Parametric Model Order Reduction: An Introduction

Maria Cruz Varona

Chair of Automatic Control
Department of Mechanical Engineering
Technical University of Munich

Model Order Reduction Summer School
September 24th 2019
Linear Model Order Reduction
Projective Non-Parametric MOR

Linear time-invariant (LTI) system

\[
G(s) : \begin{cases} 
E \dot{x}(t) = Ax(t) + Bu(t) \\
y(t) = Cx(t) 
\end{cases} 
\]

\(E, A \in \mathbb{R}^{n \times n}\)
\(B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{q \times n}\)

\[r \ll n\]

Projection

\[
V, W \in \mathbb{R}^{n \times r}
\]
\(E_r = W^T E V, A_r = W^T A V, B_r = W^T B, C_r = C V\)

Reduced order model (ROM)

\[
G_r(s) : \begin{cases} 
E_r \dot{x}_r(t) = A_r x_r(t) + B_r u(t) \\
y_r(t) = C_r x_r(t) 
\end{cases} 
\]

\(E_r, A_r \in \mathbb{R}^{r \times r}\)
\(B_r \in \mathbb{R}^{r \times m}, C_r \in \mathbb{R}^{q \times r}\)
Linear MOR methods – Overview

1. **Modal Reduction (modalMOR)**
   - Preservation of dominant eigenmodes
   - Frequently used in structural dynamics / second order systems

2. **Truncated Balanced Realization (TBR) / Balanced Truncation (BT)**
   - Retention of state-space directions with highest energy transfer
   - Requires solution of Lyapunov equations, i.e. linear matrix equations (LMEs)
   - Applicable for medium-scale models: $n \approx 5000$

3. **Rational Krylov subspaces (RK)**
   - “Moment Matching”: matching some Taylor-series coefficients of the transfer function
   - Requires solution of linear systems of equations (LSEs) – applicable for $n \approx 10^6$
   - Also employed for: approximate solution of eigenvalue problems, LSEs, LMEs,…

4. **Iterative Rational Krylov algorithm (IRKA)**
   - H2-optimal reduction
   - Adaptive choice of Krylov reduction parameters (e.g. shifts)
Truncated Balanced Realization (TBR)

**Goal:** Preserve state-space directions with highest energy transfer

**Controllability and Observability Gramians:**

\[
P = \int_0^\infty e^{(E^{-1}A)t} E^{-1} \, B \, B^T \, E^{-T} e^{A^T E^{-T} t} \, dt
\]

\[
Q = \int_0^\infty e^{A^T E^{-T} t} C^T C e^{E^{-1}A t} \, dt
\]

Energy interpretation:

\[
\min \int_0^\infty |u(t)|^2 \, dt = x_e^T \, P^{-1} \, x_e
\]

\[
\|y(t)\|_2^2 = x_0^T \, Q \, x_0
\]

Lyapunov equations:

\[
A \, P \, E^T + E \, P \, A^T + B \, B^T = 0, \quad A^T \, Q \, E + E^T \, Q \, A + C^T \, C = 0
\]

**Procedure:**

1. **Balancing step:** Compute balanced realization, where \( P = E^T \, Q \, E = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n) \)

   \[
P = RR^T, \quad Q = SS^T
\]

   \[
S^T \, ER = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & \Sigma_2 \end{bmatrix} \begin{bmatrix} N_1^T \\ N_2^T \end{bmatrix}
\]

2. **Truncation step:** \( \sigma_i \gg \sigma_j, \quad i = 1, \ldots, r, \quad j = r + 1, \ldots, n \)

   \[
W^T = \Sigma_1^{-1/2} \, U_1^T \, S^T, \quad V = R \, N_1 \, \Sigma_1^{-1/2}
\]
Rational Interpolation by Krylov subspaces

Moments of a transfer function

\[ G(s) = C(sE - A)^{-1} B \]

\[ = G(\Delta s + \sigma) = \sum_{i=0}^{\infty} M_i(\sigma) (s - \sigma)^i \]

\( \sigma \): interpolation point (shift)

\( M_i(\sigma) \): i-th moment around \( \sigma \)

(Multi)-Moment Matching by Rational Krylov (RK) subspaces

Bases for input and output Krylov-subspaces:

\[ \text{Ran}(V) \supseteq \text{span} \left\{ A_\sigma^{-1}B, \ A_\sigma^{-1}EA_\sigma^{-1}B, \ldots, (A_\sigma^{-1}B)^{r-1}A_\sigma^{-1}B \right\} \]

\[ \text{Ran}(W) \supseteq \text{span} \left\{ A_\sigma^{-T}C^T, \ A_\sigma^{-T}EA_\sigma^{-T}C^T, \ldots, (A_\sigma^{-T}E^T)^{r-1}A_\sigma^{-T}C^T \right\} \]

\( M_i(\sigma) = M_{r,i}(\sigma) \) for \( i = 0, \ldots, 2r - 1 \)

Moments from full and reduced order model around certain shifts match!
Comparison: BT vs. Krylov subspaces

**Balanced Truncation (BT)**
- + stability preservation
- + automatable
- + error bound (a priori)

- – computing-intensive
- – storage-intensive
- – n < 5000

**Rational Krylov (RK) subspaces**
- + numerically efficient
- + n ≈ 10^6
- + $H_2$-optimal (IRKA)
- + many degrees of freedom

- – many degrees of freedom
- – stability gen. not preserved
- – no error bounds

**Subject of research**
- Numerically efficient solution of large-scale Lyapunov equations
  - Krylov-based Low-Rank Approximation
    - ADI (Alternating Directions Implicit)
    - RKSM (Rational Krylov Subspace Method)

**Subject of research**
- Adaptive choice of reduction parameters
  - Reduced order
  - Interpolation data (shifts, etc.)
- Stability preservation
- Numerically efficient computation of rigorous error bounds
Parametric Model Order Reduction
Parametric Model Order Reduction (pMOR)

Large-scale parametric model

\[ E(p) \dot{x} = A(p) x + B(p) u \]
\[ y = C(p) x \quad p \in D \subset \mathbb{R}^d \]

Reduced order parametric model

\[ E_r(p) \dot{x}_r = A_r(p) x_r + B_r(p) u \]
\[ y_r = C_r(p) x_r \quad x_r \in \mathbb{R}^r, \ r \ll n \]

- Linear dynamic systems with design parameters (e.g. material / geometry parameters, …)
- **Goal:** numerically efficient reduction with preservation of the parameter dependency
  → variation of the parameters in the ROM without having to repeat the reduction every time!

Flow sensing anemometer

Timoshenko beam

Microthruster unit
Projective Parametric MOR

Linear parametric system

\[ \begin{align*}
G(s, p) : & \quad E(p) \dot{x}(t) = A(p)x(t) + B(p)u(t) \\
& \quad y(t) = C(p)x(t)
\end{align*} \]

\( x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, y(t) \in \mathbb{R}^q, p \in D \subset \mathbb{R}^d \)

High computational effort and storage requirement needed for simulation, optimization and control

\[ \begin{array}{c}
pMOR \\
\end{array} \]

\[ r \ll n \]

Reduced parametric system

\[ \begin{align*}
G_r(s, p) : & \quad E_r(p) \dot{x}_r(t) = A_r(p)x_r(t) + B_r(p)u(t) \\
& \quad y_r(t) = C_r(p)x_r(t)
\end{align*} \]

\[ E_r(p) = W(p)^T E(p)V(p), \quad A_r(p) = W(p)^T A(p)V(p) \]

\[ B_r(p) = W(p)^T B(p), \quad C_r(p) = C(p)V(p) \]
# Overview pMOR approaches

## Global approaches

<table>
<thead>
<tr>
<th>Common subspaces $V(p), W(p)$ for all $p \in D \subset \mathbb{R}^d$</th>
</tr>
</thead>
</table>

## Local approaches

<table>
<thead>
<tr>
<th>Individual subspaces $V(p_i), W(p_i)$ for local systems at $p_1, p_2, \ldots, p_K$</th>
</tr>
</thead>
</table>

### Multi-Parameter Moment Matching

- [Weile ’99, Daniel ’04]
  - Moment Matching w.r.t. $s$ and $p$
  - Explicit parameter dependency required
  - Curse of dimensionality

### Concatenation of local bases

- [Leung ’05, Li ’05, Baur et al. ’11]
  - Computation of $V_1, W_1, \ldots, V_K, W_K$ using BT, RK, IRKA or POD
  - Concatenation of the local bases $V(p) = [V_1, \ldots, V_K], W(p) = [W_1, \ldots, W_K]$
  - Reduced order: $r = K \cdot r'$
  - Affine parameter dependency required

### Interpolation of transfer functions

- [Baur ’09]
  - Local reduction using BT
  - Error bounds and stability
  - Reduced order: $r = K \cdot r'$

### Interpolation of subspaces

- [Amsallem ’08]
  - Interpolation of the reduction bases
  - Reduced order: $r = r'$

### Interpolation of reduced matrices

- [Eid ’09, Panzer ’10, Amsallem ’11]
  - No explicit or affine parameter dependency required
  - Reduced order: $r = r'$
Concatenation of bases

(Weighted) concatenation of bases

• (Weighted) concatenation of local bases: \( \tilde{V}(\mathbf{p}^{\text{int}}) = \left[ w_1(\mathbf{p}^{\text{int}}) \mathbf{V}(\mathbf{p}_1), \ldots, w_K(\mathbf{p}^{\text{int}}) \mathbf{V}(\mathbf{p}_K) \right] \in \mathbb{R}^{n \times K \cdot r} \)

• Singular Value Decomposition: \( \tilde{V}(\mathbf{p}^{\text{int}}) \overset{\text{SVD}}{=} \tilde{U}(\mathbf{p}^{\text{int}}) \tilde{\Sigma}(\mathbf{p}^{\text{int}}) \tilde{T}(\mathbf{p}^{\text{int}})^T \)

• Reduction basis for new query point: \( \mathbf{V}(\mathbf{p}^{\text{int}}) = [\tilde{u}_1(\mathbf{p}^{\text{int}}), \ldots, \tilde{u}_r(\mathbf{p}^{\text{int}})] \in \mathbb{R}^{n \times r} \)

(Weighted) concatenation of snapshots

Concatenation of snapshots is suitable, when the local bases \( \mathbf{V}(\mathbf{p}_i) = \mathbf{U}(\mathbf{p}_i)(:, 1 : r), \quad i = 1, \ldots, K \) are calculated via an SVD-based technique like POD: \( \mathbf{X}(\mathbf{p}_i) = \mathbf{U}(\mathbf{p}_i) \mathbf{\Sigma}(\mathbf{p}_i) \mathbf{T}(\mathbf{p}_i)^T. \)

• (Weighted) concatenation of snapshots: \( \tilde{\mathbf{X}}(\mathbf{p}^{\text{int}}) = \left[ w_1(\mathbf{p}^{\text{int}}) \mathbf{X}(\mathbf{p}_1), \ldots, w_K(\mathbf{p}^{\text{int}}) \mathbf{X}(\mathbf{p}_K) \right] \in \mathbb{R}^{n \times K \cdot n_s} \)

• Singular Value Decomposition: \( \tilde{\mathbf{X}}(\mathbf{p}^{\text{int}}) \overset{\text{SVD}}{=} \tilde{\mathbf{U}}_{\mathbf{X}}(\mathbf{p}^{\text{int}}) \tilde{\mathbf{\Sigma}}_{\mathbf{X}}(\mathbf{p}^{\text{int}}) \tilde{\mathbf{T}}_{\mathbf{X}}(\mathbf{p}^{\text{int}})^T \)

• Reduction basis for new query point: \( \mathbf{V}(\mathbf{p}^{\text{int}}) = \tilde{\mathbf{U}}_{\mathbf{X}}(\mathbf{p}^{\text{int}})(:, 1 : r) \in \mathbb{R}^{n \times r} \)
Interpolation of subspaces

Interpolation of local bases is suitable for linear and nonlinear MOR!

**Starting point:** Local bases \( \{ V(p_i) \}_{i=1}^{K} \) at parameter sample points \( p_1, p_2, \ldots, p_K \).

**Procedure:**

1. Choose a local basis \( V_{i_0} \) for the reference subspace \( V_{i_0} \).
2. Mapping of all subspaces \( V(p_i) \) onto the tangent space \( T_{V_{i_0}} \) using the logarithmic map:
   \[
   (I - V_{i_0}V_{i_0}^T)V(p_i)(V_{i_0}^TV(p_i))^{-1} = U(p_i)\Sigma(p_i)T(p_i)^T,
   \]
   \[
   \Gamma(p_i) = U(p_i)\arctan(\Sigma(p_i))T(p_i)^T
   \]
3. Interpolation in the tangent space:
   \[
   \Gamma(p^{\text{int}}) = \sum_{i=1}^{K} w_i(p^{\text{int}}) \Gamma(p_i)
   \]
4. Backmapping of the interpolated subspace onto the manifold using the exponential map:
   \[
   \Gamma(p^{\text{int}})^{\text{SVD}} = U(p^{\text{int}})\Sigma(p^{\text{int}})T(p^{\text{int}})^T,
   \]
   \[
   V(p^{\text{int}}) = V_{i_0}T(p^{\text{int}}) \cos(\Sigma(p^{\text{int}})) + U(p^{\text{int}})\sin(\Sigma(p^{\text{int}}))
   \]
Interpolation of reduced matrices: pMOR by Matrix Interpolation
pMOR by Matrix Interpolation – Main Idea

Model at $p_1$

$V_1, W_1$

Reduced model at $p_1$

Model at $p_{\text{int}}$

$V_{p_{\text{int}}}, W_{p_{\text{int}}}$

Reduced model at $p_{\text{int}}$

Model at $p_2$

$V_2, W_2$

Reduced model at $p_2$

Interpolation of reduced matrices

Reduced model at $p_{\text{int}}$ from reduced models at sample points $p_1$ & $p_2$
pMOR by Matrix Interpolation – Procedure

[Panzer et al. ’10]

1.) Individual reduction

\[ E_{r,i} \dot{x}_{r,i}(t) = A_{r,i} x_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} x_{r,i}(t) \]

\[ E_{r,i} = W_i^T E_i V_i \]
\[ A_{r,i} = W_i^T A_i V_i \]
\[ B_{r,i} = W_i^T B_i \]
\[ C_{r,i} = C_i V_i \]

\[ V_i := V(p_i) \]
\[ W_i := W(p_i) \]
pMOR by Matrix Interpolation – Procedure

1.) Individual reduction

\[ E_{r,i} \ddot{x}_{r,i}(t) = A_{r,i} x_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} x_{r,i}(t) \]

\[ E_{r,i} = W_i^T E_i V_i, \quad A_{r,i} = W_i^T A_i V_i \]
\[ B_{r,i} = W_i^T B_i, \quad C_{r,i} = C_i V_i \]

2.) Transformation to generalized coordinates

\[ E_{r,i} T_i \ddot{x}_{r,i}(t) = A_{r,i} T_i \ddot{x}_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} T_i \ddot{x}_{r,i}(t) \]

\[ T_i = (R_V^T V_i)^{-1} \]

How do we choose \( T_i \)?

**Goal:** Adjustment of the local bases \( V_i \) to \( \hat{V}_i = V_i T_i \), in order to make the gen. coordinates \( \hat{x}_{r,i} \) compatible w.r.t. a reference subspace \( R_V \).

High correlation \( \hat{V}_i \leftrightarrow R_V \):
\[ T_i^T V_i R_V = I \]
pMOR by Matrix Interpolation – Procedure

1.) Individual reduction
\[ E_{r,i} \dot{x}_{r,i}(t) = A_{r,i} x_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} x_{r,i}(t) \]

2.) Transformation to generalized coordinates
\[ \ddot{E}_{r,i} T_{i} \dot{x}_{r,i}(t) = M_{i} A_{r,i} T_{i} \dot{x}_{r,i}(t) + M_{i} B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} T_{i} \dot{x}_{r,i}(t) \]

How do we choose \( M_{i} \)?

Goal: Adjustment of the local bases \( W_{i} \) to \( \hat{W}_{i} = W_{i} M_{i} \), in order to describe the local reduced models w.r.t. the same reference basis \( R_{W} \).

\[ M_{i} = (R_{V}^{T} V_{i})^{-1} \]

\[ T_{i} = (R_{W}^{T} W_{i})^{-1} \]

Analogous to \( R_{V} \) or \( R_{W} = R_{V} := R \)

High correlation \( \hat{W}_{i} \leftrightarrow R_{W} : M_{i}^{T} W_{i}^{T} R_{W} = I \)
pMOR by Matrix Interpolation – Procedure

1.) Individual reduction

\[ E_{r,i} \dot{x}_{r,i}(t) = A_{r,i} x_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} x_{r,i}(t) \]

\[ \begin{align*}
E_{r,i} &= W_i^T E_i V_i, \\
A_{r,i} &= W_i^T A_i V_i \\
B_{r,i} &= W_i^T B_i, \\
C_{r,i} &= C_i V_i
\end{align*} \]

2.) Transformation to generalized coordinates

\[ \begin{align*}
\dot{\hat{E}}_{r,i} &= M_i^T E_{r,i} T_i \ddot{x}_{r,i}(t) \\
\dot{\hat{A}}_{r,i} &= M_i^T A_i T_i \dot{x}_{r,i}(t) + M_i^T B_i u(t) \\
\hat{\dot{B}}_{r,i} &= M_i^T C_i T_i \dot{x}_{r,i}(t) \\
y_{r,i}(t) &= C_{r,i} T_i \dot{x}_{r,i}(t)
\end{align*} \]

\[ \begin{align*}
T_i &= (R_V^T V_i)^{-1} \\
M_i &= (R_W^T W_i)^{-1} \\
R_W &= R_V := R
\end{align*} \]

\[ \begin{align*}
V_{all} &= [V_1, \ldots, V_K] \\
V_{all} &= USN^T \\
R_V &= U(:, 1:r)
\end{align*} \]
pMOR by Matrix Interpolation – Procedure

\[ E_{r,1} \dot{x}_{r,1} = \dot{A}_{r,1} x_{r,1} + \dot{B}_{r,1} u \]
\[ y_{r,1} = \dot{C}_{r,1} x_{r,1} \]

\[ E_{r,2} \dot{x}_{r,2} = \dot{A}_{r,2} x_{r,2} + \dot{B}_{r,2} u \]
\[ y_{r,2} = \dot{C}_{r,2} x_{r,2} \]

1.) Individual reduction

\[ E_{r,i} \dot{x}_{r,i}(t) = A_{r,i} x_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} x_{r,i}(t) \]

\[ E_{r,i} = W_i^T E_i V_i \]
\[ A_{r,i} = W_i^T A_i V_i \]
\[ B_{r,i} = W_i^T B_i \]
\[ C_{r,i} = C_i V_i \]

\[ p_i, \; i = 1, \ldots, K \]
\[ V_i := V(p_i) \]
\[ W_i := W(p_i) \]

2.) Transformation to generalized coordinates

\[ \dot{E}_{r,i} \]
\[ M_i^T E_{r,i} T_i \dot{x}_{r,i}(t) = M_i^T A_{r,i} T_i x_{r,i}(t) + M_i^T B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} T_i x_{r,i}(t) \]

\[ T_i = (R_{V_i} V_i)^{-1} \]
\[ M_i = (R_{W_i} W_i)^{-1} \]
\[ R_W = R_V := R \]

\[ V_{all} = [V_1, \ldots, V_K] \]
\[ V_{all} \xrightarrow{SVD} USN^T \]
\[ R_V = U(:, 1 : r) \]
pMOR by Matrix Interpolation – Procedure

1.) Individual reduction
\[ E_{r,i} \dot{x}_{r,i}(t) = A_{r,i} x_{r,i}(t) + B_{r,i} u(t) \]
\[ y_{r,i}(t) = C_{r,i} x_{r,i}(t) \]

2.) Transformation to generalized coordinates
\[ \hat{E}_{r,i} = M_i^T E_{r,i} T_i \hat{x}_{r,i}(t) = M_i^T A_{r,i} T_i \dot{x}_{r,i}(t) + M_i^T B_{r,i} u(t) \]
\[ y_{r,i}(t) = \hat{C}_{r,i} T_i \dot{x}_{r,i}(t) \]

3.) Interpolation
\[ \hat{E}_r(p_{\text{int}}) = \sum_{i=1}^{K} \omega_i(p_{\text{int}}) \hat{E}_{r,i}, \quad \hat{A}_r(p_{\text{int}}) = \sum_{i=1}^{K} \omega_i(p_{\text{int}}) \hat{A}_{r,i} \]
\[ \hat{B}_r(p_{\text{int}}) = \sum_{i=1}^{K} \omega_i(p_{\text{int}}) \hat{B}_{r,i}, \quad \hat{C}_r(p_{\text{int}}) = \sum_{i=1}^{K} \omega_i(p_{\text{int}}) \hat{C}_{r,i} \]

\[ \sum_{i=1}^{K} \omega_i(p_{\text{int}}) = 1 \]

\[ \mathbf{p}_i, \quad i = 1, \ldots, K \]
\[ V_i := V(p_i) \]
\[ W_i := W(p_i) \]

\[ \mathbf{x}_{r,i} = T_i \hat{x}_{r,i} \]

\[ \mathbf{T}_i = (R_V V_i)^{-1} \]
\[ \mathbf{M}_i = (R_W W_i)^{-1} \]
\[ \mathbf{R}_W = \mathbf{R}_V := \mathbf{R} \]
\[ \mathbf{V}_{all} = [V_1, \ldots, V_K] \]
\[ \mathbf{V}_{all} = \text{SVD}(\mathbf{USN}^T) \]
\[ \mathbf{R}_V = \mathbf{U}(:, 1 : r) \]
Offline/Online decomposition

**Offline phase:**
1. Choose appropriate sample points $p_i, \ i = 1, \ldots, K$ in the parameter space
2. Build local models at the parameter sample points
3. Reduce the local models separately with desired MOR technique (e.g. modalMOR, BT, rational Krylov, IRKA, ...)
4. Compute $R$, all transformation matrices $T_i, M_i$ and transform the local reduced models to generalized coordinates (step 4. in online phase, if weighted SVD)

**Online phase:**
1. Calculate the weights $\omega_i(p^{\text{int}}), \ i = 1, \ldots, K$ depending on the actual parameter value $p^{\text{int}}$ and the chosen interpolation method (linear, spline,...)
2. Interpolate between the reduced system matrices
Sampling of the parameter space

Interpolation method (weighting functions)
Choice of parameter sample points is a critical question, specially in high-dimensional spaces!

**Small number of parameters (d < 3)**
- Full grid-based sampling or Latin hypercube sampling: Structured/uniform sampling, random sampling, logarithmic sampling
- Moderate/high number of samples generated that covers the parameter space

**Moderate number of parameters (3 ≤ d ≤ 10)**
- In this case, full grid sampling quickly becomes expensive (curse of dimensionality)
- Latin hypercube sampling remains tractable
- Non-uniform sampling, sparse grid sampling

**Large number of parameters (d > 10)**
- Difficult to balance: number of sample points vs. coverage of the parameter space
- Problem-aware, adaptive sampling schemes required!
- Adaptive greedy search, sensitivity analysis using Taylor series, subspace angles, etc…
Adaptive Sampling

Requirements:
- Parametric space should be adequately sampled
- Avoid undersampling and oversampling
- More parameter samples should be placed in highly sensitive zones

Quantification of parametric sensitivity:
- System-theoretic measure that quantifies the parametric sensitivity is needed in order to guide the adaptive refinement
- Adaptive sampling using angle between subspaces

Uniform Sampling:
- \( p_1 \ p_2 \ p_3 \ p_4 \ p_{K-1} \ p_K \)
- \( V_1 \ V_2 \ V_3 \ V_4 \ V_{K-1} \ V_K \)

Adaptive Sampling:
- \( p_1 \ p_2 \ p_3 \ p_4 \ p_{K-1} \ p_K \)
- \( V_1 \ V_2 \ V_3 \ V_4 \ V_{K-1} \ V_K \)
Adaptive Sampling via subspace angles

Concept of subspace angles:

- \( \mathbf{V}_1 \) and \( \mathbf{V}_2 \) are orthonormal bases for the subspaces \( \mathcal{V}_1 \) and \( \mathcal{V}_2 \)
- The largest angle between the subspaces can be determined by
  \[ \theta_{12} = \arcsin \left( \sqrt{1 - \sigma_r^2} \right) = \arccos(\sigma_r) \]
  \( \sigma_r \): smallest singular value of \( \mathbf{V}_1^T \mathbf{V}_2 \)

Usage for adaptive grid refinement:
- The larger the subspace angle, the more different are the projection matrices, and thus:
  - the higher the parametric sensitivity
  - and the more sample points can be introduced in the respective sub-span
## Automatic Adaptive Sampling: Pseudo-Code

[Cruz et al. '17]

1) Input $\theta_{\text{max}}$

2) Divide the entire parameter range into a uniform grid, calling it $p_1, p_2, \cdots, p_K$

3) **While** all $l_{i,i+1} > 1$ **do**
   
   a) Calculate the projection matrices $V_1, V_2, \cdots, V_K$
      corresponding to each of these values $p_1, p_2, \cdots, p_K$
   
   b) Compute subspace angles $\theta_{12}, \theta_{23}, \cdots, \theta_{K-1,K}$
      between these $V_i$'s, each taken pairwise
   
   c) Calculate
      
      $$
      l_{12} = \left[\frac{\theta_{12}}{\theta_{\text{max}}}\right],
      l_{23} = \left[\frac{\theta_{23}}{\theta_{\text{max}}}\right], \cdots,
      l_{K-1,K} = \left[\frac{\theta_{K-1,K}}{\theta_{\text{max}}}\right]
      $$
   
   d) Divide the interval between $p_1$ and $p_2$ into $l_{12}$ further intervals. Likewise, do the same for all the other intervals.
   
   e) Obtain new grid points $p_1, p_2, \cdots, p_N$, whereas $N > K$

   **End While**

---

**Local reduction at sample points** possible using any preferred MOR technique

\[
\text{theta}(i) = \text{subspace}(V_{p\{i\}}, V_{p\{i+1\}})
\]

**Quantitative indicator** of how many pieces each parameter interval is to be further broken

**Stopping criterion:**
1. All ratios are equal to 1
2. Specified maximum number of samples points reached

**Next iteration:** local reduction, etc. only at points that got added in the last while-loop iteration (efficient!)
Interpolation method – Weighting functions

For the weighting or the interpolation, appropriate weighting functions should be selected!

Basically, any multivariate interpolation method could be used for this purpose:

- Polynomial interpolation (Lagrange polynomials)
- Piecewise linear interpolation
- Piecewise polynomial interpolation (e.g. bi-/trilinear, cubic (splines), …)
- Radial basis functions (RBF)
- Kriging interpolation (Gaussian regression)
- Inverse distance weighting (IDW) based on nearest-neighbor interpolation
- Sparse grid interpolation

[Benner et al. ‘15]
pMOR Software
ssMOR Toolbox – Analysis and Reduction of Parametric Models in

✔ Definition of **parametric sparse state-space models**

```
psys = psss(func,userData,names)
p.sys = loadFemBeam3D(Opts)
p.sys = loadAnemometer3parameter
```

✔ **Manipulation of psss-class objects**

```
p.sys = fixParameter(psys,2,1.7)
p.sys = unfixParameter(psys,3)
```

✔ **Compatible** with the sss & sssMOR toolboxes

```
param = [p1, p2, p3, p4]
sys = psys(param)
```

✔ Different **parametric reduction methods** available (offline- & online-phase)

```
psysr = matrInterpOffline(psys, param, r, Opts);
```

```
psysr = globalPmorOffline(psys, param, r, Opts)
```

```
sysr = psysr(pQuery)
```

✔ **localReduction & adaptiveSampling as core functions**

```
[sysrp,Vp,Wp] = localReduction(psys, param, r, Opts)
```

```
paramRef = adaptiveSampling(psys, param, r)
```

www.rt.mw.tum.de/?morlab

www.rt.mw.tum.de/?psssmor
Numerical Examples

pMOR in Applications
Numerical example: Beam model

Parameter: Length $L$
Thickness and width: 10 mm
Young Modulus: $2.10^5$ Pa.
Damping: Proportional/Rayleigh

Order of the original system: 720
Order of the reduced system: 5

4 local models; Weights: Lagrange interp.
$s_0$: ICOP (Eid2009);
Numerical example: Solar panel model

Parameter: Thickness $t$ of the panel (varies between 0.25 and 0.5 mm)

Order of the original system: 5892
Order of the reduced system: 60

2 local models; Weights: Linear interpolation
Numerical example: Timoshenko Beam

- Finite element 3D model of a Timoshenko beam
- Parameter is the length of the beam: \( p \equiv L \)
- One-sided Krylov reduction with shifts at \( s_0 = 0 \)
- \( \theta_{\text{max}} = 10° \) chosen

### Table 1: Sample points \( p_i \), subspace angles \( \theta_{i,i+1} \) and ratios \( l_{i,i+1} \)

<table>
<thead>
<tr>
<th>Iter</th>
<th>( p_i [\text{m}] )</th>
<th>0.5</th>
<th>1.5</th>
<th>2.5</th>
<th>3.5</th>
<th>4.5</th>
<th>5.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \theta_{i,i+1} [^\circ] )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( l_{i,i+1} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( p_i [\text{m}] )</td>
<td>0.5</td>
<td>0.833</td>
<td>1.167</td>
<td>1.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \theta_{i,i+1} [^\circ] )</td>
<td>10.15</td>
<td>8.59</td>
<td>7.05</td>
<td>8.18</td>
<td>6.03</td>
<td>8.61</td>
</tr>
<tr>
<td></td>
<td>( l_{i,i+1} )</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>( p_i [\text{m}] )</td>
<td>0.5</td>
<td>0.667</td>
<td>0.833</td>
<td>1.167</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>( \theta_{i,i+1} [^\circ] )</td>
<td>5.26</td>
<td>4.89</td>
<td>8.59</td>
<td>7.05</td>
<td>8.18</td>
<td>6.03</td>
</tr>
<tr>
<td></td>
<td>( l_{i,i+1} )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Initial uniform grid with \( K = 6 \)

### Numerical example: Timoshenko Beam

[Cruz et al. '17]

<table>
<thead>
<tr>
<th>Iteration</th>
<th>( p_i [m] )</th>
<th>0.5</th>
<th>1.5</th>
<th>2.5</th>
<th>3.5</th>
<th>4.5</th>
<th>5.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter 1</td>
<td>( \theta_{i,i+1} [^\circ] )</td>
<td>25.79</td>
<td>14.20</td>
<td>8.61</td>
<td>7.06</td>
<td>5.73</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( l_{i,i+1} )</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration</th>
<th>( p_i [m] )</th>
<th>0.5</th>
<th>0.833</th>
<th>1.167</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3.5</th>
<th>4.5</th>
<th>5.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter 2</td>
<td>( \theta_{i,i+1} [^\circ] )</td>
<td>10.15</td>
<td>8.59</td>
<td>7.05</td>
<td>8.18</td>
<td>6.03</td>
<td>8.61</td>
<td>7.06</td>
<td>5.73</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( l_{i,i+1} )</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration</th>
<th>( p_i [m] )</th>
<th>0.5</th>
<th>0.667</th>
<th>0.833</th>
<th>1.167</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3.5</th>
<th>4.5</th>
<th>5.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter 3</td>
<td>( \theta_{i,i+1} [^\circ] )</td>
<td>5.26</td>
<td>4.89</td>
<td>8.59</td>
<td>7.05</td>
<td>8.18</td>
<td>6.03</td>
<td>8.61</td>
<td>7.06</td>
<td>5.73</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( l_{i,i+1} )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Interpolation point \( p^{\text{int}} = 1.0 \) between ROM 3 & ROM 4

Final refined grid with \( N = 10 \)
Numerical results – Direct vs. Interpolated ROM

With MatrInterp: no need to reduce the model for every new parameter value
Numerical results – Initial vs. Final Grid

ROMs calculated with the final grid yield better approximations
Numerical results – Initial vs. Final Grid

- Quantitative evaluation of the approximation
- Relative H2 error for \( nP = 100 \) different query points \( p^{\text{int}} \)
- Errors particularly small in the proximity of the sample points
- Final grid yields smaller errors for smaller beam lengths due to the adaptive refinement in this region

Relative H2 error between FOMs and interpolated ROMs for different parameter values and grids: FOM size \( n = 240 \), ROM size \( r = 17 \)
pMOR in Applications

Off-line applications:
• Efficient numerical simulation – “solves in seconds vs. hours”
• Design optimization – analysis for different parameters and “what if” scenarios
• Computer-aided failure mode and effects analysis (FMEA) – validation

On-line applications:
• Parameter estimation, Uncertainty Quantification
• Inverse problems, Real-time optimization
• Digital Twin, Predictive Maintenance

Physical domains:
mechanical, electrical, thermal, fluid, acoustics, electromagnetism, …

Application areas:
CSD, CFD, FSI, EMBS, MEMS, crash simulation, vibroacoustics, civil & geo, biomedical, …
pMOR in Applications – Some success stories


Summary & Outlook

References
Summary & Outlook

Takehome Messages:

• Large, parametric FEM/FVM models (linear/nonlinear) arise in many technical applications!
• Parametric MOR (pMOR) is indispensable to reduce the computational effort!
• Global and local pMOR approaches exist: e.g. concatenation of bases, interpolation of bases and matrix interpolation
• Offline/online decomposition of the methods
• Efficient sampling of the parameter space is crucial, especially for many parameters (d>10)
• Different interpolation methods and weighting functions available (linear, splines, RBF, …)
• pROMs can be applied for an efficient design optimization, inverse analysis, uncertainty quantification, etc.

Challenges / Outlook:

• High-dimensional parameter spaces:
  ➢ Adaptive sampling schemes
  ➢ Avoiding the curse of dimensionality (tensor techniques!?)
• pMOR for systems with time-dependent parameters: p(t)MOR
References (I)


References (II)


Backup
**pMOR by Matrix Interpolation – Features**

**Properties:**
- Local pMOR approach
- Analytical expression of the parameter-dependency in general not available
- Model only available at certain parameter sample points

**Main idea:**
1. Individual reduction of each local model
2. Transformation of the local reduced models
3. Interpolation of the reduced matrices

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>• No analytically expressed parameter-dependency required</td>
<td>• Choice of degrees of freedom</td>
</tr>
<tr>
<td>• Any desired MOR technique applicable for the local reduction</td>
<td>– Parameter sample points</td>
</tr>
<tr>
<td>• Offline/Online decomposition</td>
<td>– Interpolation method</td>
</tr>
<tr>
<td>• Reduced order independent of the number of local models</td>
<td>• Stability preservation</td>
</tr>
<tr>
<td></td>
<td>• Error bounds</td>
</tr>
</tbody>
</table>
Types of weighting functions / Interpolations

Matrices of the local reduced-order models

Explicit weights

- Linear interpolation
- Nonlinear interpolation

Implicit interpolation

- Spline interpolation
- Hermite interpolation
- RBF interpolation

\[ A_r = \omega_1(p) A_{r,1}^* + \omega_2(p) A_{r,2}^* + \cdots \]
# pMOR by Matrix Interpolation

Evaluation of the method according to different criteria

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure preservation</td>
<td>🌻</td>
</tr>
<tr>
<td>Reduced order</td>
<td>🌻</td>
</tr>
<tr>
<td>Storage effort</td>
<td>🌻</td>
</tr>
<tr>
<td>Computational cost</td>
<td>🌻</td>
</tr>
<tr>
<td>Offline/Online decomposition</td>
<td>🌻</td>
</tr>
<tr>
<td>Stability preservation</td>
<td>😞</td>
</tr>
<tr>
<td>Error bounds</td>
<td>😞</td>
</tr>
</tbody>
</table>
Extensions for the Matrix Interpolation

**Vereinheitlichendes Framework**
[Geuss et al. ’13]

**Framework mit folgenden Schritten:**
1.) Wahl der Parameterstützstellen
2.) Reduktion der lokalen Modelle
3.) Anpassung der lokalen Basen
4.) Wahl der Interpolationsmannigfaltigkeit
5.) Wahl der Interpolationsmethode

**Interpolation zwischen Modellen verschiedener reduzierter Ordnung**
[Geuss et al. ’14b]

- Interpolation zwischen Modellen mit unterschiedlicher reduzierter Ordnung $r_i$ nicht möglich
- **Idee:** Basen $V_i, W_i$ auf dieselbe Größe $r_0$ bringen durch die Berechnung von $T_i, M_i$ mittels Pseudoinversen

**Stabilitätserhaltung**
[Geuss et al. ’14a]

- Interpolation (selbst stabiler) reduzierter Modelle garantiert i.A. keine Stabilität
- **Idee:** Stabile reduzierte Modelle auf dissipative Form bringen, damit ein stabiles interpoliertes System resultiert → Lösung von Lyapunov-Gleichungen

**Black-Box Methode**
[Geuss et al. ’15]

- **Ziel:** Automatisierte pMOR-Methode
- **Idee:** Kreuzvalidierungsfehler für die iterative Ermittlung von Stützstellen und die optimale Wahl der Interpolationsmannigfaltigkeit und Interpolationsmethode verwenden