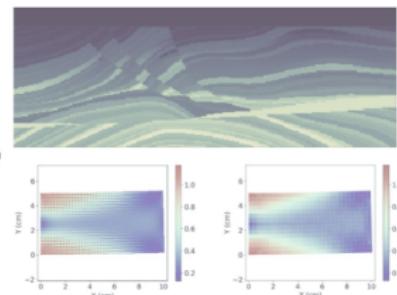
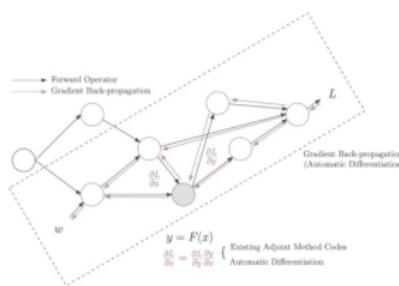
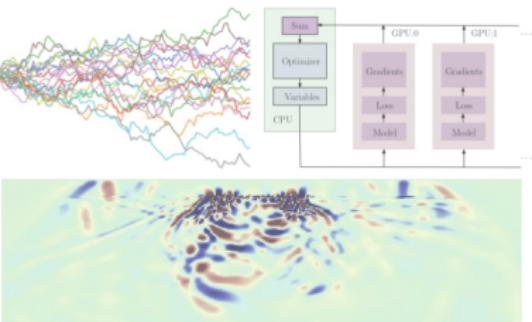


# Machine Learning for Inverse Problems in Computational Engineering

Kailai Xu and Eric Darve

<https://github.com/kailaix/ADCME.jl>



# Outline

1 Inverse Modeling

2 Automatic Differentiation

3 Code Example

4 Applications

# Inverse Modeling

## Forward Problem

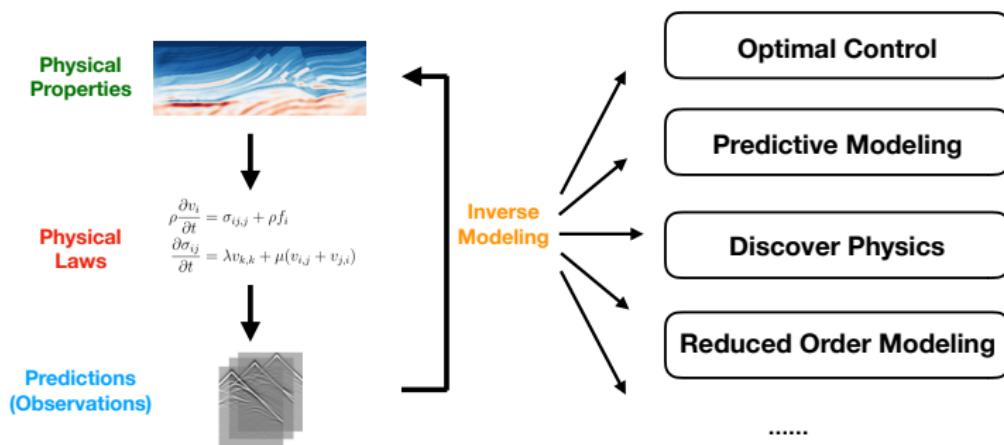


## Inverse Problem



# Inverse Modeling

- **Inverse modeling** identifies a certain set of parameters or functions with which the outputs of the forward analysis matches the desired result or measurement.
- Many real life engineering problems can be formulated as inverse modeling problems: shape optimization for improving the performance of structures, optimal control of fluid dynamic systems, etc.t



# Inverse Modeling

We can formulate inverse modeling as a PDE-constrained optimization problem

$$\min_{\theta} L_h(u_h) \quad \text{s.t. } F_h(t, \theta, u_h) = 0$$

- The **loss function**  $L_h$  measures the discrepancy between the prediction  $u_h$  and the observation  $u_{\text{obs}}$ , e.g.,  $L_h(u_h) = \|u_h - u_{\text{obs}}\|_2^2$ .
- $\theta$  is the **model parameter** to be calibrated.
- The **physics constraints**  $F_h(\theta, u_h) = 0$  are described by a system of partial differential equations or differential algebraic equations (DAEs), e.g.,  $H(u'_h, u_h, t; \theta) = 0$ . Solving for  $u_h$  may require solving linear systems or applying an iterative algorithm such as the Newton-Raphson method.

# Function Inverse Problem

$$\min_{\mathbf{f}} L_h(u_h) \quad \text{s.t. } F_h(\mathbf{f}, u_h) = 0$$

What if the unknown is a **function** instead of a set of parameters?

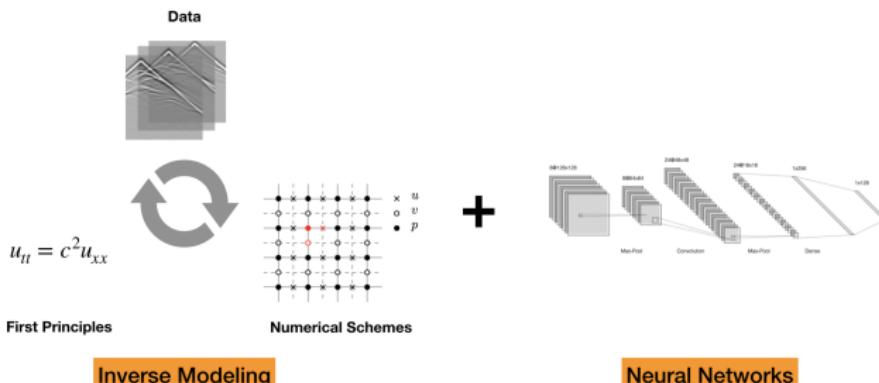
- Koopman operator in dynamical systems.
- Constitutive relations in solid mechanics.
- Turbulent closure relations in fluid mechanics.
- ...

The candidate solution space is **infinite dimensional**.

# Machine Learning for Computational Engineering

$$\min_{\theta} L_h(u_h) \quad \text{s.t. } F_h(\text{NN}_{\theta}, u_h) = 0$$

- Deep neural networks exhibit capability of approximating high dimensional and complicated functions.
- **Machine Learning for Computational Engineering:** the unknown function is approximated by a deep neural network, and the physical constraints are enforced by numerical schemes.
- Satisfy the physics to the largest extent.

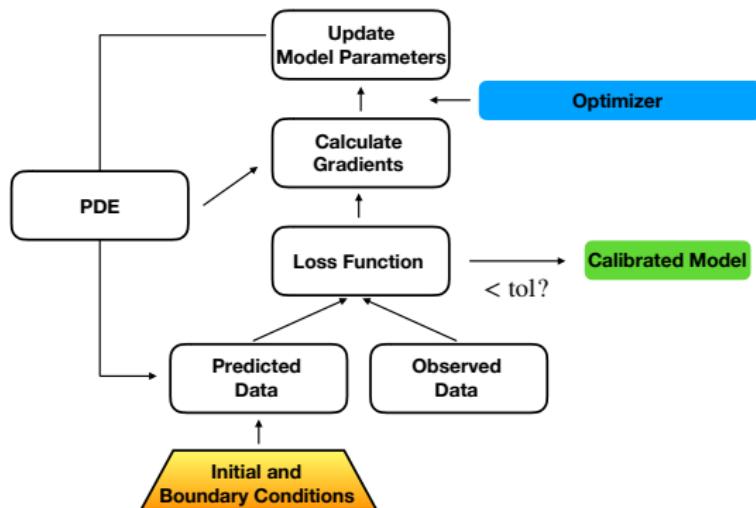


# Gradient Based Optimization

$$\min_{\theta} L_h(u_h) \quad \text{s.t. } F_h(\theta, u_h) = 0 \quad (1)$$

- We can now apply a gradient-based optimization method to (1).
- The key is to calculate the gradient descent direction  $g^k$

$$\theta^{k+1} \leftarrow \theta^k - \alpha g^k$$



# Outline

1 Inverse Modeling

2 Automatic Differentiation

3 Code Example

4 Applications

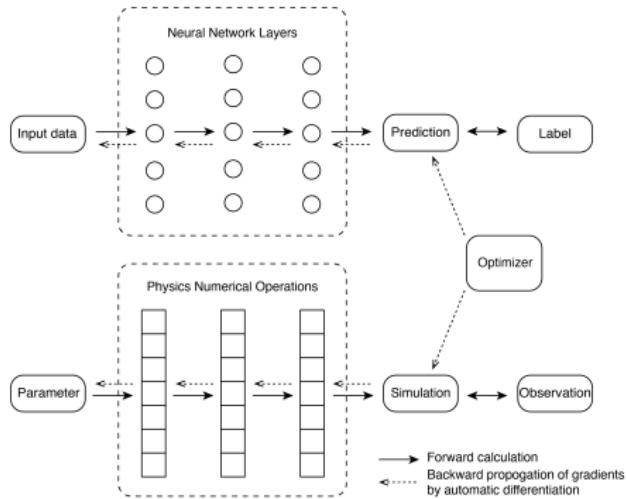
# Automatic Differentiation

The fact that bridges the **technical** gap between machine learning and inverse modeling:

- Deep learning (and many other machine learning techniques) and numerical schemes share the same computational model: composition of individual operators.

## Mathematical Fact

Back-propagation  
||  
Reverse-mode  
Automatic Differentiation  
||  
Discrete  
Adjoint-State Method



# What is the Appropriate Model for Inverse Problems?

- In general, for a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Mode	Suitable for ...	Complexity <sup>1</sup>	Application
Forward	$m \gg n$	$\leq 2.5 \text{ OPS}(f(x))$	UQ
Reverse	$m \ll n$	$\leq 4 \text{ OPS}(f(x))$	Inverse Modeling

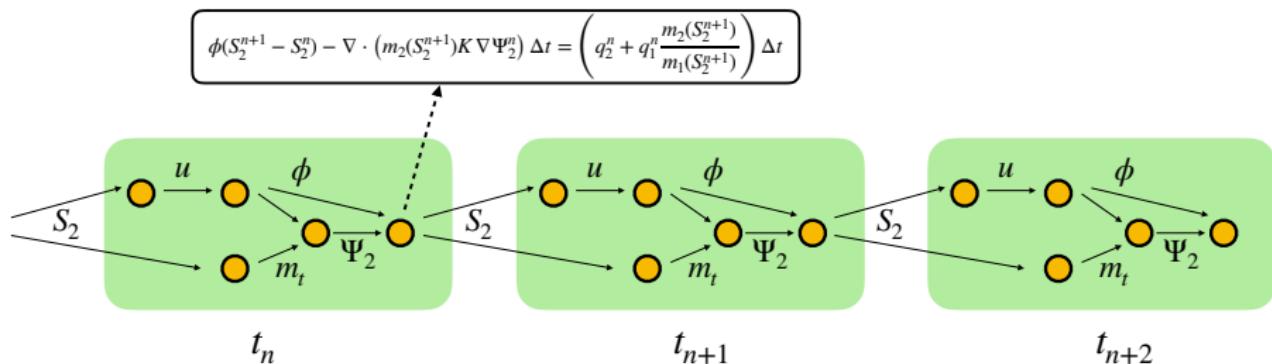
- There are also many other interesting topics
  - Mixed mode AD: many-to-many mappings.
  - Computing sparse Jacobian matrices using AD by exploiting sparse structures.

Margossian CC. A review of automatic differentiation and its efficient implementation. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2019 Jul;9(4):e1305.

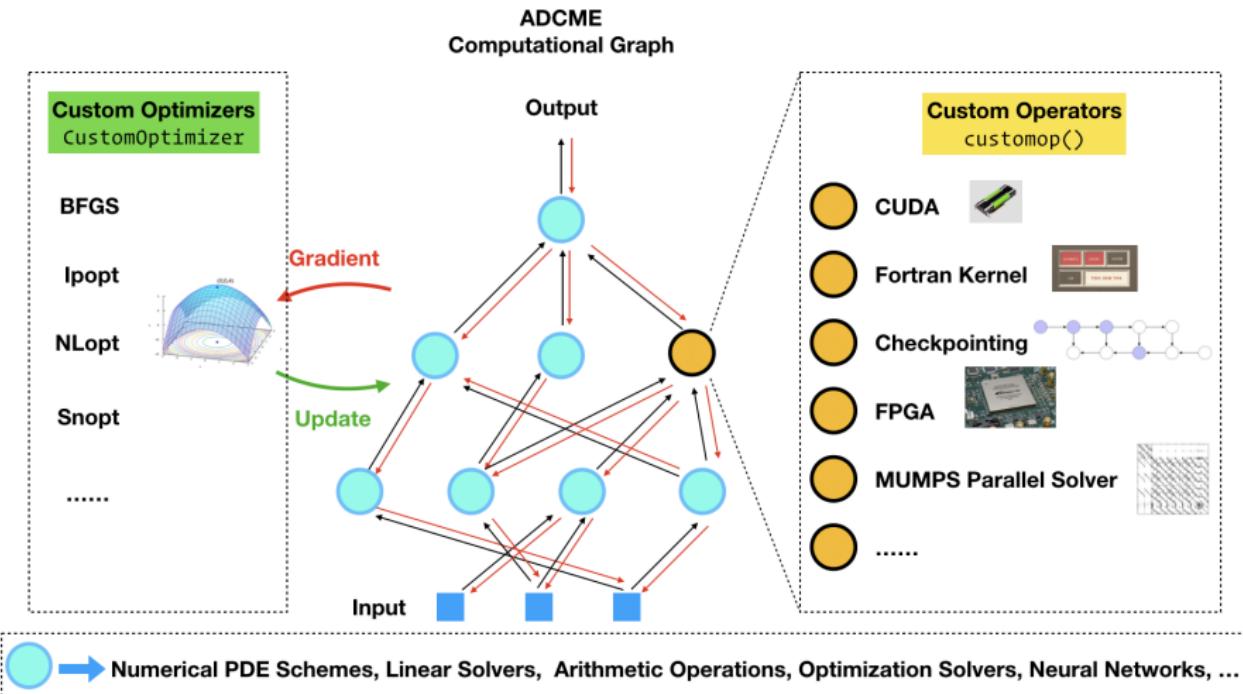
<sup>1</sup>OPS is a metric for complexity in terms of fused-multiply adds.

# Computational Graph for Numerical Schemes

- To leverage automatic differentiation for inverse modeling, we need to express the numerical schemes in the “AD language”: computational graph.
- No matter how complicated a numerical scheme is, it can be decomposed into a collection of operators that are interlinked via state variable dependencies.

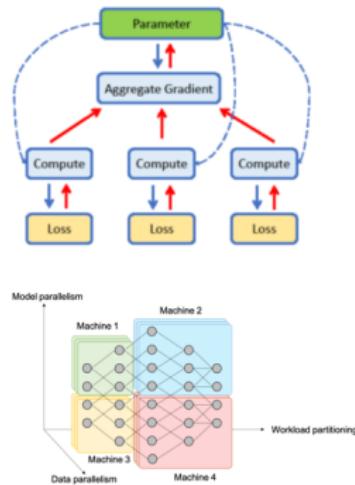


# ADCME: Computational-Graph-based Numerical Simulation

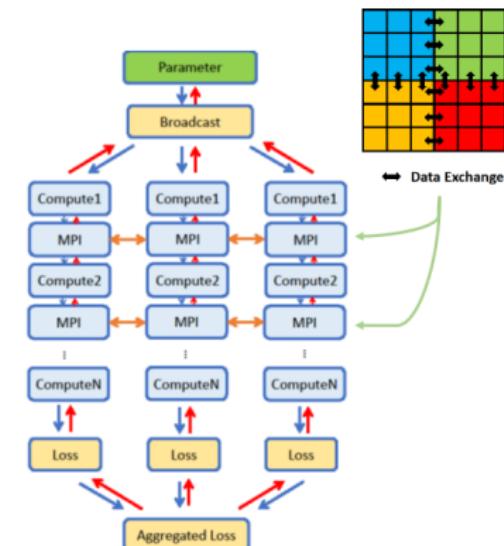


# Parallel Computing

- Parallel computing is essential for accelerating simulation and satisfying demanding memory requirements.



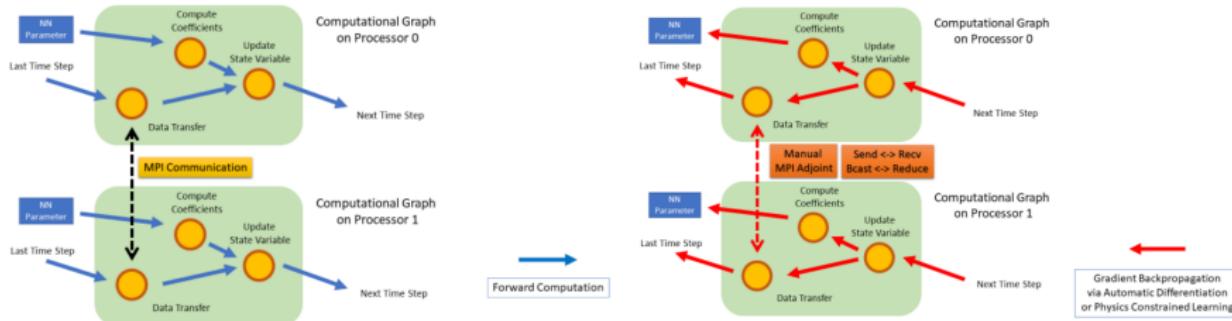
Deep Learning Data/Model Parallelism



Scientific Computing Mixed Parallelism

# Distributed Optimization

- ADCME also supports MPI-based distributed computing. The parallel model is designed specially for scientific computing.



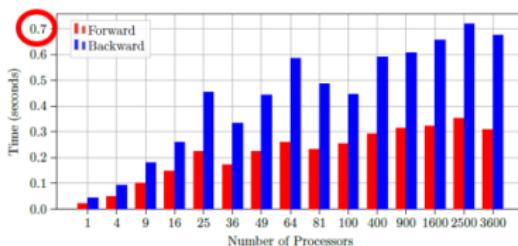
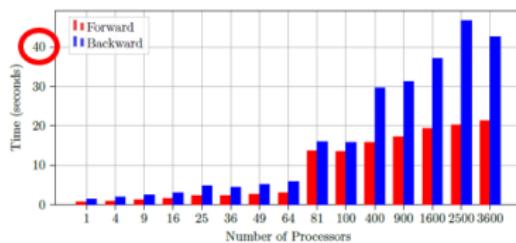
- Key idea: **Everything is an operator.** Computation and communications are converters of data streams (tensors) through the computational graph.

`mpi_bcast, mpi_sum, mpi_send, mpi_recv, mpi_halo_exchange, ...`

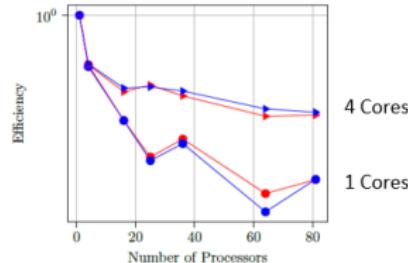
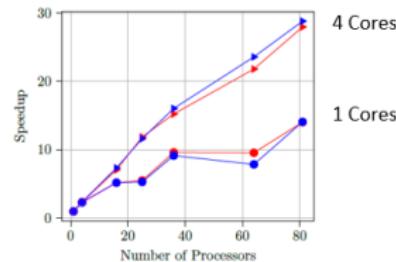
# Interoperability with Hypre

$$\nabla \cdot (\text{NN}_\theta(\mathbf{x}) \nabla u(\mathbf{x})) = f(\mathbf{x}) \quad \mathbf{x} \in \Omega$$
$$u(\mathbf{x}) = 0 \quad \mathbf{x} \in \partial\Omega$$

The discretization leads to a linear system, which is solved using Hypre.



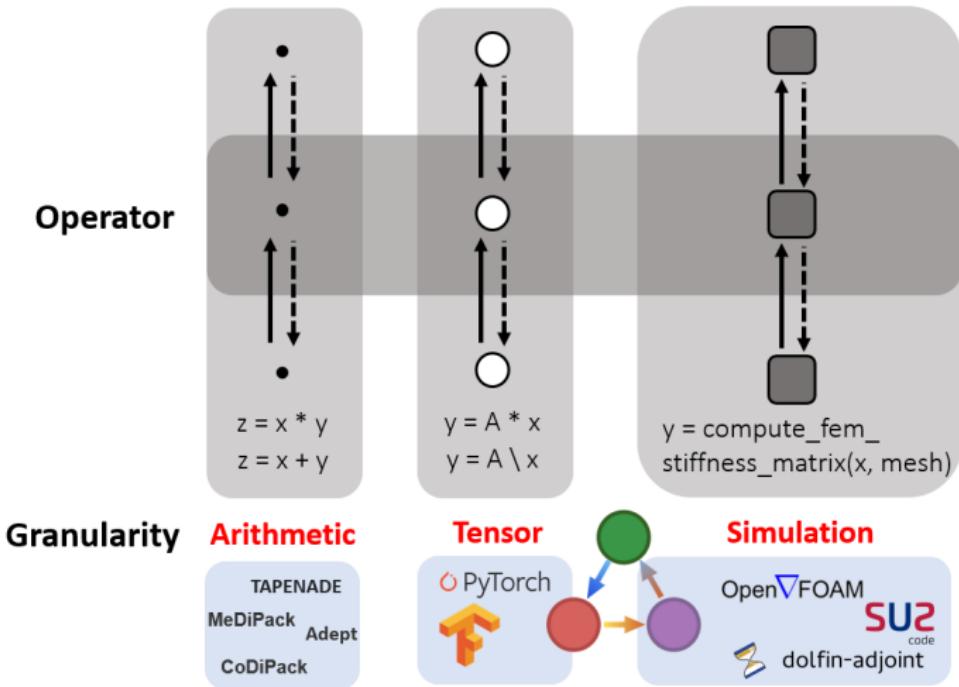
Weak Scalability



Strong Scalability

# Granularity of Automatic Differentiation

- Coarser granularity gives researchers more control over gradient back-propagation.



# Outline

1 Inverse Modeling

2 Automatic Differentiation

3 Code Example

4 Applications

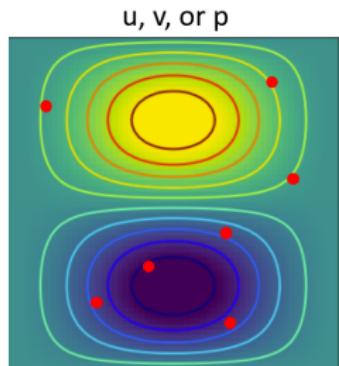
# Inverse Modeling of the Stokes Equation

- The governing equation for the Stokes problem

$$\begin{aligned}-\nu \Delta \mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega \\ \nabla \cdot \mathbf{u} &= 0 && \text{in } \Omega \\ \mathbf{u} &= \mathbf{0} && \text{on } \partial\Omega\end{aligned}$$

- The weak form is given by

$$\begin{aligned}(\nu \nabla u, \nabla v) - (p, \nabla \cdot v) &= (f, v) \\ (\nabla \cdot u, q) &= 0\end{aligned}$$



# Inverse Modeling of the Stokes Equation

```
nu = Variable(0.5)
K = nu*constant(compute_fem_laplace_matrix(m, n, h))
B = constant(compute_interaction_matrix(m, n, h))
Z = [K -B'
      -B spdiag(zeros(size(B,1)))]  
  
# Impose boundary conditions
bd = bcnode("all", m, n, h)
bd = [bd; bd .+ (m+1)*(n+1); ((1:m) .+ 2*(m+1)*(n+1))]
Z, _ = fem_impose_Dirichlet_boundary_condition1(Z, bd, m, n, h)  
  
# Calculate the source term
F1 = eval_f_on_gauss_pts(f1func, m, n, h)
F2 = eval_f_on_gauss_pts(f2func, m, n, h)
F = compute_fem_source_term(F1, F2, m, n, h)
rhs = [F;zeros(m*n)]
rhs[bd] .= 0.0  
  
sol = Z\rhs
```

# Inverse Modeling of the Stokes Equation

- The distinguished feature compared to traditional forward simulation programs: **the model output is differentiable with respect to model parameters!**

```
loss = sum((sol[idx] - observation[idx])^2)  
g = gradients(loss, nu)
```

- Optimization with a one-liner:

```
BFGS!(sess, loss)
```



ADCME/AdFem



Simulation Program

# Outline

1 Inverse Modeling

2 Automatic Differentiation

3 Code Example

4 Applications

# Constitutive Modeling

$$\underbrace{\sigma_{ij,j}}_{\text{stress}} + \rho \underbrace{b_i}_{\text{external force}} = \rho \underbrace{\ddot{u}_i}_{\text{velocity}} \quad (2)$$
$$\underbrace{\varepsilon_{ij}}_{\text{strain}} = \frac{1}{2}(u_{j,i} + u_{i,j})$$

- **Observable:** external/body force  $b_i$ , displacements  $u_i$  (strains  $\varepsilon_{ij}$  can be computed from  $u_i$ ); density  $\rho$  is known.
- **Unobservable:** stress  $\sigma_{ij}$ .
- Data-driven Constitutive Relations: modeling the strain-stress relation using a neural network

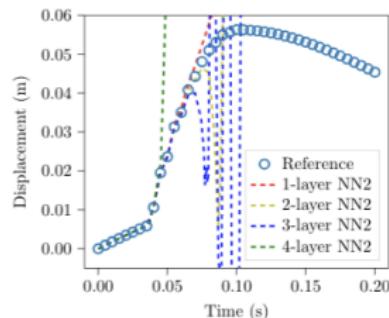
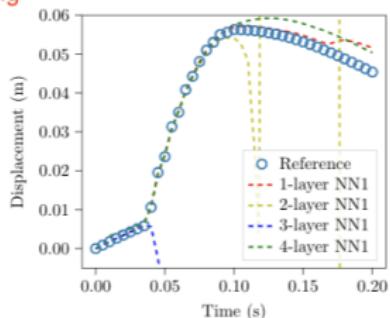
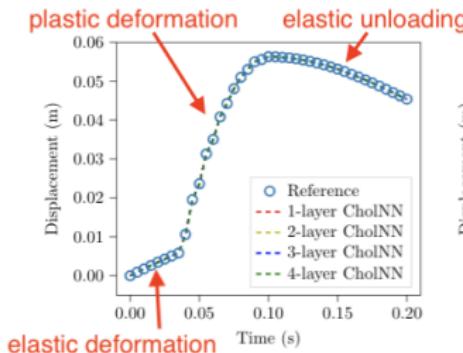
$$\boxed{\text{stress} = \mathcal{M}_\theta(\text{strain}, \dots)} \quad (3)$$

and the neural network is trained by coupling Eq. 2 and Eq. 3.

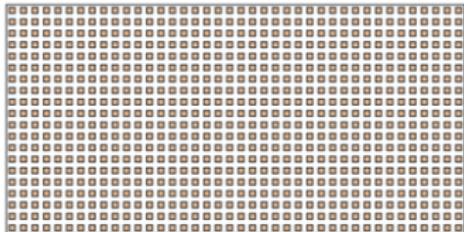
# Modeling Elasto-plasticity

- Comparison of different neural network architectures

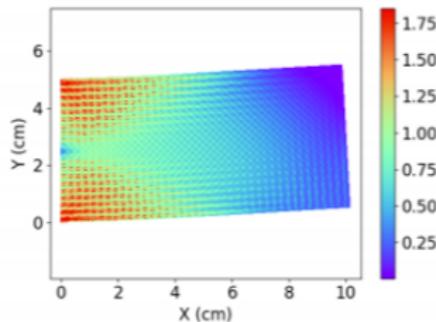
$$\sigma^{n+1} = L_\theta(\epsilon^{n+1}, \epsilon^n, \sigma^n) L_\theta(\epsilon^{n+1}, \epsilon^n, \sigma^n)^T (\epsilon^{n+1} - \epsilon^n) + \sigma^n$$
$$\sigma^{n+1} = \text{NN}_\theta(\epsilon^{n+1}, \epsilon^n, \sigma^n)$$
$$\sigma^{n+1} = \text{NN}_\theta(\epsilon^{n+1}, \epsilon^n, \sigma^n) + \sigma^n$$



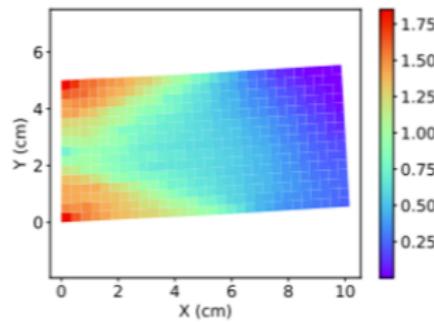
# Modeling Elasto-plasticity: Multi-scale



Fiber Reinforced Thin Plate



Reference von Mises stress



SPD-NN

# Poroelasticity

- Multi-physics Interaction of Coupled Geomechanics and Multi-Phase Flow Equations

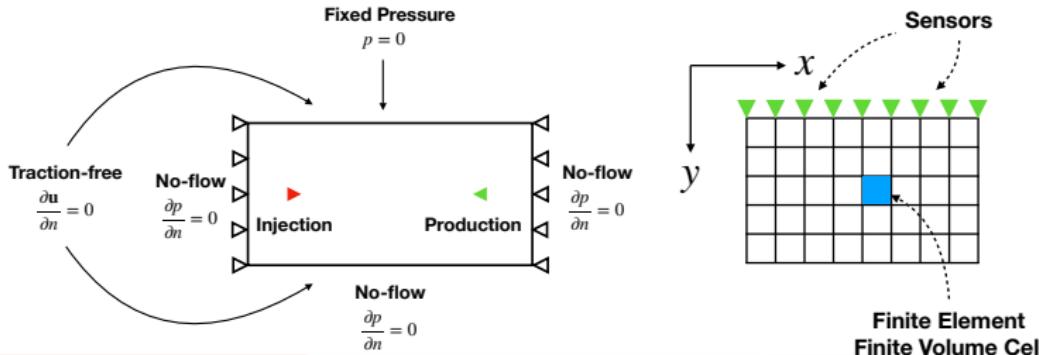
$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{u}) - b\nabla p = 0$$

$$\frac{1}{M} \frac{\partial p}{\partial t} + b \frac{\partial \epsilon_v(\mathbf{u})}{\partial t} - \nabla \cdot \left( \frac{k}{B_f \mu} \nabla p \right) = f(x, t)$$

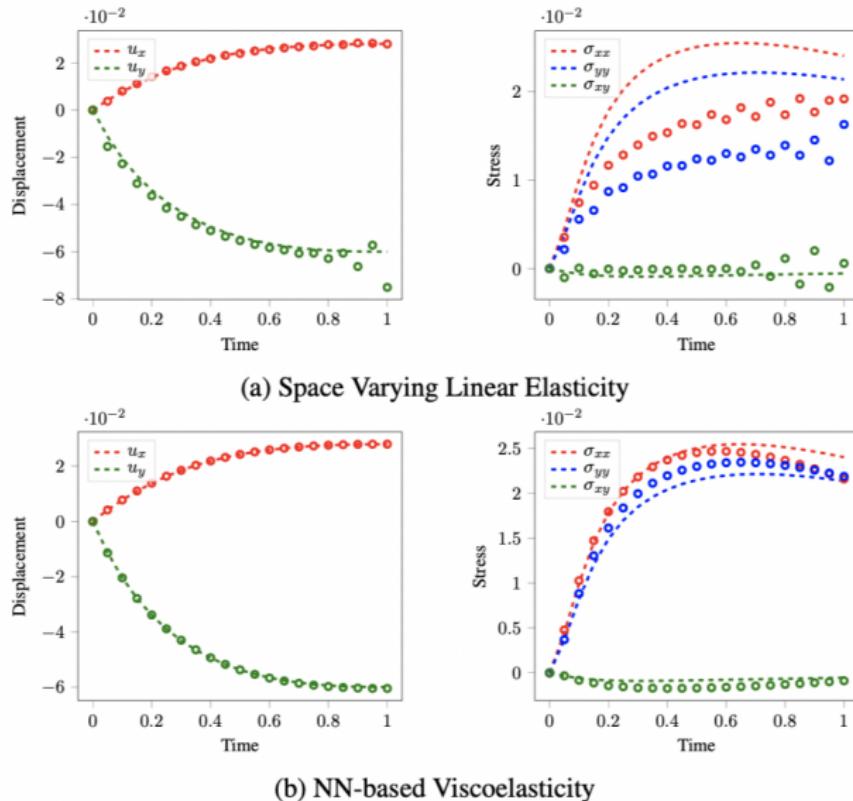
$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}, \dot{\boldsymbol{\epsilon}})$$

- Approximate the constitutive relation by a neural network

$$\boldsymbol{\sigma}^{n+1} = \mathcal{NN}_{\theta}(\boldsymbol{\sigma}^n, \boldsymbol{\epsilon}^n) + H\boldsymbol{\epsilon}^{n+1}$$



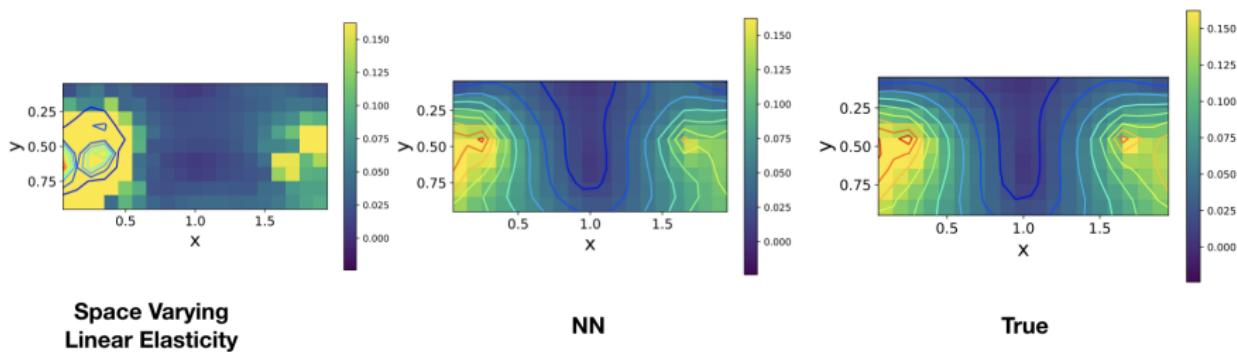
# Poroelasticity



# Poroelasticity

- Comparison with space varying linear elasticity approximation

$$\sigma = H(x, y)\epsilon$$

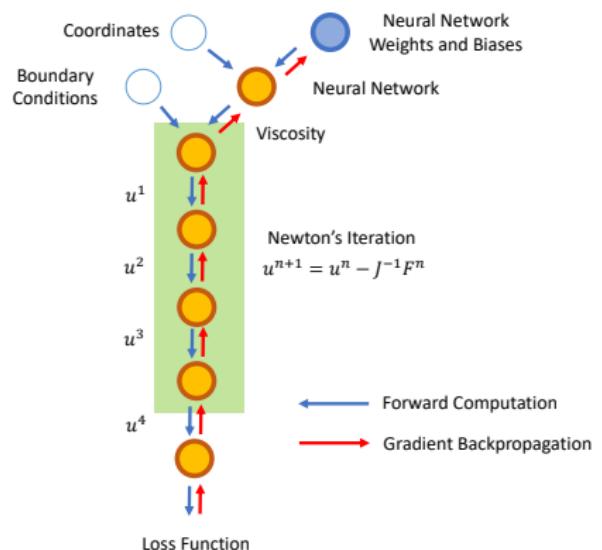
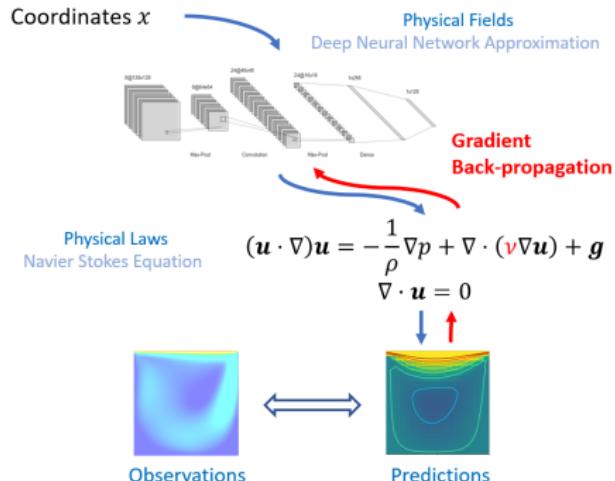


Space Varying  
Linear Elasticity

NN

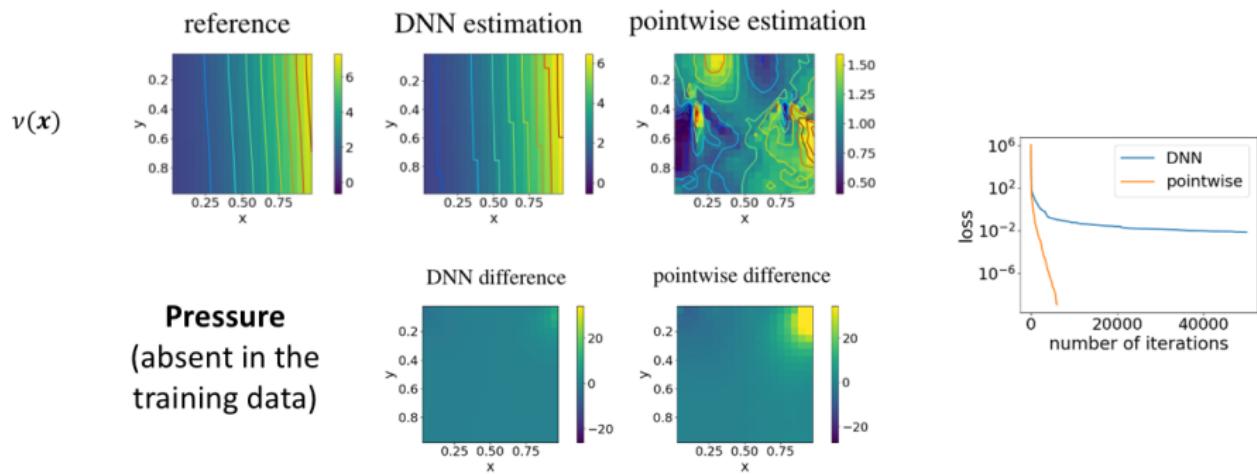
True

# Navier-Stokes Equation



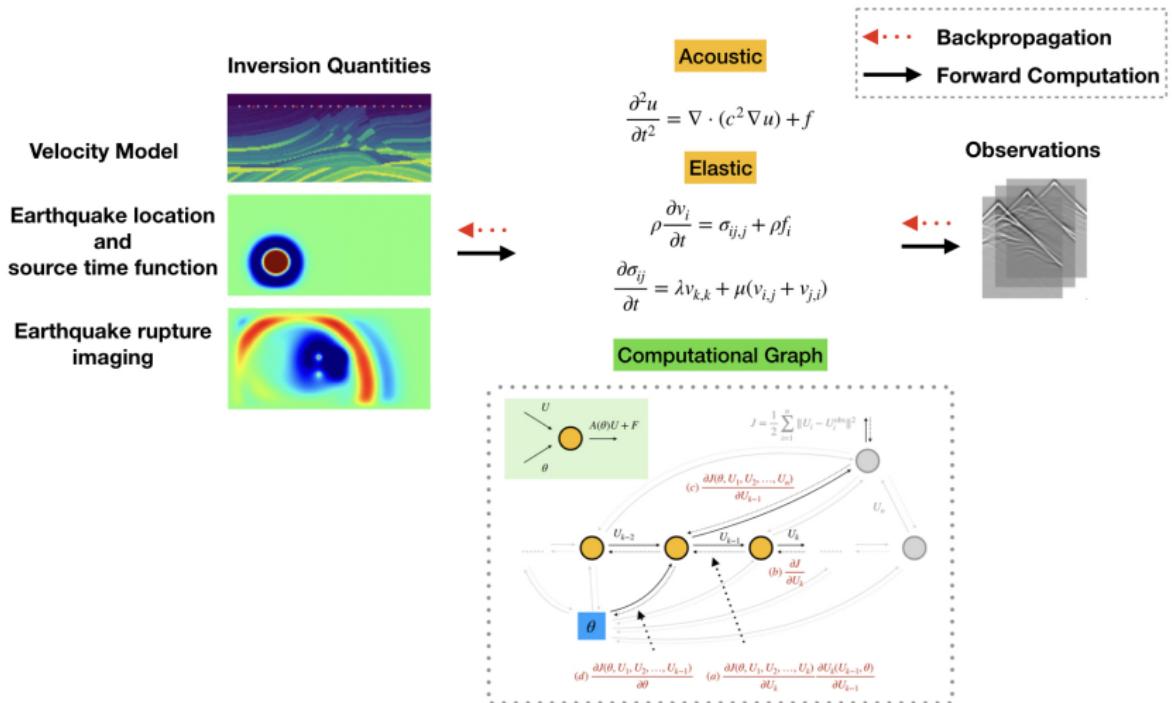
# Navier-Stokes Equation

- Data:  $(u, v)$
- Unknown:  $v(x)$  (represented by a deep neural network)
- Prediction:  $p$  (absent in the training data)
- The DNN provides regularization, which generalizes the estimation better!



# ADSeismic.jl: A General Approach to Seismic Inversion

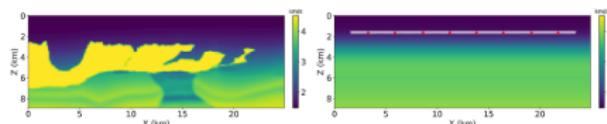
- Many seismic inversion problems can be solved within a unified framework.



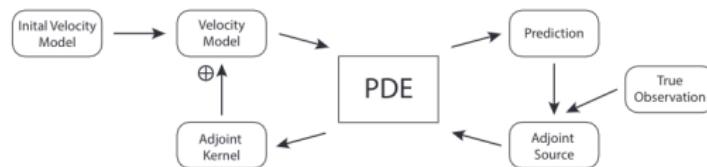
# NNFWI: Neural-network-based Full-Waveform Inversion

- Estimate velocity models from seismic observations.

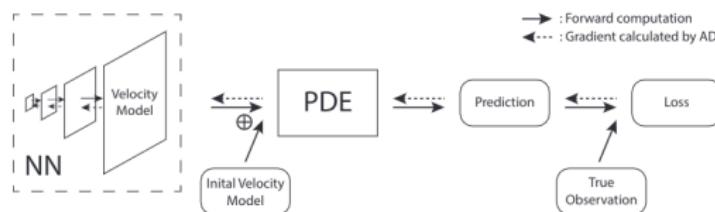
$$\frac{\partial^2 u}{\partial t^2} = \nabla \cdot (\mathbf{m}^2 \nabla u) + f$$



(a) Traditional FWI:

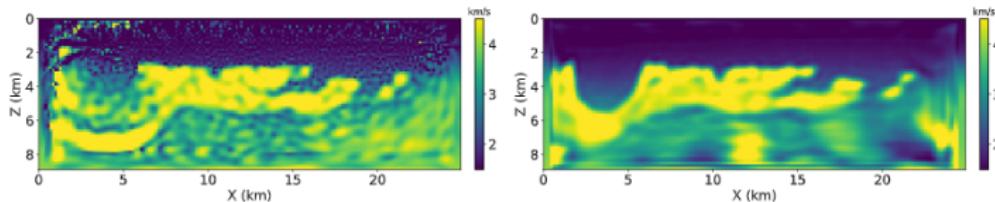


(b) NNFWI:

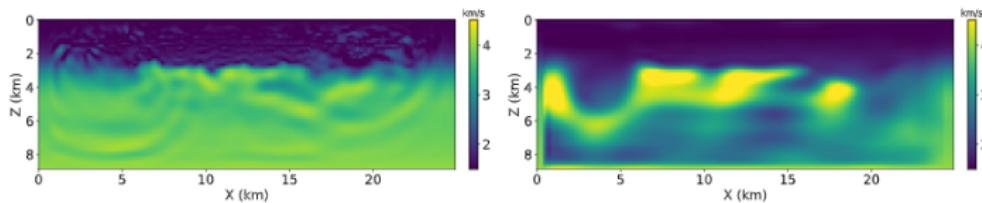


# NNFWI: Neural-network-based Full-Waveform Inversion

- Inversion results with a noise level  $\sigma = \sigma_0$

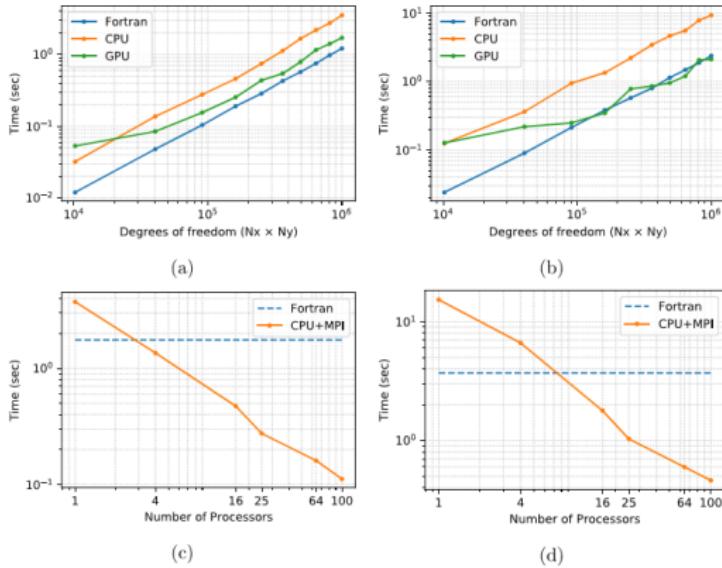


- Inversion results for the same loss function value:

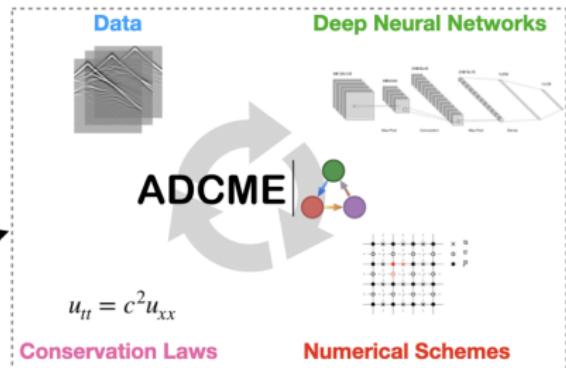


# ADSeismic.jl: Performance Benchmark

- Performance is a key focus of ADCME.
- ADCME enables us to utilize heterogeneous (CPUs, GPUs, and TPUs) and distributed (CPU clusters) computing environments.  
Fortran: open-source Fortran90 programs SEISMIC\_CPM1



# A General Approach to Inverse Modeling



AdFem.jl  
Finite Element Library  
for Inverse Modeling  
<https://github.com/kailaix/AdFem.jl>

NNFEM.jl  
NNFEM.jl  
Constitutive Law Modeling  
Hyperelasticity  
Elasto-Plasticity  
<https://github.com/kailaix/NNFEM.jl>