# Accelerating Numerical Simulations by Model Reduction with Scientific and Physics-Informed Machine Learning

PINN-PAD conference - University of Padova, Italy
February 22-23, 2024

**Gianluigi Rozza**

***mathLab*, Mathematics Area, SISSA International School for Advanced Studies, Trieste, Italy**
**email: grozza@sissa.it**

# Introduction and Leading Motivations

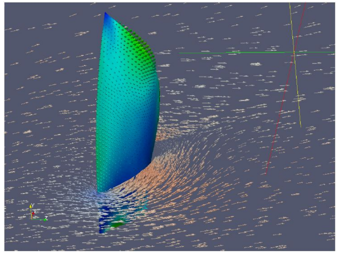
#HPC #PDEs #DL #offline-online #software

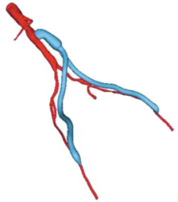
# Leading Motivation: Computational Sciences challenges

- → **Reduced Order Modelling** is a quickly emerging field in applied mathematics and computational science and engineering for speeding up **Numerical Simulations**

- → Growing demand of
  - **efficient computational tools**
  - **many query** and **real time** computations
  - **parametric formulations**
  - **uncertainty quantification**

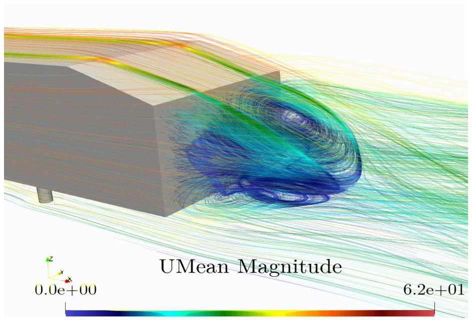
- → The need of a computational collaboration rather than a competition between **High Performance Computing** (HPC) and **Reduced Order Methods** (ROM), as well as Full/High Order and Reduced Order Methods.



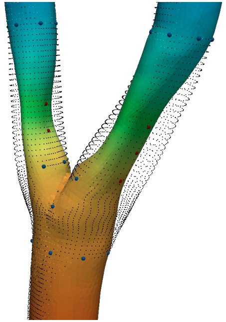


# Physical Parametric Differential Problems Overview

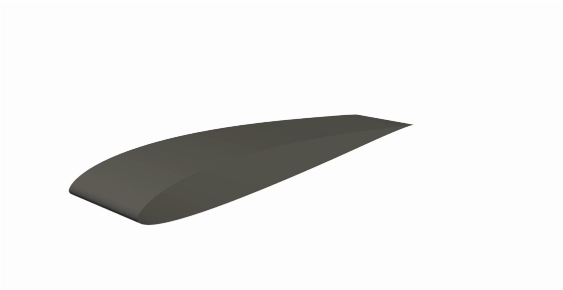
**Parametric Differential Problem** are ubiquitous in many field of Natural Science from **naval** and **nautical** engineering, to **aeronautical** engineering and **industrial** engineering.




**automotive**



**biomedics**



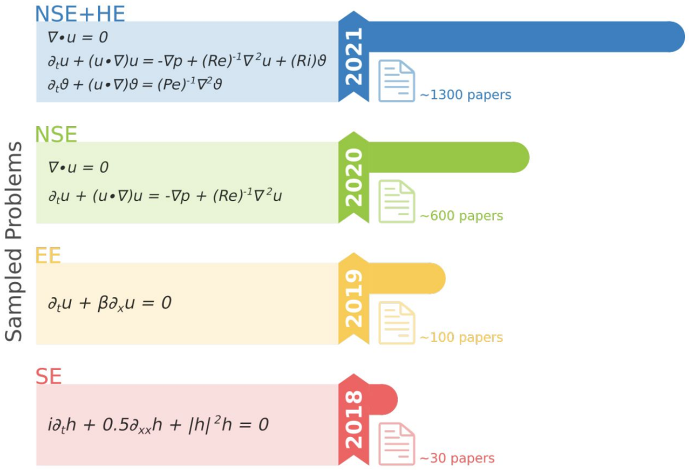
**aeronautics**

**References:**

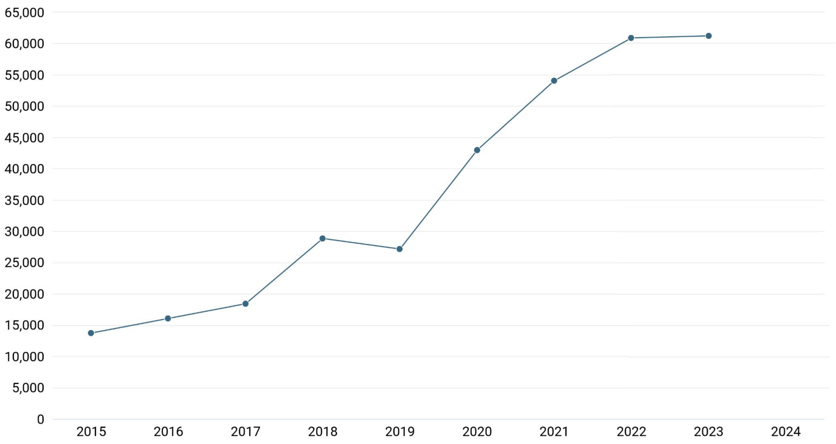
1. *Rozza, Gianluigi, Giovanni Stabile, and Francesco Ballarin (2022) eds. Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics. Society for Industrial and Applied Mathematics.*

# The Deep Learning New Era

**Physics Informed Neural Networks** (PINNs), **Deep Learning ROM** (DL-ROMs) and **Neural Solvers** are revolutionizing the field of Computational Science bringing high generalization capability



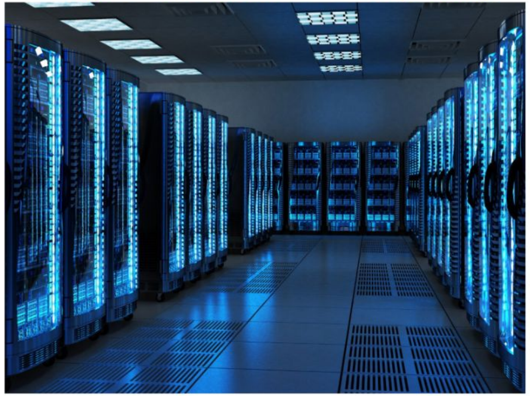

**Research articles on CSE using PINNs**




**Research articles per year on learning PDEs**

# Towards real-time computation (hardware)

**OFFLINE (full order)**
**High Performance Computing**



* **Very expensive** and time demanding;
* basis calculation done once after suitable parameters sampling (ex: **Proper Orthogonal Decomposition, RB, PGD**, ...);
* *HPC facilities.*

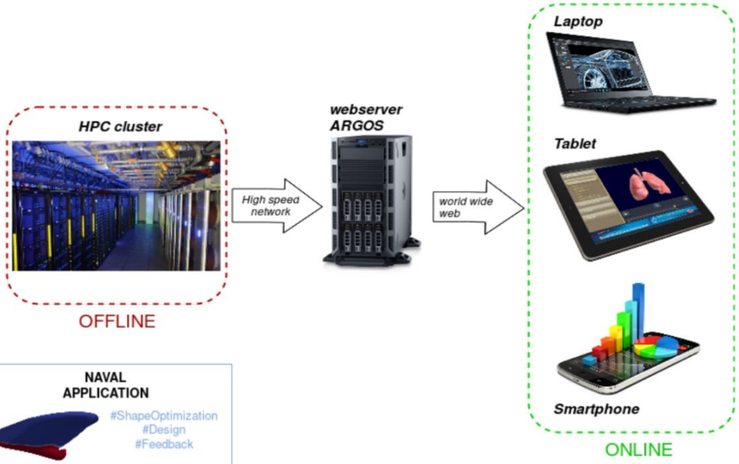
**ONLINE (reduced order)**
**Advanced ROM techniques**

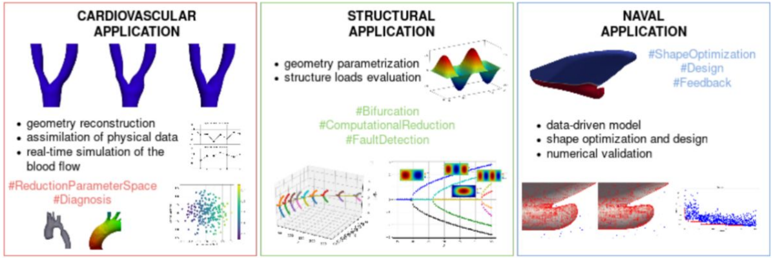
* **Extremely fast**;
* **real-time** input-output evaluation;
* computational **webserver** via browser;
* *in situ, tablets or smartphones.*

# Computational Webserver/Computational Apps

**Model order reduction for computational web server:** to real world applications **argos.sissa.it**

- HPC
- data science
- Digital twin
- SMACT Industry 4.0
- 3D Printing



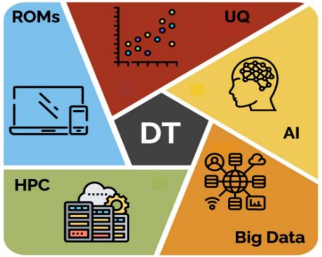
# Digital Twin (DT): integration of emerging fields

A large amount of data (**Big Data**) can be collected, **Artificial Intelligence** (AI) can help to store and **organize** them (data-driven approaches).

By using **black box models**, AI techniques are able to find **fitting functions**. They do not require knowledge about the physics of the problem, even if we do prefer integrated "**Big Models**" Physics informed approaches.

The development of **High Performance Computing** (HPC) and its integration with reduced order models allowed to reach better performances.

* **Uncertainty quantification** (UQ),
* **Data analytics**,
* **Artificial intelligence** (AI),
* **Digital Twins** of products and processes.




Thanks to ROMs we have a more sustainable framework, energy savings, reduced computational times and resources.

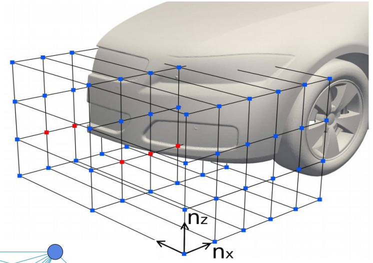
# SISSA mathLab: our current efforts and perspectives

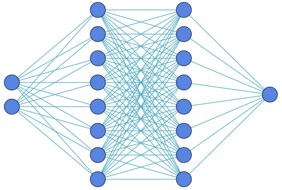
A team developing **Advanced Reduced Order Method**s for parametric PDEs!

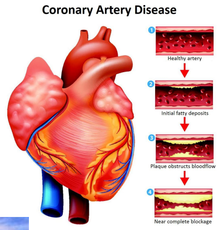
# SISSA mathLab: our current efforts and perspectives

**Goals of our research group**:

- ➔ Face and overcome **several limitations** of the state of the art for parametric ROM by means of **Deep Learning**
- ➔ Improve capabilities of reduced order methodologies for **more demanding applications** in industrial, medical and applied sciences settings
- ➔ Carry out important **methodological developments** in Numerical Analysis, with special emphasis on **mathematical modelling** and a more extensive exploitation of Computational Science and Engineering
- ➔ Focus on Computational Fluid Dynamics as a central topic to enhance broader applications in multiphysics and coupled settings (e.g. aeronautical, mechanical, naval, cardiovascular surgery, ...)






# SISSA mathLab: our current efforts and perspectives

➔ Development of new open-source tools based on reduced order methods:

- **ITHACA**, In real Time Highly Advanced Computational Applications, as an add-on to integrate already well established CSE/CFD open-source software
- **RBniCS** as educational initiative (FEM) for newcomer ROM users (training).
- **Argos A**dvanced **R**educed order modellin**G O**nline computational web server for parametric **S**ystems
- **PINA** a deep learning library to solve differential equations
- **EzyRB** data-driven model order reduction for parametrized problems
- **PyDMD** a Python package designed for Dynamic Mode Decomposition ( in collaboration with University of Texas, CERN, and University of Washington)

PiNA ITHACA ITHACA - FV PyDMD ITHACA-DG RBniCS multiphenics ARGOS ATLAS EZyRB

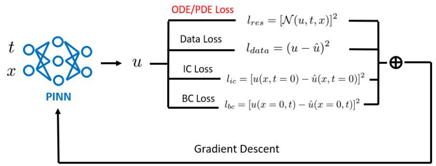
# A short history of Scientific Machine Learning

#roms #history #pinns
#offline-online #neuraloperators

# Scientific Machine Learning for PDEs


**Linear Algebra based Reduced Order Models**

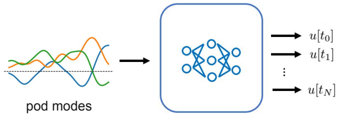
< 2017




**Physics Informed Machine Learning**

2017

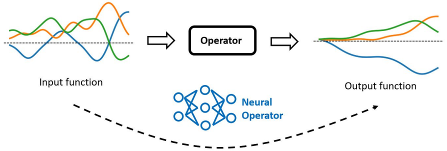
**Artificial Neural Networks as Reduced Order Models**




2019

2021

**Neural Operator Learning**




**Symmetries, High Dimensional Systems, Stochastic Equations, …**

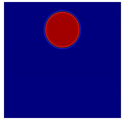
....

# How to solve PDEs by Scientific Machine Learning

The **ML pipeline can be divided into four stages**

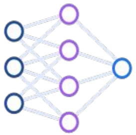
1. Select a **problem to solve** e.g. fluid dynamics, stochastic pdes, ...
2. Generate the **data**, e.g. high fidelity simulations, scattered data from the domain, ...
3. Build a **ML model**, e.g. NNs, POD + Interpolation, Neural Operators, ...
4. **Optimize** the model, e.g. by Supervised, Physics-Informed losses and gradient descent

$$\partial_t \phi + \mathbf{u} \cdot \nabla \phi = 0$$

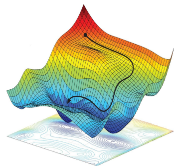


problem to solve

data generation



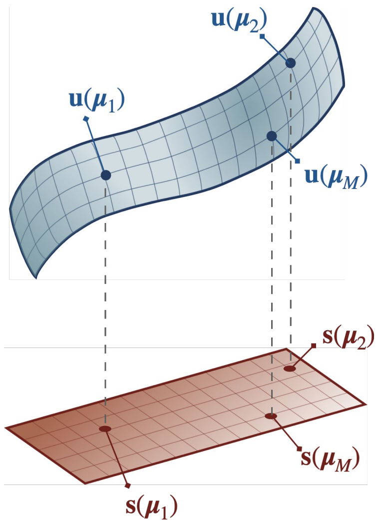
build ML model

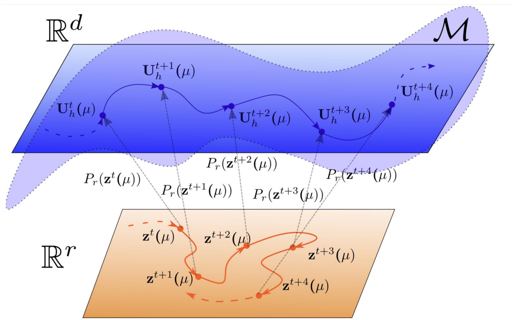


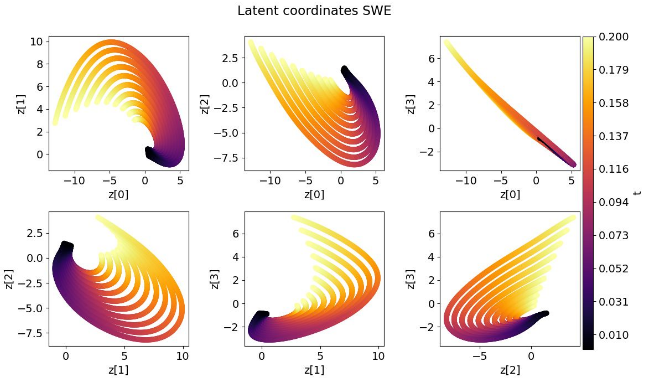
optimization

# The Data-Driven Approach to Reduced Models

- Reducing Parameter Space
- Applicable for Sensor and Incomplete Data
- Fast Online Phase



# Reduced Order Model - Accelerating Numerics

* $()_h$: **Full Order Methods** (FEM, FV, FD, SEM) are **high fidelity solutions** - to be **accelerated**;
* $()^{ROM}$: **Reduced Order Methods** (ROM) - the **accelerator**.

* Input parameters:

$$\boldsymbol{\mu} \text{ (geometry, physical properties, etc.)}$$

* Parametrized PDE:

$$\mathcal{A}(u(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0$$

* Output:

$$u(\boldsymbol{\mu}) \approx \underbrace{\mathbf{u}_h(\boldsymbol{\mu})}_{\text{full order}} \approx \underbrace{\mathbf{u}^{ROM}(\boldsymbol{\mu})}_{\text{reduced order}}$$

* Input-Output evaluation: (black-box)

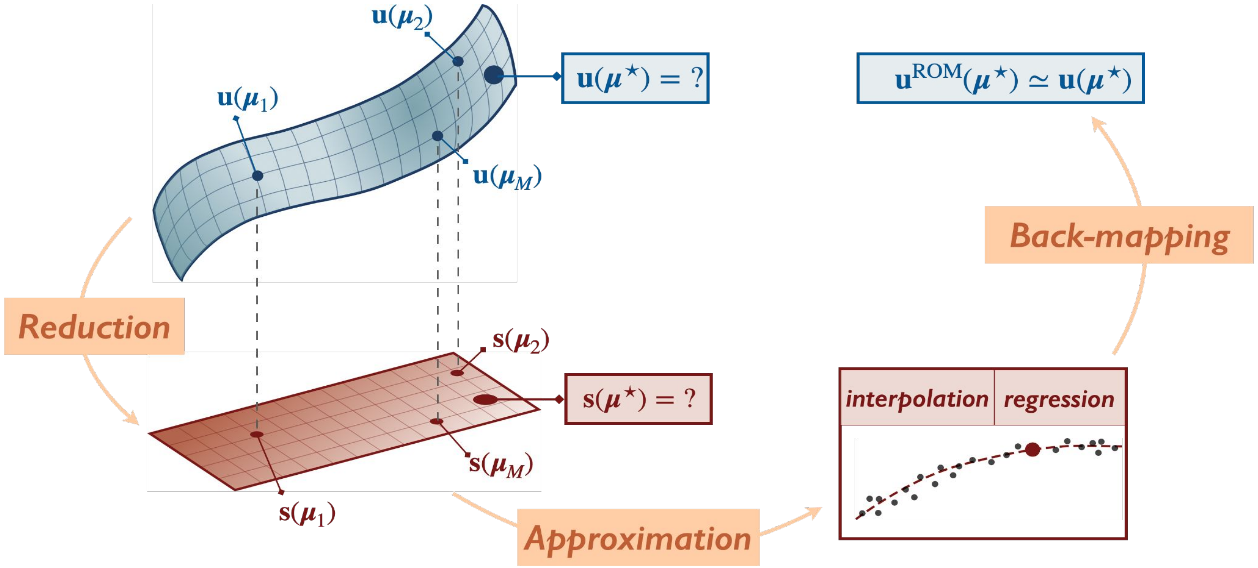
$$\boldsymbol{\mu} \rightarrow \mathbf{u}_h(\boldsymbol{\mu}) \rightarrow \mathbf{u}^{ROM}(\boldsymbol{\mu})$$

**References:**

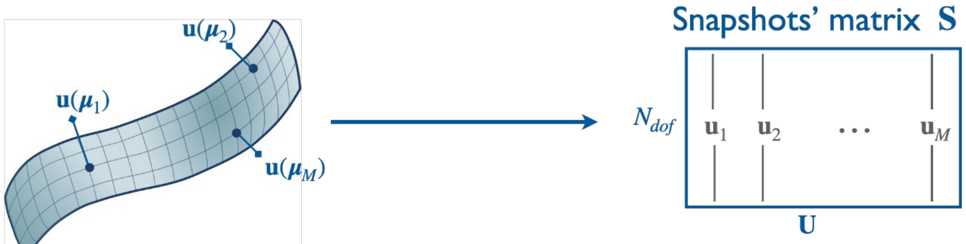

1. Hesthaven, J. S., Rozza, G., & Stamm, B. (2016). Certified reduced basis methods for parametrized partial differential equations (Vol. 590, pp. 1-131). Berlin: Springer.
2. Rozza, G., Stabile, G., & Ballarin, F. (Eds.). (2022). Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics. Society for Industrial and Applied Mathematics.

# Data-Driven approach to ROM

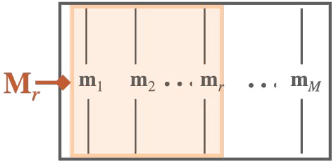
**ROM** approximate the high dimensional solution manifold by dimensionality reduction and perform interpolation to predict for unseen parameters

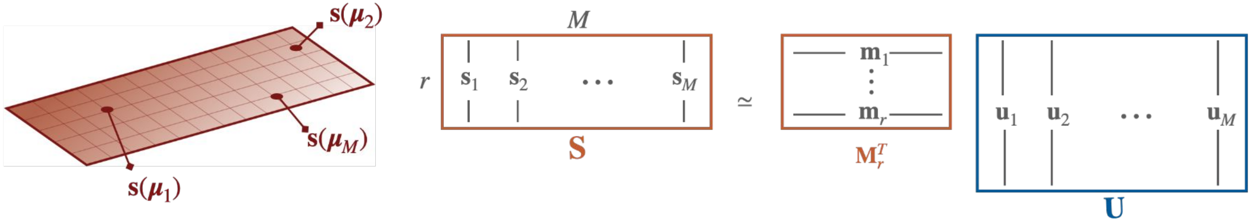


# Manifold Reduction - extracting latent features




- *Singular Value Decomposition:* $\mathbf{U} = \mathbf{M}\Sigma\mathbf{V}^T$
- The first $r$ columns of $\mathbf{M}$ span the ***reduced space*** $r \ll N_{dof}$
- Evaluation of the ***modal coefficients*** $\mathbf{S}$



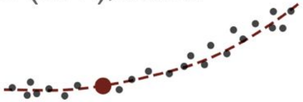
# Interpolation - approximate the low dimensional manifold

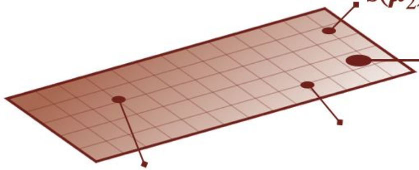
## *Approximation*

Evaluate the ***modal coefficients*** at unknown parameter:

- ***Interpolation*** techniques: Radial Basis Function (***RBF***), …
- ***Regression*** techniques:

  Gaussian Process Regression (***GPR***), neural networks, …






## *Back-mapping*

LIBRARY

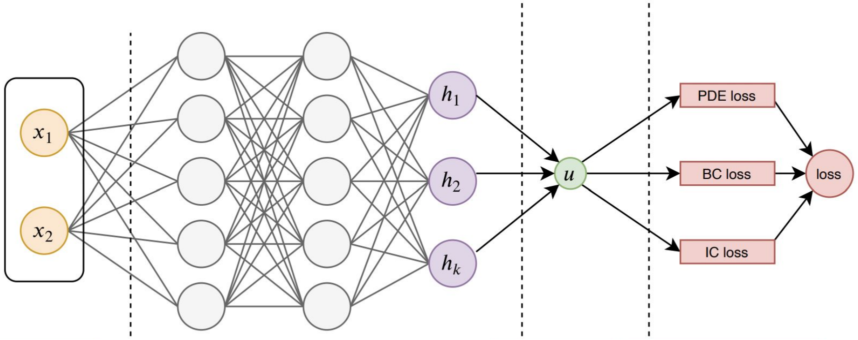
EZyRB

***ROM prediction***

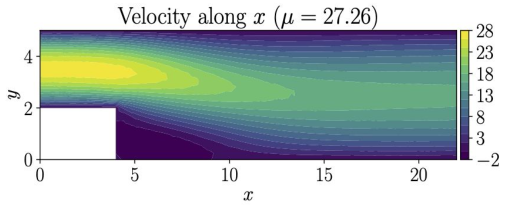
$$\mathbf{u}^{\mathrm{ROM}}(\boldsymbol{\mu}^\star) = \mathbf{M}_r\, \mathbf{s}(\boldsymbol{\mu}^\star) \longrightarrow \mathbf{u}^{\mathrm{ROM}}(\boldsymbol{\mu}^\star) \simeq \mathbf{u}(\boldsymbol{\mu}^\star)$$

# Physics Informed Neural Network

- ➔ No need of Data, only Equations
- ➔ Scatter Domain Data -> Avoiding Meshing
- ➔ General (inverse forward problems) and Fast




$$
\begin{aligned}
\nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p &= 0 \\
\nabla \cdot \mathbf{u} &= 0 \\
\mathbf{u} &= \mu \left\{ \tfrac{1}{2.25}(x_1 - 2)(5 - x_1), 0 \right\} \\
\mathbf{u} &= 0 \\
\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} &= 0
\end{aligned}
$$

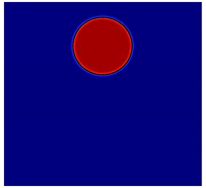


# The Physics Informed Neural Network (PINN)

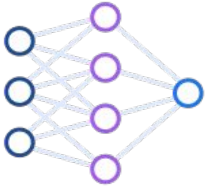
Physics Informed Neural Network is an optimization technique to compute solution of differential equation using Neural Networks

$$\begin{cases} \mathcal{A}(u(x,\mu)) = 0 & x \in \Omega \\ \mathcal{B}(u(x,\mu)) = 0 & x \in \partial\Omega \end{cases}$$

$$u_\theta(x\ ,\ \mu)$$



**problem to solve**



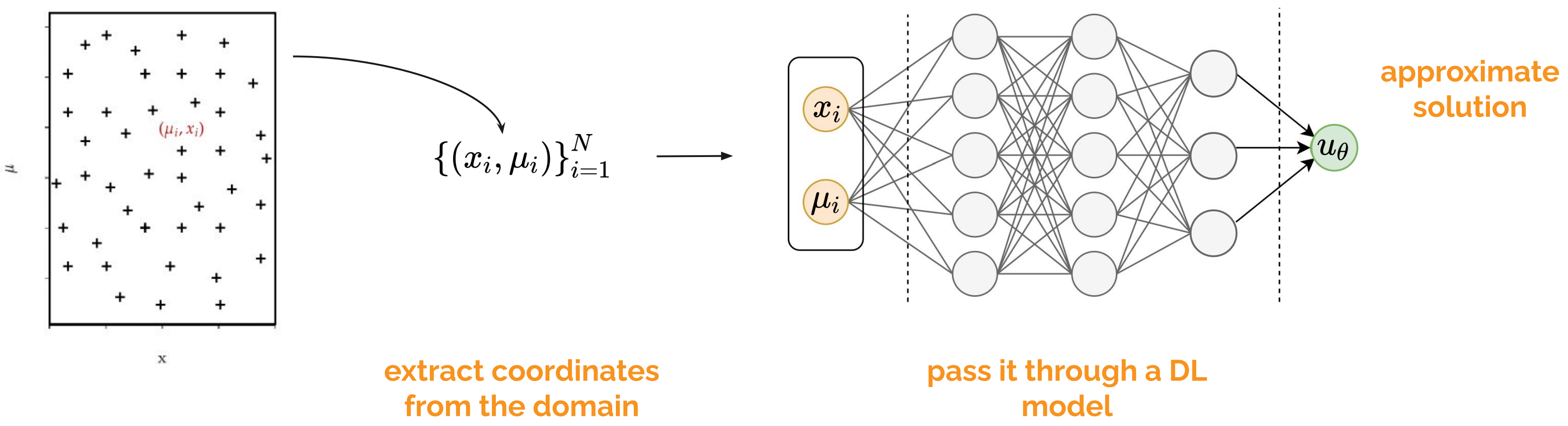
**model**


**References:**

1. 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.
2. Cuomo, Salvatore, et al. "Scientific machine learning through physics–informed neural networks: Where we are and what's next." Journal of Scientific Computing 92.3 (2022): 88.

# The Physics Informed Neural Network (PINN)

A parametrized ML model $u_\theta(x\,,\,\mu)$ is used to approximate the true solution $u(x\,,\,\mu)$ on some samples of scattered data inside the domain



# The Physics Informed Neural Network (PINN)

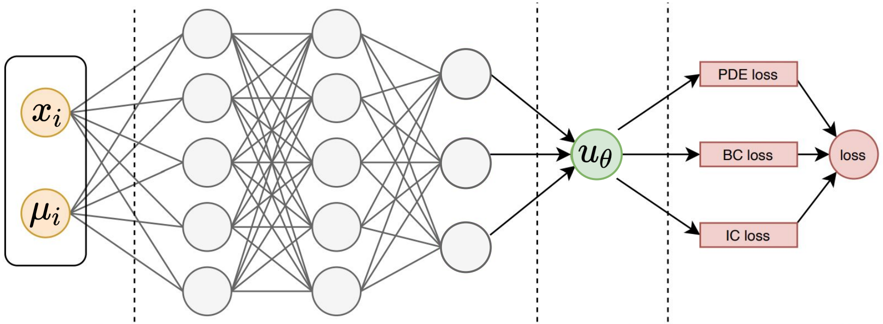
The underlying differential equation in PINNs is used to derive the loss function, where the differential operators are computed by automatic differentiation

$$\begin{cases} \mathcal{A}(u(x,\mu)) = 0 & x \in \Omega \\ \mathcal{B}(u(x,\mu)) = 0 & x \in \partial\Omega \end{cases}$$

**differential problem**

$$L = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{A}(u_\theta(x_i,\mu_i))\|^2 + \|\mathcal{B}(u_\theta(x_i,\mu_i))\|^2$$

**residual loss**




**model**

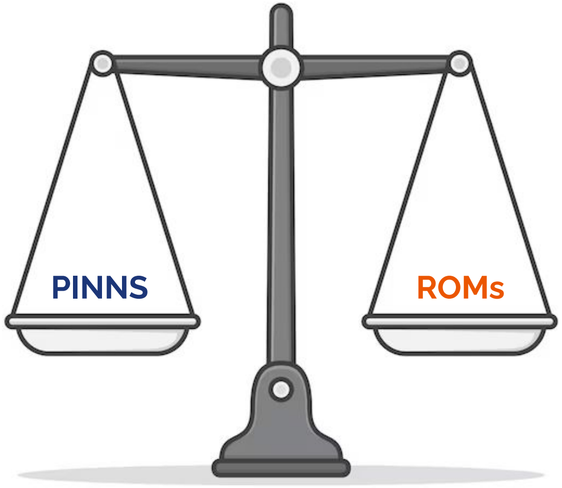
# Inductive Bias vs Real Data

➔ Data and Physical knowledge must be **balanced** to build a **truthful** and **reliable** ML model

## Inductive Bias

Physical Equations

Constraints and Symmetries




## Data

Full Order Models simulations

Sensor Data

# Physics Informed Neural Networks - latest advancements and software

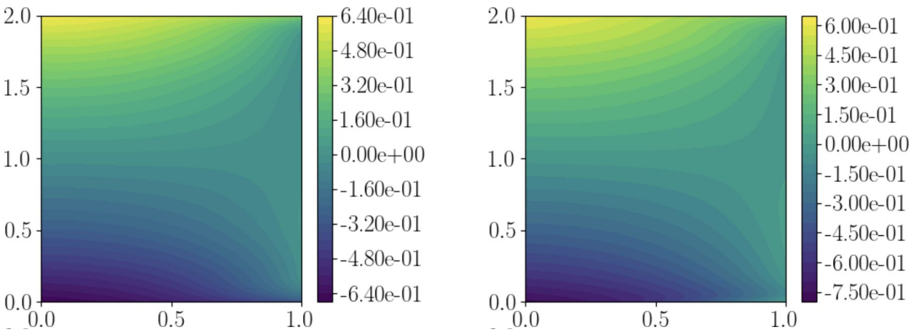
#data-free #software #pinns
#pde-modelling #mesh-agnostic

# Applications of Physics Informed Neural Networks

- Inverse Modelling and Optimal Control in PINNs
- Inverse Problem for Heating Steel Bar

**References::**
*Demo, Nicola, Maria Strazzullo, and Gianluigi Rozza (2023). "An extended physics informed neural network for preliminary analysis of parametric optimal control problems." Computers & Mathematics with Applications 143 ..*

Figure : Parametric Stokes Optimal Control Problem. The field $p$, $r$, $u$, $v$ and $z$ are shown for $\boldsymbol{\mu} = 1$, from the top to the bottom. *Left column*. Standard FNN approximation. *Right column*. PI-Arch approximation.




$$\min_{(v(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))} \frac{1}{2}\|v(\mathbf{x},\boldsymbol{\mu}) - x_2\|^2_{L^2(\Omega)} + \frac{\alpha}{2}\|u(\mathbf{x},\boldsymbol{\mu})\|^2_{L^2(\Omega)},$$

constrained to

$$\begin{cases} -0.1\Delta v(\mathbf{x},\boldsymbol{\mu}) + \nabla p(\mathbf{x},\boldsymbol{\mu}) = f(\mathbf{x},\mu_1) + u(\mathbf{x},\boldsymbol{\mu}) & \text{in } \Omega, \\ \nabla \cdot v(\mathbf{x},\boldsymbol{\mu}) = 0 & \text{in } \Omega, \\ v(\mathbf{x},\boldsymbol{\mu})_1 = x_2 \text{ and } v(\mathbf{x},\boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{D}}, \\ -p(\mathbf{x},\boldsymbol{\mu})\mathbf{n}_1 + 0.1\dfrac{\partial v(\mathbf{x},\boldsymbol{\mu})_1}{\partial \mathbf{n}_1} \text{ and } v(\mathbf{x},\boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{N}}. \end{cases}$$

# Solve Inverse Problems with PINNs

- **General formulation**:
  infer unknown parameters $(\mu_i)_{i=1}^{n}$ such that:

  ➡ The model **equations** are fullfilled:

  $$\frac{\partial u}{\partial t} + f\left(\frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \mu_1, \mu_2, \ldots, \mu_n\right) = 0$$

  ➡ Pre-computed **data** are fitted:

  $$u(t,x) = u_{data}(t,x)$$

- **PINN formulation**:
  find $\boldsymbol{u}$ and $(\mu_i)_{i=1}^{n}$ minimizing the loss:

  $$\mathscr{L}_{eq} = \frac{\partial u}{\partial t} + f\left(\frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \mu_1, \mu_2, \ldots, \mu_n\right)$$

  +

  $$\mathscr{L}_{data} = u - u_{data}$$

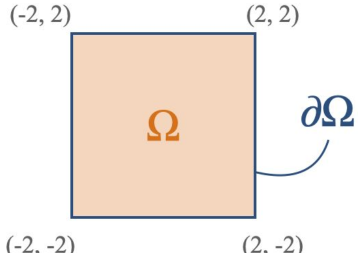
- **Examples of applications**: find properties of materials to satisfy specific operating conditions.

# A first preliminary inverse problem with PINN

## Poisson parametric inverse problem:

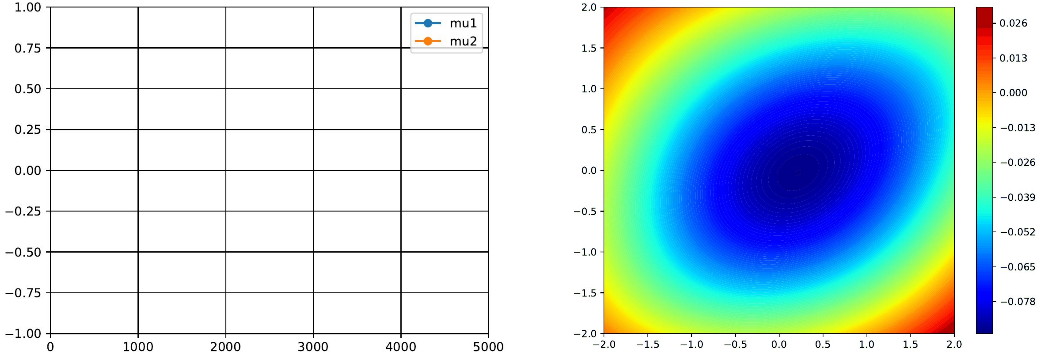
$$\begin{cases}\Delta u = e^{-2(x-\mu_1)^2-2(y-\mu_2)^2} \text{ in } \Omega \\ u = 0 \text{ on } \partial\Omega\end{cases}$$

Unknown parameters
in range (-1, 1)




## Result:

quick convergence to the expected result




*Solutions and parameters through training epochs*

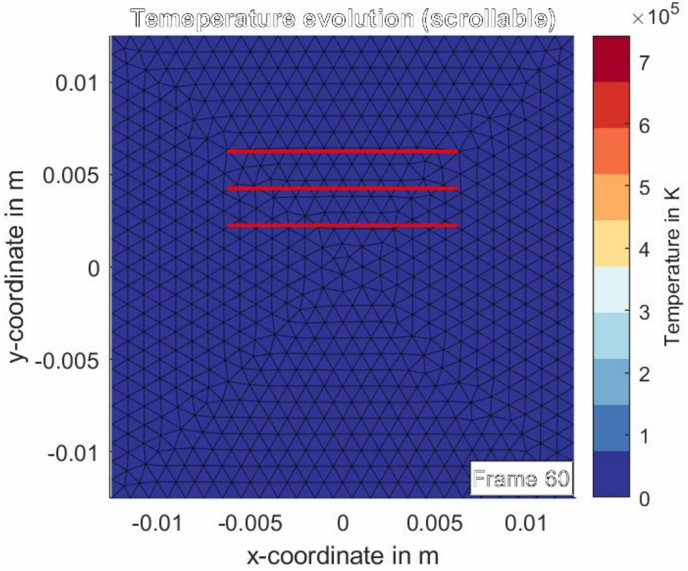
# The heat problem test case

**Goal:** understanding the thermal behaviour of **Additive Manufacturing (AM)** components to improve the process design and enhance quality control

**Our test case:** a squared plate heated by a moving laser source having a constant velocity.

**Unknown parameters**: material properties of the plate (thermal conductivity **k** and diffusivity constant **m**)

*FEM simulation: evolution of the temperature on the plate surface as the laser is moving*



# The heat problem test case

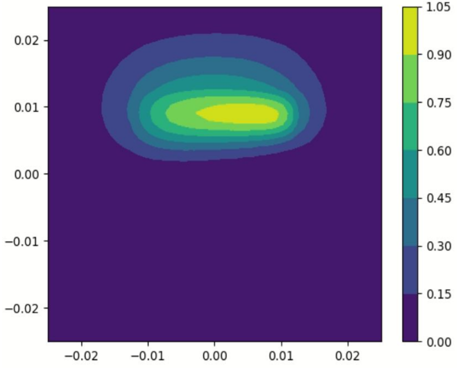
**Test case:** a squared plate heated by a moving laser source.

- **Data:** $\theta = T - T_\infty$
- **Equation:**

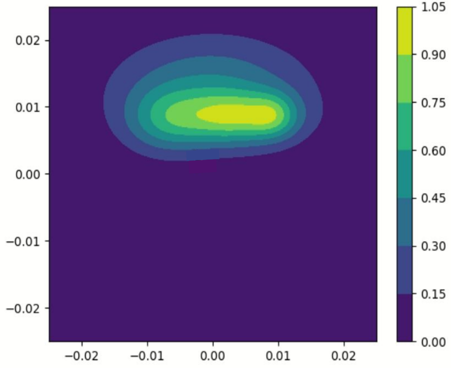
$$m\frac{\partial \theta}{\partial t} - k\Delta\theta = \text{laser source}(x, y, t) - h\theta + \dots$$

Unknown material properties

***Preliminary results:***



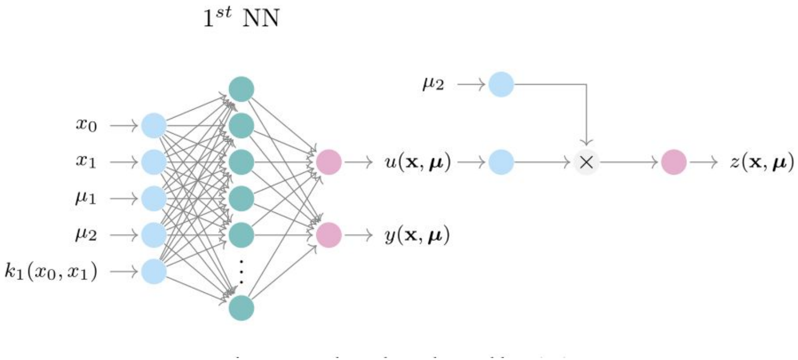

*Truth at t=10.7 s*

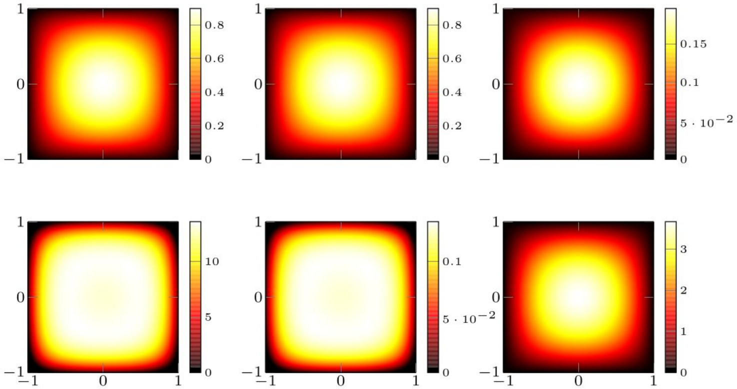



*PINN solution at t=10.7 s*

# Optimal Control Applications

- Parametric optimal control problem can easily be solved leveraging PINNs
- Physics-informed Architecture: fit the architecture model to your problem (hard constrained)

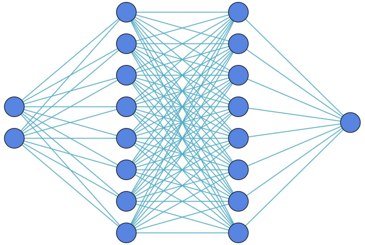



**References::**
*Demo, Nicola, Maria Strazzullo, and Gianluigi Rozza (2023). "An extended physics informed neural network for preliminary analysis of parametric optimal control problems." Computers & Mathematics with Applications 143 ..*

# Physics Informed Neural Network and ROMs Software

- User friendly
- Multiple HPC Devices (GPU, TPU, …)
- ROMs, PINNs, NOs, and all the state-of-the-art methods implemented

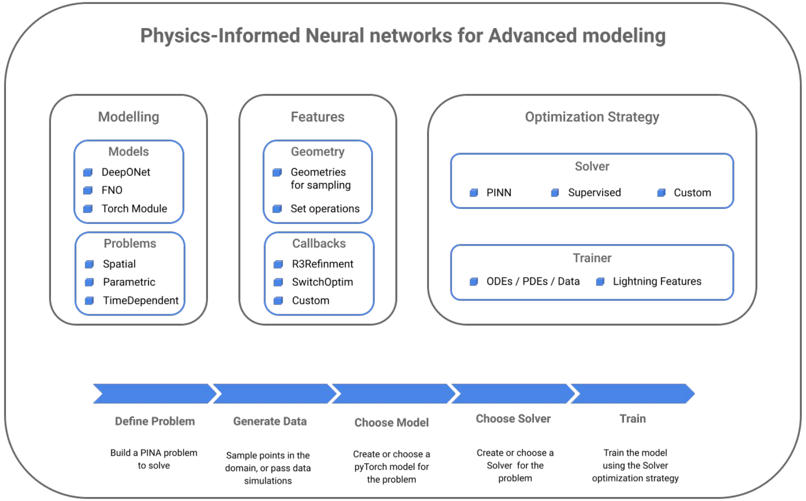


PiNA

PyTorch

PyTorch Lightning
Created by Lightning AI

# PINA - Learning Solution to PDE with simple code




→ **P**hysics **I**nformed **N**eural network for **A**dvanced modelling is Python software for solving PDEs using **State-Of-The-Art Models**

- Highly sectorized and customizable
- Multi-device / hardware training
- PyTorch Lightning backhand
- Easy tutorials to start!

PINA

PyTorch Lightning
Created by Lightning AI

PyTorch

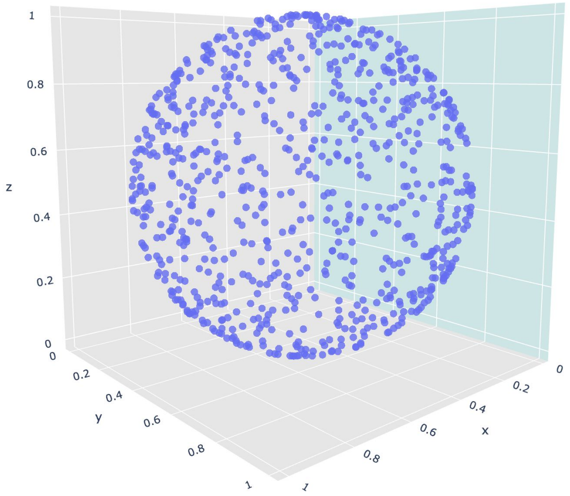
# Solving PDEs with PINA - Problem Definition

```python
class Poisson(SpatialProblem):

    # define laplace equation
    def laplace_equation(input_, output_):
        force_term = (torch.sin(input_.extract(['x'])*torch.pi) *
                      torch.sin(input_.extract(['y'])*torch.pi))
        return laplacian(output_.extract(['u']), input_) - force_term

    # output variables and spatial domain
    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)),
          'gamma2': Condition(location=CartesianDomain({'x': [0, 1], 'y': 0}), equation=FixedValue(0.0)),
          'gamma3': Condition(location=CartesianDomain({'x':  1, 'y': [0, 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=Equation(laplace_equation))}
```

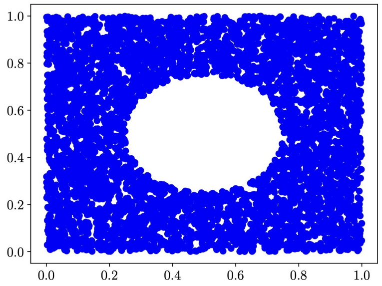
$$\begin{cases} \Delta u(x,y) = \sin(\pi x)\sin(\pi y) & (x,y) \in [0,1]^2 \\ u(x,y) = 0 & (x,y) \in \partial[0,1]^2 \end{cases}$$

Define Problem | Generate Data | Choose Model | Choose Solver | Train

# Solving PDEs with PINA - Data Generation



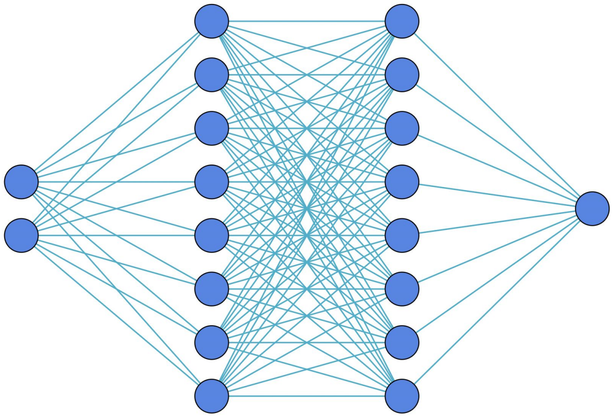

PINNs equations are evaluated over the neural network on some **scattered** sample points of the domain




Define Problem
**Generate Data**
Choose Model
Choose Solver
Train

# Solving PDEs with PINA - Model Selection

```python
# make model
model = FeedForward(input_dimensions=2,output_dimensions=1, layers=[8, 8],func=Softplus)
```



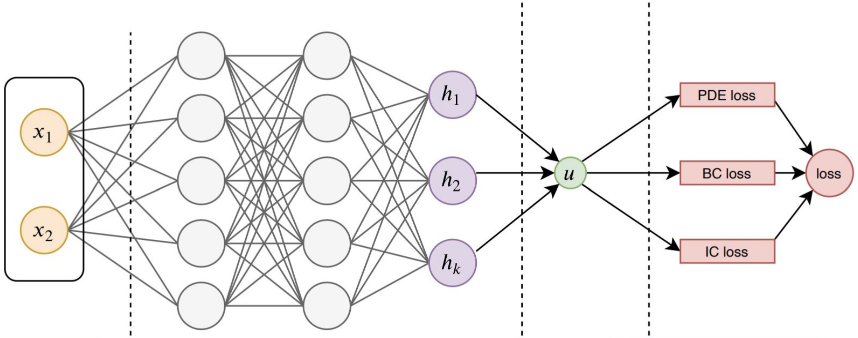
PINA implements most SOTA models:

- FeedForward
- Residual Networks
- Fourier Neural Operator (FNO)
- Deep Operator Network (DeepONet)
- …..

Define Problem → Generate Data → **Choose Model** → Choose Solver → Train

# Solving PDEs with PINA - Solver Selection

```
# make solver
pinn = PINN(problem, model, loss, optimizer,scheduler, extra_features)
```

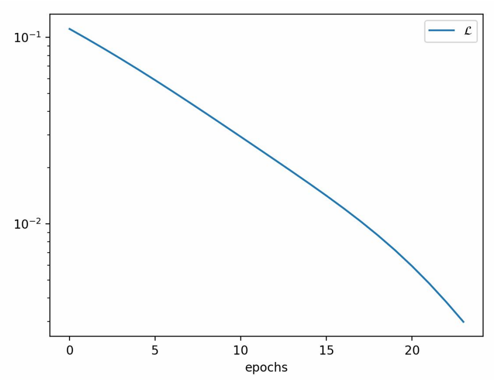



PINA implements most SOTA solvers:

- PINN (and extensions, gPINN, causalPINN, ...,)
- GAN solvers (GAN, GAROM)
- Graph Neural Solvers (MP-PDE, GNO, ....)
- .....

# Solving PDEs with PINA - Training

```
# trainer
trainer = Trainer(pinn, max_epochs, accelerator, batch_size, gradient_clip_val,gradient_clip_algorithm)
# train
trainer.train()
```




Define Problem | Generate Data | Choose Model | Choose Solver | **Train**

# PINA for Differential Equations Learning

## PINA is much more than a simple software for PINNs

- SOTA Neural Operators and customizable trainings
- TensorBoard API for model training visualization
- Data-Driven Reduced Order Modelling
- Deformation Models by Physics Informed Networks
- .....

**Visit Our Web Page!**

**Reference:**
Coscia, D., Ivagnes, A., Demo, N., & Rozza, G. (2023). Physics-Informed Neural networks for Advanced modeling. Journal of Open Source Software, 8(87), 5352.

# Reduced Order Models enhanced by Deep Learning

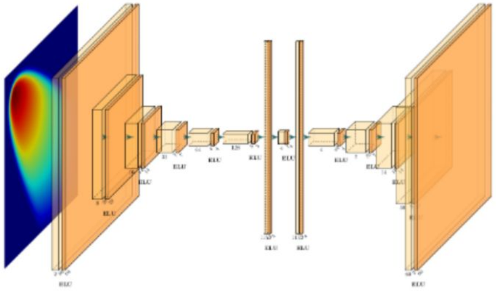
# deep learning # generalization
#offline-online #fast-computing

# Enhancing ROM techniques by Deep Learning

Artificial Intelligence can **enhance** classical ROM techniques for **Computational Fluid Dynamics**

## Enhancing data-driven reduction methods

- Approximation in Reduced Order Model (POD-NN, AE-NN)
- Automatic preprocess data for **dominant advection models**
- Auto-encoders for **dimensionality reduction** and **manifold learning**
- Reduction in **wide parameter space** by means of deep learning **parameter domain decomposition**




**References:**

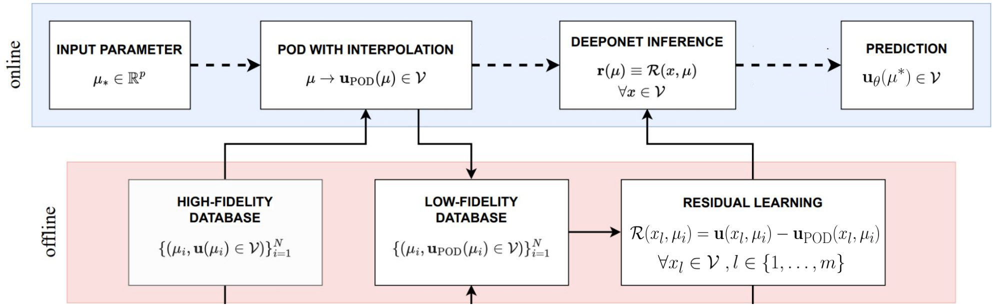
1. F. Romor, G. Stabile, and G. Rozza, (2023). "Non-linear manifold ROM with Convolutional Autoencoders and Reduced Over-Collocation method." *Journal Scientific Computing.*
2. D. Papapicco, N. Demo, M. Girfoglio, G. Stabile, and G. Rozza, (2022). "The Neural Network shifted-proper orthogonal decomposition: A machine learning approach for non-linear reduction of hyperbolic equations.", accepted for *Computer Methods in Applied Mechanics and Engineering.*

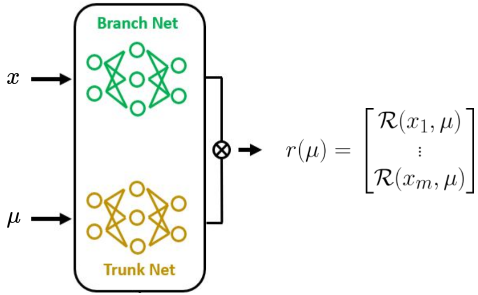
# Multi-fidelity Approach: overcoming POD linearity limitation

## Neural Networks can be adopted for improving the accuracy of a POD-based model

$$\mathbf{u}(\mu) = \mathbf{u}_{\text{POD}}(\mu) + \mathbf{r}(\mu)$$

Residual Learned by DeepONet





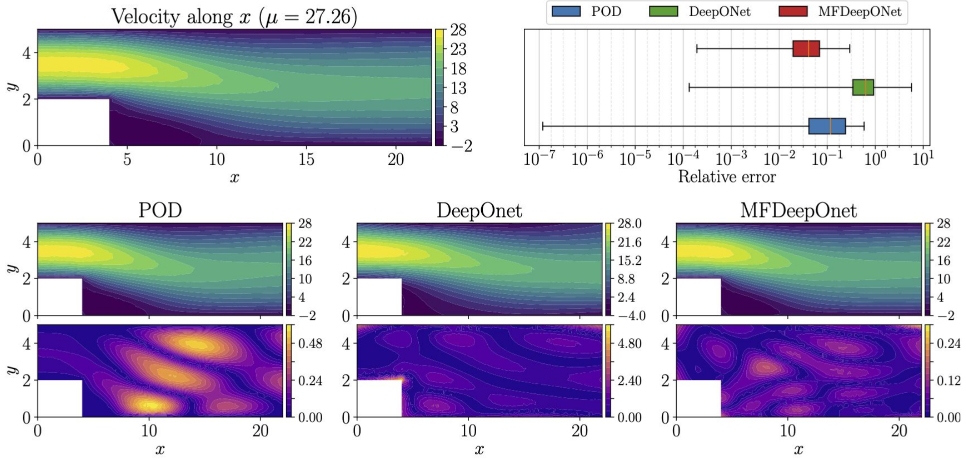
**References:**

1. *Demo, N., Tezzele, M. & Rozza, G. (2023). A DeepONet multi-fidelity approach for residual learning in reduced order modeling. Adv. Model. and Simul. in Eng. Sci. 10, 12 . https://doi.org/10.1186/s40323-023-00249-9*

# Multi-fidelity Approach: overcoming POD linearity limitation

## Advantages

- ➔ **Multi fidelity perspective**
  - ◆ NN learns the difference between the fidelities
- ➔ **Generalization**
  - ◆ Learning the residual using the same snapshots employed for building the POD space increase generalization



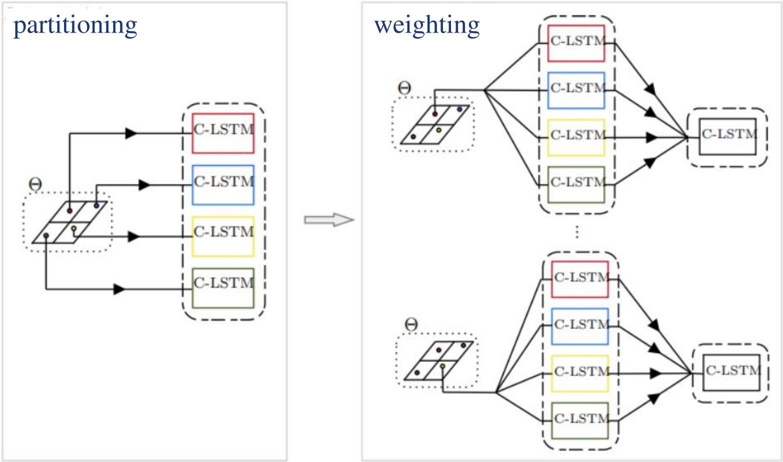

***Navier-Stokes 2d backstep problem test case***


**References:**

1. *Demo, Nicola, Marco Tezzele, and Gianluigi Rozza (2023). "A DeepONet multi-fidelity approach for residual learning in reduced order modeling." arXiv preprint arXiv:2302.12682.*

# Tackling the Curse of Dimensionality by Deep Domain Decomposition

## Generalize the evolution of a system over initial conditions in an extended parameter space




- ➔ **Curse of dimensionality (CoD)** for sampling high dimensional parameter spaces
- ➔ Tackling CoD by **partitioning the parameter space** and averaging the ROM solutions
- ➔ Long-short term memory network (LSTM) coupled with convolutional networks (C-LSTM) for extracting **temporal correlations**


**References:**

1. *Gonnella, I. C., Hess, M. W., Stabile, G., & Rozza, G. (2023). A two-stage deep learning architecture for model reduction of parametric time-dependent problems. Computers & Mathematics with Applications, 149, 115–127. doi:10.1016/j.camwa.2023.08.026*

# Tackling the Curse of Dimensionality by Deep Domain Decomposition

## Results and applications

- Succesfully applied to **ODEs systems** with both periodic and non-periodic dynamics.
- Applied to large discretized **PDE systems** with a previous POD decomposition.

## Rayleigh-Benard cavity

- **97% reduction** in the computational time
- mean percentage error **lower than 1%** in a time-window 50 times larger than the input one.

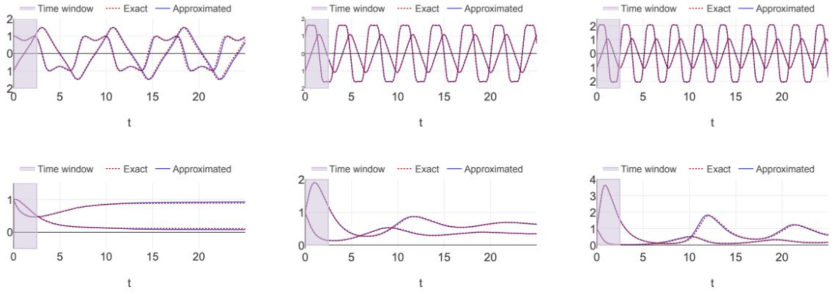



Figure: Duffing Oscillator (above) and Predator Prey system predictions (below) for random testing parameters compared with the exact solutions.

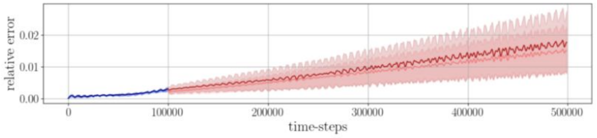



Figure: Relative error progression in time

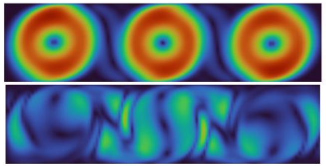


Figure: Velocity field and error at a given advanced time-step.

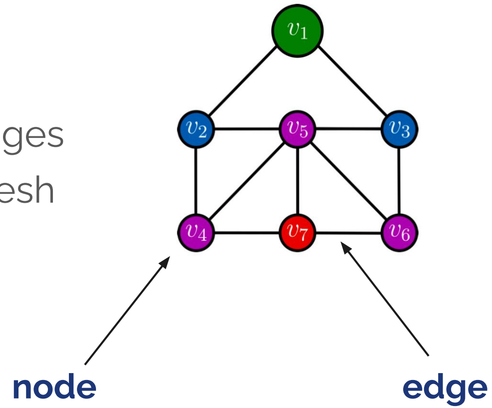

References:

1. *Gonnella, I. C., Hess, M. W., Stabile, G., & Rozza, G. (2023). A two-stage deep learning architecture for model reduction of parametric time-dependent problems. Computers & Mathematics with Applications, 149, 115–127. doi:10.1016/j.camwa.2023.08.026*

# Graph Neural Networks - defeat mesh discrete ROMs

## ROMs based on Graph Neural Networks work on scattered data, no need all simulations to the have same discretization

A Graph is a collection of nodes and edges similar to a mesh




### Message Passing Optimization

**[1] MESSAGE**: for each node $v \in \mathcal{V}$ to be sent $\mathbf{m}_v^{(k)} = \mathfrak{m}^{(k)}(\mathbf{h}_v^{(k-1)})$

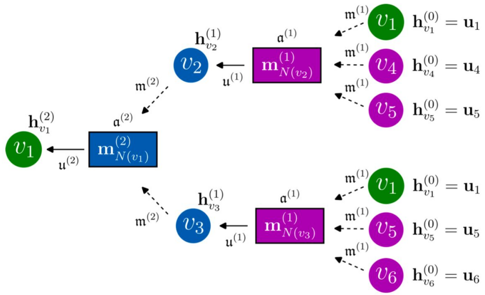
⇝ linear layer: $\mathbf{m}_v^{(k)} = \mathbf{W}^{(k)}\mathbf{h}_v^{(k-1)}$, where $\mathbf{W}^{(k)}$ is the weight matrix

**[2] AGGREGATION**: gather messages $\mathbf{m}_{N(u)}^{(k)} = \mathfrak{a}^{(k)}(\{\mathbf{m}_v^{(k)}, \forall v \in N(u)\})$

⇝ perm. invariant: sum, mean, max over neighbors $N(u)$

**[3] TRANSFORMATION**: nonlinear priors $\mathbf{h}_u^{(k)} = \mathfrak{u}^{(k)}(\mathbf{m}_{N(u)}^{(k)})$

⇝ activation functions: ReLU, tanh, ...

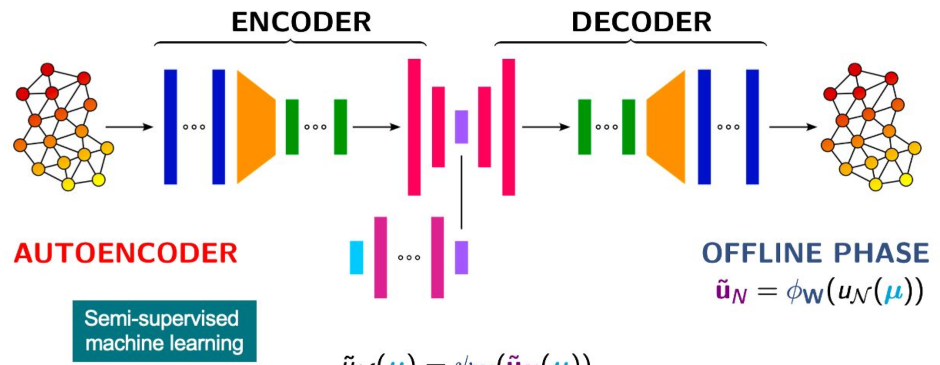


# Nonlinear mesh invariant ROMs via Graph Neural Networks

**GCA-ROM**

Training

https://github.com/fpichi/gca-rom

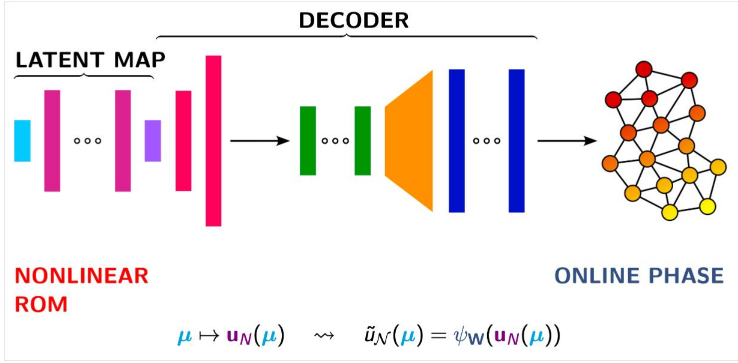



$$\tilde{u}_{\mathcal{N}}(\boldsymbol{\mu}) = \psi_{\mathbf{W}}(\tilde{\mathbf{u}}_N(\boldsymbol{\mu}))$$

**LOSS:** $$\mathcal{L} = \frac{1}{N_{tr}}\sum_{i=1}^{N_{tr}} \|\mathbf{u}_N(\boldsymbol{\mu}^i) - \tilde{\mathbf{u}}_N(\boldsymbol{\mu}^i)\|_2^2 + \frac{1}{N_{tr}}\sum_{i=1}^{N_{tr}} \|\tilde{u}_{\mathcal{N}}(\boldsymbol{\mu}^i) - u_{\mathcal{N}}(\boldsymbol{\mu}^i)\|_2^2$$

latent space loss — full space loss

Testing




- **3D** and **time-dependent** extensions
- **physics-based** loss
- **multi-fidelity** context
- **neural operators**
- **generative** architectures

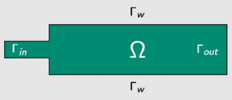

References:

1. *Pichi, Federico, Beatriz Moya, and Jan S. Hesthaven (2024). "A graph convolutional autoencoder approach to model order reduction for parametrized PDEs." Journal of Computational Physics.*

# Nonlinear mesh invariant ROMs via Graph Neural Networks

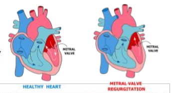
**Bifurcating PDEs in computational Fluid Dynamics**

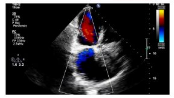
$$\begin{cases} -\mu_0 \Delta \mathbf{u}(\boldsymbol{\mu}) + (\mathbf{u}(\boldsymbol{\mu}) \cdot \nabla)\, \mathbf{u}(\boldsymbol{\mu}) + \nabla p(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\mu_1), \\ \nabla \cdot \mathbf{u}(\boldsymbol{\mu}) = 0 & \text{in } \Omega(\mu_1), \\ \mathbf{u}(\boldsymbol{\mu}) = \mathbf{u}_{\text{in}} & \text{on } \Gamma_{\text{in}}, \\ \mathbf{u}(\boldsymbol{\mu}) = \mathbf{0} & \text{on } \Gamma_{\text{w}}(\mu_1), \\ \mu_0 \frac{\partial \mathbf{u}}{\partial \mathbf{n}}(\boldsymbol{\mu}) - p(\boldsymbol{\mu})\, \mathbf{n} = 0 & \text{on } \Gamma_{\text{out}}, \end{cases}$$




**Parameters**:

- $\mu_0 \in [0.5, 2]$ kinematic viscosity
- $\mu_1 \in [0.5, 2]$ inlet's width
- $N_h = 8157$ dofs
- $M = 3171$ snapshots



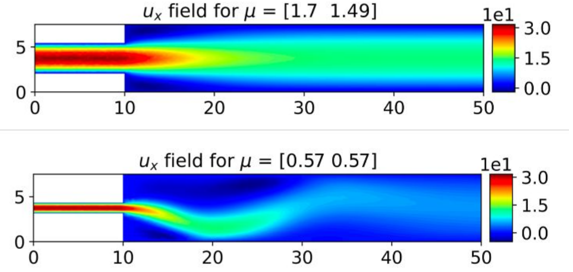
- param. geometry
- nonlinear terms
- vector problem

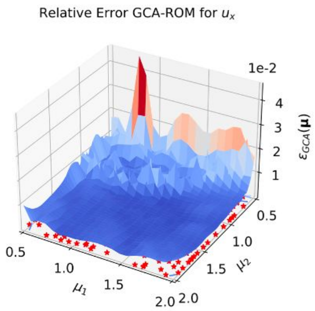
**Hyperparameters**:

- train rate $r_t = 10\%$
- latent $n = 25$, epochs $N_{ep} = 5000$

**Relative errors**:

- mean: $4.62 \cdot 10^{-3}$
- max: $4.55 \cdot 10^{-2}$



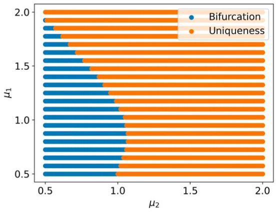




Figure: GCA-ROM

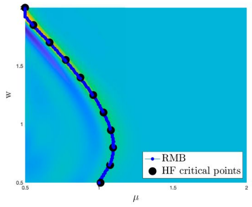


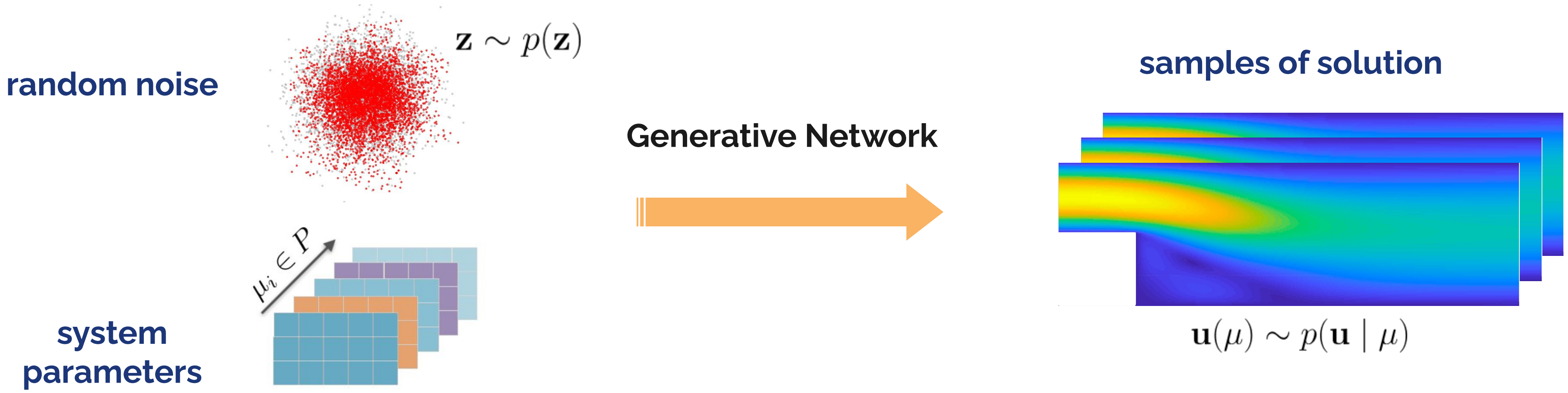

Figure: POD-NN


**References:**

1. *Pichi, Federico, Beatriz Moya, and Jan S. Hesthaven (2024). "A graph convolutional autoencoder approach to model order reduction for parametrized PDEs." Journal of Computational Physics.*

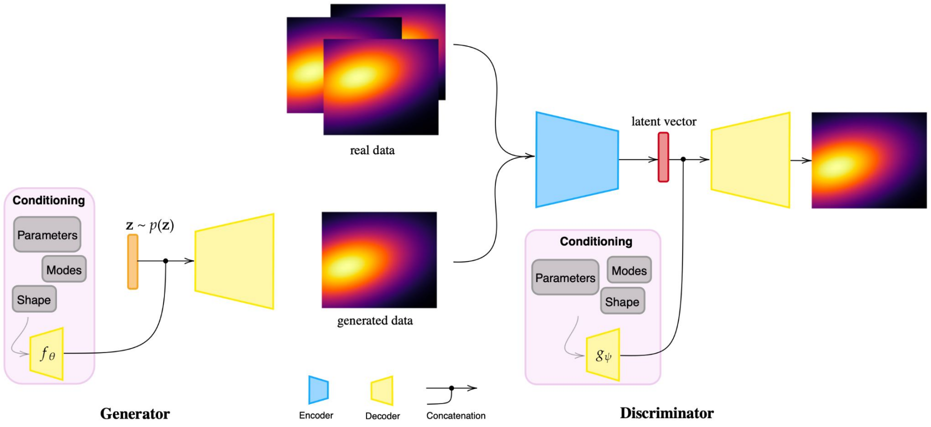
# Generative Models - Quantify Model Uncertainty

- Generative modelling learns **probability distributions** on the data
- **A priori uncertainty quantification** can be done with probability distributions
- **Learning distribution of solutions** to Partial and Stochastic Partial Differential equations

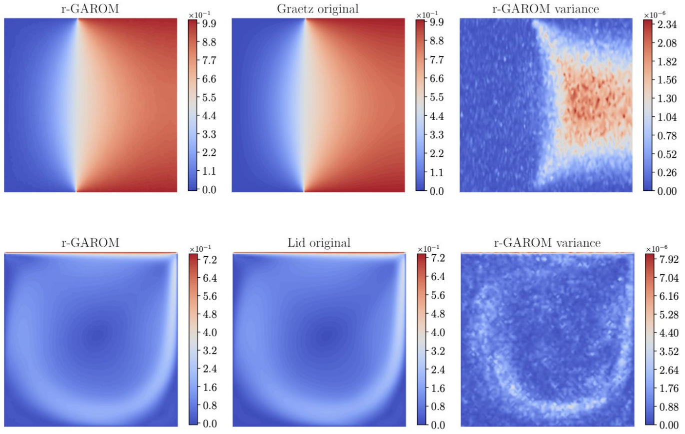


# Generative Models - Quantify Model Uncertainty

## GAN approach to learn the distribution of solutions




## High generation accuracy with error estimates!




solution variance (UQ)

### References


*Coscia, D., Demo, N., & Rozza, G. (2024). Generative Adversarial Reduced Order Modelling. Nature Scientific Report.*

# Conclusions

- It is time to better integrate **Data, Modelling, Analysis, Numerics, Control, Optimization and Uncertainty Quantification** in a new parametrized, reduced and coupled paradigm;

- We need to draw the attention to the fact that "**Science and Engineering could advance with Mathematics (CSE)**"

- **Applied Mathematics as propeller for methodological innovation and technology transfer** by a new generation of talented computational scientists