Physics-Informed Neural Networks: General Concept

International DIADEM Summer School

Chih-Kang Huang

Institut Jean Lamour, Université de Lorraine

August 27th 2025

Deep Learning: Black Box Approach

Example: Learn the solution of **1D wave equation** $\partial_{tt}^2 u = c^2 \partial_{xx} u$.
with initial condition u_0.

$$(t, x) \longrightarrow \boxed{\begin{array}{c}\text{Black Box}\\ \text{Neural Network}\end{array}} \longrightarrow u_\theta(t, x) \simeq u(t, x)$$

where θ is the set of **trainable parameters** of the neural network.
Thanks to the Universal approximation theorem (Cybenko '89, Hornik et al. '89, Chen & Chen '95, etc.), there exists θ such that

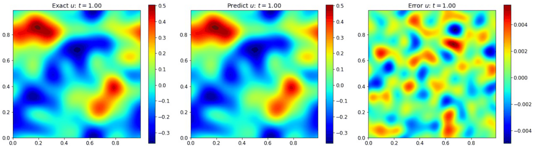
$$u_\theta(0, x) \;\approx\; u_0(x) \quad \text{and} \quad u_\theta(t, x) \;\approx\; u(t, x).$$

where $u(t, x)$ is the exact solution of the wave equation.

We train the neural network using a *data-driven loss*:

$$\mathcal{L}_{data} = \frac{1}{N_k}\sum_{i=1}^{N_k} \|u_\theta(t_i, x_i) - u_{data}(t_i, x_i)\|^2 + \frac{1}{N_I}\sum_{j=1}^{N_I} \|u_\theta(0, x_j) - u_0(x_j)\|^2$$

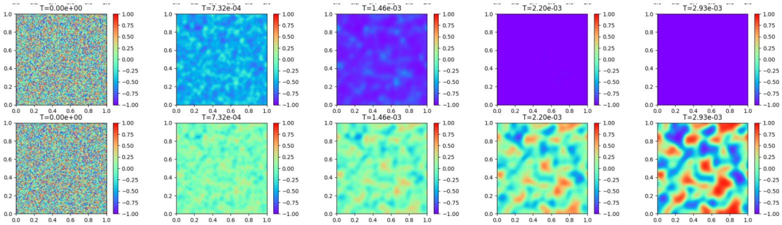
where $(u_0(x_j), u_{data}(t_i, x_i))$ are labeled data obtained by simulation or observation.



Left: True solution — Middle: NN prediction — Right: Absolute error

Black Box: Drawbacks

- Training requires lots of (well-prepared) data $(u_0(x_j), u_{data}(t_i, x_i)) \implies$ Risk of overfitting.
- Poor generalization to unseen cases (Operator learning).



Prediction of a heterogeneous mixture of two phases by NN

- Lack of interpretability: the NN may learn the wrong physical principles that only fit the training data.

Since data-driven learning has drawbacks, why not use **physical laws (PDEs)** to guide neural network training?

What we will discover

- The general concept of **Physics-Informed Neural Networks (PINNs)** for PDEs.
- **Adaptive sampling** and adaptive weighting for stability and accuracy in training.
- From PINNs to the broader paradigm of **Physics-Informed Machine Learning (PIML)**.

Outline

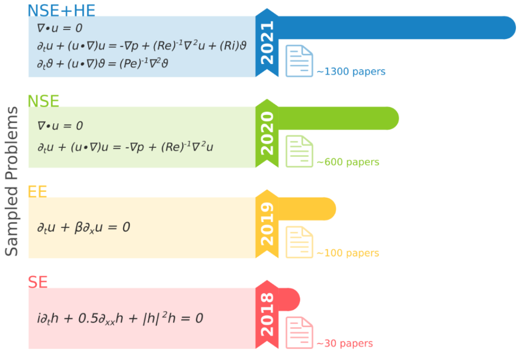
1. Physics-Informed Neural Networks
2. Hands-on Session

What are Physics-Informed Neural Networks (PINNs)?

- Neural networks that integrate **physical laws** (e.g., PDEs) into the training process.
- Loss function combines:
  - Data mismatch
  - PDE residuals
  - Boundary/initial conditions
- Applications: solving PDEs, inverse problems, scientific modeling.

A Short History of PINNs

- **1990s:** Early attempts at using neural networks for solving PDEs: Dissanayake et al.(1994), Lagaris et al.(1998), etc.
- **2019:** Raissi et al.,introduced the modern PINNs framework.
- **Today:** Widely applied in physics, engineering, and materials science.



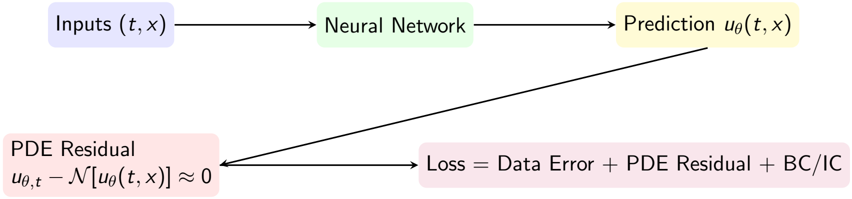
Numbers of papers related to PINNs (Cuomo et al.)

How do PINNs work?

Assume we wan't to solve

$$\partial_t u(t,x) = \mathcal{N}[u(t,x)]$$

where $\mathcal{N}$ is a *differential operator*.



Physics enters as a **soft constraint** in the training process.

In this talk, we focus on the **PDE part** only (no data) $\implies$ *unsupervised learning*

Physics-Informed Neural Networks (PINNs)

In 2019, Raissi et al* incorporate physical laws and constraints directly into the learning process:

Physics-Informed Loss

$$\mathcal{L}_{phys}(\theta) = \lambda_{IC}\,\mathcal{L}_{IC} + \lambda_{BC}\,\mathcal{L}_{BC} + \lambda_{PDE}\,\mathcal{L}_{PDE}$$

with $\lambda_{PDE}, \lambda_{IC}, \lambda_{BC} > 0$, where

$$\mathcal{L}_{PDE} = \frac{1}{N}\sum_{i=1}^{N}\left(\partial_t u_\theta(t_i, x_i) - \mathcal{N}[u_\theta(t_i, x_i)]\right)^2.$$

where $(t_i, x_i)_{i=1}^{N}$ are called *colocation points*.

Remark: This method is *mesh-free* — no grids are needed; training uses collocation points that can be adaptively updated.

*Raissi, Perdikaris, and Karniadakis 2019.

The derivatives in the PDE are often computed with *auto-differentiation* (Jax, Pytorch, Julia, etc.), but can also approximated by classical schemes or FFTs.
We minimize the total loss $\mathcal{L}_{phys}(\theta)$ using gradient-based optimization:

$$\theta^{(k+1)} = \theta^{(k)} - \eta \, \nabla_\theta \mathcal{L}_{phys}(\theta^{(k)}).$$

Key idea

Backpropagation computes $\nabla_\theta \mathcal{L}$, which includes contributions from PDE residuals $\implies$ the NN learns to satisfy the physical laws.

Remark : DO **NOT** use ReLU as activation function in PINNs !

For instance, consider the heat equation in 1D:

$$\begin{cases} \partial_t u(t,x) & = u_{xx}(t,x) \text{ on } (0,T] \times (-1,1) \\ u(0,x) & = h(x) \text{ on } (-1,1) \\ u(t,-1) & = u(t,1) = 0 \text{ on } (0,T] \end{cases}$$

Then we have

$$\mathcal{L}_{PDE} = \frac{1}{|\mathcal{I}_{PDE}|} \sum_{i \in \mathcal{I}_{PDE}} (\partial_t u_\theta(t_i, x_i) - \partial_{xx} u_\theta(t_i, x_i))^2$$

$$\mathcal{L}_{IC} = \frac{1}{|\mathcal{I}_{IC}|} \sum_{j \in \mathcal{I}_{IC}} (u_\theta(0, x_j) - h(x_j))^2, \quad \mathcal{L}_{BC} = \frac{1}{|\mathcal{I}_{BC}|} \sum_{k \in \mathcal{I}_{BC}} (u_\theta(t_k, \pm 1))^2$$

And then $\mathcal{L}_{phys}(u_\theta) = 0 \iff u_\theta$ is the solution.

Hands On: Examples with PINNs to the Heat Equation and the Allen–Cahn Equation.

PINNs and Curse of Dimensionality I

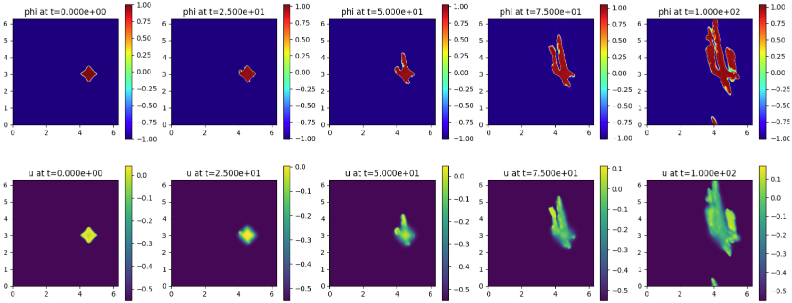
Classical Numerical Methods:

- Grid-based solvers (FEM, FDM, etc.) scale poorly with dimension d.
- Computational cost $\sim \mathcal{O}(N^d)$ (exponential growth).
- Known as the **Curse of Dimensionality**(CoD).

Why PINNs could overcome CoD :

- Neural network approximates $u(x)$ directly.
- Complexity depends on network size, not grid resolution.
- (De Ryck and Mishra (2022)) The size of NN grows at $O(poly(d))$ for nonlinear parabolic PDEs (Heat, Allen-Cahn, etc.)

Evalutation speed of a (trained) PINN is much more faster than classical schemes.
However, most of the time, PINNs for complex PDEs...



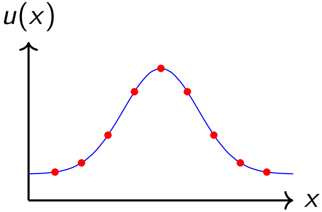
my first PINN

Motivation:

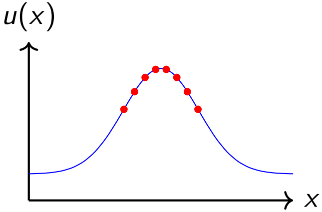
- Uniform sampling may waste resources in smooth regions.
- Some regions (e.g., sharp gradients, boundary layers) require denser sampling.

Idea: Dynamically adjust the sampling distribution based on PDE residuals.

Uniform Sampling



Adaptive Sampling



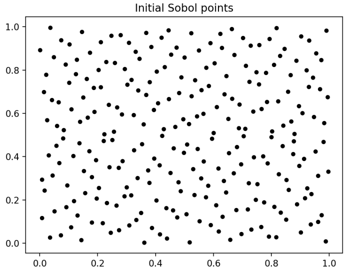
Benefits:

- Faster convergence and improved accuracy.
- Better resolution in critical regions.

R3 Sampling: Idea

Retain–Resample–Release (R3)

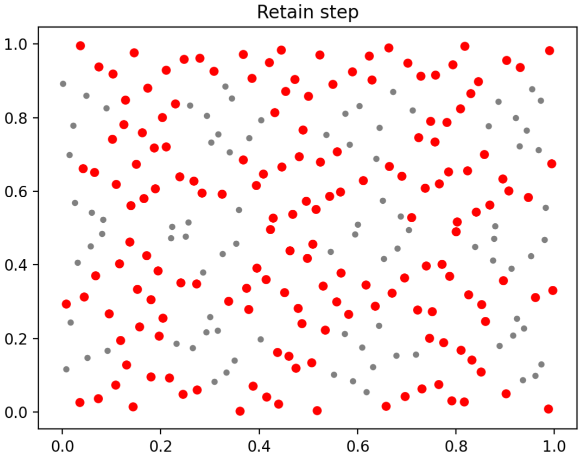
- **Retain**: keep points with high residuals (where $\mathcal{L}_{PDE}$ is "large")
- **Resample**: add new sampling points to explore the domain
- **Release**: discard points with consistently low residuals



Initial Sobol collocation points†

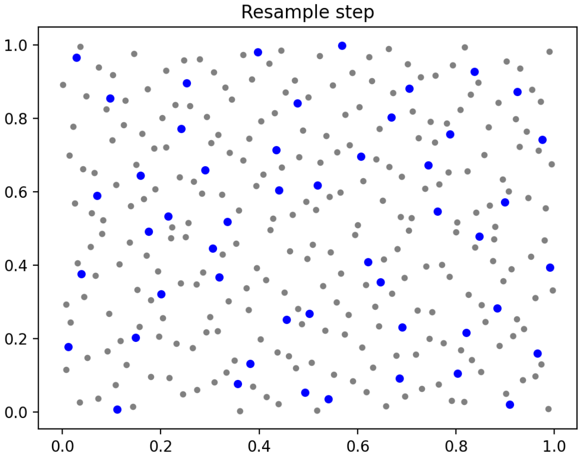
†Daw et al. 2022.

R3 Sampling: Retain Step

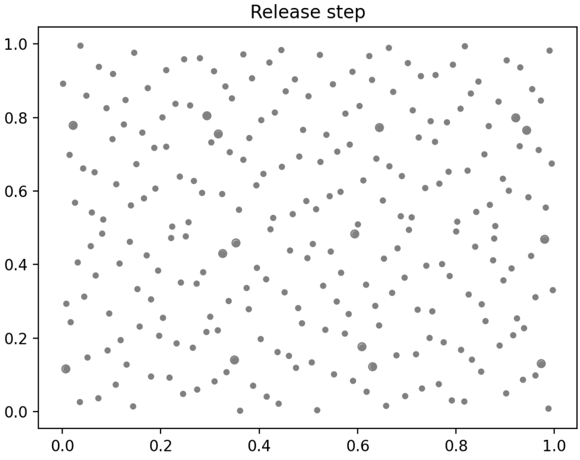


High-residual points are retained.

R3 Sampling: Resample Step

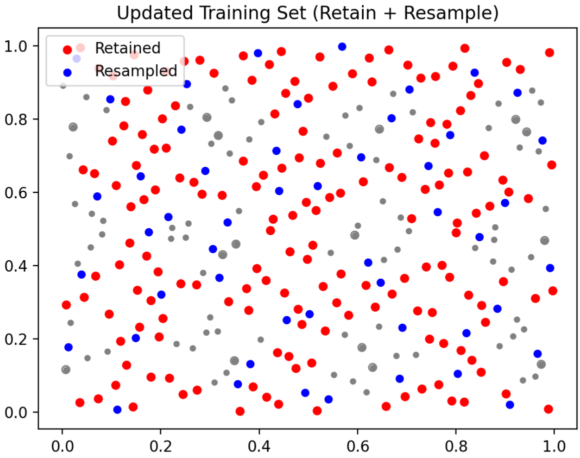


New Sobol points are resampled.



Low-residual points are released.

R3 Sampling: Updated Training Set



Final collocation set after one R3 cycle: retained + resampled points.

R3 example

Training Loop with R3

```
points = Sobol.sample(N)
for epoch in range(n_epochs):
    # Compute residuals
    residuals = PDE_residual(model, points),

    # Retain
    retain = points[residuals > tau_high]

    # Resample
    resample = Sobol.sample(M)

    # Release
    release = points[residuals < tau_low]

    # Update training set
    points = retain + resample
```

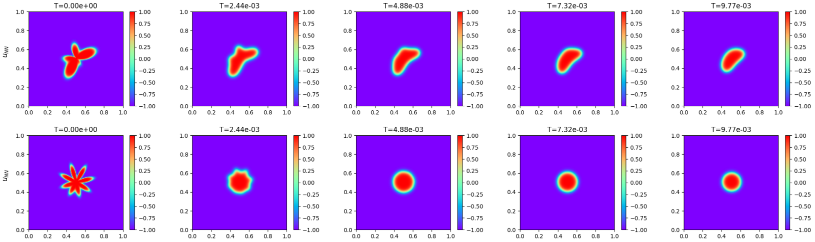
The (isotropic) Allen-Cahn equation is given by:

$$\partial_t \varphi = -\partial_\varphi F(\varphi)$$

where

$$F(\varphi) = \int \left(\frac{1}{2} |\nabla \varphi|^2 + \frac{W(\varphi)}{\epsilon^2} \right) \mathrm{dx}$$

with $\epsilon =$ thickness of the transition phase, $W(s) = (1 - s^2)^2/4$ double potential.



Perturbed disks evolving under Allen-Cahn equation

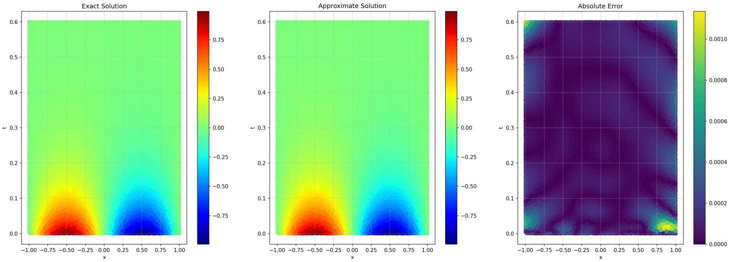
Adaptive Weighting in PINNs: Motivation

Problem:

- PINN loss combines multiple terms:

$$\mathcal{L} = \lambda_{IC}\mathcal{L}_{IC} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{PDE}\mathcal{L}_{PDE}$$

- These terms often have very different scales.
- Poorly balanced weights λ_i can lead to:
  - Slow convergence ; Training instability
  - Neglect of certain constraints



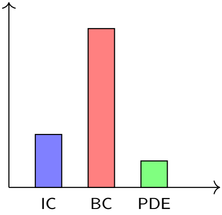
Idea: Adjust weights *adaptively* during training to balance contributions.

Solution: Adaptive Weighting

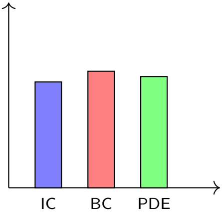
- Dynamically update λ_i during training.
- Methods: e.g Gradient-based normalization[‡]:

$$\lambda_{IC} \nabla_\theta \mathcal{L}_{IC} \simeq \lambda_{BC} \nabla_\theta \mathcal{L}_{BC} \simeq \lambda_{PDE} \nabla_\theta \mathcal{L}_{PDE}$$

Before (Imbalance)



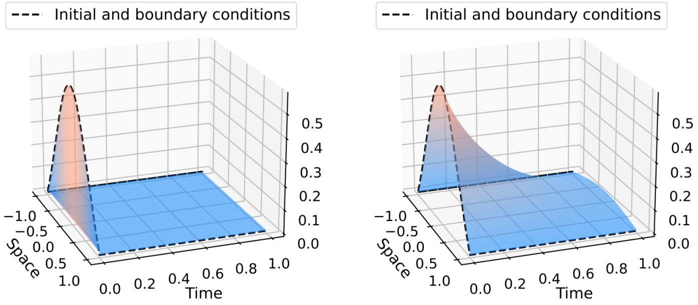
After (Balanced)



[‡] Xiang et al. 2022.

PINNs can "overfit" !

PINNs can *cheat*, 1D heat equation :



Inconsistent PINN (left) compared to the true solution (right) for the heat propagation case (cf. Doumèche et al., '23, Sorbonne University)

Solution: L^2-regularization on trainable parameters of NN

$$\mathcal{L}_{reg} = \mathcal{L}_{phys} + \lambda_\theta \|\theta\|_2^2.$$

Data-driven & PINNs : limitations

Data-driven: minimizing the discrepancy prediction-data

- $\oplus$ Easy implementation
- $\ominus$ Lots of data required [Oomen et al.'24]
- $\ominus$ Energy unstability
- $\ominus$ Lack of interpretability

Physics-informed: minimizing PDE residuals

- $\oplus$ Unsupervised Learning
- $\oplus$ Better generalization
- $\ominus$ Lack of convergence and error bounds.
- $\ominus$ Computationally intensive

$\Longrightarrow$ *Hybrid* approaches combine data-driven and physics-informed neural networks.

PINNs: Pros and Cons

Advantages

- **Unsupervised**, no training data required
- **Mesh-free**: adaptive collocation points
- Fast evaluation once trained
- Less prone to curse of dimensionality

Disadvantages

- Lack of theoretical guarantees (convergence, error bounds)
- High computational cost during training
- Retrain NN once the IC/BC or some parameters in PDE change

How can we integrate physical knowledge into machine learning while minimizing the drawbacks of PINNs?

Physics-Informed Machine Learning

A broader paradigm that integrates physical knowledge into machine learning $\implies$ Physics-Informed Machine Learning (PIML)

- **Operator Learning** : Construct an operator G : $u \mapsto v$ mapping between functional spaces .
- **Inverse Problems** (e.g. Tomography, Data-driven discovery of physics)
- **Residual Modeling** (Enhancement of numerical schemes)
- **Downstream tasks** (e.g. Airfoil Optimizations, Image reconstruction)

Alternatives/Improvements over PINNs: Deep Ritz method, Deep Galerkin method.

Operator Learning

Idea

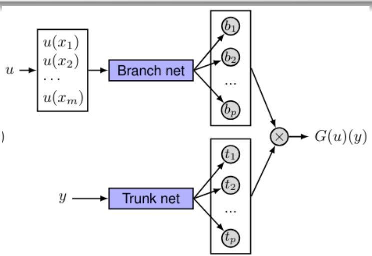
Instead of learning a single solution $u(t, x)$, **Operator Learning** trains neural networks to approximate the mapping:

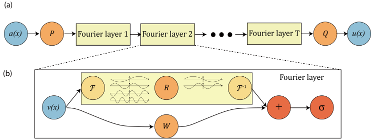
$$\mathcal{G}_\theta : f \;\mapsto\; u,$$

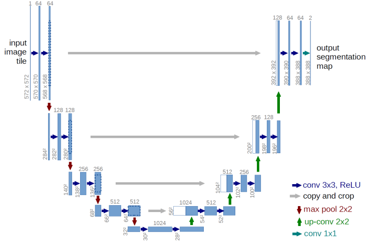
where f may represent coefficients, forcing terms, or initial/boundary conditions.

- Learns families of PDE solutions, not just one instance
- Generalizes across different inputs and geometries
- Provides fast PDE solvers once trained

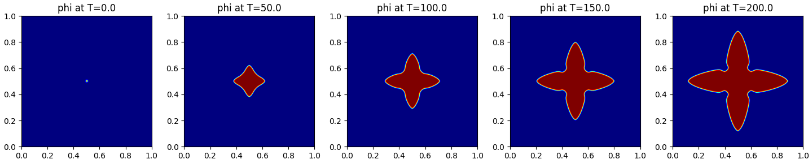
Illustrations: Operator Learning Architectures







- **DeepONet** – branch/trunk networks for nonlinear operators
- **FNO** – spectral representation in Fourier space
- **U-Net** – convolutional-based NN



Phase-field simulation of dendritic growth in a pure melt

L^2-gradient flow of the free energy E

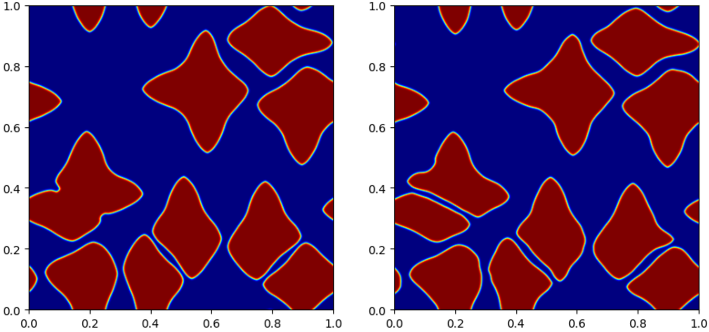
$$E(\varphi, U) = \int \left(\frac{1}{2} [a(\mathbf{n})]^2 |\nabla \varphi|^2 + \frac{W(\varphi)}{\varepsilon^2} + \frac{\lambda}{\varepsilon} h(\varphi) U \right) \mathrm{dx},$$

with $\varphi =$ phase field solution, $U =$ temperature field,
$a(\mathbf{n}) = 1 + \sigma \cos(m\theta)$ anisotropy strength.

$$\begin{cases} \tau \partial_t \varphi = -\partial_\varphi E(\varphi, U), \\ \partial_t U = D \Delta U + \frac{1}{2} \partial_t \varphi. \end{cases} \tag{1}$$

Evalutation on multiple grains

(Huang, Appolaire, Gagnon, Založnik, '25) Train the NN with 1-3 grains, then evaluate on multi-grains:



Spectral method v.s Neural Network

$\Longrightarrow$ Up to 3 to 4x speed-ups compared to spectral method.

Goal

Estimate unknown parameters or coefficients in a PDE from sparse observations.

- Example: Identify diffusion coefficient $\kappa(x)$ in

$$\partial_t u = \kappa(x)\,\Delta u, \quad u|_{\partial\Omega} = 0$$

from limited $u(x, t)$ measurements.

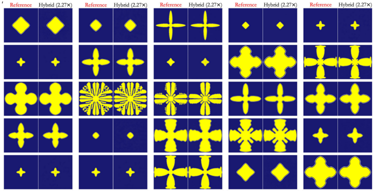
- PINN loss combines data misfit and PDE residual:

$$\begin{aligned}\mathcal{L}(\theta, \kappa) = \lambda_{data} \sum_i \|u_\theta(x_i, t_i) - u_{obs}(x_i, t_i)\|^2 \\ + \lambda_{PDE} \sum_j \|\partial_t u_\theta - \kappa(x_j)\Delta u_\theta\|^2\end{aligned}$$

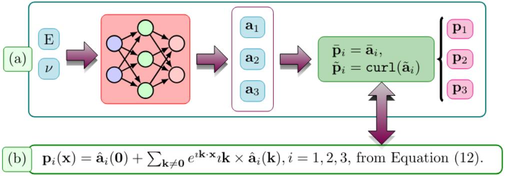
- The NN learns *both* the solution $u_\theta(x, t)$ and the parameter $\kappa(x)$.

Residual Modeling

- (Ooman et al. 2024) Alternating iterative classical solver and neural network prediction for speed-ups with a cost of numerical precision.



- (Sarkari Khorrami et al. 2024) Surrogate models for polycrystalline that stress fields satisfy mechanical equilibrium (divergence-free).



Outline

1 Physics-Informed Neural Networks

2 Hands-on Session

Introduction to JAX

What is JAX?

- A high-performance numerical computing library for Python.
- Combines NumPy-like API with automatic differentiation.
- Supports GPU/TPU acceleration seamlessly.

Key Features

- `grad`: Automatic differentiation.
- `jit`: Just-In-Time compilation for speed.
- `vmap`: Vectorization across batch dimensions.
- `pmap`: Parallelism across multiple devices.

Hands-on

In this hands-on session, you will:

- Learn how to use JAX.
- Implement a minimal example to solve the 1D heat equation and Allen-Cahn equation with PINNs.
- Apply adaptive sampling to achieve a more robust and efficient training process in the case of 2D Allen-Cahn equation.

References

- Raissi et al. *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 2019
- Cuomo et al. Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next, 2022
- Fidle CNRS : Online introduction courses to Deep Learning (French).

Take-Away

We have discovered:

- The general concept of **PINNs**
- Techniques to enhance training (e.g., adaptive sampling)
- The broader paradigm of **PIML**: Operator Learning, Inverse Problems, Residual Modeling

Now, it's time to **get your hands dirty!**

Thank you for your attention!

… any questions ?