Physics-Informed Deep Learning

Learning from Small Data

Review Article | Published: 24 May 2021

Physics-informed machine learning

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang & Liu Yang

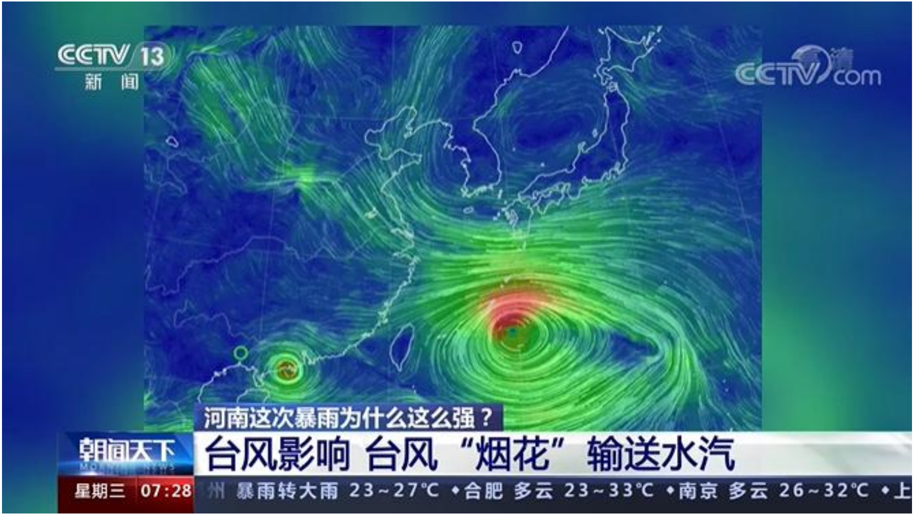
Nature Reviews Physics **3**, 422–440 (2021) | Cite this article

Lu Lu

Applied Mathematics Instructor, MIT

Assistant Professor of Chemical and Biomolecular Engineering, UPenn

Simulate and predict physical systems *in real time*

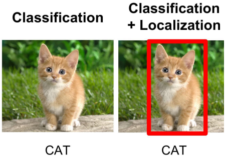


Numerical discretization of partial differential equations

- Cannot seamlessly incorporate noisy data
- Mesh generation remains complex
- High-dimensional problems cannot be tackled
- Solving inverse problems is often prohibitively expensive

$\Rightarrow$ New opportunity: Machine learning

- Computer vision



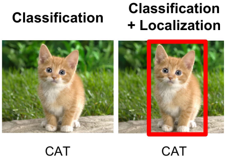
- Machine translation

- Black-box & Trial-and-error
- Big data

Scientific Machine Learning: Learning from Small Data

- *Dinky, Dirty, Dynamic, Deceptive* Data

- Computer vision



- Machine translation

- Black-box & Trial-and-error
- Big data

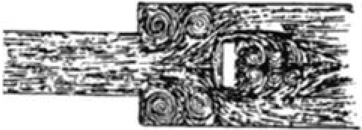
Scientific Machine Learning: Learning from Small Data

- *Dinky, Dirty, Dynamic, Deceptive* Data
- Scientific domain knowledge

  e.g., physical principles, constraints, symmetries, computational simulations
- Accurate, Robust, Reliable, Interpretable, Explainable

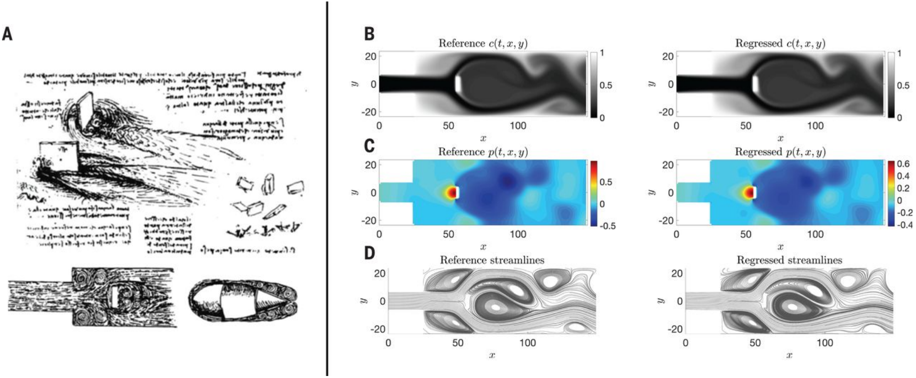
Leonardo da Vinci's drawing of turbulence

"Observe the motion of the surface of the water, Which resembles that of hair, which has two motions ..."



Machine learning with physics

Infer pressure p and velocity fields (u, v) from concentration c

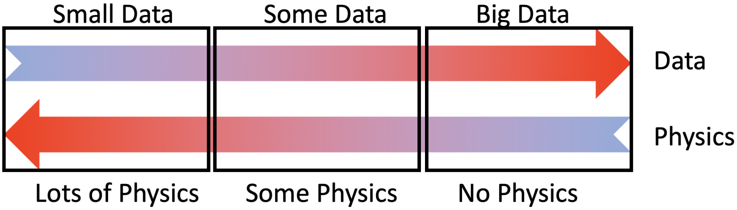


Raissi, Yazdani, & Karniadakis, *Science*, 2020

Data & Physics

Karniadakis, Kevrekidis, **Lu**, et al., *Nature Rev Phys*, 2021

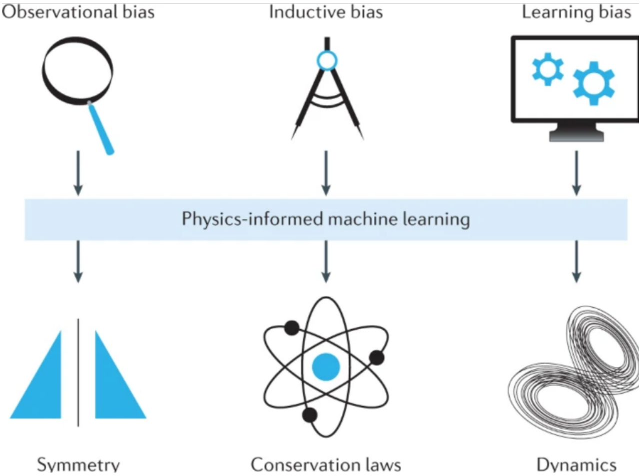
Three scenarios:



1. Lots of physics—Forward problems
   - Finite difference/elements
2. Some physics—Inverse problems
   - Multi-fidelity learning
   - Physics-informed neural network (PINN)
   - DeepM&Mnet
3. No physics—System identification/discovery
   - Operator learning (DeepONet)

How to embed physics in ML?

Principles of physics-informed learning:



Karniadakis, Kevrekidis, **Lu**, et al., *Nature Rev Phys*, 2021

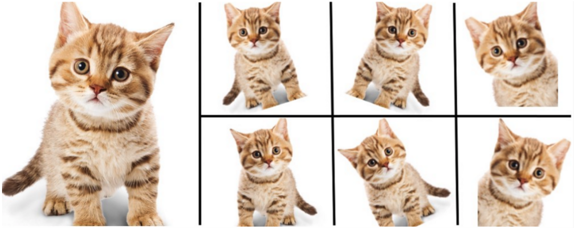
Observational bias

Directly from **data**

- e.g., $f \mapsto u$ is symmetry invariant

$$\text{sym}(f) \mapsto u$$

- Data augmentation



- Brute-force: Let the machine learn.
- May need a large dataset

Inductive bias

ML model architecture: Embed any prior knowledge

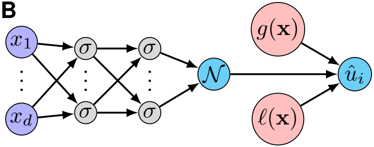
- CNN: translation invariance
- Partially fixed-value network (Dirichlet BC):
  $u(0) = 0, u(1) = 1$: $g(x) = x$, $\ell(x) = x(1 - x)$

$$\mathcal{N}(x) = g(x), \quad x \in \Gamma_D \subset \Omega$$

$$\hat{u}(x) = g(x) + \ell(x)\mathcal{N}(x)$$

$$\begin{cases} \ell(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_D, \\ \ell(\mathbf{x}) > 0, & \mathbf{x} \in \Omega - \Gamma_D. \end{cases}$$

B

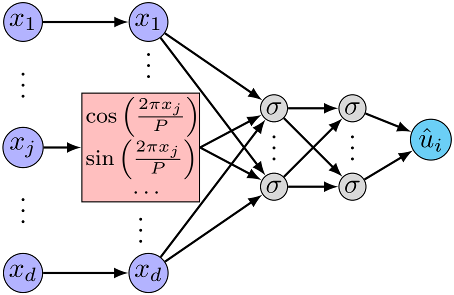


Lu et al., arXiv:2102.04626

Inductive bias

- Periodic network (Periodic BC) via Fourier series:

$$\{1, \cos(\frac{2\pi x_j}{P}), \sin(\frac{2\pi x_j}{P}), \cos(\frac{4\pi x_j}{P}), \sin(\frac{4\pi x_j}{P}), \cdots\}$$



Can use as few as two terms $\{\cos(\frac{2\pi x_j}{P}), \sin(\frac{2\pi x_j}{P})\}$

Lu et al., arXiv:2102.04626

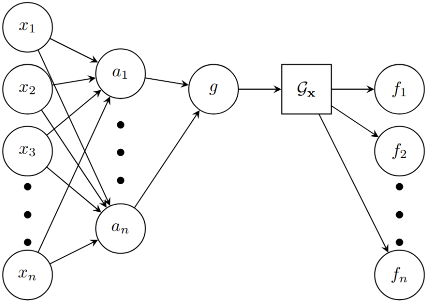
Inductive bias

- Linearly constrained neural networks, e.g., divergence-free

$$\nabla \cdot \mathbf{f} = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} = 0$$

Divergence of the curl is 0: $\nabla \cdot (\nabla \times g) \equiv 0$, so choose

$$\mathbf{f} = \nabla \times g$$



Hendriks et al., arXiv:2002.01600

Inductive bias

Free lunch: The network satisfies the physical constraints **automatically** and **exactly**.

- Do it if possible
- Designed case by case
- Currently limited to relatively simple and well-defined physics

Learning bias

Idea:

- loss functions
- modulate the training phase

e.g., Physics-informed neural network (PINN)

Inverse problems

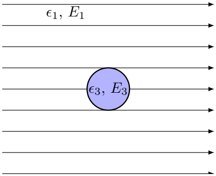
Challenge: *small* data + *incomplete* physics laws

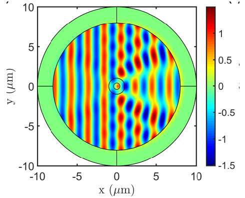
Invisible cloaking

joint work with Prof. Luca Dal Negro (Boston U)

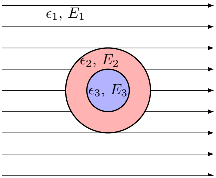
Permittivity ϵ, permeability μ

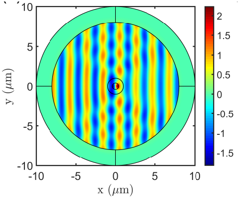
Electric field E without coating





Electric field E with coating

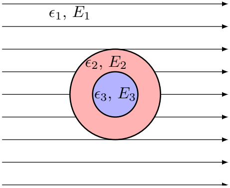




Chen, **Lu**, et al., *Opt Express*, 2020

Invisible cloaking

Goal: Given ϵ_1 and ϵ_3, find $\epsilon_2(x, y)$ s.t. $E_1 \approx E_{1,target}$



Helmholtz equation ($k_0 = \frac{2\pi}{\lambda_0}$)

$$\nabla^2 E_i + \epsilon_i k_0^2 E_i = 0, \qquad i = 1, 2, 3$$

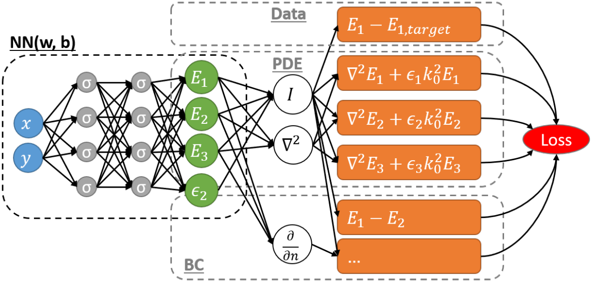
Boundary conditions:

- Outer circle: $E_1 = E_2$, $\frac{1}{\mu_1}\frac{\partial E_1}{\partial \mathbf{n}} = \frac{1}{\mu_2}\frac{\partial E_2}{\partial \mathbf{n}}$
- Inner circle: $E_2 = E_3$, $\frac{1}{\mu_2}\frac{\partial E_2}{\partial \mathbf{n}} = \frac{1}{\mu_3}\frac{\partial E_3}{\partial \mathbf{n}}$

Chen, **Lu**, et al., *Opt Express*, 2020

Physics-informed neural networks (PINNs)

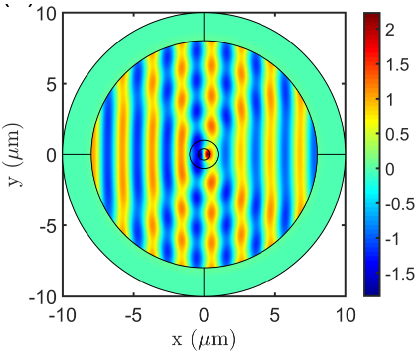
Idea: Embed a PDE into the loss via automatic differentiation (AD)



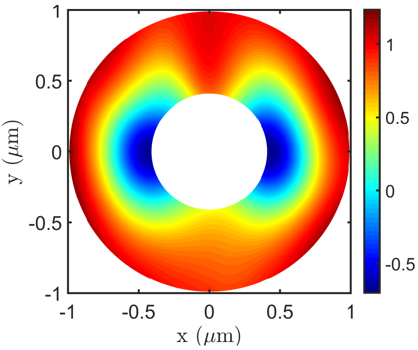
- mesh-free & particle-free
- inverse problems: seamlessly integrate data and physics
- black-box or noisy IC/BC/forcing terms (Pang*, **Lu***, et al., *SIAM J Sci Comput*, 2019)
- a unified framework: PDE, integro-differential equations (**Lu** et al., *SIAM Rev*, 2021), fractional PDE (Pang*, **Lu***, et al., *SIAM J Sci Comput*, 2019), stochastic PDE (Zhang, **Lu**, et al., *J Comput Phys*, 2019)

Invisible cloaking

Electric field E_i



Permittivity ϵ_2



Chen, **Lu**, et al., *Opt Express*, 2020

Physics-informed neural networks (PINNs)

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) = 0, \quad \mathbf{x} \in \Omega$$

- Initial/boundary conditions $\mathcal{B}(u, \mathbf{x}) = 0$ on $\partial\Omega$
- Extra information $\mathcal{I}(u, \mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{T}_i$

$$\min_{\boldsymbol{\theta}, \boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_b) + w_i \mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_i)$$

where

$$\mathcal{L}_f = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots; \boldsymbol{\lambda}\right) \right\|_2^2$$

$$\mathcal{L}_b = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \mathbf{x})\|_2^2$$

$$\mathcal{L}_i = \frac{1}{|\mathcal{T}_i|} \sum_{\mathbf{x} \in \mathcal{T}_i} \|\mathcal{I}(\hat{u}, \mathbf{x})\|_2^2$$

Error analysis

Theorem (Universal approximation theorem; Cybenko, 1989)

Let σ be any continuous sigmoidal function. Then finite sums of the form $G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j \cdot x + b_j)$ are dense in $C(I_d)$.

Theorem (Pinkus, 1999)

Let $\mathbf{m}^i \in \mathbb{Z}_+^d$, $i = 1, \ldots, s$, and set $m = \max_{i=1,\ldots,s}(m_1^i + \cdots + m_d^i)$. Assume $\sigma \in C^m(\mathbb{R})$ and σ is not a polynomial. Then the space of single hidden layer neural nets

$$\mathcal{M}(\sigma) := span\{\sigma(\mathbf{w} \cdot \mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

is dense in

$$C^{\mathbf{m}^1,\ldots,\mathbf{m}^s}(\mathbb{R}^d) := \cap_{i=1}^{s} C^{\mathbf{m}^i}(\mathbb{R}^d).$$

Optimization & Generalization:

Shin et al., 2020; Mishra & Molinaro, 2020; Luo & Yang, 2020

Systems biology described by ODEs

Ultradian model for the glucose-insulin interaction (Sturis et al., 1991)

Goal: Infer 21 unknown parameters

$$\frac{dI_p}{dt} = f_1(G) - E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_p}{t_p}$$

$$\frac{dI_i}{dt} = E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_i}{t_i}$$

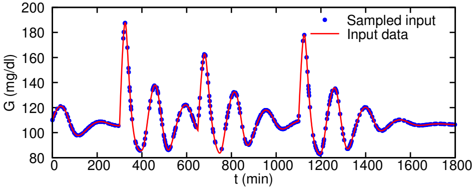
$$\frac{dG}{dt} = f_4(h_3) + I_G(t) - f_2(G) - f_3(I_i)G$$

$$\frac{dh_1}{dt} = \frac{1}{t_d}\left(I_p - h_1\right)$$

$$\frac{dh_2}{dt} = \frac{1}{t_d}\left(h_1 - h_2\right)$$

$$\frac{dh_3}{dt} = \frac{1}{t_d}\left(h_2 - h_3\right)$$

- I_p: plasma insulin concentration
- I_i: interstitial insulin concentration
- G: glucose concentration (Data)
- I_G: glucose from nutritional intake

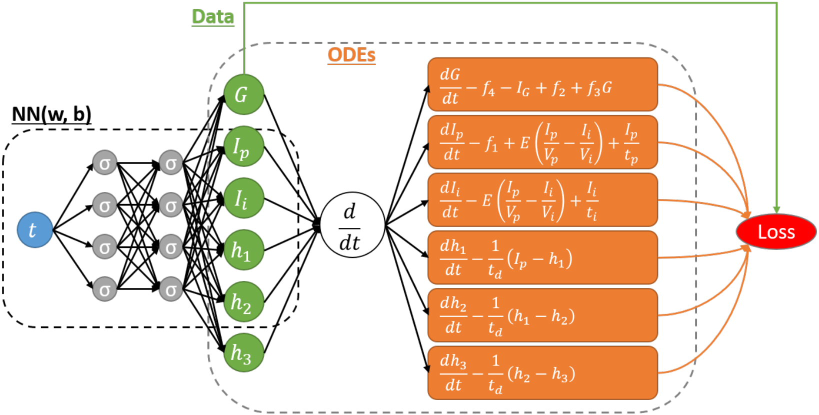


$$f_1(G) = \frac{R_m}{1 + \exp(\frac{-G}{V_g C_1} + a_1)},\ f_2(G) = U_b\left(1 - \exp(\frac{-G}{C_2 V_g})\right),\ f_3(I_i) = \frac{1}{C_3 V_g}\left(U_0 + \frac{U_m}{1 + (\kappa I_i)^{-\beta}}\right)$$

$$f_4(h_3) = \frac{R_g}{1 + \exp(\alpha(\frac{h_3}{C_5 V_p} - 1))},\ \kappa = \frac{1}{C_4}\left(\frac{1}{V_i} + \frac{1}{E t_i}\right),\ I_G(t) = \sum_{j=1}^{N} m_j k \exp(k(t_j - t))$$

Physics-informed neural networks (PINNs)

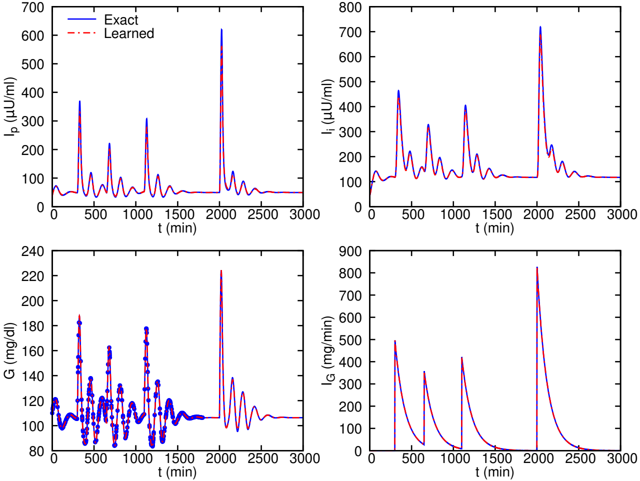
Idea: Embed ODEs with unknown parameters into the loss via automatic differentiation (backpropagation)



- Seamlessly integrate *sparse* data and *incomplete* physics

Yazdani*, **Lu***, et al., *PLoS Comput Biol*, 2020

Inferred dynamics and forecasting



Research Highlight in

Nature Computational Science

SYSTEMS BIOLOGY

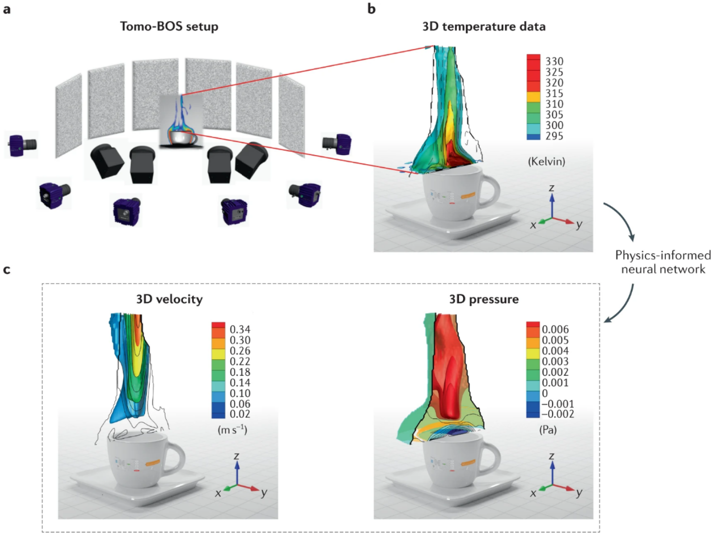
Parameter inference for systems biology models

Ananya Rastogi

Nature Computational Science **1**, 16(2021) | Cite this article

Inferring the flow over an espresso cup

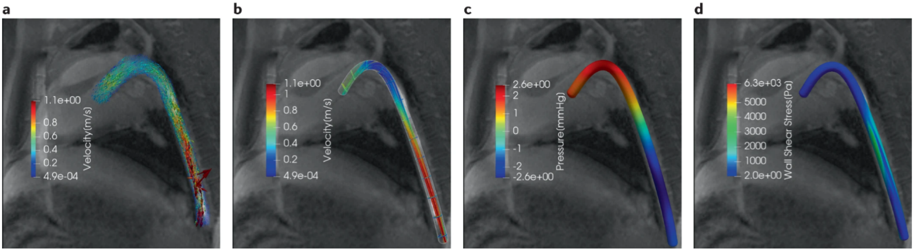
Data: A video of temperature field.



Cai et al., *J Fluid Mech*, 2021

Filtering of *in-vivo* 4D-flow magnetic resonance imaging (MRI) data of blood flow in a porcine descending aorta

- MRI data: Very coarse resolution & heavily corrupted by noise



- Reconstructions of the velocity and pressure fields
- Identify regions of no-slip flow, from which one can reconstruct the location and motion of the arterial wall

Wang, Kissas, & Perdikaris

hPINN: PINN with hard constraints

Augmented Lagrangian method

$$\mathcal{L}^k(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma) = \mathcal{J} + \mu_\mathcal{F}^k \mathcal{L}_\mathcal{F} + \mu_h^k \mathbb{1}_{\{h>0 \vee \lambda_h^k>0\}} h^2 + \frac{1}{MN} \sum_{j=1}^M \sum_{i=1}^N \lambda_{i,j}^k \mathcal{F}_i \left[\hat{\mathbf{u}}(\mathbf{x}_j); \hat{\gamma}(\mathbf{x}_j)\right] + \lambda_h^k h$$

Algorithm 2.2 hPINNs via the augmented Lagrangian method.

Hyperparameters: initial penalty coefficients $\mu_\mathcal{F}^0$ and μ_h^0, factors $\beta_\mathcal{F}$ and β_h

$k \longleftarrow 0$

$\lambda_{i,j}^0 \longleftarrow 0$ for $1 \le i \le N$, $1 \le j \le M$

$\lambda_h^0 \longleftarrow 0$

$\boldsymbol{\theta}_u^0, \boldsymbol{\theta}_\gamma^0 \longleftarrow \arg\min_{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma} \mathcal{L}^0(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma)$: Train the networks $\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u)$ and $\hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma)$ from random initialization, until the training loss is converged

repeat

$k \longleftarrow k + 1$

$\mu_\mathcal{F}^k \longleftarrow \beta_\mathcal{F} \mu_\mathcal{F}^{k-1}$

$\mu_h^k \longleftarrow \beta_h \mu_h^{k-1}$

$\lambda_{i,j}^k \longleftarrow \lambda_{i,j}^{k-1} + 2\mu_\mathcal{F}^{k-1} \mathcal{F}_i \left[\hat{\mathbf{u}}(\mathbf{x}_j; \boldsymbol{\theta}_u^{k-1}); \hat{\gamma}(\mathbf{x}_j; \boldsymbol{\theta}_\gamma^{k-1})\right]$ for $1 \le i \le N$, $1 \le j \le M$

$\lambda_h^k \longleftarrow \max\left(\lambda_h^{k-1} + 2\mu_h^{k-1} h\left(\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u^{k-1}), \hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma^{k-1})\right), 0\right)$

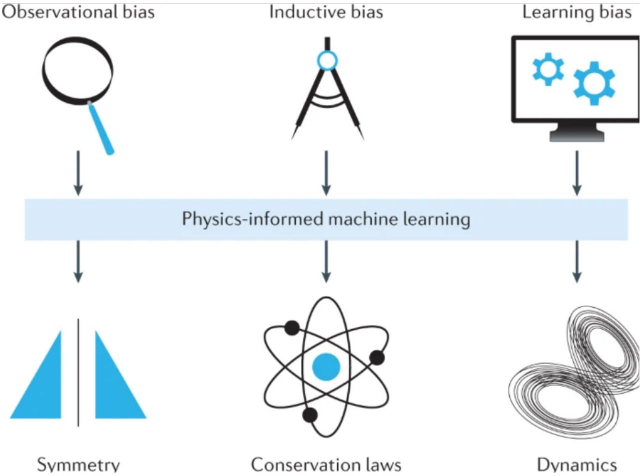
$\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k \longleftarrow \arg\min_{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma} \mathcal{L}^k(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma)$: Train the networks $\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u)$ and $\hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma)$ from the initialization of $\boldsymbol{\theta}_u^{k-1}$ and $\boldsymbol{\theta}_\gamma^{k-1}$, until the training loss is converged

until $\mathcal{L}_\mathcal{F}(\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k)$ and $\mathcal{L}_h(\boldsymbol{\theta}_u^k, \boldsymbol{\theta}_\gamma^k)$ are smaller than a tolerance

Lu et al., arXiv:2102.04626

How to embed physics in ML?

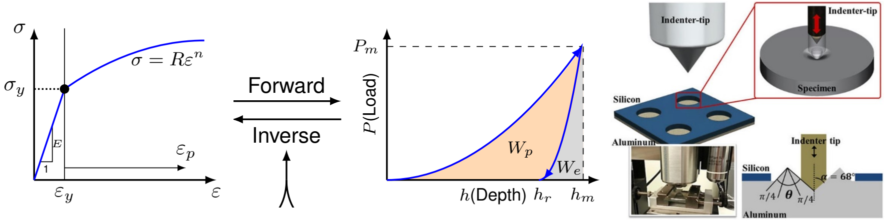
Principles of physics-informed learning:



Karniadakis, Kevrekidis, **Lu**, et al., *Nature Rev Phys*, 2021

Inverse indentation

joint work with Prof. Subra Suresh (NTU) and Dr. Ming Dao (MIT)

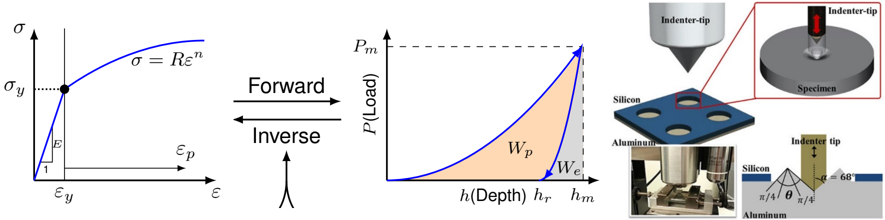


Goal: Stress-strain response $\Leftarrow$ Penetration depth vs Loading force

Lu et al., *PNAS*, 2020

Inverse indentation

joint work with Prof. Subra Suresh (NTU) and Dr. Ming Dao (MIT)



Goal: Stress-strain response $\Leftarrow$ Penetration depth vs Loading force

Challenge: small data

- Physical laws: $E, \sigma_y, n \Leftrightarrow h_m, P_m, C, \frac{dP}{dh}|_{h_m}, \frac{W_p}{W_p+W_e}$
- Multi-fidelity: small experimental data + computational modeling

Lu et al., *PNAS*, 2020

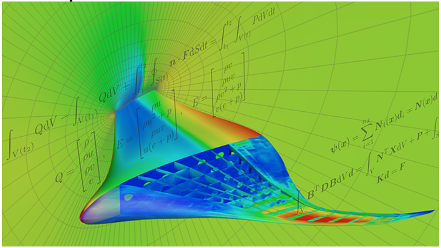
Multi-fidelity learning

Indentation

- Experiments (a few)
- 3D simulations ($\sim$10)
- 2D simulations ($\sim$100)

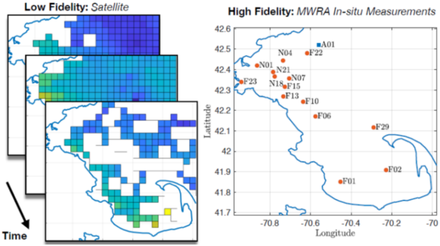
No Data Left Behind.

Examples:



Aircraft design

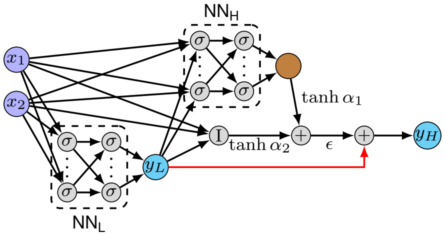
- High fidelity: Experiment
- Low fidelity: Simulation



Sea surface temperature forecast

- High fidelity: In-situ measurements
- Low fidelity: Satellite

Algorithm: Residual multi-fidelity NN



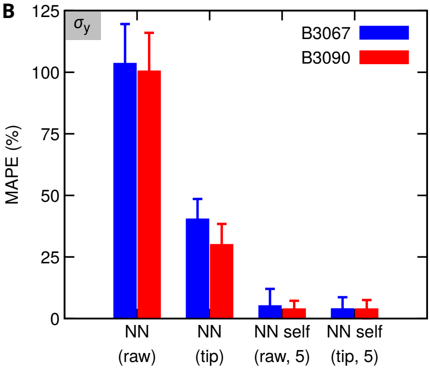
Idea: Correlate low and high fidelity

$$
\begin{aligned}
y_H(\mathbf{x}) - y_L(\mathbf{x}) &= f(\mathbf{x}, y_L(\mathbf{x})) \\
&= \epsilon[\tanh \alpha_1 \cdot f_{\text{linear}}(\mathbf{x}, y_L(\mathbf{x})) + \tanh \alpha_2 \cdot f_{\text{nonlinear}}(\mathbf{x}, y_L(\mathbf{x}))]
\end{aligned}
$$

Lu et al., *PNAS*, 2020

Experiments: 3D printing Ti-6Al-4V alloys

- Experiments (5)
- 3D simulations (14)
- 2D simulations (100)



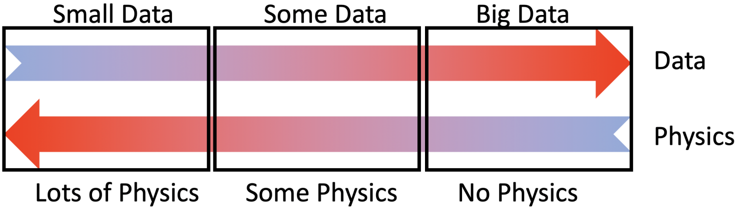
Traditional method: 140%

Lu et al., *PNAS*, 2020

Data & Physics

Karniadakis, Kevrekidis, **Lu**, et al., *Nature Rev Phys*, 2021

Three scenarios:



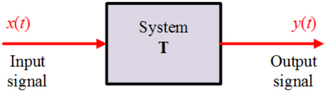
1. Lots of physics—Forward problems
   - Finite difference/elements
2. Some physics—Inverse problems
   - Multi-fidelity learning
   - Physics-informed neural network (PINN)
   - DeepM&Mnet
3. No physics—System identification/discovery
   - Operator learning (DeepONet)

From function to operator

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

- Operator: function (∞-dim) $\mapsto$ function (∞-dim)
  e.g., derivative (local): $x(t) \mapsto x'(t)$
  e.g., integral (global): $x(t) \mapsto \int K(s,t)x(s)ds$

  e.g., dynamic system: 

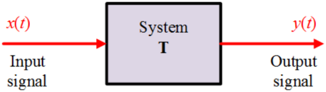
  e.g., biological system
  e.g., social system

From function to operator

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

- Operator: function (∞-dim) $\mapsto$ function (∞-dim)
  e.g., derivative (local): $x(t) \mapsto x'(t)$
  e.g., integral (global): $x(t) \mapsto \int K(s,t)x(s)ds$

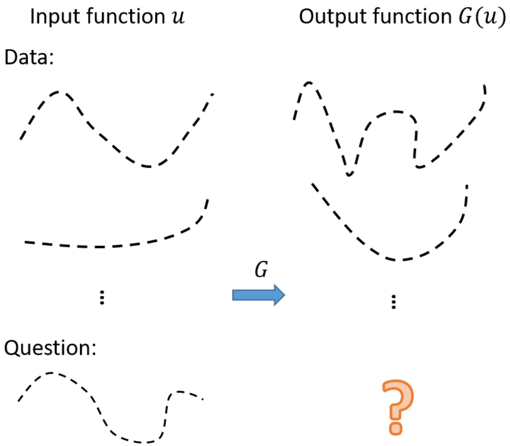
  e.g., dynamic system: 

  e.g., biological system
  e.g., social system

  $\Rightarrow$ Can we learn operators via neural networks?
  $\Rightarrow$ How?

Problem setup

$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$



Universal Approximation Theorem for Operator

$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

Theorem (Chen & Chen, 1995)

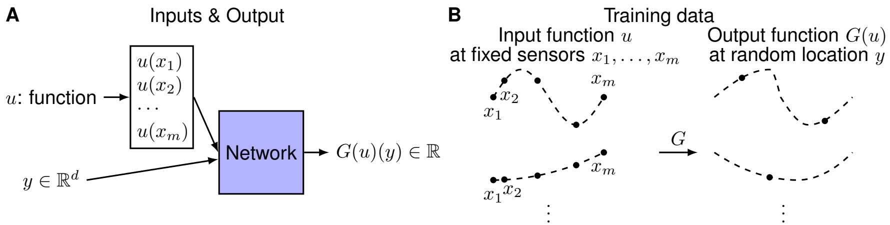
Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a continuous operator, which maps V into $C(K_2)$.
Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \dots, n$, $k = 1, \dots, p$, $j = 1, \dots, m$, such that

$$\left| G(u)(y) - \sum_{k=1}^{p} \sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w_k \cdot y + \zeta_k) \right| < \epsilon$$

holds for all $u \in V$ and $y \in K_2$.

Problem setup

$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$



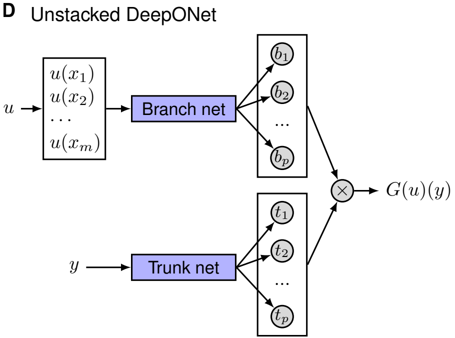
- Inputs: u at sensors $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^m$, $y \in \mathbb{R}^d$
- Output: $G(u)(y) \in \mathbb{R}$

Lu et al., *Nature Mach Intell*, 2021

Deep operator network (DeepONet)

Lu et al., *Nature Mach Intell*, 2021

$$G(u)(y) \approx \sum_{k=1}^{p} \underbrace{b_k(u)}_{\text{branch}} \underbrace{t_k(y)}_{\text{trunk}}$$



Idea: $G(u)(y)$: a function of y conditioning on u

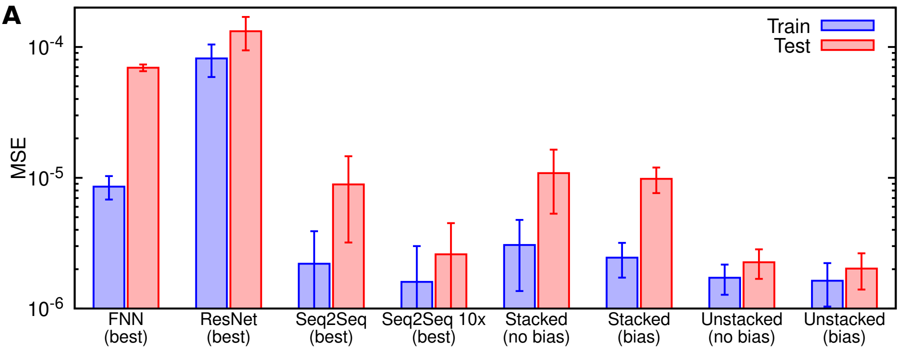
- $t_k(y)$: basis function of y
- $b_k(u)$: u-dependent coefficient, i.e., functional of u

Explicit operator: A simple ODE case

$$\frac{ds(x)}{dx} = u(x), \quad x \in [0,1], \quad s(0) = 0$$

$$G : u(x) \mapsto s(y) = s(0) + \int_0^y u(\tau) d\tau$$

Very small generalization error!

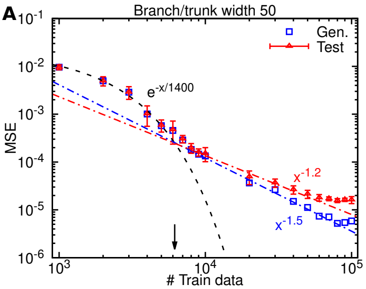


Lu et al., *Nature Mach Intell*, 2021

Implicit operator: Gravity pendulum with an external force

$$\frac{ds_1}{dt} = s_2, \quad \frac{ds_2}{dt} = -k \sin s_1 + u(t)$$

$G : u(t) \mapsto \mathbf{s}(t)$



Test/generalization error:

- small dataset: exponential convergence
- large dataset: polynomial rates
- larger network has later transition point

Lu et al., *Nature Mach Intell*, 2021

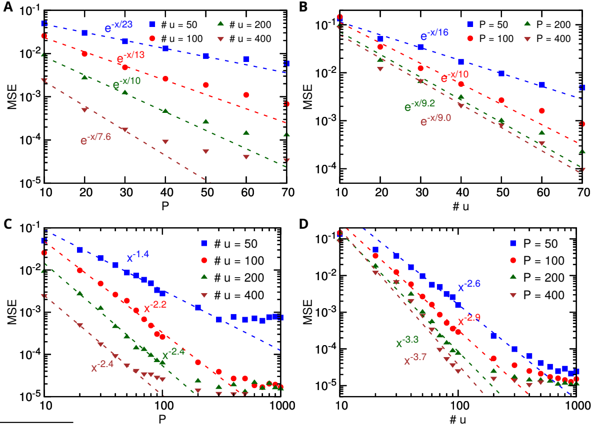
Implicit operator: Diffusion-reaction system

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \quad x \in [0,1], t \in [0,1]$$

$\#$ Training points $= \#u \times P$

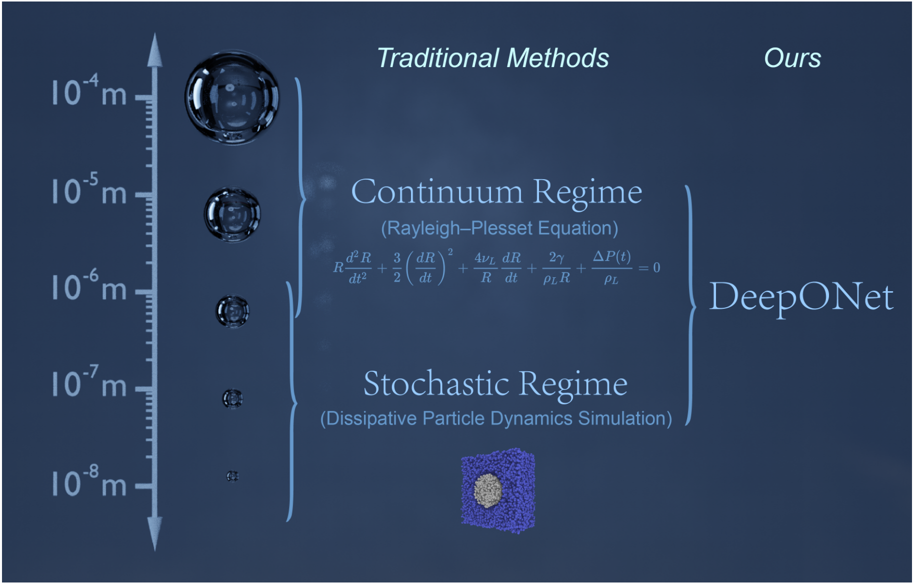
Small dataset: Exponential convergence

Large dataset: Polynomial convergence

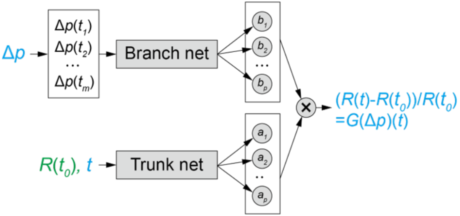


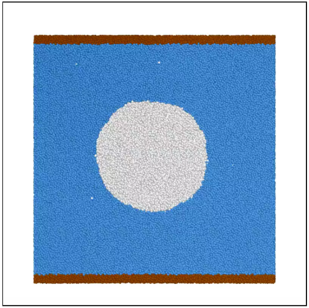
Lu et al., *Nature Mach Intell*, 2021

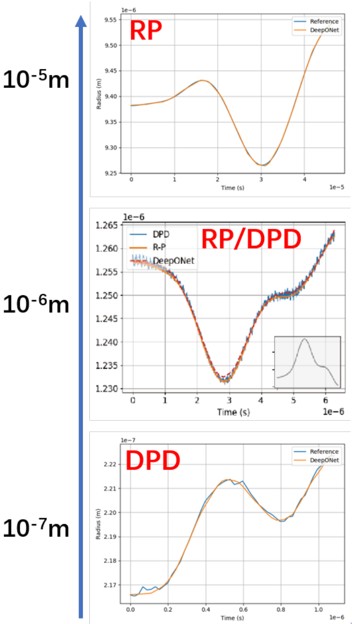
DeepONet for bubble growth dynamics



DeepONet for bubble growth dynamics

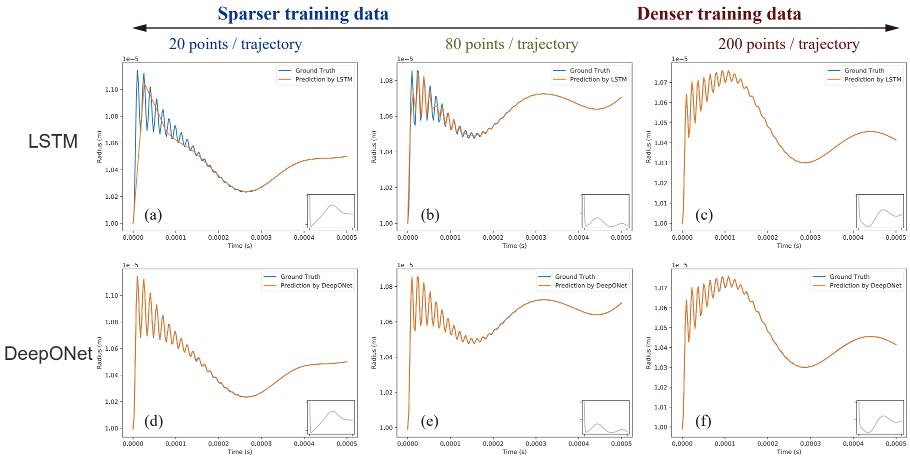






Lin et al., *J Chem Phys*, 2021

DeepONet vs. LSTM



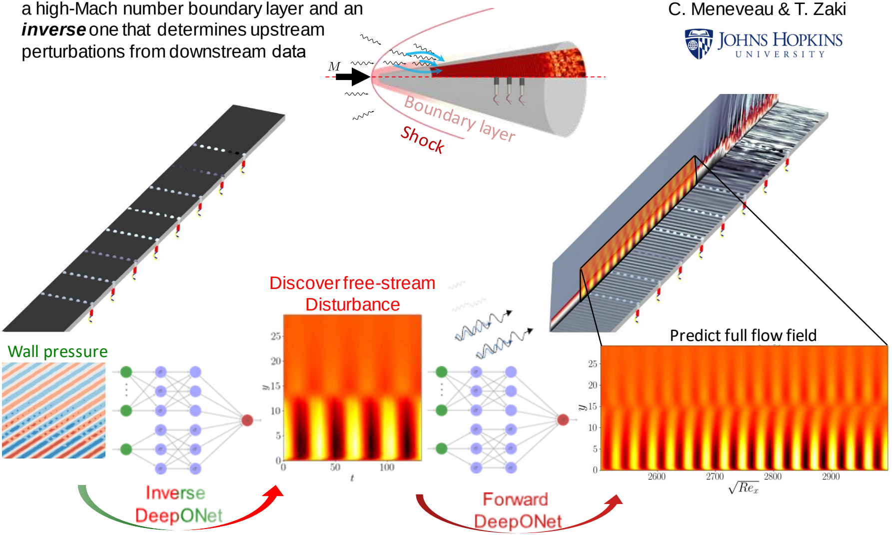
Lin et al., *J Chem Phys*, 2021

Data assimilation in high-Mach boundary layers

We trained a ***forward*** DeepONet that predicts the evolution of instability waves in a high-Mach number boundary layer and an ***inverse*** one that determines upstream perturbations from downstream data

In collaboration with
P. Clark Di Leoni, Y. Hao,
C. Meneveau & T. Zaki

JOHNS HOPKINS UNIVERSITY



Clark Di Leoni et al., arXiv:2105.08697

Multiphysics & Multiscale problems

Electroconvection problem

- Stokes equation

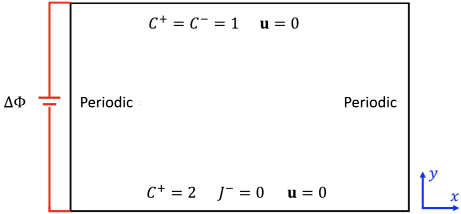
$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla p + \nabla^2 \mathbf{u} - \frac{\kappa}{2\epsilon^2}\rho_e \nabla \phi$$
$$\nabla \cdot \mathbf{u} = 0$$

- Electric potential

$$-2\epsilon^2 \nabla^2 \phi = \rho_e = z(c^+ - c^-)$$

- Ion transport equation

$$\frac{\partial c^\pm}{\partial t} = -\nabla \cdot \mathbf{J}^\pm$$
$$\mathbf{J}^\pm = c^\pm \mathbf{u} - \nabla c^\pm \mp c^\pm \nabla \phi$$



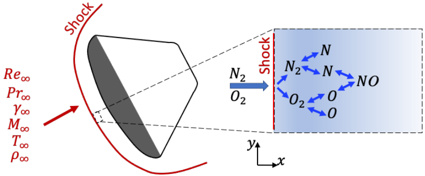
$$\Delta \phi \in [5, 75]$$

- $\mathbf{u}$: velocity
- p: pressure
- ϕ: electric potential
- ρ_e: free charge density
- $c^\pm$: cation/anion concentrations
- $\mathbf{J}^\pm$: flux

Cai et al., *J Comput Phys*, 2021

DeepM&Mnet: Multiphysics & Multiscale problems

Hypersonics with chemical reaction: fluid flow & chemical reaction



$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0$$

$$\frac{\partial \rho^s}{\partial t} + \frac{\partial}{\partial x_j}(\rho^s u_j) = \dot{w}^s, \quad s = 1, \dots, 5$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij}) = \frac{\partial \sigma_{ij}}{\partial x_j}, \quad i = 1, 2$$

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_j}[(E + p)u_j] = -\frac{\partial q_j}{\partial x_j} + \frac{\partial}{\partial x_k}(u_j \sigma_{jk})$$

Mao et al., arXiv:2011.03349

DeepONet for learning operators

DeepONet (**Lu** et al., *Nature Mach Intell*, 2021)

- **Algorithms & Applications**
  - 16 ODEs/PDEs (nonlinear, fractional & stochastic)
  - Multiphysics & Multiscale problems
    - Bubble growth dynamics from nm to mm (Lin et al., *J Chem Phys*, 2021)
    - Linear instability waves in high-speed boundary layers (Clark Di Leoni et al., arXiv:2105.08697)
    - Electroconvection (Cai et al., *J Comput Phys*, 2021)
    - Hypersonics (Mao et al., arXiv:2011.03349)
- **Observation**: Good performance
  - Small generalization error
  - Exponential/polynomial error convergence
- **Theory**
  - Convergence rate for advection-diffusion equations (Deng et al., arXiv:2102.10621)

Open-source software: **DeepXDE**

nature

NEWS FEATURE · 20 JANUARY 2021

Ten computer codes that transformed science

From Fortran to arXiv.org, these advances in programming and platforms sent biology, climate science and physics into warp speed.

Search projects

DeepXDE 0.13.3

Physics-informed deep learning

> 100,000 downloads, 580 GitHub Stars

BROWN

University of Victoria

PUC RIO

Los Alamos NATIONAL LABORATORY

Sandia National Laboratories

Ansys

SIEMENS

Other topics

- Uncertainty quantification (UQ)
- Tackling high dimensionality
- Connections between machine learning and classical numerical methods
- Understanding deep learning from physics
- Current limitations
  - Multiscale and multiphysics problems
  - Nonlinear and nonconvex optimization of networks
  - Data generation
  - Mathematics of theoretical foundation

PhD & Postdoc Opening

Looking for PhD students and Postdocs to work on scientific machine learning. Please feel free to contact me with CV (and/or transcripts, sample publications) attached if you are interested.

Penn Engineering
UNIVERSITY of PENNSYLVANIA

- Personal website: `https://lululxvi.github.io`
- Chemical and Biomolecular Engineering (CBE)
- Applied Mathematics and Computational Science (AMCS)
- Email: lulu1@seas.upenn.edu