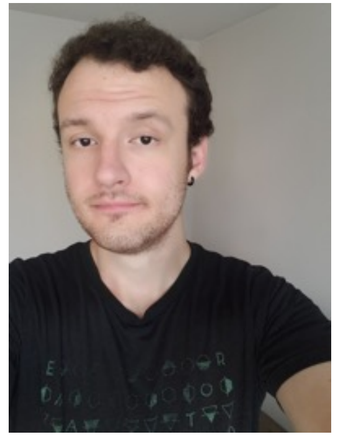
Potentials (promises?) of Physics-Informed Deep Learning in Earthquake Seismology

Brittany A. Erickson[1,2] and Cody Rucker[1]

1. Department of Computer Science
2. Department of Earth Sciences
University of Oregon

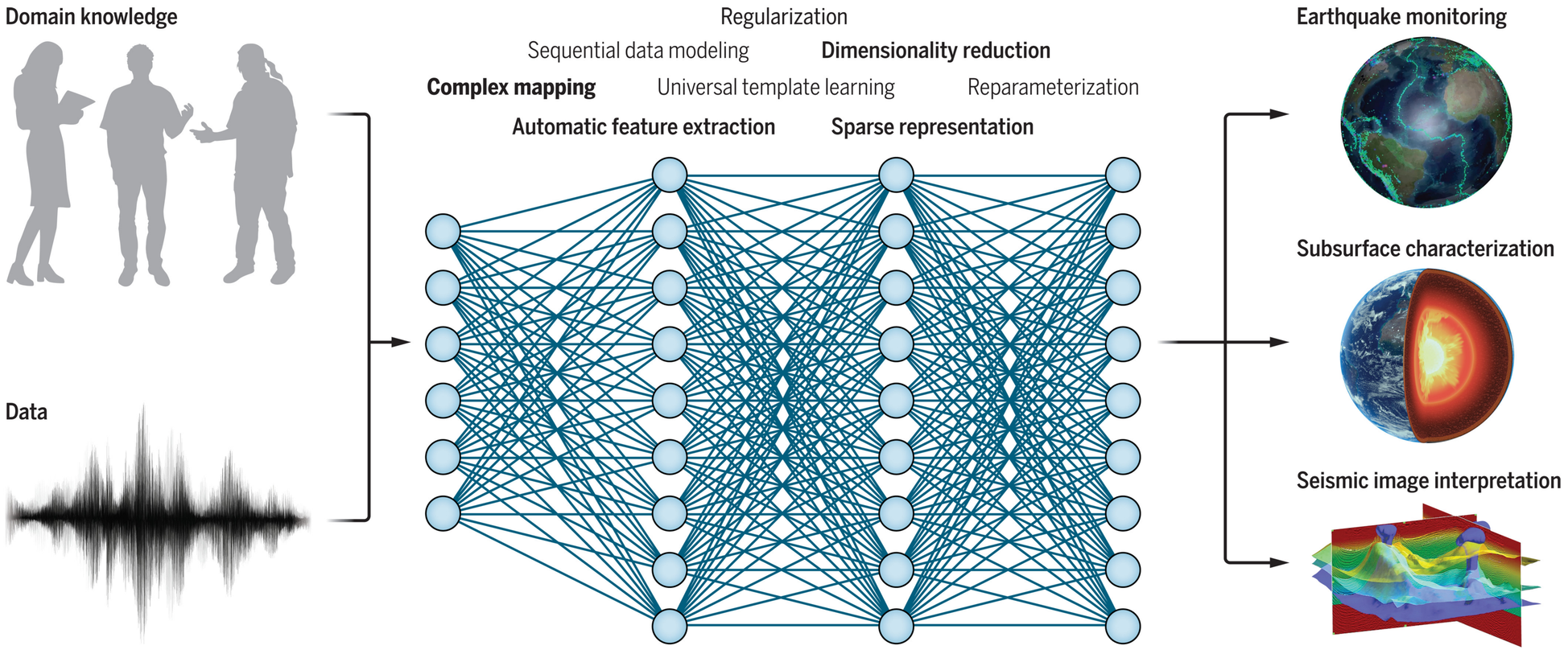


Statewide California Earthquake Center Annual Meeting 2024

Overview of Talk

- The emergence of physics-informed deep learning
- Recent advances and applications with PiNNs
- Inversions with rate-and-state friction
- Advantages, limitations (a lot) and theoretical needs
- Future directions and areas ripe for research

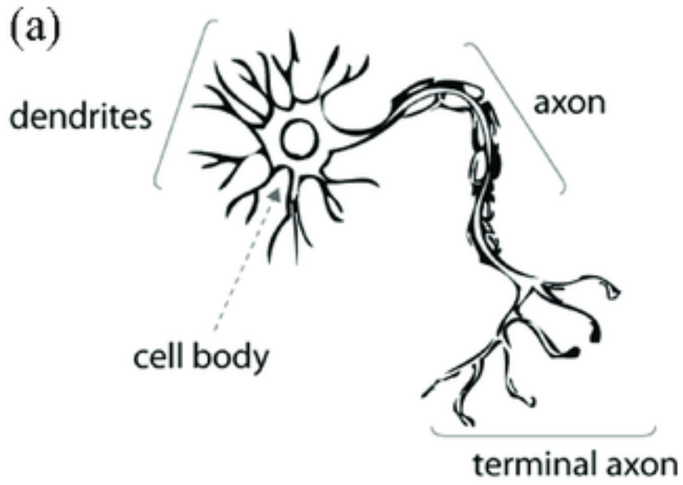
Explosion of Deep-Learning in Seismology

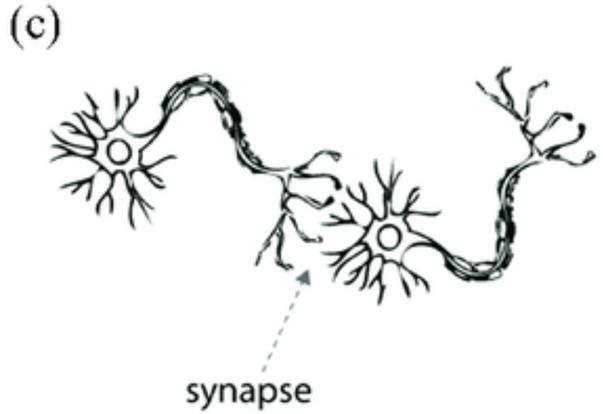


Mousavi and Beroza (*Science*, '22)

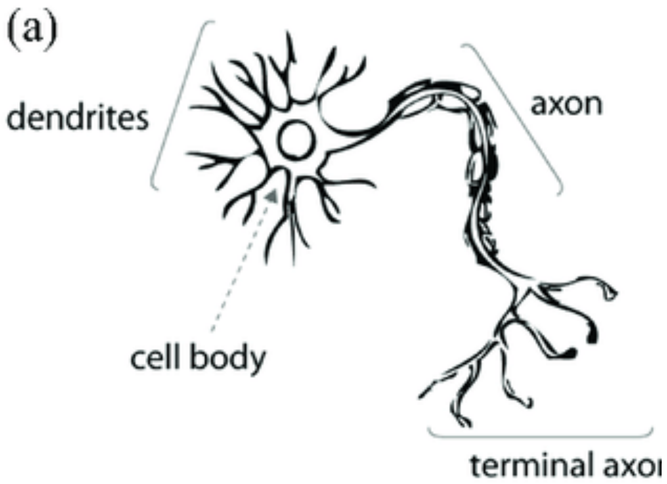
ML excels in presence of large data and neural-network (NN) common. Deep NN developed as new (or alternative method) to learn complex patterns and relationships, understand multiscale features, possibly better/faster.

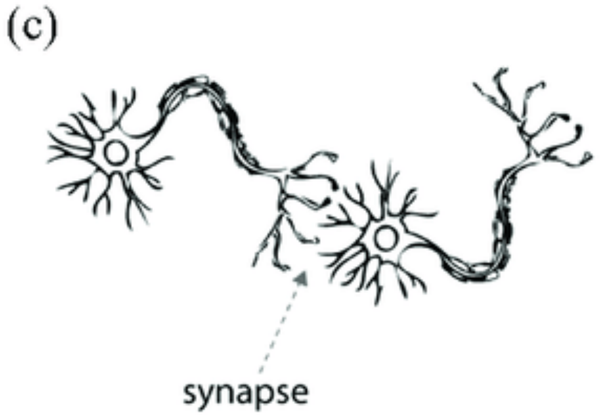
Historically: artificial neural networks (ANNs) developed to mimic **neurons** in the human **brain,** which were understood to perform their functions through a massive inter-connected **network** known as synapses.



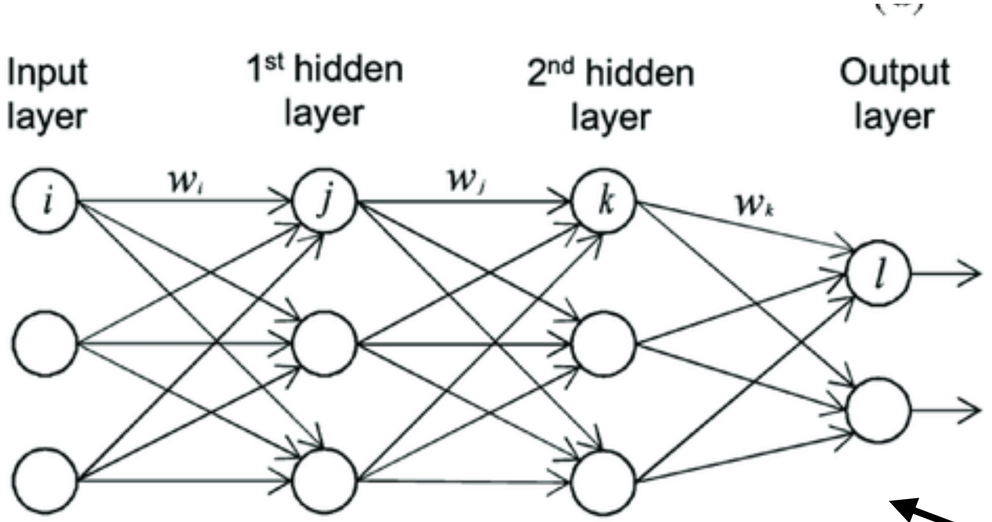


Historically: artificial neural networks (ANNs) developed to mimic **neurons** in the human **brain,** which were understood to perform their functions through a massive inter-connected **network** known as synapses.





Trained to learn from data and experience:



LOSS

Surrogate model of bio/physical system

Network is trained so that outputs are "close" to known data, i.e. we minimize a loss function based on some measure of error.

Science Drivers:

- “What is an earthquake?” — NASEM. 2020.

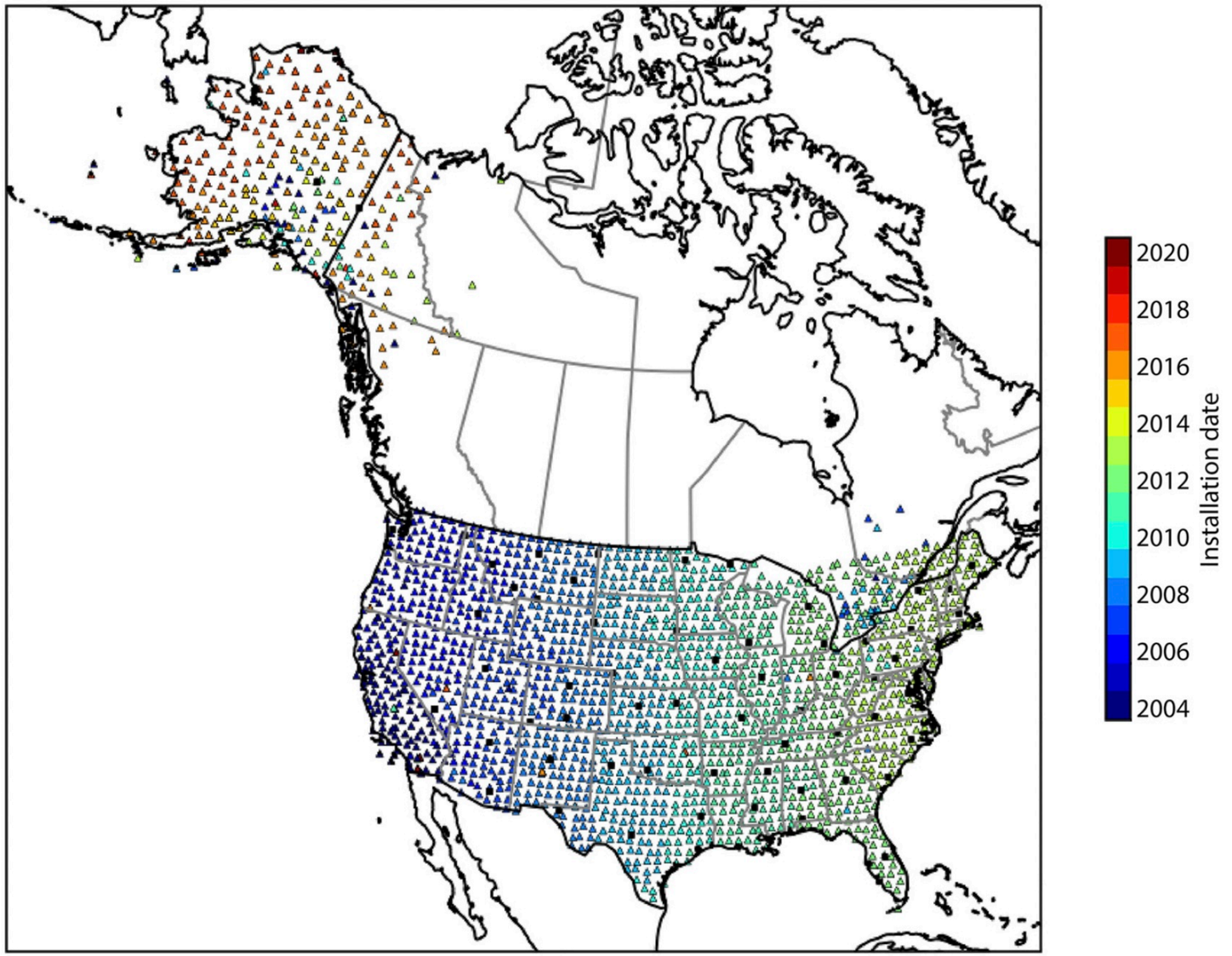
- Improve the predictive analyses of seismicity and ground motion
    - Make progress by combining physics-based, statistical, and machine learning — SCEC 2024 Research Plan

Persistent Challenges:

Direct observations of Earth’s interior largely inaccessible, great EQ infrequent.

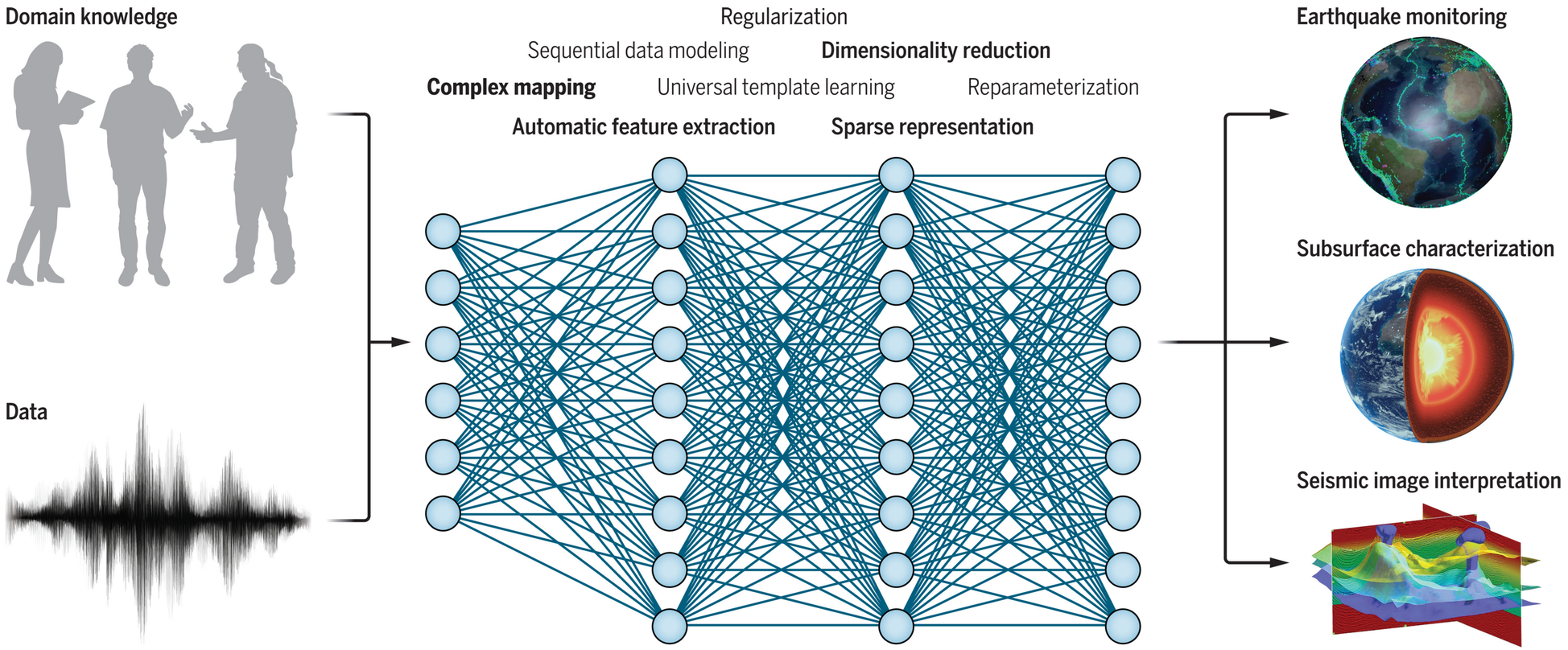
“Discoveries increasingly will come from the analysis of large datasets, new developments in inverse theory, and procedures enabled by computationally intensive simulations” — Bergen et al. (*Science*, 2019)

Sensors and permanent stations across the US:



Arrowsmith et al. (Rev. Geophys., ’22)

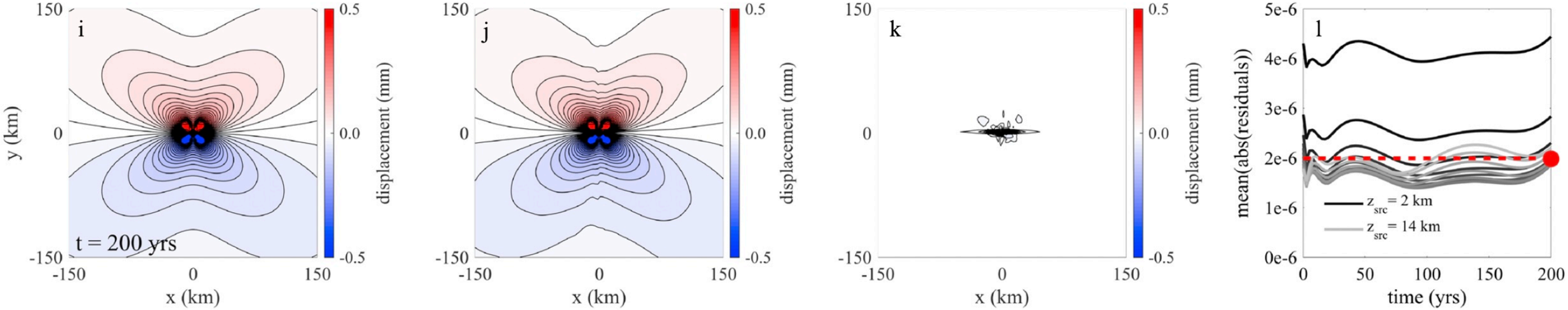
Deep-Learning in Seismology



Mousavi and Beroza (*Science*, '22)

Basic NN don't impose physics. In cases where system is only data driven, loss function usually defined via the difference between NN outputs and target data. How to integrate physics and could this supplement limited data?

Recent efforts to Integrate Physics into NN



Devries et al. (*GRL*, '17)

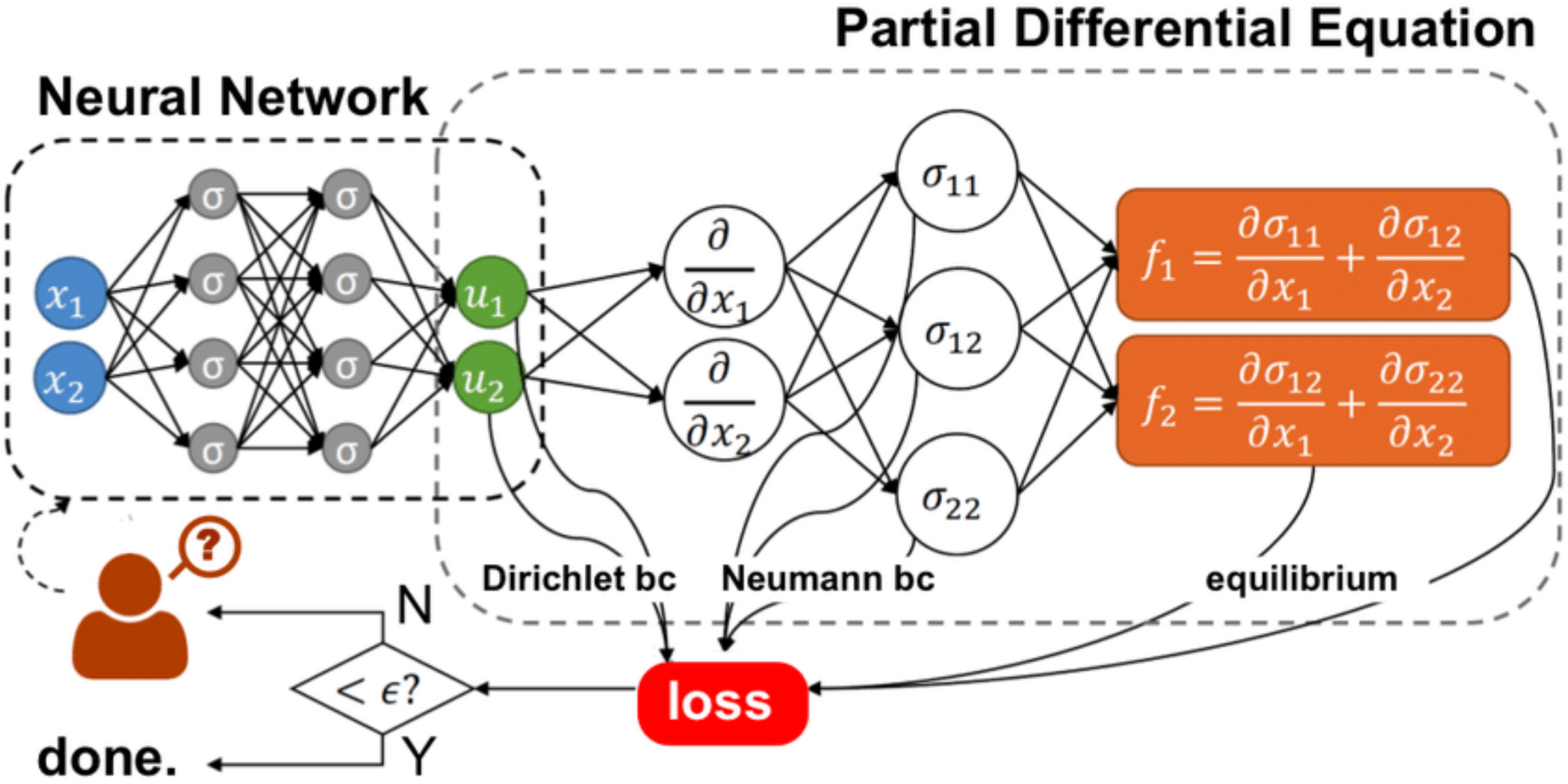
- **Option 1: train network on outputs from a classical method or traditional numerical method (e.g. finite difference), expensive training. Example of physics-guided neural networks (PgNNs).**

- **FOCUS OF THIS TALK: physics-informed neural networks (PiNNs) - "data" is simply collocation points!**

Other physics-based DL methods:
- **physics-encoded neural networks (PeNNs)**
- **DeepONets**

Physics-informed Neural Networks (PiNNs)

Arose out of successes of DNN, which are purely data-driven, desire to derive a surrogate model whose outputs are constrained by known physics but with limited data and no need for alternative forward methods.



Feed-forward Neural Networks (FFNN) with L hidden layers

Let $\mathbf{x} \in \mathbb{R}^n$ be the network input. Then

$$\ell_0 = \mathbf{x},$$
$$\ell_k = \varphi_k(W_k \ell_{k-1} + b_k) \quad \text{for } 0 < k < L,$$

defines a feed-forward, deep neural network $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^d$ (of depth L) by

$$\mathcal{N}(\mathbf{x}; \theta) = W_L \ell_{L-1} + b_L.$$

Here $\{\varphi_i\}_{i=1}^L$ is a collection of activation functions along with a sequence of trainable network parameters (weights and biases) $\{\theta_i\}_{i=0}^L$ where $\theta_k = (W_k, b_k)$ for each $0 \leq k \leq L$.

The PiNN then extends the FFNN by considering physical governing laws described by a parametric partial differential equation (PDE) of the form:

$$\mathscr{L}[u;\lambda] = 0, \quad x \in \Omega, \quad t \in \mathscr{T},$$

where $\mathscr{L}$ is a nonlinear differential operator with coefficients λ,

e.g. acoustic wave equation, $\mathscr{L} = \dfrac{\partial^2}{\partial t^2} - c^2\Delta$, $\lambda = c$, where c is the wave speed.

The PiNN then extends the FFNN by considering physical governing laws described by a parametric partial differential equation (PDE) of the form:

$$\mathscr{L}[u;\lambda] = 0, \quad x \in \Omega, \quad t \in \mathscr{T},$$

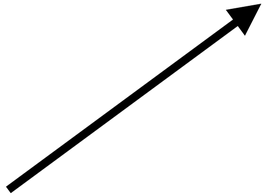
where $\mathscr{L}$ is a nonlinear differential operator with coefficients λ,

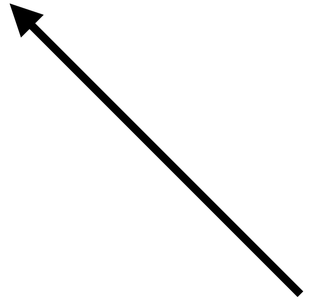
e.g. acoustic wave equation, $\mathscr{L} = \dfrac{\partial}{\partial t} - c^2\Delta$, $\lambda = c$, where c is the wave speed.

tl;dr: the cost/loss function is then defined (at least partially) by $\mathscr{L}[\mathscr{N};\lambda]$. Note here that we have assumed $\mathscr{N} \approx u$.

The general representation of the cost function is given by

$$C = MSE_u + MSE_f$$





Mean-squared error at known data points

PDE residual on a set of collocation points within the domain

Data could be observables or known boundary data (or both).

PDEs in two contexts

Forward problems **vs.** **Inverse problems**

Data-driven inference (assume λ are known, need to find u).

Data-driven identification of PDEs (given some info on u, try to learn λ).

PDEs in two contexts

Forward problems **vs.** **Inverse problems**

Data-driven inference (assume λ are known, need to find u).

Data-driven identification of PDEs (given some info on u, try to learn λ).

"Traditional" methods (Finite Difference etc.), might include data but cannot over-specify.

Tons of work in this area for DECADES… high-order accurate, provably convergent, massively parallel linear algebra computations (e.g. $Ax = b$).

PDEs in two contexts

Forward problems vs. **Inverse problems**

Data-driven inference (assume λ are known, need to find u).

Data-driven identification of PDEs (given some info on u, try to learn λ).

PiNNs - can include as much data as desired, solve different types of problems!

"Traditional" methods (Finite Difference etc.), might include data but cannot over-specify.

Tons of work in this area for DECADES… high-order accurate, provably convergent, massively parallel linear algebra computations (e.g. $Ax = b$).

PDEs in two contexts

Forward problems vs. **Inverse problems**

Data-driven inference (assume λ are known, need to find u).

PiNNs - can include as much data as desired, solve different types of problems!

"Traditional" methods (Finite Difference etc.) for IBVP might include data but cannot over-specify.

Data-driven identification of PDEs (given some info on u, try to learn λ).

"Traditional" methods (Bayesian etc.), require many forward runs

Lots of work here too

Tons of work in this area for DECADES… high-order accurate, provably convergent, massively parallel linear algebra computations (e.g. $Ax = b$).

PDEs in two contexts

Forward problems vs. **Inverse problems**

Data-driven inference (assume λ are known, need to find u).

PiNNs - can include as much data as desired, solve different types of problems!

"Traditional" methods (Finite Difference etc.) for IBVP might include data but cannot over-specify.

Data-driven identification of PDEs (given some info on u, try to learn λ).

"Traditional" methods (Bayesian etc.), require many forward runs

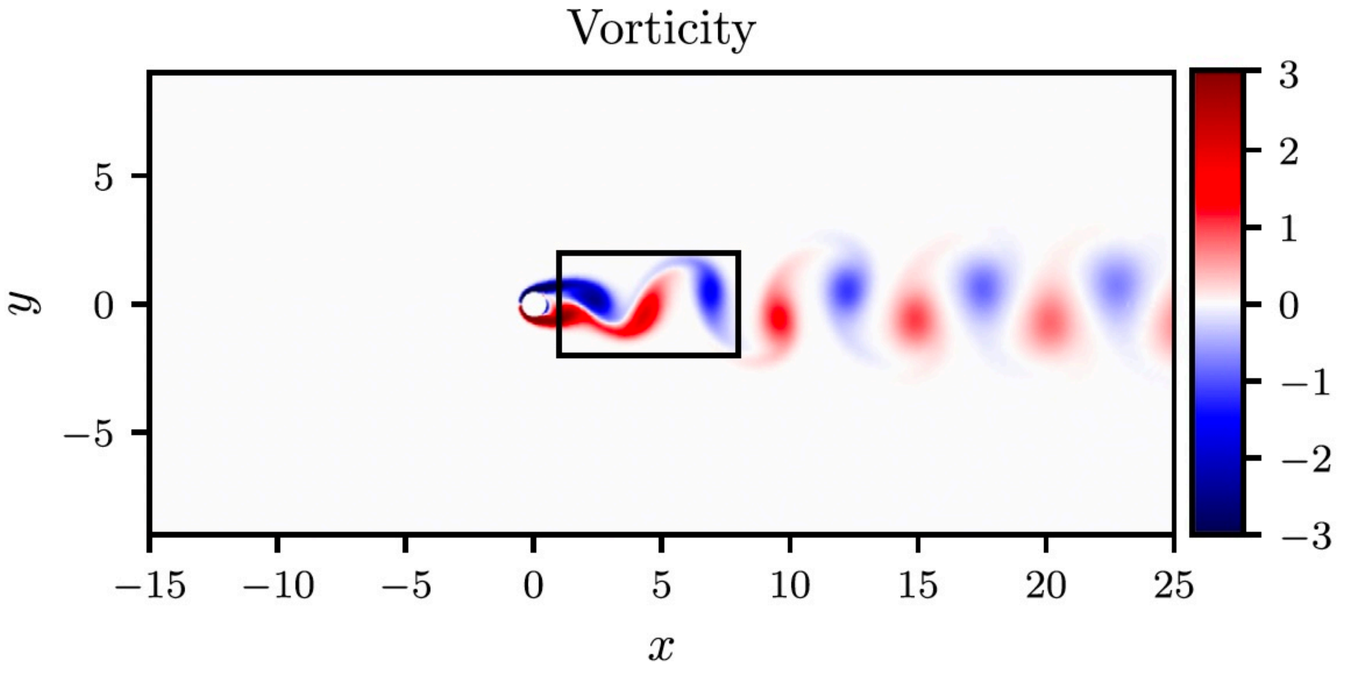
PiNNs, code only slightly modified from forward problem

Lots of work here too

Tons of work in this area for DECADES… high-order accurate, provably convergent, massively parallel linear algebra computations (e.g. $Ax = b$).

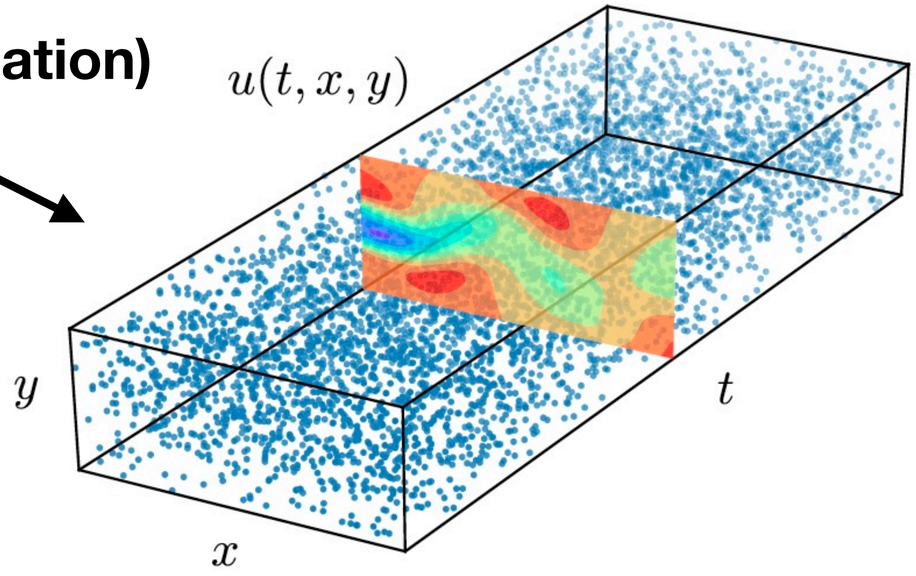
PiNNs offer both alternative solution strategies AND methods for addressing new problems.

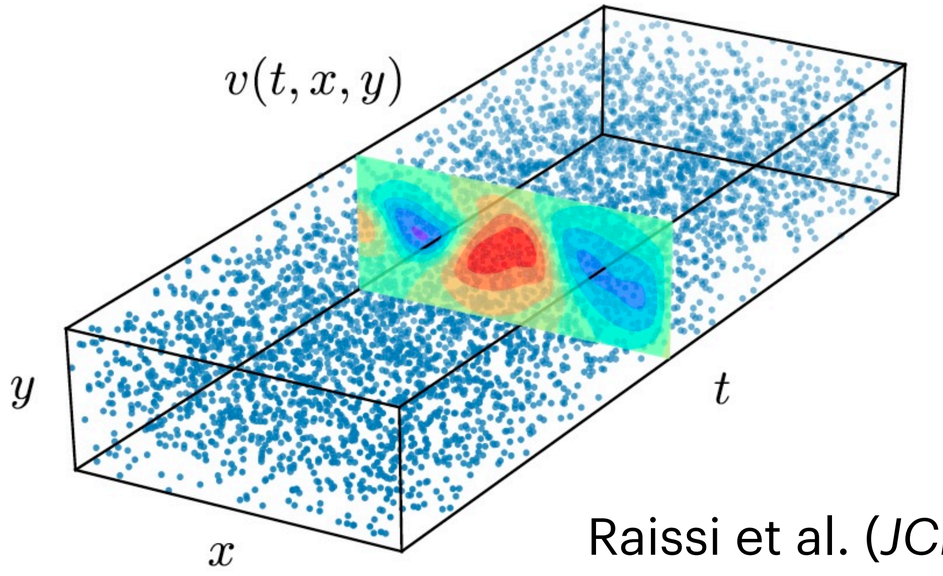
Fundamentals of PiNNs attributed to 2018 seminal paper for both forward and inverse problems.



incompressible flow and dynamic vortex shedding

Training (collocation) data-points

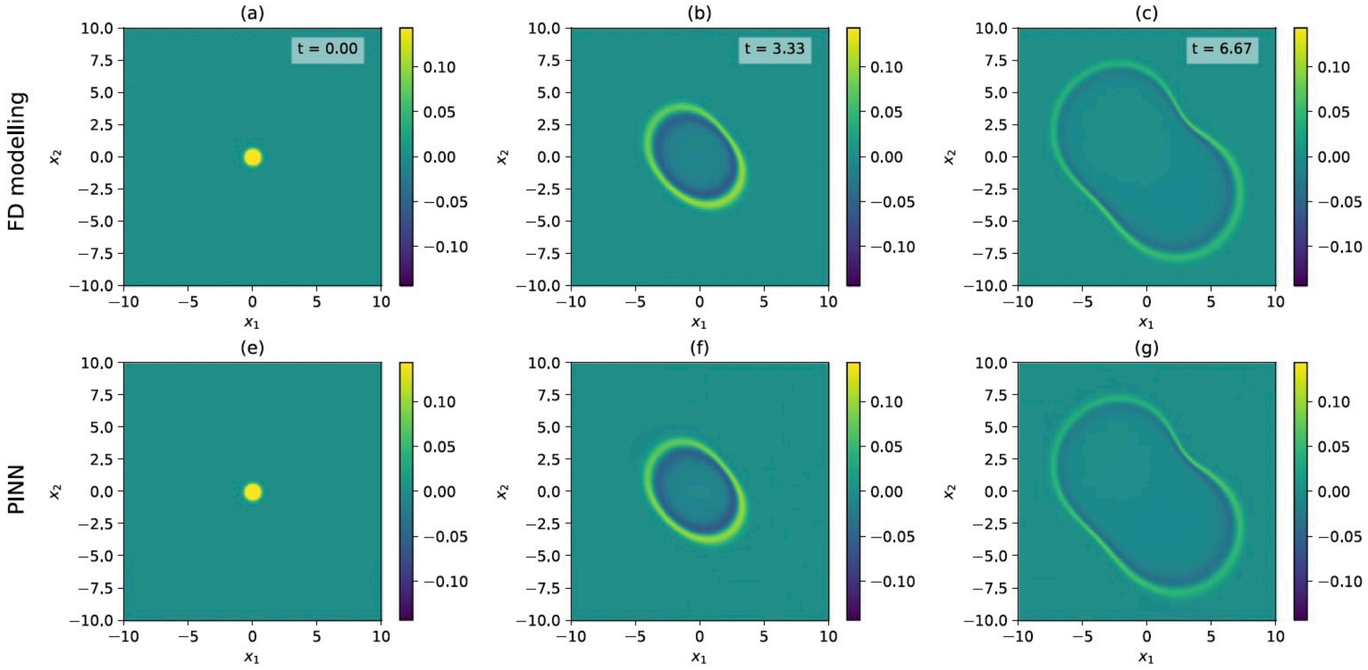




Raissi et al. (*JCP*, '18)

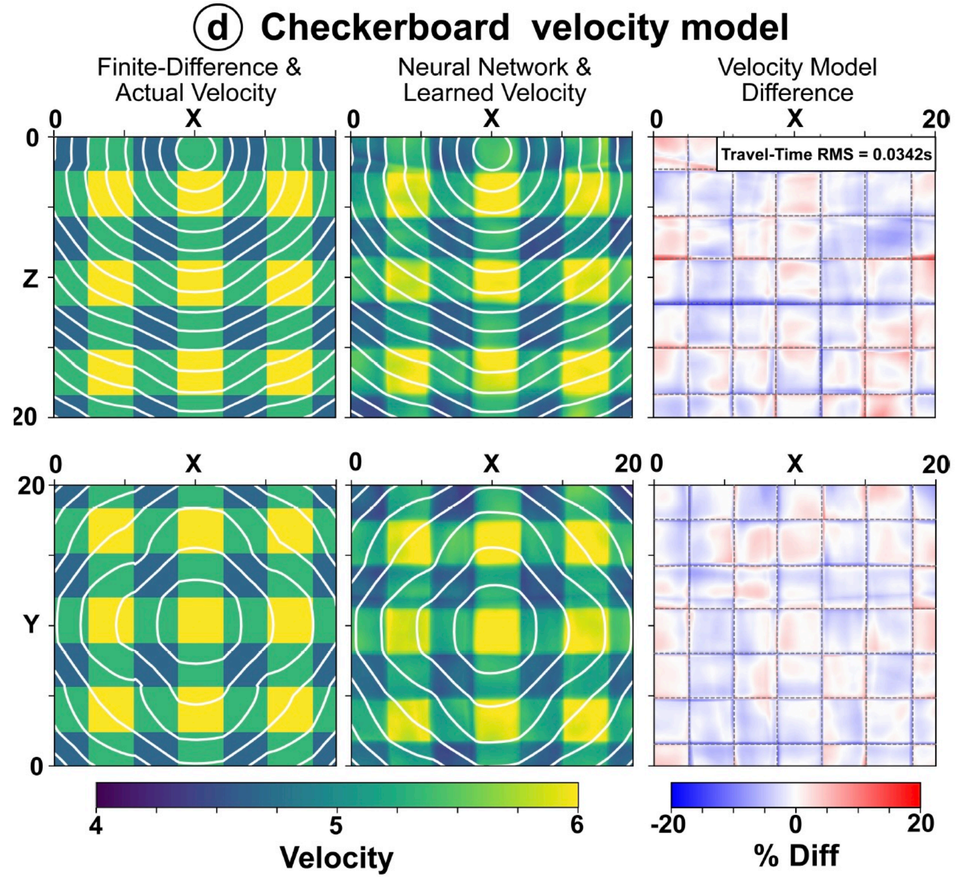
PiNNs for forward problems in seismology applications

Some recent advances/applications:



Moseley et al. (ACM, ’23)

(FBPINNs) shown to be effective in solving multi-scale problems.

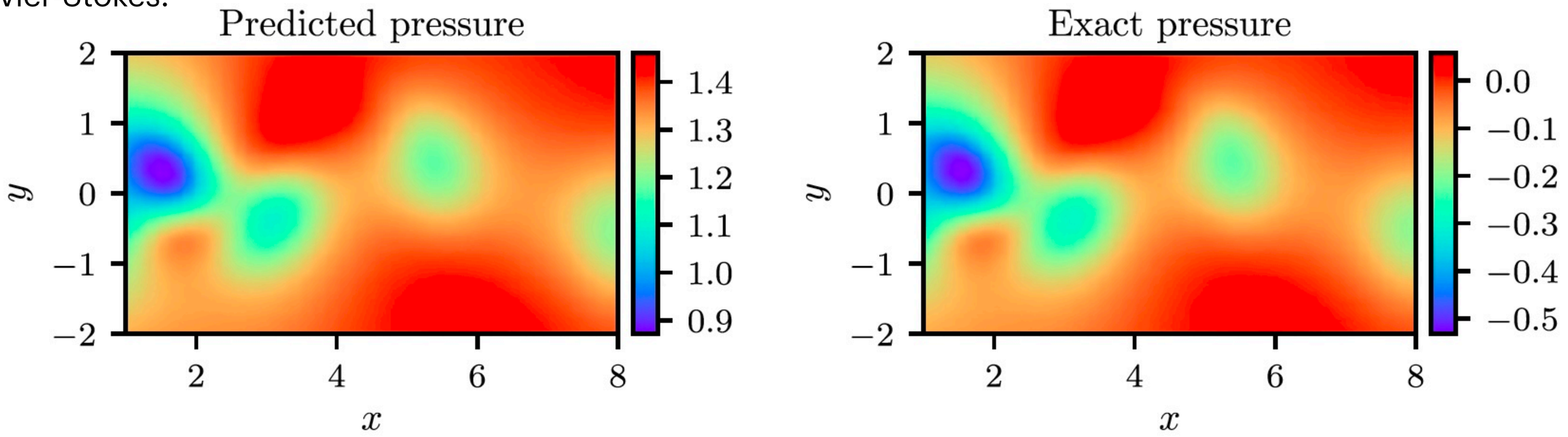


Smith et al. (IEEE, ’21)

Eikonal equation which allows for rapid determination of the travel time between any two points.

Raissi et al. (2018) highlights PiNNs for inverse problems

Navier-Stokes:

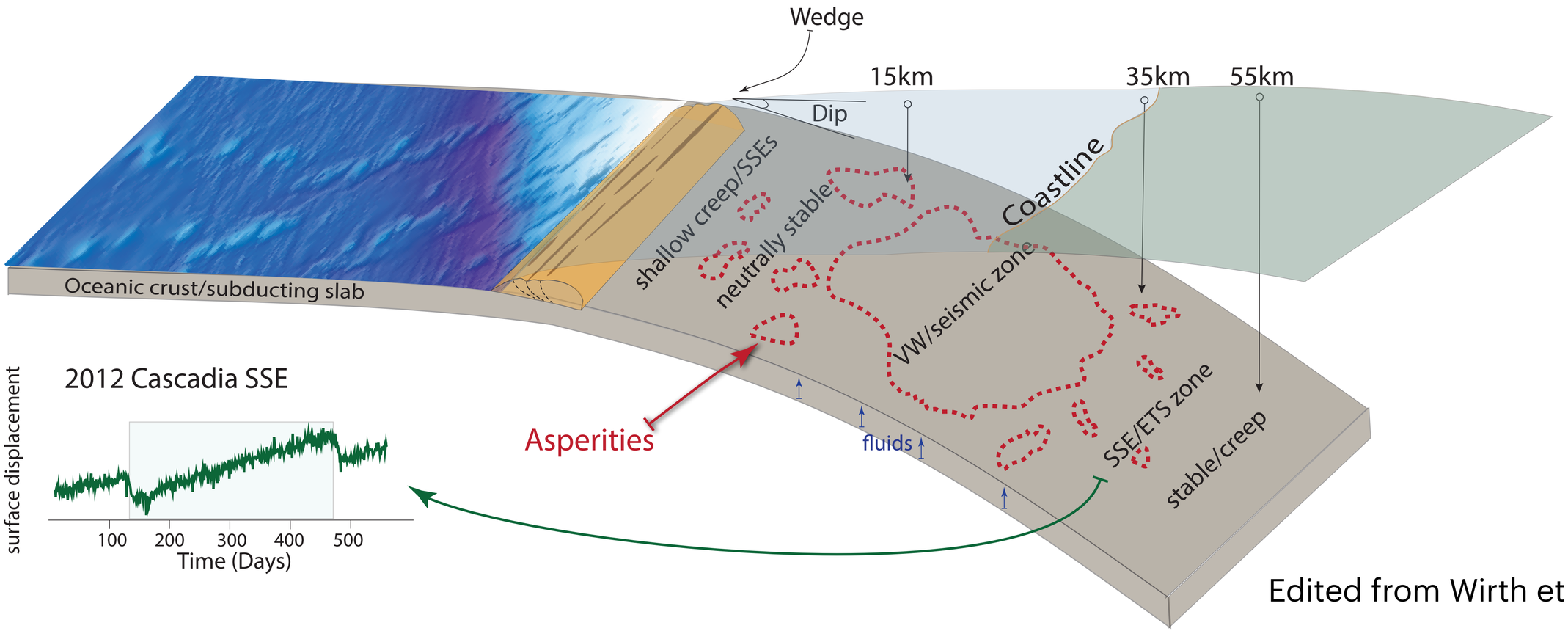


Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t +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

Recovered both solution and coefficients of governing equation

Raissi et al. (*JCP*, '18)

Using PiNNs for inverse problems: inferring depth-dependency of frictional parameters.



Edited from Wirth et al. ('22)

Slip thought to correlate with depth and the distribution of asperities along the fault. Understanding the physical mechanisms for diversity of slip styles (from slow slip to dynamic rupture) important for hazard mitigation, but frictional properties largely unknown.

Model set-up: posed as an IBVP (PDE + BC + IC):

$$\mathscr{L}\left[\mathbf{u};\lambda\right] = \mathbf{k}, \qquad \text{in } \Omega,$$

$$\mathscr{B}\left[\mathbf{u};\lambda\right] = \mathbf{g}, \qquad \text{on } \partial\Omega$$

where $\mathscr{L}, \mathscr{B}$ are differential and boundary operators parameterized by λ, $\mathbf{k}$ and $\mathbf{g}$ are known source terms.

Model set-up: posed as an IBVP (PDE + BC + IC):

$$\mathscr{L}\left[\mathbf{u};\lambda\right] = \mathbf{k}, \qquad \text{in } \Omega,$$

$$\mathscr{B}\left[\mathbf{u};\lambda\right] = \mathbf{g}, \qquad \text{on } \partial\Omega$$

where $\mathscr{L}, \mathscr{B}$ are differential and boundary operators parameterized by λ, $\mathbf{k}$ and $\mathbf{g}$ are known source terms.

E.g. 1D wave equation with displacement BC and wave speed c:

$$u_{tt} = c^2 u_{xx} + k(x,t), \quad x \in [0,1], \quad t \geq 0$$

$$u(0,t) = g_0(t)$$

$$u(1,t) = g_1(t)$$

Then $\mathscr{L} = \dfrac{\partial^2}{\partial t^2} - c^2 \dfrac{\partial^2}{\partial x^2}$ and $\mathscr{B}$ is the identity operator.

Model set-up: Rate-and-State Friction (RSF)

Fault interface condition:

$$\tau = \bar{\sigma}_n \left[f_0 + a \ln\left(\frac{V}{V_0}\right) + b \ln\left(\frac{V_0 \psi}{D_c}\right) \right],$$

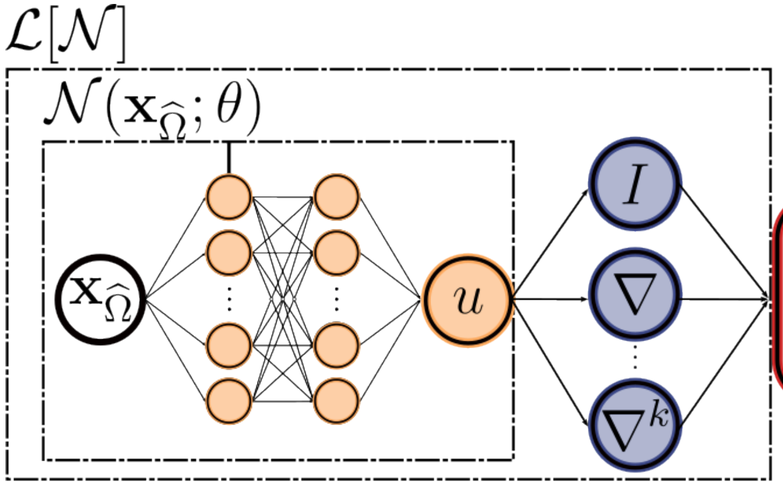
State variable ψ evolves according to

$$\frac{d\psi}{dt} = G(V, \psi)$$
$$\psi(0) = \psi_0$$

- **τ is shear stress**
- **V is slip rate**
- **$\bar{\sigma}_n$ the effective normal stress**
- **f_0 is a reference friction coefficient for sliding at speed V_0**
- **D_c is the characteristic slip distance**
- **a and b are material parameters**

(Dieterich, 1979; Ruina, 1983; Marone, 1998).

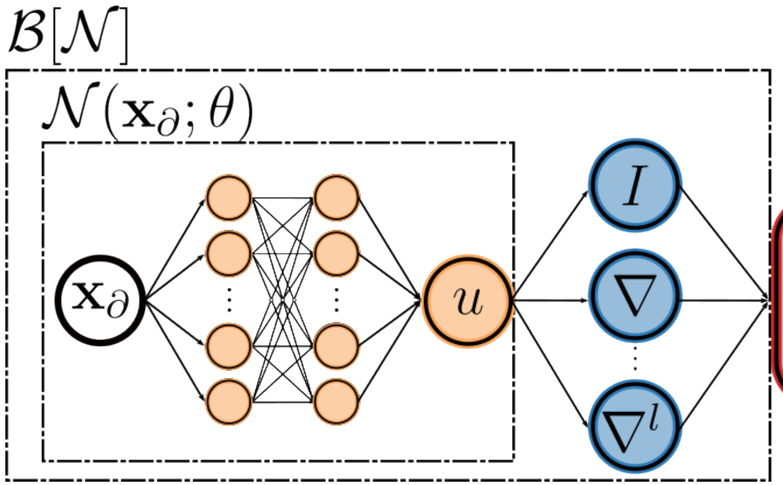
General PiNN architecture



$$MSE_{\widehat{\Omega}} = \frac{1}{N_{\widehat{\Omega}}} \sum_{i=1}^{N_{\widehat{\Omega}}} \left| \mathcal{L}[\mathcal{N};\lambda](\mathbf{x}^i_{\widehat{\Omega}};\theta) - \mathbf{k}^i \right|^2$$

Cost function: (mean-squared-error, MSE); could also include data.

$$MSE(\theta) = MSE_{\widehat{\Omega}} + MSE_{\partial}$$

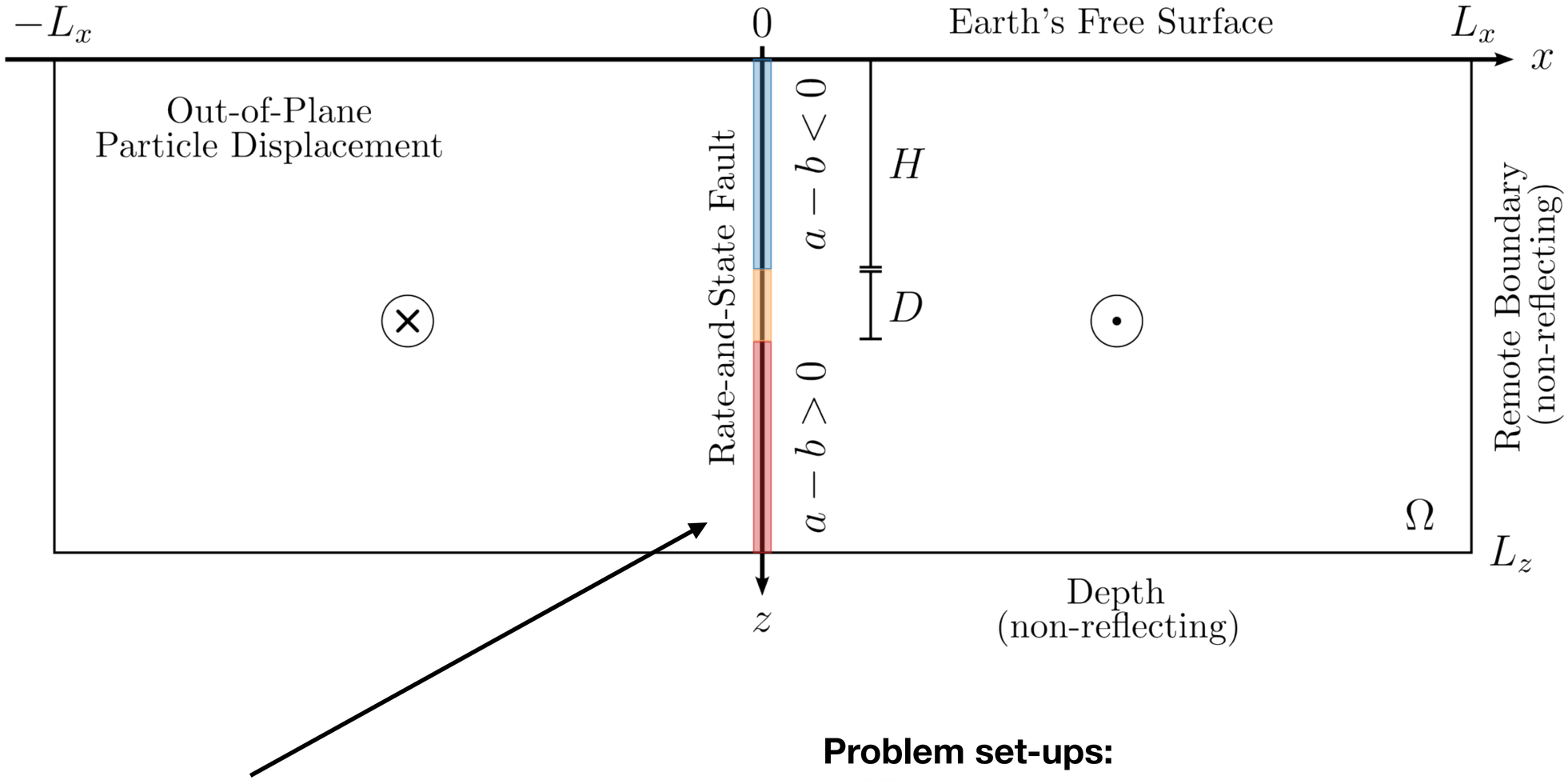


$$MSE_{\partial} = \frac{1}{N_{\partial}} \sum_{i=1}^{N_{\partial}} \left| \mathcal{B}[\mathcal{N};\lambda](\mathbf{x}^i_{\partial};\theta) - \mathbf{g}^i \right|^2$$

Rucker and Erickson (CMAME, '24)

Assume $\mathbf{u}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x};\theta)$ and consider $N_{\widehat{\Omega}}$ collocation points chosen in volume and N_{∂} on the boundaries. An additional connected network computes state evolution loss.

Inversions with rate-and-state friction

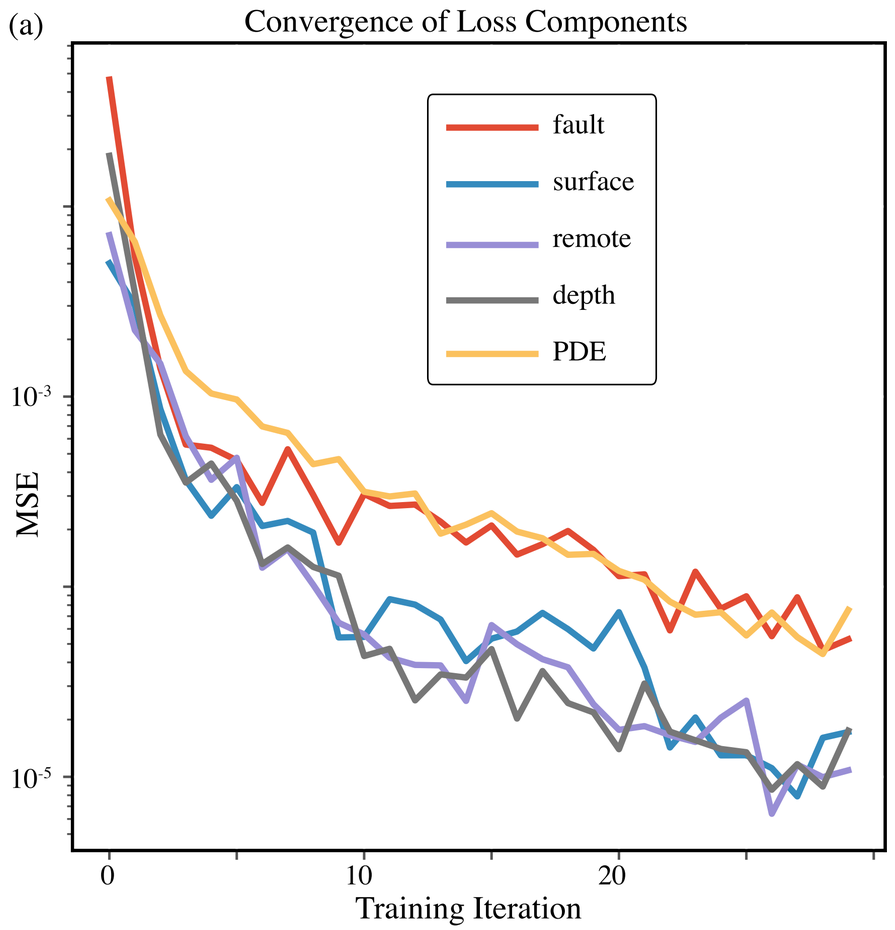


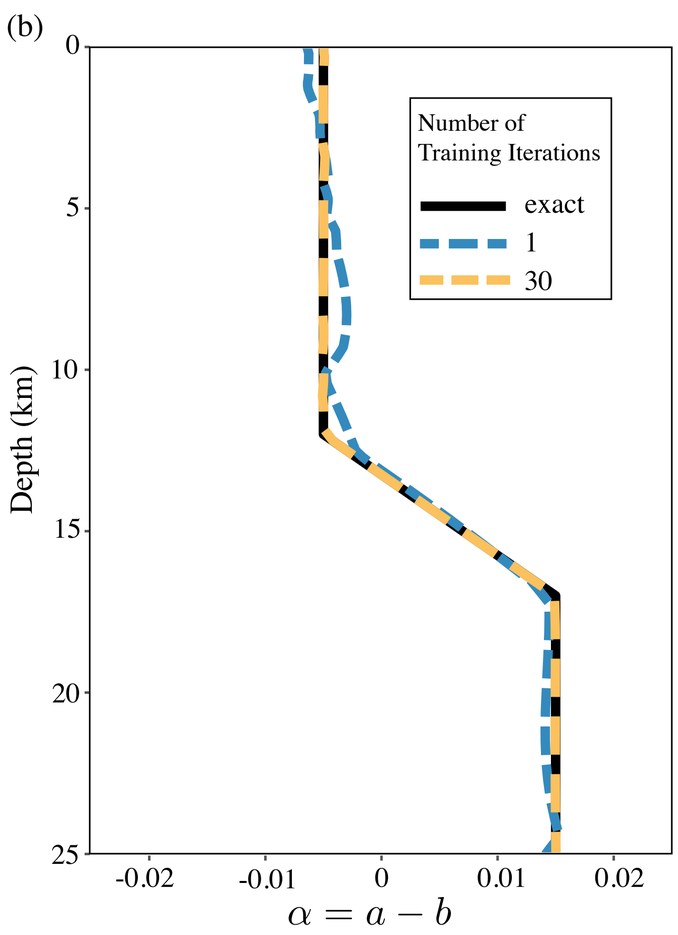
Additional connected network to learn depth-dependency of $a - b$

Problem set-ups:

- **2D antiplane**
- **Elastodynamic**
- **Manufactured solution allows for ground truth**

Inversions with rate-and-state friction





Findings:

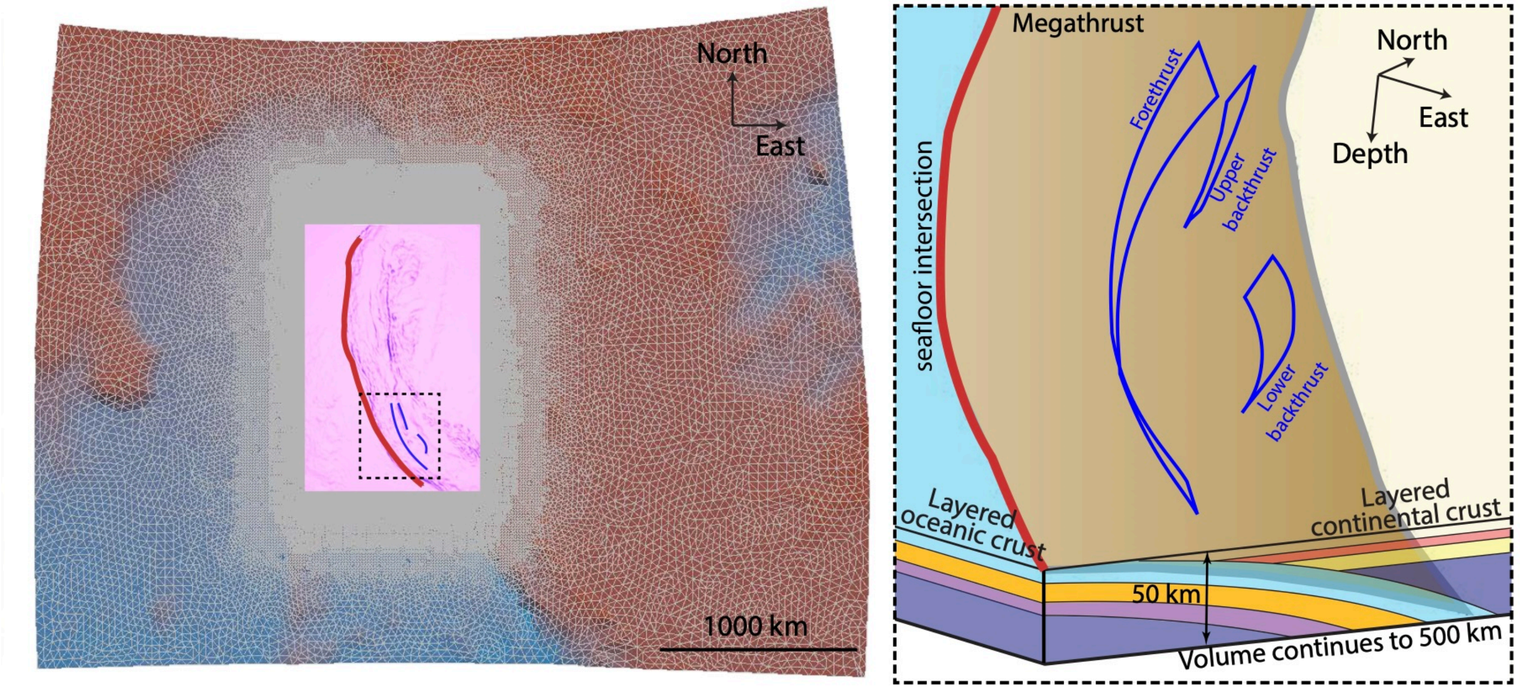
- **MSE non-monotonically decreasing**
- **Connected network learns $a - b$ faster than PDE solution**
- **Helps to have ground truth, in practical applications won't have this**

Rucker and Erickson (CMAME, '24)

- Computed in data-parallel, single-node GPU, in Python using PyTorch
- 20 min, comparable to forward solve

PiNNs compared to traditional methods:

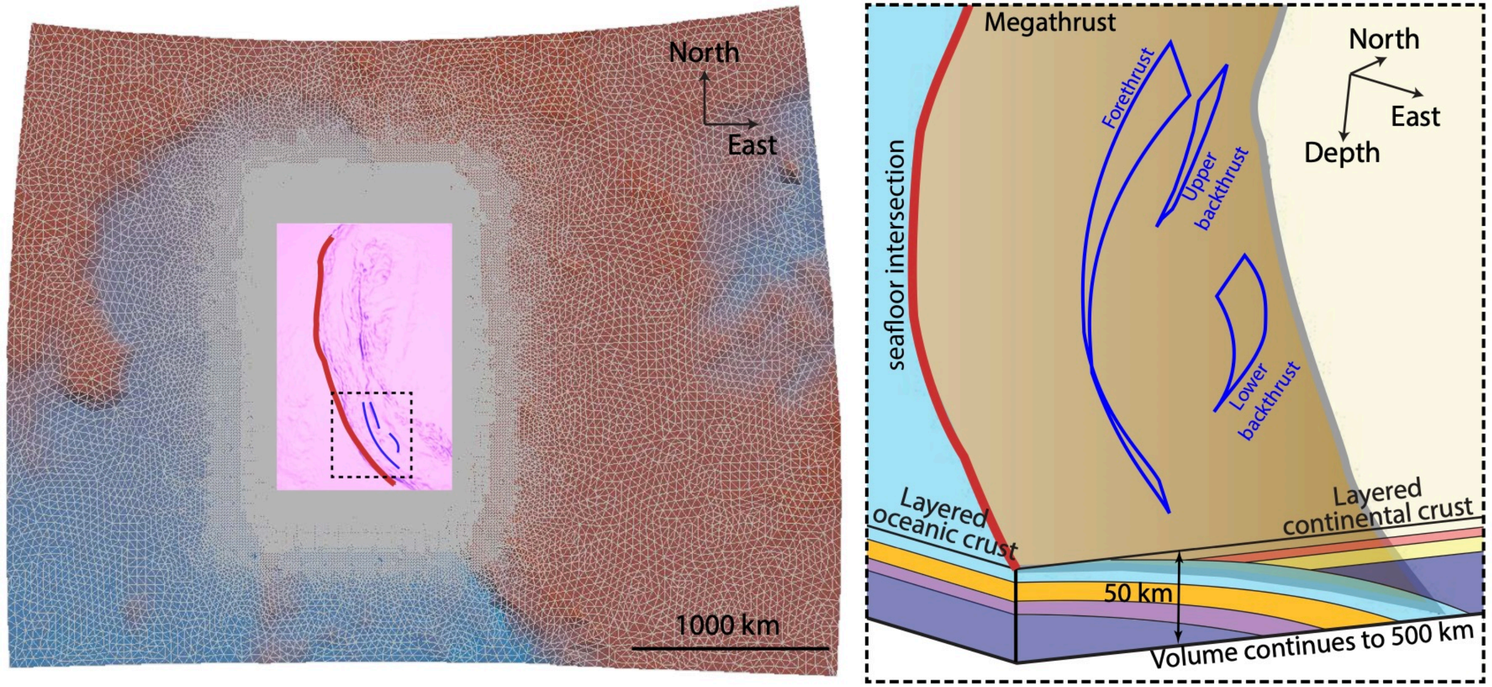
e.g. SeisSol (traditional): high order accuracy on tetrahedral meshes, complex fault geometries and material properties, regularly run on supercomputers, etc.



High-resolution simulation of the 2004 Sumatra-Andaman earthquake, Uphoff et al. (SC, ’17)

PiNNs compared to traditional methods:

e.g. SeisSol (traditional): high order accuracy on tetrahedral meshes, complex fault geometries and material properties, regularly run on supercomputers, etc.



High-resolution simulation of the 2004 Sumatra-Andaman earthquake, Uphoff et al. (SC, '17)

Some potential advantages of PiNNs

- Mesh-free
- May handle complex interfaces more easily
- Suitable for ill-posed inverse problems
- Forward/inverse same computational method, inverse problems may be key
- Learning of of multi-scale physics?

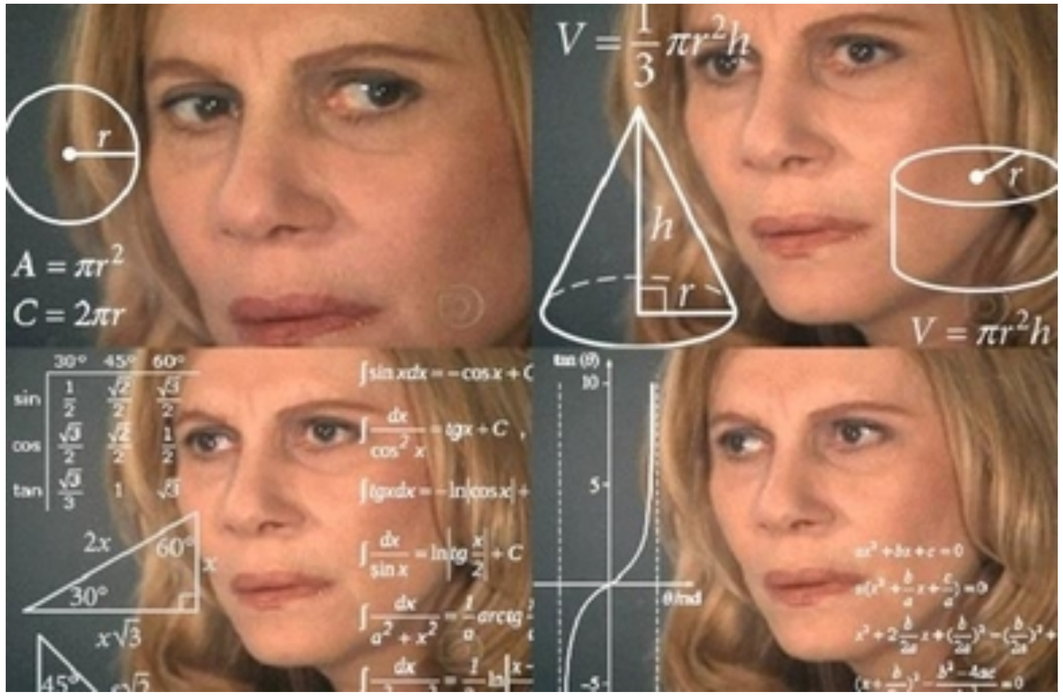
- Transfer Learning
- Auto satisfy absorbing BC (Rasht-Behesht et al., 2022)
- Solutions to new problems (e.g. alternative BC)
- Learning of physics itself - coefficients of a PDE or Deep Neural operators (learn PDE operator itself)

Limitations and Needs

- Not obviously better than traditional methods for traditional forward problems (Cuomo et al., Grossman et al.), except maybe in higher dimensions, not more accurate.

- Loss function - convex? How to quantify errors? What about convergence? How to initialize weights, pick collocation points?

- How to make sense of output?
  - Model reproducibility, verification, validation and valuation
  - UQ - although some has been done here (BPiNNs), same with error estimates



"Math Lady Meme"

Future directions and possibilities

- Somebody needs to sit down and quantify the computational benefits (or lack of) when comparing PiNNs to traditional numerical methods (e.g. finite difference), for both forward (follow up on Grossman) and inverse problems.
    - What is the computational complexity, memory requirements etc. for two algorithms to say, reach the same level of accuracy?
- More work needs to be done in terms of UQ

For earthquake science:

- Explore learning of multi-scale processes
    - SCEC benchmark exercises?
- Sensitivity analysis, in particular with well-instrumented lab studies
- ???

*Please visit SCEC Group A poster #115 (Rucker and Erickson)

References on Deep Learning Material

Cuomo et al. (2022), Scientific Machine Learning Through Physics–Informed Neural Networks:Where we are and What's Next, JSC.

Grossman et al. (2024), Can Physics-Informed Neural Networks beat the Finite Element Method?, IMA J Appl Math.

Kollmannsburger et al. (2021), Deep Learning in Computational Mechanics (Springer).

Mousavi & Beroza (2022), Deep-learning Seismology, Science.

Raissi et al. (2018), Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, JCP.

Moseley et al. (2023), Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM.

Rucker & Erickson (2024), Physics-informed deep learning of rate-and-state fault friction, CMAME.

Smith et al. (2021), EikoNet: Solving the Eikonal Equation With Deep Neural Networks, IEEE Trans. Geosci. Remote Sens.

Rasht-Behesht, et al., (2022), Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions, JGR.

Questions: bae@uoregon.edu, crucker@uoregon.edu