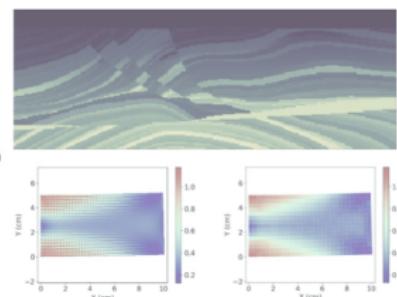
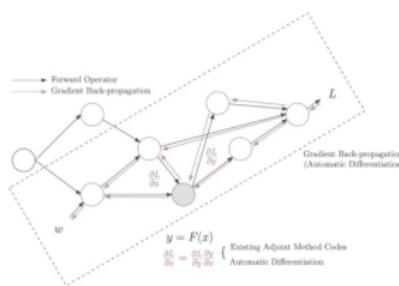
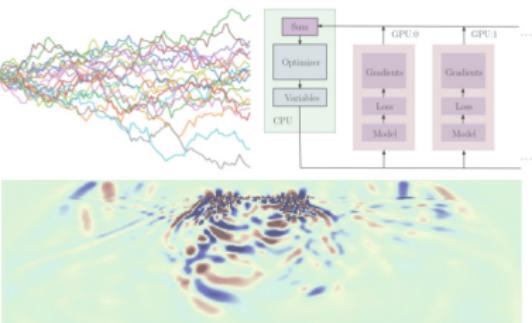


Machine Learning for Inverse Problems in Computational Engineering

Kailai Xu, and Eric Darve

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



Outline

1 Inverse Modeling

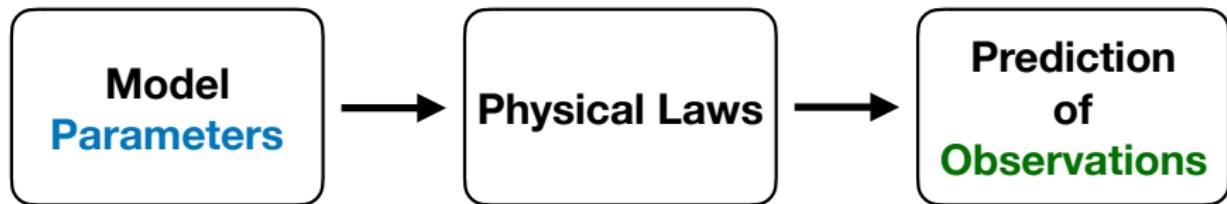
2 Automatic Differentiation

3 Code Example

4 Applications

Inverse Modeling

Forward Problem



Inverse Problem



Inverse Modeling

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_{obs} , e.g., $L_h(u_h) = \|u_h - u_{\text{obs}}\|_2^2$.
- θ 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

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

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.
- ...

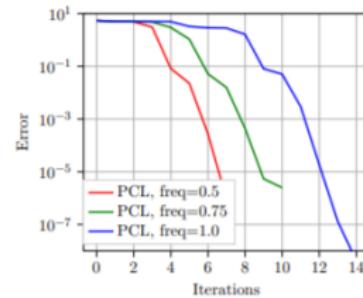
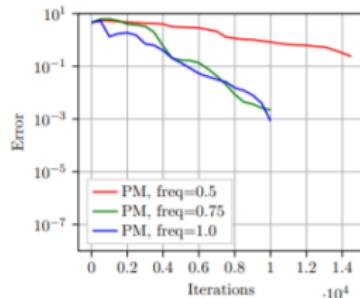
The candidate solution space is **infinite dimensional**.

Penalty Methods

- Parametrize f with f_θ and incorporate the physical constraint as a **penalty term** (regularization, prior, ...) in the loss function.

$$\min_{\theta, u_h} L_h(u_h) + \lambda \|F_h(f_\theta, u_h)\|_2^2$$

- May not satisfy physical constraint $F_h(f_\theta, u_h) = 0$ accurately;
- Slow convergence for **stiff** problems;

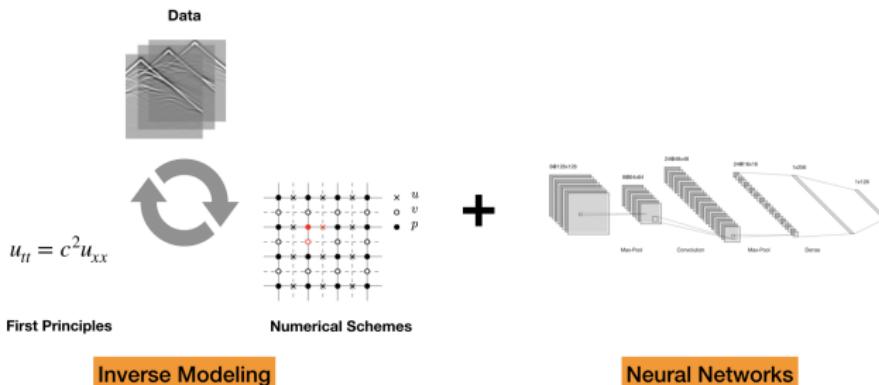


- High dimensional optimization problem; both θ and u_h are variables.

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.

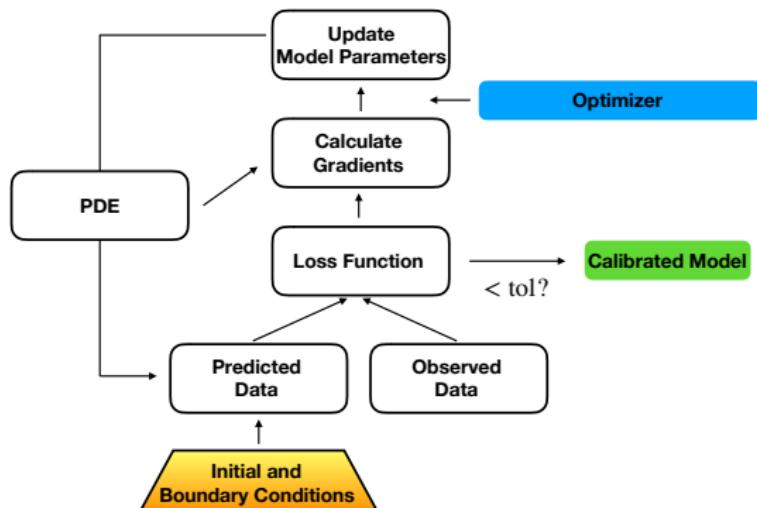


Gradient Based Optimization

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

- 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

||

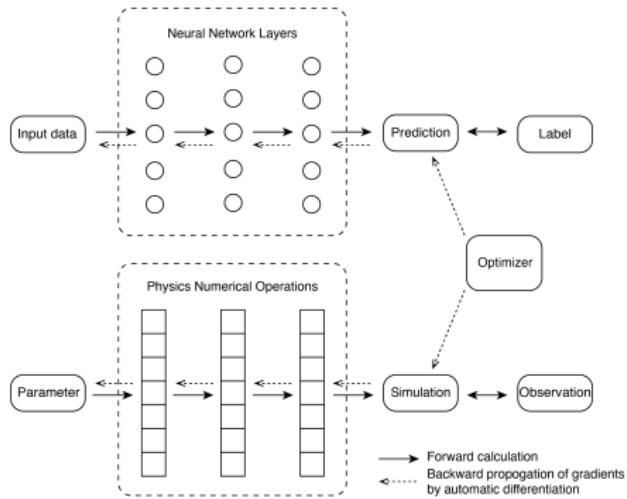
Reverse-mode

Automatic Differentiation

||

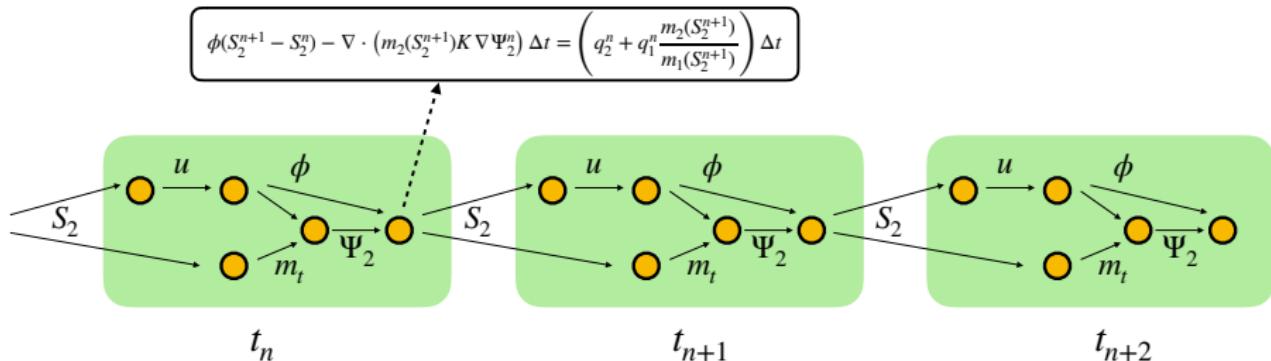
Discrete

Adjoint-State Method

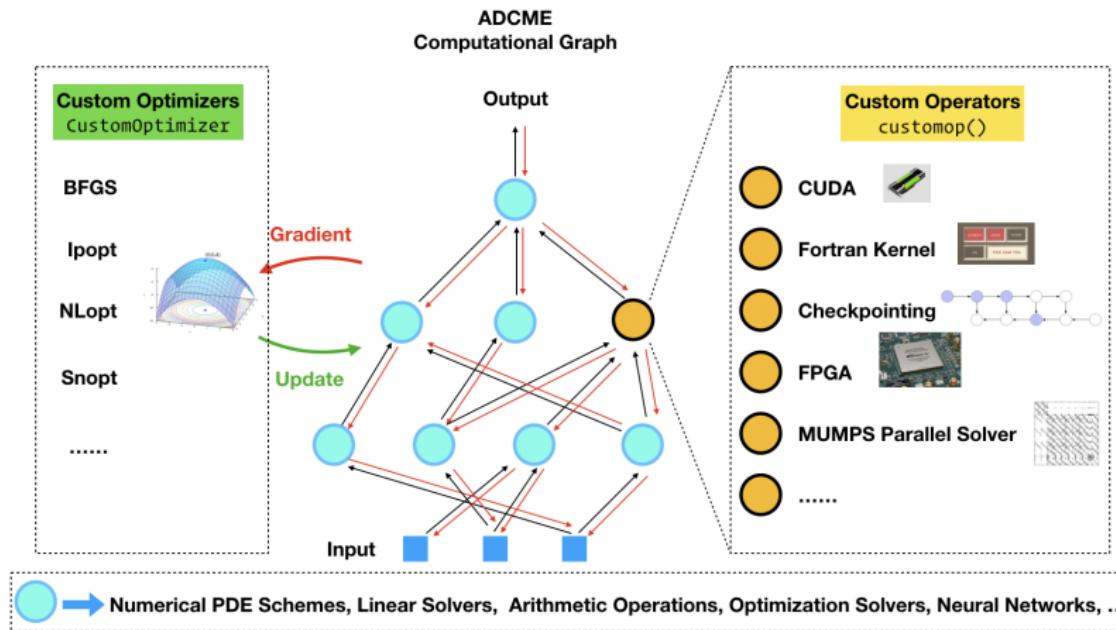


Computational Graph for Numerical Schemes

- 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.

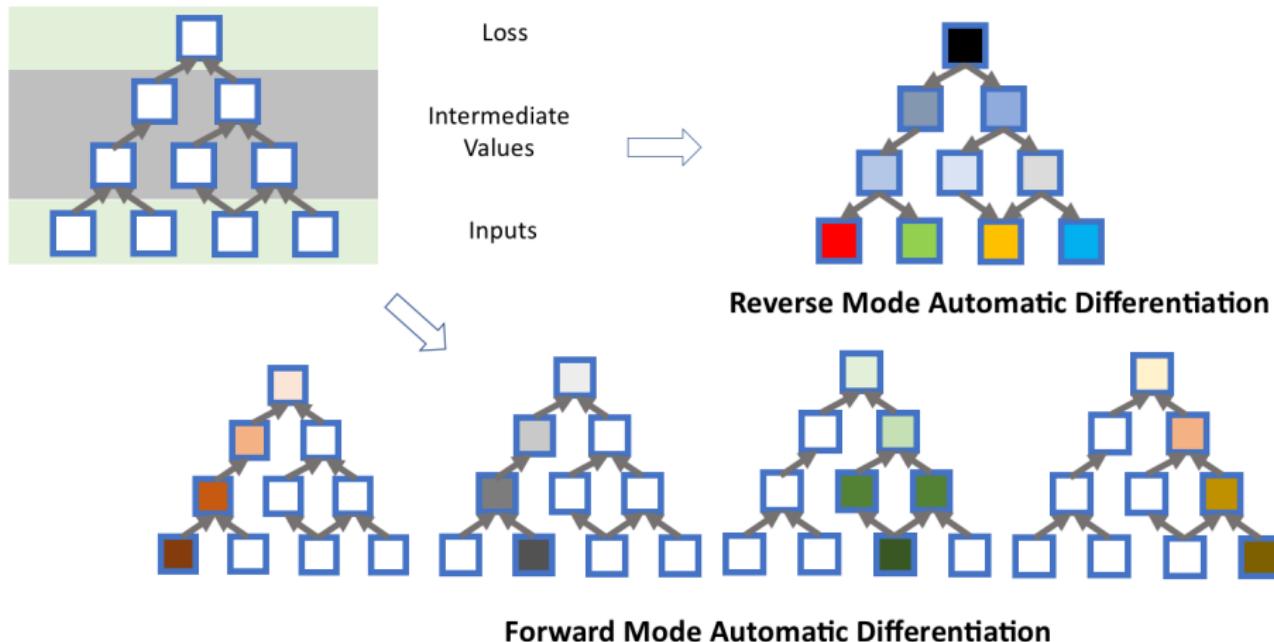


ADCME: Computational-Graph-based Numerical Simulation



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

Automatic Differentiation: Forward-mode and Reverse-mode



What is the Appropriate Model for Inverse Problems?

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

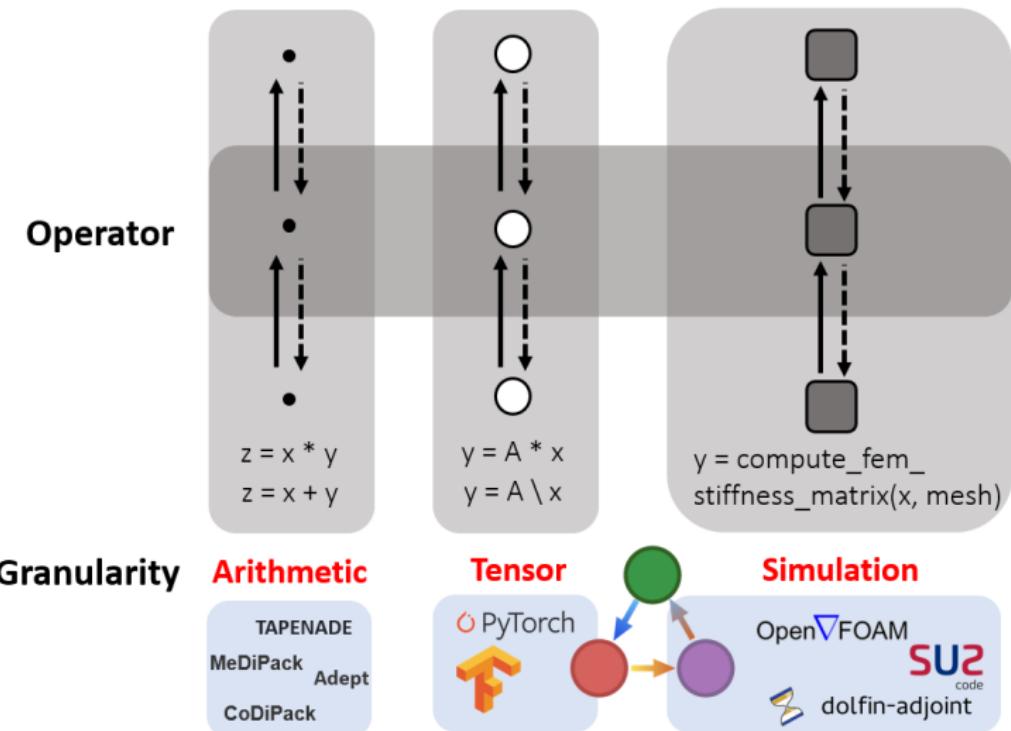
| Mode | Suitable for ... | Complexity ¹ | Application |
|---------|------------------|------------------------------|------------------|
| Forward | $m \gg n$ | $\leq 2.5 \text{ OPS}(f(x))$ | UQ |
| Reverse | $m \ll n$ | $\leq 4 \text{ OPS}(f(x))$ | Inverse Modeling |

- 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.

¹OPS is a metric for complexity in terms of fused-multiply adds.

Granularity of Automatic Differentiation



Outline

1 Inverse Modeling

2 Automatic Differentiation

3 Code Example

4 Applications

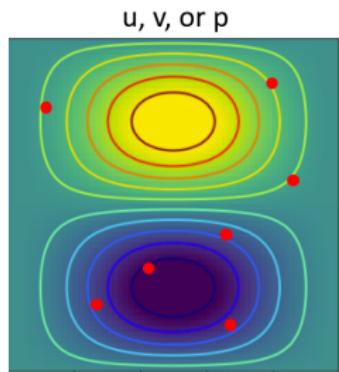
Inverse Modeling of the Stokes Equation

- The governing equation for the Stokes problem

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

- The weak form is given by

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



Inverse Modeling of the Stokes Equation

```
nu = squeeze(fc(gauss_nodes(m, n, h), [20,20,20,1]))  
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

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

Outline

1 Inverse Modeling

2 Automatic Differentiation

3 Code Example

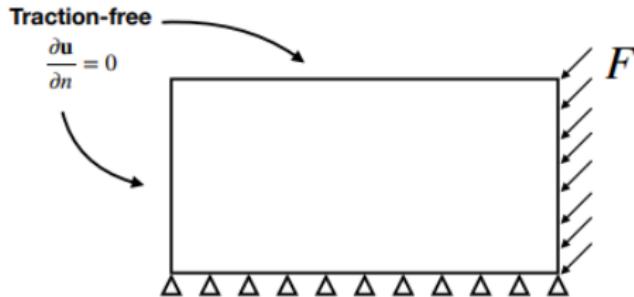
4 Applications

Linear Poroelasticity

- The governing equation for linear poroelasticity with a spatially-varying viscosity coefficient

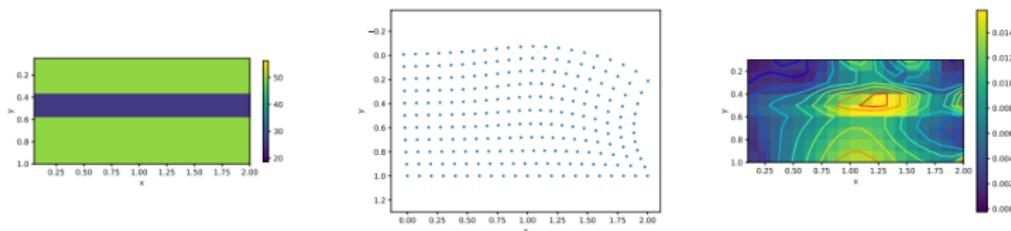
$$\operatorname{div} \boldsymbol{\sigma} + \rho g = \ddot{\mathbf{u}}$$

$$\dot{\sigma}_{ij} + \frac{\mu}{\eta(y)} \left(\sigma_{ij} - \frac{\sigma_{kk}}{3} \delta_{ij} \right) = 2\mu \dot{\epsilon}_{ij} + \lambda \dot{\epsilon}_{kk} \delta_{ij}$$

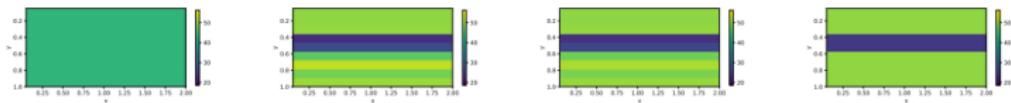


Linear Poroelasticity

- The true model of $\eta(y)^{-1}$, the displacement at the terminal time, and the von Mises stress distribution at the terminal time.



- Evolution of learned $\eta(y)^{-1}$ at iteration 0, 80, 160, and 240.



Viscoelasticity

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

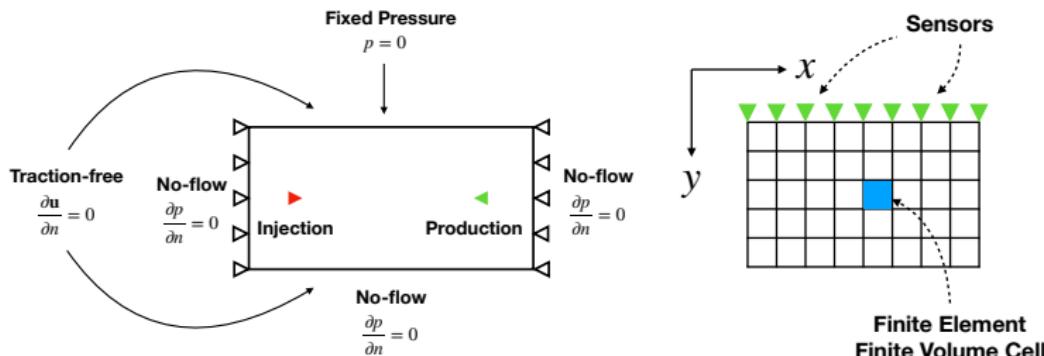
$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{u}) - b\nabla p = 0$$

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

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

- Approximate the constitutive relation by a neural network

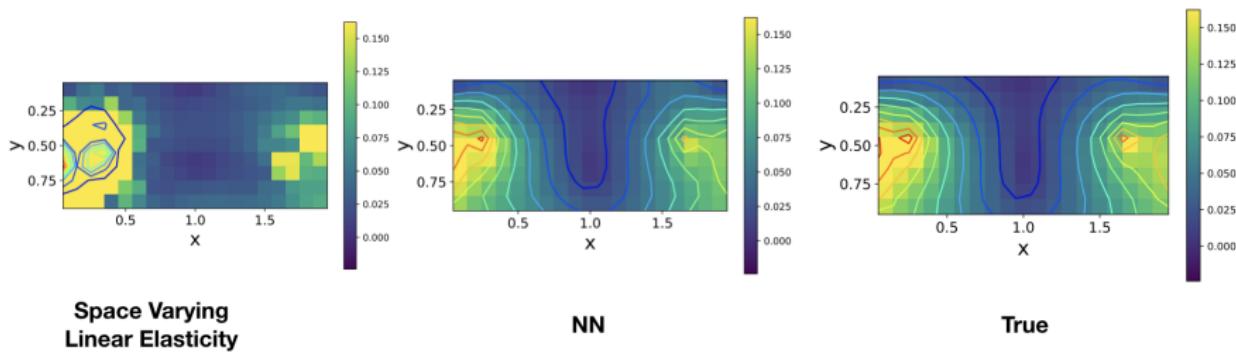
$$\boldsymbol{\sigma}^{n+1} = \mathcal{NN}_{\theta}(\boldsymbol{\sigma}^n, \boldsymbol{\epsilon}^n) + H\boldsymbol{\epsilon}^{n+1}$$



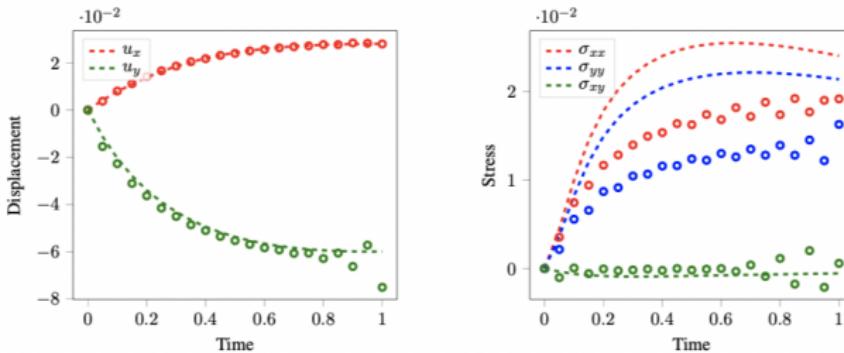
Viscoelasticity

- Comparison with space varying linear elasticity approximation

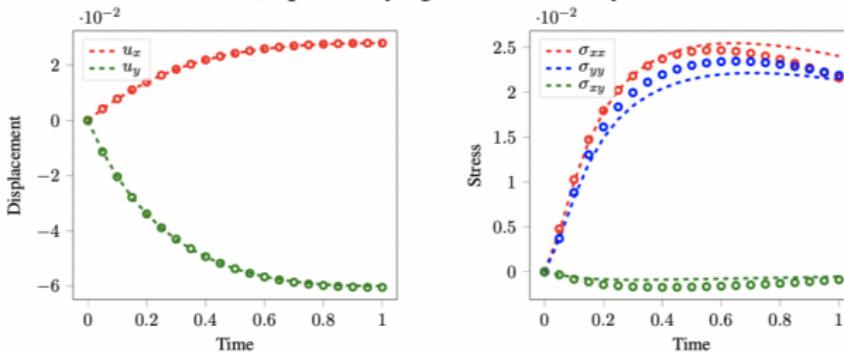
$$\boldsymbol{\sigma} = H(x, y)\boldsymbol{\epsilon}$$



Viscoelasticity



(a) Space Varying Linear Elasticity



(b) NN-based Viscoelasticity

Navier-Stokes Equation

- Steady-state Navier-Stokes equation

$$(u \cdot \nabla) u = -\frac{1}{\rho} \nabla p + \nabla \cdot (\nu \nabla u) + g$$
$$\nabla \cdot u = 0$$

- Inverse problem are ubiquitous in fluid dynamics:

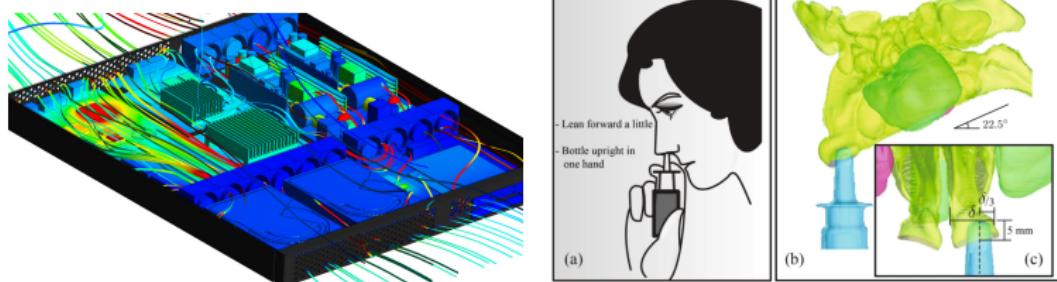
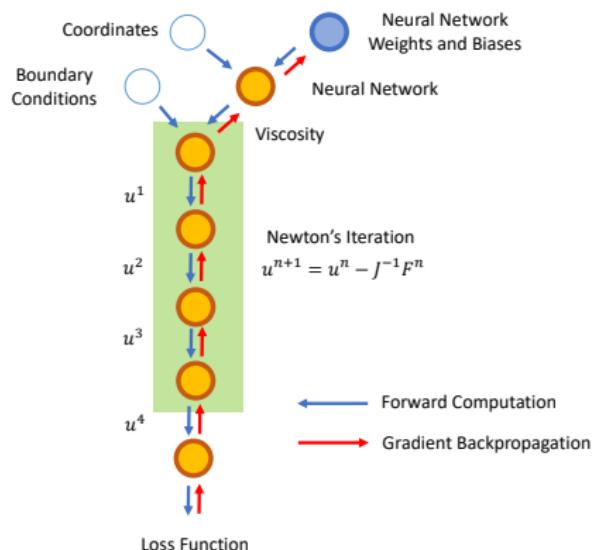
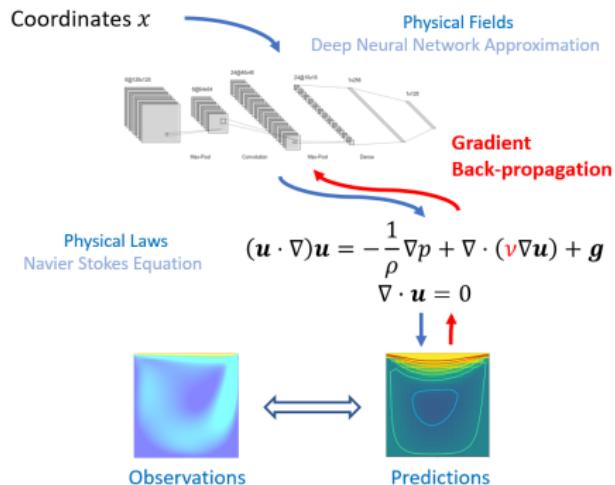


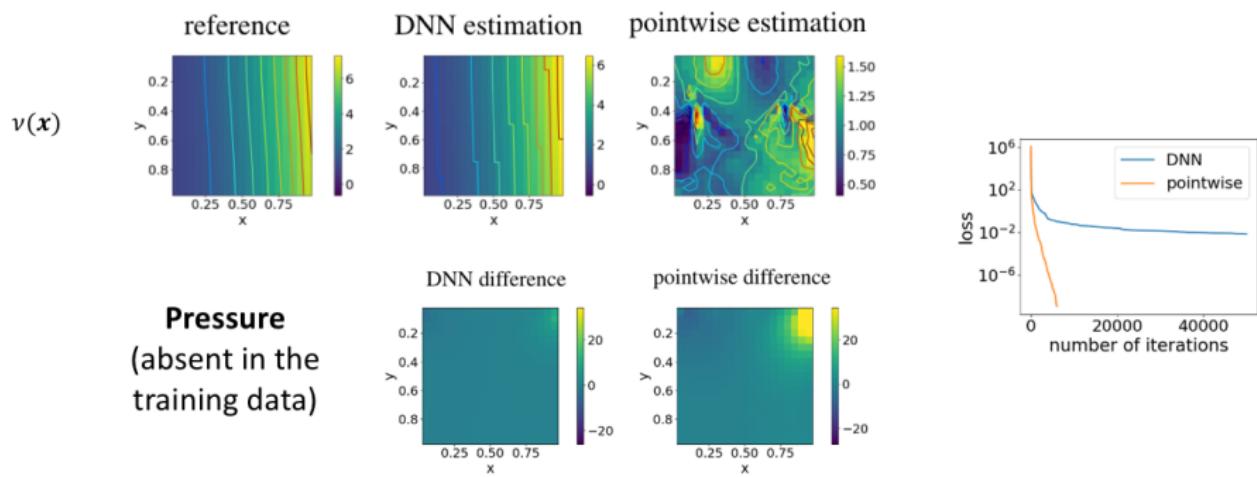
Figure: Left: electronic cooling; right: nasal drug delivery.

Navier-Stokes Equation



Navier-Stokes Equation

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



A General Approach to Inverse Modeling

