AA216/CME345: PROJECTION-BASED MODEL ORDER REDUCTION

Hyperreduction of Projection-Based Reduced-Order Models

Charbel Farhat
Stanford University
cfarhat@stanford.edu

Outline

- Note: The material covered in this chapter is based on the following published documents:
  - 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 l'Academie 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.

- Note: The material covered in this chapter is based on the following published documents (continue):

  - 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.
  - C. Farhat, P. Avery, T. Chapman, Julien Cortial. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. International Journal for Numerical Methods in Engineering 2014; 98(9):625-662.

- Note: The material covered in this chapter is based on the following published documents (continue):
  - C. Farhat, T. Chapman and P. Avery. Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. International Journal for Numerical Methods in Engineering 2015; 102(5):1077-1110.
  - T. Chapman, P. Avery, P. Collins and C. Farhat. Accelerated mesh sampling for the hyper reduction of nonlinear computational models. International Journal for Numerical Methods in Engineering 2017; 109(12):1623-1654.
  - S. Grimberg, C. Farhat, R. Tezaur and C. Bou-Mosleh. Mesh sampling and weighting for the hyperreduction of nonlinear Petrov-Galerkin reduced-order models with local reduced-order bases. International Journal for Numerical Methods in Engineering 2021; 122(7):1846-1874.

- Note: The material covered in this chapter is based on the following published documents (continue):
  - R. Tezaur, F. As'ad and C. Farhat. Robust and globally efficient reduction of parametric, highly nonlinear computational models and real time online performance. Computer Methods in Applied Mechanics and Engineering 2022; 399(1):115392.

- HDM of interest

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

  - $\mathbf{w} \in \mathbb{R}^N$: Vector of state variables
  - $\mathbf{u} \in \mathbb{R}^{\text{in}}$: Vector of input variables – typically, in $\ll N$
  - $\mathbf{y} \in \mathbb{R}^q$: Vector of output variables – typically, $q \ll N$
  - $\boldsymbol{\mu} \in \mathbb{R}^p$: Vector of parameter variables – typically, $p \ll N$
  - f: **Nonlinear** function

- Usually, there is no closed form solution for $\mathbf{w}(t;\boldsymbol{\mu})$

- Approximation of the state using a right reduced-order basis (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

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

- Petrov-Galerkin PROM

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

- k equations with k unknowns

- Petrov-Galerkin PROM

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

- k equations with k unknowns
- For a given reduced state vector $\mathbf{q}(t;\boldsymbol{\mu})$ and a given vector-valued parameter $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^p$, 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 can be performed in 3 steps as follows
  1. compute $\tilde{\mathbf{w}}(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{q}(t;\boldsymbol{\mu})$
  2. evaluate $\mathbf{f}(\tilde{\mathbf{w}}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$
  3. left-multiply by $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T \Rightarrow (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\tilde{\mathbf{w}}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$

- Petrov-Galerkin PROM

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

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

- Petrov-Galerkin PROM

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

- k equations with k unknowns
- For a given reduced state vector $\mathbf{q}(t;\boldsymbol{\mu})$ and a given vector-valued parameter $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^p$, 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 can be performed in 3 steps as follows
  1. compute $\tilde{\mathbf{w}}(t;\boldsymbol{\mu}) = \mathbf{V}\mathbf{q}(t;\boldsymbol{\mu})$
  2. evaluate $\mathbf{f}(\tilde{\mathbf{w}}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$
  3. left-multiply by $(\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T \Rightarrow (\mathbf{W}^T\mathbf{V})^{-1}\mathbf{W}^T\mathbf{f}(\tilde{\mathbf{w}}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$
- The computational cost associated with the three steps described above **scales linearly with the dimension N of the HDM**
- Hence, for nonlinear problems – and for parametric linear 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)
- There are two main classes of nonlinear function approximations
  - linearization approaches (TPWL, ManiMOR,...)
  - hyperreduction approaches (DEIM, GNAT, ECSW, ...)

- 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 projection-based model order reduction (PMOR)

- Face recognition

Procedure

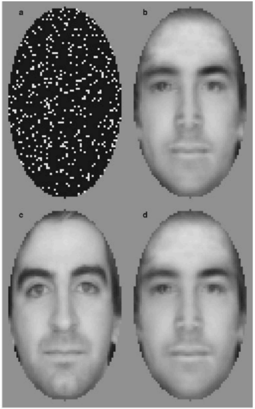


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions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

- Face recognition

Procedure

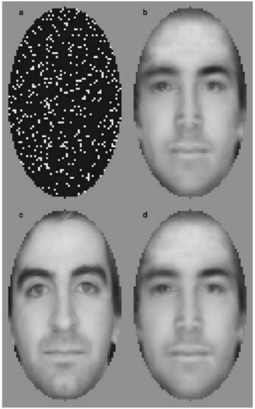


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions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

- Face recognition

Procedure

1. build a database of N_{snap} faces (snapshots)

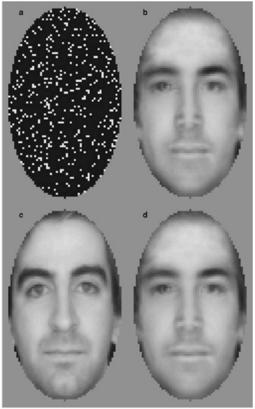


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions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

- Face recognition

Procedure

1. build a database of N_{snap} faces (snapshots)
2. construct a proper orthogonal decomposition (POD) basis $\mathbf{V}_\mathbf{f}$ for the database

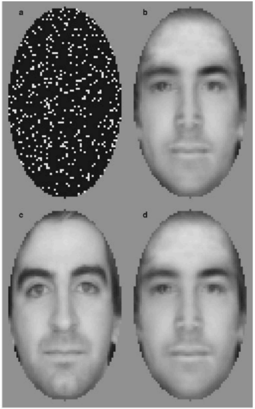


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions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

- Face recognition

Procedure

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

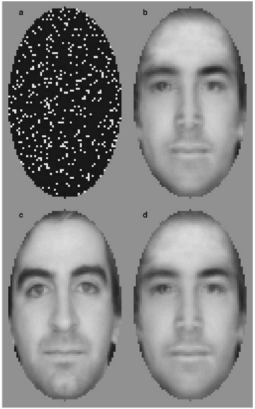


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions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

- Face recognition

Procedure

1. build a database of N_{snap} faces (snapshots)
2. construct a proper orthogonal decomposition (POD) basis $\mathbf{V}_\mathbf{f}$ 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 $\mathbf{V}_\mathbf{f}$, approximately reconstruct the new face $\mathbf{f}$ (in the least-squares sense)

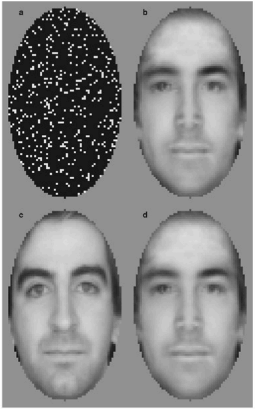


Fig. 1. Reconstruction of a face, not in the original ensemble, from a 10% mask. The reconstructed face, b, was determined with 50 empirical eigenfunctions and only the white pixels shown in a. The original face is shown in c, and a projection (with all the pixels) of the face onto 50 empirical eigenfunctions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

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

$$\frac{d\mathbf{q}}{dt}(t) = \left(\mathbf{W}^T\mathbf{V}\right)^{-1}\mathbf{W}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

  where the evaluation of all entries of $\mathbf{f}(\cdot, t)$ scales with N and thus can be computationally intensive (for simplicity, the input function $\mathbf{u}(t)$ is not considered here and the dependence on a parameter $\boldsymbol{\mu}$ is not emphasized)

- Gappy approach
  - evaluate only a small subset of the entries of $\mathbf{f}(\cdot, t)$
  - pre-compute a ROB $\mathbf{V_f}$ and use it to approximately reconstruct all other entries of $\mathbf{f}(\cdot, t)$ by interpolation or a least-squares strategy
  - $\Rightarrow$ approximate-then-project approach

- 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 $\Rightarrow \tilde{\mathbf{f}}(\cdot, t)$

- 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

$$\mathbf{F} = \mathbf{U_f} \boldsymbol{\Sigma}_\mathbf{f} \mathbf{Z}_\mathbf{f}^T$$

- 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 $\tilde{\mathbf{w}}(t) = \mathbf{Vq}(t)$, 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: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall\, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall\, j = 1, \cdots, k_i$

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

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall\, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall\, j = 1, \cdots, k_i$
  - this means that $\mathbf{P}^T\tilde{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T\mathbf{f}(\tilde{\mathbf{w}}(t), t)$
  - recalling that $\tilde{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_{\mathbf{f}}$ – that is,

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

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

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall\, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall\, j = 1, \cdots, k_i$
  - this means that $\mathbf{P}^T \tilde{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T \mathbf{f}(\tilde{\mathbf{w}}(t), t)$
  - recalling that $\tilde{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_{\mathbf{f}}$ – that is,

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

    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 the square matrix $\mathbf{P}^T \mathbf{V}_{\mathbf{f}}$ is nonsingular

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

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation
  - idea: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) = f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall\, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall\, j = 1, \cdots, k_i$
  - this means that $\mathbf{P}^T\tilde{\mathbf{f}}(\tilde{\mathbf{w}}(t), t) = \mathbf{P}^T\mathbf{f}(\tilde{\mathbf{w}}(t), t)$
  - recalling that $\tilde{\mathbf{f}}(\cdot, t)$ belongs to the range of $\mathbf{V}_{\mathbf{f}}$ – that is,

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

    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 the square matrix $\mathbf{P}^T\mathbf{V}_{\mathbf{f}}$ is nonsingular

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

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

$$\boxed{\tilde{\mathbf{f}}(\cdot, t) = \mathbf{V}_{\mathbf{f}}\underbrace{\left(\mathbf{P}^T\mathbf{V}_{\mathbf{f}}\right)^{-1}}_{k_{\mathbf{f}} \times k_{\mathbf{f}}}\mathbf{P}^T\mathbf{f}(\cdot, t) = \mathbf{\Pi}_{\mathbf{V}_{\mathbf{f}}, \mathbf{P}}\mathbf{f}(\cdot, t)}$$

- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation (continue)
    - note that the complexity of the projection

$$\left(\mathbf{W}^T\mathbf{V}\right)^{-1}\mathbf{W}^T\tilde{\mathbf{f}}(\cdot,t) = \underbrace{\left(\mathbf{W}^T\mathbf{V}\right)^{-1}\mathbf{W}^T\mathbf{V}_{\mathbf{f}}}_{pre-computable,\ \in\mathbb{R}^{k\times k_{\mathbf{f}}}}\left(\mathbf{P}^T\mathbf{V}_{\mathbf{f}}\right)^{-1}\mathbf{P}^T\mathbf{f}(\cdot,t)$$

is independent of N!

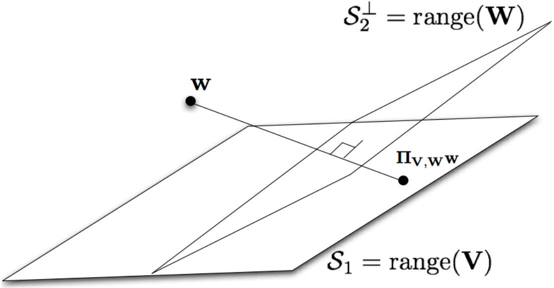
- Case where $k_i = k_{\mathbf{f}} \Rightarrow$ interpolation (continue)
  - note that the complexity of the projection

$$\left(\mathbf{W}^T\mathbf{V}\right)^{-1}\mathbf{W}^T\tilde{\mathbf{f}}(\cdot,t) = \underbrace{\left(\mathbf{W}^T\mathbf{V}\right)^{-1}\mathbf{W}^T\mathbf{V}_{\mathbf{f}}}_{pre-computable,\ \in\mathbb{R}^{k\times k_{\mathbf{f}}}}\left(\mathbf{P}^T\mathbf{V}_{\mathbf{f}}\right)^{-1}\mathbf{P}^T\mathbf{f}(\cdot,t)$$

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

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

- 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: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall \, j = 1, \cdots, N$

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

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

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

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

  - 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: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall \, j = 1, \cdots, N$
  - this leads to the minimization problem

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

  - 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: $\tilde{f}_{i_j}(\tilde{\mathbf{w}}, t) \approx f_{i_j}(\tilde{\mathbf{w}}, t), \ \forall \, \tilde{\mathbf{w}} \in \mathbb{R}^N, \ \forall \, j = 1, \cdots, N$
  - this leads to the minimization problem

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

  - 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

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

  - it follows that the left inverse of $\mathbf{M}$, $\mathbf{M}^\dagger$ – that is, the generalized inverse of $\mathbf{M}$ that satisfies $\mathbf{M}\mathbf{M}^\dagger\mathbf{M} = \mathbf{M}$ – is

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

where $\boldsymbol{\Sigma}^\dagger = \text{diag}\left(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0\right)$ if $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$, where $\sigma_1 \geq \cdots \sigma_r > 0$

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

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

  - 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

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

  - it follows that the left inverse of $\mathbf{M}$, $\mathbf{M}^\dagger$ – that is, the generalized inverse of $\mathbf{M}$ that satisfies $\mathbf{M}\mathbf{M}^\dagger\mathbf{M} = \mathbf{M}$ – is

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

  where $\boldsymbol{\Sigma}^\dagger = \operatorname{diag}\left(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0\right)$ if $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$, where $\sigma_1 \geq \cdots \sigma_r > 0$
  - therefore

$$\tilde{\mathbf{f}}(\mathbf{q}(t), t) = \mathbf{V}_{\mathbf{f}} \left(\mathbf{P}^T \mathbf{V}_{\mathbf{f}}\right)^\dagger \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t) = \mathbf{V}_{\mathbf{f}} \left(\mathbf{Z}\boldsymbol{\Sigma}^\dagger\mathbf{U}^T\right) \mathbf{P}^T \mathbf{f}(\mathbf{V}\mathbf{q}(t), t)$$

- 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 $\mathbf{V}_{\mathbf{f}} = [\mathbf{v}_{\mathbf{f},1} \quad \cdots \quad \mathbf{v}_{\mathbf{f},k_{\mathbf{f}}}]$ has been computed using, for example, POD

- Greedy algorithm

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

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, ...)
- typically destabilizes the PROM of a second-order dynamical system

- Energy-Conserving Sampling and Weighting (ECSW): The first hyperreduction method of the project-then-approximate type
  - developed first for nonlinear, finite element (FE)-based, solid mechanics and structural dynamics Galerkin PROMs
    - for such PROMs, the Gappy POD, Empirical Interpolation Method (EIM), Discrete EIM (DEIM), and the unassembled DEIM (UDEIM) are in general numerically unstable
    - reason: in the above hyperreduction methods, the interpolation procedure and the construction of the basis $\mathbf{V}$ are driven by accuracy considerations only

- Energy-Conserving Sampling and Weighting (ECSW): The first hyperreduction method of the project-then-approximate type
  - developed first for nonlinear, finite element (FE)-based, solid mechanics and structural dynamics Galerkin PROMs
    - for such PROMs, the Gappy POD, Empirical Interpolation Method (EIM), Discrete EIM (DEIM), and the unassembled DEIM (UDEIM) are in general numerically unstable
    - reason: in the above hyperreduction methods, the interpolation procedure and the construction of the basis $\mathbf{V}$ are driven by accuracy considerations only
  - for Galerkin PROMs constructed for second-order dynamical systems, it preserves the Lagrangian structure associated with Hamilton's principle

- Energy-Conserving Sampling and Weighting (ECSW): The first hyperreduction method of the project-then-approximate type
  - developed first for nonlinear, finite element (FE)-based, solid mechanics and structural dynamics Galerkin PROMs
    - for such PROMs, the Gappy POD, Empirical Interpolation Method (EIM), Discrete EIM (DEIM), and the unassembled DEIM (UDEIM) are in general numerically unstable
    - reason: in the above hyperreduction methods, the interpolation procedure and the construction of the basis $\mathbf{V}$ are driven by accuracy considerations only
  - for Galerkin PROMs constructed for second-order dynamical systems, it preserves the Lagrangian structure associated with Hamilton's principle
  - developed more recently for Petrov-Galerkin PROMs constructed for nonlinear, finite-volume-based, computational fluid dynamics applications

- Galerkin PROM $\Rightarrow \mathbf{W} = \mathbf{V} \in \mathbb{R}^{N\times k}$ (and $\mathbf{V}^T\mathbf{V} = \mathbf{I}_k$) $\Rightarrow$ PROM ($\star$) simplifies to

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

(for Petrov-Galerkin PROMs, see S. Grimberg, C. Farhat, R. Tezaur and C. Bou-Mosleh, 2021)

  - explicit time-integration: $\mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) \in \mathbb{R}^k \Rightarrow \mathcal{O}(kN)$ complexity
  - implicit time-integration: the above and $\mathbf{V}^T\mathbf{K}_t(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})\mathbf{V} \in \mathbb{R}^{k\times k}$, where

$$\mathbf{K}_t = \frac{\partial \mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}{\partial(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}))}$$

denotes the tangent operator $\Rightarrow \mathcal{O}(kN^2)$

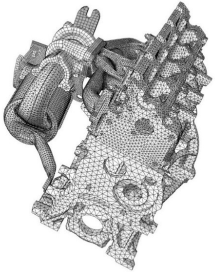
- Hyperreduction of the reduced "force" vector

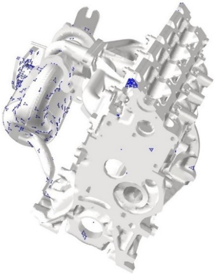
$$\mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) \quad = \quad \sum_{e\in\mathcal{E}}\left(\mathbf{L}^e\mathbf{V}\right)^T\mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$$

- Hyperreduction of the reduced "force" vector

$$\begin{aligned}\mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) &= \sum_{e\in\mathcal{E}} \left(\mathbf{L}^e\mathbf{V}\right)^T \mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) \\ &\approx \sum_{e\in\widetilde{\mathcal{E}}\subset\mathcal{E}} \xi^e \left(\mathbf{L}^e\mathbf{V}\right)^T \mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})\end{aligned}$$

where $\widetilde{N_e}(=|\widetilde{\mathcal{E}}|) \ll N_e(=|\mathcal{E}|)$





and $\xi^e > 0$ to preserve the strain energy property (also beneficial in general to the solution of the associated optimization problem)

- Hyperreduction of the reduced tangent (stiffness) matrix

$$\mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) \approx \sum_{e\in\widetilde{\mathcal{E}}\subset\mathcal{E}} \xi^e \left(\mathbf{L}^e\mathbf{V}\right)^T \mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})$$

and

$$\mathbf{K}_t = \frac{\partial\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})}{\partial(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}))}$$

$\Rightarrow$

$$\mathbf{V}^T\mathbf{K}_t(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})\mathbf{V} \approx \sum_{e\in\widetilde{\mathcal{E}}\subset\mathcal{E}} \xi^e \left(\mathbf{L}^e\mathbf{V}\right)^T \mathbf{K}^e(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})(\mathbf{L}^e\mathbf{V})$$

- The **reduced mesh** $\widetilde{\mathcal{E}}$ and the corresponding set of weights $\{\xi^e\}$ – or equivalently, the vector of weights $\boldsymbol{\xi} \in \mathbb{R}^{N_e}$ – are determined by training the hyperreduction approximation on force functions (or tangent stiffness matrices, see R. Tezaur, F. As'ad and C. Farhat, 2022) associated with sampled parameter points and thus with performed simulations

$$
\begin{aligned}
\mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) &\approx \sum_{e\in\widetilde{\mathcal{E}}\subset\mathcal{E}} \xi^e \left(\mathbf{L}^e\mathbf{V}\right)^T \mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu}) \\
\mathbf{V}^T\mathbf{K}_t(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})\mathbf{V} &\approx \sum_{e\in\widetilde{\mathcal{E}}\subset\mathcal{E}} \xi^e \left(\mathbf{L}^e\mathbf{V}\right)^T \mathbf{K}^e(\mathbf{V}\mathbf{q}(t;\boldsymbol{\mu}),\mathbf{u}(t),t;\boldsymbol{\mu})(\mathbf{L}^e\mathbf{V})
\end{aligned}
$$

- The ECSW hyperreduction approximation is a generalized (in a hyper space) quadrature rule approximation where
    - the sampled elements defining $\widetilde{\mathcal{E}}$ are the quadrature points
    - the coefficients $\{\xi^e > 0\}$ are the quadrature weights

- Using a lighter notation, where only the critical dependences are highlighted, let

$$\begin{aligned} g_i^e(\mathbf{q}_i) &= (\mathbf{L}^e\mathbf{V})^T\mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{q}_i) = (\mathbf{L}^e\mathbf{V})^T\mathbf{L}^e\mathbf{f}(\mathbf{V}\mathbf{V}^T\mathbf{w}_i) \in \mathbb{R}^k \\ b_i &= \sum_{e\in\mathcal{E}} g_i^e(\mathbf{q}_i) \in \mathbb{R}^k, \quad i = 1, \cdots, N_{\text{snap}} \\ \mathbf{G} &= \begin{pmatrix} g_{11} & \cdots & g_{1N_e} \\ \cdots & \cdots & \cdots \\ g_{N_{\text{snap}}1} & \cdots & g_{N_{\text{snap}}N_e} \end{pmatrix} \in \mathbb{R}^{(kN_{\text{snap}})\times N_e} \end{aligned}$$

and

$$\mathbf{b} = \begin{pmatrix} b_1 & \cdots & b_{N_{\text{snap}}} \end{pmatrix}^T \in \mathbb{R}^{kN_{\text{snap}}}$$

- Let now $\boldsymbol{\Phi} = \{\boldsymbol{\xi} \in \mathbb{R}^{N_e} : \|\mathbf{G}\boldsymbol{\xi} - \mathbf{b}\|_2 < \epsilon\|\mathbf{b}\|_2, \quad \boldsymbol{\xi} \geq \mathbf{0}\}$
- Then, the optimal reduced mesh and associated weights are given by

$$\underbrace{\boldsymbol{\xi}^{\text{opt}} = \arg\min_{\boldsymbol{\xi}\in\boldsymbol{\Phi}} \|\boldsymbol{\xi}\|_0}_{\text{optimal weights}} \quad \text{and} \quad \underbrace{\widetilde{\mathcal{E}}^{\text{opt}} = \{e \in \mathcal{E} / \xi^{\text{opt}^e} > 0\}}_{\text{optimal reduced mesh}}$$

$$\underbrace{\boldsymbol{\xi}^{\text{opt}} = \underset{\boldsymbol{\xi} \in \boldsymbol{\Phi}}{\arg\min} \|\boldsymbol{\xi}\|_0}_{\text{optimal weights}} \quad \text{and} \quad \underbrace{\widetilde{\mathcal{E}}^{\text{opt}} = \{e \in \mathcal{E} / \xi^{\text{opt}^e} > 0\}}_{\text{optimal reduced mesh}}$$

- Fastest alternative approach
  - alternative optimization problem

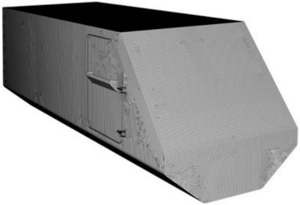
$$\boldsymbol{\xi}^{\text{opt}} = \underset{\substack{\boldsymbol{\xi} \in \mathbb{R}^{N_e} \\ \boldsymbol{\xi} \geq \mathbf{0}}}{\arg\min} \|\mathbf{G}\boldsymbol{\xi} - \mathbf{b}\|_2^2$$

  - solver: sparse non-negative least-squares (NNLS) solver, where the iterative process is termined as soon as

$$\|\mathbf{G}\boldsymbol{\xi} - \mathbf{b}\|_2 < \epsilon \|\mathbf{b}\|_2$$

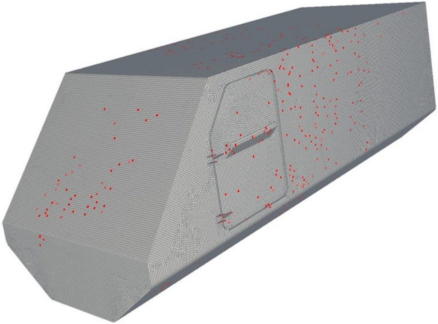
  - the above solution algorithm is amenable to parallelization

- Nonlinear FE structural model of a generic V-hull
  - J2 plasticity constitutive model
  - shell and beam elements with finite rotations: $|\mathcal{E}| = 236,995$ elements
  - 233 276 nodes and 1 399 056 dofs



- Air blast loading: 10 kg charge
- Simulation time-interval: $[0,\ 10^{-3}]$ s
- POD-based PROM of dimension $k = 100 \ll N = 1\,399\,056$
- Explicit analysis: $(\Delta t)_{\text{HDM}} = 10^{-8}$ s; $(\Delta t)_{\text{PROM}} = 2.5 \times 10^{-6}$ s

- ECSW with: $\epsilon = 2.5 \times 10^{-2}$ and $N_{\text{snap}} = k = 100 \Rightarrow$ delivers in 16 minutes on 64 cores of a Unix cluster an optimal reduced mesh with $\widetilde{N}_e = |\widetilde{\mathcal{E}}| = 2\,000 = 0.8\%$ of N_e $(= |\mathcal{E}| = 236\,995)$ – Circa 2014



- For this application, ECSW and the reduced mesh it produces enable the PROM to deliver a speedup factor of 28 935, while achieving 96% relative accuracy – Circa 2014