# APAC-Net: Alternating the Population and Agent Control via Two Neural Networks to Solve High-Dimensional Stochastic Mean Field Games

Alex Tong Lin* Samy Wu Fung* Wuchen Li Levon Nurbekyan Stanley Osher

Department of Mathematics
University of California, Los Angeles
{atlin,swufung,wcli,lnurbek,sjo}@math.ucla.edu

June 19, 2020

**Abstract**

We present APAC-Net, an alternating population and agent control neural network for solving stochastic mean field games (MFGs). Our algorithm is geared toward high-dimensional instances of MFGs that are beyond reach with existing solution methods. We achieve this in two steps. First, we take advantage of the underlying variational primal-dual structure that MFGs exhibit and phrase it as a convex-concave saddle point problem. Second, we parameterize the value and density functions by two neural networks, respectively. By phrasing the problem in this manner, solving the MFG can be interpreted as a special case of training a generative adversarial network (GAN). We show the potential of our method on up to 100-dimensional MFG problems.

## 1 Introduction

Mean field games (MFGs) are a class of problems that model large populations of interacting agents. They have been widely used in economics [2, 3, 28, 31], finance [3, 9, 12, 24], industrial engineering [20, 27, 36], swarm robotics [22, 44], epidemic modelling [13, 39] and data science [10, 33, 59]. In mean field games, a continuum population of small rational agents play a non-cooperative differential game on a time horizon $[0, T]$. At the optimum, the agents reach a Nash equilibrium, where they can no longer unilaterally improve their objectives. Given the initial distribution of agents $\rho_0 \in \mathcal{P}(\mathbb{R}^n)$, where $\mathcal{P}(\mathbb{R}^n)$ is the space of all probability densities, the solution to MFGs are obtained by solving the system of partial differential equations (PDEs),

$$\begin{aligned} &-\partial_t \phi - \nu\Delta\phi + H(x, \nabla\phi) = f(x, \rho) && \text{(HJB)} \\ &\partial_t \rho - \nu\Delta\rho - \operatorname{div}(\rho \nabla_p H(x, \nabla\phi)) = 0 && \text{(FP)} \\ &\rho(x, 0) = \rho_0, \quad \phi(x, T) = g(x, \rho(\cdot, T)) \end{aligned} \tag{1.1}$$

which couples a Hamilton-Jacobi-Bellman (HJB) equation and a Fokker-Planck (FP) equation. Here, $\phi\colon \mathbb{R}^n \times [0, T] \to \mathbb{R}$ is the value function, i.e., the policy that guides the agents, $H\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the Hamiltonian, which describes the physics of the environment, $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^n)$ is the distribution of agents at time $t$, $f\colon \mathbb{R}^n \times \mathcal{P}(\mathbb{R}^n) \to \mathbb{R}$ denotes the interaction between the agents and the population, and $g\colon \mathbb{R}^n \times \mathcal{P}(\mathbb{R}^n) \to \mathbb{R}$ is the terminal

*Equal contribution.

condition, which guides the agents to the final distribution. Under standard assumptions, i.e., convexity of $H$, and monotonicity of $f$ and $g$, the solution to (1.1) exists and is unique. See [15, 37, 38] for more details. Although there is a plethora of fast solvers for the solution of (1.1) in two and three dimensions [1, 7, 15, 16, 17, 35], numerical methods for solving (1.1) in high dimensions are practically nonexistent due to the need for grid-based spatial discretization. These grid-based methods are prone to the curse of dimensionality, i.e., their computational complexity grows exponentially with spatial dimension [6]. Thus, grid-based methods cannot be tractably used on, e.g., modeling an energy efficient heating, ventilation, and air conditioning system in a complex building, where the dimensions can be as high as 1000 [40].

**Our Contribution** We present APAC-Net, an alternating population and agent control neural network approach for tractably solving high-dimensional MFGs in the stochastic case ($\nu > 0$). To this end, we phrase the MFG problem as a saddle-point problem [7, 18, 38] and parameterize the value function *and* the density function. This formulation allows us to circumvent using spatial grids or uniformly sampling in high dimensions, i.e., the curse of dimensionality. While spatial grids for MFGs are also avoided in [52], their work is limited to the deterministic setting ($\nu = 0$). Thus, to the best of our knowledge, APAC-Net is the first model that solves high-dimensional MFGs in the stochastic setting ($\nu > 0$). APAC-Net does this by drawing from a natural connection between MFGs and generative adversarial neural networks (GANs) [29] (see 3), a powerful class of generative models that have shown remarkable success on various types of datasets [5, 11, 21, 29, 32, 42].

## 2 Variational Primal-Dual Formulation of Mean Field Games

We derive the mathematical formulation of MFGs for our framework; in particular, we arrive at a primal-dual convex-concave formulation tailored for our alternating networks approach. An MFG system (1.1) is called potential, if there exist functionals $\mathcal{F}, \mathcal{G}$ such that

$$\delta_\rho \mathcal{F} = f(x, \rho) \quad \text{and} \quad \delta_\rho \mathcal{G} = g(x, \rho), \tag{2.1}$$

where

$$\langle \delta_\rho \mathcal{F}(\rho), \mu \rangle = \lim_{h \to 0} \frac{\mathcal{F}(\rho + h\mu) - \mathcal{F}(\rho)}{h}, \quad \langle \delta_\rho \mathcal{G}(\rho), \mu \rangle = \lim_{h \to 0} \frac{\mathcal{G}(\rho + h\mu) - \mathcal{G}(\rho)}{h}, \ \forall\ \mu. \tag{2.2}$$

That is, there exist functionals $\mathcal{F}, \mathcal{G}$ such that their variational derivatives with respect to $\rho$ are the interaction and terminal costs $f$ and $g$ from (1.1). A critical feature of potential MFGs is that the solution to (1.1) can be formulated as the solution to a convex-concave saddle point optimization problem. To this end, we begin by stating (1.1) as a variational problem [7, 38] akin to the Benamou-Brenier formulation for the Optimal Transport (OT) problem:

$$\begin{aligned} &\inf_{\rho, v} \int_0^T \left\{ \int_\Omega \rho(x,t) L(x, v(x,t)) dx \ + \mathcal{F}(\rho(\cdot, t)) \right\} dt + \mathcal{G}(\rho(\cdot, T)) \\ &\text{s.t. } \partial_t \rho - \nu \Delta \rho + \nabla \cdot (\rho v) = 0, \quad \rho(x, 0) = \rho_0(x), \end{aligned} \tag{2.3}$$

where $L \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the Lagrangian function corresponding to the Legendre transform of the Hamiltonian $H$, and $\mathcal{F}, \mathcal{G} \colon \mathbb{R}^n \times \mathcal{P}(\mathbb{R}^n) \to \mathbb{R}$ are mean field interaction terms, and $v \colon \mathbb{R}^n \times [0, T] \to \mathbb{R}^n$ is the velocity field. This formulation can be viewed as a multi-agent reinforcement learning (RL) problem where there are infinitely many players [10, 33, 52], the key difference is that unlike in RL, the reward function and the dynamics (FP) are known

in our setting. Next, setting $\phi$ as a Lagrange multiplier, we insert the PDE constraint into the objective to get

$$\begin{aligned}\sup_{\phi}\inf_{\rho(x,0)=\rho_0(x),v}\int_0^T\left\{\int_\Omega \rho(x,t)L(x,v(x,t))dx\ +\mathcal{F}(\rho(\cdot,t))\right\}dt+\mathcal{G}(\rho(\cdot,T))\\ -\int_0^T\int_\Omega \phi(x,t)\left(\partial_t\rho-\nu\Delta\rho+\nabla\cdot(\rho(x,t)v(x,t))\right)\,dx\,dt.\end{aligned}\tag{2.4}$$

Finally, integrating by parts and minimizing with respect to $v$ to obtain the Hamiltonian via $H(x,p)=\inf_v\{-p\cdot v+L(x,v)\}$, we obtain

$$\begin{aligned}\inf_{\rho(x,0)=\rho_0(x)}\sup_{\phi}\int_0^T\left\{\int_\Omega\left(\partial_t\phi+\nu\Delta\phi-H(x,\nabla\phi)\right)\rho(x,t)\,dx+\mathcal{F}(\rho(\cdot,t))\right\}dt\\ +\int_\Omega\phi(x,0)\rho_0(x)dx+\mathcal{G}(\rho(\cdot,T))-\int_\Omega\phi(x,T)\rho(x,T)dx.\end{aligned}\tag{2.5}$$

This formula can also be obtained in the context of HJB equations in density spaces [17], or by integrating the HJB and the FP equations in (1.1) with respect to $\rho$ and $\phi$, respectively [18]. In [18], it was observed that all MFG systems admit an infinite-dimensional two-player general-sum game formulation, and the potential MFGs are the ones that correspond to zero-sum games. In this interpretation, Player 1 represents the *mean-field* or the *population as a whole* and their strategy is the population density $\rho$. Furthermore, Player 2 represents the *generic agent* and their strategy is the value function $\phi$. The aim of Player 2 is to provide a strategy that yields the best response of a generic agent against the population. This interpretation is in accord with the intuition behind generative adversarial networks (GANs), as the key observation is that under mild assumptions on $\mathcal{F}$ and $\mathcal{G}$, each spatial integral is really an expectation from $\rho$. The formulation (2.5) is the cornerstone of our method.

# 3 Connections to GANs

**Generative Adversarial Networks** In GANs [29], we have a discriminator and generator, and the goal is to obtain a generator that is able to produce samples from a desired distribution. The generator does this by taking samples from a known distribution $\mathcal{N}$ and transforming them into samples from the desired distribution. Meanwhile, the purpose of the discriminator is to aid the optimization of the generator. Given a generator network $G_\theta$ and a discriminator network $D_\omega$, the original GAN objective is to find an equilibrium to the minimax problem

$$\inf_{G_\theta}\sup_{D_\omega}\mathbb{E}_{x\sim\rho_0}\left[\log D_\omega(x)\right]+\mathbb{E}_{z\sim\mathcal{N}}\left[\log(1-D_\omega(G_\theta(z)))\right].\tag{3.1}$$

Here, the discriminator acts as a classifier that attempts to distinguish real images from fake/generated images, and the goal of the generator is to produce samples that "fool" the discriminator.

**Wasserstein GANs** In Wasserstein GANs [5], the motivation is drawn from OT theory, where now the objective function is changed to the Wasserstein-1 (W1) distance in the Kantorovich-Rubenstein dual formulation

$$\inf_{G_\theta}\sup_{D_\omega}\mathbb{E}_{x\sim\rho_0}\left[D_\omega(x)\right]-\mathbb{E}_{z\sim\mathcal{N}}\left[D_\omega(G_\theta(z))\right],\quad\text{s.t.}\quad\|\nabla D\|\leqslant 1,\tag{3.2}$$

and the discriminator is required to be 1-Lipschitz. In this setting, the goal of the discriminator is to compute the W1 distance between the distribution of $\rho_0$ and $G_\theta(z)$. In practice,

using the W1 distance helps prevent the generator from suffering "mode collapse," a situation where the generator produces samples from only one mode of the distribution $\rho_0$; for instance, if $\rho_0$ is the distribution of images of handwritten digits, then mode collapse entails producing only, say, the 0 digit. Originally, weight-clipping was to enforce the Lipschitz condition of the discriminator network [5], but an improved method using a penalty on the gradient was used in [32].

**GANs ↔ MFGs** A Wasserstein GAN can be seen as a particular instance of a deterministic MFG [7, 8, 38]. Specifically, consider the MFG (2.5) in the following setting. Let $\nu = 0$, $\mathcal{G}$ be a hard constraint with target measure $\rho_T$ (as in optimal transport), and let $H$ be the Hamiltonian defined by

$$H(x,p) = \mathbb{1}_{\|p\|\leqslant 1} = \begin{cases} 0 & \|p\| \leqslant 1 \\ \infty & \text{otherwise} \end{cases}, \tag{3.3}$$

where we note that this Hamiltonian arises when the Lagrangian is given by $L(x,v) = \|v\|_2$. Then (2.5) reduces to,

$$\begin{aligned} \sup_{\phi} \int_\Omega \phi(x)\rho_0(x)\,dx - \int_\Omega \phi(x)\rho_T(x)\,dx \\ \text{s.t. } \|\nabla\phi(x)\| \leqslant 1, \end{aligned} \tag{3.4}$$

where we note that the optimization in $\rho$ leads to $\partial_t\phi - H(x,\nabla\phi) = 0$. And since $H(p) = \mathbb{1}_{\|p\|\leqslant 1}$, we have that $\partial_t\phi = 0$, and $\phi(x,t) = \phi(x)$ for all $t$. We observe that the above is precisely the Wasserstein-1 distance in the Kantorovich-Rubenstein duality [57].

# 4 APAC-Net

The training process for our MFG is similar to that of GANs. We initialize neural networks $N_\omega(x,t)$ and $N_\theta(z,t)$. We then let

$$\phi_\omega(x,t) = (1-t)N_\omega(x,t) + tg(x), \quad G_\theta(z,t) = (1-t)z + tN_\theta(z,t), \tag{4.1}$$

where $z \sim \rho_0$ are samples drawn from the initial distribution. Thus, $G_\theta$ is the pushforward of $\rho_0$. One difference between our formulation and GANs is that $\phi_\omega$ automatically encodes the terminal condition by design. More generally, APAC-Net encodes the underlying structure of MFGs via (2.5) and (4.1), which absolves the network from learning the entire MFG solution from the ground up.

Our strategy for training this GAN-like MFG consists of alternately training $G_\theta$ (the population), and $\phi_\omega$ (the value function for an individual agent). Intuitively, this means we are *alternating the population and agent control neural networks* (APAC-Net) in order to find the equilibrium. Specifically, we train $\phi_\omega$ by first sampling a batch $\{z_b\}_{b=1}^B$ from the given initial density $\rho_0$, and $\{t_b\}_{b=1}^B$ uniformly from $[0,1]$. Next, we compute the push-forward $x_b = G_\theta(z_b,t_b)$ for $b = 1,\dots,B$. We then compute the loss,

$$\text{loss}_\phi = \frac{1}{B}\sum_{b=1}^B \phi_\omega(x_b,0) + \frac{1}{B}\sum_{b=1}^B \partial_t\phi_\omega(x_b,t_b) + \nu\Delta\phi_\omega(x_b,t_b) - H(\nabla_x\phi_\omega(x_b,t_b)) \tag{4.2}$$

where we can optionally add a regularization term

$$\lambda\frac{1}{B}\sum_{b=1}^B \|\partial_t\phi_\omega(x_b,t_b) + \nu\Delta\phi_\omega(x_b,t_b) - H(\nabla_x\phi_\omega(x_b,t_b)) + f(x_b,t_b)\| \tag{4.3}$$

**Algorithm 1** APAC-Net

**Require:** $\nu$ diffusion parameter, $g$ terminal cost, $H$ Hamiltonian, $f$ interaction term.
**Require:** Initialize neural networks $N_\omega$ and $N_\theta$, batch size $B$
**Require:** Set $\phi_\omega$ and $G_\theta$ as in (4.1)
**while** not converged **do**
  **train** $\phi_\omega$:
  Sample batch $\{(z_b, t_b)\}_{b=1}^B$ where $z_b \sim \rho_0$ and $t_b \sim \text{Unif}(0, T)$
  $x_b \leftarrow G_\theta(z_b, t_b)$ for $b = 1, \dots, B$.
  $\ell_0 \leftarrow \frac{1}{B}\sum_{b=1}^B \phi_\omega(x_b, 0)$
  $\ell_t \leftarrow \frac{1}{B}\sum_{b=1}^B \partial_t\phi_\omega(x_b, t_b) + \nu\Delta\phi_\omega(x_b, t_b) - H(\nabla_x\phi_\omega(x_b, t_b))$
  $\ell_{\text{HJB}} \leftarrow \lambda\frac{1}{B}\sum_{b=1}^B \|\partial_t\phi_\omega(x_b, t_b) + \nu\Delta\phi_\omega(x_b, t_b) - H(\nabla_x\phi_\omega(x_b, t_b)) + f(x_b, t_b)\|$
  Backpropagate the loss $\ell_{\text{total}} = \ell_0 + \ell_t + \ell_{\text{HJB}}$ to $\omega$ weights.

  **train** $G_\theta$:
  Sample batch $\{(z_b, t_b)\}_{b=1}^B$ where $z_b \sim \rho_0$ and $t_b \sim \text{Unif}(0, T)$
  $\ell_t \leftarrow \frac{1}{B}\sum_{b=1}^B \partial_t\phi_\omega(G_\theta(z_b, t_b), t_b) + \nu\Delta\phi_\omega(G_\theta(z_b, t_b), t_b) - H(\nabla_x\phi_\omega(G_\theta(z_b, t_b), t_b)) + f(G_\theta(z_b, t_b), t_b)$
  Backpropagate the loss $\ell_{\text{total}} = \ell_t$ to $\theta$ weights.
**end while**

to penalize deviations from the HJB equations [46, 52]. This extra regularization term has also been found effective in, e.g., Wasserstein GANs [31], where the norm of the gradient (i.e., the HJB equations) is penalized. Finally, we backpropagate the loss to the weights of $\phi_w$. To train the generator, we again sample $\{z_b\}_{b=1}^B$ and $\{t_b\}_{b=1}^B$ as before, and compute

$$\text{loss}_G = \frac{1}{B}\sum_{b=1}^B \partial_t\phi_\omega(G_\theta(z_b), t_b) + \nu\Delta\phi_\omega(G_\theta(z_b), t_b) - H(\nabla_x\phi_\omega(G_\theta(z_b), t_b)) + f(G_\theta(z_b), t_b). \tag{4.4}$$

Finally, we backpropagate this loss with respect to the weights of $G_\theta$ (see Alg 1).

# 5 Related Works

**High-dimensional MFGs and Optimal Control** To the best of our knowledge, the first work to solve MFGs efficiently in high dimensions ($d = 100$) was done in [52]. Their work consisted of using Lagrangian coordinates and parameterizing the value function using a neural network. Finally, to estimate the densities, the instantaneous change of variables formula [14]. This combination allowed them to successfully avoid using spatial grids when solving deterministic MFG problems ($\nu = 0$) with quadratic Hamiltonians. Besides only computing MFGs with $\nu = 0$, another limitation is that for non-quadratic Hamiltonians, the instantaneous change of variables formula may lead to high computational costs when estimating the density. APAC-Net circumvents this limitation by rephrasing the MFG as a saddle point problem (2.5) and using a GAN-based approach to train two neural networks instead. For problems involving high-dimensional optimal control and differential games, spatial grids were also avoided [15, 16, 17, 19, 41]. However, these methods are based on generating individual trajectories per agent, and cannot be directly applied to MFGs without spatial discretization of the density, thus limiting their use in high dimensions.

**Reinforcement Learning** Our work bears connections with multi-agent reinforcement learning (RL), where neither the Lagrangian $L$ nor the dynamics (constraint) in (2.3) are known. Here, a key difference is that multi-agent RL generally considers a finite number of players. [58] proposes a primal-dual distributed method for multi-agent RL. [33] proposes a

Q-Learning approach to solve these multi-agent RL problems. [10] studies the convergence of policy gradient methods on mean field reinforcement learning (MFRL) problems, i.e., problems where the agents try instead to learn the control which is socially optimal for the entire population. [60] uses an inverse reinforcement learning approach to learn the MFG model along with its reward function. [25] proposes an actor-critic method for finding the Nash equilibrium in linear-quadratic mean field games and establish linear convergence.

**Generative Modeling with Optimal Transport** There is a class of works that focus on using OT, a class of MFGs, to solve problems arising in data science, and in particular, GANs. [26] presents a tractable method to train large scale generative models using the Sinkhorn distance, which consist of loss functions that interpolate between Wasserstein (OT) distance and Maximum Mean Discrepancy (MMD). [53] proposes a mini-batch MMD-based distance to improve training GANs. [54] proposes a class of regularized Wasserstein GAN problems with theoretical guarantees. [56] uses a trained discriminator from GANs to further improve the quality of generated samples. [43] phrases the adversarial problem as a matching problem in order to avoid solving a minimax problem. Finally, [49] provides an excellent survey on recent numerical methods for OT and their applications to GANs.

**GAN-based Approach for MFGs** Our work is most similar to [8], where a connection between MFGs and GANs is also made. However, APAC-Net differs from [8] in two fundamental ways. First, instead of choosing the value function to be the generator, we set the *density* function as the generator. This choice is motivated by the fact that the generator outputs samples from a desired distribution. It is also aligned with other generative modeling techniques arising in continuous normalizing flows [23, 30, 45, 46]. Second, rather than setting the generator/discriminator losses as the residual errors of (1.1), we follow the works of [7, 17, 18, 38] and utilize the underlying variational primal-dual structure of MFGs, see (2.5); this allows us to arrive at the Kantorovich-Rubenstein dual formulation of Wasserstein GANs [57].

# 6 Numerical Experiments

We demonstrate the potential of APAC-Net on a series of high-dimensional MFG problems. We also illustrate the behavior the MFG solutions for different values of $\nu$ and use an analytical solution to illustrate the accuracy of APAC-Net. We also provide additional high-dimensional results in App. B.

**Experimental Setup** We assume without loss of generality $T = 1$. In all experiments, our neural networks have three hidden layers, with 100 hidden units per layer. We use a residual neural network (ResNet) for both networks, with skip connection weight 0.5. For $\phi_\omega$, we use the Tanh activation function, and for $G_\theta$, we use the ReLU activation function. For training, we use ADAM with $\beta = (0.5, 0.9)$, learning rate $5 \times 10^{-4}$ for $\phi_\omega$, learning rate $1 \times 10^{-4}$ for $G_\theta$, weight decay of $10^{-4}$ for both networks, batch size 50, and $\lambda = 1$ (the HJB penalty parameter) in Algorithm 1.

The Hamiltonians in our experiments have the form

$$H(x, p, t) = c\|p\|_2 + f(x, \rho(x, t)), \tag{6.1}$$

where $f(x, \rho(x, t))$ varies with the environment (either avoiding obstacles, or avoiding congestion, etc.), and $c$ is a constant (that represents maximal speed). Furthermore, we choose as terminal cost

$$\mathcal{G}(\rho(\cdot, T)) = \int_\Omega \|x - x_T\|_2 \rho(x, T) dx, \tag{6.2}$$

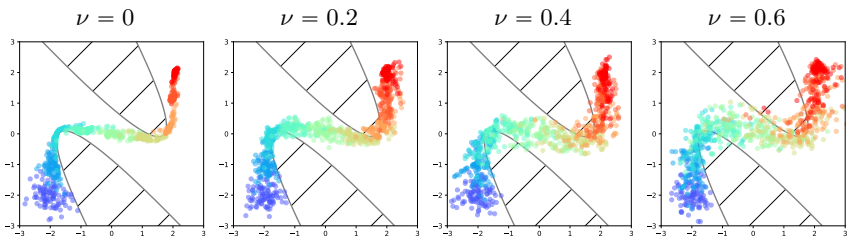


**Figure 1:** Comparison of 2D solutions for different values of $\nu$. The agents start at the blue points ($t = 0$) and end at the red points ($t = 1$).

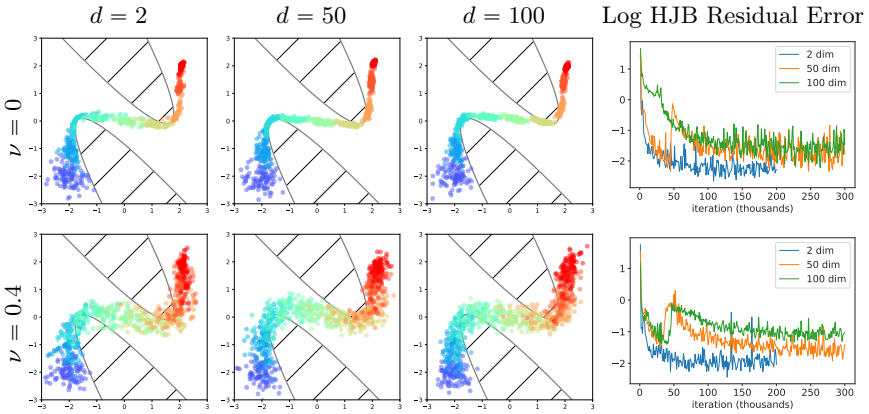


**Figure 2:** Computation of the obstacle problem in dimensions 2, 50, and 100 with stochasticity parameter $\nu = 0$ and 0.4. For dimension 50 and 100, we plot the first two dimensions.

which is the distance between the population and a target destination. To allow for verification of the high-dimensional solutions, we set the obstacle and congestion costs to only affect the first two dimensions. In Figs. 1, 2, 3, and 4, time is represented by color. Specifically, blue denotes starting time, red denotes final time, and the intermediate colors denote intermediate times. We also plot the HJB residual error, that is, $\ell_{\text{HJB}}$ in Alg. 1, on 4096 fixed sampled points which helps us monitor the convergence of APAC-Net. As in standard machine learning methods, all the plots in this section are generated using *validation* data, i.e., data not used in training, in order to gauge generalizability of APAC-Net. Further details as well as additional experiments can be found in the appendix.

**Effect of Stochasticity Parameter $\nu$** We investigate the effect of the stochasticity parameter $\nu$ on the behavior of the MFG solutions. In Fig. 1, we show the solutions for 2-dimensional MFGs using $\nu = 0, 0.2, 0.4$, and 0.6. As $\nu$ increases, the density of agents widens along the paths due to the added the added diffusion term in the HJB and FP equations in (1.1). These results are consistent with those in [47].

**Obstacles** We compute the solution to a MFG where the agents are required to avoid obstacles. In this case, we let

$$f(x_1, x_2, \ldots, x_d) = \gamma_{\text{obst}}(\max\{f_1(x_1, x_2), 0\} + \max\{f_2(x_1, x_2), 0\}) \tag{6.3}$$

with $\gamma_{\text{obst}} = 5$, and denoting $R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$ with $\theta = \pi/5$, $Q = \left(\begin{smallmatrix} 5 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, and $b = (0, 2)$, then

$$f_1(x_1, x_2) = -v^\top Q v - b \cdot v - 1, \quad \text{with } v = ((x_1, x_2) - (-2, 0.5))R. \tag{6.4}$$

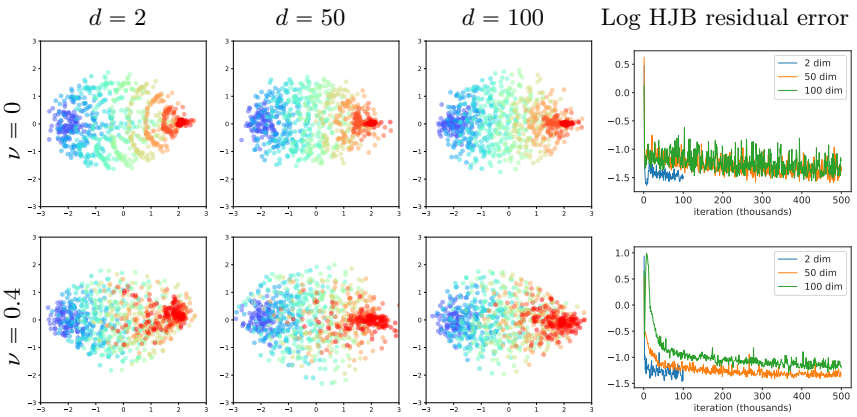


**Figure 3:** Computation of the congestion problem in dimensions 2, 50, and 100 with stochasticity parameter $\nu = 0$ and 0.4. For dimensions 50 and 100, we plot the first two dimensions.

Similarly, we let

$$f_2(x_1, x_2) = -w^\top Q w + b \cdot w - 1, \quad \text{with } w = ((x_1, x_2) - (2, -0.5))R. \tag{6.5}$$

The obstacles $f_1$ and $f_2$ are shown in hatched markings in Fig. 2. Our initial density $\rho_0$ is a Gaussian centered at $(-2, -2, 0, \ldots, 0)$ with standard deviation $1/\sqrt{10} \approx 0.32$, and the terminal function is $g(x) = \|(x_1, x_2) - (2, 2)\|_2$. We chose $c = 8$ in (6.1). The numerical results are shown in Fig. 2. Observe that results are similar across dimensions, verifying our high-dimensional computation. We run the two-dimensional problem for 200k iterations and the 50 and 100-dimensional problems for 300k iterations.

**Congestion** We choose the interaction term to penalize congestion, so that the agents are encouraged to spread out. In particular, we have

$$\mathcal{F}(\rho(x,t)) = \int_\Omega \int_\Omega \frac{1}{\|(x_1, x_2) - (y_1, y_2)\|^2 + 1} \, d\rho(x,t) \, d\rho(y,t), \tag{6.6}$$

which is the (bounded) inverse squared distance, averaged over pairs of agents. Computationally, we sample from $\rho$ twice and then calculate the integrand. Here, our initial density $\rho_0$ is a Gaussian centered at $(-2, 0, -2, \ldots, -2)$ with standard deviation $1/\sqrt{10} \approx 0.32$, the terminal function is $\mathcal{G}(x) = \|(x_1, x_2) - (2, 0)\|_2$, and we chose $c = 5$ in (6.1). Results are shown in Fig. 3, where we see qualitatively similar results across dimensions. We run the two-dimensional problem for 100k iterations and the 50 and 100-dimensional problems for 500k iterations.

**Congestion with Bottleneck Obstacle** We combine the congestion problem with a bottleneck obstacle. The congestion penalization is the same as (B.1), and the obstacle represents a bottleneck – thus agents are encouraged to spread out, but must squeeze together to avoid the obstacle. The initial density, terminal functions, $c$ in (6.1), and the expression penalizing congestion are the same as in the congestion experiment above. The obstacle is chosen to be

$$f(v) = \gamma_{\text{obst}} \max\left\{ -v^\top \begin{pmatrix} 5 & 0 \\ 0 & -1 \end{pmatrix} v - 0.1, 0 \right\}, \quad \text{with } v = (x_1, x_2) \tag{6.7}$$

with $\gamma_{\text{obst}} = 5$. As intuitively expected, the agents spread out before and after the bottleneck, but squeeze together in order to avoid the obstacle (see Fig. 4). We run the two-dimensional problem for 100k iterations and the 50 and 100-dimensional problems for 500k iterations. We observe similar results across dimensions.

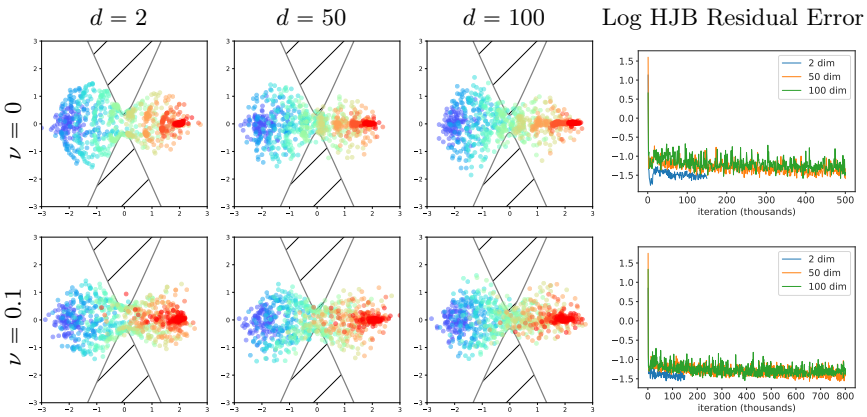


**Figure 4:** Computation of the congestion problem with a bottleneck in dimensions 2, 50, and 100 with stochasticity parameter $\nu = 0$ and 0.1. For dimensions 50 and 100, we plot the first two dimensions.

**Analytic Comparison** As a last experiment, we verify our method by comparing it to an analytic solution for dimensions 2, 50, and 100 with congestion ($\gamma = 0.1$) and without congestion ($\gamma = 0$). For

$$f = \gamma \ln(\rho), \quad H(x,p) = \frac{\|p\|^2}{2} - \frac{\beta\|x\|^2}{2}, \quad g(x) = \frac{\alpha|x|^2}{2} - \left(\nu d\alpha + \frac{\gamma d}{2} \ln \frac{\alpha}{2\pi\nu}\right) \tag{6.8}$$

and $\nu = \beta = 1$ in (1.1), the explicit formula for $\phi$ is given by

$$\phi(x,t) = \frac{\alpha|x|^2}{2} - \left(d\alpha + \frac{\gamma d}{2} \ln \frac{\alpha}{2\pi}\right) t, \quad \rho(x,t) = \left(\frac{\alpha}{2\pi}\right)^{\frac{d}{2}} e^{-\frac{\alpha|x|^2}{2}}, \tag{6.9}$$

where $\alpha = \frac{-\gamma + \sqrt{\gamma^2+4}}{2}$. For the $\gamma = 0.1$ case, we use Kernel Density Estimation [48, 50] to estimate $\rho$ from samples of the generator. The derivation of the analytic solution can be found in App. A.

For $d = 2$, we compute the relative error on a grid of size $32 \times 32 \times 16$, where we discretize the spatial domain $\Omega = [-2, 2]^2$ with $32 \times 32$ points and the time domain $[0, 1]$ with 16 points. Note here that the grid points are *validation* points, i.e., points that were not used in training. For $d = 50$ and $d = 100$, we use 4096 sampled points for validation since we cannot build a grid. We run a total of 30k and 60k iterations for $\gamma = 0$ and $\gamma = 0.1$, respectively. We validate every 1k iterations. Fig. 5 shows that our learned model approaches the true solution across all dimensions for both values of $\gamma$, indicating that APAC-Net generalizes well.

# 7 Conclusion

We present APAC-Net, an alternating population-agent control neural network for solving high-dimensional stochastic mean field games. To this end, our algorithm avoids the use of spatial grids by parameterizing the controls, $\phi$ and $\rho$, using two neural networks, respectively. Consequently, our method is geared toward high-dimensional instances of these problems that are beyond reach with existing grid-based methods. APAC-Net therefore sets the stage for solving realistic high-dimensional MFGs arising in, e.g., economics [2, 3, 28, 31], swarm robotics [22, 44], and perhaps most important/relevant, epidemic modelling [13, 39]. Our method also has natural connections with Wasserstein GANs, where $\rho$ acts as a generative network and $\phi$ acts as a discriminative network. Unlike GANs, however, APAC-Net incorporates the structure of MFGs via (2.5) and (4.1), which absolves the network

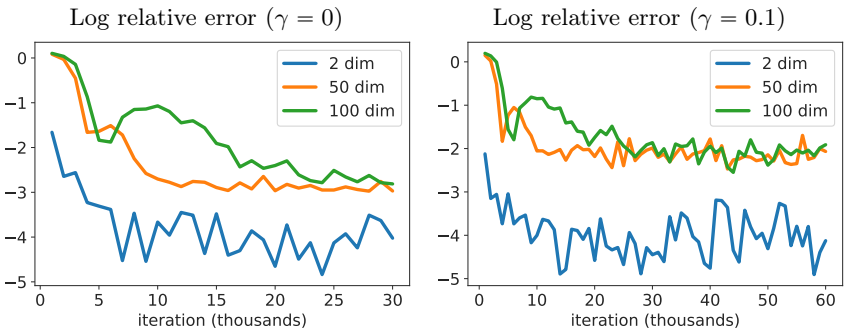


**Figure 5:** log relative errors in 2, 50, and 100 dimensions, and for $\gamma = 0, 0.1$. Here, $\gamma = 0$ means no interaction. For the $d = 2$ case, we compute the validation on a $32 \times 32$ grid over 16 uniformly-spaced timesteps with the true $\phi$ from eq. (6.9). For the $d = 50$ and 100 case, we compute on a sample of 4096 sample points, sampled from the initial density.

from learning an entire MFG solution from the ground up. Our experiments show that our method is able to solve 100-dimensional MFGs. As a future direction, we intend to investigate guidelines on the design of more effective network architectures, e.g., PDE-based networks [34, 51], neural ODEs [14], or sorting networks [4]. We also intend to use of our framework to solve more realistic problems.

## Acknowledgments and Disclosure of Funding

The authors are supported by AFOSR MURI FA9550-18-1-0502, AFOSR Grant No. FA9550-18-1-0167, and ONR Grant No. N00014-18-1-2527.

## References

[1] Yves Achdou and Italo Capuzzo-Dolcetta. Mean field games: numerical methods. *SIAM Journal on Numerical Analysis*, 48(3):1136–1162, 2010.

[2] Yves Achdou, Francisco J. Buera, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll. Partial differential equation models in macroeconomics. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 372(2028):20130397, 19, 2014. ISSN 1364-503X. doi: 10.1098/rsta.2013.0397. URL https://doi.org/10.1098/rsta.2013.0397.

[3] Yves Achdou, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll. Income and wealth distribution in macroeconomics: A continuous-time approach. Working Paper 23732, National Bureau of Economic Research, August 2017. URL http://www.nber.org/papers/w23732.

[4] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301, 2019.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[6] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[7] Jean-David Benamou, Guillaume Carlier, and Filippo Santambrogio. Variational mean field games. In *Active Particles, Volume 1*, pages 141–171. Springer, 2017.

[8] Haoyang Cao, Xin Guo, and Mathieu Laurière. Connecting gans and mfgs. *arXiv:2002.04112*, 2020.

[9] Pierre Cardaliaguet and Charles-Albert Lehalle. Mean field game of controls and an application to trade crowding. *Math. Financ. Econ.*, 12(3):335–363, 2018. ISSN 1862-9679. doi: 10.1007/s11579-017-0206-z. URL https://doi.org/10.1007/s11579-017-0206-z.

[10] René Carmona, Mathieu Laurière, and Zongjun Tan. Linear-quadratic mean-field reinforcement learning: convergence of policy gradient methods. *arXiv:1910.04295*, 2019.

[11] Kenji Fukumizu Casey Chu, Kentaro Minami. Smoothness and stability in gans. *arXiv:2002.04185*, 2020.

[12] Philippe Casgrain and Sebastian Jaimungal. Algorithmic trading in competitive markets with mean field games. *SIAM News*, 52(2), 2019.

[13] Sheryl L Chang, Mahendra Piraveenan, Philippa Pattison, and Mikhail Prokopenko. Game theoretic modelling of infectious disease dynamics and intervention methods: a review. *Journal of Biological Dynamics*, 14(1):57–89, 2020.

[14] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[15] Yat Tin Chow, Jérôme Darbon, Stanley Osher, and Wotao Yin. Algorithm for overcoming the curse of dimensionality for time-dependent non-convex hamilton–jacobi equations arising from optimal control and differential games problems. *Journal of Scientific Computing*, 73(2-3):617–643, 2017.

[16] Yat Tin Chow, Jérôme Darbon, Stanley Osher, and Wotao Yin. Algorithm for overcoming the curse of dimensionality for certain non-convex hamilton–jacobi equations, projections and differential games. *Annals of Mathematical Sciences and Applications*, 3(2):369–403, 2018.

[17] Yat Tin Chow, Wuchen Li, Stanley Osher, and Wotao Yin. Algorithm for hamilton–jacobi equations in density space via a generalized hopf formula. *Journal of Scientific Computing*, 80(2):1195–1239, 2019.

[18] Marco Cirant and Levon Nurbekyan. The variational structure and time-periodic solutions for mean-field games systems. *Minimax Theory Appl.*, 3(2):227–260, 2018. ISSN 2199-1413.

[19] Jérôme Darbon and Stanley Osher. Algorithms for overcoming the curse of dimensionality for certain hamilton–jacobi equations arising in control theory and elsewhere. *Research in the Mathematical Sciences*, 3(1):19, 2016.

[20] A. De Paola, V. Trovato, D. Angeli, and G. Strbac. A mean field game approach for distributed control of thermostatic loads acting in simultaneous energy-frequency response markets. *IEEE Transactions on Smart Grid*, 10(6):5987–5999, Nov 2019. ISSN 1949-3061. doi: 10.1109/TSG.2019.2895247.

[21] Yonatan Dukler, Wuchen Li, Alex Lin, and Guido Montufar. Wasserstein of wasserstein loss for learning generative models. In *International Conference on Machine Learning*, pages 1716–1725, 2019.

[22] Karthik Elamvazhuthi and Spring Berman. Mean-field models in swarm robotics: a survey. *Bioinspiration & Biomimetics*, 15(1):015001, 2019.

[23] Chris Finlay, Björn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. *arXiv:2002.02798*, 2020.

[24] D. Firoozi and P. E. Caines. An optimal execution problem in finance targeting the market trading speed: An mfg formulation. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 7–14, Dec 2017.

[25] 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*, 2020. URL https://openreview.net/forum?id=H1lhqpEYPr.

[26] Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617. PMLR, 2018.

[27] Diogo A. Gomes and J. Saúde. A mean-field game approach to price formation in electricity markets. *arXiv:1807.07088*, 2018.

[28] Diogo A Gomes, Levon Nurbekyan, and Edgard A Pimentel. *Economic models and mean-field games theory*. IMPA Mathematical Publications. Instituto Nacional de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, 2015.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[30] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*, 2019.

[31] Olivier Guéant, Jean-Michel Lasry, and Pierre-Louis Lions. Mean field games and applications. In *Paris-Princeton lectures on mathematical finance 2010*, pages 205–266. Springer, 2011.

[32] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[33] Xin Guo, Anran Hu, Renyuan Xu, and Junzi Zhang. Learning mean-field games. In *Advances in Neural Information Processing Systems*, pages 4967–4977, 2019.

[34] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[35] Matt Jacobs, Flavien Léger, Wuchen Li, and Stanley Osher. Solving large-scale optimization problems with a convergence rate independent of grid size. *SIAM Journal on Numerical Analysis*, 57(3):1100–1123, 2019.

[36] Arman C. Kizilkale, Rabih Salhab, and Roland P. Malhamé. An integral control formulation of mean field game based large scale coordination of loads in smart grids. *Automatica*, 100:312 – 322, 2019. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2018.11.029. URL http://www.sciencedirect.com/science/article/pii/S0005109818305612.

[37] Jean-Michel Lasry and Pierre-Louis Lions. Jeux à champ moyen. ii–horizon fini et contrôle optimal. *Comptes Rendus Mathématique*, 343(10):679–684, 2006.

[38] 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. URL https://doi.org/10.1007/s11537-007-0657-8.

[39] Wonjun Lee, Siting Liu, Hamidou Tembine, and Stanley Osher. Controlling propagation of epidemics via mean-field games. *UCLA CAM preprint:20-19*, 2020.

[40] Sisi Li, Shengbo Eben Li, and Kun Deng. Mean-field control for improving energy efficiency. In *Automotive Air Conditioning*, pages 125–143. Springer, 2016.

[41] Alex Tong Lin, Yat Tin Chow, and Stanley J Osher. A splitting method for overcoming the curse of dimensionality in hamilton–jacobi equations arising from nonlinear optimal control and differential games with applications to trajectory generation. *Communications in Mathematical Sciences*, 16(7), 2018.

[42] Alex Tong Lin, Wuchen Li, Stanley Osher, and Guido Montúfar. Wasserstein proximal of gans. *UCLA CAM preprint:18-53*, 2018.

[43] Jingrong Lin, Keegan Lensink, and Eldad Haber. Fluid flow mass transport for generative networks. *arXiv:1910.01694*, 2019.

[44] Zhiyu Liu, Bo Wu, and Hai Lin. A mean field game approach to swarming robots control. In *2018 Annual American Control Conference (ACC)*, pages 4293–4298. IEEE, 2018.

[45] Derek Onken and Lars Ruthotto. Discretize-optimize vs. optimize-discretize for time-series regression and continuous normalizing flows. *arXiv:2005.13420*, 2020.

[46] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. OT-Flow: Fast and accurate continuous normalizing flows via optimal transport. *arXiv:2006.00104*, 2020.

[47] Christian Parkinson, David Arnold, Andrea L Bertozzi, and Stanley Osher. A model for optimal human navigation with stochastic effects. *arXiv:2005.03615*, 2020.

[48] Emanuel Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09 1962. doi: 10.1214/aoms/1177704472. URL https://doi.org/10.1214/aoms/1177704472.

[49] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[50] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3):832–837, 09 1956. doi: 10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

[51] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2019.

[52] 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 mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193, 2020.

[53] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving GANs using optimal transport. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkQkBnJAb.

[54] Maziar Sanjabi, Jimmy Ba, Meisam Razaviyayn, and Jason D Lee. On the convergence and robustness of training gans with regularized optimal transport. In *Advances in Neural Information Processing Systems*, pages 7091–7101, 2018.

[55] David W Scott and Stephan R Sain. Multidimensional density estimation. *Handbook of statistics*, 24:229–261, 2005.

[56] Akinori Tanaka. Discriminator optimal transport. In *Advances in Neural Information Processing Systems*, pages 6813–6823, 2019.

[57] Cédric Villani. *Topics in Optimal Transportation*. American Mathematical Soc., 2003.

[58] Hoi-To Wai, Zhuoran Yang, Zhaoran Wang, and Mingyi Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Advances in Neural Information Processing Systems*, pages 9649–9660, 2018.

[59] E Weinan, Jiequn Han, and Qianxiao Li. A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences*, 6(1):10, 2019.

[60] Jiachen Yang, Xiaojing Ye, Rakshit Trivedi, Huan Xu, and Hongyuan Zha. Deep mean field games for learning optimal behavior policy of large populations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HktK4BeCZ.

# A Derivation of Analytic Solution

We derive explicit formulas used to test our approximate solutions in Sec. 6. Assume that $\nu, \beta > 0, \gamma \geqslant 0$ and

$$H(x,p,t) = \frac{|p|^2}{2} - \frac{\beta|x|^2}{2}, \quad f(x,\rho) = \gamma \ln \rho. \tag{A.1}$$

Then (1.1) becomes

$$\begin{aligned} -\partial_t \phi - \nu\Delta\phi + \frac{|\nabla\phi|^2}{2} - \frac{\beta|x|^2}{2} &= \gamma \ln \rho, \\ \partial_t \rho - \nu\Delta\rho - \operatorname{div}(\rho\nabla\phi) &= 0, \\ \rho(x,0) = \rho_0, \quad \phi(x,T) &= \Psi(x). \end{aligned} \tag{A.2}$$

We find solutions to this system by searching for stationary solutions first:

$$\begin{aligned} -\nu\Delta\phi + \frac{|\nabla\phi|^2}{2} - \frac{\beta|x|^2}{2} &= \gamma \ln \rho + \bar{H}, \\ -\nu\Delta\rho - \operatorname{div}(\rho\nabla\phi) &= 0, \end{aligned} \tag{A.3}$$

and then writing

$$\phi(x,t) = \phi(x) - t\bar{H}, \quad \rho(x,t) = \rho(x). \tag{A.4}$$

The second equation in (A.3) yields $\rho = ce^{-\frac{\phi}{\nu}}$, where $c$ is chosen so that $\int \rho = 1$. Plugging this in the first equation in (A.3) we obtain

$$-\nu\Delta\phi + \frac{|\nabla\phi|^2}{2} - \frac{\beta|x|^2}{2} = \gamma \ln c - \frac{\gamma}{\nu}\phi + \bar{H}. \tag{A.5}$$

Now we make an ansatz that $\phi(x) = \frac{\alpha|x|^2}{2}$. Then we have that $\Delta\phi = d\alpha$, $\nabla\phi = \alpha x$, and obtain

$$-\nu d\alpha + \frac{\alpha^2|x|^2}{2} - \frac{\beta|x|^2}{2} = \gamma \ln c - \frac{\gamma\alpha|x|^2}{2\nu} + \bar{H}. \tag{A.6}$$

Therefore, we have that

$$\alpha^2 + \frac{\gamma\alpha}{\nu} = \beta, \quad \bar{H} = -\nu d\alpha - \gamma \ln c. \tag{A.7}$$

From the first equation, we obtain that

$$\alpha = \frac{-\gamma + \sqrt{\gamma^2 + 4\nu^2\beta}}{2\nu}. \tag{A.8}$$

On the other hand, we have that

$$\int \rho = c \int e^{-\frac{\alpha|x|^2}{2\nu}} dx = c\left(\frac{2\pi\nu}{\alpha}\right)^{\frac{d}{2}} = 1, \tag{A.9}$$

so

$$c = \left(\frac{\alpha}{2\pi\nu}\right)^{\frac{d}{2}} \quad \text{and} \quad \bar{H} = -\nu d\alpha - \frac{\gamma d}{2} \ln \frac{\alpha}{2\pi\nu}. \tag{A.10}$$

Summarizing, we get that for any $\nu, \beta > 0, \gamma \geqslant 0$ the following is a solution for (A.2)

$$\phi(x,t) = \frac{\alpha|x|^2}{2} - \left(\nu d\alpha + \frac{\gamma d}{2} \ln \frac{\alpha}{2\pi\nu}\right) t, \quad \rho(x,t) = \left(\frac{\alpha}{2\pi\nu}\right)^{\frac{d}{2}} e^{-\frac{\alpha|x|^2}{2\nu}} \tag{A.11}$$

where $\alpha$ is given by (A.8), and

$$g(x) = \frac{\alpha|x|^2}{2} - \left(\nu d\alpha + \frac{\gamma d}{2} \ln \frac{\alpha}{2\pi\nu}\right) T, \quad \rho_0(x) = \left(\frac{\alpha}{2\pi\nu}\right)^{\frac{d}{2}} e^{-\frac{\alpha|x|^2}{2\nu}}. \tag{A.12}$$

Choosing $\beta = \nu = 1$, (A.11) gives the analytic solution used in Sec. 6.

# B Details on Numerical Results and More Experiments

**Congestion** Here we elaborate on how we compute the congestion term,

$$\mathcal{F}(\rho(x,t)) = \int_\Omega \int_\Omega \frac{1}{\|(x_1,x_2)-(y_1,y_2)\|^2+1}\, d\rho(x,t)\, d\rho(y,t), \tag{B.1}$$

We do this by first using the batch $\{z_b\}_{b=1}^B$, which was used for training (and sampled from $\rho_0$), and then compute another batch $\{y_b\}_{b=1}^B$ again sampled from $\rho_0$. Letting $\{t_b\}_{b=1}^B$ be a batch of time-points uniformly sampled in $[0,1]$, we estimate the interaction cost with,

$$\mathcal{F}(\rho(x,t)) \approx \sum_{i=1}^B \frac{1}{\|G_\theta(z_b,t_b)-G_\theta(y_b,t_b)\|^2+1}. \tag{B.2}$$

**Congestion with Bottleneck Obstacle and Higher Stochasticity** As mentioned in the main text, when choosing a stochasticity parameter $\nu > 0.1$, the stochasticity dominates the dynamics and the obstacles do not interact as much with the obstacle. We plot these results in Fig. 6, where for 2 dimensions, we trained for 150k iterations, and for 50 and 100 dimensions, we trained for 800k iterations. All environment and training parameters are the same as in the main text, except that now $\nu = 0.4$.

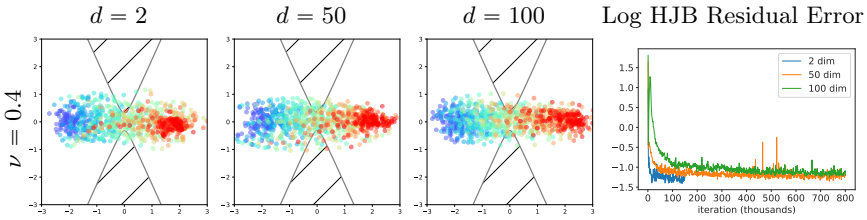


**Figure 6:** Computation of the congestion problem with a bottleneck in dimensions 2, 50, and 100 with stochasticity parameter $\nu = 0.4$. For dimensions 50 and 100, we plot the first two dimensions.

**Analytic Comparison** Here we mention specifically how we performed Kernel Density Estimation. Namely, in order to estimate the density $\rho$, we take a batch of samples $\{z_b\}_{b=1}^B$ (during training, this is the training batch). Then at uniformly spaced time-points $\{t_b\}_{b=1} \subseteq [0,1]$, we estimate the density with the formula,

$$\rho(z_b,t_b) \approx \frac{1}{B}\frac{1}{(\sigma h\sqrt{2\pi})^d}\sum_{i=1}^B\sum_{j=1}^B \exp\left(\frac{\|z_i-z_j\|^2}{(h\sigma)^2}\right) \tag{B.3}$$

where we choose $\sigma = \sqrt{\frac{\gamma}{\nu}}$, and $d$ is the dimension, and $h = B^{-\frac{1}{d+4}}$, in accordance with Scott's rule for multivariate kernel density estimation [55].

**Density Splitting via Symmetric Obstacle** Here we compute an example where we have a symmetric obstacle, and thus the generator will learn to split the density. Agents will go left or right of the obstacle depending on their starting position. Here we chose the obstacle as,

$$f(x_1,x_2,\ldots,x_d) = \alpha_{\text{obst}} \max\left\{-v^\top Q v + 0.1, 0\right\}, \quad Q = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}, \; v = (x_1,x_2). \tag{B.4}$$

and we choose $\gamma_{\text{obst}} = 20$. The environment and training parameters are the same as in the main text, except we choose the HJB penalty $\lambda$ in Alg. 1 to be 0.1. Qualitatively, we see the

solution agrees with our intuition: the agents will go left or right depending on their starting position. Note that the results are similar across dimensions, verifying our computation. For the $2d$, $\nu = 0$ case we trained for 100k iterations, for the $2d$, $\nu = 0.1$ case we trained fro 300k iterations, for the $50d$ and $100d$, $\nu = 0$ case we trained for 500k iterations, for the $50d$ $\nu = 0.1$ case we trained for 1000k iterations, and the for the $100d$, $\nu = 0.1$ case we trained for 2000k iterations.

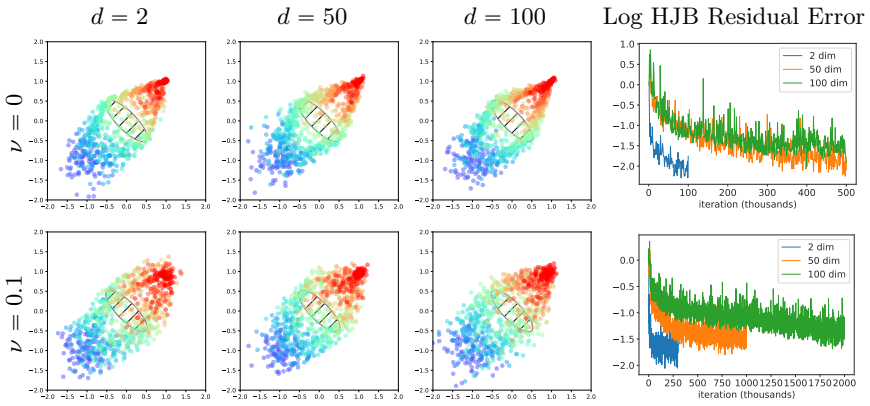


**Figure 7:** Computation of the an obstacle problem where the obstacle is symmetric. We plot the results for dimensions 2 and 100, and for $\nu = 0$ and $\nu = 0.1$.