AUTHORS

Dario Coscia, Nicola Demo, Gianluigi Rozza

dario.coscia@sissa.it
nicola.demo@fastcomputing.net
gianluigi.rozza@sissa.it

PINA: a PyTorch Framework for Solving Differential Equations by Deep Learning for Research and Production Environments

PINA is an open-source software powered by PyTorch and Lightning, designed for solving differential equations with neural networks. It supports Physics Informed Neural Networks and Neural Operators, offering flexibility for users to craft models tailored to their needs. PINA is modular and adaptable to different hardware setups, including GPUs and TPUs.

AFFILIATIONS

SISSA mathLab, Trieste, Italy

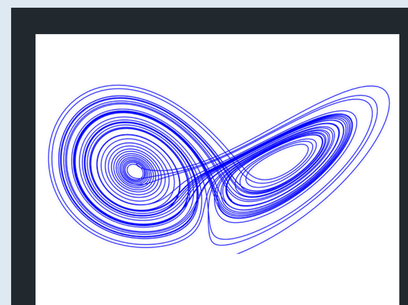
FAST Computing Srl, Trieste, Italy

SOLVE YOUR DIFFERENTIAL EQUATION STEP BY STEP

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

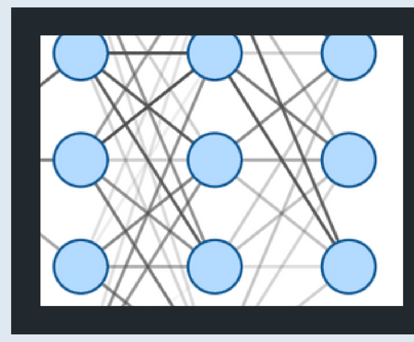
Problem definition

Specify the mathematical problem aimed to be solved and the specific physical condition to be satified



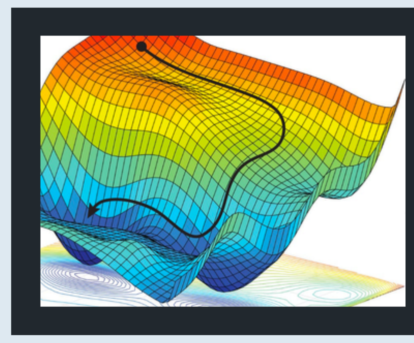
Sample the domain

Prepare the **input** of the model by discretising the **physical domain**, or import **data** from numerical solvers



Model & Solver selection

Build a **Model** as a PyTorch Module and choose the **Solver** strategy to optimize the model and solve the problem



Training

Optimize the **Model** with the specific **Solver** strategy with all the additional Pytorch Lightning feautures

CHECK THE GITHUB PAGE & LEAVE US A STAR ★

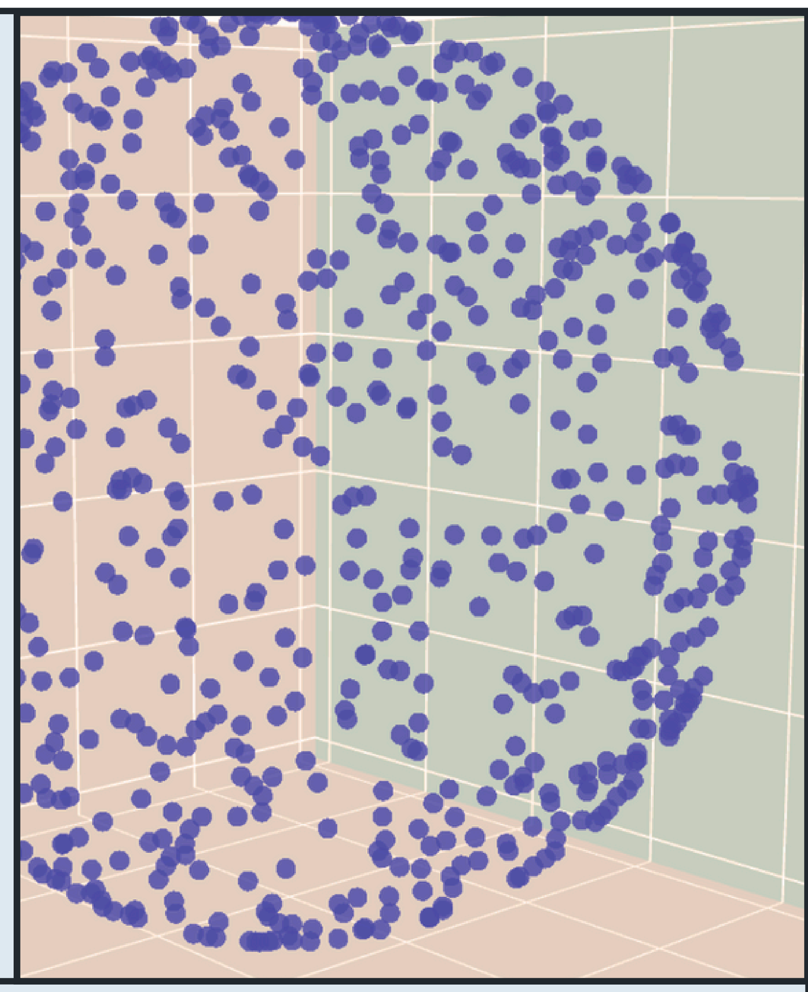
TRASLATE YOUR PROBLEM INTO PINA LANGUAGE

pina_poisson.py

```python
class Poisson(SpatialProblem):
    output_variables = ['u']
    spatial_domain = CartesianDomain({'x': [0, 1], 'y': [0, 1]})

    conditions = {
        'gamma1': Condition(
            location=CartesianDomain({'x': [0, 1], 'y': 1}),
            equation=FixedValue(0.0)),
        ...
        'gamma4': Condition(
            location=CartesianDomain({'x': 0, 'y': [0, 1]}),
            equation=FixedValue(0.0)),
        'D': Condition(
            location=CartesianDomain({'x': [0, 1], 'y': [0, 1]}),
            equation=my_laplace),
        'data': Condition(input_points=in_, output_points=out_)
    }
```

- **Spatial, Time-dependent, Inverse, and Parametric problems** are already available!
- We implemented the most common **differential operators** for easiness of usage.
  - grad
  - div
  - laplacian
- Triangular, cartesian, and elliptic shapes can be used to define the **problem domain**. Moreover, boolean operations have been implemented to make possible the definition of complex domains!



A HIERACHICAL PERSPECTIVE: SOLVER, MODEL, LAYER

PINA makes available the latest methodologies for equation learning. To maximize flexibility and modularity, the methodological implementation is divided into Solver, Model, and Layer

Solver

The "strategy" we want to apply to solve the problem --- e.g. PINN

- **Physics-informed paradigm, the solver minimizes the physical residual**
  - PINN, GPINN, CausalPINN, CompetitivePINN, SAPINN, ...
- **Supervised learning paradigm, the solver minimizes the difference between data and network output**
  - SupervisedSolver, GAROM, MessagePassingNeuralPDE, ReducedOrderModellingSolver ...
- **Abstract Interfaces to easily build new solvers**
  - SolverInterface, PINNInterface, ...

Model

The architecture to apply --- e.g. DeepONet

- **Standard and customizable deep learning architectures**
  - FeedForward, MultiFeedForward, ResidualFeedForward, ...
- **Specific Neural Operator architectures**
  - FourierNeuralOperator, LowRankNeuralOperator, AveragingNeuralOperator, MIONet, DeepONet, ...
- **.Easily build your PytorchModel or use our abstract Interfaces**
  - KernelNeuralOperator, Network, ...

Layer

The minimal brick that can be used to build a Model --- e.g. SpectralConv

- **Many PyTorch implementations of deep learning Layers**
  - ContinuousConvBlock, ResidualBlock, EnhancedLinear
  - SpectralConvBlock1D, SpectralConvBlock2D, SpectralConvBlock3D,
  - FourierBlock1D, FourierBlock2D, FourierBlock3D,
  - PODBlock, PeriodicBoundaryEmbedding, AVNOBlock, LowRankBlock
- **Adaptive Activation Functions**

HIGHLIGHTS

PINA is built upon PyTorch Lightning

- CPU GPU and TPU training support
- Loggers and Checkpoints for monitoring training
- Gradient Clipping, SWA, Gradient accumulation ....
- Callbacks for Solvers and Trainer

LabelTensor

- Extension of PyTorch Tensor class to handle labels
- Easily extract variables with strings and compute differential operator in symbolic notation
- Compatible with PyTorch main Tensor operations

$$\begin{array}{cccc} x & y & z & t \\ \end{array}$$
$$\begin{bmatrix} 2.3 & 4.5 & 1.2 & 0.2 \\ 3.7 & 6.1 & 9.8 & 1.3 \\ 5.2 & 7.4 & 3.9 & 4.5 \\ 8.6 & 2.4 & 6.7 & 2.9 \\ 1.5 & 9.3 & 4.8 & 6.1 \end{bmatrix}$$

High-level design

- Modular components and object-oriented structure
- Abstract interfaces to easily add new components
- Natively support PyTorch Model compilation

Documentation online

- Fully documented package
- Installation and contribution guidelines
- Tutorials for getting start with the software

Tested for several operating systems

- Fully compatible and tested for Python ≥ 3.7
- Running on Windows macOS and Ubuntu

Easy installation

- pip install pina-mathlab
- New release the 1st of any month