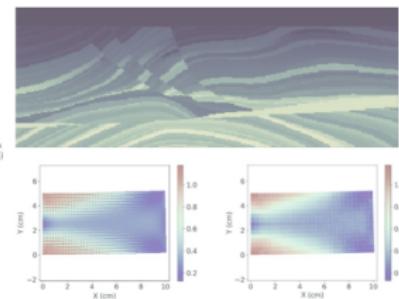
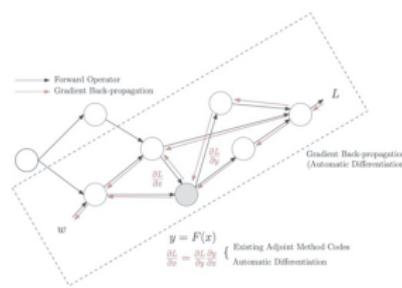
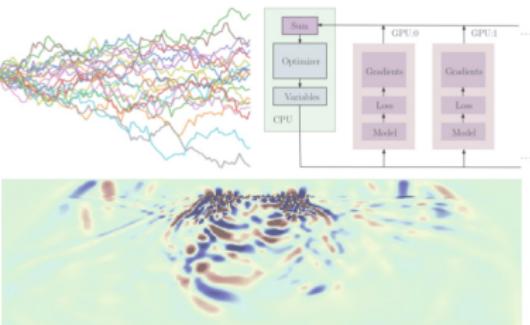


# Subsurface Inverse Modeling with Physics Based Machine Learning

Kailai Xu and Dongzhuo Li  
Jerry M. Harris, Eric Darve



# Outline

1 Inverse Modeling

2 Automatic Differentiation

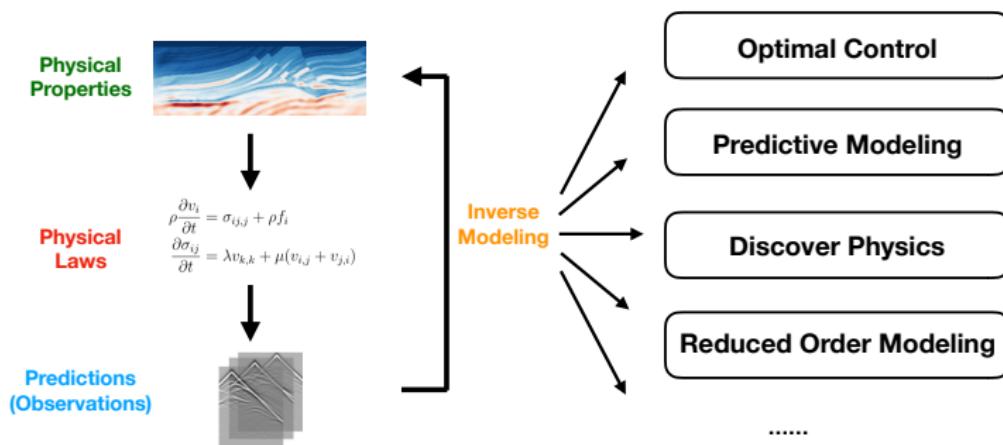
3 Physics Constrained Learning

4 Applications

5 ADCME: Scientific Machine Learning for Inverse Modeling

# 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

## Forward Problem

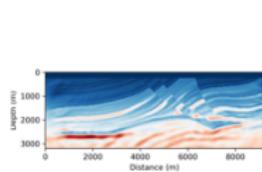


## Inverse Problem

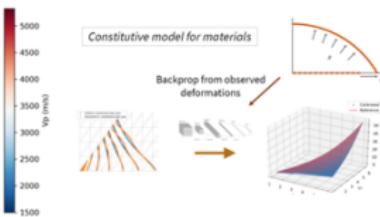


# Inverse Modeling for Subsurface Properties

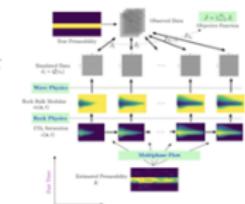
There are many forms of subsurface inverse modeling problems.



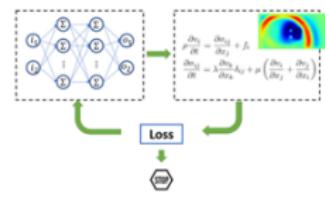
Parameter Inverse Problem



Function Inverse Problem



Coupled Inversion



Embedded Neural Networks in PDEs

## The Central Challenge

Can we have a general approach for solving these inverse problems?

# Parameter Inverse Problem

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

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

- 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. Solving for  $u_h$  may require solving linear systems or applying an iterative algorithm such as the Newton-Raphson method.

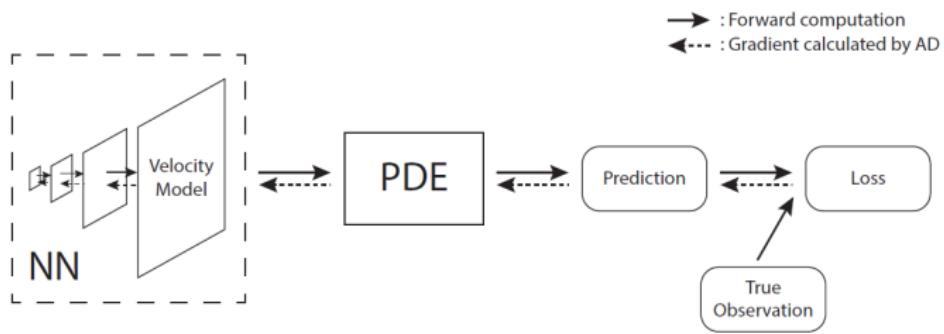
# Function Inverse Problem

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

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.
- Neural-network-based physical properties.
- ...

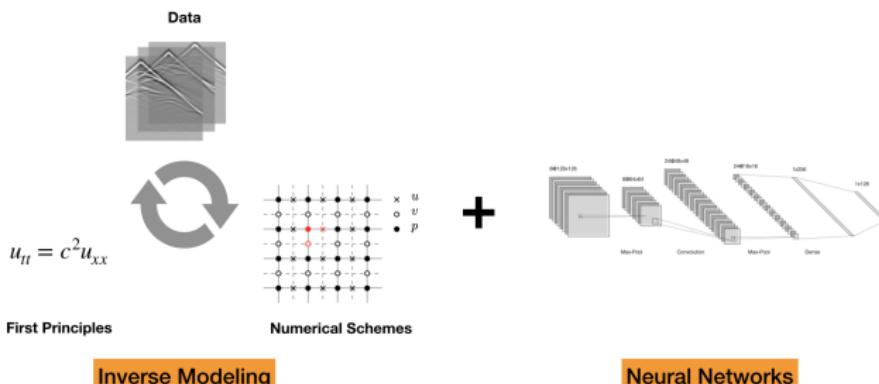
The candidate solution space is **infinite dimensional**.



# Physics Based Machine Learning

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

- Deep neural networks exhibit capability of approximating high dimensional and complicated functions.
- **Physics based machine learning:** 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.



Inverse Modeling

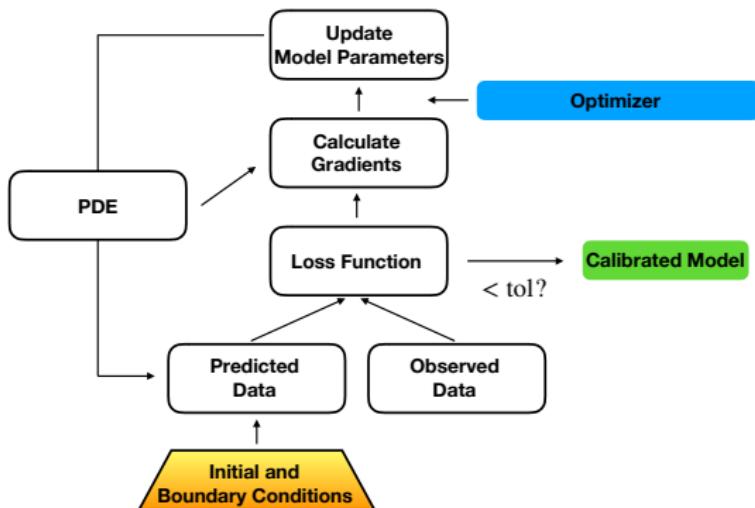
Neural Networks

# 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 Physics Constrained Learning

4 Applications

5 ADCME: Scientific Machine Learning for Inverse Modeling

# 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

||

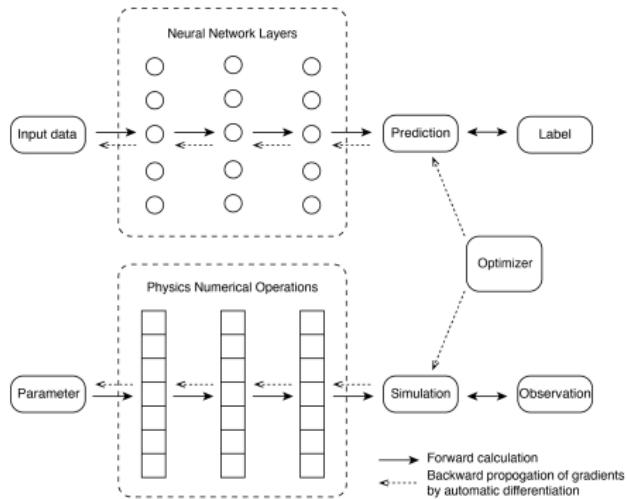
Reverse-mode

Automatic Differentiation

||

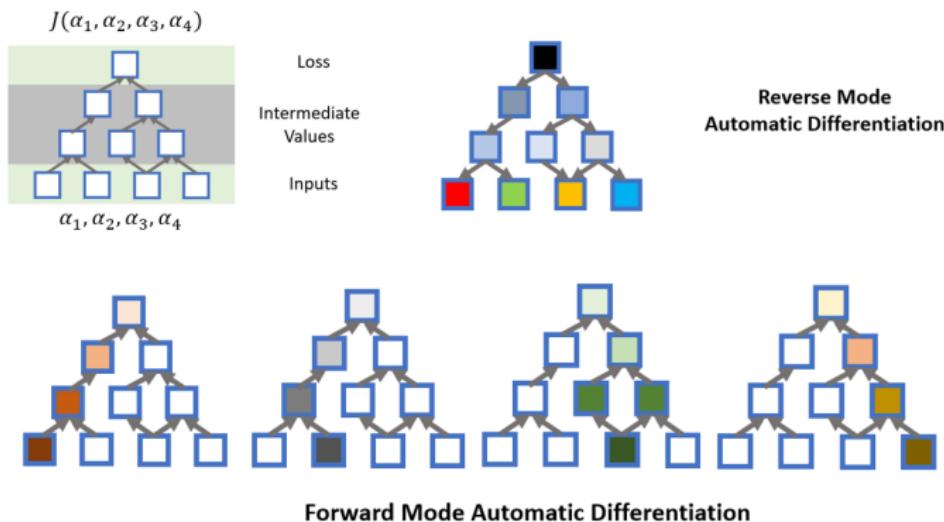
Discrete

Adjoint-State Method



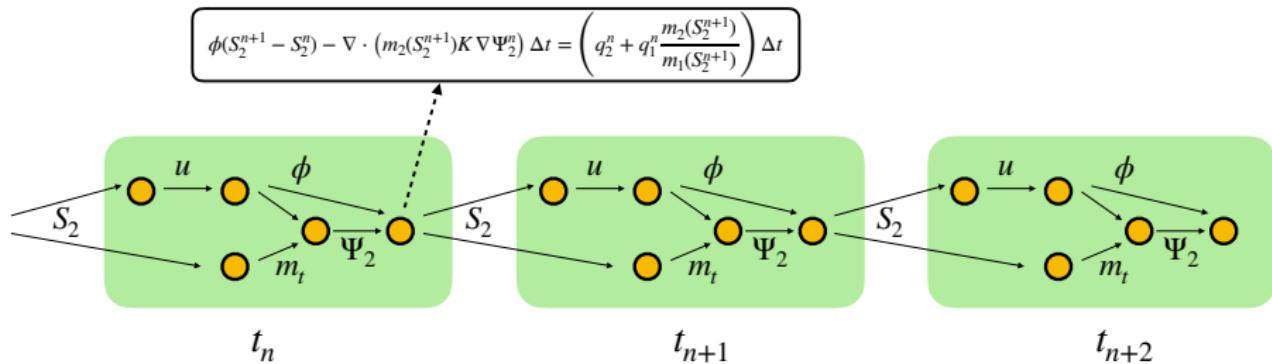
# Forward Mode vs. Reverse Mode

- Reverse mode automatic differentiation evaluates gradients in the **reverse order** of forward computation.
- Reverse mode automatic differentiation is a more efficient way to compute gradients of a many-to-one mapping  $J(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \Rightarrow$  suitable for minimizing a loss (misfit) function.



# 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.

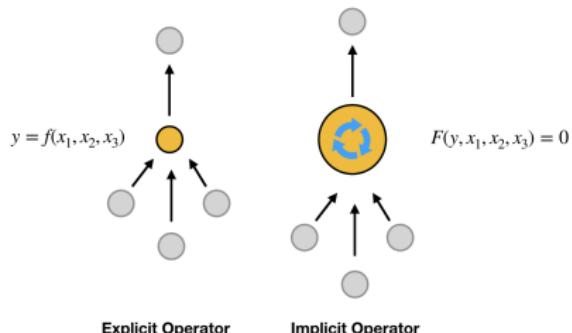


# Outline

- 1 Inverse Modeling
- 2 Automatic Differentiation
- 3 Physics Constrained Learning
- 4 Applications
- 5 ADCME: Scientific Machine Learning for Inverse Modeling

# Challenges in AD

- Most AD frameworks only deal with **explicit operators**, i.e., the functions with analytical derivatives that are easy to implement.
- Many scientific computing algorithms are **iterative** or **implicit** in nature.



Linear/Nonlinear	Explicit/Implicit	Expression
Linear	Explicit	$y = Ax$
Nonlinear	Explicit	$y = F(x)$
<b>Linear</b>	<b>Implicit</b>	$Ay = x$
<b>Nonlinear</b>	<b>Implicit</b>	$F(x, y) = 0$

# Implicit Operators in Subsurface Modeling

- For reasons such as nonlinearity and stability, implicit operators (schemes) are almost everywhere in subsurface modeling...

## Navier-Stokes Equations

$$\begin{aligned}x: \quad & \rho \left( \frac{\partial u_x}{\partial t} + u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} + u_z \frac{\partial u_x}{\partial z} \right) = - \frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} \right) + \frac{1}{3} \mu \frac{\partial}{\partial x} \left( \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) + \rho g_x \\y: \quad & \rho \left( \frac{\partial u_y}{\partial t} + u_x \frac{\partial u_y}{\partial x} + u_y \frac{\partial u_y}{\partial y} + u_z \frac{\partial u_y}{\partial z} \right) = - \frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_y}{\partial z^2} \right) + \frac{1}{3} \mu \frac{\partial}{\partial y} \left( \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) + \rho g_y \\z: \quad & \rho \left( \frac{\partial u_z}{\partial t} + u_x \frac{\partial u_z}{\partial x} + u_y \frac{\partial u_z}{\partial y} + u_z \frac{\partial u_z}{\partial z} \right) = - \frac{\partial p}{\partial z} + \mu \left( \frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2} + \frac{\partial^2 u_z}{\partial z^2} \right) + \frac{1}{3} \mu \frac{\partial}{\partial z} \left( \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) + \rho g_z\end{aligned}$$

## Two-phase Flow Equations

$$\partial_t (\alpha_k \rho_k A) + \nabla \cdot [\alpha_k A (\rho_k \mathbf{u}_k \otimes \mathbf{u}_k + P_k \mathbb{I})] =$$

$$-\nabla A + P_{int} A \nabla \alpha_k + A \lambda_u (\mathbf{u}_j - \mathbf{u}_k)$$

$$\frac{\partial}{\partial t} \left[ \phi \left( \frac{S_o}{B_o} + \frac{R_V S_g}{B_g} \right) \right] + \nabla \cdot \left( \frac{1}{B_o} \vec{u}_o + \frac{R_V}{B_g} \vec{u}_g \right) = 0$$

$$\frac{\partial}{\partial t} \left[ \phi \left( \frac{S_w}{B_w} \right) \right] + \nabla \cdot \left( \frac{1}{B_w} \vec{u}_w \right) = 0$$

$$\eta_k E_k + P_k)] =$$

$$A(P_k - P_i) + A \lambda_u \bar{u}_{int} \cdot (\mathbf{u}_i - \mathbf{u}_k)$$

$$\frac{\partial}{\partial t} \left[ \phi \left( \frac{R_S S_o}{B_o} + \frac{S_g}{B_g} \right) \right] + \nabla \cdot \left( \frac{R_S}{B_o} \vec{u}_o + \frac{1}{B_g} \vec{u}_g \right) = 0$$

## Black Oil Equations

- The ultimate solution: design “differentiable” implicit operators.

## Example

- Consider a function  $f : x \rightarrow y$ , which is implicitly defined by

$$F(x, y) = x^3 - (y^3 + y) = 0$$

If not using the cubic formula for finding the roots, the forward computation consists of iterative algorithms, such as the Newton's method and bisection method

```
y0 ← 0  
k ← 0  
while |F(x, yk)| > ε do  
    δk ← F(x, yk)/F'y(x, yk)  
    yk+1 ← yk - δk  
    k ← k + 1  
end while  
Return yk
```

```
I ← -M, r ← M, m ← 0  
while |F(x, m)| > ε do  
    c ←  $\frac{a+b}{2}$   
    if F(x, m) > 0 then  
        a ← m  
    else  
        b ← m  
    end if  
end while  
Return c
```

## Example

- An efficient way is to apply the **implicit function theorem**. For our example,  $F(x, y) = x^3 - (y^3 + y) = 0$ , treat  $y$  as a function of  $x$  and take the derivative on both sides

$$3x^2 - 3y(x)^2y'(x) - 1 = 0 \Rightarrow y'(x) = \frac{3x^2 - 1}{3y(x)^2}$$

The above gradient is **exact**.

**Can we apply the same idea to inverse modeling?**

# Physics Constrained Learning

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

- Assume that we solve for  $u_h = G_h(\theta)$  with  $F_h(\theta, u_h) = 0$ , and then

$$\tilde{L}_h(\theta) = L_h(G_h(\theta))$$

- Applying the **implicit function theorem**

$$\frac{\partial F_h(\theta, u_h)}{\partial \theta} + \frac{\partial F_h(\theta, u_h)}{\partial u_h} \frac{\partial G_h(\theta)}{\partial \theta} = 0 \Rightarrow \frac{\partial G_h(\theta)}{\partial \theta} = - \left( \frac{\partial F_h(\theta, u_h)}{\partial u_h} \right)^{-1} \frac{\partial F_h(\theta, u_h)}{\partial \theta}$$

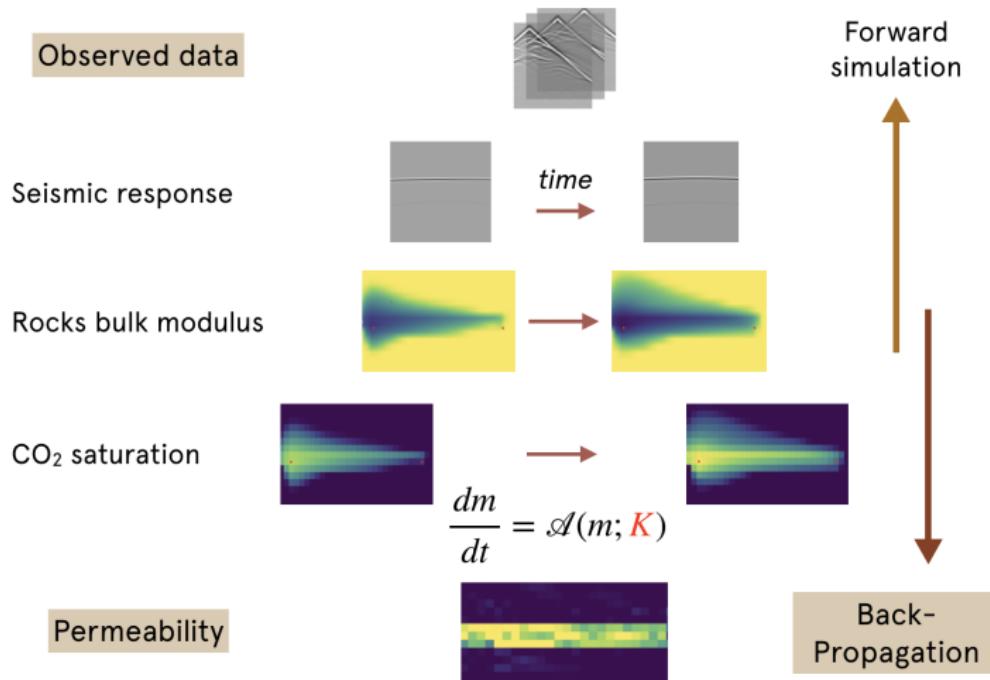
- Finally we have

$$\frac{\partial \tilde{L}_h(\theta)}{\partial \theta} = \frac{\partial L_h(u_h)}{\partial u_h} \frac{\partial G_h(\theta)}{\partial \theta} = - \frac{\partial L_h(u_h)}{\partial u_h} \left( \frac{\partial F_h(\theta, u_h)}{\partial u_h} \Big|_{u_h=G_h(\theta)} \right)^{-1} \frac{\partial F_h(\theta, u_h)}{\partial \theta} \Big|_{u_h=G_h(\theta)}$$

# Outline

- 1 Inverse Modeling
- 2 Automatic Differentiation
- 3 Physics Constrained Learning
- 4 Applications
- 5 ADCME: Scientific Machine Learning for Inverse Modeling

# Parameter Inverse Problem: Elastic Full Waveform Inversion for Subsurface Flow Problems



# Fully Nonlinear Implicit Schemes

- The governing equation is a nonlinear PDE

$$\frac{\partial}{\partial t}(\phi S_i \rho_i) + \nabla \cdot (\rho_i \mathbf{v}_i) = \rho_i q_i, \quad i = 1, 2$$

$$S_1 + S_2 = 1$$

$$\mathbf{v}_i = -\frac{K k_{ri}}{\tilde{\mu}_i} (\nabla P_i - g \rho_i \nabla Z), \quad i = 1, 2$$

$$k_{r1}(S_1) = \frac{k_{r1}^o S_1^{L_1}}{S_1^{L_1} + E_1 S_2^{T_1}}$$

$$k_{r2}(S_1) = \frac{S_2^{L_2}}{S_2^{L_2} + E_2 S_1^{T_2}}$$

$$\rho \frac{\partial v_z}{\partial t} = \frac{\partial \sigma_{zz}}{\partial z} + \frac{\partial \sigma_{xz}}{\partial x}$$

$$\rho \frac{\partial v_x}{\partial t} = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xz}}{\partial z}$$

$$\frac{\partial \sigma_{zz}}{\partial t} = (\lambda + 2\mu) \frac{\partial v_z}{\partial z} + \lambda \frac{\partial v_x}{\partial x}$$

$$\frac{\partial \sigma_{xx}}{\partial t} = (\lambda + 2\mu) \frac{\partial v_x}{\partial x} + \lambda \frac{\partial v_z}{\partial z}$$

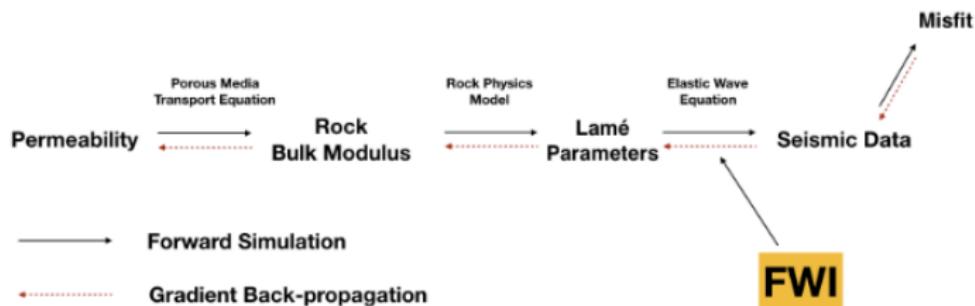
$$\frac{\partial \sigma_{xz}}{\partial t} = \mu \left( \frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z} \right),$$

- For stability and efficiency, implicit methods are the industrial standards.

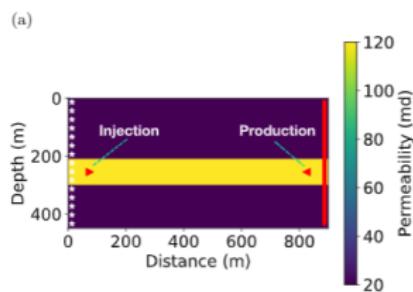
$$\phi(S_2^{n+1} - S_2^n) - \nabla \cdot (m_2(S_2^{n+1}) K \nabla \Psi_2^n) \Delta t = \left( q_2^n + q_1^n \frac{m_2(S_2^{n+1})}{m_1(S_2^{n+1})} \right) \Delta t \quad m_i(s) = \frac{k_{ri}(s)}{\tilde{\mu}_i}$$

# Inverse Modeling Workflow

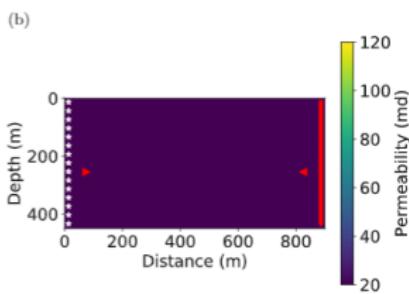
Traditionally, the inversion is typically solved by separately inverting the wave equation (FWI) and the flow transport equations.



(a)

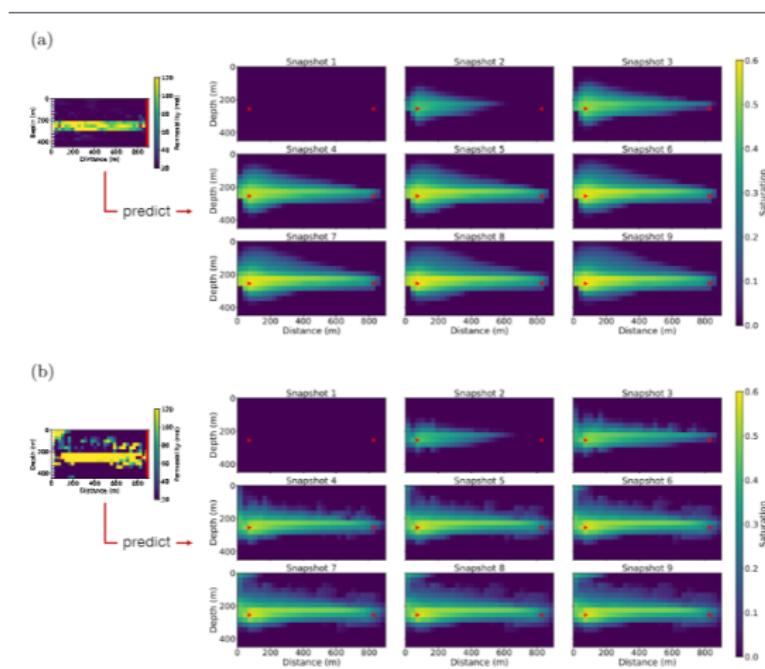


(b)



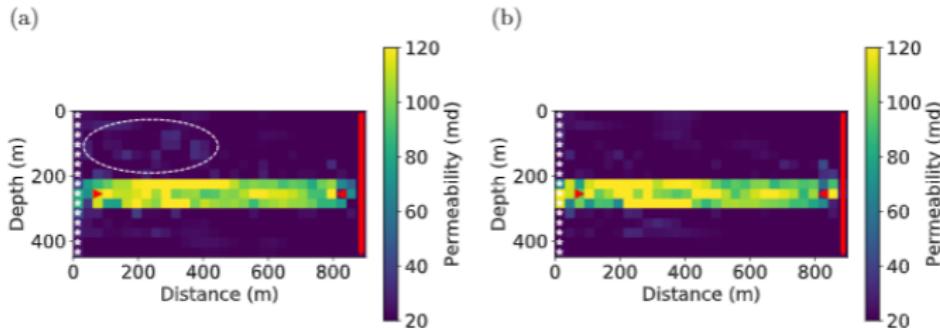
# Coupled Inversion vs. Decoupled Inversion

We found that coupled inversion reduces the artifacts from FWI significantly and yields a substantially better results.



# Travel Time vs. Full Waveforms

We also compared using only travel time (left, Eikonal equation) versus using full waveforms (right, FWI) for inversion. We found that **full waveforms do contain more information for making a better estimation of the permeability property.**



The Eikonal equation solver was also implemented with physics constrained learning!

Check out our package FwiFlow.jl for wave and flow inversion and our recently published paper for this work.

 [lidongzh / FwiFlow.jl](#)

Elastic Full Waveform Inversion for subsurface flow problems with intrusive automatic differentiation

 MIT License

 9 stars  2 forks

 Unstar 

**Water Resources Research**

Research Article |  Full Access | 

Coupled Time-lapse Full Waveform Inversion for Subsurface Flow Problems using Intrusive Automatic Differentiation

Dongzhuo Li, Kailai Xu, Jerry M. Harris, Eric Darve

First published: 06 July 2020 | <https://doi.org.stanford.idm.oclc.org/10.1029/2019WR027032>

Find it @ Stanford

Dongzhuo Li and Kailai Xu contributed equally to this work.  
This article has been accepted for publication and undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1029/2019WR027032



## High Performance

Solves inverse modeling problems faster with our GPU-accelerated FWI module.

## Designed for Subsurface Modeling

Provides many operators that can be reused for different subsurface modeling problems.

## Easy to Extend

Allows users to implement and insert their own custom operators and solve new problems.

# Function Inverse Problem: Modeling Viscoelasticity

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

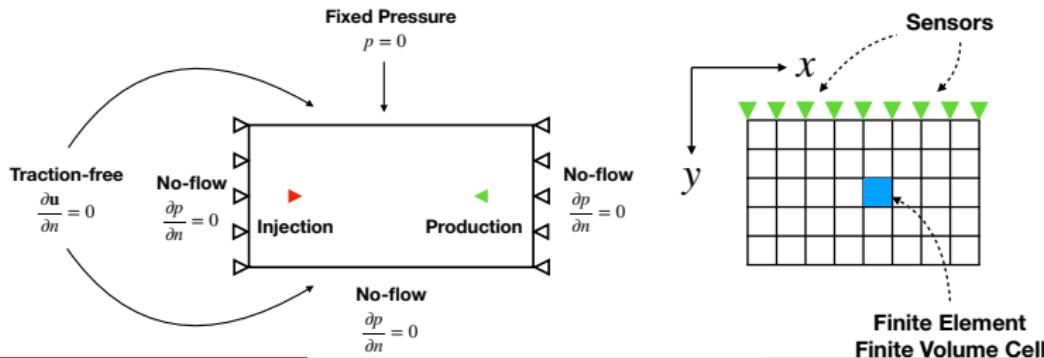
$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{u}) - b\nabla p = 0$$

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

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

- Approximate the constitutive relation by a neural network

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



# Neural Networks: Inverse Modeling of Viscoelasticity

- We propose the following form for modeling viscosity (assume the time step size is fixed):

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

- $H$  is a free optimizable **symmetric positive definite matrix** (SPD). Hence the numerical stiffness matrix is SPD.
- Implicit linear equation

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

- Linear system to solve in each time step  $\Rightarrow$  good balance between **numerical stability** and **computational cost**.
- Good performance in our numerical examples.

# Training Strategy and Numerical Stability

- Physics constrained learning = improved numerical stability in predictive modeling.
- For simplicity, consider two strategies to train an NN-based constitutive relation using direct data  $\{(\epsilon_o^n, \sigma_o^n)\}_n$

$$\Delta\sigma^n = H\Delta\epsilon^n + \mathcal{NN}_{\theta}(\sigma^n, \epsilon^n), \quad H \succ 0$$

- Training with input-output pairs

$$\min_{\theta} \sum_n \left( \sigma_o^{n+1} - (H\epsilon_o^{n+1} + \mathcal{NN}_{\theta}^*(\sigma_o^n, \epsilon_o^n)) \right)^2$$

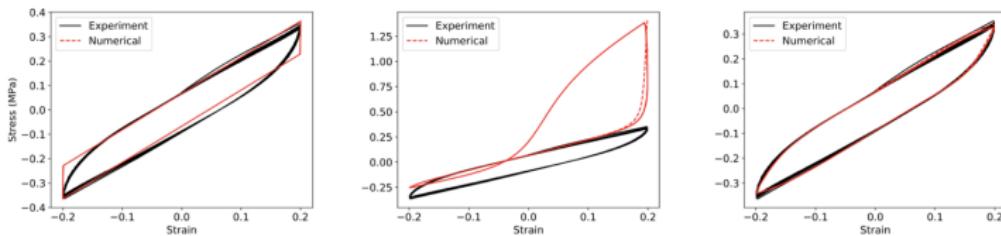
- Better stability using training on trajectory = **physics constrained learning**

$$\min_{\theta} \sum_n (\sigma^n(\theta) - \sigma_o^n)^2$$

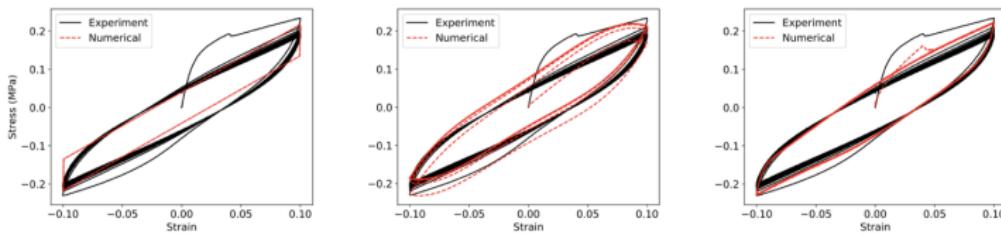
s.t. I.C.  $\sigma^1 = \sigma_o^1$  and time integrator  $\Delta\sigma^n = H\Delta\epsilon^n + \mathcal{NN}_{\theta}(\sigma^n, \epsilon^n)$

# Experimental Data

Dataset 1



Dataset 2



Kevin-Voigt Model

Trained with  
Input-output Pairs

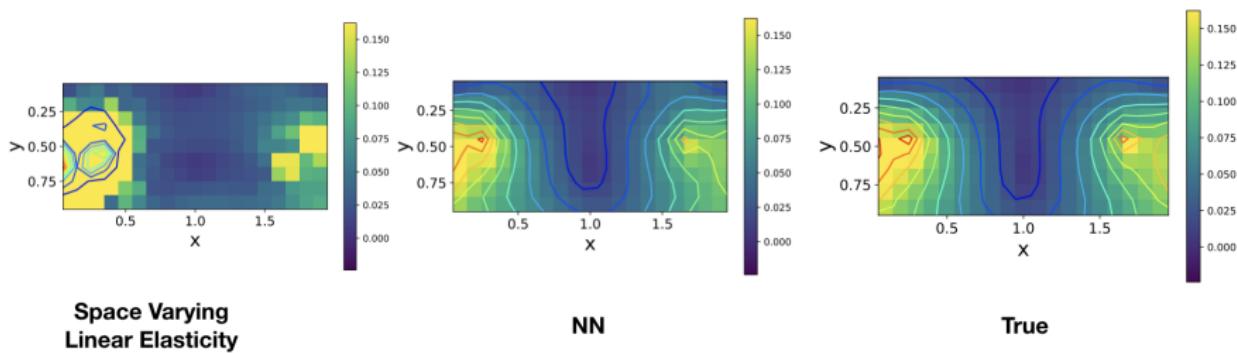
Trained with  
Physics Constrained Learning

Experimental data from: Javidan, Mohammad Mahdi, and Jinkoo Kim. "Experimental and numerical Sensitivity Assessment of Viscoelasticity for polymer composite Materials." Scientific Reports 10.1 (2020): 1–9.

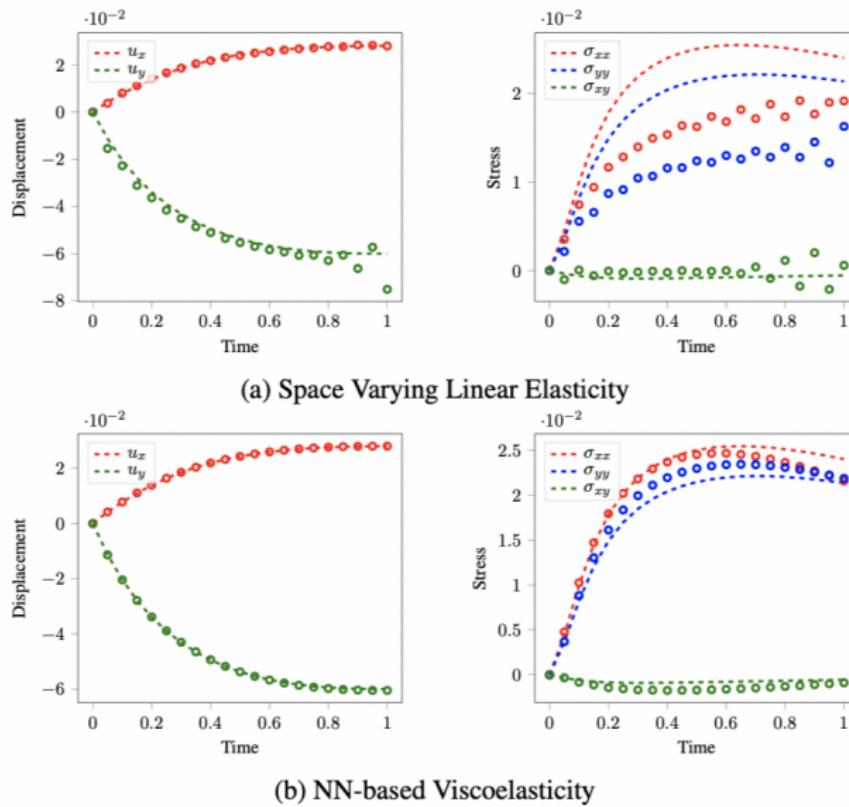
# Inverse Modeling of Viscoelasticity

- Comparison with space varying linear elasticity approximation

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



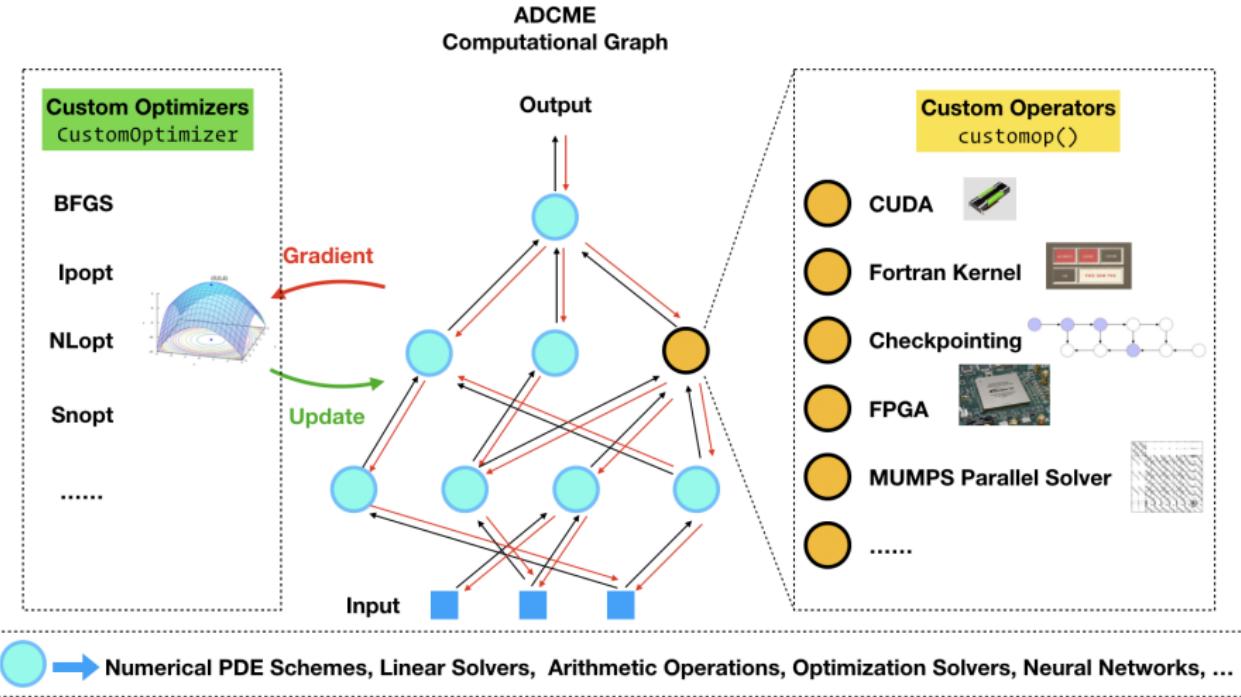
# Inverse Modeling of Viscoelasticity



# Outline

- 1 Inverse Modeling
- 2 Automatic Differentiation
- 3 Physics Constrained Learning
- 4 Applications
- 5 ADCME: Scientific Machine Learning for Inverse Modeling

# Physical Simulation as a Computational Graph



# A General Approach to Inverse Modeling

