# Hyper-reduction Techniques for Efficient Simulation of Large-Scale Engineering Systems

Suparno Bhattacharyya, *[1], Jian Tao[1,2], Eduardo Gildin[3], and Jean C. Ragusa[4]

[1]Institute of Data Science, Texas A&M University, College Station, TX, USA
[2]College of Performance, Visualization & Fine Arts, Texas A&M University, College Station, TX, USA
[3]Department of Petroleum Engineering, Texas A&M University, College Station, TX, USA
[4]Department of Nuclear Engineering, Texas A&M University, College Station, TX, USA

## Contents



*corresponding author: suparno.bhattacharyya@gmail.com

**Abstract**

Reduced-order models (ROMs) offer compact representations of complex engineering systems governed by partial differential equations or high-dimensional ordinary differential equations, enabling efficient simulations of otherwise computationally intensive problems. These models are typically constructed by projecting the high-dimensional governing equations onto reduced subspaces derived using techniques such as Singular Value Decomposition (SVD) or Proper Orthogonal Decomposition (POD).

However, conventional ROMs struggle with nonlinear systems due to the high computational cost of repeatedly accessing high-dimensional solution spaces for nonlinear term evaluations. Hyper-reduction methods address this challenge by efficiently approximating nonlinear term evaluations, significantly improving ROM performance. They are also essential for solving large parametric linear problems that lack an efficient parameter-affine decomposition.

This paper provides a comprehensive overview of hyper-reduction algorithms, emphasizing both their theoretical foundations and practical implementations in academic research and industry. With the rapid advancement of data-driven methods, reduced-order modeling has become indispensable for analyzing and simulating large-scale

systems, including fluid dynamics, thermal processes, and structural mechanics. As the demand for efficient computational tools in science and engineering continues to grow, a detailed discussion of hyper-reduction techniques is both timely and valuable.

The paper explores state-of-the-art hyper-reduction techniques, including discrete empirical interpolation methods (DEIM), energy-conserving sampling and weighting (ECSW), and emerging machine learning-based approaches. A nonlinear parametric heat conduction example is presented to illustrate the implementation of these methods. The analysis evaluates their strengths and weaknesses using standard metrics, providing insights into their practical utility.

Finally, the paper concludes by discussing future research directions and potential applications of hyper-reduction, including its integration with real-time simulations and digital twin systems.

# 1 Introduction

Large-scale simulations [1, 2] are prevalent across numerous engineering fields [3] including computational fluid dynamics [4–7]; aerodynamic and structural simulations in the automotive and aviation industries; reservoir simulations in the petroleum sector [5, 8]; extreme-scale thermal, thermomechanical, and hydrodynamic simulations [9–12]; alternative energy applications such as nuclear engineering [13, 14], as well as microelectromechanical systems (MEMS) simulations [3, 15–21]. These systems, often governed by nonlinear multi-physics, multi-scale equations [22–24], are computationally expensive, particularly in many-query applications. Employing Finite element [25–28] and finite volume methods [29, 30] to solve such problems often lead to prohibitively large systems of equations, requiring days or weeks of computation [31]. Nevertheless, these systems often exhibit low intrinsic dimensionality [32, 33], meaning a small set of dominant features can effectively describe the solution. Data-driven techniques identify this low-dimensional *latent space* [34, 35], to construct reduced-order models (ROMs) [36–44], which substantially enhances computational efficiency while maintaining solution accuracy.

Proper orthogonal decomposition [39, 45–50] is widely utilized for constructing low-dimensional linear latent spaces that capture the dominant trends in the data by maximizing captured data-variance. This subspace is then used with Galerkin (or Petrov-Galerkin) projection [51–55] of the governing equations, resulting in a ROM, commonly referred to as projection-based ROMs (pROMs) [56–61].

This paper focuses on pROMs of large-scale nonlinear models [62, 63] derived from discretized nonlinear partial differential equations (PDEs) [64]. Contrary to expectations, pROMs for large-scale nonlinear systems can be computationally expensive to evaluate, and in some cases, costlier than the original high-fidelity model (HFM) they intend to reduce [31, 65]. Nonlinear ROMs typically employ iterative schemes such as Newton-Raphson methods [66] to solve the system, which requires repeated evaluation and projection of the full-order nonlinear terms. The evaluation of these nonlinearities requires access to full-order solutions, which are obtained by lifting the solution from the reduced latent space back to the full-order space, thus, effectively negating the intended computational efficiency

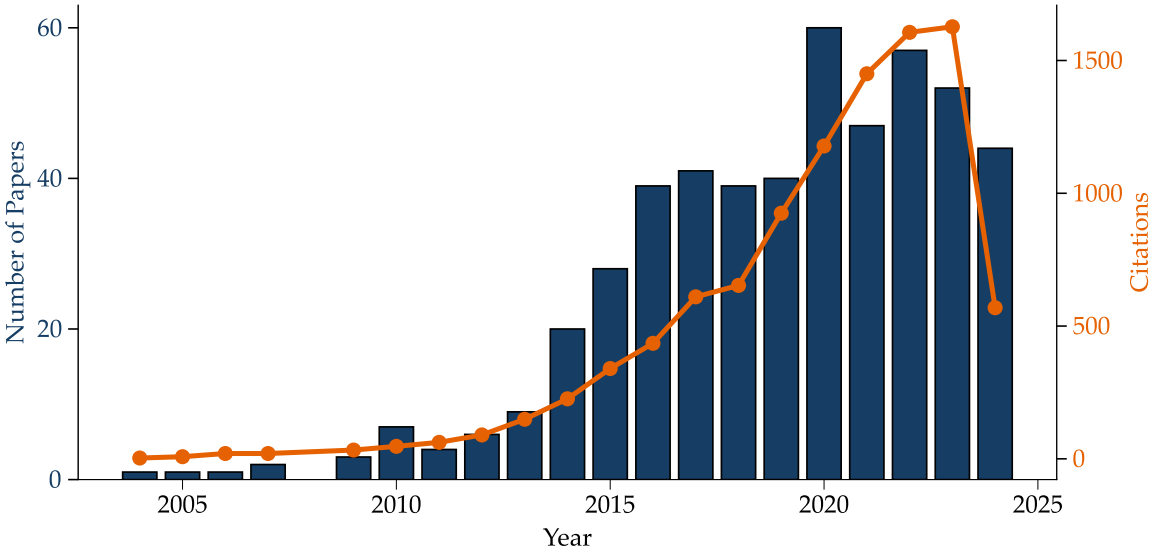



**Figure 1:** Published papers on hyper-reduction over the last two decades (as of 2024, *source: web of science*)

of the pROM. Therefore, evaluating model nonlinearities efficiently is critical to maintaining the computational efficiency of ROMs.

Hyper-reduction methods, a term coined by Ryckelynck [67], provide efficient strategies to handle nonlinear terms in reduced-order models by selectively *sampling* the computational mesh. Instead of evaluating nonlinearities across the entire dense mesh, these methods construct a *reduced mesh*—composed of strategically chosen nodes, cell centers, or grid points—to approximate nonlinear contributions. This is achieved through two primary approaches: (1) interpolation of nonlinear terms using empirical basis functions, as in empirical interpolation method (EIM) [68, 69] (ECSW), discrete EIM (DEIM) [70], or best points interpolation [71]; and (2) direct estimation of projected nonlinearities via learned quadrature rules, including the energy-conserving sampling and weighting method [31, 72–74], the empirical cubature method (ECM) [75–78], and the linear program-based empirical quadrature method (EQP) [79]. By focusing computations on this reduced subset, hyper-reduction methods drastically lower computational costs while maintaining accuracy, as demonstrated in this paper. Henceforth, we shall refer to pROMs that implement hyper-reduction strategies as Hyper-ROMs.

In addition to hyper-reduction, alternative methods address nonlinearities through exact reduction for polynomial forms [103], transformation of nonlinear systems into quadratic-bilinear forms via auxiliary variables [104–107], or nonlinear model reduction techniques that represent solutions on manifolds rather than linear subspaces [80, 108–120]. These include quadratic manifolds for structural dynamics [121, 122], neural network-based manifold approximation [123], and invariant manifolds for slow-fast dynamical systems [117, 124–134]. Additionally, trajectory piecewise linear (TPWL) techniques approximate nonlinear dynamics by linearizing the system along precomputed trajectories and combin-

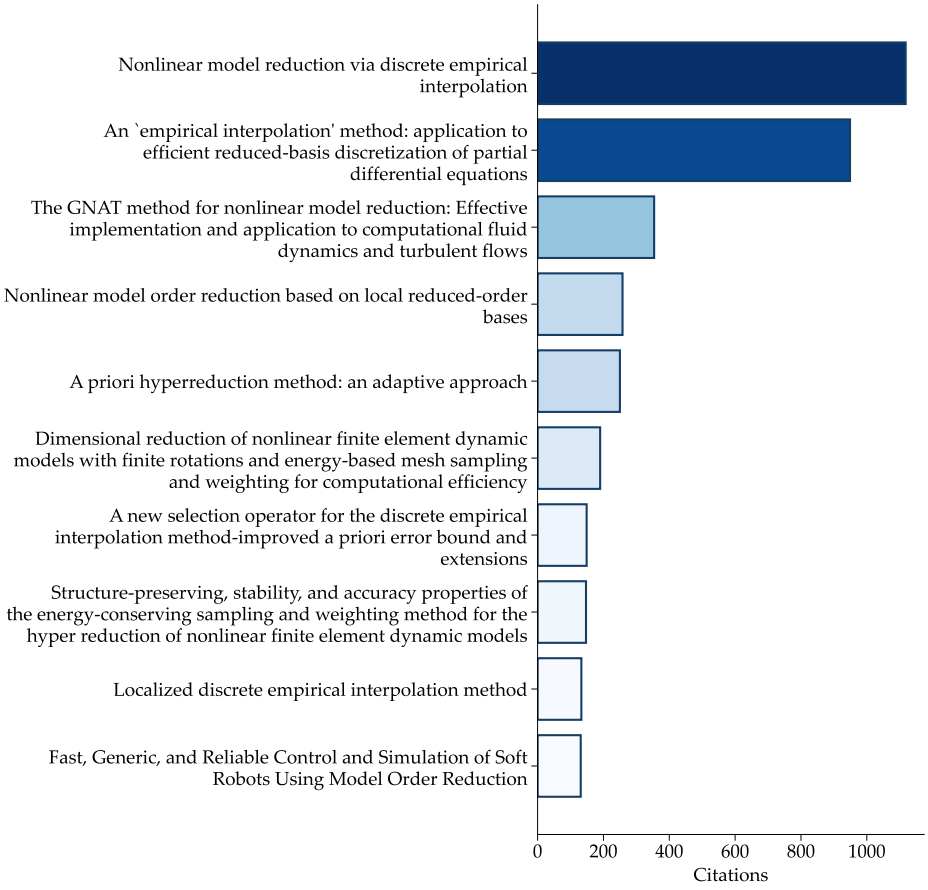



**Figure 2:** Top 10 highly cited papers on hyper-reduction (as of 2024, *source: web of science*) [31, 67, 69, 70, 72, 80–84]

ing these local linear models to reconstruct the system's behavior [135–139]. While these approaches demonstrate efficacy, hyper-reduction remains more broadly adopted across applications.

Over the past two decades, hyper-reduction techniques have gained significant traction in the scientific community, as illustrated by the increase in published papers on the topic as seen in Fig. 1. The number of publications reached its peak around 2021, while citations have continued to grow steadily, highlighting the sustained impact and active engagement of researchers in the field [140–143]. Among the most highly cited papers, methods such as (D)EIM and the GNAT [80, 144, 145] stand out as pivotal contributions (Fig. 2).

Hyper-ROMs find extensive application in domains such as subsurface simulations [135, 137, 146–151], nuclear engineering [152–156], mechanical and aerospace engineering [31, 55, 72, 157–162]. In particular, they are also capable of playing a critical role in the development and deployment of digital twin technology [163–166]. Digital twins [166, 167] serve as virtual counterparts to physical systems, enabling real-time monitoring [168], analysis, and decision making. These functionalities depend on rapid and precise simulations, which can

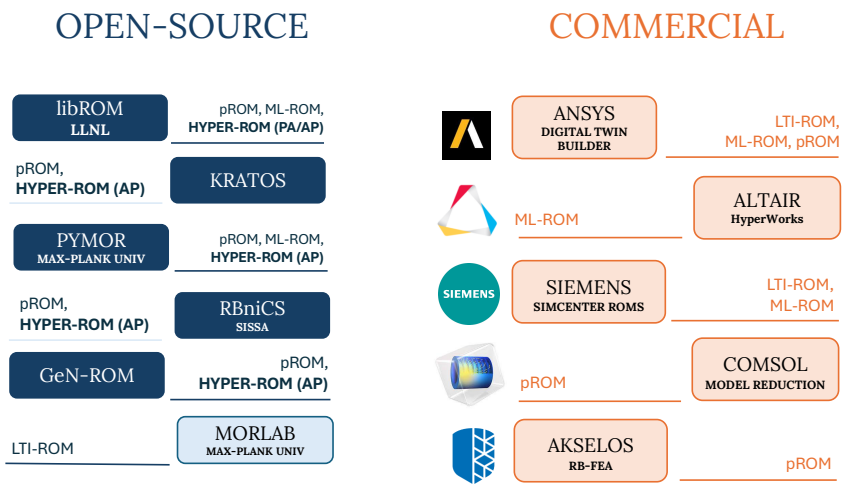



**Figure 3:** Comparison of open-source and commercial software tools for reduced-order modeling (ROM) [1]. Open-source tools include libROM [85], Kratos [86], PyMOR [87], RBniCS [88], and MORLAB [89]. These tools support ROM methods like projection-based Reduced Order Model (pROM), Machine Learning-based ROM (ML-ROM) [90, 91], Hyper-ROM (AP/PA), and Linear Time-Invariant ROM (LTI-ROM) [92, 93]. Dark blue boxes indicate support for Hyper-ROM, where (AP) denotes "Approximate-then-Project," and (PA) denotes "Project-then-Approximate" type hyper-reduction (discussed in Section 4). Commercial tools include ANSYS Digital Twin Builder [94, 95], Altair HyperWorks [96], Siemens Simcenter ROMS [97], COMSOL Model Reduction [98–100], and Akselos RB-FEA [101, 102], which support various ROM methods. However, none of these commercial tools support hyper-reduction.

be provided by such Hyper-ROMs, facilitating the prediction of system responses across various scenarios.

Given the growing interest, developing an understanding of hyper-reduction methods is becoming critical. Although existing works offer brief overviews [169] or broader discussions on data-driven surrogates [103, 170–172], dedicated articles on hyper-reduction remain limited [173]. Furthermore, the principles and algorithms of hyper-reduction are often deeply mathematical in nature, which presents a steep learning curve for researchers new to the field. In this review, our aim is to bridge this gap by adopting a more intuitive approach whenever possible.

The current state of software supporting hyper-reduction, as illustrated in Fig. 3, highlights a clear distinction between open-source and commercial platforms. Only open-source tools,

[1] list not exhaustive

such as libROM [85], PyMOR [87], and MORLAB [89], provide hyper-reduction capabilities (albeit limited), which serve as important resources for academic researchers. However, reverse engineering these tools to understand how the algorithm is implemented can be challenging due to complex code structures and dependencies, making it difficult to trace the execution flow and extract key methodological insights.

Commercial platforms such as ANSYS [174, 175], Siemens Simcenter ROMS [176], Altair HyperWorks [177], etc., are yet to adopt these methods. Although these platforms offer deep learning-based ROMs, they do not currently incorporate hyper-reduction techniques. This gap arises mainly from the intrusive nature of hyper-reduction, which requires substantial modifications to existing code and data infrastructures. Although open-source software can accommodate such changes, commercial tools prioritize broad applicability, making integration challenging without major disruptions to computational pipelines. Nevertheless, integrating hyper-reduction presents a significant opportunity for commercial software to enhance computational efficiency, particularly in large-scale simulation contexts.

To support researchers and improve accessibility, along with this detailed review, we present a GitHub repository [178] that includes an in-house finite element framework with integrated hyper-reduction techniques. Intended as a practical learning resource, the repository prioritizes interpretability by focusing on simpler nonlinear problems and offers easily adaptable code along with detailed instructions for reproducing all data and figures from the study.

The paper is organized as follows: in section 2, we present a detailed overview of projection-based ROMs, focusing on POD. In Section 3, we present a one-dimensional heat conduction problem as a representative case to illustrate model reduction both without and with various hyper-reduction techniques. We selected this problem for its conceptual clarity and ease of understanding, while recognizing that hyper-reduction methods have already been established as effective in large-scale, complex settings. Our choice of this problem facilitates a more tutorial-oriented demonstration. Finally, in Section 4, we discuss different hyper-reduction methods, including DEIM, S-OPT, Gauss Newton Approximation Tensor (GNAT), ECSW, and ECM, comparing them using the example problem. We briefly touch upon deep-learning-based hyper-reduction using neural networks. The review concludes with future directions and applications in various engineering disciplines. This tutorial review emphasizes quantitative understanding in a manner suitable for back-of-the-envelope calculations, with the aim of providing reusable insights.

## 2 Background and preliminaries

Projection-based reduced-order modeling (ROM) problems can generally be categorized into two main types [170]. The first category of problems deals with large-scale models formulated based on a system-theoretic approach. These models are characterized by a large set of Ordinary Differential Equations (ODEs) or Differential-Algebraic Equations (DAEs), e.g., linear-time-invariant (LTI) systems, where the dynamics is represented using

state variables. The state variables evolve with time, being driven by external excitation, and are partially captured at the output phase based on the quantities of interest. The ROMs preserve the original structure of these large-scale models and use a reduced state while closely matching the input-output relation of the actual system. Furthermore, the model reduction algorithms used for these problems are not strictly data-driven, but are often performed based on the characteristics of the full-order model without relying upon simulation data. In essence, these algorithms determine a reduced subspace for projecting the full-order models which results in eliminating the state variables that are poorly coupled with the input-forcing and barely contribute to the observed outputs. Examples of such systems include Balanced truncation [38, 179], moment matching methods [40, 57, 180], etc. While effective for large-scale *linear systems* (e.g., Linear Time Invariant –LTI), their applicability to multi-input multi-output nonlinear systems is limited.

The second category of problems centers on the accelerated numerical solution of Partial Differential Equations (PDEs), which are inherently infinite-dimensional. As depicted in Fig. 4, the numerical solution of PDEs typically involves a three-stage process. First, the infinite-dimensional domain is discretized into a finite-dimensional space. This discretization may take the form of nodal values (finite element method), point values (finite difference method), or cell averages (finite volume method), among others, depending on the chosen numerical method. To ensure sufficiently fidelity, the dimensionality of this finite representation is often chosen very large (e.g., using a fine spatial mesh), resulting in a transformation of the PDEs into a high-dimensional system of algebraic, differential, or differential-algebraic equations. Next, standard numerical techniques are applied to solve these equations, a process that is computationally demanding due to the large dimensionality. Finally, once a solution is obtained, it is mapped back from the finite-dimensional space encoded space to the original infinite-dimensional space using polynomial interpolation or Galerkin projections.

*Reduced-order models* simplify the computational burden of these high-fidelity, large-scale systems by compressing the finite-dimensional space into a much smaller latent space. The large-scale equations are then projected onto this latent space to formulate a reduced system. After solving these reduced equations, the solution is first interpolated back to the finite, high-dimensional space, and then further mapped to the infinite-dimensional space, as in the standard process described above. The latent space is derived directly from the high-fidelity data of the large-scale system and is typically represented by either a low-dimensional nonlinear manifold or a low-dimensional linear subspace. This paper focuses on the latter –linear subspace reduced order models (LS-ROMs)– while providing a brief complementary discussion on nonlinear manifold reduced order models (NM-ROM) towards the end.

The subspace for LS-ROMs is typically obtained by applying proper orthogonal decomposition (POD) or singular value decomposition (SVD) onto a small volume of the high-fidelity (training) data. POD hierarchically captures the dominant correlations within the data using orthonormal basis vectors, and thereby captures the few dominant low-rank characteristics of the high-fidelity *model* itself. The process of deriving a projection based LS-ROM is described in detail in the following subsections.

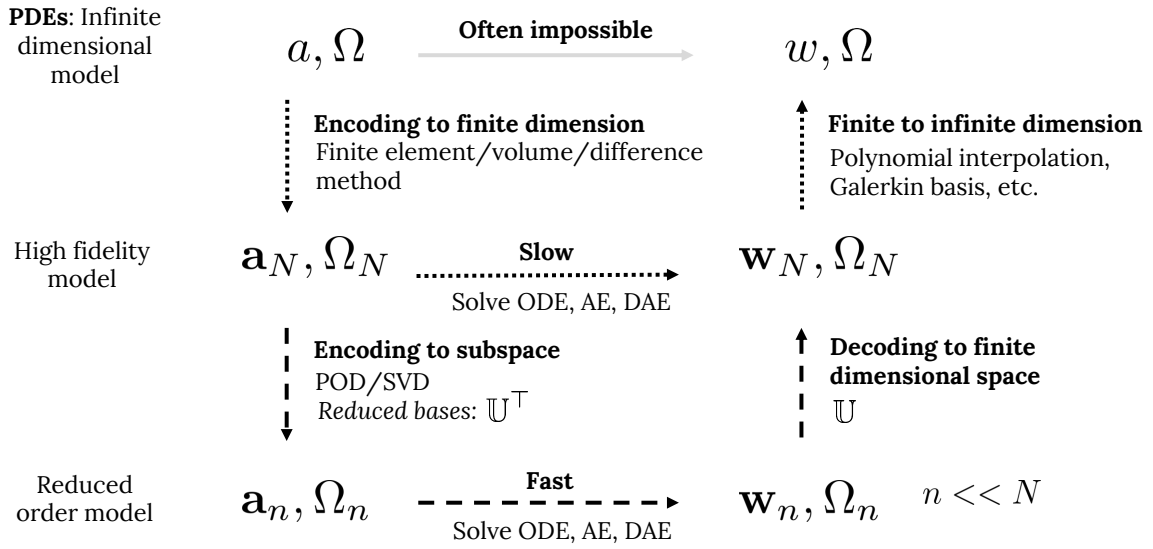



**Figure 4:** Numerical solution of PDEs. High-fidelity models and reduced order models. Here $a$ represents an input to the system; $\Omega$ is the problem domain in the infinite dimensional space, $w$ is the infinite-dimensional solution to the PDE; $\mathbf{a}_N$, $\mathbf{w}_N$ and $\Omega_N$ are the high-dimensional finite approximation, and $\mathbf{a}_n$, $\mathbf{w}_n$ and $\Omega_n$ are the reduced representation of $\mathbf{a}$, $\Omega$, and $\mathbf{w}$, respectively.

### 2.1 The high-fidelity model

Let us consider a parametric, time-dependent, semi-discrete (discretized in space only), $N$-dimensional ($N$ prohibitively large) high-fidelity model defined over domain $\Omega_N$, which is of the form [170]

$$\begin{cases} \mathsf{M}(\boldsymbol{\mu})\dot{\mathbf{w}}(t;\boldsymbol{\mu}) + \mathbf{f}(\mathbf{w}(t;\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbf{g}(t;\boldsymbol{\mu}), \\ \mathbf{w}(0;\boldsymbol{\mu}) = \mathbf{w}^0(\boldsymbol{\mu}), \end{cases} \tag{1}$$

where $t$ is time; $\boldsymbol{\mu}$ is a parameter vector in bounded space $P \subset \mathcal{R}^p$; $M(\boldsymbol{\mu}) \in \mathcal{R}^{N\times N}$ is the symmetric, positive definite mass matrix; $\mathbf{w}(t;\boldsymbol{\mu}) \in \mathcal{R}^N$ is the time-dependent state solution vector; $\mathbf{w}^0(\mathbf{x};\boldsymbol{\mu})$ is the initial condition; $\mathbf{f}(\mathbf{w}, t;\boldsymbol{\mu}) \in \mathcal{R}^N$ is the non-linear internal force vector; $\mathbf{g}(t;\boldsymbol{\mu}) \in \mathcal{R}^N$ is the time-dependent external force vector.

As shown in Fig. 4, these high-fidelity models are obtained from the governing partial differential equations by employing techniques like finite difference/element/volume methods. Constructing a projection-based reduced-order model (ROM) for Eq. (1) involves three stages: (a) generating a small dataset of high-fidelity training data by solving Eq. (1), (b) determining a reduced subspace from this data using a dimension reduction method like proper orthogonal decomposition (POD) [39], and (c) projecting the high-fidelity model onto the reduced subspace to obtain a ROM [57].

In the following two sections we discuss stages (b) and (c) assuming that a small volume

of high-fidelity training data has been generated by solving Eq. (1). We present a brief description of POD first followed by the derivation of the ROM.

### 2.2 Proper orthogonal decomposition (POD)

Proper Orthogonal Decomposition (POD), also known as Principal Component Analysis (PCA) or Karhunen-Loève Decomposition (KLD), reduces dimensionality of data by capturing dominant correlated patterns with minimal information loss [39, 45]. The goal is to find a reduced set of orthonormal basis functions that represent the dataset optimally [181].

More formally, POD seeks an optimal basis $U(x)$ by minimizing the error in the time-averaged projection of a spatio-temporal field $w(x,t)$:

$$\min_{U \in L^2(\Omega)} \left\langle \|w(x,t) - (w(x,t), U(x))U(x)\|^2 \right\rangle \tag{2}$$

subject to $\|U(x)\| = 1$, where $\|\cdot\|$ denotes the $L^2$ norm and angle brackets $\langle \cdot \rangle$ represent the time average over $T$, the time-window of interest. This can be reformulated as a maximization problem [39], leading to the eigenvalue problem:

$$\mathfrak{R}U = \lambda U, \quad \mathfrak{R}U = \langle (w, U)w \rangle \,, \tag{3}$$

where $\mathfrak{R}$ is the two-point correlation tensor [171] (or covariance operator when $w$ is mean-centered) defined as

$$\mathfrak{R}(x,y) = \frac{1}{T} \int_0^T w(x,t)w(y,t)\, dt \,. \tag{4}$$

The eigenfunctions $U$ are termed proper orthogonal modes (POMs), which capture the dominant features in $w$. The associated eigenvalues $\lambda$ quantify the variance explained by each mode. POMs are ranked in order of decreasing eigenvalue magnitude.

The dimension of the reduced subspace, spanned by $U$, is determined based on the cumulative variance captured. Let $k$ denote the number of leading POMs retained. The cumulative variance fraction associated with the first $k$ modes is given by

$$r_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i} \tag{5}$$

where $\lambda_i$ is the eigenvalues corresponding to the $U_i$. The value of $k$ is chosen as the smallest integer for which $r_k$ exceeds a prescribed threshold $\alpha$, typically close to 1; that is, $\alpha \leq r_k \lesssim 1$. The resulting reduced subspace dimension is denoted by $n$, with $n \ll N$.

The reconstruction error after projecting onto the $n$-dimensional subspace is given by [182]:

$$\epsilon_{\text{POD}} = \sqrt{\sum_{i=n+1}^{N} \lambda_i} \,. \tag{6}$$

In practical applications, POD is commonly computed on the discrete data snapshots. Assuming a spatial discretization $N$, and temporal discretization $N_t$, we define the snapshot matrix, $\mathsf{W} \in \mathsf{R}^{N \times N_t}$ as:

$$\mathsf{W} = \begin{bmatrix} \mathbf{w}_1 - \mathbf{w}^{ref} & \mathbf{w}_2 - \mathbf{w}^{ref} & \cdots & \mathbf{w}_{N_t} - \mathbf{w}^{ref} \end{bmatrix}, \tag{7}$$

where typically $\mathbf{w}^{ref} \in \mathcal{R}^N$ denotes the snapshot mean and columns of $\mathsf{W}$ are the mean-subtracted solution snapshots. Additionally, in some cases, the initial condition also serves as the reference. The covariance *matrix* $\mathsf{R}$ (the discrete version of $\mathfrak{R}$) is then defined as:

$$\mathsf{R} = \frac{1}{N_t} \mathsf{W}\mathsf{W}^\top, \tag{8}$$

which, similar to Eq. (3), leads to the matrix eigenvalue problem:

$$\mathsf{R}\mathbf{U} = \lambda \mathbf{U}. \tag{9}$$

We note that, up to this point, our discussion of POD has assumed that the data $\mathbf{w}$ is of spatiotemporal in nature, which, however, is not necessary. Discrete POD can also be applied in a purely parametric context, where the snapshot matrix $\mathsf{W}$ is constructed using $N_\mu$ snapshots corresponding to different *parameter values* rather than different time instances.

Solving the eigenvalue problem in Eq. (9) directly is computationally expensive for large $N$. To address this, the *method of snapshots* was introduced [183]. This method leverages the insight that each POM $\mathbf{U}$ can be expressed as a linear combination of the snapshot vectors $\mathbf{w}_k$, and any property of the ensemble $\mathbf{w}_k$ that is preserved under linear combination is inherited by $\mathbf{U}$. As a result, instead of Eq. (9), we can solve a smaller eigenvalue problem of size $N_t \times N_t$ and solve the reduced eigenvalue problem to calculate $\mathbf{U}$ in the following manner:

$$\frac{1}{N_t} \mathsf{W}^\top \mathsf{W}\mathbf{q} = \lambda \mathbf{q}, \quad \mathbb{U} = \mathsf{W}\mathbf{q}. \tag{10}$$

On the other hand, it can be shown that POD of the snapshot matrix $\mathsf{W}$ is closely related to its Singular Value Decomposition (SVD):

$$\mathsf{W} = \underline{\mathbb{U}} \Sigma \underline{\mathbb{V}}^\top, \tag{11}$$

where $\underline{\mathbb{U}} \in \mathcal{R}^{N \times N}$ and $\underline{\mathbb{V}} \in \mathcal{R}^{N_t \times N_t}$ contain orthonormal left and right singular vectors, respectively, with $\underline{\mathbb{U}}^\top \underline{\mathbb{U}} = \mathsf{I}_N$ and $\underline{\mathbb{V}}^\top \underline{\mathbb{V}} = \mathsf{I}_{N_t}$. The diagonal matrix $\Sigma \in \mathcal{R}^{N \times N_t}$ contains non-negative singular values $\sigma_i$ in descending order. Substituting (11) into (8) yields:

$$\mathsf{R} = \frac{1}{N_t} \underline{\mathbb{U}} \Sigma \Sigma^\top \underline{\mathbb{U}}^\top = \underline{\mathbb{U}} \Lambda \underline{\mathbb{U}}^\top, \tag{12}$$

where $\Lambda = \frac{1}{N_t} \Sigma \Sigma^\top$. Eq. (12) shows that the POMs of $\mathsf{W}$ are essentially the same as the left singular vectors of $\mathsf{W}$, with eigenvalues linked to squared singular values. The elements of $\Lambda$ satisfy:

$$\frac{\sigma_i^2}{N_t} = \lambda_i^{\text{POD}}. \tag{13}$$

Hence, SVD is applied directly to $\mathsf{W}$ for reduced subspace construction.

### 2.3 Reduced order model

The subspace for the ROM is spanned by a truncated set of basis vectors contained in a matrix $\widetilde{\mathbb{U}} \in \mathcal{R}^{N \times n}$:

$$\widetilde{\mathbb{U}} = \mathbb{U}[:, 1:n]\,, \tag{14}$$

where $n$ is the number of vectors retained after truncation. Then the solution of Eq. (1) is approximated as

$$\mathbf{w}_N(t;\boldsymbol{\mu}) \approx \widetilde{\mathbf{w}}_N(t;\boldsymbol{\mu}) = \mathbf{w}_N^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n(t;\boldsymbol{\mu})\,, \tag{15}$$

where $\mathbf{w}_n$ are the reduced coordinates of the basis vectors in $\widetilde{\mathbb{U}}$. Formally $\widetilde{\mathbb{U}}$ is called the right reduced order basis or right ROB (but these are also the *left* singular vectors of W). Substituting $\widetilde{\mathbf{w}}_N$ in Eq. (1), we obtain the following residual:

$$\mathbf{r}_N(\mathbf{w}_n;\boldsymbol{\mu}) = \mathsf{M}_N(\boldsymbol{\mu})\widetilde{\mathbb{U}}\dot{\mathbf{w}}_n(t;\boldsymbol{\mu}) + \mathbf{f}_N(\mathbf{w}_N^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n(t;\boldsymbol{\mu});\boldsymbol{\mu}) - \mathbf{g}_N(t;\boldsymbol{\mu})\,. \tag{16}$$

A direct solution to $\mathbf{r}_N = 0$ is not possible since the system is over-determinate ($N$ equations but $n$ unknowns). Typically, we constrain the residual by enforcing it to be orthogonal to a test subspace spanned by another set of basis vectors $\mathbb{W} \in \mathcal{R}^{N \times n}$, termed as the left reduced order basis or left ROB, such that

$$\mathbb{W}^\top \left[\mathsf{M}_N(\boldsymbol{\mu})\widetilde{\mathbb{U}}\dot{\mathbf{w}}_n + \mathbf{f}_N(\mathbf{w}_N^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n;\boldsymbol{\mu}) - \mathbf{g}_N\right] = 0\,. \tag{17}$$

This yields a $n$-dimensional solvable system of equations. Defining the reduced-order matrices and vectors as

$$\mathsf{M}_n(\boldsymbol{\mu}) = \mathbb{W}^\top \mathsf{M}_N(\boldsymbol{\mu})\widetilde{\mathbb{U}}, \tag{18}$$

$$\mathbf{f}_n(\mathbf{w};\boldsymbol{\mu}) = \mathbb{W}^\top \mathbf{f}_N(\mathbf{w}_N^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n;\boldsymbol{\mu}), \tag{19}$$

$$\mathbf{g}_n = \mathbb{W}^\top \mathbf{g}_N, \tag{20}$$

and assuming that $(\mathbb{W}^\top\widetilde{\mathbb{U}})$ is invertible, we write this descriptive form of the *Petrov-Galerkin* reduced-order model as [184]

$$\begin{cases} \mathsf{M}_n(\boldsymbol{\mu})\dot{\mathbf{w}}_n + \mathbf{f}_n(\mathbf{w}_n;\boldsymbol{\mu}) = \mathbf{g}_n, \\ \mathbf{w}_N(0,\boldsymbol{\mu}) = (\mathbb{W}^\top\widetilde{\mathbb{U}})^{-1}\mathbb{W}^\top(\mathbf{w}_N^0(\boldsymbol{\mu}) - \mathbf{w}_N^{ref})\,. \end{cases} \tag{21}$$

The $n-$dimensional ROM in Eq. (21) is solved at a much lower computational cost, and the approximate full order solution is reconstructed using Eq. (15). Note that, Eq. (21) can also be obtained using the more commonly employed *Galerkin Projection*, in which $\mathbb{W} = \widetilde{\mathbb{U}}$ and $(\mathbb{W}^\top\widetilde{\mathbb{U}}) = \mathsf{I}_n$.

### 2.4 POD-greedy-Galerkin reduced basis method

The reduced basis method (RBM) is a more recently developed and widely accepted approach to building a projection basis for ROMs in a parametric setting [68, 185–187], which optimally selects parameters from a given set to generate high-fidelity solutions used in deriving ROMs, aiming at low prediction errors throughout the parameter space. The method iteratively constructs the reduced-order basis $\widetilde{\mathbb{U}}$ using high-fidelity solutions corresponding to parameters iteratively selected using a greedy approach.

Initially, a parameter $\boldsymbol{\mu}_0$, from the given parameter space $P$ is selected (by random sampling or a priori criteria). Next, the corresponding high-fidelity solution snapshots, which can be obtained in multiple time steps for time-dependent problems or in a single instance for purely parametric cases. These snapshots are stored in the matrix $\mathsf{W} \in \mathcal{R}^{N \times N_t}$, where $N$, as before, corresponds to dimension of the HFM, and $N_t$ denotes the number of snapshots.

After this, a reduced-order basis $\widetilde{\mathbb{U}}$ is derived from $\mathsf{W}$ and a Galerkin ROM is built by projecting the HFM onto $\widetilde{\mathbb{U}}$. It is used to predict solutions for other parameter values within $P$. The accuracy of these predictions is assessed via an *a posteriori* error estimate, typically by computing the residual magnitude. For any new parameter $\boldsymbol{\mu} \in P$, the ROM solution $\mathbf{w}_n$ is "lifted" to the high-dimensional space (as defined in Eq. 15) and substituted into the HFM equations to compute the residual $\mathbf{r}_N(\mathbf{w}_n; \boldsymbol{\mu})$. The *a posteriori error estimate*, thus obtained quantifies the ROM's accuracy without requiring HFM re-solves.

Subsequently, the parameter $\boldsymbol{\mu}^*$ associated with the highest a posteriori error across the remaining parameters in $P$ is identified:

$$\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in P} \left\| \mathbf{r}_N(\mathbf{w}_n; \boldsymbol{\mu}) \right\|. \tag{22}$$

Once $\boldsymbol{\mu}^*$ is identified, the corresponding high-fidelity solution snapshots are computed and are stored in a data matrix $\mathsf{W}_{\mu^*} \in \mathcal{R}^{N \times N_t}$, which is then used to update the reduced basis $\widetilde{\mathbb{U}}$.

For updating $\widetilde{\mathbb{U}}$, the contribution of $\widetilde{\mathbb{U}}$ is first removed from $\mathsf{W}_{\mu^*}$ to avoid redundancy, producing the matrix $\overline{\mathsf{W}}$:

$$\overline{\mathsf{W}} = \mathsf{W}_{\mu^*} - \widetilde{\mathbb{U}}\widetilde{\mathbb{U}}^\top \mathsf{W}_{\mu^*}. \tag{23}$$

SVD is then applied to $\overline{\mathsf{W}}$ and the most dominant singular vector (the first column), denoted as $\widetilde{\mathbb{U}}_{\mu^*}$, is appended to $\widetilde{\mathbb{U}}$ to update it as $\widetilde{\mathbb{U}} \leftarrow [\widetilde{\mathbb{U}}, \widetilde{\mathbb{U}}_{\mu*}]$.

This iterative procedure continues until the a posteriori error estimate for all parameters falls below a specified tolerance. To further improve computational efficiency when dealing with large parameter spaces, RBM is often used with *Affine* decomposition (discussed in the next section), whenever possible.

The RBM algorithm for a steady-state purely parametric system follows a slightly different approach. In each iteration, the solution corresponding to $\boldsymbol{\mu}_*$ is directly appended to the existing set of basis vectors, and instead of using Singular Value Decomposition (SVD), a Gram-Schmidt orthonormalization is implemented to maintain orthonormality after each update. A clear discussion of both algorithmic variants can be found in [188].

Thus, the RBM offers a more systematic way to handle parametric dependencies in PDEs and provides a posteriori error bounds, unlike POD, which lacks inherent error estimates for new parameters. Furthermore, the greedy parameter sampling strategy employed in RBM can sometimes yield a smaller set of basis vectors (compared to a naive or global POD basis) for the same accuracy while providing an estimate of the quality of the reduced solutions.

## 3 Model reduction for 1D parametric heat conduction

In this section, we discuss the model reduction of a 1D linear and a 1D nonlinear heat conduction problem, characterized by parametric elliptic PDEs, and demonstrate the efficacy of projection-based reduced order modeling. While in Sec. 2, reduced order modeling was introduced in a more general spatio-temporal setting, the present discussion is restricted to purely parametric problems. Nevertheless, the methods employed in this and subsequent sections, as well as the insights and conclusions drawn, remain relevant for systems with temporal evolution.

We briefly describe the finite element formulation of the governing PDE to obtain the high-fidelity model, and subsequently derive the corresponding reduced order model following the steps delineated in the previous sections. Although our discussion here centers on a finite element model, as discussed earlier, any other numerical techniques, such as a finite difference or finite volume method can be employed as well [189].

The governing equation for steady state heat conduction over a 1D domain $\Omega \triangleq [0, 0.5]$ is given by

$$-\nabla \cdot (k \nabla T) = q \quad \text{in} \quad \Omega\,, \tag{24}$$

where $k$ ($\mathrm{W\,m^{-1}\,K^{-1}}$) is the thermal conductivity, and $q$ ($\mathrm{W\,m^{-1}}$) is the internal heat generation per unit length. These can either be linear functions of the parameter vectors $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$, (for $k$ and $q$, respectively), or nonlinear functions of both parameter vectors and the temperature field $T$ (K). While this section covers both cases—linear and nonlinear $k$ and $q$—the following formulations assume $k$ and $q$ are nonlinear. Equivalent formulations apply when $k$ and $q$ are linear.

Denoting $g$ as the heat flux on the boundary, the Neumann and Dirichlet boundary conditions for both linear and nonlinear problems are defined as

$$g|_{x=0} = 0\,\mathrm{W} \quad \text{and} \quad T|_{x=0.5} = 573.15\,\mathrm{K}\,. \tag{25}$$

To obtain a finite element model, we begin by deriving the weak form of Eq. (24) [26, 28]. This process requires defining the trial space $\mathcal{U}$ and the test space $\mathcal{V}$. The trial space $\mathcal{U}$ encompasses admissible temperature functions $T$ that comply with essential (Dirichlet) boundary conditions, while the test space $\mathcal{V}$ includes test functions $v$ that vanish on the Dirichlet boundaries. The weak form is derived by multiplying the governing PDE by a test function $v \in \mathcal{V}$ and integrating over the domain $\Omega$. Applying vector calculus identity

along with the Divergence theorem, we obtain

$$\int_\Omega k(T;\boldsymbol{\mu})\nabla T\cdot\nabla v\,d\Omega - \cancelto{0}{\int_{\partial\Omega} k(T;\boldsymbol{\mu})\nabla T\cdot\mathbf{n}\,v\,\mathrm{d}\Gamma} = \int_\Omega q(T;\boldsymbol{\beta})v\,d\Omega\,. \tag{26}$$

The boundary term cancels out because of the zero-flux boundary condition at $x = 0$ and Dirichlet boundary condition at $x = 0.5$.

We write the approximate solution of Eq. (26) $\widetilde{T}(x) \in \mathcal{U}$, in terms of finite element basis functions also known as the shape function as

$$\widetilde{T}(x) \approx \sum_{j\in\mathcal{B}} g_j\,\phi_j(x) + \sum_{i\in\mathcal{I}} T_i\,\phi_i(x), \tag{27}$$

where $\phi_k(x) \in \mathcal{U}$ are the standard 1D linear Lagrange basis functions defined over $\Omega$, $\mathcal{B}$ and $\mathcal{I}$ denote the sets of Dirichlet and interior nodes, respectively. The Dirichlet values $g_j$ are imposed directly at the boundary nodes (in this case the node at $x = 0.5$). The unknown coefficients $T_i$ are determined by solving the discrete variational problem over the interior.

We select the test functions $v$ to be identical to the interior basis functions $\phi_i, i \in \mathcal{I}$ (Galerkin projection), which yields a system of nonlinear algebraic equations (or linear, if $k$ and $q$ are linear):

$$\mathsf{K}_N(\mathbf{T}_N;\boldsymbol{\mu})\mathbf{T}_N = \mathbf{q}_N(\mathbf{T}_N;\boldsymbol{\beta})\,, \tag{28}$$

where the unknown coefficient vector contains only the interior degrees of freedom,

$$\mathbf{T}_N = \left[T_i\right]_{i\in\mathcal{I}}^\top \tag{29}$$

and the global stiffness matrix and source vector are assembled from element contributions,

$$\mathsf{K}_N(\mathbf{T}_N;\boldsymbol{\mu}) = \sum_{e=1}^{n_{\text{cell}}} \mathsf{Q}_e^\top \mathsf{K}^e(\mathsf{Q}_e\mathbf{T}_N;\boldsymbol{\mu})\mathsf{Q}_e, \tag{30}$$

$$\mathbf{q}_N(\mathbf{T}_N;\boldsymbol{\beta}) = \sum_{e=1}^{n_{\text{cell}}} \mathsf{Q}_e^\top \mathbf{q}^e(\mathsf{Q}_e\mathbf{T}_N;\boldsymbol{\beta}), \tag{31}$$

where $n_{\text{cell}}$ denotes the total number of cells or elements after meshing the domain $\Omega$; $\Omega^e$ represents the domain of each cell; the matrix $\mathsf{Q}_e \in \mathcal{R}^{d_e\times N}$ is a Boolean assembly matrix restricted to the interior nodes of element $e$ with $d_e$ denoting the local degrees of freedom (DOF) of the $e$th cell. Matrix $\mathsf{Q}_e$ maps $d_e$ to the global DOFs $N$ across the entire mesh (refer to Figs. 5 and 17 for visual representations); $\mathsf{K}^e \in \mathcal{R}^{d_e\times d_e}$ denotes the elemental stiffness matrix, with its $(i,j)$th component defined as

$$\mathsf{K}^e_{ij} = \int_{\Omega^e} k\left(\sum_{s=1}^{d_e}\bar{T}_s\phi^e_s;\boldsymbol{\mu}\right)\nabla\phi^e_i\cdot\nabla\phi^e_j\,d\Omega^e\,, \tag{32}$$

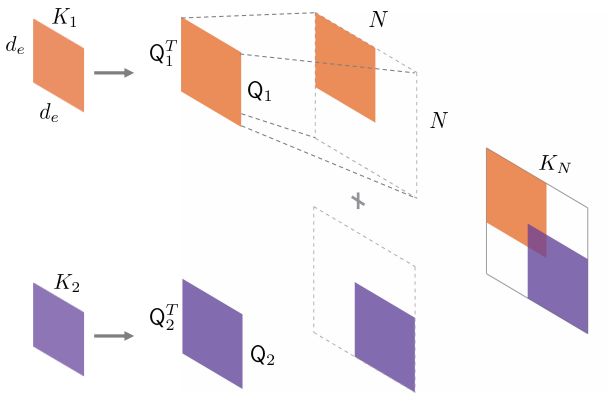



**Figure 5:** Visual representation of the assembly of elemental stiffness matrices for the finite element model of the running example in Sec. 3. Local stiffness matrices $\mathsf{K}_1$ and $\mathsf{K}_2$, corresponding to individual finite elements, are first transformed into the global coordinate system using their respective transformation matrices $\mathsf{Q}_1$ and $\mathsf{Q}_2$. These transformations expand the local matrices into the global framework, mapping local degrees of freedom to the global degrees of freedom. The global stiffness matrix $\mathsf{K}_N$ is then constructed by adding these transformed matrices, summing contributions from overlapping degrees of freedom. Color-coded blocks highlight each local matrix's contribution to the global system.

where $\bar{T}_s = g_s$ if $s \in \mathcal{B}_e$ and $\bar{T}_s = T_s$ if $s \in \mathcal{I}_e$, with $\mathcal{B}_e$ and $\mathcal{I}_e$ denoting the sets of boundary and interior nodes for element $e$; the elemental source vector $\mathbf{q}^e \in \mathbb{R}^{d_e}$, with

$$q_i^e = \int_{\Omega^e} q\left(\sum_{s=1}^{d_e} \bar{T}_s \phi_s^e; \boldsymbol{\beta}\right) \phi_i^e \, d\Omega^e \,, \tag{33}$$

is computed using the same nodal values $\bar{T}_s$. For interior assembly, only components with $i \in \mathcal{I}_e$ are retained, and boundary contributions are subtracted:

$$q_i^e \leftarrow q_i^e - \sum_{j \in \mathcal{B}_e} \mathsf{K}_{ij}^e g_j, \qquad i \in \mathcal{I}_e.$$

The resulting entries are assembled into the global system as in Eq. (28).

We then solve Eq. (28) for the coefficients $\mathbf{T}_N$, and reconstruct the approximate temperature distribution $T(x)$. For the linear example, we assume the parametric linear thermal conductivity is given by

$$k(\mu) = \begin{cases} k_1 = \mu + 16, & \text{for} \quad 0.0 \leq x < 0.4 \\ k_2 = \mu + 30, & \text{for} \quad 0.4 \leq x \leq 0.5 \,, \end{cases} \tag{34}$$

whereas for the nonlinear problem, the nonlinear thermal conductivity is defined as

$$k(T, \mu) = \begin{cases} k_1 = \mu + 16 + 2150/(T - 73.15), & 0.0 \leq x < 0.4 \\ k_1 = \mu + 30 + 2.09 \times 10^{-2} T - 1.45 \times 10^{-5} T^2 + 7.67 \times 10^{-9} T^3 \,. & 0.4 \leq x \leq 0.5 \end{cases} \tag{35}$$

Additionally, the parametric internal heat generation is expressed as

$$q = \begin{cases} q_1(\beta) = \beta + 35000, & \text{linear, for} \quad 0.0 \leq x < 0.4 \\ q_1(T, \beta) = \beta + 35000 + T/10, & \text{nonlinear, for} \quad 0.0 \leq x < 0.4 \\ q_2(\beta) = 10\beta + 5000 & \text{(both) for} \quad 0.4 \leq x \leq 0.5\,. \end{cases} \tag{36}$$

Here both $\mu$ and $\beta$ are parameters. The functional forms of $q$ and $k$ used here are somewhat ad hoc and were chosen primarily for illustrative purposes.

We solve both the linear and the nonlinear problem for 32 pairs of $(\mu, \beta)$, where $\mu \in (-4, 4)$ and $\beta \in (-1000, 1000)$, using our native finite element code library using 5000 1D Lagrangian elements. Though this problem does not require such fine mesh discretization, the mesh was intentionally refined to emulate a large-scale problem.

For the *linear* system, both $\mathbf{q}_N$ and $\mathsf{K}_N$ for different parameter values were derived using *affine decomposition* (see Appendix A). Affine decomposition separates parameter dependence from spatial and temporal parts of a problem. As a result, one can precompute and store parameter-independent components during an *offline* phase, whereas in the *online* phase, evaluate only parameter-dependent functions. This saves the computational effort of iteratively evaluating $\mathbf{q}_N$ and $\mathsf{K}_N$ for different parameters.

For the linear problem, affine decomposition of $\mathbf{q}_N$ and $\mathsf{K}_N$ is written as:

$$\mathsf{K}_N(\mu) = \sum_{j=1}^{2} k_j(\mu) \{\mathsf{K}_N^j\}_{\text{offline}}\,, \tag{37a}$$

$$\mathbf{q}_N(\beta) = \sum_{j=1}^{2} q_j(\beta) \{\mathbf{q}_N^j\}_{\text{offline}}\,, \tag{37b}$$

where $k_1(\mu)$, $k_2(\mu)$, $q_1(\beta)$, and $q_2(\beta)$ are purely parameter dependent, as in Eqs. (34) and (36), and are calculated during the online phase; $\{\mathsf{K}_N^j\}_{\text{offline}}$ and $\{\mathbf{q}_N^j\}_{\text{offline}}$ are calculated offline assuming unit thermal conductivity and unit internal heat generation, and by assembling the elemental contributions from all cells associated with respective properties $k_j$ and $q_j$. With Affine decomposition, the computational cost of many-query computations involving numerous parameter values is drastically reduced as the associated computational effort grows linearly with the number of parameters.

For the *nonlinear* problem, however, an affine decomposition is not feasible because $\mathbf{q}_N$ and $\mathsf{K}_N$ must be re-evaluated for each parameter, that too iteratively while solving for $T_N$ via Newton's method. Consequently, solving the nonlinear problem is considerably more expensive. While solving the linear problem across 32 parameters requires approximately 32 seconds of CPU time, the nonlinear problem demands around 500 seconds. For both the linear and the nonlinear problems the parameters were selected using the Sobol scheme [190] to ensure comprehensive coverage of the parameter space as shown in Figs. 6a and 6b. The corresponding solution snapshots for both problems are shown in Figs. 6c and 6d.

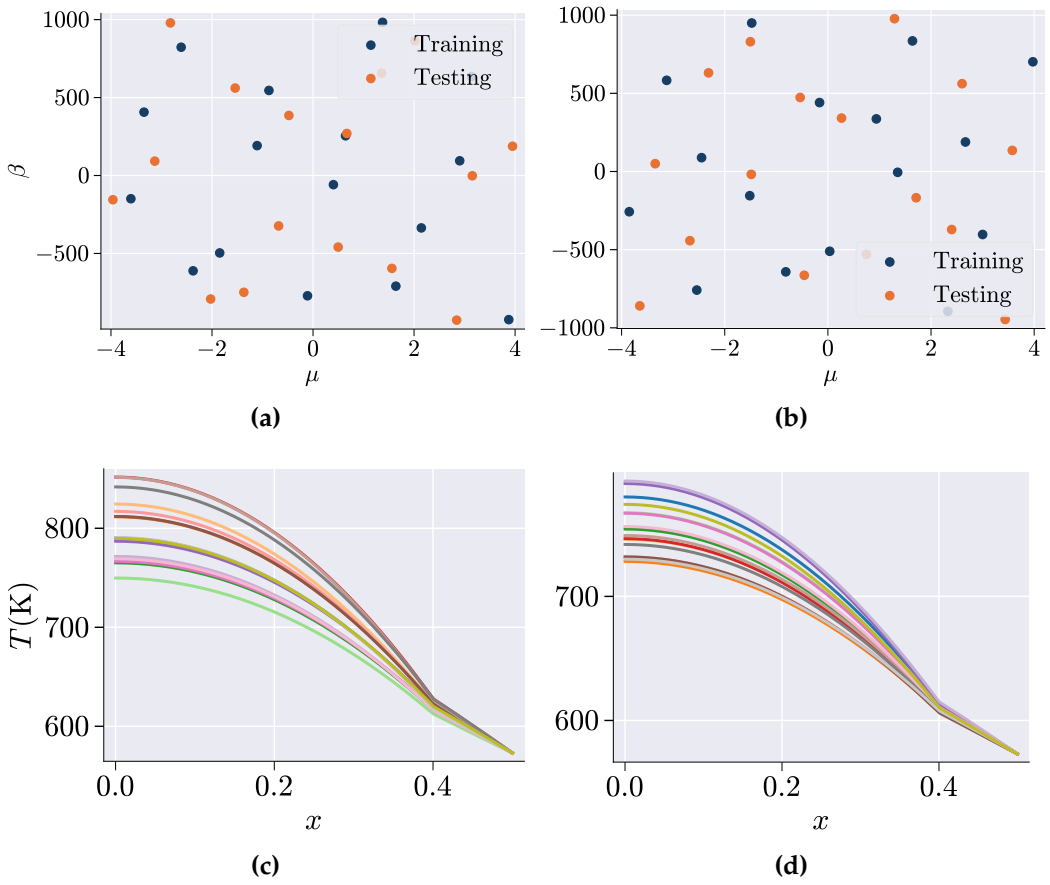



**Figure 6:** High Fidelity solution snapshots for the linear and the nonlinear system corresponding to 16 pairs of the thermal conductivity and heat source values as defined in Eqs. (34) to (36). Figs. (a) and (c) correspond to the linear system, whereas (b) and (d) correspond to the nonlinear system. Here the parameter pairs are selected based on the sobol sequence [190], which explores the entire parameter space.

To obtain the reduced order models, POD was applied on the 50% of the snapshots (treated as training data) and the dominant POD basis vectors, that span the reduced subspaces, were stored in $\widetilde{\mathbb{U}}$. The number of basis vectors or the ROM dimension was selected by determining the minimum $k$, for which, $1 - r_k$ (Eq. 5), which indicates the fraction of the uncaptured data-variance for a $k$ dimensional subspace, was below a tolerance $\epsilon = 10^{-12}$, which yielded a three dimensional linear ROM ($n = 3$) and a five dimensional nonlinear ROM ($n = 5$) as shown in Figs. 7a and 7b. The corresponding modes selected are visualized in Figs. 7c and 7d. We note that, in this case, the reduced basis method (Section 2.4) could also be implemented for constructing $\widetilde{\mathbb{U}}$.

We express the approximate full order solution of Eq. (28) as

$$\widetilde{\mathbf{T}}_N = \overline{\mathbf{T}}_N + \widetilde{\mathbb{U}}\mathbf{T}_n\,, \tag{38}$$

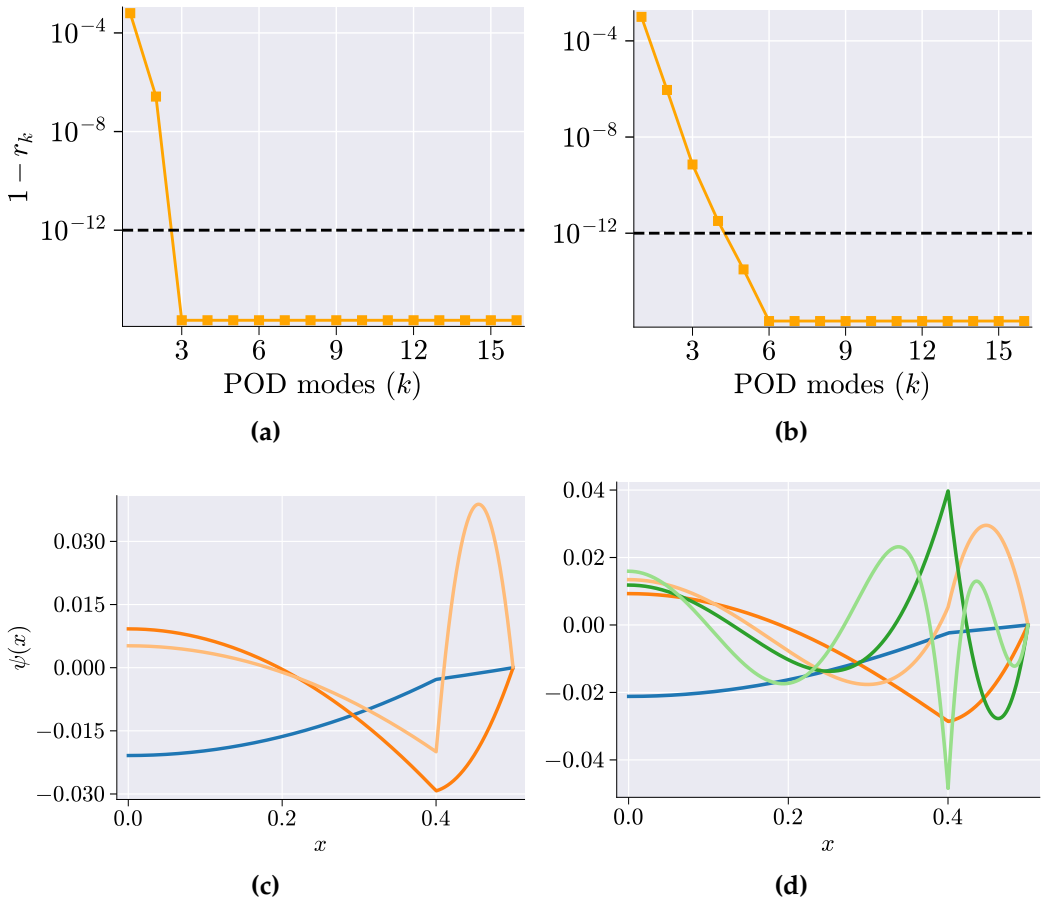



**Figure 7:** Singular values and left singular vectors obtained from the solution snapshots of the linear (left column) and nonlinear (right column) problems. Figures (a) and (b) display the decay of the singular values, with the dashed line indicating the chosen tolerance threshold, while (c) and (d) depict the corresponding singular vectors.

where $\mathbf{T}_n$ denotes the reduced order solution and $\overline{\mathbf{T}}_N$ (equivalent to $\mathbf{w}^{ref}$ in Eq. (7)) represents the mean of the different temperature snapshots used for POD. Then, following Section 2.3, and assuming Galerkin projection, Eq. (28) is reduced to

$$\mathsf{K}_n \mathbf{T}_n = \mathbf{q}_n - \widetilde{\mathbb{U}}^\top \mathsf{K}_N \overline{\mathbf{T}}_N \,, \tag{39}$$

where

$$\mathbf{q}_n = \begin{cases} \widetilde{\mathbb{U}}^\top \mathbf{q}_N(\mu), & \text{linear}, \\ \widetilde{\mathbb{U}}^\top \mathbf{q}_N(\widetilde{\mathbf{T}}_N; \beta), & \text{nonlinear}, \end{cases} \tag{40a}$$

$$\mathsf{K}_n = \begin{cases} \widetilde{\mathbb{U}}^\top \mathsf{K}_N(\mu) \widetilde{\mathbb{U}}, & \text{linear} \\ \widetilde{\mathbb{U}}^\top \mathsf{K}_N(\widetilde{\mathbf{T}}_N; \mu) \widetilde{\mathbb{U}}, & \text{nonlinear} \end{cases} \tag{40b}$$

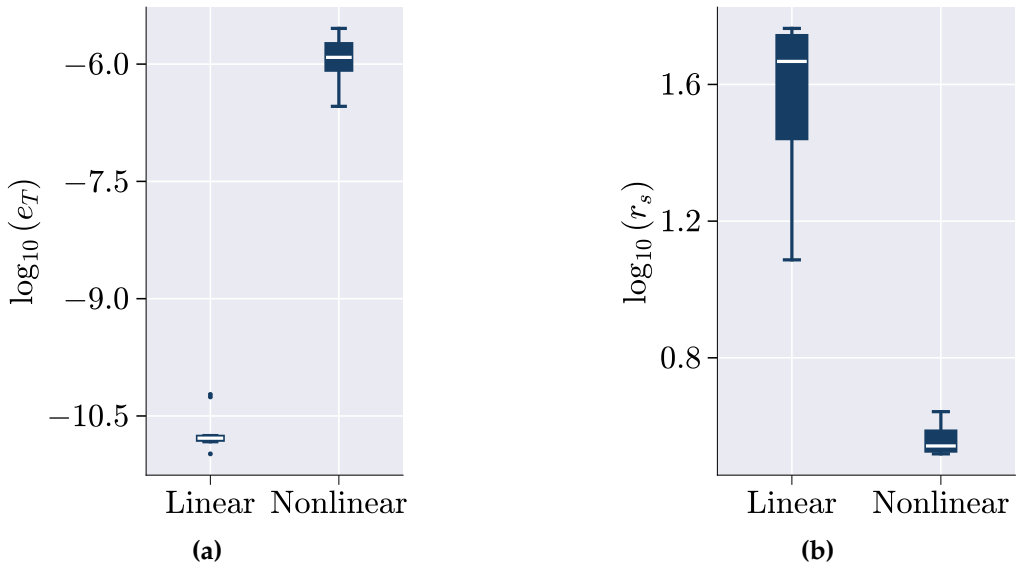



**Figure 8:** Difference in accuracy ($e_T$) and speed-up ($r_s$) between projection based linear and nonlinear ROMs.

For the linear problem, due to affine decomposition, the computation of $\mathsf{K}_n(\mu)$ and $\mathbf{q}_n(\beta)$ is simple and fast since one can write:

$$\mathsf{K}_n(\mu) = \sum_{i=1}^{2} k_i(\mu)\widetilde{\mathbb{U}}^\top \{\mathsf{K}_i\}_{\text{offline}}\widetilde{\mathbb{U}} \tag{41a}$$

$$\mathbf{q}_n(\beta) = \sum_{j=1}^{2} q_j(\beta)\widetilde{\mathbb{U}}^\top \{\mathbf{q}_j\}_{\text{offline}}\,. \tag{41b}$$

The performance of the derived ROM was then tested using the remaining 50% of the *test* parameters and the corresponding snapshots.

As a measure of the ROM performance, we define the associated percentage relative error ($e_T$) and speed-up factor ($r_s$) as follows:

$$e_T = 100 \times \frac{\|\mathbf{T}_N(\mu,\beta)_k - \widetilde{\mathbf{T}}_N(\mu,\beta)_k\|}{\|\mathbf{T}_N(\mu,\beta)_k\|}\,, \tag{42a}$$

where as before $\|\cdot\|$ indicates the $L_2$ norm; $(\mu,\beta)_k$ denotes the $k$th parameter pair, and

$$r_s = \frac{t_{HFM}}{t_{ROM}}\,, \tag{42b}$$

where $t_{HFM}$ and $t_{ROM}$ indicate the corresponding CPU-time required by the high-fidelity model and the reduced order model, respectively. With $n = 3$, for the test parameters, the

linear ROM demonstrates high accuracy with mean $e_T \approx 10^{-10}$ and significant speed-up with mean $r_s \approx 40$, as shown using the box plots in Figs. 8a and 8b.

For the nonlinear ROM, although the accuracy was quite reasonable, it was lower than in the linear case ($e_T = 10^{-6}$ compared to $10^{-10}$) (Although not shown, increasing $n$ was found to lower $e_T$). More critically, the speed-up factor was substantially reduced: $4\times$ compared to $40\times$ in the linear case. The primary reason for this reduced speed-up is evident from Eq. (40b), which indicates that the evaluation of the reduced stiffness matrix during the Newton iteration involves iterative evaluation of $\mathsf{K}_N$ and its projection onto the reduced subspace. And for every evaluation, $\mathsf{K}_N$ requires the full-order solution $\widetilde{\mathbf{T}}_N$, which is derived from $\mathbf{T}_n$ following Eq. (38). Consequently, for nonlinear problems, regular projection-based reduced order models experience a significant decline in efficiency.

# 4 Hyper-reduction techniques

As discussed in the previous section, the efficiency of the projection-based ROMs is hindered by the need to iteratively map the reduced solutions back to the full-dimensional state to compute the solution-dependent non-linearities of the system and then again project them back to the reduced space. In this section, we review the idea of hyper-reduction in detail, which offers efficient strategies to fast-compute the nonlinearities of a model with sufficient accuracy.

In general, nonlinearities can either be inherently polynomials, reducible to a polynomial form [104], or non-polynomial in nature. Consider the following example of polynomial nonlinearity written in Kronecker product form:

$$\mathbf{f}_N = \mathsf{K}_N\mathbf{w}_N + \mathsf{B}_N(\mathbf{w}_N \otimes \mathbf{w}_N) + \mathsf{C}_N(\mathbf{w}_N \otimes \mathbf{w}_N \otimes \mathbf{w}_N) + \dots \tag{43}$$

where $\mathsf{K}_N \in \mathcal{R}^{N\times N}$, $\mathsf{B}_N \in \mathcal{R}^{N\times N^2}$, and $\mathsf{C}_N \in \mathcal{R}^{N\times N^3}$ are the coefficient tensors for the linear, quadratic, and cubic terms, respectively, and $\otimes$ denotes the Kronecker product [191].

For an $n$-dimensional column vector $\mathbf{w}_N = [w_1, w_2, \dots, w_n]^\top$, the outer product $\mathbf{w}_N \otimes \mathbf{w}_N$ is defined as:

$$\mathbf{w}_N \otimes \mathbf{w}_N = [w_1^2, w_1w_2, \dots, w_1w_n, w_2w_1, w_2^2, \dots, w_2w_n, \dots, w_n^2] \in \mathcal{R}^{N^2}. \tag{44}$$

The projection of $\mathbf{f}_N$ onto the reduced basis space $\mathbb{U}$ can then be expressed *exactly* as [103, 106]:

$$\widetilde{\mathbb{U}}^\top\mathbf{f}_N = \mathbf{f}_n = \mathsf{K}_n\mathbf{w}_n + \mathsf{B}_n(\mathbf{w}_n \otimes \mathbf{w}_n) + \mathsf{C}_n(\mathbf{w}_n \otimes \mathbf{w}_n \otimes \mathbf{w}_n) + \dots \tag{45}$$

where $\mathbf{w}_N = \widetilde{\mathbb{U}}\mathbf{w}_n$, and the reduced matrices are defined as:

$$\mathsf{K}_n = \widetilde{\mathbb{U}}^\top\mathsf{K}\widetilde{\mathbb{U}}, \quad \mathsf{B}_n = \widetilde{\mathbb{U}}^\top\mathsf{B}(\widetilde{\mathbb{U}} \odot \widetilde{\mathbb{U}}), \quad \mathsf{C}_n = \widetilde{\mathbb{U}}^\top\mathsf{C}(\widetilde{\mathbb{U}} \odot \widetilde{\mathbb{U}} \odot \widetilde{\mathbb{U}}), \tag{46}$$

and so on for higher-order terms. The symbol $\odot$ denotes the Khatri-Rao product (column-wise Kronecker product) of two matrices [191]. For a matrix $\widetilde{\mathbb{U}} = [\mathbf{U}_1\, \mathbf{U}_2 \ldots \mathbf{U}_n] \in \mathcal{R}^{N \times n}$:

$$\widetilde{\mathbb{U}} \odot \widetilde{\mathbb{U}} = [\mathbf{U}_1 \otimes \mathbf{U}_1 \quad \mathbf{U}_2 \otimes \mathbf{U}_2 \quad \ldots \quad \mathbf{U}_n \otimes \mathbf{U}_n] \in \mathcal{R}^{N^2 \times n}. \tag{47}$$

These reduced matrices $\mathsf{K}_n$, $\mathsf{B}_n$, and $\mathsf{C}_n$ can be precomputed, thereby eliminating the need to access the full order solution.

However, in cases where reduction to such a polynomial form is *not* possible or preferred, and iterative access to full order solution is the only viable option to evaluate the nonlinear term, hyper-reduction is adopted. Hyper-reduction focuses on scaling the computation with the size of the reduced coordinate vector $n$ rather than $N$, the size of the high-fidelity model (HFM). This is achieved through sparse probing of the nonlinear terms across the computational domain, rather than calculating them at every nodal point. This approach is characterized by the use of a reduced/sampled mesh, where contributions from a significant portion of the mesh elements are omitted and accounted for through approximation. These methods introduce an additional layer of reduction on top of the projection-based reduction while maintaining the accuracy of the reduced-order model.

Hyper-reduction algorithms are classified into two major categories: approximate-after-project and project-after-approximate [55]. These names indicate the sequence of operations to handle nonlinearities. In the approximate-after-project category, nonlinearities are first approximated within the full-dimensional space and then projected onto the reduced subspace. Conversely, in the project-after-approximate category, this sequence is reversed. In Fig. 9, we show the different hyper-reduction algorithm widely used in each category.

The approximate-then-project methods originated with the gappy proper orthogonal decomposition (POD) method [192], initially developed for reconstructing images from limited pixel data. These methods approximate nonlinear terms by interpolating values at selected nodes within the computational mesh using a few empirical basis functions, then exactly projecting these approximations onto the reduced subspace in ROM formulation. Various techniques (refer to Fig. 9), including the empirical interpolation method (EIM) [68, 69], discrete empirical interpolation method (DEIM) [70], best-points interpolation [71], missing-point estimation, and collocation methods [193, 194], achieve this by selectively evaluating nonlinear terms at a few grid points. This selective evaluation effectively yields a reduced or sampled mesh significantly enhancing computational efficiency.

Project-then-approximate methods, on the other hand, estimate the *reduced-order* vectors and matrices resulting from projecting the nonlinearities of the high-fidelity model (HFM) onto the latent space. These methods also generate a reduced mesh by selecting a subset of elements or entities from the full computational mesh with an aim to closely approximate the true projections. These methods aim for more accurate and robust pROM approximations compared to the approximate-then-project approach. They can be viewed as generalized quadrature rules, where quadrature points and weights are learned, leading to effective mesh sampling [195]. Examples include the energy-conserving sampling and weighting method [31, 72–74], the empirical cubature method (ECM) [75–78], and the linear program-based empirical quadrature method (EQP) [79].

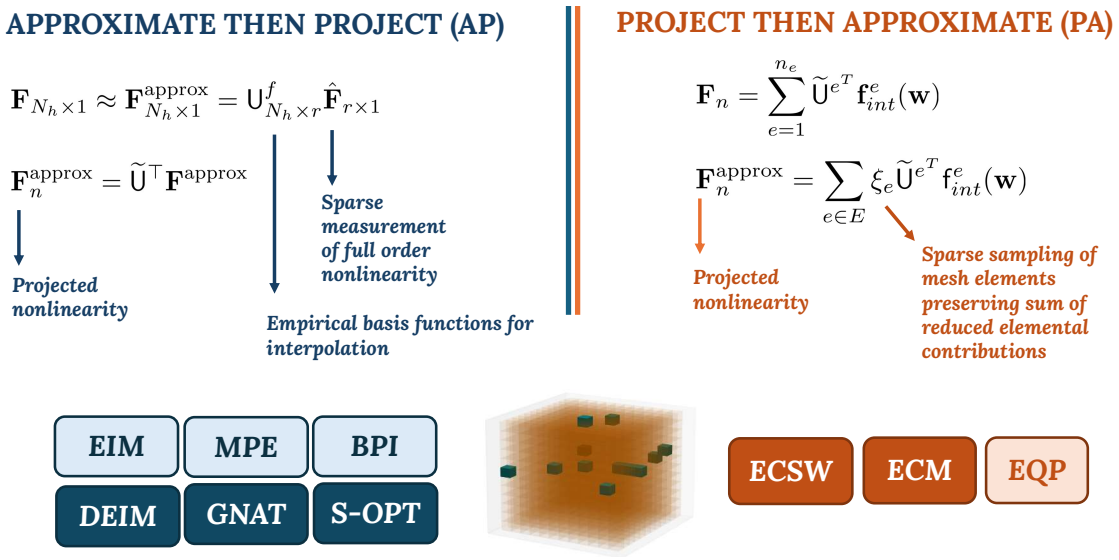



**Figure 9:** Hyper-reduction algorithms are broadly grouped into two categories: Approximate Then Project (AP) and Project Then Approximate (PA). The AP category uses sparse measurements taken directly from the full-order nonlinear terms and interpolates these with empirical basis functions to approximate the projected nonlinearity. In contrast, the PA category first projects the nonlinear terms onto a reduced-order basis, followed by sparse sampling of mesh elements to approximate the sum of the projected elemental contributions. Examples of hyper-reduction algorithms include the Empirical Interpolation Method (EIM) [69], Missing Point Estimation (MPE) [193], Best Point Interpolation (BPI) [71], Discrete Empirical Interpolation Method (DEIM) [70], Gauss-Newton Approximation Tensor (GNAT) [51], S-optimality-based point selection (S-OPT) [196], Energy Conserving Sampling and Weighing (ECSW) [31], Empirical Cubature Method (ECM) [159], and Empirical Quadrature Procedure (EQP) [197]. Among these, the current paper explicitly focuses on DEIM, GNAT, S-OPT, ECSW, and ECM, highlighted using darker background boxes.

In the remainder of this section, we review several of these algorithms along with their implementation details. Within the category of approximate-then-project methods, we examine the Discrete Empirical Interpolation Method (DEIM) and several DEIM-inspired algorithms. For project-then-approximate methods, we discuss the Energy Conserving Sampling and Weighing (ECSW) method and the Empirical Cubature Method (ECM).

## 4.1 Approximate-then-project hyper-reduction strategies

We begin by briefly introducing gappy-POD [192], which was seminal in putting forward the idea behind the approximate-then-project hyper-reduction, and subsequently move onto describing the different hyper-reduction algorithms. In Fig. 10 we present a brief timeline of the seminal papers on the approximate-then-project (AP) hyper-reduction algorithms.

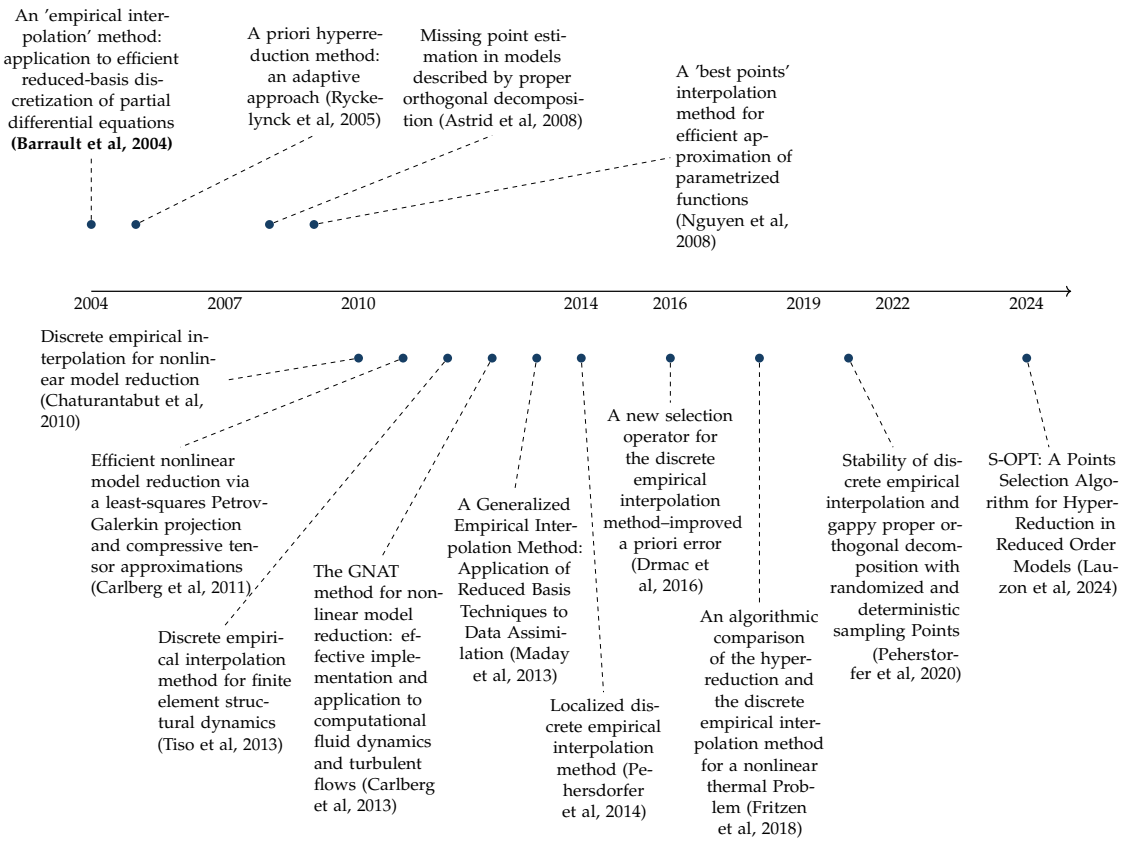



**Figure 10:** Timeline of Seminal Papers* on Approximate-then-Project hyper-reduction Algorithms. [51, 67, 69–71, 80, 82, 83, 157, 193, 196, 198–200] (**list not exhaustive*).

### 4.1.1 Gappy-Proper orthogonal decomposition (Gappy-POD)

---

- **Objective:** Reconstruct facial images using sparsely distributed pixels.
- Build a database containing complete images of various faces.
- Apply POD on the face database to construct an "eigenfaces" database.
- Approximately reconstruct a new face using the eigenfaces and partially available pixels of an out-of-sample image.

---

The Gappy proper orthogonal decomposition (POD) method was developed in the context of facial image reconstruction from sparsely sampled pixels, given an existing large database of grayscale images of different human faces [183].

In the first step, proper orthogonal decomposition (POD) is applied to the entire facial image database to identify and express the most dominant facial features in the form of the proper orthogonal modes (POMs), which are termed as the *eigenfaces*. The subset of the most dominant eigenfaces are then utilized to approximate an image from a finite number of sampled pixels. Figure 11a shows an example of a marred facial image with sparsely distributed pixels. As discussed below, Gappy POD essentially reconstructed a complete image from these sparse pixels by *interpolating* with the eigenfaces.

Let a complete image be expressed as a 1D vector $\mathbf{w}_N \in \mathcal{R}^{N\times 1}$, which contains the grayscale values at different pixel locations (obtained by reshaping the 2D image-matrix into 1D). We define a sampling matrix $\mathsf{Z} \in \mathcal{R}^{N\times r}$, which samples a complete image at sparse locations to produce images as in Fig. 11a (sparse pixels on black background). More formally, $\mathsf{Z}^\top$ projects the vector representation of a complete image, $\mathbf{w}_N$, onto an $r$-dimensional space, producing $\mathbf{w}_r$, which represents the marred image:

$$\mathbf{w}_r = \mathsf{Z}^\top \mathbf{w}_N \,. \tag{48}$$

As an example, $\mathsf{Z}$ may take the following form:

$$\mathsf{Z} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \vdots & \cdots & \vdots \\ \vdots & 0 & \cdots & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} , \tag{49}$$

which has $r$ columns, with each column consisting of zeros except for a single element that is set to one, representing the specific pixel location being sampled.

For reconstruction, the image is expressed as the linear combination of the eigenfaces. We write this as

$$\widetilde{\mathbf{w}}_N = \widetilde{\mathbb{U}}_f \mathbf{w}_n \,, \tag{50}$$

where vector $\widetilde{\mathbf{w}}_N$ represents the reconstructed image, matrix $\widetilde{\mathbb{U}}_f \in \mathcal{R}^{N\times n}$ contains the first $n$ dominant eigenfaces reshaped into 1D vector of length $N$, and $\mathbf{w}_n \in \mathcal{R}^{n\times 1}$ is a vector containing the coefficients of $\widetilde{U}_f$. It is enforced that the pixels of the marred image $\mathbf{w}_r$ match closely with the pixels of $\widetilde{\mathbf{w}}_N$ at the sampled location such that $\mathbf{w}_n$ minimizes the following reconstruction error:

$$\mathbf{w}_n = \arg\min_{\mathbf{w}_n} \|\mathbf{w}_r - \mathsf{Z}^\top \widetilde{\mathbb{U}}_f \mathbf{w}_n\| \,, \tag{51}$$

where $\|\cdot\|$ denotes the $L_2$ norm.

The normal equation [201] for this optimization problem is given by

$$\mathbf{w}_n = \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \mathbf{w}_r = \Omega^{-1} \mathsf{Z}^\top \widetilde{\mathbb{U}}_f \mathbf{w}_r \,, \tag{52}$$

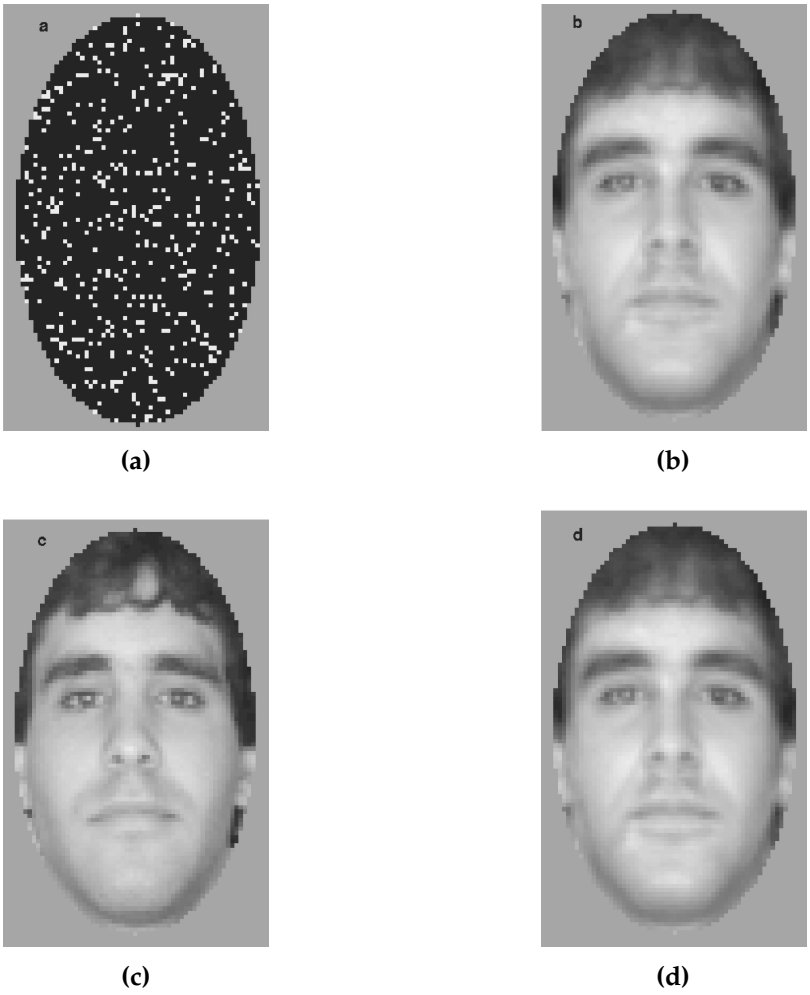



**Figure 11:** Reconstruction of marred images using Gappy-POD (Adapted with permission from [192] © Optical Society of America).

where $^\dagger$ denotes the pseudo-inverse [202] and $\Omega = \left(\widetilde{\mathbb{U}}_f^\top \mathsf{Z}\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$. Subsequently, the complete image is reconstructed using Eq. (50). Figure 11b shows the image reconstructed from the pixels in Fig. 11a using 50 of eigenfaces. The reconstructed images look reasonably close to the true image shown in Fig. 11c. Figure 11d on the other hand, represents the direct projection (with all pixels) of the face onto 50 eigenfaces, which looks identical to Fig. 11c.

**Remark 1** Note that the columns of $\mathsf{Z}^\top \widetilde{\mathbb{U}}_f$ contain the row-wise sampled columns of $\widetilde{\mathbb{U}}_f$. Unlike the columns of $\widetilde{\mathbb{U}}_f$, which are the basis vectors, these sampled columns are not orthonormal, and hence, $\Omega$ is not an identity matrix. However, as the sampling space becomes denser, then $\Omega \to \mathsf{I}_n$. As a matter of fact, the efficacy of gappy-POD depends

upon the condition number of $\Omega$. If the image sampling is such that resulting $\Omega$ has a large condition number, which indicates $\Omega$ is close to being singular and non-invertible, the reconstructed image may deviate significantly from the original image. Further details on how sampling affects the condition number can be found in [171]. However, with a proper sampling matrix $\mathsf{Z}$, which yields lower (preferably, close to unity) condition number for $\Omega$, it may be possible to achieve reasonably accurate reconstruction of an image from a limited number of pixels. This core idea is leveraged in the AP hyper-reduction algorithms, as we will discuss next.

### 4.1.2 Discrete empirical interpolation method (DEIM)

---

- **Objective:** Determine interpolation points on the computational domain associated with a High-Fidelity Model (HFM) to approximate a nonlinear term and reduce computational complexity.
- Gather snapshots of system state variables over time or training parameters and apply POD to construct a reduced basis for the HFM.
- Collect snapshots of the nonlinear term separately and apply POD to construct a reduced basis specific to the nonlinear term.
- Select interpolation points that retains maximize information of the nonlinear term using the POD basis from its snapshots.
- Approximate the nonlinear term in the ROM by evaluating it at selected points and reconstructing it with the reduced basis, bypassing full-order complexity.

---

The Discrete Empirical Interpolation Method (DEIM), introduced in [70], was developed to efficiently compute the projected nonlinearity, such as $\mathbf{f}_n$ in Eq. (21), by approximating the full-order nonlinear term, such as $\mathbf{f}_N$ in Eq. (1), via a low-dimensional representation constructed from a set of reduced basis vectors and carefully chosen sampling locations. The approach is very similar to Gappy POD. However, unlike Gappy POD, which reconstructs full images from a *given* incomplete image data, DEIM *selects* specific components (sampling/interpolation points) of the nonlinear force vector according to a sampling strategy.

Given a parametric nonlinear vector $\mathbf{f}_N : \Omega \times \mathcal{P}_{\text{DEIM}} \to \mathcal{R}^N$, where $\Omega$ is the spatial domain and $\mathcal{P}_{\text{DEIM}}$ is the parameter space, DEIM approximates $\mathbf{f}_N$ using a given set of basis vectors $\widetilde{\mathbb{U}}_f \in \mathcal{R}^{N\times r}$ (derivation discussed later) and the values of $\mathbf{f}_N$ at specific sampling points. The approximation, denoted by $\widetilde{\mathbf{f}}_N$, is expressed as

$$\widetilde{\mathbf{f}}_N(\mathbf{w}_{\boldsymbol{\mu}};\boldsymbol{\mu}) = \widetilde{\mathbb{U}}_f \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \mathsf{Z}^\top \mathbf{f}_N(\mathbf{w}_{\boldsymbol{\mu}};\boldsymbol{\mu}) = \mathsf{M}_D \mathbf{f}_N(\mathbf{w}_{\boldsymbol{\mu}};\boldsymbol{\mu}), \tag{53}$$

**Algorithm 1** Discrete Empirical Interpolation Method: Offline Phase [68, 70]

**function** $[\mathbf{Q}, \mathcal{J}, \mathbf{Z}] = \text{DEIM\_OFFLINE}(\mathsf{F}, tol)$
$[\mathbf{U}_f^1, \ldots, \mathbf{U}_f^r] = \text{POD}(\mathsf{F}, tol)$ // Perform POD on force snapshots $\mathsf{F}$, obtaining $r$ basis vectors based on tolerance *tol*
$i_1 = \arg\max_{i=1,\ldots,N} |\mathbf{U}_f^1|_i$ // Select the index of the maximum absolute entry in the first POD basis vector
$\mathsf{Q} = \mathbf{U}_f^1$, $\mathcal{J} = \{i_1\}$ // Initialize basis matrix $\mathsf{Q}$ and index set $\mathcal{J}$ with the first POD basis vector and index
$\mathsf{Z} = [\mathbf{e}_{i_1}]$ // Initialize sampling matrix $\mathsf{Z}$ with the standard basis vector corresponding to $i_1$
**for** $m = 2 : r$ **do**
  Compute $\tilde{\mathbf{q}} = \mathsf{Q}(\mathsf{Z}^\top \mathsf{Q})^\dagger \mathsf{Z}^\top \mathbf{U}_f^m$ // Project the current POD basis vector $\mathbf{U}_f^m$ onto the span of accumulated basis vectors in $\mathsf{Q}$, using $\mathsf{Z}$ for row selection
  $\mathbf{r} = \mathbf{U}_f^m - \tilde{\mathbf{q}}$ // Calculate the residual between $\mathbf{U}_f^m$ and its projection
  $i_m = \arg\max_{i \notin \mathcal{J}} |\mathbf{r}|_i$ // Select index of the maximum absolute entry in the residual that is not in $\mathcal{J}$
  $\mathbf{Q} \leftarrow [\mathsf{Q}\, \mathbf{U}_f^m]$, $\mathcal{J} \leftarrow \mathcal{J} \cup \{i_m\}$ // Update basis matrix $\mathsf{Q}$ and index set $\mathcal{J}$ with the new POD basis vector and index
  $\mathsf{Z} \leftarrow [\mathsf{Z}\, \mathbf{e}_{i_m}]$ // Append the standard basis vector for $i_m$ to $\mathsf{Z}$, updating the sampling matrix
**end for**
**end function**

where the sampling matrix $\mathsf{Z} \in \mathcal{R}^{N \times r}$ contains the sampling locations, and $\mathsf{M}_D$ defined as

$$M_D = \widetilde{\mathbb{U}}_f \left( \mathsf{Z}^\top \widetilde{\mathbb{U}}_f \right)^\dagger \mathsf{Z}^\top \tag{54}$$

is an oblique projection matrix (see Appendix A). Note that Equation (53) follows directly from Eqs. (50) and (52).

This approximation is subsequently used to calculate the projected nonlinearity $\widetilde{\mathbf{f}}_n$ for Eq. (21):

$$\widetilde{\mathbf{f}}_n = \widetilde{\mathbb{U}}^\top \widetilde{\mathbf{f}}_N \,, \tag{55}$$

where $\widetilde{\mathbb{U}}$ contains the reduced basis vectors derived from the solution snapshots (Eq. 14).

### DEIM approximation framework

The basis vectors in $\widetilde{\mathbb{U}}_f \in \mathcal{R}^{N \times r}$ are obtained by directly applying SVD to the snapshot matrix consisting of snapshots of $\mathbf{f}_N(\mathbf{w}_{\boldsymbol{\mu}_q}; \boldsymbol{\mu}_q)$ , which we will refer to as DEIM-snapshots, for each $\boldsymbol{\mu}_q \in \mathcal{P}_{\text{DEIM}}$:

$$\mathsf{F} = \left[ \mathbf{f}_N(\mathbf{w}_{\boldsymbol{\mu}_1}; \boldsymbol{\mu}_1), \cdots, \mathbf{f}_N(\mathbf{w}_{\boldsymbol{\mu}_{N_s}}; \boldsymbol{\mu}_{N_s}) \right] \approx \widetilde{\mathbb{U}}_f \Sigma_f \mathbb{V}_f^\top \,. \tag{56}$$

The left singular vectors corresponding to the first $r$ largest singular values form the columns of $\widetilde{\mathbb{U}}_f$.

The error associated with the approximation in Eq. (53) is bounded as [70]:

$$\|\mathbf{f}_N - \widetilde{\mathbf{f}}_N\| \leq \| \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \| \, \|(\mathsf{I} - \widetilde{\mathbb{U}}_f \widetilde{\mathbb{U}}_f^\top)\mathbf{f}_N\| \,, \tag{57}$$

where $\|\mathbf{f}_N - \widetilde{\mathbf{f}}_N\|$ denotes the $L_2$ norm of the approximation error. The term $\| \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \|$ reflects the sensitivity to the selected sampling points. As discussed in Section 4.1.1, a large value of this term indicates a poor condition number of $\mathsf{Z}^\top \widetilde{\mathbb{U}}_f$, which in turn indicates that the sampling points may not be optimal for reconstructing the nonlinear term (note, by optimal here we actually mean quasi-optimal since true optimality cannot be achieved in general due to dependence on $\mathbf{f}_N$) [70, 82]. Finally, the term $\|(\mathsf{I} - \widetilde{\mathbb{U}}_f \widetilde{\mathbb{U}}_f^\top)\mathbf{f}_N\|$ denotes the projection error, which quantifies the residual portion of $\mathbf{f}_N$ not captured by the span of reduced basis $\widetilde{\mathbb{U}}_f$.

In DEIM, the approximation error is reduced by reducing $\| \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \|$ on the right-hand side by maximizing its smallest singular value $\sigma_{\min}\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$, which in turn minimizes the condition number $\kappa = \sigma_{\max}\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right) / \sigma_{\min}\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$ ensuring that $\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$ is invertible. The DEIM algorithm as described in Algorithm 1 achieves this via an appropriate construction of the sampling matrix $\mathsf{Z}$.

This greedy Algorithm 1 selects sampling points by iteratively searching for the index where the residual $\mathbf{r}$ (step 8) is maximum, thereby limiting the step-wise growth of $\| \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \|$ and minimizing $\kappa$, ensuring invertibility. In essence, this process results in the selection of $r$ independent *rows* of $\widetilde{\mathbb{U}}_f$.

**Remark 2** DEIM reduces complexity of calculating $\mathbf{f}_N$ from $\mathcal{O}(N)$ to $\mathcal{O}(r)$, enabling efficient handling of large-scale problems.

**Remark 3** Computation of $\mathbf{f}_N$ using Eq. (53) for any $\boldsymbol{\mu} \in \mathcal{P}_{\text{DEIM}}$ during the online phase is accelerated by the offline computation of $\mathsf{M}_D$ in Eq. (54), sometimes referred to as the DEIM-matrix.

**Remark 4** By selecting $\mathsf{Z}$ that maximizes $\sigma_{\min}\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$, the DEIM algorithm achieves *E-optimality* [196, 203], reducing the maximum approximation error defined in Eq. (57).

**Remark 5** Discrete Empirical Interpolation Method (DEIM) is a discrete adaptation of its predecessor Empirical Interpolation Method (EIM) [69], as outlined in Appendix B.

EIM focuses on interpolating nonlinear *functions* by simultaneously constructing empirical basis functions and determining sampling nodes in an iterative manner for a quasi-optimal representation of the nonlinear function. In contrast, DEIM decouples these steps in a discrete framework by first applying Singular Value Decomposition (SVD) to the DEIM snapshots to generate basis functions, and then selecting the indices.

**Implementation on the running example problem**

Consider the nonlinear heat conduction problem in Section 3. We use the same training and test datasets each containing 16 parameter pairs, $(\mu, \beta)_k$ for $k = 1, \ldots, 16$, to formulate and evaluate DEIM-based hyper-reduced order models (hyper-ROMs). The POD basis matrix, $\widetilde{\mathbb{U}}$, in this case is constructed using 4 basis vectors.

To implement Algorithm 1 during the offline phase, we started by gathering snapshots of the source term $\mathbf{q}_N$ for the 16 parameter pairs. Given this is a steady-state heat conduction problem, selecting either the source term or the product of the stiffness matrix $\mathsf{K}_N$ with the temperature solution yields the same DEIM snapshots. Figure 12a visualizes the snapshots of the source term for different parameter values, while Fig. 12b shows the singular value decay of the DEIM snapshots, which informs the selection of the reduced basis size $r$ for constructing $\mathbb{U}_f \in \mathcal{R}^{N \times r}$. We selected $r = 4$ (though Fig. 12b suggests $r = 3$ would also suffice), which results in 4 DEIM points. This led to a section of 8 elements out of a total of 5000, as each node in the 1D mesh (excluding boundaries) is shared by two elements. This is shown in Fig. 13. The DEIM sampling algorithm then generated the DEIM matrix $\mathsf{M}_D$ as in Eq. (54), facilitating efficient evaluations of both the right-hand and left-hand sides of Eq. (28). The dimension of the reduced solution subspace $n$ was also chosen to be 4 (See Remark 6 for the rationale).

In Fig. 14, we compare the computational speed-up (Eq. 42b) and the relative percentage error (Eq. 42a) between the hyper-ROM and the conventional projection-based ROM. As shown in Figure 14a, the hyper-ROM achieves a relative error of 0.001%, which, although slightly higher than that of the regular ROM, is accompanied by a substantial computational speed-up—nearly 30 times that of the regular ROM.

**Remark 6** In DEIM-based hyper-reduction, instability often results from selecting an inconsistent number of modes for DEIM ($r$) and solution snapshots ($n$), leading to non-invertible projected quantities. To maintain stability in the reduced-order model, it is recommended that the number of solution modes should not exceed the number of singular vectors obtained from the DEIM snapshots, i.e. $n \leq r$. This rationale informed our choice to set $r = n = 4$ in our example problem. We observed that increasing $n$ further leads to instability. Moreover, singular values associated with the DEIM snapshots for $r > 4$ are so small that the corresponding singular vectors are essentially noise and also leads to instability.

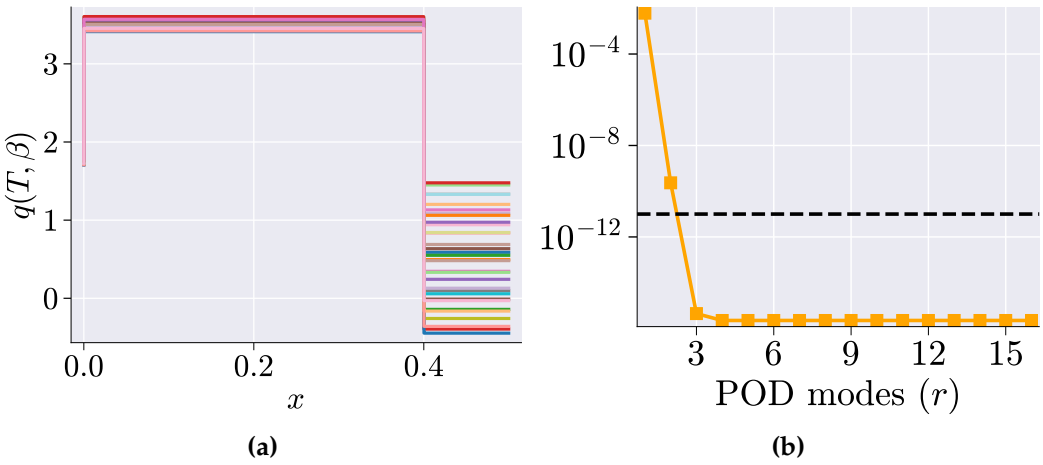



**Figure 12:** The snapshots of the source function corresponding to various parameter values used for DEIM are shown in (a). In (b), the decay of singular values, obtained by applying SVD on the snapshots, is illustrated. This decay informs the selection of the number of singular vectors, $r$, to use in the DEIM algorithm. Here, $r = 4$ was selected.

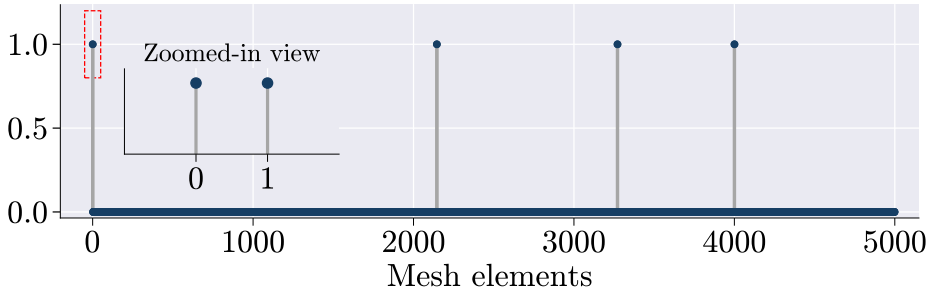



**Figure 13:** The reduced mesh obtained using DEIM. Excluded elements lie on the blue line. As shown in the inset, for each DEIM point, all associated elements (in this case, two per point) are selected. The resulting reduced mesh in this case comprises only 8 elements, which constitutes 0.16% of the total 5,000 elements in the original high-fidelity model.

**Remark 7** If the DEIM snapshots exhibit limited variability, the number of singular vectors available for selection will be small, constraining the reduced dimension and, consequently, the accuracy of the hyper-reduced model. To achieve a stable and accurate ROM, it is essential to ensure adequate variation in the DEIM snapshots, allowing for a larger set of singular vectors and a robust reduced representation.

### 4.1.3 Variants of the DEIM algorithm

Over time, several (D)EIM-inspired alogorithms have been developed [204–210] to address specific challenges and improve performance. Notable among these are Q-DEIM, Localized

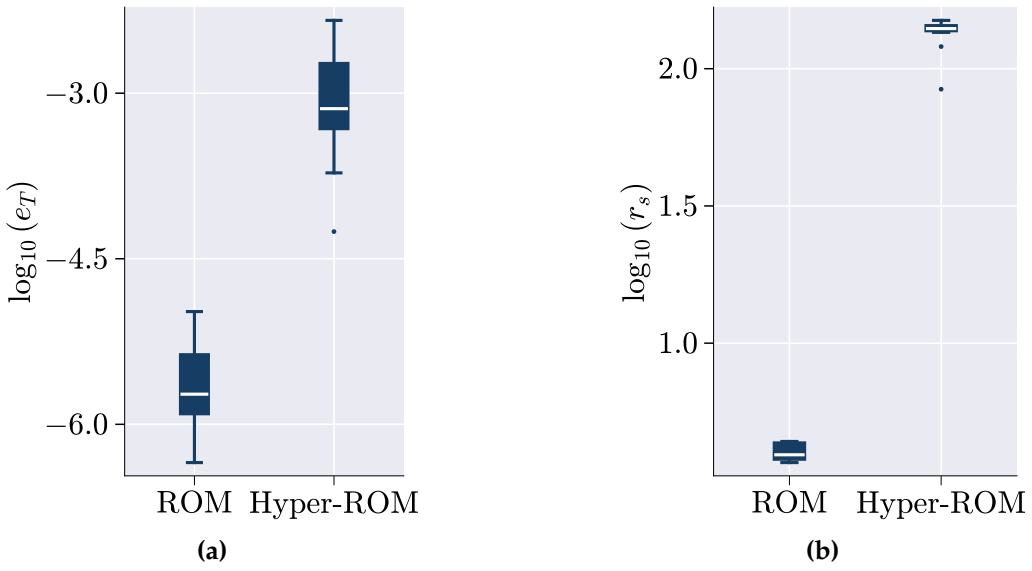



**Figure 14:** Difference in accuracy ($e_T$) and speed-up ($r_s$) between the regular projection based ROM and DEIM-based Hyper-ROM.

DEIM (LDEIM), Unassembled DEIM (U-DEIM), and S-OPT.

**Q-DEIM** The Q-DEIM (QR-based Discrete Empirical Interpolation Method) [82] improves upon the traditional DEIM algorithm by using QR decomposition with column pivoting to select interpolation points. A key numerical linear algebra question motivating Q-DEIM is whether one can construct an sampling matrix $Z$ that ensures the condition number of the DEIM projection remains moderate, regardless of the chosen orthonormal basis $\widetilde{\mathbb{U}}_f$. Traditional DEIM is sensitive to the specific basis of singular vectors, especially when singular values are closely clustered or repeated, which can lead to instability and variable interpolation quality. Q-DEIM addresses this issue by determining interpolation points independently of any specific orthonormal basis through QR factorization with column pivoting. This approach provides a priori assurance of a moderate condition number for the DEIM projection, enhancing numerical stability and reducing errors. As a result, Q-DEIM offers a robust and efficient method for constructing the interpolation matrix $S$, making it particularly suitable for applications like sensor placement in sparse interpolation contexts where reliable interpolation point selection is crucial.

**LDEIM** Localized DEIM (LDEIM), enhances the DEIM approach by employing multiple local subspaces instead of a single global subspace for approximating the nonlinear term [83]. This method is particularly advantageous in problems where the nonlinear term varies greatly over different regions. LDEIM accomplishes this by using machine learning-based clustering algorithms to discover the regions that should form each local subspace. During the online phase, classification-based machine learning methods are then used to adaptively

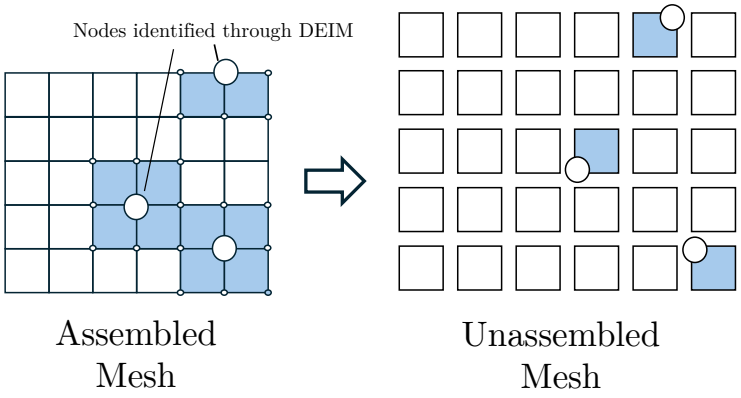



**Figure 15:** Unsassembled DEIM requires less elements to be sampled than the traditional DEIM algorithm for Finite Element problems. (Adapted from [157])

select the appropriate local subspace for the DEIM approximation. While the additional computational overhead of LDEIM may not be justified for problems where the nonlinear term behaves relatively uniformly, it has been demonstrated to provide substantial benefits in cases with large variation in the nonlinear term. In such scenarios, LDEIM can achieve speedups of up to two orders of magnitude compared to traditional DEIM algorithms.

**U-DEIM** Unassembled DEIM (U-DEIM) improves the applicability of DEIM to simulations using the Finite Element Method (FEM) [157]. It can be difficult to efficiently apply DEIM to FEM simulations because nodes in FEM can be shared by multiple elements (as was seen for the running example as well). If a node is selected with the typical DEIM algorithm, it will require the contributions of all adjacent elements to be calculated. This greatly increases the number of elements required, which decreases the efficiency of DEIM. U-DEIM solves this problem by applying the DEIM approximation to the unassembled form of the FEM system. This means only the contributions from one element are required to calculate the DEIM approximation.

The difference between the assembled and unassembled DEIM is illustrated in Fig. 15, where there is a significant decrease in the number of elements included in the U-DEIM case. After the DEIM approximation is calculated, the force matrix can be assembled in the same way it is done in typical FEM. U-DEIM is more efficient to calculate for FEM problems than the traditional DEIM method. However, U-DEIM comes with a trade-off: the system matrix assembled with U-DEIM is not symmetric, which leads to instability in the solution of the nonlinear system. There are strategies to mitigate this instability, but they do not completely eliminate it.

**S-OPT** S-OPT, or S-optimality method was originally introduced by Shin and Xiu to obtain quasi-optimal sample set for ordinary least squares regression, providing regression results comparable to those obtained from substantially larger candidate sets [211]. Its

application to hyper-reduction was later explored in [196].

Similar to DEIM, S-OPT aims to construct a sampling matrix $\mathsf{Z}$, which reduces the function approximation error described in Eq. (57) by reducing $\| \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger \|$.

Recalling Eq. (52), we write:

$$\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)^\dagger = \Omega^{-1} \mathsf{Z}^\top \widetilde{\mathbb{U}}_f \,, \tag{58}$$

where $\Omega = \left(\widetilde{\mathbb{U}}_f^\top \mathsf{Z}\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$. As discussed in Section 4.1.1, the magnitude of the pseudo-inverse in Eq. (58) decreases when (a) the determinant of $\Omega$ is large in mangitude, which will ensure its invertibility and (b) the columns of $\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$ are orthogonal.

The S-OPT achieves both objectives simultaneously by constructing a $\mathsf{Z}$ that maximizes a scalar function $\mathcal{S}$ of $\left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_f\right)$, defined as

$$\mathcal{S}(\mathsf{P}) = \left( \frac{\sqrt{|\det(\mathsf{P}^\top \mathsf{P})|}}{\Pi_{i=1}^{r} \|\mathsf{P}[:,i]\|} \right)^{\frac{1}{r}} \,, \tag{59}$$

where $\mathsf{P} = \mathsf{Z}^\top \mathsf{Q}_f$, and $\mathsf{Q}_f$ is obtained from the QR decompostion of $\widetilde{\mathbb{U}}_f = \mathsf{Q}_f \mathsf{R}$. Here $\mathsf{P}[:,i]$ denotes the $i$th column of $\mathsf{P}$. The maximization problem is written as

$$\mathsf{Z}^*_{\text{S-OPT}} = \underset{\mathsf{Z} \in \mathcal{R}^{N \times n}}{\arg\max}\, \mathcal{S}(\mathsf{Z}^\top \mathsf{Q}) \,. \tag{60}$$

The S-OPT algorithm, detailed in Algorithm 3 yields a quasi-optimal solution to this maximization problem.

Using Hadamard's inequality [212], it can be shown that $\mathcal{S} \in [0,1]$. The inequality states that for any symmetric matrix $\mathsf{M} \in \mathcal{R}^{p \times p}$

$$|\det(\mathsf{M})| \leq \prod_{i=1}^{p} M_{ii} \,, \tag{61}$$

where $M_{ii}$ denotes the $i$th diagonal entry of $\mathsf{M}$ and $\det(\mathsf{M})$ denotes its determinant. The equality holding only when the columns of $\mathsf{M}$ are orthogonal. For $\mathcal{S}$, we substitute $\mathsf{M} = \mathsf{P}^\top \mathsf{P}$.

### 4.1.4 Gauss–Newton with approximated tensors (GNAT)

---

- **Objective:** Reduce the computational complexity of minimizing residual errors when numerically solving a reduced nonlinear spatio-temporal PDE at each time step.

- In a Galerkin setting, residual originating from a trial reduced solution is rendered orthogonal to the subspace containing the reduced PDE solution. This orthogonality condition gives rise to ordinary differential equations (ODEs), which are subsequently solved using the method of lines (time is treated as a continuous variable).
- Alternatively, in a time-discrete setting, the residual may also be minimized in an $L_2$ sense at each time step, effectively resulting in a Petrov-Galerkin projection of the residual.
- This nonlinear minimization problem requires iterative evaluation of the residual and its Jacobian at every time step when solved using Gauss-Newton method.
- GNAT, like DEIM, constructs a sampling matrix that selectively samples the residual and its Jacobian. These sampled values are then used to approximate the residual and Jacobian via interpolation, thereby facilitating more efficient computation.

---

As outlined in Section 2.3, when reducing Eq. (1) through the Galerkin approach, the residual due to the approximation $\mathbf{w}_N \approx \mathbf{w}_N^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n$ is enforced to be orthogonal to $\widetilde{\mathbb{U}}$. This leads to a reduced set of ODEs, which are then solved for $\mathbf{w}_n$, as shown in Eq. (16). However, It can be shown that Galerkin projection essentially yields a $\dot{\mathbf{w}}_n$ (*not* $\mathbf{w}_n$) that minimizes the $L_2$ norm of $\mathbf{r}_N$ over $t$ [123].

Alternatively, in a discrete time setting, implicit or explicit, $\mathbf{w}_n$ can also be determined by *minimizing* the $L_2$ norm of the residual at each time step. For example, the residual associated with Eq. (1), when solved using the Backward-Euler integration scheme, takes the following form:

$$\mathbf{r}_N^{k+1} = \mathsf{M}_N(\boldsymbol{\mu})\widetilde{\mathbb{U}}\left(\mathbf{w}_n^{k+1} - \mathbf{w}_n^k\right) + \Delta t\, \mathbf{f}_N^{k+1}(\mathbf{w}^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n^{k+1}, \mu) - \Delta t\, \mathbf{g}_N^{k+1}\,, \tag{62}$$

where $\mathbf{w}_n^{k+1} \triangleq \mathbf{w}_N(t_{k+1}; \mu)$. We intend to calculate $\mathbf{w}_n^{k+1}$ that minimizes $\mathbf{r}_N$ by solving the following minimization problem:

$$\mathbf{w}_n^{k+1} = \arg\min_{\mathbf{w}_n^{k+1}} \|\mathbf{r}_N^{k+1}(\mathbf{w}_n^{k+1})\|\,. \tag{63}$$

The nonlinear least-square problem in Eq. (63) can equivalently be written as a minimization of a scalar function $\phi : \mathcal{R}^N \to \mathcal{R}$. Dropping the suffix $N$ we write:

$$\phi(\mathbf{z}) = \frac{1}{2}\|\mathbf{r}(\mathbf{z})\|^2 = \frac{1}{2}\mathbf{r}(\mathbf{z})^\top \mathbf{r}(\mathbf{z})\,, \tag{64}$$

where $\mathbf{z} \triangleq \mathbf{w}_n^{k+1}$. The minimization of $\phi$ requires:

$$\nabla\phi(\mathbf{z}) = 2 \cdot \frac{1}{2}\mathsf{J}^\top(\mathbf{z})\mathbf{r}(\mathbf{z}) = 0\,, \tag{65a}$$

where

$$\mathsf{J} = \frac{\partial \mathbf{r}}{\partial \mathbf{z}} = \left(\mathsf{M}(\boldsymbol{\mu}) + \Delta t\, \mathsf{J}_{\mathbf{f}}^{k+1}\right) \widetilde{\mathbb{U}}\,, \tag{65b}$$

and $\mathsf{J}_{\mathbf{f}}^{k+1} = \left.\dfrac{\partial \mathbf{f}_N}{\partial \mathbf{w}_\mathbf{N}}\right|_{t=t_{k+1}}$.

Note that, unlike the Galerkin method, the residual in Eq. (65a), is always orthogonal to the columns of J, which corresponds to a Petrov-Galerkin (PG) projection [213, 214] (see Fig. 23a), as in Eq. (17), where the left ROB is J. The PG approach is particularly useful for problems where the Jacobian matrix is not symmetric as in the HFM obtained from the Navier-Stokes equations.

The *root-finding problem* in Eq. (65a) can be solved using the *Gauss-Newton* method [80, 184] iteratively. Considering $\mathbf{z}_j$ to be the approximation for the $j$th Gauss-Newton iteration, we write the $(j+1)$th approximation as:

$$\mathbf{z}_{j+1} = \mathbf{z}_j + \boldsymbol{\Delta}_{j+1}\,, \tag{66}$$

where

$$\nabla^2\phi(\mathbf{z}_j)\boldsymbol{\Delta}_{j+1} = -\nabla\phi(\mathbf{z}_j)\,. \tag{67}$$

The Hessian matrix $\nabla^2\phi(\mathbf{z}_j)$ is given by

$$\nabla^2\phi(\mathbf{z}_j) = \mathsf{J}(\mathbf{z})^\top \mathsf{J}(\mathbf{z}) + \sum_{i=1}^{N} \frac{\partial^2 r_i(\mathbf{z})}{\partial \mathbf{z}^2} r_i(\mathbf{z})\,. \tag{68}$$

The Gauss-Newton method approximates the Hessian of $\phi$ by using only the first term, $\mathsf{J}(\mathbf{z})^\top \mathsf{J}(\mathbf{z})$ and neglects the second term yielding the following equation:

$$\mathsf{J}(\mathbf{z}_j)^\top \mathsf{J}(\mathbf{z}_j)\boldsymbol{\Delta}_{j+1} = -\mathsf{J}(\mathbf{z}_j)^\top \mathbf{r}(\mathbf{z}_j)\,. \tag{69}$$

A QR decomposition of $\mathsf{J}(\mathbf{z})$ yields (assuming $\mathsf{R}$ is invertible)

$$\boldsymbol{\Delta}_{j+1} = -\mathsf{R}^{-1}(\mathbf{z}_j)\mathsf{Q}(\mathbf{z}_j)\mathbf{r}(\mathbf{z}_j)\,. \tag{70}$$

We note that Eq. (69) is also the normal equation of the optimization problem

$$\boldsymbol{\Delta}_{j+1} = \arg\min_{\mathbf{q}} \|\mathsf{J}(\mathbf{z}_j)\mathbf{q} + \mathbf{r}(\mathbf{z}_j)\|\,. \tag{71}$$

Although the dimension of the reduced subspace $\widetilde{\mathbb{U}}$ is small, the computational cost of iteratively evaluating $\mathbf{r}(\mathbf{z}_j)$ and $\mathsf{J}(\mathbf{z}_j)$ scale with the full-space dimension $N$. This creates a computational bottleneck for the projection-based model reduction methods. The purpose of hyper-reduction is to mitigate this computational cost.

We assume that $\mathbf{r}(\mathbf{z}_j)$ and the columns of $\mathsf{J}(\mathbf{z}_j)$ lie within the low-dimensional subspaces spanned by the columns of $\widetilde{\mathbb{U}}_R \in \mathcal{R}^{N\times m_R}$ and $\widetilde{\mathbb{U}}_J \in \mathcal{R}^{N\times m_J}$, respectively (how to determine these is discussed later). GNAT uses gappy POD to approximate $\mathbf{r}(\mathbf{z}_j)$ and $\mathsf{J}(\mathbf{z}_j)$ as follows:

$$\widetilde{\mathbf{r}}(\mathbf{z}_j) = \arg\min_{\mathbf{x}} \|\mathsf{Z}^\top \mathbf{r}(\mathbf{z}_j) - \mathsf{Z}^\top \widetilde{\mathbb{U}}_R \mathbf{x}\|\,, \tag{72}$$

$$\widetilde{\mathsf{J}}(\mathbf{z}_j) = \arg\min_{\mathsf{X}} \|\mathsf{Z}^\top \mathsf{J}(\mathbf{z}_j) - \mathsf{Z}^\top \widetilde{\mathbb{U}}_J \mathsf{X}\|_F\,, \tag{73}$$

where, as in Eq. (51), $\mathsf{Z}$ is a sampling matrix, $\mathbf{x} \in \mathcal{R}^{m_R\times 1}$, and $\mathsf{X} \in \mathcal{R}^{m_J\times m_J}$. These approximations minimize the errors at the sample indices defined by $\mathsf{Z}^\top$. The solutions to these linear least-squares problems are:

$$\widetilde{\mathbf{r}}(\mathbf{z}_j) = \widetilde{\mathbb{U}}_R \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_R\right)^\dagger \mathsf{Z}^\top \mathbf{r}(\mathbf{z}_j)\,, \tag{74a}$$

$$\widetilde{\mathsf{J}}(\mathbf{z}_j) = \widetilde{\mathbb{U}}_J \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_J\right)^\dagger \mathsf{Z}^\top \mathsf{J}(\mathbf{z}_j)\,. \tag{74b}$$

The symbol † indicates the Moore-Penrose pseudo-inverse.

The reduced subspaces $\widetilde{\mathbb{U}}_R$ and $\widetilde{\mathbb{U}}_J$ are computed by applying singular value decomposition (SVD) to the snapshots of the residual and the columns of the Jacobian matrix, respectively, at each time step *for each Gauss-newton iteration* during the high-fidelity training data generation for a set of parameters. To ensure that the interpolation problems in Eq. (74) are well-posed, we enforce $m_R \geq \mathcal{C}(\mathsf{Z})$ and $m_J \geq \mathcal{C}(\mathsf{Z})$, where $\mathcal{C}(\mathsf{Z})$ denotes the cardinality of $\mathsf{Z}$, the number of nonzero entries of $\mathsf{Z}$, which indicates the number of points selected.

Substituting $\widetilde{\mathsf{J}}(\mathbf{z}_j)$ and $\widetilde{\mathbf{r}}(\mathbf{z}_j)$ for $\mathsf{J}(\mathbf{z}_j)$ and $\mathbf{r}(\mathbf{z}_j)$ while noticing $\widetilde{\mathbb{U}}_J^\top \widetilde{\mathbb{U}}_J = \mathsf{I}_{m_J}$, Eq. (71) can be transformed into a hyper-reduced, linear, least-squares minimization problem:

$$\tilde{\mathbf{\Delta}}_{j+1} = \arg\min_{\mathbf{q}} \|\mathcal{P}\mathsf{J}(\mathbf{z}_j)\mathbf{q} + \mathcal{Q}\mathbf{r}(\mathbf{z}_j)\|\,, \tag{75}$$

where $\mathcal{P} = \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_J\right)^\dagger \mathsf{Z}^\top$ and $\mathcal{Q} = \widetilde{\mathbb{U}}_J^\top \widetilde{\mathbb{U}}_R \left(\mathsf{Z}^\top \widetilde{\mathbb{U}}_R\right)^\dagger \mathsf{Z}^\top$.

As with Eqs. (69) and (70), this can be solved using the QR decomposition of $\mathcal{P}\mathsf{J}(\mathbf{z}_j)$ so that:

$$\tilde{\mathbf{\Delta}}_{j+1} = -\widetilde{\mathsf{R}}^{-1}(\mathbf{z}_j)\widetilde{\mathsf{Q}}(\mathbf{z}_j)\widetilde{\mathbb{U}}_J^\top \widetilde{\mathbf{r}}(\mathbf{z}_j)\,. \tag{76}$$

The solution $\tilde{\mathbf{\Delta}}_{j+1}$ is substituted for $\mathbf{\Delta}_{j+1}$ in Eq. (66). For $m_J \geq n$, Eq. (76) yields a unique solution. We note that both $\mathcal{P}$ and $\mathcal{Q}$ can be precomputed during the offline phase to reduce computational cost in the online phase. The algorithm for computing the selection matrix $\mathsf{Z}$ is described in Appendix B.

The *online stage* utilizes the precomputed data (such as $\mathcal{P}$, $\mathcal{Q}$) from the offline phase to perform reduced-order model simulations for new input parameters. The reduced sample meshes and precomputed matrices enable fast and efficient computations of the nonlinearities. After the simulation, post-processing is performed to compute the desired outputs based on the reduced-order model results.

**Remark 8** In GNAT, the ROM solution subspace $\widetilde{\mathbb{U}}$ is obtained by applying SVD to the set of snapshots $\{\mathbf{w}_N^k(\mu) - \mathbf{w}_N^0(\mu) \mid k = 1, \ldots, n_t, \mu \in \mathcal{D}_{\text{train}}\}$, not to $\mathbf{w}_N^k(\mu)$ directly.

**Remark 9** In practice, one can set $\widetilde{\mathbb{U}}_R = \widetilde{\mathbb{U}}_J$, and this subspace is computed by applying SVD only to the stored snapshots of $\mathbf{r}(\mathbf{z}_j)$ at different time steps and parameter values. This avoids storing the Jacobian matrix iteratively, reducing memory requirements.

**Remark 10** GNAT is specific to Petrov-Galerkin reduced-order models (ROMs) and is primarily applicable to time-dependent problems. Since the running problem is steady-state and purely parametric by nature, the GNAT method has not been discussed in this context.

## 4.2 Project-then-approximate hyper-reduction strategies

The project-then-approximate (PA) hyper-reduction methods approximate the *projected* high-dimensional vectors and matrices derived from high-fidelity models. These methods aim to surpass the performance of the approximate-then-project approach. Similar to the AP methods, this class reduces the mesh by sampling from a highly discretized mesh associated with high-dimensional models, enabling faster evaluation of the projected matrices and nonlinear terms. This section explores two widely used PA hyper-reduction techniques: Energy Conserving Sampling and weighing (ECSW) and Empirical Cubature Methods (ECM). In Fig. 16, we show a brief evolution of various project-then-approximate hyper-reduction schemes since 2009.

### 4.2.1 Energy conserving sampling and weighing (ECSW)

---

- **Objective:** Construct a reduced mesh from a highly discretized HDM to accelerate projected nonlinear term calculations.
- Preserve the total virtual work performed by internal forces on displacements induced by the reduction basis while sampling the mesh.
- Compute residual internal forces for each element and project them onto the reduction basis to obtain virtual work.
- Sample and weight elements sparsely to maintain approximate virtual work consistency, forming the reduced mesh.
- Use the reduced mesh for faster evaluation of projected nonlinear terms and matrices.

---

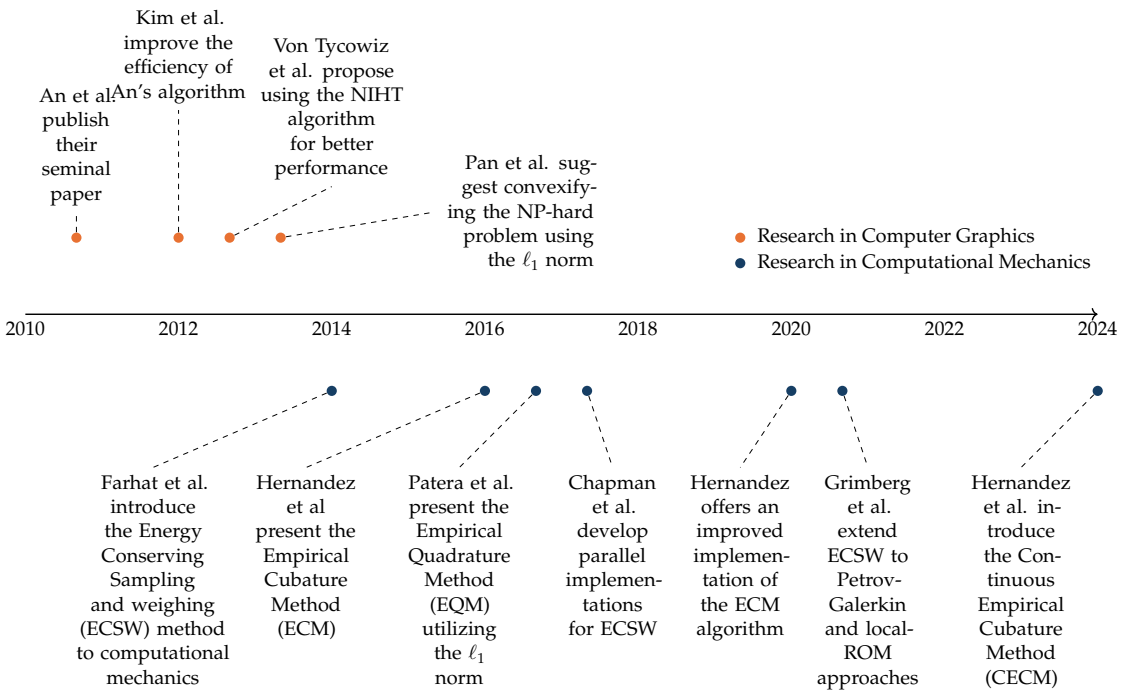



**Figure 16:** A brief history of the development of the most important cubature rules/mesh sampling procedures [31, 55, 77, 78, 195, 197, 215–218]. Adapted from [76].

Energy Conserving Sampling and weighting (ECSW) technique, developed by Farhat et al [31], generates a reduced mesh that can estimate the projected nonlinear vector $\mathbf{f}_n(\mathbf{w}_N(t;\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbb{W}^\top \mathbf{f}_N(\mathbf{w}^{ref} + \widetilde{\mathbb{U}}\mathbf{w}_n;\boldsymbol{\mu})$ and the projected parametric vector $\mathbf{g}_n(t;\mu) = \mathbb{W}^\top \mathbf{g}_N(t;\boldsymbol{\mu})$, as described in Eq. (21), where $\widetilde{\mathbb{U}}, \mathbb{W} \in \mathcal{R}^{N\times n}$ are the right and left ROB, respectively.

We introduce a balance vector $\mathbf{b}_n \in \mathcal{R}^n$ comprising both $\mathbf{f}_N$ and $\mathbf{g}_N$, defined as

$$\mathbf{b}_n(\mathbf{w}_n;\boldsymbol{\mu}) = \mathbf{f}_n(\mathbf{w}_n;\boldsymbol{\mu}) - \mathbf{g}_n(t;\boldsymbol{\mu})\,. \tag{77}$$

Considering that the domain of the high-fidelity model (finite element/volume) contains $n_{\text{cell}}$ cells (nodes for finite difference models), $\mathbf{b}_n$ can be written as (See Eqs. 30 and 31 for reference)

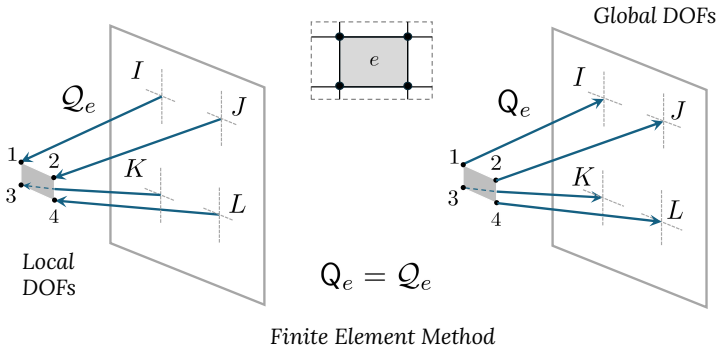
$$\mathbf{b}_n(\mathbf{w}_n;\mu) = \sum_{e=1}^{n_{\text{cell}}} \mathbb{W}_e^\top \mathbf{b}_e(\mathcal{Q}_e\mathbf{w}^{ref} + \widetilde{\mathbb{U}}_e\mathbf{w}_n;\boldsymbol{\mu})\,, \tag{78}$$

where

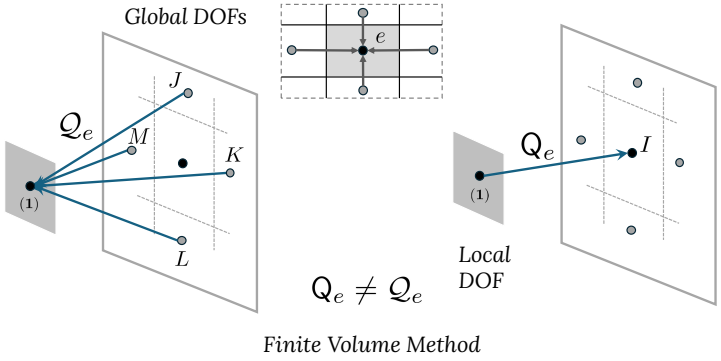
$$\mathbf{b}_e(\mathcal{Q}_e\mathbf{w}^{ref} + \widetilde{\mathbb{U}}_e\mathbf{w}_n;\boldsymbol{\mu}) = \mathbf{f}_e(\mathcal{Q}_e\mathbf{w}^{ref} + \widetilde{\mathbb{U}}_e\mathbf{w}_n;\boldsymbol{\mu}) - \mathbf{g}_e(t;\boldsymbol{\mu}) \tag{79}$$

is the elemental balance vector with $\mathbf{b}_e, \mathbf{f}_e, \mathbf{g}_e \in \mathcal{R}^{d_e}$ ($d_e$ denotes the elemental degrees of freedom) and

$$\mathbb{W}_e = \mathbb{Q}_e\mathbb{W} \tag{80a}$$




**(a)**




**(b)**

**Figure 17:** Illustration of the difference between the mapping operators $\mathcal{Q}$ (global-to-local) and $\mathsf{Q}$ (local-to-global) in the 2D Finite Element Method (FEM) and 2D Finite Volume Method (FVM), assuming one quantity of interest per cell-center or node. (a) In FEM, local DOFs are defined at element nodes. $\mathcal{Q}_e$ extracts the global DOFs (denoted as $I$, $J$, $K$, $L$) associated with element $e$, while $\mathsf{Q}_e$ distributes local DOFs (1-4) back into the global. Due to this one-to-one nature of the local-global mapping $\mathcal{Q}_e = \mathsf{Q}_e$. (b) In FVM, local DOFs correspond to cell centers, with flux evaluations at cell faces involving neighboring cells ($J$, $K$, $L$, $M$). $\mathcal{Q}_e$ extracts global cell-centered values, and $\mathsf{Q}_e$ maps the computed local DOF back into the global system. While information from multiple cell centers is used to evaluate the quantity at a given cell center, each cell has exactly one equation, and its evaluated quantity corresponds to a single global cell center. Hence, $\mathcal{Q}_e \neq \mathsf{Q}_e$.

$$\widetilde{\mathbb{U}}_e = \mathcal{Q}_e \widetilde{\mathbb{U}}\,. \tag{80b}$$

Matrix $\mathbb{Q}_e$, as in Section 3, maps the local degrees of freedom of the cell $e$ to the global cell DOFs over the entire domain, and matrix $\mathcal{Q}_e$ depends on the connectivity of cell $e$ with its neighboring cells that contribute to evaluating $\mathbf{b}_e$. As a result, $\mathbb{Q}_e$ and $\mathcal{Q}_e$ may differ depending on whether the model is a finite element/volume/difference method; for finite element model $\mathcal{Q}_e = \mathbb{Q}_e$ (refer to Fig. 17). In essence, $\mathbb{W}_e$ and $\widetilde{\mathbb{U}}_e$ denote the portions of $\mathbb{W}$ and $\widetilde{\mathbb{U}}$ defined over the $e$th element.

The projected balance vector $\mathbf{b}_n$ in Eq. (78) may be physically interpreted as an expression of virtual work done by an elemental balance "force" vector $\mathbf{b}_e$ onto the virtual "displacements" of the elemental DOFs, given by the columns of $\mathbb{W}_e$.

In ECSW, one aims to *approximate* $\mathbf{b}_n$ using a *weighted sum* of the projected $\mathbf{b}_e$'s on the right hand side of Eq. (78), using fewer elements than $n_{\text{cell}}$:

$$\mathbf{b}_n(\mathbf{w}_n;\boldsymbol{\mu}) \approx \sum_{e=1}^{n_{\text{cell}}} \xi_e \mathbb{W}_e^\top \mathbf{b}_e(\mathcal{Q}_e \mathbf{w}^{ref} + \widetilde{\mathbb{U}}_e \mathbf{w}_n;\boldsymbol{\mu})\,, \tag{81}$$

where $\xi_e$ is the weight associated with the $e$th element, contained in the weight vector $\boldsymbol{\xi} = [\xi_1, \xi_2, ..., \xi_{n_{\text{cell}}}]^\top \in \mathcal{R}^{n_{\text{cell}}\times 1}$. We aim to make $\boldsymbol{\xi}$ as sparse as possible so that the elemental contribution $\mathbf{b}_e$ is evaluated only for the nonzero entries, thereby reducing the mesh elements. To ensure positive definiteness of the reduced mass or stiffness matrices on this reduced mesh, we impose the constraint $\xi_e \geq 0$. Next, we describe a mathematical framework for calculating the sparse $\boldsymbol{\xi}$ vector.

### Determining $\boldsymbol{\xi}$

During the offline stage, high-fidelity snapshots at various parameter values are projected onto the reduced space $\widetilde{\mathbb{U}}$ to compute reduced balance vectors $\mathbf{b}_n$.

For the $k$th parameter-time pair $(\boldsymbol{\mu}_k, t_k)$, the projected solution $\widetilde{\mathbf{w}}_n$ is defined as

$$\widetilde{\mathbf{w}}_n(\boldsymbol{\mu}_k, t_k) = \widetilde{\mathbb{U}}^\top \left(\mathbf{w}_N(t_k;\boldsymbol{\mu}_k) - \mathbf{w}^{ref}\right)\,, \tag{82}$$

which is used in place of $\mathbf{w}_n$ in Eq. (81) to evaluate $\mathbf{b}_n$. We write from Eq. (78)

$$\mathbf{b}_n^k(\widetilde{\mathbf{w}}_n(t_k;\boldsymbol{\mu}_k);\boldsymbol{\mu}_k) = \sum_{e=1}^{n_{\text{cell}}} \gamma_n^{k,e}, \tag{83}$$

where $\mathbf{b}_n^k$ is the balance vector for the $k$th parameter and

$$\gamma_n^{k,e} = \mathbb{W}_e^\top \mathbf{b}_e(\mathcal{Q}_e \mathbf{w}^{ref} + \widetilde{\mathbb{U}}_e \widetilde{\mathbf{w}}_n(t_k;\boldsymbol{\mu}_k))\,. \tag{84}$$

Note that here time $t$ is considered as a parameter as well.

We rewrite Eq. (83) as:

$$\mathbf{b}_n^k = \bar{\mathsf{G}}^k \, \mathbf{1}\,, \tag{85}$$

where $\bar{\mathsf{G}}^k \in \mathcal{R}^{n \times n_{\text{cell}}}$ and $\mathbf{1} \in \mathcal{R}^{n_{\text{cell}} \times 1}$ are defined as:

$$\bar{\mathsf{G}}^k = \left[\gamma_n^{k,1},\ \gamma_n^{k,2},\ \ldots,\ \gamma_n^{k,n_{\text{cell}}}\right], \tag{86a}$$

$$\mathbf{1} = \left[1,\ 1,\ \ldots,\ 1\right]^\top. \tag{86b}$$

If $N_s$ solution snapshots are generated for $N_s$ pairs of $\mu$ and $t$, Eq. (85) can be extended to:

$$\mathbf{\Gamma} = \mathsf{G}\,\mathbf{1}\,, \tag{87}$$

where $\mathsf{G} \in \mathcal{R}^{nN_s \times n_{\text{cell}}}$ (typically $nN_s < n_{\text{cell}}$) and $\mathbf{\Gamma} \in \mathcal{R}^{nN_s \times 1}$ are defined as:

$$\mathbf{\Gamma} = \left[\mathbf{b}_n^{1\,\top}, \mathbf{b}_n^{2\,\top}, \ldots, \mathbf{b}_n^{N_s\,\top}\right]^\top, \tag{88a}$$

$$\mathsf{G} = \left[\bar{\mathsf{G}}^{1\,\top}, \bar{\mathsf{G}}^{2\,\top}, \ldots, \bar{\mathsf{G}}^{N_s\,\top}\right]^\top. \tag{88b}$$

Our goal is to replace the vector $\mathbf{1}$ in Eq. (87), which assigns equal weight to the elemental contributions, with a sparse weighing vector $\boldsymbol{\xi}$ such that:

$$\mathbf{\Gamma} \approx \mathsf{G}\,\boldsymbol{\xi}, \tag{89}$$

implying:

$$\mathbf{b}_n^k \approx \sum_{e=1}^{n_{\text{cell}}} \xi_e \, \gamma_n^{k,e}\,. \tag{90}$$

To obtain a sparse $\boldsymbol{\xi}$, we ideally want to minimize its zero norm, which indicates the number of non-zero elements in it:

$$\boldsymbol{\xi}^* = \arg \min_{\boldsymbol{\xi} \in \mathcal{R}^{n_{\text{cell}}},\ \boldsymbol{\xi} \geq 0} \|\boldsymbol{\xi}\|_0 \quad \text{subject to} \quad \|\mathsf{G}\,\boldsymbol{\xi} - \mathbf{\Gamma}\| \leq \epsilon \|\mathbf{\Gamma}\|\,, \tag{91}$$

where $\epsilon$ is some pre-defined tolerance. However, this problem is NP-hard [31]. Therefore, we use approximations such as non-negative $L_1$-norm minimization, non-negative $L_2$-norm minimization with $L_1$-norm regularization, or non-negative least squares (NNLS).

Among these, NNLS [219] is most commonly employed for ECSW. NNLS solves the linear least squares problem under the constraint that the solution must be non-negative:

$$\boldsymbol{\xi}^* = \arg \min_{\boldsymbol{\xi} \in \mathcal{R}^{n_{\text{cell}}},\ \boldsymbol{\xi} \geq 0} \|\mathsf{G}\,\boldsymbol{\xi} - \mathbf{\Gamma}\|^2\,. \tag{92}$$

While NNLS does not explicitly promote sparsity, it often results in a sparse $\boldsymbol{\xi}$ in practice, providing an acceptable approximation of $\mathbf{\Gamma}$ with a non-negative and sparse solution.

Once a sparse vector $\boldsymbol{\xi}$ is obtained, during the *online phase*, elemental quantities, such as stiffness, are computed only for elements with nonzero weights, which substantially improves the efficiency of computing the reduced quantities.

**Remark 11** For a dense mesh with very large number of elements, NNLS computations can become very slow. An effective approach to speed up NNLS computations in Eq. (92) is by parallelizing the NNLS algorithm via domain decomposition, thereby distributing the workload among multiple processors. This method involves dividing the finite element (FE) mesh into subdomains, effectively partitioning the matrix $\mathsf{G}$ column-wise. By computing $\boldsymbol{\Gamma}$ for each column partition and employing the NNLS solver on each subdomain independently, this technique minimizes offline computation costs.

**Remark 12** We observed that the NNLS computations can be made more efficient by using a low-rank approximation of $\mathsf{G}$ through its SVD. By decomposing $\mathsf{G} = \mathsf{U}_r \mathsf{\Sigma}_r \mathsf{V}_r^\top$, where $\mathsf{U}_r$, $\mathsf{\Sigma}_r$, and $\mathsf{V}_r$ correspond to the top $r$ singular values, we can reformulate the objective function in Eq. (92) as $\|\mathsf{V}_r^\top \boldsymbol{\zeta} - \mathbf{d}_r\|^2$, where $\mathbf{d}_r = \mathsf{V}_r^\top \cdot \mathbf{1}$. This reduction improves computational efficiency and, most importantly, eliminates redundancy in the snapshot data.

**Remark 13** Parallel implementations of NNLS in `C++` can be found in libraries such as `libROM` [85]. Such implementations are suitable for handling large-scale problems by utilizing parallel computing architectures.

**Remark 14** Energy-Conserving Sampling and weighing (ECSW) preserves the Lagrangian structure crucial for second-order hyperbolic problems governed by Hamilton's principle. This preservation ensures that time integrators unconditionally stable for Parametrically Reduced-Order Models (PROMs) remain stable when applied to the hyper-ROMs produced by ECSW [72].

**Remark 15** ECSW demonstrates superior numerical stability and accuracy in structural dynamics problems where other hyper-reduction methods, particularly those using the approximate-then-project approach, often fail—making it especially valuable for applications requiring robust numerical performance [72].

**Remark 16** Hyper-reduction can be performed either on individual reduced-order terms or collectively on multiple terms. For instance, in Eq. (77), rather than combining $\mathbf{f}_n$ and $\mathbf{g}_n$ into the balance vector $\mathbf{b}_n$, hyper-reduction could have been applied separately to each term. However, this would result in multiple reduced meshes making the process computationally inefficient. Therefore, performing hyper-reduction collectively on multiple reduced-order terms, which generates only a single reduced mesh is more efficient and is generally preferred [55].

### Implementation on the running example

As before, we use the same training and test datasets—each consisting of 16 parameter pairs $(\mu, \beta)_k$ for $k = 1, \ldots, 16$—to formulate and evaluate the ECSW-based hyper-reduced order models (hyper-ROMs). Here, $\widetilde{\mathbb{U}}$, as in Section 3, has $n = 5$ columns.

In the offline phase, to compute a reduced mesh, we construct the matrix $\mathsf{G}$ by incorporating element-level contributions $\gamma_n^{k,e}$ from $k$ solution snapshots as in Eq. (83). For the $e$th element, the mean subtracted projection of the $k$th solution snapshot is given by

$$\widetilde{\mathbf{T}}_n^k = \widetilde{\mathbb{U}}^\top (\mathbf{T}_N^k - \overline{\mathbf{T}}_N)\,. \tag{93}$$

Following Eq. (81), we then write the element-level balance vector as

$$\gamma_n^{k,e} = \widetilde{\mathbb{U}}_e^\top \mathsf{K}_e^k \widetilde{\mathbf{T}}_e^k - \widetilde{\mathbb{U}}_e^\top \mathbf{q}_e^k\,, \tag{94}$$

where

$$\widetilde{\mathbf{T}}_e^k = \mathsf{Q}_e \overline{\mathbf{T}}_N + \widetilde{\mathbb{U}}_e \widetilde{\mathbf{T}}_n^k \tag{95}$$

denotes the segment of the projected full-order temperature vector associated with the degrees of freedom corresponding to element $e$, and $\mathsf{K}_e^k$, $\mathbf{q}_e^k$ denote the corresponding elemental stiffness matrix and source vector, respectively. Given $\mathsf{K}_e^k$, $\mathbf{q}_e^k$ are functions of $T$ (Eqs. 35, 36), these are also evaluated using $\widetilde{\mathbf{T}}_e^k$.

Using these snapshots alongside Equations (83) to (92), we determine the sparse elemental weight vector $\boldsymbol{\xi}$. This process results in the selection of only 11 elements out of the 5000 in the domain, as depicted in Fig. 18. The value of the objective function in Eq. (92) is approximately $10^{-9}$, indicating a successful minimization. This remarkable reduction in the number of mesh elements obviates the need to evaluate 5000 elemental stiffness matrices during every Newton-Raphson iteration.

In Fig. 19, the relative percentage error (Eq. 42a) and speed-up (Eq. 42b) of the hyper-ROM are compared with those of the regular projection-based ROM. Figure 19a demonstrates that the hyper-ROM achieves high accuracy, with a relative error of approximately 0.0001%. While this error is marginally higher than that of the ROM, the computational speed-up is remarkable—nearly 300$\times$ the speed-up of the regular ROM and almost 1000$\times$ compared to the HFM. This performance gain is almost an order of magnitude greater than the speed-up achieved through DEIM. We attribute this additional improvement to the high accuracy with which ECSW approximates the reduced quantities, which presumably results in a faster Newton iteration convergence compared to DEIM.

As a visual aid to the reader, in Fig. 20, we pictorially represent how ECSW reduces the computational cost of evaluating the reduced nonlinear stiffness matrix. Figure 20a depicts the derivation of the reduced stiffness matrix where elemental contributions are equally weighted. In contrast, Fig. 20b shows an approximate reduced stiffness in which the stiffness matrix at the bottom has a non-zero and non-unit weight of $w_i$, and the weight of the top element is zero. The resulting reduced stiffness matrix looks approximately similar to the actual reduced stiffness matrix; however, the cost associated with evaluating the top stiffness matrix is eliminated.

**Remark 17** The Non-Negative Least Squares (NNLS) algorithm was implemented in Python using the `scipy.optimize.nnls` function [220]. However, the standard version may be inefficient for large-scale problems.

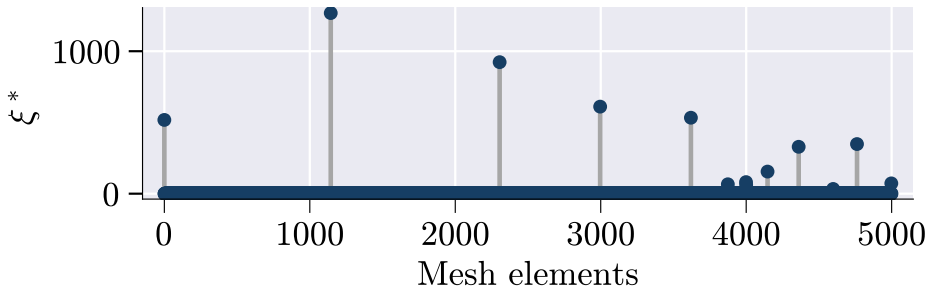



**Figure 18:** Stem plot illustrating the reduced mesh obtained via ECSW-based hyper-reduction. The plot highlights the mesh elements with non-zero weights $\xi^*$ (Eq. 92). The weights for all elements lying on the blue line are zero. The reduced mesh comprises only 11 elements, which constitutes 0.2% of the total 5,000 elements in the original mesh.

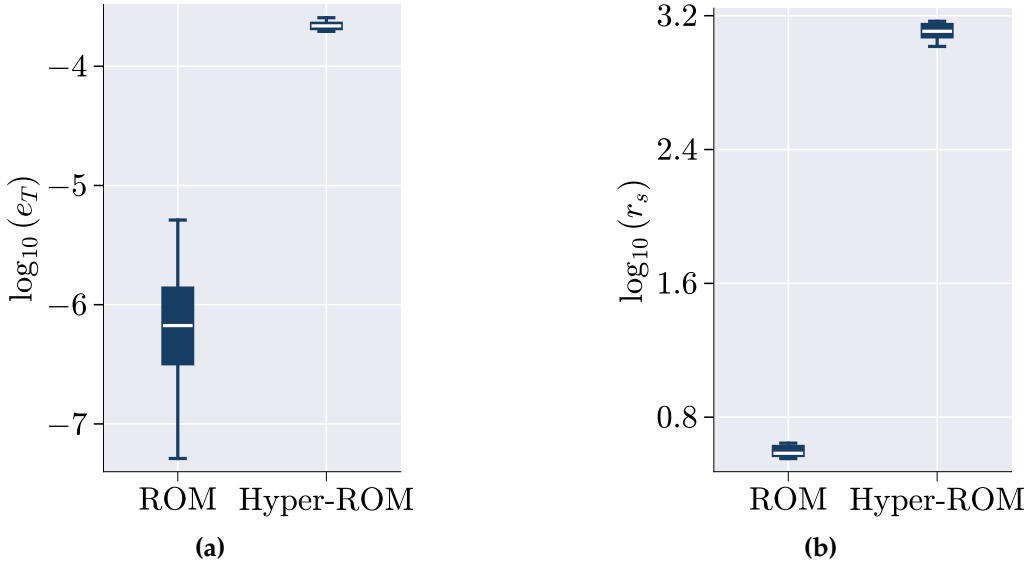



**Figure 19:** Difference in accuracy ($e_T$) and speed-up ($r_s$) between the regular projection based ROM and ECSW Hyper-ROM.

**Remark 18** In SciPy version 1.13.1, the `nnls` function was improved by integrating the fast Non-Negativity-Constrained Least Squares (fast-NNLS [221]) algorithm. This method reduces the per-iteration cost through precomputation, benefiting large-scale applications. The updated function also allows users to adjust error tolerance via the `atol` parameter. For the example problem, we implemented the `fast-NNLS` algorithm from `scipy 1.13.1` with a tolerance of $10^{-4}$.

**Remark 19** While performing NNLS, it is important to check that the magnitude of $\Gamma$ in Equation (92) is at least a few orders of magnitude larger than the tolerance. A small $\Gamma$ often

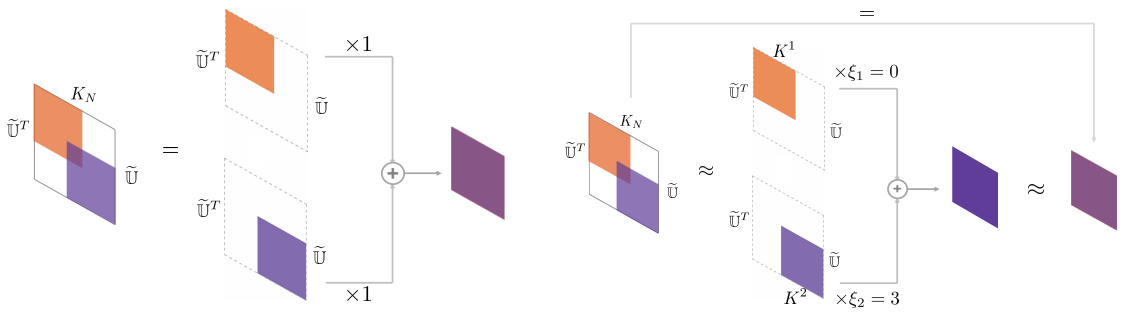



**Figure 20:** Reduction in the computational cost of evaluating the reduced nonlinear stiffness matrix achieved with ECSW during the iterative solution process. (a) Derivation of the reduced stiffness matrix with equally weighted elemental contributions without hyper-reduction. (b) Efficient *approximation* of the reduced stiffness matrix where the top element's weight $\xi_1$ is zero, and the bottom element has a much larger weight $\xi_2$, eliminating the cost of evaluating the top stiffness matrix during solution iteration.

leads to a dense weight vector $\boldsymbol{\xi}$ instead of sparse.

### 4.2.2 Empirical cubature method (ECM)

---

- **Objective:** Develop empirical cubature rules for accurately computing integrals in reduced-order equations obtained through inner products with reduced basis functions.
- The existence of low-dimensional structures in a system's solutions implies that the nonlinear integrands also exhibit low-dimensionality.
- Consequently, these integrals can be computed more efficiently using significantly fewer cubature points than those required by high-fidelity models.
- To compute projection integrals, evaluate the integrand over all selected cubature points across multiple training parameter instances and represent the exact integrals as a matrix-vector product.
- Construct a matrix encoding element contributions at selected cubature points and assemble a vector of Gauss weights. The integral is then computed as a matrix-vector product.
- Identify a subset of cubature points and adjust the associated weights to approximate the vector of exact integrals with desired accuracy.

The Empirical Cubature Method (ECM) [159] aims to efficiently approximate the integrals of a system's internal forces, which are crucial for computing system matrices in methods such as the Finite Element Method. The fundamental premise is that internal forces, like high-fidelity solutions, lie in a low-dimensional subspace independent of the finite element mesh size. This property enables the use of a cubature rule with significantly fewer points than conventional methods while maintaining accuracy.

**Problem formulation**

The Empirical Cubature Method (ECM) identifies cubature points using a limited set of high-fidelity training data, as seen in other methods. To illustrate the approach, we first outline its application for a general $n$-dimensional parameterized vector-valued function $\mathbf{a}$, and subsequently demonstrate its use in hyper-reduced order models. This section is adapted from [78], and follows the notation presented therein.

Let $\mathbf{a} : \Omega \times \mathcal{P} \to \mathcal{R}^n$ be a parametric vector-valued function, where domain $\Omega$ partitioned into $n_{\text{cell}}$ finite element subdomains $\{\Omega^e\}_{e=1}^{n_{\text{cell}}}$ such that $\Omega = \bigcup_{e=1}^{n_{\text{cell}}} \Omega^e \subset \mathcal{R}^d$, with $d = 1, 2$, or 3. The parameter domain is denoted as $\mathcal{P}$. Each element within $\Omega^e$ is assumed to be isoparametric. All elements share the same interpolation order and contain $r$ Gauss integration points.

For a set of $N_s$ parameter instances $\{\mu_j\}_{j=1}^{N_s} \subset \mathcal{P}$, and given the values of the integrand at all Gauss points, the integral of the function $\mathbf{a}$ over $\Omega$ for each parameter $\mu_j$ is computed using the element-wise Gauss quadrature rule:

$$b_k = \sum_{e=1}^{n_{\text{cell}}} \int_{\Omega^e} a_i(x, \mu_j)\, d\Omega = \sum_{e=1}^{n_{\text{cell}}} \sum_{g=1}^{r} a_i(x_g^e, \mu_j) W_g^e, \quad \text{where} \quad k = (j-1)n + i, \tag{96}$$

for $j = 1, \ldots, N_s$ and $i = 1, \ldots, n$. In this equation, $x_g^e \in \Omega^e$ denotes the position of the $g$th Gauss point within element $\Omega^e$, and $W_g^e > 0$ is the weight associated with the $g$th Gauss point, calculated as the product of the Gauss weight and the Jacobian determinant for the isoparametric transformation. The quantity $b_k$ represents the "exact" integral value for each component $i$ of $\mathbf{a}$ and parameter instance $\mu_j$, serving as the reference value we aim to approximate.

This computation can be concisely expressed in matrix form:

$$\mathbf{b}_{\text{FE}} = \mathsf{A}_{\text{FE}}^{\top} \mathbf{W}_{\text{FE}}\,, \tag{97}$$

where $\mathbf{b}_{\text{FE}} \in \mathcal{R}^{nN_s}$ is the vector of "exact" integrals for all components and parameter instances. The matrix $\mathsf{A}_{\text{FE}}$ contains the values of the integrand $a_i(x_g^e, \mu_j)$ at all Gauss points for each parameter $\mu_j$, and $\mathbf{W}_{\text{FE}}$ is a column vector containing all the finite element weights $W_g^e$. The ECM matrix $\mathsf{A}_{FE} \in \mathcal{R}^{M \times nN_s}$ ($M = rn_{cell}$ is the total number of original cubature

points) can be expressed in terms of element contributions as:

$$\mathsf{A}_{FE} = \left[\mathsf{A}_{FE}^{(1)\top}, \mathsf{A}_{FE}^{(2)\top}, \quad \dots \quad, \mathsf{A}_{FE}^{(n_{\text{cell}})\top}\right]^{\top}, \tag{98}$$

where each block matrix $\mathsf{A}_{FE}^{(e)} \in \mathcal{R}^{r \times nN_s}$ corresponds to the $r$ Gauss points of element $e$ and is defined as:

$$\mathsf{A}_{FE}^{(e)} = \begin{bmatrix} a_1(x_1^e, \mu_1) & a_2(x_1^e, \mu_1) & \cdots & a_n(x_1^e, \mu_1) & a_1(x_1^e, \mu_2) & \cdots & a_n(x_1^e, \mu_{N_s}) \\ a_1(x_2^e, \mu_1) & a_2(x_2^e, \mu_1) & \cdots & a_n(x_2^e, \mu_1) & a_1(x_2^e, \mu_2) & \cdots & a_n(x_2^e, \mu_{N_s}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_1(x_r^e, \mu_1) & a_2(x_r^e, \mu_1) & \cdots & a_n(x_r^e, \mu_1) & a_1(x_r^e, \mu_2) & \cdots & a_n(x_r^e, \mu_{N_s}) \end{bmatrix}. \tag{99}$$

Similarly, the vector of quadrature weights $\mathbf{W}_{FE} \in \mathcal{R}^{M \times 1}$ is expressed as:

$$\mathbf{W}_{FE} = \left[\mathbf{W}_{FE}^{(1)\top}, \mathbf{W}_{FE}^{(2)\top}, \quad \dots \quad, \mathbf{W}_{FE}^{(n_{\text{cell}})\top}\right]^{\top}, \tag{100}$$

where each sub-vector $\mathbf{W}_{FE}^{(e)} \in \mathcal{R}^{r \times 1}$ contains the weights for the Gauss points of element $e$:

$$\mathbf{W}_{FE}^{(e)} = \left[W_1^e, W_2^e, \quad \dots \quad, W_r^e\right]^{\top}. \tag{101}$$

This formulation organizes the integrand values and corresponding weights for each finite element and Gauss point.

We further define vector $\mathbf{X}_{FE} \in \mathcal{R}^{M \times 1}$ which contains the corresponding cubature points:

$$\mathbf{X}_{FE} = \left[\mathbf{X}_{FE}^{(1)\top}, \mathbf{X}_{FE}^{(2)\top}, \quad \dots \quad, \mathbf{X}_{FE}^{(n_{\text{cell}})\top}\right]^{\top}, \tag{102}$$

and

$$\mathbf{X}_{FE}^{(e)} = \left[x_1^e, x_2^e, \quad \dots \quad, x_r^e\right]^{\top}. \tag{103}$$

### Finding cubature points

Similar to ECSW, we aim to find a sparse set of cubature points $X = \{x_g\}_{g=1}^{m}$ with $x_g \in \Omega$ and their associated positive weights $\{\omega_g\}_{g=1}^{m}$, where $m \ll M$. The objective is to minimize $m$ while approximating the vector of "exact" integrals $\mathbf{b}_{FE}$ to a desired accuracy $0 \leq \epsilon \leq 1$ by solving this best subset selection optimization problem:

$$\min_{\boldsymbol{\omega} \geq 0} \|\boldsymbol{\omega}\|_0, \quad \text{subject to} \quad \left\| \mathsf{A}_{FE}^{\top} \boldsymbol{\omega} - \mathsf{A}_{FE}^{\top} \mathbf{W}_{FE} \right\| \leq \epsilon_b \left\| \mathbf{b}_{FE} \right\|, \tag{104}$$

where $\boldsymbol{\omega} \in \mathcal{R}^{M \times 1}$ and $\|\boldsymbol{\omega}\|_0$ denotes the $l_0$-norm, counting the number of nonzero entries in $\boldsymbol{\omega}$, which we denote using $m$, and $\epsilon_b$ is a specified tolerance. However, as discussed

in Section 4.2.1, such sparsity-promoting optimization problems are NP-hard and are typically solved using suboptimal greedy heuristics or convex relaxation techniques, such as non-negative least squares [219, 221] and non-negative $l_1$-norm minimization [197].

However, the newly developed cubature point selection algorithm (see Algorithm 4) introduced in [76] offers an efficient and optimal *sparse* solution for the following alternative linear-least square problem:

$$\boldsymbol{\omega} = \arg \min_{\boldsymbol{\omega}^* \in \mathcal{R}^{M\times 1},\ \boldsymbol{\omega}^* \geq 0} \quad \left\| \mathsf{A}_{FE}^{\top} \boldsymbol{\omega}^* - \mathsf{A}_{FE}^{\top} \mathbf{W}_{FE} \right\| . \tag{105}$$

The python and MATLAB implementations of the algorithm are available at Ref. [222].

### Dimension reduction of $\mathsf{A}_{FE}$

To reduce the computational cost of solving Eq. (105), we approximate and decompose the ECM matrix $\mathsf{A}_{FE}$ using singular value decomposition (SVD):

$$\mathsf{A}_{FE} \approx \widetilde{\mathbb{W}}\, \mathsf{S}\, \underline{\mathbb{V}}^{\top} , \tag{106}$$

where $\widetilde{\mathbb{W}} \in \mathcal{R}^{M\times m}$ contains the left singular vectors for the $m$ retained modes, $\mathsf{S} \in \mathcal{R}^{m\times m}$ is the diagonal matrix of singular values, and $\underline{\mathbb{V}} \in \mathcal{R}^{nN_s\times m}$ contains the corresponding right singular vectors.

As a result of this reduction, the objective function in Eq. (105) transforms into:

$$\left\| \mathsf{V}\Sigma \left( \widetilde{\mathbb{W}}^{\top} \boldsymbol{\omega} - \widetilde{\mathbb{W}}^{\top} \mathbf{W}_{FE} \right) \right\| . \tag{107}$$

This reformulation simplifies the optimization problem, making it computationally more tractable. We then use the cubature point selection strategy described in Algorithm 4 to solve this approximate optimization problem, which essentially boils down to finding a sparse $\boldsymbol{\omega}$ such that

$$\widetilde{\mathbb{W}}^{\top} \boldsymbol{\omega} \approx \widetilde{\mathbb{W}}^{\top} \mathbf{W}_{FE} . \tag{108}$$

The algorithm yields $\boldsymbol{\omega}$ and a selection matrix $\mathsf{Z} \in \mathcal{R}^{m\times M}$ with each row containing a single non-zero entry such that

$$(\mathsf{Z}\widetilde{\mathbb{W}})^{\top} \mathsf{Z}\boldsymbol{\omega} = \widetilde{\mathbb{W}}^{\top} \boldsymbol{\omega} . \tag{109}$$

where

$$\mathsf{Z}\mathbf{X}_{FE} = X^* = [x_1^*, x_2^*, \ldots, x_m^*]^{\top} \tag{110}$$

are the cubature points corresponding to nonzero entries of $\boldsymbol{\omega}$.

This sparse selection of the cubature points leads to a reduced mesh, which is then used to compute the parametric nonlinearity more efficiently in the *online phase*.

### Implementation on the running example

To implement ECM on the example problem in Section 3, we begin by writing the nonlinear reduced stiffness matrix and the force vector in terms of the gauss-quadrature points for the $j-$th parameter pair $(\mu_j, \beta_j)$:

$$\mathsf{K}_n(\mathbf{T}_n; \mu) = \sum_{e=1}^{n_{\text{cell}}} \mathsf{K}_n^e \,, \tag{111}$$

where

$$\mathsf{K}_n^e = \sum_{g=1}^{r} w_g \; \mathsf{K}_n^e(\mathbf{x}_g^e) \,, \tag{112}$$

and

$$\mathsf{K}_n^e(\mathbf{x}_g^e) = k(T(\mathbf{x}_g^e; \mu_j, \beta_j), \mu_j) \, \widetilde{\mathbb{U}}^{e\,\top} \nabla \mathbf{\Phi}^e(\mathbf{x}_g^e) \nabla \mathbf{\Phi}^e(\mathbf{x}_g^e)^\top \,. \tag{113}$$

Here $k(T(\mathbf{x}_g^e; \mu_j, \beta_j); \mu_j)$ is the nonlinear thermal conductivity evaluated at $\mathbf{x}_g^e$, where $\mathbf{x}_g^e \in \Omega^e$ is the $g$th cubature point within the $e-$th cell; $w_g^e$ is the cubature weight associated with $\mathbf{x}_g^e$; vector $\mathbf{\Phi}^e(\mathbf{x}_g^e) = [\phi_1^e(\mathbf{x}_g^e), \phi_2^e(\mathbf{x}_g^e)]^\top$ contains the two Lagrange basis functions associated with element $e$, which were used in the high fidelity finite element model in Eq. (27); and $\widetilde{\mathbb{U}}^e$ is as defined in Eq. (80).

Similarly, the nonlinear reduced force vector is given by

$$\mathbf{q}_n(\mathbf{T}_n; \mu) = \sum_{e=1}^{n_{\text{cell}}} \mathbf{q}_n^e \,, \tag{114}$$

where

$$\mathbf{q}_n^e = \sum_{g=1}^{r} w_g \; \mathbf{q}_n^e(\mathbf{x}_g^e), \tag{115}$$

and

$$\mathbf{q}_n^e(\mathbf{x}_g^e) = q(T(\mathbf{x}_g^e; \mu_j, \beta_j), \beta_j) \widetilde{\mathbb{U}}^{e\,\top} \mathbf{\Phi}^e(\mathbf{x}_g^e) \,, \tag{116}$$

where $q(T(\mathbf{x}_g^e; \mu_j, \beta_j); \beta_j)$ represents the nonlinear source term evaluated at $\mathbf{x}_g^e \in \Omega^e$.

In ECM, similar to ECSW, we compute the elemental residual force vector *at each quadrature point* to construct the $\mathsf{A}_{FE}$ matrix, and subsequently generate the reduced mesh by solving Eq. (105). Recall from Eq. (94) that the residual balance vector for element $e$, corresponding to the $i$th snapshot $\widetilde{\mathbf{T}}_n^i$ (see Eq. 93) is given by

$$\gamma_n^{i,e} = \mathsf{K}_n^{i,e} \, (\mathsf{Q}_e \overline{\mathbf{T}}_N + \widetilde{\mathbb{U}}_e \widetilde{\mathbf{T}}_n^i) - \mathbf{q}_n^{i,e} \,. \tag{117}$$

We rewrite this as

$$\gamma_n^{i,e} = \sum_{g=1}^{r} w_g \, \gamma_{n,g}^{i,e} \,, \tag{118}$$

where

$$\gamma_{n,g}^{i,e} = \mathsf{K}_n^{i,e}(\mathbf{x}_g^e)(\mathsf{Q}_e \overline{\mathbf{T}}_N + \widetilde{\mathbb{U}}_e \widetilde{\mathbf{T}}_n^i) - \mathbf{q}_n^{i,e}(\mathbf{x}_g^e) \,. \tag{119}$$

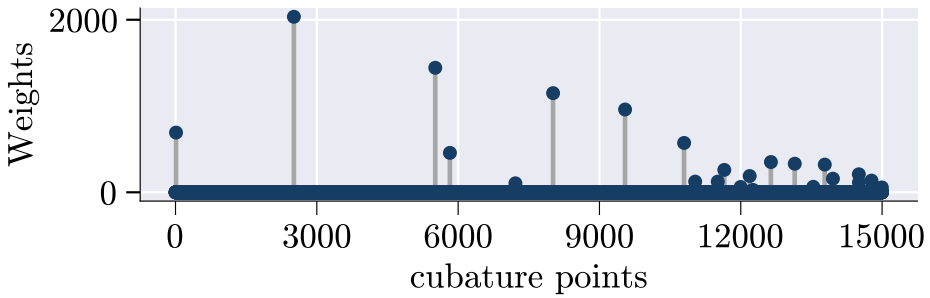



**Figure 21:** Sparse distribution of cubature points and corresponding weights achieved using the ECM-based hyper-reduction technique. The ECM algorithm selected 30 points out of 15,000 cubature points (3 cubature points/element) employed in the high-fidelity model, leading to a substantially reduced mesh.

The matrix $\mathsf{A}_{FE} \in \mathcal{R}^{M \times nN_s}$ is then built for the $N_s$ snapshots:

$$\mathsf{A}_{FE}^{(e)} = \begin{bmatrix} \left(\gamma_{n,1}^{1,e}\right)_1 & \cdots & \left(\gamma_{n,1}^{1,e}\right)_n & \left(\gamma_{n,1}^{2,e}\right)_1 & \cdots & \left(\gamma_{n,1}^{s,e}\right)_n \\ \left(\gamma_{n,2}^{1,e}\right)_1 & \cdots & \left(\gamma_{n,2}^{1,e}\right)_n & \left(\gamma_{n,2}^{2,e}\right)_1 & \cdots & \left(\gamma_{n,2}^{s,e}\right)_n \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \left(\gamma_{n,r}^{1,e}\right)_1 & \cdots & \left(\gamma_{n,r}^{1,e}\right)_n & \left(\gamma_{n,r}^{2,e}\right)_1 & \cdots & (\gamma_{n,r}^{s,e})_n \end{bmatrix}_{r \times nN_s}, \tag{120}$$

where $\left(\gamma_{n,g}^{i,e}\right)_i$ denotes the $i$th component of the elemental balance vector $\gamma_{n,g}^{i,e}$. The weight vector $\mathbf{W}_{FE}$ is given by Eq. (100). We then performed singular value decomposition (SVD) of $\mathsf{A}_{FE}$ while applying a truncation criterion of $10^{-8}$. This process resulted in the selection of the first 30 left singular vectors as shown in Fig. 22a, which were contained in $\widetilde{\mathbb{W}}$. Using $\widetilde{\mathbb{W}}$ and $\mathbf{W}_{FE}$ the reduced weight vector $\boldsymbol{\omega}$ was calculated using Algorithm 4, which led to the generation of the reduced mesh shown in Fig. 21. Specifically, ECM led to the selection of only 30 cubature points out of the 15000 in the domain, as depicted in Fig. 21. The value of the objective function in Eq. (107) is approximately $10^{-14}$, indicating a successful minimization.

In Figs. 22b and 22c, we compare the speed-up (Eq. 42b) and the relative percentage error (Eq. 42a) associated with the hyper-ROM with the regular projection-based ROM. Figure 22b shows very high accuracy for the hyper-ROM, with a relative error percentage of $e_T = 10^{-5}$. This error is almost comparable to the ROM. Like ECSW, the gain in computational speed-up $r_s$ is significant almost $100\times$ that of the regular ROM and $400\times$ that of the HFM as shown in Fig. 22c. This is lower than that observed for ECSW-based hyper-ROM but higher than DEIM-based hyper-ROM.

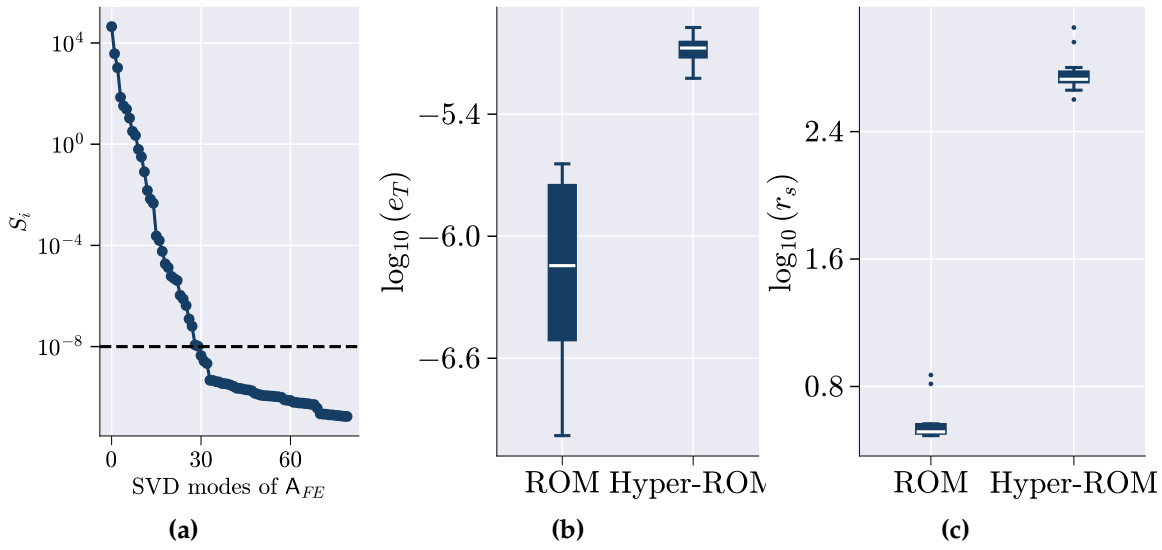



**Figure 22:** (a) Decay of the singular values ($S_i$) of the ECM matrix $A_{FE}$, which is compressed to enhance computational efficiency in identifying a sparse set of cubature points. (b) Difference in accuracy ($e_T$) and (c) speed-up ($r_s$) between the regular projection-based ROM and the ECM Hyper-ROM.

### 4.3 Deep-learning-based strategies

In recent years, deep-learning-assisted hyper-reduced ROMs have seen significant advancements [223, 224]. This section offers a concise overview of a few of such strategies.

Deep-learning-based approaches are especially relevant for Nonlinear Manifold ROMs (NM-ROMs) [122, 123]. An NM-ROM represents the solution on a nonlinear manifold $\mathcal{S} \triangleq \{\mathcal{G}(\mathbf{v}) \mid \mathbf{v} \in \mathcal{R}^n\}$, where $\mathcal{G} : \mathcal{R}^n \to \mathcal{R}^N$ is a nonlinear function mapping a latent space of dimension $n$ (with $n \ll N$) to the full-order model space of dimension $N$. The NM-ROM approximates the PDE solution within a trial manifold as follows:

$$\mathbf{w}(t,\mu)_N \approx \widetilde{\mathbf{w}}(t,\mu)_N = \mathbf{w}_N^{\text{ref}} + \mathcal{G}_n(\mathbf{w}_n(t,\mu))\,, \tag{121}$$

where $\mathbf{w}_n \in \mathcal{R}^n$ denotes the reduced coordinates. The associated time derivative is given by

$$\dot{\mathbf{w}}_N \approx \dot{\widetilde{\mathbf{w}}}_N = \mathsf{J}_g(\mathbf{w}_n)\dot{\mathbf{w}}_n\,, \tag{122}$$

where $\mathsf{J}_g$ denotes the Jacobian of $\mathcal{G}_n$, with the initial condition $\mathbf{w}_n(0,\mu) = \mathcal{H}(\mathbf{w}_N(0,\mu) - \mathbf{w}_N^{\text{ref}})$, where the nonlinear function $\mathcal{H} \approx \mathcal{G}^{-1}$ satisfies the following:

$$\widetilde{\mathbf{w}}_N - \mathbf{w}_N^{\text{ref}} \approx \mathcal{G}(\mathcal{H}(\mathbf{w}_N - \mathbf{w}_N^{\text{ref}}))\,. \tag{123}$$

So far, we have introduced $\mathcal{G}$ and $\mathcal{H}$ in an abstract sense. However, identifying $\mathcal{G}$ and $\mathcal{H}$ can be challenging; therefore, neural networks, particularly autoencoders, are frequently

employed to approximate these functions [123]. Specifically, the decoder serves as $\mathcal{G}$, while the encoder serves as $\mathcal{H}$.

The NM-ROM framework deals with two layers of nonlinearities: the nonlinearity in the original governing equations and the nonlinearity of the decoder, which influences the residual of the PDE. The first layer can be addressed using the hyper-reduction strategies discussed earlier; however, the second layer necessitates special treatment. The Jacobian of the decoder must be computed at each solver iteration, and its computational cost scales with the number of learnable parameters, potentially reducing computational efficiency. To mitigate this, a subnet mask can be constructed to compute only the necessary outputs, bypassing the need for the full decoder or its Jacobian.

NM-ROMs are particularly effective for transport-dominated systems (e.g. advection- or convection-dominated systems). Such systems exhibit moving coherent structures (waves, vortices, steep gradients) that results in a slow decay of Kolmogorov n-width [225]. As a result the dimension of the linear subspace ($\widetilde{\mathbb{U}}$) required to capture the solution is typically characterized by a prohibitively large $n$, diminishing the effectiveness of pROMs and hyper-reduced ROMs. If an insufficient number of modes is used (due to the above slow decay), the ROM can produce inaccurate or oscillatory solutions around steep fronts. Furthermore, in a parametric setting, the speed of transported features may vary with parameters, further enlarging the space of possible solutions. By leveraging a nonlinear manifold, NM-ROMs directly address the $n$-width problem, providing a more efficient solution. A few other hyper-reduction algorithms developed to tackle this class of problems include adaptive sampling methods [198, 226], structure-preserving approaches [227], and hybrid strategies such as the DG-RB-EQP framework (discontinuous Galerkin–reduced basis with empirical quadrature) [228], among others.

In a parallel effort to overcome the same $n$-width problem, Barnett et al. [229] introduced a neural-network-augmented strategy, but within the framework of LS-ROMs instead of NM-ROMs. In their proposed framework, termed PROM-ANN, the PDE solution is retained within a linear subspace derived from the solution snapshot matrix, while incorporating an artificial neural network $\mathcal{N}$, which acts as a corrector to enhance the approximation:

$$\mathbf{w}_N(t;\boldsymbol{\mu}) \approx \widetilde{\mathbf{w}}_N(t;\boldsymbol{\mu}) = \mathbf{w}_N^{\text{ref}} + \widetilde{\mathbb{U}}\mathbf{w}_n(t;\boldsymbol{\mu}) + \widetilde{\mathbb{U}}^c\mathcal{N}(\mathbf{w}_n(t;\boldsymbol{\mu}))\,, \tag{124}$$

where $\widetilde{\mathbb{U}}$ is a rank-$n$ matrix obtained by truncating the first $n$ dominant proper orthogonal decomposition (POD) modes of the snapshot matrix, and $\widetilde{\mathbb{U}}^c$ is a rank-$n_c$ matrix containing the next $n_c$ POD modes. The sum of $n$ and $n_c$ is chosen to be less than or equal to the rank of the snapshot matrix. Note that $\widetilde{\mathbb{U}}^\top\widetilde{\mathbb{U}} = \mathsf{I}_n$, $\widetilde{\mathbb{U}}^{c\top}\widetilde{\mathbb{U}}^c = \mathsf{I}_{n_c}$, and $\widetilde{\mathbb{U}}^\top\widetilde{\mathbb{U}}^c = \mathsf{0}$. Here, the value of $n$ is deliberately chosen to be small to maintain a low-dimensional ROM, implying that $\widetilde{\mathbb{U}}$ captures a smaller fraction of the snapshot data variance compared to conventional approaches (e.g., 0.9999).

Given a training snapshot $\mathbf{w}_N(t_i,\boldsymbol{\mu}_j)$, it can be shown that ideally

$$\widetilde{\mathcal{U}}^{c\top}(\mathbf{w}_N(t_i,\boldsymbol{\mu}_j) - \mathbf{w}_N^{\text{ref}}) = \mathcal{N}(\mathbf{w}_n(t;\boldsymbol{\mu})) = \mathbf{w}_n^c\,. \tag{125}$$

Although the equality in Eq. (125) is difficult to achieve in practice, the neural network $\mathcal{N}$ can be trained for all solution snapshots in the snapshot matrix, enabling it to produce a

vector that closely resembles $\mathbf{w}_n^c \in \mathcal{R}^{n_c \times 1}$ for a given $\mathbf{w}_n$. Once trained, it is then sufficient to compute $\mathbf{w}_n$ using an $n$-dimensional ROM and subsequently determine $\widetilde{\mathbf{w}}_N$ via Eq. (124). The authors further extended this formulation to develop ECSW-based hyper-reduced LSPG pROMs [229].

# 5 Discussion and outlook

In this paper, we examined the role of projection-based reduced-order models (ROMs) in addressing computational challenges in large-scale nonlinear systems. While ROMs significantly reduce computational complexity by projecting the system onto a lower-dimensional reduced subspace, they often struggle with efficiently handling nonlinear terms, which can lead to increased computational costs instead of the intended speed-up. To mitigate this issue, we focused on hyper-reduction techniques that approximate or directly project nonlinear terms in a way that preserves both accuracy and efficiency.

Through a case study of a nonlinear heat conduction problem, modeled using the finite element method, we demonstrated and compared the implementation of several hyper-reduction approaches. The selected problem was conceptually simple, yet nonlinear, which made it well-suited for the demonstration. First, we formulated its reduced-order model without hyper-reduction to illustrate the computational challenges posed by nonlinear terms. We then systematically applied various hyper-reduction methods to this example, evaluating their performance in terms of computational speed-up and accuracy.

In our discussion of the state-of-the art hyper-reduction techniques, we categorized the methods into two main classes: approximate-then-project (AP) and project-then-approximate (PA). The former included methods such as the Discrete Empirical Interpolation Method (DEIM), which approximate nonlinear terms before projecting them onto the reduced subspace. The latter included techniques such as the Energy Conserving Sampling and Weighting method and the Empirical Cubature Method, which estimate reduced-order quantities directly by sampling a subset of elements or integration points from the full-order computational domain. Through our example problem, we compared these techniques and assessed their effectiveness in reducing computational complexity while maintaining accuracy. Our findings suggest that PA methods often yield more accurate, stable, and faster hyper-ROMs compared to their AP counterparts.

Despite advancements in hyper-reduction, challenges remain in integrating these methods into commercial simulation software. As highlighted in our comparison of open-source and commercial tools, the latter largely lack support for hyper-reduction due to the deeply intrusive nature of these techniques. Implementing hyper-reduction requires significant modifications to existing computational frameworks, which can hinder widespread adoption. However, given the increasing demand for real-time simulations in fields such as digital twins, there is strong motivation for commercial vendors to explore ways to incorporate hyper-reduction methodologies without disrupting existing workflows. This review aimed to provide an accessible introduction to hyper-reduction methods, emphasizing

their practical implementation and potential impact. By offering a structured overview, accompanied by an *open-source GitHub repository*, we hope to facilitate further research and adoption of these techniques within the computational science community.

Future work should focus on developing robust and scalable hyper-reduction frameworks that can be effectively implemented in both academic research and industrial applications, ensuring practical usability and seamless integration with existing computational tools. Recent developments in deep-learning-assisted hyper-reduction also offer promising directions for further enhancing computational efficiency. Additionally, strategies are needed to simultaneously handle both spatial and temporal complexities in large-scale nonlinear dynamical systems. Hyper-reduction techniques primarily address spatial complexity by reducing the number of spatial degrees of freedom required for nonlinear evaluations. However, in many dynamical systems, temporal complexity also contributes significantly to computational cost. While traditional hyper-reduction methods focus on spatial reduction, identifying and formulating low-dimensional nonlinear manifolds in the state space provides an alternative approach to model reduction, especially for capturing long-term dynamics. These manifolds, inferred from physics or data-driven methods, offer a compact representation of the evolution of the system. Integrating manifold-based approaches with hyper-reduction—an area that remains relatively unexplored—could enhance model order reduction frameworks, particularly for dynamical systems.

Another promising direction for future research is integrating hyper-ROMs with digital twins. Digital twins rely on rapid model updates and data processing to interact with their physical counterparts, which often exhibit highly nonlinear behavior. Hyper-reduction may facilitate this by accelerating computations and reducing latency in model updates, allowing digital twins to provide timely insights and support decision-making. Additionally, hyper-reduction could contribute to verification, validation, and uncertainty quantification (VVUQ) by improving computational efficiency and model fidelity, thereby enhancing the reliability of digital twin frameworks. Beyond digital twins, other important research directions include the VVUQ of hyper-reduction algorithms themselves, ensuring their reliability, accuracy, and robustness across different applications. These advancements in hyper-reduction have the potential to significantly enhance digital twin applications and broader computational modeling efforts, thereby paving the way for more efficient and scalable simulations.

# 6 Acknowledgments

This work was supported by the Digital Twin Lab at the Texas A&M Institute of Data Science. We thank the pylibROM team at Lawrence Livermore National Laboratory (LLNL) for valuable discussions. We also acknowledge inputs from Quincy Huhn and Naveen Jagadeesan, as well as brief exchanges with Prof. Charbel Farhat and Prof. Francesco Ballarin.

# References


[1] A.C. Antoulas. Approximation of large-scale dynamical systems: An overview. *IFAC Proceedings Volumes*, 37(11):19–28, Jul 2004.

[2] Santosh Kumar Suman and Awadhesh Kumar. Investigation and implementation of model order reduction technique for large scale dynamical systems. *Archives of Computational Methods in Engineering*, 29(5):3087–3108, Jan 2022.

[3] Peter Benner, Wil Schilders, Stefano Grivet-Talocia, Alfio Quarteroni, Gianluigi Rozza, and Luís Miguel Silveira. *Model Order Reduction: Volume 3 Applications*. De Gruyter, 2020.

[4] Parviz Moin & Krishnan Mahesh. DIRECT NUMERICAL SIMULATION: A Tool in Turbulence. *Annu. Rev. Fluid Mech.*, 30(Volume 30, 1998):539–578, Jan 1998.

[5] Massimo Germano & Ugo Piomelli & Parviz Moin & William H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A*, 3(7):1760–1765, Jul 1991.

[6] Jeffrey Slotnick & Abdollah Khodadoust & Juan Alonso & David Darmofal & William Gropp & Elizabeth Lurie & Dimitri Mavriplis. CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences. Technical Report NASA/CR-2014-218178, NASA, 2014. URL https://ntrs.nasa.gov/api/citations/20140003093/downloads/20140003093.pdf. Accessed: 2024-11-28.

[7] Earl H. Dowell & Kenneth C. Hall. MODELING OF FLUID-STRUCTURE INTERACTION. *Annu. Rev. Fluid Mech.*, (Volume 33, 2001):445–490, Jan 2001.

[8] P Jenny & S.H Lee & H.A Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation . *Journal of Computational Physics*, 187(1):47–67, May 2003.

[9] Hamed Jamshidi Aval. Comprehensive thermo-mechanical simulation of friction surfacing of aluminum alloys using smoothed particle hydrodynamics method. *Surface and Coatings Technology*, 419:127274, 2021.

[10] F. Dambakizi, P. Le Tallec, and J.P. Perlat. Multiscale thermomechanical modeling of shock-driven dry friction in hydrodynamics. *Computer Methods in Applied Mechanics and Engineering*, 198:1701–1715, 2009.

[11] Zhi-Chao Zhang and Xiao-Hui Cheng. A thermo-mechanical coupled constitutive model for clay based on extended granular solid hydrodynamics. *Computers and Geotechnics*, 80:373–382, 2016.

[12] Elizabeth J. Tasker & Riccardo Brunino & Nigel L. Mitchell & Dolf Michielsen & Stephen Hopton & Frazer R. Pearce & Greg L. Bryan & Tom Theuns. A test suite for quantitative comparison of hydrodynamic codes in astrophysics. *Mon. Not. R. Astron. Soc.*, 390(3):1267–1281, Nov 2008.

[13] R. L. Williamson & J. D. Hales & S. R. Novascone & M. R. Tonks & D. R. Gaston & C. J. Permann & D. Andrs & R. C. Martineau. Multidimensional multiphysics simulation of nuclear fuel behavior. *J. Nucl. Mater.*, 423(1):149–163, Apr 2012.

[14] Derek R. Gaston & Cody J. Permann & John W. Peterson & Andrew E. Slaughter & David Andrš & Yaqi Wang & Michael P. Short & Danielle M. Perez & Michael R. Tonks & Javier Ortensi & Ling Zou & Richard C. Martineau. Physics-based multiscale coupling for full core nuclear reactor simulation. *Ann. Nucl. Energy*, 84:45–54, Oct 2015.

[15] D. Mehta & A. H. van Zuijlen & B. Koren & J. G. Holierhoek & H. Bijl. Large Eddy Simulation of wind farm aerodynamics: A review. *J. Wind Eng. Ind. Aerodyn.*, 133: 1–17, Oct 2014.

[16] M. Ripepi & M. J. Verveld & N. W. Karcher & T. Franz & M. Abu-Zurayk & S. Görtz & T. M. Kier. Reduced-order models for aerodynamic applications, loads and MDO. *CEAS Aeronautical Journal*, 9:171–193, 2018.

[17] Dimitri Mavriplis, David Darmofal, David Keyes, and Mark Turner. Petaflops Opportunities for the NASA Fundamental Aeronautics Program (Invited). In *18th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, Jun 2007.

[18] ALI H. NAYFEH, MOHAMMAD I. YOUNIS, and EIHAB M. ABDEL-RAHMAN. Reduced-order models for mems applications. *Nonlinear Dynamics*, 41(1–3):211–236, Aug 2005.

[19] Giorgio Gobat, Andrea Opreni, Stefania Fresca, Andrea Manzoni, and Attilio Frangi. Reduced order modeling of nonlinear microstructures through proper orthogonal decomposition. *Mechanical Systems and Signal Processing*, 171:108864, May 2022.

[20] J. B. Rützmoser. *Model Order Reduction for Nonlinear Structural Dynamics*. PhD thesis, Technische Universität München, München, Germany, 2018.

[21] Ming-Chen Hsu and Yuri Bazilevs. Fluid–structure interaction modeling of wind turbines: simulating the full machine. *Computational Mechanics*, 50(6):821–833, Aug 2012.

[22] V. Guillot, A. Ture Savadkoohi, and C.-H. Lamarque. Analysis of a reduced-order nonlinear model of a multi-physics beam. *Nonlinear Dynamics*, 97:1371–1401, 2019.

[23] Dongli Huang, Hany Abdel-Khalik, Cristian Rabiti, and Frederick Gleicher. Dimensionality reducibility for multi-physics reduced order modeling. *Annals of Nuclear Energy*, 110:526–540, 2017.

[24] Boris Kramer. Model reduction for control of a multiphysics system: Coupled Burgers' equation. In *2016 American Control Conference (ACC)*, pages 6146–6151, 2016.

[25] Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2005.

[26] J. N. Reddy. *Introduction to the Finite Element Method*. McGraw-Hill, 4 edition, 2018.

[27] Thomas J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[28] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[29] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*, volume 791 of *Fluid Mechanics and Its Applications*. Springer Cham, 1 edition, 2016.

[30] H. Versteeg. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson, Harlow, 2nd edition edition, Feb 2007.

[31] Charbel Farhat, Philip Avery, Todd Chapman, and 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*, 98(9):625–662, 2014.

[32] Subhash Kak. The Intrinsic Dimensionality of Data. *Circuits, Systems, and Signal Processing*, 40:2599–2607, 2020.

[33] P.J. Verveer and R.P.W. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:81–86, 1995.

[34] William D. Fries, Xiaolong He, and Youngsoo Choi. LaSDI: Parametric Latent Space Dynamics Identification. *Computer Methods in Applied Mechanics and Engineering*, 399: 115436, 2022.

[35] Benyamin Ghojogh, Mark Crowley, Fakhri Karray, and Ali Ghodsi. *Elements of dimensionality reduction and manifold learning*. Springer, 2023.

[36] Danish Rafiq and Mohammad Abid Bazaz. Model order reduction via moment-matching: A state of the art review. *Archives of Computational Methods in Engineering*, 29(3):1463–1483, Jun 2021.

[37] David J. Lucia, Philip S. Beran, and Walter A. Silva. Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40:51–117, 2004.

[38] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. Autom. Control*, 26(1):17–32, Feb 1981. doi: 10.1109/TAC.1981.1102568.

[39] Philip Holmes, John L. Lumley, and Gal Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 1996.

[40] C. W. ROWLEY. MODEL REDUCTION FOR FLUIDS, USING BALANCED PROPER ORTHOGONAL DECOMPOSITION. *International Journal of Bifurcation and Chaos*, 15: 997–1013, 2005.

[41] Gaetan Kerschen & Jean claude Golinval & Alexander F. Vakakis & Lawrence A. Bergman. The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview. *Nonlinear Dyn.*, 41(1):147–169, Aug 2005.

[42] Kenneth C. Hall & Jeffrey P. Thomas & Earl H. Dowell. Proper Orthogonal Decomposition Technique for Transonic Unsteady Aerodynamic Flows. *AIAA Journal*, 38(10), Oct 2010.

[43] Earl Dowell. Reduced-order modeling: a personal journey. *Nonlinear Dyn.*, 111(11): 9699–9720, Jun 2023.

[44] Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, Jul 2015.

[45] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current Science*, 78:808–817, 2000.

[46] Y.C. LIANG, H.P. LEE, S.P. LIM, W.Z. LIN, K.H. LEE, and C.G. WU. PROPER ORTHOGONAL DECOMPOSITION AND ITS APPLICATIONS–PART I: THEORY. *Journal of Sound and Vibration*, 252:527–544, 2002.

[47] P. Cusumano J. & T. Sharkady M. & W. Kimble B. Experimental measurements of dimensionality and spatial coherence in the dynamics of a flexible-beam impact oscillator. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 347(1683):421–438, May 1994.

[48] B. F. Feeny & R. Kappagantu. ON THE PHYSICAL INTERPRETATION OF PROPER ORTHOGONAL MODES IN VIBRATIONS. *J. Sound Vib.*, 211(4):607–616, Apr 1998.

[49] Nadine Aubry. On the hidden beauty of the proper orthogonal decomposition. *Theor. Comput. Fluid Dyn.*, 2(5):339–352, Aug 1991.

[50] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237–240:10–26, Sep 2012.

[51] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for numerical methods in engineering*, 86(2):155–181, 2011.

[52] Eric J. Parish, Christopher R. Wentland, and Karthik Duraisamy. The Adjoint Petrov–Galerkin method for non-linear model reduction. *Computer Methods in Applied Mechanics and Engineering*, 365:112991, 2020.

[53] D. Xiao, F. Fang, J. Du, C.C. Pain, I.M. Navon, A.G. Buchan, A.H. ElSheikh, and G. Hu. Non-linear Petrov–Galerkin methods for reduced order modelling of the Navier–Stokes equations using a mixed finite element pair. *Computer Methods in Applied Mechanics and Engineering*, 255:147–157, 2013.

[54] Zhongying Chen and Yuesheng Xu. The Petrov–Galerkin and Iterated Petrov–Galerkin Methods for Second-Kind Integral Equations. *SIAM Journal on Numerical Analysis*, 35:406–434, 1998.

[55] Sebastian Grimberg & Charbel Farhat & Radek Tezaur & Charbel 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*, 122(7):1846–1874, 2021.

[56] Eric Parish, Masayuki Yano, Irina Tezaur, and Traian Iliescu. Residual-based stabilized reduced-order models of the transient convection–diffusion–reaction equation obtained through discrete and continuous projection. *Archives of Computational Methods in Engineering*, Nov 2024.

[57] Peter Benner, Serkan Gugercin, and Karen Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Review*, 57: 483–531, 2015.

[58] M. Hinze and S. Volkwein. Proper Orthogonal Decomposition Surrogate Models for Nonlinear Dynamical Systems: Error Estimates and Suboptimal Control. In P. Benner, V. Mehrmann, and D. C. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 261–306. Springer-Verlag, Berlin Heidelberg, 2005.

[59] Mostafa Abbaszadeh, Mehdi Dehghan, and Ionel Michael Navon. A pod reduced-order model based on spectral galerkin method for solving the space-fractional gray–scott model with error estimate. *Engineering with Computers*, 38(3):2245–2268, Nov 2020.

[60] Fleurianne Bertrand, Daniele Boffi, and Abdul Halim. A reduced order model for the finite element approximation of eigenvalue problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115696, Feb 2023.

[61] David Amsallem and Charbel Farhat. Stabilization of Projection-Based Reduced-Order Models. *International Journal for Numerical Methods in Engineering*, 91(4):358–377, 2012.

[62] Ryan J. Klock and Carlos E. S. Cesnik. Nonlinear thermal reduced-order modeling for hypersonic vehicles. *AIAA Journal*, 55(7):2358–2368, Jul 2017.

[63] Sridhar Chellappa, Lihong Feng, and Peter Benner. Adaptive basis construction and improved error estimation for parametric nonlinear dynamical systems. *International Journal for Numerical Methods in Engineering*, 121(23):5320–5349, Aug 2020.

[64] Alfio Quarteroni and Alberto Valli. *Numerical Approximation of Partial Differential Equations*, volume 23. Springer Science & Business Media, 2008.

[65] Lihong Feng, Guosheng Fu, and Zhu Wang. A FOM/ROM Hybrid Approach for Accelerating Numerical Simulations. *Journal of Scientific Computing*, 89(3), Oct 2021.

[66] Tjalling J. Ypma. Historical Development of the Newton–Raphson Method. *SIAM Review*, 37:531–551, 1995.

[67] D. Ryckelynck. A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics*, 202:346–366, 2005.

[68] Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2016. ISBN 9783319224701. doi: 10.1007/978-3-319-22470-1. URL http://dx.doi.org/10.1007/978-3-319-22470-1.

[69] Maxime Barrault & Yvon Maday & Ngoc Cuong Nguyen & Anthony T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus. Mathématique*, 339:667–672, 2004.

[70] Saifon Chaturantabut & Danny C. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM Journal on Scientific Computing*, 32:2737–2764, 2010.

[71] N. C. Nguyen, A. T. Patera, and J. Peraire. A 'best points' interpolation method for efficient approximation of parametrized functions. *International Journal for Numerical Methods in Engineering*, 73:521–543, 2007.

[72] Charbel Farhat, Todd Chapman, and Philip 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*, 102:1077–1110, 2015.

[73] Radek Tezaur, Faisal As'ad, and Charbel Farhat. Robust and globally efficient reduction of parametric, highly nonlinear computational models and real time online performance. *Computer Methods in Applied Mechanics and Engineering*, 399:115392, 2022.

[74] Johannes Maierhofer and Daniel J. Rixen. Model order reduction using hyperreduction methods (DEIM, ECSW) for magnetodynamic FEM problems. *Finite Elements in Analysis and Design*, 209:103793, 2022.

[75] J. A. Hernández, Javier Oliver, Alfredo Edmundo Huespe, MA Caicedo, and JC Cante. High-performance model reduction techniques in computational multiscale homogenization. *Computer Methods in Applied Mechanics and Engineering*, 276:149–189, 2014.

[76] J. R. Bravo, J. A. Hernández, S. Ares de Parga, and R. Rossi. A subspace-adaptive weights cubature method with application to the local hyperreduction of parameterized finite element models, Sep 2024. ISSN 1097-0207. URL http://dx.doi.org/10.1002/nme.7590.

[77] J. A. Hernández. A multiscale method for periodic structures using domain decomposition and ECM-hyperreduction. *Computer Methods in Applied Mechanics and Engineering*, 368:113192, 2020.

[78] J. A. Hernández, J. R. Bravo, and S. Ares de Parga. CECM: A continuous empirical cubature method with application to the dimensional hyperreduction of parameterized finite element models. *Computer Methods in Applied Mechanics and Engineering*, 418:116552, 2024.

[79] Masayuki Yano and Anthony T. Patera. An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs. *Computer Methods in Applied Mechanics and Engineering*, 344:1104–1123, 2019.

[80] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[81] David Amsallem, Matthew J. Zahr, and Charbel Farhat. Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92:891–916, 2012.

[82] Zlatko Drmac & Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.

[83] Benjamin Peherstorfer, Daniel Butnaru, Karen Willcox, and Hans-Joachim Bungartz. Localized Discrete Empirical Interpolation Method. *SIAM Journal on Scientific Computing*, 36:A168–A192, 2014.

[84] Olivier Goury and Christian Duriez. Fast, Generic, and Reliable Control and Simulation of Soft Robots Using Model Order Reduction. *IEEE Transactions on Robotics*, 34: 1565–1576, 2018.

[85] Youngsoo Choi, William J. Arrighi, Dylan M. Copeland, Robert W. Anderson, and Geoffrey M. Oxberry. libROM. [Computer Software] https://doi.org/10.11578/dc.20190408.3, 2019.

[86] Vicente Mataix Ferrándiz, Philipp Bucher, Rubén Zorrilla, Riccardo Rossi, jcotela, Alejandro Cornejo Velázquez, Miguel Angel Celigueta, Josep Maria, tteschemacher, Carlos Roig, Guillermo Casas, miguelmaso, Suneth Warnakulasuriya, Marc Nú nez, Pooyan Dadvand, Salva Latorre, Ignasi de Pouplana, Joaquín Irazábal González, Ferran Arrufat, riccardotosi, Aditya Ghantasala, Peter Wilson, dbaumgaertner, Bodhinanda Chandra, AFranci, Armin Geiser, Klaus Bernd Sautter, Inigo Lopez, lluís, and Javi Gárate. KratosMultiphysics/Kratos:, 2022.

[87] René Milk, Stephan Rave, and Felix Schindler. pyMOR – Generic Algorithms and Interfaces for Model Order Reduction. *SIAM Journal on Scientific Computing*, 38(5): S194–S216, 2016. doi: 10.1137/15M1026614.

[88] Gianluigi Rozza, Francesco Ballarin, Leonardo Scandurra, and Federico Pichi. *Real Time Reduced Order Computational Mechanics*. SISSA Springer Series. Springer Cham, 2024.

[89] P. Benner & S. W. R. Werner. MORLAB—The Model Order Reduction LABoratory. In P. Benner & T. Breiten & H. Faßbender & M. Hinze & T. Stykel & R. Zimmermann, editor, *Model Reduction of Complex Dynamical Systems*, volume 171 of *International Series of Numerical Mathematics*, pages 393–415. Birkhäuser, Cham, 2021. doi: 10.1007/978-3-030-72983-7_19.

[90] G.I. Drakoulas, T.V. Gortsas, G.C. Bourantas, V.N. Burganos, and D. Polyzos. FastSVD-ML–ROM: A reduced-order modeling framework based on machine learning for real-time applications. *Computer Methods in Applied Mechanics and Engineering*, 414: 116155, Sep 2023.

[91] Mikhael TANNOUS, Chady Ghnatios, Eivind Fonn, Trond Kvamsdal, and Francisco Chinesta. Machine Learning (ML) Based Reduced Order Modelling (ROM) for Linear and Non-Linear Solid and Structural Mechanics. 2024.

[92] Afzal Sikander and Rajendra Prasad. A New Technique For Reduced-Order Modelling of Linear Time-Invariant System. *IETE Journal of Research*, 63(3):316–324, Jan 2017.

[93] Vladimir Puzyrev, Mehdi Ghommem, and Shiv Meka. pyROM: A Computational Framework for Reduced Order Modeling. *Journal of Computational Science*, 30:157–173, 2019.

[94] Jose Luis Gonzalez. Boost simulation with reduced order models using ansys twin builder, October 2023. URL https://play.vidyard.com/eQZU7c9vMcWsExv6z61Sr4. Senior Application Engineer, Digital Twin.

[95] Inc. Ansys. Ansys twin builder, 2025. URL https://www.ansys.com/products/digital-twin/ansys-twin-builder. Accessed: 2025-03-20.

[96] Altair. Altair romai elearning course, 2025. URL https://learn.altair.com/course/view.php?id=538. Accessed: 2025-03-20.

[97] Siemens Digital Industries Software. Simcenter reduced order modeling, 2023. URL https://plm.sw.siemens.com/en-US/simcenter/integration-solutions/reduced-order-modeling/. Accessed: 2025-03-20.

[98] Sudipta Saha, Syed Muhammad Amrr, Mashuq Un Nabi, and Atif Iqbal. Reduced Order Modeling and Sliding Mode Control of Active Magnetic Bearing. *IEEE Access*, 7:113324–113334, 2019.

[99] Zhongwei Deng & Xiaosong Hu & Xianke Lin & Le Xu & Jiacheng Li & Wenchao Guo. A Reduced-Order Electrochemical Model for All-Solid-State Batteries. *IEEE Transactions on Transportation Electrification*, 7(2):464–473, 2021.

[100] Qiaozhen Zhang, Zhenglin Chen, Yanguang Chen, Jiahe Dong, Panliang Tang, Sulei Fu, Haodong Wu, Jinyi Ma, and Xiangyong Zhao. Periodic Analysis of Surface Acoustic Wave Resonator with Dimensionally Reduced PDE Model Using COMSOL Code. *Micromachines*, 12(2):141, Jan 2021.

[101] Felice Di Nicola, Graziano Lonardi, Nicholas Fantuzzi, and Raimondo Luciano. Structural integrity assessment of an offshore platform using RB-FEA. *International Journal of Structural Integrity*, Aug 2024.

[102] Partha Sharma, David Knezevic, Phuong Huynh, and Grzegorz Malinowski. RB-FEA Based Digital Twin for Structural Integrity Assessment of Offshore Structures. In *Day 3 Wed, May 02, 2018*. OTC, Apr 2018.

[103] Omar Ghattas and Karen Willcox. Learning physics-based models from data: perspectives from inverse problems and model reduction. *Acta Numerica*, 30:445–554, 2021.

[104] J. Gu. QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30:1307–1320, 2011.

[105] P. Benner and T. Breiten. Two-sided projection methods for nonlinear model order reduction. *SIAM Journal on Scientific Computing*, 37:B239–B260, 2015.

[106] B. Kramer and K. Willcox. Nonlinear model order reduction via lifting transformations and proper orthogonal decomposition. *AIAA Journal*, 57:2297–2307, 2019.

[107] B. Kramer and K. Willcox. Balanced truncation model reduction for lifted nonlinear systems. In C. Beattie et al., editors, *Realization and Model Reduction of Dynamical Systems: A Festschrift in Honor of the 70th Birthday of Thanos Antoulas*. Springer, 2021. To appear.

[108] David Amsallem, Matthew J. Zahr, and Charbel Farhat. Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92(10):891–916, 2012.

[109] Juergen Hahn and Thomas F. Edgar. An improved method for nonlinear model reduction using balancing of empirical gramians. *Computers & Chemical Engineering*, 26:1379–1397, 2002.

[110] Ning Dong and Jaijeet Roychowdhury. Piecewise polynomial nonlinear model reduction. In *Proceedings of the 40th annual Design Automation Conference*, volume 46 of *DAC03*, page 484–489. ACM, 2003.

[111] Kevin Carlberg, Ray Tuminaro, and Paul Boggs. Preserving Lagrangian Structure in Nonlinear Model Reduction with Application to Structural Dynamics. *SIAM Journal on Scientific Computing*, 37:B153–B184, 2015.

[112] Saifon Chaturantabut and Danny C. Sorensen. A State Space Error Estimate for POD-DEIM Nonlinear Model Reduction. *SIAM Journal on Numerical Analysis*, 50: 46–63, 2012.

[113] Clarence W. Rowley and Scott T.M. Dawson. Model Reduction for Flow Analysis and Control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[114] A. Da Ronch, K. Badcock, Y. Wang, A. Wynn, and R. Palacios. Nonlinear Model Reduction for Flexible Aircraft Control Design. In *AIAA Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, 2012.

[115] Swaroop Mallick and Monika Mittal. Ai-based model order reduction techniques: A survey. *Archives of Computational Methods in Engineering*, Jan 2025.

[116] Cyril Touzé, Alessandra Vizzaccaro, and Olivier Thomas. Model order reduction methods for geometrically nonlinear structures: a review of nonlinear techniques. *Nonlinear Dynamics*, 105:1141–1190, 2021.

[117] S.W. Shaw and C. Pierre. Non-linear normal modes and invariant manifolds. *Journal of Sound and Vibration*, 150:170–173, 1991.

[118] Shyh-Leh Chen and Steven W. Shaw. Normal modes for piecewise linear vibratory systems. *Nonlinear Dynamics*, 10:135–164, 1996.

[119] Thomas Daniel, Fabien Casenave, Nissrine Akkari, Ali Ketata, and David Ryckelynck. Physics-informed cluster analysis and a priori efficiency criterion for the construction of local reduced-order bases. *Journal of Computational Physics*, 458:111120, 2022.

[120] Jeff Borggaard, Zhu Wang, and Lizette Zietsman. A goal-oriented reduced-order modeling approach for nonlinear systems. *Computers & Mathematics with Applications*, 71(11):2155–2169, 2016.

[121] J.B. Rutzmoser & D.J. Rixen & P. Tiso & S. Jain. Generalization of quadratic manifolds for reduced order modeling of nonlinear structural dynamics. *Computers & Structures*, 192:196–209, 2017.

[122] S. Jain, P. Tiso, J. B. Rutzmoser, and D. J. Rixen. A quadratic manifold for model order reduction of nonlinear structural dynamics. *Computers & Structures*, 188:80–94, 2017.

[123] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

[124] Alexander N. Gorban, Iliya V. Karlin, and Andrei Yu. Zinovyev. Constructive methods of invariant manifolds for kinetic problems. *Physics Reports*, 396:197–403, 2004.

[125] Xavier Cabré, Ernest Fontich, and Rafael de la Llave. The parameterization method for invariant manifolds III: overview and applications. *Journal of Differential Equations*, 218:444–515, 2005.

[126] Carles Sim'o. On the Analytical and Numerical Approximation of Invariant Manifolds. In Daniel Benest and Claude Froeschl'e, editors, *Les M'ethodes Modernes de la M'ecanique C'eleste*, pages 285–329. Editions Fronti'eres, Paris, 1990.

[127] John Guckenheimer and Alexander Vladimirsky. A Fast Method for Approximating Invariant Manifolds. *SIAM Journal on Applied Dynamical Systems*, 3:232–260, 2004.

[128] Uri M. Ascher, Hongsheng Chin, and Sebastian Reich. Stabilization of DAEs and invariant manifolds. *Numerische Mathematik*, 67:131–149, 1994.

[129] Zein Alabidin Shami, Yichang Shen, Christophe Giraud-Audine, Cyril Touzé, and Olivier Thomas. Nonlinear dynamics of coupled oscillators in 1:2 internal resonance: effects of the non-resonant quadratic terms and recovery of the saturation effect. *Meccanica*, 57:2701–2731, 2022.

[130] W. Lacarbonara, G. Rega, and A.H. Nayfeh. Resonant non-linear normal modes. Part I: analytical treatment for structural one-dimensional systems. *International Journal of Non-Linear Mechanics*, 38:851–872, 2003.

[131] David D. Nolte. The tangled tale of phase space. *Physics Today*, 63:33–38, 2010.

[132] Mattia Cenedese & Joar Axås & Bastian Bäuerlein & Kerstin Avila & George Haller. Data-driven modeling and prediction of non-linearizable dynamics via spectral submanifolds. *Nat. Commun.*, 13(872):1–13, Feb 2022.

[133] Mingwu Li, Shobhit Jain, and George Haller. Model reduction for constrained mechanical systems via spectral submanifolds. *Nonlinear Dynamics*, 111(10):8881–8911, Feb 2023.

[134] Mingwu Li, Thomas Thurnher, Zhenwei Xu, and Shobhit Jain. Data-free non-intrusive model reduction for nonlinear finite element models via spectral submanifolds. *Computer Methods in Applied Mechanics and Engineering*, 434:117590, Feb 2025.

[135] Jincong He and Louis J. Durlofsky. Reduced-Order Modeling for Compositional Simulation by Use of Trajectory Piecewise Linearization. *SPE J.*, 19(05):858–872, Mar 2014.

[136] Shifali Kalra and Mashuq un Nabi. Automated scheme for linearisation points selection in tpwl method applied to non-linear circuits. *The Journal of Engineering*, 2019(20):7150–7154, 2019.

[137] Xiaosi Tan & Eduardo Gildin & Horacio Florez & Sumeet Trehan & Yahan Yang & Nazish Hoda. Trajectory-based DEIM (TDEIM) model reduction applied to reservoir simulation. *Computational Geosciences*, 23(1):35–53, Oct 2018.

[138] Jincong He and Louis J. Durlofsky. Reduced-order modeling for compositional simulation by use of trajectory piecewise linearization. *SPE Journal*, 19(05):858–872, 03 2014.

[139] M. Rewienski and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):155–170, 2003.

[140] Tianshu Wen and Matthew J. Zahr. An augmented lagrangian trust-region method with inexact gradient evaluations to accelerate constrained optimization problems

using model hyperreduction. *International Journal for Numerical Methods in Fluids*, 97 (4):621–645, Dec 2024.

[141] Han Gao and Matthew J. Zahr. An adaptive model reduction method leveraging locally supported basis functions. *International Journal of Computational Fluid Dynamics*, 37(6):451–473, Jul 2023.

[142] Tianshu Wen and Matthew J. Zahr. A globally convergent method to accelerate large-scale optimization using on-the-fly model hyperreduction: Application to shape optimization. *Journal of Computational Physics*, 484:112082, Jul 2023.

[143] Marzieh Alireza Mirhoseini and Matthew J. Zahr. Accelerated solutions of convection-dominated partial differential equations using implicit feature tracking and empirical quadrature. *International Journal for Numerical Methods in Fluids*, 96(1):102–124, Sep 2023.

[144] Rui Jiang and Louis J. Durlofsky. Implementation and detailed assessment of a GNAT reduced-order model for subsurface flow simulation. *Journal of Computational Physics*, 379:192–213, 2019.

[145] Feng Bai and Yi Wang. A Reduced Order Modeling Method Based on GNAT-Embedded Hybrid Snapshot Simulation. *Mathematics and Computers in Simulation*, 199:100–132, 2022.

[146] Jan Dirk Jansen and Louis J. Durlofsky. Use of reduced-order models in well control optimization. *Optimization and Engineering*, 18:105–132, 2016.

[147] Mohamadreza Ghasemi and Eduardo Gildin. Localized model order reduction in porous media flow simulation. *Journal of Petroleum Science and Engineering*, 145: 689–703, 2016.

[148] Seonkyoo Yoon & Zeid M. Alghareeb & John R. Williams. Hyper-Reduced-Order Models for Subsurface Flow Simulation. *SPE Journal*, 21:2128–2140, 2016.

[149] Rui Jiang & Louis J. Durlofsky. Implementation and detailed assessment of a GNAT reduced-order model for subsurface flow simulation. *Journal of Computational Physics*, 379:192–213, Feb 2019.

[150] Mehdi Ghommem, Eduardo Gildin, and Mohammadreza Ghasemi. Complexity Reduction of Multiphase Flows in Heterogeneous Porous Media. *SPE J.*, 21(01): 144–151, Feb 2016.

[151] Mehdi Dehghan and Mostafa Abbaszadeh. A combination of proper orthogonal decomposition–discrete empirical interpolation method (pod–deim) and meshless local rbf-dq approach for prevention of groundwater contamination. *Computers & Mathematics with Applications*, 75(4):1390–1412, Feb 2018.

[152] Péter German, Jean C. Ragusa, and Carlo Fiorina. Application of Multiphysics Model Order Reduction to Doppler/Neutronic Feedback. *EPJ Nuclear Sciences & Technologies*, 5:17, 2019.

[153] Péter German, Mauricio Tano, Jean C. Ragusa, and Carlo Fiorina. Comparison of reduced-basis techniques for the model order reduction of parametric incompressible fluid flows. *Progress in Nuclear Energy*, 130:103551, 2020. ISSN 0149-1970. doi: https://doi.org/10.1016/j.pnucene.2020.103551.

[154] Péter German, Mauricio Tano, Carlo Fiorina, and Jean C. Ragusa. GeN-ROM—an OpenFOAM©-based multiphysics reduced-order modeling framework for the analysis of molten salt reactors. *Progress in Nuclear Energy*, 146:104148, 2022. ISSN 0149-1970. doi: https://doi.org/10.1016/j.pnucene.2022.104148.

[155] Mauricio Tano, Peter German, and Jean Ragusa. Evaluation of pressure reconstruction techniques for Model Order Reduction in incompressible convective heat transfer. *Thermal Science and Engineering Progress*, 23:100841, 2021.

[156] Quincy Huhn, Suparno Bhattacharyya, and Jean Ragusa. Hyperreduction for neutron transport. In *Proceedings of the 2024 ANS Winter Conference and Expo*, volume 131, pages 462–465. American Nuclear Society, Nov 2024.

[157] Paolo Tiso and Daniel J. Rixen. *Discrete Empirical Interpolation Method for Finite Element Structural Dynamics*, page 203–212. Springer New York, 2013.

[158] Harbir Antil, Matthias Heinkenschloss, and Danny C. Sorensen. *Application of the Discrete Empirical Interpolation Method to Reduced Order Modeling of Nonlinear and Parametric Systems*, page 101–136. Springer International Publishing, 2014.

[159] J. A. Hernández, M. A. Caicedo, and A. Ferrer. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Computer Methods in Applied Mechanics and Engineering*, 313:687–722, 2017.

[160] Charbel Farhat & Sebastian Grimberg & Andrea Manzoni & Alfio Quarteroni. *Computational bottlenecks for PROMs: precomputation and hyperreduction*, pages 181–244. De Gruyter, Berlin, Boston, 2021.

[161] Miguel Fosas de Pando, Peter J. Schmid, and Denis Sipp. Nonlinear model-order reduction for compressible flow solvers using the discrete empirical interpolation method. *Journal of Computational Physics*, 324:194–209, Nov 2016.

[162] David Amsallem, Matthew J. Zahr, and Kyle Washabaugh. Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction. *Advances in Computational Mathematics*, 41(5):1187–1230, Feb 2015.

[163] Angira Sharma, Edward Kosasih, Jie Zhang, Alexandra Brintrup, and Anisoara Calinescu. Digital Twins: State of the art theory and practice, challenges, and open research questions. *Journal of Industrial Information Integration*, 30:100383, 2022.

[164] Diego M. Botín-Sanabria, Adriana-Simona Mihaita, Rodrigo E. Peimbert-García, Mauricio A. Ramírez-Moreno, Ricardo A. Ramírez-Mendoza, and Jorge de J. Lozoya-Santos. Digital Twin Technology Challenges and Applications: A Comprehensive Review. *Remote Sensing*, 14:1335, 2022.

[165] Mengnan Liu, Shuiliang Fang, Huiyue Dong, and Cunzhi Xu. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361, 2021.

[166] Michael G Kapteyn, David J Knezevic, and Karen Willcox. Toward predictive digital twins via component-based reduced-order models and interpretable machine learning. In *AIAA Scitech 2020 Forum*, page 0418, 2020.

[167] National Academy of Engineering and National Academies of Sciences, Engineering, and Medicine. *Foundational Research Gaps and Future Directions for Digital Twins*. The National Academies Press, Washington, DC, 2024.

[168] José V. Aguado, Antonio Huerta, Francisco Chinesta, and Elías Cueto. Real-time monitoring of thermal processes by reduced-order modeling. *International Journal for Numerical Methods in Engineering*, 102(5):991–1017, Oct 2014.

[169] Yongse Kim, Seung-Hoon Kang, Haeseong Cho, and SangJoon Shin. Improved Nonlinear Analysis of a Propeller Blade Based on Hyper-Reduction. *AIAA Journal*, 60 (3):1909–1922, Mar 2022.

[170] *Model Order Reduction*. De Gruyter, Berlin, Boston, 2021.

[171] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[172] Hongfei Fu, Hong Wang, and Zhu Wang. POD/DEIM reduced-order modeling of time-fractional partial differential equations with applications in parameter identification. *Journal of Scientific Computing*, 74(1):220–243, 2018.

[173] Ngoc Cuong Nguyen. *Model reduction techniques for parametrized nonlinear partial differential equations*, page 149–204. Elsevier, 2024. ISBN 9780443294488.

[174] Saeed Asgari & Xiao Hu & Michael Tsuk & Shailendra Kaushik. Application of POD plus LTI ROM to Battery Thermal Modeling: SISO Case. *SAE International Journal of Commercial Vehicles*, 7:278–285, 2014.

[175] Vladyslav Pliuhin, Yevgen Tsegelnyk, Sergiy Plankovskyy, Oleksandr Aksonov, and Volodymyr Kombarov. *Implementation of Induction Motor Speed and Torque Control System with Reduced Order Model in ANSYS Twin Builder*, page 514–531. Springer Nature Switzerland, 2023.

[176] Siemens Digital Industries Software. Simcenter STAR-CCM+ User Guide, version 2021.1. In *Adaptive Mesh Refinement for Overset Meshes*, pages 3067–3070. Siemens, 2021.

[177] Olle Lindqvist. A Method of CFD-based System Identification for Standardized Co-Simulation. Master's thesis, Linköping University, Fluid and Mechatronic Systems, 2022.

[178] Suparno Bhattacharyya, Jian Tao, and Jean Ragusa. pyhyperrom, 2025. URL https://github.com/TAMIDS-HYPERREDUCTION/pyhyperrom. Accessed: 2025-02-03.

[179] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, Nov 2002.

[180] Lihong Feng, Jan G. Korvink, and Peter Benner. A fully adaptive scheme for model order reduction based on moment matching. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 5(12):1872–1884, Dec 2015.

[181] Karl Kunisch and Stefan Volkwein. Proper orthogonal decomposition for optimality systems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(1):1–23, Jan 2008.

[182] Suparno Bhattacharyya and Joseph P. Cusumano. An Energy Closure Criterion for Model Reduction of a Kicked Euler–Bernoulli Beam. *Journal of Vibration and Acoustics*, 143:041001, 2020.

[183] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4:519–524, 1987.

[184] Charbel Farhat. CME 345: Projection-Based Model Order Reduction, 2024. URL https://web.stanford.edu/group/frg/course_work/CME345.html. Accessed: 2024-10-16.

[185] Pierfrancesco Siena, Michele Girfoglio, and Gianluigi Rozza. Chapter 6 - an introduction to pod-greedy-galerkin reduced basis method. In Francisco Chinesta, Elías Cueto, Yohan Payan, and Jacques Ohayon, editors, *Reduced Order Models for the Biomechanics of Living Organs*, Biomechanics of Living Organs, pages 127–145. Academic Press, 2023.

[186] Bernard Haasdonk. Reduced basis methods for parametrized PDEs – a tutorial introduction for stationary and instationary problems. In Peter Benner, Albert Cohen, Mario Ohlberger, and Karen Willcox, editors, *Model Reduction and Approximation: Theory and Algorithms*, pages 65–136. SIAM, Philadelphia, 2017. doi: 10.1137/1.9781611974829.ch2.

[187] Mehran Ebrahimi and Masayuki Yano. A hyperreduced reduced basis element method for reduced-order modeling of component-based nonlinear systems. *Computer Methods in Applied Mechanics and Engineering*, 431:117254, Nov 2024.

[188] Peter Benner and Lihong Feng. Model reduction for dynamical systems – lecture 11. https://www.mpi-magdeburg.mpg.de/2956437/Lecture-11.pdf, 2015. Lecture notes, Computational Methods in Systems and Control Theory, Summer term 2015.

[189] Charbel Farhat, Sebastian Grimberg, Andrea Manzoni, Alfio Quarteroni, et al. Computational bottlenecks for proms: precomputation and hyperreduction. *Model order reduction, Berlin: De Gruyter*, pages 181–244, 2020.

[190] I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, Jan 1967.

[191] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.

[192] R. Everson and L. Sirovich. Karhunen–Loève procedure for gappy data. *J. Opt. Soc. Am. A*, 12:1657–1664, 1995.

[193] Patricia Astrid, Siep Weiland, Karen Willcox, and Ton Backx. Missing Point Estimation in Models Described by Proper Orthogonal Decomposition. *IEEE Transactions on Automatic Control*, 53:2237–2251, 2008.

[194] Hector Gomez and Laura De Lorenzis. The variational collocation method. *Computer Methods in Applied Mechanics and Engineering*, 309:152–181, 2016.

[195] Todd Chapman, Philip Avery, Pat Collins, and Charbel Farhat. Accelerated mesh sampling for the hyper reduction of nonlinear computational models. *International Journal for Numerical Methods in Engineering*, 2016.

[196] Jessica T. Lauzon, Siu Wun Cheung, Yeonjong Shin, Choi, and Kevin Huynh. S-OPT: A Points Selection Algorithm for Hyper-Reduction in Reduced Order Models. *SIAM Journal on Scientific Computing*, 46(4):B474–b501, 2024.

[197] Anthony T. Patera and Masayuki Yano. An LP empirical quadrature procedure for parametrized functions. *Comptes Rendus. Mathématique*, 355(11):1161–1167, Nov 2017. ISSN 1778-3569.

[198] Benjamin Peherstorfer, Zlatko Drmač, and Serkan Gugercin. Stability of Discrete Empirical Interpolation and Gappy Proper Orthogonal Decomposition with Randomized and Deterministic Sampling Points. *SIAM Journal on Scientific Computing*, 42: A2837–A2864, 2020.

[199] Felix Fritzen, Bernard Haasdonk, David Ryckelynck, and Sebastian Schöps. An Algorithmic Comparison of the Hyper-Reduction and the Discrete Empirical Interpolation Method for a Nonlinear Thermal Problem. *Mathematical and Computational Applications*, 23(1):8, Feb 2018.

[200] Yvon Maday and Olga Mula. *A Generalized Empirical Interpolation Method: Application of Reduced Basis Techniques to Data Assimilation*, page 221–235. Springer Milan, 2013.

[201] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, Hoboken, NJ, 5th edition, 2012.

[202] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, Jul 1955.

[203] Friedrich Pukelsheim. *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics, Jan 2006.

[204] Diana Bonomi, Andrea Manzoni, and Alfio Quarteroni. A matrix DEIM technique for model reduction of nonlinear parametrized problems in cardiac mechanics. *Computer Methods in Applied Mechanics and Engineering*, 324:300–326, 2017.

[205] Feng Bai and Yi Wang. Deim-embedded hybrid snapshot simulation for reduced order model generation. *Engineering Computations*, 39(10):3321–3353, Nov 2022.

[206] Arvind K. Saibaba. Randomized discrete empirical interpolation method for nonlinear model reduction. *SIAM Journal on Scientific Computing*, 42(3):A1582–A1608, 2020.

[207] Zlatko Drmač and Arvind K. Saibaba. The discrete empirical interpolation method: Canonical structure and formulation in weighted inner product spaces. *SIAM Journal on Scientific Computing*, 40(2):A703–A731, 2018.

[208] Emily P. Hendryx, Béatrice M. Rivière, and Craig G. Rusin. An extended deim algorithm for subset selection and class identification. *Machine Learning*, 110(4): 621–650, Mar 2021.

[209] Cecilia Pagliantini and Federico Vismara. Gradient-preserving hyper-reduction of nonlinear dynamical systems via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 45(5):A2725–A2754, Oct 2023.

[210] R.B. Klein and B. Sanderse. Energy-conserving hyper-reduction and temporal localization for reduced order models of the incompressible navier-stokes equations. *Journal of Computational Physics*, 499:112697, Feb 2024.

[211] Yeonjong Shin and Dongbin Xiu. On a near optimal sampling strategy for least squares polynomial regression. *J. Comput. Phys.*, 326:931–946, Dec 2016.

[212] Jacques Hadamard. Résolution d'une question relative aux déterminants. *Bulletin des Sciences Mathématiques*, 17:240–246, 1893.

[213] Kevin Carlberg, Matthew Barone, and Harbir Antil. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.

[214] S. Ares de Parga, J.R. Bravo, J.A. Hernández, R. Zorrilla, and R. Rossi. Hyper-reduction for petrov–galerkin reduced order models. *Computer Methods in Applied Mechanics and Engineering*, 416:116298, Nov 2023.

[215] S.S. An, T. Kim, and D.L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM transactions on graphics*, 27(5):165, 2009.

[216] Theodore Kim and John Delaney. Subspace fluid re-simulation. *ACM Transactions on Graphics (TOG)*, 32(4):62, 2013.

[217] Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. An efficient construction of reduced deformable objects. *ACM Transactions on Graphics (TOG)*, 32(6):213, 2013.

[218] Zherong Pan, Hujun Bao, and Jin Huang. Subspace dynamic simulation using rotation-strain coordinates. *ACM Transactions on Graphics (TOG)*, 34(6):242, 2015.

[219] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. ISBN 0-89871-356-0. Revised reprint of the 1974 original.

[220] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, Tim Cournapeau, ..., Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, Yoshiki Vázquez-Baeza, and SciPy 1.0 Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods*, 17(3):261–272, Mar 2020.

[221] Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, Sep 1997.

[222] Rbravo555 - GitHub. https://github.com/Rbravo555, 2024. Accessed: 2024-11-05.

[223] Ludovica Cicci, Stefania Fresca, and Andrea Manzoni. Deep-HyROMnet: A Deep Learning-Based Operator Approximation for Hyper-Reduction of Nonlinear Parametrized PDEs. *Journal of Scientific Computing*, 93, 2022.

[224] Francesco Romor, Giovanni Stabile, and Gianluigi Rozza. Explicable hyper-reduced order models on nonlinearly approximated solution manifolds of compressible and incompressible Navier-Stokes equations. *J. Comput. Phys.*, 524:113729, Mar 2025.

[225] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[226] Sara Grundel and Neeraj Sarna. Hyper-reduction for parametrized transport dominated problems via adaptive reduced meshes. *Partial Differential Equations and Applications*, 5(1), Feb 2024.

[227] R.B. Klein and B. Sanderse. Energy-conserving hyper-reduction and temporal localization for reduced order models of the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 499:112697, 2024.

[228] Masayuki Yano. Discontinuous galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws. *Advances in Computational Mathematics*, 45(5–6):2287–2320, Jun 2019.

[229] Joshua Barnett, Charbel Farhat, and Yvon Maday. Neural-network-augmented projection-based model order reduction for mitigating the Kolmogorov barrier to reducibility. *Journal of Computational Physics*, 492:112420, 2023.

# A Key technical jargon

- **Subspace:** A subspace is a subset of a vector space that is itself a vector space under the same operations. A subspace must contain the zero vector, be closed under addition, and be closed under scalar multiplication.

- **Affine Subspace:** An affine subspace is a subset of a vector space that is closed under affine combinations. It can be viewed as a linear subspace that has been translated from the origin.
- **Manifold:** A manifold is a mathematical space that locally resembles Euclidean space near each point. Manifolds are used to generalize concepts such as curves and surfaces to higher dimensions.
- **Latent Space:** A latent space is a lower-dimensional space that captures the essential features of the data. For example, subspace spanned by the dominant POD modes. In a neural network setting, encoders map data to this space and decoders reconstruct the data from it.
- **Modal Energies (SVD):** In singular value decomposition (SVD), modal *energies* refer to singular values, which represent the amount of **variance** captured by each mode in the data. The term *energy* is essentially a misnomer, which often as nothing to do with the physical energy of the system [182].
- **Offline and Online:** In model reduction, "offline" refers to computations performed prior to real-time use, often involving expensive and time-consuming tasks associated with formulating a data-driven model. "Online" refers to computations done in real-time, typically leveraging precomputed data to achieve fast and efficient results.
- **Residual:** The residual is the difference between the left and right sides of a differential equation after substituting an approximate solution. It provides a measure of how well the approximate solution satisfies the original equation.
- **Snapshots:** In model reduction, snapshots are sample solutions of the full-order model used to construct a reduced-order basis. These solutions are typically obtained by solving the original problem for different parameter values and/or at different time instants.
- **Affine Decomposition:** Affine decomposition is a mathematical technique commonly used in reduced order modeling (ROM), particularly for parametric problems. It separates the parametric dependencies in equations, enabling efficient computations for a wide range of parameter values.

  **Parametric Separation**:

  - A parametric problem is expressed in a form where the parameter dependency is isolated.

  - For example, consider a linear problem:

$$A(\boldsymbol{\mu})\mathbf{u} = \mathbf{f}(\boldsymbol{\mu})\,, \tag{126}$$

  where $\boldsymbol{\mu}$ is a vector of parameters. Using affine decomposition, $A(\boldsymbol{\mu})$ and $\mathbf{f}(\boldsymbol{\mu})$ are represented as:

$$A(\boldsymbol{\mu}) = \sum_{q=1}^{Q} \Theta_q^A(\boldsymbol{\mu}) A_q, \tag{127}$$

$$\mathbf{f}(\boldsymbol{\mu}) = \sum_{q=1}^{Q} \Theta_q^f(\boldsymbol{\mu}) \mathbf{f}_q. \tag{128}$$

**Offline-Online Decoupling**:

- *Offline Phase*: Parameter-independent terms ($A_q$ and $\mathbf{f}_q$) are precomputed and stored.

- *Online Phase*: For a new parameter $\boldsymbol{\mu}$, the computational effort involves only evaluating the parameter-dependent functions ($\Theta_q^A(\boldsymbol{\mu})$ and $\Theta_q^f(\boldsymbol{\mu})$) and combining the precomputed terms.

- **Galerkin Projection:** The Galerkin projection is a method for approximating the solution of differential equations by projecting the governing equation onto a finite-dimensional subspace. Let $V$ be the function space in which the exact solution resides, and let $V_r$ be a finite-dimensional subspace spanned by basis functions $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_r\}$. The approximate solution $u_r \in V_r$ is determined such that the residual $R(u_r)$ is orthogonal to $V_r$ with respect to an inner product:

$$\text{Find } u_r \in V_r \text{ such that } (R(u_r), \phi_i) = 0, \quad \forall i = 1, 2, \ldots, r,$$

  where $R(u_r)$ is the residual, and $(\cdot, \cdot)$ denotes the inner product. This ensures that the error is minimized in the subspace $V_r$, leading to an optimal projection of the true solution onto the chosen basis.

- **Petrov-Galerkin Projection:** Petrov-Galerkin projection is a generalization of the Galerkin method, where the trial and test spaces are different. Given a trial space $V_r$ and a test space $W_r$, the Petrov-Galerkin method finds an approximate solution $u_r$ in $V_r$ such that the residual $R(u_r)$ is orthogonal to the test space $W_r$:

$$\text{Find } u_r \in V_r \text{ such that } (R(u_r), \psi_i) = 0, \quad \forall\, i = 1, 2, \ldots, r\,, \tag{129}$$

  where $\psi_i$ are the basis functions of the test space $W_r$. This approach is often used to improve stability and accuracy, especially in non-symmetric or convection-dominated problems.

- **Left ROB and right ROB:** In projection-based model reduction, the *right* reduced-order basis (ROB), denoted in this paper as $\mathbb{U}$, is associated with the solution (trial) space and is used to map from a low-dimensional approximation back into the full-dimensional state space. Conversely, the *left* ROB, denoted as $\mathbb{W}$, is associated with the test space (e.g., in a Petrov–Galerkin setting), and it maps vectors from the full-dimensional space down to the reduced dimension.

- **Weak Form and Variational Problem:** The weak form of a partial differential equation (PDE) is obtained by multiplying the PDE $\mathcal{L}(u) = f$ by a test function $v$, integrating over $\Omega$, and applying integration by parts or the divergence theorem to relax smoothness requirements on $u$. This leads to the variational problem:

$$\text{Find } u \in V \text{ such that } \int_\Omega \mathcal{L}(u) v \, d\Omega = \int_\Omega f v \, d\Omega \quad \forall\, v \in V\,, \tag{130}$$

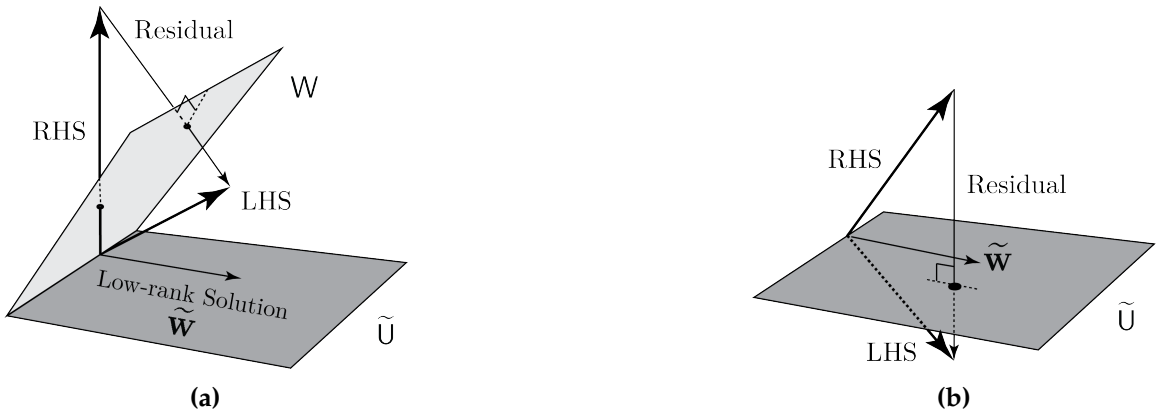



**Figure 23:** (a) Petrov-Galerkin Projection (b) Galerkin Projection

where $V$ is an appropriate function space (e.g., a Sobolev space). Instead of enforcing the PDE pointwise, the weak form ensures the equation holds in an integral sense for all $v$, broadening the class of admissible solutions and forming the foundation for numerical methods such as the finite element method (FEM).

- **Oblique Projection Matrix**: An oblique projection matrix is a linear transformation matrix $P$ that projects vectors onto a subspace $M$ along a direction that is not necessarily perpendicular to $M$. Given matrices $\mathsf{A}$ and $\mathsf{B}$:

$$\mathsf{P} = \mathsf{A}(\mathsf{B}^\top \mathsf{A})^{-1}\mathsf{B}^\top \tag{131}$$

provided $\mathsf{B}^\top \mathsf{A}$ is invertible.

# B Algorithms

## B.1 Empirical interpolation method (EIM)

Given a *continuous* nonlinear function $f : \Omega \times \mathcal{P}_{\text{EIM}} \to \mathbb{R}$, where $\Omega$ is the spatial domain and $\mathcal{P}_{\text{EIM}}$ is the parameter space, EIM aims to approximate $f$ using a finite sum of basis functions $\{U_q\}_{q=1}^{Q}$ evaluated at specific interpolation points $\{x_q\}_{q=1}^{Q}$:

$$f(x, w^\mu; \mu) \approx \sum_{q=1}^{Q} c_q(\mu) U_q(x)\,, \tag{132}$$

where the coefficients $c_q(\mu)$ are determined by enforcing the interpolation conditions:

$$f(x_i, w^\mu; \mu) = \sum_{q=1}^{Q} c_q(\mu) U_q(x_i), \quad \text{for } i = 1, \ldots, Q\,. \tag{133}$$

This approach reduces the problem of evaluating $f$ over the entire domain $\Omega$ to computing it at a finite number of points $\{x_q\}$, thus enhancing computational efficiency [69].

**Algorithm 2** EIM Algorithm for Nonlinear Term Approximation [68]

**Input:** $f$, $\Omega$, $\mathcal{P}_{\text{EIM}}$, $tol$
**Output**: Basis functions $\{U_q\}_{q=1}^{Q}$, Interpolation points $\{x_q\}_{q=1}^{Q}$
Initialization:
**Set:** iteration counter $q = 1$
**Initialize:** the interpolation operator $\mathcal{I}_0[f] = 0$
**Collect:** snapshots $[f(x, w^{\mu_1}; \mu_1), \ldots, f(x, w^{\mu_Q}; \mu_Q)]$ evaluated at selected parameters
**while** $err_q > tol$ AND $q \leq Q$ **do**
  $\mu_q = \arg\max_{\mu \in \mathcal{P}_{\text{EIM}}} \left\| f(\cdot, w^{\mu}; \mu) - \mathcal{I}_{q-1}[f(\cdot, w^{\mu}; \mu)] \right\|_{L^\infty(\Omega)}$ // Selection of Parameter
  $x_q = \arg\max_{x \in \Omega} \left| f(x, w^{\mu_q}; \mu_q) - \mathcal{I}_{q-1}[f(x, w^{\mu_q}; \mu_q)] \right|$ // Selection of Interpolation Point:
  $U_q(x) = \dfrac{f(x, w^{\mu_q}; \mu_q) - \mathcal{I}_{q-1}[f(x, w^{\mu_q}; \mu_q)]}{f(x_q, w^{\mu_q}; \mu_q) - \mathcal{I}_{q-1}[f(x_q, w^{\mu_q}; \mu_q)]}$ // Construction of Basis Function:
  $T_{ij} = U_j(x_i)$ satisfying $T_{ii} = 1$ and $T_{ij} = 0$ for $1 \leq i < j$ // Properties of Interpolation Matrix:
  $\mathcal{I}_q[f(x, w^{\mu}; \mu)] = \sum_{j=1}^{q} c_j(\mu) U_j(x)$ // Update of Interpolation Operator, where $\mathbf{c}(\mu)$ is solved using the linear system $\mathsf{T}\mathbf{c}(\mu) = \mathbf{f}(\mu)$
  $err_q = \sup_{\mu \in \mathcal{P}_{\text{EIM}}} \left\| f(\cdot, w^{\mu}; \mu) - \mathcal{I}_q[f(\cdot, w^{\mu}; \mu)] \right\|_{L^\infty(\Omega)}$ // check error
  **if** $err_q > tol$ **then**
    increment $q = q + 1$
  **else**
    exit loop
  **end if**
**end while**

## B.2 S-OPT sampling algorithm

**Algorithm 3** S-OPT Sampling Algorithm (Adapted from [196]).

**Input:** Basis matrix $\mathsf{F}_{\text{basis}}$ of shape $(n_{\text{rows}}, n_f)$
**QR Factorization** - Obtain orthogonal basis $\mathsf{Q}$ from $\mathsf{F}_{\text{basis}}$ // Perform QR factorization on the input basis
$\mathsf{Q}, \_ = \text{qr}(\mathsf{F}_{\text{basis}})$ // Q is an orthogonal matrix representing the basis
**Initialize** $\mathcal{Z} = \{i^*\}$ where $i^* = \arg\max_i |\mathsf{Q}_{i,1}|$ // Select the index with the largest measure of $S$
**for** $j = 1$ **to** $n_f - 1$ **do**
  Construct $\mathsf{Z} = [e_i]_{i\in\mathcal{Z}}$, $\mathsf{A} = \mathsf{Z}^\top\mathsf{Q}[\mathsf{E}_1, \cdots, \mathsf{E}_j]$, $\mathbf{c} = \mathsf{Z}^\top\mathsf{Q}e_{j+1}$, and $\mathbf{g} = (\mathsf{A}^\top\mathsf{A})^{-1}\mathsf{A}^\top\mathbf{c}$ // Construct necessary components, where $e \in \mathcal{R}^{n_{\text{rows}}}$ and $\mathsf{E} \in \mathcal{R}^{n_{f\text{rows}}\times 1}$
  Find $i^* = \arg\max_{i\notin\mathcal{Z}} \dfrac{1+\mathbf{r}^\top\mathbf{b}}{\prod_{k=1}^{j}(\|\mathsf{A}\mathsf{e}_\mathsf{k}\|^2 + r_k^2)} \dfrac{\mathbf{c}^\top\mathbf{c}+\gamma^2-\alpha}{\mathbf{c}^\top\mathbf{c}+\gamma^2}$ // Identify the index with the optimal value
  where $\mathbf{r}^\top = e_i^\top\mathsf{Q}[\mathsf{E}_1, \cdots, \mathsf{E}_j]$, $\mathbf{b} = (\mathsf{A}^\top\mathsf{A})^{-1}\mathbf{r}$, $\gamma = \mathsf{Q}_{i,j+1}$, $\alpha = (\mathbf{c}^\top\mathsf{A} + \gamma\mathbf{r}^\top)\left(\mathsf{I} - \dfrac{\mathbf{b}\mathbf{r}^\top}{1+\mathbf{r}^\top\mathbf{b}}\right)(\mathbf{g}+\gamma\mathbf{b})$ // Calculate necessary components for optimization
  Enrich $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{i^*\}$ // Add the new index to the sampling set
**end for**
**Output** $\mathcal{Z}$ // Oversampling Procedure
**Initialize** $\mathcal{Z} = \{i^*\}$ where $i^* = \arg\max_i |\mathsf{Q}_{i,1}|$ // Select the index with the largest measure of $S$
**for** $j = 1$ **to** $n_f - 1$ **do**
  Construct $\mathsf{Z} = [e_i]_{i\in\mathcal{Z}}$ and $\mathsf{A} = \mathsf{Z}^\top\mathsf{Q}$ // Construct the current basis matrix, where $e \in \mathcal{R}^{n_{\text{rows}}}$
  Find $i^* = \arg\max_{i\notin\mathcal{Z}} \dfrac{1+\mathbf{r}^\top(\mathsf{A}^\top\mathsf{A})^{-1}\mathbf{r}}{\prod_{k=1}^{n_f}(\|\mathsf{A}\mathsf{e}_\mathsf{k}\|^2 + r_k^2)}$ with $\mathbf{r}^\top = e_i^\top\mathsf{Q}$ // Select index based on the oversampling criterion
  Enrich $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{i^*\}$ // Add the new index to the sampling set
**end for**
**Output** $\mathcal{Z}$

## B.3 Point Selection in empirical cubature method (ECM)

**Algorithm 4** Empirical Cubature Method (Reproduced from [76])

**Function** $[\varepsilon, \omega] = \text{ECM}(\widetilde{\mathbf{W}}, \mathbf{W}_{FE}, y^0)$
**Data**: $\widetilde{\mathbf{W}} \in \mathcal{R}^{K\times M}$, where $\widetilde{\mathbf{W}}^\top\widetilde{\mathbf{W}} = I$; $\mathbf{W}_{FE} \in \mathcal{R}^{M\times 1}$; $y^0 \subset \{1, 2, ..., M\}$: initial candidates; $\lambda \leftarrow 10$, threshold number of ineffective iterations
**Result**: $\varepsilon \subset \{1, 2, ..., M\}$; $\omega > 0$ such that $\widetilde{\mathbf{W}}(\varepsilon, :)^\top\omega = \widetilde{\mathbf{W}}^\top\mathbf{W}_{FE}$ and $\text{card}(\varepsilon \cap y^0)$ is maximum.

**if** $y^0 = \emptyset$ // no initial candidate set given **then**
  $y \leftarrow \{1, 2, ..., M\}$
  $y \leftarrow \{h_1, h_2, \dots\}$ such that $\|\widetilde{\mathbb{W}}(h_i, :)\| \leq \varepsilon$ // Remove low norm points on the candidate set ($\sim 10^{-10}$)
**else**
  $y' \leftarrow \{1, 2, ..., M\} \setminus y^0$;  $y \leftarrow y^0$
  $y' \leftarrow \{h_1, h_2, \dots\}$ such that $\|\widetilde{\mathbb{W}}(h_i, :)\| \leq \varepsilon$ // Remove low norm points on the complement set ($\sim 10^{-10}$)
**end if**
$\varepsilon \leftarrow \emptyset$; $\mathbf{b} \leftarrow \widetilde{\mathbb{W}}^\top \mathbf{W}_{FE}$; $\mathbf{r} \leftarrow \mathbf{b}$; $\boldsymbol{\omega} \leftarrow \emptyset$; $H \leftarrow \emptyset$ // Initializations
$\phi \leftarrow 0$ // Failed iterations counter
**while** length$(\varepsilon) < p$ **AND** length$(y) > 0$ **do**
  **if** $\phi > \lambda$ **then**
    $y \leftarrow y \cup y'$ // Enlarge candidate set to include the complement set
  **end if**
  $j = \arg\max_{j \in y} \mathbf{g}_j^\top \mathbf{r}$, where $\mathbf{g}_j = \widetilde{\mathbb{W}}(j, :) / \|\widetilde{\mathbb{W}}(j, :)\|$ // Select the row most "positively" parallel to the residual
  **if** $\varepsilon = 0$ **then**
    $\mathsf{H} \leftarrow (\widetilde{\mathbb{W}}(i, :)\widetilde{\mathbb{W}}(i, :)^\top)^{-1}$ // Inverse Hermitian matrix (first iteration)
    $\boldsymbol{\omega} \leftarrow \mathsf{H}\widetilde{\mathbb{W}}(i, :)^\top \mathbf{b}$ // Weights computed through least-squares
  **else**
    $[\boldsymbol{\omega}, \mathsf{H}] \leftarrow \text{LSTQNER}(i, \mathsf{H}, \widetilde{\mathbb{W}}(\varepsilon, :), \widetilde{\mathbb{W}}(i, :), \mathbf{r}, \mathbf{b})$ // Least-squares via rank-one update, see Algorithm 8 in Ref. [76]
  **end if**
  $\varepsilon \leftarrow \varepsilon \cup i$; $y \leftarrow y \setminus \{i\}$ // Move index $i$ from $y$ to $\varepsilon$
  $n \leftarrow$ Indexes such that $\boldsymbol{\omega}(n) < 0$ // Identify negative weights
  $y \leftarrow y \cup \{\varepsilon(n)\}$; $\varepsilon \leftarrow \varepsilon \setminus \{\varepsilon(n)\}$ // Remove indexes with negative weights
  $\mathsf{H} \leftarrow \text{UPHERM}(\mathsf{H}, n)$ // Update inverse Hermitian Matrix (see Algorithm 9 in Ref. [76])
  $\boldsymbol{\omega} \leftarrow \mathsf{H}\widetilde{\mathbb{W}}(\varepsilon, :)^\top \mathbf{b}$ // Recalculate weights
  **if** successful iteration **then**
    $\phi \leftarrow 0$
  **else**
    $\phi \leftarrow \phi + 1$
  **end if**
  $\mathbf{r} \leftarrow \mathbf{b} - \widetilde{\mathbb{W}}(\varepsilon, :)^\top \boldsymbol{\omega}$ // Update the residual
**end while**

## B.4 Point selection in GNAT hyper-reduction

**Algorithm 5** Greedy algorithm for selecting sample nodes from a given mesh (reproduced from [80])

**Input:** $\widetilde{\mathbb{U}}_R$ (POD basis for residuals), $\widetilde{\mathbb{U}}_J$ (POD basis for Jacobians), target sample nodes $m_{\text{node}}$, initial sample-node set $\mathcal{N}$ (see Remark 2), and number of working columns $n_c \leq \min(m_R, m_J, \nu m_{\text{node}})$, where $\nu$ denotes the number of unknowns at a node (e.g., $\nu = 5$ for three-dimensional compressible flows without a turbulence model).

**Output:** sample-node set $\mathcal{N}$

Compute the additional number of nodes to sample: $n_a = m_{\text{node}} - |\mathcal{N}|$
Initialize counter for the number of working basis vectors used: $n_b \leftarrow 0$
Set the number of greedy iterations to perform: $n_{\text{it}} = \min(n_c, n_a)$
Compute the maximum number of right-hand sides in the least-squares problems: $n_{\text{RHS}} = \text{ceil}(n_c/n_a)$
Compute the minimum number of working basis vectors per iteration: $n_{ci,\min} = \text{floor}(n_c/n_{\text{it}})$
Compute the minimum number of sample nodes to add per iteration: $n_{ai,\min} = \text{floor}(n_a n_{\text{RHS}}/n_c)$
**for** $i = 1, \ldots, n_{\text{it}}$ **do** {greedy iteration loop}
  Compute the number of working basis vectors for this iteration: $n_{ci} \leftarrow n_{ci,\min}$;
if $(i \leq n_c \mod n_{\text{it}})$, then $n_{ci} \leftarrow n_{ci} + 1$
  Compute the number of sample nodes to add during this iteration: $n_{ai} \leftarrow n_{ai,\min}$;
if $(n_{\text{RHS}} = 1)$ and $(i \leq n_a \mod n_c)$, then $n_{ai} \leftarrow n_{ai} + 1$
  **if** $i = 1$ **then**
    $[\mathbf{r}^1 \ldots \mathbf{r}^{n_c}] \leftarrow [\widetilde{\mathbb{U}}_R^1 \ldots \widetilde{\mathbb{U}}_R^{n_c}]$
    $[\mathsf{J}^1 \ldots \mathsf{J}^{n_c}] \leftarrow [\widetilde{\mathbb{U}}_J^1 \ldots \widetilde{\mathbb{U}}_J^{n_c}]$
  **else**
    **for** $q = 1, \ldots, n_{ci}$ **do** {basis vector loop}
      $\mathbf{r}^q \leftarrow \widetilde{\mathbb{U}}_R^{n_b+q} - [\widetilde{\mathbb{U}}_R^1 \ldots \widetilde{\mathbb{U}}_R^{n_b}]\alpha$, with $\alpha = \arg\min_{\gamma \in \mathcal{R}^{n_b}} \|[Z^\top \widetilde{\mathbb{U}}_R^1 \ldots Z^\top \widetilde{\mathbb{U}}_R^{n_b}]\gamma - Z^\top \widetilde{\mathbb{U}}_R^{n_b+q}\|$
      $\mathsf{J}^q \leftarrow \widetilde{\mathbb{U}}_J^{n_b+q} - [\widetilde{\mathbb{U}}_J^1 \ldots \widetilde{\mathbb{U}}_J^{n_b}]\beta$, with $\beta = \arg\min_{\gamma \in \mathcal{R}^{n_b}} \|[Z^\top \widetilde{\mathbb{U}}_J^1 \ldots Z^\top \widetilde{\mathbb{U}}_J^{n_b}]\gamma - Z^\top \widetilde{\mathbb{U}}_J^{n_b+q}\|$
    **end for**
  **end if**
  **for** $j = 1, \ldots, n_{ai}$ **do** {sample node loop}
    Choose node with largest average error: $n \leftarrow \arg\max_{l \notin \mathcal{N}} \sum_{q=1}^{n_{ci}} \left( \sum_{i \in \delta(l)} \left( (\mathbf{r}_i^q)^2 + (\mathsf{J}_i^q)^2 \right) \right)$,
    where $\delta(l)$ denotes the degrees of freedom associated with node $l$.
    $\mathcal{N} \leftarrow \mathcal{N} \cup \{n\}$
  **end for**
  $n_b \leftarrow n_b + n_{ci}$
**end for**