Fast Sub-optimal Mean-Field Control with Drift Imitation for Large-scale Heterogeneous Agents

Zejian Zhou
University of Wyoming, Laramie, WY 82071
zzhou2@uwyo.edu

***Abstract*—We propose a fast and scalable framework for multiagent decision-making that eliminates the need for iterative fictitious play in large-scale systems. Leveraging the Fokker-Planck-Kolmogorov (FPK) equation, our method computes both the population-level control policy (drift) and the corresponding agent distribution in a single forward pass. This one-shot solution avoids the repeated equilibrium updates required by conventional continuous mean field control (MFC) approaches. Once the population dynamics are established, individual heterogeneous agents derive their actions via deep behavior cloning from the learned drift field. Our framework introduces two core innovations: (1) direct, non-iterative computation of the team policy and distribution, and (2) an imitation mechanism that seamlessly handles agent heterogeneity, capabilities not afforded by traditional MFC methods. Experimental results show that our approach achieves precise distribution control with large-scale heterogeneous agents.**

***Index Terms*—mean field control, multi-agent systems, scalable decision-making, behavior cloning.**

I. Introduction

Embedding AI into physical systems is a critical step toward achieving artificial general intelligence (AGI) [1], [2]. As a result, learning effective decision-making methods for agents interacting in dynamic environments plays a vital role. However, the most successful algorithms, including reinforcement learning [3], are typically designed for single-agent settings [4], [5] or systems with only a small number of agents [6]–[9]. As real-world agents increasingly operate in coordinated settings, conventional multi-agent decision-making algorithms face a significant scalability bottleneck, with computational complexity growing rapidly with team size [10]. Recent advancements in large language model (LLM) agents [11]–[13] further highlight this challenge. As LLM-based agents scale in number and sophistication, traditional multi-agent reinforcement learning (MARL) approaches become computationally intractable because of the exponential growth in complexity relative to the number of agents. This severely limits their practical deployment in large-scale coordination scenarios.

To date, relatively few efforts have directly addressed the challenges posed by very large-scale multi-agent systems. The mean field control (MFC) framework [14]–[18] has emerged as one of the most promising directions, leveraging game-theoretic insights to reduce the computational burden. The core idea of MFCs is to approximate many-to-many agent interactions with a one-to-mean interaction: each agent interacts not with every other agent individually, but with the aggregate behavior of the population [19]. The key contribution of MFCs is that they decouple decision-making complexity from team size. By reducing the original multi-agent system to an effective two-agent interaction, i.e., an agent versus the population mean, the computational cost becomes independent of the number of agents, depending instead on the size of the agent's own policy model. Despite this advantage, the adoption of MFCs in real-world AI-agent systems remains limited. One major obstacle is the inherent difficulty of solving MFCs' game-theoretic fixed-point equations, which typically require backward-forward partial differential equation (PDE) coupling and equilibrium convergence. Moreover, standard MFC frameworks impose strong assumptions on agent homogeneity and offer limited scalability to heterogeneous or high-dimensional environments. Existing approaches, including DeepMind's work [15] and [14], continue to rely on iterative fictitious play to approximate equilibrium, which is computationally intensive and slow to converge in continuous settings.

To pave the way for scalable, heterogeneous, multi-agent LLM systems, we propose a departure from strict equilibrium-seeking and instead embrace a pragmatic tradeoff: we forgo exact Nash optimality in favor of computational efficiency and heterogeneity support. Our method introduces a two-stage decision-making framework that extends beyond traditional MFC formulations.

In the first stage, we solve for a global population-level policy using a physics-informed neural network (PINN). This universal drift field acts as the control policy of a virtual expert agent whose goal is to steer the population distribution toward a desired terminal state. Unlike MFCs, we do not require fixed-point solutions or the backward Bellman equation. In the second stage, each individual agent, possibly with distinct dynamics or constraints, recovers its own policy by imitating the expert drift. This is achieved through behavior cloning from the learned drift field, allowing for decentralized and heterogeneous agent deployment without additional coordination or joint training.

The contributions of our paper can be summarized as:

- We develop a scalable decision-making framework capable of handling heterogeneous agents within a mean field game-like setup for large-scale systems.
- We eliminate the need for fictitious play by introducing a fast, forward-learning alternative that trades off strict

This work was supported by NSF #2502140 and NASA #80NSSC25M0015.

equilibrium optimality for practical scalability.
- We design a two-step architecture combining a PINN for global policy learning and behavior cloning for decentralized agent-level imitation.

II. Problem Formulation & Related Work

A. Continuous-time Multi-agent Decision Making

We consider a very large-scale population of N agents evolving in continuous time over a finite horizon $t \in [0, T]$. The state of agent $i \in \{1, \dots, N\}$ is denoted by $s_i(t) \in \mathbb{R}^d$, and its continuous dynamics follow a differential equation

$$\dot{s}_i(t) = f(s_i(t), a_i(t)) \tag{1}$$

where $a_i(t) \in \mathbb{R}^m$ is the agent's action at time t, $f : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$ is the environment's transition function. Each agent evolves independently according to its own trajectory so that the team as a whole completes a task by reaching a certain group state.

B. Multi-Agent Reinforcement Learning (MARL)

In a typical MARL scheme [6], each agent learns a policy to maximize its expected cumulative reward through interactions with an environment shared by other agents. The system is typically modeled as a Markov game, where the state of agent i at time t is denoted by $s_i(t) \in \mathbb{R}^d$, and the joint state of all agents is $s(t) = (s_1(t), \dots, s_N(t))$.

Each agent selects actions $a_i(t)$ according to a policy $\pi_i(a_i \mid s_i)$ and receives a local reward $r_i(s(t), a(t))$, where $a(t) = (a_1(t), \dots, a_N(t))$ is the joint action profile. The standard objective of agent i is to maximize its expected return:

$$V_i(\pi_i) = \mathbb{E}\left[\int_0^T r_i\big(s(t), a(t)\big)\, dt\right], \tag{2}$$

where the joint policy $\pi = (\pi_1, \dots, \pi_N)$ induces a distribution over state-action trajectories. Learning proceeds by estimating gradients of the return with respect to each agent's policy parameters, often via actor-critic or policy gradient methods.

This approach typically requires solving a coupled learning problem in which each agent must adapt its policy in response to the evolving behavior of other agents. In large-scale systems, this results in high variance, poor sample efficiency, and coordination challenges, especially when the agent size N goes to infinity.

C. Mean Field Control (MFCs)

Mean Field Control (MFCs) [15], [20], [21] provide a tractable approximation to large-population multi-agent systems by considering the limit as the number of agents $N \to \infty$. Instead of reasoning about each individual agent's interactions, MFC theory models the interaction between a single representative agent and the aggregate population distribution [22]. This reduces the multi-agent problem to a pair of coupled equations describing the agent's optimal control and the evolution of the population [23].

MFCs provide a scalable and principled way to model large populations under decentralized decision-making. Classical MFC solutions [15] use fictitious play (FP) to solve the two equations asynchronously. However, FP is computationally expensive and difficult to converge [24], [25], and traditional MFC cannot handle heterogeneous agents due to the constraints imposed by the FPK equation.

D. Imitation Learning

Imitation learning (IL) enables agents to acquire policies by mimicking expert behavior, bypassing the need for environment-driven exploration [26]. Among IL paradigms, behavior cloning (BC) is the most straightforward and widely used approach. BC treats policy learning as a supervised learning problem, where the agent minimizes the discrepancy between its predicted actions and those taken by an expert in demonstration data [27]. This method has proven effective in robotics [28], and manipulation [29], particularly in low-noise or well-distributed datasets. Recent developments extend BC to multi-agent settings by learning decentralized policies that imitate a centralized expert or global behavior fields, often derived from simulation or planning [30]. In such systems, BC is appealing because it allows scalable offline training, avoids joint optimization across agents, and enables flexible transfer to heterogeneous agents when combined with population-level priors.

III. Fast Mean Field Control - Drift Imitation (DI)

We propose the novel Drift Imitation (DI) algorithm to learn the group policy for large-scale MAS. Rather than solving for equilibrium, our method directly trains a PINN to produce a drift vector field that drives a system of agents from an initial state distribution to a desired final distribution. The learned drift acts as an expert so that individual agents can imitate it regardless of their own dynamics.

A. Drift Learning Driven by the FPK Equation

We begin with the Fokker-Planck-Kolmogorov (FPK) equation from MFC, which describes the evolution of the team PDF $\mu(s, t)$:

$$\frac{\partial \mu(s,t)}{\partial t} = -\nabla \cdot (f(s,t)\, \mu(s,t)) + D\nabla^2 \mu(s,t), \tag{3}$$

where s is the state, $f(s, t)$ is the drift vector field, D is the diffusion coefficient, and ∇^2 is the Laplacian operator. Given the initial distribution $\mu_0(s)$, the goal is to find a proper drift $f^*(s, a)$ so that the target distribution $\mu_T(x)$ can be reached.

We frame our drift learning approach as a PINN tailored to FPK dynamics. Rather than optimizing agent-level rewards or solving for equilibrium, we learn a time-dependent drift field that governs the evolution of a population distribution.

The solution of the FPK equation is the PDF time evolution $\mu(s, t)$. We represent both the density $\mu(s, t)$ and drift $f(s, t)$ using neural networks $\hat{\mu}_{\theta_\mu}$ and $\hat{f}_{\theta_f}$. For each collocation

point (s,t) in the domain, we use automatic differentiation to compute the PDE residual using Eq. 3:

$$\mathcal{R}_\theta(s,t) = \frac{\partial \hat{\mu}_{\theta_\mu}(s,t)}{\partial t} + \nabla \cdot \left(\hat{f}_{\theta_f}(s,t)\, \hat{\mu}_{\theta_\mu}(s,t) \right) - D\nabla^2 \hat{\mu}_{\theta_\mu}(s,t). \tag{4}$$

The loss function penalizes this residual in the least-squares sense:

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{R}_\theta(x_i, t_i) \right)^2. \tag{5}$$

We additionally enforce boundary conditions by penalizing deviation from the initial and final distributions:

$$\mathcal{L}_{\text{BC}} = \text{MSE}\left(\hat{\mu}_{\theta_\mu}(s,0), \mu_0(x) \right) + \text{MSE}\left(\hat{\mu}_{\theta_\mu}(s,t), \mu_T(x) \right). \tag{6}$$

The total training loss is given by:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PDE}} + \lambda \mathcal{L}_{\text{BC}}, \tag{7}$$

where λ controls the weight of the boundary condition loss.

This residual-based PINN formulation allows the model to learn drift fields that are physically consistent with the FPK dynamics, while simultaneously satisfying the boundary conditions defined by the initial and target densities.

A vanilla PINN often struggles with higher-dimensional FPK problems, exhibiting erratic gradients, loss of probability mass across boundaries, and deviations from prescribed boundary conditions. To address these challenges, we introduce an integral-based boundary constraint, known as the *zero-flux condition*, to enforce strict mass conservation. Specifically, we penalize the flux of probability mass across the domain boundary, yielding the boundary loss term:

$$\mathcal{L}_{\text{flux}} = \int_0^T \int_{\partial\Omega_b} \left| \begin{pmatrix} \hat{f}_{\theta_f}(s,t) \hat{\mu}_{\theta_\mu}(s,t) \\ -D\, \nabla \hat{\mu}_{\theta_\mu}(s,t) \end{pmatrix} \cdot n(s) \right|^2 dS\, dt, \tag{8}$$

where $\hat{f}_{\theta_f}(s,t)$ denotes the neural drift vector at state s and time t, $\hat{\mu}_{\theta_\mu}(s,t)$ is the learned probability density, $\partial\Omega_b$ is the spatial boundary of the domain $\Omega_b \subset \mathbb{R}^d$, and $n(x)$ denotes the outward-pointing normal vector at boundary point $s \in \partial\Omega$. Intuitively, this zero-flux boundary term ensures that the net probability flow across domain boundaries is minimized, thereby preventing unintended leakage of mass and guaranteeing that the learned solution adheres strictly to physical conservation laws. Consequently, this modification significantly stabilizes training and yields more accurate solutions, particularly in higher-dimensional scenarios.

To ensure that the learned PDF $\hat{\mu}_{\theta_\mu}(s,t)$ remains a valid probability distribution throughout its evolution, we explicitly enforce normalization at each discrete time step. Specifically, at every time step t, we normalize the PDF by dividing it by its integral over the entire spatial domain $\Omega \subseteq \mathbb{R}^d$:

$$\hat{\mu}_{\theta_\mu}(s,t) \;\leftarrow\; \frac{\hat{\mu}_{\theta_\mu}(s,t)}{\int_\Omega \hat{\mu}_{\theta_\mu}(s,t)\, dx}. \tag{9}$$

This normalization step is critical to maintain total probability mass equal to one, effectively preventing numerical drift in mass that may arise due to approximation errors, discretization artifacts, or gradient-based training instabilities. Consequently, it stabilizes the training and guarantees that the solution remains physically consistent at all times.

The final loss function is

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{PDE}} + \lambda \mathcal{L}_{\text{BC}} + \beta \mathcal{L}_{\text{flux}}, \tag{10}$$

where α, β, and λ are weighting coefficients. This loss encourages the learned drift to generate population flows that visually and numerically align with the desired mass transfer. It enables the model to learn dynamics without solving a coupled PDE system or coordinating multiple agents. The drift learning algorithm is summarized in Algorithm 1.

B. Behavior Cloning as Imitation Learning for Heterogeneous Agents

Once the population-level drift field $\hat{f}_{\theta_f}(s,t)$ is learned, individual agents with heterogeneous systems can independently extract local policies by imitating the population-level drift field as an expert policy. This avoids the need for equilibrium computation, reward design, or agent-specific optimization.

Let agent i have state $s_i(t)$ and belong to a class or type $\tau_i \in \mathcal{T}$ representing its heterogeneous attributes (e.g., dynamics, control constraints, or goals). The goal is to construct a policy $\hat{\pi}_\delta^{\tau_i}(s_i,t)$ for each agent type τ_i that approximates the shared population drift:

$$a_i(t) = \pi_\delta(s_i(t),t), s.t., f(s_i(t),a_i) \approx \hat{f}_{\theta_f}(s_i(t),t), \tag{11}$$

where the target drift $\hat{f}_{\theta_f}(s,t)$ encodes the population-wide transport strategy.

Each agent type τ is equipped with its own policy network $\pi_\delta^\tau(s,t)$. Using a dataset $\mathcal{D}^\tau = \{(s_k, t_k, \hat{f}_{\theta_f}(s_k,t_k))\}_{k=1}^N$ sampled from the learned drift, we train π_δ^τ via supervised behavior cloning:

$$\mathcal{L}_{\text{BC}}^\tau(\delta) = \frac{1}{N} \sum_{k=1}^{N} \left\| f(x_i, \pi_\delta^\tau(s_k,t_k)) - \hat{f}_{\theta_f}(s_k,t_k) \right\|_2^2. \tag{12}$$

This approach allows each agent class to adapt its control outputs to match the global drift field while respecting type-specific parameterization.

Remark 1. Relation to Classical Mean Field Control.: Although our formulation is inspired by the Fokker–Planck equation commonly used in mean field control, the proposed DI framework is not a classical MFC method in the strict game-theoretic sense. Classical MFC seeks a control law that is optimal for a representative agent while remaining consistent with the population evolution, which typically requires solving a coupled forward–backward system or an equilibrium fixed point. In contrast, our method directly learns a population-level drift field that transports the density from a prescribed initial distribution to a desired terminal distribution, without solving the backward HJB equation or enforcing fixed-point optimality. Therefore, DI should be viewed as a population-level transport and imitation framework inspired by mean-field modeling, rather than a traditional equilibrium-based MFC solver.

Algorithm 1 Drift Learning via FPK-PINN

Require: Initial PDF $\mu_0(x)$, target PDF $\mu_T(x)$, domain Ω, diffusion coefficient D, time horizon $[0, T]$

Require: Neural networks: density $\hat{\mu}_{\theta_\mu}(s,t)$, drift $\hat{f}_{\theta_f}(s,t)$

Initialize neural network parameters θ, θ_f

for $i = 1, 2, \ldots, N$ **do**

Sample minibatch of points $(s_j, t_j) \in \Omega \times [0, T]$, $j = 1, \ldots, N_b$

Compute predictions: $\hat{\mu}_j = \hat{\mu}_{\theta_\mu}(s_j, t_j)$, $\hat{f}_j = \hat{f}_{\theta_f}(s_j, t_j)$

Compute PDE residual via automatic differentiation.

Evaluate boundary condition loss:

$$\mathcal{L}_{\text{BC}} = MSE(\hat{\mu}_{\theta_\mu}(\cdot, 0), \mu_0) + MSE(\hat{\mu}_{\theta_\mu}(\cdot, T), \mu_T)$$

Evaluate zero-flux boundary loss $\mathcal{L}_{\text{flux}}$.

Compute total loss:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{PDE}} + \lambda\mathcal{L}_{\text{BC}} + \beta\mathcal{L}_{\text{flux}}$$

Update neural network parameters via gradient descent:

$$\theta_\mu, \theta_f \leftarrow \theta_\mu, \theta_f - \eta\nabla_{\theta_\mu, \theta_f}\mathcal{L}_{\text{total}}$$

Normalize PDF at each sampled timestep:

end for

return Learned drift network $\hat{f}_{\theta_f}(s,t)$ and density network $\hat{\mu}_{\theta_\mu}(s,t)$

C. Optimality and Computational Tradeoffs

A key motivation for our framework is its ability to handle heterogeneous agents without requiring centralized training or coordination. In many real-world multi-agent systems, agents differ in their dynamics, sensing capabilities, control constraints, or even objectives. Traditional mean field control (MFC) formulations assume homogeneous or nearly identical agents and struggle to generalize when agent types diverge significantly. In contrast, our method learns a population-level drift field that any agent, regardless of type, can imitate via behavior cloning. This decouples agent-level policy learning from population-level planning and enables scalable, decentralized execution across diverse agent classes.

While our approach does not aim to compute a Nash equilibrium, it provides a practical alternative for coordinating large populations. Classical MFCs define equilibrium through fixed-point consistency between optimal control and distribution evolution, often requiring backward PDEs or iterative fictitious play. These procedures are computationally intensive and sensitive to initialization and convergence. Our method bypasses these challenges by directly learning a time-varying drift that transports the population from an initial to a target distribution, without relying on best-response dynamics.

IV. Experiments

A. Drift Imitation in 1D Population Transport

We follow the evaluation strategy from DeepMind's research [15], adapting it to a continuous-time, continuous-space mean field decision-making environment. Our goal is to test the accuracy and stability of the learned population-level drift field in transporting a probability density from a given initial distribution to a desired terminal distribution. This setup isolates the performance of the drift-learning stage and serves as a basis for validating the PINN-based Fokker-Planck training procedure.

Environment Setup: We first consider a one-dimensional spatial domain $x \in [-5, 5] \subset \mathbb{R}$. The system evolves over a finite time horizon $t \in [0, 1]$. The initial distribution $\mu_0(x)$ is chosen as a binomial distribution augmented by two Gaussian distributions with means at -2.0 and 2.0. The target distribution $\mu_T(x)$ is chosen as a normalized Gaussian mixture. The objective is to learn a deterministic policy for the agents such that the team distribution $\mu(s,t)$ evolves to the target PDF.

Drift Learning: We train a PINN $\hat{f}_{\theta_f}$ to minimize the residual of the FPK equation as well as the initial and terminal conditions defined by the initial and target distributions.

We first plot the learned distribution evolution in Figure 1, which shows that the distribution network $\hat{\mu}_{\theta_\mu}(s,t)$ accurately captures the predicted PDF evolution. To evaluate the PINN's approximation performance, we use a finite-difference solution as a baseline, and the error $|\mu_{baseline} - \hat{\mu}_{\theta_\mu}|$ is plotted in Figure 3. Note that we substitute the drift network's output $\hat{f}_{\theta_f}(s,t)$ into the FPK equation to construct the finite-difference baseline.

Imitation Learning: After learning the drift field $\hat{f}_{\theta_f}(s,t)$, we test its usability by deploying it as an expert policy for heterogeneous agents. Each agent samples its own trajectory and uses supervised regression to clone the drift. This step evaluates the practicality of our method for enabling distributed, decentralized behavior learning without requiring equilibrium computations.

We used system dynamics similar to those in [15], i.e., $\dot{s}(t) = ks(t) + a(s)$ with $k = 0.5$. To simulate a large-scale MAS, we employed 1000 agents to imitate the expert drift provided by $\hat{f}_{\theta_f}(s,t)$, and the PDF estimated from the agents' trajectories is shown in Figure 2.

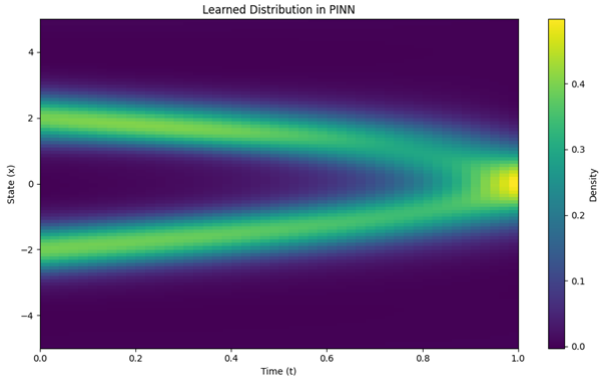

Fig. 1: The PDF evolution predicted by the PINN.

1) Mean Field Control Comparison: We acknowledge that the developed algorithm sacrifices optimality for computational simplicity and tolerance for heterogeneous agents. To

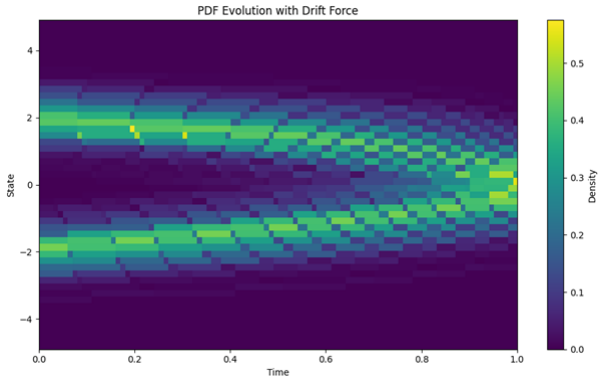

Fig. 2: The PDF evolution of real agents.

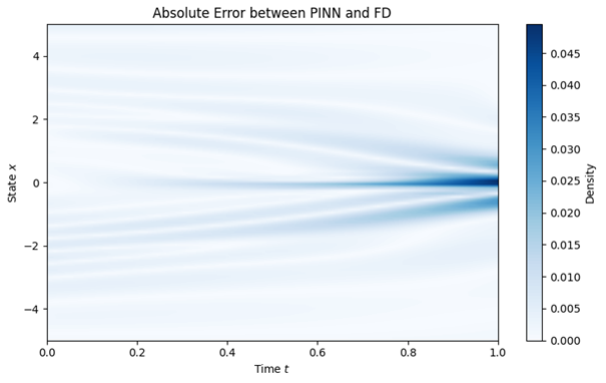

Fig. 3: PINN's approximation error compared to finite difference baseline.

show how much optimality has been sacrificed, we set another MFC baseline similar to [15], where each agent's reward has two parts: reducing the distance from the target distribution mean and reducing the distance from the team mean. We calculated the accumulated reward for both methods and list the results in Table I. Note that we used similar NN structures for PINN, behavior cloning, and MFC. Thus, the computational complexity can be compared by the number of times each neural network is trained. From Table I, we can see that although our algorithm incurs a 13.59% optimality loss, it trains three NNs only once, whereas traditional MFC with fictitious play must train two NNs across 10 iterations.

2) Heterogeneous Agents: Another contribution of DI is its ability to handle heterogeneous agents. We tested a variety of linear and nonlinear agents by comparing their trajectory differences with the expert's trajectory. The MSE between each agent type and the expert agent is reported in Table II.

TABLE I: Reward comparison with MFC baseline.

MFC reward sum	DI reward sum	Decrement
-1.8698	-2.1240	13.59%

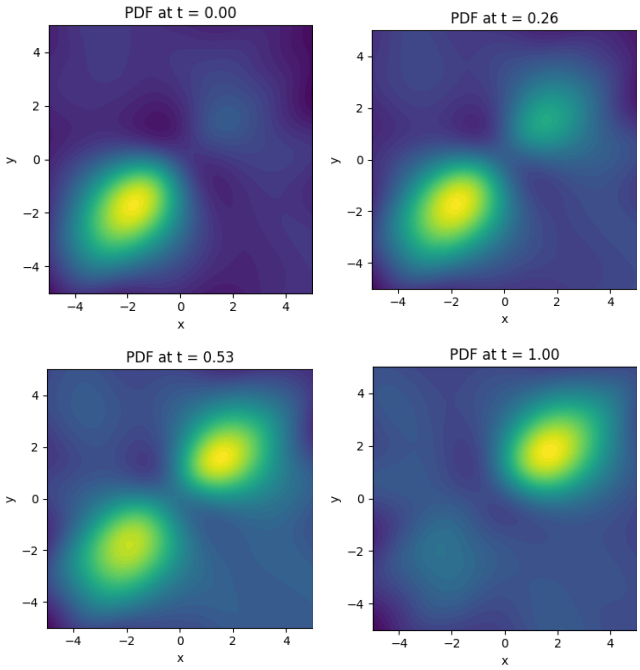

Fig. 4: Target PDF in a square shape.

TABLE II: Imitation error for heterogeneous agents.

Agent system	$\dot{x} = x + u$	$\dot{x} = \cos(x) + u$
Imitation error	0.0179	0.00896
Agent system	$\dot{x} = x^2 + u$	$\dot{x} = x^3 + u$
Imitation error	0.0479	0.00426
Agent system	$\dot{x} = \tanh(x) + u$	
Imitation error	0.000664	

B. Robotic Swarm Navigation in Continuous Space

Similar to [15], we next evaluate the applicability of our framework to a robotic swarm navigation task, where a population of agents must collectively migrate from an initial spatial configuration to a desired target formation. Unlike the previous setup, this scenario directly reflects real-world constraints of physical systems, including spatial continuity, decentralized control, and agent heterogeneity.

Environment Setup: We simulate a population of robotic agents moving in a 2D continuous domain $(s_1, s_2) \in [-5, 5]^2$. The collective state of the swarm is represented as a time-varying density $\mu(s_1, s_2, t)$, initialized from a 2D Gaussian distribution centered at $(-2, 2)$ and driven toward another distribution.

Similarly, a neural drift model $\hat{f}_{\theta_f}(s_1, s_2, t)$ is trained via residual minimization of the 2D Fokker–Planck equation. A supervision signal is constructed by interpolating between the initial and final PDFs, and a smoothing kernel is applied to stabilize the target distributions. The learned drift generalizes to the full domain and can be queried continuously.

We start the simulation with a 2D Gaussian distribution centered on $(2, 2)$ as the target distribution. The PDF evolution is plotted in Figure 4.

TABLE III: 2D PINN approximation error.

Distribution	Gaussian	Bi-Gaussian
Error	0.000143	0.000158
Distribution	Box	Ring
Error	0.00138	0.0000898

V. Conclusions and Future Work

We introduced a scalable MAS decision-making framework that leverages a PINN to directly solve the population-level drift field in MFC-type problems. Unlike traditional iterative equilibrium approaches, our Drift Imitation (DI) method produces a one-shot solution driven by the FPK equation. By providing a global drift field as a virtual expert, our approach naturally supports decentralized imitation learning by heterogeneous agents without requiring iterative coordination or reward specification. The results show that DI achieves precise distribution control with significantly reduced computational overhead, while still enabling effective policy learning across diverse agent types. These benefits make DI particularly well-suited for large-scale, real-time multi-agent systems, such as robotic swarms, autonomous fleets, or LLM-based agent populations. However, the developed DI algorithm explicitly trades off exact Nash equilibrium optimality for computational speed and structural flexibility. While this tradeoff is acceptable in many practical settings, it may limit applicability in domains where theoretical optimality and game-theoretic consistency are essential. In future work, we plan to address these limitations by exploring ways to recover approximate optimality guarantees.

References

[1] G. Paolo, J. Gonzalez-Billandon, and B. Kégl, "Position: A call for embodied AI," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., vol. 235. PMLR, 21–27 Jul 2024, pp. 39 493–39 508. [Online]. Available: https://proceedings.mlr.press/v235/paolo24a.html

[2] S. Li, K. Wu, C. Zhang, and Y. Zhu, "On the learning mechanisms in physical reasoning," in *NeurIPS*, 2022.

[3] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[4] J. Farebrother, J. Orbay, Q. Vuong, A. A. Taïga, Y. Chebotar, T. Xiao, A. Irpan, S. Levine, P. S. Castro, A. Faust *et al.*, "Stop regressing: Training value functions via classification for scalable deep rl," *arXiv preprint arXiv:2403.03950*, 2024.

[5] B. S. Pavse, M. Zurek, Y. Chen, Q. Xie, and J. P. Hanna, "Learning to stabilize online reinforcement learning in unbounded state spaces," *arXiv preprint arXiv:2306.01896*, 2023.

[6] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[7] M. Wen, J. Kuba, R. Lin, W. Zhang, Y. Wen, J. Wang, and Y. Yang, "Multi-agent reinforcement learning is a sequence modeling problem," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 509–16 521, 2022.

[8] T. Phan, F. Ritz, P. Altmann, M. Zorn, J. Nüßlein, M. Kölle, T. Gabor, and C. Linnhoff-Popien, "Attention-based recurrence for multi-agent reinforcement learning under stochastic partial observability," in *International Conference on Machine Learning*. PMLR, 2023, pp. 27 840–27 853.

[9] J. Hu, Y. Sun, H. Chen, S. Huang, Y. Chang, L. Sun *et al.*, "Distributional reward estimation for effective multi-agent deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 619–12 632, 2022.

[10] K. Gogineni, P. Wei, T. Lan, and G. Venkataramani, "Scalability bottlenecks in multi-agent reinforcement learning systems," *arXiv preprint arXiv:2302.05007*, 2023.

[11] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges." in *33rd International Joint Conference on Artificial Intelligence (IJCAI 2024)*. IJCAI; Cornell arxiv, 2024.

[12] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang, "A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges," *Vicinagearth*, vol. 1, no. 1, p. 9, 2024.

[13] X. Guo, K. Huang, J. Liu, W. Fan, N. Vélez, Q. Wu, H. Wang, T. L. Griffiths, and M. Wang, "Embodied llm agents learn to cooperate in organized teams," in *Language Gamification-NeurIPS 2024 Workshop*.

[14] X. Guo, A. Hu, R. Xu, and J. Zhang, "Learning mean-field games," *Advances in neural information processing systems*, vol. 32, 2019.

[15] S. Perrin, J. Pérolat, M. Laurière, M. Geist, R. Elie, and O. Pietquin, "Fictitious play for mean field games: Continuous time analysis and applications," *Advances in neural information processing systems*, vol. 33, pp. 13 199–13 213, 2020.

[16] M. Lauriere, S. Perrin, S. Girgin, P. Muller, A. Jain, T. Cabannes, G. Piliouras, J. Pérolat, R. Elie, O. Pietquin *et al.*, "Scalable deep reinforcement learning algorithms for mean field games," in *International conference on machine learning*. PMLR, 2022, pp. 12 078–12 095.

[17] Y. Guan, M. Afshari, and P. Tsiotras, "Zero-sum games between mean-field teams: Reachability-based analysis under mean-field sharing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 9, 2024, pp. 9731–9739.

[18] Y. Eich, C. Fabian, K. Cui, and H. Koeppl, "Bounded rationality equilibrium learning in mean field games," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 13, 2025, pp. 13 796–13 804.

[19] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 5571–5580.

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

[21] G. Fu, S. Liu, S. Osher, and W. Li, "High order computation of optimal transport, mean field planning, and potential mean field games," *Journal of Computational Physics*, vol. 491, p. 112346, 2023.

[22] M. Huang, P. E. Caines, and R. P. Malhamé, "Large-population cost-coupled lqg problems with nonuniform agents: Individual-mass behavior and decentralized nash equilibria," *IEEE transactions on automatic control*, vol. 52, no. 9, pp. 1560–1571, 2007.

[23] A. Bensoussan, J. Frehse, P. Yam *et al.*, *Mean field games and mean field type control theory*. Springer, 2013, vol. 101.

[24] P. Van Der Vaart, A. Mahajan, and S. Whiteson, "Model based multi-agent reinforcement learning with tensor decompositions," *arXiv preprint arXiv:2110.14524*, 2021.

[25] M. Bowling, "Convergence and no-regret in multiagent learning," *Advances in neural information processing systems*, vol. 17, 2004.

[26] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[27] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NeurIPS*, 1989, pp. 305–313.

[28] S. Ross, G. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011, pp. 627–635.

[29] E. Jang, A. Srinivas, M. Laskin, K. Lee, P. Abbeel, and S. Levine, "Bc-z: Zero-shot task generalization with robotic imitation learning," *arXiv preprint arXiv:2203.16876*, 2022.

[30] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *AAMAS*, 2017, pp. 66–83.