Physics-informed neural networks (PINN)

by C. Yao Lai for 2025 Glamacles summer school

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:

Basal friction (spatially varying)

Ice viscosity (spatially varying)

Need to solve inverse problem to infer them:

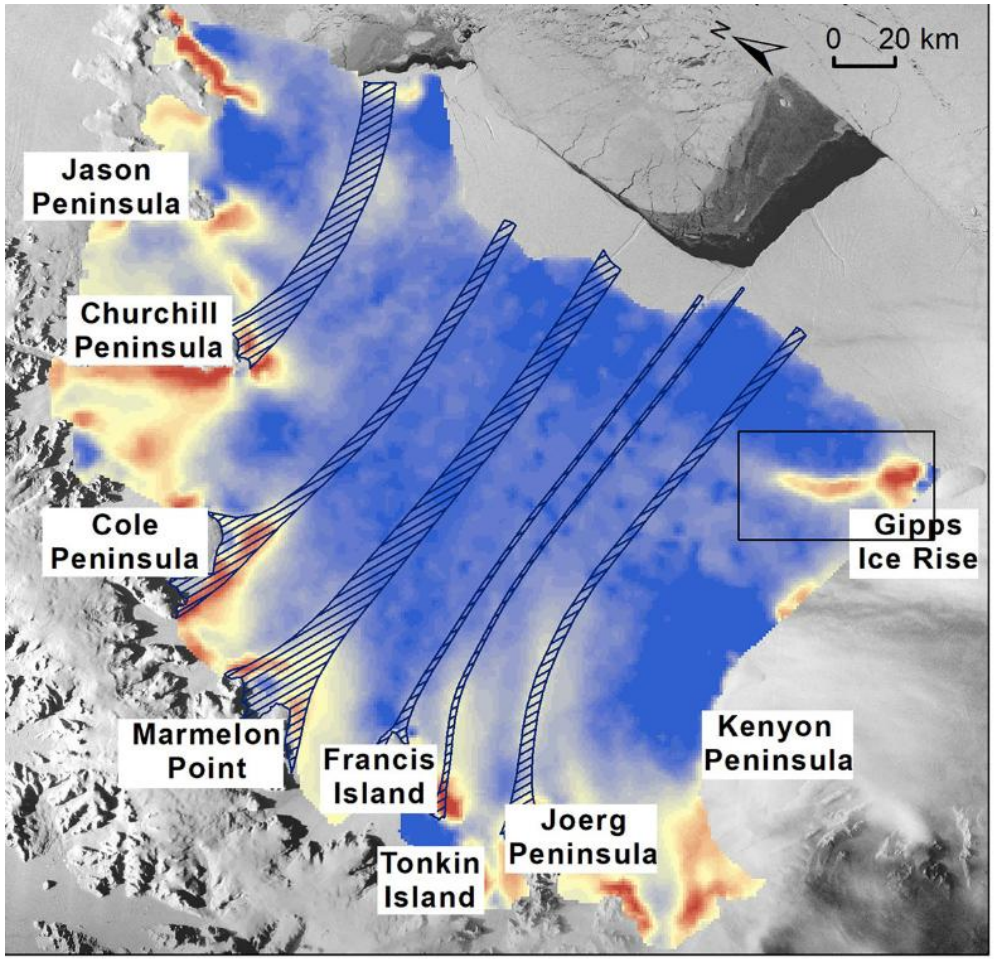
Adjoint method, Bayesian inference, PINN, CNN Gaussian Process…etc

modeled basal friction [kPa yr/m]

1.0e-01 1 10 100 1.0e+03

Mauro Perego

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

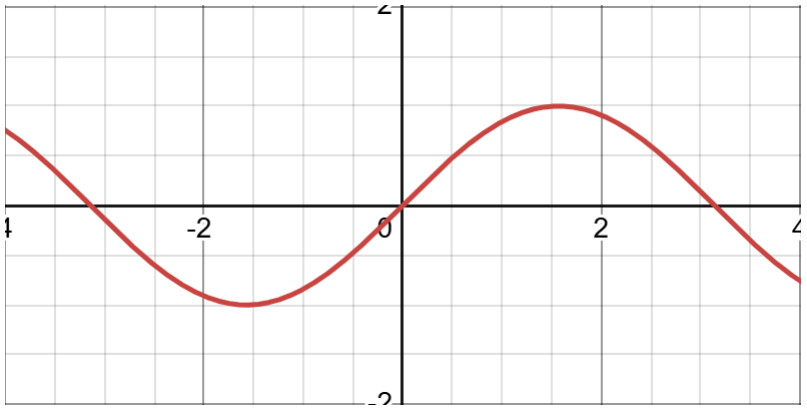


Shujie Wang

2.5 5 7.5 10 12.5 15 17.5 20

How to represent functions/data/solution?

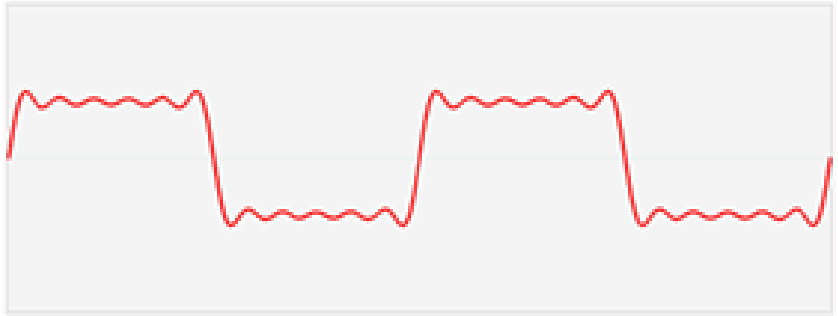
$$u(x) = \sin(x)$$



1. **Put it on a numerical grid**
   Sample N points in $x = [-2\pi, 2\pi]$
   # of parameters: N

# 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: ?

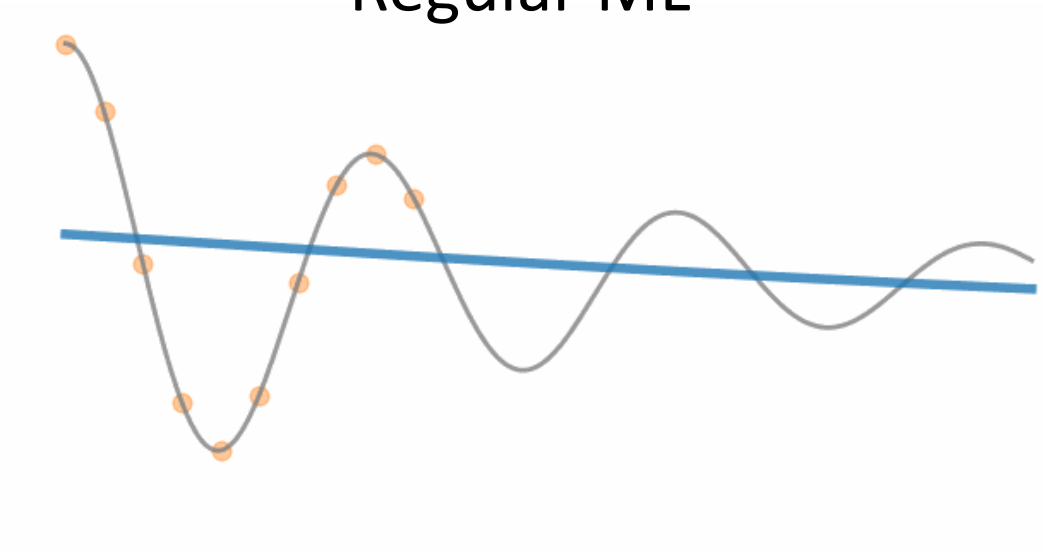
3. **Other basis functions**
4. **NN:** Universal function approximator. Reduce a priori assumption about basis fn. Let the data decide!

Outline

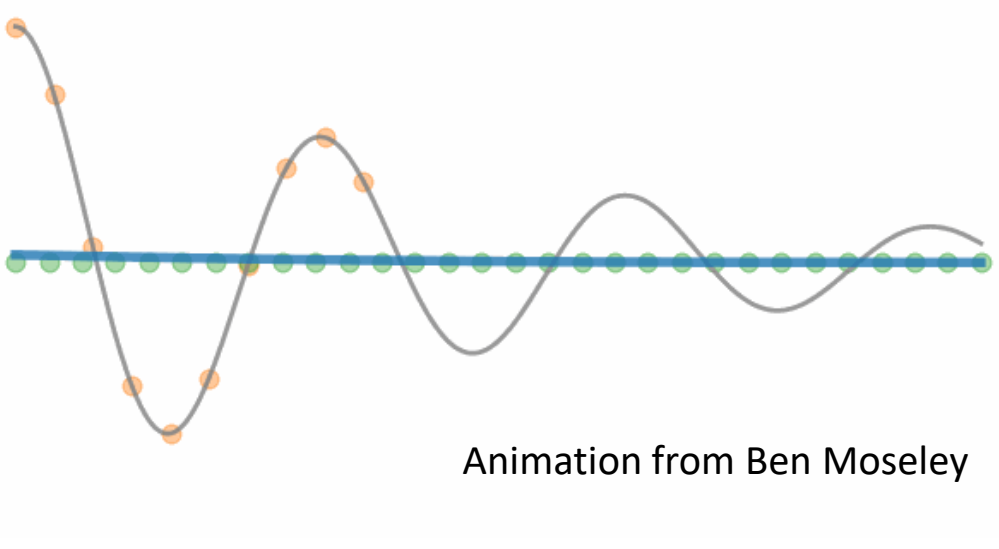
- Part I: Physics-informed neural networks
- Part II: Application to inverse modeling on ice shelves

Better extrapolation beyond training data

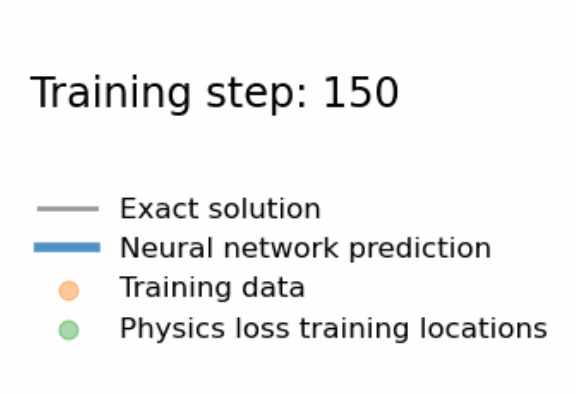
Regular ML



Physics-informed ML



Animation from Ben Moseley



Data loss: fit the data

Eqn loss: fit the physics equations

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

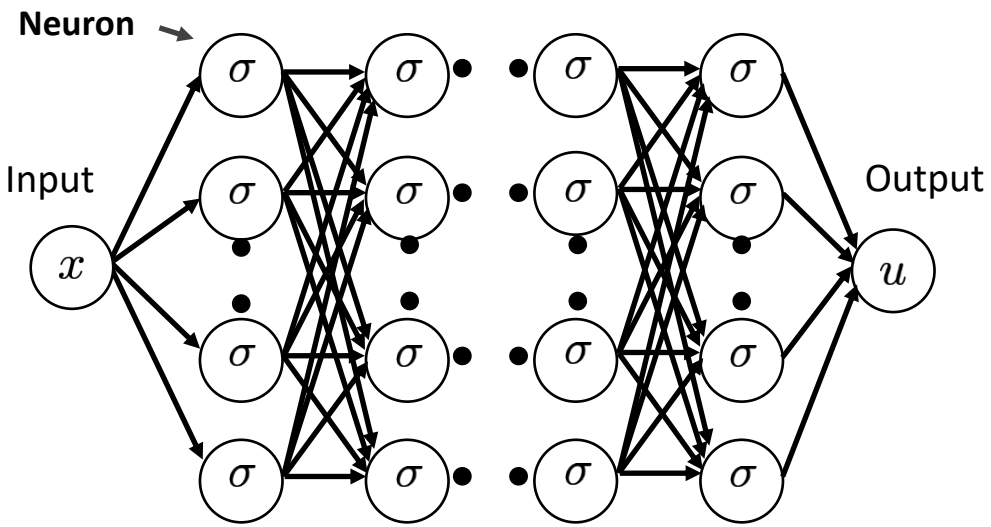
The new loss function:

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

Neural network

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

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

(free parameters to be trained)

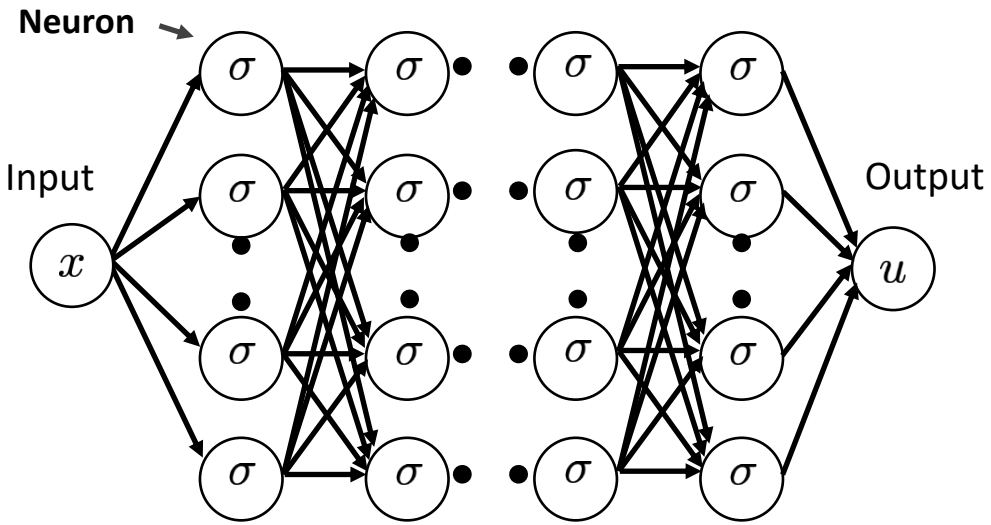
$\sigma(x)$: activation function

(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

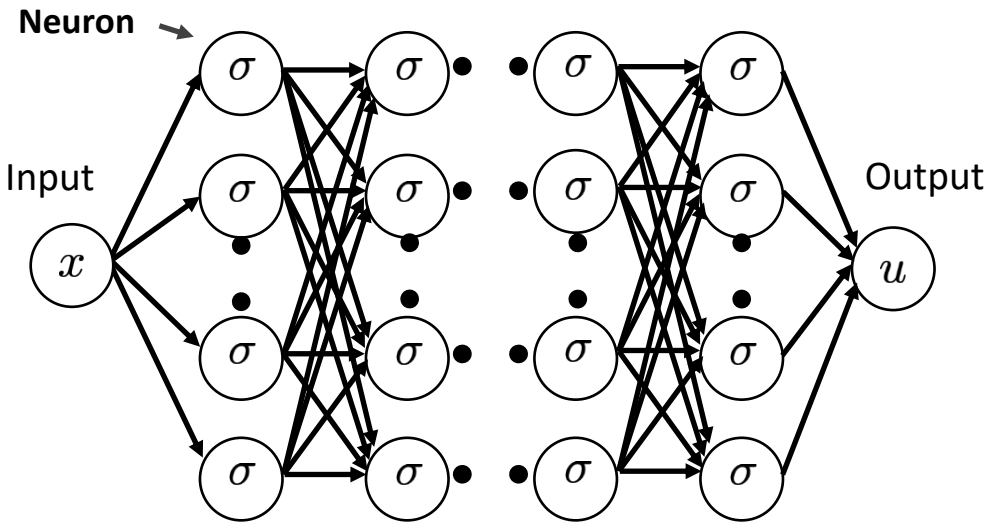
Universal function approximator

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

Neural network and Fourier series

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

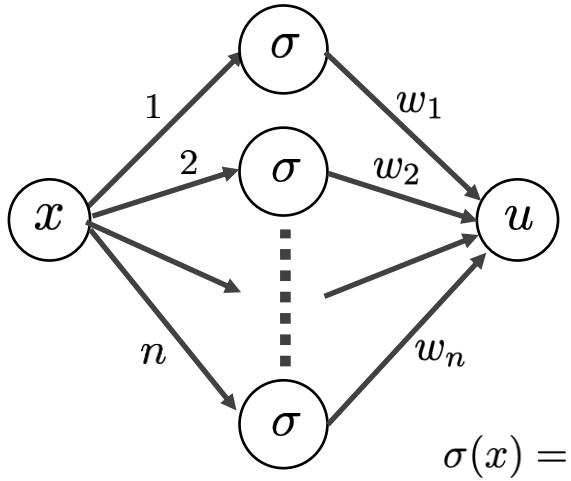
Fully-connected Neural network



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

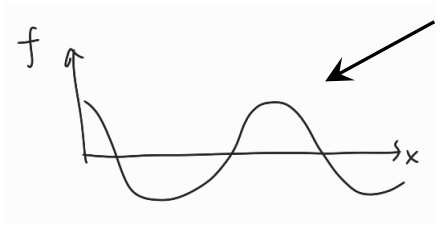
$\mathbf{w}$: weights $\quad$ $\mathbf{b}$: biases $\quad$ $\sigma(x)$: activation function

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

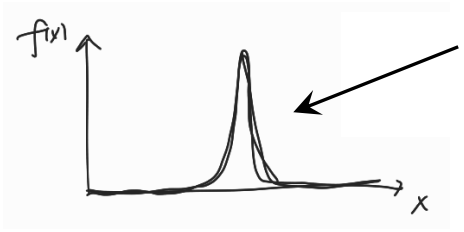


Universal function approximator

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



Use sin() activation



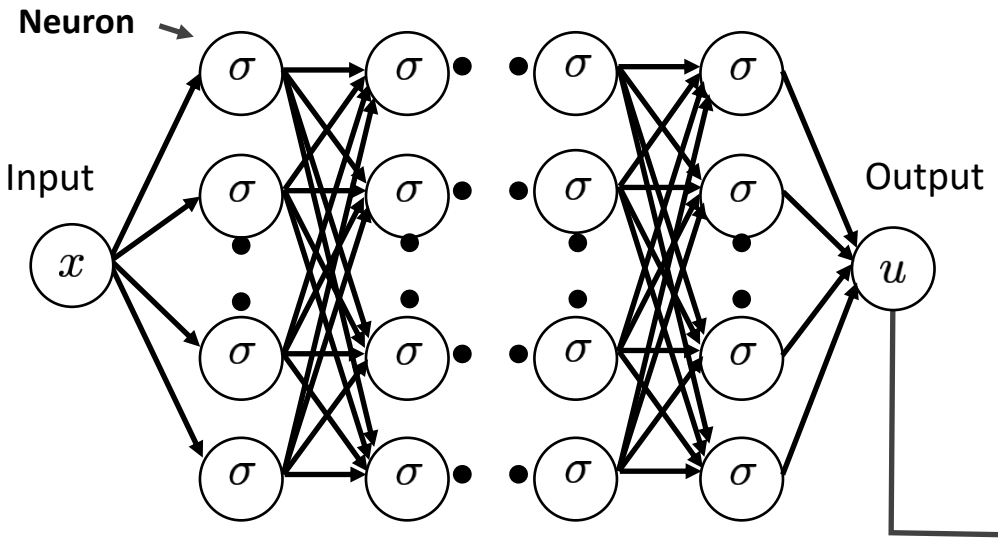
If using sin() activation, need lots of neurons (frequencies)

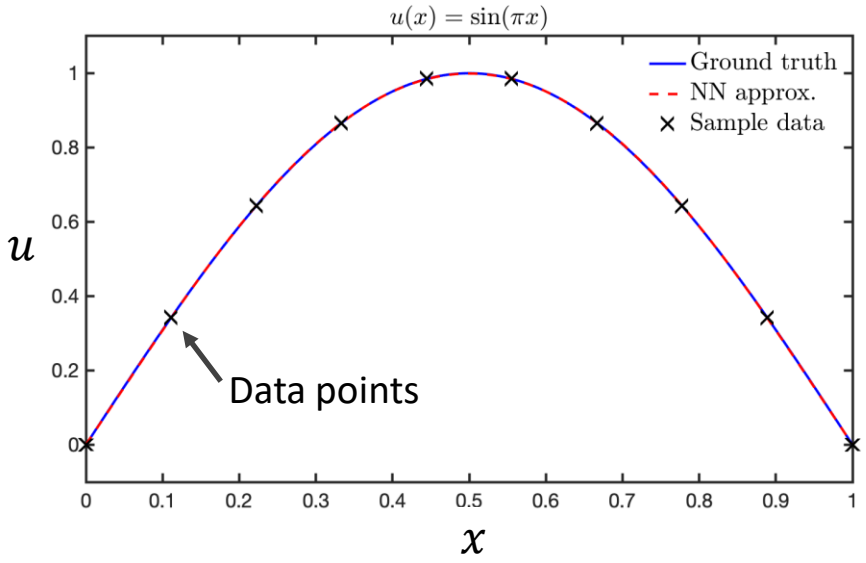
Better to use tanh() activation

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

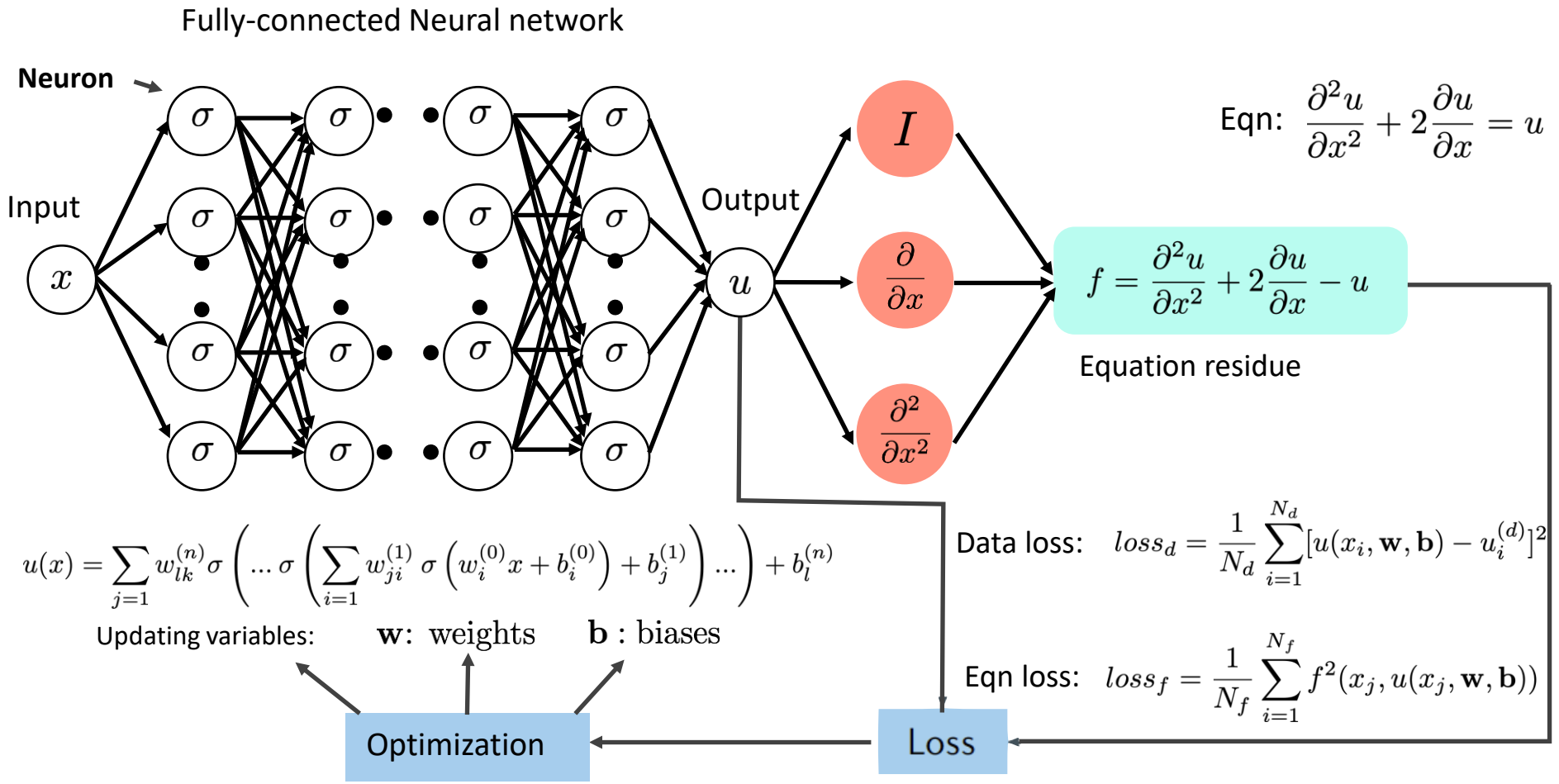
Cost function: *mean squared error*

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)

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations 16005 * 2019

M Raissi, P Perdikaris, GE Karniadakis

Journal of Computational physics 378, 686-707

- A 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*

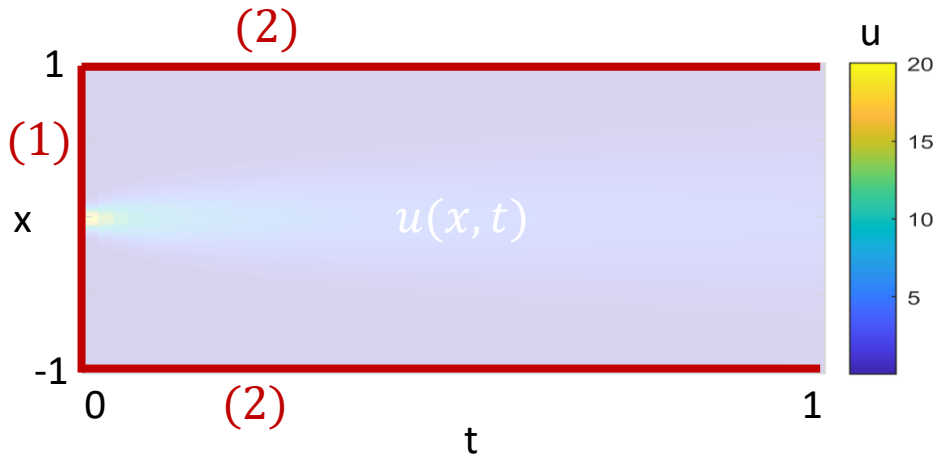
Learn u with physics equation + ICs + BCs

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

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

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

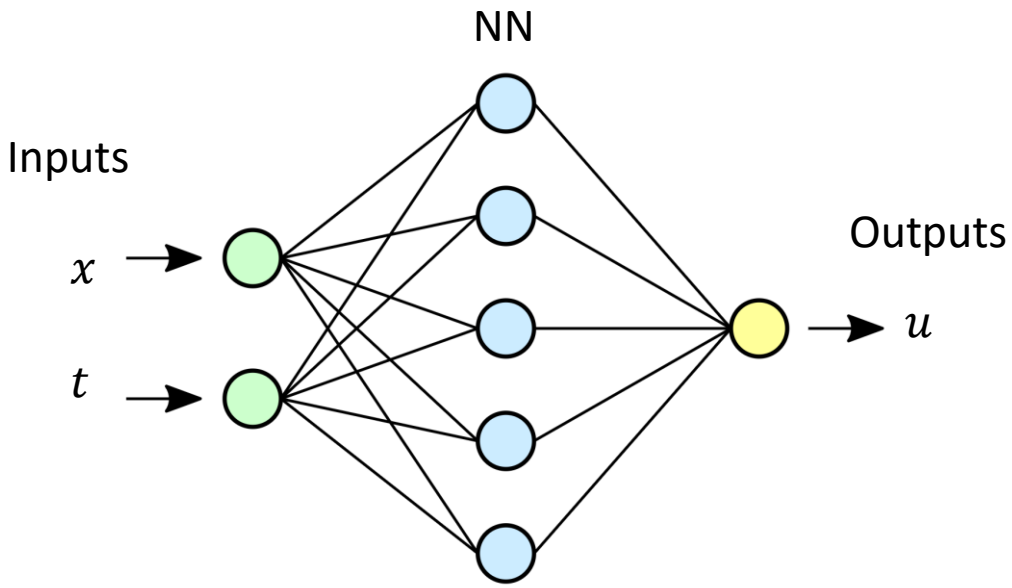
→ $GOAL$: $solve\ for\ u(x,t)$

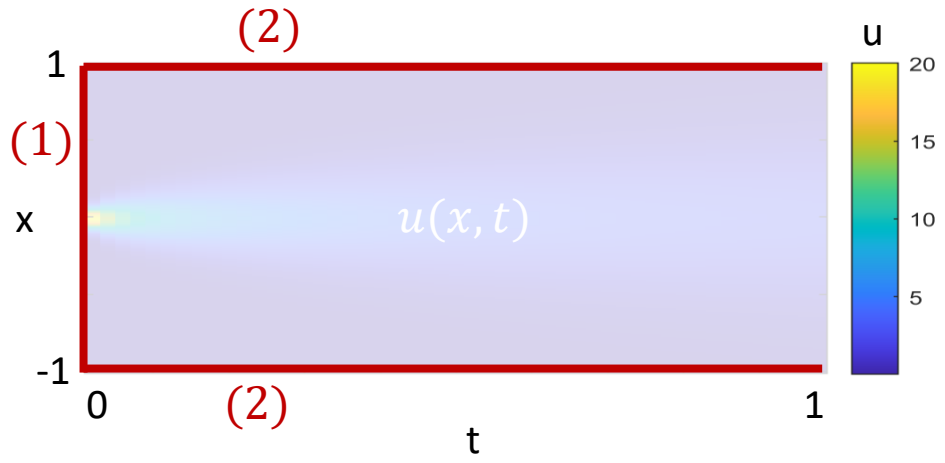


Mathematically, the information we need to get u(x,t) is **(1) ICs + (2) BCs + (3) physics equation.**

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

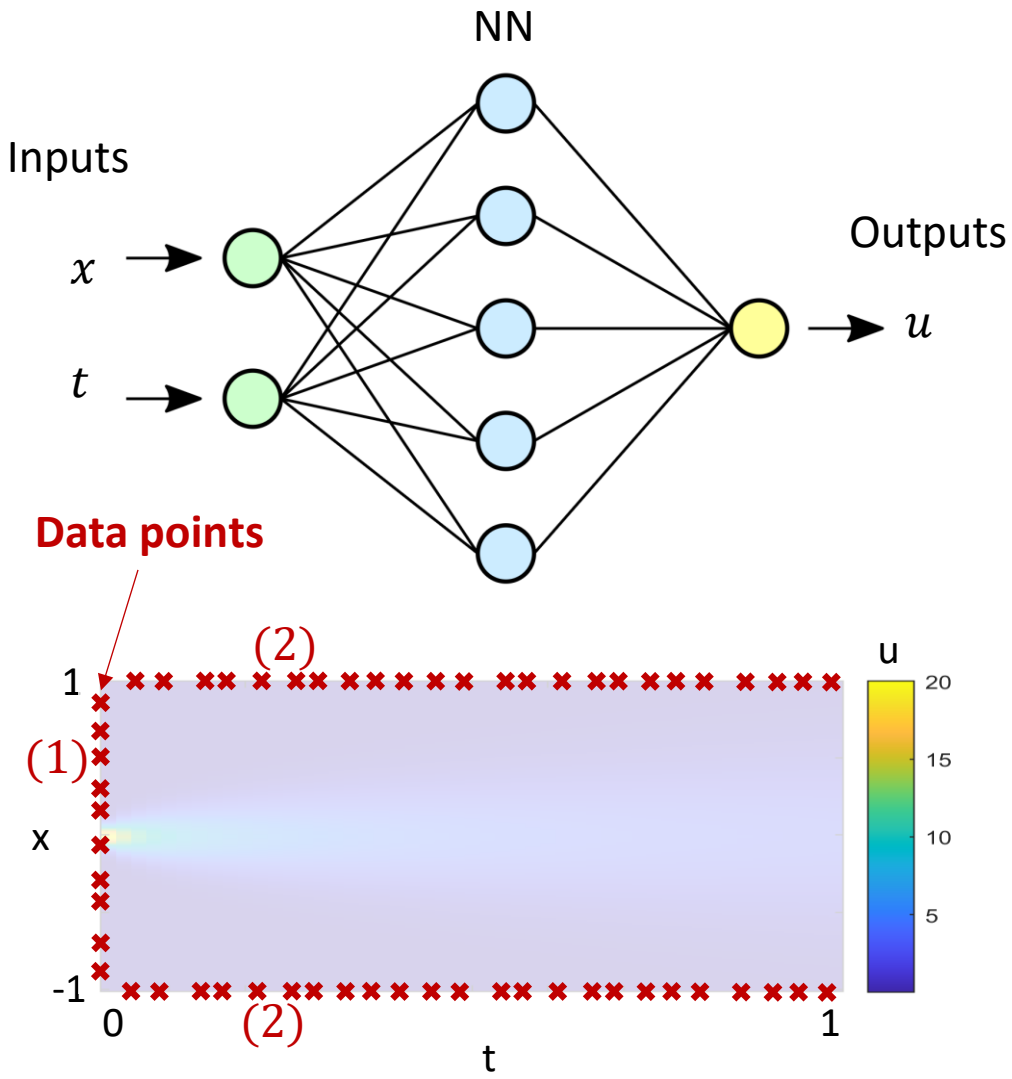
2. For a NN, all derivatives of u w.r.t x and t can be calculated analytically because u(x,t) is exact!

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*:

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

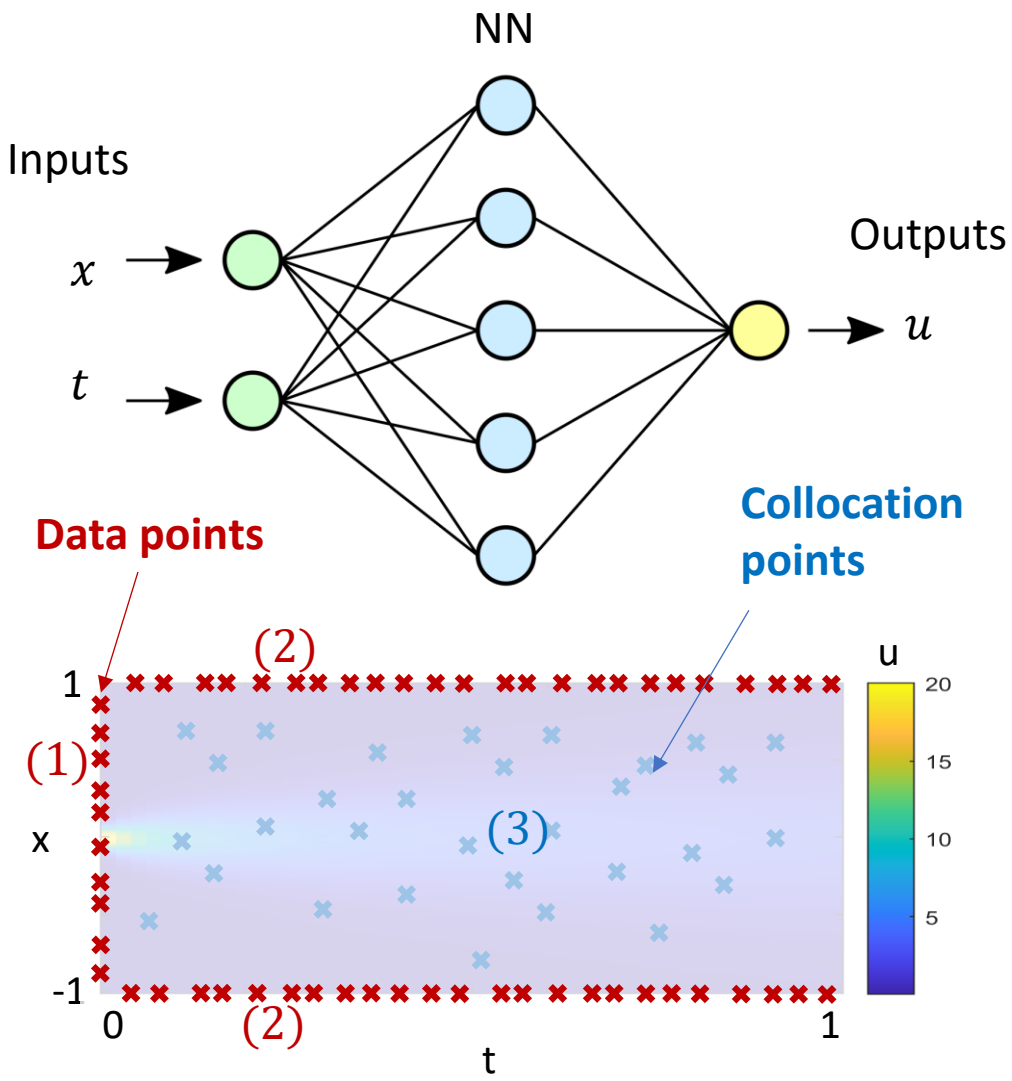
What would be the loss function?

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

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

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:

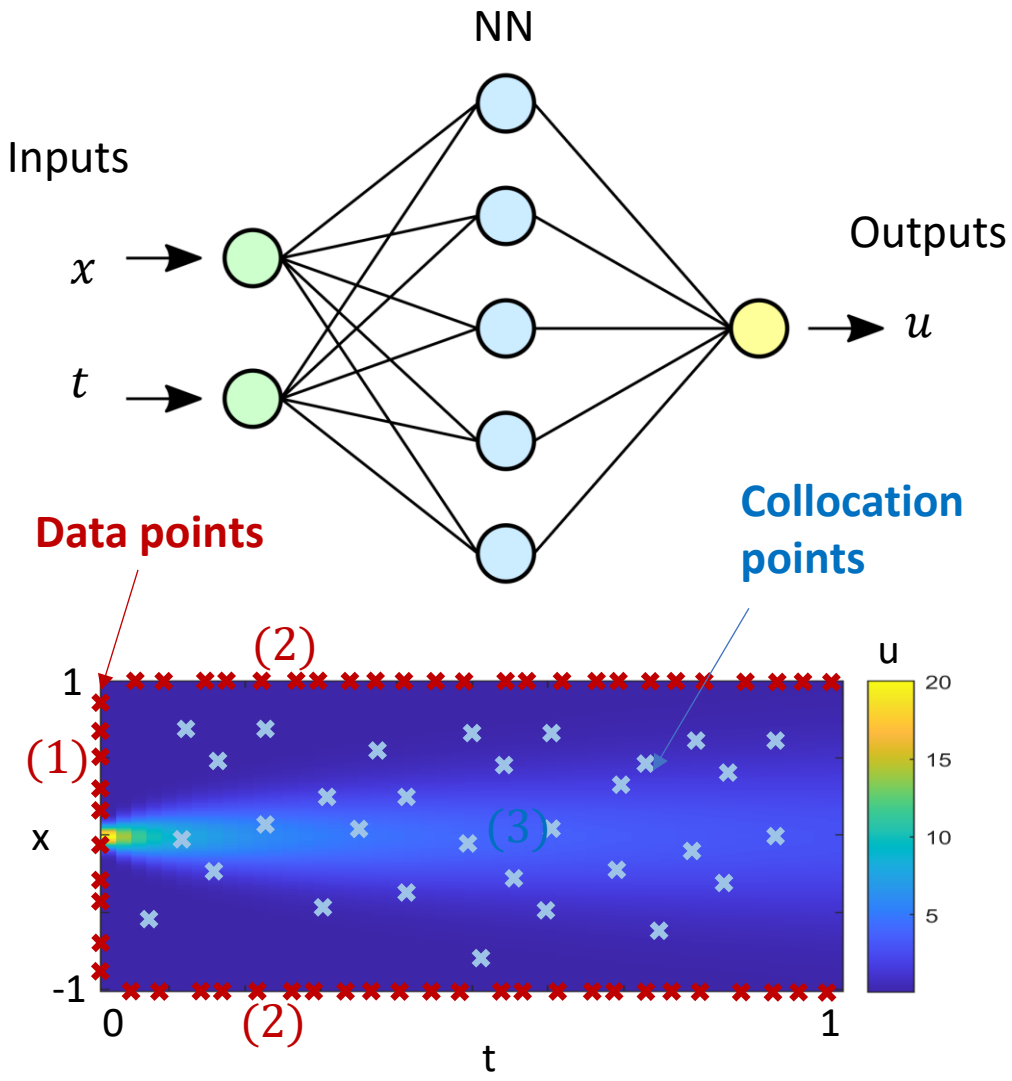
minimize $$\frac{1}{N_f}\sum_{k}^{N_f}\left|\frac{\partial u_{pred}^k}{\partial t} - a\frac{\partial^2 u_{pred}^k}{\partial x^2}\right|^2$$ (3) Eqn

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)

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

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

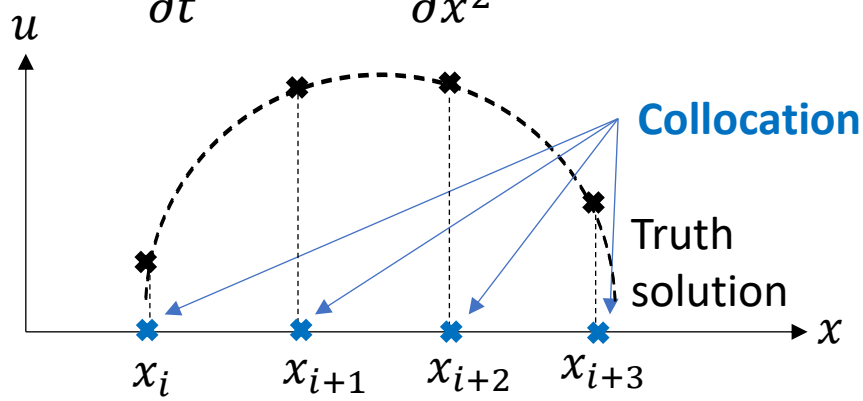
Data points

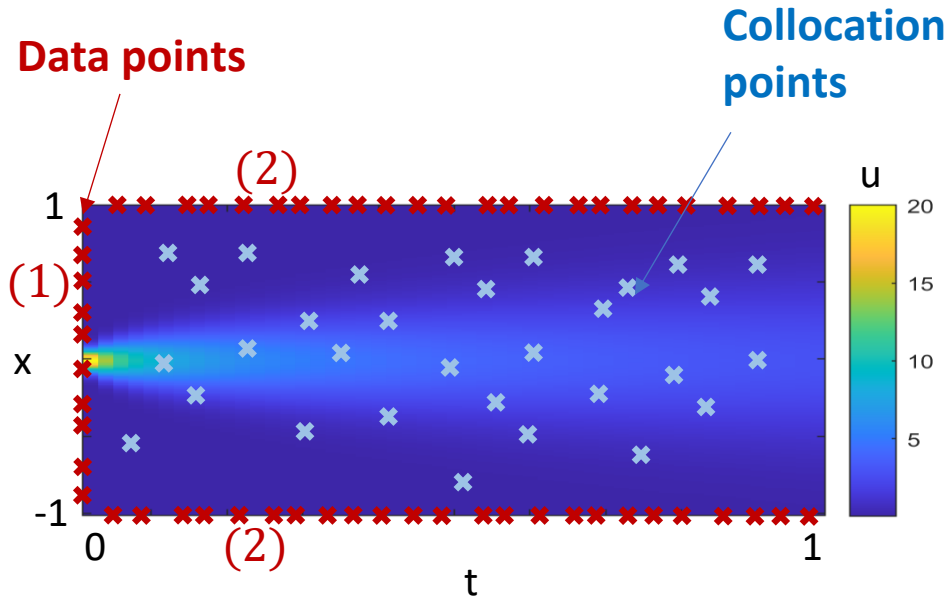
$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

PINN: searches for a curve that satisfies

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

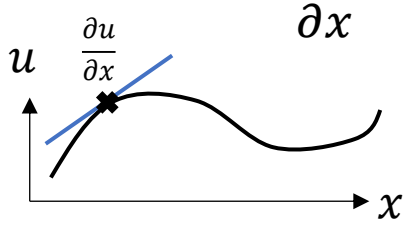




Automatic differentiation: differentiate NN output with respect to their input coordinates.

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** (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?

Raissi et al (2019)

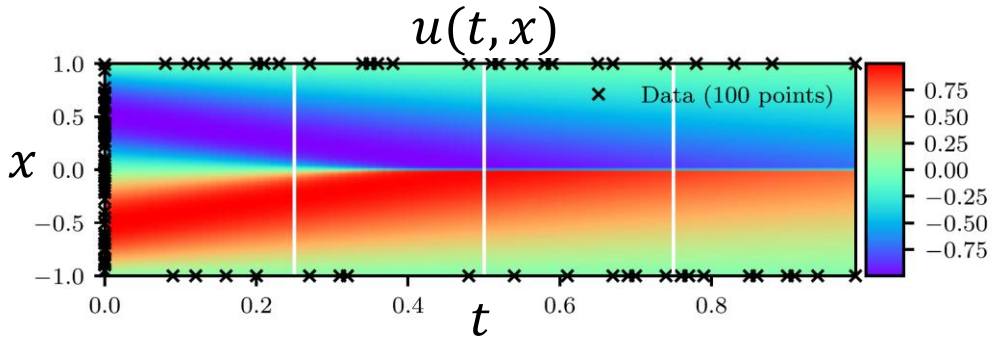
E.g., Burgers' equation (inference)

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

Raissi et al (2019)

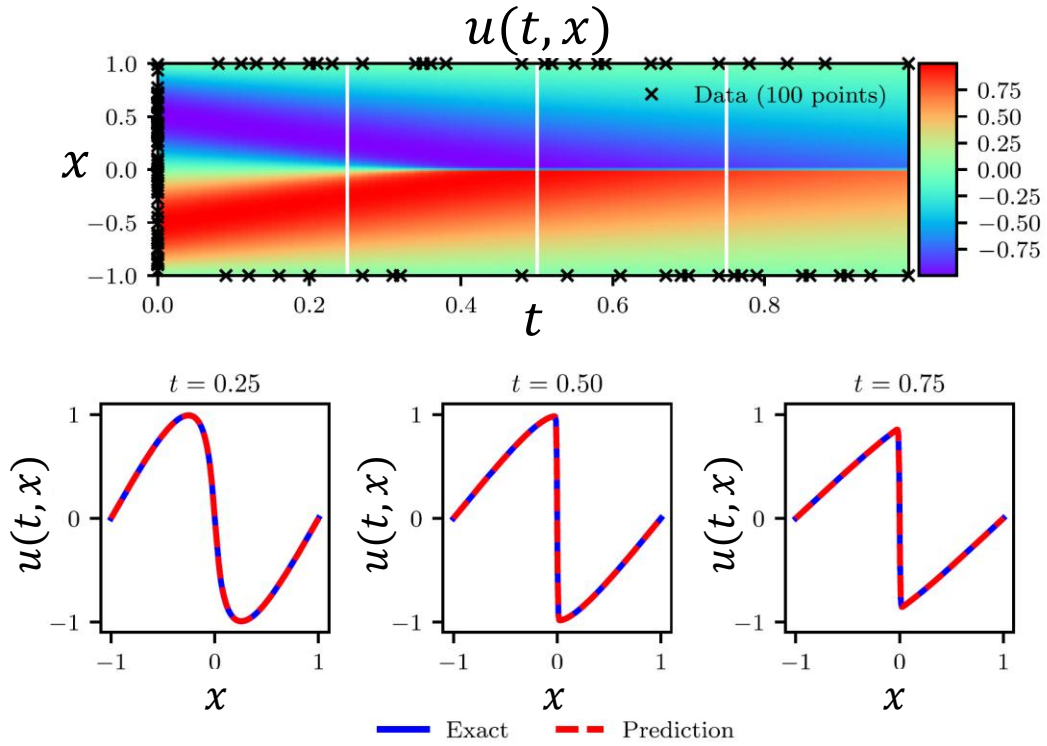
E.g., Burgers' equation (inference)

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

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

- 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

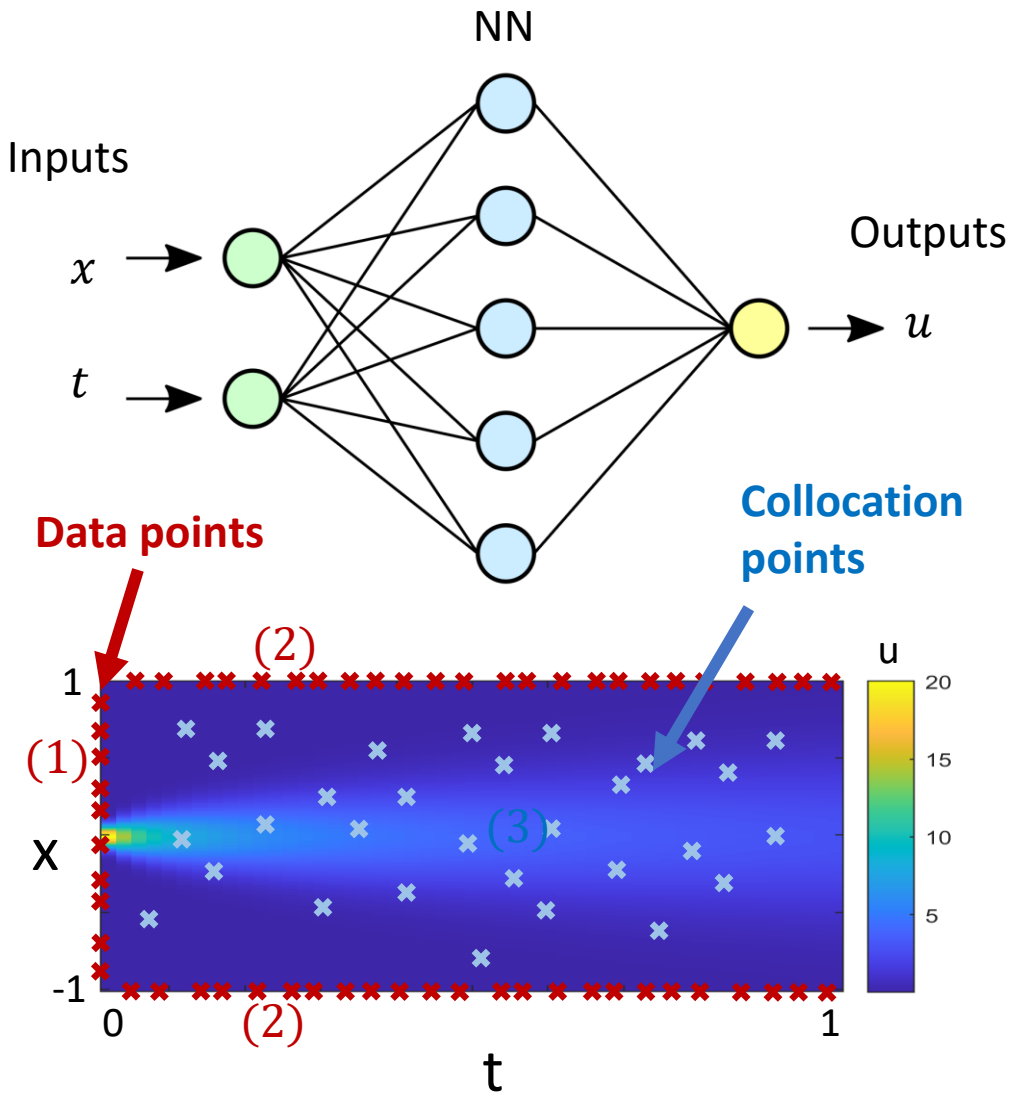
  Given an eqn and parameters λ, what's the model prediction best describes the data?

- Application 3: Data-driven discovery of unknown parameters

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

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)

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

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

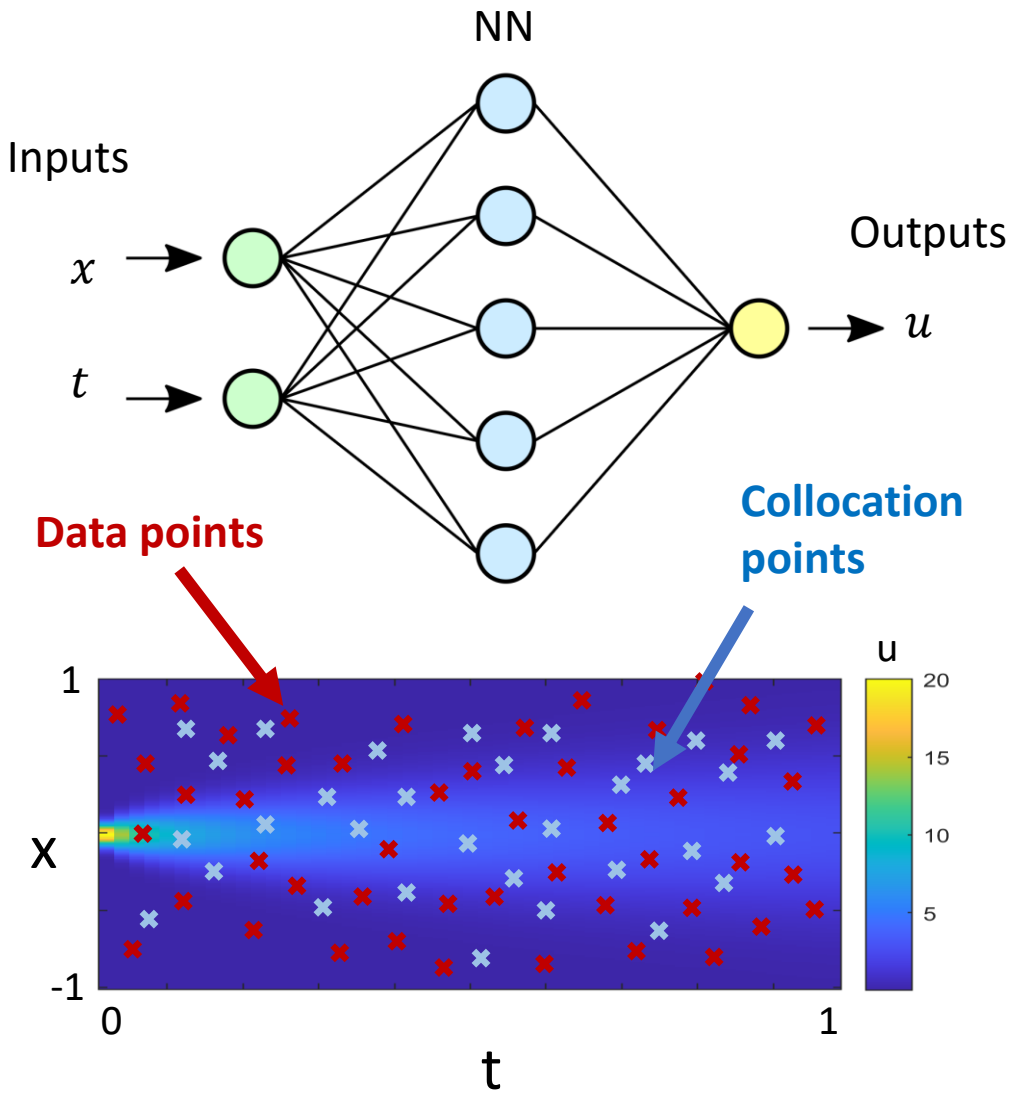
Data points

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

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)

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

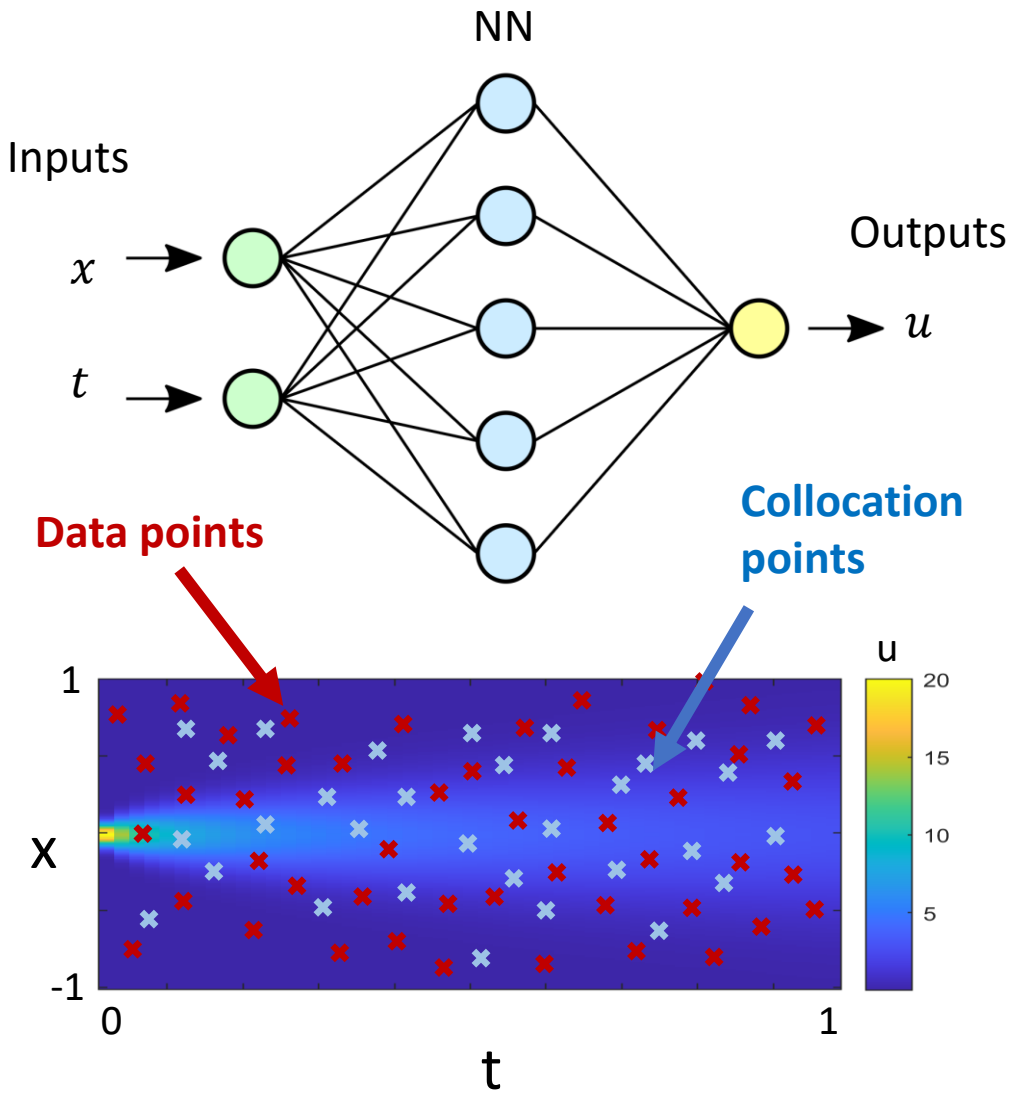
$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

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

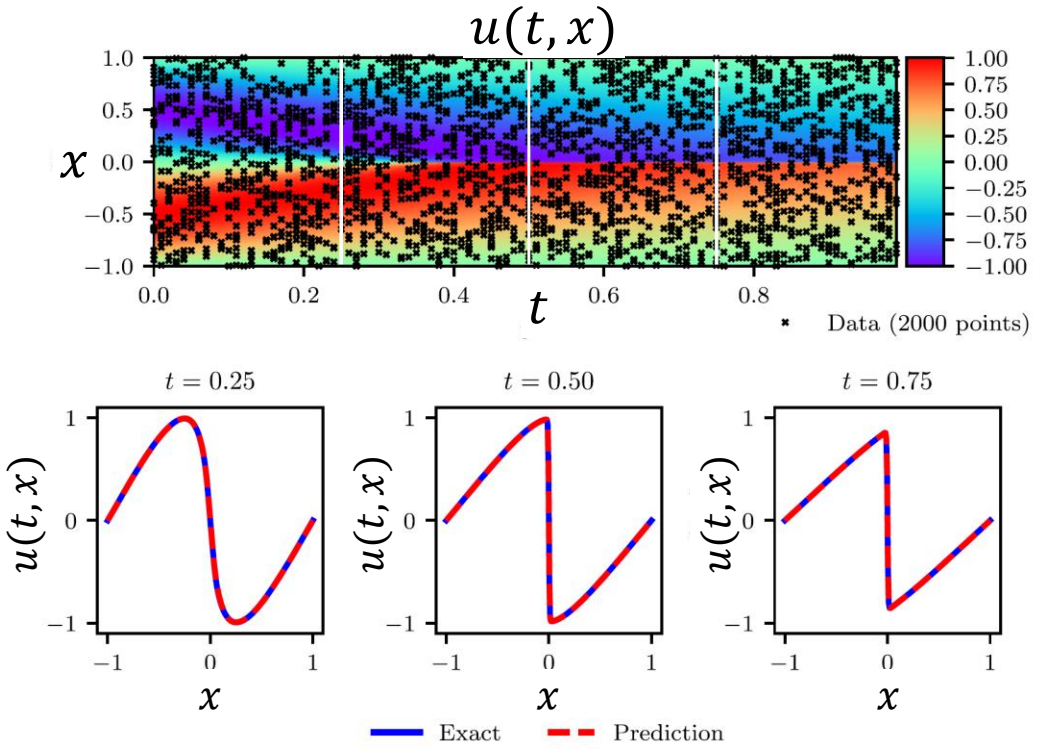
Raissi et al (2019)

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

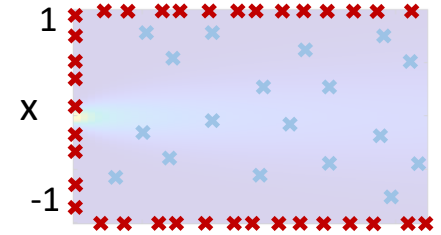
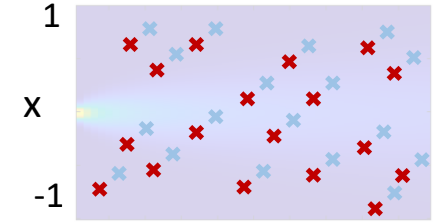
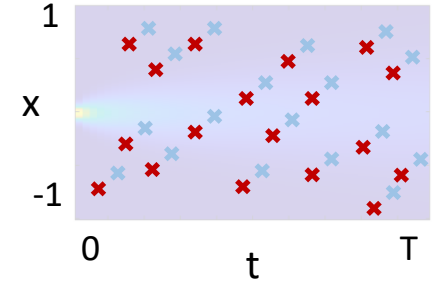
Equation loss

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

Collocation points

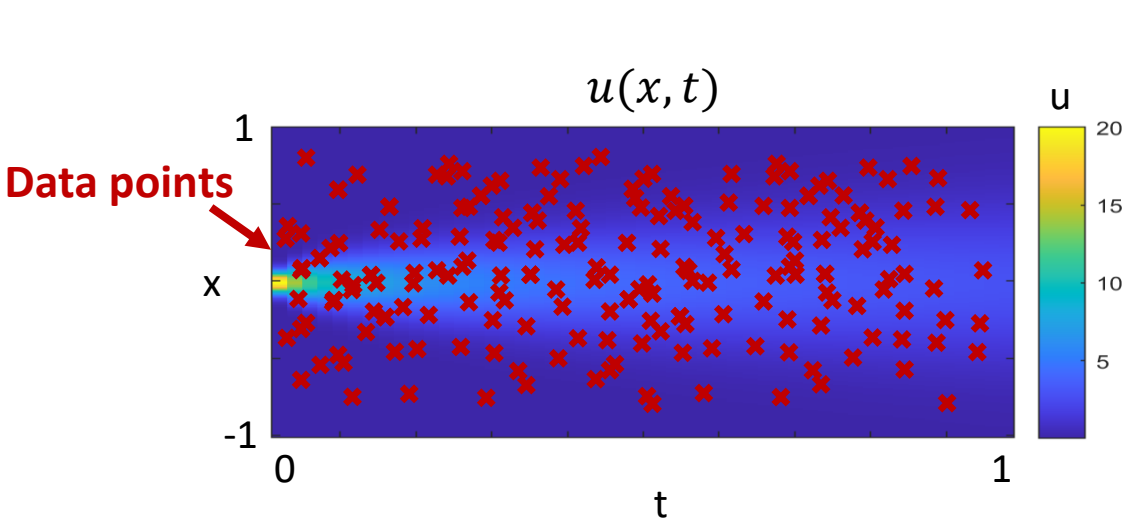
PINN

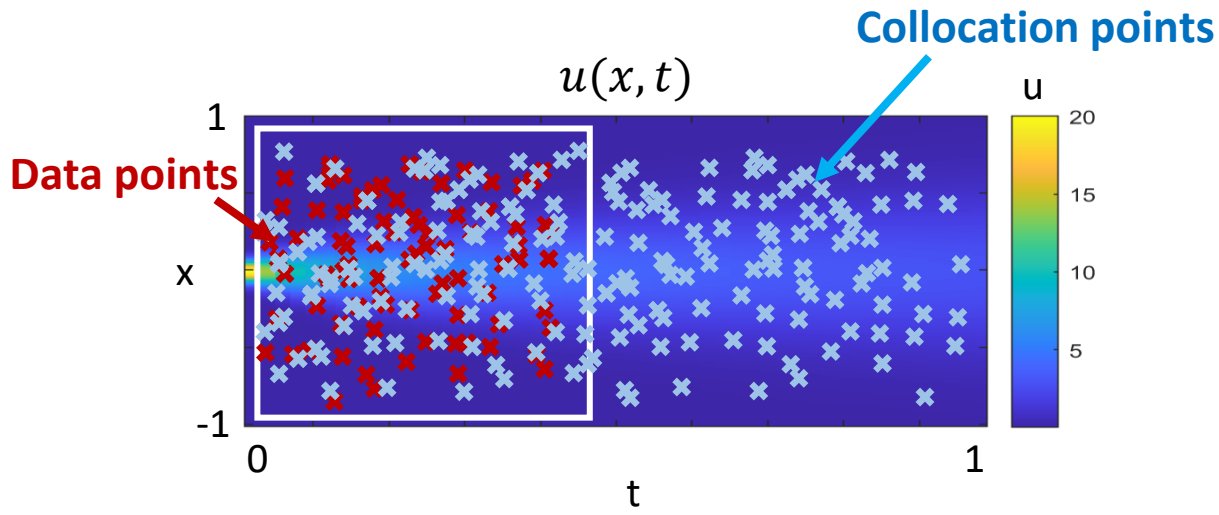
$\frac{\partial u}{\partial t} + N(u, \lambda) = 0, \quad x \in [-1, 1], \quad t \in [0\ T]$

Applications	Training data	NN prediction	
1. Prediction of solution of a **well-posed problem**	$IC, (BC)$	$u(x,t)$	
2. Prediction of solution when data is available within the domain but not at the IC, BC	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**	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 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

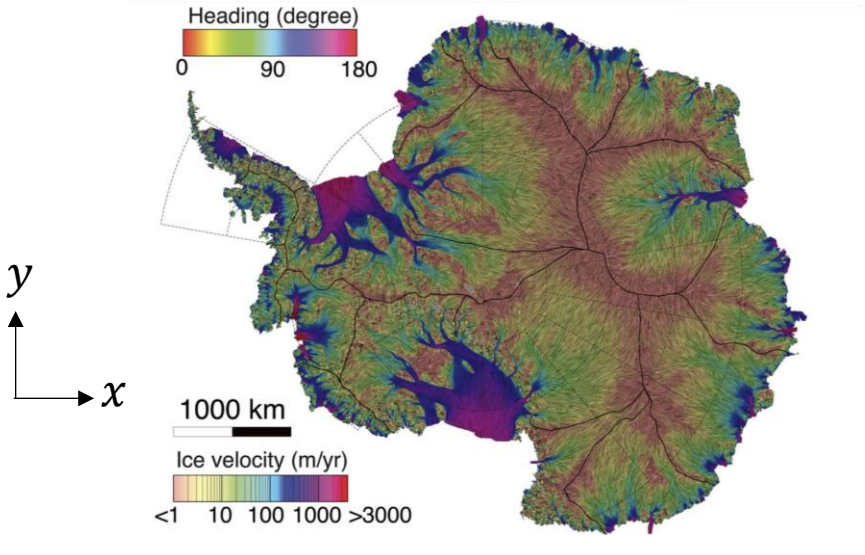
We will focus on ice shelves to eliminate one unknown (basal friction=0)

How to infer the ice effective viscosity from data?

Data:

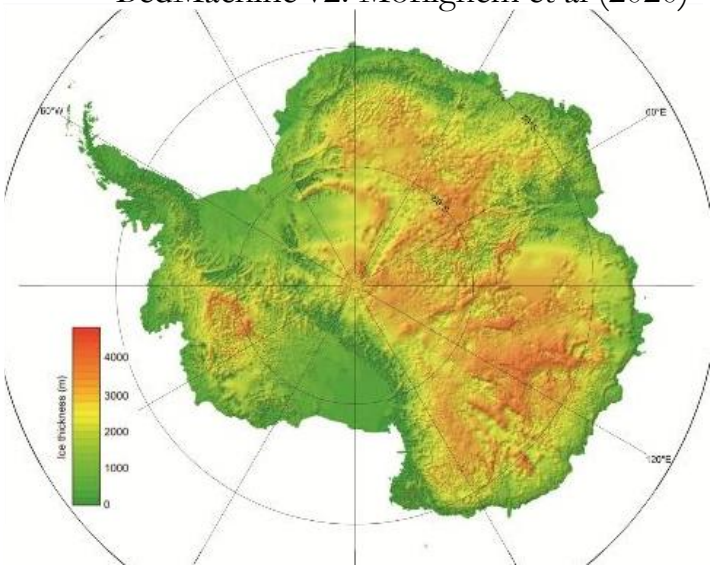
Ice velocity u

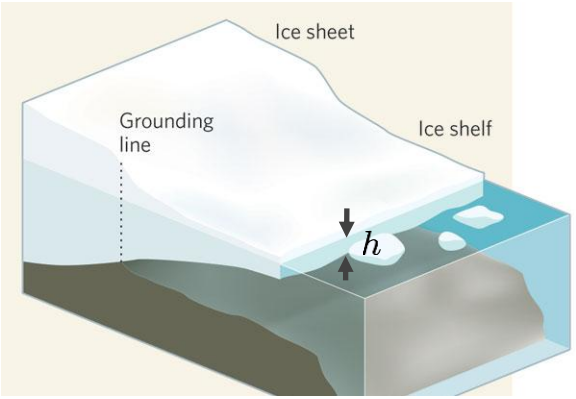
MEaSUREs v1: Mouginot et al (2019)



Ice thickness h

BedMachine v2: Morlighem et al (2020)





Equation:

Ice effective viscosity: impossible to measure

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

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

Morland (1987), MacAyeal (1989)

u, v: horizontal velocity

ρ_i: ice density

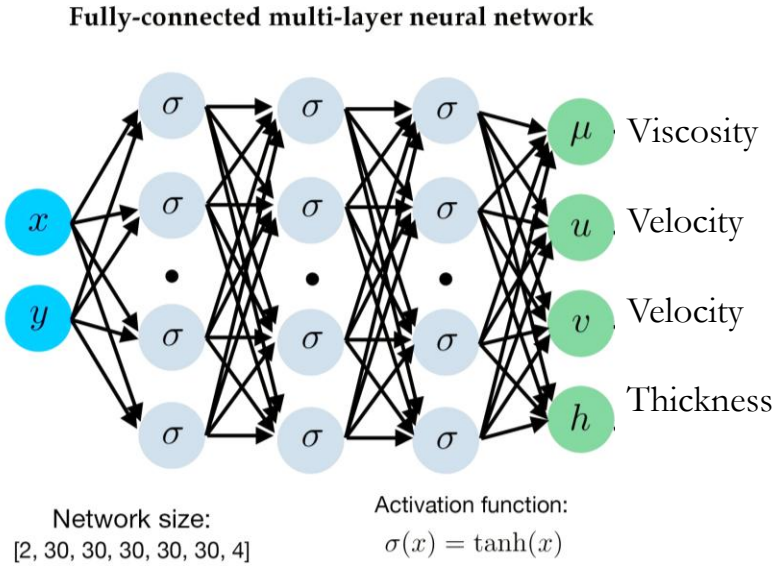
ρ_w: sea water density

h : ice thickness

} measurable

Ice viscosity inversion using physics-informed NN

Raissi *et. al.* (2019)

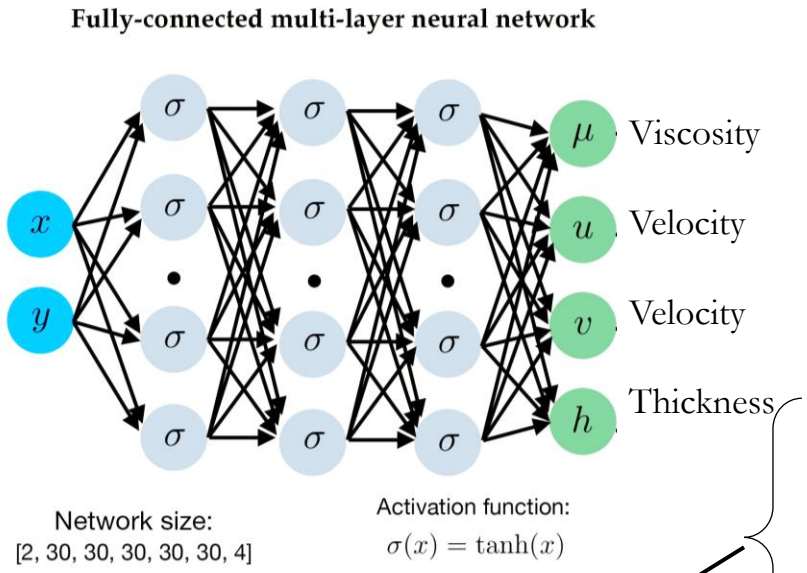


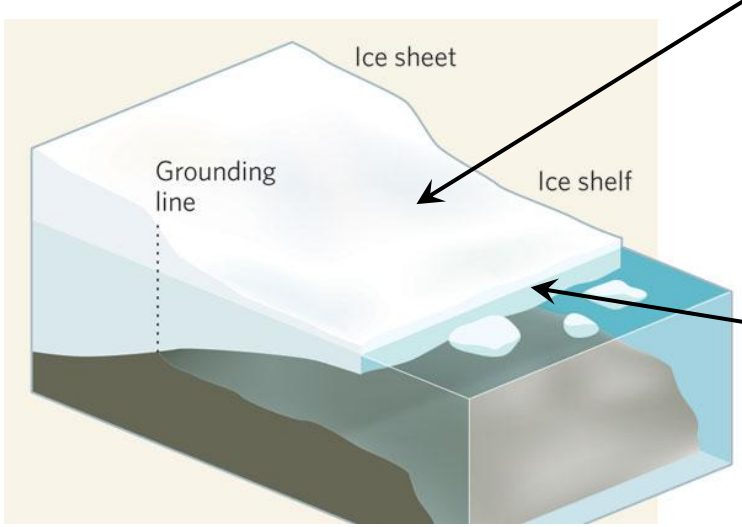
- Utilize the NN's representation power (a universal function approximator)

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

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

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

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

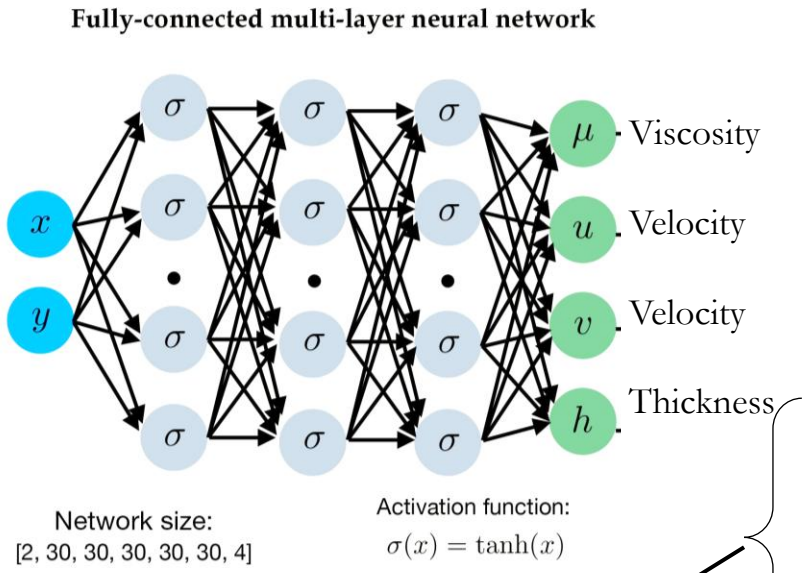
Stress balance at the **ice front**

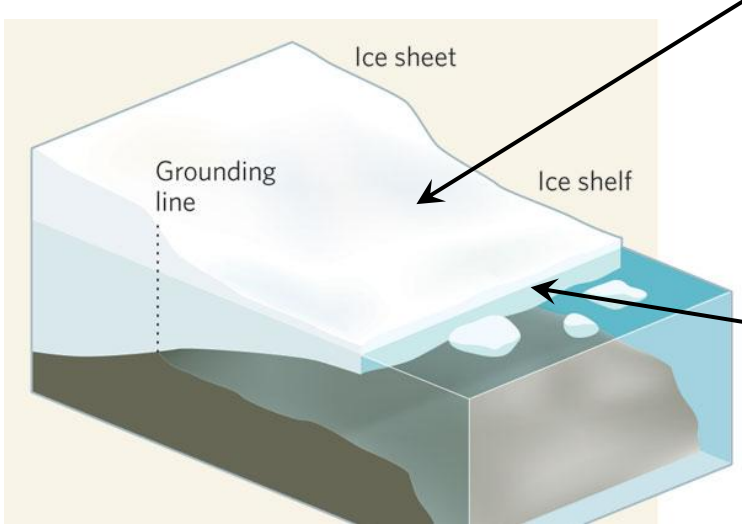
$$f_3 = 2\mu h\left(2\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)n_x + \mu h\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_y - \frac{1}{2}\rho g\left(1 - \frac{\rho_i}{\rho_w}\right)h^2 n_x$$

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

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

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

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

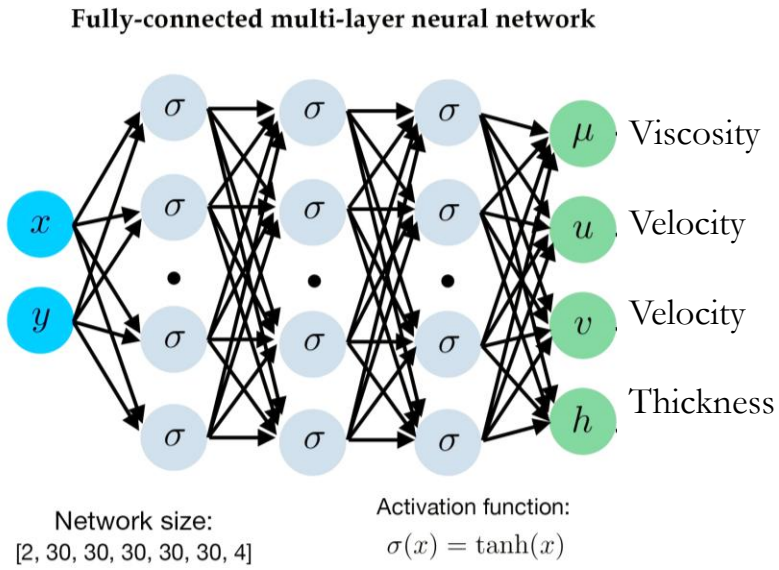
Stress balance at the **ice front**

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

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

Ice viscosity inversion using physics-informed NN

Raissi *et. al.* (2019)



- 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**

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

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

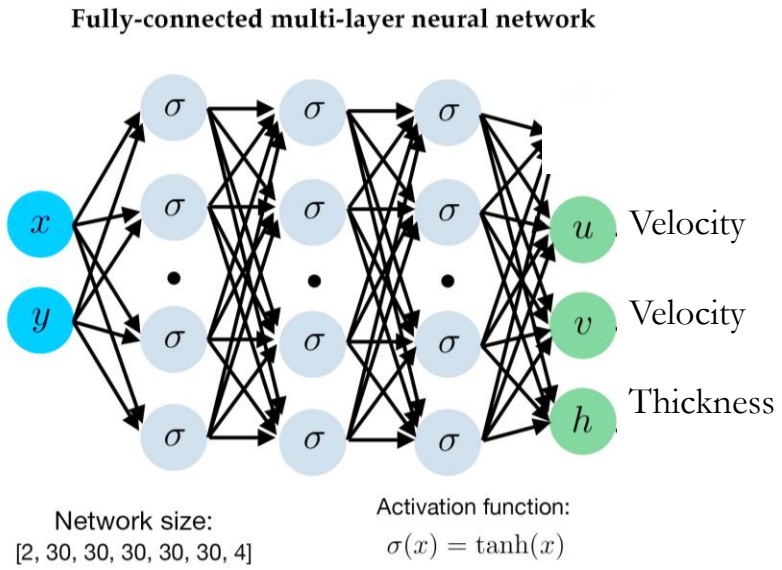
Minimize this cost function:

$$J = \int (u_{obs} - u)^2 d\Omega + \int (v_{obs} - v)^2 d\Omega + \int (h_{obs} - h)^2 d\Omega + \gamma \Sigma_i \int f_i \, d\Omega$$

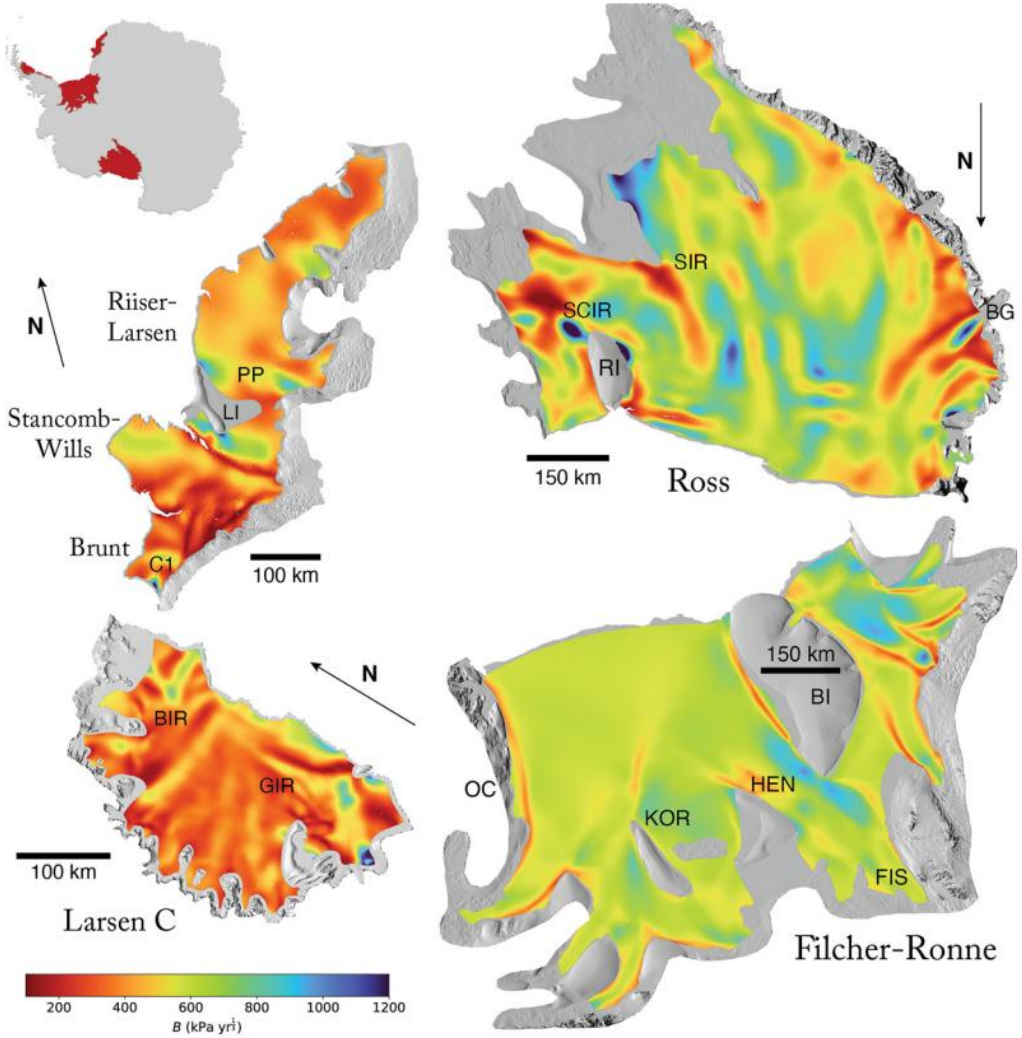
Data misfit

$\gamma \Sigma_i \int f_i \, d\Omega \rightarrow$ Equation misfit

Relevant work using physics-informed NN



$B(x, y)$: ice rigidity

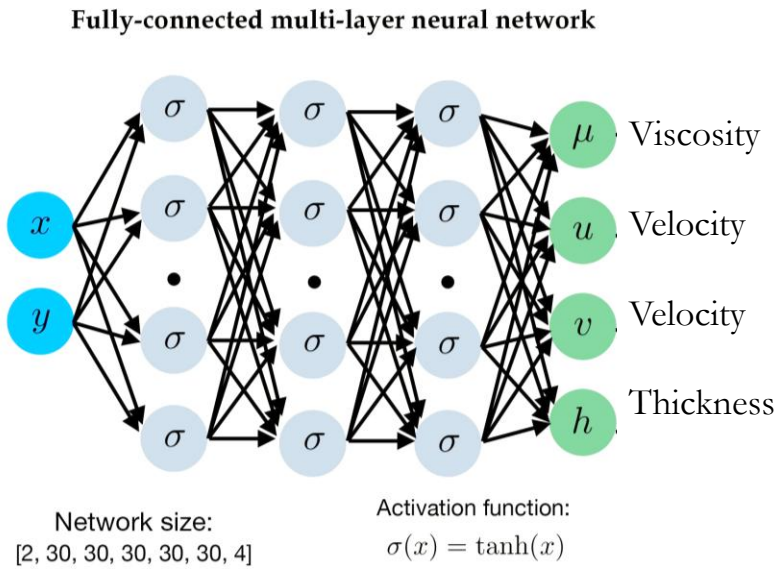


Riel and Minchew (2023):

Use physics-inform NN to infer $B(x, y)$

Glen's Flow Law: $\mu = \frac{B}{2}\dot{\varepsilon}_e^{\frac{1}{n}-1}$ $n = 3$

Ice viscosity inversion using physics-informed NN



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

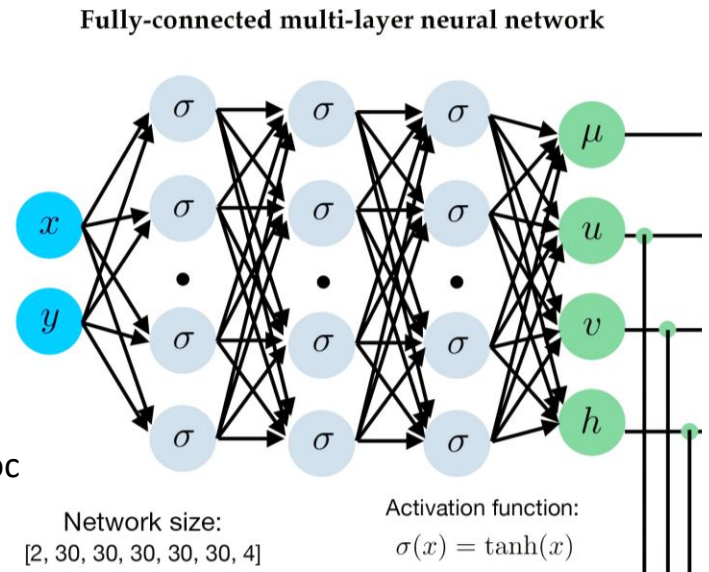
Here, we want to find $\mu(x, y)$

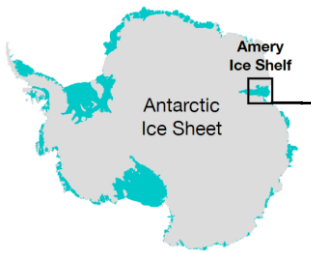
Glen's Flow Law $u = 2\dot{\varepsilon}_e^{\frac{1}{n}-1}$ $n = 3$

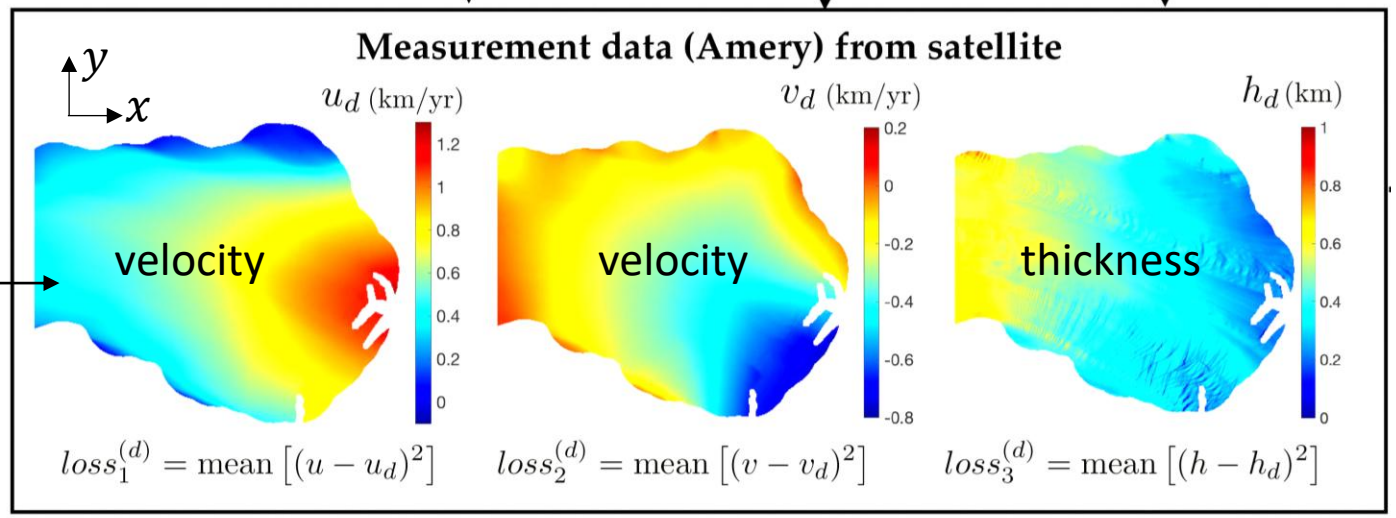
$\mu = ?$

Ice viscosity inversion using physics-informed NN

Yongji Wang, postdoc

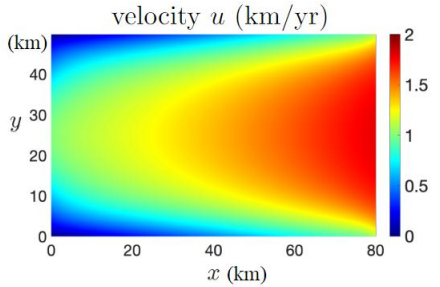


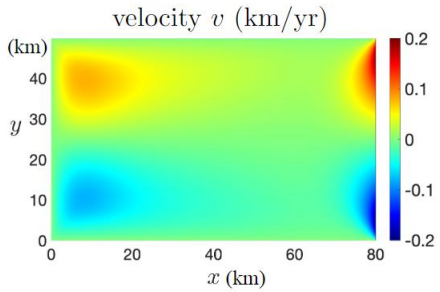


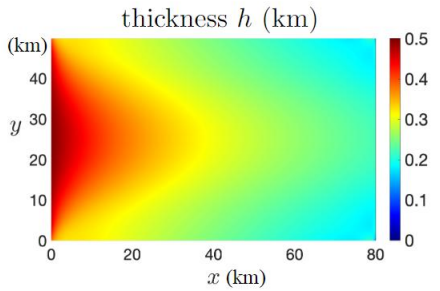


Ice viscosity inversion using PINN (synthetic data)

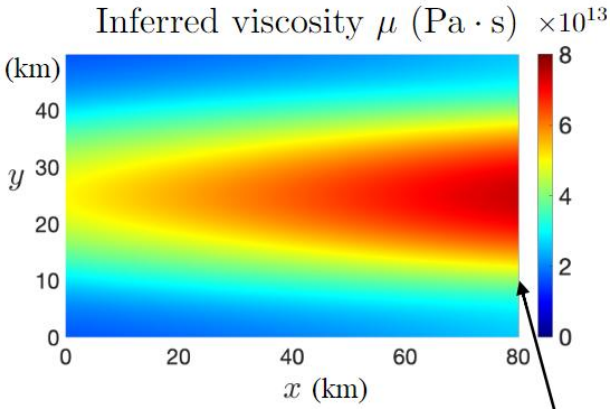
1. Neural network regression





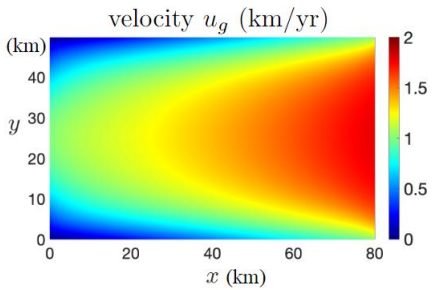


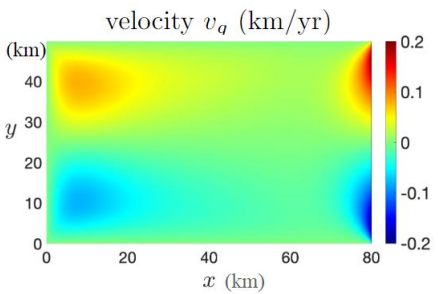
No training data, predicted purely based on the equations

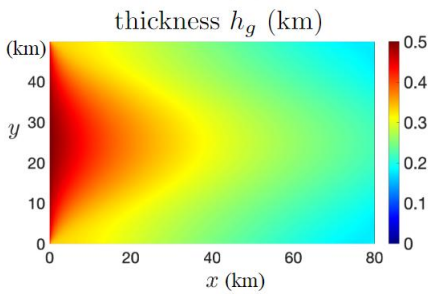


2. Ground truth (synthetic data)

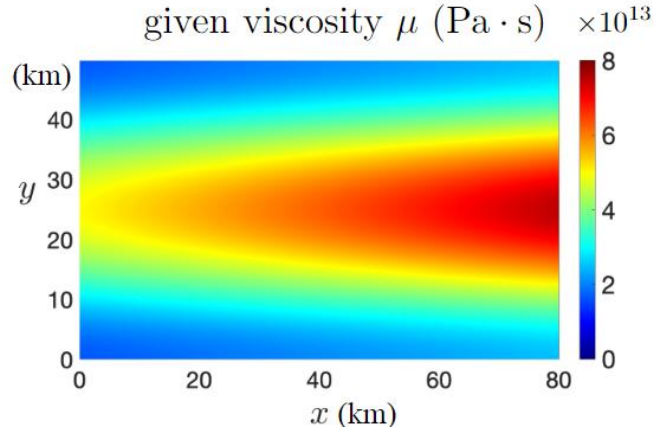
Training data







Ground truth viscosity

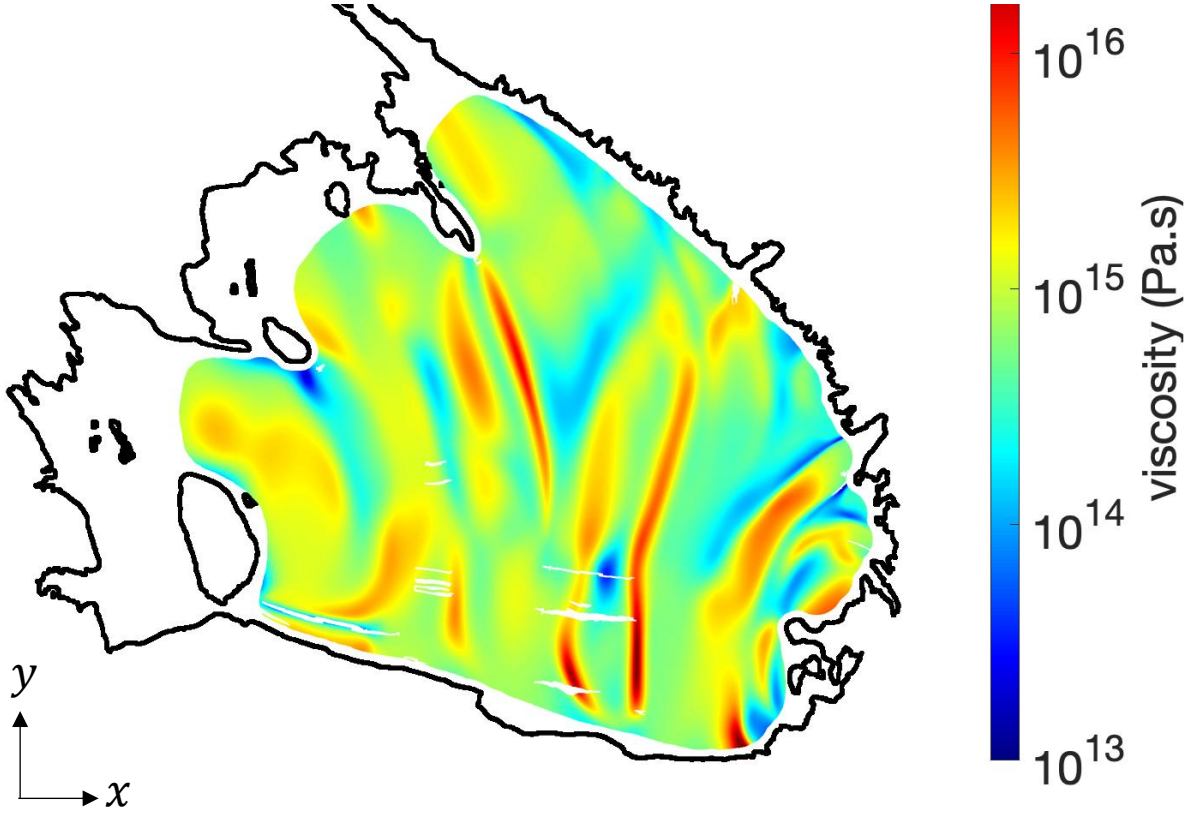


y
x

Colab: https://github.com/YaoGroup/DIFFICE_jax/tree/main/tutorial

Ice viscosity inversion using PINN

Ice effective viscosity μ



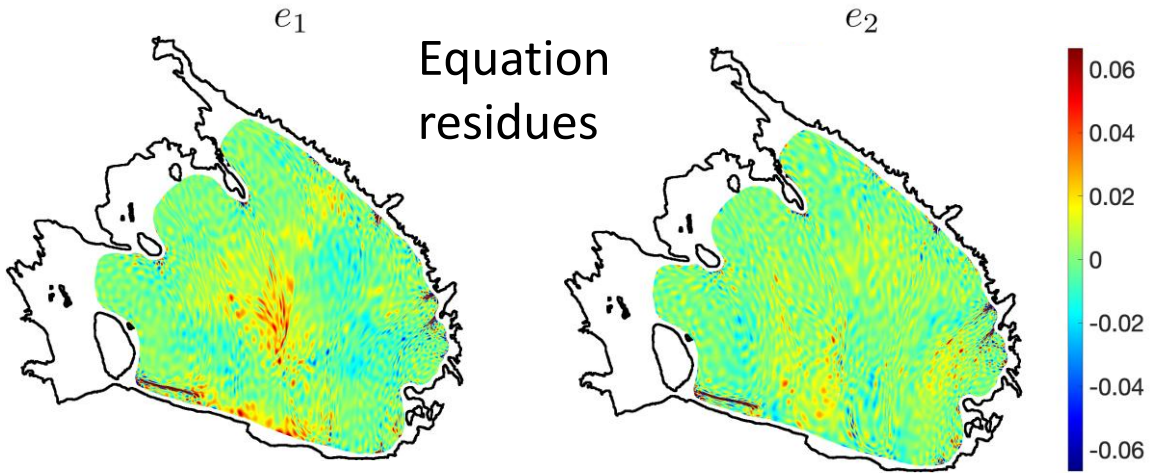
Wang, Lai, Prior, Cowen-Breen (2025).

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

Governing ice-shelf equations:

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

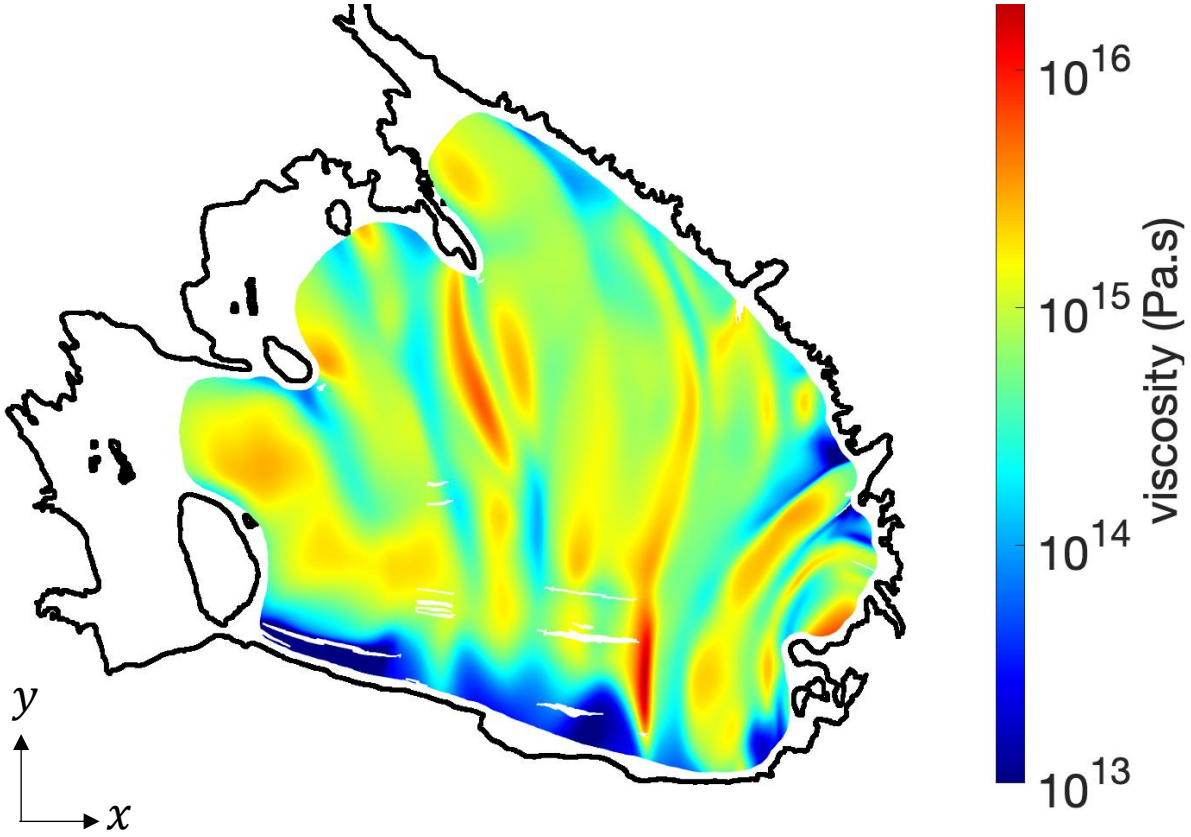
$$e_2 = \frac{\partial}{\partial y}\left(4\mu h\frac{\partial v}{\partial y} + 2\mu h\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - h\frac{\partial h}{\partial y}$$



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



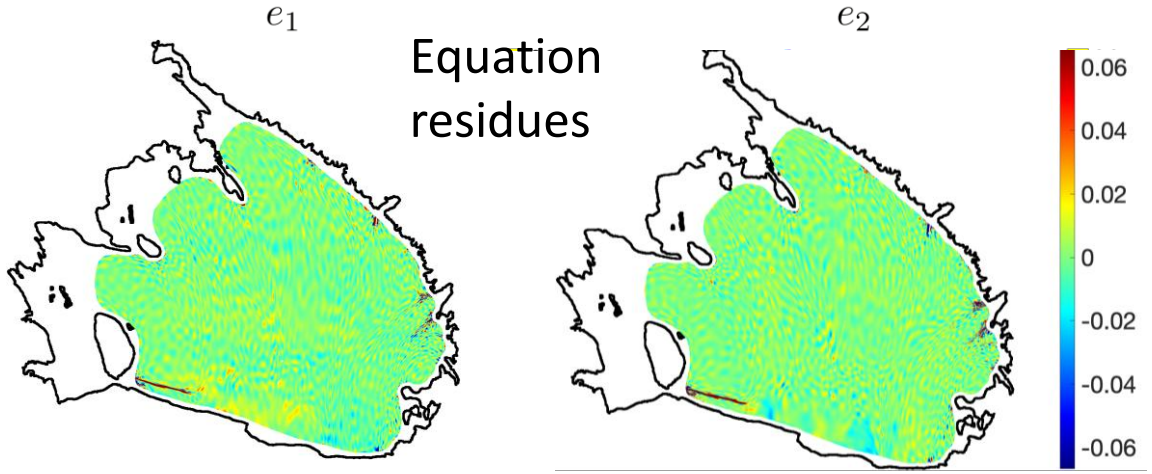
~~Assumption in equation: Isotropic viscosity~~

Anisotropic ice-shelf equations:

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

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

Equation residues



e1 and e2 are further reduced.

Wang, Lai, Prior, Cowen-Breen (2025).

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

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

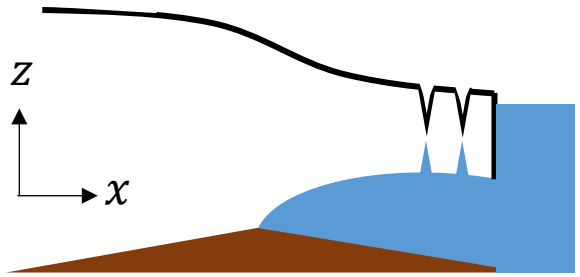
Under-constrained

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

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

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

Isotropic inversion fails to reduce the PDE residue

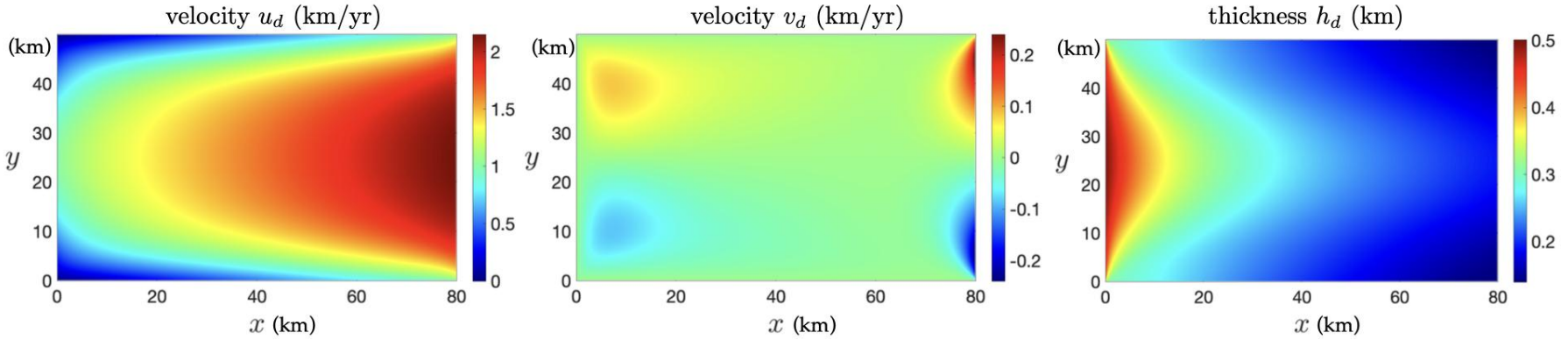
Isotropic Eqn:

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

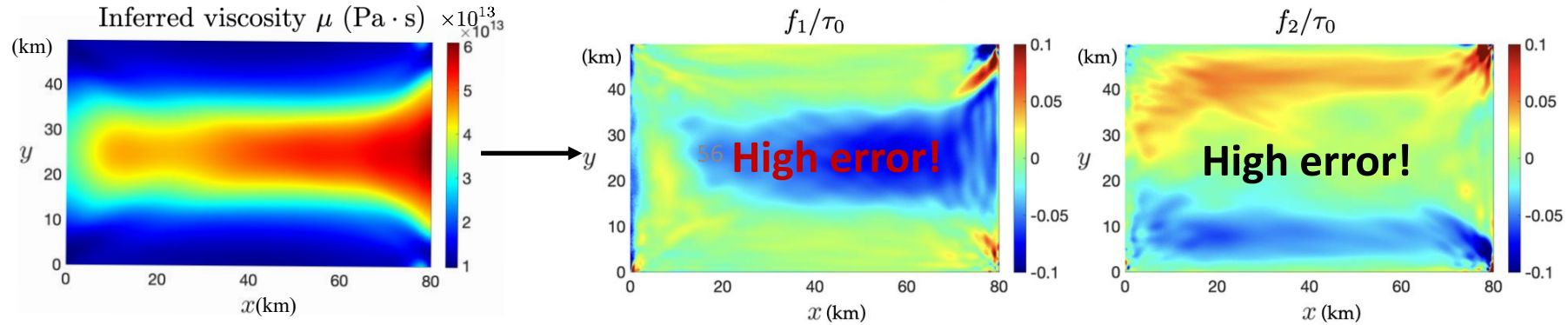
$$f_2 = \frac{\partial}{\partial y}\left(4\mu h\frac{\partial v}{\partial y} + 2\mu h\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - \rho_i g\left(1 - \frac{\rho_i}{\rho_w}\right)h\frac{\partial h}{\partial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

Anisotropic Data:

Synthetic data simulated using **anisotropic** viscosity



Predict: (infer)



Validation of anisotropic model

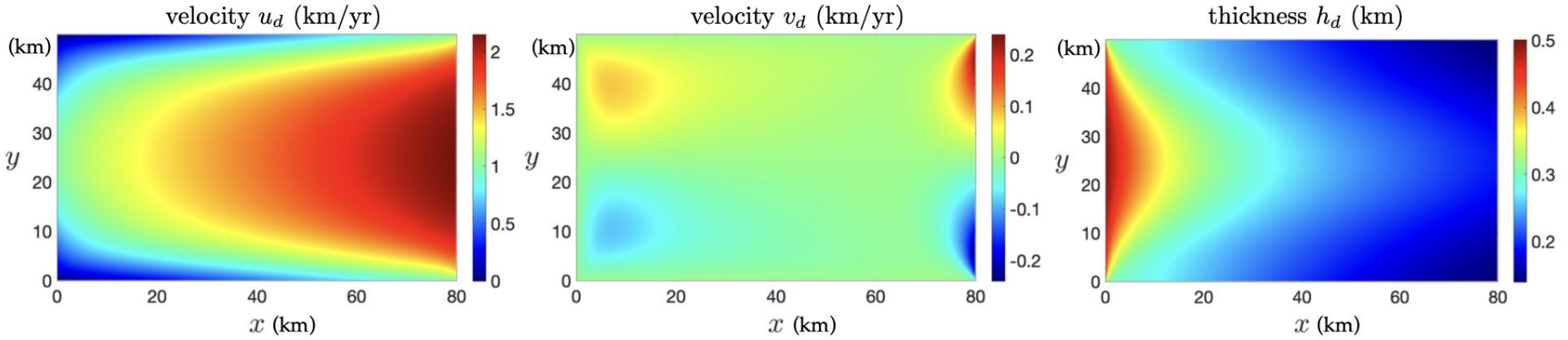
Anisotropic Eqn:

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

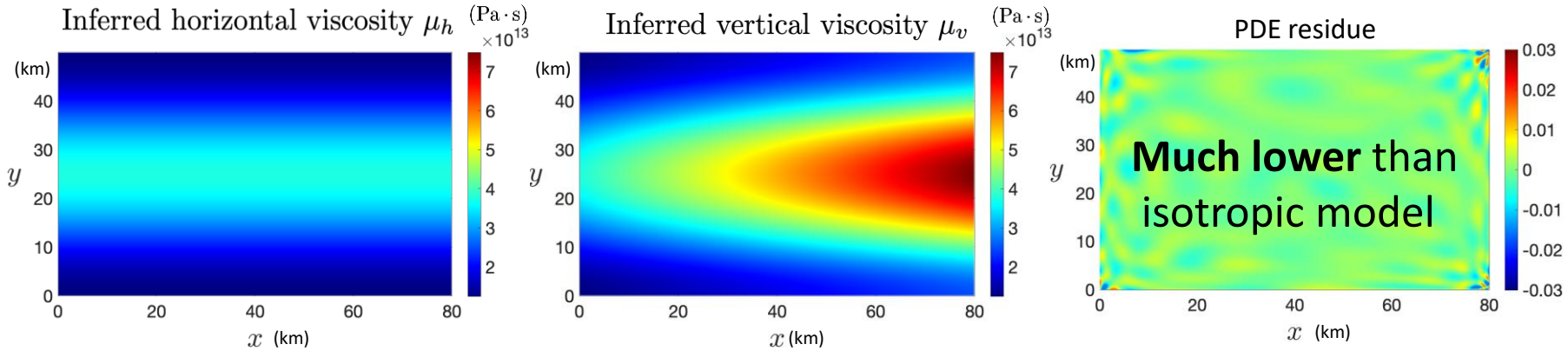
$$\frac{\partial}{\partial y}\left(2\mu_h h\frac{\partial v}{\partial y}+2\mu_v h\left[\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right]\right)+\frac{\partial}{\partial x}\left(\mu_h h\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]\right)=\rho_i g\left(1-\frac{\rho}{\rho_w}\right)h\frac{\partial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

Anisotropic Data:

Synthetic data simulated using **anisotropic** viscosity



Predict: (infer)



Two inversions perform equally well if data is isotropic

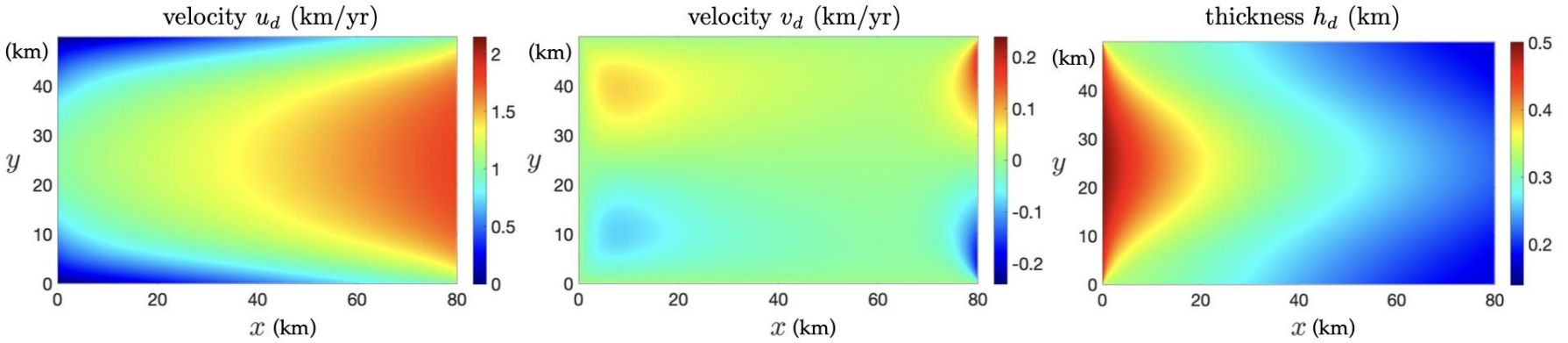
Isotropic Eqn:

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

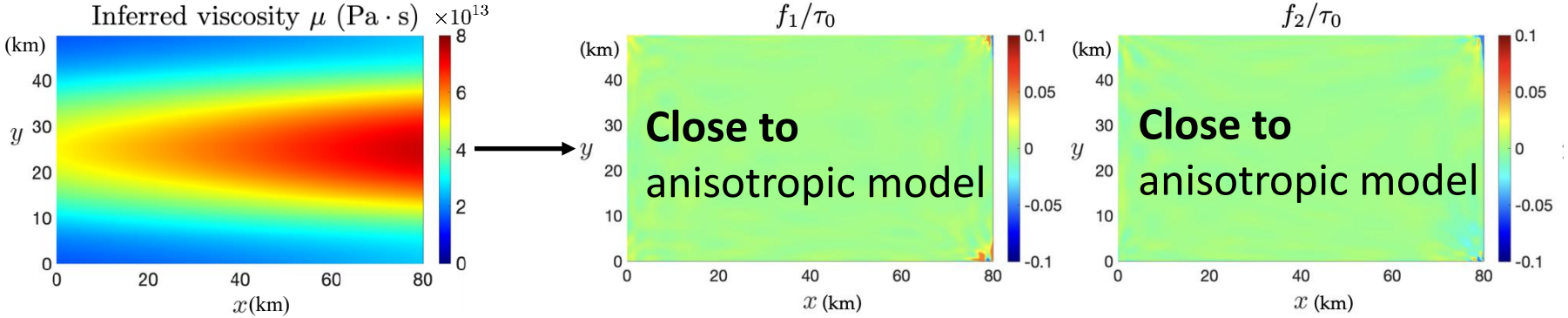
$$f_2 = \frac{\partial}{\partial y}\left(4\mu h\frac{\partial v}{\partial y} + 2\mu h\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu h\frac{\partial u}{\partial y} + \mu h\frac{\partial v}{\partial x}\right) - \rho_i g\left(1 - \frac{\rho_i}{\rho_w}\right)h\frac{\partial h}{\partial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

Isotropic Data:

Synthetic data simulated using **isotropic** viscosity



Predict: (infer)



Isotropic viscosity is a special case of anisotropic viscosity

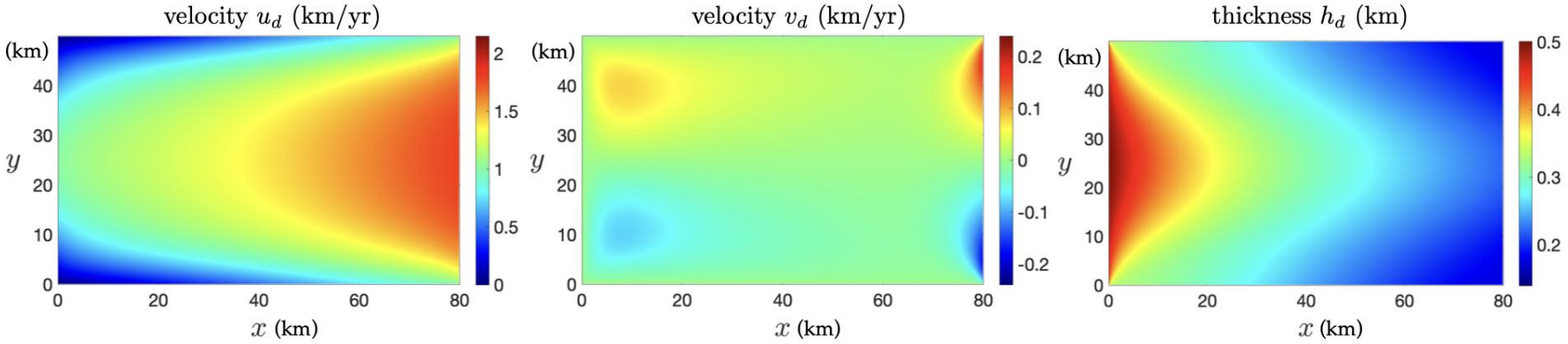
Anisotropic Eqn:

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

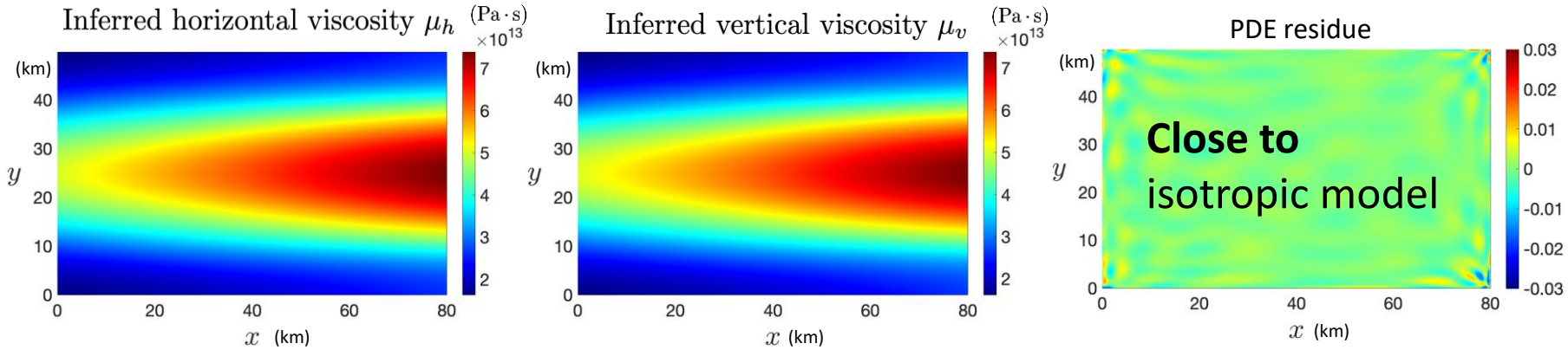
$$\frac{\partial}{\partial y}\left(2\mu_h h\frac{\partial v}{\partial y}+2\mu_v h\left[\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right]\right)+\frac{\partial}{\partial x}\left(\mu_h h\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]\right)=\rho_i g\left(1-\frac{\rho}{\rho_w}\right)h\frac{\partial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

Isotropic Data:

Synthetic data simulated using **isotropic** viscosity

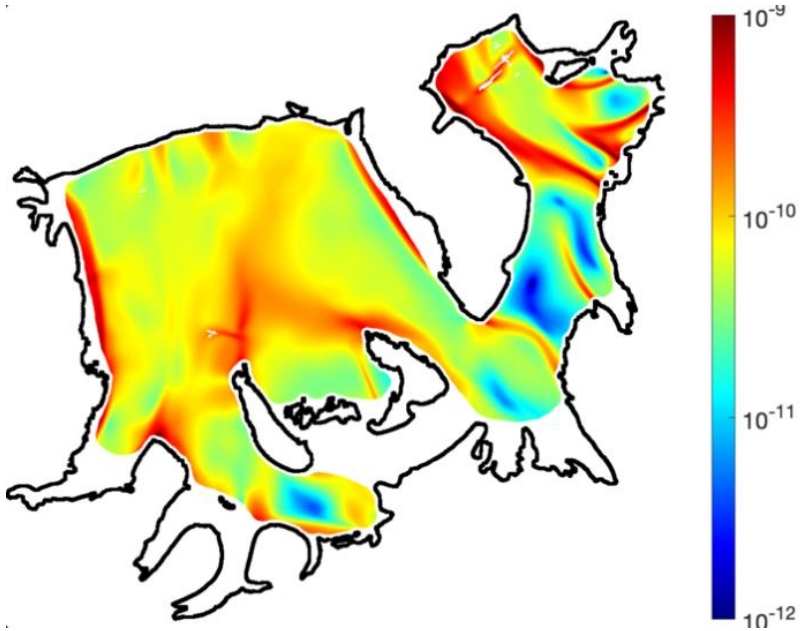


Predict: (infer)

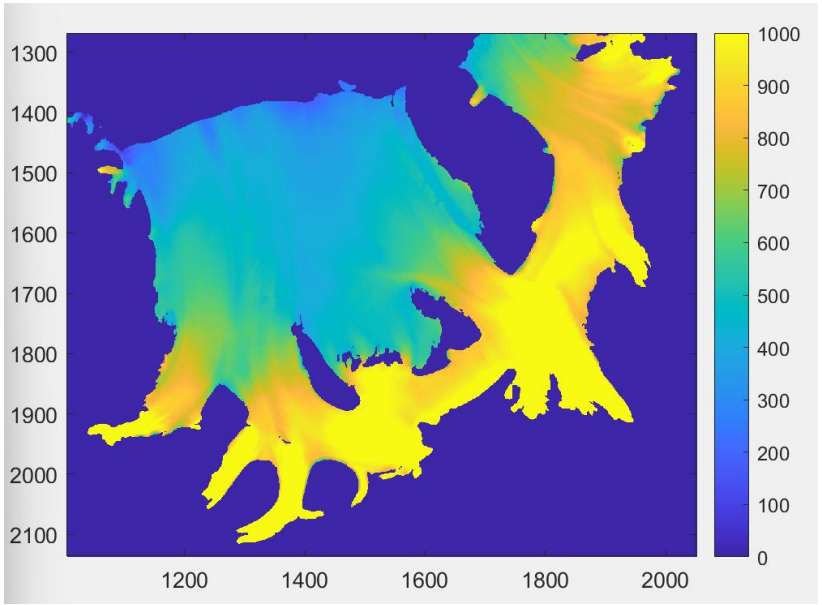


Viscosity patterns are not from data

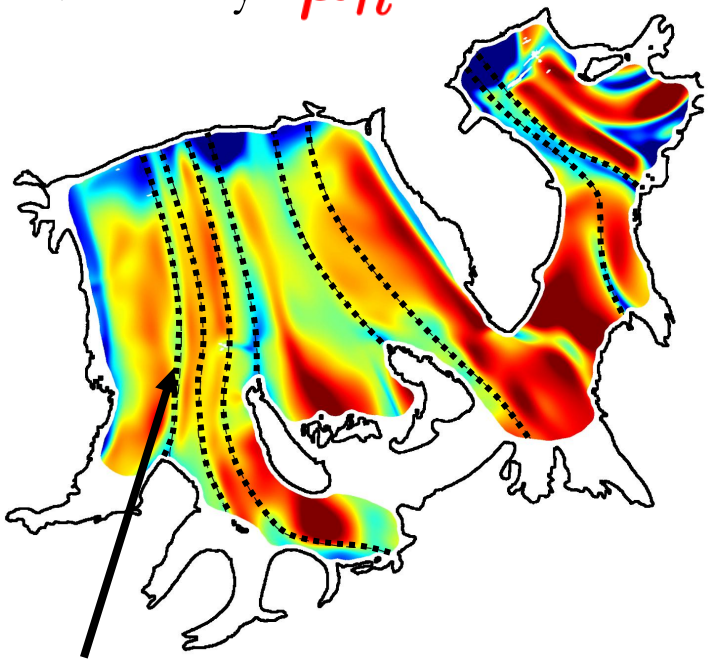
Strain rate (1/s)



Thickness (m)

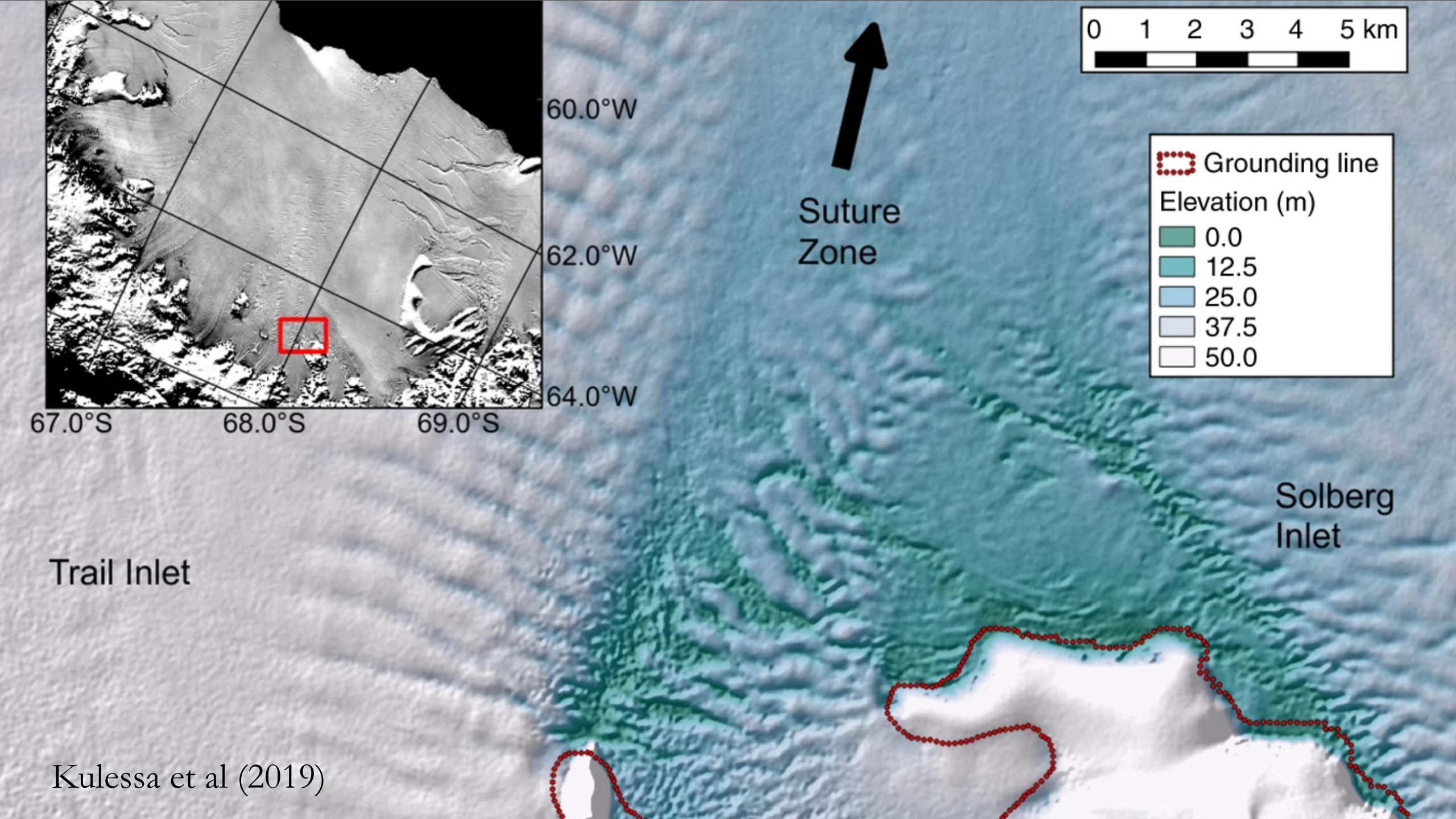


Anisotropic horizontal viscosity μ_h



Suture zones

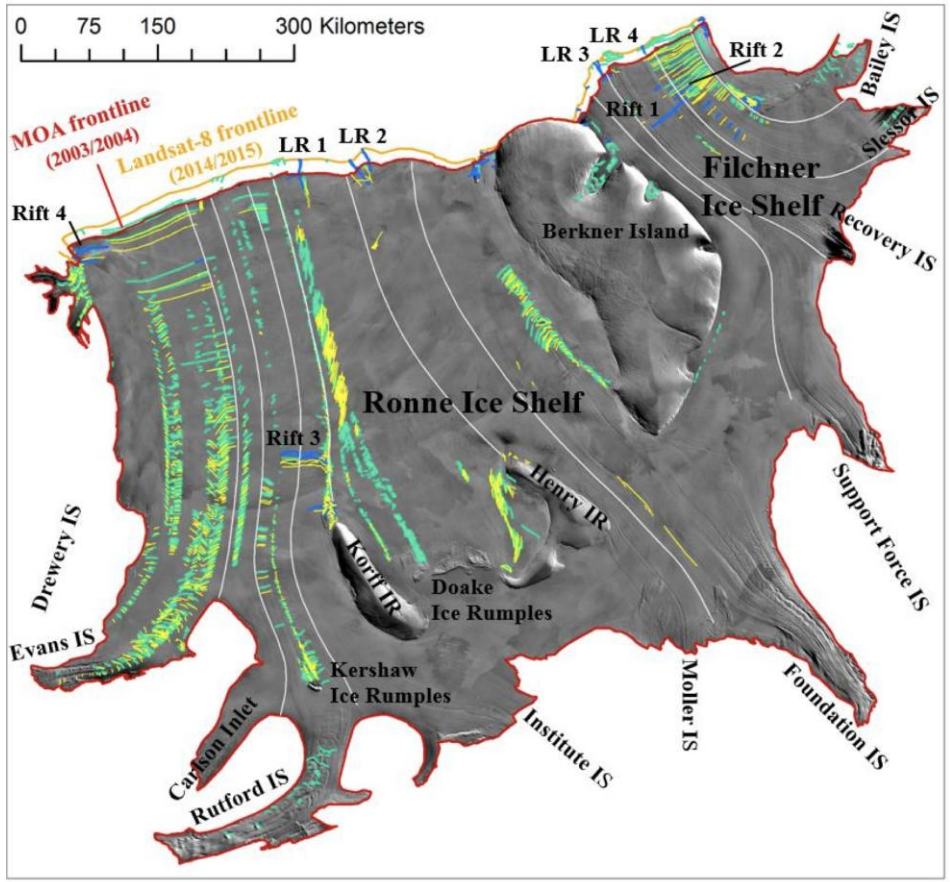
Why does viscosity vary spatially?



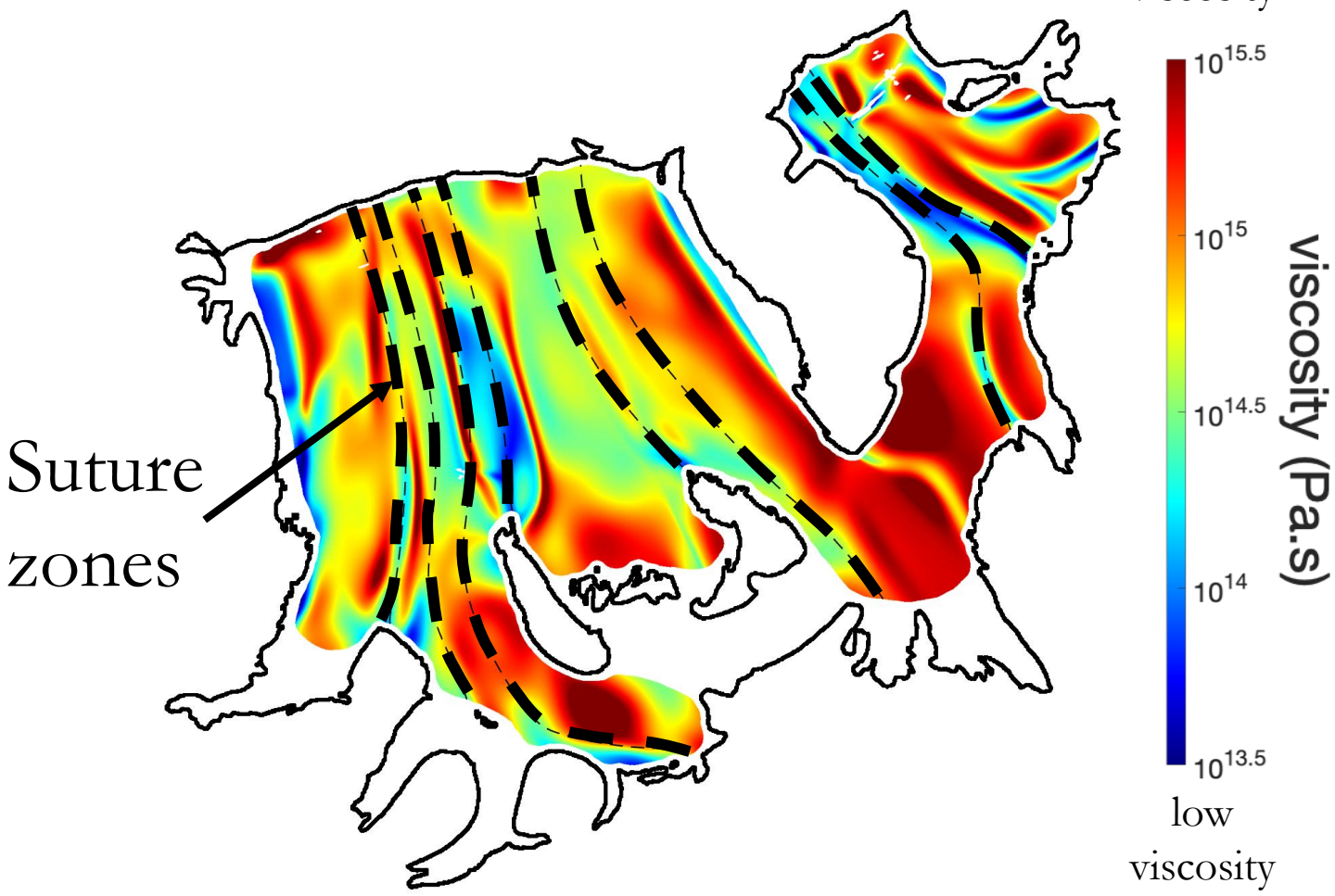
Kulessa et al (2019)

Suture zone → low-viscosity areas?

Li et al (2017)

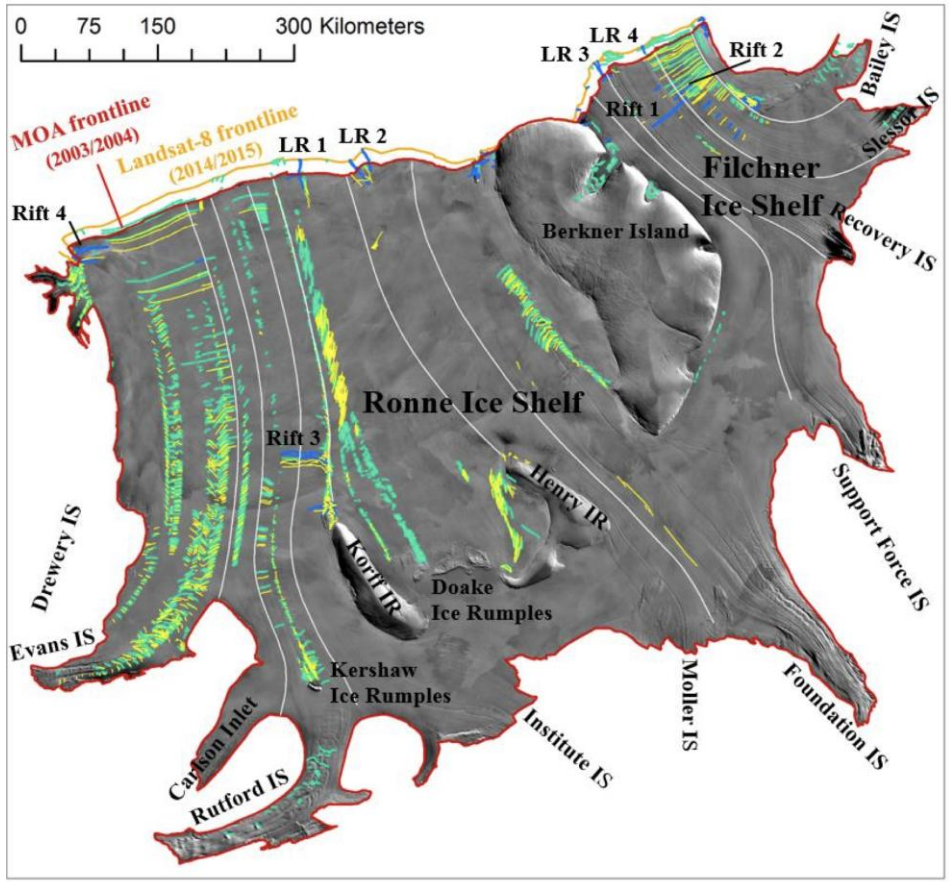


Isotropic viscosity μ

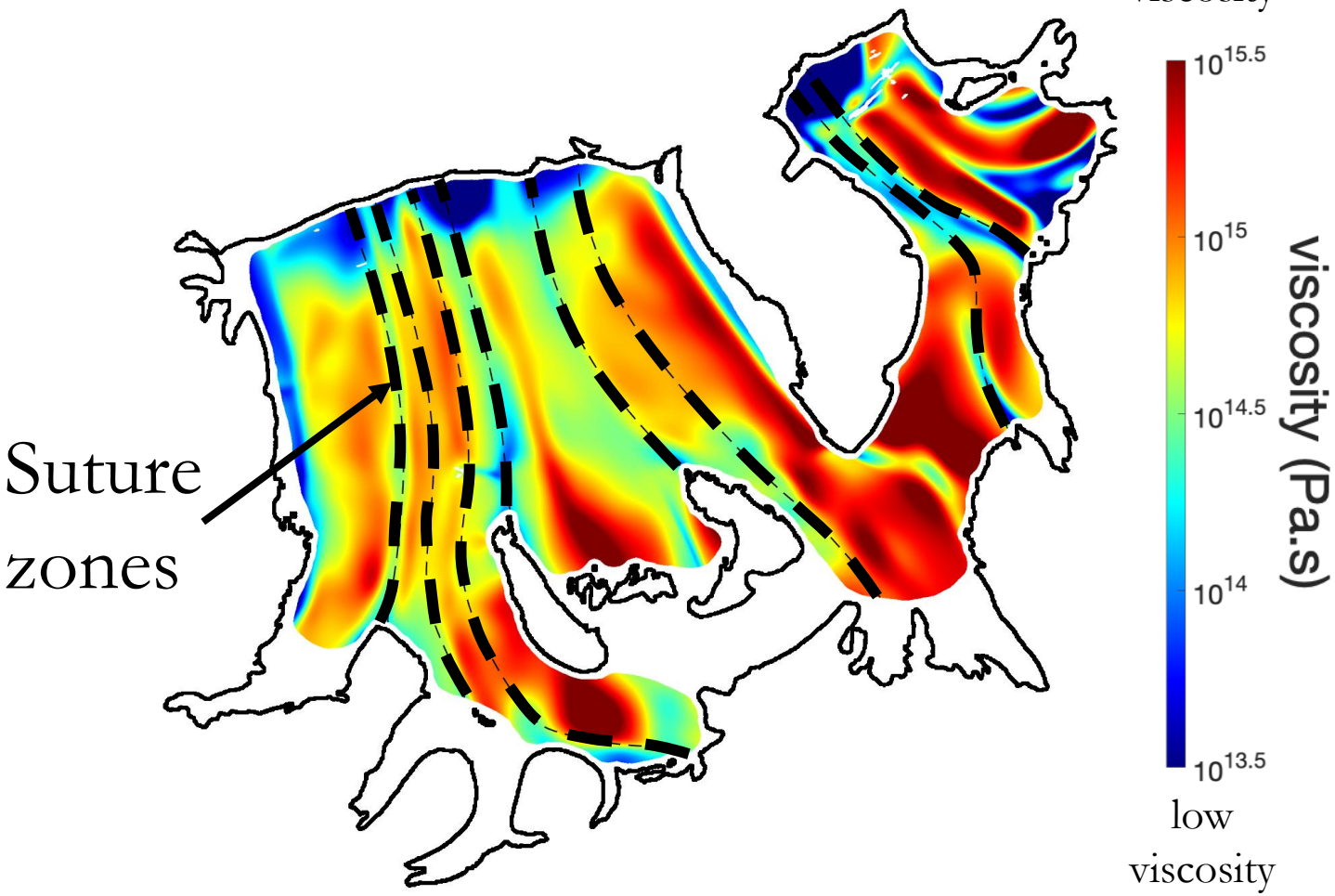


Suture zone matches with low-viscosity areas!

Li et al (2017)

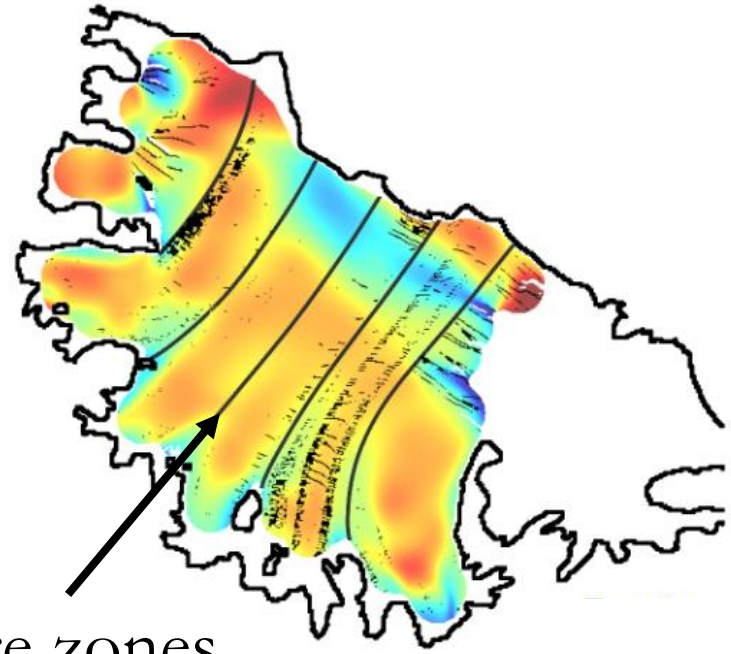


Anisotropic viscosity μ_h



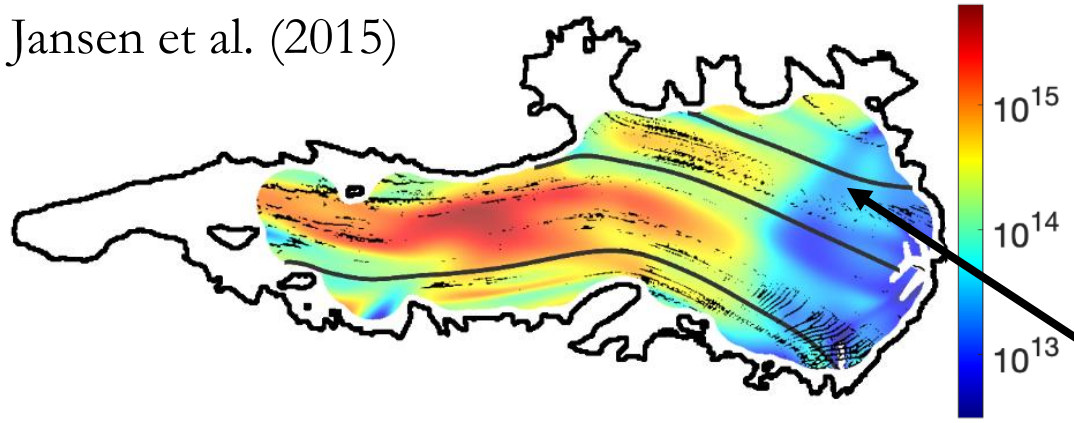
Without representing **low-viscosity suture zones** in ice-sheet models, it would be difficult to predict horizontal **crack/rift propagation**.

Suture zone matches with low-viscosity areas!



Suture zones

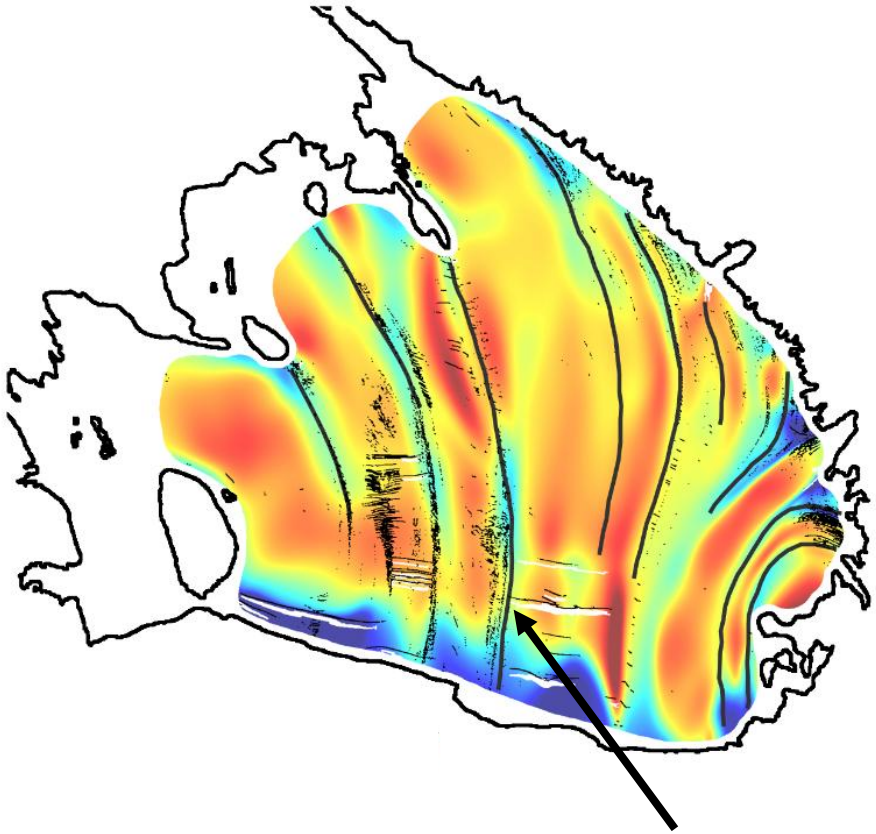
Jansen et al. (2015)



Suture zones

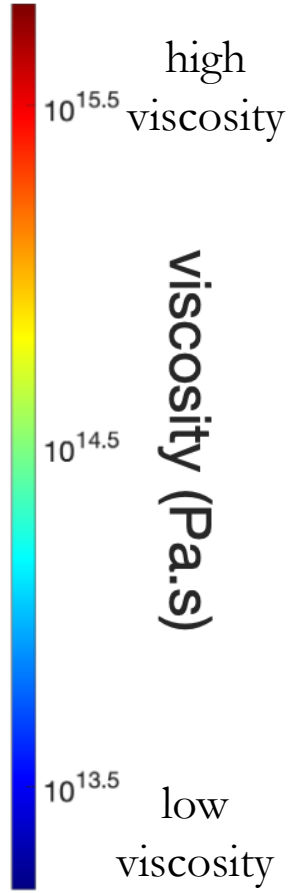
Kulessa et al. (2019)

Anisotropic horizontal viscosity μ_h

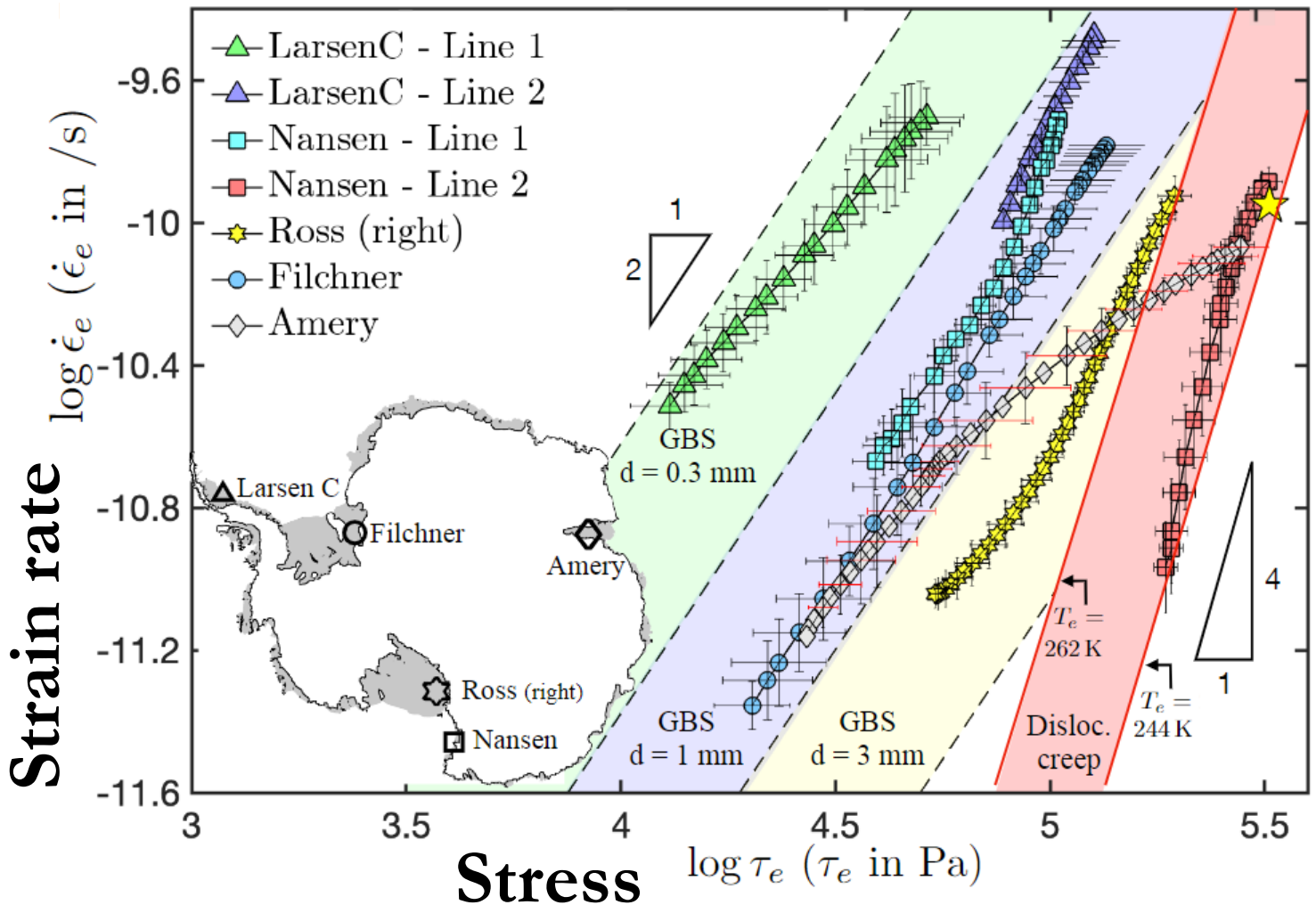


Suture zones

Christine et al. (2017)



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

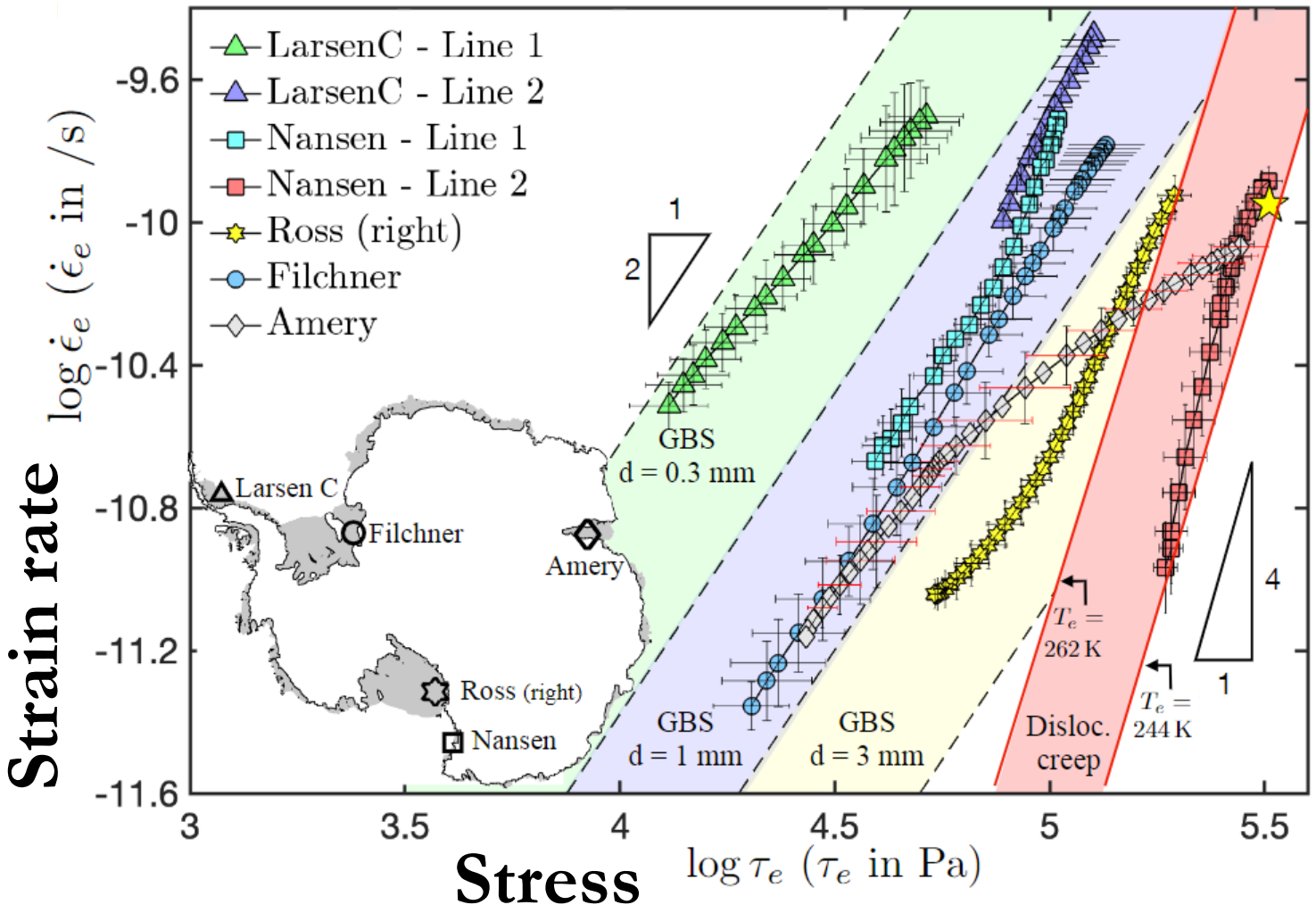
Error bars indicate the standard deviation of 15 repetition of PINN inversions with different random initialization.

$$\dot{\epsilon}_e = A_g d^{-1.4} \tau_e^{1.8} \exp\left(-\frac{Q_g}{RT}\right) + A_d \tau_e^{4} \exp\left(-\frac{Q_d}{RT}\right)$$

Grain boundary sliding (GBS) | 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 avaeraged)
d: grain size (fitted parameter)



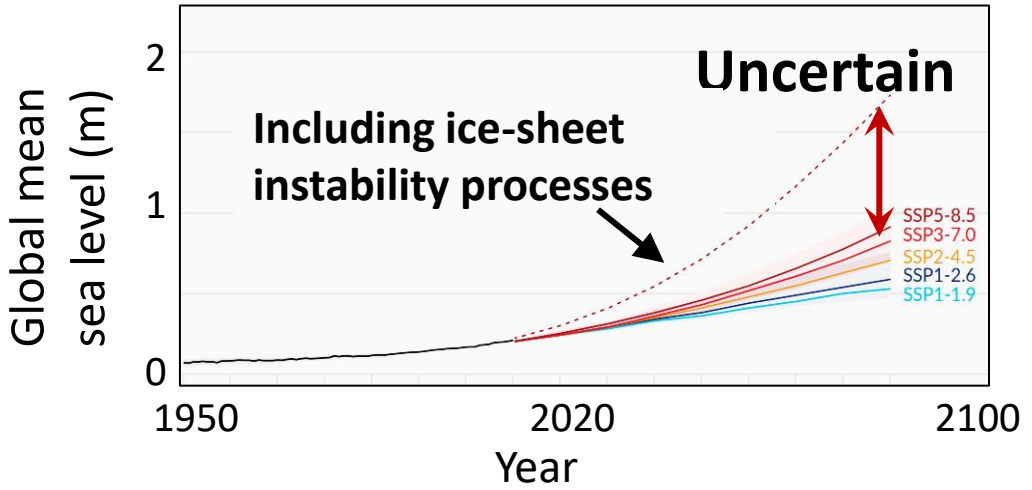
- In conventional ice sheet models, n=3 is assumed.
- But here we observe: n increases with stress
  Composite rheology:
  - - - Grain boundary sliding (n=2)
  — Dislocation Creep (n=4)
- Amery upstream close to n=1 (Field work needed)

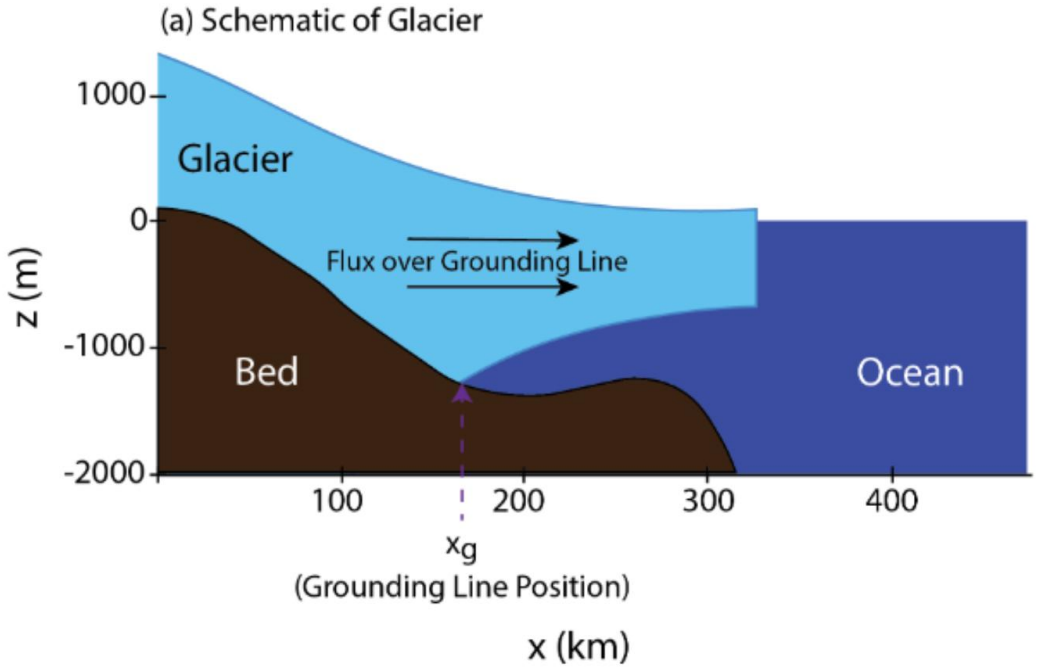
Wang, Lai, Prior, Cowen-Breen, Science (2025).

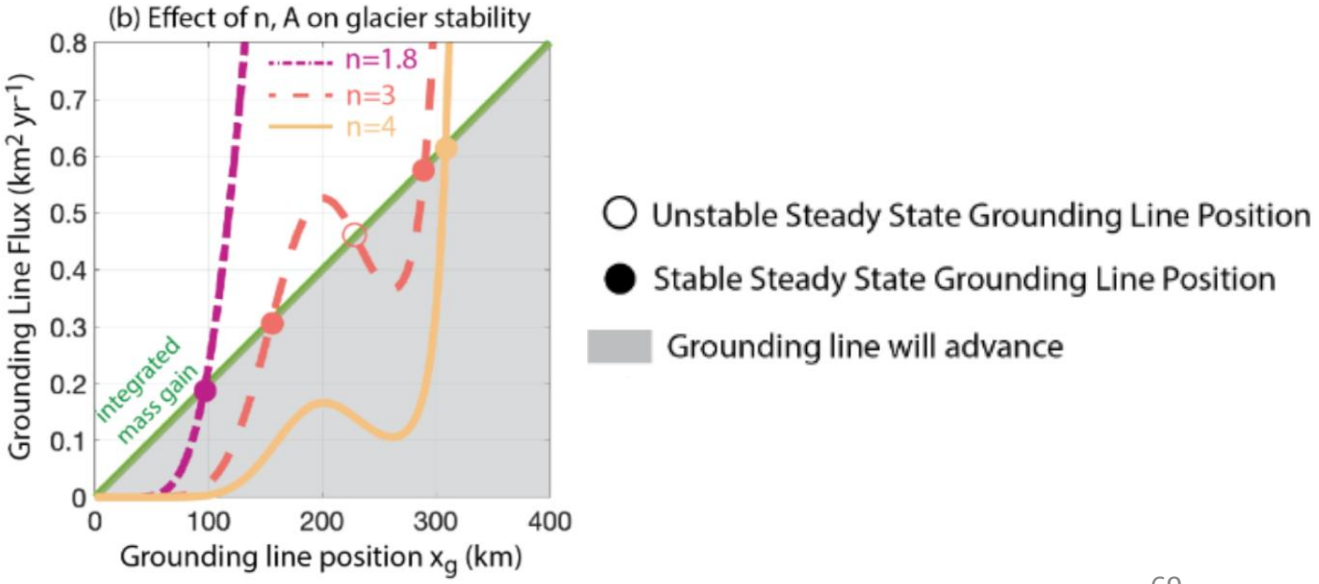
Why is stress exponent “n” important?

IPCC AR6 (2021)

Ranganathan and Minchew, (2024):
“The deviation from the canonical n=3 changes the stability portrait of marine ice sheets by reducing the likelihood of unstable steady-state configurations on reverse bed slopes under given climate conditions”







Why is stress exponent “n” important?

Geophysical Research Letters®

RESEARCH LETTER
10.1029/2024GL112516

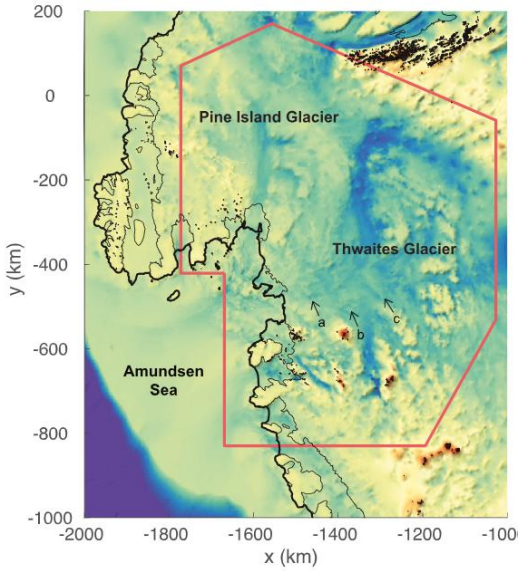
Key Points:

- Increasing Glen–Nye exponent from $n = 3$ to 4 leads to 32 ± 14% greater Amundsen Sea Embayment ice loss by 2100, 70 ± 15% by 2300 in our model
- Uncertainty in ice loss due to n is

Increasing the Glen–Nye Power-Law Exponent Accelerates Ice-Loss Projections for the Amundsen Sea Embayment, West Antarctica

Benjamin Getraer[1] and **Mathieu Morlighem**[1]

[1]Department of Earth Sciences, Dartmouth College, Hanover, NH, USA

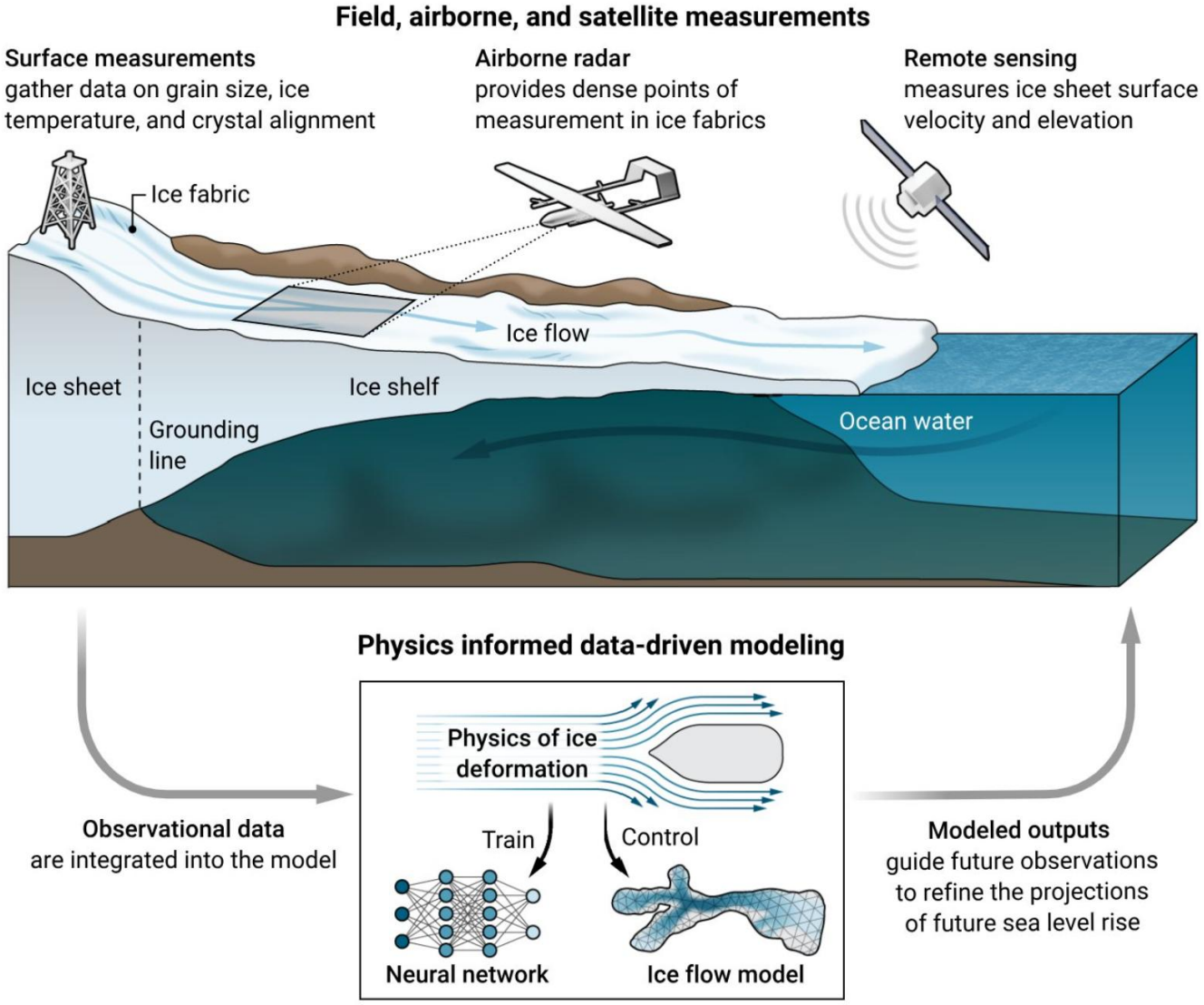


Getraer and Morlighem (2025):

- Increasing Glen–Nye exponent from n=3 to 4 leads to 32±14% greater Amundsen Sea Embayment ice loss by 2100, 70±15% by 2300 in our model
- Uncertainty in ice loss due to n is comparable to uncertainty from different high-emission (RCP8.5) climate forcings by 2100
- Regions most sensitive to n are located where changes in strain rates are largest, such as close to the grounding line and shear margins

How can we learn new physics if the ingested physics itself is known?

How can we learn new physics if the ingested physics itself is known?



Method:

We inject only the physics that we believe are correct, e.g. **conservation of momentum**, into the loss function, and leaving the uncertain physics, e.g. the **viscosity model** $\mu =?$, to be determined through data assimilation using observations.

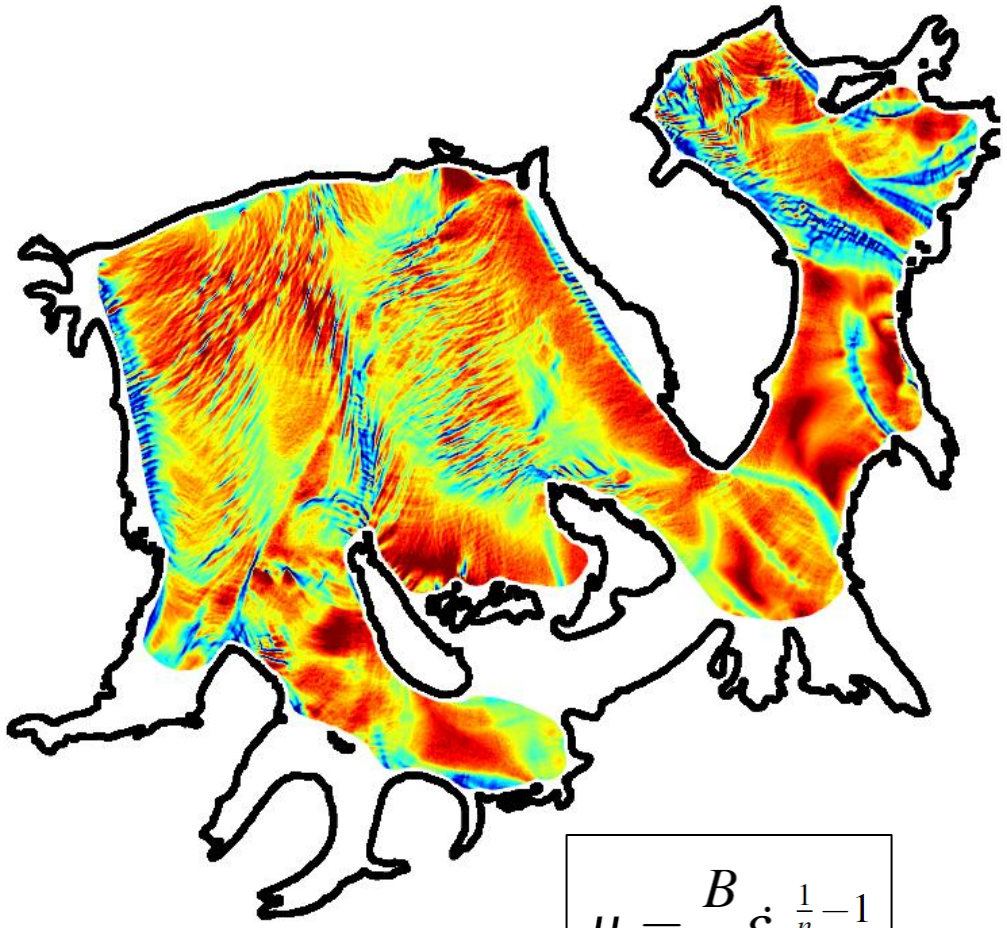
Wang, Lai, Prior, Cowen-Breen, *Science* (2025)

Similarities and differences between PINNs and the adjoint method

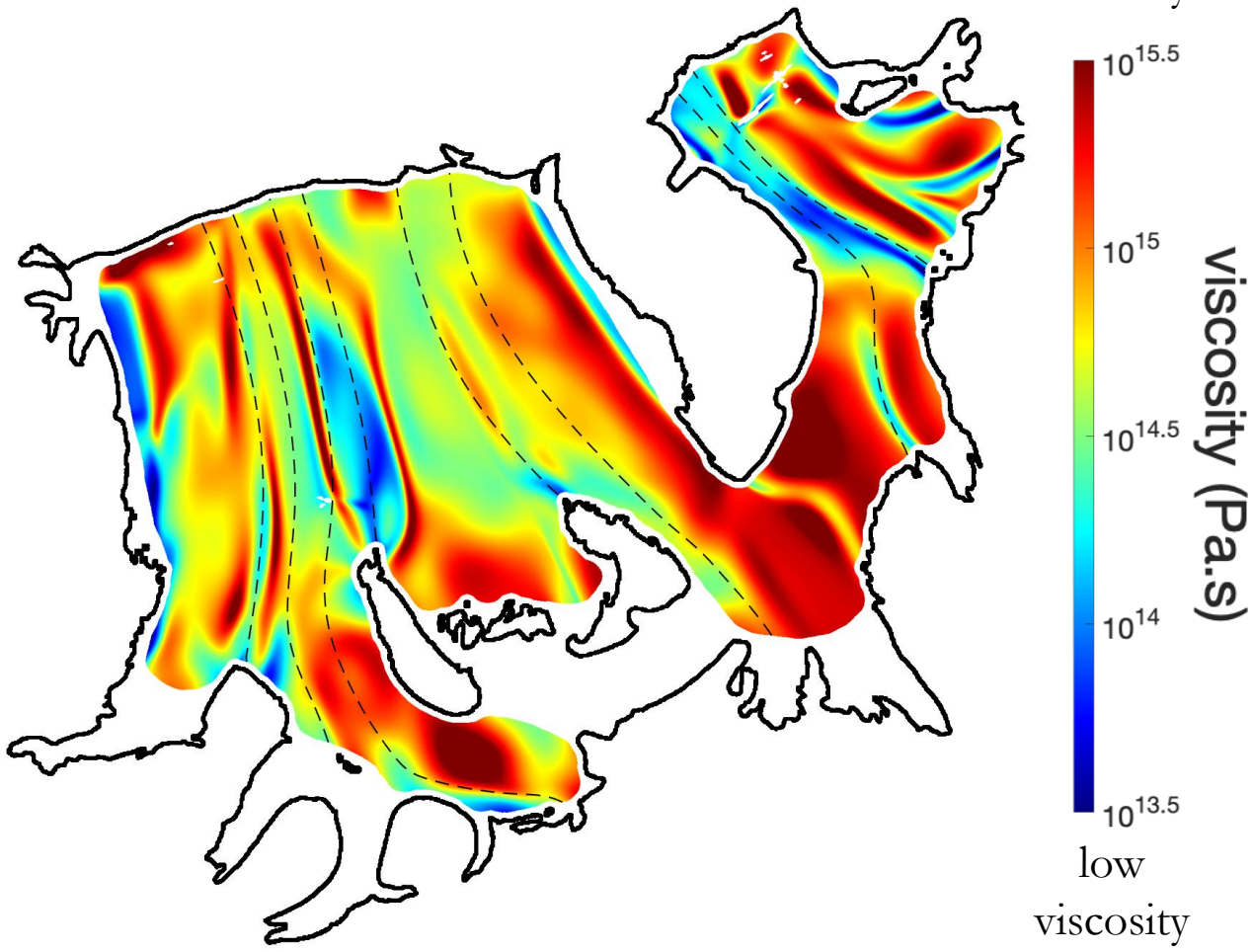
Comparison (isotropic viscosity)

Furst et al (2016) with control method

PINN



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



Similarities and differences

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

Momentum equation:

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

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

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

Boundary condition at the calving front:

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

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

i. Control Method (1993):

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

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)

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 constrains:

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

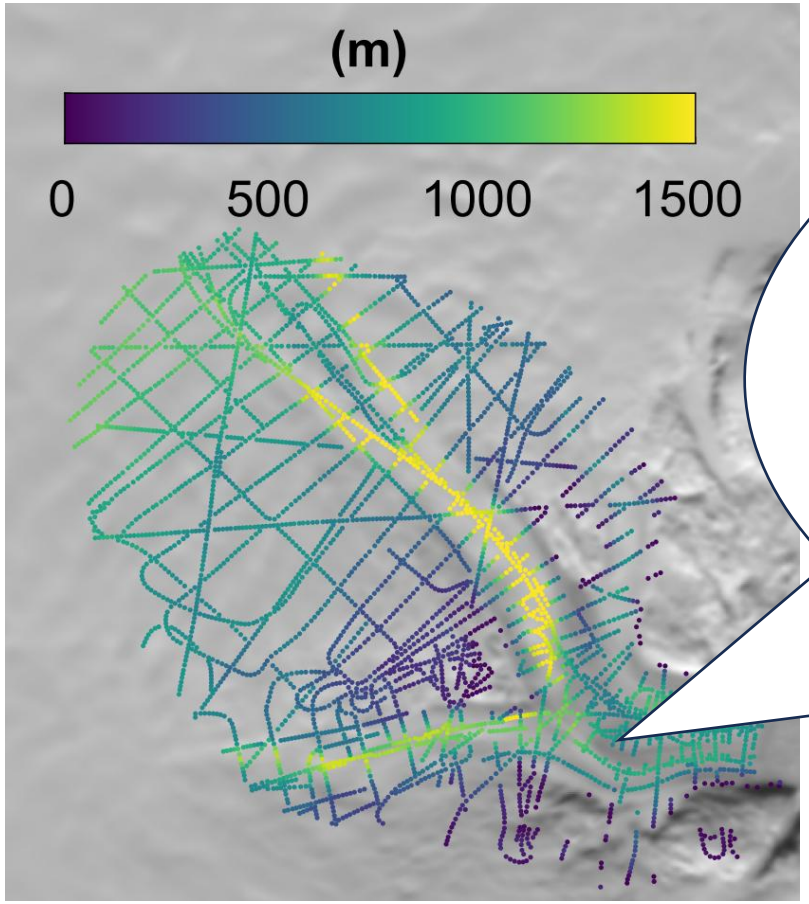
Notations:
modeled quantity: q
observed quantity: q_{obs}

Thickness data is actually sparse/sporadic

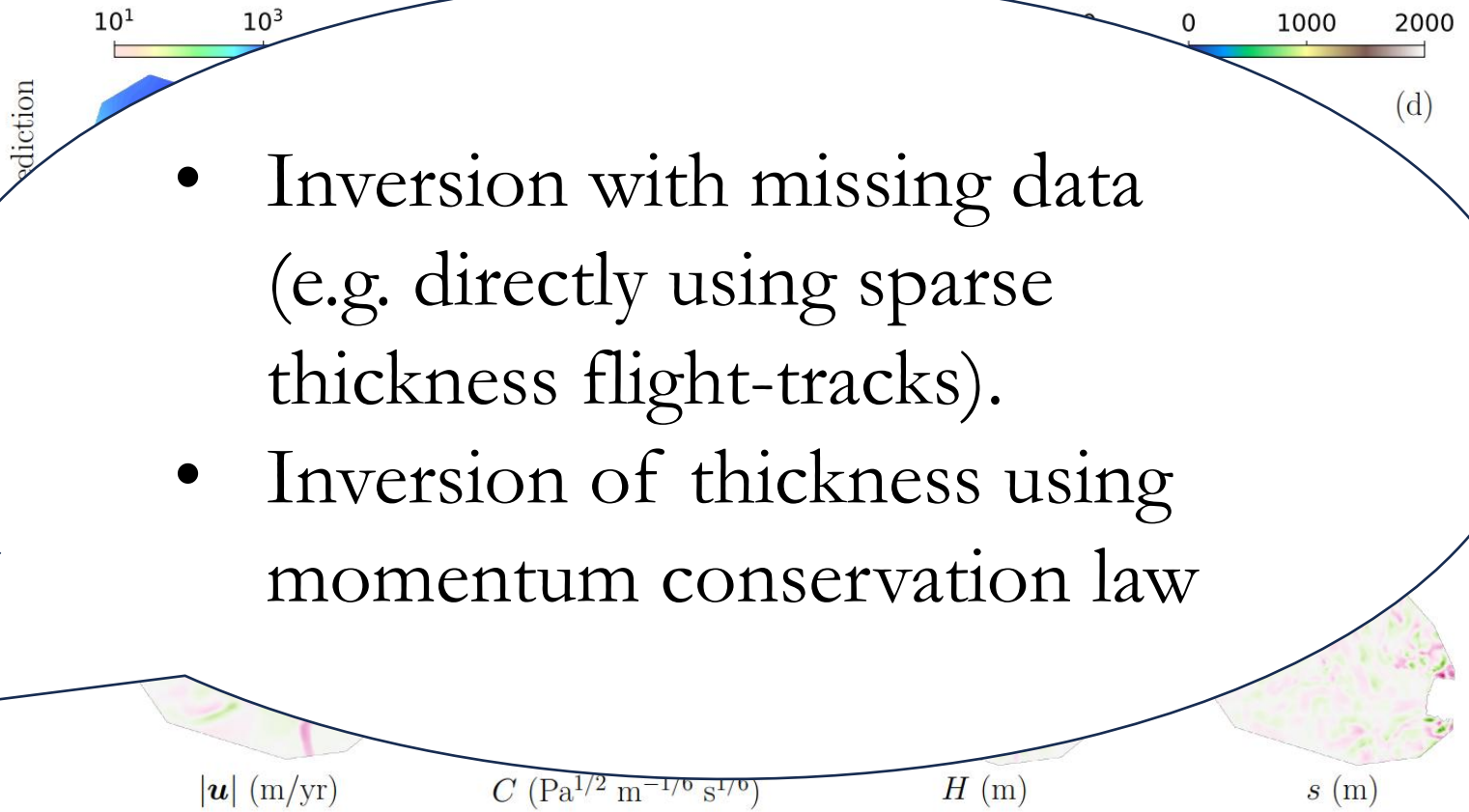
Cheng, Morlighem, and Francis, JGR: Machine Learning and Computation (2024)

Thickness data is actually sparse

The actual observed thickness



Inferring both ice thickness and basal friction

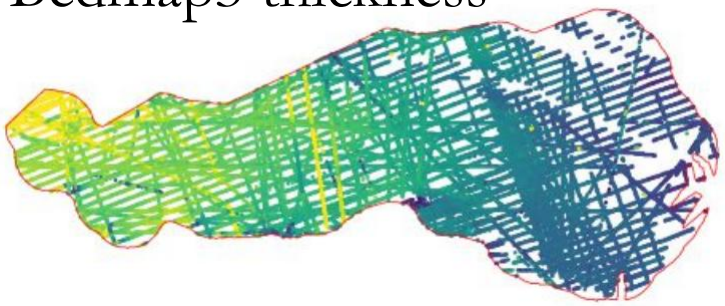


- Inversion with missing data (e.g. directly using sparse thickness flight-tracks).
- Inversion of thickness using momentum conservation law

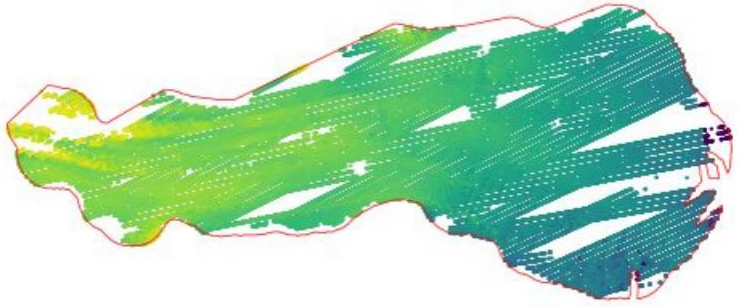
Infer viscosity and thickness with sporadic thickness data (last year's project)

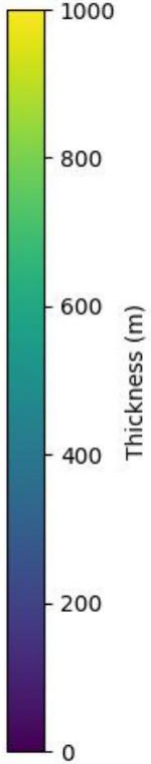
Data:

Bedmap3 thickness



IceSAT-2 thickness

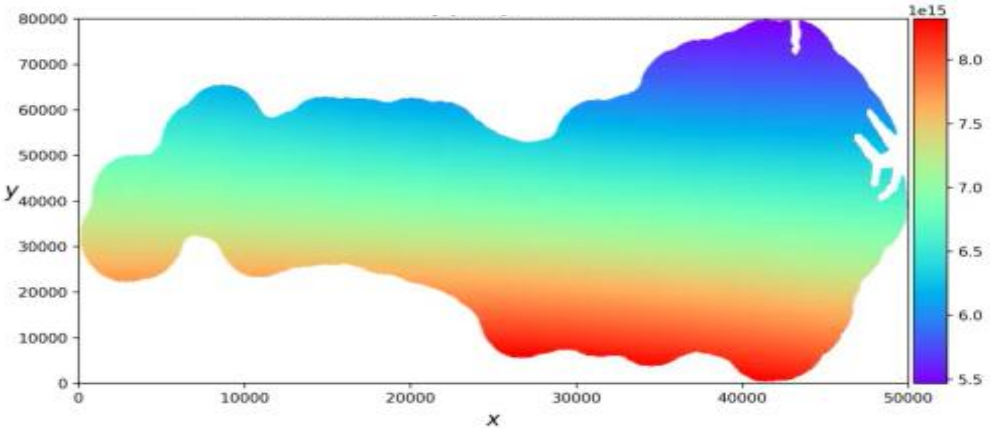




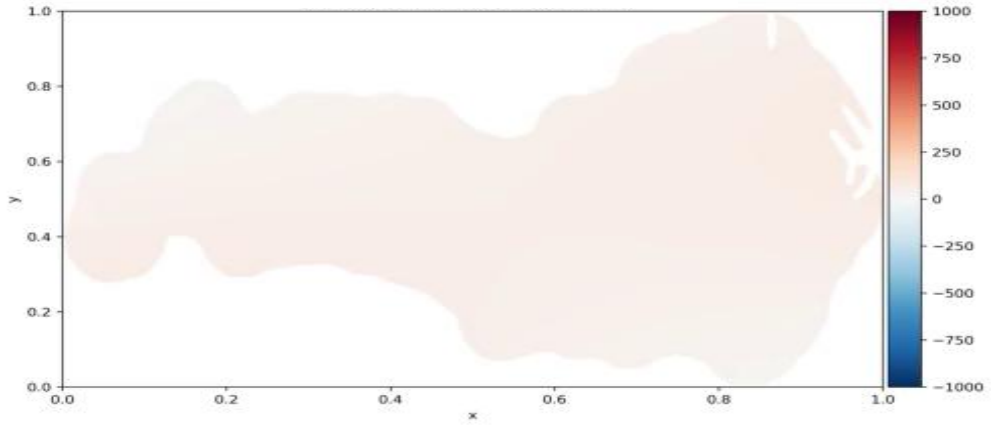
Predict (infer):

- Adam Iteration 0

Viscosity:



PDE residue:



Colabs:

2D ice viscosity inversion

https://github.com/YaoGroup/DIFFICE_jax

DIFFICE.jax v1.0: 2D inversion using PINN

DIFFerentiable neural-network solver for data assimilation of ICE shelves written in JAX.

DIFFICE_jax

Search docs

INVERSE PROBLEMS:
- Isotropic viscosity inversion
- Anisotropic viscosity inversion

EXTENDED-PINNS:
- Why we need XPINNs
- Loss function for XPINNs

TUTORIAL:
- Overview
- Forward problem setup
- File descriptions
- Results

TRAINING DATA:
- Data source
- Data Format

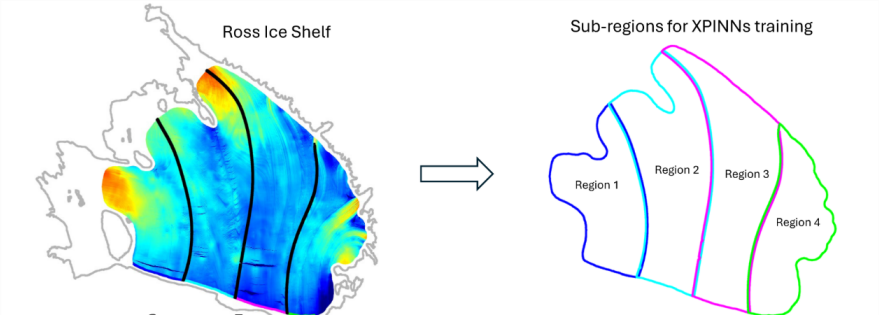
REAL-DATA EXAMPLES:
- Overview
- Isotropic case via PINNs
- Isotropic case via XPINNs

DIFFICE_jax documentation

DIFFICE_jax documentation

Introduction

`DIFFICE_jax` is a Python package that solves the depth-integrated Stokes equation for **ice shelves**, and can be adopted for **ice sheets** by modifying the partial differential equations (PDE) in the neural network loss function. It uses PDEs to interpolate descretized remote-sensing data into meshless and differentible functions, and infer ice shelves' viscosity structure via back-propagation and automatic differentiation (AD). The algorithm is based on physics-informed neural networks (PINNs) and implemented in JAX. The `DIFFICE_jax` package involves several advanced features in addition to vanilla PINNs algorithms, including collocation points resampling, non-dimensionalization of the data adnd equations, extended-PINNs (XPINNs) (see figure below), viscosity exponential scaling function, which are essential for accurate inversion. The package is designed to be user-friendly and accessible for beginners. The Github respository also provides **Colab Notebooks** for both the synthetic and real ice-shelf examples in the `tutorial` and `examples` folders, respectively.



2D SSA: https://github.com/YaoGroup/DIFFICE_jax
- Synthetic case
- Real data case

Package can be installed via
`python -m pip install DIFFICE_jax`

Wang and Lai, J. Open Source Softw. (2025). doi.org /10.21105/joss.07254

Algorithm features

Wang and Lai, J. Open Source Softw. (2025). doi.org /10.21105/joss.07254

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

$$\mu = \exp(\mathrm{NN}_\mu),$$

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.

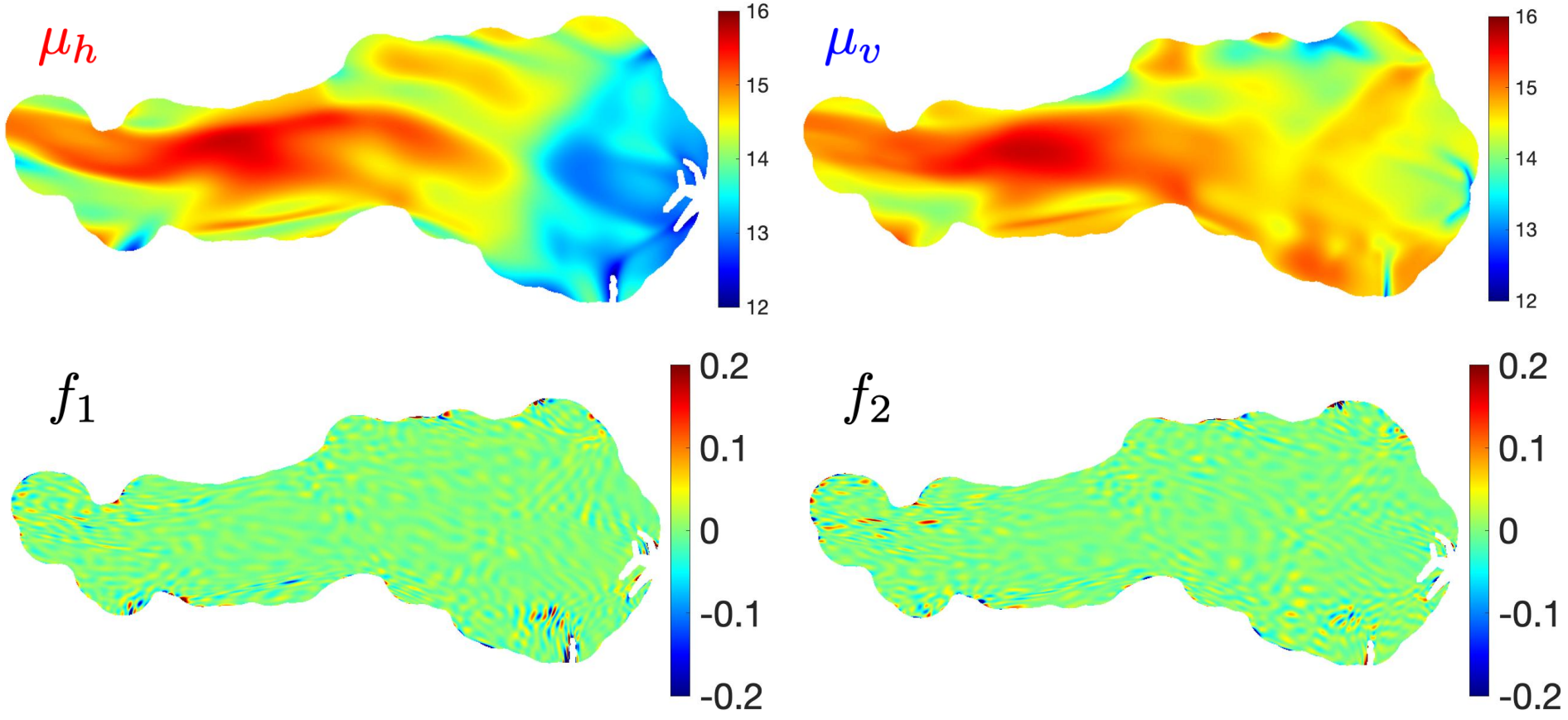
(5) Extended-PINNs (XPINNs) for large ice shelve: Large ice shelves, such as Ross, poses a multiscale challenge for capturing both local-scale and large-scale spatial variation of u, v, h, μ. These local variations are difficult to capture with a single NN due to the spectral biases of NNs [@rahaman2019spectral,@xu2020frequency]. 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.

Backup Slides

Anisotropic viscosity of Amery

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

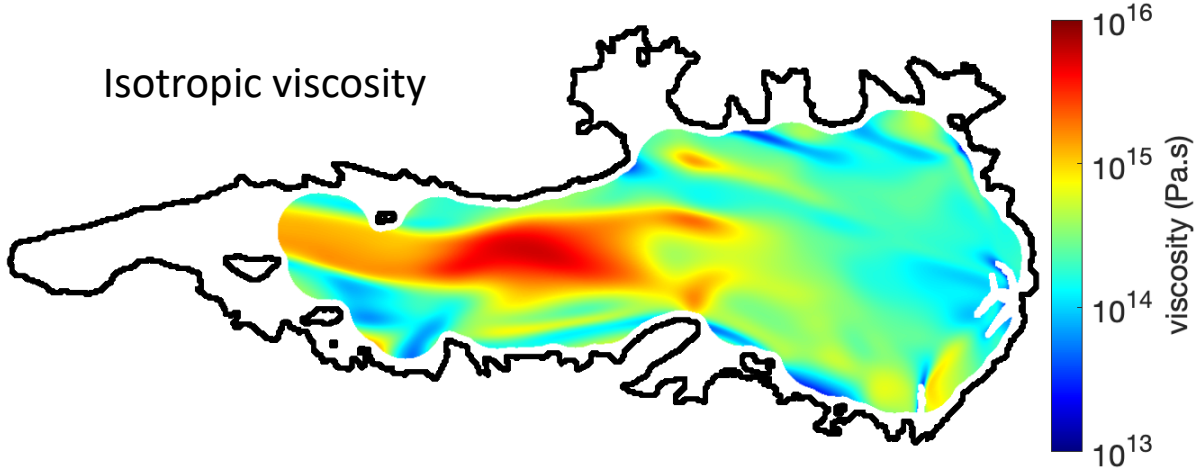
$$f_2 = \frac{\partial}{\partial y}\left(2\mu_h h\frac{\partial v}{\partial y} + 2\mu_v h\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]\right) + \frac{\partial}{\partial x}\left(\mu_h h\left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right]\right) - \rho_i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

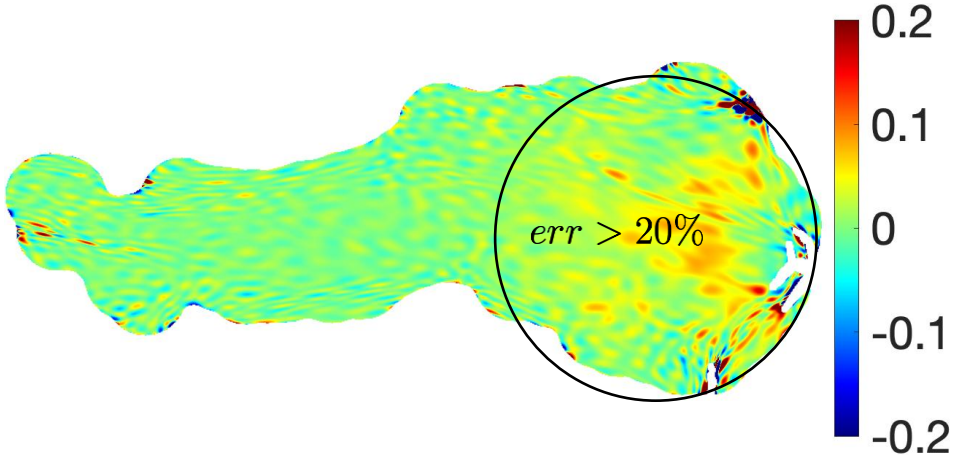


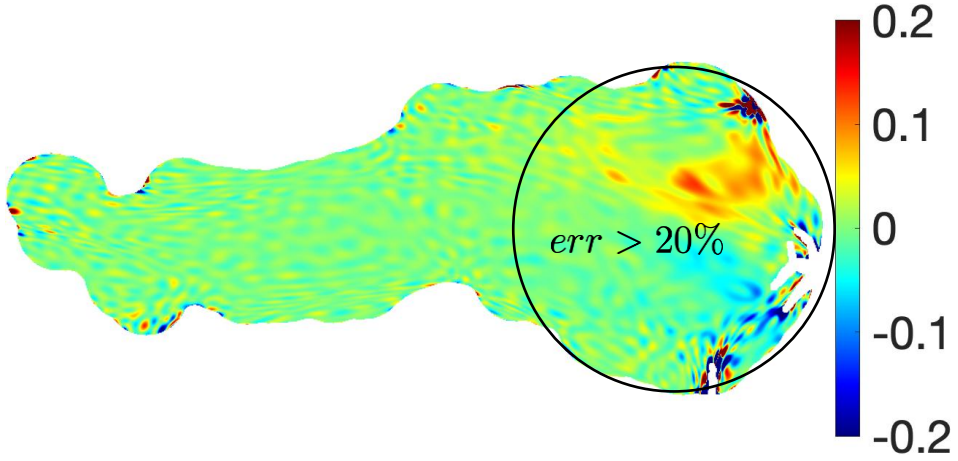
Isotopic viscosity of Amery

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

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



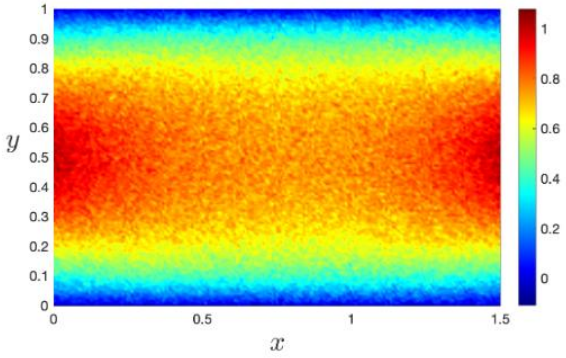




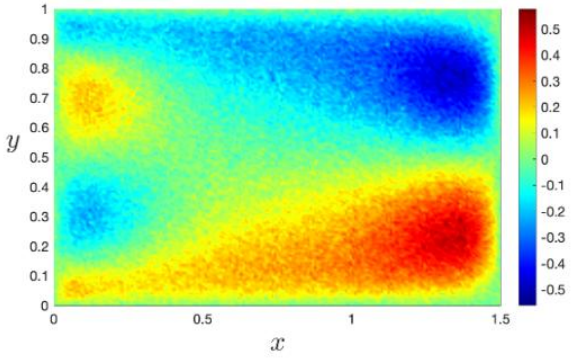
PINN: Stability against data noise

PINN inversion using training data with **noise (20%)**

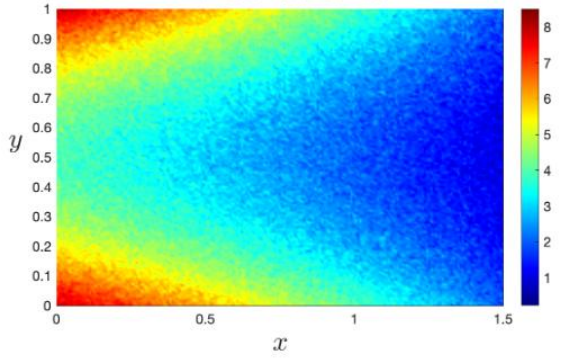
noise data $\tilde{u}$



noise data $\tilde{v}$



noise data $\tilde{h}$



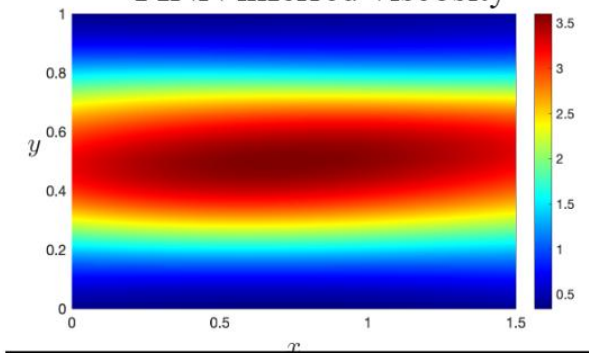
$$\tilde{q} = q(1 + 0.2X)$$

q represents u, v, h

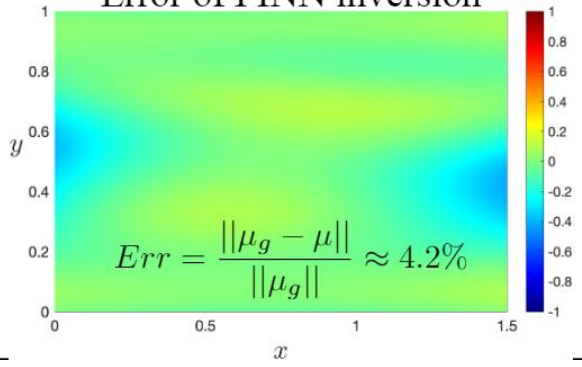
random variable $X \sim N(0, 1)$

$$Err = \frac{||\mu_g - \mu||}{||\mu_g||} \approx 63\%$$

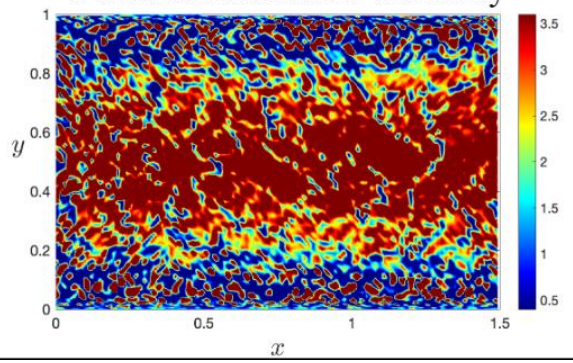
PINN inferred viscosity



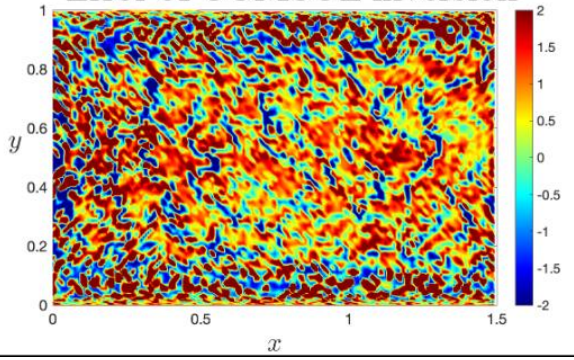
Error of PINN inversion



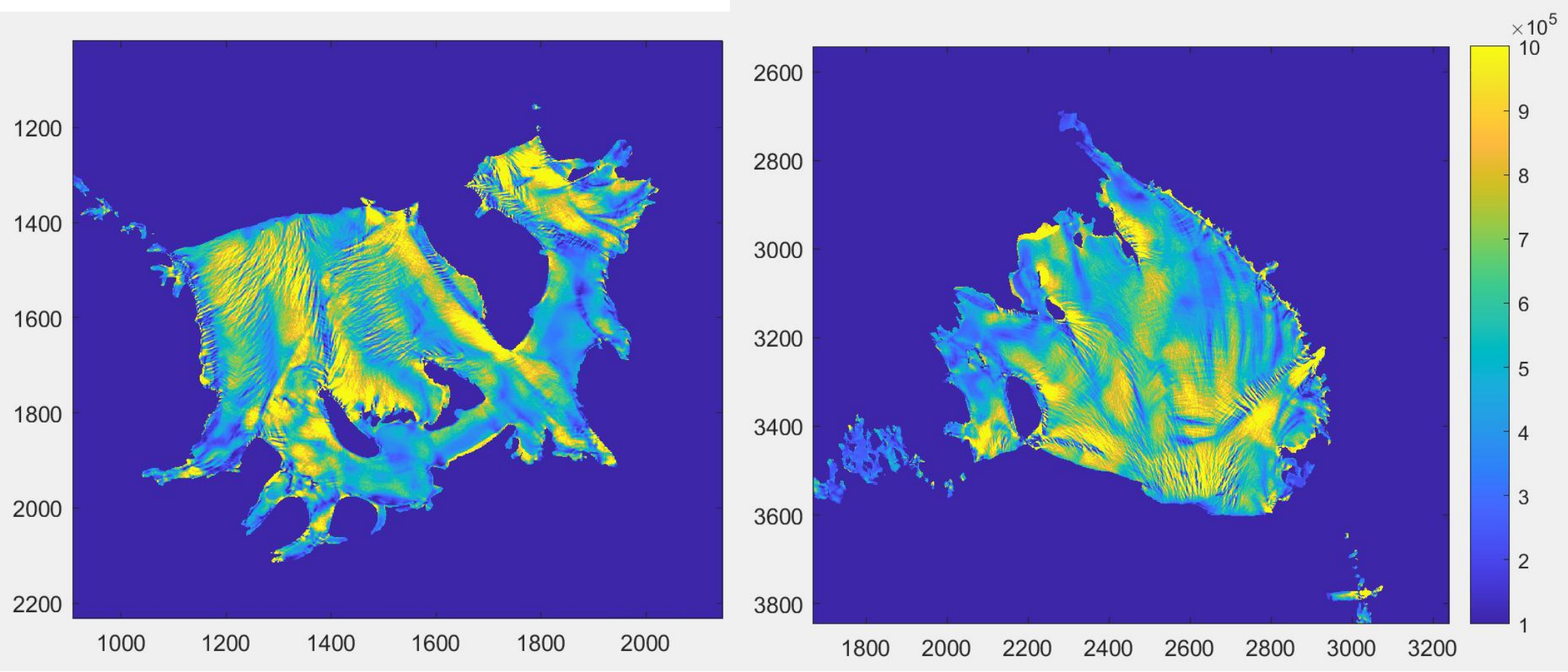
COMSOL inferred viscosity



Error of COMSOL inversion



B in Furst et al (2023)



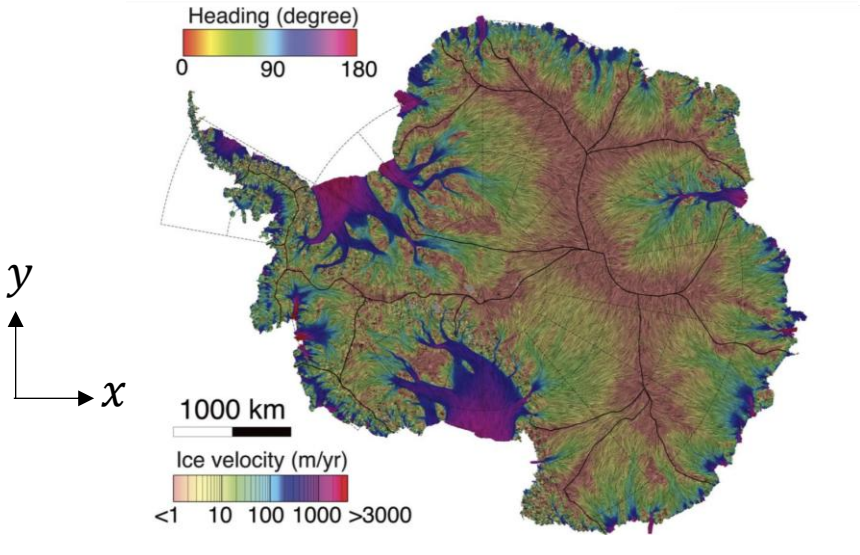
Project

Ice shelves: inversion of viscosity

Data:

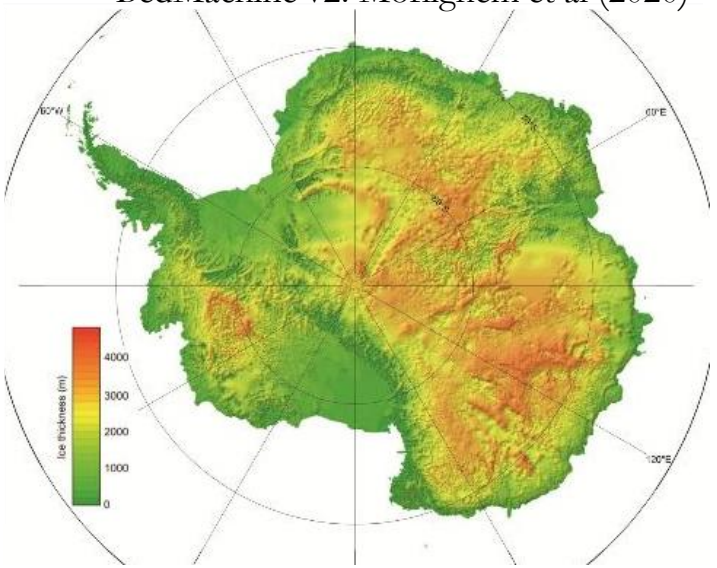
Ice velocity u

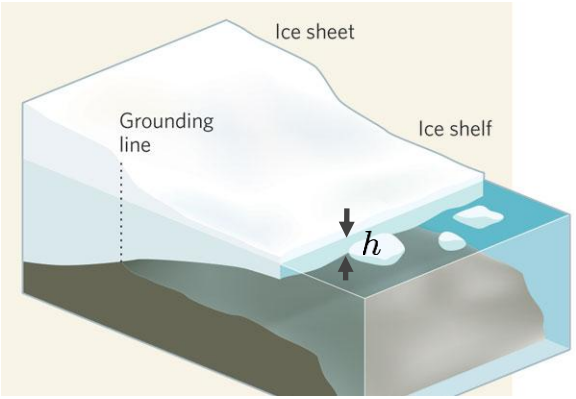
MEaSUREs v1: Mouginot et al (2019)



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

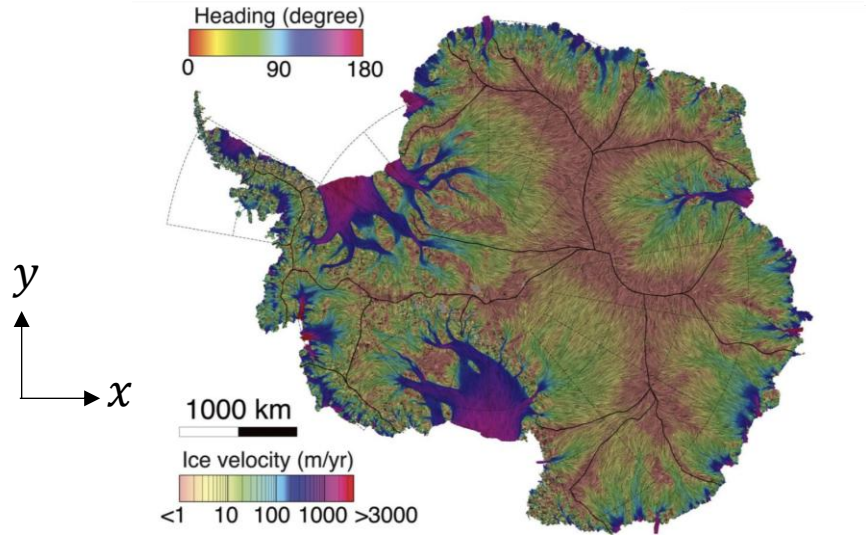
} measurable

Ice sheet: inversion of both viscosity and basal traction

Data:

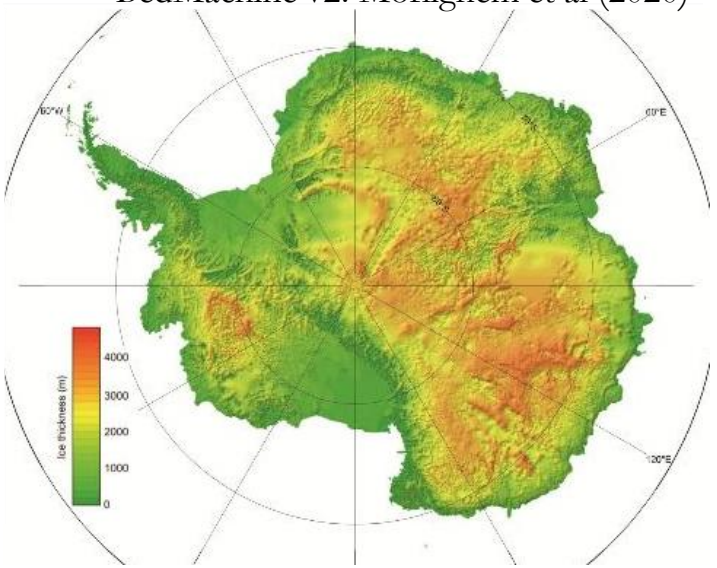
Ice velocity u

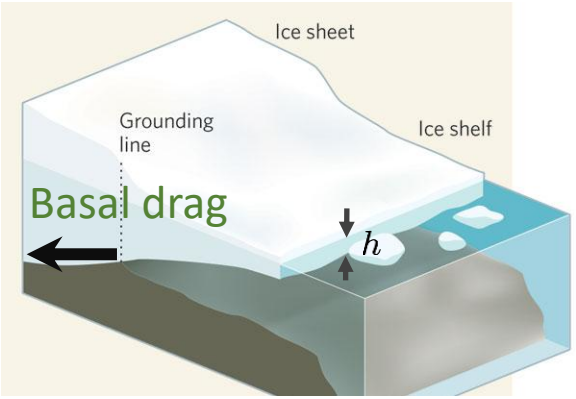
MEaSUREs v1: Mouginot et al (2019)



Ice thickness h

BedMachine v2: Morlighem et al (2020)





Equation:

Ice effective viscosity: impossible to measure

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

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

Basal traction: impossible to measure

Morland (1987), MacAyeal (1989)

u, v: horizontal velocity

ρ_i: ice density

h : ice thickness

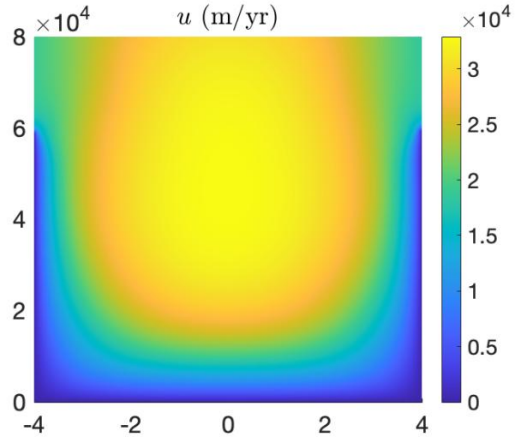
s : surface elevation

} measurable

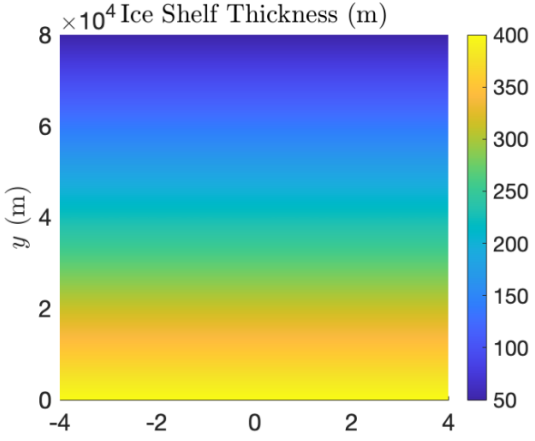
Ice sheet: inversion of both viscosity and basal traction

Data:

Ice velocity u



Ice thickness h

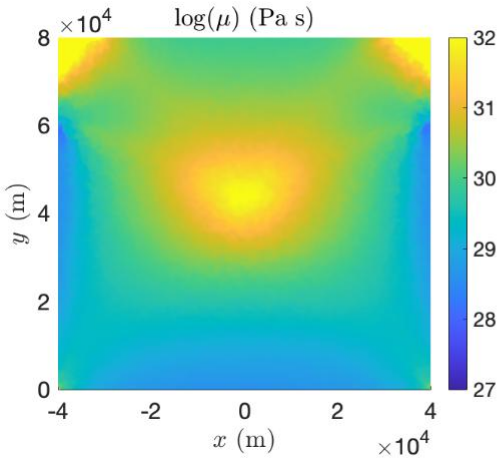


y

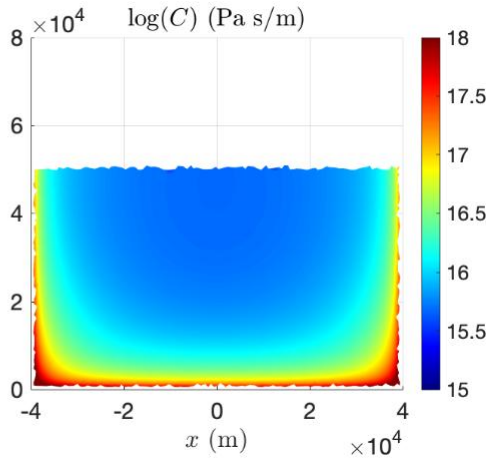
x

Predict (infer):

Ice viscosity μ



Basal traction



Equation:

Ice effective viscosity: impossible to measure

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

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

Basal traction: impossible to measure

Morland (1987), MacAyeal (1989)

u, v: horizontal velocity

ρ_i: ice density

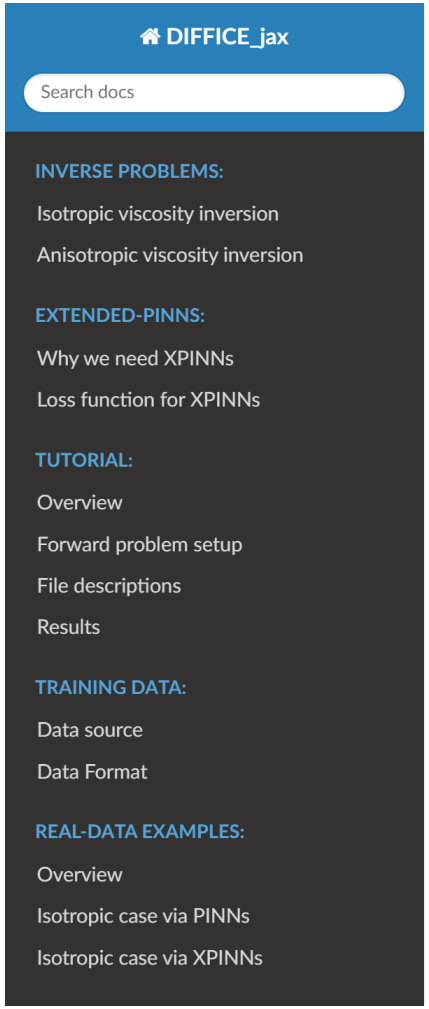
h : ice thickness

s : surface elevation

} measurable

DIFFICE.jax v1.0:

DIFFerentiable neural-network solver for data assimilation of ICE shelves written in JAX.

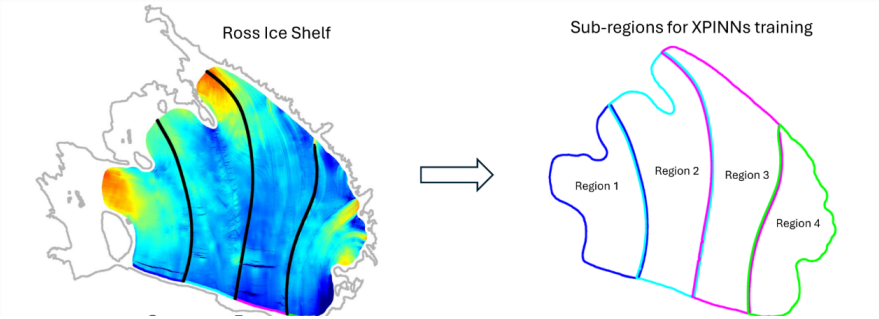


DIFFICE_jax documentation

DIFFICE_jax documentation

Introduction

`DIFFICE_jax` is a Python package that solves the depth-integrated Stokes equation for **ice shelves**, and can be adopted for **ice sheets** by modifying the partial differential equations (PDE) in the neural network loss function. It uses PDEs to interpolate descretized remote-sensing data into meshless and differentible functions, and infer ice shelves' viscosity structure via back-propagation and automatic differentiation (AD). The algorithm is based on physics-informed neural networks (PINNs) and implemented in JAX. The `DIFFICE_jax` package involves several advanced features in addition to vanilla PINNs algorithms, including collocation points resampling, non-dimensionalization of the data adnd equations, extended-PINNs (XPINNs) (see figure below), viscosity exponential scaling function, which are essential for accurate inversion. The package is designed to be user-friendly and accessible for beginners. The Github respository also provides **Colab Notebooks** for both the synthetic and real ice-shelf examples in the `tutorial` and `examples` folders, respectively.



2D SSA: https://github.com/YaoGroup/DIFFICE_jax
- Synthetic case
- Real data case

Package can be installed via
`python -m pip install DIFFICE_jax`