# Meeting with Kent-André Mardal

Felipe Rocha
felipe.figueredorocha@epfl.ch
Lausanne, 26 October 2021

October 26, 2021

# Outline

# Main Differences

**Conventional Machine-Learning**

- Big data scenario.
- Models almost entirely unknown.
- Predominantly classification.

**Scientific Machine-Learning**

- Small data scenario ($$).
- Knowledge of at least part of the model.
- Predominantly regression
- Accuracy is normally a **must**.
- Physical constraints, symmetries.

# High Dimensional Problems

Classical Numerical Methods work extremely well for $d \leq 4$ but too expensive for high-dimensions.

**Explicitly High-Dimensional**

- Stochastic PDEs.
- Black-Scholes.
- ...

**Implicitly High-Dimensional (Many-Queries)**

- Parametric PDEs.
- Uncertainty Quantification.
- Design Optimisation.
- Inverse Problems.
- ...

# My Personal Answer

**NO, we don't, if:**

- The "physics" is known: model and parameters.
- When the dimension is low ($\Omega \subset \mathbb{R}^d$, $u : \Omega \to \mathbb{R}^{N_u}$).
- Speed is not a issue and/or computational resources are available.
- Accuracy is the main requisite.

# My Personal Answer

**NO, we don't, if:**

- The "physics" is known: model and parameters.
- When the dimension is low ($\Omega \subset \mathbb{R}^d$, $u : \Omega \to \mathbb{R}^{N_u}$).
- Speed is not a issue and/or computational resources are available.
- Accuracy is the main requisite.

**MAYBE, we might need, if:**

- The "physics" is partially unknown: model and/or parameters.
- When the dimension is high (curse of dimensionality).
- Speed is an issue and we can cope with some loss of accuracy.

# My Personal Answer

**NO, we don't, if:**

- The "physics" is known: model and parameters.
- When the dimension is low ($\Omega \subset \mathbb{R}^d$, $u : \Omega \to \mathbb{R}^{N_u}$).
- Speed is not a issue and/or computational resources are available.
- Accuracy is the main requisite.

**MAYBE, we might need, if:**

- The "physics" is partially unknown: model and/or parameters.
- When the dimension is high (curse of dimensionality).
- Speed is an issue and we can cope with some loss of accuracy.

**YES, we do, in case:**

- all together.
- we have data but no model.

# Parametric PDEs

- $\Omega \subset \mathbb{R}^d$, $u : \Omega \to \mathbb{R}^{N_u}$, $d$ small.
- Strong Format: Given $\boldsymbol{\mu} \in P \subset \mathbb{R}^{N_d}$, find $u \in U$ such that

$$\mathcal{L}_{\boldsymbol{\mu}} u = f_{\boldsymbol{\mu}}. \tag{1}$$

- Weak format : Given $\boldsymbol{\mu} \in P \subset \mathbb{R}^{N_d}$, find $u \in U$ such that

$$a(\boldsymbol{\mu}; u, v) = b(\boldsymbol{\mu}; v) \quad \forall v \in V. \tag{2}$$

- Discrete format (Linear): Given $\mathbf{A} : P \to \mathbb{R}^{N_h, N_h}$, $\mathbf{b} : P \to \mathbb{R}^{N_h}$, solve $\mathbf{U}_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$

$$\mathbf{A}(\boldsymbol{\mu})\mathbf{U}_h(\boldsymbol{\mu}) = \mathbf{b}(\boldsymbol{\mu}). \tag{3}$$

- Observations: $\boldsymbol{\eta} : \mathbb{R}^{N_p} \times U \to \mathbb{R}^{N_\eta}$, $(\boldsymbol{\mu}, u) \mapsto \boldsymbol{\eta}(\boldsymbol{\mu}, u)$.

# Reduced Basis (POD)

## Offline phase:

- $\mathbf{A}(\mu)\mathbf{U}_h = \mathbf{f}_h(\mu)$ (size $N_h$ big).
- Simulate $\mathbb{S} = \{\mathbf{u}^{(i)}\}_{i=1}^{N_s}$, $\mathbf{u}^{(i)} \in L^2(\Omega)$
- $[\mathbf{C}]_{ij} = (\mathbf{u}^{(i)}, \mathbf{u}^{(j)})_{L^2(\Omega)}$.
- $\mathbf{C} = \mathbf{V} diag(\lambda_1, \ldots, \lambda_{N_s^0})\mathbf{V}^T$, $\mathbf{V} \in \mathrm{Orth}$.
- for $i = 1, \ldots, N_{max}(= N_h)$ $\xi^{(i)} = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{N_s^0} V_{ij}\mathbf{u}^{(j)}$
- $\mathcal{B}_{rb}(N_{rb}) = \{\xi^{(i)}\}_{i=1}^{N_r b}$.

# Reduced Basis (POD)

**Offline phase:**

- $\mathbf{A}(\boldsymbol{\mu})\mathbf{U}_h = \mathbf{f}_h(\boldsymbol{\mu})$ (size $N_h$ big).
- Simulate $\mathbb{S} = \{\mathbf{u}^{(i)}\}_{i=1}^{N_s}$, $\mathbf{u}^{(i)} \in L^2(\Omega)$
- $[\mathbf{C}]_{ij} = (\mathbf{u}^{(i)}, \mathbf{u}^{(j)})_{L^2(\Omega)}$.
- $\mathbf{C} = \mathbf{V} diag(\lambda_1, \ldots, \lambda_{N_s^0})\mathbf{V}^T$, $\mathbf{V} \in \mathrm{Orth}$.
- for $i = 1, \ldots, N_{max}(= N_h)$ $\boldsymbol{\xi}^{(i)} = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{N_s^0} V_{ij}\mathbf{u}^{(j)}$
- $\mathcal{B}_{rb}(N_{rb}) = \{\boldsymbol{\xi}^{(i)}\}_{i=1}^{N_r b}$.

**Online phase:**

- Choose $N_{rb} << N_h$.
- $\mathbf{V}_{rb} := \mathbf{V}[:, : N_{rb}] \in \mathbb{R}^{N_h, N_{rb}}$
- Given $\boldsymbol{\mu} \in P$

$$\mathbf{A}_{rb}(\boldsymbol{\mu}) := \mathbf{V}_{rb}^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}_{rb}$$
$$\mathbf{b}_{rb}(\boldsymbol{\mu}) := \mathbf{V}_{rb}^T\mathbf{b}(\boldsymbol{\mu})$$

- Solve the reduced system $N_{rb} \times N_{rb}$:

$$\mathbf{A}_{rb}(\boldsymbol{\mu})\mathbf{U}_{rb}(\boldsymbol{\mu}) = \mathbf{b}_{rb}(\boldsymbol{\mu})$$

- Reconstructed solution:

$$\mathbf{U}_{rb}^R(\boldsymbol{\mu}) = \mathbf{V}_{rb}\mathbf{U}_{rb}(\boldsymbol{\mu})$$

# Reduced Basis

## Certified Error Analysis

- $(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)})_{L^2(\Omega)} = \delta_{ij}$
- $\Pi_N \mathbf{w} := \sum_{i=1}^{N} (\mathbf{w}, \boldsymbol{\xi}^{(i)})_{L^2(\Omega)}$
- $\mathcal{E}_{POD}(N) = \sum_{j=N+1}^{N_{max}} \lambda_j = \sum_{i=1}^{N_s^0} \|\mathbf{u}^{(i)} - \Pi_N \mathbf{u}^{(i)}\|^2$
- $\mathcal{E}_{POD}^{mse}(N) = \frac{1}{N_s} \sum_{j=N+1}^{N_{max}} \lambda_j$

## Affine Decomposition

If $\mathbf{A}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_A} \alpha_i^A(\boldsymbol{\mu}) \mathbf{A}^q$, and $\mathbf{b}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \alpha_i^b(\boldsymbol{\mu}) \mathbf{b}^q$ then

$$\mathbf{A}_{rb}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_A} \alpha_i^A(\boldsymbol{\mu}) \mathbf{A}_{rb}^q \quad , \quad \mathbf{b}_{rb}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \alpha_i^b(\boldsymbol{\mu}) \mathbf{b}_{rb}^q. \tag{4}$$

with $\mathbf{A}_{rb}^q = \mathbf{V}_{rb}^T \mathbf{A}^q \mathbf{V}_{rb}$, $\mathbf{b}_{rb}^q = \mathbf{V}_{rb}^T \mathbf{b}^q$.

# Physically Inspired Neural Networks

The *Journal of Computational...*

Read more

Most Downloaded    Recent Articles    **Most Cited**

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

- Open access

M. Raissi | P. Perdikaris | ...

Hidden physics models: Machine learning of nonlinear partial differential equations

Maziar Raissi | George Em Karniadakis

| TITLE | CITED BY | YEAR |
|---|---|---|
| Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations<br>M Raissi, P Perdikaris, GE Karniadakis<br>Journal of Computational Physics 378, 686-707 | 1156 * | 2019 |
| Hidden physics models: Machine learning of nonlinear partial differential equations<br>M Raissi, GE Karniadakis<br>Journal of Computational Physics 357, 125-141 | 426 | 2018 |

# Physically Inspired Neural Networks (PINN)

- Basic idea: regularise the loss function of the PDE residual.
- It can be used in both forward and inverse problems.

# Physically Inspired Neural Networks (PINN)

- Basic idea: regularise the loss function of the PDE residual.
- It can be used in both forward and inverse problems.
- Example forward problem: Burguer's:

$$\begin{cases} u_t + uu_x - \nu u_{xx} = 0 = f(u(x,t), x, t) & (x,t) \in (-1,1) \times (0,1) \\ u(0,x) = -\operatorname{sen}(\pi x) \\ u(t,-1) = u(t,1) = 0 \end{cases}$$

# Physically Inspired Neural Networks (PINN)

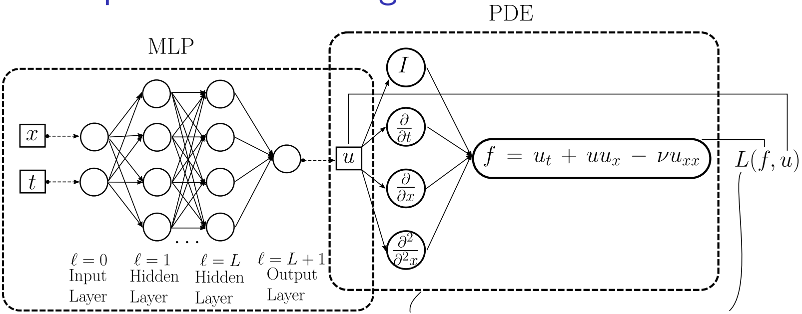
- Basic idea: regularise the loss function of the PDE residual.
- It can be used in both forward and inverse problems.
- Example forward problem: Burguer's:

$$\begin{cases} u_t + uu_x - \nu u_{xx} = 0 = f(u(x,t), x, t) & (x,t) \in (-1,1) \times (0,1) \\ u(0,x) = -\operatorname{sen}(\pi x) \\ u(t,-1) = u(t,1) = 0 \end{cases}$$

- $\mathbf{x}^i = (x^i, t^i)$, $\mathbf{y}^i = (u^i)$ at $i = 1, \ldots, N_u$ points on boundary.
- Loss for $\mathcal{N}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^2 \to \mathbb{R}$

$$L(\boldsymbol{\theta}) = \frac{1}{N_u} \sum_{i=1}^{N_u} |\mathcal{N}(\mathbf{x}^i, \boldsymbol{\theta}) - \mathbf{y}^{\mathbf{i}}|^2 + \frac{1}{N_f} \sum_{i=1}^{N_f} |f(\mathcal{N}(\mathbf{x}^i, \boldsymbol{\theta}), \mathbf{x}^i)|^2 \qquad (5)$$

# PINN - Implementation using automatic differentiation




```python
# The actual PINN
def f_model(self):
    # ...
    with tf.GradientTape(persistent=True) as tape:
        # ...
        u = self.MLPmodel(X_f)
        u_x = tape.gradient(u, self.x_f)

    u_xx = tape.gradient(u_x, self.x_f)
    u_t = tape.gradient(u, self.t_f)

    return u_t + u*u_x - self.nu*u_xx
```

```python
def loss(self, u, u_pred):
    f_pred = self.f_model()
    return tf.reduce_mean(tf.square(u - u_pred)) + \
        tf.reduce_mean(tf.square(f_pred))
```

- https://github.com/maziarraissi/PINNs
- https://github.com/pierremtb/PINNs-TF2.0
- Tensorflow ( https://www.tensorflow.org/).

# Comparison Methods

| | FEM | PINN | ROM |
|---|---|---|---|
| Space | Basis Functions | Neural Networks | Smart Basis Functions |
| Operators | Weak-form | Automatic Differentiation | Weak-form |
| Solver | Linear (Iterative) | (Stochastic) Gradient Descedent | Linear (Direct) |
| Evaluation | Interpolation | Inference | Interpolation |
| Offline phase | No | No | Yes |

- PINN does not enforce exactly solutions satisfy the differential equations, boundary conditions, physical constraints, etc.
- PINNs are slow compared to classical methods. The advantage is for high-dimensional PDEs (Raissi,2018).

**Idea**

Reduced-Order methods are fast, satisfy physical constraints and have good approximation guarantees. Why not combine them with DNN?

# Two ideas to enrich DNN + RB

## POD-DNN (Hesthaven and Ubbiali, 2018)[a]

[a] J. S. Hesthaven, S. Ubbiali, *N*on-intrusive reduced order modeling of nonlinear problems using neural networks, JCP, 2018

Find the mapping from parameters to PDE solution (without linear systems) $\mathcal{N}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^{N_p} \to \mathbb{R}^{N_{rb}}$, with $N_{rb} << N_h$ and $\mathbf{u}_h(\boldsymbol{\mu}) \approx \sum_{i=1}^{N_{rb}} [\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\theta})]_i \boldsymbol{\xi}^{(i)}$

## RB-DNN (Dal Santo et al, 2020)[a]

[a] N. Dal Santo, S. Deparis, L. Pegolotti, *D*ata driven approximation of parametrized PDEs by reduced basis and neural networks, JCP, 2020

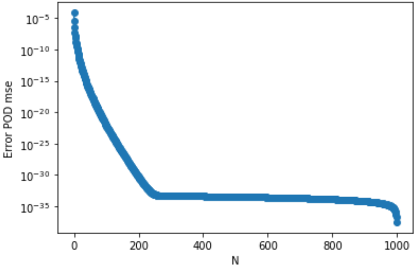
- RB solvers as activation functions.
- Used for parameter identification methods ($\boldsymbol{\mu}$) and reconstruction of the solution of PDEs from scattered measurements.
- $\mathcal{N}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^{N^{in}} \to \mathbb{R}^{N^{out}}$, $\mathbf{u}_h^{in} \in \mathbb{R}^{N^{in}}, \mathbf{u}_h^{out} \in \mathbb{R}^{N^{out}}$

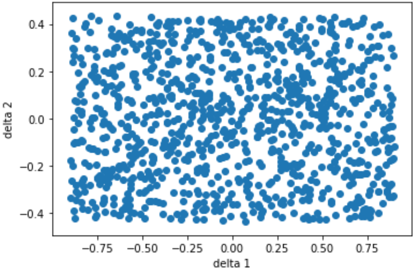
# Poisson with anisotropic diffusibility

- Let $\Omega = [0,1] \times [0,1] \subset \mathbb{R}^2$, find $u : \Omega \to \mathbb{R}$ tal que

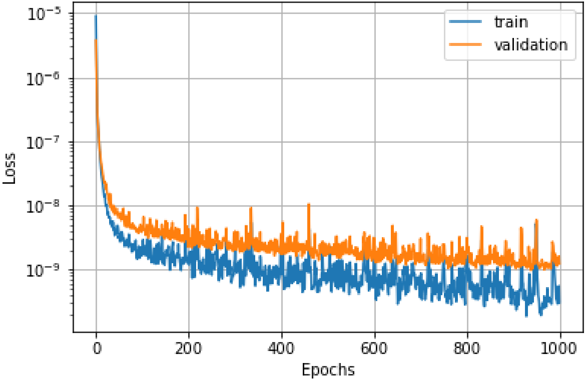
$$\begin{cases} -\operatorname{div}(K(\mu_1,\mu_2)\nabla u) & = f \quad \text{em } \Omega \\ u & = 0 \quad \text{em } \partial\Omega \end{cases} \tag{6}$$

com $K(\mu_1,\mu_2) = \begin{bmatrix} 1+\mu_1 & \mu_2 \\ \mu_2 & 1-\mu_1 \end{bmatrix}$, $\mu_1 \in [-0.9, 0.9]$, $\mu_2 \in [-0.43, 0.43]$, $N_s = 1000$.

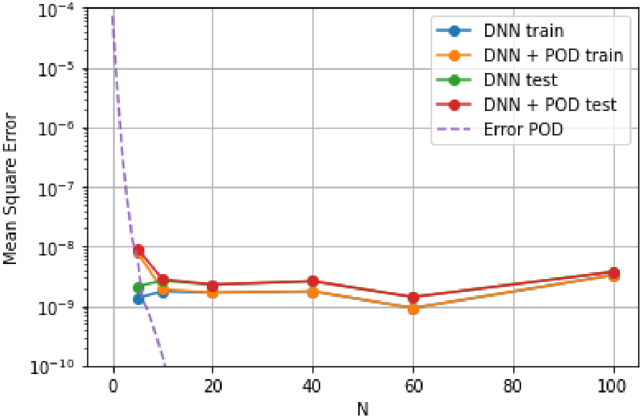


# Training N = 60: $L = 3$, $n_i = 100$, all ReLU



# Results $N \in [5, 10, 20, 40, 60, 80, 100]$



# POD-NN: final comments

- The final solution is **physical**, since is composed by linear combinations of a physical reduced base.
- The solution is not improved increasing the components since the problem is simple.
- Implementation of the example in the EAMC Repository `https://github.com/felipefr/galerkinML_EAMC2021.git`.
- It uses Fenics (`https://fenicsproject.org/`) and Tensorflow ( `https://www.tensorflow.org/`).

# RB-DNN : Formulation

Choose $N_{rb} << N_h$, $\mathbf{V}_{rb} \in \mathbb{R}^{N_h,N_{rb}}$ and $\mathbf{R}_{out} \in \mathbb{R}^{N_{out},N_h}$

$$\text{Remember} \begin{cases} \mathbf{A}_{rb}(\boldsymbol{\mu}) = \mathbf{V}_{rb}^T \mathbf{A}(\boldsymbol{\mu}) \mathbf{V} = \sum_{q=1}^{Q_A} \alpha_i^A(\boldsymbol{\mu}) \mathbf{A}_{rb}^q \\ \mathbf{b}_{rb}(\boldsymbol{\mu}) = \mathbf{V}_{rb}^T \mathbf{b}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \alpha_i^b(\boldsymbol{\mu}) \mathbf{b}_{rb}^q \end{cases} \tag{7}$$

# RB-DNN : Formulation

Choose $N_{rb} << N_h$, $\mathbf{V}_{rb} \in \mathbb{R}^{N_h, N_{rb}}$ and $\mathbf{R}_{out} \in \mathbb{R}^{N_{out}, N_h}$

$$\text{Remember} \begin{cases} \mathbf{A}_{rb}(\boldsymbol{\mu}) = \mathbf{V}_{rb}^T \mathbf{A}(\boldsymbol{\mu}) \mathbf{V} = \sum_{q=1}^{Q_A} \alpha_i^A(\boldsymbol{\mu}) \mathbf{A}_{rb}^q \\ \mathbf{b}_{rb}(\boldsymbol{\mu}) = \mathbf{V}_{rb}^T \mathbf{b}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \alpha_i^b(\boldsymbol{\mu}) \mathbf{b}_{rb}^q \end{cases} \tag{7}$$

$$\sigma_{RB} : \mathbb{R}^{Q_A+Q_b} \to \mathbb{R}^{out}$$

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}^A, \boldsymbol{\alpha}^b) \mapsto \mathbf{u}_h^{out} = \sigma_{RB}(\boldsymbol{\alpha}) = \mathbf{R}_{out} \mathbf{V}_{rb} \mathbf{A}_{rb}^{-1}(\boldsymbol{\alpha}) \mathbf{b}_{rb}(\boldsymbol{\alpha})$$

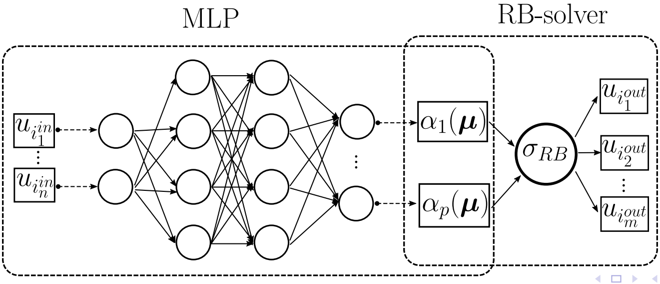
# RB-DNN : Formulation

Choose $N_{rb} << N_h$, $\mathbf{V}_{rb} \in \mathbb{R}^{N_h, N_{rb}}$ and $\mathbf{R}_{out} \in \mathbb{R}^{N_{out}, N_h}$

$$\text{Remember} \begin{cases} \mathbf{A}_{rb}(\boldsymbol{\mu}) = \mathbf{V}_{rb}^T \mathbf{A}(\boldsymbol{\mu}) \mathbf{V} = \sum_{q=1}^{Q_A} \alpha_i^A(\boldsymbol{\mu}) \mathbf{A}_{rb}^q \\ \mathbf{b}_{rb}(\boldsymbol{\mu}) = \mathbf{V}_{rb}^T \mathbf{b}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \alpha_i^b(\boldsymbol{\mu}) \mathbf{b}_{rb}^q \end{cases} \tag{7}$$

$$\sigma_{RB} : \mathbb{R}^{Q_A + Q_b} \to \mathbb{R}^{out}$$

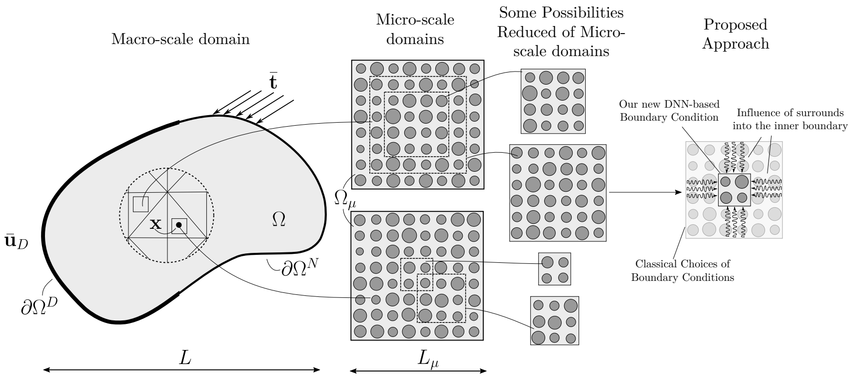
$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}^A, \boldsymbol{\alpha}^b) \mapsto \mathbf{u}_h^{out} = \sigma_{RB}(\boldsymbol{\alpha}) = \mathbf{R}_{out} \mathbf{V}_{rb} \mathbf{A}_{rb}^{-1}(\boldsymbol{\alpha}) \mathbf{b}_{rb}(\boldsymbol{\alpha})$$



# RB-DNN: final comments

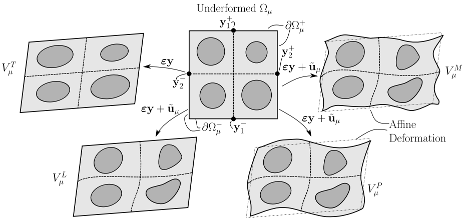
- For the identification of properties, comparable errors are found ( MSE $\mathcal{O}(10^{-3})$ (in normalised parameters) ) with the "trivial" network.
- "Trivial" network . $\mathcal{N}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^{N^{in}} \to \mathbb{R}^{N^{out}+N^{p}}$, $\mathbf{u}_h^{in} \in \mathbb{R}^{N^{in}}, (\mathbf{u}_h^{out}, \boldsymbol{\mu}) \in \mathbb{R}^{N^{out}}$
- Usually $\boldsymbol{\mu}$ is not available from experiments.
- The trivial approach does not allow to predict out of the region $N_{out}$ have been collected.
- The final solution is **physical**, since is composed by linear combinations of a physical reduced base.
- Implementation for reference https://github.com/ndalsanto/PDE-DNN.
- I have my own version for this implementation, let me know if you need.

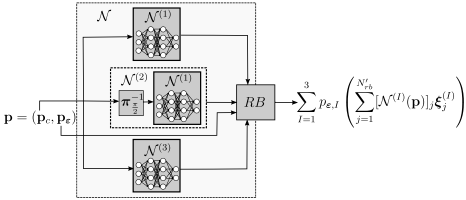
# Enhancing Computational Homogenisation

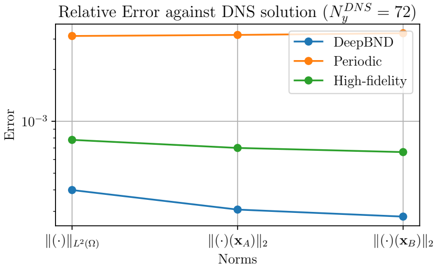



- F. Rocha, S. Deparis, P. Antolin, A. Buffa. Enhancing defective Multi-scale Solid Mechanics formulations via Machine Learning (in preparation). Journal of Computational Physics, 2021.

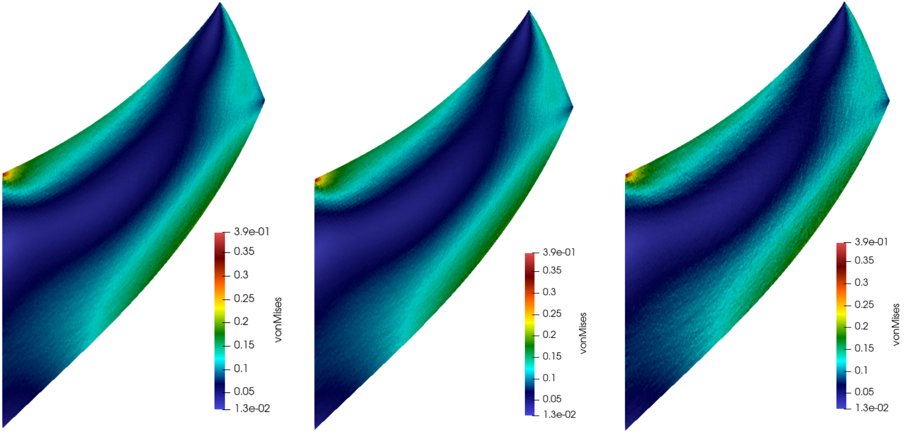
# Enhancing Computational Homogenisation



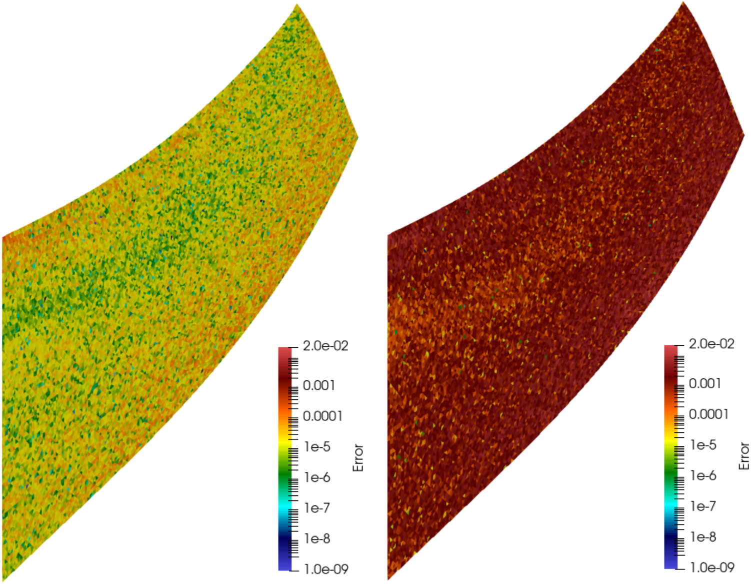



# Enhancing Computational Homogenisation



# Enhancing Computational Homogenisation



# Other uses : enhancing the Numerics and Physics

- Control numerical oscillations in HDG (Discacciati et al, 2020). Artificial viscosity in spectral methods (Lukas Schwander et al, 2021). Troubled-cells in finite volume schemes (Ray and Hestaven, 2019).

# Other uses : enhancing the Numerics and Physics

- Control numerical oscillations in HDG (Discacciati et al, 2020).
  Artificial viscosity in spectral methods (Lukas Schwander et al, 2021).
  Troubled-cells in finite volume schemes (Ray and Hestaven, 2019).
- Replace constitutive laws : Data-driven material modelling.

$$\boldsymbol{\sigma}_{\boldsymbol{\mu}}(\nabla^s \mathbf{u}) = \mathcal{N}((\boldsymbol{\mu}, \nabla^s \mathbf{u}), \boldsymbol{\theta}) \tag{8}$$

# Other uses : enhancing the Numerics and Physics

- Control numerical oscillations in HDG (Discacciati et al, 2020). Artificial viscosity in spectral methods (Lukas Schwander et al, 2021). Troubled-cells in finite volume schemes (Ray and Hestaven, 2019).
- Replace constitutive laws : Data-driven material modelling.

$$\boldsymbol{\sigma}_{\boldsymbol{\mu}}(\nabla^s \mathbf{u}) = \mathcal{N}((\boldsymbol{\mu}, \nabla^s \mathbf{u}), \boldsymbol{\theta}) \tag{8}$$

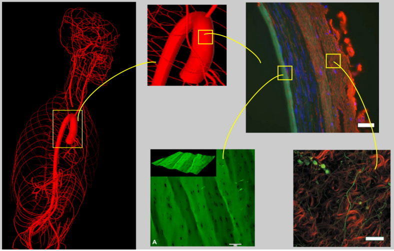
- Truncated domains: To make a computation affordable, we may need truncate a large or a infinite domain to a small region of interest. Boundary conditions are generally not known in these contexts.

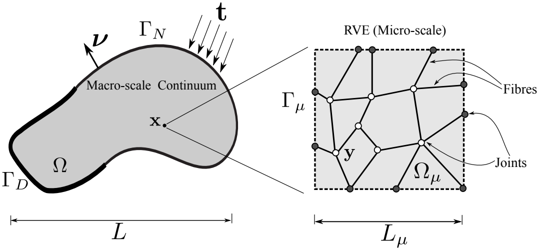
$$\begin{cases} -\nabla a(\boldsymbol{\mu})u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \qquad \begin{cases} -\nabla a|_{\bar{\Omega}}(\boldsymbol{\mu})\bar{u} = f|_{\hat{\Omega}} & \text{in } \bar{\Omega} \subset \Omega \\ \bar{u} = ? = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\theta}) & \text{on } \partial\bar{\Omega} \end{cases}$$

# Final Comments

1. We might need ML methods in case:
   - The "physics" is partially unknown: model and/or parameters.
   - When the dimension is high (curse of dimensionality).
   - Speed is an issue and we can cope with some loss of accuracy.
2. Challenges:
   - Stability: Adversarial attacks, Deepfool (Moosavi-Dezfool et al, 2015)
   - Accuracy: Little about how to achieve the architecture.
   - Sample complexity: ImageNet database which contains 14 million hand-annotated images of more than 20,000 categories of subjects. Data-starved Scenario by the computational cost or cost experimental data.
   - Curse of dimensionality: High-dimensional PDEs occur in numerous applications, and parametrized PDEs in UQ applications often involve tens to hundreds of variables. Moreover, the curse of dimensionality is an important consideration in the sample complexity, as the cost of obtaining samples can often dominate the overall cost. There is hope (J. Berner et al, 2020).

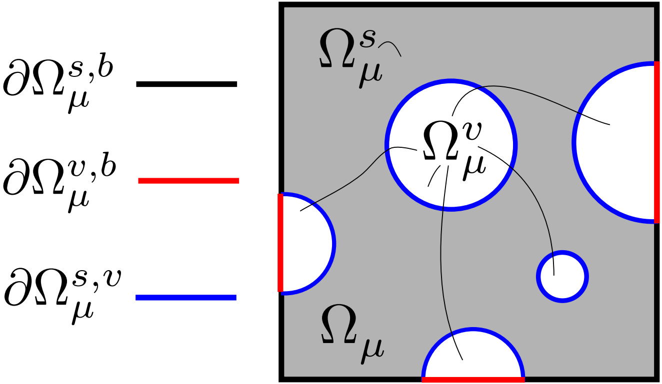
# Computational Homogenisation of Fibrous Materials






- 2018, F. Rocha, P. Blanco, P. Sánchez, and R. Feijóo. Multi-scale modelling of arterial tissue: Linking networks of fibres to continua. CMAME

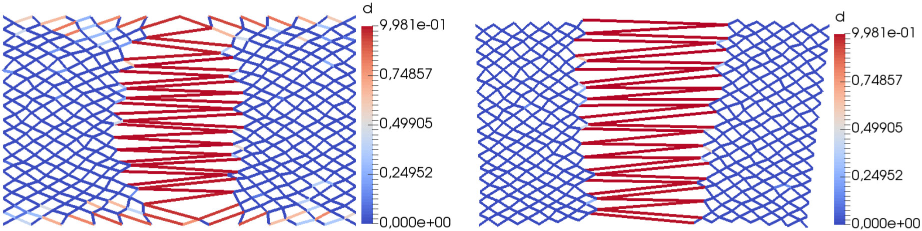
# Boundary Conditions

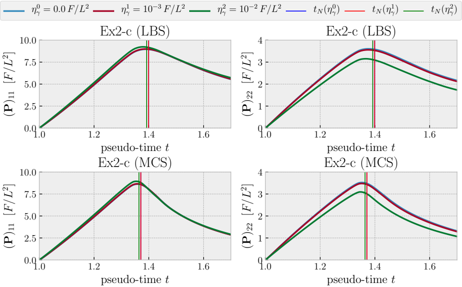



$$\int_{\Gamma_\mu^{s,b}} \tilde{\mathbf{u}}_\mu \otimes (\mathbf{n}_\mu - \bar{\mathbf{n}}_\mu)\, \mathrm{d}\Gamma_\mu = \mathbf{O}.$$

- 2019 , F. Rocha, Multiscale Modelling of Fibrous Materials: from the elastic regime to failure detection in soft tissues, PhD Thesis

# Strain Localization




- 2021, F. Rocha, P. Blanco, P. Sánchez, E. de Souza Neto, and R. Feijóo. Damage-driven strain localisation in networks of fibres: A computational homogenisation approach. Computer and Structures