Reduced order methods for optimal flow control: FEniCS-based applications

Maria Strazzullo

mathLab, Mathematics Area, SISSA International School for Advanced Studies, Trieste, Italy

26 March 2021

FEniCS 2021

Motivations

Starting Point

Reduced Order Methods (ROMs) for parameterized Optimal Flow Control Problems (OFCPs) in environmental sciences

PDEs-based

Several simulations for different values of physical and/or geometrical parameters

(Uncertainty Quantification, Parameter Estimation...)

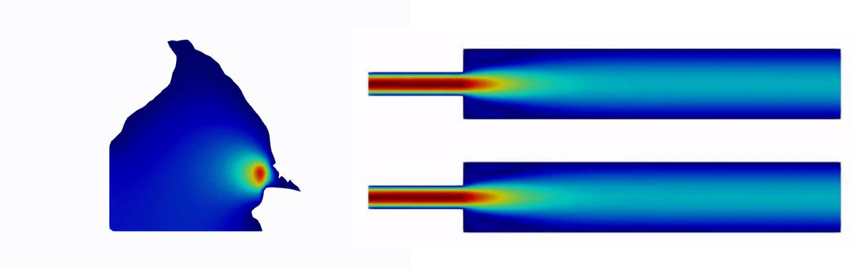
OFCPs

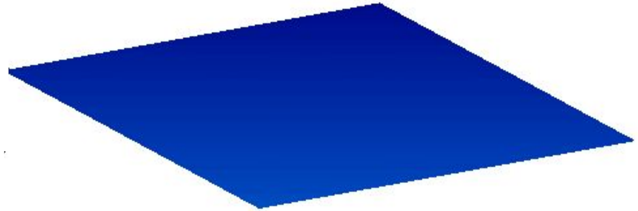
DATA-based

Scattered

expensive and difficult to collect

complex to interpret





By Dan Copsey (DanCopsey1 at English Wikipedia) — Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1692219

Motivations

Starting Point

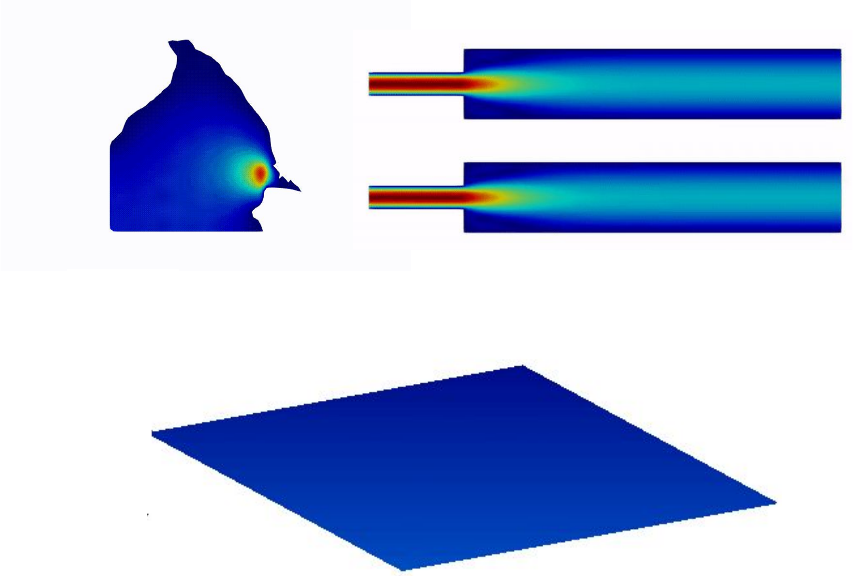
Reduced Order Methods (ROMs) for parameterized Optimal Flow Control Problems (OFCPs)
in environmental sciences

ROMs

PDEs-based
Several simulations for different values of physical and/or geometrical parameters
(Uncertainty Quantification, Parameter Estimation...)

OFCPs

DATA-based
Scattered
expensive and difficult to collect
complex to interpret



The Optimal Control Pipeline

PDEs
Data μ
Control

Cost
Functional
Minimization

Optimal
Solution

Solution
Interpretation

μ changes

ROM framework
time-dependent nonlinear problems
(POD, RB)

General Formulation
easy to adapt to several PDEs

Advanced Applications
(UQ, Bifurcations)

multiphenics RBniCS

https://mathlab.sissa.it/multiphenics
https://www.rbnicsproject.org/

Outline:

1. Problem Formulation
2. Ideas behind ROMs
3. Advanced Applications
    - Shallow Waters Equations (SWE)
    - Steering Bifurcations
    - Uncertainty Quantification (UQ)

Advisors:
Prof. Gianluigi Rozza (SISSA)
Dr. Francesco Ballarin (UNICATT)

Collaborators:
Prof. Rob Stevenson (UvA)
Giuseppe Carere (UvA)
Dr. Federico Pichi (SISSA/EPFL)

Problem Formulation

$$\min_{(y,u)\in\mathcal{Y}\times\mathcal{U}} J(y,u;\boldsymbol{\mu}) := \min_{(y,u)\in\mathcal{Y}\times\mathcal{U}} \frac{1}{2}\int_0^T \|y(\boldsymbol{\mu})-y_d(\boldsymbol{\mu})\|^2_{Y(\Omega_{\mathrm{OBS}})}\,dt + \int_0^T \frac{\alpha}{2}\|u(\boldsymbol{\mu})\|^2_{U(\Omega_u)}\,dt$$

such that $\boxed{\mathcal{E}(y,u;\boldsymbol{\mu}) = 0}$

Linear

Non-linear

Time-dependent

$$(\mathrm{DIM} = 3\mathcal{N}) \begin{cases} D_y\mathcal{L}((y,u,p);y_{\mathrm{d}},\boldsymbol{\mu})[\omega] = 0 & \forall \omega \in \mathbb{Y}, \\ D_u\mathcal{L}((y,u,p);y_{\mathrm{d}},\boldsymbol{\mu})[\kappa] = 0 & \forall \kappa \in \mathbb{U}, \\ D_p\mathcal{L}((y,u,p);y_{\mathrm{d}},\boldsymbol{\mu})[\zeta] = 0 & \forall \zeta \in \mathbb{Y}. \end{cases}$$

PDEs
Graetz Flows (temperature)
Advection Diffusion (pollutant)
Shallow Waters (coastal management)
Navier-Stokes (haemodynamics)

Controls
Boundary, Part of the domain, the whole domain, forcing terms…
Control **changes** the usual behaviour of the solution.

Optimal Control is a very powerful mathematical tool

- more reliable solutions
- great versatility

However it is **costly** (based on the solution of three equations)

Problem Formulation

$$\min_{(y,u)\in\mathcal{Y}\times\mathcal{U}} J(y,u;\boldsymbol{\mu}) := \min_{(y,u)\in\mathcal{Y}\times\mathcal{U}} \frac{1}{2}\int_0^T \|y(\boldsymbol{\mu}) - y_d(\boldsymbol{\mu})\|^2_{Y(\Omega_{\text{OBS}})}\,dt + \int_0^T \frac{\alpha}{2}\|u(\boldsymbol{\mu})\|^2_{U(\Omega_u)}\,dt$$

such that $\boxed{\mathcal{E}(y,u;\boldsymbol{\mu}) = 0}$

- Linear
- Non-linear
- Time-dependent

$$(\text{DIM} = 3\mathcal{N}) \begin{cases} D_y\mathcal{L}((y,u,p);y_{\text{d}},\boldsymbol{\mu})[\omega] = 0 & \forall \omega \in \mathbb{Y}, \\ D_u\mathcal{L}((y,u,p);y_{\text{d}},\boldsymbol{\mu})[\kappa] = 0 & \forall \kappa \in \mathbb{U}, \\ D_p\mathcal{L}((y,u,p);y_{\text{d}},\boldsymbol{\mu})[\zeta] = 0 & \forall \zeta \in \mathbb{Y}. \end{cases}$$

PDEs
Graetz Flows (temperature)
Advection Diffusion (pollutant)
Shallow Waters (coastal management)
Navier-Stokes (haemodynamics)

Controls
Boundary, Part of the domain, the whole domain, forcing terms…
Control **changes** the usual behaviour of the solution.

Optimal Control is a very powerful mathematical tool

- more reliable solution
- great versatility

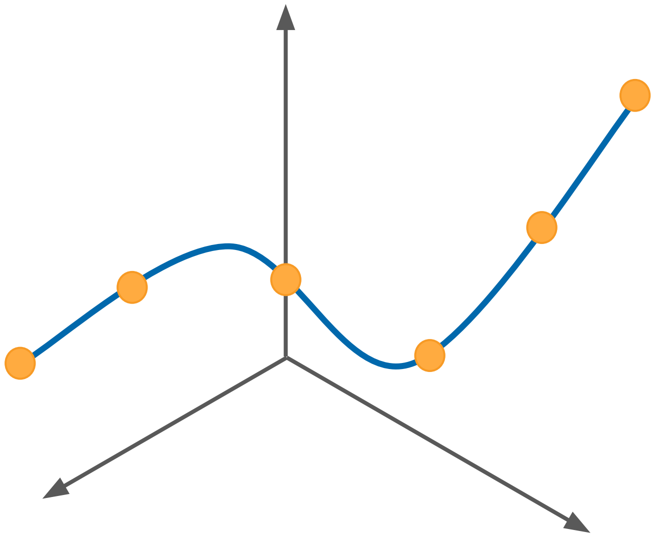
However it is costly (based on the solution of three equations) **most of all in a parametrized setting**

ROMs

Exploit the parametric structure of the problem

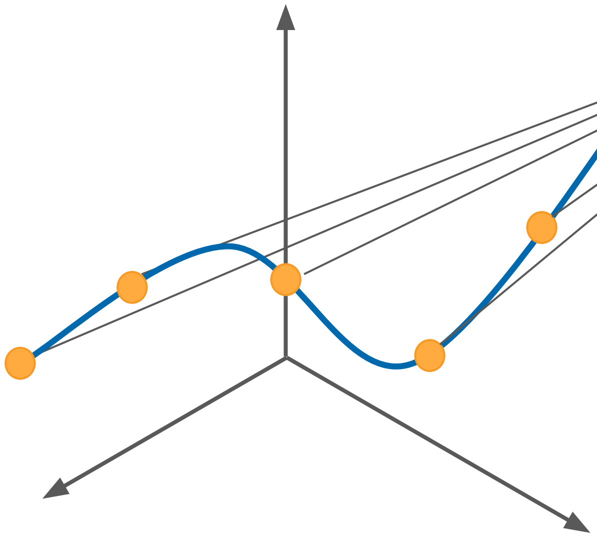
Ideas behind ROMs

Explore the solutions **snapshots** and how they change with respect to the parameter



Ideas behind ROMs

Explore the solutions **snapshots** and how they change with respect to the parameter



Building phase
POD (SVD, PCA)
RB (Adaptive technique)

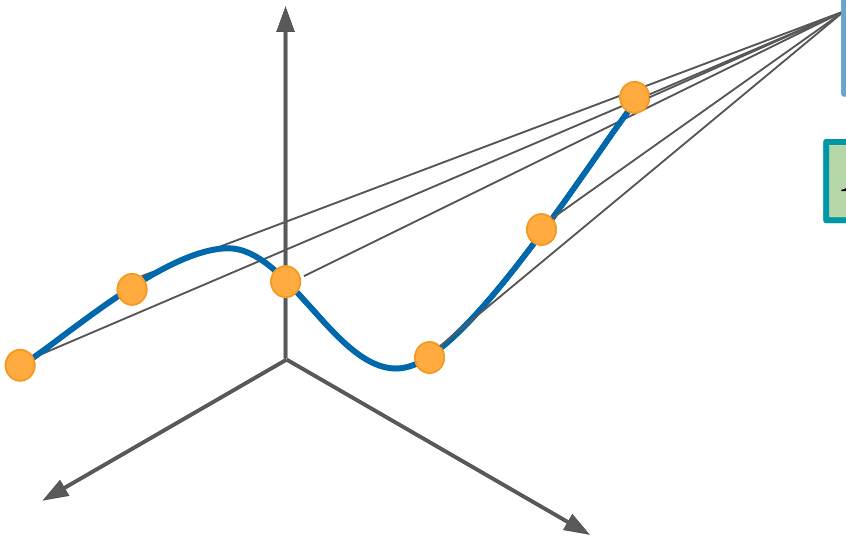
$\longrightarrow$ Z^T

$$A_N = Z^T A Z$$

use μ A_N for each new parameter evaluation

Ideas behind ROMs

Explore the solutions **snapshots** and how they change with respect to the parameter



Building phase
POD (SVD, PCA)
RB (Adaptive technique)

$\rightarrow$ Z^T

$$A_N = Z^T \; A \; Z$$

Saving computational time

use μ A_N for each new parameter evaluation

How do you use RBniCS?

Example of Steady Coercive State Equation

```
class EllipticOptimalControl(EllipticOptimalControlProblem):

    # Default initialization of members
    def __init__(self, V, **kwargs):
        # Call the standard initialization
        EllipticOptimalControlProblem.__init__(self, V, **kwargs)
        # ... and also store FEniCS data structures for assembly
        assert "subdomains" in kwargs
        assert "boundaries" in kwargs
        self.subdomains, self.boundaries = kwargs["subdomains"], kwargs["boundaries"]
        yup = TrialFunction(V)
        (self.y, self.u, self.p) = split(yup)
        zvq = TestFunction(V)
        (self.z, self.v, self.q) = split(zvq)
        self.dx = Measure("dx")(subdomain_data=subdomains)
        self.ds = Measure("ds")(subdomain_data=boundaries)
        # Regularization coefficient
        self.alpha = 0.01
        # Desired state
        self.y_d = Constant(1.0)
        # Customize linear solver parameters
        self._linear_solver_parameters.update({
            "linear_solver": "mumps"
        })
```

In the class you define the parameters multiplied by the various forms defined as in FEniCS

Simple modifications for more complicated problems or to use other algorithms

```
# 1. Read the mesh for this problem
mesh = Mesh("data/mesh1.xml")
subdomains = MeshFunction("size_t", mesh, "data/mesh1_physical_region.xml")
boundaries = MeshFunction("size_t", mesh, "data/mesh1_facet_region.xml")

# 2. Create Finite Element space (Lagrange P1)
scalar_element = FiniteElement("Lagrange", mesh.ufl_cell(), 1)
element = MixedElement(scalar_element, scalar_element, scalar_element)
V = FunctionSpace(mesh, element, components=["y", "u", "p"])

# 3. Allocate an object of the EllipticOptimalControl class
elliptic_optimal_control = EllipticOptimalControl(V, subdomains=subdomains, boundaries=boundaries)
mu_range = [(1.0, 3.5), (0.5, 2.5)]
elliptic_optimal_control.set_mu_range(mu_range)

# 4. Prepare reduction with a reduced basis method
pod_galerkin_method = PODGalerkin(elliptic_optimal_control)
pod_galerkin_method.set_Nmax(10)

# 5. Perform the offline phase
lifting_mu = (1.0, 1.0)
elliptic_optimal_control.set_mu(lifting_mu)
pod_galerkin_method.initialize_training_set(100)
reduced_elliptic_optimal_control = pod_galerkin_method.offline()

# 6. Perform an online solve
online_mu = (2.0, 2.)
reduced_elliptic_optimal_control.set_mu(online_mu)
reduced_elliptic_optimal_control.solve()
reduced_elliptic_optimal_control.export_solution(filename="online_solution")
print("Reduced output for mu =", online_mu, "is", reduced_elliptic_optimal_control.compute_output())
```

How do you use RBniCS?

Example of Steady Coercive State Equation

```
class EllipticOptimalControl(EllipticOptimalControlProblem):

    # Default initialization of members
    def __init__(self, V, **kwargs):
        # Call the standard initialization
        EllipticOptimalControlProblem.__init__(self, V, **kwargs)
        # ... and also store FEniCS data structures for assembly
        assert "subdomains" in kwargs
        assert "boundaries" in kwargs
        self.subdomains, self.boundaries = kwargs["subdomains"], kwargs["boundaries"]
        yup = TrialFunction(V)
        (self.y, self.u, self.p) = split(yup)
        zvq = TestFunction(V)
        (self.z, self.v, self.q) = split(zvq)
        self.dx = Measure("dx")(subdomain_data=subdomains)
        self.ds = Measure("ds")(subdomain_data=boundaries)
        # Regularization coefficient
        self.alpha = 0.01
        # Desired state
        self.y_d = Constant(1.0)
        # Customize linear solver parameters
        self._linear_solver_parameters.update({
            "linear_solver": "mumps"
        })
```

In the class you define the parameters multiplied by the various forms defined as in FEniCS

Simple modifications for more complicated problems or to use other algorithms

```
# 1. Read the mesh for this problem
mesh = Mesh("data/mesh1.xml")
subdomains = MeshFunction("size_t", mesh, "data/mesh1_physical_region.xml")
boundaries = MeshFunction("size_t", mesh, "data/mesh1_facet_region.xml")

# 2. Create Finite Element space (Lagrange P1)
scalar_element = FiniteElement("Lagrange", mesh.ufl_cell(), 1)
element = MixedElement(scalar_element, scalar_element, scalar_element)
V = FunctionSpace(mesh, element, components=["y", "u", "p"])

# 3. Allocate an object of the EllipticOptimalControl class
elliptic_optimal_control = EllipticOptimalControl(V, subdomains=subdomains, boundaries=boundaries)
mu_range = [(1.0, 3.5), (0.5, 2.5)]
elliptic_optimal_control.set_mu_range(mu_range)
```

```
# 4. Prepare reduction with a reduced basis method
pod_galerkin_method = PODGalerkin(elliptic_optimal_control)
pod_galerkin_method.set_Nmax(10)

# 5. Perform the offline phase
lifting_mu = (1.0, 1.0)
elliptic_optimal_control.set_mu(lifting_mu)
pod_galerkin_method.initialize_training_set(100)
reduced_elliptic_optimal_control = pod_galerkin_method.offline()
```

```
# 6. Perform an online solve
online_mu = (2.0, 2.)
reduced_elliptic_optimal_control.set_mu(online_mu)
reduced_elliptic_optimal_control.solve()
reduced_elliptic_optimal_control.export_solution(filename="online_solution")
print("Reduced output for mu =", online_mu, "is", reduced_elliptic_optimal_control.compute_output())
```

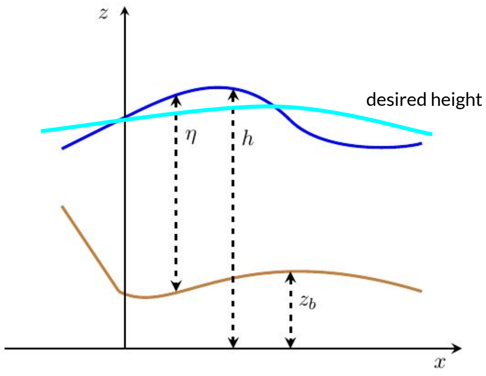
Coastal Height Management

Minimisation of

$$\frac{1}{2}\int_0^T \int_\Omega (h - h_d(\mu_3))^2 dxdt + \frac{1}{2}\int_0^T \int_\Omega (\boldsymbol{v} - \boldsymbol{v}_d(\mu_3))^2 dxdt + \frac{\alpha}{2}\int_0^T \int_\Omega \boldsymbol{u}^2 dxdt$$

constrained to

$$\begin{aligned}
&\boldsymbol{v}_t + \mu_1 \Delta \boldsymbol{v} + \mu_2 (\boldsymbol{v}\cdot\nabla)\boldsymbol{v} + g\nabla\eta - \boldsymbol{u} = 0 && \text{in } \Omega \times [0,T],\\
&h_t + \operatorname{div}(\eta \boldsymbol{v}) = 0 && \text{in } \Omega \times [0,T],\\
&\boldsymbol{v} = \boldsymbol{v}_0 && \text{on } \Omega \times \{0\},\\
&h = h_0 && \text{on } \Omega \times \{0\},\\
&\boldsymbol{v} = \boldsymbol{0} && \text{on } \partial\Omega \times [0,T].
\end{aligned}$$

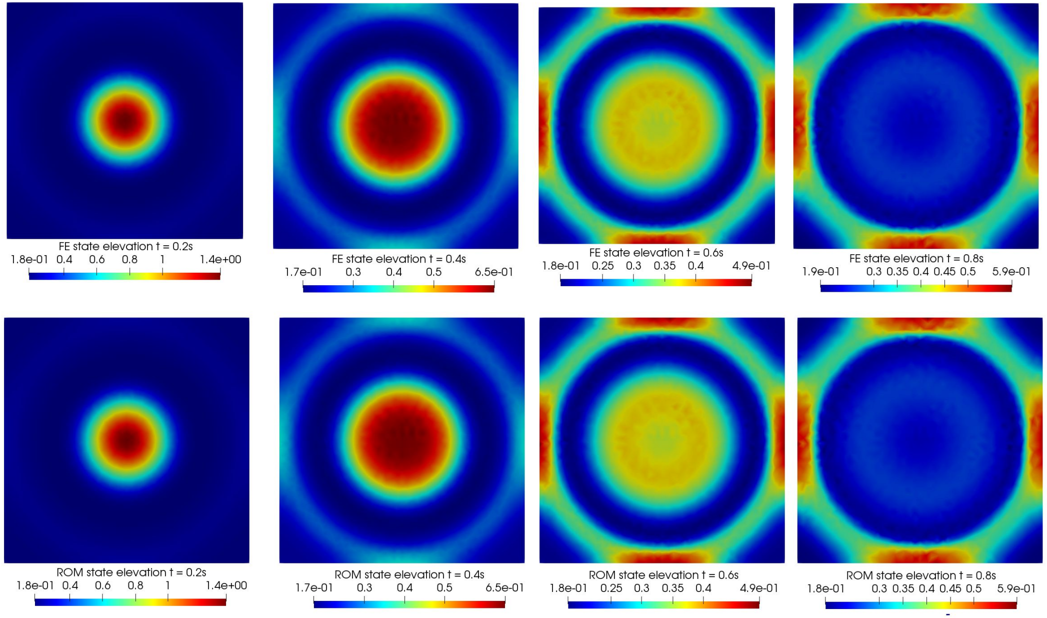


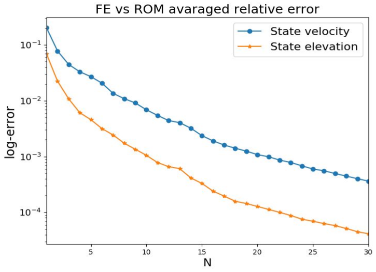
GOAL: recover parametrized desired height and velocity profiles with distributed control

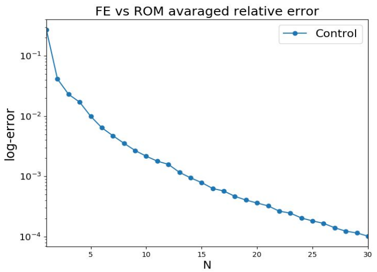
$$\begin{aligned}
[0,T] &= [0, 0.8],\\
\Omega &= [0,10]\times[0,10],\\
\alpha &= 10^{-5},\\
\Delta t &= 0.1.
\end{aligned}$$

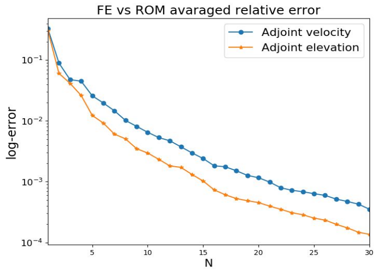
Straightforward implementation in RBniCS with standard modifications of Nonlinear problems classes

Coastal Height Management









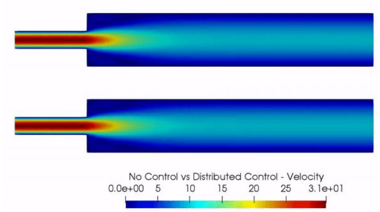
[Strazzullo, Ballarin, Rozza, POD-Galerkin Model Order Reduction for Parametrized Nonlinear Time Dependent Optimal Flow Control: an Application to Shallow Water Equations. *Submitted*, 2021]

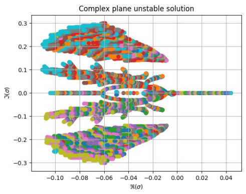
Errors ∼ 1e-4, Speedup ∼ 30
ROM vs FE dim = 270 vs 94’016.

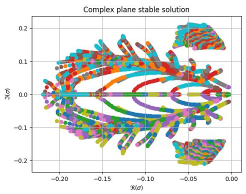
Advanced Applications

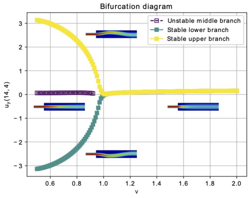
Optimal control + Bifurcations in Navier-Stokes:

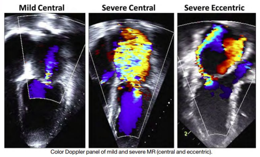
- Does optimal control affect the system?
- Does optimal control change stability?
- Does it change the stability analysis?
- What is the most “natural” configuration?



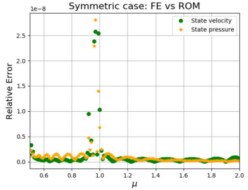






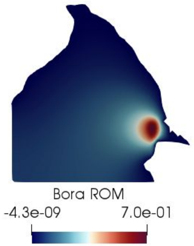


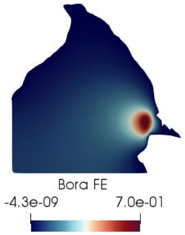
credits: American Society of Echocardiography Guideline Valvular Regurgitation

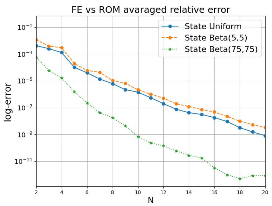


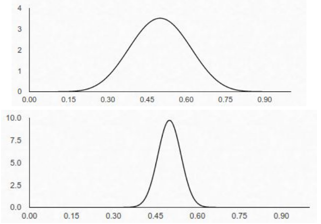
Optimal control + Uncertainty Quantification:

- Input-output relation?
- **Weighted**-POD
- More knowledge is better (less basis need)









[Pichi, Strazzullo, Ballarin, Rozza, Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier-Stokes equations with model order reduction. *Submitted, 2020*]

[Carere, Strazzullo, Ballarin, Rozza, Stevenson, Weighted POD-reduction for parametrized PDE-constrained Optimal Control Problems with random inputs and its applications to environmental sciences *Submitted, 2021*]

Conclusions

ROMs and OCPs

- Great mathematical tool to reach a desired configuration
- Expensive to solve
- RBniCS and multiphenics application to OFCPs
  - great versatility
  - simple to code

https://www.rbnicsproject.org/

Applications

- POD for nonlinear time-dependent problems (SWEs)
- OFCPs and bifurcations
- OFCPs and UQ

Acknowledgements

European Union Funding for Research and Innovation -- Horizon 2020 Program -- in the framework of European Research Council Executive Agency: Consolidator Grant H2020 ERC CoG 2015 AROMA-CFD project 681447 ``Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics''.

Thank you for your attention!