A Machine Learning Framework for High-Dimensional Mean Field Games

Samy Wu Fung\textsuperscript{1}  Stan Osher\textsuperscript{1}  Levon Nurbekyan\textsuperscript{1}  Wuchen Li\textsuperscript{1}  Lars Ruthotto\textsuperscript{2}

\textsuperscript{1}Department of Mathematics, University of California, Los Angeles  
\textsuperscript{2}Department of Mathematics, Emory University

May 8, 2020
Team and Acknowledgments
Agenda

1. Introduction
2. Lagrangian Formulation
3. A Machine Learning Framework
4. Enforcing the Physics
5. Numerical Results
6. Summary and Outlook
Introduction
Mean Field Games

**Main Idea:** describe the evolution of a population of agents that play a (non-cooperative) differential game on $[0, T]$.

---

1. Y Achdou, J Lasry, 2019
2. D. Firoozi, P. E. Caines, 2017
3. Z Liu, B Wu, H Lin, 2018
4. W E, J Han, and Q Li, 2018
Mean Field Games

**Main Idea:** describe the evolution of a population of agents that play a (non-cooperative) differential game on \([0, T]\).

**Applications:** Crowd Motion\(^1\), Finance\(^2\), Swarm Robotics\(^3\), Data Science\(^4\), and many more...

---

\(^1\) Y Achdou, J Lasry, 2019
\(^2\) D. Firoozi, P. E. Caines, 2017
\(^3\) Z Liu, B Wu, H Lin, 2018
\(^4\) W E, J Han, and Q Li, 2018
Microscopic Model

Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P} (\mathbb{R}^d)$. 

$v : [0, T] \rightarrow \mathbb{R}^d$ is the strategy (control), $z : [0, T] \rightarrow \mathbb{R}^d$ is the state $L : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the running cost (Lagrangian) $F, G : \mathbb{R}^d \times \mathcal{P} (\mathbb{R}^d) \rightarrow \mathbb{R}$ are interaction/terminal costs
Microscopic Model

Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$. Each agent solves

$$\inf_v \int_t^T L(z(s), v(s)) + F(z(s), \rho(z(s), s)) \, ds + G(z(T), \rho(z(T), T)),$$

s.t. $\partial_s z = v(s), \quad t \leq s \leq T, \quad z(t) = x.$
Microscopic Model

Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$.

Each agent solves

$$\inf_v \int_t^T L(z(s), v(s)) + F(z(s), \rho(z(s), s)) \, ds + G(z(T), \rho(z(T), T)),$$

s.t. $\partial_s z = v(s)$, $t \leq s \leq T$, $z(t) = x$.

- $v : [0, T] \rightarrow \mathbb{R}^d$ is the strategy (control),
Microscopic Model

Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$. Each agent solves

$$\inf_{v} \int_{t}^{T} L(z(s), v(s)) + F(z(s), \rho(z(s), s)) ds + G(z(T), \rho(z(T), T)),$$

s.t. $\partial_s z = v(s), \quad t \leq s \leq T, \quad z(t) = x$.

- $v : [0, T] \rightarrow \mathbb{R}^d$ is the strategy (control),
- $z : [0, T] \rightarrow \mathbb{R}^d$ is the state
Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$. Each agent solves

$$\inf_v \int_t^T L(z(s), v(s)) + F(z(s), \rho(z(s), s)) \, ds + G(z(T), \rho(z(T), T)),$$

s.t. $\partial_s z = v(s), \quad t \leq s \leq T, \quad z(t) = x$.

- $v : [0, T] \rightarrow \mathbb{R}^d$ is the strategy (control),
- $z : [0, T] \rightarrow \mathbb{R}^d$ is the state
- $L : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the running cost (Lagrangian)
Microscopic Model

Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$. Each agent solves

$$\inf_v \int_t^T L(z(s), v(s)) + F(z(s), \rho(z(s), s)) ds + G(z(T), \rho(z(T), T)),$$

s.t. $\partial_s z = v(s), \quad t \leq s \leq T, \quad z(t) = x.$

- $v : [0, T] \to \mathbb{R}^d$ is the strategy (control),
- $z : [0, T] \to \mathbb{R}^d$ is the state
- $L : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the running cost (Lagrangian)
- $F, G : \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ are interaction/terminal costs
Microscopic Model

Suppose an agent is at position $x \in \mathbb{R}^d$ at time $t \in [0, T]$. Denote agents’ population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$. Each agent solves

$$\inf_v \int_t^T L(z(s), v(s)) + F(z(s), \rho(z(s), s)) ds + G(z(T), \rho(z(T), T)),$$

s.t. $\partial_s z = v(s), \ t \leq s \leq T, \ z(t) = x.$

- $v : [0, T] \rightarrow \mathbb{R}^d$ is the strategy (control),
- $z : [0, T] \rightarrow \mathbb{R}^d$ is the state
- $L : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the running cost (Lagrangian)
- $F, G : \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \rightarrow \mathbb{R}$ are interaction/terminal costs
Macroscopic Model: PDE Formulation

An equivalent formulation involves the coupled system of PDEs\(^6\)

\[-\partial_t \phi(x, t) + H(x, \nabla \phi(x, t)) = F(x, \rho(x, t)), \quad (HJB)\]

\[\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t)v(x, t)) = 0, \quad (CE)\]

\[\rho(x, 0) = \rho_0(x), \quad \phi(x, T) = G(x, \rho(x, T)),\]

---

\(^5\) J Lasry, P Lions, 2006  
\(^6\) M Huang, R Malhamé, P Caines 2006
Macroscopic Model: PDE Formulation

An equivalent formulation involves the coupled system of PDEs\(^{56}\)

\[- \partial_t \phi(x, t) + H(x, \nabla \phi(x, t)) = F(x, \rho(x, t)), \quad \text{(HJB)}\]
\[\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t)v(x, t)) = 0, \quad \text{(CE)}\]
\[
\rho(x, 0) = \rho_0(x), \quad \phi(x, T) = G(x, \rho(x, T)),
\]

where

- \(\phi: \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}\) is the value function

\(^{5}\) J Lasry, P Lions, 2006

\(^{6}\) M Huang, R Malhamé, P Caines 2006
Macroscopic Model: PDE Formulation

An equivalent formulation involves the coupled system of PDEs\(^\text{56}\)

\[- \partial_t \phi(x, t) + H(x, \nabla \phi(x, t)) = F(x, \rho(x, t)), \quad (\text{HJB})\]
\[\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t)v(x, t))) = 0, \quad (\text{CE})\]
\[\rho(x, 0) = \rho_0(x), \quad \phi(x, T) = G(x, \rho(x, T)),\]

where

- \(\phi: \mathbb{R}^d \times [0, T] \to \mathbb{R}\) is the value function
- \(H: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}\) is the Hamiltonian defined as
  \[H(x, p) = \inf_v L(x, v) + \langle p, v \rangle,\]

\(^{5}\) J Lasry, P Lions, 2006

\(^{6}\) M Huang, R Malhamé, P Caines 2006
Macroscopic Model: PDE Formulation

An equivalent formulation involves the coupled system of PDEs\(^5\)\(^6\)

\[
- \partial_t \phi(x, t) + H(x, \nabla \phi(x, t)) = F(x, \rho(x, t)), \quad \text{(HJB)}
\]

\[
\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t) v(x, t)) = 0, \quad \text{(CE)}
\]

\[
\rho(x, 0) = \rho_0(x), \quad \phi(x, T) = G(x, \rho(x, T)),
\]

where

- \(\phi: \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}\) is the value function
- \(H: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}\) is the Hamiltonian defined as
  \(H(x, p) = \inf_v L(x, v) + \langle p, v \rangle\),
- \(v(x, t) = -\nabla_p H(x, \nabla \phi(x, t))\) (Pontryagin Maximum Principle)

\(^5\) J Lasry, P Lions, 2006

\(^6\) M Huang, R Malhamé, P Caines 2006
An equivalent formulation involves the coupled system of PDEs\(^5\)\(^6\)

\[-\partial_t \phi(x, t) + H(x, \nabla \phi(x, t)) = F(x, \rho(x, t)), \quad \text{(HJB)}\]
\[
\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t)v(x, t))) = 0, \quad \text{(CE)}
\]
\[
\rho(x, 0) = \rho_0(x), \quad \phi(x, T) = G(x, \rho(x, T)),
\]

where

- \(\phi: \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}\) is the value function
- \(H: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}\) is the Hamiltonian defined as
  \[H(x, p) = \inf_v L(x, v) + \langle p, v \rangle,\]
- \(v(x, t) = -\nabla_p H(x, \nabla \phi(x, t))\) (Pontryagin Maximum Principle)
- forward-backward structure + high dimensions \(\implies\) system of PDEs difficult to solve!

\(^5\) J Lasry, P Lions, 2006
\(^6\) M Huang, R Malhamé, P Caines 2006

---

Macroscopic Model: PDE Formulation

An equivalent formulation involves the coupled system of PDEs\(^5\)\(^6\)
Variational Principle for (Potential) MFGs

Another equivalent formulation involves a variational problem\(^7\)\(^8\)

$$\inf_{\rho, \nu} \mathcal{J}_{\text{MFG}}(\nu, \rho) =$$

$$\int_0^T \int_{\mathbb{R}^d} L(x, \nu) \rho(x, t) dx dt + \int_0^T \mathcal{F}(\rho(\cdot, t)) dt + \mathcal{G}(\rho(\cdot, T)),$$

s.t. \(\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t) \nu(x, t)) = 0, \quad \rho(x, 0) = \rho_0(x)\)

where

\[
F(x, \rho) = \frac{\delta \mathcal{F}(\rho)}{\delta \rho}(x), \quad G(x, \rho) = \frac{\delta \mathcal{G}(\rho)}{\delta \rho}(x).
\]

\(^7\) J Lasry, P Lions, 2006

\(^8\) M Huang, R Malhamé, P Caines 2006
Variational Principle for (Potential) MFGs

Another equivalent formulation involves a variational problem\(^7\)\(^8\)

\[
\inf_{\rho,v} J_{\text{MFG}}(v, \rho) = \\
\int_0^T \int_{\mathbb{R}^d} L(x, v) \rho(x, t) dx dt + \int_0^T F(\rho(\cdot, t)) dt + G(\rho(\cdot, T)), \\
\text{s.t. } \partial_t \rho(x, t) + \nabla \cdot (\rho(x, t) v(x, t)) = 0, \quad \rho(x, 0) = \rho_0(x)
\]

where

\[
F(x, \rho) = \frac{\delta F(\rho)}{\delta \rho}(x), \quad G(x, \rho) = \frac{\delta G(\rho)}{\delta \rho}(x).
\]

- PDE formulation are KKT conditions of (1)

---

\(^7\) J Lasry, P Lions, 2006

\(^8\) M Huang, R Malhamé, P Caines 2006
Existing Work

- Y. Achdou. Finite difference methods for mean field games in Hamilton-Jacobi equations: approximations, numerical analysis and applications, 2013

- Y. T. Chow, W. Li, S. Osher, and W. Yin. Algorithm for Hamilton-Jacobi equations in density space via a generalized Hopf formula, 2018

- M. Jacobs, F. Leger, W. Li, and S. Osher. Solving Large-Scale Optimization Problems with a Convergence Rate Independent of Grid Size, 2018

- L. Briceño Arias. On the implementation of a primal-dual algorithm for second order time-dependent mean field games with local couplings in CEMRACS, (2019)
Existing Work

- Y. Achdou. Finite difference methods for mean field games in Hamilton-Jacobi equations: approximations, numerical analysis and applications, 2013

- Y. T. Chow, W. Li, S. Osher, and W. Yin. Algorithm for Hamilton-Jacobi equations in density space via a generalized Hopf formula, 2018

- M. Jacobs, F. Leger, W. Li, and S. Osher. Solving Large-Scale Optimization Problems with a Convergence Rate Independent of Grid Size, 2018

- L. Briceño Arias. On the implementation of a primal-dual algorithm for second order time-dependent mean field games with local couplings in CEMRACS, (2019)

A common difficulty: need to build a grid
Existing Work

- Y. Achdou. Finite difference methods for mean field games in Hamilton-Jacobi equations: approximations, numerical analysis and applications, 2013

- Y. T. Chow, W. Li, S. Osher, and W. Yin. Algorithm for Hamilton-Jacobi equations in density space via a generalized Hopf formula, 2018

- M. Jacobs, F. Leger, W. Li, and S. Osher. Solving Large-Scale Optimization Problems with a Convergence Rate Independent of Grid Size, 2018

- L. Briceño Arias. On the implementation of a primal-dual algorithm for second order time-dependent mean field games with local couplings in CEMRACS, (2019)

**A common difficulty:** need to build a grid

**Today:** develop grid-free scheme to handle high-dimensional MFGs.
Lagrangian Formulation
A Lagrangian Method

- eliminate continuity equation (CE) by discretizing the problem using a Lagrangian method, i.e.,

\[
\partial_t z(x, t) = -\nabla_p H\left(z(x, t), \nabla \Phi(z(x, t))\right)
\]

\[
= -\nabla \Phi(z(x, t), t), \quad z(x, 0) = x
\]

where for simplicity, we assume \( H \) quadratic.
A Lagrangian Method

- eliminate continuity equation (CE) by discretizing the problem using a Lagrangian method, i.e.,

$$\partial_t z(x, t) = -\nabla_p H(z(x, t), \nabla \Phi(z(x, t)))$$

$$= -\nabla \Phi(z(x, t), t), \quad z(x, 0) = x$$

where for simplicity, we assume $H$ quadratic.

- We know that (CE) preserves mass, i.e.,

$$\rho(z(x, t), t) \det(\nabla z(x, t)) = \rho_0(x), \quad \text{for all } t$$
A Lagrangian Method

- eliminate continuity equation (CE) by discretizing the problem using a Lagrangian method, i.e.,

\[
\partial_t z(x, t) = -\nabla_p H\left(z(x, t), \nabla \Phi(z(x, t))\right) \\
= -\nabla \Phi(z(x, t), t), \quad z(x, 0) = x
\]

where for simplicity, we assume \( H \) quadratic.

- We know that (CE) preserves mass, i.e.,

\[
\rho(z(x, t), t) \det(\nabla z(x, t)) = \rho_0(x), \quad \text{for all } t
\]

- Applying the change of variables to running cost in objective:

\[
\implies \int_{\mathbb{R}^d} L(x, -\nabla \Phi) \rho(x, t) dx = \int_{\mathbb{R}^d} L(z(x, t), -\nabla \Phi) \rho_0(x) dx
\]
For other terms in the objective, $\mathcal{F}$ and $\mathcal{G}$, we need to compute the determinant of induced transformation. For $d \gg 1$, intractable!

A Lagrangian Method

- For other terms in the objective, $\mathcal{F}$ and $\mathcal{G}$, we need to compute the determinant of induced transformation. For $d \gg 1$, intractable!

- **Remedy**: evolve the logarithm of the determinant$^9$

$$ \partial_t l(x, t) = -\Delta \Phi(z(x, t), t), \quad l(x, 0) = 0,$$

where

$$ l(x, t) = \log \left( \det \left( \nabla z(x, t) \right) \right) $$

---

A Lagrangian Method

In general, we accumulate running costs incurred by the agents as follows:

\[
\partial_t \begin{pmatrix}
z(x, t) \\
l(x, t) \\
c_L(x, t) \\
c_F(x, t)
\end{pmatrix} = \begin{pmatrix}
-\nabla \Phi(z(x, t), t) \\
-\Delta \Phi(z(x, t), t) \\
L(z(x, t), -\nabla \Phi(z(x, t), t)) \\
F(z(x, t), t)
\end{pmatrix},
\]

where \( z(x, 0) = x \), and the rest initialized with zero.
A Lagrangian Method

In general, we accumulate running costs incurred by the agents as follows:

$$\partial_t \begin{pmatrix} z(x, t) \\ l(x, t) \\ c_L(x, t) \\ c_F(x, t) \end{pmatrix} = \begin{pmatrix} -\nabla \Phi(z(x, t), t) \\ -\Delta \Phi(z(x, t), t) \\ L(z(x, t), -\nabla \Phi(z(x, t), t)) \\ F(z(x, t), t) \end{pmatrix},$$

(2)

where $z(x, 0) = x$, and the rest initialized with zero.

Note:

- we are interested in high-dimensional problems $\implies$ we use Monte-Carlo:

$$\min_\Phi \mathbb{E}_{\rho_0} (c_L(x, T) + c_F(x, T) + G(z(x, T)))$$
A Lagrangian Method

In general, we accumulate running costs incurred by the agents as follows:

\[
\partial_t \begin{pmatrix} z(x, t) \\ l(x, t) \\ c_L(x, t) \\ c_F(x, t) \end{pmatrix} = \begin{pmatrix} -\nabla \Phi(z(x, t), t) \\ -\Delta \Phi(z(x, t), t) \\ L(z(x, t), -\nabla \Phi(z(x, t), t)) \\ F(z(x, t), t) \end{pmatrix},
\]

where \(z(x, 0) = x\), and the rest initialized with zero.

Note:

- we are interested in high-dimensional problems \(\Rightarrow\) we use Monte-Carlo:

\[
\min_{\Phi} \mathbb{E}_{\rho_0} (c_L(x, T) + c_F(x, T) + G(z(x, T))
\]

- we solve (2) for each agent independently \(\Rightarrow\) parallel!
A Lagrangian Method

In general, we accumulate running costs incurred by the agents as follows:

$$\partial_t \begin{pmatrix} z(x, t) \\ l(x, t) \\ c_L(x, t) \\ c_F(x, t) \end{pmatrix} = \begin{pmatrix} -\nabla \Phi(z(x, t), t) \\ -\Delta \Phi(z(x, t), t) \\ L(z(x, t), -\nabla \Phi(z(x, t), t)) \\ F(z(x, t), t) \end{pmatrix},$$

(2)

where $z(x, 0) = x$, and the rest initialized with zero.

**Note:**

- we are interested in high-dimensional problems $\Rightarrow$ we use Monte-Carlo:
  $$\min_{\Phi} \mathbb{E}_{\rho_0} (c_L(x, T) + c_F(x, T) + G(z(x, T)))$$

- we solve (2) for each agent independently $\Rightarrow$ parallel!

- How to deal with $\nabla$ and $\Delta$?
A Machine Learning Framework
A Neural Network Parameterization

- parameterize \( \Phi \) by

\[
\Phi(s, \theta) = w^\top N(s, \theta_N) + \frac{1}{2} s^\top (A + A^\top) s + c^\top s + b,
\]

where \( s = (x, t) \in \mathbb{R}^{d+1} \) is the input feature and \( N(s, \theta_N) : \mathbb{R}^{d+1} \to \mathbb{R}^m \) is a neural network chosen by user
A Neural Network Parameterization

- parameterize $\Phi$ by

$$
\Phi(s, \theta) = w^\top N(s, \theta_N) + \frac{1}{2} s^\top (A + A^\top) s + c^\top s + b,
$$

$$
\theta = (w, \theta_N, \text{vec}(A), c, b),
$$

where $s = (x, t) \in \mathbb{R}^{d+1}$ is the input feature and $N(s, \theta_N): \mathbb{R}^{d+1} \rightarrow \mathbb{R}^m$ is a neural network chosen by user.

**Learning Problem:**

$$
\min_{\theta} \mathbb{E}_{\rho_0} (c_L(x, T) + c_F(x, T) + G(z(x, T)))
$$
A Neural Network Parameterization

- parameterize $\Phi$ by

$$
\Phi(s, \theta) = w^\top N(s, \theta_N) + \frac{1}{2} s^\top (A + A^\top) s + c^\top s + b,
$$

$$
\theta = (w, \theta_N, \text{vec}(A), c, b),
$$

where $s = (x, t) \in \mathbb{R}^{d+1}$ is the input feature and $N(s, \theta_N): \mathbb{R}^{d+1} \to \mathbb{R}^m$ is a neural network chosen by user.

**Learning Problem:**

$$
\min_{\theta} \mathbb{E}_{\rho_0} (c_L(x, T) + c_F(x, T) + G(z(x, T))
$$

Let $x_1, \ldots, x_N \in \mathbb{R}^d$ be random samples from $\rho_0$. The optimization problem can also be written as

$$
\min_{\theta} \frac{1}{N} \sum_{k=1}^{N} (c_L(x_k, T) + c_F(x_k, T) + G(z(x_k, T))
$$
Network Architecture

Recall our base architecture:

\[
\Phi(s, \theta) = w^\top N(s, \theta_N) + \frac{1}{2} s^\top (A + A^\top) s + c^\top s + b,  \\
\theta = (w, \theta_N, \operatorname{vec}(A), c, b),
\]

In our work, we set \(N(s, \theta_N)\) to be a ResNet, i.e., \(N(s, \theta_N) = u_M\), where

\[
\begin{align*}
    u_0 &= \sigma(K_0 s + b_0) \\
    u_1 &= u_0 + h\sigma(K_1 u_0 + b_1) \\
    \vdots & \quad \vdots \\
    u_M &= u_{M-1} + h\sigma(K_M u_{M-1} + b_M),
\end{align*}
\]
Network Architecture

Recall our base architecture:

$$\Phi(s, \theta) = w^\top N(s, \theta_N) + \frac{1}{2} s^\top (A + A^\top) s + c^\top s + b,$$

$$\theta = (w, \theta_N, \text{vec}(A), c, b),$$

(4)

In our work, we set $N(s, \theta_N)$ to be a ResNet, i.e., $N(s, \theta_N) = u_M$, where

$$u_0 = \sigma(K_0 s + b_0)$$
$$u_1 = u_0 + h\sigma(K_1 u_0 + b_1)$$
$$\vdots \quad \vdots$$
$$u_M = u_{M-1} + h\sigma(K_M u_{M-1} + b_M),$$

(5)

**Required Ingredients:** $\nabla_s \Phi(s, \theta)$ and $\Delta_s \Phi(s, \theta)$. 
Gradient Computation

The gradient of the potential is given by

$$\nabla_s \Phi(s, \theta) = \nabla_s N(s, \theta_N)w + (A + A^T)s + c,$$

(6)

where $\nabla_s N(s, \theta_N)w = z_0$ is obtained by backpropagation:

$$z_M = w + hK_M^\top \text{diag}(\sigma'(K_Mu_{M-1} + b_M))w,$$

$$: \quad :$$

$$z_1 = z_2 + hK_1^\top \text{diag}(\sigma'(K_1u_0 + b_1))z_2,$$

$$z_0 = K_0^\top \text{diag}(\sigma'(K_0s + b_0))z_1.$$
Laplacian Computation

The Laplacian of the potential is given by

$$\Delta \Phi(s, \theta) = \text{tr} \left( E^T (\nabla_s^2 (N(s, \theta_N)w) + (A + A^T))E \right),$$

where the columns of $E \in \mathbb{R}^{d+1 \times d}$ are given by the first $d$ standard basis vectors in $\mathbb{R}^{d+1}$. 
Laplacian Computation

The Laplacian of the potential is given by

$$\Delta \Phi(s, \theta) = \text{tr} \left( E^\top \left( \nabla_s^2 (N(s, \theta_N)w) + (A + A^\top) \right) E \right),$$

where the columns of $E \in \mathbb{R}^{d+1 \times d}$ are given by the first $d$ standard basis vectors in $\mathbb{R}^{d+1}$.

The trace of the first layer of $N(s, \theta_N)$ can be computed as

$$t_0 = \left( \sigma''(K_0 s + b_0) \odot z_1 \right)^\top \left( (K_0 E) \odot (K_0 E) \right) 1,$$

and we continue with the remaining rows in reverse order to obtain

$$\Delta (N(s, \theta_N)w) = t_0 + h \sum_{i=1}^{M} t_i,$$

$$t_i = \left( \sigma''(K_i u_{i-1} + b_i) \odot z_{i+1} \right)^\top \left( (K_i J_{i-1}) \odot (K_i J_{i-1}) \right) 1,$$

where

$$J_{i-1} = \nabla_s u_{i-1}^\top \in \mathbb{R}^{m \times d}.$$
Laplacian Computation

Remarks:

- Trace computed in one forward pass, where Jacobian matrices are updated and overwritten
Laplacian Computation

Remarks:

- Trace computed in one forward pass, where Jacobian matrices are updated and overwritten
- At each layer, cost of updating Jacobian is $O(m^2 \cdot d)$ FLOPS

Note: AD to obtain $\nabla s \Phi$, $\Delta s \Phi$ at least as expensive as direct computation done?
Laplaceian Computation

Remarks:

- Trace computed in one forward pass, where Jacobian matrices are updated and overwritten.
- At each layer, cost of updating Jacobian is $O(m^2 \cdot d)$ FLOPS.
- For $M$ layers, total cost of trace computation: $O(m^2 \cdot d \cdot M)$ FLOPS $\implies$ use deep instead of wide networks.
Laplacian Computation

Remarks:

- Trace computed in one forward pass, where Jacobian matrices are updated and overwritten.
- At each layer, cost of updating Jacobian is $O(m^2 \cdot d)$ FLOPS.
- For $M$ layers, total cost of trace computation: $O(m^2 \cdot d \cdot M)$ FLOPS $\Rightarrow$ use deep instead of wide networks.
- Note: AD to obtain $\nabla_s \Phi, \Delta_s \Phi$ at least as expensive as direct computation.
Remarks:

- Trace computed in one forward pass, where Jacobian matrices are updated and overwritten
- At each layer, cost of updating Jacobian is $O(m^2 \cdot d)$ FLOPS
- For $M$ layers, total cost of trace computation: $O(m^2 \cdot d \cdot M)$ FLOPS $\Rightarrow$ use deep instead of wide networks
- Note: AD to obtain $\nabla_s \Phi, \Delta_s \Phi$ at least as expensive as direct computation
- **done?**
2D OT Example (2 RK steps)

\[ \rho_0, \text{ initial density} \]
\[ \rho_1, \text{ target density} \]
\[ \text{push forward} \]
\[ \text{pull back} \]
\[ \text{characteristics} \]

Remark:
characteristics not straight!
2D OT Example (2 RK steps)

Remark: characteristics not straight!
2D OT Example (8 RK steps)

- $\rho_0$, initial density
- $\rho_1$, target density
- pull back
- push forward
- characteristics

Remark: characteristics not straight!
Enforcing the Physics
Enforcing the Physics

**Idea:** Add penalty that enforces HJB equations

\[
\inf_{\rho, \Phi} \mathcal{J}_{\text{MFG}} (-\nabla \rho H(x, \nabla \Phi), \rho) + C(\Phi, \rho),
\]

where

\[
C(\Phi, \rho) = \alpha_1 \int_0^T \int_{\mathbb{R}^d} |\partial_t \Phi(x, t) - H(x, \nabla \Phi(x, t)) + F(x, \rho(x, t))| \rho(x, t) \, dx \, dt
\]

\[
+ \alpha_2 \int_{\mathbb{R}^d} |\Phi(x, T) - G(x, \rho(x, T))| \rho(x, t) \, dx.
\]
Enforcing the Physics

**Idea:** Add penalty that enforces HJB equations

\[
\inf_{\rho, \Phi} \mathcal{J}_{\text{MFG}}(-\nabla_p H(x, \nabla \Phi), \rho) + C(\Phi, \rho),
\]

where

\[
C(\Phi, \rho) = \alpha_1 \int_0^T \int_{\mathbb{R}^d} \left( |\partial_t \Phi(x, t) - H(x, \nabla \Phi(x, t)) + F(x, \rho(x, t))| \right) \rho(x, t) dx dt
\]

\[
+ \alpha_2 \int_{\mathbb{R}^d} \left( |\Phi(x, T) - G(x, \rho(x, T))| \right) \rho(x, t) dx.
\]

and its Lagrangian Formulation is given by

\[
\partial_t \begin{pmatrix}
z(x, t) \\
l(x, t) \\
c_L(x, t) \\
c_F(x, t) \\
c_{\text{HJB}}(x, t)
\end{pmatrix} = \begin{pmatrix}
-\nabla \Phi(z(x, t), t) \\
-\Delta \Phi(z(x, t), t) \\
L(z(x, t), -\nabla \Phi(z(x, t), t)) \\
F(z(x, t), t) \\
c_1(z(x, t), t)
\end{pmatrix},
\]
2D OT Example

HJB penalty improves accuracy and(!) lowers computational costs
Numerical Results
OT Example for Different Dimensions

\( \rho_0, \) initial density
\( \rho_1, \) target density
\( d = 2, \) pull back
\( d = 2, \) push fwd
\( d = 2, \) characteristics
\( d = 10, \) pull back
\( d = 10, \) push fwd
\( d = 10, \) characteristics
\( d = 50, \) pull back
\( d = 50, \) push fwd
\( d = 50, \) characteristics
\( d = 100, \) pull back
\( d = 100, \) push fwd
\( d = 100, \) characteristics

qualitatively similar results in all dimensions
OT Example for Different Dimensions

<table>
<thead>
<tr>
<th>$d$</th>
<th>$N$</th>
<th>$\mathcal{L}$</th>
<th>$\mathcal{G}$</th>
<th>$\mathcal{C}_{HJB}$</th>
<th>time/iter (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1,024</td>
<td>1.08e+01</td>
<td>1.41e-01</td>
<td>1.53e+00</td>
<td>1.437</td>
</tr>
<tr>
<td>10</td>
<td>4,096</td>
<td>1.08e+01</td>
<td>1.85e-01</td>
<td>1.25e+00</td>
<td>8.408</td>
</tr>
<tr>
<td>50</td>
<td>16,384</td>
<td>1.10e+01</td>
<td>2.41e-01</td>
<td>4.85e+00</td>
<td>65.706</td>
</tr>
<tr>
<td>100</td>
<td>36,864</td>
<td>1.11e+01</td>
<td>3.22e-01</td>
<td>7.37e+00</td>
<td>283.259</td>
</tr>
</tbody>
</table>

moderate growth in runtime across dimensions
Comparing Optimization Algorithms

BFGS converges faster and reduces HJB penalty cost more substantially
Crowd Motion Example for Different Dimensions

Initial density, $\rho_0$

Preference and characteristics

Final density, $\rho_1$

Push-forward of $\rho_0$

Input data results, $d = 2$

Results, $d = 10$

Results, $d = 50$

Results, $d = 100$
Crowd Motion Example for Different Dimensions

Table: Overview of numerical results for instances of the optimal transport and crowd motion problem in growing space dimensions.
Verifying (2d) Solutions

Haber E, Horesh, R, 2015
Verifying (2d) Solutions

Haber E, Horesh, R, 2015

Wu Fung, Nurbekyan, Osher, Li, Ruthotto
Summary:

- developed a machine learning framework that gives a promising new direction toward much-anticipated large-scale applications of MFG
- neural networks are informed by underlying MFG theory \(\implies\) easier to train!
- more details can be found in\(^{11}\)
- high-dimensional stochastic MFGs via GANs\(^{12}\)

Outlook:

- develop better network architectures
- solve more realistic MFGs
- generative modeling

\(^{11}\)Ruthotto L, Osher S, Li W, Nurbekyan L, Wu Fung S. PNAS, 2020