Proper orthogonal decomposition and discrete empirical interpolation in CFD applications

M. Isoz

α 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
Problem setting
Modeling of large systems of differential equations

Original system

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

- system matrix \( A \in \mathbb{R}^{m \times m} \),
- nonlinearities \( f(t, y) \in \mathbb{R}^m \)

Reduced-order system

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

- 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

- Prerequisites:
  \[ \dot{y} = Ay + f(t, y), \quad y(t) \in \mathbb{R}^m, \quad y(0) = y_0, \quad t \in [0, T] \]
  \[ y(t) \in V = \text{span}\{\psi_j\}_{j=1}^d \quad \forall t \in [0, T] \]
  \[ \Psi = \{\psi_j\}_{j=1}^d \ldots \text{orthonormal basis} \]
  \[ y(t) = \sum_{j=1}^d \langle y(t), \psi_j \rangle_W \psi_j, \quad \forall t \in [0, T], \quad W \ldots \text{appropriate weights} \]

- Ansatz for Galerkin projection, \( \ell < d \)
  \[ y^\ell(t) := \sum_{j=1}^\ell \langle y^\ell(t), \psi_j \rangle_W \psi_j, \quad \forall t \in [0, T], \quad \eta^\ell_j(t) := \langle y^\ell(t), \psi_j \rangle_W \]

- Put the above together, !! \( \psi_j \in \mathbb{R}^m, \quad j = 1, \ldots, \ell, \quad m > \ell \) !!
  \[ \sum_{j=1}^\ell \eta^\ell_j \psi_j = \sum_{j=1}^\ell \eta^\ell_j A\psi_j + f(t, y^\ell(t)), \quad t \in (0, T) \]
  \[ y_0 = \sum_{j=1}^\ell \eta^\ell_j(0) \psi_j \]
Introduce the reduced-order model

- Assume, that the above holds after projection on $V^\ell = \text{span}\{\psi_j\}_{j=1}^{\ell}$, 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 $A^\ell = (a^\ell_{ij}) \in \mathbb{R}^{l \times l}$ with $a^\ell_{ij} = \langle A\psi_j, \psi_i \rangle_W$
- Define the vector valued mapping $\eta^\ell = (\eta_1^\ell, \ldots, \eta_l^\ell)^T : [0, T] \rightarrow \mathbb{R}^l$
- Define the non-linearity $f^\ell = (f_1^\ell, \ldots, f_l^\ell)^T : [0, T] \rightarrow \mathbb{R}^l$, 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, $\eta^\ell(0) = \eta_0^\ell = (\langle y_0, \psi_1 \rangle_W, \ldots, \langle y_0, \psi_1 \rangle_W)^T$
- Write the ROM, $\dot{\eta}^\ell = A^\ell \eta^\ell + f^\ell(t, \eta^\ell)$, for $t \in (0, T]$, $\eta^\ell(0) = \eta_0^\ell$
Where to get a suitable base \( \{ \psi_j \}_{j=1}^d \)?

Discrete version of Proper orthogonal decomposition

Original system

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

Solution snapshots ← Approximation obtained from FOM

\[
S = \left\{ y_j = y(t_j) = e^{At_j} y_0 + \int_0^{t_j} e^{A(t_j-s)} b(s, y(s)) \, ds \right\}_{j=1}^n \approx \tilde{S} ← \text{FOM}
\]

Matrix of snapshots (tildes denoting approximate solutions are omitted)

\[
Y = [y_1, \ldots, y_n] \in \mathbb{R}^{m \times n}, \quad \text{rank}(Y) = d \leq \min\{m, n\},
\]
Where to get a suitable base $\{\psi_j\}_{j=1}^d$?

Discrete version of Proper orthogonal decomposition

Goal

Approximate all the spatial coordinate vectors $y_j$ of $Y$ simultaneously by $\ell \leq d$ normalized vectors as well as possible.

\[
\max_{\tilde{\psi}_1, \ldots, \tilde{\psi}_\ell \in \mathbb{R}^m} \sum_{i=1}^\ell \sum_{j=1}^n \left| \langle y_j, \tilde{\psi}_i \rangle_{\mathbb{R}^m} \right|^2
\]

subject to

\[
\langle \tilde{\psi}_i, \tilde{\psi}_j \rangle_{\mathbb{R}^m} = \delta_{ij} \text{ for } 1 \leq i, j \leq \ell,
\]
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 $\Psi = [\psi_1, \ldots, \psi_m] \in \mathbb{R}^{m \times m}$ and $\Phi = [\phi_1, \ldots, \phi_n] \in \mathbb{R}^{n \times n}$ are orthogonal matrices and the matrix $\Sigma$ has the structure of

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

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 $\{\psi_i\}_{i=1}^{\ell}$, i.e. by the first $\ell$ columns of $\Psi$. Moreover,

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

Proof

- Obtained via Lagrange framework
- Rather long and technical, can be found in literature (e.g. [VolkweinBook])
Proper Orthogonal Decomposition
SVD based technique for model order reduction

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

Notes:
- All the operations on $W$ have to be cheap, including its inversion.
- Do not perform the full SVD, $\Sigma \in \mathbb{R}^{d \times d}$, $d = \text{rank}(\tilde{Y})$. 
Deal with the non-linearities I

- Identify the problem,

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

- Approximate the non-linearities via the POD basis, \( \Phi \),

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

- Approximate \( f^\ell(t, \eta^\ell) \) through \( \Psi, W, \Phi \),

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

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

\[ P := [e_{i1}, \ldots, e_{ip}] \in \mathbb{R}^{m \times p}, \quad e_{ik} = (0, \ldots, 0, 1, 0, \ldots, 0)^T \in \mathbb{R}^m \]
Deal with the non-linearities II (yes, almost done)

- Plug in the matrix $P$,
  \[ P^T \Phi c(t) \approx P^T b(t), \quad \iff c(t) \in \mathbb{R}^p, \Phi \in \mathbb{R}^{m \times p}, b(t) \in \mathbb{R}^m \]

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

- If $f(t, \Psi \eta^\ell)$ is pointwise evaluable,
  \[ (P^T \Phi)^{-1} P^T f(t, \Psi \eta^\ell) = (P^T \Phi)^{-1} f(t, P^T \Psi \eta^\ell), \quad P^T \Psi \eta^\ell \in \mathbb{R}^p \]

- Write the final ROM
  \[ \dot{\eta}^\ell = A^\ell \eta^\ell + f^\ell(t, \eta^\ell), \quad \text{for } t \in (0, T], \quad \eta^\ell(0) = \eta_0^\ell, \]

where
  \[ f^\ell(t, \eta^\ell) = \Psi^T W \Phi (P^T \Phi)^{-1} f(t, P^T \Psi \eta^\ell) \]
Algorithm 2 DEIM

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

1. Compute POD basis $\Phi = [\phi_1, \ldots, \phi_p]$ for $F$
2. $\text{idx} \leftarrow \arg \max_{j=1,\ldots,m} |(\phi_1)_j|$
3. $U = [\phi_1]$ and $\vec{i} = \text{idx}$
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. $\text{idx} \leftarrow \arg \max_{j=1,\ldots,m} |(r)_j|$
   9. $U \leftarrow [U, u]$ and $\vec{i} \leftarrow [\vec{i}, \text{idx}]$
5. **end for**
6. **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.
<table>
<thead>
<tr>
<th>Introduction</th>
<th>POD &amp; DEIM</th>
<th>Link with OpenFOAM</th>
<th>Applications</th>
<th>Conclusions</th>
<th>Discussion</th>
</tr>
</thead>
</table>

Link with OpenFOAM
Rewrite OpenFOAM discretization as above studied problem

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

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

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

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

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

- 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
Modifications to POD-DEIM ROM creation
Extended snapshots and interpolation

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,

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

\[
\hat{A}^\ell(t) = \Psi^T W \hat{A}(t) \Psi = \Psi^T W (\varpi(t)A_{i-1} + (1 - \varpi(t))A_i) \Psi = \\
= \varpi(t)\Psi^T WA_{i-1} \Psi + (1 - \varpi(t))\Psi^T WA_i \Psi = \varpi(t)A_{i-1}^\ell + (1 - \varpi(t))A_i^\ell
\]

- Same trick can be done for \( b(t, y) \) and after the ROM creation, I do not need to store the full data.
**Example 1 – Passive scalar advection**

Phase-volume fraction advection in multiphase flow

**interFoam – Volume-of-Fluid model for multiphase flow**

\[
\begin{align*}
\alpha_t + \nabla \cdot (u \alpha) + \nabla \cdot (u_r \alpha (1 - \alpha)) &= 0 \\
\alpha_t + \mathcal{L}^h_{\alpha}(t, \alpha) &= 0 \rightarrow \alpha_t = A_{\alpha}(t)\alpha + b_{\alpha}(t, \alpha) \rightarrow \dot{\eta}_{\alpha} = \dot{A}_{\alpha}(t)\eta_{\alpha} + \dot{b}_{\alpha}(t, \eta_{\alpha})
\end{align*}
\]

Wanted:

\[\dot{y}_{\alpha} = A_{\alpha}(t)y_{\alpha} + b_{\alpha}(t, y)\]

**Example of implementation in OpenFOAM**

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

**Link:**

\[fvm \rightarrow A_{\alpha}(t), \ fvc \rightarrow b_{\alpha}(t, y)\]
Example 1 – Passive scalar advection

Numerical results

Time: 1.000 s
Example 1 – Passive scalar advection

Numerical results

\[ \varepsilon_R := \frac{1}{m \left( \max \alpha_{CFD} - \min \alpha_{CFD} \right)} \sum_{i=1}^{m} |\alpha_{i,CFD} - \alpha_{i,ROM}| \]
Example 1 – Passive scalar advection
Numerical results

Modes 1,2,3

Modes 2,3,4

Modes 3,4,5

Modes 4,5,6

Modes 5,6,7

Modes 6,7,8
Saddle-point problem

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

Jacobi iterations with Schur-complement based p-U coupling

\[ u^* \leftarrow A u^* = f - B^T p^{k-1} \]
\[ p^k \leftarrow B D^{-1} B^T p^k = B D^{-1} (f - (L + U) u^*) \]
\[ u^k \leftarrow D^{-1} (f - (L + U) u^* - B^T p^k) \]

At convergence

\[ B D^{-1} B^T p^k = B D^{-1} (f - (L + U) u^*) \approx B A^{-1} B^T p = B A^{-1} f \]
\[ u = D^{-1} (f - (L + U) u^*) - D^{-1} B^T p \]

Outcome for ROM

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

\[ D^{-1} \rightarrow rAU \quad \text{and} \quad D^{-1}(f - (L + U)u^*) \rightarrow HbyA \quad (\text{in oF, } *\text{Eqn.A}() \rightarrow D) \]

### Implementation of pressure equation in OpenFOAM

\[
\text{fvm} :: \text{laplacian}(rAU, p) \quad \Rightarrow \quad \text{fvc} :: \text{div}(HbyA)
\]

**Wanted:**

\[ \dot{y}_p = A_p(t)y_p + b_p(t, y_p) \]

### Implicit definition of time derivative for pressure

\[
\begin{align*}
\nabla \cdot (u \otimes u) - \nabla \cdot (\nu \nabla u) &= -\nabla \tilde{p} + \tilde{f} & & \text{UEqnMORE} \\
\nabla \cdot u &= 0 & & \Rightarrow \quad D_h^{-1} \rightarrow rAUMORE \\

fvm :: \text{laplacian}(rAUMORE, p) & \Rightarrow \quad \text{fvc} :: \text{div}(HMOREbyAMORE)
\end{align*}
\]

**Link:**

\[ \text{fvm} \rightarrow A_p(t), \text{fvc} \rightarrow b_p(t, y_p) \]
Reconstruction of the velocity field
Create ROM or expand snapshot

### Expansion of snapshots for pressure

**Standard approach snapshots:**

\[ S = \{(y_{k,i}, A_{k,i}, b_{k,i})\}_{i=1}^{n}, \, k = p, U \]  

**Expanded snapshots for pressure:**

\[ S^e = \{(y_{p,i}, A_{p,i}, b_{p,i}, r_{AUMORE_i}, HMOREbyAMORE_i)\}_{i=1}^{n} \]

### Storage

<table>
<thead>
<tr>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \ldots n[(1 + 3)m + (5 + 5)m + (1 + 3)m] \approx 15nm ) values</td>
</tr>
<tr>
<td>( S^e \ldots n(m + 5m + m + 1m + 3m) \approx 11nm ) values</td>
</tr>
</tbody>
</table>

### Computational cost

<table>
<thead>
<tr>
<th>Computational cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \ldots \sim 4n ) calculations of ( \Psi^TWA(t)\Psi ), evaluation of ( \sim 4 ) ROMs</td>
</tr>
<tr>
<td>( S^e \ldots )</td>
</tr>
<tr>
<td>( \sim n ) calculations of ( \Psi^TWA(t)\Psi ),</td>
</tr>
<tr>
<td>( \sim n ) calculations of ( \Psi^TWR_{AUMORE_i}\Psi ),</td>
</tr>
<tr>
<td>( \sim n ) calculations of ( \Psi^TWHMOREbyAMORE_j\Psi ),</td>
</tr>
<tr>
<td>evaluation of 1 ROM + interpolation between ( r_{AUMORE_i}^{ROM} ) and between ( HMOREbyAMORE_i^{ROM} )</td>
</tr>
<tr>
<td>( U_i \approx HMOREbyAMORE^{ROM} + r_{AUMORE^{ROM}}N_{ROM}^{ROM} )</td>
</tr>
</tbody>
</table>
Example 2 – Von Karman vortex street
Validation of the approach – incompressible single phase flow

\[ t = 15.00 \text{ [s]} \]

\( \tilde{p} \) [m² s⁻²]

\( ||\mathbf{U}|| \) [m s⁻¹]
Example 2 – Von Karman vortex street
Validation of the approach – incompressible single phase flow

\[ \varepsilon_R \] isozm(at)it.cas.cz

Simulation time [s]
Example 3 – 2D mixer
Validation of the approach – arbitrary mesh interface

\[ \|\mathbf{U}\| \text{ [m s}^{-1}] \]

\[ \tilde{p} \text{ [m}^2\text{s}^{-2}] \]

\[ t = 10.00 \text{ [s]} \]
Example 3 – 2D mixer
Validation of the approach – arbitrary mesh interface
Example 4 – 2D mixer
Validation of the approach – multiple reference frames

ROM

FOM

\[ \| \mathbf{U} \| \text{ [m s}^{-1}] \]

\[ t = 10.00 \text{ [s]} \]
Example 4 – 2D mixer
Validation of the approach – multiple reference frames

\[ \varepsilon_R \]
Example 5 – Sliding drop
Validation of the approach – multiphase flow

\[ \alpha_L \]

\[ t = 0.2947 \, [s] \]
plate inclination = \( \pi/3 \)

\[ \|U\| \, [\text{m/s}] \]

ROM

FOM
Example 5 – Sliding drop
Validation of the approach – multiphase flow

\[ \frac{\varepsilon_R}{\varepsilon_p} \text{ isozm(at)it.cas.cz} \]

COMPDYN'17, Rhodos, June 15 - June 17, 2017. POD and DEIM in CFD applications
Applications
Real-life applications
ROM is a tremendous tool for parametric studies or repeated model evaluations

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 → 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]

\[ \Delta p_h := \frac{p_{\text{above}} - p_{\text{bellow}}}{h_{\text{packing}}} \]

\[ \Delta p_h, [\text{Pa m}^{-1}] \]

\[ u_i, [\text{m s}^{-1}] \]
Comparison with experimental data: [Haidl, J. UCT Prague]
**Full case:** Flow through the Mellapak 250.X packing

![Graph showing number of iterations vs. problem inlet velocity.](image-url)

- **Reference simulations**
- **ROM predicted IC, L1**
- **ROM predicted IC, L2**
**Semi-industrial scale application**

ROM based initial guess prediction for full NS solver (simpleFoam)

**Full case:** Predicted vs. converged solution in L1

$$U_0 = (-0.3, 0, 0), \quad [\text{m s}^{-1}]$$
**Full case:** Predicted vs. converged solution in L1

\[ U_0 = (-0.3, 0, 0), \ [\text{m s}^{-1}] \]
Semi-industrial scale application
ROM based initial guess prediction for full NS solver (simpleFoam)

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)

![Graph showing solution quality comparison](image-url)
Solution quality: Comparison of RO and Full models results
Conclusions
Conclusions

POD-DEIM ROMs backed by OpenFOAM CFD machinery

Currently available

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

Advantages

- Snapshots 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 snapshots have to be stored - a lot of data
- Creation of $A_i^\ell$, $i = 1, \ldots, n$ is time consuming
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.


Thank you for your attention
Next steps
Towards ROM size reduction and multiparametric systems

ROM size reduction: $S^e$ selection based on greedy algorithm

$Re = 5000 \; \text{[-]}$
Next steps
Towards ROM size reduction and multiparametric systems

ROM size reduction: $S^e$ selection based on greedy algorithm

![Graph showing Iteration 0 with local and maxVal markers.](image)

- Continuity errors
- Problem Reynolds number, $[-]$
Next steps
Towards ROM size reduction and multiparametric systems

**ROM size reduction:** $S^e$ selection based on greedy algorithm

### Iteration 1

- **Continuity errors**
  - Problem Reynolds number $[10^{-4}, 10^4]$
  - Continuity errors $[10^{-4}, 10^3]$

- **Cost function**
  - Problem Reynolds number $[10^{-1}, 10^4]$
  - Cost function $[10^{-1}, 10^3]$

---

isozm(at)it.cas.cz

COMPDYN'17, Rhodos, June 15 - June 17, 2017. POD and DEIM in CFD applications
**Next steps**
Towards ROM size reduction and multiparametric systems

**ROM size reduction:** $S^e$ selection based on greedy algorithm

**Iteration 2**

- **Continuity errors**
- **Problem Reynolds number, $[-]$**

- **Cost function**
- **Problem Reynolds number, $[-]$**
Next steps
Towards ROM size reduction and multiparametric systems

**ROM size reduction:** $S^e$ selection based on greedy algorithm

**Iteration 3**

![Graph showing Continuity errors and Cost function over Problem Reynolds number for Iteration 3.](image)
Next steps
Towards ROM size reduction and multiparametric systems

ROM size reduction: $S^e$ selection based on greedy algorithm

Iteration 4

Problem Reynolds number, $[-]$

Continuity errors

Cost function

maxVal

Problem Reynolds number, $[-]$

isozm(at)it.cas.cz

COMPDYN'17, Rhodos, June 15 - June 17, 2017, POD and DEIM in CFD applications
**Next steps**
Towards ROM size reduction and multiparametric systems

**ROM size reduction:** $S^e$ selection based on greedy algorithm

**Iteration 5**

- Continuity errors
- Problem Reynolds number, $[-]$:
  - $10^{-4}$
  - $10^{-3}$
  - $10^{-2}$
  - $10^{-1}$
  - $10^0$

- Cost function
- Problem Reynolds number, $[-]$:
  - $10^{-4}$
  - $10^{-3}$
  - $10^{-2}$
  - $10^{-1}$
  - $10^0$

Local and maxVal errors over iterations.
Next steps
Towards ROM size reduction and multiparametric systems

**ROM size reduction:** $S^e$ selection based on greedy algorithm

![Graph showing ROM size reduction](image)

- Reference simulations
- ROM predicted IC

<table>
<thead>
<tr>
<th>Problem Reynolds number</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

POD and DEIM in CFD applications

COMPDYN’17, Rhodos, June 15 - June 17, 2017
Next steps
Towards ROM size reduction and multiparametric systems

ROM size reduction: $S^e$ selection based on greedy algorithm

![Graph showing ROM size reduction and greedy algorithm selection](image-url)
Better snapshot selection
Optimal selection of snapshot to include into POD basis

\[ \sum_{i=1}^{m} |\nabla \cdot u_i| \text{ [s}^{-1}] \]

\[ \text{Re} [-] \]

\[ S [-] \]

\[ \text{Re} [-] \]
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,**

\[ \langle \varphi, \tilde{\varphi} \rangle_{L^2(\Omega)} = \int_{\Omega} \varphi \tilde{\varphi} \, dx, \quad ||\varphi||_{L^2(\Omega)} = \sqrt{\langle \varphi, \varphi \rangle_{L^2(\Omega)}} \]

**Denote** \( \Omega^h \) **a FVM discretization of** \( \Omega \) **and** \( \delta \Omega_i^h \) **the volume of the** \( i \)-**th cell,**

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

**Introduce a discrete inner product,** \( \langle \varphi, \tilde{\varphi} \rangle_{L^2_h} \),

\[ \langle \varphi, \tilde{\varphi} \rangle_{L^2(\Omega)} = \int_{\Omega} \varphi \tilde{\varphi} \, dx \approx \sum_{i=1}^{\text{nCells}} \int_{\Omega_i^h} \varphi \tilde{\varphi} \, dx = \sum_{i=1}^{\text{nCells}} \varphi_i^h \tilde{\varphi}_i^h \delta \Omega_i^h = \langle \varphi, \tilde{\varphi} \rangle_{L^2_h} \]

**Denote** \( W = \text{diag}(\delta \Omega_1^h, \ldots, \delta \Omega_{\text{nCells}}^h) \). **Hence,** \( \langle \varphi, \tilde{\varphi} \rangle_{L^2_h} = (\varphi^h)^T W \varphi^h \).
Full case: Residuals evolution, from potentialFoam initialized fields

Altix UV 2000, 4 cores, 3000000.0MM cells, case: sF_u0_2.4_Mellapak250XV1, solver: simpleFoam
-parallel, version: v3.0+-e941ee6c15e9
Full case: Residuals evolution, from potentialFoam initialized fields

Altix UV 2000, 4 cores, 3000000.0MM cells, case: sF_u0_2.4_Mellapak250XV1, solver: simpleFoam -parallel, version: v3.0+-e941ee6c15e9
**Full case:** Residuals evolution, from ROM predicted fields, L1

Altix UV 2000, 4 cores, 3000000.0MM cells, case: sF_u0_2.4_ROM, solver: simpleFoam -parallel, version: v3.0+-e941ee6c15e9
Residuals evolution
Comparison of the residuals evolution for Mellapak cases in L0, L1 and L2

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

Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz, 4 cores, 3000000.0MM cells, case: sF_u0_1.5_ROM2, solver: simpleFoam -parallel, version: v3.0+-e941ee6c15e9