Learning to accelerate the simulation of PDEs

Tailin Wu

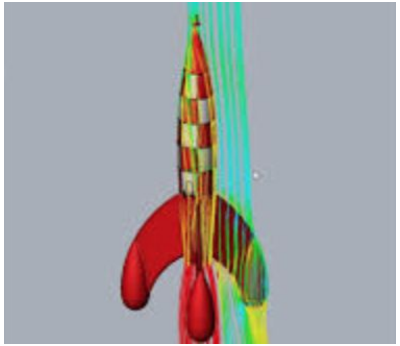
Mar 10

Collaborators:
Stanford: Sophia Kivelson, Jacqueline Yau, Rex Ying, Jure Leskovec
SLAC: Jason Chou, Frederico Fiuza
UCLA: Paulo Alves

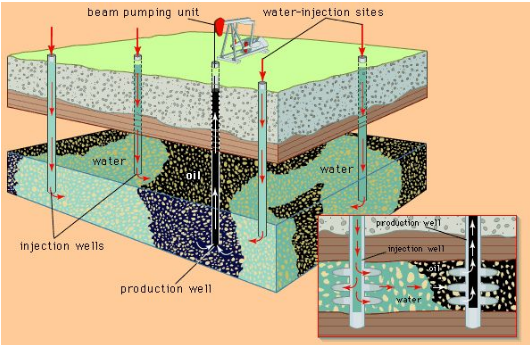
Simulations in science and engineering

1. **PDEs (on a fixed grid or mesh)**

Weather prediction



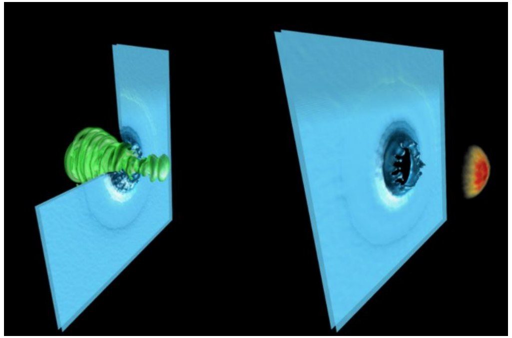
Aerodynamics
for Rocket



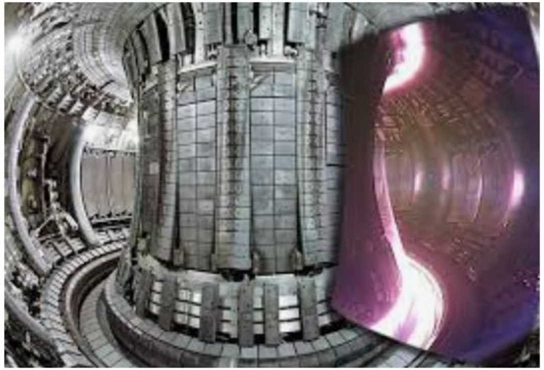
Oil production

Simulations in science and engineering

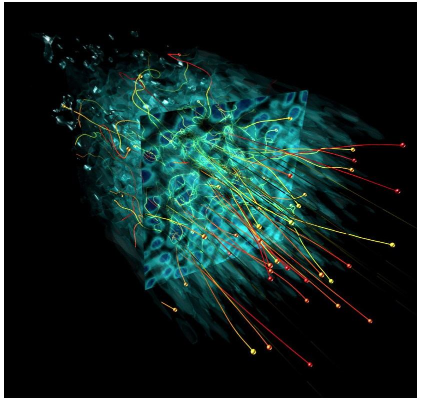
2. Particle-in-Cell (involves both grid and particles)



Laser-plasma
particle acceleration



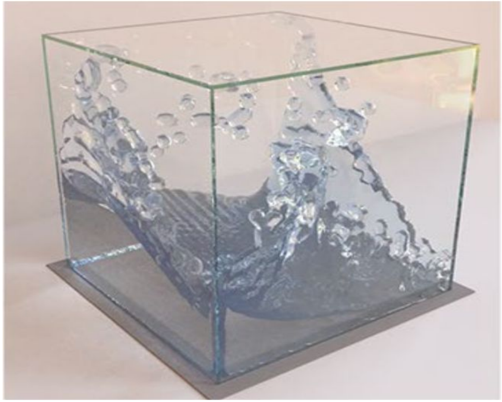
Fusion



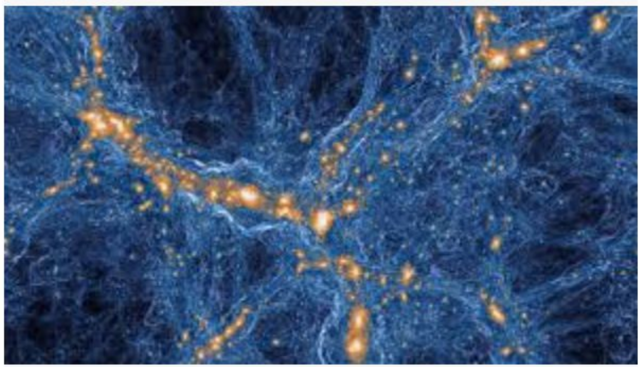
Cosmic-ray
acceleration

Simulations in science and engineering

3. Particle-particle (graph-based):



Water simulation



Galaxy formation

Simulations in science and engineering

Characteristics:

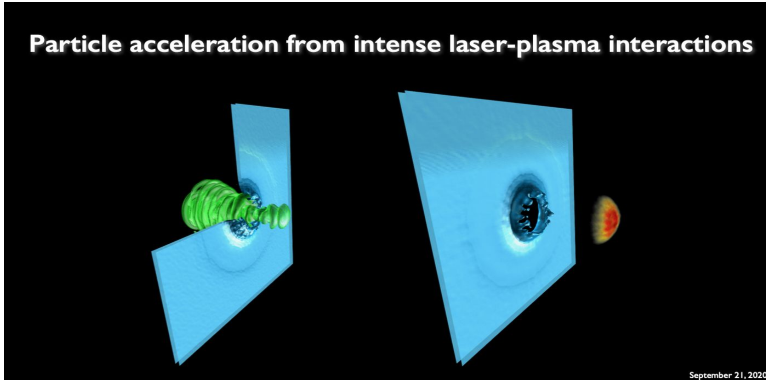
- **Large scale in size:** at the forefront of HPC
  - Nevertheless, even those large compute with long-time simulation may only do reasonably **small systems** in practice
  - E.g. for a reasonable 3D laser-plasma interaction system, it has 100B grid vertices, 1T particles, over 100k time steps
  - Largest simulations (1/year): 10^{-1} of that scale, most studies: < 10^{-2} of that scale

Simulations in science and engineering

Characteristics:

- **Multi-scale and large dynamic range**
  - The dynamics involves multiple scales, and cannot be simulated faithfully only considering the largest scale
  - Kinetic, many-body processes operating at microscopic scales significantly influence the fluid dynamics at large scales (and vice-versa)

E.g. Only ~0.01% of the particles are accelerated but can carry 10-50% of system energy



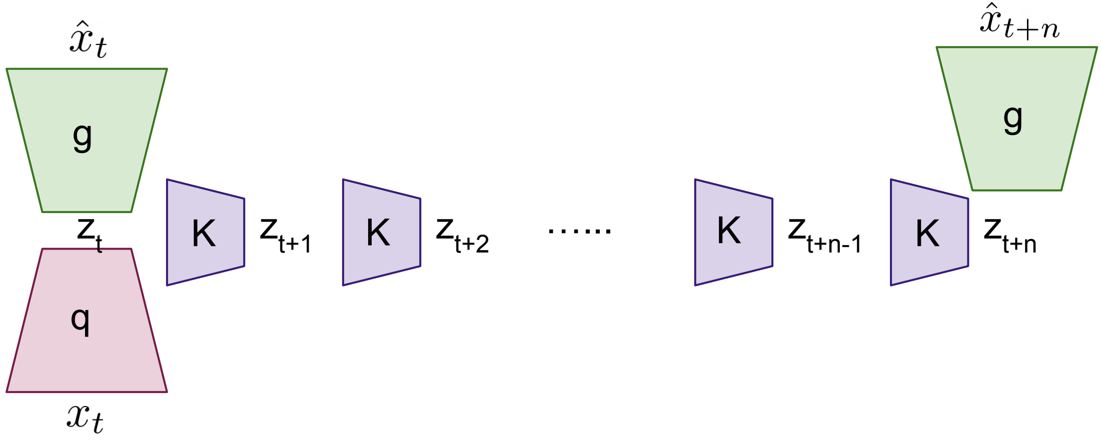
Opportunity for optimization!

Goal:

For large-scale PDE systems, can we design **accurate** and **generalizable** ML models that capture the **essential dynamics** of the system with significant **speed ups**?

Strategy: Latent evolution of PDEs (LE-PDE)

Compress the input into some *suitable* **latent space**, and evolve the dynamics **fully in the latent space**.



If the dimension of the latent representation << dimension of input, we can speed up the evolution

Strategy: Latent evolution of PDEs (LE-PDE)

Compress the input into some *suitable* **latent space**, and evolve the dynamics **fully in the latent space**.

- How to compress the input into latent representation?

Architecture

E.g.
  - GNN
  - CNN
  - LSTM

Based on characteristics of the input, prior knowledge

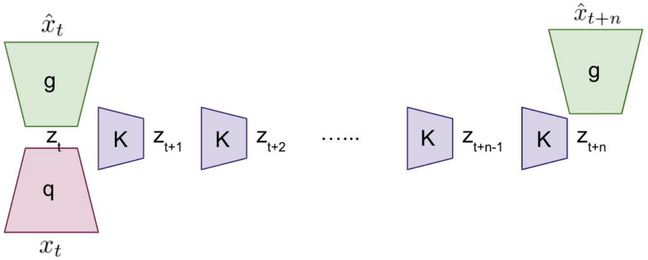
- How to let the ML model learn the dynamics faithfully?

Objective function

E.g.
  - Main objective
  - Regularization
  - Auxiliary objective

Difficulty: during rollout, small error in the learned evolution model will accumulate

Objective of LE-PDE

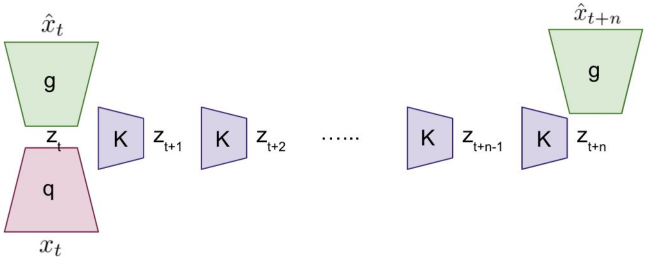


$$L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$$

(1) $$L_{\text{1-step}} = \mathbb{E}_{\mathbf{x} \sim P(\mathbf{x})} \left[||g \circ K \circ q(x_t) - x_{t+1}||_2^2 \right]$$

(2) $$L_{\text{recons}} = \mathbb{E}_{\mathbf{x} \sim P(\mathbf{x})} \left[||g \circ q(x_t) - x_t||_2^2 \right]$$

Objective of LE-PDE



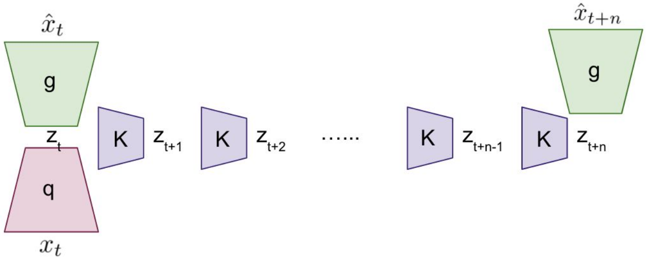
$$L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$$

(3) Latent consistency: $$L_{\text{consistency}} = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\left[\sum_{i=1}^{M} \frac{||K^{(i)} \circ q(x_t) - q(x_{t+i})||_2^2}{||q(x_{t+i})||_2^2}\right]$$

Comments:

- Encourages that the evolved latent representation by K applying multiple times is consistent with the encoding of future target.
- The denominator ensures that the network q cannot “cheat” by simply multiplying with a small scaler.
- If latent dimension << input dimension, this loss component is much more efficient than computing loss in input space.

Objective of LE-PDE



$$L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$$

(4) Spectral normalization:

Let $h\colon Z \to Z$ be an invertible function

$$z_t = q(x_t) \qquad z'_t = q'(x_t) = h \circ q(x_t)$$

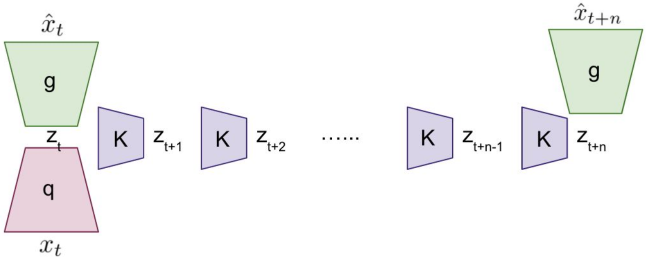
$$\hat{x}_t = g(z_t) \qquad \hat{x}_t = g'(z'_t) = g \circ h^{-1}(z'_t)$$

$$z_{t+1} = K(z_t) \qquad z'_{t+1} = K'(z'_t) = h \circ K \circ h^{-1}(z'_t)$$

Among infinite choice of q, g, K, which one is better?

During inference time, we want to do $g \circ K^{(T)} \circ q(x_t)$, the error in K will accumulate with increasing T, we want K to be generalizable.

Objective of LE-PDE



$$L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$$

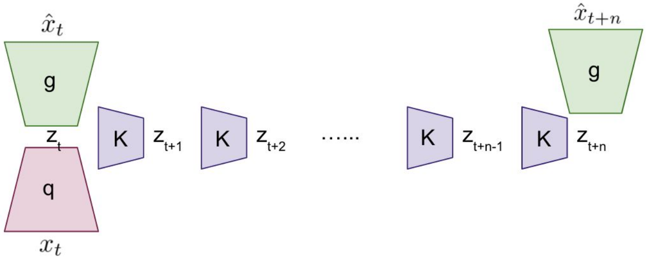
(4) Spectral normalization:

Intuition: we want network K to be smooth and not crazy wiggled (otherwise needs much more data to learn)

Quantitative: We want the Lipschitz constant $\|K\|_{Lip}$ of function K to be small

$$d\big(K(z_1), K(z_2)\big) \leq \|K\|_{Lip} d(z_1, z_2), \qquad \forall z_1, z_2$$

Objective of LE-PDE



$$L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$$

(4) Spectral normalization:

derivation from [2]

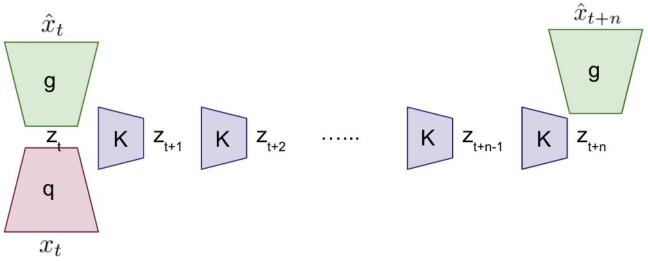
$$K(\mathbf{x}) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(\ldots a_1(W^1 \boldsymbol{x})\ldots)))),$$

$$||K||_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^L \boldsymbol{h}_{L-1})\|_{\text{Lip}}$$
$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^1 \boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^l \boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

$\sigma(A)$: Spectral norm of matrix A (= largest singular value of A)

[1] Yoshida, Yuichi, and Takeru Miyato. *arXiv:1705.10941* (2017).
[2] Miyato, Takeru, et al. "Spectral normalization for generative adversarial networks." arXiv preprint arXiv:1802.05957 (2018).
[3] Sanyal, Amartya, Philip HS Torr, and Puneet K. Dokania. *arXiv:1906.04659* (2019).

Objective of LE-PDE



$$L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$$

(4) Spectral normalization:

$$R_{\text{sn}} = \sum_{l=1}^{L} \sigma(W^l)^2$$

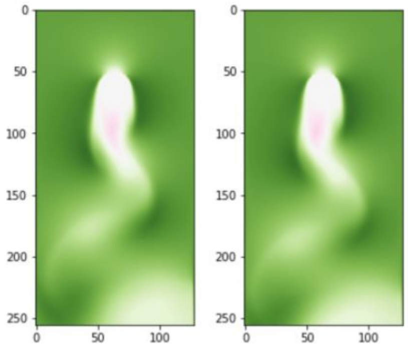
[1][2][3] also shows that regularizing the **spectral norm** improves generalization in deep neural networks and GANs.

[1] Yoshida, Yuichi, and Takeru Miyato. *arXiv:1705.10941* (2017).
[2] Miyato, Takeru, et al. "Spectral normalization for generative adversarial networks." arXiv preprint arXiv:1802.05957 (2018).
[3] Sanyal, Amartya, Philip HS Torr, and Puneet K. Dokania. *arXiv:1906.04659* (2019).

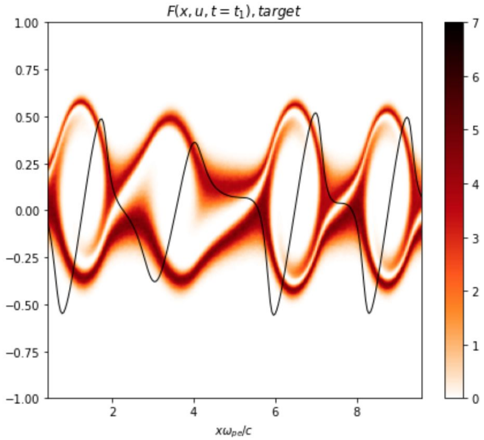
Goal:

For large-scale PDE systems, can we design **accurate** and **generalizable** ML models that capture the **essential dynamics** of the system with significant **speed ups**?

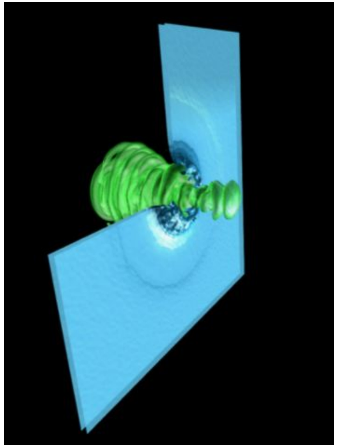
Navier-Stokes equation



Plasma 2-stream Vlasov equation



Plasma full Vlasov equation, simulated by Particle-in-Cell



System 1: Incompressible Navier-Stokes in 2D:

$$\frac{\partial u_x}{\partial t} + \boldsymbol{u} \cdot \nabla u_x = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla u_x$$

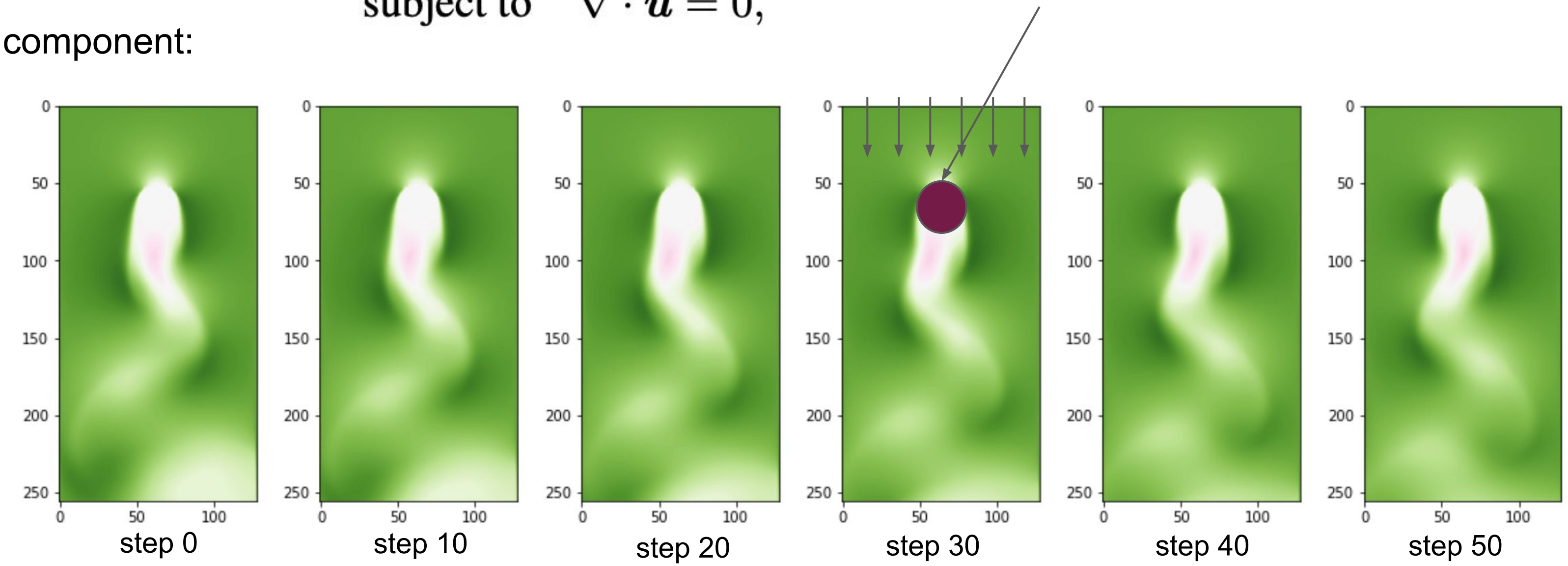
$$\frac{\partial u_y}{\partial t} + \boldsymbol{u} \cdot \nabla u_y = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla u_y$$

$$\text{subject to} \quad \nabla \cdot \boldsymbol{u} = 0,$$

unsteady wake-flow through a cylinder

Generated using [PhiFlow]()

u_x component:



System 1: Incompressible Navier-Stokes in 2D:

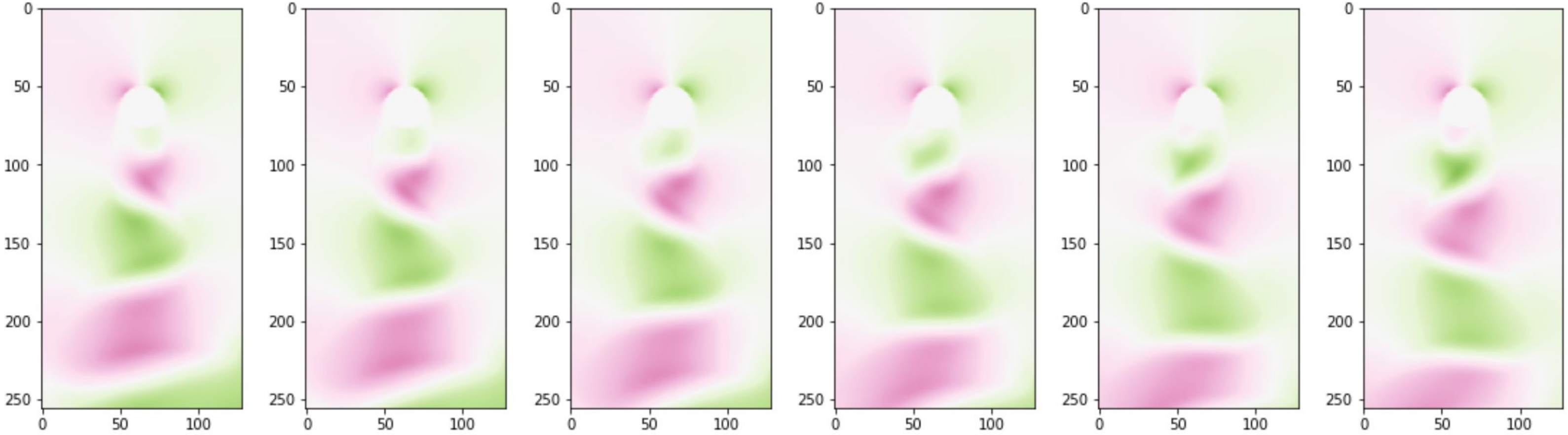
$$\frac{\partial u_x}{\partial t} + \boldsymbol{u} \cdot \nabla u_x = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla u_x$$

$$\frac{\partial u_y}{\partial t} + \boldsymbol{u} \cdot \nabla u_y = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla u_y$$

$$\text{subject to} \quad \nabla \cdot \boldsymbol{u} = 0,$$

dynamic time scale $\tau = \frac{D_{\text{cynlinder}}}{u_x} = 25 \text{ steps}$

u_y component:



step 0 step 10 step 20 step 30 step 40 step 50

System 1: Incompressible Navier-Stokes in 2D:

$$\frac{\partial u_x}{\partial t} + \boldsymbol{u} \cdot \nabla u_x = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla u_x$$
$$\frac{\partial u_y}{\partial t} + \boldsymbol{u} \cdot \nabla u_y = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla u_y$$
$$\text{subject to} \quad \nabla \cdot \boldsymbol{u} = 0,$$

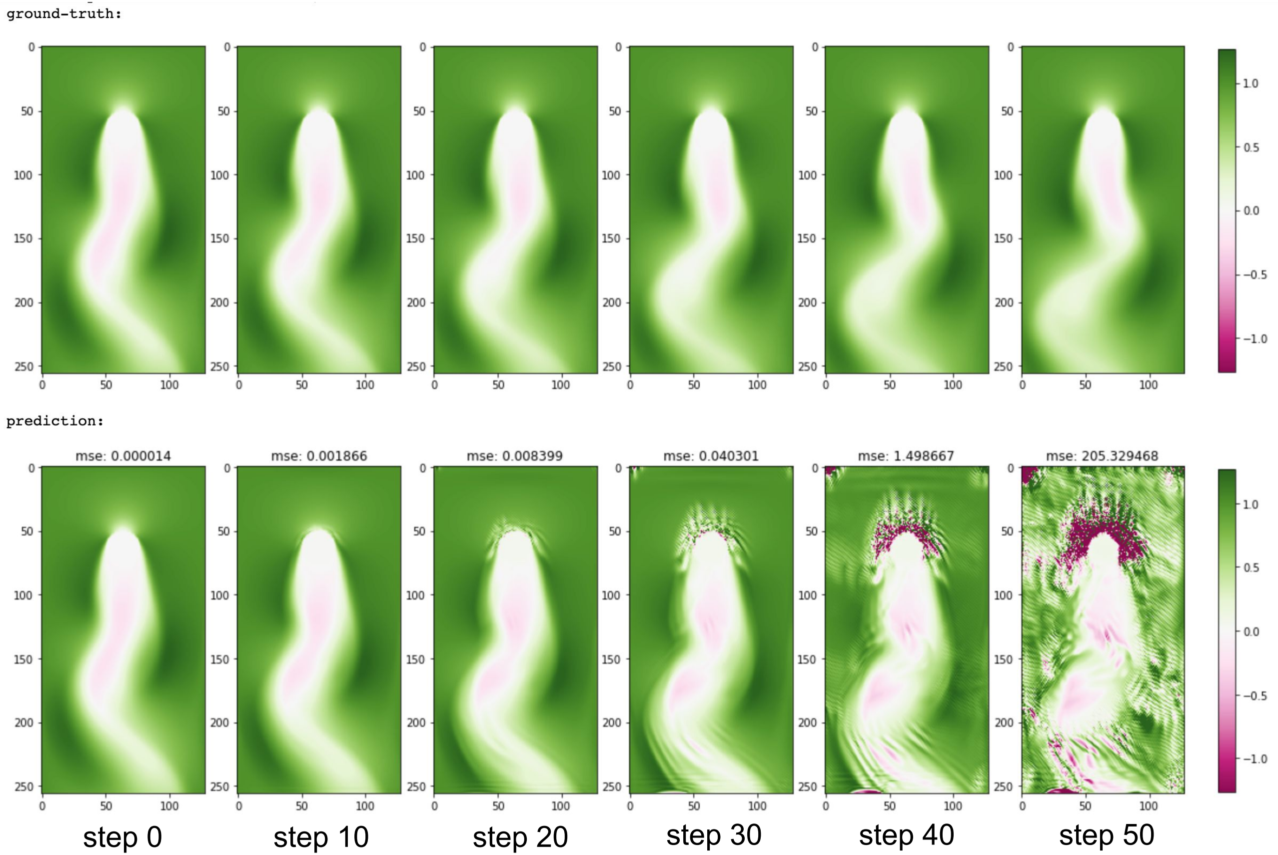
Architecture:

Encoder: CNN + MLP
Evolution model: MLP
Decoder: MLP + CNN

Input dimension: 256 x 128
Latent dimension: 16
2048-fold dimension reduction

System 1: Incompressible Navier-Stokes in 2D:

Failed case #1:



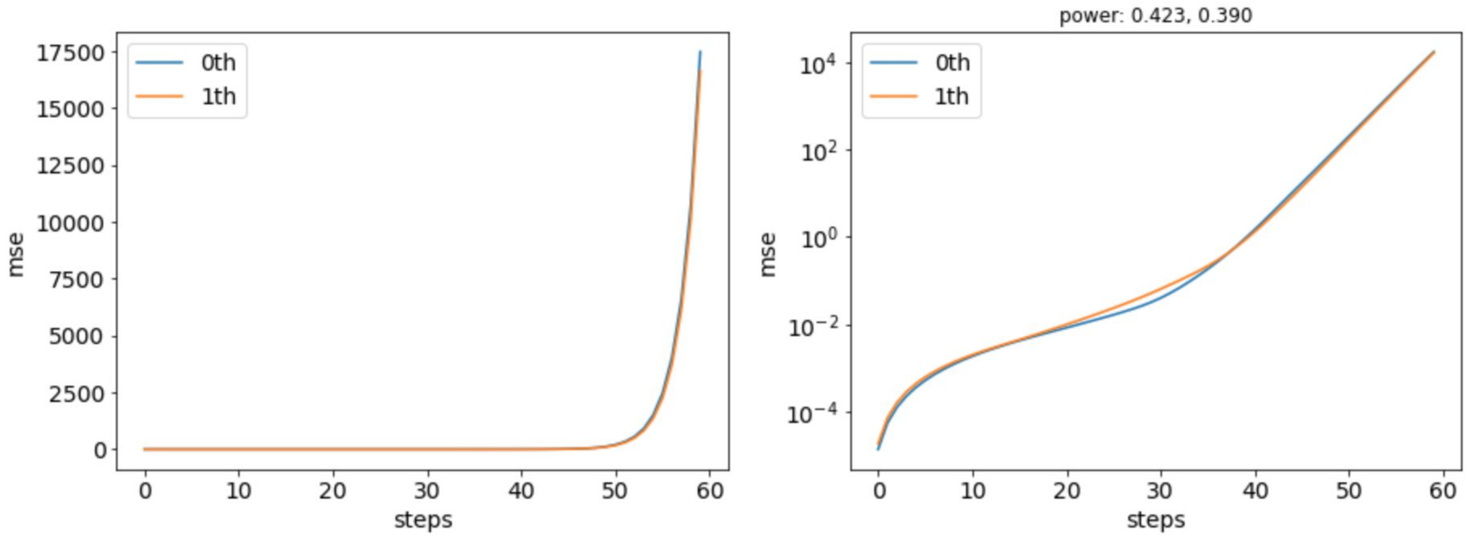
use diff

System 1: Incompressible Navier-Stokes in 2D:

Failed case #1:

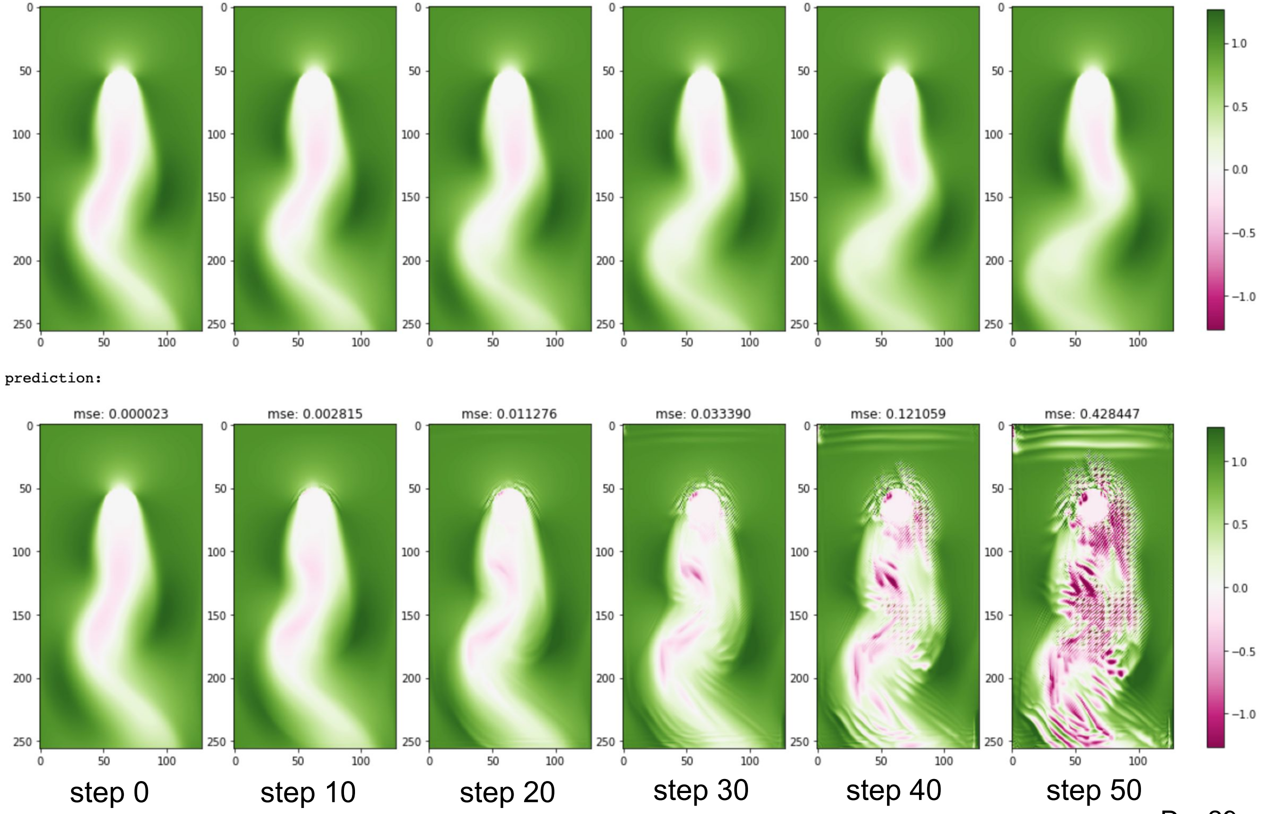
MSE vs. rollout steps:

`MSE, cumulative:`



System 1: Incompressible Navier-Stokes in 2D:

Failed case #2:



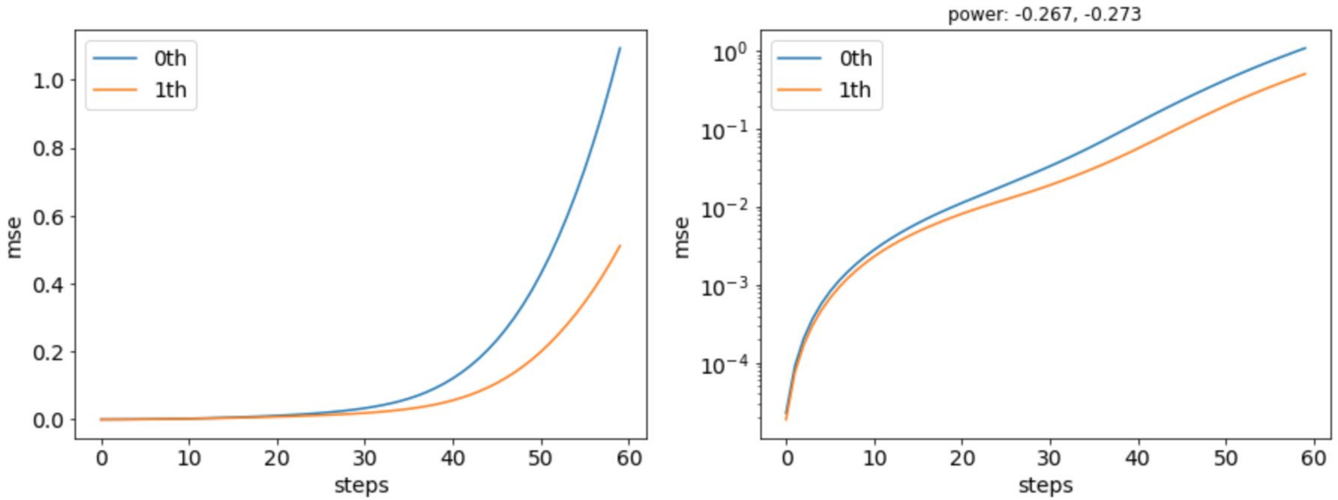
Dec29, elu, diff

System 1: Incompressible Navier-Stokes in 2D:

Failed case #2:

MSE vs. rollout steps:

```
MSE, cumulative:
```

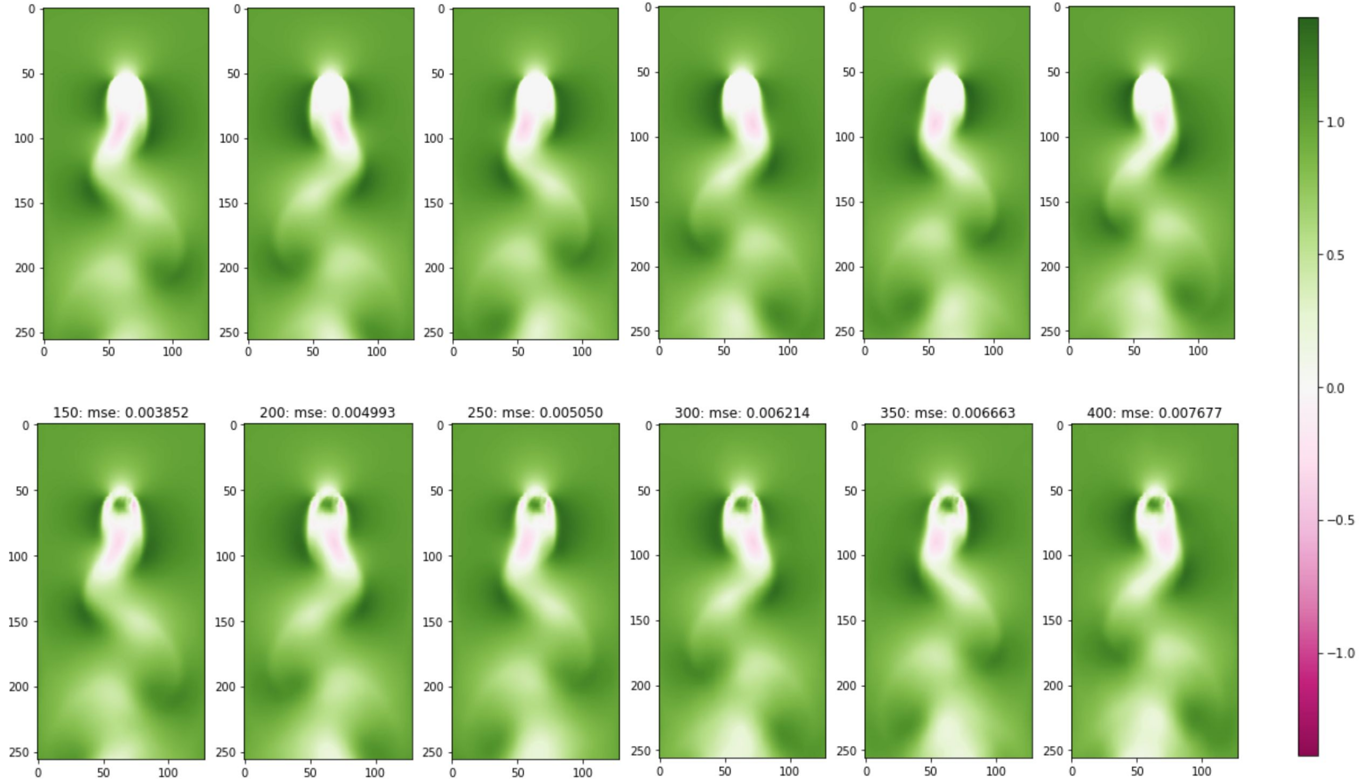


System 1: Incompressible Navier-Stokes in 2D

A. Full objective:

Ground-truth:

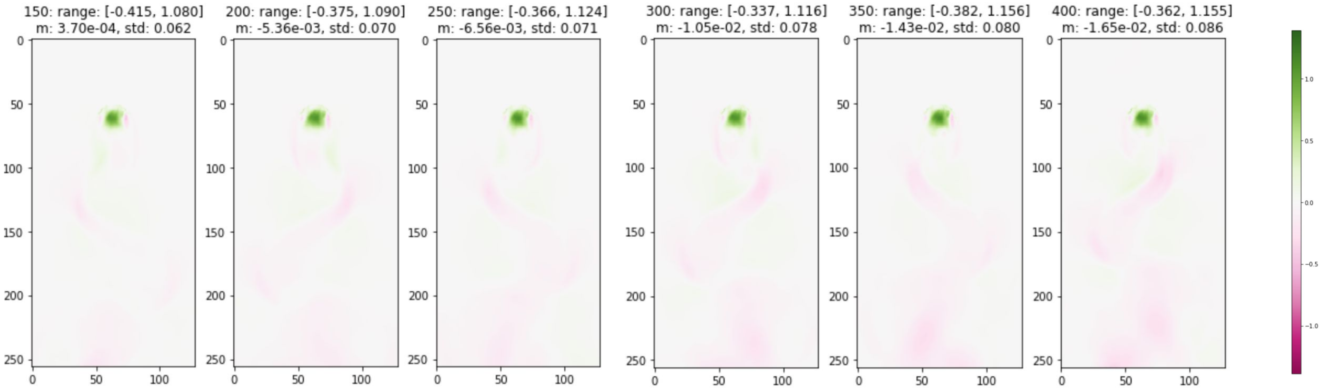
Model rollout:



System 1: Incompressible Navier-Stokes in 2D

A. Full objective:

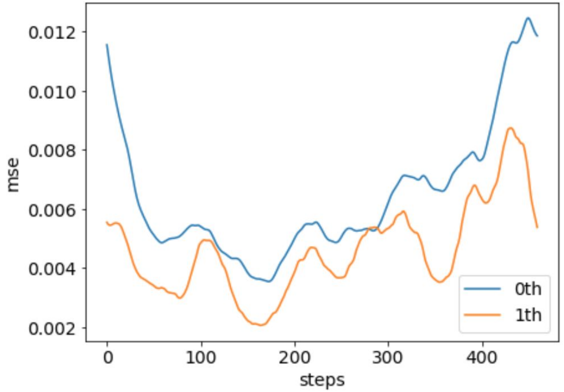
difference:

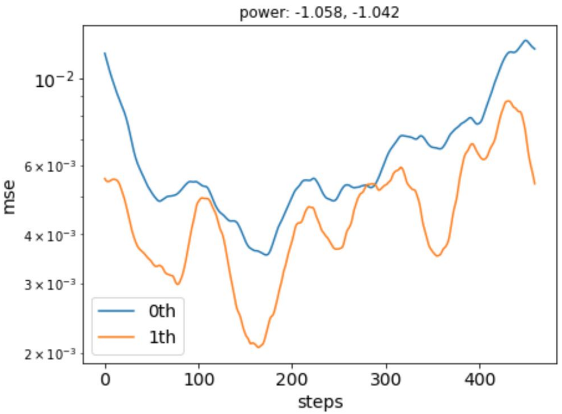


System 1: Incompressible Navier-Stokes in 2D

A. Full objective:

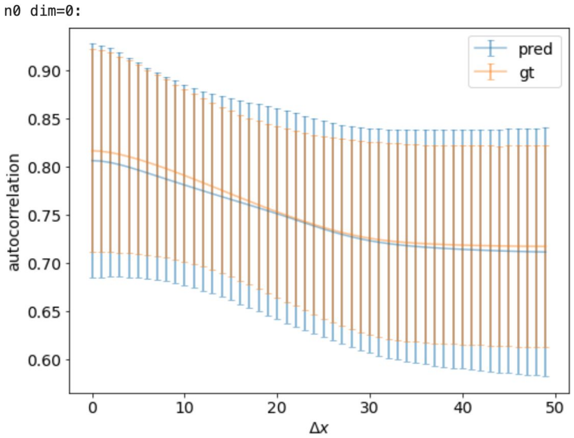
MSE vs. rollout steps:

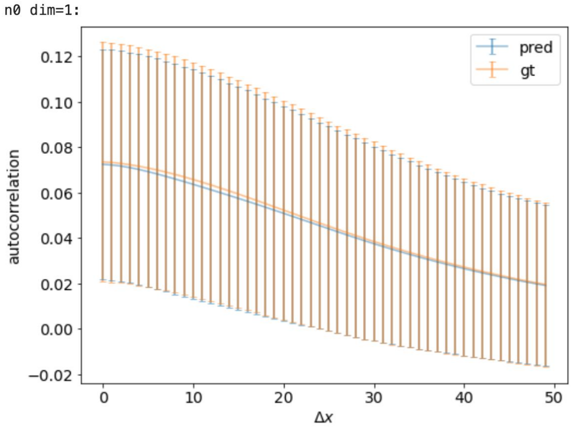




18 dynamical time scale

Autocorrelation:

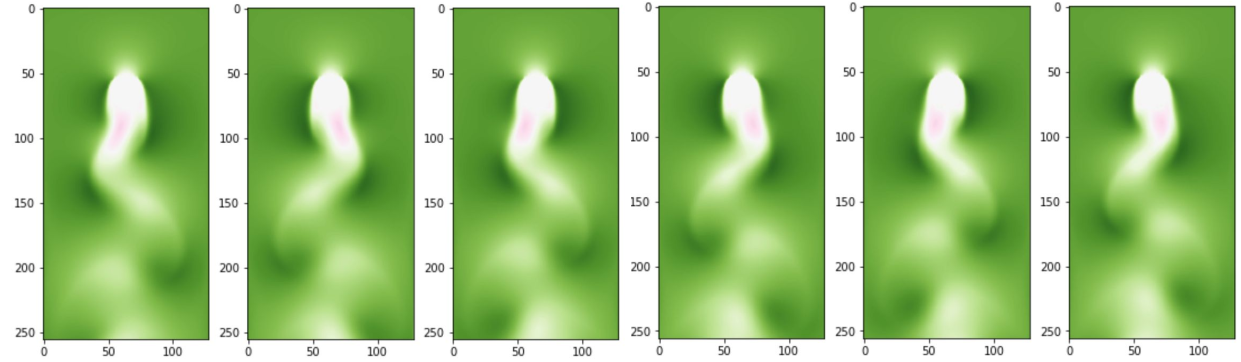




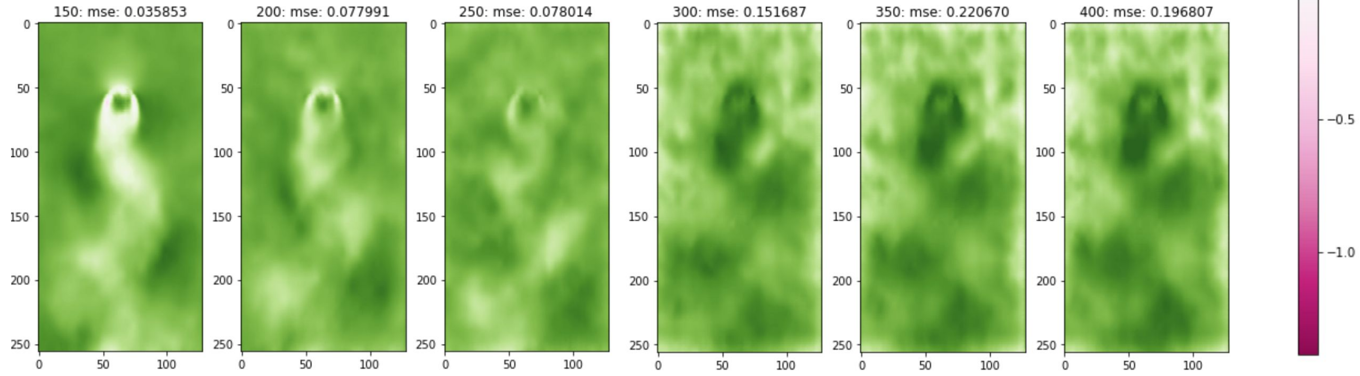
System 1: Incompressible Navier-Stokes in 2D

B. Without latent evolution:

Ground-truth:



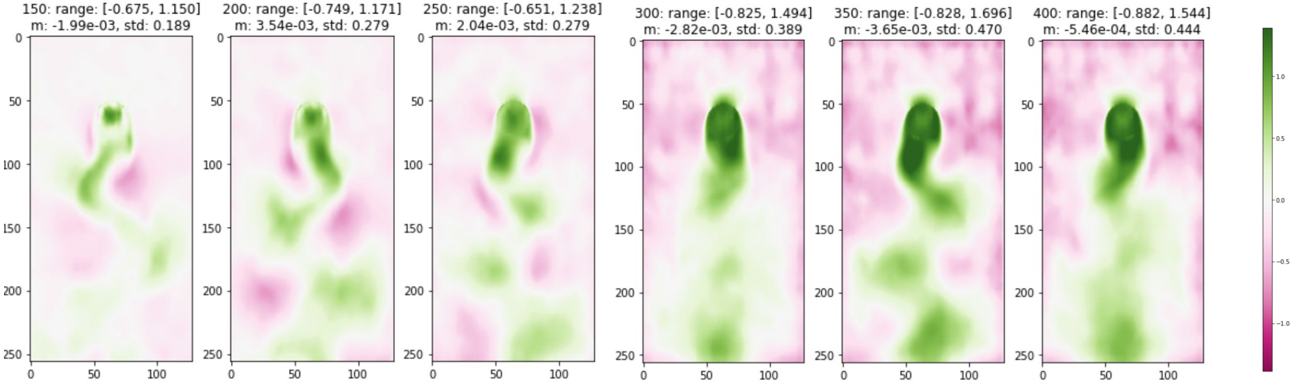
Model rollout:



System 1: Incompressible Navier-Stokes in 2D

B. Without latent evolution:

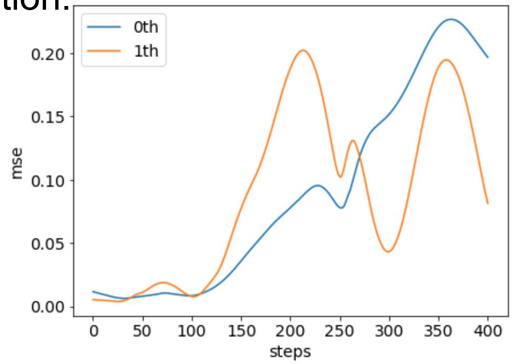
difference:

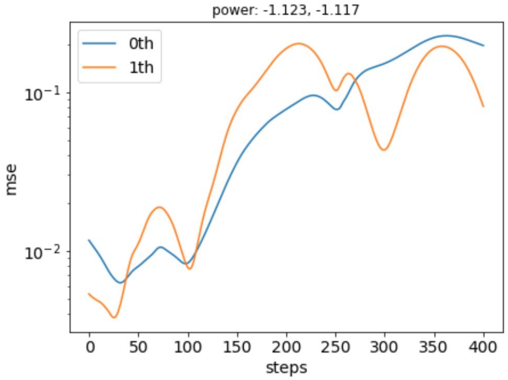


System 1: Incompressible Navier-Stokes in 2D

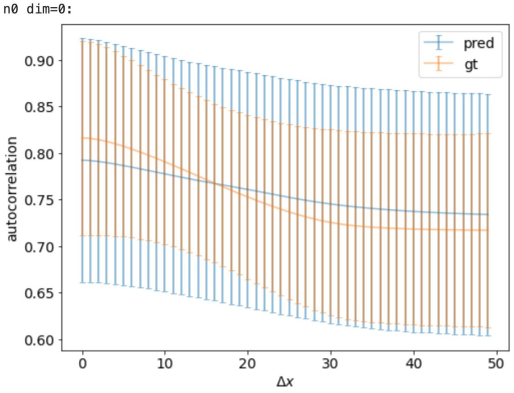
B. Without latent evolution:

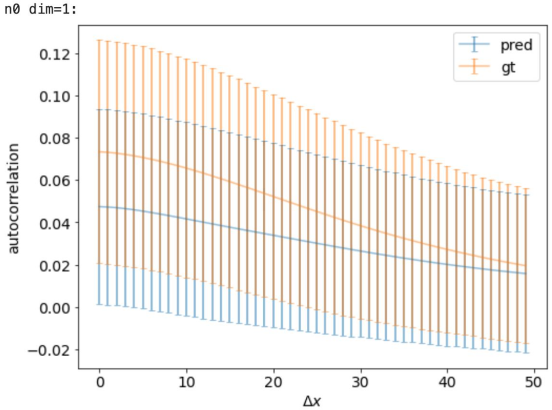
MSE vs. rollout steps:





Autocorrelation:



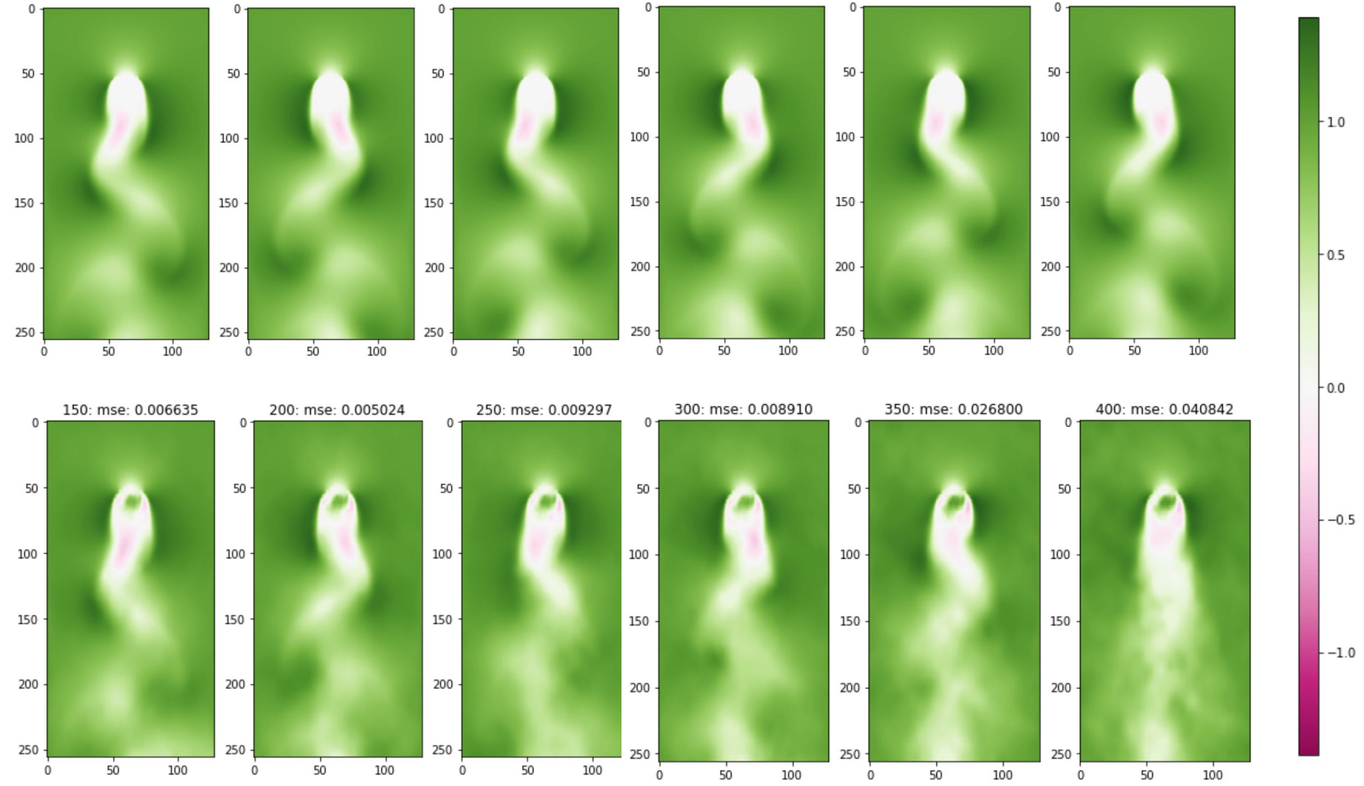


System 1: Incompressible Navier-Stokes in 2D

C. $L_{\text{consistency}}$ without normalization:

Ground-truth:

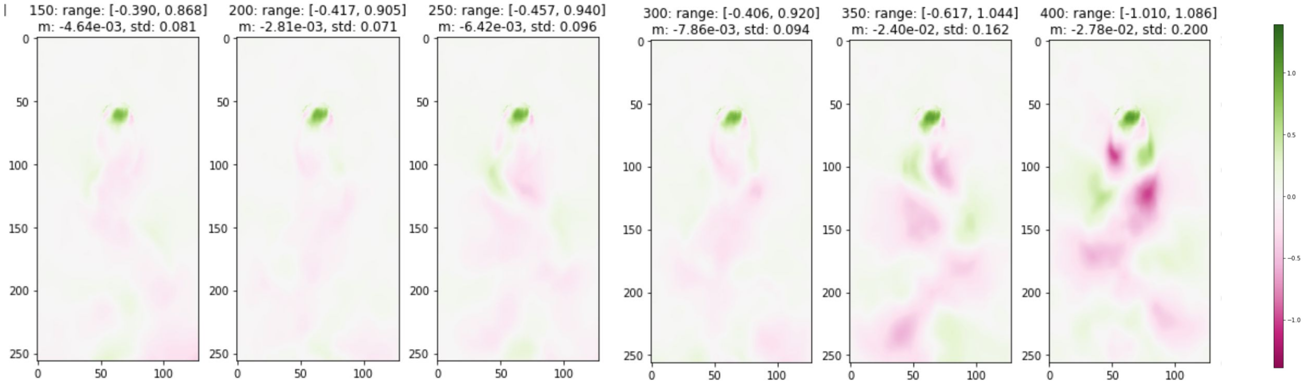
Model rollout:



System 1: Incompressible Navier-Stokes in 2D

C. $L_{\text{consistency}}$ without normalization:

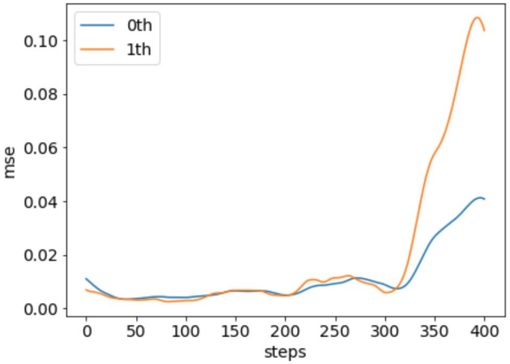
difference:

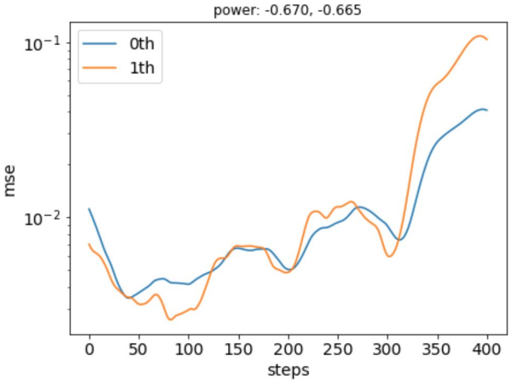


System 1: Incompressible Navier-Stokes in 2D

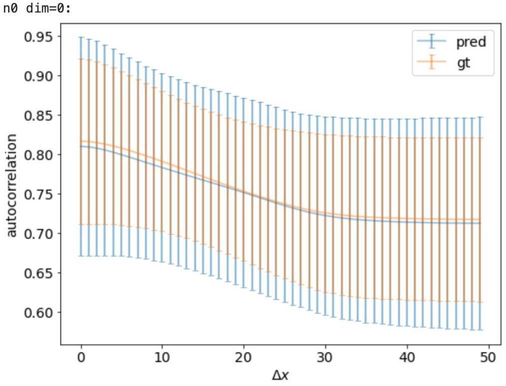
C. $L_{\text{consistency}}$ without normalization:

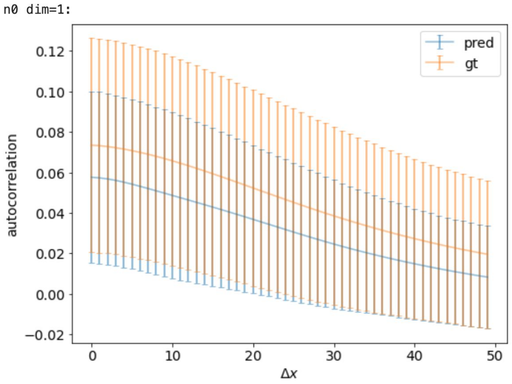
MSE vs. rollout steps:





Autocorrelation:



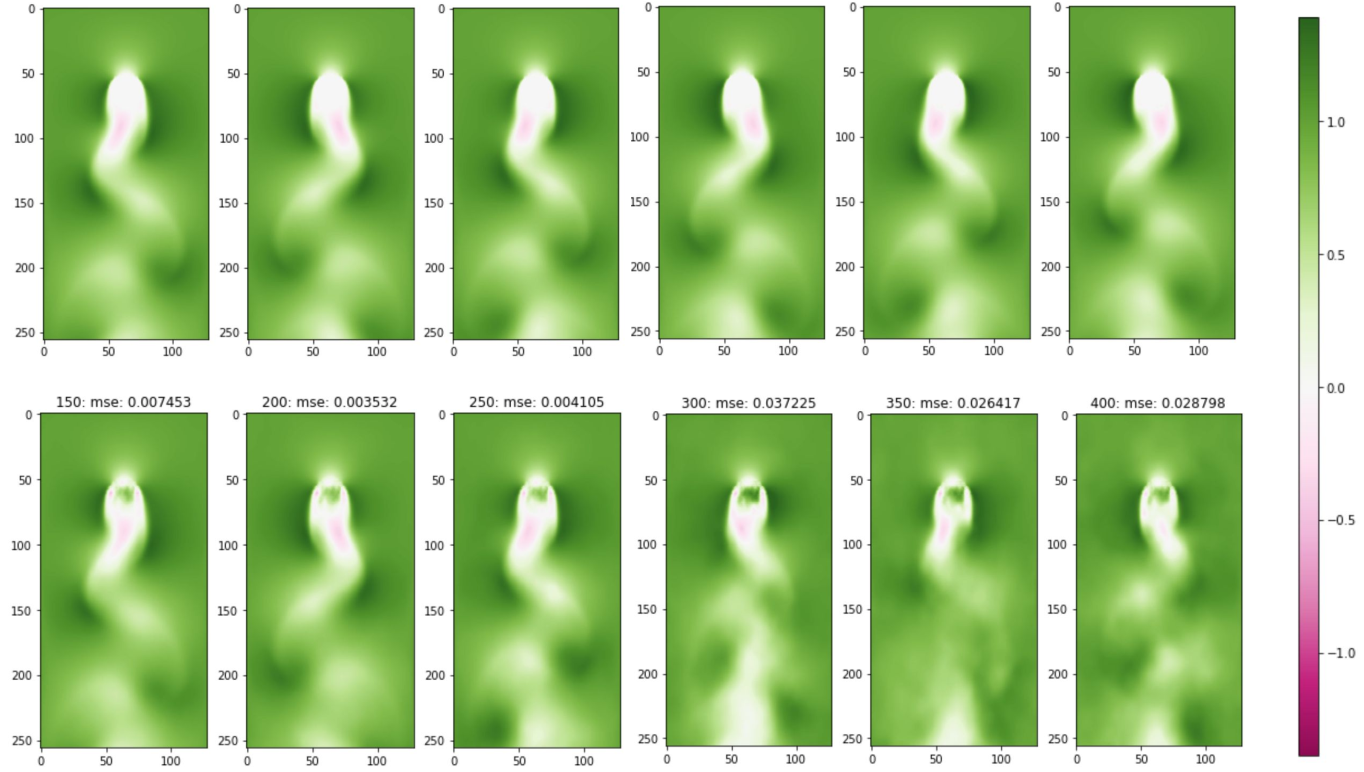


System 1: Incompressible Navier-Stokes in 2D

D. Without L_{recons} (with $L_{\text{1-step}}$) :

Ground-truth:

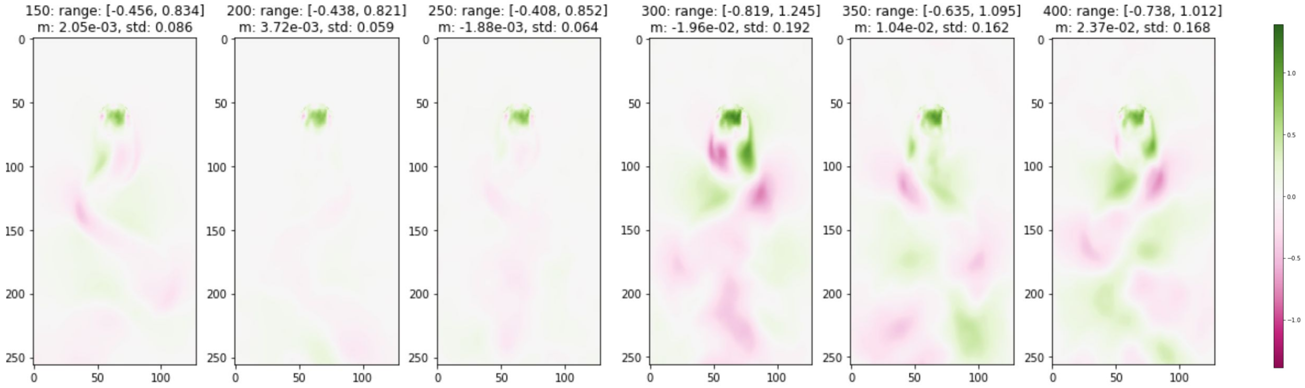
Model rollout:



System 1: Incompressible Navier-Stokes in 2D

D. Without L_{recons} (with $L_{\text{1-step}}$) :

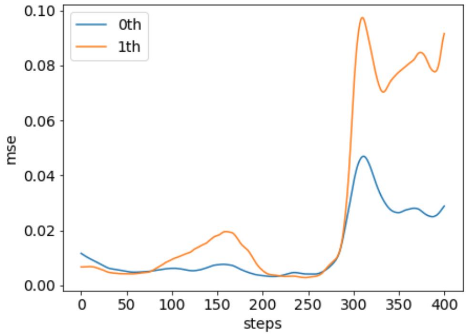
difference:

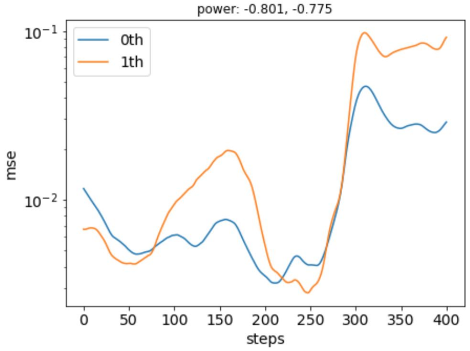


System 1: Incompressible Navier-Stokes in 2D

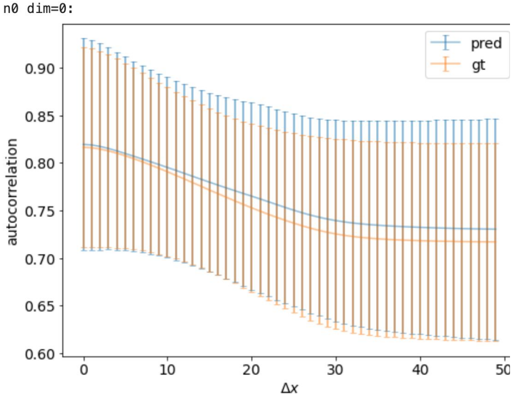
D. Without L_{recons} (with $L_{\text{1-step}}$) :

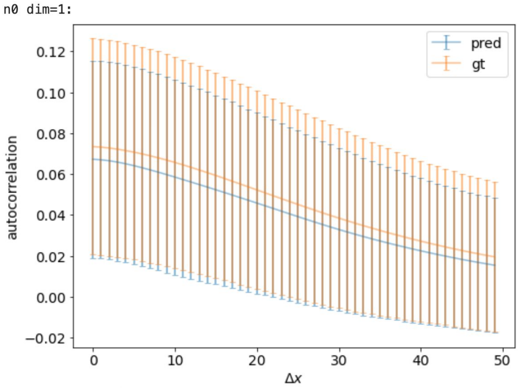
MSE vs. rollout steps:





Autocorrelation:



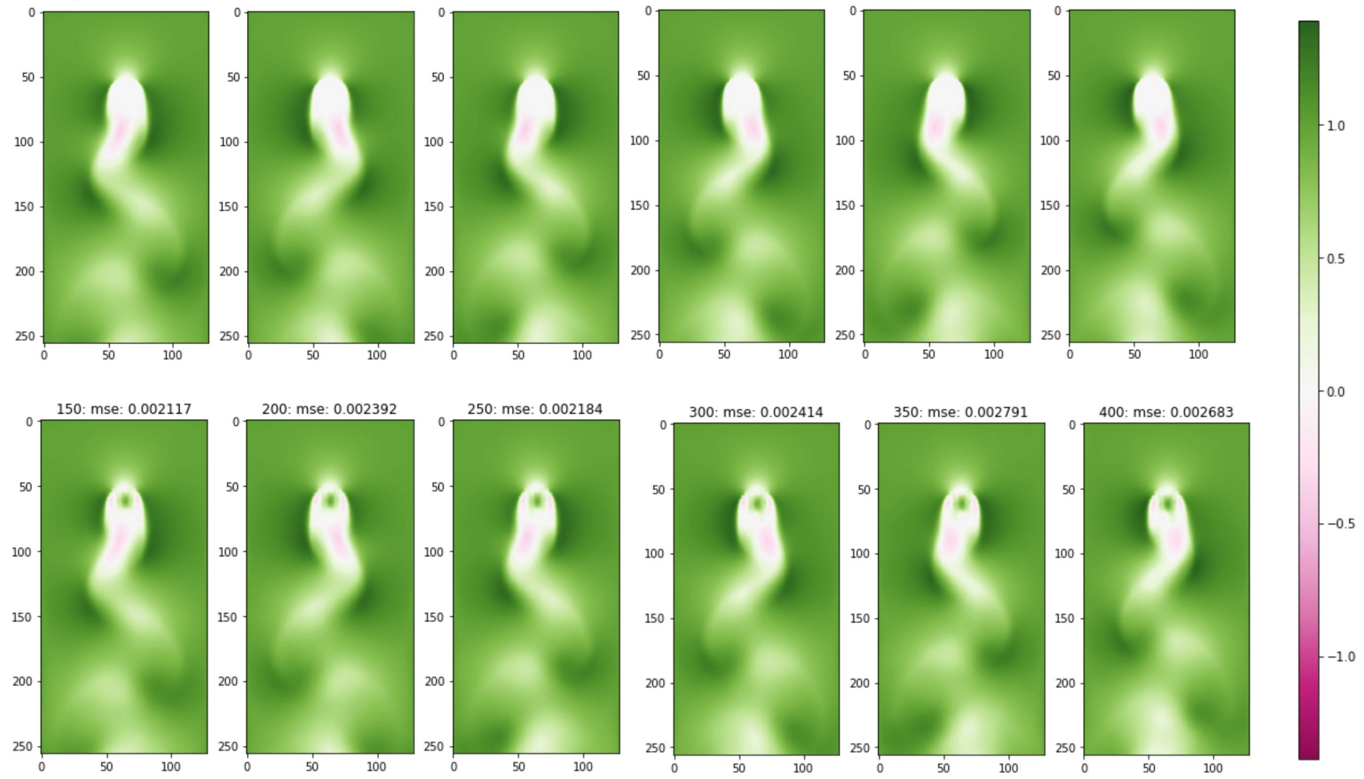


System 1: Incompressible Navier-Stokes in 2D

E. Without $L_{\text{1-step}}$ (with L_{recons}) :

Ground-truth:

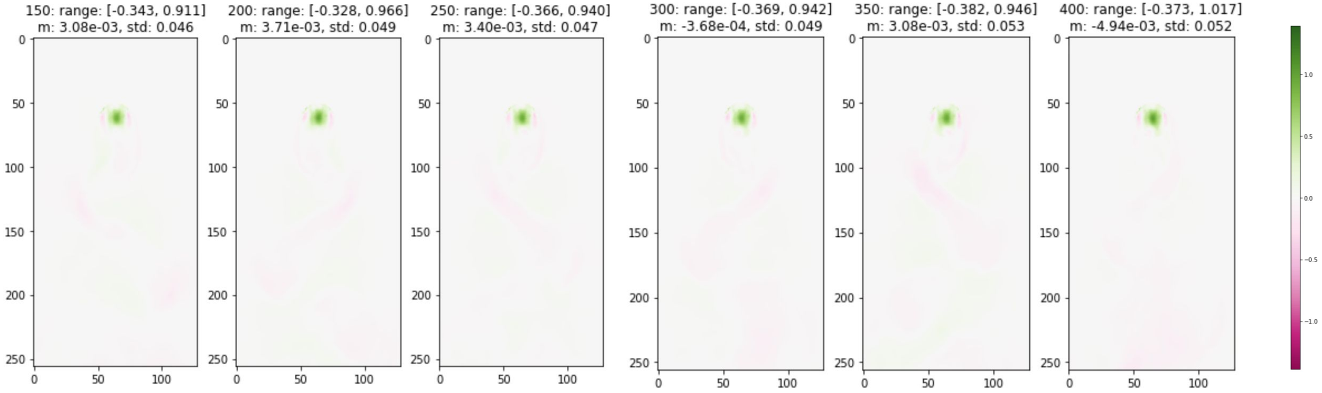
Model rollout:



System 1: Incompressible Navier-Stokes in 2D

E. Without $L_{\text{1-step}}$ (with L_{recons}) :

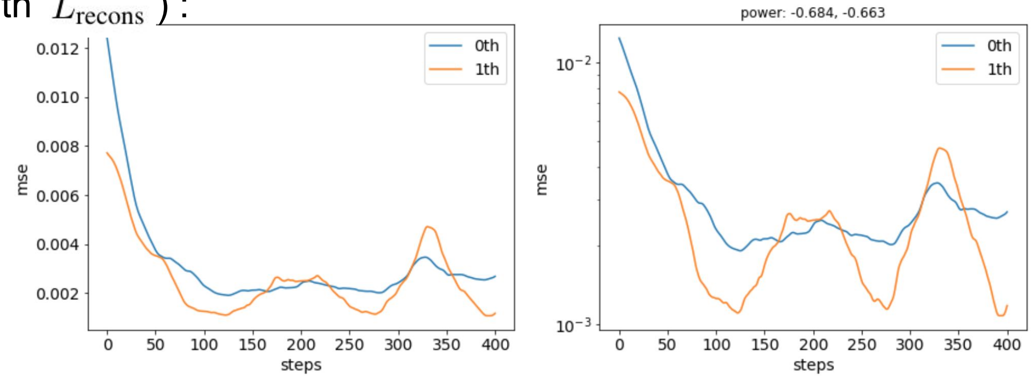
difference:



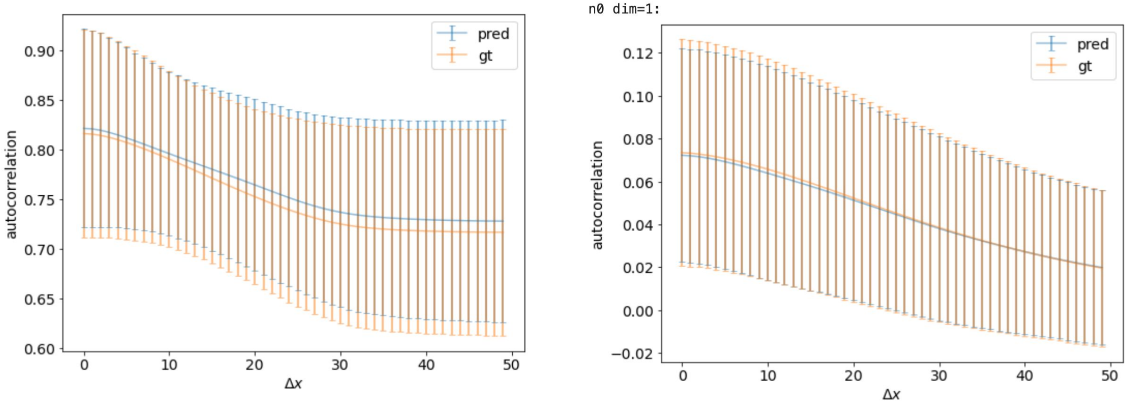
System 1: Incompressible Navier-Stokes in 2D

E. Without $L_{1\text{-step}}$ (with L_{recons}) :

MSE vs. rollout steps:



Autocorrelation:



System 1: Incompressible Navier-Stokes in 2D

Other knowledge learned:

1. Predicting change in the target (instead of target itself) can have smaller loss in the short term, but may **diverge faster** in the long term
2. Cosine annealing of learning rate [1] works better than reducing learning rate on plateau
3. 1-step Validation loss is not **necessarily correlated to** the long-term rollout performance. However, a very low 1-step validation loss is a good sign

[1] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." *arXiv preprint arXiv:1608.03983* (2016).

How to encourage obeying physical laws?

- **Architecture:** design architecture that automatically obeying such laws
  - e.g. Hamiltonian neural networks [1]
- **Objective:**
  - Use *known* physical constraint/laws as a regularization

Can we encourage obeying physical laws in latent space, without knowing specific form of such laws?

[1] Greydanus, Sam, Misko Dzamba, and Jason Yosinski. "Hamiltonian neural networks." *arXiv preprint arXiv:1906.01563* (2019).

How to encourage obeying physical laws?

$C(x_t) = C(x_{t+i})$

C: unknown conserved quantity

$C\big(g(z_t)\big) = C\big(g(z_{t+i})\big)$

$$L_{\text{contrast-conserv}} = \max_D \{\mathbb{E}[\log D(q(x_t), q(x_{t+i}))] + \mathbb{E}[\log \left(1 - D(q(x_t), K^{(i)} \circ q(x_t)))\right)]\}$$

D: discriminator

Training:

positive pairs: $q(x_t), q(x_{t+i})$

negative pairs: $q(x_t), K^{(i)} \circ q(x_t)$ (not strictly obeying)

We use the Siamese architecture: $D(z_1, z_2) = \text{sigmoid}(W\|s(z_1) - s(z_2)\|_2^2 + b)$
similar to [1].

[1] Ha, Seungwoong, and Hawoong Jeong. "Discovering conservation laws from trajectories via machine learning." *arXiv preprint arXiv:2102.04008* (2021).

How to encourage obeying physical laws?

$$C(x_t) = C(x_{t+i})$$

C: unknown conserved quantity

$$C\big(g(z_t)\big) = C\big(g(z_{t+i})\big)$$

$$L_{\text{contrast-conserv}} = \max_D \{\mathbb{E}[\log D(q(x_t), q(x_{t+i}))] + \mathbb{E}[\log \left(1 - D(q(x_t), K^{(i)} \circ q(x_t)))\right)]\}$$

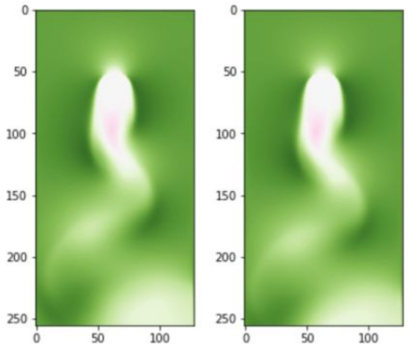
D: discriminator

Inference: the discriminator can help correct the trajectory if z_t evolves out of the conserved surface.

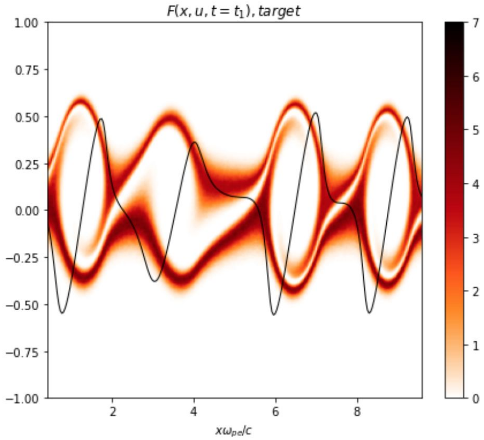
Goal:

For large-scale PDE systems, can we design **accurate** and **generalizable** ML models that capture the **essential dynamics** of the system with significant **speed ups**?

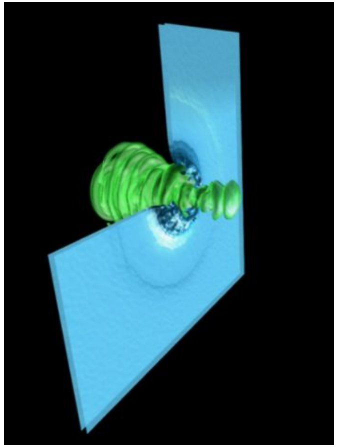
Navier-Stokes equation



Plasma 2-stream Vlasov equation



Plasma full Vlasov equation, simulated by Particle-in-Cell

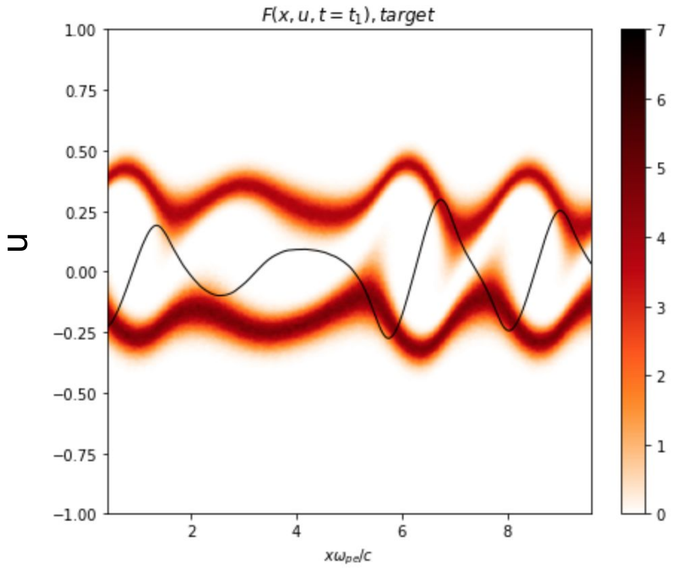


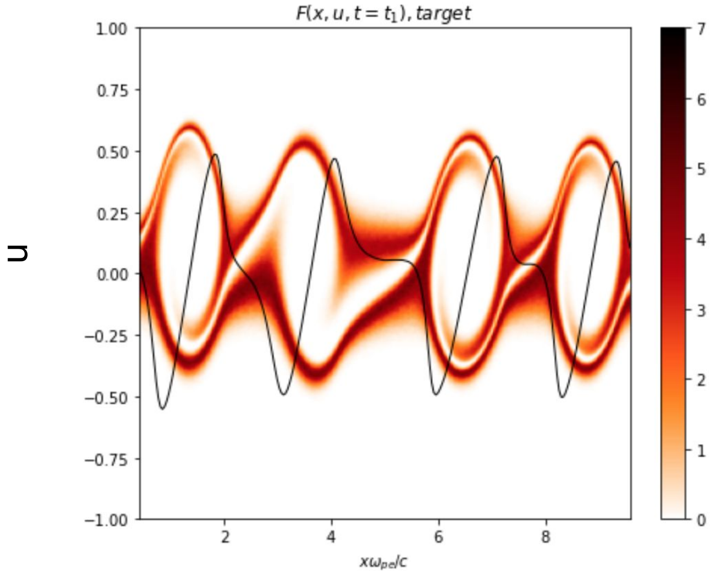
System 2: Plasma 2-stream Vlasov equation

$$\frac{\partial f_e(\boldsymbol{x},\boldsymbol{u},t)}{\partial t} = -\boldsymbol{v}(\boldsymbol{u})\cdot\nabla_x f_e(\boldsymbol{x},\boldsymbol{u},t) - \frac{q_e}{m_e}\boldsymbol{E}(\boldsymbol{x})\cdot\nabla_v f_e(\boldsymbol{x},\boldsymbol{u},t)$$

$$\frac{\partial \boldsymbol{E}}{\partial t} = -4\pi q_e \int d\boldsymbol{u}\, \boldsymbol{v} f_e(\boldsymbol{x},\boldsymbol{u},t)$$

curve: E field
density: electron density



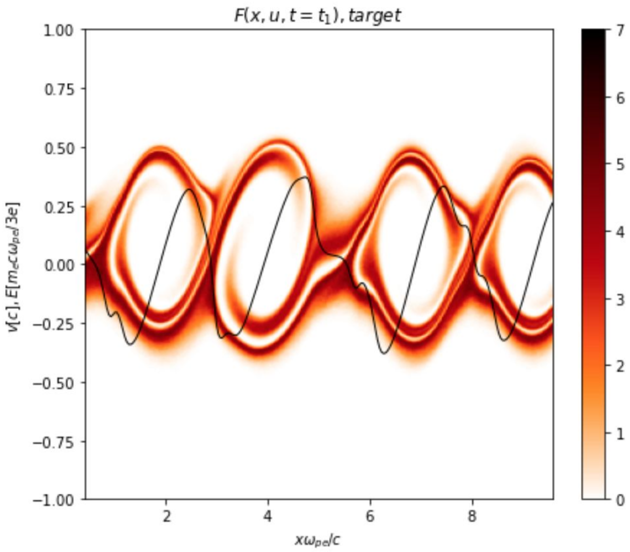


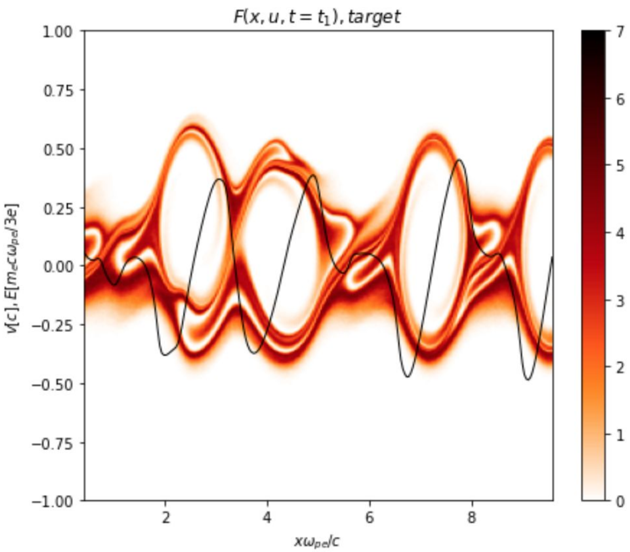
System 2: Plasma 2-stream Vlasov equation

$$\frac{\partial f_e(\boldsymbol{x},\boldsymbol{u},t)}{\partial t} = -\boldsymbol{v}(\boldsymbol{u}) \cdot \nabla_x f_e(\boldsymbol{x},\boldsymbol{u},t) - \frac{q_e}{m_e}\boldsymbol{E}(\boldsymbol{x}) \cdot \nabla_v f_e(\boldsymbol{x},\boldsymbol{u},t)$$

$$\frac{\partial \boldsymbol{E}}{\partial t} = -4\pi q_e \int d\boldsymbol{u}\, \boldsymbol{v} f_e(\boldsymbol{x},\boldsymbol{u},t)$$

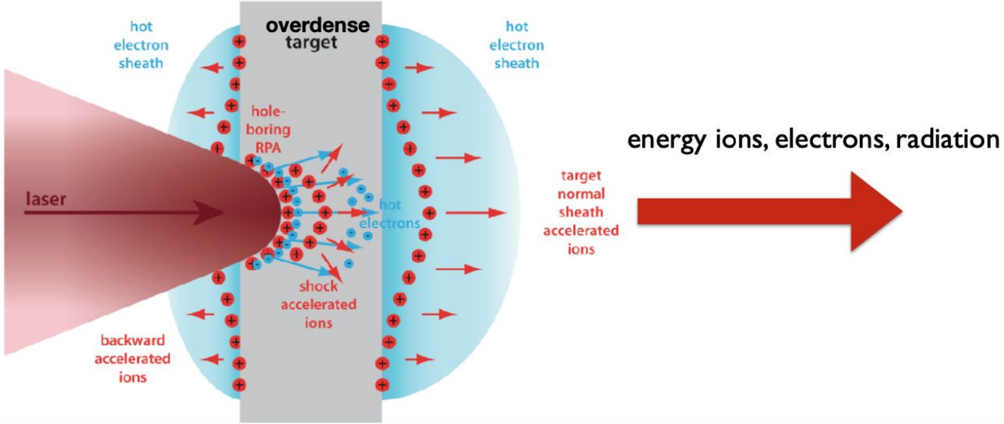
curve: E field
density: electron density



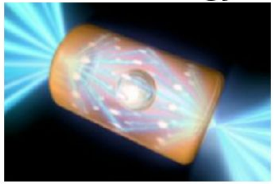


System 3: Laser-plasma acceleration

High-intensity laser-plasma interactions can produce electric fields of ~10 TeV/m

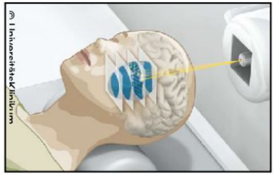


Fusion energy



Materials

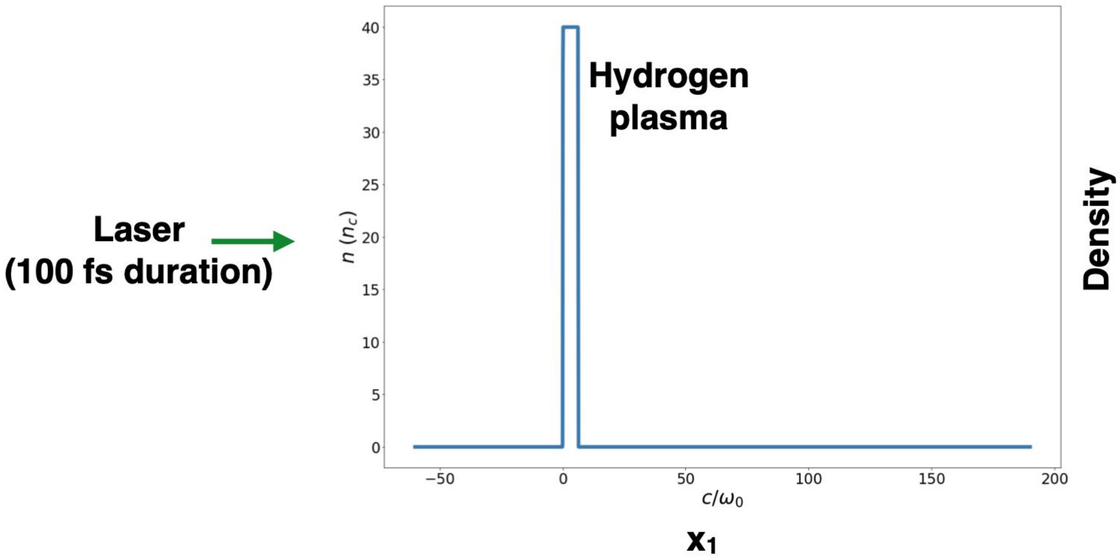
Medicine



Detailed understanding and control of highly nonlinear plasma processes is critical (e.g. laser absorption, electron transport, instabilities)

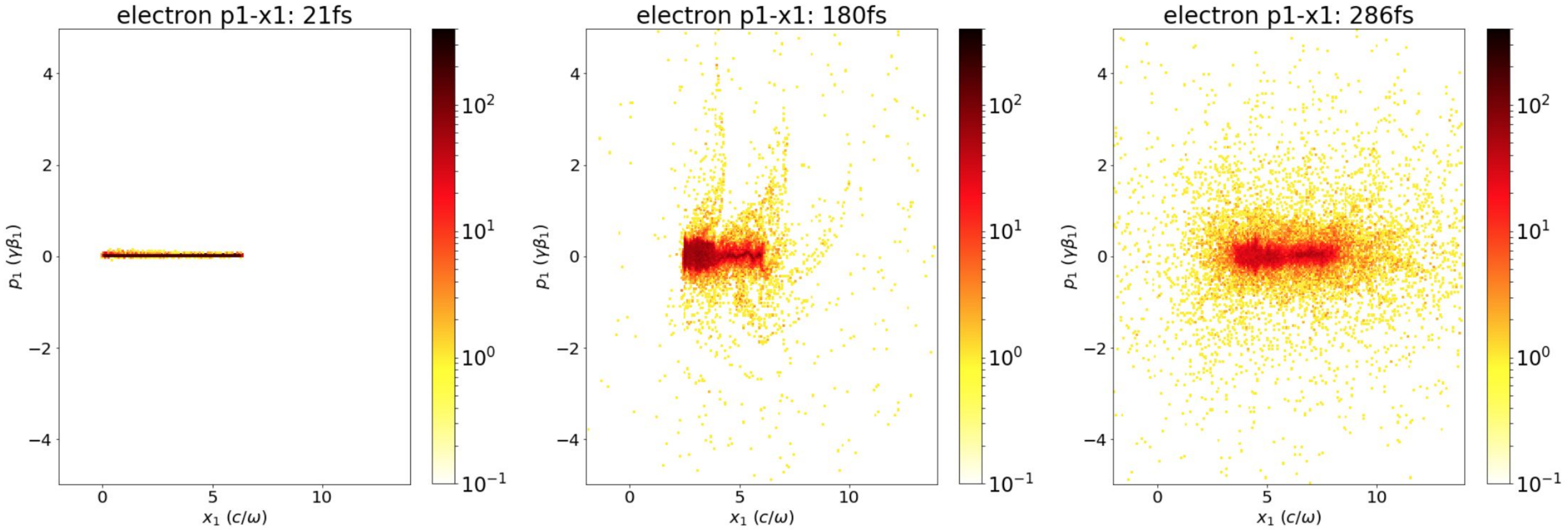
System 3: Laser-plasma acceleration

1D PIC simulation of an intense laser pulse interaction with a 1 μm thick hydrogen target



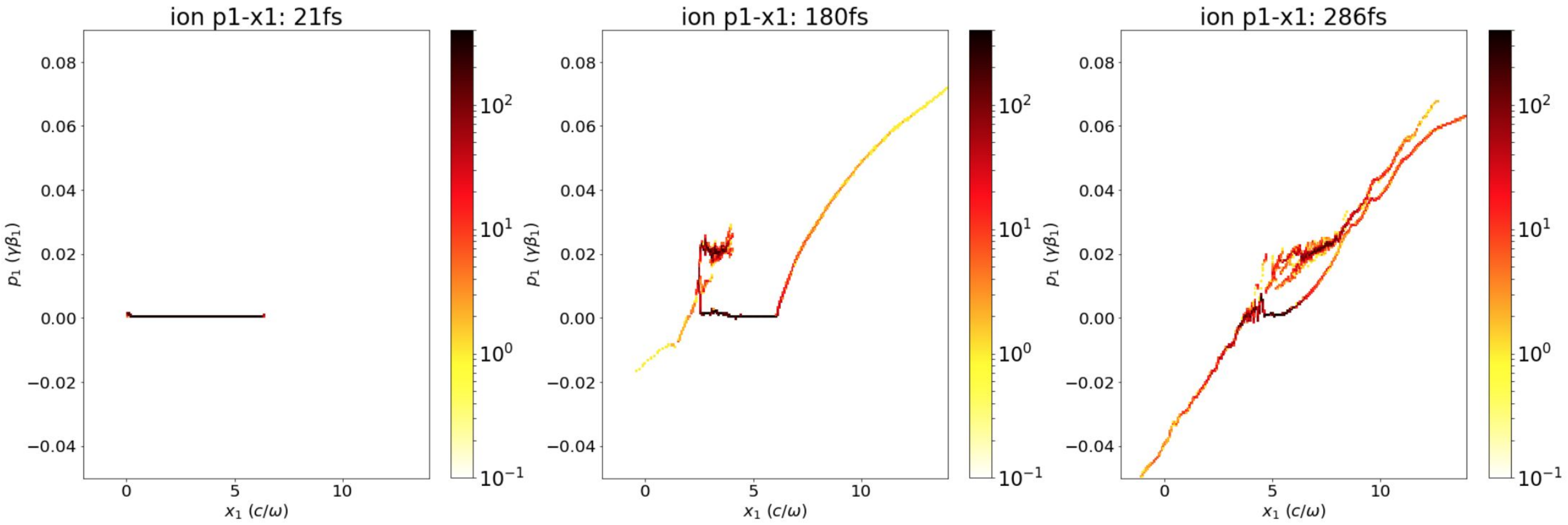
System 3: Laser-plasma acceleration

Electron phase space



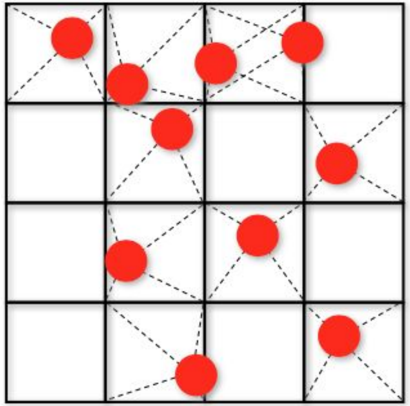
System 3: Laser-plasma acceleration

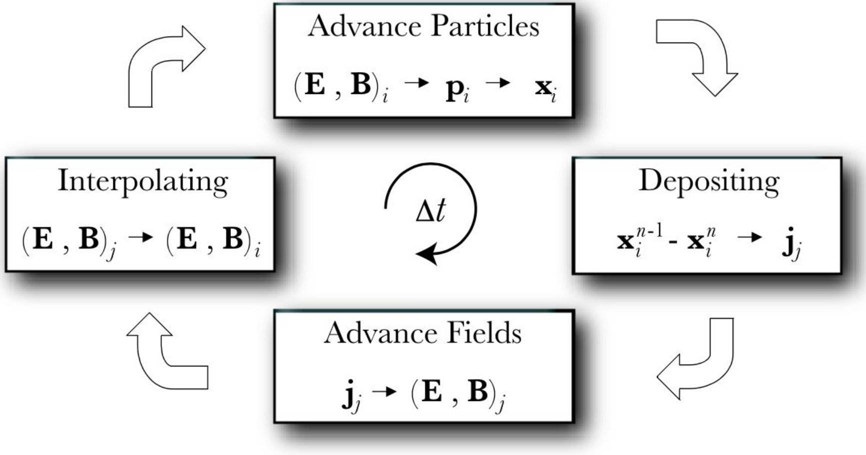
Ion phase space



System 3: Plasma full Vlasov equation: laser-plasma acceleration

Standard numerical method: Particle-In-Cell (PIC)





New challenges:

- Transients, not periodic
- Multi-scale, has fine-grained structure
- Kinetic; simulation is noisy

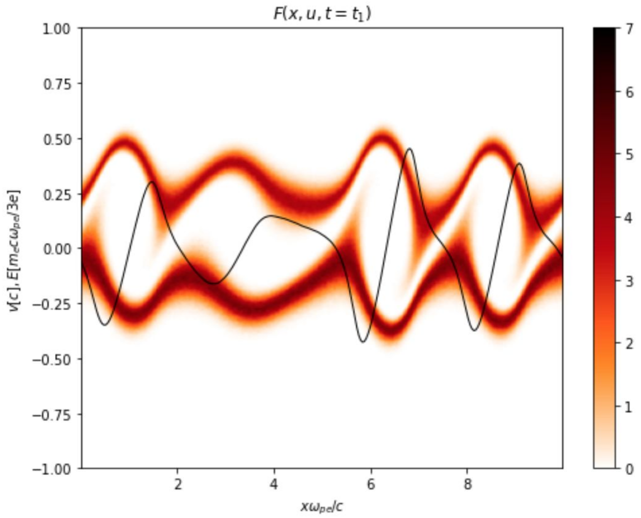
Ongoing

For Plasma 2-stream Vlasov equation: compress velocity distribution

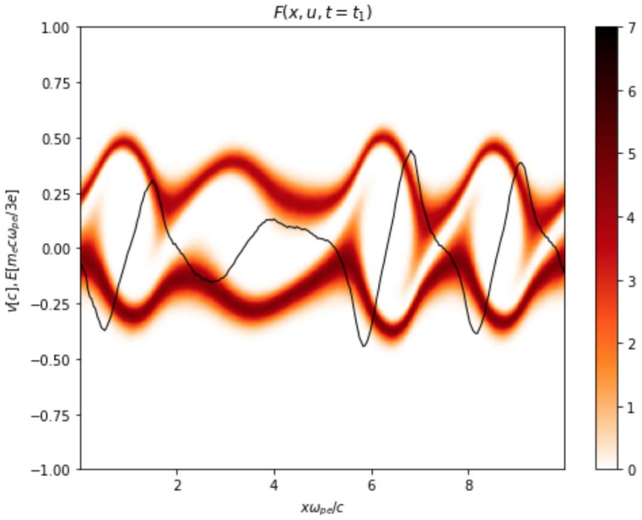
Encoder: CNN + MLP on u direction
Evolution: CNN
Decoder: MLP + CNN on u direction

u dimension: 256
latent dimension: 32

Ground-truth:



Reconstruction:



Ongoing

For Plasma 2-stream Vlasov equation: compress velocity distribution

Learning a compressed velocity representation has close connection with moment closure problem.

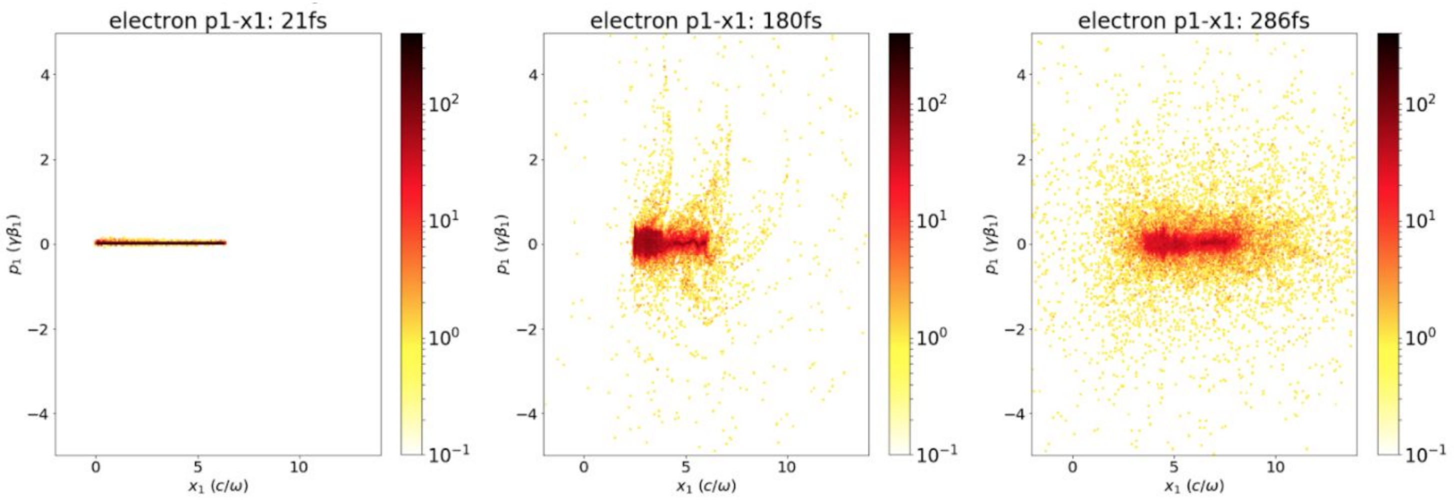
In [1], the authors addresses the moment closure problem using neural networks, by requiring the equation of the last moment to be closed.

[1] Han, Jiequn, et al. "Uniformly accurate machine learning-based hydrodynamic models for kinetic equations." *Proceedings of the National Academy of Sciences* 116.44 (2019): 21983-21991.

Ongoing

For laser-plasma acceleration:

Encoder: GNN + pooling
Evolution: GNN
Decoder: GNN + unpooling



Summary: Latent evolution of PDEs (LE-PDE)

Compress the input into some *suitable* **latent space**, and evolve the dynamics **fully in the latent space**.

Objective: $L = L_{\text{1-step}} + \alpha L_{\text{recons}} + \beta L_{\text{consistency}} + \gamma R_{sn} + \eta R_{\text{contrast-conserv}}$

Architecture:

- Latent evolution
- Discriminator for conservation laws
- Specific encoder/decoder for different problems

Our pipeline allows switching objective and architecture (e.g. CNN, GNN) independently, for a wide range of PDE and Particle-in-Cell systems including the NV equation, Vlasov equation and laser-plasma acceleration

Thank you!

Questions?