Material


PINA: Unified Framework for Scientific Machine Learning

Dario Coscia, PINA Team

email: dcoscia@sissa.it
email: pina.mathlab@gmail.com

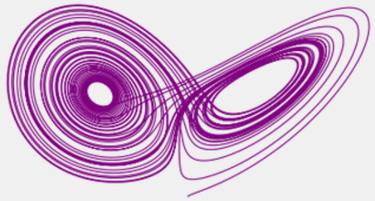
Amortization Opportunity for Scientific Computing

Modelling

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

slow

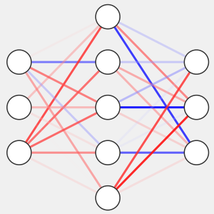
Single Simulation



Traditional Computing

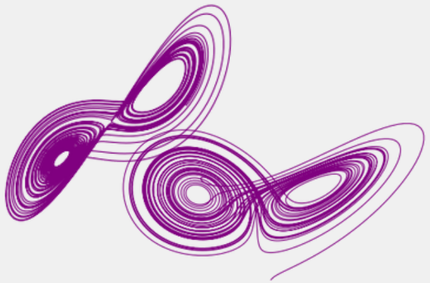
Data Storage

slow train



AI Surrogate

fast

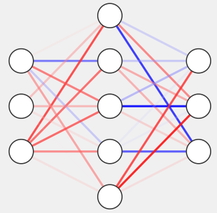


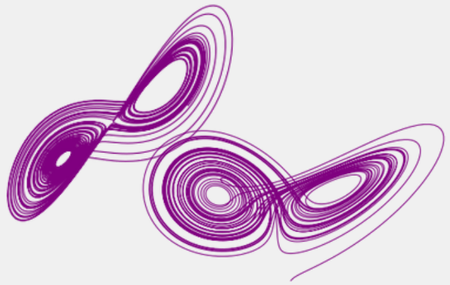
Multiple Simulations

AI Computing

Need for a Unique Engine: From Data to Simulations

pina





Different Data

- Meshes
- Fields
- Point Clouds
- ....

Different Models

- Neural Networks
- Reduced Order Models
- Hybrid Solvers
- ....

Different Objectives

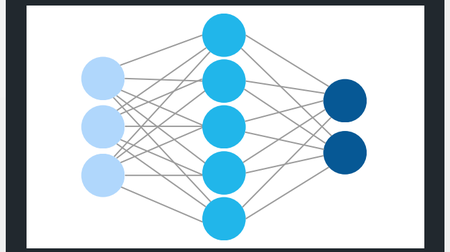
- Prediction
- System Identification
- Generation
- ....

Solving your Problem with PINA

$$\begin{cases} \dot{x} &= \sigma(y-x) \\ \dot{y} &= x(\rho-z)-y \\ \dot{z} &= xy-\beta z \end{cases}$$

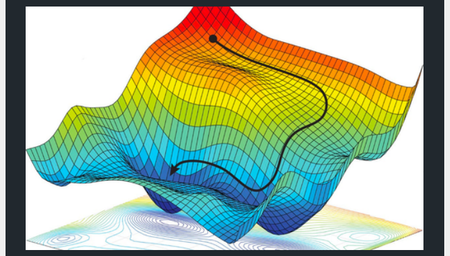
Problem & Data

Define the mathematical problem and identify the physical constraints. Discretize the domain or import external data.



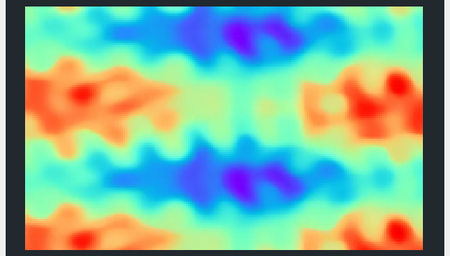
Model Design

Build your model as a PyTorch Module. Choose from available architectures or create a custom design tailored to your problem.



Solver Selection

Utilize available solvers or craft your own optimization strategy to effectively solve your problem.

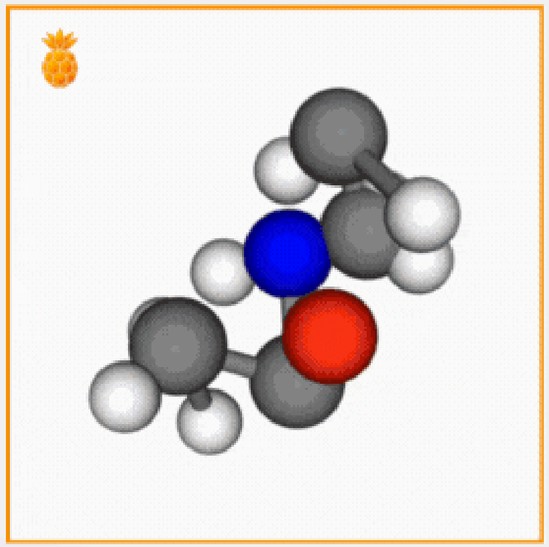


Training

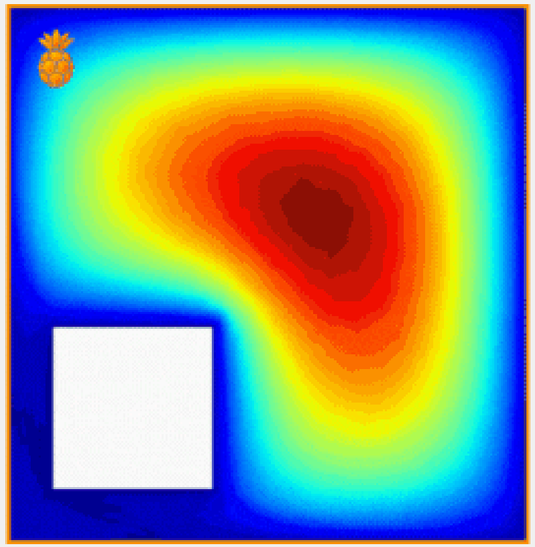
Optimize the model using the selected strategy, leveraging PyTorch Lightning features for enhanced training efficiency.

STEPS TO FOLLOW

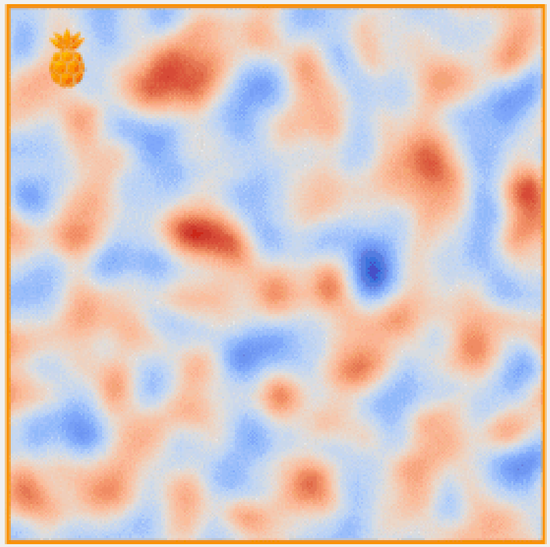
Endless Possible Applications



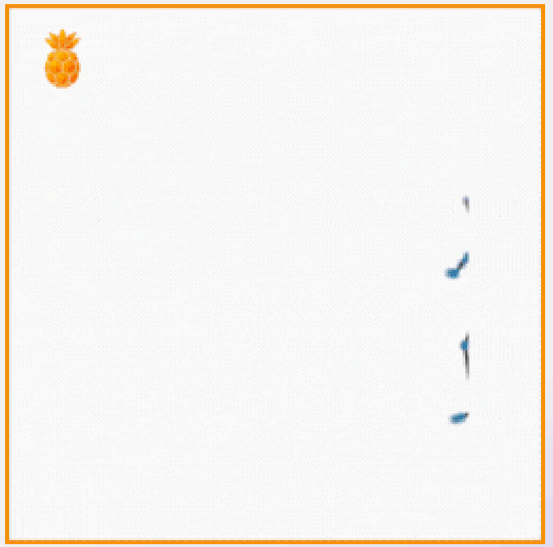
generative modelling



reduced order models



neural operators



robotics

Explore the Ecosystem

GITHUB.COM/MATHLAB/PINA

MATHLAB.GITHUB.IO/PINA

PYTORCH.ORG/BLOG/PINA

Reference - Coscia, Dario, et al. "Physics-informed neural networks for advanced modeling." Journal of Open Source Software 8.87 (2023): 5352.

Supervised learning

Given a dataset of input-output pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N},\ x_i \in \mathbb{X},\ y \in \mathbb{Y}$

We learn a model $\mathcal{M}_\theta$, **(relationship between input and output)**

By minimizing:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} l(y_i, \mathcal{M}_\theta(x_i)) \quad \text{with}, l = \mathbb{Y} \times \mathbb{Y} \rightarrow \mathbb{R}$$

* in unsupervised learning we don’t have golden labels

Machine Learned Interatomic Potentials

It is a supervised learning task where:

$$\mathcal{D} = \{(\underbrace{\boldsymbol{Z}_i, \boldsymbol{r}_i}_{\text{input}}, \underbrace{E_i, \boldsymbol{F}_i = -\nabla_{\boldsymbol{r}_i} E_i}_{\text{output}})\}_{i=1}^{N}$$

Energy is invariant under SO(3) transformations of atoms' positions

$$E_i(\boldsymbol{Z}_i, \boldsymbol{Q}\boldsymbol{r}_i + g) = E_i(\boldsymbol{Z}_i, \boldsymbol{r}_i), \quad \forall g \in \mathbb{R}^3,\ \boldsymbol{Q} \in \mathbb{R}^{3\times 3}\ \text{orthogonal}$$

Forces are equivariant under SO(3) transformations of atoms' positions

$$\boldsymbol{F}_i(\boldsymbol{Z}_i, \boldsymbol{Q}\boldsymbol{r}_i + g) = \boldsymbol{Q}\boldsymbol{F}_i(\boldsymbol{Z}_i, \boldsymbol{r}_i) + g, \quad \forall g \in \mathbb{R}^3,\ \boldsymbol{Q} \in \mathbb{R}^{3\times 3}\ \text{orthogonal}$$

Reduced Order Models

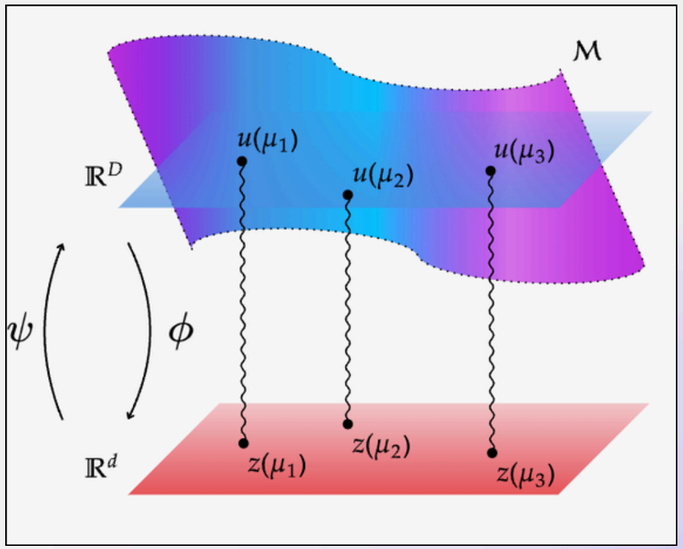
It is both a supervised and unsupervised learning task

1. **Unsupervised (Dimensionality Reduction):**

   learn ϕ, ψ (e.g. POD, autoencoders, …)

2. **Supervised (parameters to mode regression):**

   learn z (e.g. RBF, NN …)



$$\mu_i \to z(\mu_i) \to \psi(z(\mu_i)) \approx u(\mu_i)$$

Physics Informed Networks

It is an unsupervised learning task used for solving differential equations

Approximate PDE solution with a neural network $u(x,t) \approx u_\theta(x,t)$

- Example PDE: $\begin{cases} \Delta u(x) = \sin(\pi x) & x \in [-1,1] \\ u(x) = 0 & x \in \partial[-1,1] \end{cases}$
- Train: Minimize Data Loss Function $\mathcal{L}(\theta) = \sum_{x_i \in [-1,1]} |\Delta u_\theta(x_i) - \sin(\pi x_i)|^2 + |u(-1)|^2 + |u(1)|^2$

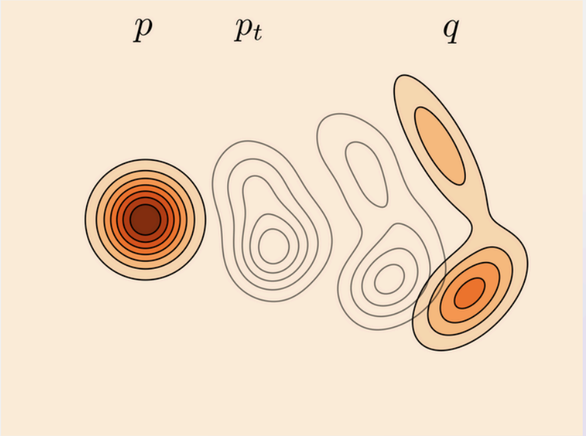
Generative Models

It is an unsupervised learning task where the goal is to learn:

$$q(x)$$

Flow as generative model:

$$x = z_0 + \int_0^1 u(z_t, t)dt, \quad x \sim q(x), z_0 \sim p(z_0)$$



To learn the flow we need to learn the field:

$$\mathcal{M}_\theta(z_t, t) \approx u(z_t, t)$$