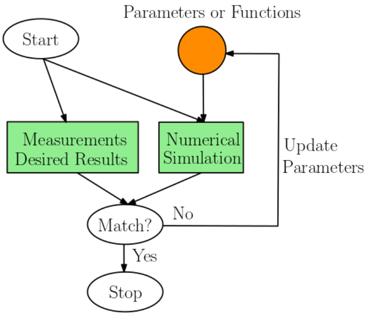
# ADCME: Automatic Differentiation Library for Computational and Mathematical Engineering

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

November 26, 2019

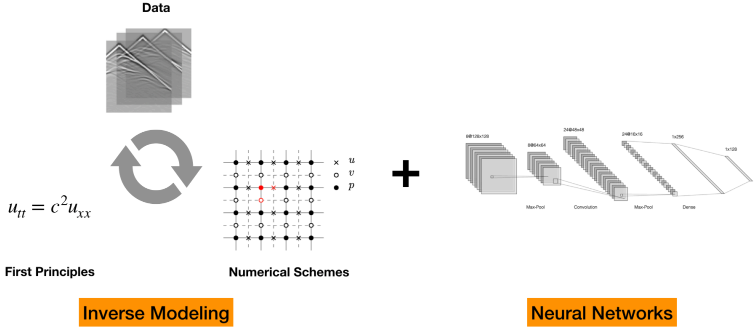
# Inverse Modeling

- **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.



# Physics Based Machine Learning

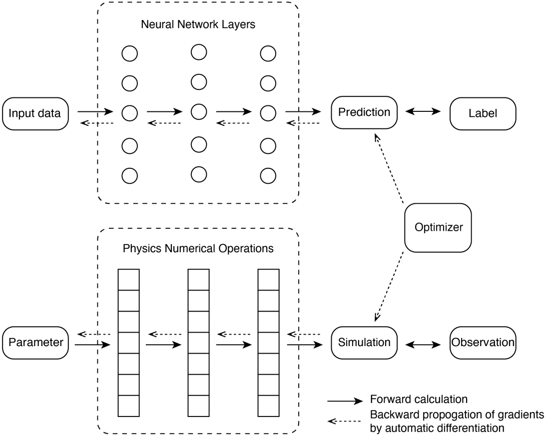
- Traditional inverse modeling methods utilize efficient numerical schemes and incorporate physical knowledge (first principles); deep learning learns statistical relations from large amounts of training data.
- We combine the best of the two worlds and invent **physics based machine learning**.



# Automatic Differentiation

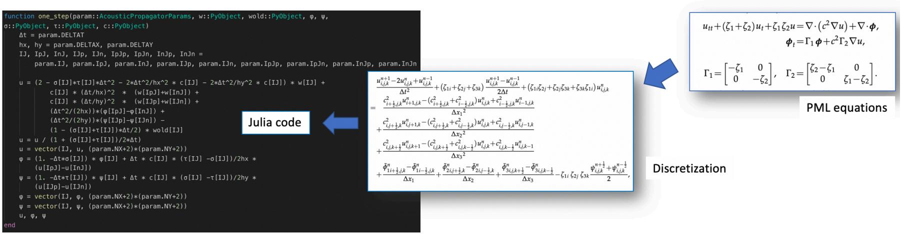
- Deep learning and inverse modeling have the same computational model but are disguised under different terminologies.

Back-propagation = Automatic Differentiation = Discrete Adjoint State Method



# AD Implementation in ADCME

- ADCME allows users to use high level script language Julia to implement numerical simulation codes, but obtain the powerful parallelism and scalability provided by TensorFlow and Julia itself.
- Gradients are computed automatically.

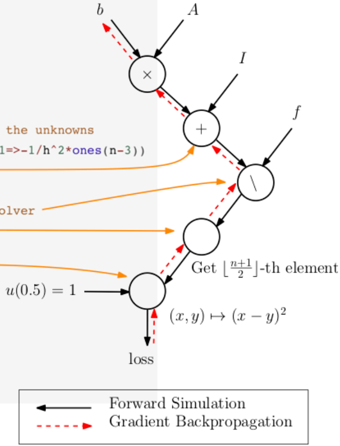


# Code Example

- Find $b$ such that $u(0.5) = 1.0$ and

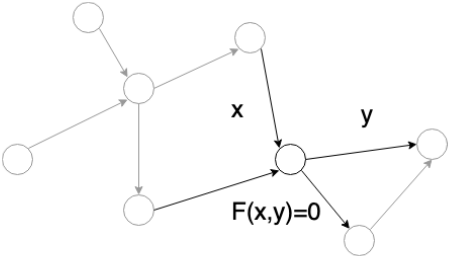
$$-bu''(x) + u(x) = 8 + 4x - 4x^2, x \in [0, 1], u(0) = u(1) = 0$$

```julia
using LinearAlgebra
using ADCME

n = 101 # number of grid nodes in [0,1]
h = 1/(n-1)
x = LinRange(0,1,n)[2:end-1]

b = Variable(10.0) # we use Variable keyword to mark the unknowns
A = diagm(0=>2/h^2*ones(n-2), -1=>-1/h^2*ones(n-3), 1=>-1/h^2*ones(n-3))
B = b*A + I  # I stands for the identity matrix
f = @. 4*(2 + x - x^2)
u = B\f # solve the equation using built-in linear solver
ue = u[div(n+1,2)] # extract values at x=0.5

loss = (ue-1.0)^2

# Optimization
sess = Session(); init(sess)
BFGS!(sess, loss)

println("Estimated b = ", run(sess, b))
```



# Challenges in AD

- ADCME aims to solve the nonlinear implicit operator case via custom operators.
- Another ongoing effort is automatic calculation of Jacobian (`IGACS.jl`).




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

# ADCME Solutions

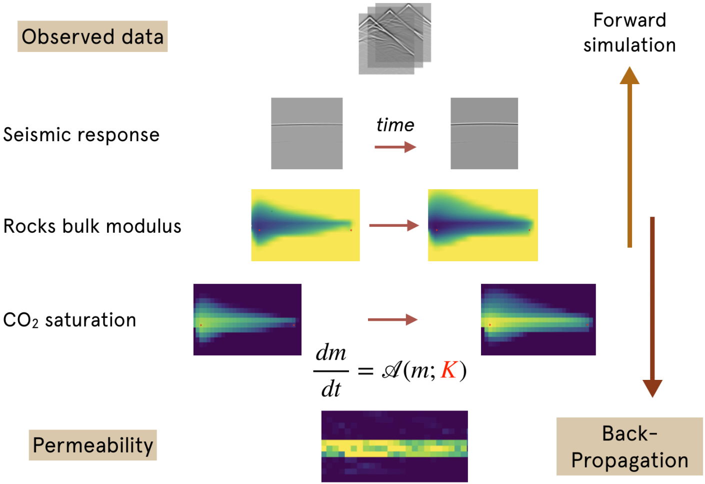
- Most inverse modeling problems can be classified into 4 categories. For example, the PDE for describing physics is

$$\nabla \cdot (X \nabla u(x)) = 0 \quad \mathcal{BC}(u(x)) = 0 \tag{1}$$

We observe some quantities depending on the solution $u$ and want to estimate $X$.

| Expression | Description | ADCME Solution | Note |
|---|---|---|---|
| $\nabla \cdot (c \nabla u(x)) = 0$ | Parameter Inverse Problem | Discrete Adjoint State Method | Direct optimize the constant $c$ |
| $\nabla \cdot (f(x) \nabla u(x)) = 0$ | Functional Inverse Problem | Neural Network Functional Approximator | $f(x) \approx f_\theta(x)$ |
| $\nabla \cdot (f(u) \nabla u(x)) = 0$ | Relation Inverse Problem | Deep Learning for Indirect Data | $f(u) \approx f_\theta(u)$ |
| $\nabla \cdot (\varpi \nabla u(x)) = 0$ | Stochastic Inverse Problem | Adversarial Numerical Analysis | Generative Neural Nets for $\varpi$ (unknown random processes) |

# Parameter Inverse Problem: Learning Hidden Geophysical Processes



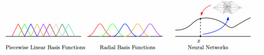
# Functional Inverse Problem: Calibrating Lévy Processes

$$\phi(\boldsymbol{\xi}) = \mathbb{E}[e^{\mathrm{i}\langle \boldsymbol{\xi}, \mathbf{X}_t\rangle}] =$$

$$\exp\left[t\left(\mathrm{i}\langle \mathbf{b}, \boldsymbol{\xi}\rangle - \frac{1}{2}\langle \boldsymbol{\xi}, \mathbf{A}\boldsymbol{\xi}\rangle + \int_{\mathbb{R}^d}\left(e^{\mathrm{i}\langle \boldsymbol{\xi}, \mathbf{x}\rangle} - 1 - \mathrm{i}\langle \boldsymbol{\xi}, \mathbf{x}\rangle \mathbf{1}_{\|\mathbf{x}\|\leq 1}\right)\nu(d\mathbf{x})\right)\right]$$

**Model**

$$\nu(\mathbf{x}) \approx \nu_{\boldsymbol{\theta}}(\mathbf{x})$$

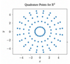



↓

$$\phi(\boldsymbol{\xi}) \approx \phi_{\boldsymbol{\theta}}(\boldsymbol{\xi}) := \exp\left[\Delta t \sum_{i=1}^{n_q}\left(e^{\mathrm{i}\langle \boldsymbol{\xi}, \mathbf{x}_i\rangle} - 1 - \mathrm{i}\langle \boldsymbol{\xi}, \mathbf{x}_i\rangle \mathbf{1}_{\|\mathbf{x}_i\|\leq 1}\right)\nu_{\boldsymbol{\theta}}(\mathbf{x}_i) w_i\right]$$

↓

$$\min_{\boldsymbol{\theta}} \frac{1}{m}\sum_{i=1}^{m}\|\hat{\phi}_n(\boldsymbol{\xi}_i) - \phi_{\boldsymbol{\theta}}(\boldsymbol{\xi}_i)\|^2$$



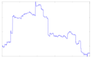
↑

**Data**

$$\hat{\phi}_n(\boldsymbol{\xi}) := \frac{1}{n}\sum_{i=1}^{n}\exp(\mathrm{i}\langle \boldsymbol{\xi}, \mathbf{X}_{i\Delta t} - \mathbf{X}_{(i-1)\Delta t}\rangle),\ \boldsymbol{\xi} \in \mathbb{R}^d$$

↑

$$\{\mathbf{X}_{i\Delta t}\}_i$$



# Relation Inverse Problem: Learning Constitutive Relations

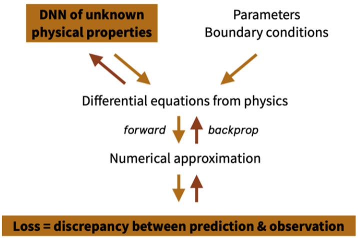
- Equilibrium equation

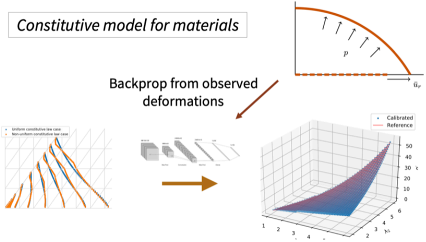
$$\mathcal{P}(u(\mathbf{x}), \mathcal{M}(u(\mathbf{x}), \dot{u}(\mathbf{x}), \mathbf{x})) = \mathcal{F}(u(\mathbf{x}), \mathbf{x}, p)$$

- Neural Network Approximation:

$$\mathcal{M}_\theta(\mathbf{u}) \approx \mathcal{M}(u(\mathbf{x}), \dot{u}(\mathbf{x}), \mathbf{x})$$

$$\boxed{\min_\theta \|\mathcal{P}(\mathbf{u}, \mathcal{M}_\theta(\mathbf{u})) - \mathcal{F}(\mathbf{u}, \mathbf{x}, p)\|_2^2}$$

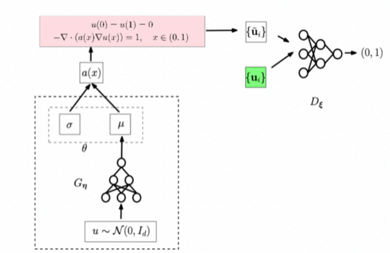


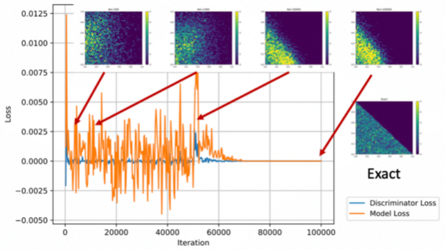



# Probability Inverse Problem: Adversarial Numerical Analysis

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = 1 & x \in (0,1) \\ u(0) = u(1) = 0 & \text{otherwise} \end{cases}$$

$$a(x) = 1 - 0.9\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



# A Cool Application: ADSeismic.jl (Coming soon)

- An Open Source High Performance Package for General Seismic Inversion Problems
- Problems include:
  - Full waveform inversion (FWI);
  - Rupture inversion;
  - Source-time inversion.
- Features:
  - (Multi-)GPU support;
  - Easy-to-use;
  - Easily extendable.