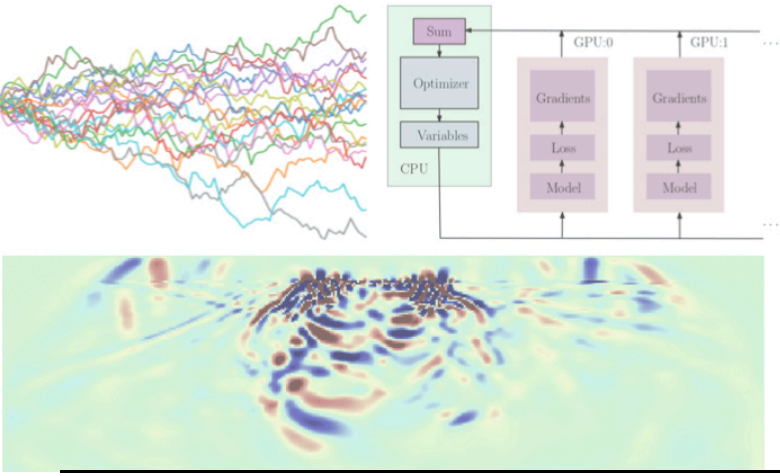
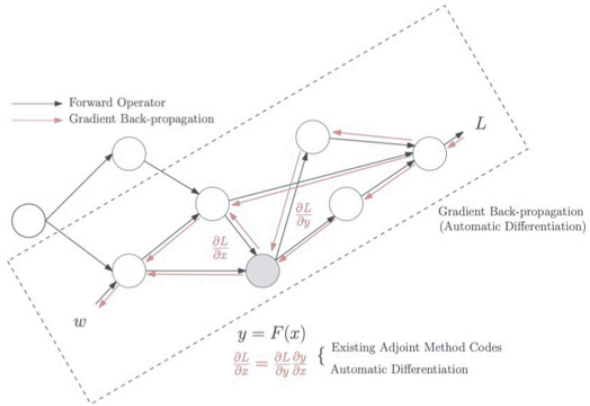
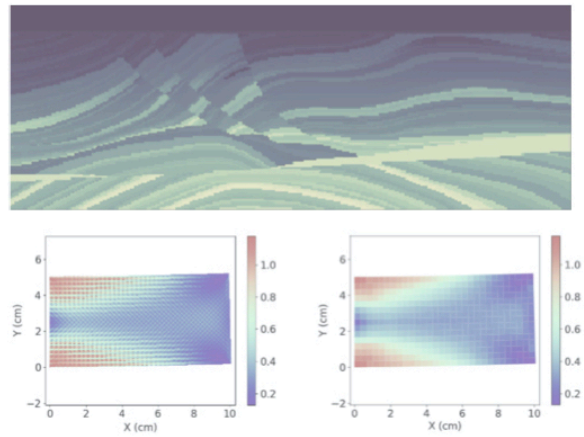
Data-driven Inverse Modeling with Incomplete Observation

Kailai Xu and Eric Darve
Alexandre M. Tartakovsky, Jeff Burghardt, Dongzhuo Li, Jerry M. Harris, Weiqiang Zhu, Gregory C. Beroza





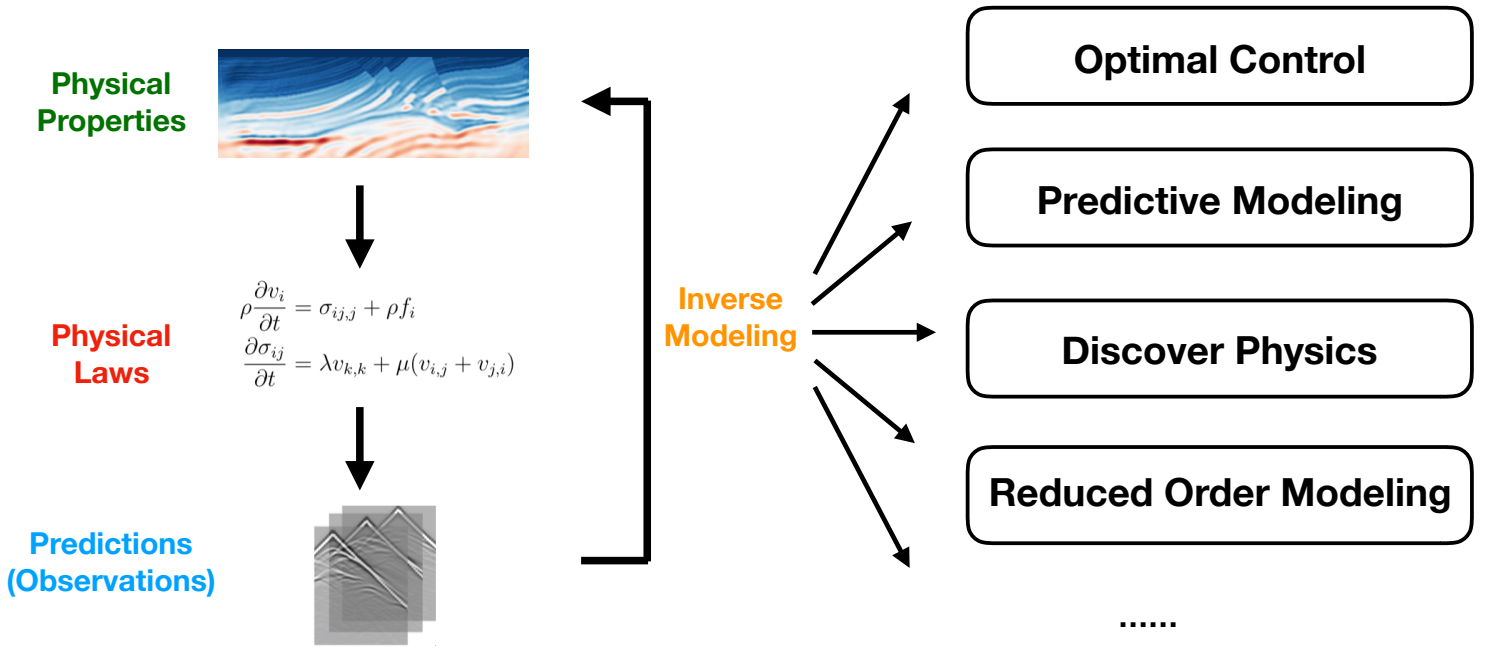


Full version: `https://kailaix.github.io/ADCME.jl/dev/assets/Slide/ADCME.pdf`

Outline

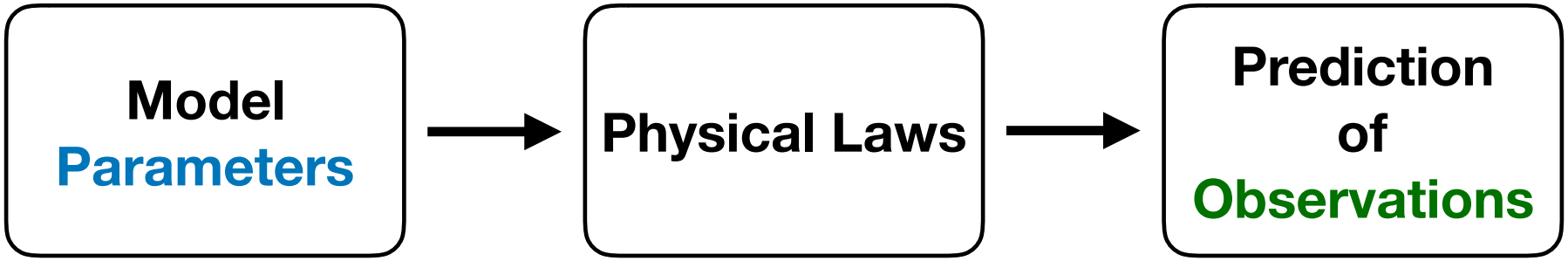
Inverse Modeling

- **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

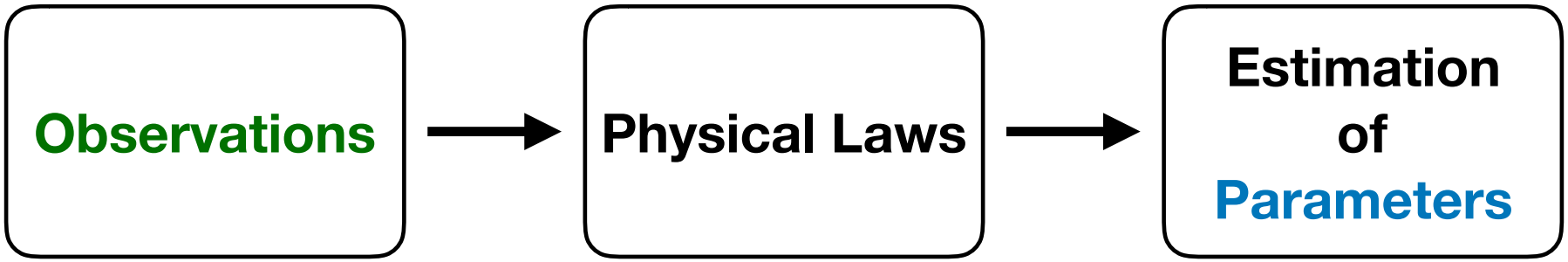


Inverse Modeling

Forward Problem



Inverse Problem



Inverse Modeling

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

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

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

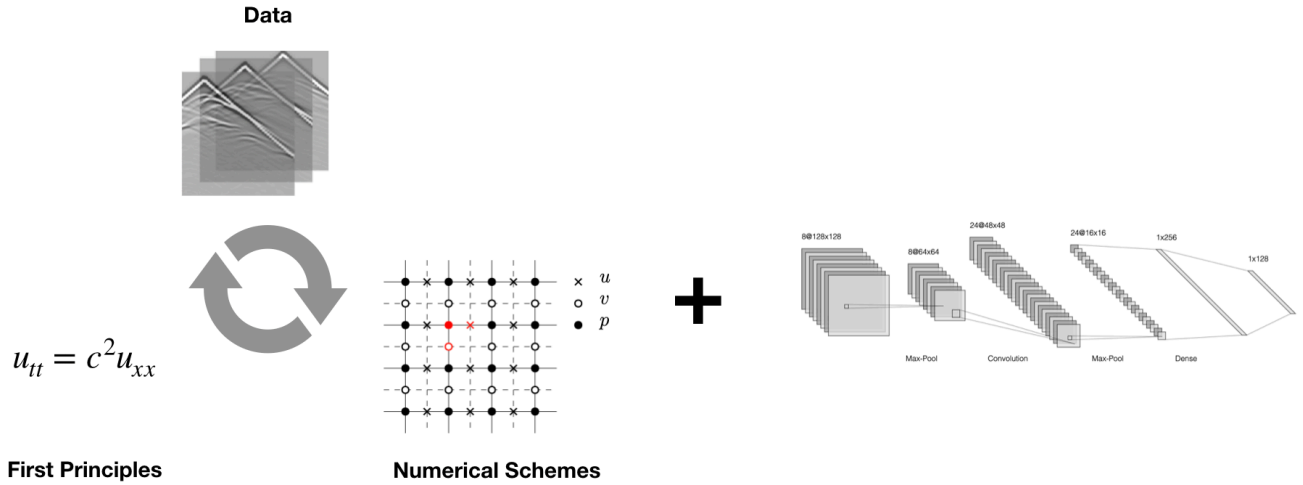
- Koopman operator in dynamical systems.
- Constitutive relations in solid mechanics.
- Turbulent closure relations in fluid mechanics.
- ...

The candidate solution space is infinite dimensional.

Physics Based Machine Learning

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

- 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.

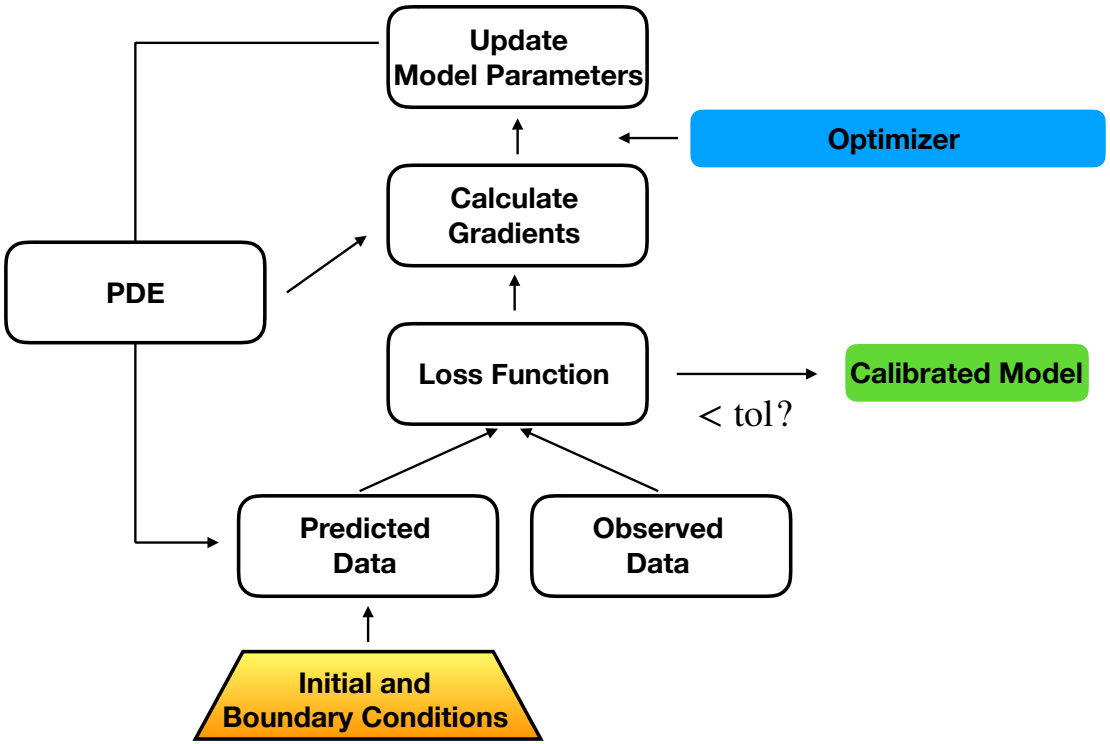


Gradient Based Optimization

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

- 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

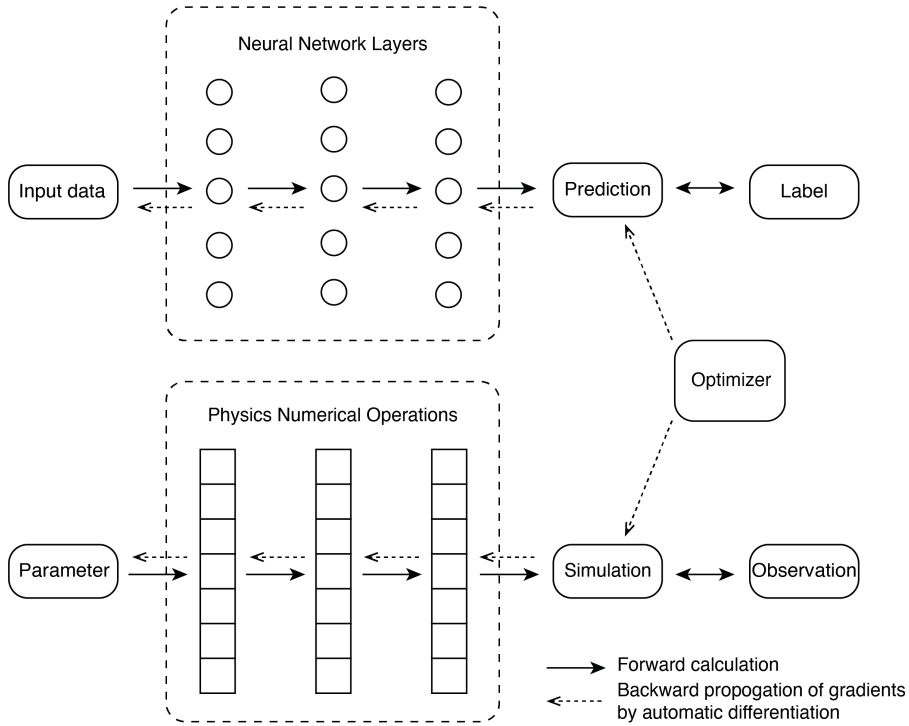
Automatic Differentiation

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.

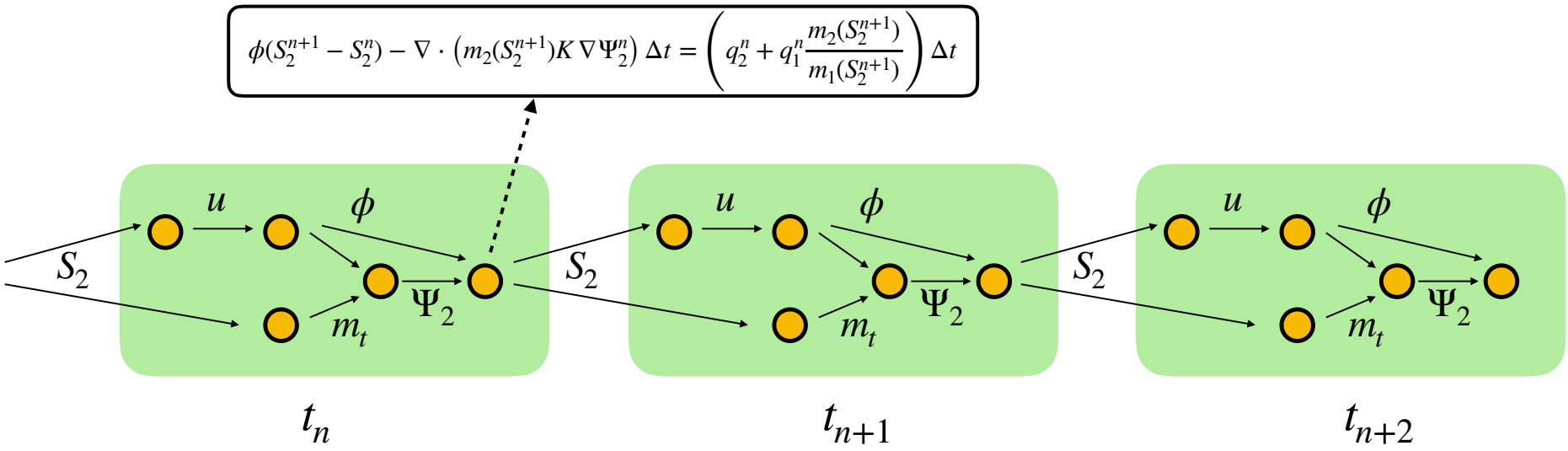
Mathematical Fact

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



Computational Graph for Numerical Schemes

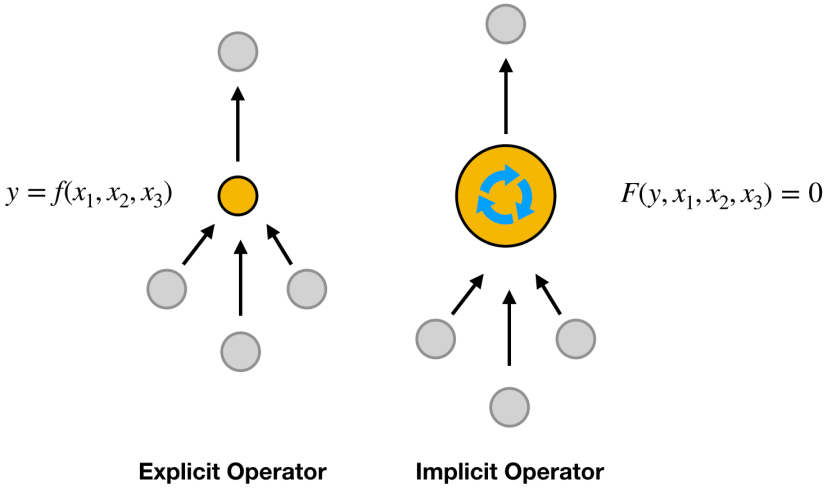
- To leverage automatic differentiation for inverse modeling, we need to express the numerical schemes in the "AD language": computational graph.
- No matter how complicated a numerical scheme is, it can be decomposed into a collection of operators that are interlinked via state variable dependencies.



Outline

Challenges in AD

- Most AD frameworks only deal with explicit operators, i.e., the functions that has analytical derivatives, or composition of these functions.
- Many scientific computing algorithms are iterative or implicit in nature.



Linear/Nonlinear	Explicit/Implicit	Expression
Linear	Explicit	$y = Ax$
Nonlinear	Explicit	$y = F(x)$
Linear	**Implicit**	$Ax = y$
Nonlinear	**Implicit**	$F(x, y) = 0$

Example

- Consider a function $f : x \to y$, 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

$y^0 \leftarrow 0$
$k \leftarrow 0$
while $|F(x, y^k)| > \epsilon$ **do**
  $\delta^k \leftarrow F(x, y^k)/F'_y(x, y^k)$
  $y^{k+1} \leftarrow y^k - \delta^k$
  $k \leftarrow k + 1$
end while
Return y^k

$l \leftarrow -M$, $r \leftarrow M$, $m \leftarrow 0$
while $|F(x, m)| > \epsilon$ **do**
  $c \leftarrow \frac{a+b}{2}$
  if $F(x, m) > 0$ **then**
    $a \leftarrow m$
  else
    $b \leftarrow m$
  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

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

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 in the forward computation, we solve for $u_h = G_h(\theta)$ in $F_h(\theta, u_h) = 0$, and then

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

- Applying the implicit function theorem

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

- Finally we have

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

Physics Constrained Learning

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

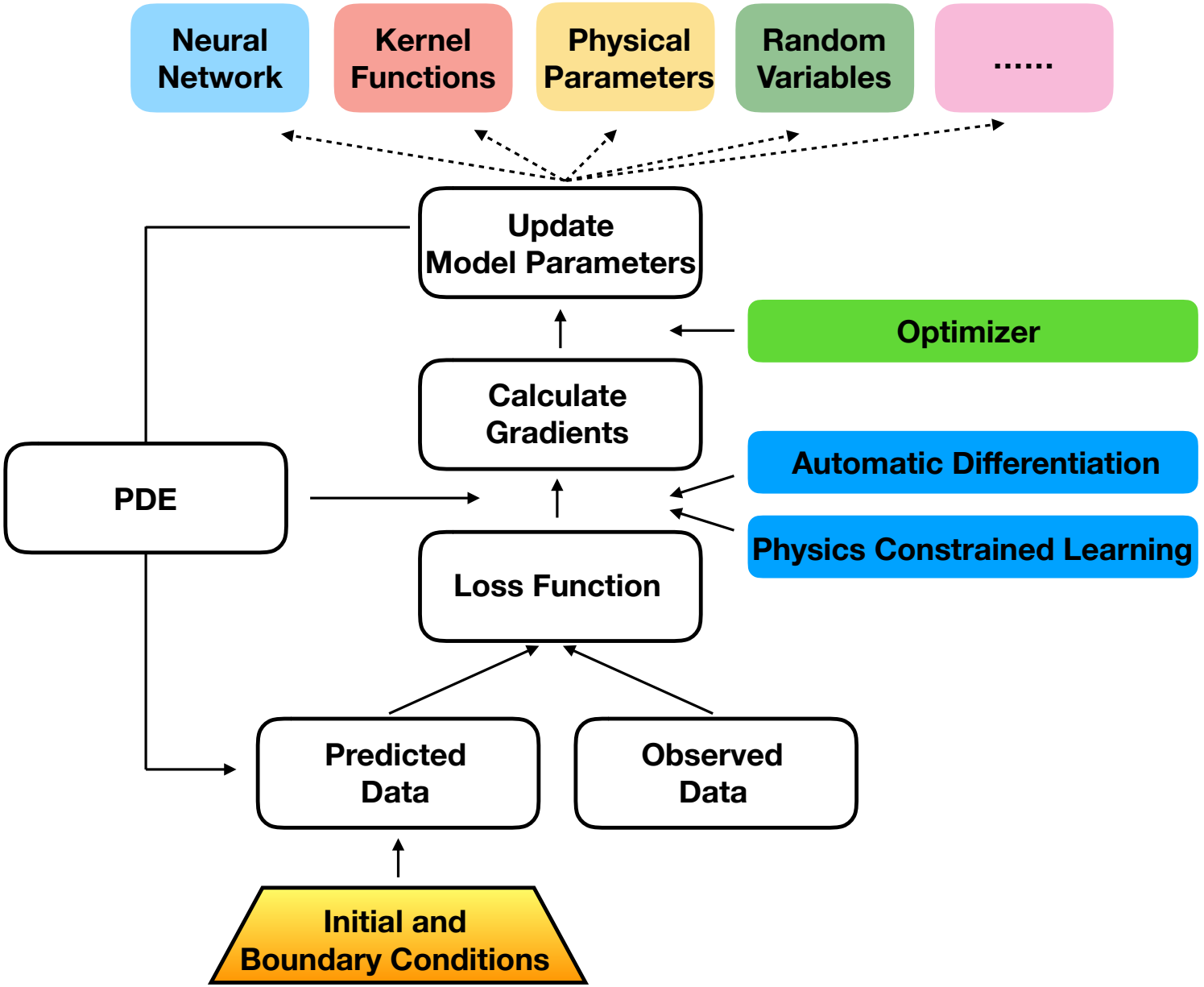
Step 1: Calculate w by solving a linear system (never invert the matrix!)

$$w^T = \underbrace{\frac{\partial L_h(u_h)}{\partial u_h}}_{1\times N}\underbrace{\left(\frac{\partial F_h}{\partial u_h}\Big|_{u_h=G_h(\theta)}\right)^{-1}}_{N\times N}$$

Step 2: Calculate the gradient by automatic differentiation

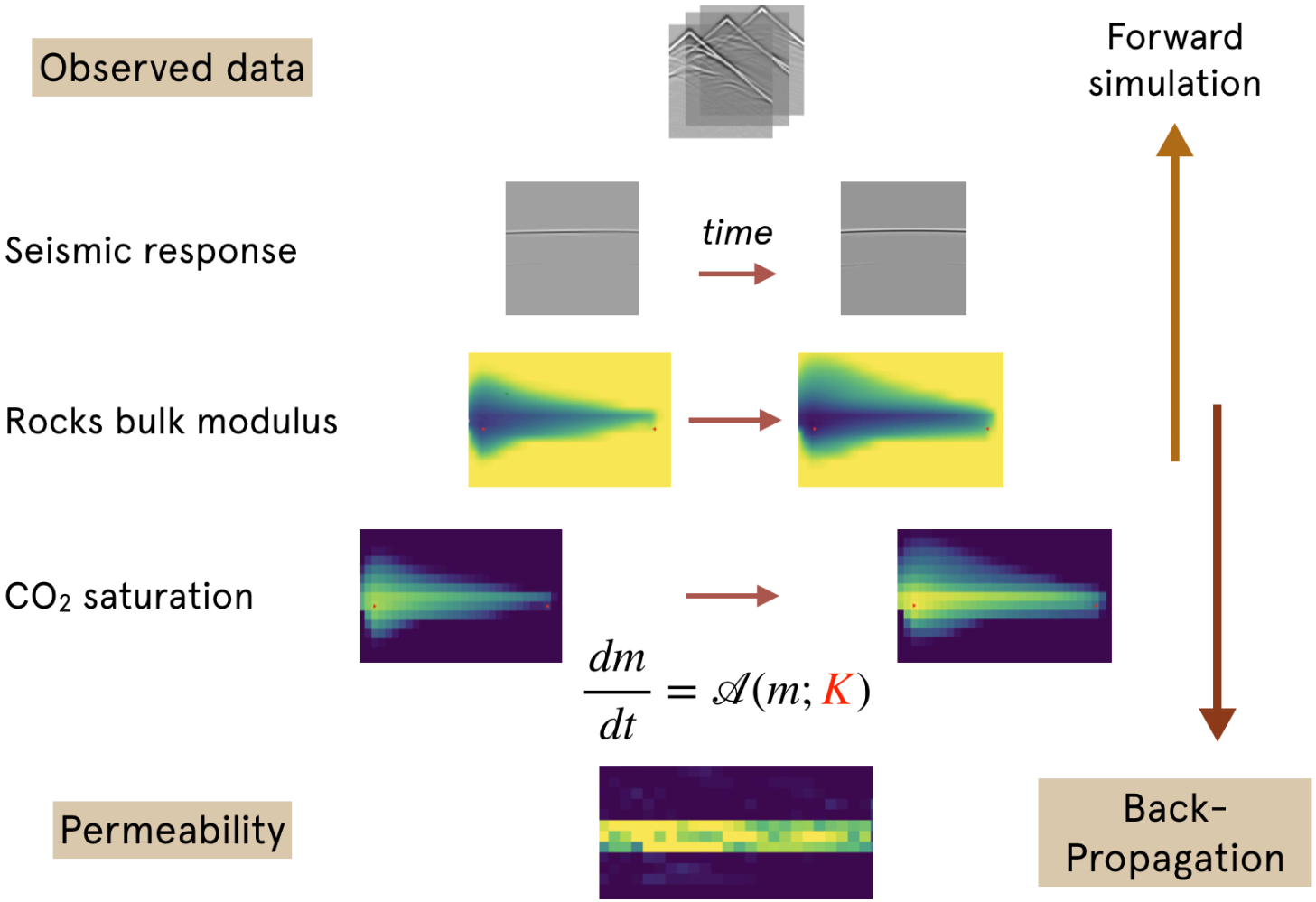
$$w^T\underbrace{\frac{\partial F_h}{\partial \theta}\Big|_{u_h=G_h(\theta)}}_{N\times p} = \frac{\partial (w^T F_h(\theta, u_h))}{\partial \theta}\Big|_{u_h=G_h(\theta)}$$

Methodology Summary



Outline

FwiFlow.jl: Elastic Full Waveform Inversion for Subsurface Flow Problems



FwiFlow.jl: Fully Nonlinear Implicit Schemes

- The governing equation is a nonlinear PDE

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

$$S_1 + S_2 = 1$$

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

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

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

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

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

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

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

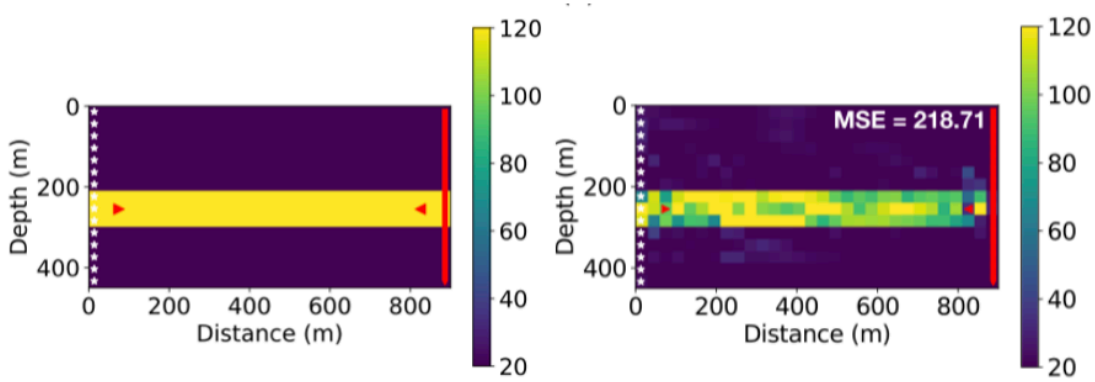
$$\frac{\partial \sigma_{xz}}{\partial t} = \mu\left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z}\right),$$

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

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

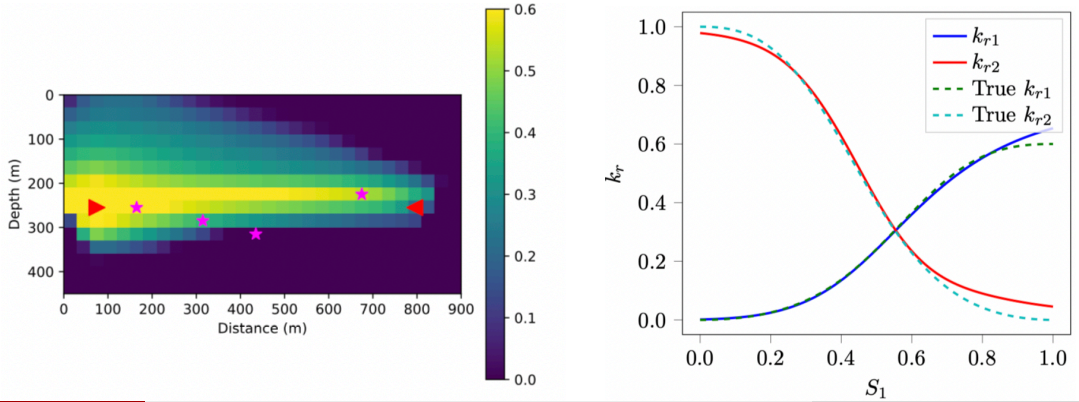
FwiFlow.jl: Showcase

- Task 1: Estimating the permeability from seismic data
  B.C. + Two-Phase Flow Equation + Wave Equation $\Rightarrow$ Seismic Data



- Task 2: Learning the rock physics model from sparse saturation data. The rock physics model is approximated by neural networks

$$f_1(S_1;\theta_1) \approx k_{r1}(S_1) \qquad f_2(S_1;\theta_2) \approx k_{r2}(S_1)$$



FwiFlow.jl: Showcase

- Task 3: Learning the nonlocal (space or time) hidden dynamics from seismic data. This is very challenging using traditional methods (e.g., the adjoint-state method) because the dynamics are history dependent.

B.C. + Time-/Space-fractional PDE + Wave Equation $\Rightarrow$ Seismic Data

Governing Equation	$\sigma = 0$	$\sigma = 5$
${}_0^C D_t^{\mathbf{0.8}} m = 10\Delta m$	$a/a^* = 1.0000$ $\alpha = \mathbf{0.8000}$	$a/a^* = 0.9109$ $\alpha = \mathbf{0.7993}$
${}_0^C D_t^{\mathbf{0.2}} m = 10\Delta m$	$a/a^* = 0.9994$ $\alpha = \mathbf{0.2000}$	$a/a^* = 0.3474$ $\alpha = \mathbf{0.1826}$
$\frac{\partial m}{\partial t} = -10(-\Delta)^{\mathbf{0.2}} m$	$a/a^* = 1.0000$ $s = \mathbf{0.2000}$	$a/a^* = 1.0378$ $s = \mathbf{0.2069}$
$\frac{\partial m}{\partial t} = -10(-\Delta)^{\mathbf{0.8}} m$	$a/a^* = 1.0000$ $s = \mathbf{0.8000}$	$a/a^* = 1.0365$ $s = \mathbf{0.8093}$

PoreFlow.jl: Inverse Modeling of Viscoelasticity

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

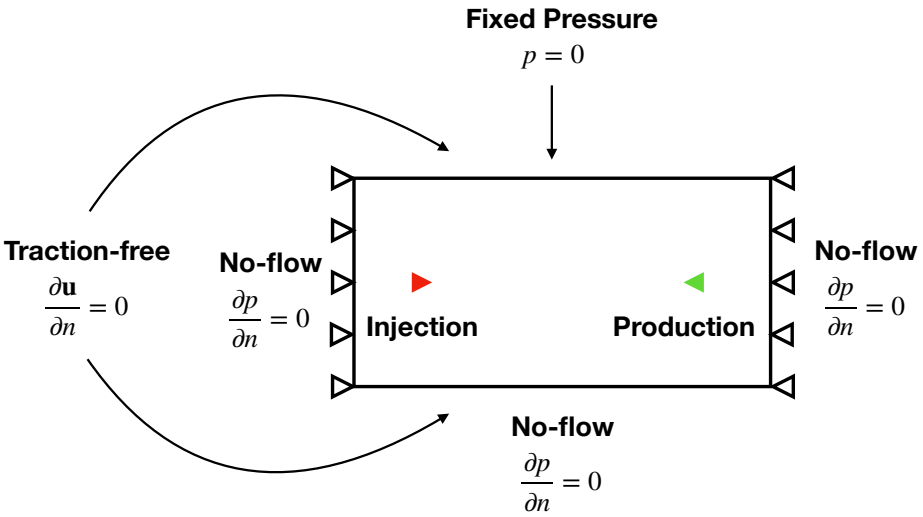
$$\mathrm{div}\boldsymbol{\sigma}(\mathbf{u}) - b\nabla p = 0$$

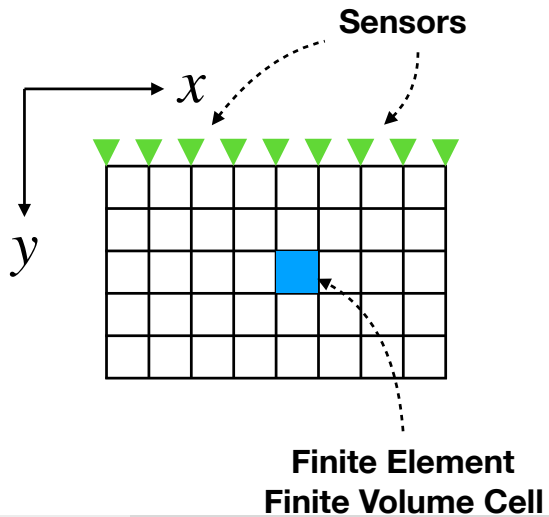
$$\frac{1}{M}\frac{\partial p}{\partial t} + b\frac{\partial \epsilon_v(\mathbf{u})}{\partial t} - \nabla \cdot \left(\frac{k}{B_f \mu}\nabla p\right) = f(x,t)$$

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

- Approximate the constitutive relation by a neural network

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

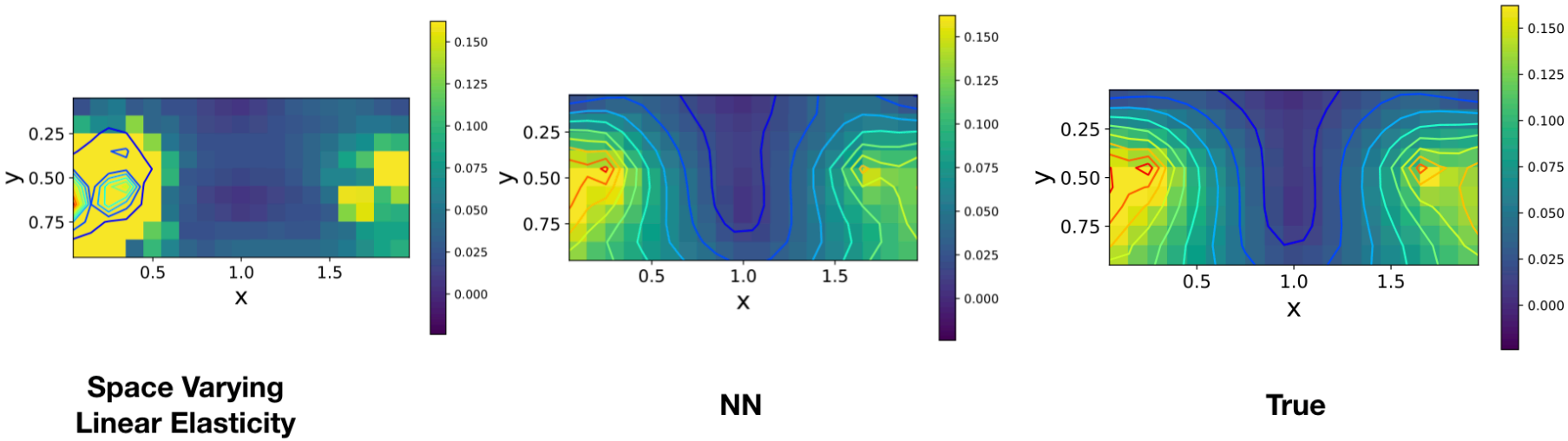




PoreFlow.jl: Inverse Modeling of Viscoelasticity

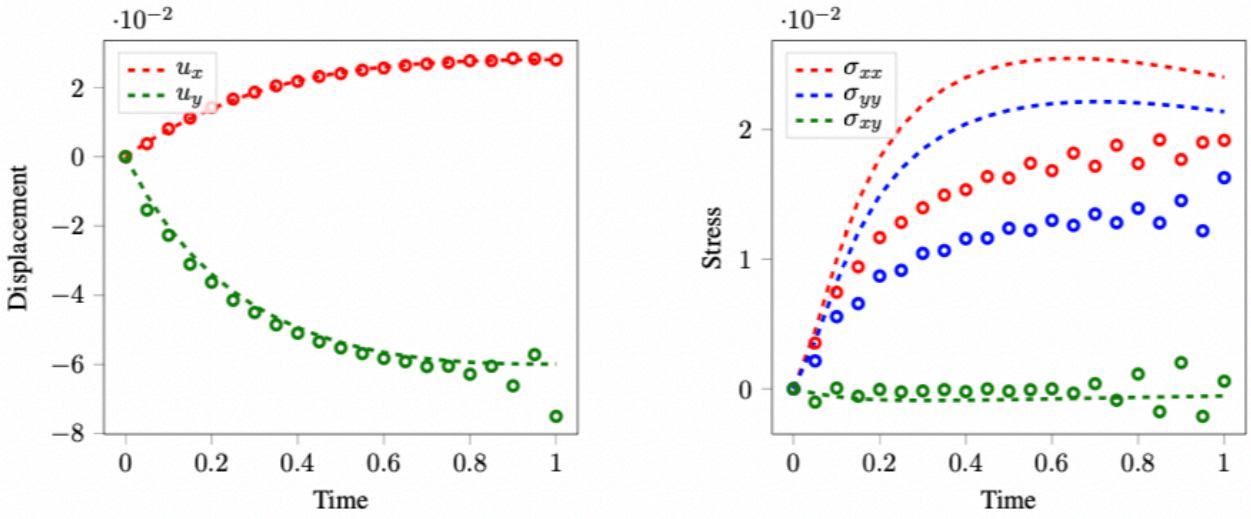
- Comparison with space varying linear elasticity approximation

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

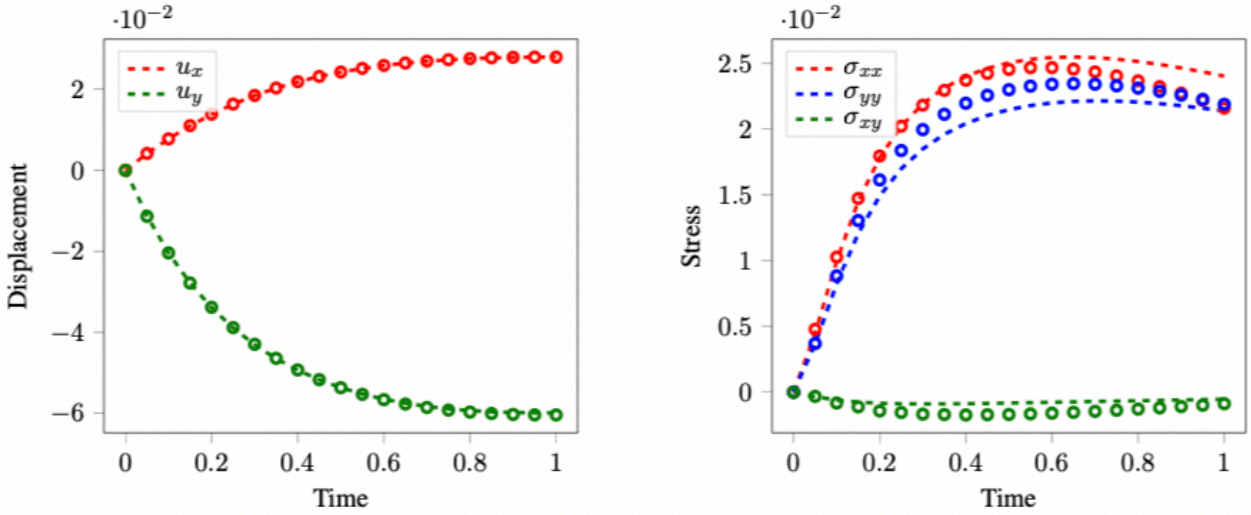


Space Varying Linear Elasticity **NN** **True**

PoreFlow.jl: Inverse Modeling of Viscoelasticity



(a) Space Varying Linear Elasticity



(b) NN-based Viscoelasticity

Outline

Scopes, Challenges, and Future Work

Physics based Machine Learning: an innovative approach to inverse modeling.

1. Deep neural networks provide a novel function approximator that outperforms traditional basis functions in certain scenarios.
2. Numerical PDEs are not on the opposite side of machine learning. By expressing the known physical constraints using numerical schemes and approximating the unknown with machine learning models, we combine the best of the two worlds, leading to efficient and accurate inverse modeling tools.

Automatic Differentiation: the core technique of physics based machine learning.

1. The AD technique is not new; it has existed for several decades and many software exists.
2. The advent of deep learning drives the development of robust, scalable and flexible AD software that leverages the high performance computing environment.
3. As deep learning techniques continue to grow, crafting the tool to incorporate machine learning and AD techniques for inverse modeling is beneficial in scientific computing.
4. However, AD is not a panacea. Many scientific computing algorithms cannot be directly translated to the AD language.

ADCME

- ADCME is the materialization of the physics based machine learning concept.
- ADCME allows users to use high performance and mathematical friendly programming language Julia to implement numerical schemes, and obtain the comprehensive automatic differentiation functionality, heterogeneous computing capability, parallelism and scalability provided by the TensorFlow backend.

`https://github.com/kailaix/ADCME.jl`

```
function one_step(param::AcousticPropagatorParams, w::PyObject, wold::PyObject, φ, ψ,
σ::PyObject, τ::PyObject, c::PyObject)
    Δt = param.DELTAT
    hx, hy = param.DELTAX, param.DELTAY
    IJ, IpJ, InJ, IJp, IJn, IpJp, IpJn, InJp, InJn =
        param.IJ, param.IpJ, param.InJ, param.IJp, param.IJn, param.IpJp, param.IpJn, param.InJp, param.InJn

    u = (2 - σ[IJ]*τ[IJ]*Δt^2 - 2*Δt^2/hx^2 * c[IJ] - 2*Δt^2/hy^2 * c[IJ]) * w[IJ] +
            c[IJ] * (Δt/hx)^2  *  (w[IpJ]+w[InJ]) +
            c[IJ] * (Δt/hy)^2  *  (w[IJp]+w[IJn]) +
            (Δt^2/(2hx))*(φ[IpJ]-φ[InJ]) +
            (Δt^2/(2hy))*(ψ[IJp]-ψ[IJn]) -
            (1 - (σ[IJ]+τ[IJ])*Δt/2) * wold[IJ]
    u = u / (1 + (σ[IJ]+τ[IJ])/2*Δt)
    u = vector(IJ, u, (param.NX+2)*(param.NY+2))
    φ = (1. -Δt*σ[IJ]) * φ[IJ] + Δt * c[IJ] * (τ[IJ] -σ[IJ])/2hx *
        (u[IpJ]-u[InJ])
    ψ = (1. -Δt*τ[IJ]) * ψ[IJ] + Δt * c[IJ] * (σ[IJ] -τ[IJ])/2hy *
        (u[IJp]-u[IJn])
    φ = vector(IJ, φ, (param.NX+2)*(param.NY+2))
    ψ = vector(IJ, ψ, (param.NX+2)*(param.NY+2))
    u, φ, ψ
end
```

Julia code

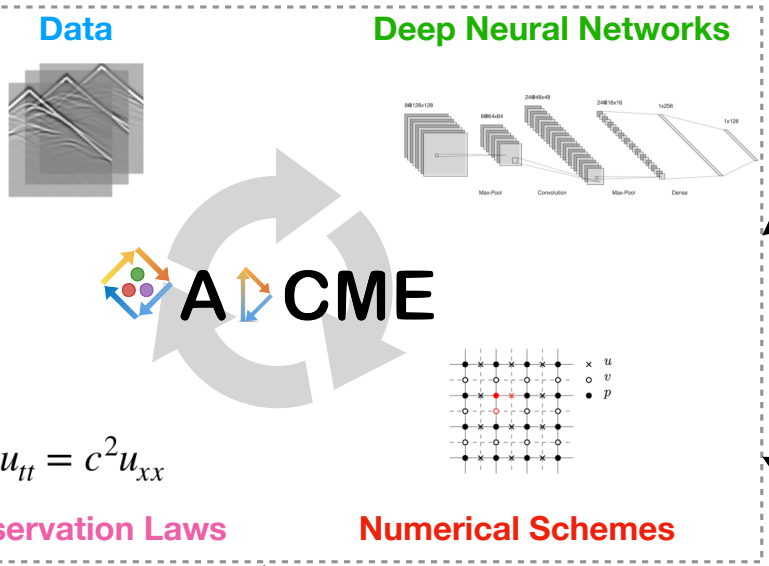
$$\frac{u_{i,j,k}^{n+1}-2u_{i,j,k}^{n}+u_{i,j,k}^{n-1}}{\Delta t^2}+(\zeta_{1i}+\zeta_{2j}+\zeta_{3k})\frac{u_{i,j,k}^{n+1}-u_{i,j,k}^{n-1}}{2\Delta t}+(\zeta_{1i}\zeta_{2j}+\zeta_{2j}\zeta_{3k}+\zeta_{3k}\zeta_{1i})u_{i,j,k}^{n}$$
$$=\frac{c_{i+\frac{1}{2},j,k}^2 u_{i+1,j,k}^n-(c_{i+\frac{1}{2},j,k}^2+c_{i-\frac{1}{2},j,k}^2)u_{i,j,k}^n+c_{i-\frac{1}{2},j,k}^2 u_{i-1,j,k}^n}{{\Delta x_1}^2}$$
$$+\frac{c_{i,j+\frac{1}{2},k}^2 u_{i,j+1,k}^n-(c_{i,j+\frac{1}{2},k}^2+c_{i,j-\frac{1}{2},k}^2)u_{i,j,k}^n+c_{i,j-\frac{1}{2},k}^2 u_{i,j-1,k}^n}{{\Delta x_2}^2}$$
$$+\frac{c_{i,j,k+\frac{1}{2}}^2 u_{i,j,k+1}^n-(c_{i,j,k+\frac{1}{2}}^2+c_{i,j,k-\frac{1}{2}}^2)u_{i,j,k}^n+c_{i,j,k-\frac{1}{2}}^2 u_{i,j,k-1}^n}{{\Delta x_3}^2}$$
$$+\frac{\tilde{\phi}_{1i+\frac{1}{2},j,k}^n-\tilde{\phi}_{1i-\frac{1}{2},j,k}^n}{\Delta x_1}+\frac{\tilde{\phi}_{2i,j+\frac{1}{2},k}^n-\tilde{\phi}_{2i,j-\frac{1}{2},k}^n}{\Delta x_2}+\frac{\tilde{\phi}_{3i,j,k+\frac{1}{2}}^n-\tilde{\phi}_{3i,j,k-\frac{1}{2}}^n}{\Delta x_3}-\zeta_{1i}\,\zeta_{2j}\,\zeta_{3k}\frac{\psi_{i,j,k}^{n+\frac{1}{2}}+\psi_{i,j,k}^{n-\frac{1}{2}}}{2},$$

Discretization

$$u_{tt}+(\zeta_1+\zeta_2)u_t+\zeta_1\zeta_2 u=\nabla\cdot(c^2\nabla u)+\nabla\cdot\boldsymbol{\phi},$$
$$\boldsymbol{\phi}_t=\Gamma_1\boldsymbol{\phi}+c^2\Gamma_2\nabla u,$$
$$\Gamma_1=\begin{bmatrix}-\zeta_1 & 0\\ 0 & -\zeta_2\end{bmatrix},\quad \Gamma_2=\begin{bmatrix}\zeta_2-\zeta_1 & 0\\ 0 & \zeta_1-\zeta_2\end{bmatrix}.$$

PML equations

A General Approach to Inverse Modeling



PoreFlow

PoreFlow.jl
Geomechanics
Viscoelasticity
Multiphase Flow
Multiphysics
* coming soon

FwiFlow.jl
Multiphase Flow
Nonlocal Operators
https://github.com/lidongzh/FwiFlow.jl

ADSeismic.jl
General Seismic Inversion
https://github.com/kailaix/ADSeismic.jl

NNFEM

NNFEM.jl
Constitutive Law Modeling
Hyperelasticity
Elasto-Plasticity
* coming soon