Innovative HPC workflows for industry
Reduced Order Models

Sebastian Ares de Parga – (UPC, CIMNE)

José Raúl Bravo – (UPC, CIMNE)

Riccardo Rossi – (UPC, CIMNE)

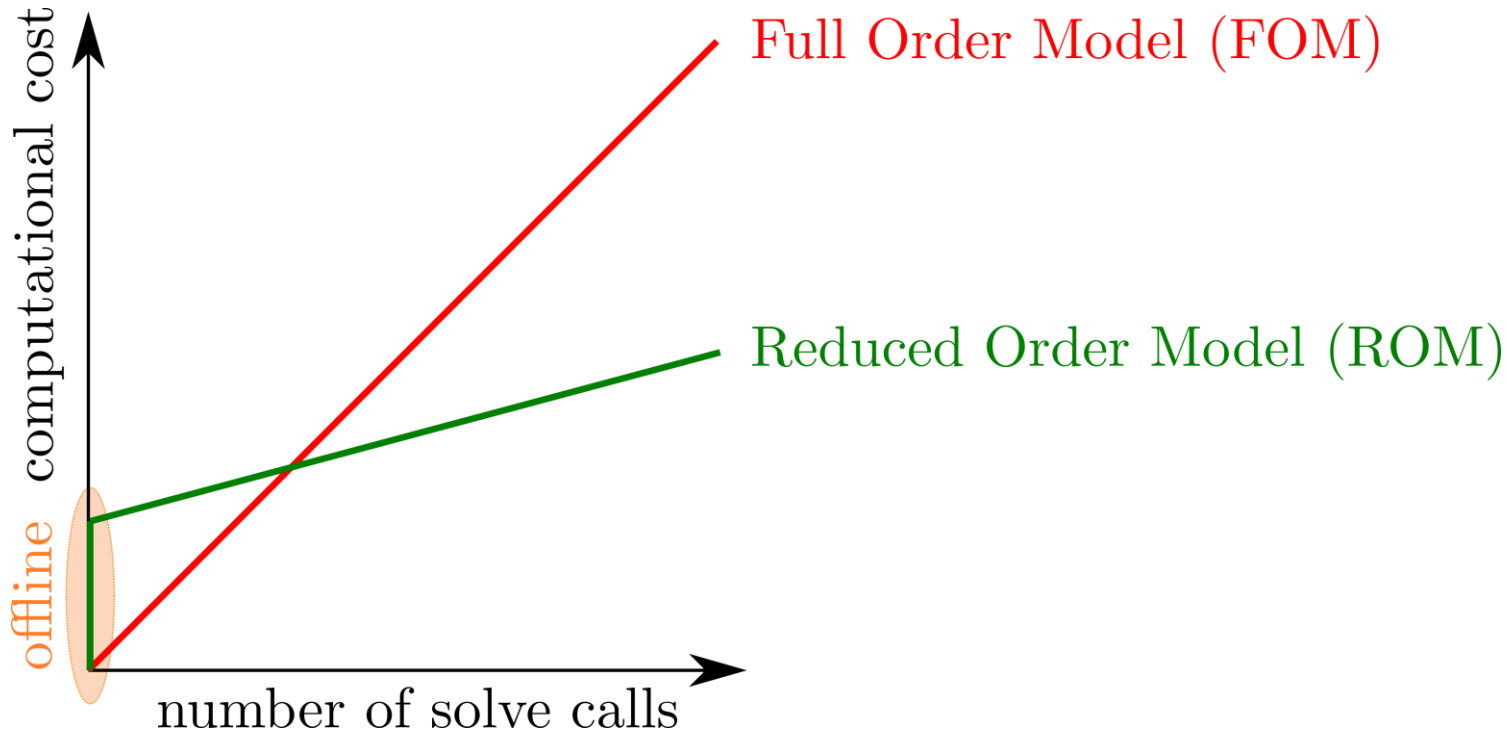
This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Germany, France, Italy, Poland, Switzerland, Norway.

Index

- ROM Theory
  - Motivation
  - Parametrized Problem
  - Approximation Space (Parametric Exploration)
  - Reduced Order Basis
  - System Projection
  - Hyper-reduction
- Demo
- eFlows4HPC Use Case
  - HPC Workflow
  - Results
- HPCWaaS Small Demo

ROM THEORY

Motivation


computational cost
offline
number of solve calls
Full Order Model (FOM)
Reduced Order Model (ROM)

Parametrized Problem

Convection-Diffusion Equation:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u - \mathrm{D}\nabla^2 u = 0$$

Variational Formulation:

$$\int_\Omega w \left(\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) \right) \mathrm{d}\mathbf{x} + \int_\Omega \nabla w \cdot (\mathrm{D}\nabla u) \mathrm{d}\mathbf{x} = 0$$

Semi-Discrete Problem:

$$M_C \frac{\mathrm{d}u}{\mathrm{d}t} = (K - L)u$$

Parametrized Problem

State Variable at Time t:

$$\mathbf{u}_t = \mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_t$$

Adapted Time Discretization:

$$[M_C - \theta\Delta t(K - L)]\,(\mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_{t+1}) = [M_C + (1-\theta)\Delta t(K - L)]\,(\mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_t)$$

Residual:

$$\mathbf{R}_t(\mathbf{u}_t) = \mathbf{R}(\mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_t(t;\boldsymbol{\mu}), t;\boldsymbol{\mu})$$

Given:

$$\mathbf{R}_t(\mathbf{u}_t) = [M_C - \theta\Delta t(K - L)]\,(\mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_{t+1}) - [M_C + (1-\theta)\Delta t(K - L)]\,(\mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_t)$$

Parametrized Problem

Residual:

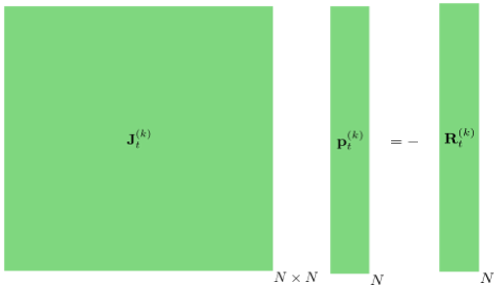
$$\mathbf{R}_t(\mathbf{u}_t) = \mathbf{R}(\mathbf{u}_{\text{ref}} + \boldsymbol{\Delta}\mathbf{u}_t(t;\boldsymbol{\mu}), t;\boldsymbol{\mu})$$

Jacobian (Derivative of Residual):

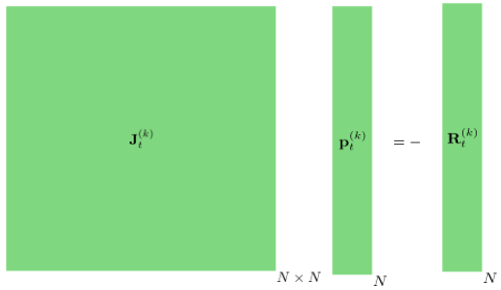
$$J_t = \frac{\partial \mathbf{R}_t}{\partial \mathbf{u}_{t+1}} = M_C - \theta \Delta t (K - L)$$

Newton-Raphson Iteration

$$\begin{aligned}
\mathbf{J}_t^{(k)}(\mathbf{u}_t^{(k)};\boldsymbol{\mu})\mathbf{p}_t^{(k)} &= -\mathbf{R}_t^{(k)}(\mathbf{u}_t^{(k)};\boldsymbol{\mu}) \\
\boldsymbol{\Delta}\mathbf{u}_t^{(k+1)} &= \boldsymbol{\Delta}\mathbf{u}_t^{(k)} + \alpha_t^{(k)}\mathbf{p}_t^{(k)} \\
\mathbf{u}_t^{(k+1)} &= \mathbf{u}_t^{(k)} + \boldsymbol{\Delta}\mathbf{u}_t^{(k+1)},
\end{aligned}$$



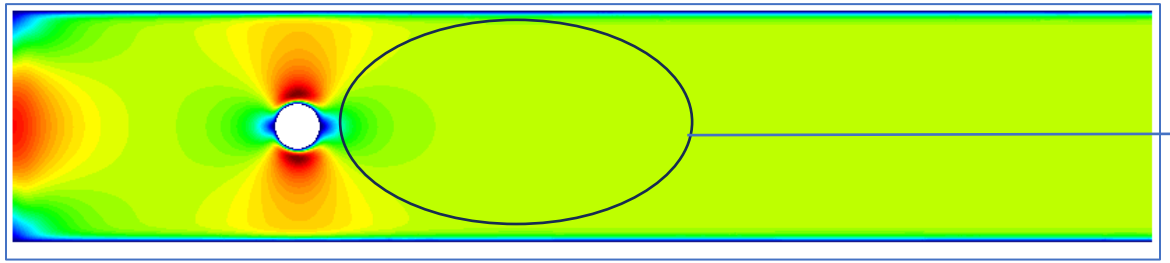
Parametrized Problem



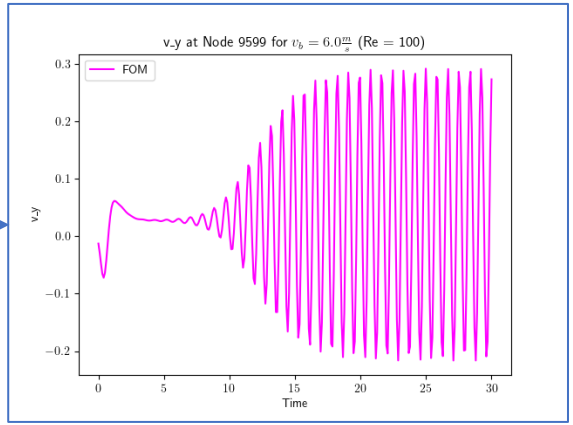
FOM Assembling:

$$\begin{aligned}\mathbf{J}_t^{(k)}(\mathbf{u}_t^{(k)};\boldsymbol{\mu})\mathbf{p}_t^{(k)} &= -\mathbf{R}_t^{(k)}(\mathbf{u}_t^{(k)};\boldsymbol{\mu}) \\ \Delta\mathbf{u}_t^{(k+1)} &= \Delta\mathbf{u}_t^{(k)} + \alpha_t^{(k)}\mathbf{p}_t^{(k)} \\ \mathbf{u}_t^{(k+1)} &= \mathbf{u}_t^{(k)} + \Delta\mathbf{u}_t^{(k+1)},\end{aligned}$$

$$\mathbf{R}_t = \sum_{e=1}^{L} \mathbf{L}^{e^T} \mathbf{R}_t^e$$

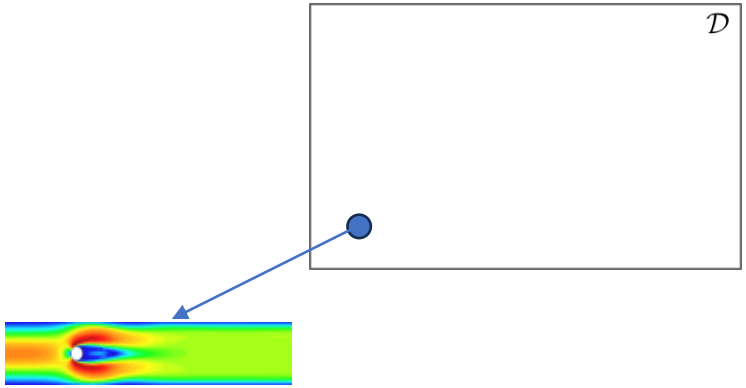


68298 elements



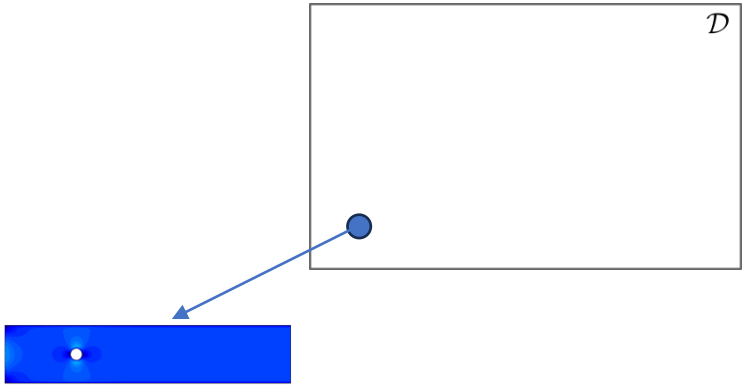
Approximation Space (Parametric Exploration)

$$\mathbf{A}^u = \mathbf{u}_1(\boldsymbol{\mu}_1)$$



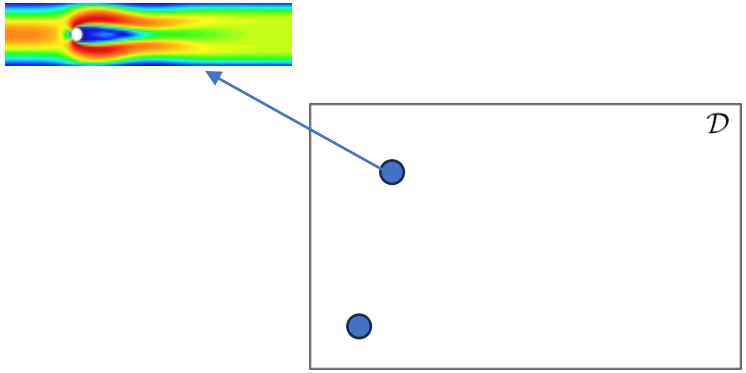
Approximation Space (Parametric Exploration)

$$\mathbf{A}^u = \mathbf{u}_1(\boldsymbol{\mu}_1), \mathbf{u}_2(\boldsymbol{\mu}_1), \ldots, \mathbf{u}_T(\boldsymbol{\mu}_1)$$



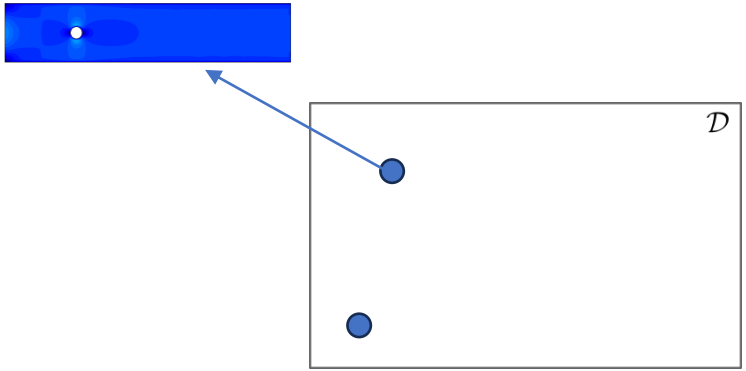
Approximation Space (Parametric Exploration)

$$\mathbf{A}^u = \mathbf{u}_1(\boldsymbol{\mu}_1), \mathbf{u}_2(\boldsymbol{\mu}_1), \ldots, \mathbf{u}_T(\boldsymbol{\mu}_1), \mathbf{u}_1(\boldsymbol{\mu}_2)$$



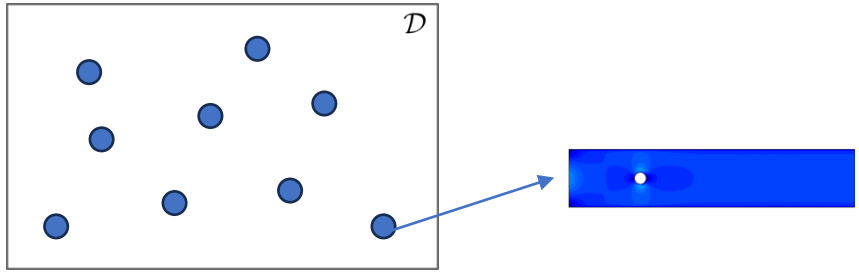
Approximation Space (Parametric Exploration)

$$\mathbf{A}^u = \mathbf{u}_1(\boldsymbol{\mu}_1), \mathbf{u}_2(\boldsymbol{\mu}_1), \ldots, \mathbf{u}_T(\boldsymbol{\mu}_1), \mathbf{u}_1(\boldsymbol{\mu}_2), \mathbf{u}_2(\boldsymbol{\mu}_2), \ldots, \mathbf{u}_T(\boldsymbol{\mu}_2)$$



Approximation Space (Parametric Exploration)

$$\mathbf{A}^u = \mathbf{u}_1(\boldsymbol{\mu}_1), \mathbf{u}_2(\boldsymbol{\mu}_1), \ldots, \mathbf{u}_T(\boldsymbol{\mu}_1) \, , \mathbf{u}_1(\boldsymbol{\mu}_2), \mathbf{u}_2(\boldsymbol{\mu}_2), \ldots, \mathbf{u}_T(\boldsymbol{\mu}_2) \, , \ldots, \mathbf{u}_T(\boldsymbol{\mu}_P)$$



Right Reduced Order Basis (ROB)

$$\mathbf{A}^u = \begin{bmatrix} \mathbf{u}_1(\boldsymbol{\mu}_1), & \mathbf{u}_2(\boldsymbol{\mu}_1), & \dots, & \mathbf{u}_T(\boldsymbol{\mu}_1), & \mathbf{u}_1(\boldsymbol{\mu}_2), & \mathbf{u}_2(\boldsymbol{\mu}_2), & \dots, & \mathbf{u}_T(\boldsymbol{\mu}_2), & \dots, & \mathbf{u}_T(\boldsymbol{\mu}_P) \end{bmatrix}$$

$N \times (T * P)$

Right Reduced Order Basis (ROB)

Singular Value Decomposition:

$$\mathbf{A}^u = \left[\mathbf{u}_1(\boldsymbol{\mu}_1),\ \mathbf{u}_2(\boldsymbol{\mu}_1),\ \ldots,\ \mathbf{u}_T(\boldsymbol{\mu}_1),\ \mathbf{u}_1(\boldsymbol{\mu}_2),\ \mathbf{u}_2(\boldsymbol{\mu}_2),\ \ldots,\ \mathbf{u}_T(\boldsymbol{\mu}_2),\ \ldots,\ \mathbf{u}_T(\boldsymbol{\mu}_P)\right] \approx \boldsymbol{\Phi}\ \boldsymbol{\Sigma}\ \mathbf{V}^T$$

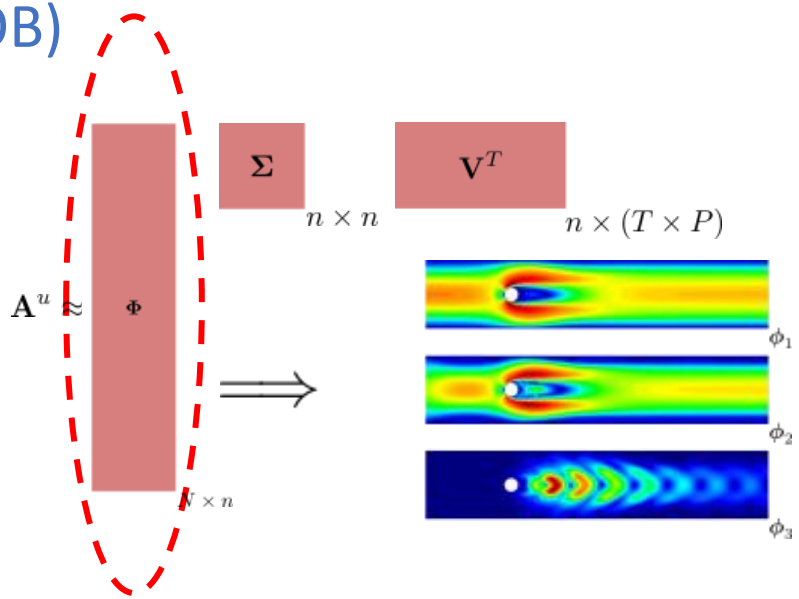
$N \times n$ $\quad n \times n$ $\quad n \times (T \times P)$

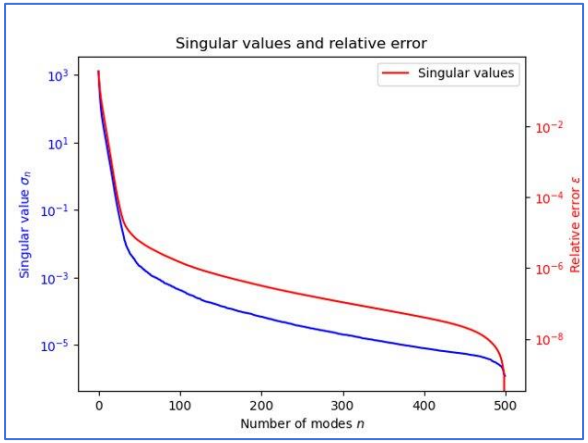
$$A_{n \times d} = \hat{U}_{n \times r}\ \hat{\Sigma}_{r \times r}\ \hat{V}^T_{r \times d}$$

U $\quad \Sigma$ $\quad V^T$

$n \times n$ $\quad n \times d$ $\quad d \times d$

Right Reduced Order Basis (ROB)





Note that:

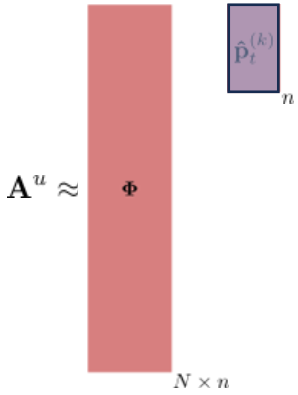
$$\mathbf{A}^u = \mathbf{\Phi}_n \mathbf{\Sigma}_n \mathbf{V}_n^T + \mathbf{E}$$

Where:

$$n \ll N$$

Right Reduced Order Basis (ROB)

Reconstruct each solution field with only few coefficients.



$$\mathbf{A}^u = [\mathbf{u}_1(\boldsymbol{\mu}_1),\ \mathbf{u}_2(\boldsymbol{\mu}_1),\ \ldots,\ \mathbf{u}_T(\boldsymbol{\mu}_1),\ \mathbf{u}_1(\boldsymbol{\mu}_2),\ \mathbf{u}_2(\boldsymbol{\mu}_2),\ \ldots,\ \mathbf{u}_T(\boldsymbol{\mu}_2),\ \ldots,\ \mathbf{u}_T(\boldsymbol{\mu}_P)]$$

Note that:

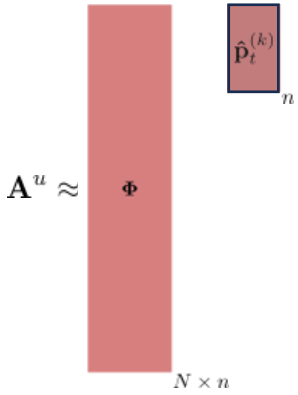
$$\mathbf{A}^u = \boldsymbol{\Phi}_n \boldsymbol{\Sigma}_n \mathbf{V}_n^T + \mathbf{E}$$

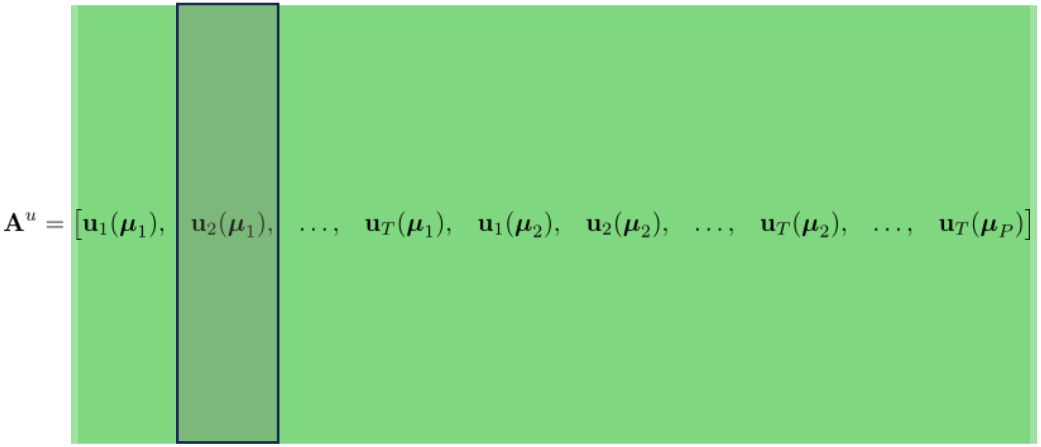
Where:

$$n \ll N$$

Right Reduced Order Basis (ROB)

Reconstruct each solution field with only few coefficients.





Note that:

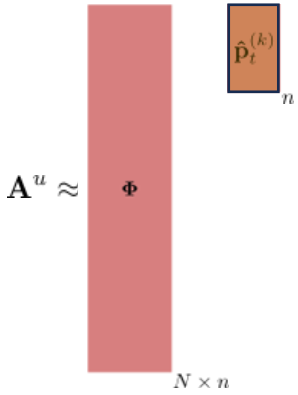
$$\mathbf{A}^u = \boldsymbol{\Phi}_n \boldsymbol{\Sigma}_n \mathbf{V}_n^T + \mathbf{E}$$

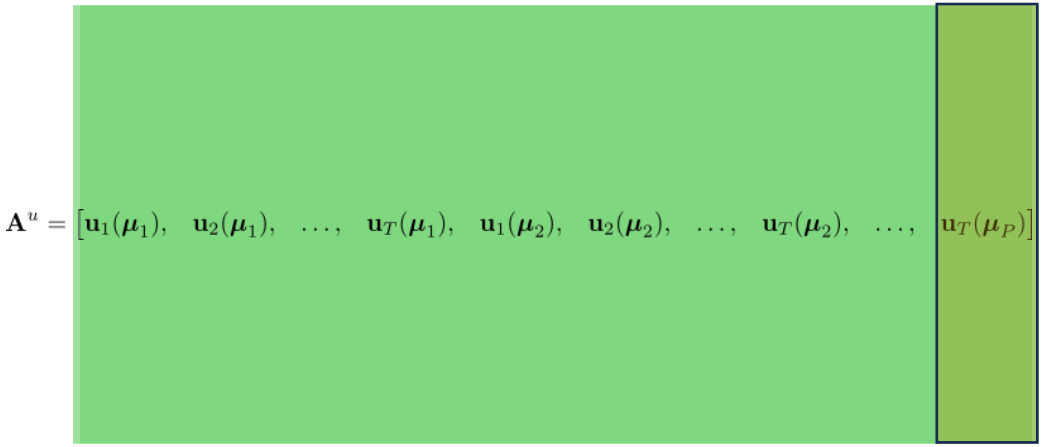
Where:

$$n \ll N$$

Right Reduced Order Basis (ROB)

Reconstruct each solution field with only few coefficients.





Note that:

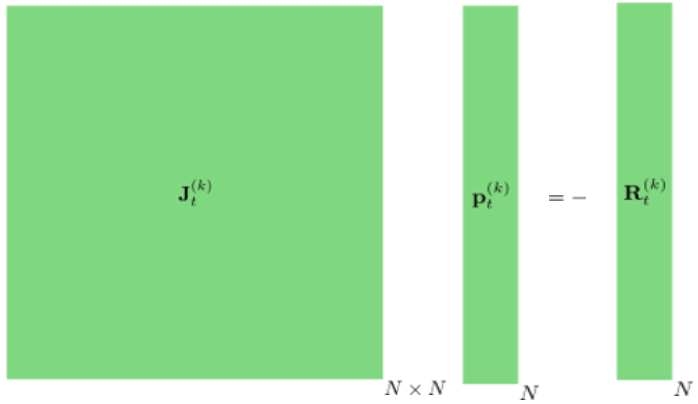
$$\mathbf{A}^u = \boldsymbol{\Phi}_n \boldsymbol{\Sigma}_n \mathbf{V}_n^T + \mathbf{E}$$

Where:

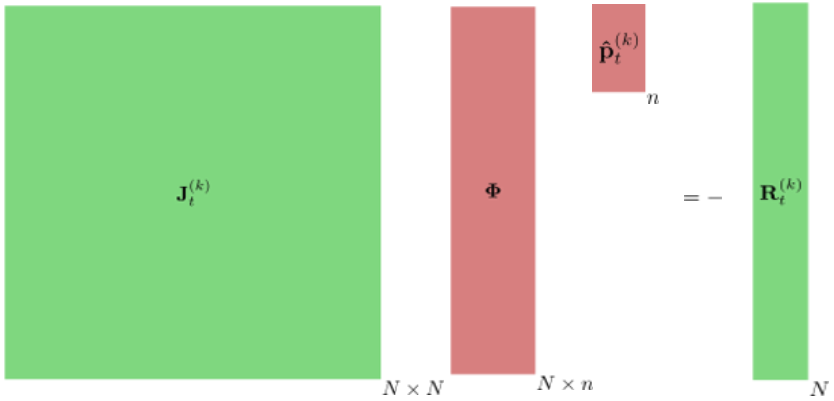
$$n \ll N$$

FOM
Newton-Raphson

$$
\begin{aligned}
\mathbf{J}_t^{(k)}(\mathbf{u}_t^{(k)};\boldsymbol{\mu})\mathbf{p}_t^{(k)} &= -\mathbf{R}_t^{(k)}(\mathbf{u}_t^{(k)};\boldsymbol{\mu}) \\
\Delta\mathbf{u}_t^{(k+1)} &= \Delta\mathbf{u}_t^{(k)} + \alpha_t^{(k)}\mathbf{p}_t^{(k)} \\
\mathbf{u}_t^{(k+1)} &= \mathbf{u}_t^{(k)} + \Delta\mathbf{u}_t^{(k+1)},
\end{aligned}
$$

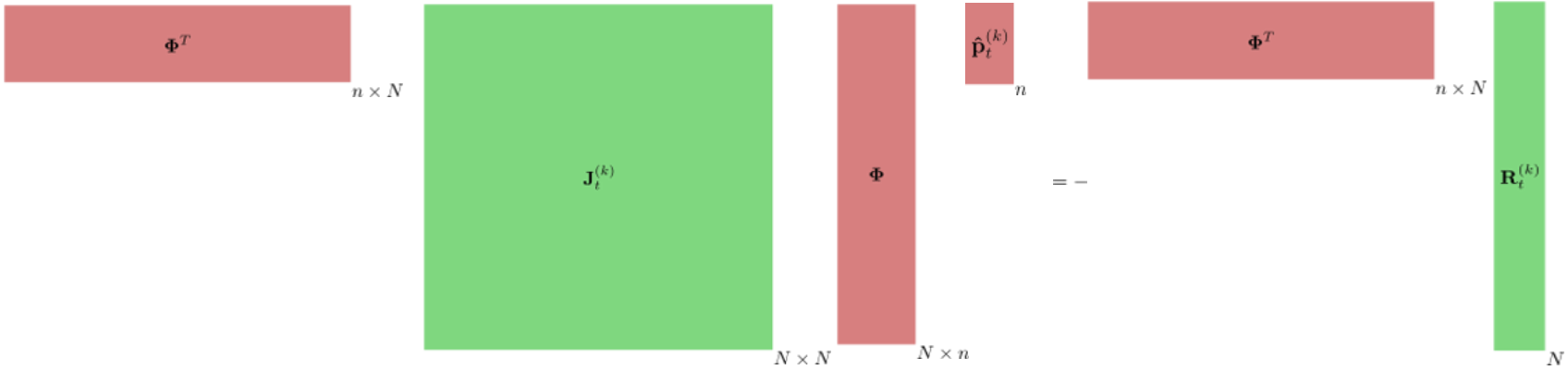


FOM
Newton-Raphson

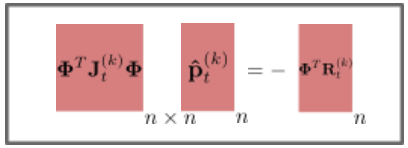


Galerkin Projection

$$
\begin{aligned}
\boldsymbol{\Psi}_t^{T(k)} \mathbf{J}_t^{(k)}(\tilde{\mathbf{u}}_t^{(k)}; \boldsymbol{\mu}) \boldsymbol{\Phi} \hat{\mathbf{p}}_t^{(k)} &= -\boldsymbol{\Psi}_t^{T(k)} \mathbf{R}_t^{(k)}(\tilde{\mathbf{u}}_t^{(k)}; \boldsymbol{\mu}) \\
\boldsymbol{\Delta} \hat{\mathbf{u}}_t^{(k+1)} &= \boldsymbol{\Delta} \hat{\mathbf{u}}_t^{(k)} + \alpha_t^{(k)} \hat{\mathbf{p}}_t^{(k)} \\
\tilde{\mathbf{u}}_t^{(k+1)} &= \tilde{\mathbf{u}}_t^{(k)} + \boldsymbol{\Phi} \boldsymbol{\Delta} \hat{\mathbf{u}}_t^{(k+1)},
\end{aligned}
$$



Galerkin HROM



ROM Assembling:

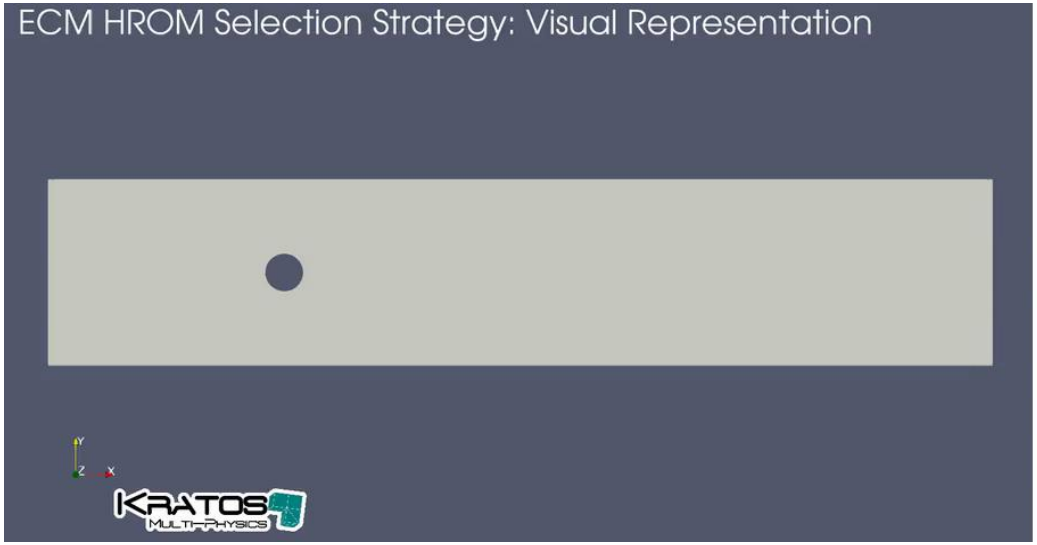
$$\mathbf{\Phi}^T \mathbf{R}_t = \sum_{e=1}^{L} \mathbf{\Phi}^{eT} \mathbf{R}_t^e$$

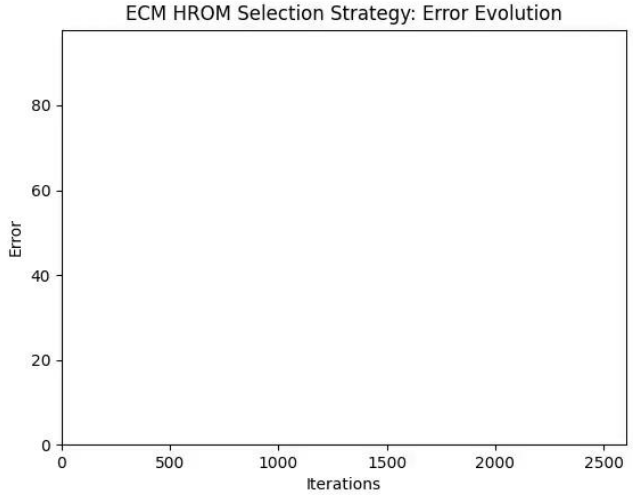
HROM Assembling:

$$\sum_{e \in \mathbf{z}} \omega^e \mathbf{\Phi}_t^{eT} \mathbf{R}_t^e = \mathbf{0}.$$

Galerkin HROM

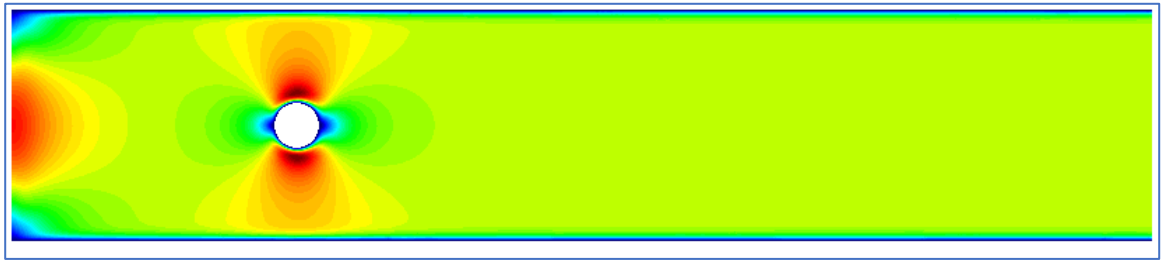
$$\sum_{e \in \mathbf{z}} \omega^e \mathbf{\Phi}_t^{e^T} \mathbf{R}_t^e = \mathbf{0}.$$



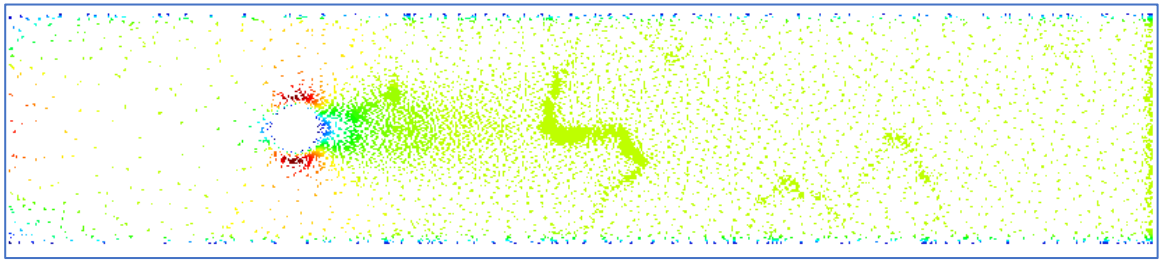


J.A. Hernández, A multiscale method for periodic structures using domain decomposition and ECM-hyperreduction, Comput. Methods Appl. Mech. Engrg. 368 (2020).

ROM vs HROM



68298 elements – Speed up: 4.04



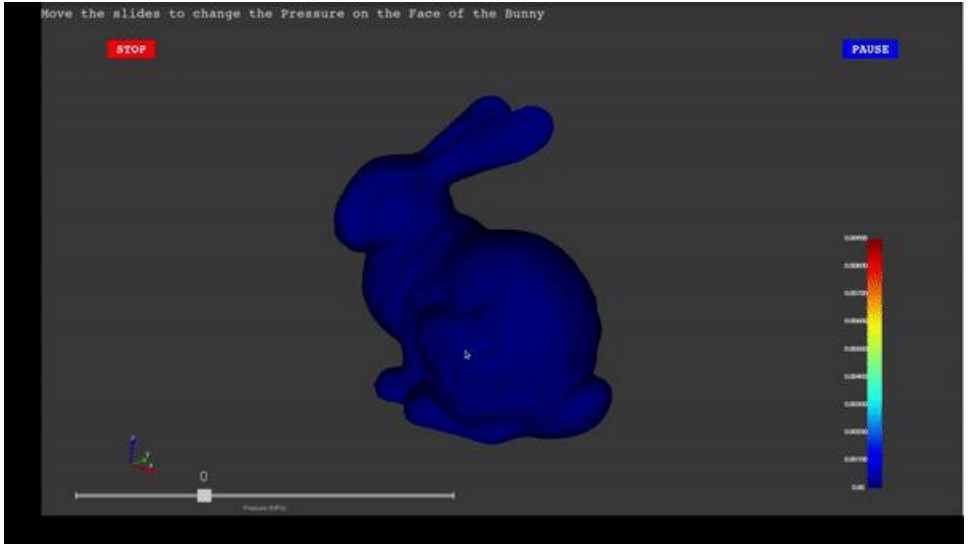
6216 elements – Speed up: 49

Real Time Results

Real Time Results

SIEMENS

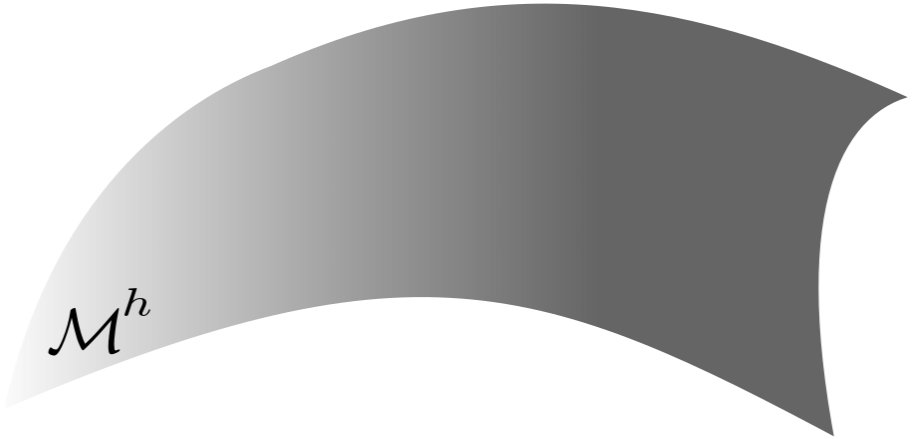
Real Time Results



NONLINEAR REDUCTION

Manifold

$$\mathcal{M}^h = \{\mathbf{u}(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^{N_{\text{dofs}}}$$

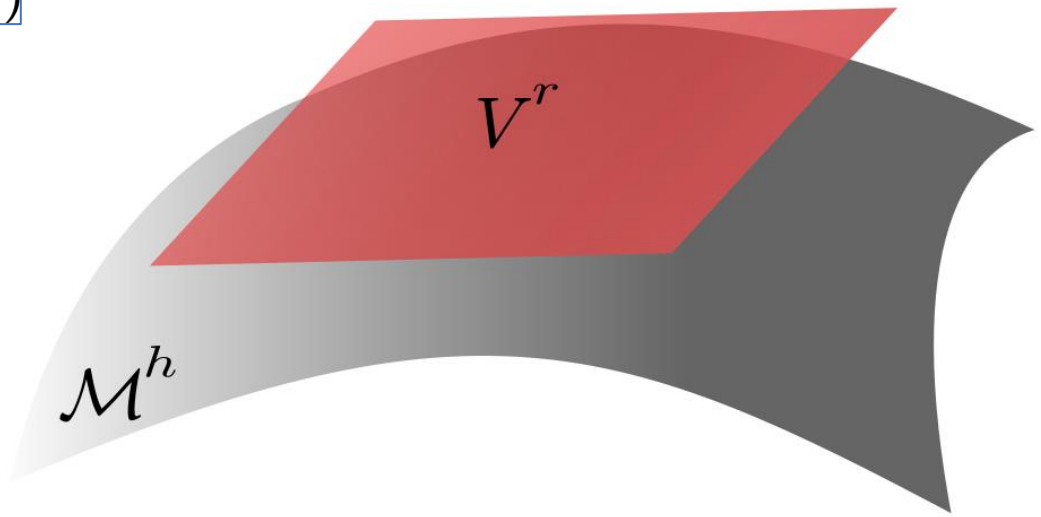


Manifold: Linear Approximation

$$\mathcal{M}^h = \{\mathbf{u}(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^{N_{\text{dofs}}}$$

$$\tilde{\mathbf{u}} \approx \mathbf{D}(\hat{\mathbf{p}}) = \boldsymbol{\Phi}\hat{\mathbf{p}}$$

$$V^r := col(\boldsymbol{\Phi})$$

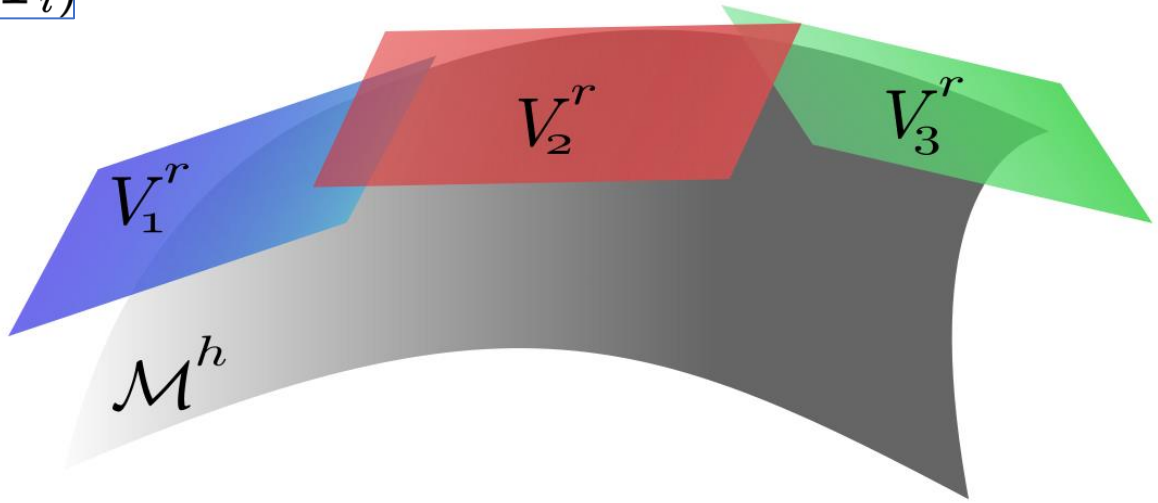


Manifold: Local Linear Approximation (Non-linear)

$$\mathcal{M}^h = \{\mathbf{u}(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^{N_{\text{dofs}}}$$

$$\tilde{\mathbf{u}} \approx \mathbf{D}(\hat{\mathbf{p}}) = \boldsymbol{\Phi}_i \hat{\mathbf{p}}$$

$$V_i^r := col(\boldsymbol{\Phi}_i)$$

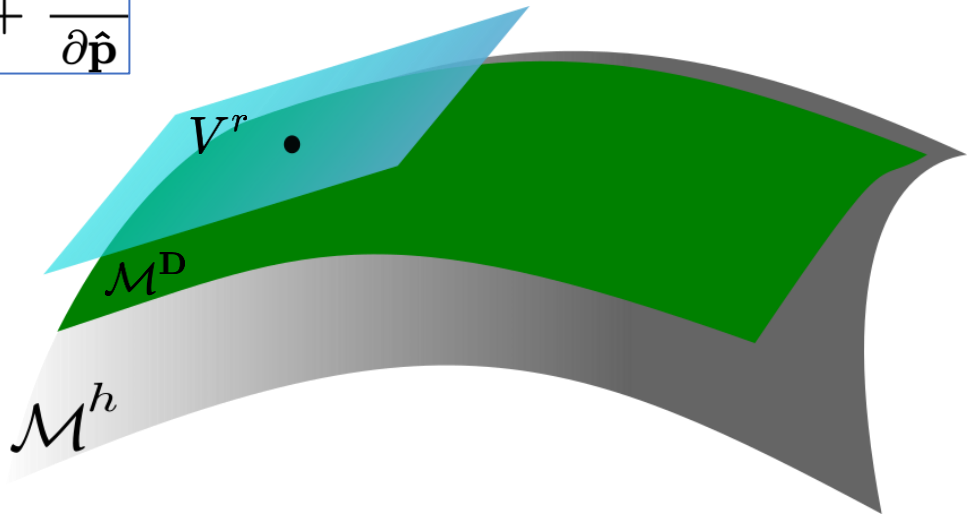


Manifold: Quadratic Approximation (Non-linear)

$$\mathcal{M}^h = \{\mathbf{u}(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^{N_{\text{dofs}}}$$

$$\tilde{\mathbf{u}} \approx \mathbf{D}(\hat{\mathbf{p}}) = \boldsymbol{\Phi}\hat{\mathbf{p}} + \mathbf{H}\hat{\mathbf{p}} \otimes \hat{\mathbf{p}}$$

$$V^r := \frac{\partial \mathbf{D}}{\partial \hat{\mathbf{p}}} = \boldsymbol{\Phi} + \frac{\partial \mathbf{H}}{\partial \hat{\mathbf{p}}}$$

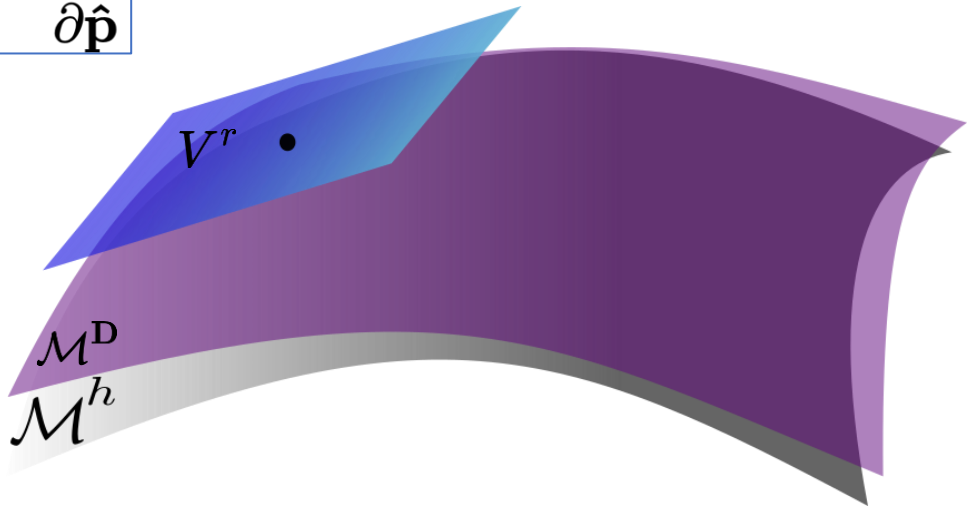


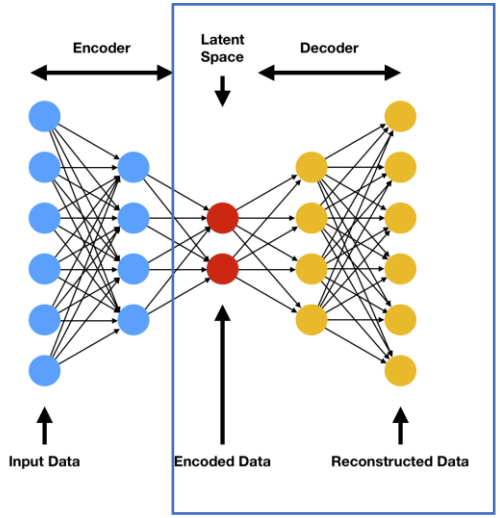
Manifold: Neural Networks (Non-linear)

$$\mathcal{M}^h = \{\mathbf{u}(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset \mathbb{R}^N$$

$$\tilde{\mathbf{u}} \approx \mathbf{D}(\hat{\mathbf{p}}) = \mathcal{NN}(\hat{\mathbf{p}})$$

$$V^r := \frac{\partial \mathbf{D}}{\partial \hat{\mathbf{p}}}$$

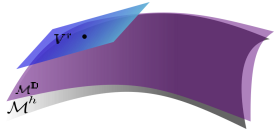


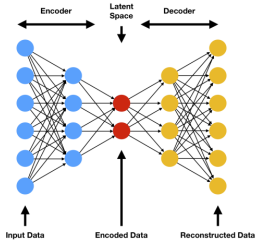


Summary

$$\tilde{\mathbf{u}} \approx \mathbf{D}(\hat{\mathbf{p}})$$

- POD
  - Global
    - Quadratic
    - Lineal
      - Galerkin
      - LSPG
      - Petrov-Galerkin
  - Local
- Neural Networks

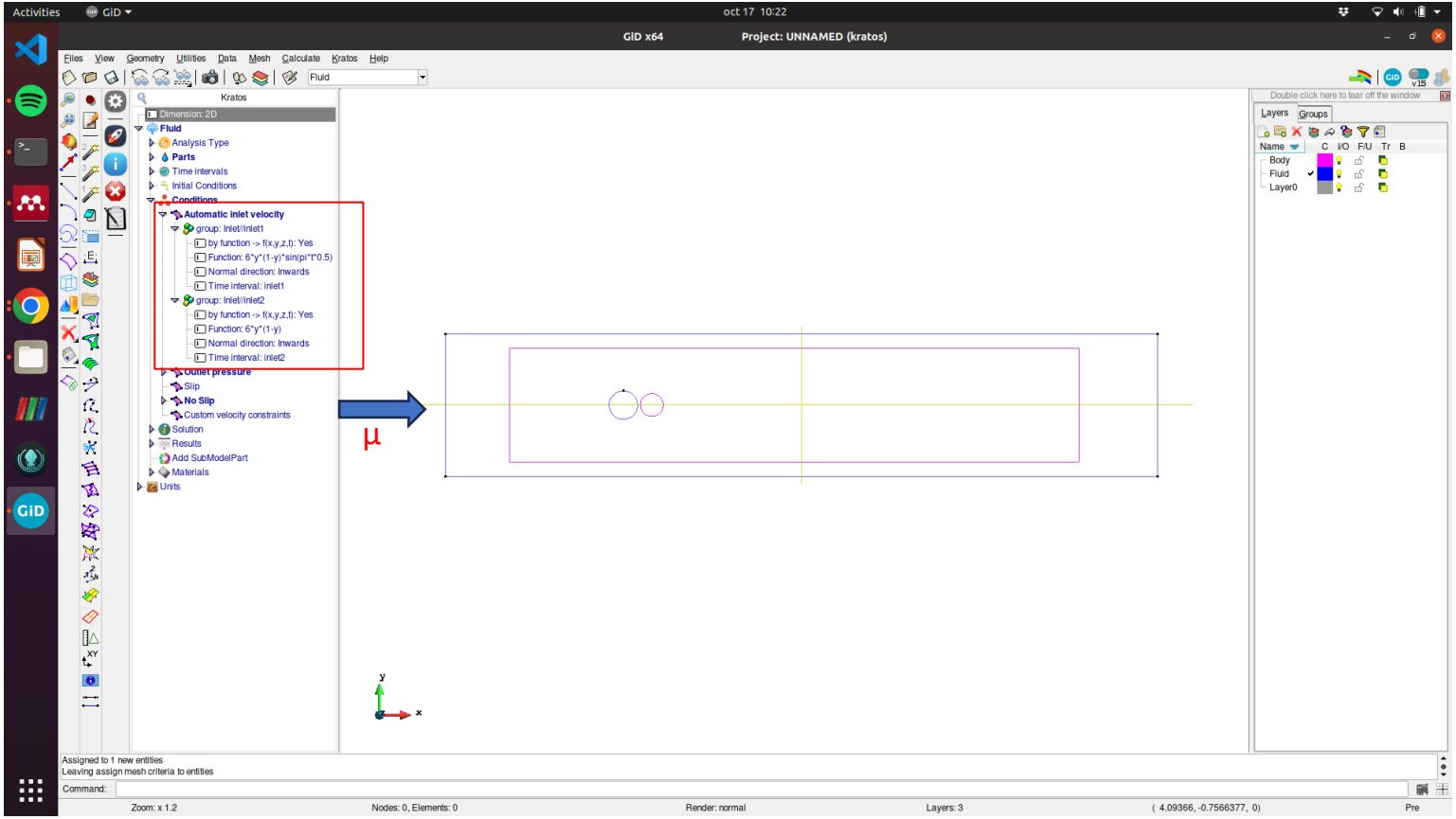




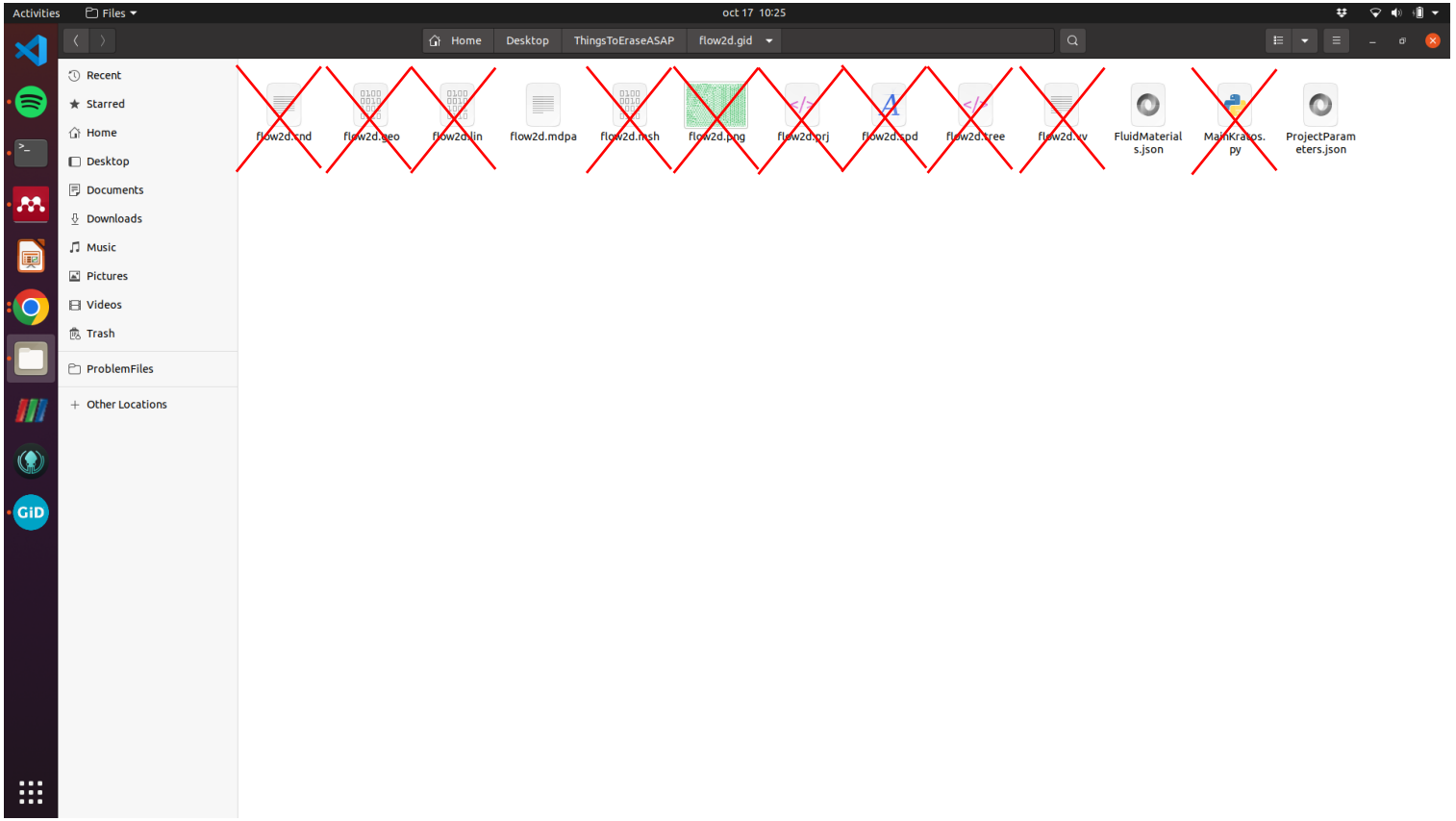
DEMO

https://shorturl.at/fxABR

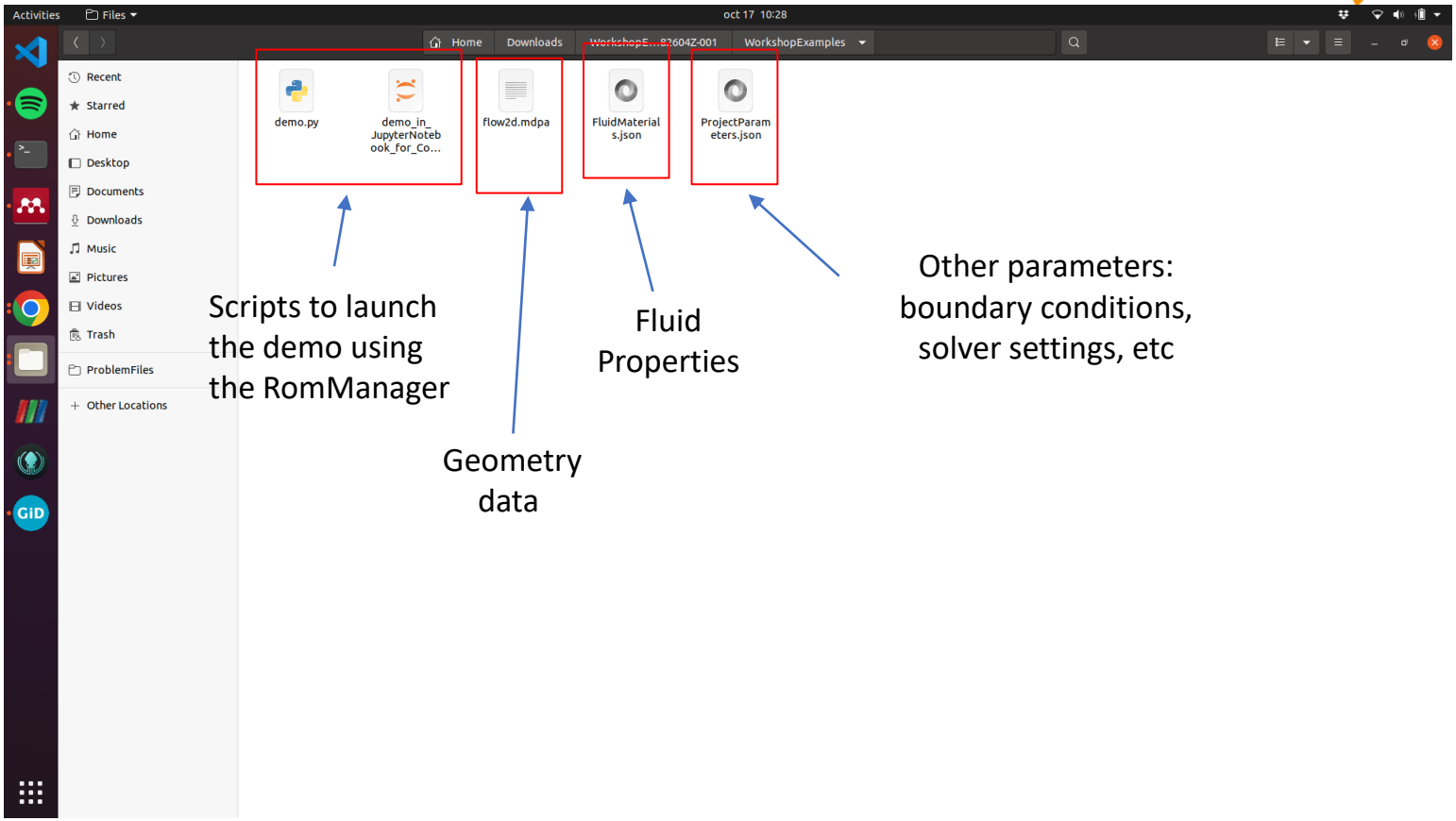
Demo


Activities
GiD
oct 17 10:22
GiD x64
Project: UNNAMED (kratos)
Files View Geometry Utilities Data Mesh Calculate Kratos Help
Fluid
Kratos
Dimension: 2D
Fluid
Analysis Type
Parts
Time intervals
Initial Conditions
Conditions
Automatic inlet velocity
group: Inlet//Inlet1
by function -> f(x,y,z,t): Yes
Function: 6*y*(1-y)*sin(pi*t*0.5)
Normal direction: Inwards
Time interval: Inlet1
group: Inlet//Inlet2
by function -> f(x,y,z,t): Yes
Function: 6*y*(1-y)
Normal direction: Inwards
Time interval: Inlet2
Outlet pressure
Slip
No Slip
Custom velocity constraints
Solution
Results
Add SubModelPart
Materials
Units
μ
Layers
Groups
Name
Body
Fluid
Layer0
Assigned to 1 new entities
Leaving assign mesh criteria to entities
Command:
Zoom: x 1.2
Nodes: 0, Elements: 0
Render: normal
Layers: 3
(4.09366, -0.7566377, 0)

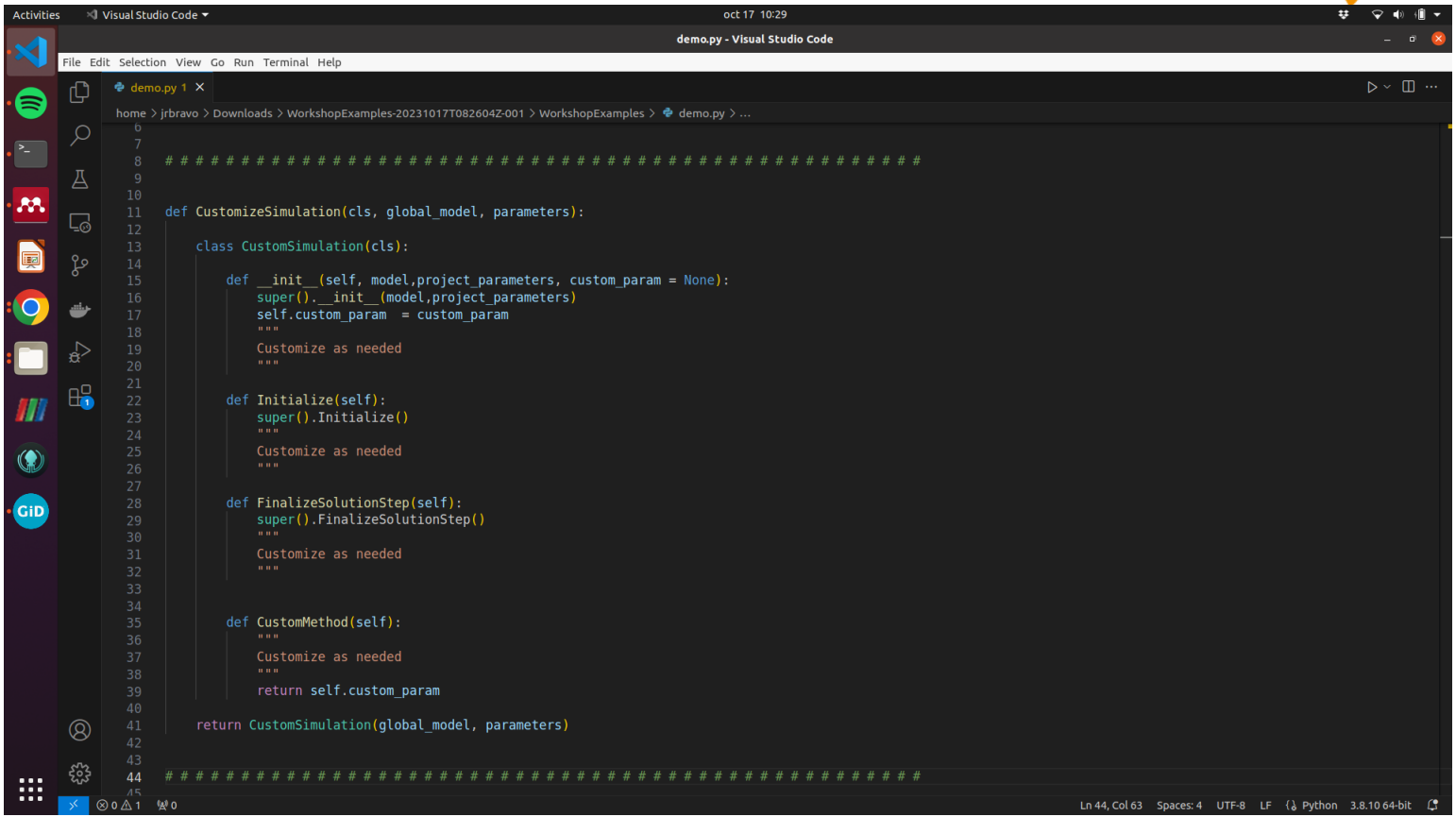
Demo


Activities
Files
oct 17 10:25
Home
Desktop
ThingsToEraseASAP
flow2d.gid
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
ProblemFiles
Other Locations
flow2d.cnd
flow2d.geo
flow2d.lin
flow2d.mdpa
flow2d.msh
flow2d.png
flow2d.prj
flow2d.spd
flow2d.tree
flow2d.vv
FluidMaterials.json
MainKratos.py
ProjectParameters.json

Demo

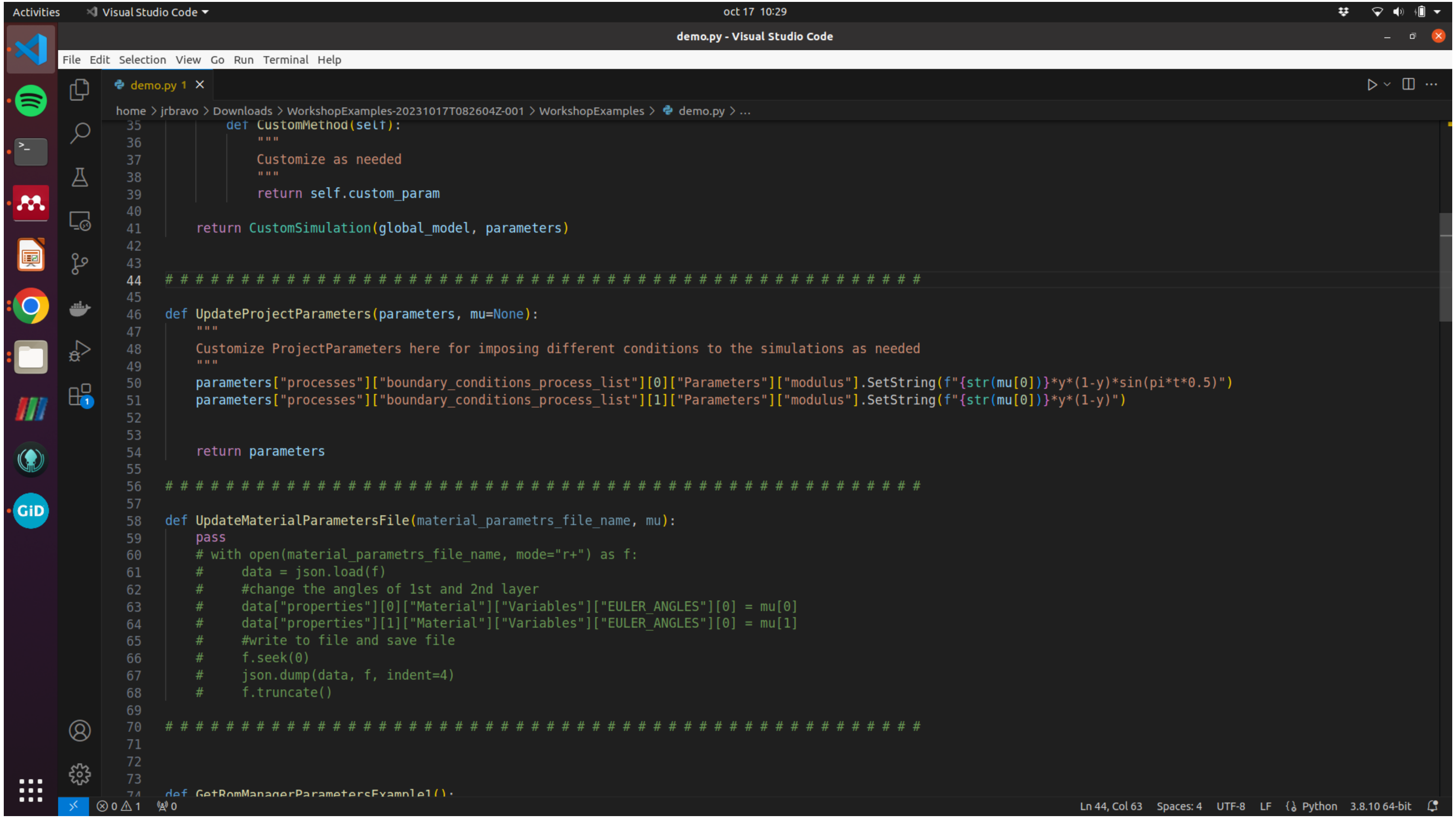

Activities
Files
oct 17 10:28
Home
Downloads
WorkshopExamples
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
ProblemFiles
Other Locations
demo.py
demo_in_JupyterNotebook_for_Co...
flow2d.mdpa
FluidMaterials.json
ProjectParameters.json
Scripts to launch the demo using the RomManager
Geometry data
Fluid Properties
Other parameters: boundary conditions, solver settings, etc

Demo

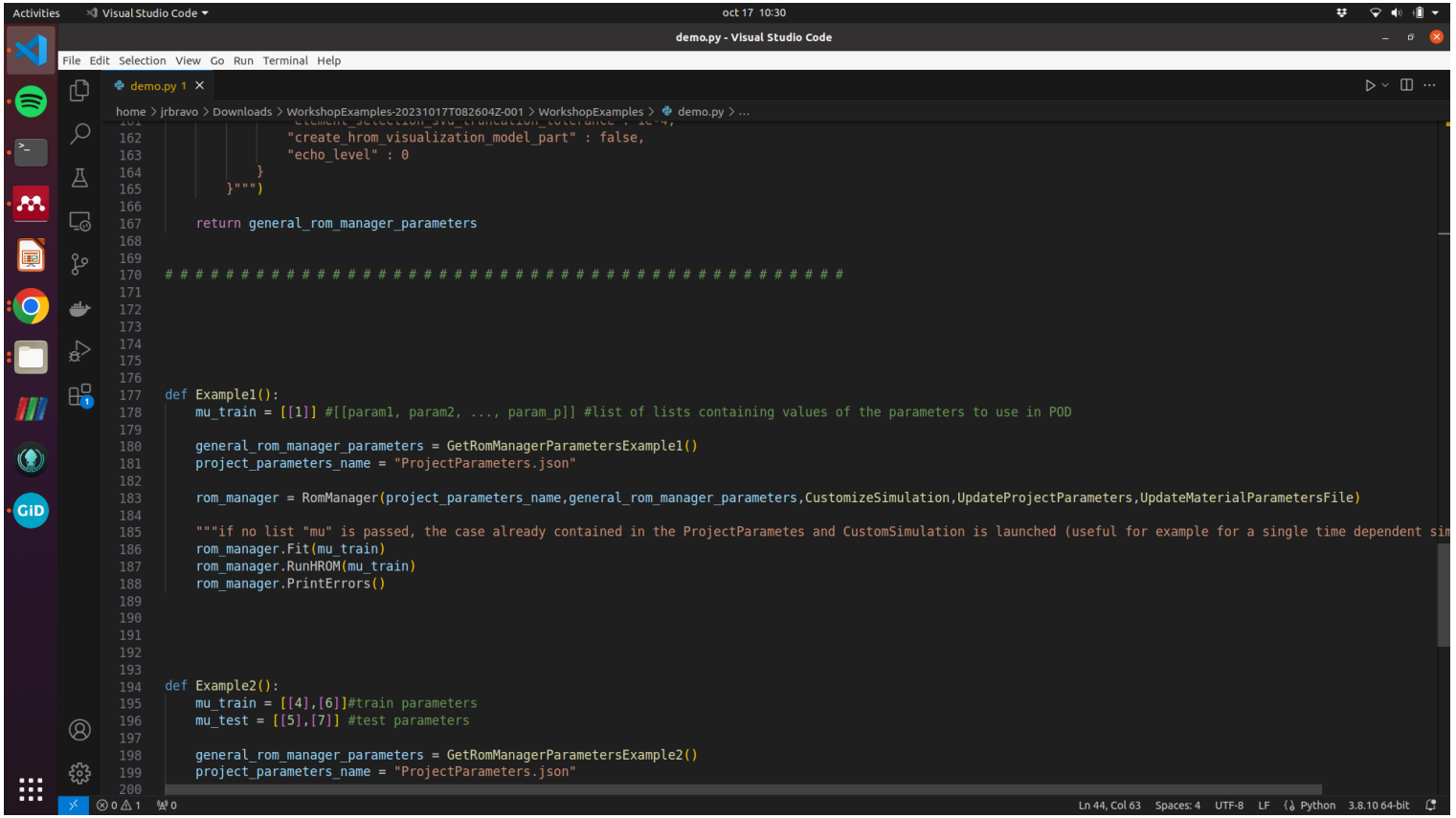



```python
# ##########################################################

def CustomizeSimulation(cls, global_model, parameters):

    class CustomSimulation(cls):

        def __init__(self, model,project_parameters, custom_param = None):
            super().__init__(model,project_parameters)
            self.custom_param  = custom_param
            """
            Customize as needed
            """

        def Initialize(self):
            super().Initialize()
            """
            Customize as needed
            """

        def FinalizeSolutionStep(self):
            super().FinalizeSolutionStep()
            """
            Customize as needed
            """


        def CustomMethod(self):
            """
            Customize as needed
            """
            return self.custom_param

    return CustomSimulation(global_model, parameters)


# ##########################################################
```

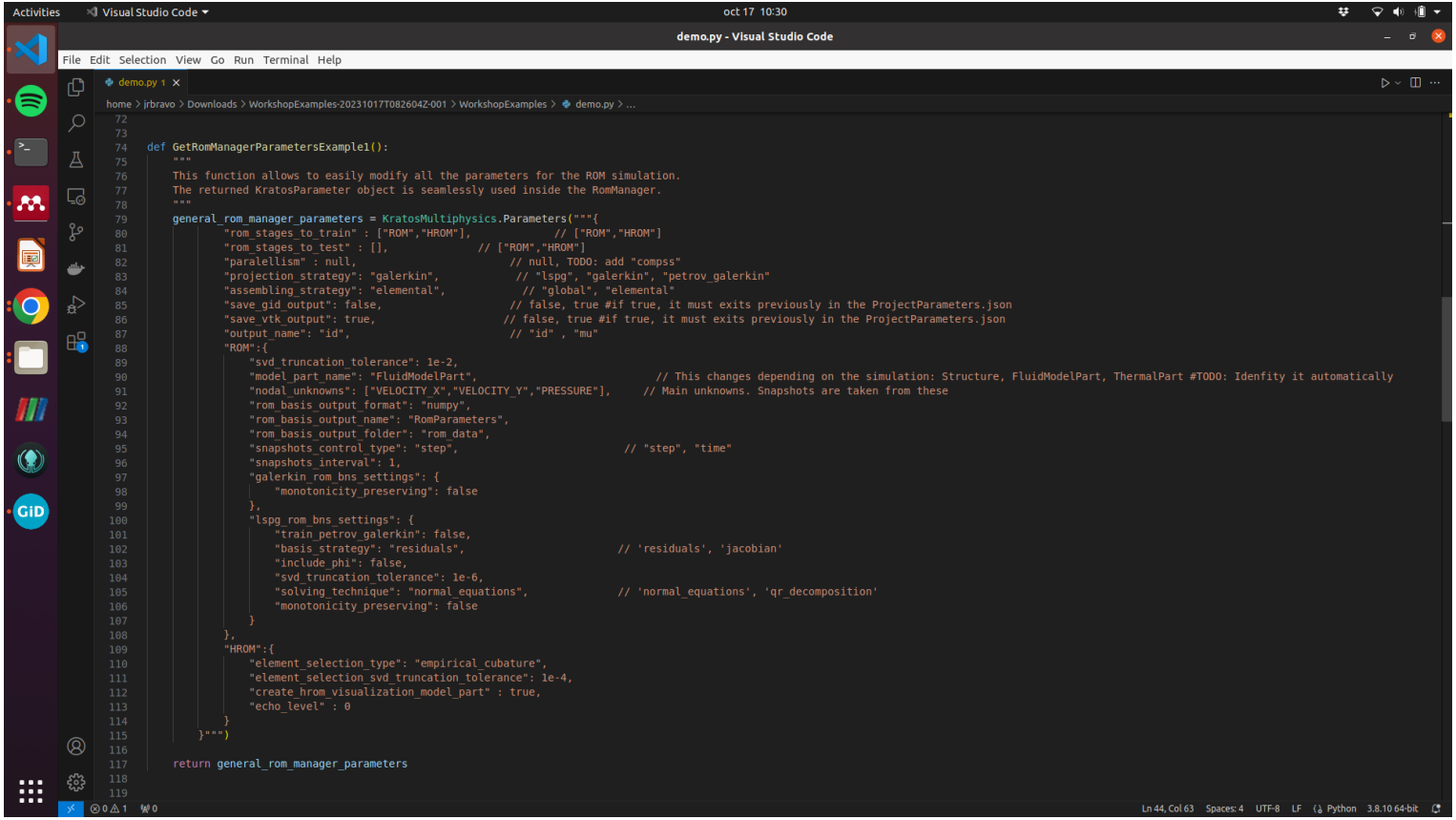
Demo


Activities
Visual Studio Code
oct 17 10:29
demo.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
demo.py 1
home > jrbravo > Downloads > WorkshopExamples-20231017T082604Z-001 > WorkshopExamples > demo.py > ...
def CustomMethod(self):
"""
Customize as needed
"""
return self.custom_param
return CustomSimulation(global_model, parameters)
# #
def UpdateProjectParameters(parameters, mu=None):
"""
Customize ProjectParameters here for imposing different conditions to the simulations as needed
"""
parameters["processes"]["boundary_conditions_process_list"][0]["Parameters"]["modulus"].SetString(f"{str(mu[0])}*y*(1-y)*sin(pi*t*0.5)")
parameters["processes"]["boundary_conditions_process_list"][1]["Parameters"]["modulus"].SetString(f"{str(mu[0])}*y*(1-y)")
return parameters
# #
def UpdateMaterialParametersFile(material_parametrs_file_name, mu):
pass
with open(material_parametrs_file_name, mode="r+") as f:
data = json.load(f)
#change the angles of 1st and 2nd layer
data["properties"][0]["Material"]["Variables"]["EULER_ANGLES"][0] = mu[0]
data["properties"][1]["Material"]["Variables"]["EULER_ANGLES"][0] = mu[1]
#write to file and save file
f.seek(0)
json.dump(data, f, indent=4)
f.truncate()
# #
Ln 44, Col 63
Spaces: 4
UTF-8
LF
Python
3.8.10 64-bit

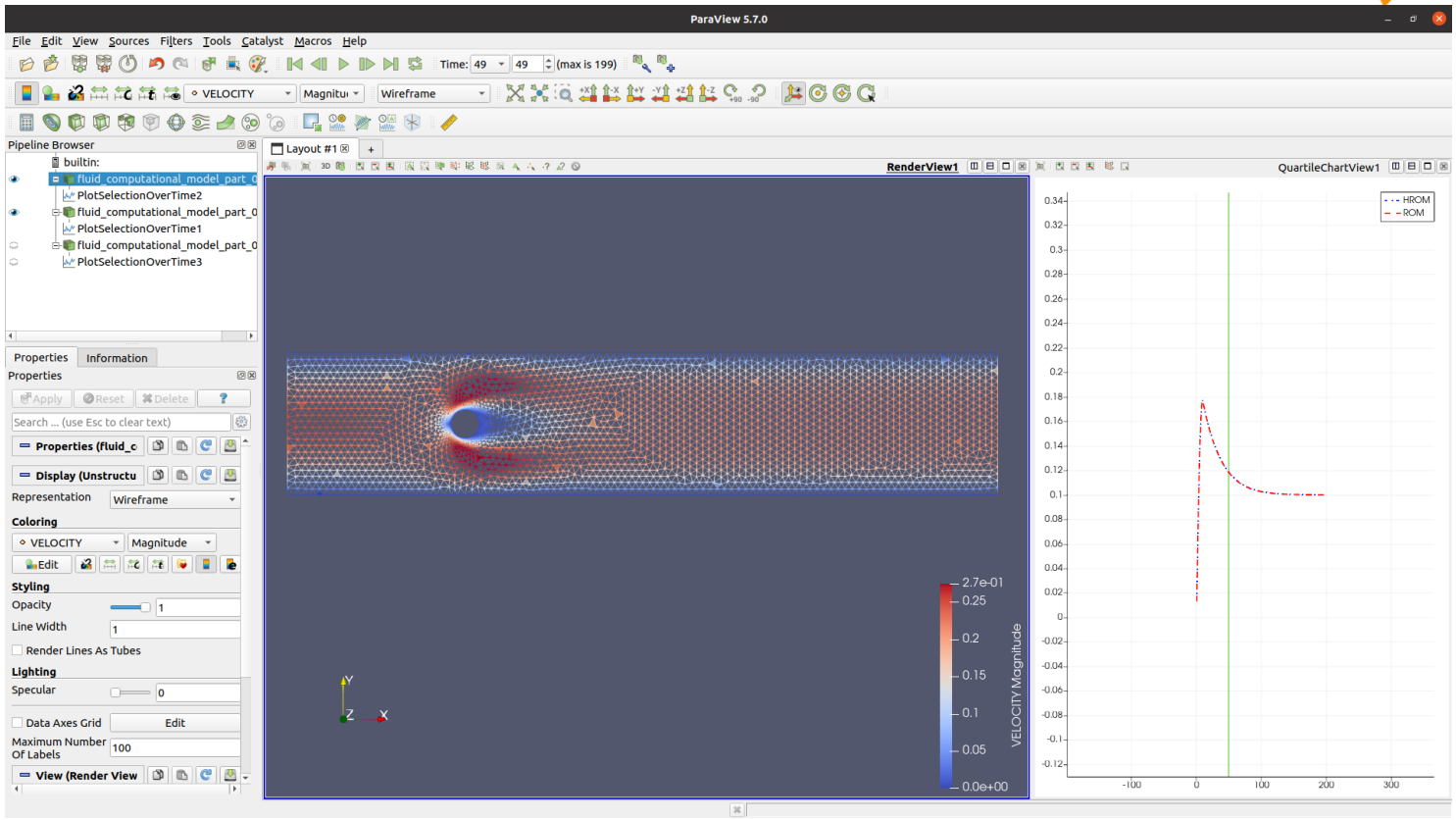
Demo


Activities
Visual Studio Code
oct 17 10:30
demo.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
demo.py 1
home > jrbravo > Downloads > WorkshopExamples-20231017T082604Z-001 > WorkshopExamples > demo.py > ...
"create_hrom_visualization_model_part" : false,
"echo_level" : 0
}
}""")
return general_rom_manager_parameters
# ##
def Example1():
mu_train = [[1]] #[[param1, param2, ..., param_p]] #list of lists containing values of the parameters to use in POD
general_rom_manager_parameters = GetRomManagerParametersExample1()
project_parameters_name = "ProjectParameters.json"
rom_manager = RomManager(project_parameters_name,general_rom_manager_parameters,CustomizeSimulation,UpdateProjectParameters,UpdateMaterialParametersFile)
"""if no list "mu" is passed, the case already contained in the ProjectParametes and CustomSimulation is launched (useful for example for a single time dependent sim
rom_manager.Fit(mu_train)
rom_manager.RunHROM(mu_train)
rom_manager.PrintErrors()
def Example2():
mu_train = [[4],[6]]#train parameters
mu_test = [[5],[7]] #test parameters
general_rom_manager_parameters = GetRomManagerParametersExample2()
project_parameters_name = "ProjectParameters.json"
Ln 44, Col 63
Spaces: 4
UTF-8
LF
Python
3.8.10 64-bit

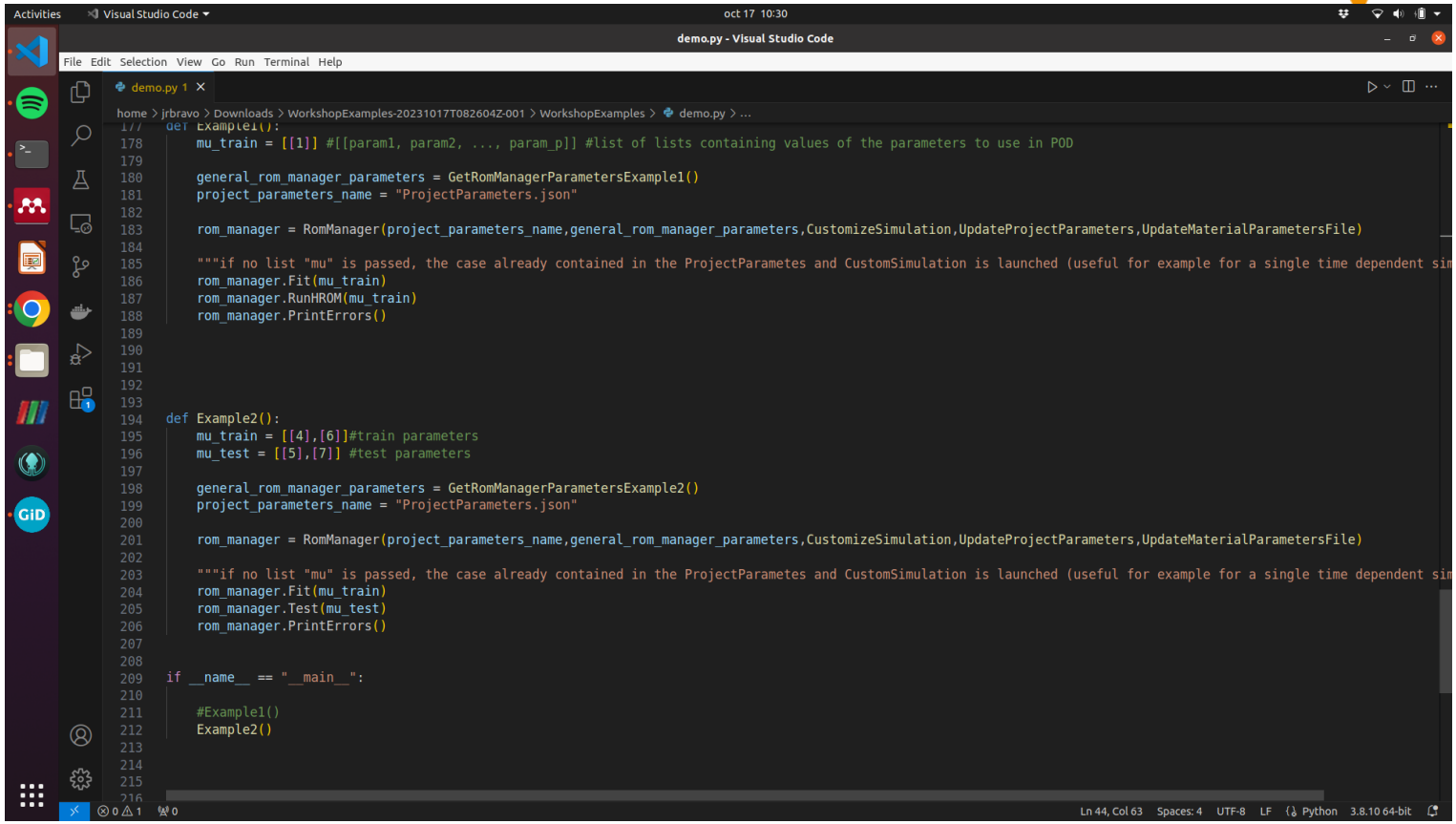
Demo


Activities
Visual Studio Code
oct 17 10:30
demo.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
demo.py 1
home > jrbravo > Downloads > WorkshopExamples-20231017T082604Z-001 > WorkshopExamples > demo.py > ...
def GetRomManagerParametersExample1():
"""
This function allows to easily modify all the parameters for the ROM simulation.
The returned KratosParameter object is seamlessly used inside the RomManager.
"""
general_rom_manager_parameters = KratosMultiphysics.Parameters("""{
"rom_stages_to_train" : ["ROM","HROM"], // ["ROM","HROM"]
"rom_stages_to_test" : [], // ["ROM","HROM"]
"paralellism" : null, // null, TODO: add "compss"
"projection_strategy": "galerkin", // "lspg", "galerkin", "petrov_galerkin"
"assembling_strategy": "elemental", // "global", "elemental"
"save_gid_output": false, // false, true #if true, it must exits previously in the ProjectParameters.json
"save_vtk_output": true, // false, true #if true, it must exits previously in the ProjectParameters.json
"output_name": "id", // "id" , "mu"
"ROM":{
"svd_truncation_tolerance": 1e-2,
"model_part_name": "FluidModelPart", // This changes depending on the simulation: Structure, FluidModelPart, ThermalPart #TODO: Idenfity it automatically
"nodal_unknowns": ["VELOCITY_X","VELOCITY_Y","PRESSURE"], // Main unknowns. Snapshots are taken from these
"rom_basis_output_format": "numpy",
"rom_basis_output_name": "RomParameters",
"rom_basis_output_folder": "rom_data",
"snapshots_control_type": "step", // "step", "time"
"snapshots_interval": 1,
"galerkin_rom_bns_settings": {
"monotonicity_preserving": false
},
"lspg_rom_bns_settings": {
"train_petrov_galerkin": false,
"basis_strategy": "residuals", // 'residuals', 'jacobian'
"include_phi": false,
"svd_truncation_tolerance": 1e-6,
"solving_technique": "normal_equations", // 'normal_equations', 'qr_decomposition'
"monotonicity_preserving": false
}
},
"HROM":{
"element_selection_type": "empirical_cubature",
"element_selection_svd_truncation_tolerance": 1e-4,
"create_hrom_visualization_model_part" : true,
"echo_level" : 0
}
}""")
return general_rom_manager_parameters
Ln 44, Col 63 Spaces: 4 UTF-8 LF Python 3.8.10 64-bit

Demo


ParaView 5.7.0
File Edit View Sources Filters Tools Catalyst Macros Help
Time: 49 49 (max is 199)
VELOCITY
Magnitu
Wireframe
Pipeline Browser
builtin:
fluid_computational_model_part_0
PlotSelectionOverTime2
fluid_computational_model_part_0
PlotSelectionOverTime1
fluid_computational_model_part_0
PlotSelectionOverTime3
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Properties (fluid_c
Display (Unstructu
Representation
Wireframe
Coloring
VELOCITY
Magnitude
Edit
Styling
Opacity
Line Width
Render Lines As Tubes
Lighting
Specular
Data Axes Grid
Edit
Maximum Number Of Labels
100
View (Render View
Layout #1
RenderView1
QuartileChartView1
HROM
ROM
2.7e-01
0.25
0.2
0.15
0.1
0.05
0.0e+00
VELOCITY Magnitude
Y
Z
X

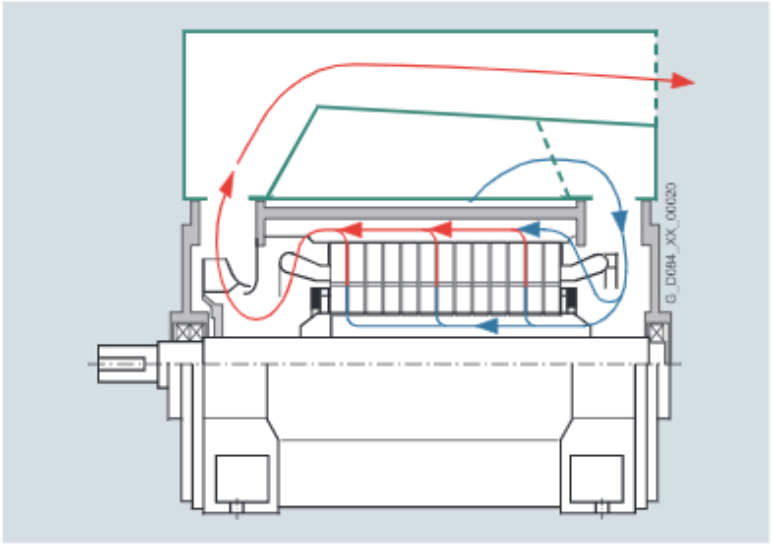
Demo


Activities
Visual Studio Code
oct 17 10:30
demo.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
demo.py 1
home > jrbravo > Downloads > WorkshopExamples-20231017T082604Z-001 > WorkshopExamples > demo.py > ...
def Example1():
mu_train = [[1]] #[[param1, param2, ..., param_p]] #list of lists containing values of the parameters to use in POD
general_rom_manager_parameters = GetRomManagerParametersExample1()
project_parameters_name = "ProjectParameters.json"
rom_manager = RomManager(project_parameters_name,general_rom_manager_parameters,CustomizeSimulation,UpdateProjectParameters,UpdateMaterialParametersFile)
"""if no list "mu" is passed, the case already contained in the ProjectParametes and CustomSimulation is launched (useful for example for a single time dependent sim
rom_manager.Fit(mu_train)
rom_manager.RunHROM(mu_train)
rom_manager.PrintErrors()
def Example2():
mu_train = [[4],[6]]#train parameters
mu_test = [[5],[7]] #test parameters
general_rom_manager_parameters = GetRomManagerParametersExample2()
project_parameters_name = "ProjectParameters.json"
rom_manager = RomManager(project_parameters_name,general_rom_manager_parameters,CustomizeSimulation,UpdateProjectParameters,UpdateMaterialParametersFile)
"""if no list "mu" is passed, the case already contained in the ProjectParametes and CustomSimulation is launched (useful for example for a single time dependent sim
rom_manager.Fit(mu_train)
rom_manager.Test(mu_test)
rom_manager.PrintErrors()
if __name__ == "__main__":
#Example1()
Example2()
Ln 44, Col 63
Spaces: 4
UTF-8
LF
Python
3.8.10 64-bit

EFLOWS4HPC USE CASE

Manufacturing Industry

Simotics H compact PLUS

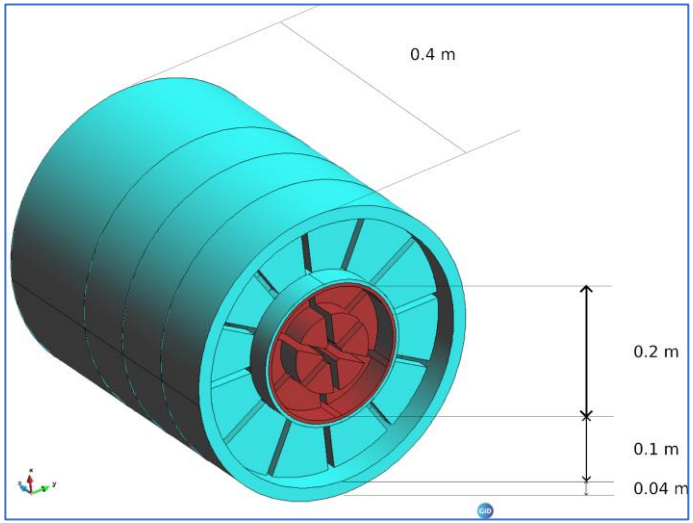


Schematic airflow

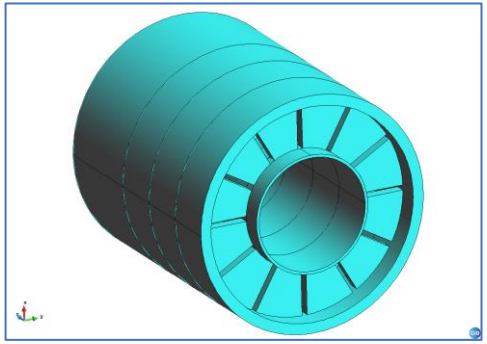
Manufacturing Industry

Model Definition

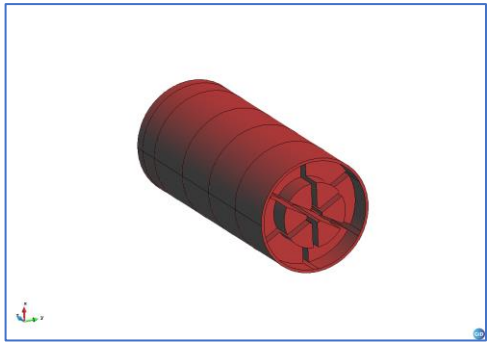
Simotics H compact PLUS



10 million elements

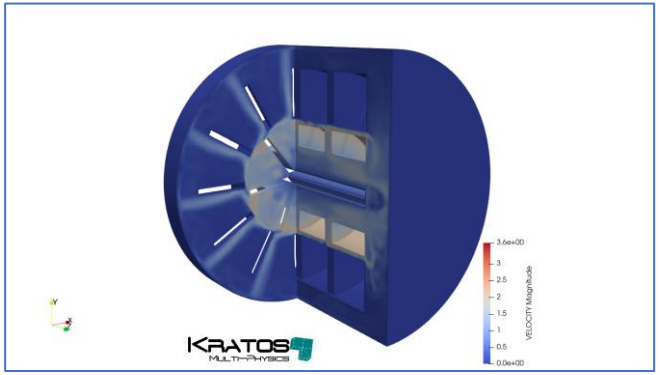


Stator

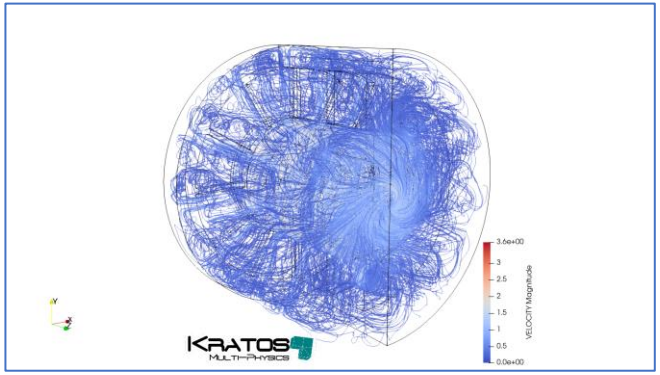


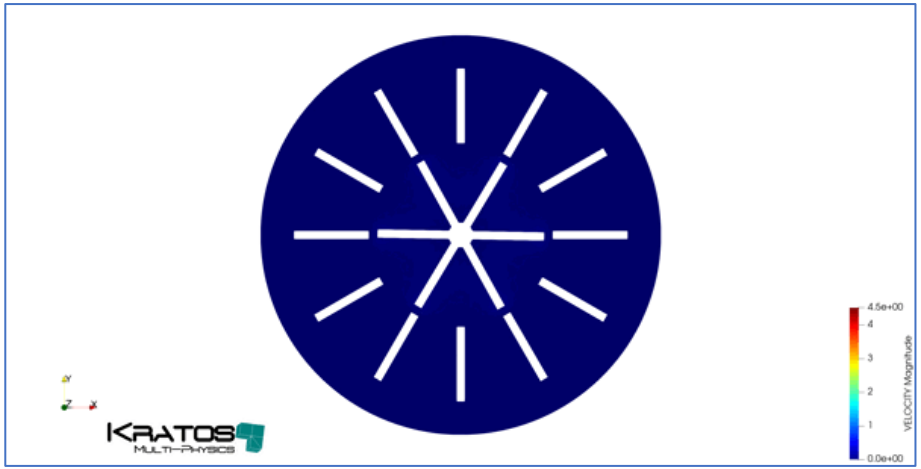
Rotor

Convection Using Navier-Stokes

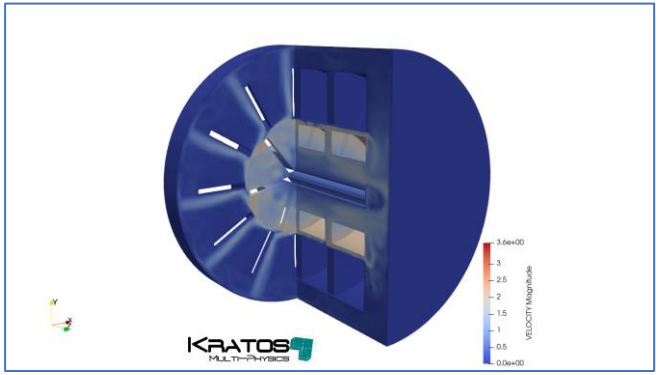


Snapshot of Velocity Contour Field with Sliding Mesh (t=4 sec).

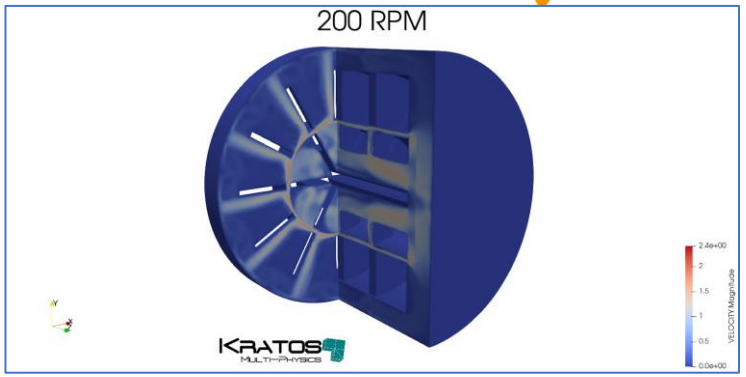




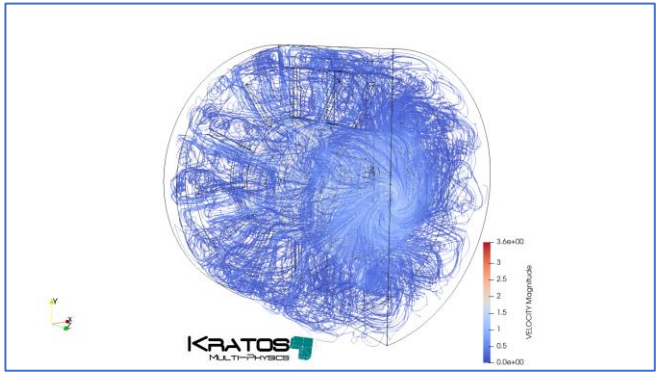
Convection Using Navier-Stokes

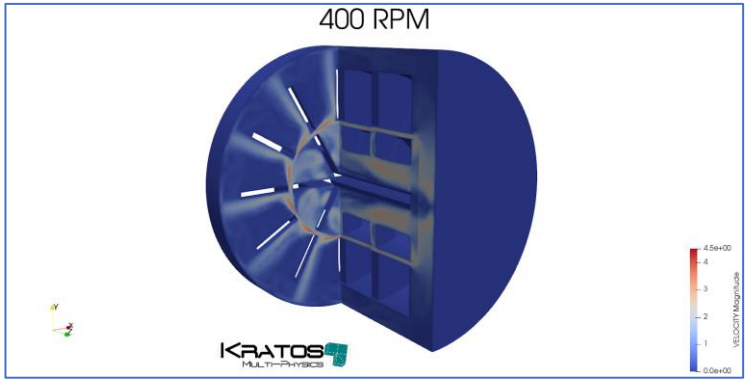


Snapshot of Velocity Contour Field with Sliding Mesh (t=4 sec).

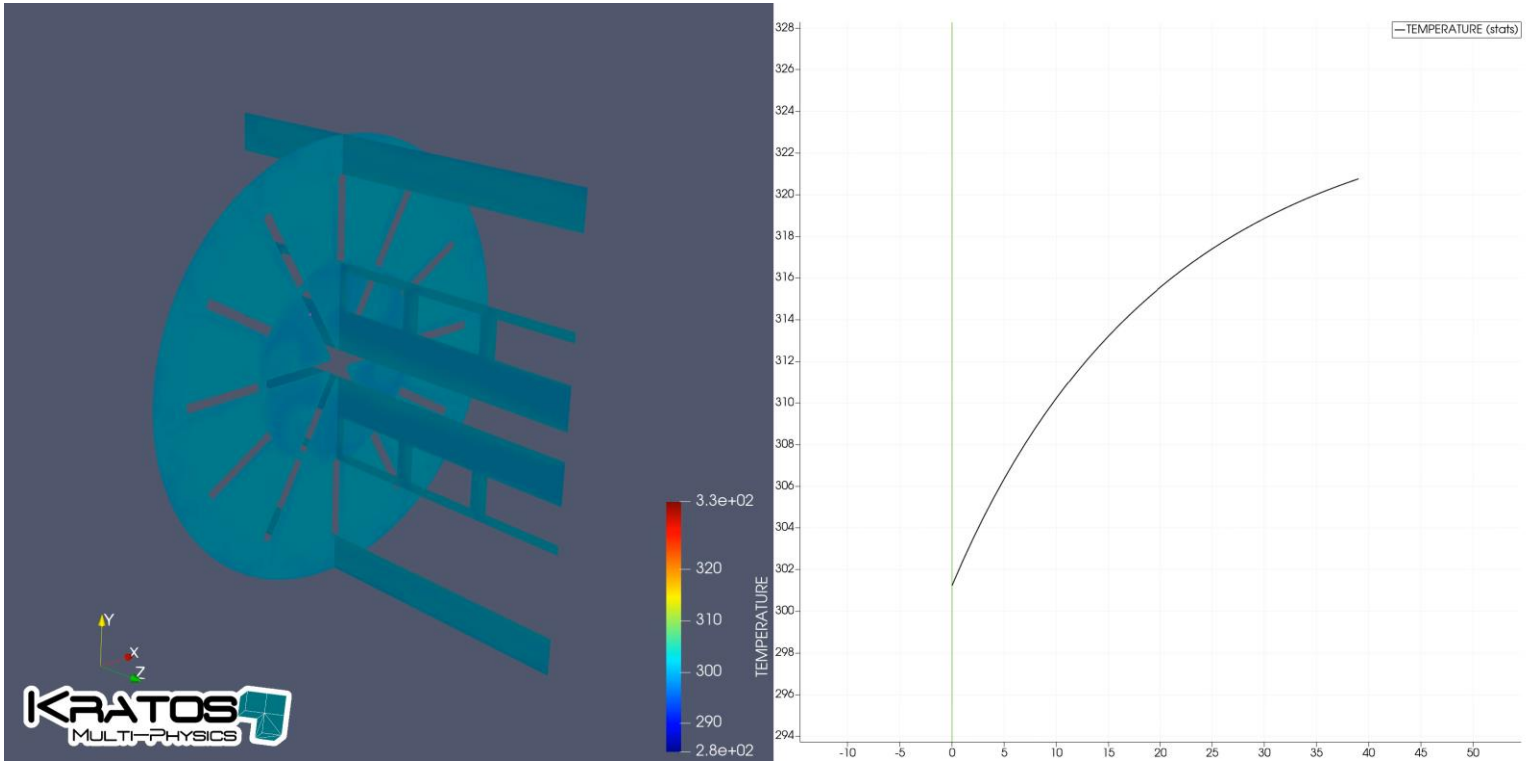


Relative Temporal Average Velocity for Various RPMs (Mesh Velocity Subtracted).



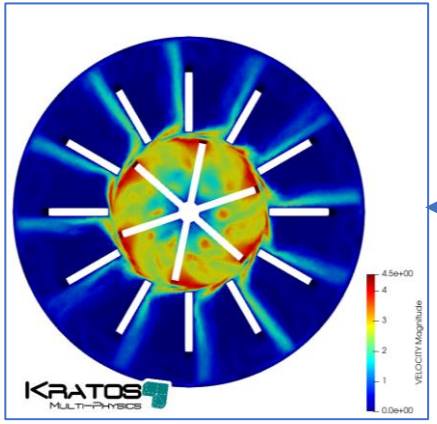


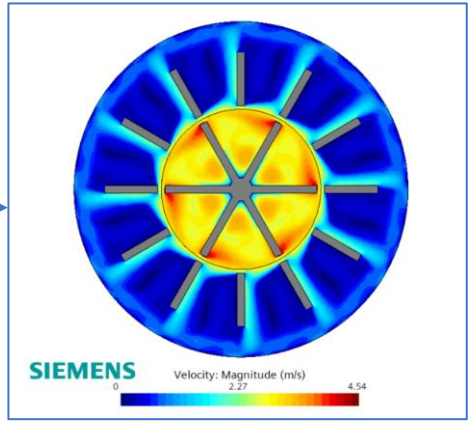
Thermal solution

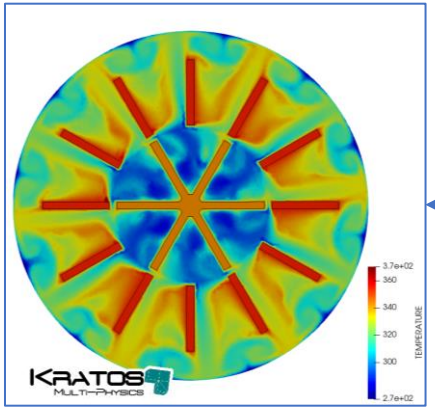


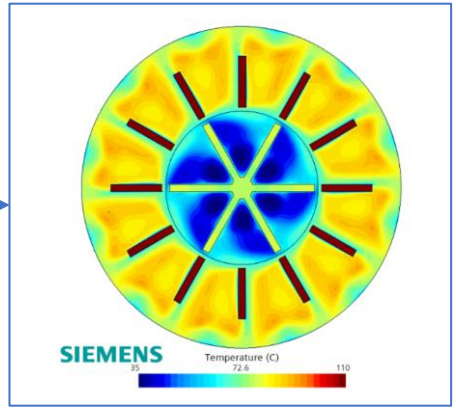
Solution of thermal case.

Validation

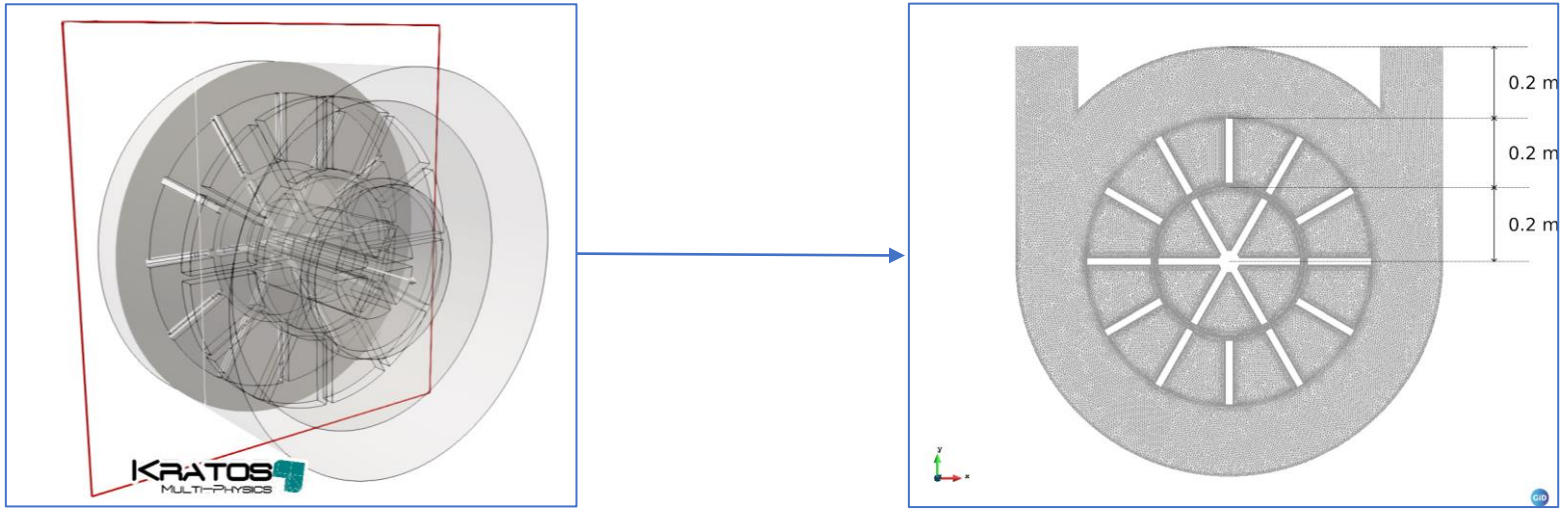

KRATOS
MULTI-PHYSICS
4.5e+00
4
3
2
1
0.0e+00
VELOCITY Magnitude


SIEMENS
Velocity: Magnitude (m/s)
0
2.27
4.54


KRATOS
MULTI-PHYSICS
3.7e+02
360
340
320
300
2.7e+02
TEMPERATURE

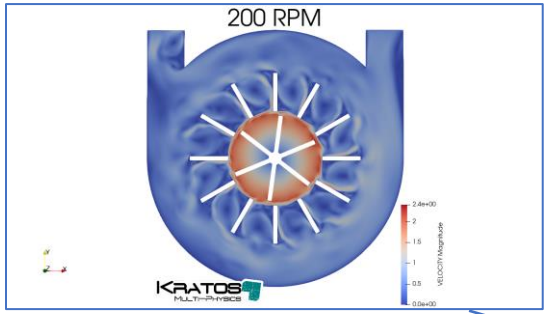

SIEMENS
Temperature (C)
35
72.6
110

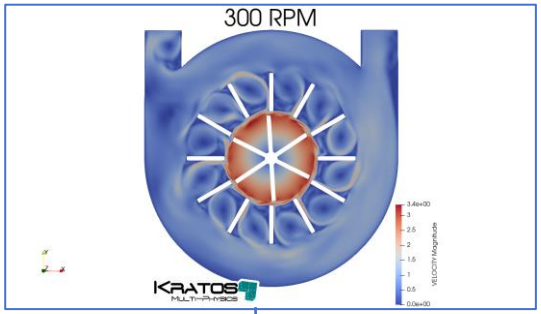
2D Demo Case

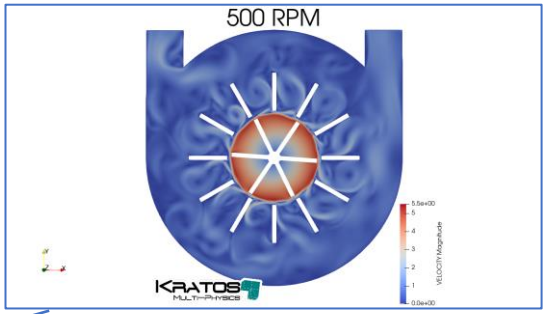


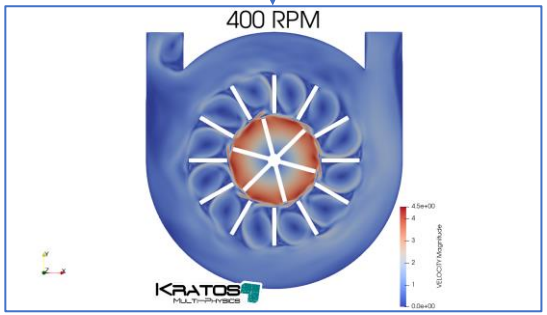
3D cut plane used for building 2D simplification.

Convection Interpolation using EZyRB

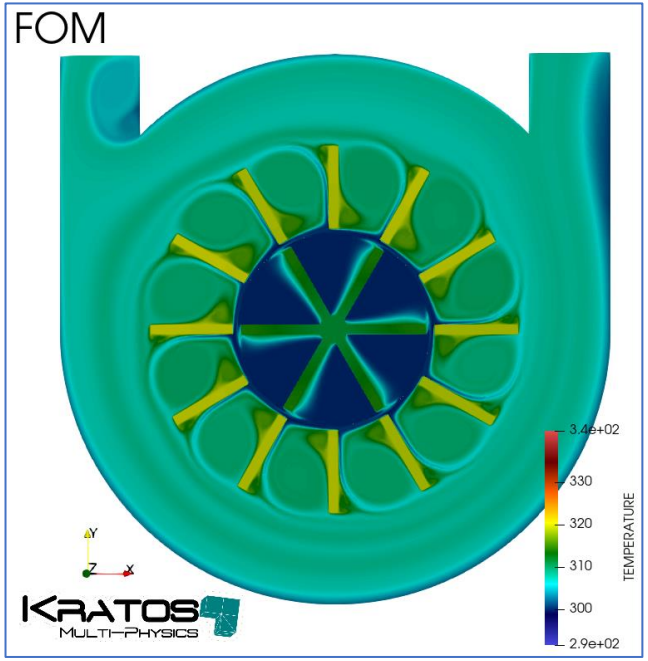




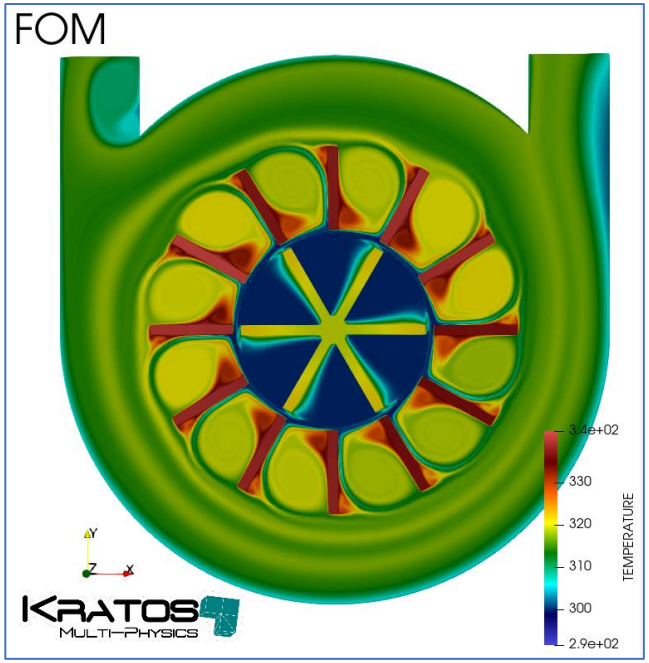




Thermal solution

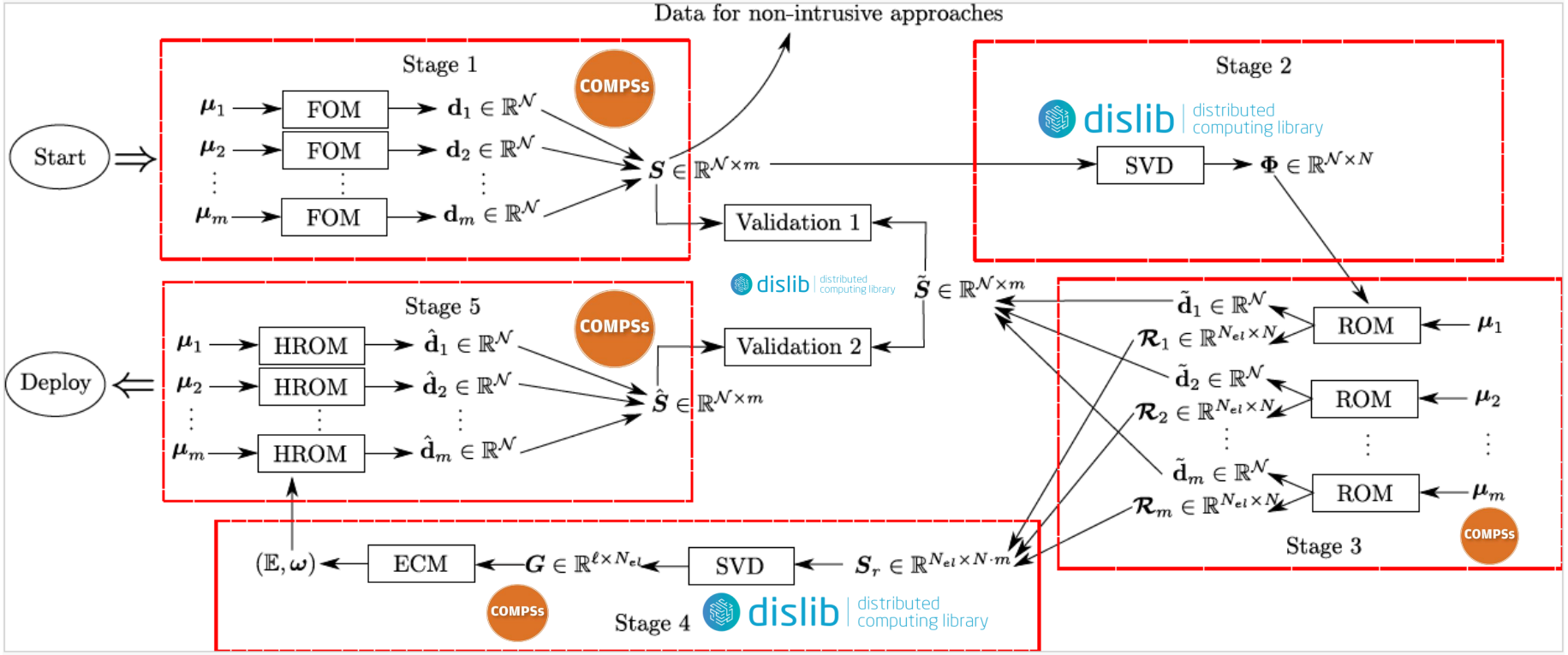


Temperature Field at 4200 secs for 200 RPM.

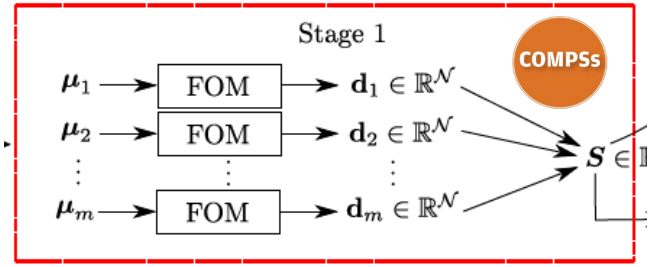


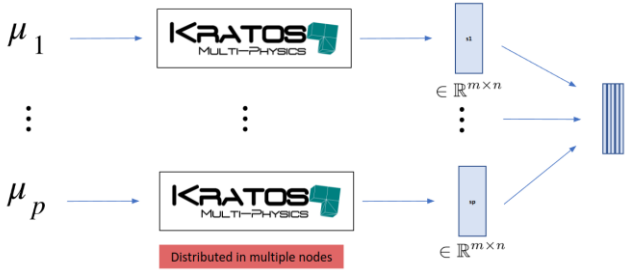
Temperature Field at 4200 secs for 400 RPM.

High-Performance Computing: Parallel Workflow



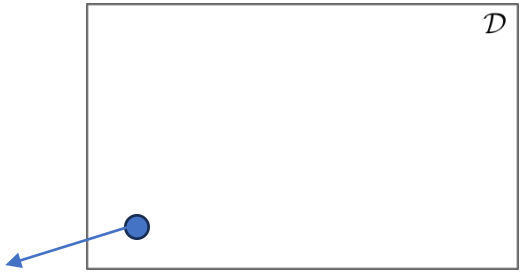
High-Performance Computing: Decorator






```python
@constraint(computingUnits=argv[2])
@task(returns = 3)
def ExecuteInstance_Task(pickled_parameters,Cases,instance, path):
    # overwrite the old parameters serializer with the unpickled one
    serialized_parameters = pickle.loads(pickled_parameters)
    current_parameters = KratosMultiphysics.Parameters()
    serialized_parameters.Load("ParametersSerialization",current_parameters)
    del(serialized_parameters)
    # get sample
    sample = GetValueFromListList(Cases,instance) # take one of them
    simulation = TrainROM(current_parameters,sample,path)
    simulation.Run()
    snapshots = simulation.GetSnapshotsMatrices()
    control_point_matrix = simulation.GetSolutionsAtControlPoint()
    return snapshots[0], snapshots[1], control_point_matrix
```

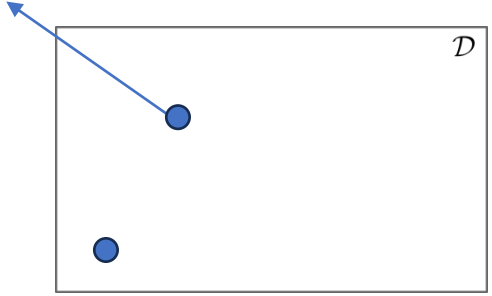
Convection-Diffusion Parameters



$q_1 = 50,000\,\mathrm{W/m^2}$ with $\omega_1 = 200\,\mathrm{RPMs}$

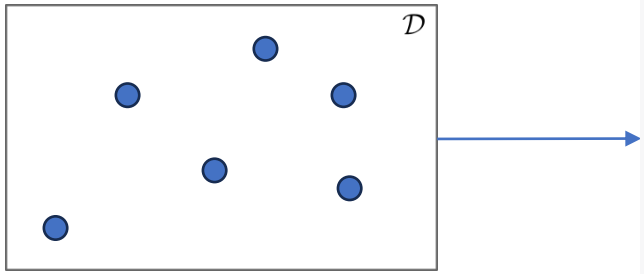
Convection-Diffusion Parameters

$q_2 = 62,500\,\mathrm{W/m^2}$ with $\omega_2 = 250\,\mathrm{RPMs}$



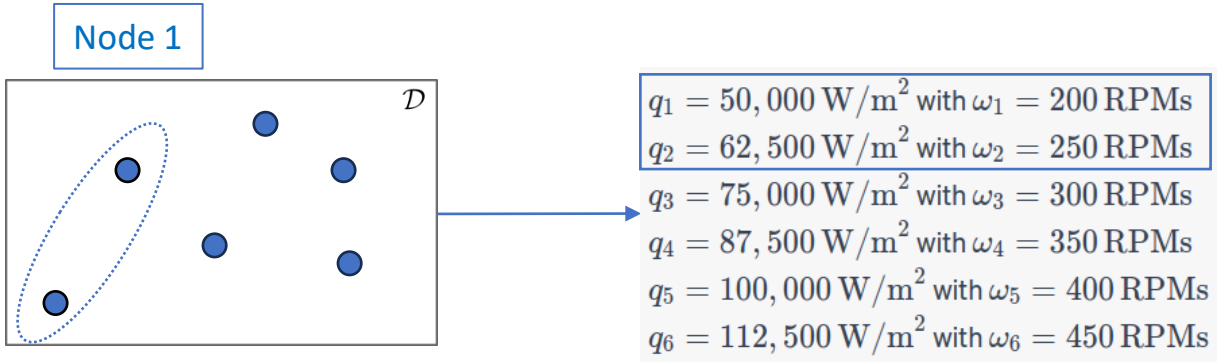
$q_1 = 50,000\,\mathrm{W/m^2}$ with $\omega_1 = 200\,\mathrm{RPMs}$

Convection-Diffusion Parameters

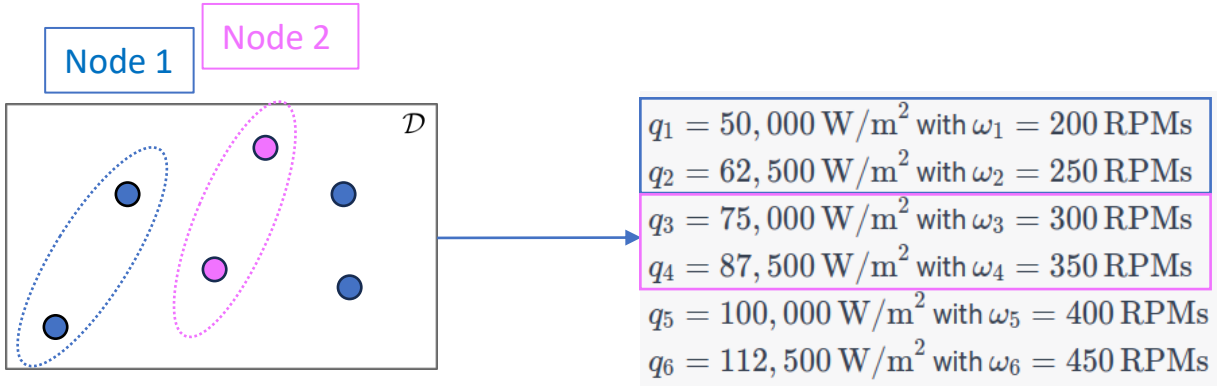


$$
\begin{aligned}
q_1 &= 50,000\,\mathrm{W/m^2} \text{ with } \omega_1 = 200\,\mathrm{RPMs} \\
q_2 &= 62,500\,\mathrm{W/m^2} \text{ with } \omega_2 = 250\,\mathrm{RPMs} \\
q_3 &= 75,000\,\mathrm{W/m^2} \text{ with } \omega_3 = 300\,\mathrm{RPMs} \\
q_4 &= 87,500\,\mathrm{W/m^2} \text{ with } \omega_4 = 350\,\mathrm{RPMs} \\
q_5 &= 100,000\,\mathrm{W/m^2} \text{ with } \omega_5 = 400\,\mathrm{RPMs} \\
q_6 &= 112,500\,\mathrm{W/m^2} \text{ with } \omega_6 = 450\,\mathrm{RPMs}
\end{aligned}
$$

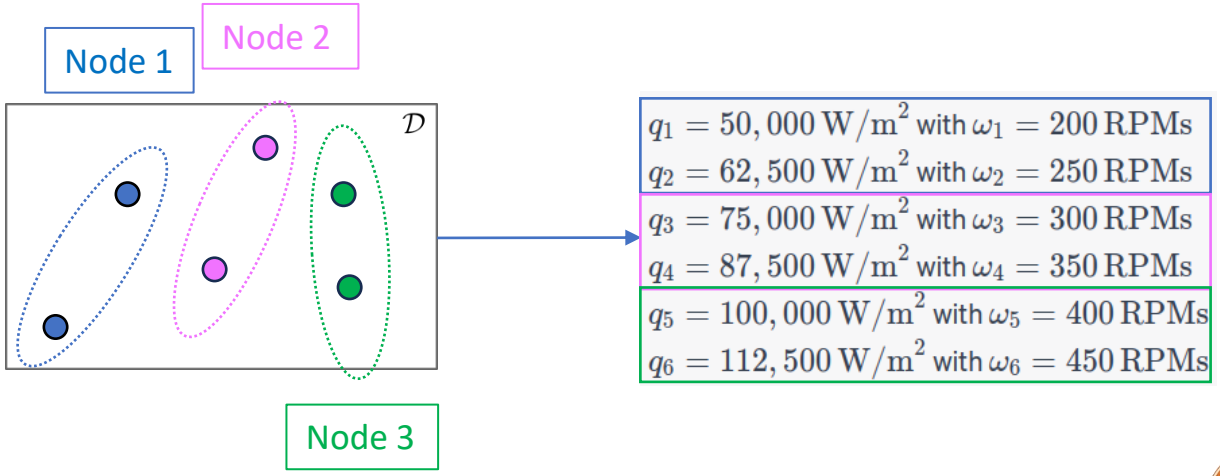
Convection-Diffusion Parameters: Parallel Simulations



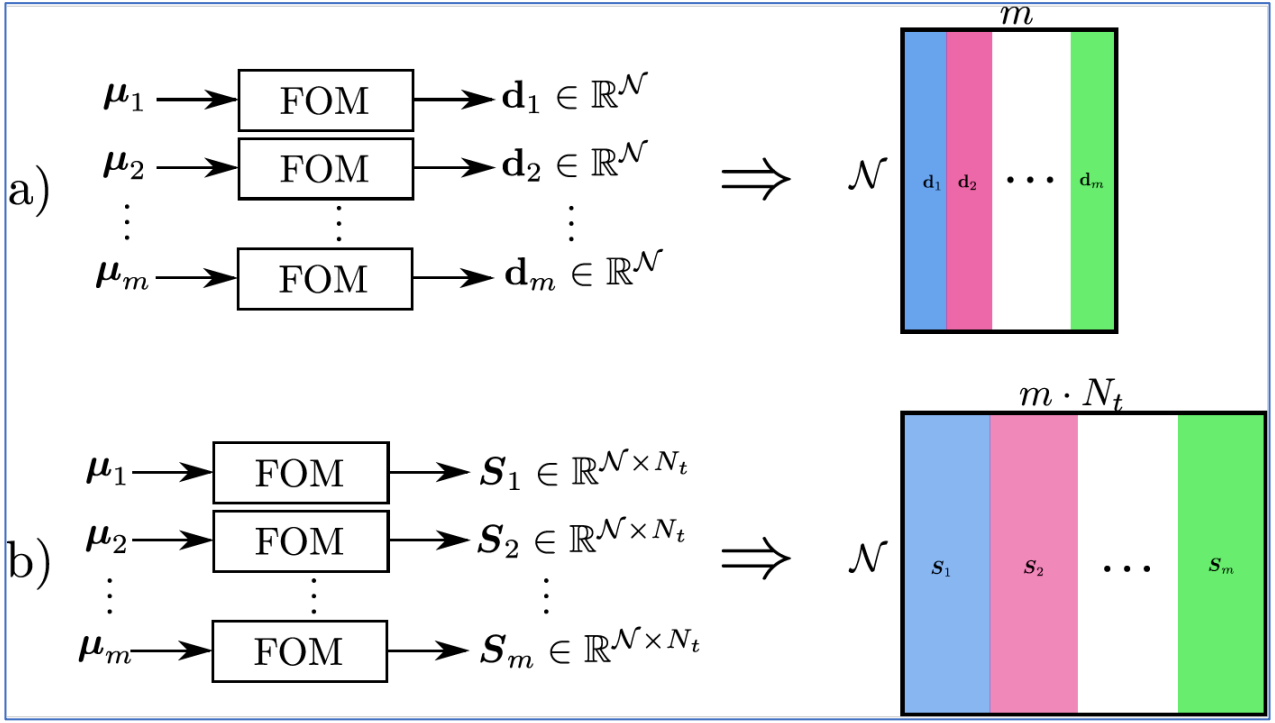
Convection-Diffusion Parameters: Parallel Simulations



Convection-Diffusion Parameters: Parallel Simulations

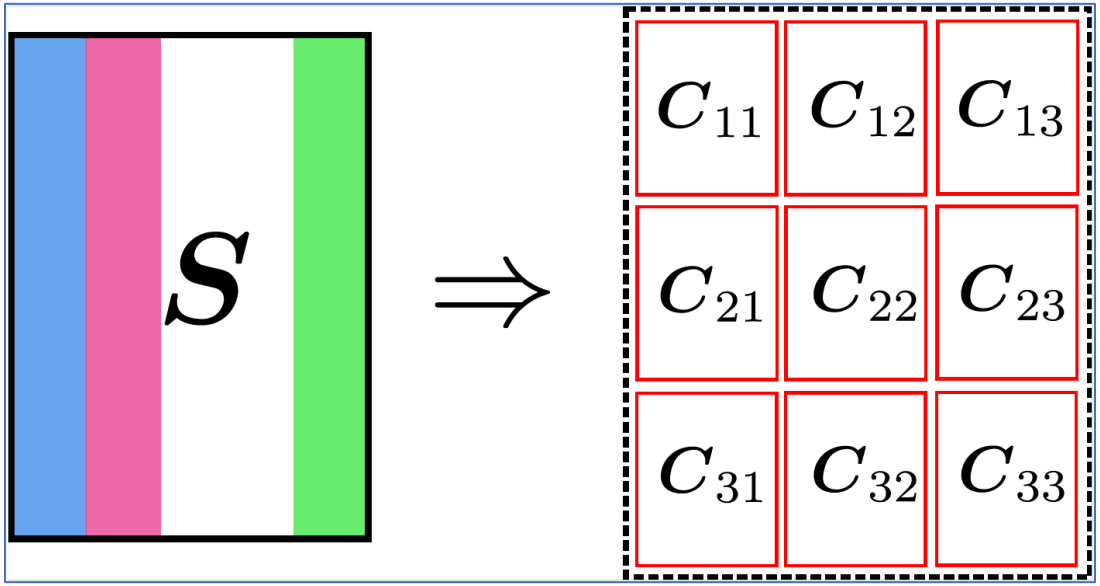


High-Performance Computing: Parallel Storage



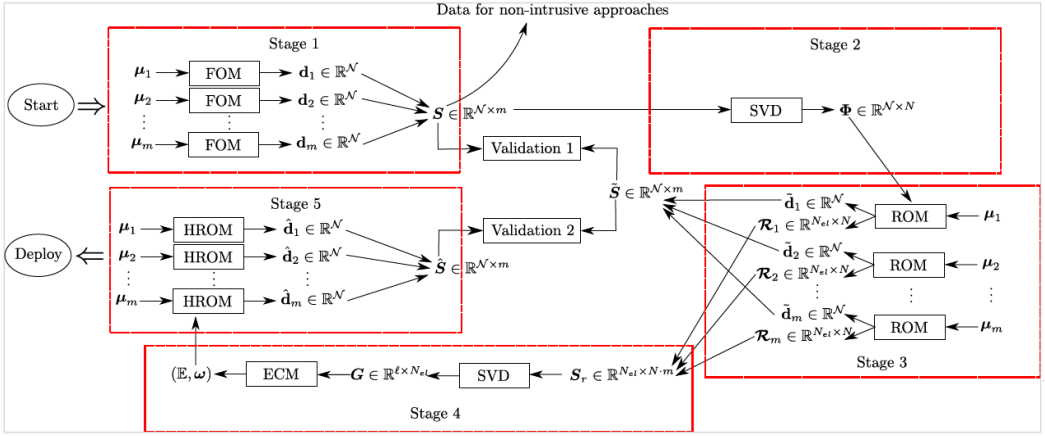
dislib | distributed computing library

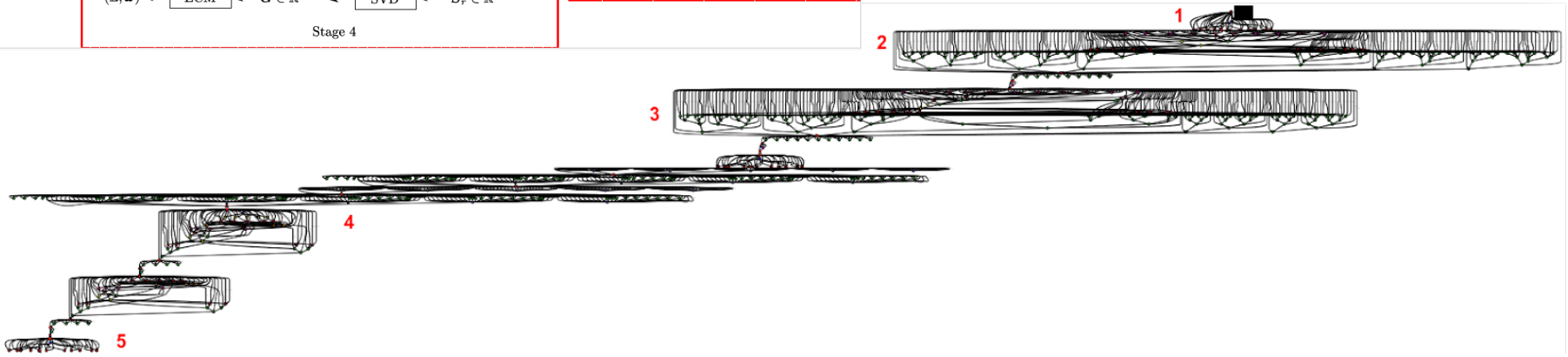
High-Performance Computing: Parallel SVD (Tall and Skinny QR)



dislib | distributed computing library

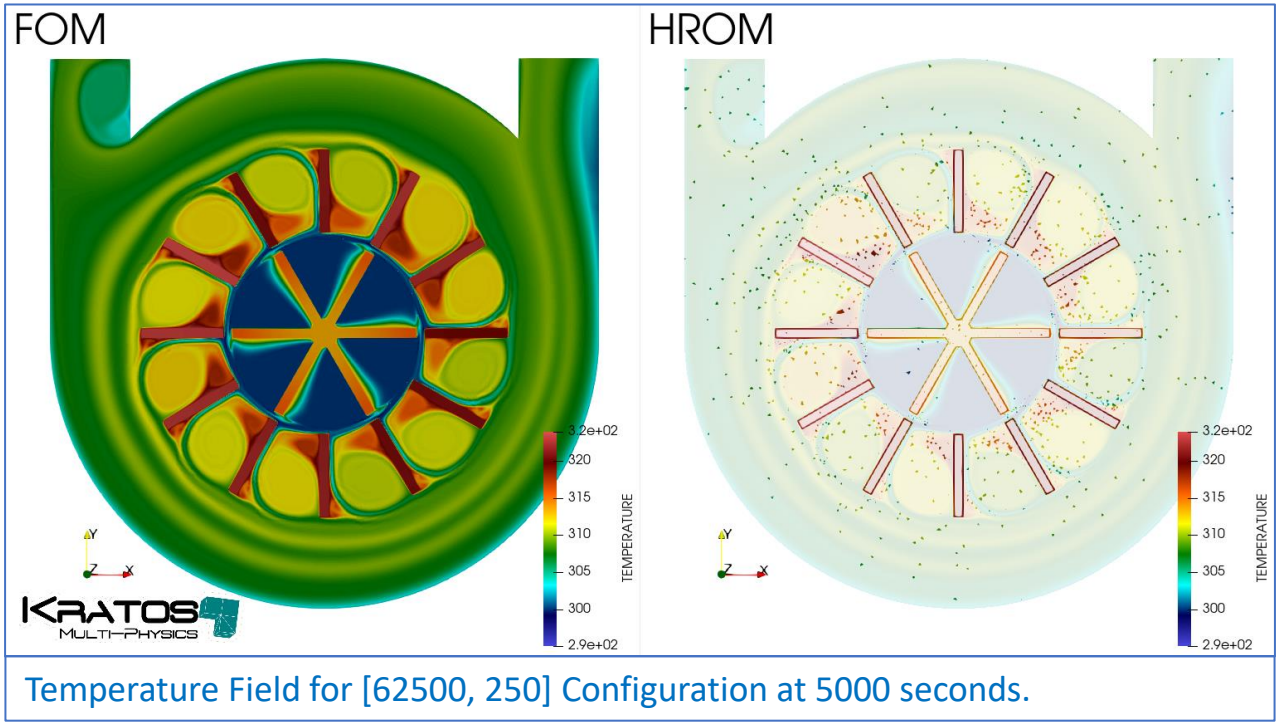
High-Performance Computing: Graph





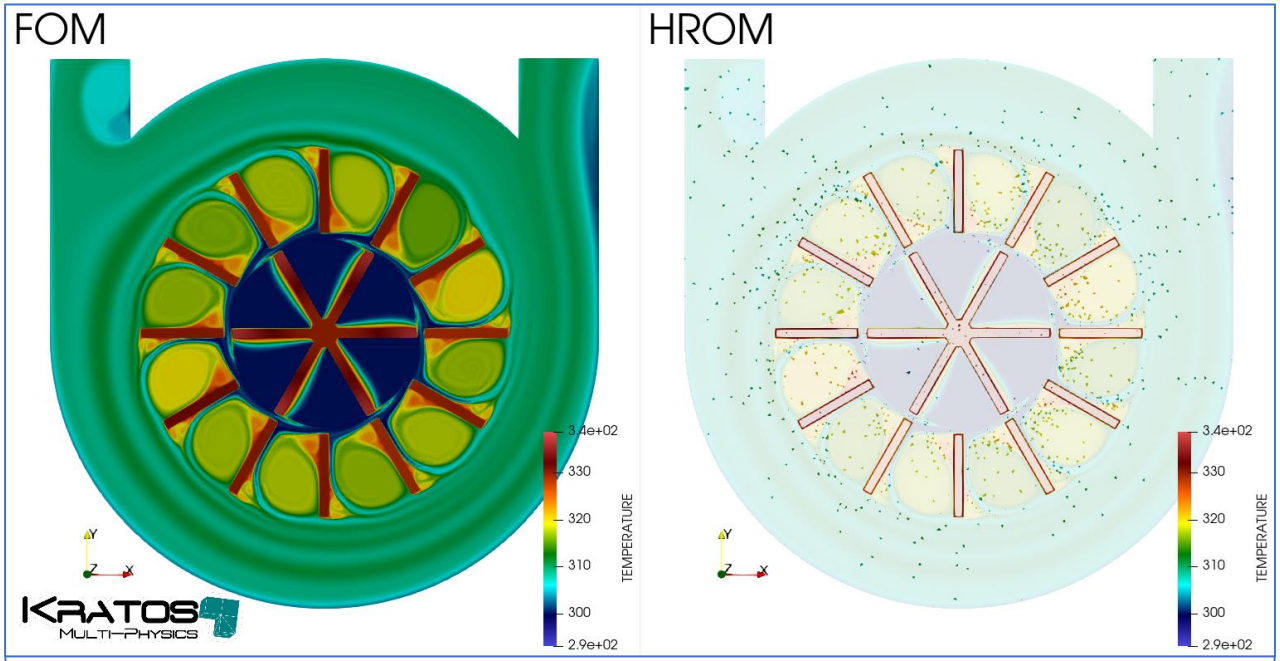
Graph of the Workflow execution using the Co-Simulation Example

High-Performance Computing: Demo Results



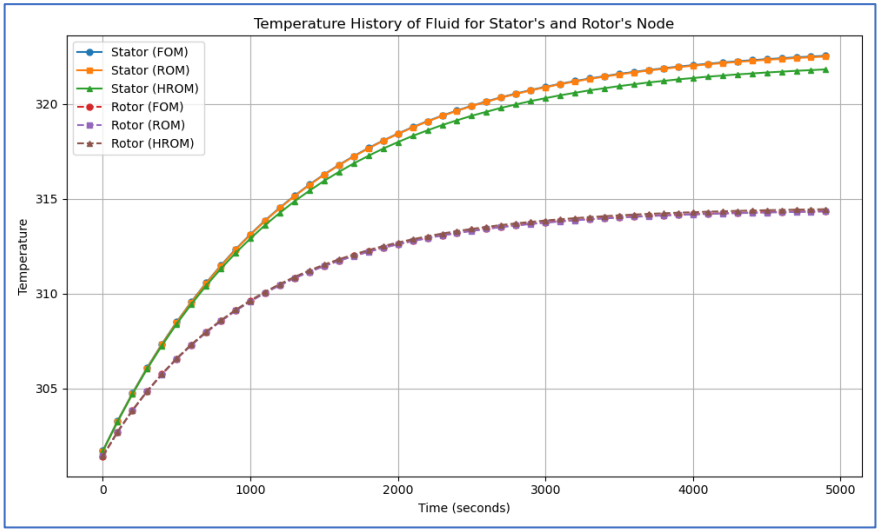
Temperature Field for [62500, 250] Configuration at 5000 seconds.

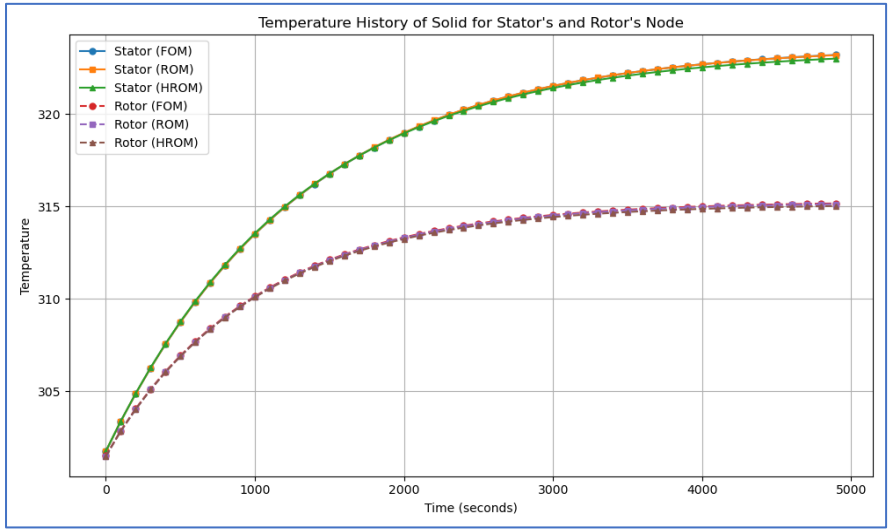
High-Performance Computing: Demo Results



Temperature Field for [100000, 400] Configuration at 5000 seconds.

High-Performance Computing: Demo Results





Fluid and Solid Control Point Response for [62500, 250] Configuration at 5000 seconds.

High-Performance Computing: Speed-ups

Model	Construction (Solid)	Solving + Projection (Solid)	Total (Solid)	Construction (Fluid)	Solving + Projection (Fluid)	Total (Fluid)
ROM	0.72	22.25	3.05	0.84	62.69	6.41
HROM	1.13	24.19	4.56	2.01	59.96	13.52
HHROM	1.54	38.52	6.37	5.57	260.80	40.39

Speed-ups for Configuration 1 (RPM of 250 and heat flux of 62500 W/m^3)

Model	Construction (Solid)	Solving + Projection (Solid)	Total (Solid)	Construction (Fluid)	Solving + Projection (Fluid)	Total (Fluid)
ROM	0.71	21.74	3.03	0.86	64.42	6.61
HROM	1.15	24.56	4.72	2.10	62.13	14.10
HHROM	1.56	39.33	6.50	5.84	275.32	42.27

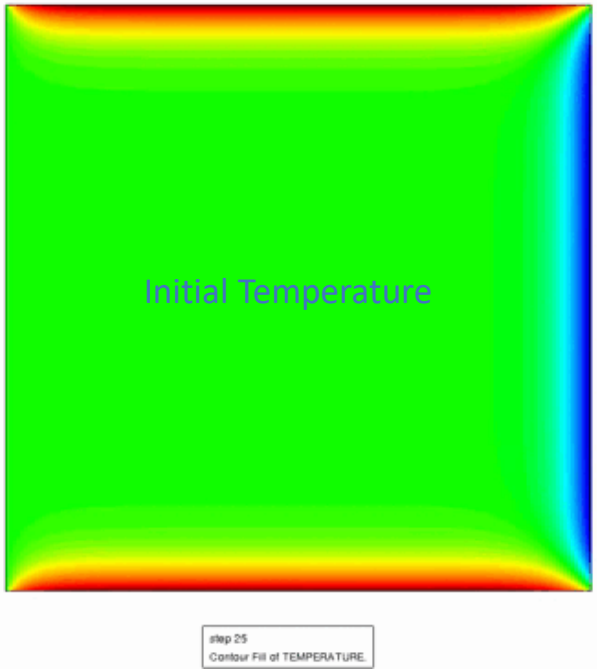
Speed-ups for Configuration 2 (RPM of 400 and heat flux of 100000 W/m^3)

HPCWAAS SMALL DEMO

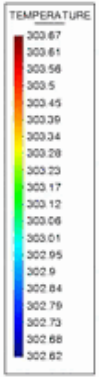
HPCWaaS: Small Demo

External heat flux:
2000 W/m^2

Fixed Temperature:
303.15 K

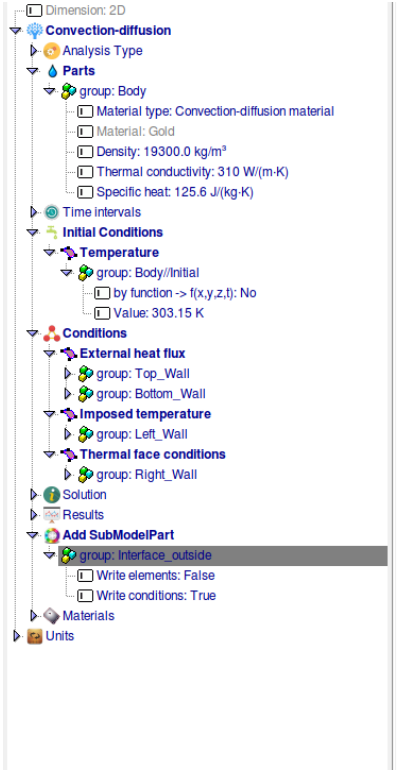


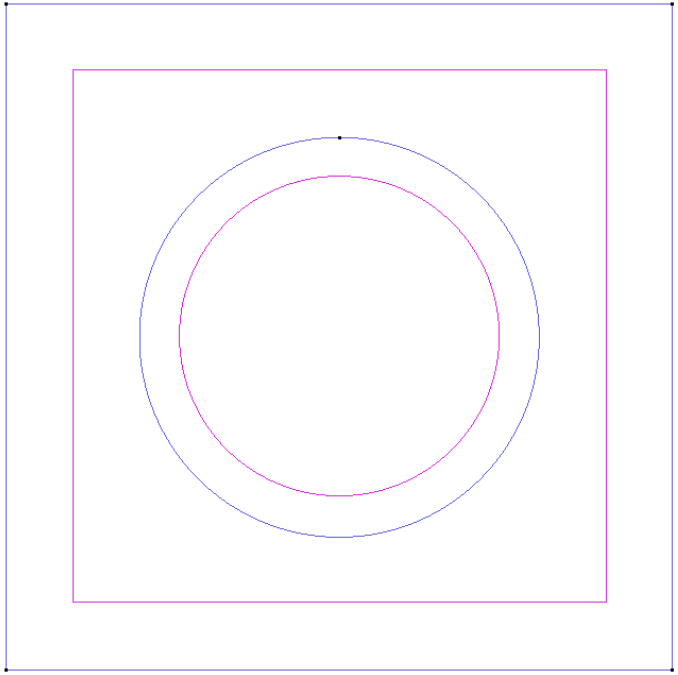
Ambient Radiation:
Emissivity = 0.8 at
283.15 K
Ambient Convection:
Convection
Coefficient = 100
W/m^2/K at 283.15 K



External heat flux:
2000 W/m^2

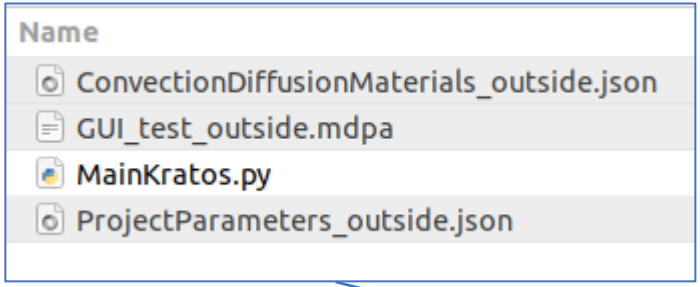
HPCWaaS: Coupling Different Domains



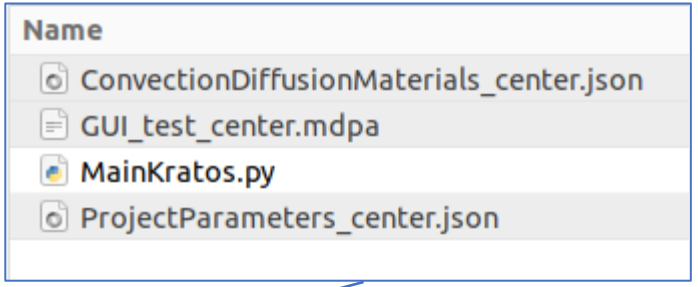


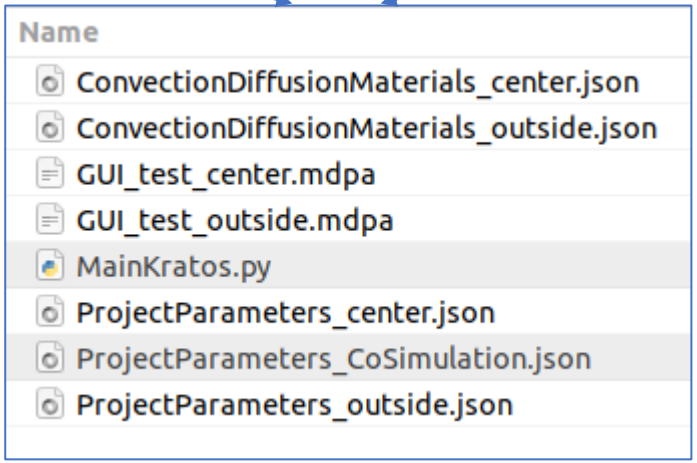
HPCWaaS: Coupling Different Domains

Outer part:

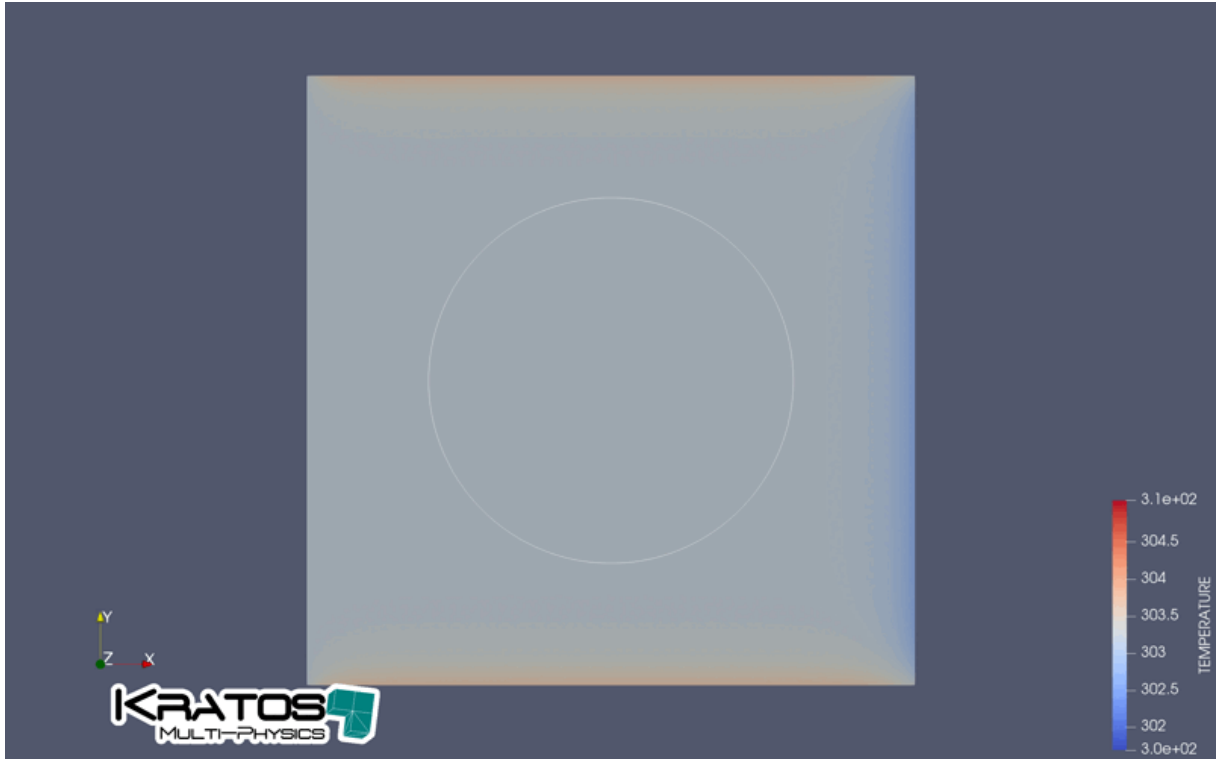


Central part:





HPCWaaS: Coupling Different Domains



Coupled ROM: Temperature
Contour Field

Acknowledgements

The authors acknowledge financial support from the Spanish Ministry of Economy and Competitiveness, through the "Severo Ochoa Programme for Centres of Excellence in R&D" (CEX2018-000797-S)". We acknowledge financial support from the Generalitat de Catalunya through the FI_SDUR-2021 grant, which provided funding for the predoctoral training of Sebastian Ares de Parga Regalado. This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement Nos 955558 and 956104. The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Spain, Germany, France, Italy, Poland, Switzerland, and Norway. Spanish Ministry of Science and Innovation (MICINN) supports the project through the project PCI2021-121945. This publication is part of the R&D project PCI2021-121944, financed by MCIN/AEI/10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR".

eFlows4HPC

Enabling dynamic and Intelligent workflows
in the future EuroHPC ecosystem

www.eFlows4HPC.eu

@eFlows4HPC eFlows4HPC Project

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Germany, France, Italy, Poland, Switzerland, Norway.