

Physics-informed neural networks (PINN)

by C. Yao Lai for 2024 Glamacles summer school

Image generated by Generative AI: Firefly

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

State parameters that impacts ice dynamics yet difficult, if not impossible, to measure:

Ice viscosity (spatially varying)

Basal friction (spatially varying)

Need to solve inverse problem to infer them:

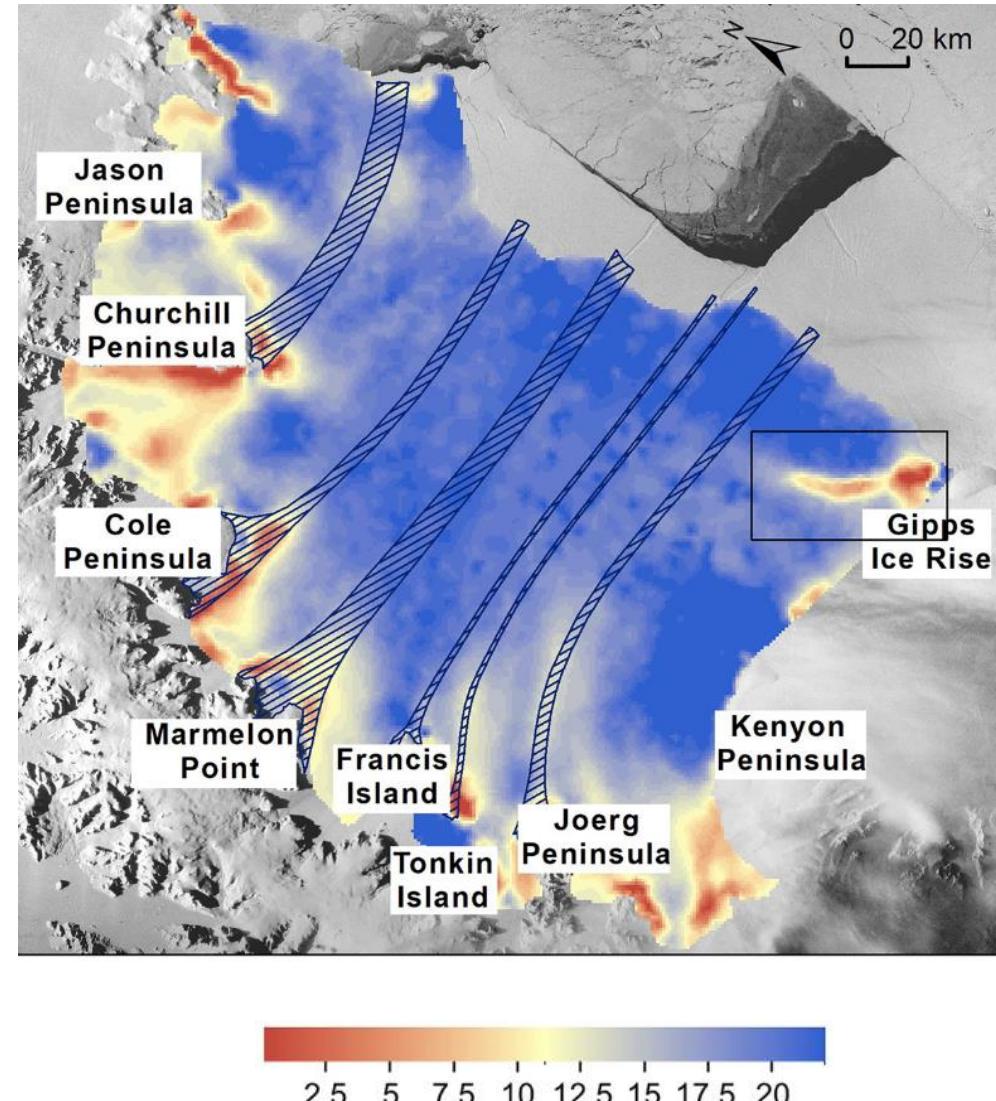
Adjoint method, Bayesian inference, PINN, CNN

Gaussian Process...etc

modeled basal friction [kPa yr/m]

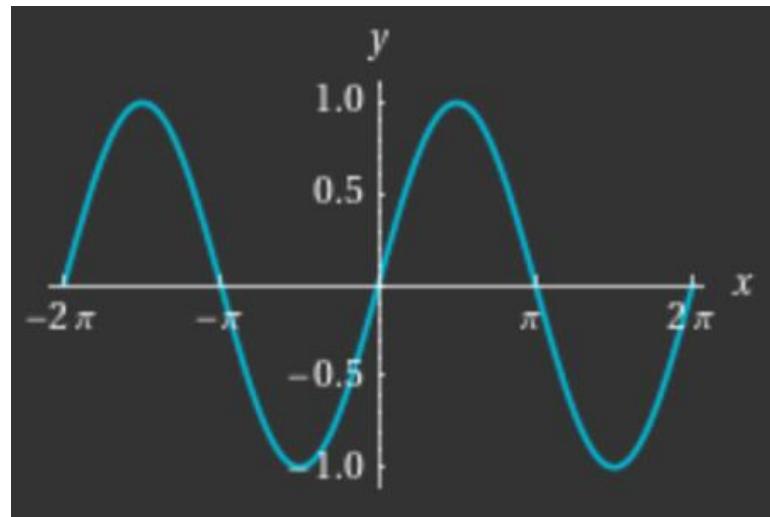


Ice rigidity parameter \bar{B}
($\times 10^7$ Pa s $^{1/3}$)



How to represent functions/data/solution?

$$u(x) = \sin(x)$$



1. Put it on a numerical grid

Sample N points in $x = [-2\pi, 2\pi]$
of parameters: N

2. Fourier Series

Parameterize function $u(x)$ by

$$u(x) = A_0 + \sum_{n=1}^N \left(A_n \cos\left(2\pi \frac{n}{P} x\right) + B_n \sin\left(2\pi \frac{n}{P} x\right) \right)$$

of parameters: 1



How to represent functions/data/solution?

1. Put it on a numerical grid

Sample N points in $x = [-2\pi, 2\pi]$
of parameters: N

2. Fourier Series

Parameterize function $u(x)$ by

$$u(x) = A_0 + \sum_{n=1}^N \left(A_n \cos\left(2\pi \frac{n}{P} x\right) + B_n \sin\left(2\pi \frac{n}{P} x\right) \right)$$

of parameters: ?



3. Bessel function....or other basis functions

4. **NN**: Universal function approximator. Reduce a priori assumption about basis fn. Let the data decide!

Outline

- Part I: Physics-informed neural networks
- Part II: Application to inverse modeling on ice shelves

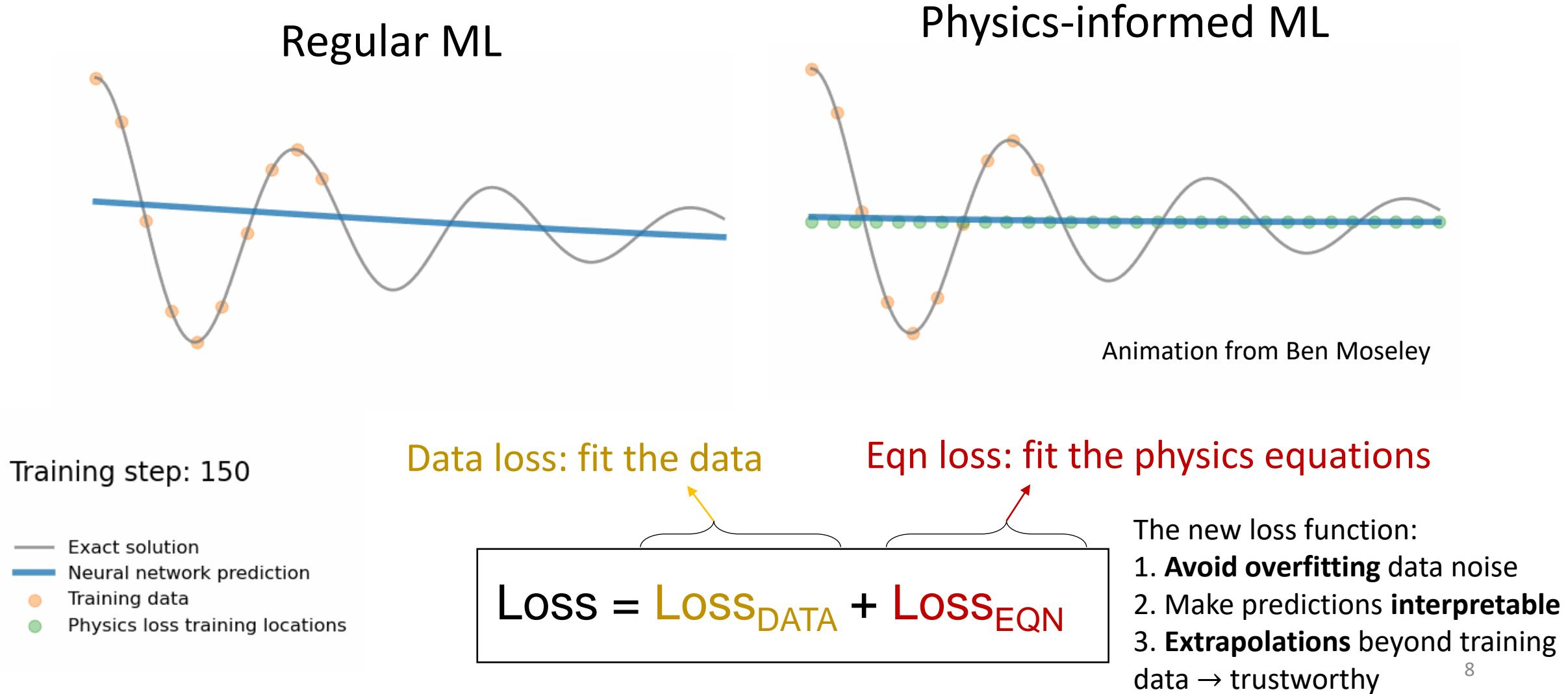
Cost function

- Fits the NN to the data (Universal function approximator)

$\text{Loss} = f(\text{ground truth} - \text{prediction}) = \text{Data misfit}$

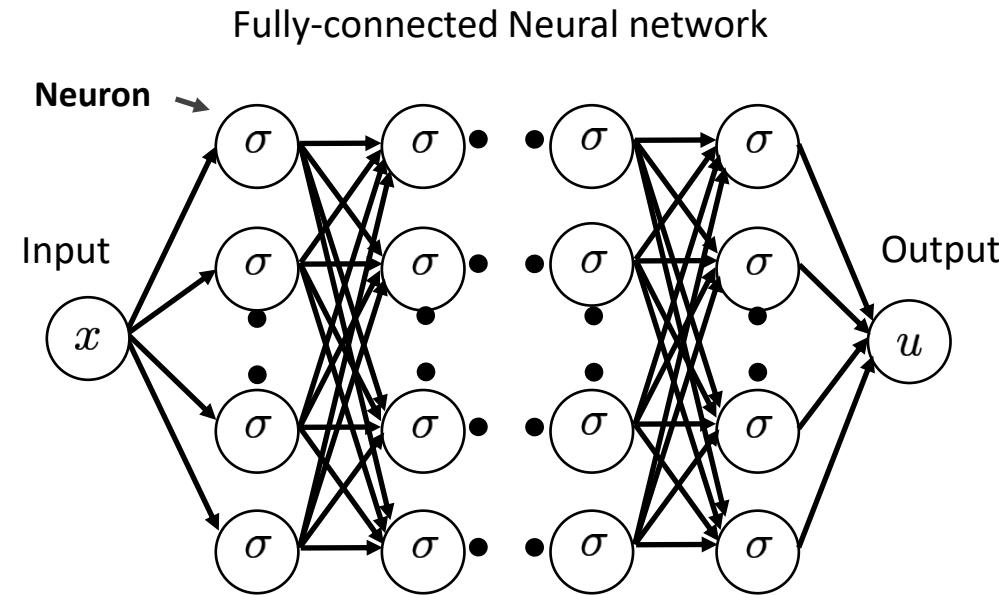
- Observation data is **almost certainly noisy**. How to avoid overfitting?
- Extrapolation: to what extent can the NN generalize to unseen data?

Better extrapolation beyond training data



Neural network

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



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

w: weights

b : biases

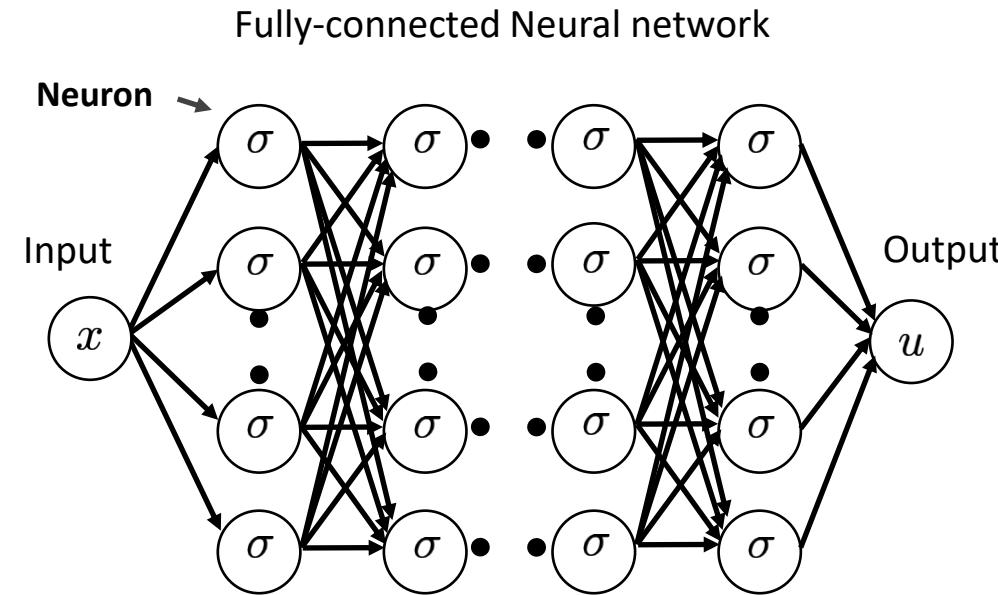
$\sigma(x)$: activation function

(free parameters to be trained)

(fixed and selected by users)

Neural network

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



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

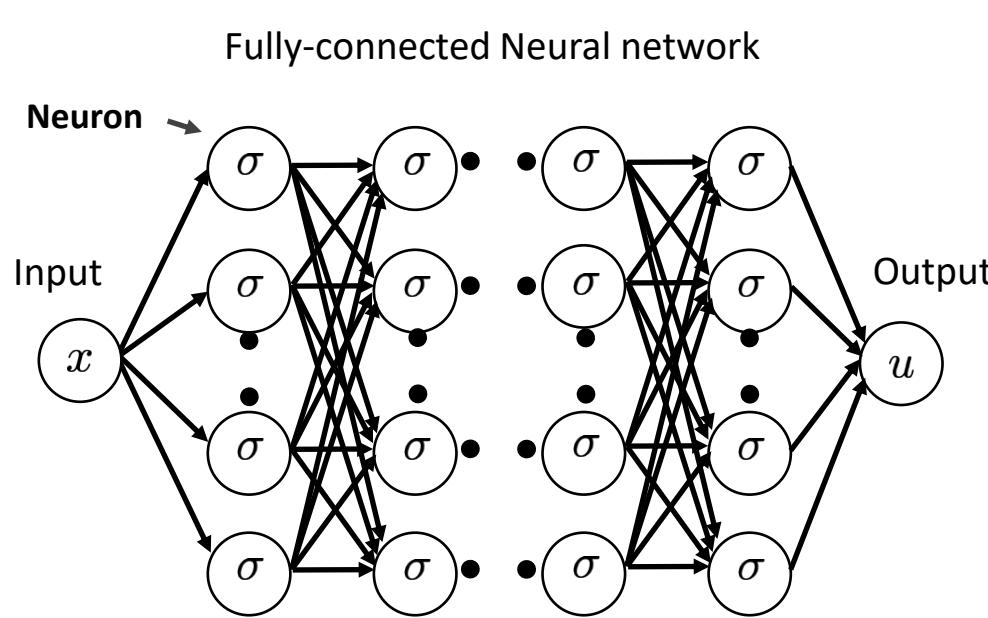
w: weights **b** : biases $\sigma(x)$: activation function

Universal function approximator

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

Neural network and Fourier series

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



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

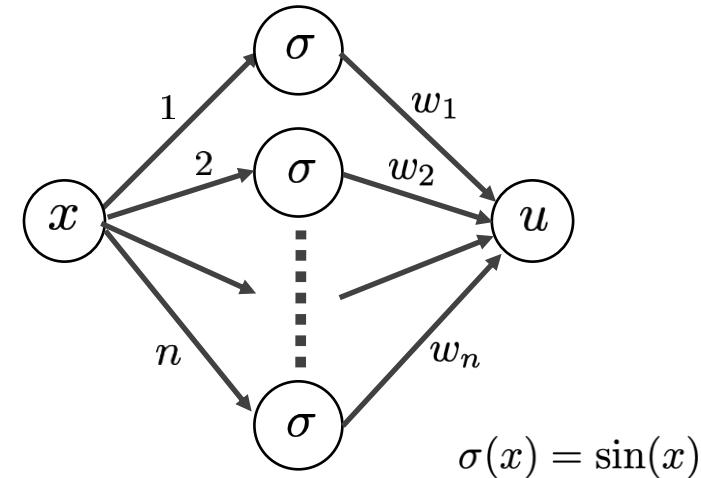
w: weights

b : biases

$\sigma(x)$: activation function

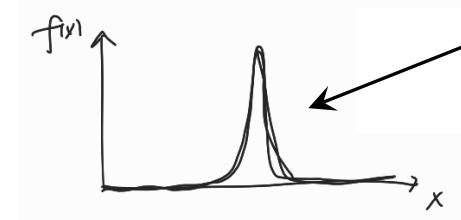
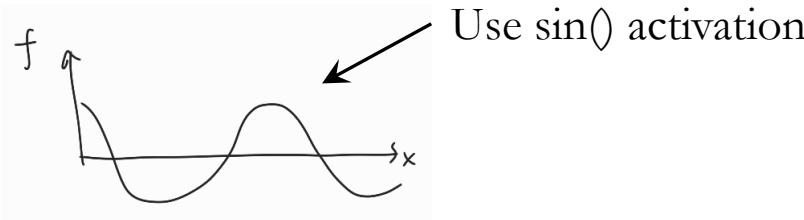
Fourier series:

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



Universal function approximator

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

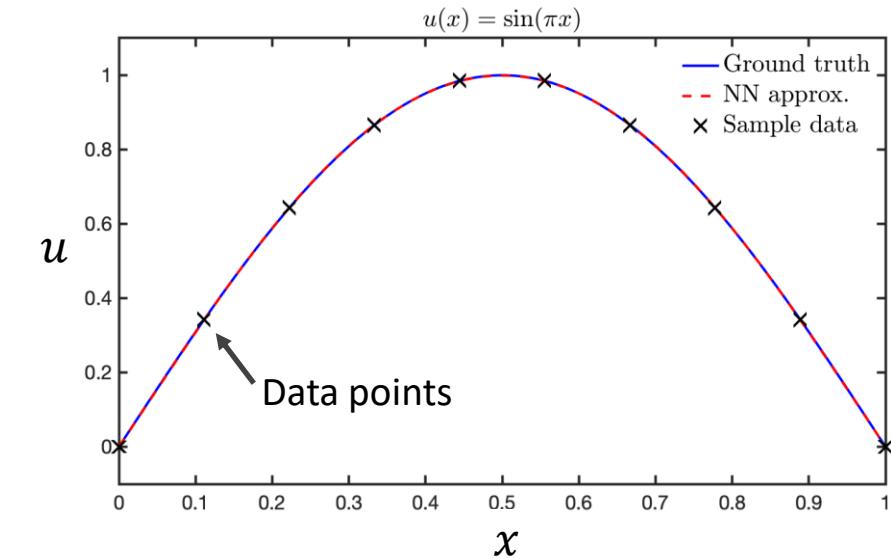
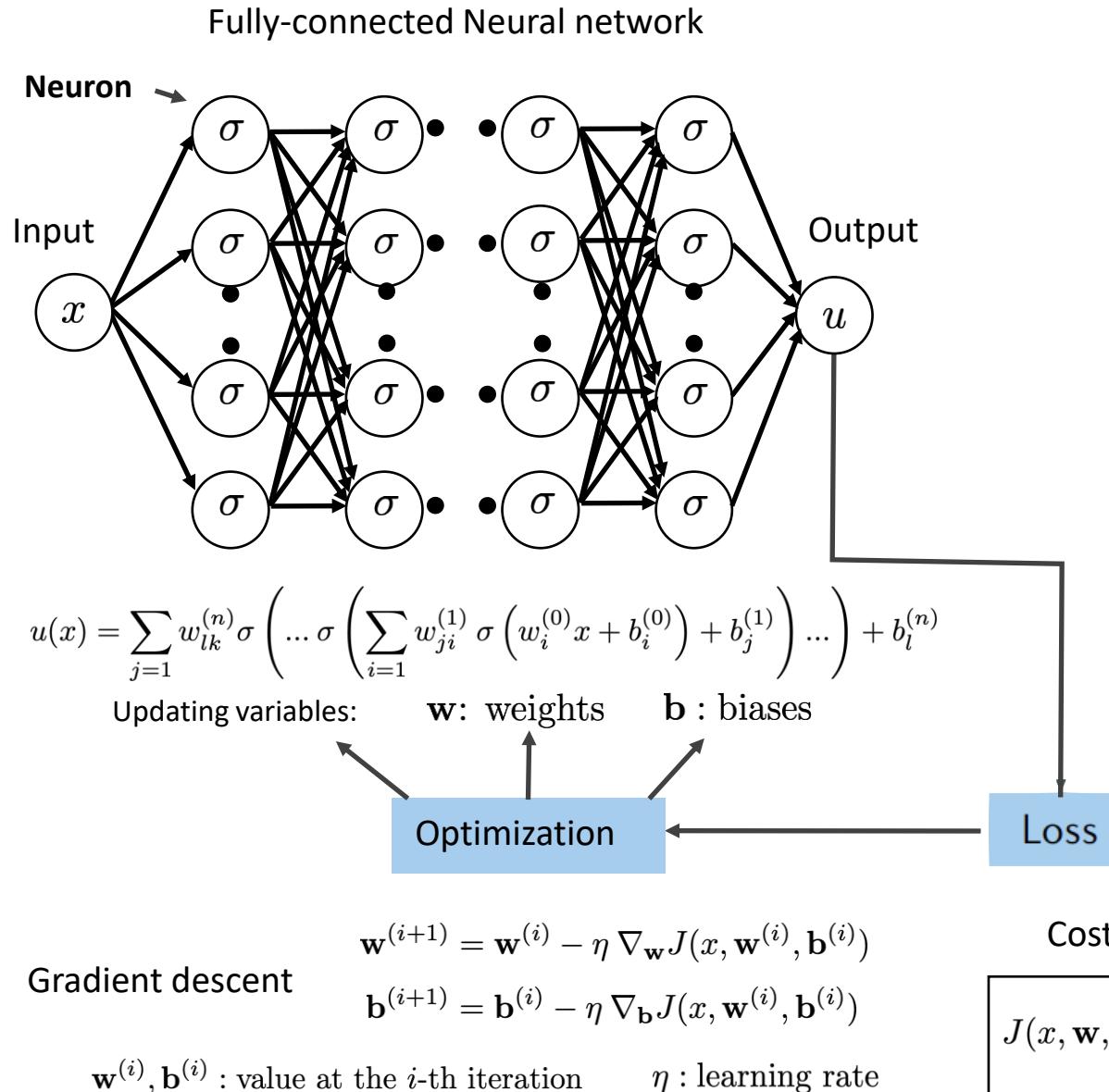


If using $\sin()$ activation, need lots of neurons (frequencies)

Better to use $\tanh()$ activation

Neural network for curve fitting

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



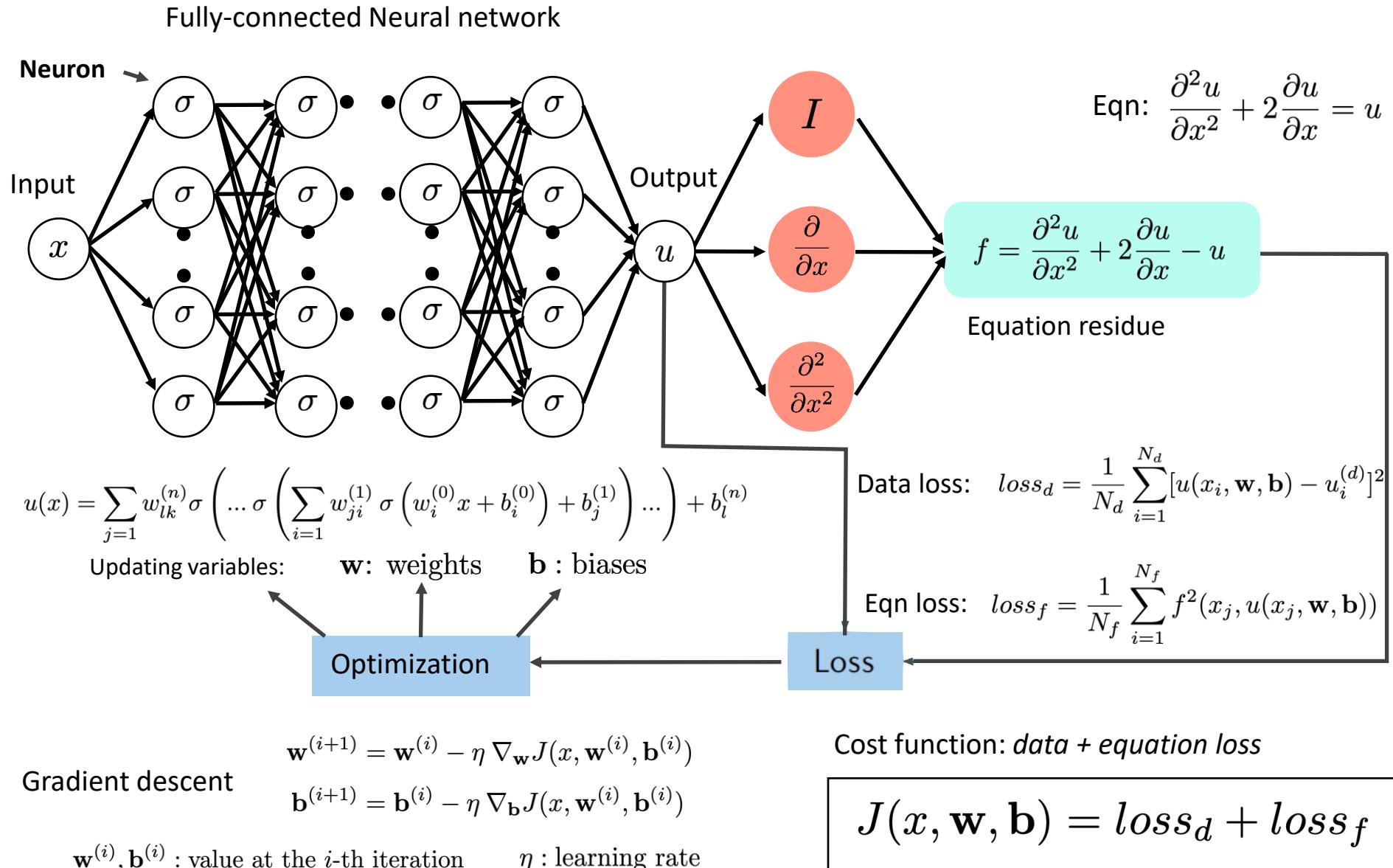
Cost function: *mean squared error*

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

data of u at $x = x_i$

Physics-informed neural networks

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



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 $u(x, t)$!

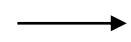
Example of solving PDEs: Diffusion equation

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

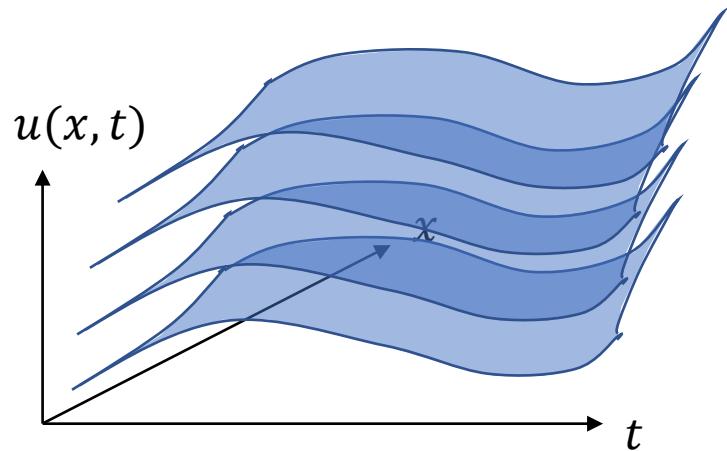
1st order derivative in time t
2nd order derivatives in space x



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

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

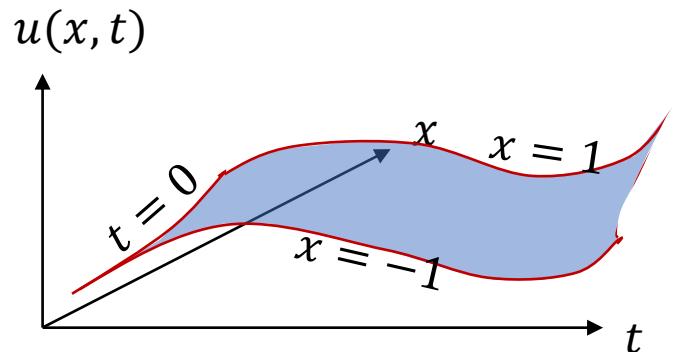
1st order derivative in time t
2nd order derivatives in space x



1 initial condition
2 boundary conditions



GOAL: solve for $u(x, t)$



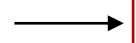
Need to give boundary and initial conditions to properly find a unique surface $u(x, t)$

Example of solving PDEs: Diffusion equation

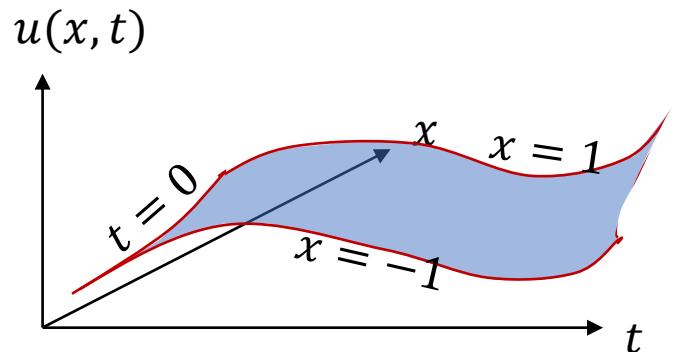
$$\frac{\partial u}{\partial t} = a \frac{\partial^2 u}{\partial x^2}$$

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

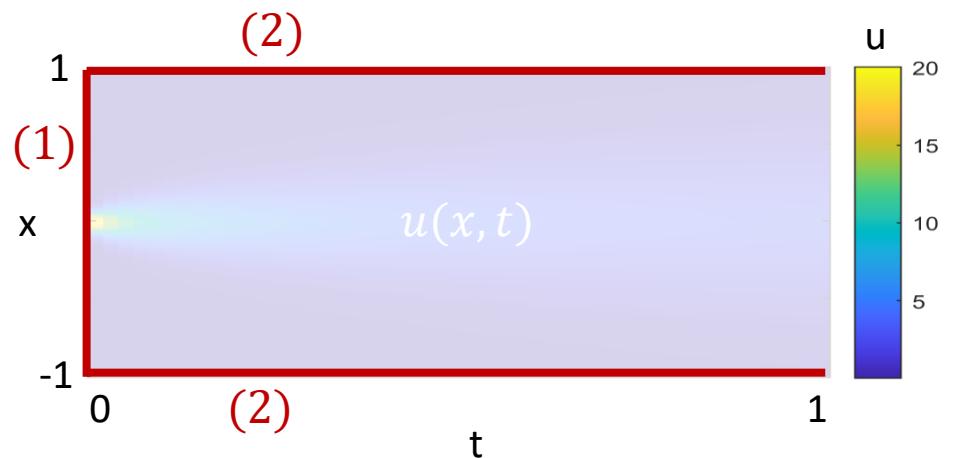
$$(3) \frac{\partial u}{\partial t} = a \frac{\partial^2 u}{\partial x^2}$$

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

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



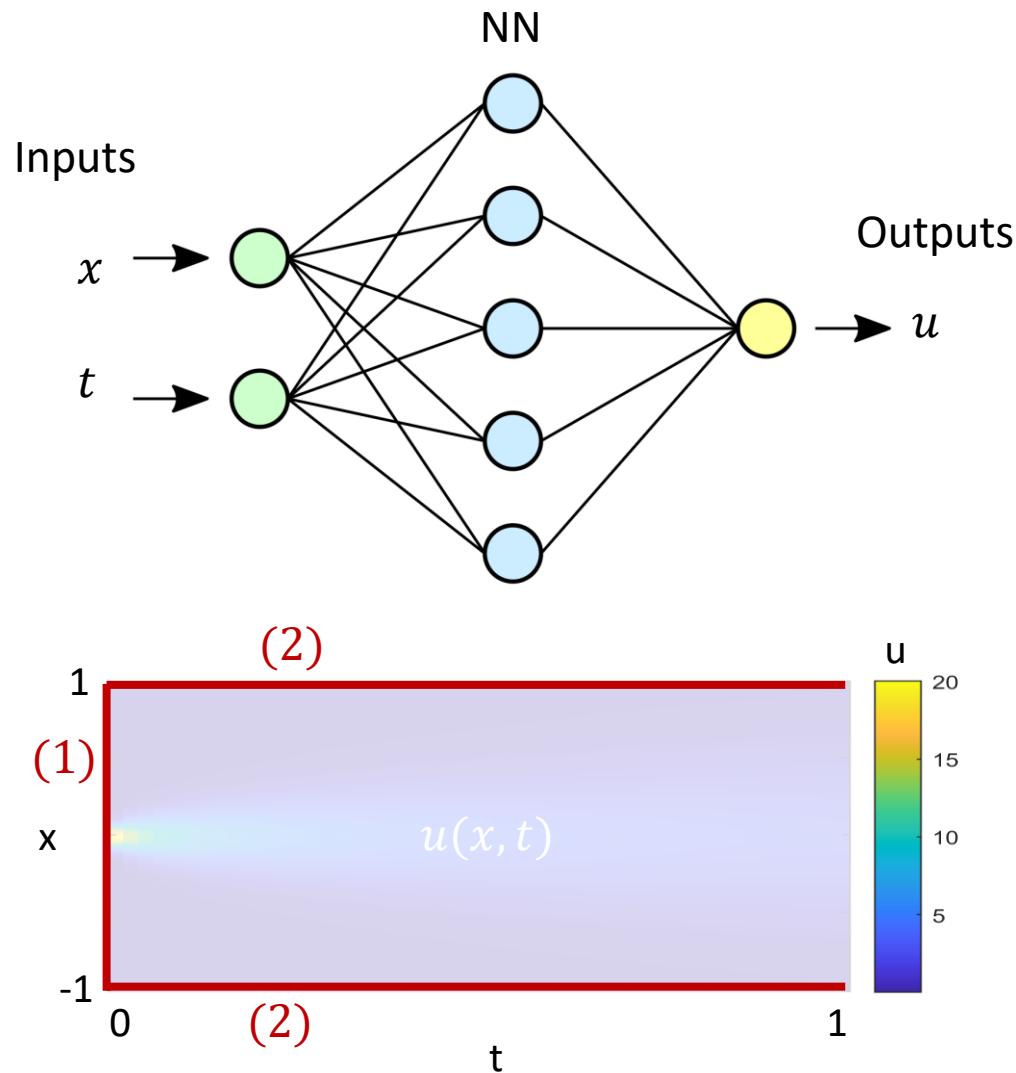
GOAL: solve for $u(x, t)$



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

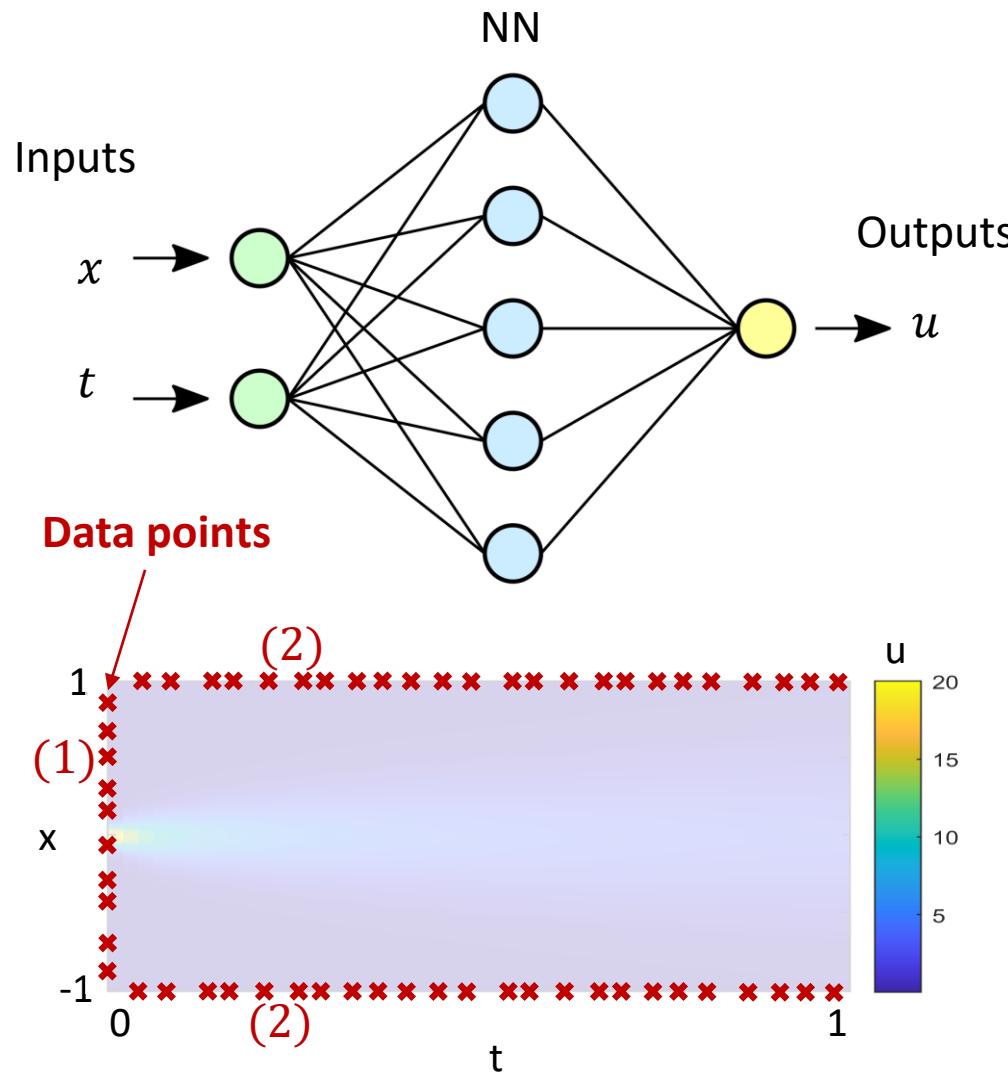


Use NN to search for a surface that satisfies
(1) ICs + (2) BCs + (3) physics equation.

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:

$$u_0^i \text{ at } \{t = 0, x_i\}, i = 1 \dots N$$

(2) Boundary conditions:

$$u_{lb}^j \text{ at } \{t_j, x = -1\} \text{ & } u_{ub}^j \text{ at } \{t_j, x = 1\}, j = 1 \dots M$$

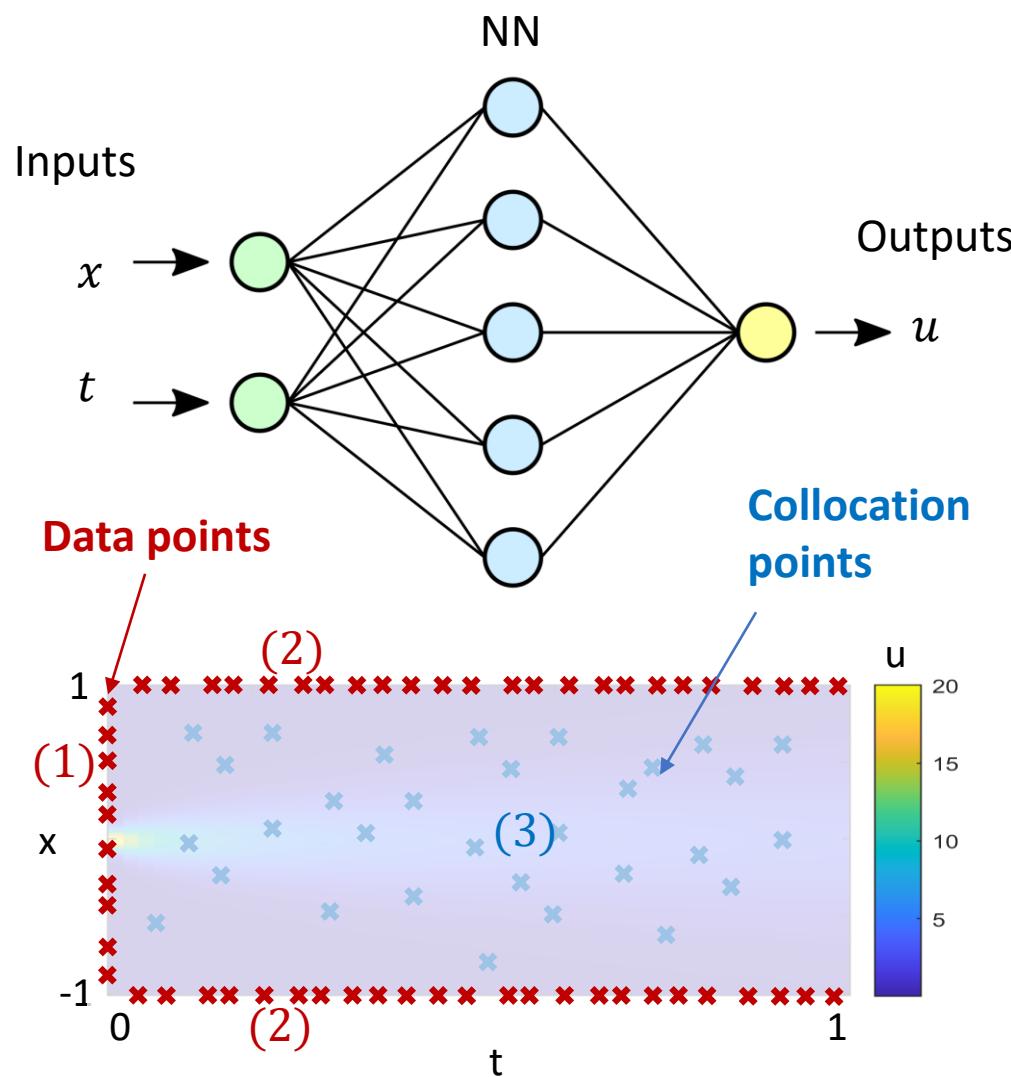
What would be the loss function?

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

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

GOAL: Find NN that minimizes the loss function

Physics-informed NN



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

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

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.

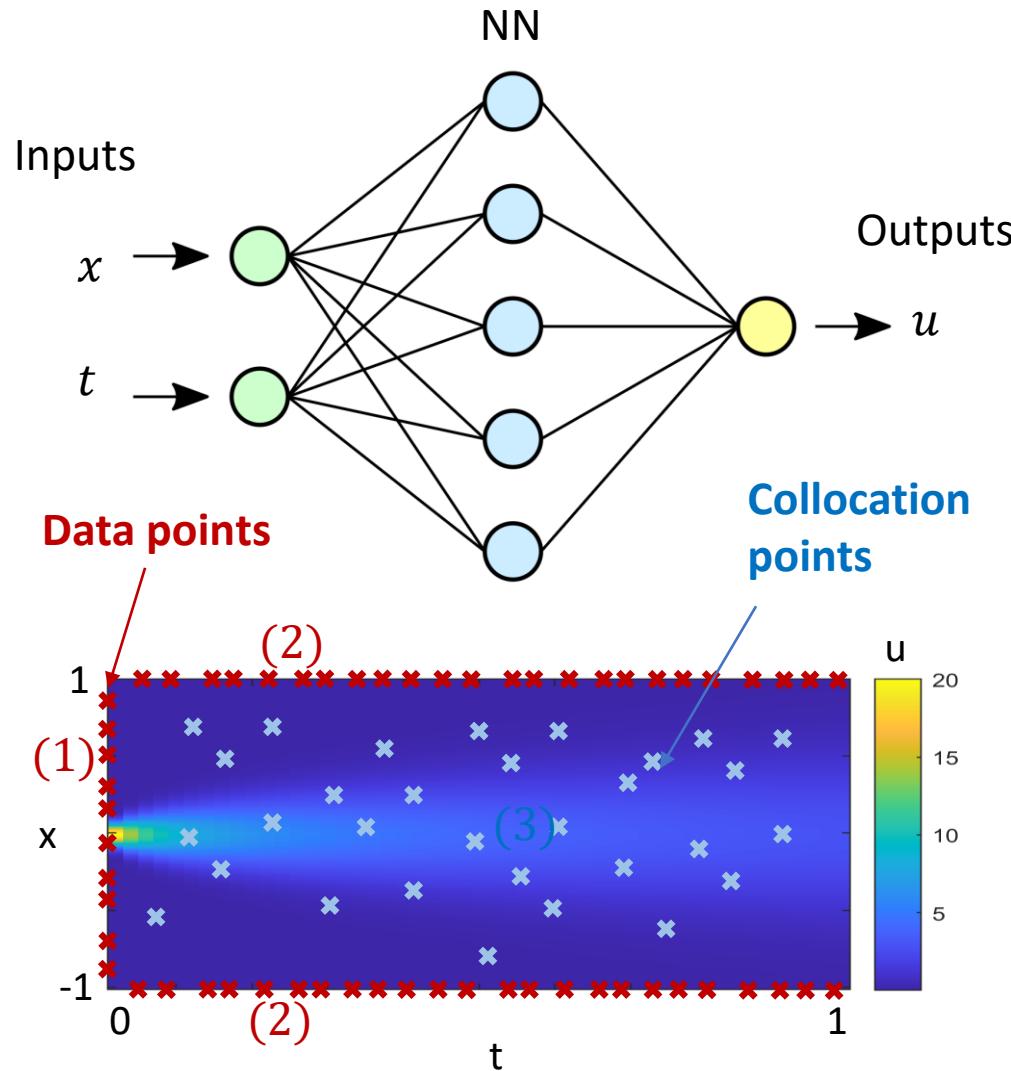
What would be the loss function?

Equation loss:

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

GOAL: Find NN that minimizes the loss function
→ solving for $u(x, t)$ satisfying (1) ICs + (2) BCs + (3) Eqn

Physics-informed NN



Given a partial differential equation of a general form:

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

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)

$$MSE = MSE_u + MSE_f,$$

Data loss

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

Data points

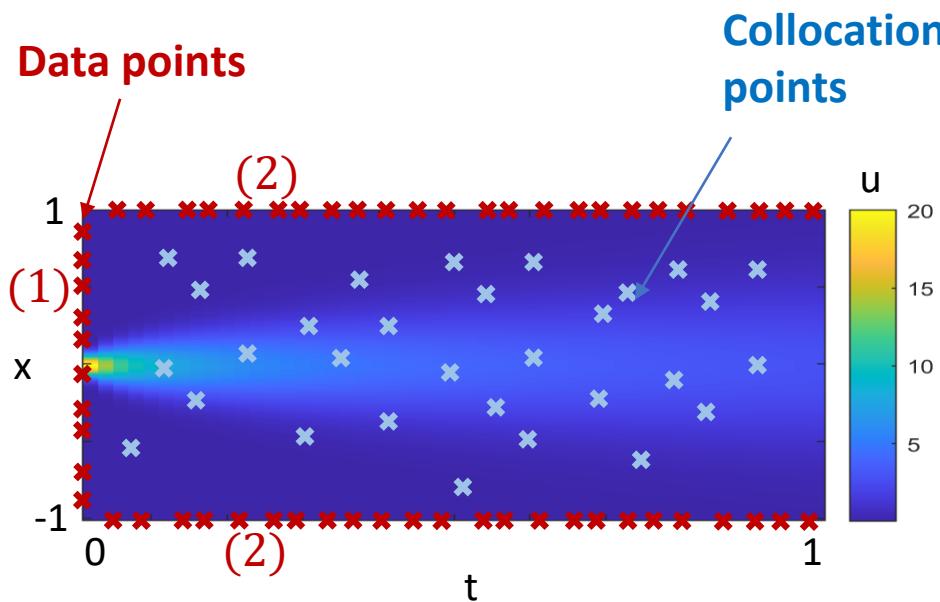
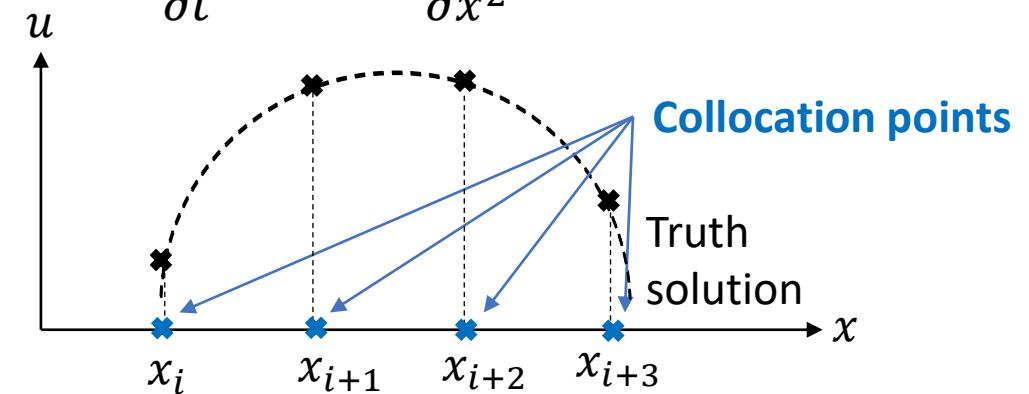
Equation loss

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

Collocation points

PINN: searches for a curve that satisfies

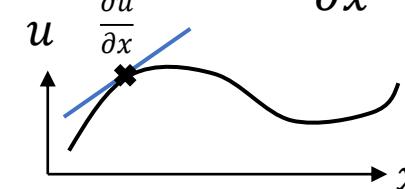
$$\frac{\partial u(x_i)}{\partial t} - a \frac{\partial^2 u(x_i)}{\partial x^2} \approx 0 \text{ at the collocation pts}$$



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

e.g. $\text{NN}(x) = u = \sigma(w_2\sigma(w_1x + b_1) + b_2)$,

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



PINN

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

- Application 1: Prediction of solution for a **well-posed forward problem** (I don't recommend doing this)

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

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

- Application 3: Data-driven discovery of unknown parameters

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

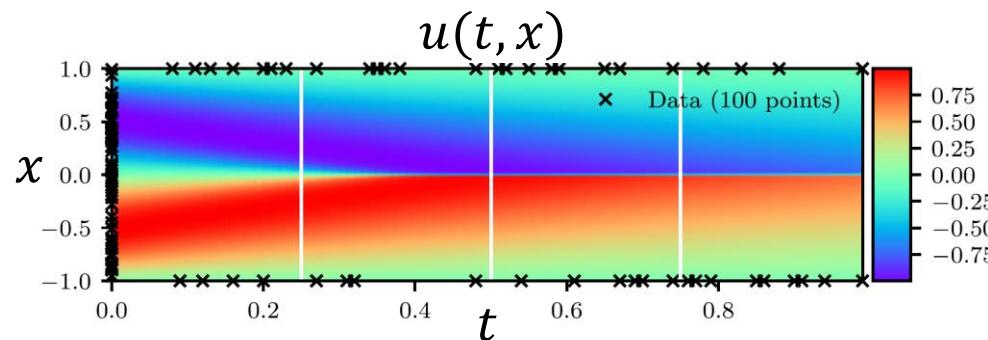
E.g., Burgers' equation (inference)

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):

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

Collocation points:

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

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

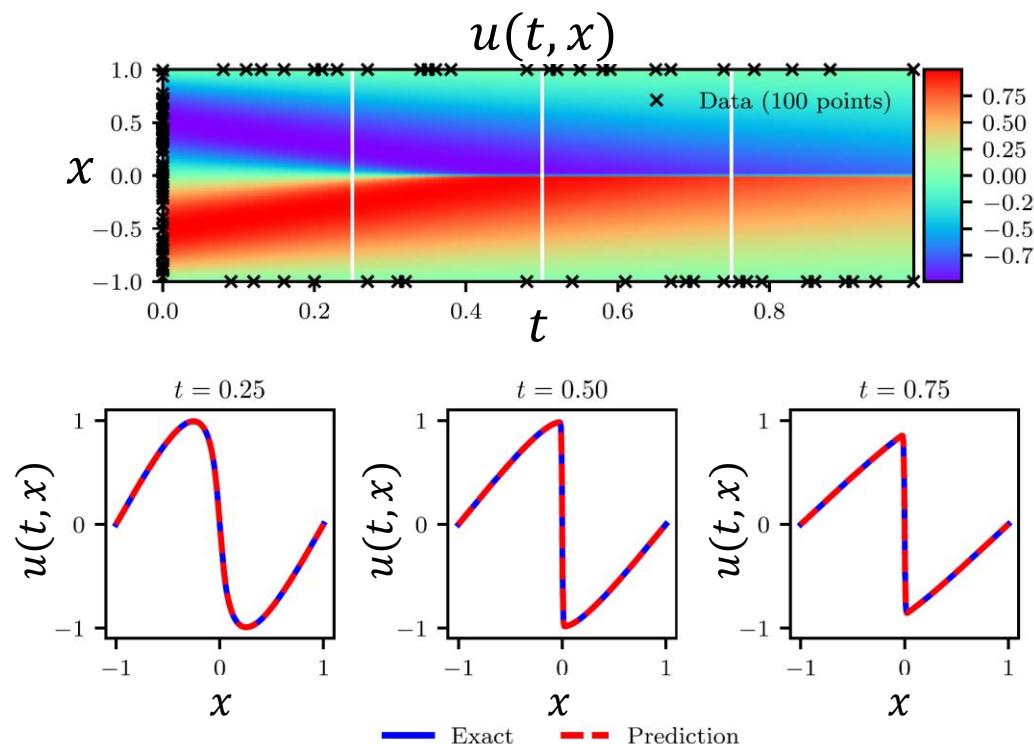
E.g., Burgers' equation (inference)

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):

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

Collocation points:

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

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** (I don't recommend doing this)

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

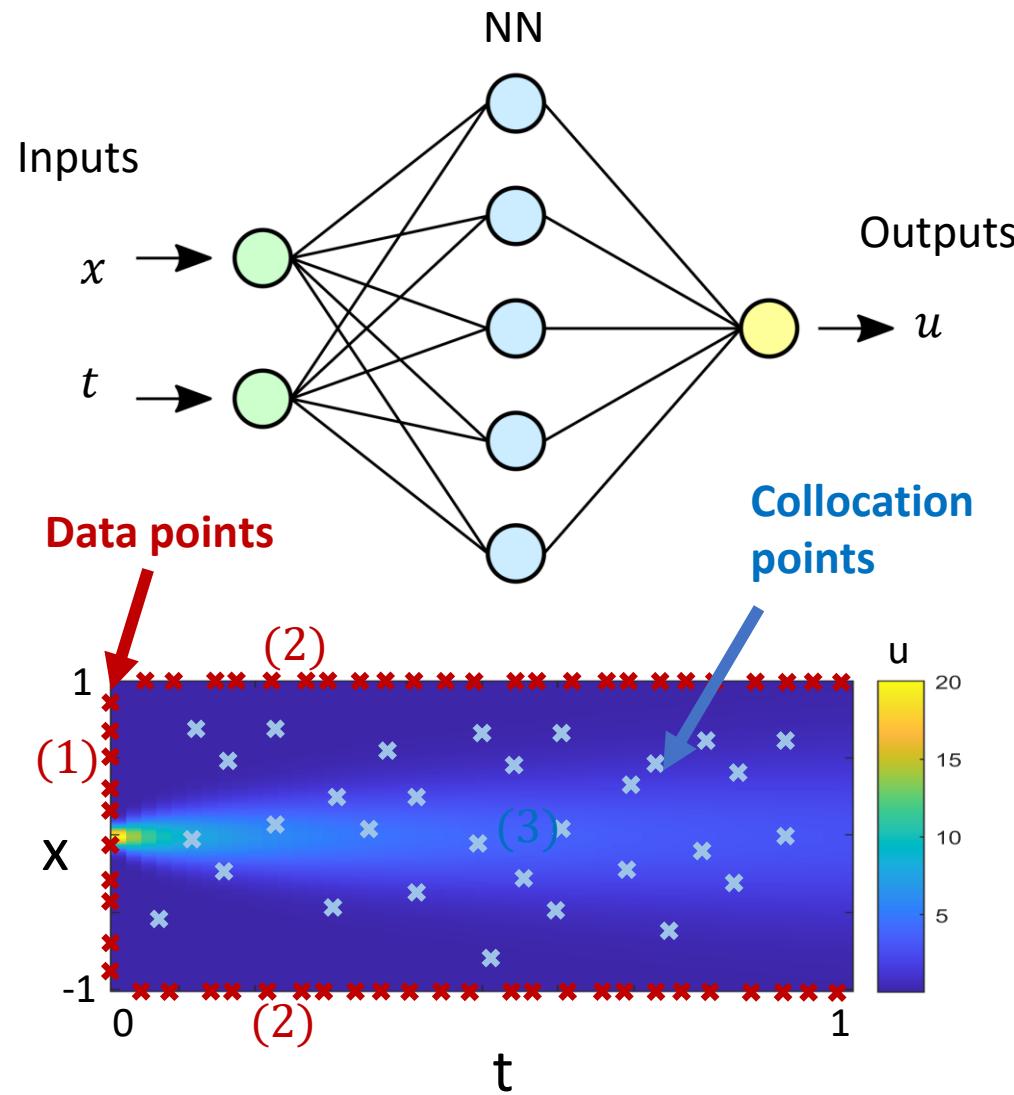
- Application 2: Prediction of solution when data is available within the domain but not at the IC, BC

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

- Application 3: Data-driven discovery of unknown parameters

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

Physics-informed NN



Given a partial differential equation of a general form:

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

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)

$$MSE = MSE_u + MSE_f,$$

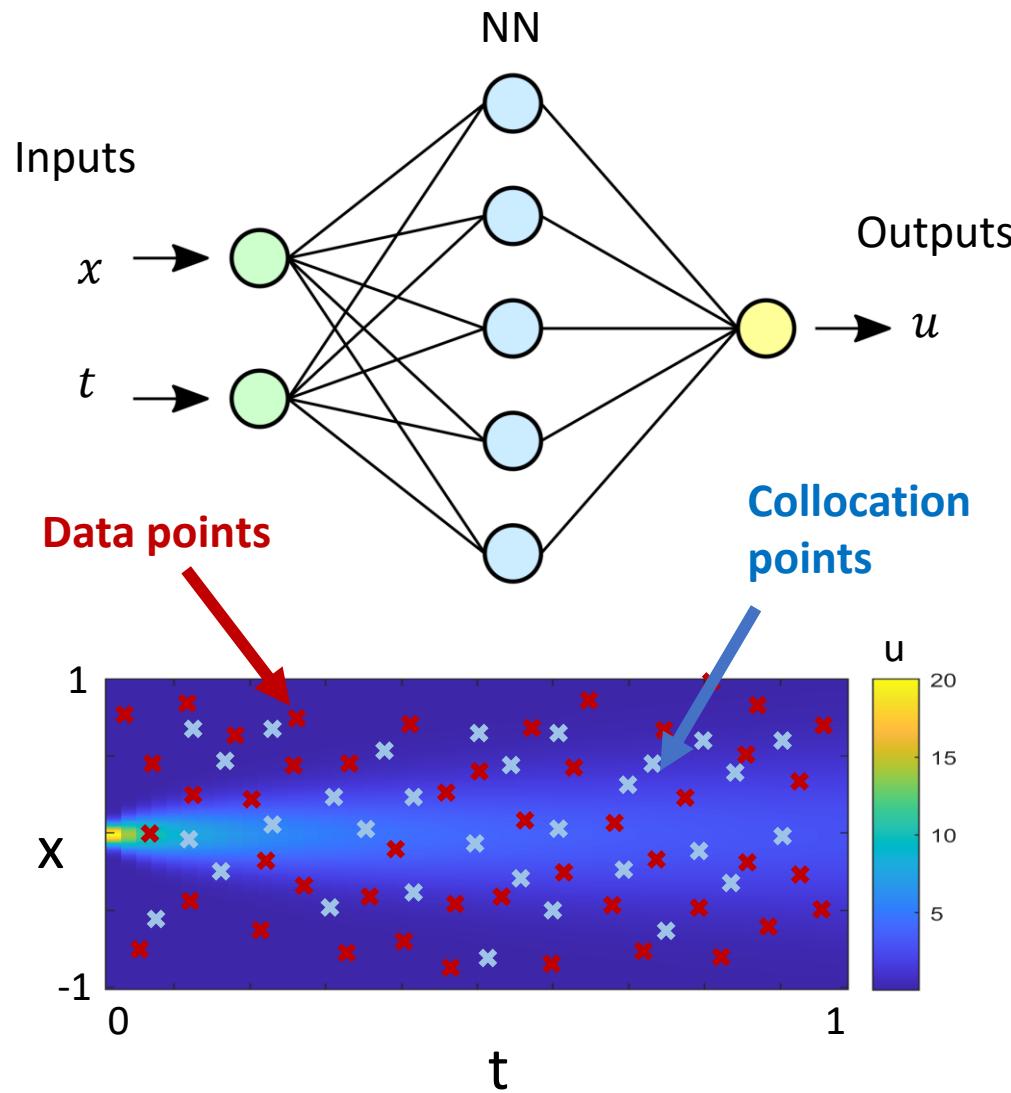
$$\frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2, \quad \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Data points

Equation loss

Collocation points

Data-driven prediction of solution



Given a partial differential equation of a general form:

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

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)

$$MSE = MSE_u + MSE_f,$$

Data loss

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

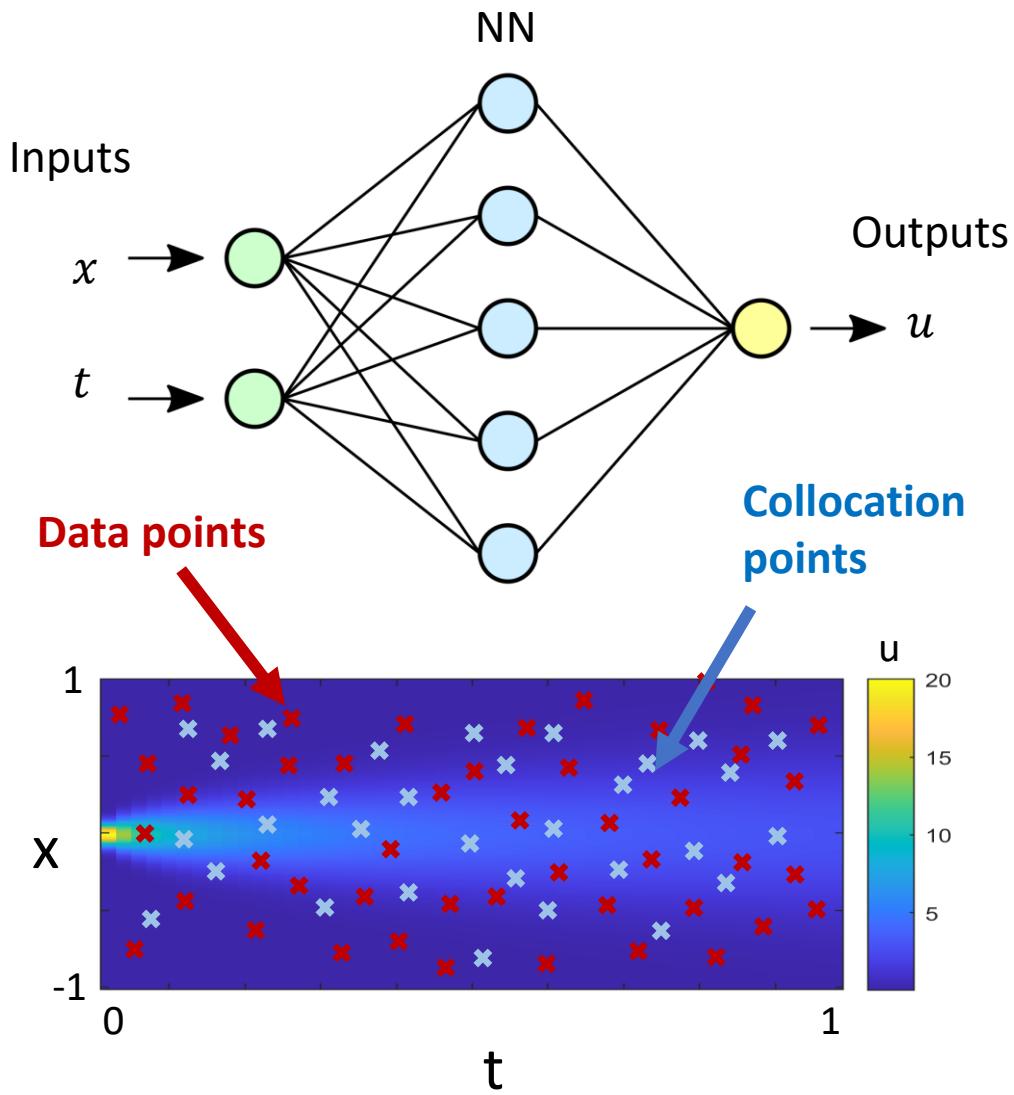
Data points

Equation loss

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

Collocation points

Data-driven discovery of unknown parameters



Given a partial differential equation of a general form:

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

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)

$$MSE = MSE_u + MSE_f,$$

Data loss

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

Data points

Equation loss

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

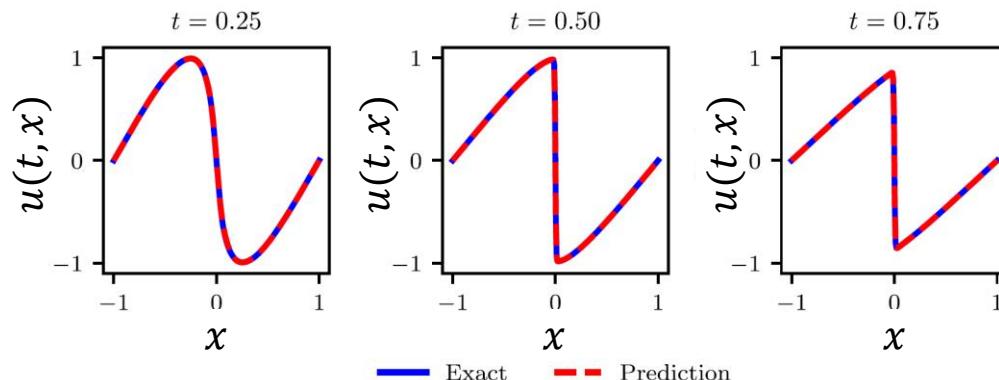
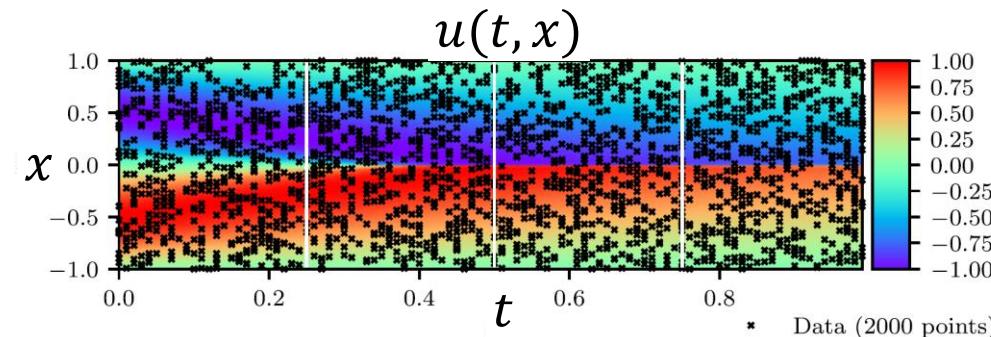
Collocation points

E.g., Burgers' equation (identification)

Given training data of u, t, x find λ_1, λ_2

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

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



Training data (from ground truth):

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

Collocation points:

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

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

E.g., Navier-Stokes equation

Given training data of u, v ,
find λ_1, λ_2

Problem statement

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}),$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}),$$

$$u_x + v_y = 0 \longrightarrow u = \psi_y, \quad v = -\psi_x$$

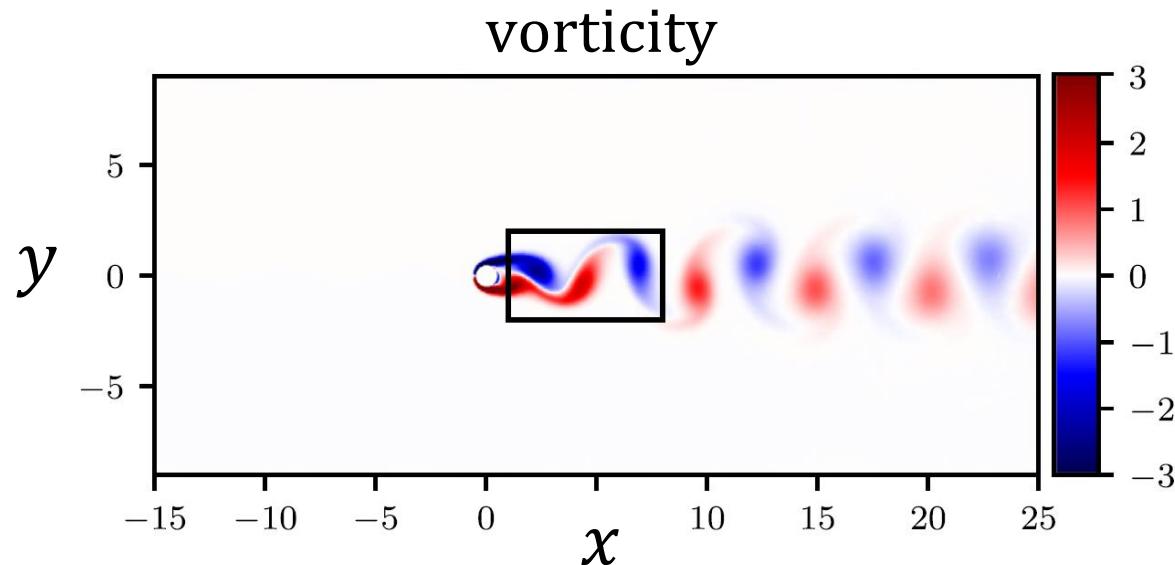
λ_1, λ_2 : unknown parameters to be identified.

ψ : the stream function.

u, v : velocities

p: pressure

Ground truth from numerical simulation:



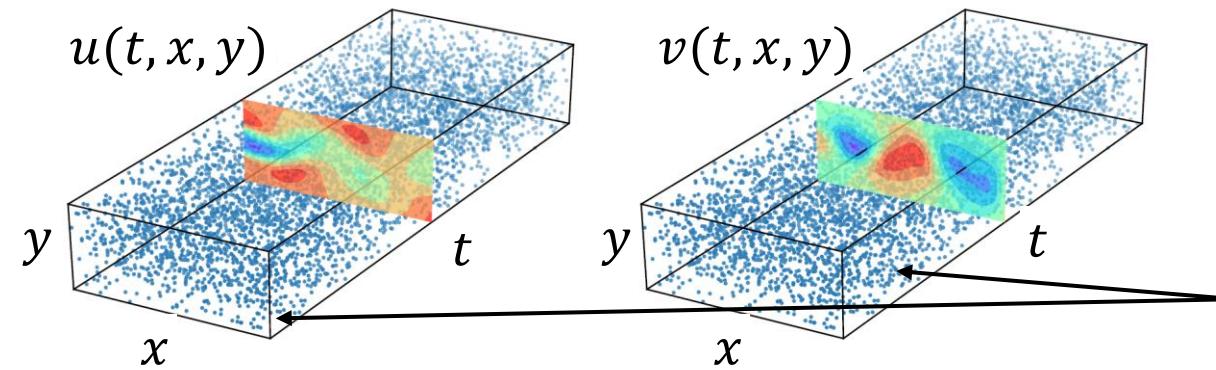
NN input: x, y, t

NN output: ψ, p

NN architecture: 9 layers 20 neurons per layer

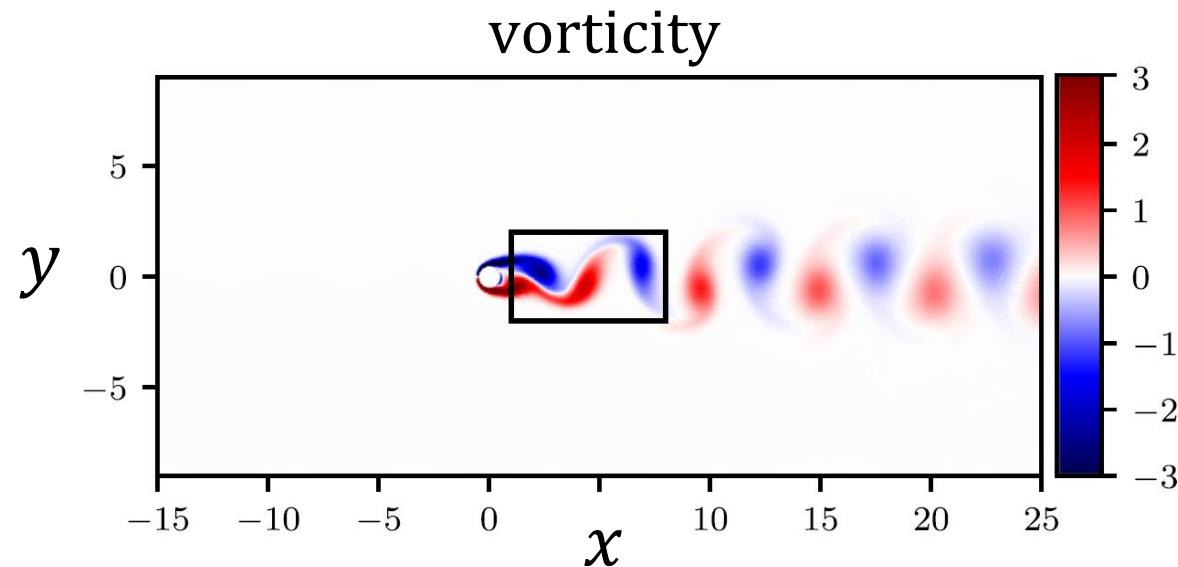
Training parameters: weights, biases, λ_1, λ_2

E.g., Navier-Stokes equation



Given training data of u, v ,
find λ_1, λ_2

Ground truth from numerical simulation:



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

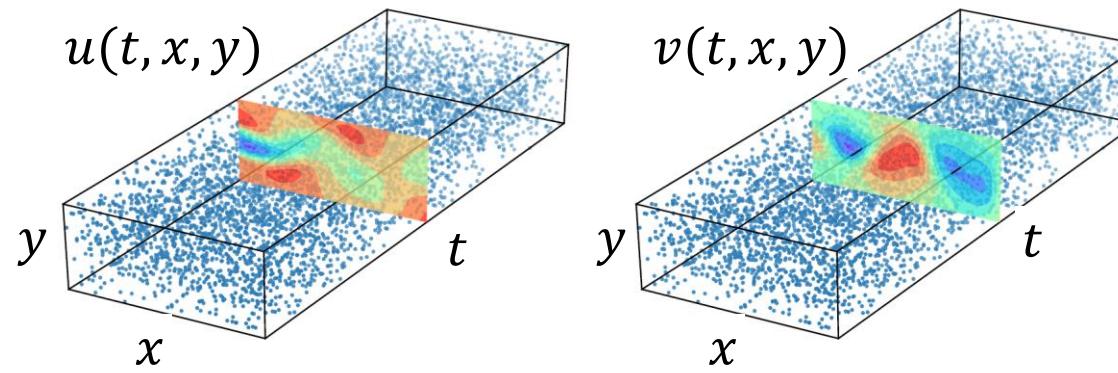
NN input: x, y, t

NN output: ψ, p

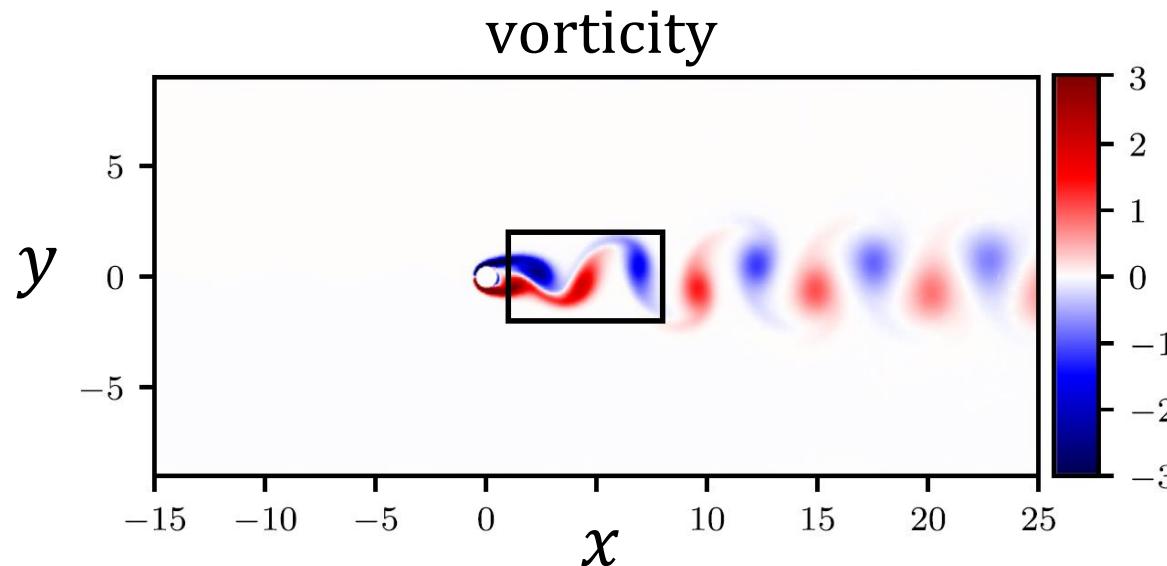
NN architecture: 9 layers 20 neurons per layer

Training parameters: weights, biases, λ_1, λ_2

E.g., Navier-Stokes equation



Ground truth from numerical simulation:



Given training data of u, v ,
find λ_1, λ_2

Training data (from ground truth):

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

Collocation points:

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

Physics equations:

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

Loss function:

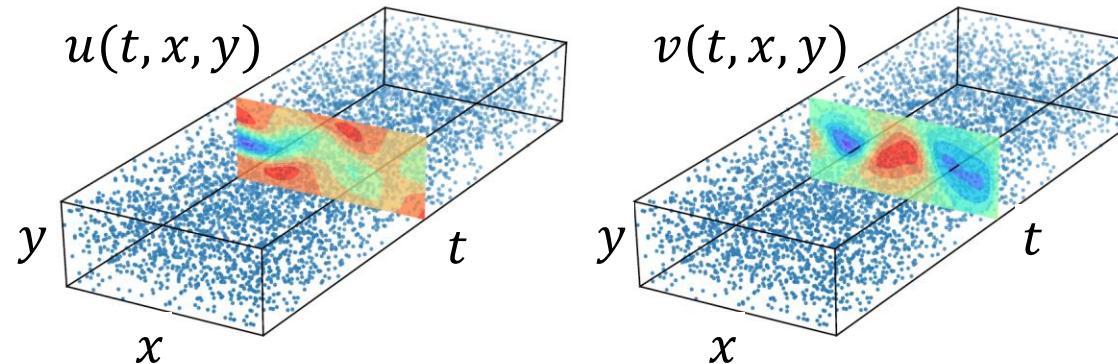
$$MSE := \frac{1}{N} \sum_{i=1}^N \left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2 \right)$$

Data loss

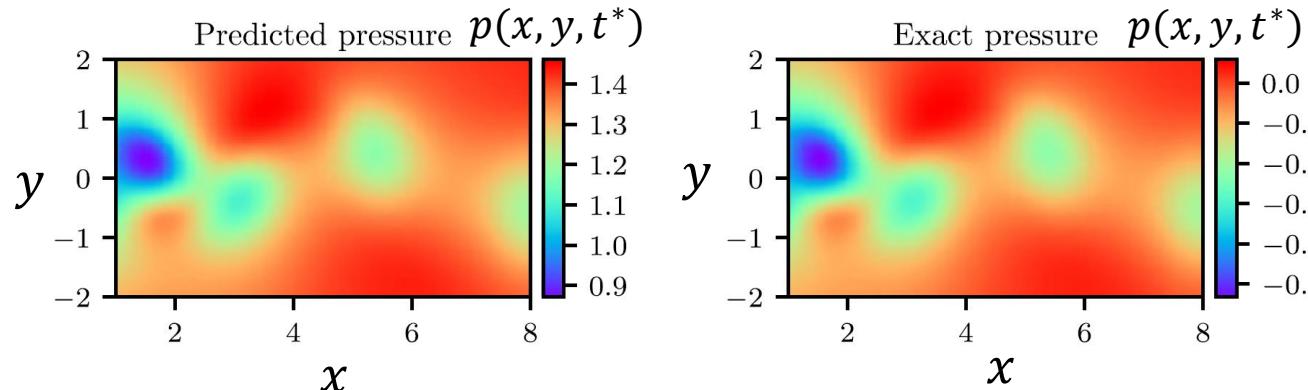
$$+ \frac{1}{N} \sum_{i=1}^N \left(|f(t^i, x^i, y^i)|^2 + |g(t^i, x^i, y^i)|^2 \right)$$

Equation loss

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 λ_1, λ_2

Training data (from ground truth):

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

Collocation points:

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

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:

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

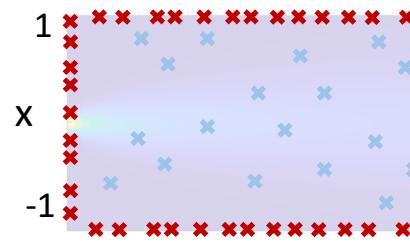
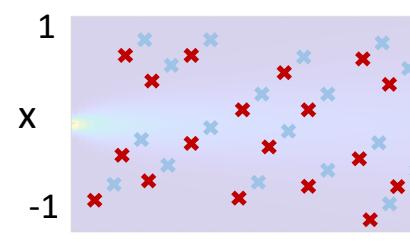
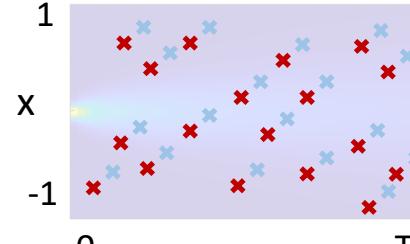
Data loss

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

Equation loss

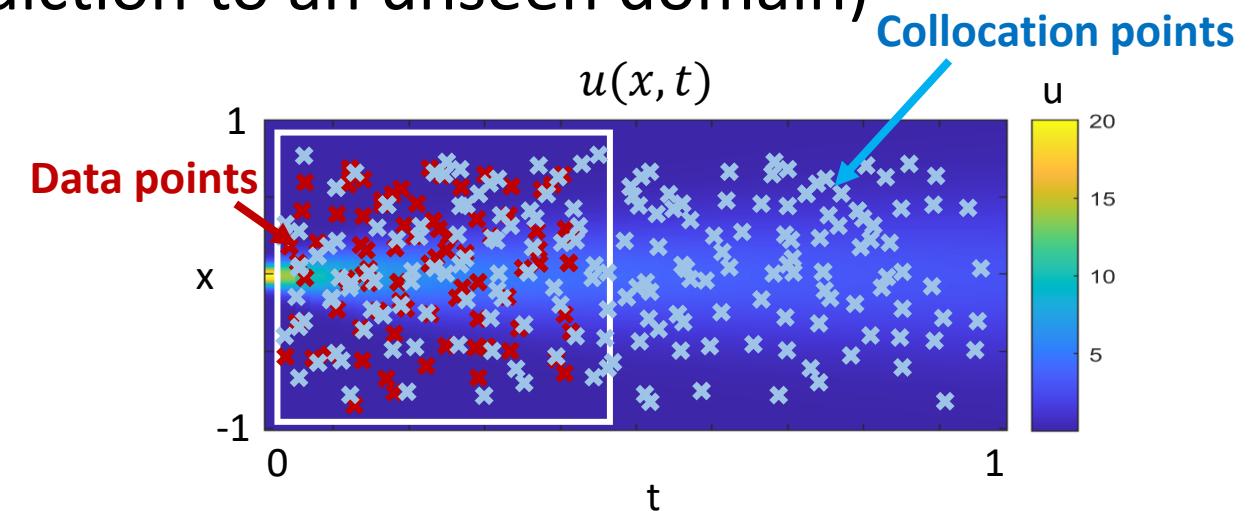
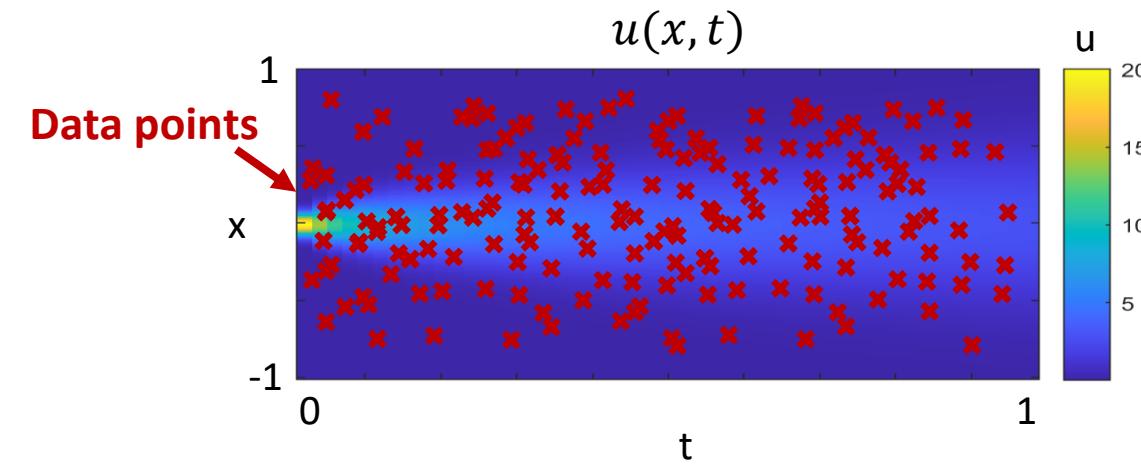
PINN

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

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

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)



Outline

- Part I: Physics-informed neural networks
- Part II: Application to inverse modeling on ice shelves

State parameters that impacts ice dynamics yet difficult, if not impossible, to measure:

Ice viscosity (spatially varying)

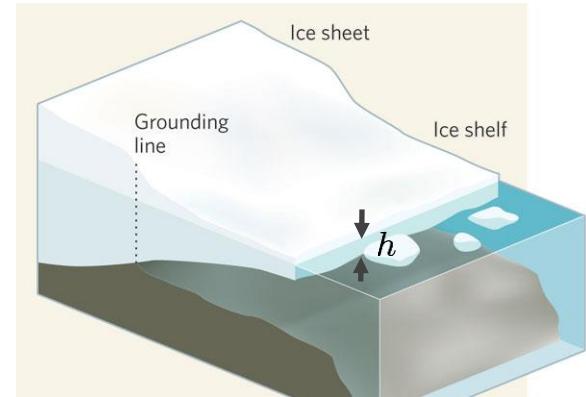
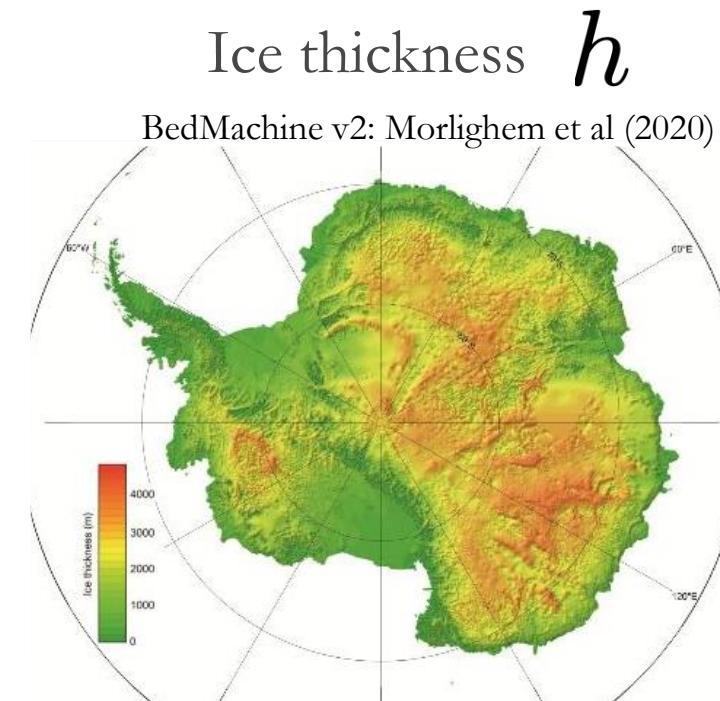
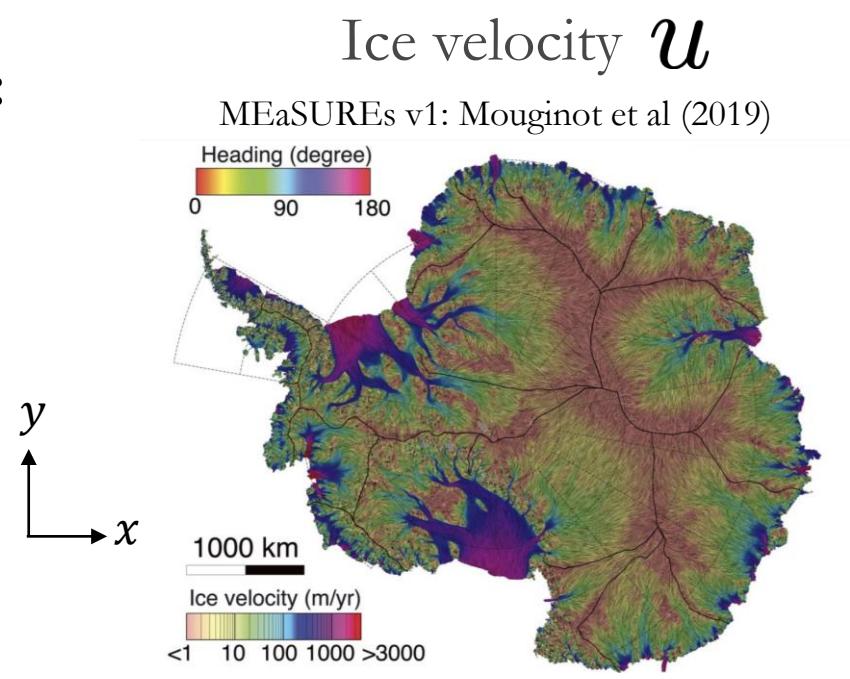
Basal friction (spatially varying)

Need to solve inverse problem to infer them

We will focus on ice shelves to eliminate one unknown (basal friction=0)

How to infer the ice effective viscosity from data?

Data:



Equation:

Ice effective viscosity: impossible to measure

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

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

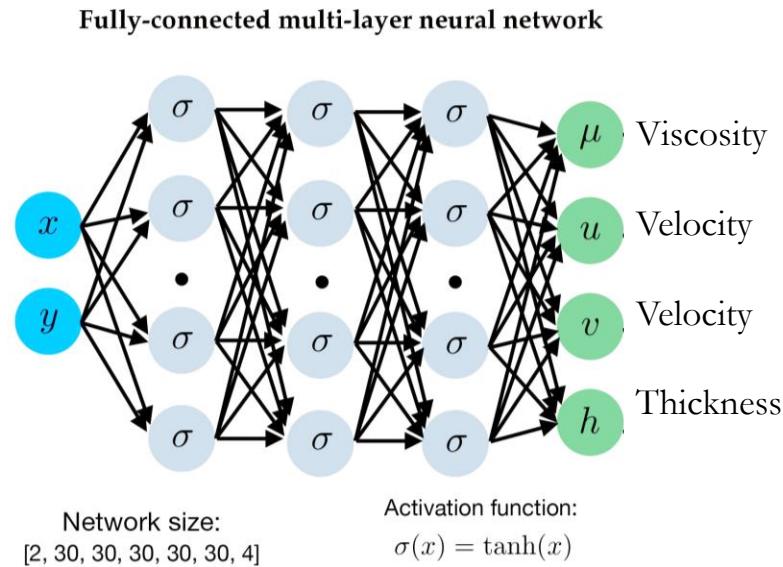
Morland (1987), MacAyeal (1989)

u, v : horizontal velocity
 ρ_i : ice density
 ρ_w : sea water density
 h : ice thickness

} measurable

Ice viscosity inversion using physics-informed NN

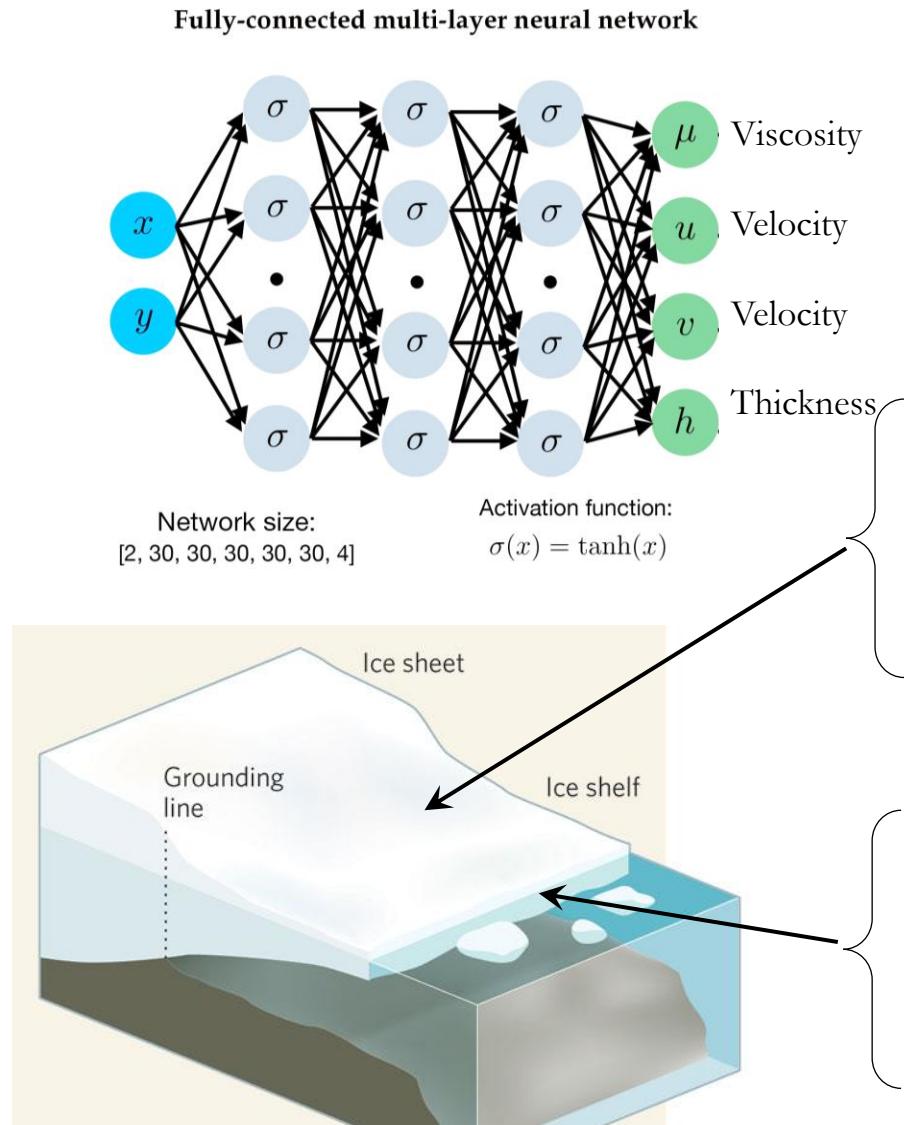
Raissi *et. al.* (2019)



- Utilize the NN's representation power (a universal function approximator)
 $\mu(x, y), u(x, y), v(x, y), h(x, y)$

Ice viscosity inversion using physics-informed NN

Raissi et. al. (2019)



- Utilize the NN's representation power (a universal function approximator)

$$\mu(x, y), u(x, y), v(x, y), h(x, y)$$

Shallow-Shelf Approximation (SSA) equations

$$f_1 = \frac{\partial}{\partial x} \left(4\mu h \frac{\partial u}{\partial x} + 2\mu h \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial y} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial x}$$

$$f_2 = \frac{\partial}{\partial y} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial x} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

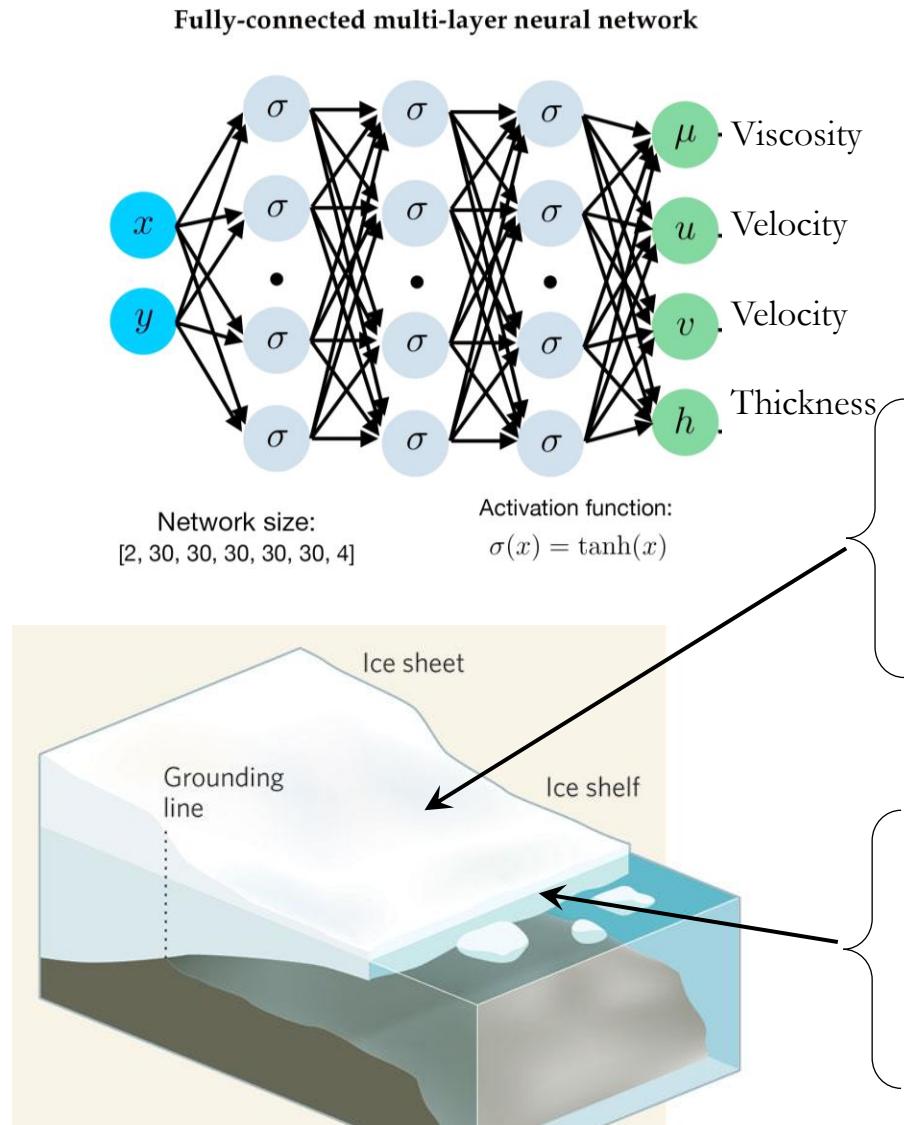
Stress balance at the **ice front**

$$f_3 = 2\mu h \left(2 \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) n_x + \mu h \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_x$$

$$f_4 = \mu h \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_x + 2\mu h \left(\frac{\partial u}{\partial x} + 2 \frac{\partial v}{\partial y} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_y$$

Ice viscosity inversion using physics-informed NN

Raissi et. al. (2019)



- Utilize the NN's representation power (a universal function approximator)

$$\mu(x, y), u(x, y), v(x, y), h(x, y)$$

Shallow-Shelf Approximation (SSA) equations

$$f_1 = \frac{\partial}{\partial x} \left(4\mu h \frac{\partial u}{\partial x} + 2\mu h \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial y} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial x}$$

$$f_2 = \frac{\partial}{\partial y} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial x} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

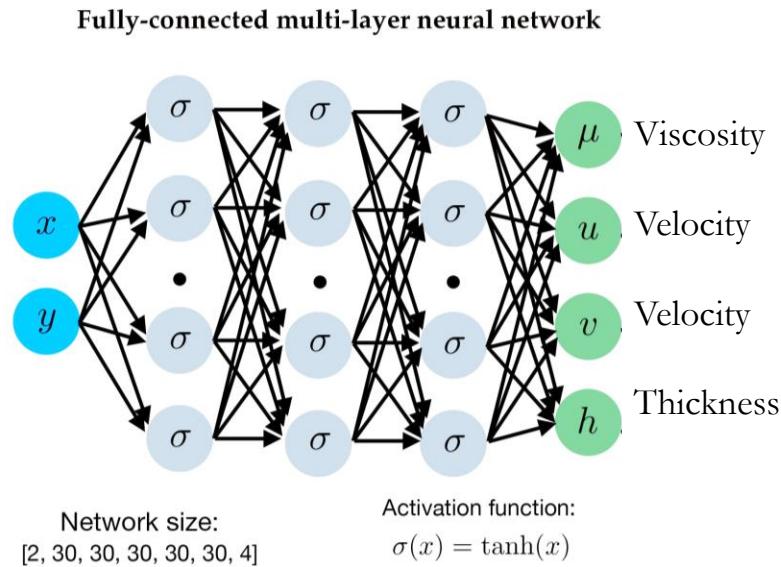
Stress balance at the **ice front**

$$f_3 = 2\mu h \left(2 \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) n_x + \mu h \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_x$$

$$f_4 = \mu h \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_x + 2\mu h \left(\frac{\partial u}{\partial x} + 2 \frac{\partial v}{\partial y} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_y$$

Ice viscosity inversion using physics-informed NN

Raissi et. al. (2019)



Minimize this cost function:

$$J = \int (u_{obs} - u)^2 d\Omega + \left\{ \int (v_{obs} - v)^2 d\Omega + \int (h_{obs} - h)^2 d\Omega + \gamma \sum_i \int f_i d\Omega \right\} \rightarrow \text{Equation misfit}$$

- Utilize the NN's representation power (a universal function approximator)

$$\mu(x, y), u(x, y), v(x, y), h(x, y)$$

Shallow-Shelf Approximation (SSA) equations

$$f_1 = \frac{\partial}{\partial x} \left(4\mu h \frac{\partial u}{\partial x} + 2\mu h \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial y} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial x}$$

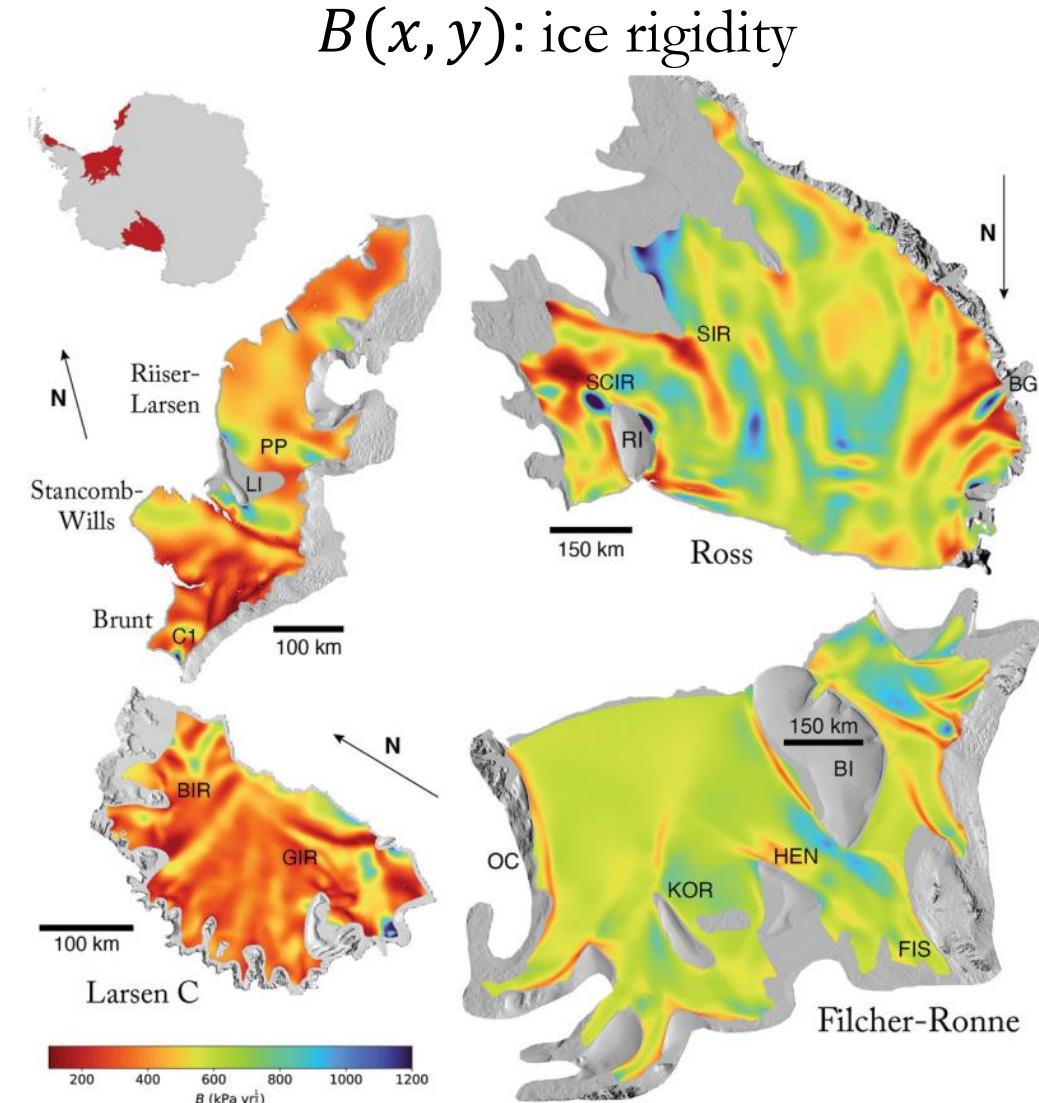
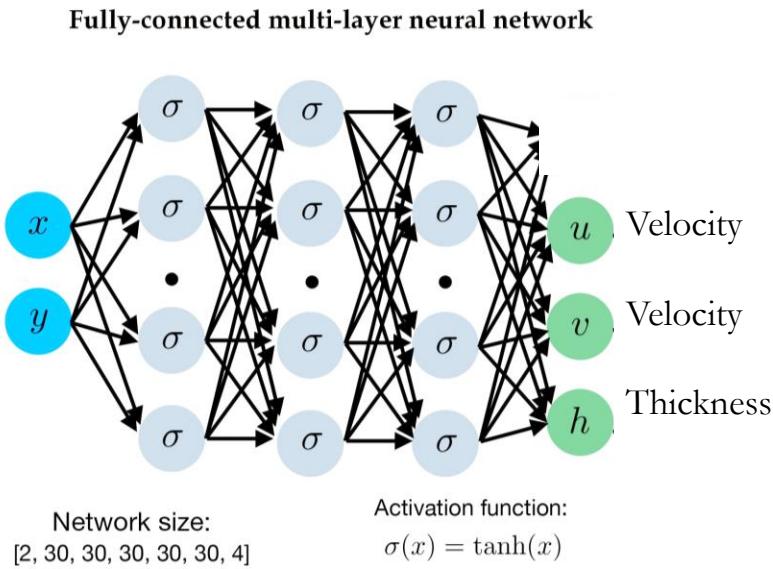
$$f_2 = \frac{\partial}{\partial y} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial x} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

Stress balance at the **ice front**

$$f_3 = 2\mu h \left(2 \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) n_x + \mu h \left(\frac{\partial u}{\partial y} + 2 \frac{\partial v}{\partial x} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_x$$

$$f_4 = \mu h \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_x + 2\mu h \left(\frac{\partial u}{\partial x} + 2 \frac{\partial v}{\partial y} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_y$$

Relevant work using physics-informed NN

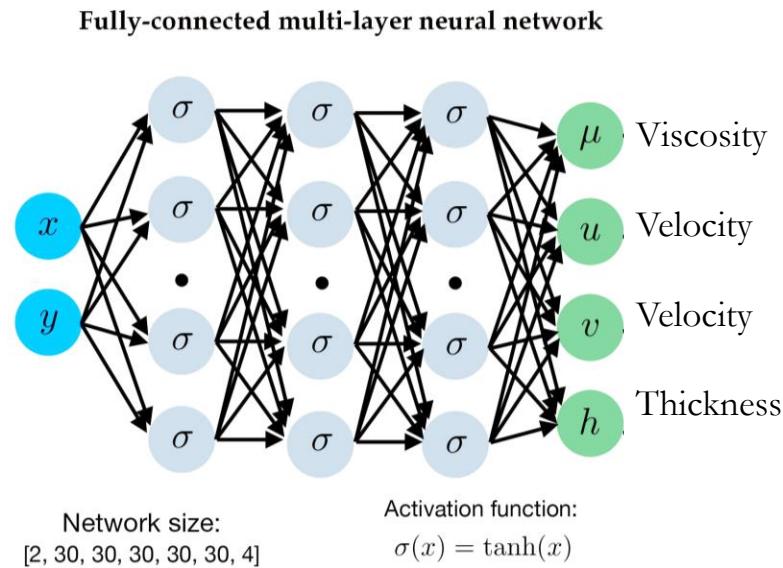


Riel and Minchew (2023):
Use physics-inform NN to infer $B(x, y)$

Glen's Flow Law:

$$\mu = \frac{B}{2} \dot{\epsilon}_e^{\frac{1}{n}-1} \quad n = 3$$

Ice viscosity inversion using physics-informed NN



$\mu(x, y)$: ice effective viscosity

Here, we want to find $\mu(x, y)$

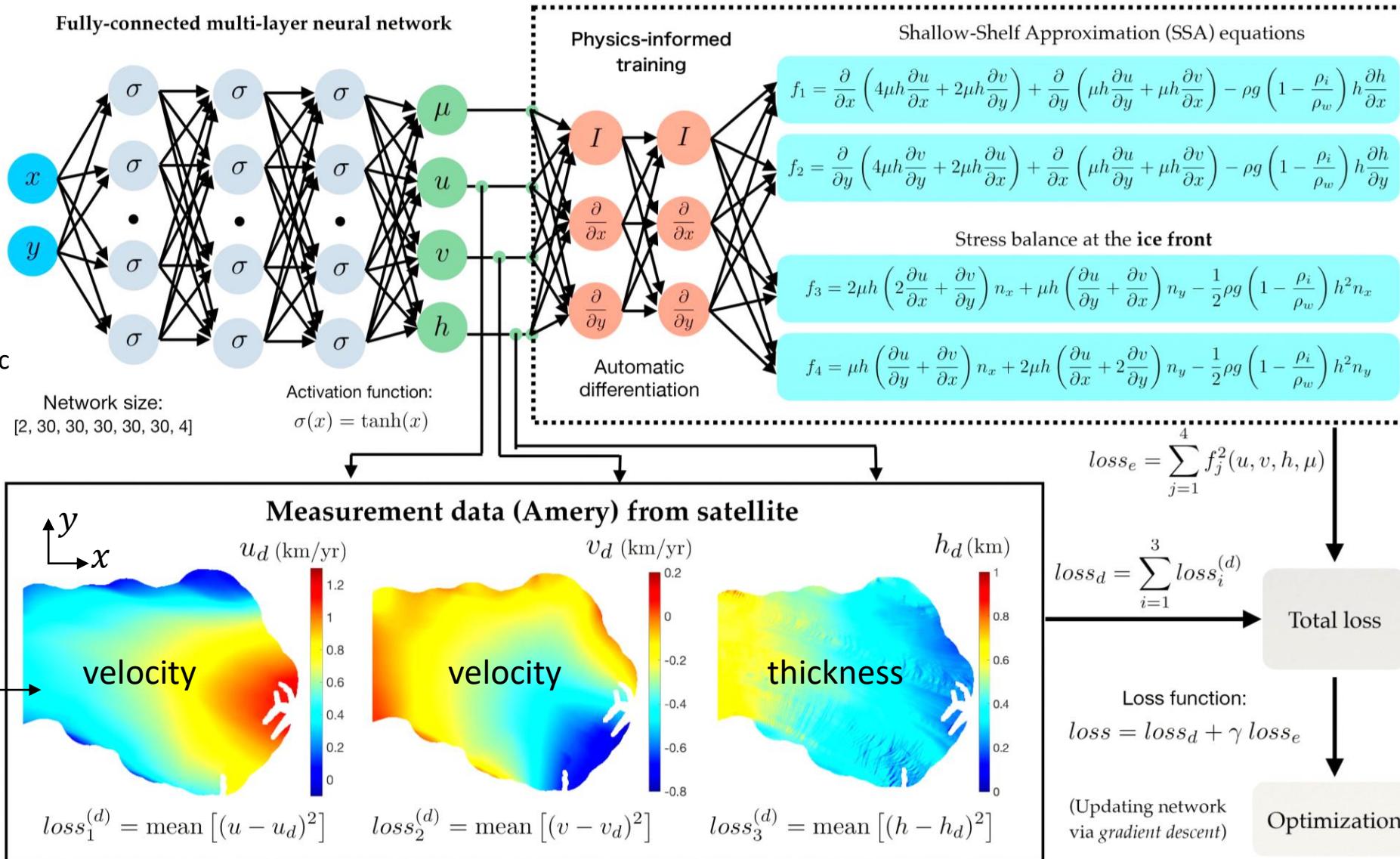
Glen's Flow Law

$$\mu = ? \quad \dot{\varepsilon}_e^{\frac{1}{n}-1} \quad n = 3$$

Ice viscosity inversion using physics-informed NN

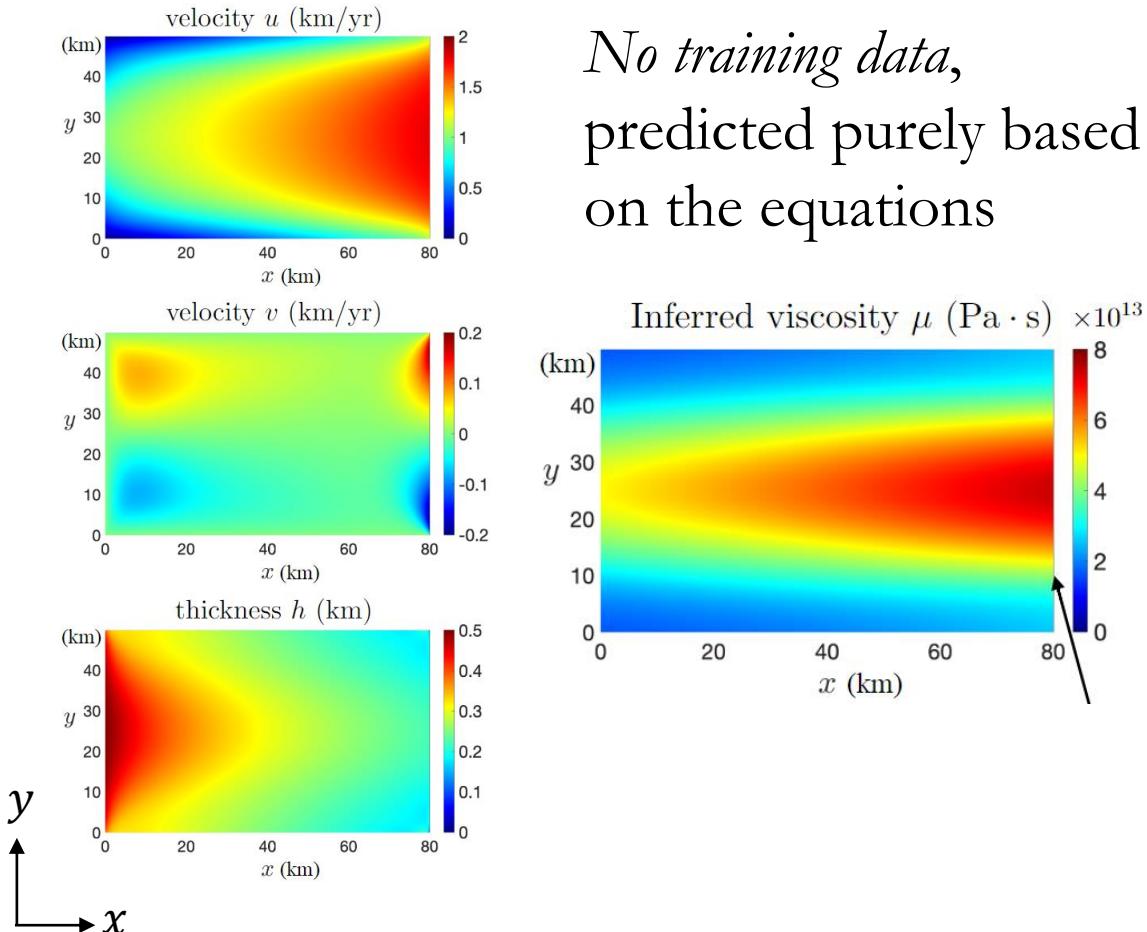


Yongji Wang, postdoc



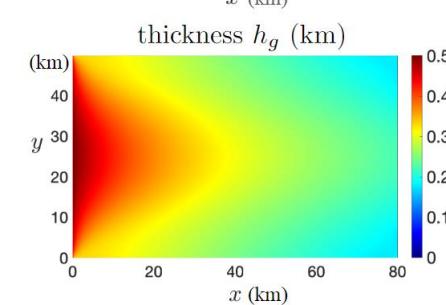
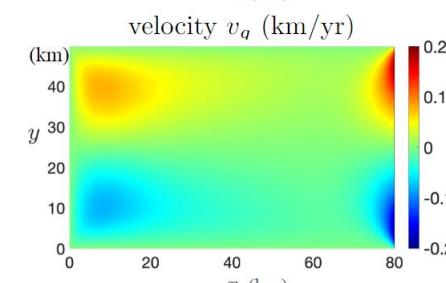
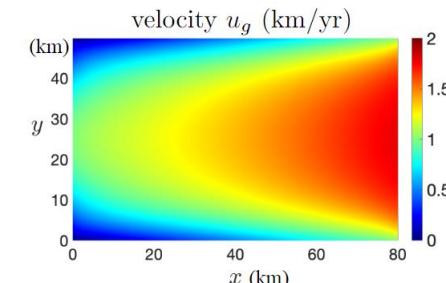
Ice viscosity inversion using PINN (synthetic data)

1. Neural network regression



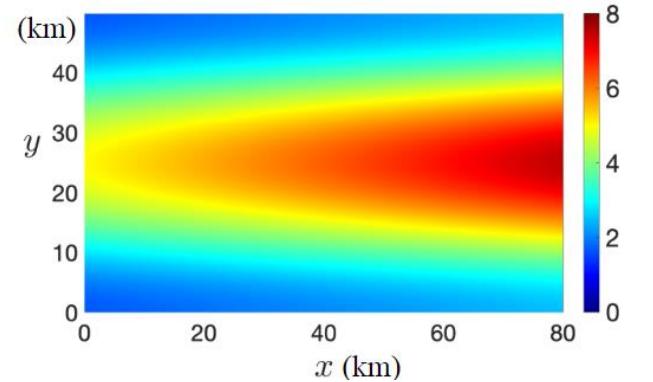
2. Ground truth (synthetic data)

Training data



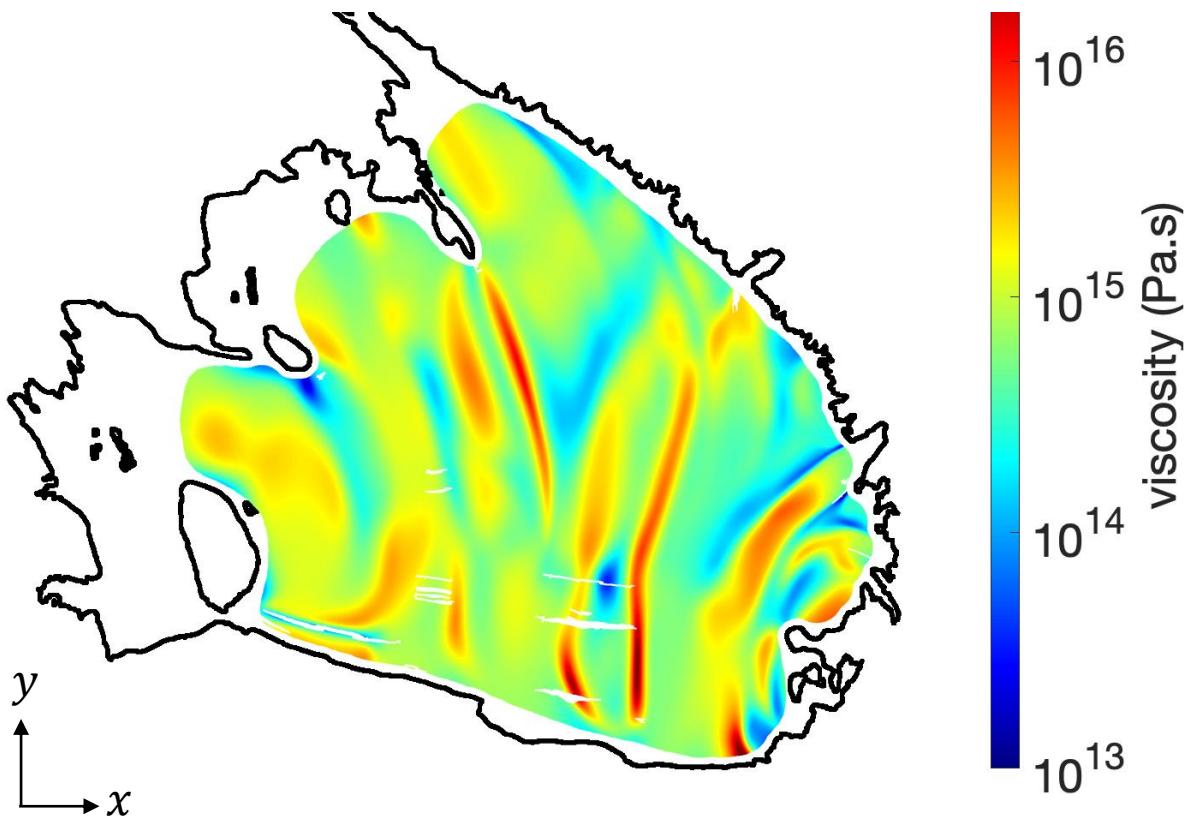
Ground truth viscosity

given viscosity μ ($\text{Pa} \cdot \text{s}$) $\times 10^{13}$



Ice viscosity inversion using PINN

Ice effective viscosity μ

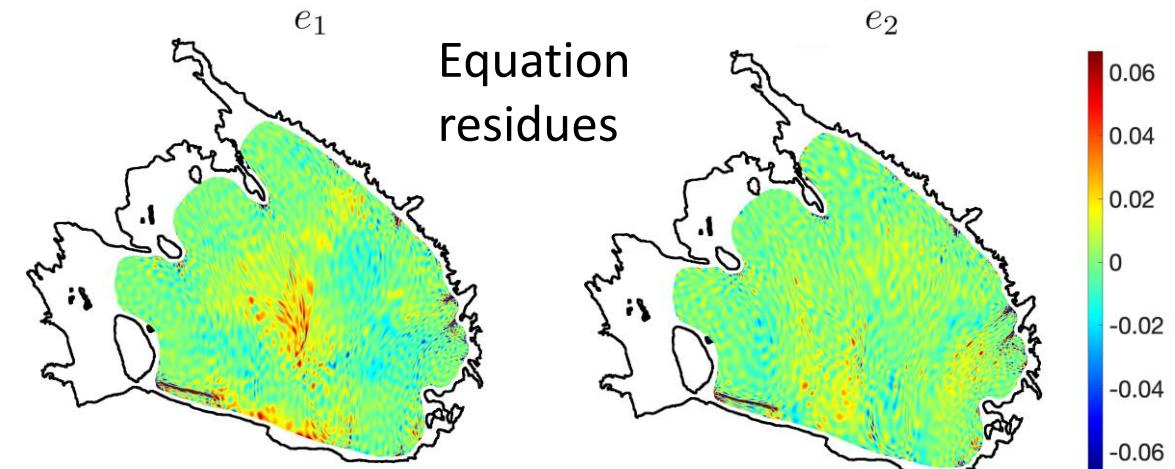


Difficult to measure. Prediction is only possible due to equations!

Governing ice-shelf equations:

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

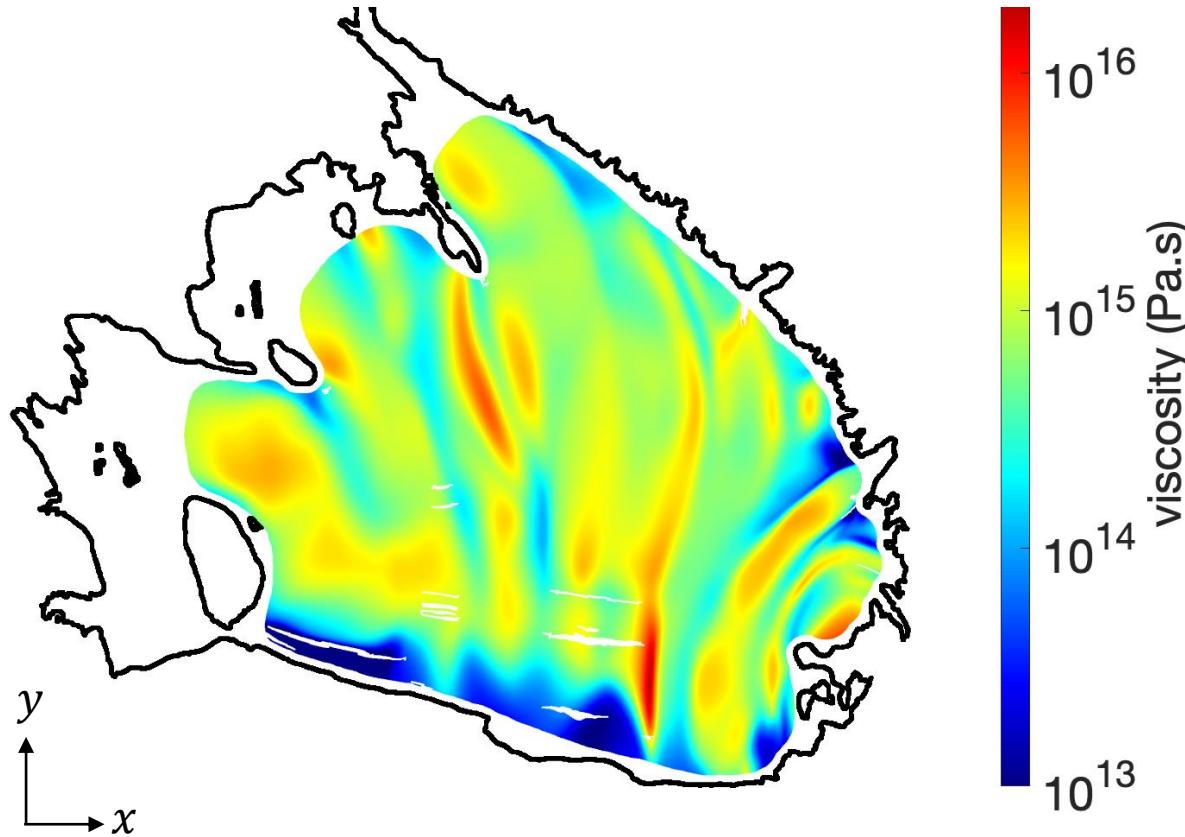
$$e_2 = \frac{\partial}{\partial y} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial x} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - h \frac{\partial h}{\partial y}$$



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

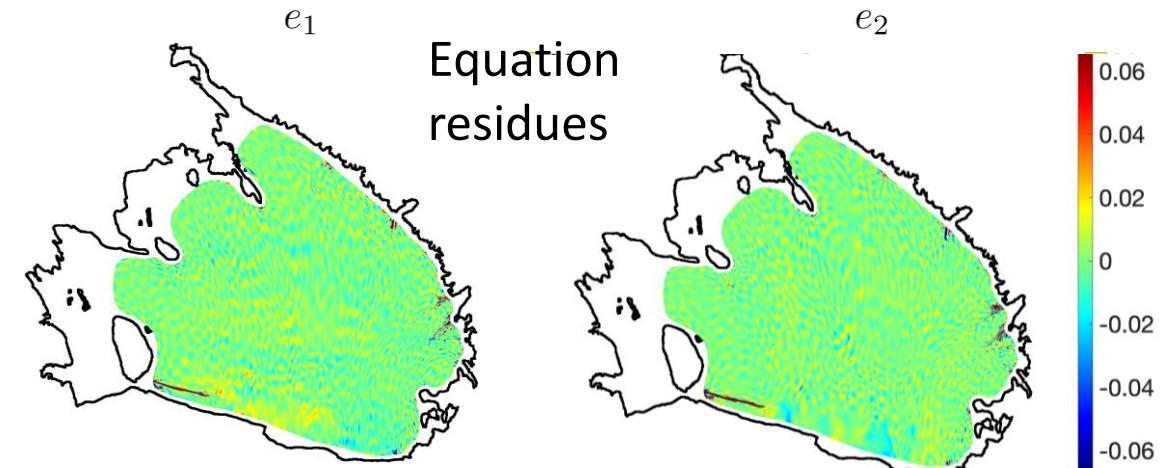


~~Assumption in equation: Isotropic viscosity~~

Anisotropic ice-shelf equations:

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

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



e1 and e2 are further reduced.

Anisotropic viscosity

$$f_1 = \frac{\partial}{\partial x} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho_i g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

$$f_2 = \frac{\partial}{\partial y} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial x} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho_i g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

Isotropic viscosity $\boldsymbol{\tau} = 2\mu\dot{\epsilon}$

$$\begin{bmatrix} \tau_{xx} \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zz} \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & \mu \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{xy} \\ \epsilon_{yy} \\ \epsilon_{zz} \end{bmatrix}$$

Two equations but one unknown

→ Over-constrained

Anisotropy viscosity

$$\begin{bmatrix} \tau_{xx} \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zz} \end{bmatrix} = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} \\ \mu_{31} & \mu_{32} & \mu_{33} & \mu_{34} \\ \mu_{41} & \mu_{42} & \mu_{43} & \mu_{44} \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{xy} \\ \epsilon_{yy} \\ \epsilon_{zz} \end{bmatrix},$$

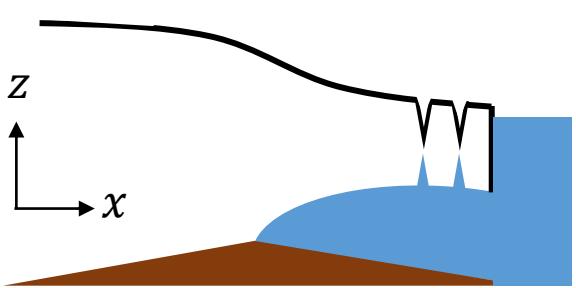
Under-constrained

Anisotropic viscosity

$$f_1 = \frac{\partial}{\partial x} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho_i g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

$$f_2 = \frac{\partial}{\partial y} \left(4\mu h \frac{\partial v}{\partial y} + 2\mu h \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial x} \left(\mu h \frac{\partial u}{\partial y} + \mu h \frac{\partial v}{\partial x} \right) - \rho_i g \left(1 - \frac{\rho_i}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

Vertical crevasses



Orthotropic material

Material properties in three perpendicular directions (axial, radial, and azimuthal) are different. E.g. Wood.



Orthotropic viscosity

$$\begin{bmatrix} \tau_{xx} \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zz} \end{bmatrix} = \begin{bmatrix} \mu_h & 0 & 0 & 0 \\ 0 & \mu_h & 0 & 0 \\ 0 & 0 & \mu_h & 0 \\ 0 & 0 & 0 & \mu_v \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{xy} \\ \epsilon_{yy} \\ \epsilon_{zz} \end{bmatrix}$$

Two equations two unknown

Orthotropic damage (Huth et al; 2020)

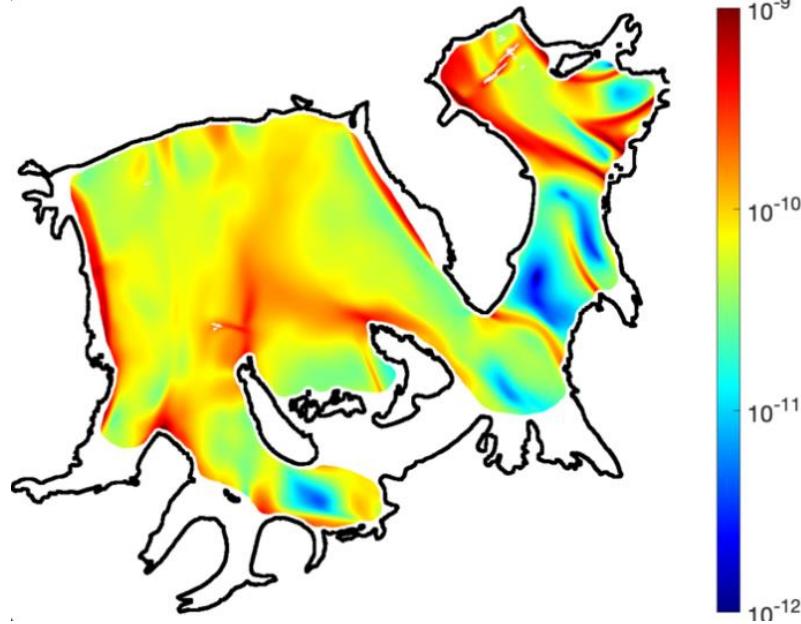
New equations

$$f_1 = \frac{\partial}{\partial x} \left(2\mu_h h \frac{\partial u}{\partial x} + 2\mu_v h \left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right] \right) + \frac{\partial}{\partial y} \left(\mu_h h \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] \right) - \rho_i g \left(1 - \frac{\rho}{\rho_w} \right) h \frac{\partial h}{\partial x}$$

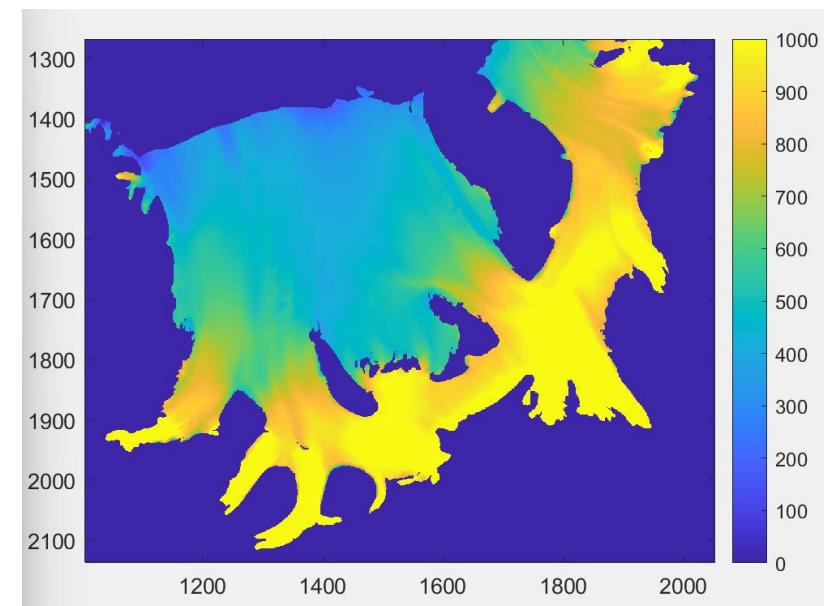
$$f_2 = \frac{\partial}{\partial y} \left(2\mu_h h \frac{\partial v}{\partial y} + 2\mu_v h \left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right] \right) + \frac{\partial}{\partial x} \left(\mu_h h \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] \right) - \rho_i g \left(1 - \frac{\rho}{\rho_w} \right) h \frac{\partial h}{\partial y}$$

Viscosity patterns are not from data

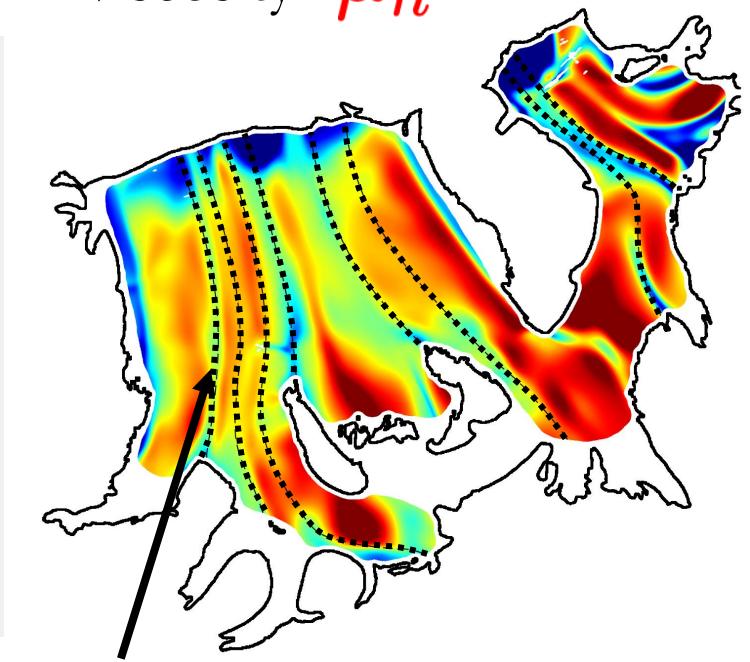
Strain rate (1/s)



Thickness (m)



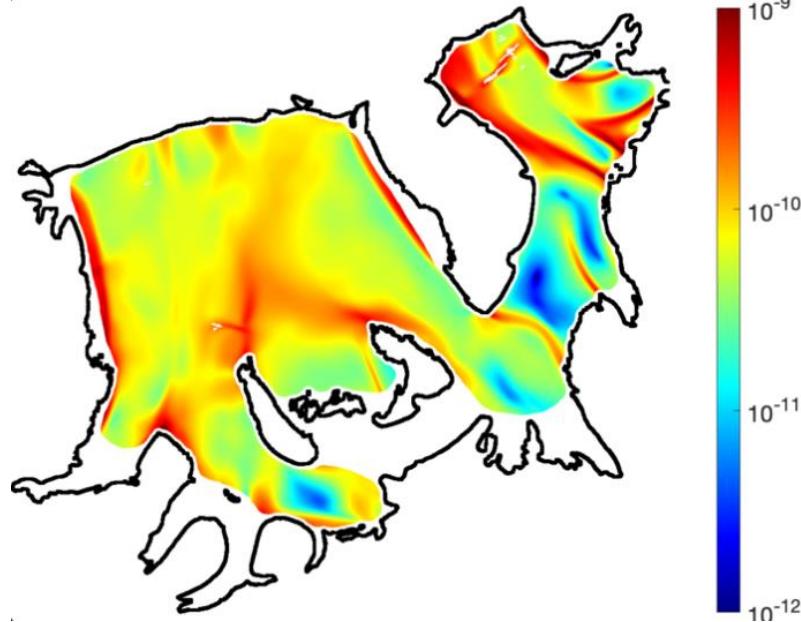
Anisotropic horizontal viscosity μ_h



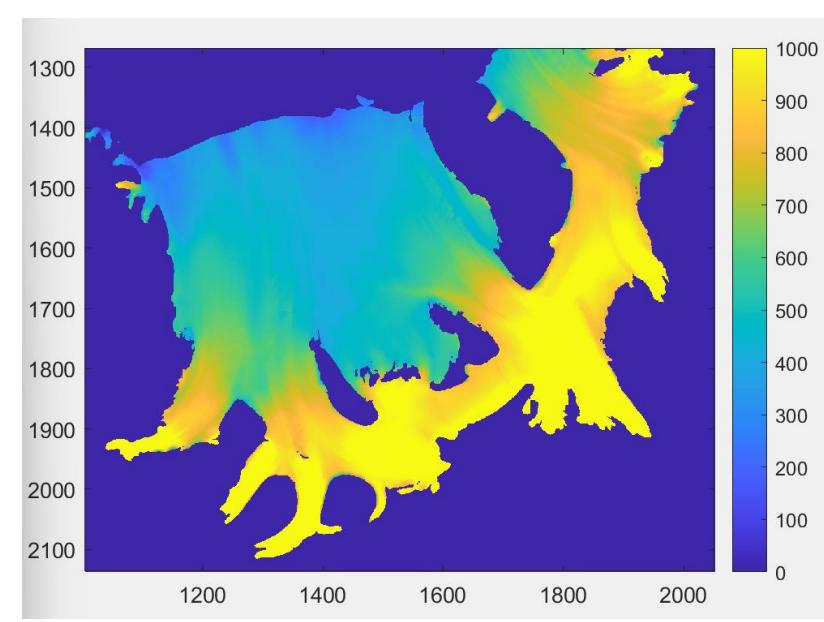
Suture
zones

Viscosity patterns are not from data

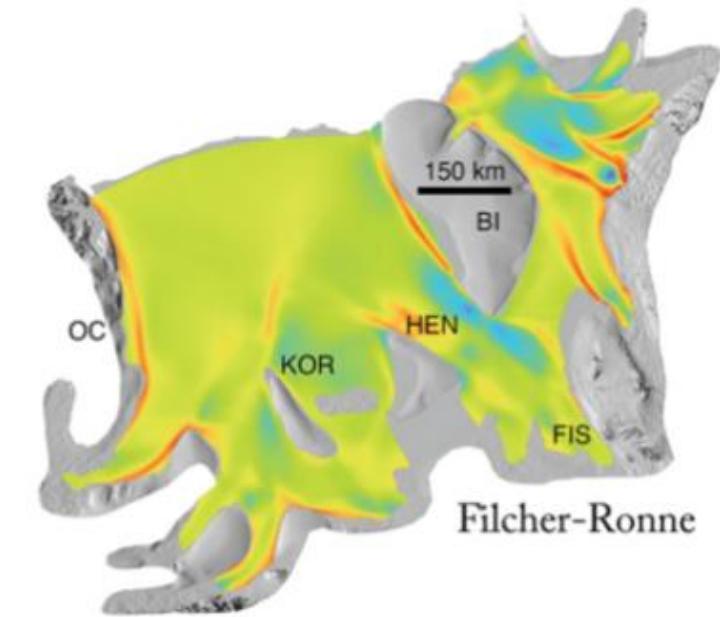
Strain rate (1/s)



Thickness (m)

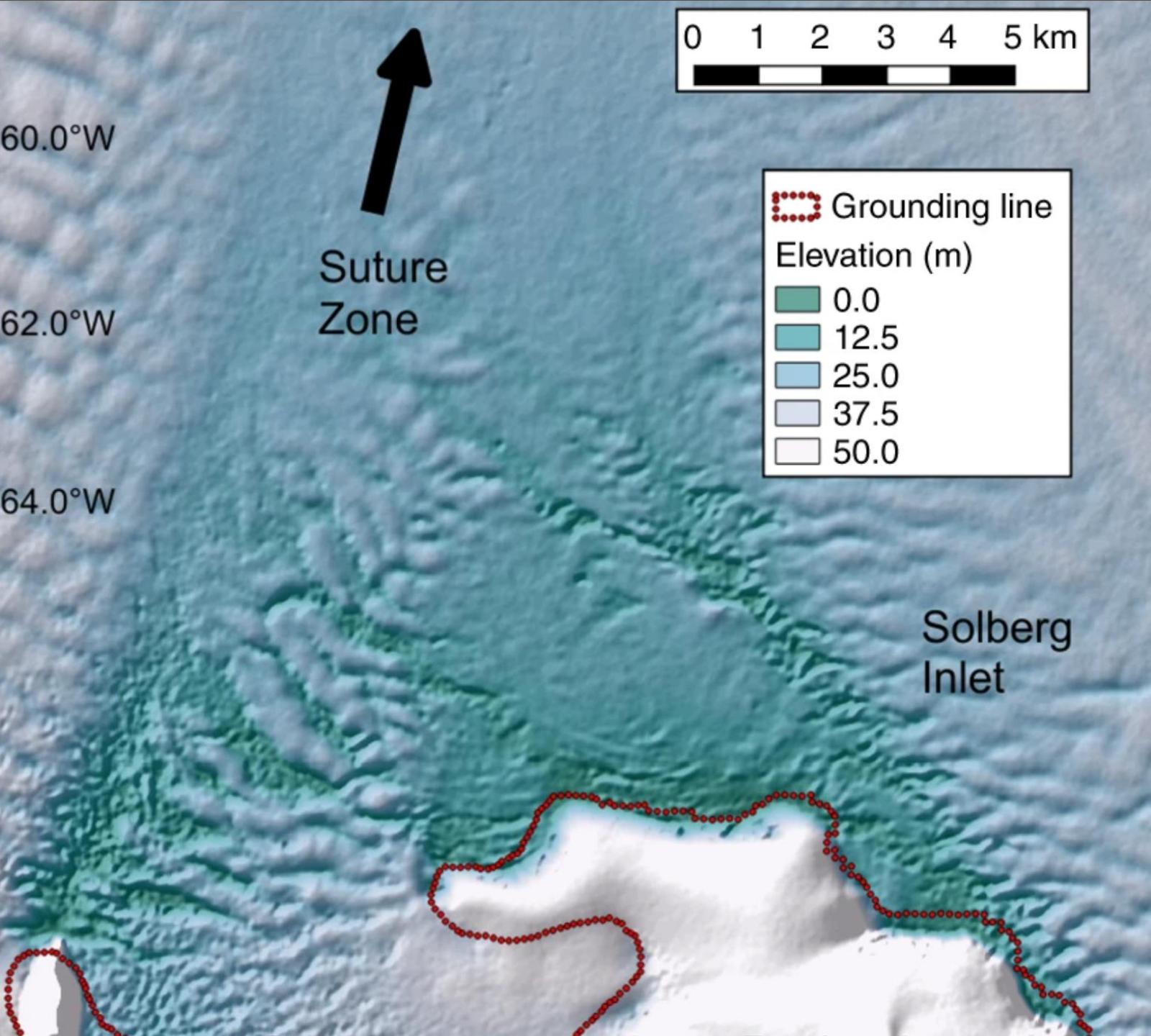
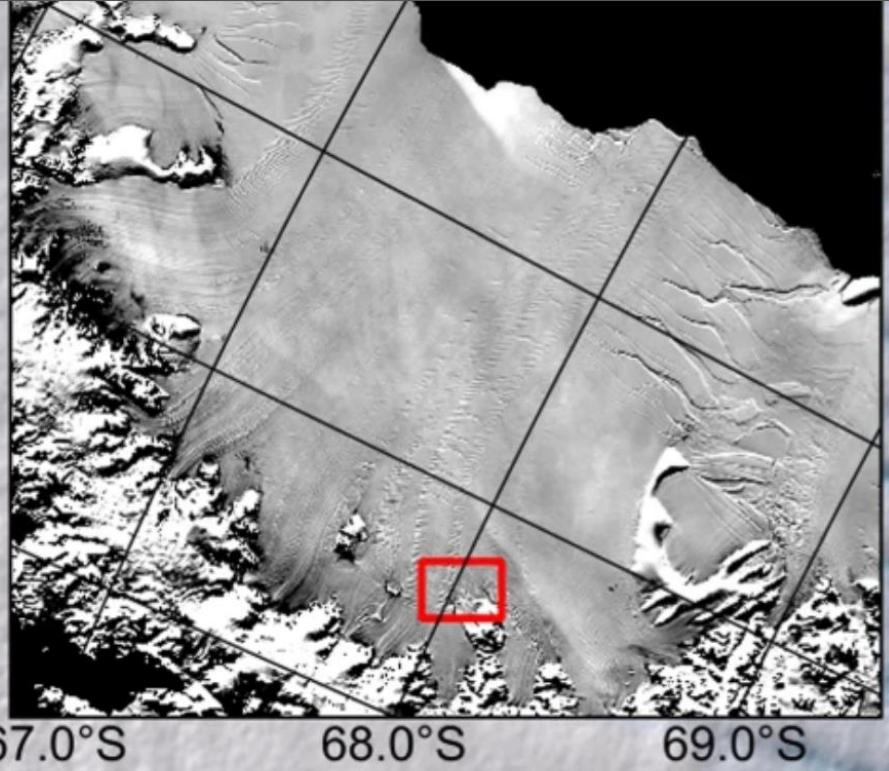


PINN-Ice hardness B
Riel et al. 2023



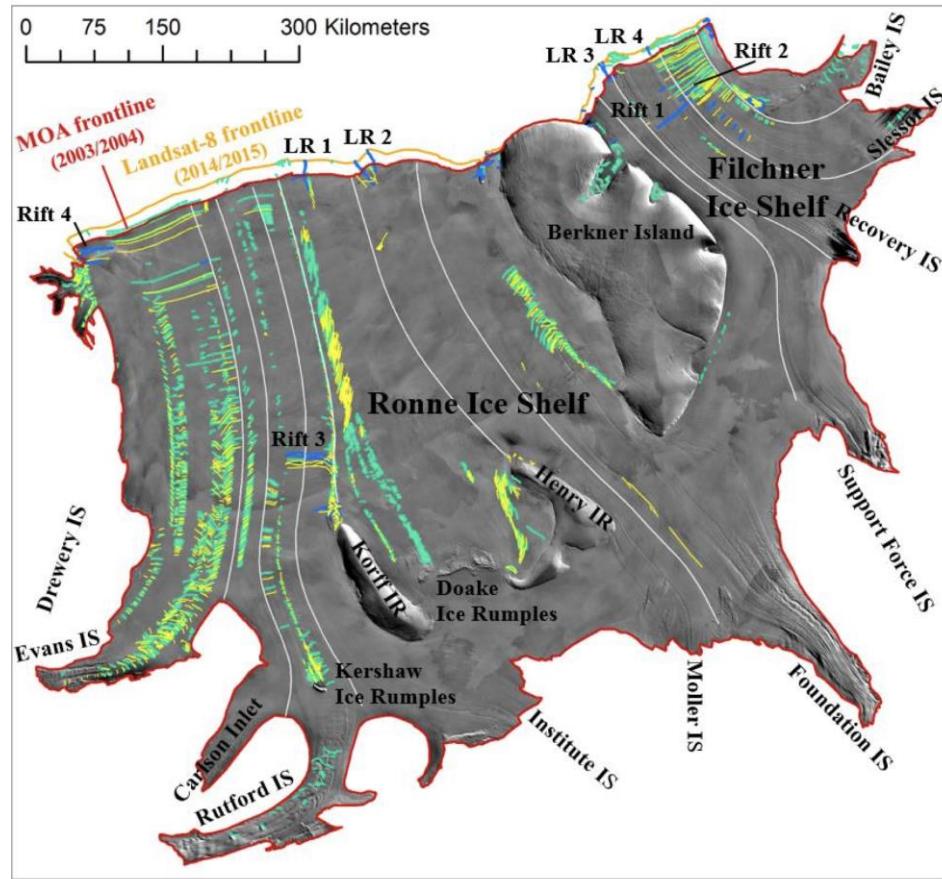
$$\mu = \frac{B}{2} \dot{\varepsilon}_e^{\frac{1}{n}-1}$$

Why does viscosity vary spatially?

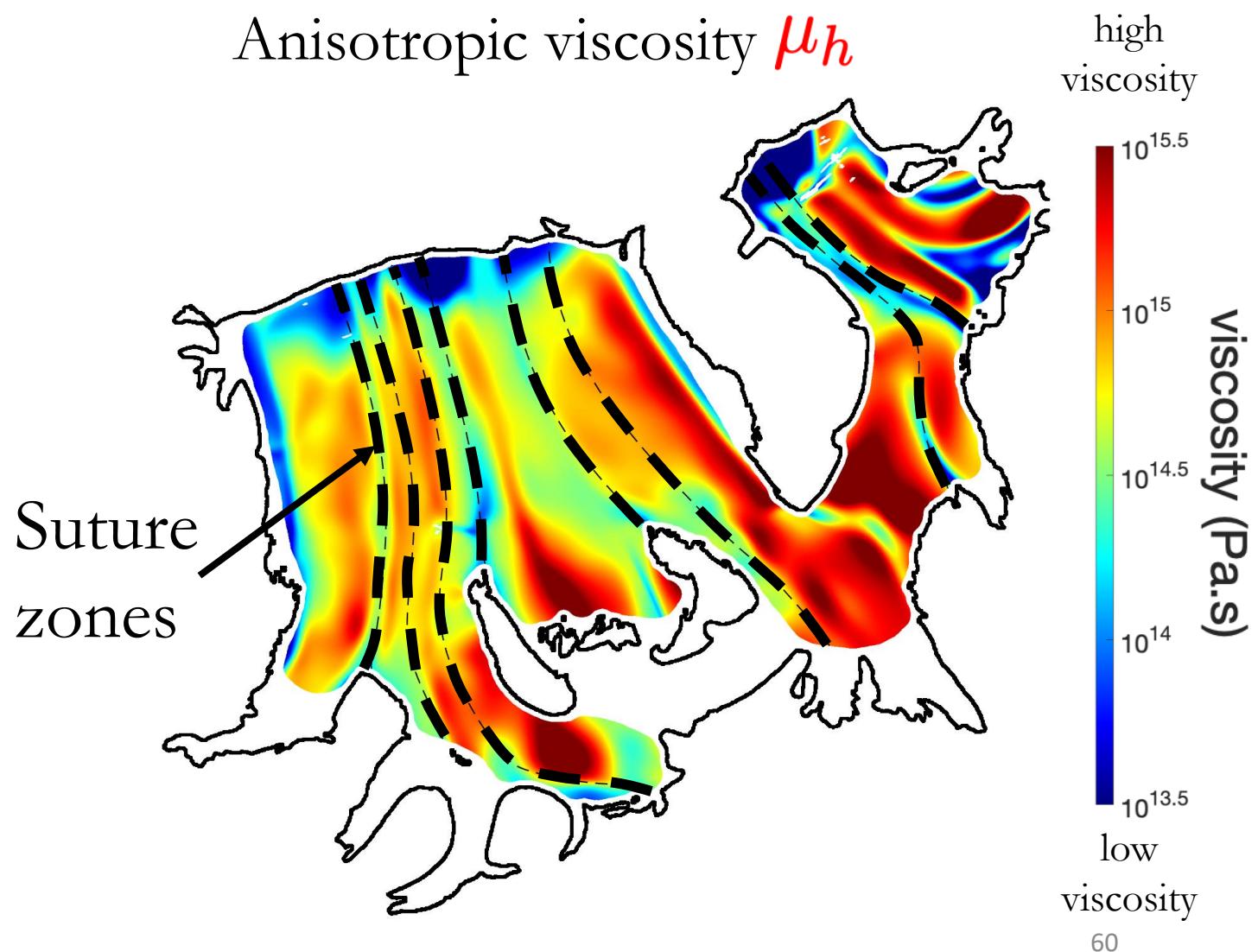


Suture zone matches with low-viscosity areas!

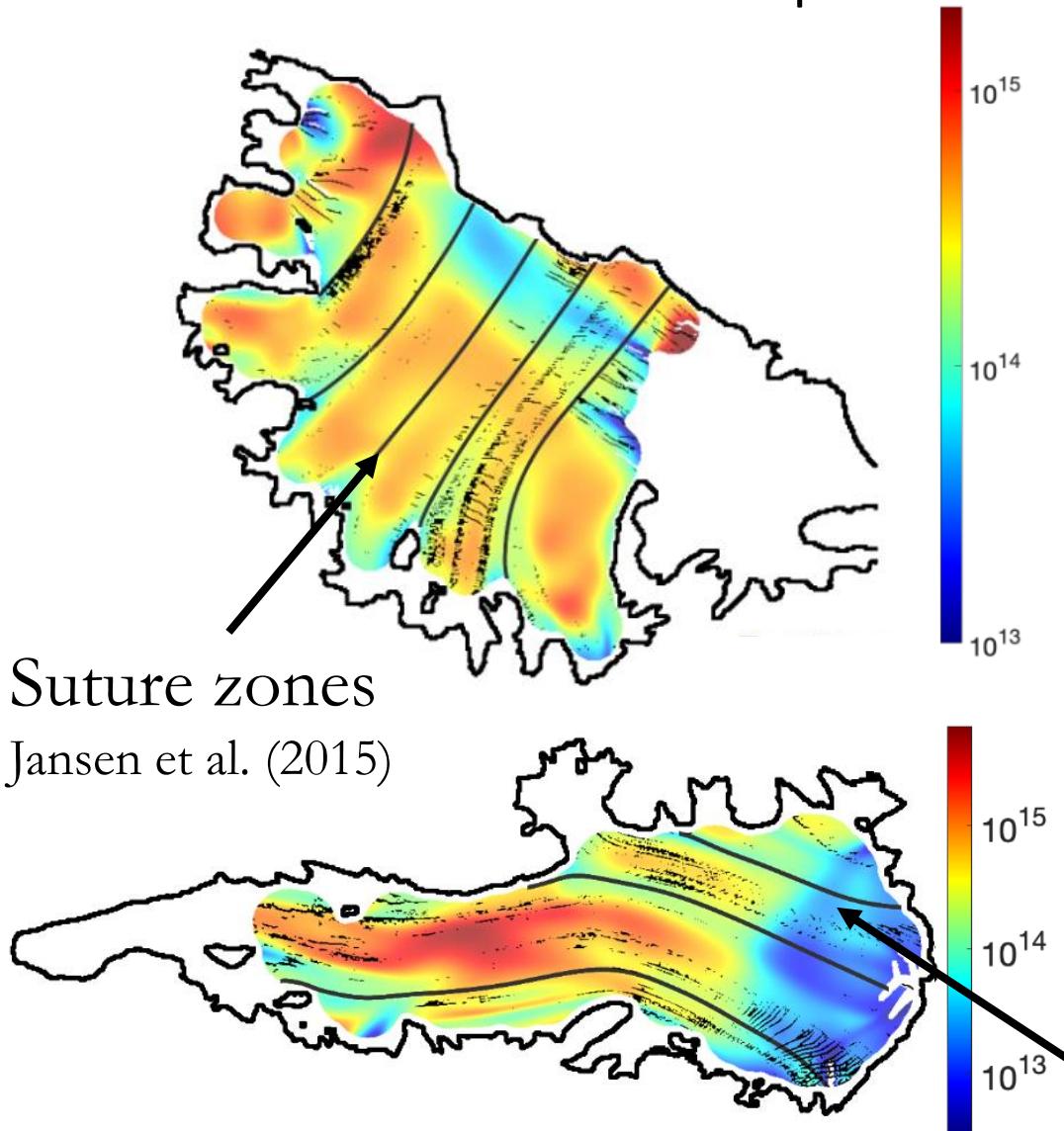
Li et al (2017)



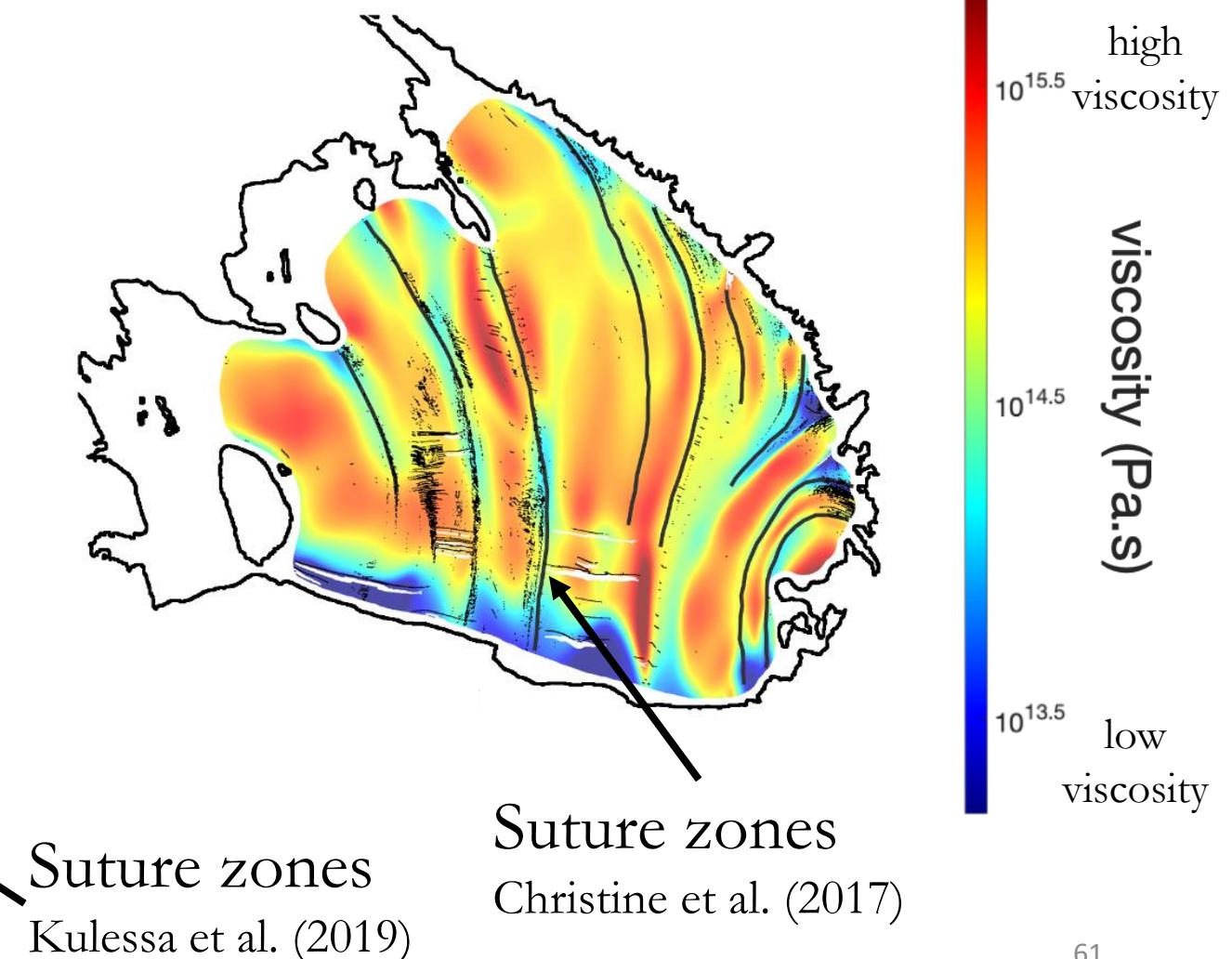
Anisotropic viscosity μ_h



Without representing low-viscosity suture zones in ice-sheet models,
Suture zone matches with low-viscosity areas!
it would be difficult to predict horizontal crack/rift propagation.



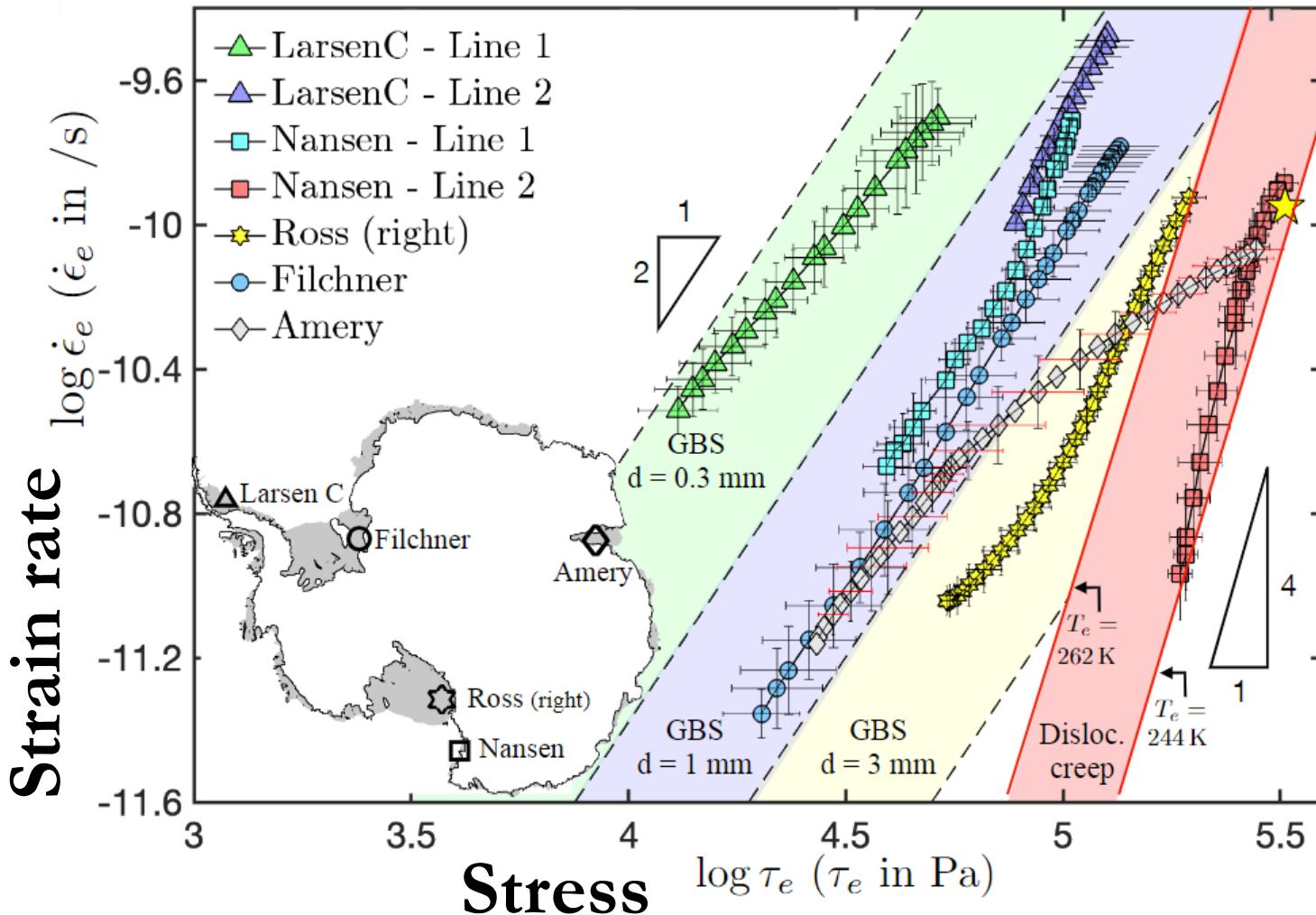
Anisotropic horizontal viscosity μ_h



Suture zones

Kulessa et al. (2019)

In isotropic areas we find excellent fit to strain rate \propto stress n with stress exponent $1 < n < 4$



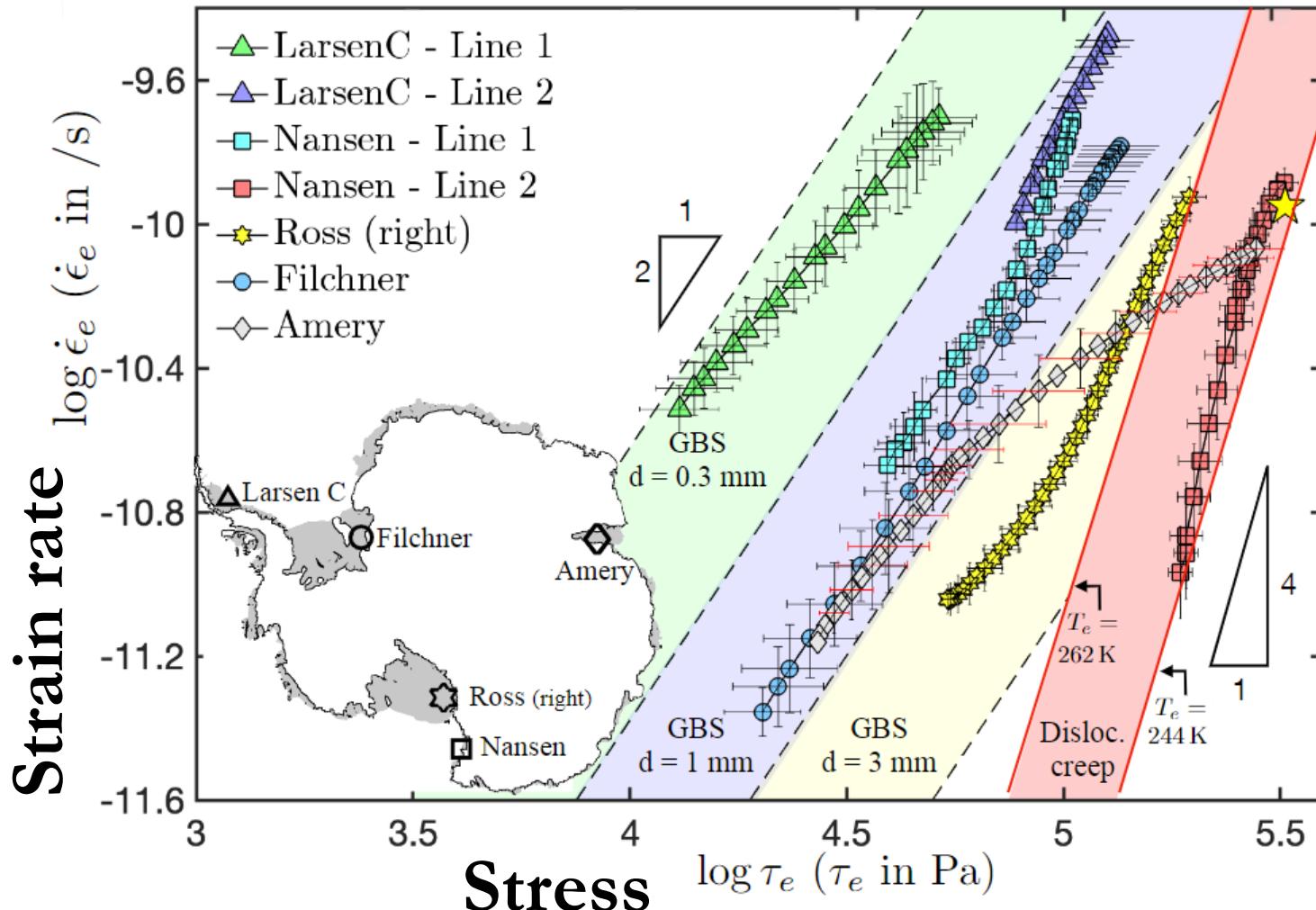
- In conventional ice sheet models, $n=3$ is assumed.
 - But here we observe:
 n increases with stress
- Error bars indicate the standard deviation of 15 repetition of PINN inversions with different random initialization.

$$\dot{\epsilon}_e = \underbrace{A_g d^{-1.4} \tau_e^{1.8} \exp\left(-\frac{Q_g}{RT}\right)}_{\text{Grain boundary sliding (GBS)}} + \underbrace{A_d \tau_e^4 \exp\left(-\frac{Q_d}{RT}\right)}_{\text{Dislocation Creep}}$$

Goldsby and Kohstedt (2001), Kuiper et al. (2020)

A_g, A_d, Q_g, Q_d, R : constants from experiments
 T : temperature (depth averaged)
 d : grain size (fitted parameter)

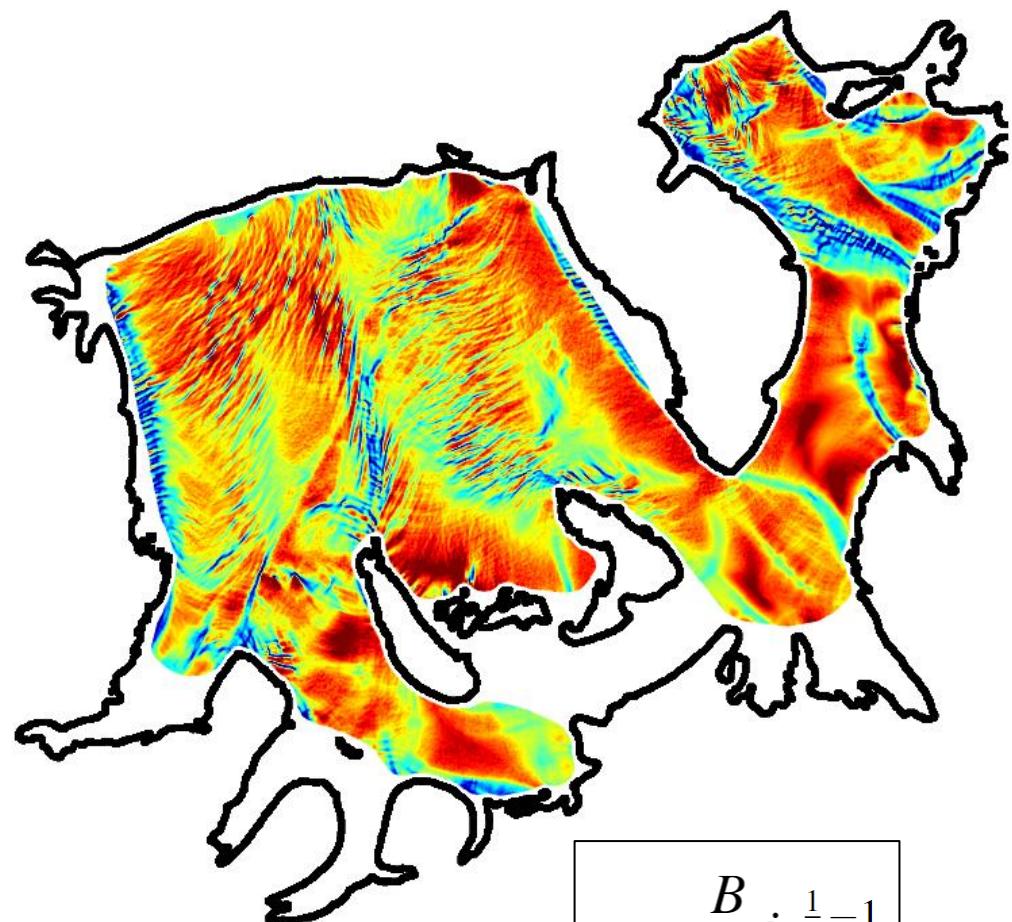
Grain boundary sliding (GBS) Dislocation Creep



- In conventional ice sheet models, $n=3$ is assumed.
- But here we observe:
 n increases with stress
 Composite rheology:
 --- Grain boundary sliding ($n=2$)
 — Dislocation Creep ($n=4$)
- Amery upstream close to $n=1$
 (Field work needed)

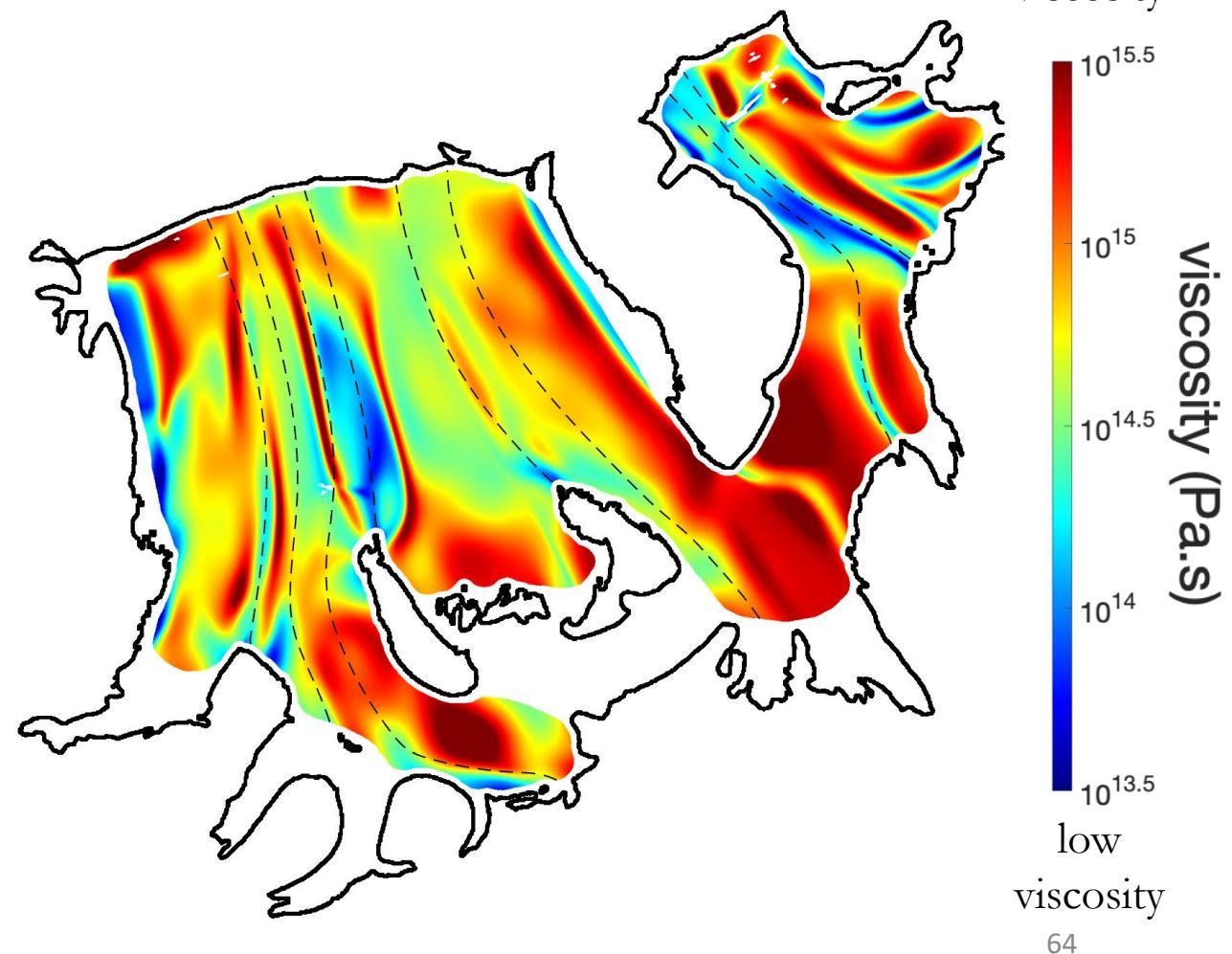
Comparison (isotropic viscosity)

Furst et al (2016) with control method



$$\mu = \frac{B}{2} \dot{\varepsilon}_e^{\frac{1}{n}-1}$$

PINN



high
viscosity

$10^{15.5}$

10^{15}

$10^{14.5}$

10^{14}

$10^{13.5}$

low
viscosity

64

viscosity (Pa.s)

Similarities and differences between PINNs and the adjoint method

Similarities and differences

Given data: $h_{obs}, u_{obs}, v_{obs}$ (real observations)

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

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

with $\mu = \frac{B}{2} \left[\left(u_x^2 + v_y^2 + \frac{1}{4}(u_y + v_x)^2 + u_x v_y \right)^{1/2} \right]^{\frac{1}{n}-1}$

i. Control Method (1993):

$$J = \int (u - u_{obs})^2 dx dy + \int (v - v_{obs})^2 dx dy + \Lambda_1 \int e_1 dx dy + \Lambda_2 \int e_2 dx dy$$

Hard constraints:

1. $h = h_{obs}$
2. $e_1 = e_2 = 0$
3. BCs

Soft constraints:

1. $u \approx u_{obs}$
2. $v \approx v_{obs}$

(The deviation is related to the Lagrange multipliers Λ_1, Λ_2)

Boundary condition at the calving front:

$$e_3 = 2\mu h \left(2 \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) n_x + \mu h \left(\frac{\partial u}{\partial y} + 2 \frac{\partial v}{\partial x} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_x$$

$$e_4 = \mu h \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_x + 2\mu h \left(\frac{\partial u}{\partial x} + 2 \frac{\partial v}{\partial y} \right) n_y - \frac{1}{2} \rho g \left(1 - \frac{\rho_i}{\rho_w} \right) h^2 n_y$$

ii. Physics-Informed Neural Net (2017):

$$J = (1 - \lambda) \sum_i^N [(u^i - u_{obs}^i)^2 + (v^i - v_{obs}^i)^2 + (h^i - h_{obs}^i)^2] + \lambda \sum_i^N [e_1(x^i, y^i)^2 + e_2(x^i, y^i)^2]$$

Hard constraints: None

Soft constraints:

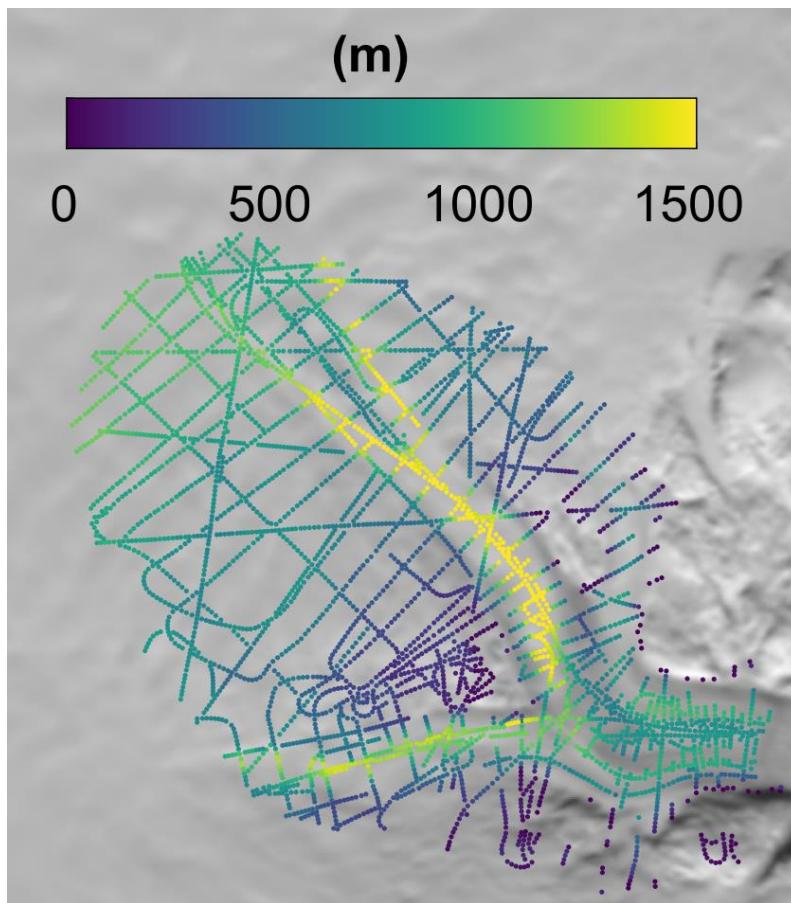
1. $h \approx h_{obs}, u \approx u_{obs}, v \approx v_{obs}$
2. $e_1 \approx e_2 \approx 0$
3. $e_3 \approx e_4 \approx 0$

Notations:

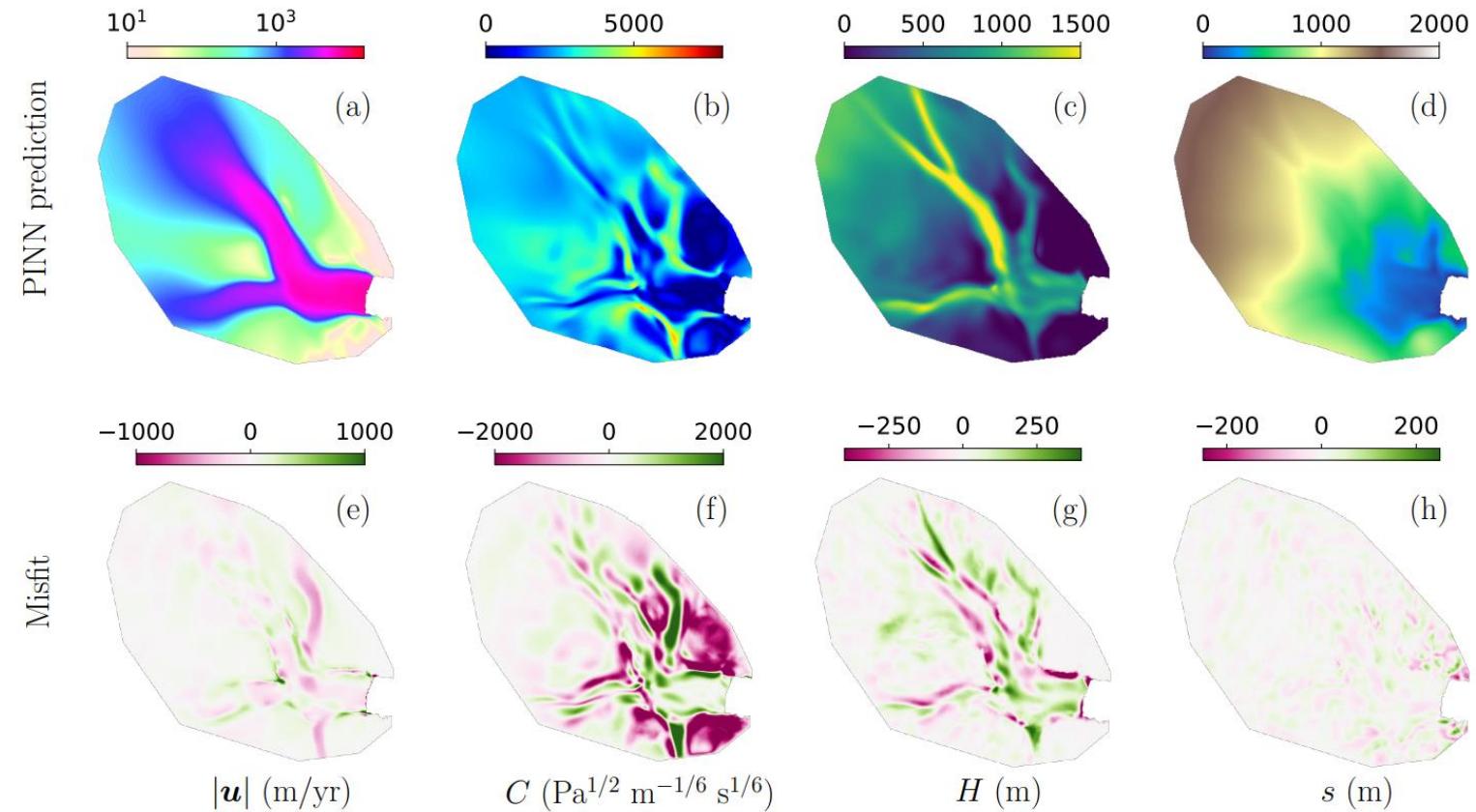
modeled quantity: q
observed quantity: q_{obs}

Thickness data is actually sparse

The actual observed thickness



Inferring both ice thickness and basal friction



Similarities and differences

	MacAyeal's control method (1993)	PINN
θ representation	Numerical grid (or basis functions)	Neural network (e.g. adaptive basis functions)
Optimization	Gradient descent, Adam, L-BFGS...etc	Gradient descent, Adam, L-BFGS...etc
Loss function	Data misfit	Data misfit + PDE misfit
Training	Each iteration solves a forward problem precisely, update θ until Loss is minimized	Each iteration minimizes PDE + data misfit, approximately u , h and θ simultaneously until Loss is minimized
Regularization of θ	Regularization term of θ to avoid data noise of h_{obs} , u_{obs} contaminates θ	PDE loss already de-noises h_{obs} , u_{obs} , no need to further regularize θ via adding a loss term
Theoretical foundation	Mature theory	Theory lacking
Resolution	# of Grid points	>> # of collocation pts if using stochastic sampling
Pitfalls	Representation of fixed resolution	Spectral bias, A NN is often not large enough for an entire ice shelf
Computational time	May not be too different between CPU and GPU?	Runs a lot faster on GPUs than CPUs

Colabs:

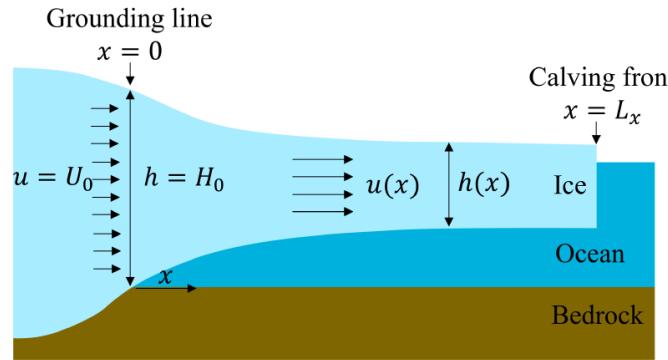
- 1D ice hardness inversion B

https://github.com/YaoGroup/pinn_clusters

- 2D ice viscosity inversion

https://github.com/YaoGroup/DIFFICE_jax

1D Ice viscosity inversion using PINN (synthetic data)



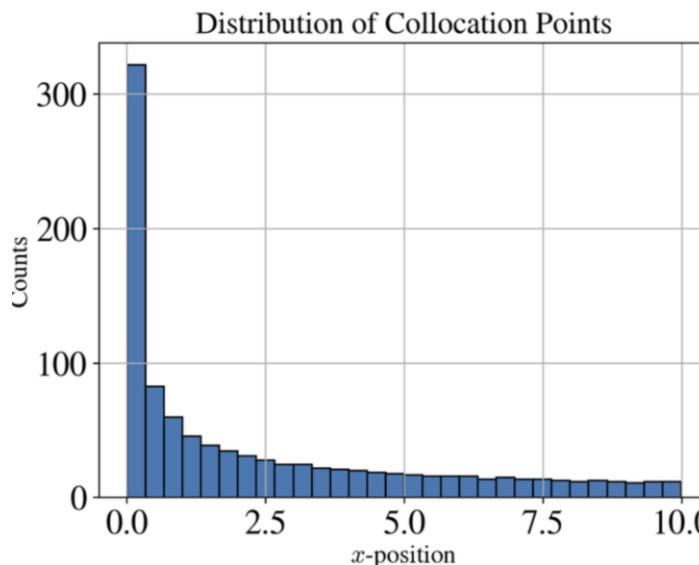
$$\begin{cases} \frac{d}{dx} \left(4\mu h \frac{du}{dx} \right) = \rho_i g \left(1 - \frac{\rho_w}{\rho_i} \right) h \frac{dh}{dx} \\ \frac{d(uh)}{dx} = a \\ \mu = \frac{B}{2} \left| \frac{du}{dx} \right|^{\frac{1}{n}-1}, \quad n = 3 \end{cases}$$

Analytical soln:

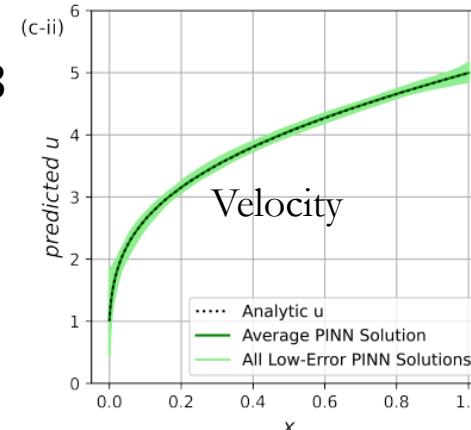
$$\begin{cases} u(x) = \frac{h_0 + x}{h(x)} \\ h(x) = \left(1 + \frac{h_0^{n+1} (h_0^{-n-1} - 1)}{(h_0 + x)^{n+1}} \right)^{-\frac{1}{n+1}} \end{cases}$$

Data: $d = M(\theta) + \eta$,

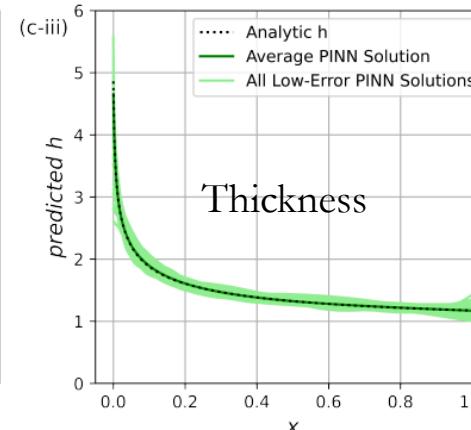
where $\eta \sim N(0, \Sigma_{noise})$, $\sqrt{\Sigma_{noise}} = 0.3$



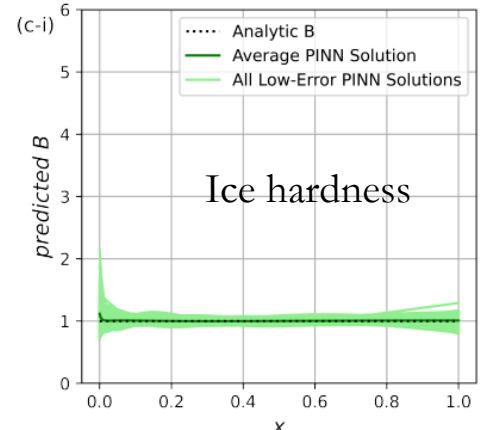
Training data



Training data



No training data



$$\text{Loss} = (1-\gamma)\text{Loss}_{\text{DATA}} + \gamma\text{Loss}_{\text{EQN}}$$

$$\gamma \in [0,1]$$

Re-sampling collocation pts!

1D SSA: https://github.com/YaoGroup/pinn_clusters

Data:

$$d = M(\theta) + \eta,$$

where $\eta \sim N(0, \Sigma_{noise})$,

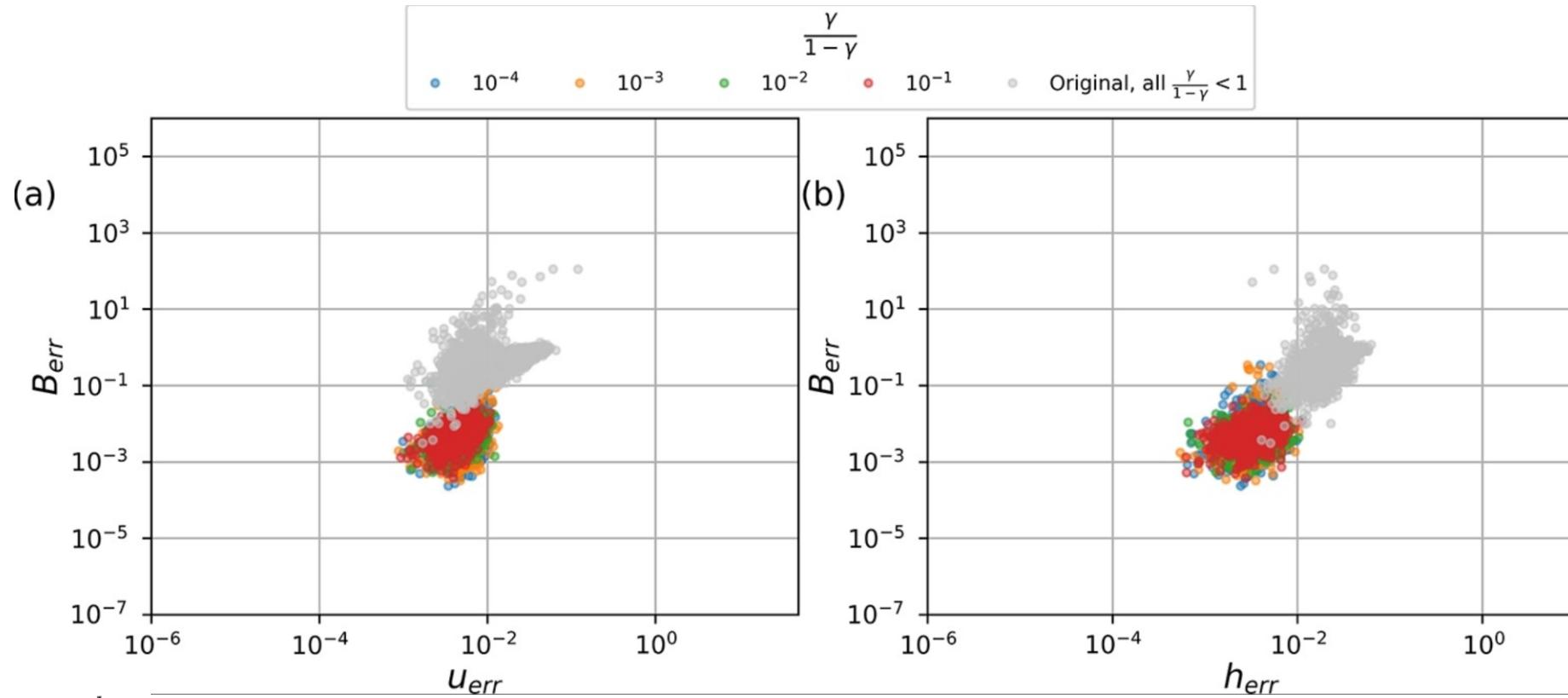
$$\sqrt{\Sigma_{noise}} = 0.3$$

Distance between prediction and ground truth:

$$B_{err} = \frac{1}{k} \sum_{i=1}^k \|\hat{B}(x_i) - B(x_i)\|^2, \quad \{x_i\}_{i=1}^k$$

$$u_{err} = \frac{1}{k} \sum_{i=1}^k \|\hat{u}(x_i) - u(x_i)\|^2, \quad \{x_i\}_{i=1}^k$$

$$h_{err} = \frac{1}{k} \sum_{i=1}^k \|\hat{h}(x_i) - h(x_i)\|^2, \quad \{x_i\}_{i=1}^k$$



Iwasaki and Lai, JCP 2023

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

Algorithm features

Synthetic case
Real data case

Critical features of `DIFFICE_jax` that go beyond off-the-shelf PINNs, and the necessity of these features to ensure the success and robustness of viscosity inference is explained below:

(1) Data and equation normalization/non-dimensionalization: Proper training of NNs requires both input and output of the NN to be normalized, typically within the range of $[-1, 1]$. However, the values of observational data of ice velocity and thickness differ by several order of magnitude in their original units. Therefore, both their values (output) and spatial positions (input) need to be normalized before training. After normalizing the data, the new governing equations and associated boundary conditions, expressed in terms of the normalized quantities need to be re-derived. Each term in the new equation should have a magnitude of $O(1)$. The `DIFFICE_jax` package provides the algorithm that can automatically normalize (non-dimensionalize) the observational data and generate the associated normalized SSA equations for different ice shelves.

(2) Optimal setting of equation weight: The cost function of PINNs involves two terms: the data loss \mathcal{L}_d and the equation loss \mathcal{L}_e . For viscosity inference, the data loss \mathcal{L}_d quantifies the mismatch between the observed data and corresponding NN prediction NN_d via mean squared error, while the equation loss (or boundary condition loss) is defined as the mean squared error between the right and left-hand side of the equations (or boundary conditions). With the normalized data and equations, the data loss and equation loss are also normalized. The weighting pre-factors of the data and equation loss (γ_b and γ_e defined in the section below) are set to 1 and 0.1, respectively, optimized to minimize the training error and are verified to be universal for studying different ice shelves.

(3) Design of NNs to enforce positive-definiteness: The effective viscosity μ must be positive everywhere. In addition, evidence shows that the spatial variation of μ within the ice shelf could cover several order of magnitude. We introduce the viscosity expression as

$$\mu = \exp(NN_\mu),$$

where NN_μ is the output of the fully-connected NN created for μ . This setting ensures the positive-definiteness of the inferred viscosity and enhance the training to capture both the local and global profile of viscosity with high accuracy over large spatial domain.

(4) Residual-based re-sampling of collocation points during training: PINN training with observational data can often result in the cheating effects [@wang2022discovering, @charlie2024euler] due to errors and noise in the data. Here, cheating refers to the situation where the NN overfits the training data, leading to a small training loss but a large validation error elsewhere. A basic way to mitigate this issue is to randomly re-sample both data points and collocation points at regular intervals during training [@lu2021deepxde,@daw2022mitigating]. This can reduce the likelihood of overfitting. Collocation points refer to the points used to compute the equation residue. Additionally, a more effective approach to prevent overfitting and enhance training efficiency is to re-sample data and collocation points with higher concentration in areas where the spatial profile of the NN error or equation residue is larger. This residual-based resampling scheme is embedded in the `DIFFICE_jax` package as a default training setting.

(5) Extended-PINNs (XPINNs) for large ice shelfe: Large ice shelves, such as Ross, poses a multiscale challenge for capturing both local-scale and large-scale spatial variation of u, v, h, μ . These local variations are difficult to capture with a single NN due to the spectral biases of NNs [@rahaman2019spectral,@xu2020frequency]. To address this challenge and ensure that PINNs capture those spatial variation precisely, the `DIFFICE_jax` package adopts the approach of extended PINNs (XPINNs) [@jagtap2020extended] for studying large ice shelves. This method divides the training domains into several sub-regions, with different NNs assigned to each. In this approach, each NN is trained to learn a specific sub-region of the large ice shelf, allowing it to capture local variations with high precision. We note that XPINNs require extra pre-processing of the observational data and additional constraints or penalty terms in the cost function to ensure successful training. Detailed requirements are documented in the `XPINNs` subfolder in the GitHub repository.