Outline

1. Nested Approximations
2. Trajectory PieceWise Linear (TPWL) Method
3. Hyper-Reduction Methods
4. Local Approaches
5. References
Nested Approximations

Nonlinear HDM

HDM of interest

\[
\frac{dw(t; \mu)}{dt} = f(w(t; \mu), u(t); t; \mu)
\]

\[
y(t; \mu) = g(w(t; \mu), u(t); t, \mu)
\]

- \(w \in \mathbb{R}^N\): Vector of state variables
- \(u \in \mathbb{R}^p\): Vector of input variables, typically \(p \ll N\)
- \(y \in \mathbb{R}^q\): Vector of output variables, typically \(q \ll N\)
- \(\mu \in \mathbb{R}^m\): Vector of parameter variables, typically \(m \ll N\)
- \(f\): Nonlinear function

Usually, there is no closed form solution for \(w(t; \mu)\)
Approximation of the state using a right ROB

\[ \mathbf{w}(t; \mu) \approx \mathbf{Vq}(t; \mu) \]

Resulting nonlinear ODE

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

Enforcement of the orthogonality of the residual \( \mathbf{r} \) to a left ROB \( \mathbf{W} \)

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

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

\[ \frac{d}{dt} \mathbf{q}(t; \mu) = (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T \mathbf{f} (\mathbf{Vq}(t; \mu), \mathbf{u}(t); t; \mu) \]
Petrov-Galerkin projection-based ROM

\[ \frac{d}{dt} q(t; \mu) = (W^T V)^{-1} W^T f(Vq(t; \mu), u(t); t; \mu) \]

- \( k \) equations in terms of \( k \) unknowns
- For a given reduced state \( q(t; \mu) \), the evaluation of \( f_k(q(t; \mu), u(t); t, \mu) = (W^T V)^{-1} W^T f(Vq(t; \mu), u(t); t; \mu) \) at a given time \( t \) and parameter \( \mu \) is performed in 3 steps
  1. compute \( w(t; \mu) = Vq(t; \mu) \)
  2. evaluate \( f(Vq(t; \mu), u(t); t; \mu) \)
  3. left-multiply the result by \((W^T V)^{-1} W^T\) to obtain \((W^T V)^{-1} W^T f(Vq(t), t)\)

The computational cost associated with these three steps scales linearly with the dimension \( N \) of the HDM

Hence, for nonlinear problems, dimensional reduction as described above 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

There are two main classes of nonlinear function approximation:

- linearization approaches (TPWL, ManiMOR,...)
- hyper-reduction approaches (DEIM, ECSW, GNAT,...)
Consider a nonlinear HDM of the form

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

- stationary system
- no parametric dependence for now
- separable linear input

For linear HDMs, reduced-order operators of the type

\[
\mathbf{A}_r = (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T \mathbf{AV}
\]

can be pre-computed offline once for all

Idea: linearize \( f \) around an operating point \( \mathbf{w}_1 \)

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

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

\[
\frac{d}{dt} \mathbf{w}(t) = \mathbf{A}(\mathbf{w}_1)\mathbf{w}(t) + [\mathbf{B}u(t) + (f(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1)]
\]
Approximated HDM system

\[
\frac{d}{dt} \mathbf{w}(t) = \mathbf{A}(\mathbf{w}_1)\mathbf{w}(t) + [\mathbf{B}\mathbf{u}(t) + (\mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1)]
\]

Reduced-order system after Petrov-Galerkin projection

\[
\frac{d}{dt} \mathbf{q}(t) = (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T \mathbf{A}(\mathbf{w}_1)\mathbf{V}\mathbf{q}(t) \\
+ (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T [\mathbf{B}\mathbf{u}(t) + (\mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1)]
\]

The following linear time-invariant operators can be pre-computed

- \( \mathbf{A}_r = (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T \mathbf{A}(\mathbf{w}_1)\mathbf{V} \in \mathbb{R}^{k \times k} \)
- \( \mathbf{B}_r = (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T \mathbf{B} \in \mathbb{R}^{k \times p} \)
- \( \mathbf{F}_r = (\mathbf{W}^T \mathbf{V})^{-1} \mathbf{W}^T [(\mathbf{f}(\mathbf{w}_1) - \mathbf{A}(\mathbf{w}_1)\mathbf{w}_1)] \in \mathbb{R}^k \)
■ 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

\[
\frac{d}{dt} w(t) = \sum_{i=1}^{s} \omega_i(w(t))(f(w_i) + A_i(w(t) - w_i)) + Bu(t)
\]

\[
y(t) = g(w(t), u(t); t)
\]

■ the s points \( \{w_i\}_{i=1}^{s} \) are linearization points

■ the s coefficients \( \{\omega_i\}_{i=1}^{s} \) are weights such that

\[
\sum_{i=1}^{s} \omega_i(w) = 1, \quad \forall w \in \mathbb{R}^N
\]
For simplicity, assume $W^TV = I_k$: In this case, the ROM obtained via Petrov-Galerkin projection is

$$\frac{dq(t)}{dt} = \sum_{i=1}^{s} \tilde{\omega}_i(q(t)) (W^Tf(w_i) + W^TA_i(Vq(t) - w_i)) + W^TBu(t)$$

$$y(t) = g(Vq(t), u(t); t)$$

where

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

Equivalently

$$\frac{dq(t)}{dt} = \left( \sum_{i=1}^{s} \tilde{\omega}_i(q(t)) A_{ri} \right) q(t) + \left( \sum_{i=1}^{s} \tilde{\omega}_i(q(t)) \right) F_{ri} + B_ru(t)$$

- $A_{ri} = W^T A_i V$, $i = 1, \ldots, s$
- $B_r = W^T B$
- $F_{ri} = W^T (f(w_i) - A_i w_i)$, $i = 1, \ldots, s$
In this context, a complete model order reduction method should provide algorithms for:

- selecting the linearization points \( \{w_i\}_{i=1}^s \)
- selecting the ROBs \( V \) and \( W \)
- Determining the weights \( \{\tilde{\omega}_i(q)\}_{i=1}^s, \forall q \in \mathbb{R}^k \)
Note that each linear approximation of the nonlinear function $f$ is valid only in a neighborhood of each $w_i$.

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) model order reduction 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 $V$ include

- if the input function is linear in $u$, constructing Krylov subspaces $\mathcal{K}_i = \mathcal{K}(A_i^{-1}, A_i^{-1}B) = \text{range}(V_i)$ at each linearization point $w_i$ and assembling a global basis $V$ such that

\[
\text{range}(V) = \text{range}\left([V_1 \ \cdots \ V_s]\right)
\]

- ad-hoc methods (Balanced truncation, POD...)

The left ROB $W$ can be chosen based on the output of interest (two-sided Krylov moment matching), or simply as $W = V$ (Galerkin projection)
The weights are used to characterize in the reduced space $\mathbb{R}^k$ the distance of the current point $q(t)$ to the projection of the linearization points onto $\text{range}(V)$ – that is,

$$\{q_i = (V^T V)^{-1} V^T w_i\}_{i=1}^s$$

one possible choice is

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

where $\beta$ is a constant, $d_i = \|q - q_i\|_2$, and $m = \min_{j=1}^s d_j$

other choices can be found in the literature
A posteriori error estimators are available when $f$ is negative monotone.

Stability guarantee is possible under some assumptions on $f$ and specific choices for $V$ and the weights $\{\tilde{\omega}_i(q)\}_{i=1}^s$.

Passivity preservation (i.e. no energy creation in a passive system) is possible under similar assumptions.

TPWL using local ROBs (ManiMOR)
**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, ...)

---

CME 345: MODEL REDUCTION - Methods for Nonlinear Systems

- Trajectory PieceWise Linear (TPWL) Method
- Analysis of the TPWL Method
Hyper-Reduction Methods

The Gappy POD

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

- Other applications
  - flow sensing and estimation
  - flow reconstruction
  - nonlinear model order reduction
CME 345: MODEL REDUCTION - Methods for Nonlinear Systems

Hyper-Reduction Methods

The Gappy POD

- Face recognition

**Procedure**

1. build a database of $N_s$ faces (snapshots)
2. construct a POD basis $V_f$ for the database
3. for a new face $f$, record a small number $k_i$ of pixels $f_{i1}, \cdots, f_{ik}$
4. using the POD basis $V_f$, approximately reconstruct the new face $f$ (in the least-squares sense)

![Reconstruction of a face](image)

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 $f$ in the reduced equations

$$\frac{d}{dt}q(t) = W^T f(Vq(t); t)$$

(for simplicity, the input function $u(t)$ is not considered here)

The evaluation of all the entries of $f(\cdot; t)$ is computationally expensive (scales with $N$)

Gappy approach
- only a small subset of these entries is evaluated
- the other entries are reconstructed either by interpolation or a least-squares strategy using a pre-computed ROB $V_f$

The dimension of the solution space is still reduced using any preferred model order reduction method (for example, POD)
A complete model order reduction method based on the Gappy approach should then provide algorithms for

- selecting the evaluation entries \( I = \{i_1, \cdots, i_{k_I}\} \)
- selecting a reduced-order basis \( V_f \) for the nonlinear function \( f \)
- reconstructing the complete approximated nonlinear function \( \hat{f}(\cdot; t) \)
Construction of a POD basis $\mathbf{V}_f$ of dimension $k_f$

1. collect snapshots for the nonlinear function $f$ from one or several transient simulations

$$\mathbf{F} = \begin{bmatrix} f(\mathbf{w}(t_1); t_1) & \cdots & f(\mathbf{w}(t_{m_f}); t_{m_f}) \end{bmatrix} \in \mathbb{R}^{N \times m_f}$$

2. compute a thin SVD

$$\mathbf{F} = \mathbf{U}_f \mathbf{\Sigma}_f \mathbf{Z}_f^T$$

3. truncate the ROB to a dimension $k_f \leq m_f$ by selecting the first $k_f$ vectors in $\mathbf{U}_f$

$$\mathbf{V}_f = \begin{bmatrix} \mathbf{u}_{f,1} & \cdots & \mathbf{u}_{f,k_f} \end{bmatrix}$$
Assume that $k_i$ indices (entries) have been chosen (the choice of indices will be specified later)

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

Consider the $N \times k_i$ “mask” matrix

$$P = \begin{bmatrix} e_{i_1} & \cdots & e_{i_{k_i}} \end{bmatrix}$$

At each time $t$, given a value of the state $w(t) = Vq(t)$, evaluate only those entries of $f$ corresponding to the above indices

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

This is computationally economical if $k_i \ll N$

Usually, only a subset of the entries of $w(t)$ are required to construct the above vector (case of a sparse Jacobian)
Hyper-Reduction Methods

Discrete Empirical Interpolation Method (DEIM)

- Case where $k_i = k_f \Rightarrow$ interpolation
  - idea: $\hat{f}_{ij}(w; t) = f_{ij}(w; t), \ \forall w \in \mathbb{R}^N, \ \forall j = 1, \ldots, k_i$
  - this means that
    \[ P^T \hat{f}(w(t); t) = P^T f(w(t); t) \]
  - recalling that $\hat{f}(\cdot; t)$ belongs to the range of $V_f$ – that is,
    \[ \hat{f}(Vq(t); t) = V_f f_r(q(t); t), \ \text{where} \ f_r(q(t); t) \in \mathbb{R}^{k_f} \]
  - then
    \[ P^T V_f f_r(q(t); t) = P^T f(Vq(t); t) \]
  - assuming that $P^T V_f$ is nonsingular
    \[ f_r(q(t); t) = (P^T V_f)^{-1} P^T f(Vq(t); t) \]
  - interpolating the high-dimensional nonlinear function $\hat{f}(\cdot; t)$ as follows
    \[ \hat{f}(\cdot; t) = V_f (P^T V_f)^{-1} P^T f(\cdot; t) = \Pi_{V_f, P} f(\cdot; t) \]
  - the Discrete Empirical Interpolation Method (DEIM) results in an oblique projection of the high-dimensional nonlinear vector
Hyper-Reduction Methods

Oblique Projection of the High-Dimensional Nonlinear Vector

\[ \hat{f}(\cdot, t) = V_f (P^T V_f)^{-1} P^T f(\cdot, t) = \Pi_{V_f, P} f(\cdot, t) \]

- \( \Pi_{V, W} = V (W^T V)^{-1} W^T \): Oblique projector onto \( V \), orthogonally to \( W \)

![Diagram showing the oblique projection](image-url)
■ Case where \( k_i > k_f \) \( \Rightarrow \) least-squares reconstruction

■ idea: \( \hat{f}_{ij}(w; t) \approx f_{ij}(w; t), \ \forall w \in \mathbb{R}^N, \ \forall j = 1, \cdots, N, \) in the least-squares sense

■ this leads to the minimization problem

\[
    \hat{f}_r(q(t); t) = \text{argmin} \left\| P^T V f_r y_r - P^T f(V q(t); t) \right\|_2, \quad y_r \in \mathbb{R}^{k_f}
\]

■ note that \( M = P^T V f \in \mathbb{R}^{k_i \times k_f} \) is a skinny matrix

■ its singular value decomposition can be written as

\[
    M = U \Sigma Z^T
\]

■ then, the left inverse of \( M \) can be defined as

\[
    M^\dagger = Z \Sigma^\dagger U^T
\]

where \( \Sigma^\dagger = \text{diag}(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0) \) if

\[
    \Sigma = \text{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0), \text{ where } \sigma_1 \geq \cdots \sigma_r > 0
\]

■ and therefore

\[
    \hat{f}(q(t); t) = V_f \left( Z \Sigma^\dagger U^T \right) P^T f(V q(t); t)
\]

\[
    = V_f \left( P^T V_f \right)^\dagger P^T f(V q(t); t)
\]
This selection takes place after the matrix \( V_f = [v_{f,1} \ldots v_{f,k_f}] \) has been computed using, for example, POD.

**Greedy algorithm**

1. \([s, i_1] = \max\{|v_{f,1}|\}\)
2. \(V_f = [v_{f,1}], P = [e_{i_1}]\)
3. **for** \(l = 2 : k_f\) **do**
4. solve \(P^T V_f c = P^T v_{f,l}\) for \(c\)
5. \(r = v_{f,l} - V_f c\)
6. \([s, i_l] = \max\{|r|\}\)
7. \(V_f = [V_f, v_{f,l}], P = [P, e_{i_l}]\)
8. **end for**
Hyper-Reduction Methods

Analysis of Hyper-Reduction Methods

**Strengths**

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

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

- **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
Hyper-Reduction Methods

Application to the Reduction of the Burgers Equation

- \( k = 15, k_f = 40, k_i = 40 \)

![Graph showing results for different values of \( k_i \)]

- Similar results for \( k_i > 40 \) (least-squares reconstruction)
Results of the greedy algorithm
The dimension $k_f$ of the ROB $\mathbf{V}_f$ is reduced from 40 to 30

- $k = 15$, $k_f = 30$, $k_i = 80$

Similar results for $k_i = 100$ (no gaps)

- $k_f$ is too small in that case
Hyper-Reduction Methods

Model Reduction at the Fully Discrete Level

- Semi-discrete level: $\frac{d}{dt} \mathbf{w}(t) = \mathbf{f}(\mathbf{w}(t); t)$
- Subspace approximation: $\mathbf{w}(t) \approx \mathbf{Vq}(t)$

$$\mathbf{V} \frac{d}{dt} \mathbf{q}(t) \approx \mathbf{f}(\mathbf{Vq}(t); t)$$

- Fully discrete level (implicit, backward Euler scheme)

$$\mathbf{V} \frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t^n} \approx \mathbf{f}(\mathbf{Vq}^{n+1}; t^{n+1})$$

- Fully discrete residual

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

- Residual minimization (a.k.a model order reduction by least-squares or Petrov-Galerkin projection)

$$\mathbf{q}^{n+1} = \arg\min_{\mathbf{y} \in \mathbb{R}^k} \| \mathbf{r}^{n+1}(\mathbf{y}) \|_2$$

- $\mathbf{r}(\mathbf{q}^{n+1})$ is nonlinear $\Rightarrow$ use the gappy POD hyper-reduction
Gappy POD procedure for the fully discrete residual $r$

Algorithm

1. build a reduced-order basis $V_r \in \mathbb{R}^{N \times k_r}$ for $r$ such that $V_r^T V = I_{k_r}$
2. construct a sample mesh $\mathcal{I}$ (indices $i_1, \cdots, i_{k_i}$) using the greedy procedure
3. consider the gappy approximation

$$r^{n+1}(q^{n+1}) \approx V_r r_{k_r}(q^{n+1}) \approx V_r \left( P^T V_r \right)^\dagger P^T r^{n+1}(Vq^{n+1})$$

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

$$q^{n+1} = \arg\min_{y \in \mathbb{R}^k} \|V_r r_{k_r}(y)\|_2$$

$$= \arg\min_{y \in \mathbb{R}^k} \|r_{k_r}(y)\|_2$$

$$= \arg\min_{y \in \mathbb{R}^k} \left\| \left( P^T V_r \right)^\dagger P^T r^{n+1}(Vy) \right\|_2$$
Nonlinear least-squares problem: \( \min_y \| r(y) \|_2 \), where \( r \in \mathbb{R}^N \), \( y \in \mathbb{R}^k \), and \( k \ll N \)

Equivalent function to be minimized: \( \phi(y) = \frac{1}{2} \| r(y) \|^2 = r(y)^T r(y) \)

Gradient: \( \nabla \phi(y) = J(y)^T r(y) \), where \( J(y) = \frac{\partial r}{\partial y}(y) \)

Iterative solution using Newton’s method

\[
y^{(j+1)} = y^{(j)} + \Delta y^{(j+1)}
\]

where

\[
\nabla^2 \phi(y^{(j)}) \Delta y^{(j+1)} = -\nabla \phi(y^{(j)})
\]

What is \( \nabla^2 \phi(y) \)?

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

Gauss-Newton method

\[
\nabla^2 \phi(y) \approx J(y)^T J(y)
\]
Gauss-Newton method

\[ y^{(j+1)} = y^{(j)} + \Delta y^{(j+1)} \]

where

\[ J(y^{(j)})^T J(y^{(j)}) \Delta y^{(j+1)} = -J(y^{(j)})^T r(y^{(j)}) \]

This is the normal equation for

\[ \Delta y^{(j+1)} = \arg\min_z \| J(y^{(j)})z + r(y^{(j)}) \|_2 \]

QR decomposition of the Jacobian

\[ J(y^{(j)}) = Q^{(j)} R^{(j)} \]

Equivalent solution using the QR decomposition (assuming that \( R^{(j)} \) is full rank)

\[ \Delta y^{(j+1)} = -J(y^{(j)})^\dagger r(y^{(j)}) = - \left( R^{(j)} \right)^{-1} \left( Q^{(j)} \right)^T r(y^{(j)}) \]
GNNAT (Gauss-Newton with Approximated Tensors) = Gauss-Newton + gappy POD

**Minimization problem**

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

**Jacobian:** \( \hat{J}(y) = (P^T V_r)^\dagger P^T J^{n+1}(Vy) \)

**Define a small dimensional operator (and construct it offline)**

\[
A = (P^T V_r)^\dagger
\]

**Least-squares problem at Gauss-Newton iteration \( j \) of \( t^{n+1} \)**

\[
\Delta y^{(j)} = \arg\min_{z \in \mathbb{R}^k} \left\| AP^T J^{n+1}(Vy^{(j)}) Vz + AP^T r^{n+1}(Vy^{(j)}) \right\|_2
\]

**GNAT solution using QR**

\[
AP^T J^{n+1}(Vy^{(j)}) V = Q^{(j)} R^{(j)}
\]

\[
\Delta y^{(j)} = - \left( R^{(j)} \right)^{-1} \left( Q^{(j)} \right)^T AP^T r^{n+1}(Vy^{(j)})
\]
Further developments

- concept of a reduced mesh
- concept of an output mesh
- error bounds
- GNAT using local reduced-order bases
Hyper-Reduction Methods

Application: Compressible Navier-Stokes Equations

- Turbulent flow past the Ahmed body (CFD benchmark in the automotive industry)
- 3D compressible Navier-Stokes equations
- $N = 1.73 \times 10^7$
- $Re = 4.48 \times 10^6$, $M_\infty = 0.175$ (216km/h)
Model order reduction (POD + GNAT): $k = 283$, $k_f = 1,514$, $k_i = 2,268$

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Time</th>
<th>Number of CPUs</th>
<th>Relative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDM</td>
<td>13.28 h</td>
<td>512</td>
<td>–</td>
</tr>
<tr>
<td>ROM (GNAT)</td>
<td>3.88 h</td>
<td>4</td>
<td>0.68%</td>
</tr>
</tbody>
</table>
Comparison to other model order reduction techniques

- full-order model
- GNAT(1)
- collocation + Gal
- collocation + LS
- DEIM-like

Drag coefficient, $C_D$

Time, seconds
Hyper-Reduction Methods

Application: Design Optimization of a Nozzle

HDM:

\[ N = 2 \]
\[ n = 5 \] shape parameters

Model order reduction (POD + DEIM):

\[ k = 8 \]
\[ k_f = 20 \]
\[ k_i = 20 \]

Steady, parameterized problem

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

s.t. \[ f(w(\mu), \mu) = 0 \]

where \( M \) denotes the local Mach number
Hyper-Reduction Methods

Application: Design Optimization of a Nozzle

<table>
<thead>
<tr>
<th>Method</th>
<th>Offline CPU Time</th>
<th>Online CPU Time</th>
<th>Total CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDM</td>
<td>–</td>
<td>78.8 s</td>
<td>78.8 s</td>
</tr>
<tr>
<td>ROM (DEIM)</td>
<td>5.08 s</td>
<td>4.87 s</td>
<td>9.96 s</td>
</tr>
</tbody>
</table>

![Graph of M(x)](image1)

![Graph of A(x)](image2)
Approximating the solution manifold $\mathcal{M}$ by a single subspace $S$ can lead to a large-dimensional such subspace.

Idea: Approximate $\mathcal{M}$ using \textbf{local} subspaces $\{S_i\}_{i=1}^L$. 

![Diagram showing local subspaces approximating the solution manifold $\mathcal{M}$]
In practice, the local approximation of the state takes place at the **fully discrete level**

Each local subspace $S_l$ is associated with a local ROB $V_l$

At each time-step $n$, the state $w^n$ is computed as

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

The increment $\Delta w^n$ is then approximated in a subspace $S_{l,n} = \text{range}(V_{l,n})$ as

$$\Delta w^n \approx V_{l,n}q^n$$

The choice of the reduced-order basis $V_{l,n}$ is specified later

By induction, the state $w^n$ is computed as

$$w^n = w^0 + \sum_{i=1}^{n} V_{l,i}q^n$$
Local Approaches

Local Approximation of the State

- The state $w^n$ is computed as

$$w^n = w^0 + \sum_{i=1}^{n} V_{l,i} \tilde{q}^n$$

- In practice, the ROBs $\{V_{l,i}\}_{i=1}^{n}$ are chosen among a finite set of local ROBs $\{V_{l}\}_{l=1}^{L}$

- Hence

$$w^n = w^0 + \sum_{l=1}^{L} V_{l} q_{l}^n$$

- This shows that

$$w^n \in w^0 + \text{range}([V_1 \cdots V_V])$$

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

$$V_l \in \mathbb{R}^{N \times k_l}$$
Intuitively, a given local subspace $S_l$ 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 $S_l = \text{range}(V_l)$.

In practice, a set of solution snapshots $\{w_i\}_{i=1}^{N_s}$ 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 Approaches

Construction of the Local ROBs

- Local ROBs construction procedure
Local ROBs construction procedure
Local ROBs construction procedure
Local ROBs construction procedure
Online, at time-step $n$, a local ROB $V_{l,n}$ needs to be chosen.

The selection is based on the current location of $w^{n-1}$ on the solution manifold $M$.

The local approximation subspace is selected as that associated with the cluster whose center is the closest to $w^{n-1}$

$$l, n = \arg\min_{l \in \{1, \ldots, L\}} d(w^{n-1}, w^c_l)$$

Consider the case of the distance based on a weighted Euclidean norm

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

where $H \in \mathbb{R}^{N \times N}$ is a symmetric positive definite matrix.
Choice of the local approximation subspace at time-step $n$

$$l, n = \arg\min_{l \in \{1, \ldots, L\}} d(w^{n-1}, w^c_l)$$

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

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

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

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

$$\Delta_{i,j} = d(w^{n-1}, w^c_i)^2 - d(w^{n-1}, w^c_j)^2$$

$$= \|w^{n-1} - w^c_i\|_H^2 - \|w^{n-1} - w^c_j\|_H^2$$

$$= \|\sum_{l=1}^{L} V_l q^{n-1}_l\|_H^2 + \|w^c_i - w^0\|_H^2 - 2 \sum_{l=1}^{L} [w^c_i]^T V_l q^{n-1}_l$$

$$- \|\sum_{l=1}^{L} V_l q^{n-1}_l\|_H^2 - \|w^c_j - w^0\|_H^2 + 2 \sum_{l=1}^{L} [w^c_j]^T V_l q^{n-1}_l$$

$$= \|w^c_i - w^0\|_H^2 - \|w^c_j - w^0\|_H^2 + 2 \sum_{l=1}^{L} [w^c_i - w^c_j]^T V_l q^{n-1}_l$$

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

$$a_{i,j} = \|w^c_i - w^0\|_H^2 - \|w^c_j - w^0\|_H^2 \in \mathbb{R}, \quad g_{i,j} = [w^c_i - w^c_j]^T V_l \in \mathbb{R}^{k_l}$$
The local approach to nonlinear model reduction can be easily extended to hyper-reduction as follows.

The hyper-reduction approach is applied independently to each subset of snapshots.

It leads to the definition of:
- the local ROBs for the state: $V_l$, $l = 1, \ldots, L$
- the local ROBs for the residual: $V_{r,l}$, $l = 1, \ldots, L$
- the local masks: $I_l$, $l = 1, \ldots, L$

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 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 \( M_\infty = 0.9 \)
Model order reduction using a global ROB
Model order reduction using 5 local ROBs

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


References