CS839: AI for Scientific Computing
Physics-Informed Neural Networks

Misha Khodak

University of Wisconsin-Madison

5 February 2026

Outline

- **Upcoming classes: guest lectures and student presentations**
- **Introduction to PINNs**
- **Challenges of PINNs**
- **Recap of neural PDE solvers**

Outline

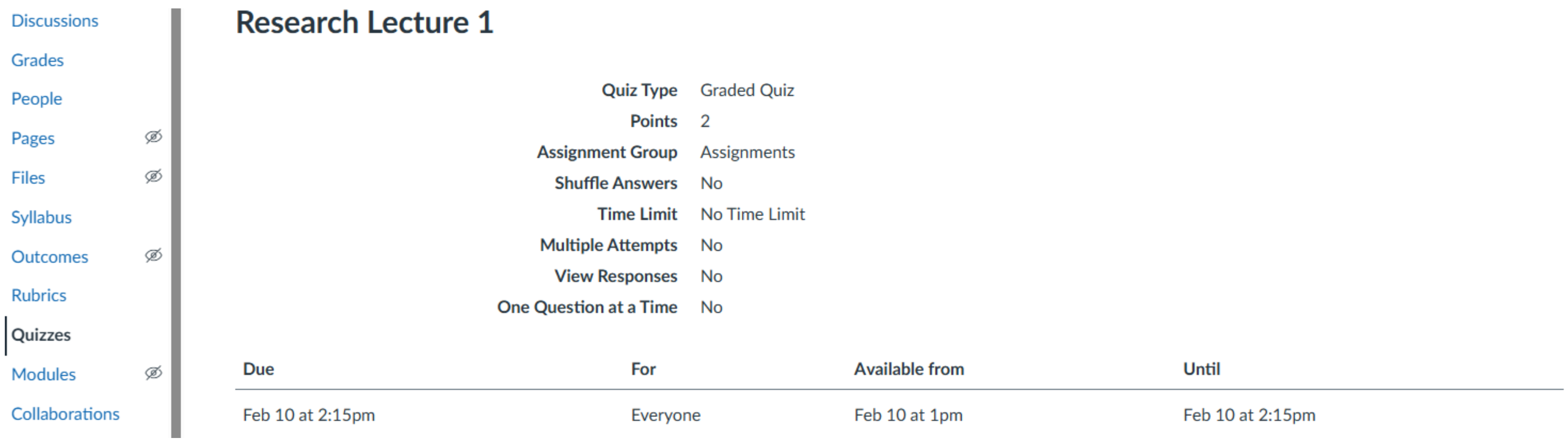
- **Upcoming classes: guest lectures and student presentations**
- **Introduction to PINNs**
- **Challenges of PINNs**
- **Recap of neural PDE solvers**

Upcoming classes: Research lectures

- 10 Feb – Mariel Pettee: *Invisible Cities: Imagining the next era of AI-enabled fundamental physics research* [abstract online]
- 17 Feb – Qin Li: control in kinetic equations
- 19 Feb – Misha Khodak: learned preconditioners
- 24 Feb – Rogerio Jorge: ML for plasma physics
- 26 Feb – Xuhui Huang + Zige Liu: ML for computational chemistry
- 5 March – Wenxiao Pan: data-driven simulation of complex fluids

Upcoming classes: Participation

- policy outlined on website
- roughly: submit two question during each of half the lectures
- encouraged but not required to ask questions during the talk

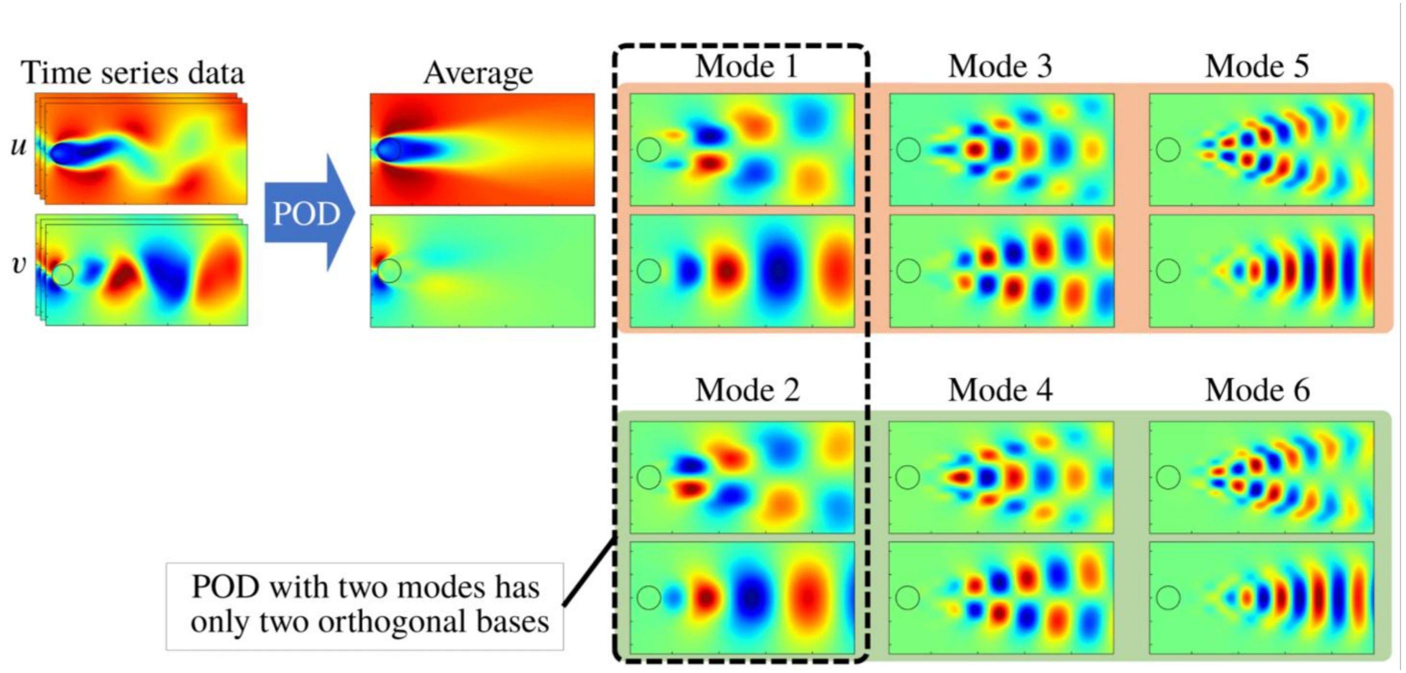


Upcoming classes: Student presentations

- first presentation March 3rd
- 2-3 people per presentation
- will send out sign-up sheet soon, but start thinking about what you might like to present
- standard approaches:
  - deep dive into a single paper
  - overview of an area via several papers

Presentation ideas: Reduced-order models

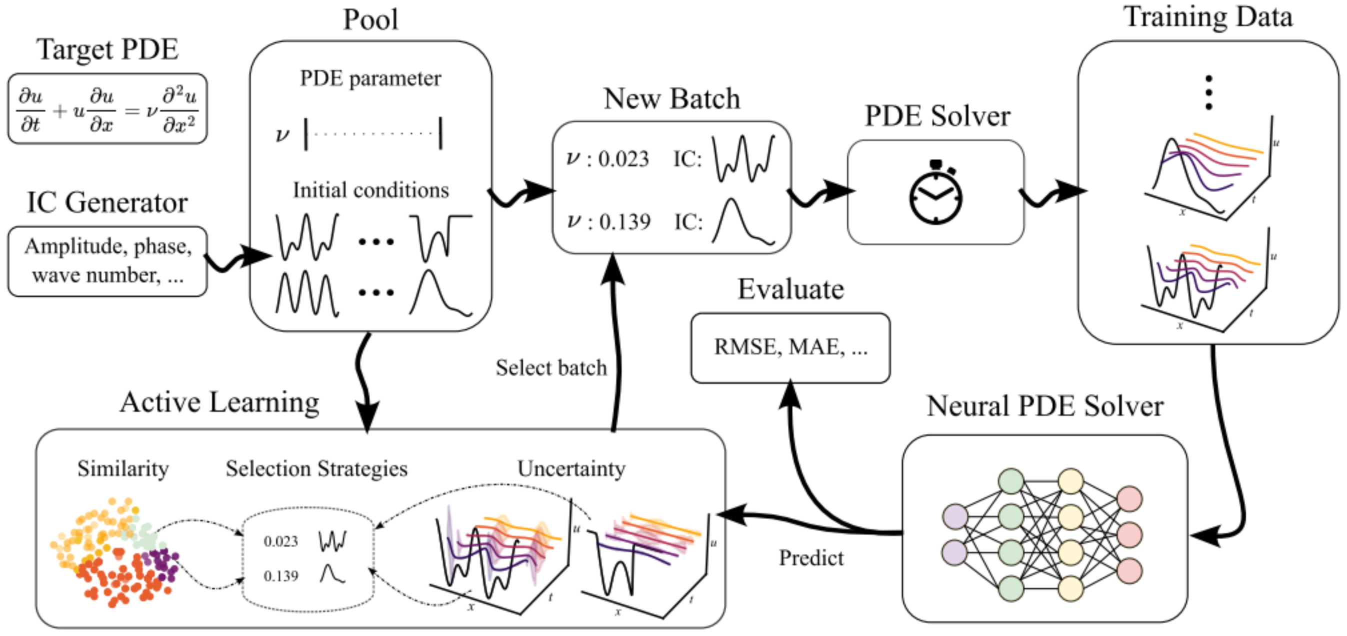
- dynamic mode decomposition
- closure modeling



[Murata-Fukami-Fukagata, 2019]

Presentation ideas: Data-efficiency

- active learning
- foundation models



[Musekamp et al., 2025]

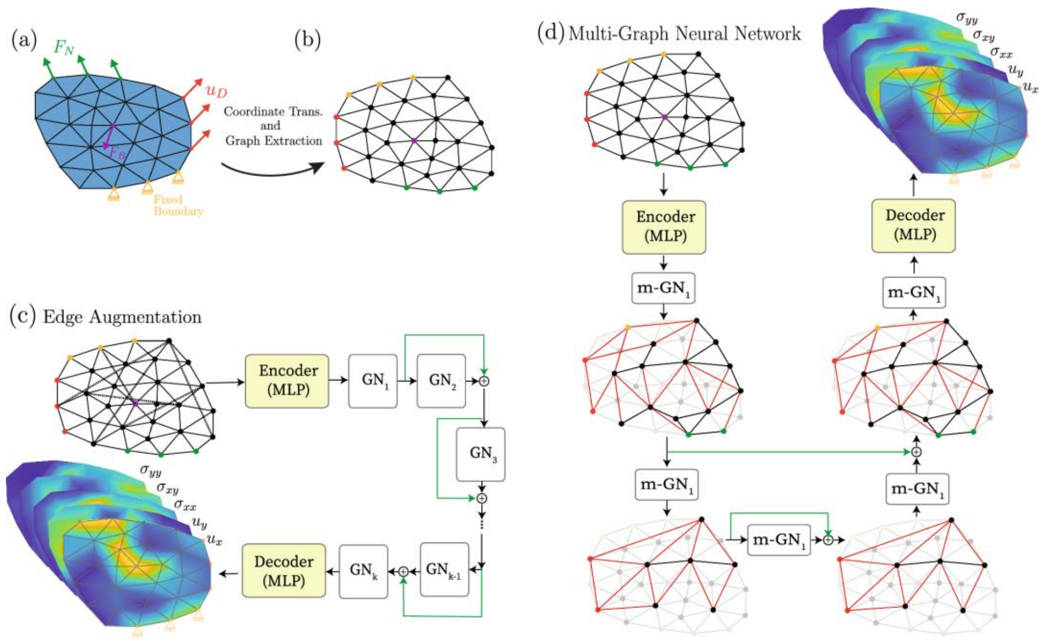
Presentation ideas: learning-augmented (scientific computing) algorithms

- neural preconditioners
- neural multigrid
- meta-learned optimizers
- meta-learned optimal transport

[Amos et al., 2022]

Presentation ideas: Advanced architectures

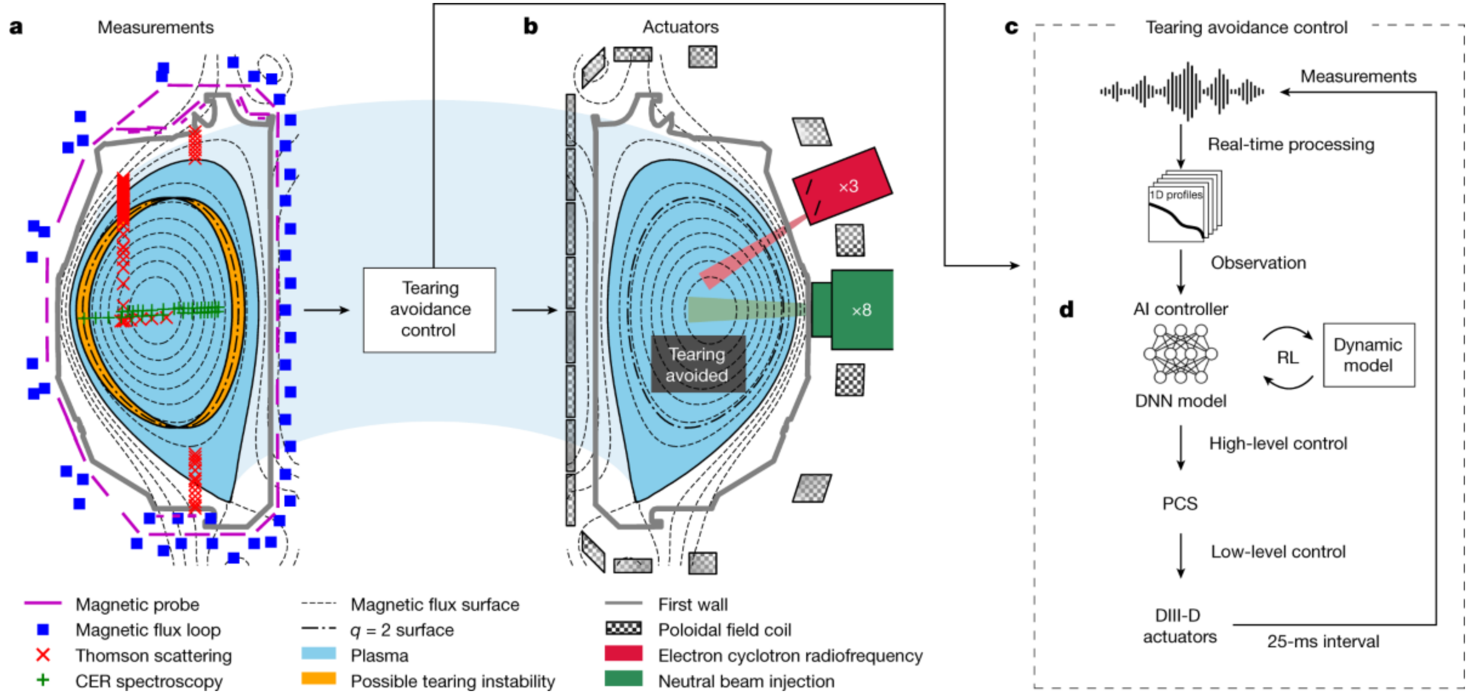
- neural ODEs for PDEs
- GNNs for PDEs
- Transformers for PDEs
- geometry-adaptive architectures
- hybrid (neural and classical) solvers



[Gladstone et al., 2024]

Presentation ideas: Applications

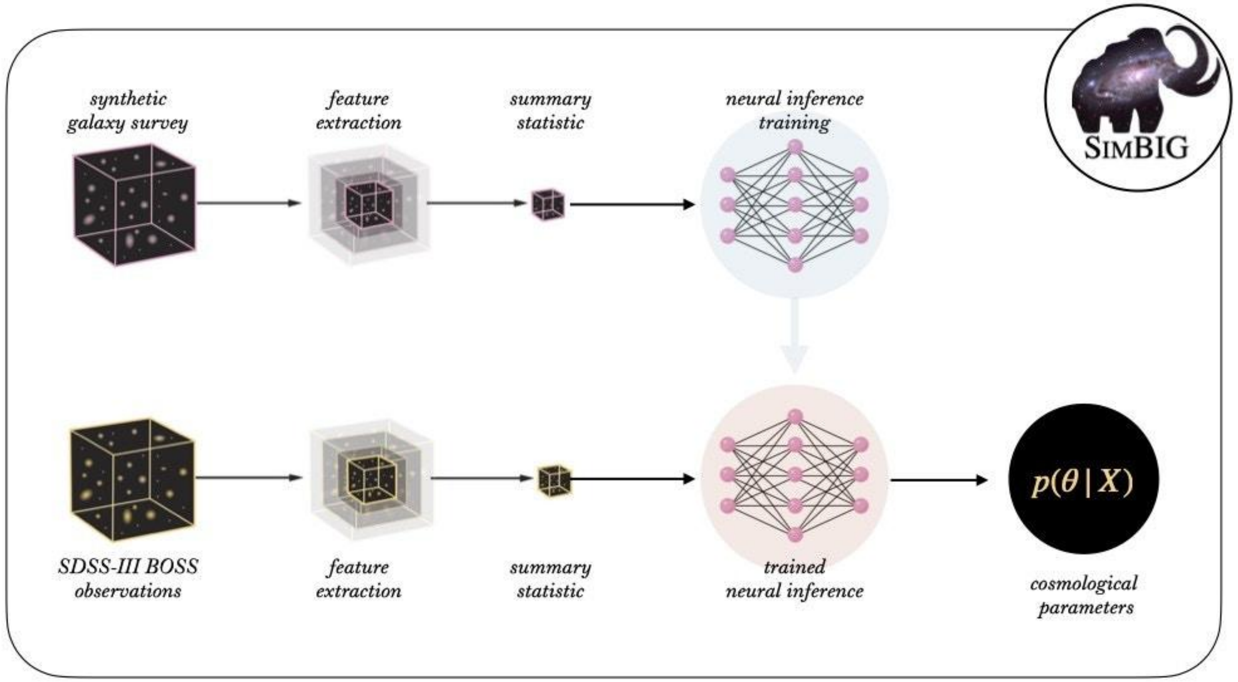
- molecular dynamics
- drug discovery
- materials science
- plasma control
- …



[Seo et al., 2024]

Presentation ideas: Other topics

- theory for neural operators
- uncertainty quantification
- simulation-based inference



[Ho & Parker]

Outline

- Upcoming classes: guest lectures and student presentations
- **Introduction to PINNs**
- Challenges of PINNs
- Recap of neural PDE solvers

What Are PINNs? – Core Concept

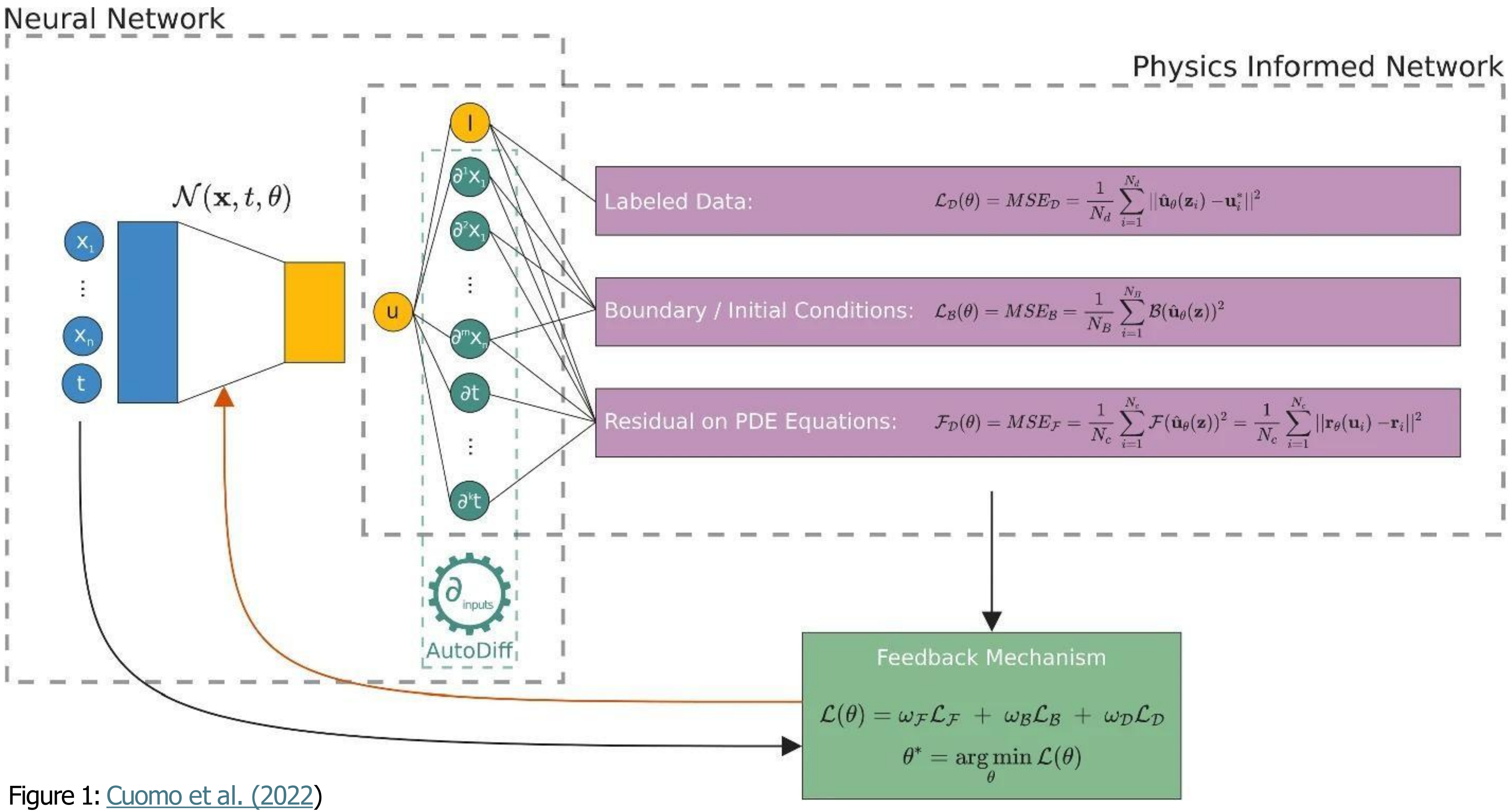
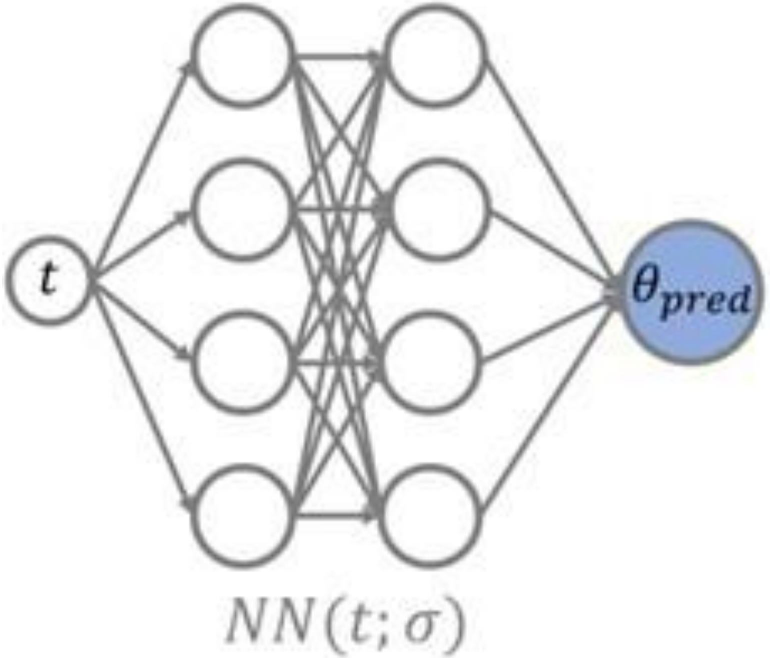


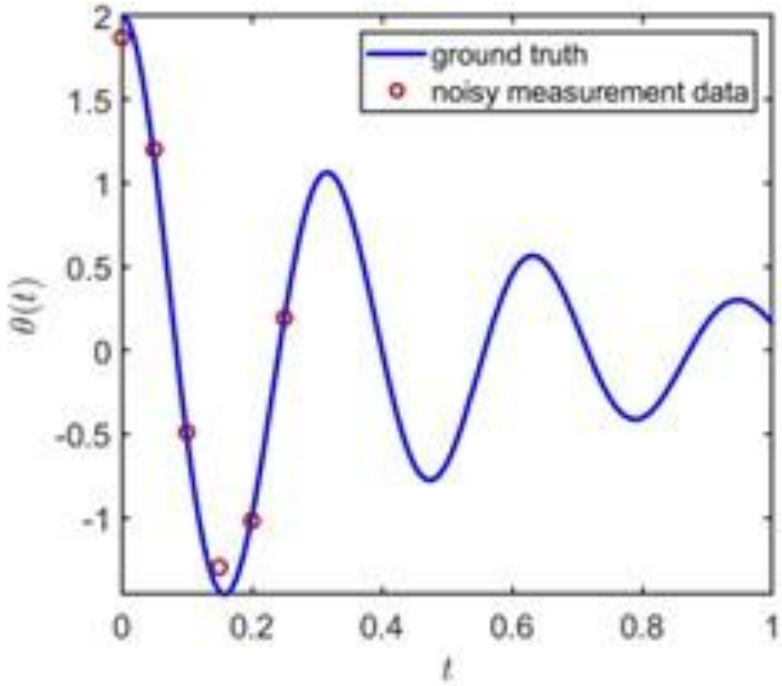
Figure 1: Cuomo et al. (2022)

How PINNs Are Trained

- Define NN: inputs (coords/params), outputs (field(s)).
- Form composite loss: $L = L_D + L_F + L_B$.
- Sample collocation points; compute residuals with AutoDiff.
- Optimize (Adam → L-BFGS); monitor residuals/BCs
- Enforce BCs softly (penalty) or hard (by design); validate.

Example: Pendulum - ML





$$\min_{\sigma} \frac{1}{N} \sum_{i=1}^{N} \left| \theta_{pred}(t_i; \sigma) - \theta_{meas}(t_i) \right|^2$$

Figure 2: MathWorks: PINNs

Example: a damped pendulum - ML solution

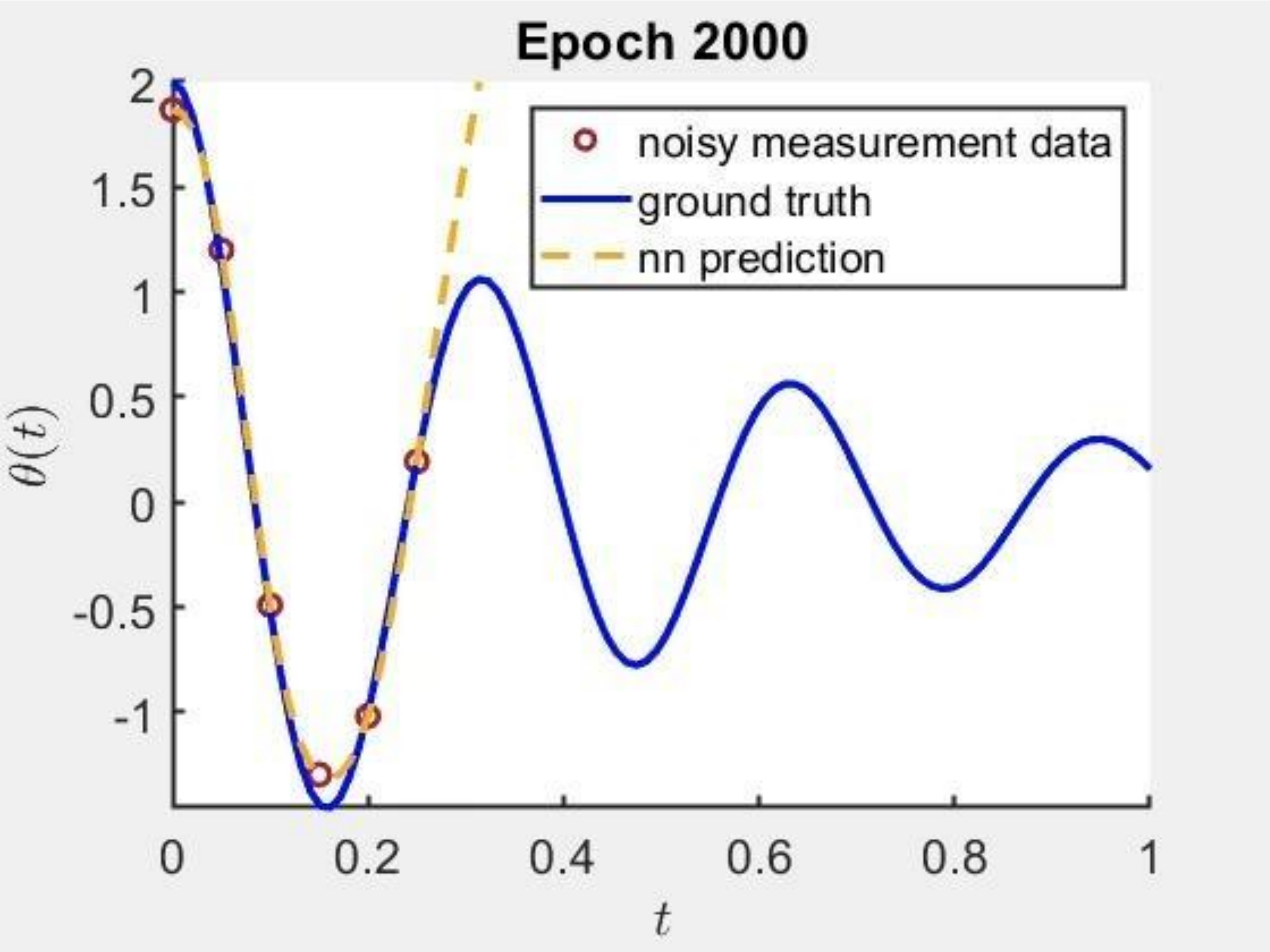
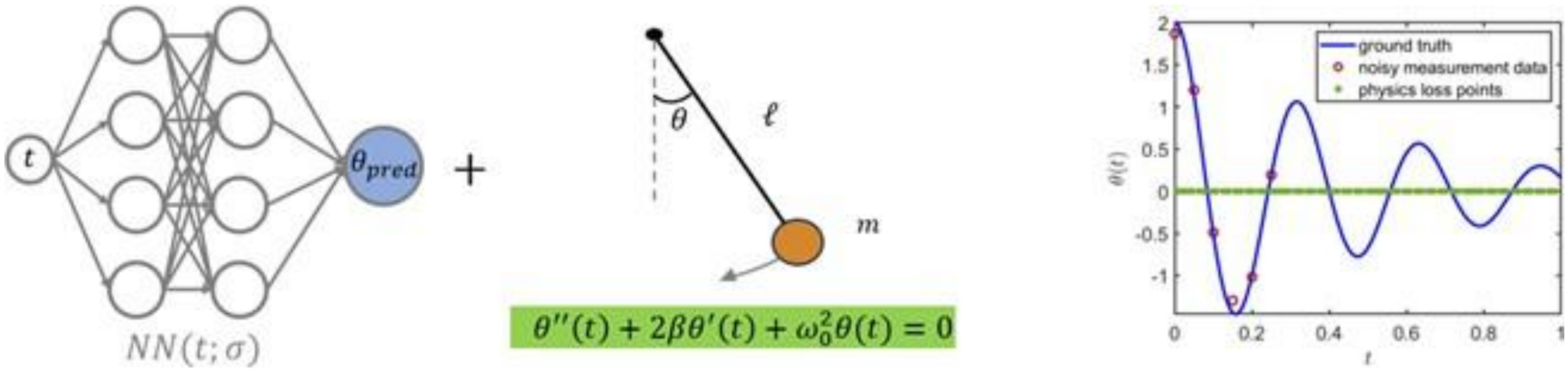


Figure 3: [MathWorks: PINNs]

Example: a damped pendulum - PINN



$$\theta''(t) + 2\beta\theta'(t) + \omega_0^2\theta(t) = 0$$

$$\min_{\sigma} \; \frac{1}{N}\sum_{i=1}^{N} \left|\theta_{pred}(t_i;\sigma) - \theta_{meas}(t_i)\right|^2$$

$$+\frac{1}{M}\sum_{j=1}^{M} \left|\frac{\partial^2}{\partial t^2}\theta_{pred}(t_j;\sigma) + 2\beta\frac{\partial}{\partial t}\theta_{pred}(t_j;\sigma) + \omega_0^2\theta_{pred}(t_j;\sigma)\right|^2$$

Figure 4: MathWorks: PINNs

Example: a damped pendulum - PINN solution

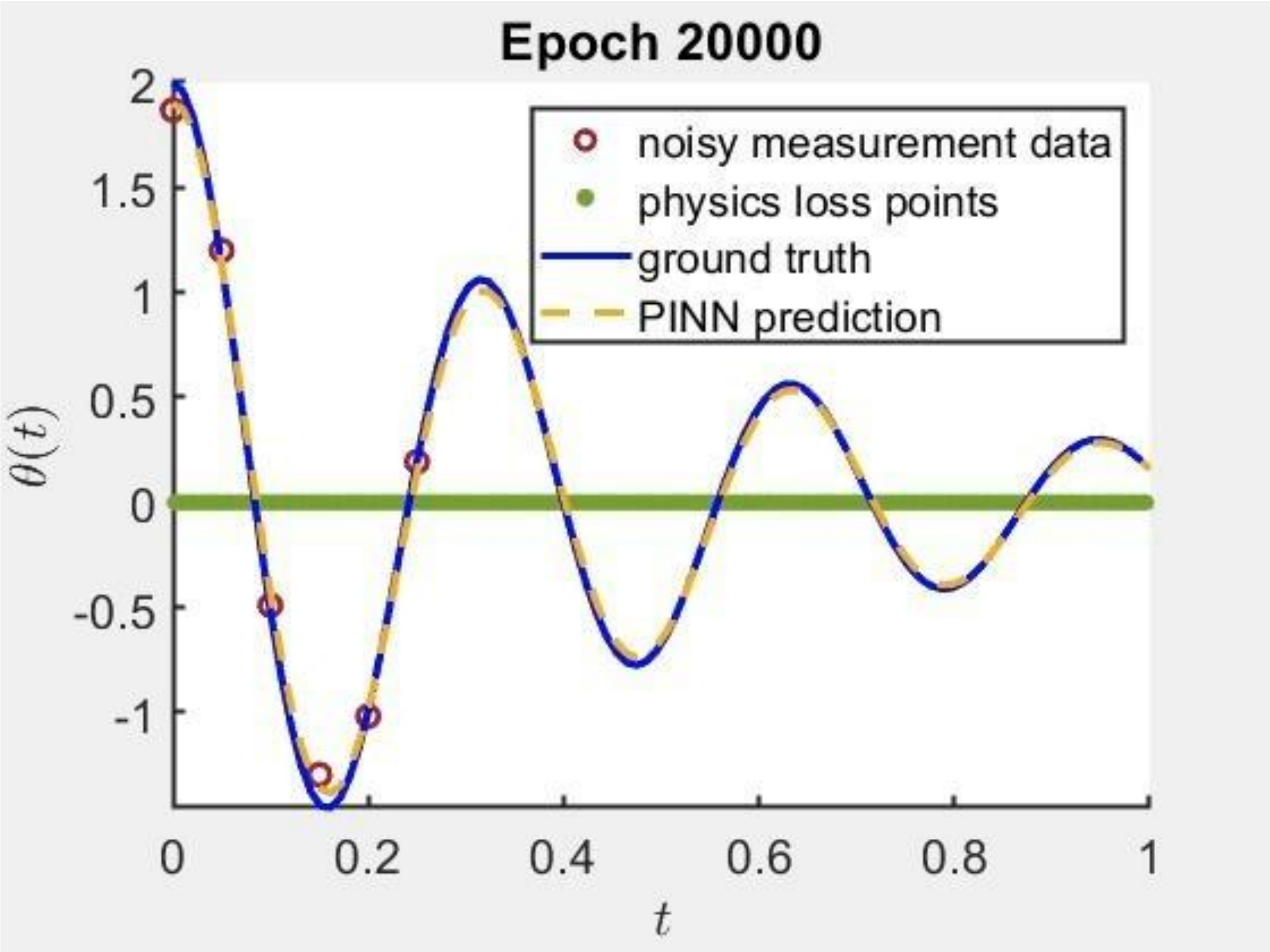


Figure 5: MathWorks: PINNs

Why PINNs? – Key Advantages

- Mesh-free and flexible; continuous solutions over domain.
- Physics-consistent; integrates sparse data.
- Unified forward and inverse framework.
- Often scales better in higher dimensions; substituting modeling.

Trade-off: ML vs. Numerical Methods vs. PINNs

	Purely Data-Driven Approaches	Traditional Numerical Methods	PINNs
Incorporate known physics	✗	✓	✓
Generalize well with limited or noisy training data	✗	✗	✓
Solve forward and inverse problems simultaneously	✓	✗	✓
Solve high-dimensional PDEs	✗	✗	✓
Enable fast "online" prediction	✓	✗	✓
Are mesh-free	✓	✗	✓
Have well-understood convergence theory	✗	✓	✗
Scale well to high-frequency and multiscale PDEs	✗	✓	✗

Figure 6: MathWorks: PINNs

Outline

- **Upcoming classes: guest lectures and student presentations**
- **Introduction to PINNs**
- **Challenges of PINNs**
- **Recap of neural PDE solvers**

Major challenges of PINNs

Optimization: actually training the PINN can be very difficult

- ad hoc / non-standardized training approaches
- convergence not guaranteed
- imbalance between losses (data / residual / BC) can lead to instability

Limited theory: underdeveloped convergence theory relative to classical methods

Higher-order derivatives needed to solve higher-order PDEs become expensive

Computational cost of calculating high-order derivatives

Difficulty expressing high-frequency, large-gradient, or multi-scale PDE solutions

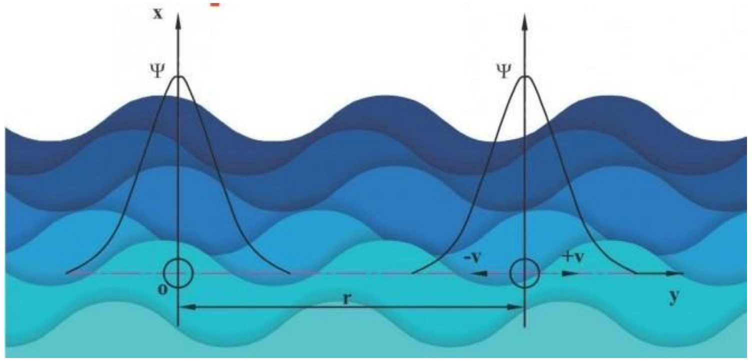
PINNs can fail to fit a simple convection problem

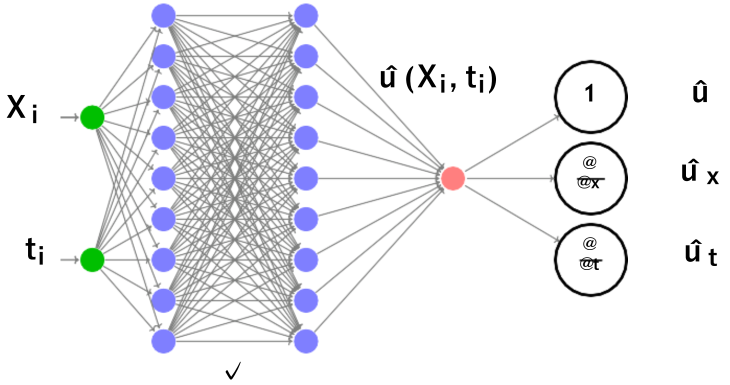
Problem formulation. We first consider a one-dimensional convection problem, a hyperbolic PDE which is commonly used to model transport phenomena:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega, \; t \in [0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$

Here, β is the convection coefficient and $h(x)$ is the initial condition. For constant β and periodic boundary conditions, this problem has a simple analytical solution:

$$u_{\text{analytical}}(x, t) = F^{-1}(F(h(x))e^{-i\beta k t}),$$





$$\min L = \lambda_F \left\| \hat{u}_t + \beta \hat{u}_x \right\|_2^2$$ PDE Residual

$$+ \left\| \hat{u}(x, 0) - \sin(x) \right\|_2^2$$ Initial Condition

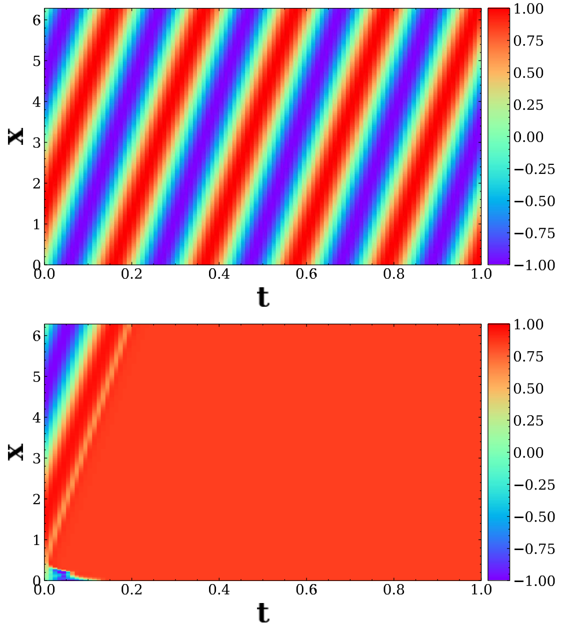
$$+ \left\| \hat{u}(x = 2\pi) - \hat{u}(x = 0) \right\|_2^2$$ Boundary Condition

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

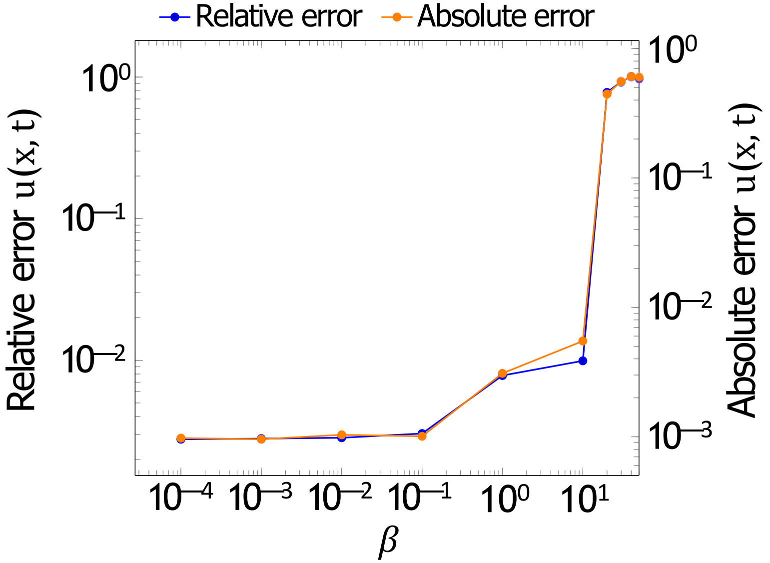
PINNs can fail to fit a simple convection problem

Exact Solution

PINN Prediction

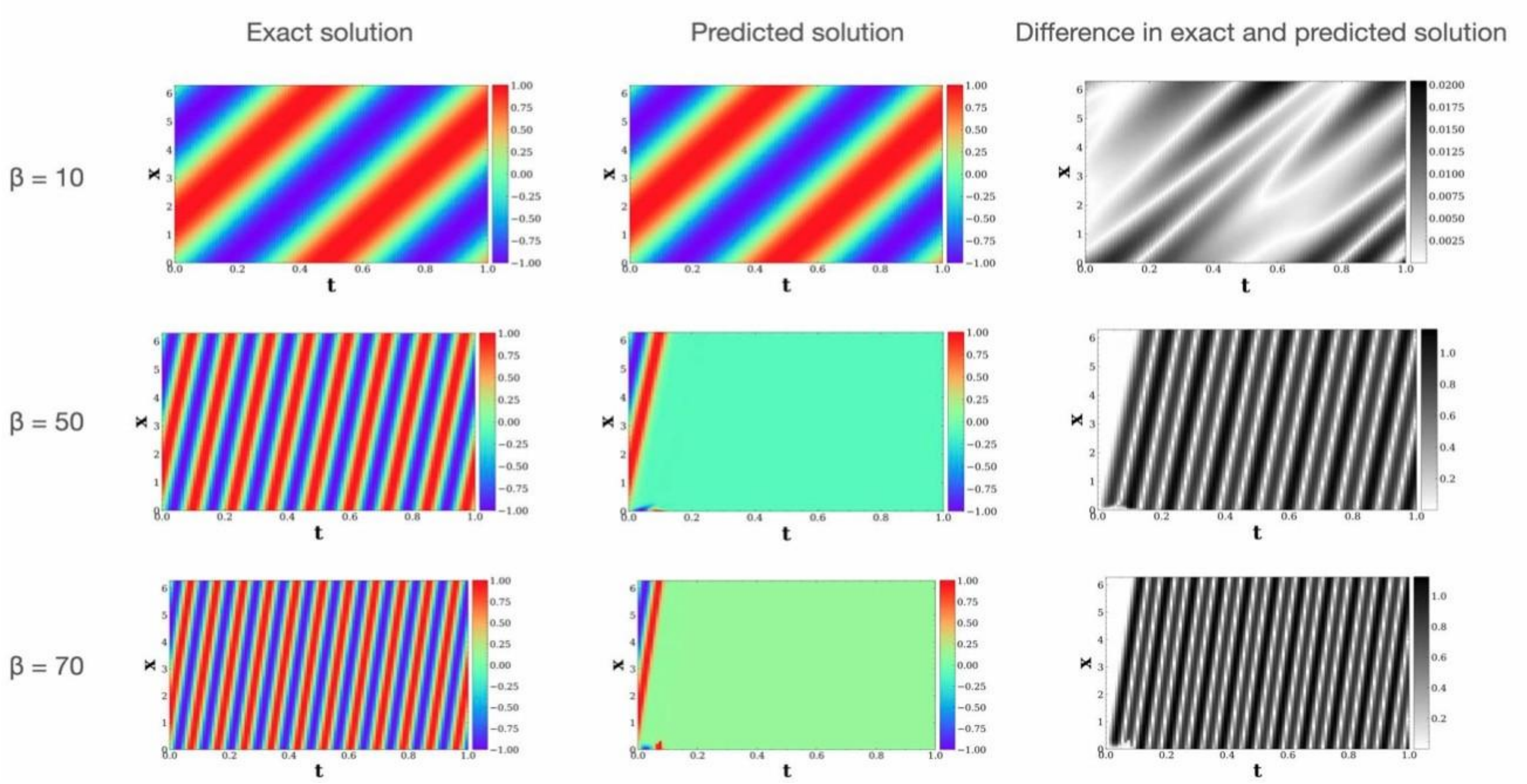


$\beta = 30$



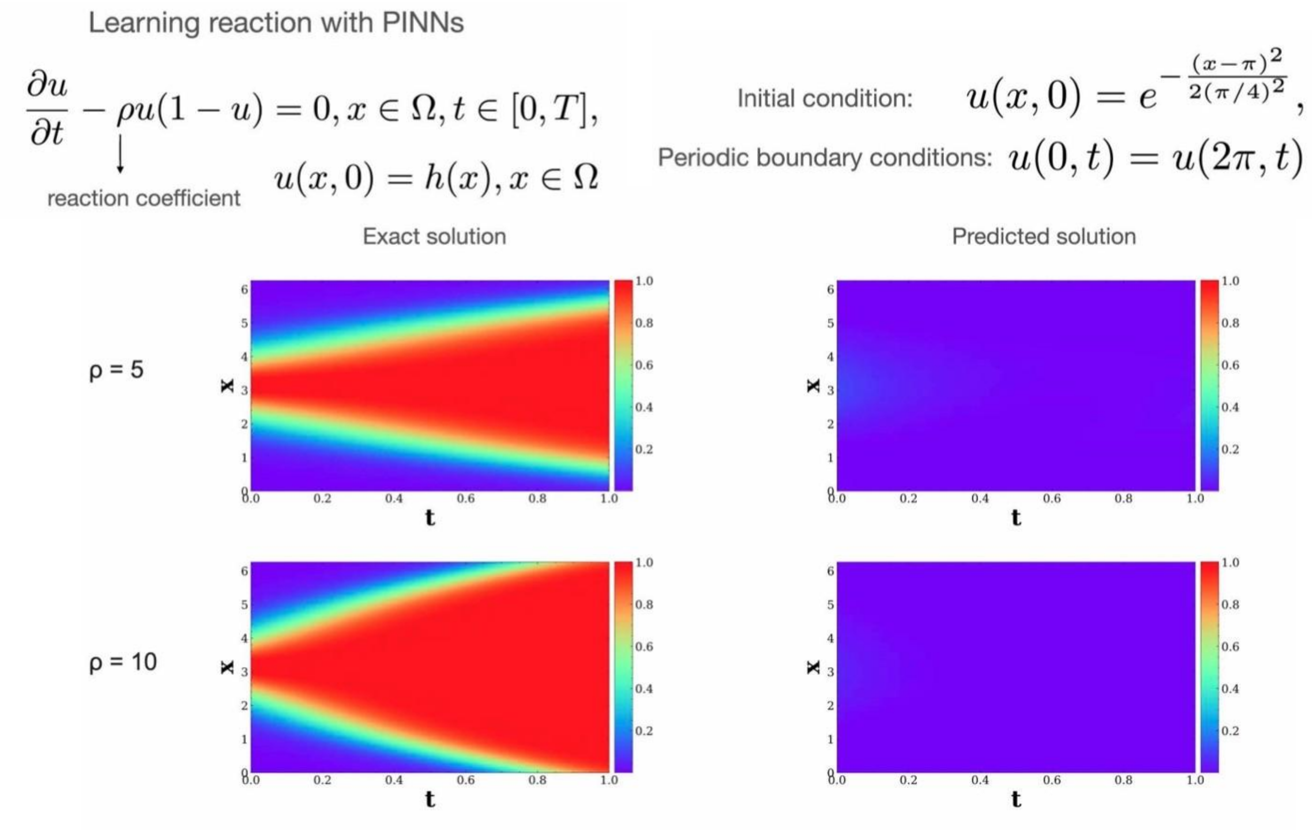
Krishnapriyan* AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. *Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

PINNs can fail to fit a simple convection problem



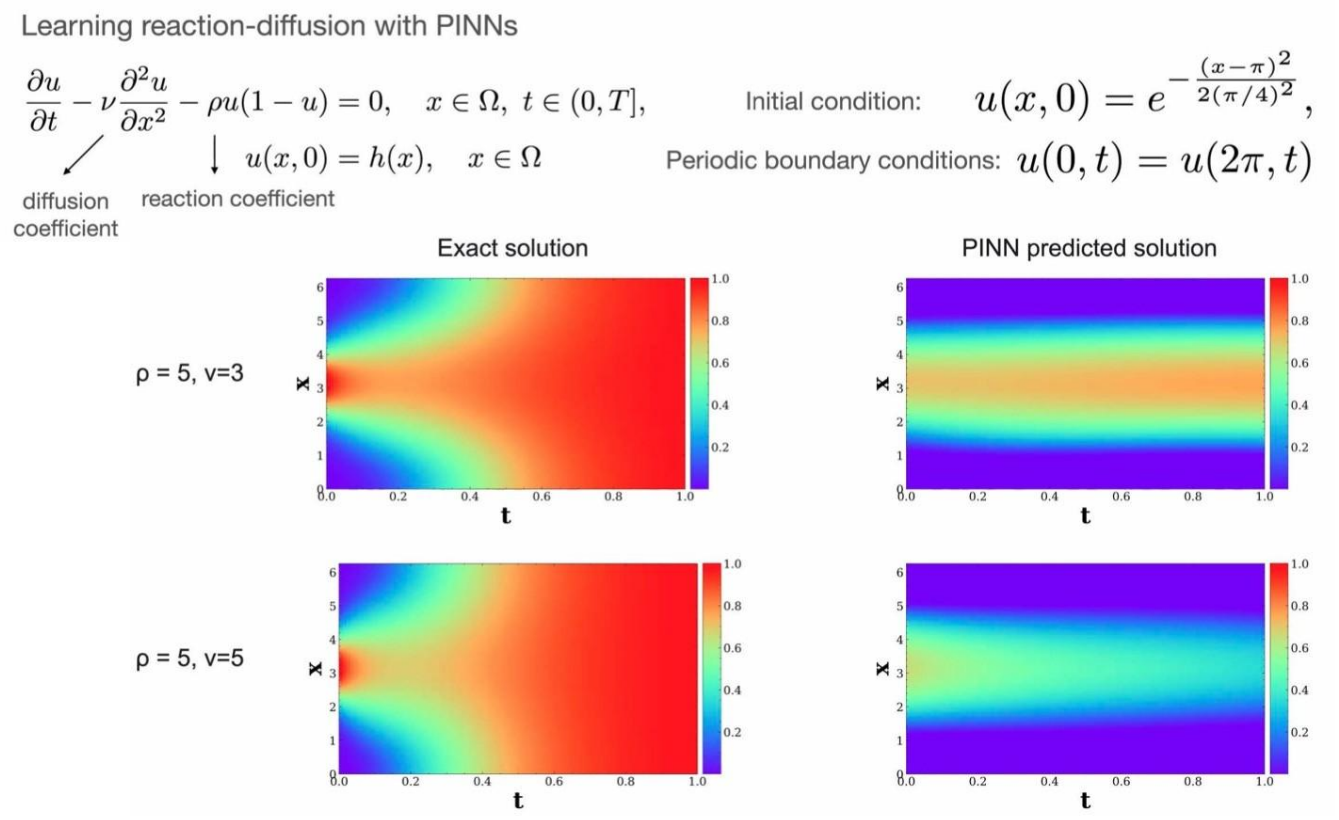
Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

PINNs can fail to fit a reaction equation



Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

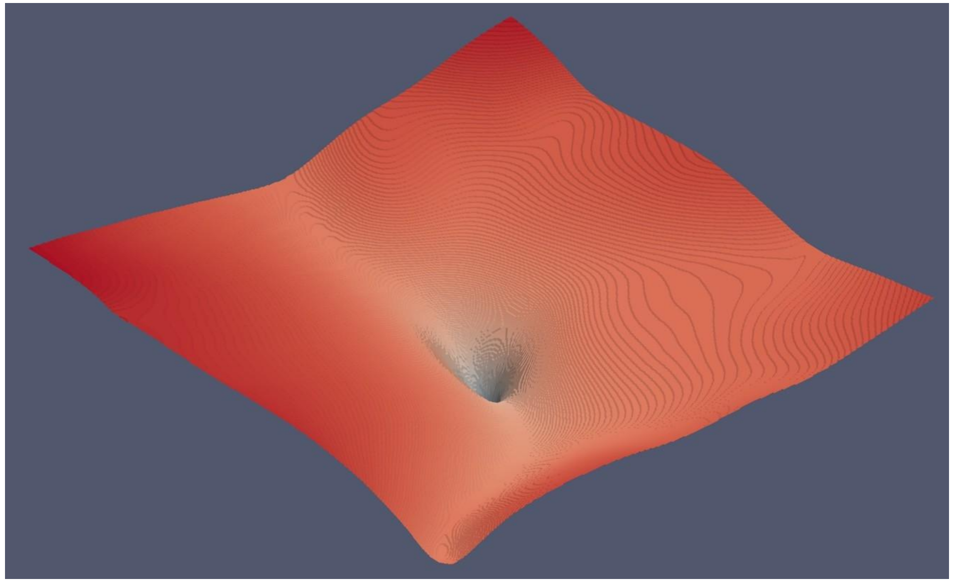
PINNs can fail to fit a reaction-diffusion equation



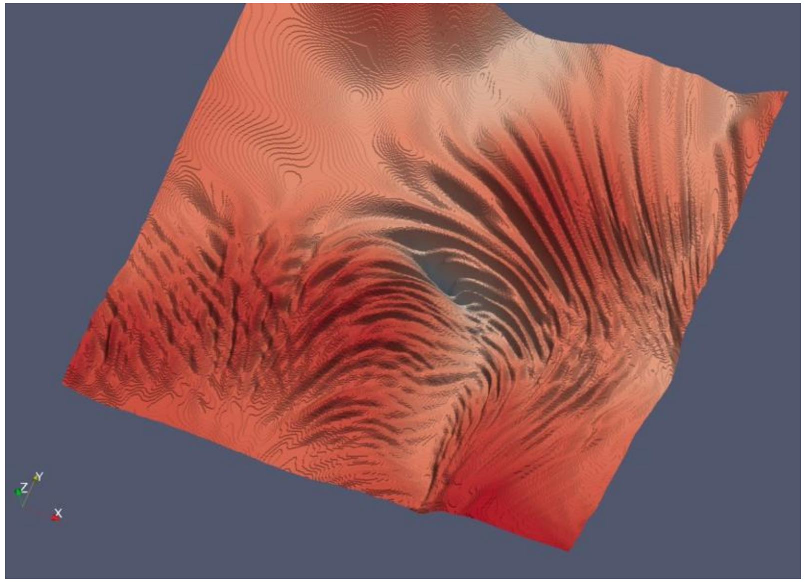
Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

Optimization challenges with PINNs

data loss : $L_u = \|\hat{u} - u\|_2^2$



Without Physics Loss



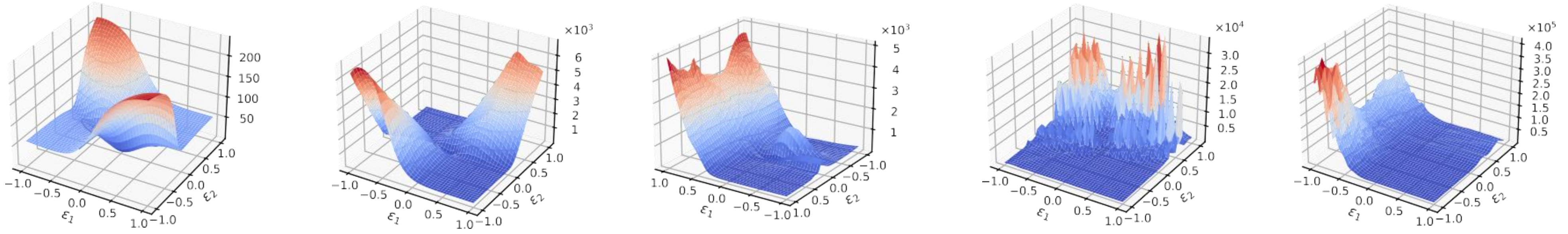
With Physics Loss

Roman Amici, Mike Kirby

Characterizing the issue

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$

What does the convection loss landscape look like for different β?



(a) $\beta = 1.0$ (b) $\beta = 10.0$ (c) $\beta = 20.0$ (d) $\beta = 30.0$ (e) $\beta = 40.0$

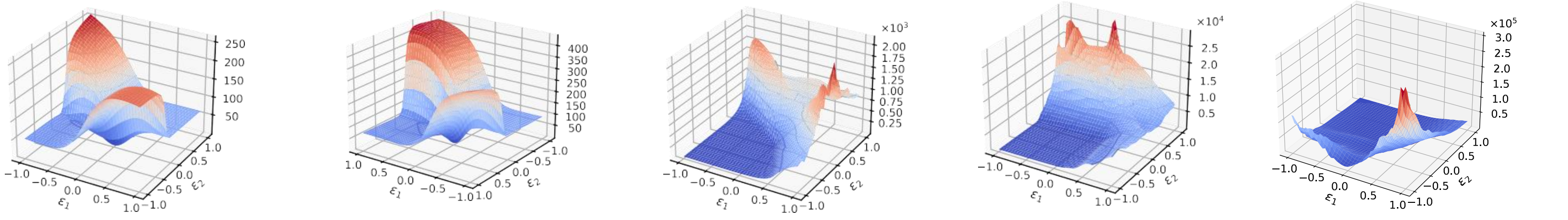
β	1	10	20	30	40
Relative error	7.84×10^{-3}	1.08×10^{-2}	7.50×10^{-1}	8.97×10^{-1}	9.61×10^{-1}
Absolute error	3.17×10^{-3}	6.03×10^{-3}	4.32×10^{-1}	5.42×10^{-1}	5.82×10^{-1}

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

Characterizing the issue

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$

As we reduce weight on the residual loss the optimization gets easier but the PINN's solution has ~100% error



(a) $\lambda = 1 \times 10^{-6}$ (b) $\lambda = 1 \times 10^{-5}$ (c) $\lambda = 1 \times 10^{-3}$ (d) $\lambda = 1 \times 10^{-1}$ (e) $\lambda = 1 \times 10^{1}$

λ	1×10^{-6}	1×10^{-5}	1×10^{-3}	1×10^{-1}	1×10^{1}
Relative error	1.69	1.65	1.00	1.08	0.982
Absolute error	0.987	0.987	0.623	0.647	0.595

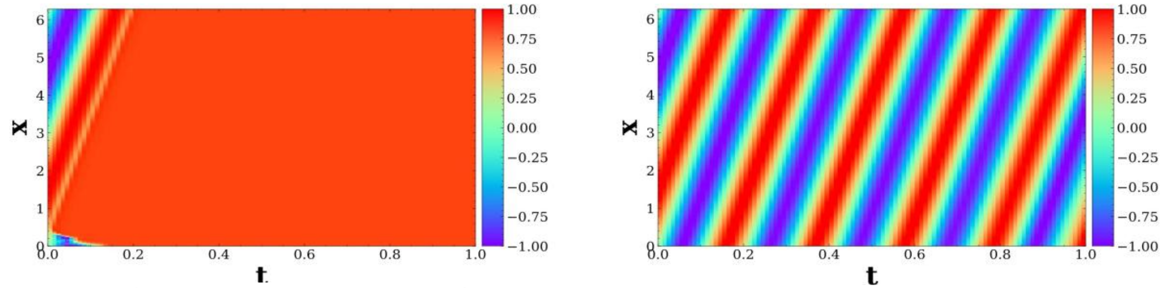
$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2 \quad \text{PDE Residual}$$
$$+ \|\hat{u}(x, 0) - sin(x)\|_2^2 \quad \text{Initial Condition}$$
$$+ \|\hat{u}(x = 2\pi) - \hat{u}(x = 0)\|_2^2 \quad \text{Boundary Condition}$$

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

Candidate solution: Curriculum learning

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0,T],$$
$$u(x,0) = h(x), \quad x \in \Omega.$$

Gradually increase the β starting from an easier-to-fit setting



Regular training PINN solution for $\beta = 30$

Curriculum training PINN solution for $\beta = 30$

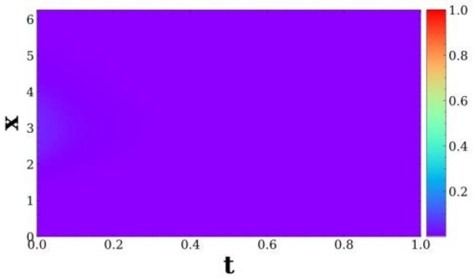
		Regular PINN	Curriculum training
1D convection: $\beta = 20$	Relative error	7.50×10^{-1}	$\mathbf{9.84 \times 10^{-3}}$
	Absolute error	4.32×10^{-1}	$\mathbf{5.42 \times 10^{-3}}$
1D convection: $\beta = 30$	Relative error	8.97×10^{-1}	$\mathbf{2.02 \times 10^{-2}}$
	Absolute error	5.42×10^{-1}	$\mathbf{1.10 \times 10^{-2}}$
1D convection: $\beta = 40$	Relative error	9.61×10^{-1}	$\mathbf{5.33 \times 10^{-2}}$
	Absolute error	5.82×10^{-1}	$\mathbf{2.69 \times 10^{-2}}$

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

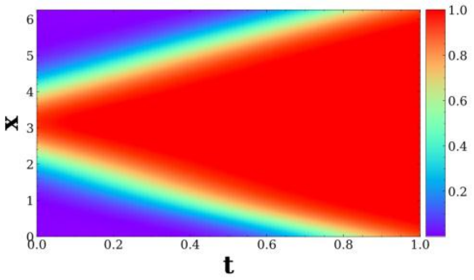
Candidate solution: Curriculum learning

Also works for the reaction equation:

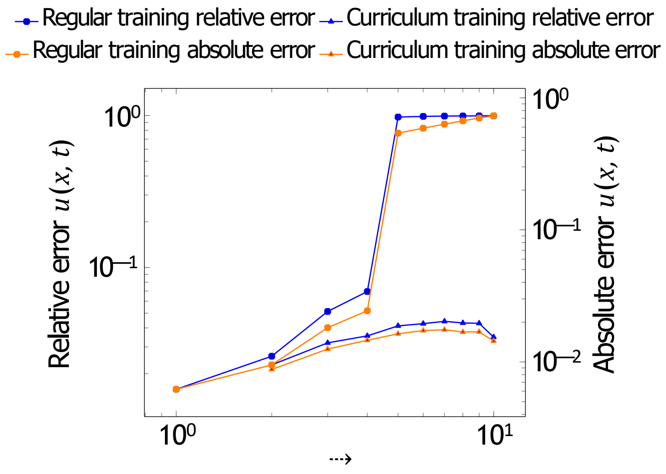
$$\frac{\partial u}{\partial t} - \rho u(1-u) = 0, \quad x \in \Omega, \ t \in (0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$



Regular training PINN solution for $\rho = 10$



Curriculum training PINN solution for $\rho = 10$

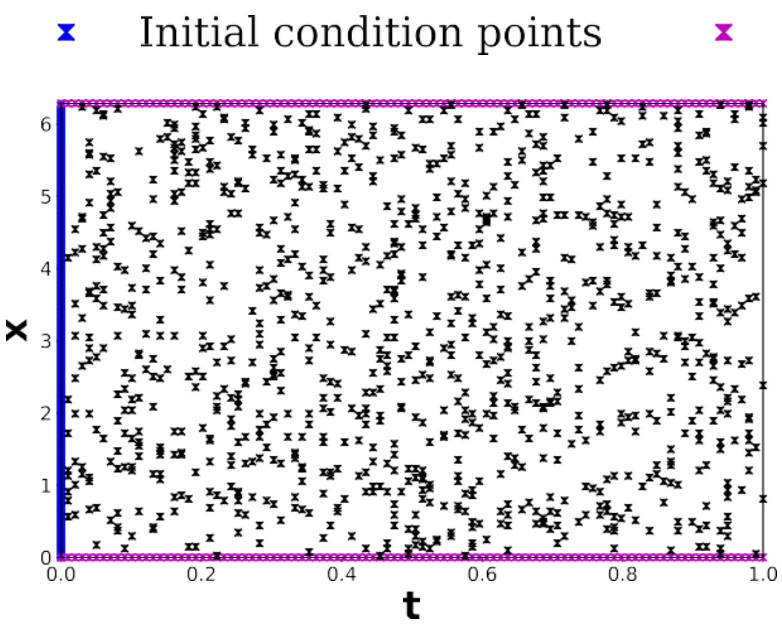


Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

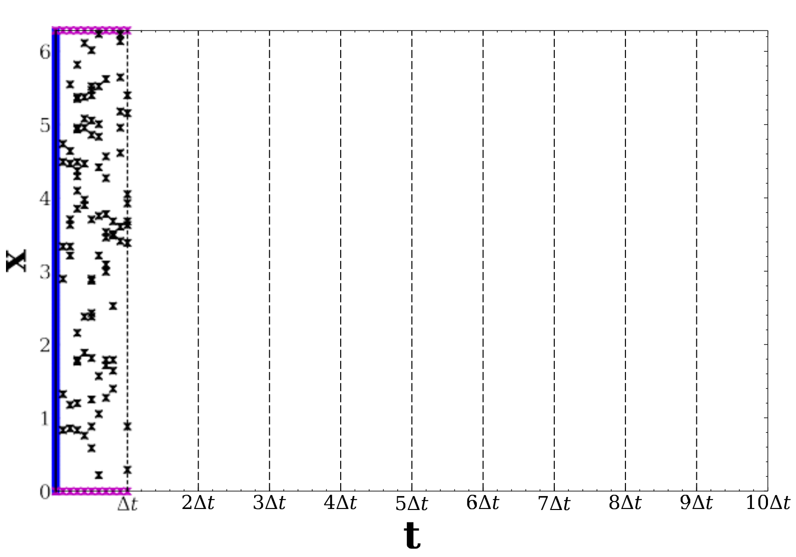
Candidate solution: Fit one step at a time

PINNs try to fit a function over all space and time simultaneously

This may be too hard, but it might be easier to only fit one timestep at a time and take timesteps (like a traditional solver…)



Regular PINN Training



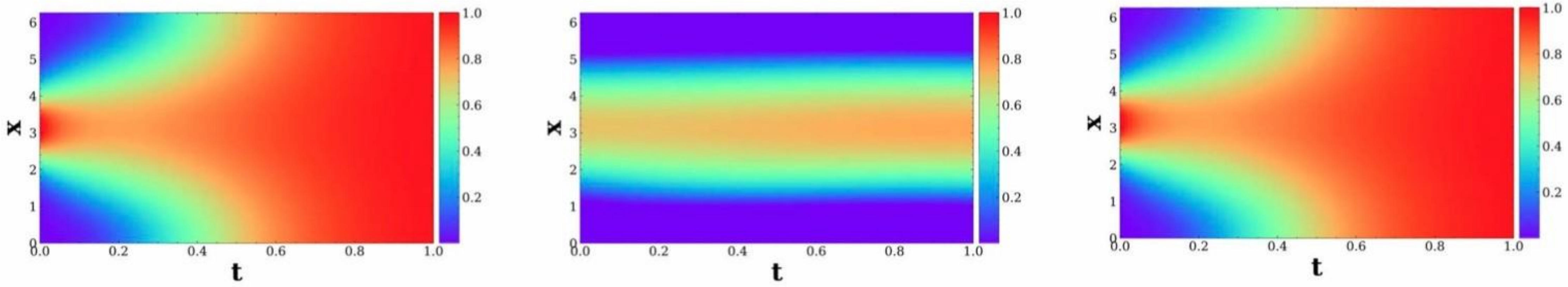
Seq2Seq Training

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

Candidate solution: Fit one step at a time

Works for reaction-diffusion

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \quad x \in \Omega,\ t \in (0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$



Exact solution for ρ = 5, v=3

Regular PINN solution for ρ = 5, v=3

seq2seq PINN solution for ρ = 5, v=3

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

Outline

- Upcoming classes: guest lectures and student presentations
- Introduction to PINNs
- Challenges of PINNs
- **Recap of neural PDE solvers**

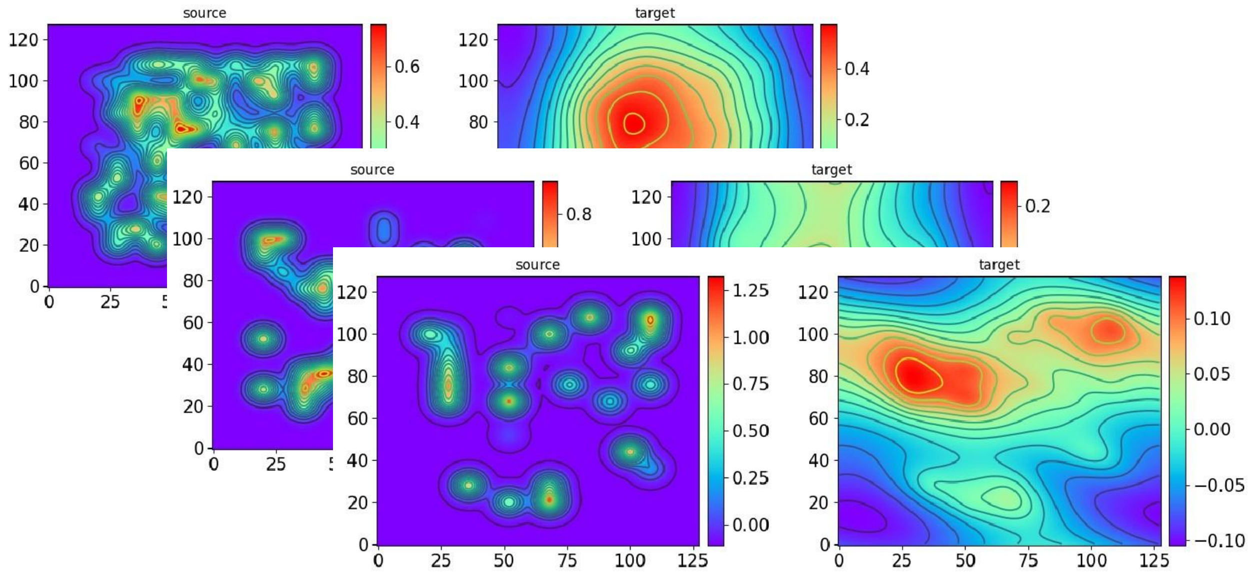
Neural PDE solver methods: Neural operators

Generate and train on a large amount of data

Pros:

- can learn the operator and apply it without specific constraints on mesh/discretization (with enough data)
- train once -> apply to different configurations (with enough data)
- does not need explicit knowledge of the underlying physics
- easy-to-implement
- very fast

Cons:
- often needs a lot of data
- no way to penalize the method



Neural PDE solver methods: Hard constraints

Enforce physics as hard constraints in the architecture or optimization

Pros:

- model will always obey the physics

Cons:

- very difficult to constrain the architecture to obey the laws
- constrained approaches are often really difficult to work with

Neural PDE solver methods: PINNs

Optimize PDE residual, experimental data, and constraints as penalties

Pros

- can vary the strictness of enforcing various constraints
- relatively easy to formulate and compute via autodiff

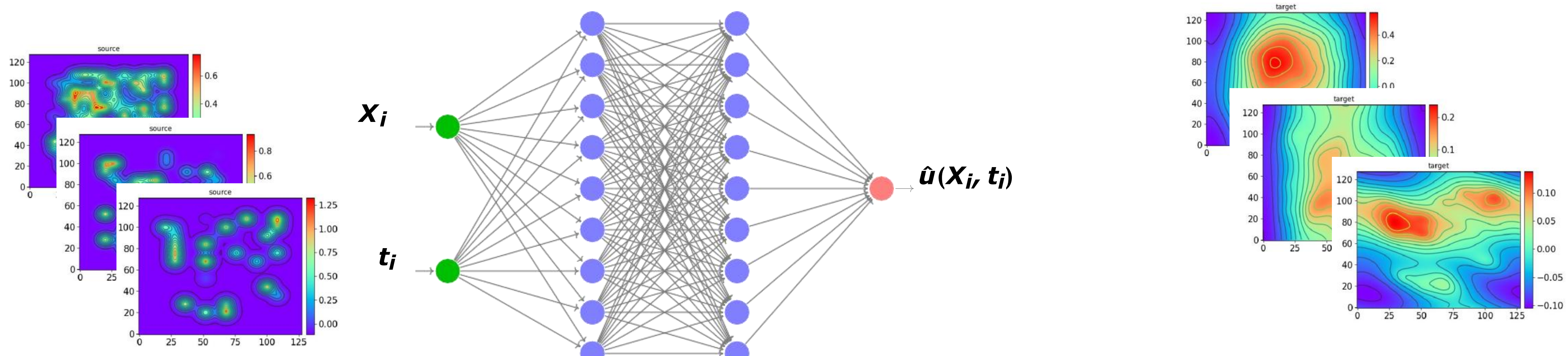
Cons:

- adding the soft constraint often makes the loss landscape very difficult
- needs to be retrained if PDE configuration is changed

Neural PDE solver methods: PINNs + neural operators

PINO method [Li et al., 2021]

- add empirical data and physics constraints as soft penalties to loss
- trades-off pros and cons of PINNs and neural operators…



Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Marek Cieślar and Amir Gholami.