Introduction to Scientific ML : Day 3

By Juan Felipe Osorio Ramirez

MindLab
Department of Computer Systems and Industrial Engineering, UN
Summer 2023

UNIVERSITY of WASHINGTON

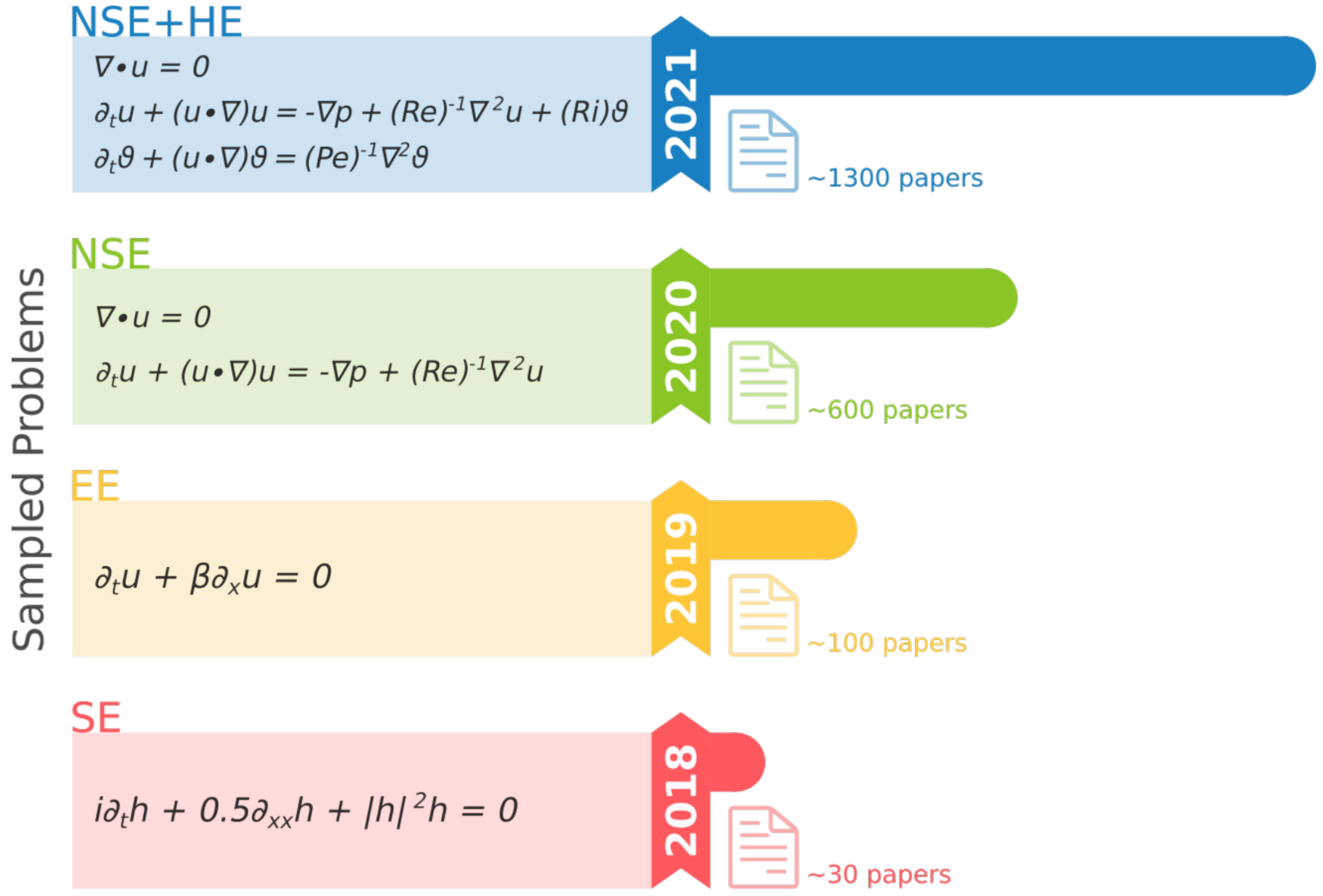
Physics-informed machine learning

Using Neural Networks

Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next

Salvatore Cuomo[1], Vincenzo Schiano Di Cola[2*], Fabio Giampaolo[1], Gianluigi Rozza[2], Maziar Raissi[3] and Francesco Piccialli[1*]

[1*]Department of Mathematics and Applications "R. Caccioppoli", University of Naples Federico II, Napoli, 80126, Italy .
[2*]Department of Electrical Engineering and Information Technology, University of Naples Federico II, Via Claudio, Napoli, 80125, Italy .
[2]SISSA – International School for Advanced Studies, Street, City, 10587, Italy.
[3]Department of Applied Mathematics, University of Colorado Boulder, Street, Boulder, 610101, United States.



Open source libraries

For Physics informed ML

Lu Lu

https://github.com/lululxvi

Lu Group
University of Pennsylvania

DeepXDE

Physics informed Neural Networks

- Solving integral equations.
- Solving fractional PDEs.
- Solving SPDEs

Deep Operator Network

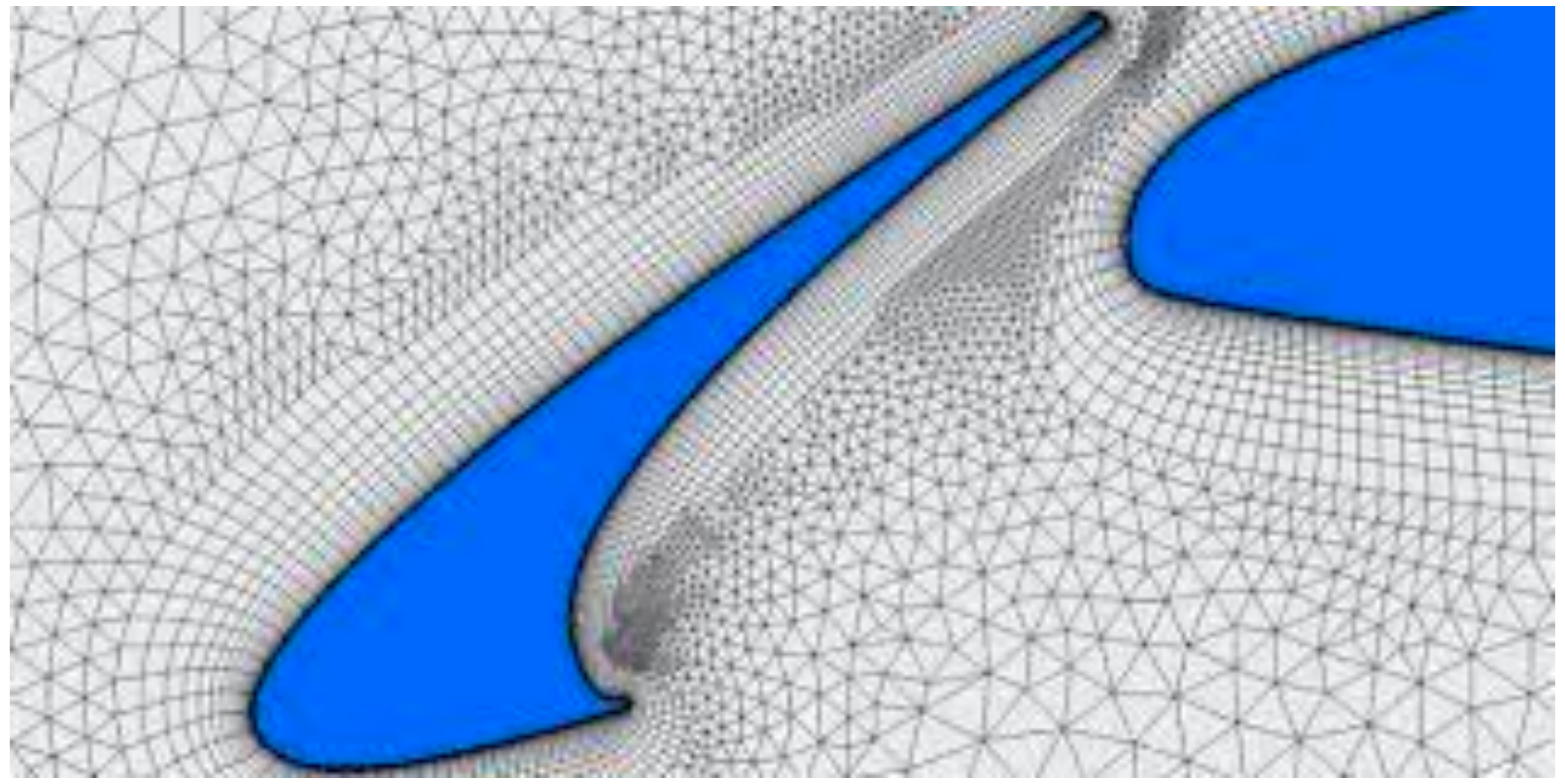
Multifidelity Neural Network

See also page 55 in arXiv:2201.05624

Numerical solutions of PDEs

Some issues

- Noisy observations as inputs.
- Mesh generation is costly
- High dimensional problems cannot be tackled
- Solving inverse problems is expensive



Machine learning comes to attack these problems !

Taken from: https://lululxvi.github.io/files/talks/2020SIAMMDS_MS70.pdf

Physics informed neural networks

Raissi, Perdikaris, Karniadakis (2019) JCP

Physical problem (PDE)

PDE $$\frac{\partial y}{\partial t} = \frac{\partial^2 y}{\partial x^2} - e^{-t}(sin(\pi x) - \pi^2 sin(\pi x)) \quad (x, t) \in \Gamma$$

IC $$y(x,0) = sin(\pi x)$$

BC $$y(-1,t) = y(1,t) = 0 \quad t \in T$$

Problem considered in the spacio-temporal domain $\Gamma = \Omega \times T = [-1,1] \times [0,1]$.

Numerical solution

$$\longrightarrow \quad y^M \approx y \text{ at } (x, t) \in \Gamma$$

Neural Network

Maziar
Raissi

Paris
Perdikaris

George
Karniadakis

Collocation methods

PINNs, kernel methods are also part of this !

To solve the problem $\begin{cases} Pu(\underline{x}) = f(\underline{x}), & \underline{x} \in \Omega \\ u(\underline{x}) = 0 & \underline{x} \in \partial\Omega \end{cases}$

Polynomials, trigonometric basis, etc

Family of methods that parametrize the solution u as

$$u^n(\underline{x}) = \sum_{j=1}^{n} \alpha_j \psi_j(\underline{x})$$

Then solve for the system of equations

$$\begin{cases} P\left(u^n\right)\left(\underline{x}_i\right) = \sum_{j=1}^{n} \alpha_j P\psi_j\left(\underline{x}_i\right) = f\left(\underline{x}_i\right), & \underline{x}_i \in \Omega \\ u^n\left(\underline{x}_i\right) = 0 & \underline{x}_i \in \partial\Omega \end{cases} \longrightarrow \underline{\underline{A}}\,\underline{\alpha} = \underline{f}$$

where $X = \left\{\underline{x}_1, \ldots, \underline{x}_m\right\} \subset \bar{\Omega}$ is a set of collocation points.

Method

Main idea

Rearrange to leave zero on the RHS

To solve the problem

PDE $$\frac{\partial y}{\partial t} - \frac{\partial^2 y}{\partial x^2} + e^{-t}(sin(\pi x) - \pi^2 sin(\pi x)) = 0 \quad (x, t) \in \Gamma$$

IC $$y(x,0) = sin(\pi x)$$

BC $$y(-1,t) = y(1,t) = 0 \quad t \in T$$

Main idea: Parametrize its solution with a neural network and constrain it to satisfy the PDE, IC and BC in the loss function.

$$\underbrace{NN(x, t)}_{\text{Feedforward}} \approx y(x, t)$$

Method

How to satisfy the PDE ?

Notice that we can calculate any gradient of the function $NN(x, t)$ via **automatic differentiation,** say $\frac{\partial^2}{dx^2}NN(x, t), \frac{\partial}{dt}NN(x, t)$, etc.

Then

$$\left(\frac{\partial NN}{\partial t} - \frac{\partial^2 NN}{\partial x^2}\right) + e^{-t}(sin(\pi x) - \pi^2 sin(\pi x)) \approx \left(\frac{\partial y}{\partial t} - \frac{\partial^2 y}{\partial x^2}\right) + e^{-t}(sin(\pi x) - \pi^2 sin(\pi x)) = 0$$

Define the function

$$f(t, x) = \left(\frac{\partial NN}{\partial t} - \frac{\partial^2 NN}{\partial x^2}\right) + e^{-t}(sin(\pi x) - \pi^2 sin(\pi x))$$

If $f \to 0$ for any $(x, t) \in \Gamma$ then our NN respects the PDE.

Method

Let's build the loss function !

We evaluate our PDE in a certain number of collocation points $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ inside our domain Γ, to build the loss w.r.t f

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \longrightarrow \text{Respects the PDE}$$

Training data $\left\{t_u^i, x_u^i, y^i\right\}_{i=1}^{N_u}$ will be sampled at locations where BC and IC hold, i.e., at $\partial\Gamma$. Then define

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |y(t_u^i, x_u^i) - NN(t_u^i, x_u^i)|^2 \longrightarrow \text{Respect IC and BC}$$

To get the **total loss function**

$$MSE = MSE_u + MSE_f$$

Theoretical guarantee

NNs are universal approximators

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE

University of California, San Diego

(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

Theorem 2.2

For every continuous nonconstant function G, every r, and every probability measure μ on (R^r, B^r), $\Sigma\Pi^r(G)$ is ρ_μ-dense in M^r $\square$

In other words, single hidden layer $\Sigma\Pi$ feedforward networks can approximate any measurable function arbitrarily well, regardless of the continuous nonconstant function G used, regardless of the dimension of the input space r, and regardless of the input space environment μ. In this precise and satisfying sense, $\Sigma\Pi$ networks are universal approximators.

An interesting future direction

Using kernels...

When and why PINNs fail to train: A neural tangent kernel perspective

Sifan Wang [a], Xinling Yu [a], Paris Perdikaris [b,*]

[a] *Graduate Group in Applied Mathematics and Computational Science, University of Pennsylvania, Philadelphia, PA 19104, United States of America*

[b] *Department of Mechanichal Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104, United States of America*

Authors use the idea of **Neural Tangent Kernel** of the trained PINN to understand better its convergence properties !

Neural Tangent Kernel: Convergence and Generalization in Neural Networks

Arthur Jacot, Franck Gabriel, Clément Hongler

Let’s run the demo