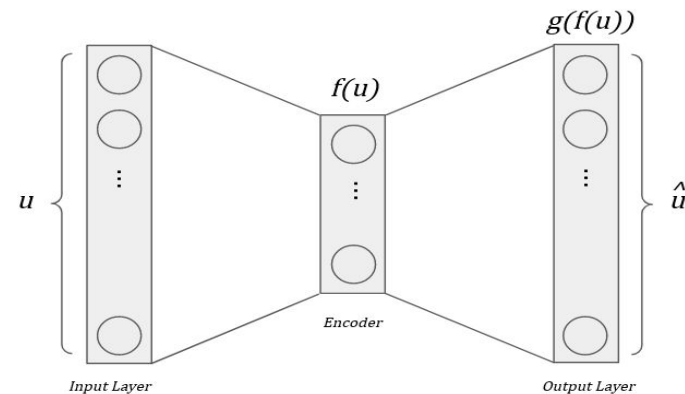
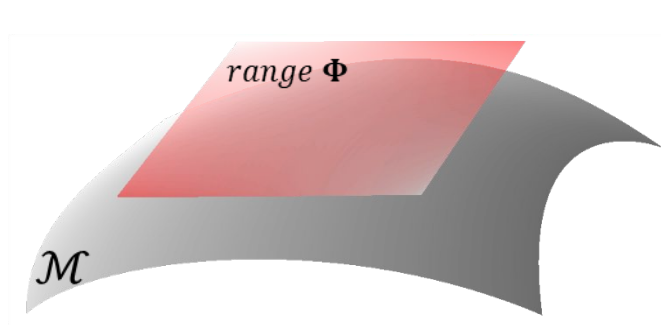


# *From Linear Mappings to Deep Learning for Model Reduction of Numerical Simulations of Industrial Interest:*



**Mr. Raul Bravo**  
Prof. Riccardo Rossi  
Prof. Joaquin Hernandez  
Mr. Carlos Roig

# Presenting ourselves



Prof. Riccardo Rossi  
UPC BarcelonaTech  
CIMNE  
Kratos co-founder  
[rrossi@cimne.upc.edu](mailto:rrossi@cimne.upc.edu)



Prof. Joaquin Hernandez  
Aerospace Engineering School  
UPC BarcelonaTech  
CIMNE  
[jhortega@cimne.upc.edu](mailto:jhortega@cimne.upc.edu)



Kratos github site



Raul Bravo  
PhD Student  
Projection-based ROMs  
[jrbravo@cimne.upc.edu](mailto:jrbravo@cimne.upc.edu)



Carlos Roig  
PhD Student  
Deep Learning ROMs  
[croig@cimne.upc.edu](mailto:croig@cimne.upc.edu)

# Objetives of the talk

- Presenting a ROM framework implemented on a powerful open source FEM software.

FOM:  $\mathbf{u} \in \mathbb{R}^n$

ROM:  $\mathbf{q} \in \mathbb{R}^k$

- POD

$$\mathbf{u} \approx \Phi \mathbf{q}$$

- Local POD

$$\mathbf{u}_{new} \approx \mathbf{u}_{old} + \Phi^i \mathbf{q}$$

- Autoencoders

$$\mathbf{u} \approx g(\mathbf{q})$$

# Kratos Project

The screenshot shows the GitHub repository page for Kratos Multiphysics. The repository is on the master branch, has 659 branches, and 32 tags. It has 78,746 commits. The repository is owned by farrufat-cimne. The repository contains several folders and files, including .github, applications, cmake\_modules, documents, external\_libraries, kratos, scripts, .gitignore, .pydocstyle, CMakeLists.txt, INSTALL.md, README.md, and license.txt. The README.md file is selected and shows the Kratos Multiphysics logo and a description of the project. The logo consists of the word 'KRATOS' in a bold, black, sans-serif font, with 'MULTI-PHYSICS' in a smaller, black, sans-serif font below it. To the right of the text is a teal-colored 3D cube. Below the logo, there are several badges: 'release 8.1', 'license BSD', 'Nightly Build passing', and 'DOI'. The description of the project states: 'KRATOS Multiphysics ("Kratos") is a framework for building parallel, multi-disciplinary simulation software, aiming at

**About**  
Kratos Multiphysics (A.K.A Kratos) is a framework for building parallel multi-disciplinary simulation software. Modularity, extensibility and HPC are the main objectives. Kratos has BSD license and is written in C++ with extensive Python interface.

[www.cimne.com/kratos/](http://www.cimne.com/kratos/)

python c-plus-plus multi-platform  
openmp mpi parallel-computing  
fem bsd-license numerical-methods  
multiphysics dem kratos  
kratos-multiphysics

Readme  
View license

**Releases** 32  
Kratos Multiphysics 8.1 Latest  
on Nov 25, 2020  
+ 31 releases

**Packages**  
No packages published

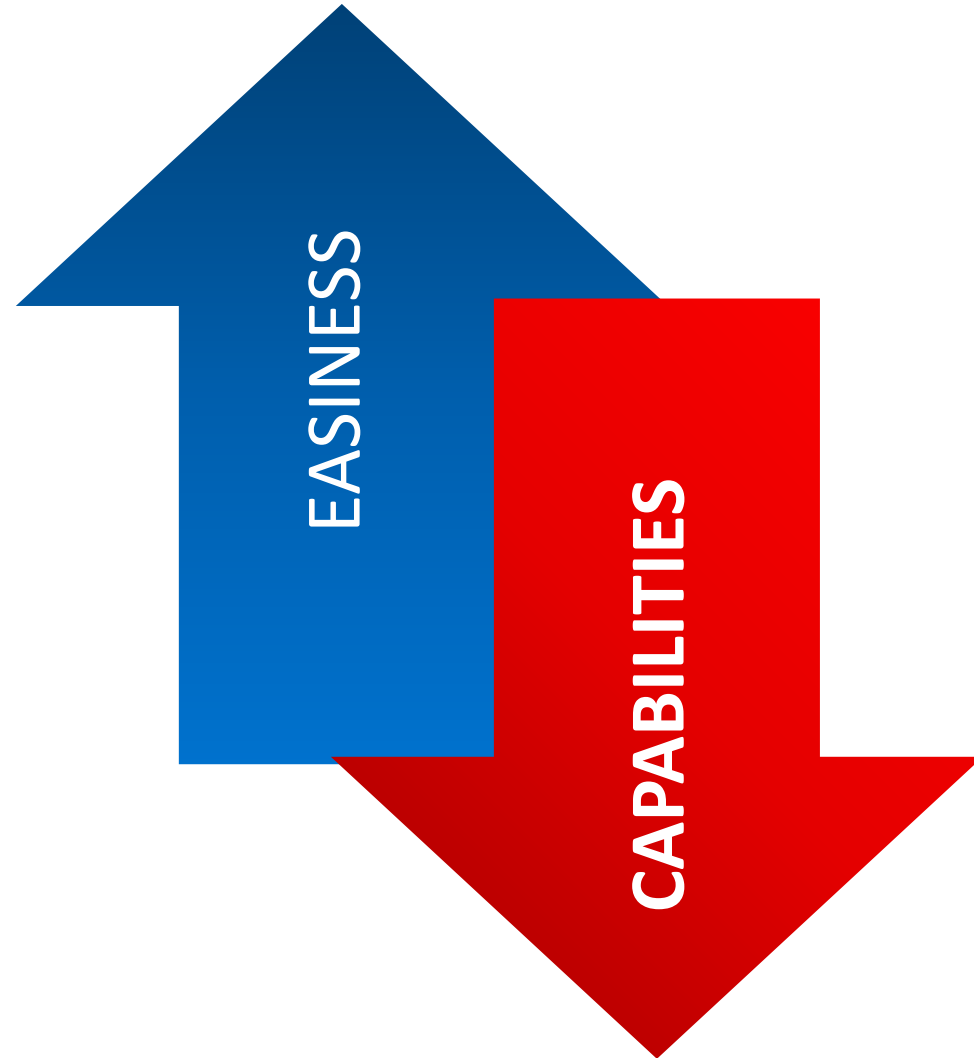
**Contributors** 112

# Using Kratos

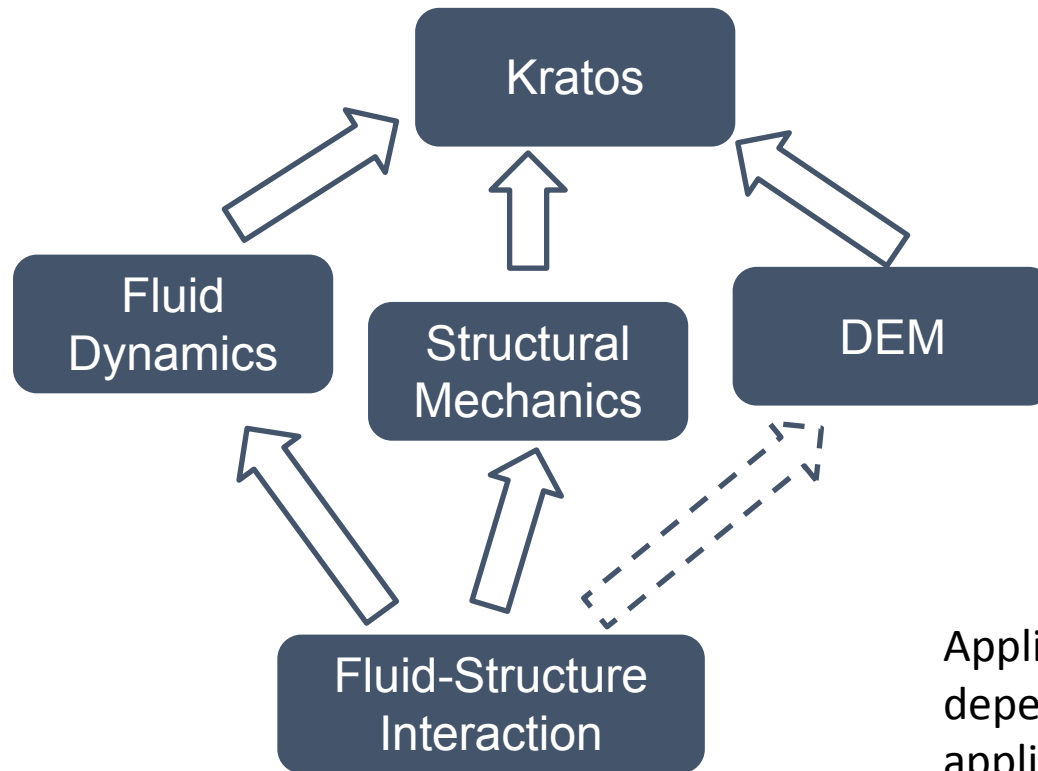
Graphical Interface

Python Interface

Applications



# How is Kratos structured?

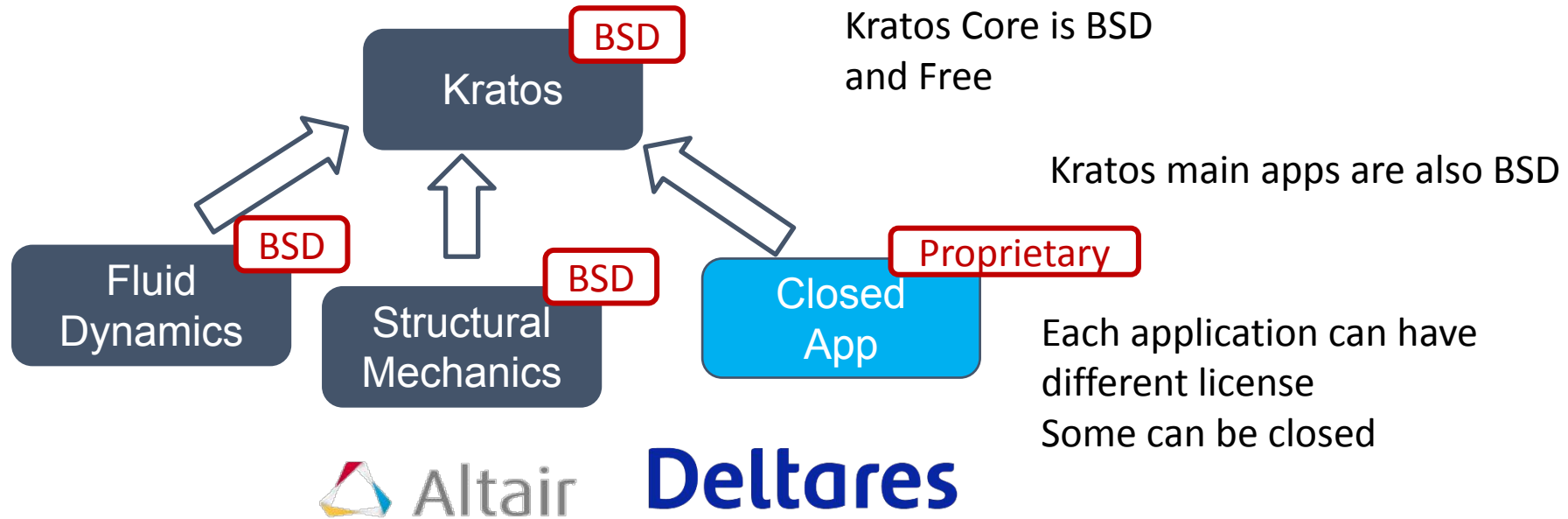


Numerical and programming core

Physics of the problem

Applications can depend on other applications

# Kratos Framework Flexible License

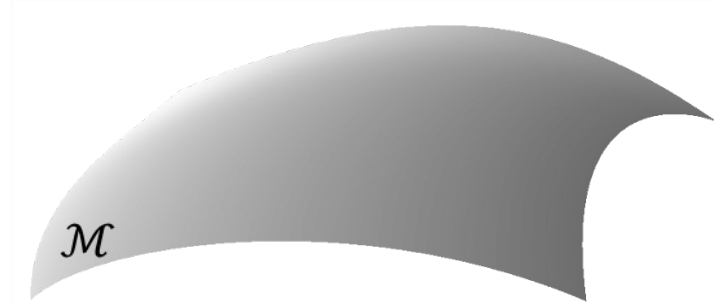


# Proper Orthogonal Decomposition

Full Order Model (FOM)

$$A u = b$$

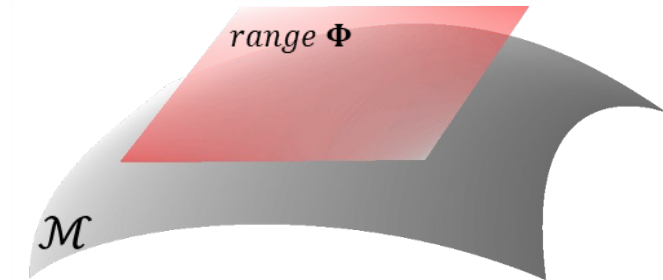
Solution manifold:  $\mathcal{M} = \{ \mathbf{u}(t; \boldsymbol{\mu}) \mid t \in (0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^n$



Let  $u \approx \Phi q$

Reduced Order Model (ROM)

$$\Phi^T A \Phi q = \Phi^T b$$

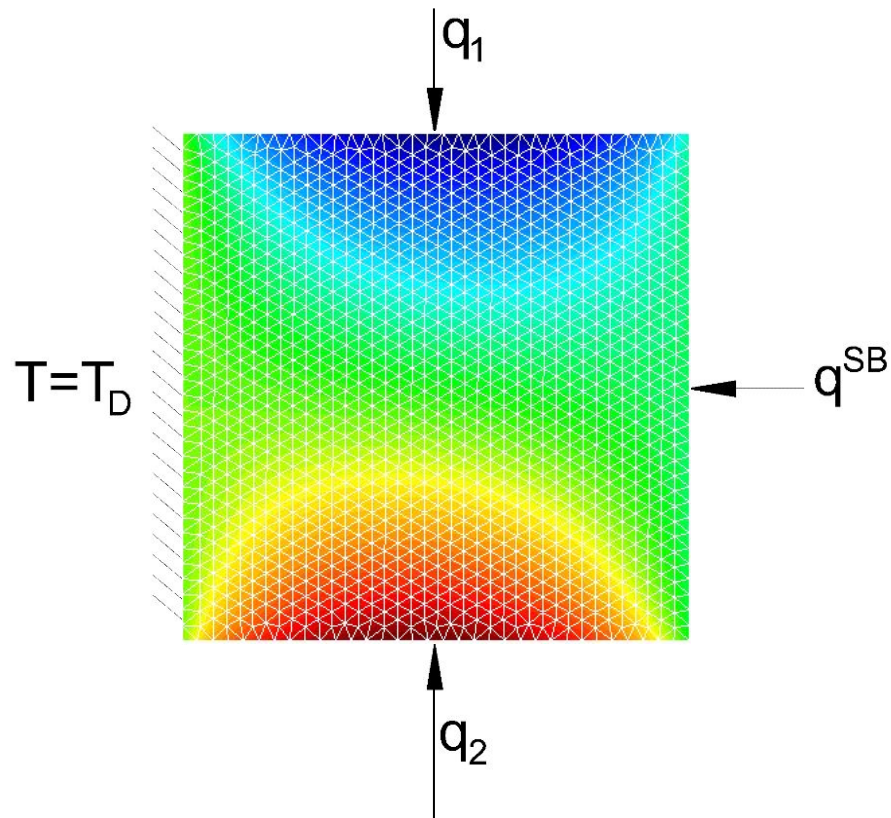


A MUCH SMALLER SYSTEM!  $A^* q = b^*$



# Proper Orthogonal Decomposition

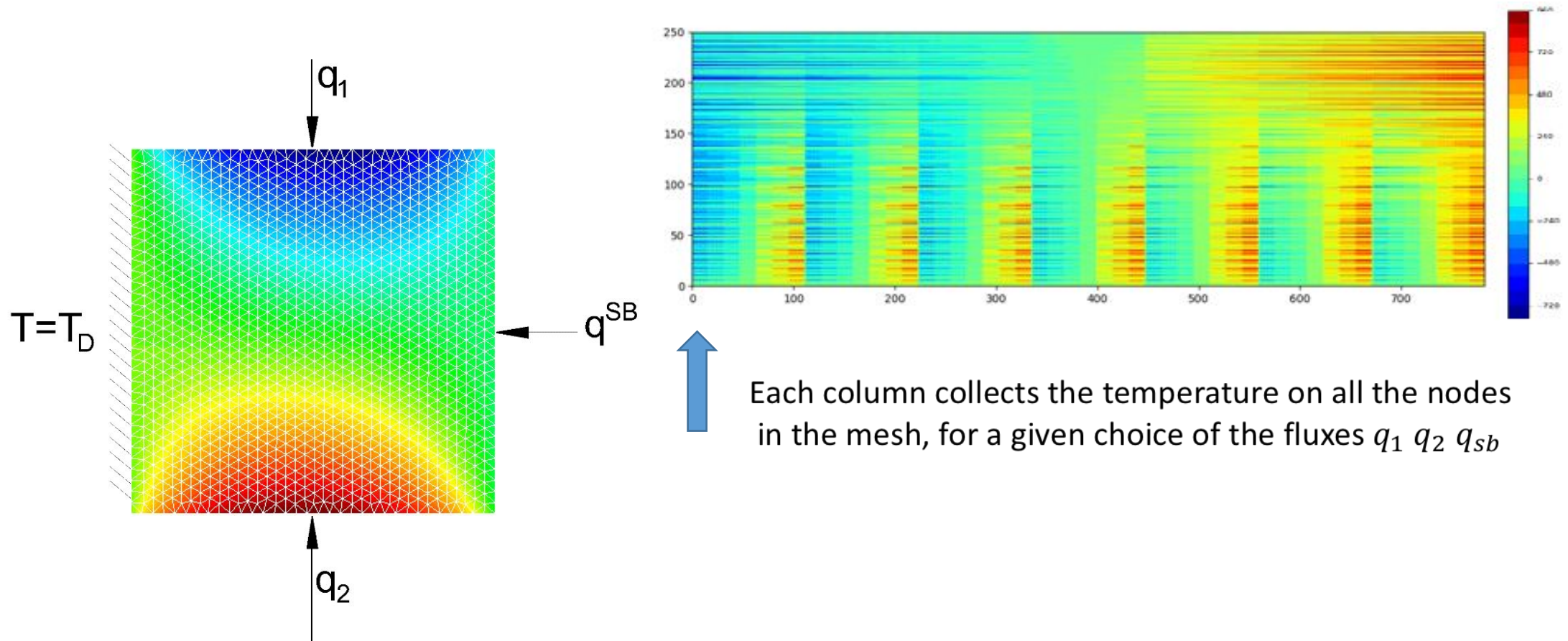
Solve the FOM using Finite Elements



$$\mu = (T_D, q_1, q_2, q_{SB}) \in \mathcal{P}$$

# Proper Orthogonal Decomposition

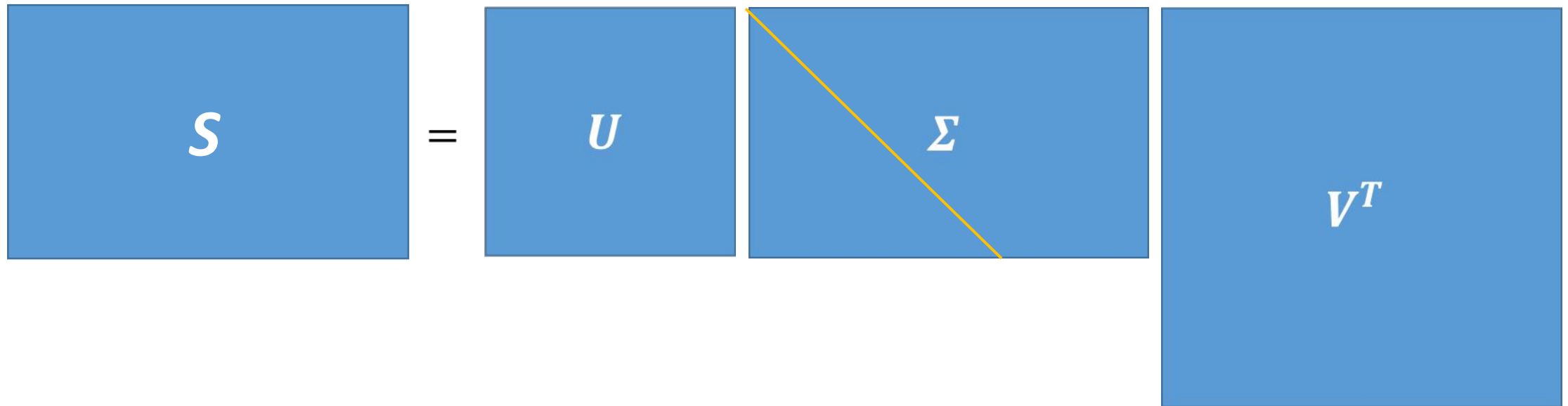
Store the nodal solution  $\mathbf{u}$  the Snapshots matrix  $\mathbf{S} = [ \mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_p ]$



Each column collects the temperature on all the nodes in the mesh, for a given choice of the fluxes  $q_1$   $q_2$   $q_{sb}$

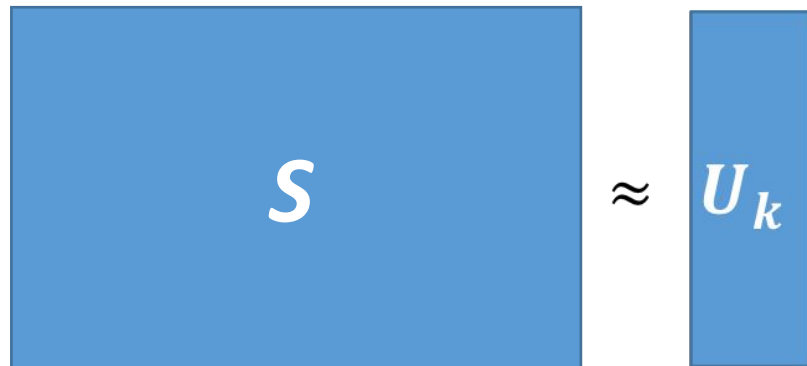
# Proper Orthogonal Decomposition

- Take the SVD of  $S = U\Sigma V^T \approx U_k \Sigma_k V_k^T$



# Proper Orthogonal Decomposition

- Take the SVD of  $S = U\Sigma V^T \approx U_k \Sigma_k V_k^T$

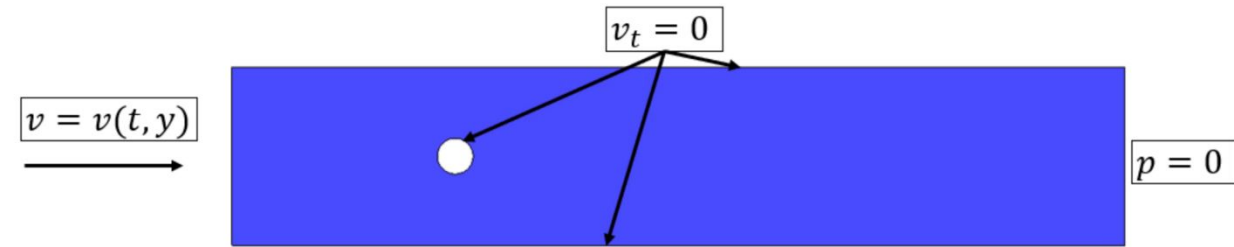


# Proper Orthogonal Decomposition

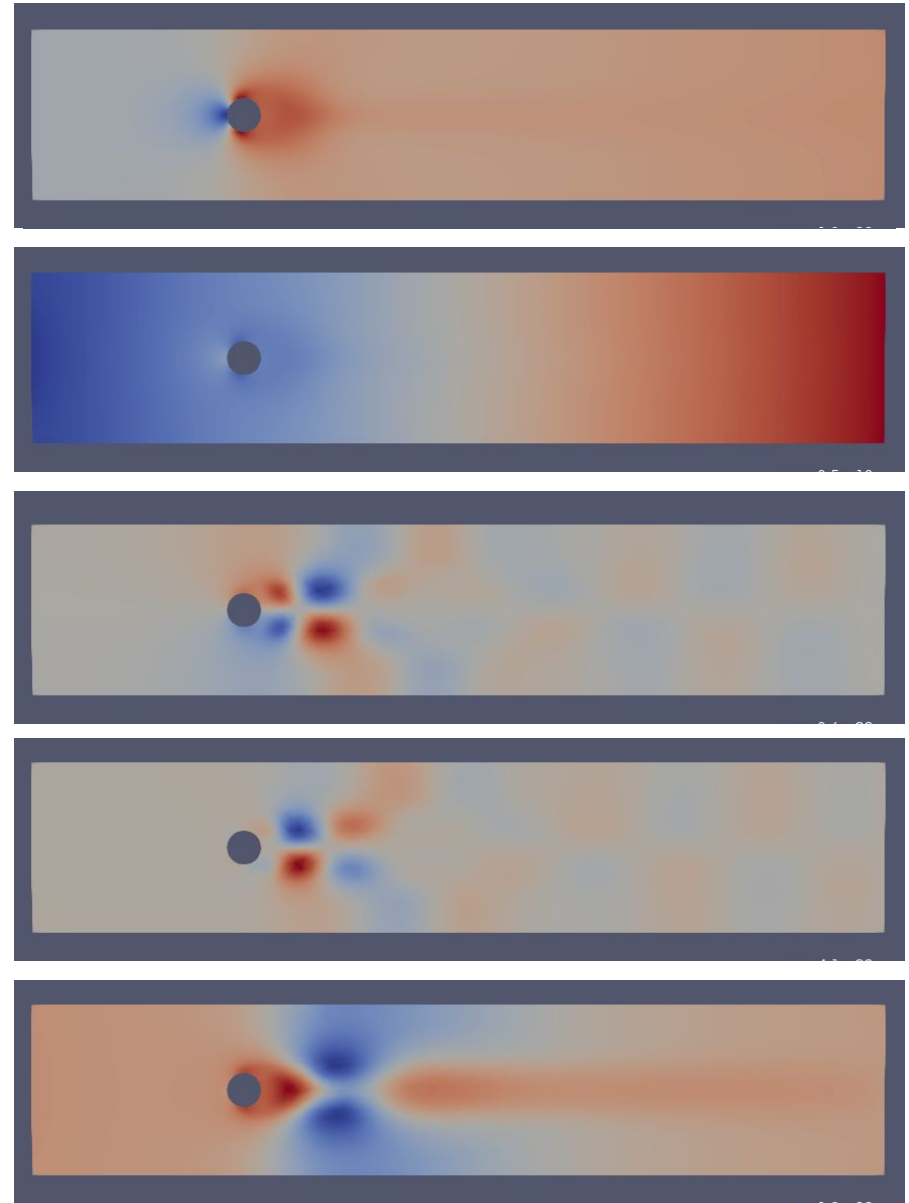
- Take the SVD of  $S = U\Sigma V^T \approx U_k \Sigma_k V_k^T$

$$\Phi := U_k$$

# Example in CFD



$\Phi =$

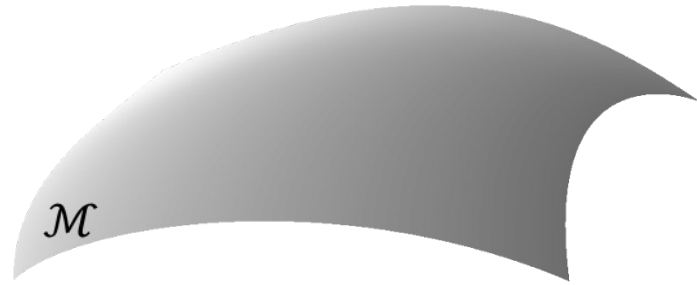


# Proper Orthogonal Decomposition

Full Order Model (FOM)

$$A u = b$$

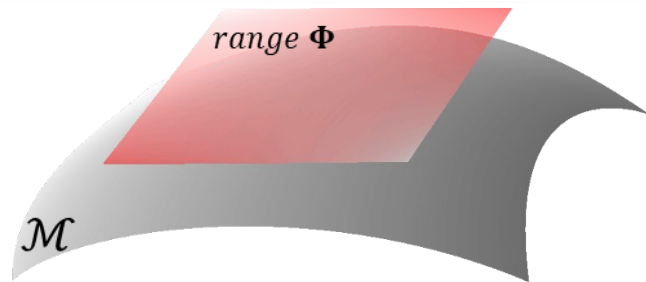
Solution manifold:  $\mathcal{M} = \{ \mathbf{u}(t; \boldsymbol{\mu}) \mid t \in (0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^n$



Let  $u \approx \Phi q$

Reduced Order Model (ROM)

$$\Phi^T A \Phi q = \Phi^T b$$



A MUCH SMALLER SYSTEM!  $A^* q = b^*$

**PROBLEM: STILL EXPENSIVE TO MOUNT THE SYSTEM**

# Hyper-reduction

In general, a second reduction layer is required. The goal is to find a subset of elements and corresponding weights by solving an optimization problem [1].

$$\begin{aligned} (\mathbf{E}, \mathbf{W}) &= \arg \min \| J(\mathbf{R}, \Phi) \|_2^2 \\ &\text{subject to } \mathbf{W} > \mathbf{0} \end{aligned}$$

NP-HARD. Solving via greedy procedure

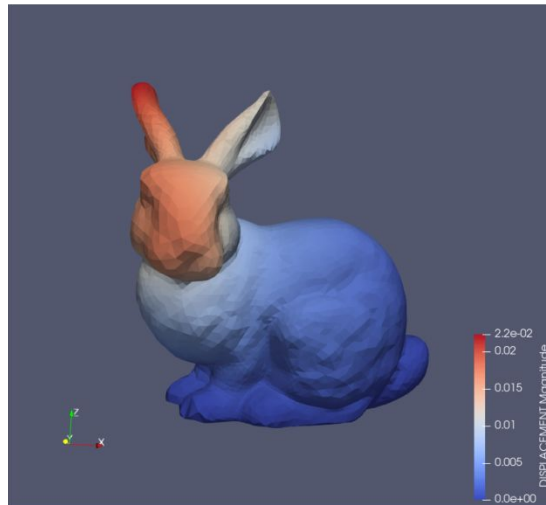


# Hyper-reduction

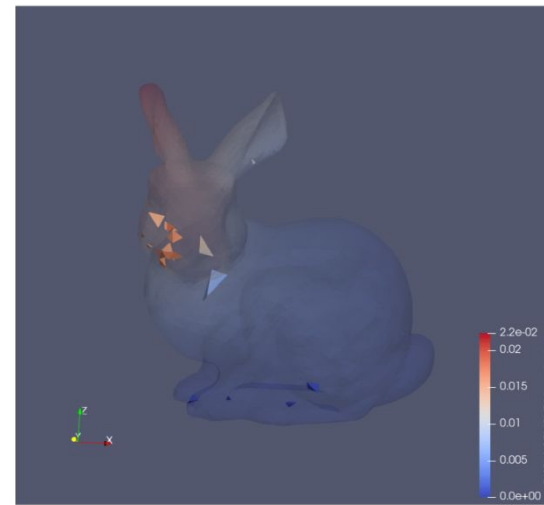
Assembly comparison ROM vs HROM:

$$\left( \prod_{e=1}^{n \text{ elem}} A_e \right) \mathbf{u} = \prod_{e=1}^{n \text{ elem}} \mathbf{b}_e \quad \longrightarrow \quad \left( \sum_{e \in E} \Phi_e^T A_e \Phi_e \omega_e \right) \mathbf{q} = \sum_{e \in E} \Phi_e^T \mathbf{b}_e \omega_e$$

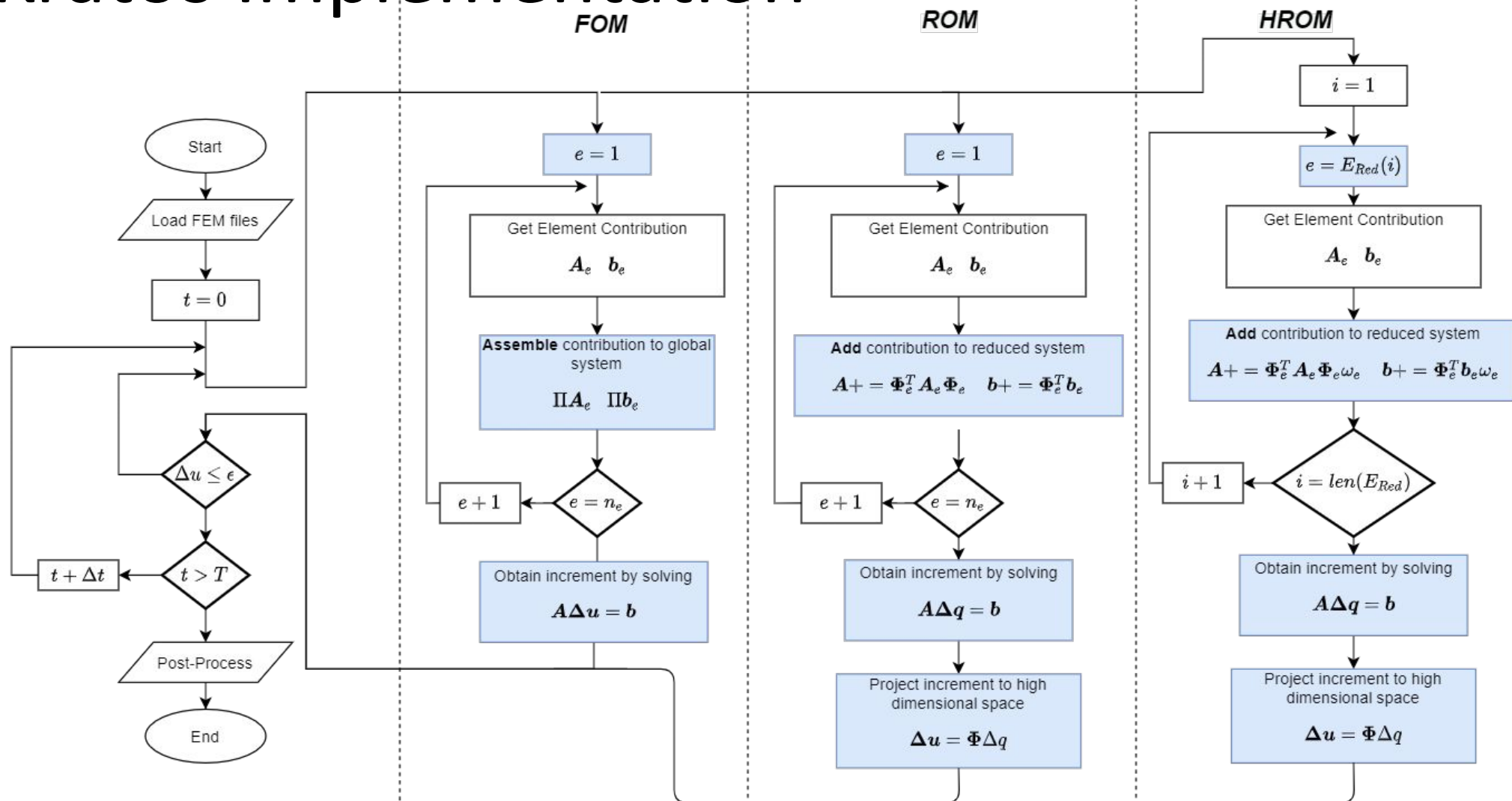
FOM Simulation



HROM Simulation

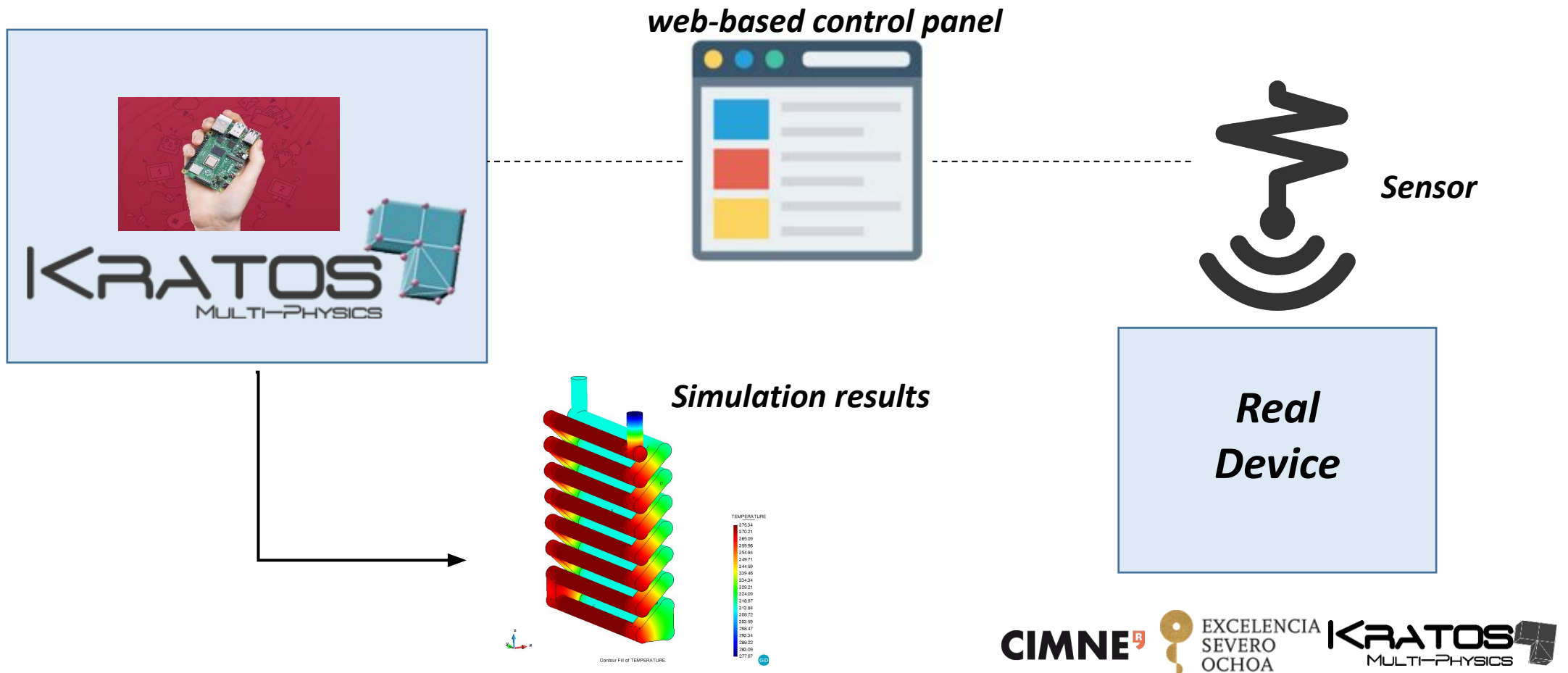


# Kratos Implementation

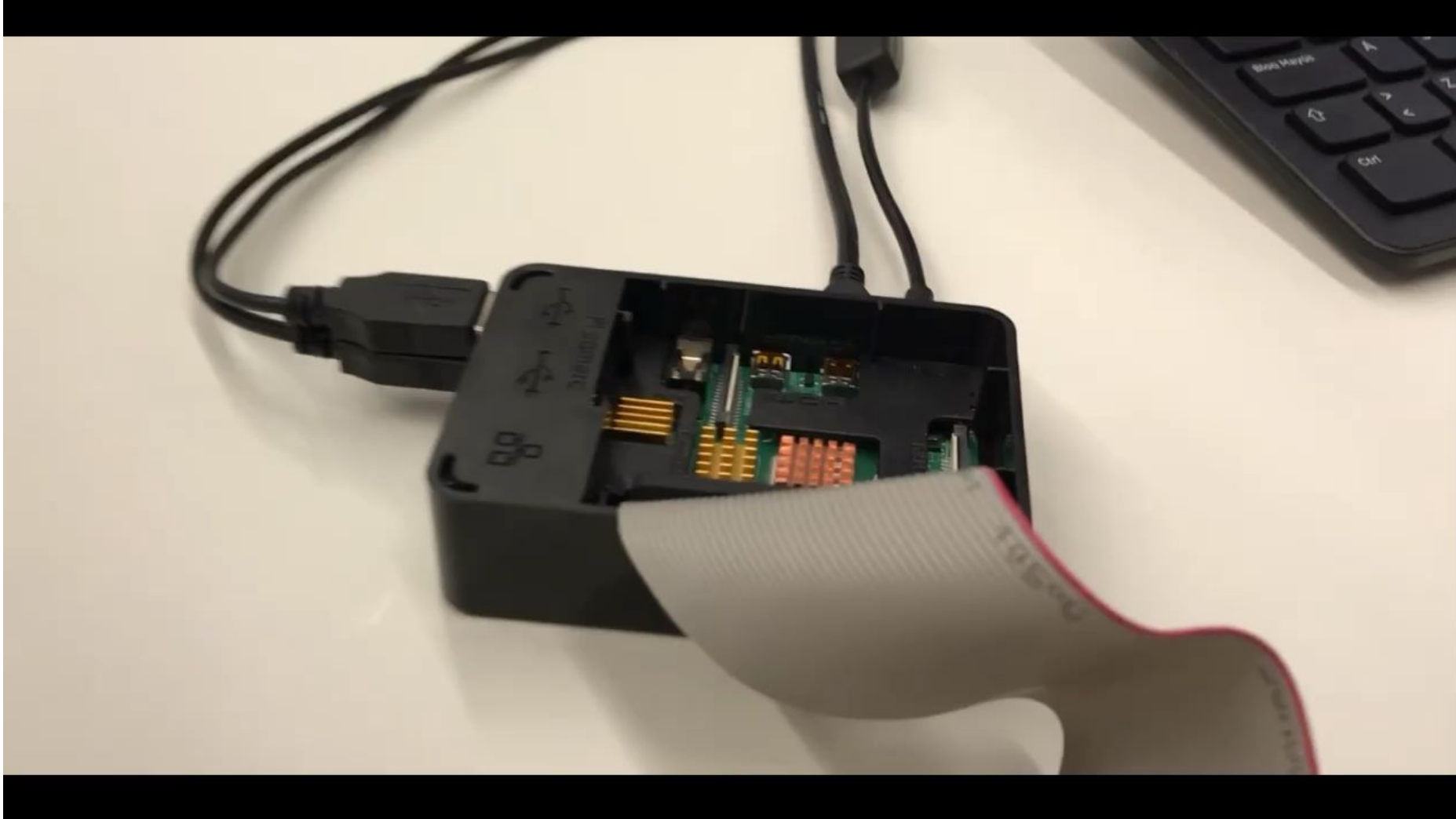


# Link to the external world

Kratos provides an interface to retrieve data from sensors placed in situ.



# Kratos ROM on a Raspberry Pi



# POD weaknesses and strengths

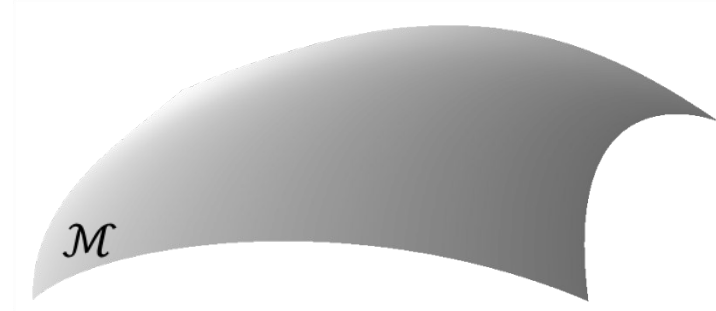
- Straightforward procedure for training and inference
- Not ideal for certain problems (convection dominated, highly nonlinear)

# Local POD

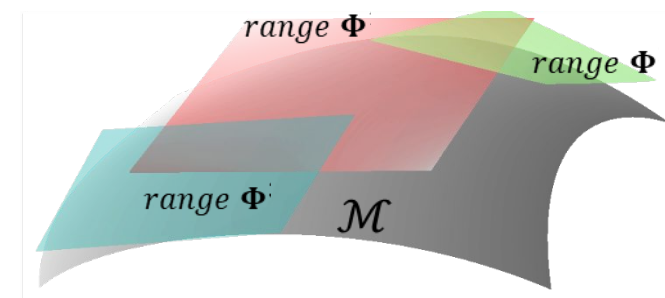
Full Order Model (FOM)

$$\mathbf{A} \mathbf{u} = \mathbf{b}$$

Solution manifold:  $\mathcal{M} = \{ \mathbf{u}(t; \boldsymbol{\mu}) \mid t \in (0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^n$



$$\text{Let } \mathbf{u}_{new} \approx \mathbf{u}_{old} + \Phi^i \mathbf{q}$$

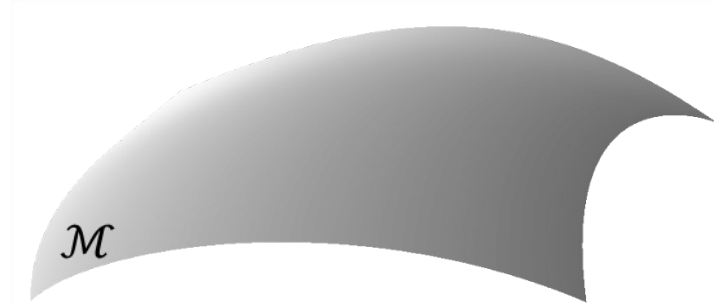


# Local POD

Full Order Model (FOM)

$$A u = b$$

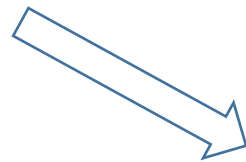
Solution manifold:  $\mathcal{M} = \{ \mathbf{u}(t; \boldsymbol{\mu}) \mid t \in (0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^n$



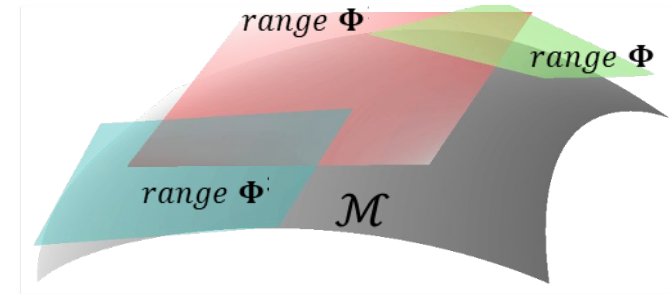
$$\text{Let } \mathbf{u}_{new} \approx \mathbf{u}_{old} + \Phi^i \mathbf{q}$$

Reduced Order Model (ROM)

$$\Phi^{2T} A \Phi^2 \mathbf{q} = \Phi^{2T} \mathbf{b}$$



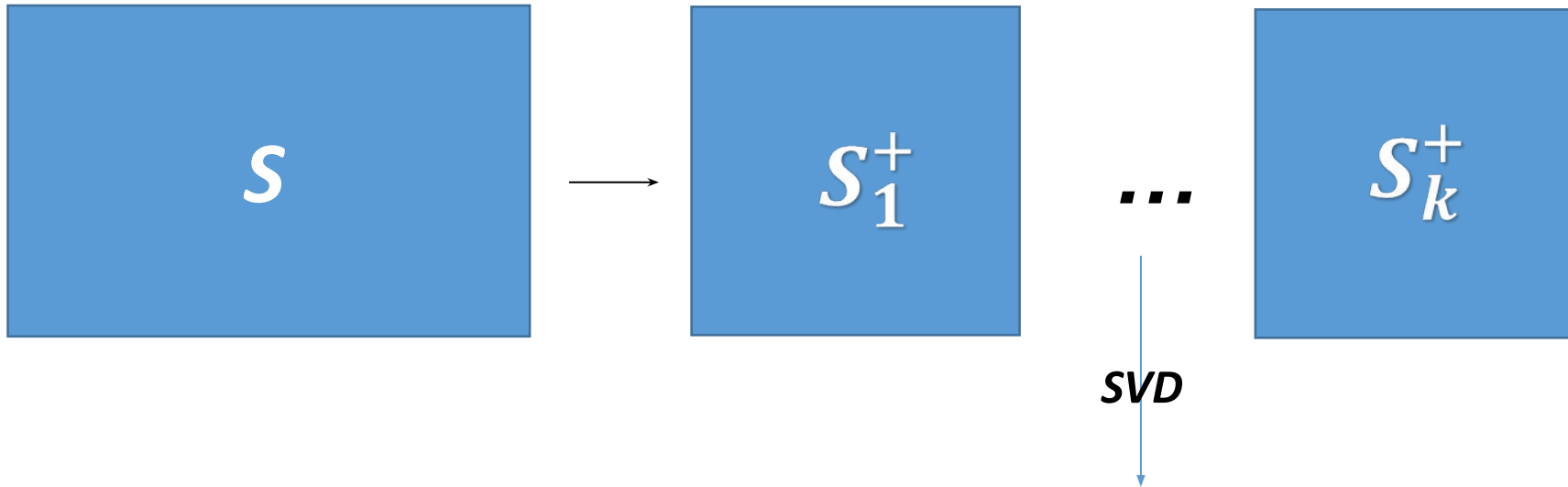
$$A^* \mathbf{q} = \mathbf{b}^*$$



# Local POD

Use an unsupervised learning method to build clusters

- 1.  $S_i = kmeans(S)$  2. Add overlapping  $S_i^+ = overlap(S_i)$  . See ref. [2]
- 1.  $S_i^+ = fuzzy-c - means(S_i)$  . See ref. [3]

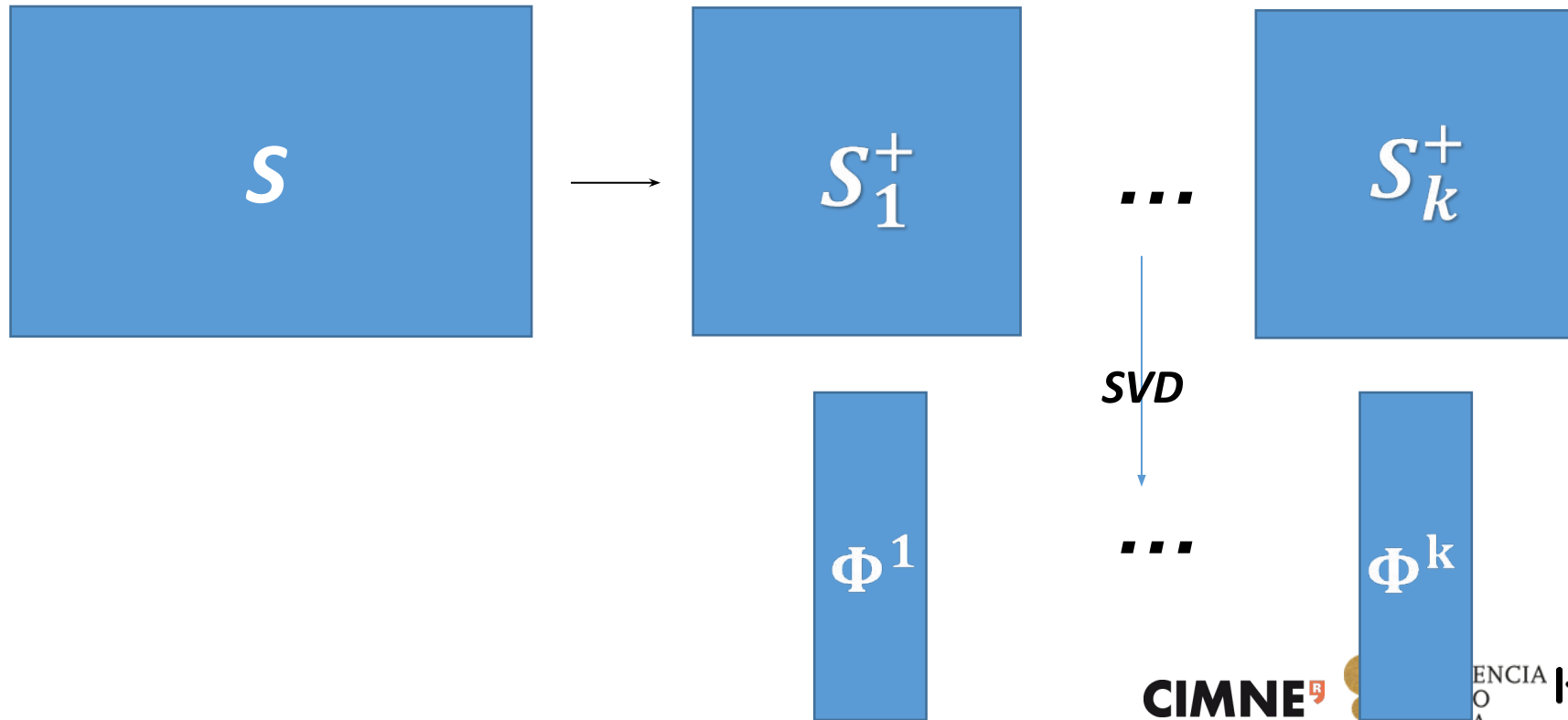




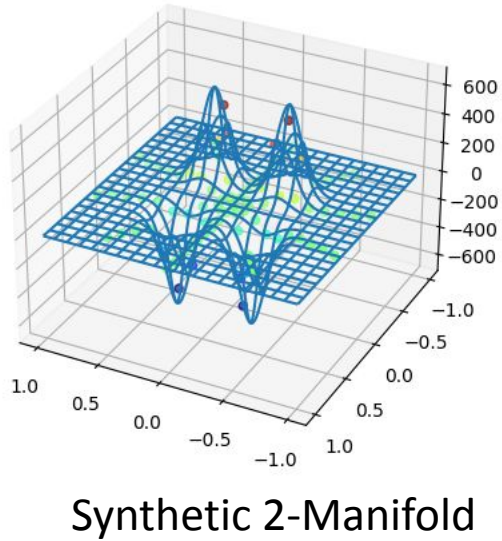
# Local POD

Use an unsupervised learning method to build clusters

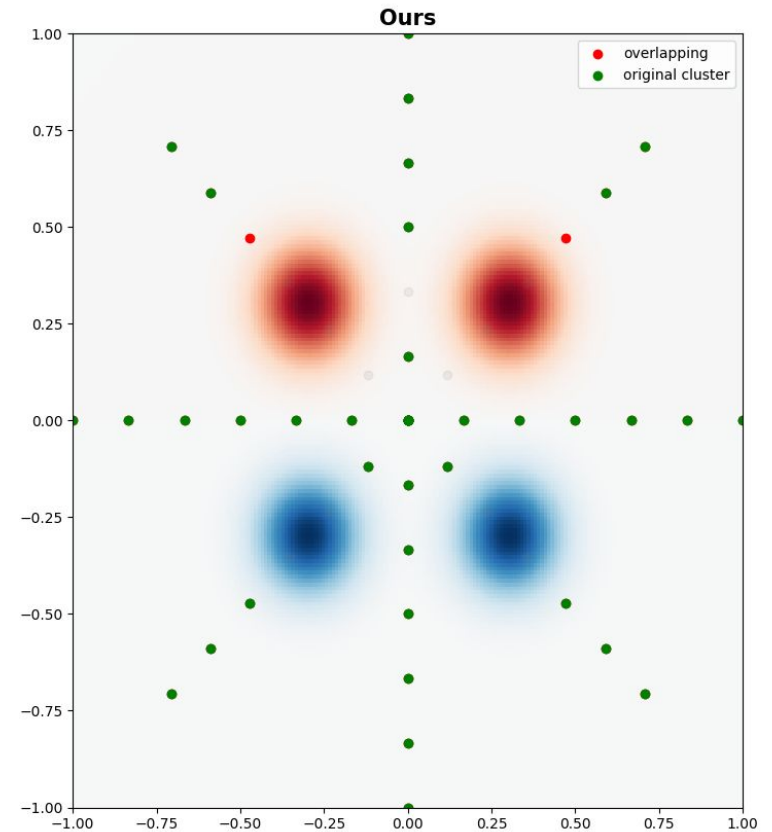
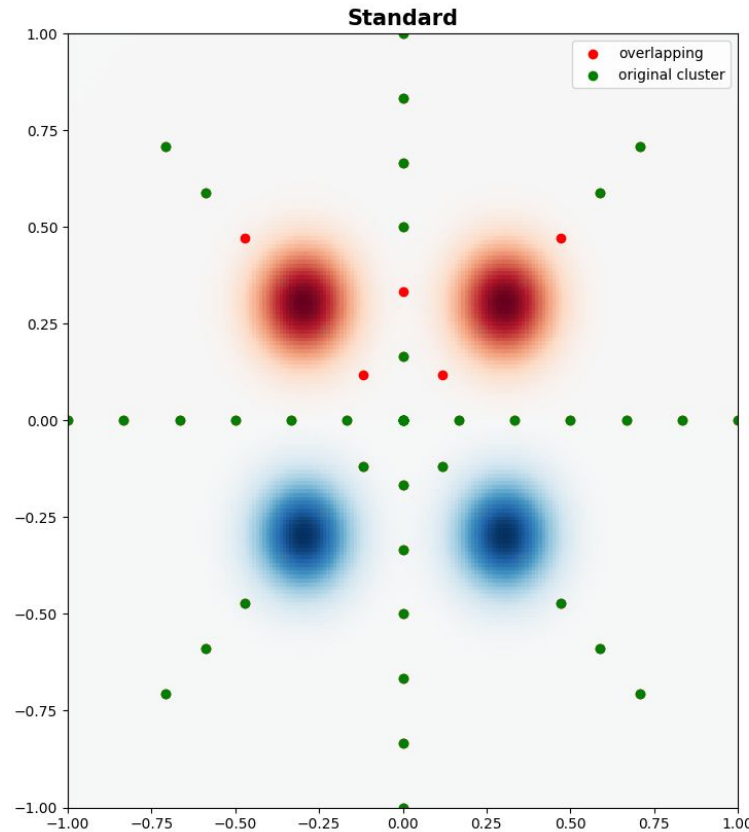
- 1.  $S_i = kmeans(S)$  2. Add overlapping  $S_i^+ = overlap(S_i)$ . See ref. [2]
- 1.  $S_i^+ = fuzzy-c - means(S_i)$ . See ref. [3]



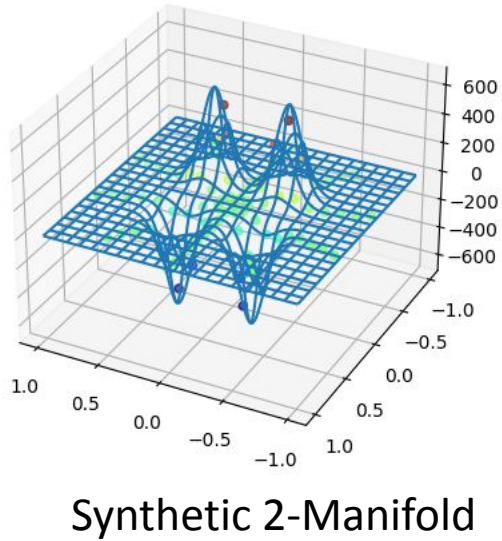
# Local POD. The importance of overlapping



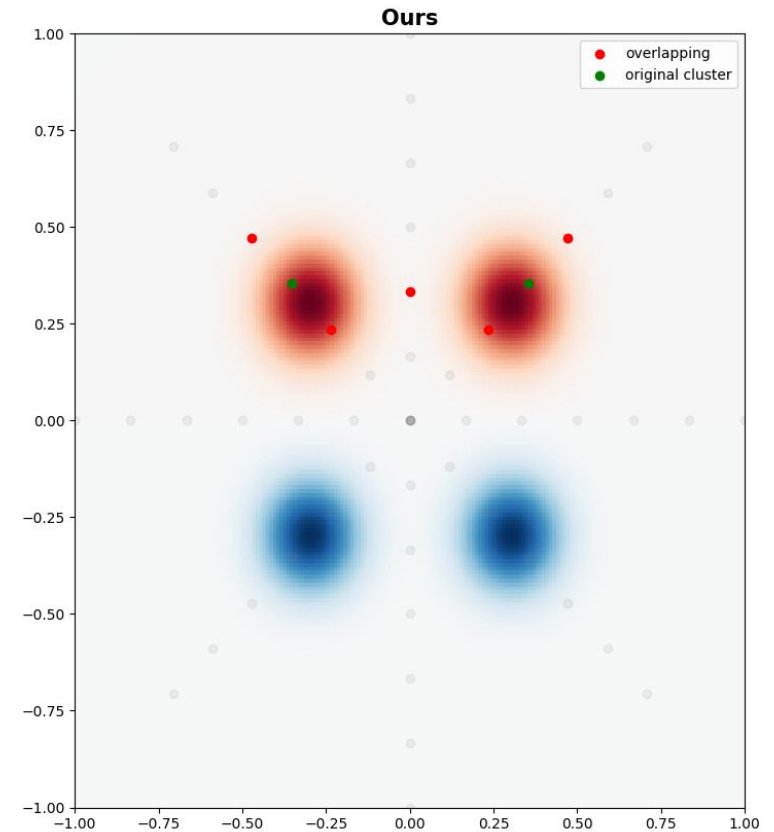
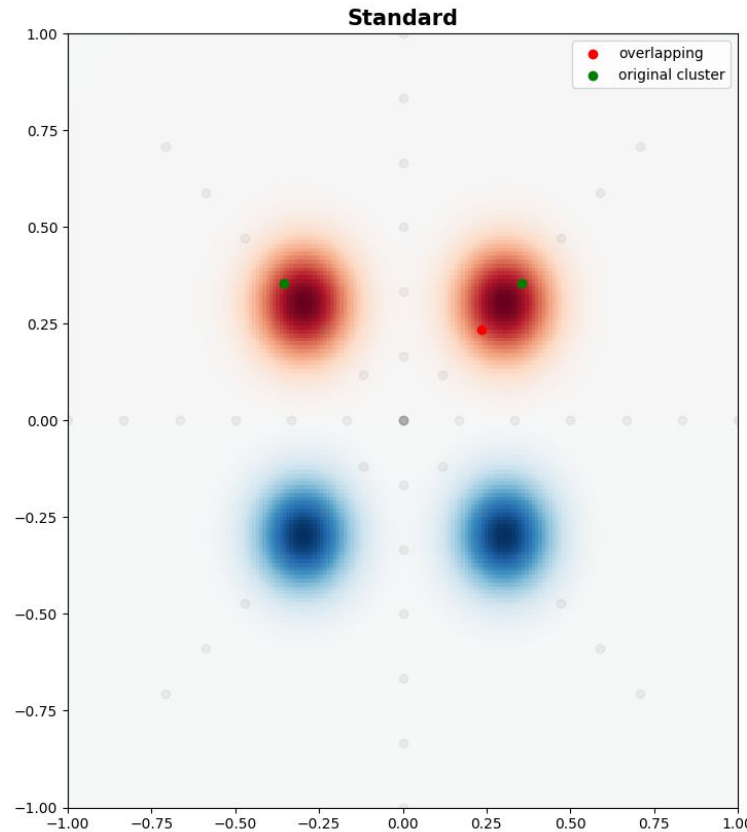
Cluster 0



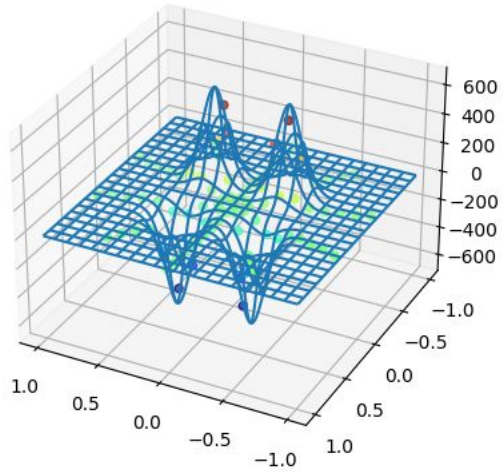
# Local POD. The importance of overlapping



Cluster 1

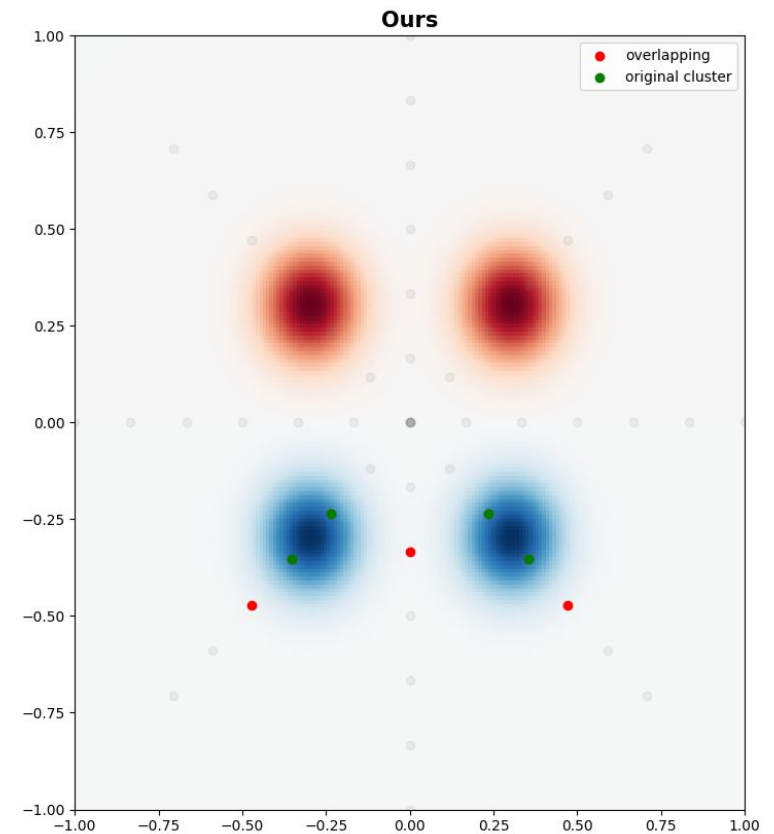
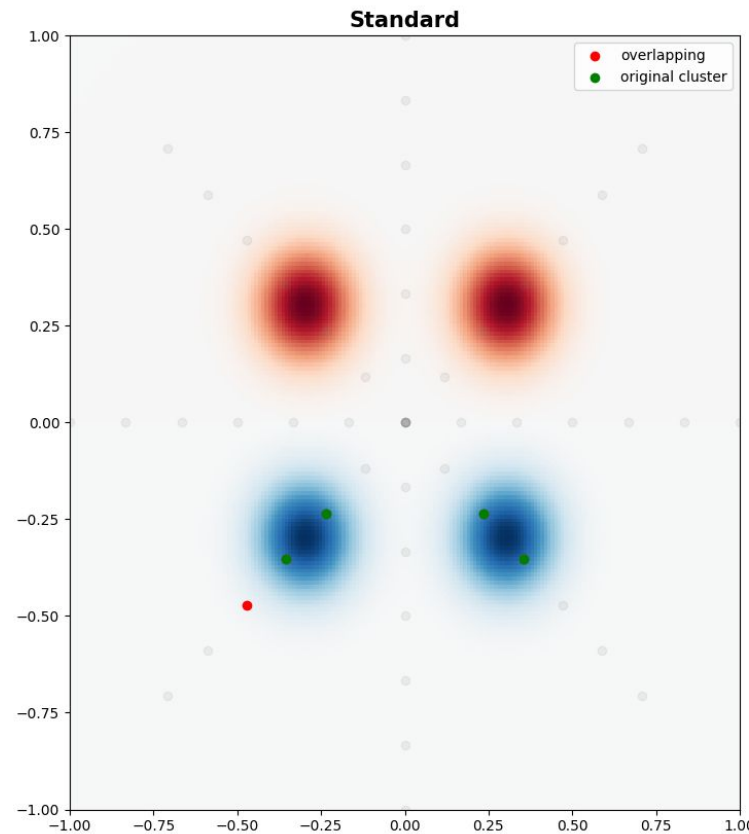


# Local POD. The importance of overlapping

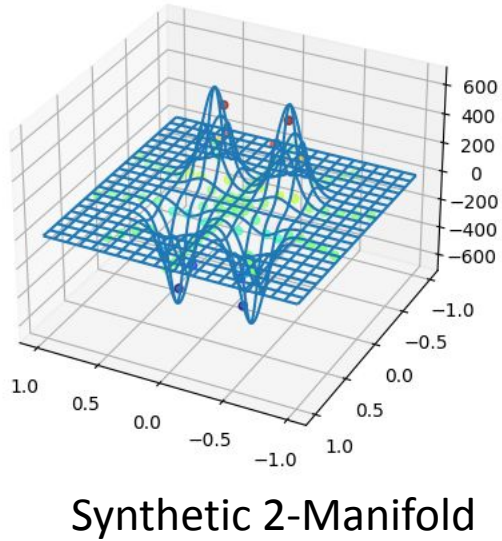


Synthetic 2-Manifold

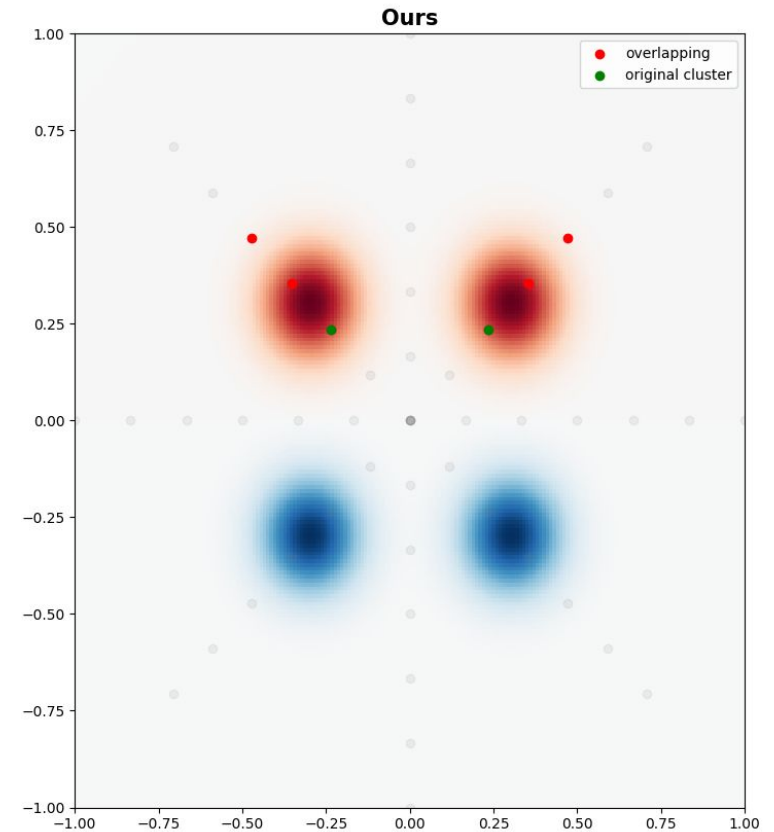
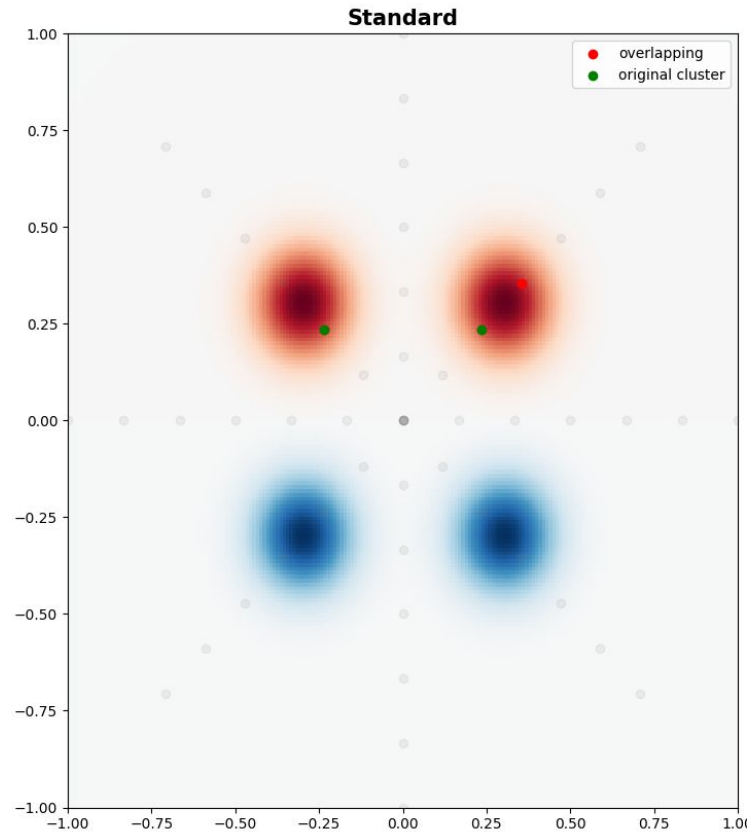
Cluster 2



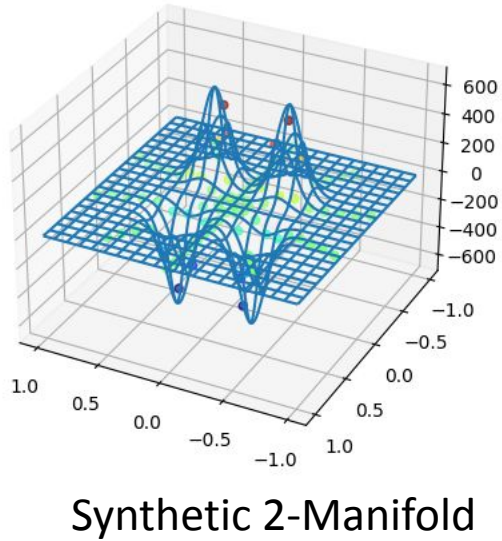
# Local POD. The importance of overlapping



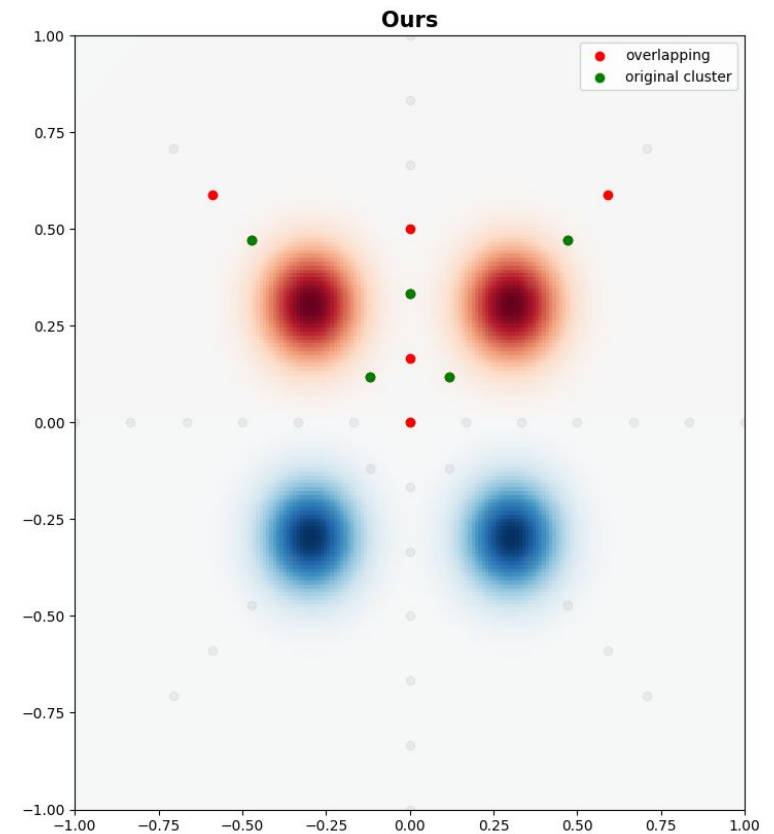
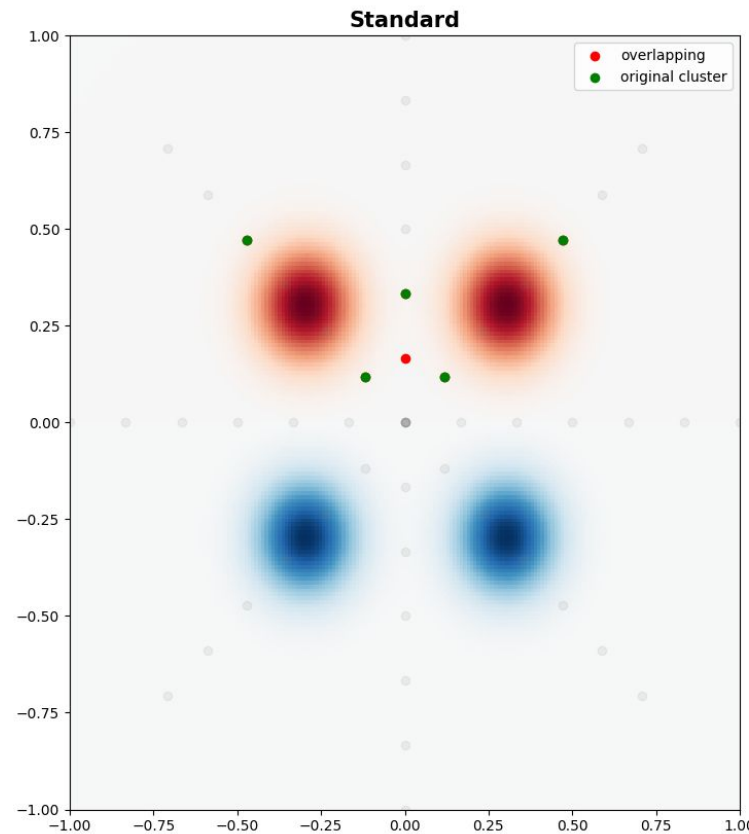
Cluster 3



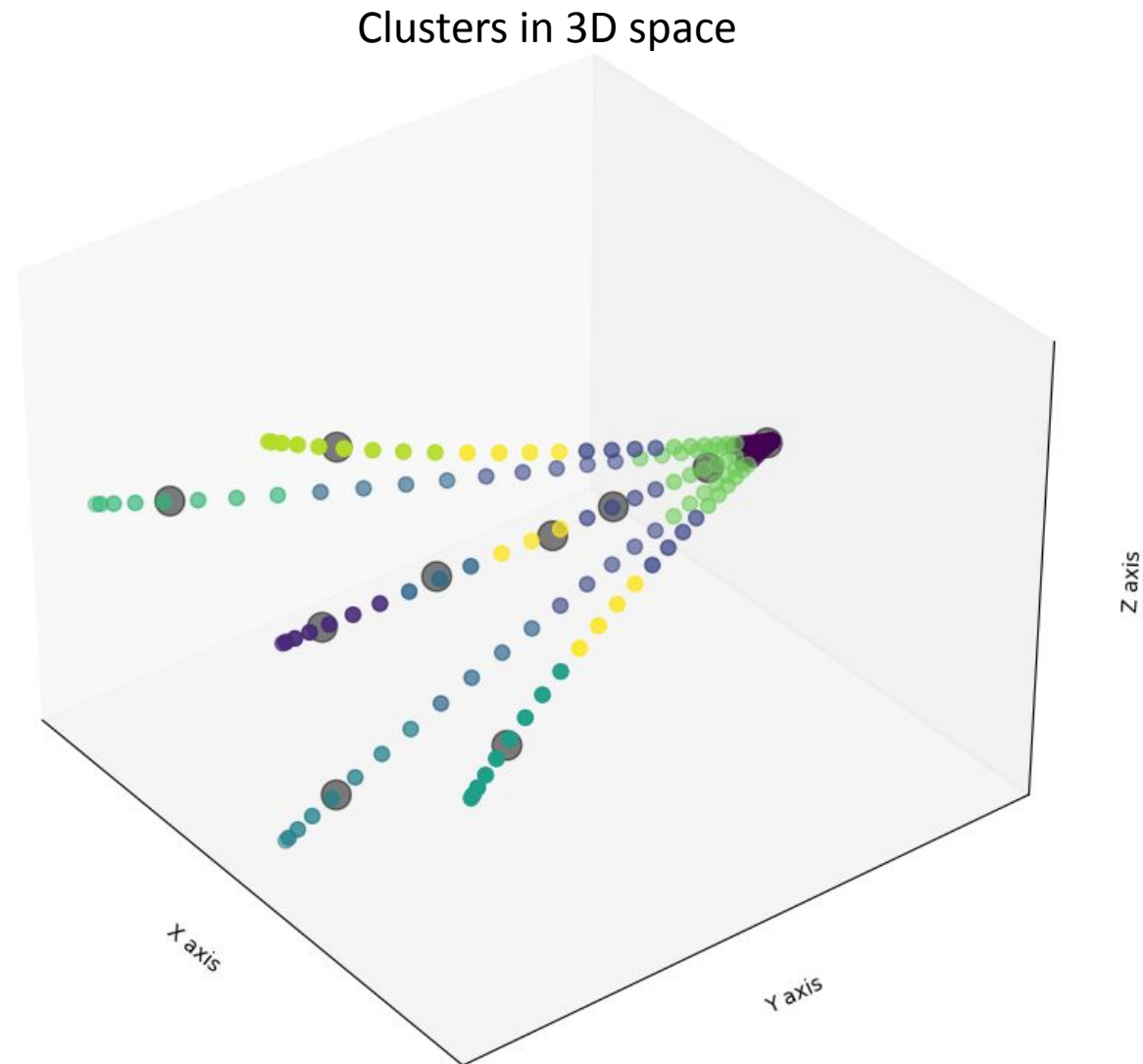
# Local POD. The importance of overlapping



Cluster 4

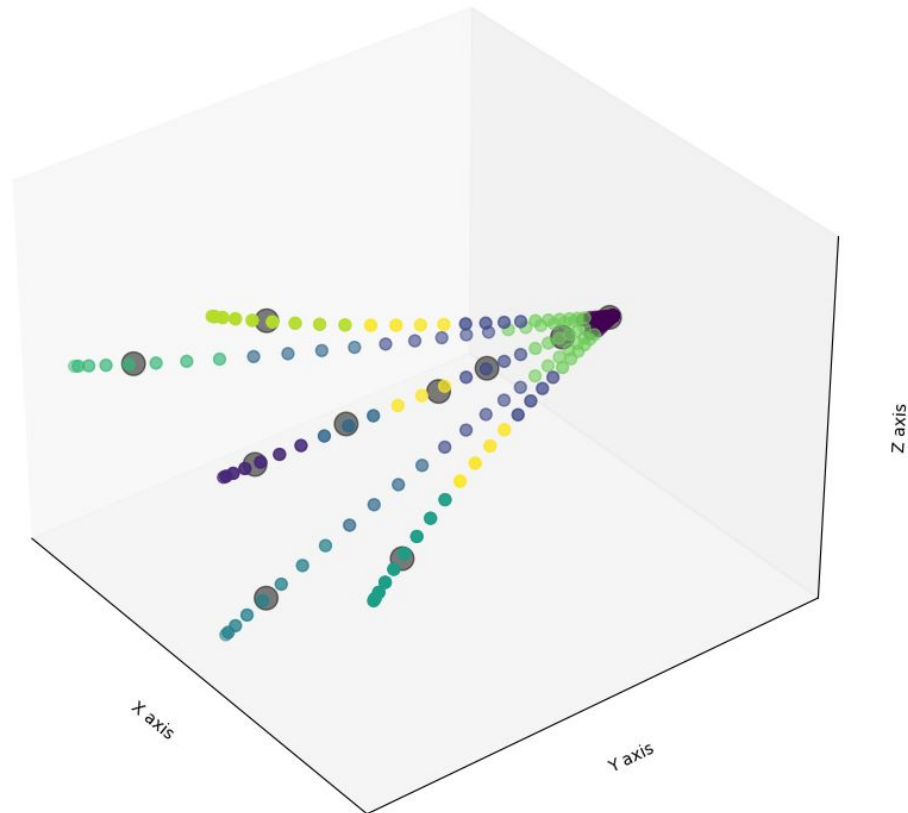


# Local POD. Example

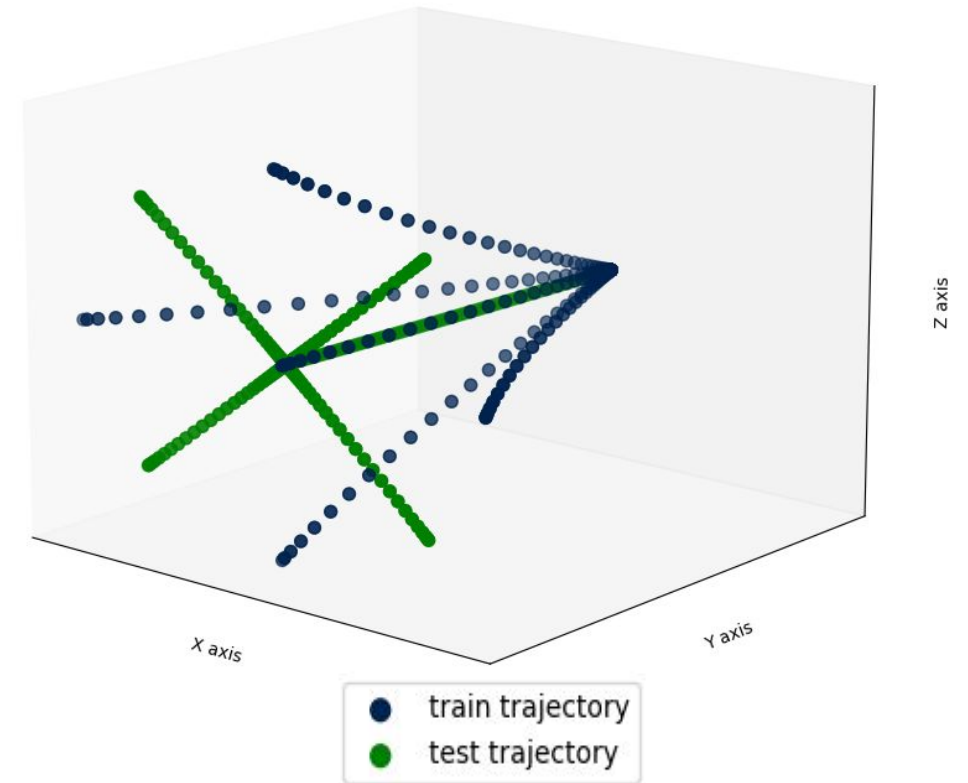


# Local POD. Example

Clusters in 3D space



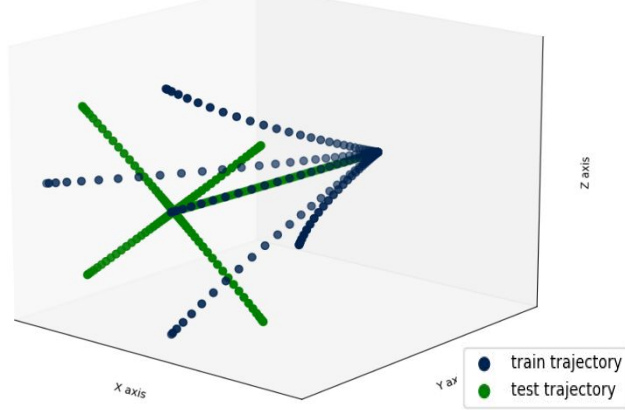
Train and Test trajectories



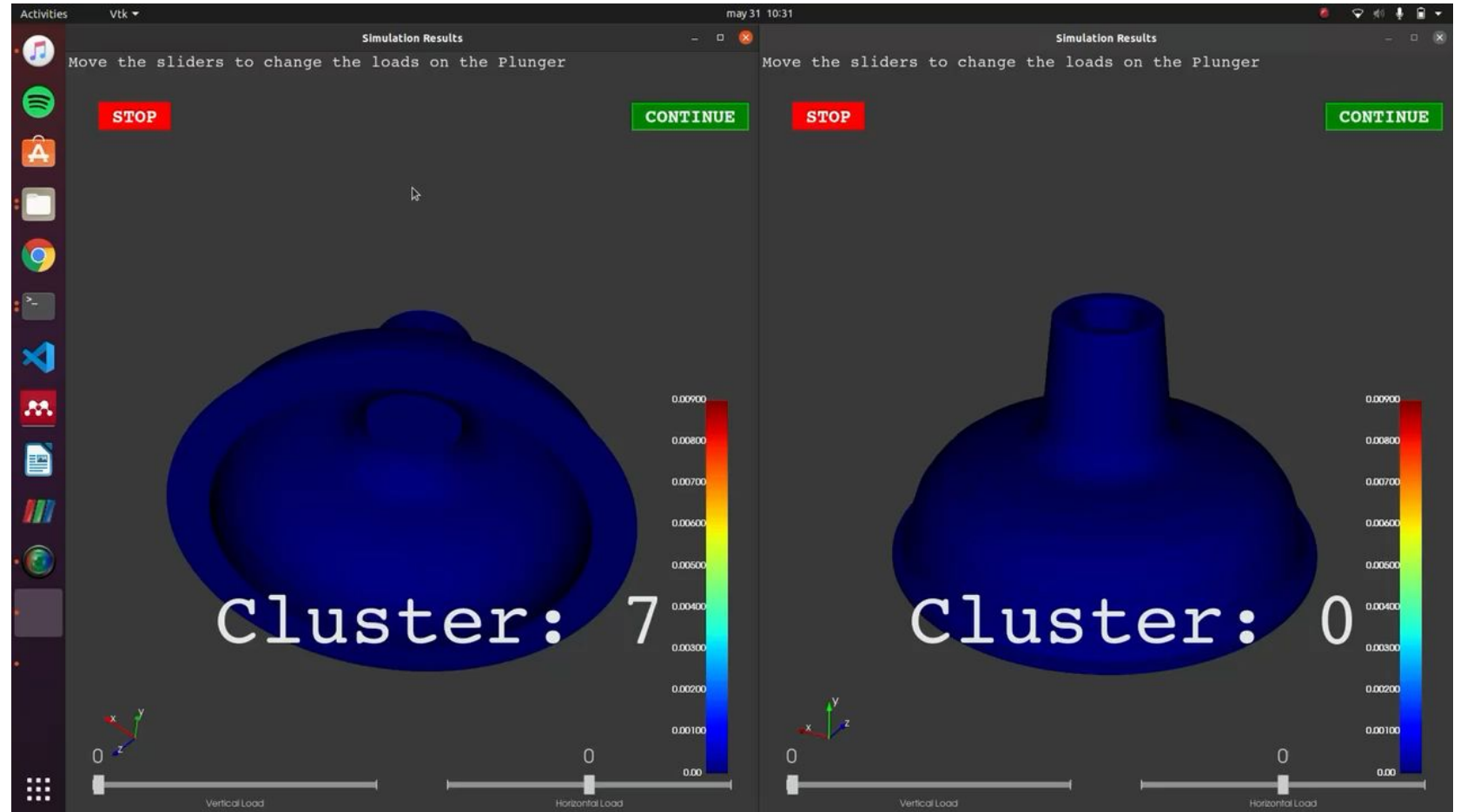
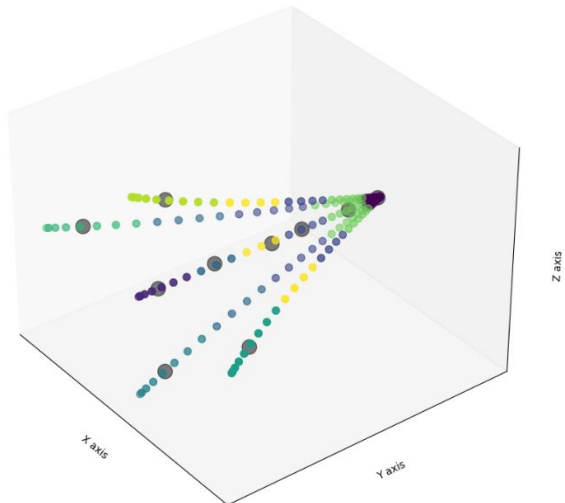


# Local POD. Example

Train and Test trajectories



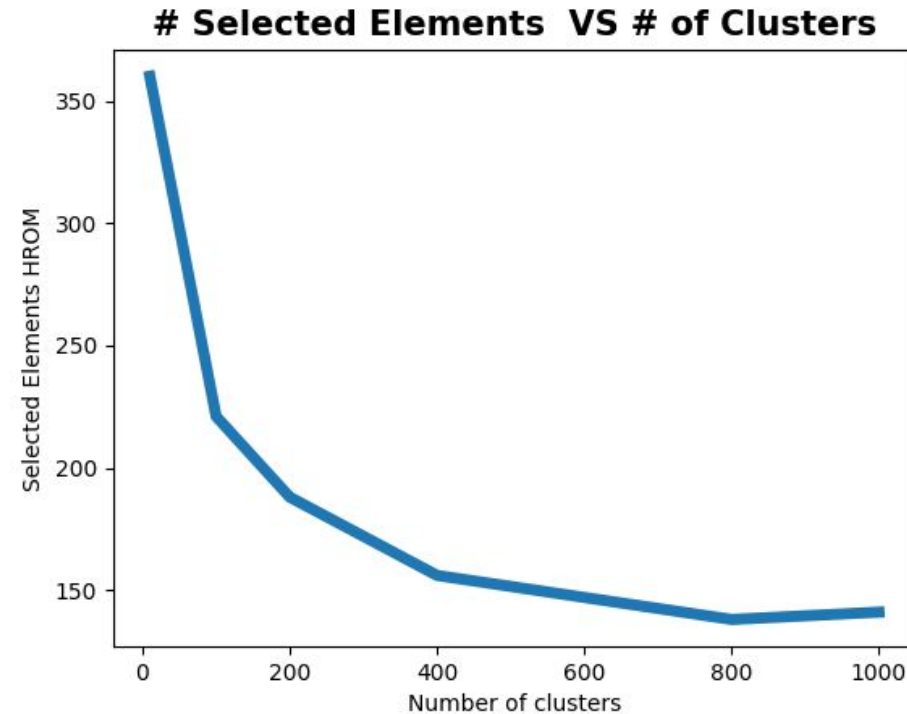
Clusters in 3D space



# Local POD. Improved hyper-reduction

$$(\mathbf{E}, \mathbf{W}) = \arg \min \| J(\mathbf{R}, \Phi) \|_2^2$$

subject to  $\mathbf{W} > \mathbf{0}$



# Local POD. Strengths and weaknesses

- Reasonable overhead in training and negligible in inference
- Smaller elements sets, therefore faster ROMs
  
- Easy to overfit to training trajectories

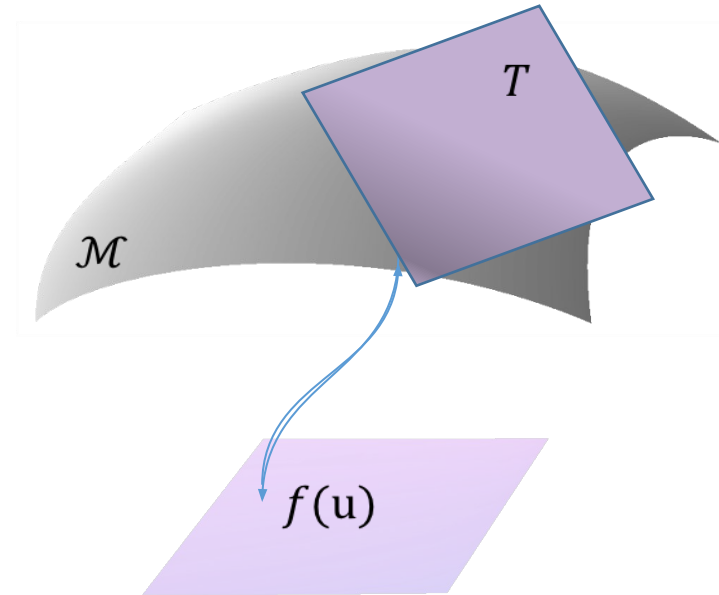
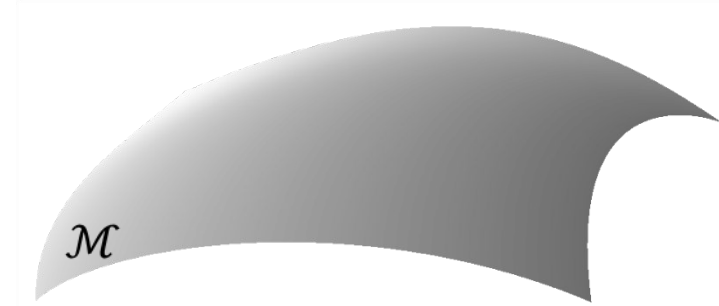
# Deep autoencoders

Full Order Model (FOM)

$$A u = b$$

Solution manifold:  $\mathcal{M} = \{ \mathbf{u}(t; \boldsymbol{\mu}) \mid t \in (0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^n$

$$\text{Let } u \approx g(f(u))$$

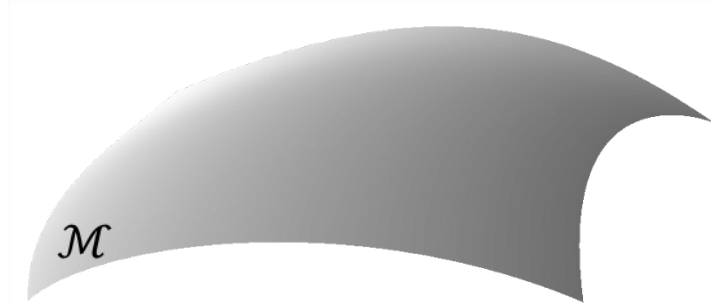


# Deep autoencoders

Full Order Model (FOM)

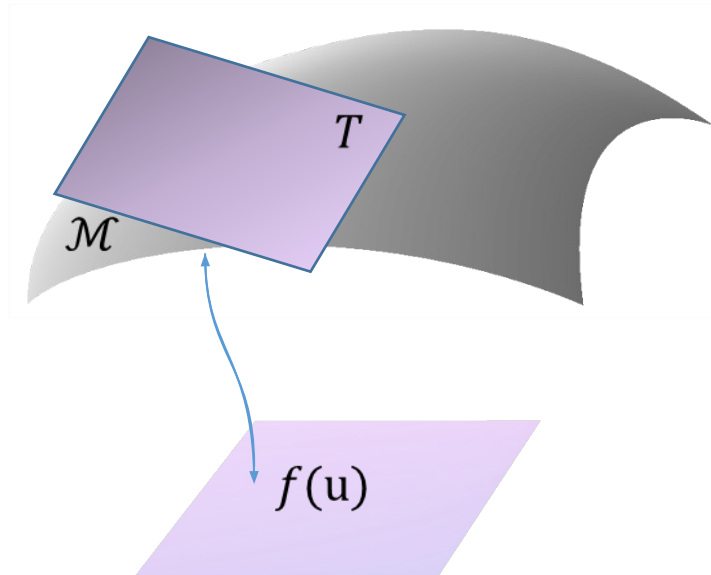
$$A u = b$$

Solution manifold:  $\mathcal{M} = \{ \mathbf{u}(t; \boldsymbol{\mu}) \mid t \in (0, T], \boldsymbol{\mu} \in \mathcal{P} \} \subset \mathbb{R}^n$



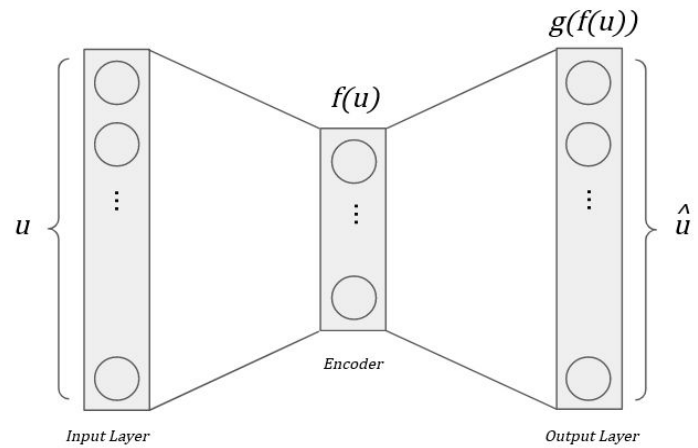
Let  $u \approx g(f(u))$

$$T^T A \hat{u} \begin{pmatrix} f(u) \end{pmatrix} = T^T b$$

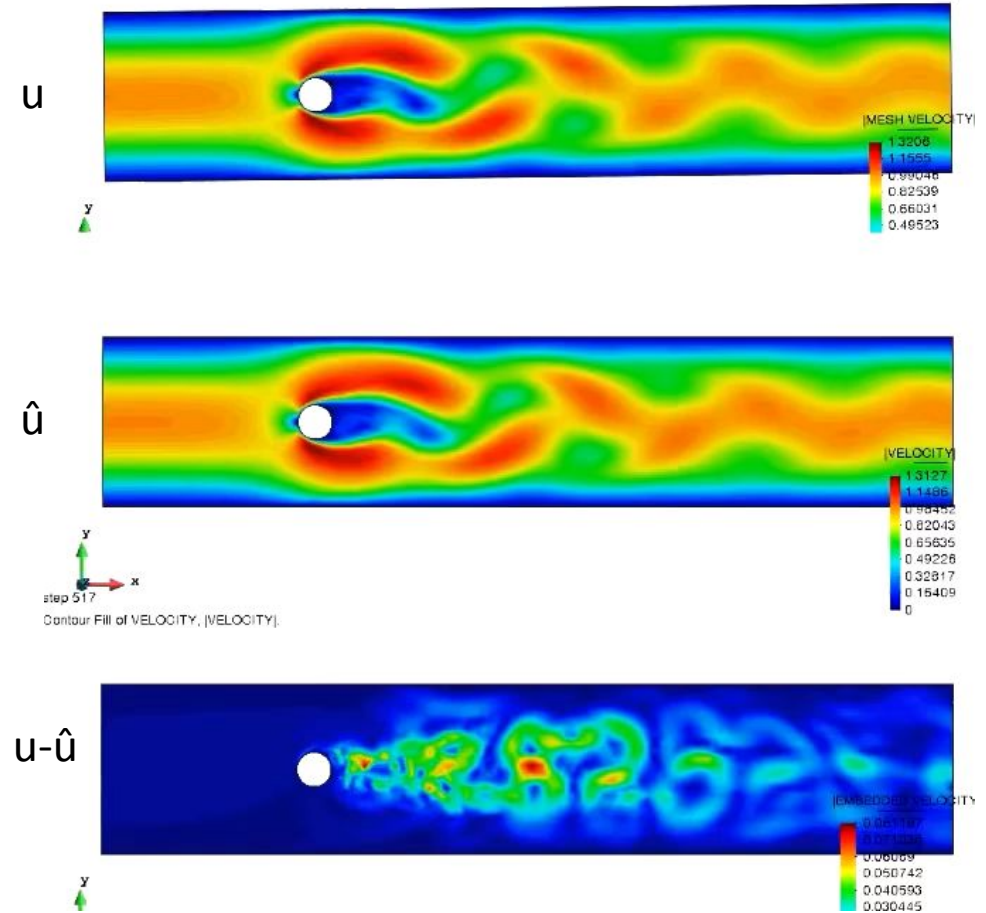


\*be careful, slight oversimplifications

# Deep autoencoders



$$\text{minimize } \|u - \hat{u}\|_2^2$$

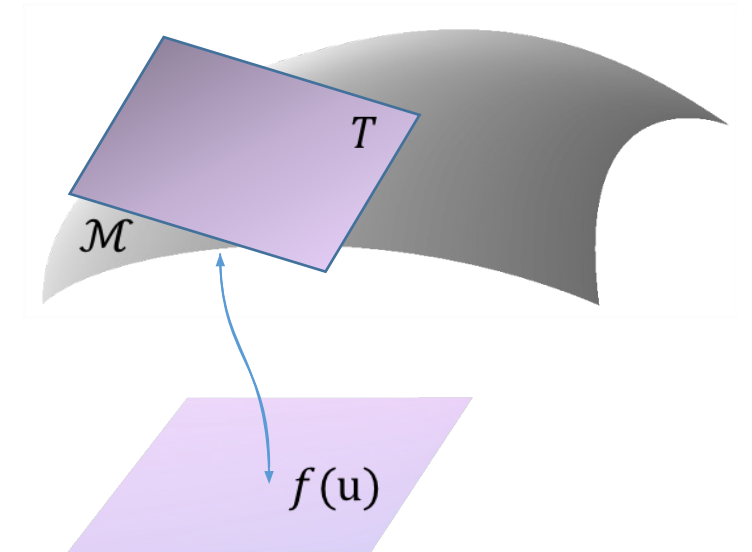
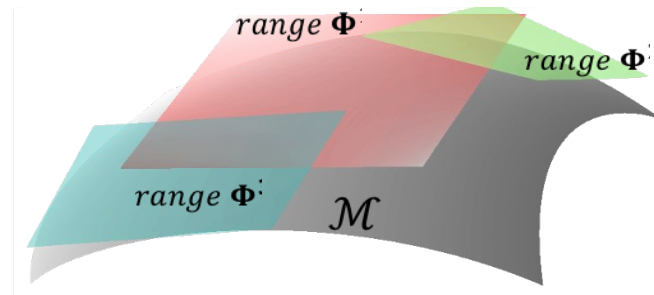
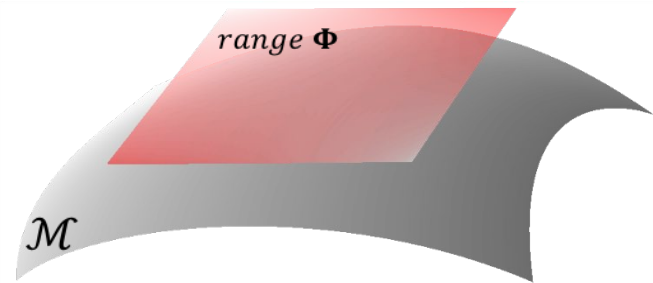


# Deep autoencoders. Some comments

- Not difficult to integrate within Kratos (Python libraries)
- Long training time
- Not much literature on nonstructured meshes FEM

# General conclusions

- The ROM capabilities of Kratos
- Promising results and exciting challenges





# THANK YOU

GRATEFUL TO:



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 946009



Link to Kratos github site

# References:

- [1] Hernández, J. A. (2020). A multiscale method for periodic structures using domain decomposition and ECM-hyperreduction. *Computer Methods in Applied Mechanics and Engineering*, 368, 113192.
- [2]Bezdek J, Ehrlich R, Full W. FCM: the fuzzy c-means clustering algorithm. *Comput Geosci*. 1984;10(2–3):191-203
- [3]Amsallem D, Zahr MJ, Farhat C. Nonlinear model order reduction based on local reduced-order bases. *Int J Numer Methods Eng*. 2012;92(10):891-916