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

Gianluigi Rozza

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

PINN-PAD conference - University of Padova, Italy
February 22-23, 2024
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.
Parametric Differential Problem are ubiquitous in many field of Natural Science from naval and nautical engineering, to aeronautical engineering and industrial engineering.

References:
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 Methods** 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, ...)
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** Advanced Reduced order modelling Online computational web server for parametric Systems
- **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)
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

2019

Symmetries, High Dimensional Systems, Stochastic Equations, ...

< 2017

Artificial Neural Networks as Reduced Order Models

2019

Neural Operator Learning

2021

---

pod modes

\[ \begin{align*}
\text{Input function} & \xrightarrow{\text{Operator}} \text{Output function} \\
\end{align*} \]
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

\[ \partial_t \phi + u \cdot \nabla \phi = 0 \]
The Data-Driven Approach to Reduced Models

- Reducing Parameter Space
- Applicable for Sensor and Incomplete Data
- Fast Online Phase
Reduced Order Model - Accelerating Numerics

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

| * Input parameters: | $\mu$ (geometry, physical properties, etc.) |
| Parametrized PDE: | $A(u(\mu); \mu) = 0$ |
| * Output: | $u(\mu) \approx u_h(\mu) \approx u^{ROM}(\mu)$ |
| * Input-Output evaluation: | $\mu \rightarrow u_h(\mu) \rightarrow u^{ROM}(\mu)$ |

References:

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:** \( U = M \Sigma V^T \)
- The first \( r \) columns of \( M \) span the **reduced space**
- Evaluation of the **modal coefficients** \( S \)
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**

\[ u^{\text{ROM}}(\mu^*) = M_r \cdot s(\mu^*) \]

*ROM prediction*

\[ u^{\text{ROM}}(\mu^*) \approx u(\mu^*) \]
Physics Informed Neural Network

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

\[
\nu \Delta u + (u \cdot \nabla)u + \nabla p = 0 \\
\nabla \cdot u = 0 \\
\mathbf{u} = \mu \left\{ \frac{1}{223} (x_1 - 2)(5 - x_1), 0 \right\} \\
\mathbf{u} = 0 \\
\nu \frac{\partial u}{\partial n} - pn = 0
\]

Velocity along \( x \) (\( \mu = 27.26 \))
The Physics Informed Neural Network (PINN)

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

\[
\begin{align*}
A(u(x, \mu)) &= 0 \quad x \in \Omega \\
B(u(x, \mu)) &= 0 \quad x \in \partial \Omega
\end{align*}
\]

References:
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.

- Extract coordinates from the domain
- Pass it through a DL model
- Approximate solution
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.

\[
\begin{align*}
    A(u(x, \mu)) &= 0 \quad x \in \Omega \\
    B(u(x, \mu)) &= 0 \quad x \in \partial\Omega
\end{align*}
\]

differential problem

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

residual loss

model
Data and Physical knowledge must be balanced to build a truthful and reliable ML model.
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:
Solve Inverse Problems with PINNs

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

  - The model **equations** are fulfilled:
    \[
    \frac{\partial u}{\partial t} + f\left(\frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \mu_1, \mu_2, \ldots, \mu_n\right) = 0
    \]
  
  - Pre-computed **data** are fitted:
    \[
    u(t, x) = u_{data}(t, x)
    \]

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

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

  \[
  \mathcal{L}_{data} = u - u_{data}
  \]

- **Examples of applications**: find properties of materials to satisfy specific operating conditions.
A first preliminary inverse problem with PINN

Poisson parametric inverse problem:
\[
\begin{align*}
\Delta u &= e^{-2(x-\mu_1)^2-2(y-\mu_2)^2} \quad \text{in } \Omega \\
u &= 0 \quad \text{on } \partial \Omega 
\end{align*}
\]

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:**

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

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:
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
PINAX - Learning Solution to PDE with simple code

Physics Informed Neural network for Advanced 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!
Solving PDEs with PINA - Problem Definition

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))}

\[
\begin{align*}
\Delta u(x, y) &= \sin(\pi x) \sin(\pi y) & (x, y) \in [0, 1]^2 \\
u(x, y) &= 0 & (x, y) \in \partial[0, 1]^2
\end{align*}
\]
Solving PDEs with PINA - Data Generation

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

PINNs equations are evaluated over the neural network on some scattered sample points of the domain.
Solving PDEs with PINA - Model Selection

# 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)
- …..
Solving PDEs with PINA - Solver Selection

```python
# 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

```python
# trainer
trainer = Trainer(pinn, max_epochs, accelerator, batch_size, gradient_clip_val, gradient_clip_algorithm)
# train
trainer.train()
```
PINAs for Differential Equations Learning

PINAs 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
- ...

Reference:
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:
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:
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:
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:
Tackling the Curse of Dimensionality by Deep Domain Decomposition

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

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

References:
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 $v \in V$ to be sent $m_v^{(k)} = m^{(k)}(h_{v}^{(k-1)})$
   \[ m_v^{(k)} = W^{(k)}h_{v}^{(k-1)} \]
   where $W^{(k)}$ is the weight matrix

2. **AGGREGATION**: gather messages $m_{N(u)}^{(k)} = a^{(k)}(\{m_v^{(k)}, \forall v \in N(u)\})$
   \[ \sim \text{perm. invariant: sum, mean, max over neighbors } N(u) \]

3. **TRANSFORMATION**: nonlinear priors $h_v^{(k)} = u^{(k)}(m_{N(u)}^{(k)})$
   \[ \sim \text{activation functions: ReLU, tanh, \ldots} \]
Nonlinear mesh invariant ROMs via Graph Neural Networks

References:
Nonlinear mesh invariant ROMs via Graph Neural Networks

References:

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

random noise

\[ z \sim p(z) \]

system parameters

Generative Network

\[ u(\mu) \sim p(u | \mu) \]

samples of solution
Generative Models - Quantify Model Uncertainty

GAN approach to learn the distribution of solutions

High generation accuracy with error estimates!

References
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