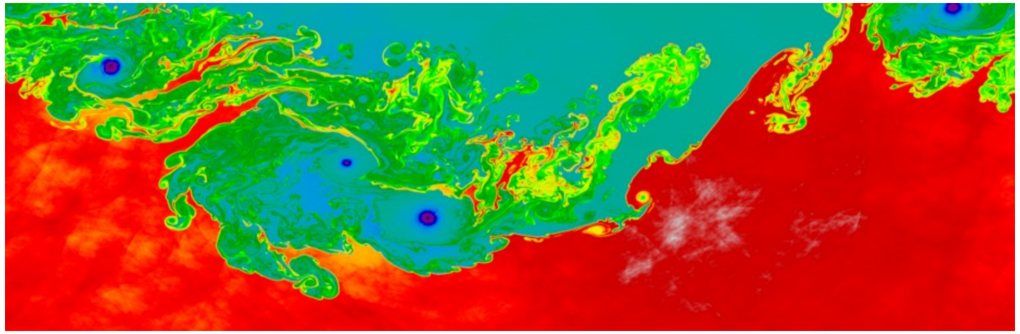
Physics-Informed Neural Networks (PINNs)

Ben Tapley

SINTEF

Outline

1. Data-driven neural network approach to solving differential equations (DEs)
2. Physics-informed neural network approach for DEs
3. Applications and limitations
4. Coding exercise



[HTML] **Physics-informed neural networks**: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M Raissi, P Perdikaris, GE Karniadakis - Journal of Computational physics, 2019 - Elsevier

… We introduce **physics-informed neural networks** – **neural networks** that are trained to solve supervised learning tasks while respecting any given laws of physics described by general …

Neural networks recap

- A neural network can be viewed as a function

$$u^{\theta} : R^{n} \rightarrow R^{m}$$

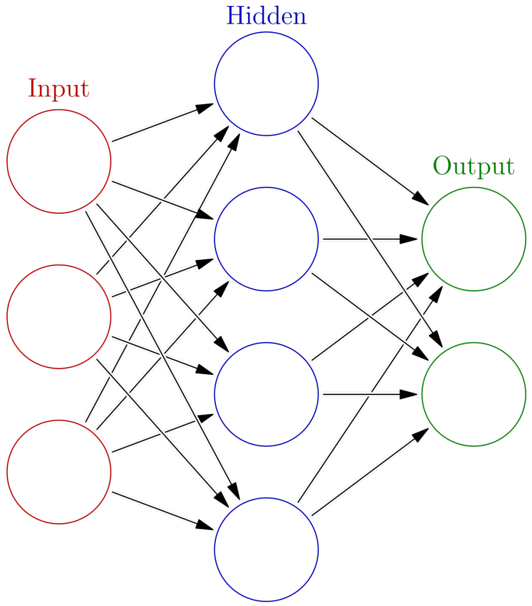
that depends on trainable parameters θ.

- Given observations $(x_i,\ t_i) \in X$ and their labels $u_i \in Y$ then we can train u^{θ} by minimising the loss function

$$\mathrm{L} = \sum |u^{\theta}(x_i, t_i) - u_i|$$

- Neural networks u_{θ} are *universal approximators*

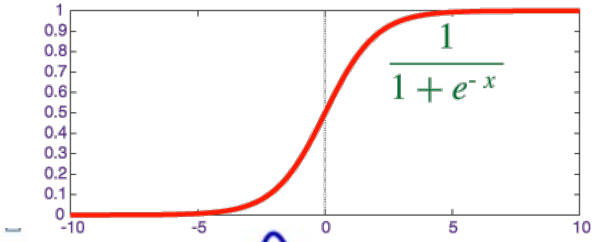
$$\mathrm{E.g.}: u^{\theta} : R^{3} \rightarrow R^{2}$$

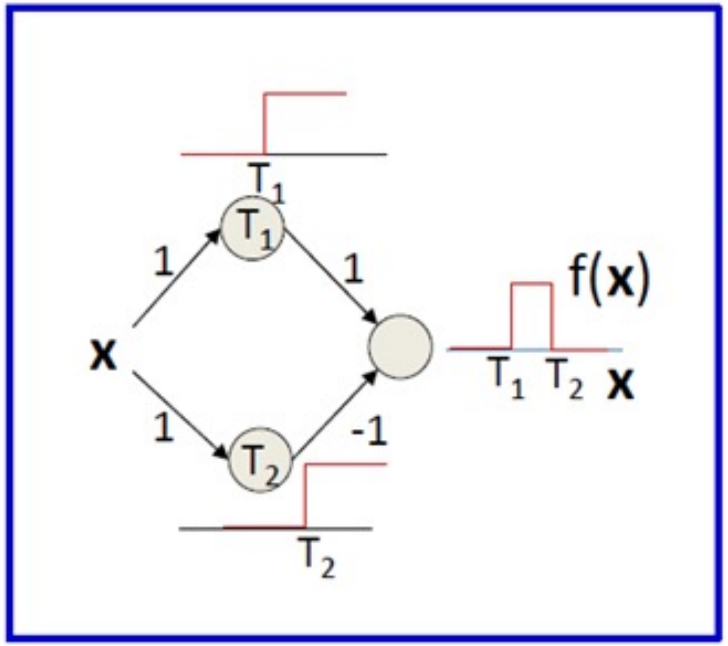


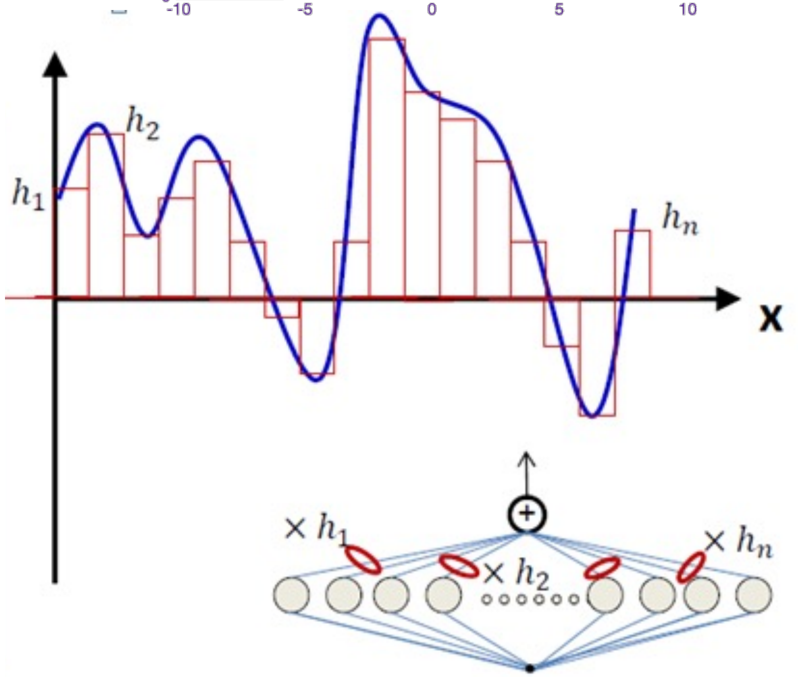
Neural networks recap

Single layer: $f_i(x) = \sigma(W_i x + b_i)$
Multiple layers: $y = f_n \circ \cdots \circ f_1(x)$
Universality:

Sigmoid activation $\sigma(x)$





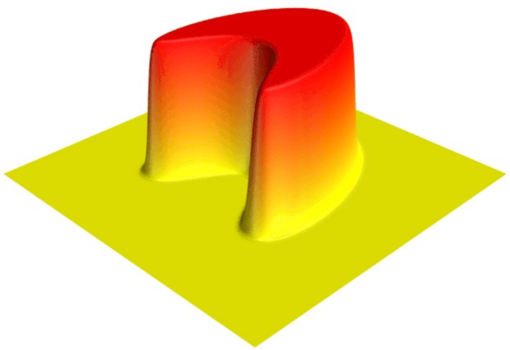


https://medium.com/analytics-vidhya/neural-networks-and-the-universal-approximation-theorem-e5c387982eed

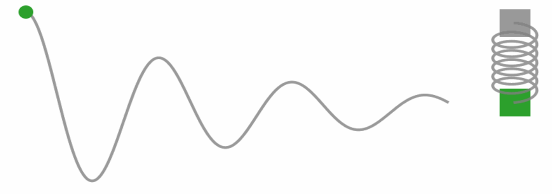
Solving differential equations

- A differential equation (DE) can be expressed as
$$DE[u, \lambda] = 0$$
- Examples:
  - PDE: heat equation $DE[u,\ \lambda] = u_t - \lambda\, u_x$, where $u(x, t)$ is a scalar temperature field.
  - ODE: dampened oscillator $DE[u,\ \lambda] = \ddot{u} + \lambda_1 \dot{u} + \lambda_2 u$, where $u(t)$ is the amplitude of the mass.
- The forward problem (traditional numerics): Given $DE[u, \lambda]$, solve for u.
  - Is feasable when we know all the physics.
  - That is, we know $DE[u,\ \lambda]$ and all parameters λ
- Problem: what if we have ***incomplete knowledge of physics***
  - Example: we know $DE[u,\ \lambda]$ is the heat equation, but we don't know the thermal diffusivity constant λ

Heat equation



Dampened oscillator

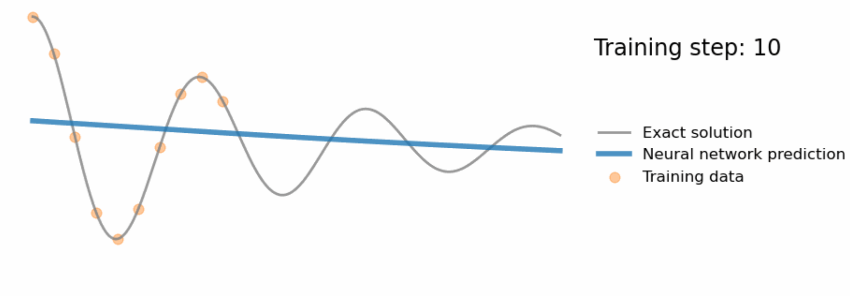


Potential solution: data-driven approach

- Given incomplete knowledge of physics, and lots of labeled data $(x_i, t_i), u_i \in X \times Y$, we can take a ***traditional data-driven approach*** to train $u^{\theta}(x, t)$.
- This is done by optimising the loss function for θ:

$$L = \sum |u^{\theta}(x_i, t_i) \ - u_i|$$

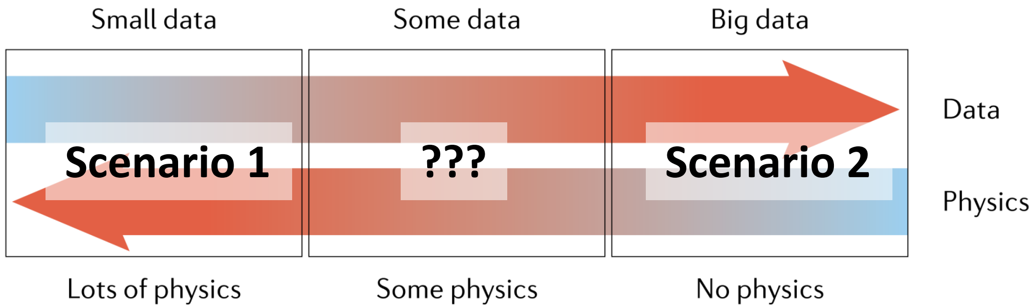
- This approach can suffice given enough data, but usually leads poor generalisation, when making predictions on unseen data.
  - Especially in the common case where we have sparse data!
- Why? u_{θ} has ***no knowledge of the underlying physics*** that generates the data!



https://benmoseley.blog/blog/

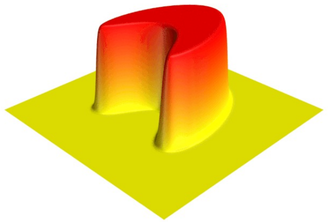
Some physics and some data

- We have outlined two scenarios for solving PDEs:
  1. Forward problem: complete knowledge of physics, no data.
     - Traditional numerics: given the DE $DE[u,\lambda]=0$ and initial conditions, solve for $u(x,t)$.
  2. Inverse problem: no knowledge of physics, lots of data.
     - Traditional learning: given enough data $\{(x_i,t_i),u_i\}_{i=1}^{N}$, solve for $u(x,t)$.
- What about a third scenario where we have some physics and some data?



Karniadakis, George Em, et al. "Physics-informed machine learning." *Nature Reviews Physics* 3.6 (2021): 422-440.

Physics informed neural networks (PINNs)



- What does it mean to have some physics and some data?
- Example: The heat equation $\mathrm{u_t}(x,t) = \lambda\, u_x(x,t)$, where λ is unknown
  - We know the structure of the DE $DE[u,\lambda]$, but not all the physical constants λ
  - We also have some observational data $(x_i, t_i), u_i$
  - Here we should be able to use the data to infer what λ must be!
- How can we use the data plus the PDE to find u? Train it on the ***informed*** loss function

$$L \quad = \quad \sum |u^\theta(x_i, t_i) - u_i| \quad + \quad \sum |\mathrm{u}_\mathrm{t}^\theta(x_i,\, t_i) - \lambda \mathrm{u}_\mathrm{x}^\theta(x_i,\, t_i)|$$

PINN Loss = **Traditional, data-driven loss** + **Additional physics loss**

Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational physics* 378 (2019): 686-707.

Physics informed neural networks (PINNs)

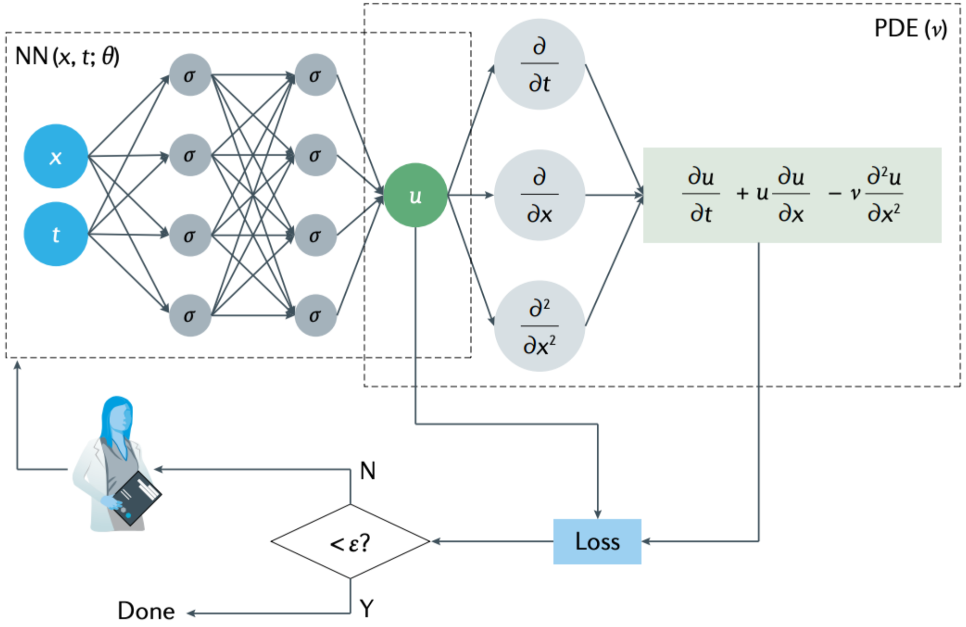
- In general, for $DE[u, \lambda] = 0$ the loss function

$$L = \sum |u^\theta(x_i, t_i) - u_i|^2 + \sum |DE[u^\theta(x_i, t_i), \lambda]|^2$$

Is (weakly) constraining the neural net to learn a form that satisfies

$$DE[u^\theta, \lambda] < tol$$

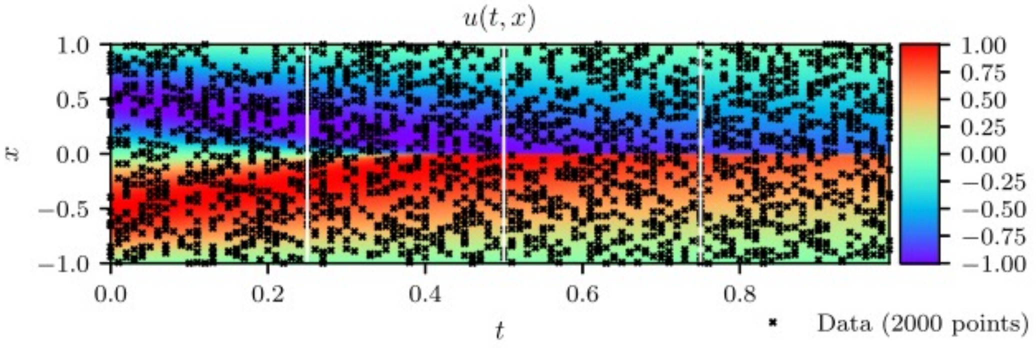
- The equation $DE[u^\theta, \lambda]$ can contain partial derivatives w.r.t time and space
  – This is computed using automatic differentiation



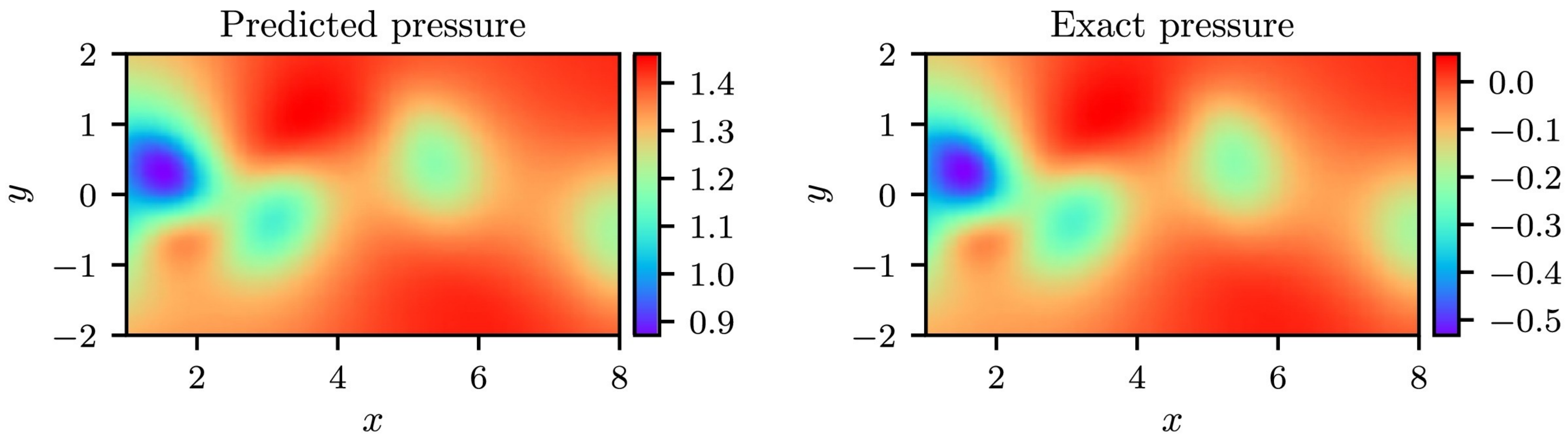
Karniadakis, George Em, et al. "Physics-informed machine learning." *Nature Reviews Physics* 3.6 (2021): 422-440.

Advantages of PINNs

- Data Efficiency
- Interpretable Models
- Incomplete models and imperfect data
- Strong generalization in small data regime
- Tackling high dimensionality
- Difficult geometries where meshing is not feasible



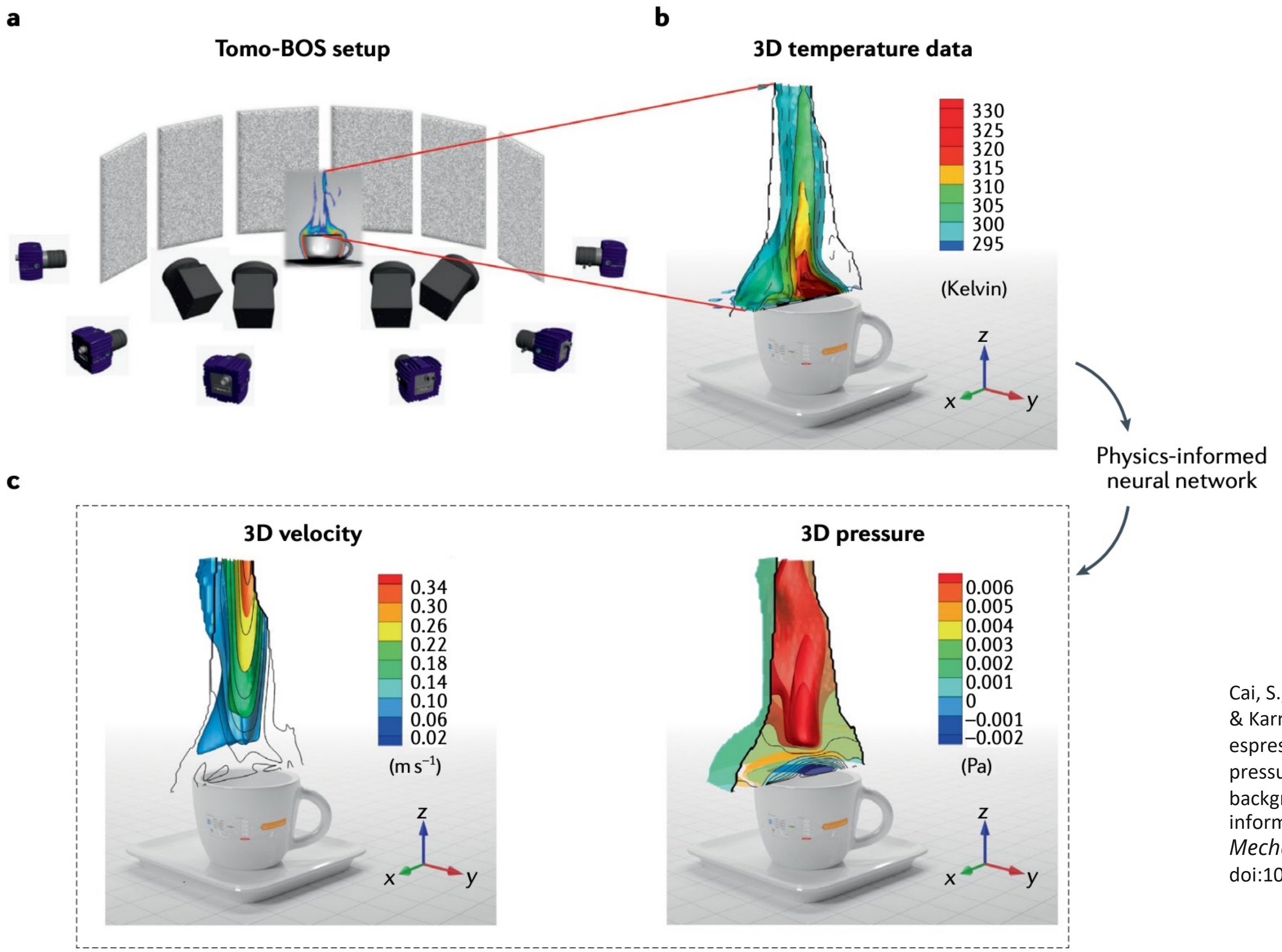
Applications – 2D Navier-Stokes toy problem



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

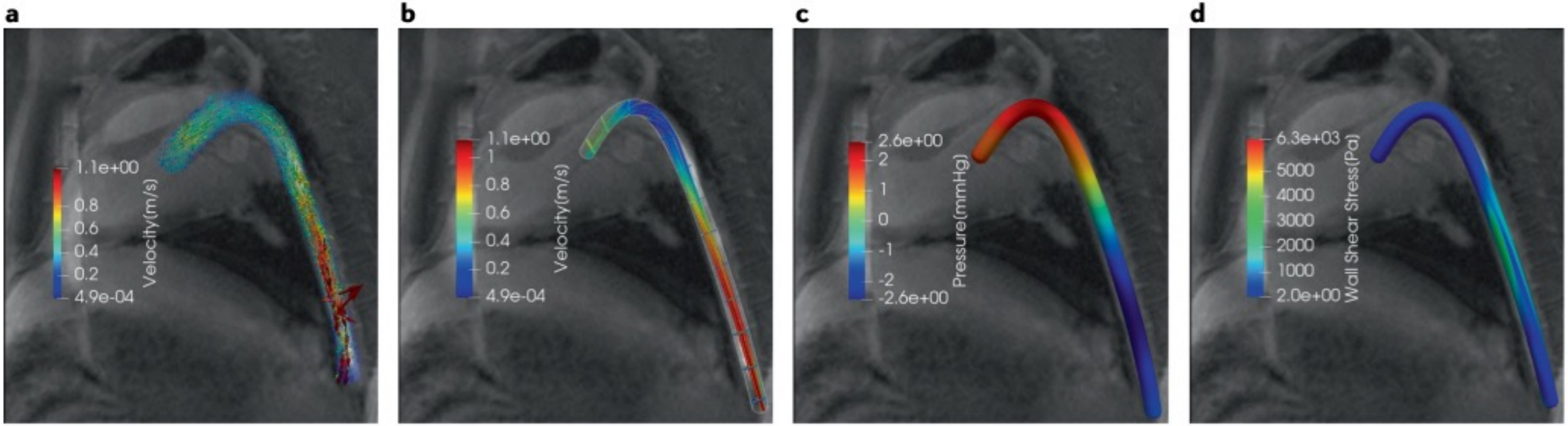
M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics

Applications – Flow over coffee cup



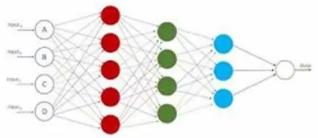
Cai, S., Wang, Z., Fuest, F., Jeon, Y., Gray, C., & Karniadakis, G. (2021). Flow over an espresso cup: Inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. *Journal of Fluid Mechanics, 915*, A102. doi:10.1017/jfm.2021.135

Applications – Flow through aorta



Software

- DeepXDE – Python - github.com/lululxvi/deepxde
  - Implementation from original authors
  - Supports: TensorFlow, JAX, PyTorch
  - DeepONets, MFNNs, PINNs



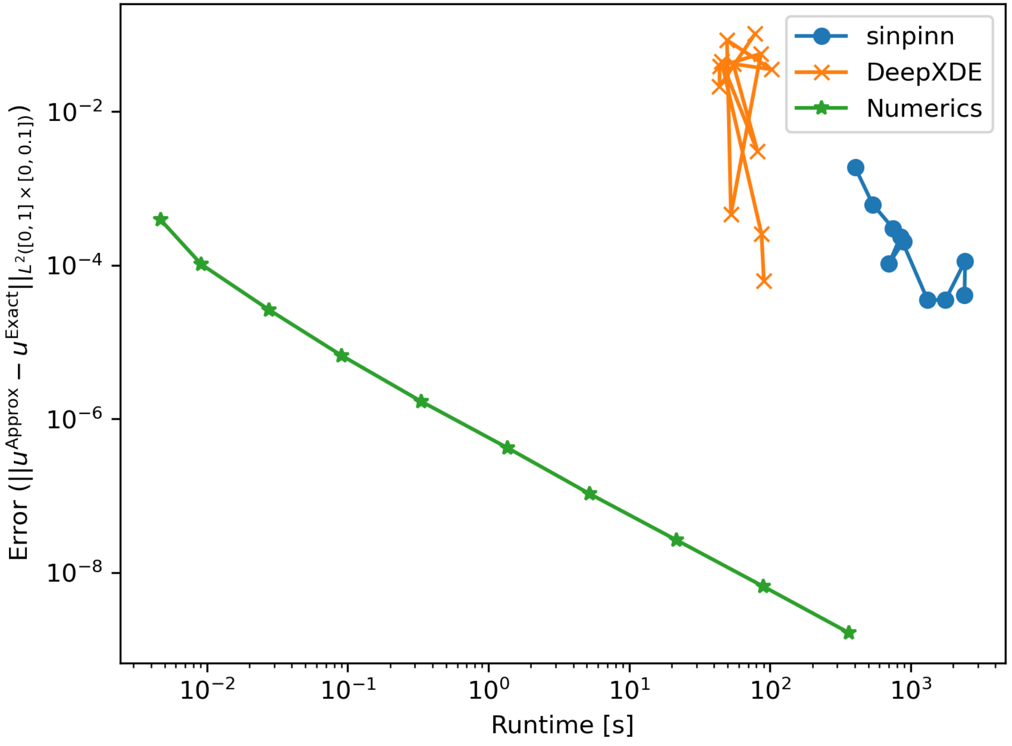
Others:

- PyDEns – Python, TensorFlow
- NeuroDiffEq – Python, PyTorch
- NeuralPDE - Julia

Challenges and limitations

- Computational complexity
- Requirement for physical laws
- “Weakly” enforces physical constraints.
- Multiscale and multi-physics problems
- Non-robust training and convergence

Comparison DeepXDE vs PyTorch vs Crank—Nicolson for 1D heat equation



Coding a PINN from scratch

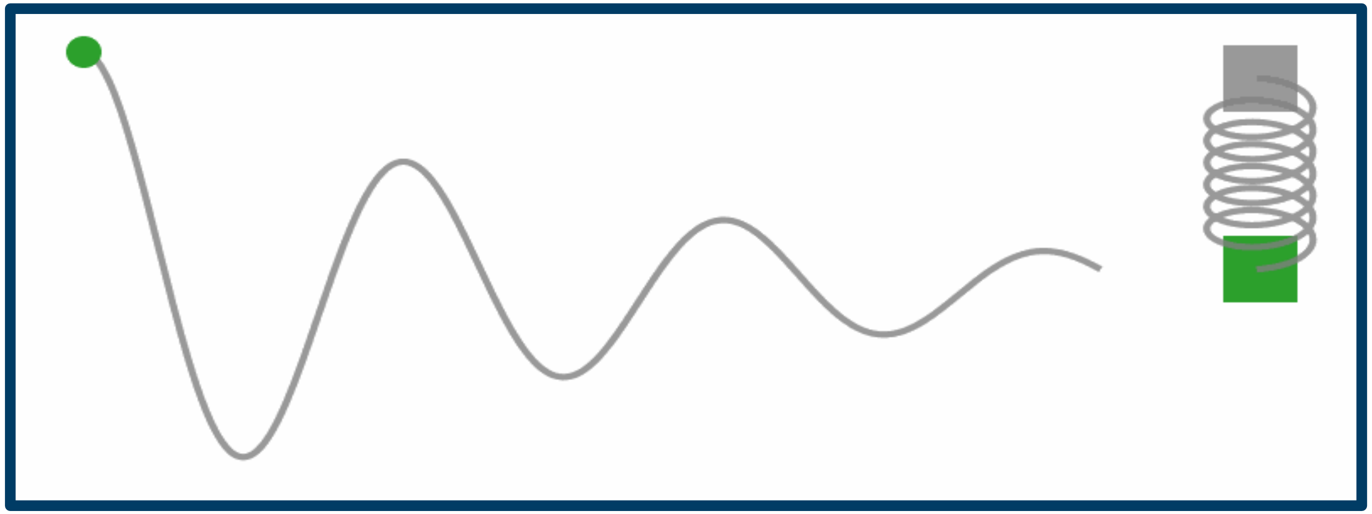
github.com/bentaps/PINN-example

- Simulate the dynamics of an oscillating spring with resistance.
- Use a data-driven, physics agnostic approach
- Inform the loss function about the physics (i.e., the ODE)
- First, assume μ and k are known
- Second, assume μ and k are unknown and learn them.
- Add noise
- Experiment with hyperparams

- ODE is given by

$$DE[x(t), (\mu, k)] = \ddot{x} + \mu\dot{x} + kx = 0$$

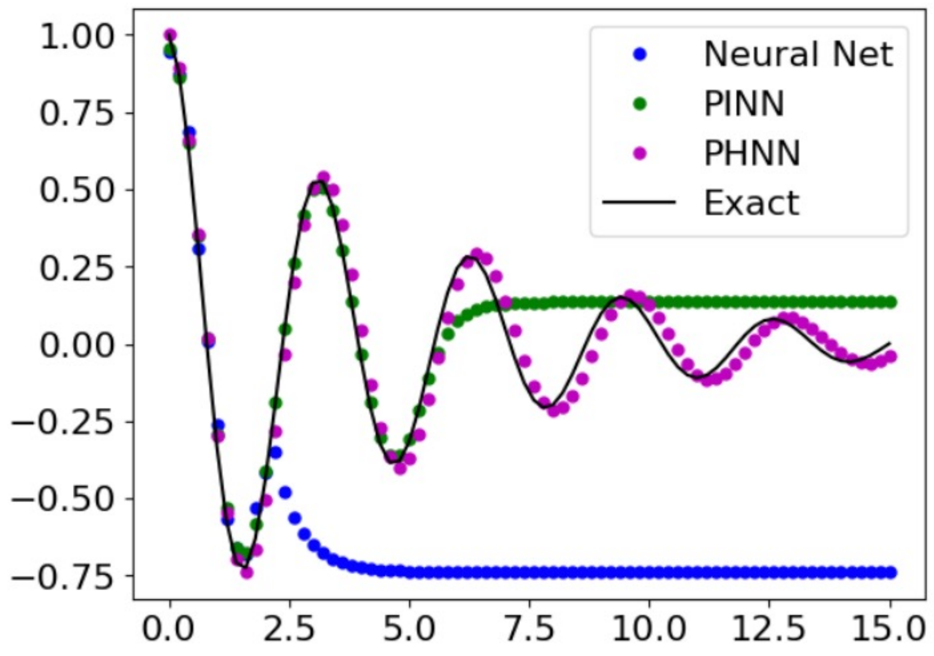
Where μ is the dampening constant and k is the spring constant.



https://benmoseley.blog/blog/

Pseudo-Hamiltonian Neural Nets

- An alternative to PINNs
- Symmetries and laws are enforced intrisically through network architechture
  - Exact preservation of physics (as opposed to “weak”)
- More data efficient
  - Does not require user-defined collocation points
- Simpler loss function
  - More efficient back prop.



Eidnes, Sølve, et al. "Pseudo-Hamiltonian neural networks with state-dependent external forces." *Physica D: Nonlinear Phenomena* 446 (2023): 133673