AA216/CME345: MODEL REDUCTION

Methods for Nonlinear Systems

Charbel Farhat
Stanford University
cfarhat@stanford.edu

Outline

1 Nested Approximations

2 Trajectory PieceWise Linear (TPWL) Method

3 Hyperreduction Methods

4 Local Approaches

5 References

- HDM of interest

$$\begin{aligned} \frac{d\mathbf{w}}{dt}(t;\boldsymbol{\mu}) &= \mathbf{f}\left(\mathbf{w}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}\right) \\ \mathbf{y}(t;\boldsymbol{\mu}) &= \mathbf{g}\left(\mathbf{w}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}\right) \end{aligned}$$

  - $\mathbf{w} \in \mathbb{R}^N$: Vector of state variables
  - $\mathbf{u} \in \mathbb{R}^p$: Vector of input variables – typically, $p \ll N$
  - $\mathbf{y} \in \mathbb{R}^q$: Vector of output variables – typically, $q \ll N$
  - $\boldsymbol{\mu} \in \mathbb{R}^m$: Vector of parameter variables – typically, $m \ll N$
  - f: **Nonlinear** function
- Usually, there is no closed form solution for $\mathbf{w}(t;\boldsymbol{\mu})$

- Approximation of the state using a right ROB

$$\mathbf{w}(t;\boldsymbol{\mu}) \approx \tilde{\mathbf{w}}(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{q}(t;\boldsymbol{\mu})$$

- Resulting nonlinear ODE

$$\mathbf{V}\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) + \mathbf{r}(t;\boldsymbol{\mu})$$

- Enforcement of the orthogonality of the residual $\mathbf{r}$ to a left ROB $\mathbf{W}$

$$\mathbf{W}^T\mathbf{V}\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$$

- If $\mathbf{W}^T\mathbf{V}$ is nonsingular, the above equation can be re-written as

$$\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$$

- Petrov-Galerkin PROM

$$\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \underbrace{(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T}_{k\times N}\underbrace{\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}_{N\times 1}$$

- Petrov-Galerkin PROM

$$\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \underbrace{(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T}_{k\times N}\underbrace{\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}_{N\times 1}$$

- k equations with k unknowns

- Petrov-Galerkin PROM

$$\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \underbrace{(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T}_{k\times N}\underbrace{\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}_{N\times 1}$$

- k equations with k unknowns
- For a given reduced state vector $\mathbf{q}(t;\boldsymbol{\mu})$, the evaluation of $\mathbf{f}_k(\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t,\boldsymbol{\mu}) = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$ at a given time t and a given parameter vector $\boldsymbol{\mu}$ can be performed in 3 steps as follows
  1. compute $\mathbf{w}(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{q}(t;\boldsymbol{\mu})$
  2. evaluate $\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$
  3. left-multiply the result by $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T$ to obtain $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t),t)$

- Petrov-Galerkin PROM

$$\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \underbrace{(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T}_{k\times N}\underbrace{\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}_{N\times 1}$$

- k equations with k unknowns
- For a given reduced state vector $\mathbf{q}(t;\boldsymbol{\mu})$, the evaluation of $\mathbf{f}_k(\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t,\boldsymbol{\mu}) = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$ at a given time t and a given parameter vector $\boldsymbol{\mu}$ can be performed in 3 steps as follows
  1. compute $\mathbf{w}(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{q}(t;\boldsymbol{\mu})$
  2. evaluate $\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$
  3. left-multiply the result by $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T$ to obtain $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t),t)$
- The computational cost associated with the three steps described above **scales linearly with the dimension N of the HDM**

- Petrov-Galerkin PROM

$$\frac{d\mathbf{q}}{dt}(t;\boldsymbol{\mu}) = \underbrace{(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T}_{k\times N}\underbrace{\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}_{N\times 1}$$

- k equations with k unknowns
- For a given reduced state vector $\mathbf{q}(t;\boldsymbol{\mu})$, the evaluation of $\mathbf{f}_k(\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t,\boldsymbol{\mu}) = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$ at a given time t and a given parameter vector $\boldsymbol{\mu}$ can be performed in 3 steps as follows
  1. compute $\mathbf{w}(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{q}(t;\boldsymbol{\mu})$
  2. evaluate $\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$
  3. left-multiply the result by $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T$ to obtain $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t),t)$
- The computational cost associated with the three steps described above **scales linearly with the dimension N of the HDM**
- Hence, for nonlinear problems, dimensional reduction as described so far does not necessarily lead to significant CPU time reduction

- In this case, an **additional level of approximation** is required to ensure that the online cost associated with solving the reduced nonlinear equations does not scale with the dimension N of the HDM
- This leads to nested approximations
  - state approximation
  - nonlinear function approximation (approximate-then-project) or projection approximation (project-then-approximate $\leftarrow$ new!)
- There are two main classes of nonlinear function approximations
  - linearization approaches (TPWL, ManiMOR,...)
  - hyperreduction approaches (DEIM, ECSW, GNAT,...)

- Consider a nonlinear HDM of the form

$$\frac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t)) + \mathbf{B}\mathbf{u}(t)$$

  - stationary system
  - no parametric dependence for now
  - separable linear input
- For linear HDMs, reduced-order operators of the type

$$\mathbf{A}_r = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{V}$$

  can be pre-computed offline once for all
- Idea: linearize $\mathbf{f}$ around an operating point $\mathbf{w}_1$

$$\mathbf{f}(\mathbf{w}) \approx \mathbf{f}(\mathbf{w}_1) + \frac{\partial \mathbf{f}}{\partial \mathbf{w}}(\mathbf{w}_1)(\mathbf{w} - \mathbf{w}_1) = \mathbf{f}(\mathbf{w}_1) + \mathbf{A}(\mathbf{w}_1)(\mathbf{w} - \mathbf{w}_1)$$

- Then, the resulting approximated system is **linear** in the state $\mathbf{w}(t)$

$$\frac{d\mathbf{w}}{dt}(t) \approx \mathbf{A}(\mathbf{w}_1)\mathbf{w}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1$$

- Approximated HDM system

$$\frac{d\mathbf{w}}{dt}(t) \approx \mathbf{A}(\mathbf{w}_1)\mathbf{w}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1$$

- Reduced-order system after Petrov-Galerkin projection

$$\begin{aligned}\frac{d\mathbf{q}}{dt}(t) &= (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{A}(\mathbf{w}_1)\mathbf{V}\mathbf{q}(t) \\ &\quad + (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\left(\mathbf{B}\mathbf{u}(t) + \mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1\right)\end{aligned}$$

- The following linear time-invariant operators can be pre-computed
  - $\mathbf{A}_r = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{A}(\mathbf{w}_1)\mathbf{V} \in \mathbb{R}^{k\times k}$
  - $\mathbf{B}_r = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{B} \in \mathbb{R}^{k\times p}$
  - $\mathbf{F}_r = (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\left(\mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1\right) \in \mathbb{R}^{k}$

- Idea: Linearize the nonlinear function **at multiple locations in the state space**
- Extend the domain of validity of the linearization assumptions
- Approximated high-dimensional dynamical system

$$\begin{aligned}\frac{d\mathbf{w}}{dt}(t) &\approx \sum_{i=1}^{s} \omega_i(\mathbf{w}(t))(\mathbf{f}(\mathbf{w}_i) + \mathbf{A}_i(\mathbf{w}(t) - \mathbf{w}_i)) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{w}(t), \mathbf{u}(t), t)\end{aligned}$$

  - the s points $\{\mathbf{w}_i\}_{i=1}^{s}$ are linearization points
  - the s coefficients $\{\omega_i\}_{i=1}^{s}$ are weights such that

$$\sum_{i=1}^{s} \omega_i(\mathbf{w}) = 1, \ \forall \mathbf{w} \in \mathbb{R}^N$$

- For simplicity, assume $\mathbf{W}^T\mathbf{V} = \mathbf{I}_k$: In this case, the PROM obtained via Petrov-Galerkin projection is

$$\begin{aligned}\frac{d\mathbf{q}}{dt}(t) &= \sum_{i=1}^{s} \tilde{\omega}_i(\mathbf{q}(t))(\mathbf{W}^T\mathbf{f}(\mathbf{w}_i) + \mathbf{W}^T\mathbf{A}_i(\mathbf{V}\mathbf{q}(t) - \mathbf{w}_i)) + \mathbf{W}^T\mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{V}\mathbf{q}(t), \mathbf{u}(t), t)\end{aligned}$$

where

$$\sum_{i=1}^{s} \tilde{\omega}_i(\mathbf{q}) = 1, \ \forall \mathbf{q} \in \mathbb{R}^k$$

- Equivalently

$$\frac{d\mathbf{q}}{dt}(t) = \left(\sum_{i=1}^{s} \tilde{\omega}_i(\mathbf{q}(t))\mathbf{A}_{r_i}\right)\mathbf{q}(t) + \left(\sum_{i=1}^{s} \tilde{\omega}_i(\mathbf{q}(t))\right)\mathbf{F}_{r_i} + \mathbf{B}_r\mathbf{u}(t)$$

  - $\mathbf{A}_{r_i} = \mathbf{W}^T\mathbf{A}_i\mathbf{V}, \ i = 1, \cdots, s$
  - $\mathbf{B}_r = \mathbf{W}^T\mathbf{B}$
  - $\mathbf{F}_{r_i} = \mathbf{W}^T(\mathbf{f}(\mathbf{w}_i) - \mathbf{A}_i\mathbf{w}_i), \ i = 1, \cdots, s$

- In this context, a complete Projection-based Model Order Reduction (PMOR) method should incorporate algorithms for
  - selecting the linearization points $\{\mathbf{w}_i\}_{i=1}^{s}$
  - selecting the ROBs $\mathbf{V}$ and $\mathbf{W}$
  - determining the weights $\{\widetilde{\omega}_i(\mathbf{q})\}_{i=1}^{s}, \ \forall \mathbf{q} \in \mathbb{R}^k$

- Note that each linear approximation of the nonlinear function $\mathbf{f}$ is valid only in a neighborhood of each $\mathbf{w}_i$
- Note also that, in practice, it is impossible to cover the entire state-space $\mathbb{R}^N$ by local linear approximations
- The Trajectory PieceWise Linear (TPWL) PMOR method (2001)
  - uses pre-computed trajectories of the HDM (offline) to select the linearization regions
  - selects an additional linearization point from the HDM trajectory if it is sufficiently far away from the previously selected points

- Possible methods for constructing a **global** basis $\mathbf{V}$ include
  - if the input function is linear in $\mathbf{u}$, constructing Krylov subspaces $\mathcal{K}_i = \mathcal{K}(\mathbf{A}_i^{-1}, \mathbf{A}_i^{-1}\mathbf{B}) = \text{range}(\mathbf{V}_i)$ at each linearization point $\mathbf{w}_i$ and assembling a global basis $\mathbf{V}$ such that

$$\text{range}(\mathbf{V}) = \text{range}\left(\begin{bmatrix}\mathbf{V}_1 & \cdots & \mathbf{V}_s\end{bmatrix}\right)$$

  - ad-hoc methods (Balanced truncation, POD...)
- The left ROB $\mathbf{W}$ can be chosen based on the output of interest (two-sided Krylov moment matching), or simply as $\mathbf{W} = \mathbf{V}$ (Galerkin projection)

- The weights are used to characterize in the reduced space $\mathbb{R}^k$ the distance of the current point $\mathbf{q}(t)$ to the projection of the linearization points onto range($\mathbf{V}$) – that is, $\left\{\mathbf{q}_i = \left(\mathbf{V}^T\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{w}_i\right\}_{i=1}^{s}$
  - one possible choice is

$$\tilde{\omega}_i(\mathbf{q}) = \frac{\exp\left(-\frac{\beta d_i^2}{m^2}\right)}{\sum_{j=1}^{s}\exp\left(-\frac{\beta d_j^2}{m^2}\right)}$$

    where β is a constant, $d_i = \|\mathbf{q} - \mathbf{q}_i\|_2$, and $m = \min_{j=1}^{s} d_j$
  - other choices can be found in the literature

- A posteriori error estimators are available when $\mathbf{f}$ is negative monotone
- Stability guarantee is possible under some assumptions on $\mathbf{f}$ and specific choices for $\mathbf{V}$ and the weights $\{\tilde{\omega}_i(\mathbf{q})\}_{i=1}^{s}$
- Passivity preservation (i.e. no energy creation in a passive system) is possible under similar assumptions
- TPWL using local ROBs (ManiMOR)

Weaknesses

Strengths

- The cost of the online phase does not scale with the size N of the HDM

Strengths

- The cost of the online phase does not scale with the size N of the HDM

Weaknesses

- It is essential to choose good linearization points offline

Strengths

- The cost of the online phase does not scale with the size N of the HDM
- The online phase is not software-intrusive

Weaknesses

- It is essential to choose good linearization points offline

Strengths

- The cost of the online phase does not scale with the size N of the HDM
- The online phase is not software-intrusive

Weaknesses

- It is essential to choose good linearization points offline
- Requires the extraction of Jacobians from the HDM software

Strengths

- The cost of the online phase does not scale with the size N of the HDM
- The online phase is not software-intrusive

Weaknesses

- It is essential to choose good linearization points offline
- Requires the extraction of Jacobians from the HDM software
- Many parameters to adjust (number of linearization points, weights, ...)

- First applied to face recognition (Emerson and Sirovich, "Karhunen-Loeve Procedure for Gappy Data", 1996)

- First applied to face recognition (Emerson and Sirovich, "Karhunen-Loeve Procedure for Gappy Data", 1996)
- Other applications
  - flow sensing and estimation
  - flow (approximate) reconstruction
  - nonlinear model order reduction

- Face recognition

Procedure

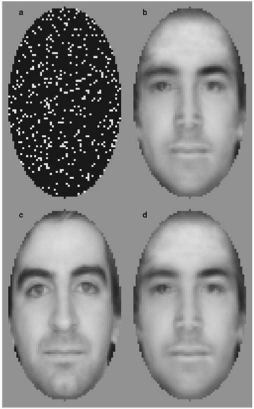


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions is shown in d.

- Face recognition

Procedure

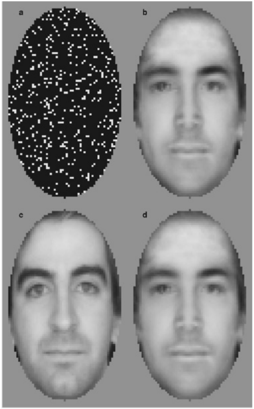


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions is shown in d.

- Face recognition

Procedure

1. build a database of N_s faces (snapshots)

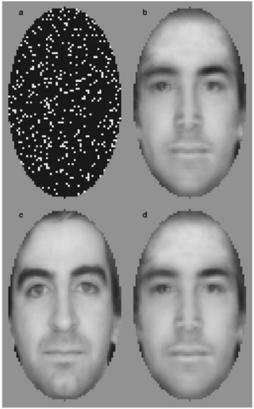


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions is shown in d.

- Face recognition

Procedure

1. build a database of N_s faces (snapshots)
2. construct a POD basis $\mathbf{V_f}$ for the database

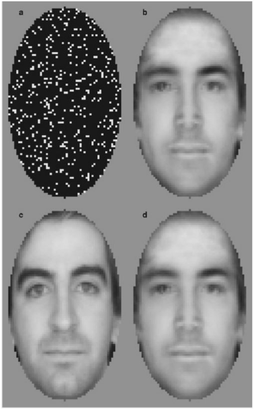


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions is shown in d.

- Face recognition

Procedure

1. build a database of N_s faces (snapshots)
2. construct a POD basis $\mathbf{V_f}$ for the database
3. for a new face $\mathbf{f}$, record a small number k_i of pixels $f_{i_1}, \cdots, f_{i_{k_i}}$

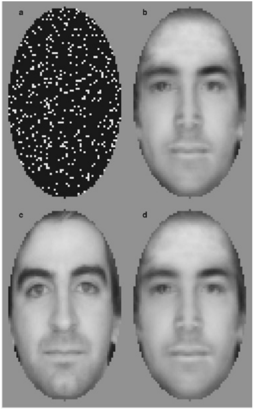


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions is shown in d.

- Face recognition

Procedure

1. build a database of N_s faces (snapshots)
2. construct a POD basis $\mathbf{V_f}$ for the database
3. for a new face $\mathbf{f}$, record a small number k_i of pixels $f_{i_1}, \cdots, f_{i_{k_i}}$
4. using the POD basis $\mathbf{V_f}$, approximately reconstruct the new face $\mathbf{f}$ (in the least-squares sense)

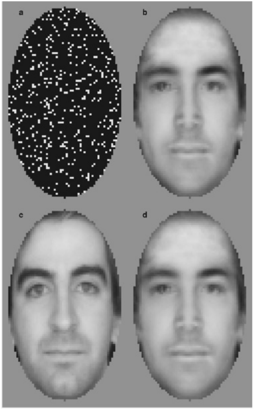


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions is shown in d.

- The gappy approach can also be used to approximate the nonlinear function $\mathbf{f}$ in the reduced equations

$$\frac{d\mathbf{q}}{dt}(t) = \mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

(for simplicity, the input function $\mathbf{u}(t)$ is not considered here)

- The gappy approach can also be used to approximate the nonlinear function $\mathbf{f}$ in the reduced equations

$$\frac{d\mathbf{q}}{dt}(t) = \mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

  (for simplicity, the input function $\mathbf{u}(t)$ is not considered here)
- The evaluation of all entries of $\mathbf{f}(\cdot, t)$ is computationally intensive (scales with N)

- The gappy approach can also be used to approximate the nonlinear function $\mathbf{f}$ in the reduced equations

$$\frac{d\mathbf{q}}{dt}(t) = \mathbf{W}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

(for simplicity, the input function $\mathbf{u}(t)$ is not considered here)

- The evaluation of all entries of $\mathbf{f}(\cdot, t)$ is computationally intensive (scales with N)
- Gappy approach
  - evaluate only a small subset of these entries
  - pre-compute a ROB $\mathbf{V_f}$ and use it to approximately reconstruct all other entries by interpolation or a least-squares strategy

- A complete PMOR method based on the Gappy approach for hyperreduction should then provide algorithms for
  - selecting the evaluation entries $\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$

- A complete PMOR method based on the Gappy approach for hyperreduction should then provide algorithms for
  - selecting the evaluation entries $\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$
  - pre-computing a ROB $\mathbf{V_f}$ for the nonlinear function $\mathbf{f}$

- A complete PMOR method based on the Gappy approach for hyperreduction should then provide algorithms for
  - selecting the evaluation entries $\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$
  - pre-computing a ROB $\mathbf{V_f}$ for the nonlinear function $\mathbf{f}$
  - approximately reconstructing the nonlinear function at all its other entries $\Rightarrow \hat{\mathbf{f}}(\cdot, t)$

- Construction of a POD basis $\mathbf{V_f}$ of dimension $k_\mathbf{f}$
  1. collect snapshots for the nonlinear function $\mathbf{f}$ from one or several transient simulations

$$\mathbf{F} = [\mathbf{f}(\mathbf{w}(t_1), t_1) \quad \cdots \quad \mathbf{f}(\mathbf{w}(t_{m_\mathbf{f}}), t_{m_\mathbf{f}})] \in \mathbb{R}^{N \times m_\mathbf{f}}$$

- Construction of a POD basis $\mathbf{V_f}$ of dimension $k_\mathbf{f}$
  1. collect snapshots for the nonlinear function $\mathbf{f}$ from one or several transient simulations

$$\mathbf{F} = [\mathbf{f}(\mathbf{w}(t_1), t_1) \quad \cdots \quad \mathbf{f}(\mathbf{w}(t_{m_\mathbf{f}}), t_{m_\mathbf{f}})] \in \mathbb{R}^{N \times m_\mathbf{f}}$$

  2. compute a thin SVD

$$\mathbf{F} = \mathbf{U_f} \mathbf{\Sigma_f} \mathbf{Z_f}^T$$

- Construction of a POD basis $\mathbf{V_f}$ of dimension $k_\mathbf{f}$
  1. collect snapshots for the nonlinear function $\mathbf{f}$ from one or several transient simulations

$$\mathbf{F} = [\mathbf{f}(\mathbf{w}(t_1), t_1) \quad \cdots \quad \mathbf{f}(\mathbf{w}(t_{m_\mathbf{f}}), t_{m_\mathbf{f}})] \in \mathbb{R}^{N \times m_\mathbf{f}}$$

  2. compute a thin SVD

$$\mathbf{F} = \mathbf{U_f \Sigma_f Z_f}^T$$

  3. construct a ROB of dimension $k_\mathbf{f} \leq m_\mathbf{f}$ as the set of first $k_\mathbf{f}$ vectors in $\mathbf{U_f}$ (truncation)

$$\mathbf{V_f} = [\mathbf{u}_{\mathbf{f},1} \quad \cdots \quad \mathbf{u}_{\mathbf{f},k_\mathbf{f}}]$$

- Assume for now that k_i indices (entries of $\mathbf{f}$) have been chosen (see later for how to choose these indices)

$$\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$$

- Assume for now that k_i indices (entries of $\mathbf{f}$) have been chosen (see later for how to choose these indices)

$$\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$$

- Consider the $N \times k_i$ "mask" matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{e}_{i_1} & \cdots & \mathbf{e}_{i_{k_i}} \end{bmatrix}$$

- Assume for now that k_i indices (entries of $\mathbf{f}$) have been chosen (see later for how to choose these indices)

$$\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$$

- Consider the $N \times k_i$ "mask" matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{e}_{i_1} & \cdots & \mathbf{e}_{i_{k_i}} \end{bmatrix}$$

- At each time t, given a value of the state approximation $\tilde{\mathbf{w}}(t) = \mathbf{V}\mathbf{q}(t)$, evaluate only those entries of $\mathbf{f}$ corresponding to the above indices

$$\begin{bmatrix} f_{i_1}(\tilde{\mathbf{w}}(t), t) \\ \vdots \\ f_{i_{k_i}}(\tilde{\mathbf{w}}(t), t) \end{bmatrix} = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

- Assume for now that k_i indices (entries of $\mathbf{f}$) have been chosen (see later for how to choose these indices)

$$\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$$

- Consider the $N \times k_i$ "mask" matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{e}_{i_1} & \cdots & \mathbf{e}_{i_{k_i}} \end{bmatrix}$$

- At each time t, given a value of the state approximation $\tilde{\mathbf{w}}(t) = \mathbf{V}\mathbf{q}(t)$, evaluate only those entries of $\mathbf{f}$ corresponding to the above indices

$$\begin{bmatrix} f_{i_1}(\tilde{\mathbf{w}}(t), t) \\ \vdots \\ f_{i_{k_i}}(\tilde{\mathbf{w}}(t), t) \end{bmatrix} = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

- This is computationally economical if $k_i \ll N$

- Assume for now that k_i indices (entries of $\mathbf{f}$) have been chosen (see later for how to choose these indices)

$$\mathcal{I} = \{i_1, \cdots, i_{k_i}\}$$

- Consider the $N \times k_i$ "mask" matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{e}_{i_1} & \cdots & \mathbf{e}_{i_{k_i}} \end{bmatrix}$$

- At each time t, given a value of the state approximation $\tilde{\mathbf{w}}(t) = \mathbf{V}\mathbf{q}(t)$, evaluate only those entries of $\mathbf{f}$ corresponding to the above indices

$$\begin{bmatrix} f_{i_1}(\tilde{\mathbf{w}}(t), t) \\ \vdots \\ f_{i_{k_i}}(\tilde{\mathbf{w}}(t), t) \end{bmatrix} = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

- This is computationally economical if $k_i \ll N$
- Usually, only a subset of the entries of $\tilde{\mathbf{w}}(t)$ are required to construct the above vector (case of a sparse Jacobian)

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, k_i$

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, k_i$
  - this means that

$$\mathbf{P}^T \hat{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, k_i$
  - this means that

$$\mathbf{P}^T \hat{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

  - recalling that $\hat{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_{\mathbf{f}}$ – that is,

$$\hat{\mathbf{f}}(\mathbf{V}\mathbf{q}(t), t) = \mathbf{V}_{\mathbf{f}} \mathbf{f}_r(\mathbf{q}(t), t), \text{ where } \mathbf{f}_r(\mathbf{q}(t), t) \in \mathbb{R}^{k_{\mathbf{f}}}$$

    it follows that

$$\mathbf{P}^T \mathbf{V}_{\mathbf{f}} \mathbf{f}_r(\mathbf{q}(t), t) = \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

- Case where $k_i = k_\mathbf{f} \Rightarrow$ interpolation
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, k_i$
  - this means that

$$\mathbf{P}^T \hat{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

  - recalling that $\hat{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_\mathbf{f}$ – that is,

$$\hat{\mathbf{f}}(\mathbf{V}\mathbf{q}(t), t) = \mathbf{V}_\mathbf{f}\mathbf{f}_r(\mathbf{q}(t), t), \ \text{where } \mathbf{f}_r(\mathbf{q}(t), t) \in \mathbb{R}^{k_\mathbf{f}}$$

    it follows that

$$\mathbf{P}^T \mathbf{V}_\mathbf{f} \mathbf{f}_r(\mathbf{q}(t), t) = \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

  - assuming that $\mathbf{P}^T \mathbf{V}_\mathbf{f}$ is nonsingular

$$\Longrightarrow \mathbf{f}_r(\mathbf{q}(t), t) = (\mathbf{P}^T \mathbf{V}_\mathbf{f})^{-1} \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \;\; \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \;\; \forall j = 1, \cdots, k_i$
  - this means that

$$\mathbf{P}^T \hat{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

  - recalling that $\hat{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_{\mathbf{f}}$ – that is,

$$\hat{\mathbf{f}}(\mathbf{V}\mathbf{q}(t), t) = \mathbf{V}_{\mathbf{f}} \mathbf{f}_r(\mathbf{q}(t), t), \;\; \text{where } \mathbf{f}_r(\mathbf{q}(t), t) \in \mathbb{R}^{k_{\mathbf{f}}}$$

    it follows that

$$\mathbf{P}^T \mathbf{V}_{\mathbf{f}} \mathbf{f}_r(\mathbf{q}(t), t) = \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

  - assuming that $\mathbf{P}^T \mathbf{V}_{\mathbf{f}}$ is nonsingular

$$\Longrightarrow \mathbf{f}_r(\mathbf{q}(t), t) = (\mathbf{P}^T \mathbf{V}_{\mathbf{f}})^{-1} \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

  - hence, the high-dimensional nonlinear function $\hat{\mathbf{f}}(\cdot, t)$ is interpolated as follows

$$\boxed{\hat{\mathbf{f}}(\cdot, t) = \mathbf{V}_{\mathbf{f}} (\mathbf{P}^T \mathbf{V}_{\mathbf{f}})^{-1} \mathbf{P}^T \mathbf{f}(\cdot, t) = \mathbf{\Pi}_{\mathbf{V}_{\mathbf{f}}, \mathbf{P}} \mathbf{f}(\cdot, t)}$$

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \;\forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \;\forall j = 1, \cdots, k_i$
  - this means that

$$\mathbf{P}^T \hat{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$$

  - recalling that $\hat{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_{\mathbf{f}}$ – that is,

$$\hat{\mathbf{f}}(\mathbf{V}\mathbf{q}(t), t) = \mathbf{V}_{\mathbf{f}} \mathbf{f}_r(\mathbf{q}(t), t), \text{ where } \mathbf{f}_r(\mathbf{q}(t), t) \in \mathbb{R}^{k_{\mathbf{f}}}$$

    it follows that

$$\mathbf{P}^T \mathbf{V}_{\mathbf{f}} \mathbf{f}_r(\mathbf{q}(t), t) = \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

  - assuming that $\mathbf{P}^T \mathbf{V}_{\mathbf{f}}$ is nonsingular

$$\Longrightarrow \mathbf{f}_r(\mathbf{q}(t), t) = (\mathbf{P}^T \mathbf{V}_{\mathbf{f}})^{-1} \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

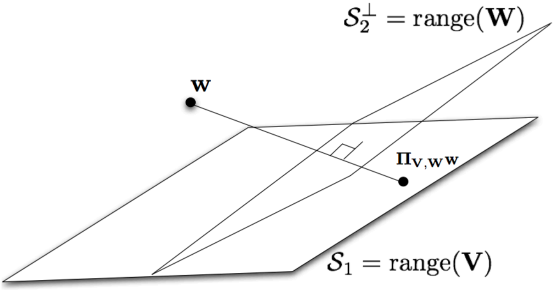
  - hence, the high-dimensional nonlinear function $\hat{\mathbf{f}}(\cdot, t)$ is interpolated as follows

$$\boxed{\hat{\mathbf{f}}(\cdot, t) = \mathbf{V}_{\mathbf{f}} (\mathbf{P}^T \mathbf{V}_{\mathbf{f}})^{-1} \mathbf{P}^T \mathbf{f}(\cdot, t) = \mathbf{\Pi}_{\mathbf{V}_{\mathbf{f}}, \mathbf{P}} \mathbf{f}(\cdot, t)}$$

  - interpretation: the Discrete Empirical Interpolation Method (DEIM) is an oblique projection of the high-dimensional nonlinear vector-valued function

$$\boxed{\hat{\mathbf{f}}(\cdot, t) = \mathbf{V_f}(\mathbf{P}^T\mathbf{V_f})^{-1}\mathbf{P}^T\mathbf{f}(\cdot, t) = \mathbf{\Pi}_{\mathbf{V_f},\mathbf{P}}\mathbf{f}(\cdot, t)}$$

- Recall that $\mathbf{\Pi}_{\mathbf{V},\mathbf{W}} = \mathbf{V}(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T$ is the oblique projector onto $\mathbf{V}$, orthogonally to $\mathbf{W}$



- Case where $k_i > k_{\mathbf{f}} \Rightarrow$ least-squares reconstruction
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, N$

- Case where $k_i > k_\mathbf{f} \Rightarrow$ least-squares reconstruction
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, N$
  - this leads to the minimization problem

$$\mathbf{f}_r(\mathbf{q}(t), t) = \underset{\mathbf{y}_r \in \mathbb{R}^{k_f}}{\operatorname{argmin}} \|\mathbf{P}^T \mathbf{V}_\mathbf{f} \mathbf{y}_r - \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)\|_2$$

- Case where $k_i > k_{\mathbf{f}} \Rightarrow$ least-squares reconstruction
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, N$
  - this leads to the minimization problem

$$\mathbf{f}_r(\mathbf{q}(t), t) = \underset{\mathbf{y}_r \in \mathbb{R}^{k_f}}{\text{argmin}} \|\mathbf{P}^T \mathbf{V}_{\mathbf{f}} \mathbf{y}_r - \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)\|_2$$

  - note that $\mathbf{M} = \mathbf{P}^T \mathbf{V}_{\mathbf{f}} \in \mathbb{R}^{k_i \times k_{\mathbf{f}}}$ is a skinny matrix

- Case where $k_i > k_\mathbf{f} \Rightarrow$ least-squares reconstruction
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, N$
  - this leads to the minimization problem

$$\mathbf{f}_r(\mathbf{q}(t), t) = \underset{\mathbf{y}_r \in \mathbb{R}^{k_f}}{\text{argmin}} \, \|\mathbf{P}^T \mathbf{V}_\mathbf{f} \mathbf{y}_r - \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)\|_2$$

  - note that $\mathbf{M} = \mathbf{P}^T \mathbf{V}_\mathbf{f} \in \mathbb{R}^{k_i \times k_\mathbf{f}}$ is a skinny matrix
  - its singular value decomposition can be written as

$$\mathbf{M} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{Z}^T$$

- Case where $k_i > k_{\mathbf{f}} \Rightarrow$ least-squares reconstruction
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, N$
  - this leads to the minimization problem

$$\mathbf{f}_r(\mathbf{q}(t), t) = \underset{\mathbf{y}_r \in \mathbb{R}^{k_f}}{\text{argmin}} \, \|\mathbf{P}^T \mathbf{V}_\mathbf{f} \mathbf{y}_r - \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)\|_2$$

  - note that $\mathbf{M} = \mathbf{P}^T \mathbf{V}_\mathbf{f} \in \mathbb{R}^{k_i \times k_\mathbf{f}}$ is a skinny matrix
  - its singular value decomposition can be written as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{Z}^T$$

  - then, the left inverse of $\mathbf{M}$ $\left((\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\right)$ is given by

$$\mathbf{M}^\dagger = \mathbf{Z}\mathbf{\Sigma}^\dagger\mathbf{U}^T$$

where $\mathbf{\Sigma}^\dagger = \text{diag}(\dfrac{1}{\sigma_1}, \cdots, \dfrac{1}{\sigma_r}, 0, \cdots, 0)$ if
$\mathbf{\Sigma} = \text{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$, where $\sigma_1 \geq \cdots \sigma_r > 0$

- Case where $k_i > k_\mathbf{f} \Rightarrow$ least-squares reconstruction
  - idea: $\hat{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall j = 1, \cdots, N$
  - this leads to the minimization problem

$$\mathbf{f}_r(\mathbf{q}(t), t) = \underset{\mathbf{y}_r \in \mathbb{R}^{k_f}}{\text{argmin}} \|\mathbf{P}^T \mathbf{V}_\mathbf{f} \mathbf{y}_r - \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)\|_2$$

  - note that $\mathbf{M} = \mathbf{P}^T \mathbf{V}_\mathbf{f} \in \mathbb{R}^{k_i \times k_\mathbf{f}}$ is a skinny matrix
  - its singular value decomposition can be written as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{Z}^T$$

  - then, the left inverse of $\mathbf{M}$ $\left((\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\right)$ is given by

$$\mathbf{M}^\dagger = \mathbf{Z}\mathbf{\Sigma}^\dagger\mathbf{U}^T$$

  where $\mathbf{\Sigma}^\dagger = \text{diag}(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0)$ if $\mathbf{\Sigma} = \text{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$, where $\sigma_1 \geq \cdots \sigma_r > 0$
  - and therefore

$$\begin{aligned} \hat{\mathbf{f}}(\mathbf{q}(t), t) &= \mathbf{V}_\mathbf{f} \left(\mathbf{P}^T \mathbf{V}_\mathbf{f}\right)^\dagger \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t) \\ &= \mathbf{V}_\mathbf{f} \left(\mathbf{Z}\mathbf{\Sigma}^\dagger\mathbf{U}^T\right) \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t) \end{aligned}$$

- The selection of the indices in $\mathcal{I}$ takes place after the matrix $\mathbf{V_f} = [\mathbf{v}_{\mathbf{f},1} \quad \cdots \quad \mathbf{v}_{\mathbf{f},k_\mathbf{f}}]$ has been computed using, for example, POD

- The selection of the indices in $\mathcal{I}$ takes place after the matrix $\mathbf{V_f} = [\mathbf{v}_{\mathbf{f},1} \quad \cdots \quad \mathbf{v}_{\mathbf{f},k_\mathbf{f}}]$ has been computed using, for example, POD
- Greedy algorithm

$[s, i_1] = \max\{|\mathbf{v}_{\mathbf{f},1}|\}$
$\mathbf{V_f} = [\mathbf{v}_{\mathbf{f},1}]$, $\mathbf{P} = [\mathbf{e}_{i_1}]$
for $l = 2 : k_\mathbf{f}$ **do**
    solve $\mathbf{P}^T\mathbf{V_f}\mathbf{c} = \mathbf{P}^T\mathbf{v}_{\mathbf{f},l}$ for $\mathbf{c}$
    $\mathbf{r} = \mathbf{v}_{\mathbf{f},l} - \mathbf{V_f}\mathbf{c}$
    $[s, i_l] = \max\{|\mathbf{r}|\}$
    $\mathbf{V}_f = [\mathbf{V_f}, \mathbf{v}_{\mathbf{f},l}]$, $\mathbf{P} = [\mathbf{P}, \quad \mathbf{e}_{i_l}]$
end for

Strengths

- The cost of the online phase does not scale with the size N of the HDM

Weaknesses

Strengths

- The cost of the online phase does not scale with the size N of the HDM

Weaknesses

- The online phase is software-intrusive

Strengths

- The cost of the online phase does not scale with the size N of the HDM
- The hyperreduced function is usually robust with respect to deviations from the original training trajectory

Weaknesses

- The online phase is software-intrusive

Strengths

- The cost of the online phase does not scale with the size N of the HDM
- The hyperreduced function is usually robust with respect to deviations from the original training trajectory

Weaknesses

- The online phase is software-intrusive
- Many parameters to adjust (ROB sizes, mask size, ...)

- Consider the inviscid Burgers equation

$$\frac{\partial U}{\partial t}(x,t) + \frac{1}{2}\frac{\partial U^2}{\partial x}(x,t) = g(x)$$

  - source term

$$g(x) = 0.02\exp(0.02x)$$

  - initial condition

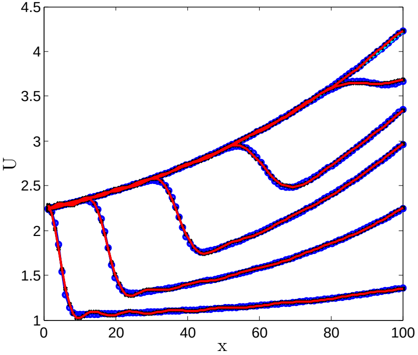
$$U(x,0) = 1$$

  - inlet boundary condition

$$U(0,t) = \sqrt{5}$$

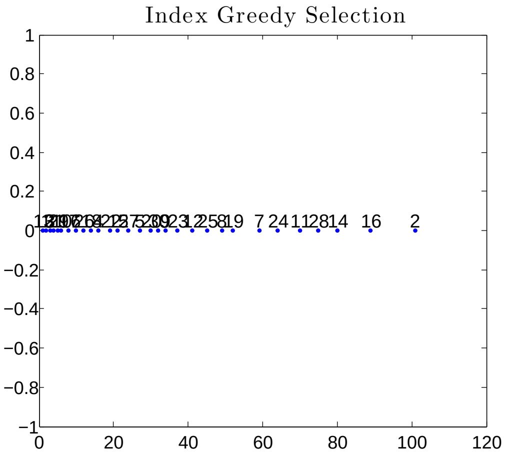
- Discretize it by a Finite Volume (Godunov) method

- $k = 15,\ k_{\mathbf{f}} = 40,\ k_i = 40$

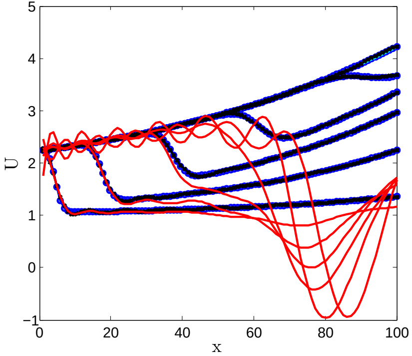


- Similar results for $k_i > 40$ (least-squares reconstruction)

- Results of the greedy algorithm



- The dimension $k_{\mathbf{f}}$ of the ROB $\mathbf{V_f}$ is reduced from 40 to 30
- $k = 15,\ k_{\mathbf{f}} = 30,\ k_i = 80$



- Similar results for $k_i = 100$ (no gaps) $\Rightarrow$ $k_{\mathbf{f}}$ is too small in that case

- Semi-discrete level: $\dfrac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t), t)$

- Semi-discrete level: $\dfrac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t), t)$
- Subspace approximation: $\mathbf{w}(t) \approx \mathbf{V}\mathbf{q}(t) \Rightarrow \mathbf{V}\dfrac{d\mathbf{q}}{dt}(t) \approx \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$

- Semi-discrete level: $\frac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t), t)$
- Subspace approximation: $\mathbf{w}(t) \approx \mathbf{V}\mathbf{q}(t) \Rightarrow \mathbf{V}\frac{d\mathbf{q}}{dt}(t) \approx \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$
- Discrete level (backward Euler implicit time-integration scheme)

$$\mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t^{n}} \approx \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Semi-discrete level: $\frac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t), t)$
- Subspace approximation: $\mathbf{w}(t) \approx \mathbf{V}\mathbf{q}(t) \Rightarrow \mathbf{V}\frac{d\mathbf{q}}{dt}(t) \approx \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$
- Discrete level (backward Euler implicit time-integration scheme)

$$\mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t^n} \approx \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Discrete residual

$$\mathbf{r}^{n+1}\left(\mathbf{q}^{n+1}\right) = \mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t^n} - \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Semi-discrete level: $\dfrac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t), t)$
- Subspace approximation: $\mathbf{w}(t) \approx \mathbf{V}\mathbf{q}(t) \Rightarrow \mathbf{V}\dfrac{d\mathbf{q}}{dt}(t) \approx \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$
- Discrete level (backward Euler implicit time-integration scheme)

$$\mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t^n} \approx \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Discrete residual

$$\mathbf{r}^{n+1}\left(\mathbf{q}^{n+1}\right) = \mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t^n} - \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Residual minimization (a.k.a PMOR by least-squares or Petrov-Galerkin projection)

$$\boxed{\mathbf{q}^{n+1} = \underset{\mathbf{y}\in\mathbb{R}^k}{\text{argmin}} \left\|\mathbf{r}^{n+1}(\mathbf{y})\right\|_2}$$

- Semi-discrete level: $\dfrac{d\mathbf{w}}{dt}(t) = \mathbf{f}(\mathbf{w}(t), t)$
- Subspace approximation: $\mathbf{w}(t) \approx \mathbf{V}\mathbf{q}(t) \Rightarrow \mathbf{V}\dfrac{d\mathbf{q}}{dt}(t) \approx \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$
- Discrete level (backward Euler implicit time-integration scheme)

$$\mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t^n} \approx \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Discrete residual

$$\mathbf{r}^{n+1}\left(\mathbf{q}^{n+1}\right) = \mathbf{V}\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t^n} - \mathbf{f}\left(\mathbf{V}\mathbf{q}^{n+1}, t^{n+1}\right)$$

- Residual minimization (a.k.a PMOR by least-squares or Petrov-Galerkin projection)

$$\boxed{\mathbf{q}^{n+1} = \underset{\mathbf{y}\in\mathbb{R}^k}{\operatorname{argmin}} \left\|\mathbf{r}^{n+1}(\mathbf{y})\right\|_2}$$

- $\mathbf{r}\left(\mathbf{q}^{n+1}\right)$ is nonlinear $\Rightarrow$ approximate it using a gappy POD approach for hyperreduction

- Gappy POD procedure for the fully discrete residual $\mathbf{r}$

- Gappy POD procedure for the fully discrete residual $\mathbf{r}$
- Algorithm

- Gappy POD procedure for the fully discrete residual $\mathbf{r}$
- Algorithm
  1. build an orthogonal ROB $\mathbf{V_r} \in \mathbb{R}^{N \times k_\mathbf{r}}$ for $\mathbf{r}$ $\left(\mathbf{V_r}^T \mathbf{V_r} = \mathbf{I}_{k_\mathbf{r}}\right)$

- Gappy POD procedure for the fully discrete residual $\mathbf{r}$
- Algorithm
  1. build an orthogonal ROB $\mathbf{V_r} \in \mathbb{R}^{N \times k_\mathbf{r}}$ for $\mathbf{r}$ $\left(\mathbf{V_r}^T \mathbf{V_r} = \mathbf{I}_{k_\mathbf{r}}\right)$
  2. construct a sample mesh $\mathcal{I}$ (indices $i_1, \cdots, i_{k_i}$) using the greedy procedure

- Gappy POD procedure for the fully discrete residual $\mathbf{r}$
- Algorithm
  1. build an orthogonal ROB $\mathbf{V_r} \in \mathbb{R}^{N \times k_\mathbf{r}}$ for $\mathbf{r}$ $\left(\mathbf{V_r}^T\mathbf{V_r} = \mathbf{I}_{k_\mathbf{r}}\right)$
  2. construct a sample mesh $\mathcal{I}$ (indices $i_1, \cdots, i_{k_i}$) using the greedy procedure
  3. consider the gappy approximation

$$\mathbf{r}^{n+1}\left(\mathbf{q}^{n+1}\right) \approx \mathbf{V_r}\mathbf{r}_{k_\mathbf{r}}\left(\mathbf{q}^{n+1}\right) \approx \mathbf{V_r}\left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger}\mathbf{P}^T\mathbf{r}^{n+1}\left(\mathbf{V}\mathbf{q}^{n+1}\right)$$

- Gappy POD procedure for the fully discrete residual $\mathbf{r}$
- Algorithm
  1. build an orthogonal ROB $\mathbf{V_r} \in \mathbb{R}^{N \times k_\mathbf{r}}$ for $\mathbf{r}$ $\left(\mathbf{V_r}^T\mathbf{V_r} = \mathbf{I}_{k_\mathbf{r}}\right)$
  2. construct a sample mesh $\mathcal{I}$ (indices $i_1, \cdots, i_{k_i}$) using the greedy procedure
  3. consider the gappy approximation

$$\mathbf{r}^{n+1}\left(\mathbf{q}^{n+1}\right) \approx \mathbf{V_r}\mathbf{r}_{k_\mathbf{r}}\left(\mathbf{q}^{n+1}\right) \approx \mathbf{V_r}\left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger}\mathbf{P}^T\mathbf{r}^{n+1}\left(\mathbf{V}\mathbf{q}^{n+1}\right)$$

  4. determine the vector of generalized coordinates at t^{n+1}

$$\begin{aligned}\mathbf{q}^{n+1} &= \underset{\mathbf{y}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{V_r}\mathbf{r}_{k_\mathbf{r}}(\mathbf{y})\|_2 \\ &= \underset{\mathbf{y}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{r}_{k_\mathbf{r}}(\mathbf{y})\|_2 \\ &= \underset{\mathbf{y}\in\mathbb{R}^k}{\operatorname{argmin}} \left\|\left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger}\mathbf{P}^T\mathbf{r}^{n+1}(\mathbf{V}\mathbf{y})\right\|_2\end{aligned}$$

- Nonlinear least-squares problem: $\min_{\mathbf{y}} \|\mathbf{r}(\mathbf{y})\|_2$, where $\mathbf{r} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^k$, and $k \ll N$

- Nonlinear least-squares problem: $\min_{\mathbf{y}} \|\mathbf{r}(\mathbf{y})\|_2$, where $\mathbf{r} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^k$, and $k \ll N$
- Equivalent function to minimize: $\phi(\mathbf{y}) = \frac{1}{2}\|\mathbf{r}(\mathbf{y})\|_2^2 = \mathbf{r}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$

- Nonlinear least-squares problem: $\min_{\mathbf{y}} \|\mathbf{r}(\mathbf{y})\|_2$, where $\mathbf{r} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^k$, and $k \ll N$
- Equivalent function to minimize: $\phi(\mathbf{y}) = \dfrac{1}{2}\|\mathbf{r}(\mathbf{y})\|_2^2 = \mathbf{r}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$
- Gradient: $\nabla\phi(\mathbf{y}) = \mathbf{J}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$, where $\mathbf{J}(\mathbf{y}) = \dfrac{\partial \mathbf{r}}{\partial \mathbf{y}}(\mathbf{y})$

- Nonlinear least-squares problem: $\min_{\mathbf{y}} \|\mathbf{r}(\mathbf{y})\|_2$, where $\mathbf{r} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^k$, and $k \ll N$
- Equivalent function to minimize: $\phi(\mathbf{y}) = \dfrac{1}{2}\|\mathbf{r}(\mathbf{y})\|_2^2 = \mathbf{r}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$
- Gradient: $\nabla\phi(\mathbf{y}) = \mathbf{J}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$, where $\mathbf{J}(\mathbf{y}) = \dfrac{\partial \mathbf{r}}{\partial \mathbf{y}}(\mathbf{y})$
- Iterative solution of equivalent minimization problem using the Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta\mathbf{y}^{(j+1)}$$

where

$$\nabla^2\phi\left(\mathbf{y}^{(j)}\right)\Delta\mathbf{y}^{(j+1)} = -\nabla\phi\left(\mathbf{y}^{(j)}\right)$$

- Nonlinear least-squares problem: $\min_{\mathbf{y}} \|\mathbf{r}(\mathbf{y})\|_2$, where $\mathbf{r} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^k$, and $k \ll N$
- Equivalent function to minimize: $\phi(\mathbf{y}) = \dfrac{1}{2}\|\mathbf{r}(\mathbf{y})\|_2^2 = \mathbf{r}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$
- Gradient: $\nabla\phi(\mathbf{y}) = \mathbf{J}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$, where $\mathbf{J}(\mathbf{y}) = \dfrac{\partial \mathbf{r}}{\partial \mathbf{y}}(\mathbf{y})$
- Iterative solution of equivalent minimization problem using the Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta\mathbf{y}^{(j+1)}$$

where

$$\nabla^2\phi\left(\mathbf{y}^{(j)}\right)\Delta\mathbf{y}^{(j+1)} = -\nabla\phi\left(\mathbf{y}^{(j)}\right)$$

- What is $\nabla^2\phi(\mathbf{y})$?

$$\nabla^2\phi(\mathbf{y}) = \mathbf{J}(\mathbf{y})^T\mathbf{J}(\mathbf{y}) + \sum_{i=1}^{N} \frac{\partial^2 r_i}{\partial \mathbf{y}^2}(\mathbf{y})\, r_i(\mathbf{y})$$

- Nonlinear least-squares problem: $\min_{\mathbf{y}} \|\mathbf{r}(\mathbf{y})\|_2$, where $\mathbf{r} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^k$, and $k \ll N$
- Equivalent function to minimize: $\phi(\mathbf{y}) = \dfrac{1}{2}\|\mathbf{r}(\mathbf{y})\|_2^2 = \mathbf{r}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$
- Gradient: $\nabla\phi(\mathbf{y}) = \mathbf{J}(\mathbf{y})^T\mathbf{r}(\mathbf{y})$, where $\mathbf{J}(\mathbf{y}) = \dfrac{\partial \mathbf{r}}{\partial \mathbf{y}}(\mathbf{y})$
- Iterative solution of equivalent minimization problem using the Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta\mathbf{y}^{(j+1)}$$

  where

$$\nabla^2\phi\left(\mathbf{y}^{(j)}\right)\Delta\mathbf{y}^{(j+1)} = -\nabla\phi\left(\mathbf{y}^{(j)}\right)$$

- What is $\nabla^2\phi(\mathbf{y})$?

$$\nabla^2\phi(\mathbf{y}) = \mathbf{J}(\mathbf{y})^T\mathbf{J}(\mathbf{y}) + \sum_{i=1}^{N}\frac{\partial^2 r_i}{\partial \mathbf{y}^2}(\mathbf{y})r_i(\mathbf{y})$$

- Gauss-Newton method with $\nabla^2\phi(\mathbf{y}) \approx \mathbf{J}(\mathbf{y})^T\mathbf{J}(\mathbf{y})$

- Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta\mathbf{y}^{(j+1)}$$

where

$$\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{J}\left(\mathbf{y}^{(j)}\right) \Delta\mathbf{y}^{(j+1)} = -\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{r}\left(\mathbf{y}^{(j)}\right)$$

- Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta\mathbf{y}^{(j+1)}$$

where

$$\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{J}\left(\mathbf{y}^{(j)}\right) \Delta\mathbf{y}^{(j+1)} = -\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{r}\left(\mathbf{y}^{(j)}\right)$$

- This is the normal equation for

$$\Delta\mathbf{y}^{(j+1)} = \underset{\mathbf{z}}{\text{argmin}} \left\| \mathbf{J}\left(\mathbf{y}^{(j)}\right) \mathbf{z} + \mathbf{r}\left(\mathbf{y}^{(j)}\right) \right\|_2$$

- Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta\mathbf{y}^{(j+1)}$$

where

$$\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{J}\left(\mathbf{y}^{(j)}\right) \Delta\mathbf{y}^{(j+1)} = -\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{r}\left(\mathbf{y}^{(j)}\right)$$

- This is the normal equation for

$$\Delta\mathbf{y}^{(j+1)} = \underset{\mathbf{z}}{\text{argmin}} \left\| \mathbf{J}\left(\mathbf{y}^{(j)}\right)\mathbf{z} + \mathbf{r}\left(\mathbf{y}^{(j)}\right) \right\|_2$$

- QR decomposition of the Jacobian

$$\mathbf{J}\left(\mathbf{y}^{(j)}\right) = \mathbf{Q}^{(j)}\mathbf{R}^{(j)}$$

- Gauss-Newton method

$$\mathbf{y}^{(j+1)} = \mathbf{y}^{(j)} + \Delta \mathbf{y}^{(j+1)}$$

where

$$\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{J}\left(\mathbf{y}^{(j)}\right) \Delta \mathbf{y}^{(j+1)} = -\mathbf{J}\left(\mathbf{y}^{(j)}\right)^T \mathbf{r}\left(\mathbf{y}^{(j)}\right)$$

- This is the normal equation for

$$\Delta \mathbf{y}^{(j+1)} = \underset{\mathbf{z}}{\operatorname{argmin}} \left\| \mathbf{J}\left(\mathbf{y}^{(j)}\right) \mathbf{z} + \mathbf{r}\left(\mathbf{y}^{(j)}\right) \right\|_2$$

- QR decomposition of the Jacobian

$$\mathbf{J}\left(\mathbf{y}^{(j)}\right) = \mathbf{Q}^{(j)} \mathbf{R}^{(j)}$$

- Equivalent solution using the QR decomposition (assuming that $\mathbf{R}^{(j)}$ has full column rank)

$$\Delta \mathbf{y}^{(j+1)} = -\mathbf{J}\left(\mathbf{y}^{(j)}\right)^{\dagger} \mathbf{r}\left(\mathbf{y}^{(j)}\right) = -\left(\mathbf{R}^{(j)}\right)^{-1} \left(\mathbf{Q}^{(j)}\right)^T \mathbf{r}\left(\mathbf{y}^{(j)}\right)$$

- GNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD

- GNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD
- Minimization problem

$$\min_{\mathbf{y}\in\mathbb{R}^k} \left\| \left(\mathbf{P}^T\mathbf{V}_\mathbf{r}\right)^\dagger \mathbf{P}^T\mathbf{r}^{n+1}(\mathbf{V}\mathbf{y}) \right\|_2$$

- GNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD
- Minimization problem

$$\min_{\mathbf{y}\in\mathbb{R}^k} \left\| \left(\mathbf{P}^T\mathbf{V_r}\right)^\dagger \mathbf{P}^T\mathbf{r}^{n+1}(\mathbf{V}\mathbf{y}) \right\|_2$$

- Jacobian: $\widehat{\mathbf{J}}(\mathbf{y}) = \left(\mathbf{P}^T\mathbf{V_r}\right)^\dagger \mathbf{P}^T\mathbf{J}^{n+1}(\mathbf{V}\mathbf{y})$

- GNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD
- Minimization problem

$$\min_{\mathbf{y}\in\mathbb{R}^k} \left\| \left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger} \mathbf{P}^T\mathbf{r}^{n+1}(\mathbf{V}\mathbf{y}) \right\|_2$$

- Jacobian: $\widehat{\mathbf{J}}(\mathbf{y}) = \left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger} \mathbf{P}^T\mathbf{J}^{n+1}(\mathbf{V}\mathbf{y})$
- Construct offline the small-dimensional matrix $\mathbf{A} = \left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger}$

- GNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD
- Minimization problem

$$\min_{\mathbf{y}\in\mathbb{R}^k} \left\| \left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger} \mathbf{P}^T\mathbf{r}^{n+1}(\mathbf{V}\mathbf{y}) \right\|_2$$

- Jacobian: $\widehat{\mathbf{J}}(\mathbf{y}) = \left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger} \mathbf{P}^T\mathbf{J}^{n+1}(\mathbf{V}\mathbf{y})$
- Construct offline the small-dimensional matrix $\mathbf{A} = \left(\mathbf{P}^T\mathbf{V_r}\right)^{\dagger}$
- Solve at each j-th Gauss-Newton iteration of t^{n+1} the least-squares problem

$$\Delta\mathbf{y}^{(j)} = \underset{\mathbf{z}\in\mathbb{R}^k}{\text{argmin}} \left\| \mathbf{A}\mathbf{P}^T\mathbf{J}^{n+1}\left(\mathbf{V}\mathbf{y}^{(j)}\right)\mathbf{V}\mathbf{z} + \mathbf{A}\mathbf{P}^T\mathbf{r}^{n+1}\left(\mathbf{V}\mathbf{y}^{(j)}\right) \right\|_2$$

- GNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD
- Minimization problem

$$\min_{\mathbf{y}\in\mathbb{R}^k} \left\| \left(\mathbf{P}^T\mathbf{V_r}\right)^\dagger \mathbf{P}^T\mathbf{r}^{n+1}(\mathbf{V}\mathbf{y})\right\|_2$$

- Jacobian: $\widehat{\mathbf{J}}(\mathbf{y}) = \left(\mathbf{P}^T\mathbf{V_r}\right)^\dagger \mathbf{P}^T\mathbf{J}^{n+1}(\mathbf{V}\mathbf{y})$
- Construct offline the small-dimensional matrix $\mathbf{A} = \left(\mathbf{P}^T\mathbf{V_r}\right)^\dagger$
- Solve at each j-th Gauss-Newton iteration of t^{n+1} the least-squares problem

$$\Delta\mathbf{y}^{(j)} = \underset{\mathbf{z}\in\mathbb{R}^k}{\operatorname{argmin}} \left\| \mathbf{A}\mathbf{P}^T\mathbf{J}^{n+1}\left(\mathbf{V}\mathbf{y}^{(j)}\right)\mathbf{V}\mathbf{z} + \mathbf{A}\mathbf{P}^T\mathbf{r}^{n+1}\left(\mathbf{V}\mathbf{y}^{(j)}\right)\right\|_2$$

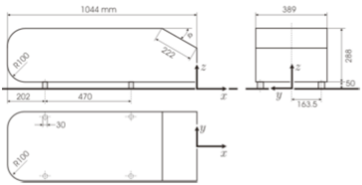
- Compute the GNAT solution using QR

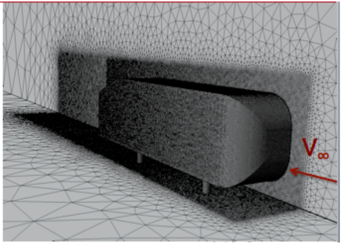
$$\mathbf{A}\mathbf{P}^T\mathbf{J}^{n+1}\left(\mathbf{V}\mathbf{y}^{(j)}\right)\mathbf{V} = \mathbf{Q}^{(j)}\mathbf{R}^{(j)}$$

$$\Delta\mathbf{y}^{(j)} = -\left(\mathbf{R}^{(j)}\right)^{-1}\left(\mathbf{Q}^{(j)}\right)^T\mathbf{A}\mathbf{P}^T\mathbf{r}^{n+1}\left(\mathbf{V}\mathbf{y}^{(j)}\right)$$

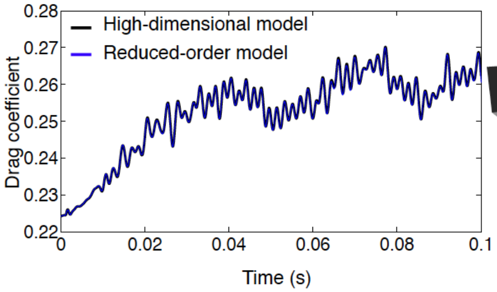
- Further developments (see the references by: Carlberg, Farhat, Cortial, Amsallem; Carlberg, Bou-Mosleh, Farhat; and Amsallem, Zahr, Farhat)
  - concept of a reduced mesh
  - concept of an output mesh
  - error bounds
  - GNAT using local reduced-order bases

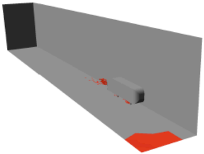
- Turbulent flow past the Ahmed body (CFD benchmark in the automotive industry)
- 3D compressible Navier-Stokes equations with turbulence modeling (Spalart-Allmaras)
- $N = 1.73 \times 10^7$
- $Re = 4.48 \times 10^6$, $M_\infty = 0.175$ (216km/h)





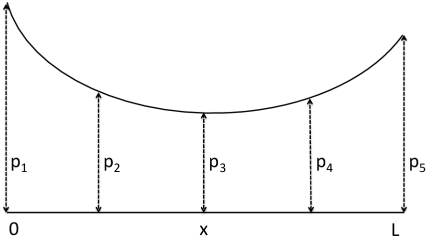
- PMOR: POD + GNAT, $k = 283$, $k_{\mathbf{f}} = 1,514$, and $k_i = 2,268$





Method	CPU Time	Number of CPUs	Relative Error
HDM	13.28 h	512	–
PROM (GNAT)	3.88 h	4	0.68%

- HDM: $N = 2,048$ and $m = 5$ shape parameters
- PMOR: POD + DEIM: $k = 8$, $k_{\mathbf{f}} = 20$, and $k_i = 20$
- Parameterized steady-state problem

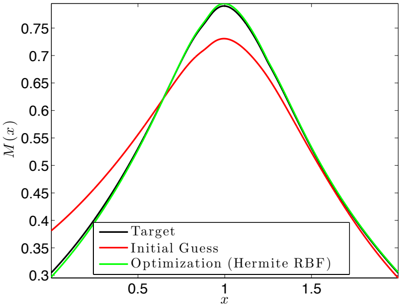


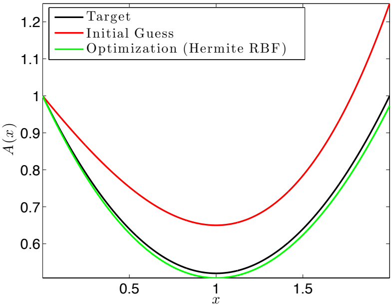
$$\min_{\boldsymbol{\mu} \in \mathbb{R}^5} \|M(\mathbf{w}(\boldsymbol{\mu})) - M_{\text{target}}\|_2$$

$$\text{s.t.} \quad \mathbf{f}(\mathbf{w}(\boldsymbol{\mu})), \boldsymbol{\mu}) = \mathbf{0}$$

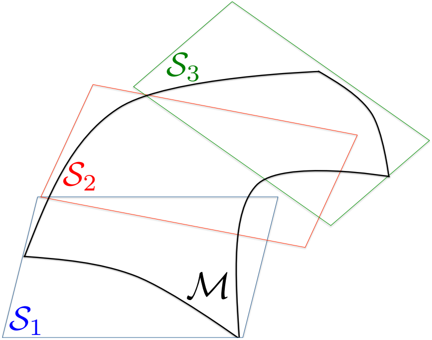
where M denotes the local Mach number

Method	Offline CPU Time	Online CPU Time	Total CPU Time
HDM	–	78.8 s	78.8 s
PROM (DEIM)	5.08 s	4.87 s	9.96 s





- Approximating the solution manifold $\mathcal{M}$ by a single subspace $\mathcal{S}$ can lead to a large-dimensional subspace
- Idea: Approximate $\mathcal{M}$ using **local** subspaces $\{\mathcal{S}_l\}_{l=1}^{L}$



- In practice, the local approximation of the state takes place at the **fully discrete level**
- Each local subspace $\mathcal{S}_l$ is associated with a pre-computed local ROB $\mathbf{V}_l$
- At each time-step n, the state $\mathbf{w}^n$ is computed as

$$\mathbf{w}^n = \mathbf{w}^{n-1} + \Delta\mathbf{w}^n$$

- The increment $\Delta\mathbf{w}^n$ is then approximated in a subspace $\mathcal{S}_{l,n} = \text{range}(\mathbf{V}_{l,n})$ as

$$\Delta\mathbf{w}^n \approx \mathbf{V}_{l,n}\widetilde{\mathbf{q}}^n$$

- The choice of the pre-computed reduced-order basis $\mathbf{V}_{l,n}$ is specified later
- By induction, the state $\mathbf{w}^n$ is computed as

$$\mathbf{w}^n = \mathbf{w}^0 + \sum_{i=1}^{n} \mathbf{V}_{l,i}\widetilde{\mathbf{q}}^n$$

- The state $\mathbf{w}^n$ is computed as

$$\mathbf{w}^n = \mathbf{w}^0 + \sum_{i=1}^{n} \mathbf{V}_{l,i} \widetilde{\mathbf{q}}^n$$

- In practice, the ROBs $\{\mathbf{V}_{l,i}\}_{i=1}^{n}$ are chosen among a finite set of pre-computed local ROBs $\{\mathbf{V}_l\}_{l=1}^{L}$
- Hence

$$\mathbf{w}^n = \mathbf{w}^0 + \sum_{l=1}^{L} \mathbf{V}_l \mathbf{q}_l^n$$

- This shows that

$$\mathbf{w}^n \in \mathbf{w}^0 + \text{range}([\mathbf{V}_1 \quad \cdots \quad \mathbf{V}_L])$$

- Note that each local ROB can be of a different dimension

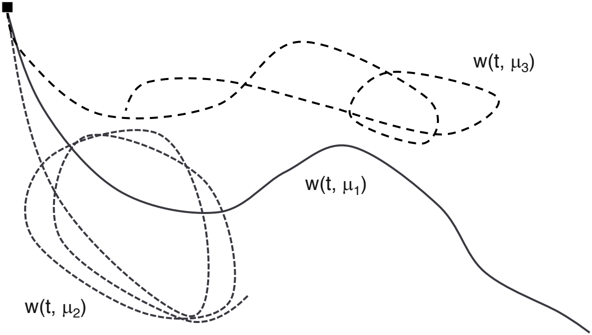
$$\mathbf{V}_l \in \mathbb{R}^{N \times k_l}$$

- Intuitively, a given local subspace $\mathcal{S}_l$ should approximate only a portion of the solution manifold $\mathcal{M}$
- The solution manifold is a subset of the solution space $\mathbb{R}^N$

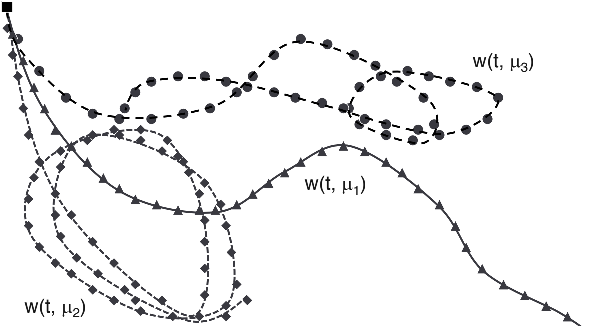
$$\mathcal{M} \subset \mathbb{R}^N$$

- The solution space $\mathbb{R}^N$ is partitioned into L subdomains, where each subdomain is associated with a local approximation subspace $\mathcal{S}_l = \text{range}(\mathbf{V}_l)$
- In practice, a set of solution snapshots $\{\mathbf{w}_i\}_{i=1}^{N_s}$ can be partitioned into L subsets using the k-means clustering algorithm
- This leads to a Voronoi tessellation of $\mathbb{R}^N$
- The k-means clustering algorithm is distance-dependent
- After clustering, each snapshot subset can be compressed into a local ROB, for example, using POD

- Local ROBs construction procedure

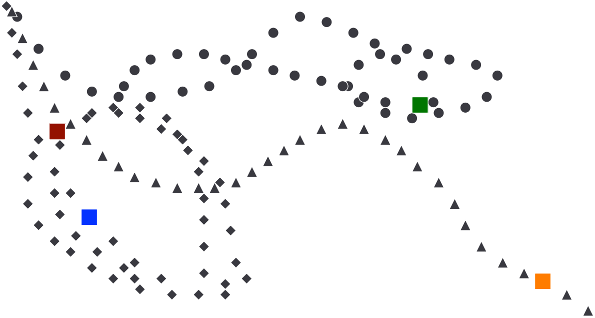


- Local ROBs construction procedure

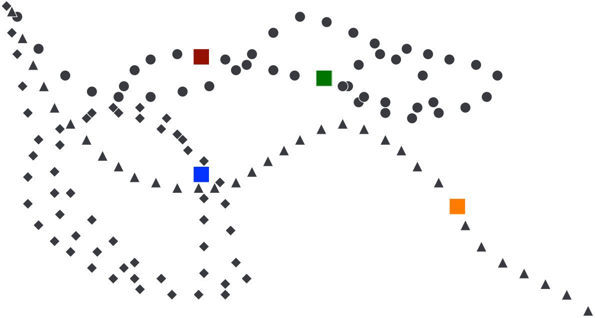


- Local ROBs construction procedure

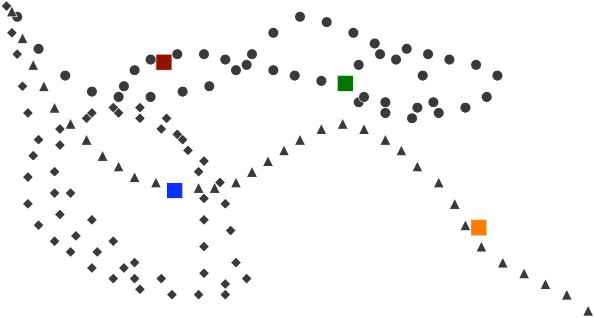
- Local ROBs construction procedure



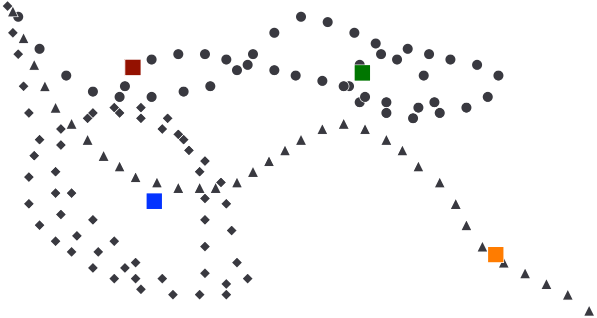
- Local ROBs construction procedure



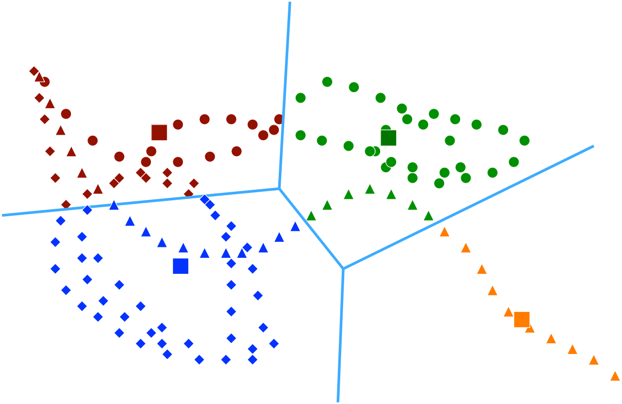
- Local ROBs construction procedure



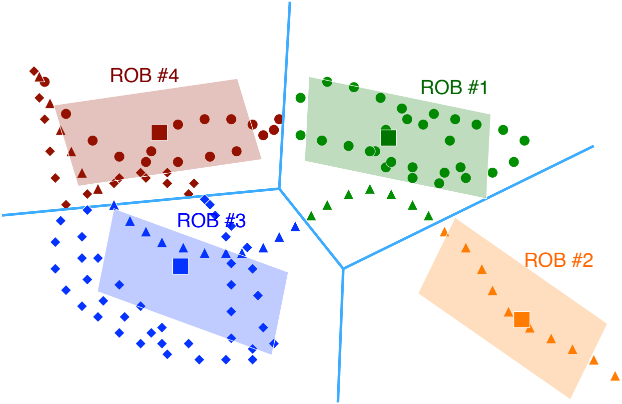
- Local ROBs construction procedure



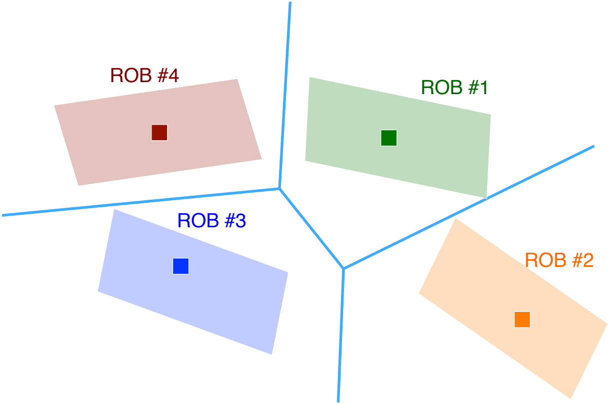
- Local ROBs construction procedure



- Local ROBs construction procedure



- Local ROBs construction procedure



- Online, at time-step n, a pre-computed local ROB $\mathbf{V}_{l,n}$ must be chosen
- The selection is based on the current location of $\mathbf{w}^{n-1}$ on the solution manifold $\mathcal{M}$
- The local approximation subspace is selected as that associated with the cluster whose center is the closest to $\mathbf{w}^{n-1}$

$$l, n = \underset{l \in \{1, \cdots, L\}}{\operatorname{argmin}} \; d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^c\right)$$

- Consider the case of the distance based on a weighted Euclidian norm

$$d(\mathbf{w}, \mathbf{z}) = \|\mathbf{w} - \mathbf{z}\|_{\mathbf{H}} = \sqrt{(\mathbf{w} - \mathbf{z})^T \mathbf{H} (\mathbf{w} - \mathbf{z})}$$

where $\mathbf{H} \in \mathbb{R}^{N \times N}$ is a symmetric positive definite matrix

- Choice of the local approximation subspace at time-step n

$$l, n = \underset{l \in \{1, \cdots, L\}}{\text{argmin}} \; d(\mathbf{w}^{n-1}, \mathbf{w}_l^c)$$

- For a distance based on a weighted Euclidian norm, the solution of the above problem can be computed efficiently at a cost that does not depend on the large dimension N
- To show this, consider the special form of the solution

$$\mathbf{w}^{n-1} = \mathbf{w}^0 + \sum_{l=1}^{L} \mathbf{V}_l \mathbf{q}_l^{n-1}$$

- Then, one needs to compare the distances $d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^{c_i}\right)$ and $d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^{c_j}\right)$ for $1 \leq i \neq j \leq L$

- The two distances $d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^{c_i}\right)$ and $d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^{c_j}\right)$ can be compared as follows

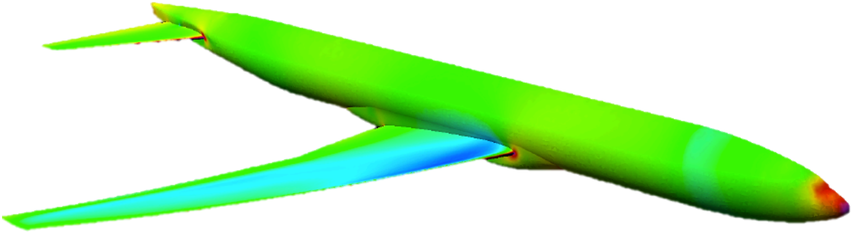
$$\begin{aligned}
\Delta_{i,j} &= d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^{c_i}\right)^2 - d\left(\mathbf{w}^{n-1}, \mathbf{w}_l^{c_j}\right)^2 \\
&= \left\|\mathbf{w}^{n-1} - \mathbf{w}_l^{c_i}\right\|_{\mathbf{H}}^2 - \left\|\mathbf{w}^{n-1} - \mathbf{w}_l^{c_j}\right\|_{\mathbf{H}}^2 \\
&= \left\|\sum_{l=1}^{L} \mathbf{V}_l \mathbf{q}_l^{n-1}\right\|_{\mathbf{H}}^2 + \left\|\mathbf{w}_l^{c_i} - \mathbf{w}^0\right\|_{\mathbf{H}}^2 - 2\sum_{l=1}^{L} \left[\mathbf{w}^{c_i}\right]^T \mathbf{V}_l \mathbf{q}_l^{n-1} \\
&\quad - \left\|\sum_{l=1}^{L} \mathbf{V}_l \mathbf{q}_l^{n-1}\right\|_{\mathbf{H}}^2 - \left\|\mathbf{w}_l^{c_j} - \mathbf{w}^0\right\|_{\mathbf{H}}^2 + 2\sum_{l=1}^{L} \left[\mathbf{w}^{c_j}\right]^T \mathbf{V}_l \mathbf{q}_l^{n-1} \\
&= \left\|\mathbf{w}_l^{c_i} - \mathbf{w}^0\right\|_{\mathbf{H}}^2 - \left\|\mathbf{w}_l^{c_j} - \mathbf{w}^0\right\|_{\mathbf{H}}^2 + 2\sum_{l=1}^{L} \left[\mathbf{w}^{c_i} - \mathbf{w}^{c_j}\right]^T \mathbf{V}_l \mathbf{q}_l^{n-1}
\end{aligned}$$

- The following small quantities can be pre-computed offline and used online to compute economically $\Delta_{i,j}$, $1 \leq i \neq j \leq L$

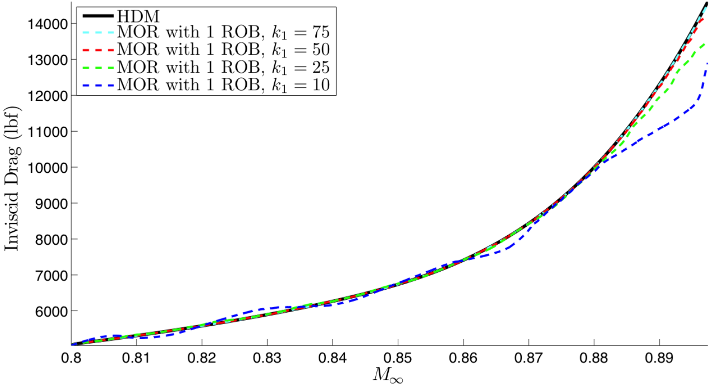
$$a_{i,j} = \left\|\mathbf{w}_l^{c_i} - \mathbf{w}^0\right\|_{\mathbf{H}}^2 - \left\|\mathbf{w}_l^{c_j} - \mathbf{w}^0\right\|_{\mathbf{H}}^2 \in \mathbb{R}, \; \mathbf{g}_{i,j} = \left[\mathbf{w}^{c_i} - \mathbf{w}^{c_j}\right]^T \mathbf{V}_l \in \mathbb{R}^{k_l}$$

- The local approach to nonlinear PMOR can be easily extended to hyperreduction as follows
  - hyperreduction is applied independently to each cluster of snapshots
  - It leads to the definition of
    - the local ROBs for the state: $\mathbf{V}_l, \ l = 1, \cdots, L$
    - the local ROBs for the residual: $\mathbf{V}_{\mathbf{r},l}, \ l = 1, \cdots, L$
    - the local masks: $\mathcal{I}_l, \ l = 1, \cdots, L$
- The choice of the local ROBs and masks is still dictated by the location of the current time-iterate in the state space

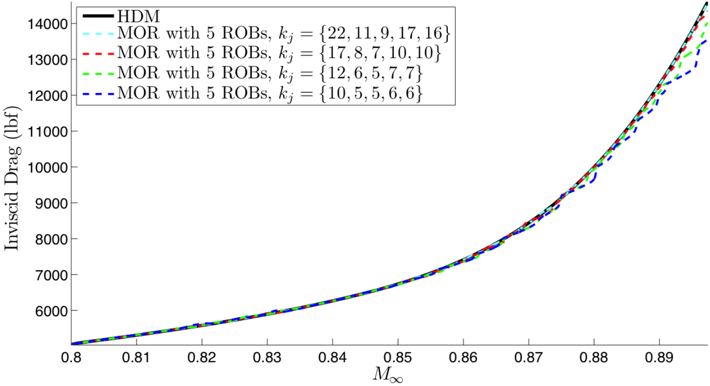
- Flow past the NASA CRM (Common Research Model) – (CFD benchmark in the aeronautical industry)
- 3D compressible Euler equations
- $N = 3.1 \times 10^6$
- Constant acceleration of 2.5 m/s^2, from $M_\infty = 0.8$ to $M_\infty = 0.9$



- PMOR using a global ROB



- PMOR using 5 local ROBs



- Very good accuracy can be obtained with $k_l \leq 17$ as opposed to $k = 50$ with a global ROB

- M. Rewienski, J. White. Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. Linear Algebra and its Applications 2006; 415(2-3):426-454.
- R. Everson, L. Sirovich. Karhunen–Loeve procedure for gappy data. Journal of the Optical Society of America A 1995; 12(8):1657–1664.
- M. Barrault et al. An empirical interpolation method: application to efficient reduced basis discretization of partial differential equations. Comptes Rendus de lAcademie des Sciences Paris 2004; 339:667–672.
- K. Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. Computers and Fluids 2006; 35:208–226.
- S. Chaturantabut, D.C. Sorensen. Nonlinear model reduction via Discrete Empirical Interpolation. SIAM Journal on Scientific Computing 2010; 32:2737–2764.

- K. Carlberg, C. Farhat, J. Cortial, D. Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. Journal of Computational Physics 2013; 242:623–647.
- K. Carlberg, C. Bou-Mosleh, C. Farhat. Efficient nonlinear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. International Journal for Numerical Methods in Engineering 2011; 86(2):155–181.
- D. Amsallem, M. Zahr, Y. Choi, C. Farhat. Design optimization using hyperreduced-order models. Structural and Multidisciplinary Optimization 2015; 51:919–940.
- D. Amsallem, M. Zahr, C. Farhat. Nonlinear model reduction based on local reduced order bases. International Journal for Numerical Methods in Engineering 2012; 92(12):891-916.
- K. Washabaugh, M. Zahr, C. Farhat, On the use of discrete nonlinear reduced-order models for the prediction of steady-state flows past parametrically deformed complex geometries. AIAA-2016-1814, AIAA SciTech 2016, San Diego, CA, January 4-8, 2016.