AI for complex physical simulation and inverse design

吴泰霖
西湖大学工学院AI方向 特聘研究员、博导
人工智能与科学仿真发现实验室PI
07/12/2024 @ 实验粒子物理计算研讨会

Homepage: tailin.org

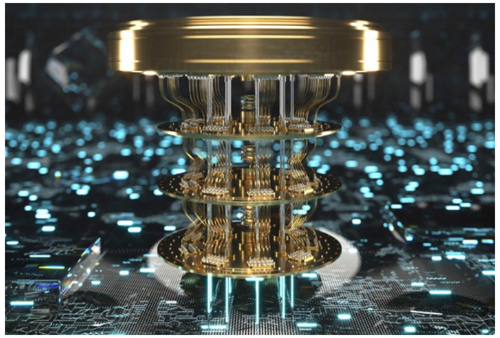
My research: AI for Science

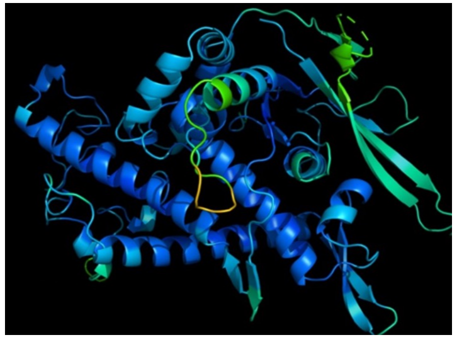
Forward:

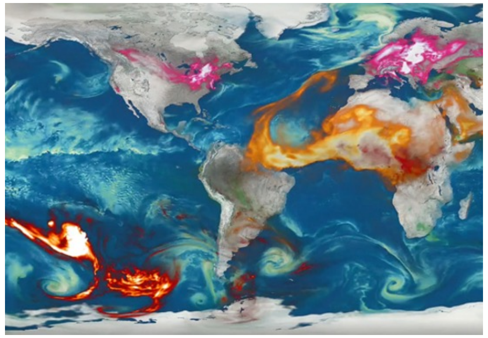
Simulate the system, predicting its evolution

Inverse:

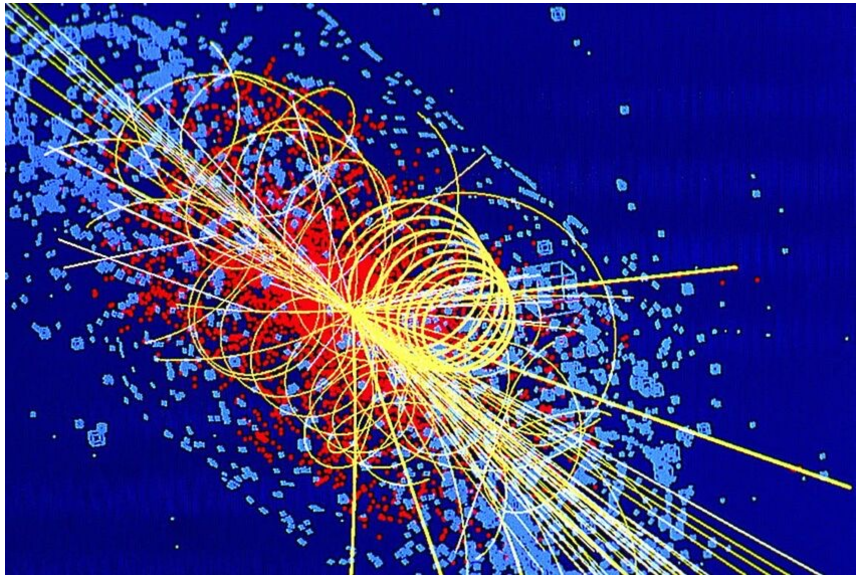
Inverse problem: Infer its state/parameters given observation
Inverse design: Optimize the system's parameters to optimize design objectives



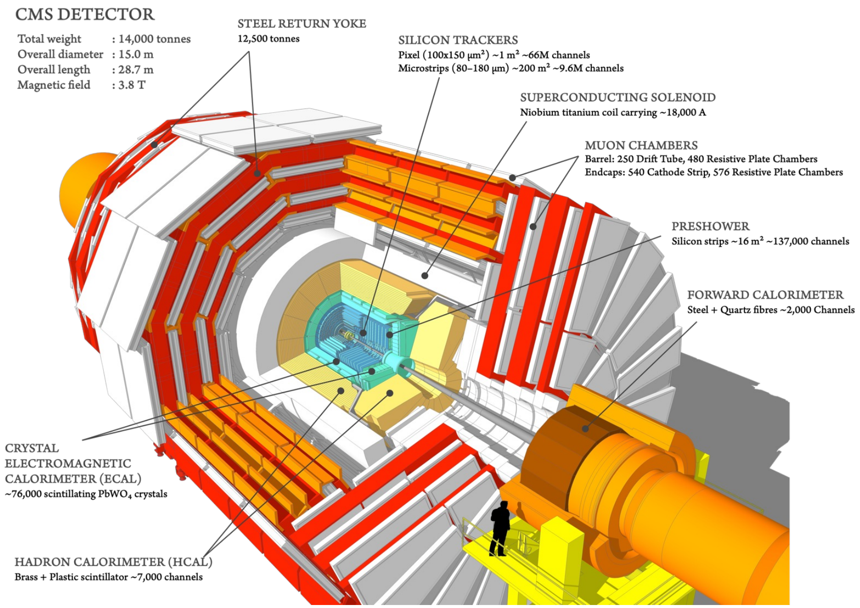




Forward ⇄ Inverse



[Image from University of Vienna]

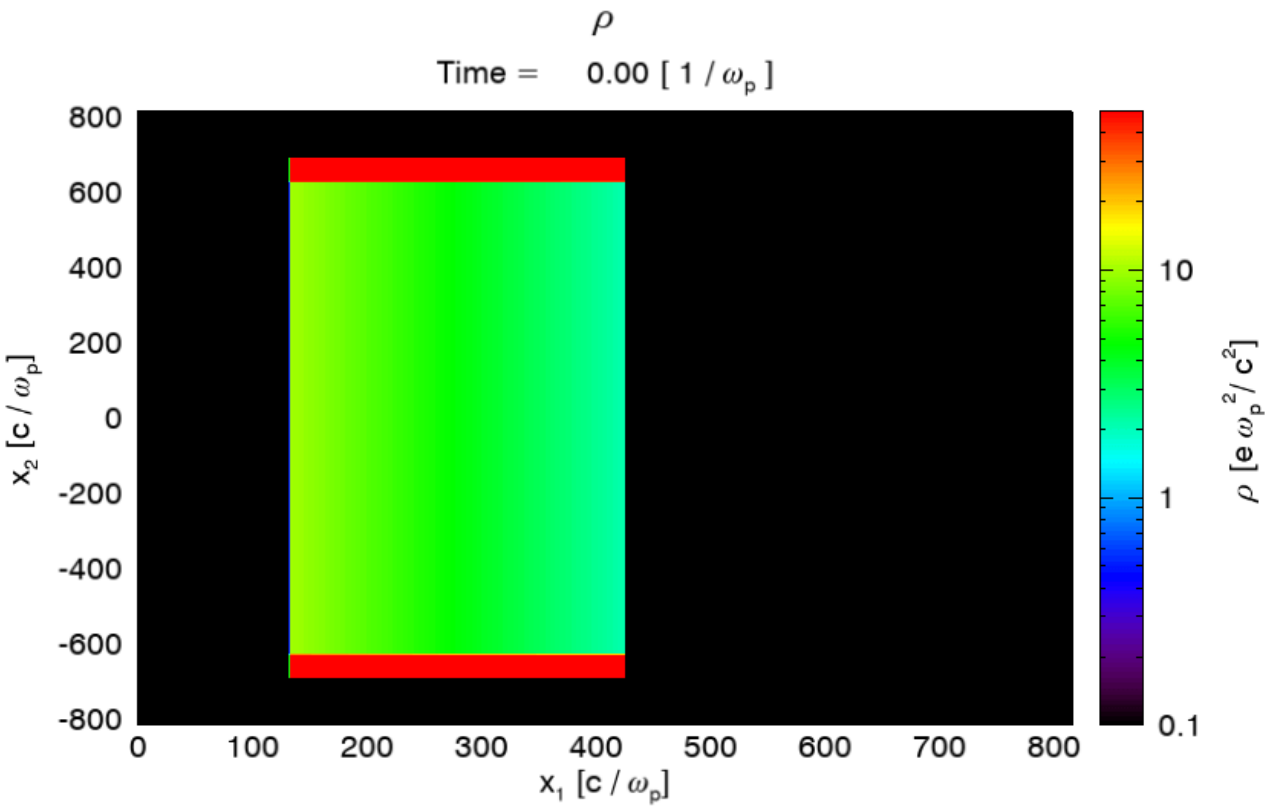


- Simulation
- Reconstruction (inverse problem)

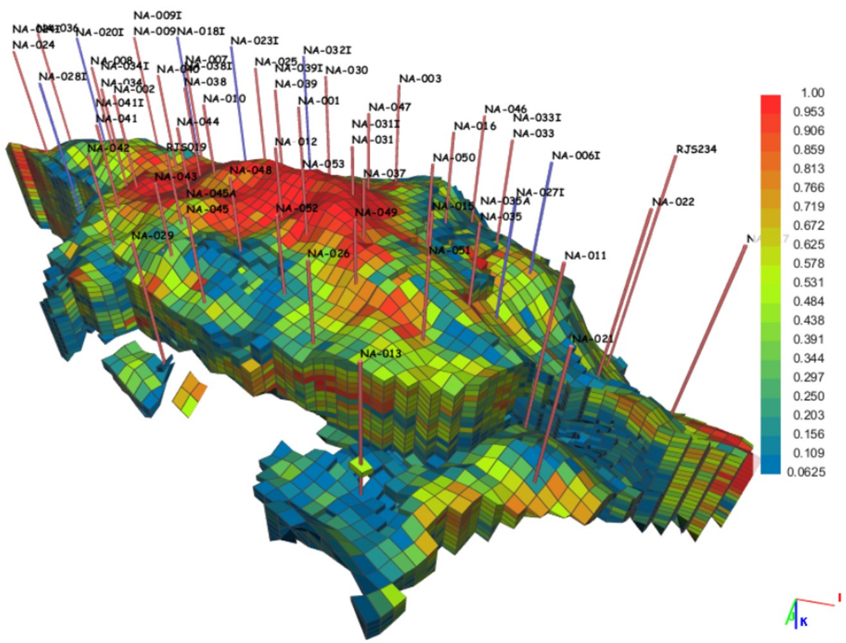
- Detector design (inverse design)

AI for scientific simulation

Develop machine learning (ML) methods for scientific simulation, improving its **speed** and/or **accuracy**



Laser-plasma interaction @ my collaboration with SLAC（斯坦佛国家加速器实验室）



GNN用于地下流体模拟，>10 million node graph，在工业界部署

AI for scientific simulation

Scientific simulation: simulate the **dynamics** or **steady state** of the system, given initial state, boundary condition and parameters of the system

AI for scientific simulation: develop machine learning (ML) methods for scientific simulation, improving its **speed** and/or **accuracy**

AI for scientific simulation

Tasks 任务		Architecture 架构		Learning paradigm 学习范式		Challenge 挑战
• Learning dynamics • Learning steady state	×	• GNN • Neural operators • Transformer • U-Net	×	• Regression • Diffusion model • Reinforcement learning • Physics-informed	×	• Multi-resolution • Function mappings

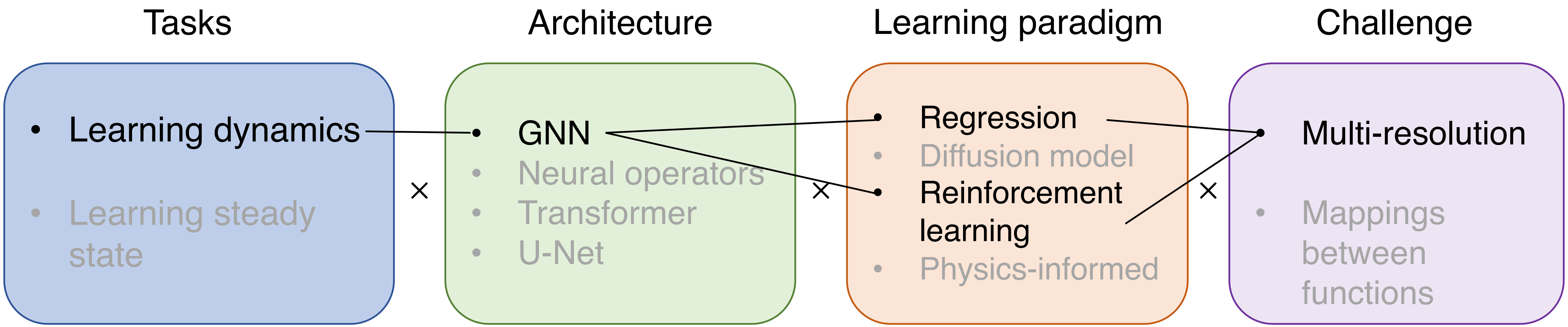
These parts are orthogonal, i.e., each task can choose suitable architecture and learning paradigm, and they can face multi-resolution and/or complex boundary conditions

AI for scientific simulation: part I

两类任务：学习系统动力学和稳态

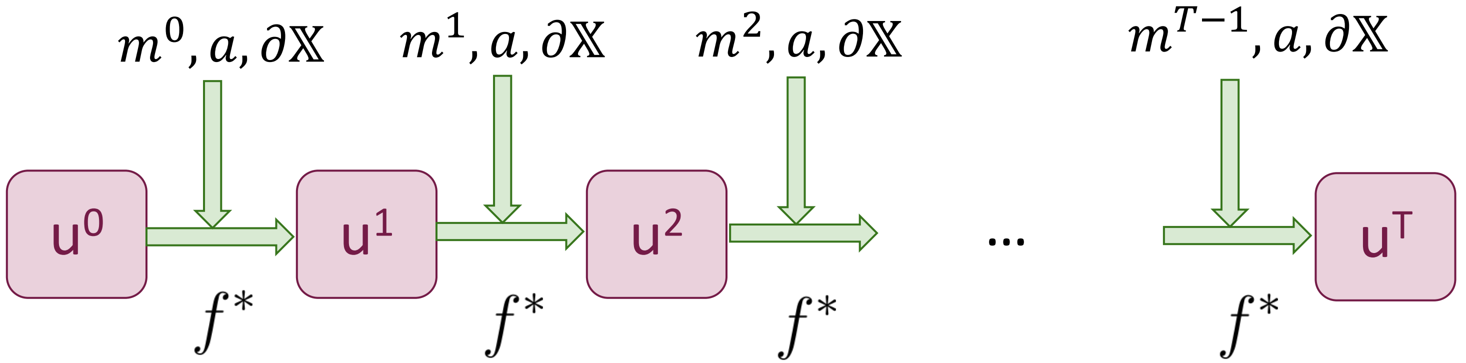
两个重要架构：图神经网络和神经算子

两个挑战和解决方式：多分辨率和函数映射



Task setup for evolving dynamics

Goal: 根据状态 u^t 预测未来状态 $u^{t+1}, u^{t+2}, \ldots$:



u^t: original **state**（状态）of the system. Can be an infinite-dimensional function $u(t, x)$ as solution to a PDE, or a graph (e.g., mesh, particle-based systems, molecules)

f^*: Evolution **（系统演化）**. By a classical solver（求解器） or in the real world（真实观测）

m^t: **external control**（外界控制）

a: **static parameters**（静态参数） of the system that does not change with time (e.g. parameters of PDE, spatially varying diffusion coefficient)

$\partial\mathbb{X}$: **boundary condition**（边界条件） of the system

PDE: partial differential equation

Spectrum of methods for simulating dynamics

First principle ⟷ Data driven

Classical solver

Accurate;
Interpretable;
Error guarantee

Slow;
Assumption may be incorrect

Pure deep learning

Fast ($10 – 10^4$x);
Can directly learn from data

Challenge in long-term acc.

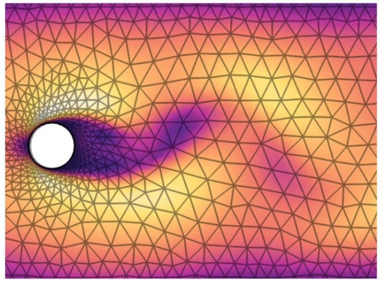
Classical solvers and limitations

Classical solvers:

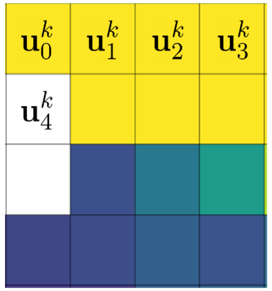
Based on Partial Differential Equations (PDEs) or ODEs

$$\frac{\partial \mathbf{u}}{\partial t} = F(x, \mathbf{u}, \frac{\partial \mathbf{u}}{\partial \mathbf{x}}, \frac{\partial^2 \mathbf{u}}{\partial \mathbf{x}^2}, \ldots)$$

$\mathbf{u}$: state
$\mathbf{x}$: spatial coordinate
t : time



mesh



grid

Discretize the PDE, then use finite difference, finite element, finite volume, *etc*. to evolve the system.

$$\mathbf{u}(t, \mathbf{x}) \longrightarrow \mathbf{u}_i^k$$

discrete time index k
discrete cell id i

Pros and challenges:

- **Pros:** (1) Based on first principles and interpretable, (2) accurate, (3) have error guarantee.
- **Challenges: Slow and computational expensive**, due to
  (1) Small time interval to ensure numerical stability, or use implicit method.
  (2) For multi-resolution systems, typically need to resolve to the lowest resolution

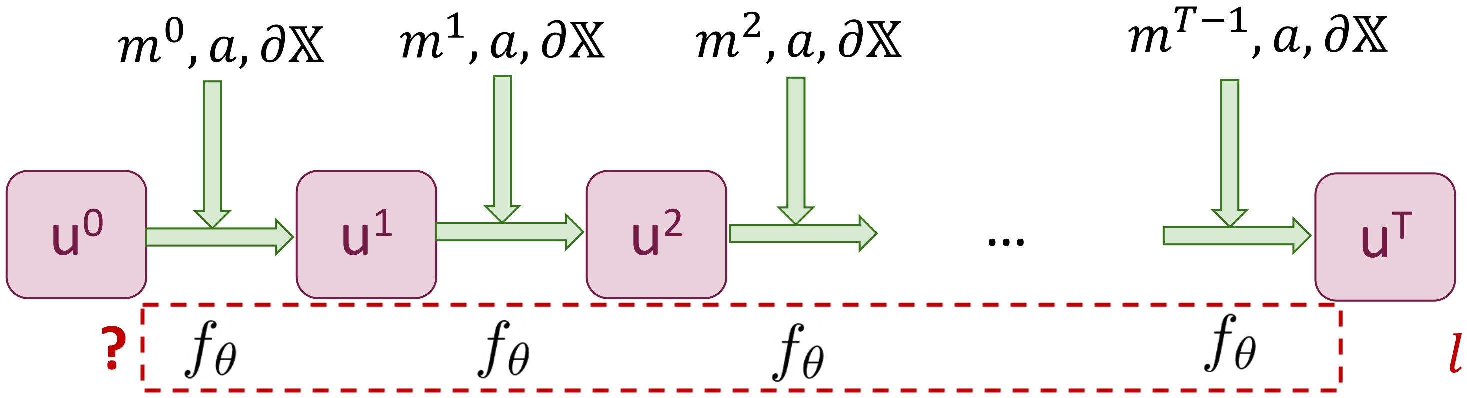
Deep learning-based surrogate models

Recently, deep learning based surrogate modeling has emerged as attractive alternative to replace or complement classical solvers. They:

- Offer **speedup (>10-1000 fold)** via:
  - Larger spatial resolution
  - Larger time intervals
  - Use explicit forward
  - Better representations

Task setup for learning dynamics

Goal: learn the mapping f_θ from u^t to u^{t+1}:



$loss = \mathbb{E}[MSE(f_\theta(u^t), u^{t+1})]$

u^t: original **state**（状态）of the system. Can be an infinite-dimensional function $u(t, x)$ as solution to a PDE, or a graph (e.g., mesh, particle-based systems, molecules)

f_θ: **neural surrogate models**（神经网络代理模型）

m^t: **external control**（外界控制）

a: **static parameters**（静态参数） of the system that does not change with time (e.g. parameters of PDE, spatially varying diffusion coefficient)

$\partial\mathbb{X}$: **boundary condition**（边界条件） of the system

PDE: partial differential equation

Spectrum of methods for simulating dynamics

First principle

Data driven

Classical solver

Accurate;
Interpretable;
Error guarantee

Slow;
Assumption may be incorrect

Solver-in-the-loop

Combine the advantages of both method

Deep learning incorporating inductive biases (symmetry, structures, etc.)

more accurate;
faster

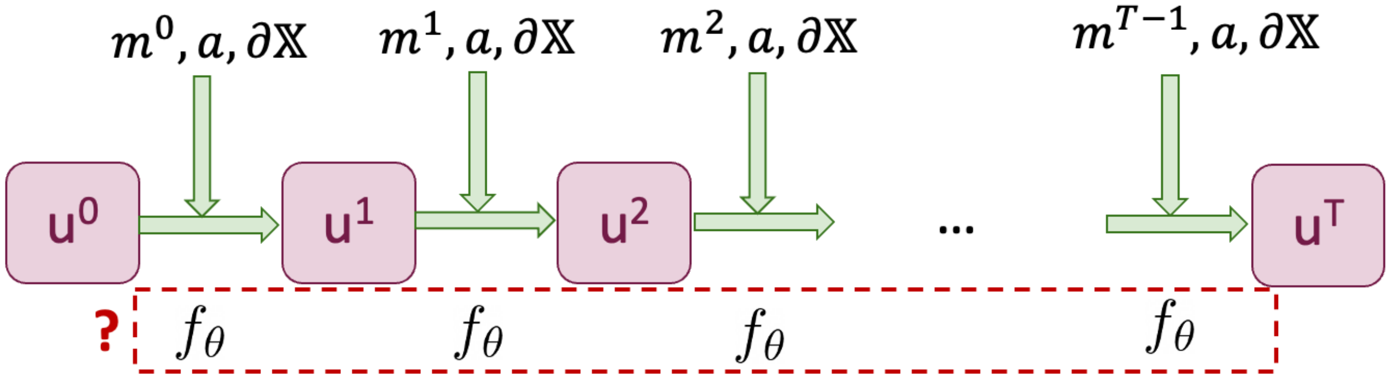
Pure deep learning

Fast ($10 – 10^4$x);
Can directly learn from data

Challenge in long-term acc.

+ better training objectives

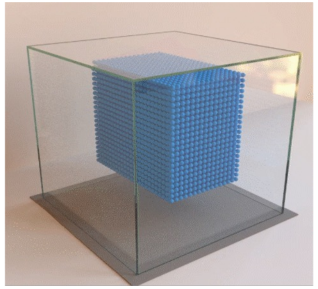
Case study: GNN（图神经网络）-based simulation



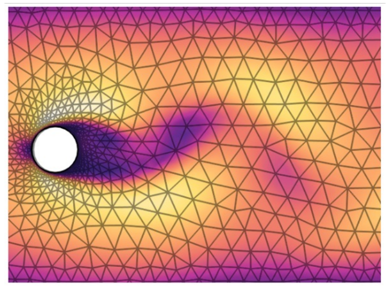
u^t: **state** of the system. Represented as a **graph** (e.g., mesh, particles, molecules)

f_θ: Graph Neural Network (GNN)

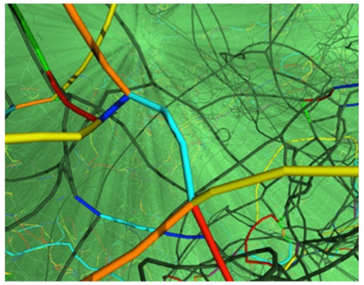
Such graph-structured data is universal across disciplines:



Fluid dynamics, computer graphics



Mesh-based simulation for PDEs

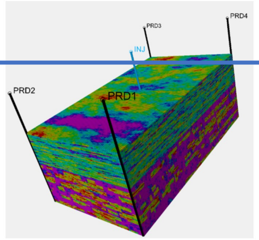


Dislocation in materials

Proteins and small molecules

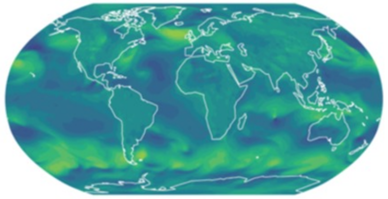
Case study: GNN-based simulation

HGNS



(Wu et al., KDD 2022)

GraphCast

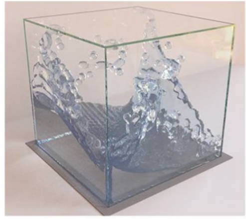


(Lam et al., Science 2023)

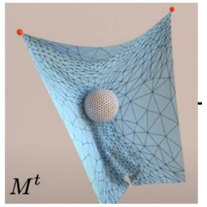
large-scale

GNS

(Sanchez-Gonzalez et al., ICML 2020)



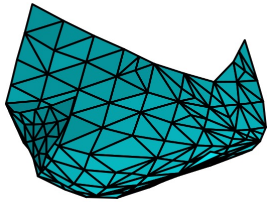
MeshGraphNets



(Pfaff et al., ICLR 2021, outstanding paper)

multi-resolution

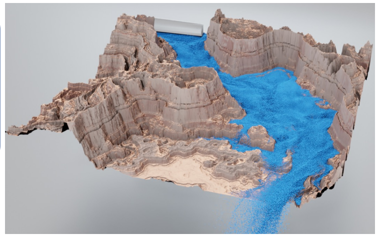
LAMP



(Wu et al., ICLR 2023, spotlight)

symmetry

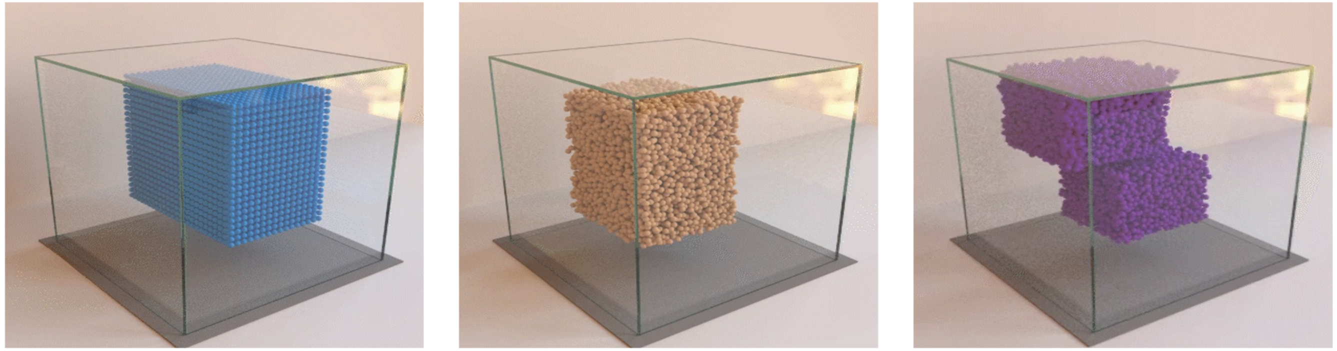
Momentum-conserved CConv



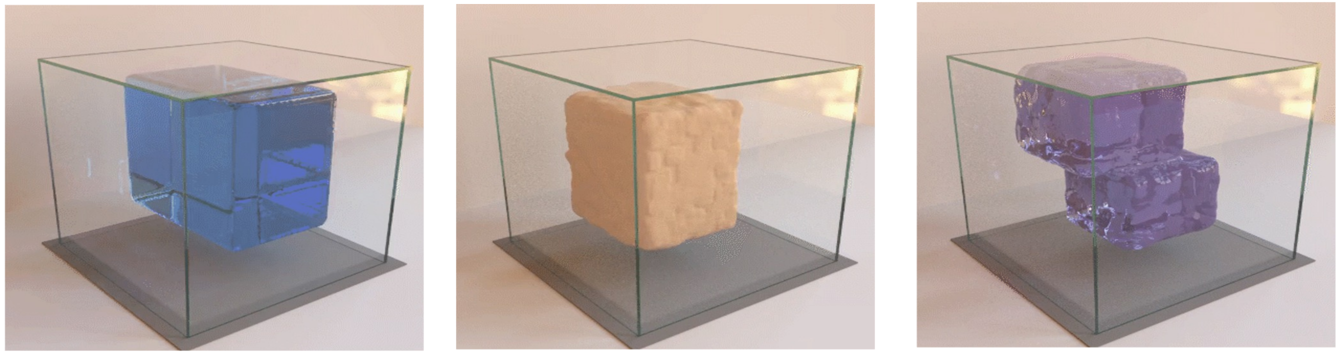
(Prantl et al., NeurIPS 2022)

Case study: Graph Network Simulator (GNS)

Graph Network Simulator (GNS) [1] introduced a GNN-based simulator that learns to simulate particle-based systems



GNN Model predicts particle positions and velocity



Predicted simulation after rendering

[1] Sanchez-Gonzalez, Alvaro, et al. "Learning to simulate complex physics with graph networks." *International conference on machine learning*. PMLR, 2020.

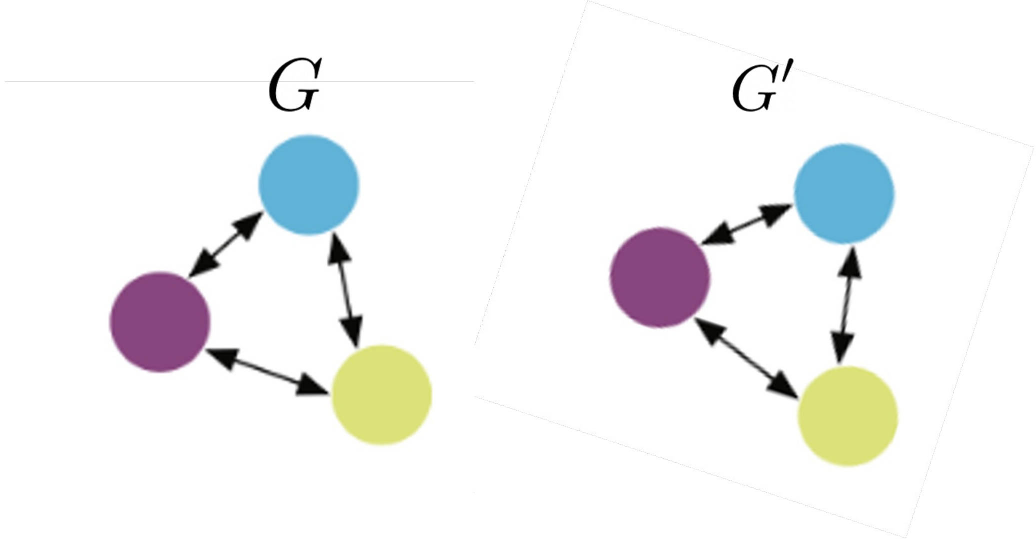
Preliminary: graph neural networks (GNNs)

A GNN f_θ takes **graph-structured data** G=(V, E) as input, and typically maps to another G'=(V', E'):

$$G' = f_\theta(G)$$

n-body system:

1. Compute **forces** on each **edge**
2. **Accumulate** forces on each **node**
3. Update the **node feature** (position, velocity) based on the **accumulated force** and **previous node feature**



node feature: position, velocity

Battaglia *et al.* (2018)

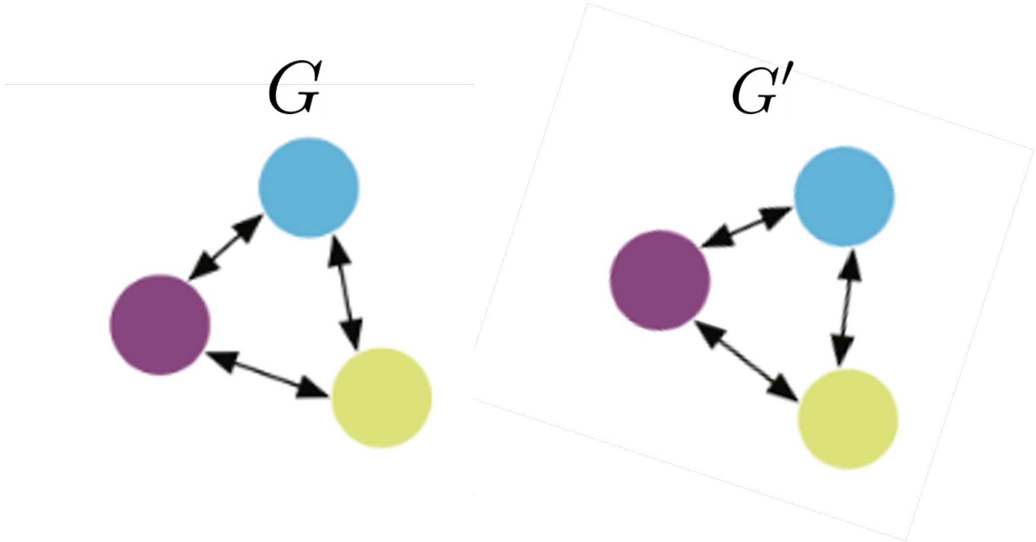
Preliminary: graph neural networks (GNNs)

A GNN f_θ takes **graph-structured data** G=(V, E) as input, and typically maps to another G'=(V', E'):

$$G' = f_\theta(G)$$

Generic GNN:

1. Compute learnable **messages** on each **edge**
2. **Accumulate messages** on each **node**
3. Perform a learnable update on the **node feature** using the **accumulated messages** and **previous node feature**

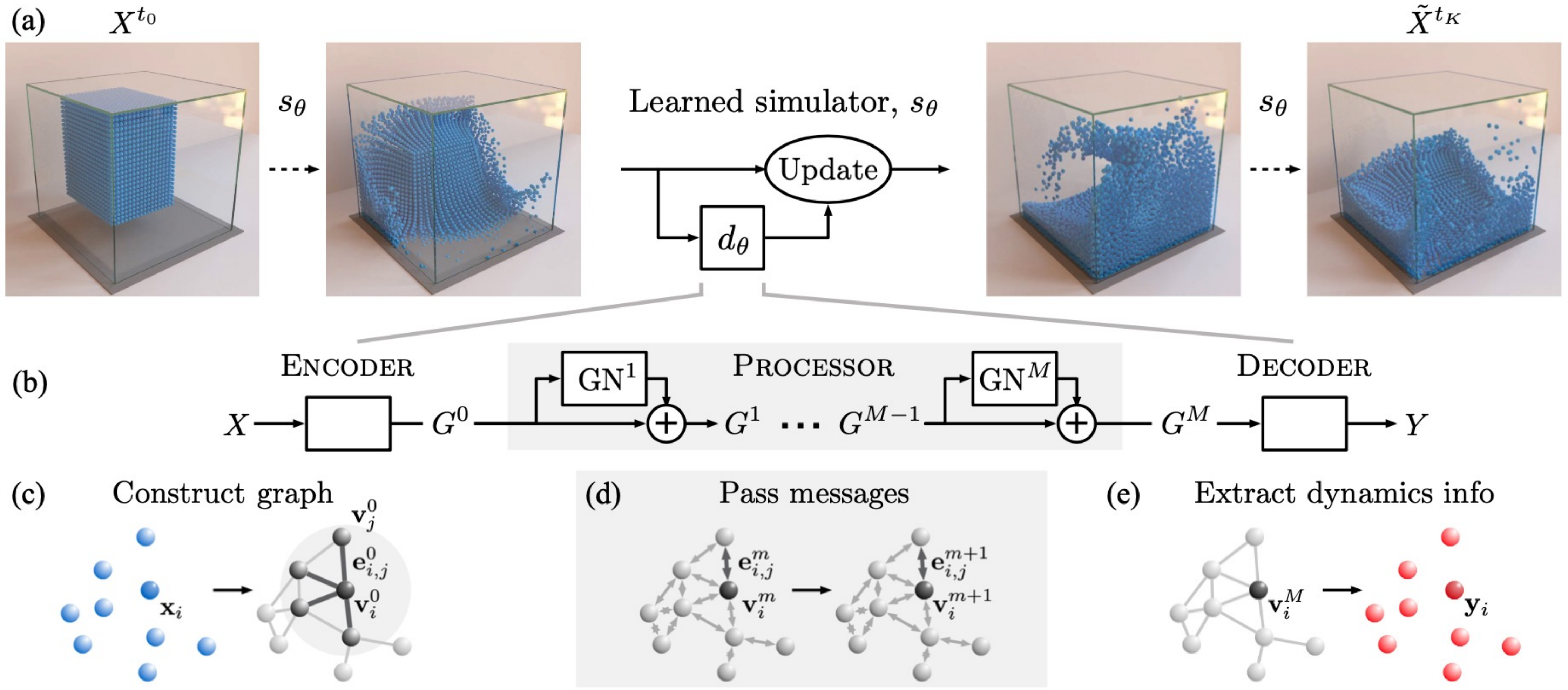


The GNN learns by minimizing the prediction loss w.r.t. the parameter θ

$$loss = \mathbb{E}[MSE(f_\theta(u^t), u^{t+1})]$$

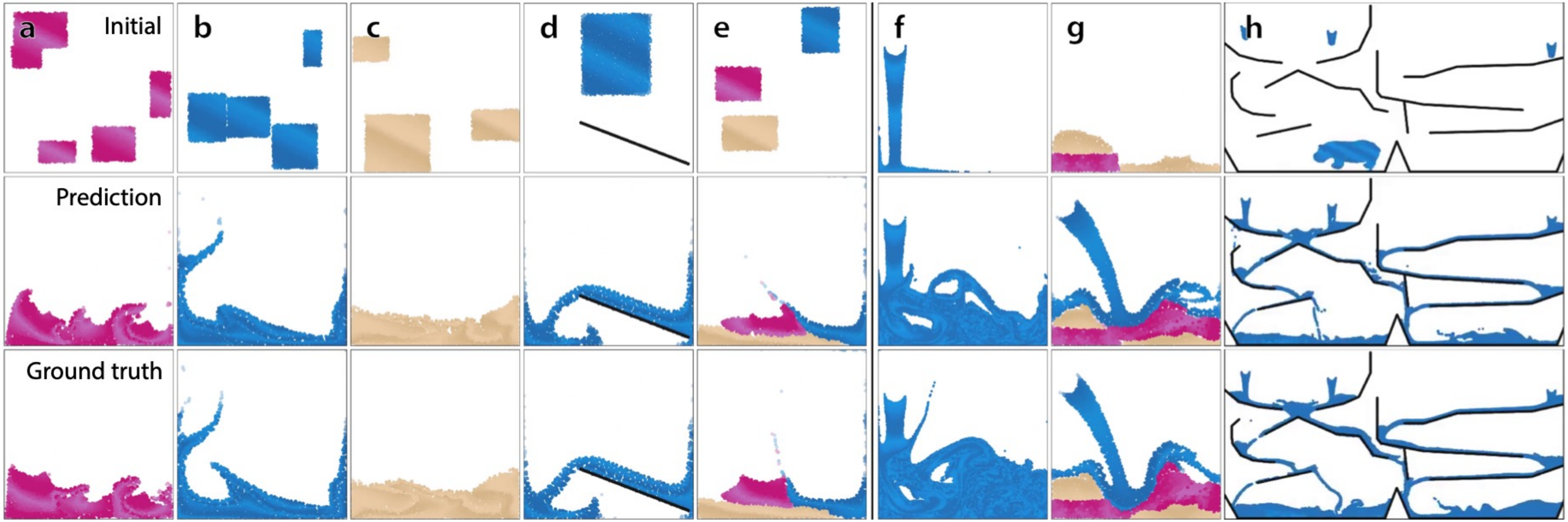
Battaglia *et al.* (2018)

Case study: Graph Network Simulator (GNS)

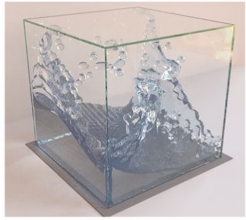


Case study: Graph Network Simulator (GNS)

Result:



Case study: GNN-based simulation



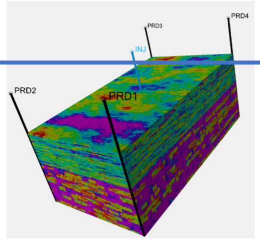
GNS

(Sanchez-Gonzalez et al., ICML 2020)

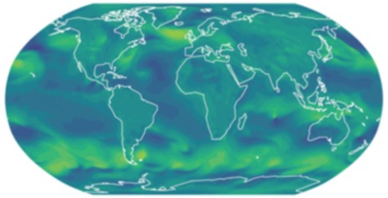
large-scale

HGNS

(Wu et al., KDD 2022)



GraphCast

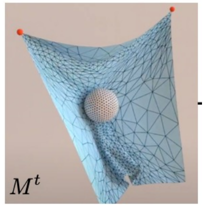


(Lam et al., Science 2023)

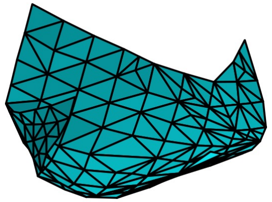
MeshGraphNets

multi-resolution

LAMP



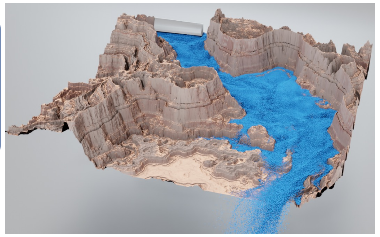
(Pfaff et al., ICLR 2021, outstanding paper)



(Wu et al., ICLR 2023, spotlight)

symmetry

Momentum-conserved CConv



(Prantl et al., NeurIPS 2022)

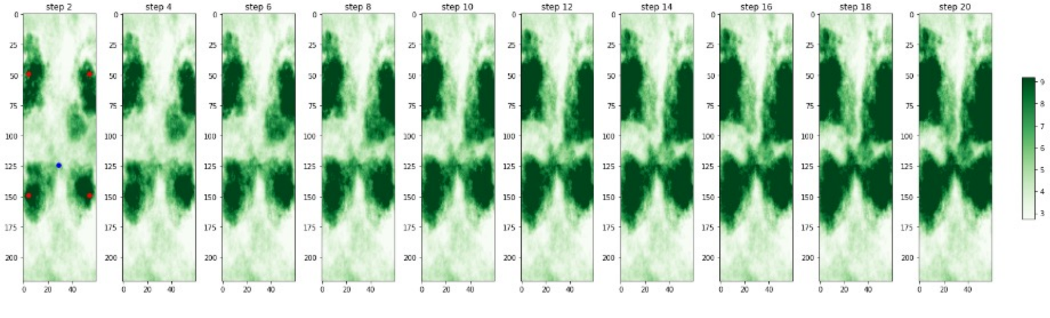
Limitation:
- Small scale (up to 20k particles)
- Long-term accuracy

Case study: Hybrid Graph Network Simulator (HGNS)

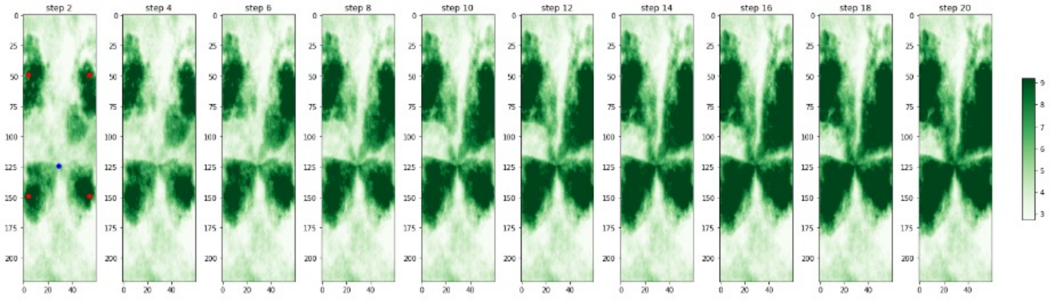
Task: Subsurface fluid simulation (critical in energy, carbon capture, etc.)

Main contribution: Introduced HGNS [1] for fluid simulation, which use
- multi-step prediction during training to improve long-term prediction accuracy
- Sector-based training and inference

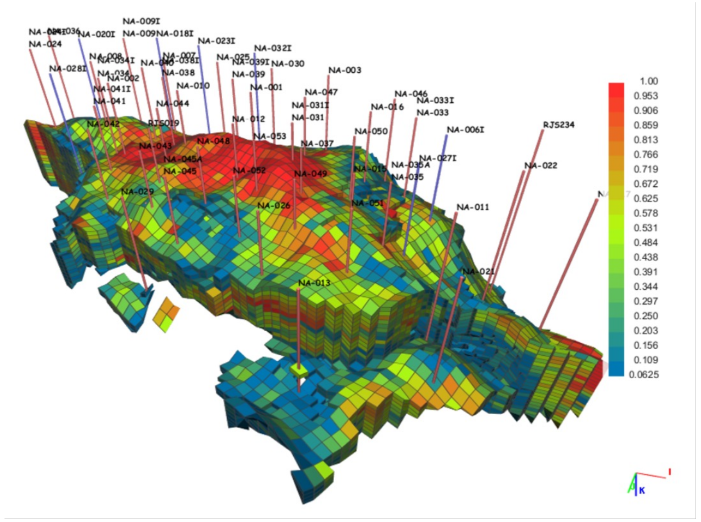
Results: Up to 18x faster than classical solver. Apply to **10 million** cells per step. Deployed in industry



(a) HGNS rollout of water volume (barrel) for 20 steps (months)



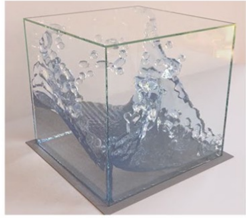
(b) Ground-truth of water volume (barrel) for 20 steps



Subsurface (consisting of cells, wells, fractures, etc.)

[1] Wu, Tailin, *et al.* "Learning large-scale subsurface simulations with a hybrid graph network simulator." SIGKDD 2022.

Case study: GNN-based simulation



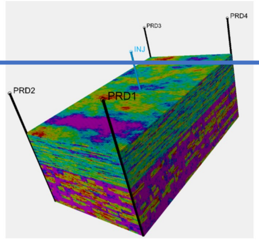
GNS

(Sanchez-Gonzalez et al., ICML 2020)

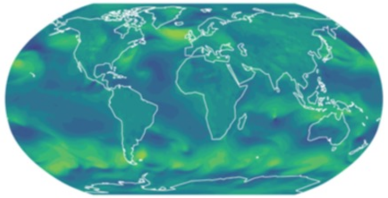
large-scale

HGNS

(Wu et al., KDD 2022)



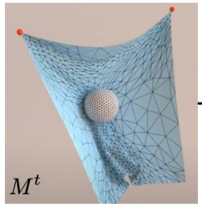
GraphCast



(Lam et al., Science 2023)

MeshGraphNets

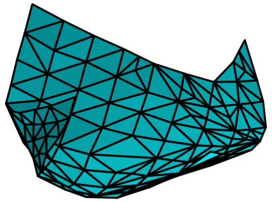
(Pfaff et al., ICLR 2021, outstanding paper)



multi-resolution

LAMP

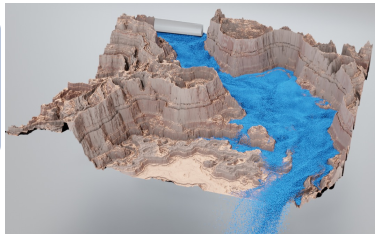
(Wu et al., ICLR 2023, spotlight)



symmetry

Momentum-conserved CConv

(Prantl et al., NeurIPS 2022)



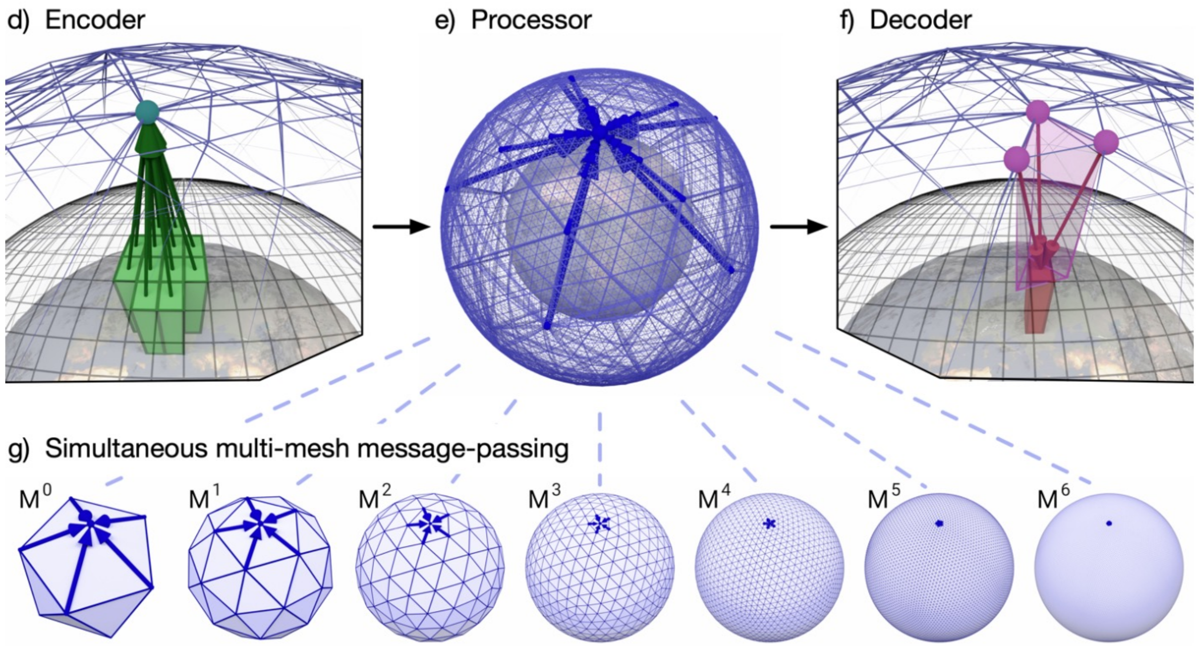
Case study: GraphCast

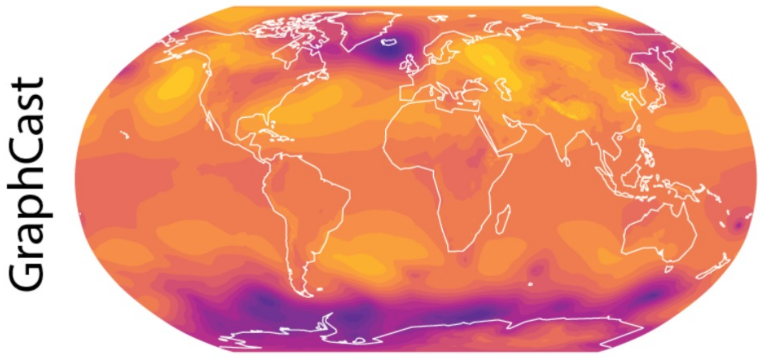
Task: Weather forecasting (mid-range, 10-day)

Main contribution: Introduced GraphCast [1]:

- Multi-scale GNN（多尺度图神经网络）
- Annealed multi-step learning objective

Results: outperforms state-of-the-art weather forecasting method (HRES) in 10-day prediction acc.

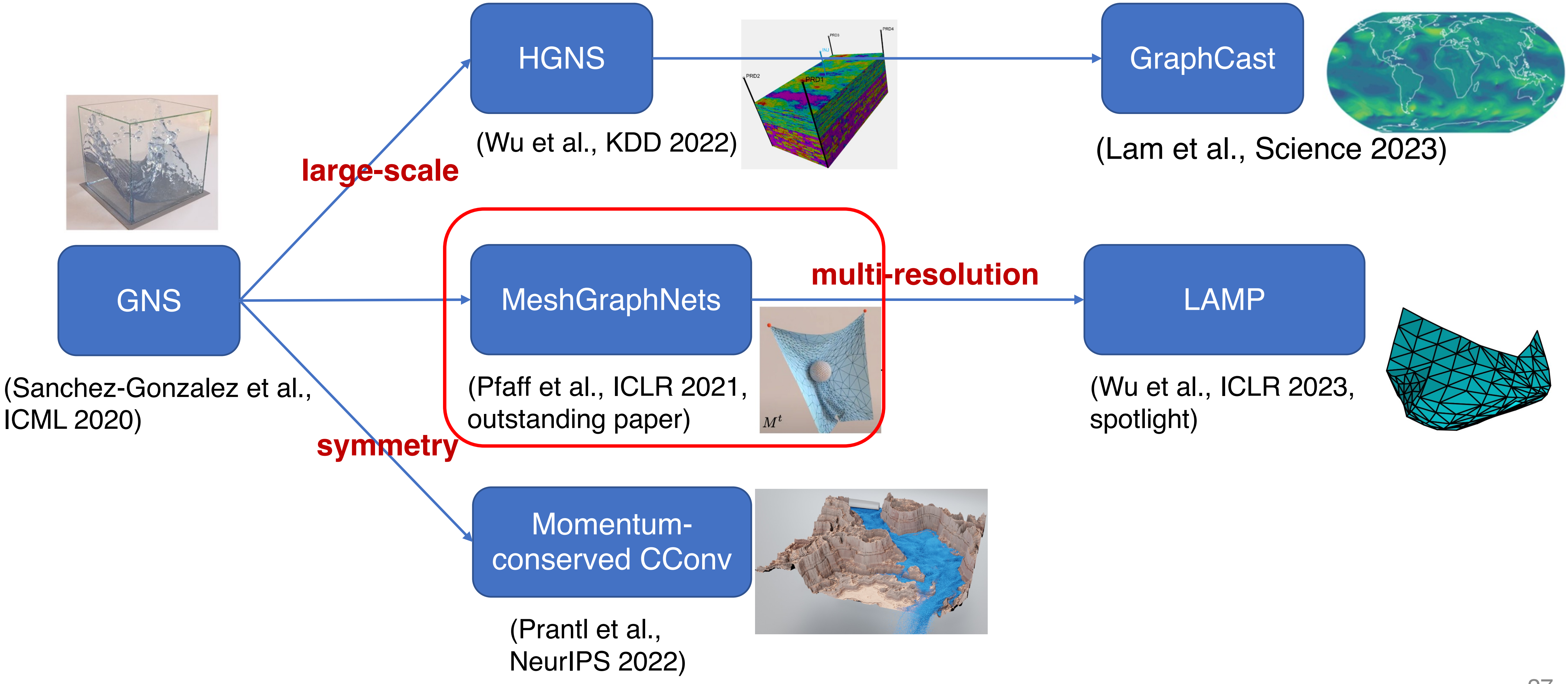




Prediction by GraphCast

[1] Lam, Remi, et al. "Learning skillful medium-range global weather forecasting." *Science* (2023): eadi2336.

Case study: GNN-based simulation

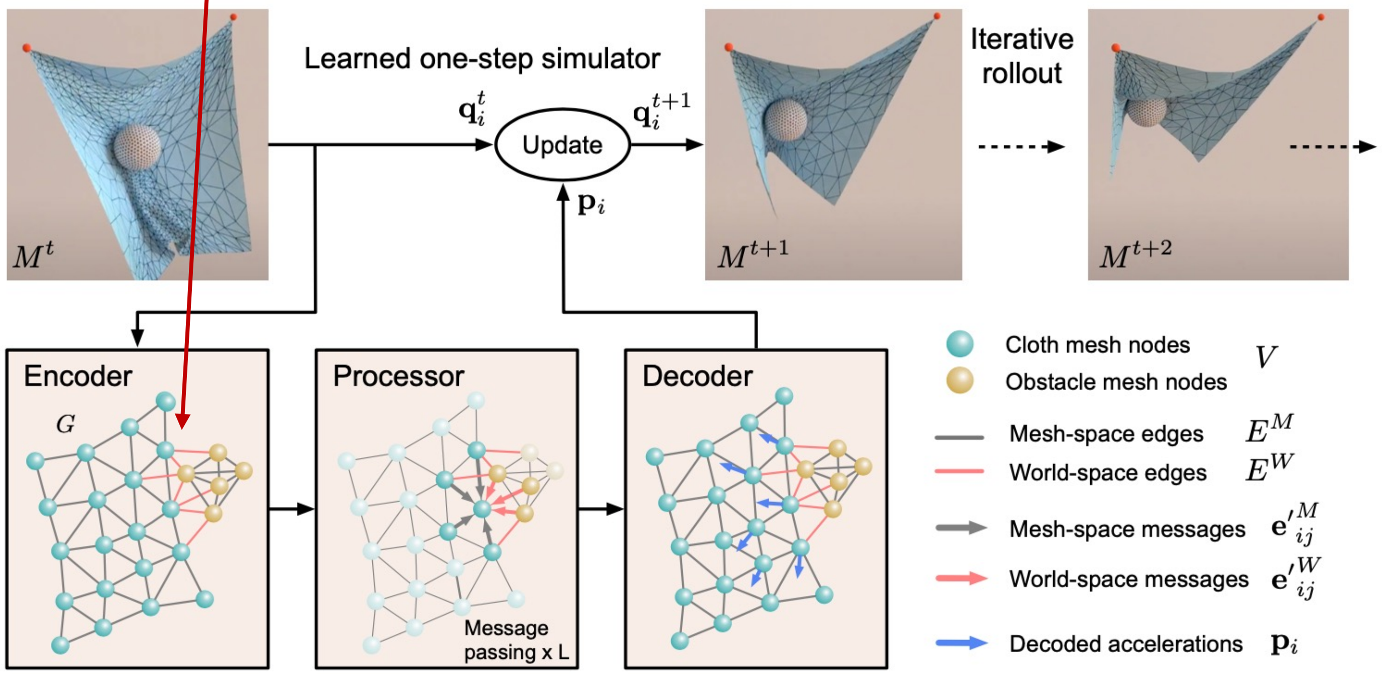


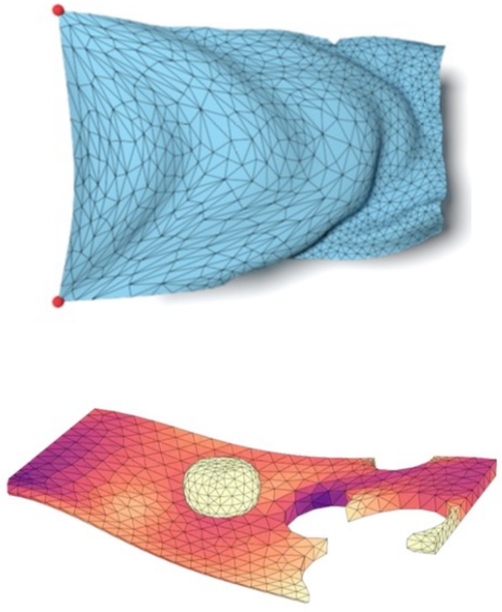
Case study: MeshGraphNets

Task: Mesh-based simulation

Main contribution: Introduced MeshGraphNets [1]:
- Two types of edges (mesh-space and world-space edges)
- Supervised remeshing

Results: accurate prediction on many different systems.

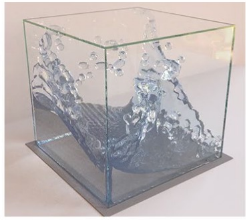




Example predictions

[1] Pfaff, Tobias, et al. "Learning mesh-based simulation with graph networks." *ICLR* 2021

Case study: GNN-based simulation



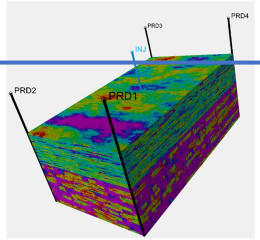
GNS

(Sanchez-Gonzalez et al., ICML 2020)

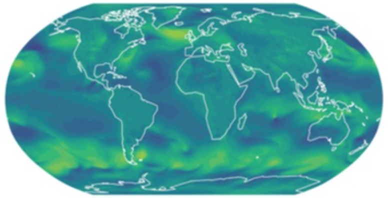
large-scale

HGNS

(Wu et al., KDD 2022)



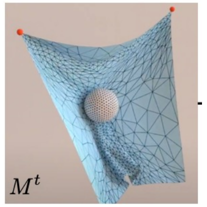
GraphCast



(Lam et al., Science 2023)

MeshGraphNets

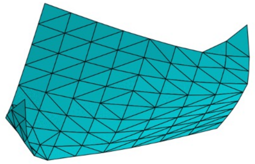
(Pfaff et al., ICLR 2021, outstanding paper)



multi-resolution

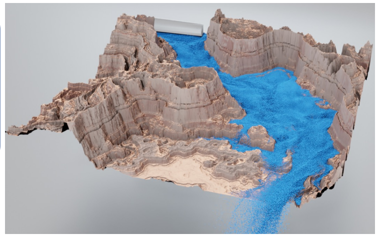
LAMP

(Wu et al., ICLR 2023, spotlight)



symmetry

Momentum-conserved CConv



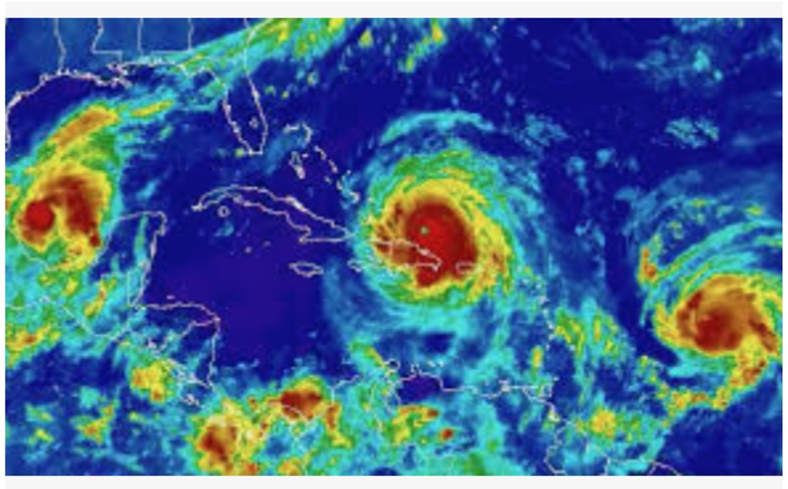
(Prantl et al., NeurIPS 2022)

Simulating multi-resolution dynamics: significance

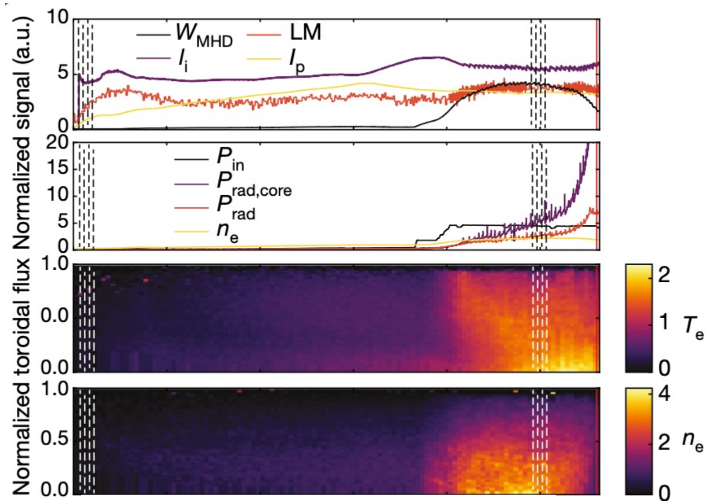
(less computation)

How to simulate a **multi-resolution**（多分辨率） dynamical system in an **accurate** and **efficient** way.

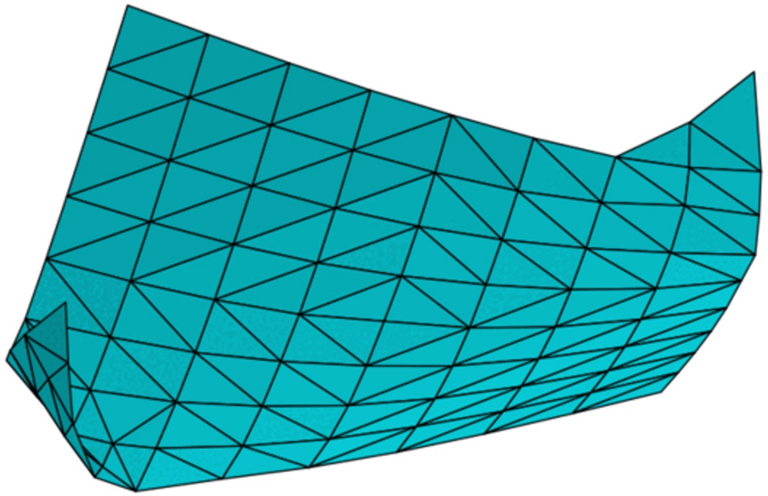
Multi-resolution systems are prevalent across different disciplines, where a small subset of the system is highly dynamic, and requires delicate simulation



Weather prediction



Disruptive instabilities in controlled fusion plasmas



Simulating cloth

Limitation of prior methods for multi-resolution challenges

However, current methods are **insufficient** to address the **multi-resolution** challenges

- Today's deep learning-based surrogate models mostly optimize the prediction accuracy, **without optimizing** the computational cost
- Classical solvers use heuristics for remeshing, which is suboptimal

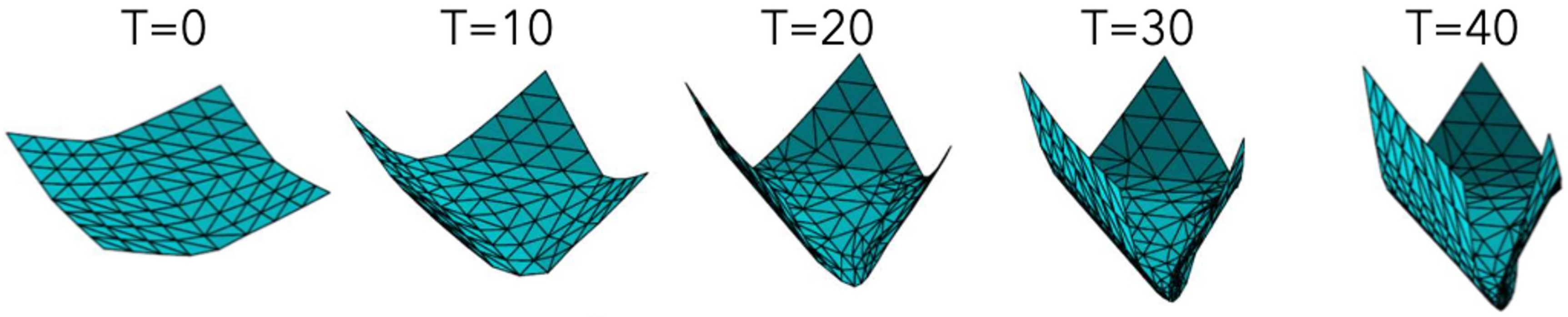
My contribution

Wu, Tailin, et al. "Learning Controllable Adaptive Simulation for Multi-resolution Physics." ICLR 2023 notable top-25%

We introduced the **first** deep learning-based surrogate model that jointly learns the **evolution** and optimize **computational cost**.

$$L = (1 - \beta) \cdot \mathrm{Error} + \beta \cdot \mathrm{Computation}$$

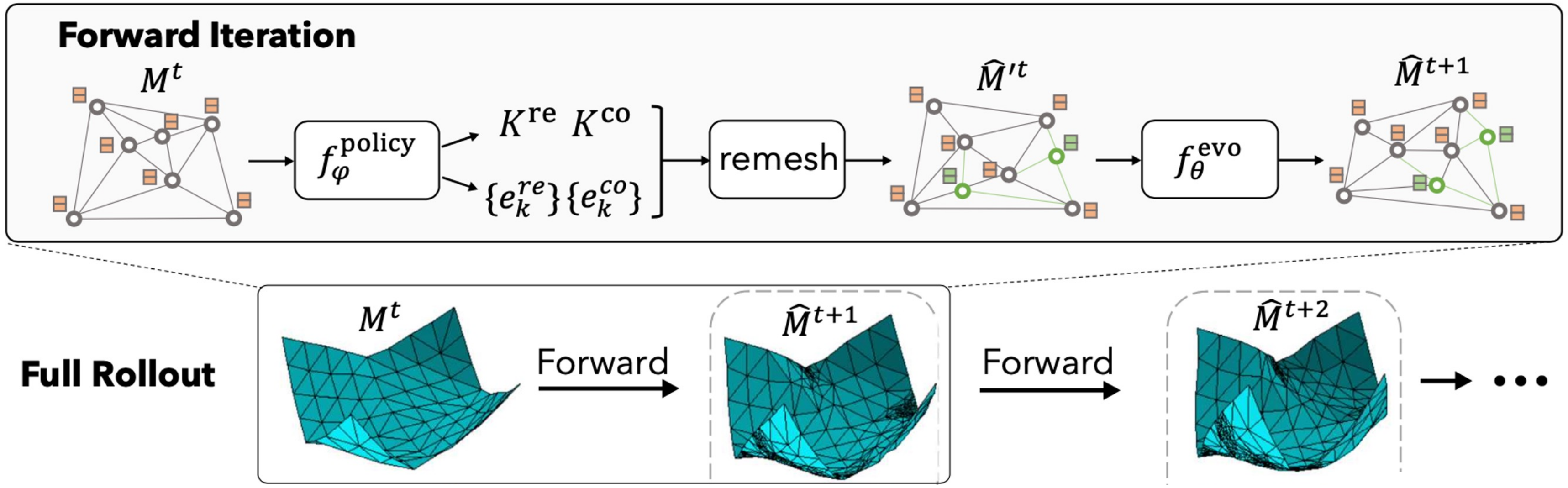
Key component: GNN-based RL agent, which learns to coarsen or refine the mesh, to achieve a controllable **tradeoff** between **prediction error** and **computational cost**.



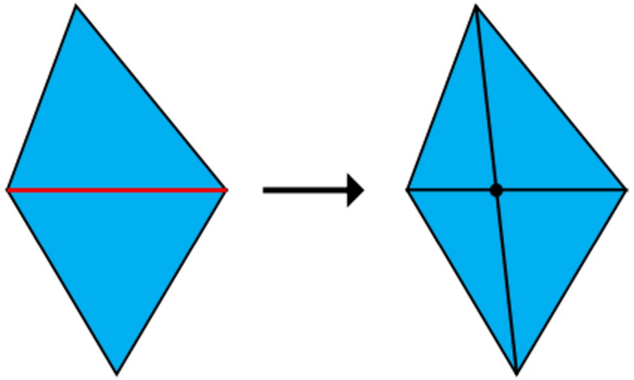
Method

f_θ^{evo} : GNN-based evolution model, evolving the system while keeping the mesh topology

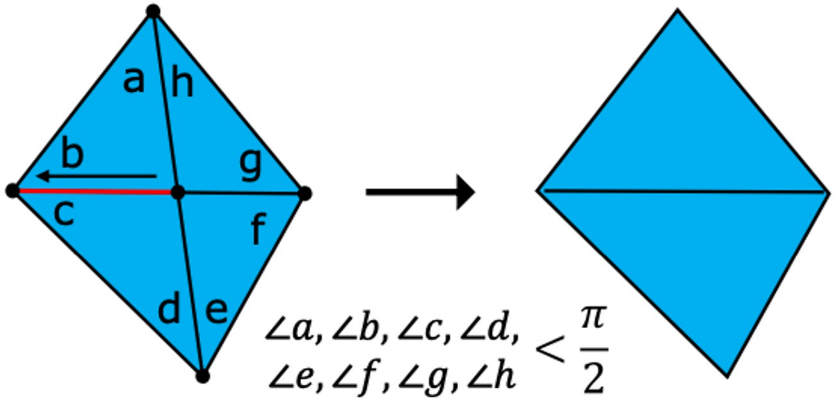
$f_\varphi^{\text{policy}}$: GNN-based policy, which refines/coarsens the mesh based on current state and β



Action space



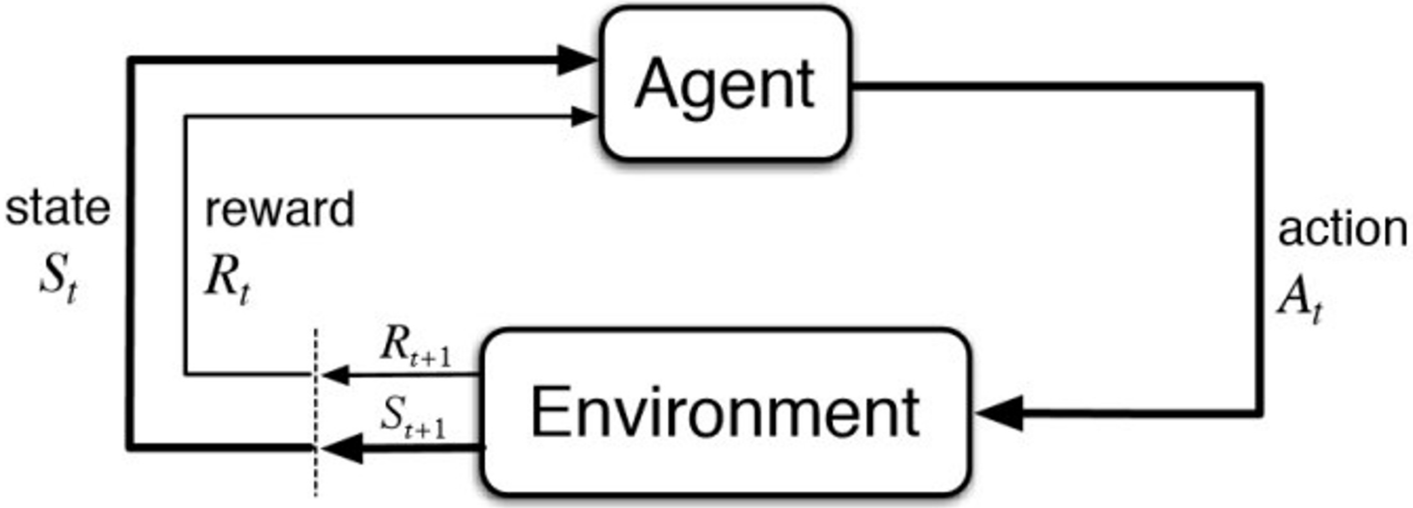
(1) **Refining** an edge



(2) **Coarsening** an edge

Such action is performed on **all** the cells simultaneously

Reinforcement learning 强化学习



Environment: the tokamak

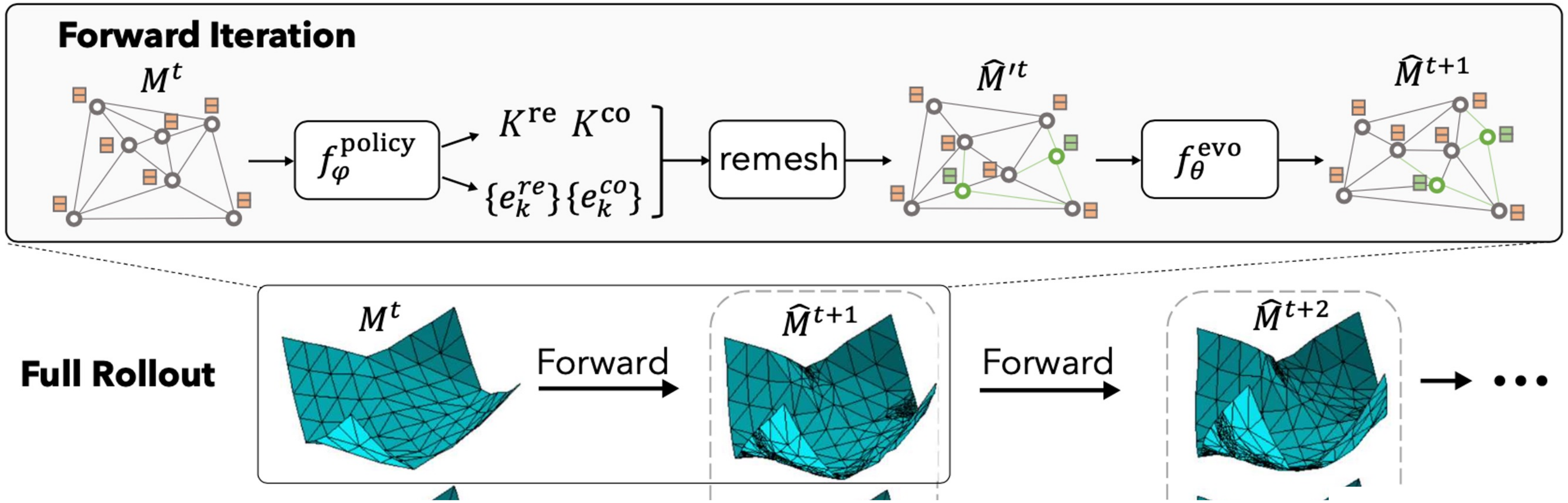
Goal: maximize the long-term expected reward w.r.t. to the policy $\pi(A_t|S_t)$

$$\max_{\pi(A_t|S_t)} \mathbb{E}_t[R_t]$$

Method

f_θ^{evo} : GNN-based evolution model, evolving the system while keeping the mesh topology

$f_\varphi^{\text{policy}}$: GNN-based policy, which refines/coarsens the mesh based on current state and β



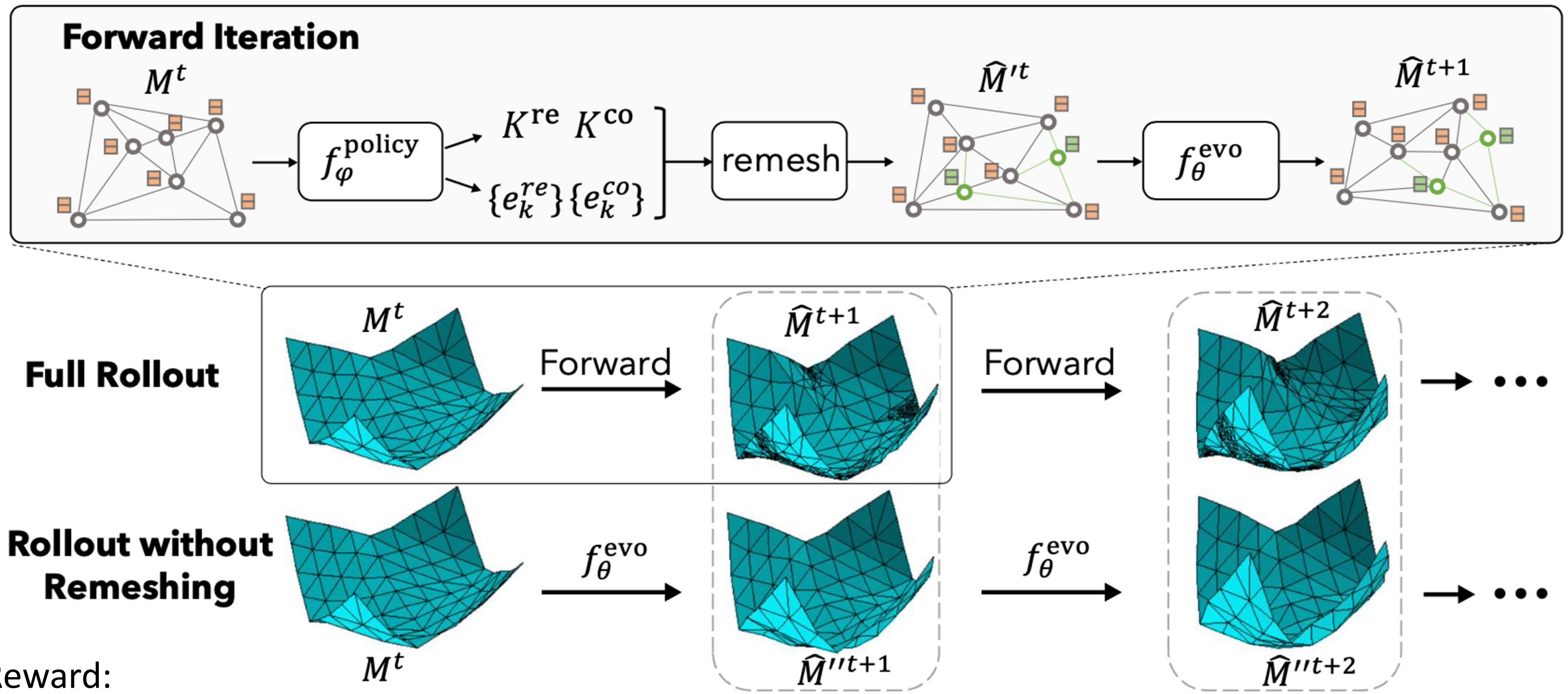
奖励:

$$r^t = (1 - \beta) \cdot \Delta\text{Error} + \beta \cdot \Delta\text{Computation}$$

预测误差的降低　　　　计算成本的降低

Method

- f_θ^{evo} : GNN-based evolution model, evolving the system while keeping the mesh topology
- $f_\varphi^{\text{policy}}$: GNN-based policy, which refines/coarsens the mesh based on current state and β



奖励 Reward:

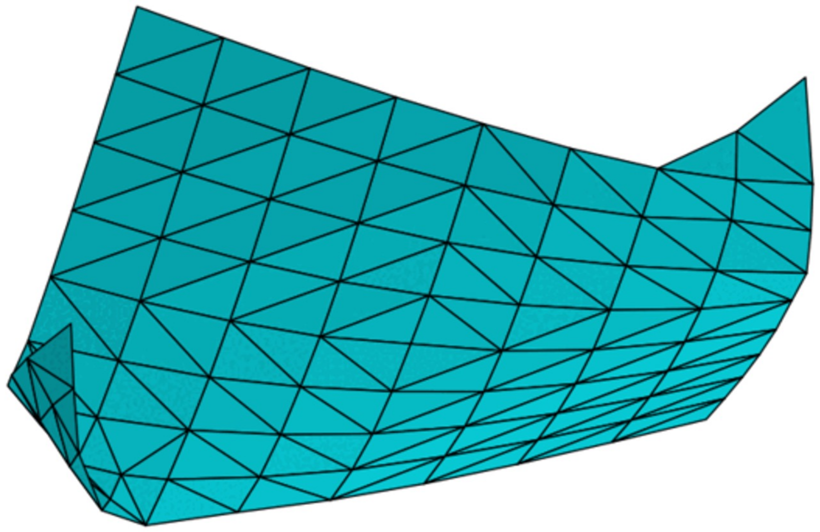
$$\boxed{r^t = (1-\beta)\cdot\Delta\text{Error} + \beta\cdot\Delta\text{Computation}}$$

Reward is based on the **improvement** of both **error** and **computational cost**.

- **Error** is the multi-step prediction error
- **Computational cost** is measured by number of vertices in the mesh

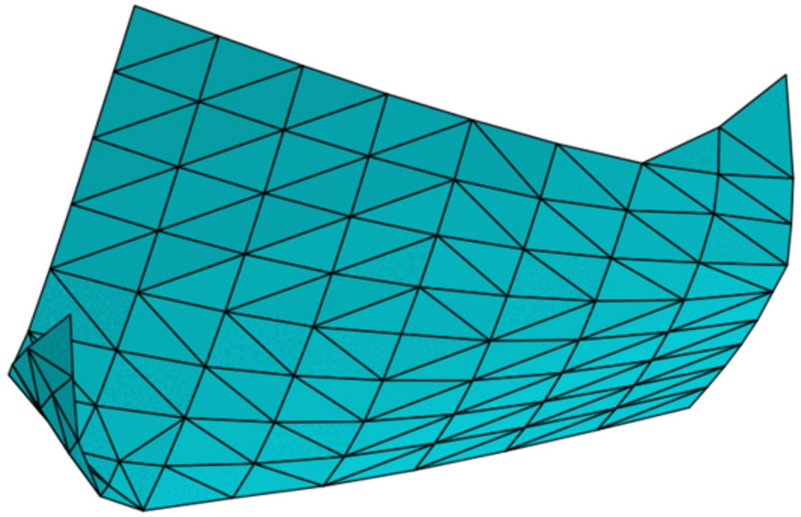
β is also an input to the policy

Experiment 2: mesh-based simulation visualization



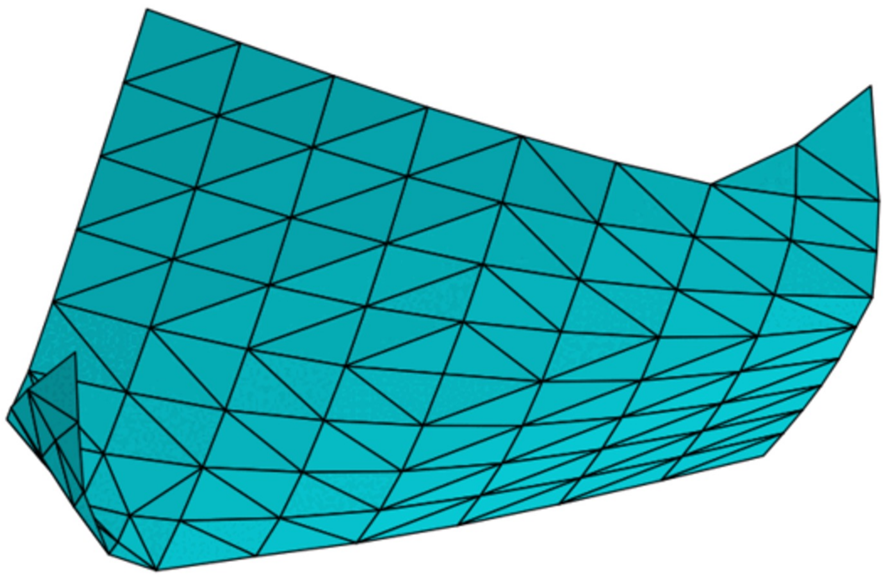
MeshGraphNets + GT remeshing

MSE: 5.91e-4



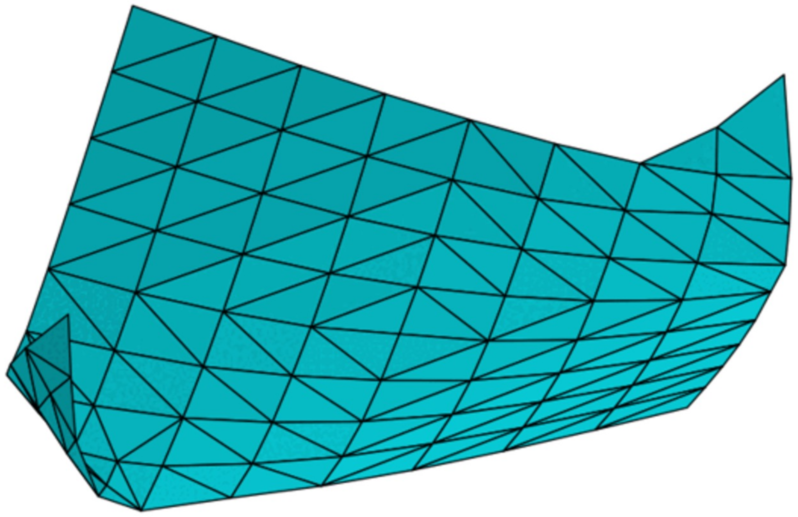
ground-truth (fine-grained)

Experiment 2: mesh-based simulation visualization



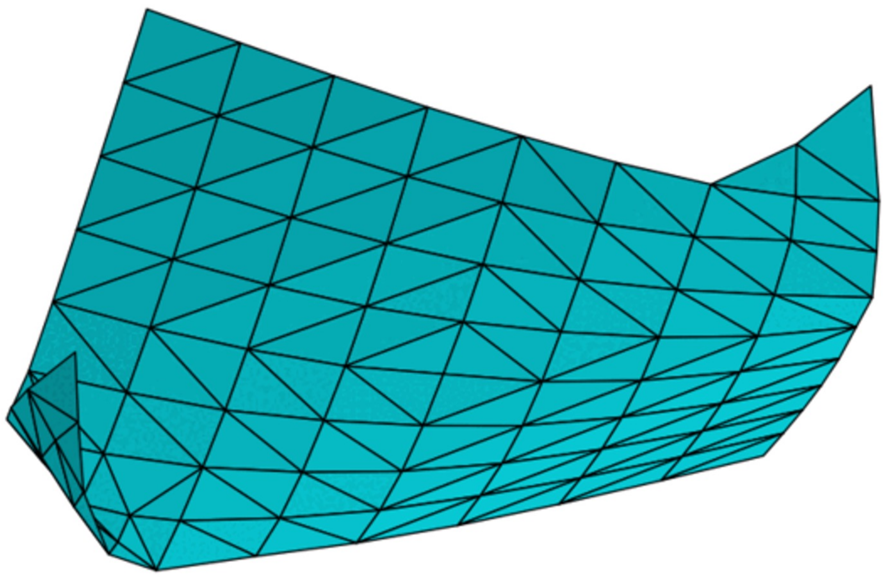
LAMP + no remeshing

MSE: 6.13e-4



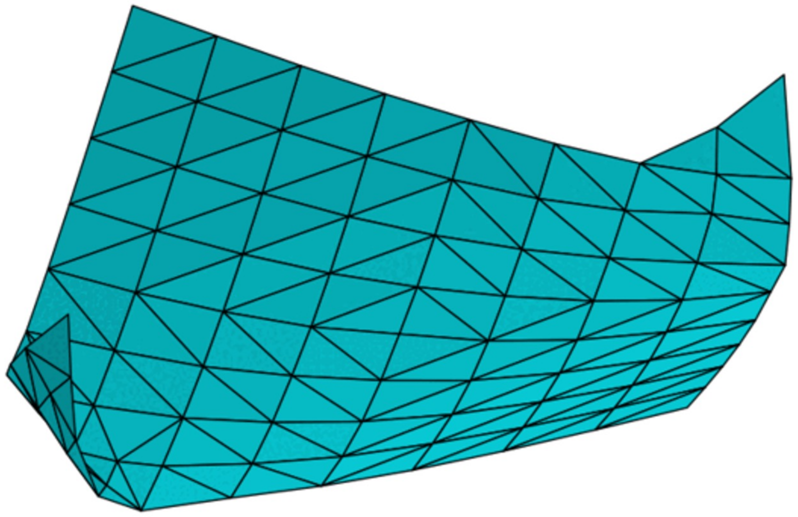
ground-truth (fine-grained)

Experiment 2: mesh-based simulation visualization



LAMP (ours)

MSE: 5.80e-4



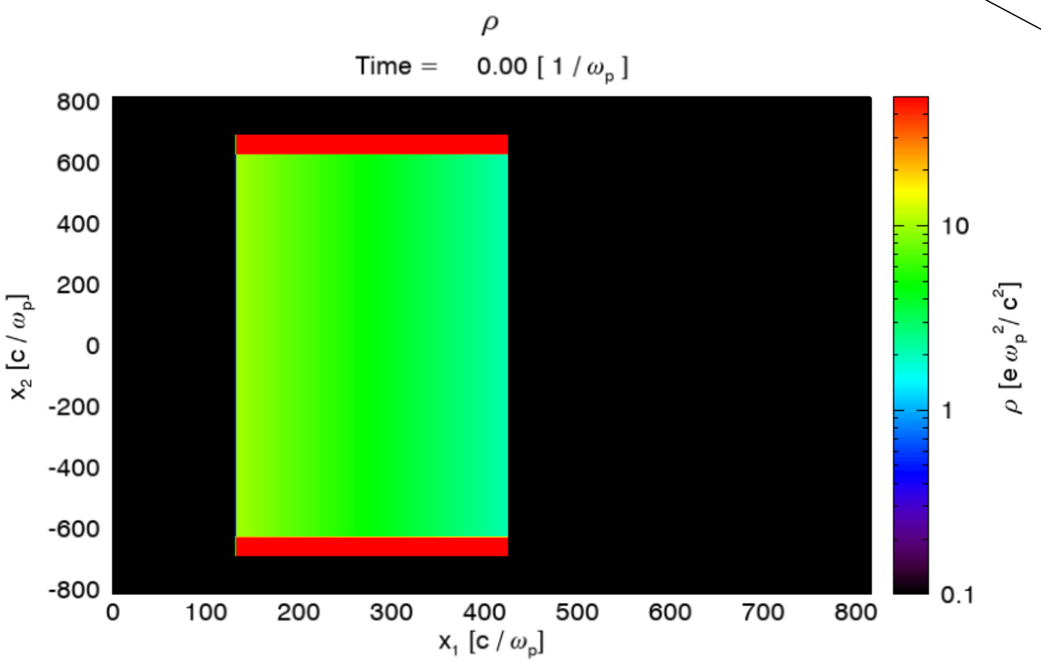
ground-truth (fine-grained)

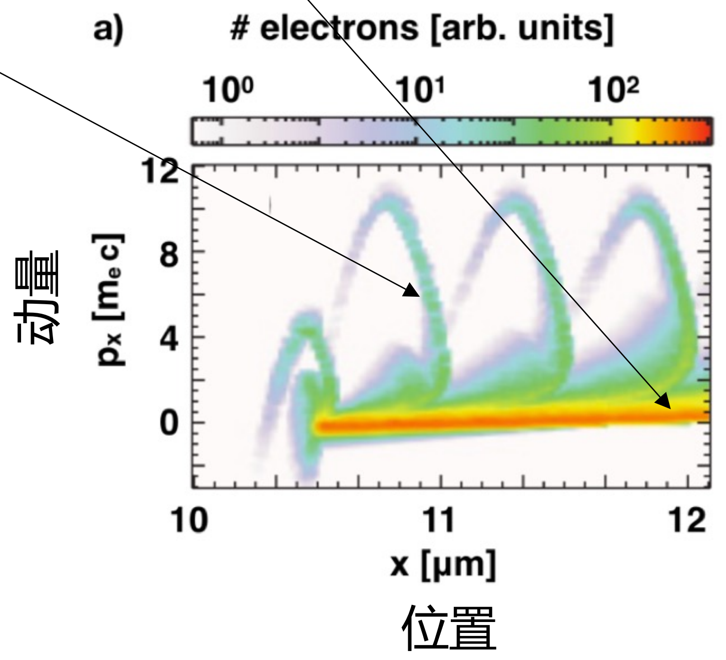
我其他相关工作：

激光-等离子体相互作用（与SLAC合作）

任务： 更快、更好地模拟这一过程

难点： 动量空间多尺度（少部分粒子接近光速远离平衡，大部分粒子近平衡，动量分布符合高斯分布）





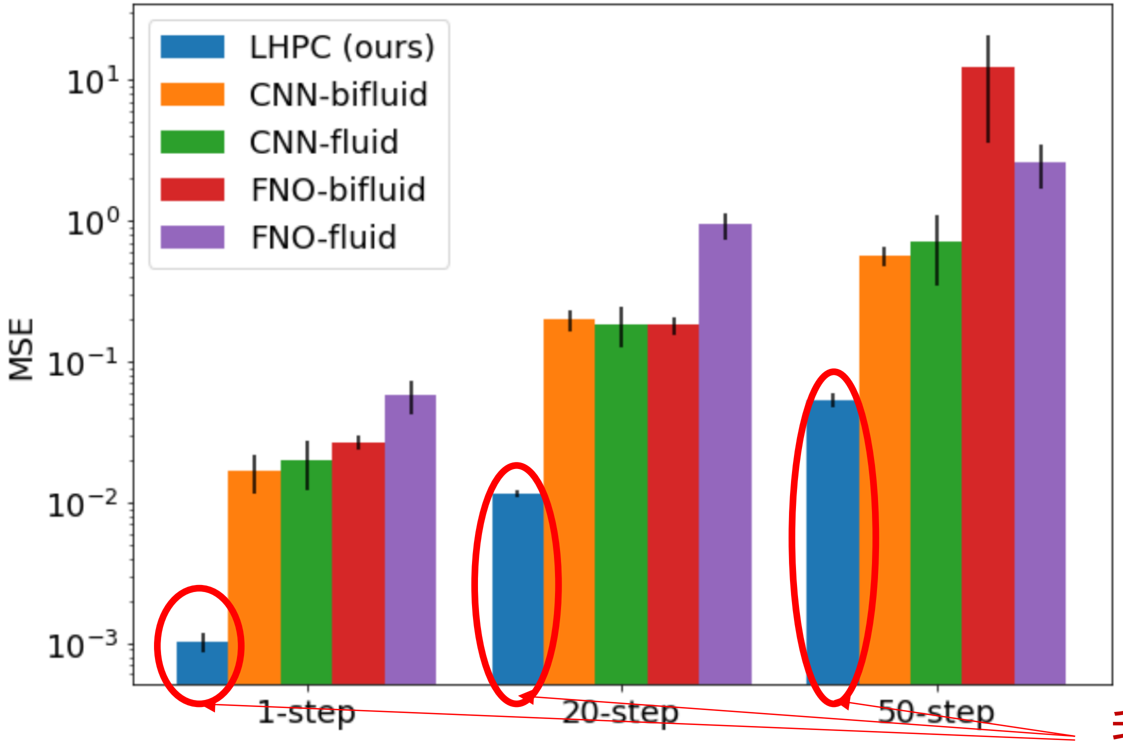
我其他相关工作：

[1] **Wu, Tailin**, et al. "Learning Efficient Hybrid Particle-continuum Representations of Non-equilibrium N-body Systems." (2022). NeurIPS 2022 AI4Science

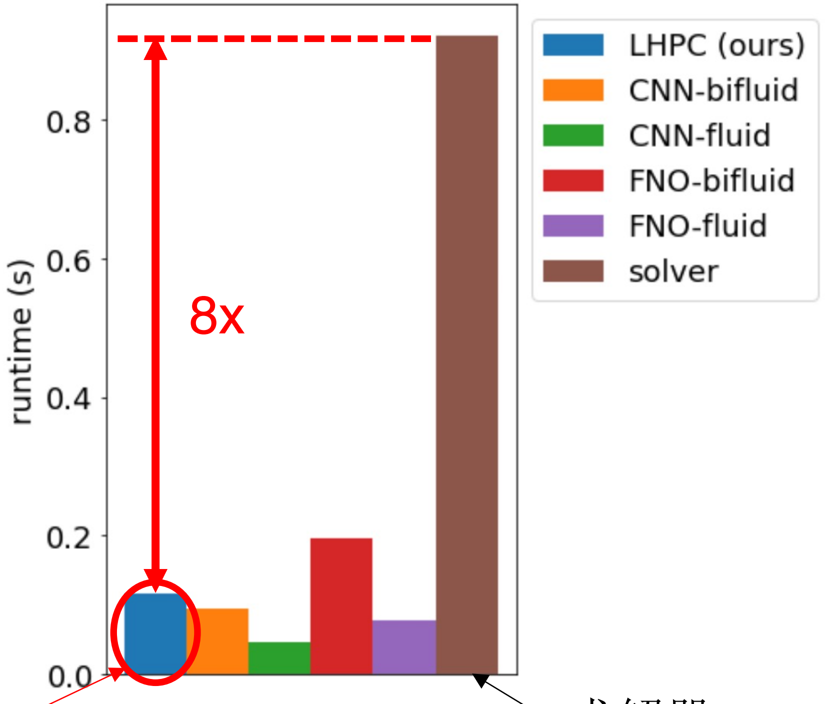
方法[1]：学习粒子-流体表示。用**粒子**表示远离平衡的粒子并通过求解器仿真，用**流体**表示大部分的近平衡粒子并通过神经网络学习其演化和粒子-流体的**耦合**

结果：相对求解器**8倍加速**，相对其他神经网络模型**10倍误差减小**，实现误差和计算成本的均衡

预测误差（EM field)：



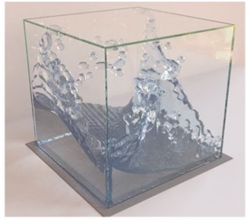
运行时间：



我们模型

求解器

Case study: GNN-based simulation



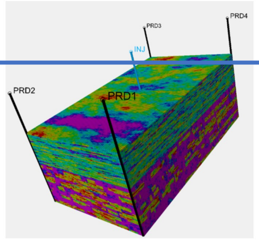
GNS

(Sanchez-Gonzalez et al., ICML 2020)

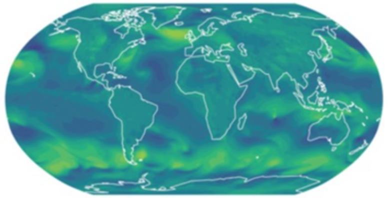
large-scale

HGNS

(Wu et al., KDD 2022)



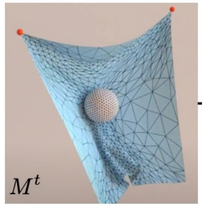
GraphCast



(Lam et al., Science 2023)

MeshGraphNets

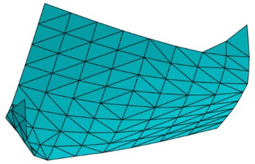
(Pfaff et al., ICLR 2021, outstanding paper)



multi-resolution

LAMP

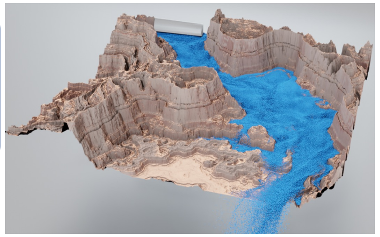
(Wu et al., ICLR 2023, spotlight)



symmetry

Momentum-conserved CConv

(Prantl et al., NeurIPS 2022)



Other examples where **symmetry** helps:

- Equivariant GNNs [1] for molecular prediction
- Deep Potential Molecular Dynamics [2]

[1] Satorras, Vıctor Garcia, Emiel Hoogeboom, and Max Welling. "E (n) equivariant graph neural networks." *ICML*, 2021.
[2] Zhang, Linfeng, et al. "Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics." *Physical review letters* 120.14 (2018): 143001.

AI for scientific simulation: part I summary

两类任务：学习系统动力学和稳态

两个重要架构：图神经网络和神经算子

两个挑战和解决方式：多分辨率和函数映射

Tasks

- Learning dynamics
- Learning steady state

×

Architecture

- GNN
- Neural operators
- Transformer
- U-Net

×

Learning paradigm

- Regression
- Diffusion model
- Reinforcement learning
- Physics-informed

×

Challenge

- Multi-resolution
- Mappings between functions

AI for scientific simulation: part II

两类任务：学习系统动力学和稳态

两个重要架构：图神经网络和神经算子

两个挑战和解决方式：多分辨率和函数映射

Tasks

- Learning dynamics
- Learning steady state

×

Architecture

- GNN
- Neural operators
- Transformer
- U-Net

×

Learning paradigm

- Regression
- Diffusion model
- Reinforcement learning
- Physics-informed

×

Challenge

- Multi-resolution
- Mappings between functions

Neural operator：Mapping from functions to functions

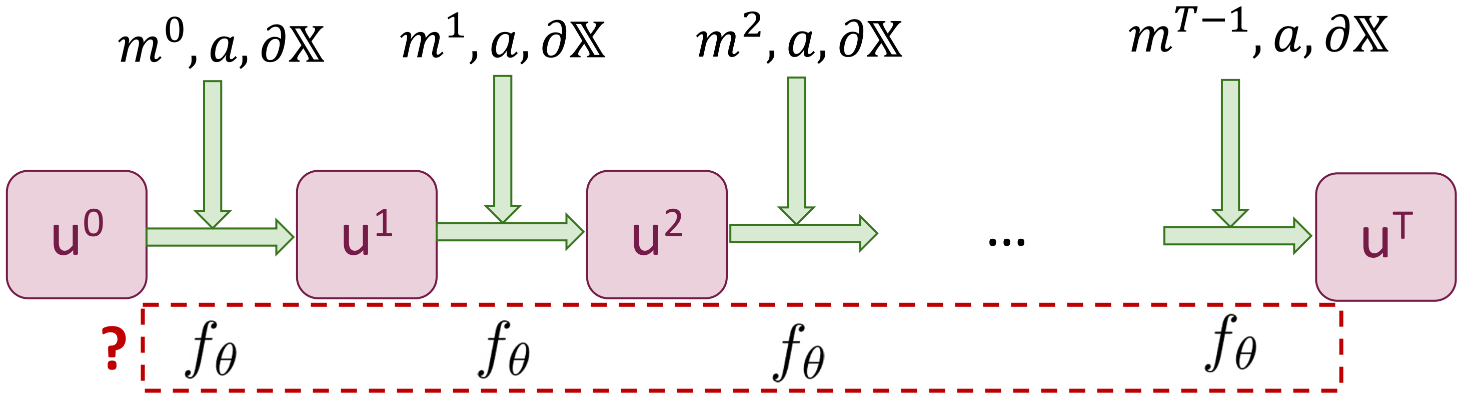
neural operator（神经算子）

$$a(x) \xrightarrow{f_\theta\ ?} u(x,t)$$

函数　　　　　　　　函数

Task setup 1: learning dynamics

Goal: learn the mapping f_θ from u^t to u^{t+1}:



u^t: original **state**（状态）of the system. Can be an **infinite-dimensional function** $u(t, x)$ as solution to a PDE, or a graph (e.g., mesh, particle-based systems, molecules)

f_θ: **neural operators（神经算子）**

m^t: **external control**（外界控制）

a: **static parameters**（静态参数） of the system that does not change with time (e.g. parameters of PDE, spatially varying diffusion coefficient)

$\partial\mathbb{X}$: **boundary condition**（边界条件） of the system

PDE: partial differential equation

Task setup 2: learning steady state 学习稳态解

neural operator（神经算子）

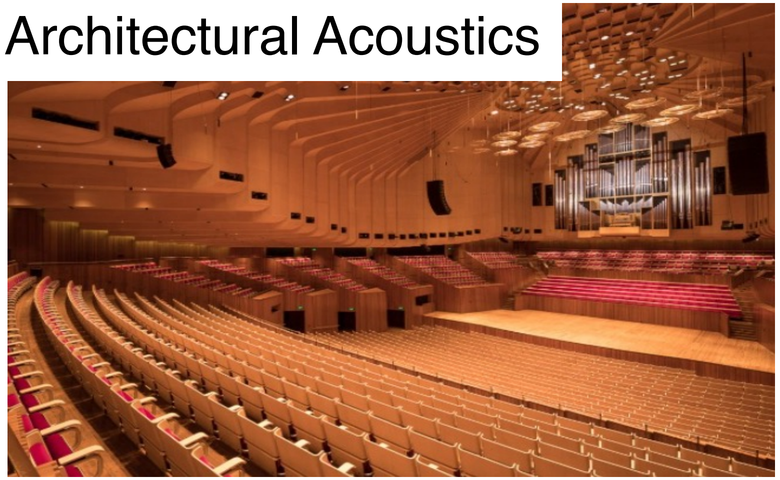
$$a(x), \partial\mathbb{X} \xrightarrow{f_\theta\ ?} u(x)$$

Parameter and boundary
参数和边界条件

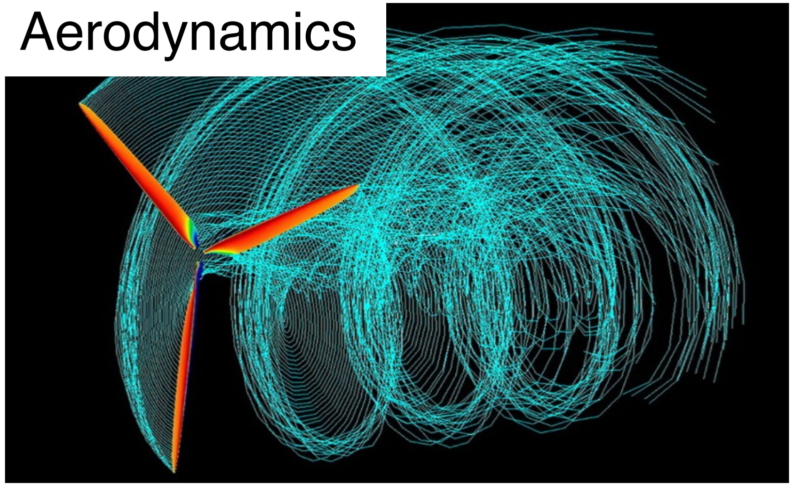
Solution
方程解

(boundary value problems)

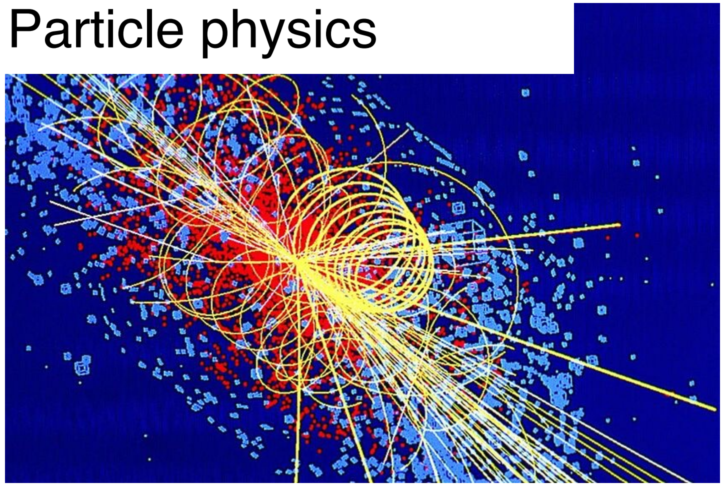
Architectural Acoustics



Aerodynamics



Particle physics



Elliptic PDEs（椭圆型偏微分方程）

There are three types of second-order PDEs, elliptic, parabolic (e.g., N-S equation), and hyperbolic (e.g., wave equation)

Elliptic PDEs are important across different scientific fields. Examples:

Poisson's equation 泊松方程:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in \mathbb{X}$$
$$\mathbb{B}[u(x)] = g(x), \quad x \in \partial\mathbb{X}$$

Important in materials, plasma physics, elasticity, hydrology.

Grad–Shafranov equation:

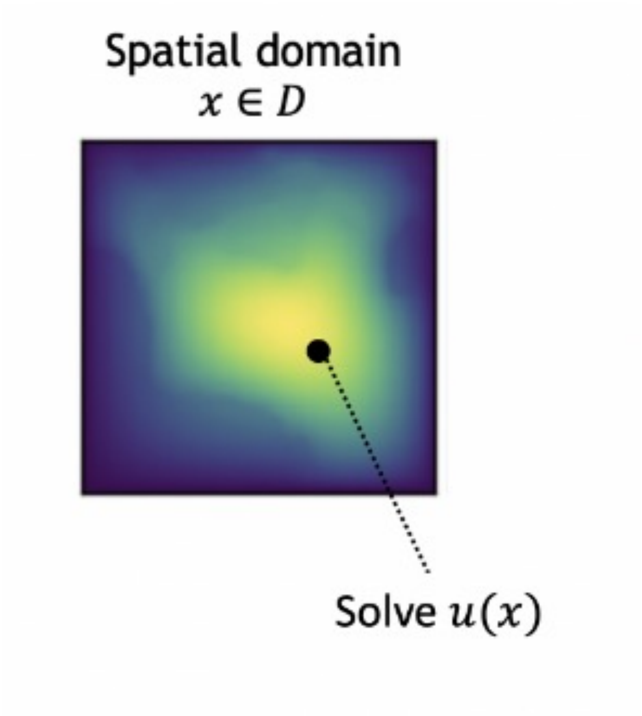
$$-\mu_\circ r^2 \frac{d}{d\Psi} - \frac{1}{2}\frac{dF^2}{d\Psi} = \frac{\partial^2\Psi}{\partial r^2} - \frac{1}{r}\frac{\partial\Psi}{\partial z^2} + \frac{\partial^2\Psi}{\partial z^2}$$

Important in *controlled nuclear fusion*（可控核聚变）

Solving vs. learning PDE

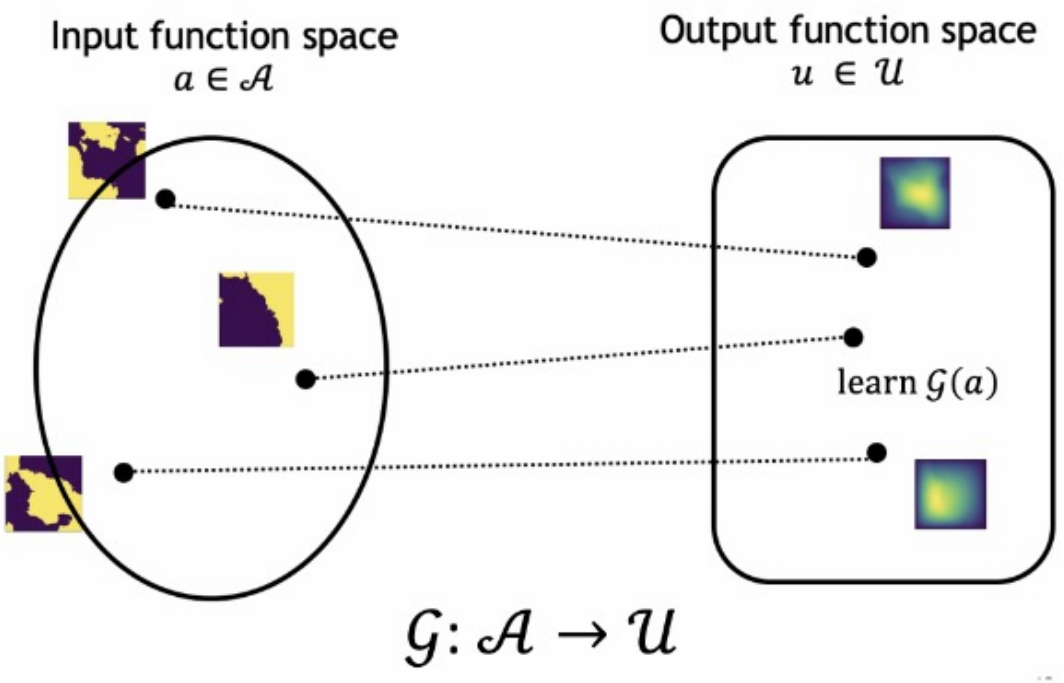
Solving for a PDE instance

Numerical solvers and PINNs focus on solving one specific instance



Learn solution operator $\mathcal{G}$

Neural operators learn the solution operator for a family of equations



Zongyi Li *et al.* (2021)

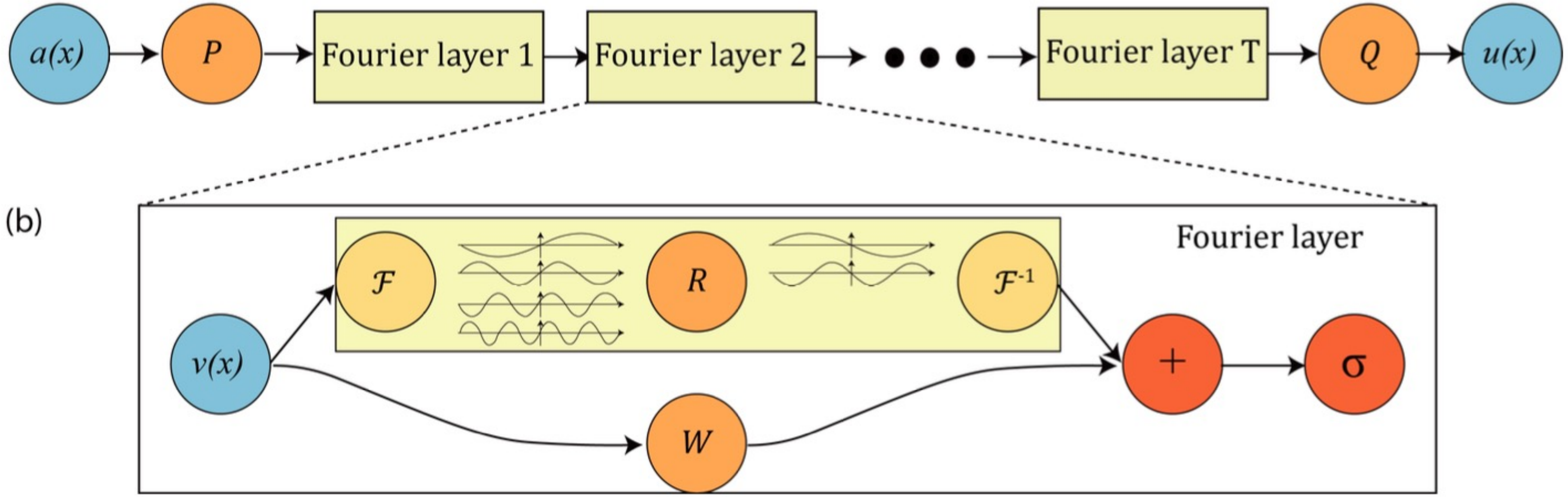
Neural operators

- Fourier Neural Operator (FNO)
    - Based on Fourier transformation

- Many others

Li, Zongyi, et al. "Fourier neural operator for parametric partial differential equations." ICLR 2021

Fourier Neural Operators (FNO)



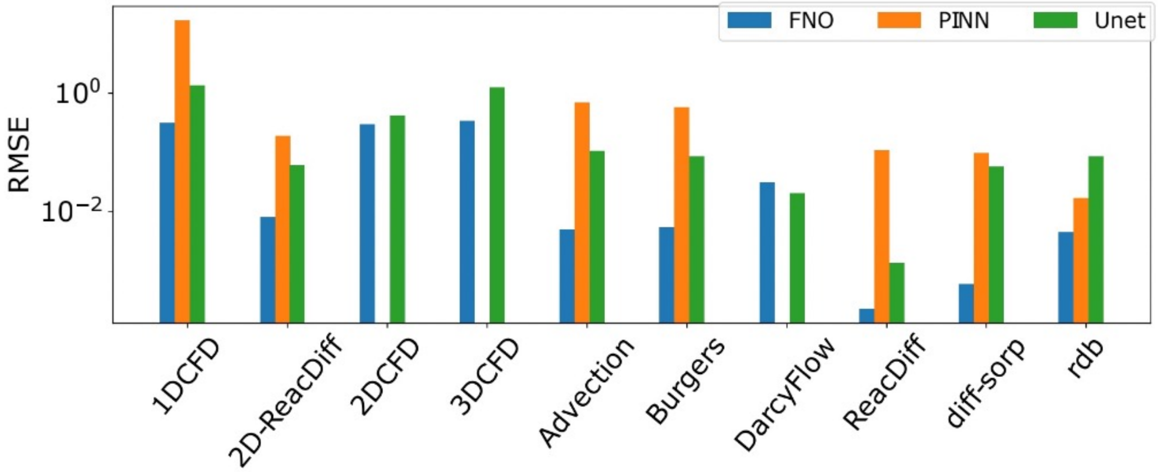
类比有限维的一层神经网络:

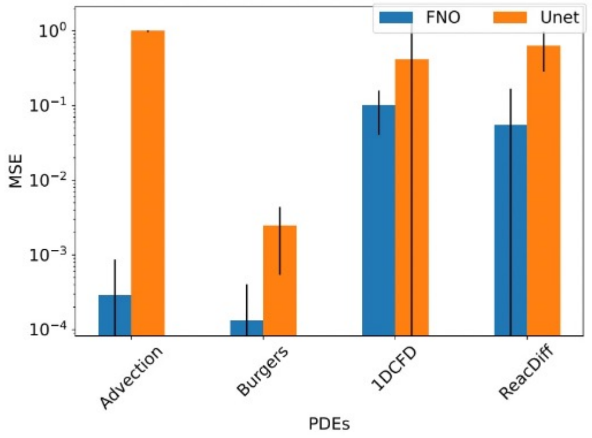
$$v_{L+1} = \sigma(b + Wv_L),\ v_L \in R^d, W \in R^{d'\times d}$$

$$v_{t+1}(x) := \underset{\text{非线性}}{\sigma}\Big(\underbrace{Wv_t(x)}_{\text{局域变换}} + \underbrace{\big(\mathcal{K}(a;\phi)v_t\big)(x)}_{\text{全局积分}}\Big)$$

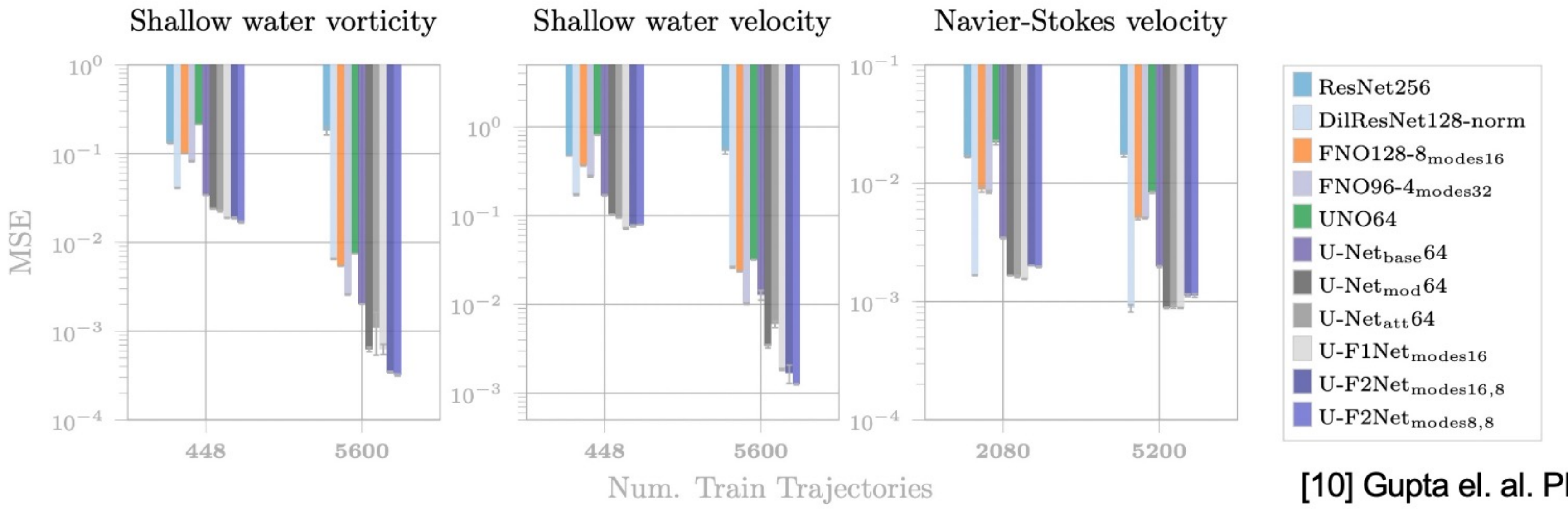
$$\big(\mathcal{K}(a;\phi)v_t\big)(x) := \int_D \kappa(x-y)v_t(y)dy = \mathcal{F}^{-1}\big(R_\phi \cdot \mathcal{F}v_t\big)(x)$$ （卷积在傅里叶空间中为相乘）

FNO compared to other neural networks



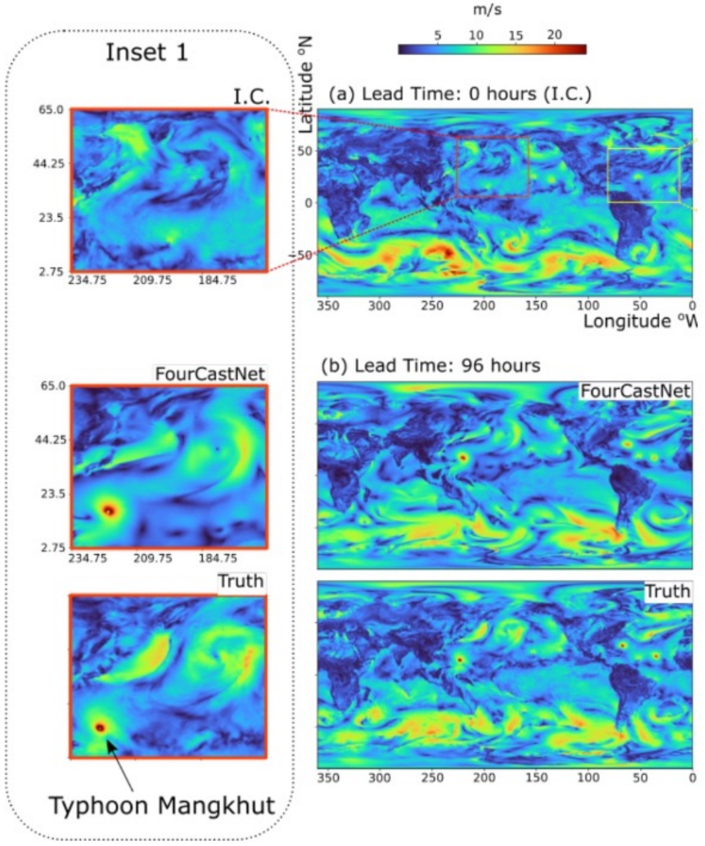


[9] Takamoto el. al. PDEBench Fig.2

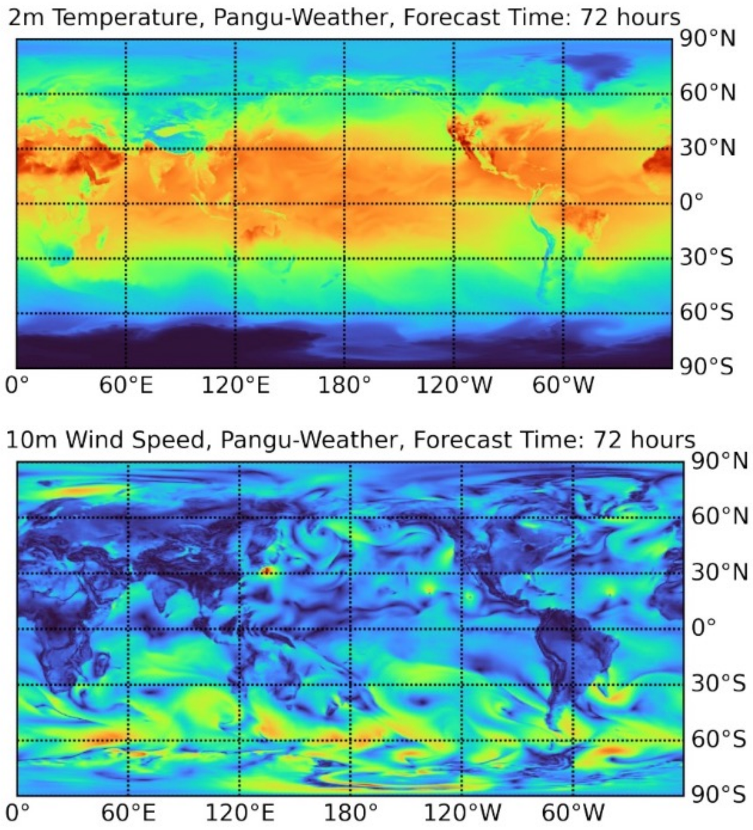


[10] Gupta el. al. PDEarena Fig.4

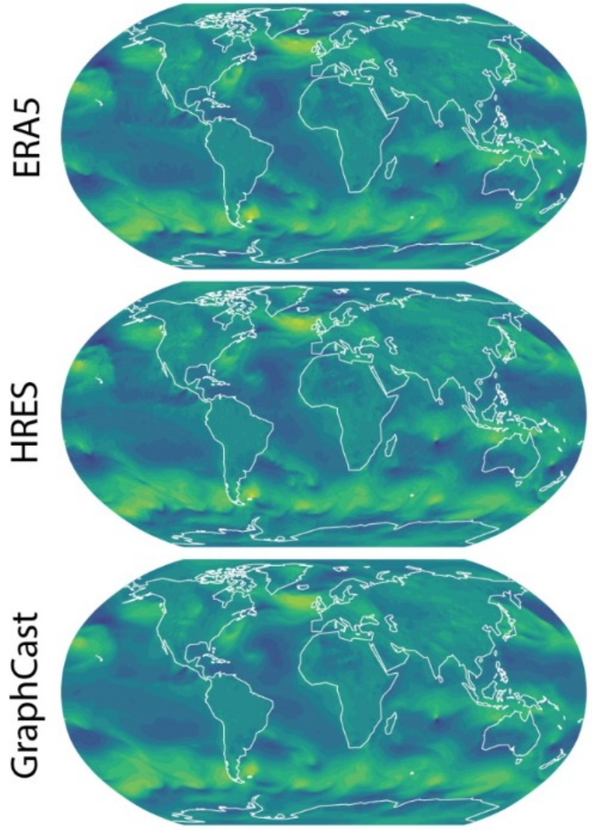
Weather forecast



Fourcastnet [1]
Jaideep Pathak et. al. Feb 2022



Pangu-weather [11]
Kaifeng Bi el. al. Nov 2022



GraphCast [12]
Remi Lam el. al. Dec 2022

FNO: advantages and disadvantages

Advantages:
- Typically very good accuracy, near state-of-the-art
- Can do super-resolution (超分辨率)

Disadvantages:
- Learns a global mapping, requires large amount of training samples
- Requires regular grid

Summary for neural architectures

	优点	缺点
GNN	显示地对对象和它们之间关系建模，适用于描述相互作用复杂、非规则网格等。需要较少的样本	基础的GNN难以对长程影响建模，需要添加多尺度的边（类似GraphCast）
Transformer	比较适合建模长程关系	参数量较多，需要较多的训练数据
U-Net	能够建模规则网格中多尺度的动力学	只能用于规则网格
FNO	能够实现超分辨率	只能用于规则网格、需要较多训练数据

Summary for learning paradigms

	适合场景	缺点
Regression	最常用场景	学习的代理模型对于分布外数据泛化性较差；预测效果不会超出所给的目标
Diffusion model	适用于任何regression用的场景，更适合于高维系统的预测、设计和控制，泛化性更强	需要一定量的训练数据（但随维度增加，训练数据需要量增加没有Regression快）
Reinforcement learning	预测效果需要超出所给的目标；整个环境无法求导	样本效率较低，需要与环境的大量的交互
Physics-informed	知道系统的控制方程，可以减少样本的需要量	难以泛化到新的边界或者初始条件；系统控制方程不一定准确

*Diffusion model: See second part

Open questions

AI for scientific simulation:

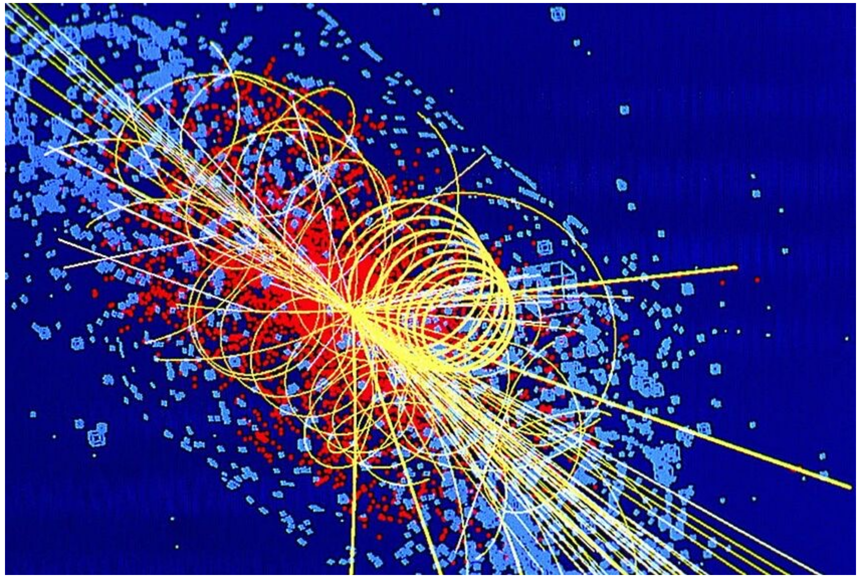
- Multi-scale
- Improving trustworthiness:
  - Uncertainty quantification 不确定性估计 [1]
  - Error guarantees, 误差保证:
    - e.g., with conformal prediction [2]
- Better incorporation of data and physics equation
  - Existing works:
    - Solver-in-the-loop [3]
    - PEDS [4]
    - Physics-informed DeepONet [5]
    - Physics-informed diffusion model [6]

[1] Wu, Tailin *et al.*, Uncertainty Quantification for Forward and Inverse Problems of PDEs via Latent Global Evolution, AAAI 2024, https://github.com/AI4Science-WestlakeU/le-pde-uq
[2] Stankeviciute, Kamile, Ahmed M Alaa, and Mihaela van der Schaar. "Conformal time-series forecasting." *Advances in neural information processing systems* 34 (2021): 6216-6228.
[3] Um, Kiwon, et al. "Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers." *NeurIPS* 2020, 6111-6122.
[4] Pestourie, Raphaël, et al. "Physics-enhanced deep surrogates for partial differential equations." *Nature Machine Intelligence* (2023): 1-8.
[5] Wang, Sifan, Hanwen Wang, and Paris Perdikaris. "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets." *Science advances* 7.40 (2021): eabi8605.
[6] Shu, Dule, Zijie Li, and Amir Barati Farimani. "A physics-informed diffusion model for high-fidelity flow field reconstruction." *Journal of Computational Physics* 478 (2023): 111972.

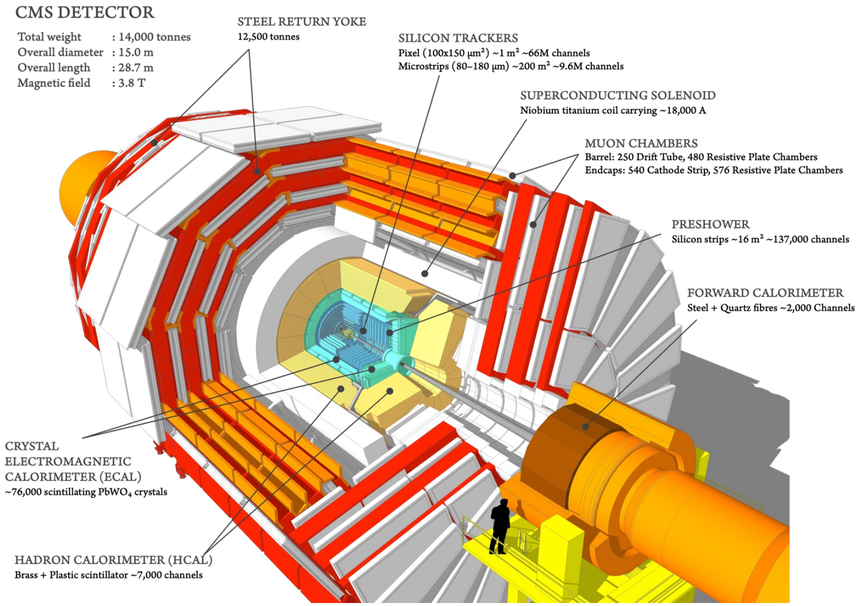
Useful AI4Science resources

- AI for Science综述：Review paper "Artificial intelligence for science in quantum, atomistic, and continuum systems."
- Scientific Discovery in the Age of Artificial Intelligence, *Nature* 2023
- My course: Frontiers in Computer Science and Technology, introducing important topics of AI and AI + Science.
- 集智AI + Science读书会
    - 第一期：AI + Science: motivation, advances and open problems [slides]
    - 第三期: 利用AI代理模型和扩散模型辅助科学设计 [代理模型slides][扩散模型slides]
    - 第八期：数据驱动的物理仿真模拟：神经算子与图神经网络 [神经算子slides][图神经网络slides]

Forward ⇄ Inverse



[Image from University of Vienna]

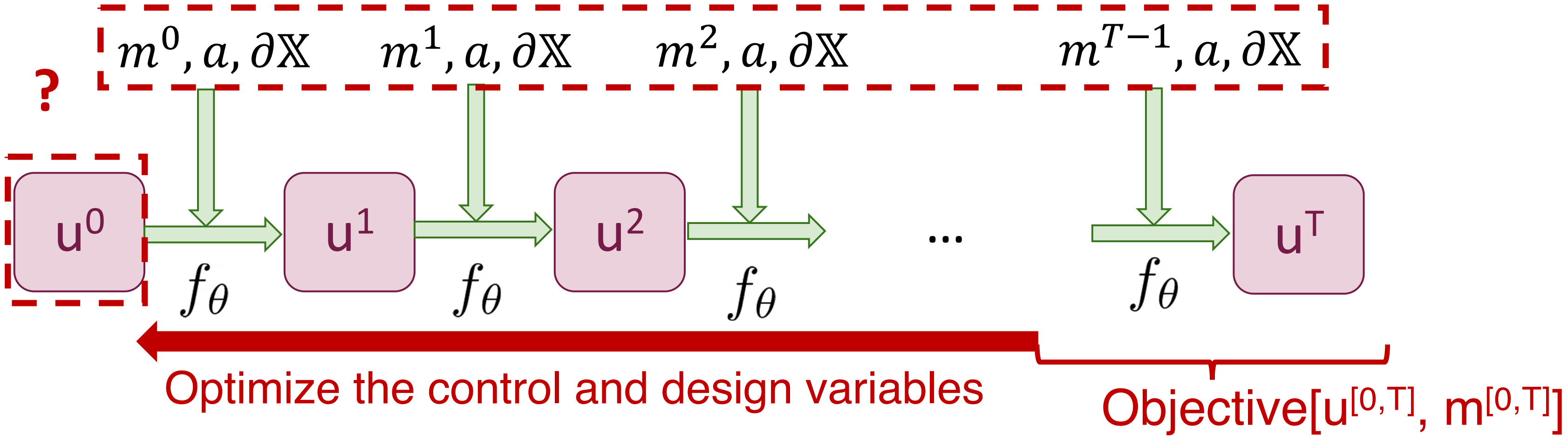


- Simulation
- Reconstruction (inverse problem)

- Detector design (inverse design)

Task setup for learning control/design

Goal: learn the mapping f_θ from u^t to u^{t+1}:



u^t: original **state** of the system. Can be an infinite-dimensional function $u(t, x)$ as solution to a PDE, or a graph (e.g., mesh, particle-based systems, molecules)

f_θ: neural surrogate models

m^t: external **control（外界控制）**

a: **static parameters（静态参数）** of the system that does not change with time (e.g. parameters of PDE, spatially varying diffusion coefficient)

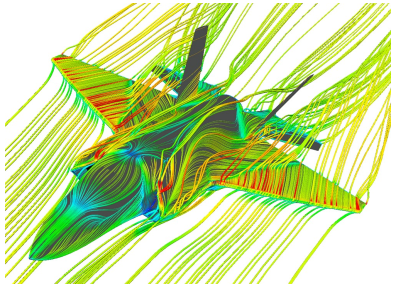
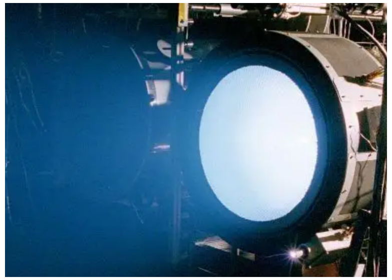
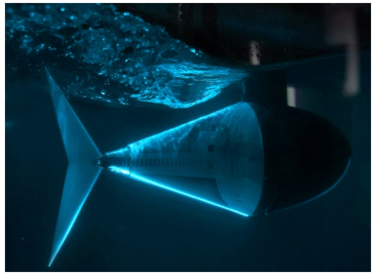
$\partial\mathbb{X}$: **boundary condition（边界条件）** of the system

control（控制）

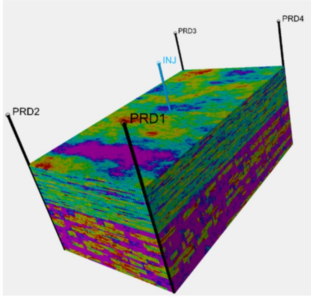
inverse design（反向设计）

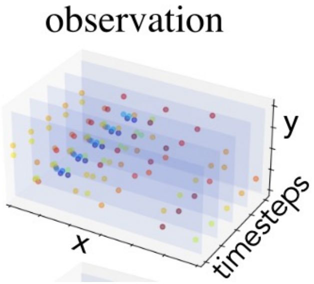
Types of tasks

- **Inverse design:** boundary $\partial\mathbb{X}$, initial state u^0, parameter a to optimize design objective: plane design, rocket shape, underwater robot shape

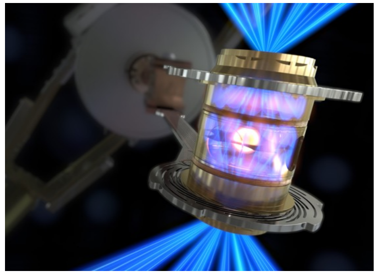
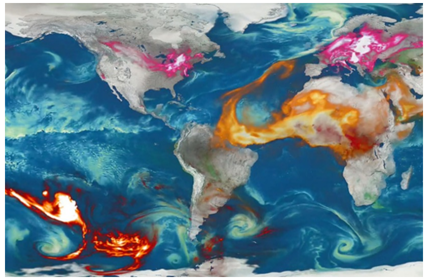




- **Inverse problem: infer** initial state u^0, parameter a to match prediction with data



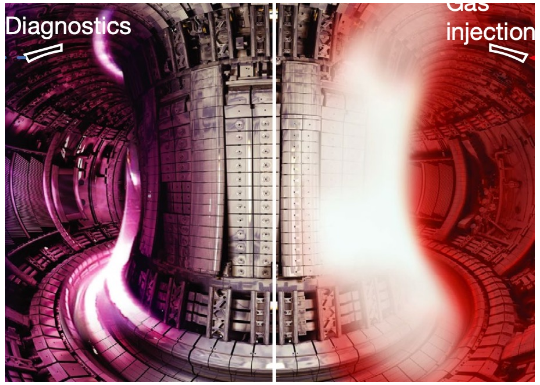


- **Control:** optimize control sequence m^t to optimize control objective**:** control pulses for controlled nuclear fusion

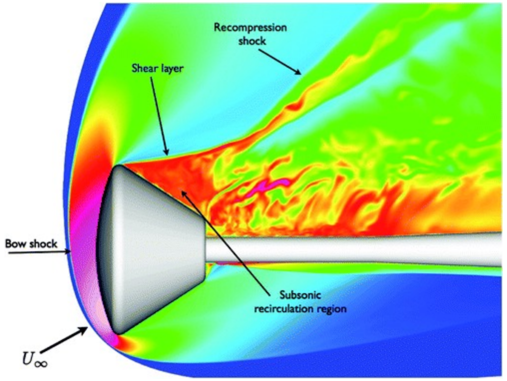



AI for control and inverse design: significance

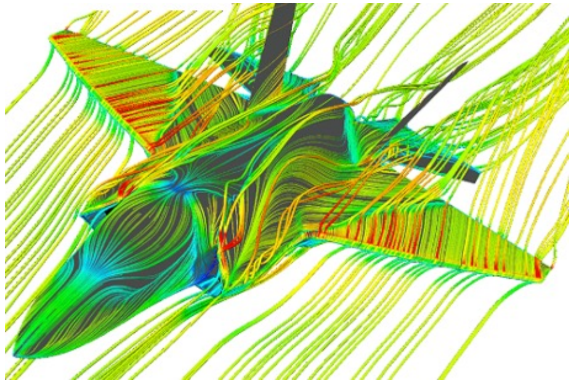
- Control and Inverse design is **prevalent** in science and engineering:



Controlled nuclear fusion



Mechanical engineering



Plane design

- Helps to explore continuous, **high-dimensional** design space, potential to find control/designs not imagined by humans

AI for control and inverse design: difficulty

- **Complex control/design space:**
  - High Computational cost
  - Complex composition relations
- **Complex dynamics**:
  - How to characterize interaction between optimization of shape with physical process
- **Generalization:**
  - How to generalize to more complex compositional scenarios

Limitation of prior methods

1. **Traditional physical simulation methods**

   For **control**: PID is the most prevalent

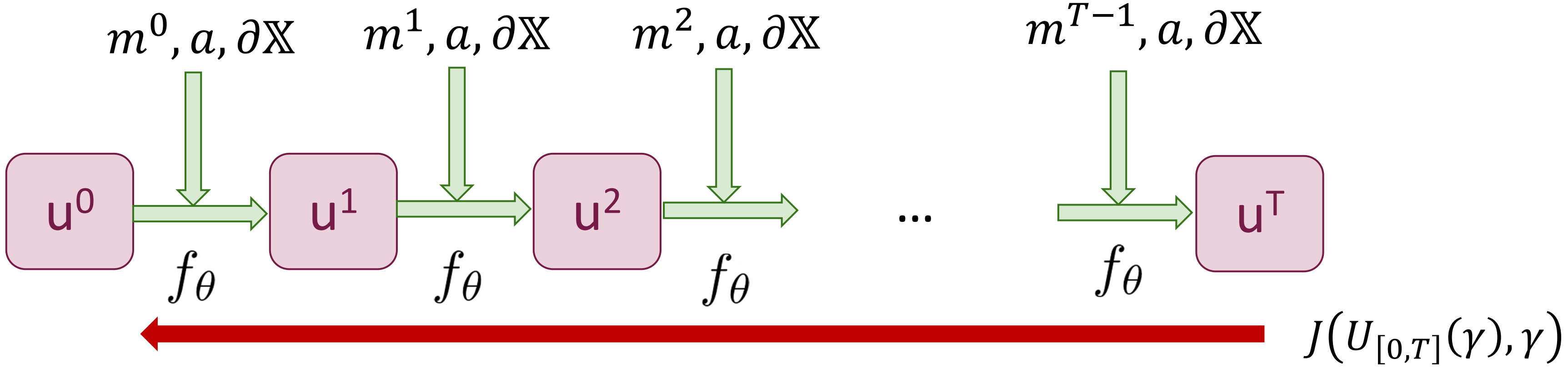
   - Hard to deal with nonlinear systems with high-dimensional, coupled controls
   - Tuning it requires with expertise

   For **design**: e.g., cross-entropy method：

   - High accuracy but low efficiency
   - Need rich expert knowledge
   - Hard to deal with high-dimensional design space

Limitation of prior methods

2. Deep learning-based surrogate models for control or design [1][2]



- First, learn a surrogate forward model that autoregressively predict the dynamics $U_{[0,T]}$ from the parameters $\gamma^t := (m^t, a, \partial\mathbb{X})$

- Then, using the objective $J\left(U_{[0,T]}(\gamma), \gamma\right)$, doing backpropagation（反向传播）and optimize γ

[1] Allen, Kelsey R., et al. "Physical design using differentiable learned simulators." *arXiv preprint arXiv:2202.00728* (2022).
[2] Hwang, Rakhoon, et al. "Solving pde-constrained control problems using operator learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 4. 2022.

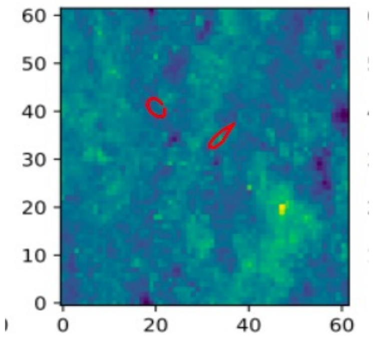
Limitation of prior methods

2. Deep learning-based surrogate models for control or design [1][2]

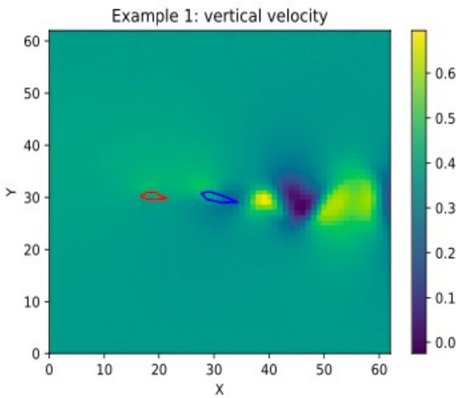
Limitations:

(1) Easy to fall into adversarial modes

Designed boundary shape and fluid velocity



Reasonable boundary shape and fluid velocity

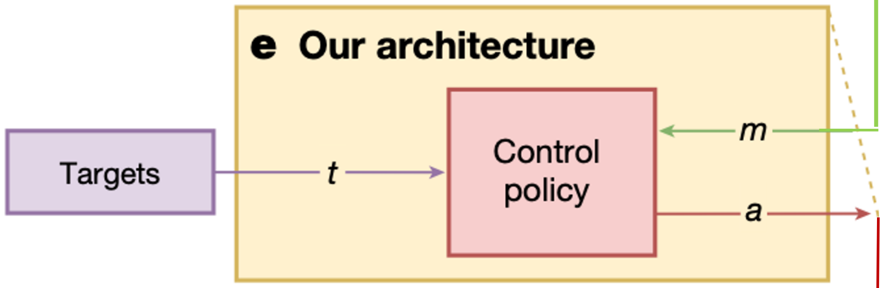


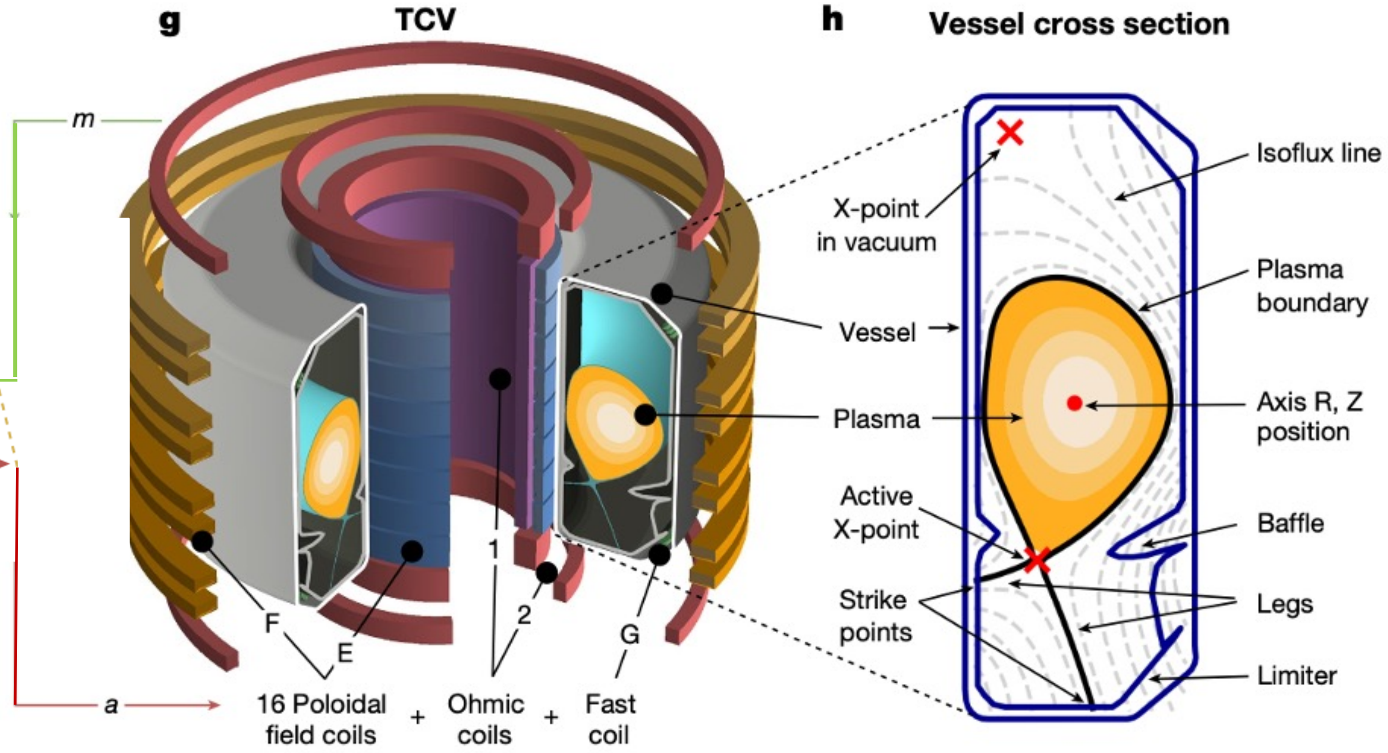
(2) Hard to design more complex parameters

Limitation of prior methods

3. Reinforcement learning for control [1][2]

观测m: input, R^{92}
控制a: output, R^{19}
frequency: 10 kHz





Limitations:

(1) Low sample efficiency: requires large number of interactions
(2) Difficulty for **high-dimensional** control (action space)
(3) Generalization: not flexible to generalize to unseen objectives

[1] Degrave, Jonas, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning." *Nature* 602.7897 (2022): 414-419.
[2] Haarnoja, Tuomas, et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.” ICML 2018.

我们的方法[1]

[1] **Wu, Tailin**, et al. "Compositional Generative Inverse Design." *NeurIPS 2023 AI for Science Workshop*. 2023.

- 将仿真和设计/控制融为同一个任务，通过扩散生成模型同时生成设计和控制变量，以及对应的仿真结果，同时优化目标函数：

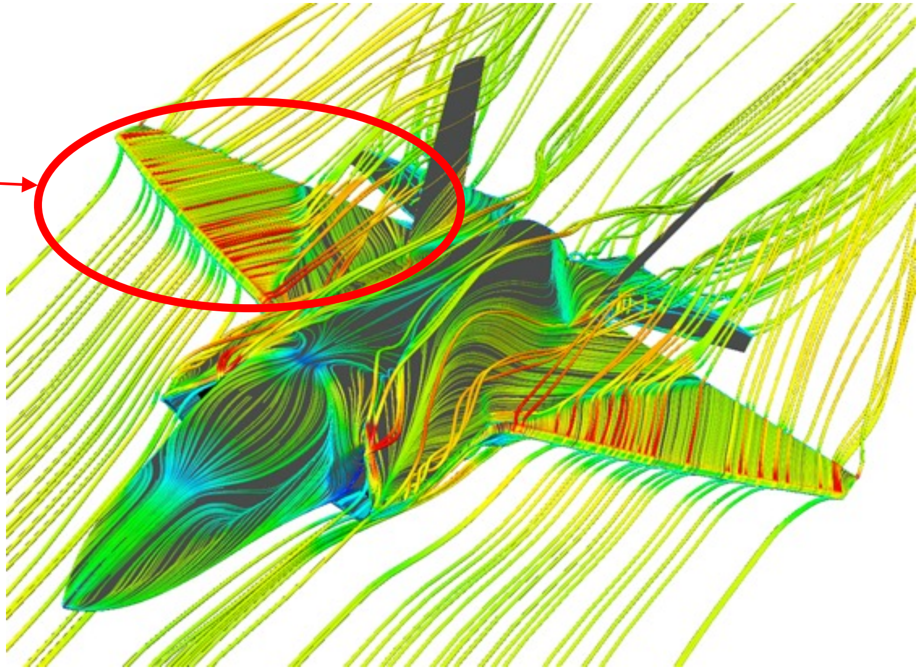
$$\hat{\gamma} = \arg\min_{\gamma, U_{[0,T]}} \left[E_\theta(U_{[0,T]}, \gamma) + \lambda \cdot \mathcal{J}(U_{[0,T]}, \gamma) \right]$$

- 组合逆向设计：能够泛化到更加复杂的设计参数空间

组合逆向设计：任务定义

Given objective $J(U(\gamma),\gamma)$, find design parameters γ that minimize J, where the parameters γ and/or the state U are more complex than in training.

For example:
Training: we only see how the fluid interacts with each part of the airplane



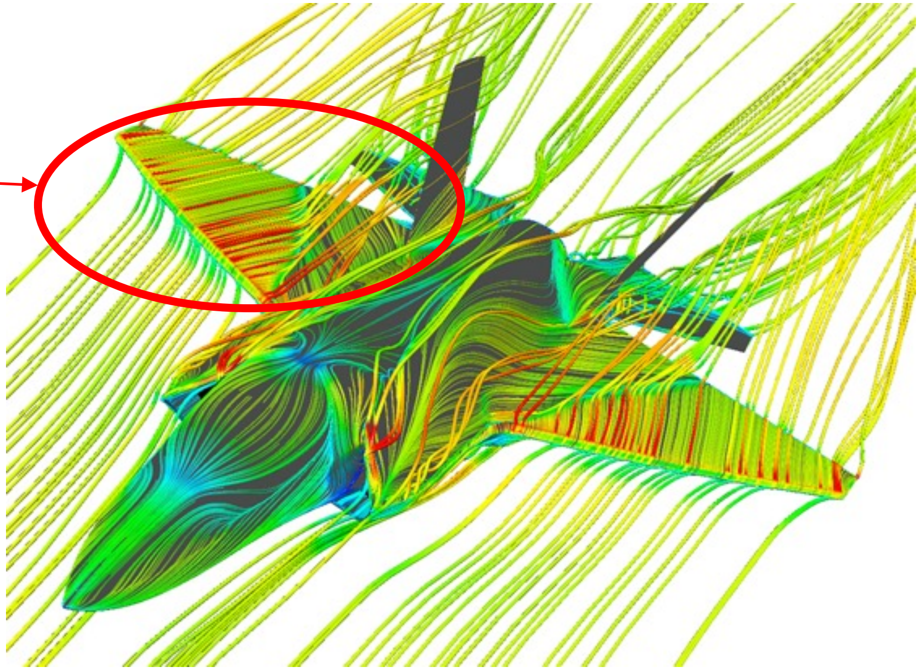
Test: design the whole airplane shape

Compositional inverse design（组合逆向设计）: definition

Given objective $J(U(\gamma),\gamma)$, find design parameters γ that minimize J, where the parameters γ and/or the state U are more complex than in training.

For example:
Training: we only see how the fluid interacts with each part of the airplane

Test: design the whole airplane shape



Key components of our method

[1] **Wu, Tailin**, et al. "Compositional Generative Inverse Design." *NeurIPS 2023 AI for Science Workshop*. 2023.
ICLR: https://openreview.net/forum?id=wmX0CqFSd7

- Simutaneously design the state U and the control/design variable γ
- Joint objective with diffusion models

$$\hat{\gamma} = \underset{\gamma, U_{[0,T]}}{\arg\min} \left[E_\theta(U_{[0,T]}, \gamma) + \lambda \cdot \mathcal{J}(U_{[0,T]}, \gamma) \right]$$

- Compositional

Diffusion models（扩散模型）

本质：给定数据$\{(x_1, c_1), (x_2, c_2), \ldots (x_n, c_n)\}$，其中$x$为一个高维变量，学习一个（条件）概率模型

$$p_\theta(x|c)$$

使得

Diffusion models（扩散模型）

Images and shapes generated by diffusion models:

By DallE 2

By MeshDiffusion [1]

[1] Liu, Zhen, et al. "Meshdiffusion: Score-based generative 3d mesh modeling." *ICLR 2023*

Diffusion models（扩散模型）

Text to video generation by Sora [1]:

[1] OpenAI team. “Video generation models as world simulators”, 2024

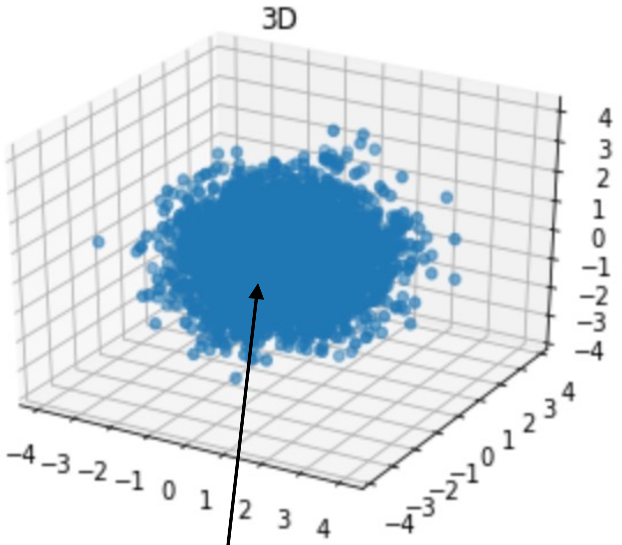
Diffusion models（扩散模型）

Robotic policy by diffusion models [1]

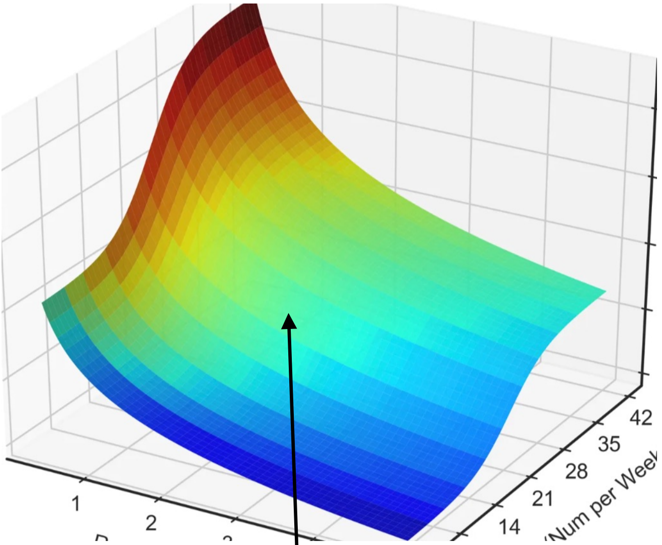
[1] Fu, Zipeng, Tony Z. Zhao, and Chelsea Finn. "Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation." *arXiv preprint arXiv:2401.02117* (2024).

Diffusion models（扩散模型）

Insight: to construct a complex mapping from A to B, it is much easier to compose simple mappings



Diffusion model learns how to go from a random Gaussian sample to manifold

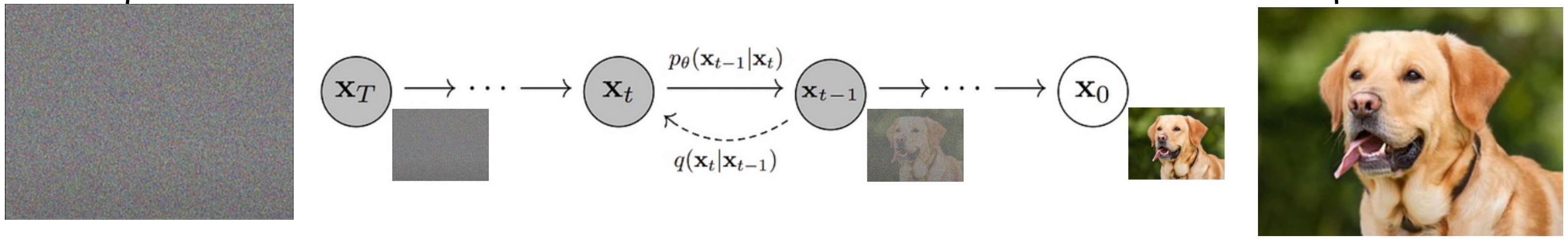


Each data sample is a point in a **manifold** in a high-dimensional space R^d

Gaussian distribution

diffusion model

data distribution



DDPM: denoising diffusion probabilistic models [1]

x_0: 训练数据
x_t: 加了t步高斯噪声的数据
ϵ_θ: 需要学习的去噪网络

训练:

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\underline{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t}) \right\|^2$$
6: **until** converged

输入加噪t步的样本，预测噪声

推理（采样）：

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\underline{\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

一步一步去噪

局限性：未考虑对于目标函数的优化，只能采样出类似训练数据的分布

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in neural information processing systems* 33 (2020): 6840-6851.

Our method: intuition

[1] **Wu, Tailin**, et al. “Compositional Generative Inverse Design.” ICLR 2024 **Spotlight**

x: data samples (e.g., image, trajectory)

$$\hat{\gamma} = \underset{\gamma, U_{[0,T]}}{\arg\min} \left[E_\theta(U_{[0,T]}, \gamma) + \lambda \cdot \mathcal{J}(U_{[0,T]}, \gamma) \right]$$

Diffusion model essentially learns a “energy”-based model E_θ to model the probability distribution

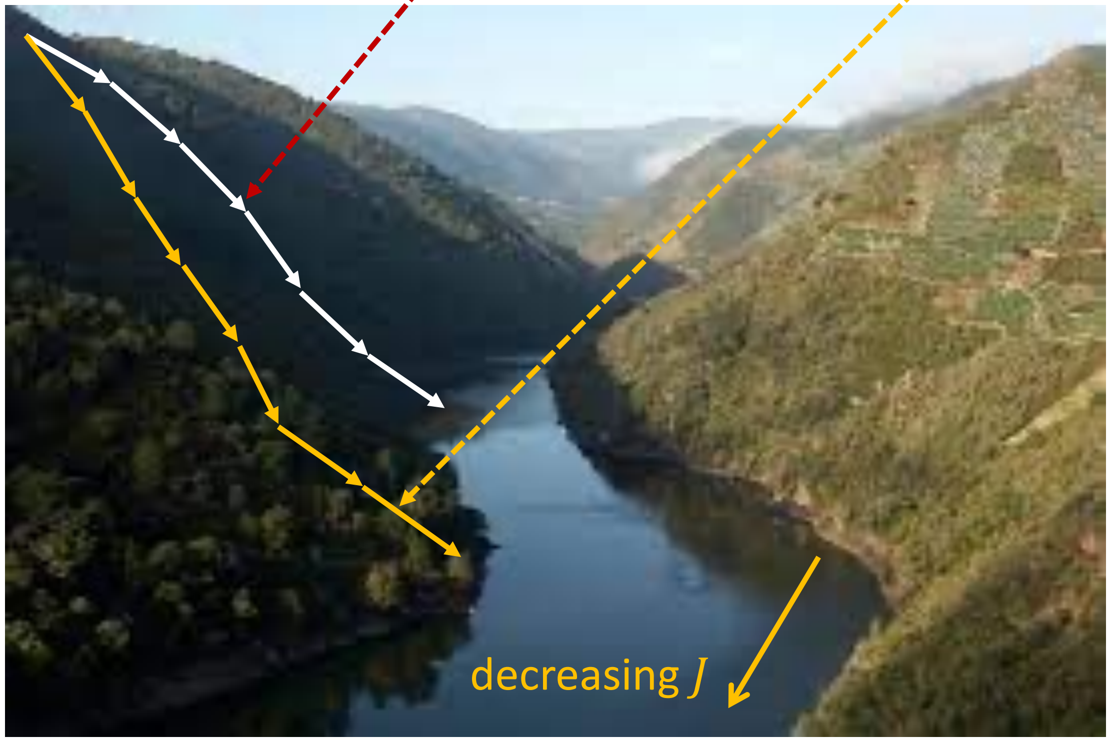
$$p_\theta(x) \propto e^{-E_\theta(x)}$$

The denoising function $\epsilon_\theta(x_t)$ is essentially the gradient of the energy-based model

$$\epsilon_\theta(x) = \nabla_x E_\theta(x)$$

训练时候只训练E_θ

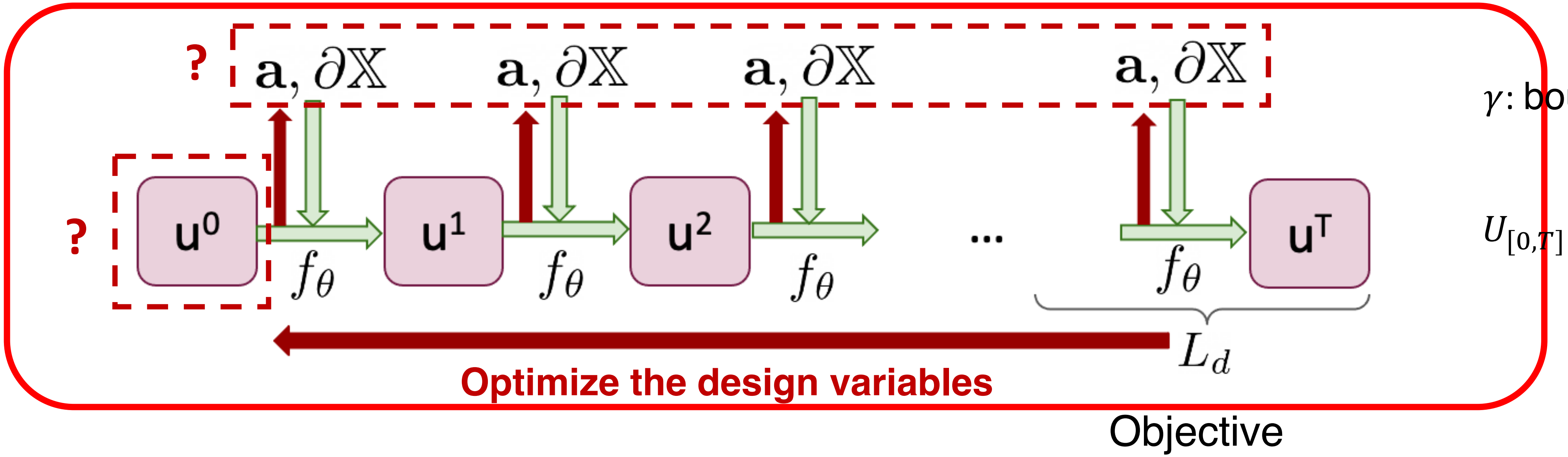
推理时候加上J



$U_{[0,T]}$: state sequence (状态序列)

γ: control variable（控制序列）and boundary condition（边界条件）

我们方法：将仿真和设计融为同一任务，联合生成



γ: boundary condition

$U_{[0,T]}$: state sequence

1.训练时学习能量函数 $E_\theta(U_{[0,T]},\gamma)$ 后，在设计时可同时生成系统的状态轨迹 $U_{[0,T]}$ 和设计参数 γ

2. 设计时，将能量函数 E_θ进行相加，同时加上设计目标 J:

$$\hat{\gamma} = \operatorname{argmin}_{\gamma,U_{[0,T]}}\left(\sum_{k=1}^{M} E_\theta(U_{[0,T]},\gamma_k) + \lambda \cdot J(U_{[0,T]},\gamma)\right)$$

Our method: architecture

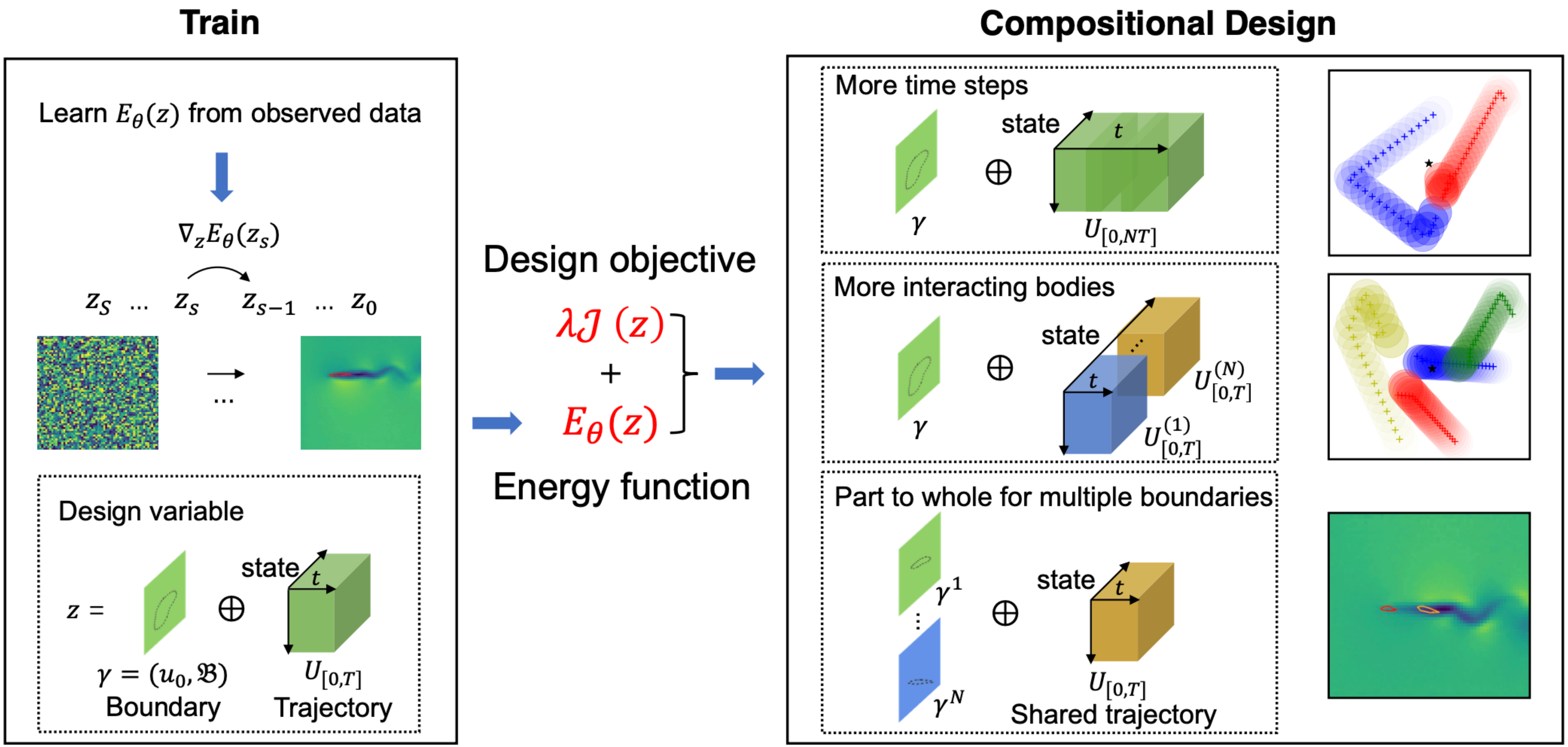


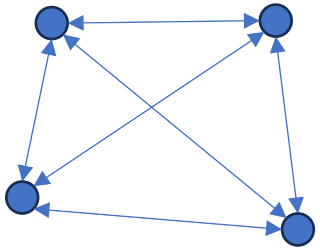
Fig.1 of CinDM. By composing generative models specified over subsets of inputs, we present an approach that design materials significantly more complex than those seen at training.

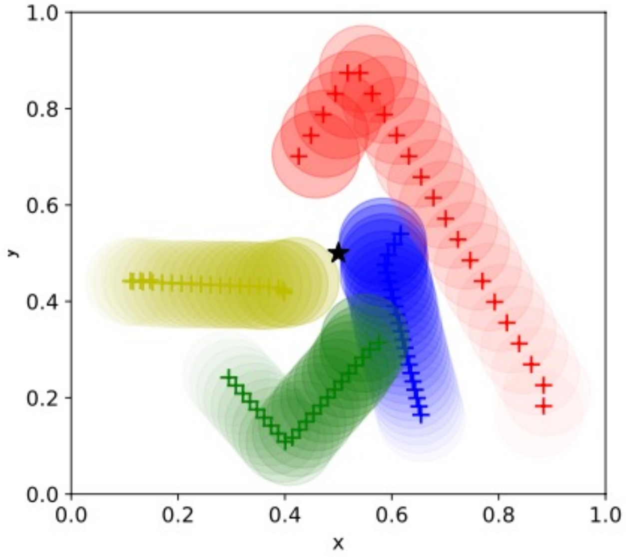
Experiment 1: n-body simulation, state composition

Train: generate the simulation on 2 bodies

Inference: simulation on **more bodies** (e.g., 4 or 8 bodies)

Objective J: design the initial position and velocity of the n bodies such that their end position close to the center





(b) 4-body 44 steps

treat the n-body interaction as composition of multiple 2-body interactions

Experiment 1: n-body simulation, state composition

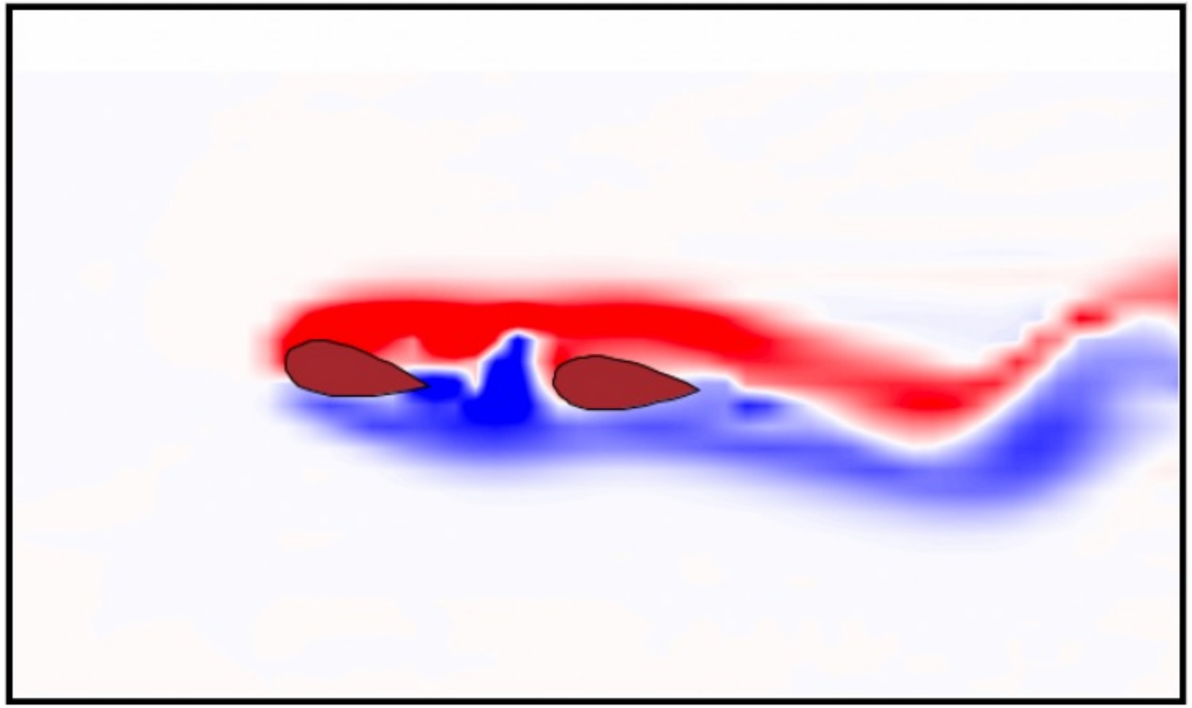
Method	4-body 24 steps design obj	4-body 24 steps MAE	4-body 44 steps design obj	4-body 44 steps MAE	8-body 24 steps design obj	8-body 24 steps MAE	8-body 44 steps design obj	8-body 44 steps MAE
CEM, GNS (1-step)	0.3173	0.23293	0.3307	0.53521	0.3323	0.38632	0.3306	0.53839
CEM, GNS	0.3314	0.25325	0.3313	0.28375	0.3314	0.25325	0.3313	0.28375
Backprop, GNS (1-step)	0.2947	0.06008	0.2933	0.30416	0.3280	0.46541	0.3317	0.72814
Backprop, GNS	0.3221	0.09871	0.3195	0.15745	0.3251	0.15917	0.3299	0.21489
CinDM (ours)	**0.2034**	**0.03928**	**0.2254**	**0.03163**	**0.3062**	**0.09241**	**0.3212**	**0.09249**

Our method (CinDM) achieves the best design objective with lowest simulation MAE

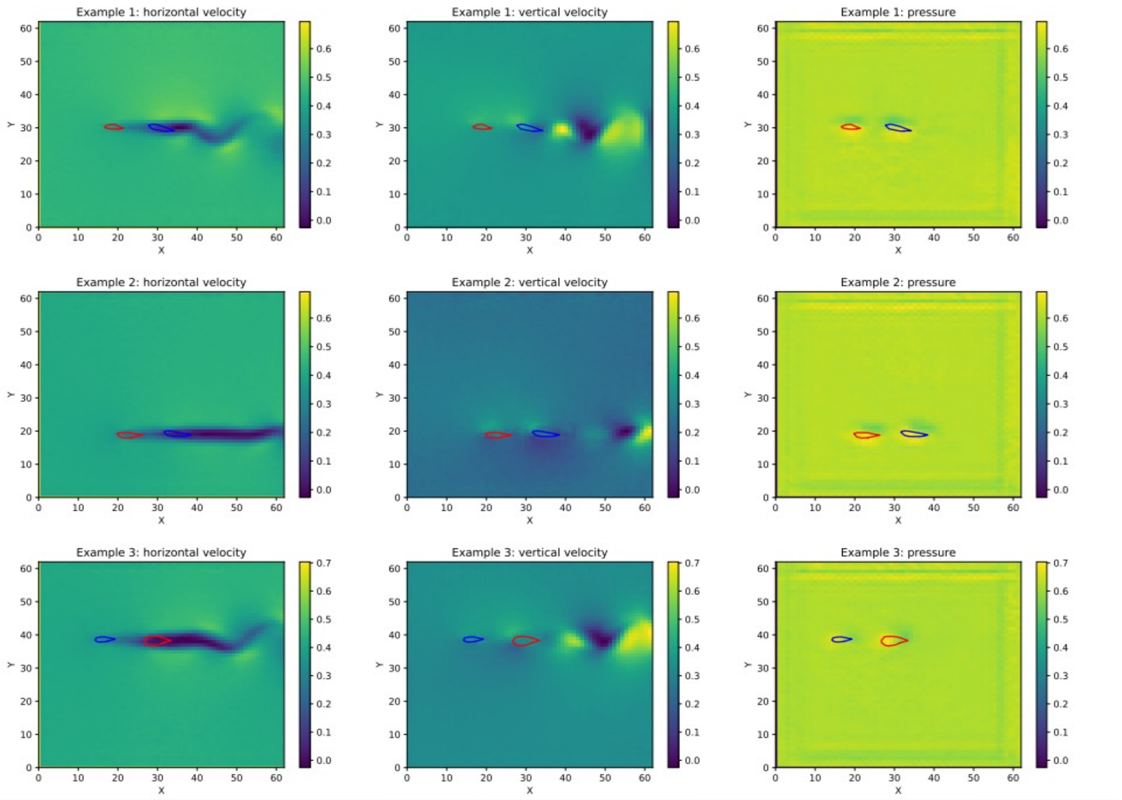
Experiment 2: airfoil design, part-to-whole composition

Training: consider a **single airfoil** interacting with the air flow

Inference: consider **multiple airfoils**, maximize life-to-drag ratio: $\left(=\frac{lift}{drag}\right)$



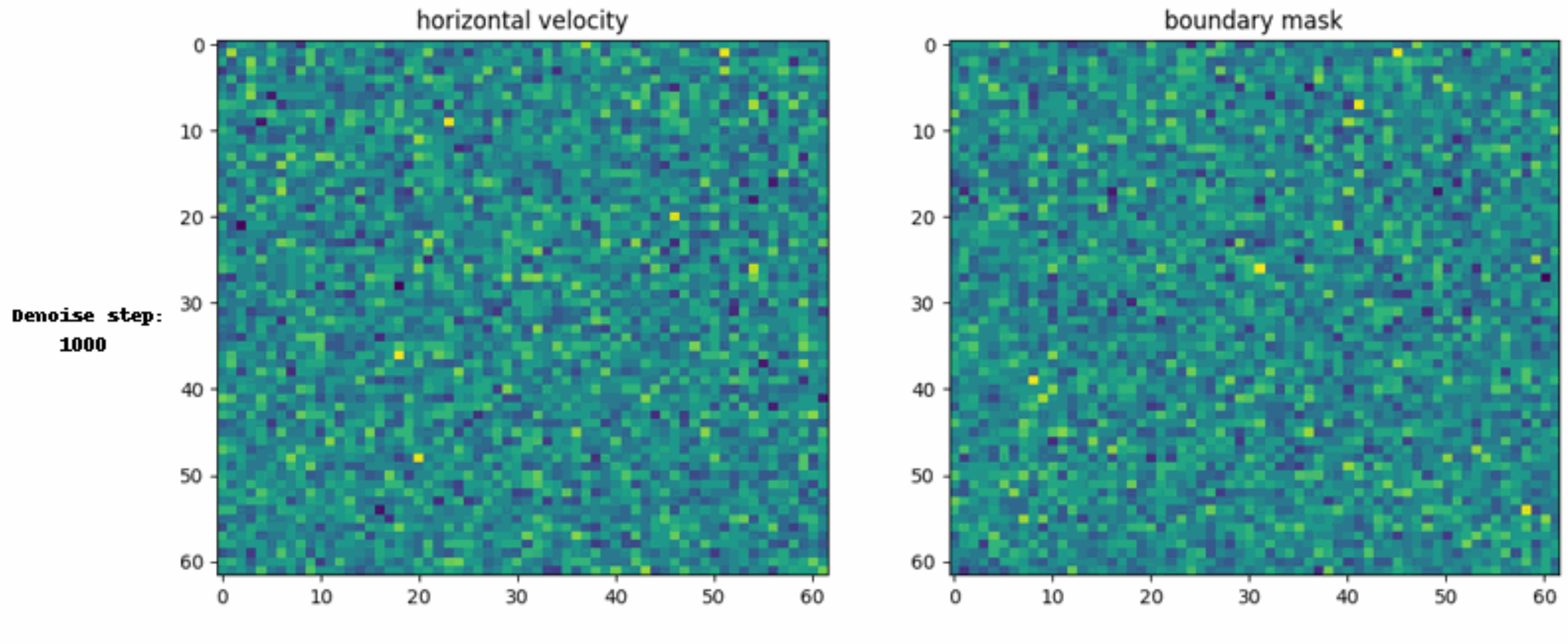
Example of Lily-Pad simulation



Compositional design results of our method in 2D airfoil generation. Each row represents an example. We show the heatmap of velocity in horizontal and vertical direction and pressure in the initial time step, inside which we plot the generated airfoil boundaries.

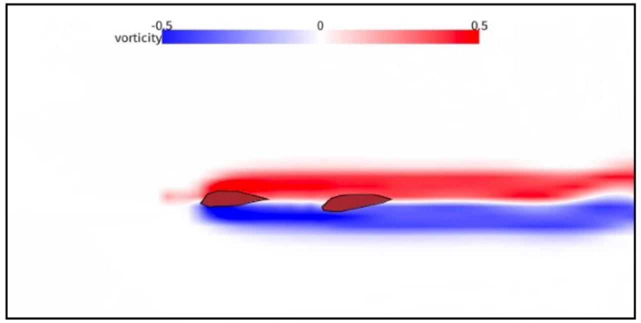
Experiment 2: airfoil design, part-to-whole composition

推理生成过程：从高斯分布到设计出来的速度场轨迹和机翼边界设计

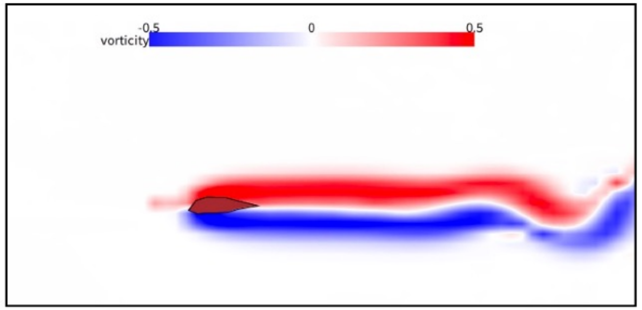


流体速度场　　　　机翼边界设计

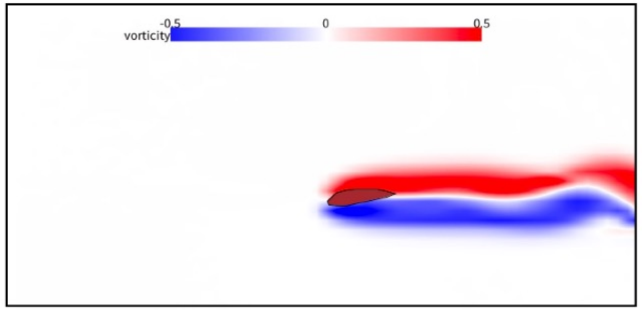
Experiment 2: airfoil design, part-to-whole composition



(a) Formation flying of airfoils A and B



(b) Single flying of airfoil A



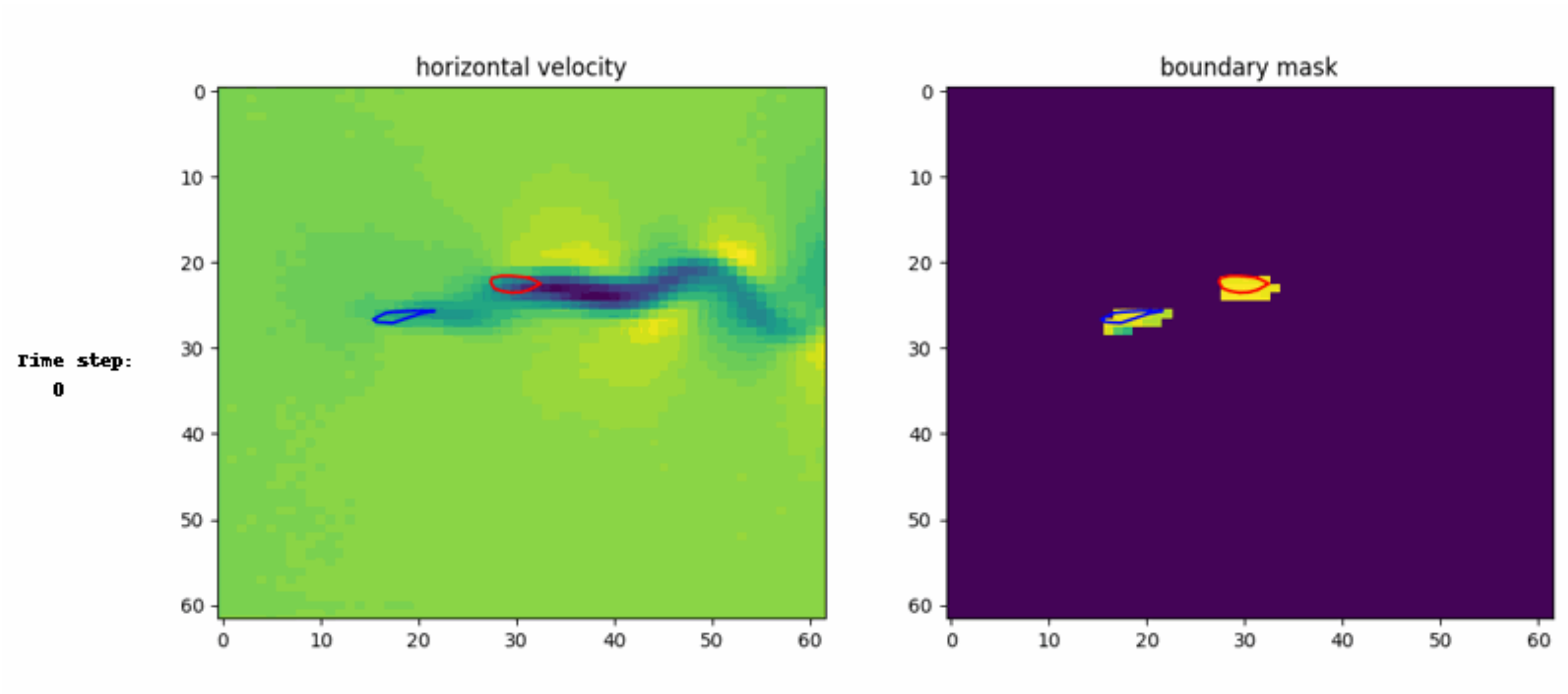
(c) Single flying of airfoil B

Our model discovers **formation flying**

- Reducing the drag by 53.6%
- increasing the lift-to-drag ratio by 66.1%

Experiment 2: airfoil design, part-to-whole composition

生成的速度场轨迹和机翼边界设计：

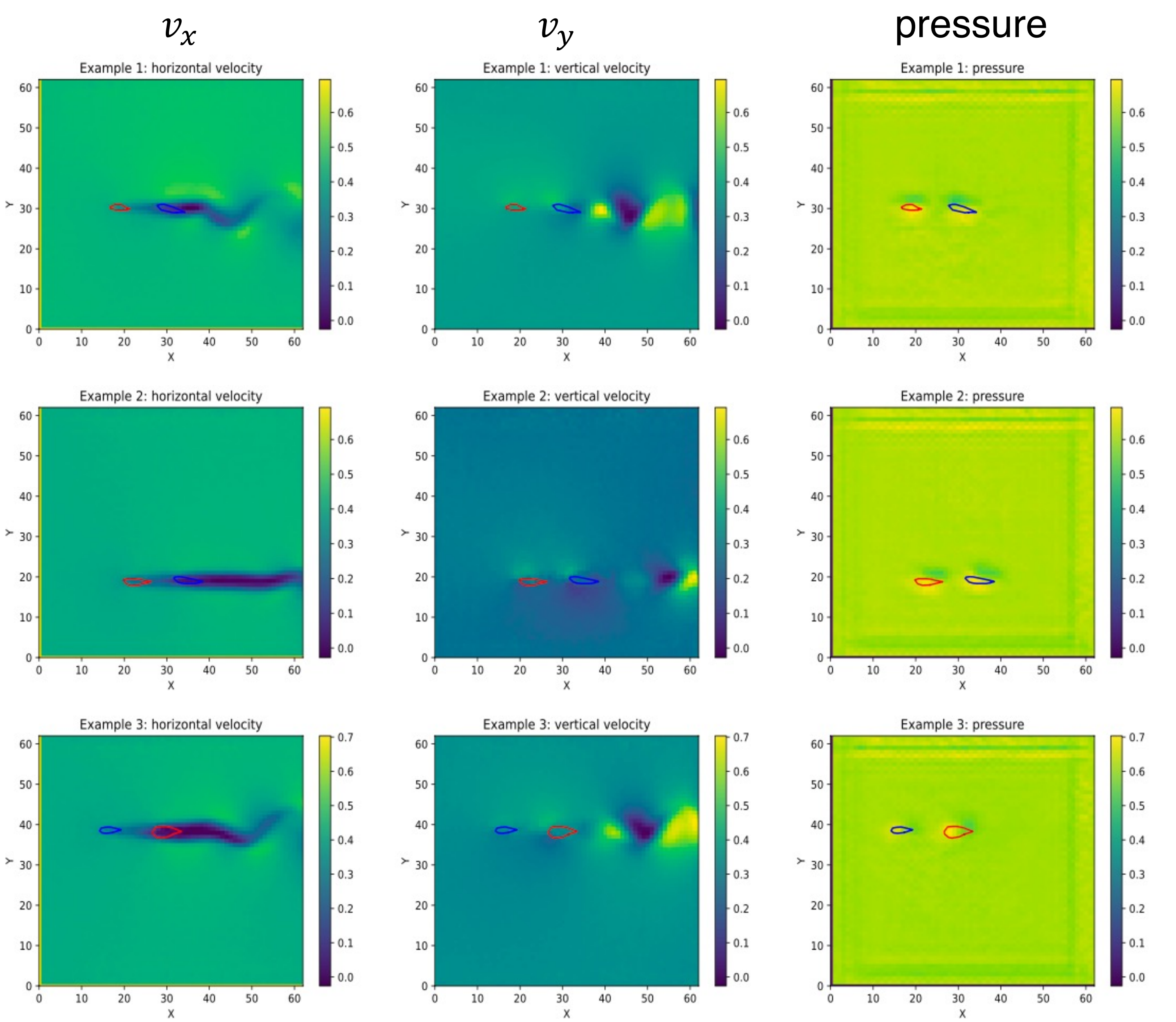


Experiment 2: airfoil design, part-to-whole composition

Method	1 airfoil		2 airfoils	
	design obj ↓	lift-to-drag ratio ↑	design obj ↓	lift-to-drag ratio ↑
CEM, FNO	0.0932	1.4005	0.3890	1.0914
CEM, LE-PDE	0.0794	1.4340	0.1691	1.0568
Backprop, FNO	**0.0281**	1.3300	0.1837	0.9722
Backprop, LE-PDE	0.1072	1.3203	**0.0891**	0.9866
CinDM (ours)	0.0797	**2.1770**	0.1986	**1.4216**

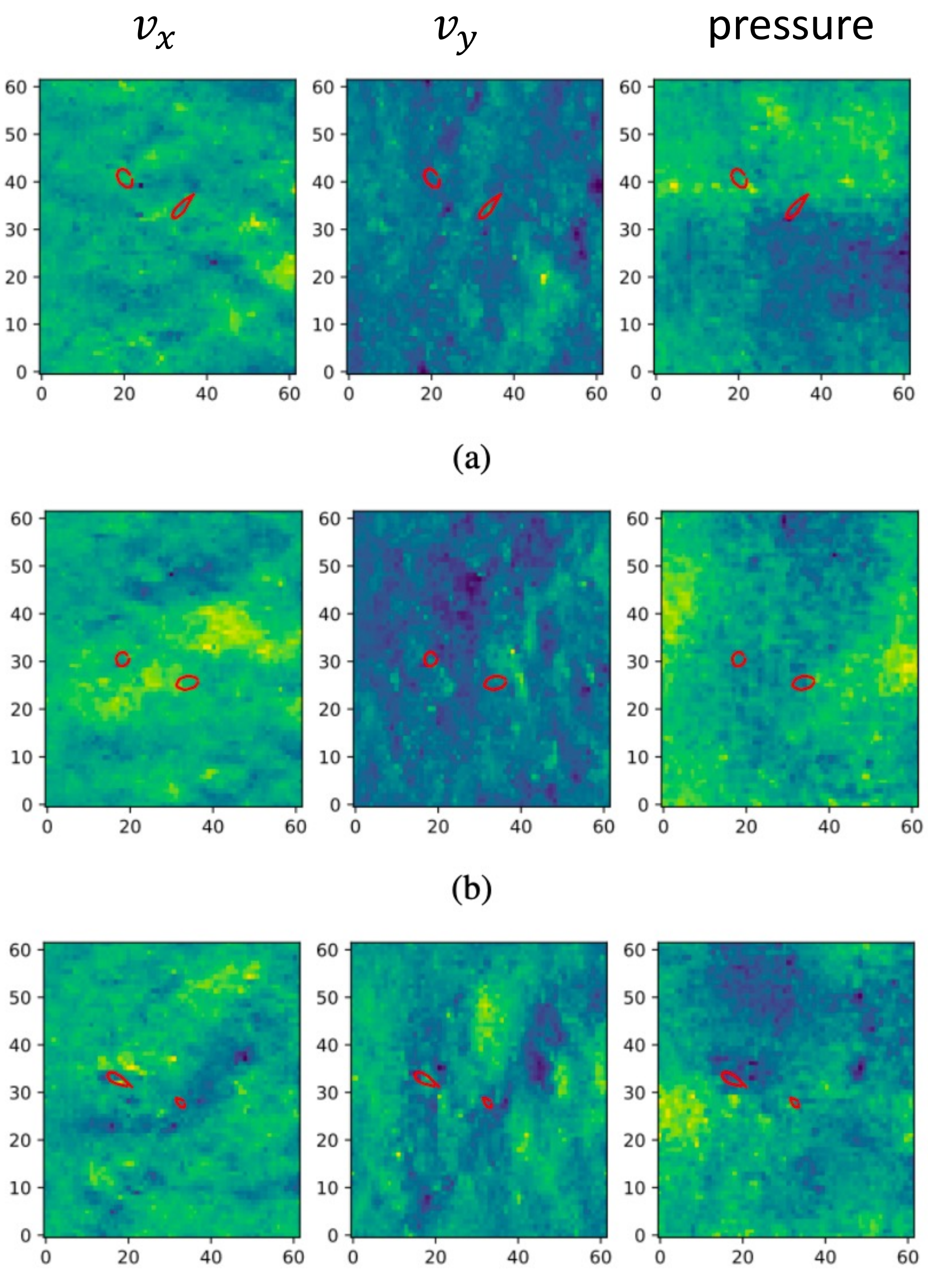
Experiment 2: airfoil design, part-to-whole composition

CinDM generated:

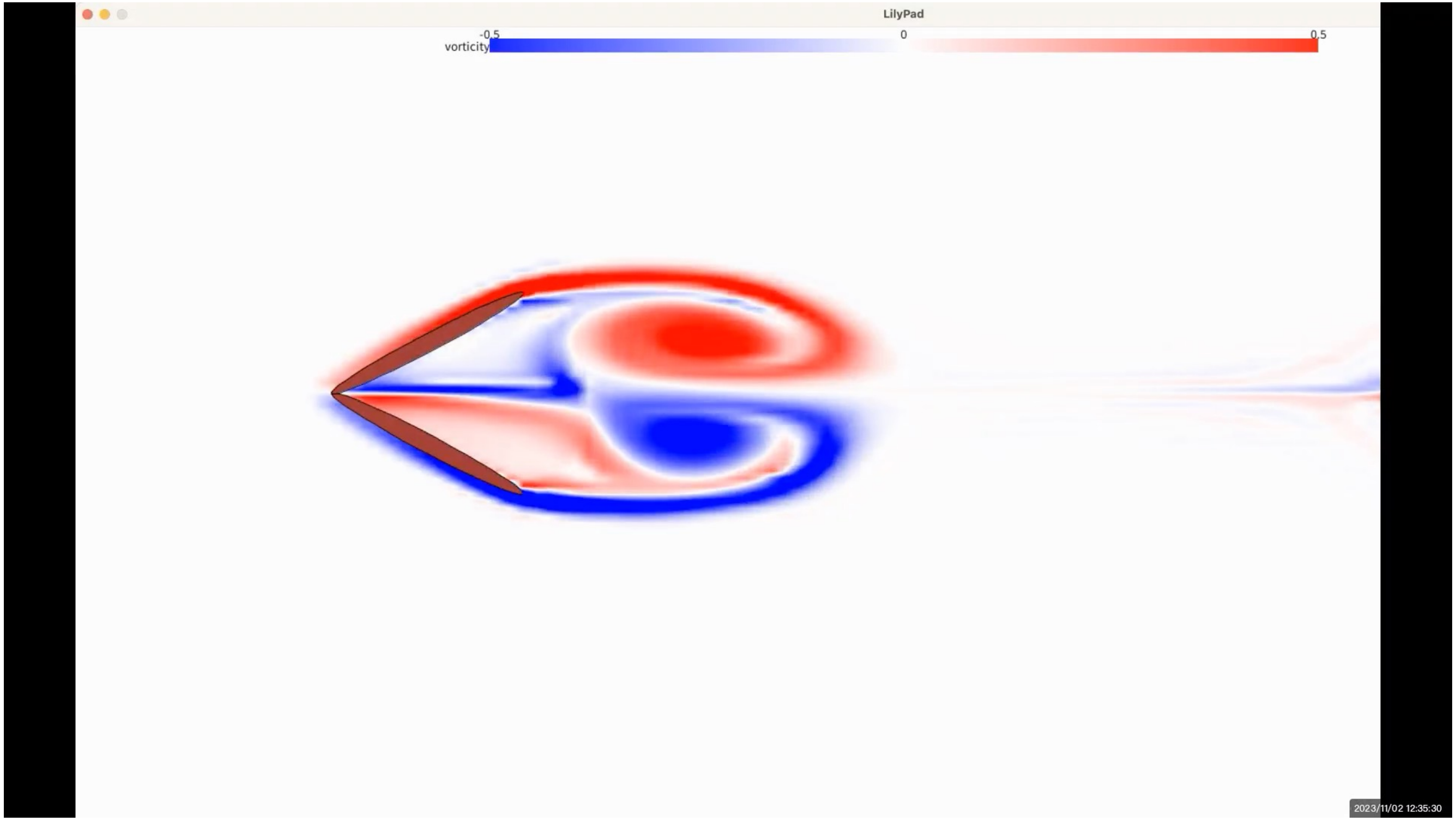


Experiment 2: inverse design by baseline neural models

Adversarial modes by baseline neural model (FNO with CEM):

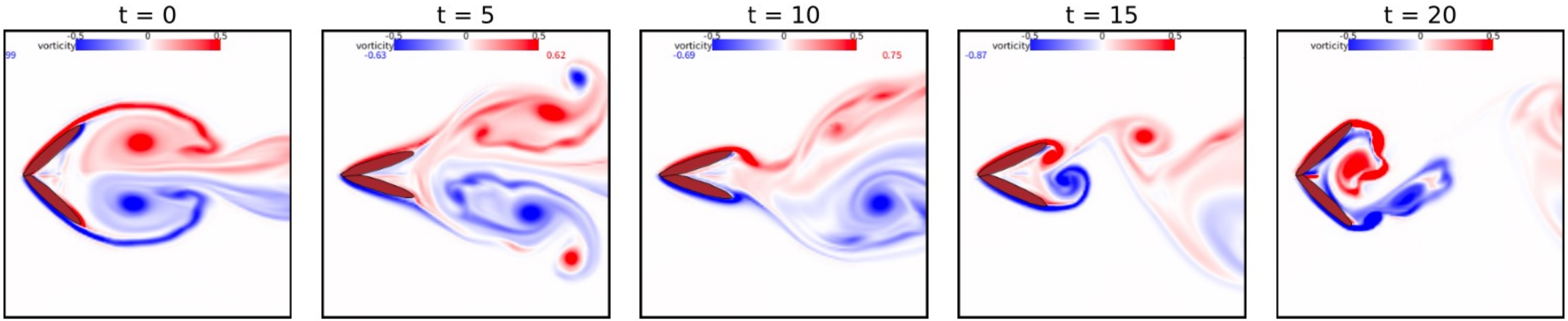


Experiment 3: Controlling a jellyfish



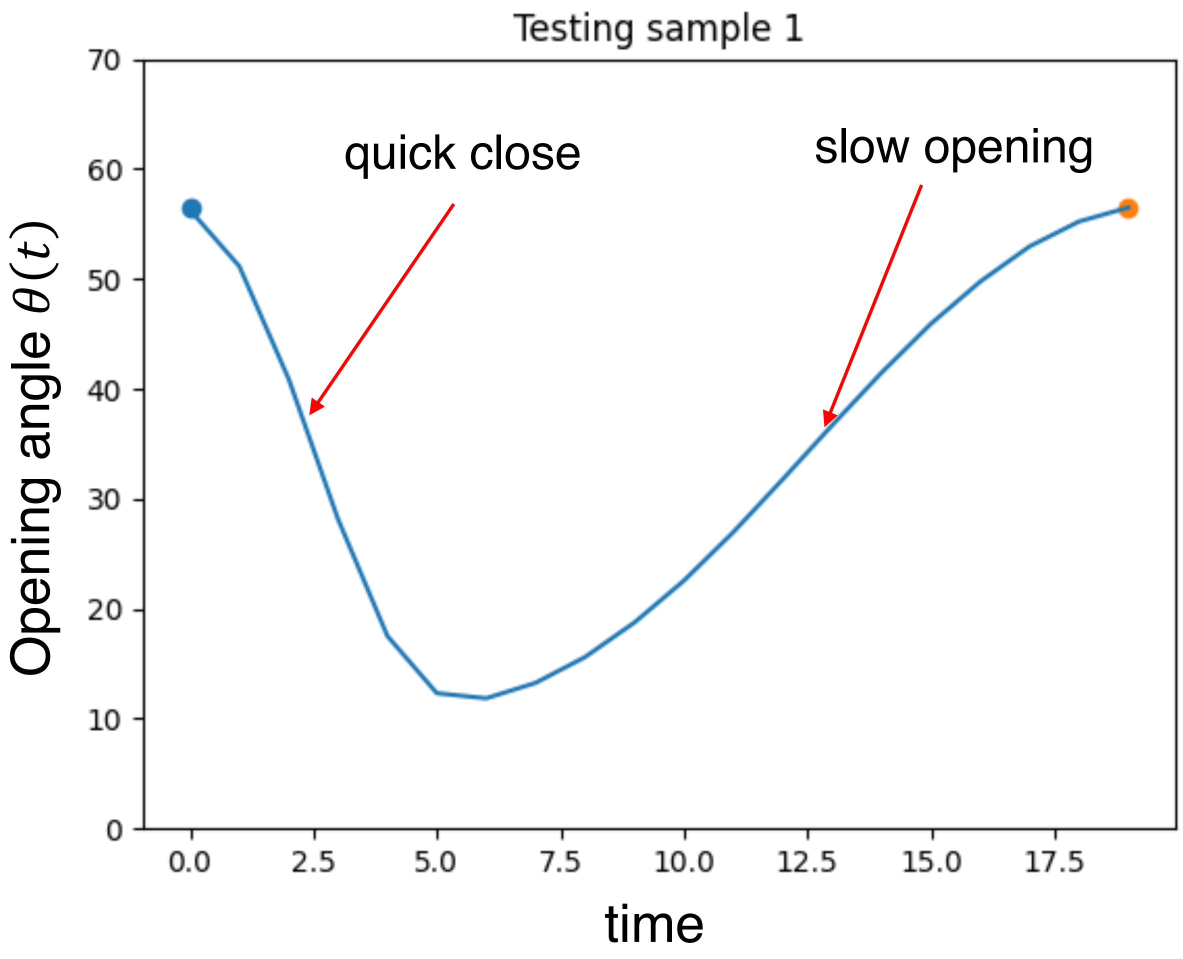
Experiment 3: Controlling a jellyfish

Trajectory controlled by our model:



Experiment 3: Controlling a jellyfish

Discovered $\theta(t)$ by our diffusion model



Paper and code

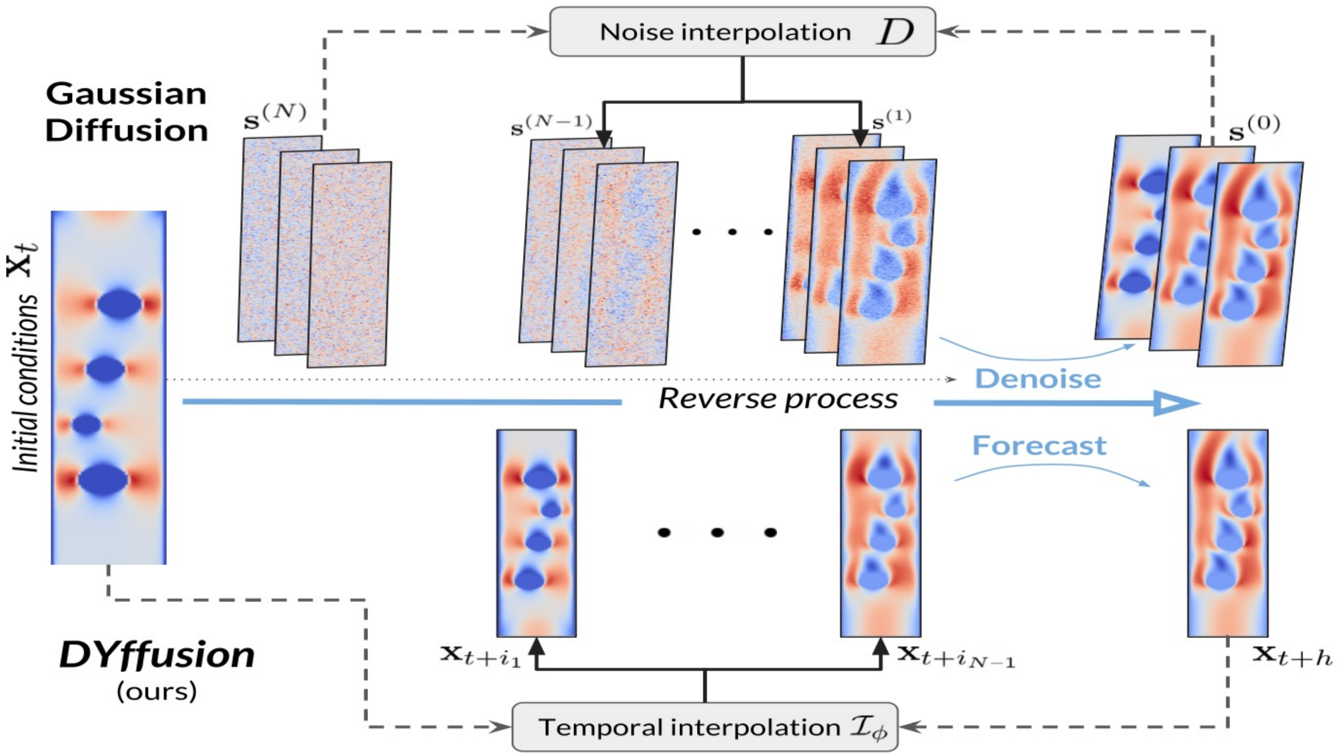
Paper: Compositional Generative Inverse Design
ICLR 2024 spotlight: https://openreview.net/forum?id=wmX0CqFSd7

Code: https://github.com/AI4Science-WestlakeU/cindm

Other important application of Diffusion Models

For simulation（仿真）：

Learn $P\left(U^{[1,T]}|U^{0}\right)$:



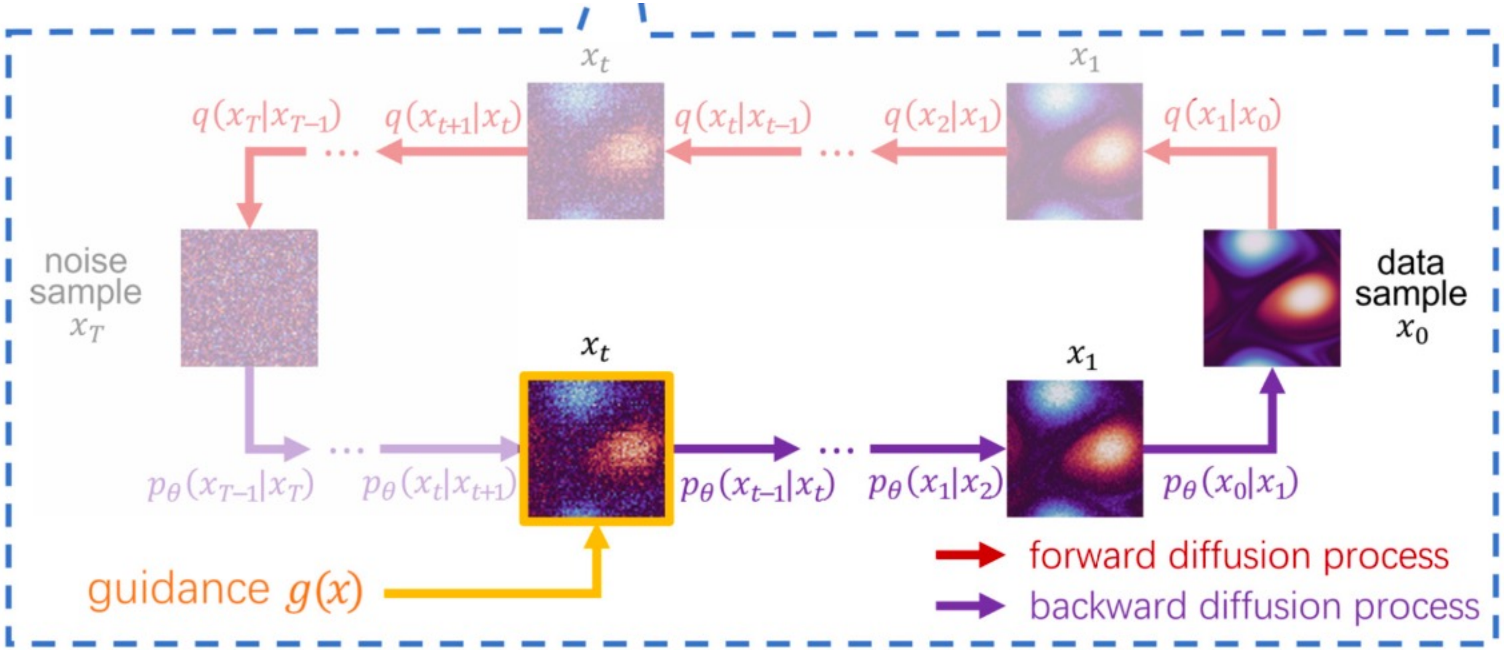
[1] Cachay, Salva Rühling, et al. "DYffusion: A Dynamics-informed Diffusion Model for Spatiotemporal Forecasting." NeurIPS 2023, *arXiv preprint arXiv:2306.01984* (2023).

Other important application of Diffusion Models

Physics-informed diffusion models（物理信息+扩散模型）：

$$\hat{\gamma} = \arg\min_{\gamma, U_{[0,T]}} \left[E_\theta(U_{[0,T]}, \gamma) + \lambda \cdot \mathcal{J}(U_{[0,T]}, \gamma) \right]$$

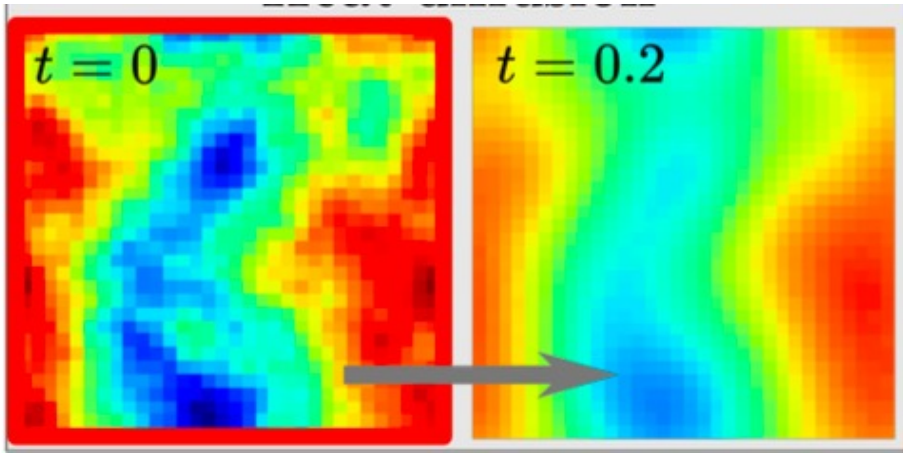
Key idea: using the PDE residual as additional term for the objective J.



[1] Shu, Dule, Zijie Li, and Amir Barati Farimani. "A physics-informed diffusion model for high-fidelity flow field reconstruction." *Journal of Computational Physics* 478 (2023): 111972.

Other important application of Diffusion Models

For inverse problems（逆问题）：



Holzschuh, Benjamin, Simona Vegetti, and Nils Thuerey. "Solving Inverse Physics Problems with Score Matching." *NeurIPS* 2023

我们组进一步研究方向

开发通用AI方法用于AI4Science核心、普适问题：

- 基于扩散生成模型、大模型的多物理、多尺度、复杂装置的仿真、控制和设计
- 安全性和可信度：使得模型在全新物和物理条件下，给出可证明的不确定性区间和误差保证
- 更好结合数据与物理先验

与关键领域合作

- 流体
- 飞机机翼设计
- 能源
- 材料

期待合作，共同解决重要的问题！

联系方式: wutailin@westlake.edu.cn，主页: http://tailin.org

AI for Scientific Simulation & Discovery Lab

西湖大学人工智能与
科学仿真发现实验室

仿真	设计
控制	发现

↓

流体	航空航天
能源	材料

总结

任务:
- 仿真
- 逆向设计系统参数

方法:

1. 图神经网络用于系统动力学模拟 [1]

$$L = (1-\beta)\cdot \mathrm{Error} + \beta \cdot \mathrm{Computation}$$

2. 神经算子用于函数空间映射
3. 扩散模型用于逆向设计 [2]

$$\hat{\gamma} = \underset{\gamma, U_{[0,T]}}{\arg\min} \left[E_\theta(U_{[0,T]}, \gamma) + \lambda \cdot \mathcal{J}(U_{[0,T]}, \gamma)\right]$$

- 训练时学习能量函数 $E_\theta\left(U_{[0,T]}, \gamma\right)$ 后，在设计和控制时时可同时生成系统的状态轨迹 $U_{[0,T]}$ 和设计参数/控制序列 γ
- 能量相加用于组合设计

Welcome collaborations and tackle important problems together!
Contact: wutailin@westlake.edu.cn. Homepage: http://tailin.org

Paper [1]:

Paper [2]: