Proper orthogonal decomposition and discrete empirical interpolation in CFD applications

M. Isoz[a]

[a] Institute of Thermomechanics, Academy of Sciences of the Czech Republic

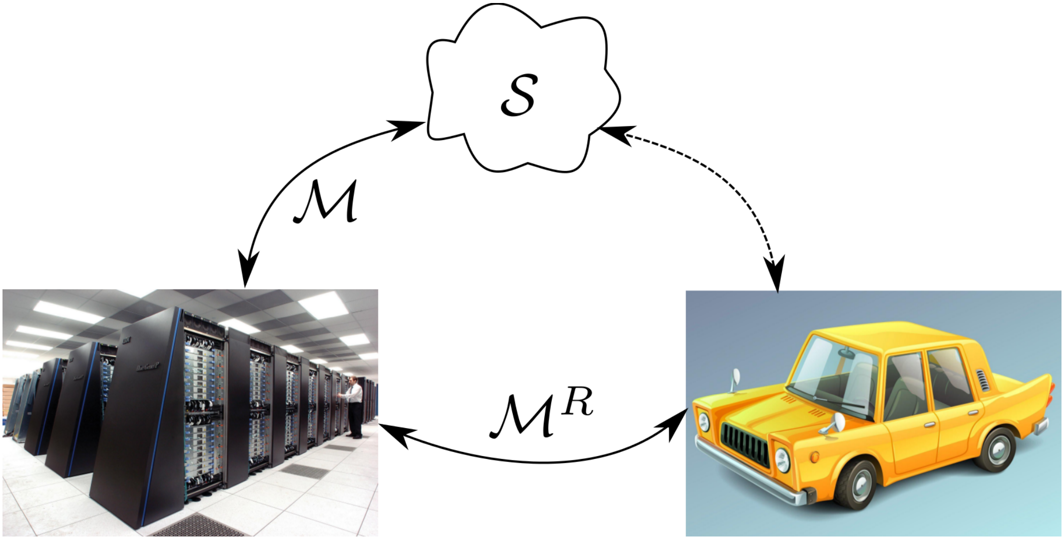
June 17, 2017

COMPDYN 2017

Introduction

Research motivation

Reducing the computational cost of modeling of complex systems



Original system

$$\dot{y} = Ay + f(t, y), \quad y(t) \in \mathbb{R}^m, \quad y(0) = y_0,\, t \in [0, T],$$

system matrix ... $A \in \mathbb{R}^{m \times m}$,

nonlinearities ... $f(t, y) \in \mathbb{R}^m$

Reduced-order system

$$\dot{\eta}^\ell = A^\ell \eta^\ell + f^\ell(t, \eta^\ell), \quad \eta^\ell(t) \in \mathbb{R}^\ell, \quad \eta^\ell(0) = \eta_0^\ell,\, t \in [0, T],$$

system matrix ... $A^\ell \in \mathbb{R}^{\ell \times \ell}$,

nonlinearities ... $f^\ell(t, \eta^\ell) \in \mathbb{R}^\ell$

gain ... $\ell \ll m$

Proper orthogonal decomposition & Discrete empirical interpolation method

Introduce the Galerking ansatz and Fourier modes

- Prerequisities:

$$\dot{y} = Ay + f(t,y), \quad y(t) \in \mathbb{R}^m, \quad y(0) = y_0,\, t \in [0,T]$$

$$y(t) \in V = \mathrm{span}\{\psi_j\}_{j=1}^{d} \quad \forall t \in [0,T]$$

$\Psi = \{\psi_j\}_{j=1}^{d}$ . . . orthonormal basis

$$y(t) = \sum_{j=1}^{d} \langle y(t), \psi_j \rangle_W \, \psi_j,\, \forall t \in [0,T], \quad W \ldots \text{appropriate weights}$$

- Ansatz for Galerkin projection, $\ell < d$

$$y^\ell(t) := \sum_{j=1}^{\ell} \langle y^\ell(t), \psi_j \rangle_W \, \psi_j,\, \forall t \in [0,T], \quad \eta_j^\ell(t) := \langle y^\ell(t), \psi_j \rangle_W$$

- Put the above together, !! $\psi_j \in \mathbb{R}^m$, $j = 1, \ldots, \ell$, $m > \ell$!!

$$\begin{aligned} \textstyle\sum_{j=1}^{\ell} \dot{\eta}_j^\ell \psi_j &= \textstyle\sum_{j=1}^{\ell} \eta_j^\ell A \psi_j + f(t, y^\ell(t)), \quad t \in (0,T) \\ y_0 &= \textstyle\sum_{j=1}^{\ell} \eta_j^\ell(0) \psi_j \end{aligned}$$

Introduce the reduced-order model

- Assume, that the above holds after projection on $V^\ell = \mathrm{span}\{\psi_j\}_{j=1}^{\ell}$, remember that $\langle \psi_j, \psi_i \rangle_W = \delta_{ij}$ and write,

$$\dot{\eta}_i^\ell = \sum_{j=1}^{\ell} \eta_j^\ell \langle A\psi_j, \psi_i \rangle_W + \langle f(t, y^\ell), \psi_i \rangle_W, \quad 1 \leq i \leq l \text{ and } t \in (0, T]$$

- Define the matrix $A^\ell = (a_{ij}^\ell) \in \mathbb{R}^{l \times l}$ with $a_{ij}^\ell = \langle A\psi_j, \psi_i \rangle_W$
- Define the vector valued mapping $\eta^\ell = (\eta_1^\ell, \dots, \eta_l^\ell)^{\mathrm{T}} : [0, T] \to \mathbb{R}^\ell$
- Define the non-linearity $f^\ell = (f_1^\ell, \dots, f_l^\ell)^{\mathrm{T}} : [0, T] \to \mathbb{R}^\ell$, where

$$f_i^\ell(t, \eta) = \left\langle f\left(t, \sum_{j=1}^{\ell} \eta_j \psi_j\right), \psi_i \right\rangle_W$$

- Introduce the IC, $\eta^\ell(0) = \eta_0^\ell = (\langle y_0, \psi_1 \rangle_W, \dots, \langle y_0, \psi_1 \rangle_W)^{\mathrm{T}}$
- Write the ROM, $\dot{\eta}^\ell = A^\ell \eta^\ell + f^\ell(t, \eta^\ell)$, for $t \in (0, T]$, $\eta^\ell(0) = \eta_0^\ell$

Original system

$$\dot{y} = Ay + f(t, y), \quad y(t) \in \mathbb{R}^m, \quad y(0) = y_0,\, t \in [0, T],$$

Solution snapshots←Approximation obtained from FOM

$$\boldsymbol{S} = \left\{ \boldsymbol{y}_j = \boldsymbol{y}(t_j) = \mathrm{e}^{At_j}\boldsymbol{y}_0 + \int_0^{t_j} \mathrm{e}^{A(t_j - s)}\boldsymbol{b}(s, \boldsymbol{y}(s))\,\mathrm{d}s \right\}_{j=1}^{n} \approx \tilde{\boldsymbol{S}} \leftarrow \text{FOM}$$

Matrix of snapshots (tildes denoting approximate solutions are omitted)

$$Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n] \in \mathbb{R}^{m \times n}, \quad \mathrm{rank}(Y) = d \le \min\{m, n\}\,,$$

Goal

Approximate all the spatial coordinate vectors $\boldsymbol{y}_j$ of Y simultaneously by $\ell \leq d$ normalized vectors as well as possible.

(P)

$$\boxed{\begin{gathered}\max_{\tilde{\boldsymbol{\psi}}_1,\ldots,\tilde{\boldsymbol{\psi}}_\ell \in \mathbb{R}^m} \sum_{i=1}^{\ell} \sum_{j=1}^{n} \left| \langle \boldsymbol{y}_j, \tilde{\boldsymbol{\psi}}_i \rangle_{\mathbb{R}^m} \right|^2 \\ \text{subject to} \\ \langle \tilde{\boldsymbol{\psi}}_i, \tilde{\boldsymbol{\psi}}_j \rangle_{\mathbb{R}^m} = \delta_{ij} \quad \text{for} \quad 1 \leq i, j \leq \ell\,,\end{gathered}}$$

Fundamental theorem of Proper orthogonal decomposition

Let Y be a given matrix of snapshots. Also, let $Y = \Psi\Sigma\Phi^T$ be the singular value decomposition of Y, where $\Psi = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_m] \in \mathbb{R}^{m\times m}$ and $\Phi = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n] \in \mathbb{R}^{n\times n}$ are orthogonal matrices and the matrix Σ has the structure of

$$\Sigma = \begin{bmatrix} \mathrm{diag}(\sigma_1, \ldots, \sigma_d) & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{m\times n},$$

where $\sigma_1, \ldots, \sigma_d$ are the singular values of the matrix Y. Then, for any $\ell \in \{1, \ldots, d\}$ the solution to problem (**P**) is given by the singular vectors $\{\boldsymbol{\psi}_i\}_{i=1}^{\ell}$, i.e. by the first ℓ columns of Ψ. Moreover,

$$\mathrm{argmax}(\mathbf{P}) = \sum_{i=1}^{\ell} \sigma^2 .$$

Proof

- Obtained via Lagrange framework
- Rather long and technical, can be found in literature (e.g. [**VolkweinBook**])

Algorithm 1 POD basis of rank ℓ with weighted inner product

Require: Snapshots $\{y_j\}_{j=1}^n$, POD rank $\ell \leq d$, symmetric positive-definite matrix of weights $W \in \mathbb{R}^{m\times m}$

1: Set $Y = [y_1, \ldots, y_n] \in \mathbb{R}^{m\times n}$;
2: Determine $\bar{Y} = W^{1/2} Y \in \mathbb{R}^{m\times n}$;
3: Compute SVD, $[\bar{\Psi}, \Sigma, \bar{\Phi}] = \mathrm{svd}(\bar{Y})$;
4: Set $\sigma = \mathrm{diag}(\Sigma)$;
5: Compute $\varepsilon(\ell) = \sum_{i=1}^{\ell} \sigma_i / \sum_{i=1}^{d} \sigma_i$;
6: Truncate $\bar{\Psi} \leftarrow [\bar{\psi}_1, \ldots, \bar{\psi}_l] \in \mathbb{R}^{m\times\ell}$;
7: Compute $\Psi = W^{-1/2} \bar{\Psi} \in \mathbb{R}^{m\times\ell}$;
8: **return** POD basis, Ψ, and ratio $\varepsilon(\ell)$

Notes:

- All the operations on W have to be cheap, including its inversion.
- Do not perform the full SVD, $\Sigma \in \mathbb{R}^{d\times d}$, $d = \mathrm{rank}(\bar{Y})$.

Deal with the non-linearities I

- Identify the problem,

$$f_i^\ell(t,\eta) = \left\langle f\left(t, \sum_{j=1}^{\ell} \eta_j \psi_j\right), \psi_i \right\rangle_W \quad \dots \sum_{j=1}^{\ell} \eta_j \psi_j \in \mathbb{R}^m \leftarrow \text{ FO}$$

- Approximate the non-linearities via the POD basis, Φ,

$$b(t) := f(t, \Psi\eta^\ell) \approx \sum_{k=1}^{p} \phi_k c_k(t) = \Phi c(t) \ \dots \text{ Galerkin ansatz}$$

- Approximate $f^\ell(t,\eta^\ell)$ through Ψ, W, Φ,

$$f^\ell(t,\eta^\ell) = \Psi^{\mathrm{T}} W f(t, \Psi\eta^\ell) = \Psi^{\mathrm{T}} W b(t) \approx \Psi^{\mathrm{T}} W \Phi c(t), \quad c(t) \in \mathbb{R}^p$$

- Plug-in the last output of the DEIM algorithm, $\vec{i}$

$$P := [e_{\vec{i}1}, \dots, e_{\vec{i}p}] \in \mathbb{R}^{m\times p},\ e_{\vec{i}k} = (0,\dots,0,1,0,\dots,0)^{\mathrm{T}} \in \mathbb{R}^m$$

Deal with the non-linearities II (yes, almost done)

- Plug in the matrix P,

$$P^{\mathrm{T}}\Phi c(t) \approx P^{\mathrm{T}}b(t), \leftarrow c(t) \in \mathbb{R}^p,\ \Phi \in \mathbb{R}^{m\times p},\ b(t) \in \mathbb{R}^m$$

$$\det(P^{\mathrm{T}}\Phi) \neq 0 \implies c(t) \approx (P^{\mathrm{T}}\Phi)^{-1}P^{\mathrm{T}}b(t) = (P^{\mathrm{T}}\Phi)^{-1}P^{\mathrm{T}}f(t, \Psi\eta^\ell)$$

- If $f(t, \Psi\eta^\ell)$ is pointwise evaluable,

$$(P^{\mathrm{T}}\Phi)^{-1}P^{\mathrm{T}}f(t, \Psi\eta^\ell) = (P^{\mathrm{T}}\Phi)^{-1}f(t, P^{\mathrm{T}}\Psi\eta^\ell), \quad P^{\mathrm{T}}\Psi\eta^\ell \in \mathbb{R}^p$$

- Write the final ROM

$$\dot{\eta}^\ell = A^\ell\eta^\ell + f^\ell(t, \eta^\ell),\ \text{for}\ t \in (0, T], \quad \eta^\ell(0) = \eta_0^\ell,$$

where

$$f^\ell(t, \eta^\ell) = \Psi^{\mathrm{T}}W\Phi(P^{\mathrm{T}}\Phi)^{-1}f(t, P^{\mathrm{T}}\Psi\eta^\ell)$$

Discrete Empirical Interpolation Method

POD & Greedy algorithm based method for handling non-linearities

---

Algorithm 2 DEIM

Require: p and matrix $F = [f(t_1, y_1), \dots, f(t_1, y_1)] \in \mathbb{R}^{m \times n}$

1: Compute POD basis $\Phi = [\phi_1, \dots, \phi_p]$ for F

2: $\mathrm{idx} \leftarrow \arg\max_{j=1,\dots,m} |(\phi_1)_{\{j\}}|$;

3: $U = [\phi_1]$ and $\vec{i} = \mathrm{idx}$;

4: **for** $i = 2$ **to** p **do**

5:     $u \leftarrow \phi_i$;

6:     Solve $U_{\vec{i}}c = u_{\vec{i}}$;

7:     $r \leftarrow u - Uc$;

8:     $\mathrm{idx} \leftarrow \arg\max_{j=1,\dots,m} |(r)_{\{j\}}|$;

9:     $U \leftarrow [U, u]$ and $\vec{i} \leftarrow [\vec{i}, \mathrm{idx}]$;

10: **end for**

11: **return** $\Phi \in \mathbb{R}^{m \times p}$ and index vector, $\vec{i} \in \mathbb{R}^p$

Notes:

- Most of the computational cost is hidden on line 6.

Link with OpenFOAM

Rewrite OpenFOAM discretization as above studied problem

- With $\Delta\Omega^h := \mathrm{diag}(\delta\Omega_i^h) \in \mathbb{R}^{m\times m}$ a FVM semi-discretized problem can be written as,

$$\Delta\Omega^h \dot{y} + \mathcal{L}^h(t,y) = 0 \implies \dot{y} = -(\Delta\Omega^h)^{-1}\mathcal{L}^h(t,y),$$

  $\mathcal{L}^h = -\tilde{A}(t)y - \tilde{b}(t,y)$ ... FVM spatial discretization operator

- It is possible to formally write (almost) the same system as before,

$$\dot{y} = A(t)y + b(t,y), \quad A(t) = (\Delta\Omega^h)^{-1}\tilde{A}(t),\, b(t,y) = (\Delta\Omega^h)^{-1}\tilde{b}(t,y)$$

- The time dependence of A is a result of the linearization process. E.g. $\nabla\cdot(u^k \otimes u^k) \approx \nabla\cdot(u^{k-1}\otimes u^k)$
- The POD-DEIM approach to ROM creation will have to be slightly modified

Address the risen difficulties

- Needed snapshots, $\{(y_i, A_i, b_i)\}_{i=1}^{n}$, $A_i \in \mathbb{R}^{m\times m}$, $i = 1, \ldots, m$ but A_i are sparse matrices, with $\sim 5m$ non-zero elements $\implies \sim 5m$ floats and $\sim 8m$ integers will be stored.
- A way for ROM evaluation between the stored snapshots is needed $\implies$ I need to interpolate between A_{i-1} and A_i and b_{i-1} and b_i, $i = 2, n$
- Simplest case: linear interpolation,

$$\varpi(t) = \frac{t - t_{i-1}}{t_i - t_{i-1}},\ \hat{A}(t) = \varpi(t)A_{i-1} + (1 - \varpi(t))A_i$$

$$\hat{A}^\ell(t) = \Psi^{\mathrm{T}} W \hat{A}(t)\Psi = \Psi^{\mathrm{T}} W \left(\varpi(t)A_{i-1} + (1 - \varpi(t))A_i\right)\Psi =$$

$$= \varpi(t)\Psi^{\mathrm{T}} W A_{i-1}\Psi + (1 - \varpi(t))\Psi^{\mathrm{T}} W A_i\Psi = \varpi(t)A^\ell_{i-1} + (1 - \varpi(t))A^\ell_i$$

- Same trick can be done for $b(t, y)$ and after the ROM creation, I do not need to store the full data.

interFoam – Volume-of-Fluid model for multiphase flow

$$\alpha_t + \nabla \cdot (u\alpha) + \nabla \cdot (u_r \alpha(1-\alpha)) = 0$$

$$\alpha_t + \mathcal{L}^h_\alpha(t,\alpha) = 0 \to \alpha_t = A_\alpha(t)\alpha + b_\alpha(t,\alpha) \to \dot{\eta}^\ell_\alpha = \hat{A}^\ell_\alpha(t)\eta^\ell_\alpha + \hat{b}^\ell_\alpha(t,\eta^\ell_\alpha)$$

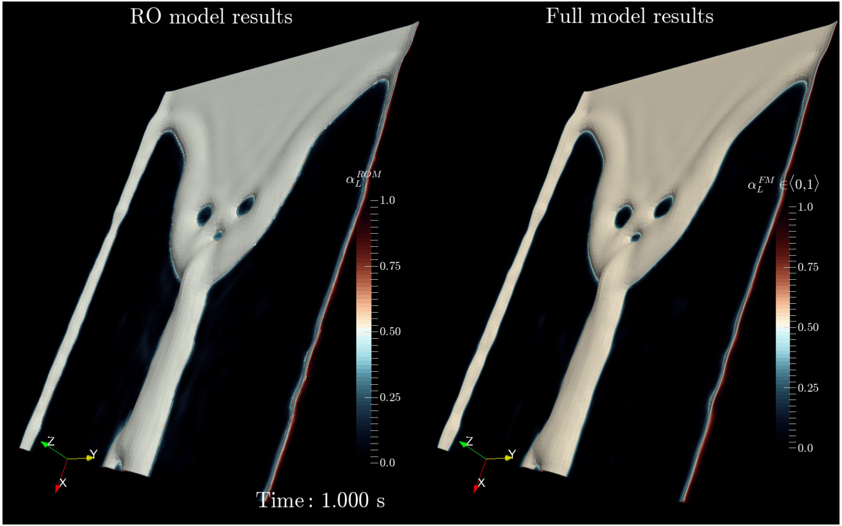
Wanted: $\dot{y}_\alpha = A_\alpha(t) y_\alpha + b_\alpha(t,y)$

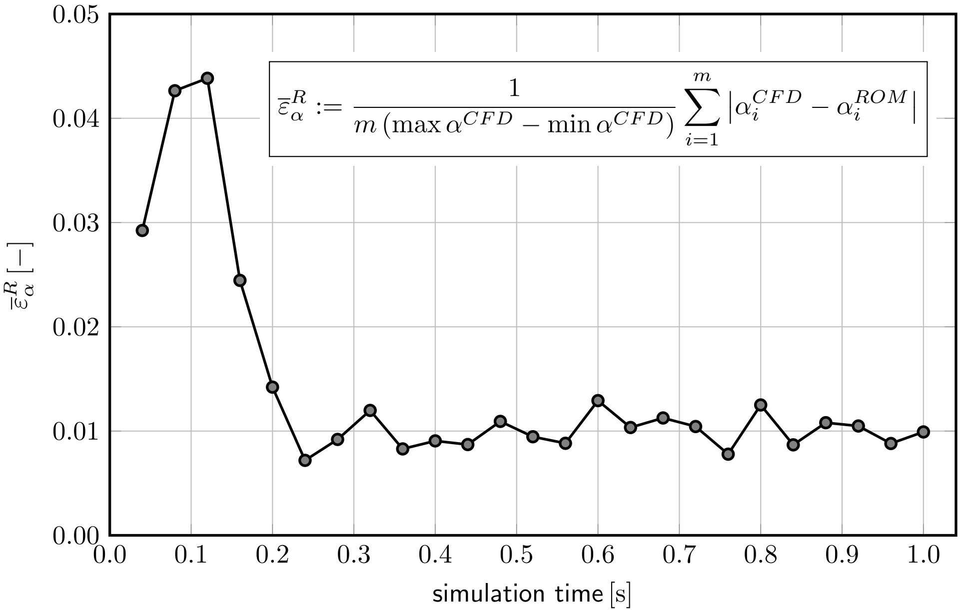
Example of implementation in OpenFOAM

```
    fvm::div( phi, alpha1, alphaScheme )
  + fvc::div(
        -fvc::flux(
            -phir, scalar(1)-alpha1, alpharScheme
        ),
        alpha1, alpharScheme
    ) == 0
```

Link: $\texttt{fvm} \to A_\alpha(t)$, $\texttt{fvc} \to b_\alpha(t,y)$

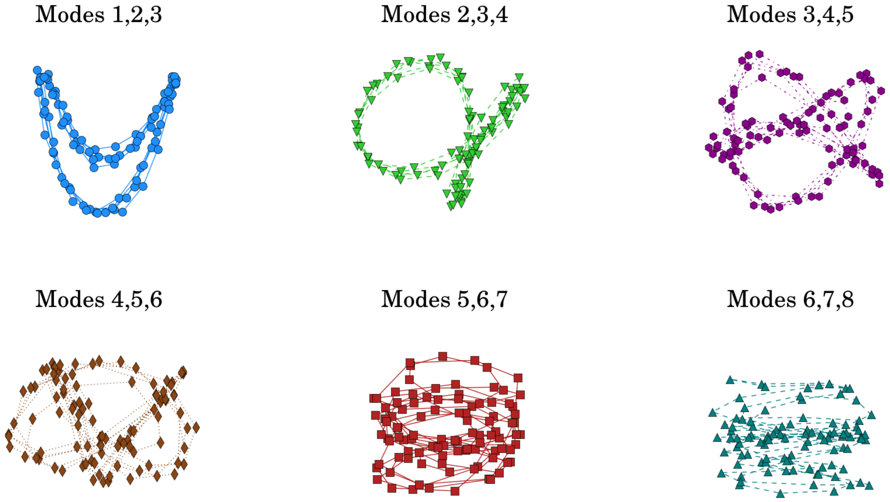
Example 1 – Passive scalar advection
Numerical results




$\overline{\varepsilon}_\alpha^R := \frac{1}{m\left(\max \alpha^{CFD} - \min \alpha^{CFD}\right)} \sum_{i=1}^{m} \left|\alpha_i^{CFD} - \alpha_i^{ROM}\right|$
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
$\overline{\varepsilon}_\alpha^R\,[-]$
simulation time $[\mathrm{s}]$

Example 1 – Passive scalar advection

Numerical results



Saddle-point problem

$$\begin{aligned} u_t + \nabla\cdot(u\otimes u) - \nabla\cdot(\nu\nabla u) &= -\nabla\tilde{p} + \tilde{f} \\ \nabla\cdot u &= 0 \end{aligned} \rightsquigarrow \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$

Jacobi iterations with Schur-complement based p-U coupling

$$u^* \leftarrow Au^* = f - B^{\mathrm{T}}p^{k-1}$$

$$p^k \leftarrow BD^{-1}B^{\mathrm{T}}p^k = BD^{-1}\left(f - (L+U)u^*\right)$$

$$u^k \leftarrow D^{-1}\left(f - (L+U)u^* - B^{\mathrm{T}}p^k\right)$$

At convergence

$$BD^{-1}B^{\mathrm{T}}p^k = BD^{-1}\left(f - (L+U)u^*\right) \approx BA^{-1}B^{\mathrm{T}}p = BA^{-1}f$$

$$u = D^{-1}\left(f - (L+U)u^*\right) - D^{-1}B^{\mathrm{T}}p$$

Outcome for ROM

- "Natural" is to construct ROM for p
- For the velocity, I can choose between computational cost and consistency and accuracy

Notation

$D^{-1} \to \mathtt{rAU}$ and $D^{-1}(f - (L+U)u^*) \to \mathtt{HbyA}$ (in oF, $*\mathtt{Eqn.A()} \to D$)

Implementation of pressure eqauation in OpenFOAM

```
fvm::laplacian(rAU, p) == fvc::div(HbyA)
```

Wanted: $\dot{y}_p = A_p(t) y_p + b_p(t, y_p)$

Implicit definition of time derivative for pressure

$$\begin{aligned} \nabla \cdot (u \otimes u) - \nabla \cdot (\nu \nabla u) &= -\nabla \tilde{p} + \tilde{f} \\ \nabla \cdot u &= 0 \end{aligned} \quad \rightsquigarrow \quad \begin{array}{c} \mathtt{UEqnMORE} \\ D_h^{-1} \to \mathtt{rAUMORE} \\ D_h^{-1}(f_h - (L_h + U_h)u_h *) \to \\ \to \mathtt{HMOREbyAMORE} \end{array}$$

```
fvm::laplacian(rAUMORE, p) == fvc::div(HMOREbyAMORE)
```

Link: $\mathtt{fvm} \to A_p(t)$, $\mathtt{fvc} \to b_p(t, y_p)$

Expansion of snapshots for pressure

Standard approach snapshots: $\mathcal{S} = \{(y_{k,i}, A_{k,i}, b_{k,i}\}_{i=1}^{n}, k = p, U$

Expanded snapshots for pressure:

$$\mathcal{S}^e = \{(y_{p,i}, A_{p,i}, b_{p,i}, \mathtt{rAUMORE}_i, \mathtt{HMOREbyAMORE}_i\}_{i=1}^{n}$$

Storage

$\mathcal{S}$ $\ldots n\left[(1+3)m + (5+5)m + (1+3)m\right] \approx 15nm$ values

$\mathcal{S}^e$ $\ldots n(m + 5m + m + 1m + 3m) \approx 11nm$ values

Computational cost

$\mathcal{S}$ $\ldots \sim 4n$ calculations of $\Psi^{\mathrm{T}} W A(t) \Psi$, evaluation of ~ 4 ROMs

$\mathcal{S}^e$ $\ldots$

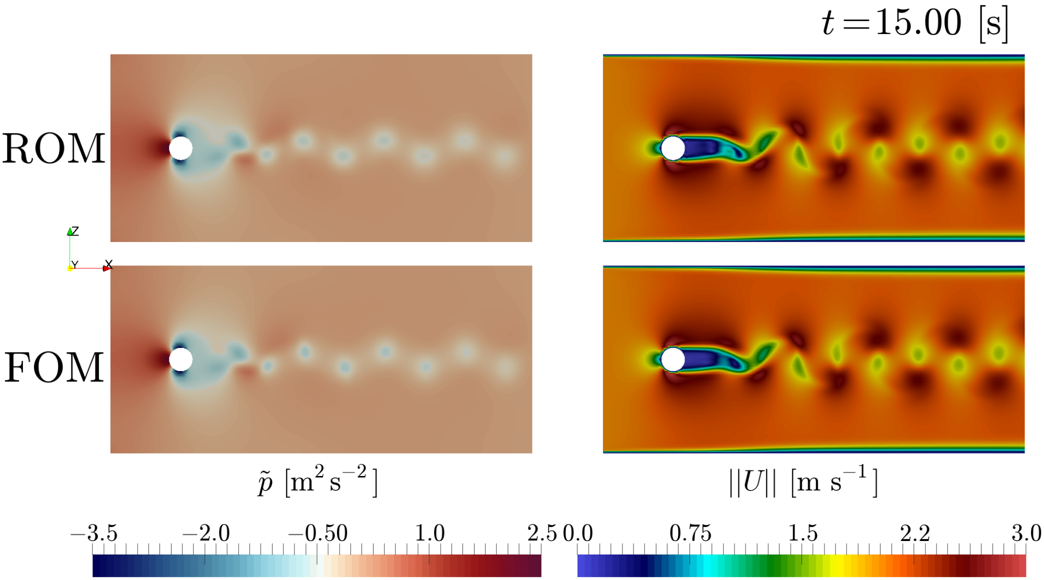
$\sim n$ calculations of $\Psi^{\mathrm{T}} W A(t) \Psi$,

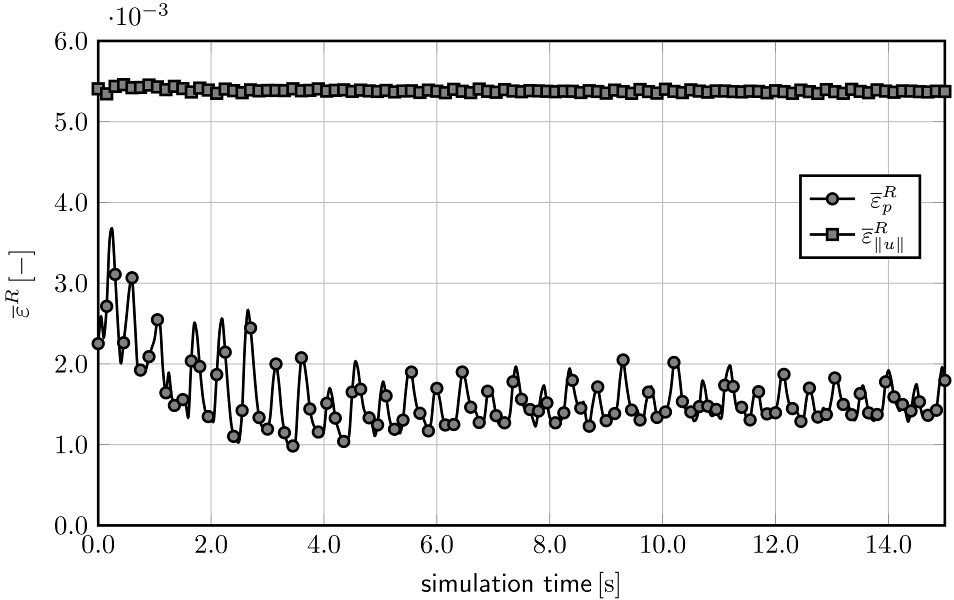
$\sim n$ calculations of $\Psi^{\mathrm{T}} W \mathtt{rAUMORE}_i \Psi$,

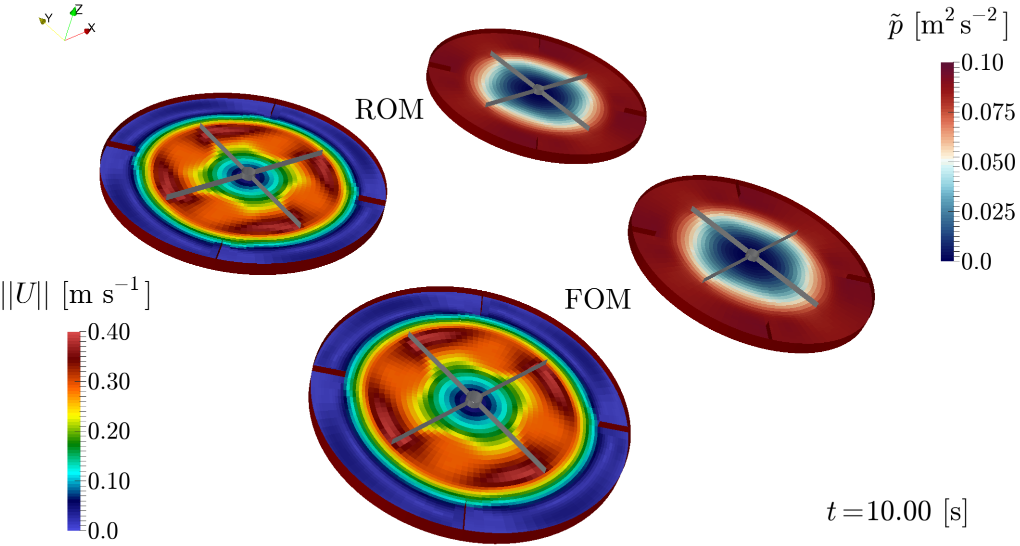
$\sim n$ calculations of $\Psi^{\mathrm{T}} W \mathtt{HMOREbyAMORE}_i \Psi$,

evaluation of 1 ROM + interpolation between $\mathtt{rAUMORE}_i{}^{ROM}$ and between $\mathtt{HMOREbyAMORE}_i{}^{ROM}$

$$U_i \approx \mathtt{HMOREbyAMORE}^{ROM} + \mathtt{rAUMORE}^{ROM} \nabla p^{ROM}$$

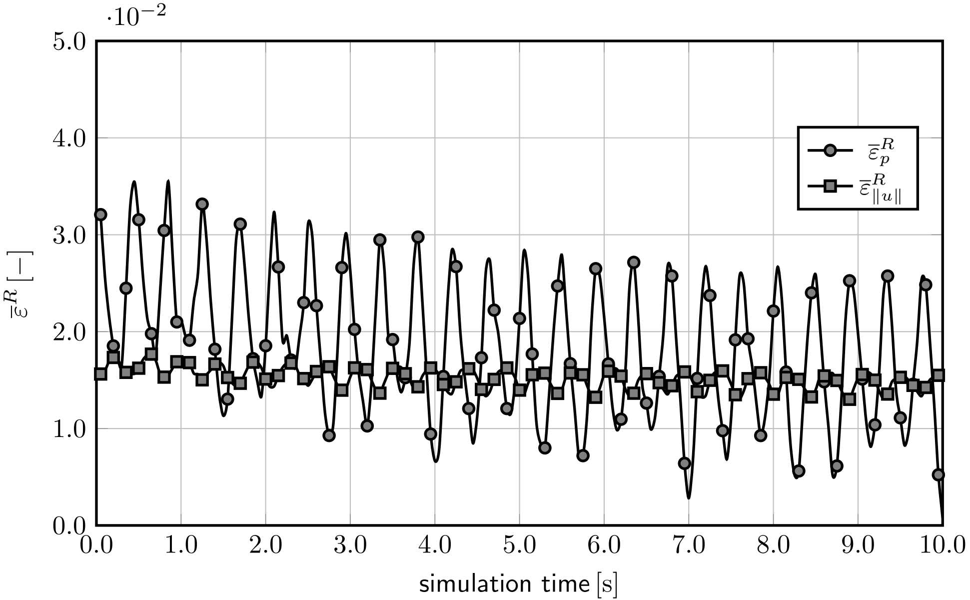

$t = 15.00$ [s]
ROM
FOM
Z
Y
X
$\tilde{p}$ [$m^2 s^{-2}$]
$||U||$ [$m\ s^{-1}$]
−3.5
−2.0
−0.50
1.0
2.5
0.0
0.75
1.5
2.2
3.0


$\cdot 10^{-3}$
6.0
5.0
4.0
3.0
2.0
1.0
0.0
$\overline{\varepsilon}^R$ $[-]$
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
simulation time $[\mathrm{s}]$
$\overline{\varepsilon}_p^R$
$\overline{\varepsilon}_{\|u\|}^R$

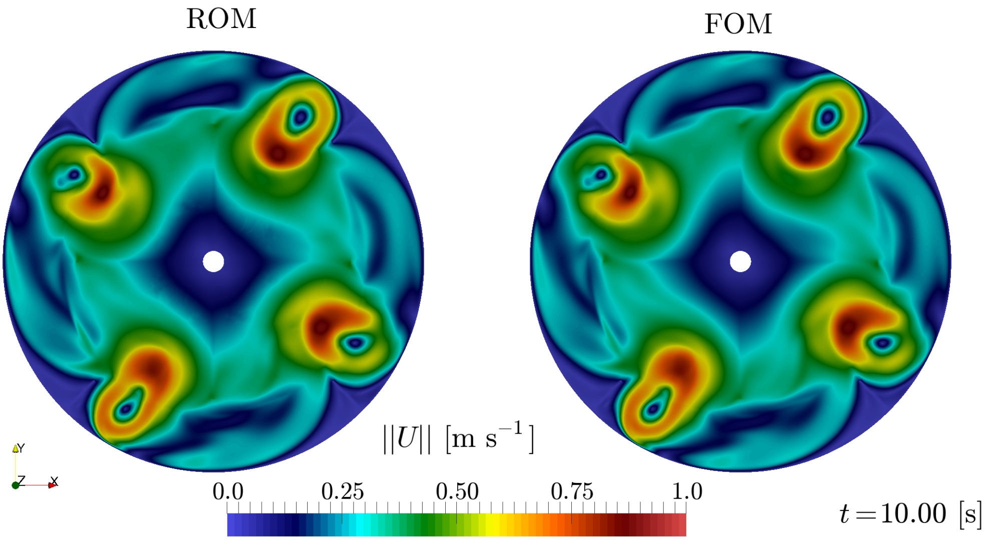

Y
Z
X
ROM
FOM
$\tilde{p}$ [m^2 s^{-2}]
0.10
0.075
0.050
0.025
0.0
$||U||$ [m s^{-1}]
0.40
0.30
0.20
0.10
0.0
$t = 10.00$ [s]

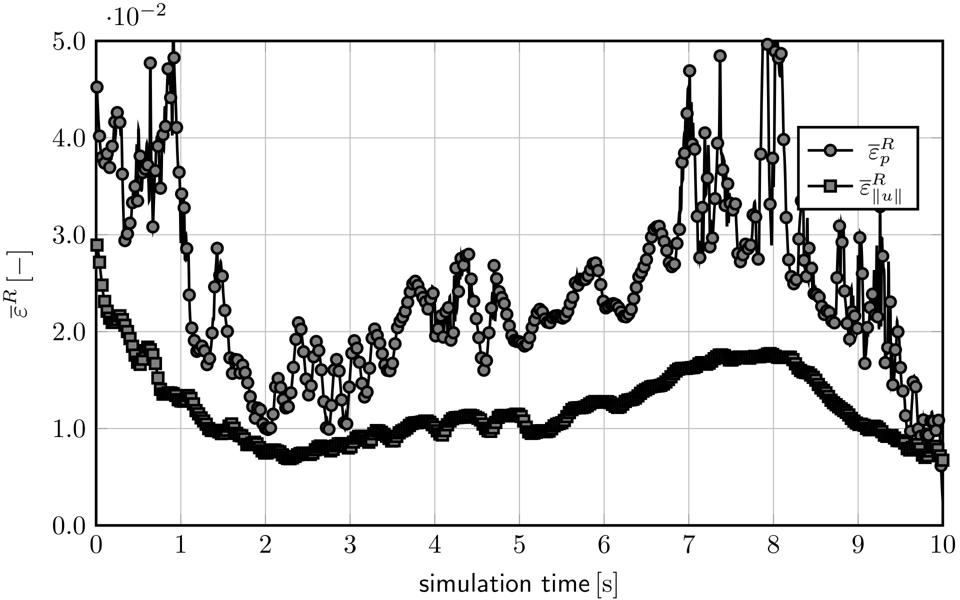
Example 3 – 2D mixer

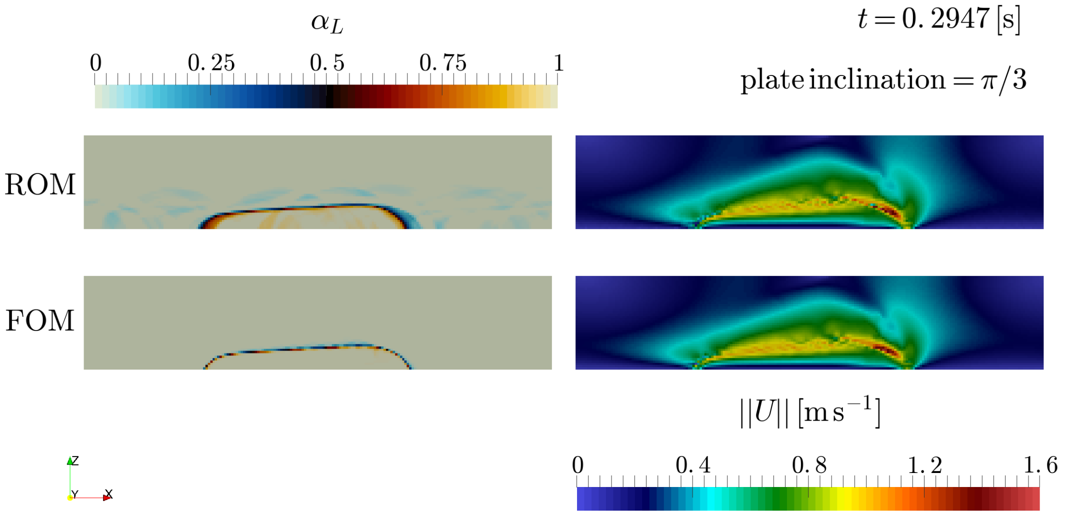
Validation of the approach – arbitrary mesh interface



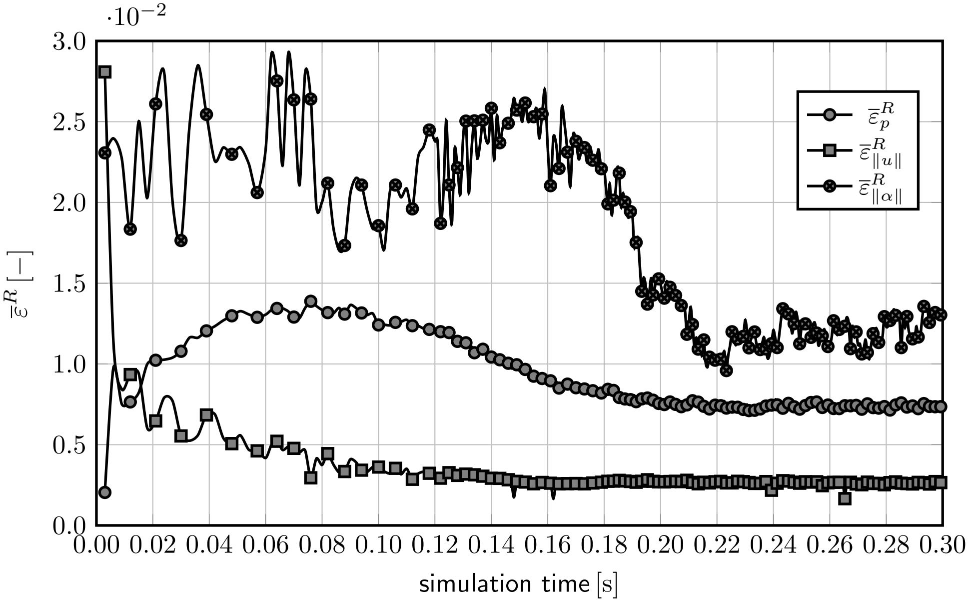
Validation of the approach – multiple reference frames




$\cdot 10^{-2}$
5.0
4.0
3.0
2.0
1.0
0.0
0
1
2
3
4
5
6
7
8
9
10
$\overline{\varepsilon}^R\,[-]$
simulation time $[\mathrm{s}]$
$\overline{\varepsilon}_p^R$
$\overline{\varepsilon}_{\|u\|}^R$



α_L
0
0.25
0.5
0.75
1
$t = 0.2947$ [s]
plate inclination $= \pi/3$
ROM
FOM
$||U||$ [m s^{-1}]
0
0.4
0.8
1.2
1.6
Z
Y
X

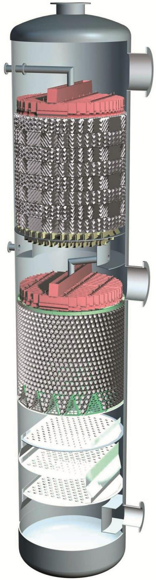


$\cdot 10^{-2}$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24 0.26 0.28 0.30
simulation time [s]
$\overline{\varepsilon}^R$ [−]
$\overline{\varepsilon}_p^R$
$\overline{\varepsilon}_{\|u\|}^R$
$\overline{\varepsilon}_{\|\alpha\|}^R$

Applications

Real-life applications

ROM is a tremendous tool for parametric studies or repeated model evaluations



[Sulzer ChemTech]

Importance

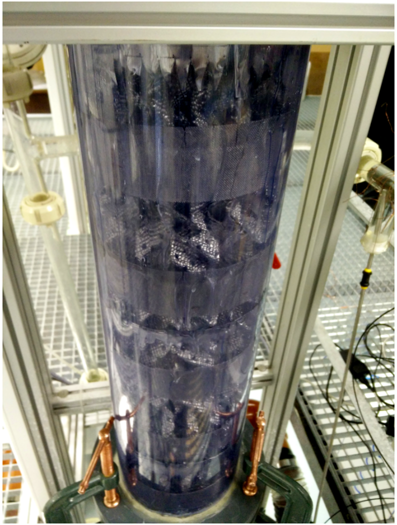
- Chemical industry creates mixtures but sells "pure species" (e.g. oil)
- 2014, 3% of energy consumption of the USA was due to the separation columns

Challenges

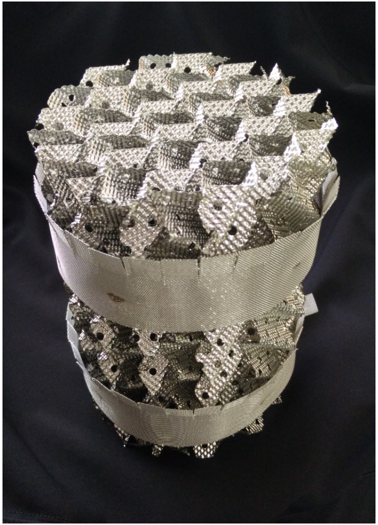
- Multiphase flow $\rightarrow$ non-steady process
- Complex geometry
- Simultaneous heat and mass transfer

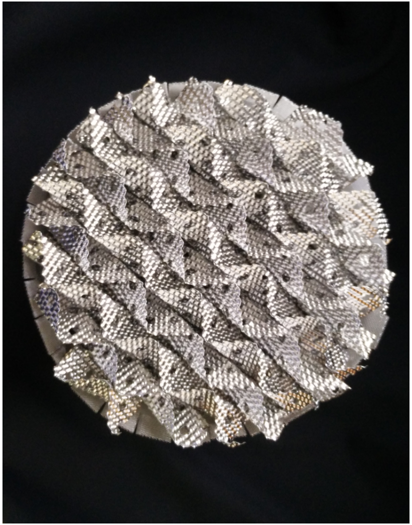
Packed column

Complex multiphase flow

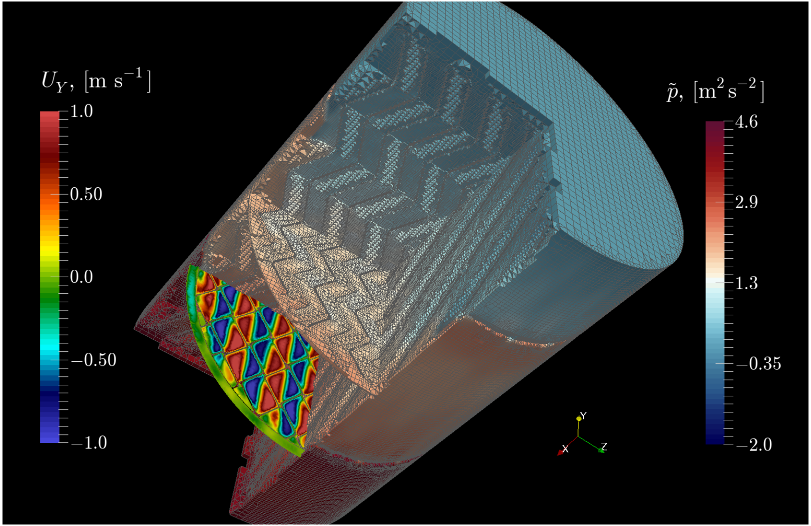


Challenge: Geometry of structured packing

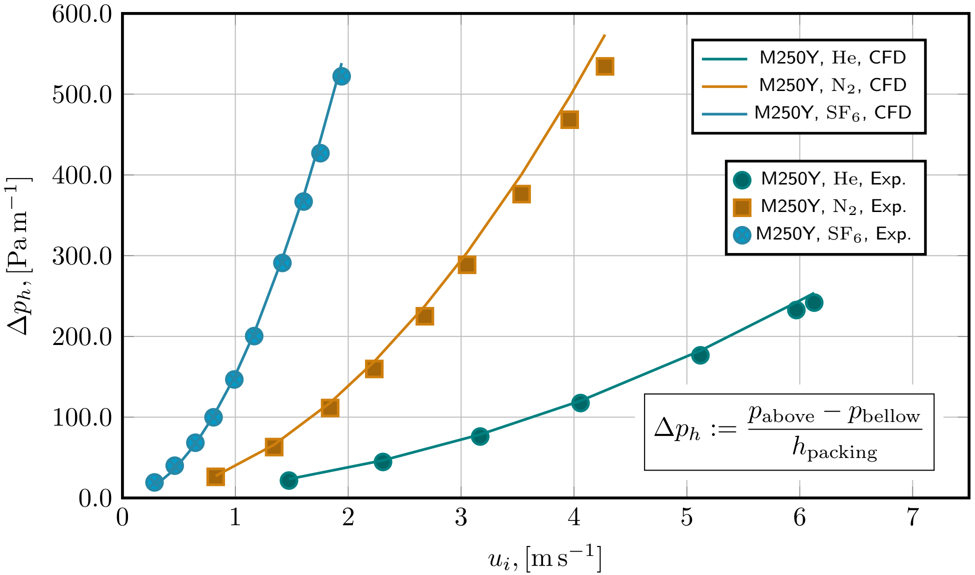




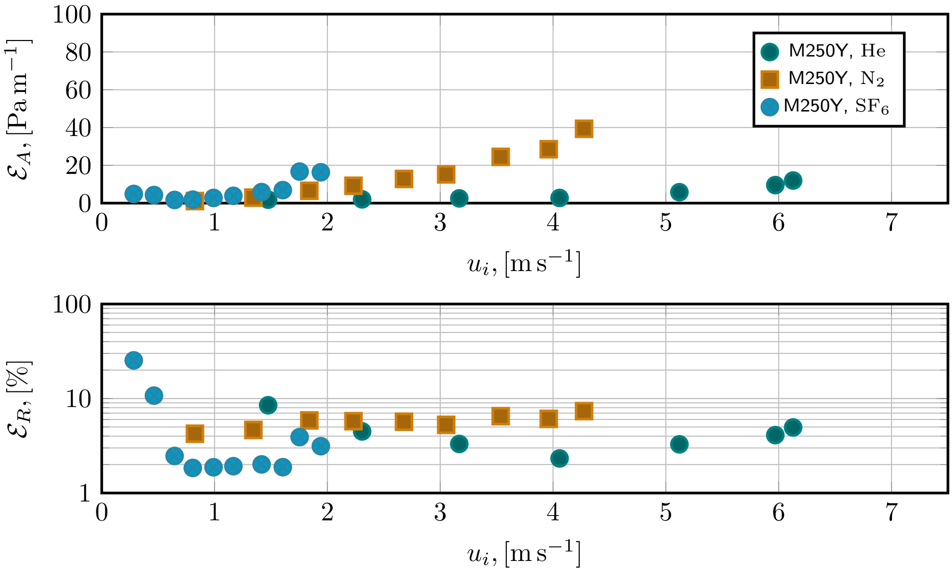
Gas flow simulation: Incompressible steady state RANS simulation



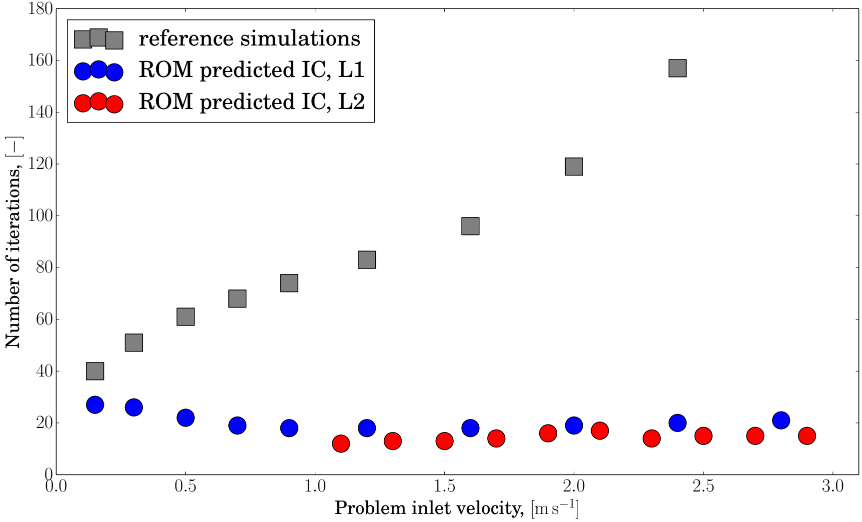
Comparison with experimental data: [Haidl, J. UCT Prague]



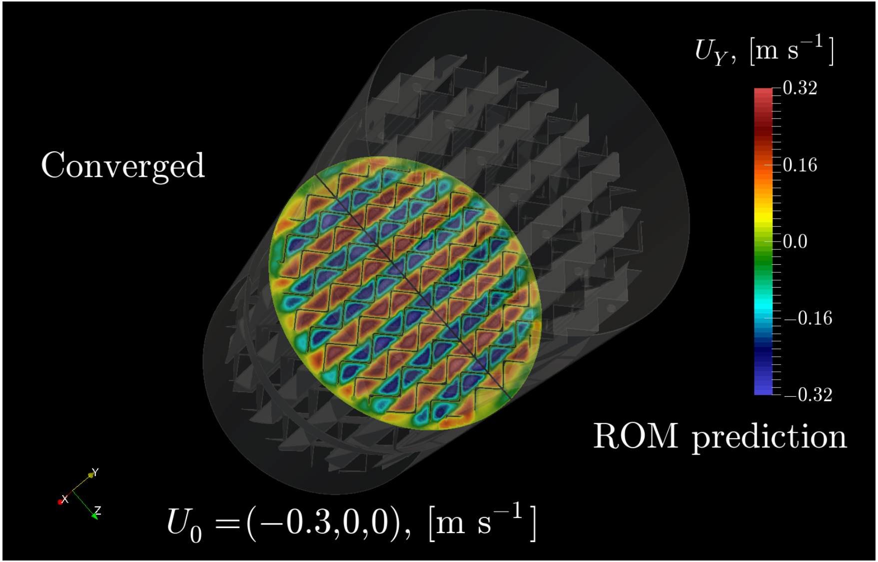
Comparison with experimental data: [Haidl, J. UCT Prague]



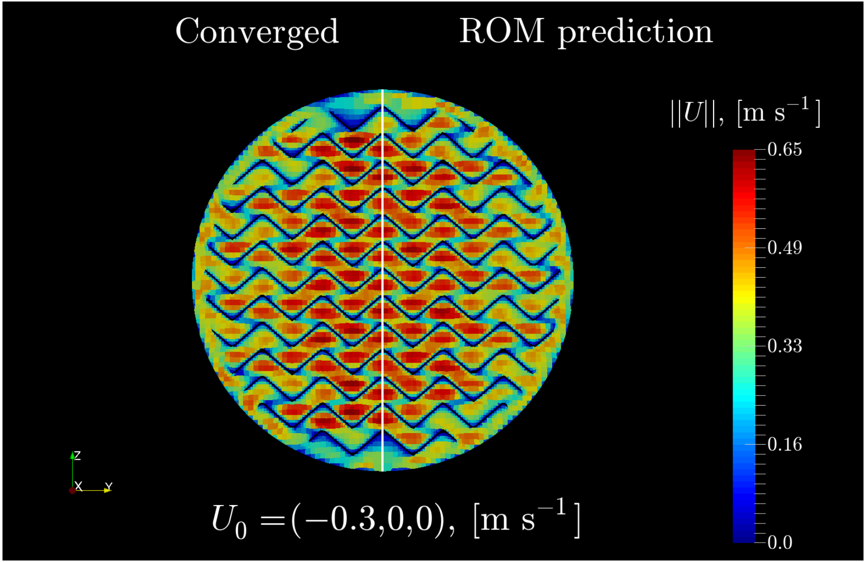
Full case: Flow through the Mellapak 250.X packing



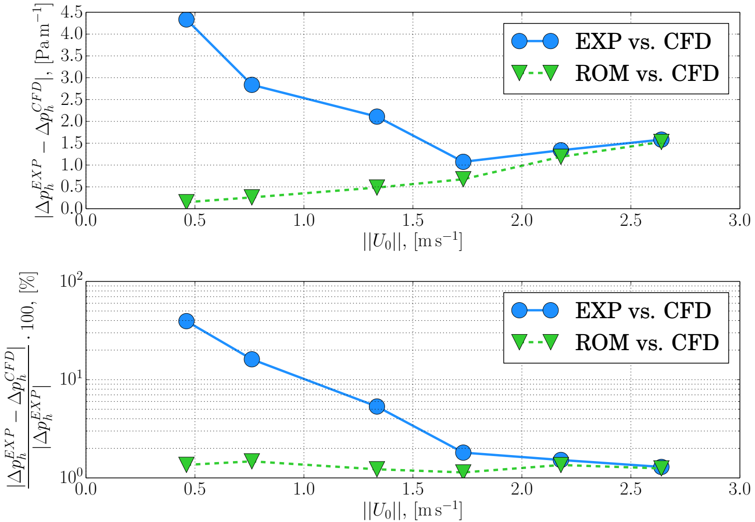
Full case: Predicted vs. converged solution in L1



Full case: Predicted vs. converged solution in L1



Comparison with experimental data: [Haidl, J. UCT Prague]



Cost function: Single phase, toy problem

$$F(u_0) = \frac{\Delta\tilde{p} - \Delta\tilde{p}_{Max}}{\Delta\tilde{p}_{Max}} + K\frac{Q^2 - 2Q_{Max}Q + Q_{Min}(2Q_{Max} - Q_{Min})}{(Q_{Max} - Q_{Min})^2},$$

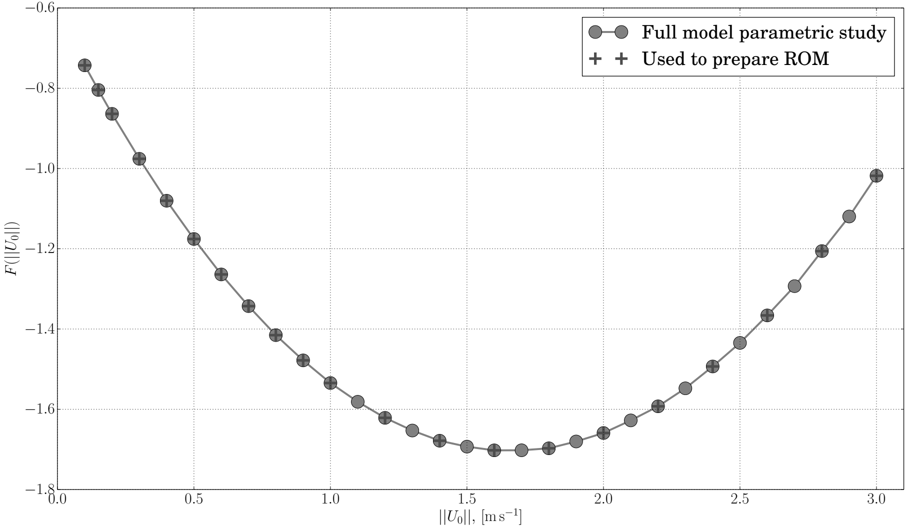
$$\Delta\tilde{p} = \Delta\tilde{p}(u_0), \quad Q = Q(u_0), \quad U_0 = (-u_0, 0, 0),$$

$\Delta\tilde{p}_{Max}$ .... maximal allowable pressure loss

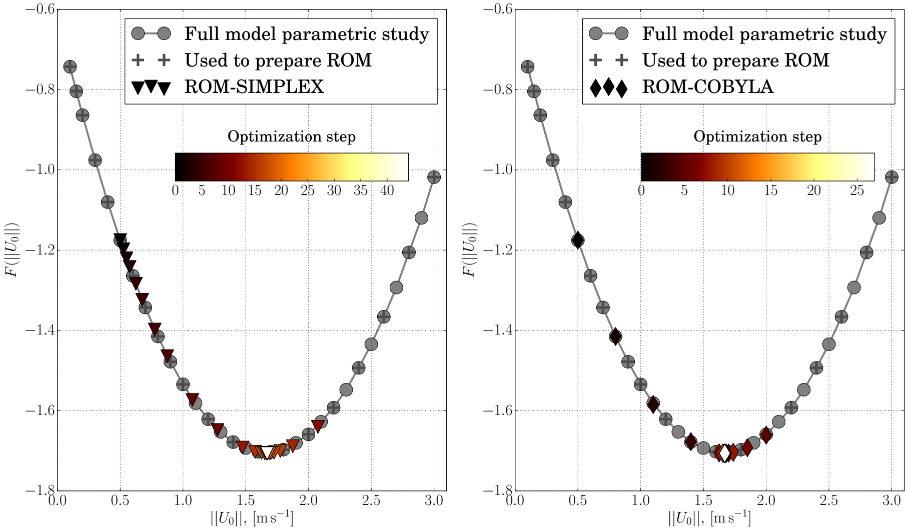
Q_{Max}, (Q_{Min}) .... maximal, (minimal) allowable gas flow rate

K .... relative importance of the two terms

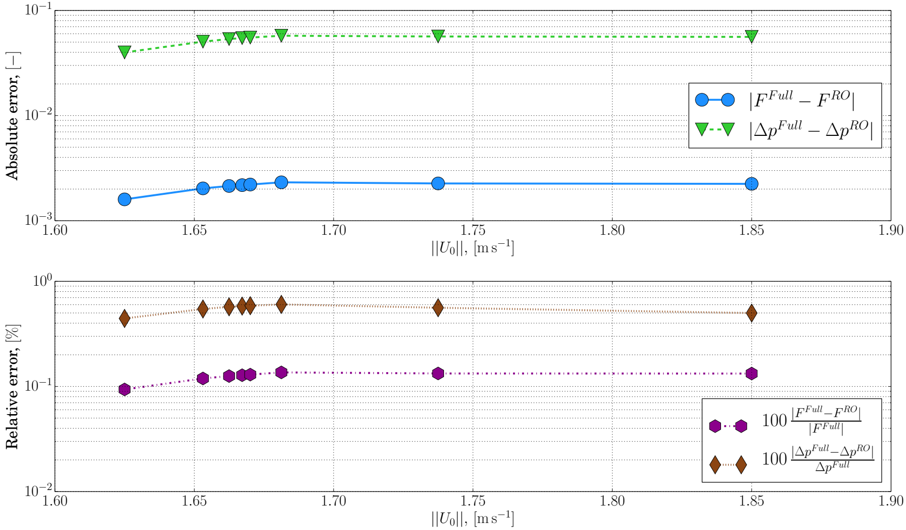
Available data: Cost function curve, $F(u_0)$, $u_0 \in \langle 0.1, 3.0 \rangle$



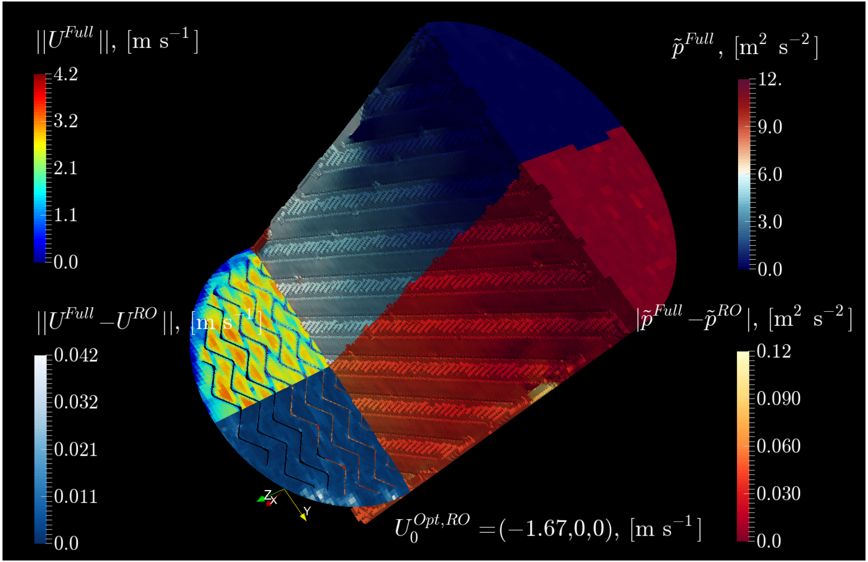
Cost function minimization: Results of SIMPLEX and COBYLA algorithms



Solution quality: Comparison of ROM results with reference simulations (COBYLA)



Solution quality: Comparison of RO and Full models results



Conclusions

Currently available

- Extended snapshot preparation for `simpleFoam`, `pimpleFoam` and `interFoam`
- Python module for ROM creation based on prepared outputs from OpenFOAM

Advantages

- Snaphots are created during postprocessing - simulations can be ran in parallel
- All the OpenFOAM capabilities are accessible (including e.g. MRF or turbulence modeling)

Disadvantages

- Extended shapshots have to be stored - a lot of data
- Creation of A_i^ℓ, $i = 1, \ldots, n$ is time consuming

Acknowledgments

The work of M. Isoz was supported by the Centre of Excellence for nonlinear dynamic behaviour of advanced materials in engineering CZ.02.1.01/0.0/0.0/15_003/0000493 (Excellent Research Teams) in the framework of Operational Programme Research, Development and Education.

Special thanks to Prof. Michael Hinze from Hamburg university for his inputs and discussions during the preparation of the presented work.

References

[1] Volkwein, S.: Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. *LN*, University of Konstanz, 2013.

[2] Volkwein, S.: Proper Orthogonal Decomposition: Applications in Optimization and Control

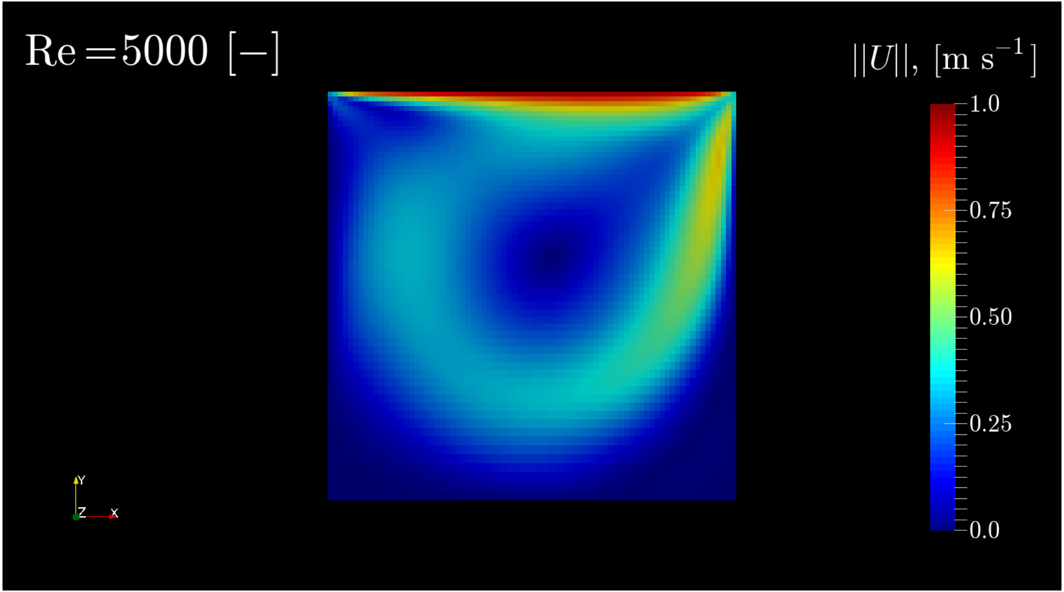
[3] Chaturantabut, S. Sorensen, D. C.: Nonlinear Model Reduction Via Discrete Empirical Interpolation, *SIAM J. Sci. Comput.*, vol. 32, (2010) pp. 2737–2764.

[4] Chaturantabut, S. Sorensen, D. C.: Application of POD and DEIM on Dimension Reduction of Nonlinear Miscible Viscous Fingering in Porous Media, *Math. Comput. Model. Dyn. Syst., (Technical Report: CAAM)*, Rice University, TR09-25

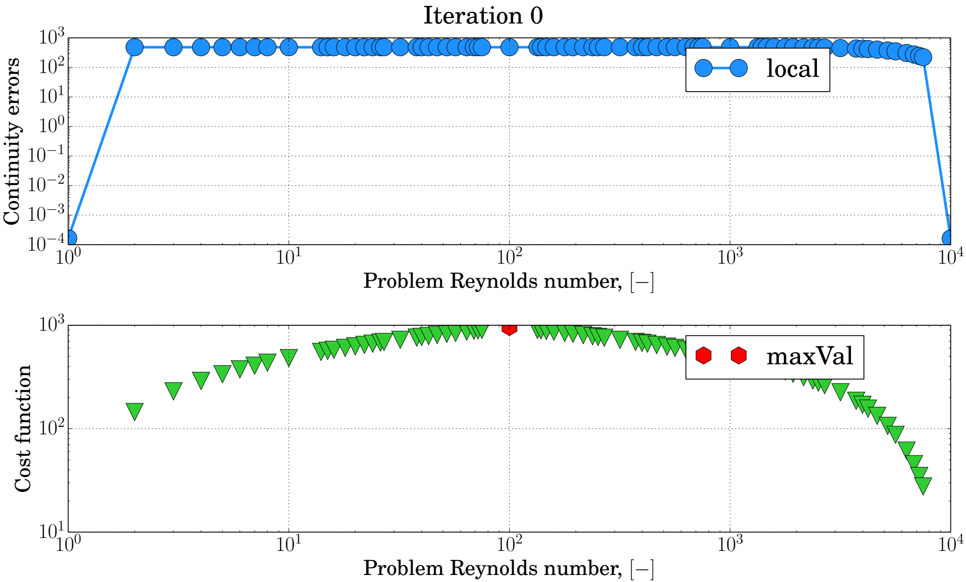
[5] Alla, A. Kutz, J. N.: Nonlinear Model Order Reduction Via Dynamic Mode Decomposition, *preprint*

Thank you for your attention

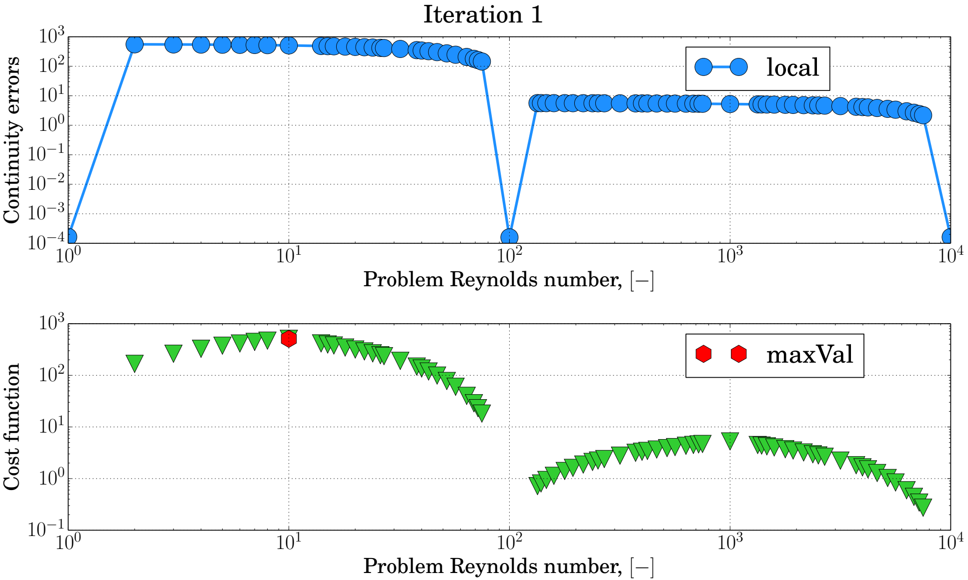
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



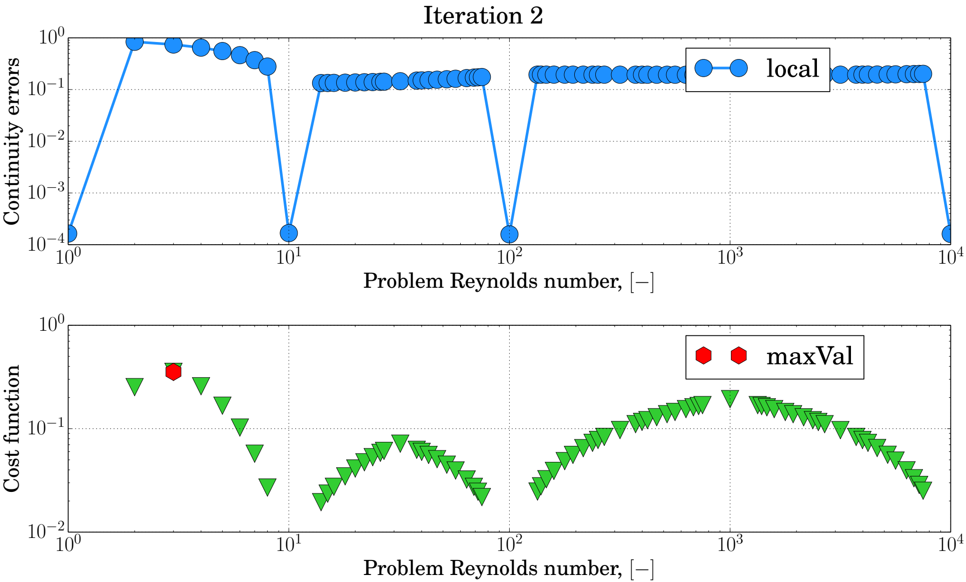
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



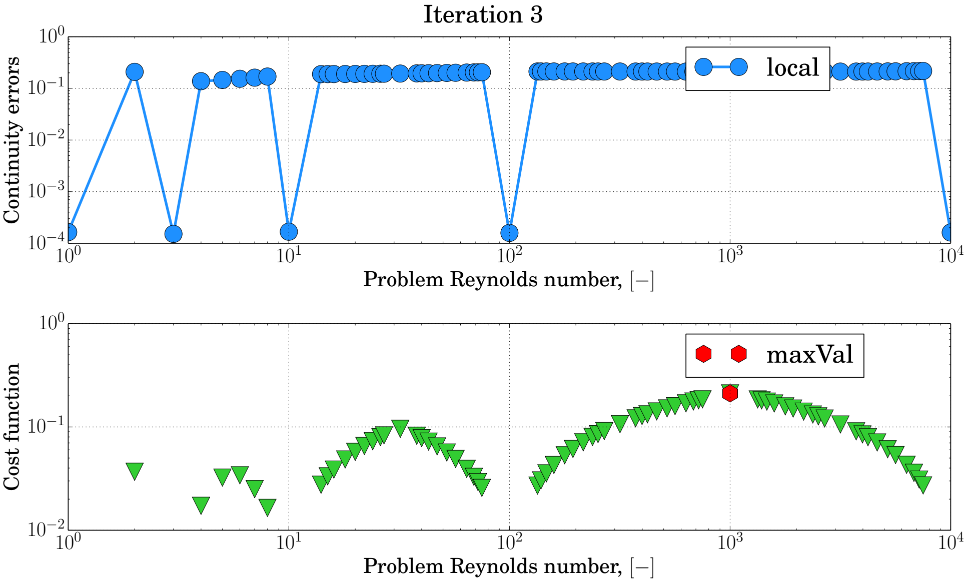
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



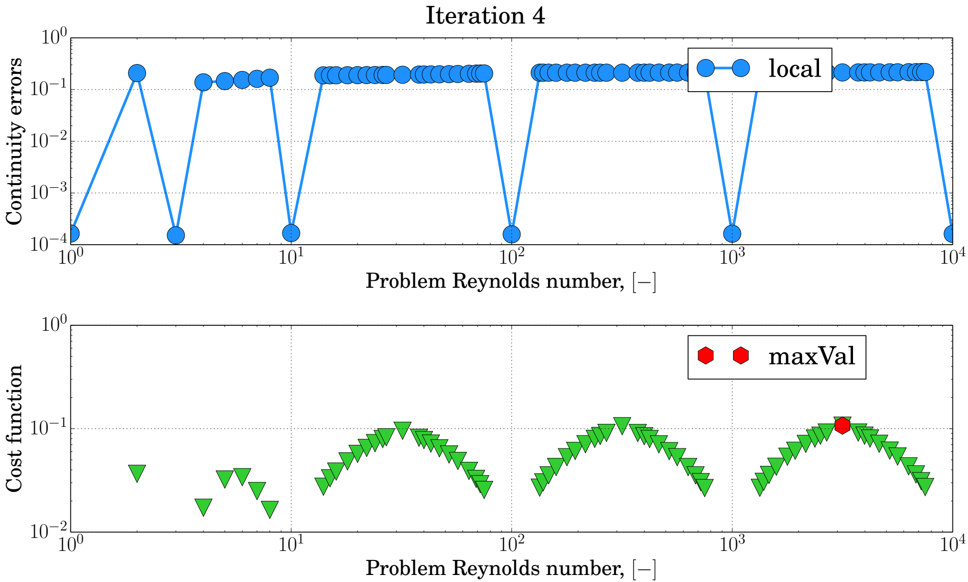
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



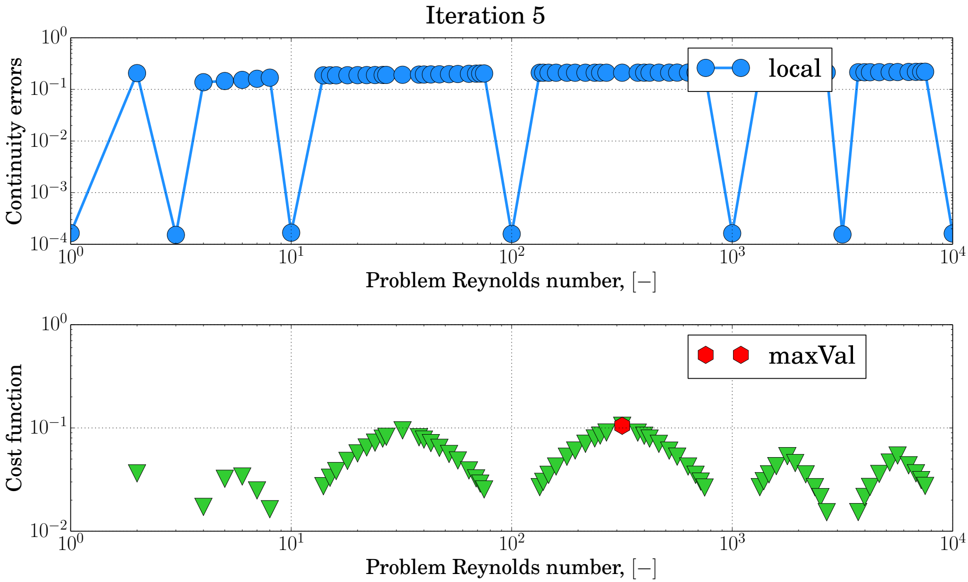
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



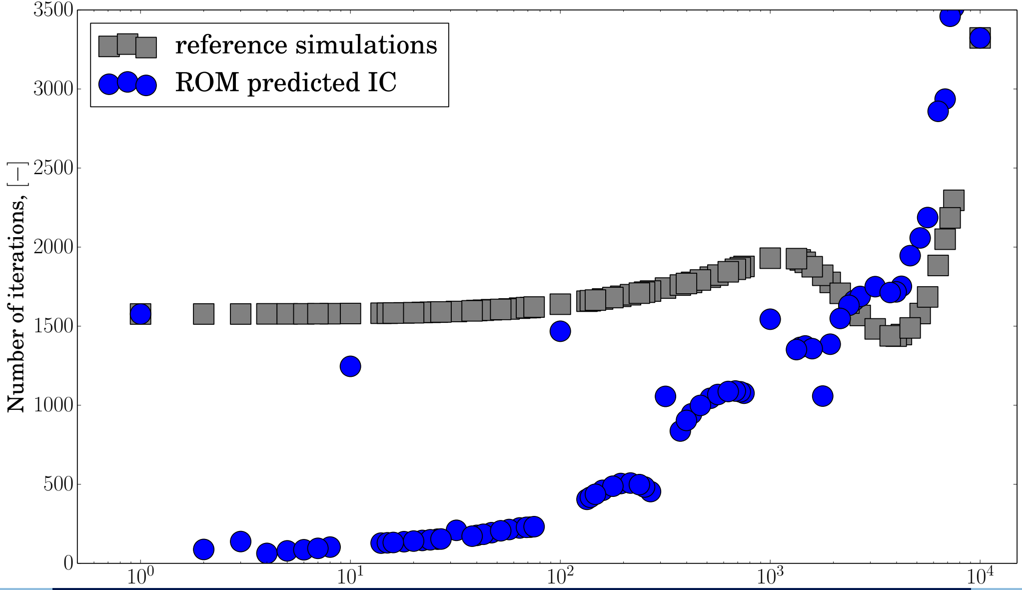
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



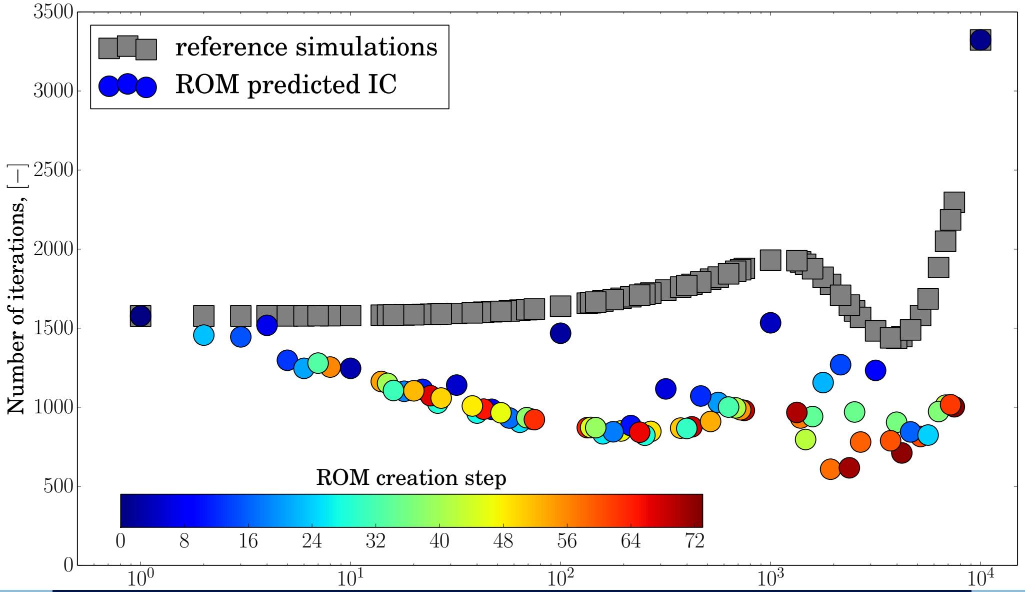
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm

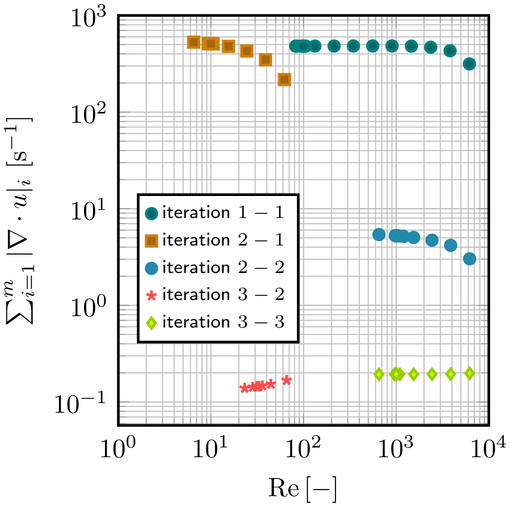


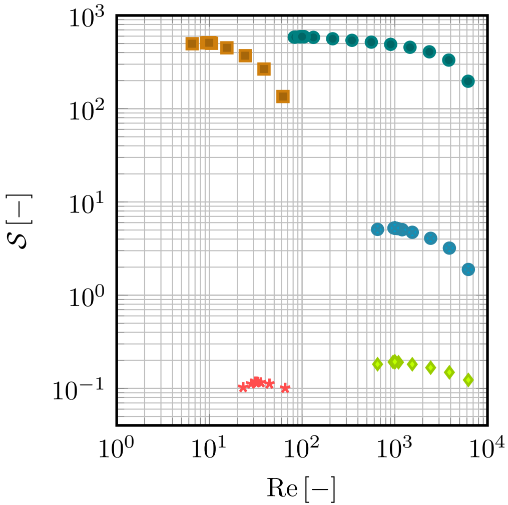
ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm



ROM size reduction: $\mathcal{S}^e$ selection based on greedy algorithm




10^3
10^2
10^1
10^0
10^{-1}
$\sum_{i=1}^{m} |\nabla \cdot u|_i \, [s^{-1}]$
iteration 1 — 1
iteration 2 — 1
iteration 2 — 2
iteration 3 — 2
iteration 3 — 3
10^0 10^1 10^2 10^3 10^4
Re [−]


10^3
10^2
10^1
10^0
10^{-1}
S [−]
10^0 10^1 10^2 10^3 10^4
Re [−]

- Let us have rather nice functions defined on a nice domain,

$$\varphi, \tilde{\varphi} \in L^2(\Omega), \quad \Omega \subset \mathbb{R}^3 \ldots \text{bounded, connected, } \ldots$$

- A brief reminder,

$$\langle \varphi, \tilde{\varphi} \rangle_{L^2(\Omega)} = \int_\Omega \varphi \tilde{\varphi} \, \mathrm{d}x, \quad ||\varphi||_{L^2(\Omega)} = \sqrt{\langle \varphi, \varphi \rangle_{L^2(\Omega)}}$$

- Denote Ω^h a FVM discretization of Ω and $\delta\Omega_i^h$ the volume of the i-th cell,

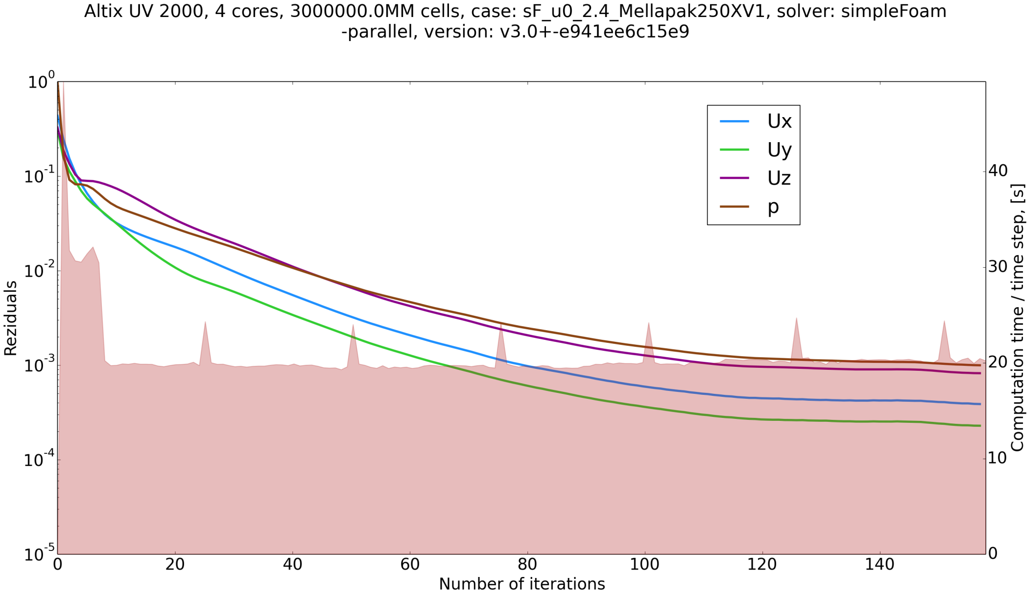
$$\Omega \approx \Omega^h = \bigcup_{i=1}^{\texttt{nCells}} \Omega_i^h, \quad V(\Omega) \approx V(\Omega^h) = \sum_{i=1}^{\texttt{nCells}} \delta\Omega_i^h$$

- Introduce a discrete inner product, $\langle \varphi, \tilde{\varphi} \rangle_{L_h^2}$,

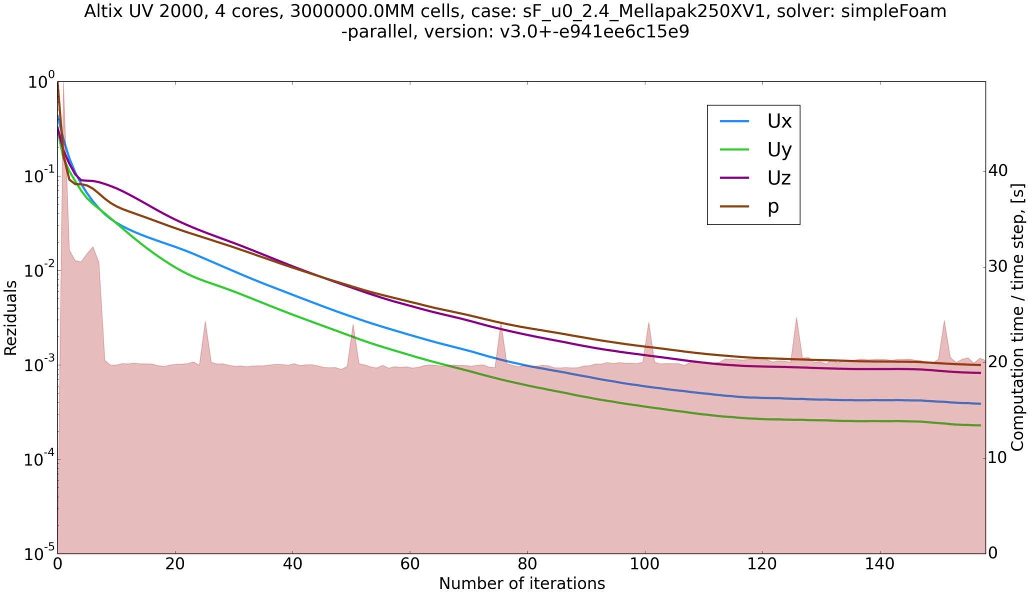
$$\langle \varphi, \tilde{\varphi} \rangle_{L^2(\Omega)} = \int_\Omega \varphi \tilde{\varphi} \, \mathrm{d}x \approx \sum_{i=1}^{\texttt{nCells}} \int_{\Omega_i^h} \varphi \tilde{\varphi} \, \mathrm{d}x = \sum_{i=1}^{\texttt{nCells}} \varphi_i^h \tilde{\varphi}_i^h \delta\Omega_i^h = \langle \varphi, \tilde{\varphi} \rangle_{L_h^2}$$

- Denote $W = \mathrm{diag}(\delta\Omega_1^h, \ldots, \delta\Omega_{\texttt{nCells}}^h)$. Hence, $\langle \varphi, \tilde{\varphi} \rangle_{L_h^2} = (\varphi^h)^{\mathrm{T}} W \varphi^h$.

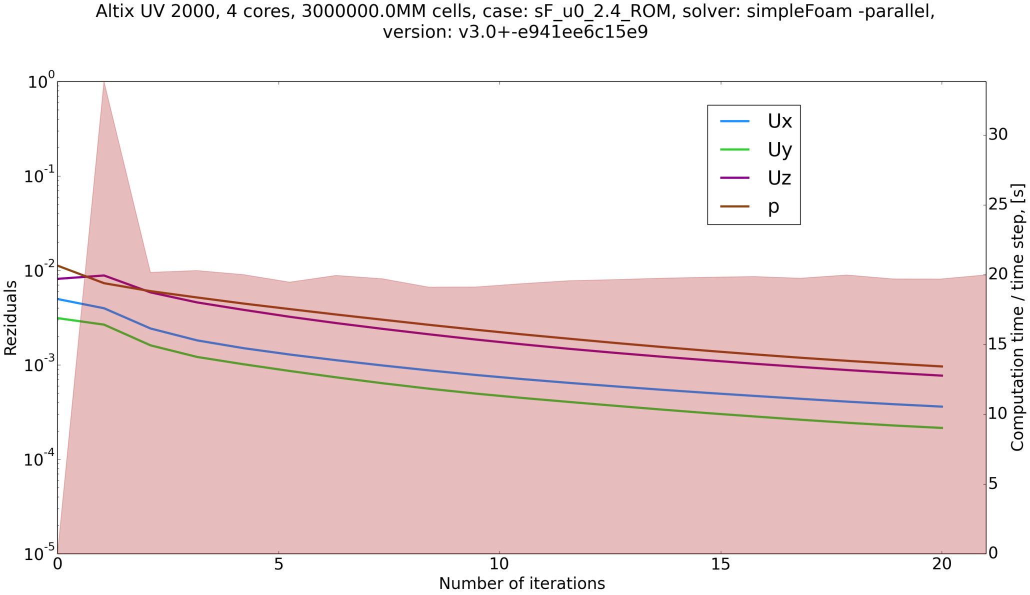
Full case: Residuals evolution, from potentialFoam initialized fields



Full case: Residuals evolution, from potentialFoam initialized fields



Full case: Residuals evolution, from ROM predicted fields, L1



Full case: Residuals evolution, from ROM predicted fields, L2