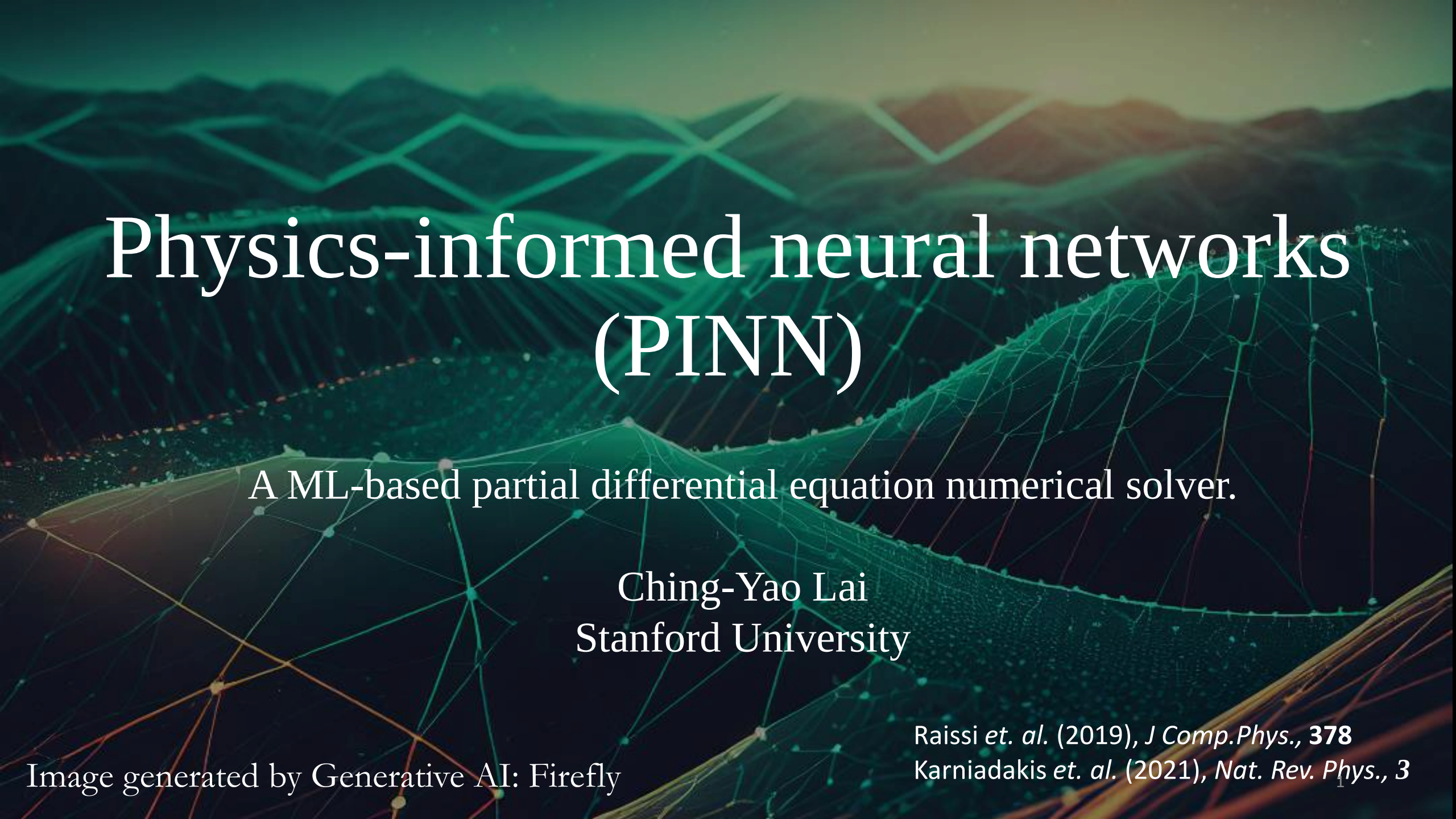


Physics-informed neural networks (PINN)

A ML-based partial differential equation numerical solver.

Ching-Yao Lai
Stanford University

Image generated by Generative AI: Firefly

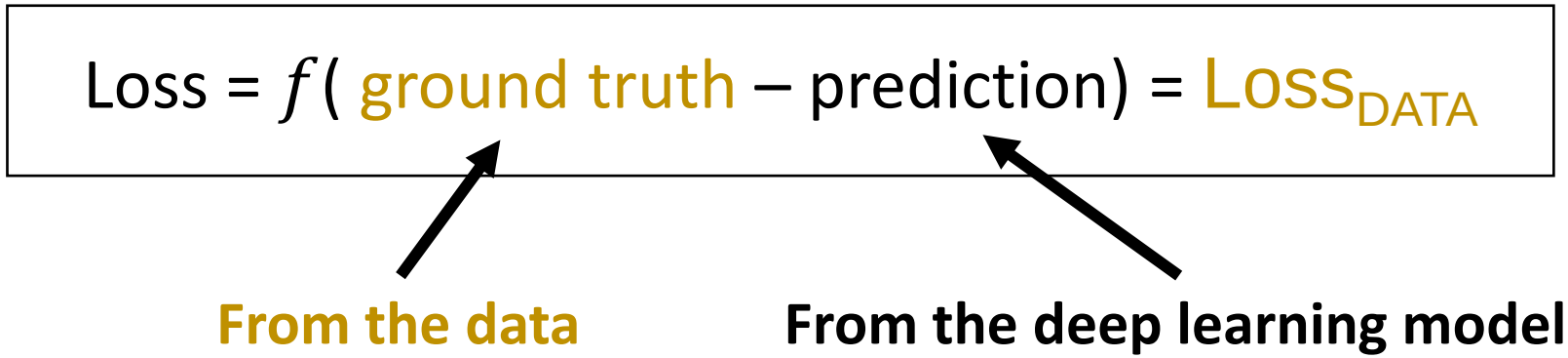
Raissi *et. al.* (2019), *J Comp.Phys.*, **378**
Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, ***3***

Outline

- Physics-informed neural networks
- Three applications
- Challenges and opportunities

Cost function

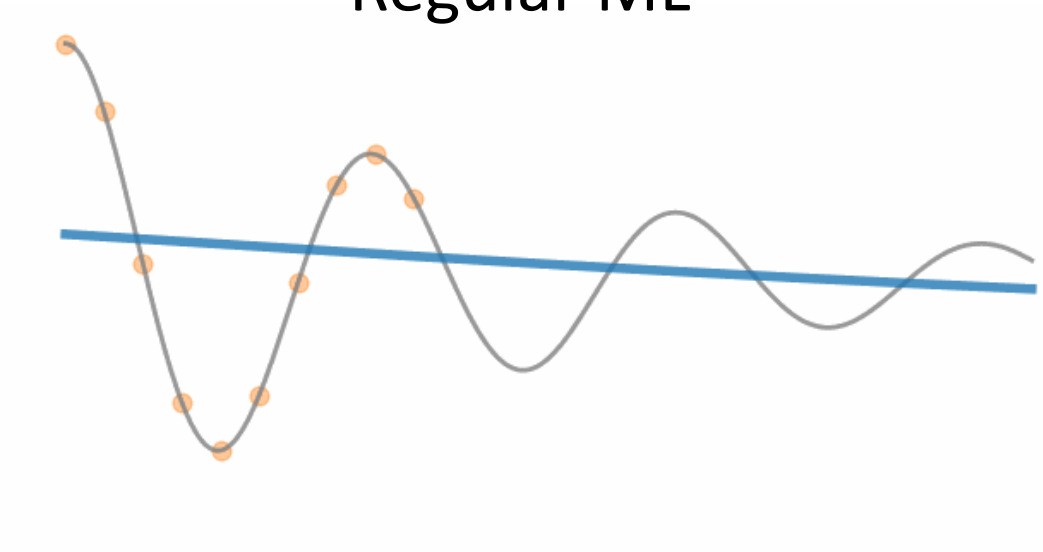
- Fits the ML model to the data



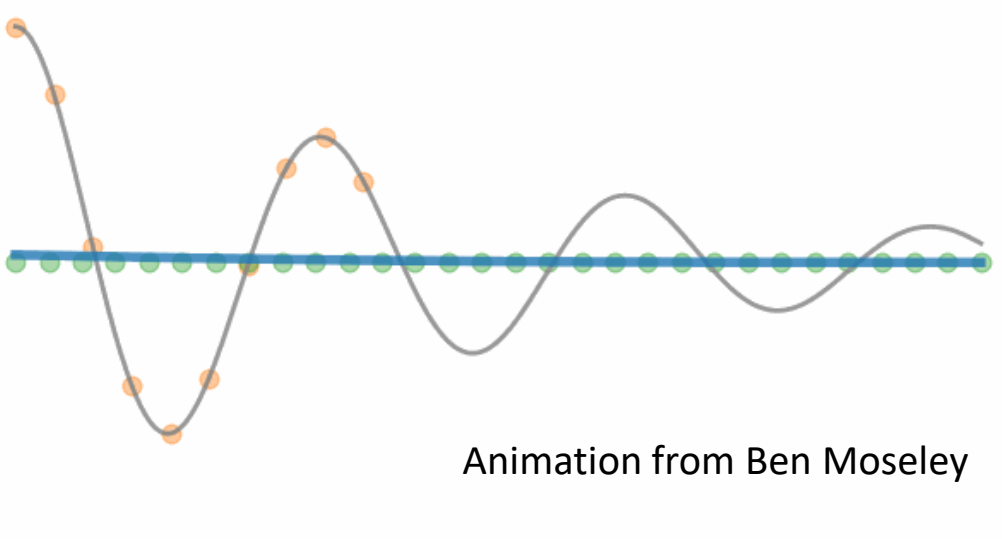
- Observation data is **almost certainly noisy**. How to avoid overfitting?
- Extrapolation: to what extent can the ML model generalize to unseen data?

Better extrapolation beyond training data

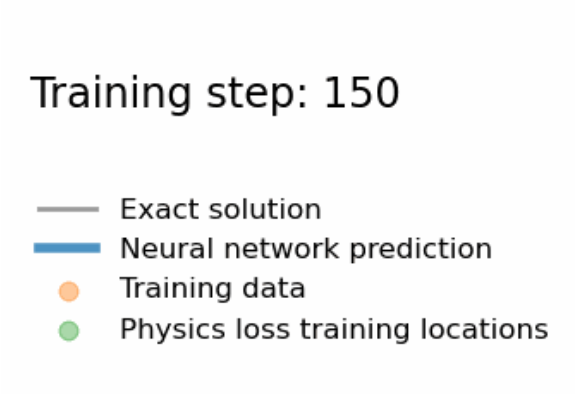
Regular ML



Physics-informed ML



Animation from Ben Moseley



Data loss: fit the data

Eqn loss: fit the physics equations

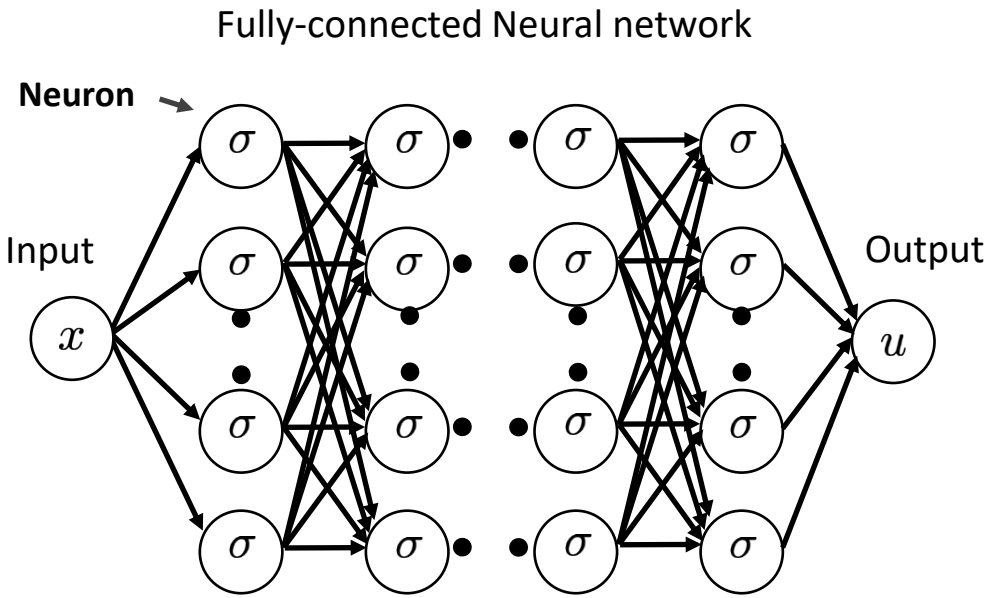
$$\text{Loss} = \text{Loss}_{\text{DATA}} + \text{Loss}_{\text{EQN}}$$

The new loss function:

1. **Avoid overfitting** data noise
2. Make predictions **interpretable**
3. **Extrapolations** beyond training data → trustworthy

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*



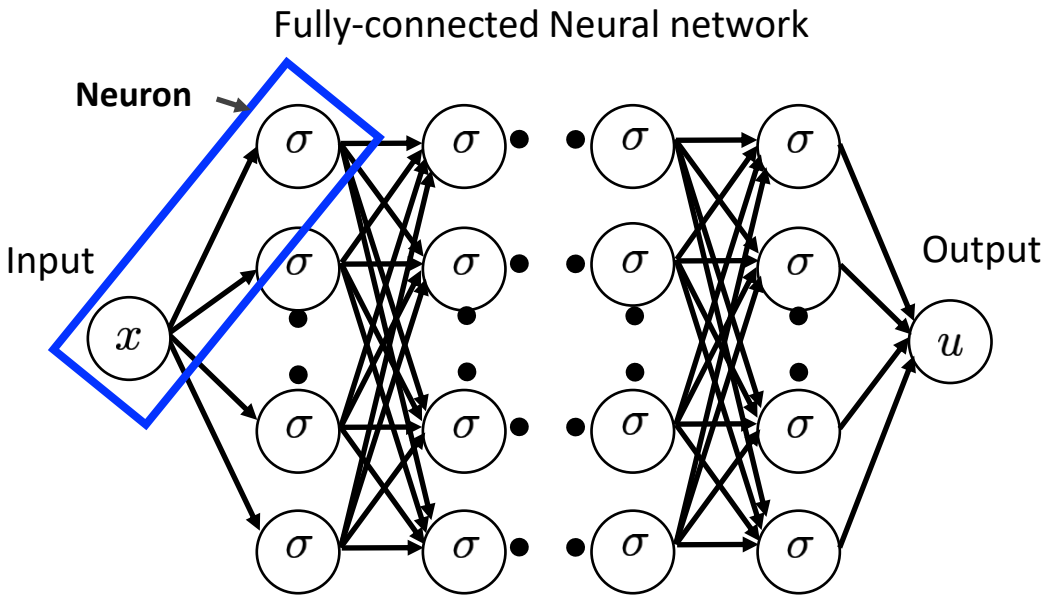
Function $u(x)$

Output

Input

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*



Function $u(x)$

Weights

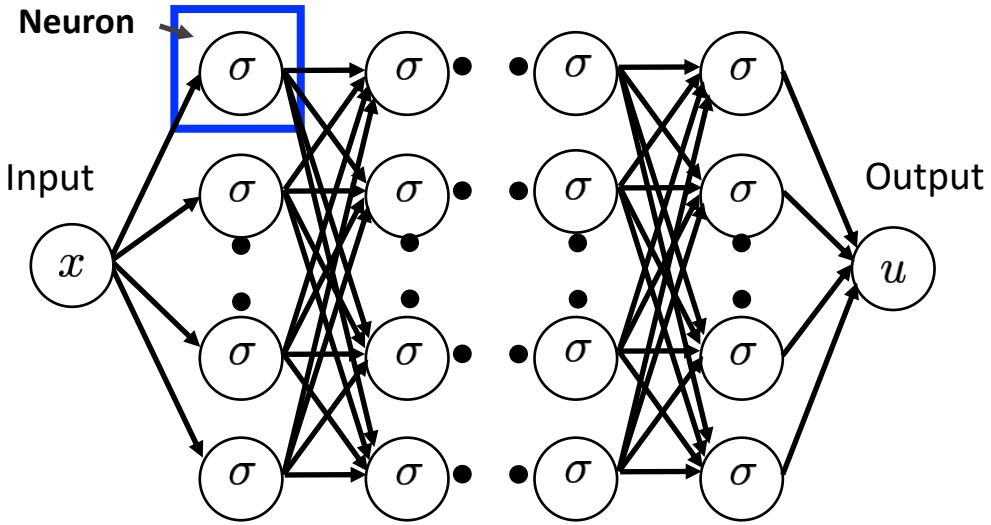
$w_i^{(0)} x + b_i^{(0)}$ *(both are free parameter)*

Biases

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*

Fully-connected Neural network



Function $u(x)$

Common choice

$\sigma(x) = \tanh(x)$

$\sigma(x) = \sin(x)$

Activation function (*nonlinearity*)

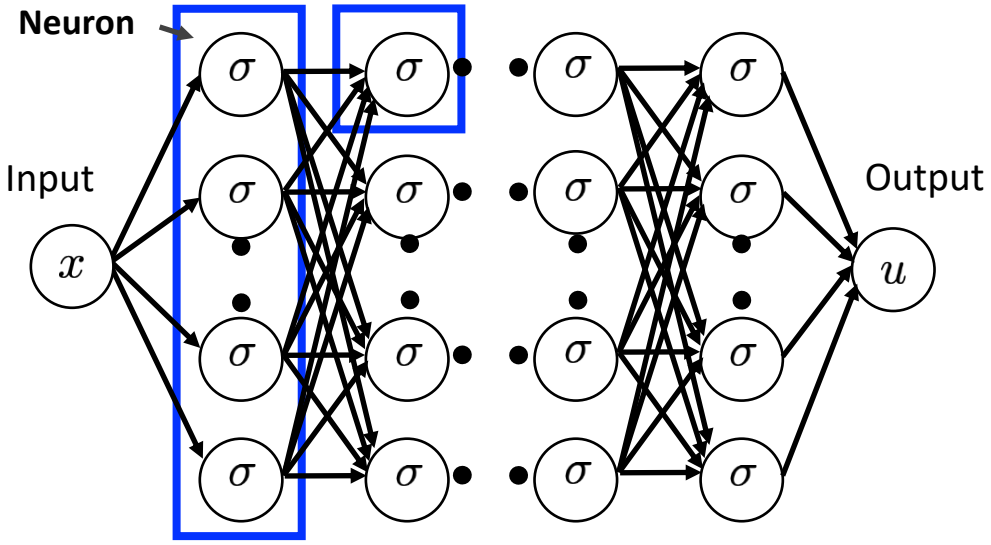
$$\sigma\left(w_i^{(0)} x + b_i^{(0)}\right)$$

Output of a neuron

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*

Fully-connected Neural network



Function $u(x)$

Sum of the outputs from previous layer

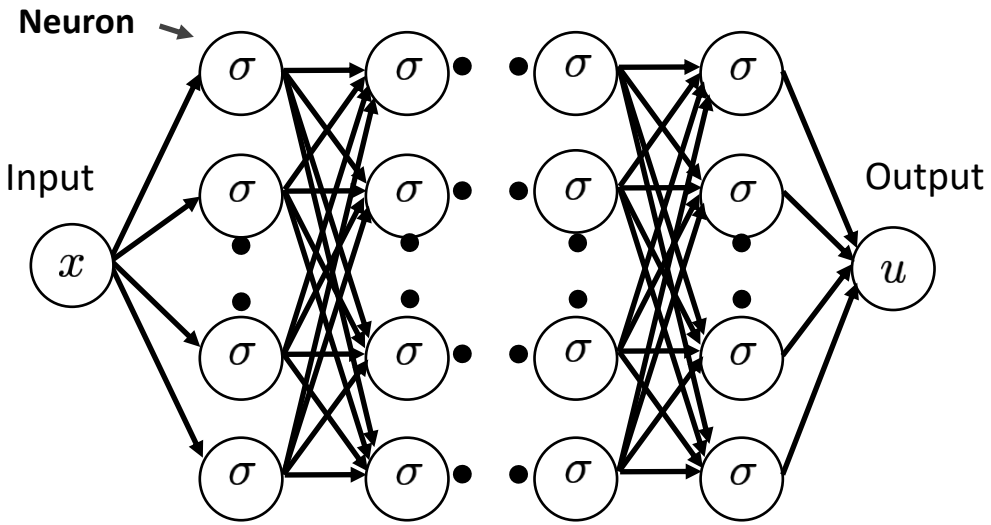
$$\sigma\left(\sum_{i=1} w_{ji}^{(1)}\,\sigma\left(w_i^{(0)}x + b_i^{(0)}\right) + b_j^{(1)}\right)$$

Output of a neuron in the second layer

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*

Fully-connected Neural network



Function $u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\sum_{i=1} w_{kj}^{(n-1)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_k^{(n-1)} \right) + b_l^{(n)}$

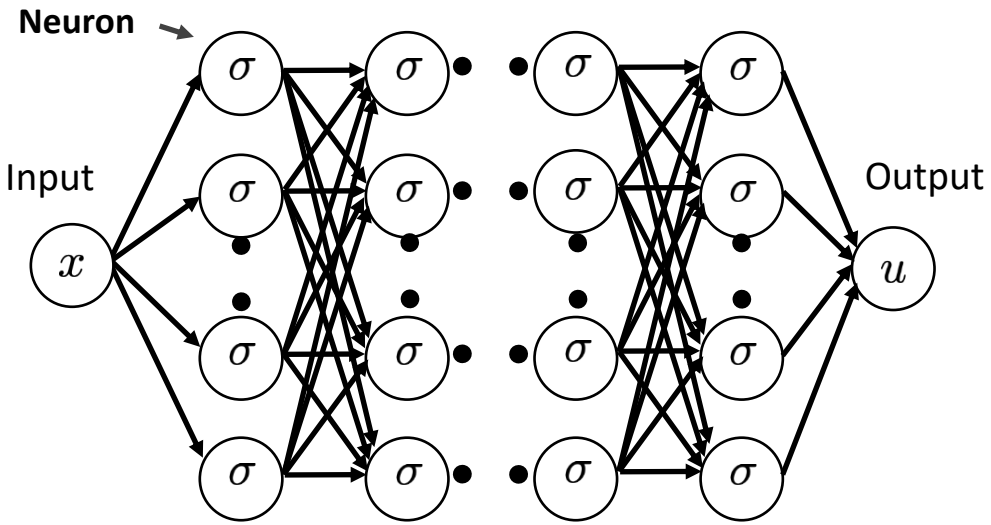
$\mathbf{w}$: weights $\quad \mathbf{b}$: biases $\quad \sigma(x)$: activation function

(free parameters to be trained) $\quad$ *(fixed and selected by users)*

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*

Fully-connected Neural network



$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

$\mathbf{w}$: weights $\quad$ $\mathbf{b}$: biases $\quad$ $\sigma(x)$: activation function

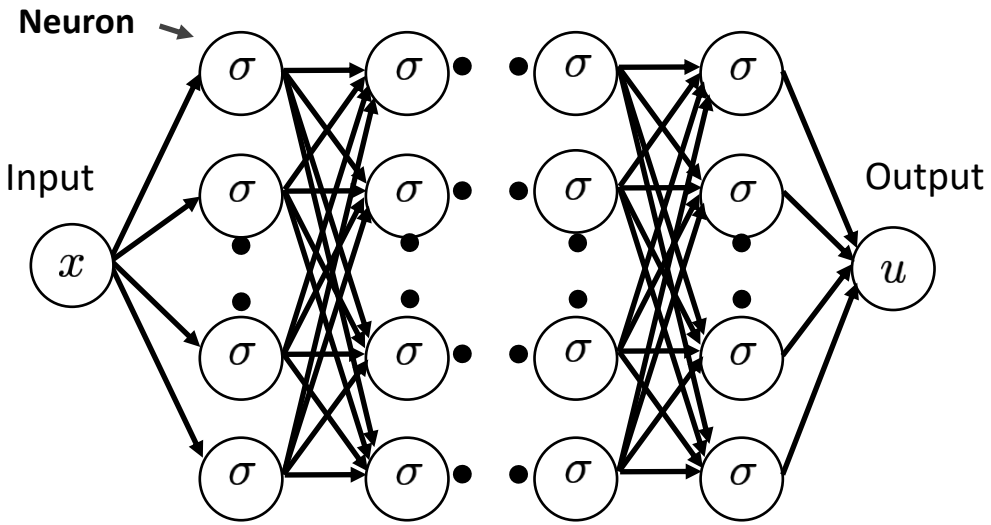
Universal function approximator

Hornik et. al. (1989), *Neural Netw*. **2**

Neural network and Fourier series

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*

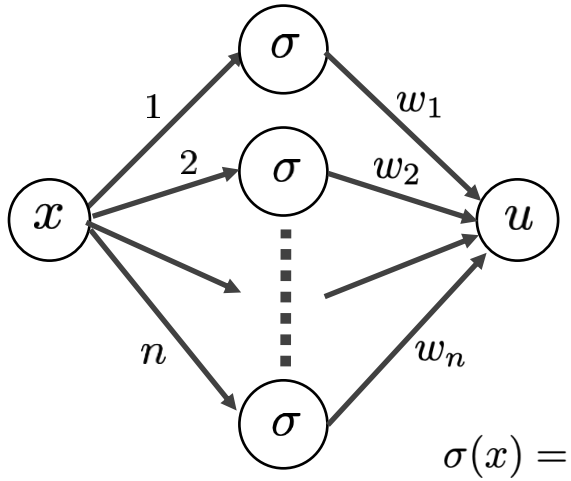
Fully-connected Neural network



$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

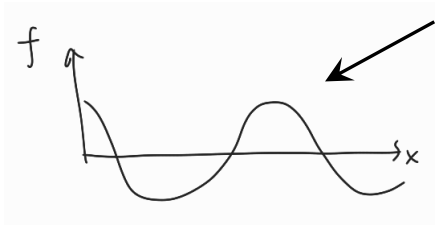
$\mathbf{w}$: weights $\quad$ $\mathbf{b}$: biases $\quad$ $\sigma(x)$: activation function

Fourier series: $u(x, w_n, b_n) = \sum_{n=0}^{N} w_n \sin(nx)$

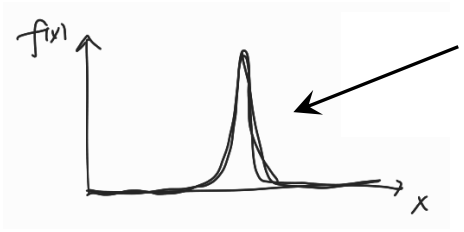


Universal function approximator

Hornik et. al. (1989), *Neural Netw.* **2**



Use sin() activation



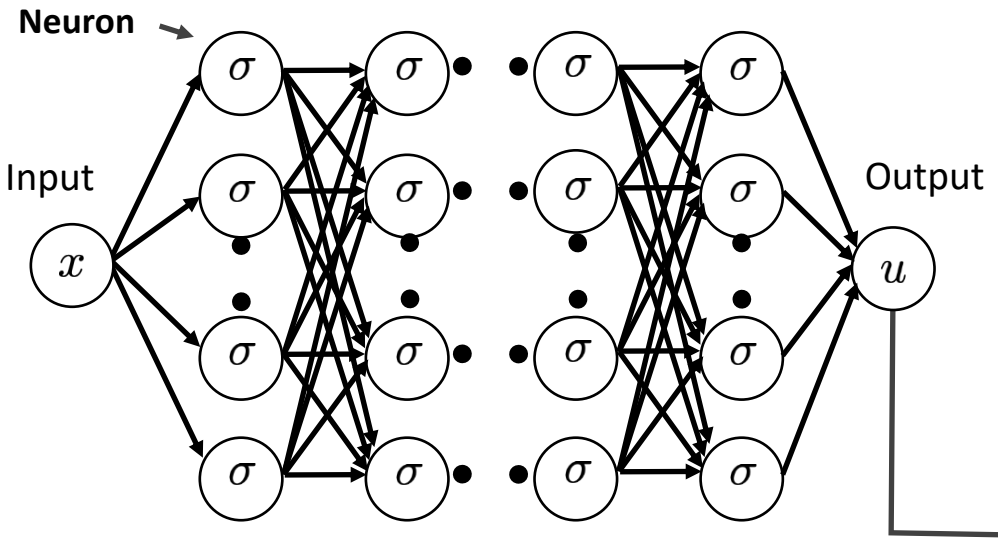
If using sin() activation, need lots of neurons (frequencies)

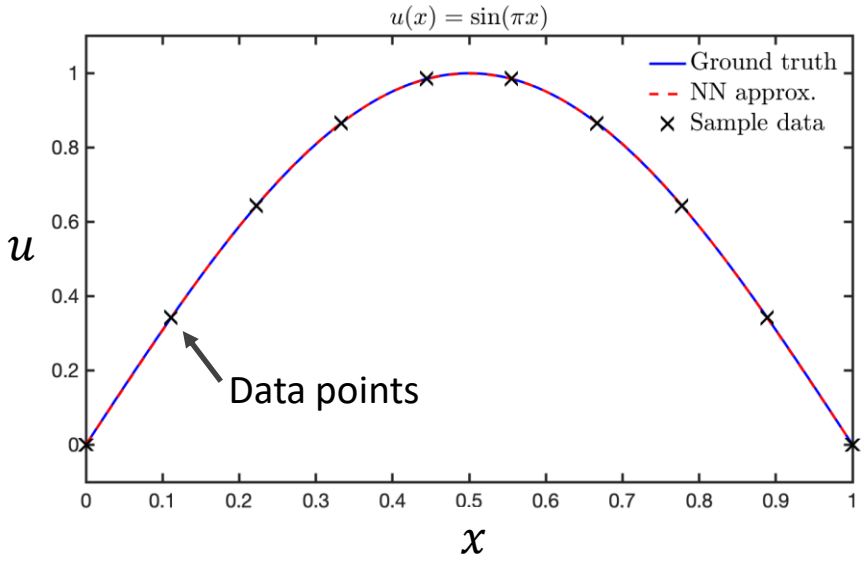
Better to use tanh() activation

Neural network for curve fitting

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*

Fully-connected Neural network





$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

Updating variables: $\mathbf{w}$: weights $\mathbf{b}$: biases

Optimization ← Loss

Gradient descent

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \, \nabla_{\mathbf{w}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

$$\mathbf{b}^{(i+1)} = \mathbf{b}^{(i)} - \eta \, \nabla_{\mathbf{b}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

$\mathbf{w}^{(i)}, \mathbf{b}^{(i)}$: value at the i-th iteration η : learning rate

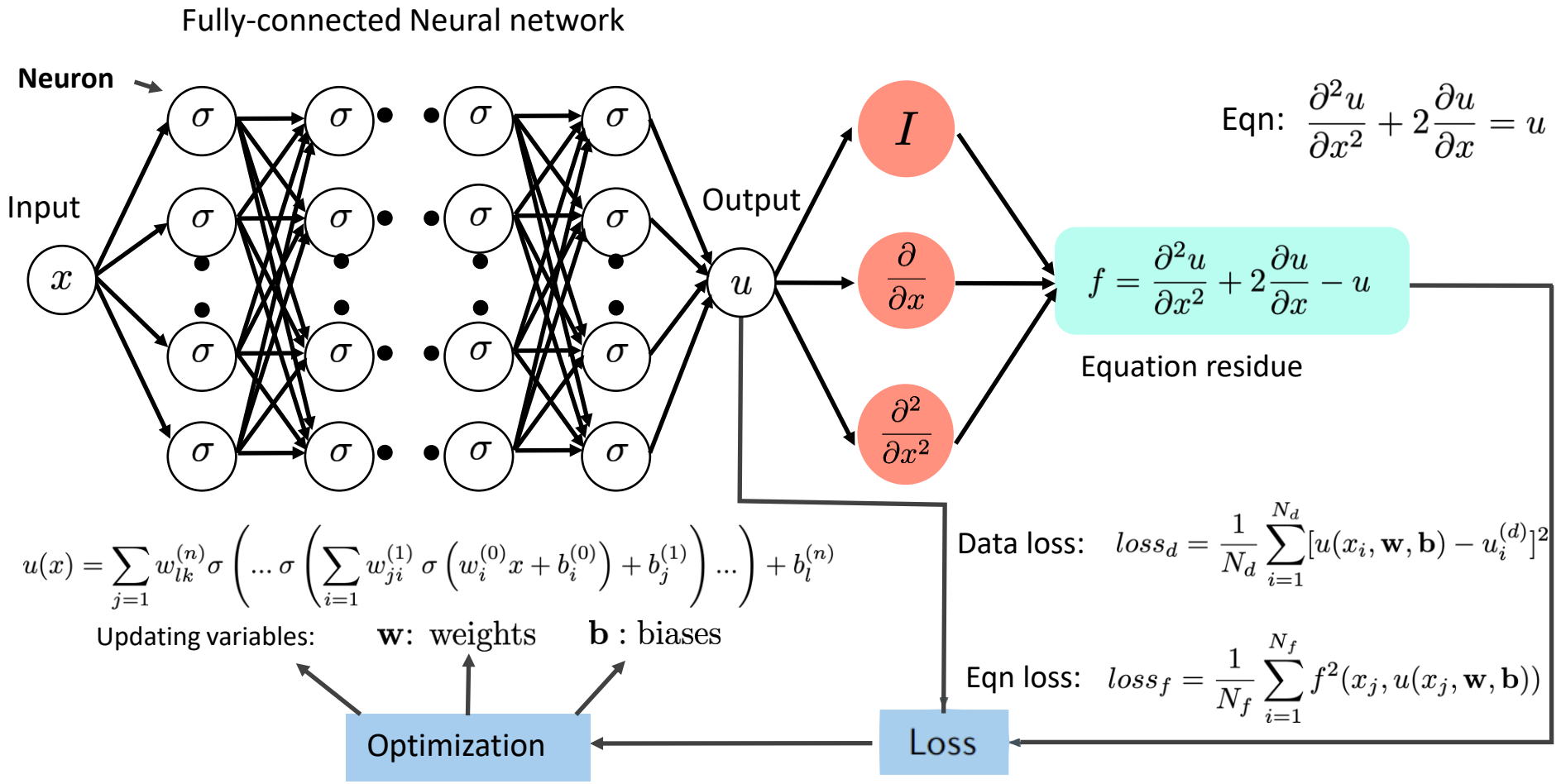
Cost function: *mean squared error*

data of u at $x = x_i$

$$J(x, \mathbf{w}, \mathbf{b}) = loss_d = \frac{1}{N_d} \sum_{i=1}^{N_d} [u(x_i, \mathbf{w}, \mathbf{b}) - u_i^{(d)}]^2$$

Physics-informed neural networks

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*



Gradient descent

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \nabla_{\mathbf{w}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

$$\mathbf{b}^{(i+1)} = \mathbf{b}^{(i)} - \eta \nabla_{\mathbf{b}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

$\mathbf{w}^{(i)}, \mathbf{b}^{(i)}$: value at the i-th iteration η : learning rate

Cost function: *data + equation loss*

$$J(x, \mathbf{w}, \mathbf{b}) = loss_d + loss_f$$

Physics-informed neural networks (PINN)

- A new class of partial differential equation numerical solver.
- Utilize the NN's representation power (a universal function approximator)
- Derivatives of the equation's solution $u(x,t)$ w.r.t x and t can be calculated exactly because we have an explicit expression of $u(x,t)$!

Raissi *et. al.* (2019), *J Comp.Phys.*, **378**
Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, ***3***

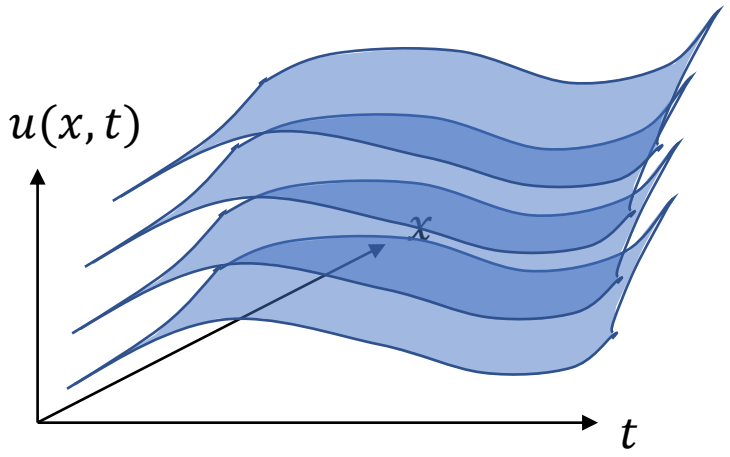
Example of solving PDEs: Diffusion equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2}$$

1st order derivative in time t
2nd order derivatives in space x

→ 1 initial condition
2 boundary conditions

→ $GOAL:\ solve\ for\ u(x,t)$



Need to give boundary and initial conditions to properly find a unique surface u(x,t)

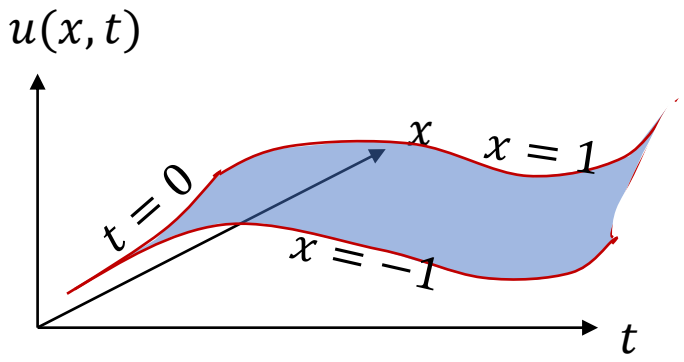
Example of solving PDEs: Diffusion equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2}$$

1st order derivative in time t
2nd order derivatives in space x

→ 1 initial condition
2 boundary conditions

→ $GOAL: \ solve \ for \ u(x,t)$



Need to give boundary and initial conditions to properly find a unique surface u(x,t)

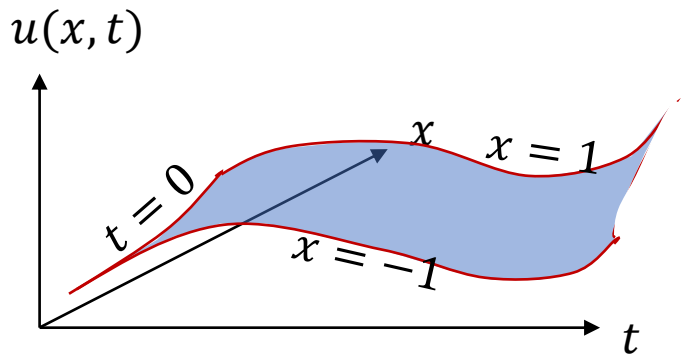
Example of solving PDEs: Diffusion equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2}$$

$Initial\ condition$: $u(t=0,x) = f(x)$

$Boundary\ conditions$: $u(t,x=-1) = u(t,x=1) = 0$

→ $GOAL$: $solve\ for\ u(x,t)$



Need to give boundary and initial conditions to properly find a unique surface u(x,t)

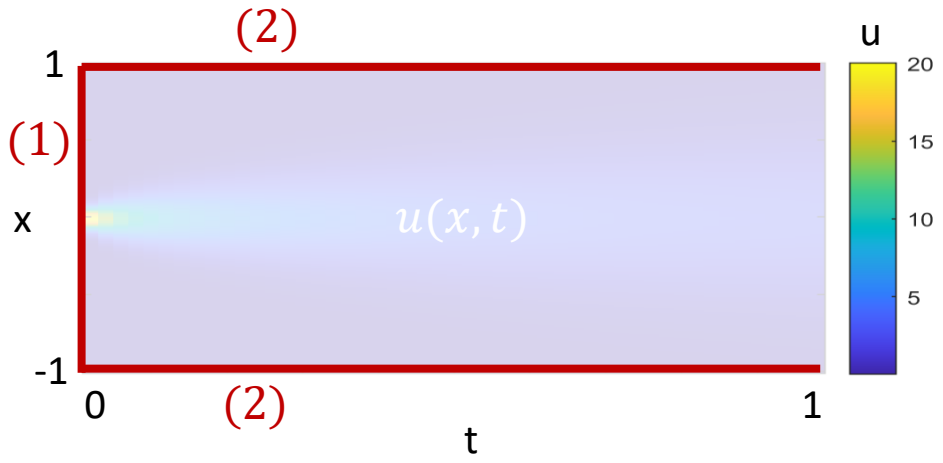
Learn u with physics equation + ICs + BCs

(3) $\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2}$

(1) *Initial condition*: $u(t=0,x) = f(x)$

(2) *Boundary conditions*: $u(t,x=-1) = u(t,x=1) = 0$

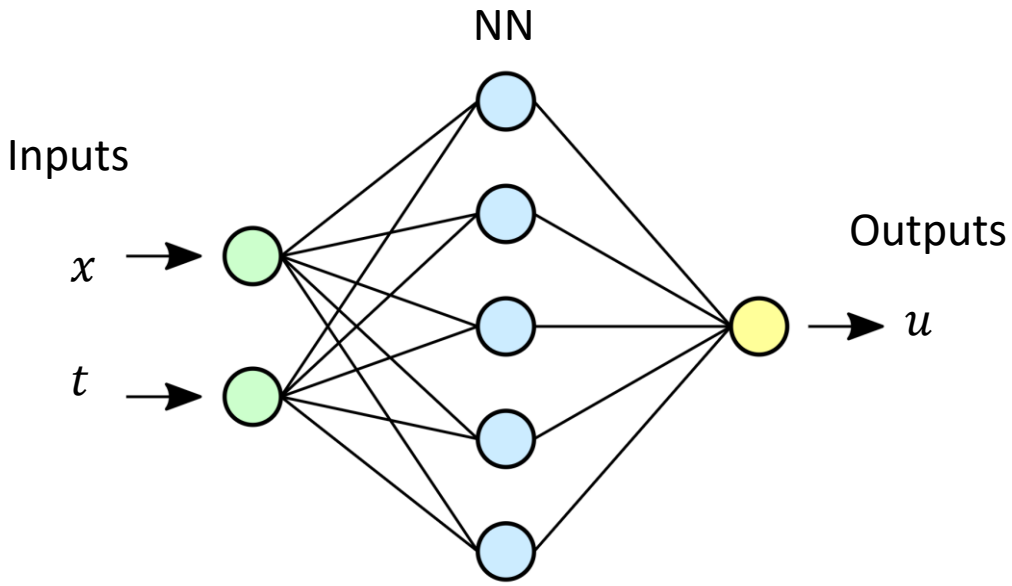
→ $GOAL$: $solve\ for\ u(x,t)$

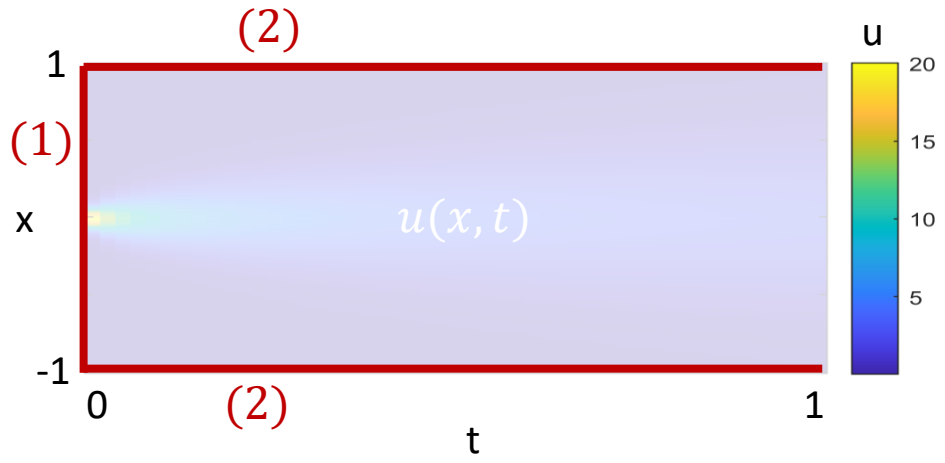


Mathematically, the information we need to get u(x,t) is **(1) ICs + (2) BCs + (3) physics equation.**

Instead of traditional numerical solver, can we use a NN to predict u(x,t) using (1), (2), (3)?

Learn u with physics equation + ICs + BCs





Use NN to search for a surface that satisfies (1) ICs + (2) BCs + (3) physics equation.

1. Initial NN(x,t) = u(x,t). x,t are inputs, u is output

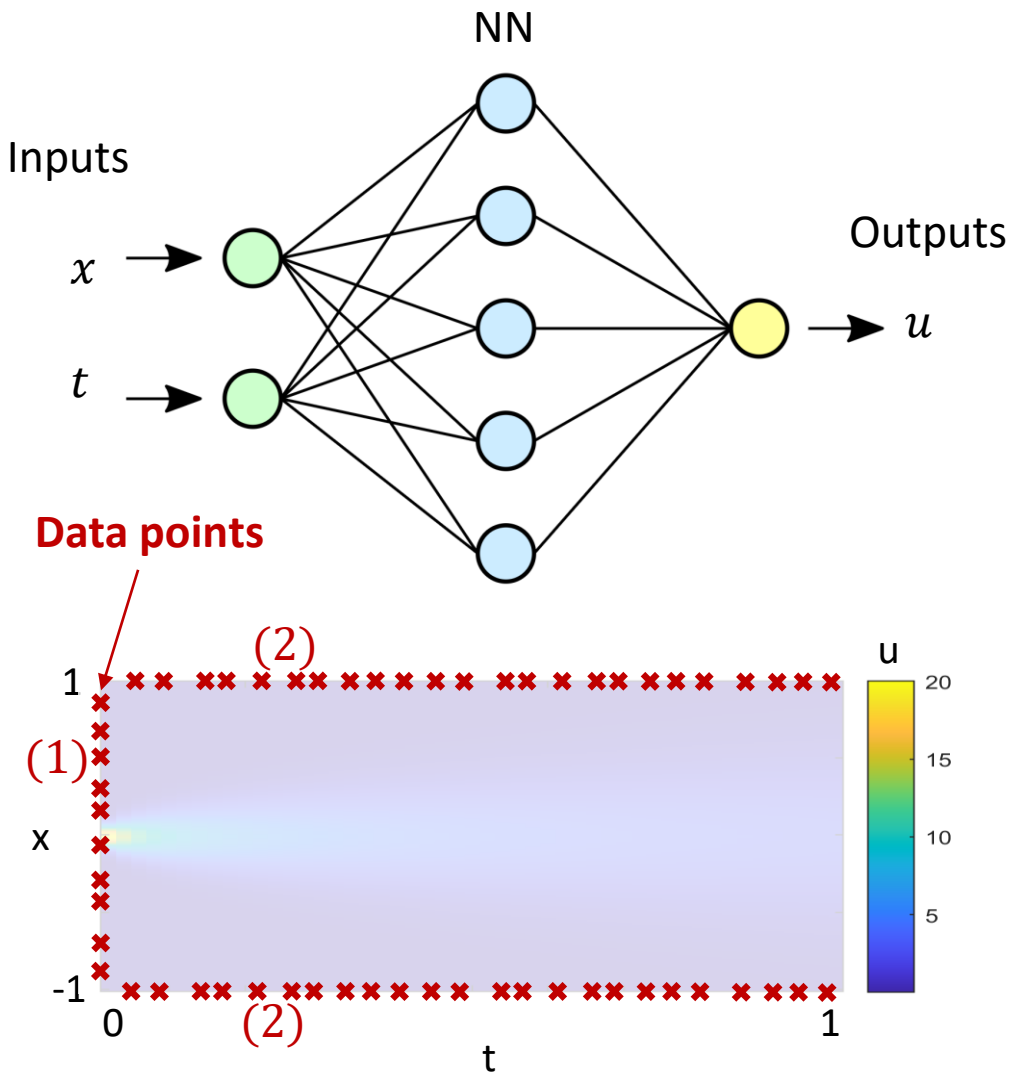
2. For a NN, all derivatives of u w.r.t x and t can be calculated analytically because u(x,t) is exact!

3. We can calculate $\frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial x^2}$

4. In the cost function, minimize $\left[\frac{\partial u}{\partial t} - a\frac{\partial^2 u}{\partial x^2}\right]^2$

Turn the problem into an optimization problem!

Physics-informed NN



Training data (ground truth):

(1) *Initial condition:*

$$u_0^i \; at \; \{t = 0, x_i\}, \; i = 1 \ldots . N$$

(2) *Boundary conditions*:

$$u_{lb}^j \; at \; \{t_j, x = -1\} \;\; \& \;\; u_{ub}^j \; at \; \{t_j, x = 1\}, j = 1 \ldots . M$$

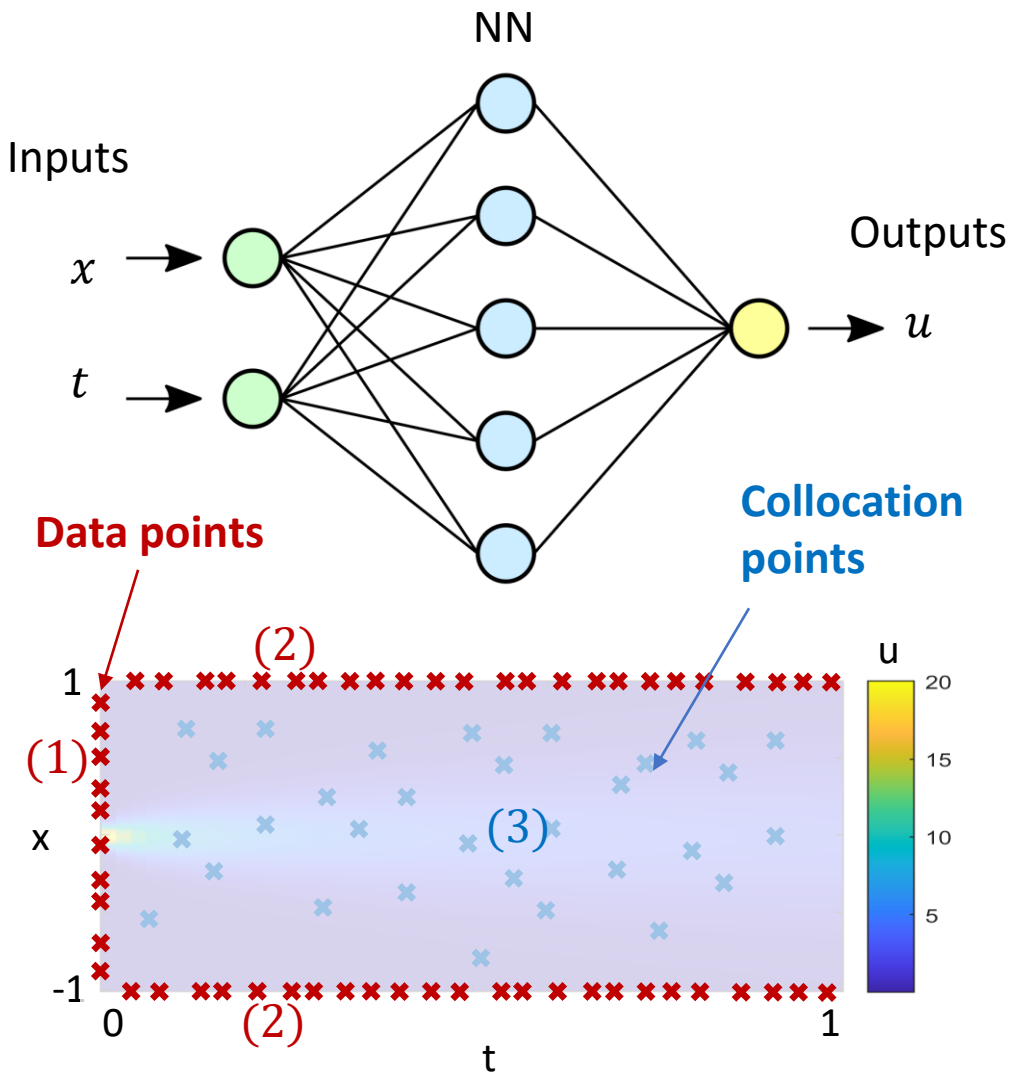
What would be the loss function?

$$\text{minimize} \quad \frac{1}{N}\sum_i^N \left|u_0^i - u_{pred}(t = 0, x_i)\right|^2 \qquad \text{(1) IC}$$

$$\text{minimize} \quad \frac{1}{M}\sum_j^M \left|u_{lb}^j - u_{pred}(t_j, x = -1)\right|^2 + \frac{1}{M}\sum_j^M \left|u_{ub}^j - u_{pred}(t_j, x = 1)\right|^2 \qquad \text{(2) BC}$$

GOAL: Find NN that minimizes the loss function

Physics-informed NN



Q: How to incorporate physics equation in the loss function?

$$(3)\quad \frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} \longrightarrow \frac{\partial u}{\partial t} - a\frac{\partial^2 u}{\partial x^2} = 0$$

Recall: NN(x,t) = u(x,t) is a smooth, analytical function

$\frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial x^2}$ can be directly calculated at the collocation points.

What would be the loss function?

Equation loss:

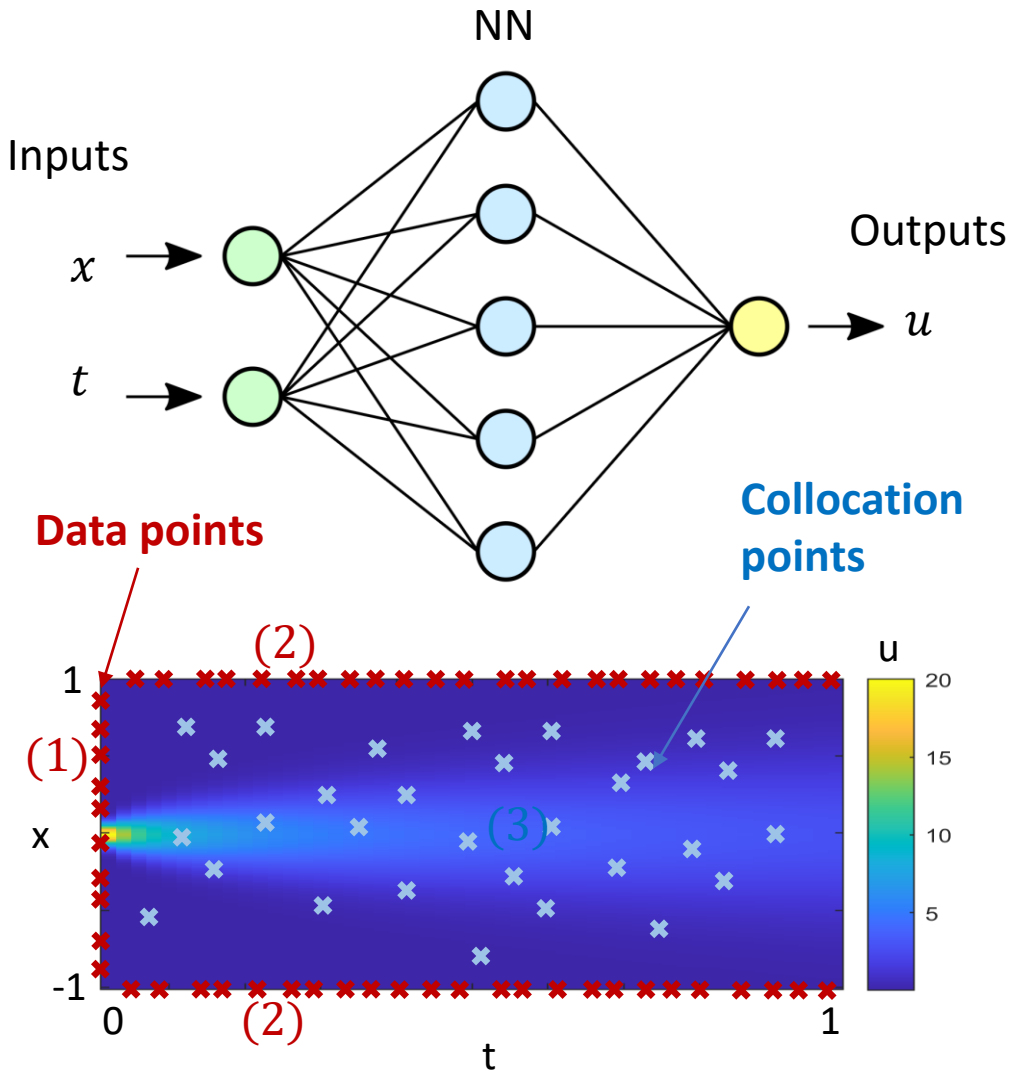
minimize $$\frac{1}{N_f}\sum_{k}^{N_f}\left|\frac{\partial u_{pred}^k}{\partial t} - a\frac{\partial^2 u_{pred}^k}{\partial x^2}\right|^2$$ (3) Eqn

GOAL: Find NN that minimizes the loss function

→ *solving for* $u(x,t)$ *satisfying* **(1) ICs + (2) BCs + (3) Eqn**

Raissi et al (2019)

Physics-informed NN



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u]$$

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

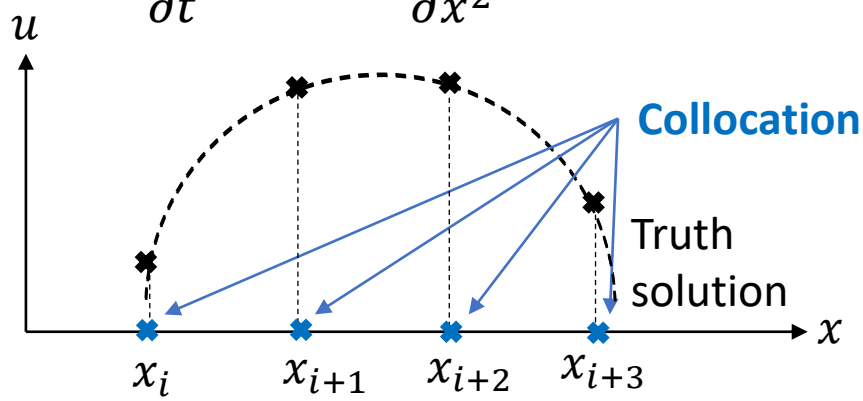
Data points

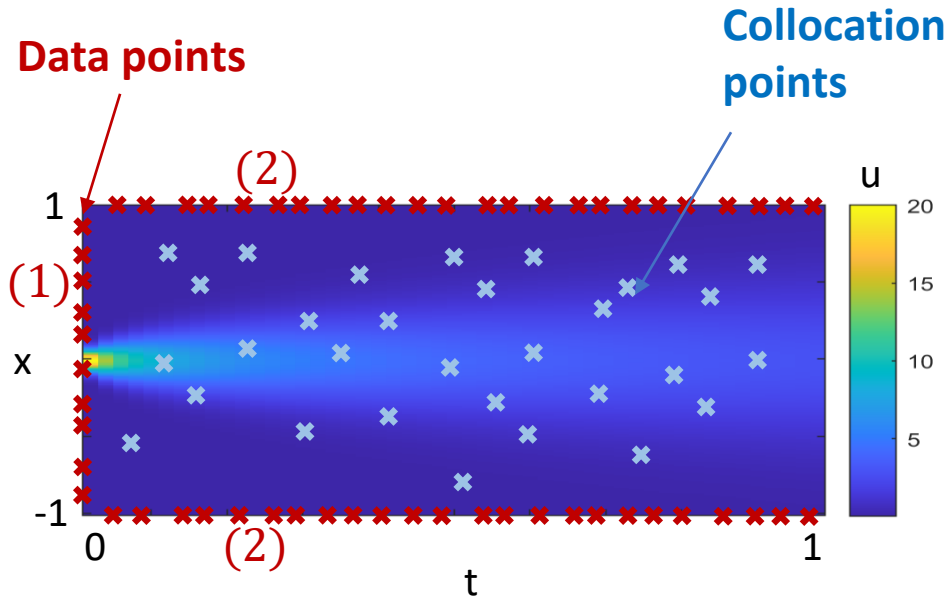
$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

PINN: searches for a curve that satisfies

$$\frac{\partial u(x_i)}{\partial t} - a\frac{\partial^2 u(x_i)}{\partial x^2} \approx 0 \text{ at the collocation pts}$$

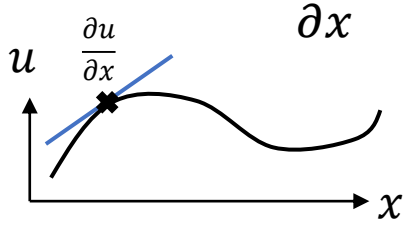




Automatic differentiation: differentiate NN output with respect to their input coordinates.

e.g. $\mathrm{NN(x)} = u = \sigma(w_2\sigma(w_1 x + b_1) + b_2)$,

$$\frac{\partial u}{\partial x} = \sigma'(z_2)w_2\sigma'(z_1)w_1$$



PINN $\frac{\partial u}{\partial t} + N(u, \lambda) = 0, \qquad x \in [-L, L], \qquad t \in [0\ T]$

- Application 1: Prediction of solution for a **well-posed forward problem** (this is what a traditional numerical solver can do)

  Given an eqn + BC + IC and parameters λ, what's the model prediction?

- Application 2: Prediction of solution when data is available within the domain but not at the IC, BC
  (difficult for a traditional numerical solver!)

  Given an eqn and parameters λ, what's the model prediction best describes the data?

- Application 3: Data-driven discovery of unknown parameters
  (difficult for a traditional numerical solver!)

  What are the parameters λ that best describe the data and the eqn?

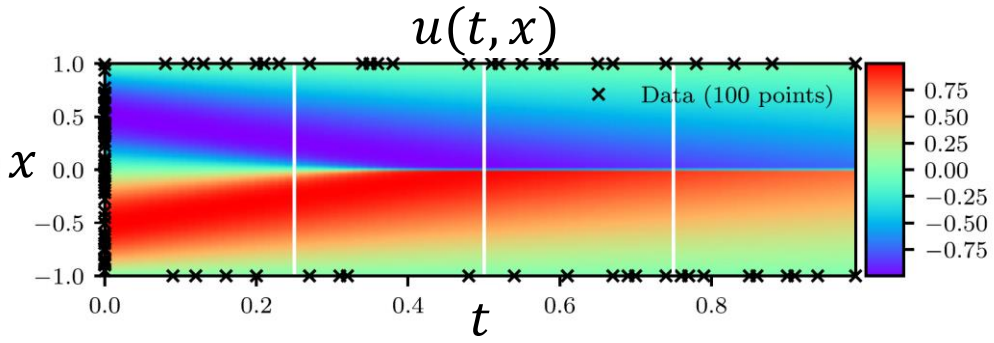
E.g., Burgers' equation (inference)

Problem statement

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$

$$IC: \; u(0, x) = -\sin(\pi x),$$

$$BC: \; u(t, -1) = u(t, 1) = 0.$$



Training data (from ground truth):

$$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u} \qquad N_u = 100$$

Collocation points:

$$\{t_f^i, x_f^i\}_{i=1}^{N_f} \qquad N_f = 10{,}000$$

Physics equations:

$$f := u_t + uu_x - (0.01/\pi)u_{xx}$$

Loss function:

Data loss

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Raissi et al (2019)

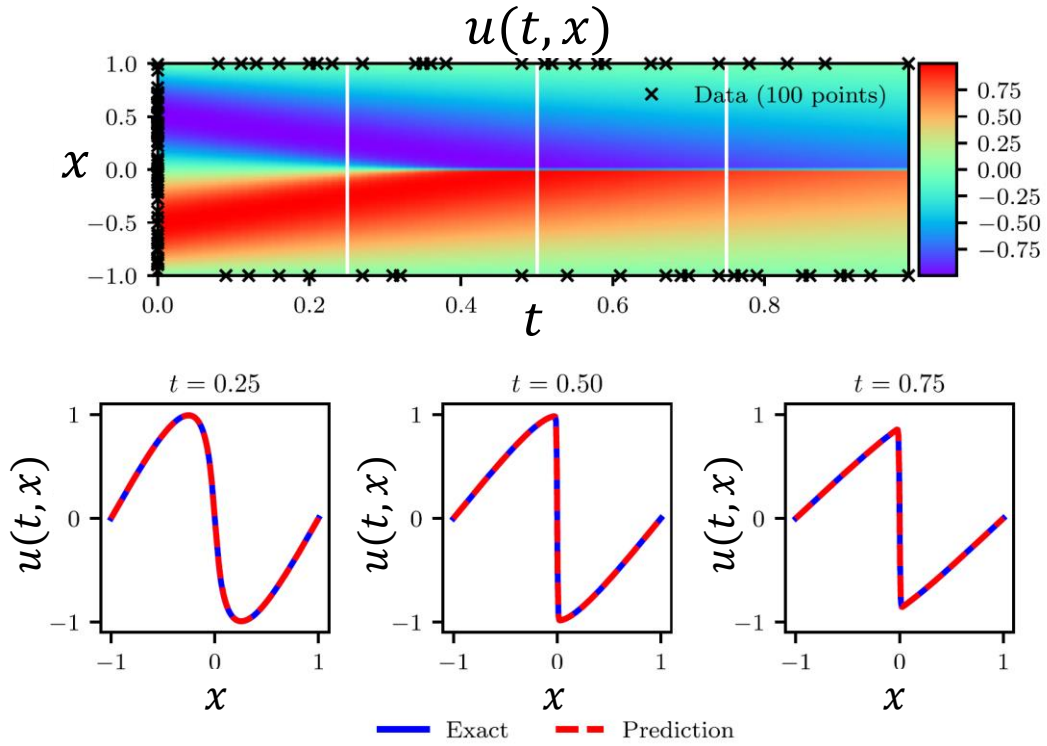
E.g., Burgers' equation (inference)

Problem statement

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$

$$IC: \; u(0, x) = -\sin(\pi x),$$

$$BC: \; u(t, -1) = u(t, 1) = 0.$$



Training data (from ground truth):

$$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u} \qquad N_u = 100$$

Collocation points:

$$\{t_f^i, x_f^i\}_{i=1}^{N_f} \qquad N_f = 10{,}000$$

Physics equations:

$$f := u_t + uu_x - (0.01/\pi)u_{xx}$$

Loss function:

Data loss

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

PINN $\frac{\partial u}{\partial t} + N(u, \lambda) = 0, \quad x \in [-L, L], \quad t \in [0\ T]$

- Application 1: Prediction of solution for a **well-posed forward problem** (this is what a traditional numerical solver can do)

  Given an eqn + BC + IC and parameters λ, what's the model prediction?

- Application 2: Prediction of solution when data is available within the domain but not at the IC, BC (difficult for a traditional numerical solver!)

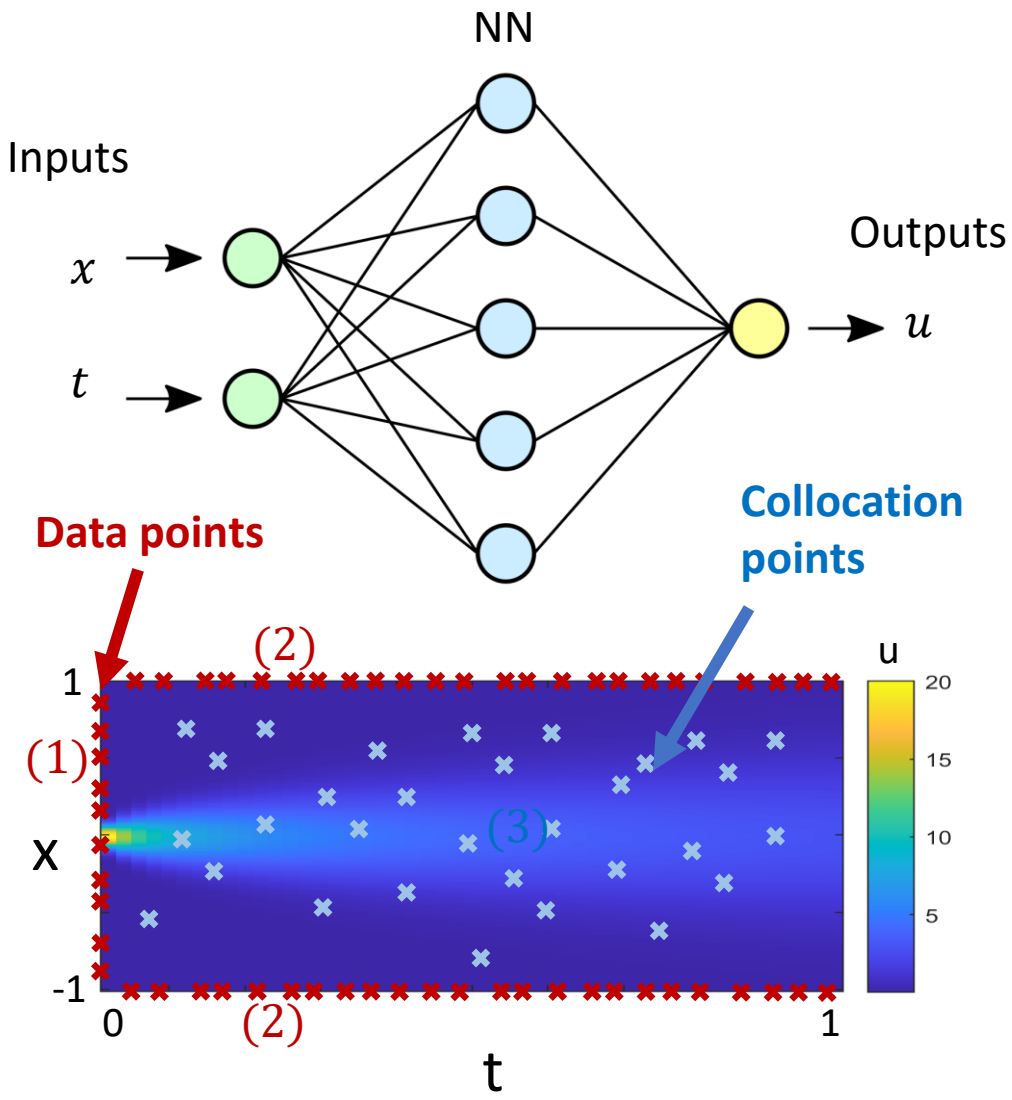
  Given an eqn and parameters λ, what's the model prediction best describes the data?

- Application 3: Data-driven discovery of **unknown parameters** (difficult for a traditional numerical solver!)

  What are the parameters λ that best describe the data and the eqn?

Raissi et al (2019)

Physics-informed NN



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u]$$

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

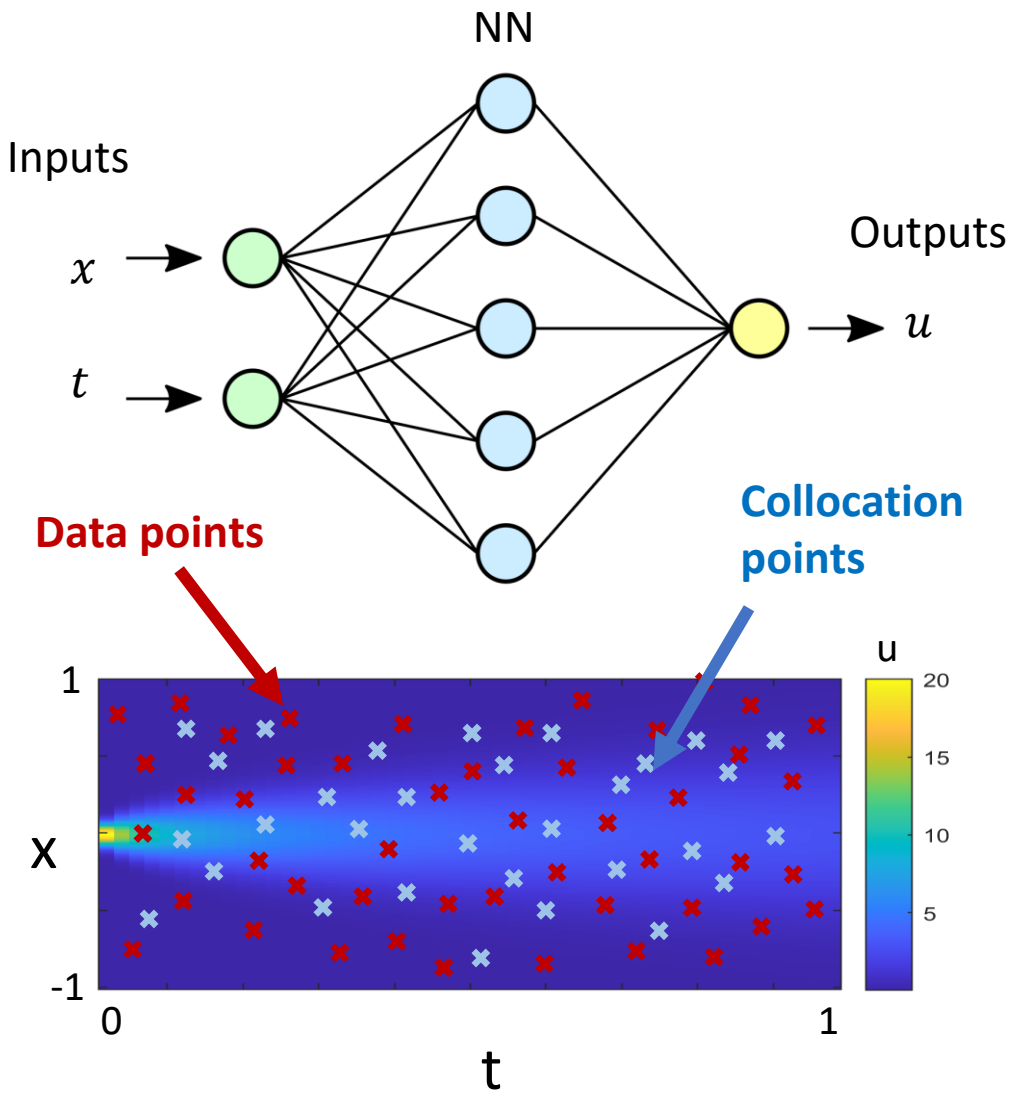
Data points

$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Raissi et al (2019)

Data-driven prediction of solution



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u]$$

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

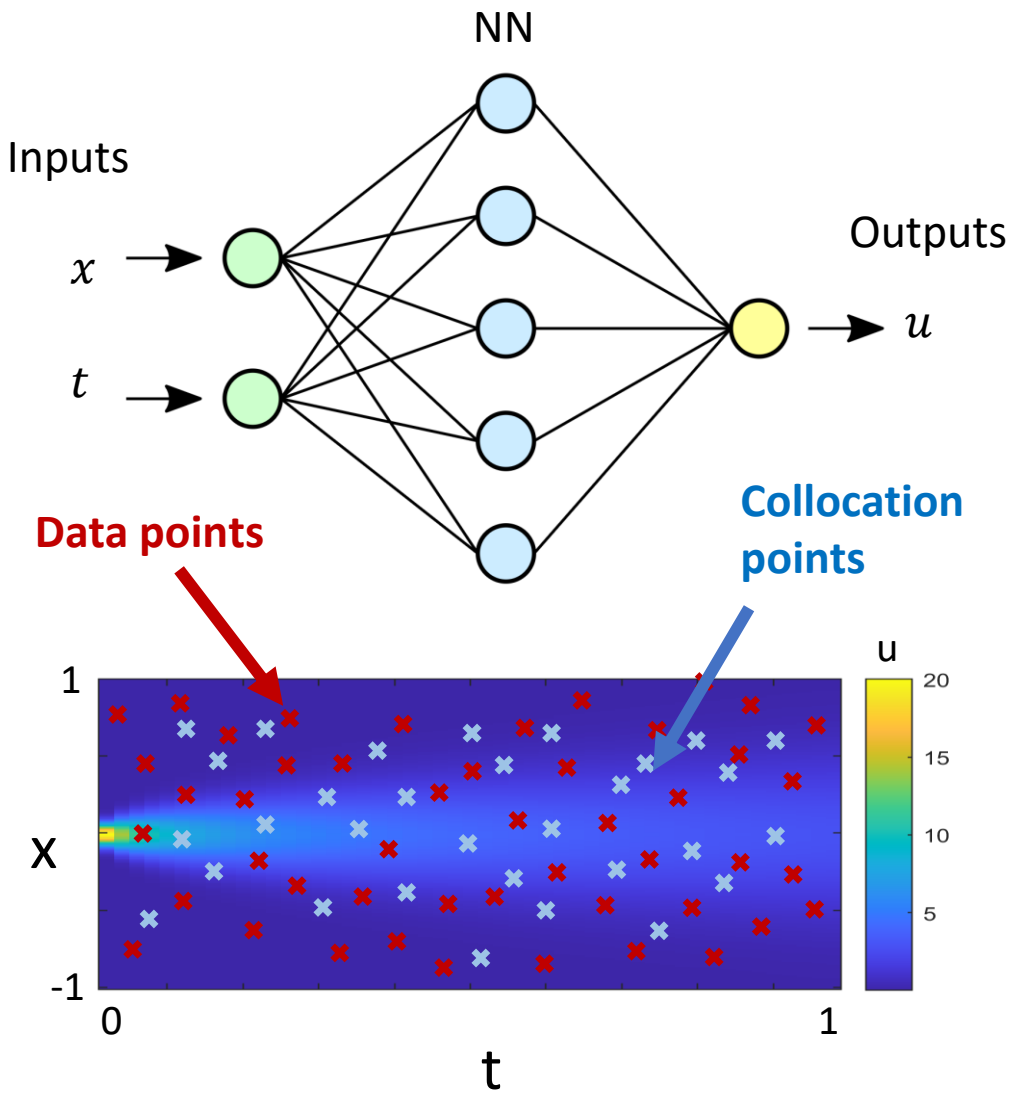
$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Data-driven discovery of unknown parameters



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u; \lambda] = 0, \; x \in \Omega, \; t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u; \lambda]$$

What are the parameters λ that best describe the data?

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

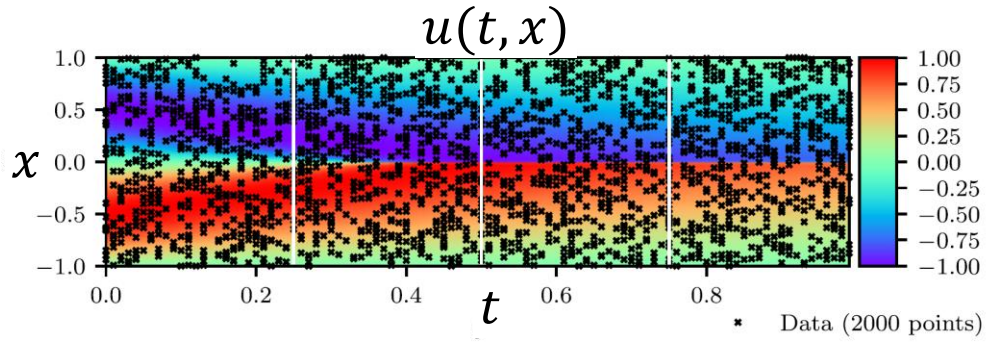
Raissi et al (2019)

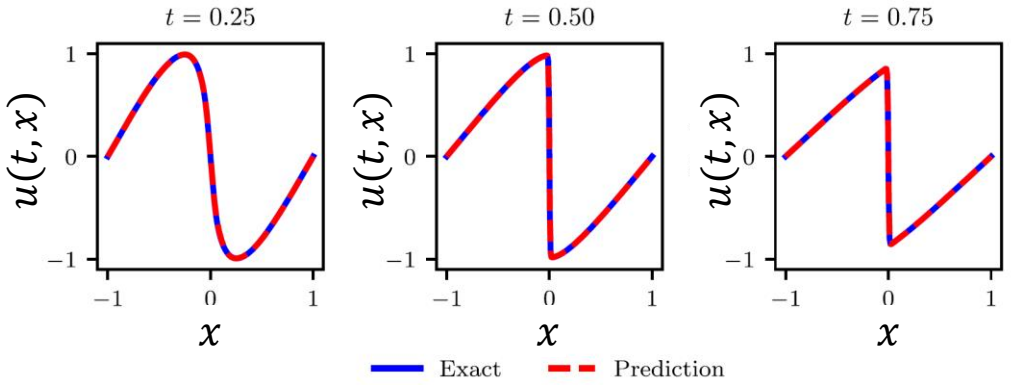
E.g., Burgers' equation (identification)

Given training data of u, t, x find $\boldsymbol{\lambda_1, \lambda_2}$

$$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0 \qquad x \in [-1, 1], \quad t \in [0, 1],$$

Correct PDE	$u_t + uu_x - 0.0031831 u_{xx} = 0$
Identified PDE (clean data)	$u_t + 0.99915 uu_x - 0.0031794 u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.00042 uu_x - 0.0032098 u_{xx} = 0$





Training data (from ground truth):

$$\{t_u^i, x_u^i, u^i\}_{i=1}^N \qquad N = 2{,}000$$

Collocation points:

$$\{t_u^i, x_u^i\}_{i=1}^N \qquad N = 2{,}000$$

Physics equations:

$$f := u_t + \lambda_1 u u_x - \lambda_2 u_{xx}$$

Loss function:

Data loss

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Raissi et al (2019)

E.g., Navier-Stokes equation

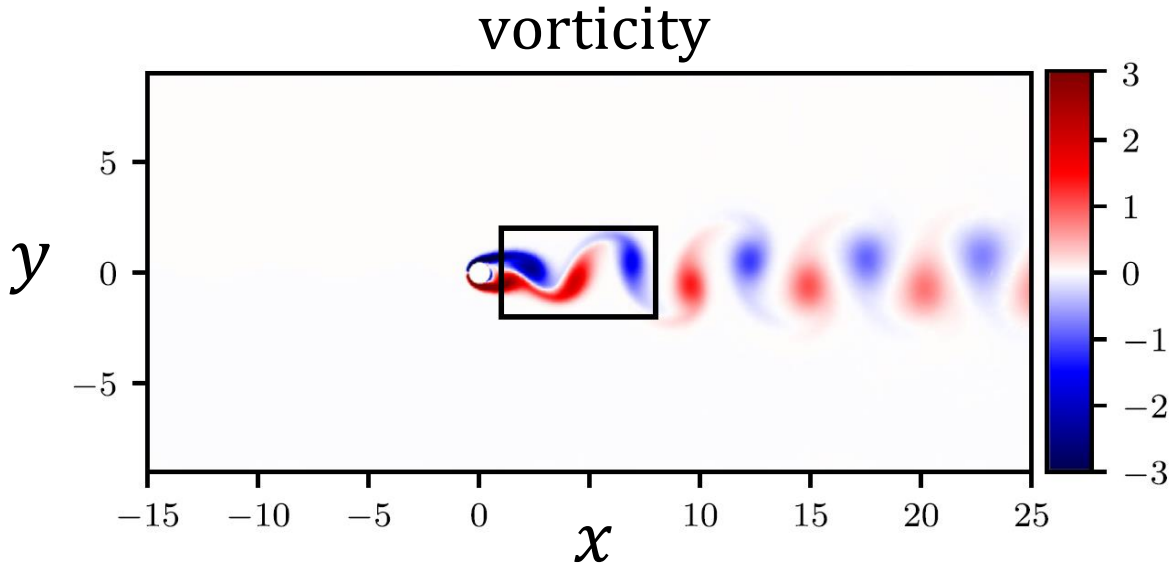
Given training data of u, v, find $\boldsymbol{\lambda_1, \lambda_2}$

Problem statement

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}),$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}),$$
$$u_x + v_y = 0 \longrightarrow u = \psi_y, \quad v = -\psi_x$$

λ_1, λ_2: unknown parameters to be identified.
ψ: the stream function.
u, v: velocities
p: pressure

Ground truth from numerical simulation:

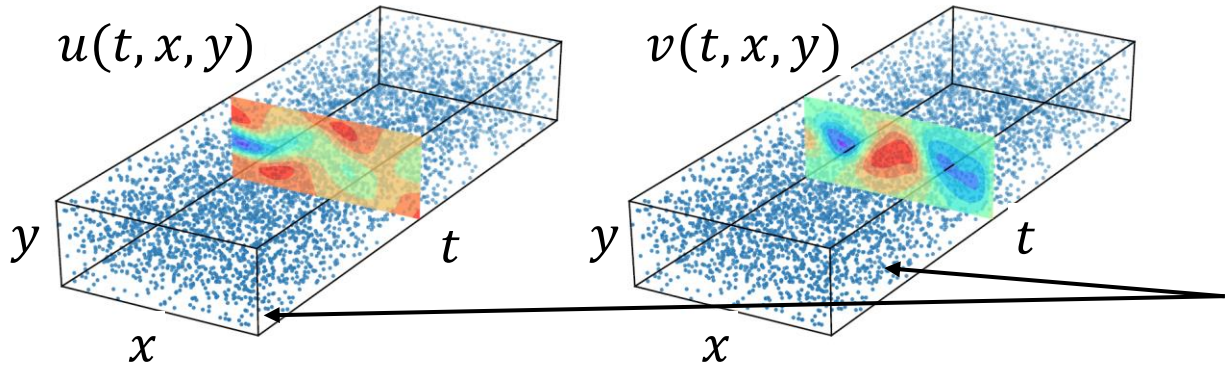


NN input: x, y, t
NN output: ψ, p
NN architecture: 9 layers 20 neurons per layer
Training parameters: weights, biases, λ_1, λ_2

Raissi et al (2019)

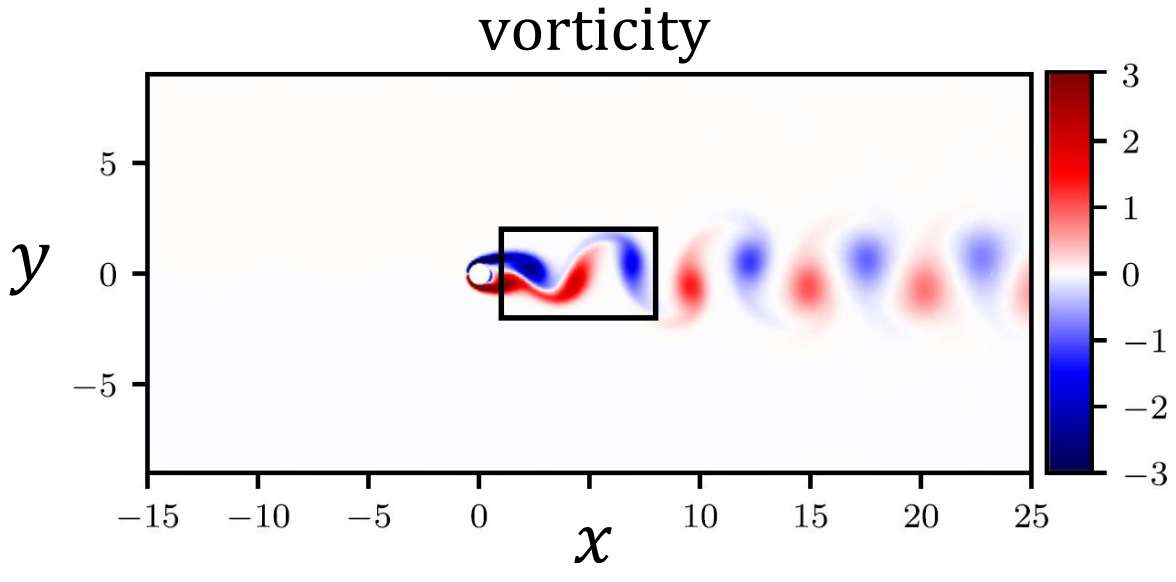
E.g., Navier-Stokes equation

Given training data of u, v, find $\boldsymbol{\lambda_1, \lambda_2}$



Training data: 5000 pts of u, v within the domain
Collocation points: 5000 pts within the domain

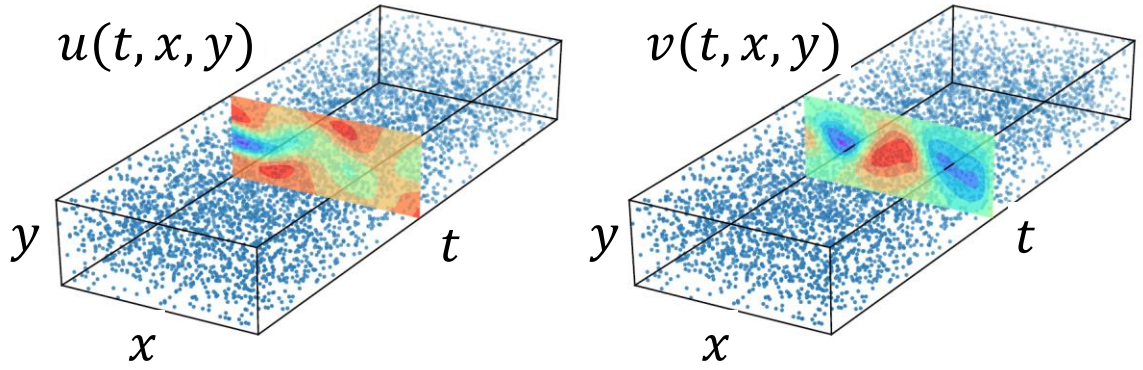
Ground truth from numerical simulation:



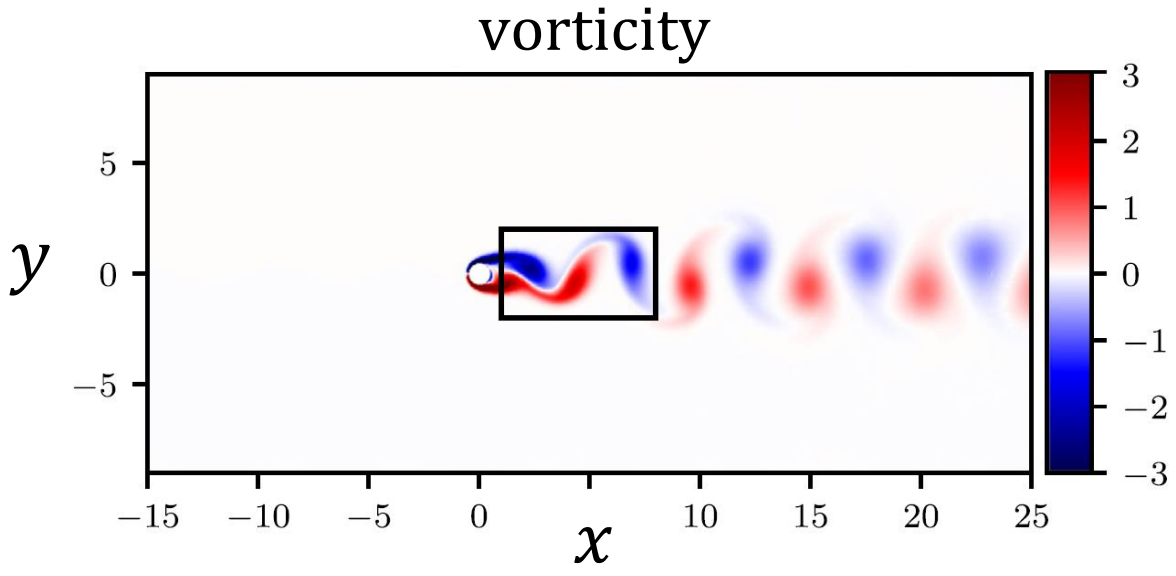
NN input: x, y, t
NN output: ψ, p
NN architecture: 9 layers 20 neurons per layer
Training parameters: weights, biases, λ_1, λ_2

Raissi et al (2019)

E.g., Navier-Stokes equation



Ground truth from numerical simulation:



Given training data of u, v, find $\boldsymbol{\lambda_1, \lambda_2}$

Training data (from ground truth):

$$\{t^i, x^i, y^i, u^i, v^i\}_{i=1}^N \qquad N = 5{,}000$$

Collocation points:

$$\{t^i, x^i, y^i\}_{i=1}^N \qquad N = 5{,}000$$

Physics equations:

$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy})$$
$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy})$$

Loss function:

$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2\right)$$

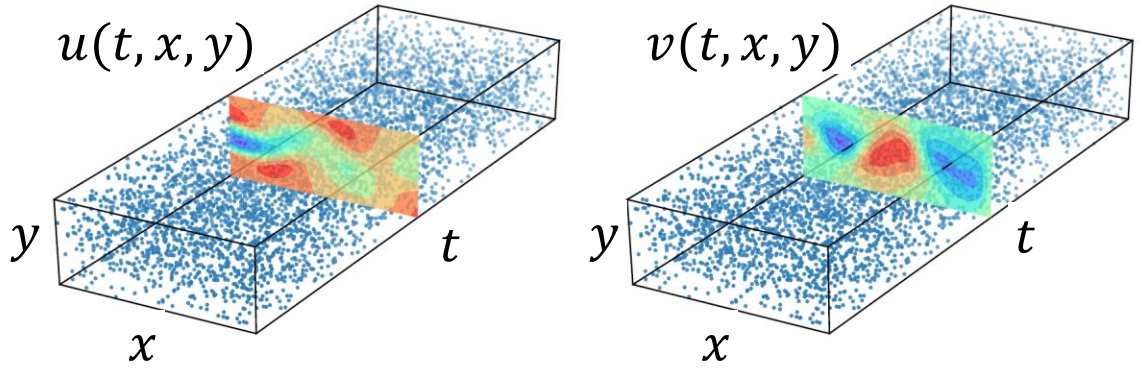
Data loss

$$+ \frac{1}{N}\sum_{i=1}^{N}\left(|f(t^i, x^i, y^i)|^2 + |g(t^i, x^i, y^i)|^2\right)$$

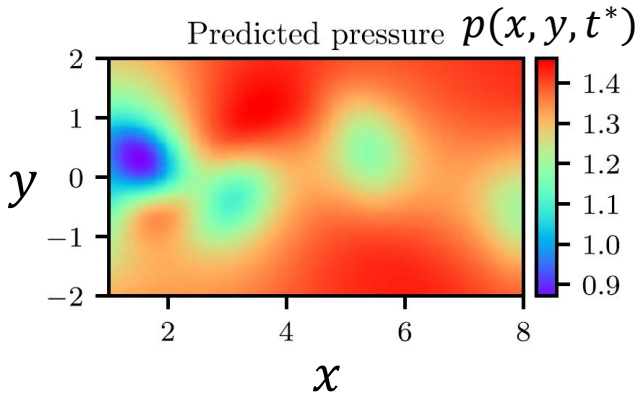
Equation loss

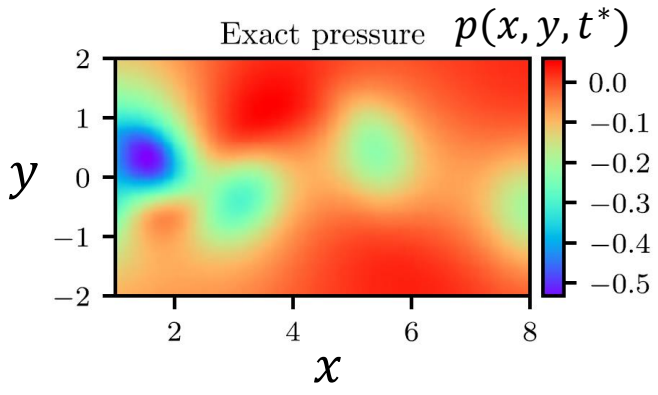
Raissi et al (2019)

E.g., Navier-Stokes equation



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$





Given training data of u, v, find $\boldsymbol{\lambda_1, \lambda_2}$

Training data (from ground truth):

$$\{t^i, x^i, y^i, u^i, v^i\}_{i=1}^N \qquad N = 5{,}000$$

Collocation points:

$$\{t^i, x^i, y^i\}_{i=1}^N \qquad N = 5{,}000$$

Physics equations:

$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy})$$
$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy})$$

Loss function:

$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2\right)$$

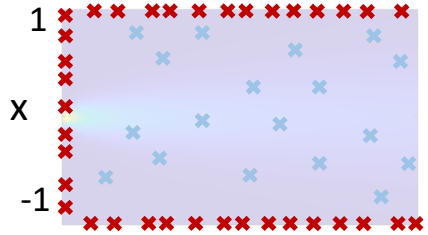
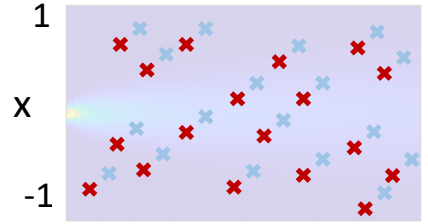
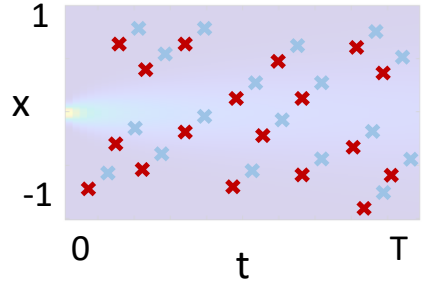
Data loss

$$+ \frac{1}{N}\sum_{i=1}^{N}\left(|f(t^i, x^i, y^i)|^2 + |g(t^i, x^i, y^i)|^2\right)$$

Equation loss

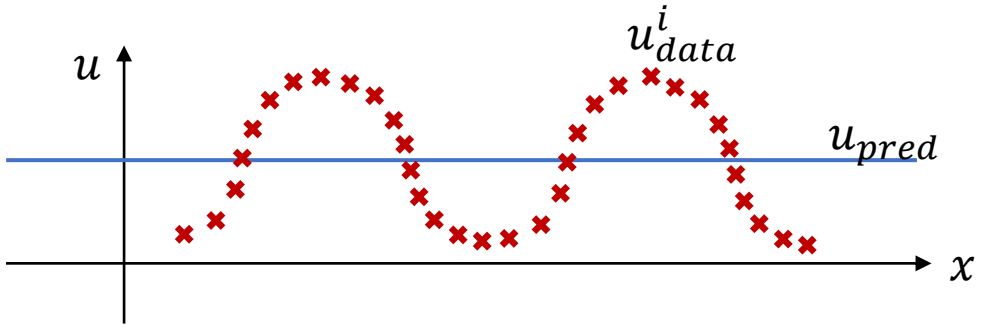
PINN

$$\frac{\partial u}{\partial t} + N(u, \lambda) = 0, \qquad x \in [-1, 1], \qquad t \in [0\ T]$$

Applications	Training data	NN prediction	
1. Prediction of solution of a **well-posed problem** (this is what a traditional numerical solver can do)	$IC, (BC)$	$u(x,t)$	
2. Prediction of solution when data is available within the domain but not at the IC, BC (difficult for a traditional numerical solver)	t_i, x_i, u_i from i = 0 to m $x_i \in [-1, 1], t_i \in [0\ T]$	$u(x,t)$	
3. Data-driven discovery of **unknown parameters** (difficult for a traditional numerical solver)	t_i, x_i, u_i from i = 0 to m $x_i \in [-1, 1], t_i \in [0\ T]$	$u(x,t)$ λ	

Pause and Ponder

- Can we replace $\sum_i^N \left|u_{data}^i - u_{pred}^i\right|^2$ with $\sum_i^N \left(u_{data}^i - u_{pred}^i\right)$ in the cost function?



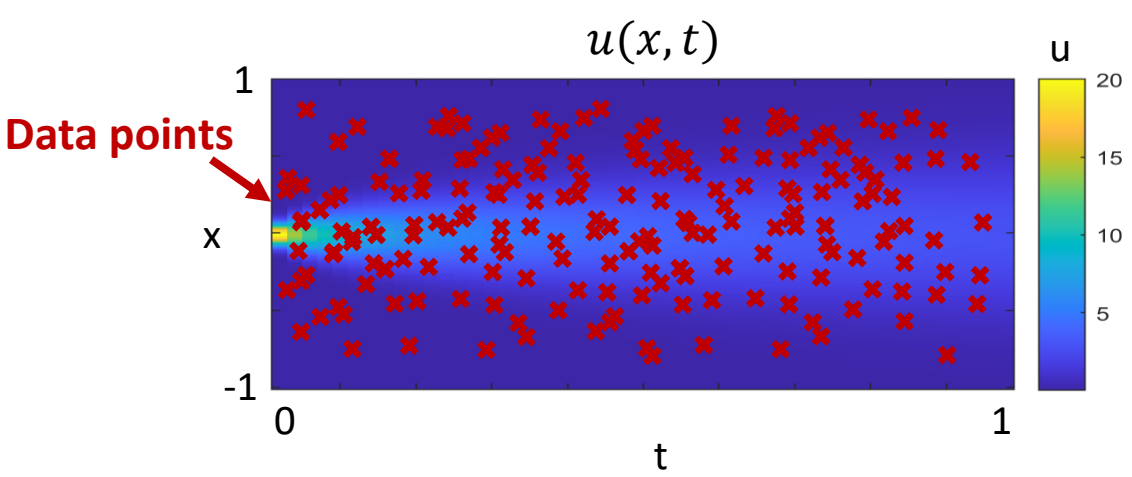
$\sum_i^N \left(u_{data}^i - u_{pred}^i\right) \approx 0$

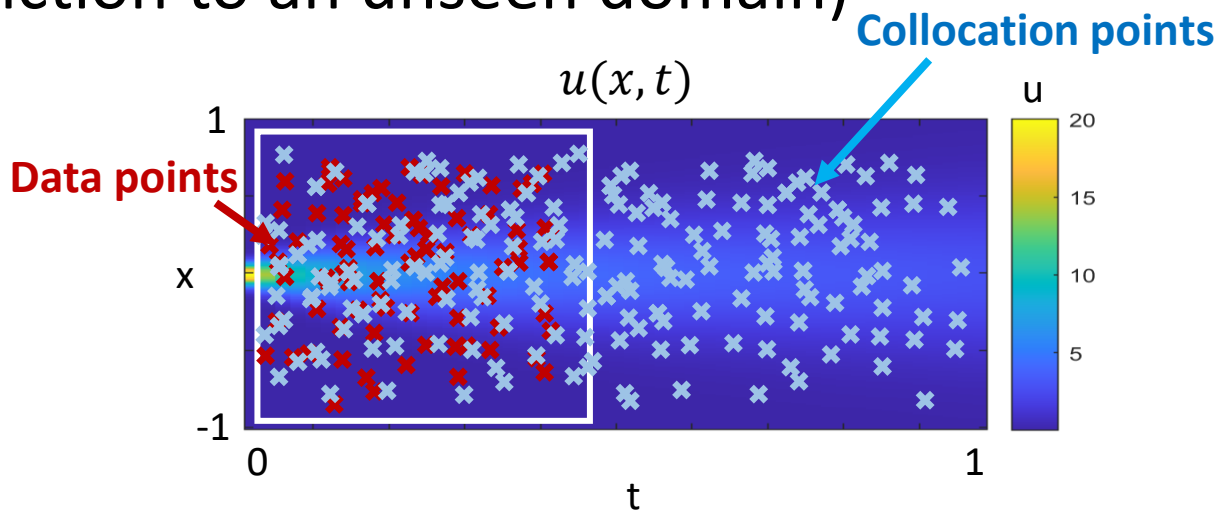
$\sum_i^N \left|u_{data}^i - u_{pred}^i\right|^2$ can be large!

- Is PINN using supervised or unsupervised learning?
  - The training at collocation points is unsupervised learning!
  - The training at data points is supervised learning

When can you use only data to train NN without physics loss, and still get a good NN prediction (i.e. consistent with the physics)?

- When the data is perfect without noise and available everywhere (physics emulator)
- When the prediction is within the same {t,x} domain as the training data (no need to generalize the prediction to an unseen domain)





When can you use only data to train NN without physics loss, and still get a good NN prediction (i.e. consistent with the physics)?

- When the data is perfect without noise and available everywhere (physics emulator)
- When the prediction is within the same {t,x} domain as the training data (no need to generalize the prediction to an unseen domain)

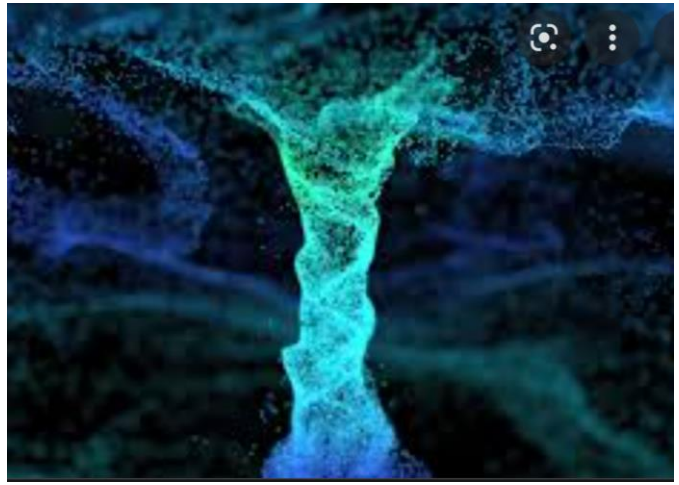
- **The point of PINN is that its prediction can be generalized to a domain without observations!**

Three applications of PINN

- Inferring ice viscosity

- Hurricane data assimilation

- Finding singularities



Ice dynamics and sea-level rise

credit: NASA/Goddard Space Flight Center Scientific Visualization Studio

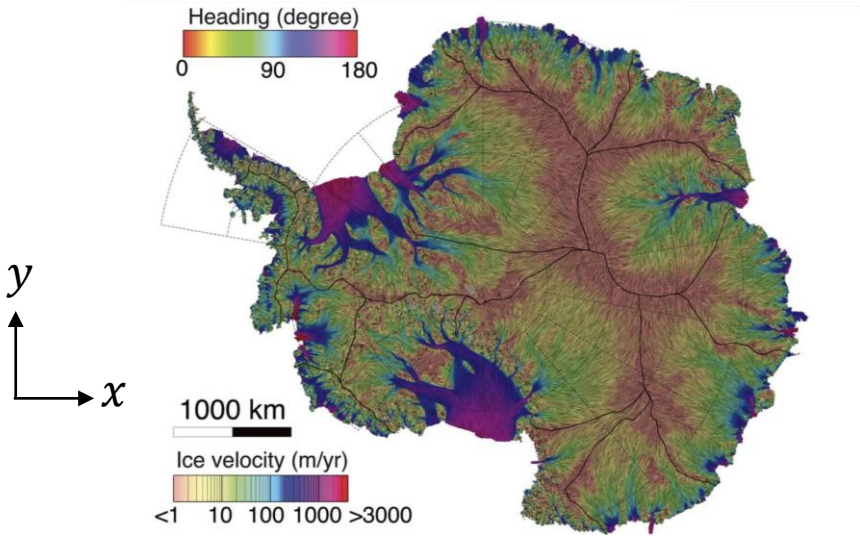
Prediction of ice dynamics relies on the viscosity structure of ice sheet, which is not easily measurable.

How to "learn" the ice effective viscosity from data?

Data:

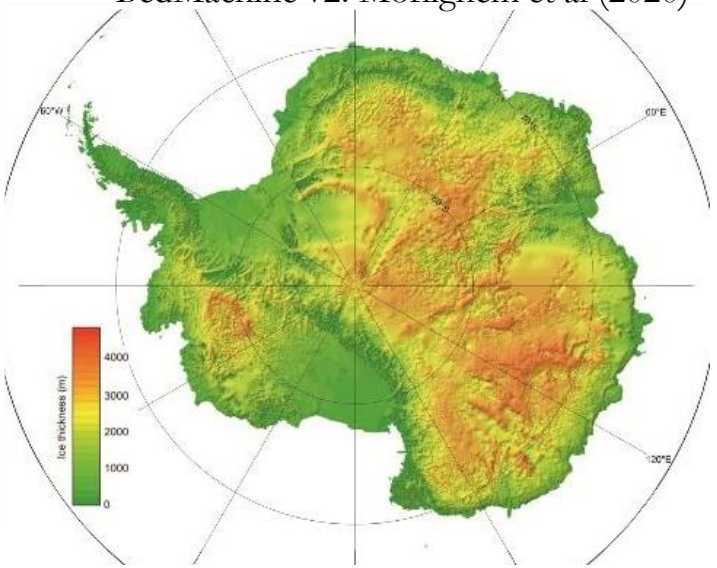
Ice velocity u

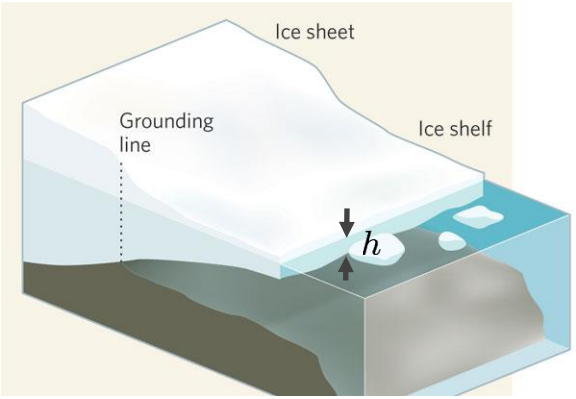
MEaSUREs v1: Mouginot et al (2019)



Ice thickness h

BedMachine v2: Morlighem et al (2020)





Equation:

Ice effective viscosity: impossible to measure

$$\frac{\partial}{\partial x}\left(4\mu h\frac{\partial u}{\partial x}+2\mu h\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial y}\left(\mu h\frac{\partial u}{\partial y}+\mu h\frac{\partial v}{\partial x}\right)=\rho_i g\left(1-\frac{\rho_i}{\rho_w}\right)h\frac{\partial h}{\partial x}$$

$$\underbrace{\frac{\partial}{\partial y}\left(4\mu h\frac{\partial v}{\partial y}+2\mu h\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial x}\left(\mu h\frac{\partial u}{\partial y}+\mu h\frac{\partial v}{\partial x}\right)}_{\text{viscous stress}}=\underbrace{\rho_i g\left(1-\frac{\rho_i}{\rho_w}\right)h\frac{\partial h}{\partial y}}_{\text{gravity}}$$

Morland (1987), MacAyeal (1989)

u, v: horizontal velocity

ρ_i: ice density

ρ_w: sea water density

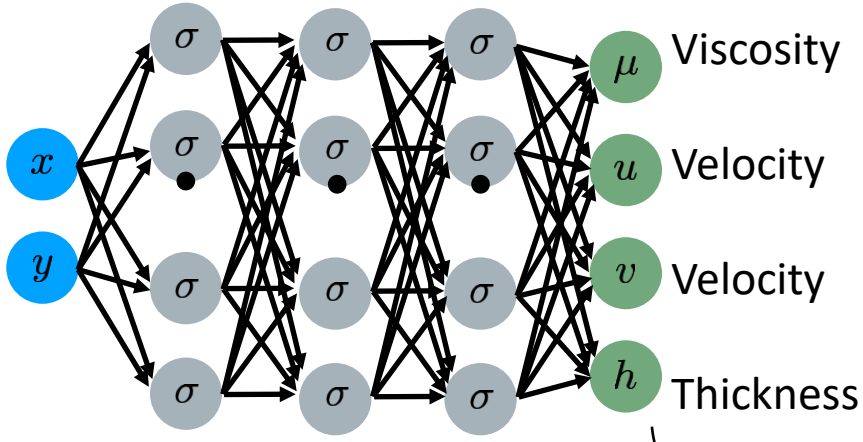
h : ice thickness

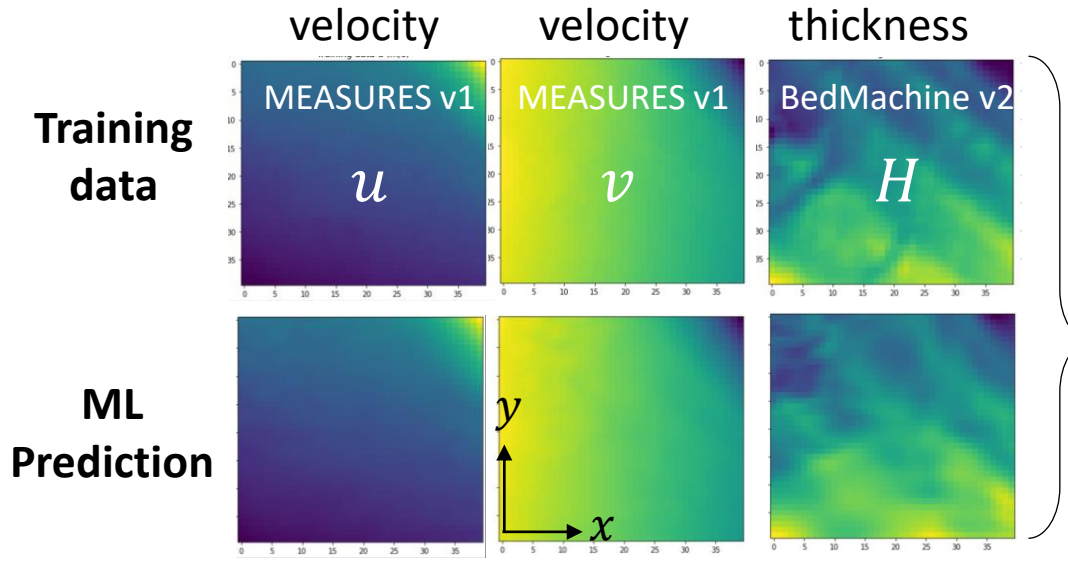
} measurable

How to “learn” the ice effective viscosity from data?

Yongji Wang, postdoc

A deep neural network





$\text{Loss}_{\text{DATA}}$

Governing equations

$$e_1 = \frac{\partial}{\partial x}\left(4\mu h\frac{\partial u}{\partial x} + 2\mu h\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial y}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - h\frac{\partial h}{\partial x}$$

$$e_2 = \frac{\partial}{\partial y}\left(4\mu h\frac{\partial v}{\partial y} + 2\mu h\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - h\frac{\partial h}{\partial y}$$

Loss_{EQN}

Loss function

Update neural net parameters

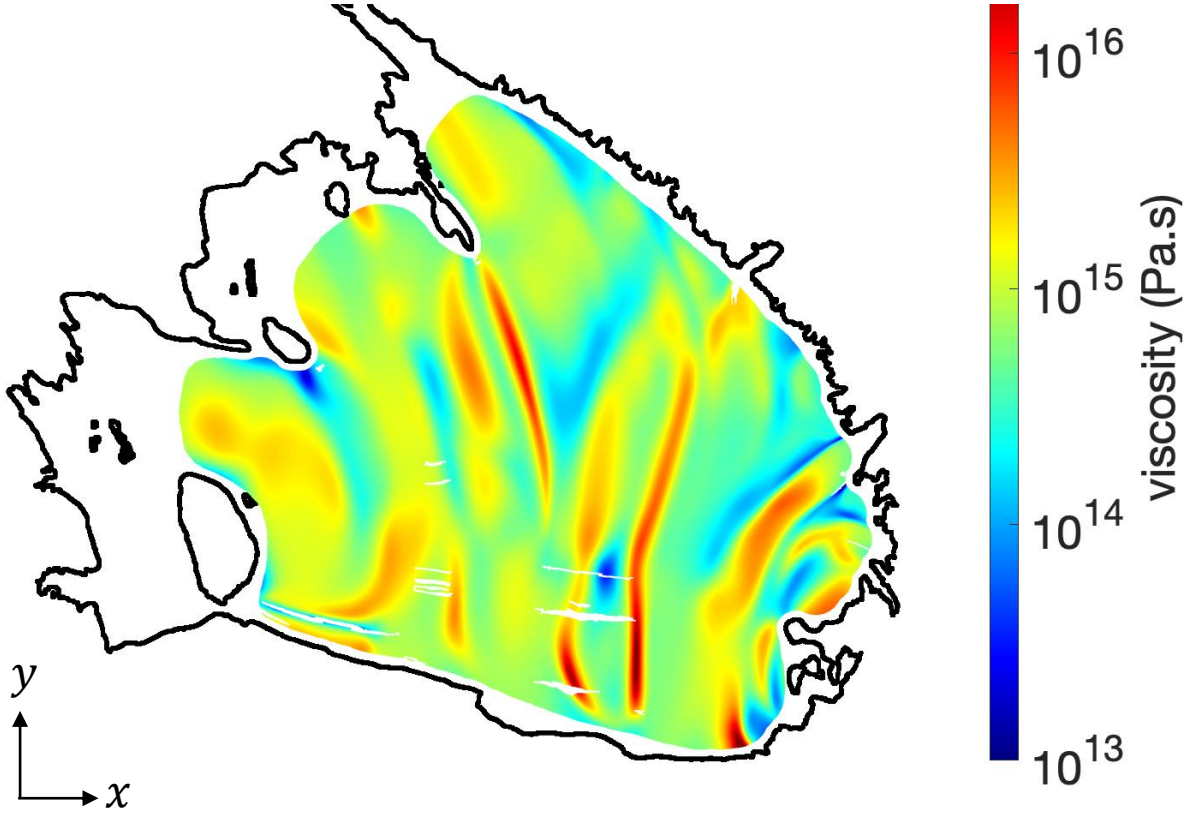
$$\text{Loss} = (1-\gamma)\text{Loss}_{\text{DATA}} + \gamma\text{Loss}_{\text{EQN}} \qquad \gamma \in [0,1]$$

Iwasaki and Lai, JCP (2023)
Wang, Lai, Prior, Cowen-Breen, under review.

Inversion of ice **effective viscosity** via PINNs

Yongji Wang, postdoc

Ice effective viscosity μ



Wang, Lai, Prior, Cowen-Breen, under review.

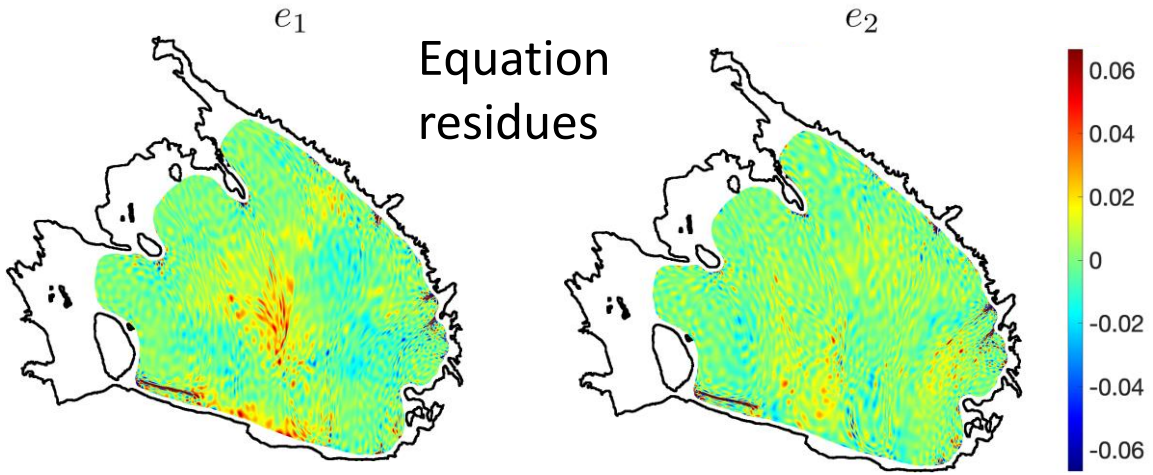
Difficult to measure. Prediction is only possible due to equations!

Governing ice-shelf equations:

$$e_1 = \frac{\partial}{\partial x}\left(4\mu h\frac{\partial u}{\partial x} + 2\mu h\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial y}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - h\frac{\partial h}{\partial x}$$

$$e_2 = \frac{\partial}{\partial y}\left(4\mu h\frac{\partial v}{\partial y} + 2\mu h\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - h\frac{\partial h}{\partial y}$$

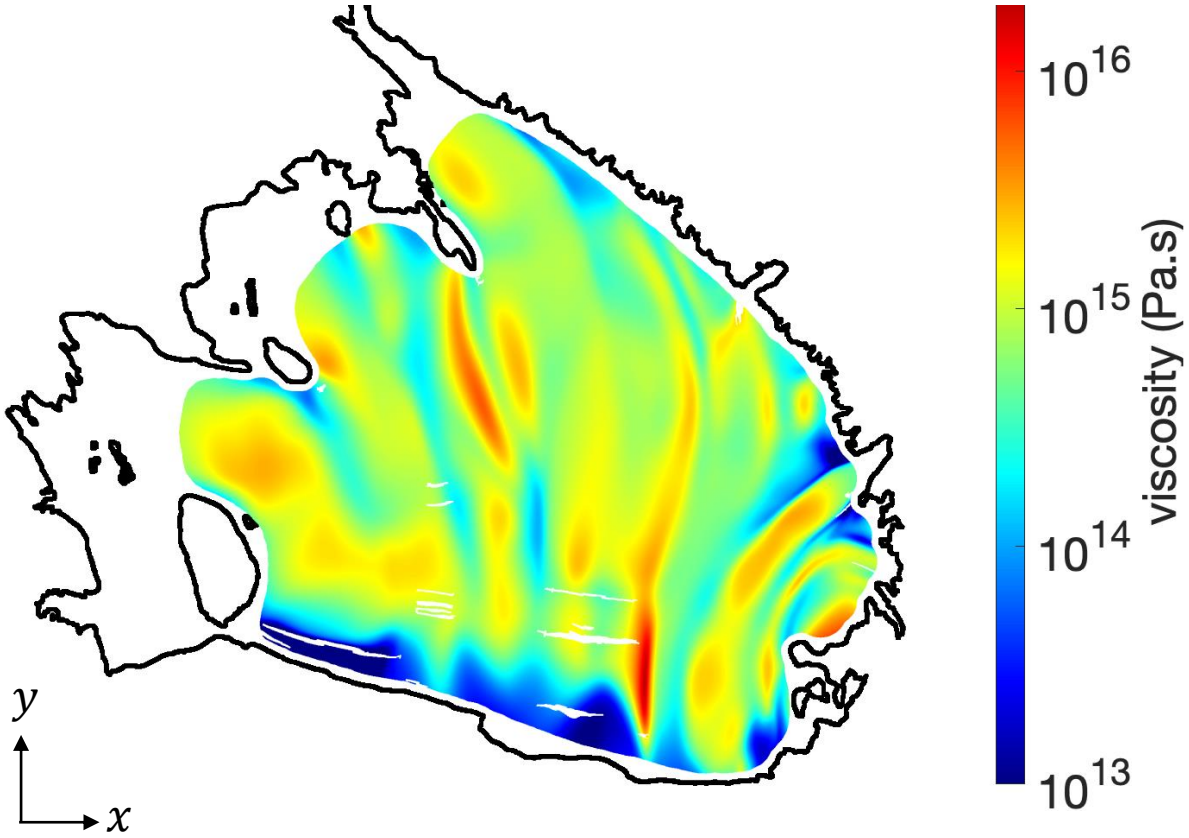
Equation residues



e1 and e2 are both 2 orders of magnitude smaller than each term in the equation

Anisotropic viscosity models fit the data better

Ice effective viscosity μ_h



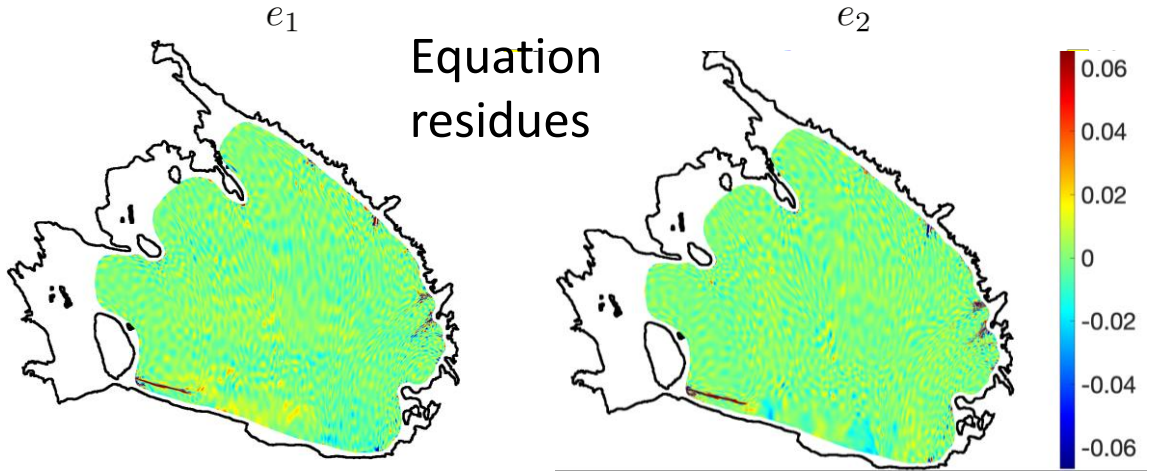
~~Assumption in equation: Isotropic viscosity~~

Anisotropic ice-shelf equations:

$$e_1 = \frac{\partial}{\partial x}\left(2\mu_h h\frac{\partial u}{\partial x} + 2\mu_v h\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]\right) + \frac{\partial}{\partial y}\left(\mu_h h\left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right]\right) - \rho_i g\left(1 - \frac{\rho}{\rho_w}\right)h\frac{\partial h}{\partial x}$$

$$e_2 = \frac{\partial}{\partial y}\left(2\mu_h h\frac{\partial v}{\partial y} + 2\mu_v h\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]\right) + \frac{\partial}{\partial x}\left(\mu_h h\left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right]\right) - \rho_i g\left(1 - \frac{\rho}{\rho_w}\right)h\frac{\partial h}{\partial y}$$

Equation residues



e1 and e2 are further reduced.

Wang, Lai, Prior, Cowen-Breen, under review.

Three applications of PINN

- Inferring ice viscosity
- Hurricane data assimilation
- Finding singularities

Hurricane data assimilation

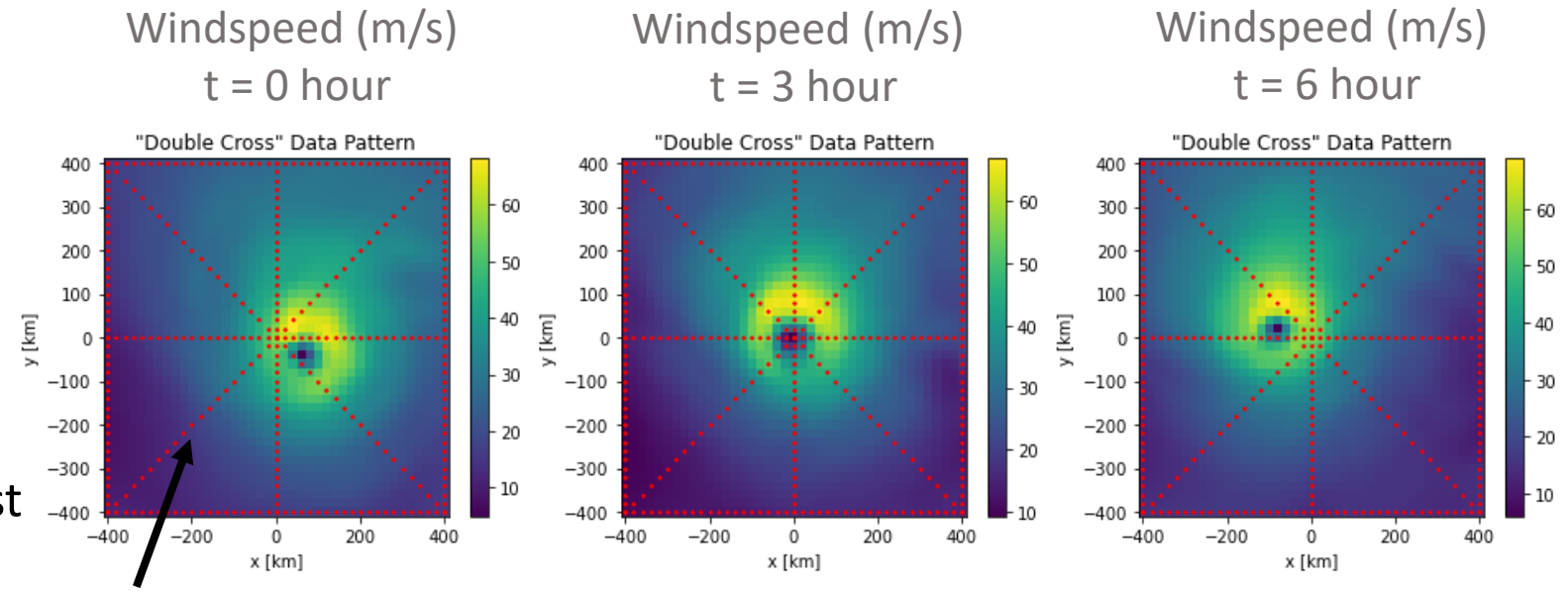
Ryan Eusebi ‘22 (CS)

- **Hurricane Hunter Data**
  Horizontal transects through the eye of the storm

- **Training data (from climate model)**

  Poorly spatial and temporal resolution!
  3 time stamps, provided at the red dots

Hurricane track and intensity forecast relies on accurate knowledge of a hurricane’s initial state



Training data

Physics-informed loss function

Ryan Eusebi ‘22 (CS)

$$\text{Loss} = (1-\gamma)\text{Loss}_{DATA} + \gamma\text{Loss}_{EQN} \qquad \gamma \in [0,1]$$

$$\text{Loss}_{DATA} = (u_{pred} - u_{actual})^2 + (v_{pred} - v_{actual})^2 + (\Phi_{pred} - \Phi_{actual})^2$$

$$\text{Loss}_{EQN} = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial \omega}{\partial p}\right)^2$$

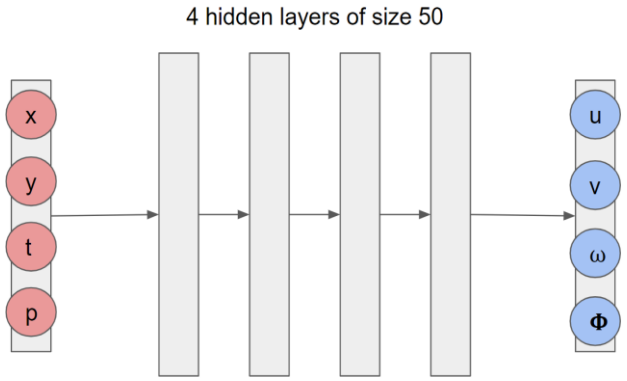
$$+ \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \omega\frac{\partial u}{\partial p} - (f_o + \beta y)v + \frac{\partial \Phi}{\partial x}\right)^2$$

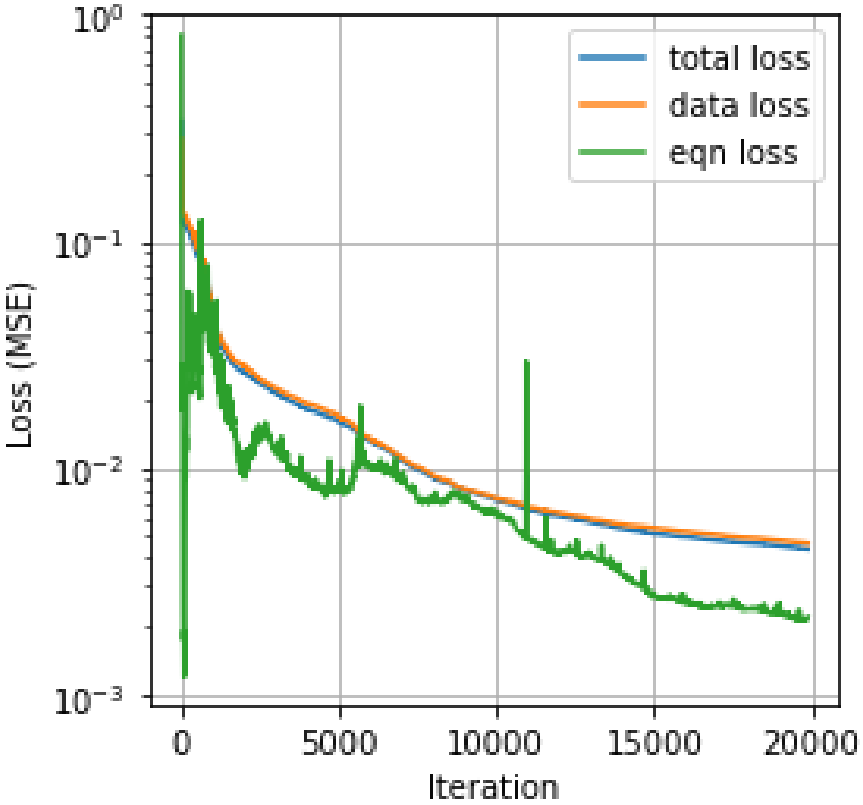
$$+ \left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \omega\frac{\partial v}{\partial p} + (f_o + \beta y)u + \frac{\partial \Phi}{\partial y}\right)^2$$

Conservation of mass

Conservation of momentum

NN architecture



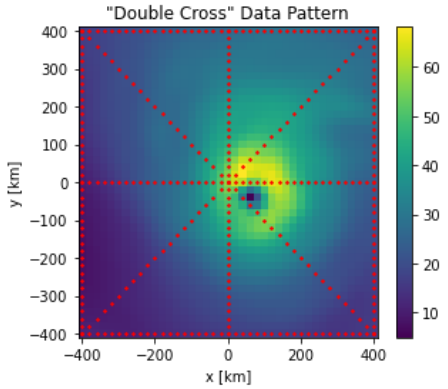


Eusebi, Vecchi, Lai, Tong, *Nature Commun. Earth Environ* (2024)

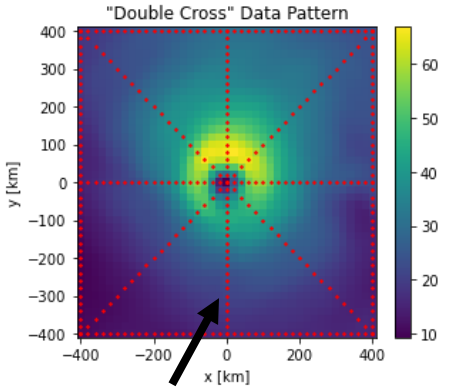
Hurricane wind fields prediction with sparse data

Training data- red dots (3 hour resolution)

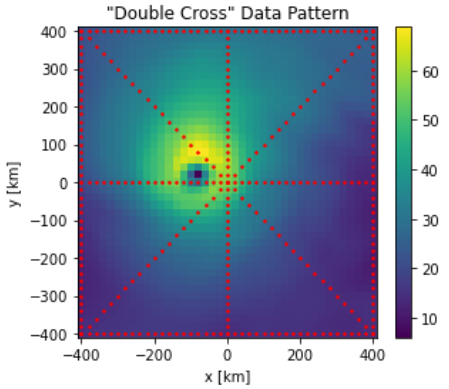
Windspeed (m/s)
t = 0 hour



Windspeed (m/s)
t = 3 hour

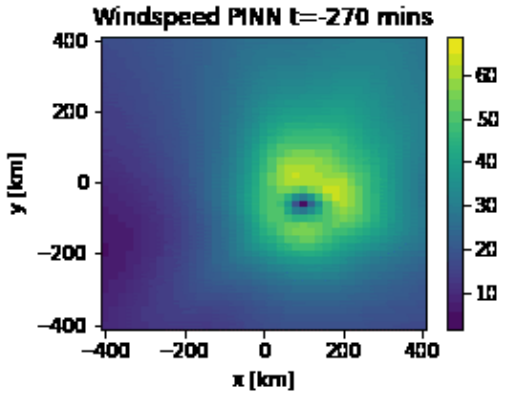


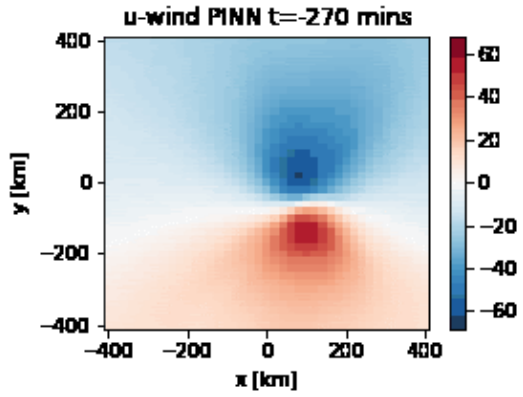
Windspeed (m/s)
t = 6 hour

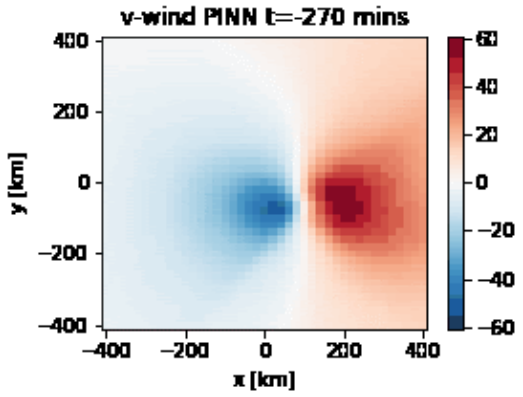


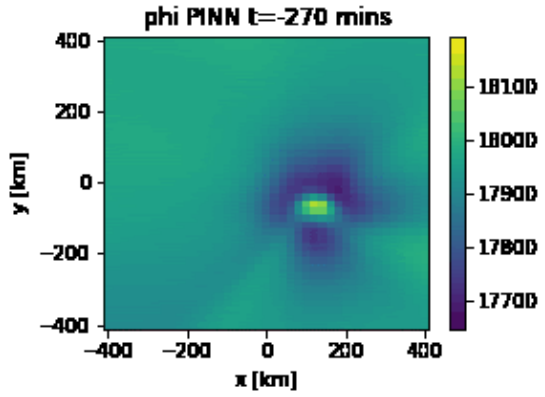
Training data

PINN output (10 min resolution)



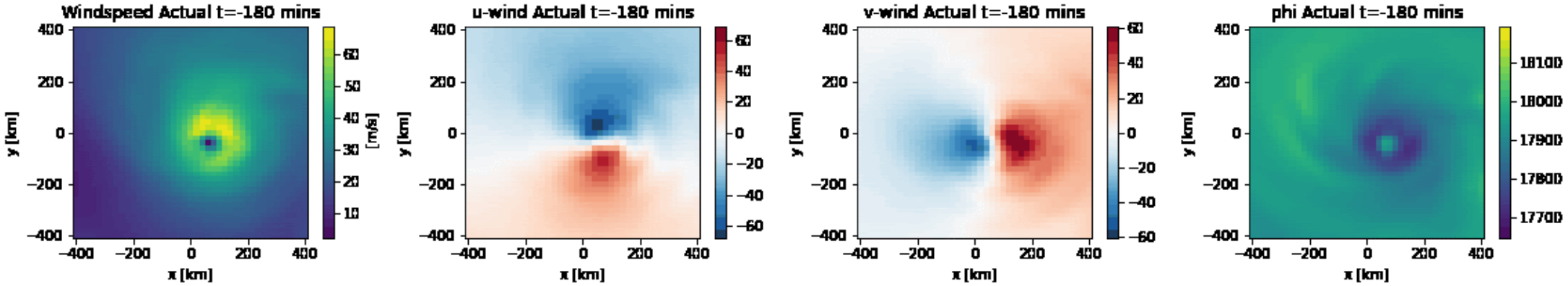




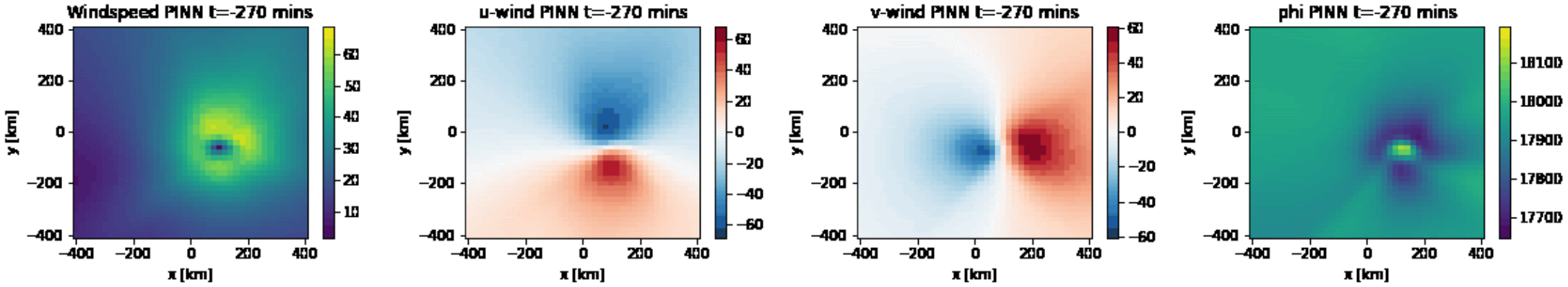


Hurricane wind fields prediction with sparse data

Ground truth (3 hour resolution; climate model)

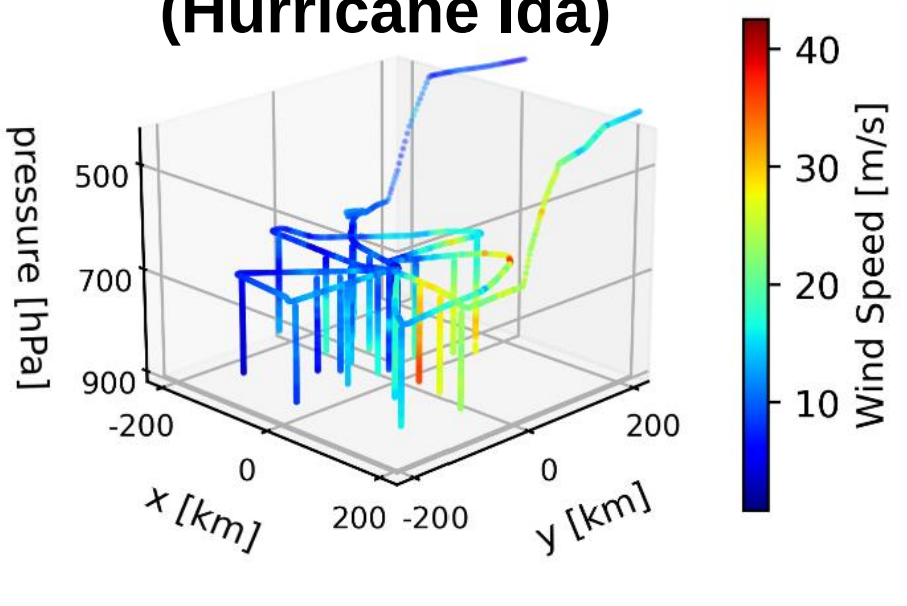


PINN output (10 min resolution)

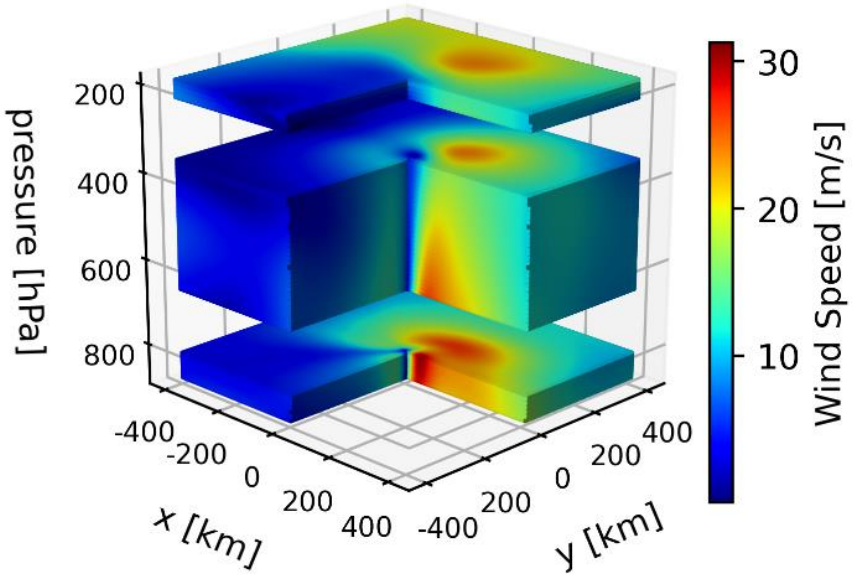


3D wind reconstruction with real sparse data

Training data (Hurricane Ida)



PINN wind speed prediction



- PINN: 50 min on A100 GPU
- Established data assimilation (e.g. HAFS model): 85 hours on CPU

100 times faster! Similar accuracy.

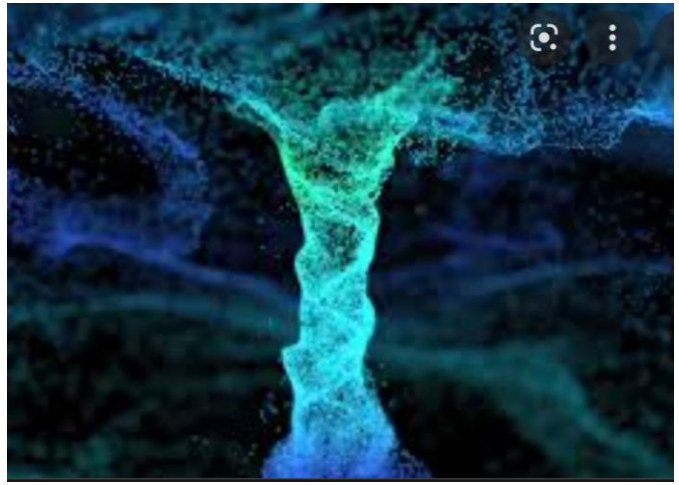
Eusebi, Vecchi, Lai, Tong, *Nature Commun. Earth Environ* (2024)

Three applications of PINN

- Inferring ice viscosity

- Hurricane data assimilation

- Finding singularities



A Millennium Prize problem

But there are many unsolved questions concerning the fundamental behavior of the Navier-Stokes equation.

Does the Navier-Stokes equation always exist a unique and smooth (infinitely differentiable) solution at any times?

Prove or give a counter-example of the following statement:

In three space dimensions and time, given an initial velocity field, there exists a vector velocity and a scalar pressure field, which are both smooth and globally defined, that solve the Navier–Stokes equations.

- One pathway: If we find a solution to the Navier Stokes that "blows up" at finite time, that would be a counter-example of the above statement.

The Euler equation…

Steady self-similar equations for axisymmetric Euler with boundary

$$
\begin{aligned}
\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Omega &= \Phi \\
(2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Phi &= -\partial_{y_1} U_2 \Psi \\
(2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Psi &= -\partial_{y_2} U_1 \Phi \\
\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad & \operatorname{div} \mathbf{U} = 0
\end{aligned}
$$

The goal for NN:
1. Find the solutions
$\boldsymbol{U}(y_1, y_2), \Omega(y_1, y_2), \Psi(y_1, y_2), \Phi(y_1, y_2)$
2. Find the self-similar exponent λ

In addition, we impose

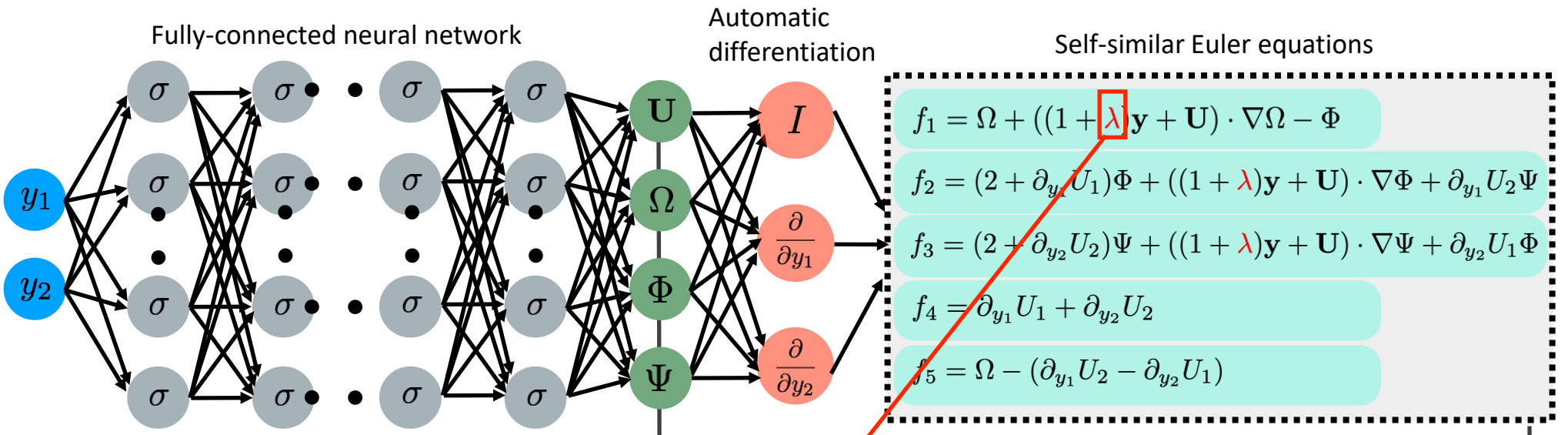
1. U_1, Φ, Ω are odd in y_1
2. U_2, Ψ are even in y_1

→ Symmetry of the solutions

3. $U_2(y_1, 0) = 0$ → No-penetration condition

PDE-constrained NN

6 hidden layer; 30 units per layer



$$q(\mathbf{y}) = \sum_{j=1} w_{lk}^{(n)} \sigma\left(\sum_{i=1} w_{kj}^{(n-1)} \left(\ldots\left(\sum_{i=1} w_{ji}^{(1)} y_i + b_j^{(1)}\right)\ldots\right) + b_k^{(n-1)}\right) + b_l^{(n)}$$

BC loss:

$$loss_c^{(j)} = \frac{1}{N_c^{(j)}} \sum_{i=1}^{N_c^{(j)}} [q^{(j)}(\mathbf{y}_i, \mathbf{w}, \mathbf{b}) - q_i^{(j)}]^2 \quad (j = 1, 2, \ldots, 5)$$

Eqn loss: $loss_f^{(k)} = \frac{1}{N_f^{(k)}} \sum_{i=1}^{N_f^{(k)}} f_k^2(\mathbf{y}_i, q(\mathbf{y}_i, \mathbf{w}, \mathbf{b})) \quad (k = 1, 2, \ldots, 5)$

Updating variables: $\mathbf{w}$: weights, $\mathbf{b}$: biases + λ: unknown in eqns

Optimization ← Loss

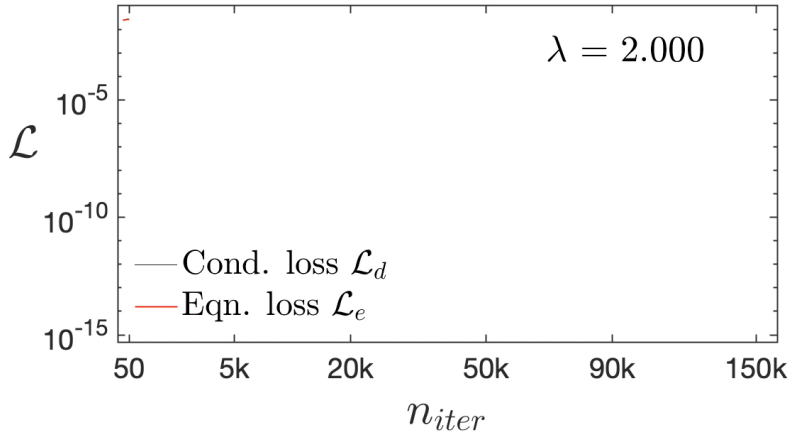
$q^{(j)}$: j-th output variable

$q_i^{(j)}$: data of $q^{(j)}$ at $\mathbf{y} = \mathbf{y}_i$

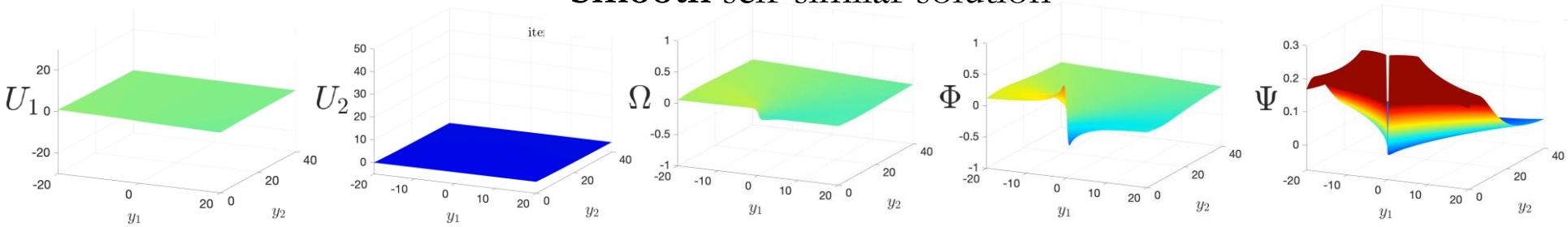
Self-similar blow-up solution for Luo-Hou scenario

We also need to make λ a free parameter to be determined

Loss function

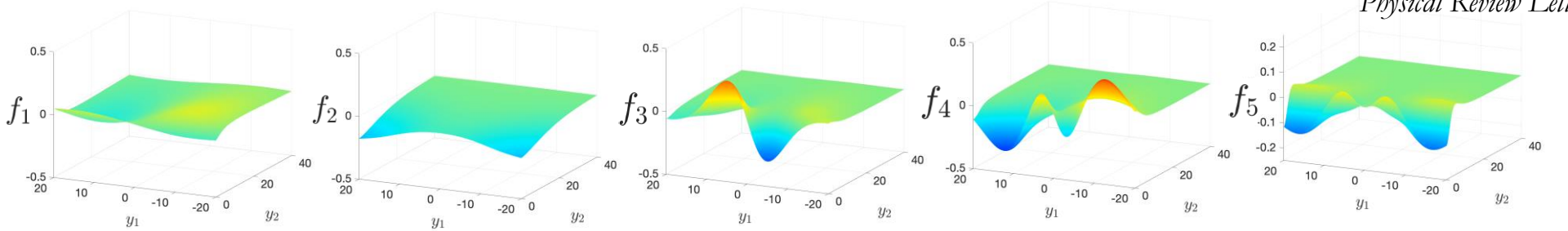


Smooth self-similar solution



iter. $= 50$

Uniform and small equation residues everywhere



Wang-Lai-GómezSerrano-Buckmaster, *Physical Review Letters* (2023)

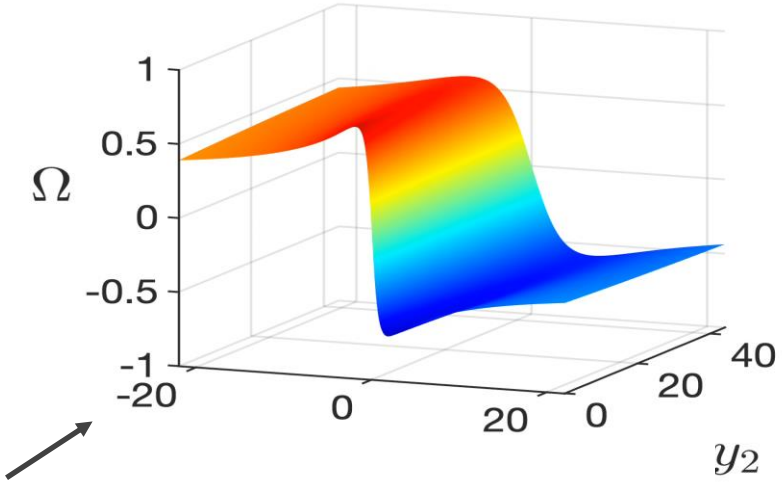
Self-similar blow-up solution for Luo-Hou scenario

Inferred $\lambda = 1.917$

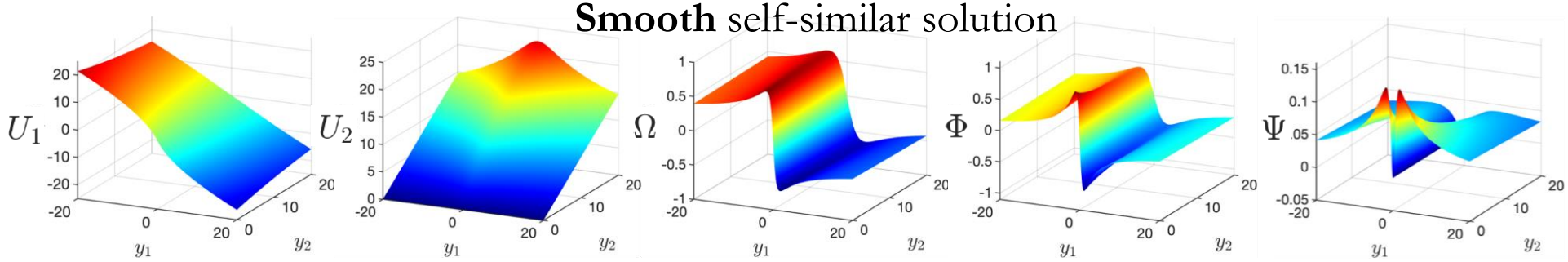
(Luo-Hou $\lambda = 1.91$)

~10 hours on A100 GPUs

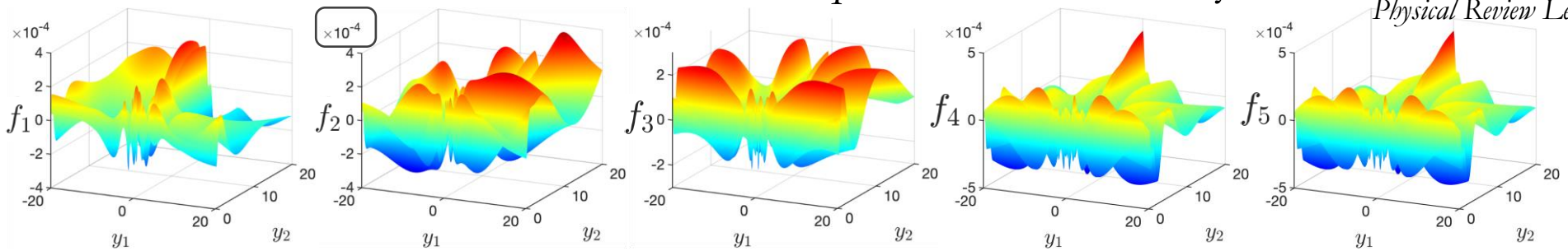
A rigorous proof of blow-up from smooth initial data was proven by Chen and Hou, arXiv:2210.07191



Smooth self-similar solution



Uniform and small equation residues everywhere



Wang-Lai-GómezSerrano-Buckmaster, *Physical Review Letters* (2023)

Quantamagazine

MATHEMATICAL PHYSICS

Deep Learning Poised to 'Blow Up' Famed Fluid Equations

5

For centuries, mathematicians have tried to prove that Euler's fluid equations can produce nonsensical answers. A new approach to machine learning has researchers betting that "blowup" is near.

Wang, Lai, Gómez-Serrano, Buckmaster,
Physical Review Letters (2023)
https://arxiv.org/abs/2201.06780

"[PINN solution] is quantitative and precise and has a much better chance of being made rigorous,"

EXISTENCE AND SMOOTHNESS OF THE NAVIER–STOKES EQUATION

CHARLES L. FEFFERMAN

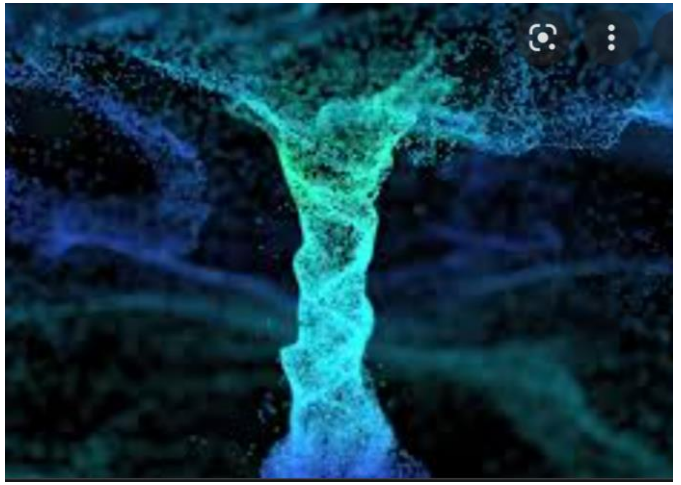
The Euler and Navier–Stokes equations describe the motion of a fluid in $\mathbb{R}^n$ ($n = 2$ or 3). These equations are to be solved for an unknown velocity vector $u(x,t) = (u_i(x,t))_{1\le i\le n} \in \mathbb{R}^n$ and pressure $p(x,t) \in \mathbb{R}$, defined for position $x \in \mathbb{R}^n$ and time $t \ge 0$. We restrict attention here to incompressible fluids filling all of $\mathbb{R}^n$. The *Navier–Stokes* equations are then given by

So what have we learned?

- Inferring ice viscosity

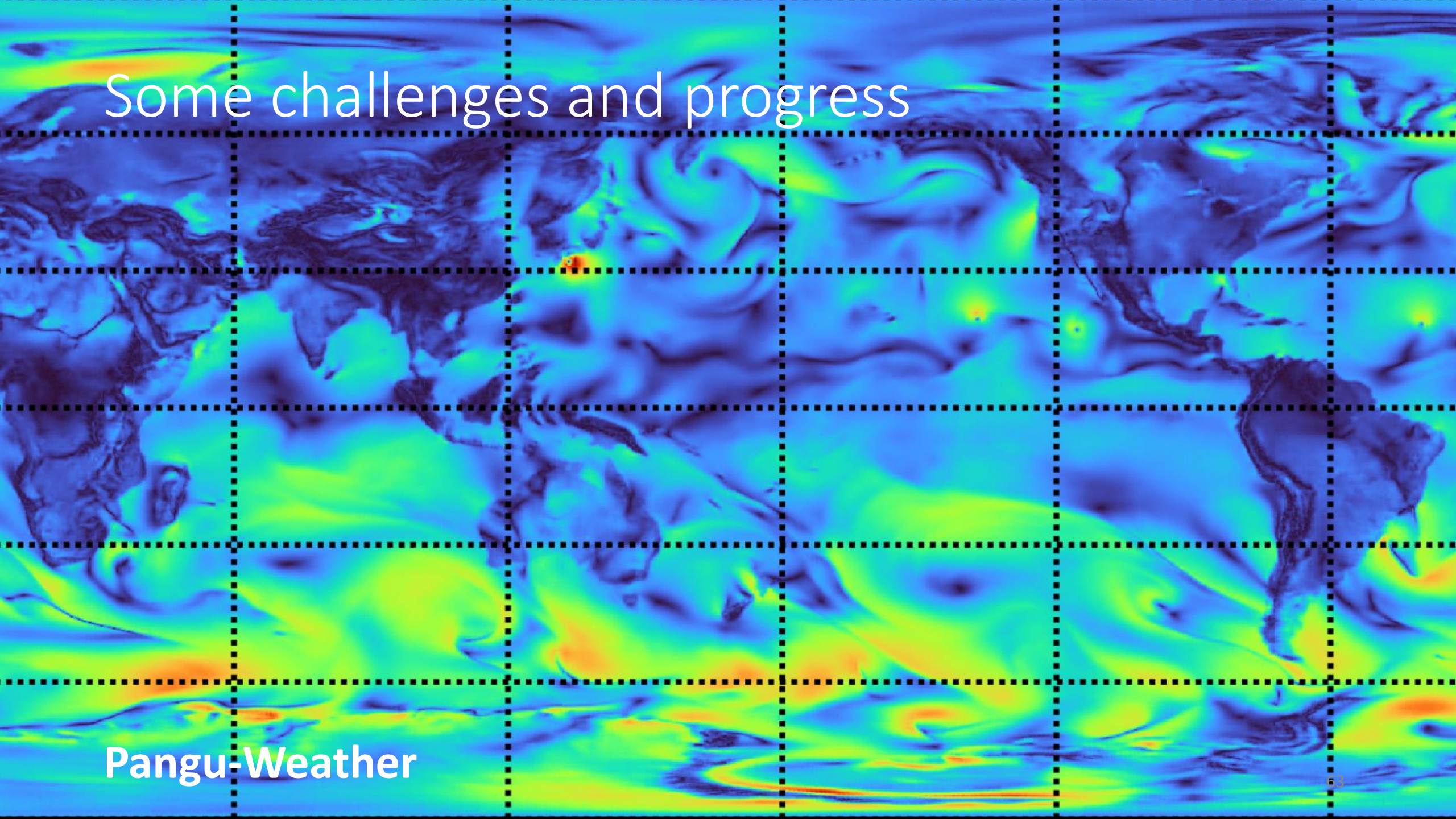
- Hurricane data assimilation

- Finding singularities



- Offer new knowledge that we didn't already know!
- It gives you an answer that can take much longer time to figure out with traditional methods.

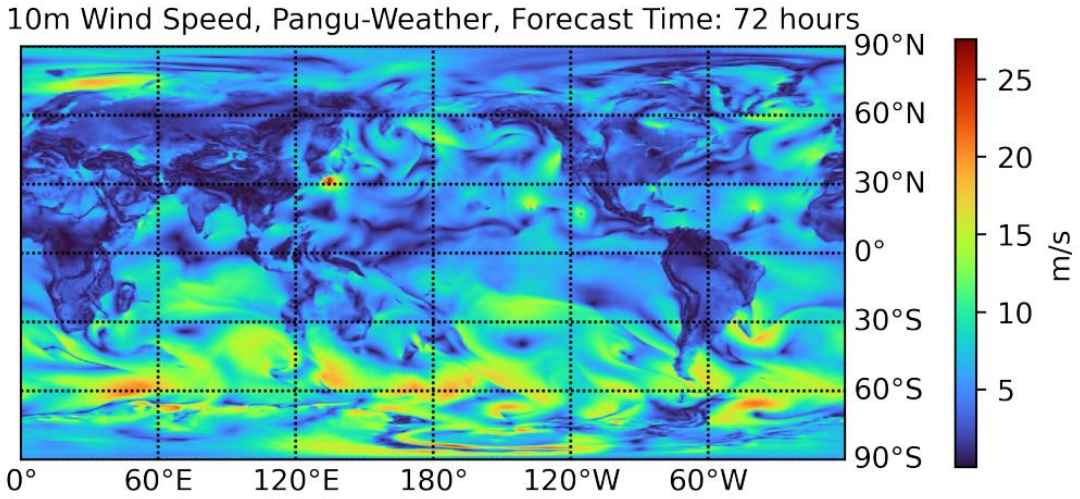
Some challenges and progress



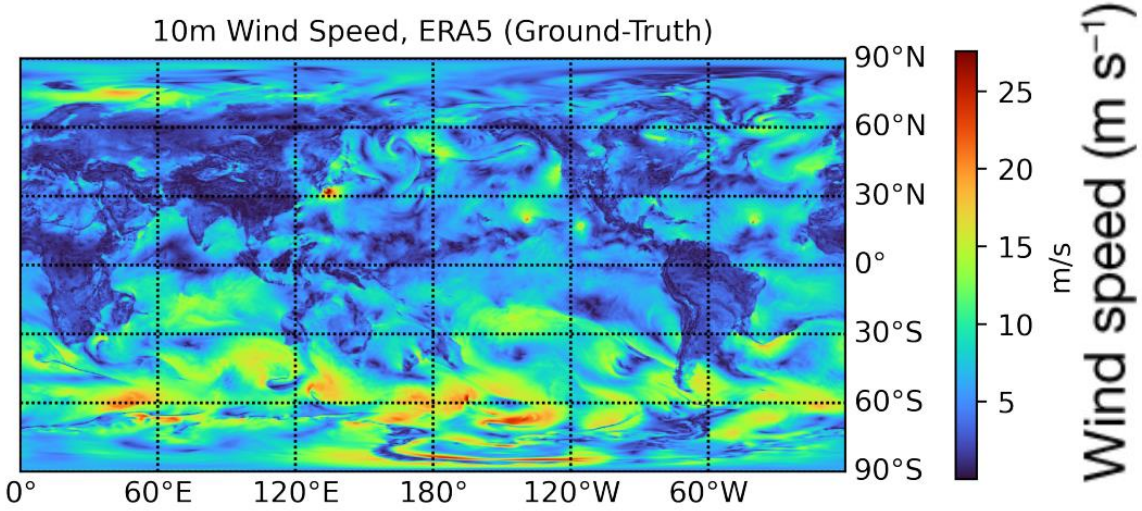
Pangu-Weather

Some challenges and progress

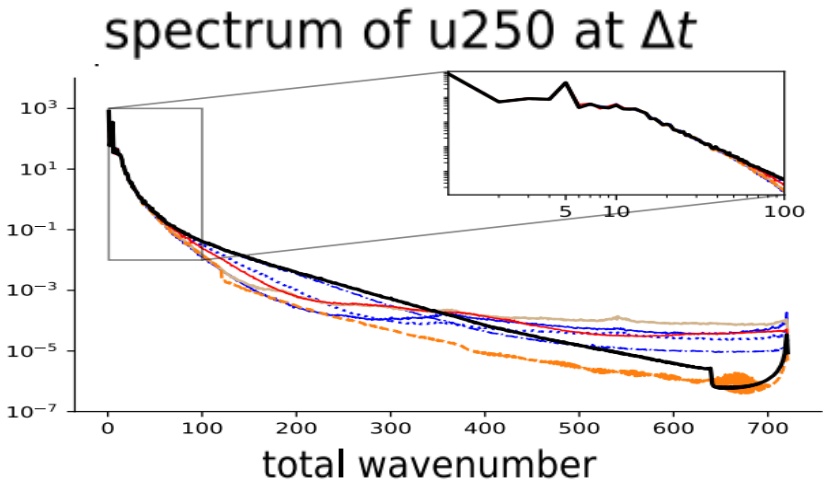
AI weather forecast

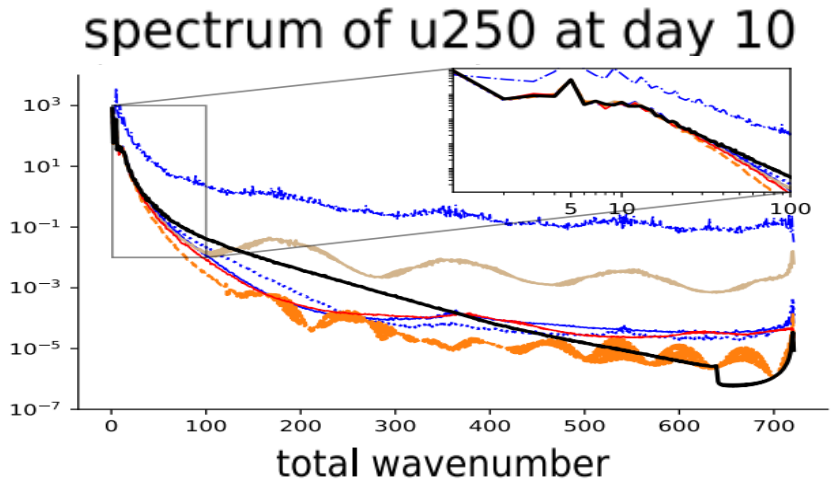


Training data (Re-analysis)



- Spectral bias of NN





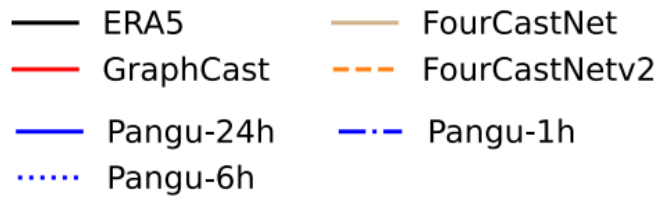
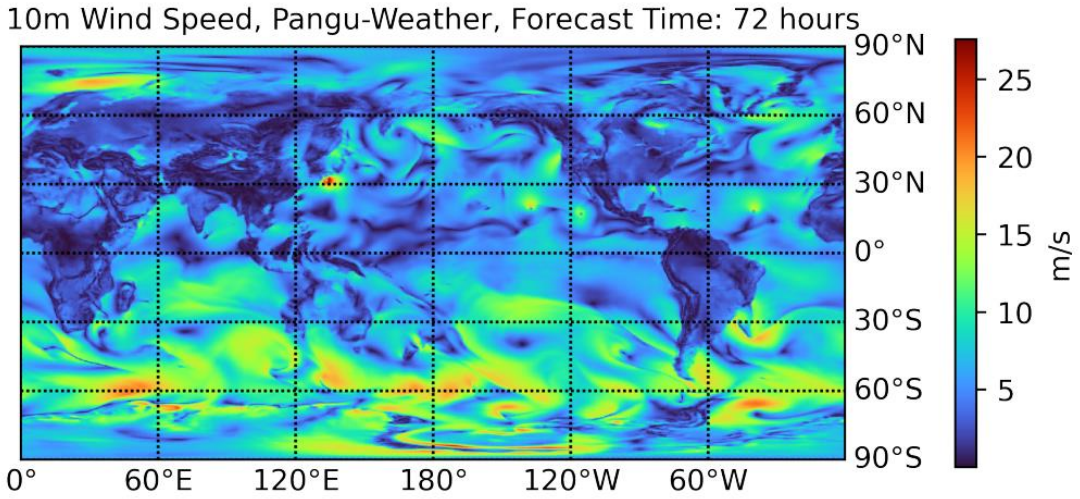


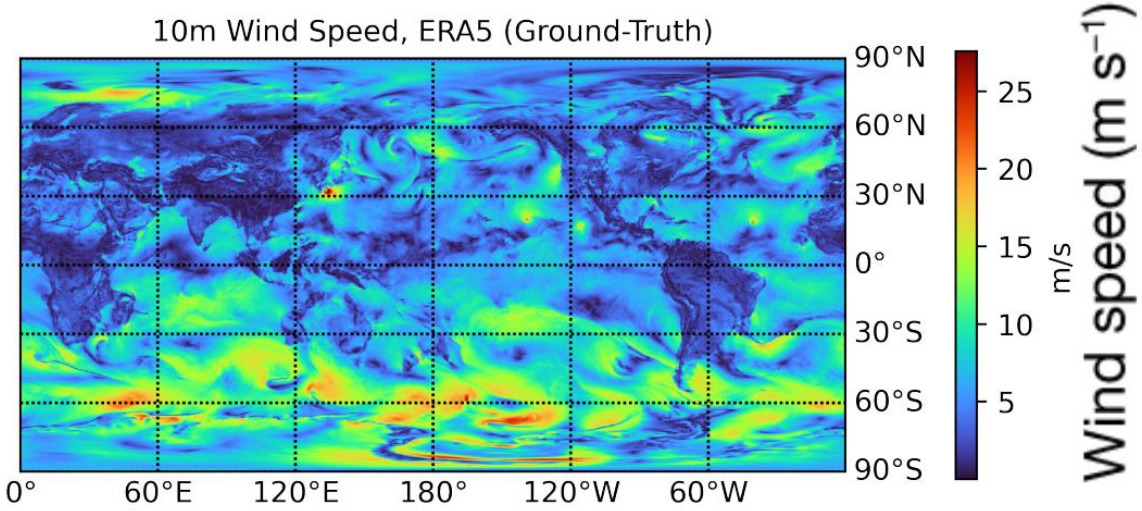
Image source: Qiang Sun & Pedram Hassanzadeh

Some challenges and progress

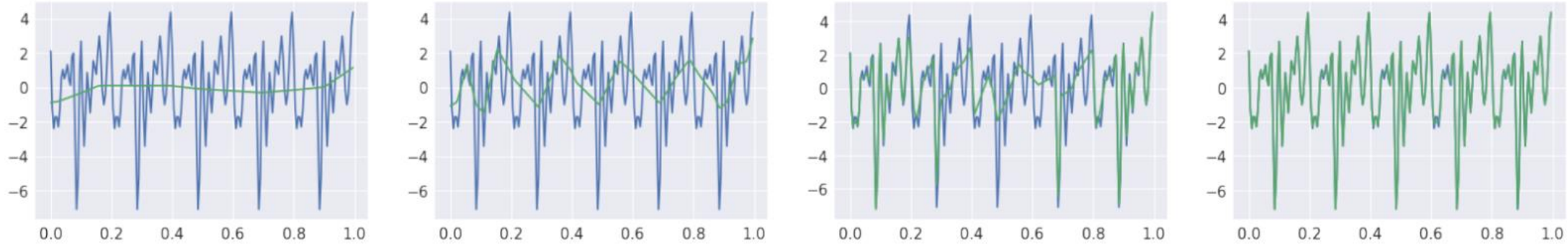
AI weather forecast



Training data (Re-analysis)



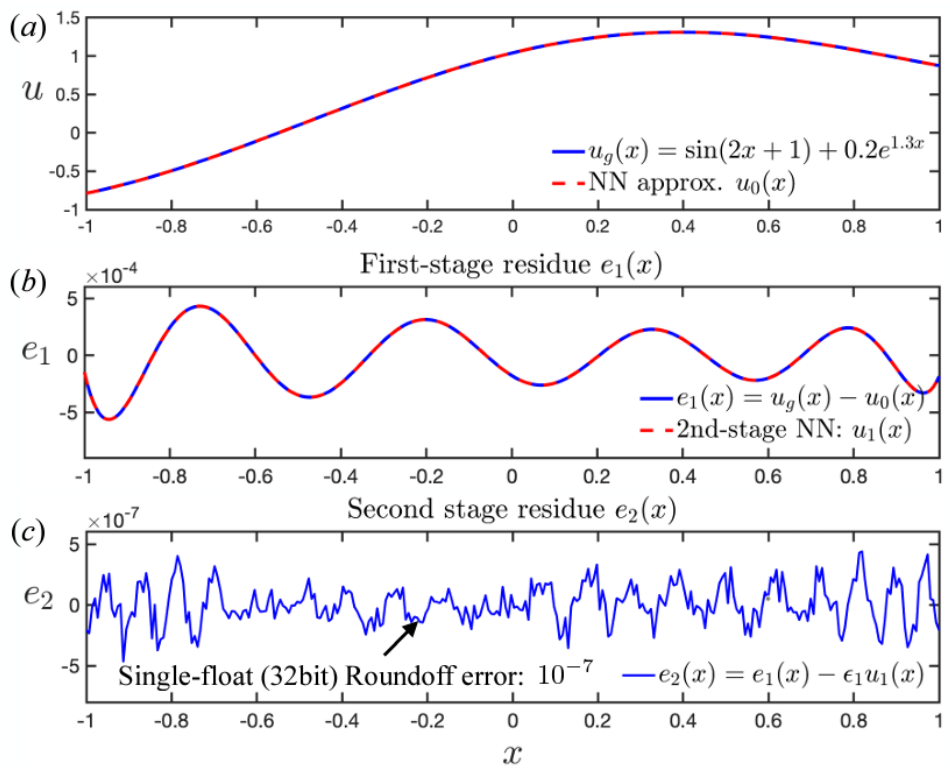
- Spectral bias of NN: Rahaman et al (2019)

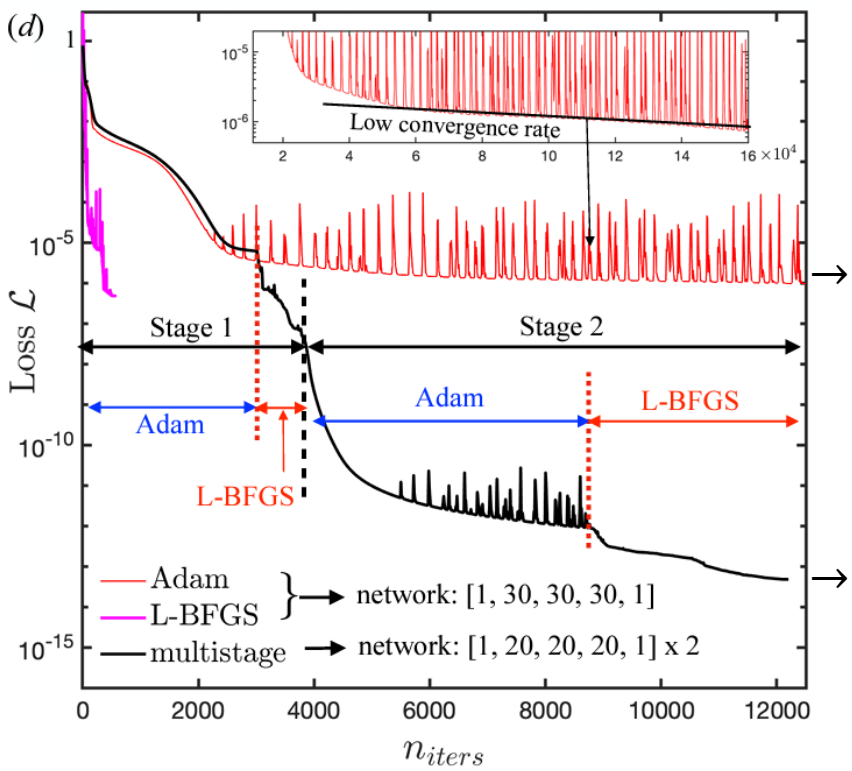


(a) Iteration 100 (b) Iteration 1000 (c) Iteration 10000 (d) Iteration 80000

Inability capture small-scale processes is a problem

- NN's function approximation errors





→ Regular NN

→ Multistage NN
Machine precision

Some other challenges and progress

- NN's function approximation errors → Multistage neural network

  Wang & Lai, J. Comput. Phys. (2024)

- Approximating high-frequency functions → Jagtap et al., JCP (2020)
  Introduce a scaling variable in the activation function, which can be optimized

  $\sigma(a\,\mathcal{L}_k(z^{k-1}))$,

- Determine the weights for different terms in the cost function
  → Wang et al. SIAM J. Sci. Comput. (2021), Wang et al., JCP (2022), McClenny & Braga-Neto, SAPINN arXiv:2009.04544

- Error bounds of the PINN solutions → Mishra & Molinaro, IMA J. Numer. Anal. (2023)
  In addition to saying PDE residues are small, we know that the distance between the PINN approximated solution itself with the true solution, which is not available, would be small

From predicting to understanding

- ML models are powerful tools due to capacity to represent data.
- MLs are just tools. ML model tells us empirical findings from data, but not (yet) reasoning of its finding.
- Our job: reason, interpret, and understand the ML predictions.
- In addition to predicting, the excitement of a data-driven world lies in the possibility of transforming predictions into new understandings..