ChEESE

Artificial Intelligence and Machine Learning for Geosciences

Physics-informed neural networks (PINNs)

H. Frezat, A. Fournier, L. Seydoux, G. Moguilny

Barcelona, November 7th, 2024

IPGP

Université Paris Cité

Project funded by EuroHPC under the grant agreement No 101093038.

Trained models and physics

$$f_\theta(\mathbf{x}) : \mathbf{x} \rightarrow \mathbf{y}, \qquad \operatorname{argmin}_\theta \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})$$

Can I say my model f_θ is physically **accurate**?

Trained models and physics

$$f_\theta(\mathbf{x}) : \mathbf{x} \to \mathbf{y}, \qquad \operatorname{argmin}_\theta \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})$$

Can I say my model f_θ is physically **accurate**?

- Respect some invariances.
- Solution of some equation.
- Stable when used in simulations (for IVPs).

Trained models and physics

$$f_\theta(\mathbf{x}) : \mathbf{x} \to \mathbf{y}, \qquad \operatorname{argmin}_\theta \ \mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})$$

Can I say my model f_θ is physically **accurate**?

- Respect some invariances.
- Solution of some equation.
- Stable when used in simulations (for IVPs).

How? Any guesses?

Inductive biases

	Technique	Nature	Instances
1	Data manipulation	Both	Augmentation / post processing
2	Loss function	Approx.	PINNs
3	Architecture	Exact	Basic blocks
4	Free parameters	Exact	

Different techniques to physically constrain a model.

Inductive biases

	Technique	Nature	Instances
1	Data manipulation	Both	Augmentation / post processing
2	Loss function	Approx.	PINNs
3	Architecture	Exact	Basic blocks
4	Free parameters	Exact	

Different techniques to physically constrain a model.

Many success stories:

- Atmospheric and ocean science (Bolton and Zanna, 2019; Beucler, Pritchard, et al., 2021).
- Weather and climate science (Kashinath et al., 2021; Beucler, Gentine, et al., 2024).
- Any fluid dynamics-based field.
- How about solid earth? See Moseley et al. (2020) for the wave equation in complex media

Solving a PDE: the modern way

Let's say we want to solve a non-linear (time-dependent) PDE.

$$
\begin{aligned}
u_t + \mathcal{N}(u) &= 0 \\
u(t = 0, x) &= g(x) \\
u(t, x = \partial\Omega) &= c(t)
\end{aligned}
$$

Solving a PDE: the modern way

Let's say we want to solve a non-linear (time-dependent) PDE.

$$
\begin{aligned}
u_t + \mathcal{N}(u) &= 0 \\
u(t = 0, x) &= g(x) \\
u(t, x = \partial\Omega) &= c(t)
\end{aligned}
$$

- Spatial discretization: finite-elements/differences/volumes: mesher, linear algebra solvers, etc.
- Temporal discretization.
- Efficient parallelization of the entire process.

Solving a PDE: the modern way

Let's say we want to solve a non-linear (time-dependent) PDE.

$$u_t + \mathcal{N}(u) = 0$$
$$u(t = 0, x) = g(x)$$
$$u(t, x = \partial\Omega) = c(t)$$

- Spatial discretization: finite-elements/differences/volumes: mesher, linear algebra solvers, etc.
- Temporal discretization.
- Efficient parallelization of the entire process.
- A couple hundred lines of Python?

PINNs: a powerful revolution

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

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

M. Raissi [a], P. Perdikaris [b,*], G.E. Karniadakis [a]

[a] Division of Applied Mathematics, Brown University, Providence, RI, 02912, USA
[b] Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, 19104, USA

(Huge success in the scientific community ($>$ 11k citations).

- Mesh-free (live in abstract latent space).
- Data-free (but supervised in the strict sense).
- Interpolation capabilities / super-resolution.
- Zero-effort parallelization.

The idea behind PINNs

$$
\begin{aligned}
f_\theta(t,x)_t + \mathcal{N}[f_\theta(t,x)] &= 0 \\
f_\theta(t=0,x) &= g(x) \\
f_\theta(t, x=\partial\Omega) &= c(t)
\end{aligned}
$$

A loss for each prior term

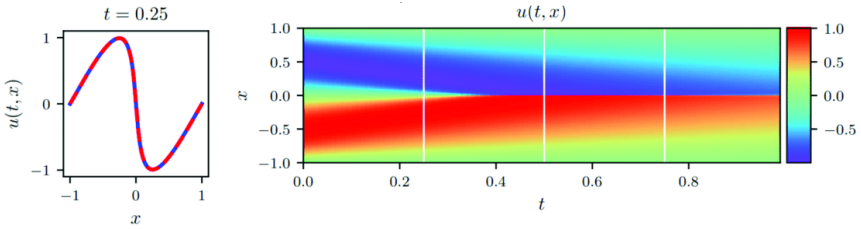
- Boundary conditions loss $\mathcal{L}_{\mathrm{BC}}(f_\theta(t, x=\partial\Omega), c(t))$.
- Initial conditions loss $\mathcal{L}_{\mathrm{IC}}(f_\theta(t=0,x), g(x))$.
- Equation residual loss $\mathcal{L}_{\mathrm{eq}}(f_\theta(t,x)_t + \mathcal{N}[f_\theta(t,x)])$.

The total loss is a sum of each:

$$\mathcal{L} = \mathcal{L}_{\mathrm{BC}} + \mathcal{L}_{\mathrm{IC}} + \mathcal{L}_{\mathrm{eq}}.$$

An illustration with Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0$$



Solution of Burgers' equation (Raissi et al., 2019).

- Let's create and train a neural solver for Burgers' equation in the notebook.

Bonus: hybrid modeling

Sometimes the problem is either **too slow** to solve, or contains **some unknown quantities**:

- Model complex phenomena not resolved by the equations (microphysics, clouds, etc.).
- Accelerate simulations by only resolving large scales (sub-grid modeling).
- Probably other instances.

- Keep the physics components with classical numerical methods.
- We can explore an example of sub-grid modeling in the bonus notebook.

References

- Beucler, Tom, Pierre Gentine, et al. (2024). “Climate-invariant machine learning”. In: *Science Advances* 10.6, eadj7250.
- Beucler, Tom, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine (2021). “Enforcing analytic constraints in neural networks emulating physical systems”. In: *Physical Review Letters* 126.9, p. 098302.
- Bolton, Thomas and Laure Zanna (2019). “Applications of deep learning to ocean data inference and subgrid parameterization”. In: *Journal of Advances in Modeling Earth Systems* 11.1, pp. 376–399.
- Kashinath, Karthik et al. (2021). “Physics-informed machine learning: case studies for weather and climate modelling”. In: *Philosophical Transactions of the Royal Society A* 379.2194, p. 20200093.
- Moseley, Ben, Andrew Markham, and Tarje Nissen-Meyer (2020). *Solving the wave equation with physics-informed deep learning*.
- Raissi, Maziar, Paris Perdikaris, and George E Karniadakis (2019). “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational physics* 378, pp. 686–707.