Solving Continuous Mean Field Games: Deep Reinforcement Learning for Non-Stationary Dynamics

Lorenzo Magnino[*]
University of Cambridge

Kai Shao[†]
KTH Royal Institute of Technology

Zida Wu
University of California, Los Angeles

Jiacheng Shen
NYU Center for Data Science

Mathieu Laurière
NYU Shanghai

Abstract

Mean field games (MFGs) have emerged as a powerful framework for modeling interactions in large-scale multi-agent systems. Despite recent advancements in reinforcement learning (RL) for MFGs, existing methods are typically limited to finite spaces or stationary models, hindering their applicability to real-world problems. This paper introduces a novel deep reinforcement learning (DRL) algorithm specifically designed for non-stationary continuous MFGs. The proposed approach builds upon a Fictitious Play (FP) methodology, leveraging DRL for best-response computation and supervised learning for average policy representation. Furthermore, it learns a representation of the time-dependent population distribution using a Conditional Normalizing Flow. To validate the effectiveness of our method, we evaluate it on three different examples of increasing complexity. By addressing critical limitations in scalability and density approximation, this work represents a significant advancement in applying DRL techniques to complex MFG problems, bringing the field closer to real-world multi-agent systems.

1 Introduction

Learning in multiplayer games poses significant challenges due to the interplay between strategic decision-making and the dynamics, often non-stationary, interactions among agents. Deep reinforcement learning (DRL) has recently achieved remarkable success in two-player or small-team games such as Go [Silver et al., 2016, 2018], chess [Silver et al., 2017], poker [Heinrich and Silver, 2016], StarCraft [Samvelyan et al., 2019], and Stratego [Perolat et al., 2022]. However, scaling these methods to large populations of agents remains difficult. As the number of agents grows, the joint strategy space becomes prohibitively large, and traditional multi-agent reinforcement learning (MARL) techniques often become computationally intractable; see [Busoniu et al., 2008, Yang and Wang, 2020, Zhang et al., 2021, Gronauer and Diepold, 2022, Wong et al., 2023] for recent reviews.

Mean Field Games (MFGs) [Lasry and Lions, 2007, Huang et al., 2006] offer a principled framework for approximating such large-scale systems by modeling the interaction between a single representative agent and an evolving population distribution. Drawing on tools from statistical physics and optimal control, MFGs reduce the dimensionality of the problem by considering the limiting behavior as the number of agents tends to infinity. At equilibrium, each agent solves a Markov decision process (MDP) given the population distribution, and the distribution itself must evolve consistently with

[*]Work done during period at NYU Shanghai Center for Data Science and the NYU-ECNU Institute of Mathematical Sciences at NYU Shanghai. Contacts: `lm2183@cam.ac.uk`, `kshao@kth.se`, `zdwu@ucla.edu`, `shen.patrick.jiacheng@nyu.edu`, `mathieu.lauriere@nyu.edu`.

[†]Work done during period at NYU Shanghai

the agents' policies. Since individual agents are infinitesimal in the limit, they do not affect the population, allowing each agent to ignore second-order feedback effects.

This approximation is particularly relevant for applications involving large populations and continuous state-action spaces, such as economics [Lachapelle et al., 2016, Achdou et al., 2022], finance [Carmona et al., 2017, Cardaliaguet and Lehalle, 2018, Carmona, 2021], engineering [Djehiche et al., 2017], crowd motion [Lachapelle and Wolfram, 2011, Djehiche et al., 2017, Achdou and Lasry, 2018], flocking and swarming [Fornasier and Solombrino, 2014, Nourian et al., 2010], cloud computing [Hanif et al., 2015, Mao et al., 2022], and telecommunication networks [Yang et al., 2016, Ge et al., 2019]. In many of these domains, both the state dynamics and control policies are naturally continuous, and the population distribution often evolves over time rather than remaining stationary.

While classical numerical solvers for MFGs can handle low-dimensional problems in simple domains [Achdou and Laurière, 2020], they are limited by the curse of dimensionality and do not scale to complex or high-dimensional settings. Deep learning methods have been proposed to solve high-dimensional problems (see e.g. [Hu and Laurière, 2024] for an overview) but these methods generally struggle to solve MFGs in very complex environments. To address these limitations, recent work has turned to model-free reinforcement learning (RL) as promising approaches to solving MFGs [Guo et al., 2019, Subramanian and Mahajan, 2019, Elie et al., 2020, Fu et al., 2019, Cui and Koeppl, 2021, Angiuli et al., 2022, Yardim et al., 2023, Ocello et al., 2024]; see [Laurière et al., 2022b] for a recent survey. However, most existing methods are restricted to **finite state and action spaces** and **stationary population distributions**. Extending RL frameworks to continuous-space, non-stationary MFGs presents significant challenges, especially for learning time-dependent population dynamics and solving the resulting fixed-point problems. In contrast to MDPs, where the goal is to optimize a single agent's trajectory, solving MFGs requires learning both an optimal response and a consistent population evolution. **To the best of our knowledge, no existing RL algorithms are capable of learning the solution of non-stationary MFGs with continuous state and action spaces**.

Contributions. This paper introduces Density-Enhanced Deep-Average Fictitious Play (DEDA-FP) (see Figure 1), a novel deep reinforcement learning (DRL) algorithm for **non-stationary** Mean Field Games (MFGs) with **continuous** state and action spaces. Our approach extends **Fictitious Play (FP)**, a classical game-theoretic learning scheme that iteratively updates each agent's policy to optimally respond to the evolving population behavior.

To address the challenge of averaging neural policies, we use **DRL** (Soft Actor-Critic and Proximal Policy Optimization) to compute approximate best responses and **supervised learning** to represent the averaged policy across FP iterations. This hybrid strategy ensures scalability and accurate policy approximation.

We also train a time-dependent Conditional Normalizing Flow (CNF) to model the non-stationary evolution of the **population distribution**, enabling **sampling** from the equilibrium mean field and **density estimation**. This model accurately captures MFGs with local dependence on population density, unlike empirical distributions, and **improves sampling time efficiency tenfold** compared to our benchmarks.

We validate our method with three experiments of increasing complexity and provide an error propagation analysis (Theorem 1). Our contributions address key challenges in applying RL to MFGs, including **time-dependence**, **continuous spaces**, and **local density effects**, representing a significant step toward scalable, model-free solutions for real-world multi-agent systems.

1.1 Related Work

The following works are particularly relevant to our context and help clarify our methodological contributions (see Table 1 for a summary). [Perrin et al., 2021] develop a DRL algorithm based on **Fictitious Play (FP)** for continuous state and action spaces. However, their focus is restricted to **stationary MFGs** (i.e., time-invariant mean fields), and their method does *not* learn the Nash equilibrium policy. Instead, it learns a collection of best responses from which a player may sample to imitate the average behavior. [Laurière et al., 2022a] propose two DRL algorithms and in particular a variant of FP which does learn the **Nash equilibrium policy**. While we draw inspiration from their approach to represent the average policy, their method is limited to **discrete state and action spaces**.

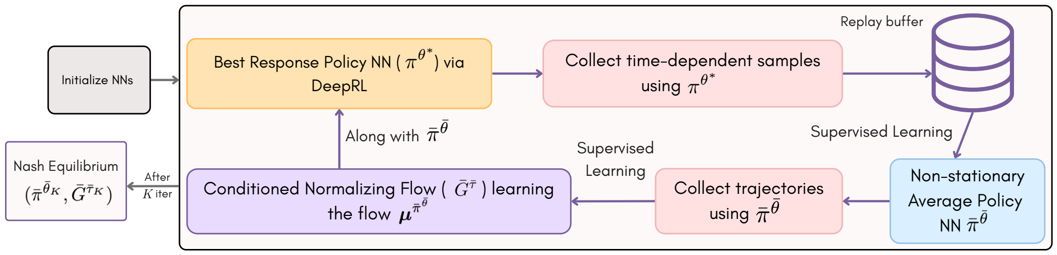

Initialize NNs
Best Response Policy NN (π^{θ^*}) via DeepRL
Collect time-dependent samples using π^{θ^*}
Replay buffer
Supervised Learning
Non-stationary Average Policy NN $\bar{\pi}^{\bar{\theta}}$
Collect trajectories using $\bar{\pi}^{\bar{\theta}}$
Supervised Learning
Conditioned Normalizing Flow ($\bar{G}^{\tau}$) learning the flow $\boldsymbol{\mu}^{\bar{\pi}^{\bar{\theta}}}$
Along with $\bar{\pi}^{\bar{\theta}}$
After K iter
Nash Equilibrium ($\bar{\pi}^{\bar{\theta}_K}, \bar{G}^{\bar{\tau}_K}$)

Figure 1: Overview of our **DEDA-FP** model. Our framework uses three main steps, built upon the Fictitious Play algorithm, to fully solve the MFG problem (details in Section 3): (1) computation of the best response using **Deep RL algorithms**; (2) learning a **policy neural network** to approximate the average policy over past policies; and (3) learning a **Time-Conditioned Normalizing Flow** to approximate the average distribution over past mean-field flows.

[Zaman et al., 2020] tackle **non-stationary MFGs** using actor-critic methods in a discrete-time **linear-quadratic (LQ)** setting. Although their model operates in continuous state and action spaces, it is confined to the LQ regime, where optimal policies are deterministic and linear. [Angiuli et al., 2023] study DRL algorithms for MFGs with continuous states and actions, and include a generative model for the population distribution. However, they only address **stationary LQ problems**, and their generative model can produce samples from the mean field but *cannot* estimate the **density at a given location**, making it unsuitable for models with local mean field dependence.

Method	Cont. space	General r, P	NE policy	Local. dep.	Non-stat.
DEDA-FP	✓	✓	✓	✓	✓
Zaman et al. [2020]	✓	✗	✓	✗	✓
Perrin et al. [2021]	✓	✓	✗	✓	✗
Laurière et al. [2022a]	✗	✓	✓	✓	✓
Angiuli et al. [2023]	✓	✗	✓	✗	✗

Table 1: Comparison between our approach and related works. Our approach is the first to learn the Nash equilibrium policy and distribution for continuous space non-stationary MFGs with general dynamics and rewards, including possibly local dependence on the mean field.

2 Non-stationary continuous MFGs

Notations. Let $\mathcal{X}$ and $\mathcal{A}$ be respectively the state and action spaces, which can be continuous. To fix the ideas, we will take $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{A} = \mathbb{R}^k$, where d and k are the respective dimensions. Let $\mathcal{P}(\mathcal{X})$ and $\mathcal{P}(\mathcal{A})$ denote the sets of probability distributions on $\mathcal{X}$ and $\mathcal{A}$, respectively. Let N_T be the number of time steps. We will use bold symbols for sequences, i.e., functions of time. The state distribution of the population at time t is called the **mean field** and will be denoted by $\mu_t \in \mathcal{P}(\mathcal{X})$. We denote by μ_0 the initial distribution, which is assumed to be fixed and known from the players.

Dynamics. To define the mean field game, we first need to define the dynamics of the state x_t of a representative player when the mean field sequence $\boldsymbol{\mu} = (\mu_t)_{t=0,\dots,N_T}$ is given. We consider a general dynamics: if the agent is in state x_t, takes action a_t and the mean field is currently μ_t, then the next state is sampled according to:

$$x_{t+1} \sim P_t(\cdot|x_t, a_t, \mu_t), \tag{1}$$

where $P : \{0, \dots, N_T\} \times \mathcal{X} \times \mathcal{A} \times \mathcal{P}(\mathcal{X}) \to \mathcal{P}(\mathcal{X})$ is the *transition kernel*. A typical setting is when the transitions are given by a transition function, namely, $x_{t+1} = F_t(x_t, a_t, \mu_t, \epsilon_t)$, where $F : \{0, \dots, N_T\} \times \mathcal{X} \times \mathcal{A} \times \mathcal{P}(\mathcal{X}) \to \mathcal{X}$ is the *transition function* and $(\epsilon_t)_{t\geq 0}$ is a sequence of i.i.d. noises. A typical example is the time-discretization of a continuous time stochastic differential equation (SDE): $\mathcal{X} = \mathbb{R}^d, \mathcal{A} = \mathbb{R}^k$ and $F_t(x_t, a_t, \mu_t, \epsilon_t) = x_t + b_t(x_t, a_t, \mu_t) + \sigma\epsilon_t$, where $b : \{0, \dots, N_T\} \times \mathcal{X} \times \mathcal{A} \times \mathcal{P}(\mathcal{X}) \to \mathcal{X}$ is the *drift*, and σ is the *volatility*. This setting is particularly relevant because most of the MFG literature focuses on SDE-type dynamics, see e.g. [Carmona and Delarue, 2018]. But we stress that here, only time is discretized; space is continuous, in contrast with most of the literature on RL for MFGs, see e.g. [Laurière et al., 2022b]. From the dynamics of the individual player, we can deduce a dynamics for the whole population, i.e., a transition from μ_t to μ_{t+1}, as explained below after introducing the notion of policy.

Policies. A policy $\boldsymbol{\pi}$ for an individual player is a function from $\{0, \ldots, N_T\} \times \mathcal{X}$ to $\mathcal{P}(\mathcal{A})$ and $\pi_t(\cdot|x) = \pi(\cdot|t, x)$ is the distribution used to pick the next action when time is t and the player's state is x. As is common in the MFG literature [Guo et al., 2019, Elie et al., 2020, Cui and Koeppl, 2021, Guo et al., 2023], we consider here decentralized policies, which are functions of the individual state and not of the population distribution. This is because, at equilibrium, the mean field is completely determined by the policy and the initial population distribution, which is assumed to be known. If the whole population uses the same policy $\boldsymbol{\pi}$, this induces a mean field flow $\boldsymbol{\mu}^{\boldsymbol{\pi}} = (\mu_t^{\boldsymbol{\pi}})_{t\geq 0}$, which is determined by the evolution: $\mu_0^{\boldsymbol{\pi}} = \mu_0$, and for $t = 0, \ldots, N_T - 1$,

$$\mu_{t+1}^{\boldsymbol{\pi}}(x') = \int_{\mathcal{X}\times\mathcal{A}} \mu_t^{\boldsymbol{\pi}}(x)\pi_t(a|x)P(x'|x, a, \mu_t^{\boldsymbol{\pi}})dxda, \quad x' \in \mathcal{X}. \tag{2}$$

Rewards. The reward is a function $r : \mathcal{X} \times \mathcal{A} \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$. When the mean field flow is given by $\boldsymbol{\mu} = (\mu_t)_{t\geq 0}$, the representative agent aims to find a policy $\boldsymbol{\pi} = (\pi_t)_{t\geq 0}$ that maximizes the total expected reward: with $(\mu_t^{\boldsymbol{\pi}})_{t\geq 0}$ is the distribution flow induced by $\boldsymbol{\pi}$, see (2),

$$J_{\boldsymbol{\mu}}(\boldsymbol{\pi}) = \mathbb{E}_{x_t,a_t}\Big[\sum_{t=0}^{N_T} r(x_t, a_t, \mu_t)\Big] = \sum_{t=0}^{N_T} \int_{\mathcal{X}\times\mathcal{A}} \mu_t^{\boldsymbol{\pi}}(x)\pi_t(a|x)r(x, a, \mu_t)dxda.$$

We will use the notations $J_{\boldsymbol{\mu}}(\boldsymbol{\pi})$ and $J(\boldsymbol{\pi}, \boldsymbol{\mu})$ interchangeably.

Nash equilibrium. We focus on the notion of Nash equilibrium, in which no player has any incentives to deviate unilaterally. It is defined as follows in the mean field setting.

Definition 1. *A* **mean-field Nash equilibrium (MFNE)** *is a pair* $(\boldsymbol{\mu}^*, \boldsymbol{\pi}^*) = (\mu_t, \pi_t)_{t\geq 0}$ *of a sequence of population distributions and policies that satisfies:*

1. $\boldsymbol{\pi}^*$ *maximizes* $\boldsymbol{\pi} \mapsto J_{\boldsymbol{\mu}^*}(\boldsymbol{\pi})$*;*
2. *For every* $t \geq 0$, μ_t^* *is the distribution of* x_t *given by dynamics* (1) *with* $(\boldsymbol{\pi}^*, \boldsymbol{\mu}^*)$.

It can be expressed more concisely using the notion of exploitability, which quantifies how much a player can be better of by deviating from the policy used by the rest of the players.

Definition 2. *The* **exploitability** *of a policy* $\boldsymbol{\pi}$ *is defined as:* $\mathcal{E}(\boldsymbol{\pi}) = \max_{\boldsymbol{\pi}'} J_{\boldsymbol{\mu}^{\boldsymbol{\pi}}}(\boldsymbol{\pi}') - J_{\boldsymbol{\mu}^{\boldsymbol{\pi}}}(\boldsymbol{\pi})$.

Then, $(\boldsymbol{\pi}, \boldsymbol{\mu}^{\boldsymbol{\pi}})$ is Nash equilibrium if and only if $\mathcal{E}(\boldsymbol{\pi}) = 0$. The notion of exploitability has been used in several works to assess the convergence of RL algorithms, see e.g. [Perrin et al., 2020].

We discuss our assumption in App. A. Continuous space implies that the mean field, in $\mathcal{P}(\mathcal{X})$, is a priori infinite-dimensional. Furthermore, the non-stationarity of the model ensures that the policy should be time-dependent, departing from most of the RL literature. Another difficulty is that we want to compute the equilibrium policy, which is in general, a mixed policy, not always learned correctly by algorithms such as FP. Although ad hoc methods have been proposed for linear dynamics and quadratic rewards, an RL algorithm for general models is still lacking.

3 Solving Continuous Mean Field Games

Solving a mean field game necessitates finding the flow $(\boldsymbol{\mu}^*, \boldsymbol{\pi}^*) = (\mu_t, \pi_t)_{t\geq 0}$ that satisfies Def. 1. To achieve this, we employ a version of **FP** algorithm. FP was originally introduced in [Brown, 1951] and adapted to MFGs in [Cardaliaguet and Hadikhanloo, 2017] and [Perrin et al., 2020] respectively in continuous time and discrete time MFGs. Our approach relies on computing the best response against the weighted average of previous distributions and updated the distribution consequently (see details in Appendix C). **However, in continuous action spaces, calculating the best response, learning the average policy, and determining the average distribution present significant challenges due to the problem's infinite dimensionality.** To illustrate the effectiveness of our approach, we proceed with a *step-by-step construction*: first, we present a basic algorithm (Algo. 1), followed by a demonstration of the benefits of learning the equilibrium policy (Algo. 2). Finally, we detail our approach, named **DEDA-FP** (Algo. 3), which overcomes the limitations highlighted by these preceding methods.

Algo. 1: Simple Approach. As a first method, we implement a simple version of FP, in the spirit of Perrin et al. [2021]. At each iteration of FP, the agent learns the best response against the approximated average population distribution using DRL algorithms such as Soft Actor-Critic (SAC) or Proximal Policy Optimization (PPO). Subsequently, this policy is added to a behavioral buffer. The population state at iteration k is approximated by simulating $N-1$ trajectories, each generated by one of $N-1$ policies sampled uniformly from the buffer. Note that this constitutes a two-level approximation: first, we approximate the average policy by sampling from the buffer, and then we mimic the population distribution by sampling trajectories. Details are provided in Appendix C. **However, at the conclusion of this algorithm, we do not obtain either the Nash equilibrium policy or the mean field distribution.**

Algo. 2: Learning the Nash equilibrium policy. To address the main limitation of the simple approach, we train a policy network to learn the average policy. To this end, we draw inspiration from [Heinrich and Silver, 2016, Laurière et al., 2022a] and use supervised learning with a suitably defined replay buffer. We present here the main ideas and the details are provided in Appendix C. Concretely, at every iteration we collect samples from the best response and we train a neural network (NN) to minimize the Negative Log-Likelihood:

$$\mathcal{L}_{\text{NLL}}(\theta) = \mathbb{E}_{(t,s,a)\sim\mathcal{M}_{SL}}\left[-\log \pi^{\theta}(a|t,s)\right] = -\frac{1}{M}\sum_{i=1}^{M} \log \mathcal{N}(a_i; \mu_\theta(s_i,t_i), \sigma_\theta(s_i,t_i))$$

where $\mathcal{N}(\cdot)$ is the Gaussian probability density function, M is the size of the replay buffer $\mathcal{M}_{SL}$ containing all the triples (t,s,a) sampled from the previous policies and μ_θ and σ_θ are the mean and standard deviation predicted by the policy network for the state s_i at time t_i. Furthermore, this approach allows us, at iteration k, to compute the best response against $N-1$ agents using the learned average policy, thereby avoiding the need to sample uniformly from the behavioral buffer. Further details are provided in Appendix C.

Remaining Challenges. At this point, the algorithm can learn the **equilibrium policy**, which is one part of the solution to the MFGs (see Def. 1). However, the other part, namely the **equilibrium mean field**, is still lacking. Most existing works then approximate the optimal mean field distribution by sampling a large number of trajectories to adequately cover the state space. However, as we will elaborate in Sec. 5, there are several key limitations to this approach:

1. Many mean field games derive their complexity and richness from **local interactions**, where the dynamics or rewards depend on the population density (e.g., congestion, entropy maximization). Without a direct model for the mean field distribution, the density must be estimated indirectly (e.g., via Gaussian convolution), which can alter the nature of the problem.

2. In the **evaluation** or rollout phase, estimating the mean field and its density requires sampling many trajectories at each step. This becomes computationally expensive, especially in state spaces of dimension $d \geq 2$, and can significantly slow down execution.

For the reasons mentioned above, we propose a novel algorithm, **Density-Enhanced Deep Average Fictitious Play** solver (DEDA-FP), that fully solves MFGs by learning both the Nash equilibrium policy and the mean field distribution, enabling us to both sample from it and the compute its density.

3.1 DEDA-FP

Our method incorporates best response computation using deep reinforcement learning and learns the average policy with supervised learning, as depicted in the previous scheme. Furthermore, it completely solves the mean field game problem by learning the Nash equilibrium mean field distribution. Inspired by and extending the approach of Perrin et al. [2021], we utilize a time-conditioned generative model to learn the mean field flow $\boldsymbol{\mu}^{\boldsymbol{\pi}} = (\mu_t^{\boldsymbol{\pi}})_{t\geq 0}$. Specifically, we employ a Conditional Normalizing Flow [Winkler et al., 2019, Rezende and Mohamed, 2015, Kobyzev et al., 2020] which is a particular generative model that learns a complex probability distribution $p(\mathbf{x}|t)$ conditioned on a time variable $t \in [0,T]$. It achieves this by transforming a simple base distribution $p_0(\mathbf{z})$ (e.g., a standard Gaussian) into the target distribution $p(\mathbf{x}|t)$ through a sequence of invertible transformations $f_1, f_2, \ldots, f_K$, where each transformation is conditioned on the time t. The probability density of a sample $\mathbf{x}$ from the target distribution $p(\mathbf{x}|t)$ is computed as:

$$p(\mathbf{x}|t) = p_0(f^{-1}(\mathbf{x},t))\left|\det\left(\frac{\partial f^{-1}(\mathbf{x},t)}{\partial \mathbf{x}}\right)\right|,$$

where $f^{-1}(\mathbf{x}, t) = f_K^{-1}(f_{K-1}^{-1}(\dots f_1^{-1}(\mathbf{x}, t)\dots))$ is the inverse of the entire flow, and $\det\left(\frac{\partial f^{-1}(\mathbf{x},t)}{\partial \mathbf{x}}\right)$ is the determinant of the Jacobian of the inverse transformation.

The conditioning on time t is typically incorporated into the parameters of the transformation functions f_k. For example, if f_k is an affine transformation, its parameters (e.g., the scaling and translation) would be functions of t. Similarly, for more complex flow architectures like neural spline flows, the parameters of the splines would be conditioned on t (see Sec. 5 for more details about the architecture we used).

To learn the parameters of the Conditional Normalizing Flow, we employ Maximum Likelihood Estimation (MLE). This is equivalent to minimizing the Negative Log-Likelihood (NLL) of the observed data. Given a dataset of N samples $\{\mathbf{x}_i, t_i\}_{i=1}^N$ drawn from the time-dependent distribution, the NLL loss function is given by:

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{N}\sum_{i=1}^{N}\left[\log p_0(f^{-1}(\mathbf{x}_i, t_i)) + \log\left|\det\left(\frac{\partial f^{-1}(\mathbf{x}_i, t_i)}{\partial \mathbf{x}_i}\right)\right|\right].$$

By learning the parameters of these time-conditioned transformations using maximum likelihood estimation on a dataset of time-dependent distributions, the model learns to generate samples from and estimate the probability density of $p(\mathbf{x}|t)$ for any $t \in [0, T]$. Algo. 3 summarizes our method. For notations and further discussion on the choices of the individual components refer to App. C.

Algo. 3 Density-Enhanced Deep Average Fictitious Play (**DEDA-FP**)

```
1: Input: $\mu_0$: initial distribution; $N_{sa}$: number of state-action pairs to collect at every iteration, $N$:
   population size in population simulation; $K$: number of iterations.
2: Initialize: $(\theta_0^* = \bar{\theta}_0, \bar{\tau}_0)$ at random; empty replay buffer $\mathcal{M}_{SL}$ for supervised learning of average
   policy; using $\boldsymbol{\pi}_0^* := \boldsymbol{\pi}^{\theta_0^*}$, sample $N_{sa}$ triples $(0, s, a)$ and store them in $\mathcal{M}_{SL}$.
3: for iteration $k = 1$ to $K$ do
4:     Find the best response $\boldsymbol{\pi}_k^* := \boldsymbol{\pi}^{\theta_k^*}$ vs the $N-1$ agents using $\bar{\boldsymbol{\pi}}_{k-1} := \bar{\boldsymbol{\pi}}^{\bar{\theta}_{k-1}}$ using Deep RL:
           $\boldsymbol{\pi}_k^* = \arg\max_{\boldsymbol{\pi}} J_{\mu_0}^N(\boldsymbol{\pi}, \bar{\mathbf{G}}_{k-1})$
5:     Collect $N_{sa}$ time-state-action samples of the form $(t, s, a)$ using $\boldsymbol{\pi}_k^*$ and store in $\mathcal{M}_{SL}$.
6:     Train the NN policy $\bar{\boldsymbol{\pi}}_k := \bar{\boldsymbol{\pi}}^{\bar{\theta}_k}$ using supervised learning to minimize the categorical loss:
           $\mathcal{L}_{\text{NLL}}(\bar{\theta}) = \mathbb{E}_{(t,s,a)\sim\mathcal{M}_{SL}}\left[-\log \bar{\boldsymbol{\pi}}^{\bar{\theta}}(a|t, s)\right]$
7:     Train a Conditional Normalizing Flow $\bar{\mathbf{G}}_k := \bar{\mathbf{G}}^{\bar{\tau}_k}$ for the time-dependent mean field $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}$
       using trajectories generated by $\bar{\boldsymbol{\pi}}_k$ and initialization $\bar{\mathbf{G}}_{k-1}$.
8: end for
9: return $\bar{\boldsymbol{\pi}}_K$, $\bar{\mathbf{G}}_K$
```

4 On the convergence of DEDA-FP

We analyze the convergence of the DEDA-FP algorithm by extending the theoretical framework developed by [Elie et al., 2020]. Our goal is to show that the exploitability of the learned policy converges towards a value determined by the sum of three specific accumulated errors, thus establishing convergence to an ϵ-Nash Equilibrium.

We first formalize the **error sources** based on distance d_{N_T} between sequences of mean fields and distance d_Π between policies (see App. B for details).

1. **Best Response Error.** The sub-optimality of the DRL policy $\boldsymbol{\pi}_k^*$ wrt the mean-field flow $\bar{\mathbf{G}}_K$ from the previous iteration: $\epsilon_{br}^k := J(\text{BR}(\bar{\mathbf{G}}_{k-1}), \bar{\mathbf{G}}_{k-1}) - J(\boldsymbol{\pi}_k^*, \bar{\mathbf{G}}_{k-1}) \geq 0$.
2. **Average Policy Error.** The error in the supervised learning step: $\epsilon_{sl}^k := d_\Pi(\bar{\boldsymbol{\pi}}_k, \boldsymbol{\Pi}_k^{\text{true}})$ where $\boldsymbol{\Pi}_k^{\text{true}} := \frac{1}{k}\sum_{i=1}^k \boldsymbol{\pi}_i^*$ is the true average policy.
3. **Distribution Error.** The error of the CNF model $\bar{\mathbf{G}}_k$ in approximating the true mean-field flow $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}$: $\epsilon_{cnf}^k := d_{N_T}(\bar{\mathbf{G}}_k, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k})$.

We will use the following two assumptions, which are satisfied under mild conditions on the reward and transition functions:

Assumption 1 (Lipschitz continuity of J)**.** *For any policy $\boldsymbol{\pi}$ and any two flows $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$: $|J(\boldsymbol{\pi}, \boldsymbol{\mu}_1) - J(\boldsymbol{\pi}, \boldsymbol{\mu}_2)| \leq L \cdot d_{N_T}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$.*

Assumption 2 (Lipchitz continuity of MF)**.** *For any policies $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$, the generated mean fields satisfy: $d_{N_T}(\boldsymbol{\mu}^{\boldsymbol{\pi}_1}, \boldsymbol{\mu}^{\boldsymbol{\pi}_2}) \leq L_{MF} d_{\Pi}(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$.*

The following result provides a bound on the exploitability of the policy computed by our algorithm.

Theorem 1 (Convergence to approximate Nash equilibrium)**.** *Let $e_k^{true} := J(BR(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}), \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\bar{\boldsymbol{\pi}}_k, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) \geq 0$ be the **true exploitability** at iteration k, which measures the incentive to deviate from the policy $\bar{\boldsymbol{\pi}}_k$ in the true distribution it generates, $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}$. Under Assumptions 1 and 2, we have:*

$$e_k^{true} < C_0 e_0^{cnf} + \frac{1}{k} \sum_{i=1}^{k-1} \left[(i+1)\epsilon_{br}^{i+1} + C_1(\epsilon_{sl}^{i+1} + \epsilon_{cnf}^{i+1}) + \frac{C_2}{i} \right]$$

for some constants $C_0, C_1, C_2 > 0$.

The proof is provided in App. B. It relies on analyzing the propagation of errors.

5 Experiments

To validate our method, we present three distinct scenarios, each designed to showcase specific strengths and potential applications. The initial two examples illustrate the benefit of learning the average policy without performance degradation, while also revealing limitations in capturing local dependencies. We then introduce a more complex case study to demonstrate the effectiveness of DEDA-FP against established benchmarks, highlighting its ability to directly learn the environment and achieve a richer problem representation. Further results and an additional example (the *price impact model*) can be found in Appendix E.

Evaluation: To evaluate the algorithms' performance, we approximate the exploitability (defined in Def. 2). Since the model-free setting prevents direct computation of the optimal value we approximate the first term . In all cases, we conduct 4 independent runs and report the mean and standard deviation of the exploitability across these runs to assess the consistency of our results.

Architecture details: For the DRL algorithms, we used the Stable Baselines library by Hill et al. [2018] for the implementation of the DeepRL algorithm (SAC for the first two examples, PPO for the case study). We used an RTX4090 GPU with 24 GB RAM for each experiment. For the policy network we use a multi-layer perceptron (MLP) consisting of two hidden layers, each with 256 units and two parallel output layers for mean and standard deviation. To model distributions, we adapt a version of Neural Spline Flows (NSF) with autoregressive layers [Durkan et al., 2019] for handling time dependencies. More details about NSF work and implementation see Appendix D.

5.1 Beach Bar Problem

Environment: *"I would like to go to that nice bar on the beach, unless it is too crowded!".* We consider a continuous space version of the beach bar problem introduced in [Perrin et al., 2020] in discrete space. We take $\mathcal{X} = [0, 1]$, $\mathcal{A} = [-0.3, 0.3]$ as a continuous state space with dynamics: $x_{t+1} = x_t + a_t + \epsilon_t$. If the agent reaches the boundary of $\mathcal{X}$ and tries to exit, it is pushed back inside. ϵ_t represents the noise and is uniformly distributed over $[-0.1, 0.1]$. We take the initial distribution $\mu_0 = Unif[0, 1]$. The time horizon is set as $N_T = 10$. For the one-step reward function, we take $r(x, a, \mu) = -C_1|x - x_{\text{bar}}|^2 - C_2\mu(x) - C_3|a|^2$, where $C_1, C_2, C_3 \geq 0$, $x_{bar} = 0.5$ and $\mu(x)$ denotes the value of the density at x and represent the *congestion avoidance*.

Numerical results: Fig. 2 presents a comparison between the algorithms, illustrating the final flow $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_K}$ after the last Fictitious Play iteration (K). As can be seen in the figure, **DEDA-FP** demonstrates a superior interpretation of the problem formulation, achieving a smoother distribution concentrated around the center. Furthermore, the exploitability decay analysis indicates that this improved distributional representation in Algo. 3 does not come at the cost of performance. An important challenge highlighted by this problem, revealing a limitation of both benchmarks Algo. 1

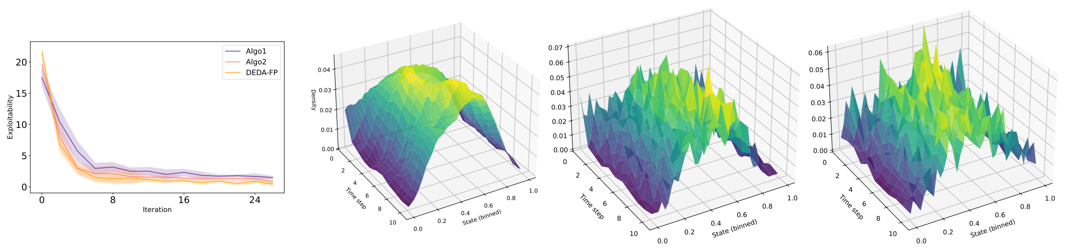


Figure 2: **Beach Bar Problem.** a): exploitability of Algo. 1, Algo. 2 and DEDA-FP; b) NE distribution DEDA-FP; c) NE distribution Algo. 2; d) NE distribution Algo. 1

and Algo. 2, lies in the approach to compute the local dependence $\mu(x)$ within the reward function. Due to the absence of an accessible approximation model for querying, we redefined the mean field cost as follows: $\mu(x) = (\mu^N * \rho)(x)$, and $(\mu * \rho)(x) := \int_{\mathcal{X}} \mu(y)\rho(x-y)dy$, ρ is the Gaussian density $\rho(x) := \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}$ and $\mu_t^N := \frac{1}{N}\sum_{i=1}^N \delta_{X_t^i}$ where X_t^i is the position at time t of the agent i and δ is the Dirac delta measure.

5.2 Linear-Quadratic (LQ) model

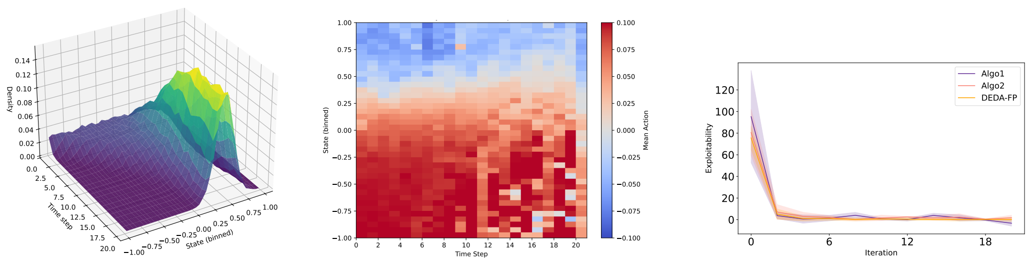


Figure 3: **LQ Model DEDA-FP** Left: mean field flow $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_K}$ at the last Fictitious Play (FP) iteration. Middle: policy $\bar{\boldsymbol{\pi}}_K$ at the last FP iteration. Right: comparison of exploitability between **DEDA-FP** and Algo. 1 and Algo. 2.

Environment: We consider a linear-quadratic (LQ) model with continuous state and action spaces and a finite time horizon N_T. Similar LQ models have been considered in [Laurière et al., 2022a] with discrete spaces and [Angiuli et al., 2023] with stationary mean field. We take the state space $\mathcal{X} = [-1, 1]$ and the action space $\mathcal{A} = [-0.1, 0.1]$. The dynamics is: $x_{t+1} = Ax_t + Ba_t + \bar{A}\bar{\mu}_t + \epsilon_t$, where A, B and $\bar{A}$ are real constants, $\bar{\mu}_t = \int_{\mathcal{X}} x\mu_t(x)\mathrm{d}x$ is the first moment (mean) of the distribution at time t, and ϵ_t represents the noise that is uniformly distributed over $[-0.1, 0.1]$. The reward is: $r(x, a, \mu) = -c_X|x - x_{\text{target}}|^2 - c_A|a|^2 - c_M|x - \bar{\mu}|^2$, where c_X, c_A and c_M are positive constants. The dynamics and the reward are linear and quadratic in the state x, the action a, and the mean $\bar{\mu}$ of the distribution. The agent learns to maximize the reward by finding an optimal policy that balances staying close to the target state, minimizing the action cost, and remaining near the population mean. In the experiment, we set $N_T = 20$, $A = 1$, $B = 1$, $\bar{A} = 0.06$. The reward coefficients are $c_{\mathcal{X}} = 5$, $c_A = 0.1$, and $c_M = 1$.

Numerical results: Fig. 3 shows the mean field flow and the learned policy after the last FP iteration by Algo. 2. The distribution concentrates near the target position $x_{\text{target}} = 0.6$ that aligns with the reward's high weight target discrepancy term. The learned policy also shows its linearity with respect to the state. The agent takes action that converges to the target position with increasing $|a|$ as the distance from the target increases. It is important to note that at later time steps, the policy's predictions far from the target may exhibit inconsistencies. However, this is not a limitation, as it is caused by the low agent density in those regions, rendering the action choices effectively arbitrary at those times. **This example demonstrates the policy network's effectiveness in approximating the average policy, and, importantly, shows that performance is not degraded.** The averaged exploitability curves of Algo. 1 and Algo. 2 both show the exploitability converges to zero quickly after several FP iterations and stay near zero, confirm the latter statement.

5.3 Case Study: *4-rooms exploration*

Building upon the limitations observed with Algo. 1 and Algo. 2, we now introduce a more complex setting to further highlight the strengths of the **DEDA-FP** (Algo. 3) against the benchmarks. In this problem, the approximation of the mean-field distribution becomes a critical aspect, primarily due to its inherent nature of entropy maximization.

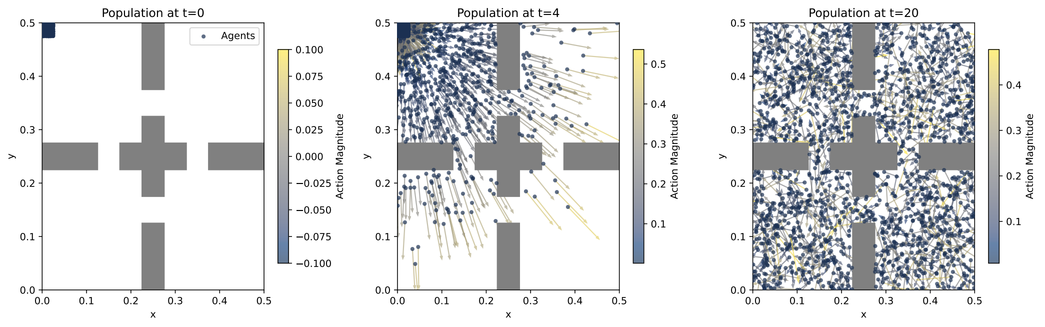


Figure 4: **4-rooms Exploration.** Visualization of a large, finite population of 2000 agents and their velocity vectors during exploration in the 4-rooms environment. The agents' behavior is governed by the mean-field Nash equilibrium policy learned by the DEDA-FP solver, Algo. 3. This shows how well the mean-field approximation captures the behavior of a large-population system.

Environment: The 4-rooms exploration MFG has been introduced using discrete spaces in [Geist et al., 2022] and has served as a benchmark e.g. in [Laurière et al., 2022a, Algumaei et al., 2023]. The state space and the action space are 2 dimensional ($d = 2$) and the states have constraints represented by walls forming four connected rooms. While the original model was discrete, we consider here a continuous space generalization, which is more natural because pedestrians move in continuous space. The action is the vector of movement (velocity) and the reward decreases with the mean field density: a larger density at the player's location means a smaller reward. Hence the players are encouraged to move in order to go to less crowded regions. In the end, if the time horizon is long enough, the mean field density becomes uniform. In this example, the stationary distribution is trivial and the key point is to learn the entire flow from the initial distribution to the stationary one. Following, the mathematical formulation. We take $\mathcal{X} = [0, 1]^2$, interpreted as a 2D domain. Time horizon N_T is set to 20. There are obstacles (walls) such that the domain has the shape of four connected rooms (see Fig. 5). The dynamics are: $x_{t+1} = x_t + v_t + \epsilon_t$, except that the agent cannot cross a wall. The reward is: $r(x, v, \mu) = -c_A||v||_2^2 - c_M \log(\mu(x) + \epsilon)$, where c_A, c_M and ϵ are positive constants, with ϵ very close to 0. **This reward (entropy maximization) discourages the agent from taking large actions (i.e., from moving a lot) and from being at a crowded location**. The initial distribution is uniform over a small square at the top left corner. We expect the agents to spread throughout the domain and, if the time horizon is long enough, the population distribution should converge to the uniform distribution over the domain.

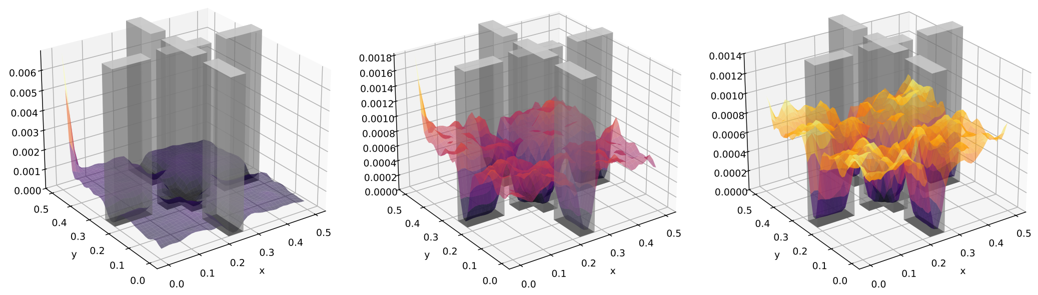
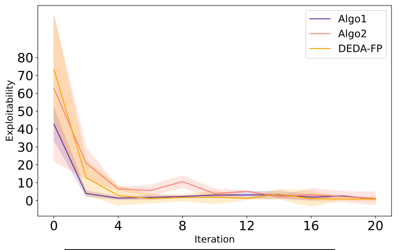


Figure 5: **4-rooms Exploration - NE flow.** The three plots represent the dynamics of the Nash Equilibrium mean field flow $\bar{\mathbf{G}}_K$ at time $t = 6, 15, 20$, obtained by **DEDA-FP**. It can be seen how the population is spreading across the 4 different rooms.

Discussion: The aim of this case study was to demonstrate the effectiveness of DEDA-FP (Algo. 3) compared to Algo. 1 and Algo. 2. Figs. 5, 6 and 7 summarize our results. First, DEDA-FP enables direct access to the local dependence $\mu(x)$ in the reward function, without the need to compute the convolution or any other approximation of the local density. Secondly, by leveraging the generative model's sample efficiency, our method yields a superior mean-field distribution representation during training (given a fixed-time budget for all algorithms) while maintaining comparable final performance.



Algorithm	Time (s)
Algo1	16.76±1.54
Algo2	15.14±1.19
DEDA-FP	1.52±0.23

Figure 6: (top) Exploitability decay. (bottom) Time to sample 5000 trajectories.

Scalable Sampling: In Fig. 6 (bottom) we display a table highlighting DEDA-FP's $> 10\times$ efficiency advantage over Algo. 1 and Algo. 2 in generating 5000 trajectories (time horizon: $T = 20$). This represents a substantial reduction in the computational cost of rollouts (critical, for instance, when applying the mean field policy in finite agent settings; see Fig. 4), enabling a considerably faster approximation of the environment (distribution and reward).

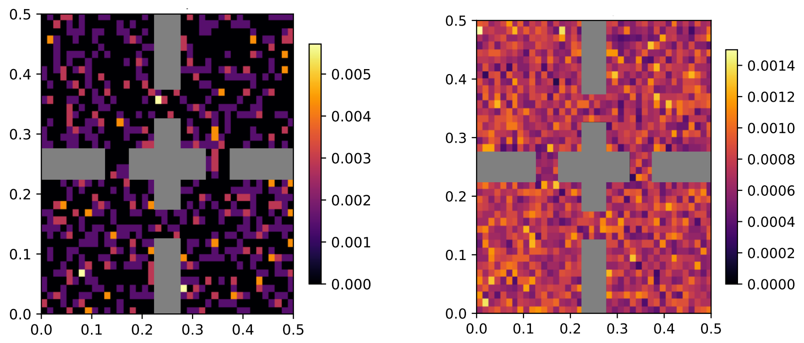


Figure 7: **4-rooms Exploration - Algo 2 vs DEDA-FP** The figure shows the last step Nash equilibrium policy generate by (left) Algo. 2 and (right) **DEDA-FP** with a fixed-time budget of $10s$. Remarkably, DEDA-FP can approximate the mean field distribution by sampling 10 times more agents than Algo. 2.

6 Conclusion and Limitations

Conclusions. In this paper, we introduce a deep reinforcement learning algorithm for non-stationary continuous MFGs. Our primary contribution lies in the development of DEDA-FP, which combines the strengths of Fictitious Play and supervised learning to accurately compute both the Nash equilibrium policy and the mean field distribution. Our approach enables efficient sampling and density approximation by leveraging time-conditioned normalizing flows, addressing the critical limitations in scalability and local mean field dependencies. In Theorem 1, we provide an error propagation analysis of DEDA-FP. Through three increasingly complex numerical experiments, we demonstrate the effectiveness of DEDA-FP in solving continuous space non-stationary MFGs with general dynamics and rewards. The results show that our approach yields a significant contribution to the application of RL techniques to continuous MFGs.

Limitations and future work. As of now, we are still lacking a complete theoretical understanding of the proposed algorithm, particularly due to the complexity of analyzing deep neural networks training. We also left for future work extensions beyond standard MFGs, such as multiple populations and graphon games, or MFGs with common noise and real-world applications. Furthermore, our present evaluation relies on approximate exploitability, which, while a state-of-the-art technique for assessing Nash equilibria, provides an evaluation that is inherently dependent on the environment approximation. Future research will investigate this aspect further.

Acknowledgments and Disclosure of Funding

M.L. is affiliated with the NYU Shanghai Center for Data Science and the NYU-ECNU Institute of Mathematical Sciences at NYU Shanghai. J.S. is partially supported by NSF Award 1922658. Computing resources were provided by NYU Shanghai HPC.

References

Yves Achdou and Jean-Michel Lasry. Mean field games for modeling crowd motion. In *Contributions to partial differential equations and applications*, pages 17–42. Springer, 2018.

Yves Achdou and Mathieu Laurière. Mean field games and applications: Numerical aspects. *Mean Field Games: Cetraro, Italy 2019*, 2281:249–307, 2020.

Yves Achdou, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll. Income and wealth distribution in macroeconomics: A continuous-time approach. *The review of economic studies*, 89(1):45–86, 2022.

Talal Algumaei, Ruben Solozabal, Reda Alami, Hakim Hacid, Merouane Debbah, and Martin Takáč. Regularization of the policy updates for stabilizing mean field games. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 361–372. Springer, 2023.

Andrea Angiuli, Jean-Pierre Fouque, and Mathieu Laurière. Unified reinforcement Q-learning for mean field game and control problems. *Mathematics of Control, Signals, and Systems*, 34(2):217–271, 2022.

Andrea Angiuli, Jean-Pierre Fouque, Ruimeng Hu, and Alan Raydan. Deep reinforcement learning for infinite horizon mean field problems in continuous spaces. *arXiv preprint arXiv:2309.10953*, 2023.

George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2): 156–172, 2008.

Pierre Cardaliaguet and Saeed Hadikhanloo. Learning in mean field games: the fictitious play. *ESAIM Control Optim. Calc. Var.*, 23(2):569–591, 2017. ISSN 1292-8119. doi: 10.1051/cocv/2016004.

Pierre Cardaliaguet and Charles-Albert Lehalle. Mean field game of controls and an application to trade crowding. *Mathematics and Financial Economics*, 12:335–363, 2018.

René Carmona. Applications of mean field games in financial engineering and economic theory. In *Mean Field Games on Agent Based Models to Nash Equilibria, AMS 2020*, pages 165–218. American Mathematical Society, 2021.

René Carmona and François Delarue. *Probabilistic theory of mean field games with applications. I*, volume 83 of *Probability Theory and Stochastic Modelling*. Springer, Cham, 2018. ISBN 978-3-319-56437-1; 978-3-319-58920-6. Mean field FBSDEs, control, and games.

René Carmona, François Delarue, and Daniel Lacker. Mean field games of timing and models for bank runs. *Applied Mathematics & Optimization*, 76(1):217–260, 2017.

Kai Cui and Heinz Koeppl. Approximately solving mean field games via entropy-regularized deep reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1909–1917. PMLR, 2021.

Boualem Djehiche, Alain Tcheukam, and Hamidou Tembine. Mean-field-type games in engineering. *AIMS Electronics and Electrical Engineering*, 1(1):18–73, 2017.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

Romuald Elie, Julien Perolat, Mathieu Laurière, Matthieu Geist, and Olivier Pietquin. On the convergence of model free learning in mean field games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7143–7150, 2020.

Massimo Fornasier and Francesco Solombrino. Mean-field optimal control. *ESAIM: Control, Optimisation and Calculus of Variations*, 20(4):1123–1152, 2014.

Zuyue Fu, Zhuoran Yang, Yongxin Chen, and Zhaoran Wang. Actor-critic provably finds Nash equilibria of linear-quadratic mean-field games. In *International Conference on Learning Representations*, 2019.

Xiaohu Ge, Haoming Jia, Yi Zhong, Yong Xiao, Yonghui Li, and Branka Vucetic. Energy efficient optimization of wireless-powered 5G full duplex cellular networks: A mean field game approach. *IEEE Transactions on Green Communications and Networking*, 3(2):455–467, 2019.

Matthieu Geist, Julien Pérolat, Mathieu Laurière, Romuald Elie, Sarah Perrin, Oliver Bachem, Rémi Munos, and Olivier Pietquin. Concave utility reinforcement learning: The mean-field game viewpoint. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 489–497, 2022.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

Xin Guo, Anran Hu, Renyuan Xu, and Junzi Zhang. Learning mean-field games. *Advances in Neural Information Processing Systems*, 32:4966–4976, 2019.

Xin Guo, Anran Hu, Renyuan Xu, and Junzi Zhang. A general framework for learning mean-field games. *Mathematics of Operations Research*, 48(2):656–686, 2023.

Ahmed Farhan Hanif, Hamidou Tembine, Mohamad Assaad, and Djamal Zeghlache. Mean-field games for resource sharing in cloud-based networks. *IEEE/ACM Transactions on Networking*, 24(1):624–637, 2015.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines, 2018.

Ruimeng Hu and Mathieu Laurière. Recent developments in machine learning methods for stochastic control and games. *Numerical Algebra, Control and Optimization*, 14(3):435–525, 2024.

Minyi Huang, Roland P Malhamé, Peter E Caines, et al. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252, 2006.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Aimé Lachapelle and Marie-Therese Wolfram. On a mean field game approach modeling congestion and aversion in pedestrian crowds. *Transportation research part B: methodological*, 45(10):1572–1589, 2011.

Aimé Lachapelle, Jean-Michel Lasry, Charles-Albert Lehalle, and Pierre-Louis Lions. Efficiency of the price formation process in presence of high frequency participants: a mean field game analysis. *Mathematics and Financial Economics*, 10:223–262, 2016.

Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Jpn. J. Math.*, 2(1):229–260, 2007. ISSN 0289-2316. doi: 10.1007/s11537-007-0657-8.

Mathieu Laurière, Sarah Perrin, Sertan Girgin, Paul Muller, Ayush Jain, Theophile Cabannes, Georgios Piliouras, Julien Pérolat, Romuald Elie, and Olivier Pietquin. Scalable deep reinforcement learning algorithms for mean field games. In *International Conference on Machine Learning*, pages 12078–12095. PMLR, 2022a.

Mathieu Laurière, Sarah Perrin, Julien Pérolat, Sertan Girgin, Paul Muller, Romuald Élie, Matthieu Geist, and Olivier Pietquin. Learning in mean field games: A survey. *arXiv preprint arXiv:2205.12944*, 2022b.

Weichao Mao, Haoran Qiu, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Ravishankar Iyer, and Tamer Basar. A mean-field game approach to cloud resource management with function approximation. *Advances in Neural Information Processing Systems*, 35:36243–36258, 2022.

Mojtaba Nourian, Peter E Caines, and Roland P Malhamé. Synthesis of Cucker-Smale type flocking via mean field stochastic control theory: Nash equilibria. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 814–819. IEEE, 2010.

Antonio Ocello, Daniil Tiapkin, Lorenzo Mancini, Mathieu Lauriere, and Eric Moulines. Finite-Sample Convergence Bounds for Trust Region Policy Optimization in Mean Field Games. In *Forty-second International Conference on Machine Learning*, 2024.

Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

Sarah Perrin, Julien Pérolat, Mathieu Laurière, Matthieu Geist, Romuald Elie, and Olivier Pietquin. Fictitious play for mean field games: Continuous time analysis and applications. *Advances in Neural Information Processing Systems*, 2020.

Sarah Perrin, Mathieu Laurière, Julien Pérolat, Matthieu Geist, Romuald Élie, and Olivier Pietquin. Mean Field Games Flock! The Reinforcement Learning Way. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 356–362. International Joint Conferences on Artificial Intelligence Organization, 8 2021.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

Naci Saldi, Tamer Başar, and Maxim Raginsky. Approximate markov-nash equilibria for discrete-time risk-sensitive mean-field games. *Mathematics of Operations Research*, 45(4):1596–1620, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Jayakumar Subramanian and Aditya Mahajan. Reinforcement learning in stationary mean-field games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 251–259, 2019.

Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019.

Annie Wong, Thomas Bäck, Anna V Kononova, and Aske Plaat. Deep multiagent reinforcement learning: Challenges and directions. *Artificial Intelligence Review*, 56(6):5023–5056, 2023.

Chungang Yang, Jiandong Li, Prabodini Semasinghe, Ekram Hossain, Samir M Perlaza, and Zhu Han. Distributed interference and energy-aware power control for ultra-dense D2D networks: A mean field game. *IEEE Transactions on Wireless Communications*, 16(2):1205–1217, 2016.

Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

Batuhan Yardim, Semih Cayci, Matthieu Geist, and Niao He. Policy mirror ascent for efficient and independent learning in mean field games. In *International Conference on Machine Learning*, pages 39722–39754. PMLR, 2023.

Muhammad Aneeq uz Zaman, Kaiqing Zhang, Erik Miehling, and Tamer Başar. Reinforcement learning in non-stationary discrete-time linear-quadratic mean-field games. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2278–2284, 2020.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

A Model Assumptions

We discuss our assumption here.

- **Existence:** Our work assumes the existence of a Mean Field Nash Equilibrium (MFNE). This assumption is grounded in established results for related problems, as existence proofs typically rely on fixed-point arguments under specific regularity conditions on the coefficients not yet proven for our exact setting. Specifically, our justification relies on the work of [Saldi et al., 2020], who established the existence of an equilibrium for a broad class of discrete-time MFGs in an infinite-horizon, risk-sensitive setting. Crucially, their work also provides an argument that this infinite-horizon, risk-sensitive cost can be approximated by a finite-horizon game. Since our finite-horizon setting is a well-posed approximation of a problem where existence is proven, we can assume that an equilibrium also exists in our case. Furthermore, we expect that existence holds under mild assumptions (typically, continuity) using the Schauder fixed point theorem in a suitable functional space. Under more restrictive assumptions (typically, Lipschitz continuity with a small Lipschitz constant), the Banach fixed point theorem yields both existence and uniqueness. While a full proof is beyond the scope of the present paper, which focuses on a DRL algorithm, we included this discussion in Section 2 of the paper, after the definition of MFNE (Definition 1).
- **Uniqueness of Equilibrium:** For our theoretical convergence discussion, we also assume uniqueness. This is not required for the DEDA-FP algorithm itself to run. Following the approach of e.g. [Elie et al., 2020], uniqueness can be ensured by adopting the standard Lasry-Lions monotonicity assumption. Intuitively, this condition means that the reward function discourages agents from being too concentrated.

B Proof of the convergence of DEDA-FP

We are going to prove Theorem 1.

As mentioned in Section 4, we follow the methodology developed by [Elie et al., 2020]. Their analysis rigorously bounds the propagation of errors from an approximate best-response computation. We extend this framework to account for two additional sources of error inherent to our algorithm: the average policy error and the distribution error.

We first bound the exploitability with respect to the tractable distribution $\bar{\mathbf{G}}_k$ learned by the algorithm (Lemma 1). Then we connect this result with the true value of the exploitability to show that the pair is an approximate Nash equilibrium $(\bar{\boldsymbol{\pi}}_k, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k})$ (Lemma 2). Combining these two bounds yield Theorem 1 stated above.

We define $d_{N_T}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = \max_{t=0,\dots,N_T} W_1(\mu_{1,t}, \mu_{2,t})$ and $d_\Pi(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) := \max_{t=0,\dots,N_T} \int_{\mathcal{X}} W_1(\pi_{1,t}(\cdot|x), \pi_{2,t}(\cdot|x))dx$, where W_1 is the Wasserstein-1 distance on $\mathcal{X} = \mathbb{R}^d$ and we consider only a class of regular policies.

Lemma 1. *Let Assumptions 1 and 2 above hold. We have:*

$$e_k \leq \frac{e_0}{k} + \frac{1}{k}\sum_{i=1}^{k}\left((i+1)\epsilon_{br}^{i+1} + C_1(\epsilon_{sl}^{i+1} + \epsilon_{cnf}^{i+1}) + \frac{C_2}{i}\right)$$

*for some constants $C_1, C_2 > 0$, where $e_k := J(BR(\bar{\mathbf{G}}_k), \bar{\mathbf{G}}_k) - J(\bar{\boldsymbol{\pi}}_k, \bar{\mathbf{G}}_k)$ is the **tractable exploitability** defined with respect to the learned distribution $\bar{\mathbf{G}}_k$.*

Proof. The proof adapts the derivation of estimate (8) in Theorem 6 of [Elie et al., 2020]. Let $\hat{e}_k := J(\boldsymbol{\pi}^*_{k+1}, \bar{\mathbf{G}}_k) - J(\bar{\boldsymbol{\pi}}_k, \bar{\mathbf{G}}_k)$. The total tractable exploitability is $e_k = J(\mathrm{BR}(\bar{\mathbf{G}}_k), \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}^*_{k+1}, \bar{\mathbf{G}}_k) + \hat{e}_k = \epsilon_{br}^{k+1} + \hat{e}_k$. We analyze the evolution of the term $(k+1)\hat{e}_{k+1} - k\hat{e}_k$. Following the original proof structure, we can prove that:

$$(k+1)\hat{e}_{k+1} - k\hat{e}_k \leq (k+1)\left(J(\mathrm{BR}(\bar{\mathbf{G}}_{k+1}), \bar{\mathbf{G}}_{k+1}) - J(\boldsymbol{\pi}^*_{k+1}, \bar{\mathbf{G}}_{k+1})\right).$$

The term in the parentheses is exactly ϵ_{br}^{k+1}. This appears to simplify things, but the derivation used in the original proof is made using exact flows $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}$ whereas our value functions are defined with respect to the tractable flows $\bar{\mathbf{G}}_k$. It can be seen that the extra term's magnitude depends on $d_{N_T}(\bar{\mathbf{G}}_{k+1}, \bar{\mathbf{G}}_k)$. We bound this distance using the triangle inequality and our error definitions:

$$\begin{aligned} d_{N_T}(\bar{\mathbf{G}}_{k+1}, \bar{\mathbf{G}}_k) &\leq d_{N_T}(\bar{\mathbf{G}}_{k+1}, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_{k+1}}) + d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_{k+1}}, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) + d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}, \bar{\mathbf{G}}_k) \\ &\leq \epsilon_{cnf}^{k+1} + d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_{k+1}}, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) + \epsilon_{cnf}^k. \end{aligned}$$

The term $d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_{k+1}}, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k})$ is further bounded by:

$$\begin{aligned} d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_{k+1}}, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) &\leq d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_{k+1}}, \boldsymbol{\mu}^{\boldsymbol{\Pi}_{k+1}^{true}}) + d_{N_T}(\boldsymbol{\mu}^{\boldsymbol{\Pi}_{k+1}^{true}}, \boldsymbol{\mu}^{\boldsymbol{\Pi}_k^{true}}) + d_{N_T}(\boldsymbol{\mu}^{\boldsymbol{\Pi}_k^{true}}, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) \\ &\leq \epsilon_{sl}^{k+1} + d_{N_T}(\boldsymbol{\mu}^{\boldsymbol{\Pi}_{k+1}^{true}}, \boldsymbol{\mu}^{\boldsymbol{\Pi}_k^{true}}) + \epsilon_{sl}^k. \end{aligned}$$

The term $d_{N_T}(\boldsymbol{\mu}^{\boldsymbol{\Pi}_{k+1}^{true}}, \boldsymbol{\mu}^{\boldsymbol{\Pi}_k^{true}})$ corresponds to the change in the exact Fictitious Play update, which is known to be of order $O(1/k)$ (Lemma 5 in [Elie et al., 2020]). Let's denote the stability constant as C_{FP}. So, $d_{N_T}(\boldsymbol{\mu}^{\boldsymbol{\Pi}_{k+1}^{true}}, \boldsymbol{\mu}^{\boldsymbol{\Pi}_k^{true}}) \leq C_{FP}/k$. Combining these, we get a bound on the change in our tractable distributions:

$$d_{N_T}(\bar{\mathbf{G}}_{k+1}, \bar{\mathbf{G}}_k) \leq \epsilon_{cnf}^{k+1} + \epsilon_{cnf}^k + \epsilon_{sl}^{k+1} + \epsilon_{sl}^k + \frac{C_{FP}}{k}.$$

Substituting these bounds back into the recursive inequality for $(k+1)e_{k+1} - ke_k$ introduces additive terms related to ϵ_{br}, ϵ_{sl}, and ϵ_{cnf}. The final form of the recursive inequality becomes:

$$(k+1)e_{k+1} - ke_k \leq (k+1)\epsilon_{br}^{k+1} + C_1(\epsilon_{sl}^{k+1} + \epsilon_{cnf}^{k+1}) + \frac{C_2}{k}.$$

Applying Lemma 9 from [Elie et al., 2020] to solve this recursion yields the result. □

Now, it is necessary to establish a connection between the tractable exploitability (bounded in Lemma 1) and the actual exploitability as defined in (1). The subsequent lemma serves to create this connection.

Lemma 2. *Under the same assumptions, the true exploitability e_k^{true} is close to the tractable exploitability e_k. Specifically,*

$$|e_k^{true} - e_k| \leq 4L \cdot \epsilon_{cnf}^k,$$

where L is the Lipschitz constant of Assumption 1.

Proof. We seek to bound $|e_k^{true} - e_k|$. Using the triangle inequality:

$$|e_k^{true} - e_k| \leq \underbrace{|J(\mathrm{BR}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}), \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\mathrm{BR}(\bar{\mathbf{G}}_k), \bar{\mathbf{G}}_k)|}_{\text{Term 1}} + \underbrace{|J(\bar{\boldsymbol{\pi}}_k, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\bar{\boldsymbol{\pi}}_k, \bar{\mathbf{G}}_k)|}_{\text{Term 2}}.$$

Let us bound each term separately.

Term 2 (Bound on Policy Value): This term is straightforward using Assumption 1.

$$\begin{aligned} |J(\bar{\boldsymbol{\pi}}_k, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\bar{\boldsymbol{\pi}}_k, \bar{\mathbf{G}}_k)| &\leq L \cdot d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}, \bar{\mathbf{G}}_k) \\ &\leq L \cdot \epsilon_{cnf}^k. \end{aligned}$$

Term 1 (Bound on Best Response Value): This term requires more care. Let $\boldsymbol{\pi}_1 = \mathrm{BR}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k})$ and $\boldsymbol{\pi}_2 = \mathrm{BR}(\bar{\mathbf{G}}_k)$. We want to bound $|J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k)|$. We add and subtract $J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)$:

$$|J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k)| \leq |J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)| + |J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k)|.$$

The first part is again bounded by Lipschitz continuity: $|J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)| \leq L \cdot d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}, \bar{\mathbf{G}}_k) \leq L \cdot \epsilon_{cnf}^k$. For the second part, $|J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k)|$, we know by definition of best response that $J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) \geq J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)$. So we need to bound the non-negative quantity $J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)$. Using the fact that $\boldsymbol{\pi}_1$ is a best response against $\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}$,

$$\begin{aligned} 0 \leq J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k) = J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) &- J(\boldsymbol{\pi}_2, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) \\ &+ J(\boldsymbol{\pi}_2, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) + J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k) \\ &\leq \left(J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_2, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k})\right) + \left(J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)\right). \end{aligned}$$

Both remaining terms can be bounded by Assumption 1:

$$\begin{aligned} J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k) &\leq |J(\boldsymbol{\pi}_2, \bar{\mathbf{G}}_k) - J(\boldsymbol{\pi}_2, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k})| + |J(\boldsymbol{\pi}_1, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\boldsymbol{\pi}_1, \bar{\mathbf{G}}_k)| \\ &\leq L \cdot d_{N_T}(\bar{\mathbf{G}}_k, \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) + L \cdot d_{N_T}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}, \bar{\mathbf{G}}_k) \\ &\leq 2L \cdot \epsilon_{cnf}^k. \end{aligned}$$

Combining the parts for Term 1, we get $|J(\mathrm{BR}(\boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}), \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}) - J(\mathrm{BR}(\bar{\mathbf{G}}_k), \bar{\mathbf{G}}_k)| \leq L\epsilon_{cnf}^k + 2L\epsilon_{cnf}^k = 3L \cdot \epsilon_{cnf}^k$.

Final Bound: Combining the bounds for Term 1 and Term 2:

$$|e_k^{true} - e_k| \leq (3L \cdot \epsilon_{cnf}^k) + (L \cdot \epsilon_{cnf}^k) = 4L \cdot \epsilon_{cnf}^k.$$

□

This completes the proof of Theorem 1.

C Algorithms Details

In the context of MFGs, Fictitious Play (FP) operates by iteratively computing the best response of a representative agent against the distribution induced by the average of past best responses of the entire population. The discrete-time FP process involves several key steps at each iteration $k = 0 \ldots, K$:

1. **Best Response Computation**: An agent finds its best response policy $\boldsymbol{\pi}_k^{BR}$ against the approximated average population distribution $\bar{\boldsymbol{\mu}}_{k-1}$.
2. **Average Policy Update**: The average policy $\bar{\boldsymbol{\pi}}_k$ is computed by averaging the current best response policy with all previous policies. In particular we have $\forall t = 0, \ldots, N_T$

$$\bar{\boldsymbol{\pi}}_{k-1}(\cdot|x,t) = \frac{\sum_{i=0}^{i=k} \pi_i^{BR}(\cdot|x,t) \mu_t^{\boldsymbol{\pi}_k^{BR}}(x)}{\sum_{i=0}^{i=k} \mu_t^{\boldsymbol{\pi}_k^{BR}}(x)}$$

3. **Average Distribution Update**: The average population distribution $\bar{\boldsymbol{\mu}}_k$ is updated by averaging the current population distribution with past distributions. In particular we have,

$$\bar{\boldsymbol{\mu}}_k = \frac{k-1}{k} \bar{\boldsymbol{\mu}}_{k-1} + \frac{1}{k} \boldsymbol{\mu}^{\boldsymbol{\pi}_k^{BR}}$$

It can be seen that $\bar{\boldsymbol{\mu}}_k = \boldsymbol{\mu}^{\bar{\boldsymbol{\pi}}_k}$

At the end of the algorithm, the pair $(\bar{\boldsymbol{\pi}}_K, \bar{\boldsymbol{\mu}}_K)$ represents the Nash equilibrium of the mean-field game problem. In a model-free framework, however, direct analytical computation is not feasible. For this reason, all three steps must be approximated to fully solve the game. While Algo. 1 and Algo. 2 only address parts of this problem, **DEDA-FP** (described in Algo. 3) provides the complete solution.

Notation. Here, we provide the notation used in the pseudocodes.

- $\mathcal{M}$ represent the policy buffer use to store all the best responses during FP.
- $\mu_t^{N,\mathcal{M}}$ is the empirical distribution at time t, generated by N agents using a policy assigned uniformly at random from the buffer $\mathcal{M}$.
- $\mu_t^{N,\pi}$ is the empirical distribution, at time t, generated by N agents using the policy $\bar{\pi}$.
- $J_{\mu_0,\mathcal{M}}^N(\boldsymbol{\pi})$ is the approximated cost function, computed against $N-1$ trajectories. These trajectories originate from points sampled from the initial distribution μ_0 and subsequently evolve according to $N-1$ policies sampled uniformly from the buffer $\mathcal{M}$.
- $J_{\mu_0,\bar{\boldsymbol{\pi}}}^N(\boldsymbol{\pi})$ is the approximated cost function, computed against $N-1$ trajectories. These trajectories originate from points sampled from the initial distribution μ_0 and subsequently all evolve according to $\bar{\boldsymbol{\pi}}$.

- $\mathcal{L}_{\text{NLL}}(\theta) = \mathbb{E}_{(t,s,a)\sim\mathcal{M}_{SL}}\left[-\log \boldsymbol{\pi}^{\theta}(a|t,s)\right] = -\frac{1}{M}\sum_{i=1}^{M}\log\mathcal{N}(a_i;\mu_\theta(s_i,t_i),\sigma_\theta(s_i,t_i))$, where M is the size of the replay buffer $\mathcal{M}_{SL}$ containing all the triples (t,s,a) sampled from the previous policies and μ_θ and σ_θ are the mean and standard deviation predicted by the policy network for the state s_i at time t_i.

Algo. 1 Simple Approach

```
1: Input: Initial distribution $\mu_0$; population size $N$; Number of iterations $K$.
2: Initialize: $\theta_0^*$, $\mathcal{M} := \{\boldsymbol{\pi}_0^*\}$ where $\boldsymbol{\pi}_0^* := \boldsymbol{\pi}^{\theta_0^*}$.
3: for iteration $k = 1$ to $K$ do
4:     Using DRL, find the best response $\boldsymbol{\pi}_k^* := \boldsymbol{\pi}^{\theta_k^*}$ such that:
           $$\boldsymbol{\pi}_k^* = \arg\max_{\boldsymbol{\pi}} J^N_{\mu_0,\mathcal{M}}(\boldsymbol{\pi})$$
5:     Add $\boldsymbol{\pi}_k^*$ in $\mathcal{M}$
6: end for
7: return $\mathcal{M}$
```

Algo. 2 Learning the NE Policy

```
1: Input: Initial distribution $\mu_0$; $N_{sa}$: number of state-action pairs to collect at every iteration, $N$: population
   size in population simulation; number of iterations $K$.
2: Initialize: $\theta_0^*$ and set $\bar{\theta}_0 = \theta_0^*$ since ($\bar{\boldsymbol{\pi}}^{\bar{\theta}_0} = \boldsymbol{\pi}^{\theta_0^*}$); sample, according to $\boldsymbol{\pi}_0^* := \boldsymbol{\pi}^{\theta_0^*}$, $N_{sa}$ (time)-state-action
   triples $(0,s,a)$ and define $\mathcal{M}_{SL}$ to store them.
3: for iteration $k = 1$ to $K$ do
4:     Using DRL, find the best response $\boldsymbol{\pi}_k^* := \boldsymbol{\pi}^{\theta_k^*}$ such that:
           $$\boldsymbol{\pi}_k^* = \arg\max_{\boldsymbol{\pi}} J^N_{\mu_0,\bar{\boldsymbol{\pi}}_{k-1}}(\boldsymbol{\pi})$$
5:     Collect $N_{sa}$ state-action samples of the form $(t,s,a)$ using $\boldsymbol{\pi}_k^*$ and store in $\mathcal{M}_{SL}$.
6:     Train the NN policy $\bar{\boldsymbol{\pi}}_k := \bar{\boldsymbol{\pi}}^{\bar{\theta}_k}$ using supervised learning to minimize the categorical loss:
           $$\mathcal{L}_{\text{NLL}}(\bar{\theta}) = \mathbb{E}_{(t,s,a)\sim\mathcal{M}_{SL}}\left[-\log\bar{\boldsymbol{\pi}}^{\bar{\theta}}(a|t,s)\right]$$
       This NN aims to mimic the behaviour of the average policy $\frac{1}{k}(\boldsymbol{\pi}_0^* + \cdots + \boldsymbol{\pi}_k^*)$.
7: end for
8: return $\bar{\boldsymbol{\pi}}^{\bar{\theta}_K}$
```

C.1 DEDA-FP components

In **DEDA-FP** (Algo. 3), both the overall orchestration and the individual components are chosen for a specific purpose. While other choices could be made, we explain below the rationale behind our choices.

Fictitious Play: this is the backbone of our method. The main advantage is that it is known to converge in larges classes of games. One drawback is that convergence can be lower than some other methods, but we prefered to sacrifice the convergence speed and ensure robustness rather than the opposite.

DRL for Best Response: Policies are functions defined on the continuous state and action spaces so they are infinite dimentsional. Hence we had to approximate them using parameterized functions. We chose neural networks due to the empirical success in a variety of machine learning tasks. As for the training, model-free RL has the advantage to avoid exact dynamic programming and hence scale well to highly complex problems. In the implementation we chose SAC and PPO but other choices could be made, depending on the specific MFG at hand.

Supervised learning for average policy: In general, convergence results for Fictitious Play are not for the last iterate policy (the best response computed in the last iteration) but only for the average policy. So computing the average policy is crucial to ensure convergence. However, our policies are neural networks and averaging neural networks is hard due to non-linearities. We thus have to train a new neural network for the average policy. Here, we chose to use supervised learning, drawing inspiration from Neural Fictitious Self-Play ([Heinrich and Silver, 2016]). **Conditional Normalizing Flow (CNF) for the Mean-Field:** This is a critical design choice that directly enables one of our paper's main contributions. To solve MFGs with local density dependence (e.g., congestion), we require a model that can both (1) sample from the population distribution and (2) compute its exact probability density at any given point in time and space ($p(x|t)$). Normalizing Flows (NFs) are well-established generative models that have been introduced precisely to provide both of these capabilities without time and CNFs are an extension of NFs, which allow us to take time into account in a natural way. Other generative models, like GANs or score-based diffusion models, can sample effectively but do not allow direct density evaluation, making them unsuitable for our goal.

D Implementation Details

D.1 Time Conditioned Neural Spline Flow

We employ the Neural Spline Flow (NSF) with autoregressive layers [Durkan et al., 2019] as the flow component in our time conditioned normalizing flow.

Neural Spline Flows The key idea in NSF is to transform a simple distribution (like a standard Gaussian) into a complex one using a series of invertible transformations. To make these transformations very flexible and efficient, NSF uses "rational-quadratic splines."

Rational-Quadratic Splines A spline can be seen as a flexible curve made up of pieces. In our case, each piece is a "rational-quadratic" function, which is a ratio of two quadratic polynomials. These functions are smooth and can be easily inverted, which is important for our model. A rational-quadratic spline is defined by a set of $K+1$ knots $\{(x^{(k)}, y^{(k)})\}_{k=0}^{K}$. The value of the spline at a given x is determined by which interval $[x^{(k)}, x^{(k+1)}]$ it falls into. Letting $\xi = (x - x^{(k)})/(x^{(k+1)} - x^{(k)})$ represent the normalized position within that interval, the spline segment is:

$$g(x) = \frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)}$$

where

$$\alpha^{(k)}(\xi) = s^{(k)} y^{(k+1)} \xi^2 + [y^{(k)} \delta^{(k+1)} + y^{(k+1)} \delta^{(k)}] \xi(1-\xi) + s^{(k)} y^{(k)} (1-\xi)^2$$

$$\beta^{(k)}(\xi) = s^{(k)} \xi^2 + [\delta^{(k+1)} + \delta^{(k)}] \xi(1-\xi) + s^{(k)} (1-\xi)^2$$

and $s^{(k)} = (y^{(k+1)} - y^{(k)})/(x^{(k+1)} - x^{(k)})$ is the slope of the line connecting the knots at the interval's boundaries. $\delta^{(k)}$ represents the derivative of the spline at knot k.

Autoregressive Neural Spline Flows In our implementation, we use the variant of NSF with autoregressive layers. This means that the parameters of the rational-quadratic spline transformation for each dimension of the data are predicted by an autoregressive neural network. Specifically, for each dimension i of the input x, the spline parameters are computed as a function of the previous dimensions $x_{1:i-1}$:

$$\theta_i = \text{NN}(x_{1:i-1})$$

where NN_{AR} denotes an autoregressive neural network. This autoregressive approach allows the model to capture complex dependencies between the dimensions of the data, as the transformation applied to each dimension is conditioned on the values of the preceding dimensions.

Time Conditioning To handle the non-stationary nature of the mean-field distribution, we explicitly condition the Neural Spline Flow on time t. This means that the entire transformation, and specifically the parameters of the rational-quadratic splines, are made dependent on the current time step. In our autoregressive setup, the neural network that predicts the spline parameters (NN in the equation above) not only takes the previous dimensions $x_{1:i-1}$ as input but also the time t. The time variable t is typically concatenated with the input features or fed into the neural network as an additional input, allowing the network to learn time-dependent transformations. This enables the flow to dynamically adjust its shape and density characteristics as time evolves from $t = 0$ to $t = T$, thereby capturing the non-stationary dynamics of the mean-field.

E Numerical Experiments details

This section provides further experimental results and detailed comparisons between our proposed DEDA-FP approach and the benchmark algorithms considered in the main paper.

E.1 Beach Bar Problem

Further numerical results for the Beach Bar problem are shown in Figures 8 and 9.

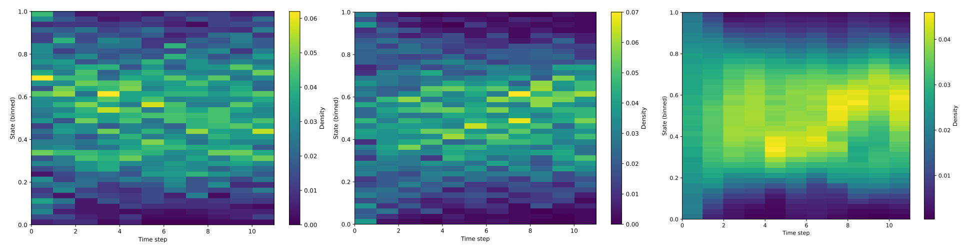


Figure 8: **Comparison of Nash equilibrium distribution heatmaps in the Beach Bar Problem**. From left to right: Algo. 1, Algo. 2, and DEDA-FP (Algo. 3). Thanks to its remarkable speed, DEDA-FP can utilize a high volume of samples (6x times in the displayed figure) for robust distribution approximation, a scale that proves computationally prohibitive for existing benchmarks.

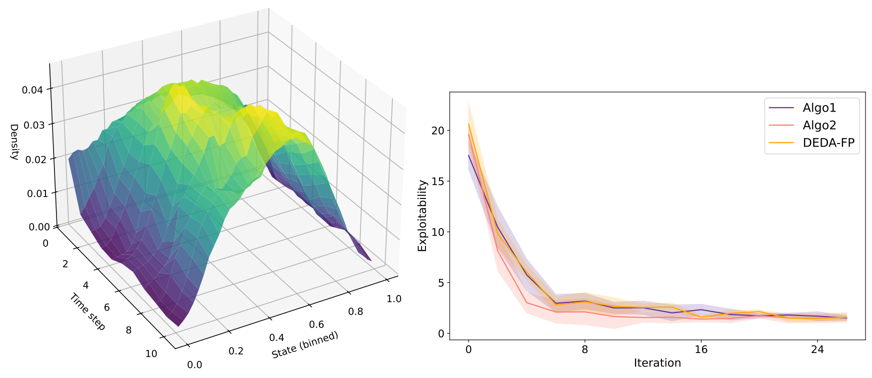


Figure 9: **Beach Bar Problem Results for DEDA-FP**. Left: Nash equilibrium distribution. Right: Exploitability decay comparison across algorithms.

E.2 LQ model

Further numerical results for the LQ problem are shown in Figures 10 and 11.

E.3 4-rooms exploration

Further numerical results for the 4-rooms exploration problem are shown in Figures 16 and 17.

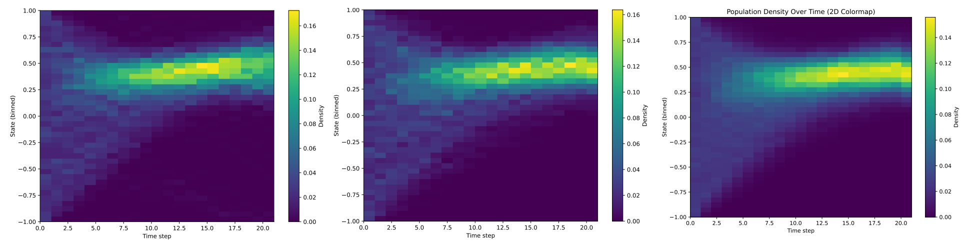


Figure 10: **Comparison of Nash equilibrium distribution heatmaps in the LQ Problem**. From left to right: Algo Algo. 1, Algo Algo. 2, and DEDA-FP (Algo. 3).

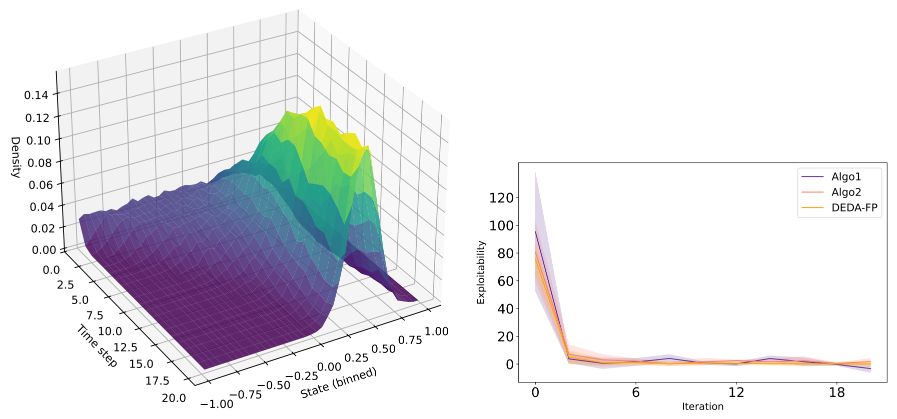


Figure 11: **LQ Problem Results for DEDA-FP**. Left: Nash equilibrium distribution. Right: Exploitability decay comparison across algorithms.

E.4 Market Model

Here we present one more example on an environment of a different type, with a financial application. It is a discrete time version of the price impact model in MFG literature, introduced by Carmona and Lacker.[3]

Environment: We consider a market model where $\mathcal{X} = [-5, 5]$ represents the inventory for a stock. The action space $\mathcal{A} = [-1, 1]$ represents the rate of trading for the stock. Each agent controls the inventory for the stock. The dynamics is: $x_{t+1} = x_t + a_t + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, 1)$. The reward is $r(x, a, \bar{a}) = -C_{\mathcal{X}} x^2 - C_{\mathcal{A}} a^2 + hx\bar{a}$, where $C_{\mathcal{X}}$, $C_{\mathcal{A}}$, and h are positive constants, $\bar{a}$ is the mean of the action. At each time t, the representative agent wants to minimize the shares held. In this model, the agent interacts with the distribution of the action instead of the population distribution. The mean field term $hx\bar{a}$ reflects the impact of the action on the price.

Numerical results: Results are shown in Figures 12, 13 and 14. We observe that traders tend to liquidate their portfolios (given to the x^2 term in the reward function). However, a proportion of agents is incentivized to buy instead of sell due to the interaction term. Moreover, we observe that our model (DEDA-FP) consistently provides a superior representation of the distribution, which is ensured by its efficiency in sampling a large number of agent positions at every time step.

[3]René Carmona and Daniel Lacker. A probabilistic weak formulation of mean field games and applications. *Annals of applied probability: an official journal of the Institute of Mathematical Statistics*, 25(3):1189–1231, 2015.

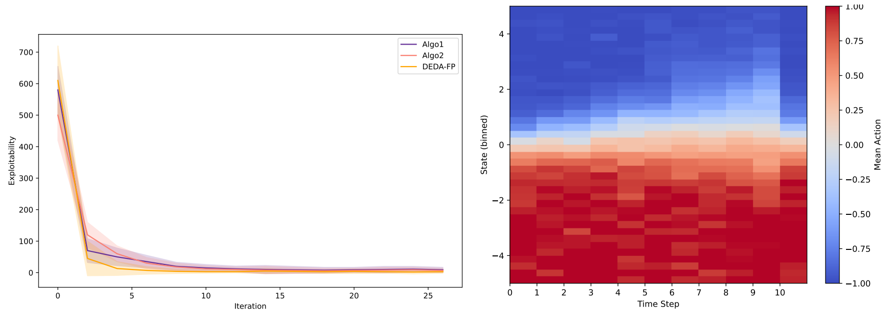


Figure 12: **Market Model Results:** Left: Exploitability decay comparison across algorithms; Right: Nash Equilibrium Policy learned by DEDA-FP

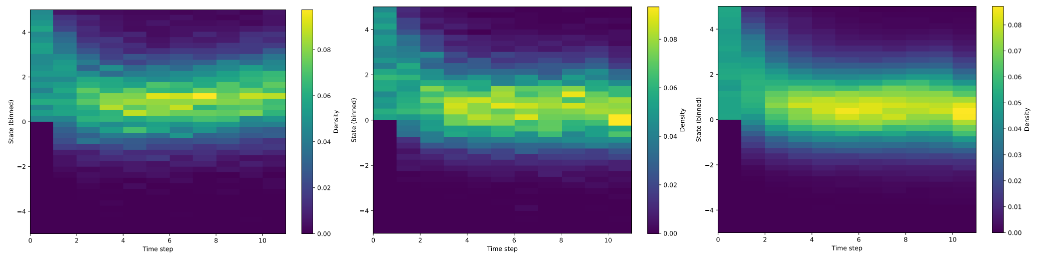


Figure 13: **Comparison of Nash equilibrium distribution heatmaps in the Market Model Problem**. From left to right: Algo Algo. 1, Algo Algo. 2, and DEDA-FP (Algo. 3).

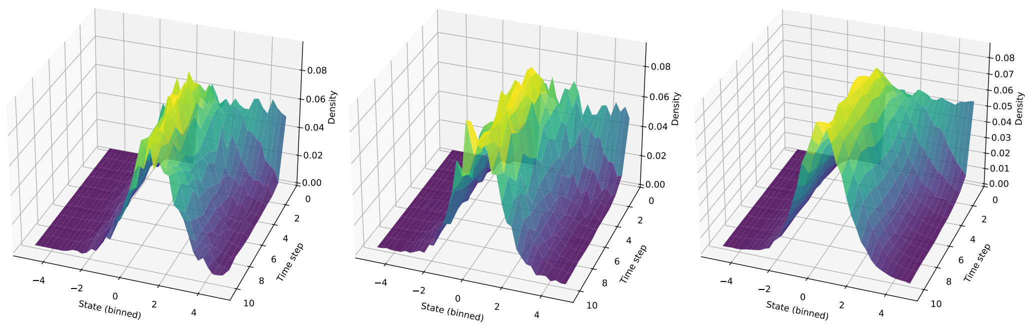


Figure 14: **Comparison of Nash equilibrium distribution in the Market Model Problem ((3D plots))**. From left to right: Algo Algo. 1, Algo Algo. 2, and DEDA-FP (Algo. 3).

F Hyperparameter Sweep

We sweep the learning rate over the set $\{3 \times 10^{-2}, 3 \times 10^{-3}, 3 \times 10^{-4}, 3 \times 10^{-5}, 3 \times 10^{-6}\}$ for Deep RL in Algo. 1 in to the center environment shown in Figure 15. We observe that a learning rate of 3×10^{-4} yields more stable training and faster convergence. Based on this observation, we adopt 3×10^{-4} for the Deep RL component in Algo. 2 and Algo. 3 as well.

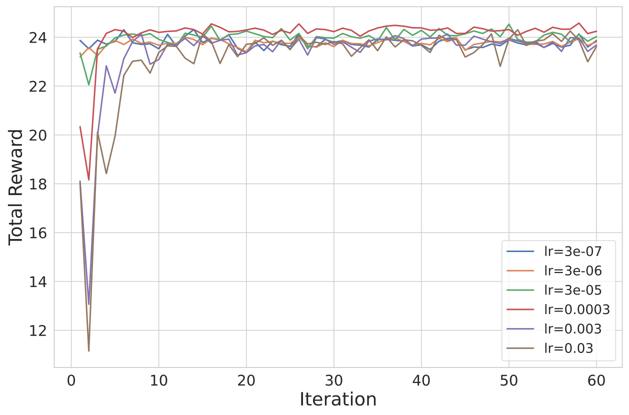


Figure 15: Total reward vs iterations for different learning rate of Deep RL in Algo. 1 in the center environment

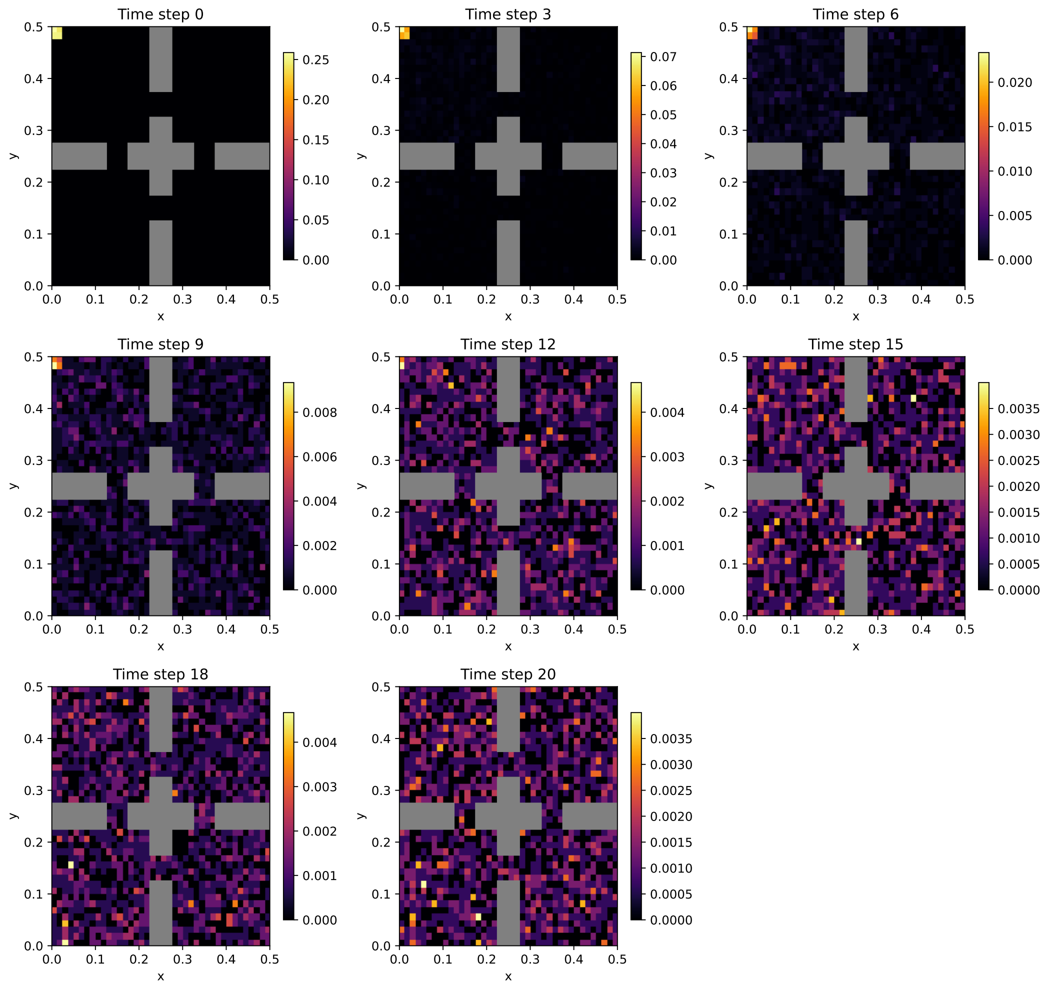


Figure 16: **4 rooms explorations**. Nash Equilibrium mean field flow obtained by Algo. 2 sampling 1500 trajectories

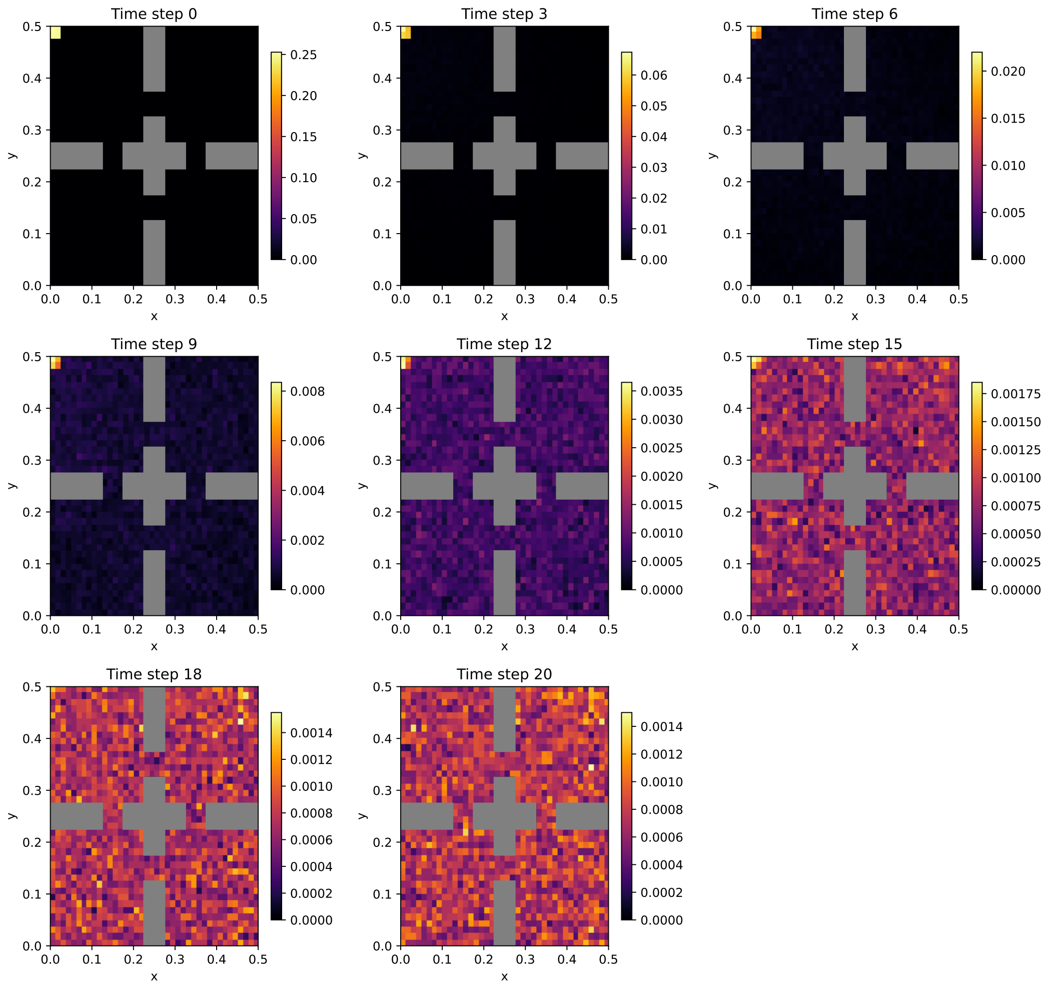


Figure 17: **4 rooms explorations**. Nash Equilibrium mean field flow obtained by Algo. 3 sampling 8000 trajectories $10x$ faster than Algo. 2 and Algo. 1.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provided the the detailed description of the MFGs models and our algorithm.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We are lack of theoretical analysis of the approximation of NE of current method.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We state our assumptions and provide a detailed proof of the theoretical result provided in the paper.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe both the detailed and each algorithm in the main text and appendix.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will open access to the code after the paper gets accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided the details of each experiments in the environment setup sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We averaged all the numerical results over 4 seeds and reported the error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mentioned in section 4 the computer resources we used to run the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We follow the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the societal impacts of the work in section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our model doesn't have a high risk for misuse

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited all the papers that studied the models we used in our paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new algorithm is well documented in our paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper doesn't include the human subjects in our experiments.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper doesn't include the human subjects in our experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We don't use LLMs to generate the methods.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.