho^A = 0.0005$, $\rho^C = 0.025$ $L = 4$, $\rho^A = 0.0002$ and $\rho^C = 0.01$

Figure 11: Relative error in a non LQ model with controlled volatility.



Trajectory 1 Trajectory 2 Trajectory 3

Figure 12: Control trajectories in a non LQ model with controlled volatility, using $L = 3$, with $(\bar{\mu}_0, \upsilon_0^2) = (0.04, 0.4)$.

6 Conclusion

In this study, we have presented a robust effective resolution to the challenging problem of mean-field control within a partially model-free continuous-time framework. Leveraging policy gradient techniques and actor-critic algorithms, our approach has demonstrated the valuable role of moment neural networks in the sampling of distributions. We have illustrated the performance and accuracy of our algorithms for various examples including fully non linear MFC problems with generalized (not polynomial) moments. We emphasized the significance of maintaining a low number of moments (typically two or three) as the consideration of too large number of moments (even if it is in theory optimal) would lead in practice to instability and local minimal in the stochastic gradient descent. This overfitting phenomenon is analog to what is observed when using large number of layers and neurons in the numerical resolution of PDEs and classical stochastic control problems. Finally, we also point out the critical role played by fine-tuning learning rates for actor and critic updates.

Subsequent developments could encompass the extension to non-separable forms within the state and control components of drift and diffusion coefficients. Furthermore, a compelling direction for further investigation could involve mean-field dynamics governed by jump diffusion processes, where the intensities of the jumps remain unknown.

References

[1] Andrea Angiuli, Jean-Pierre Fouque, and Mathieu Laurière. “Unified Reinforcement Q-Learning for Mean Field Game and Control Problems”. In: *Mathematics of Control, Signals and Systems* 34 (2022), pp. 217–271.

[2] René Carmona and François Delarue. *Probabilistic theory of mean-field games: vol. I, Mean field FBSDEs, Control, and Games*. Springer, 2018.

[3] René Carmona and François Delarue. *Probabilistic Theory of Mean Field Games: vol. II, Mean Field game with common noise and Master equations*. Springer, 2018.

[4] René Carmona, Jean-Pierre Fouque, and Li-Hsien Sun. “Mean field games and systemic risk”. In: *Commun. Math. Sci.* 13.4 (2015), pp. 911–933.

[5] René Carmona and Mathieu Laurière. “Convergence analysis of machine learning algorithms for the numerical solution of mean-field control and games: II-the finite horizon case”. In: *Annals of Applied Probability* 32.6 (2022), pp. 4065–4105.

[6] René Carmona, Mathieu Laurière, and Zongjun Tan. “Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning”. In: *to appear in Annals of Applied Probability* (2022).

[7] Noufel Frikha, Maximilien Germain, Mathieu Laurière, Huyên Pham, and Xuanye Song. “Actor-critic learning for mean-field control in continuous time”. In: *arXiv:2303.06993* (2023).

[8] Maximilien Germain, Mathieu Laurière, Huyên Pham, and Xavier Warin. “DeepSets and their derivative networks for solving symmetric PDEs”. In: *Journal of Scientific Computing* 91, article 63 (2022).

[9] Maximilien Germain, Joseph Mikael, and Xavier Warin. “Numerical resolution of McKean-Vlasov FBSDEs using neural networks”. In: *Methodology and Computing in Applied Probability* (2022).

[10] Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. “Mean field controls with Q-learning for cooperative MARL: convergence and complexity analysis”. In: *SIAM Journal on Mathematics of Data Science* (2021).

[11] Jiequn Han, Ruimeng Hu, and Jihao Long. “Learning high-dimensional McKean-Vlasov forward-backward stochastic differential equations with general distribition dependence”. In: *arXiv: 2204.11924* (2022).

[12] Yanwei Jia and Xun Yu Zhou. “Policy gradient and actor critic learning in continuous time and space: theory and algorithms”. In: *to appear in Journal of Machine Learning Research* (2021).

[13] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[14] Huyên Pham and Xavier Warin. “Mean-field neural networks-based algorithms for McKean-Vlasov control problems”. In: *arXiv: 2212.11518* (2022).

[15] Huyên Pham and Xavier Warin. “Mean-field neural networks: learning mappings on Wasserstein space”. In: *arXiv preprint arXiv:2210.15179, to appear in Neural networks* (2022).

[16] Christoph Reisinger, Wolfgang Stockingeatr, and Yufei Zhang. “A fast iterative PDE-based algorithm for feedback controls of nonsmooth mean-field control problems”. In: *arXiv: 2108.06740* (2021).

[17] Lars Ruthotto, Stanley Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu Fung. “A machine learning framework for solving high-dimensional mean field game and mean field control problems”. In: *Proc. Natl. Acad. Sci. USA* 117.17 (2020), pp. 9183–9193.

[18] Yeneng Sun. “The exact law of large numbers via Fubini extension and characterization of insurable risks”. In: *Journal of Economic Theory* 126 (2006), pp. 31–69.

[19] Richard Sutton and Andrew Barto. *Reinforcement learning: an introduction.* Cambridge, MA:MIT, 2018, 2nd edition.

[20] Lukasz Szpruch, Tanut Treetanthiploet, and Yufei Zhang. “Optimal scheduling of entropy regularizer for continuous-time linear-quadratic reinforcement learning”. In: *SIAM Journal on Control and Optimization* 62.1 (2024), pp. 335–166.

[21] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. “Reinforcement learning in continuous time and space: A stochastic control approach”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 8145–8178.

[22] Xavier Warin. “Quantile and moment neural networks for learning functionals of distributions”. In: *arXiv preprint arXiv:2303.11060* (2023).

Funding
There is no funding.

Competing Interests
The authors declare no competing interests.

Contributions
Huyên PHAM: Methodology, writing, editing.
Xavier WARIN: Algorithms, Numerical implementations, writing, editing.
All authors read and approved the final manuscript

Data Availability Statement
All data generated or analysed during this study are included.