

Mean Field Games:
Numerical Methods and
Applications in Machine Learning
Part 6: Deep Learning for MFG PDEs

Mathieu LAURIÈRE

<https://mlauriere.github.io/teaching/MFG-PKU-6.pdf>

Peking University
Summer School on Applied Mathematics
July 26 – August 6, 2021

RECAP

1. Deep Galerkin Method for MFG PDEs

- Warm-up: ODE
- Solving MFG PDE system
- Link with Generative Adversarial Networks

2. Master Equation

1. Deep Galerkin Method for MFG PDEs

- Warm-up: ODE
- Solving MFG PDE system
- Link with Generative Adversarial Networks

2. Master Equation

- Look for $\varphi : \mathbb{R} \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), \varphi'(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi(a), \varphi'(a), \dots) = 0 \end{cases}$$

- Look for $\varphi : \mathbb{R} \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), \varphi'(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi(a), \varphi'(a), \dots) = 0 \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), \varphi'_\theta(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi_\theta(a), \varphi'_\theta(a), \dots) = 0 \end{cases}$$

- Look for $\varphi : \mathbb{R} \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), \varphi'(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi(a), \varphi'(a), \dots) = 0 \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), \varphi'_\theta(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi_\theta(a), \varphi'_\theta(a), \dots) = 0 \end{cases}$$

- Rephrase as minimization problem: minimizer over θ

$$\begin{aligned} & \mathbb{E}_{X \sim \mathcal{U}([a, b])} [|F(X, \varphi_\theta(X), \varphi'_\theta(X), \dots)|^2] \\ & + |G(a, \varphi_\theta(a), \varphi'_\theta(a), \dots)|^2 \end{aligned}$$

- Look for $\varphi : \mathbb{R} \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), \varphi'(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi(a), \varphi'(a), \dots) = 0 \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), \varphi'_\theta(x), \dots) = 0, & x \in [a, b] \\ G(a, \varphi_\theta(a), \varphi'_\theta(a), \dots) = 0 \end{cases}$$

- Rephrase as minimization problem: minimizer over θ

$$\begin{aligned} & \mathbb{E}_{X \sim \mathcal{U}([a, b])} [|F(X, \varphi_\theta(X), \varphi'_\theta(X), \dots)|^2] \\ & + |G(a, \varphi_\theta(a), \varphi'_\theta(a), \dots)|^2 \end{aligned}$$

- Use SGD

Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$

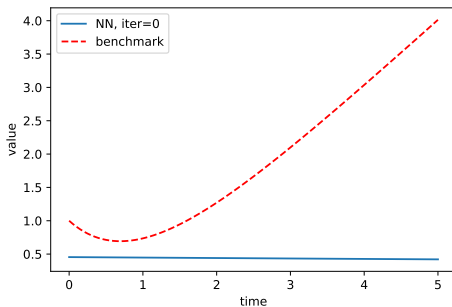
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



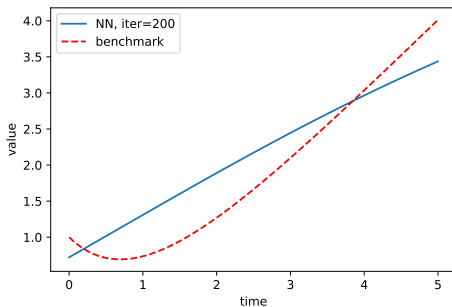
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



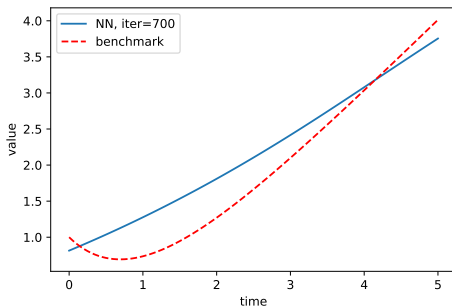
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



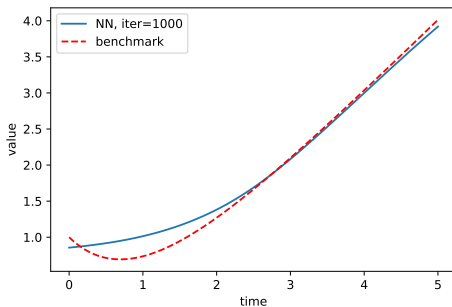
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



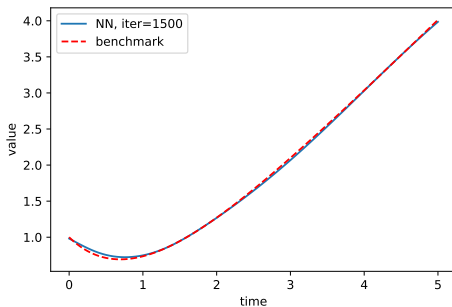
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



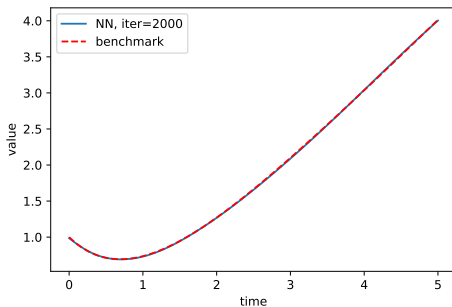
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



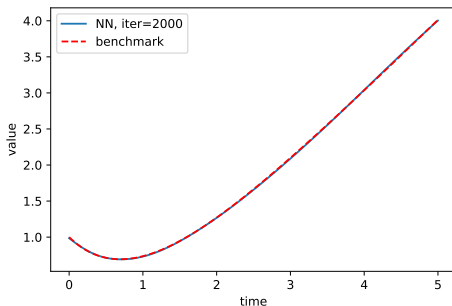
Numerical Illustration

Application to the ODE:

$$\begin{cases} F(x, \varphi(x), \varphi'(x)) = \varphi'(x) - (x - \varphi(x)), & x \in [0, 5] \\ \varphi(0) = 1 \end{cases}$$

Solution:

$$\varphi(x) = x - 1 + 2e^{-x}$$



https://colab.research.google.com/drive/1LHuV1oE6eyO6AQgw3joQjow_uozQWSTw?usp=sharing

1. Deep Galerkin Method for MFG PDEs

- Warm-up: ODE
- Solving MFG PDE system
- Link with Generative Adversarial Networks

2. Master Equation

Deep Galerkin Method (DGM), proposed by [Sirignano & Spiliopoulos \[SS18\]](#)¹

- Look for $\varphi : \mathbb{R}^d \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

¹ Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375, 1339-1364.

Deep Galerkin Method (DGM), proposed by Sirignano & Spiliopoulos [SS18]¹

- Look for $\varphi : \mathbb{R}^d \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

¹ Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375, 1339-1364.

Deep Galerkin Method (DGM), proposed by Sirignano & Spiliopoulos [SS18]¹

- Look for $\varphi : \mathbb{R}^d \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Rephrase as minimization problem: minimizer over θ

$$\begin{aligned} & \mathbb{E}_{X \sim \mathcal{U}(\Omega)} [|F(X, \varphi_\theta(X), D\varphi_\theta(X), D^2\varphi_\theta(X), \dots)|^2] \\ & + \mathbb{E}_{Y \sim \mathcal{U}(\partial\Omega)} [|G(Y, \varphi_\theta(Y), D\varphi_\theta(Y), D^2\varphi_\theta(Y), \dots)|^2] \end{aligned}$$

¹ Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375, 1339-1364.

Deep Galerkin Method (DGM), proposed by Sirignano & Spiliopoulos [SS18]¹

- Look for $\varphi : \mathbb{R}^d \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Rephrase as minimization problem: minimizer over θ

$$\begin{aligned} & \mathbb{E}_{X \sim \mathcal{U}(\Omega)} [|F(X, \varphi_\theta(X), D\varphi_\theta(X), D^2\varphi_\theta(X), \dots)|^2] \\ & + \mathbb{E}_{Y \sim \mathcal{U}(\partial\Omega)} [|G(Y, \varphi_\theta(Y), D\varphi_\theta(Y), D^2\varphi_\theta(Y), \dots)|^2] \end{aligned}$$

- Use SGD

¹ Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375, 1339-1364.

Deep Galerkin Method (DGM), proposed by Sirignano & Spiliopoulos [SS18]¹

- Look for $\varphi : \mathbb{R}^d \ni x \mapsto \varphi(x) \in \mathbb{R}$ s.t.

$$\begin{cases} F(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi(x), D\varphi(x), D^2\varphi(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Look among NN φ_θ

$$\begin{cases} F(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \Omega \\ G(x, \varphi_\theta(x), D\varphi_\theta(x), D^2\varphi_\theta(x), \dots) = 0, & x \in \partial\Omega \end{cases}$$

- Rephrase as minimization problem: minimizer over θ

$$\begin{aligned} & \mathbb{E}_{X \sim \mathcal{U}(\Omega)} [|F(X, \varphi_\theta(X), D\varphi_\theta(X), D^2\varphi_\theta(X), \dots)|^2] \\ & + \mathbb{E}_{Y \sim \mathcal{U}(\partial\Omega)} [|G(Y, \varphi_\theta(Y), D\varphi_\theta(Y), D^2\varphi_\theta(Y), \dots)|^2] \end{aligned}$$

- Use SGD
- Remarks on the implementation:
 - ▶ Choice of distribution
 - ▶ Boundary conditions
 - ▶ Higher order derivatives computation

¹ Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375, 1339-1364.

- Let $\vec{x} = (t, x)$ be the input
- Architecture: $L + 1$ hidden layers (\odot denotes element-wise multiplication):

$$\begin{aligned}S^1 &= \sigma(W^1 \vec{x} + b^1), \\Z^\ell &= \sigma(U^{z,\ell} \vec{x} + W^{z,\ell} S^\ell + b^{z,\ell}), \quad \ell = 1, \dots, L, \\G^\ell &= \sigma(U^{g,\ell} \vec{x} + W^{g,\ell} S^1 + b^{g,\ell}), \quad \ell = 1, \dots, L, \\R^\ell &= \sigma(U^{r,\ell} \vec{x} + W^{r,\ell} S^\ell + b^{r,\ell}), \quad \ell = 1, \dots, L, \\H^\ell &= \sigma(U^{h,\ell} \vec{x} + W^{h,\ell} (S^\ell \odot R^\ell) + b^{h,\ell}), \quad \ell = 1, \dots, L, \\S^{\ell+1} &= (1 - G^\ell) \odot H^\ell + Z^\ell \odot S^\ell, \quad \ell = 1, \dots, L, \\f(t, x; \theta) &= WS^{L+1} + b,\end{aligned}$$

- The parameters are

$$\theta = \left\{ W^1, b^1, \left(U^{\alpha,\ell}, W^{\alpha,\ell}, b^{\alpha,\ell} \right)_{\ell=1, \dots, L, \alpha \in \{z, g, r, h\}}, W, b \right\}.$$

- The number of units in each layer is M and $\sigma : \mathbb{R}^M \rightarrow \mathbb{R}^M$ is an element-wise nonlinearity:

$$\sigma(z) = \left(\phi(z_1), \phi(z_2), \dots, \phi(z_M) \right),$$

where $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is a nonlinear activation function.

Reminder: (m, u) solving, on $[0, T] \times \mathbb{T}^d$,

$$\begin{cases} 0 = -\frac{\partial u}{\partial t}(t, x) - \nu \Delta u(t, x) + H(x, m(t, \cdot), \nabla u(t, x)) \\ 0 = \frac{\partial m}{\partial t}(t, x) - \nu \Delta m(t, x) - \operatorname{div}(m(t, \cdot) \partial_p H(\cdot, m(t), \nabla u(t, \cdot))) (x) \\ u(T, x) = g(x, m(T, \cdot)), \quad m(0, x) = m_0(x) \end{cases}$$

Reminder: (m, u) solving, on $[0, T] \times \mathbb{T}^d$,

$$\begin{cases} 0 = -\frac{\partial u}{\partial t}(t, x) - \nu \Delta u(t, x) + H(x, m(t, \cdot), \nabla u(t, x)) \\ 0 = \frac{\partial m}{\partial t}(t, x) - \nu \Delta m(t, x) - \operatorname{div}(m(t, \cdot) \partial_p H(\cdot, m(t, \cdot), \nabla u(t, \cdot))) (x) \\ u(T, x) = g(x, m(T, \cdot)), \quad m(0, x) = m_0(x) \end{cases}$$

Or ergodic version: (m, u, λ) on \mathbb{T}^d

$$\begin{cases} 0 = -\nu \Delta u(x) + H(x, m(\cdot), \nabla u(x)) + \lambda \\ 0 = -\nu \Delta m(x) - \operatorname{div}(m(\cdot) \partial_p H(\cdot, m, \nabla u(\cdot))) (x) \\ \int u(x) dx = 0, \quad \int m(x) dx = 1, m > 0 \end{cases}$$

See [Lasry & Lions \[LL07\]](#), [Bensoussan, Frehse and Yam \[BFY13, Chapter 7\]](#)

Reminder: (m, u) solving, on $[0, T] \times \mathbb{T}^d$,

$$\begin{cases} 0 = -\frac{\partial u}{\partial t}(t, x) - \nu \Delta u(t, x) + H(x, m(t, \cdot), \nabla u(t, x)) \\ 0 = \frac{\partial m}{\partial t}(t, x) - \nu \Delta m(t, x) - \operatorname{div}(m(t, \cdot) \partial_p H(\cdot, m(t, \cdot), \nabla u(t, \cdot))) (x) \\ u(T, x) = g(x, m(T, \cdot)), \quad m(0, x) = m_0(x) \end{cases}$$

Or ergodic version: (m, u, λ) on \mathbb{T}^d

$$\begin{cases} 0 = -\nu \Delta u(x) + H(x, m(\cdot), \nabla u(x)) + \lambda \\ 0 = -\nu \Delta m(x) - \operatorname{div}(m(\cdot) \partial_p H(\cdot, m, \nabla u(\cdot))) (x) \\ \int u(x) dx = 0, \quad \int m(x) dx = 1, m > 0 \end{cases}$$

See [Lasry & Lions \[LL07\]](#), [Bensoussan, Frehse and Yam \[BFY13, Chapter 7\]](#)

Analogous PDE systems for MFC problems

Numerical Illustration 1: Ergodic Example with Explicit Solution

Example (of MFC) with explicit solution on \mathbb{T}^d ($d = 10$)

Following [Almulla et al. \[AFG17\]](#), take

$$f(x, \textcolor{blue}{m}, \textcolor{red}{v}) = \frac{1}{2}|\textcolor{red}{v}|^2 + \tilde{f}(x) + \ln(\textcolor{blue}{m}(x)),$$

with $\tilde{f}(x) = 2\pi^2 \left[-\sum_{i=1}^d c \sin(2\pi x_i) + \sum_{i=1}^d |c \cos(2\pi x_i)|^2 \right] - 2 \sum_{i=1}^d c \sin(2\pi x_i)$,

then the solution is given by $\textcolor{green}{u}(x) = c \sum_{i=1}^d \sin(2\pi x_i)$ and $\textcolor{blue}{m}(x) = e^{2\textcolor{green}{u}(x)} / \int e^{2\textcolor{blue}{u}}$

Numerical Illustration 1: Ergodic Example with Explicit Solution

Example (of MFC) with explicit solution on \mathbb{T}^d ($d = 10$)

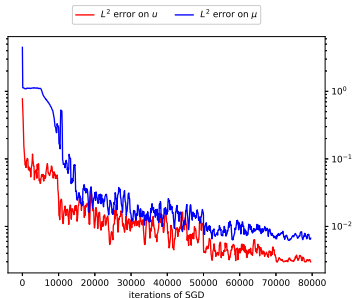
Following [Almulla et al. \[AFG17\]](#), take

$$f(x, m, v) = \frac{1}{2}|v|^2 + \tilde{f}(x) + \ln(m(x)),$$

with $\tilde{f}(x) = 2\pi^2 \left[-\sum_{i=1}^d c \sin(2\pi x_i) + \sum_{i=1}^d |c \cos(2\pi x_i)|^2 \right] - 2 \sum_{i=1}^d c \sin(2\pi x_i)$,

then the solution is given by $u(x) = c \sum_{i=1}^d \sin(2\pi x_i)$ and $m(x) = e^{2u(x)} / \int e^{2u}$

Error vs SGD iterations (see [\[Carmona, L'21\]](#)):



Relative L^2 error on u and m

Numerical Illustration 2: Ergodic Example without Explicit Solution

Example (of MFG) without explicit solution on \mathbb{T}^d ($d = 30$)

Inspired by [Achdou & Capuzzo-Dolcetta \[ACD10\]](#), take

$$f(x, m, v) = \frac{1}{2}|v|^2 + \tilde{f}(x) + |m(x)|^2,$$

with $\tilde{f}(x) = 2\pi^2 c \sum_{i=1}^d [\sin(2\pi x_i) + \cos(2\pi x_i)]$

Numerical Illustration 2: Ergodic Example without Explicit Solution

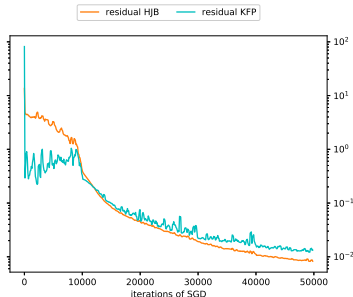
Example (of MFG) without explicit solution on \mathbb{T}^d ($d = 30$)

Inspired by [Achdou & Capuzzo-Dolcetta \[ACD10\]](#), take

$$f(x, m, v) = \frac{1}{2}|v|^2 + \tilde{f}(x) + |m(x)|^2,$$

with $\tilde{f}(x) = 2\pi^2 c \sum_{i=1}^d [\sin(2\pi x_i) + \cos(2\pi x_i)]$

PDE residuals vs SGD iterations (see [\[Carmona, L'21\]](#)):



L^2 norm of residuals for HJB and KFP

Numerical Illustration 3: Crowd Trading

Model of crowd trading by Cardaliaguet & Lehalle [CL18]:

$$\begin{cases} dS_t^{\bar{v}} = \gamma \bar{v}_t dt + \sigma dW_t & \text{(price)} \\ dQ_t^v = v_t dt & \text{(player's inventory)} \\ dX_t^{v, \bar{v}} = -v_t (S_t^{\bar{v}} + \kappa v_t) dt & \text{(player's wealth)} \end{cases}$$

Objective: given $(\bar{v}_t)_t$, maximize

$$\mathbb{E} \left[X_T^{v, \bar{v}} + Q_T^v S_T^{\bar{v}} - A |Q_T^v|^2 - \phi \int_0^T |Q_t^v|^2 dt \right]$$

where: $\phi, A > 0 \Rightarrow$ penalty for holding inventory

Numerical Illustration 3: Crowd Trading

Model of crowd trading by Cardaliaguet & Lehalle [CL18]:

$$\begin{cases} dS_t^{\bar{v}} = \gamma \bar{v}_t dt + \sigma dW_t & \text{(price)} \\ dQ_t^v = v_t dt & \text{(player's inventory)} \\ dX_t^{v, \bar{v}} = -v_t(S_t^{\bar{v}} + \kappa v_t) dt & \text{(player's wealth)} \end{cases}$$

Objective: given $(\bar{v}_t)_t$, maximize

$$\mathbb{E} \left[X_T^{v, \bar{v}} + Q_T^v S_T^{\bar{v}} - A |Q_T^v|^2 - \phi \int_0^T |Q_t^v|^2 dt \right]$$

where: $\phi, A > 0 \Rightarrow$ penalty for holding inventory

Ansatz (Cartea & Jaimungal [CJ16]): $V(t, x, s, q) = x + qsu(t, q)$, $\hat{v}_t(q) = \frac{\partial_q u(t, q)}{2\kappa}$

where $u(\cdot)$ solves

$$-\gamma \bar{v} q = \partial_t u - \phi q^2 + \sup_v \{v \partial_q u - \kappa v^2\}, \quad u(T, q) = -Aq^2$$

Numerical Illustration 3: Crowd Trading

Model of crowd trading by Cardaliaguet & Lehalle [CL18]:

$$\begin{cases} dS_t^{\bar{v}} = \gamma \bar{v}_t dt + \sigma dW_t & \text{(price)} \\ dQ_t^v = v_t dt & \text{(player's inventory)} \\ dX_t^{v, \bar{v}} = -v_t(S_t^{\bar{v}} + \kappa v_t) dt & \text{(player's wealth)} \end{cases}$$

Objective: given $(\bar{v}_t)_t$, maximize

$$\mathbb{E} \left[X_T^{v, \bar{v}} + Q_T^v S_T^{\bar{v}} - A |Q_T^v|^2 - \phi \int_0^T |Q_t^v|^2 dt \right]$$

where: $\phi, A > 0 \Rightarrow$ penalty for holding inventory

Ansatz (Cartea & Jaimungal [CJ16]): $V(t, x, s, q) = x + qsu(t, q)$, $\hat{v}_t(q) = \frac{\partial_q u(t, q)}{2\kappa}$
where $u(\cdot)$ solves

$$-\gamma \bar{v} q = \partial_t u - \phi q^2 + \sup_v \{ v \partial_q u - \kappa v^2 \}, \quad u(T, q) = -Aq^2$$

Mean field term: at equilibrium

$$\bar{v}_t = \int \hat{v}_t(q) \hat{m}(t, dq) = \int \frac{\partial_q \hat{u}(t, q)}{2\kappa} \hat{m}(t, dq),$$

where \hat{m} solves the KFP equation:

$$m(0, \cdot) = m_0, \quad \partial_t m + \partial_q \left(m \frac{\partial_q \hat{u}(t, q)}{2\kappa} \right) = 0$$

Reduced forward-backward PDE system:

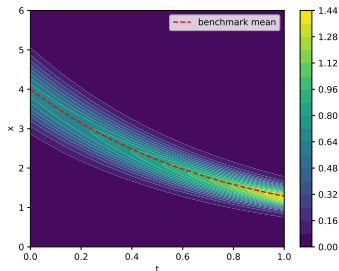
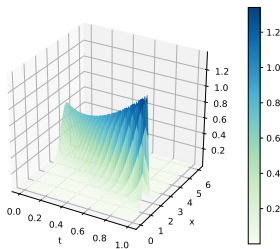
$$\left\{ \begin{array}{l} 0 = -\partial_t u(t, q) + \phi q^2 - \frac{|\partial_q u(t, q)|^2}{4\kappa} = \gamma \bar{\nu}_t q \\ 0 = \partial_t m(t, q) + \partial_q \left(m(t, q) \frac{\partial_q u(t, q)}{2\kappa} \right) \\ \bar{\nu}_t = \int \frac{\partial_q u(t, q)}{2\kappa} m(t, q) dq \\ m(0, \cdot) = m_0, u(T, q) = -Aq^2. \end{array} \right.$$

Numerical Illustration 3: Crowd Trading

Numerical results by DGM versus ODE solution

(More details in [Carmona & L.'21 \[CL21\]](#)²; see also [Al-Aradi *et al.* \[AACN⁺19\]](#)³):

Evolution of m :



²Carmona, R., & Laurière, M. (2021). Deep Learning for Mean Field Games and Mean Field Control with Applications to Finance. arXiv preprint arXiv:2107.04568.

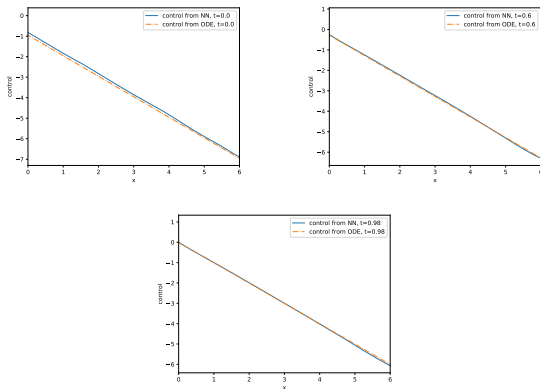
³Al-Aradi, A., Correia, A., Naiff, D., Jardim, G., & Saporito, Y. (2018). Solving nonlinear and high-dimensional partial differential equations via deep learning. arXiv preprint arXiv:1811.08782.

Numerical Illustration 3: Crowd Trading

Numerical results by DGM versus ODE solution

(More details in [Carmona & L.'21 \[CL21\]](#)²; see also [Al-Aradi *et al.* \[AACN⁺19\]](#)³):

Evolution of equilibrium control \hat{v} :



²Carmona, R., & Laurière, M. (2021). Deep Learning for Mean Field Games and Mean Field Control with Applications to Finance. arXiv preprint arXiv:2107.04568.

³Al-Aradi, A., Correia, A., Naiff, D., Jardim, G., & Saporito, Y. (2018). Solving nonlinear and high-dimensional partial differential equations via deep learning. arXiv preprint arXiv:1811.08782.

1. Deep Galerkin Method for MFG PDEs

- Warm-up: ODE
- Solving MFG PDE system
- **Link with Generative Adversarial Networks**

2. Master Equation

Examples



Examples



thispersondoesnotexist.com



thiscatdoesnotexist.com

Examples



thispersondoesnotexist.com



thiscatdoesnotexist.com

Karras et al. [KLA⁺20]: Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8110-8119).

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$

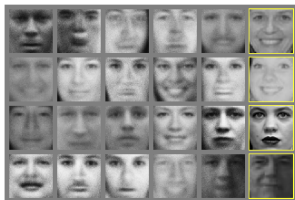
Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$



Goodfellow *et al.* [GPAM⁺14]

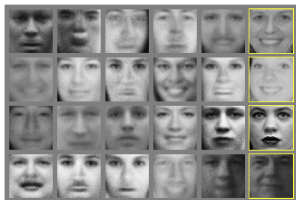
Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$



Goodfellow *et al.* [GPAM⁺14]



NVIDIA'19

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$

Idea++: also learn $D : \mathcal{S} \rightarrow \mathbb{R}$ to distinguish between samples from $p_z \circ G^{-1}$ and p_{data}

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$

Idea++: also learn $D : \mathcal{S} \rightarrow \mathbb{R}$ to distinguish between samples from $p_z \circ G^{-1}$ and p_{data}

Mathematically: min-max game between two neural networks D_δ, G_γ (params: δ, γ)

$$\min_{\gamma} \max_{\delta} \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D_\delta(x)] + \mathbb{E}_{z \sim \mathbb{P}_z} [\log(1 - D_\delta(G_\gamma(z)))] \right\}.$$

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$

Idea++: also learn $D : \mathcal{S} \rightarrow \mathbb{R}$ to distinguish between samples from $p_z \circ G^{-1}$ and p_{data}

Mathematically: min-max game between two neural networks D_δ, G_γ (params: δ, γ)

$$\min_{\gamma} \max_{\delta} \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D_\delta(x)] + \mathbb{E}_{z \sim \mathbb{P}_z} [\log(1 - D_\delta(G_\gamma(z)))] \right\}.$$

Variational MFG: $\inf_{u: [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}} \sup_{m: [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}} \Phi(m, u)$, where

$$\Phi(m, u) = \int_0^T \int_{\mathbb{T}^d} [m(-\partial_t u - \epsilon \Delta_x u) + m H(x, \nabla_x u, m)] dx dt + \int_{\mathbb{T}^d} [m(T)u(T) - m_0 u(0)] dx$$

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$

Idea++: also learn $D : \mathcal{S} \rightarrow \mathbb{R}$ to distinguish between samples from $p_z \circ G^{-1}$ and p_{data}

Mathematically: min-max game between two neural networks D_δ, G_γ (params: δ, γ)

$$\min_{\gamma} \max_{\delta} \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D_\delta(x)] + \mathbb{E}_{z \sim \mathbb{P}_z} [\log(1 - D_\delta(G_\gamma(z)))] \right\}.$$

Variational MFG: $\inf_{u: [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}} \sup_{m: [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}} \Phi(m, u)$, where

$$\Phi(m, u) = \int_0^T \int_{\mathbb{T}^d} [m(-\partial_t u - \epsilon \Delta_x u) + m H(x, \nabla_x u, m)] dx dt + \int_{\mathbb{T}^d} [m(T)u(T) - m_0 u(0)] dx$$

→ Conceptual connection GANs/MFGs: Cao, Guo, L. [CGL20]

Generative Adversarial Nets Goodfellow *et al.* [GPAM⁺14]:

Setup: data space \mathcal{S} (e.g. images of fixed size); *unknown* data distribution p_{data}

Goal: be able to generate samples according p_{data}

Given: samples from data, and random noise generator p_z over some space \mathcal{Z}

Idea: learn $G : \mathcal{Z} \rightarrow \mathcal{S}$ such that $p_z \circ G^{-1} \approx p_{data}$

Idea++: also learn $D : \mathcal{S} \rightarrow \mathbb{R}$ to distinguish between samples from $p_z \circ G^{-1}$ and p_{data}

Mathematically: min-max game between two neural networks D_δ, G_γ (params: δ, γ)

$$\min_{\gamma} \max_{\delta} \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D_\delta(x)] + \mathbb{E}_{z \sim \mathbb{P}_z} [\log(1 - D_\delta(G_\gamma(z)))] \right\}.$$

Variational MFG: $\inf_{u: [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}} \sup_{m: [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}} \Phi(m, u)$, where

$$\Phi(m, u) = \int_0^T \int_{\mathbb{T}^d} [m(-\partial_t u - \epsilon \Delta_x u) + m H(x, \nabla_x u, m)] dx dt + \int_{\mathbb{T}^d} [m(T)u(T) - m_0 u(0)] dx$$

→ Conceptual connection GANs/MFGs: Cao, Guo, L. [CGL20]

Related work: Domingo-Enrich *et al.* [DEJM⁺20], Onken *et al.* [LFL⁺20], ...

Outline

1. Deep Galerkin Method for MFG PDEs

2. Master Equation

- Master Equation for Finite State MFG
- Master Bellman PDE of MFC

- Reminder: equilibrium: $(u, \mu) = \text{sol.}$ starting with m_0 at $t = 0$
- Idea: express the **value function** of a typical player as $u(t, x) = \mathcal{U}(t, x, \mu_t)$

- Reminder: equilibrium: $(u, \mu) = \text{sol.}$ starting with m_0 at $t = 0$
- Idea: express the **value function** of a typical player as $u(t, x) = \mathcal{U}(t, x, \mu_t)$
- Value function \mathcal{U} : PDE on the Wasserstein space

- Reminder: equilibrium: $(u, \mu) = \text{sol.}$ starting with m_0 at $t = 0$
- Idea: express the **value function** of a typical player as $u(t, x) = \mathcal{U}(t, x, \mu_t)$
- Value function \mathcal{U} : PDE on the Wasserstein space
- Motivations:
 - ▶ Unknown initial distribution μ_0
 - ▶ Macroscopic shocks, **common noise**
- Main upshot: optimal behavior for every population configuration

- Reminder: equilibrium: $(u, \mu) = \text{sol.}$ starting with m_0 at $t = 0$
- Idea: express the **value function** of a typical player as $u(t, x) = \mathcal{U}(t, x, \mu_t)$
- Value function \mathcal{U} : PDE on the Wasserstein space
- Motivations:
 - ▶ Unknown initial distribution μ_0
 - ▶ Macroscopic shocks, **common noise**
- Main upshot: optimal behavior for every population configuration
- Convergence of N -player games, large deviation principles, ...

1. Deep Galerkin Method for MFG PDEs

2. Master Equation

- Master Equation for Finite State MFG
- Master Bellman PDE of MFC

Finite state MFG:

- Finite state space \mathcal{S}
- $\mu \in \Delta^{|\mathcal{S}|}$
- $\dot{\mu}_t = \mu_t Q(\mu_t)$, Q = transition rate matrix

Finite state MFG:

- Finite state space \mathcal{S}
- $\mu \in \Delta^{|\mathcal{S}|}$
- $\dot{\mu}_t = \mu_t Q(\mu_t)$, Q = transition rate matrix

Master PDE for \mathcal{U} :

$$\begin{cases} \mathcal{U}(T, x, \mu) = g(x, \mu) \\ -\partial_t \mathcal{U}(t, x, \mu) = \underbrace{H^*(t, x, \mu, \mathcal{U}(t, \cdot, \mu))}_{\text{Hamiltonian}} + \sum_{x' \in \mathcal{S}} \underbrace{\bar{Q}^*(t, \mu, \mathcal{U}(t, \cdot, \mu))(x')}_{\text{avg transition}} \underbrace{\frac{\partial \mathcal{U}(t, \cdot, \mu)}{\partial \mu(x')}}_{\text{classical deriv.}} \end{cases}$$

for $(t, x, \mu) \in [0, T] \times \mathcal{S} \times \Delta^{|\mathcal{S}|}$

Finite state MFG:

- Finite state space \mathcal{S}
- $\mu \in \Delta^{|\mathcal{S}|}$
- $\dot{\mu}_t = \mu_t Q(\mu_t)$, Q = transition rate matrix

Master PDE for \mathcal{U} :

$$\begin{cases} \mathcal{U}(T, x, \mu) = g(x, \mu) \\ -\partial_t \mathcal{U}(t, x, \mu) = \underbrace{H^*(t, x, \mu, \mathcal{U}(t, \cdot, \mu))}_{\text{Hamiltonian}} + \underbrace{\sum_{x' \in \mathcal{S}} \bar{Q}^*(t, \mu, \mathcal{U}(t, \cdot, \mu))(x')}_{\text{avg transition}} \underbrace{\frac{\partial \mathcal{U}(t, \cdot, \mu)}{\partial \mu(x')}}_{\text{classical deriv.}} \end{cases}$$

for $(t, x, \mu) \in [0, T] \times \mathcal{S} \times \Delta^{|\mathcal{S}|}$

Numerical solution?

Example: Cyber-Security Model

Cyber-security model (see [Kolokoltsov & Bensoussan \[KB16\]](#))

- **State space:** $\mathcal{S} = \{DI, DS, UI, US\}$
 - ▶ defended/unprotected
 - ▶ infected/susceptible
- **Actions:**
 - ▶ $\alpha = 1$ (want to switch level of protection)
 - ▶ or 0 (happy)
 - ▶ in each case: event happens at rate λ
- **Time:** continuous time, finite time horizon T

Example: Cyber-Security Model

Cyber-security model (see [Kolokoltsov & Bensoussan \[KB16\]](#))

- **State space:** $\mathcal{S} = \{DI, DS, UI, US\}$
 - ▶ defended/unprotected
 - ▶ infected/susceptible
- **Actions:**
 - ▶ $\alpha = 1$ (want to switch level of protection)
 - ▶ or 0 (happy)
 - ▶ in each case: event happens at rate λ
- **Time:** continuous time, finite time horizon T
- **Mean field interactions:** more infected units \Rightarrow higher infection rate

Example: Cyber-Security Model

Cyber-security model (see [Kolokoltsov & Bensoussan \[KB16\]](#))

- **State space:** $\mathcal{S} = \{DI, DS, UI, US\}$
 - ▶ defended/unprotected
 - ▶ infected/susceptible
- **Actions:**
 - ▶ $\alpha = 1$ (want to switch level of protection)
 - ▶ or 0 (happy)
 - ▶ in each case: event happens at rate λ
- **Time:** continuous time, finite time horizon T
- **Mean field interactions:** more infected units \Rightarrow higher infection rate

$$\dot{\mu}(t) = \mu(t) \underbrace{\begin{pmatrix} \dots & q_{\text{rec}}^D & \alpha\lambda & 0 \\ q_{\text{inf}}^D + \beta_D(\mu_{DI}(t) + \mu_{UI}(t)) & \dots & 0 & \alpha\lambda \\ \alpha\lambda & 0 & \dots & q_{\text{rec}}^U \\ 0 & \alpha\lambda & q_{\text{inf}}^U + \beta_U(\mu_{UI}(t) + \mu_{DI}(t)) & \dots \end{pmatrix}}_{\text{transition rates}}$$

Example: Cyber-Security Model

Cyber-security model (see [Kolokoltsov & Bensoussan \[KB16\]](#))

- **State space:** $\mathcal{S} = \{DI, DS, UI, US\}$
 - ▶ defended/unprotected
 - ▶ infected/susceptible
- **Actions:**
 - ▶ $\alpha = 1$ (want to switch level of protection)
 - ▶ or 0 (happy)
 - ▶ in each case: event happens at rate λ
- **Time:** continuous time, finite time horizon T
- **Mean field interactions:** more infected units \Rightarrow higher infection rate

$$\dot{\mu}(t) = \mu(t) \underbrace{\begin{pmatrix} \dots & q_{\text{rec}}^D & \alpha\lambda & 0 \\ q_{\text{inf}}^D + \beta_D(\mu_{DI}(t) + \mu_{UI}(t)) & \dots & 0 & \alpha\lambda \\ \alpha\lambda & 0 & \dots & q_{\text{rec}}^U \\ 0 & \alpha\lambda & q_{\text{inf}}^U + \beta_U(\mu_{UI}(t) + \mu_{DI}(t)) & \dots \end{pmatrix}}_{\text{transition rates}}$$

- **Running cost:**

$$k_D 1_{\{DI, DS\}} + k_I 1_{\{DI, UI\}} = \text{cost of defense} + \text{penalty for being infected}$$

- **Terminal cost:** 0

Numerical Illustration: DGM for Master Equation

We apply the Deep Galerkin Method (see L.'21 [Lau21])

- Neural network: \mathcal{U}_θ to approximate \mathcal{U}
- Samples: Pick points $(t, x, \mu) \in [0, T] \times \mathcal{S} \times \Delta^{|\mathcal{S}|}$
- Loss: PDE residual + terminal condition

Comparison:

- $\mathcal{U}_\theta(t, x, \mu(t, \cdot))$
- $\mu(t, x), u(t, x)$: finite state space \rightarrow forward-backward ODE system

Numerical Illustration: DGM for Master Equation

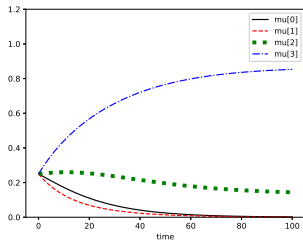
We apply the Deep Galerkin Method (see L.'21 [Lau21])

- Neural network: \mathcal{U}_θ to approximate \mathcal{U}
- Samples: Pick points $(t, x, \mu) \in [0, T] \times \mathcal{S} \times \Delta^{|\mathcal{S}|}$
- Loss: PDE residual + terminal condition

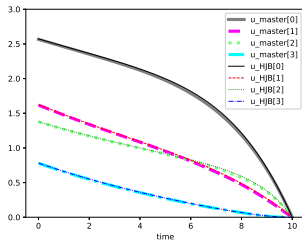
Comparison:

- $\mathcal{U}_\theta(t, x, \mu(t, \cdot))$
- $\mu(t, x), u(t, x)$: finite state space \rightarrow forward-backward ODE system

Test 1: $m_0 = (1/4, 1/4, 1/4, 1/4)$



Evolution of μ



Evolution of u, \mathcal{U}

Numerical Illustration: DGM for Master Equation

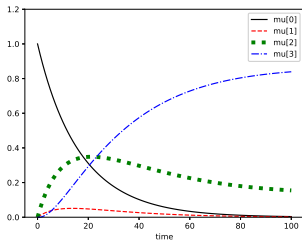
We apply the Deep Galerkin Method (see L.'21 [Lau21])

- Neural network: \mathcal{U}_θ to approximate \mathcal{U}
- Samples: Pick points $(t, x, \mu) \in [0, T] \times \mathcal{S} \times \Delta^{|\mathcal{S}|}$
- Loss: PDE residual + terminal condition

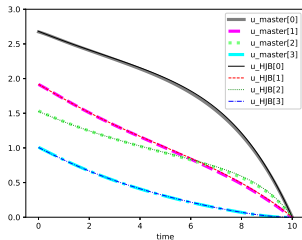
Comparison:

- $\mathcal{U}_\theta(t, x, \mu(t, \cdot))$
- $\mu(t, x), u(t, x)$: finite state space \rightarrow forward-backward ODE system

Test 2: $m_0 = (1, 0, 0, 0)$



Evolution of μ



Evolution of u, \mathcal{U}

Numerical Illustration: DGM for Master Equation

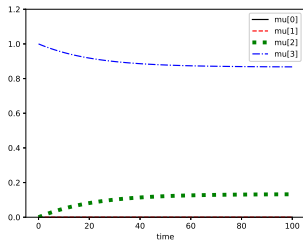
We apply the Deep Galerkin Method (see L.'21 [Lau21])

- Neural network: \mathcal{U}_θ to approximate \mathcal{U}
- Samples: Pick points $(t, x, \mu) \in [0, T] \times \mathcal{S} \times \Delta^{|\mathcal{S}|}$
- Loss: PDE residual + terminal condition

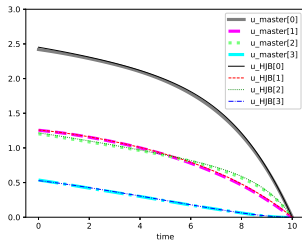
Comparison:

- $\mathcal{U}_\theta(t, x, \mu(t, \cdot))$
- $\mu(t, x), u(t, x)$: finite state space \rightarrow forward-backward ODE system

Test 3: $m_0 = (0, 0, 0, 1)$



Evolution of μ



Evolution of u, \mathcal{U}

1. Deep Galerkin Method for MFG PDEs

2. Master Equation

- Master Equation for Finite State MFG
- Master Bellman PDE of MFC

- **MFC problem** with **common noise**:

$$J^{MFC}(v) = \mathbb{E} \left[\int_0^T f(X_t, \mathbb{P}_{X_t}^0, v_t) dt + g(X_T, \mathbb{P}_{X_T}^0) \right].$$

subj. to: $dX_t = b(X_t, \mathbb{P}_{X_t}^0, v_t)dt + \sigma dW_t + \sigma_0 dW_t^0$,

where $\mathbb{P}_{X_t}^0$ = conditional law of X_t given the **common noise** W^0

- **MFC problem** with **common noise**:

$$J^{MFC}(\boldsymbol{v}) = \mathbb{E} \left[\int_0^T f(X_t, \mathbb{P}_{X_t}^0, \boldsymbol{v}_t) dt + g(X_T, \mathbb{P}_{X_T}^0) \right].$$

subj. to: $dX_t = b(X_t, \mathbb{P}_{X_t}^0, \boldsymbol{v}_t) dt + \sigma dW_t + \sigma_0 dW_t^0$,

where $\mathbb{P}_{X_t}^0$ = conditional law of X_t given the **common noise** W^0

- **Master Bellman equation** in the Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$:

$$\begin{cases} \partial_t V + \mathcal{F}(\boldsymbol{\mu}, V, \partial_{\boldsymbol{\mu}} V, \partial_x \partial_{\boldsymbol{\mu}} V, \partial_{\boldsymbol{\mu}}^2 V) &= 0, & (t, \boldsymbol{\mu}) \in [0, T) \times \mathcal{P}_2(\mathbb{R}^d) \\ V(T, \boldsymbol{\mu}) &= \mathcal{G}(\boldsymbol{\mu}), & \boldsymbol{\mu} \in \mathcal{P}_2(\mathbb{R}^d), \end{cases}$$

where:

- **MFC problem with common noise:**

$$J^{MFC}(v) = \mathbb{E} \left[\int_0^T f(X_t, \mathbb{P}_{X_t}^0, v_t) dt + g(X_T, \mathbb{P}_{X_T}^0) \right].$$

subj. to: $dX_t = b(X_t, \mathbb{P}_{X_t}^0, v_t)dt + \sigma dW_t + \sigma_0 dW_t^0$,

where $\mathbb{P}_{X_t}^0$ = conditional law of X_t given the common noise W^0

- **Master Bellman equation** in the Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$:

$$\begin{cases} \partial_t V + \mathcal{F}(\mu, V, \partial_\mu V, \partial_x \partial_\mu V, \partial_\mu^2 V) &= 0, & (t, \mu) \in [0, T) \times \mathcal{P}_2(\mathbb{R}^d) \\ V(T, \mu) &= \mathcal{G}(\mu), & \mu \in \mathcal{P}_2(\mathbb{R}^d), \end{cases}$$

where:

- ▶ $\partial_\mu V(\mu)(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$, $\partial_x \partial_\mu V(\mu)(\cdot) : \mathbb{R}^d \rightarrow \mathbb{S}^d$, $\partial_\mu^2 V(\mu)(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{S}^d$, are the L -derivatives of V on $\mathcal{P}_2(\mathbb{R}^d)$ (see [Carmona, Delarue'18])
- ▶ and

$$\mathcal{F}(\mu, y, Z(\cdot), \Gamma(\cdot), \Gamma_0(\cdot, \cdot)) = \int_{\mathbb{R}^d} h(x, \mu, Z(x), \Gamma(x)) \mu(dx) + \int_{\mathbb{R}^d \times \mathbb{R}^d} \frac{1}{2} \text{tr} \left(\sigma_0 \sigma_0^\top \Gamma_0(x, x') \right) \mu(dx) \mu(dx'),$$

$$\mathcal{G}(\mu) = \int_{\mathbb{R}^d} g(x, \mu) \mu(dx),$$

$$h(x, \mu, z, \gamma) = \inf_{a \in A} \left[b(x, \mu, a) \cdot z + \frac{1}{2} \text{tr} \left(\sigma \sigma^\top \gamma \right) + f(x, \mu, a) \right].$$

- N agents \rightarrow mean field: $\mu^N = \frac{1}{N} \sum_{i=1}^N \delta_{x^i}$

$$v^N(t, x, x^1, \dots, x^N) = V^N(t, x, \mu^N) \rightarrow V(t, x, \mu^N)$$

- Approximate $V(t, x, \cdot)$ by a **symmetric** function of N inputs (N large)

- N agents \rightarrow mean field: $\mu^N = \frac{1}{N} \sum_{i=1}^N \delta_{x^i}$

$$v^N(t, x, x^1, \dots, x^N) = V^N(t, x, \mu^N) \rightarrow V(t, x, \mu^N)$$

- Approximate $V(t, x, \cdot)$ by a **symmetric** function of N inputs (N large)

- **Symmetric Neural Networks:**

- ▶ Symmetry by construction; e.g. with a sum:

$$(x^i)_{i=1, \dots, N} \mapsto \sum_{i=1}^N \psi_{\omega}(x^i) \mapsto \varphi_{\theta} \left(\sum_{i=1}^N \psi_{\omega}(x^i) \right)$$

- ▶ DeepSets [ZKR⁺17]⁴, PointNet [QSMG17]⁵, ...

⁴Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., & Smola, A. J. (2017, December). Deep Sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 3394-3404).

⁵Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652-660).

Deep Learning for MFC with DPP and Symmetric NN (Germain *et al.* [GLPW21]⁶)

- **Symmetric NN:** $\mathcal{V}(t, x^1, \dots, x^N)$
- **D-Symmetric NN:** sym. except in one space variable:

$$\mathcal{Z}(x^1, \dots, x^N, x^i) \leftrightarrow \partial_{x^i} \mathcal{V}(x^1, \dots, x^N) = \frac{1}{N} \partial_{\mu} \mathcal{V} \left(\frac{1}{N} \sum_j x^j \right) (x^i)$$

Output: $(\widehat{\mathcal{V}}_n, \widehat{\mathcal{Z}}_n)_{n=0, \dots, N_T}$ s.t. $\widehat{\mathcal{V}}_n(\underline{x}) \approx V(t_n, \mu_{\underline{x}}^N)$, $\widehat{\mathcal{Z}}_n(\underline{x}, x^i) \approx \frac{1}{N} \partial_{\mu} V(t_n, \mu_{\underline{x}}^N)(x^i)$

- 1 Set $\widehat{\mathcal{V}}_{N_T}(\cdot) = G(\cdot)$
- 2 **for** $n = N_T - 1, N_T - 2, \dots, 1, 0$ **do**
- 3 Compute $(\widehat{\mathcal{V}}_n, \widehat{\mathcal{Z}}_n)$ as a minimizer of:

$$(\mathcal{V}_n, \mathcal{Z}_n) \mapsto \mathbb{E} \left| \widehat{\mathcal{V}}_{n+1}(\mathbf{X}_{n+1}) - \mathcal{V}_n(\mathbf{X}_n) + H(t_n, \mathbf{X}_n, \mathcal{V}_n(\mathbf{X}_n), \mathbf{Z}_n(\mathbf{X}_n)) \Delta t \right. \\ \left. - \sum_{i=1}^N \sum_{j=0}^N (\mathcal{Z}_n(\mathbf{X}_n, X_n^i))^{\top} \sigma_{ij} \Delta W_n^j \right|^2,$$

where $\widehat{\mathcal{V}}_n$ is a sym. NN, $\widehat{\mathcal{Z}}_n$ is a D-sym. NN, H = sym. version of h

- 4 **return** $(\widehat{\mathcal{V}}_n, \widehat{\mathcal{Z}}_n)_{n=0, \dots, N_T}$
-

⁶ Germain, M., Laurière, M., Pham, H., & Warin, X. (2021). DeepSets and their derivative networks for solving symmetric PDEs. arXiv preprint arXiv:2103.00838.

References I

- [AACN⁺19] Ali Al-Aradi, Adolfo Correia, Danilo de Frietas Naiff, Gabriel Jardim, and Yuri Saporito, *Applications of the deep galerkin method to solving partial integro-differential and hamilton-jacobi-bellman equations*, arXiv preprint arXiv:1912.01455 (2019).
- [ACD10] Yves Achdou and Italo Capuzzo-Dolcetta, *Mean field games: numerical methods*, SIAM J. Numer. Anal. **48** (2010), no. 3, 1136–1162. MR 2679575
- [AFG17] Noha Almula, Rita Ferreira, and Diogo Gomes, *Two numerical approaches to stationary mean-field games*, Dyn. Games Appl. **7** (2017), no. 4, 657–682. MR 3698446
- [BFY13] Alain Bensoussan, Jens Frehse, and Sheung Chi Phillip Yam, *Mean field games and mean field type control theory*, Springer Briefs in Mathematics, Springer, New York, 2013.
- [CGL20] Haoyang Cao, Xin Guo, and Mathieu Laurière, *Connecting gans, mfgs, and ot*, arXiv preprint arXiv:2002.04112 (2020).
- [CJ16] Álvaro Cartea and Sebastian Jaimungal, *Incorporating order-flow into optimal execution*, Mathematics and Financial Economics **10** (2016), no. 3, 339–364.
- [CL18] Pierre Cardaliaguet and Charles-Albert Lehalle, *Mean field game of controls and an application to trade crowding*, Mathematics and Financial Economics **12** (2018), no. 3, 335–363.

References II

- [CL21] René Carmona and Mathieu Laurière, *Deep learning for mean field games and mean field control with applications to finance*, arXiv preprint arXiv:2107.04568 (2021).
- [DEJM⁺20] Carles Domingo-Enrich, Samy Jelassi, Arthur Mensch, Grant Rotskoff, and Joan Bruna, *A mean-field analysis of two-player zero-sum games*, arXiv preprint arXiv:2002.06277 (2020).
- [GLPW21] Maximilien Germain, Mathieu Laurière, Huy  n Pham, and Xavier Warin, *Deepsets and their derivative networks for solving symmetric pdes*, arXiv preprint arXiv:2103.00838 (2021).
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in neural information processing systems **27** (2014).
- [KB16] Vassili N. Kolokoltsov and Alain Bensoussan, *Mean-field-game model for botnet defense in cyber-security*, Appl. Math. Optim. **74** (2016), no. 3, 669–692. MR 3575619
- [KLA⁺20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila, *Analyzing and improving the image quality of stylegan*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8110–8119.

References III

- [Lau21] Mathieu Laurière, *Numerical methods for mean field games and mean field type control*, arXiv preprint arXiv:2106.06231 (2021).
- [LFL⁺20] Alex Tong Lin, Samy Wu Fung, Wuchen Li, Levon Nurbekyan, and Stanley J Osher, *Apac-net: Alternating the population and agent control via two neural networks to solve high-dimensional stochastic mean field games*, arXiv preprint arXiv:2002.10113 (2020).
- [LL07] Jean-Michel Lasry and Pierre-Louis Lions, *Mean field games*, Jpn. J. Math. **2** (2007), no. 1, 229–260. MR 2295621
- [QSMG17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.
- [SS18] Justin Sirignano and Konstantinos Spiliopoulos, *DGM: a deep learning algorithm for solving partial differential equations*, J. Comput. Phys. **375** (2018), 1339–1364. MR 3874585
- [ZKR⁺17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola, *Deep sets*, arXiv preprint arXiv:1703.06114 (2017).

Unless otherwise specified, the images are from <https://unsplash.com>

