# Numerical Methods for Mean Field Games

## *Lecture 4*
## *Deep Learning Methods: Part I*
## *MFC and MKV FBSDE*

Mathieu Laurière

New York University Shanghai

UM6P Vanguard Center, Université Cadi AYYAD,
University Côte d'Azur, & GE2MI
Open Doctoral Lectures
July 5 – 7, 2023

# Outline

## Summary so far

Numerical methods discussed so far:

- ODE system for LQ setting
- FBPDE system
- FBSDE system

# “Classical” Numerical Methods for MFG: Some references

Some methods based on the deterministic approach to MFG/MFC:

- Finite difference & Newton method: [Achdou and Capuzzo-Dolcetta, 2010], [Achdou et al., 2012], ...
- (Semi-)Lagrangian approach: [Carlini and Silva, 2014, Carlini and Silva, 2015], [Carlini and Silva, 2018], [Calzola et al., 2022], ...
- Augmented Lagrangian & ADMM: [Benamou and Carlier, 2015], [Andreev, 2017a], [Achdou and Laurière, 2016], ...
- Primal-dual algo.: [Briceño Arias et al., 2018], [Briceño Arias et al., 2019], ...
- Gradient descent based methods [Laurière and Pironneau, 2016], [Pfeiffer, 2016], [Lavigne and Pfeiffer, 2022], ...
- Monotone operators [Almulla et al., 2017], [Gomes and Saúde, 2018], [Gomes and Yang, 2020], ...
- Policy iteration [Cacace et al., 2021], [Cui and Koeppl, 2021], [Camilli and Tang, 2022], [Tang and Song, 2022], [Laurière et al., 2023], ...
- Finite elements [Benamou and Carlier, 2015], [Andreev, 2017b], ...

# “Classical” Numerical Methods for MFG: Some references

Some methods based on the probabilistic approach to MFG/MFC:

- Cubature [de Raynal and Trillos, 2015], . . .
- Markov chain approximation: [Bayraktar et al., 2018], . . .
- Probabilistic approach and Picard: [Chassagneux et al., 2019], [Angiuli et al., 2019], . . .
- Probabilistic approach and regression: [Balata et al., 2019], . . .
- . . .

# “Classical” Numerical Methods for MFG: Shortcomings

Many of these methods are very efficient and have been analyzed in detail

However, they are usually limited to problems with:

- (relatively) small dimension
- (relatively) simple structure

$\Rightarrow$ motivations to develop machine learning methods (see lectures 4, 5, 6)

# Deep learning

- In this lecture and the following one, we will use deep learning to solve MFGs
- At a high level, there are two main ingredients:
  - Approximation using deep neural networks
  - Minimization of a loss function using stochastic gradient descent
- Many variants and refinements, ...
- See e.g. [LeCun et al., 2015, Goodfellow et al., 2016], ...

# Ingredient 1: Neural Networks

- **Goal:** Minimize over $\varphi(\cdot)$, $\mathbb{J}(\varphi) := \mathbb{E}_\xi[\mathbb{L}(\varphi, \xi)]$
- Example: Regression: $\xi = (x, f(x))$ for some $f$, $\mathbb{L}(\varphi, \xi) = \|\varphi(x) - f(x)\|^2$

- **Goal:** Minimize over $\varphi(\cdot)$, $\mathbb{J}(\varphi) := \mathbb{E}_\xi[\mathbb{L}(\varphi, \xi)]$
- Example: Regression: $\xi = (x, f(x))$ for some $f$, $\mathbb{L}(\varphi, \xi) = \|\varphi(x) - f(x)\|^2$
- **Idea:** Instead of min. over all $\varphi(\cdot)$, min. over parameters $\theta$ of $\varphi_\theta(\cdot)$
- Example: Feedforward fully-connected neural network:
  - $\varphi_\theta(\cdot)$
  - with weights & biases $\theta = (\beta^{(k)}, w^{(k)})_{k=1,\dots,\ell}$
  - activation functions $\psi^{(i)}$: sigmoid, tanh, ReLU, . . . ; applied coordinate-wise

$$\underbrace{\varphi_\theta(x)}_{\varphi(\theta, x)} = \psi^{(\ell)}\Big(\beta^{(\ell)} + w^{(\ell)} \dots \psi^{(2)}\Big(\beta^{(2)} + w^{(2)} \underbrace{\psi^{(1)}(\beta^{(1)} + w^{(1)} x)}_{\text{one hidden layer}}\Big) \dots\Big)$$

  - Depth = number of layers; width of a layer = dimension of bias vector

# Ingredient 1: Neural Networks – Comments

- Many other architectures (convolutional neural networks, recurrent neural networks, . . . ), see e.g. [Leijnen and Veen, 2020]
- Successes of deep learning in many fields: natural language processing, computer vision, drug design, . . . and even games!
- Combination with reinforcement learning (see lecture 6)
- Universal approximation theorems [Cybenko, 1989], [Hornik, 1991], . . .
- Connections with numerical analysis, see e.g. [Després, 2022]

# Ingredient 1: Neural Networks – Gradients

**Differentiation:** can compute partial derivatives by automatic differentiation (AD) at every $(\theta, x)$:

- With respect to parameters: $\nabla_\theta \varphi(\theta, x)$

$$\nabla_{\beta^{(\ell)}} \varphi(\theta, x) = \ldots, \qquad \nabla_{w^{(2)}} \varphi(\theta, x) = \ldots$$

  $\Rightarrow$ can perform gradient descent on these parameters

# Ingredient 1: Neural Networks – Gradients

**Differentiation:** can compute partial derivatives by automatic differentiation (AD) at every $(\theta, x)$:

- With respect to parameters: $\nabla_\theta \varphi(\theta, x)$

$$\nabla_{\beta^{(\ell)}} \varphi(\theta, x) = \dots, \qquad \nabla_{w^{(2)}} \varphi(\theta, x) = \dots$$

$\Rightarrow$ can perform gradient descent on these parameters

- With respect to state variable: $\nabla_x \varphi(\theta, x)$ can be computed by AD too

$$\partial_{x_1} \varphi(\theta, x) = \dots$$

$\Rightarrow$ can be used in PDEs (see lecture 5)

# Ingredient 2: Stochastic Gradient Descent

- **Goal:** Minimize over $\varphi(\cdot)$, $\mathbb{J}(\varphi) := \mathbb{E}_\xi[\mathbb{L}(\varphi, \xi)]$
- **Parameterization:** $\widetilde{\mathbb{J}}(\theta) := \mathbb{E}_\xi[\widetilde{\mathbb{L}}(\theta, \xi)]$, where $\widetilde{\mathbb{L}}(\theta, \xi) := \mathbb{L}(\varphi_\theta, \xi)$

# Ingredient 2: Stochastic Gradient Descent

- **Goal:** Minimize over $\varphi(\cdot)$, $\mathbb{J}(\varphi) := \mathbb{E}_\xi[\mathbb{L}(\varphi, \xi)]$
- **Parameterization:** $\widetilde{\mathbb{J}}(\theta) := \mathbb{E}_\xi[\widetilde{\mathbb{L}}(\theta, \xi)]$, where $\widetilde{\mathbb{L}}(\theta, \xi) := \mathbb{L}(\varphi_\theta, \xi)$
- **Setting:** the distribution of $\xi$ is unknown so we cannot compute $\mathbb{E}_\xi$, but
  - we have some samples (i.e. random realizations) of $\xi$
  - we know $\mathbb{L}$

# Ingredient 2: Stochastic Gradient Descent

- **Goal:** Minimize over $\varphi(\cdot)$, $\mathbb{J}(\varphi) := \mathbb{E}_\xi[\mathbb{L}(\varphi, \xi)]$
- **Parameterization:** $\widetilde{\mathbb{J}}(\theta) := \mathbb{E}_\xi[\widetilde{\mathbb{L}}(\theta, \xi)]$, where $\widetilde{\mathbb{L}}(\theta, \xi) := \mathbb{L}(\varphi_\theta, \xi)$
- **Setting:** the distribution of $\xi$ is unknown so we cannot compute $\mathbb{E}_\xi$, but
  - we have some samples (i.e. random realizations) of $\xi$
  - we know $\mathbb{L}$
- Example: Regression: $\xi = (x, f(x))$, $\widetilde{\mathbb{J}}(\theta) := \mathbb{E}_\xi[\,\|\varphi_\theta(x) - f(x)\|^2\,]$

# Ingredient 2: Stochastic Gradient Descent

- **Goal:** Minimize over $\varphi(\cdot)$, $\mathbb{J}(\varphi) := \mathbb{E}_\xi[\mathbb{L}(\varphi, \xi)]$
- **Parameterization:** $\widetilde{\mathbb{J}}(\theta) := \mathbb{E}_\xi[\widetilde{\mathbb{L}}(\theta, \xi)]$, where $\widetilde{\mathbb{L}}(\theta, \xi) := \mathbb{L}(\varphi_\theta, \xi)$
- **Setting:** the distribution of $\xi$ is unknown so we cannot compute $\mathbb{E}_\xi$, but
  - we have some samples (i.e. random realizations) of $\xi$
  - we know $\mathbb{L}$
- Example: Regression: $\xi = (x, f(x))$, $\widetilde{\mathbb{J}}(\theta) := \mathbb{E}_\xi[\, \|\varphi_\theta(x) - f(x)\|^2 \,]$

**Algorithm:** Stochastic Gradient Descent

**Input:** Initial param. $\theta_0$; data $S = (\xi_s)_{s=1,\dots,|S|}$; nb of steps $\mathtt{K}$; learning rates $(\eta^{(\mathtt{k})})_{\mathtt{k}}$

**Output:** Parameter $\theta^\star$ s.t. $\varphi_{\theta^\star}$ (approximately) minimizes $\widetilde{\mathbb{J}}$

Initialize $\theta^{(0)} = \theta_0$

**for** $\mathtt{k} = 0, 1, 2, \dots, \mathtt{K} - 1$ **do**

  Pick $s \in S$ randomly

  Compute the gradient $\nabla_\theta \widetilde{\mathbb{L}}(\theta^{(\mathtt{k}-1)}, \xi_s) = \frac{d}{d\theta} \mathbb{L}(\varphi_{\theta^{(\mathtt{k}-1)}}, \xi_s)$

  Set $\theta^{(\mathtt{k})} = \theta^{(\mathtt{k}-1)} - \eta^{(\mathtt{k})} \nabla_\theta \widetilde{\mathbb{L}}(\theta^{(\mathtt{k}-1)}, \xi_s)$

**return** $\theta^{(\mathtt{K})}$

# Ingredient 2: Stochastic Gradient Descent – Comments

- Many variants:
  - Learning rate: `ADAM` (Adaptive Moment Estimation) [Kingma and Ba, 2014], ...
  - Samples: Mini-batches, ...

- Proofs of convergence e.g. using stochastic approximation [Robbins and Monro, 1951], [Borkar, 2009]

- In practice: many details to be discussed, see e.g.[Bottou, 2012]; choice of hyperparameters
  - architecture
  - initialization
  - learning rate
  - loss function
  - ...

# Analysis: Error Types

- Consider the task: minimize over $\varphi$ the population risk:

$$\mathcal{R}(\varphi) = \mathbb{E}_{x,y}[L(\varphi(x), y)]$$

with $x \sim \mu$ and $y = f(x) + \epsilon$ for some noise $\epsilon$ where $f$ is unknown

## Analysis: Error Types

- Consider the task: minimize over $\varphi$ the population risk:

$$\mathcal{R}(\varphi) = \mathbb{E}_{x,y}[L(\varphi(x), y)]$$

  with $x \sim \mu$ and $y = f(x) + \epsilon$ for some noise $\epsilon$ where $f$ is unknown

- In practice:
  - minimize over a hypothesis class $\mathcal{F}$ of $\varphi$
  - finite number of samples, $S = (x_m, y_m)_{m=1,\dots,M}$: empirical risk:

$$\hat{\mathcal{R}}_S(\varphi) = \frac{1}{M} \sum_{m=1}^{M} L(\varphi(x_m), y_m) \qquad (+\text{ regu})$$

  - finite number of optimization steps, say `k`

## Analysis: Error Types

We are interested in:

- Approximation error: Letting $\varphi^* = \operatorname{argmin}_{\varphi \in \mathcal{F}} \operatorname{dist}(\varphi, f)$,

$$\epsilon_{\text{approx}} = \operatorname{dist}(\varphi^*, f)$$

- Estimation error: Letting $\hat{\varphi}_S = \operatorname{argmin}_{\varphi \in \mathcal{F}} \hat{\mathcal{R}}_S(\varphi)$

$$\epsilon_{\text{estim}} = \operatorname{dist}(\hat{\varphi}_S, \varphi^*)$$

- Optimization error: After `k` steps, we get $\varphi_S^{(\mathtt{k})}$;

$$\epsilon_{\text{optim}} = \operatorname{dist}(\varphi_S^{(\mathtt{k})}, \hat{\varphi}_S)$$

# Analysis: Error Types

We are interested in:

- Approximation error: Letting $\varphi^* = \operatorname{argmin}_{\varphi \in \mathcal{F}} \operatorname{dist}(\varphi, f)$,

$$\epsilon_{\text{approx}} = \operatorname{dist}(\varphi^*, f)$$

- Estimation error: Letting $\hat{\varphi}_S = \operatorname{argmin}_{\varphi \in \mathcal{F}} \hat{\mathcal{R}}_S(\varphi)$

$$\epsilon_{\text{estim}} = \operatorname{dist}(\hat{\varphi}_S, \varphi^*)$$

- Optimization error: After $\mathtt{k}$ steps, we get $\varphi_S^{(\mathtt{k})}$;

$$\epsilon_{\text{optim}} = \operatorname{dist}(\varphi_S^{(\mathtt{k})}, \hat{\varphi}_S)$$

- Generalization error of the learnt $\varphi_S^{(\mathtt{k})}$:

$$\epsilon_{\text{gene}} = \epsilon_{\text{approx}} + \epsilon_{\text{estim}} + \epsilon_{\text{optim}}$$

# Outline

# Outline

# From optimal control to optimization

- An optimal control is a "temporally extended" optimization problem
- Numerically, we cannot minimize over all possible controls
- We can parameterize the control function
- and then optimize over the parameters
- See e.g. [Gobet and Munos, 2005], [Han and E, 2016], ...

**Stochastic optimal control problem:**

Minimize over $\alpha(\cdot,\cdot)$

$$J(\alpha(\cdot,\cdot)) = \mathbb{E}\Big[\int_0^T f(X_t, \alpha(t, X_t))\, dt + g(X_T)\Big],$$

with

$$X_0 \sim m_0\,, \quad dX_t = b(X_t, \alpha(t, X_t))\, dt + \sigma dW_t$$

# Stochastic Optimal Control: Approximate Problem

**Stochastic optimal control problem:** (1) neural network $\varphi_\theta$,

Minimize over **neural network** parameters $\theta$

$$J(\theta) = \mathbb{E}\Big[ \int_0^T f\left(X_t, \varphi_\theta(t, X_t)\right)\, dt + g\left(X_T\right) \Big],$$

with

$$X_0 \sim m_0\,, \quad dX_t = b(X_t, \varphi_\theta(t, X_t))\, dt + \sigma dW_t$$

**Stochastic optimal control problem:** (1) neural network $\varphi_\theta$, (2) time discretization

Minimize over **neural network** parameters $\theta$ and $N_T$ time steps

$$J^{N_T}(\theta) = \mathbb{E}\Big[ \sum_{n=0}^{N_T-1} f\left(X_n, \varphi_\theta(t_n, X_n)\right) \Delta t + g\left(X_{N_T}\right) \Big],$$

with

$$X_0 \sim m_0 , \quad X_{n+1} - X_n = b(X_n, \varphi_\theta(t_n, X_n))\Delta t + \sigma \Delta W_n$$

# Stochastic Optimal Control: Approximate Problem

**Stochastic optimal control problem:** (1) neural network $\varphi_\theta$, (2) time discretization

Minimize over **neural network** parameters $\theta$ and $N_T$ time steps

$$J^{N_T}(\theta) = \mathbb{E}\Big[\sum_{n=0}^{N_T-1} f\left(X_n, \varphi_\theta(t_n, X_n)\right)\Delta t + g\left(X_{N_T}\right)\Big],$$

with

$$X_0 \sim m_0\,, \quad X_{n+1} - X_n = b(X_n, \varphi_\theta(t_n, X_n))\Delta t + \sigma \Delta W_n$$

$\rightarrow$ neural network induces an approximation error

$\rightarrow$ time discretization induce extra errors

**Stochastic optimal control problem:** (1) neural network $\varphi_\theta$, (2) time discretization

Minimize over **neural network** parameters $\theta$ and $N_T$ time steps

$$J^{N_T}(\theta) = \mathbb{E}\Big[ \sum_{n=0}^{N_T-1} f\left(X_n, \varphi_\theta(t_n, X_n)\right) \Delta t + g\left(X_{N_T}\right) \Big],$$

with

$$X_0 \sim m_0\,, \quad X_{n+1} - X_n = b(X_n, \varphi_\theta(t_n, X_n))\Delta t + \sigma \Delta W_n$$

→ neural network induces an approximation error

→ time discretization induce extra errors

To implement SGD, at each iteration we pick a sample $\xi = (X_0, \Delta W_0, \dots, \Delta W_{N_T-1})$

# Outline

# MFC: Approximate Problem

**MFC problem:**

Minimize over $\alpha(\cdot, \cdot)$

$$J(\alpha(\cdot, \cdot)) = \mathbb{E}\Big[ \int_0^T f(X_t, \mu_t, \alpha(t, X_t))\, dt + g(X_T, \mu_T) \Big],$$

where $\mu_t = \mathcal{L}(X_t)$ with

$$X_0 \sim m_0 \,, \quad dX_t = b(X_t, \mu_t, \alpha(t, X_t))\, dt + \sigma dW_t$$

# MFC: Approximate Problem

**MFC problem:** (1) Finite pop.,

Minimize over **decentralized** controls $\alpha(\cdot,\cdot)$ with $N$ agents

$$J^N(\alpha(\cdot,\cdot)) = \mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\int_0^T f\left(X_t^i, \mu_t^N, \alpha(t, X_t^i)\right)\, dt + g\left(X_T^i, \mu_T^N\right)\Big],$$

where $\mu_t^N = \frac{1}{N}\sum_{j=1}^N \delta_{X_t^j}$, with

$$X_0^j \sim m_0\,, \quad dX_t^j = b(X_t^j, \mu_t^N, \alpha(t, X_t^j))\, dt + \sigma dW_t^j$$

# MFC: Approximate Problem

**MFC problem:** (1) Finite pop., (2) neural network $\varphi_\theta$,

Minimize over **neural network** parameters $\theta$ with $N$ agents

$$J^N(\theta) = \mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\int_0^T f\left(X_t^i, \mu_t^N, \varphi_\theta(t, X_t^i)\right)\, dt + g\left(X_T^i, \mu_T^N\right)\Big],$$

where $\mu_t^N = \frac{1}{N}\sum_{j=1}^N \delta_{X_t^j}$, with

$$X_0^j \sim m_0\,, \quad dX_t^j = b(X_t^j, \mu_t^N, \varphi_\theta(t, X_t^j))\, dt + \sigma dW_t^j$$

# MFC: Approximate Problem

**MFC problem:** (1) Finite pop., (2) neural network $\varphi_\theta$, (3) time discretization

Minimize over **neural network** parameters $\theta$ with $N$ agents and $N_T$ time steps

$$J^{N,N_T}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{n=0}^{N_T-1} f\left(X_n^i, \mu_n^N, \varphi_\theta(t_n, X_n^i)\right)\Delta t + g\left(X_{N_T}^i, \mu_{N_T}^N\right)\right],$$

where $\mu_n^N = \frac{1}{N}\sum_{j=1}^{N}\delta_{X_n^j}$, with

$$X_0^j \sim m_0\,, \quad X_{n+1}^j - X_n^j = b(X_n^j, \mu_n^N, \varphi_\theta(t_n, X_n^j))\Delta t + \sigma \Delta W_n^j$$

# MFC: Approximate Problem

**MFC problem:** (1) Finite pop., (2) neural network $\varphi_\theta$, (3) time discretization

Minimize over **neural network** parameters $\theta$ with $N$ agents and $N_T$ time steps

$$J^{N,N_T}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{n=0}^{N_T-1} f\left(X_n^i, \mu_n^N, \varphi_\theta(t_n, X_n^i)\right)\Delta t + g\left(X_{N_T}^i, \mu_{N_T}^N\right)\right],$$

where $\mu_n^N = \frac{1}{N}\sum_{j=1}^N \delta_{X_n^j}$, with

$$X_0^j \sim m_0\,, \quad X_{n+1}^j - X_n^j = b(X_n^j, \mu_n^N, \varphi_\theta(t_n, X_n^j))\Delta t + \sigma \Delta W_n^j$$

$\rightarrow$ neural network induces an approximation error

$\rightarrow$ finite population and time discretization induce extra errors

# MFC: Approximate Problem

**MFC problem:** (1) Finite pop., (2) neural network $\varphi_\theta$, (3) time discretization

Minimize over **neural network** parameters $\theta$ with $N$ agents and $N_T$ time steps

$$J^{N,N_T}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{n=0}^{N_T-1} f\left(X_n^i, \mu_n^N, \varphi_\theta(t_n, X_n^i)\right)\Delta t + g\left(X_{N_T}^i, \mu_{N_T}^N\right)\right],$$

where $\mu_n^N = \frac{1}{N}\sum_{j=1}^{N}\delta_{X_n^j}$, with

$$X_0^j \sim m_0\,, \quad X_{n+1}^j - X_n^j = b(X_n^j, \mu_n^N, \varphi_\theta(t_n, X_n^j))\Delta t + \sigma\Delta W_n^j$$

$\rightarrow$ neural network induces an approximation error

$\rightarrow$ finite population and time discretization induce extra errors

Note: we aim for a decentralized control, whereas for a general $N$-agent control problem, the optimal control is not always of this type

# Convergence Analysis

- The following kind of convergence result (bound on the **approximation error**) can be proved, see [Carmona and Laurière, 2022]:

**Approximation theorem**

Under suitable assumptions (in particular regularity of the value function),

$$\left| \inf_{\alpha(\cdot,\cdot)} J(\alpha(\cdot,\cdot)) - \inf_{\theta \in \Theta} J^{N,N_T}(\theta) \right| \leq \epsilon_1(N) + \epsilon_2(\dim(\theta)) + \epsilon_3(N_T)$$

# Convergence Analysis

- The following kind of convergence result (bound on the **approximation error**) can be proved, see [Carmona and Laurière, 2022]:

**Approximation theorem**

Under suitable assumptions (in particular regularity of the value function),

$$\left| \inf_{\alpha(\cdot,\cdot)} J(\alpha(\cdot,\cdot)) - \inf_{\theta\in\Theta} J^{N,N_T}(\theta) \right| \leq \epsilon_1(N) + \epsilon_2(\dim(\theta)) + \epsilon_3(N_T)$$

- The **estimation error** for shallow neural networks can be analyzed using techniques similar to [Carmona and Laurière, 2021]

## Convergence Analysis

- The following kind of convergence result (bound on the **approximation error**) can be proved, see [Carmona and Laurière, 2022]:

**Approximation theorem**

Under suitable assumptions (in particular regularity of the value function),

$$\left| \inf_{\alpha(\cdot,\cdot)} J(\alpha(\cdot,\cdot)) - \inf_{\theta \in \Theta} J^{N,N_T}(\theta) \right| \leq \epsilon_1(N) + \epsilon_2(\dim(\theta)) + \epsilon_3(N_T)$$

- The **estimation error** for shallow neural networks can be analyzed using techniques similar to [Carmona and Laurière, 2021]

- The **optimization error** remains to be studied

# Convergence Analysis

- The following kind of convergence result (bound on the **approximation error**) can be proved, see [Carmona and Laurière, 2022]:

**Approximation theorem**

Under suitable assumptions (in particular regularity of the value function),

$$\left| \inf_{\alpha(\cdot,\cdot)} J(\alpha(\cdot,\cdot)) - \inf_{\theta \in \Theta} J^{N,N_T}(\theta) \right| \leq \epsilon_1(N) + \epsilon_2(\dim(\theta)) + \epsilon_3(N_T)$$

- The **estimation error** for shallow neural networks can be analyzed using techniques similar to [Carmona and Laurière, 2021]
- The **optimization error** remains to be studied
- Many extensions and refinements to be investigated

# Approximation Error Analysis: Main Ingredients of the Proof

**Proposition 1** ($N$ agents & decentralized controls):

Under suitable assumptions, there exists a decentralized control $\alpha^*$ s.t. *($d$ = dimension of $X_t$)*

$$\left| \inf_{\alpha(\cdot)} J(\alpha(\cdot)) - J^N(\alpha^*(\cdot)) \right| \leq \epsilon_1(N) \in \widetilde{O}\left(N^{-1/d}\right).$$

**Proof: propagation of chaos** type argument [Carmona and Delarue, 2018]

# Approximation Error Analysis: Main Ingredients of the Proof

**Proposition 1** ($N$ agents & decentralized controls):

Under suitable assumptions, there exists a decentralized control $\alpha^*$ s.t. *($d$ = dimension of $X_t$)*

$$\left| \inf_{\alpha(\cdot)} J(\alpha(\cdot)) - J^N(\alpha^*(\cdot)) \right| \leq \epsilon_1(N) \in \widetilde{O}\left(N^{-1/d}\right).$$

**Proof: propagation of chaos** type argument [Carmona and Delarue, 2018]

**Proposition 2** (approximation by neural networks): Under suitable assumptions

There exists a set of parameters $\theta \in \Theta$ for a one-hidden layer $\hat{\varphi}_\theta$ s.t.

$$\left| J^N(\alpha^*(\cdot)) - J^N(\hat{\varphi}_\theta(\cdot)) \right| \leq \epsilon_2(\dim(\theta)) \in O\left(\dim(\theta)^{-\frac{1}{3(d+1)}}\right).$$

**Proof: Key difficulty:** approximate $v^*(\cdot)$ by $\hat{\varphi}_\theta(\cdot)$ while controlling $\|\nabla \hat{\varphi}_\theta(\cdot)\|$ by $\|\nabla v^*(\cdot)\|$

$\rightarrow$ universal approximation without rate of convergence is not enough

$\rightarrow$ approximation rate for the derivative too, e.g. from [Mhaskar and Micchelli, 1995]

# Approximation Error Analysis: Main Ingredients of the Proof

**Proposition 1** ($N$ agents & decentralized controls):

Under suitable assumptions, there exists a decentralized control $\alpha^*$ s.t. *($d$ = dimension of $X_t$)*

$$\left| \inf_{\alpha(\cdot)} J(\alpha(\cdot)) - J^N(\alpha^*(\cdot)) \right| \leq \epsilon_1(N) \in \widetilde{O}\left(N^{-1/d}\right).$$

**Proof: propagation of chaos** type argument [Carmona and Delarue, 2018]

**Proposition 2** (approximation by neural networks): Under suitable assumptions

There exists a set of parameters $\theta \in \Theta$ for a one-hidden layer $\hat{\varphi}_\theta$ s.t.

$$\left| J^N(\alpha^*(\cdot)) - J^N(\hat{\varphi}_\theta(\cdot)) \right| \leq \epsilon_2(\dim(\theta)) \in O\left(\dim(\theta)^{-\frac{1}{3(d+1)}}\right).$$

**Proof: Key difficulty:** approximate $v^*(\cdot)$ by $\hat{\varphi}_\theta(\cdot)$ while controlling $\|\nabla \hat{\varphi}_\theta(\cdot)\|$ by $\|\nabla v^*(\cdot)\|$

$\rightarrow$ universal approximation without rate of convergence is not enough

$\rightarrow$ approximation rate for the derivative too, e.g. from [Mhaskar and Micchelli, 1995]

**Proposition 3** (Euler-Maruyama scheme):

For a specific neural network $\hat{\varphi}_\theta(\cdot)$,

$$\left| J^N(\hat{\varphi}_\theta(\cdot)) - J^{N,N_T}(\hat{\varphi}_\theta(\cdot)) \right| \leq \epsilon_3(N_T) \in O\left(N_T^{-1/2}\right).$$

**Key point:** $O(\cdot)$ independent of $N$ and $\dim(\theta)$

**Proof:** analysis of **strong error rate** for Euler scheme (reminiscent of [Bossy and Talay, 1997])

- Key idea: replace optimal control problem by (finite dim.) optimization problem:
  - Loss function = cost: $J^{N,N_T}(\theta) = \mathbb{E}[\mathbb{L}(\varphi_\theta, \xi)]$
  - One sample: $\xi = \left(X_0^j, (\Delta W_n^j)_{n=0,\dots,N_T-1}\right)_{j=1,\dots,N}$

$\rightarrow$ can use **Stochastic Gradient Descent**

# Implementation

- Key idea: replace optimal control problem by (finite dim.) optimization problem:
  - Loss function = cost: $J^{N,N_T}(\theta) = \mathbb{E}[\mathbb{L}(\varphi_\theta, \xi)]$
  - One sample: $\xi = \left(X_0^j, (\Delta W_n^j)_{n=0,\dots,N_T-1}\right)_{j=1,\dots,N}$

  $\rightarrow$ can use **Stochastic Gradient Descent**

- Structure:

- Key idea: replace optimal control problem by (finite dim.) optimization problem:
  - Loss function = cost: $J^{N,N_T}(\theta) = \mathbb{E}[\mathbb{L}(\varphi_\theta, \xi)]$
  - One sample: $\xi = \left(X_0^j, (\Delta W_n^j)_{n=0,\dots,N_T-1}\right)_{j=1,\dots,N}$

$\rightarrow$ can use **Stochastic Gradient Descent**

- Structure:

- Key idea: replace optimal control problem by (finite dim.) optimization problem:
  - Loss function = cost: $J^{N,N_T}(\theta) = \mathbb{E}[\mathbb{L}(\varphi_\theta, \xi)]$
  - One sample: $\xi = \left(X_0^j, (\Delta W_n^j)_{n=0,\dots,N_T-1}\right)_{j=1,\dots,N}$

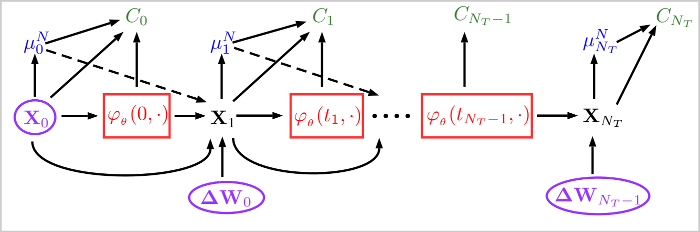
$\rightarrow$ can use **Stochastic Gradient Descent**

- Structure:

- Key idea: replace optimal control problem by (finite dim.) optimization problem:
  - Loss function = cost: $J^{N,N_T}(\theta) = \mathbb{E}[\mathbb{L}(\varphi_\theta, \xi)]$
  - One sample: $\xi = \left(X_0^j, (\Delta W_n^j)_{n=0,\dots,N_T-1}\right)_{j=1,\dots,N}$

$\rightarrow$ can use **Stochastic Gradient Descent**

- Structure:

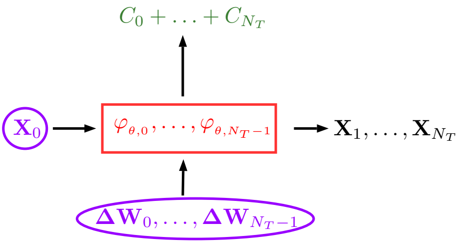


# Implementation

- Key idea: replace optimal control problem by (finite dim.) optimization problem:
  - Loss function = cost: $J^{N,N_T}(\theta) = \mathbb{E}[\mathbb{L}(\varphi_\theta, \xi)]$
  - One sample: $\xi = \left(X_0^j, (\Delta W_n^j)_{n=0,\dots,N_T-1}\right)_{j=1,\dots,N}$

$\rightarrow$ can use **Stochastic Gradient Descent**

- Structure:



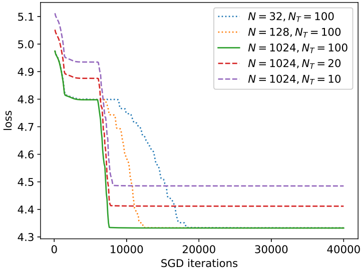
# Numerical Illustration 1: LQ MFC

**Benchmark** to assess **empirical convergence of SGD:** LQ problem with explicit sol.

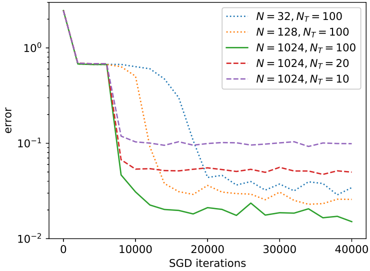
**Example:** Linear dynamics, quadratic costs of the type

$$f(x,\mu,v) = \underbrace{(\bar{\mu} - x)^2}_{\text{distance to mean position}} + \underbrace{v^2}_{\text{cost of moving}}, \qquad \bar{\mu} = \underbrace{\int \mu(\xi)d\xi}_{\text{mean position}}, \qquad g(x) = x^2$$

Numerical example with $d = 10$ (see [Carmona and Laurière, 2022]):




total cost (= loss function)




$L^2$-error on the control

# Numerical Illustration 2: min-LQ MFC with common noise

The following model is inspired by [Salhab et al., 2015] and [Achdou and Lasry, 2019].

**MFC with simple CN:**

Dynamics: $dX_t = \phi_t(X_t, \epsilon^0_t)dt + \sigma dW_t$, $\epsilon^0_t = 0$ until $t = T/2$, and then $\xi_1$ or $\xi_2$ w.p. $1/2$

Running cost $|\phi_t(X_t, \epsilon^0_t)|^2$, final cost $(X_T - \epsilon^0_T)^2 + \bar{Q}_T(\bar{m}_T - X_T)^2$

Parameter values: $\sigma = 0.1$, $T = 1$, $\xi_1 = -1.5$, $\xi_2 = +1.5$

Numerical results:

- neural network $\varphi_\theta(t, X_t, \epsilon^0_t)$, taking as an input the common noise
- benchmark solution computed by solving a **system of 6 PDEs** (see [Achdou and Lasry, 2019])

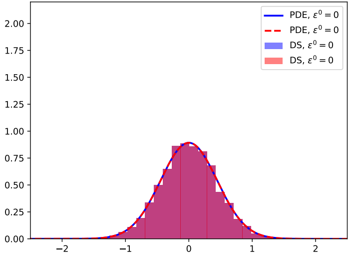
# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.



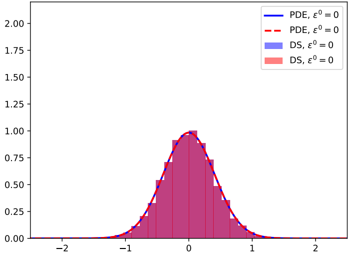

$t = 0$

Until $T/2$: concentrate around mid-point $= 0$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.



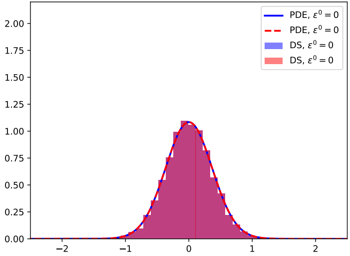

$t = 0.1$

Until $T/2$: concentrate around mid-point $= 0$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.



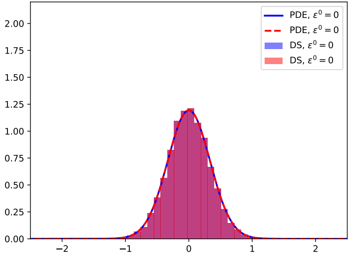

$t = 0.2$

Until $T/2$: concentrate around mid-point $= 0$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.



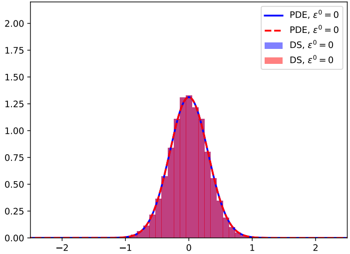

$t = 0.3$

Until $T/2$: concentrate around mid-point $= 0$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.



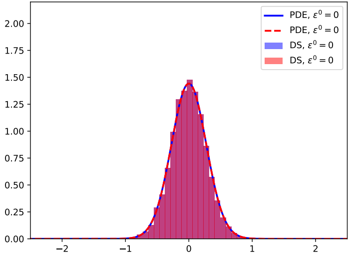

$$t = 0.4$$

Until $T/2$: concentrate around mid-point $= 0$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.



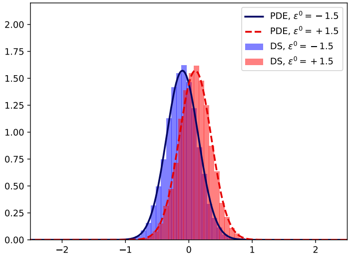

$t = 0.5$

Until $T/2$: concentrate around mid-point $= 0$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.




$t = 0.6$

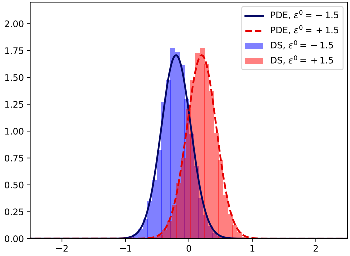
Until $T/2$: concentrate around mid-point $= 0$

After $T/2$: move towards the target selected by common noise

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.




$t = 0.7$

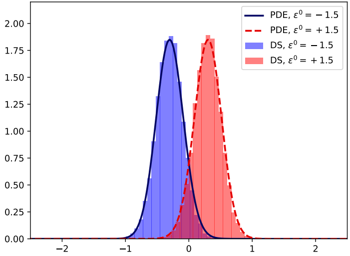
Until $T/2$: concentrate around mid-point $= 0$

After $T/2$: move towards the target selected by common noise

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.




$t = 0.8$

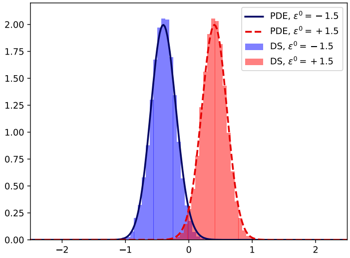
Until $T/2$: concentrate around mid-point $= 0$

After $T/2$: move towards the target selected by common noise

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.




$t = 0.9$

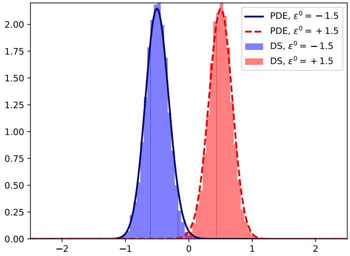
Until $T/2$: concentrate around mid-point $= 0$

After $T/2$: move towards the target selected by common noise

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: min-LQ MFC with common noise

Here the common noise takes one among two values, at time $T/2$.




$t = 1$

Until $T/2$: concentrate around mid-point $= 0$

After $T/2$: move towards the target selected by common noise

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 3: MFC with Interactions Through the Controls

**Price Impact Model** [Carmona and Lacker, 2015, Carmona and Delarue, 2018]:

- Price process: with $\nu^{\alpha}$ = population's distribution over actions,

$$dS_t^{\alpha} = \gamma \int_{\mathbb{R}} a d\nu_t^{\alpha}(a) dt + \sigma_0 dW_t^0$$

- Typical agent's inventory: $dX_t^{\alpha} = \alpha_t dt + \sigma dW_t$
- Typical agent's wealth: $dK_t^{\alpha} = -\big(\alpha_t S_t^{\alpha} + c_{\alpha}(\alpha_t)\big) dt$
- Typical agent's portfolio value: $V_t^{\alpha} = K_t^{\alpha} + X_t^{\alpha} S_t^{\alpha}$

# Numerical Illustration 3: MFC with Interactions Through the Controls

**Price Impact Model** [Carmona and Lacker, 2015, Carmona and Delarue, 2018]:

- Price process: with $\nu^\alpha$ = population's distribution over actions,

$$dS_t^\alpha = \gamma \int_{\mathbb{R}} a d\nu_t^\alpha(a) dt + \sigma_0 dW_t^0$$

- Typical agent's inventory: $dX_t^\alpha = \alpha_t dt + \sigma dW_t$
- Typical agent's wealth: $dK_t^\alpha = -\big(\alpha_t S_t^\alpha + c_\alpha(\alpha_t)\big)dt$
- Typical agent's portfolio value: $V_t^\alpha = K_t^\alpha + X_t^\alpha S_t^\alpha$

**Objective:** minimize

$$J(\alpha) = \mathbb{E}\Big[\int_0^T c_X(X_t^\alpha)dt + g(X_T^\alpha) - V_T^\alpha\Big]$$

# Numerical Illustration 3: MFC with Interactions Through the Controls

**Price Impact Model** [Carmona and Lacker, 2015, Carmona and Delarue, 2018]:

- Price process: with $\nu^\alpha$ = population's distribution over actions,

$$dS_t^\alpha = \gamma \int_{\mathbb{R}} a d\nu_t^\alpha(a) dt + \sigma_0 dW_t^0$$

- Typical agent's inventory: $dX_t^\alpha = \alpha_t dt + \sigma dW_t$
- Typical agent's wealth: $dK_t^\alpha = -\big(\alpha_t S_t^\alpha + c_\alpha(\alpha_t)\big) dt$
- Typical agent's portfolio value: $V_t^\alpha = K_t^\alpha + X_t^\alpha S_t^\alpha$

**Objective:** minimize

$$J(\alpha) = \mathbb{E}\Big[ \int_0^T c_X(X_t^\alpha) dt + g(X_T^\alpha) - V_T^\alpha \Big]$$

Equivalent problem:

$$J(\alpha) = \mathbb{E}\Big[ \int_0^T \left( c_\alpha(\alpha_t) + c_X(X_t^\alpha) - \gamma X_t^\alpha \int_{\mathbb{R}} a d\nu_t^\alpha(a) \right) dt + g(X_T^\alpha) \Big]$$

# Numerical Illustration 3: MFC with Interactions Through the Controls

**Price Impact Model** [Carmona and Lacker, 2015, Carmona and Delarue, 2018]:

- Price process: with $\nu^\alpha$ = population's distribution over actions,

$$dS_t^\alpha = \gamma \int_{\mathbb{R}} a d\nu_t^\alpha(a) dt + \sigma_0 dW_t^0$$

- Typical agent's inventory: $dX_t^\alpha = \alpha_t dt + \sigma dW_t$
- Typical agent's wealth: $dK_t^\alpha = -\big(\alpha_t S_t^\alpha + c_\alpha(\alpha_t)\big) dt$
- Typical agent's portfolio value: $V_t^\alpha = K_t^\alpha + X_t^\alpha S_t^\alpha$

**Objective:** minimize

$$J(\alpha) = \mathbb{E}\Big[ \int_0^T c_X(X_t^\alpha) dt + g(X_T^\alpha) - V_T^\alpha \Big]$$

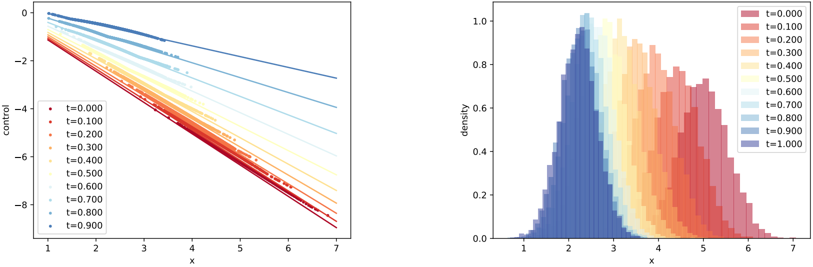
Equivalent problem:

$$J(\alpha) = \mathbb{E}\Big[ \int_0^T \left( c_\alpha(\alpha_t) + c_X(X_t^\alpha) - \gamma X_t^\alpha \int_{\mathbb{R}} a d\nu_t^\alpha(a) \right) dt + g(X_T^\alpha) \Big]$$

We take: $c_\alpha(v) = \frac{1}{2} c_\alpha v^2$, $c_X(x) = \frac{1}{2} c_X x^2$ and $g(x) = \frac{1}{2} c_g x^2$

# Numerical Illustration 3: MFC with Interactions Through the Controls

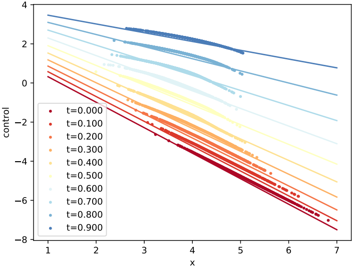
Control learnt (left) and associated state distribution (right)

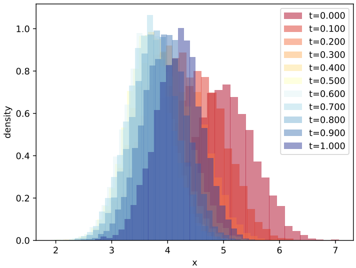



$$T = 1,\ c_X = 2,\ c_\alpha = 1,\ c_g = 0.3,\ \sigma = 0.5,\ \gamma = 0.2$$

See Section 2 in [Carmona and Laurière, 2023] for more details.

Control learnt (left) and associated state distribution (right)




$$T = 1,\ c_X = 2,\ c_\alpha = 1,\ c_g = 0.3,\ \sigma = 0.5,\ \gamma = 1$$

See Section 2 in [Carmona and Laurière, 2023] for more details.

# Sample code

## Code

Sample code to illustrate: IPython notebook

```
https://colab.research.google.com/drive/1QYWz4Sclw9goRZsbd0uB6wR6a0Uu0a3k?usp=sharing
```

- Deep learning for MFC using a direct approach where the control is parameterized as a neural network
- Applied to the price impact model discussed above

# Related works

- DL for stochastic control [Gobet and Munos, 2005], [Han and E, 2016], . . .
- Various possible implementations; example: 1 NN per time step instead of a single 1 NN as a function of time
- Extensions to finite-player games [Hu, 2021]
- Extension to MFC presented here [Carmona and Laurière, 2022]; see also [Carmona and Laurière, 2023]
- Related works with mean field: [Fouque and Zhang, 2020] (MFC with delay), [Germain et al., 2019], [Agram et al., 2020], [Dayanikli et al., 2023] (with population-dependent controls), . . .

# Outline

# Shooting Method for FBSDE

- Goal: solve an FBSDE system
- The backward process has a value $Y_0$ at time $0$, but it is not known
- Try to guess the correct initial condition so that the terminal condition is satisfied
- This yields a new optimal control problem
- See e.g. [Kohlmann and Zhou, 2000], [Sannikov, 2008], ...
- For the new optimal control problem, use deep learning [E et al., 2017]

## DeepBSDE (E et al.)

Solutions of sto. control problems can be characterized by **FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0 & \to \text{state} \\ dY_t = -F(t, X_t, Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T) & \to \text{control/cost} \end{cases}$$

(stemming from sto. Pontryagin's or Bellman's principle: $F = f$ or $F = \partial_x H$)

## DeepBSDE (E et al.)

Solutions of sto. control problems can be characterized by **FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0 & \rightarrow \text{state} \\ dY_t = -F(t, X_t, Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T) & \rightarrow \text{control/cost} \end{cases}$$

(stemming from sto. Pontryagin's or Bellman's principle: $F = f$ or $F = \partial_x H$)

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\rightarrow$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

# DeepBSDE (E et al.)

Solutions of sto. control problems can be characterized by **FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0 & \rightarrow \text{state} \\ dY_t = -F(t, X_t, Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T) & \rightarrow \text{control/cost} \end{cases}$$

(stemming from sto. Pontryagin's or Bellman's principle: $F = f$ or $F = \partial_x H$)

**Shooting:** Guess $Y_0$ and $(Z_t)_t$

$\rightarrow$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

**Reformulation** as a new **optimal control problem**

**Minimize** over $y_0(\cdot)$ and $\mathbf{z}(\cdot) = (z_t(\cdot))_{t \geq 0}$

$$\mathfrak{J}(y_0(\cdot), \mathbf{z}(\cdot)) = \mathbb{E}\left[ \|Y_T^{y_0,\mathbf{z}} - G(X_T^{y_0,\mathbf{z}})\|^2 \right],$$

under the constraint that $(X^{y_0,\mathbf{z}}, Y^{y_0,\mathbf{z}})$ solve: $\forall t \in [0, T]$

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0, \\ dY_t = -F(t, X_t, Y_t)dt + z(t, X_t) \cdot dW_t, & Y_0 = y_0(X_0). \end{cases}$$

# DeepBSDE (E et al.)

Solutions of sto. control problems can be characterized by **FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0 & \to \text{state} \\ dY_t = -F(t, X_t, Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T) & \to \text{control/cost} \end{cases}$$

(stemming from sto. Pontryagin's or Bellman's principle: $F = f$ or $F = \partial_x H$)

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\to$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

**Reformulation** as a new **optimal control problem**

**Minimize** over $y_0(\cdot)$ and $\mathbf{z}(\cdot) = (z_t(\cdot))_{t\geq 0}$

$$\mathfrak{J}(y_0(\cdot), \mathbf{z}(\cdot)) = \mathbb{E}\left[ \|Y_T^{y_0,\mathbf{z}} - G(X_T^{y_0,\mathbf{z}})\|^2 \right],$$

under the constraint that $(X^{y_0,\mathbf{z}}, Y^{y_0,\mathbf{z}})$ solve: $\forall t \in [0, T]$

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0, \\ dY_t = -F(t, X_t, Y_t)dt + z(t, X_t) \cdot dW_t, & Y_0 = y_0(X_0). \end{cases}$$

$\to$ New optimal control problem: apply previous method, replacing $y_0(\cdot), z(\cdot, \cdot)$ by NN

# DeepBSDE (E et al.)

Solutions of sto. control problems can be characterized by **FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0 & \to \text{state} \\ dY_t = -F(t, X_t, Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T) & \to \text{control/cost} \end{cases}$$

(stemming from sto. Pontryagin's or Bellman's principle: $F = f$ or $F = \partial_x H$)

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\to$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

**Reformulation** as a new **optimal control problem**

**Minimize** over $y_0(\cdot)$ and $\mathbf{z}(\cdot) = (z_t(\cdot))_{t \geq 0}$

$$\mathfrak{J}(y_0(\cdot), \mathbf{z}(\cdot)) = \mathbb{E}\left[ \|Y_T^{y_0,\mathbf{z}} - G(X_T^{y_0,\mathbf{z}})\|^2 \right],$$

under the constraint that $(X^{y_0,\mathbf{z}}, Y^{y_0,\mathbf{z}})$ solve: $\forall t \in [0, T]$

$$\begin{cases} dX_t = B(t, X_t, Y_t)dt + dW_t, & X_0 \sim m_0, \\ dY_t = -F(t, X_t, Y_t)dt + z(t, X_t) \cdot dW_t, & Y_0 = y_0(X_0). \end{cases}$$

$\to$ New optimal control problem: apply previous method, replacing $y_0(\cdot), z(\cdot, \cdot)$ by NN

Note: This problem is *not* the original stochastic control problem !

# Application to Solve PDEs

This method can be used to solve PDEs [E et al., 2017]

**Feynman-Kac formula**: correspondence $u(t, X_t) = Y_t$ where

# Application to Solve PDEs

This method can be used to solve PDEs [E et al., 2017]

**Feynman-Kac formula**: correspondence $u(t, X_t) = Y_t$ where

- $u$ solves the PDE

$$\begin{cases} u(T,x) = G(x) \\ \frac{\partial u}{\partial t}(t,x) + B(t,x)\frac{\partial u}{\partial x}(t,x) + \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x \partial x}(t,x) + F(t,x) = 0 \end{cases}$$

- $X$ solves the SDE:

$$dX_t = B(t,x)dt + \sigma dW_t$$

- $(Y, Z)$ solves the BSDE:

$$\begin{cases} Y_T = G(X_T) \\ dY_t = -F(t, X_t)dt + Z_t dW_t \end{cases}$$

# Application to Solve PDEs

This method can be used to solve PDEs [E et al., 2017]

**Feynman-Kac formula**: correspondence $u(t, X_t) = Y_t$ where

- $u$ solves the PDE

$$\begin{cases} u(T,x) = G(x) \\ \frac{\partial u}{\partial t}(t,x) + B(t,x)\frac{\partial u}{\partial x}(t,x) + \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x \partial x}(t,x) + F(t,x) = 0 \end{cases}$$

- $X$ solves the SDE:

$$dX_t = B(t,x)dt + \sigma dW_t$$

- $(Y, Z)$ solves the BSDE:

$$\begin{cases} Y_T = G(X_T) \\ dY_t = -F(t, X_t)dt + Z_t dW_t \end{cases}$$

- In fact $Z_t = \sigma \partial_x u(t, X_t)$

# Application to Solve PDEs

This method can be used to solve PDEs [E et al., 2017]

**Feynman-Kac formula**: correspondence $u(t, X_t) = Y_t$ where

- $u$ solves the PDE

$$\begin{cases} u(T, x) = G(x) \\ \frac{\partial u}{\partial t}(t, x) + B(t, x)\frac{\partial u}{\partial x}(t, x) + \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x \partial x}(t, x) + F(t, x) = 0 \end{cases}$$

- $X$ solves the SDE:

$$dX_t = B(t, x)dt + \sigma dW_t$$

- $(Y, Z)$ solves the BSDE:

$$\begin{cases} Y_T = G(X_T) \\ dY_t = -F(t, X_t)dt + Z_t dW_t \end{cases}$$

- In fact $Z_t = \sigma \partial_x u(t, X_t)$
- Connection also works with $dX_t = dW_t$ and a different $Y_t$ ...

# Application to Solve PDEs

This method can be used to solve PDEs [E et al., 2017]

**Feynman-Kac formula**: correspondence $u(t, X_t) = Y_t$ where

- $u$ solves the PDE

$$\begin{cases} u(T, x) = G(x) \\ \frac{\partial u}{\partial t}(t, x) + B(t, x)\frac{\partial u}{\partial x}(t, x) + \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x \partial x}(t, x) + F(t, x) = 0 \end{cases}$$

- $X$ solves the SDE:

$$dX_t = B(t, x)dt + \sigma dW_t$$

- $(Y, Z)$ solves the BSDE:

$$\begin{cases} Y_T = G(X_T) \\ dY_t = -F(t, X_t)dt + Z_t dW_t \end{cases}$$

- In fact $Z_t = \sigma \partial_x u(t, X_t)$
- Connection also works with $dX_t = dW_t$ and a different $Y_t$ ...
- Application: solve a PDE by solving the corresponding (F)BSDE

# Application to Solve PDEs

This method can be used to solve PDEs [E et al., 2017]

**Feynman-Kac formula**: correspondence $u(t, X_t) = Y_t$ where

- $u$ solves the PDE

$$\begin{cases} u(T, x) = G(x) \\ \frac{\partial u}{\partial t}(t, x) + B(t, x)\frac{\partial u}{\partial x}(t, x) + \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x \partial x}(t, x) + F(t, x) = 0 \end{cases}$$

- $X$ solves the SDE:

$$dX_t = B(t, x)dt + \sigma dW_t$$

- $(Y, Z)$ solves the BSDE:

$$\begin{cases} Y_T = G(X_T) \\ dY_t = -F(t, X_t)dt + Z_t dW_t \end{cases}$$

- In fact $Z_t = \sigma \partial_x u(t, X_t)$
- Connection also works with $dX_t = dW_t$ and a different $Y_t$ ...
- Application: solve a PDE by solving the corresponding (F)BSDE
- Ex. HJB equation. Many variations/extensions

# Deep MKV FBSDE

Solutions of MFG (and MFC) can be characterized by **MKV FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0 & \rightarrow \text{state} \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T, \mathcal{L}(X_T)) & \rightarrow \text{control/cost} \end{cases}$$

# Deep MKV FBSDE

Solutions of MFG (and MFC) can be characterized by **MKV FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0 & \rightarrow \text{state} \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T, \mathcal{L}(X_T)) & \rightarrow \text{control/cost} \end{cases}$$

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\rightarrow$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

# Deep MKV FBSDE

Solutions of MFG (and MFC) can be characterized by **MKV FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0 & \rightarrow \text{state} \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T, \mathcal{L}(X_T)) & \rightarrow \text{control/cost} \end{cases}$$

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\rightarrow$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

**Reformulation** as a **MFC problem** [Carmona and Laurière, 2022]

**Minimize** over $y_0(\cdot)$ and $\mathbf{z}(\cdot) = (z_t(\cdot))_{t\geq 0}$

$$\mathfrak{J}(y_0(\cdot), \mathbf{z}(\cdot)) = \mathbb{E}\Big[ \|Y_T^{y_0,\mathbf{z}} - G(X_T^{y_0,\mathbf{z}}, \mathcal{L}(X_T^{y_0,\mathbf{z}}))\|^2 \Big],$$

under the constraint that $(X^{y_0,\mathbf{z}}, Y^{y_0,\mathbf{z}})$ solve: $\forall t \in [0, T]$

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0, \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + z(t, X_t) \cdot dW_t, & Y_0 = y_0(X_0). \end{cases}$$

# Deep MKV FBSDE

Solutions of MFG (and MFC) can be characterized by **MKV FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0 & \to \text{state} \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T, \mathcal{L}(X_T)) & \to \text{control/cost} \end{cases}$$

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\to$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

**Reformulation** as a **MFC problem** [Carmona and Laurière, 2022]

**Minimize** over $y_0(\cdot)$ and $\mathbf{z}(\cdot) = (z_t(\cdot))_{t\geq 0}$

$$\mathfrak{J}(y_0(\cdot), \mathbf{z}(\cdot)) = \mathbb{E}\left[ \|Y_T^{y_0,\mathbf{z}} - G(X_T^{y_0,\mathbf{z}}, \mathcal{L}(X_T^{y_0,\mathbf{z}}))\|^2 \right],$$

under the constraint that $(X^{y_0,\mathbf{z}}, Y^{y_0,\mathbf{z}})$ solve: $\forall t \in [0, T]$

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0, \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + z(t, X_t) \cdot dW_t, & Y_0 = y_0(X_0). \end{cases}$$

$\to$ New MFC problem: apply previous method, replacing $y_0(\cdot), z(\cdot,\cdot)$ by NN

## Deep MKV FBSDE

Solutions of MFG (and MFC) can be characterized by **MKV FBSDEs** of the form

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0 & \rightarrow \text{state} \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + Z_t \cdot dW_t, & Y_T = G(X_T, \mathcal{L}(X_T)) & \rightarrow \text{control/cost} \end{cases}$$

**Shooting:** Guess $Y_0$ and $(Z_t)_t$
$\rightarrow$ recover sol. $(X, Y, Z)$ is found by opt. control of 2 **forward** SDEs

**Reformulation** as a **MFC problem** [Carmona and Laurière, 2022]

**Minimize** over $y_0(\cdot)$ and $\mathbf{z}(\cdot) = (z_t(\cdot))_{t \geq 0}$

$$\mathfrak{J}(y_0(\cdot), \mathbf{z}(\cdot)) = \mathbb{E}\left[ \|Y_T^{y_0,\mathbf{z}} - G(X_T^{y_0,\mathbf{z}}, \mathcal{L}(X_T^{y_0,\mathbf{z}}))\|^2 \right],$$

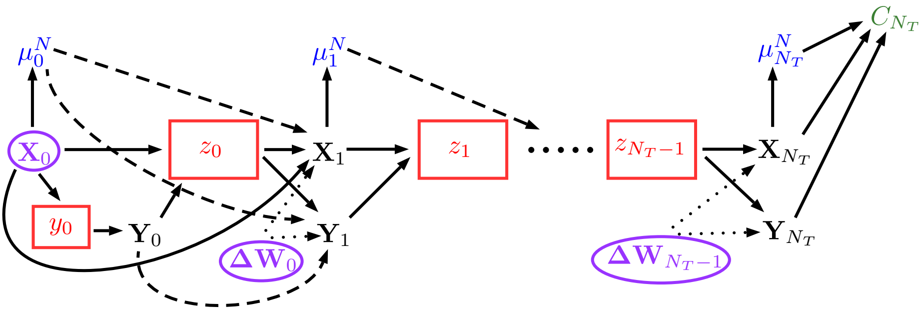
under the constraint that $(X^{y_0,\mathbf{z}}, Y^{y_0,\mathbf{z}})$ solve: $\forall t \in [0, T]$

$$\begin{cases} dX_t = B(t, X_t, \mathcal{L}(X_t), Y_t)dt + dW_t, & X_0 \sim m_0, \\ dY_t = -F(t, X_t, \mathcal{L}(X_t), Y_t)dt + z(t, X_t) \cdot dW_t, & Y_0 = y_0(X_0). \end{cases}$$

$\rightarrow$ New MFC problem: apply previous method, replacing $y_0(\cdot), z(\cdot, \cdot)$ by NN

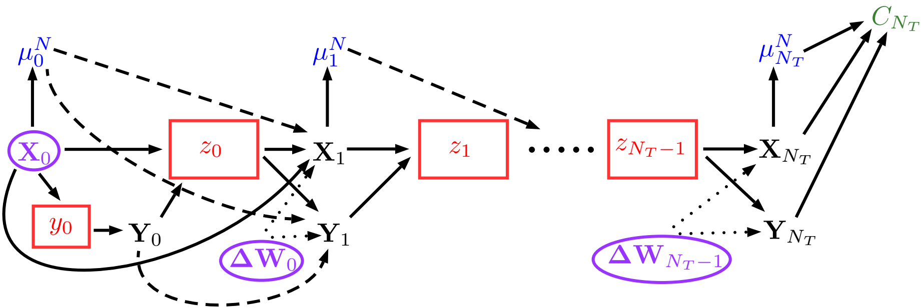
NB: This problem is *not* the original MFG or MFC

# Implementation




- **Inputs:** initial positions $\mathbf{X}_0 = (X_0^i)_i$, BM increments: $\mathbf{\Delta W}_n = (\Delta W_n^i)_i$, for all $n$
- **Loss function:** total cost = $C_{N_T}$ $=$ terminal penalty; state = $(X_n, Y_n)$
- **SGD** to optimize over the param. $\theta_y, \theta_z$ of 2 NN for $y_{\theta_y}(\cdot) \approx y_0(\cdot), z_{\theta_z}(\cdot,\cdot) \approx z(\cdot,\cdot)$

# Implementation




- **Inputs:** initial positions $\mathbf{X}_0 = (X_0^i)_i$, BM increments: $\mathbf{\Delta W}_n = (\Delta W_n^i)_i$, for all $n$
- **Loss function:** total cost = $C_{N_T}$ = terminal penalty; state = $(X_n, Y_n)$
- **SGD** to optimize over the param. $\theta_y, \theta_z$ of 2 NN for $y_{\theta_y}(\cdot) \approx y_0(\cdot), z_{\theta_z}(\cdot,\cdot) \approx z(\cdot,\cdot)$
- Alternative implementation: $1 + N_T$ NNs for $y_0(\cdot), z_0(\cdot), \ldots, z_{N_T-1}(\cdot)$

# Numerical Illustration 1: Comparison with Picard Solver

**Example of MKV FBSDE** from [Chassagneux et al., 2019] ($\boldsymbol{\rho} =$ coupling parameter)

$$dX_t = -\boldsymbol{\rho} Y_t dt + \sigma dW_t, \qquad X_0 = x_0$$

$$dY_t = \operatorname{atan}(\mathbb{E}[X_t])dt + Z_t dW_t, \qquad Y_T = G'(X_T) := \operatorname{atan}(X_T)$$

Comes from the **MFG** defined by $dX_t^\alpha = \alpha_t dt + dW_t$ and

$$J(\alpha;\mu) = \mathbb{E}\left[G(X_T^\alpha) + \int_0^T \left(\frac{1}{2\boldsymbol{\rho}}\alpha_t^2 + X_t^\alpha \operatorname{atan}\left(\int x\mu_t(dx)\right)\right)dt\right]$$

# Numerical Illustration 1: Comparison with Picard Solver

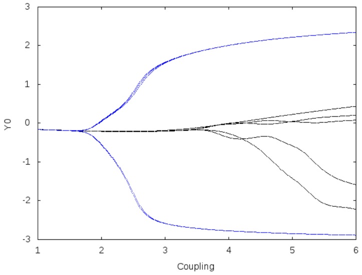
**Example of MKV FBSDE** from [Chassagneux et al., 2019] ($\rho$ = coupling parameter)

$$dX_t = -\rho Y_t dt + \sigma dW_t, \qquad X_0 = x_0$$

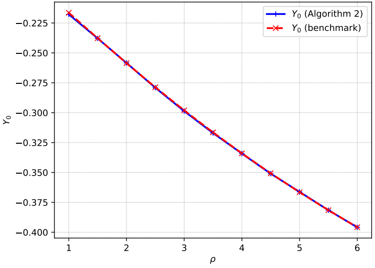
$$dY_t = \mathrm{atan}(\mathbb{E}[X_t])dt + Z_t dW_t, \qquad Y_T = G'(X_T) := \mathrm{atan}(X_T)$$

Comes from the **MFG** defined by $dX_t^\alpha = \alpha_t dt + dW_t$ and

$$J(\alpha;\mu) = \mathbb{E}\left[G(X_T^\alpha) + \int_0^T \left(\frac{1}{2\rho}\alpha_t^2 + X_t^\alpha \mathrm{atan}\left(\int x\mu_t(dx)\right)\right) dt\right]$$




[Chassagneux et al., 2019]




NN (FBSDE system)

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: LQ MFG with Common Noise

**Example: MFG for inter-bank borrowing/lending** [Carmona et al., 2015]

$X$ = log-monetary reserve, $\alpha$ = rate of borrowing/lending to central bank, cost:

$$J(\alpha; \bar{m}) = \mathbb{E}\left[\int_0^T \left[\frac{1}{2}\alpha_t^2 - q\alpha_t(\bar{m}_t - X_t) + \frac{\epsilon}{2}(\bar{m}_t - X_t)^2\right] dt + \frac{c}{2}(\bar{m}_T - X_T)^2\right]$$

where $\bar{m} = (\bar{m}_t)_{t\geq 0}$ = conditional mean of the population states given $W^0$, and

$$dX_t = [a(\bar{m}_t - X_t) + \alpha_t]dt + \sigma\left(\sqrt{1-\rho^2}dW_t + \rho\, dW_t^0\right)$$

# Numerical Illustration 2: LQ MFG with Common Noise

**Example: MFG for inter-bank borrowing/lending** [Carmona et al., 2015]

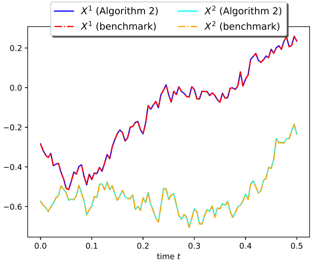
$X$ = log-monetary reserve, $\alpha$ = rate of borrowing/lending to central bank, cost:

$$J(\alpha;\bar{m}) = \mathbb{E}\left[\int_0^T \left[\frac{1}{2}\alpha_t^2 - q\alpha_t(\bar{m}_t - X_t) + \frac{\epsilon}{2}(\bar{m}_t - X_t)^2\right]dt + \frac{c}{2}(\bar{m}_T - X_T)^2\right]$$

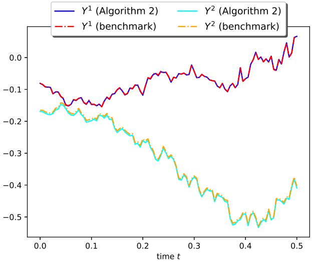
where $\bar{m} = (\bar{m}_t)_{t\geq 0}$ = conditional mean of the population states given $W^0$, and

$$dX_t = [a(\bar{m}_t - X_t) + \alpha_t]dt + \sigma\left(\sqrt{1-\rho^2}dW_t + \rho\, dW_t^0\right)$$

**NN** for FBSDE system VS (semi) analytical solution (LQ structure)




Samples of $X$




Samples of $Y$

More details in [Carmona and Laurière, 2022]

# Numerical Illustration 2: LQ MFG with Common Noise

**Example: MFG for inter-bank borrowing/lending** [Carmona et al., 2015]

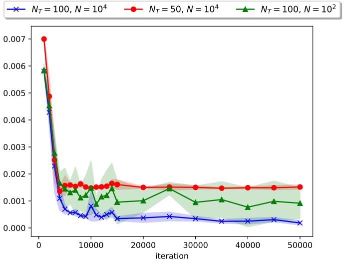
$X$ = log-monetary reserve, $\alpha$ = rate of borrowing/lending to central bank, cost:

$$J(\alpha;\bar{m}) = \mathbb{E}\left[\int_0^T \left[\frac{1}{2}\alpha_t^2 - q\alpha_t(\bar{m}_t - X_t) + \frac{\epsilon}{2}(\bar{m}_t - X_t)^2\right] dt + \frac{c}{2}(\bar{m}_T - X_T)^2\right]$$

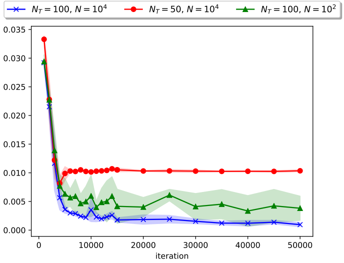
where $\bar{m} = (\bar{m}_t)_{t\geq 0}$ = conditional mean of the population states given $W^0$, and

$$dX_t = [a(\bar{m}_t - X_t) + \alpha_t]dt + \sigma\left(\sqrt{1-\rho^2}dW_t + \rho\, dW_t^0\right)$$

**NN** for FBSDE system VS (semi) analytical solution (LQ structure)




$L^2$ error on $X$




$L^2$ error on $Y$

More details in [Carmona and Laurière, 2022]

# Sample code

## Code

Sample code to illustrate: IPython notebook

https://colab.research.google.com/drive/1RXmRaf2OqtNensLMt0ASn25PfSC58saY?usp=sharing

- Deep learning for MKV FBSDEs
- Applied to the systemic risk model discussed above

# Comments

- Convergence of the DeepBSDE method [Han and Long, 2020]
- Extension to finite-player games [Han et al., 2022]
- Analysis of the different types of errors to be done for MKV case
- The new MFC problem is not standard
- Deep learning of MKV FBSDEs as presented here [Carmona and Laurière, 2022]; see also [Carmona and Laurière, 2023]
- Related works on deep learning for MKV FBSDEs: [Fouque and Zhang, 2020] (MFC with delay), [Germain et al., 2019], [Aurell et al., 2022b], . . .
- Similar "shooting" strategy can be applied to (infinite-dimensional) ODE systems obtained in graphon games [Aurell et al., 2022a]. Code (Gökçe Dayanıklı):

  `https://github.com/gokce-d/GraphonEpidemics`

# Outline

# Outline

# Stackelberg MFG

MFG with a Stackelberg (leader-follower) structure:

- A Principal chooses a policy $\lambda$
- A population of agents react and form a Nash equilibrium:

$$J^{\boldsymbol{\lambda}}(\boldsymbol{\alpha}, \boldsymbol{\mu}) := \mathbb{E}\left[\int_0^T f(t, X_t, \alpha_t, \mu_t; \lambda(t))dt + g(X_T, \mu_T; \lambda(T))\right],$$

- This is an MFG parameterized by $\lambda$
- The resulting mean field flow $\hat{\boldsymbol{\mu}}^{\lambda}$ incurs a cost to the principal

$$J^0(\boldsymbol{\lambda}) := \int_0^T f_0(t, \hat{\mu}_t^{\boldsymbol{\lambda}}, \lambda(t))dt + g_0(\hat{\mu}_T^{\boldsymbol{\lambda}}, \lambda(T))$$

Related works: Holmström-Milgrom (1987), Sannikov (2008, 2013), Djehiche-Helgesson (2014), Cvitanić *et al* (2018), Carmona-Wang (2018), Elie *et al* (2019)

# DL for Stackelberg MFG

Reminder:

- MFG solution can be characterized using a MKV FBSDE system
- This MKV FBSDE can be rewritten as a control problem
  - 2 forward equations
  - terminal cost

Stackelberg MFG:

- The above terminal cost can be combined with the principal's cost
- We obtain an MFC problem [Elie et al., 2019]
- From here we can apply the methods discussed previously

# DL for Stackelberg MFG

Reminder:

- MFG solution can be characterized using a MKV FBSDE system
- This MKV FBSDE can be rewritten as a control problem
  - 2 forward equations
  - terminal cost

Stackelberg MFG:

- The above terminal cost can be combined with the principal's cost
- We obtain an MFC problem [Elie et al., 2019]
- From here we can apply the methods discussed previously

For more details, see:

- [Aurell et al., 2022b] with application to epidemics management (finite state MFG): the principal gives guidelines (social distancing, etc.) and the population reacts
- **Code available** ((Gökçe Dayanıklı)):

  `https://github.com/gokce-d/StackelbergMFG`
- Extension to other Stackelberg MFGs: [Dayanikli and Lauriere, 2023]

# Outline

1. Introduction

2. Deep Learning for MFC

3. Deep Learning for MKV FBSDE

4. Two Examples of Extensions
   - Solving Stackelberg MFG with Deep MKV FBSDE
   - Computing MFC Value Function with DBDP

5. Conclusion

# Social optimum: Mean Field Control

Reminder from lecture 2 about mean field (type) control or control of McKean-Vlasov (MKV) dynamics

**Definition (Mean field control (MFC) problem)**

$\alpha^*$ is a solution to the MFC problem if it minimizes

$$J^{MFC}(\alpha) = \mathbb{E}\left[\int_0^T f(X_t^\alpha, \alpha_t, m_t^\alpha)dt + g(X_T^\alpha, m_T^\alpha)\right].$$

Main difference with MFG: here not only $X$ but $m$ too is controlled by $\alpha$.

## Social optimum: Mean Field Control

Reminder from lecture 2 about mean field (type) control or control of McKean-Vlasov (MKV) dynamics

**Definition (Mean field control (MFC) problem)**

$\alpha^*$ is a solution to the MFC problem if it minimizes

$$J^{MFC}(\alpha) = \mathbb{E}\left[\int_0^T f(X_t^\alpha, \alpha_t, m_t^\alpha)dt + g(X_T^\alpha, m_T^\alpha)\right].$$

Main difference with MFG: here not only $X$ but $m$ too is controlled by $\alpha$.

**Optimality conditions?** Several approaches:

- Dynamic programming value function depending on $m$; value function $V$
- Calculus of variations taking $m$ as a state; adjoint state $u$
- Pontryagin's maximum principle for the (MKV process) $X$; adjoint state $Y$

**Dynamic programming** for MFC [Laurière and Pironneau, 2014], [Bensoussan et al., 2015], [Pham and Wei, 2017], [Djete et al., 2022], ...

$\to$ Algorithm?

# DBDP for Non-Mean Field Control

For standard (non-mean field) stochastic optimal control problems, [Huré et al., 2019] have introduced the Deep Backward Dynamic Programming (DBDP):

**Idea:** learn $Y_n$ and $Z_n$ at each $n$ as functions of $X_n$, backward in time:

- Initialize $\hat{Y}_{N_T} = g$ and then, for $n = N_T - 1, \dots, 0$, either:
- Version 1: Let $(\hat{Y}_n, \hat{Z}_n)$ = minimizer over $(Y_n, Z_n)$ of:

$$\mathbb{E}\left[|\hat{Y}_{n+1}(X_{n+1}) - Y_n(X_n) - f(t_n, X_n, Y_n(X_n), Z_n(X_n))\Delta t - Z_n(X_n) \cdot \Delta W_{n+1}|\right]$$

# DBDP for Non-Mean Field Control

For standard (non-mean field) stochastic optimal control problems, [Huré et al., 2019] have introduced the Deep Backward Dynamic Programming (DBDP):

**Idea:** learn $Y_n$ and $Z_n$ at each $n$ as functions of $X_n$, backward in time:

- Initialize $\hat{Y}_{N_T} = g$ and then, for $n = N_T - 1, \dots, 0$, either:
- Version 1: Let $(\hat{Y}_n, \hat{Z}_n)$ = minimizer over $(Y_n, Z_n)$ of:

$$\mathbb{E}\left[|\hat{Y}_{n+1}(X_{n+1}) - Y_n(X_n) - f(t_n, X_n, Y_n(X_n), Z_n(X_n))\Delta t - Z_n(X_n) \cdot \Delta W_{n+1}|\right]$$

- or Version 2: Let $(\hat{Y}_n, \hat{Z}_n)$ = minimizer over $(Y_n, Z_n)$ of:

$$\mathbb{E}\left[|\hat{Y}_{n+1}(X_{n+1}) - Y_n(X_n) - f(t_n, X_n, Y_n(X_n), \sigma^\top D_x Y_n(X_n))\Delta t - D_x Y_n(X_n)^\top \sigma \Delta W_{n+1}|\right]$$

# DBDP for Non-Mean Field Control

For standard (non-mean field) stochastic optimal control problems, [Huré et al., 2019] have introduced the Deep Backward Dynamic Programming (DBDP):

**Idea:** learn $Y_n$ and $Z_n$ at each $n$ as functions of $X_n$, backward in time:

- Initialize $\hat{Y}_{N_T} = g$ and then, for $n = N_T - 1, \dots, 0$, either:
- Version 1: Let $(\hat{Y}_n, \hat{Z}_n)$ = minimizer over $(Y_n, Z_n)$ of:

$$\mathbb{E}\left[|\hat{Y}_{n+1}(X_{n+1}) - Y_n(X_n) - f(t_n, X_n, Y_n(X_n), Z_n(X_n))\Delta t - Z_n(X_n) \cdot \Delta W_{n+1}|\right]$$

- or Version 2: Let $(\hat{Y}_n, \hat{Z}_n)$ = minimizer over $(Y_n, Z_n)$ of:

$$\mathbb{E}\left[|\hat{Y}_{n+1}(X_{n+1}) - Y_n(X_n) - f(t_n, X_n, Y_n(X_n), \sigma^\top D_x Y_n(X_n))\Delta t - D_x Y_n(X_n)^\top \sigma \Delta W_{n+1}|\right]$$

For more details on deep learning methods for (non-mean field) optimal control problems, see e.g. [Germain et al., 2021b]

# DBDP for MFC

- Can we apply the same idea to MFC, replacing $V$ by a neural network?
- Main challenge: the value function $V$ takes $m \in \mathcal{P}(\mathbb{R}^d)$ as an input
- We need to approximate $m$

# DBDP for MFC

- Can we apply the same idea to MFC, replacing $V$ by a neural network?
- Main challenge: the value function $V$ takes $m \in \mathcal{P}(\mathbb{R}^d)$ as an input
- We need to approximate $m$
- One possibility:

$$V(t, m_t) \approx \tilde{V}(t, m_t^N) \approx \tilde{V}_\theta(t, X_t^1, \dots, X_t^N)$$

  where $\tilde{V}_\theta$ is a neural network which is symmetric with respect to the inputs
- See the next lecture for more details
- See [Germain et al., 2021a] for more details about the implementation and [Germain et al., 2022] for the analysis
- See also e.g. [Dayanikli et al., 2023] for different approximations of the population

# Outline

# Summary

- Two algorithms based on the stochastic approach
- Direct approach without any optimality condition
- DeepBSDE: recasting (MKV) FBSDEs as control problems
- Many possible extensions and variations
- Some surveys on DL for control/games:
  [Germain et al., 2021b, Carmona and Laurière, 2023, Hu and Laurière, 2023]

Next lecture: deep learning methods for the PDE approach

Thank you for your attention

Questions?

Feel free to reach out: `mathieu.lauriere@nyu.edu`

# References I

[Achdou et al., 2012] Achdou, Y., Camilli, F., and Capuzzo-Dolcetta, I. (2012).
Mean field games: numerical methods for the planning problem.
*SIAM J. Control Optim.*, 50(1):77–109.

[Achdou and Capuzzo-Dolcetta, 2010] Achdou, Y. and Capuzzo-Dolcetta, I. (2010).
Mean field games: numerical methods.
*SIAM J. Numer. Anal.*, 48(3):1136–1162.

[Achdou and Lasry, 2019] Achdou, Y. and Lasry, J.-M. (2019).
Mean field games for modeling crowd motion.
In Chetverushkin, B. N., Fitzgibbon, W., Kuznetsov, Y. A., Neittaanmäki, P., Periaux, J., and Pironneau, O., editors, *Contributions to Partial Differential Equations and Applications*, chapter 4, pages 17–42. Springer International Publishing.

[Achdou and Laurière, 2016] Achdou, Y. and Laurière, M. (2016).
Mean Field Type Control with Congestion (II): An augmented Lagrangian method.
*Appl. Math. Optim.*, 74(3):535–578.

[Agram et al., 2020] Agram, N., Bakdi, A., and Oksendal, B. (2020).
Deep learning and stochastic mean-field control for a neural network model.
*Available at SSRN 3639022*.

[Almulla et al., 2017] Almulla, N., Ferreira, R., and Gomes, D. (2017).
Two numerical approaches to stationary mean-field games.
*Dyn. Games Appl.*, 7(4):657–682.

[Andreev, 2017a] Andreev, R. (2017a).
Preconditioning the augmented Lagrangian method for instationary mean field games with diffusion.
*SIAM J. Sci. Comput.*, 39(6):A2763–A2783.

[Andreev, 2017b] Andreev, R. (2017b).
Preconditioning the augmented lagrangian method for instationary mean field games with diffusion.
*SIAM Journal on Scientific Computing*, 39(6):A2763–A2783.

[Angiuli et al., 2019] Angiuli, A., Graves, C. V., Li, H., Chassagneux, J.-F., Delarue, F., and Carmona, R. (2019).
Cemracs 2017: numerical probabilistic approach to MFG.
*ESAIM: ProcS*, 65:84–113.

[Aurell et al., 2022a] Aurell, A., Carmona, R., Dayanıklı, G., and Laurière, M. (2022a).
Finite state graphon games with applications to epidemics.
*Dynamic Games and Applications*, 12(1):49–81.

[Aurell et al., 2022b] Aurell, A., Carmona, R., Dayanikli, G., and Lauriere, M. (2022b).
Optimal incentives to mitigate epidemics: a stackelberg mean field game approach.
*SIAM Journal on Control and Optimization*, 60(2):S294–S322.

# References III


[Balata et al., 2019] Balata, A., Huré, C., Laurière, M., Pham, H., and Pimentel, I. (2019).
A class of finite-dimensional numerically solvable mckean-vlasov control problems.
*ESAIM: Proceedings and Surveys*, 65:114–144.

[Bayraktar et al., 2018] Bayraktar, E., Budhiraja, A., and Cohen, A. (2018).
A numerical scheme for a mean field game in some queueing systems based on markov chain approximation method.
*SIAM Journal on Control and Optimization*, 56(6):4017–4044.

[Benamou and Carlier, 2015] Benamou, J.-D. and Carlier, G. (2015).
Augmented lagrangian methods for transport optimization, mean field games and degenerate elliptic equations.
*Journal of Optimization Theory and Applications*, 167(1):1–26.

[Bensoussan et al., 2015] Bensoussan, A., Frehse, J., and Yam, S. C. P. (2015).
The master equation in mean field theory.
*J. Math. Pures Appl. (9)*, 103(6):1441–1474.

[Borkar, 2009] Borkar, V. S. (2009).
*Stochastic approximation: a dynamical systems viewpoint*, volume 48.
Springer.

[Bossy and Talay, 1997] Bossy, M. and Talay, D. (1997).
A stochastic particle method for the McKean-Vlasov and the Burgers equation.
*Math. Comp.*, 66(217):157–192.

[Bottou, 2012] Bottou, L. (2012).
Stochastic gradient descent tricks.
In *Neural Networks: Tricks of the Trade: Second Edition*, pages 421–436. Springer.

[Briceño Arias et al., 2019] Briceño Arias, L. M., Kalise, D., Kobeissi, Z., Laurière, M., Mateos González, A., and Silva, F. J. (2019).
On the implementation of a primal-dual algorithm for second order time-dependent mean field games with local couplings.
*ESAIM: ProcS*, 65:330–348.

[Briceño Arias et al., 2018] Briceño Arias, L. M., Kalise, D., and Silva, F. J. (2018).
Proximal methods for stationary mean field games with local couplings.
*SIAM J. Control Optim.*, 56(2):801–836.

[Cacace et al., 2021] Cacace, S., Camilli, F., and Goffi, A. (2021).
A policy iteration method for mean field games.
*ESAIM: Control, Optimisation and Calculus of Variations*, 27:85.

[Calzola et al., 2022] Calzola, E., Carlini, E., and Silva, F. J. (2022).
A high-order lagrange-galerkin scheme for a class of fokker-planck equations and applications to mean field games.
*arXiv preprint arXiv:2207.08463*.

# References V

[Camilli and Tang, 2022] Camilli, F. and Tang, Q. (2022).
Rates of convergence for the policy iteration method for mean field games systems.
*Journal of Mathematical Analysis and Applications*, 512(1):126138.

[Carlini and Silva, 2014] Carlini, E. and Silva, F. J. (2014).
A fully discrete semi-Lagrangian scheme for a first order mean field game problem.
*SIAM J. Numer. Anal.*, 52(1):45–67.

[Carlini and Silva, 2015] Carlini, E. and Silva, F. J. (2015).
A semi-Lagrangian scheme for a degenerate second order mean field game system.
*Discrete Contin. Dyn. Syst.*, 35(9):4269–4292.

[Carlini and Silva, 2018] Carlini, E. and Silva, F. J. (2018).
On the discretization of some nonlinear fokker–planck–kolmogorov equations and applications.
*SIAM Journal on Numerical Analysis*, 56(4):2148–2177.

[Carmona and Delarue, 2018] Carmona, R. and Delarue, F. (2018).
*Probabilistic theory of mean field games with applications. I*, volume 83 of *Probability Theory and Stochastic Modelling*.
Springer, Cham.
Mean field FBSDEs, control, and games.

# References VI


[Carmona et al., 2015] Carmona, R., Fouque, J.-P., and Sun, L.-H. (2015).
Mean field games and systemic risk.
*Commun. Math. Sci.*, 13(4):911–933.

[Carmona and Lacker, 2015] Carmona, R. and Lacker, D. (2015).
A probabilistic weak formulation of mean field games and applications.
*Ann. Appl. Probab.*, 25(3):1189–1231.

[Carmona and Laurière, 2021] Carmona, R. and Laurière, M. (2021).
Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games i: The ergodic case.
*SIAM Journal on Numerical Analysis*, 59(3):1455–1485.

[Carmona and Laurière, 2022] Carmona, R. and Laurière, M. (2022).
Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: Ii—the finite horizon case.
*The Annals of Applied Probability*, 32(6):4065–4105.

[Carmona and Laurière, 2023] Carmona, R. and Laurière, M. (2023).
Deep learning for mean field games and mean field control with applications to finance.
*Machine Learning and Data Sciences for Financial Markets: A Guide to Contemporary Practices*, page 369.

[Chassagneux et al., 2019] Chassagneux, J.-F., Crisan, D., and Delarue, F. (2019).
Numerical method for FBSDEs of McKean-Vlasov type.
*Ann. Appl. Probab.*, 29(3):1640–1684.

[Cui and Koeppl, 2021] Cui, K. and Koeppl, H. (2021).
Approximately solving mean field games via entropy-regularized deep reinforcement learning.
In *International Conference on Artificial Intelligence and Statistics*, pages 1909–1917. PMLR.

[Cybenko, 1989] Cybenko, G. (1989).
Approximation by superpositions of a sigmoidal function.
*Mathematics of control, signals and systems*, 2(4):303–314.

[Dayanikli and Lauriere, 2023] Dayanikli, G. and Lauriere, M. (2023).
A machine learning method for stackelberg mean field games.
*arXiv preprint arXiv:2302.10440*.

[Dayanikli et al., 2023] Dayanikli, G., Lauriere, M., and Zhang, J. (2023).
Deep learning for population-dependent controls in mean field control problems.
*arXiv preprint arXiv:2306.04788*.

[de Raynal and Trillos, 2015] de Raynal, P. C. and Trillos, C. G. (2015).
A cubature based algorithm to solve decoupled mckean–vlasov forward–backward stochastic differential equations.
*Stochastic Processes and their Applications*, 125(6):2206–2255.

# References VIII


[Després, 2022] Després, B. (2022).
*Neural Networks and Numerical Analysis*, volume 6.
Walter de Gruyter GmbH & Co KG.

[Djete et al., 2022] Djete, M. F., Possamaï, D., and Tan, X. (2022).
Mckean–vlasov optimal control: the dynamic programming principle.
*The Annals of Probability*, 50(2):791–833.

[E et al., 2017] E, W., Han, J., and Jentzen, A. (2017).
Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations.
*Commun. Math. Stat.*, 5(4):349–380.

[Elie et al., 2019] Elie, R., Mastrolia, T., and Possamaï, D. (2019).
A tale of a principal and many, many agents.
*Mathematics of Operations Research*, 44(2):440–467.

[Fouque and Zhang, 2020] Fouque, J.-P. and Zhang, Z. (2020).
Deep learning methods for mean field control problems with delay.
*Frontiers in Applied Mathematics and Statistics*, 6:11.

[Germain et al., 2021a] Germain, M., Laurière, M., Pham, H., and Warin, X. (2021a).
Deepsets and their derivative networks for solving symmetric pdes.
*arXiv preprint arXiv:2103.00838*.

[Germain et al., 2019] Germain, M., Mikael, J., and Warin, X. (2019).
Numerical resolution of mckean-vlasov fbsdes using neural networks.
*arXiv preprint arXiv:1909.12678*.

[Germain et al., 2021b] Germain, M., Pham, H., and Warin, X. (2021b).
Neural networks-based algorithms for stochastic control and pdes in finance.
*arXiv preprint arXiv:2101.08068*.

[Germain et al., 2022] Germain, M., Pham, H., and Warin, X. (2022).
Rate of convergence for particle approximation of pdes in wasserstein space.
*Journal of Applied Probability*, 59(4):992–1008.

[Gobet and Munos, 2005] Gobet, E. and Munos, R. (2005).
Sensitivity analysis using Itô-Malliavin calculus and martingales, and application to stochastic optimal control.
*SIAM J. Control Optim.*, 43(5):1676–1713.

[Gomes and Saúde, 2018] Gomes, D. A. and Saúde, J. (2018).
Numerical methods for finite-state mean-field games satisfying a monotonicity condition.
*Applied Mathematics & Optimization*.

[Gomes and Yang, 2020] Gomes, D. A. and Yang, X. (2020).
The hessian riemannian flow and newton's method for effective hamiltonians and mather measures.
*ESAIM: Mathematical Modelling and Numerical Analysis*, 54(6):1883–1915.

# References X


[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016).
*Deep learning*.
MIT press.

[Han and E, 2016] Han, J. and E, W. (2016).
Deep learning approximation for stochastic control problems.
*Deep Reinforcement Learning Workshop, NIPS, arXiv preprint arXiv:1611.07422*.

[Han et al., 2022] Han, J., Hu, R., and Long, J. (2022).
Convergence of deep fictitious play for stochastic differential games.
*Frontiers of Mathematical Finance*, 1(2):279–311.

[Han and Long, 2020] Han, J. and Long, J. (2020).
Convergence of the deep bsde method for coupled fbsdes.
*Probability, Uncertainty and Quantitative Risk*, 5:1–33.

[Hornik, 1991] Hornik, K. (1991).
Approximation capabilities of multilayer feedforward networks.
*Neural networks*, 4(2):251–257.

[Hu, 2021] Hu, R. (2021).
Deep fictitious play for stochastic differential games.
*Communications in Mathematical Sciences*, 19(2):325–353.

[Hu and Laurière, 2023] Hu, R. and Laurière, M. (2023).
Recent developments in machine learning methods for stochastic control and games.
*arXiv preprint arXiv:2303.10257*.

[Huré et al., 2019] Huré, C., Pham, H., and Warin, X. (2019).
Some machine learning schemes for high-dimensional nonlinear pdes.
*arXiv preprint arXiv:1902.01599*, page 2.

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

[Kohlmann and Zhou, 2000] Kohlmann, M. and Zhou, X. Y. (2000).
Relationship between backward stochastic differential equations and stochastic controls: a linear-quadratic approach.
*SIAM Journal on Control and Optimization*, 38(5):1392–1407.

[Laurière and Pironneau, 2014] Laurière, M. and Pironneau, O. (2014).
Dynamic programming for mean-field type control.
*C. R. Math. Acad. Sci. Paris*, 352(9):707–713.

[Laurière and Pironneau, 2016] Laurière, M. and Pironneau, O. (2016).
Dynamic programming for mean-field type control.
*J. Optim. Theory Appl.*, 169(3):902–924.

[Laurière et al., 2023] Laurière, M., Song, J., and Tang, Q. (2023).
Policy iteration method for time-dependent mean field games systems with non-separable hamiltonians.
*Applied Mathematics & Optimization*, 87(2):17.

[Lavigne and Pfeiffer, 2022] Lavigne, P. and Pfeiffer, L. (2022).
Generalized conditional gradient and learning in potential mean field games.
*arXiv preprint arXiv:2209.12772*.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015).
Deep learning.
*nature*, 521(7553):436–444.

[Leijnen and Veen, 2020] Leijnen, S. and Veen, F. v. (2020).
The neural network zoo.
In *Proceedings*, volume 47, page 9. MDPI.

[Mhaskar and Micchelli, 1995] Mhaskar, H. N. and Micchelli, C. A. (1995).
Degree of approximation by neural and translation networks with a single hidden layer.
*Advances in Applied Mathematics*, 16:151–183.

[Pfeiffer, 2016] Pfeiffer, L. (2016).
Numerical methods for mean-field type optimal control problems.
*Pure Appl. Funct. Anal.*, 1(4):629–655.

[Pham and Wei, 2017] Pham, H. and Wei, X. (2017).
Dynamic programming for optimal control of stochastic McKean-Vlasov dynamics.
*SIAM J. Control Optim.*, 55(2):1069–1101.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951).
A stochastic approximation method.
*The annals of mathematical statistics*, pages 400–407.

[Salhab et al., 2015] Salhab, R., Malhamé, R. P., and Le Ny, J. (2015).
A dynamic game model of collective choice in multi-agent systems.
In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, pages 4444–4449, Osaka, Japon.

[Sannikov, 2008] Sannikov, Y. (2008).
A continuous-time version of the principal-agent problem.
*The Review of Economic Studies*, 75(3):957–984.

[Tang and Song, 2022] Tang, Q. and Song, J. (2022).
Learning optimal policies in potential mean field games: Smoothed policy iteration algorithms.
*arXiv preprint arXiv:2212.04791*.