Mean Field Games Flock! The Reinforcement Learning Way

Sarah Perrin[1] , **Mathieu Laurière**[2] , **Julien Pérolat**[3] , **Matthieu Geist**[4] , **Romuald Élie**[3] and **Olivier Pietquin**[4]
[1]Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL
[2]Princeton University, ORFE
[3]DeepMind Paris
[4]Google Research, Brain Team
sarah.perrin@inria.fr, lauriere@princeton.edu, {perolat, mfgeist, relie, pietquin}@google.com,

Abstract

We present a method enabling a large number of agents to learn how to flock. This problem has drawn a lot of interest but requires many structural assumptions and is tractable only in small dimensions. We phrase this problem as a Mean Field Game (MFG), where each individual chooses its own acceleration depending on the population behavior. Combining Deep Reinforcement Learning (RL) and Normalizing Flows (NF), we obtain a tractable solution requiring only very weak assumptions. Our algorithm finds a Nash Equilibrium and the agents adapt their velocity to match the neighboring flock's average one. We use Fictitious Play and alternate: (1) computing an approximate best response with Deep RL, and (2) estimating the next population distribution with NF. We show numerically that our algorithm can learn multi-group or high-dimensional flocking with obstacles.

1 Introduction

The term flocking describes the behavior of large populations of birds that fly in compact groups with similar velocities, often exhibiting elegant motion patterns in the sky. Such behavior is pervasive in the animal realm, from fish to birds, bees or ants. This intriguing property has been widely studied in the scientific literature [Shaw, 1975; Reynolds, 1987] and its modeling finds applications in psychology, animation, social science, or swarm robotics. One of the most popular approaches to model flocking was proposed in [Cucker and Smale, 2007] and allows predicting the evolution of each agent's velocity from the speeds of its neighbors.

To go beyond pure description of population behaviours and emphasize on the decentralized aspect of the underlying decision making process, this model has been revisited to integrate an optimal control perspective, see *e.g.* [Caponigro *et al.*, 2013; Bailo *et al.*, 2018]. Each agent controls its velocity and hence its position by dynamically adapting its acceleration so as to maximize a reward that depends on the others' behavior. An important question is a proper understanding of the nature of the equilibrium reached by the population of agents, emphasizing how a consensus can be reached in a group without centralized decisions. Such question is often studied using the notion of Nash equilibrium and becomes extremely complex when the number of agents grows.

A way to approximate Nash equilibria in large games is to study the limit case of an continuum of identical agents, in which the local effect of each agent becomes negligible. This is the basis of the Mean Field Games (MFGs) paradigm introduced in [Lasry and Lions, 2007]. MFGs have found numerous applications from economics to energy production and engineering. A canonical example is crowd motion modeling in which pedestrians want to move while avoiding congestion effects. In flocking, the purpose is different since the agents intend to remain together as a group, but the mean-field approximation can still be used to mitigate the complexity.

However, finding an equilibrium in MFGs is computationally intractable when the state space exceeds a few dimensions. In traditional flocking models, each agent's state is described by a position and a velocity, while the control is the acceleration. In terms of computational cost, this typically rules out state-of-the-art numerical techniques for MFGs based on finite difference schemes for partial differential equations (PDEs) [Achdou and Capuzzo-Dolcetta, 2010]. In addition, PDEs are in general hard to solve when the geometry is complex and require full knowledge of the model.

For these reasons, Reinforcement Learning (RL) to learn control strategies for MFGs has recently gained in popularity [Guo *et al.*, 2019; Elie *et al.*, 2020; Perrin *et al.*, 2020]. Combined with deep neural nets, RL has been used successfully to tackle problems which are too complex to be solved by exact methods [Silver *et al.*, 2018] or to address learning in multi-agent systems [Lanctot *et al.*, 2017]. Particularly relevant in our context, are works providing techniques to compute an optimal policy [Haarnoja *et al.*, 2018; Lillicrap *et al.*, 2016] and methods to approximate probability distributions in high dimension [Rezende and Mohamed, 2015; Kobyzev *et al.*, 2020].

Our **main contributions** are: **(1)** we cast the flocking problem into a MFG and propose variations which allow multi-group flocking as well as flocking in high dimension with complex topologies, **(2)** we introduce the Flock'n RL algorithm that builds upon the Fictitious Play paradigm and involves deep neural networks and RL to solve the model-free flocking MFG, and **(3)** we illustrate our approach on several numerical examples and evaluate the solution with approximate performance matrix and exploitability.

2 Background

Three main formalisms will be combined: flocking, Mean Field Games (MFG), Reinforcement Learning (RL). $\mathbb{E}_z$ stands for the expectation w.r.t. the random variable z.

2.1 The Model of Flocking

To model a flocking behaviour, we consider the following system of N agents derived in a discrete time setting in [Nourian *et al.*, 2011] from Cucker-Smale flocking modeling. Each agent i has a position and a velocity, each in dimension d and denoted respectively by x^i and v^i. We assume that it can control its velocity by choosing the acceleration, denoted by u^i. The dynamics of agent $i \in \{1, \dots, N\}$ is:

$$x^i_{t+1} = x^i_t + v^i_t \Delta t, \qquad v^i_{t+1} = v^i_t + u^i_t \Delta t + \epsilon^i_{t+1},$$

where Δt is the time step and ϵ^i_t is a random variable playing the role of a random disturbance. We assume that each agent is optimizing for a *flocking criterion* f that is underlying to the flocking behaviour. For agent i at time t, f is of the form:

$$f^i_t = f(x^i_t, v^i_t, u^i_t, \mu^N_t), \tag{1}$$

where the interactions with other agents are only through the empirical distribution of states and velocities denoted by: $\mu^N_t = \frac{1}{N}\sum_{j=1}^N \delta_{(x^j_t, v^j_t)}$.

We focus on criteria incorporating a term of the form:

$$f^{\text{flock}}_\beta(x, v, u, \mu) = -\left\| \int_{\mathbb{R}^{2d}} \frac{(v - v')}{(1 + \|x - x'\|^2)^\beta} \, d\mu(x', v') \right\|^2, \tag{2}$$

where $\beta \geq 0$ is a parameter and $(x, v, u, \mu) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d \times \mathbb{R}^d)$, with $\mathcal{P}(E)$ denoting the set of probability measures on a set E. This criterion incentivises agents to align their velocities, especially if they are close to each other. Note that β parameterizes the level of interactions between agents and strongly impacts the flocking behavior: if $\beta = 0$, each agent tries to align its velocity with all the other agents of the population irrespective of their positions, whereas the larger $\beta > 0$, the more importance is given to its closest neighbors (in terms of position).

In the N-agent case, for agent i, it becomes:

$$f^{\text{flock},i}_{\beta,t} = -\left\| \frac{1}{N}\sum_{j=1}^N \frac{(v^i_t - v^j_t)}{(1 + \|x^i_t - x^j_t\|^2)^\beta} \right\|^2. \tag{3}$$

The actual criterion will typically include other terms, for instance to discourage agents from using a very large acceleration, or to encourage them to be close to a specific position. We provide such examples in Sec. 4.

Since the agents may be considered as selfish (they try to maximize their own criterion) and may have conflicting goals (*e.g.* different desired velocities), we consider Nash equilibrium as a notion of solution to this problem and the individual criterion can be seen as the payoff for each agent. The total payoff of agent i given the other agents' strategies $u^{-i} = (u^1, \dots, u^{i-1}, u^{i+1}, \dots, u^N)$ is: $F^i_{u^{-i}}(u^i) = \mathbb{E}_{x^i_t, v^i_t}\left[\sum_{t\geq 0} \gamma^t f^i_t\right]$, with f^i_t defined Eq. (1). In this context, a *Nash equilibrium* is a strategy profile $(\hat{u}^1, \hat{u}^2, \dots, \hat{u}^N)$ such that there's no profitable unilateral deviation, *i.e.*, for every $i = 1, \dots, N$, for every control u^i, $F^i_{\hat{u}^{-i}}(\hat{u}^i) \geq F^i_{\hat{u}^{-i}}(u^i)$.

2.2 Mean Field Games

An MFG describes a game for a continuum of identical agents and is fully characterized by the dynamics and the *payoff* function of a representative agent. More precisely, denoting by μ_t the state distribution of the population, and by $\xi_t \in \mathbb{R}^\ell$ and $\alpha_t \in \mathbb{R}^k$ the state and the control of an infinitesimal agent, the dynamics of the infinitesimal agent is given by

$$\xi_{t+1} = \xi_t + b(\xi_t, \alpha_t, \mu_t) + \sigma \epsilon_{t+1}, \tag{4}$$

where $b : \mathbb{R}^\ell \times \mathbb{R}^k \times \mathcal{P}(\mathbb{R}^\ell) \to \mathbb{R}^\ell$ is a drift (or transition) function, σ is a $\ell \times \ell$ matrix and ϵ_{t+1} is a noise term taking values in $\mathbb{R}^\ell$. We assume that the sequence of noises $(\epsilon_t)_{t\geq 0}$ is i.i.d. (*e.g.* Gaussian). The objective of each infinitesimal agent is to maximize its total expected payoff, defined given a flow of distributions $\mu = (\mu_t)_{t\geq 0}$ and a strategy α (*i.e.*, a stochastic process adapted to the filtration generated by $(\epsilon_t)_{t\geq 0}$) as: $J_\mu(\alpha) = \mathbb{E}_{\xi_t, \alpha_t}\left[\sum_{t\geq 0} \gamma^t \varphi(\xi_t, \alpha_t, \mu_t)\right]$, where $\gamma \in (0, 1)$ is a discount factor and $\varphi : \mathbb{R}^\ell \times \mathbb{R}^k \times \mathcal{P}(\mathbb{R}^\ell) \to \mathbb{R}^\ell$ is an instantaneous payoff function. Since this payoff depends on the population's state distribution, and since the other agents would also aim to maximize their payoff, a natural approach is to generalize the notion of Nash equilibrium to this framework. A *mean field (Nash) equilibrium* is defined as a pair $(\hat{\mu}, \hat{\alpha}) = (\hat{\mu}_t, \hat{\alpha}_t)_{t\geq 0}$ of a flow of distributions and strategies such that the following two conditions are satisfied: $\hat{\alpha}$ is a best response against $\hat{\mu}$ (optimality) and $\hat{\mu}$ is the distribution generated by $\hat{\alpha}$ (consistency), *i.e.*,

1. $\hat{\alpha}$ maximizes $\alpha \mapsto J_{\hat{\mu}}(\alpha)$;
2. for every $t \geq 0$, $\hat{\mu}_t$ is the distribution of ξ_t when it follows the dynamics (4) with (α_t, μ_t) replaced by $(\hat{\alpha}_t, \hat{\mu}_t)$.

Finding a mean field equilibrium thus amounts to finding a fixed point in the space of (flows of) probability distributions. The existence of equilibria can be proven through classical fixed point theorems [Carmona and Delarue, 2018]. In most mean field games considered in the literature, the equilibrium is unique, which can be proved using either a strict contraction argument or the so-called Lasry-Lions monotonicity condition [Lasry and Lions, 2007]. Computing solutions to MFGs is a challenging task, even when the state is in small dimension, due to the coupling between the optimality and the consistency conditions. This coupling typically implies that one needs to solve a forward-backward system where the forward equation describes the evolution of the distribution and the backward equation characterizes the optimal control. One can not be solved prior to the other one, which leads to numerical difficulties. The basic approach, which consists in iteratively solving each equation, works only in very restrictive settings and is otherwise subject to cycles. A method which does not suffer from this limitation is Fictitious Play, summarized in Alg. 1. It consists in computing the best response against a weighted average of past

Algorithm 1: Generic Fictitious Play in MFGs

input : MFG = $\{\xi, \varphi, \mu_0\}$; # of iterations J ;
Define $\bar{\mu}_0 = \mu_0$ **for** $j = 1, \ldots, J$ **do**
  1. Set best response $\alpha_j = \arg\max_\alpha J_{\bar{\mu}_{j-1}}(\alpha)$ and let $\bar{\alpha}_j$ be the average of $(\alpha_i)_{i=0,\ldots,j}$
  2. Set $\mu_j = \gamma$-stationary distribution induced by α_j
  3. Set $\bar{\mu}_j = \frac{j-1}{j}\bar{\mu}_{j-1} + \frac{1}{j}\mu_j$
return $\bar{\alpha}_J, \bar{\mu}_J$

distributions instead of just the last distribution. This algorithm has been shown to converge for more general MFGs. State-of-the-art numerical methods for MFGs based on partial differential equations can solve such problems with a high precision when the state is in small dimension and the geometry is elementary [Achdou and Capuzzo-Dolcetta, 2010; Carlini and Silva, 2014]. More recently, numerical methods based on machine learning tools have been developed [Carmona and Laurière, 2019; Ruthotto *et al.*, 2020]. These techniques rely on the full knowledge of the model and are restricted to classes of quite simple MFGs.

2.3 Reinforcement Learning

The Reinforcement Learning (RL) paradigm is the machine learning answer to the optimal control problem. It aims at learning an optimal policy for an agent that interacts in an environment composed of states, by performing actions. Formally, the problem is framed under the Markov Decision Processes (MDP) framework. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is a transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a *reward* function and γ is a discount factor (see Eq. (5)). Using action a when the current state is s leads to a new state distributed according to $P(s, a)$ and produces a reward $R(s, a)$. A policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, $s \mapsto \pi(\cdot|s)$ provides a distribution over actions for each state. RL aims at learning a policy π^* which maximizes the total return defined as the expected (discounted) sum of future rewards:

$$R(\pi) = \mathbb{E}_{a_t, s_{t+1}}\Big[\sum_{t\geq 0} \gamma^t r(s_t, a_t)\Big], \tag{5}$$

with $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim p(\cdot|s_t, a_t)$. Note that if the dynamics (p and r) is known to the agent, the problem can be solved using *e.g.* dynamic programming. Most of the time, these quantities are unknown and RL is required. A plethora of algorithms exist to address the RL problem. Yet, we need to focus on methods that allow continuous action spaces as we want to control accelerations. One category of such algorithms is based on the Policy Gradient (PG) theorem [Sutton *et al.*, 1999] and makes use of the gradient ascent principle: $\pi \leftarrow \pi + \alpha\frac{\partial R(\pi)}{\partial \pi}$, where α is a learning rate. Yet, PG methods are known to be high-variance because they use Monte Carlo rollouts to estimate the gradient. A vast literature thus addresses the variance reduction problem. Most of the time, it involves an hybrid architecture, namely Actor-Critic, which relies on both a representation of the policy and of the so-called state-action value function $(s, a) \mapsto Q^\pi(s, a)$. $Q^\pi(s, a)$ is the total return conditioned on starting in state s and using action a before using policy π for subsequent time steps. It can be estimated by bootstrapping, using the Markov property, through the Bellman equations. Most recent implementations rely on deep neural networks to approximate π and Q (*e.g.* [Haarnoja *et al.*, 2018]).

3 Our Approach

In this section, we put together the pieces of the puzzle to numerically solve the flocking model. Based on a mean-field approximation, we first recast the flocking model as an MFG with decentralized decision making, for which we propose a numerical method relying on RL and deep neural networks.

3.1 Flocking as an MFG

Mean field limit. We go back to the model of flocking introduced in Sec. 2.1. When the number of agents grows to infinity, the empirical distribution μ_t^N is expected to converge towards the law μ_t of (x_t, v_t), which represents the position and velocity of an infinitesimal agent and have dynamics:

$$x_{t+1} = x_t + v_t\Delta t, \qquad v_{t+1} = v_t + u_t\Delta t + \epsilon_{t+1}.$$

This problem can be viewed as an instance of the MFG framework discussed in Sec. 2.2, by taking the state to be the position-velocity pair and the action to be the acceleration, *i.e.*, the dimensions are $\ell = 2d$, $k = d$, and $\xi = (x, v)$, $\alpha = u$. To accommodate for the degeneracy of the noise as only the velocities are disturbed, we take $\sigma = \begin{pmatrix} 0_d & 0_d \\ 0_d & 1_d \end{pmatrix}$ where 1_d is the d-dimensional identity matrix.

The counterpart of the notion of N-agent Nash equilibrium is an MFG equilibrium as described in Sec. 2.2. We focus here on equilibria which are stationary in time. In other words, the goal is to find a pair $(\hat{\mu}, \hat{u})$ where $\hat{\mu} \in \mathcal{P}(\mathbb{R}^\ell \times \mathbb{R}^\ell)$ is a position-velocity distribution and $\hat{u} : \mathbb{R}^\ell \times \mathbb{R}^\ell \mapsto \mathbb{R}^\ell$ is a feedback function to determine the acceleration given the position and velocity, such that: **(1)** $\hat{\mu}$ is an invariant position-velocity distribution if the whole population uses the acceleration given by $\hat{u}$, and **(2)** $\hat{u}$ maximizes the rewards when the agent's initial position-velocity distribution is $\hat{\mu}$ and the population distribution is $\hat{\mu}$ at every time step. In mathematical terms, $\hat{u}$ maximizes $J_{\hat{\mu}}(u) = \mathbb{E}_{x_t, v_t, u_t}\Big[\sum_{t\geq 0} \gamma^t \varphi(x_t, v_t, u_t, \hat{\mu})\Big]$, where $(x_t, v_t)_{t\geq 0}$ is the trajectory of an infinitesimal agent who starts with distribution $\hat{\mu}$ at time $t = 0$ and is controlled by the acceleration $(u_t)_{t\geq 0}$. As the payoff function φ we use f_β^{flock} from Eq. (2). Moreover, the consistency condition rewrites as: $\hat{\mu}$ is the stationary distribution of $(x_t, v_t)_{t\geq 0}$ if controlled by $(\hat{u}_t)_{t\geq 0}$.

Theoretical analysis. The analysis of MFG with flocking effects is challenging due to the unusual structure of the dynamics and the payoff, which encourages gathering of the population. This is running counter to the classical Lasry-Lions monotonicity condition [Lasry and Lions, 2007], which typically penalizes the agents for being too close to each other. However, existence and uniqueness have been proved in some cases. If $\beta = 0$, every agent has the same influence over the representative agent and it is possible to show that

Algorithm 2: Flock'n RL

input : MFG = $\{(x, v), f_\beta^{\text{flock}}, \mu_0\}$; # of iterations J

1 Define $\bar{\mu}_0 = \mu_0$ **for** $j = 1, \ldots, J$ **do**
2 **1.** Set best response $\pi_j = \arg\max_\pi J_{\bar{\mu}_{j-1}}(\pi)$ with **SAC** and let $\bar{\pi}_j$ be the average of $(\pi_0, \ldots, \pi_j)$
3 **2.** Using a **Normalizing Flow**, compute $\mu_j =$ γ-stationary distribution induced by π_j
4 **3.** Using a **Normalizing Flow** and samples from $(\mu_0, \ldots, \mu_{j-1})$, estimate $\bar{\mu}_j$
5 **return** $\bar{\pi}_J, \bar{\mu}_J$

the problem reduces to a Linear-Quadratic setting. Th 2 in [Nourian *et al.*, 2011] shows that a mean-consensus in velocity is reached asymptotically with individual asymptotic variance $\frac{\sigma^2}{2}$. If $\beta > 0$, [Nourian *et al.*, 2011] shows that if the MF problem admits a unique solution, then there exists an ϵ_N Nash equilibrium for the N agents problem and $\lim_{N \to +\infty} \epsilon_N = 0$. Existence has also been proved when $\beta > 0$ in [Carmona and Delarue, 2018, Section 4.7.3], with a slight modification of the payoff, namely considering, with φ a bounded function, $r_t = -\varphi\left(\left\| \int_{\mathbb{R}^{2d}} \frac{(v_t - v')}{(1+\|x_t - x'\|^2)^\beta} \mu_t(dx', dv') \right\|^2\right)$.

3.2 The Flock'n RL Algorithm

We propose Flock'n RL, a deep RL version of the Fictitious Play algorithm for MFGs [Elie *et al.*, 2020] adapted to flocking. We consider a γ-discounted setting with continuous state and action spaces and we adapt Alg. 2 from its original tabular formulation [Perrin *et al.*, 2020]. It alternates 3 steps: (1) estimation of a best response (using Deep RL) against the mean distribution $\bar{\mu}_{j-1}$, which is fixed during the process, (2) estimation (with Normalizing Flows [Kobyzev *et al.*, 2020]) of the new induced distribution from trajectories generated by the previous policy, and (3) update of the mean distribution $\bar{\mu}_j$.

Computing the Best Response with SAC

The first step in the loop of Alg. 2 is the computation of a best response against $\bar{\mu}_j$. In fact, the problem boils down to solving an MDP in which $\bar{\mu}_j$ enters as a parameter. Following the notation introduced in Sec. 2.3, we take the state and action spaces to be respectively $\mathcal{S} = \mathbb{R}^{2d}$ (for position-velocity pairs) and $\mathcal{A} = \mathbb{R}^d$ (for accelerations). Letting $s = (x, v)$ and $a = u$, the reward is: $(x, v, u) = (s, a) \mapsto r(s, a) = f(x, v, u, \bar{\mu}_j)$, which depends on the given distribution $\bar{\mu}_j$. Remember that r is the reward function of the MDP while f is the optimization criterion in the flocking model.

As we set ourselves in continuous state and action spaces and in possibly high dimensions, we need an algorithm that scales. We choose to use Soft Actor Critic (SAC) [Haarnoja *et al.*, 2018], an off-policy actor-critic deep RL algorithm using entropy regularization. SAC is trained to maximize a trade-off between expected return and entropy, which allows to keep enough exploration during the training. It is designed to work on continuous action spaces, which makes it suited for acceleration controlled problems such as flocking.

The best response is computed against $\bar{\mu}_j$, the fixed average distribution at step j of Flock'n RL. SAC maximizes the reward which is a variant of $f_{\beta,t}^{\text{flock},i}$ from Eq. (3). It needs samples from $\bar{\mu}_j$ in order to compute the positions and velocities of the fixed population. Note that, in order to measure more easily the progress during the learning at step j, we sample N agents from $\bar{\mu}_j$ at the beginning of step 1 (*i.e.* we do not sample new agents from $\bar{\mu}_j$ every time we need to compute the reward). During the learning, at the beginning of each episode, we sample a starting state $s_0 \sim \bar{\mu}_j$.

In the experiments, we will not need $\bar{\pi}_j$ but only the associated reward (see the exploitability metric in Sec. 4). To this end, it is enough to keep in memory the past policies $(\pi_0, \ldots, \pi_j)$ and simply average the induced rewards.

Normalizing Flow for Distribution Embedding

We choose to represent the different distributions using a generative model because the continuous state space prevents us from using a tabular representation. Furthermore, even if we could choose to discretize the state space, we would need a huge amount of data points to estimate the distribution using methods such as kernel density estimators. In dimension 6 (which is the dimension of our state space with 3-dimensional positions and velocities), such methods already suffer from the curse of dimensionality.

Thus, we choose to estimate the second step of Alg. 2 using a Normalizing Flow (NF) [Rezende and Mohamed, 2015; Kobyzev *et al.*, 2020], which is a type of generative model, different from Generative Adversarial Networks (GAN) or Variational Autoencoders (VAE). A flow-based generative model is constructed by a sequence of invertible transformations and allows efficient sampling and distribution approximation. Unlike GANs and VAEs, the model explicitly learns the data distribution and therefore the loss function simply identifies to the negative log-likelihood. An NF transforms a simple distribution (*e.g.* Gaussian) into a complex one by applying a sequence of invertible transformations. In particular, a single transformation function f of noise z can be written as $x = f(z)$ where $z \sim h(z)$. Here, $h(z)$ is the noise distribution and will often be in practice a normal distribution.

Using the change of variable theorem, the probability density of x under the flow can be written as: $p(x) = h(f^{-1}(x)) \left| \det\left(\frac{\partial f^{-1}}{\partial x}\right) \right|$. We thus obtain the probability distribution of the final target variable. In practice, the transformations f and f^{-1} can be approximated by neural networks. Thus, given a dataset of observations (in our case rollouts from the current best response), the flow is trained by maximizing the total log likelihood $\sum_n \log p(x^{(n)})$.

Computation of $\bar{\mu}_j$

Due to the above discussion on the difficulty to represent the distribution in continuous space and high dimension, the third step (Line 4 of Alg. 2) can not be implemented easily. We represent every μ_j as a generative model, so we can not "average" the normalizing flows corresponding to $(\mu_i)_{i=1,\ldots,j}$ in a straightforward way but we can sample data points $x \sim \mu_i$ for each $i = 1, \ldots, j$. To have access to $\bar{\mu}_j$, we keep in memory every model μ_j, $j \in \{1, \ldots, J\}$ and, in order to sample points according to $\bar{\mu}_j$ for a fixed j, we sample points from $\mu_i, i \in \{1, \dots, j\}$, with probability $1/j$. These points are then used to learn the distribution $\bar{\mu}_j$ with an NF, as it is needed both for the reward and to sample the starting state of an agent during the process of learning a best response policy.

4 Experiments

Environment. We implemented the environment as a custom OpenAI gym environment to benefit from the powerful gym framework and use the algorithms available in stable baselines [Hill *et al.*, 2018]. We define a state $s \in S$ as $s = (x, v)$ where x and v are respectively the vectors of positions and velocities. Each coordinate x_i of the position can take any continuous value in the d-dimensional box $x_i \in [-100, +100]$, while the velocities are also continuous and clipped $v_i \in [-1, 1]$. The state space for the positions is a torus, meaning that an agent reaching the box limit reappears at the other side of the box. We chose this setting to allow the agents to perfectly align their velocities (except for the effect of the noise), as we look for a stationary solution.

At the beginning of each iteration j of Fictitious Play, we initialize a new gym environment with the current mean distribution $\bar{\mu}_j$, in order to compute the best response.

Model - Normalizing Flows. To model distributions, we use Neural Spline Flows (NSF) with a coupling layer [Durkan *et al.*, 2019]. More details about how coupling layers and NSF work can be found in the appendix.

Model - SAC. To compute the best response at each Flock'n RL iteration, we use Soft Actor Critic (SAC) [Haarnoja *et al.*, 2018] (but other PG algorithms would work). SAC is an off-policy algorithm which, as mentioned above, uses the key idea of regularization: instead of considering the objective to simply be the sum of rewards, an entropy term is added to encourage sufficient randomization of the policy and thus address the exploration-exploitation trade-off. To be specific, in our setting, given a population distribution μ, the objective is to maximize: $J_\mu(\pi) = \mathbb{E}_{(s_t,u_t)}\left[\sum_{t=0}^{+\infty} \gamma^t r(x_t, v_t, u_t, \mu_t) + \delta \mathcal{H}(\pi(\cdot|s_t))\right]$, where $\mathcal{H}$ denotes the entropy and $\delta \geq 0$ is a weight.

To implement the optimization, the SAC algorithm follows the philosophy of actor-critic by training parameterized Q-function and policy. To help convergence, the authors of SAC also train a parameterized value function V. In practice, the three functions are often approximated by neural networks.

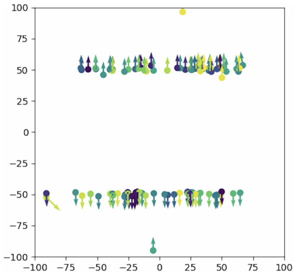
In comparison to other successful methods such as Trust Region Policy Optimization (TRPO) [Schulman *et al.*, 2015] or Asynchronous Actor-Critic Agents (A3C), SAC is expected to be more efficient in terms of number of samples required to learn the policy thanks to the use of a replay buffer in the spirit of methods such as Deep Deterministic Policy Gradient (DDPG) [Lillicrap *et al.*, 2016].

Metrics. An issue with studying our flocking model is the absence of a gold standard. Especially, we can not compute the exact exploitability [Perrin *et al.*, 2020] of a policy against a given distribution since we can not compute the exact best response. The exploitability measures how much an agent can gain by replacing its policy π with a best response π', when the rest of the population plays π: $\phi(\pi) := \max_{\pi'} J(\mu_0, \pi', \mu^\pi) - J(\mu_0, \pi, \mu^\pi)$. If $\phi(\bar{\pi}_j) \to 0$ as j increases, FP approaches a Nash equilibrium. To cope with these issues, we introduce the following ways to measure progress of the algorithm:

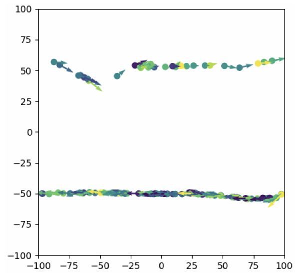
- *Performance matrix:* we build the matrix $\mathcal{M}$ of performance of learned policies versus estimated distributions. The entry $\mathcal{M}_{i,j}$ on the i-th row and the j-th column is the total γ-discounted sum of rewards: $\mathcal{M}_{i,j} = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{t,i} \mid s_0 \sim \bar{\mu}_{i-1},\ u_t \sim \pi_j(.|s_t)\right]$, where $r_{t,i} = r(s_t, u_t, \bar{\mu}_{i-1})$, obtained with π_j against $\bar{\mu}_{i-1}$. The diagonal term $\mathcal{M}_{j,j}$ corresponds to the value of the best response computed at iteration j.
- *Approximate exploitability:* We do not have access to the exact best response due to the model-free approach and the continuous spaces. However, we can approximate the first term of $\phi(\bar{\pi})$ directly in the Flock'n RL algorithm with SAC. The second term, $J(\mu_0, \bar{\pi}, \mu^{\bar{\pi}})$, can be approximated by replacing $\bar{\pi}$ with the average over past policies, *i.e.*, the policy sampled uniformly from the set $\{\pi_0, \dots, \pi_j\}$. At step j, the approximate exploitability is $e_j = \mathcal{M}_{j,j} - \frac{1}{j-1}\sum_{k=1}^{j-1} \mathcal{M}_{j,k}$. To smoothen the exploitability, we take the best response over the last 5 policies and use a moving average over 10 points. Please note that only relative values are important as it depends on the scale of the reward.

A 4-Dimensional example. We illustrate in a four dimensional setting (*i.e.* two-dimensional positions and velocities) how the agents learn to adopt similar velocities by controlling their acceleration. We focus on the role of β in the flocking effect. We consider noise $\epsilon_t^i \sim \mathcal{N}(0, \Delta t)$ and the following reward: $r_t^i = f_{\beta,t}^{\text{flock},i} - \|u_t^i\|_2^2 + \|v_t^i\|_\infty - \min\{\|x_{2,t}^i \pm 50\|\}$, where $x_{2,t}^i$ stands for the second coordinate of the i-th agent's position at time t. The last term attracts the agents' positions towards one of two lines corresponding to the second coordinate of x being either -50 or $+50$. We added a term regarding the norm of the velocity to prevent agents from stopping. Here we take $\|v_t^i\|_\infty = \max\{|v_{1,t}^i|, |v_{2,t}^i|\}$. Hence, a possible equilibrium is with two groups of agents, one for each line. When $\beta = 0$, the term $f_{\beta,t}^{\text{flock},i}$ encourages agent i to have the same velocity vector as the rest of the whole population. At equilibrium, the agents in the two groups should thus move in the same direction (to the left or to the right, in order to stay on the two lines of x's). On the other hand, when $\beta > 0$ is large enough (*e.g.* $\beta = 100$), agent i gives more importance to its neighbors when choosing its control and it tries to have a velocity similar to the agents that are position-wise close to. This allows the emergence of two groups moving in different directions: one group moves towards the left (overall negative velocity) and the other group moves towards the right (overall positive velocity).

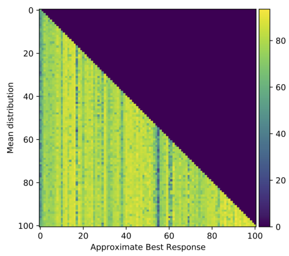
This is confirmed by Fig. 1. In the experiment, we set the initial velocities perpendicular to the desired ones to illustrate the robustness of the algorithm. We observe that the approximate exploitability globally decreases. In the case $\beta = 0$, we experimentally verified that there is always a global consen-



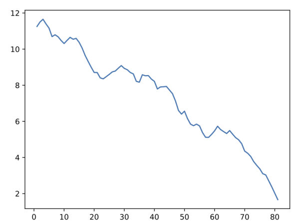
(a) Initial positions and velocities



(b) At convergence

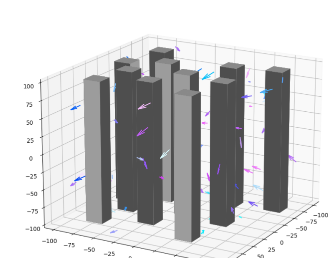


(c) Performance matrix

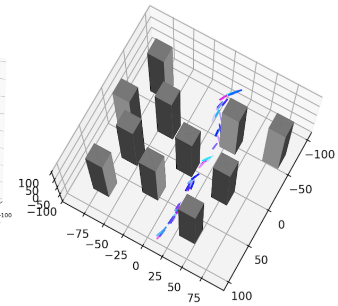


(d) Approximate exploitability

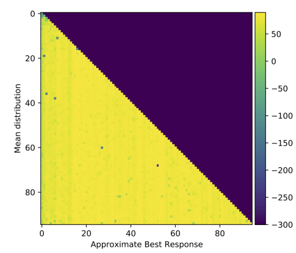
Figure 1: Multi-group flocking with noise and $\beta = 100$.



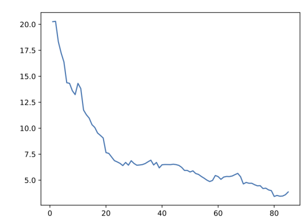
(a) Initial positions and velocities



(b) At convergence



(c) Performance matrix



(d) Approximate exploitability

Figure 2: Flocking with noise and many obstacles.

sus, *i.e.*, only one line or two lines but moving in the same direction.

Scaling to 6 dimensions and non-smooth topology. We now present an example with arbitrary obstacles (and thus non-smooth topology) in dimension 6 (position and velocity in dimension 3) which would be very hard to address with classical numerical methods. In this setting, we have multiple columns that the agents are trying to avoid. The reward has the following form: $r_t^i = f_{\beta,t}^{\text{flock},i} - \|u_t^i\|_2^2 + \|v_t^i\|_\infty - \min\{\|x_{2,t}^i\|\} - c * \mathbf{1}_{obs}$, If an agent hits an obstacle, it gets a negative reward and bounces on it like a snooker ball. After a few iterations, the agents finally find their way through the obstacles. This situation can model birds trying to fly in a city with tall buildings. In our experiments, we noticed that different random seeds lead to different solutions. This is not surprising as there are a lot of paths that the agents can take to avoid the obstacles and still maximizing the reward function. The exploitability decreases quicker than in the previous experiment. We believe that this is because agents find a way through the obstacles in the first iterations.

5 Related Work

Numerical methods for flocking models. Most work using flocking models focus on the dynamical aspect without optimization. To the best of our knowledge, the only existing numerical approach to tackle a MFG with flocking effects is in [Carmona and Delarue, 2018, Section 4.7.3], but it is restricted to a very special and simpler type of rewards.

Learning in MFGs. MFGs have attracted a surge of interest in the RL community as a possible way to remediate the scalability issues encountered in MARL when the number of agents is large [Perolat *et al.*, 2018]. [Guo *et al.*, 2019] combined a fixed-point method with Q-learning, but the convergence is ensured only under very restrictive Lipschitz conditions and the method can be applied efficiently only to finite-state models. [Subramanian and Mahajan, 2019] solve MFG using a gradient-type approach. The idea of using FP in MFGs has been introduced in [Cardaliaguet and Hadikhanloo, 2017], assuming the agent can compute perfectly the best response. [Elie *et al.*, 2020; Perrin *et al.*, 2020] combined FP with RL methods. However, the numerical techniques used therein do not scale to higher dimensions.

6 Conclusion

In this work we introduced Flock'n RL, a new numerical approach which allows solving MFGs with flocking effects where the agents reach a consensus in a decentralized fashion. Flock'n RL combines Fictitious Play with deep neural networks and reinforcement learning techniques (normalizing flows and soft actor-critic). We illustrated the method on challenging examples, for which no solution was previously known. In the absence of existing benchmark, we demonstrated the success of the method using a new kind of approximate exploitability. Thanks to the efficient representation of the distribution and to the model-free computation of a best response, the techniques developed here could be used to solve other acceleration controlled MFGs [Achdou *et al.*, 2020] or, more generally, other high-dimensional MFGs. Last, the flexibility of RL, which does not require a perfect knowledge of the model, allow us to tackle MFGs with complex topologies (such as boundary conditions or obstacles), which is a difficult problem for traditional methods based on partial differential equations.

References

[Achdou and Capuzzo-Dolcetta, 2010] Yves Achdou and Italo Capuzzo-Dolcetta. Mean field games: numerical methods. *SIAM J. Numer. Anal.*, 2010.

[Achdou *et al.*, 2020] Yves Achdou, Paola Mannucci, Claudio Marchi, and Nicoletta Tchou. Deterministic mean field games with control on the acceleration. *NoDEA*, 2020.

[Bailo *et al.*, 2018] Rafael Bailo, Mattia Bongini, José A Carrillo, and Dante Kalise. Optimal consensus control of the Cucker-Smale model. *IFAC*, 2018.

[Caponigro *et al.*, 2013] Marco Caponigro, Massimo Fornasier, Benedetto Piccoli, and Emmanuel Trélat. Sparse stabilization and optimal control of the Cucker-Smale model. *Mathematical Control and Related Fields*, 2013.

[Cardaliaguet and Hadikhanloo, 2017] Pierre Cardaliaguet and Saeed Hadikhanloo. Learning in mean field games: the fictitious play. *ESAIM Cont. Optim. Calc. Var.*, 2017.

[Carlini and Silva, 2014] Elisabetta Carlini and Francisco J. Silva. A fully discrete semi-Lagrangian scheme for a first order mean field game problem. *SIAM J. Numer. Anal.*, 2014.

[Carmona and Delarue, 2018] René Carmona and François Delarue. *Probabilistic theory of mean field games with applications. I.* 2018.

[Carmona and Laurière, 2019] René Carmona and Mathieu Laurière. Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: II - The finite horizon case. (Under review at Annals of Applied Probability), 2019.

[Cucker and Smale, 2007] Felipe Cucker and Steve Smale. Emergent behavior in flocks. *IEEE Transactions on automatic control*, 2007.

[Durkan *et al.*, 2019] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *proc. of NeurIPS*, 2019.

[Elie *et al.*, 2020] Romuald Elie, Julien Perolat, Mathieu Laurière, Matthieu Geist, and Olivier Pietquin. On the convergence of model free learning in mean field games. In *proc. of AAAI*, 2020.

[Guo *et al.*, 2019] Xin Guo, Anran Hu, Renyuan Xu, and Junzi Zhang. Learning mean-field games. In *proc. of NeurIPS*, 2019.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[Hill *et al.*, 2018] 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. https://github.com/hill-a/stable-baselines, 2018.

[Kobyzev *et al.*, 2020] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *PAMI*, 2020.

[Lanctot *et al.*, 2017] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *proc. of NeurIPS*, 2017.

[Lasry and Lions, 2007] Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Jpn. J. Math.*, 2007.

[Lillicrap *et al.*, 2016] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *proc. of ICLR*, 2016.

[Nourian *et al.*, 2011] Mojtaba Nourian, Peter E Caines, and Roland P Malhamé. Mean field analysis of controlled Cucker-Smale type flocking: Linear analysis and perturbation equations. *IFAC*, 2011.

[Perolat *et al.*, 2018] Julien Perolat, Bilal Piot, and Olivier Pietquin. Actor-critic fictitious play in simultaneous move multistage games. In *Proc. of AISTATS*, 2018.

[Perrin *et al.*, 2020] Sarah Perrin, Julien Pérolat, Mathieu Laurière, Matthieu Geist, Romuald Elie, and Olivier Pietquin. Fictitious play for mean field games: Continuous time analysis and applications. In *proc. of NeurIPS*, 2020.

[Reynolds, 1987] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proc. of SIGGRAPH*, 1987.

[Rezende and Mohamed, 2015] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *proc. of ICML*, 2015.

[Ruthotto *et al.*, 2020] Lars Ruthotto, Stanley J Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu Fung. A machine learning framework for solving high-dimensional mean field game and control problems. *PNAS*, 2020.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[Shaw, 1975] E Shaw. Naturalist at large-fish in schools. *Natural History*, 1975.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 2018.

[Subramanian and Mahajan, 2019] Jayakumar Subramanian and Aditya Mahajan. Reinforcement learning in stationary mean-field games. In *proc. of AAMAS*, 2019.

[Sutton *et al.*, 1999] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *proc. of NeurIPS*. MIT Press, 1999.